Representation Learning @ Scale

DIVERSITY, INTERPRETABILITY, SCALABILITY

Manzil Zaheer

CMU-ML-2018-1234

July 2018

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Barnabas Poczos, Co-Chair
Ruslan Salakhutdinov, Co-Chair
Alexander J Smola
Andrew McCallum
Geoffrey J Gordon

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (PhD).

Support: Thank funding agencies
Representation Learning @ Scale
Diversity, Interpretability, Scalability

Committee:
1. Barnabas Poczos
2. Ruslan Salakhutdinov
3. Alexander Smola
4. Andrew McCallum
5. Geoffrey Gordon

Date of submission: July 2018
Date of defense: July 2018

Please cite this document as:
Manzil Zaheer, Representation Learning @ Scale, CMU PhD Thesis, 2018
URN: CMU-ML-18-1234
URL: http://reports-archive.adm.cs.cmu.edu/anon/ml2018/1234

This document is provided by MLD,
SCS Technical Report Collection,
Carnegie Mellon University
http://reports-archive.adm.cs.cmu.edu
reports@cs.cmu.edu

The publication is under the Creative Commons license:
Attribution – No Commercial Use – No Editing 2.0 USA
http://creativecommons.org/licenses/by-nc-nd/2.0/
To my parents
ABSTRACT

Machine learning techniques are reaching or exceeding human level performances in tasks involving simple data like image classification, translation, and text-to-speech. The success of these machine learning algorithms is attributed to highly versatile representations learnt from data using deep networks or intricately designed Bayesian models. Representation learning has also provided hints in neuroscience, e.g. understanding how humans might categorize objects. Despite these instances of success, progress has been limited to simple data-types so far.

Most real-world data come in all shapes and sizes, not just as images or text, but also as point clouds, sets, graphs, compressed or even heterogeneous combinations thereof. In this thesis, we develop representation learning algorithms for such complex data types by leveraging structure and establishing new mathematical properties. Representations learned in this fashion were applied on diverse domains and found to be competitive with task-specific state-of-the-art methods.

Having representations is not enough in various applications - its interpretability is as crucial as its accuracy. Deep models often yield better accuracy but require a large number of parameters, often in contrast to the simplicity of the underlying data, rendering it uninterpretable. This is highly undesirable in tasks like user modeling. In this thesis, we show that by leveraging structure by incorporating domain knowledge in the form of Bayesian components on top of deep models, we learn sparser representations with discrete components that are more amenable to human interpretation. Our experimental evaluations show that the proposed techniques compare favorably with several state-of-the-art baselines.

Finally, inferring interpretable representations from large-scale data is desirable, but often hindered by a mismatch between computational resources and statistical models. In this thesis, we bridge this gap by again leveraging structure, albeit of a different kind. Our solutions are based on a combination of modern computational techniques/data structures on one side and modified statistical inference algorithms on the other which exploit topological properties of the training objective. This introduces new ways to parallelize, reduce look-ups, handle variable state space size, and escape saddle points. On latent variable models, like latent Dirichlet allocation (LDA), we observe significant gains in performance.

To summarize, in this thesis, we advance the three major aspects of representation learning — diversity: being able to handle different types of data, interpretability: being accessible to and understandable by humans, and scalability: being able to process massive datasets in a reasonable time and budget — all by leveraging some form of structure.
ACKNOWLEDGMENTS

Five years of Ph.D. studies at Carnegie Mellon University (CMU) were a fantastic experience for me, to say the least. I was extremely fortunate to be advised by a number of professors at CMU, starting from Xin Li to Alexander Smola to Ruslan Salakhutdinov and Barnabas Poczos. I learned tremendously from each of them. Alexander Smola has been an exceptional mentor. His sheer brilliance and multifaceted expertise of machine learning and science (and life in general) have been an excellent source of inspiration for me. Barnabas Poczos is a brilliant researcher and an excellent mentor who taught me that patience and persistence is the key to successful research. Ruslan Salakhutdinov is a well-known researcher with an extremely friendly and calm demeanor which spurred me to ask many questions, even silly ones, without hesitation. His response always were highly creative and filled with his incredibly sharp insights.

I am grateful that all of my advisors gave me a lot of freedom and leeway in choosing my research problems and how to pursue them. Although perhaps sometimes I was a bit too stubborn and should have listened more carefully to my advisors. My thesis committee gave me also excellent feedback after my thesis proposal. I am very proud of my committee.

My internships at Amazon, Google, Intel, and Oracle, introduced me to many amazing people, which had a significant impact on my research and in learning to become a researcher. Under Jean-Baptiste Tristan’s mentorship in Oracle, I learned a lot about how to be a proper researcher. He asked the right questions and pushed me to concentrate on the essential problems.

I have been extremely fortunate to collaborate with a wonderful set of colleagues: Amr Ahmed, Francis Bach, Rajarshi Das, Shehzaad Dhuliawala, Bhuwan Dhingra, Avinava Dubey, Chris Dyer, Chenjie Gu, Abhishek Khetan, Satwik Kottur, Jay-Yoon Lee, Golan Levin, Chun-Liang Li, Andrew McCallum, Junier Oliva, Siyam Ravanbakhsh, Sashank Reddi, Shashank Singh, Suvrit Sra, Hsiao-Yu Fish Tung, Fa Wang, Michael Wick, Chao-Yuan Wu, Eric Xing, Yang Zhang, and Xun Zheng. Many chapters of this thesis are the culmination of these collaborations. Without their expertise in their respective field, I would have been totally lost.

I am very grateful and hugely indebted to all my teachers at CMU – Christos Faloutsos, Geoff Gordon, Venkat Guruswami, Gary Miller, Ryan Tibshirani, Larry Wasserman, – for sharing their valuable insights and research expertise with me. Also, thanks to the department staff members, especially Diane Stidle, Sandy Winkler, and Amy Protos, for their invaluable support throughout my time at CMU.

I was lucky enough to have a wonderful set of friends at CMU. I would like to thank my officemates Alnur Ali, Benjamin Cowley, Jing Xiang, and department colleagues Emmanouil Antonios Platanios, Leqi Liu, Ritesh Noothigattu, Yifei Ma, William Neiswanger, Otilia Stretcu for countless exciting discussions, helping me to take gradient steps for improvement. Also, I would like to thank Anita, Avinava, Chun-Liang, Guru, Jay, Jing, Rajarshi, and Sashank for being such awesome friends. Without their company, it would not have been so much fun.

Also, I would like to acknowledge the service of shuttle and escort drivers, especially Chuck and Han, who allowed me to work on my schedule – the times I am productive most without worrying about safety while commuting between university and home.

Finally and most importantly, I am deeply indebted to my loving mom Srinvanti Pal Zaheer and dad Hasan Zaheer. I am lucky to have a such a great family who always supported me.


Contents

1 Introduction
1.1 Aspects of Representation Learning ................................. 2
1.2 Our Approach .......................................................... 2
1.3 Overview of Thesis & Our Results .................................. 3
1.4 Bibliographic Notes ...................................................... 7
1.5 How to read this Thesis? ............................................... 7

I Diversity by Handling Complex Data

2 Set Data........................................................................... 11
2.1 Introduction ................................................................. 11
2.2 What are Invariance and Equivariance? .................. 12
  2.2.1 Example: CNN as Translation Invariance .......... 13
2.3 Permutation Invariance and Equivariance ............... 15
  2.3.1 Problem Definition ................................................. 15
  2.3.2 Countable Case ...................................................... 15
  2.3.3 Uncountable Case .................................................. 16
  2.3.4 Specialized Layer ................................................. 21
  2.3.5 Related Results ....................................................... 22
2.4 Deep Sets .................................................................. 23
  2.4.1 Architecture ............................................................ 23
  2.4.2 Other Related Works .............................................. 24
2.5 Experiments on Discriminative Tasks ....................... 25
  2.5.1 Estimate Population Statistics .............................. 25
  2.5.2 Sum of Digits .......................................................... 27
  2.5.3 Point Cloud Classification ..................................... 27
  2.5.4 Set Anomaly Detection ........................................... 29
2.6 Set Expansion ............................................................... 30
  2.6.1 Bayes Set ............................................................... 31
  2.6.2 Using DeepSets ....................................................... 33
2.7 Experiments on Set Expansion ................................. 33
  2.7.1 Text Concept Set Retrieval ................................. 33
  2.7.2 Image Tagging ......................................................... 35
2.8 Set Generation ............................................................... 37
  2.8.1 Proposed Method .................................................... 39
  2.8.2 Tighter Solutions via Sandwiching ...................... 41
  2.8.3 Simplified Theoretical Justification .................... 42
2.9 Experiments on Set Generation ................................. 44
  2.9.1 Synthetic Datasets ................................................ 45
  2.9.2 ModelNet40 .......................................................... 46
2.10 Summary ....................................................................... 50

3 Compressed Data .......................................................... 53
3.1 Introduction ................................................................. 53
3.2 Background ................................................................. 55
  3.2.1 Action Recognition ................................................ 55
3.2.2 Video Compression .................................................. 55
3.3 Modeling Compressed Representations ................................. 57
3.4 Experiments ............................................................... 60
  3.4.1 Ablation Study .................................................... 60
  3.4.2 Speed and Efficiency .............................................. 61
  3.4.3 Accuracy .......................................................... 62
3.5 Conclusion ............................................................... 65

4 HETEROGENEOUS DATA ............................................. 67
  4.1 Introduction ........................................................... 67
  4.2 Background ........................................................... 69
  4.3 Model ................................................................. 69
  4.4 Experiments ........................................................... 70
    4.4.1 Evaluation Dataset .............................................. 70
    4.4.2 Models .......................................................... 71
    4.4.3 Implementation Details ....................................... 72
    4.4.4 Results and Discussions ..................................... 72
  4.5 Related Work ........................................................ 73
  4.6 Conclusions .......................................................... 74

II INTERPRETABILITY BY INCORPORATING DOMAIN KNOWLEDGE 

5 WORD EMBEDDINGS + LDA ........................................... 77
  5.1 Introduction ........................................................... 77
  5.2 Background ........................................................... 78
    5.2.1 Vector Space Semantics ....................................... 78
    5.2.2 Latent Dirichlet Allocation (LDA) .............................. 78
  5.3 Gaussian LDA .......................................................... 79
  5.4 Posterior Inference ................................................... 80
    5.4.1 Faster sampling using Cholesky decomposition of covariance matrix ..................................................... 81
    5.4.2 Further reduction of sampling complexity using Alias Sampling ..................................................... 82
  5.5 Experiments ........................................................... 83
    5.5.1 Topic Coherence .................................................. 83
    5.5.2 Performance on document containing new words .......... 84
  5.6 Conclusion and Future Work ........................................ 87
  5.7 Derivation of Rank 1 update of the covariance matrix .......... 88
  5.8 Alias Sampling for LDA with Gaussian Emission Model ........ 89
    5.8.1 Alias Method .................................................... 90
    5.8.2 Gaussian LDA with Alias ...................................... 91

6 LSTM + LDA ............................................................... 93
  6.1 Introduction ........................................................... 93
  6.2 Background ........................................................... 94
    6.2.1 User/Language Modeling ...................................... 94
    6.2.2 Long Short-Term Memories .................................... 94
  6.3 Latent LSTM Allocation ............................................. 95
    6.3.1 Generative model ................................................ 96
    6.3.2 Inference ........................................................ 97
    6.3.3 Adding Context ................................................ 99
  6.4 Experimental Results ............................................... 101
    6.4.1 Setup ........................................................... 101
    6.4.2 Language modeling ............................................. 102
10.3 Generic Framework .................................................. 170
10.3.1 Convergence Analysis ........................................ 172
10.4 Concrete Instantiations ............................................. 174
  10.4.1 Hessian descent .............................................. 177
  10.4.2 Cubic Descent .................................................. 179
  10.4.3 Practical Considerations .................................... 180
10.5 Experiments ............................................................ 181
  10.5.1 Synthetic Problem ............................................ 181
  10.5.2 Deep Networks ............................................... 182
10.6 Discussion ............................................................. 184

PUBLICATIONS .............................................................. 187

BIBLIOGRAPHY ............................................................... 189
# List of Figures

| Figure 1.1 | Important challenges in machine learning and contributions of this thesis. | 3 |
| Figure 1.2 | Caption | 4 |
| Figure 2.1 | Outline of the proof strategy for Theorem 2.1. The proof consists of two parts. First, we desire to show that we can find unique embeddings for each possible input, *i.e.* we show that there exists a homeomorphism $E$ of the form $E(X) = \sum_{x \in X} \phi(x)$ between original domain and some higher dimensional space $Z$. The second part of the proof consists of showing we can map the embedding to desired target value, *i.e.* to show the existence of the continuous map $\rho$ between $Z$ and original target space such that $f(X) = \rho(\sum_{x \in X} \phi(x))$. | 19 |
| Figure 2.2 | Architecture of DeepSets: Invariant | 23 |
| Figure 2.3 | Illustration of permutation equivariant layer. Same color indicates weight sharing. | 24 |
| Figure 2.4 | Architecture of DeepSets: Equivariant | 24 |
| Figure 2.5 | Using multiple permutation equivariant layers. Since permutation equivariance compose we can stack multiple such layers | 25 |
| Figure 2.6 | Population statistic estimation: Top set of figures, show prediction of DeepSets vs SDM for $N = 2^{10}$ case. Bottom set of figures, depict the mean squared error behavior as number of sets is increased. SDM has lower error for small $N$ and DeepSets requires more data to reach similar accuracy. But for high dimensional problems deep sets easily *scales* to large number of examples and produces much lower estimation error. Note that the $N \times N$ matrix inversion in SDM makes it prohibitively expensive for $N > 2^{14} = 16384$. | 26 |
| Figure 2.7 | Accuracy of digit summation with text (*left*) and image (*right*) inputs. All approaches are trained on tasks of length 10 at most, tested on examples of length up to 100. We see that DeepSets generalizes better. | 27 |
| Figure 2.8 | Each box is the particle-cloud maximizing the activation of a unit at the firsts (*top*) and second (*bottom*) permutation-equivariant layers of our model. Two images of the same column are two different views of the same point-cloud. | 29 |
| Figure 2.9 | Each row shows a set, constructed from CelebA dataset, such that all set members except for an outlier, share at least two attributes (on the right). The outlier is identified with a red frame. The model is trained by observing examples of sets and their anomalous members, *without access to the attributes*. The probability assigned to each member by the outlier detection network is visualized using a red bar at the bottom of each image. The probabilities in each row sum to one. | 30 |
Figure 2.10 Each row shows a set, constructed from CelebA dataset, such that all set members except for an outlier, share at least two attributes (on the right). The outlier is identified with a red frame. The model is trained by observing examples of sets and their anomalous members, without access to the attributes. The probability assigned to each member by the outlier detection network is visualized using a red bar at the bottom of each image. .............................................. 31

Figure 2.11 Examples from our LDA-1k datasets. Notice that each of these are latent topics of LDA and hence are semantically similar. .................. 35

Figure 2.12 Qualitative examples of image tagging using DeepSets. Top row: Positive examples where most of the retrieved tags are present in the ground truth (brown) or are relevant but not present in the ground truth (green). Bottom row: Few failure cases with irrelevant/wrong tags (red). From left to right, (i) Confusion between snowboarding and skiing, (ii) Confusion between back of laptop and refrigerator due to which other tags are kitchen-related, (iii) Hallucination of airplane due to similar shape of surfboard. .............................................. 38

Figure 2.14 Overview .............................................. 39

Figure 2.13 Unlike in case of images, the marginal distribution \( p(x_i) = \int p(x_i|\theta)p(\theta)d\theta \) is not useful. We have to learn the joint distribution \( p(X, \theta) \). An illustrative parallel can be drawn between pixels of an image and point cloud — marginal distribution of pixels is quite uninformative, one has to consider the joint distribution of all pixels in one image (not across different images). .............................................. 39

Figure 2.15 Faces generated from a GAN .............................................. 41

Figure 2.16 The reconstructed center and radius distributions. (a) (top) the true center distribution and (bottom) one example of the 2D circle point cloud. (b-d) are the reconstructed center and radius distributions of different algorithms. .............................................. 46

Figure 2.17 The reconstruction on test objects from seen categories. For each object, from left to right is training data, AAE, and PC-GAN. PC-GAN is better in capturing fine details like wheels of aeroplane or proper chair legs. ... 47

Figure 2.18 Sample mesh from ModelNet40 .............................................. 48

Figure 2.19 Randomly sampled objects and corresponding point cloud from the hierarchical sampling. Even if there are some defects, the objects are smooth, symmetric and structured. It suggests PC-GAN captures inherent patterns and learns basic building blocks of objects. .............................................. 49

Figure 2.20 Interpolating between a table and a chair point clouds, using our latent space representation. .............................................. 49

Figure 2.21 Interpolating between rotation of an aeroplane, using our latent space representation. .............................................. 49

Figure 2.22 The reconstructed objects from unseen categories. In each plot, LHS is true data while RHS is PC-GAN. PC-GAN generalizes well as it can match patterns and symmetries from categories seen in the past to new unseen categories. .............................................. 50

Figure 3.1 Traditional architectures first decode the video and then feed it into a network. Instead, we propose to use the compressed video directly. ... 54
Figure 3.2 We trace all motion vectors back to the reference I-frame and accumulate the residual. Now each P-frame depends only on the I-frame but not other P-frames. .......................................................... 56

Figure 3.3 Original motion vectors and residuals describe only the change between two frames. Usually the signal to noise ratio is very low and hard to model. The accumulated motion vectors and residuals consider longer term difference and show clearer patterns. Assume I-frame is at $t = 0$. Motion vectors are plotted in HSV space, where the H channel encodes the direction of motion, and the S channel shows the amplitude. For residuals we plot the absolute values in RGB space. Best viewed in color. 57

Figure 3.4 We decouple the dependencies between P-frames so that they can be processed in parallel. .......................................................... 58

Figure 3.5 Decoupled model. All networks can be trained independently. Models are shared across P-frames. .......................................................... 59

Figure 3.6 Speed and accuracy on UCF-101 Soomro, Zamir, and Shah, 2012, compared to Two-stream Network Simonyan and Zisserman, 2014, Res3D Tran, Ray, et al., 2017, and ResNet-152 He et al., 2016b trained using RGB frames. Node size denotes the input data size. Training on compressed videos is both accurate and efficient, while requiring a minimal amount of data. .......................................................... 63

Figure 3.7 Two videos of “Jumping Jack” from UCF-101 in their RGB, motion vector, and residual representations plotted in t-SNE Maaten and G. Hinton, 2008 space. The curves show video trajectories. While in the RGB space the two videos are clearly separated, in the motion vector and residual space they overlap. This suggests that with compressed signals, videos of the same action can share statistical strength better. Also note that the RGB images contain no motion information, and thus the two ways of the trajectories overlap. This is in contrast to the circular patterns in the trajectories of motion vectors. Best viewed on screen. .......................................................... 66

Figure 4.1 Existing approaches utilizes only one type of information source. However combination of the heterogeneous information source is powerful. We want to develop a question answering system that can leverage both the information source. .......................................................... 67

Figure 4.2 Memory network attending the facts in the universal schema (matrix on the left). The color gradients denote the attention weight on each fact. ................................. 68

Figure 4.3 Performance on varying the number of available KB facts during test time. UniSCHEMA model consistently outperforms ONLYKB .................. 74

Figure 5.1 Plot comparing average log-likelihood vs time (in sec) achieved after applying each trick on the NIPS dataset. The shapes on each curve denote end of each iteration. ......................................................... 83

Figure 5.2 The first two principal components for the word embeddings of the top words of topics shown in Table ?? have been visualized. Each blob represents a word color coded according to its topic in the Table ?? .......................... 86

Figure 6.1 LLA has better perplexity (lower is better) than LDA but much fewer parameters than LSTMs, as shown in a language modeling task on Wikipedia. 93

Figure 6.2 Graphical models for LDA and variants of proposed latent LSTM Allocation (LLA). In a slight abuse of plate notation, we do not imply exchangibility by the dashed plate diagram and -1 on an edge means the dependence is from one time step back. .......................................................... 96
<table>
<thead>
<tr>
<th>Figure</th>
<th>Caption</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>Different parameters employed by LDA, LSTM and LLA. K is number of topics, V is vocabulary size, and H is the dimension of LSTM state. (a) Topics of LDA, (b) word embedding of LSTM (c) factored topic embedding of LLA.</td>
</tr>
<tr>
<td>6.4</td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>The effect of the number of topics over the performance of LDA and LLA.</td>
</tr>
<tr>
<td>6.6</td>
<td>Topic embedding discovered using LLA.</td>
</tr>
<tr>
<td>6.8</td>
<td>Convergence speed for various models. The x-axis represents time in second while the Y-axis gives the perplexity over the test dataset.</td>
</tr>
<tr>
<td>7.1</td>
<td>Efficient (re)use of buffers</td>
</tr>
<tr>
<td>7.2</td>
<td>Evolution of log likelihood on Wikipedia and Pubmed over number of iterations and time.</td>
</tr>
<tr>
<td>8.1</td>
<td>Canopy is much faster yet as accurate as other methods like EM or ESCA.ZAHEERETAL16. The bar graph shows time per iteration while line plots the likelihood on held-out test set. Results shown are for inference of a Gaussian mixture model with 32 million points having 4096 clusters at 1024 dimensions.</td>
</tr>
<tr>
<td>8.2</td>
<td>Hierarchical partitioning over both data observations and clusters. Once we sample clusters at a coarser level, we descend the hierarchy and sample at a finer level, until we have few number of points per cluster. We then use Section 8.3.2.1 for rejection sampler.</td>
</tr>
<tr>
<td>8.3</td>
<td>Showing scalability of per-iteration runtime of different algorithms with increasing dataset size. From Fig. 8.3a, 8.3b, and 8.3c we see that our approaches take orders of magnitude less time compared to the traditional EM and ESCA methods, while varying the number of points and clusters respectively. Note that we trade off memory for speed as seen from Figure 8.3d. For instance, with (n, m, d) = (32mil, 4096, 1024), we see that there is a speed-up of 150× for a mere 2× memory overhead.</td>
</tr>
<tr>
<td>8.4</td>
<td>Plots of cluster purity and loglikelihood of ESCA, Canopy I, and Canopy II on benchmark real datasets –MNIST8m and CIFAR-100. All three methods have roughly same performance on cluster purity. See Section 8.5.2 for more details.</td>
</tr>
<tr>
<td>8.5</td>
<td>Illustration of concepts captured by clustering images in the ResNet HeZhaRenSun15; HeZhaRenSun16 feature space. We randomly pick three clusters and show four closest images (one in each row), possibly denoting the semantic concepts of ‘crowd’, ‘ocean rock scenery’ and ‘horse mounted police’. Our clustering discovers new concepts beyond the Resnet supervised categories (does not include ‘crowd’).</td>
</tr>
<tr>
<td>9.1</td>
<td>Framework</td>
</tr>
<tr>
<td>9.2</td>
<td>Typical posterior distribution of the HDP using a sampling table configuration representation. Without loss of generality we order local and global terms into a joint list. What is displayed are unnormalized probabilities.</td>
</tr>
<tr>
<td>9.3</td>
<td>Log structured alias table</td>
</tr>
</tbody>
</table>
Samples produced per second by different algorithms across datasets of varying size. In case of LDA over big datasets (NY Times, PubMed, Wikipedia), Light LDA is the fastest computationally. Similarly for HDPs, Alias HDP is the fastest computationally for all datasets. However, looking at only computational speed paints an incomplete picture. The computationally fastest may not converge as fast overall.

SingleThreaded HDP on small datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number.

Multi Threaded HDP on small datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number.

Single Threaded HDP on medium datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number. While it was not observable previously due to small datasets, the difference between each method is clearer. In the comparison between time, Alias HDP was the fastest.

Multi Threaded HDP on medium datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number. While it was not observable previously due to small datasets, the difference between each method is clearer. In the comparison between time, SDA HDP was the fastest.

Multi Threaded HDP on large datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number. While it was not observable previously due to small datasets, the difference between each method is clearer. In the comparison between time, Alias HDP was the fastest.

Comparison of perplexity between LDA and HDP on a small, medium, and large dataset.

First order methods like GD can potentially get stuck at saddle points. Second-order methods can escape it in very few iterations (as observed in the left plot) but at the cost of expensive Hessian based iterations (see time plot to the right). The proposed framework, which is a novel mix of the two strategies, can escape saddle points faster in time by carefully trading off computation and iteration complexity.

Comparison of various methods on a synthetic problem. Our mix framework successfully escapes saddle point and uses relatively few ISO calls in comparison to CUBIC DESCENT.

Comparison of various methods on a Deep Autoencoder on CURVES (top) and MNIST (bottom). Our mix approach converges faster than the baseline methods and uses relatively few ISO calls in comparison to APPROX CUBIC DESCENT.

Zoomed in version of plots with respect to time. Here we show progress from time=10s onwards. This better exhibits the relative differences between the methods, and is illustrative of the advantage of our method.
List of Tables

Table 2.1 Classification accuracy and the representation-size used by different methods on the ModelNet40. ................................. 28
Table 2.2 Results on Text Concept Set Retrieval on LDA-1k, LDA-3k, and LDA-5k. Our DeepSets model outperforms other methods on LDA-3k and LDA-5k. However, all neural network based methods have inferior performance to w2v-Near baseline on LDA-1k, possibly due to small data size. Higher the better for recall@k and mean reciprocal rank (MRR). Lower the better for median rank (Med.) ................................. 34
Table 2.3 Results of image tagging on ESPgame and IAPRTC-12.5 datasets. Performance of our DeepSets approach is roughly similar to the best competing approaches, except for precision. Refer text for more details. Higher the better for all metrics – precision (P), recall (R), f1 score (F1), and number of non-zero recall tags (N+). ................................. 36
Table 2.4 Results on COCO-Tag dataset. Clearly, DeepSets outperforms other baselines significantly. Higher the better for recall@K and mean reciprocal rank (MRR). Lower the better for median rank (Med.) ................................. 37
Table 2.5 The quantitative results of different models trained on different subsets of ModelNet40. MultiNet10 is a subset containing 10 classes of objects, while MultiNet40 is a full training set. ................................. 47
Table 2.6 Classification accuracy results. ................................. 49
Table 3.2 Action recognition accuracy on UFC-101 Soomro, Zamir, and Shah, 2012 (Split 1). The two rows show the performance of the models trained using the original motion vectors/residuals and the models using the accumulated ones respectively. I: I-frame RGB image. M: motion vectors. R: residuals. ................................. 61
Table 3.3 Network computation complexity and accuracy of each method. Our method is 4.6x more efficient than state-of-the-art 3D CNN, while being much more accurate. ................................. 62
Table 3.4 Speed (ms) per frame of each method. Our method is fast in both preprocessing and CNN computation. The preprocessing speed of our method is presented for both single-thread / multi-thread settings. ................................. 62
Table 3.5 Action recognition accuracy on UFC-101 Soomro, Zamir, and Shah, 2012 and HMDB-51 Kuehne et al., 2011. Combining our model with a temporal-stream network achieves a new state-of-the-art. ................................. 63
Table 3.6  Action recognition accuracy on UCF-101 Soomro, Zamir, and Shah, 2012 and HMDB-51 Kuehne et al., 2011. The upper part of the table lists real-time methods that do not require computing optical flow. The lower part of the table lists methods that requires computing optical flow. Our method outperforms all baselines in both settings. Asterisk indicates results evaluated only on split 1 of the datasets. They are listed purely for reference. ......................................................... 64
Table 3.7  Action recognition accuracy on Charades Gunnar A Sigurdsson et al., 2016. Without using additional annotations as Sigurdsson et al. Gunnar A. Sigurdsson et al., 2017 or complicated feature aggregation, our method achieves the best performance. ......................................................... 65
Table 4.1  QA results on SPADES. ......................................................... 65
Table 4.2  A few questions on which ONLYKB fails to answer but UniSCHEMA succeeds. ......................................................... 71
Table 4.3  Detailed results on SPADES. ......................................................... 72
Table 5.1  Top words of some topics from Gaussian-LDA and multinomial LDA on 20-newsgroups for K = 50. Words in Gaussian LDA are ranked based on density assigned to them by the posterior predictive distribution. The last row for each method indicates the PMI score (w.r.t. Wikipedia co-occurrence) of the topic’s fifteen highest ranked words. ......................................................... 85
Table 5.2  Accuracy of our model and infovec on the synthetic datasets. In Gaussian LDA fix, the topic distributions learnt during training were fixed; G-LDA(1, 100, 1932) is the online implementation of our model where the documents comes in minibatches. The number in parenthesis denote the size of the batch. The full size of the test corpus is 1932. .................. 87
Table 5.3  This table shows the Average $L_1$ Deviation, Average $L_2$ Deviation, Average $L_\infty$ Deviation for the difference of the topic distribution of the actual document and the synthetic document on the NIPS corpus. Compared to infovec, G-LDA achieves a lower deviation of topic distribution inferred on the synthetic documents with respect to actual document. The full size of the test corpus is 174. ................................. 87
Table 6.1  Enumerating different input and output representation, leading to variants of LLA models. Note we exclude char level output transformation due to its inapplicability to user modeling. ......................................................... 101
Table 6.2  Cleaner topics produced by LLA vs LDA ......................................................... 105
Table 6.3  Factored topics produced by LLA vs LDA ......................................................... 106
Table 6.4  Advantage in terms of perplexity for joint learning ......................................................... 106
Table 7.1  Examples of some popular models to which ESCA is applicable. ......................................................... 111
Table 9.1  A quick reference for notations ......................................................... 141
Table 9.2  A quick reference for extra notations used in describing hierarchical models in addition to Table 9.1 ......................................................... 142
Table 9.3  Experimental datasets and their statistics. V denotes vocabulary size, L denotes the number of training tokens, D denotes the number of documents, L/V indicates the average number of occurrences of a word, L/D indicates the average length of a document. ......................................................... 158
Introduction

The digitally connected world of today involves potentially valuable data growing at a tremendous pace like text, images, and speech. The value generation lies in transforming unstructured data into a usable information or representation, which is well organized, searchable and understandable. This has been a major enabler for personalized recommendation (of articles to read, videos to watch, locations to visit, restaurants to dine at), fraud detection, medical image analysis, face recognition, etc. Machine learning and intelligent systems are used to carry out the task listed above and have started to become an indispensable part of our modern society.

The current machine learning paradigms mostly focus on individual objects that have simple representations. For example in the tasks of classification, regression, and clustering, an object of interest is often described by a “feature vector”, and abstracted as a point in certain metric space. The objective of learning is to estimate functions that map these points to target variables such as the class labels or cluster memberships, with the goal of achieving both empirical accuracies on the training data as well as the generalization power on unseen data. This “one point per object” abstraction has led to very concise representations, elegant mathematical theories, and very successful algorithms. The performance of machine learning on above mentioned well defined tasks with simple data have reached or exceeded human levels like image classification (He et al., 2016a) or speech recognition (W. Xiong et al., 2016). The success of the methods heavily depends on the choice of data representation (or features) used (Y. Bengio, Courville, and Vincent, 2013), for which deep learning has been a boon.

However, the data comes in all shapes and sizes, i.e. ranging from text to images & videos to point clouds to graphs to detailed user activity logs and shopping histories in sizes ranging from a few megabytes to hundreds of terabytes and combinations thereof. For instance, in the field of language modeling and text processing, an article can be considered as a group of paragraph or sections. In computer vision, the recently gaining 3D imaging consists of a group of coordinates. In recommendation systems, a user is mainly described by the heterogeneous data of products bought, videos watched, articles read. We call these kinds of data that are organized by groups or heterogeneous combinations as the complex data.

This thesis is centered around representation learning, i.e., learning representations of the above mentioned complex data that make it easier to extract useful information when building classifiers or other predictors for human and/or machine consumption. Traditionally, machine learning has focuses on individual objects, each of which is described by a feature vector and studied as a point in some metric space. When approaching more complex data, researchers often reduce the groups into vectors to which traditional methods can be applied. We, on the other hand, will try to develop machine learning methods that learn from them directly. In order to build such methods, one needs to leverage structure present in the data. Furthermore, by utilization of structure one can build faster inference techniques as well more interpretable models.

Thesis statement: Leveraging structure can not only lead to better representation learning, but also results in speeding up inference and more interpretable models.
1.1 ASPECTS OF REPRESENTATION LEARNING

There are two main categories for learning representations. First is using Bayesian models with a parsimonious set of latent random variables that describe distribution over the observed data. The representation is the posterior distribution of the underlying explanatory factors for the observed input. A good representation is also one that is useful as input to a supervised predictor. Second way of learning representations is utilizing deep learning methods, those that are formed by the composition of multiple non-linear transformations, with the goal of yielding more abstract – and ultimately more useful representations. For either categories following are the open challenges:

**diversity:** Data come in all shapes and sizes: not just as images or text, but also as point clouds, sets, graphs, compressed, or even heterogeneous mixture of these data types. In this thesis, we want to develop representation learning algorithms for such unconventional data types. Our methodology relies on leveraging the structure present in the data which involve establishing new mathematical properties. Representations learned in this fashion were applied on diverse domains and found to be competitive with task specific state-of-the-art methods.

**interpretability:** Once we have the representations, in various applications its interpretability crucial. Deep models often yield better accuracy, require a large number of parameters, often notwithstanding the simplicity of the underlying data, rendering it uninterpretable which is highly undesirable in tasks like user modeling. On the other hand, Bayesian models produce sparse discrete representations, easily amenable to human interpretation. In this thesis, we want to explore that are capable of learning mixed representations retaining best of both the worlds. Our experimental evaluations show that the proposed techniques compare favorably with several state-of-the-art baselines.

**scalability:** Finally, one would want such interpretable representations to be inferred from large-scale data, however, often there is a mismatch between our computational resources and the statistical models. In this thesis, we want to bridge this gap by solutions based on a combination of modern computational techniques/data structures on one side and modified statistical inference algorithms on the other. We introduce new ways to parallelize, reduce look-ups, handle variable state space size, and escape saddle points. On latent variable models, like latent Dirichlet allocation (LDA), we find significant gains.

1.2 OUR APPROACH

The versatility of representation learning makes it highly desirable to apply them to variety of data, make the representations learned interpretable, and scale them to large datasets of many terabytes of observations and billions of objects. We would like to take a more holistic approach considering statistical and computational aspects simultaneously. This thesis is a step in the direction of addressing these new challenges in modern ML applications. We begin by tackling the issues case by case to begin with. Eventually we wish to extend these to generality.

- **Find mathematical properties to be leveraged:** Exploiting structure or finding mathematical properties present in data has found useful in improving performance of many machine learning tasks like optimization (Hsieh et al., 2015) or classification (Goel, Knoblock, and Lerman, 2012). We carry forward the idea to case of representation learning. In particular, we characterize function on sets to design an universal deep network architecture, or how to efficiently handle Stirling numbers which occur in many posterior distributions,
or how using the additivity of sufficient statistics for exponential family can lead to a lock free and parallel inference algorithm. We show methods leveraging these structure outperform simply throwing deep networks at the problem.

- **Find correct data structures for the task:** Data structures have been designed to solve computational problems efficiently by organizing the data and making it cheap to locate and retrieve data. In this thesis we want to identify inference algorithms having specific access patterns, invariances, where efficient data structures can be adopted. We also consider modifying the inference algorithm so as to enable us in leveraging some highly efficient data structures. For example, in case of clustering only a small part of the space needs to examined carefully, thus space partitioning data structures like cover trees can come handy but requires novel modifications to the inference algorithm. Similarly to reduce communication bandwidth, approximate counters can be employed in Gibbs sampling instead of keeping track of exact counts. But approximate counters do not support decrements, thus their adoption again calls for modifications in the inference strategies. We want to find such solutions.

- **Modify inference goal/strategy:** Often the inference goal can be much stronger than needed, e.g. one always does not need the full posterior but many a times just a point estimate would suffice. As we will see the altered goal allows much better utilization of modern computational resource, which are heavily distributed and each node itself being heavily multithreaded, e.g. servers can have in excess of 64 threads and GPUs over 2048 of them, but having finite memory bandwidth. One can exploit it for tasks like clustering and topic modeling. Furthermore, the alterations can open paths to use of efficient data structures as described above.

## 1.3 Overview of Thesis & Our Results

This thesis presents a selection of my work on representation learning addressing the three important issues raised above. The content of the thesis is divided into three parts: Part I, consisting of Chapters 2 to 4, describes my work on handling complex data; Part II, organized as the next two chapters, describes my work on making the representation more interpretable while being expressive by mixing deep networks with graphical model incorporating domain knowledge; and finally Part III, composed of Chapters 7 to 11, describes my work on speeding up the inference procedure. We provide a brief overview of various chapters here and defer the exact details to the respective chapters.
Chapter 2: Set Data

We study the problem of designing models for machine learning tasks defined on sets. In contrast to traditional approach of operating on fixed dimensional vectors, we consider objective functions defined on sets that are invariant to permutations. Such problems are widespread, ranging from estimation of population statistics Poczos, Rinaldo, et al., 2013, to anomaly detection in piezometer data of embankment dams Jung et al., 2015, to cosmology Ntampaka et al., 2016; M. Ravanbakhsh et al., 2016. Our main theorem characterizes the permutation invariant functions and provides a family of functions to which any permutation invariant objective function must belong. This family of functions has a special structure which enables us to design a deep network architecture that can operate on sets and which can be deployed on a variety of scenarios including both unsupervised and supervised learning tasks. We also derive the necessary and sufficient conditions for permutation equivariance in deep models. We demonstrate the applicability of our method on population statistic estimation, point cloud classification, set expansion, and outlier detection.

Chapter 3: Compressed Data

Training robust deep video representations has proven to be much more challenging than learning deep image representations and consequently hampered tasks like video action recognition. This is in part due to the enormous size of raw video streams, the associated amount of computation required, and the high temporal redundancy. The ‘true’ and interesting signal is often drowned in too much irrelevant data. Motivated by the fact that the superfluous information can be reduced by up to two orders of magnitude with video compression techniques (like H.264, HEVC, etc.), in this work, we propose to train a deep network directly on the compressed video, devoid of redundancy, rather than the traditional highly redundant RGB stream. This representation has a higher information density and we found the training to be easier. In addition, the signals in a compressed video provide free, albeit noisy, motion information. We propose novel techniques to use them effectively. Our approach is about 4.6 times faster than a state-of-the-art 3D-CNN model, 2.7 times faster than a ResNet-152, and very easy to implement. On the task of action recognition, our approach outperforms all the other methods on the UCF-101, HMDB-51, and Charades dataset.

Chapter 4: Heterogeneous Data

Existing question answering methods infer answers either from a knowledge base or from raw text. While knowledge base (KB) methods are good at answering compositional questions,
their performance is often affected by the incompleteness of the KB. Au contraire, web text contains millions of facts that are absent in the KB, however in an unstructured form. Universal schema can support reasoning on the union of both structured KBs and unstructured text by aligning them in a common embedded space. In this paper we extend universal schema to natural language question answering, employing memory networks to attend to the large body of facts in the combination of text and KB. Our models can be trained in an end-to-end fashion on question-answer pairs. Evaluation results on Spades fill-in-the-blank question answering dataset show that exploiting universal schema for question answering is better than using either a KB or text alone. This model also outperforms the current state-of-the-art by 8.5 $F_1$ points. In this paper we look at a more practical setting, namely QA over a combination of a KB and entity-linked text, which is appropriate when an incomplete KB is available with a large text corpus. Building on recent advances in graph representation learning we propose a novel model GRAFT-Net for extracting answers from a question-specific subgraph consisting of both KB entities and text documents. We construct a suite of benchmark tasks for this problem, varying the difficulty of questions, the amount of training data, and KB completeness. We show that GRAFT-Net is competitive with the state-of-the-art when tested using either KBs or text only, and vastly outperforms existing methods in the combined setting.

Chapter 5: Word Embedding + Topic Model

Continuous space word embeddings learned from large, unstructured corpora have been shown to be effective at capturing semantic regularities in language. In this paper we replace LDA’s parameterization of “topics” as categorical distributions over opaque word types with multivariate Gaussian distributions on the embedding space. This encourages the model to group words that are \textit{a priori} known to be semantically related into topics. To perform inference, we introduce a fast collapsed Gibbs sampling algorithm based on Cholesky decompositions of covariance matrices of the posterior predictive distributions. We further derive a scalable algorithm that draws samples from stale posterior predictive distributions and corrects them with a Metropolis–Hastings step. Using vectors learned from a domain-general corpus (English Wikipedia), we report results on two document collections (20-newsgroups and NIPS). Qualitatively, Gaussian LDA infers different (but still very sensible) topics relative to standard LDA. Quantitatively, our technique outperforms existing models at dealing with OOV words in held-out documents.

Chapter 6: LSTM + Topic Model

Recurrent neural networks, such as long-short term memory (LSTM) networks, are powerful tools for modeling sequential data like user browsing history Korpusik, Sakaki, and F. C. Y.-Y. Chen, 2016; Tan, X. Xu, and Y. Liu, 2016 or natural language text Mikolov, Karafiát, et al., 2010. However, to generalize across different user types, LSTMs require a large number of parameters, notwithstanding the simplicity of the underlying dynamics, rendering it uninterpretable, which is highly undesirable in user modeling. The increase in complexity and parameters arises due to a large action space in which many of the actions have similar intent or topic. In this paper, we introduce Latent LSTM Allocation (LLA) for user modeling combining hierarchical Bayesian models with LSTMs. In LLA, each user is modeled as a sequence of actions, and the model jointly groups actions into topics and learns the temporal dynamics over the topic sequence, instead of action space directly. This leads to a model that is highly interpretable, concise, and can capture intricate dynamics. We present an efficient Stochastic EM inference algorithm for our model that scales to millions of users/documents. Our experimental evaluations show that the proposed model compares favorably with several state-of-the-art baselines.
Chapter 7: Parallelization

We propose an embarrassingly parallel, memory efficient inference algorithm for latent variable models in which the complete data likelihood is in the exponential family. The algorithm is a stochastic cellular automaton and converges to a valid maximum a posteriori fixed point. Applied to latent Dirichlet allocation we find that our algorithm is over an order of magnitude faster than the fastest current approaches. A simple C++/MPI implementation on a 20-node Amazon EC2 cluster samples at more than 1 billion tokens per second. We process 3 billion documents and achieve predictive power competitive with collapsed Gibbs sampling and variational inference.

Chapter 8: Reducing Lookups

Hierarchical Bayesian models often capture distributions over a very large number of distinct atoms. The need for these models arises when organizing huge amount of unsupervised data, for instance, features extracted using deep convnets that can be exploited to organize abundant unlabeled images. Inference for hierarchical Bayesian models in such cases can be rather non-trivial, leading to approximate approaches. In this work, we propose Canopy, a sampler based on Cover Trees that is exact, has guaranteed runtime logarithmic in the number of atoms, and is provably polynomial in the inherent dimensionality of the underlying parameter space. In other words, the algorithm is as fast as search over a hierarchical data structure. We provide theory for Canopy and demonstrate its effectiveness on both synthetic and real datasets, consisting of over 100 million images.

Chapter 9: Handling Variable State-Space Size

Latent variable models, such as topic models, have accumulated a considerable amount of interest from academia and industry for their versatility in a wide range of applications. Bayesian nonparametrics has used frequently in enabling latent variable models to adapt in accordance to the size of data. However, it is non-trivial to scale the inference procedures to deal with real life problems in industry consisting of massive datasets (1 TB+) in a reasonable time with reasonable resources. In this paper, we wish to suggest a scalable inference procedure for Bayesian Nonparametrics based on a combination of smart data structures for handling probability distributions on one side and modified statistical inference algorithms on the other. In particular, we take up the task of topic modelling with Hierarchical Dirichlet Process to demonstrate effectiveness of the proposed inference procedure. We provide experimental results for the proposed method on big datasets like PubMed and Wikipedia obtaining 10x speed-up.

Chapter 10: Escaping Saddle Points

A central challenge to using first-order methods for optimizing nonconvex problems is the presence of saddle points. First-order methods often get stuck at saddle points, greatly deteriorating their performance. Typically, to escape from saddles one has to use second-order methods. However, most works on second-order methods rely extensively on expensive Hessian-based computations, making them impractical in large-scale settings. To tackle this challenge, we introduce a generic framework that minimizes Hessian-based computations while at the same time provably converging to second-order critical points. Our framework carefully alternates between a first-order and a second-order subroutine, using the latter only close to saddle points, and yields convergence results competitive to the state-of-the-art. Empirical results suggest that our strategy also enjoys a good practical performance.
1.4 BIBLIOGRAPHIC NOTES

The research presented in this thesis is based on joint work with many co-authors, as described below.

1.5 HOW TO READ THIS THESIS?

The introduction chapter provides a brief overview of all the chapters and relationship between results of various chapters, and is a prerequisite to rest of the thesis. Each chapter is self-contained and independent of other chapters so that it can be read without any reference to rest of this thesis. Furthermore, the reader is not assumed to have detailed knowledge of machine learning; we rather try to provide the necessary knowledge, allowing the thesis to be accessible to graduate students.
Part I

DIVERSITY BY HANDLING COMPLEX DATA
We study the problem of designing models for machine learning tasks defined on sets. In contrast to traditional approaches of operating on fixed dimensional vectors, we consider objective functions defined on sets that are invariant to permutations. Such problems are widespread, ranging from estimation of population statistics (Poczos, Rinaldo, et al., 2013), to anomaly detection in piezometer data of embankment dams (Jung et al., 2015), to cosmology (Ntampaka et al., 2016; M. Ravanbakhsh et al., 2016). Our main theorem characterizes the permutation invariant functions and provides a family of functions to which any permutation invariant objective function must belong. This family of functions has a special structure which enables us to design a deep network architecture that can operate on sets and which can be deployed on all three types of learning scenarios: supervised, semi-supervised, and unsupervised learning tasks. We also derive the necessary and sufficient conditions for permutation equivariance in deep models. We demonstrate the applicability of our method on discriminative as well as generative tasks.

2.1 INTRODUCTION

A typical machine learning algorithm, like regression or classification, is designed for fixed dimensional data instances. Their extensions to handle the case when the inputs or outputs are permutation invariant sets rather than fixed dimensional vectors is not trivial and researchers have only recently started to investigate them (Muandet, Balduzzi, and Schoelkopf, 2013; Muandet, Fukumizu, et al., 2012; J. Oliva, Poczos, and J. Schneider, 2013; Szabo et al., 2016). In this paper, we present a generic framework to deal with the setting where input and possibly output instances in a machine learning task are sets.

Similar to fixed dimensional data instances, we can characterize two learning paradigms in case of sets. In supervised learning, we have an output label for a set that is invariant or equivariant to the permutation of set elements. Examples include tasks like estimation of population statistics (Poczos, Rinaldo, et al., 2013), where applications range from giga-scale cosmology (Ntampaka et al., 2016; M. Ravanbakhsh et al., 2016) to nano-scale quantum chemistry (Faber et al., 2016).

Next, there can be the semi-supervised setting, where the “set” structure needs to be learned, e.g. by leveraging the homophily/heterophily tendencies within sets. An example is the task of set expansion (a.k.a. audience expansion), where given a set of objects that are similar to each other (e.g. set of words \{lion, tiger, leopard\}), our goal is to find new objects from a large pool of candidates such that the selected new objects are similar to the query set (e.g. find words like jaguar or cheetah among all English words). This is a standard problem in similarity search and metric learning, and a typical application is to find new image tags given a small set of possible tags. Likewise, in the field of computational advertisement, given a set of high-value customers, the goal would be to find similar people. This is an important problem in many scientific applications, e.g. given a small set of interesting celestial objects, astrophysicists might want to find similar ones in large sky surveys.
Finally, among unsupervised tasks generative modeling is perhaps the most fundamental and important where also “sets” can play a significant role. In this setting, one has to learn a generative model of the distribution of the provided training set and capable of generating arbitrary many new sample points from the domain of this distribution. We would like to have detailed generative models for set data, like point clouds. This would allow data augmentation or style transfer directly in point cloud domain which are gaining prominence in field of self-driving cars and virtual reality.

**Main contributions.** In this work, (i) we propose a fundamental architecture, DeepSets, to deal with sets as inputs and show that the properties of this architecture are both necessary and sufficient (Section 2.3). (ii) We extend this architecture to allow for conditioning on arbitrary objects, and (iii) based on this architecture we develop a deep network that can operate on sets with possibly different sizes (Section 2.4). We show that a simple parameter-sharing scheme enables a general treatment of sets within supervised and semi-supervised settings. (iv) We extend the standard GAN to a hierarchical implicit model by combining Bayesian hierarchical modeling to learn to generate set data like point clouds. (Section 2.8) (v) Finally, we demonstrate the wide applicability of our framework through experiments on diverse problems (Section 2.5, 2.7, 2.9).

### 2.2 WHAT ARE INVARIANCE AND EQUIVARIANCE?

Although multilayer perceptrons are universal approximators, however, often in practice rather than throwing large multilayer perceptrons at the problem, we often use specially designed layers and architectures. The specialized layers perform better and computationally more efficient as they encode prior knowledge about the domain. Typically, this domain specific knowledge is incorporated by specifying invariance and equivariance that needs to be obeyed by the network. So, we begin by defining a standard neural network layer and then when would it be said to satisfy given invariance or equivariance.

**Definition 2.1** A standard neural network is a function \( f : \mathbb{R}^N \rightarrow \mathbb{R}^M \), of the form:

\[
y = f(x) = \sigma(\Theta x + b)
\]

where \( \Theta \) is the weight matrix and \( \sigma \) is a nonlinearity such as sigmoid function. \( \sigma \) is the point-wise application of \( \sigma \) to its input vector.

In the above definition, the affine portion is often generalized in literature to arbitrary input and output vector spaces over the fields of reals \( \mathbb{R} \). However, since elements of any vector space can be expressed in terms of basis (K. Hoffman and Kunze, n.d.) and any linear transformation can be expressed as a matrix (K. Hoffman and Kunze, n.d.), the definition above suffices.

Invariance is supposed to enforce notions of a feature to remain unaffected by some inconsequential alteration to the input. Formally,

**Definition 2.2** A function \( f : \mathcal{X} \rightarrow \mathcal{Y} \) is said to be invariant with respect to a transformation \( T : \mathcal{X} \rightarrow \mathcal{X} \) when \( f \)'s value is left unchanged upon application of \( T \) to the input, i.e.

\[
g(x) = g(T(x))
\]

On the other hand, equivariance does not require the values of \( f \) to be left unchanged by a transformation, but only that the input and output get transformed in a coupled fashion. To elaborate, consider the simple case when input space and output space are the same, i.e.
X = Y, then equivariance mean a transformation applied to the input of f gives the same result as applying it directly to the output y, i.e. T(f(x)) = f(T(x)). In general, the equivariance is defined formally as:

**Definition 2.3** A function \( f : X \to Y \) is said to be equivariant with respect the coupled transformations \( T : X \to X \) and \( T' : Y \to Y \) when f’s output is changed exactly according to \( T' \) upon application of \( T \) to the input, i.e.

\[
T'(g(x)) = g(T(x))
\]

(2.3)

An interesting property relating invariance and equivariance is that: any pooling of an equivariant function (e.g. max, sum, or requiring \( T' \) to be identity function) leads to an invariant function.

A common method of capitalizing on invariance and equivariance is through augmentation, i.e. adding extra training examples by applying appropriate transformations to account for changes to the data that should not affect changes to representation or corresponding label [7]. The invariance or equivariance is thus learned by seeing many examples from the same class (i.e. related to each other by an invariance transformation). Although sometimes effective, this method works by training over the extra augmented examples, which needs increased model capacity, more computation, extra training time, and there is still no guarantee that the resulting representation will be invariant or equivariant to the desired transformation.

We restrict the scope of this work to cases when the invariance and equivariance transformations are linear, i.e. it can be expressed by as matrix. Follow-up of this work, generalized to handling equivariance to any coupled group actions (S. Ravanbakhsh, Jeff Schneider, and Poczos, 2017). We will see next an example where such invariance/equivariance requirement is behind a highly successful specialized neural network architecture.

### 2.2.1 Example: CNN as Translation Invariance

Convolutional Neural Network (CNN) are popular in image processing, like object detection. In this section, instead of using minimalistic description of a vector space, we stick to the popular convention that input is 2D (e.g. images) and output is 2D as well, i.e. \( X = Y = \mathbb{R}^{M \times N} \). Thus, input-output relation of neural network layer would be:

\[
y_{ij} = \sigma \left( \sum_{k,l} \Theta_{ijkl} x_{kl} + b_{ij} \right)
\]

(2.4)

which can be conveniently expressed in tensor-matrix form as:

\[
y = \sigma(\Theta x + b)
\]

(2.5)

For image processing a desirable property would be translation equivariant. For example, if the layer is designed to identify a ball in an image, then a translation in any direction should not affect the outcome, as it’s still a rectangle after all. The outcome should also be translated, like the input. Formally, the translation transformation is a linear transformation and thus can be expressed as tensor \( T \). Keeping this in mind the translation equivariance can be expressed as:

\[
Ty = \sigma(\Theta Tx + b)
\]

\[
T\sigma(\Theta x + b) = \sigma(\Theta Tx + b)
\]

\[
\sigma(T\Theta x + Tb) = \sigma(\Theta Tx + b)
\]

\[
T\Theta x + Tx = \Theta Tx + b
\]

(2.6)
Step two to three follows from the fact that $\sigma$ is applied pointwise. Step three to four follows by assuming $\sigma$ is one-to-one.

Since the polynomial of $x$ in LHS and RHS are always equal, they must be the same and have equal coefficient. This implies:

$$T\Theta = \Theta T \text{ and } Tb = b$$

(2.7)

Thus for bias the condition reduces to mean that bias is same for all position $(i, j)$ and can be succinctly denoted by a scalar $c$. As for the weight, we need to further simplify

$$T\Theta = \Theta T \Rightarrow \Theta = T^{-1}\Theta T$$

(2.8)

Notice that the translation transformation tensor for a translation of $s$ units right and $t$ units up can be explicitly expressed as:

$$T_{ijkl} = \delta[(i-k)\%M = s, (j-l)\%N = t], -M < s < M \text{ and } -N < t < N$$

(2.9)

Note that inverse of translation by $(s, t)$ would be just translation in reverse direction, i.e. $(-s, -t)$. We plug these into (2.8) step by step. First let us evaluate:

$$(\Theta T)_{ijkl} = \sum_{m,n} \theta_{ijmn} T_{mnkl}$$

$$= \sum_{m,n} \theta_{ijmn} \delta[(m-k)\%M = s, (n-l)\%N = t]$$

$$= \delta_{ijkl}(k+s)\%M, (l+t)\%N$$

(2.10)

Now continuing with remaining $T^{-1}$, we obtain:

$$\Theta_{ijkl} = (T^{-1}\Theta T)_{ijkl}$$

$$= \sum_{m,n} T^{-1}_{ijklmn} (\Theta T)_{mnkl}$$

$$= \sum_{m,n} \delta[(i-m)\%M = -s, (j-n)\%N = -t] \Theta_{m,n,(k+s)\%M,(l+t)\%N}$$

$$= \delta_{ijkl}(i+s)\%M, (j+t)\%N, (k+s)\%M, (l+t)\%N$$

(2.11)

As the above holds for all $(s, t)$, there are many values in $\Theta$ are equal. To be specific, there are only $M \times N$ unique value, which can represented in a matrix $W \in \mathbb{R}^{M \times N}$ as:

$$W_{ij} = \Theta_{ij00} = \Theta_{(i+s)\%M,(j+t)\%N,0}, \forall 0 \leq i, s < M, 0 \leq j, t < N.$$  

(2.12)

or in other way round:

$$\Theta_{ijkl} = W_{(i-k)\%M,(j-l)\%N}.$$  

(2.13)

Now notice the application of the translation invariant $\Theta$ on a 2D input, like an image:

$$y_{ij} = \sigma \left( \sum_{m,n} \Theta_{ijmn} x_{mn} + b_{ij} \right)$$

$$= \sigma \left( \sum_{m,n} W_{(i-m)\%M,(j-n)\%N} x_{mn} + c \right)$$  

(2.14)

convolution!
This can be easily recognized as the popular Convolutional Neural Network (CNN) (LeCun et al., 1998). Thus the popular CNN automatically pops out when translation equivariance is imposed on a general neural network. It is interesting to note that original CNN in 1968 was not motivated on basis on invariance/equivariance (Hubel and Wiesel, 1968). It was not until in 1988, that CNN were understood to be consequence of shift invariance (W. Zhang, 1988). Motivated by this, in next section we will study effect of another type of invariance that has not be studied before, namely the permutation invariance.

### 2.3 Permutation Invariance and Equivariance

#### 2.3.1 Problem Definition

A function \( f \) transforms its domain \( \mathcal{X} \) into its range \( \mathcal{Y} \). Usually, the input domain is a vector space \( \mathbb{R}^d \) and the output response range is either a discrete space, e.g. \( \{0, 1\} \) in case of classification, or a continuous space \( \mathbb{R} \) in case of regression. Now, if the input is a set \( X = \{x_1, \ldots, x_M\}, x_m \in \mathcal{X} \), i.e., the input domain is the power set \( \mathcal{X} = 2^\mathcal{X} \), then we would like the response of the function to be “indifferent” to the ordering of the elements. In other words,

**Property 2.4** A function \( f : 2^\mathcal{X} \to \mathcal{Y} \) acting on sets must be permutation invariant to the order of objects in the set, i.e. for any permutation \( \pi : \)

\[
f([x_1, \ldots, x_m]) = f([x_{\pi(1)}, \ldots, x_{\pi(M)}])
\]  

(2.15)

In the supervised setting, given \( N \) examples of of \( X^{(1)}, \ldots, X^{(N)} \) as well as their labels \( y^{(1)}, \ldots, y^{(N)} \), the task would be to classify/regress (with variable number of predictors) while being permutation invariant w.r.t. predictors. Under unsupervised setting, the task would be to assign high scores to valid sets and low scores to improbable sets. These scores can then be used for set expansion tasks, such as image tagging or audience expansion in field of computational advertisement. In transductive setting, each instance \( x^{(n)}_m \) has an associated labeled \( y^{(n)}_m \). Then, the objective would be instead to learn a permutation equivariant function \( f : \mathcal{X}^M \to \mathcal{Y}^M \) that upon permutation of the input instances permutes the output labels, i.e. for any permutation \( \pi : \)

\[
f([x_{\pi(1)}, \ldots, x_{\pi(M)}]) = [f_{\pi(1)}(x), \ldots, f_{\pi(M)}(x)]
\]  

(2.16)

We want to study the structure of functions on sets. Their study in total generality is extremely difficult, so we analyze case-by-case. Roughly speaking, we claim that such functions must have a structure of the form \( f(X) = \rho(\sum_{x \in X} \phi(x)) \) for some functions \( \rho \) and \( \phi \). Over the next two sections we try to formally prove this structure of the permutation invariant functions.

#### 2.3.2 Countable Case

We begin by analyzing the invariant case when \( \mathcal{X} \) is a countable set and \( \mathcal{Y} = \mathbb{R} \), where the next theorem characterizes its structure.

**Theorem 2.5** Assume the elements are countable, i.e. \( |\mathcal{X}| < \aleph_0 \). A function \( f : 2^\mathcal{X} \to \mathbb{R} \) operating on a set \( X \) can be a valid set function, i.e. it is permutation invariant to the elements in \( \mathcal{X} \), if and only if it can be decomposed in the form \( \rho(\sum_{x \in X} \phi(x)) \), for suitable transformations \( \phi \) and \( \rho \).
Proof Permutation invariance follows from the fact that sets have no particular order, hence any function on a set must not exploit any particular order either. The sufficiency follows by observing that the function \( \rho (\sum_{x \in X} \phi(x)) \) satisfies the permutation invariance condition.

To prove necessity, i.e. that all functions can be decomposed in this manner, we begin by noting that there must be a mapping from the elements to natural numbers functions, since the elements are countable. Let this mapping be denoted by \( c : X \to \mathbb{N} \). Now if we let \( \phi(x) = 4^{-c(x)} \) then \( \sum_{x \in X} \phi(x) \) constitutes an unique representation for every set \( X \in 2^X \). Now a function \( \rho : \mathbb{R} \to \mathbb{R} \) can always be constructed such that \( f(X) = \rho (\sum_{x \in X} \phi(x)) \).

2.3.3 Uncountable Case

The extension to case when \( X \) is uncountable, e.g. \( X = [0, 1] \), is not so trivial. We could only prove in case of fixed set size, e.g. \( X = [0, 1]^M \) instead of \( X = 2^X = 2^{[0,1]} \), that any permutation invariant continuous function can be expressed as \( \rho (\sum_{x \in X} \phi(x)) \). Also, we show that there is a universal approximator of the same form. These results are discussed below.

To illustrate the uncountable case, we assume a fixed set size of \( M \). Without loss of generality we can let \( X = [0, 1] \). Then the domain becomes \([0,1]^M \). Also, to handle ambiguity due to permutation, we often define the domain to be the set \( X = \{ (x_1, ..., x_M) \in [0,1]^M : x_1 \leq x_2 \leq \cdots \leq x_M \} \) for some ordering of the elements in \( X \).

The proof builds on the famous Newton-Girard formulae which connect moments of a sample set (sum-of-power) to the elementary symmetric polynomials. But first we present some results needed for the proof. The first result establishes that sum-of-power mapping is injective.

**Lemma 2.6** Let \( X = \{ (x_1, ..., x_M) \in [0,1]^M : x_1 \leq x_2 \leq \cdots \leq x_M \} \). The sum-of-power mapping \( E : X \to \mathbb{R}^{M+1} \) defined by the coordinate functions

\[
Z_q := E_q(X) := \sum_{m=1}^{M} (x_m)^q, \quad q = 0, \ldots, M.
\]

is injective.

**Proof** Suppose for some \( u, v \in X \), we have \( E(u) = E(v) \). We will now show that it must be the case that \( u = v \). Construct two polynomials as follows:

\[
P_u(x) = \prod_{m=1}^{M} (x - u_m) \quad P_v(x) = \prod_{m=1}^{M} (x - v_m)
\]

If we expand the two polynomials we obtain:

\[
P_u(x) = x^M - a_1 x^{M-1} + \cdots + (-1)^{M-1} a_{M-1} x + (-1)^{M} a_M
\]

\[
P_v(x) = x^M - b_1 x^{M-1} + \cdots + (-1)^{M-1} b_{M-1} x + (-1)^{M} b_M
\]

with coefficients being elementary symmetric polynomials in \( u \) and \( v \) respectively, i.e.

\[
a_m = \sum_{1 \leq j_1 < j_2 < \cdots < j_m \leq M} u_{j_1} u_{j_2} \cdots u_{j_m} \quad b_m = \sum_{1 \leq j_1 < j_2 < \cdots < j_m \leq M} v_{j_1} v_{j_2} \cdots v_{j_m}
\]
These elementary symmetric polynomials can be uniquely expressed as a function of \(E(u)\) and \(E(v)\) respectively, by Newton-Girard formula. The \(m\)-th coefficient is given by the determinant of \(m \times m\) matrix having terms from \(E(u)\) and \(E(v)\) respectively:

\[
a_m = \frac{1}{m} \det \begin{pmatrix}
E_1(u) & 1 & 0 & 0 & \cdots & 0 \\
E_2(u) & E_1(u) & 1 & 0 & \cdots & 0 \\
E_3(u) & E_2(u) & E_1(u) & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
E_{m-1}(u) & E_{m-2}(u) & E_{m-3}(u) & E_{m-4}(u) & \cdots & 1 \\
E_m(u) & E_{m-1}(u) & E_{m-2}(u) & E_{m-3}(u) & \cdots & E_1(u)
\end{pmatrix}
\]

\[
b_m = \frac{1}{m} \det \begin{pmatrix}
E_1(v) & 1 & 0 & 0 & \cdots & 0 \\
E_2(v) & E_1(v) & 1 & 0 & \cdots & 0 \\
E_3(v) & E_2(v) & E_1(v) & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
E_{m-1}(v) & E_{m-2}(v) & E_{m-3}(v) & E_{m-4}(v) & \cdots & 1 \\
E_m(v) & E_{m-1}(v) & E_{m-2}(v) & E_{m-3}(v) & \cdots & E_1(v)
\end{pmatrix}
\]

Since we assumed \(E(u) = E(v)\) implying \([a_1, ..., a_M] = [b_1, ..., b_M]\), which in turn implies that the polynomials \(P_u\) and \(P_v\) are the same. Therefore, their roots must be the same, which shows that \(u = v\).

The second result we borrow from Ćurgus and Mascioni (2006) which establishes a homeomorphism between coefficients and roots of a polynomial.

**Theorem 2.7** (Ćurgus and Mascioni, 2006) The function \(f : \mathbb{C}^M \to \mathbb{C}^M\), which associates every \(a \in \mathbb{C}^M\) to the multiset of roots, \(f(a) \in \mathbb{C}^M\), of the monic polynomial formed using \(a\) as the coefficient i.e. \(x^M + a_1 x^{M-1} + \cdots + (-1)^{M-1} a_{M-1} x + (-1)^M a_M\), is a homeomorphism.

Among other things, this implies that (complex) roots of a polynomial depends continuously on the coefficients. We will use this fact for our next lemma.

Finally, we establish a continuous inverse mapping for the sum-of-power function.

**Lemma 2.8** Let \(X = \{(x_1, ..., x_M) \in [0, 1]^M : x_1 \leq x_2 \leq \cdots \leq x_M\}\). We define the sum-of-power mapping \(E : X \to \mathcal{Z}\) by the coordinate functions

\[
Z_q := E_q(X) := \sum_{m=1}^{M} (x_m)^q, \quad q = 0, ..., M.
\]

where \(\mathcal{Z}\) is the range of the function. The function \(E\) has a continuous inverse mapping.

**Proof** First of all note that \(\mathcal{Z}\), the range of \(E\), is a compact set. This follows from following observations:

- The domain of \(E\) is a bounded polytope (i.e. a compact set),
• $E$ is a continuous function, and
• image of a compact set under a continuous function is a compact set.

To show the continuity of inverse mapping, we establish connection to the continuous dependence of roots of polynomials on its coefficients.

As in Lemma 4, for any $u \in \mathcal{X}$, let $z = E(u)$ and construct the polynomial:

$$P_u(x) = \prod_{m=1}^{M} (x - u_m)$$

(2.23)

If we expand the polynomial we obtain:

$$P_u(x) = x^M - a_1 x^{M-1} + \cdots + (-1)^M a_M$$

(2.24)

with coefficients being elementary symmetric polynomials in $u$, i.e.

$$a_m = \sum_{1 \leq i_1 < i_2 < \cdots < i_m \leq M} u_{i_1} u_{i_2} \cdots u_{i_m}$$

(2.25)

These elementary symmetric polynomials can be uniquely expressed as a function of $z$ by Newton-Girard formula:

$$a_m = \frac{1}{m} \det\begin{pmatrix}
  z_1 & 1 & 0 & 0 & \cdots & 0 \\
  z_2 & z_1 & 1 & 0 & \cdots & 0 \\
  z_3 & z_2 & z_1 & 1 & \cdots & 0 \\
  \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
  z_{m-1} & z_{m-2} & z_{m-3} & z_{m-4} & \cdots & 1 \\
  z_m & z_{m-1} & z_{m-2} & z_{m-3} & \cdots & z_1
\end{pmatrix}$$

(2.26)

Since determinants are just polynomials, $a$ is a continuous function of $z$. Thus to show continuity of inverse mapping of $E$, it remains to show continuity from $a$ back to the roots $u$. In this regard, we invoke Theorem 5. Note that homeomorphism implies the mapping as well as its inverse is continuous. Thus, restricting to the compact set $\mathcal{Z}$ where the map from coefficients to roots only goes to the reals, the desired result follows. To explicitly check the continuity, note that limit of $E^{-1}(z)$, as $z$ approaches $z^*$ from inside $\mathcal{Z}$, always exists and is equal to $E^{-1}(z^*)$ since it does so in the complex plane.

With the lemma developed above we are in a position to tackle the main theorem.

**Theorem 2.9** Let $f : [0, 1]^M \to \mathbb{R}$ be a permutation invariant continuous function iff it has the representation

$$f(x_1, \ldots, x_M) = \rho \left( \sum_{m=1}^{M} \phi(x_m) \right)$$

(2.27)

for some continuous outer and inner function $\rho : \mathbb{R}^{M+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^{M+1}$ respectively. The inner function $\phi$ is independent of the function $f$. 
Proof The sufficiency follows by observing that the function $\rho \left( \sum_{m=1}^{M} \phi(x_m) \right)$ satisfies the permutation invariance condition.

To prove necessity, \textit{i.e.} that all permutation invariant continuous functions over the compact set can be expressed in this manner, we divide the proof into two parts, with outline in Figure 2.1. We begin by looking at the continuous embedding formed by the inner function: $E(X) = \sum_{m=1}^{M} \phi(x_m)$. Consider $\phi : \mathbb{R} \to \mathbb{R}^{M+1}$ defined as $\phi(x) = [1, x, x^2, \ldots, x^M]$. Now as $E$ is a polynomial, the image of $[0,1]^M$ in $\mathbb{R}^{M+1}$ under $E$ is a compact set as well, denote it by $Z$. Then by definition, the embedding $E : [0,1]^M \to Z$ is surjective. Using Lemma 4 and 6, we know that upon restricting the permutations, \textit{i.e.} replacing $[0,1]^M$ with $\mathcal{X} = \{(x_1,\ldots,x_M) \in [0,1]^M : x_1 \leq x_2 \leq \cdots \leq x_M\}$, the embedding $E : \mathcal{X} \to Z$ is injective with a continuous inverse. Therefore, combining these observation we get that $E$ is a homeomorphism between $\mathcal{X}$ and $Z$. Now it remains to show that we can map the embedding to desired target value, \textit{i.e.} to show the existence of the continuous map $\rho : Z \to \mathbb{R}$ such that $\rho(E(X)) = f(X)$. In particular consider the map $\rho(z) = f(E^{-1}(z))$. The continuity of $\rho$ follows directly from the fact that composition of continuous functions is continuous. Therefore we can always find continuous functions $\phi$ and $\rho$ to express any permutation invariant function $f$ as $\rho \left( \sum_{m=1}^{M} \phi(x_m) \right)$.

A very similar but more general results holds in case of any continuous function (not necessarily permutation invariant). The result is known as Kolmogorov–Arnold representation theorem (Khesin and Tabachnikov, 2014, Chap. 17) which we state below:

**Theorem 2.10 (Kolmogorov–Arnold representation)** Let $f : [0,1]^M \to \mathbb{R}$ be an arbitrary multivariate continuous function iff it has the representation

$$f(x_1,\ldots,x_M) = \rho \left( \sum_{m=1}^{M} \lambda_m \phi(x_m) \right)$$

(2.28)

with continuous outer and inner functions $\rho : \mathbb{R}^{2M+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^{2M+1}$. The inner function $\phi$ is independent of the function $f$.

This theorem essentially states a representation theorem for any multivariate continuous function. Their representation is very similar to the one we are proved, except for the dependence of inner transformation on the co-ordinate through $\lambda_m$. Thus it is reassuring that behind all

![Figure 2.1: Outline of the proof strategy for Theorem 2.1. The proof consists of two parts. First, we desire to show that we can find unique embeddings for each possible input, \textit{i.e.} we show that there exists a homeomorphism $E$ of the form $E(X) = \sum_{x \in X} \phi(x)$ between original domain and some higher dimensional space $Z$. The second part of the proof consists of showing we can map the embedding to desired target value, \textit{i.e.} to show the existence of the continuous map $\rho$ between $Z$ and original target space such that $f(X) = \rho \left( \sum_{x \in X} \phi(x) \right)$.](image)
the beautiful mathematics something intuitive is happening. If the function is permutation invariant, this dependence on co-ordinate of the inner transformation gets dropped!

Further we can show that arbitrary approximator having the same form can be obtained for continuous permutation-invariant functions.

**Theorem 2.11** Assume the elements are from a compact set in $\mathbb{R}^d$, i.e. possibly uncountable, and the set size is fixed to $M$. Then any continuous function operating on a set $X$, i.e. $f : \mathbb{R}^{d \times M} \rightarrow \mathbb{R}$ which is permutation invariant to the elements in $X$ can be approximated arbitrarily close in the form of $\rho \left( \sum_{x \in X} \phi(x) \right)$, for suitable transformations $\phi$ and $\rho$.

**Proof** Permutation invariance follows from the fact that sets have no particular order, hence any function on a set must not exploit any particular order either. The sufficiency follows by observing that the function $\rho \left( \sum_{x \in X} \phi(x) \right)$ satisfies the permutation invariance condition.

To prove necessity, i.e. that all continuous functions over the compact set can be approximated arbitrarily close in this manner, we begin noting that polynomials are universal approximators by Stone–Weierstrass theorem (Marsden and M. J. Hoffman, 1993, sec. 5.7). In this case the Chevalley-Shephard-Todd (CST) theorem (Bourbaki, 1990, chap. V, theorem 4), or more precisely, its special case, the Fundamental Theorem of Symmetric Functions states that symmetric polynomials are given by a polynomial of homogeneous symmetric monomials. The latter are given by the sum over monomial terms, which is all that we need since it implies that all symmetric polynomials can be written in the form required by the theorem. 

Finally, we still conjecture that even in case of sets of all sizes, i.e. when the domain is $2^{[0,1]}$, a representation of the form $f(X) = \rho \left( \sum_{x \in X} \phi(x) \right)$ should exist for all “continuous” permutation invariant functions for some suitable transformations $\rho$ and $\phi$. However, in this case even what a “continuous” function means is not clear as the space $2^{[0,1]}$ does not have any natural topology. As a future work, we want to study further by defining various topologies, like using Fréchet distance as used in Curgus and Mascioni (2006) or MMD distance. Our preliminary findings in this regards hints that using MMD distance if the representation is allowed to be in $\ell^2$, instead of being finite dimensional, then the conjecture seems to be provable. Thus, clearly this direction needs further exploration. We end this section by providing some examples:

**Examples:**

- $x_1 x_2 (x_1 + x_2 + 3)$, Consider $\phi(x) = [x, x^2, x^3]$ and $\rho([u, v, w]) = uv - w + 3(u^2 - v)/2$, then $\rho(\phi(x_1) + \phi(x_2))$ is the desired function.
- $x_1 x_2 x_3 + x_1 + x_2 + x_3$, Consider $\phi(x) = [x, x^2, x^3]$ and $\rho([u, v, w]) = (u^3 + 2w - 3uv)/6 + u$, then $\rho(\phi(x_1) + \phi(x_2) + \phi(x_3))$ is the desired function.
- $1/n(x_1 + x_2 + x_3 + ... + x_m)$, Consider $\phi(x) = [1, x]$ and $\rho([u, v]) = v/u$, then $\rho(\phi(x_1) + \phi(x_2) + \phi(x_3) + ... + \phi(x_m))$ is the desired function.
- $\max(x_1, x_2, x_3, ... , x_m)$, Consider $\phi(x) = [e^{ax}, xe^{ax}]$ and $\rho([u, v]) = v/u$, then as $\alpha \to \infty$, then we have $\rho(\phi(x_1) + \phi(x_2) + \phi(x_3) + ... + \phi(x_m))$ approaching the desired function.
- Second largest among $\{x_1, x_2, x_3, ... , x_m\}$, Consider $\phi(x) = [e^{ax}, xe^{ax}]$ and $\rho([u, v]) = (v - (v/u)e^{av}/u)/(u - e^{av}/u)$, then as $\alpha \to \infty$, we have $\rho(\phi(x_1) + \phi(x_2) + \phi(x_3) + ... + \phi(x_m))$ approaching the desired function.
2.3.4 Specialized Layer

Next, we analyze the equivariant case when \( x = y = \mathbb{R} \) and \( f \) is restricted to be a neural network layer. Our goal is to design neural network layers that are equivariant to permutations of elements in the input \( x \). The function \( f : \mathbb{X}^M \to \mathbb{Y}^M \) is \textit{equivariant} to the permutation of its inputs iff

\[
f(\pi x) = \pi f(x) \quad \forall \pi \in S_M
\]

where the symmetric group \( S_M \) is the set of all permutation of indices \( 1, \ldots, M \).

Consider the standard neural network layer

\[
f_\Theta(x) = \sigma(\Theta x) \quad \Theta \in \mathbb{R}^{M \times M}
\]

(2.29)

where \( \Theta \) is the weight vector and \( \sigma : \mathbb{R} \to \mathbb{R} \) is a nonlinearity such as sigmoid function. The following lemma states the necessary and sufficient conditions for permutation-equivariance in this type of function.

**Theorem 2.12** The function \( f_\Theta : \mathbb{R}^M \to \mathbb{R}^M \) as defined in (2.29) is permutation equivariant if and only if all the off-diagonal elements of \( \Theta \) are tied together and all the diagonal elements are equal as well. That is,

\[
\Theta = \lambda I + \gamma (1 1^T)
\]

\( \lambda, \gamma \in \mathbb{R} \quad 1 = [1, \ldots, 1]^T \in \mathbb{R}^M \)

where \( I \in \mathbb{R}^{M \times M} \) is the identity matrix.

**Proof** From definition of permutation equivariance \( f_\Theta(\pi x) = \pi f_\Theta(x) \) and definition of \( f \) in (2.29), the condition becomes \( \sigma(\Theta \pi x) = \pi \sigma(\Theta x) \), which (assuming sigmoid is a bijection) is equivalent to \( \Theta \pi = \pi \Theta \). Therefore we need to show that the necessary and sufficient conditions for the matrix \( \Theta \in \mathbb{R}^{M \times M} \) to commute with all permutation matrices \( \pi \in S_M \) is given by this proposition. We prove this in both directions:

- To see why \( \Theta = \lambda I + \gamma (1 1^T) \) commutes with any permutation matrix, first note that commutativity is linear – that is

  \[
  \Theta_1 \pi = \pi \Theta_1 \land \Theta_2 \pi = \pi \Theta_2 \implies (a \Theta_1 + b \Theta_2) \pi = \pi (a \Theta_1 + b \Theta_2).
  \]

  Since both Identity matrix \( I \), and constant matrix \( 1 1^T \), commute with any permutation matrix, so does their linear combination \( \Theta = \lambda I + \gamma (1 1^T) \).

- We need to show that in a matrix \( \Theta \) that commutes with “all” permutation matrices

  - All diagonal elements are identical: Let \( \pi_{k,l} \) for \( 1 \leq k, l \leq M, k \neq l \), be a transposition (i.e. a permutation that only swaps two elements). The inverse permutation matrix of \( \pi_{k,l} \) is the permutation matrix of \( \pi_{l,k} = \pi_{k,l}^{-1} \). We see that commutativity of \( \Theta \) with the transposition \( \pi_{k,l} \) implies that \( \Theta_{k,k} = \Theta_{l,l} \):

    \[
    \pi_{k,l} \Theta = \Theta \pi_{k,l} \implies \pi_{k,l} \Theta \pi_{l,k} = \Theta \implies (\pi_{l,k} \Theta \pi_{l,k})_{l,1} = \Theta_{l,1} \implies \Theta_{k,k} = \Theta_{l,l}
    \]

  Therefore, \( \pi \) and \( \Theta \) commute for any permutation \( \pi \), they also commute for any transposition \( \pi_{k,l} \) and therefore \( \Theta_{l,i} = \lambda i \forall i \).

  - All off-diagonal elements are identical: We show that since \( \Theta \) commutes with any product of transpositions, any choice two off-diagonal elements should be identical. Let
(i, j) and (i', j') be the index of two off-diagonal elements (i.e. \( i \neq j \) and \( i' \neq j' \)). Moreover for now assume \( i \neq i' \) and \( j \neq j' \). Application of the transposition \( \pi_i : \Theta \rightarrow \Theta \), sways swaps the rows \( i, i' \) in \( \Theta \). Similarly, \( \Theta \pi_{i,j'} \) switches the \( j \)th column with \( j' \)th column. From commutativity property of \( \Theta \) and \( \pi_i \in S_n \) we have

\[
\pi_{i,j} \pi_{i,\pi} \Theta = \Theta \pi_{i,j} \pi_{i,\pi} \Rightarrow \pi_{i,j} \pi_{i,\pi} \Theta (\pi_{i,j} \pi_{i,\pi})^{-1} = \Theta \Rightarrow \pi_{i,j} \pi_{i,\pi} \Theta \pi_{i,j'} \pi_{i,\pi} = \Theta_{i,j} \Rightarrow \Theta_{i,j'} = \Theta_{i,j}
\]

where in the last step we used our assumptions that \( i \neq i', j \neq j', i \neq j \) and \( i' \neq j' \). In the cases where either \( i = i' \) or \( j = j' \), we can use the above to show that \( \Theta_{i,j} = \Theta_{i',j'} \) and \( \Theta_{i',j'} = \Theta_{i,j} \), for some \( i'' \neq i, i' \) and \( j'' \neq j, j' \), and conclude \( \Theta_{i,j} = \Theta_{i',j'} \).

This result can be easily extended to higher dimensions, i.e., \( \mathcal{X} = \mathbb{R}^d \) when \( \lambda, \gamma \) can be matrices.

2.3.5 Related Results

The general form of Theorem 2.5 is closely related with important results in different domains. Here, we quickly review some of these connections.

**De Finetti Theorem.** A related concept is that of an exchangeable model in Bayesian statistics. It is backed by deFinetti’s theorem which states that any exchangeable model can be factored as

\[
p(X|\alpha, M_0) = \int d\theta \left[ \prod_{m=1}^{M} p(x_m|\theta) \right] p(\theta|\alpha, M_0),
\]

where \( \theta \) is some latent feature and \( \alpha, M_0 \) are the hyper-parameters of the prior. To see that this fits into our result, let us consider exponential families with conjugate priors, where we can analytically calculate the integral of (2.30). In this special case \( p(x|\theta) = \exp \left( \langle \phi(x), \theta \rangle - g(\theta) \right) \) and \( p(\theta|\alpha, M_0) = \exp \left( \langle \theta, \alpha \rangle - M_0 g(\theta) - h(\alpha, M_0) \right) \). Now if we marginalize out \( \theta \), we get a form which looks exactly like the one in Theorem 2.5

\[
p(X|\alpha, M_0) = \exp \left( \alpha + \sum_{m} \phi(x_m), M_0 + M \right) - h(\alpha, M_0).
\]

**Reprenter Theorem and Kernel Machines.** Support distribution machines use \( f(p) = \sum_{i} \alpha_i y_i K(p_i, p) + b \) as the prediction function (Muandet, Fukumizu et al., 2012; Poczos, L. Xiong, et al., 2012), where \( p_i, p \) are distributions and \( \alpha_i, b \in \mathbb{R} \). In practice, the \( p_i, p \) distributions are never given to us explicitly, usually only i.i.d. sample sets are available from these distributions, and therefore we need to estimate kernel \( K(p, q) \) using these samples. A popular approach is to use \( \tilde{K}(p, q) = \frac{1}{MM'} \sum_{i,j} k(x_i, y_j) \), where \( k \) is another kernel operating on the samples \( \{x_i\}_{i=1}^{M} \sim p \) and \( \{y_j\}_{j=1}^{M'} \sim q \). Now, these prediction functions can be seen fitting into the structure of our Theorem.

**Spectral Methods.** A consequence of the polynomial decomposition is that spectral methods (Anandkumar et al., 2014) can be viewed as a special case of the mapping \( \rho \circ \phi(X) \): in that case one can compute polynomials, usually only up to a relatively low degree (such as \( k = 3 \)), to perform inference about statistical properties of the distribution. The statistics are exchangeable in the data, hence they could be represented by the above map.
2.4 DEEP SETS

2.4.1 Architecture

2.4.1.1 Invariant model

The structure of permutation invariant functions in Theorem 2.5 hints at a general strategy for inference over sets of objects, which we call deep sets. Replacing $\phi$ and $\rho$ by universal approximators leaves matters unchanged, since, in particular, $\phi$ and $\rho$ can be used to approximate arbitrary polynomials. Then, it remains to learn these approximators. This yields in the following model:

- Each instance $x_m \forall 1 \leq m \leq M$ is transformed (possibly by several layers) into some representation $\phi(x_m)$.
- The addition $\sum_m \phi(x_m)$ of these representations processed using the $\rho$ network very much in the same manner as in any deep network (e.g. fully connected layers, nonlinearities, etc).
- Optionally: If we have additional meta-information $z$, then the above mentioned networks could be conditioned to obtain the conditioning mapping $\phi(x_m|z)$.

In other words, the key to deep sets is to add up all representations and then apply nonlinear transformations.

The overall model structure is illustrated in Figure 2.2.

This architecture has a number of desirable properties in terms of universality and correctness. We assume in the following that the networks we choose are, in principle, universal approximators. That is, we assume that they can represent any functional mapping. This is a well established property, cf. Micchelli (1984) for details in the case of radial basis function networks.

What remains is to state the derivatives with regard to this novel type of layer. Assume parametrizations $w_\rho$ and $w_\phi$ for $\rho$ and $\phi$ respectively. Then we have

\[
\partial_w \phi \rho \left( \sum_{x' \in X} \phi(x') \right) = \rho' \left( \sum_{x' \in X} \phi(x) \right) \sum_{x' \in X} \partial_w \phi(x')
\]

This result reinforces the common knowledge of parameter tying in deep networks when ordering is irrelevant. Our result backs this practice with theory and strengthens it by proving that it is the only way to do it.

2.4.1.2 Equivariant model

Our goal is to design neural network layers that are equivariant to the permutations of elements in the input $x$. Based on Theorem 2.12, a neural network layer $f_\Theta(x)$ is permutation equivariant if and only if all the off-diagonal elements of $\Theta$ are tied together and all the diagonal elements are equal as well, i.e., $\Theta = \lambda I + \gamma (11^T)$ for $\lambda, \gamma \in \mathbb{R}$. 
This function is simply a non-linearity applied to a weighted combination of i) its input $1x$ and; ii) the sum of input values $(11^T)x$. Since summation does not depend on the permutation, the layer is permutation-equivariant. Therefore we can manipulate the operations and parameters in this layer, for example to get another variation, e.g.:

$$f(x) \doteq \sigma(\lambda x + \gamma \text{maxpool}(x)1).$$ (2.32)

where the maxpooling operation over elements of the set (similar to sum) is commutative. In practice, this variation performs better in some applications. This may be due to the fact that for $\lambda = \gamma$, the input to the non-linearity is max-normalized. Since composition of permutation equivariant functions is also permutation equivariant, we can build DeepSets by stacking such layers as explained below.

So far we assumed that each instance $x_m \in \mathbb{R}$ – i.e. a single input and also output channel. For **multiple input-output channels**, we may speed up the operation of the layer using matrix multiplication. For $D/D'$ input/output channels (i.e. $x \in \mathbb{R}^{M \times D}$, $y \in \mathbb{R}^{M \times D'}$, this layer becomes

$$f(x) = \sigma(x\Lambda - 11^T x\Gamma)$$ (2.33)

where $\Lambda, \Gamma \in \mathbb{R}^{D \times D'}$ are model parameters. As before, we can have a maxpool version as: $f(x) = \sigma(x\Lambda - 1\text{maxpool}(x)\Gamma$) where maxpool$(x) = (\max x_m x) \in \mathbb{R}^{1 \times D}$ is a row-vector of maximum value of $x$ over the “set” dimension. We may further reduce the number of parameters in favor of better generalization by factoring $\Gamma$ and $\Lambda$ and keeping a single $\Lambda \in \mathbb{R}^{D,D'}$ and $\beta \in \mathbb{R}^{D'}$

$$f(x) = \sigma(\beta + (x - 1\text{maxpool}(x))\Gamma)$$ (2.34)

**Stacking**: Since composition of permutation equivariant functions is also permutation equivariant, we can build deep models by stacking layers of (2.34). Moreover, application of any commutative pooling operation (e.g. max-pooling) over the set instances produces a permutation invariant function.

### 2.4.2 Other Related Works

Several recent works study equivariance and invariance in deep networks w.r.t. general group of transformations (Cohen and Welling, 2016; Gens and Domingos, 2014; S. Ravanbakhsh, Jeff Schneider, and Póczos, 2017). For example, X. Chen, Cheng, and Mallat (2014) construct deep permutation invariant features by pairwise coupling of features at the previous layer, where $f_{ij}(x_i, x_j) = \|x_i - x_j, x_i + x_j\|$ is invariant to transposition of $i$ and $j$. Pairwise interactions within sets have also been studied in M. B. Chang et al. (2016) and Guttenberg et al. (2016). Vinyals, S. Bengio, and Kudlur (2015) approach unordered instances by finding “good” orderings.

The idea of pooling a function across set-members is not new. In Lopez-Paz et al. (2016), pooling was used binary classification task for causality on a set of samples. B. Shi et al. (2015) use
pooling across a panoramic projection of 3D object for classification, while Hang Su et al. (2015) perform pooling across multiple views. Hartford, Wright, and Leyton-Brown (2016) observe the invariance of the payoff matrix in normal form games to the permutation of its rows and columns (i.e. player actions) and leverage pooling to predict the player action. The need of permutation equivariance also arise in deep learning over sensor networks and multi-agent settings, where a special case of Theorem 2.12 has been used as the architecture (Sukhbaatar, Fergus, et al., 2016).

In light of these related works, we would like to emphasize our novel contributions: (i) the universality result of Theorem 2.5 for permutation invariance that also relates DeepSets to other machine learning techniques, see Section 2.3.5; (ii) the permutation equivariant layer of (2.32), which, according to Theorem 2.12 identifies necessary and sufficient form of parameter-sharing in a standard neural layer and; (iii) novel application settings that we study next.

2.5 EXPERIMENTS ON DISCRIMINATIVE TASKS

2.5.1 Estimate Population Statistics

In the first experiment, we learn entropy and mutual information of Gaussian distributions, without providing any information about Gaussianity to DeepSets. The Gaussians are generated as follows:

- Rotation: We randomly chose a 2 × 2 covariance matrix \( \Sigma \), and then generated \( N \) sample sets from \( N(0, R(\alpha) \Sigma R(\alpha)^T) \) of size \( M = [300 - 500] \) for \( N \) random values of \( \alpha \in [0, \pi] \). Our goal was to learn the entropy of the marginal distribution of first dimension. \( R(\alpha) \) is the rotation matrix.

- Correlation: We randomly chose a \( d \times d \) covariance matrix \( \Sigma \) for \( d = 16 \), and then generated \( N \) sample sets from \( N(0, [\Sigma, \alpha \Sigma; \alpha \Sigma, \Sigma]) \) of size \( M = [300 - 500] \) for \( N \) random values of \( \alpha \in (-1, 1) \). Goal was to learn the mutual information of among the first \( d \) and last \( d \) dimension.

- Rank 1: We randomly chose \( v \in \mathbb{R}^{32} \) and then generated a sample sets from \( N(0, I + \lambda vv^T) \) of size \( M = [300 - 500] \) for \( N \) random values of \( \lambda \in (0, 1) \). Goal was to learn the mutual information.
Figure 2.6: Population statistic estimation: Top set of figures, show prediction of DeepSets vs SDM for N = 2^{10} case. Bottom set of figures, depict the mean squared error behavior as number of sets is increased. SDM has lower error for small N and DeepSets requires more data to reach similar accuracy. But for high dimensional problems deep sets easily scales to large number of examples and produces much lower estimation error. Note that the N x N matrix inversion in SDM makes it prohibitively expensive for N > 2^{14} = 16384.

- Random: We chose N random d x d covariance matrices Σ for d = 32, and using each, generated a sample set from N(0, Σ) of size M = [300 – 500]. Goal was to learn the mutual information.

[July 17, 2018 at 16:21 – MT version 0.1]
2.5 Experiments on Discriminative Tasks

Figure 2.7: Accuracy of digit summation with text (left) and image (right) inputs. All approaches are trained on tasks of length 10 at most, tested on examples of length up to 100. We see that DeepSets generalizes better.

We train using $L_2$ loss with a DeepSets architecture having 3 fully connected layers with ReLU activation for both transformations $\phi$ and $\rho$. We compare against Support Distribution Machines (SDM) using a RBF kernel (Poczos, L. Xiong, et al., 2012), and analyze the results in Figure 2.6.

2.5.2 Sum of Digits

Next, we compare to what happens if our set data is treated as a sequence. We consider the task of finding sum of a given set of digits. We consider two variants of this experiment:

**Text** We randomly sample a subset of maximum $M = 10$ digits from this dataset to build $100k$ “sets” of training images, where the set-label is sum of digits in that set. We test against sums of $M$ digits, for $M$ starting from 5 all the way up to 100 over another $100k$ examples.

**Image** MNIST8m (Loosli, Canu, and Léon Bottou, 2007) contains 8 million instances of $28 \times 28$ grey-scale stamps of digits in $\{0, \ldots, 9\}$. We randomly sample a subset of maximum $M = 10$ images from this dataset to build $N = 100k$ “sets” of training and $100k$ sets of test images, where the set-label is the sum of digits in that set (i.e. individual labels per image is unavailable). We test against sums of $M$ images of MNIST digits, for $M$ starting from 5 all the way up to 50.

We compare against recurrent neural networks – LSTM and GRU. All models are defined to have similar number of layers and parameters. The output of all models is a scalar, predicting the sum of $N$ digits. Training is done on tasks of length 10 at most, while at test time we use examples of length up to 100. The accuracy, i.e. exact equality after rounding, is shown in Figure 2.7. DeepSets generalize much better. Note for image case, the best classification error for single digit is around $p = 0.01$ for MNIST8m, so in a collection of $N$ of images at least one image will be misclassified is $1 - (1 - p)^N$, which is 40% for $N = 50$. This matches closely with observed value in Figure 2.7(b).

2.5.3 Point Cloud Classification

A point-cloud is a set of low-dimensional vectors. This type of data is frequently encountered in various applications like robotics, vision, and cosmology. In these applications, existing methods often convert the point-cloud data to voxel or mesh representation as a preprocessing step, e.g. H.-W. Lin, Tai, and G.-J. Wang (2004), Maturana and Scherer (2015), and S. Ravanbakhsh, Junier Oliva, et al. (2016). Since the output of many range sensors, such as LiDAR, is in the form of point-cloud, direct application of deep learning methods to point-cloud is highly desirable.
Moreover, it is easy and cheaper to apply transformations, such as rotation and translation, when working with point-clouds than voxelized 3D objects.

As point-cloud data is just a set of points, we can use DeepSets to classify point-cloud representation of a subset of ShapeNet objects (A. X. Chang et al., 2015), called ModelNet40 (Z. Wu et al., 2015). This subset consists of 3D representation of 9,843 training and 2,468 test instances belonging to 40 classes of objects. We produce point-clouds with 100, 1000 and 5000 particles each \((x, y, z)\)-coordinates) from the mesh representation of objects using the point-cloud-library’s sampling routine (Rusu and Cousins, 2011). Each set is normalized by the initial layer of the deep network to have zero mean (along individual axes) and unit (global) variance. Table 2.1 compares our method using three permutation equivariant layers against the competition. Note that we achieve our best accuracy using \(5000 \times 3\) dimensional representation of each object, which is much smaller than most other methods. All other techniques use either voxelization or multiple view of the 3D object for classification.

**Training Details** We use a network comprising of 3 permutation-equivariant layers with 256 channels followed by max-pooling over the set structure. The resulting vector representation of the set is then fed to a fully connected layer with 256 units followed by a 40-way softmax unit. We use Tanh activation at all layers and dropout on the layers after set-max-pooling (i.e. two dropout operations) with 50% dropout rate. Applying dropout to permutation-equivariant layers for point-cloud data deteriorated the performance. We observed that using different types of permutation-equivariant layers (see Section 2.4) and as few as 64 channels for set layers changes the result by less than 5% in classification accuracy.

For the setting with 5000 particles, we increase the number of units to 512 in all layers and randomly rotate the input around the \(z\)-axis. We also randomly scale the point-cloud by \(s \sim U(0.8, 1.2)\). For this setting only, we use Adamax (D. P. Kingma and Ba, 2014) instead of Adam and reduce learning rate from 0.001 to 0.0005.

<table>
<thead>
<tr>
<th>Model</th>
<th>Instance Size</th>
<th>Representation</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DShapeNets (Z. Wu et al., 2015)</td>
<td>(30^3)</td>
<td>voxels (using convolutional deep belief net)</td>
<td>77%</td>
</tr>
<tr>
<td>VoxNet (Maturana and Scherer, 2015)</td>
<td>(32^3)</td>
<td>voxels (voxels from point-cloud + 3D CNN)</td>
<td>83.10%</td>
</tr>
<tr>
<td>MVCNN (Hang Su et al., 2015)</td>
<td>(164 \times 164 \times 12)</td>
<td>multi-view images (2D CNN + view-pooling)</td>
<td>90.1%</td>
</tr>
<tr>
<td>VRN Ensemble (Brock et al., 2016)</td>
<td>(32^3)</td>
<td>voxels (3D CNN, variational autoencoder)</td>
<td>95.54%</td>
</tr>
<tr>
<td>3D GAN (J. Wu et al., 2016)</td>
<td>(64^3)</td>
<td>voxels (3D CNN, generative adversarial training)</td>
<td>83.3%</td>
</tr>
<tr>
<td>DeepSets</td>
<td>(5000 \times 3)</td>
<td>point-cloud</td>
<td>90 ± 3%</td>
</tr>
<tr>
<td>DeepSets</td>
<td>(1000 \times 3)</td>
<td>point-cloud</td>
<td>87 ± .1%</td>
</tr>
<tr>
<td>DeepSets</td>
<td>(100 \times 3)</td>
<td>point-cloud</td>
<td>82 ± 2%</td>
</tr>
</tbody>
</table>

Table 2.1: Classification accuracy and the representation-size used by different methods on the ModelNet40.
Figure 2.8: Each box is the particle-cloud maximizing the activation of a unit at the first (top) and second (bottom) permutation-equivariant layers of our model. Two images of the same column are two different views of the same point-cloud.

**Features.** To visualize the features learned by the set layers, we used Adamax (D. P. Kingma and Ba, 2014) to locate 1000 particle coordinates maximizing the activation of each unit. Activating the tanh units beyond the second layer proved to be difficult. Figure 2.8 shows the particle-cloud-features learned at the first and second layers of our deep network. We observed that the first layer learns simple localized (often cubic) point-clouds at different \((x, y, z)\) locations, while the second layer learns more complex surfaces with different scales and orientations.

### 2.5.4 Set Anomaly Detection

The objective here is to find the anomalous face in each set, simply by observing examples and without any access to the attribute values. CelebA dataset (Z. Liu et al., 2015) contains 202,599 face images, each annotated with 40 boolean attributes. We build \(N = 18,000\) sets of \(64 \times 64\) stamps, using these attributes each containing \(M = 16\) images (on the training set) as follows: randomly select 2 attributes, draw 15 images having those attributes, and a single target image where both attributes are absent. Using a similar procedure we build sets on the test images. No individual person’s face appears in both train and test sets.

Our deep neural network consists of 9 convolution layers with \(3 \times 3\) receptive fields. The model has convolution layers with 32, 32, 64 feature-maps followed by max-pooling followed by 2D convolution layers with 64, 64, 128 feature-maps followed by another max-pooling layer. The final set of convolution layers have 128, 128, 256 feature-maps, followed by a max-pooling layer with pool-size of 5 that reduces the output dimension to batch-size \(M \times 256\), where the set-size \(M = 16\). This is then forwarded to three permutation-equivariant layers with 256, 128 and 1 output channels. The output of final layer is fed to the softmax, to identify the outlier. (Note that one could identify arbitrary number of outliers using a sigmoid activation at the output.) We use exponential linear units (Clevert, Unterthiner, and Hochreiter, 2015), drop out with 20% dropout rate at convolutional layers and 50% dropout rate at the first two set layers. When applied to set layers, the selected feature (channel) is simultaneously dropped in all the set members of that particular set. We use Adam (D. P. Kingma and Ba, 2014) for optimization and use batch-normalization only in the convolutional layers. We use mini-batches of 8 sets, for a

---

1 We started from uniformly distributed set of particles and used a learning rate of \(0.01\) for Adamax, with first and second order moments of \(1\) and \(0.9\) respectively. We optimized the input in \(10^5\) iterations. The results of Figure 2.8 are limited to instances where tanh units were successfully activated. Since the input at the first layer of our deep network is normalized to have a zero mean and unit standard deviation, we do not need to constrain the input while maximizing unit’s activation.
Figure 2.9: Each row shows a set, constructed from CelebA dataset, such that all set members except for an outlier, share at least two attributes (on the right). The outlier is identified with a red frame. The model is trained by observing examples of sets and their anomalous members, without access to the attributes. The probability assigned to each member by the outlier detection network is visualized using a red bar at the bottom of each image. The probabilities in each row sum to one.

total of 128 images per batch. Our trained model successfully finds the anomalous face in 75% of test sets. Visually inspecting these instances suggests that the task is non-trivial even for humans; see Figure 2.9.

As a baseline, we repeat the same experiment by using a set-pooling layer after convolution layers, and replacing the permutation-equivariant layers with fully connected layers of same size, where the final layer is a 16-way softmax. The resulting network shares the convolution filters for all instances within all sets, however the input to the softmax is not equivariant to the permutation of input images. Permutation equivariance seems to be crucial here as the baseline model achieves a training and test accuracy of $\sim 6.3\%$; the same as random selection.

## 2.6 Set Expansion

In the set expansion task, we are given a set of objects that are similar to each other and our goal is to find new objects from a large pool of candidates such that the selected new objects are similar to the query set. To achieve this one needs to reason out the concept connecting the given set and then retrieve words based on their relevance to the inferred concept. It is an important task due to wide range of potential applications including personalized information retrieval, computational advertisement, tagging large amounts of unlabeled or weakly labeled datasets.

Going back to de Finetti’s theorem in Section 2.3.5, where we consider the marginal probability of a set of observations, the marginal probability allows for very simple metric for scoring additional elements to be added to X. In other words, this allows one to perform set expansion via the following score

$$ s(x|X) = \log p(X \cup \{x\} | \alpha) - \log p(X|\alpha)p(\{x\} | \alpha) $$

(2.35)

Note that $s(x|X)$ is the point-wise mutual information between $x$ and $X$. Moreover, due to exchangeability, it follows that regardless of the order of elements we have

$$ S(X) = \sum_m s(x_m | \{x_{m-1}, \ldots, x_1\}) = \log p(X|\alpha) - \sum_{m=1}^M \log p(\{x_m\} | \alpha) $$

(2.36)
Figure 2.10: Each row shows a set, constructed from CelebA dataset, such that all set members except for an outlier, share at least two attributes (on the right). The outlier is identified with a red frame. The model is trained by observing examples of sets and their anomalous members, without access to the attributes. The probability assigned to each member by the outlier detection network is visualized using a red bar at the bottom of each image.

When inferring sets, our goal is to find set completions \( \{x_{m+1}, \ldots, x_M\} \) for an initial set of query terms \( \{x_1, \ldots, x_m\} \), such that the aggregate set is coherent. This is the key idea of the Bayesian Set algorithm (Ghahramani and Heller, 2005).

### 2.6.1 Bayes Set

Bayes Set (Ghahramani and Heller, 2005) assumes parametric formulation in the above approach. If we consider exponential families, the above approach assumes a particularly nice form whenever we have conjugate priors. Here we have

\[
p(x|\theta) = \exp (\langle \phi(x), \theta \rangle - g(\theta)) \quad \text{and} \quad p(\theta|\alpha, M_0) = \exp (\langle \theta, \alpha \rangle - M_0 g(\theta) - h(\alpha, M_0)).
\]

(2.37)

The mapping \( \phi : x \rightarrow F \) is usually referred as sufficient statistic of \( x \) which maps \( x \) into a feature space \( F \). Moreover, \( g(\theta) \) is the log-partition (or cumulant-generating) function. Finally, \( p(\theta|\alpha, M_0) \) denotes the conjugate distribution which is in itself a member of the exponential
family. It has the normalization \( h(\alpha, M_0) = \int d\theta \exp ((\theta, \alpha) - M_0 g(\theta)) \). The advantage of this is that \( s(x|X) \) and \( S(X) \) can be computed in closed form (Ghahramani and Heller, 2005) via

\[
s(X) = h(\alpha + \phi(X), M_0 + M) + (M - 1)h(\alpha, M_0) - \sum_{m=1}^{M} h(\alpha + \phi(x_m), M + 1) \tag{2.38}
\]

\[
s(x|X) = h(\alpha + \phi(x|X), M_0 + M + 1) + h(\alpha, M_0) - h(\alpha + \phi(x), M + 1) \tag{2.39}
\]

For convenience we defined the sufficient statistic of a set to be the sum over its constituents, i.e. \( \phi(x) = \sum_m \phi(x_m) \). It allows for very simple computation and maximization over additional elements to be added to \( X \), since \( \phi(X) \) can be precomputed.

2.6.1.1 Beta-Binomial Model

The model is particularly simple when dealing with the Binomial distribution and its conjugate Beta prior, since the ratio of Gamma functions allows for simple expressions. In particular, we have

\[
h(\beta) = \log \Gamma(\beta^+) + \log \Gamma(\beta^-) - \Gamma(\beta).
\]

With some slight abuse of notation we let \( \alpha = (\beta^+, \beta^-) \) and \( M_0 = \beta^+ + \beta^- \). Setting \( \phi(1) = (1,0) \) and \( \phi(0) = (0,1) \) allows us to obtain \( \phi(x) = (M^+, M^-) \), i.e. \( \phi(X) \) contains the counts of occurrences of \( x_m = 1 \) and \( x_m = 0 \) respectively. This leads to the following score functions

\[
s(X) = \log \Gamma(\beta^+ + M^+) + \log \Gamma(\beta^- + M^-) - \log \Gamma(\beta + M)
\]

\[
- \log \Gamma(\beta^+) - \log \Gamma(\beta^-) + \log \Gamma(\beta) - M^+ \log \frac{\beta^+}{\beta} - M^- \log \frac{\beta^-}{\beta}
\]

\[
s(x|X) = \begin{cases} 
\log \frac{\beta^+ + M^+}{\beta^+ M^+} - \log \frac{\beta^+}{\beta^+} \text{ if } x = 1 \\
\log \frac{\beta^- + M^-}{\beta^- M^-} - \log \frac{\beta^-}{\beta^-} \text{ otherwise}
\end{cases}
\tag{2.42}
\]

This is the model used by Ghahramani and Heller (2005) when estimating Bayesian Sets for objects. In particular, they assume that for any given object \( x \) the vector \( \phi(x) \in \{0,1\}^d \) is a d-dimensional binary vector, where each coordinate is drawn independently from some Beta-Binomial model. The advantage of the approach is that it can be computed very efficiently while only maintaining minimal statistics of \( X \).

In a nutshell, the algorithmic operations performed in the Beta-Binomial model are as follows:

\[
s(x|X) = \mathbf{1}^T \left[ \sigma \left( \sum_{m=1}^{M} \phi(x_m) + \phi(x) \right) - \sigma \left( \phi(x) + \beta \right) \right]
\tag{2.43}
\]

In other words, we sum over statistics of the candidates \( x_m \), add a bias term \( \beta \), perform a coordinate-wise nonlinear transform over the aggregate statistic (in our case a logarithm), and finally we aggregate over the so-obtained scores, weighing each contribution equally. \( s(X) \) is expressed analogously.

2.6.1.2 Gauss Inverse Wishart Model

Before abstracting away the probabilistic properties of the model, it is worth paying some attention to the case where we assume that \( x_i \sim N(\mu, \Sigma) \) and \( (\mu, \Sigma) \sim \text{NIW}(\mu_0, \Sigma, \Psi, \nu) \), for a suitable set of conjugate parameters. While the details are (arguably) tedious, the overall structure of the model is instructive.
First note that the sufficient statistic of the data \( x \in \mathbb{R}^d \) is now given by \( \phi(x) = (x, xx^\top) \). Secondly, note that the conjugate log-partition function \( h \) amounts to computing determinants of terms involving \( \sum_m x_m x_m^\top \) and moreover, nonlinear combinations of the latter with \( \sum_m x_m \).

The algorithmic operations performed in the Gauss Inverse Wishart model are as follows:

\[
s(x|X) = \sigma \left( \sum_{m=1}^{M} \phi(x_m) + \phi(x) + \beta \right) - \sigma (\phi(x) + \beta)
\]

Here \( \sigma \) is a nontrivial convex function acting on a (matrix, vector) pair and \( \phi(x) \) is no longer a trivial map but performs a nonlinear dimension altering transformation on \( x \). We will use this general template to fashion the DeepSets algorithm.

### 2.6.2 Using DeepSets

Using DeepSets, we can solve this problem in more generality as we can drop the assumption of data belonging to certain exponential family.

For learning the score \( s(x|X) \), we take recourse to large-margin classification with structured loss functions (Taskar, Guestrin, and Koller, 2004) to obtain the relative loss objective

\[
l(x, x'|X) = \max(0, s(x'|X) - s(x|X) + \Delta(x, x')).
\]

In other words, we want to ensure that \( s(x|X) \geq s(x'|X) + \Delta(x, x') \) whenever \( x \) should be added and \( x' \) should not be added to \( X \).

**Conditioning** Often machine learning problems do not exist in isolation. For example, task like tag completion from a given set of tags is usually related to an object \( z \), for example an image, that needs to be tagged. Such meta-data are usually abundant, e.g. author information in case of text, contextual data such as the user click history, or extra information collected with LiDAR point cloud.

Conditioning graphical models with meta-data is often complicated. For instance, in the Beta-Binomial model we need to ensure that the counts are always nonnegative, regardless of \( z \). Fortunately, DeepSets does not suffer from such complications and the fusion of multiple sources of data can be done in a relatively straightforward manner. Any of the existing methods in deep learning, including feature concatenation by averaging, or by max-pooling, can be employed. Incorporating these meta-data often leads to significantly improved performance as will be shown in experiments; Section 2.7.2.

## 2.7 Experiments on Set Expansion

### 2.7.1 Text Concept Set Retrieval

We consider the task of text concept set retrieval, where the objective is to retrieve words belonging to a ‘concept’ or ‘cluster’, given few words from that particular concept. For example, given the set of words \{tiger, lion, cheetah\}, we would need to retrieve other related words like jaguar, puma, etc, which belong to the same concept of big cats. The model implicitly needs to reason out the concept connecting the given set and then retrieve words based on their relevance to the inferred concept. Concept set retrieval is an important due to wide range of potential applications including personalized information retrieval, tagging large amounts of unlabeled or weakly labeled datasets, etc. This task of concept set retrieval can be seen as a set completion task conditioned on the latent semantic concept, and therefore our DeepSets form a desirable approach.
Table 2.2: Results on Text Concept Set Retrieval on LDA-1k, LDA-3k, and LDA-5k. Our DeepSets model outperforms other methods on LDA-3k and LDA-5k. However, all neural network based methods have inferior performance to w2v-Near baseline on LDA-1k, possibly due to small data size. Higher the better for recall@k and mean reciprocal rank (MRR). Lower the better for median rank (Med.).

<table>
<thead>
<tr>
<th>Method</th>
<th>LDA-1k (Vocab = 17k)</th>
<th>LDA-3k (Vocab = 38k)</th>
<th>LDA-5k (Vocab = 61k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.06</td>
<td>0.6</td>
<td>5.9</td>
</tr>
<tr>
<td>Bayes Set</td>
<td>1.69</td>
<td>11.9</td>
<td>37.2</td>
</tr>
<tr>
<td>w2v Near</td>
<td><strong>6.00</strong></td>
<td><strong>28.1</strong></td>
<td><strong>54.7</strong></td>
</tr>
<tr>
<td>NN-max</td>
<td>4.78</td>
<td>22.5</td>
<td>53.1</td>
</tr>
<tr>
<td>NN-sum-con</td>
<td>4.58</td>
<td>19.8</td>
<td>48.5</td>
</tr>
<tr>
<td>NN-max-con</td>
<td>3.36</td>
<td>16.9</td>
<td>46.6</td>
</tr>
<tr>
<td>DeepSets</td>
<td>5.53</td>
<td>24.2</td>
<td>54.3</td>
</tr>
</tbody>
</table>

DATASET To construct a large dataset containing sets of related words, we make use of Wikipedia text due to its huge vocabulary and concept coverage. First, we run topic modeling on publicly available wikipedia text with K number of topics. Specifically, we use the famous latent Dirichlet allocation (D. M. Blei, A. Y. Ng, Michael I. Jordan, and Lafferty, 2003; Pritchard, Stephens, and Donnelly, 2000), taken out-of-the-box. Next, we choose top $N_T = 50$ words for each latent topic as a set giving a total of $K$ sets of size $N_T$. To compare across scales, we consider three values of $K = \{1k, 3k, 5k\}$ giving us three datasets LDA-1k, LDA-3k, and LDA-5k, with corresponding vocabulary sizes of 17k, 38k, and 61k. Few of the topics from LDA-1k are visualized in Table 2.11.

METHODS Our DeepSets model uses a feedforward neural network (NN) to represent a query and each element of a set, i.e., $\phi(x)$ for an element $x$ is encoded as a NN. Specifically, $\phi(x)$ represents each word via 50-dimensional embeddings that are learned jointly, followed by two fully connected layers of size 150, with ReLU activations. We then construct a set representation or feature, by sum pooling all the individual representations of its elements, along with that of the query. Note that this sum pooling achieves permutation invariance, a crucial property of our DeepSets (Theorem 2.5). Next, use input this set feature into another NN to assign a single score to the set, shown as $\rho(.)$. We instantiate $\rho(.)$ as three fully connected layers of sizes $\{150, 75, 1\}$ with ReLU activations. In summary, our DeepSets consists of two neural networks – (a) to extract representations for each element, and (b) to score a set after pooling representations of its elements.

BASELINES We compare to several baselines: (a) Random picks a word from the vocabulary uniformly at random. (b) Bayes Set (Ghahramani and Heller, 2005), and (c) w2v-Near that computes the nearest neighbors in the word2vec (Mikolov, Sutskever, et al., 2013b) space. Note that both Bayes Set and w2v NN are strong baselines. The former runs Bayesian inference using Beta-Binomial conjugate pair, while the latter uses the powerful 300 dimensional word2vec trained on the billion word GoogleNews corpus. (d) NN-max uses a similar architecture as our DeepSets with an important difference. It uses max pooling to compute the set feature.

---

2 github.com/dmlc/experimental-lda
3 code.google.com/archive/p/word2vec/
as opposed to DeepSets which uses sum pooling. (e) \textbf{NN-max-con} uses max pooling on set elements but concatenates this pooled representation with that of query for a final set feature. (f) \textbf{NN-sum-con} is similar to NN-max-con but uses sum pooling followed by concatenation with query representation.

\textbf{Evaluation} To quantitatively evaluate, we consider the standard retrieval metrics – recall@K, median rank and mean reciprocal rank. To elaborate, recall@K measures the number of true labels that were recovered in the top K retrieved words. We use three values of K = \{10, 100, 1k\}. The other two metrics, as the names suggest, are the median and mean of reciprocals of the true label ranks, respectively. Each dataset is split into TRAIN (80\%), VAL (10\%) and TEST (10\%). We learn models using TRAIN and evaluate on TEST, while VAL is used for hyperparameter selection and early stopping.

\textbf{Results and Observations} Table 2.2 contains the results for the text concept set retrieval on LDA-1k, LDA-3k, and LDA-5k datasets. We summarize our findings below: (a) Our /deepsets model outperforms all other approaches on LDA-3k and LDA-5k by any metric, highlighting the significance of permutation invariance property. For instance, /deepsets is better than the w2v-Near baseline by 1.5\% in Recall@10 on LDA-5k. (b) On LDA-1k, neural network based models do not perform well when compared to w2v-Near. We hypothesize that this is due to small size of the dataset insufficient to train a high capacity neural network, while w2v-Near has been trained on a billion word corpus. Nevertheless, our approach comes the closest to w2v-Near amongst other approaches, and is only 0.5\% lower by Recall@10.

<table>
<thead>
<tr>
<th>Topic 1</th>
<th>Topic 2</th>
<th>Topic 3</th>
<th>Topic 4</th>
<th>Topic 5</th>
<th>Topic 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>legend</td>
<td>president</td>
<td>plan</td>
<td>newspaper</td>
<td>round</td>
<td>point</td>
</tr>
<tr>
<td>airy</td>
<td>vice</td>
<td>proposed</td>
<td>daily</td>
<td>teams</td>
<td>angle</td>
</tr>
<tr>
<td>tale</td>
<td>served</td>
<td>plans</td>
<td>paper</td>
<td>final</td>
<td>axis</td>
</tr>
<tr>
<td>witch</td>
<td>office</td>
<td>proposal</td>
<td>news</td>
<td>played</td>
<td>plane</td>
</tr>
<tr>
<td>devil</td>
<td>elected</td>
<td>planning</td>
<td>press</td>
<td>redirect</td>
<td>direction</td>
</tr>
<tr>
<td>giant</td>
<td>secretary</td>
<td>approved</td>
<td>published</td>
<td>won</td>
<td>distance</td>
</tr>
<tr>
<td>story</td>
<td>presidency</td>
<td>planned</td>
<td>newspapers</td>
<td>competition</td>
<td>surface</td>
</tr>
<tr>
<td>folklore</td>
<td>presidential</td>
<td>development</td>
<td>editor</td>
<td>tournament</td>
<td>curve</td>
</tr>
</tbody>
</table>

Figure 2.11: Examples from our LDA-1k datasets. Notice that each of these are latent topics of LDA and hence are semantically similar.

2.7.2 \textbf{Image Tagging}

We next experiment with image tagging, where the task is to retrieve all relevant tags corresponding to an image. Images usually have only a subset of relevant tags, therefore predicting other tags can help enrich information that can further be leveraged in a downstream supervised task. In our setup, we learn to predict tags by conditioning /deepsets on the image. Specifically, we train by learning to predict a partial set of tags from the image and remaining tags. At test time, we the test image is used to predict relevant tags.

\textbf{Datasets} We report results on the following three datasets:
(a) \textit{ESPgame} (Von Ahn and Dabbish, 2004): Contains around 20k images spanning logos, drawings, and personal photos, collected interactively as part of a game. There are a total of 268 unique tags, with each image having 4.6 tags on average and a maximum of 15 tags.
<table>
<thead>
<tr>
<th>Method</th>
<th>ESP game</th>
<th>IAPRTC-12.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P  R  F1 N+</td>
<td>P  R  F1 N+</td>
</tr>
<tr>
<td>Least Sq.</td>
<td>35 19 25 215</td>
<td>40 19 26 198</td>
</tr>
<tr>
<td>MBRM</td>
<td>18 19 18 209</td>
<td>24 23 23 223</td>
</tr>
<tr>
<td>JEC</td>
<td>24 19 21 222</td>
<td>29 19 23 211</td>
</tr>
<tr>
<td>FastTag</td>
<td>46 22 30 247</td>
<td>47 26 34 280</td>
</tr>
<tr>
<td>Least Sq.(D)</td>
<td>44 32 37 232</td>
<td>46 36 36 218</td>
</tr>
<tr>
<td>FastTag(D)</td>
<td>44 32 37 229</td>
<td>46 33 38 254</td>
</tr>
<tr>
<td>DeepSets</td>
<td>39 34 36 246</td>
<td>42 31 36 247</td>
</tr>
</tbody>
</table>

Table 2.3: Results of image tagging on ESPgame and IAPRTC-12.5 datasets. Performance of our DeepSets approach is roughly similar to the best competing approaches, except for precision. Refer text for more details. Higher the better for all metrics – precision (P), recall (R), f1 score (F1), and number of non-zero recall tags (N+).

(b) IAPRTC-12.5 (Grubinger, 2007): Comprises of around 20k images including pictures of different sports and actions, photographs of people, animals, cities, landscapes, and many other aspects of contemporary life. A total of 291 unique tags have been extracted from captions for the images. For the above two datasets, train/test splits are similar to those used in previous works (M. Chen, Zheng, and Weinberger, 2013; Guillaumin et al., 2009).

(c) COCO-Tag: We also construct a dataset in-house, based on MSCOCO dataset (T.-Y. Lin et al., 2014). COCO is a large image dataset containing around 80k train and 40k test images, along with five caption annotations. We extract tags by first running a standard spell checker and lemmatizing these captions. Stopwords and numbers are removed from the set of extracted tags. Each image has 15.9 tags on an average and a maximum of 46 tags. We show examples of image tags from COCO-Tag in Figure 2.12. The advantages of using COCO-Tag are three fold–richer concepts, larger vocabulary and more tags per image, making this an ideal dataset to learn image tagging using /deepssets.

**Image and Word Embeddings**

Our models use features extracted from Resnet, which is the state-of-the-art convolutional neural network (CNN) on ImageNet 1000 categories dataset using the publicly available 152-layer pretrained model. To represent words, we jointly learn embeddings with the rest of /deepssets neural network for ESPgame and IAPRTC-12.5 datasets. But for COCO-Tag, we bootstrap from 300 dimensional word2vec embeddings as the vocabulary for COCO-Tag is significantly larger than both ESPgame and IAPRTC-12.5 (13k vs 0.3k).

**Methods**

The setup for DeepSets to tag images is similar to that described in Section 2.7.1. The only difference being the conditioning on the image features, which is concatenated with the set feature obtained from pooling individual element representations. In particular, \( \phi(x) \) represents each word via 300-dimensional word2vec embeddings, followed by two fully connected layers of size 300, with ReLU activations, to construct the set representation or features. As mentioned earlier, we concatenate the image features and pass this set features into another NN to assign a single score to the set, shown as \( \rho(\cdot) \). We instantiate \( \rho(\cdot) \) as three fully connected layers of sizes \( \{300, 150, 1\} \) with ReLU activations. The resulting feature forms the new input to a neural network used to score the set, in this case, score the relevance of a tag to the image.

---

4 [http://hunspell.github.io/](http://hunspell.github.io/)
5 [github.com/facebook/fb.resnet.torch](https://github.com/facebook/fb.resnet.torch)
6 [https://code.google.com/p/word2vec/](https://code.google.com/p/word2vec/)
<table>
<thead>
<tr>
<th>Method</th>
<th>Recall @10</th>
<th>Recall @100</th>
<th>Recall @1k</th>
<th>MRR</th>
<th>Med.</th>
</tr>
</thead>
<tbody>
<tr>
<td>w2v NN (blind)</td>
<td>5.6</td>
<td>20.0</td>
<td>54.2</td>
<td>0.021</td>
<td>823</td>
</tr>
<tr>
<td>DeepSets (blind)</td>
<td>9.0</td>
<td>39.2</td>
<td>71.3</td>
<td>0.044</td>
<td>310</td>
</tr>
<tr>
<td>DeepSets</td>
<td>31.4</td>
<td>73.4</td>
<td>95.3</td>
<td>0.131</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 2.4: Results on COCO-Tag dataset. Clearly, DeepSets outperforms other baselines significantly. Higher the better for recall@K and mean reciprocal rank (MRR). Lower the better for median rank (Med).

**Baselines** We perform comparisons against several baselines, previously reported from M. Chen, Zheng, and Weinberger (2013). Specifically, we have Least Sq., a ridge regression model, MBRM (S. L. Feng, Manmatha, and Lavrenko, 2004), JEC (Makadia, Pavlovic, and S. Kumar, 2008) and FastTag (M. Chen, Zheng, and Weinberger, 2013). Note that these methods do not use deep features for images, which could lead to an unfair comparison. As there is no publicly available code for MBRM and JEC, we cannot get performances of these models with Resnet extracted features. However, we report results with deep features for FastTag and Least Sq., using code made available by the authors.

**Evaluation** For ESPgame and IAPRTC-12.5, we follow the evaluation metrics as in Guillaumin et al. (2009) – precision (P), recall (R), F1 score (F1) and number of tags with non-zero recall (N+). Note that these metrics are evaluate for each tag and the mean is reported. We refer to Guillaumin et al. (2009) for further details. For COCO-Tag, however, we use recall@K for three values of $K = \{10, 100, 1000\}$, along with median rank and mean reciprocal rank (see evaluation in Section 2.7.1 for metric details).

**Results and Observations** Table 2.3 contains the results of image tagging on ESPgame and IAPRTC-12.5, and Table 2.4 on COCO-Tag. Here are the key observations from Table 2.3:

(a) The performance of DeepSets is comparable to the best of other approaches on all metrics but precision. (b) Our recall beats the best approach by 2% in ESPgame. On further investigation, we found that /deepsets retrieves more relevant tags, which are not present in list of ground truth tags due to a limited 5 tag annotation. Thus, this takes a toll on precision while gaining on recall, yet yielding improvement in F1. On the larger and richer COCO-Tag, we see that /deepsets approach outperforms other methods comprehensively, as expected. We show qualitative examples in Figure 2.12.

We present examples of our in-house tagging datasets, COCO-Tag in Figure 2.12.

### 2.8 Set Generation

A fundamental problem in machine learning is to develop efficient methods that can learn a generative model of the distribution of the provided training set and capable of generating arbitrary many new sample points from the domain of this distribution (Christopher, 2016). Deep generative models use deep neural networks as a tool for learning complex distributions. These approaches include variational autoencoders (D. Kingma and Welling, 2013), autoregressive models (Oord, Kalchbrenner, and Kavukcuoglu, 2016) and generative adversarial networks (GAN) (Goodfellow et al., 2014). The compelling results have been demonstrated on simple

7 http://www.cse.wustl.edu/~mchen/
Figure 2.12: Qualitative examples of image tagging using DeepSets. *Top row:* Positive examples where most of the retrieved tags are present in the ground truth (brown) or are relevant but not present in the ground truth (green). *Bottom row:* Few failure cases with irrelevant/wrong tags (red). From left to right, (i) Confusion between snowboarding and skiing, (ii) Confusion between back of laptop and refrigerator due to which other tags are kitchen-related, (iii) Hallucination of airplane due to similar shape of surfboard.
data types like images, speech, and text (Denton, Chintala, Fergus, et al., 2015; Finn, Goodfellow, and Levine, 2016; Karras et al., 2017; Lamb et al., 2016; Radford, Metz, and Chintala, 2015; Van Den Oord et al., 2016). Their wide range of applicability was also demonstrated in many important problems, including data augmentation (Salimans et al., 2016), image style transformation (J.-Y. Zhu et al., 2017), image captioning (Kelvin Xu et al., 2015) and art creations (Kang, 2017).

However, there are many unexplored, yet useful complex data type, like sets which has been theme of this chapter. In particular set of 3D points, such as CAD designs, 3D meshes, and point clouds, have recently garnered a lot of attention because various easily accessible sensors like LiDAR on self-driving cars, Kinect for xbox, and face identification sensor on phones. Similar to useful applications of generative models previous works on simple data type, we want to extend to set data.

We propose a deep generative adversarial network for set data, in particular for point clouds (PC-GAN). The proposed architecture learns a stochastic procedure which can draw samples for point clouds without explicitly modeling the underlying density function.

2.8.1 Proposed Method

We begin by considering true generative process for point cloud over objects. Formally, a point cloud for an object $\theta$ is a collection of $n$ low dimensional vectors $X = \{x_1, ..., x_n\}$, where $n$ can be infinite and typically each $x_i \in \mathbb{R}^3$ or $x_i \in \mathbb{R}^2$. Given an object, $\theta$, the points $x_i$ in the point cloud can be considered as i.i.d. samples from $p(x | \theta)$, an unknown latent distribution representing the object $\theta$. The joint likelihood can be expressed as:

$$p(X, \theta) = p(\theta) \prod_{i=1}^{n} p(x_i | \theta)$$  \hspace{1cm} (2.46)
Attempts have been made to characterize these conditional probabilities with parametric models like Gaussian Mixture Models or parametric hierarchical models (Eckart et al., 2015; Jian and Vemuri, 2005; Strom, A. Richardson, and Olson, 2010). However, such approaches have limited success as the point cloud conditional density \( p(x|\theta) \) is highly non-linear and complicated (example of point clouds can be seen in Figure 2.14, 2.17).

With advent of implicit generative models, like GANs (Goodfellow et al., 2014), it is possible to model complicated distributions, but most of the work focus on learning marginal distributions (Arjovsky, Chintala, and Léon Bottou, 2017; Goodfellow et al., 2014) or conditional distribution (Isola et al., 2017; Mirza and Osindero, 2014). Learning marginal distribution \( p(x) \), in case of point clouds, is not of much use because the marginal distribution is quite uninformative as can be seen from Figure 2.13. Another approach can be to model the distribution of the point cloud set together, i.e., \( \{ (x_1^{(1)})_i, \ldots, (x_1^{(m)})_i \} \). But this would require either treating them as finite dimensional vector by fixing number and order of the points or using complicated objectives like chamfer distance, which did not find much success (Achlioptas et al., 2017). On the other hand, conditional GAN (Mirza and Osindero, 2014) often require conditional variable to be observed, such as labels. While in case of point clouds we only have partial knowledge of the conditional, i.e. we only have groupings of point coming from the same object but we have no representation of the conditional or the object other than the points themselves.

We present a solution to this problem by learning a two-level implicit generative models for modeling the sampling process of \( p(x|\theta) \) and \( p(\theta) \). In Isola et al. (2017) and Mirza and Osindero (2014), the \( \theta \) is a simple one-hot vector of class labels or a given image for transformations. However, naively modeling \( \theta \) to be a one-hot vector to indicate which object can not be generalized to unseen testing data. Instead, different from Isola et al. (2017) and Mirza and Osindero (2014), our \( \theta \) is an unobserved latent variable for modeling different objects with semantic meaning.

Our solution comprises of a generator \( G_x(z, \psi) \) which takes in a noise source \( z \in \mathbb{R}^{d_1} \) and a descriptor \( \psi \in \mathbb{R}^{d_1} \) encoding information about distribution of \( \theta \). For a given \( \theta_0 \), the descriptor \( \psi \) would encode information about the distribution \( \delta(\theta \cdot \theta_0) \) and samples generated as \( x = G_x(z, \psi) \) would follow the distribution \( p(x|\theta_0) \). More generally, \( \psi \) can used to encode more complicated distributions regarding \( \theta \) as well. In particular, it could be used to encode the posterior \( p(\theta|X) \) for a given sample set \( X \) in order to produce samples that follow the posterior predictive distribution:

\[
p(x|X) = \int p(x|\theta)p(\theta|X)d\theta. \tag{2.47}
\]

We can train such a generator \( G_x \) using a GAN-like objective, for which we would need two more components. First, we would need a discriminator \( D_\theta(\cdot) \) which distinguishes between the generated samples and true samples conditioned on \( \theta \). Second, as \( \psi \) is unobserved, we would need an inference network \( Q(X) \) which would try to predict the descriptor \( \psi \) about the distribution \( p(\theta|X) \) given the sample set \( X \). Under this formulation, if we use an IPM-based GAN (Arjovsky, Chintala, and Léon Bottou, 2017; Mroueh, C.-L. Li, et al., 2017; Mroueh and Sercu, 2017), the objective can be written as

\[
\mathbb{E}_{\theta \sim p(\theta)} \left[ \min_{D_\theta \in \Omega_{D_\theta}} \max_{G_x, Q} \mathbb{E}_{X \sim p(X|\theta)} [D_\theta(X)] - \mathbb{E}_{z \sim p(z), X \sim p(X|\theta)} [D_\theta(G_x(z, Q(X)))] \right] \tag{2.48}
\]
where $\Omega_D$ is the constraint for different probabilistic distances, such as 1-Lipschitz (Arjovsky, Chintala, and Léon Bottou, 2017), $L^2$ ball (Mroueh and Sercu, 2017) or Sobolev ball (Mroueh, C.-L. Li, et al., 2017).

A major hurdle in taking this path is that $X$ is a set of points, which can vary in size and permutation of elements. Thus, making design of $Q$ complicated as traditional neural network can not handle this and possibly is the reason for absence of such framework in the literature despite being a natural solution for the important problem of generative modeling of point clouds. However, we can overcome this challenge and We propose to construct the inference network by utilizing the permutation equivariant layers from DeepSets as presented in Section 2.4.1.2. This allows it handle variable number of inputs points in arbitrary order, yet yielding a consistent descriptor $\psi$.

After training $G_x(\cdot, \theta)$ and the inference network $Q$, we use trained $Q$ to collect inferred $\psi = Q(x)$ and train another generator $G_{\psi}(z)$ with a different noise source $z \in \mathbb{R}^d$ such that samples correspond to $p(\theta)$ to enable hierarchical sampling. Moreover, the two generators can be combined and viewed as $G(z_1, z_2) = G_x(z_2, G_{\psi}(z_1))$ for carrying out the hierarchical sampling at one go. In addition to the layer-wise training, a joint training may further boost the performance.

### 2.8.2 Tighter Solutions via Sandwiching

In our setting each point $x_i$ in the point cloud can be considered to correspond to single images in case of training GANs over images. When multiple images are sampled from a GAN on images, many great images are generated, but still some not so good images can be generated. An example is illustrated in Figure 2.15 where samples from MMD-GAN (C.-L. Li et al., 2017) trained on celebA faces consists of both good and bad faces. In case of images, when quality is evaluated, it primarily focuses on coherence individual images and the few bad ones can easily be left out. Whereas in case of point cloud, to get representation of an object we need many points together and presence of outlier points degrades the quality of the object. Thus, when training a generative model for point cloud, we need to ensure a much lower distance $D(P||P_G)$ between true distribution $P$ and generator distribution $P_G$ than would be needed in case of images.

We begin by noting that the popular Wasserstein GAN (Arjovsky, Chintala, and Léon Bottou, 2017), tries to optimize $G$ for reducing the Wasserstein distance $w(P,G)$ between the truth $P$ and generated distribution $G$, i.e. one aims to solve

$$
\min_G w(P, G) \quad (2.49)
$$

Direct evaluation of wasserstein distance is computationally very expensive in general (Arjovsky, Chintala, and Léon Bottou, 2017). To practically solve this problem, the dual form of Wasserstein distance, is invoked, which is

$$
w(P, G) = \sup_{\|f\|_L \leq 1} E_{x \sim P} f(x) - E_{x \sim G} f(x), \quad (2.50)
$$

where $\|f\|_L \leq 1$ is the set of functions whose Lipschitz constant is no larger than 1. Then in practice, deep neural networks with parameters $\phi$ and representing function $f_\phi$ with weight...
clipping (Arjovsky, Chintala, and Léon Bottou, 2017) or gradient penalty (Gulrajani et al., 2017) have been used to approximate \( \|f\|_L \leq 1 \), i.e. solve
\[
\hat{w}(P, G) = \max_{\Phi} \mathbb{E}_{x \sim P} f_\Phi(x) - \mathbb{E}_{x \sim G} f_\Phi(x)
\] (2.51)

But this yields a lower bound on the Wasserstein distance, as \( \hat{w}(P, G) \leq w(P, G) \). When solving a minimization problem, it is known that optimizing the lower bound is not an ideal idea always.

On the other hand, there have been developed \( 1 + \epsilon \) approximations \( w_\epsilon(P, G) \) to Wasserstein distance between to sample sets \( X \) and \( Y \), where
\[
w(P, G) \leq w_\epsilon(P, G) \leq (1 + \epsilon)w(P, G),
\]
which is an upper bound to Wasserstein distance (Bertsekas, 1985; H. Fan, Hao Su, and Guibas, n.d.). We propose to combine both the lower bound and upper bound, in order to sandwich the solution between the two, i.e. we solve the following minimization problem:
\[
\min_G w_\epsilon(P, G)
\]
\[
s.t. \quad w_\epsilon(P, G) - \hat{w}(P, G) < \lambda
\] (2.52)

This optimization problem be solved using Lagrange multipliers. In our experiments we use this formulation and see significantly better recovery of true distribution.

2.8.3 Simplified Theoretical Justification

We want to understand what happens to GANs under weak discriminators (lower bounds) from statistical perspective. Ignoring the implementation of these models, from the perspective of statistical analysis, generative model using GANs are different from classical density estimators as loss is measured not with \( L^p \) distances (as is conventional in nonparametric statistics (Tsybakov, 2009; Wasserman, 2006)), but rather with weaker losses, such as
\[
d_{T_D}(P, Q) = \sup_{f \in T_D} \left| \mathbb{E}_{X \sim P} [f(X)] - \mathbb{E}_{X \sim Q} [f(X)] \right|
\] (2.53)

where \( T_D \) is a discriminator class of bounded, Lebesgue-measurable functions, and \( P \) and \( Q \) lie in a generator class \( T_G \) of Borel probability measures on a sample space \( X \). To allow probability measures \( P \) without densities (i.e., \( P \ll \mu \)), we assume each basis element \( \phi_z : X \to \mathbb{R} \) is a bounded function, so that \( \hat{P}_z := \mathbb{E}_{X \sim P} [\phi_z(X)] \) is well-defined for a fixed orthonormal basis \( B = \{\phi_z\}_{z \in \mathcal{Z}} \) of \( L^2_\mu \) indexed by a countable family \( \mathcal{Z} \).

We want to upper bound the adversarial risk of the following density estimator. Define, for constants \( L > 0 \) and \( p \geq 1 \) and real-valued net \( \{a_z\}_{z \in \mathcal{Z}} \), generalized ellipses of the form
\[
\mathcal{H}_{p, a}(L) = \left\{ f \in L^1(X) : \left( \sum_{z \in \mathcal{Z}} a_z^p |f_z|^p \right)^{1/p} \leq L \right\}
\] (2.54)

(where \( f_z := \int_X f \phi_z \, d\mu \) is the \( z^{th} \) coefficient of \( f \) in the basis \( B \)). For any finite set \( Z \subseteq \mathcal{Z} \), let \( \hat{P}_Z \) be the truncated series estimate
\[
\hat{P}_Z := \sum_{z \in Z} \hat{P}_z \phi_z, \quad \text{where, for any } z \in \mathcal{Z}, \quad \hat{P}_z := \frac{1}{n} \sum_{i=1}^n \phi_z(X_i).
\] (2.55)
We now present our main upper bound on the minimax rate of density estimation under adversarial losses. The upper bound is given for the orthogonal series estimator given in Equation (2.55), but we expect kernel and other standard linear density estimators to converge at the same rate.

**Theorem 2.13** Fix constants \( L_D, L_G > 0 \), \( p, q < 2 \) and real-valued families \( \{a_z\}_{z \in \mathbb{Z}} \) and \( \{b_z\}_{z \in \mathbb{Z}} \). Suppose \( \mathcal{F}_D = \mathcal{H}_{p,a}(L_D) \) and \( \mathcal{F}_G = \mathcal{H}_{q,b}(L_G) \). Then, for all \( P \in \mathcal{F}_G \) and \( Z \subseteq \mathbb{Z} \),

\[
\mathbb{E}_{X_1, \ldots, X_n \sim P}[d_{\mathcal{F}_D}(P, \hat{P})] \leq \frac{L_D}{\sqrt{n}} \left( \sum_{z \in \mathbb{Z}} \frac{\|f_z\|_{L_p}^2}{a_z^2} \right)^{1/2} + L_D L_G \left( \sup_{z' \in \mathbb{Z} \setminus Z} a_z^{-1} \right) \left( \sup_{z' \in \mathbb{Z} \setminus Z} b_{z'}^{-1} \right).
\]

**Proof** One can check, using the fact that \( \mathcal{F}_D \) is ortho-symmetric (i.e. for every \( f \in \mathcal{F}_D \) and \( Z \subseteq \mathbb{Z} \), \( f - 2 \sum_{z \in Z} f_z \phi_z \in \mathcal{F}_D \), so that \( \mathcal{F}_D \) is symmetric across every axis defined by the basis \( \mathcal{B} \)), Lebesgue’s monotone convergence theorem, and the assumption that each \( f \in \mathcal{F}_D \) is bounded, that, for all probability measures \( P \) and \( Q \),

\[
\mathbb{E}_{X_1, \ldots, X_n}[d_{\mathcal{F}_D}(P, Q)] \leq \mathbb{E}_{X_1, \ldots, X_n} \left[ \sup_{f \in \mathcal{F}_D} \sum_{z \in \mathbb{Z}} |f_z(P_z - Q_z)| \right].
\]

Therefore,

\[
\mathbb{E}_{X_1, \ldots, X_n}[d_{\mathcal{F}_D}(P, \hat{P})] \leq \mathbb{E}_{X_1, \ldots, X_n} \left[ \sup_{f \in \mathcal{F}_D} \sum_{z \in \mathbb{Z}} |f_z(\hat{P}_z - \hat{P}_z)| \right]
\]

\[
= \mathbb{E}_{X_1, \ldots, X_n} \left[ \sup_{f \in \mathcal{F}_D} \sum_{z \in \mathbb{Z}} |f_z(\hat{P}_z - \hat{P}_z)| + \sum_{z \in \mathbb{Z} \setminus Z} |f_z(\hat{P}_z - \hat{P}_z)| \right]
\]

\[
= \mathbb{E}_{X_1, \ldots, X_n} \left[ \sup_{f \in \mathcal{F}_D} \sum_{z \in \mathbb{Z}} |f_z(P_z - \hat{P}_z)| + \sum_{z \in \mathbb{Z} \setminus Z} |f_z\hat{P}_z| \right]
\]

\[
\leq \mathbb{E}_{X_1, \ldots, X_n} \left[ \sup_{f \in \mathcal{F}_D} \sum_{z \in \mathbb{Z}} |f_z(P_z - \hat{P}_z)| \right] + \sup_{f \in \mathcal{F}_D} \sum_{z \in \mathbb{Z} \setminus Z} |f_z\hat{P}_z|.
\]

Note that we have decomposed the risk into two terms, the first comprising estimation error (variance) and the second comprising approximation error (bias). Indeed, in the case that \( \mathcal{F}_D = \mathcal{L}^2(\mathcal{X}) \), the above becomes precisely the usual bias-variance decomposition of mean squared error.
To bound the first term, applying the Cauchy-Schwarz inequality, the fact that $f \in \mathcal{F}_D$, and Jensen’s inequality (in that order), we have

$$
\mathbb{E}_{X_1, \ldots, X_n} \left[ \sup_{f \in \mathcal{F}_D} \sum_{z \in Z} |f_z (\hat{P}_z - \hat{P}_z)| \right] = \mathbb{E}_{X_1, \ldots, X_n} \left[ \sup_{f \in \mathcal{F}_D} \sum_{z \in Z} a_z |f_z| \frac{|\hat{P}_z - \hat{P}_z|}{a_z} \right] \\
\leq \mathbb{E}_{X_1, \ldots, X_n} \left[ \sup_{f \in \mathcal{F}_D} \left( \sum_{z \in Z} a_z^2 |f_z|^2 \right)^{1/2} \left( \sum_{z \in Z} \frac{(\hat{P}_z - \hat{P}_z)^2}{a_z^2} \right)^{1/2} \right] \\
\leq \mathcal{L}_D \mathbb{E}_{X_1, \ldots, X_n} \left[ \left( \sum_{z \in Z} \frac{(\hat{P}_z - \hat{P}_z)^2}{a_z^2} \right)^{1/2} \right] \\
\leq \mathcal{L}_D \left( \sum_{z \in Z} \frac{\|\phi_z\|_p^2}{a_z^2} \right)^{1/2} \\
= \frac{\mathcal{L}_D}{\sqrt{n}} \left( \sum_{z \in Z} \frac{\|\phi_z\|_p^2}{a_z^2} \right)^{1/2},
$$

since

$$
\mathbb{E}_{X_1, \ldots, X_n} \left[ (\hat{P}_z - \hat{P}_z)^2 \right] = \text{Var}_{X_1, \ldots, X_n} \left[ \frac{1}{n} \sum_{i=1}^{n} \phi_z(X_i) \right] \leq \frac{\|\phi_z\|_p^2}{n}.
$$

For the second term, by the Cauchy-Schwarz inequality,

$$
\sup_{f \in \mathcal{F}_D} \sum_{z \in Z \setminus Z} |f_z| \leq \sup_{f \in \mathcal{F}_D} \left( \sum_{z \in Z \setminus Z} |f_z|^2 \right)^{1/2} \left( \sum_{z \in Z \setminus Z} \frac{\hat{P}_z}{a_z^2} \right)^{1/2} \\
\leq \sup_{f \in \mathcal{F}_D} \left( \sum_{z \in Z \setminus Z} \frac{a_z^2 |f_z|^2}{\inf_{z' \in Z \setminus Z} a_{z'}^2} \right)^{1/2} \left( \sum_{z \in Z \setminus Z} \frac{b_z^2 |\hat{P}_z|^2}{\inf_{z' \in Z \setminus Z} b_{z'}^2} \right)^{1/2} \\
\leq \mathcal{L}_D \mathcal{L}_G \left( \sup_{z' \in Z \setminus Z} a_{z'}^{-1} \right) \left( \sup_{z' \in Z \setminus Z} b_{z'}^{-1} \right)
$$

The two terms in the bound (2.56) demonstrate a bias-variance tradeoff, in which the first term (variance) increases with the truncation set $Z$ and is typically independent of the class $\mathcal{F}_G$ of distributions, while the second term (bias) decreases with $Z$ at a rate depending on the complexity of $\mathcal{F}_G$. Thus, if discriminator is weak, we might obtain $d_{\mathcal{F}_D}(P, Q)$ to be 0, while in reality it is not zero. Thus, combining GAN objective with an upper bound as well as presented in Section 2.8.2 would help in strengthening the discriminating and producing better samples.

### 2.9 Experiments on set generation

In this section we demonstrate the point cloud generation capabilities of PC-GAN and compare it with some of the existing methods. We train the proposed PC-GAN on synthetic 3D point
cloud and ModelNet40 benchmark datasets. The method of Achlioptas et al. (2017) could be treated as an AAE extension to point clouds, therefore we call it AAE in the discussion below. We use the implementation provided by Achlioptas et al. (2017) to train AAE with the approximated EMD objective (Bertsekas, 1985). For AAE, the decoder is a MLP, where the output is \#points \times point dimensions.

For PC-GAN, we combine the objective of Fisher GAN and Approximated Wasserstein distance (Bertsekas, 1985) with mixture 1:20 without tuning. We use Fisher GAN (Mroueh and Sercu, 2017) to replace WGAN-GP (Gulrajani et al., 2017). We found that this way the performance is similar to WGAN-GP, but the training is more stable and faster.

2.9.1 Synthetic Datasets

We created a simple 2D synthetic point cloud datasets from parametric distributions on which we can carry out thorough evaluations of our model and draw comparisons with Achlioptas et al. (2017) (AAE). We generate 2D point clouds for circles, where the center of circles is followed a mixture of four Gaussians with means equal to \{±16\} \times \{±16\}. The variances were set to be 16I and we used equal mixture weights. The radius of the circles was drawn from a uniform distribution \text{Unif}(1.6, 6.4) as shown in Figure 2.16a. We sampled 10,000 circles for the training and testing data, respectively.

For the proposed PC-GAN, the architecture is a stack of 3 Permutation Equivariance Layers with size 30. The size of the latent variables \theta in (2.46) was set to be 15. In this experiment the total number of parameters for PC-GAN was 12K. For AAE encoder, we follow the same network architecture used in Achlioptas et al. (2017). Here the decoder is a 4-layer MLP where the output is 500 \times 2 dimensions for 500 points. We consider two model configurations AAE-10 and AAE-20, which use 10 and 20 units for the hidden layers of the decoder, respectively. The total number of parameters (encoder+decoder) are 14K and 24K for AAE-10 and AAE-20, respectively. Detailed model configurations are provided in the supplementary material.

We evaluated the conditional distributions on the 10,000 testing circles. For the proposed PC-GAN, we pass the same points into the inference network \(Q\), then sample 500 points with the conditional generator \(G_x\) to match the output number of AAE. We measured the empirical distributions of the centers and the radius of the generated circles conditioning on the testing data for PC-GAN. Similarly, we measured the reconstructed circles of the testing data for AAE. The results are shown in Figure 2.16.

As we can see from the results, the models can successfully recover the center distribution, but AAE does not learn the radius distribution well. Even if we increase number the hidden layer unit to be 20 (AAE-20), which almost doubles the number of parameters, the performance is still not satisfactory. Compared with AAE, the proposed PC-GAN recovers the both center and radius distributions well with less parameters. The gap of memory usage could be larger if we configure AAE to generate more points, while the model size required for PC-GAN is independent of the number of points. The reason is MLP decoder adopted by Achlioptas et al. (2017) wasted parameters for nearby points. Using the more advanced point cloud autoencoder (Yang et al., 2018) may resolve the issue. However, it is still restricted to generate a fixed number of points for each object.

\[ \text{https://github.com/optas/latent_3d_points} \]
Figure 2.16: The reconstructed center and radius distributions. (a) (top) the true center distribution and (bottom) one example of the 2D circle point cloud. (b-d) are the reconstructed center and radius distributions of different algorithms.

2.9.2 ModelNet40

We consider the ModelNet40 (Z. Wu et al., 2015) benchmark, which contains 40 classes of objects. There are 9,843 training and 2,468 testing instances. We follow Zaheer, Kottur, et al. (2017) to do pre-processing. For each object, we sampled 10,000 points from the mesh representation and normalize it to have zero mean (for each axis) and unit (global) variance. During the training, we augment the data by uniformly rotating $0, \pi/8, \ldots, 7\pi/8$ rad on the x-y plane. For PC-GAN, the random noise $z_2$ is fixed to be 10 dimensional for all experiments. For other settings, we follow Achlioptas et al. (2017).

Training on single class. We start from a smaller model which is only trained on single class of objects. For AAE, the latent code size for its encoder is 128 and the decoder outputs 2,048 points for each object. The number of parameters for encoder and decoder are 15M in total. Therefore, we set the size of PC-GAN latent variable to be 128 dimensional. The number of parameters for G and G-inv is less than 1M in total.

Training on all classes. We also train the proposed model on all 9,843 objects in the training set. The size of AAE latent code of is increased to be 256. The number of parameters of its encoder and decoder is 15.2M. we set the size of PC-GAN latent variable to be 256 dimensional. The number of parameters for G and G-inv is around 3M in total.

2.9.2.1 Conditional Generation Results

We first evaluate the performance of trained conditional generator and the inference network. We are interested in whether the learned model can model the distribution of the unseen testing data. Therefore, for each testing point cloud, we use the inference network to infer the latent variable $x$, then use the conditional generator to generate points. We then measure the distribution between the input point cloud and the generated point cloud.

There are many criteria based on finite sample estimation for evaluation, such $f$-divergence and IPM. However, the estimator with finite samples are either biased or with high variance (Peyré, Cuturi, et al., 2017; Póczos, Liang Xiong, and Jeff Schneider, 2012; Q. Wang, Kulkarni, and
Figure 2.17: The reconstruction on test objects from seen categories. For each object, from left to right is training data, AAE, and PC-GAN. PC-GAN is better in capturing fine details like wheels of aeroplane or proper chair legs.

Table 2.5: The quantitative results of different models trained on different subsets of ModelNet40. MultiNet10 is a subset containing 10 classes of objects, while MultiNet40 is a full training set.

<table>
<thead>
<tr>
<th>Data</th>
<th>Distance</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AAE</td>
<td>Approx</td>
</tr>
<tr>
<td>Aeroplanes</td>
<td>1.99E+01</td>
<td>1.53E+01</td>
</tr>
<tr>
<td>Benches</td>
<td>1.41E+01</td>
<td>1.05E+01</td>
</tr>
<tr>
<td>Cars</td>
<td>6.23E+01</td>
<td>4.25E+01</td>
</tr>
<tr>
<td>Chairs</td>
<td>1.08E+01</td>
<td>1.06E+01</td>
</tr>
<tr>
<td>Cups</td>
<td>1.79E+03</td>
<td>1.28E+03</td>
</tr>
<tr>
<td>Guitars</td>
<td>1.93E+02</td>
<td>1.97E+02</td>
</tr>
<tr>
<td>Lamps</td>
<td>1.60E+03</td>
<td>1.64E+03</td>
</tr>
<tr>
<td>Laptops</td>
<td>3.73E+00</td>
<td>2.65E+00</td>
</tr>
<tr>
<td>Sofa</td>
<td>1.64E+01</td>
<td>1.45E+01</td>
</tr>
<tr>
<td>Tables</td>
<td>2.96E+00</td>
<td>2.44E+00</td>
</tr>
</tbody>
</table>

ModelNet10 6.89E+00 6.03E+00 9.19E+00 5.77E+00 1.90E-01 3.36E-01 3.67E-01 3.47E-01
ModelNet40 5.86E+01 5.24E+01 7.96E+01 4.84E+01 1.85E-01 3.65E-01 3.71E-01 3.80E-01
Verdú, 2009). Also, it is impossible to use these estimators with infinitely many samples if they are accessible.

For ModelNet40, the meshes of each object are available. In many statistically guaranteed distance estimates, the adopted statistics are commonly based on distance between nearest neighbors (Póczos, Liang, Xiong, and Jeff Schneider, 2012; Q. Wang, Kulkarni, and Verdú, 2009). Therefore, we propose to measure the performance with the following criteria. Given a point cloud \( \{x_i\}_{i=1}^n \) and a mesh, which is a collection of faces \( \{F_j\}_{j=1}^m \), we measure

\[
D \left( \{x_i\}_{i=1}^n, \{F_j\}_{j=1}^m \right) = \frac{1}{n} \min_{j} d(x_i, F_j),
\]

where \( d(x_i, F_j) \) is the Euclidean distance from \( x_i \) to the face \( F_j \).

This distance is similar to Chamfer distance, which is commonly used for measuring images and point clouds (Achlioptas et al., 2017; H. Fan, Hao Su, and Guibas, n.d.), with infinitely samples from true distributions (meshes).

On the other hand, mode collapse is commonly happened in GAN algorithm. We are interested in whether the generated points recover enough supports of the distribution. We compute the Coverage ratio as follows. For each points, we find the its nearest face, we then treat this face is covered. We then compute the ratio of number of faces of a mesh is covered. A sampled mesh is showed in Figure 2.18, where we can observe the distribution of mesh is not uniform. Therefore, it is difficult to get high coverage for AAE or PC-GAN, since we only sample limited number of points for training. On the other hand, we can also observe the details of the objects have more meshes. Therefore, the coverage measured based on faces could also serve an indicator for how much details of the objects covered by the generated point cloud.

The results are reported in Table 2.5. We compare four different algorithm, AAE and use three objective for training conditional generators, Approx (approximated Wasserstein distance), GAN (Fisher GAN), and the primal-dual loss by combining Approx and GAN as mentioned in Section 2.8. From Table 2.5, we observe that Approx usually results in smaller distance measure, which suggests the validity of training GAN with primal formulation. However, it loses more details with worse coverage than GAN. However, GAN results in worse distance measure. From the empirical observation, by balancing these two objectives for training a generative model, PC-GAN results in an desirable middle ground. The simple balancing strategy is also theoretically supported from “Sandwiching” objective discussed in Section 2.8.

2.9.2.2 Hierarchical Sampling

In Section 2.8, we propose a hierarchical sampling process for sampling point clouds. In the first hierarchy, the generator XX samples a object, while the second generator sample points to form the point cloud. The randomly sampled results without given any data as input are shown in Figure 2.19. The point clouds are all smooth, structured and almost symmetric. It shows PC-GAN captures inherent symmetries and patterns in all the randomly sampled objects, even if overall object is not perfectly formed. This highlights that learning point-wise generation scheme encourages learning basic building blocks of objects.

We should do thresholding to ignore outlier points. In our experiments, we observe that without excluding outliers does not change conclusion for comparison.
2.9 Experiments on Set Generation

2.9.2.3 Understand the Learned Manifold

Classification We evaluate the quality of the representation acquired from the learned inference network. We train the inference network and the generator on the training split of ModelNet40 with data augmentation as mentioned above. We then extract the latent representation and train linear SVM on the that. We apply the same setting to a linear classifier on the latent code of Achlioptas et al. (2017).

The encoder of Achlioptas et al. (2017) took 2048 points for each obj as input while we only sample 1000 as input for our inference network. Benefited by the Deep Sets architecture for the inference network, which is invariant to number of points. Therefore, we are still allowed to sample more points as input to the trained inference network for evaluation. Because of the randomness of sampling points for extracting latent representation, we repeat the experiments 20 times and report the average accuracy and standard deviation on the testing split in Table 2.6. By using 1000 points, we are already better than Achlioptas et al. (2017) with 2048 points, and competitive with the supervised learning algorithm Deep Sets.

Interpolation A commonly used method to demonstrate the quality of the learned latent space is showing whether the interpolation between two objects on the latent space results in

<table>
<thead>
<tr>
<th>Method</th>
<th># points</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC-GAN</td>
<td>1000</td>
<td>87.5 ± .6%</td>
</tr>
<tr>
<td>PC-GAN</td>
<td>2048</td>
<td>87.8 ± .2%</td>
</tr>
<tr>
<td>AAE (Achlioptas et al., 2017)</td>
<td>2048</td>
<td>85.5 ± .3%</td>
</tr>
<tr>
<td>Supervised DeepSets</td>
<td>1000</td>
<td>87 ± 1%</td>
</tr>
<tr>
<td>Supervised DeepSets</td>
<td>5000</td>
<td>90 ± .3%</td>
</tr>
</tbody>
</table>

Table 2.6: Classification accuracy results.
smooth change. We interpolate the inferred representations from two objects by the inference network, and use the generator to sample points. The inter-class result is shown in Figure 2.20.

It is also popular to show intra-class interpolation. In addition show simple intra-class interpolations, where the objects are almost aligned, we present an interesting study on interpolations between rotations. During the training, we only rotate data with 8 possible angles for augmentation, here we show it generalizes to other unseen rotations as shown in Figure 2.21.

However, if we linearly interpolate the code, the resulted change is scattered and not smooth as shown in Figure 2.21. Instead of using linear interpolation, We train a 2-layer MLP with limited hidden layer size to be 16, where the input is the angle, output is the corresponding latent representation of rotated object. We then generate the code for rotated planes with this trained MLP. It suggests although the transformation path of rotation on the latent space is not linear, it follows a smooth trajectory. It may also suggest the geodesic path of the learned manifold may not be nearly linear between rotations. Finding the geodesic path with a principal method (Shao, A. Kumar, and Fletcher, 2017) and Understanding the geometry of the manifold for point cloud worth more deeper study as future work.

**Generalization on Unseen Categories** In above, we studied the reconstruction of unseen testing objects, while PC-GAN saw the training objective from the same category. Here we study the more challenging task. We train PC-GAN on first 30 (Alphabetic order of category names) categories only, and test on the unseen 10 classes. The mean distance and coverage are 57.4 and 0.36. The reconstructed (conditionally generated) point clouds is shown in Figure 2.22. The results are generalized surprisingly well. For the object from the unseen categories, the conditionally generated point clouds still recovers main shape and reasonable geometry structure, which confirms the advantage of the proposed PC-GAN: by enforcing the point-wise transformation, the model is forced to learn the underlying geometry structure and the shared building blocks, instead of naively copying the input from the conditioning.

**2.10 Summary**

In this paper, we develop DeepSets, a model based on powerful permutation invariance and equivariance properties, along with the theory to support its performance. We demonstrate the generalization ability of DeepSets across several domains by extensive experiments, and show both qualitative and quantitative results. In particular, we explicitly show that DeepSets outperforms other intuitive deep networks, which are not backed by theory (Section 2.7.1, Section 2.5.2). Last but not least, it is worth noting that the state-of-the-art we compare to is a

---

10 By the capability of 1-layer MLP.
specialized technique for each task, whereas our one model, i.e. DeepSets, is competitive across the board.

We also proposed a new hierarchical generative adversarial neural network (PC-GAN) that is capable of learning how to generate samples from objects represented by point clouds. PC-GAN uses ideas both from hierarchical Bayesian modeling and generative adversarial networks.

In contrast to some of the existing methods like Achlioptas et al. (2017), our proposed architecture is a true generative model; it can generate arbitrary many independent and identically distributed sample points form the learned distribution. As we have shown in the paper, PC-GAN can achieve very competitive results using smaller network sizes than existing methods do. We have also demonstrated with a series of numerical experiments, that PC-GAN can capture delicate details of 3D point clouds and can generalize well even on unseen data. Our method learns ‘point-wise’ transformations which can enforce the model to learn the building components of the objects, instead of just naively copying the whole object. PC-GAN can also naturally interpolate in the hidden representations of the 3D objects and can generate arbitrarily many sample points from these artificially created objects.

Although we only focused on 3D applications in this paper, our framework can naturally generalize to higher dimensions as well. In the future we would like to explore higher dimensional applications, where each generated 3D point can have other attributes as well, for example time dimension, RGBA colors, and 3D velocity vectors.

Figure 2.22: The reconstructed objects from unseen categories. In each plot, LHS is true data while RHS is PC-GAN. PC-GAN generalizes well as it can match patterns and symmetries from categories seen in the past to new unseen categories.
Compressed Data

Training robust deep video representations has proven to be much more challenging than learning deep image representations and consequently hampered tasks like video action recognition. This is in part due to the enormous size of raw video streams, the associated amount of computation required, and the high temporal redundancy. The ‘true’ and interesting signal is often drowned in too much irrelevant data. Motivated by the fact that the superfluous information can be reduced by up to two orders of magnitude with video compression techniques (like H.264, HEVC, etc.), in this work, we propose to train a deep network directly on the compressed video, devoid of redundancy, rather than the traditional highly redundant RGB stream.

This representation has a higher information density and we found the training to be easier. In addition, the signals in a compressed video provide free, albeit noisy, motion information. We propose novel techniques to use them effectively. Our approach is about 4.6 times faster than a state-of-the-art 3D-CNN model, 2.7 times faster than a ResNet-152, and very easy to implement. On the task of action recognition, our approach outperforms all the other methods on the UCF-101, HMDB-51, and Charades dataset.

3.1 Introduction

Video nowadays commands the lion’s share of internet traffic at 70% and rising [Index, 2016]. Most cell phone cameras now capture high resolution video streams instead of just images. Many real world data sources for computer vision are video based, ranging from inventory systems at warehouses to self-driving cars or fully autonomous drones. Video is also arguably the next frontier in computer vision. Video captures a wealth of information still images cannot convey. They carry more emotion Gunnar A Sigurdsson et al., 2016, allow us to predict the future to a certain extent Mathieu, Couprie, and LeCun, 2016, provide temporal context and give us better spatial awareness Pollefeys et al., 2008. Unfortunately, very little of this information is currently exploited.

State of the art deep learning models for video analysis are quite basic. Most of them naïvely try to exploit the deep learning based methods that have been highly successful in the image domain by feeding the video into image-based convolutional neural networks (CNNs) frame by frame. They often demonstrate results no better than the traditional techniques Karpathy, Toderici, et al., 2014; H. Wang and Schmid, 2013. So, why did deep learning not yet make as transformative of an impact on video tasks, such as action recognition, as it did on images?

We argue that the reason is three-fold. First of all, videos have a very low information density, as 1h of 720p video can be compressed from 222GB raw to 1GB. In other words, videos are filled with boring and repeating patterns, drowning the ‘true’ and interesting signal. It is not just harder for convolutional neural networks to extract the meaningful information, but also the unnecessary replication of data makes training much slower. Secondly, the input representation as RGB images hinders/complicates the learning of temporal structure. While a single...
still image does not convey motion, stacking multiple RGB frames and feeding them into a 2D CNN also does not help Karpathy, Toderici, et al., 2014. One can use 3D CNNs to model temporal structures, but without additional enforcement of structures, modeling multiple frames is challenging due to the curse of dimensionality.

To address these issues, we exploit the compressed representation developed for storage and transmission of videos rather than operating on the RGB frames (Figure ??). These compression techniques (like MPEG, H.264 etc.) leverage the fact that successive frames are usually very similar and retain only a few frames completely and reconstruct other frames based on offsets from the complete images, called motion vectors and residual error. Our model consists of multiple CNNs, each of which learns to model one kind of representation in a compressed video: images, motion vectors, and residuals. To simplify the dependency in the compressed stream, we propose a novel preprocessing technique that allows training a simple decoupled model which enables parallelism during test time. In particular, we show that even though compressed representation, e.g. from MPEG, does not seem amenable to CNNs directly, our method makes it effective in practice.

Why is this better? First, video compression removes up to two orders of magnitude of superfluous information, making interesting signals prominent. Second, the motion vectors in video compression provide us the motion information that lone RGB images do not have. Furthermore, the motion signals already exclude spatial variations, e.g. two people performing the same action in different clothings or in different lighting conditions exhibit the same motion signals. This improves generalization, and the lowered variance further simplifies training. Third, with compressed video, we account for correlation in video frames, i.e. spatial view plus some small changes over time, instead of i.i.d. images. Constraining data in this structure helps us tackling the curse of dimensionality. Last but not least, our method is also much more efficient as we only look at the true signals instead of repeatedly processing near-duplicates. Efficiency is also gained by avoiding to decompress the video, because video is usually stored or transmitted in the compressed version, and access to the motion vectors and residuals are free.
We demonstrate that our approach on the task of action recognition significantly outperforms all other methods that train on traditional RGB images without using any recurrent neural networks, complicated fusion or 3D convolutions on the UCF-101 Soomro, Zamir, and Shah, 2012, HMDB-51 Kuehne et al., 2011, and Charades Gunnar A Sigurdsson et al., 2016 dataset. In addition, our model is 4.6 times faster than a state-of-the-art 3D CNN model, and 2.7 times faster than ResNet-152. When combined with scores from a standard temporal stream network, our model outperforms state-of-the-art methods on all datasets.

3.2 BACKGROUND

In this section we provide a brief overview about video action recognition and video compression.

3.2.1 Action Recognition

Traditionally, for the task of action recognition on video, the community utilized hand-crafted features, such as Histogram of Oriented Gradients (HOG) Dalal and Triggs, 2005 or Histogram of Optical Flow (HOF) Laptev et al., 2008, both sparsely Laptev et al., 2008 and densely H. Wang, Ullah, et al., 2009 sampled. While early methods consider independent interest points across frames, smarter aggregation based on dense trajectories have been used Peng et al., 2014; H. Wang, Kläser, et al., 2013; H. Wang and Schmid, 2013. Some of these traditional methods are competitive even today, like iDT which corrects for camera motion H. Wang and Schmid, 2013.

In the past few years, deep learning has brought significant improvements to video understanding Donahue et al., 2015; Karpathy, Toderici, et al., 2014. However, the improvements mainly stem from improvements in deep image representations. Temporal structure is still relatively simple — most algorithms subsample a few frames and perform average pooling to make final predictions Simonyan and Zisserman, 2014; L. Wang et al., 2016. Recurrent neural networks (RNNs) Donahue et al., 2015; Yue-Hei Ng et al., 2015, temporal CNNs Ma et al., 2017, or other feature aggregation techniques Girdhar, Ramanan, et al., 2017; L. Wang et al., 2016 on top of CNN features have also been explored. However, while introducing new computation overhead, these methods do not necessarily outperform simple average pooling L. Wang et al., 2016. Some works explore 3D CNNs to model temporal structure Tran, Bourdev, et al., 2015; Tran, Ray, et al., 2017. Nonetheless, it results in an explosion of parameters and computation time and only marginally improves the performance Tran, Ray, et al., 2017.

More importantly, evidence suggests that these methods are not sufficient to capture all temporal structures — using of pre-computed optical flow almost always boosts the performance Carreira and Zisserman, 2017; Feichtenhofer, Pinz, and Zisserman, 2016; Simonyan and Zisserman, 2014; L. Wang et al., 2016. This emphasizes the importance of using the right input representation and the inadequacy of RGB frames. Finally, note that all of these methods require raw video frame-by-frame and cannot exploit the fact that video is stored in some compressed format.

3.2.2 Video Compression

The need for efficient video storage and transmission has led to highly efficient video compression algorithms, such as MPEG, H.264, and HEVC, some of which date back to 1990s Le Gall, 1991. Most video compression algorithms leverage the fact that successive frames are usually
Figure 3.2: We trace all motion vectors back to the reference I-frame and accumulate the residual. Now each P-frame depends only on the I-frame but not other P-frames.

very similar. We can efficiently store one frame by reusing contents from another frame and only store the difference.

Most modern codecs split a video into I-frames (intra-coded frames), P-frames (predictive frames) and zero or more B-frames (bi-directional frames). I-frames are regular images and compressed as such. P-frames reference the previous frames and encode only the ‘change’. A part of the change – termed motion vectors – is represented as the movements of block of pixels from the source frame to the target frame at time \( t \), which we denote by \( \gamma(t) \). Even after this compensation for block movement, there can be difference between the original image and the predicted image at time \( t \), we denote this residual difference by \( \Delta(t) \). Putting it together, a P-frame at time \( t \) only comprises of motion vectors \( \gamma(t) \) and a residual \( \Delta(t) \). This gives the recurrence relation for reconstruction of P-frames as

\[
I_1^{(t)} = I_{1}^{(t-1)} - \gamma_1^{(t)} + \Delta_1^{(t)},
\]

where \( I^{(t)} \) denotes the RGB image at time \( t \) and the subscripts denote the index of spatial location. The motion vectors and the residuals are then passed through a discrete cosine transform (DCT) space, and entropy-encoded.

A B-frame may be viewed as a special P-frame, where motion vectors are computed bi-directionally and may reference a future frame as long as there are no circles in referencing. Both B- and P-frames capture only what changes in the video, and are easier to compress owing to smaller dynamic range. E. Richardson, 2002. See Figure ?? for a visualization of the motion estimates and the residuals. Modeling arbitrary decoding order is beyond the scope of this paper. We focus on videos encoded using only backward references, namely I- and P-frames.

**Features from compressed data.** Some prior works have utilized signals from compressed video for detection or recognition, but only as a non-deep feature Kantorov and Laptev, 2014; Sukmarg and Rao, 2000; Töreyin et al., 2005; Yeo and B. Liu, 1995. To the best of our knowledge, this is the first work that considers training deep networks on compressed videos. MV-CNN apply distillation to transfer knowledge from a optical flow network to a motion vector based network B. Zhang et al., 2016. However, unlike this work, it does not consider the general setting of representation learning on a compressed video, it still needs the entire decompressed video as RGB stream, and it requires optical flow as an additional supervision.
3.3 MODELING COMPRESSED REPRESENTATIONS

Our goal is to design a computer vision system for action recognition that operates directly on the stored compressed video. The compression is solely designed to optimize the size of the encoding, thus the resulting representation has very different statistical and structural properties than the images in a raw video. It is not clear if the successful deep learning techniques can be adapted to compressed representations in a straightforward manner. So we ask how to feed a compressed video into a computer vision system, specifically a deep network?

Feeding I-frames into a deep network is straightforward since they are just images. How about P-frames? From Figure ?? we can see that motion vectors roughly resemble optical flows, though noisy. As modeling optical flows with CNNs has been proven quite effective, it is tempting to do the same for motion vectors. The third row of Figure ?? visualizes the residuals. We can see that they roughly give us a motion boundary in addition to a change of appearance, such as change of lighting conditions. Again, CNNs are well-suited for such patterns. The outputs of corresponding CNNs from the image, motion vectors, and residual will have different properties. To combine them, we tried various fusion strategies, including mean pooling, maxi-
mum pooling, concatenation, convolution pooling, and bilinear pooling, on both middle layers and the final layer, but with limited success.

Digging deeper, one can argue that the motion vectors and residuals alone do not contain the full information of a P-frame — a P-frame depends on the reference frame, which again might be a P-frame. This chain continues all the way back to the closest I-frame. Treating each P-frame as an independent observation clearly violates this dependency. A simple strategy to address this would be to reuse features from the reference frame, and only update the features given the new information. This recurrent definition screams for recurrent neural networks (RNNs) to aggregate features along the chain. However, preliminary experiments suggest the elaborate modeling effort in vain (see supplementary material for details). The difficulty arises due to the long chain of dependencies of the P-frames. To mitigate this issue, we devise a novel yet simple back-tracing technique that decouples individual P-frames.

**Decoupled model.** To break the dependency between consecutive P-frames, we trace all motion vectors back to the reference I-frame and accumulate the residual on the way. In this way, each P-frame depends only on the I-frame but not other P-frames.

Figure 3.4 illustrates the back-tracing technique. Given a pixel at location $i$ in frame $t$, let $\mu_{\gamma(t)}(i) := i - \gamma_{\gamma(t)}(t)$ be the referenced location in the previous frame. The location traced back to frame $k < t$ is given by

$$J_{t,k}^\gamma(i) := \mu_{\gamma(k+1)} \circ \cdots \circ \mu_{\gamma(t)}(i). \quad (3.2)$$

Then the accumulated motion vectors $D_{t,i}^{(t)} \in \mathbb{R}^{H \times W \times 2}$ and the accumulated residuals $R_{t,i}^{(t)} \in \mathbb{R}^{H \times W \times 3}$ at frame $t$ are

$$D_{t,i}^{(t)} := i - J_{t,k}^{(t)}, \quad \text{and}$$

$$R_{t,i}^{(t)} := \Delta_{t}^{(k+1)} + \cdots + \Delta_{t}^{(t-1)} \circ J_{t,k+1}^{(t)} + \Delta_{t}^{(t)},$$

respectively. This can be efficiently calculated in linear time through a simple feed forward algorithm, accumulating motion and residuals as we decode the video. Each P-frame now has a different dependency

$$I_{t}^{(t)} = I^{(0)}_{t - D_{t}^{(t)}} + R_{t}^{(t)}, \quad t = 1, 2, \ldots,$$

as shown in Figure 3.4. Here P-frames depend only on the I-frame and can be processed in parallel.
A nice side effect of the back-tracing is robustness. The accumulated signals contain longer-term information, which is more robust to noise or camera motion. Figure 3.5 shows the accumulated motion vectors and residuals respectively. They exhibit clearer and smoother patterns than the original ones.

**Proposed network.** Figure 3.5 shows the graphical illustration of the proposed model. The input of our model is an I-frame, followed by T P-frames, i.e. \((I^0, D^{(1)}, R^{(1)}, \ldots, D^{(T)}, R^{(T)})\).

For simplicity of notation we set \(t = 0\) for the I-frame. Each input source is modeled by a CNN, i.e.

\[
\begin{align*}
x_{\text{RGB}}^{(0)} &:= \phi_{\text{RGB}}(I^{(0)}) \\
x_{\text{motion}}^{(1)} &:= \phi_{\text{motion}}(D^{(1)}) \\
x_{\text{residual}}^{(1)} &:= \phi_{\text{residual}}(R^{(1)})
\end{align*}
\]

While I-frame features \(x_{\text{RGB}}^{(0)}\) are used as is, P-frame features \(x_{\text{motion}}^{(1)}\) and \(x_{\text{residual}}^{(1)}\) need to incorporate the information from \(x_{\text{RGB}}^{(0)}\). There are several reasonable candidates for such a fusion, e.g. maximum, multiplicative or convolutional pooling. We also experiment with transforming RGB features according to the motion vector. Interestingly, we found a simple summing of scores to work best (see supplementary material for details). This gives us a model that is easy to train and flexible for inference.

**Implementation.** Note that most of the information is stored in I-frames, and we only need to learn the update for P-frames. We thus focus most of the computation on I-frames, and use a much smaller and simpler model to capture the updates in P-frames. This yields significant saving in terms of computation, since in modern codecs most frames are P-frames.

Specifically, we use ResNet-152 to model I-frames, and ResNet-18 to model the motion vectors and residuals. This offers a good trade-off between speed and accuracy. For video-level tasks, we use Temporal Segments L. Wang et al., 2016 to capture long term dependency, i.e. feature at each step is the average of features across \(k = 3\) segments during training.
3.4 EXPERIMENTS

We now validate for task of action recognition that (i) compressed video is a better representation (Section ??), leading to (ii) good accuracy (Section 5.2) and (iii) high speed (Section ??). However, note that the principle of the proposed method can be applied effortlessly to other tasks like video classification Abu-El-Haija et al., 2016, object detection Russakovsky et al., 2015, or action localization Gunnar A Sigurdsson et al., 2016. We pick action recognition due to its wide range of applications and strong baselines.

DATASETS AND PROTOCOL. We evaluate our method Compressed Video Action Recognition (CoViAR) on three action recognition datasets, UCF-101 Soomro, Zamir, and Shah, 2012, HMDB-51 Kuehne et al., 2011, and Charades Gunnar A Sigurdsson et al., 2016. UCF-101 and HMDB-51 contain short (< 10-second) trimmed videos, each of which is annotated with one action label. Charades contains longer (∼ 30-second) untrimmed videos. Each video is annotated with one or more action labels and their intervals (start time, end time). UCF-101 contains 13,320 videos from 101 action categories. HMDB-51 contains 6,766 videos from 51 action categories. Each dataset has 3 (training, testing)-splits. We report the average performance of the 3 testing splits unless otherwise stated. The Charades dataset contains 9,848 videos split into 7,985 training and 1,863 test videos. It contains 157 action classes.

During testing we uniformly sample 25 frames, each with flips plus 5 crops, and then average the scores for final prediction. On UCF-101 and HMDB-51 we use temporal segments, and perform the averaging before softmax following Wang et al. L. Wang et al., 2016. On Charades we use mean average precision (mAP) and weighted average precision (wAP) to evaluate the performance, following previous work Gunnar A. Sigurdsson et al., 2017.

TRAINING DETAILS. Following TSN L. Wang et al., 2016, we resize UCF-101 and HMDB-51 videos to 340 × 256. As Charades contains both portrait and landscape videos, we resize to 256 × 256. Our models are pre-trained on the ILSVRC 2012-CLS dataset Deng et al., 2009, and fine-tuned using Adam D. P. Kingma and Ba, 2014 with a batch size of 40. Learning rate starts from 0.001 for UCF-101/HMDB-51 and 0.03 for Charades. It is divided by 10 when the accuracy plateaus. Pre-trained layers use a learning rate that is 100 times smaller than the base learning rate. We apply color jittering and random cropping to 224 × 224 for data augmentation following Wang et al. L. Wang et al., 2016. Where available, we tune the hyper-parameters on splits other than the tested one. We use MPEG-4 encoded videos, which have on average 11 P-frames for every I-frame. When optical flows are used, we use TV-L1 flows Zach, Pock, and Bischof, 2007.

3.4.1 Ablation Study

Here we study the benefits of using compressed representations over RGB images. We focus on UCF-101 and HMDB-51, as they are two of the most well-studied action recognition datasets. Table 3.1 presents a detailed analysis. On both datasets, training on compressed videos significantly outperforms training on RGB frames. In particular, it provides 5.8% and 2.7% absolute improvement on HMDB-51 and UCF-101 respectively.

Quite surprisingly, while residuals contribute to a very small amount of data, it alone achieves good accuracy. Motion vectors alone perform not as well, as they do not contain spatial details. However, they offer information orthogonal to what still images provide. When added to other streams, it significantly boosts the performance. Note that we use only I-frames as full images, which is a small subset of all frames, yet CoViAR achieves good performance.

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>M</th>
<th>R</th>
<th>I+M</th>
<th>I+R</th>
<th>I+M+R (gain)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UCF-101</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Split 1</td>
<td>88.4</td>
<td>63.9</td>
<td>79.9</td>
<td>90.4</td>
<td>90.0</td>
<td>90.8 (+2.4)</td>
</tr>
<tr>
<td>Split 2</td>
<td>87.4</td>
<td>64.6</td>
<td>80.8</td>
<td>89.9</td>
<td>89.6</td>
<td>90.5 (+3.1)</td>
</tr>
<tr>
<td>Split 3</td>
<td>87.3</td>
<td>66.6</td>
<td>82.1</td>
<td>89.6</td>
<td>89.4</td>
<td>90.0 (+2.7)</td>
</tr>
<tr>
<td>Average</td>
<td>87.7</td>
<td>65.0</td>
<td>80.9</td>
<td>89.4</td>
<td>89.7</td>
<td>90.4 (+2.7)</td>
</tr>
<tr>
<td><strong>HMDB-51</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Split 1</td>
<td>54.1</td>
<td>37.8</td>
<td>44.6</td>
<td>60.3</td>
<td>55.9</td>
<td>60.4 (+6.3)</td>
</tr>
<tr>
<td>Split 2</td>
<td>51.9</td>
<td>38.7</td>
<td>43.1</td>
<td>57.9</td>
<td>54.2</td>
<td>58.2 (+6.3)</td>
</tr>
<tr>
<td>Split 3</td>
<td>54.1</td>
<td>39.7</td>
<td>44.4</td>
<td>58.5</td>
<td>55.6</td>
<td>58.7 (+4.6)</td>
</tr>
<tr>
<td>Average</td>
<td>53.3</td>
<td>38.8</td>
<td>44.1</td>
<td>58.9</td>
<td>55.2</td>
<td>59.1 (+5.8)</td>
</tr>
</tbody>
</table>

Table 3.2: Action recognition accuracy on UFC-101 Soomro, Zamir, and Shah, 2012 (Split 1). The two rows show the performance of the models trained using the original motion vectors/residuals and the models using the accumulated ones respectively. I: I-frame RGB image. M: motion vectors. R: residuals.

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>R</th>
<th>I+M</th>
<th>I+R</th>
<th>I+M+R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>58.3</td>
<td>79.0</td>
<td>90.0</td>
<td>89.8</td>
<td>90.4</td>
</tr>
<tr>
<td>Accumulated</td>
<td>63.9</td>
<td>79.9</td>
<td>90.4</td>
<td>90.0</td>
<td>90.8</td>
</tr>
</tbody>
</table>

**Accumulated motion vectors and residuals.** Accumulating them across multiple frames results in clearer patterns to model. This improves the performance, as shown in Table 3.2. On the first split of UCF-101, our accumulation technique provides 5.6% improvement on the motion vector stream network and on the full model, 0.4% improvement (4.2% error reduction). Performance of the residual stream also improves by 0.9% (4.3% error reduction).

**Visualizations.** In Figure 3.7, we qualitatively study the RGB and compressed representations of two videos of the same action in t-SNE Maaten and G. Hinton, 2008 space. We can see that in RGB space the two videos are clearly separated, and in motion vector and residual space they overlap. This suggests that a RGB-image based model needs to learn the two patterns separately, while a compressed-video based model sees a shared representation for videos of the same action, making training and generalization easier.

In addition, note that the two ways of the RGB trajectories overlap, showing that RGB images cannot distinguish between the up-moving and down-moving motion. On the other hand, compressed signals preserve motion information. The trajectories thus form circles instead of going back and forth on the same path.

### 3.4.2 Speed and Efficiency

Our method is efficient because the computation on I-frame is shared across multiple frames, and the computation on P-frames is cheaper. Table 3.3 compares the CNN computational cost
Table 3.3: Network computation complexity and accuracy of each method. Our method is 4.6x more efficient than state-of-the-art 3D CNN, while being much more accurate.

<table>
<thead>
<tr>
<th></th>
<th>GFLOPs</th>
<th>UCF-101</th>
<th>HMDB-51</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>3.8</td>
<td>82.3</td>
<td>48.9</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>11.3</td>
<td>83.4</td>
<td>46.7</td>
</tr>
<tr>
<td>C3D</td>
<td>38.5</td>
<td>82.3</td>
<td>51.6</td>
</tr>
<tr>
<td>Res3D</td>
<td>19.3</td>
<td>85.8</td>
<td>54.9</td>
</tr>
<tr>
<td>CoViAR</td>
<td>4.2</td>
<td>90.4</td>
<td>59.1</td>
</tr>
</tbody>
</table>

Table 3.4: Speed (ms) per frame of each method. Our method is fast in both preprocessing and CNN computation. The preprocessing speed of our method is presented for both single-thread / multi-thread settings.

<table>
<thead>
<tr>
<th></th>
<th>Preprocess</th>
<th>CNN (sequential)</th>
<th>CNN (concurrent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-stream</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BN-Inception</td>
<td>75.0</td>
<td>1.6</td>
<td>0.9</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>75.0</td>
<td>7.5</td>
<td>4.0</td>
</tr>
<tr>
<td>CoViAR</td>
<td>2.87/0.46</td>
<td>1.3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

of our method with state-of-the-art 2D and 3D CNN architectures. Since for our model the P- and I-frame computational costs are different, we report the average GFLOPs over all frames. As shown in the table, CoViAR is 2.7 times faster than ResNet-152 and is 4.6 times more than Res3D, while being significantly more accurate.

A more detailed speed analysis is presented in Table 3.4. The preprocessing time of two-stream methods, i.e. optical flow computation, is measured on a Tesla P100 GPU with an implementation of the TV-L1 flow algorithm from OpenCV. Our preprocessing, i.e. the calculation of the accumulated motion vectors and residuals, is measured on Intel E5-2698 v4 CPUs. CNN time is measured on the same P100 GPU. We can see that the optical flow computation is the bottleneck for two-stream networks, even with low-resolution $256 \times 340$ videos. Our preprocessing is much faster despite our CPU-only implementation.

For CNN time, we consider both settings where (i) we can forward multiple CNNs at the same time, and (ii) we do it sequentially. For both settings, our method is significantly faster than traditional methods. Overall, our method can be up to 100 times faster than traditional methods, when preprocessing time is considered, processing more than 1,300 frames per second. Figure 3.6 summarizes the results. CoViAR achieves the best efficiency and good accuracy, while requiring a far lesser amount of data.

3.4.3 Accuracy

We now compare accuracy of our method CoViAR with state-of-the-art models in Table 3.6. From the upper part of the table, we can see that our model significantly outperforms traditional RGB-image based methods. C3D Tran, Bourdev, et al., 2015, Res3D Tran, Ray, et al., 2017 and I3D Carreira and Zisserman, 2017 consider 3D convolution to learn temporal structures.
Figure 3.6: Speed and accuracy on UCF-101 Soomro, Zamir, and Shah, 2012, compared to Two-stream Network Simonyan and Zisserman, 2014, Res3D Tran, Ray, et al., 2017, and ResNet-152 He et al., 2016b trained using RGB frames. Node size denotes the input data size. Training on compressed videos is both accurate and efficient, while requiring a minimal amount of data.

![Graph showing speed and accuracy comparison](image)


<table>
<thead>
<tr>
<th></th>
<th>UCF-101</th>
<th>HMDB-51</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CoViAR</td>
<td>Flow</td>
</tr>
<tr>
<td>Split 1</td>
<td>90.8</td>
<td>87.7</td>
</tr>
<tr>
<td>Split 2</td>
<td>90.5</td>
<td>90.2</td>
</tr>
<tr>
<td>Split 3</td>
<td>90.0</td>
<td>89.1</td>
</tr>
<tr>
<td>Average</td>
<td>90.4</td>
<td>89.0</td>
</tr>
</tbody>
</table>

Karpathy et al. Karpathy, Toderici, et al., 2014 and TLE Diba, Sharma, and Van Gool, 2017 consider more complicated fusions and pooling. MV-CNN B. Zhang et al., 2016 applies distillation to transfer knowledge from an optical-flow-based model. Our method uses much faster 2D CNNs, simple late fusion without additional supervision, and still significantly outperforms these methods.

**Two-stream network.** Most state-of-the-art models use the two-stream framework, i.e. one stream trained on RGB frames and the other on optical flows. It is natural to ask: What if we replace the RGB stream with our compressed stream? Here we train a temporal-stream network using 7 segments with BN-Inception, and combine it with our model by late fusion. Despite its simplicity, this achieves very good performance as shown in Table 3.6.

The lower part of Table 3.6 compares our method with state-of-the-art models using optical flow. CoViAR outperforms all of them. In particular, LRCN Donahue et al., 2015, Composite LSTM Model Srivastava, Mansimov, and Salakhudinov, 2015, and L^2STM L. Sun et al., 2017 use RNNs to model temporal dynamics. ActionVLAD Girdhar, Ramanan, et al., 2017 and TLE Diba, 2017, despite our best efforts, we were not able to reproduce the performance reported in the original paper. Here we report the performance based on our implementation. For fair comparison, we use the same data augmentation and architecture as ours. Training follows the 2-stage procedure described in the original paper. We reached out to the authors, but they were unable to share their implementation.
Table 3.6: Action recognition accuracy on UCF-101 Soomro, Zamir, and Shah, 2012 and HMDB-51 Kuehne et al., 2011. The upper part of the table lists real-time methods that do not require computing optical flow. The lower part of the table lists methods that require computing optical flow. Our method outperforms all baselines in both settings. Asterisk indicates results evaluated only on split 1 of the datasets. They are listed purely for reference.

<table>
<thead>
<tr>
<th>Method</th>
<th>UCF-101</th>
<th>HMDB-51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without optical flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Karpathy et al. Karpathy, Toderici, et al., 2014</td>
<td>65.4</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-50 He et al., 2016b</td>
<td>82.3</td>
<td>48.9</td>
</tr>
<tr>
<td>(from ST-Mult Feichtenhofer, Pinz, and R. P. Wildes, 2017)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResNet-152 He et al., 2016b</td>
<td>83.4</td>
<td>46.7</td>
</tr>
<tr>
<td>(from ST-Mult Feichtenhofer, Pinz, and R. P. Wildes, 2017)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3D Tran, Bourdev, et al., 2015</td>
<td>82.3</td>
<td>51.6</td>
</tr>
<tr>
<td>Res3D Tran, Ray, et al., 2017</td>
<td>85.8</td>
<td>54.9</td>
</tr>
<tr>
<td>TSN (RGB-only) L. Wang et al., 2016¹</td>
<td>85.7</td>
<td>-</td>
</tr>
<tr>
<td>TLE (RGB-only) Diba, Sharma, and Van Gool, 2017²</td>
<td>87.9</td>
<td>54.2</td>
</tr>
<tr>
<td>I3D (RGB-only) Carreira and Zisserman, 2017⁴</td>
<td>84.5</td>
<td>49.8</td>
</tr>
<tr>
<td>MV-CNN B. Zhang et al., 2016</td>
<td>86.4</td>
<td>-</td>
</tr>
<tr>
<td>Attentional Pooling Girdhar and Ramanan, 2017</td>
<td></td>
<td>52.2</td>
</tr>
<tr>
<td>CoViAR</td>
<td>90.4</td>
<td>59.1</td>
</tr>
<tr>
<td>With optical flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iDT+FT H. Wang and Schmid, 2013</td>
<td></td>
<td>57.2</td>
</tr>
<tr>
<td>Two-Stream Simonyan and Zisserman, 2014</td>
<td>88.0</td>
<td>59.4</td>
</tr>
<tr>
<td>Two-Stream fusion Feichtenhofer, Pinz, and Zisserman, 2016</td>
<td>92.5</td>
<td>65.4</td>
</tr>
<tr>
<td>LRCN Donahue et al., 2015</td>
<td>82.7</td>
<td></td>
</tr>
<tr>
<td>Composite LSTM Model Srivastava, Mansimov, and Salakhudinov, 2015</td>
<td>84.3</td>
<td>44.0</td>
</tr>
<tr>
<td>ActionVLAD Girdhar, Ramanan, et al., 2017</td>
<td>92.7</td>
<td>66.9</td>
</tr>
<tr>
<td>ST-ResNet Feichtenhofer, Pinz, and R. Wildes, 2016</td>
<td>93.4</td>
<td>66.4</td>
</tr>
<tr>
<td>ST-Mult Feichtenhofer, Pinz, and R. P. Wildes, 2017</td>
<td>94.2</td>
<td>68.9</td>
</tr>
<tr>
<td>I3D Carreira and Zisserman, 2017⁴</td>
<td>93.4</td>
<td>66.4</td>
</tr>
<tr>
<td>TLE Diba, Sharma, and Van Gool, 2017²</td>
<td>93.8</td>
<td>68.8</td>
</tr>
<tr>
<td>L²STM L. Sun et al., 2017</td>
<td>93.6</td>
<td>66.2</td>
</tr>
<tr>
<td>ShuttleNet Y. Shi et al., 2017</td>
<td>94.4</td>
<td>66.6</td>
</tr>
<tr>
<td>STPN Y. Wang et al., 2017</td>
<td>94.6</td>
<td>68.9</td>
</tr>
<tr>
<td>TSN L. Wang et al., 2016</td>
<td>94.2</td>
<td>69.4</td>
</tr>
<tr>
<td>CoViAR + optical flow</td>
<td>94.9</td>
<td>70.2</td>
</tr>
</tbody>
</table>

Sharma, and Van Gool, 2017 apply more complicated feature aggregation. iDT+FT H. Wang and Schmid, 2013 is based on hand-engineered features. Again, our method simply trains 2D CNNs separately without any complicated fusion or RNN and still outperforms these models.

Finally we evaluate our method on the Charades dataset. As the Charades dataset consists of longer videos with labels annotated at frame-level, we train our network to predict the labels.
Without optical flow

<table>
<thead>
<tr>
<th>Model</th>
<th>mAP (%)</th>
<th>wAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionVLAD Girdhar, Ramanan, et al., 2017 (RGB only)</td>
<td>17.6</td>
<td>25.1</td>
</tr>
<tr>
<td>Sigurdsson et al. Gunnar A. Sigurdsson et al., 2017 (RGB only)</td>
<td>18.3</td>
<td>-</td>
</tr>
<tr>
<td>CoViAR</td>
<td>21.9</td>
<td>29.4</td>
</tr>
</tbody>
</table>

With optical flow

<table>
<thead>
<tr>
<th>Model</th>
<th>mAP (%)</th>
<th>wAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-stream Simonyan and Zisserman, 2014 (from Gunnar A Sigurdsson et al., 2016)</td>
<td>14.3</td>
<td>-</td>
</tr>
<tr>
<td>Two-stream Simonyan and Zisserman, 2014 + iDT H. Wang and Schmid, 2013 (from Gunnar A Sigurdsson et al., 2016)</td>
<td>18.6</td>
<td>-</td>
</tr>
<tr>
<td>ActionVLAD Girdhar, Ramanan, et al., 2017 (RGB only) + iDT</td>
<td>21.0</td>
<td>29.9</td>
</tr>
<tr>
<td>Sigurdsson et al. Gunnar A. Sigurdsson et al., 2017</td>
<td>22.4</td>
<td>-</td>
</tr>
<tr>
<td>CoViAR + optical flow</td>
<td>24.1</td>
<td>32.3</td>
</tr>
</tbody>
</table>

Table 3.7: Action recognition accuracy on Charades Gunnar A. Sigurdsson et al., 2016. Without using additional annotations as Sigurdsson et al. Gunnar A. Sigurdsson et al., 2017 or complicated feature aggregation, our method achieves the best performance.

of each frame. At test time we average the scores of the sampled frames as the final prediction. Table 3.7 shows the results. We can see that our method again outperforms other models trained on RGB images. Note that Sigurdsson et al. use additional annotations including objects, scenes, and intentions to train a conditional random field (CRF) model Gunnar A. Sigurdsson et al., 2017. Our model requires only action labels. When using optical flow, we beat all other state-of-the-art methods. The effectiveness on Charades demonstrates that our method is not just effective for video-level predictions, but also for frame-level predictions.

3.5 CONCLUSION

In this paper, we proposed to train deep networks directly on compressed videos. This is motivated by the practical observation that either video compression is essentially free on all modern cameras, due to hardware-accelerated video codecs or that the video is directly available in its compressed form. In other words, decompressing the video is actually an inconvenience.

We demonstrate that, quite surprisingly, this is not a drawback but rather a virtue. In particular, video compression reduces irrelevant information from the data, thus rendering it more robust. After all, compression is not meant to affect the content that humans consider pertinent. Secondly, the increased relevance and reduced dimensionality makes computation much more effective (we are able to use much simpler networks for motion vectors and residuals). Finally, the accuracy of the model actually improves when using compressed data, yielding new state of the art.

In short, our method is both faster and more accurate, while being simpler to implement than previous works.
Figure 3.7: Two videos of “Jumping Jack” from UCF-101 in their RGB, motion vector, and residual representations plotted in t-SNE Maaten and G. Hinton, 2008 space. The curves show video trajectories. While in the RGB space the two videos are clearly separated, in the motion vector and residual space they overlap. This suggests that with compressed signals, videos of the same action can share statistical strength better. Also note that the RGB images contain no motion information, and thus the two ways of the trajectories overlap. This is in contrast to the circular patterns in the trajectories of motion vectors. Best viewed on screen.
**HETEROGENEOUS DATA**

Existing question answering methods infer answers either from a knowledge base or from raw text Figure 4.1. While knowledge base (KB) methods are good at answering compositional questions, their performance is often affected by the incompleteness of the KB. Au contraire, web text contains millions of facts that are absent in the KB, however in an unstructured form. Universal schema can support reasoning on the union of both structured KBs and unstructured text by aligning them in a common embedded space. In this paper we extend universal schema to natural language question answering, employing memory networks to attend to the large body of facts in the combination of text and KB. Our models can be trained in an end-to-end fashion on question-answer pairs. Evaluation results on SPADES fill-in-the-blank question answering dataset show that exploiting universal schema for question answering is better than using either a KB or text alone. This model also outperforms the current state-of-the-art by 8.5 F1 points.

### 4.1 INTRODUCTION

Question Answering (QA) has been a long-standing goal of natural language processing. Two main paradigms evolved in solving this problem: 1) answering questions on a knowledge base; and 2) answering questions using text.

Knowledge bases (KB) contains facts expressed in a fixed schema, facilitating compositional reasoning. These attracted research ever since the early days of computer science, e.g., BASEBALL Green Jr et al., 1961. This problem has matured into learning semantic parsers from parallel question and logical form pairs Zelle and Raymond J Mooney, 1996; L. S. Zettlemoyer and Collins, 2005, to recent scaling of methods to work on very large KBs like Freebase using question and answer pairs Berant et al., 2013. However, a major drawback of this paradigm is that KBs are highly incomplete Dong et al., 2014. It is also an open question whether KB

![Figure 4.1: Existing approaches utilizes only one type of information source. However combination of the heterogeneous information source is powerful. We want to develop a question answering system that can leverage both the information source.](image-url)

The paradigm of exploiting text for questions started in the early 1990s Kupiec, 1993. With the advent of web, access to text resources became abundant and cheap. Initiatives like TREC QA competitions helped popularizing this paradigm Voorhees et al., 1999. With the recent advances in deep learning and availability of large public datasets, there has been an explosion of research in a very short time Choi, Hewlett, et al., 2016; Lee et al., 2016; Nguyen et al., 2016; Rajpurkar et al., 2016; Seo et al., 2016; Trischler et al., 2016; S. Wang and Jiang, 2016; C. Xiong, Zhong, and Socher, 2016. Still, text representation is unstructured and does not allow the compositional reasoning which structured KB supports.

An important but under-explored QA paradigm is where KB and text are exploited together Ferrucci et al., 2010. Such combination is attractive because text contains millions of facts not present in KB, and a KB’s generative capacity represents infinite number of facts that are never seen in text. However QA inference on this combination is challenging due to the structural non-uniformity of KB and text. Distinct supervision methods Bunescu and Raymond J. Mooney, 2007; Mintz et al., 2009; Riedel, L. Yao, and McCallum, 2010; L. Yao, Riedel, and McCallum, 2010; Zeng et al., 2015 address this problem partially by means of aligning text patterns with KB. But the rich and ambiguous nature of language allows a fact to be expressed in many different forms which these models fail to capture. Universal schema Riedel, L. Yao, McCallum, and Marlin, 2013 avoids the alignment problem by jointly embedding KB facts and text into a uniform structured representation, allowing interleaved propagation of information. Figure 4.2 shows a universal schema matrix which has pairs of entities as rows, and Freebase and textual relations in columns. Although universal schema has been extensively used for relation extraction, this paper shows its applicability to QA. Consider the question USA has elected _blank_, our first african-american president with its answer Barack Obama. While Freebase has a predicate for representing presidents of USA, it does not have one for ‘african-american’ presidents. Whereas in text, we find many sentences describing the presidency of Barack Obama and his
ethnicity at the same time. Exploiting both KB and text makes it relatively easy to answer this question than relying on only one of these sources.

Memory networks (MemNN; MemNN) are a class of neural models which have an external memory component for encoding short and long term context. In this work, we define the memory components as observed cells of the universal schema matrix, and train an end-to-end QA model on question-answer pairs.

The contributions of the paper are as follows (a) We show that universal schema representation is a better knowledge source for QA than either KB or text alone, (b) On the SPADES dataset Bisk et al., 2016, containing real world fill-in-the-blank questions, we outperform state-of-the-art semantic parsing baseline, with 8.5 F1 points. (c) Our analysis shows how individual data sources help fill the weakness of the other, thereby improving overall performance.

4.2 BACKGROUND

Problem definition Given a question q with words w1, w2, ..., wn, where these words contain one _blank_ and at least one entity, our goal is to fill in this _blank_ with an answer entity qa using a knowledge base K and text T. Few example question answer pairs are shown in Table 4.2.

Universal schema Traditionally universal schema is used for relation extraction in the context of knowledge base population. Rows in the schema are formed by entity pairs (e.g. USA, NYC), and columns represent the relation between them. A relation can either be a KB relation, or it could be a pattern of text that exist between these two entities in a large corpus. The embeddings of entities and relation types are learned by low-rank matrix factorization techniques. Riedel, L. Yao, McCallum, and Marlin, 2013 treat textual patterns as static symbols, whereas recent work by Verga et al., 2016 replaces them with distributed representation of sentences obtained by a RNN. Using distributed representation allows reasoning on sentences that are similar in meaning but different on the surface form. We too use this variant to encode our textual relations.

Memory networks MemNNs are neural attention models with external and differentiable memory. MemNNs decouple the memory component from the network thereby allowing it store external information. Previously, these have been successfully applied to question answering on KB where the memory is filled with distributed representation of KB triples Bordes et al., 2015, or for reading comprehension Hill et al., 2016; Sukhbaatar, Szlam, et al., 2015, where the memory consists of distributed representation of sentences in the comprehension. Recently, key-value MemNN are introduced A. H. Miller et al., 2016 where each memory slot consists of a key and value. The attention weight is computed only by comparing the question with the key memory, whereas the value is used to compute the contextual representation to predict the answer. We use this variant of MemNN for our model. A. H. Miller et al., 2016, in their experiments, store either KB triples or sentences as memories but they do not explicitly model multiple memories containing distinct data sources like we do.

4.3 Model

Our model is a MemNN with universal schema as its memory. Figure 4.2 shows the model architecture.
**MEMORY:** Our memory \( \mathcal{M} \) comprise of both KB and textual triples from universal schema. Each memory cell is in the form of key-value pair. Let \((s, r, o) \in \mathcal{K}\) represent a KB triple. We represent this fact with distributed key \( k \in \mathbb{R}^{2d} \) formed by concatenating the embeddings \( s \in \mathbb{R}^d \) and \( r \in \mathbb{R}^d \) of subject entity \( s \) and relation \( r \) respectively. The embedding \( o \in \mathbb{R}^d \) of object entity \( o \) is treated as its value \( v \).

Let \((s, [w_1, \ldots, \text{arg}_1, \ldots, \text{arg}_2, w_n], o) \in \mathcal{T}\) represent a textual fact, where \( \text{arg}_1 \) and \( \text{arg}_2 \) correspond to the positions of the entities ‘s’ and ‘o’. We represent the key as the sequence formed by replacing \( \text{arg}_1 \) with ‘s’ and \( \text{arg}_2 \) with a special ‘_blank_’ token, i.e., \( k = [w_1, \ldots, s, \ldots, _\text{blank}_-, w_n] \) and value as just the entity ‘o’. We convert \( k \) to a distributed representation using a bidirectional LSTM Graves and Jürgen Schmidhuber, 2005; Hochreiter and Jürgen Schmidhuber, 1997, where \( k \in \mathbb{R}^{2d} \) is formed by concatenating the last states of forward and backward LSTM, i.e., \( k = [\text{LSTM}(k); \overleftarrow{\text{LSTM}}(k)] \). The value \( v \) is the embedding of the object entity \( o \). Projecting both KB and textual facts to \( \mathbb{R}^{2d} \) offers a unified view of the knowledge to reason upon. In Figure 4.2, each cell in the matrix represents a memory containing the distributed representation of its key and value.

**QUESTION ENCODER:** A bidirectional LSTM is also used to encode the input question \( q \) to a distributed representation \( q \in \mathbb{R}^{2d} \) similar to the key encoding step above.

**ATTENTION OVER CELLS:** We compute attention weight of a memory cell by taking the dot product of its key \( k \) with a contextual vector \( c \) which encodes most important context in the current iteration. In the first iteration, the contextual vector is the question itself. We only consider the memory cells that contain at least one entity in the question. For example, for the input question in Figure 4.2, we only consider memory cells containing USA. Using the attention weights and values of memory cells, we compute the context vector \( c_t \) for the next iteration \( t \) as follows:

\[
c_t = W_t \left( c_{t-1} + W_p \sum_{(k, v) \in \mathcal{M}} (c_{t-1} \cdot k) v \right)
\]

where \( c_0 \) is initialized with question embedding \( q \), \( W_p \) is a projection matrix, and \( W_t \) represents the weight matrix which considers the context in previous hop and the values in the current iteration based on their importance (attention weight). This multi-iterative context selection allows multi-hop reasoning without explicitly requiring a symbolic query representation.

**ANSWER ENTITY SELECTION:** The final contextual vector \( c_t \) is used to select the answer entity \( q_a \) (among all 1.8M entities in the dataset) which has the highest inner product with it.

\[
q_a = \arg\max_e e \cdot c_t
\]

Finally, we minimize the cross-entropy loss to train the complete MemNN in an end-to-end fashion.

### 4.4 EXPERIMENTS

We describe our evaluation datasets, implementation details, baseline models and results.

#### 4.4.1 Evaluation Dataset

We use Freebase Bollacker et al., 2008 as our KB, and ClueWeb Gabrilovich, Ringgaard, and Subramanya, 2013 as our text source to build universal schema. For evaluation, literature of-
fers two options: 1) datasets for text-based question answering tasks such as answer sentence selection and reading comprehension; and 2) datasets for KB question answering.

Although the text-based question answering datasets are large in size, e.g., SQuAD Rajpurkar et al., 2016 has over 100k questions, answers to these are often not entities but rather sentences which are not the focus of our work. Moreover these texts may not contain Freebase entities at all, making these skewed heavily towards text. Coming to the alternative option, WebQuestions Berant et al., 2013 is widely used for QA on Freebase. This dataset is curated such that all questions can be answered on Freebase alone. But since our goal is to explore the impact of universal schema, testing on a dataset completely answerable on a KB is not ideal. WikiMovies dataset A. H. Miller et al., 2016 also has similar properties. Gardner and Krishnamurthy, 2017 created a dataset with motivations similar to ours, however this is not publicly released during the submission time.

Instead, we use Spades Bisk et al., 2016 as our evaluation data which contains fill-in-the-blank cloze-styled questions created from ClueWeb. This dataset is ideal to test our hypothesis for following reasons: 1) it is large with 93K sentences and 1.8M entities; and 2) since these are collected from Web, most sentences are natural. A limitation of this dataset is that it contains only the sentences that have entities connected by at least one relation in Freebase, making it skewed towards Freebase as we will see (§4.4.4). We use the standard train, dev and test splits for our experiments. For text part of universal schema, we use the sentences present in the training set.

### 4.4.2 Models

We evaluate the following models to measure the impact of different knowledge sources for QA.

- **onlyKB**: In this model, MemNN memory contains only the facts from KB. For each KB triple \((e_1, r, e_2)\), we have two memory slots, one for \((e_1, r, e_2)\) and the other for its inverse \((e_2, r^1, e_1)\).

- **onlyText**: Spades contains sentences with blanks. We replace the blank tokens with the answer entities to create textual facts from the training set. Using every pair of entities, we create a memory cell similar to as in universal schema.

- **ensemble**: This is an ensemble of the above two models. We use a linear model that combines the scores from, and use an ensemble to combine the evidences from individual models.

- **unischema**: This is our main model with universal schema as its memory, i.e., it contains memory slots corresponding to both KB and textual facts.

### Table 4.1: QA results on Spades.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev. F₁</th>
<th>Test F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bisk et al., 2016</td>
<td>32.7</td>
<td>31.4</td>
</tr>
<tr>
<td>ONLYKB</td>
<td>39.1</td>
<td>38.5</td>
</tr>
<tr>
<td>ONLYText</td>
<td>25.3</td>
<td>26.6</td>
</tr>
<tr>
<td>Ensemble</td>
<td>39.4</td>
<td>38.6</td>
</tr>
<tr>
<td>UniSchema</td>
<td><strong>41.1</strong></td>
<td><strong>39.9</strong></td>
</tr>
</tbody>
</table>

[July 17, 2018 at 16:21 – MT version 0.1]
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. USA have elected <em>blank</em>, our first african-american president.</td>
<td>Obama</td>
</tr>
<tr>
<td>2. Angelina has reportedly been threatening to leave <em>blank</em>.</td>
<td>Brad_Pitt</td>
</tr>
<tr>
<td>3. Spanish is more often a second and weaker language among many <em>blank</em>.</td>
<td>Latinos</td>
</tr>
<tr>
<td>4. <em>blank</em> is the third largest city in the United_States.</td>
<td>Chicago</td>
</tr>
<tr>
<td>5. <em>blank</em> was Belshazzar ‘s father.</td>
<td>Nabonidus</td>
</tr>
</tbody>
</table>

Table 4.2: A few questions on which ONLYKB fails to answer but UNISHEMA succeeds.

### 4.4.3 Implementation Details

The dimensions of word, entity and relation embeddings, and LSTM states were set to $d=50$. The word and entity embeddings were initialized with word2vec Mikolov, Sutskever, et al., 2013 trained on 7.5 million ClueWeb sentences containing entities in Freebase subset of SPADES. The network weights were initialized using Xavier initialization Glorot and Y. Bengio, 2010. We considered up to a maximum of 5k KB facts and 2.5k textual facts for a question. We used Adam adam with the default hyperparameters (learning rate=$1e^{-3}$, $\beta_1=0.9$, $\beta_2=0.999$, $\epsilon=1e^{-8}$) for optimization. To overcome exploding gradients, we restricted the magnitude of the $\ell_2$ norm of the gradient to 5. The batch size during training was set to 32.

To train the UNISHEMA model, we initialized the parameters from a trained ONLYKB model. We found that this is crucial in making the UNISHEMA to work. Another caveat is the need to employ a trick similar to batch normalization Ioffe and Szegedy, 2015. For each minibatch, we normalize the mean and variance of the textual facts and then scale and shift to match the mean and variance of the KB memory facts. Empirically, this stabilized the training and gave a boost in the final performance.

### 4.4.4 Results and Discussions

Table 4.1 shows the main results on SPADES. UNISHEMA outperforms all our models validating our hypothesis that exploiting universal schema for QA is better than using either KB or text alone. Despite SPADES creation process being friendly to Freebase, exploiting text still provides a significant improvement. Table 4.2 shows some of the questions which UNISHEMA answered but ONLYKB failed. These can be broadly classified into (a) relations that are not expressed in Freebase (e.g., african-american presidents in sentence 1); (b) intentional facts since curated databases only represent concrete facts rather than intentions (e.g., threatening to leave in sentence 2); (c) comparative predicates like first, second, largest, smallest (e.g., sentences 3 and 4); and (d) providing additional type constraints (e.g., in sentence 5, Freebase does not have a special relation for father. It can be expressed using the relation parent along with the type constraint that the answer is of gender male).

We have also analyzed the nature of UNISHEMA attention. In 58.7% of the cases the attention tends to prefer KB facts over text. This is as expected since KBs facts are concrete and accurate than text. In 34.8% of cases, the memory prefers to attend text even if the fact is already present in the KB. For the rest (6.5%), the memory distributes attention weight evenly, indicating for some questions, part of the evidence comes from text and part of it from KB. Table 4.3 gives a more detailed quantitative analysis of the three models in comparison with each other.
To see how reliable is UniSchema, we gradually increased the coverage of KB by allowing only a fixed number of randomly chosen KB facts for each entity. As Figure 4.3 shows, when the KB coverage is less than 16 facts per entity, UniSchema outperforms ONLYKB by a wide-margin indicating UniSchema is robust even in resource-scarce scenario, whereas ONLYKB is very sensitive to the coverage. UniSchema also outperforms Ensemble showing joint modeling is superior to ensemble on the individual models. We also achieve the state-of-the-art with 8.5 F1 points difference. Bisk et al., 2016 use graph matching techniques to convert natural language to Freebase queries whereas even without an explicit query representation, we outperform them.

### 4.5 RELATED WORK

A majority of the QA literature that focused on exploiting KB and text either improves the inference on the KB using text based features Choi, Kwiatkowski, and L. Zettlemoyer, 2015; Guu, J. Miller, and Liang, 2015; Joshi, Sawant, and Chakrabarti, 2014; Krishnamurthy and Mitchell, 2012; Neelakantan, Roth, and McCallum, 2015; Reddy, Lapata, and Steedman, 2014; Savenkov and Agichtein, 2016; Kun Xu, Reddy, et al., 2016; X. Yao and Van Durme, 2014; Yih et al., 2015 or improves the inference on text using KB H. Sun et al., 2015.

Limited work exists on exploiting text and KB jointly for question answering. Gardner and Krishnamurthy, 2017 is the closest to ours who generate a open-vocabulary logical form and rank candidate answers by how likely they occur with this logical form both in Freebase and text. Our models are trained on a weaker supervision signal without requiring the annotation of the logical forms.

A few QA methods infer on curated databases combined with OpenIE triples Fader, L. Zettlemoyer, and Etzioni, 2014; Kun Xu, Y. Feng, et al., 2016; Yahya et al., 2016. Our work differs from them in two ways: 1) we do not need an explicit database query to retrieve the answers

---

Table 4.3: Detailed results on Spades.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev. F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLYKB correct</td>
<td>39.1</td>
</tr>
<tr>
<td>ONLYTEXT correct</td>
<td>25.3</td>
</tr>
<tr>
<td>UniSchema correct</td>
<td>41.1</td>
</tr>
<tr>
<td>ONLYKB or ONLYTEXT got it correct</td>
<td>45.9</td>
</tr>
<tr>
<td>Both ONLYKB and ONLYTEXT got it correct</td>
<td>18.5</td>
</tr>
<tr>
<td>ONLYKB got it correct and ONLYTEXT did not</td>
<td>20.6</td>
</tr>
<tr>
<td>ONLYTEXT got it correct and ONLYKB did not</td>
<td>6.80</td>
</tr>
<tr>
<td>Both UniSchema and ONLYKB got it correct</td>
<td>34.6</td>
</tr>
<tr>
<td>UniSchema got it correct and ONLYKB did not</td>
<td>6.42</td>
</tr>
<tr>
<td>ONLYKB got it correct and UniSchema did not</td>
<td>4.47</td>
</tr>
<tr>
<td>Both UniSchema and ONLYTEXT got it correct</td>
<td>19.2</td>
</tr>
<tr>
<td>UniSchema got it correct and ONLYTEXT did not</td>
<td>21.9</td>
</tr>
<tr>
<td>ONLYTEXT got it correct and UniSchema did not</td>
<td>6.09</td>
</tr>
</tbody>
</table>
Figure 4.3: Performance on varying the number of available KB facts during test time. UniSchema model consistently outperforms ONLYKB

Andreas et al., 2016; Neelakantan, Q. V. Le, and Sutskever, 2015; and 2) our text-based facts retain complete sentential context unlike the OpenIE triples Banko et al., 2007; Carlson et al., 2010.

4.6 CONCLUSIONS

In this work, we showed universal schema is a promising knowledge source for QA than using KB or text alone. Our results conclude though KB is preferred over text when the KB contains the fact of interest, a large portion of queries still attend to text indicating the amalgam of both text and KB is superior than KB alone.
Part II

INTERPRETABILITY BY INCORPORATING DOMAIN KNOWLEDGE
Continuous space word embeddings learned from large, unstructured corpora have been shown to be effective at capturing semantic regularities in language. In this paper we replace LDA’s parameterization of “topics” as categorical distributions over opaque word types with multivariate Gaussian distributions on the embedding space. This encourages the model to group words that are \textit{a priori} known to be semantically related into topics. To perform inference, we introduce a fast collapsed Gibbs sampling algorithm based on Cholesky decompositions of covariance matrices of the posterior predictive distributions. We further derive a scalable algorithm that draws samples from stale posterior predictive distributions and corrects them with a Metropolis–Hastings step. Using vectors learned from a domain-general corpus (English Wikipedia), we report results on two document collections (20-newsgroups and NIPS). Qualitatively, Gaussian LDA infers different (but still very sensible) topics relative to standard LDA. Quantitatively, our technique outperforms existing models at dealing with OOV words in held-out documents.

5.1 Introduction

Latent Dirichlet Allocation (LDA) is a Bayesian technique that is widely used for inferring the topic structure in corpora of documents. It conceives of a document as a mixture of a small number of topics, and topics as a (relatively sparse) distribution over word types D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003a. These priors are remarkably effective at producing useful results. However, our intuitions tell us that while documents may indeed be conceived of as a mixture of topics, we should further expect topics to be \textit{semantically coherent}. Indeed, standard human evaluations of topic modeling performance are designed to elicit assessment of semantic coherence J. Chang, J. Boyd-Graber, et al., 2006; Newman, Karimi, and Cavedon, 2009. However, this prior preference for semantic coherence is not encoded in the model, and any such observation of semantic coherence found in the inferred topic distributions is, in some sense, accidental. In this paper, we develop a variant of LDA that operates on continuous space embeddings of words—rather than word types—to impose a prior expectation for semantic coherence. Our approach replaces the opaque word types usually modeled in LDA with continuous space embeddings of these words, which are generated as draws from a multivariate Gaussian.

How does this capture our preference for semantic coherence? Word embeddings have been shown to capture lexico-semantic regularities in language: words with similar syntactic and semantic properties are found to be close to each other in the embedding space Agirre et al., 2009; Mikolov, Yih, and Zweig, 2013. Since Gaussian distributions capture a notion of centrality in space, and semantically related words are localized in space, our Gaussian LDA model encodes a prior preference for semantically coherent topics. Our model further has several advantages. Traditional LDA assumes a fixed vocabulary of word types. This modeling assumption drawback as it cannot handle \textit{out of vocabulary} (OOV) words in “held out” documents. Zhai and J. L. Boyd-Graber, 2013 proposed an approach to address this problem by drawing topics from a
Dirichlet Process with a base distribution over all possible character strings (i.e., words). While this model can in principle handle unseen words, the only bias toward being included in a particular topic comes from the topic assignments in the rest of the document. Our model can exploit the contiguity of semantically similar words in the embedding space and can assign high topic probability to a word which is similar to an existing topical word even if it has never been seen before.

The main contributions of our paper are as follows: We propose a new technique for topic modeling by treating the document as a collection of word embeddings and topics itself as multivariate Gaussian distributions in the embedding space (§5.3). We explore several strategies for collapsed Gibbs sampling and derive scalable algorithms, achieving asymptotic speed-up over the naïve implementation (§5.4). We qualitatively show that our topics make intuitive sense and quantitatively demonstrate that our model captures a better representation of a document in the topic space by outperforming other models in a classification task (§6.4).

5.2 BACKGROUND

Before going to the details of our model we provide some background on two topics relevant to our work: vector space word embeddings and LDA.

5.2.1 Vector Space Semantics

According to the distributional hypothesis Harris, 1954, words occurring in similar contexts tend to have similar meaning. This has given rise to data-driven learning of word vectors that capture lexical and semantic properties, which is now a technique of central importance in natural language processing. These word vectors can be used for identifying semantically related word pairs Agirre et al., 2009; Turney, 2006 or as features in downstream text processing applications Guo et al., 2014; Turian, Ratinov, and Y. Bengio, 2010. Word vectors can either be constructed using low rank approximations of cooccurrence statistics Deerwester et al., 1990 or using internal representations from neural network models of word sequences Collobert and Weston, 2008. We use a recently popular and fast tool called word2vec1, to generate skip-gram word embeddings from unlabeled corpus. In this model, a word is used as an input to a log-linear classifier with continuous projection layer and words within a certain window before and after the words are predicted.

5.2.2 Latent Dirichlet Allocation (LDA)

LDA D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003a is a probabilistic topic model of corpora of documents which seeks to represent the underlying thematic structure of the document collection. They have emerged as a powerful new technique of finding useful structure in an unstructured collection as it learns distributions over words. The high probability words in each distribution gives us a way of understanding the contents of the corpus at a very high level. In LDA, each document of the corpus is assumed to have a distribution over K topics, where the discrete topic distributions are drawn from a symmetric Dirichlet distribution. The generative process is as follows.

1. for k = 1 to K
   a) Choose topic $\beta_k \sim \text{Dir}(\eta)$

2. for each document d in corpus D

---

1 https://code.google.com/p/word2vec/
5.3 GAUSSIAN LDA

As with multinomial LDA, we are interested in modeling a collection of documents. However, we assume that rather than consisting of sequences of word types, documents consist of sequences of word embeddings. We write \( v(w) \in \mathbb{R}^M \) as the embedding of word of type \( w \) or \( v_{d,i} \) when we are indexing a vector in a document \( d \) at position \( i \).

Since our observations are no longer discrete values but continuous vectors in an \( M \)-dimensional space, we characterize each topic \( k \) as a multivariate Gaussian distribution with mean \( \mu_k \) and covariance \( \Sigma_k \). The choice of a Gaussian parameterization is justified by both analytic convenience and observations that Euclidean distances between embeddings correlate with semantic similarity Collobert and Weston, 2008; Hermann and Blunsom, 2014; Turney and Pantel, 2010. We place conjugate priors on these values: a Gaussian centered at zero for the mean and an inverse Wishart distribution for the covariance. As before, each document is seen as a mixture of topics whose proportions are drawn from a symmetric Dirichlet prior. The generative process can thus be summarized as follows:

1. for \( k = 1 \) to \( K \)
   a) Draw topic covariance \( \Sigma_k \sim W^{-1}(\Psi, \nu) \)
   b) Draw topic mean \( \mu_k \sim N(\mu, \frac{1}{\nu} \Sigma_k) \)
2. for each document \( d \) in corpus \( D \)
   a) Draw topic distribution \( \theta_d \sim \text{Dir}(\alpha) \)
   b) for each word index \( n \) from \( 1 \) to \( N_d \)
      i. Draw a topic \( z_n \sim \text{Categorical}(\theta_d) \)
      ii. Draw \( v_{d,n} \sim N(\mu_{z_n}, \Sigma_{z_n}) \)

This model has previously been proposed for obtaining indexing representations for audio retrieval P. Hu et al., 2012. They use variational/EM method for posterior inference. Although we don’t do any experiment to compare the running time of both approaches, the per-iteration computational complexity is same for both inference methods. We propose a faster inference technique using Cholesky decomposition of covariance matrices which can be applied to both the Gibbs and variational/EM method. However we are not aware of any straightforward way of applying the aliasing trick proposed by Aaron Q. Li et al., 2014a on the variational/EM method which gave us huge improvement on running time (see Figure 5.1) Another work
which combines embedding with topic models is by Wan, L. Zhu, and Fergus, 2012 where they jointly learn the parameters of a neural network and a topic model to capture the topic distribution of low dimensional representation of images.

5.4 POSTERIOR INFEERENCE

In our application, we observe documents consisting of word vectors and wish to infer the posterior distribution over the topic parameters, proportions, and the topic assignments of individual words. Since there is no analytic form of the posterior, approximations are required. Because of our choice of conjugate priors for topic parameters and proportions, these variables can be analytically integrated out, and we can derive a collapsed Gibbs sampler that resamples topic assignments to individual word vectors, similar to the collapsed sampling scheme proposed by T. L. Griffiths and Steyvers, 2004.

The conditional distribution we need for sampling is shown in Figure ???. Here, \( z_{-(d,i)} \) represents the topic assignments of all word embeddings, excluding the one at \( i^{th} \) position of document \( d \); \( V_d \) is the sequence of vectors for document \( d \); \( t_{\nu'}(x | \mu', \Sigma') \) is the multivariate \( t \) - distribution with \( \nu' \) degrees of freedom and parameters \( \mu' \) and \( \Sigma' \). The tuple \( \zeta = (\mu, \kappa, \Sigma, \nu) \) represents the parameters of the prior distribution.

It should be noted that the first part of the equation which expresses the probability of topic \( k \) in document \( d \) is the same as that of LDA. This is because the portion of the model which generates a topic for each word (vector) from its document topic distribution is still the same. The second part of the equation which expresses the probability of assignment of topic \( k \) to the word vector \( v_{d,i} \) given the current topic assignments (aka posterior predictive) is given by a multivariate \( t \) distribution with parameters \( (\mu_k, \kappa_k, \Sigma_k, \nu_k) \). The parameters of the posterior predictive distribution are given as Murphy, 2012:

\[
\begin{align*}
\kappa_k &= \kappa + N_k \\
\mu_k &= \frac{\kappa \mu + N_k \bar{v}_k}{\kappa_k} \\
\nu_k &= \nu + N_k \\
\Sigma_k &= \frac{\Psi_k}{(\nu_k - M + 1)} \\
\Psi_k &= \Psi + C_k + \frac{\kappa N_k}{\kappa_k} (\bar{v}_k - \mu)(\bar{v}_k - \mu)^\top
\end{align*}
\] (5.1)

where \( \bar{v}_k \) and \( C_k \) are given by,

\[
\bar{v}_k = \frac{\sum_d \sum_{i:z_{d,i}=k} (v_{d,i})}{N_k}
\]

\[
C_k = \sum_d \sum_{i:z_{d,i}=k} (v_{d,i} - \bar{v}_k)(v_{d,i} - \bar{v}_k)^\top
\]

Here \( \bar{v}_k \) is the sample mean and \( C_k \) is the scaled form of sample covariance of the vectors with topic assignment \( k \). \( N_k \) represents the count of words assigned to topic \( k \) across all documents. Intuitively the parameters \( \mu_k \) and \( \Sigma_k \) represents the posterior mean and covariance of the topic distribution and \( \kappa_k, \nu_k \) represents the strength of the prior for mean and covariance respectively.

Analysis of running time complexity

As can be seen from (5.10), for computation of the posterior predictive we need to evaluate the determinant and inverse of the posterior covariance matrix. Direct naïve computation of
these terms require $O(M^3)$ operations. Moreover, during sampling as words get assigned to different topics, the parameters $(\mu_k, \kappa_k, \Psi_k, \nu_k)$ associated with a topic changes and hence we have to recompute the determinant and inverse matrix. Since these step has to be recomputed several times (as many times as number of words times number of topics in one Gibbs sweep, in the worst case), it is critical to make the process as efficient as possible. We speed up this process by employing a combination of modern computational techniques and mathematical (linear algebra) tricks, as described in the following subsections.

5.4.1 Faster sampling using Cholesky decomposition of covariance matrix

Having another look at the posterior equation for $\Psi_k$, we can re-write the equation as:

$$
\Psi_k = \Psi + C_k + \frac{\kappa_N}{\kappa_k} (\nu_k - \mu)(\nu_k - \mu)^T
$$

$$
= \Psi + \sum_d \sum_{i: z_{d,i} = k} v_{d,i}v_{d,i}^T - \kappa_k \mu_k \mu_k^T 
+ \kappa \mu \mu^T. \tag{5.2}
$$

During sampling when we are computing the assignment probability of topic $k$ to $v_{d,i}$, we need to calculate the updated parameters of the topic. Using (5.2) it can be shown that $\Psi_k$ can be updated from current value of $\Psi_k$, after updating $\kappa_k, \nu_k$ and $\mu_k$, as follows:

$$
\Psi_k \leftarrow \Psi_k + \frac{\kappa_k}{\kappa_k - 1} \left( \mu_k - v_{d,i} \right) \left( \mu_k - v_{d,i} \right)^T. \tag{5.3}
$$

This equation has the form of a rank 1 update, hinting towards use of Cholesky decomposition. If we have the Cholesky decomposition of $\Psi_k$ computed, then we have tools to update $\Psi_k$ cheaply. Since $\Psi_k$ and $\Sigma_k$ are off by only a scalar factor, we can equivalently talk about $\Sigma_k$. Equation (5.3) can also be understood in the following way. During sampling, when a word embedding $v_{d,i}$ gets a new assignment to a topic, say $k$, then the new value of the topic covariance can be computed from the current one using just a rank 1 update.\footnote{Similarly the covariance of the old topic assignment of the word $w$ can be computed using a rank 1 downdate.}

We next describe how to exploit the Cholesky decomposition representation to speed up computations.

For sake of completeness, any symmetric $M \times M$ real matrix $\Sigma_k$ is said to be positive definite if $\forall z \in \mathbb{R}^M : z^T \Sigma_k z > 0$. The Cholesky decomposition of such a symmetric positive definite matrix $\Sigma_k$ is nothing but its decomposition into the product of some lower triangular matrix $L$ and its transpose, i.e.

$$
\Sigma_k = LL^T.
$$

Finding this factorization also take cubic operation. However given Cholesky decomposition of $\Sigma_k$, after a rank 1 update (or downdate), i.e. the operation:

$$
\Sigma_k \leftarrow \Sigma_k + zz^T
$$

we can find the factorization of new $\Sigma_k$ in just quadratic time\cite{Stewart, 1998}. We will use this trick to speed up the computations\footnote{For our experiments, we set the prior covariance to be $\gamma^2 I$, which is a positive definite matrix.}. Basically, instead of computing determinant and inverse again in cubic time, we will use such rank 1 update (downdate) to find new determinant and inverse in an efficient manner as explained in details below.
To compute the density of the posterior predictive $t$--distribution, we need to compute the determinant $|\Sigma_k|$ and the term of the form $(v_{d,i} - \mu_k)^\top \Sigma_k^{-1} (v_{d,i} - \mu_k)$. The Cholesky decomposition of the covariance matrix can be used for efficient computation of these expression as shown below.

**Computation of determinant**: The determinant of $\Sigma_k$ can be computed from its Cholesky decomposition $L$ as:

$$\log(|\Sigma_k|) = 2 \times \sum_{i=1}^{M} \log (L_{i,i}).$$

This takes linear time in the order of dimension and is clearly a significant gain from cubic time complexity.

**Computation of $(v_{d,i} - \mu_k)^\top \Sigma_k^{-1} (v_{d,i} - \mu_k)$**: Let $b = (v_{d,i} - \mu_k)$. Now $b^\top \Sigma^{-1} b$ can be written as

$$b^\top \Sigma^{-1} b = b^\top (LL^\top)^{-1} b$$
$$= b^\top (L^{-1})^\top L^{-1} b$$
$$= (L^{-1} b)^\top (L^{-1} b)$$

Now $(L^{-1} b)$ is the solution of the equation $Lx = b$. Also since $L$ is a lower triangular matrix, this equation can be solved easily using forward substitution. Lastly we will have to take an inner product of $x$ and $x^\top$ to get the value of $(v_{d,i} - \mu_k)^\top \Sigma^{-1} (v_{d,i} - \mu_k)$. This step again takes quadratic time and is again a savings from the cubic time complexity.

### 5.4.2 Further reduction of sampling complexity using Alias Sampling

Although Cholesky trick helps us to reduce the sampling complexity of a embedding to $O(KM^2)$, it can still be impractical. In Gaussian LDA, the Gibbs sampling equation (5.3) can be split into two terms. The first term $n_{k,d} \times t_{v_{k-M+1}} \left( v_{d,i} \mid \mu_k, \kappa_k k_k \right)$ denotes the document contribution and the second term $\alpha_k \times t_{v_{k-M+1}} \left( v_{d,i} \mid \mu_k, \kappa_k k_k + 1 \right)$ denotes the language model contribution. Empirically one can make two observations about these terms. First, $n_{k,d}$ is often a sparse vector, as a document most likely contains only a few of the topics. Secondly, topic parameters $(\mu_k, \Sigma_k)$ captures global phenomenon, and rather change relatively slowly over the iterations. We can exploit these findings to avoid the naive approach to draw a sample from:

$$p(z_{d,i} = k \mid z_{-(d,i)}, V_d, \ell, \alpha) \propto (n_{k,d} + \alpha_k) \times t_{v_{k-M+1}} \left( v_{d,i} \mid \mu_k, \kappa_k k_k + 1 \right)$$

In particular, we compute the document-specific sparse term exactly and for the remainder language model term we borrow idea from Aaron Q. Li et al., 2014a. We use a slightly stale distribution for the language model. Then Metropolis Hastings (MH) algorithm allows us to convert the stale sample into a fresh one, provided that we compute ratios between successive states correctly. It is sufficient to run MH for a few number of steps because the stale distribution acting as the proposal is very similar to the target. This is because, as pointed out earlier, the language model term does not change too drastically whenever we resample a single word. The number of words is huge, hence the amount of change per word is concomitantly small. (Only if one could convert stale bread into fresh one, it would solve world’s food problem!)
The exercise of using stale distribution and MH steps is advantageous because sampling from it can be carried out in $O(1)$ amortized time, thanks to alias sampling technique Michael D. Vose, 1991. Moreover, the task of building the alias tables can be outsourced to other cores.

With the combination of both Cholesky and Alias tricks, the sampling complexity can thus be brought down to $O(K_d M^2)$ where $K_d$ represents the number of actually instantiated topics in the document and $K_d \ll K$. In particular, we plot the sampling rate achieved naively, with Cholesky (CH) trick and with Cholesky+Alias (A+CH) trick in figure 5.1 demonstrating better likelihood at much less time. Also after initial few iterations, the time per iteration of A+CH trick is 9.93 times less than CH and 53.1 times less than naive method. This is because initially we start with random initialization of words to topics, but after few iterations the $n_{k,d}$ vector starts to become sparse.

5.5 EXPERIMENTS

In this section we evaluate our Word Vector Topic Model on various experimental tasks. Specifically we wish to determine:

- Is our model is able to find coherent and meaningful topics?
- Is our model able to infer the topic distribution of a held-out document even when the document contains words which were previously unseen?

We run our experiments on two datasets 20-newsgroup$^4$ and NIPS$^5$. All the datasets were tokenized and lowercased with cdec Dyer et al., 2010.

5.5.1 Topic Coherence

Quantitative analysis Typically topic models are evaluated based on the likelihood of held-out documents. But in this case, it is not correct to compare perplexities with models

---

$^4$ A collection of newsgroup documents partitioned into 20 news groups. After pre-processing we had 18768 documents. We randomly selected 2000 documents as our test set. This dataset is publicly available at http://qwone.com/~jason/20Newsgroups/

$^5$ A collection of 1740 papers from the proceedings of Neural Information Processing System. The dataset is available at http://www.cs.nyu.edu/~roweis/data.html

[July 17, 2018 at 16:21 – MT version 0.1]
which do topic modeling on words. Since our topics are continuous distributions, the probability of a word vector is given by its density w.r.t the normal distribution based on its topic assignment, instead of a probability mass from a discrete topic distribution. Moreover, J. Chang, J. Boyd-Graber, et al., 2009 showed that higher likelihood of held-out documents doesn’t necessarily correspond to human perception of topic coherence. Instead to measure topic coherence we follow Newman, Karimi, and Cavedon, 2009 to compute the Pointwise Mutual Information (PMI) of topic words w.r.t wikipedia articles. We extract the document co-occurrence statistics of topic words from Wikipedia and compute the score of a topic by averaging the score of the top 15 words of the topic. A higher PMI score implies a more coherent topic as it means the topic words usually co-occur in the same document. In the last line of Table ??, we present the PMI score for some of the topics for both Gaussian LDA and traditional multinomial LDA. It can be seen that Gaussian LDA is a clear winner, achieving an average 275% higher score on average.

However, we are using embeddings trained on Wikipedia corpus itself, and the PMI measure is computed from co-occurrence in the Wikipedia corpus. As a result, our model is definitely biased towards producing higher PMI. Nevertheless Wikipedia PMI is a believed to be a good measure of semantic coherence.

**Qualitative analysis** Table ?? shows some top words from topics from Gaussian-LDA and LDA on the 20-news dataset for $K = 50$. The words in Gaussian-LDA are ranked based on their density assigned to them by the posterior predictive distribution in the final sample. As shown, Gaussian LDA is able to capture several intuitive topics in the corpus such as sports, government, ‘religion’, ‘universities’, ‘tech’, ‘finance’ etc. One interesting topic discovered by our model (on both 20-news and NIPS dataset) is the collection of human names, which was not captured by classic LDA. While one might imagine that names associated with particular topics might be preferable to a ‘names-in-general’ topic, this ultimately is a matter of user preference. More substantively, classic LDA failed to identify the ‘finance’ topics. We also noticed that there were certain words (‘don’, ‘writes’, etc) which often came as a top word in many topics in classic LDA. However our model was not able to capture the ‘space’ topics which LDA was able to identify.

Also we visualize a part of the continuous space where the word embedding is performed. For this task we performed the Principal Component Analysis (PCA) over all the word vectors and plot the first two components as shown in Figure 5.2. We can see clear separations between some of the clusters of topics as depicted. The other topics would be separated in other dimensions.

### 5.5.2 Performance on document containing new words

In this experiment we evaluate the performance of our model on documents which contains previously unseen words. It should be noted that traditional topic modeling algorithms will typically ignore such words while inferring the topic distribution and hence might miss out important words. The continuous topic distributions of the Word Vector Topic Model on the other hand, will be able to assign topics to an unseen word, if we have the vector representation of the word. Given the recent development of fast and scalable methods of estimating word embeddings, it is possible to train them on huge text corpora and hence it makes our model a viable alternative for topic inference on documents with new words.

**Experimental setup:** Since we want to capture the strength of our model on documents containing unseen words, we select a subset of documents and replace words of those documents by its synonyms if they haven’t occurred in the corpus before. We obtain the synonym
Table 5.1: Top words of some topics from Gaussian-LDA and multinomial LDA on 20-newsgroups for $K = 50$. Words in Gaussian LDA are ranked based on density assigned to them by the posterior predictive distribution. The last row for each method indicates the PMI score (w.r.t. Wikipedia co-occurrence) of the topic’s fifteen highest ranked words.

of a word using two existing resources and hence we create two such datasets. For the first set, we use the Paraphrase Database Ganitkevitch, Van Durme, and Callison-Burch, 2013 to get the lexical paraphrase of a word. The paraphrase database\(^6\) is a semantic lexicon containing around 169 million paraphrase pairs of which 7.6 million are lexical (one word to one word) paraphrases. The dataset comes in varying size ranges starting from S to XXXL in increasing order of size and decreasing order of paraphrasing confidence. For our experiments we selected the L size of the paraphrase database.

\(^6\) http://www.cis.upenn.edu/~ccb/ppdb/
Figure 5.2: The first two principal components for the word embeddings of the top words of topics shown in Table 5 are visualized. Each blob represents a word color coded according to its topic in the Table.

The second set was obtained using WordNet G. A. Miller, 1995, a large human annotated lexicon for English that groups words into sets of synonyms called synsets. To obtain the synonym of a word, we first label the words with their part-of-speech using the Stanford POS tagger Toutanova et al., 2003. Then we use the WordNet database to get the synonym from its synset.7 We select the first synonym from the synset which hasn’t occurred in the corpus before. On the 20-news dataset (vocab size = 18,179 words, test corpus size = 188,694 words), a total of 21,919 words (2,741 distinct words) were replaced by synonyms from PPDB and 38,687 words (2,037 distinct words) were replaced by synonyms from Wordnet.

evaluation benchmark: As mentioned before traditional topic model algorithms cannot handle OOV words. So comparing the performance of our document with those models would be unfair. Recently Zhai and J. L. Boyd-Graber, 2013 proposed an extension of LDA (infvoc) which can incorporate new words. They have shown better performances in a document classification task which uses the topic distribution of a document as features on the 20-news group dataset as compared to other fixed vocabulary algorithms. Even though, the infvoc model can handle OOV words, it will most likely not assign high probability to a new topical word when it encounters it for the first time since it is directly proportional to the number of times the word has been observed. On the other hand, our model could assign high probability to the word if its corresponding embedding gets a high probability from one of the topic gaussians. With the experimental setup mentioned before, we want to evaluate performance of this property of our model. Using the topic distribution of a document as features, we try to classify the document into one of the 20 news groups it belongs to. If the document topic distribution is modeled well, then our model should be able to do a better job in the classification task.

To infer the topic distribution of a document we follow the usual strategy of fixing the learnt topics during the training phase and then running Gibbs sampling on the test set (G-LDA (fix) in table 5.2). However infvoc is an online algorithm, so it would be unfair to compare our model which observes the entire set of documents during test time. Therefore we implement the online version of our algorithm using Gibbs sampling following L. Yao, Mimno, and McCallum, 2009a. We input the test documents in batches and do inference on those batches independently also.

7 We use the JW toolkit Finlayson, 2014
sampling for the topic parameter, along the lines of infvoc. The batch size for our experiments are mentioned in parentheses in table 5.2. We classify using the multi class logistic regression classifier available in Weka M. Hall et al., 2009.

It is clear from table 5.2 that we outperform infvoc in all settings of our experiments. This implies that even if new documents have significant amount of new words, our model would still do a better job in modeling it. We also conduct an experiment to check the actual difference between the topic distribution of the original and synthetic documents. Let $h$ and $h'$ denote the topic vectors of the original and synthetic documents. Table 5.3 shows the average $l_1$, $l_2$ and $l_\infty$ norm of $(h - h')$ of the test documents in the NIPS dataset. A low value of these metrics indicates higher similarity. As shown in the table, Gaussian LDA performs better here too.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPDB</td>
<td>WordNet</td>
</tr>
<tr>
<td>infvoc</td>
<td>28.00%</td>
</tr>
<tr>
<td>G-LDA (fix)</td>
<td>44.51%</td>
</tr>
<tr>
<td>G-LDA (1)</td>
<td>44.66%</td>
</tr>
<tr>
<td>G-LDA (100)</td>
<td>43.63%</td>
</tr>
<tr>
<td>G-LDA (1932)</td>
<td>44.72%</td>
</tr>
</tbody>
</table>

Table 5.2: Accuracy of our model and infvoc on the synthetic datasets. In Gaussian LDA fix, the topic distributions learnt during training were fixed; G-LDA(1, 100, 1932) is the online implementation of our model where the documents comes in minibatches. The number in parenthesis denote the size of the batch. The full size of the test corpus is 1932.

<table>
<thead>
<tr>
<th>Model</th>
<th>PPDB (Mean Deviation)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$L_1$</td>
</tr>
<tr>
<td>infvoc</td>
<td>94.95</td>
</tr>
<tr>
<td>G-LDA (fix)</td>
<td>15.13</td>
</tr>
<tr>
<td>G-LDA (1)</td>
<td>15.71</td>
</tr>
<tr>
<td>G-LDA (100)</td>
<td>15.76</td>
</tr>
<tr>
<td>G-LDA (174)</td>
<td>14.58</td>
</tr>
</tbody>
</table>

Table 5.3: This table shows the Average $L_1$ Deviation, Average $L_2$ Deviation, Average $L_\infty$ Deviation for the difference of the topic distribution of the actual document and the synthetic document on the NIPS corpus. Compared to infvoc, G-LDA achieves a lower deviation of topic distribution inferred on the synthetic documents with respect to actual document. The full size of the test corpus is 174.

5.6 CONCLUSION AND FUTURE WORK

While word embeddings have been incorporated to produce state-of-the-art results in numerous supervised natural language processing tasks from the word level to document level Q. Le and Mikolov, 2014; Turian, Ratинov, and Y. Bengio, 2010, inter alia; however, they have played a more minor role in unsupervised learning problems. This work shows some of the promise that they hold in this domain. Our model can be extended in a number of potentially useful, but straightforward ways. First, DPMM models of word emissions would better model the fact that identical vectors will be generated multiple times, and perhaps add flexibility to the topic
distributions that can be captured, without sacrificing our preference for topical coherence. More broadly still, running LDA on documents consisting of different modalities than just text is facilitated by using the \textit{lingua franca} of vector space representations, so we expect numerous interesting applications in this area. An interesting extension to our work would be the ability to handle polysemous words based on multi-prototype vector space models Neelakantan, Shankar, et al., 2014; Reisinger and Raymond J. Mooney, 2010 and we keep this as an avenue for future research.

\section{Derivation of Rank 1 Update of the Covariance Matrix}

Due to the conjugate properties of Normal and Normal Inverse Wishart distribution, the parameters of the posterior predictive distribution are given as:

\begin{align}
\kappa_k &= \kappa + N_k \\
\mu_k &= \frac{\kappa \mu + N_k \bar{v}_k}{\kappa_k} \\
\nu_k &= \nu + N_k \\
\Sigma_k &= \frac{\nu_k - M + 1}{\psi_k} \\
\psi_k &= \psi + C_k + \frac{\kappa N_k}{\kappa_k} (\bar{v}_k - \mu)(\bar{v}_k - \mu)^\top
\end{align}

where $\bar{v}_k$ and $C_k$ are given by:

\begin{align}
\bar{v}_k &= \sum_d \sum_{i: z_{d,i} = k} (v_{d,i}) \\
C_k &= \sum_d \sum_{i: z_{d,i} = k} (v_{d,i} - \bar{v}_k)(v_{d,i} - \bar{v}_k)^\top \\
&= \sum_{i: z_{i} = k} (v_{i} - \bar{v}_k)(v_{i} - \bar{v}_k)^\top
\end{align}

In the last step we replace two summations by one.

Intuitively $\bar{v}_k$ is the sample mean and $C_k$ is the scaled form of sample covariance of the vectors with topic assignment $k$. $N_k$ represents the count of words assigned to topic $k$ across all documents.

First let’s look at the update for the mean

\begin{align}
\mu_k &= \frac{\kappa \mu + N_k \bar{v}_k}{\kappa_k} \\
&= \frac{\kappa \mu + N_k \bar{v}_k}{\kappa_k} \\
(5.6)
\end{align}

One way of looking at equation (5.6): $\mu_k$ is the mean of topic $k$ which has presently $N_k$ words assigned to it. (Equivalently table $k$ has $N_k$ customers (words) sitting on it). When a new word is assigned to (or an existing word is removed from) a topic, we have to update the mean $\mu_k$ of the table based on the update equation (5.6). Let us denote the mean of the topic $k$ which presently has $N$ words assigned to it as $\mu_k^N$. Then from equation (5.6), it can be seen that $\mu_k^N$ can be recursively written in terms of $\mu_k^{N_k-1}$ as

\begin{align}
\mu_k^N &= \frac{(\kappa + N_k - 1) \mu_k^{N_k-1} + v_{d,i}}{\kappa + N_k} \\
(5.7)
\end{align}
Here \( v_{d,i} \) denotes the word embedding at position \( i \) of document \( d \) which was assigned topic \( k \) during sampling and which made the mean of the topic to change from \( \mu_k^{N_k-1} \) to \( \mu_k^{N_k} \). The reader should note that this could be any given word during sampling.

Now lets look at the update for \( \Psi_k \).

\[
\Psi_k = \Psi + C_k + \frac{\kappa N_k}{\kappa_k} (\bar{v}_k - \mu)(\bar{v}_k - \mu)^T
\]

\[
= \Psi + \sum_{i:z_{d,i}=k} (v_i - \bar{v}_k)(v_i - \bar{v}_k)^T + \frac{\kappa N_k}{\kappa_k} (\bar{v}_k - \mu)(\bar{v}_k - \mu)^T
\]

\[
= \Psi + \sum_{i:z_{d,i}=k} v_i v_i^T - \sum_{i:z_{d,i}=k} v_i \bar{v}_k^T - \bar{v}_k \sum_{i:z_{d,i}=k} v_i^T + N_k \bar{v}_k \bar{v}_k^T + \frac{\kappa N_k}{\kappa_k} (\bar{v}_k - \mu)(\bar{v}_k - \mu)^T
\]

\[
= \Psi + \sum_{i:z_{d,i}=k} v_i v_i^T - N_k \bar{v}_k \bar{v}_k^T + \frac{\kappa N_k}{\kappa_k} (\bar{v}_k - \mu)(\bar{v}_k - \mu)^T
\]

\[
= \Psi + \sum_{i:z_{d,i}=k} v_i v_i^T - (\kappa + N_k) \mu_k \mu_k^T + \kappa \mu \mu^T
\]  

\[ (5.8) \]

We get the last step by using equation (5.6). Also following similar nomenclature we will be denoting the scaled covariance of a topic with \( N \) words as \( \Psi^N \).

If we look closely, equation (5.8) also can be recursively written as follows:

\[
\Psi_k^{N_k} = \Psi_k^{(N_k-1)} + v_{d,i} v_{d,i}^T - (\kappa + N_k) \mu_k^{N_k} \mu_k^{N_k} + (\kappa + N_k - 1) \mu_k^{(N_k-1)} \mu_k^{(N_k-1)}
\]

Now writing \( \mu_k^{(N_k-1)} \) in terms of \( \mu_k^{(N_k)} \) using equation (5.7),

\[
\Psi_k^{N_k} = \Psi_k^{(N_k-1)} + \frac{\kappa + N_k}{\kappa + N_k - 1} (\mu_k^{(N_k)} - v_{d,i})(\mu_k^{(N_k)} - v_{d,i})^T
\]

This is the same as equation 4 in the paper! and is in the form of a Rank 1 update. Therefore, if we have the Cholesky decomposition for \( \Psi_k^{(N_k-1)} \), then we can compute the Cholesky decomposition of \( \Psi_k^{(N_k)} \) in quadratic time complexity in the number of dimensions (which is an order of magnitude improvement from the cubic time complexity)

### 5.8 Alias Sampling for LDA with Gaussian Emission Model

Graphical models with latent variables, have become a popular tool for modelling, visualization, inference, and exploratory analysis of multivariate data with diverse applications ranging from text to images to user behavior. In particular, the application for topic models, which
are very versatile tools in discovering the hidden thematic structure and drawn a lot attention recently in performing tasks such as information retrieval, document browsing, content recommendation.

In the advent of big data and high performance computing, there is a need to improve existing models or devise new algorithms which can scale to terabyte-level amount of data. Since the most common data format is text, scalable topic modelling techniques are no exceptional. However, inference procedures reported for many latent variable models in academic literature are limited and focused on providing proofs of concept. The scaling of these procedure to be able to deal with real life problems in industry consisting of massive datasets (1 TB+) in a reasonable time with reasonable resources is non-trivial.

One of the key operations in performing inference in such latent variable models with Gibbs Sampling is to draw a sample from discrete state-space associated with the data, say of size $K$. Typically we end-up with distribution of the form:

$$p(k|x) \propto p_d(k)p_k(x) + \alpha p_k(x)$$  \hspace{1cm} (5.9)

where $p_k(x)$ is the contribution from the emission model (similarity) and $p_d(k)$ is the popularity. For example, in case of topic models for each word $x$ token in a document $d$, the $p_d(t)$ is count of the topic in the document and $p_k(x)$ is word probability in that topic according to the emission model.

A naive approach to draw a sample from $p(k|x)$ would be sequential and have a computational complexity of $O(KW)$, where $W$ denotes the cost of computing $p_k(x)$, which is $O(D^2)$ for gaussian LDA. Assume that we have $N$ word tokens. Since we have to go through the sampling process per token, in total the complexity of Gibbs’ iteration is $O(NKW)$, which is significantly large in the context of big corpus of hundreds of millions of tokens and thousands of outcomes. Hence, reducing the $O(KW)$ complexity of drawing a sample is the key to having efficient Gibbs sampler.

To reduce the complexity of drawing from $p(k|x)$, there are two observations in practice, characterising the properties of $p_d(k)$ and $p_k(x)$. $p_d(k)$ is often a sparse vector in terms of non-zero outcome weights. Besides, it also accounts for most of the probability mass of $p(t)$. On the contrary, $p_k(x)$ is more dense. However as it captures global phenomenon, it slowly varies over iterations. Based on the way these two observations are utilized (Li, 2014).

On the contrary, draw sample from an approximation of $p(k|x)$. These methods do not need to update $p(k|x)$ for every iteration, thus saving a lot of computation in case of multidimensional vectors of LDA. The key point here is to draw samples from proposal distribution, which is a close approximation to $p(k|x)$, and accept/reject samples with Metropolis-Hasting algorithm. The proposal distributions are frequently updated on a separate thread, which do not affect the main sampling thread. The complexity of these methods depends on the design of proposal distributions (Li, 2014). can achieve $O(KdW)$.

### 5.8.1 Alias Method

Alias Method can achieve this within $O(1)$ by maintaining an auxiliary data structure called the Alias table. The basic idea here is to distribute the outcomes over the buckets so that all buckets share the same weight. We hence can access directly to the bucket from the given random seed. This makes the first difference between Alias Method and Linear/Binary Search. The second difference is in the organization of buckets. It is possible for buckets to have at most two outcomes (instead of 1 in Linear/Binary Search) and each outcome has multiple existences in different buckets. In this section we briefly describe the Vose’s implementation for the Alias
Table, which is the up-to-date standard and numerically stable algorithm. For the proof of existence of Alias Table given any discrete distribution, interested readers are referred to read (Vose,1991) or this technical report\(^8\).

For **Initialization**, Vose’s implementation requires two additional data structures: alias and prob. Both are arrays of K elements. The alias array can be treated as buckets, where alias[i] contains two elements, i-th and alias[i]-th outcome. Either of the outcomes is then selected with the probability of prob[i].

First we normalize the given weights to ensure their sum equal to 1. Since the weights are uniformly distributed over K buckets, each bucket has to contain some outcomes with total weight of 1/K. The strategy is as follows. From the pool of K outcomes, two groups are formed, small and large. Small contains outcomes with weight strictly less than 1/K, while Large contains the rest. We randomly pick two outcomes \(i\) and \(j\), one from each group. Assume that \(p_i < 1/K\) and \(p_j \geq 1/K\). We set prob[i] = \(p_i\) and alias[i]=j. We return the \(j\) – th outcome back the pool and put it in one of the groups corresponding to its new weight \(p_j = p_j - (1/K - p_i)\). The pairing process is repeated until the pool is empty. The overall complexity is \(O(K)\). We can discard the \(p\), small and large arrays after this procedure.

For **Generation**, we need two samples from uniform distributions. The first sample is to decide the bucket index. The second sample is to select the outcomes belong to that bucket with probability prob[i]. Since it requires only two comparison, the complexity is \(O(1)\). Algorithm ... describes the full process.

### 5.8.2 Gaussian LDA with Alias

In case of Gaussian LDA, the Gibbs sampling equation takes form of \(p_d(k) = n_{k,d}\) and \(p_k(x)\) is the posterior predictive of word vector \(x\) to belong to the topic \(k\), i.e. the multivariate t-distribution \(t_{\nu_k-M+1} \left( \frac{1}{\kappa_k} \Sigma_k \right)\). So overall we obtain:

\[
p(z_{d,i}=k \mid z_{-(d,i)}, V_d, \Theta, \alpha) \propto \left( n_{k,d} \times t_{\nu_k-M+1} \left( \frac{1}{\kappa_k} \Sigma_k \right) \right) \tag{5.10}
\]

The derivation of the posterior predictive as language model is standard and some details are provided in the supplement A.

The proposed method, based on (Li,2014) involves computing the document-specific sparse term exactly and approximating the remainder with slightly stale data. First of all computing the

---

\(^8\) http://www.keithschwarz.com/darts-dice-coins/
Recurrent neural networks, such as long-short term memory (LSTM) networks, are powerful tools for modeling sequential data like user browsing history Korpusik, Sakaki, and F. C. Y.-Y. Chen, 2016; Tan, X. Xu, and Y. Liu, 2016 or natural language text Mikolov, Karafiát, et al., 2010. However, to generalize across different user types, LSTMs require a large number of parameters, notwithstanding the simplicity of the underlying dynamics, rendering it uninterpretable, which is highly undesirable in user modeling. The increase in complexity and parameters arises due to a large action space in which many of the actions have similar intent or topic. In this paper, we introduce Latent LSTM Allocation (LLA) for user modeling combining hierarchical Bayesian models with LSTMs. In LLA, each user is modeled as a sequence of actions, and the model jointly groups actions into topics and learns the temporal dynamics over the topic sequence, instead of action space directly. This leads to a model that is highly interpretable, concise, and can capture intricate dynamics. We present an efficient Stochastic EM inference algorithm for our model that scales to millions of users/documents. Our experimental evaluations show that the proposed model compares favorably with several state-of-the-art baselines.

6.1 INTRODUCTION

Sequential data prediction is an important problem in machine learning spanning over a diverse set of applications ranging from text Mikolov, Karafiát, et al., 2010 to user behavior Köck and Paramythis, 2011. For example, when applied to statistical language modeling, the goal is to predict the next word in textual data given context, very similar to that in user activity modeling where the aim is to predict the next activity of the user given the history. Accurate user activity modeling is very important for serving relevant, personalized, and useful contents to the user. A good model of sequential data should be accurate, sparse, and interpretable. Unfortunately, none of the existing techniques for user or language modeling satisfy all of these requirements.

The state-of-the-art for modeling sequential data is to employ recurrent neural networks (RNN) Lipton, Berkowitz, and Elkan, 2015, such as LSTMs (Long-Short Term Memory) Hochreiter and Jürgen Schmidhuber, 1997. Such RNNs have been shown to be effective at capturing long and

![Figure 6.1: LLA has better perplexity (lower is better) than LDA but much fewer parameters than LSTMs, as shown in a language modeling task on Wikipedia.](image)
short patterns in data, e.g. token-level semantic as well as syntactic regularities in language Jozefowicz et al., 2016. However, the neural network representations are generally uninterpretable and inaccessible to humans Strobelt et al., 2016. Moreover, the number of parameters of the model is proportional to the number of observed word types or action types, which can grow to tens or hundreds of millions. Note that for user modeling task, character level RNN is not feasible because user actions are often not words but hash indices or URLs.

On the other hand of the spectrum, latent variable models with multi-task learning, such as LDA D. Blei, A. Ng, and M. Jordan, 2002 and other topic model variants, which are strictly not sequence models, proved to be powerful tools for uncovering latent structure in both text and user data Aly et al., 2012 with good commercial success D. J. Hu, R. Hall, and Attenberg, 2014. Topic models are popular for their ability to organize the data into a smaller set of prominent themes or topics through statistical strength sharing across users or documents. These topic representations are generally accessible to humans and easily lend themselves to being interpreted.

In this paper, we propose Latent LSTM Allocation (LLA), a model that bridges the gap between the sequential RNN’s and the non-sequential LDA. LLA borrows graphical model techniques to infer topics (groups of related word or user activities) by sharing statistical strength across users/documents and recurrent deep networks to model the dynamics of topic evolution inside each sequence (document or user activities) rather than at user action/word level (Sec. 6.3.1). LLA inherits sparsity and interpretability from LDA, while borrowing accuracy from LSTM. We provide various variants of LLA that trade model size vs. accuracy without sacrificing interpretability (Sec. 6.3.3). As shown in Figure 6.1, for the task of language modeling on the Wikipedia dataset, LLA achieves comparable accuracy to LSTM while being as sparse as LDA in terms of models size. We give an efficient inference algorithm for parameter inference in LLA (Sec. 6.3.2) and show is efficacy and interpretability over several datasets (Sec. 6.4).

6.2 BACKGROUND

In this section, we provide a brief review of user/language modeling and LSTMs.

6.2.1 User/Language Modeling

User activity modeling and language modeling amounts to learning a function that computes the log probability, \( \log p(w|\text{model}) \), of a user activity or sentence \( w = [w_1, \ldots, w_n] \). Subsequently, this function can be used to predict the next set of actions or words. This function can be decomposed and learned in different ways under various assumptions. Imposing a bag of words assumption - as used in LDA - leads to ignoring the sequence information and yields \( \sum_{i=1}^{n} \log p(w_i|\text{model}) \). Alternatively, one could decompose according to the chain rule into sum of the conditional log probabilities \( \sum_{i=1}^{n} \log p(w_i | w_1, \ldots, w_{i-1}, \text{model}) \), thereby preserving temporal information and use some RNN to model \( \log p(w_i | w_1, \ldots, w_{i-1}, \text{model}) \) Mikolov, Karafiát, et al., 2010; Sundermeyer, Schlüter, and Ney, 2012.

6.2.2 Long Short-Term Memories

Temporal aspect is very important for user activity modeling. LSTM, a type of RNN, is well suited for the task as it can learn from experience to classify, process, and predict time series when there are very long time lags of unknown size between important events. LSTM is designed to cope with the vanishing gradient problem inherent in simple RNNs Hochreiter
and Jürgen Schmidhuber, 1997. This is one of the main reasons why LSTMs outperform simple RNNs, hidden Markov models, and other sequence learning methods in numerous application.

In general, a RNN is a triplet \((\Sigma, S, \delta)\):

- \(\Sigma \subseteq \mathbb{R}^D\) is the input alphabet
- \(S \subseteq \mathbb{R}^H\) is the set of states
- \(\delta : S \times \mathbb{R} \rightarrow S\) is the state transition function made up of a neural network.

RNN maintains an internal state and at each time step take an input \(x_t\) and updates its state \(s_t\) by applying the transition function made up of neural network \(\delta\) the previous time step’s state \(s_{t-1}\) and the input.

Often the input is not available directly as elements of \(\Sigma\) and the output desired is not the state of the RNN. In such cases, input is appropriately transformed and desired output is produced at each time step from the state \(s_t\):

\[
y_t = g(s_t),
\]

where \(g\) is an arbitrary differentiable function.

For example, in a regular recurrent language model (RRLM) Mikolov, Karafiát, et al., 2010 a document is treated as a sequence and an LSTM is trained to predict directly the next word conditioned on the sequence of words before it, i.e. maximize \(p(w_t|w_{t-1}, w_{t-2}, ..., w_0; \text{model})\).

In this case, the input transformation is done by using a word lookup table from word to a vector in \(\Sigma\). This word representation is used to update the state of the LSTM. The output transformation is the projection of \(s_t\) into a vector of size of the vocabulary \(V\) followed by a softmax. However, this method will require two large matrices of dimension \(V \times H\) and cannot handle out of vocabulary words.

To overcome above mentioned shortcomings, another input transformation has been proposed, which looks at characters directly instead of words Ling et al., 2015. On the spelling of the words another LSTM is run having characters as the input and the final state is used as the word representation. In this case, the memory requirement is reduced to half and the model can handle out-of-vocabulary words like typographical errors or new words made from composing existing ones. To use character level embedding, we can simply replace the word lookup table with the representation obtained from character-level LSTM. In case of natural languages, a similar trick of using a character-level LSTM to emit words can be applied as the output transformation as well. However, user modeling outputs (hash indices and URLs) lack morphological structure, and hence cannot leverage the technique.

### 6.3 Latent LSTM Allocation

In this section, we provide a detailed description of the proposed LLA model and its variant for performing joint clustering and modeling non-linear dynamics. An ideal model for such a task should have three characteristics. First, it should have a low number of parameters. Second, it should be interpretable, allowing human analysis of the components. Lastly and most importantly, the model should be accurate in terms of predicting future events. We show how LLA satisfies all of these requirements.
6.3.1 Generative model

In this model, we try to marry LDA with its strong interpretability capabilities with LSTM having excellent track record in capturing temporal information. In this regard, we propose a factored model, i.e. we use the LSTM to model sequence of topics $p(z_t|z_{t-1}, z_{t-2}, ..., z_1)$ and multinomial-Dirichlet to model word emissions $p(w_i|z_i)$, similar to LDA. Suppose we have K topics, vocabulary size V, and a document collection $D$ where each document $d \in D$ is composed of $N_d$ words. With these notations, the complete generative process can be illustrated as in Figure 6.2a and can be described as:

1. for $k = 1$ to K
   a) Choose topic $\phi_k \sim \text{Dir}(\beta)$

2. for each document $d$ in corpus $D$
   a) Initialize LSTM with $s_0 = 0$
   b) for each word index $t$ from 1 to $N_d$
      i. Update $s_t = \text{LSTM}(z_{d,t-1}, s_{t-1})$
      ii. Get topic proportions at time $t$ from the LSTM state, $\theta = \text{softmax}_K(W_p s_t + b_p)$
      iii. Choose a topic $z_{d,t} \sim \text{Categorical}(\theta)$
      iv. Choose word $w_{d,t} \sim \text{Categorical}(\phi_{z_{d,t}})$

Under this model, the marginal probability of observing a given document $d$ can be written as:

$$p(w_d|\text{LSTM}, \phi) = \sum_{z_d} p(w_d, z_d|\text{LSTM}, \phi)$$

$$= \sum_{z_d} \prod_t p(w_{d,t}|z_{d,t}; \phi)p(z_{d,t}|z_{d,1:t-1}; \text{LSTM}).$$

Here $p(z_{d,t}|z_{d,1:t-1}; \text{LSTM})$ is the probability of generating topic for the next word in the document given topics of previous words and $p(w_{d,t}|z_{d,t}; \phi)$ is the probability of generating...
word given the topic, illustrating the simple modification of LSTM and LDA based language models.

The advantage of this modification is two fold. Firstly, we obtain a factored model as shown in Figure 6.3, thereby the number of parameters is reduced significantly compared to RRLM. This is because, unlike RRLM we operate at topic level instead of words directly and the number of topics is much smaller than the vocabulary size, i.e., $K << V$. This allows us to get rid of large $V \times H$ word embedding look-up table and $V \times H$ matrix used in softmax over the entire vocabulary. Instead we map from hidden to state to topics first using a $K \times H$ matrix, which will be dense and then from topics to words using a $V \times K$ matrix under the Dirichlet-multinomial model similar to LDA which will be very sparse. Secondly, this model is highly interpretable. We recover global themes present in the documents as $\phi$. The LSTM output represents topic proportions for document/user at time $t$. The LSTM input over topics can capture semantic notion of topics.

### 6.3.2 Inference

As the computation of the true posterior of LLA as described above is intractable, we have to resort to an approximate inference technique like mean field variational inference Wainwright and Michael I Jordan, 2008 or stochastic EM (SEM) Zaheer, Wick, et al., 2015. Below we describe the generic aspects of the inference algorithm for LLA. Model specific aspects are relegated to the appendix. ¹

Given a document collection, the inference task is to find the LSTM weights and word|topic probability matrix $\phi$. This can be carried out using SEM. We begin by writing out the likelihood and lower bounding it as,

---

¹ Available at [http://manzil.ml/lla.html](http://manzil.ml/lla.html)
\[
\sum_d \log p(w_d|\text{LSTM}, \phi) \\
\geq \sum_d \sum_{z_d} q(z_d) \log \frac{p(z_d; \text{LSTM}) \prod_t p(w_{d,t}|z_{d,t}; \phi)}{q(z_d)}, \tag{6.1}
\]

for any distribution \(q\). Then the goal is to ensure an increase in this evidence lower bound (ELBO) with each iteration in expectation. Following the suit of most EM based inference methods, each iteration involves two phases, each of which is described below:

**SE-step:** For fixed LSTM parameters and \(\phi\), we sample the topic assignments \(z\). This acts as an unbiased sample estimate of the expectation with respect to the conditional distribution of \(z\) required in traditional EM. This sampled estimate not only enjoys similar statistical convergence guarantees Nielsen, 2000a but also provides many computational benefits like speed-up and reduced memory bandwidth requirements Zaheer, Wick, et al., 2015.

Under LLA, the conditional probability for topic at time step \(t\) given word at time \(t\) and past history of topics is,

\[
p(z_{d,t} = k|w_{d,t}, z_{d,1:t-1}; \text{LSTM}, \phi) \propto p(z_{d,t} = k|z_{d,1:t-1}; \text{LSTM}) p(w_{d,t}|z_{d,t} = k; \phi)
\]

The first term represents probability of various topics predicted by LSTM dynamics after final softmax and second term comes from multinomial-Dirichlet word emission model given the topic. Naïvely sampling from this distribution would cost \(O(KH + H^2)\) per word.

*Optionally*, one could speed up this sampling. Note that the second term in (6.2) has a sparse structure. For sampling from (6.2) the computation of the normalizer is not essential. To elaborate, let us define \(n_{wk}\) as the number of times word \(w\) currently assigned to topic \(k\) and \(n_k\) as the number of tokens assigned to topic \(k\).

Explicitly writing down first term:

\[
p(w_{d,t} = w|z_{d,t} = k, \phi) = \phi_{wk} = \frac{n_{wk} + \beta}{n_k + V\beta}
\]

\[
= \frac{n_{wk}}{n_k + V\beta} + \frac{\beta}{n_k + V\beta} \tag{6.3}
\]

There is an inherent sparsity present in \(n_{wk}\), as a given word would be typically about only a handful of topics, \(K_w \ll K\). The second term represents the global count of tokens in the topic and will be dense, regardless of the number of documents, words or topics.

Following Aaron Q Li et al., 2014, we devise an *optional* fast sampling strategy for exploiting this sparsity and construct a Metropolis-Hastings sampler. The idea is to replace the low weight dense term by a cheap approximation, while keeping the high weight sparse term exact. This leads to a proposal distribution that is close to (6.2), while at the same time allowing us to draw from it efficiently:

- First draw a biased coin to decide whether to draw from the sparse \(n_{wk}\) term or from the dense term. The bias is

\[
b = \frac{p}{p + q}, \quad \text{where}
\]

\[
p = \sum_{k:n_{wk} \neq 0} \frac{n_{wk}}{n_k + V\beta} p(z_{d,t} = k|z_{d,1:t-1}; \text{LSTM})
\]

\[
q = \sum_k \frac{\beta}{n_k + V\beta} \quad \text{(pre-computed)}.
\]
• If we draw from the sparse term, the cost is $O(K_wH + H^2)$, else the cost is $O(H^2)$ using the alias trick.
• Finally, perform a MH accept/reject move by comparing the approximation with the true distribution.

**M-step:** For fixed topic assignment $z$, update the $\phi$ and LSTM so as to improve the likelihood in expectation. As the dependence of likelihood on LSTM parameters and $\phi$ are independent given $z$, we can update them independently and in parallel. For the former, we use the closed form expression for the maximizer,

$$
\phi_{wk} = \frac{\#(w_{d,t} = w \text{ and } z_{d,t} = k)}{\#(n_{wk} = k) + V\beta} = \frac{n_{wk} + \beta}{n_k + V\beta},
$$

and for the latter we resort to stochastic gradient ascent which will increase the likelihood in expectation.

The execution of the whole inference and learning process includes several iterations involving the above mentioned two phases as outlined in Algorithm 1. This inference procedure is not an ad-hoc method, but an instantiation of the well studied SEM method. We ensure that in each iteration we increase the ELBO in expectation. Also, it is just a coincidence that for all such latent variable models the equations for SE-step looks like Gibbs sampling. (Exploiting this coincidence, in fact, one can prove that parallel Gibbs update will work for such models, cf Tassarotti and Steele Jr, 2015; Zaheer, Wick, et al., 2015 whereas in general parallel Gibbs sampling fails miserably e.g. for Ising models.) Moreover, we found empirically that augmenting the SEM inference with a beam search improved the performance.

Automatic differentiation software packages such as TensorFlow Martin Abadi et al., 2015 significantly speed up development time, but have limited control structures and indexing. Whereas random sampling requires varied control statements, thus was implemented separately on the CPU. Furthermore, the SEM saved us precious GPU-CPU bandwidth by transmitting only an integer per token instead of a K-dimensional variational parameter.

### 6.3.3 Adding Context

Utilizing the word or user action, $w_{d,t}$, directly would be more informative to model the dynamics and predict the next topic. For example, rather than only knowing the previous action belonged to “electronics purchase” topic, knowing exactly that a camera was bought, makes it easier to predict user’s next interest, e.g. camera lenses.

In order to incorporate the exact context, we construct a variant of LLA, called **word LLA**, where the LSTM is provided with an embedding of previous word $w_{d,t-1}$ directly instead of $z_{d,t-1}$. The corresponding graphical model is shown in Figure 6.2b and the joint likelihood is:

$$
\sum_d \log p(w_d | \text{LSTM}, \phi) = \sum_d \sum_t \log \sum_{z_{d,t}} p(w_{d,t} | z_{d,t}; \phi) p(z_{d,t} | w_{d,1:t-1}; \text{LSTM})
$$

A SEM inference strategy can be devised similar to that of topic LLA as presented in section 3.2.

However, this modification brings back the model to the dense and high number of parameter regime as a large trainable $V \times H$ lookup table for the word embeddings is needed for the input transformation. This problem can be mitigated by using char-level LSTM (**char LSTM** for
Algorithm 1 Stochastic EM for LLA

Input: Document corpus $D$.

1: Initialize $\phi$ and LSTM with a few iterations of LDA
2: repeat
   SE-Step:
   3: for all document $d \in D$ in parallel do
   4:      for $t \leftarrow 1$ to $N_d$ (possibly with padding) do
   5:         \[
         \pi_k = \phi_{w,d,t}^k p(z_{d,t} = k | z_{d,1:t-1}; \text{LSTM}).
         \]
   6:         Sample $z_{d,t} \sim \text{Categorical}(\pi)$
   7:      end for
   8:   end for
   M-Step:
   9:   Collect sufficient statistics to obtain:
   10:      $\phi_{wk} = \frac{n_{wk} + \beta}{n_k + V\beta}$, \quad $\forall w, k$
   11:   for mini-batch of documents $B \subset D$ do
   12:      Compute the gradient by LSTM backward pass
   13:         \[
         \frac{\partial \mathcal{L}}{\partial \text{LSTM}} = \sum_{d \in B} \sum_{t=1}^{N_d} \frac{\partial \log p(z_{d,t} | z_{d,1:t-1}; \text{LSTM})}{\partial \text{LSTM}}
         \]
   14:      Update LSTM parameters by stochastic gradient descent methods such as Adam D. P. Kingma and Ba, 2014.
   15:   end for
   until Convergence

short) to construct the input transformation. Such character-level LSTM have recently shown promising results in generating structured text Karpathy, J. Johnson, and Fei-Fei, 2015 as well as in producing semantically valid word embeddings with a model we will refer to as char-LSTM) Ling et al., 2015. The character-level models do not need the huge lookup table for words, as they operate directly on the characters and the number of distinct characters is extremely small. We chose the latter as our input transformation and briefly describe it below.

The input word $w$ is decomposed into a sequence of characters $c_1, \ldots, c_m$, where $m$ is the length of $w$. Each character $c_i$ is transformed using a lookup table and fed into a bidirectional LSTM The forward LSTM evolves on the character sequence and produces a final state $h^f_m$. While the backward LSTM receives as input the reverse sequence, and yields its own final state $h^b_0$. Both LSTMs use a different set of parameters. The representation $e^c_w$ of word $w$ is obtained by combining forward and backward final states:

\[
    e^c_w = D^f s^f_m + D^b s^b_0 + b_d, \tag{6.4}
\]

where $D^f$, $D^b$ and $b_d$ are parameters that determine how the states are combined. This $e^c_w$ is provided to the topic level LSTM as the input providing information about the word. Thus we are able to incorporate more context information by providing some information about the previous word without the need to have great number of parameters. We call this model, char LLA.

The different variants of LLA and LSTM as language model can be unified and thought of having different input and out transformations over LSTM for capturing the dynamics. The possible combinations are listed in Table 7.1. We will study each of them empirically next.
### 6.4 Experimental Results

We perform a comprehensive evaluation of our model against several deep models, dynamic models, and topic models. For reproducibility we focus on the task of language modeling over the publicly available Wikipedia dataset, and for generality, we show additional experiments on the less-structured domain of user modeling.

#### 6.4.1 Setup

For all experiments we follow the standard setup for evaluating temporal models, i.e. divide each document (user history) into 60% for training and 40% for testing. The task is to predict the remaining 40% of the document (user data) based on the representation inferred from the first 60% of the document. We use perplexity as our metric (lower values are better) and compare our models (topic-LLA, word-LLA, and char-LLA) against the following baselines:

<table>
<thead>
<tr>
<th>Model</th>
<th>Input Transformation</th>
<th>Output Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>word LSTM</td>
<td>Word Embedding</td>
<td>Softmax over vocabulary</td>
</tr>
<tr>
<td>Char LSTM</td>
<td>Char Embedding</td>
<td>Softmax over vocabulary</td>
</tr>
<tr>
<td>Topic LLA</td>
<td>Topic Embedding</td>
<td>Topic based</td>
</tr>
<tr>
<td>Word LLA</td>
<td>Word Embedding</td>
<td>Topic based</td>
</tr>
<tr>
<td>Char LLA</td>
<td>Char Embedding</td>
<td>Topic based</td>
</tr>
</tbody>
</table>

Table 6.1: Enumerating different input and output representation, leading to variants of LLA models. Note we exclude char level output transformation due to its inapplicability to user modeling.

Figure 6.4: figure

Test perplexity and number of parameters of various models. Bars represent model sizes and the solid curve represents perplexity over testset (lower is better).

Figure 6.5: The effect of the number of topics over the performance of LDA and LLA.
• **Autoencoder**: A feed forward neural network that comprises of encoder and decoder. The encoder projects the input data into a low-dimensional representation and the decoder reconstructs the input from this representation. The model is trained to minimize reconstruction error.

• **LSTMs**: We consider both word-level LSTM language model (word-LSTM) and character-level hierarchical LSTM (char-LSTM) language model.

• **LDA**: The vanilla version trained using collapsed Gibbs sampling over the bag of word representation.

• **ddLDA**: Distance-dependent LDA is a variant of LDA incorporating the word sequence. It is based on a fixed-dimensional approximation of the distance-dependent Dirichlet process David Blei and Frazier, 2011. In this model a word is assigned to a topic based on the popularity of the topics assigned to nearby words. Note that this model subsumes other LDA-based temporal models such as hidden Markov topic Model Andrews and Vigliocco, 2010 and RCRP based models Ahmed and Xing, 2008.

We trained all deep models using stochastic gradient decent with Adam D. P. Kingma and Ba, 2014. All LSTM and LLA based models used the sampled softmax method for efficiency Cho, Memisevic, and Y. Bengio, 2015. Finally, all hyper-parameters of the models were tuned over a development set.

### 6.4.2 Language modeling

We used the publicly available Wikipedia dataset to evaluate the models on the task of sequence prediction. Extremely short documents, i.e. documents having length less than 500 words were excluded. The dataset contains ~0.7 billion tokens and ~1 million documents. The most frequent 50k word-types were retained as our vocabulary. Unless otherwise stated, we used 1000 topics for LLA and LDA variants. For LSTM and LLA variants, we selected the dimensions of the input embedding (word or topic) and evolving latent state (over words or topics) in the range of \{50, 150, 250\}. In case of character-based models, we tuned the dimensions of the character embedding and latent state (over characters) in the range of \{50, 100, 150\}.

**accuracy vs model size** Figure 6.4(a) compares model size in terms of number of parameters with model accuracy in terms of test perplexity. As shown in the figure, LDA yields the smallest model size due to the sparsity bias implicit in the Dirichlet prior over the topic/word distributions. On the other hand, word-LSTM gives the best accuracy due to its ability to utilize word level statistics; however, the model size is order of magnitudes larger than LDA. Char-LSTM achieves almost 50% reduction in model size over word-LSTM at the expense of a slightly worse perplexity. The figure also shows that LLA variants (topic-LLA, word-LLA, and char-LLA) can trade accuracy vs model size while still maintaining interpretability since the output of the LSTM component is always at the topic level. Note that the figure depicts the smallest word-LSTM at this perplexity level, as our goal is not to beat LSTM in terms of accuracy, but rather to provide a tuning mechanism that can trade-off perplexity vs model size while still maintaining interpretability. Finally, compared to LDA, which is a widely used tool, LLA variants achieve a significant perplexity reduction at a modest increase in model size while still maintaining topic sparsity. As shown in Figure 6.1, the autoencoder model performs poorly in terms of model size and perplexity thus we eliminated it from Figure 6.4(a) and the following figures to avoid cluttering the display.

**convergence over time** At first glance, LLA seems to be more involved than both LDA and LSTM. So, we raise the question whether the added complexity leads to a slower training.
Figure 6.8 shows that compared to LSTM based models, LLA variants achieve comparable convergence speed. Moreover, compared to fast LDA samplers Zaheer, Ahmed, et al., 2017, our LLA variants introduce only a modest increase in training time. The figure also shows that character based models (char-LSTM and char-LLA) are slightly slower to train compared to word level variants due to their nested nature and the need to propagate gradients over both word and character level LSTMs.

**ablation study** Since both LDA and LLA result in interpretable models, we want to explore if LDA can achieve a perplexity similar to a given LLA model by just increasing the number of topics in LDA. Figure 6.5 shows the performance of LDA and variants of LLA for different number of topics. As can be seen from the figure, even with 250 topics, all LLA based models achieve much lower perplexity than LDA with 1000 topics. In other words, LLA improves over LDA not because it uses a slightly larger model, but rather because it models the sequential order in the data.

**interpretability** Last but not least, we demonstrate the interpretability of our models. Similar to LDA, the proposed LLA also uncovers global themes, a.k.a topics prevalent in the dataset. We found qualitatively the topics produced by LLA to be cleaner. For example, in Table 6.2 we show a topic about funding, charity, and campaigns recovered by both. LDA includes spurious words like Iowa in the topic just because it co-occurs in the same documents. Whereas modeling the temporal aspect allows LLA to correctly switch topics at different parts of the sentence producing cleaner topic regarding the same subject.

Modeling based on only co-occurrences in LDA leads to further issues. For example, strikes among mine workers are common, so the two words will co-occur heavily but it does not imply that strikes and mining should be in the same topic. LDA assign both the words in the same topic as shown in Table 6.3. Modeling sentences as an ordered sequence allows distinction between the subjects and objects in the sentence. In the previous example, this leads to factoring into two separate topics of strikes and mine workers.

The topic LLA provides embedding of the topics in a Euclidean metric space. This embedding allows us to understand the temporal structure learned by the model: topics close in this space are expected to appear in close sequential proximity in documents. To understand this, we built a topic hierarchy using the embeddings which uncovers interesting facts about the data. For example in Figure 6.6a, we show a portion of the topic hierarchy discovered by topic-LLA with 1000 topics. For each topic we list the top few words. The theme of the figure is countries in Asia. It groups topics relating to one country together and puts topics related to neighboring countries close by. Three prominent clusters are shown form top to bottom which corresponds to moving from west to east on the map of Asia. Top cluster is about Arab world, second one represent the Indian sub-continent, and the third one starts with south-east Asia like Philippines. (The topic abs gma cbn represents TV station in south-east Asia.) The complete hierarchy of topic is very meaningful and can be viewed at [http://manzil.ml/lla.html](http://manzil.ml/lla.html).

### 6.4.3 User modeling

We use an anonymized sample of user search click history to measure the accuracy of different models on predicting users’ future clicks. An accurate model would enable better user experience by presenting the user with relevant content. The dataset is anonymized by removing all items appearing less than a given threshold, this results in a dataset of ~50 million tokens, and 40K vocabulary size. This domain is less structured than the language modeling task since users’ click patterns are less predictable than the sequence of words which follow definite syn-
(a) Wikipedia dataset
(b) user search click history

Figure 6.6: Topic embedding discovered using LLA

(a) LLA-variant against vanilla LDA
(b) LLA-variants against LSTM-variants

Figure 6.8: Convergence speed for various models. The x-axis represents time in second while the Y-axis gives the perplexity over the test dataset.
### Table 6.2: Cleaner topics produced by LLA vs LDA

<table>
<thead>
<tr>
<th>LDA</th>
<th>Fund, foundation, iowa, charity, fund, money, campaign, raised, donated, funds, donations, raise, support, charitable, million, donation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLA</td>
<td>Fund, foundation, money, funds, support, charity, funding, donations, campaign, raised, donated, trust, raising, contributions, awareness</td>
</tr>
</tbody>
</table>

tactic rules. We used a setup similar to the one used in the experiments over the Wikipedia dataset for parameters ranges and selections.

Figure 6.4(b) gives the same conclusion as Figure 6.4(a): LLA variants achieve significantly lower perplexity compared to LDA and at the same time they are considerably smaller models compared to LSTM. Furthermore, even compared to ddLDA, an improved variant of LDA, our proposed LLA achieves better perplexity. ddLDA models time by introducing smoothness constraints that bias future actions to be generated from recent topics; however, it does not possess a predictive action model. As a result, it can neither model the fact that “booking a flight” topic should not be repeated over a short course of time nor that “booking a hotel” topic would likely follow shortly, but not necessarily immediately, after “booking a flight” topic. LLA, on the other hand, is capable of capturing this intricate dynamics by modeling the evolution of user topics via an LSTM – an extremely powerful dynamic model. This point is more evident if we consider the following hand-crafted\(^2\) user click trace in context of the topics depicted in Figure 6.6b:

```
```

There are four topics represented and color-coded (best viewed in color): wedding (sixth topic from top), social media (fourth from top), food (third from bottom) and home improvement (second from top) – all in Figure 6.6b. We asked each model to predict what would be the three most likely topics that would appear next in current user’s session. LDA predicted wedding as the top topic followed by a tie for the remaining topics. ddLDA, whose output is based on exponential decay, yields wedding as most likely, followed by doityourself topic, and then the food topic as expected. In contrast, LLA ranks the most likely three topics as: doityourself topic, houzz topic (top topic in Figure 6.6b), and the wedding topic. This is indicative of LLA learning that once a user starts in the doityourself topic, the user will stay longer in this part of the tree for the task and wander in the nearby topics for a while. Moreover, LLA still remembers the wedding topic, and unlike other models, LLA did not erroneously pick neighbors of the wedding topic among the three most likely topics to follow.

Finally, to demonstrate the interpretability of our model, in Figure 6.6b, we show a portion of the topic hierarchy discovered by topic-LLA with 250 topics. For each topic we list the top few clicked items. The theme of the figure is wedding and various house chores. Three prominent clusters are shown from top to bottom. The top cluster is about house renovations and wedding. Second cluster captures makeup, designer shoes, and fashion. The bottom cluster capture kitchen, cooking and gardening. It is clear form the hierarchy that LLA groups topics into the embedding space based on sequential proximity in the search click history.

\(^2\) For privacy concerns, we construct an artificial example manually for illustration purposes.
Table 6.3: Factored topics produced by LLA vs LDA

<table>
<thead>
<tr>
<th></th>
<th>Independent learner</th>
<th>Joint learner</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>strike, strikes, striking, miners, strikers, workers, day, began, general, called, pinkerton, action, hour, hunger, keefe</td>
<td></td>
</tr>
<tr>
<td>LLA</td>
<td>union, unions, strike, workers, labor, federation, trade, afl, bargaining, cio, organization, relations, strikes, national, industrial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mining, coal, mine, mines, gold, ore, miners, copper, iron, rush, silver, mineral, deposits, minerals, mined</td>
<td></td>
</tr>
</tbody>
</table>

6.4.4 Effect of Joint Training

One might wonder what would happen if we first train LDA and then simply train LSTM over the topic assignments from the LDA. We refer to this baseline as “independent learner” since LDA and LSTM are trained independently. In Table 6.4, we compare its performance against LLA, which jointly learns the topics and the evolution dynamics. As we can see, joint training is significantly better, since the LSTM will fit the random errors introduced by the topic assignments inferred from the LDA model, and in fact LSTM will learn to be as good as the sequence produced by an LDA model (which is time-oblivious to start with). The effect is more pronounced in the user history data due to the unstructured nature of this domain.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Independent learner</th>
<th>Joint learner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikipedia</td>
<td>2119</td>
<td>1785</td>
</tr>
<tr>
<td>User Click</td>
<td>1572</td>
<td>927</td>
</tr>
</tbody>
</table>

Table 6.4: Advantage in terms of perplexity for joint learning.

6.5 RELATED WORKS & DISCUSSIONS

In this paper we present LLA: Latent LSTM allocation. LLA leverages the powerful dynamic modeling capability of LSTM without blowing up the number of parameters while adding interpretability to the model. We achieve this by shifting from modeling the temporal dynamics at the observed word level to modeling the dynamics at a higher level of abstraction: topics. As the number of topics \( K \) is much smaller than the number of words \( V \), it can act as a knob that can trade-off accuracy vs model size. In the extreme case, if we allow \( K \rightarrow V \) in LLA then we recover LSTM. Furthermore, the topics provide an informative embedding that can reveal interesting temporal relationship as shown in Figure 6.6a and 6.6b – which is a novel contribution to the best of our knowledge.

Our work is related to various dynamic topic models, however, previous works like ddLDA (ddCRP in general) or hidden Markov topic models provide only limited modeling capabilities of the temporal dynamics. Specifically, they impose smoothness constraints: i.e. topic of a word is biased toward nearby topics. This constraint cannot learn a predictive action model, e.g. after “booking a flight” topic, the “booking a hotel” topic is likely to follow. Moreover, Internet users often wander between related activities, e.g. “booking a hotel” topic will happen shortly after “booking a flight”, but not necessarily immediately as the user might have been interrupted by something else. Thus we need a much richer temporal model such as an LSTM. Our quantitative results against ddLDA confirm this claim.
Another similar work would be lda2vec Moody, 2016, where LDA is combined with local context in a fixed window size. This provides a sparse, interpretable model but lacks modeling the temporal aspect. The model cannot be used in a predictive-action setting. Also the sparsity is only in terms of per document parameters, whereas it suffers from huge dense of size vocabulary times embedding size.

Another relevant recent work is Sentence Level Recurrent Topic Model (SLRTM) Tian, Gao, and T.-Y. Liu, 2016. However, this model cannot capture long-range temporal dependencies, as the topic for each sentence is still decided using an exchangeable (non-sequential) Dirichlet-multinomial scheme similar to the LDA. It might be able to preserve local sentence structure or grammar, but that is particularly not very useful for the task of user modeling. Furthermore, as it contains RNN that operates on the entire vocabulary (not topic as in our case), the SLTRM has lots of parameters.

Finally our work is related to recent work in recurrent latent variable models Chung et al., 2015; Gemici et al., 2017 where a recurrent model is endowed with latent variables to model variability in the input data. However, they mainly focused on continuous input space such as images and speech data which enables the use of variational autoencoder techniques to learn the model parameters. Whereas in this paper, we focus on discrete data that are not amenable to the standard variational autoencoder techniques and as such we developed an efficient SEM algorithm instead.
Part III

SCALING UP INFERENCE BY EXPLOITING TOPOLOGY
Parallelization

We propose an embarrassingly parallel, memory efficient inference algorithm for latent variable models in which the complete data likelihood is in the exponential family. The algorithm is a stochastic cellular automaton and converges to a valid maximum a posteriori fixed point. Applied to latent Dirichlet allocation we find that our algorithm is over an order of magnitude faster than the fastest current approaches. A simple C++/MPI implementation on a 20-node Amazon EC2 cluster samples at more than 1 billion tokens per second. We process 3 billion documents and achieve predictive power competitive with collapsed Gibbs sampling and variational inference.

7.1 Introduction

In the past decade, frameworks such as stochastic gradient descent (SGD) Herbert Robbins and Sutton Monro, 1951 and map-reduce Dean and Ghemawat, 2008 have enabled machine learning algorithms to scale to larger and larger datasets. However, these frameworks are not always applicable to Bayesian latent variable models with rich statistical dependencies and intractable gradients. Variational methods Michael I. Jordan et al., 1999 and Markov chain Monte-Carlo (MCMC) Gilks, S. Richardson, and Spiegelhalter, 1995 have thus become the sine qua non for inferring the posterior in these models.

Sometimes—due to the concentration of measure phenomenon associated with large sample sizes—computing the full posterior is unnecessary and maximum a posteriori (MAP) estimates suffice. It is hence tempting to employ gradient descent. However, for latent variable models such as latent Dirichlet allocation (LDA), calculating gradients involves expensive expectations over rich sets of variables Patterson and Y. W. Teh, 2013.

MCMC is an appealing alternative, but traditional algorithms such as the Gibbs sampler are inherently sequential and the extent to which they can be parallelized depends heavily upon how the structure of the statistical model interacts with the data. For instance, chromatic sampling Gonzalez et al., 2011 is infeasible for LDA, due to its dependence structure. We propose an alternate approach based on stochastic cellular automata (SCA). The automaton is massively parallel like conventional cellular automata, but employs stochastic updates.

<table>
<thead>
<tr>
<th>mix. component/emitter</th>
<th>Bernoulli</th>
<th>Multinomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categorical</td>
<td>Latent Class</td>
<td>Unigram Document</td>
</tr>
<tr>
<td></td>
<td>Model Formann and Kohlmann, 1996</td>
<td>Clustering</td>
</tr>
<tr>
<td>Dirichlet Mixture</td>
<td>Grade of Membership</td>
<td>Latent Dirichlet</td>
</tr>
</tbody>
</table>
Our proposed algorithm, exponential SCA (ESCA), is a specific way of mapping inference in latent variable models with complete data likelihood in the exponential family into an instance of SCA. ESCA has a minimal memory footprint because it stores only the data and the minimal sufficient statistics (by the very definition of sufficient statistics, the footprint cannot be further reduced). In contrast, variational approaches such as stochastic variational inference (SVI) M. D. Hoffman et al., 2013 require storing the variational parameters, while MCMC-based methods, such as YahooLDA Smola and Narayanamurthy, 2010 require storing the latent variables. Thus, ESCA substantially reduces memory costs, enabling larger datasets to fit in memory, and significantly reducing communication costs in distributed environments.

Furthermore, the sufficient statistics dramatically improves efficiency. Typically, in the general case of SCA, updating a cell requires first assembling the values of all the neighboring cells before aggregating them into a local stochastic update. However, in ESCA, the sufficient statistics adequately summarize the states of the neighbors; the computational load is therefore small and perfectly balanced across the cells.

We demonstrate how ESCA is a flexible framework for exponential latent variable models such as LDA. In our experiments, we process over 3 billion documents at a rate of 570 million tokens per second on a small cluster of 4 commodity servers. That said, ESCA is much more general. Table 7.1 explicitly lists some of the more common modeling choices for which ESCA can be easily employed. Our algorithm implicitly simulates stochastic expectation maximization (SEM), and is thus provably correct in the sense that it converges in distribution to a stationary point of the posterior.

### 7.2 Exponential SCA

Stochastic cellular automata (SCA), also known as probabilistic cellular automata, or locally-interacting Markov chains, are a stochastic version of a discrete-time, discrete-space dynamical system in which a noisy local update rule is homogeneously and synchronously applied to every site of a discrete space. They have been studied in statistical physics, mathematics, and computer science, and some progress has been made toward understanding their ergodicity and equilibrium properties. A recent survey Mairesse and Marcovici, 2014 is an excellent introduction to the subject, and a dissertation Louis, 2002 contains a comprehensive and precise presentation of SCA.

The automaton, as a (stochastic) discrete-time, discrete-space dynamical system, is given by an evolution function \( \Phi : S \rightarrow S \) over the state space \( S = \mathbb{Z} \rightarrow C \) which is a mapping from the space of cell identifiers \( \mathbb{Z} \) to cell values \( C \). The global evolution function applies a local function \( \phi_\mathbb{Z}(c_1, c_2, \ldots, c_r) \rightarrow c \) s.t. \( c_i = s(\mathbb{Z}_i) \) to every cell \( \mathbb{Z} \in \mathbb{Z} \). That is, \( \phi \) examines the values of each of the neighbors of cell \( \mathbb{Z} \) and then stochastically computes a new value \( c \). The dynamics begin with a state \( s_0 \in S \) that can be configured using the data or a heuristic.

Exponential SCA (ESCA) is based on SCA but achieves better computational efficiency by exploiting the structure of the sufficient statistics for latent variable models in which the complete data likelihood is in the exponential family. Most importantly, the local update function \( \phi \) for each cell depends only upon the sufficient statistics and thus does not scale linearly with the number of neighbors.
7.2.1 Latent Variable Exponential Family

Latent variable models are useful when reasoning about partially observed data such as collections of text or images in which each i.i.d. data point is a document or image. Since the same local model is applied to each data point, they have the following form

\[ p(z, x, \eta) = p(\eta) \prod_i p(z_i, x_i | \eta). \] (7.1)

Our goal is to obtain a MAP estimate for the parameters \( \eta \) that explain the data \( x \) through the latent variables \( z \). To expose maximum parallelism, we want each cell in the automaton to correspond to a data point and its latent variable. However, in general all latent variables depend on each other via the global parameters \( \eta \), and thus the local evolution function \( \phi \) would have to examine the values of every cell in the automaton.

Fortunately, if we further suppose that the complete data likelihood is in the exponential family, i.e.,

\[ p(z_i, x_i | \eta) = \exp (\langle T(z_i, x_i), \eta \rangle - g(\eta)) \] (7.2)

then the complete and sufficient statistics are given by

\[ T(z, x) = \sum_i T(z_i, x_i) \] (7.3)

and we can thus express any estimator of interest as a function of just \( T(z, x) \). Further, when employing expectation maximization (EM), the M-step is possible in closed form for many members of the exponential family. This allows us to reformulate the cell level updates to depend only upon the sufficient statistics instead of the neighboring cells. The idea is that, unlike SCA (or MCMC in general) which produces a sequence of states that correspond to complete variable assignments \( s^0, s^1, \ldots \) via a transition kernel \( q(s^{t+1} | s^t) \), ESCA produces a sequence of sufficient statistics \( T^0, T^1, \ldots \) directly via an evolution function \( \Phi(T^t) \mapsto T^{t+1} \).

7.2.2 Stochastic EM

Before we present ESCA, we must first describe stochastic EM (SEM). Suppose we want the MAP estimate for \( \eta \) and employ a traditional expectation maximization (EM) approach:

\[ \max_{\eta} p(x, \eta) = \max_{\eta} \int p(z, x, \eta) \mu(dz) \]

EM finds a mode of \( p(x, \eta) \) by iterating two steps:

**E-step** Compute in parallel \( p(z_i | x_i, \eta^{(t)}) \).

**M-step** Find \( \eta^{(t+1)} \) that maximizes the expected value of the log-likelihood with respect to the conditional probability, i.e.

\[ \eta^{(t+1)} = \arg \max_{\eta} \mathbb{E}_{z|x, \eta^{(t)}}[\log p(z, x, \eta)] \]

\[ = \xi^{-1} \left( \frac{1}{n + n_0} \sum_i \mathbb{E}_{z|x, \eta^{(t)}}[T(z_i, x_i)] + T_0 \right) \]

where \( \xi(\eta) = \nabla g(\eta) \) is invertible as \( \nabla^2 g(\theta) > 0 \) and \( n_0, T_0 \) parametrize the conjugate prior.

\[ \xi \]

The inversion may not be always available in closed form, in which case we resort to numerical techniques.
Although EM exposes substantial parallelism, it is difficult to scale, since the dense structure \( p(z_i|x_i; \eta^{(1)}) \) defines values for all possible outcomes for \( z \) and thus puts tremendous pressure on memory bandwidth.

To overcome this we introduce sparsity by employing stochastic EM (SEM) Celeux and Diebolt, 1985. SEM substitutes the E-step for an S-step that replaces the full distribution with a single sample:

**s-step** Sample \( z_i^{(1)} \sim p(z_i|x_i; \eta^{(1)}) \) in parallel.

Subsequently, we perform the M-step using the imputed data instead of the expectation. This simple modification overcomes the computational drawbacks of EM for cases in which sampling from \( p(z_i|x_i; \eta^{(1)}) \) is feasible. We can now employ fast samplers, such as the alias method, exploit sparsity, reduce CPU-RAM bandwidth while still maintaining massive parallelism.

The S-step has other important consequences. Notice that the M-step is now a simple function of current sufficient statistics. This implies that the conditional distribution for the next S-step is expressible in terms of the complete sufficient statistics

\[
p(z_i|x_i; \eta^{(1)}) = f(z_i, T(x, z^{(1)})).
\]

Thus each S-step depends only upon the sufficient statistics generated by the previous step. Therefore, we can operate directly on sufficient statistics without the need to assign or store latent variables/states. Moreover it opens up avenues for distributed and parallel implementations that execute on an SCA.

### 7.2.3 ESCA for Latent Variable Models

SEM produces an alternating sequence of S and M steps in which the M step produces the parameters necessary for the next S step. Since we can compute these parameters on the fly there is no need for an explicit M step. Instead, ESCA produces a sequence consisting only of S steps. We require the exponential family to ensure that these steps are both efficient and compact. We now present ESCA more formally.

Define an SCA over the state space \( S \) of the form:

\[
S = Z \rightarrow K \times X
\]

where \( Z \) is the set of cell identifiers (e.g., one per token in a text corpus), \( K \) is the domain of latent variables, and \( X \) is the domain of the observed data.

The initial state \( s_0 \) is the map defined as follows: for every data point, we associate a cell \( z \) to the pair \( (k_z, x) \) where \( k_z \) is chosen at random from \( K \) and independently from \( k_z' \) for all \( z' \neq z \). This gives us the initial state \( s_0 \).

\[
s_0 = z \mapsto (k_z, x)
\]

We now need to describe the evolution function \( \Phi \). First, assuming that we have a state \( s \) and a cell \( z \), we define the following distribution:

\[
p_z(k|s) = f(z, T(s))
\]

Assuming that \( s(z) = (k, x) \) and that \( k' \) is a sample from \( p_z \) (hence the name “stochastic” cellular automaton) we define the local update function as:

\[
\phi(s, z) = (k', x)
\]

where \( s(z) = (k, x) \) and \( k' \sim p_z(\cdot|s) \)
That is, the observed data remain unchanged, but we choose a new latent variable according to the distribution $p_z$ induced by the state. We obtain the evolution function of the stochastic cellular automaton by applying the function $\phi$ uniformly on every cell.

$$\Phi(s) = z \mapsto \phi(s, z)$$  \hfill (7.8)

Finally, the SCA algorithm simulates the evolution function $\Phi$ starting with $s_0$.

As explained earlier, due to our assumption of complete data likelihood belonging to the exponential family, we never have to represent the states explicitly, and instead employ the sufficient statistics.

An implementation can, for example, have two copies of the data structure containing sufficient statistics $T^{(0)}$ and $T^{(1)}$. We do not compute the values $T(z, x)$ but keep track of the sum as we impute values to the cells/latent variables. During iteration $2t$ of the evolution function, we apply $\Phi$ by reading from $T^{(0)}$ and incrementing $T^{(1)}$ as we sample the latent variables (See Figure 1). Then in the next iteration $2t + 1$ we reverse the roles of the data structures, i.e. read from $T^{(1)}$ and increment $T^{(0)}$. We summarize in Algorithm 2.

**Algorithm 2 ESCA**

1. Randomly initialize each cell
2. for $t = 0 \rightarrow \text{num iterations}$ do
   3. for cell $z$ independently in parallel do do
      4. Read sufficient statistics from $T^{(t \mod 2)}$
      5. Compute stochastic updates using $p_z(k | s)$
      6. Write sufficient statistics to $T^{(t + 1 \mod 2)}$
   7. end for
3. end for

Use of such read/write buffers offer a virtually lock-free (assuming atomic increments) implementation scheme for ESCA and is analogous to double-buffering in computer graphics. Although there is a synchronization barrier after each round, its effect is mitigated because each cell’s work depends only upon the sufficient statistics and thus does the same amount of work. Therefore, evenly balancing the work load across computation nodes is trivial, even for a heterogeneous cluster.

Furthermore, in the case of discrete latent variable, updating sufficient statics only requires increments to the data structure $T^{(r)}$ allowing the use of approximate counters Csűrős, 2010; Morris, 1978. Approximate counters greatly reduce memory costs for the counts: e.g., only 4 or 8 bits per counter. Recent empirical evidence demonstrates that approximate counters preserve statistical performance without compromising runtime performance Tristan, Tassarotti, and
Steele Jr., 2015. In fact, speed often increases because not every increment to the counter results in a write to memory. Note, due to the compression, maintaining two buffers requires less memory than one. Finally, if the latent variables are discrete valued then we can leverage the fast Vose’ alias method Michael D Vose, 1991 to sample. The $O(|K|)$ construction cost for the alias method can be easily amortized because the rule is homogeneous and thus alias tables can be shared. Details about alias sampling method is provided in Appendix ??.

7.2.4 Wide Applicability of ESCA

As stated previously, ESCA is technically applicable to any model in which the complete data likelihood is in the exponential family. Designing an ESCA algorithm for a model of interest requires simply deriving the S-step for the local update function in the automaton. The S-step is the full conditional (Equation 7.6) which is easy to derive for many models; for example, mixture models in which (1) the data and parameter components are conjugate and (2) the latent variables and priors are conjugate. We list a few examples of such models in Table 7.1 and provide additional details in Appendix ??.

ESCA is also applicable to models such as restricted Boltzmann machines (RBMs). For example, if the data were a collection of images, each cell could independently compute the S-step for its respective image. For RBMs the cell would flip a biased coin for each latent variable, and for deep Boltzmann machines, the cells could perform Gibbs sampling. We save a precise derivation and empirical evaluation for future work.

7.2.5 Understanding the limitations of ESCA

While ESCA has tremendous potential as a computational model for machine learning, in some cases, using it to obtain MAP estimates is not clear.

Consider an Ising model on an $d$-dimensional torus $H$

$$p(x) \propto \prod_{(i,j) \in H} \exp(w_{ij}x_i x_j) \tag{7.9}$$

in which $x_i$ takes on values in $\{-1, 1\}$. The equilibrium distribution of SCA with a Gibbs update is then Neumann and Derrida, 1988

$$q(x) \propto \prod_{(i,j) \in H} \cosh(w_{ij}x_i x_j). \tag{7.10}$$

Note that the hyperbolic cosine function ($\cosh$) is symmetric in the sense that $\cosh(r) = \cosh(-r)$. For values $r \geq 0$ $\cosh$ is a good approximation and has a maximum that corresponds to the exponential function; however, for values $r < 0$, the $\cosh$ is a poor approximation for the exponential function.

Let $x_1, x_2$ be two random variables taking on values in $\{-1, 1\}$. We define a simple two-variable Ising model on a trivial one-dimensional torus:

$$p(x_1, x_2) \propto \exp(x_1 x_2) \tag{7.11}$$

We can enumerate and quantify the state space under both SCA $q(x_1, x_2)$ and the true distribution $p(x_1, x_2)$:
<table>
<thead>
<tr>
<th>state</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( q(x_1, x_2) \propto )</th>
<th>( p(x_1, x_2) \propto )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>( \cosh(1) )</td>
<td>( \exp(1) )</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>( \cosh(-1) )</td>
<td>( \exp(-1) )</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>( \cosh(-1) )</td>
<td>( \exp(-1) )</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>( \cosh(1) )</td>
<td>( \exp(1) )</td>
</tr>
</tbody>
</table>

Since \( \cosh \) is symmetric, all states are equally probable for SCA and states 1 and 2 are MAP states. Yet, under the true distribution, they are not. Consequently, SCA with a Gibbs rule for the local evolution function can yield incorrect MAP estimates.

Fortunately, in most cases we are interested in a model over a dataset in which the data is i.i.d. That is, we can fix our example as follows. Rather than parallelizing a single Ising model at the granularity of pixels (over a single torus or grid), we instead parallelize the Ising model at the granularity of the data (over multiple tori, one for each image). Then, we could employ Gibbs sampling on each image for the S-step.

### 7.2.6 Convergence

We now address the critical question of how the invariant measure of ESCA for the model presented in Section 2.1 is related to the true MAP estimates. First, note that SCA is ergodic Louis, 2002, a result that immediately applies if we ignore the deterministic components of our automata (corresponding to the observations). Now that we have established ergodicity, we next study the properties of the stationary distribution and find that the modes correspond to MAP estimates.

We make a few mild assumptions about the model:

- The observed data Fisher information is non-singular, i.e. \( I(\theta) > 0 \).
- For the Fisher information for \( z|x \), we need it to be non-singular and central limit theorem, law of large number to hold, i.e. \( \mathbb{E}_{\theta_0}[I_X(\theta_0)] > 0 \) and
  \[
  \sup_n \left| \frac{1}{n} \sum_{i=1}^{n} I_{x_i}(\theta) - \mathbb{E}_{\theta_0}[I_X(\theta)] \right| \to 0 \text{ as } n \to \infty
  \]
- We assume that \( \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log p(x_i; \theta) = 0 \) has at least one solution, let \( \hat{\theta} \) be a solution.

These assumptions are reasonable. For example in case of mixture models (or topic models), it just means all component must be exhibited at least once and all components are unique. The details of this case are worked out in Appendix ???. Also when the number of parameters grow with the data, e.g., for topic models, the second assumption still holds. In this case, we resort to corresponding result from high dimensional statistics by replacing the law of large numbers with Donsker’s theorem and everything else falls into place.

Consequently, we show ESCA converges weakly to a distribution with mean equal to some root of the score function \( \nabla_\theta \log p(x_i; \theta) \) and thus a MAP fixed point by borrowing the results known for SEM Nielsen, 2000b. In particular, we have:

**Theorem 7.1** Let the assumptions stated above hold and \( \hat{\theta} \) be the estimate from ESCA. Then as the number of i.i.d. data point goes to infinity, i.e. \( n \to \infty \), we have

\[
\sqrt{n}(\hat{\theta} - \theta) \overset{D}{\to} N\left(0, I(\theta_0)^{-1} \left[ I - F(\theta_0)^{-1} \right] \right)
\]

where \( F(\theta_0) = I + \mathbb{E}_{\theta_0}[I_X(\theta_0)](I(\theta_0) + \mathbb{E}_{\theta_0}[I_X(\theta_0)]) \).
This result implies that SEM flocks around a stationary point under very reasonable assumptions and tremendous computational benefits. Also, for such complicated models, reaching a stationary point is the best that most methods achieve anyway. Now we switch gears to adopt ESCA for LDA and perform some simple experimental evaluations.

### 7.3 ESCA FOR LDA

Topic modeling, and latent Dirichlet allocation (LDA) D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003b in particular, have become a must-have of analytics platforms and consequently needs to scale to larger and larger datasets. In LDA, we model each document \( m \) of a corpus of \( M \) documents as a distribution \( \theta_m \) that represents a mixture of topics. There are \( K \) such topics, and we model each topic \( k \) as a distribution \( \phi_k \) over the vocabulary of words that appear in our corpus. Each document \( m \) contains \( N_m \) words \( w_{mn} \) from a vocabulary of size \( V \), and we associate a latent variable \( z_{mn} \) to each of the words. The latent variables can take one of \( K \) values indicating the topic for the word. Both distributions \( \theta_m \) and \( \phi_k \) have a Dirichlet prior, parameterized respectively with a constant \( \alpha \) and \( \beta \). See Appendix ?? for more details.

#### 7.3.1 Existing systems

Many of the scalable systems for topic modeling are based on one of two core inference methods: the collapsed Gibbs sampler (CGS) T. Griffiths and Steyvers, 2004, and variational inference (VI) D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003b and approximations thereof Asuncion et al., 2009. To scale LDA to large datasets, or for efficiency reasons, we may need to distribute and parallelize them. Both algorithms can be further approximated to meet such implementation requirements.

**Collapsed Gibbs Sampling** In collapsed Gibbs sampling the full conditional distribution of the latent topic indicators given all the others is

\[
p(z_{mn} = k | z^{-mn}, w) \propto \frac{D_{mk} + \alpha}{T_k + \beta V} \frac{W_{kvw} + \beta}{T_k + \beta V}
\]

where \( D_{mk} \) is the number of latent variables in document \( m \) that equal \( k \), \( W_{kv} \) is the number of latent variables equal to \( k \) and whose corresponding word equals \( v \), and \( T_k \) is the number of latent variables that equal \( k \), all excluding current \( z_{mn} \).

CGS is a sequential algorithm in which we draw latent variables in turn, and repeat the process for several iterations. The algorithm performs well statistically, and has further benefited from breakthroughs that lead to a reduction of the sampling complexity Aaron Q. Li et al., 2014b; L. Yao, Mimno, and McCallum, 2009b. This algorithm can be approximated to enable distribution and parallelism, primarily in two ways. One is to partition the data, perform one sampling pass and then assimilate the sampler states, thus yielding an approximate distributed version of CGS (AD-LDA) Newman, Asuncion, et al., 2009. Another way is to partition the data and allow each sampler to communicate with a distributed central storage continuously. Here, each sampler sends the differential to the global state-keeper and receives from it the latest global value. A scalable system built on this principle and leveraging inherent sparsity of LDA is YahooLDA Smola and Narayanamurthy, 2010. Further improvement and sampling using alias table was incorporated in lightLDA Yuan et al., 2015. Contemporaneously, a nomadic distribution scheme and sampling using Fenwick tree was proposed in F+LDA Yu et al., 2015.

**Variational Inference** In variational inference (VI), we seek to optimize the parameters of an approximate distribution that assumes independence of the latent variables to find a
member of the family that is close to the true posterior. Typically, for LDA, document-topic proportions and topic indicators are latent variables and topics are parameter. Then, coordinate ascent alternates between them.

One way to scale VI is stochastic variational inference (SVI) which employs SGD by repeatedly updating the topics via randomly chosen document subsets M. D. Hoffman et al., 2013. Adding a Gibbs step to SVI introduces sparsity for additional efficiency Mimno, M. Hoffman, and David Blei, 2012. In some ways this is analogous to our S-step, but in the context of variational inference, the conditional is much more expensive to compute, requiring several rounds of sampling.

Another approach, CVB0, achieves scalability by approximating the collapsed posterior W. Y. Teh, Newman, and Welling, 2007. Here, they minimize the free energy of the approximate distribution for a given parameter $\gamma_{mnk}$ and then use the zero-order Taylor expansion Asuncion et al., 2009.

$$\gamma_{mnk} \propto \left( D_{mk} + \alpha \right) \times \frac{W_{kwmn} + \beta}{T_k + \beta V} \quad (7.14)$$

where $D_{mk}$ is the fractional contribution of latent variables in document $m$ for topic $k$, $W_{kv}$ is the contribution of latent variables for topic $k$ and whose corresponding word equals $v$, and $T_k$ is the contribution of latent variables for topic $k$. Inference updates the variational parameters until convergence. It is possible to distribute and parallelize CVBo over tokens Asuncion et al., 2009. VI and CVBo are the core algorithms behind several scalable topic modeling systems including Mr.LDA Zhai, J. Boyd-Graber, et al., 2012 and the Apache Spark machine-learning suite.

**Remark** It is worth noticing that Gibbs sampling and variational inference, despite being justified very differently, have at their core the very same formulas (shown in a box in formula (7.13) and (7.14)). Each of which are literally deciding how important is some topic $k$ to the word $v$ appearing in document $m$ by asking the questions: “How many times does topic $k$ occur in document $m$?”, “How many times is word $v$ associated with topic $k$?”, and “How prominent is topic $k$ overall?”. It is reassuring that behind all the beautiful mathematics, something simple and intuitive is happening. As we see next, ESCA addresses the same questions via analogous formulas for SEM.

### 7.3.2 An ESCA Algorithm for LDA

To re-iterate, the point of using such a method for LDA is that the parallel update dynamics of the ESCA gives us an algorithm that is simple to parallelize, distribute and scale. In the next section, we will evaluate how it works in practice. For now, let us explain how we design our SCA to analyze data.

We begin by writing the stochastic EM steps for LDA (derivation is in Appendix ??):

**e-step:** independently in parallel compute the conditional distribution locally:

$$q_{mnk} = \frac{\theta_{mk} \phi_{kwmn}}{\sum_{k'} \theta_{mk'} \phi_{k'wmn}} \quad (7.15)$$

**s-step:** independently in parallel draw $z_{ij}$ from the categorical distribution:

$$z_{mn} \sim \text{Categorical}(q_{mn1}, ..., q_{mnK}) \quad (7.16)$$
We now need to describe the evolution function \( \Phi \). First, assuming that we have a state \( s \) and a cell \( z \), we define the following distribution:

\[
p_z(k|s) \propto (D_{mk} + \alpha) \times \frac{W_{kv} + \beta}{\Gamma_k + \beta V}
\]

where \( D_{mk} = \left\{ z \mid \exists v. s(z) = (k, m, v) \right\} \), \( W_{kv} = \left\{ z \mid \exists m. s(z) = (k, m, v) \right\} \), and \( T_k = \left\{ z \mid \exists v. s(z) = (k, m, v) \right\} \). Note that we have chosen our local update rule slightly different without an offset of \(-1\) for the counts corresponding to the mode of the Dirichlet distributions and requiring \( \alpha, \beta > 0 \). Instead, our local update rule allows us to have the relaxed requirement \( \alpha, \beta > 0 \) which is more common for LDA inference algorithms.

Assuming that \( s(z) = (k, m, v) \) and that \( k' \) is a sample from \( p_z \) (hence the name “stochastic” cellular automaton) we define the local update function as:

\[
\phi(s, z) = (k', m, v)
\]

where \( s(z) = (k, m, v) \) and \( k' \sim p_z(\cdot|s) \)

That is, the document and word of the cell remain unchanged, but we choose a new topic according to the distribution \( p_z \) induced by the state. We obtain the evolution function of the stochastic cellular automaton by applying the function \( \phi \) uniformly on every cell.

\[
\Phi(s) = z \mapsto \phi(s, z)
\]

Finally, the SCA algorithm simulates the evolution function \( \Phi \) starting with \( s_0 \). Of course, since LDA's complete data likelihood is in the exponential family, we never have to represent the states explicitly, and instead employ the sufficient statistics.

Our implementation has two copies of the count matrices \( D^i, W^i, \) and \( T^i \) for \( i = 0 \) or \( 1 \) (as in CGS or CVBo, we do not compute the values \( D_{ik}, W_{kv}, \) and \( T_k \) but keep track of the counts as we assign topics to the cells/latent variables). During iteration \( i \) of the evolution function, we apply \( \Phi \) by reading \( D^{i \mod 2}, W^{i \mod 2}, \) and \( T^{i \mod 2} \) and incrementing \( D^{i+1 \mod 2}, W^{i+1 \mod 2}, \) and \( T^{i+1 \mod 2} \) as we assign topics.
7.3.3 Advantages of ESCA for LDA

The positive consequences of ESCA as a choice for inference on LDA are many:

- Our memory footprint is minimal since we only store the data and sufficient statistics. In contrast to MCMC methods, we do not store the assignments to latent variables \( z \). In contrast to variational methods, we do not store the variational parameters \( \gamma \). Further, variational methods require \( K \) memory accesses (one for each topic) per word. In contrast, the S-step ensures we only have a single access (for the sampled topic) per word. Such reduced pressure on the memory bandwidth can improve performance significantly for highly parallel applications.

- We can further reduce the memory footprint by compressing the sufficient statistics with approximate counters Csúrös, 2010; Morris, 1978. This is possible because updating the sufficient statistics only requires increments as in Mean-for-Mode Tristan, Tassarotti, and Steele Jr., 2015. In contrast, CGS decrements counts, preventing the use of approximate counters.

- Our implementation is lock-free (in that it does not use locks, but assumes atomic increments) because the double buffering ensures we never read or write to the same data structures. There is less synchronization, which at scale is significant.

- Finally, our algorithm is able to fully benefit from Vose’s alias method Michael D Vose, 1991 because homogeneous update rule for SCA ensures that the cost for constructing the alias table is amortized across the cells. To elaborate, the SCA update Equation (7.20) decomposes as

\[
p_z(k|s) \propto \left[ D_{mk} \frac{W_{kv} + \beta}{T_k + \beta V} \right] + \left[ \alpha \frac{W_{kv} + \beta}{T_k + \beta V} \right] \tag{7.23}
\]

allowing us to treat it as a discrete mixture and divide the sampling procedure into two steps. First, we toss a biased coin to decide which term of the equation to sample, and second, we employ a specialized sampler depending on the chosen term. The first term is extremely sparse (documents comprise only a small handful of topics) and a basic sampling procedure suffices. The second term is not sparse, but is independent of the current document \( m \) and depends only on the \( W \) and \( T \) matrices. Moreover, as mentioned earlier, during iteration \( i \), we will be only reading values from non-changing \( W^i \mod 2 \), and \( T^i \mod 2 \) matrices. As a result, at the start of each iteration we can precompute, from the \( W \) and \( T \) matrices, tables for use with Vose’s alias method, which enables sampling from the second term in a mere 3 CPU operations. Thus, the evolution for ESCA is extremely efficient.

7.3.3.1 Connection to SGD

We can view ESCA as implicit SGD on MAP for LDA. This connection alludes to the convergence rate of ESCA. To illustrate, we consider \( \theta \) only. As pointed out in R. Salakhutdinov, Roweis, and Ghahramani, n.d.; L. Xu and Michael I Jordan, 1996, one EM step is:

\[
\theta^+_m = \theta_m + M \frac{\partial \log p}{\partial \theta^+_m}
\]

which is gradient descent with a Frank-Wolfe update and line search. Similarly, for ESCA using stochastic EM, one step is

\[
\theta^+_m = \frac{Dn_{mk}}{N_m} = \frac{1}{N_m} \sum_{n=1}^{N_m} \delta(z_{mn} = k)
\]
Again vectorizing and re-writing as earlier:

\[ \theta_m^+ = \theta_m + Mg \]

where \( M = \frac{1}{N_m} \left[ \text{diag}(\theta_m) - \theta_m \theta_m^T \right] \) and \( g = \frac{1}{\theta_{mk}} \sum_{n=1}^{N_m} \delta(z_{mn} = k) \). The vector \( g \) can be shown to be an unbiased noisy estimate of the gradient, i.e.

\[ \mathbb{E}[g] = \frac{1}{\theta_{mk}} \sum_{n=1}^{N_m} \mathbb{E}[\delta(z_{ij} = k)] = \frac{\partial \log p}{\partial \theta_{mk}}. \]

Thus, a single step of SEM on our SCA is equivalent to a single step of SGD. Consequently, we could further embrace the connection to SGD and use a subset of the data for the S and M steps, similar to incremental EM R. M. Neal and G. E. Hinton, 1998. Note that in the limit in which batches comprise just a single token, the algorithm emulates a collapsed Gibbs sampler. This interpretation strengthens the theoretical justification for many existing approximate Gibbs sampling approaches.

### 7.4 EXPERIMENTS

To evaluate the strength and weaknesses of our algorithm, we compare against parallel and distributed implementations of CGS and CVBo. We also compare our results to performance numbers reported in the literature including those of F+LDA and lightLDA.

**SOFTWARE & HARDWARE** All three algorithms are implemented in simple C++11. We implement multithreaded parallelization within a node using the work-stealing Fork/Join framework, and the distribution across multiple nodes using the process binding to a socket over MPI. We also implemented a version of ESCA with a sparse representation for the array \( D \) of counts of topics per documents and Vose’s alias method to draw from discrete distributions. We run our experiments on a small cluster of 8 Amazon EC2 c4.8xlarge nodes connected through
10Gb/s Ethernet. Each node has a 36 virtual threads per node. For random number generation we employ Intel®Digital Random Number Generators through instruction RDRAND, which uses thermal noise within the silicon to output a random stream of bits at 3 Gbit/s, producing true random numbers.

**Datasets** We experiment on two public datasets, both of which are cleaned by removing stop words and rare words: PubMed abstracts and English Wikipedia. To demonstrate scalability, run on 100 copies of English Wikipedia and a third proprietary dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>V</th>
<th>M</th>
<th>Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>PubMed</td>
<td>141,043</td>
<td>8,200,000</td>
<td>737,869,085</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>210,233</td>
<td>6,631,176</td>
<td>1,133,050,514</td>
</tr>
<tr>
<td>Large</td>
<td>~140,000</td>
<td>~3 billion</td>
<td>~171 billion</td>
</tr>
</tbody>
</table>

**Evaluation** To evaluate the proposed method we use predicting power as a metric by calculating the per-word log-likelihood (equivalent to negative log of perplexity) on 10,000 held-out documents conditioned on the trained model. We set $K = 1000$ to demonstrate performance for a large number of topics. The hyper parameters are set as $\alpha = 50/K$ and $\beta = 0.1$ as suggested in T. Griffiths and Steyvers, 2004; other systems such as YahooLDA and Mallet also use this as the default parameter setting. The results are presented in Figure 7.2 and some more experiments in Appendix ??.

Finally, for the larger datasets, our implementation of ESCA (only 300 lines of C++) processes more than **1 billion tokens per second** (tps) by using a 20-node cluster. In comparison, some of the best existing systems achieve 112 million tps (F+LDA, personal communication) and 60 million tps (lightLDA) Yuan et al., 2015 using more hardware. Detailed Appendix ?? Table ??)

### 7.5 DISCUSSION

We have described a novel inference method for latent variable models that simulates a stochastic cellular automaton. The equilibrium of the dynamics are MAP fixed points and the algorithm has many desirable computational properties: it is embarrassingly parallel, memory efficient, and like HOGWILD! RecReWriNiu11, is virtually lock-free. Further, for many models, it enables the use of approximate counters and the alias method. Thus, we were able to achieve an order of magnitude speed-up over the current state-of-the-art inference algorithms for LDA with accuracy comparable to collapsed Gibbs sampling.

In general, we cannot always guarantee the correct invariant measure Dawson, 1974, and found that parallelizing improperly causes convergence to incorrect MAP fixed points. Even so, SCA is used for simulating Ising models in statistical physics Vichniac, 1984. Interestingly, in previous work Lebowitz, Maes, and Speer, 1990, it has been shown that stochastic cellular automata are closely related to equilibrium statistical models and the stationary distribution is known for a large class of finite stochastic cellular automata.
Reducing Lookups

Hierarchical Bayesian models often capture distributions over a very large number of distinct atoms. The need for these models arises when organizing huge amount of unsupervised data, for instance, features extracted using deep convnets that can be exploited to organize abundant unlabeled images. Inference for hierarchical Bayesian models in such cases can be rather non-trivial, leading to approximate approaches. In this work, we propose Canopy, a sampler based on Cover Trees that is exact, has guaranteed runtime logarithmic in the number of atoms, and is provably polynomial in the inherent dimensionality of the underlying parameter space. In other words, the algorithm is as fast as search over a hierarchical data structure. We provide theory for Canopy and demonstrate its effectiveness on both synthetic and real datasets, consisting of over 100 million images.

8.1 Introduction

Fast nearest-neighbor algorithms have become a mainstay of information retrieval Beygelzimer, Kakade, and Langford, 2006; Indyk and Motwani, 1998; T. Liu, Rosenberg, and Rowley, 2007. Search engines are able to perform virtually instantaneous lookup among sets containing billions of objects. In contrast, inference procedures for latent variable models (Gibbs sampling, EM, or variational methods) are often problematic even when dealing with thousands of distinct objects. This is largely because, for any inference methods, we potentially need to evaluate all probabilities whereas search only needs the best instance.

While the above is admittedly an oversimplification of matters (after all, we can use Markov-Chain Monte Carlo methods for inference), it is nonetheless nontrivial to perform exact sampling for large state spaces. In the current work, we propose Canopy, an inference technique to address this issue by marrying a fast lookup structure with an adaptive rejection sampler. This leads to a surprisingly simple design for a plethora of sampling-based inference algorithms. Moreover, we provide runtime guarantees for Canopy that depend only on the inherent dimensionality of both parameter and data distributions. The expected depth for lookups is never

Figure 8.1: Canopy is much faster yet as accurate as other methods like EM or ESCA Zaheeretal16. The bar graph shows time per iteration while line plots the likelihood on held-out test set. Results shown are for inference of a Gaussian mixture model with 32 million points having 4096 clusters at 1024 dimensions.
worse than logarithmic in the number of atoms and the characteristic length scale at which models can be sufficiently well distinguished. Furthermore, we can parallelize Canopy for hierarchical Bayesian models using stochastic cellular automata (ESCA) \cite{zaheer2016}, thus leading to an extremely scalable and efficient system design.

Most latent variable models, e.g., Gaussian mixture models (GMM), latent Dirichlet allocation \cite{blei2002}, hidden Markov models, Dirichlet process clustering \cite{neal1998}, or hierarchical generative models \cite{adi2010}, have the structure of the form:

\[
p(x) = \sum_z p(z)p(x|\theta_z) \tag{8.1}
\]

where \(x\) denotes observed variables, \(z\) latent variables, and \(\theta_z\) parameters of the conditional. Often the conditional distribution \(p(x|\theta_z)\) belongs to the exponential family, which we assume to be the case as well. The inference procedure on these models using either Gibbs sampling, stochastic variation methods, or ESCA would require to draw \(z \sim p(z|x)\) repeatedly. Naïvely producing these draws would be expensive, especially when the number of latent classes is huge. We aim to bring the per-iteration cost down from \(O(mn)\) to \(O(m + n)\), where \(m, n\) are the number of latent classes and data points, respectively. For example, on GMM, the proposed method Canopy is much faster than EM or ESCA, while achieving the same accuracy as shown in Figure 8.1.

Our approach is as follows: we use cover trees \cite{beygelzimer2006} to design an efficient lookup structure for \(p(x|\theta_z)\) and approximate the values of \(p(x|\theta_z)\) for a large number of \(\theta_z\). In combination with an efficient node summary for \(p(z)\), this allows us to design a rejection sampler that has an increasingly low rejection rate as we descend the tree. Moreover, for large numbers of observations \(x\), we use another cover tree to aggregate points into groups of similar points, perform expensive pre-computation of assignment probabilities \(p(z|x)\) only once, and amortize them over multiple draws. In particular, the alias method \cite{walker1977} allows us to perform sampling in \(O(1)\) time once the probabilities have been computed.

In summary, Canopy has three parts: construction of cover trees for both parameters and data (Sec. 8.3.1, 8.3.2), an adaptive rejection sampler at the top-level of the cover tree until the data representation is sufficiently high to exploit it for sampling (Section 8.3.2.1), and a rejection sampler in the leaves (Section 8.3.2.2), whenever the number of clusters is large. Most importantly, the algorithm becomes more efficient as we obtain larger amounts of data since they lead to greater utilization of the alias table in \cite{walker1977} as shown by theoretical analysis in Section 8.4. This makes it particularly well-suited to big data problems as demonstrated through experiments in Section 8.5.

## 8.2 Background

We briefly discuss latent variable models, cover trees, and the alias method needed to explain this work.

### 8.2.1 Latent Variable Models

The key motivation for this work is to make inference in latent variable models more efficient. As expressed in (8.1), we consider latent models which have mixtures of exponential family. The reasons for limiting to exponential families are two fold. First, most of the mixture models used in practice belong to this class. Second, assumptions on model structure, for instance exponential family, allows for efficient design of fast inference. In particular, we first assume
that updates to \( p(z) \) can be carried out by modifying \( O(1) \) values at any given time. For instance, for Dirichlet process mixtures, the collapsed sampler uses 
\[
p(z_{i} = j \mid Z \setminus (z_{i})) = n_{i}^{-1} / (n + \alpha - 1).
\]

Here, \( n \) is the total number of observations, \( n_{j}^{-1} \) denotes the number of occurrences of \( z_{i} = j \) when ignoring \( z_{i} \), and \( \alpha \) is the concentration parameter. Second, the conditional \( p(x \mid \theta) \) in (8.1) is assumed to be a member of the exponential family, i.e.,
\[
p(x \mid \theta) = \exp((\phi(x), \theta) - g(\theta)). \tag{8.2}
\]

Here \( \phi(x) \) represents the sufficient statistics and \( g(\theta_{z}) \) is the (normalizing) log-partition function.

Trying to find a metric data structure for fast retrieval is not necessarily trivial for the exponential family. JiaKuJor12 and Caytono8 design Bregman divergence based methods for this problem. Unfortunately, such methods are costlier to maintain and have less efficient lookup properties than those using Euclidean distance, as computing and optimizing over Bregman divergences is less straightforward. For example, whenever we end up on the boundary of the marginal polytope, as is common with natural parameters associated with single observations, optimization becomes intractable. Fortunately, this problem can be avoided entirely by rewriting the exponential family model as
\[
p(x \mid \theta) = e^{((\phi(x), -1), \theta, g(\theta))} = e^{(\phi(x), \tilde{\theta})}
\]
where \( \tilde{\phi}(x) := (\phi(x), -1) \) and \( \tilde{\theta} := (\theta, g(\theta)) \).

In this case, being able to group similar \( \tilde{\theta} \) together allows us to assess their contributions efficiently without having to inspect individual terms. Finally, we assume that \( \|\tilde{\phi}(x_{i})\| \leq R \) and \( \|\tilde{\theta}_{z}\| \leq T \) for all \( i \) and for all \( z \in \mathbb{Z} \) respectively.

### 8.2.2 Alias Sampler

A key component of Canopy is the alias sampler Walker77; Michael D Vose, 1991. Given an arbitrary discrete probability distribution on \( n \) outcomes, it allows for \( O(1) \) sampling once an \( O(n) \) preprocessing step has been performed. Hence, drawing \( n \) observations from a distribution over \( n \) outcomes costs an amortized \( O(1) \) per sample. Section ?? in appendix has more details.

### 8.2.3 Cover Trees

Cover Trees Beygelzimer, Kakade, and Langford, 2006 and their improved version IzbShe15 are a hierarchical data structure that allow fast retrieval in logarithmic time. The key properties are:
- \( O(n \log n) \) construction time,
- \( O(\log n) \) retrieval, and
- polynomial dependence on the expansion constant Karger and Ruhl, 2002 of the underlying space, which we refer to as \( c \).

Moreover, the degree of all internal nodes is well controlled, thus giving guarantees for retrieval (as exploited by (Beygelzimer, Kakade, and Langford, 2006)), and for sampling (as we will be using in this paper).

Cover trees are defined as an infinite succession of levels \( S_{i} \) with \( i \in \mathbb{Z} \). Each level \( i \) contains (a nested subset of) the data with the following properties:

- **Nesting property:** \( S_{i} \subseteq S_{i-1} \).
- All \( x, x' \in S_{i} \) satisfy \( \|x - x'\| \geq 2^{i} \).
- All \( x \in S_{i} \) have children \( x' \in S_{i-1} \), possibly with \( x = x' \), with \( \|x - x'\| \leq 2^{i} \).
- As a consequence, the subtree for any \( x \in S_{i} \) has distance at most \( 2^{i+1} \) from \( x \).
8.3 OUR APPROACH

Now we introduce notation and explain details of our approach when the number of clusters is (a) moderate (Section 8.3.1) and (b) large (Section 8.3.2). In what follows, the number of data points and clusters are denoted with \( n \) and \( m \) respectively. The function \( \text{ch}(x) \) returns children of a node \( x \) of any tree.

**Data tree** \((T_D)\): Cover tree built with levels \( S_j \) on all available data using the sufficient statistic \( \phi(x) \), constructed once for our setup. We record ancestors at level \( j \) as prototypes \( \bar{x} \) for each data point \( x \). In fact, we only need to construct the tree up to a fixed degree of accuracy \( \bar{j} \) in case of moderate number of clusters. A key observation is that multiple points can have the same prototype \( \bar{x} \), making it a many-to-one map. This helps us amortize costs over points by re-using proposal computed with \( \bar{x} \) (Section 8.3.1).

**Cluster tree** \((T_C)\): Similarly, \( T_C \) is the cover tree generated with cluster parameters \( \tilde{\theta}_z \). For simplicity, we assume that the expansion rates of clusters and data are both \( c \).

### 8.3.1 Canopy I: Moderate number of clusters

We introduce our sampler, Canopy I, when the number of clusters is relatively small compared to the total number of observations. This addresses many cases where we want to obtain a flat clustering on large datasets. For instance, it is conceivable that one might not want to infer more than a thousand clusters for one million observations. In a nutshell, our approach works as follows:

1. Construct \( T_D \) and pick a level \( j \in \mathbb{Z} \) with accuracy \( 2^j \) such that the average number of elements per node in \( S_{\bar{j}} \) is \( O(m) \).

2. For each of the prototypes \( \bar{x} \), which are members of \( S_{\bar{j}} \), compute \( p(z|\bar{x}) \) using the alias method to draw from \( m \) components \( \theta_z \). By construction, this cost amortizes \( O(1) \) per observation, \( i.e., \) a total cost of \( O(n) \).

3. For each observation \( x \) with prototype \( \bar{x} \), perform Metropolis-Hastings (MH) sampling using the draws from \( p(z|\bar{x}) =: q(z) \) as proposal. Hence we accept an MH move from \( z \) to \( z' \) with probability

\[
\pi := \min \left( 1, \frac{p(z'|x)p(z|\bar{x})}{p(z|x)p(z'|\bar{x})} \right). \quad (8.4)
\]

The key reason why this algorithm has a useful acceptance probability is that the normalizations for \( p(z|x) \) and \( p(z|\bar{x}) \), and mixing proportions \( p(z) \) and \( p(z') \) cancel out respectively. Only terms remaining in (8.4) are

\[
\pi = \min \left( 1, \exp \left( \langle \phi(x) - \phi(\bar{x}), \tilde{\theta}_{z'} - \tilde{\theta}_{z} \rangle \right) \right) \geq e^{-2^{j+1}L}
\]

for \( \|\tilde{\theta}_z\| \leq L \). This follows from the Cauchy Schwartz inequality and the covering property of cover trees, which ensures all descendants of \( \bar{x} \) are no more than \( 2^{j+1} \) apart from \( \bar{x} \), \( i.e., \) \( \|\phi(x) - \phi(\bar{x})\| \leq 2^{j+1} \).
8.3.2 Canopy II: Large number of clusters

The key difficulty in dealing with many clusters is that it forces us to truncate $T_D$ at a granularity in $x$ that is less precise than desirable in order to benefit from the alias sampler naively. In other words, for a given sampling complexity, a larger $m$ reduces the affordable granularity in $x$. The problem arises because we are trying to distinguish clusters at a level of resolution that is too coarse. A solution is to apply cover trees not only to observations but also to the clusters themselves, i.e., use both $T_D$ and $T_C$. This allows us to decrease the minimum observation-group size at the expense of having to deal with an aggregate of possible clusters.

Our method for large number of clusters operates in two phases: (a) Descend the hierarchy in cover trees while sampling (Section 8.3.2.1) (b) Sample for a single observation $x$ from a subset of clusters arranged in $T_C$ (Section 8.3.2.2), when appropriate conditions are met in (a). We begin with initialization and then elaborate each of these phases in detail.

**initialize 1**: Construct $T_C$ and for each node $\theta_\bar{z}$, assign $\alpha(i, z) = p(z)$, where $i$ is the highest level $S_i$ such that $z \in S_i$, else 0. Then perform bottom-up aggregation via

$$
\beta(i, z) = \alpha(i, z) + \sum_{z' \in ch(z)} \beta(i + 1, z')
$$

This creates at most $O(m)$ distinct entries $\beta(i, z)$. Notice that aggregated value $\beta(i, z)$ captures the mixing probability of the node and its children in $T_C$.

**initialize 2**: Partition both the observations and the clusters at a resolution that allows for efficient sampling and precomputation. More specifically, we choose accuracy levels $\hat{i}$ and $\hat{j}$ to truncate $T_D$ and $T_C$, so that there are $n'$ and $m'$ nodes respectively after truncation. These serve as partitions for data points and clusters such that $n' \cdot m' = O(m)$ is satisfied. The aggregate approximation error

$$
\delta := 2^{i+1}L + 2^{i+1}R + 2^{i+j+2}
$$

due to quantizing observations and clusters is minimized over the split, searching over the levels.

8.3.2.1 Descending $T_D$ and $T_C$

Given $T_D$ and $T_C$ with accuracy levels $\hat{j}$ and $\hat{i}$, we now iterate over the generated hierarchy, as shown in Figure 8.2. We recursively descend simultaneously in both the trees until the number of observations for a given cluster is too small. In that case, we simply default to the sampling algorithm described in Section 8.3.2.2 for each observation in a given cluster.

The reasoning works as follows: Once we have the partitioning into levels $\hat{j}, \hat{i}$ for data and clusters respectively with $n' \cdot m' = O(m)$, we draw from the proposal distribution

$$
q(\bar{z}|x) \propto \beta(i, z) \exp((\phi(\bar{x}), \theta_\bar{z}) - g(\theta_\bar{z}))
$$

for all the observations and clusters above the partitioned levels $\hat{j}$ and $\hat{i}$, respectively. That is, we draw from a distribution where both observations and clusters are grouped. We draw from the proposal for each $x$ in $T_D$ truncated at level $\hat{j}$. Here, $\beta(i, \bar{z})$ collects the prior cluster likelihood from $\bar{z}$ and all its children. As described earlier, we can use the alias method for sampling efficiently from (8.7).

Within each group of observations, drawing from (8.7) leads to a distribution over a (possibly smaller) subset of cluster groups. Whenever the number of observations per cluster group is
Figure 8.2: Hierarchical partitioning over both data observations and clusters. Once we sample clusters at a coarser level, we descend the hierarchy and sample at a finer level, until we have few number of points per cluster. We then use Section 8.3.2.1 for rejection sampler.

small, we default to the algorithm described in Section 8.3.2.2 for each observation. On the other hand, if we have a sizable number of observations for a given cluster, which should happen whenever the clusters are highly discriminative for observations (a desirable property for a good statistical model), we repeat the strategy on the subset to reduce the aggregate approximation error (8.6). In other words, we descend the hierarchy to yield a new pair \( (i', j') \) on the subset of clusters/observations with \( i' < i \) and \( j' < j \) and repeat the procedure.

The process works in a depth-first fashion in order to avoid using up too much memory. The sampling probabilities according to (8.7) are multiplied out for the path over the various hierarchy levels and used in a MH procedure. Each level of the hierarchy can be processed in \( O(1) \) operations per instance, without access to the instance itself. Moreover, we are guaranteed to descend by at least one step in the hierarchy of observations and clusters, hence the cost is at most \( O(c^2 \min(\log n, \log m)) \).

To summarize, we employ a MH scheme as before, with the aim of using a highly accurate, yet cheap proposal. To overcome the loss in precision of Canopy I proposal due to large of clusters, we devise a proposal wherein we look at aggregates of both data and clusters at comparable granularity using both \( T_D \) and \( T_C \). Note that the acceptance probabilities are always at least as high as the bounds derived in Section 8.3.1 as the errors on the paths are log-additive. Instead of MH, a rejection sampler can also be devised. Details are omitted for the sake of brevity, since they mirror the single-observation argument of the following section.

8.3.2.2 Sampling for a single observation \( x \)

Let \( x \) be the single observation for which we want to sample from possibly subset of clusters \( z \) that are arranged in \( T_C \). In this case, we hierarchically descend \( T_C \) using each aggregate as a proposal for the clusters below. As before, we can either use MH sampling or a rejection sampler. To illustrate the effects of the latter, we describe one below, whose theoretical analysis is provided in Section 8.4. Before we delve into details, let us consider a simple case without \( T_C \). If we are able to approximate \( p(x|\theta_z) \) by some \( q_z \) such that

\[
e^{-c} p(x|\theta_z) \leq q_z \leq e^c p(x|\theta_z)
\]  

(8.8)
for all \( z \), then it follows that a sampler drawing \( z \) from
\[
z = \frac{q_z p(z)}{\sum_{z'} q_{z'} p(z')}
\]
and then accepting with probability \( e^{-\epsilon} q_z^{-1} p(x | \theta_z) \) will draw from \( p(z | x) \) (see Appendix Section ?? for details). Moreover, the acceptance probability is at least \( e^{-2\epsilon} \). However, finding such \( q_z \) with a small \( \epsilon \) is not easy in general. Thus, we propose to cleverly utilize structure of the cover tree \( T_C \) to begin with a very coarse approximation and successively improving the approximation only for a subset of \( \theta_z \) which are of interest. The resultant sampler is described below:

1. Choose approximation level \( \hat{i} \) and compute normalization at accuracy level \( \hat{i} \):
\[
\gamma_0 := \sum_{z \in S_{\hat{i}}} \beta(i, z) \exp \langle \hat{\theta}_z, \hat{\phi}(x) \rangle.
\]
2. Set \( e^{-\epsilon} := e^{-2\hat{i} \| \hat{\phi}(x) \|} \) as multiplier for the acceptance threshold of the sampler and
\( \gamma := e^{\epsilon} \gamma_0 \).
3. Draw a node \( z \in S_{\hat{i}} \) with probability \( \delta_z := \gamma^{-1} e^{\epsilon} \beta(i, z) \exp \langle \hat{\theta}_z, \hat{\phi}(x) \rangle \).
4. Accept \( z_i \) at the current level with probability \( \pi := \gamma^{-1} \delta_z^{-1} p(z_i) \exp \langle \hat{\theta}_{z_i}, \hat{\phi}(x) \rangle \).
5. For \( \hat{i} := \hat{i} - 1 \) down to \(-\infty\) do
   a) Set \( e^{-\epsilon} := e^{-2\| \hat{\phi}(x) \|} \) as the new multiplier and \( \gamma := \delta_{z_{\hat{i}+1}} (1 - \pi) \) as the new normalizer.
   b) Draw one of the children \( z \) of \( z_{\hat{i}+1} \) with probability \( \delta_z := \gamma^{-1} e^{\epsilon} \beta(i, z) \exp \langle \hat{\theta}_z, \hat{\phi}(x) \rangle \).
      Exit if we do not draw any of them (since \( \sum_{z \in \text{ch}(z_{\hat{i}+1})} \delta_z \leq 1 \)) and restart from step \( z \), else denote this child by \( z_i \).
   c) Accept \( z_i \) at the current level with probability \( \pi := \gamma^{-1} \delta_z^{-1} p(z_i) \exp \langle \hat{\theta}_{z_i}, \hat{\phi}(x) \rangle \). Do not include \( z_{\hat{i}+1} \) in this setting, as we consider \( z \) only the first time we encounter it.

The above describes a rejection sampler that keeps on upper-bounding the probability of accepting a particular cluster or any of its children. It is as aggressive as possible at retaining tight lower bounds on the acceptance probability such that not too much effort is wasted in traversing the cover tree to the bottom, i.e., we attempt to reject as quickly as possible.

### 8.4 Theoretical Analysis

The main concern is to derive a useful bound regarding the runtime required for drawing a sample. Secondary concerns are those of generating the data structure. We address each of these components, reporting all costs per data point.

**Construction** The data structure \( T_D \) costs \( O(c^6 \log n) \) (per data-point) to construct and \( T_C \) costs \( O(c^6 \log m) \) (per data-point, as \( m < n \)) — all additional annotations cost negligible time and space. This includes computing \( \alpha \) and \( \beta \), as discussed above.

**Startup** The first step is to draw from \( S_{\hat{i}} \). This costs \( O(|S_{\hat{i}}|) \) for the first time to compute all probabilities and to construct an alias table. Subsequent samples only cost \( \gamma \) CPU cycles to draw from the associated alias table. The acceptance probability at this step is \( e^{-2\epsilon} \). Hence the aggregate cost for the top level is bounded by \( O \left( |S_{\hat{i}}| + e^{2\hat{i} \| \hat{\phi}(x) \|} \right) \).
To terminate the sampler successfully, we need to traverse $T_C$ at least once to its leaf in the worst case. This costs $O(c^2 \log m)$ if the leaf is at maximum depth.

The main effort of the analysis is to obtain useful guarantees for the amount of effort wasted in drawing from the cover tree. A brute-force bound immediately would yield $O\left(e^{2\epsilon+2}\|\Phi(x)\|e^6\log m\right)$. Here the first term is due to the upper bound on the acceptance probability, a term of $c^4$ arises from the maximum number of children per node and lastly the $c^2 \log m$ term quantifies the maximum depth. It is quite clear that this term would dominate all others. We now derive a more refined (and tighter) bound.

Essentially we will exploit the fact that the deeper we descend into the tree, the less likely we will have wasted computation later in the process. We use the following relations

$$e^x - 1 \leq xe^a \text{ for } x \in [0, a] \text{ and } \sum_{i=1}^{\infty} 2^{-i} = 1.$$  \hspace{1cm} (8.11)

In expectation, the first step of the sampler requires $e^{2\epsilon} = e^{2\epsilon+2}\|\Phi(x)\|$ steps in expectation until a sample is accepted. Thus, $e^{2\epsilon} - 1$ effort is wasted. At the next level below we waste at most $e^{2\epsilon+1}\|\Phi(x)\| - 1$ effort. Note that we are less likely to visit this level commensurate with the acceptance probability. These bounds are conservative since any time we terminate above the very leaf levels of the tree we are done. Moreover, not all vertices have children at all levels, and we only need to revisit them whenever they do. In summary, the wasted effort can be bounded from above by

$$c^4 \sum_{i=1}^{\infty} \left(e^{2\epsilon-1}\|\Phi(x)\| - 1\right) \leq c^4 e^{2\epsilon}\|\Phi(x)\| \sum_{i=1}^{\infty} 2^{-i} = c^4 e^{2\epsilon}\|\Phi(x)\|.$$

Here $c^4$ was a consequence of the upper bound on the number of children of a vertex. Moreover, note that the exponential upper bound is rather crude, since the inequality (8.11) is very loose for large $a$. Nonetheless we see that the rejection sampler over the tree has computational 

**overhead independent of the tree size!** This result is less surprising than it may seem. Effectively we pay for lookup plus a modicum for the inherent top-level geometry of the set of parameters.

**Theorem 8.1** *The cover tree sampler incurs worst-case computational complexity per sample of

$$O\left(|S_t| + c^6 \log n + c^6 \log m + c^4 e^{2\epsilon+2}\|\Phi(x)\| \right)$$

(8.12)*

Note that the only data-dependent terms are $c, S_t$, and $\|\Phi(x)\|$ and that nowhere the particular structure of $p(z)$ entered the analysis. This means that our method will work equally well regardless of the type of latent variable model we apply. For example, we can even apply the model to more complicated latent variable models like latent Dirichlet allocation (LDA). The aforementioned constants are all natural quantities inherent to the problems we analyze. The constant $c$ quantifies the inherent dimensionality of the parameter space, $\|\Phi(x)\|$ measures the dynamic range of the distribution, and $S_t$ measure the “packing number” of the parameter space at a minimum level of granularity.

### 8.5 Experiments

We now present empirical studies for our fast sampling techniques in order to establish that (i) Canopy is fast (Section 8.5.1), (ii) Canopy is accurate (Section 8.5.2), and (iii) it opens new
avenses for data exploration and unsupervised learning (Section 8.5.3), previously unthinkable. To illustrate these claims, we evaluate on finite mixture models, more specifically, Gaussian Mixture models (GMM), a widely used probabilistic models. However, the proposed method can be applied effortlessly to any latent variable model like topic modeling through Gaussian latent Dirichlet allocation (Gaussian LDA) DasZahDye15. We pick GMMs due to their widespread application in various fields spanning computer vision, natural language processing, neurobiology, etc.

Methods For each experiment, we compare our two samplers (Canopy I, Section 8.3.1 and Canopy II, Section 8.3.2) with both the traditional Expectation Maximization (EM) Dempster, Laird, and Rubin, 1977 and the faster Stochastic EM through ESCA (ESCA) Zaheeretal16 using execution time, cluster purity, and likelihood on a held out TEST set.

Software & hardware All the algorithms are implemented multithreaded in simple C++ using a distributed setup. Within a node, parallelization is implemented using the work-stealing Fork/Join framework, and the distribution across multiple nodes using the process binding to a socket over MPI. We run our experiments on a cluster of 16 Amazon EC2 c4.8xlarge nodes connected through 10Gb/s Ethernet. There are 36 virtual threads per node and 60GB of memory. For purpose of experiments, all data and calculations are carried out at double floating-point precision.

Initialization Recall that speed and quality of inference algorithms depend on initialization of the random variables and parameters. Random initializations often lead to poor results, and so many specific initialization schemes have been proposed, like KMeans++ ArtVaso7, K-MC2 BacLucHasKra16. However, these initializations can be costly, roughly O(mn).

Our approach provides a good initialization using cover trees free of cost, as the construction of cover tree is at the heart of our sampling approach. The proposed initialization scheme relies on the observation that cover trees partition the space of points while preserving important invariants based on its structure. They thus help in selecting initializations that span the entirety of space occupied by the points, which is desired to avoid local minima. The crux of the approach is to descend to a level l in TD such that there are no more than m points at level l. These points from level l are included in set of initial points I. We then randomly pick a point from I such that it belongs to level l and replace it with its children from level l+1 in

(a) Varying n with fixed (m, d) = (2^{10}, 2^{10})
(b) Varying m with fixed (n, d) = (2^{25}, 2^{10})
(c) Varying m with n = 2^{15} and d = 2^{10}
(d) Varying m with fixed (n, d) = (2^{25}, 2^{10})
This is repeated until we finally have $m$ elements in $I$. The chosen $m$ elements are mapped to parameter space through the inverse link function $g^{-1}()$ and used as initialization. All our experiments use cover tree based initializations. We also make comparisons against random and KMeans++ in Section 8.5.2.

### 8.5.1 Speed

To gauge the speed of Canopy, we begin with inference on GMMs using synthetic data. Working with synthetic data is advantageous as we can easily vary parameters like number of clusters, data points, or dimensionality to study its effect on the proposed method. Note that, from a computational perspective, data being real or synthetic does not matter as all the required computations are data independent, once the cover tree has been constructed.

![Figure 8.4: Plots of cluster purity and loglikelihood of ESCA, Canopy I, and Canopy II on benchmark real datasets–MNIST8m and CIFAR-100. All three methods have roughly same performance on cluster purity. See Section 8.5.2 for more details.](image)

**Synthetic dataset generation** Data points are assumed to be i.i.d. samples generated from $m$ Gaussian probability distributions parameterized by $(\mu^*_i, \Sigma^*_i)$ for $i = 1, 2, \cdots, m$, which mix with proportions given by $\pi^*_i$. Our experiments operate on three free parameters: $(n, m, d)$ where $n$ is the total number of points, $m$ is the number of distributions, and $d$ is the dimensionality. For a fixed $(n, m, d)$, we randomly generate a TRAIN set of $n$ points as follows: (1) Randomly pick parameters $(\mu^*_i, \Sigma^*_i)$ along with mixing proportions $\pi^*_i$, for $i = 1, 2, \cdots, m$, uniformly random at some scale. (2) To generate each point, select a distribution based on $\{\pi^*_i\}$ and sample from the corresponding $d$-dimensional Gaussian pdf. Additionally, we also generate another set of points as TEST set using the same procedure. For all the four models (Canopy I, Canopy II, EM, ESCA), parameters are learnt using TRAIN and log-likelihood on the TEST set is used as evaluation.

**Observations** We run all algorithms for a fixed number of iterations and vary $n, m, d$ individually to investigate the respective dependence on performance of our approach as shown in Figure 8.3. We make the following observations: (1) Overall, Figure 8.3 is in line with our claim that the proposed method reduced the per iteration complexity from $O(nm)$ of EM/ESCA to $O(n + m)$. (2) To illustrate this further, we consider $n = O(m)$ and vary $m$ (shown in Figure 8.3c). While EM and ESCA have per-iteration time of $O(mn)$, i.e., $O(m^2)$ in this case, our Canopy I and Canopy II show $O(m + n)$, i.e., $O(m)$. (3) However, there is no free lunch. The huge speed-up comes at the cost of increased memory usage (for storing the data-structures). For example, in the case of $n = 32$ mil, $m = 4096$, and $d = 1024$ (Figure 8.1), a mere $2\times$ increase in memory gets us a speed up of $150\times$. 

[July 17, 2018 at 16:21 – MT version 0.1]
8.5.2 Correctness

Next, we demonstrate correctness of Canopy using medium sized real world datasets with labels, i.e., ground truth grouping of the points are known. We setup an unsupervised classification task on these datasets and perform evaluation on both cluster purity and log-likelihood.

**Datasets** We use two benchmark image datasets–MNIST8m Loosli, Canu, and Léon Bottou, 2007 and CIFAR-100 Krizhevsky and G. Hinton, 2009. The former contains 8 million annotated handwritten digits of size $28 \times 28$, giving us data points of dimension 784. CIFAR-100, on the other hand, contains 50k images annotated with one of 100 object categories. Each image has 3 channels (RGB) and of size $32 \times 32$, resulting in a vector of dimension 3072.

**Unsupervised classification.** We run unsupervised classification on the above two datasets and evaluate using cluster purity and log-likelihood. Here, cluster purity is defined as the mean of accuracy across all clusters, where each cluster is assigned the class of majority of its members. In addition to using data points as is, we also experiment with unsupervised features learnt from a denoising autoencoder G. Hinton and R. Salakhutdinov, 2006. We extract 30 and 256 dimensional features for MNIST8m and CIFAR-100 respectively. Details of our unsupervised feature extraction are in Appendix ??.

**Observations** Figure 8.4 shows our results on MNIST8m ($m = 10, 100$) and CIFAR-100 ($m = 100, 500$), with error bars computed over 5 runs. Here are the salient observations: (1) All the methods (SEM, Canopy I, Canopy II) have roughly the same cluster purity with Canopy II outperforming in CIFAR-100 (256 dim) and MNIST8m by around 10% and 3% respectively. In CIFAR-100, SEM does slightly better than other methods by 2-3%. (2) Similar results are obtained for log-likelihood except for MNIST8m, where SEM heavily outperforms Canopy. However, note that log-likelihood results in an unsupervised task can be misleading J. Chang, Gerrish, et al., 2009, as evidenced here by superior performance of Canopy in terms of cluster purity.

8.5.3 Scalability - A New Hope

Finally, we demonstrate the scalability of our algorithm by clustering a crawled dataset having more than 100 million images that belong to more than 80,000 classes. We query Flickr\(^1\) with the key words from WordNet Fellbaum98 and downloaded the returned images for each key word, those images roughly belong to the same category. We extracted the image features of dimension 2048 with ResNet HeZhaRenSun15; HeZhaRenSun016 – the state-of-the-art convolutional neural network (CNN) on ImageNet 1000 classes data set–using publicly available pre-trained model of 200 layers\(^2\). It takes 5 days with 20 GPUs to extract these features for all the images. We then use Canopy II to cluster these images with $m = 64000$, taking around 27 hours.

**Observations** For a qualitative assessment, we randomly pick four clusters and show four images (more in Appendix ??) closest to the means in Figure 8.5 (each cluster in a row). We highlight two important observations: (a) Though the underlying visual feature extractor, ResNet, is trained on 1000 semantic classes, our clustering is able to discover semantic concepts that go beyond. To illustrate, images from the first row indicate a semantic class of crowd even though ResNet never received any supervision for such a concept. (b) The keywords associated with

\(^1\)http://www.flickr.com/
\(^2\)github.com/facebook/fb.resnet.torch
these images do not necessarily collate with the semantic concepts in the image. For example, images in first row are associated with key words ‘heave’, ‘makeshift’, ‘bloodbath’, and ‘fullfillment’, respectively. It is not too surprising as the relatedness of retrieved images for a query key word generally decreases for lower ranked images. This suggests that pre-processing images to obtain more meaningful semantic classes could potentially improve the quality of labels used to learn models. Such a cleanup would definitely prove beneficial in learning deep image classification models from weakly supervised data.

8.6 Discussion

We present an efficient sampler, Canopy, for mixture models over exponential families using cover trees that brings the per-iteration cost down from $O(mn)$ to $\tilde{O}(m + n)$. The use of cover trees over both data and clusters combined with alias sampling can significantly improve sampling time with no effect on the quality of the final clustering. We demonstrate speed, correctness, and scalability of Canopy on both synthetic and large real world datasets. To the best of our knowledge, our clustering experiment on a hundred million images is the largest to be reported. We conclude with some related works and future extensions.

Related works There has been work using nearest-neighbor search for guiding graphical model inference like kd-trees Gray and Moore, 2000; Moore, 1999. But use of kd-trees is not scalable with respect to dimensionality of the data points. Moreover, kd-trees could be deeper (especially for small c) and do not have special properties like covering, which can be exploited for speeding up sampling. We observe this empirically when training kd-tree based methods using publicly available code. The models fail to train for dimensions greater than 8, or num-
number of points greater than few thousands. In contrast, our method handles millions of points with thousands of dimensions.

**Further approximations** From our experiments, we observe that using a simplified single observation sampling in Canopy II works well in practice. Instead of descending on the hierarchy of clusters, we perform exact proposal computation for k closest clusters obtained through fast lookup from $T_C$. All other clusters are equally assigned the least out of these k exact posteriors.

In the future, we plan to integrate Canopy with:

**Coresets** Another line of work to speed up mixture models and clustering involves finding a weighted subset of the data, called coreset Feldman, Melanie Schmidt, and Sohler, 2013; Lucic, Bachem, and Krause, 2016. Models trained on the coreset are provably competitive with those trained on the original data set. Such approaches reduce the number of samples n, but perform traditional inference on the coreset. Thus, our approach can be combined with coreset for additional speedup.

**Inner product acceleration** In an orthogonal direction to Canopy, several works AhmRavNarSmo12; Mussmann and Ermon, 2016 have used maximum inner product search to speed up inference and vice versa Auvolat et al., 2015. We want to incorporate these ideas into Canopy as well, since the inner product is evaluated m times each iteration, it becomes the bottleneck for large m and d. A solution to overcome this problem would be to use binary hashing AhmRavNarSmo12 as a good approximation and therefore a proposal distribution with high acceptance rate.

Combining these ideas, one could build an extremely scalable and efficient system, which potentially could bring down the per-iteration sampling cost from $O(mnd)$ to $\tilde{O}(m + n + d)$ or less!
Handling Variable State-Space Size

variable models have become ubiquitous in statistical data analysis spanning a diverse set of applications from text to images to user behaviour. This can be attributed to their versatility in inferring structures & hierarchies from unstructured data. In such models latent variables are introduced to represent these unobserved structures and properties, or in other words the hidden causes of the observed data. However, traditional parametric models using a fixed and finite number of parameters are plagued by under-fitting of data especially in the regime of Big Data (& less frequently over-fitting) whenever there is a misfit between the complexity of the model and the amount of data available. Moreover, model selection is not straightforward for complex latent variable models.

Bayesian nonparametrics such as the Dirichlet Process (DP), the Chinese Restaurant Process, the Indian Buffet Process (IBP) and the Hierarchical Dirichlet Process (HDP) or the Chinese Restaurant Franchise pose to be successful alternatives to model selection and promising results have been reported for describing documents and networks, for modelling user interests and behaviours, for collaborative filtering as well as for priors in multi-task learning. However, inference procedures reported in the academic literature are limited and focused on providing proofs of concept. The scaling of these procedures to be able to deal with real life problems in industry consisting of massive datasets (1 TB+) in a reasonable time with reasonable resources is non-trivial.

One type of very celebrated latent variable model, the Topic models, are useful in modelling problems involving groups of data. In these problems each observation within a group is drawn from a mixture model and it is desirable to share mixture components across all the groups. They are very useful tools in discovering the hidden thematic structure in large archives of documents, and hence the name. Such categorization of data can help us perform tasks such as information retrieval, document browsing, content recommendation (e.g. articles to read, movies to watch, advertisements to display) etc.

Topic models achieve this goal by introducing a discrete latent factor, $z$, for each observation $x \in \mathcal{D}$. This latent factor corresponds to the hidden causes motivating this behaviour, e.g. the underlying theme of the document (finance, sports, etc.) or user tastes. Specifically, we model $x$ as being generated from $p(x|z)$ and the latent factor has its own distribution $p(z)$. For instance, $z$ may be the cluster indicator of a document, in which case it leads to the traditional mixture models. When $z$ is a vector of fixed dimension (i.e. number of topics), it leads to Latent Dirichlet Allocation (LDA). However, we do not know the number of topics prevalent in a given corpus apriori and model selection for LDA is prohibitively expensive. Hence, Bayesian nonparametrics, like a DP has emerged as an attractive choice for $p(z)$, leading to the HDP topic model.

However as mentioned before, given a large set of data $\mathcal{D}$ it is expensive to use traditional inference procedures for such fancy latent variable models using Markov Chain Monte Carlo methods or stochastic variational algorithms. One of the key obstacles in performing scalable
inference in topic models is to draw from \( p(z_{i|k}) \) over the discrete state space associated with the latent factor given the data. The efficiency of this step is critical as it will be repeated for every token in the dataset (which run into billions) multiple times. For example, if we have a distribution over \( K \) outcomes and we draw only a single sample before the distribution changes, then each step will require \( O(K) \) computation.

Recently, a big improvement for parametric topic models was proposed in Aaron Q Li et al., 2014 by making the key observation that in many inference problems the model parameters only change relatively slowly during sampling (e.g., the location of cluster centres, the definition of topics) and appear as decoupled terms in the posterior. Leveraging this observation, they propose generating an alias table \( \text{fishman1996mc} \) for the slow varying parts of the distribution enabling \( O(1) \) sampling from it. Next, to account for the slight drift in the distribution, they apply Metropolis-Hastings (MH) steps with samples from the stale tables acting as the proposal, thus regaining its freshness, all in \( O(1) \). Moreover, this sample generation method can be outsourced to another processor, enabling parallelism.

The problem of scalable inference is further exacerbated in case of nonparametric models and aforementioned speed-up techniques cannot be used directly, mainly for two reasons. The problem is further exacerbated in case of nonparametric models. First, in standard expressions of posterior for Bayesian nonparametrics the slowly varying part is not decoupled from the rest. Second, the size of the discrete state space is no longer pre-determined and keeps changing during the inference procedure. Now, if the size of the state space, \( K \), increases since the generation of the alias table, we cannot use the stale alias tables for MH steps because now the domain of proposal does not contain the domain of target distribution.

In order to circumvent this issue, first of all we derive an alternative parametrization of HDP, wherein the slowly varying terms are decoupled in the posterior. Next we propose a log structured alias sampling data structure for storing the variable size distribution and allowing almost a constant time generation and cheap updates. To extend the idea of Aaron Q Li et al., 2014 for HDP, we take advantage of this data structure by handling a mismatch in the state-space size by updating the proposal cheaply on the fly. Also we describe a multithreaded implementation, leveraging the fact that modern computers have multiple processors. With this new strategy we speed up the inference procedure on HDP significantly, e.g. gaining a 10x speed-up on PubMed and Wikipedia datasets.

The key contribution in our paper is to study sampling algorithms in the context of training latent variable models. This includes both improvements to sampling algorithms (for example, using Walker’s alias method to train HDP) and more efficient ways of computing Stirling numbers. Hence it is worth providing a comprehensive and self-contained overview probabilistic topic models, both flat and hierarchical. Readers intimately familiar with the subject matter will likely only want to consult the discussion below as notation reference.

Let \( F(\phi) \) have density \( f(\cdot|\phi) \) and \( H \) have density \( h(\cdot) \), then the joint density of the model is:

\[
p(X, z, \theta, \phi|\alpha, H) = \prod_{k=1}^{K} h(\phi_k) \prod_{j=1}^{J} p(\theta_j|\alpha) \prod_{l=1}^{N_l} p(z_{j|l} | \theta_j) f(x_{j|l} | z_{j|l}, \phi_{z_{j|l}})
\]

(9.1)

### 9.1 Hierarchical Models

With large numbers of objects (documents, images, users, etc.) flat models are less useful since they fail to capture the amount of detail inherent in the data. For instance, it is only marginally
useful to strive for 10,000 or more topics. The result of such a strategy would be nothing more than an extreme over-segmentation of the data. While this might be useful as feature generator, it fails to capture the inherent structure of the data. Suggestions to the contrary by WanZhaSunetal14 are likely due to the fact that their discriminative model suffers from an unsophisticated parametrization, hence over-segmentation of the space helps.

Instead, hierarchical models allow for both structured and fine-grained information representation and extraction. However, this often renders sampling inference painfully slow due to the large degree of complexity. Even worse, the size and structure of the discrete state space is no longer predetermined and keeps changing during the inference procedure. In short, the cost of scaling latent variable models to large datasets is often superlinear in practice due to inefficient data structures, unless great care is taken.

Hierarchical Bayesian Nonparametrics is an extraordinarily fertile field, leading to a vast range of models such as the Hierarchical Dirichlet Process (TehJorBeaBle06), the Nested Chinese Restaurant Process (BleGriJor10), Pachinko Allocation (LiBleMcC07), the Mondrian Process (TehRoy09), the Indian Buffet Process, or the Nested Chinese Restaurant Franchise (AhmHonSmo13; PaiWanBleJor13), just to name a few of them. It is thus virtually impossible to cover all sampling modifications of the ‘flat’ model in this paper. Instead, we focus on the HDP, as it exhibits all relevant properties and poses associated challenges: structured and time-varying datastructures, dependence between states, and computation of nontrivial combinatorial coefficients.

Table 9.1: A quick reference for notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(K)</td>
<td>Number of topics</td>
</tr>
<tr>
<td>(J)</td>
<td>Number of documents</td>
</tr>
<tr>
<td>(N_j)</td>
<td>Number of tokens in document (j)</td>
</tr>
<tr>
<td>(x_{ji})</td>
<td>Token (i) in document (j)</td>
</tr>
<tr>
<td>(\mathcal{X})</td>
<td>Entire collection of documents, i.e. (\mathcal{X} = {x_{ji}}<em>{i=1}^{N_j}</em>{j=1})</td>
</tr>
<tr>
<td>(z_{ji})</td>
<td>Topic assignment of token (i) in document (j)</td>
</tr>
<tr>
<td>(\theta_j)</td>
<td>Topic-proportions for document (j)</td>
</tr>
<tr>
<td>(\phi_k)</td>
<td>Word distribution for topic (k)</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>Dirichlet prior parameter</td>
</tr>
<tr>
<td>(n(j,k))</td>
<td>Number of tokens assigned to topic (k) in document (j)</td>
</tr>
<tr>
<td>(c(x,k))</td>
<td>Number of times word (x) assigned to topic (k)</td>
</tr>
</tbody>
</table>
### Table 9.2: A quick reference for extra notations used in describing hierarchical models in addition to Table 9.1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m(j, k)$</td>
<td>Number of tables serving dish/topic $k$ in restaurant/document $j$</td>
</tr>
<tr>
<td>$M$</td>
<td>Total number of tables globally, i.e. $M = \sum_{j, k} m(j, k)$</td>
</tr>
<tr>
<td>$u_{ji}$</td>
<td>Auxiliary variable to denote if customer/token $i$ in restaurant/document $j$ was responsible for creation of a new table</td>
</tr>
<tr>
<td>$S^m_n$</td>
<td>Stirling numbers of the first kind</td>
</tr>
<tr>
<td>$(a)_b$</td>
<td>Falling factorial, i.e. $(a)_b = a(a + 1) \cdots (a + b - 1)$</td>
</tr>
</tbody>
</table>

The HDP uses a Dirichlet Process (DP) $G_j$ for each document $j$ to handle uncertainty in number of mixture components. At the same time, in order to share mixture components and clusters across groups, each of these DPs is drawn from a global DP $G_0$.

$$
G_j \sim \text{DP}(\alpha_0, G_0) \\
G_0 \sim \text{DP}(\alpha_0, H) 
$$

We proceed to draw for each word $i$ in document $j$ a topic distribution $\psi_{ji}$ from $G_j$ and a word from it. As before the conjugate pair $(F, H)$ form the language model.

$$
\psi_{ji} \sim G_j \\
x_{ji} \sim F(\psi_{ji})
$$

Exploiting conjugacy, the HDP is best explained via the Chinese Restaurant Franchise metaphor. We have a franchise with the same menu across all of its restaurants. Every document amounts to its own restaurant belonging to this franchise, wherein the customers (words) enter sequentially.

- Whenever a new customer walks into a particular restaurant, he will sit at an existing table with probability proportional to the number of people sitting at the table and share the dish (topic) being served on that table
- Alternatively the new customer will sit at a new table with probability proportional to $\alpha_0$ and order a new dish (topic) $k$ from $G_0$. In this case a phantom customer is sent to the restaurant responsible for $G_0$.
- There, with probability proportional to $\gamma$, a new table is generated with a new dish drawn from $H$. Otherwise, an existing dish is drawn with probability proportional to the number of phantom customers associated with the dish.

$G_0$ acts as a parent restaurant which controls the menu, i.e. the creation or deletion of dishes (topic). This allows documents (or groups) to share statistical strength. By recognizing that draws from a Dirichlet process are discrete almost surely we can express the HDP in a form very similar to LDA as shown in figure below.
In particular, $G_0 \sim \text{DP}(\gamma, H)$ can be represented as (sethug4):

$$G_0 = \sum_{k=1}^{\infty} \theta_{0k} \delta_{\phi_k} \quad \text{where} \quad \phi_k \sim H(\cdot) \quad \text{and} \quad \theta_0 = \{\theta_{0k}\}_{k=1}^{\infty} \sim \text{GEM}(\gamma) \quad (9.4)$$

Here, GEM($\gamma$) denotes the Griffiths, Engen and McCloskey stick breaking process (pitman2006combinatorial). Since $G_0$ has support only at $\{\phi_k\}_{k=1}^{\infty}$, each $G_j$ necessarily has support there as well, and can thus be written as:

$$G_j = \sum_{k=1}^{\infty} \theta_{jk} \delta_{\phi_k} \quad (9.5)$$

It can be shown that each $\theta_j = \{\theta_{jk}\}_{k=1}^{\infty}$ is independently distributed according to $\text{DP}(\alpha_0, \theta_0)$, i.e. $\theta_j \sim \text{DP}(\alpha_0, \theta_0)$ TehjorBeaBlo06. Here we interpret $\theta_0$ and $\theta_j$ as probability measures on natural numbers. As in any mixture model, since each factor $\psi_{ji}$ is distributed according to $G_j$, it takes on the value $\phi_k$ with probability $\theta_{jk}$. To simplify, let $z_{ji}$ be an indicator variable such that $\psi_{ji} = \phi_{z_{ji}}$. Given $z_{ji}$, we have $x_{ji} \sim F(\phi_{z_{ji}})$. Thus we obtain an equivalent representation of the HDP, resembling LDA via the following conditional distributions:

$$\begin{align*}
\theta_0 & \sim \text{GEM}(\gamma) \\
\theta_j & \sim \text{DP}(\alpha_0, \theta_0) \\
z_{ji} & \sim \theta_j \\
x_{ji} & \sim F(\phi_{z_{ji}})
\end{align*} \quad (9.6)$$

Inference in hierarchical models is decidedly more complex. We discuss two main strategies below. This is done for self-consistency and also to derive a computationally more efficient expansion for the associated combinatorial coefficients. We discuss two samplers: sampling by direct assignment (TehjorBeaBlo06) and the more recently proposed sampling by compact representation (CheDuBun11). The latter has the property of mixing considerably faster. In a nutshell, it decouples the issue of provenance, i.e. which level in the hierarchy gives rise to a specific topic from the issue of which instance takes credit for its introduction at that level (which is uniformly distributed over the instances satisfying the former condition). We begin with the more conventional model.

9.1.0.1 Sampling by direct assignment

Similar to the case of LDA to speed up convergence, we would want to collapse out as many variables as possible. However, due to the hierarchy collapsing out all of the variables leads to a complex dependence structure and booking-keeping. As a solution, teh2006hierarchical proposed to collapse out the document specific DPs $G_j$, but retain the global DP $G_0$. The instantiation of $G_0$ (equivalently $\theta_0$) makes the posterior conditioned on $G_0$ to factorize across documents, i.e. sampling for a given document will only depend on counts from the same document given $G_0$. This leads to a much simpler book-keeping as we will see.
Let us begin by deriving the posterior of $z_{ji}$ given other topic assignments $z_{-ji}$, the global DP $G_0$ (equivalently $\theta_0$), and document specific DP $G_j$ (equivalently $\theta_j$) integrated out. Using the exchangeability property of DPs (Antoniak74) for integrating out $\theta_j$ in (9.6), one can find that posterior of $z_{ji}$ depends only on local counts and is given by:

$$z_{ji}|z_{-ji}, \alpha_0, \theta_0 \propto \sum_{k=1}^{K} \frac{n^{-ji}(j,k)}{N_j + \alpha_0} \delta_k + \frac{\alpha_0}{N_j + \alpha_0} \theta_0$$  \hspace{1cm} (9.7)

Similarly at root level, as mentioned earlier, phantom customers from document level restaurants are sent each time a new table is created. Let $m(j,k)$ denote the number of tables in restaurant $j$, serving dish $k$. This means $\bar{m}(k) = \sum_j m(j,k)$ customers are sitting in a table eating dish $k$ at the root level and in total there are $M = \sum_k \bar{m}(k)$ customers at the root level. Then posterior of $G_0$, given table counts in each of the restaurants, is another Dirichlet process:

$$G_0|m, \gamma, H \sim DP \left( M + \gamma, \sum_{k=1}^{K} \frac{\bar{m}(k)}{M + \gamma} \delta_{\phi_k} + \frac{\gamma}{M + \gamma} H \right)$$  \hspace{1cm} (9.8)

Now, to construct a finite instantiation of the posterior $G_0$, we use the definition of DP that any measurable partitions of parameter space under the DP is Dirichlet distributed. In particular, if we consider partition of space into $K + 1$ components where first $K$ corresponds to topics seen till now and the last component is rest of the space, we have:

$$\theta_0 = (\theta_{01}, \theta_{02}, ..., \theta_{0K}, \theta_{0K+1}) \sim \text{Dirichlet}(\bar{m}(1), ..., \bar{m}(K))$$  \hspace{1cm} (9.9)

Next, to incorporate the emission model, note that:

$$\Pr(z_{ji} = k|z_{-ji}, X, \phi, \alpha_0, \theta_0) \propto \Pr(z_{ji} = k|z_{-ji}\alpha_0, \theta_0)f(x_{ji}|\phi_k)$$  \hspace{1cm} (9.10)

Finally, plugging (9.7) into (9.10), we obtain

$$\Pr(z_{ji} = k|\text{rest}) \propto \begin{cases} (n^{-ji}(j,k) + \alpha_0 \theta_{0k})f(x_{ji}|\phi_k) & \text{if } k \text{ in use} \\ \alpha_0 \theta_{0u}f(x_{ji}|\phi_{k,\text{new}}) & \text{if } k \text{ is new} \end{cases}$$  \hspace{1cm} (9.11)

We can further collapse out $\phi_k$, as in the case of LDA.

In the above discussion, we skipped how we obtained table counts, $\bar{m}(k)$ in the first place. As we do not keep track of table creations at document levels (but only keep track which customer is consuming which dish for efficiency), to obtain $\bar{m}(k)$ we have to sample $m(j,k)$ for all $j$.

Note that $m(j,k)$ is just the number of tables for the DP $G_j$, for which the desired conditional distribution of has been shown by Antoniak74 to be:

$$\Pr(m(j,k) = m) \propto S_m^{n(j,k)}(\alpha_0 \theta_{0k})^m.$$  \hspace{1cm} (9.12)

Here the terms $S_m^n$ are Stirling numbers of the first kind.

In summary, this method requires three kinds of sampling, which are very similar to LDA:

- **Sampling z**:

$$\Pr(z_{ji} = k|\text{rest}) \propto \begin{cases} (n^{-ji}(j,k) + \alpha_0 \theta_{0k})f_k^{-ji}(x_{ji}) & \text{if } k \text{ in use} \\ \alpha_0 \theta_{0u}f_{k,\text{new}}^{-ji}(x_{ji}) & \text{if } k \text{ is new} \end{cases}$$  \hspace{1cm} (9.13)
• **Sampling** \( m \):

\[
\Pr(m(j, k) = m) \propto \bar{s}^n_{m}(j, k) (\alpha_0 \theta_{0k})^m. \tag{9.14}
\]

• **Sampling** \( \theta_0 \):

\[
\theta_0 \sim \text{Dirichlet}(\bar{m}(1), \bar{m}(2), ..., \bar{m}(K), \gamma). \tag{9.15}
\]

### 9.1.0.2 Sampling by table configuration

The key difference of **CheDuBun11** relative to **TehJorBeaBle06** is that rather than keeping track of relative assignments of tables to each other it simply keeps track of the level (global topic restaurant, current document restaurant, or no new table), denoted by \( u_{ji} \), within the hierarchy of restaurants at which an individual customer opens a new table, if any. The advantage is that this allows us to factor out the relative assignment of customers to specific tables but rather only keep track of the dishes that they consume, very similar to LDA.

Using the table configuration model, sampling in the HDP occurs through the following process: in addition to the topic selector variable \( z_{ij} \), we use the auxiliary variable \( u_{ji} \) to determine whether the dish existed before \((u_{ji} = 2)\), whether a new table was allocated \((u_{ji} = 1)\), or whether a new dish and a new table were allocated \((u_{ji} = 0)\). The reason for this decomposition is that for any topic we only need one token \( u_{ji} = 0 \) to instantiate the topic and only one per restaurant to instantiate a table \( u_{ji} = 1 \). Thus, the counts, \( m(j, k) \) for table serving dish \( k \) in restaurant \( j \) can be expressed in terms of \( u_{ji} \)'s as \( m(j, k) = \sum_l I[z_{ji} = k, u_{ji} < 2] \). Note that **CheDuBun11** discuss this in the context of arbitrary hierarchies, where \( u_{ji} \) could be as large as the depth of the hierarchical model. The benefit of this auxiliary variable decomposition is that we need not keep track of \( u \) explicitly but rather just resample one variable at a time as needed, as pointed out by **CheDuBun11**.

The joint posterior of \( z \) and \( u \) for the HDP is

\[
\Pr([z, u]) = \frac{\gamma^K}{(\gamma)^K} \left( \prod_{k=1}^{K} \frac{(\bar{m}(k) - 1)!}{\bar{m}(k)!} \right) \left( \prod_{j=1}^{J} \frac{\alpha_0^{n_{0j}}}{\alpha_0} \prod_{k=1}^{K} \frac{s_{n(j, k)}^{m(j, k)}}{n(j, k)!} \right) \tag{9.16}
\]

where \((x)_n\) is the falling factorial \( x(x+1) \cdots (x+n-1) \). We can introduce the emission model as before in (9.10) as well as marginalize out \( \phi \) as before. To obtain conditional posteriors for a particular \( z_{ji}, u_{ji} \), we just divide \( \Pr([z, u]) / \Pr([z, u]^{-j,i}) \) and simplify the resulting expression to obtain \( \Pr(z_{ji}, u_{ji} | [z, u]^{-j,i}) \).

In summary, we have the following (unnormalized) probabilities for document \( j \) for a word at position \( i \):

**No change —** \( u_{k1} = 2 \): Whenever we do not change the number of tables in a restaurant, we have

\[
\Pr(z_{ji} = k, u_{ji} = 2 | \text{rest}) \propto \frac{s_{m(j, k)}^{n(j, k) + 1}}{s_{m(j, k)}^{n(j, k)}} \frac{n(j, k) + 1 - m(j, k)}{n(j, k) + 1} f_{k}^{-j,i}(x_{ji}) \tag{9.17}
\]
New table — $u_{ji} = 1$: Whenever the customer starts a new table by picking one of the existing dishes in the global menu we have

$$\Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) \propto \frac{\alpha_0 \bar{m}^2(k)}{(\bar{m}(k) + 1)(M + \gamma)} \frac{\gamma^5}{\bar{m}(j,k) + 1} \frac{f_{-ji}(x_{ji})}{m(j,k) + 1}$$  \hspace{1cm} (9.18)

New table, new global dish — $u_{ji} = 0$: In this case we get

$$\Pr(z_{ji} = k^{\text{new}}, u_{ji} = 0|\text{rest}) \propto \frac{\alpha_0 \gamma}{M + \gamma} f^{\text{new}}_{ji}(x_{ji})$$  \hspace{1cm} (9.19)

Note that while the probabilities in (9.17)-(9.19) are unnormalized, they share the same normalization constant. Hence joint normalization is immediately available. Note that the original paper and its errata had some bugs, hence we re-derived the expressions.

### 9.1.1 Convergence and Composition of MCMC Algorithms

Over the past decade many different MH samplers for topic models have been proposed. To understand their relative merits we need to review some background on the mixing and composition properties of Markov Chain Monte Carlo algorithms.

We denote by $\mathcal{K}(x, y)$ the transition kernel between subsequent states $x$ and $y$ of a Markov Chain. In Metropolis-Hastings sampling we have the following:

$$\mathcal{K}(x, y) = \mathcal{A}(x, y)q(y|x) + (1 - r(x))\delta(y - x) \text{ where } r(x) = \int \mathcal{A}(x, y)q(y|x)dy.$$  \hspace{1cm} (9.20)

Under mild conditions, it is shown that such chains are ergodic with equilibrium being the target distribution (Tierney1994markov). However, MCMC is efficient only when the chain converges quickly.

In MH this is governed by an effective choice of $p$. A common misconception is that getting higher acceptance probability in MH implies faster convergence. As counterexample consider using $\mathcal{N}(5, 1)$ as proposal to sample from $\frac{1}{2} \left[ \mathcal{N}(5, 1) + \mathcal{N}(-5, 1) \right]$ as target distribution. If we begin near 5 the acceptance ratio would almost always be close to 1, yet the chain would be extremely slow in exploring the true distribution. Below we list a number of proposals whose mixing behaviour has been extensively studied.

**Random walks**

In general the selection of the next state depends on the current state. Common choices include choosing a neighbor uniformly at random in the case of graphs or a normal distribution centered around current position in the case of continuous space. Ergodicity (irreducible and aperiodic) can be proved e.g. when the proposal is non-zero on the entire state space. Likewise, if the state space is open and connected, then proposal being non-zero just on a neighbourhood of current position suffices. Rosenthal1995convergence shows that the mixing rate of the Markov chain is determined by the second largest eigenvalue of $\mathcal{K}$. The smaller it is the faster the chain converges.

**Independence chain**

A degenerate case (that we will use) is when the selection of the next state is independent of the current state, i.e. $q(j|i) = q_i$ for all $i,j$ and that moreover the support of $q$ contains that of $p$. Convergence of this proposal was analyzed by Mengersen1996rates.
Theorem 9.1  The Markov chain under the independence proposal is uniformly ergodic if there exists $C \geq 1$ such that

$$p_k \leq C q_k \text{ for all } k \quad (9.21)$$

In this case the total variation distance between $p$ and probability distribution of the chain at time $t$, i.e. $P^t$ is bounded by

$$\| P^t - p \|_1 \leq (1 - C^{-1})^t. \quad (9.22)$$

Clearly, the closer $C$ is to 1, the faster the rate of convergence. Hence, $\max_k p_k / q_k$ can be used to indicate the goodness of the proposal. Note that it is same as for rejection sampling, albeit without the requirement to know the $C$ explicitly.

**Hybrid methods**

A great benefit in designing MCMC proposal distributions is that we can combine them into a hybrid proposal. Suppose $K_1, ..., K_r$ are transition kernels with the same invariant distribution $p$, then we can combine them e.g. sequentially or in parallel.

**Parallel composition.** We define the mixture kernel via

$$K = \sum_{i=1}^{r} \alpha_i K_i \text{ where } \alpha_i \geq 0 \text{ and } \| \alpha \|_1 = 1. \quad (9.23)$$

Here $\alpha$ defines a probability distribution over $K_i$. At each step one of the kernels is selected according to $\alpha$.

**Sequential composition.** We may carry out the kernels in sequence, i.e.

$$K = K_1 \circ K_2 \circ \cdots \circ K_r. \quad (9.24)$$

Each kernel is used in turn, and when the last one is the cycle is restarted. This is desirable, e.g. whenever the eigenfunctions of the $K_i$ differ considerably. In this poor convergence with regard to some mode in one kernel can be mitigated by improved convergence in another kernel and vice versa.

While individual kernels might not be ergodic, their composition can be. Unfortunately, exact convergence analysis is only known for very limited cases ([tierney1994markov]).

**Gibbs sampling and variants**

Perhaps the most famous example of the composition principle is the Gibbs sampler. In it, each kernel $K_i$ samples only one component (or a subset of variables) at a time. That is,

$$K_i(x^t, x^{t+1}) = \begin{cases} 0 & \text{if } x_j^{t+1} \neq x_j^t \text{ for any } j \neq i, \\ p(x_i^{t+1}|x_{-i}^t) & \text{otherwise} \end{cases} \quad (9.25)$$

Clearly $K_i$ has $p$ as its stationary distribution. However, it is not irreducible. Neither of the kernels $K_i$ on their own possess ergodicity, but after parallel or sequential composition the chain becomes ergodic. See [Gonzalezetal11](#) for a discussion and analysis.

Whenever sampling from the conditional distribution $p(x_i|x_{-i})$ is too costly, we may replace $K_i$ by an MH-step, i.e. by a sub-kernel that has $p(x_i|x_{-i})$ as its invariant distribution. This
preserves mathematical validity, i.e. the chain remains ergodic and the stationary distribution is still the true posterior.

In the following we will combine the above sampling techniques for efficient inference in Bayesian latent variable methods. This will be accomplished by smart datastructures, Metropolis-within-Gibbs, and by using the composition principle, such as to speed up topic model inference on large models and data considerably.

9.1.1 Parallelization

All of the method we discussed can be easily parallelized, if we tolerate a small approximation. Modern computational resources are heavily multi-threaded and an efficient inference procedure should utilize this structure. All the inference strategies discussed keeps track and operates of two kinds of data:

- **Global statistics** which represents the word given topic counts, and
- **Local statistics** consisting of the latent topic indicators and topic given document counts.

There is a common framework behind each strategy, i.e. reading the current statistics, computing the conditional distribution and sampling from it and finally updating the statistics.

Reading and writing to the local statistics present no obstacles to parallelizing and can be handled independently by individual threads. Now the key idea for parallelizing the sampler is to observe that the global topic distribution and the topic given word counts (which we will refer to as state of the system) change only little given the changes in a single document (we may have millions of documents). Hence, we can assume that \( \hat{c}(k) \) and \( c(x,k) \) are essentially constant while sampling topics for a document. This means that instantaneously updating \( c(k) \) and \( c(x,k) \) or not during the sampling process would hardly have any manifestations and we can defer this action to a separate thread without much repercussions. This setup allows us to execute a large number of sampling threads simultaneously, speeding up the inference considerably as we will in the see in the experiments (Section 9.3).

The details of the parallelization framework are as follows: We propose to use create two thread pools. One is tasked with maintaining the data structures and the other pool consists of sampling threads which sample the documents. After drawing a new sample the sampling thread pushes into a FIFO queue (often implemented as circular buffer) and a thread from the other pool pops from this queue the sample to update the global \( c(x,k) \) table and any associated data-structure. This way one set of threads only do writes and another only reads, thereby minimizing locks required (can be eliminated completely by using atomics). Further, the writing threads can be sharded according to words, i.e. each writing thread is responsible for a subset of words thereby eliminating any contention among the writing threads. In practice, we only create 1 or few threads for the maintaining the data structure, but use many more threads, at least the number of available CPU cores, for the sampling pool. The framework is illustrated in Figure 9.1.

9.2 SPEEDING UP INference IN HIERARCHICAL MODELS

In this section we show how techniques developed for speeding up flat models in Section ?? can be extended to hierarchical non-parametric models. Scalable inference is rather more complex in this case than in LDA. Firstly, the model does not decompose into local sparse terms and
9.2 Speeding up inference in hierarchical models

Figure 9.1: Framework
global, slowly varying terms, since local choices may affect the composition in terms of topics all the way throughout the HDP. Second, the size of the discrete state space is no longer pre-determined and keeps changing during the inference procedure. This makes it difficult to design fast samplers that precompute parts of the distribution.

9.2.1 Exploiting Sparsity

In order to speed-up the sampler by leveraging sparsity, one needs to identify a partition of posterior having sparse terms, as in case of LDA (Section ??, ??). In case of inference on HDP using Sampling by Direct Assignment (Section 9.1.0.1), due to resemblance of posterior to LDA, one can directly use same partition of the posterior into a sparse local document-only term and a slowly varying global word-topic count term as in LDA. Thus same techniques as presented for LDA (Section ??) directly apply. Whereas in the other case of inference for HDP utilizing the faster mixing Sampling by Table Configuration (Section 9.1.0.2) no straightforward decoupling can be seen. The probability of sitting at an existing table in the restaurant, (9.17), is sparse and depends only on counts within restaurants. However, the probability of starting a new table, (9.18), depends both on local and global counts and must be evaluated for all active topics, which would not be sparse.

We now develop a reformulation of the posterior under Sampling by Table Configuration by separating out the sparse terms. We begin by observing that for topics which have no representation in the current restaurant, i.e. when \( n(j, k) = 0 \) and \( m(j, k) = 0 \), the probability in (9.18) just becomes:

\[
\Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) \propto \frac{\alpha_0 \bar{m}^2(k)}{\bar{m}(k) + 1}(M + \gamma) f_k(x_{ji}) := \pi
\] (9.26)

Now we show that for topics which are already represented in the current restaurant, i.e. \( n(j, k) > 0, m(j, k) > 0 \) we can collect \( \pi \) separately as:

\[
\Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) \propto \pi \frac{\sum_{m(j, k) + 1}^{n(j, k) + 1} m(j, k) + 1}{\sum_{m(j, k)}^{n(j, k)}} = \pi \left( \frac{\sum_{m(j, k) + 1}^{n(j, k) + 1} m(j, k) + 1}{\sum_{m(j, k)}^{n(j, k)}} - 1 \right) + \pi
\] (9.27)

If term in the bracket is non-zero, then it can become a local sparse distribution on its own and the leftover \( \pi \) can be combined with distribution of (9.26). But for this split to yield valid distributions, we need to prove that the term in the bracket is non-negative. To prove this, we will first need a couple of lemma.

**Lemma 9.2** For all \( m \leq n \in \mathbb{N} \), we have

\[
\frac{m + 1}{n + 1} \left( \frac{2n}{m} - 1 \right) \geq 1
\] (9.28)

**Proof** It is equivalent to show that:

\[
\frac{m + 1}{n + 1} \left( \frac{2n}{m} - 1 \right) - 1 \geq 0
\] (9.29)
Simplifying the LHS we obtain:

\[
\text{LHS} = \frac{m+1}{n+1} \left( \frac{2n}{m} \right)^2 - 1
\]

\[
= \frac{(m+1)(2n-m) - m(n+1)}{n+1}
\]

\[
= \frac{-(m+2)(m-n)}{(n+1)}
\]

(9.30)

As a concave quadratic is always non-negative between its two roots, the LHS is always non-negative for \(-2 \leq m \leq n\) for all \(n\). This completes our proof. 

\[
\text{Lemma 9.3} \quad \text{For all } n \in \mathbb{N}, \text{ the following sequence is non-increasing in } m:
\]

\[
a_{n,m} = \frac{m}{n-m} \frac{S_{m+1}^n}{S_m^n}, \quad m = 1, 2, \ldots, n - 1
\]

(9.31)

In other words \(a_{n,1} \geq a_{n,2} \geq \cdots \geq a_{n,n-1}\).

\[
\text{Proof} \quad \text{We prove the non-increasing sequence of } a_{n,m} \text{ in } m \text{ by induction. The induction hypothesis is equivalent to}
\]

\[
(m-1)(n-m) (S_m^n)^2 - m(n-m+1)S_{m+1}^n S_{m-1}^n \geq 0
\]

(9.32)

The base case for induction using \(n = 2, m = 1\) holds trivially as \(S_0^2 = 0\). Assume the statement holds till \(n\), then for \(n+1\) we need to prove:

\[
(m-1)(n-m+1) (S_{m-1}^n)^2 - m(n-m+2)S_{m+1}^n S_{m-1}^n
\]

(9.33)

We begin by simplifying the LHS, which would reveal that each term in the LHS is indeed positive or non-negative by the induction hypothesis.

\[
\text{LHS} = (m-1)(n-m+1) (S_m^n)^2 - m(n-m+2)S_{m+1}^n S_{m-1}^n
\]

\[
= (m-1)(n-m+1) (S_{m-1}^n + nS_m^n)^2
\]

\[
- m(n-m+2) (S_m^n + nS_{m+1}^n) (S_{m-2}^n + nS_{m-1}^n)
\]

\[
= (m-1)(n-m+1) (S_{m-1}^n)^2 - m(n-m+2)S_m^n S_{m-2}^n
\]

\[
= a + b + c
\]

(9.34)
Upon simplifying a further, we obtain:

\[
\begin{align*}
a &= (m - 1)(n - m + 1) \left( S_m^n \right)^2 - m(n - m + 2)S_m^n S_{m+1}^n \\
&= \frac{m}{m-1} \left[ (m - 2)(n - m + 1) \left( S_m^{n-1} \right)^2 - (m - 1)(n - m + 2)S_m^n S_{m+1}^n \right] \\
&\quad + \frac{n - m + 1}{m - 1} \left( S_m^{n-1} \right)^2, \\
&\geq \frac{n - m + 1}{m - 1} \left( S_m^{n-1} \right)^2.
\end{align*}
\]

(9.35)

where the last inequality follows by applying induction hypothesis. Similarly, proceeding for \( b \) one obtains:

\[
\begin{align*}
b &= n^2(m - 1)(n - m + 1) \left( S_m^n \right)^2 - n^2m(n - m + 2)S_{m+1}^n S_m^n \\
&= n^2 \frac{n - m + 2}{n - m + 1} \left[ (m - 1)(n - m) \left( S_m^n \right)^2 - m(n - m + 1)S_{m+1}^n S_m^n \right] \\
&\quad + n^2 \frac{m - 1}{n - m + 1} \left( S_m^n \right)^2, \\
&\geq n^2 \frac{m - 1}{n - m + 1} \left( S_m^n \right)^2.
\end{align*}
\]

(9.36)

again where the last inequality follows by applying induction hypothesis. Finally, we simplify \( c \),

\[
\begin{align*}
c &= [n(n - 2)(n - m) - 2n] S_m^n S_{m+1}^n - nm(n - m + 2)S_{m+1}^n S_m^n \\
&\geq -2n S_m^n S_{m+1}^n.
\end{align*}
\]

(9.37)

To see the last inequality, note again from induction hypothesis that \( a_{n,m-2} \geq a_{n,m} \), i.e.

\[(m - 2)(n - m)S_m^n S_{m-1}^n - m(n - m + 2)S_{m+1}^n S_{m-2}^n \geq 0\]

(9.38)

Substituting the simplified expressions for \( a, b, c \) back into (9.34):

\[
\begin{align*}
\text{LHS} &= a + b + c \\
&\geq \frac{n - m + 1}{m - 1} \left( S_m^{n-1} \right)^2 + \frac{m - 1}{n - m + 1} \left( S_m^n \right)^2 - 2n S_m^n S_{m+1}^n \\
&= \left( \sqrt{\frac{n - m + 1}{m - 1} S_m^{n-1}} - n \sqrt{\frac{m - 1}{n - m + 1} S_m^n} \right)^2, \\
&\geq 0.
\end{align*}
\]

(9.39)

This completes the proof.

\[\square\]

**Theorem 9.4** The Stirling number of first kind satisfies the following inequality for all \( 0 \leq m \leq n \):

\[
\frac{m + 1}{n + 1} \frac{S_{m+1}^n}{S_m^n} \geq 1
\]

(9.40)

**Proof** From Lemma 9.3, we have that \( a_{n,m} \) is an non-increasing sequence in \( m \) for all \( n \). Thus every element \( a_{n,m} \) in the sequence is lower bounded by the last element \( a_{n,n-1} = 2/n \), i.e.

\[a_m \geq a_{n-1} = 2/n\]

As a result

\[
\frac{S_{m+1}^n}{S_m^n} \geq \frac{2(n - m)}{nm}
\]

(9.41)
Finally to show the original claim (9.40), we use Lemma 9.2 and the recurrence relation on $S_{m+1}^{n+1}$ to obtain the desired result:

$$
m + 1 \frac{S_{m+1}^{n+1}}{n+1} \geq m + 1 \left( 1 + \frac{n S_{m+1}^n}{S_m^n} \right) \geq m + 1 \left( 2n \frac{m}{m} - 1 \right) \geq 1.
$$

(9.42)

This result allows us to partition the posterior for sampling table configurations into sparse local terms and slowly varying global terms, as follows:

**No change** — $u_{ki} = 2$: As before, whenever we do not change the number of tables in a restaurant, we have

$$
Pr(z_{ji} = k, u_{ji} = 2|\text{rest}) \propto \frac{S_{m(j,k)}^{n(j,k)+1}}{S_{m(j,k)}^n} \frac{n(j,k) + 1 - m(j,k)}{n(j,k) + 1} f_k(x_{ji})
$$

(9.43)

**New table among existing local dishes** — $u_{ji} = 1$: Whenever the customer starts a new table by picking one of the existing dishes in the restaurant we have

$$
Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) \propto \alpha_0 \frac{m^2[k]}{(m(k) + 1)(M + \gamma)} \left( \frac{S_{m(j,k)}^{n(j,k)+1}}{S_{m(j,k)}^n} \frac{m(j,k) + 1}{n(j,k) + 1} - 1 \right) f_k(x_{ji})
$$

(9.44)

**New table** — $u_{ji} = 1$: Whenever the customer starts a new table picking one of the existing dishes from the global menu (existing dish among the restaurant or new local dishes from the global menu) we have

$$
Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) \propto \alpha_0 \frac{m^2[k]}{(m(k) + 1)(M + \gamma)} f_k(x_{ji})
$$

(9.45)

**New table, new global dish** — $u_{ji} = 0$: In this case we get

$$
Pr(z_{ji} = k^{\text{new}}, u_{ji} = 0|\text{rest}) \propto \frac{\alpha_0 \gamma}{M + \gamma} f_{k^{\text{new}}}(x_{ji})
$$

(9.46)

Careful inspection of the posterior illustrates that (9.43) and (9.44) are the sparse local document terms (after all, no new dish is added to the local document), whereas (9.45) is the slowly varying global term. Thus, in this formulation of HDP the sparse and slowly varying terms are decoupled as desired. Figure 9.2 shows an instance of unnormalized posterior after the decoupling which exhibits the division of sparse local quantities and dense global ones explicitly. This allows us to exploit sparsity in a manner similar to LDA.
9.2.2 Log Structured Alias Sampling

The resemblance of the sampling by direct assignment (Section 9.1.0.1) tempts one to immediately borrow the alias sampling trick. One might consider to partition the posterior of $z_{ji}$ as before into a sparse component and a slowly varying one.

$$\Pr(z_{ji} = k | \text{rest}) \propto \begin{cases} n_{kj}^{-ji} f_{k}^{-ji}(x_{ji}) & \text{sparse local terms} \\ \alpha_0 \theta_{0k} f_{k}^{-ji}(x_{ji}) & \text{slowly varying global terms} \\ \alpha_0 \theta_{0u} f_{k_{\text{new}}}^{-ji}(x_{ji}) & \text{singleton} \end{cases} \quad \text{if } k \text{ in use}$$

$$\text{if } k \text{ is new}$$

(9.47)

Unlike the case of LDA, now the size of the discrete state space is no longer pre-determined and keeps changing during the inference procedure. This makes it difficult to precompute parts of the distribution and store it efficiently in some data structure. This is because it is costly to recompute or update the data structure every time a new topic is created/deleted for all the words.

To circumvent this issue, we propose a Log Structured Alias Sampler. It is inspired by log-structured merge data-structures such as BigTable (F. Chang et al., 2006). We process and store the counts for prevalent topics in alias tables and the counts for nascent topics are lazily stored in a list as a log as shown in Figure 9.3. Then once in a while we merge growing topics from the nascent list into the main alias table. To sample from this data structure costs $O(K_{\text{nascent}})$, as we can efficient draw from prevalent topics in $O(1)$ time, but we traverse the list of nascent topics. Typically, the number of nascent topics and the probability mass associated with it is very small. As a result it does not cause a huge performance penalty as compared to direct sampling from alias table.

Also we incorporate a reverse index for the log nascent topics to enable $O(1)$ lookup of their counts which is required in accept/reject step of Metropolis-Hastings.
9.2.3 Dealing with Stirling Numbers

In case of Sampling by Direct Assignment, we face Stirling’s number in posterior of the root DP in form of Antoniak’s distribution. Using the following trick we can sample all of $m$ from Antoniak’s distribution with a cost equal to that of flipping a coin per token in the corpus.

1. Draw auxiliary variables $s_{l|jk} \sim \text{Bernoulli}\left(\frac{\alpha_0 \theta_{0k}}{\alpha_0 \theta_{0k} + l - 1}\right)$ for $l \in \{1, \ldots, n_{kj}\}$.

2. A sample $m$ from the Antoniak distribution can be computed by $m = \sum l s_{l|jk}$.

Lemma 9.5 Denote by $A(n, \theta)$ the Antoniak distribution. That is, it denotes the distribution where $p(x = m) \propto S_n^m \theta^m$ for $m \in \{1, \ldots, m\}$ and where $S_n^m$ is the Stirling number of the first kind. Moreover, assume that we have $m$ Bernoulli random variables $s_l$, drawn from $s_l \sim \text{Bin}\left(\frac{\theta}{\theta + l - 1}\right)$ for $l \in \{1, \ldots, m\}$. Then $s := \sum_{l=1}^n s_l$ satisfies $s \sim A(n, \theta)$. Hence sampling from $A(n, \theta)$ can be accomplished in $O(n)$ time and $O(1)$ space.

Proof We begin with the defining property of the Stirling numbers, namely that they are the coefficients of a polynomial of degree $n$ given by $\frac{\Gamma(\theta + n)}{\Gamma(\theta)} = \theta(\theta + 1) \cdots (\theta + n - 1) =: \sum_{m=1}^n S_n^m \theta^m \quad (9.48)$

By construction $S_0^0 = 0$, hence it is omitted from the expansion. We obtain

\[
\Pr \left[ \sum_{l=1}^n s_l = m \right] = \sum_{s_1, \ldots, s_n | s = m} \Pr [s_1, \ldots, x_n] \\
= \frac{\theta^m}{\prod_{l=1}^n (\theta + l - 1)} \sum_{\{s_1, \ldots, s_m\} | s = m} \prod_{s_l = 0} (i - 1) \quad (9.50)
\]

\[
= \frac{\theta^m}{\prod_{l=1}^n (\theta + l - 1)} S_n^m = \frac{\theta^m S_n^m}{\sum_{l=1}^n \theta^l S_l^m} \quad (9.51)
\]

Here the equality (9.50) follows from the fact that we need to multiply over all terms $1 - \frac{\theta}{\theta + l - 1} = \frac{l - 1}{\theta + l - 1}$ for which $s_l = 0$. Moreover, (9.51) follows from the definition of the Stirling number of the first kind by decomposing $(\theta + i - 1) = \theta + (i - 1)$ and the last equality follows from the property of $\theta^n$. ■

In case of Sampling by Table Configurations, many terms of the conditional posterior involves Stirling numbers. However, Stirling numbers rapidly become very large and even computation
of Stirling numbers $S^n_m$ for moderately large $n$ and $m$, which will frequently occur for large text corpus, can cause numerical instability. Hence one would typically resort to store and compute them in log domain. The recursion for Stirling number yields:

$$\log S_{m+1}^{n+1} = \log S_m^n + \log \left( \exp \left( \log S_m^n - \log S_{m+1}^n \right) + n \right)$$

(9.52)

But such implementation can be slow as the log() and exp() functions can make the evaluation slow if implemented naively.

Fortunately, for sampling of hierarchical models we only need the Stirling numbers in the following forms:

$$U^n_t = \frac{S^n_{m+1}}{S^n_m} \quad V^n_t = \frac{S^n_{m+1}}{S^n_{m+1} + n}$$

(9.53)

since only the ratios are required, we can directly compute them. Buntine02 showed that just by manipulating the recursion obeyed by the Stirling numbers, we can obtain the following recursion directly for the ratio:

$$U^n_m = U^{n-1}_{m-1} + n - (n-1) \frac{U^{n-1}_{m-1}}{U^{n-1}_m}$$

(9.54)

with $U^0_0 = 1, U^1_0 = \infty, U^1_1 = 1$. Also in backwards direction as:

$$U^n_m = U^{n+1}_{m+1} - (n+1) + n \frac{U^{n+1}_{m+1} - (n+1)}{U^n_{m+1} - n}$$

(9.55)

At every sampling instance of HDP, $n$ or $m$ can change by at most $\pm 1$. So we can cache a few rows of Stirling numbers around the presently active values. If during sampling we move towards the boundary we can re-compute some more rows and discard values at the other end. Furthermore, numerically using the recursion (9.54) has been shown to be much more stable, even when using float instead of double (Buntine02). We do not need to cache the other ratio $V^n_m$ separately, as it can be readily expressed in terms of $U^n_m$ as follow:

$$V^n_m = \frac{U^{n+1}_m}{U^n_{m+1} - n}$$

(9.56)

To summarize, in this section we explored ways to handle Stirling numbers which are hard compute in numerically stable way. In case of sampling by direct assignment, we found an alternate way to sample from Antoniak’s distribution which totally avoids use of Stirling’s number. The new sampling strategy is much cheaper, costs $O(n)$ per token and needs no caching memory. In case of sampling table configuration, we only need ratios of certain Stirling number and we found a numerically more stable and faster recursion to directly compute the ratios.

When all of speed-up approaches, i.e. leveraging sparsity, log structured alias sampling, and tackling Stirling numbers, are combined a significant gain in speed and scalability is achieved for inference on HDP as will be seen from experiments presented in next section. Furthermore, parallelization scheme same as that for flat models (Section 9.1.2) can be applied to HDP as well.
9.3 EXPERIMENTAL STUDY

We now present empirical studies for the various fast sampling techniques in order to study (i) computation speed (Section 9.3.1), and (ii) convergence speed (Section 9.3.2). For these studies, we evaluate on both flat (Section ?? and hierarchical models (Section 9.1), with Dirichlet-Categorical emission model, i.e.

\[ f_k(x) = \frac{c(x, k) + \beta}{c(k) + \beta} \quad \text{(Dirichlet-Categorical)} \]  

(9.57)

Using this emission model results in widely used probabilistic models LDA and HDP-LDA. However, the proposed method can be applied effortlessly to any emission model like Normal-Inverse Wishart in case of topic modeling through Gaussian latent Dirichlet allocation (Gaussian LDA) DasZahDye. We pick Dirichlet-Categorical emission model due to their widespread application in various fields spanning natural language processing, computational biology, user activity modeling etc.

METHODS In case of LDA, we compare both the single-threaded and multi-threaded version (Section 9.1.2) of the following samplers:

- Vanilla LDA: A collapsed Gibbs sampler that directly utilizing the posterior (??) without any speed-up tricks or optimizations as proposed by griffiths2004finding.
- Sparse LDA: We leverage the inherent sparsity by splitting the posterior appropriately as proposed by yao2009efficient and discussed in Section ??.
- F+Nomad LDA: We use Fenwick tree data structure to speed up sampling from dense part of the posterior as proposed by yu2014scalable and discussed in Section ??.
- Alias LDA: We use a MH-within-Gibbs approach using a stale version global counts in a alias as a super cheap proposal distribution for the dense part of the posterior as proposed by LiAhmRavSmo and discussed in Section ??.
- Light LDA: We use a MH-within-Gibbs approach with a proposal cycling between document-count based proposal and stale word count based proposal as proposed by yuan2014lightlda and discussed in Section ??.

In case of HDP, we compare again both the single-threaded and multi-threaded version (Section 9.1.2) of the following samplers:

- SDA HDP: Gibbs sampler through sampling by direct assignment (without any speed-up tricks) as proposed by teh2006hierarchical and discussed in Section 9.1.0.1.
- Alias HDP: Improved version of SDA HDP by leveraging sparsity with log structured alias sampling (Section 9.2.2) and fast Antoniak sampling using Lemma 9.5.
- STC HDP: Gibbs sampler through sampling table configuration (without any speed-up tricks) as proposed by CheDuBun11 and discussed in Section 9.1.0.2.
- SSTC HDP: Improved version of STC HDP by utilizing modified posterior (Section 9.2.1) with log structured alias sampling (Section 9.2.2) and direct evaluation of ratio of Stirling numbers (Section 9.2.3).

We follow the tradition in setting \( \alpha = 50/K \) and \( \beta = 0.01 \) where \( K \) is the number of topics for all the methods (griffiths2004finding; yao2009efficient). In order to compare speed we use the
execution time as the metric. To measure the accuracy and convergence, the log-likelihood on a held out test set $X_{\text{test}}$ is computed as follows on every dataset:

$$\text{LogLikelihood}(X_{\text{test}}) = \frac{1}{|X_{\text{test}}|} \sum_{i \in X_{\text{test}}} \frac{1}{N_j} \sum_{i=1}^{N_j} \log p(x_{ji}|\text{Learnt model})$$ (9.58)

SOFTWARE & HARDWARE All the algorithms are implemented multithreaded in simple C++ using a distributed setup. Within a node, parallelization is implemented using the work-stealing Fork/Join framework, and the distribution across multiple nodes using the process binding to a socket over MPI. We run our experiments on a cluster of 16 Amazon EC2 c4.8xlarge nodes. There are 36 virtual threads per node and 60GB of memory. For purpose of experiments, all data and calculations are carried out at double floating-point precision. For random number generation we employ Intel® Digital Random Number Generators through instruction RDRAND, which uses thermal noise within the silicon to output a random stream of bits at 3 Gbit/s, producing true random numbers.

datasets The performance of all methods are evaluated against nine real-world datasets (Table 9.3). These datasets cover a various range in corpus size, which helps to test the accuracy and efficiency of each method at different data scale. The statistics for each of the dataset is given over Table 9.3 after cleaning by removing stop words and rare words.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>V</th>
<th>L</th>
<th>D</th>
<th>L/V</th>
<th>L/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 News</td>
<td>18,127</td>
<td>1,191,840</td>
<td>11,266</td>
<td>65.75</td>
<td>105.79</td>
</tr>
<tr>
<td>HEP</td>
<td>37,729</td>
<td>1,548,935</td>
<td>27,770</td>
<td>41.05</td>
<td>55.78</td>
</tr>
<tr>
<td>NIPS$^i$</td>
<td>12,305</td>
<td>2,301,375</td>
<td>1740</td>
<td>187.03</td>
<td>1322.63</td>
</tr>
<tr>
<td>Reuters</td>
<td>69,973</td>
<td>2,624,373</td>
<td>14,377</td>
<td>37.51</td>
<td>182.54</td>
</tr>
<tr>
<td>Enron$^i$</td>
<td>28,102</td>
<td>6,412,174</td>
<td>40,861</td>
<td>228.18</td>
<td>160.86</td>
</tr>
<tr>
<td>ACM</td>
<td>133,325</td>
<td>12,258,310</td>
<td>132,032</td>
<td>41.83</td>
<td>93.06</td>
</tr>
<tr>
<td>NY Times</td>
<td>101,330</td>
<td>99,542,127</td>
<td>299,753</td>
<td>982.36</td>
<td>332.08</td>
</tr>
<tr>
<td>PubMed$^i$</td>
<td>141,043</td>
<td>737,869,085</td>
<td>8,200,000</td>
<td>5,231.52</td>
<td>89.98</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>189,583</td>
<td>1,418,129,813</td>
<td>3,825,305</td>
<td>7,480.26</td>
<td>370.72</td>
</tr>
</tbody>
</table>

Table 9.3: Experimental datasets and their statistics. V denotes vocabulary size, L denotes the number of training tokens, D denotes the number of documents, L/V indicates the average number of occurrences of a word, L/D indicates the average length of a document.

9.3.1 Computation Speed

We run all algorithms for a fixed number of iterations. In Figure 9.4, the samples produced per second by different algorithms over datasets of varying size is reported. We make the following observations: (1) In case of HDP, Alias HDP seems to be clear winner across all datasets. (2) However, there is no free lunch. The huge speed-up comes at the cost of increased memory usage (for storing the data-structures). However, looking at only computational speed paints an incomplete picture. The computationally fastest method may not converge as fast overall.
We make the following key observations: (1) At the end, almost all methods, except Light LDA, produce similar per-word log likelihoods, indicating good convergence of all methods. (2) In terms of convergence and per-word log likelihoods, there does not seem to be a difference between single threaded and multi threaded implementations. This implies that our mild assumptions made in Section 9.1.2 regarding negligible difference in global word-topic count \( c(x,k) \) between sampling a few tokens holds true. (3) Overall F-Nomad LDA seems to be converging fastest among all methods. (4) The computationally fastest sampler Light LDA, exhibits poor convergence as dataset grows larger and the discrepancy between its proposal and target becomes wider. Thus, Light LDA is not the fastest method overall. (5) Alias HDP consistently converges fastest and produce highest per-word Log likelihood among all methods.

Finally, we sweep LDA for different number of topics and compare against HDP. For this experiment, since many LDA needs to be trained, we reduced the datasize to 10k most frequently words for all datasets. HDP achieves a perplexity equivalent to the best LDA model as shown in Figure 9.10a, 9.10b, and 9.10c.
Figure 9.5: Single Threaded HDP on small datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number.
Figure 9.6: Multi Threaded HDP on small datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number.
Figure 9.7: Single Threaded HDP on medium datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number. While it was not observable previously due to small datasets, the difference between each method is clearer. In the comparison between time, Alias HDP was the fastest.
Figure 9.8: Multi Threaded HDP on medium datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number. While it was not observable previously due to small datasets, the difference between each method is clearer. In the comparison between time, SDA HDP was the fastest.
Figure 9.9: Multi Threaded HDP on large datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number. While it was not observable previously due to small datasets, the difference between each method is clearer. In the comparison between time, Alias HDP was the fastest.
9.3 Experimental Study

Comparison of Perplexity between LDA and HDP on a small, medium, and large dataset.

Figure 9.10: Comparison of perplexity between LDA and HDP on a small, medium, and large dataset.
Escrping Saddle Points

Till now we had focused on scaling up the inference for latent variable portion of a model involving terms of the form \( p(x) = \sum_z p(x|z)p(z) \), which is usually the main bottleneck. For learning a versatile representation the deep neural network is often needed to achieve high accuracy. Using stochastic gradient descent, the training for the deep network can be parallelized and carried out on heavily multi-cored modern compute infrastructures like GPUs. However, a central challenge to using first-order methods like stochastic gradient descent for optimizing nonconvex problems, which arises in case of deep networks, is the presence of saddle points. First-order methods often get stuck at saddle points, greatly deteriorating their performance. Typically, to escape from saddles one has to use second-order methods. However, most works on second-order methods rely extensively on expensive Hessian-based computations, making them impractical in large-scale settings. To tackle this challenge, we introduce a generic framework that minimizes Hessian-based computations while at the same time provably converging to second-order critical points. Our framework carefully alternates between a first-order and a second-order subroutine, using the latter only close to saddle points, and yields convergence results competitive to the state-of-the-art. Empirical results suggest that our strategy also enjoys a good practical performance.

10.1 Introduction

We study nonconvex finite-sum problems of the form

\[
\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x),
\]

where neither \( f : \mathbb{R}^d \to \mathbb{R} \) nor the individual functions \( f_i : \mathbb{R}^d \to \mathbb{R} \) \((i \in [n])\) are necessarily convex. We operate in a general nonconvex setting except for few smoothness assumptions like Lipschitz continuity of the gradient and Hessian. Optimization problems of this form arise naturally in machine learning and statistics as empirical risk minimization (ERM) and M-estimation respectively.

In the large-scale settings, algorithms based on first-order information of functions \( f_i \) are typically favored as they are relatively inexpensive and scale seamlessly. An algorithm widely used in practice is stochastic gradient descent (SGD), which has the iterative update:

\[
x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t),
\]

where \( i_t \in [n] \) is a randomly chosen index and \( \eta_t \) is a learning rate. Under suitable selection of the learning rate, we can show that SGD converges to a point \( x \) that, in expectation, satisfies the stationarity condition \( \| \nabla f(x) \| \leq \epsilon \) in \( O(1/\epsilon^4) \) iterations (Ghadimi and Lan, 2013). This result has two critical weaknesses: (i) It does not ensure convergence to local optima or second-order critical points; (ii) The rate of convergence of the SGD algorithm is slow.

For general nonconvex problems, one has to settle for a more modest goal than sub-optimality, as finding the global minimizer of finite-sum nonconvex problem will be in general intractably...
First order methods like GD can potentially get stuck at saddle points. Second-order methods can escape it in very few iterations (as observed in the left plot) but at the cost of expensive Hessian based iterations (see time plot to the right). The proposed framework, which is a novel mix of the two strategies, can escape saddle points faster in time by carefully trading off computation and iteration complexity.

Unfortunately, SGD does not even ensure second-order critical conditions such as local optimality since it can get stuck at saddle points. This issue has recently received considerable attention in the ML community, especially in the context of deep learning (Choromanska et al., 2015; Y. N. Dauphin et al., 2014; Y. Dauphin, Vries, and Y. Bengio, 2015). These works argue that saddle points are highly prevalent in most optimization paths, and are the primary obstacle for training large deep networks. To tackle this issue and achieve a second-order critical point for which $\|\nabla f\| \leq \epsilon$ and $\nabla^2 f \geq -\sqrt{\epsilon} I$, we need algorithms that either use the Hessian explicitly or exploit its structure.

A key work that explicitly uses Hessians to obtain faster convergence rates is the cubic regularization (CR) method (Nesterov and Polyak, 2006). In particular, Nesterov and Polyak (2006) showed that CR requires $O(1/\epsilon^{3/2})$ iterations to achieve the second-order critical conditions. However, each iteration of CR is expensive as it requires computing the Hessian and solving multiple linear systems, each of which has complexity $O(d^\omega)$ ($\omega$ is the matrix multiplication constant), thus, undermining the benefit of its faster convergence. Recently, N. Agarwal, Allen Zhu, et al. (2016) designed an algorithm to solve the CR more efficiently, however, it still exhibits slower convergence in practice compared to first-order methods. Both of these approaches use Hessian based optimization in each iteration, which make them slow in practice.

A second line of work focuses on using Hessian information (or its structure) whenever the method gets stuck at stationary points that are not second-order critical. To our knowledge, the first work in this line is Ge et al. (2015), which shows that for a class of functions that satisfy a special property called “strict-saddle” property, a noisy variant of SGD can converge to a point close to a local minimum. For this class of functions, points close to saddle points have a Hessian with a large negative eigenvalue, which proves instrumental in escaping saddle points using an isotropic noise. While such a noise-based method is appealing as it only uses first-order information, it has a very bad dependence on the dimension $d$, and furthermore, the result only holds when the strict-saddle property is satisfied (Ge et al., 2015). More recently, Carmon et al. (2018) presented a new faster algorithm that alternates between first-order and second-order subroutines. However, their algorithm is designed for the simple case of $n = 1$ in (10.1) and hence, can be expensive in practice.

Inspired by this line of work, we develop a general framework for finding second-order critical points. The key idea of our framework is to use first-order information for the most part of the optimization process and invoke Hessian information only when stuck at stationary points.
that are not second-order critical. We summarize the key idea and main contributions of this paper below.

**Main Contributions** We develop an algorithmic framework for converging to second-order critical points and provide convergence analysis for it. Our framework carefully alternates between two subroutines that use gradient and Hessian information, respectively, and ensures second-order criticality. Furthermore, we present two instantiations of our framework and provide convergence rates for them. In particular, we show that a simple instance of our framework, based on SVRG, achieves convergence rates competitive with the current state-of-the-art methods; thus highlighting the simplicity and applicability of our framework. Finally, we demonstrate the empirical performance of a few algorithms encapsulated by our framework and show their superior performance.

**Related Work** There is a vast literature on algorithms for solving optimization problems of the form (10.1). A classical approach for solving such optimization problems is SGD, which dates back at least to the seminal work of H. Robbins and S. Monro (1951). Since then, SGD has been a subject of extensive research, especially in the convex setting (Léon Bottou, 1991; Kushner and Clark, 2012; Ljung, 1977; Poljak and Tsypkin, 1973). Recently, new faster methods, called variance reduced (VR) methods, have been proposed for convex finite-sum problems. VR methods attain faster convergence by reducing the variance in the stochastic updates of SGD, c.f. A. J. Defazio, Caetano, and Domke (2014), A. Defazio, Bach, and Lacoste-Julien (2014), R. Johnson and T. Zhang (2013), Konečný et al. (2016), Mark Schmidt, Le Roux, and Bach (2017), and Shalev-Shwartz and T. Zhang (2013). Accelerated variants of these methods achieve the lower bounds proved in A. Agarwal and Leon Bottou (2015) and Lan and Zhou (2017), thereby settling the question of their optimality. Furthermore, S. Reddi et al. (2015) developed an asynchronous framework for VR methods and demonstrated their benefits in parallel environments.

Most of the aforementioned prior works study stochastic methods in convex or very specialized nonconvex settings that admit theoretical guarantees on sub-optimality. For the general nonconvex setting, it is only recently that non-asymptotic convergence rate analysis for SGD and its variants was obtained by Ghadimi and Lan (2013), who showed that SGD ensures \(|\|\nabla f\|\leq \epsilon|\) in expectation in \(O(1/\epsilon^4)\) iterations. A similar rate for parallel and distributed SGD was shown in Lian et al. (2015). For these problems, Sashank J. Reddi, Hefny, et al. (2016), Sashank J. Reddi, Sra, et al. (2016), and Sashank J Reddi et al. (2016) proved faster convergence rates that ensure the same optimality criteria in \(O(n + n^{2/3}/\epsilon^2)\), which is an order \(n^{1/3}\) faster than GD. While these methods ensure convergence to stationary points at a faster rate, the question of convergence to local minima (or in general to second-order critical points) has not been addressed. To our knowledge, convergence rates to second-order critical points (defined in Definition 10.1) for general nonconvex functions was first studied by Nesterov and Polyak (2006). However, each iteration of the algorithm in Nesterov and Polyak (2006) is prohibitively expensive since it requires eigenvalue decompositions, and hence, is unsuitable for large-scale high-dimensional problems. More recently, N. Agarwal, Allen Zhu, et al. (2016) and Carmon et al. (2018) presented algorithms for finding second-order critical points by tackling some practical issues that arise in Nesterov and Polyak (2006). However, these algorithms are either only applicable to a restricted setting or heavily use Hessian based computations, making them unappealing from a practical standpoint. Also, recently Carmon et al. (2017) proposed a fast accelerated first-order method for nonconvex optimization; however, similar to Carmon et al. (2018), it requires computation of the full gradient at each iteration and only provides convergence to stationary points. Noisy variants of first-order methods have also been shown to escape saddle points (Ge et al., 2015; Jin et al., 2017; K. Y. Levy, 2016), however, they have strong dependence on either \(n\) or \(d\), both of which are undesirable.
10.2 BACKGROUND & PROBLEM SETUP

We assume that each of the functions $f_i$ in (10.1) is $L$-smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for all $i \in [n]$. Furthermore, we assume that the Hessian of $f$ in (10.1) is Lipschitz, i.e., we have
\[
\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq M\|x - y\|,
\]
for all $x, y \in \mathbb{R}^d$. Such a condition is typically necessary to ensure convergence of algorithms to the second-order critical points (Nesterov and Polyak, 2006). In addition to the above smoothness conditions, we also assume that the function $f$ is bounded below, i.e., $f(x) \geq B$ for all $x \in \mathbb{R}^d$.

In order to measure stationarity of an iterate $x$, similar to Ghadimi and Lan (2013), Nesterov (2004), and Nesterov and Polyak (2006), we use the condition $\|\nabla f(x)\| \leq \epsilon$. In this paper, we are interested in convergence to second-order critical points. Thus, in addition to stationarity, we also require the solution to satisfy the Hessian condition $\nabla^2 f(x) \succeq -\gamma I$ (Nesterov and Polyak, 2006). For iterative algorithms, we require both $\epsilon, \gamma \to 0$ as the number of iterations $T \to \infty$. When all saddle points are non-degenerate, such a condition implies convergence to a local optimum.

**Definition 10.1** An algorithm $A$ is said to obtain a point $x$ that is a $(\epsilon, \gamma)$-second order critical point if $\mathbb{E}[\|\nabla f(x)\|] \leq \epsilon$ and $\nabla^2 f(x) \succeq -\gamma I$, where the expectation is over any randomness in $A$.

We must exercise caution while interpreting results pertaining to $(\epsilon, \gamma)$-second order critical points. Such points need not be close to any local minima either in objective function value, or in the domain of (10.1). For our algorithms, we use only an Incremental First-order Oracle (IFO) (A. Agarwal and Leon Bottou, 2015) and an Incremental Second-order Oracle (ISO), defined below.

**Definition 10.2** An IFO takes an index $i \in [n]$ and a point $x \in \mathbb{R}^d$, and returns the pair $(f_i(x), \nabla f_i(x))$. An ISO takes an index $i \in [n]$, point $x \in \mathbb{R}^d$ and vector $v \in \mathbb{R}^d$ and returns the vector $\nabla^2 f_i(x)v$.

IFO and ISO calls are typically cheap, with ISO call being relatively more expensive. In many practical settings that arise in machine learning, the time complexity of these oracle calls is linear in $d$ (N. Agarwal, Bullins, and Hazan, 2016; Pearlmutter, 1994). For clarity and clean comparison, the dependence of time complexity on Lipschitz constant $L$, $M$, initial point and any polylog factors in the results is hidden.

10.3 GENERIC FRAMEWORK

In this section, we propose a generic framework for escaping saddle points while solving non-convex problems of form (10.1). One of the primary difficulties in reaching a second-order critical point is the presence of saddle points. To evade such points, one needs to use properties of both gradients and Hessians. To this end, our framework is based on two core subroutines: **Gradient-Focused-Optimizer** and **Hessian-Focused-Optimizer**.

The idea is to use these two subroutines, each focused on different aspects of the optimization procedure. **Gradient-Focused-Optimizer** focuses on using gradient information for decreasing the function. On its own, the **Gradient-Focused-Optimizer** might not converge to
Algorithm 3 Generic Framework

1: **Input** - Initial point: $x^0$, total iterations $T$, error threshold parameters $\epsilon, \gamma$ and probability $p$
2: for $t = 1$ to $T$ do
3:   $(y^t, z^t) = \text{Gradient-Focused-Optimizer}(x^{t-1}, \epsilon)$ (refer to $G.1$ and $G.2$)
4:   Choose $u^t$ as $y^t$ with probability $p$ and $z^t$ with probability $1 - p$
5:   $(x^{t+1}, y^{t+1}) = \text{Hessian-Focused-Optimizer}(u^t, \epsilon, \gamma)$ (refer to $H.1$ and $H.2$)
6:   if $\tau^{t+1} = 0$ then
7:       Output set $\{x^{t+1}\}$
8:   end if
9: end for
10: **Output** set $\{y^1, ..., y^T\}$

A local minimizer since it can get stuck at a saddle point. Hence, we require the subroutine $\text{Hessian-Focused-Optimizer}$ to help avoid saddle points. A natural idea is to interleave these subroutines to obtain a second-order critical point. But it is not even clear if such a procedure even converges. We propose a carefully designed procedure that effectively balances these two subroutines, which not only provides meaningful theoretical guarantees, but also translates into strong empirical gains in practice.

Algorithm 3 provides pseudocode of our framework. Observe that the presented algorithm is still abstract, since it does not specify the subroutines $\text{Gradient-Focused-Optimizer}$ and $\text{Hessian-Focused-Optimizer}$. These subroutines determine the crucial update mechanism of the algorithm. We will present specific instance of these subroutines in the next section, but we assume the following properties to hold for these subroutines:

- **Gradient-Focused-Optimizer**: Suppose $(y, z) = \text{Gradient-Focused-Optimizer}(x, n, \epsilon)$, then there exists positive function $g : N \times R^+ \to R^+$, such that
  
  $G.1$ $E[f(y)] \leq f(x)$,
  
  $G.2$ $E[\|\nabla f(y)\|^2] \leq \frac{1}{g(n, \epsilon)} E[f(x) - f(z)]$.

  Here the outputs $y, z \in R^d$. The expectation in the conditions above is over any randomness that is a part of the subroutine. The function $g$ will be critical for the overall rate of Algorithm 3. Typically, Gradient-Focused-Optimizer is a first-order method, since the primary aim of this subroutine is to focus on gradient based optimization.

- **Hessian-Focused-Optimizer**: Suppose $(y, \tau) = \text{Hessian-Focused-Optimizer}(x, n, \epsilon, \gamma)$ where $y \in R^d$ and $\tau \in \{0, \gamma\}$. If $\tau = 0$, then $y$ is a $(\epsilon, \gamma)$-second order critical point with probability at least $1 - q$. Otherwise if $\tau = \gamma$, then $y$ satisfies the following condition:

  $H.1$ $E[f(y)] \leq f(x)$,
  
  $H.2$ $E[f(y)] \leq f(x) - h(n, \epsilon, \gamma)$ when $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$ for some function $h : N \times R^+ \times R^+ \to R^+$.

  Here the expectation is over any randomness in subroutine Hessian-Focused-Optimizer. The two conditions ensure that the objective function value, in expectation, never increases and furthermore, decreases with a certain rate when $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$. In general, this subroutine utilizes the Hessian or its properties for minimizing the objective function. Typically, this is the most expensive part of the Algorithm 3 and hence, needs to be invoked judiciously.
The key aspect of these subroutines is that they, in expectation, never increase the objective function value. The functions \( g \) and \( h \) will determine the convergence rate of Algorithm 3. In order to provide a concrete implementation, we need to specify the aforementioned subroutines. Before we delve into those details, we will provide a generic convergence analysis for Algorithm 3.

### 10.3.1 Convergence Analysis

**Theorem 10.3** Let \( \Delta = f(x^0) - B \), and \( \theta = \min([1 - p)\epsilon^2 g(n, \epsilon), ph(n, \epsilon, \gamma)] \). Also, let \( \Gamma \) be the set output by Algorithm 3 with Gradient-Focused-Optimizer satisfying \( G.1 \) and \( G.2 \) and Hessian-Focused-Optimizer satisfying \( H.1 \) and \( H.2 \). Further, let \( T \) be such that \( T > \Delta/\theta \).

Suppose the multiset \( S = \{1, ..., k\} \) contains \( k \) indices selected independently and uniformly from \( \{1, ..., |\Gamma|\} \). Then the following holds for the indices in \( S \):

1. \( y^t \) (where \( t \in S \)) is an \((\epsilon, \gamma)\)-critical point with probability at least \( 1 - \max(\Delta/(T\theta), q) \).
2. If \( k = O(\log(1/\epsilon)/\min(\log(1/(T\theta)), \log(1/q))) \), with at least probability \( 1 - \zeta \), at least one iterate \( y^t \) where \( t \in S \) is an \((\epsilon, \gamma)\)-critical point.

**Proof** The case of \( \tau = 0 \) can be handled in a straightforward manner, so let us focus on the case where \( \tau = \alpha \). We split our analysis into cases, each analyzing the change in objective function value depending on second-order criticality of \( y^t \).

We start with the case where the gradient condition of second-order critical point is violated and then proceed to the case where the Hessian condition is violated.

**Case 1**: \( \mathbb{E}[\|\nabla f(y^t)\|] \geq \epsilon \) for some \( t > 0 \)

We first observe the following: \( \mathbb{E}[\|\nabla f(y^t)\|^2] \geq (\mathbb{E}[\|\nabla f(y^t)\|])^2 \geq \epsilon^2 \). This follows from a straightforward application of Jensen’s inequality. From this inequality, we have the following:

\[
e^2 \leq \mathbb{E}[\|\nabla f(y^t)\|^2] \leq \frac{1}{g(n, \epsilon)} \mathbb{E}[f(x^{t-1}) - f(z^t)].
\]  

(10.4)

This follows from the fact that \( y^t \) is the output of Gradient-Focused-Optimizer subroutine, which satisfies the condition that for \( (y, z) = \text{Gradient-Focused-Optimizer}(x, n, \epsilon) \), we have

\[
\mathbb{E}[\|\nabla f(y)\|^2] \leq \frac{1}{g(n, \epsilon)} \mathbb{E}[f(x) - f(z)].
\]

From Equation (10.4), we have

\[
\mathbb{E}[f(z^t)] \leq \mathbb{E}[f(x^{t-1})] - e^2 g(n, \epsilon).
\]

Furthermore, due to the property of non-increasing nature of Gradient-Focused-Optimizer, we also have \( \mathbb{E}[y^t] \leq \mathbb{E}[f(x^{t-1})] \).

We now focus on the Hessian-Focused-Optimizer subroutine. From the property of Hessian-Focused-Optimizer that the objective function value is non-increasing, we have \( \mathbb{E}[f(x^t)] \leq \mathbb{E}[f(u^t)] \). Therefore, combining with the above inequality, we have

\[
\mathbb{E}[f(x^t)] \leq \mathbb{E}[f(u^t)]
\]

\[
= p\mathbb{E}[f(y^t)] + (1 - p)\mathbb{E}[f(z^t)]
\]

\[
\leq p\mathbb{E}[f(x^{t-1})] + (1 - p)\mathbb{E}[f(x^{t-1})] - e^2 g(n, \epsilon)
\]

\[
= \mathbb{E}[f(x^{t-1})] - (1 - p)e^2 g(n, \epsilon).
\]  

(10.5)
The first equality is due to the definition of $u^t$ in Algorithm 3. Therefore, when the gradient condition is violated, irrespective of whether $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$ or $\nabla^2 f(y^t) \geq -\gamma I$, the objective function value always decreases by at least $\epsilon^2 g(n, \epsilon)$.

**Case II** : $\mathbb{E}[\|\nabla f(y^t)\|] < \epsilon$ and $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$ for some $t > 0$

In this case, we first note that for $y = \text{Hessian-Focused-Optimizer}(x, n, \epsilon, \gamma)$ and $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$, we have $\mathbb{E}[f(y^t)] \leq f(x) - h(n, \epsilon, \gamma)$. Observe that $x^t = \text{Hessian-Focused-Optimizer}(u^t, n, \epsilon, \gamma)$. Therefore, if $u^t = y^t$ and $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$, then we have

$$
\mathbb{E}[f(x^t)|u^t = y^t] \leq f(y^t) - h(n, \epsilon, \gamma) \leq f(x^{t-1}) - h(n, \epsilon, \gamma).
$$

The second inequality is due to the non-increasing property of $\text{Gradient-Focused-Optimizer}$. On the other hand, if $u^t = z^t$, we have hand, if we have $\mathbb{E}[f(x^t)|u^t = z^t] \leq f(z^t)$. This is due to the non-increasing property of $\text{Hessian-Focused-Optimizer}$. Combining the above two inequalities and using the law of total expectation, we get

$$
\mathbb{E}[f(x^t)] = p\mathbb{E}[f(x^t)|u^t = y^t] + (1 - p)\mathbb{E}[f(x^t)|u^t = z^t]
\leq p (\mathbb{E}[f(y^t)] - h(n, \epsilon, \gamma)) + (1 - p)\mathbb{E}[f(z^t)]
\leq p (\mathbb{E}[f(x^{t-1})] - h(n, \epsilon, \gamma)) + (1 - p)\mathbb{E}[f(x^{t-1})]
= \mathbb{E}[f(x^{t-1})] - ph(n, \epsilon, \gamma).
$$

(10.6)

The second inequality is due to the non-increasing property of $\text{Gradient-Focused-Optimizer}$. Therefore, when the hessian condition is violated, the objective function value always decreases by at least $ph(n, \epsilon, \gamma)$.

**Case III** : $\mathbb{E}[\|\nabla f(y^t)\|] < \epsilon$ and $\nabla^2 f(y^t) \geq -\gamma I$ for some $t > 0$

This is the favorable case for the algorithm. The only condition to note is that the objective function value will be non-increasing in this case too. This is, again, due to the non-increasing properties of subroutines $\text{Gradient-Focused-Optimizer}$ and $\text{Hessian-Focused-Optimizer}$. In general, greater the occurrence of this case during the course of the algorithm, higher will the probability that the output of our algorithm satisfies the desired property.

The key observation is that Case I & II cannot occur large number of times since each of these cases strictly decreases the objective function value. In particular, from Equation (10.5) and (10.6), it is easy to see that each occurrence of Case I & II the following holds:

$$
\mathbb{E}[f(x^t)] \leq \mathbb{E}[f(x^{t-1})] - \theta,
$$

where $\theta = \min((1 - p)e^2 g(n, \epsilon), ph(n, \epsilon, \gamma))$. Furthermore, the function $f$ is lower bounded by $B$, thus, Case I & II cannot occur more than $(f(x^0) - B)/\theta$ times. Therefore, the probability of occurrence of Case III is at least $1 - (f(x^0) - B)/(T\theta)$, which completes the first part of the proof.

The second part of the proof simply follows from first part. As seen above, the probability of Case I & II is at most $(f(x^0) - B)/T\theta$. Therefore, probability that an element of the set $S$ falls in Case III is at least $1 - ((f(x^0) - B)/T\theta)^k$, which gives us the required result for the second part.

The key point regarding the above result is that the overall convergence rate depends on the magnitude of both functions $g$ and $h$. Theorem 10.3 shows that the slowest amongst the subroutines $\text{Gradient-Focused-Optimizer}$ and $\text{Hessian-Focused-Optimizer}$ governs the overall
We now present specific instantiations of our framework in this section. Before we state our key results, we discuss an important subroutine that is used as a key component of our framework: SVRG. We give a brief description of the algorithm in this section and show that it meets the conditions required for a Gradient-Focused-Optimizer. SVRG (R. Johnson and T. Zhang, 2013; Sashank J. Reddi, Hefny, et al., 2016) is a stochastic algorithm recently shown to be very effective for reducing variance in finite-sum problems. We seek to understand its benefits for nonconvex optimization, with a particular focus on the issue of escaping saddle points. Algorithm 4 presents SVRG’s pseudocode.

Observe that Algorithm 4 is an epoch-based algorithm. At the start of each epoch $s$, a full gradient is calculated at the point $\tilde{x}^s$, requiring $n$ calls to the IFO. Within its inner loop SVRG performs $m$ stochastic updates. Suppose $m$ is chosen to be $O(n)$ (typically used in practice), then the total IFO calls per epoch is $\Theta(n)$. Strong convergence rates have been proved Algorithm 4 in the context of convex and nonconvex optimization (R. Johnson and T. Zhang, 2013; Sashank J. Reddi, Hefny, et al., 2016). The following result shows that SVRG meets the requirements of a Gradient-Focused-Optimizer.

**Lemma 10.4** Suppose $\eta_t = \eta = 1/4Ln^{2/3}$, $m = n$ and $T_g = T_{\epsilon}$, which depends on $\epsilon$, then Algorithm 4 is a Gradient-Focused-Optimizer with $g(n, \epsilon) = T_{\epsilon}/40Ln^{2/3}$.

**Proof** The proof follows from the analysis in Sashank J. Reddi, Hefny, et al. (2016) with some additional reasoning. We need to show two properties: G.1 and G.2, both of which are based on objective function value. To this end, we start with an update in the $s^{th}$ epoch. We have the following:

$$
\mathbb{E}[f(x^{s+1}_t)] \leq \mathbb{E}[f(x^{s+1}_t)] + \langle \nabla f(x^{s+1}_t), x^{s+1}_t - x^{s+1}_t \rangle + \frac{1}{2} \|x^{s+1}_t - x^{s+1}_t\|^2 \\
\leq \mathbb{E}[f(x^{s+1}_t)] - \eta_t \|\nabla f(x^{s+1}_t)\|^2 + \frac{Ln^2}{2} \|x^{s+1}_t - x^{s+1}_t\|^2.
$$

(10.7)
The first inequality is due to $L$-smoothness of the function $f$. The second inequality simply follows from the unbiasedness of SVRG update in Algorithm 4. For the analysis of the algorithm, we need the following Lyapunov function:

$$A_t^{s+1} := \mathbb{E}[f(x_t^{s+1}) + \mu_t ||x_t^{s+1} - \bar{x}^s||^2].$$

This function is a combination of objective function and the distance of the current iterate from the latest snapshot $\bar{x}_s$. Note that the term $\mu_t$ is introduced only for the analysis and is not part of the algorithm (see Algorithm 4). Here $(\mu_t)_{t=0}^m$ is chosen such the following holds:

$$\mu_t = \mu_{t+1}(1 + \eta_t \beta_t + 2\eta_t^2 L^2) + \eta_t^2 L^3,$$

for all $t \in \{0, \cdots, m-1\}$ and $\mu_m = 0$. For bounding the Lyapunov function $A$, we need the following bound on the distance of the current iterate from the latest snapshot:

$$\mathbb{E}[||x_t^{s+1} - \bar{x}^s||^2] = \mathbb{E}[||x_t^{s+1} - x_{t-1}^{s+1} + x_{t-1}^{s+1} - \bar{x}^s||^2]$$

$$\quad = \mathbb{E}[||x_t^{s+1} - x_{t-1}^{s+1}||^2 + ||x_{t-1}^{s+1} - \bar{x}^s||^2 + 2(x_{t-1}^{s+1} - x_{t-1}^{s+1}, x_{t-1}^{s+1} - \bar{x}^s)]$$

$$\quad = \mathbb{E}[\eta_t^2 ||v_{t-1}^{s+1}||^2 + ||x_{t-1}^{s+1} - \bar{x}^s||^2] - 2\eta_t \mathbb{E}[\langle \nabla f(x_{t-1}^{s+1}), x_{t-1}^{s+1} - \bar{x}^s \rangle]$$

$$\quad \leq \mathbb{E}[\eta_t^2 ||v_{t-1}^{s+1}||^2 + ||x_{t-1}^{s+1} - \bar{x}^s||^2] + 2\eta_t \mathbb{E}\left[\frac{1}{2\beta_t} ||\nabla f(x_{t-1}^{s+1})||^2 + \frac{1}{2}\beta_t ||x_{t-1}^{s+1} - \bar{x}^s||^2\right].$$  \hspace{1cm} (10.8)

The second equality follows from a simple application of Cauchy-Schwarz and Young’s inequality. Substituting Equation (10.7) and Equation (10.8) into the Lyapunov function $A_{t+1}^{s+1}$, we obtain the following:

$$A_{t+1}^{s+1} \leq \mathbb{E}[f(x_t^{s+1}) - \eta_t ||\nabla f(x_t^{s+1})||^2 + \frac{L\eta_t^2}{2} ||v_{t-1}^{s+1}||^2]$$

$$+ \mathbb{E}[\mu_t \eta_t^2 ||v_{t-1}^{s+1}||^2 + \mu_t ||x_{t-1}^{s+1} - \bar{x}^s||^2]$$

$$+ 2\eta_t \mu_t \mathbb{E}\left[\frac{1}{2\beta_t} ||\nabla f(x_{t-1}^{s+1})||^2 + \frac{1}{2}\beta_t ||x_{t-1}^{s+1} - \bar{x}^s||^2\right]$$

$$\leq \mathbb{E}[f(x_t^{s+1}) - \left(\eta_t - \frac{\mu_t + \eta_t^2}{\beta_t}\right) ||\nabla f(x_t^{s+1})||^2]$$

$$+ \frac{2\beta_t}{\mu_t} \mathbb{E}[||v_{t-1}^{s+1}||^2] + (\mu_t + \mu_t + \eta_t^2) \mathbb{E}\left[||x_{t-1}^{s+1} - \bar{x}^s||^2\right].$$  \hspace{1cm} (10.9)

To further bound this quantity, we need the bound on the variance of SVRG:

**Proposition 10.5** (Sashank J. Reddi, Hefny, et al., 2016) Let $v_{t-1}^{s+1}$ be computed by Algorithm 4. Then,

$$\mathbb{E}[||v_{t-1}^{s+1}||^2] \leq 2\mathbb{E}[||\nabla f(x_t^{s+1})||^2] + 2L^2 \mathbb{E}[||x_{t-1}^{s+1} - \bar{x}^s||^2].$$

We can use the above Proposition 10.5 to bound $\mathbb{E}[||v_{t-1}^{s+1}||^2]$, so that upon substituting it in Equation (10.9), we see that

$$\begin{align*}
A_{t+1}^{s+1} &\leq \mathbb{E}[f(x_t^{s+1})] - \left(\eta_t - \frac{\mu_t + \eta_t^2}{\beta_t}\right) \mathbb{E}[||\nabla f(x_t^{s+1})||^2] \\
&\quad + \left[\mu_t + (1 + \eta_t^2 L^2) + \eta_t^2 L^3\right] \mathbb{E}\left[||x_{t-1}^{s+1} - \bar{x}^s||^2\right] \\
&\leq A_t^{s+1} - \left(\eta_t - \frac{\mu_t + \eta_t^2}{\beta_t}\right) \mathbb{E}[||\nabla f(x_t^{s+1})||^2].
\end{align*}$$

The second inequality follows from the definition of $\mu_t$ and $A_t^{s+1}$. Since $\eta_t = \eta = 1/(4Ln^{2/3})$ for $j > 0$ and $t \in \{0, \cdots, j-1\}$,

$$A_t^{s+1} \leq A_0^{s+1} - \sum_{t=0}^{j-1} \mathbb{E}[||\nabla f(x_t^{s+1})||^2],$$  \hspace{1cm} (10.10)
We will prove that for the given parameter setting $\nu_n > 0$ (see the proof below). With $\nu_n > 0$, it is easy to see that $A^{s+1}_0 \leq A^{s+1}_0$. Furthermore, note that $A^{s+1}_0 = \mathbb{E}[f(x^{s+1})] + \mu_0\|x^{s+1}_0 - \bar{x}\|^2 = \mathbb{E}[f(x^{s+1})]$ since $x^{s+1}_0 = \bar{x}$ (see Algorithm 4). Also, we have

$$\mathbb{E}[f(x^{s+1}) + \mu_j\|x^{s+1}_j - \bar{x}\|^2] \leq \mathbb{E}[f(x^{s+1})]$$

and thus, we obtain $\mathbb{E}[f(x^{s+1})] \leq \mathbb{E}[f(x^{s+1})]$ for all $j \in \{0, ..., m\}$. Furthermore, using simple induction and the fact that $x^{s+1}_0 = x^*_m$ for all epoch $s \in \{0, ..., S - 1\}$, it easy to see that $\mathbb{E}[f(x^{s+1}_j)] \leq f(x^0)$. Therefore, with the definition of $\nu$ specified in the output of Algorithm 4, we see that the condition G.1 of GRADIENT-FOCUSED-OPTIMIZER is satisfied for SVRG algorithm.

We now prove that $\nu_n > 0$ and also G.2 of GRADIENT-FOCUSED-OPTIMIZER is satisfied for SVRG algorithm. By using telescoping the sum with $j = m$ in Equation (10.10), we obtain

$$\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x^{s+1}_t)\|^2] \leq \frac{A^{s+1}_0 - A^{s+1}_m}{\nu_n}. \quad (10.11)$$

This inequality in turn implies that

$$\frac{1}{t_0} \sum_{s=0}^{S-1} \sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x^{s+1}_t)\|^2] \leq \frac{\mathbb{E}[f(x^0) - f(x^*_m)]}{t_0 \nu_n}. \quad (10.12)$$

Here we used the the fact that $x^0 = x^0$. To obtain a handle on $\nu_n$ and complete our analysis, we will require an upper bound on $\nu_0$. We observe that $\nu_0 = \frac{L}{18n^{4/3}} \frac{(1 + \theta)^m - 1}{\theta}$ where $\theta = 2\eta^2L^2 + \eta\beta$. This is obtained using the relation $\nu_t = \mu_{t+1}(1 + \eta\beta + 2\eta^2L^2) + \eta^2L^3$ and the fact that $\mu_m = 0$. Using the specified values of $\beta$ and $\eta$ we have

$$\theta = 2\eta^2L^2 + \eta\beta = \frac{1}{8n^{4/3}} + \frac{1}{4n} \leq \frac{3}{4n}. \quad (10.13)$$

Using the above bound on $\theta$, we get

$$\mu_0 = \frac{L}{18n^{4/3}} \frac{(1 + \theta)^m - 1}{\theta} = \frac{L((1 + \theta)^m - 1)}{2(1 + 2n^{1/3})} \leq \frac{L((1 + \frac{3}{4n})^{4n/3} - 1)}{2(1 + 2n^{1/3})} \leq n^{-1/3}(L(e - 1)/4),$$

wherein the second inequality follows upon noting that $(1 + \frac{1}{e})^l$ is increasing for $l > 0$ and $\lim_{t \to \infty}(1 + \frac{1}{e})^l = e$ (here $e$ is the Euler’s number). Now we can lower bound $\nu_n$, as

$$\nu_n = \min_{l} (\eta - \frac{\mu_{l+1}\eta}{B} - \eta^2L - 2\mu_{l+1}\eta^2) \geq \eta - \frac{\mu_{l+1}\eta}{B} - \eta^2L - 2\mu_{l+1}\eta^2 \geq \frac{1}{40Ln^{2/3}}.$$

The first inequality holds since \( \mu_t \) decreases with \( t \). The second inequality holds since (a) \( \mu_0/\beta \) can be upper bounded by \((e-1)/4 \) (follows from Equation (10.13)), (b) \( \eta^2 L \leq \eta/4 \) and (c) \( 2\mu_0 \eta^2 \leq (e-1)\eta/8 \) (follows from Equation (10.13)). Substituting the above lower bound in Equation (10.12), we obtain the following:

\[
\frac{1}{T_g} \sum_{s=0}^{S-1} \sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_s^{t+1})\|^2] \leq \frac{40Ln^{2/3}E[f(x^0) - f(x_m^S)]}{T_g}.
\] (10.14)

From the definition of \((y, z)\) in output of Algorithm 4 i.e., \( y \) is Iterate \( x_m \) chosen uniformly random from \( \{x_t^{S}\} \) and \( z = x_m^S \), it is clear that Algorithm 4 satisfies the G.2 requirement of GRADIENT-FOCUSED-OPTIMIZER with \( g(n, \epsilon) = T_\epsilon/40Ln^{2/3} \). Since both G.1 and G.2 are satisfied for Algorithm 4, we conclude that SVRG is a GRADIENT-FOCUSED-OPTIMIZER.

In the rest of this section, we discuss approaches using SVRG as a GRADIENT-FOCUSED-OPTIMIZER. In particular, we propose and provide convergence analysis for two different methods with different HESSIAN-FOCUSED-OPTIMIZER but which use SVRG as a GRADIENT-FOCUSED-OPTIMIZER.

### 10.4.1 Hessian descent

The first approach is based on directly using the eigenvector corresponding to the smallest eigenvalue as a HESSIAN-FOCUSED-OPTIMIZER. More specifically, when the smallest eigenvalue of the Hessian is negative and reasonably large in magnitude, the Hessian information can be used to ensure descent in the objective function value. The pseudo-code for the algorithm is given in Algorithm 5.

The key idea is to utilize the minimum eigenvalue information in order to make a descent step. If \( \lambda_{\min}(\nabla^2 f(x)) \leq -\gamma \) then the idea is to use this information to take a descent step. Note the subroutine is designed in a fashion such that the objective function value never increases. Thus, it naturally satisfies the requirement H.1 of HESSIAN-FOCUSED-OPTIMIZER. The following result shows that HessianDescent is a HESSIAN-FOCUSED-OPTIMIZER.

**Lemma 10.6** HessianDescent is a HESSIAN-FOCUSED-OPTIMIZER with \( h(n, \epsilon, \gamma) = \frac{\beta}{24M^2} \gamma^3 \).

**Proof** The first important observation is that the function value never increases because \( y = \arg\min_{z \in [u, x]} f(z) \) i.e., \( f(y) \leq f(x) \), thus satisfying H.1 of HESSIAN-FOCUSED-OPTIMIZER. We now analyze the scenario where \( \lambda_{\min}(\nabla^2 f(x)) \leq -\gamma \). Consider the event where we obtain \( v \) such that

\[
\langle v, \nabla^2 f(x)v \rangle \leq \lambda_{\min}(\nabla^2 f(x)) + \frac{\gamma}{2}.
\]

This event (denoted by \( \mathcal{E} \)) happens with at least probability \( \rho \). Note that, since \( \lambda_{\min}(\nabla^2 f(x)) \leq -\gamma \), we have \( \langle v, \nabla^2 f(x)v \rangle \leq -\frac{\gamma}{2} \). In this case, we have the following relationship:

\[
f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x) + \frac{M}{6} ||y - x||^3
\]

\[
= f(x) - \alpha \langle \nabla f(x), v \rangle + \frac{\alpha^2}{2} v^T \nabla^2 f(x)v + \frac{M \alpha^3}{6} \|v\|^3
\]

\[
\leq f(x) + \frac{\alpha^2}{2} v^T \nabla^2 f(x)v + \frac{M \alpha^3}{6}
\]

\[
\leq f(x) - \frac{1}{2M^2} h^T \nabla^2 f(x)h^3 + \frac{1}{6M^2} h^T \nabla^2 f(x)h^3
\]

\[
= f(x) - \frac{1}{2M^2} h^T \nabla^2 f(x)h^3 \leq f(x) - \frac{1}{24M^2} \gamma^3.
\] (10.15)
The first inequality follows from the M-lipschitz continuity of the Hessain \( \nabla^2 f(x) \). The first equality follows from the update rule of HessianDescent. The second inequality is obtained by dropping the negative term and using the fact that \( \|v\| = 1 \). The second equality is obtained by substituting \( \alpha = \frac{\|v\| \nabla^2 f(x) v\|}{M} \). The last inequality is due to the fact that \( \langle v, \nabla^2 f(x) v \rangle \leq -\frac{\gamma}{2} \). In the other scenario where

\[
\langle v, \nabla^2 f(x) v \rangle \leq \lambda_{\min}(\nabla^2 f(x)) + \frac{\gamma}{2},
\]

we can at least ensure that \( f(y) \leq f(x) \) since \( y = \arg \min_{z \in \{u, x\}} f(z) \). Therefore, we have

\[
\mathbb{E}[f(y)] = \rho \mathbb{E}[f(y)] + (1 - \rho) \mathbb{E}[f(y)]
\leq \rho \mathbb{E}[f(y)] + (1 - \rho) f(x)
\leq \rho \left[ f(x) - \frac{\rho}{24M^2} \gamma^3 \right] + (1 - \rho) f(x)
= f(x) - \frac{\rho}{24M^2} \gamma^3.
\]

(10.16)

The last inequality is due to Equation (10.15). Hence, Hessian-Focused-Optimizer satisfies H.2 of Hessian-Focused-Optimizer with \( h(n, \epsilon, \gamma) = \frac{\rho}{24M^2} \gamma^3 \), thus concluding the proof.

With SVRG as Gradient-Focused-Optimizer and HessianDescent as Hessian-Focused-Optimizer, we show the following key result:

**Theorem 10.7** Suppose SVRG with \( m = n \), \( \eta_t = \eta = 1/4\ln^{2/3} \) for all \( t \in \{1, \ldots, m\} \) and \( T_g = 40\ln^{2/3}/\epsilon^{1/2} \) is used as Gradient-Focused-Optimizer and HessianDescent is used as Hessian-Focused-Optimizer with \( q = 0 \), then Algorithm 3 finds a \((\epsilon, \sqrt{\epsilon})\)-second order critical point in \( T = O(\Delta/\min(p, 1 - p) \epsilon^{3/2}) \) with probability at least 0.9.

The result directly follows from using Lemma 10.4 and Lemma 10.6 in Theorem 10.3. The result shows that the iteration complexity of Algorithm 3 in this case is \( O(\Delta/\epsilon^{3/2} \min(p, 1 - p)) \). Thus, the overall IFO complexity of SVRG algorithm is \( (n + T_g) \times T = O(n/\epsilon^{3/2} + n^{2/3}/\epsilon^{2/3}) \). Since each IFO call takes \( O(d) \) time, the overall time complexity of all Gradient-Focused-Optimizer steps is \( O(nd/\epsilon^{3/2} + n^{2/3}d/\epsilon^{2/2}) \). To understand the time complexity of HessianDescent, we need the following result due to N. Agarwal, Allen Zhu, et al. (2016).

**Proposition 10.8** (N. Agarwal, Allen Zhu, et al., 2016) The time complexity of finding \( v \in \mathbb{R}^d \) that \( \|v\| = 1 \), and with probability at least \( \rho \) the following inequality holds: \( \langle v, \nabla^2 f(x) v \rangle \leq \lambda_{\min}(\nabla^2 f(x)) + \frac{\gamma}{2} \) is \( O(nd + n^{3/4}d/\epsilon^{1/2}) \).

Note that each iteration of Algorithm 3 in this case has just linear dependence on \( d \). Since the total number of HessianDescent iterations is \( O(\Delta/\min(p, 1 - p) \epsilon^{3/2}) \) and each iteration has the complexity of \( O(nd + n^{3/4}d/\epsilon^{1/4}) \), using the above remark, we obtain an overall time
complexity of HessianDescent is $O(\|n\|d/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4})$. Combining this with the time complexity of Svrg, we get the following result.

**Corollary 10.9** The overall running time of Algorithm 3 to find a $(\epsilon, \sqrt{\epsilon})$-second order critical point, with parameter settings used in Algorithm 10.7, is $O(\|n\|d/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4} + n^{2/3}d/\epsilon^{2})$.

Note that the dependence on $\epsilon$ is much better in comparison to that of Noisy SGD used in Ge et al. (2015). Furthermore, our results are competitive with N. Agarwal, Allen Zhu, et al. (2016) and Carmon et al. (2018) in their respective settings, but with a much more practical algorithm and simpler analysis. More specifically, N. Agarwal, Allen Zhu, et al. (2016) has running time of $O(\|n\|d/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4})$. When $\epsilon > c/n^{2/3}$ where $c$ is some constant (which is very reasonable for most machine learning settings), our results would be better. However, our rates can be slightly worse when this is not the case. Next, Carmon et al. (2018) does not consider the finite-sum setting specifically, and if applied as it is for our finite-sum setup, it will have a running time of $O(\|n\|d/\epsilon^{7/4})$. In comparison, our rates are better in a larger regime where $\epsilon > c/n^{4/3}$. Note that in machine learning applications the goal is to obtain better generalization which roughly translates to such regimes in practice. Thus, in settings of interest to machine learning, our results are competitive (or better) in comparison to N. Agarwal, Allen Zhu, et al. (2016) and Carmon et al. (2018). Finally, we also note that our algorithm is faster than the one proposed in Jin et al. (2017), which has a time complexity of $O(\|n\|d/\epsilon^{2})$.

10.4.2 Cubic Descent

In this section, we show that the cubic regularization method in Nesterov and Polyak (2006) can be used as Hessian-Focused-Optimizer. More specifically, here Hessian-Focused-Optimizer approximately solves the following optimization problem:

$$y = \arg\min_z \frac{1}{2} \langle \nabla f(x), z - x \rangle + \frac{1}{2} \langle z - x, \nabla^2 f(x)(z - x) \rangle + \frac{M}{6} \|z - x\|^3,$$

(CubicDescent)

and returns $(y, \diamond)$ as output. The following result can be proved for this approach.

**Theorem 10.10** Suppose Svrg (same as Theorem 10.7) is used as Gradient-Focused-Optimizer and CubicDescent is used as Hessian-Focused-Optimizer with $q = 0$, then Algorithm 3 finds a $(\epsilon, \sqrt{\epsilon})$-second order critical point in $T = O(\Delta/\min(p, 1 - p)\epsilon^{3/2})$ with probability at least 0.9.

**Proof** First note that cubic method is a descent method (refer to Theorem 1 of Nesterov and Polyak (2006)); thus, H.1 is trivially satisfied. Furthermore, cubic descent is a Hessian-Focused-Optimizer with $h(n, \epsilon, \gamma) = \frac{2\gamma^3}{81M}\gamma^3$. This, again, follows from Theorem 1 of Nesterov and Polyak (2006). The result easily follows from the aforementioned observations.

In principle, Algorithm 3 with CubicDescent as Hessian-Focused-Optimizer can converge without the use of Gradient-Focused-Optimizer subroutine at each iteration since it essentially reduces to the cubic regularization method of Nesterov and Polyak (2006). However, in practice, we would expect Gradient-Focused-Optimizer to perform most of the optimization and Hessian-Focused-Optimizer to be used for far fewer iterations. Using the method...
developed in Nesterov and Polyak (2006) for solving CubicDescent, we obtain the following corollary.

**Corollary 10.11** The overall running time of Algorithm 3 to find a \((\epsilon, \sqrt{\epsilon})\)-second order critical point, with parameter settings used in Theorem 10.10, is \(O(n d^w / \epsilon^{3/2} + n^2 d / \epsilon^2)\).

Here \(w\) is the matrix multiplication constant. The dependence on \(\epsilon\) is weaker in comparison to Corollary 10.9. However, each iteration of CubicDescent is expensive (as seen from the factor \(d^w\) in the corollary above) and thus, in high dimensional settings typically encountered in machine learning, this approach can be expensive in comparison to HessianDescent.

### 10.4.3 Practical Considerations

The focus of this section was to demonstrate the wide applicability of our framework; wherein using a simple instantiation of this framework, we could achieve algorithms with fast convergence rates. To further achieve good empirical performance, we had to slightly modify these procedures. For Hessian-Focused-Optimizer, we found stochastic, adaptive and inexact approaches for solving HessianDescent and CubicDescent work well in practice. We begin by describing an inexact approach for CubicDescent which we will use for our experiments as well.

**CubicDescent** method of Nesterov and Polyak (2006), which is designed to operate on full batch, i.e., it does not exploit the finite-sum structure of the problem and requires the computation of the gradient and the Hessian on the entire dataset to make an update. However, such full-batch methods do not scale gracefully with the size of data and become prohibitively expensive on large datasets. To overcome this challenge, we devised an approximate cubic regularization method described below:

1. Pick a mini-batch \(\mathcal{B}\) and obtain the gradient and the hessian based on \(\mathcal{B}\), i.e.,

   \[
g = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla f_i(x) \quad H = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla^2 f_i(x) \tag{10.17}
\]

2. Solve the sub-problem

   \[
v^* = \arg \min_v \langle g, v \rangle + \frac{1}{2} \langle v, Hv \rangle + \frac{M}{6} \|v\|^3 \tag{10.18}
\]

3. Update: \(x \leftarrow x + v^*\)

We found that this mini-batch training strategy, which requires the computation of the gradient and the Hessian on a small subset of the dataset, to work well on a few datasets (CURVES, MNIST, CIFAR10). A similar method has been analysed in Cartis and Scheinberg (2017).

Furthermore, in many deep-networks, adaptive per-parameter learning rate helps immensely (D. P. Kingma and Ba, 2014). One possible explanation for this is that the scale of the gradients in each layer of the network often differ by several orders of magnitude. A well-suited optimization method should take this into account. This is the reason for popularity of methods like ADAM or RMSProp in the deep learning community. On similar lines, to account for different per-parameter behaviour in cubic regularization, we modify the sub-problem by adding a diagonal matrix \(M_d\) in addition to the scalar regularization coefficient \(M\), i.e.,

\[
\min_v \langle g, v \rangle + \frac{1}{2} \langle v, Hv \rangle + \frac{1}{6} M \|M_d v\|^3. \tag{10.19}
\]
10.5 EXPERIMENTS

Also we devised an adaptive rule to obtain the diagonal matrix as $M_d = \text{diag}((s + 10^{-12})^{1/9})$, where $s$ is maintained as a moving average of third order polynomial of the mini-batch gradient $g$, in a fashion similar to RMSprop and Adam:

$$s \leftarrow \beta s + (1 - \beta)\left[|g|^3 + 2g^2\right],$$

(10.20)

where $|g|^3$ and $g^2$ are vectors such that $|g|^3 = |g_i|^3$ and $g^2 = g_i^2$ respectively for all $i \in [n]$. The experiments reported on CURVES and MNIST in this paper utilizes both the above modifications to the cubic regularization, with $\beta$ set to 0.9. We refer to this modified procedure as ACubic in our results.

Finally, in the context of deep learning, empirical evidence suggests that first-order methods like Adam (D. P. Kingma and Ba, 2014) exhibit behavior that is in congruence with properties G.1 and G.2. While theoretical analysis for a setting where Adam is used as Gradient-Focused-Optimizer is still unresolved, we nevertheless demonstrate its performance through empirical results in the following section.

10.5 EXPERIMENTS

We now present empirical results for our saddle point avoidance technique with an aim to highlight three aspects of our framework: (i) it successfully escapes non-degenerate saddle points, (ii) it is fast, and (iii) it is practical on large-scale problems. All the algorithms are implemented on TensorFlow (Martin Abadi et al., 2015). In case of deep networks, the Hessian-vector product is evaluated using the trick presented in Pearlmutter (1994). We run our experiments on a commodity machine with Intel® Xeon® CPU E5-2630 v4 CPU, 256GB RAM, and NVidia® Titan X (Pascal) GPU.

10.5.1 Synthetic Problem

To demonstrate the fast escape from a saddle point of our method, we consider the following simple nonconvex finite-sum problem:

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} x^T A_i x + b_i^T x + ||x||_{10}^{10}$$

(10.21)

Here the parameters are designed such that $\sum_i b_i = 0$ and $\sum_i A_i$ matrix has exactly one negative eigenvalue of $-0.001$ and other eigenvalues randomly chosen in the interval $[1, 2]$. The total number of examples $n$ is set to be 100,000 and $d$ is 1000. It is not hard to see that this problem has a non-degenerate saddle point at the origin. This allows us to explore the behaviour of different optimization algorithms in the vicinity of the saddle point. In this experiment, we compare a mix of SVRG and HessianDescent (as in Theorem 10.7) with SGD (with
constant step size), \textsc{adam}, \textsc{svrg} and \textsc{cubicsdescent}. The parameter of these algorithms is chosen by grid search so that it gives the best performance. The subproblem of \textsc{cubicsdescent} was solved with gradient descent (Carmon et al., 2018) until the gradient norm of the subproblem is reduced below $10^{-3}$. We study the progress of optimization, i.e., decrease in function value with wall clock time, IFO calls, and ISO calls. All algorithms were initialized with the same starting point very close to origin.

The parameter selection for all the methods were carried as follows:

1. \textsc{sgd}: The scalar step-size was determined by a grid search.
2. \textsc{adam}: We performed a grid search over $\alpha$ and $\epsilon$ parameters of \textsc{adam} tied together, i.e., $\alpha = \epsilon$.
3. \textsc{svrg}: The scalar step-size was determined by a grid search.
4. \textsc{cubicsdescent}: The regularization parameter $M$ was chosen by grid search. The subproblem was solved with gradient descent (Carmon et al., 2018) with the step-size of solver to be $10^{-2}$ and run till the gradient norm of the subproblem is reduced below $10^{-3}$.

The results are presented in Figure 10.2, which shows that our proposed mix framework was the fastest to escape the saddle point in terms of wall clock time. We observe that performance of the first order methods suffered severely due to the saddle point. Note that \textsc{sgd} eventually escaped the saddle point due to inherent noise in the mini-batch gradient. The other first order methods like \textsc{adam} with higher noise could escape relatively faster whereas \textsc{svrg} with reduced noise stayed stuck at the saddle point. \textsc{cubicsdescent}, a second-order method, escaped the saddle point faster in terms of iterations using the Hessian information. But operating on Hessian information is expensive as a result this method was slow in terms of wall clock time. The proposed framework, which is a mix of the two strategies, inherits the best of both worlds by using cheap gradient information most of the time and reducing the use of relatively expensive Hessian information (ISO calls) by 100x. This resulted in faster escape from saddle point in terms of wall clock time.

10.5.2 Deep Networks

To investigate the practical performance of the framework for deep learning problems, we applied it to two deep autoencoder optimization problems from G. E. Hinton and R. R. Salakhutdinov (2006) called “\textsc{curves}” and “\textsc{mnist}”. Due to their high difficulty, performance on these problems has become a standard benchmark for neural network optimization methods, c.f. Martens (2010), Martens and Grosse (2015), Sutskever et al. (2013), and Vinyals and Povey (2012). The “\textsc{curves}” autoencoder consists of an encoder with layers of size $(28 \times 28)$-400-200-100-50-25-6 and a symmetric decoder totaling in 0.85M parameters. The six units in the code layer were linear and all the other units were logistic. The network was trained on 20,000 images and tested on 10,000 new images. The data set contains images of curves that were generated from three randomly chosen points in two dimensions. \footnote{Data available at: \url{www.cs.toronto.edu/~jmartens/digs3pts_1.mat}} The “\textsc{mnist}” autoencoder consists of an encoder with layers of size $(28 \times 28)$-1000-500-250-30 and a symmetric decoder, totaling in 2.8M parameters. The thirty units in the code layer were linear and all the other units were logistic. The network was trained on 60,000 images and tested on 10,000 new images. The data set contains images of handwritten digits 0–9. The pixel intensities were normalized to lie between 0 and 1. \footnote{Data available at: \url{www.cs.toronto.edu/~jmartens/mnist_all.mat}}

As an instantiation of our framework, we use a mix of \textsc{adam}, which is popular in deep learning community, and approximate cubic regularization method described in Section 10.4.3. For com-
Figure 10.3: Comparison of various methods on a Deep Autoencoder on CURVES (top) and MNIST (bottom). Our mix approach converges faster than the baseline methods and uses relatively few ISO calls in comparison to ApproxCubicDescent.

Comparison the algorithm in Carmon et al. (2018) is impractical for our setting because it requires computing the full gradient at each iteration and is, thus, intractable for problems of our interest. Furthermore, we tried to compare our algorithm with the method in N. Agarwal, Allen Zhu, et al. (2016), however, the algorithm as stated in the paper is not practical. To this end, we compare our algorithm with a practical version of cubic method, ApproxCubicDescent, described in Section 10.4.3. We also compared our algorithm with Adam, which is typically the de-facto method for training large deep networks. The parameters of these algorithms were chosen to produce the best generalization on a held out test set. The regularization parameter $M$ was chosen as the smallest value such that the function value does not fluctuate in the first 10 epochs. We use the initialization suggested in Martens (2010) and a mini-batch size of 1000 for all the algorithms. We report objective function value against wall clock time and ISO calls.

The parameter selection for all the methods were carried as follows:

1. **Adam**: We performed a grid search over $\alpha$ and $\epsilon$ parameters of Adam so as to produce the best generalization on a held out test set. We found it to be $\alpha = 10^{-3}$, $\epsilon = 10^{-3}$ for CURVES and $\alpha = 10^{-2}$, $\epsilon = 10^{-1}$ for MNIST.

2. **ApproxCubicDescent**: The regularization parameter $M$ was chosen as the largest value such function value does not jump in first 10 epochs. We found it to be $M = 10^3$ for both CURVES and MNIST. The sub-problem was solved with gradient descent (Carmon et al., 2018) with the step-size of solver to be $10^{-3}$ and run till the gradient norm of the sub-problem is reduced below 0.1.

The results presented in Figure 10.3 show that our proposed mix framework was the fastest to escape the saddle point in terms of wall clock time. Adam took considerably more time to escape the saddle point, especially in the case of MNIST. While ApproxCubicDescent es-

---

3 We would like to emphasize that N. Agarwal, Bullins, and Hazan (2016) does not contain any empirical results. In order to be fair, we also reached out to the authors of the paper, who via personal communication admitted that the algorithm as implemented verbatim in the paper is not practical and comparison to it is not useful.
caped the saddle point in relatively fewer iterations, each iteration required considerably large number of ISO calls; as a result, the method was extremely slow in terms of wall clock time, despite our efforts to improve it via approximations and code optimizations. On the other hand, our proposed framework seamlessly balances these two methods, resulting in fast decrease of training loss.

### 10.6 Discussion

In this paper, we examined a generic strategy to escape saddle points in nonconvex finite-sum problems and presented its convergence analysis. The key intuition is to alternate between a first-order and second-order based optimizers; the latter is mainly intended to escape points that are only stationary but are not second-order critical points. We presented two different instantiations of our framework and provided their detailed convergence analysis. In this paper, we primarily used SVRG as Gradient-Focused-Optimizer, however, investigating the use of other first-order methods, like Carmon et al. (2017), is an interesting research direction. Also, while both our methods explicitly use the Hessian information, one can also use noisy first-order methods as Hessian-Focused-Optimizer (c.f. noisy SGD in Ge et al. (2015)). In such a scenario, we exploit the negative eigenvalues of the Hessian to escape saddle points by using isotropic noise, and do not explicitly use ISO. For these methods, under strict-saddle point property (Ge et al., 2015), we can show convergence to local optima within our framework.

Our primary goal in this paper was to develop a well-founded, yet practical, framework for finding local minima in nonconvex optimization. While convergence rates in N. Agarwal, Allen Zhu, et al. (2016) may seem slightly better at first glance, we would like to point out that these running times are worst-case time complexity and might involve large constants. For example, N. Agarwal, Allen Zhu, et al. (2016) solve a subroutine based on both gradient and Hessian information at each iteration. Such methods do not scale well in practice because they rely on expensive computations based on the Hessian at every iteration.

We primarily focused on obtaining second-order critical points for nonconvex finite-sums (10.1). This does not necessarily imply low test error or good generalization capabilities. Thus, we should be careful when interpreting the results presented in this paper. A detailed discussion or analysis of these issues is out of scope of this paper. While a few prior works argue for convergence to local optima, the exact connection between generalization and local optima is not well understood, and is an interesting open problem. Nevertheless, we believe the techniques...
presented in this paper can be used alongside other optimization tools for faster and better nonconvex optimization.
Publications


Andreas, Jacob, Marcus Rohrbach, Trevor Darrell, and Dan Klein (2016). “Learning to Compose Neural Networks for Question Answering.” In: NAACL.


Blei, David and Peter I. Frazier (2011). Distance Dependent Chinese Restaurant Processes. JMLR.


Bordes, Antoine, Nicolas Usunier, Sumit Chopra, and Jason Weston (2015). “Large-scale Simple Question Answering with Memory Networks.” In: CoRR.


Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). “Imagenet: A large-scale hierarchical image database.” In: CVPR.


Fan, Haoqiang, Hao Su, and Leonidas Guibas (n.d.). “A point set generation network for 3d object reconstruction from a single image.” In:


Feichtenhofer, Christoph, Axel Pinz, and Andrew Zisserman (2016). “Convolutional two-stream network fusion for video action recognition.” In: CVPR.


Index, Cisco Visual networking (2016). “Forecast and methodology, 2016-2021, white paper.” In: San Jose, CA, USA.


Joshi, Mandar, Uma Sawant, and Soumen Chakrabarti (Oct. 2014). “Knowledge Graph and Corpus Driven Segmentation and Answer Inference for Telegraphic Entity-seeking Queries.” In: EMNLP.


Kantorov, Vadim and Ivan Laptev (2014). “Efficient feature extraction, encoding and classification for action recognition.” In: CVPR.


Mathieu, Michael, Camille Couprie, and Yann LeCun (2016). “Deep multi-scale video prediction beyond mean square error.” In: ICLR.


Miller, Alexander H., Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston (2016). “Key-Value Memory Networks for Directly Reading Documents.” In: In EMNLP.


Riedel, Sebastian, Limin Yao, and Andrew McCallum (2010). “Modeling Relations and Their Mentions Without Labeled Text.” In: ECML PKDD.

Riedel, Sebastian, Limin Yao, Andrew McCallum, and Benjamin M. Marlin (2013). “Relation Extraction with Matrix Factorization and Universal Schemas.” In: NAACL.


Salakhutdinov, Ruslan, Sam Roweis, and Zoubin Ghahramani (n.d.). Relationship between gradient and EM steps in latent variable models.


Savenkov, Denis and Eugene Agichtein (2016). “When a knowledge base is not enough: Question answering over knowledge bases with external text data.” In: SIGIR. ACM.


Sun, Huan, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang (2015). “Open domain question answering via semantic enrichment.” In: WWW. ACM.


Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio (2015). “Show, attend and tell: Neural image caption gen-
eration with visual attention.” In: International Conference on Machine Learning, pp. 2048–2057.
Xu, Kun, Yansong Feng, Songfang Huang, and Dongyan Zhao (2016). “Hybrid Question Answering over Knowledge Base and Free Text.” In: COLING.
Xu, Kun, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao (2016). “Question Answering on Freebase via Relation Extraction and Textual Evidence.” In: ACL.
Yao, Xuchen and Benjamin Van Durme (June 2014). “Information Extraction over Structured Data: Question Answering with Freebase.” In: ACL.


