Representation Learning @ Scale

DIVERSITY,
INTERPRETABILITY,
SCALABILITY

Manzil Zaheer

January 2018

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

Thesis Committee:
Barnabás Poczos, Co-Chair
Ruslan Salakhutdinov, Co-Chair
Alexander J Smola (Amazon)
Andrew McCallum (UMass Amherst)

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

Support: Thank funding agencies
Machine learning techniques are reaching or exceeding human level performances in tasks like image classification, translation, and text-to-speech. The success of these machine learning algorithms have been 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. for understanding how humans might categorize objects. Despite these instances of success, many open questions remain.

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 by leveraging their 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.

Once we have the representations, 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 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 methods 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.

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 in performance.

To summarize, in this thesis, we want to explore 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.
4.3.3 Dealing with Stirling Numbers ............................................. 35
4.3.4 Preliminary Experiments .................................................. 35
4.4 Escaping Saddle Points ......................................................... 36
  4.4.1 Problem Setup .............................................................. 36
  4.4.2 Generic Framework ....................................................... 37
  4.4.3 Convergence Analysis ................................................... 38
  4.4.4 Preliminary Experiments ................................................. 39
5 MY OTHER RESEARCH .......................................................... 41
  5.1 Random Graphs ............................................................... 41
  5.2 Small Sample Estimation .................................................. 41
  5.3 Spectral Methods ............................................................ 41

PUBLICATIONS ................................................................. 43

BIBLIOGRAPHY ................................................................. 45
Introduction

The digitally connected world of today involves potentially valuable data growing at a tremendous pace. 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. The value generation lies in transforming unstructured data into a usable information or representation, which is well organized, searchable and understandable, so as to enable visualization, summarization, personalized recommendation (of articles to read, videos to watch, locations to visit, restaurants to dine at), fraud detection, medical image analysis, or more informed decision making in general.

Machine learning and intelligent systems are used to carry out the task listed above and have become an indispensable part of our modern society. The performance of such machine learning have reached or exceeded human levels. The success of the methods heavily dependends on the choice of data representation (or features) used Bengio, Courville, and Vincent, 2013, for which deep learning has been a boon. This thesis is centered around representation learning, i.e., learning representations of the data that make it easier to extract useful information when building classifiers or other predictors for human and/or machine consumption.

There are two main categories for learning representations. First is using Bayesian models with a parsimonious set of latent random variables that describe a 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.

1.1 Aspects of Representation Learning

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 highly 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 PROPOSAL

In this chapter, a high level overview of my research was provided that forms of the core of this thesis. In Chapters 2, 3, and 4, I will discuss more details of the my research in addressing the three important issues raised above and provide details of the work that is planned to be incorporated in the thesis. Chapter 5 briefly summarizes my other research contributions that are not included in my thesis.

Key Proposed Work

- Representations for sets (in Chapter 2)
  1. More theoretical investigations
  2. Generative model: GANs
- State Space Models with LSTM (in Chapter 3)
  1. Exact inference using SMC
  2. Approximate inference
- Bayesian Nonparametrics (in Chapter 4)
  1. Finalize JMLR

Approximate Timeline

- Date of Proposal: January 12th, 2018.
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 by leveraging their structure and establishing new mathematical properties. Specifically, We look at three cases — when the data is collection of bags/sets, when the data comes in compressed form, and when we have mixture of different types of data.

## 2.1 Set Data

A typical machine learning algorithm, like regression or classification, is designed for fixed dimensional data instances. 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 to the permutation of set elements. Examples 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 unsupervised 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 an important problem in many scientific applications as well, e.g. given a small set of interesting celestial objects, astrophysicists might want to find similar ones in large sky surveys.

Extension of existing machine learning techniques like deep network to handle the case when the inputs or outputs are permutation invariant sets rather than fixed dimensional vectors is not trivial. For example, we may want to find sum of a set of numbers or sum of set of images of numbers. Here the order of elements in the set does not matter and number of elements in the set can vary. If the size of the set were fixed one could design an MLP or CNN to regress to sum value, but it is not clear how to extended the network in case of an extra digit or image. One might consider treating the input as a sequence and using RNN/LSTM as quick fix, but such a solution does not guarantee invariance to change in order of the inputs. Researchers have only recently started to investigate solutions for specific usecases Muandet, Balduzzi, and Schoelkopf, 2013; Muandet, Fukumizu, et al., 2012; Oliva, Poczos, and Schneider, 2013; Szabo et al., 2016. We want to design a generic framework to deal with the setting where input and possibly output instances in a machine learning task are sets.

### 2.1.1 Permutation Invariance

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 $\mathcal{X} = 2^X$, then we would like the response of the function to be “indifferent” to the ordering of the elements.

**Property 1** A function $f : 2^X \rightarrow \mathcal{Y}$ acting on sets must be permutation invariant to the order of objects in the set, i.e. for any permutation $\pi : f(\{x_1, \ldots, x_M\}) = f(\{x_{\pi(1)}, \ldots, x_{\pi(M)}\})$.

We want to study the structure of functions on sets. Their study in total generality is extremely difficult, so we analyze case-by-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** A function $f(X)$ operating on a set $X$ having elements from a countable universe, is a valid set function, i.e., invariant to the permutation of instances in $X$, iff it can be decomposed in the form $\rho(\sum_{x \in X} \phi(x))$, for suitable transformations $\phi$ and $\rho$.

When applied to machine learning task, in the supervised setting, given $N$ examples 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.

### 2.1.2 Architecture

The structure of permutation invariant functions in Theorem 2 hints at a general strategy for inference over sets of objects, which we call DeepSets. Replacing $\phi$ and $\rho$ by universal approximators leaves matters unchanged. Then, it remains to learn these approximators, yielding in the following model:

- Each instance $x_m$ is transformed (possibly by several layers) into representation $\phi(x_m)$.
- The representations $\phi(x_m)$ are added up and the output is processed using the $\rho$ network like 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 is to add up all representations and apply nonlinear transformations.

### 2.1.3 Preliminary Experiments

#### 2.1.3.1 Sum of Digits

We begin by comparing 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 and 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** MNIST\textsuperscript{8m} 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.
Figure 2.1: 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 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 Fig. 2.1. DeepSets generalize much better. Note for image case, the best classification error for single digit is around \( p = 0.01 \) for MNIST, 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 Fig. 2.1(b).

2.1.3.2 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. Lin, Tai, and G.-J. Wang, 2004; Maturana and Scherer, 2015; S. Ravanbakhsh 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

<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>( 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.
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. Tab. 2.1 compares our method using three layers against the competition.

2.1.4 Proposed Work

Preliminary mathematical result and experimental work seem promising. We would like to explore the following:

- Till now the permutation invariance has only been proved for case when the universe contains countable elements. We want to extend it to uncountable case.
- Till now the representation is used for discriminative models. We want to seek the representations utility in generative models like GAN.
- Finally, we would like to carry out more diverse experiments.

2.2 COMPRESSED DATA

Compressed data occurs commonly, especially in case of video. Largely video is available only in its compressed form from storage, network transmission, and even on all modern cameras there are hardware-accelerated video codecs. Current machine learning frameworks like deep networks requires decompression of the compressed data before feeding it into the system. In other words, decompressing the data is actually an inconvenience. We want explore the possibility to train deep networks directly on compressed videos.

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 highly redundant RGB stream.

2.2.1 Primer on Video Compression

Compression algorithms, such as MPEG, H.264, and HEVC, leverage the fact that successive frames are usually very similar. We can efficiently store one frame by reusing contents from another frame and only store the difference. They 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 T^{(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 \( T^{(t)} \) and a residual \( \Delta^{(t)} \). This gives the recurrence relation for reconstruction of P-frames as

\[
I^{(t)}_i = I^{(t-1)}_{i-T^{(t)}} + \Delta^{(t)}_i,
\]

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.

### 2.2.2 Modeling Compressed Representations

Our goal is to design a computer vision system for action recognition that operates directly on the stored compressed video. Feeding I-frames into a deep network is straightforward since they are just images. How about P-frames? From Fig. 2.2 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 outputs of corresponding CNNs from the image, motion vectors, and residual will have different properties. To combine them, we have to come up fusion strategies.

Digging deeper, one can see 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. A simple strategy would be to reuse features from the reference frame, and only update the features given the new information. This recurrent definition screams for RNNs to aggregate features along the chain. However, preliminary experiments suggests the elaborate modeling effort in vain. 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, as illus-
trated in Fig. 2.3. Given a pixel at location $i$ in frame $t$, let $\mu_T(t)(i) := i - T(t)_{i}$ be the referenced location in the previous frame. The location traced back to frame $k < t$ is given by

$$J_t^{(t,k)} := \mu_T(t) \circ \cdots \circ \mu_T(k). \quad (2.2)$$

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

$$D(t)_i := i - J_t^{(t,k)}, \quad \text{and} \quad R(t)_i := \Delta^{(k+1)}_{J_t^{(t,k+1)}} + \cdots + \Delta^{(t-1)}_{J_t^{(t,J-1)}} + \Delta^{(t)}_{J_t^{(t,t)}},$$

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

$$I^{(t)}_i = I^{(0)}_{i - P(t)} + R^{(t)}_i, \quad t = 1, 2, \ldots,$$

as shown in Fig. 2.3. Here P-frames depend only on the I-frame and can be processed in parallel.

**Proposed network.** Fig. 2.3 shows the graphical illustration of the proposed model. The input to our model is an I-frame, followed by $T$ P-frames, i.e. $\left( I^{(0)}, D^{(1)}, R^{(1)}, \ldots, D^{(T)}, R^{(T)} \right)$. For simplicity, we set $t = 0$ for the I-frame. Each input source is modeled by a CNN, i.e.

$$x_{\text{RGB}}^{(0)} := \phi_{\text{RGB}}(I^{(0)})$$
$$x_{\text{motion}}^{(t)} := \phi_{\text{motion}}(D^{(t)})$$
$$x_{\text{residual}}^{(t)} := \phi_{\text{residual}}(R^{(t)})$$

While I-frame features $x_{\text{RGB}}^{(0)}$ are used as is, P-frame features $x_{\text{motion}}^{(t)}$ and $x_{\text{residual}}^{(t)}$ need to incorporate the information from $x_{\text{RGB}}^{(0)}$. For fusion, we found simple sum of scores to work best.

**Implementation.** 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.
2.2 Compressed Data

<table>
<thead>
<tr>
<th>Method</th>
<th>UCF-101</th>
<th>HMDB-51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karpathy et al., 2014</td>
<td>65.4</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-50 He et al., 2016</td>
<td>82.3</td>
<td>48.9</td>
</tr>
<tr>
<td>(from ST-Mult Feichtenhofer, Pinz, and Wildes, 2017)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResNet-152 He et al., 2016</td>
<td>83.4</td>
<td>46.7</td>
</tr>
<tr>
<td>(from ST-Mult Feichtenhofer, Pinz, and Wildes, 2017)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C^3D Tran, Bourdev, et al., 2015</td>
<td>82.3</td>
<td>51.6</td>
</tr>
<tr>
<td>Res^3D Tran, Ray, et al., 2017</td>
<td>85.8</td>
<td>54.9</td>
</tr>
<tr>
<td>TSN (RGB-only) L. Wang et al., 2016*</td>
<td>85.7</td>
<td>-</td>
</tr>
<tr>
<td>TLE (RGB-only) Diba, Sharma, and Van Gool, 2017*</td>
<td>87.9</td>
<td>54.2</td>
</tr>
<tr>
<td>I^3D (RGB-only) Carreira and Zisserman, 2017*</td>
<td>84.5</td>
<td>49.8</td>
</tr>
<tr>
<td>MV-CNN 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><strong>CoViAR</strong></td>
<td><strong>90.4</strong></td>
<td><strong>59.1</strong></td>
</tr>
</tbody>
</table>

Table 2.2: Action recognition accuracy on UCF-101 Soomro, Zamir, and Shah, 2012 and HMDB-51 Kuehne et al., 2011. The table lists real-time methods that do not require computing optical flow. Our method outperforms all baselines. Asterisk indicates results evaluated only on split 1 of the datasets. They are listed purely for reference.

2.2.3 Preliminary Experiments

Datasets and Protocol. We evaluate our method Compressed Video Action Recognition (CoViAR) on two action recognition datasets: UCF-101 Soomro, Zamir, and Shah, 2012 and HMDB-51 Kuehne et al., 2011. UCF-101 and HMDB-51 contain short (< 10-second) trimmed videos, each of which is annotated with one action label. UCF-101 contains 13,320 videos from 101 action categories. HMDB-51 contains 6,766 videos from 51 action categories. For testing we uniformly sample 25 frames, each with flips plus 5 crops, and average the scores.

Training Details. Following TSN L. Wang et al., 2016, we resize UCF-101 and HMDB-51 videos to 340 × 256. Our models are pre-trained on the ILSVRC 2012-CLS dataset Deng et al., 2009, and fine-tuned using Adam D. Kingma and Ba, 2014 with a batch size of 40. Learning rate starts from 0.001 for UCF-101/HMDB-51 and 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. 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.

Accuracy. We compare accuracy of our method CoViAR with state-of-the-art models in Tab. 2.2. Some previous works consider 3D convolution to learn temporal structures whereas others consider more complicated fusions and pooling. Our method uses faster 2D CNNs and simple late fusion without additional supervision, yet significantly outperforming the baselines.

2 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.
Figure 2.4: 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.

**Visualizations.** In Fig. 2.4, 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.
2.2.4 Proposed Work

We see that, quite surprisingly, our method is both faster and more accurate, while being simpler to implement than previous works. We would like to further investigate and try to

- conduct more experiments,
- compare against methods utilizing optical flow, and
- extend the idea to longer duration compressed video tasks.

2.3 HETEROGENEOUS DATA

Often to perform a task accurately one must rely on data from multiple sources. A prominent case is of question answering. 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. 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. We want to explore creation of a common representation that can support reasoning on the union of both structured KBs and unstructured text by aligning them in a common embedded space. We try to 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.

2.3.1 For Question Answering

**Problem Definition** Given a question $q$ with words $w_1, w_2, \ldots, w_n$, where these words contain one `_blank_` and at least one entity, our goal is to fill in this `_blank_` with an answer entity $q_a$ using a knowledge base $K$ and text $T$.

**Background on Memory Networks** MemNNs are neural attention models with external differentiable memory. 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 et al., 2015, where the memory consists of distributed representation of sentences in the comprehension. Recently, key-value MemNN are introduced 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.

2.3.2 Model for Two Information Source

Our model is a MemNN with universal schema as its memory as shown in Fig. 2.5

**Memory:** Our memory $M$ comprise of both KB and textual triples from universal schema. Each memory cell is in the form of key-value pair. Let $(s, r, o) \in K$ 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 \( (s, [w_1, \ldots, w_n], o) \in T \) represent a textual fact. We represent the key as a distributed representation using a bidirectional LSTM [Graves and Jürgen Schmidhuber, 2005; Hochreiter and Jürgen Schmidhuber, 1997], \( k \in \mathbb{R}^{2d} \). 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 Fig. 2.5, each cell in the matrix represents a memory containing the encoding 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 Fig. 2.5, we only consider memory cells containing USA. We compute the context vector \( c_t \) for the next iteration \( t \) as follows:

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

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.

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

Finally, we minimize the cross-entropy loss to train in an end-to-end fashion.
2.3 HETEROGENEOUS DATA

<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>41.1</td>
<td>39.9</td>
</tr>
</tbody>
</table>

Table 2.3: QA results on Spades.

2.3.3 Preliminary Experiments

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. We use Spades Bisk et al., 2016 which contains 93k fill-in-the-blank cloze-styled questions created from ClueWeb and having 1.8M entities in total. 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.

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

- **ONLYKB**: In this model, MemNN memory contains only the facts from KB.
- **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 to combine the evidences from the above mentioned 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.

**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. We considered up to a maximum of 5k KB facts and 2.5k textual facts for a question. We used Adam D. P. Kingma and Ba, 2015 with the default hyperparameters (learning rate=$1e^{-3}$, $\beta_1=0.9$, $\beta_2=0.999$, $\epsilon=1e-8$) for optimization. The batch size during training was set to 32.

To train the UniSchema model, we initialized the parameters from a trained ONLYKB model. We found that this is crucial in making the UniSchema 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.

Tab. 2.3 shows the main results on Spades. UniSchema outperforms all our models validating our hypothesis that exploiting universal schema for QA is better than using either KB or text alone. Tab. 2.4 shows some of the questions which UniSchema answered but ONLYKB failed.

**Ablation study** 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 2.6
Table 2.4: A few questions on which ONLYKB fails to answer but UNISHEMA succeeds.

<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</td>
<td>Latinos</td>
</tr>
<tr>
<td><em>blank</em>.</td>
<td></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>

shows, when the KB coverage is less than 16 facts per entity, UNISHEMA outperforms ONLYKB by a wide-margin indicating UNISHEMA is robust even in resource-scarce scenario, whereas ONLYKB is very sensitive to the coverage. UNISHEMA 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 $F_1$ 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.

![Image](image.png)

Figure 2.6: Performance on varying the number of available KB facts during test time. UNISHEMA model consistently outperforms ONLYKB

2.3.4 Future Work

In this work, we showed universal schema is a promising knowledge source for QA than using KB or text alone. However current model is just a simplistic amalgamation of both text and KB information source. It cannot based on the text retrieve more KB entries or vice versa. In other words, current approach is limited to a single hop from question information retrieval. It would be interesting to explore Monte Carlo tree search based approach for a multi-hop reasoning reasoning across different information sources.
Mixed Representations

Once we have the representations, 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 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 methods to learn mixed (deep+continuous) representations retaining best of both the worlds. Specifically, we look at two cases — combining dense distributed word representations with sparse topic models, and combining LSTMs with topic models for joint clustering and nonlinear dynamics modeling.

3.1 CONTINUOUS WORD EMBEDDING + LDA

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 M. I. Jordan, 2003. These priors are remarkably effective at producing useful and interpretable results. We further expect topics to be semantically coherent. However, this prior preference for semantic coherence is not encoded in the model. On the other hand, 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. They have been incorporated to produce state-of-the-art results in numerous supervised tasks from the word level to document level Le and Mikolov, 2014; Turian, Ratinov, and Bengio, 2010, inter alia; however, they have played a minor role in unsupervised problems.

We want to 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 from a multivariate Gaussian. This should yield a better representation of the document. As word embeddings have been shown to capture lexico-semantic regularities in language and Gaussian distributions capture a notion of centrality in space, our Gaussian LDA model encodes a prior preference for semantically coherent topics. Furthermore, unlike traditional LDA which cannot handle out of vocabulary (OOV) words, our model should be able to exploit the contiguity of semantically similar words in the embedding space and handle words it has never been seen before.

3.1.1 Model

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 $i$. 
We characterize each topic \( k \) as a multivariate Gaussian distribution with mean \( \mu_k \) and covariance \( \Sigma_k \). 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 \( \mu_k \sim \mathcal{N}(\mu, \frac{1}{\kappa} \Sigma_k) \)
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 \( v_{d,n} \sim \mathcal{N}(\mu_{z_n}, \Sigma_{z_n}) \)

3.1.2 Posterior Inference

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:

\[
p(z_{d,i} = k \mid z_{-(d,i)}, V_d, \zeta) \propto (\kappa_k + \alpha_k) \times t_{v_k-M+1} \left( v_{d,i} \mid \mu_k, \kappa_k + 1, \Sigma_k \right)
\]

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 \( d \); \( t_v(x \mid \mu', \Sigma') \) is the multivariate \( t \)-distribution with \( v' \) 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:

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

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

\[
\begin{align*}
\bar{v}_k &= \frac{\sum_d \sum_i \mathbb{1}_{z_{d,i} = k}(v_{d,i})}{N_k} \\
C_k &= \sum_d \sum_i \mathbb{1}_{z_{d,i} = k}(v_{d,i} - \bar{v}_k)(v_{d,i} - \bar{v}_k)^	op
\end{align*}
\]
### Gaussian LDA topics

<table>
<thead>
<tr>
<th>word</th>
<th>word</th>
<th>word</th>
<th>word</th>
<th>word</th>
<th>word</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostile</td>
<td>play</td>
<td>government</td>
<td>people</td>
<td>university</td>
<td>hardware</td>
</tr>
<tr>
<td>murder</td>
<td>round</td>
<td>state</td>
<td>god</td>
<td>program</td>
<td>interface</td>
</tr>
<tr>
<td>violence</td>
<td>win</td>
<td>group</td>
<td>jews</td>
<td>public</td>
<td>mode</td>
</tr>
<tr>
<td>victim</td>
<td>players</td>
<td>initiative</td>
<td>israel</td>
<td>law</td>
<td>devices</td>
</tr>
<tr>
<td>testifying</td>
<td>games</td>
<td>board</td>
<td>christians</td>
<td>institute</td>
<td>rendering</td>
</tr>
<tr>
<td>provoking</td>
<td>goal</td>
<td>legal</td>
<td>christian</td>
<td>high</td>
<td>renderer</td>
</tr>
<tr>
<td>legal</td>
<td>challenge</td>
<td>bill</td>
<td>great</td>
<td>research</td>
<td>user</td>
</tr>
<tr>
<td>citizens</td>
<td>final</td>
<td>general</td>
<td>jesus</td>
<td>college</td>
<td>computers</td>
</tr>
<tr>
<td>conflict</td>
<td>playing</td>
<td>policy</td>
<td>muslims</td>
<td>center</td>
<td>monitor</td>
</tr>
<tr>
<td>victims</td>
<td>hitting</td>
<td>favor</td>
<td>religion</td>
<td>study</td>
<td>static</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>jones</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>corporate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>flying</td>
</tr>
</tbody>
</table>

### Multinomial LDA topics

<table>
<thead>
<tr>
<th>word</th>
<th>word</th>
<th>word</th>
<th>word</th>
<th>word</th>
<th>word</th>
</tr>
</thead>
<tbody>
<tr>
<td>turkish</td>
<td>year</td>
<td>people</td>
<td>god</td>
<td>university</td>
<td>window</td>
</tr>
<tr>
<td>armenian</td>
<td>writes</td>
<td>president</td>
<td>jesus</td>
<td>information</td>
<td>image</td>
</tr>
<tr>
<td>people</td>
<td>game</td>
<td>mr</td>
<td>people</td>
<td>national</td>
<td>color</td>
</tr>
<tr>
<td>armenians</td>
<td>good</td>
<td>don</td>
<td>bible</td>
<td>research</td>
<td>file</td>
</tr>
<tr>
<td>armenia</td>
<td>team</td>
<td>money</td>
<td>christian</td>
<td>center</td>
<td>windows</td>
</tr>
<tr>
<td>turks</td>
<td>article</td>
<td>government</td>
<td>church</td>
<td>april</td>
<td>program</td>
</tr>
<tr>
<td>turkey</td>
<td>baseball</td>
<td>stephanopoulos</td>
<td>christ</td>
<td>san</td>
<td>display</td>
</tr>
<tr>
<td>don</td>
<td>time</td>
<td>make</td>
<td>christians</td>
<td>number</td>
<td>jpeg</td>
</tr>
<tr>
<td>greek</td>
<td>games</td>
<td>life</td>
<td>year</td>
<td>problem</td>
<td>satellite</td>
</tr>
<tr>
<td>soviet</td>
<td>season</td>
<td>clinton</td>
<td>time</td>
<td>conference</td>
<td>screen</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>article</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>served</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>police</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PMI score</th>
<th>PMI score</th>
<th>PMI score</th>
<th>PMI score</th>
<th>PMI score</th>
<th>PMI score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8302</td>
<td>0.9302</td>
<td>0.4943</td>
<td>2.0306</td>
<td>0.5216</td>
<td>2.3615</td>
</tr>
<tr>
<td>2.7660</td>
<td>1.4999</td>
<td>1.1847</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Top words of some topics from Gaussian-LDA and multinomial LDA on 20-newsgroups for $K = 50$. Words ranked based on posterior predictive distribution. The last row for each method indicates the PMI score (w.r.t. Wikipedia co-occurrence).

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.

#### 3.1.3 Preliminary Experiments

We run our experiments on two datasets 20-NEWSGROUP\(^1\) and NIPS\(^2\). All the datasets were tokenized and lowercased with cdec Dyer et al., 2010.

**Quantitative Analysis** Typically topic models are evaluated based on the likelihood of held-out documents. But in this case, it is not easy to compare perplexities with LDA due to the continuous distribution. Moreover, J. Chang 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. A higher PMI score implies a more coherent topic as it means the topic words usually co-occur in the same

---

\(^1\) 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/](http://qwone.com/~jason/20Newsgroups/)

\(^2\) 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](http://www.cs.nyu.edu/~roweis/data.html)
document. In the last line of Tab. 3.1, 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 believed to be a good measure of semantic coherence.

**Qualitative Analysis**  Tab. 3.1 shows top words from topics from Gaussian-LDA and LDA on the 20-news dataset for \( K = 50 \). 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 did not capture the ‘space’ topics which classical LDA could identify.

### 3.1.4 Future Work

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 *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 et al., 2014; Reisinger and Mooney, 2010 and we keep this as an avenue for future research.

### 3.2 LSTM + LDA

Sequential data prediction is an important problem in machine learning spanning over a diverse set of applications. For example, when applied to user activity modeling where the aim is to predict the next activity of the user given the history. A good model of sequential data should be accurate, sparse, and interpretable. 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 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 require a lot of parameters and are generally uninterpretable and inaccessible to humans Strobelt et al., 2016.

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, which are 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 Hu, Hall, and Attenberg, 2014. Topic models are popular for their ability to organize the data into a smaller set of prominent themes or topics. These topic representations are generally accessible to humans and easily lend themselves to being interpreted.
We want to explore a model that bridges the gap between the sequential RNN’s and the non-sequential LDA. We call this model Latent LSTM Allocation (LLA). 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 rather than at user action/word level. LLA inherits sparsity and interpretability from LDA, while borrowing accuracy from LSTM.

3.2.1 Latent LSTM Allocation

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 \(p(w_t|z_t)\), similar to LDA. Suppose we have \(K\) topics, vocabulary size \(V\), and a document collection \(D\) with each document \(d \in D\) composed of \(N_d\) tokens. With these notations, the complete generative process 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 3.1, thereby the number of parameters is reduced significantly compared to regular recurrent language model (RRLM) Mikolov, Karafiát, et al., 2010. 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., \(K \ll V\). This allows us to get rid of large \(V \times H\) word embedding look-up table and softmax matrix. Instead we map from hidden 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.
3.2.2 Inference

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,

$$\sum_d \log p(w_d | \text{LSTM, } \phi) \geq \sum_d \sum_z q(z_d) \log \frac{p(z_d; \text{LSTM}) \prod_t p(w_{d,t} | z_{d,t}; \phi)}{q(z_d)},$$

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. 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|topic emission model.

**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,

$$\phi_{wk} = \frac{\# \{w_{d,t} = w \text{ and } z_{d,t} = k\} + \beta}{\# \{n_{wk} = k\} + V \beta} = \frac{n_{wk} + \beta}{n_k + V \beta'},$$

and for the latter we resort to stochastic gradient ascent, increasing likelihood in expectation.
3.2 LSTM + LDA

3.2.3 Preliminary Experiments

For all experiments we 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: LDA, dlLDA, and LSTMs. 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. We trained all deep models using stochastic gradient decent with Adam D. Kingma and Ba, 2014.

Accuracy vs model size  Fig. 3.2(a) compares model size in terms of number of parameters with model accuracy in terms of test perplexity. LDA yields the smallest model size due to the sparsity bias implicit in the Dirichlet prior. On the other hand, LSTMs give the best accuracies; however, the model size is order of magnitudes larger. 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 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.

Interpretability  Similar to LDA, the proposed LLA also uncovers global themes, a.k.a topics prevalent in the model. We found qualitatively the topics produced by LLA to be cleaner. For example, in Tab. 3.2 left 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 Tab. 3.2 right. Modeling sentences as an ordered sequence allows distinction between
the subjects and objects in the sentence. 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. To understand this, we built a topic hierarchy using the embeddings which uncovers interesting facts about the data. For example in Fig. 3.2(b), 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.

3.2.4 Proposed Work

Preliminary work yielded promising trade-off between accuracy and interpretability. Thus, we would like to pursue this line of work further as follows:

- Extend to general state space models with arbitrary, but parametric emission models while using LSTM for dynamics
- Develop a more general and accurate inference framework using sequential Monte Carlo. Subsequently, find fast approximations thereof.
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. In case of learning Bayesian representations, we want to introduce new ways to parallelize, reduce look-ups, and handle variable state space size. In case of learning deep representations, and escape saddle points.

4.1 PARALLELIZATION

Variational methods M. I. Jordan et al., 1999 and Markov chain Monte-Carlo (MCMC) Gilks, Richardson, and Spiegelhalter, 1995 have become the *sine qua non* for inferring the posterior in Bayesian latent variable models with rich statistical dependencies. 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, but 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. Instead, we employ stochastic cellular automata (SCA), which like conventional cellular automata are massively parallel, but with stochastic updates.

We propose exponential SCA (ESCA) for inference in latent variable models with complete data likelihood in the exponential family. ESCA is embarassingly parallel because it is an SCA, and has a minimal memory footprint because it stores only the data and the 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) Hoffman et al., 2013 require storing the variational parameters, while MCMC-based methods, such as YahooLDA Smola and Narayanamurthy, 2010 require storing the latent variable assignments. Furthermore, our algorithm employs double-buffing for lock-free parameter updates (assuming atomic increments) while enabling the use of approximate counters. Thus, we substantially reduce memory costs and communication requirements in distributed enviroments.

4.1.1 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. Formally, the automaton, is given by an evolution function \( \Phi : S \rightarrow S \) over the state space \( S = Z \rightarrow C \) which is a mapping from the space of cell identifiers \( Z \) to cell values \( C \). The global evolution function applies a local function \( \phi(z_i(c_1, c_2, \ldots, c_r) \mapsto c) \) s.t. \( c_i = s(z_i) \) to every cell \( z_i \in Z \). That is, \( \phi \) examines the values of each of the neighbors of cell \( z_i \) and then stochastically computes a new value \( c_i \). The dynamics begin with a state \( s_0 \in S \) that can be configured using the data or a heuristic. Exponential SCA (ESCA) achieves
better computational efficiency by exploiting the structure of the sufficient statistics 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.

**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 produce a sequence of states that correspond to complete variable assignments $s$ via an evolution function $\Phi$.

Suppose we want the MAP estimate for $\eta$ that explains 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, this is problematic because in general all latent variables depend on each other via the global parameters $\eta$ and a naive approach to updating a single cell would then require examining every other cell in the automaton.

Fortunately, if we further suppose that the complete data likelihood is in the exponential family, i.e., $p(z_i, x_i | \eta) = \exp\left(\langle T(z_i, x_i), \eta \rangle - g(\eta) \right)$ then the sufficient statistics are given by $T(z, x) = \sum_i T(z_i, x_i)$ and we can thus express any estimator of interest as a function of just $T(z, x)$ which factorizes over the data. 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 $\Phi(T^t) \mapsto T^{t+1}$.

**Stochastic EM** Before we present ESCA, we first describe stochastic EM (SEM) Celeux and Diebolt, 1985. Suppose we want the MAP estimate for $\eta$, max$_\eta p(x, \eta) = \max_\eta \int p(z, x, \eta) \mu(dz)$ and employ expectation maximization (EM):

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

**M-step** Find $\eta^{(t+1)}$ that maximizes the expected log-likelihood with respect to the conditional

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

where $\xi(\eta) = \nabla g(\eta)$ is invertible as $\nabla^2 g(\theta) > 0$ and $n_0, T_0$ parametrize the conjugate prior. Although EM exposes substantial parallelism, it is difficult to scale, since the dense structure $p(z_i | x_i, \eta^{(t)})$ defines values for all possible outcomes for $z$ and thus puts tremendous pressure

Figure 4.1: Efficient (re)use of buffers
on memory bandwidth. To overcome this we introduce sparsity by employing SEM. SEM introduces an S-step after the E-step that replaces the full distribution with a single sample:

**s-step** Sample $z_1^{(t)} \sim p(z_1|x_i; \eta^{(t)})$ 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 $p(z_i|x_i; \eta^{(t)})$ is feasible. We can now employ fast samplers, such as the alias method, exploit sparsity, reduce CPU-RAM bandwidth while still maintaining massive parallelism. More importantly, the S-step also enables all three steps to now be expressed in terms of the current sufficient statistics. This enables distributed and parallel implementations that efficiently execute on an SCA.

**ESCA for latent variable models** We now present ESCA as SEM on an SCA in which each cell corresponds to a data point with its associated latent variables. Define an SCA over the state space $S = Z \rightarrow K \times X$, where $Z$ is the set of cell identifiers (e.g., one per data point), $K$ 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 $s_0 = z \mapsto (k_z, x)$.

We now need to describe the evolution function $\Phi$. For state $s$ and cell $z$ define the distribution:

$$p_z(k|s) = f(z, T(s))$$  \hspace{1cm} (4.2)

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 $\Phi(s) = z \mapsto \phi(s, z)$. Finally, the SCA algorithm simulates the evolution function $\Phi$ starting with $s_0$. We remark that ESCA converges weakly to a distribution with mean equal to some root of the score function $(\nabla_\eta \log p(x_i; \eta))$ and thus a MAP fixed point.

Our implementation has two copies of the data structure containing sufficient statistics $T^{(0)}$ and $T^{(1)}$. We do not compute the values $T(z, x)$ but maintain their sum as we impute values of 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 (Fig. 4.1). Then in the next iteration $2t + 1$ we reverse the roles of data structure, i.e. read from $T^{(1)}$ and increment $T^{(0)}$.

**Algorithm 1 ESCA**

1: Randomly initialize each cell
2: for $t = 0 \rightarrow$ num iterations do
3: for cell $z$ independently in parallel 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
8: 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.
Figure 4.2: Evolution of log likelihood on Pubmed dataset over number of iterations and time.

Although there is a synchronization barrier after each round, its effect is mitigated because each cell does the same amount of work. Therefore, evenly balancing the work load across computation nodes is trivial, even for a heterogeneous cluster.

**esca for lda** Basically, ESCA simulates the inference steps of SEM on a SCA. For LDA, the state space is \( S = \mathcal{Z} \rightarrow \mathcal{K} \times \mathcal{M} \times \mathcal{V} \) where \( \mathcal{Z} \) is the set of cell identifiers (one per token in our corpus), \( \mathcal{K} \) is a set of \( K \) topics, \( \mathcal{M} \) is a set of \( M \) document identifiers, and \( \mathcal{V} \) is a set of \( V \) identifiers for the vocabulary words. From this we obtain the update for line 5 of Algorithm 1

\[
p_z(k|s) \propto \frac{(D_{mk} + \alpha) \times W_{kv} + \beta}{T_k + \beta V}
\]

where \( D_{mk} = |\{z_{mn} | z_{mn} = k\}|, W_{kv} = |\{z_{mn} | w_{mn} = v, z_{mn} = k\}|, \) and \( T_k = \sum_{v=1}^{V} W_{kv}. \)

It reassuring to see that the boxed region of Equation 4.3 is similar to respective formulas in collapsed Gibbs sampling (CGS) T. L. Griffiths and Steyvers, 2004 and collapsed variational Bayes (CVBo) W. Y. Teh, Newman, and Welling, 2007. For LDA, ESCA implicitly performs SGD with Frank-Wolfe updates, alluding to a convergence rate.

### 4.1.2 Preliminary Experiments

**software & hardware** All algorithms are implemented in C++11. We implement multi-threaded 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 run our experiments on a small cluster of 4 nodes connected through 10Gb/s Ethernet. Each node has two 9-core Intel Xeon E5 processors for a total of 36 hardware threads per node.

**datasets** We employ on a large corpora of PubMed abstracts (141,043 vocabulary words, 8.2 million documents, 737 million tokens).

**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 in Fig. 4.2.

### 4.2 Reducing Lookups

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 because, for most inference methods, we need to evaluate all probabilities whereas search only needs the best instance. Specifically, many latent variable models, e.g., Gaussian mixture models (GMM), have the structure of the form:

\[
p(x) = \sum_z p(z)p(x|\theta_z)
\]  

(4.4)

where \(x\) denotes observed variables, \(z\) are discrete latent variables over a state space of size \(m\), and \(\theta_z\) parameters of the conditional. 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 cost \(O(m)\), which can be expensive especially when the number of latent classes is huge. We want to explore an inference technique to address this issue by marrying a fast lookup structure with an adaptive rejection sampler, so that not all of the state spaces needs to be explored.

### 4.2.1 Latent Variable Models

The key motivation for this work is to make inference in latent variable models more efficient. As expressed in (4.4), we consider latent models which have mixtures of exponential family. We first assume that updates to \(p(z)\) can be carried out by modifying \(O(1)\) values at any given time. Second, the conditional \(p(x|\theta)\) in (4.4) is assumed to be in the exponential family, i.e.,

\[
p(x|\theta) = \exp(\langle \phi(x), \theta \rangle - g(\theta)).
\]  

(4.5)

Here \(\phi(x)\) represents the sufficient statistics and \(g(\theta)\) is the (normalizing) log-partition function. This can be rewritten as follows:

\[
p(x|\theta) = e^{\langle \phi(x), -1 \rangle, \theta, g(\theta)} = e^{\langle \tilde{\phi}(x), \tilde{\theta} \rangle}
\]  

(4.6)

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 \(z \in \mathcal{Z}\) respectively.

### 4.2.2 Cover Trees

Cover Trees Beygelzimer, Kakade, and Langford, 2006 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).

Cover trees are defined as an infinite succession of levels \(S_i\) with \(i \in \mathcal{Z}\). 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 \(x\).
Now we introduce notation and explain details of our approach when the number of clusters is (a) moderate and (b) large. 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 \( j \) 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 \( \bar{x} \) (Sec. 4.2.3).

**Cluster tree** (\( T_C \)): Similarly, \( T_C \) is the cover tree generated with cluster parameters \( \bar{\theta}_z \). For simplicity, we assume that the expansion rates of clusters and data are both \( c \).

### 4.2.3 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. 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 \( S_j \) is \( O(m) \).
2. For each of the prototypes \( \bar{x} \), which are members of \( S_j \), compute \( p(z|\bar{x}) \) using the alias method to draw from \( m \) components \( \bar{\theta}_z \). 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 rejection sampling using the draws from \( p(z|\bar{x}) = q(z) \) as proposal. We accept a sample \( z \sim q(\cdot) \) with probability

   \[
   \pi := \frac{p(z|\bar{x})}{p(z|x)}.
   \]

The key reason why this algorithm has a useful acceptance probability is that the normalizations for \( p(z|x) \) and \( p(z|\bar{x}) \) cancel out. Expected number of rejections can be bounded as

\[
\mathbb{E}[\#\text{Rejections}] = \max \frac{p(z|x)}{p(z|\bar{x})} \leq \exp \left\{ \| \phi(x) - \phi(\bar{x}) \| \| \bar{\theta}_z \| \right\} \leq e^{2^{j+1}L}
\]

for \( \| \bar{\theta}_z \| \leq L \). 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., \( \| \phi(x) - \phi(\bar{x}) \| \leq 2^{j+1}. \)

### 4.2.4 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 or the overhead in constructing the alias sampler dominates. To overcome this issue, we utilize the cover tree \( T_C \) on the cluster parameters as well. The sampling procedure is detailed below.

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 in a rejection sampler. Before we delve into details, let us consider a simple case without \( T_C \). If we are able to approximate \( p(x|\bar{\theta}_z) \) by some \( q_z \) such that

\[
e^{-e} p(x|\bar{\theta}_z) \leq q_z \leq e^e p(x|\bar{\theta}_z) \tag{4.8}
\]
for all $z$, then it follows that a sampler drawing $z$ from
\begin{equation}
  z \sim \frac{q_z p(z)}{\sum z' q_{z'} p(z')} \tag{4.9}
\end{equation}
and then accepting with probability $e^{-\epsilon} q_z^{-1} p(x|\theta_z)$ will draw from $p(z|x)$. 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 $i$ and compute normalization at accuracy level $l$:
\begin{equation}
  \gamma_0 := \sum_{z \in S_l} \beta(i, z) \exp \left( \bar{\theta}_z, \bar{\phi}(x) \right). \tag{4.10}
\end{equation}

2. Set $e^{-\epsilon} := e^{-2\|\bar{\phi}(x)\|}$ as multiplier for the acceptance threshold and $\gamma := e^\epsilon \gamma_0$.

3. Draw a node $z \in S_l$ with probability $\delta_z := \gamma^{-1} e^\epsilon \beta(i, z) \exp \left( \bar{\theta}_z, \bar{\phi}(x) \right)$.

4. Accept $z_i$ at the current level with probability $\pi := \gamma^{-1} \delta_{z_i}^{-1} p(z_i) \exp \left( \bar{\theta}_{z_i}, \bar{\phi}(x) \right)$.

5. For $i := i - 1$ down to $-\infty$ do
   a) Set $e^{-\epsilon} := e^{-2\|\bar{\phi}(x)\|}$ as the new multiplier and $\gamma := \delta_{z_{i+1}} (1 - \pi)$ as the normalizer.
   b) Draw one of the children $z$ of $z_{i+1}$ with probability $\delta_z := \gamma^{-1} e^\epsilon \beta(i, z) \exp \left( \bar{\theta}_z, \bar{\phi}(x) \right)$. Exit if we do not draw any of them (since $\sum_{z \in \text{ch}(z_{i+1})} \delta_z \leq 1$) and restart from step 2, else denote this child by $z_i$.
   c) Accept $z_i$ at the current level with probability $\pi := \gamma^{-1} \delta_{z_i}^{-1} p(z_i) \exp \left( \bar{\theta}_{z_i}, \bar{\phi}(x) \right)$. Do not include $z_{i+1}$ 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.

4.2.5 Preliminary Experiments

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. We compare our two samplers (Canopy I and Canopy II) with both the traditional Expectation Maximization (EM) Dempster, Laird, and Rubin, 1977 and the faster Stochastic EM through ESCA (ESCA) proposed in previous Sec. 4.1. 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.

**Software & Hardware** All the algorithms are implemented multithreaded in simple C + +11 using a distributed setup. We run our experiments on a cluster of 16 Amazon EC2 c4.8xlarge nodes connected through 10Gb/s Ethernet. There are 36 virtual threads and 60GB of memory per node. All data and calculations are carried out at double floating-point precision.

**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
Scalability

Figure 4.3: Showing scalability of per-iteration runtime of different algorithms with increasing dataset size. From Fig. 4.3a, 4.3b, and 4.3c 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 Fig. 4.3d. For instance, with \((n, m, d) = (32 \text{mil}, 4096, 1024)\), we see that there is a speed-up of \(150\times\) for a mere \(2\times\) memory overhead.

Fig. 4.3. We make the following observations: (1) Overall, Fig. 4.3 is in line with our claim that the proposed method reduced the per iteration complexity from \(O(nm)\) of EM/ESCA to \(\tilde{O}(n+m)\). (2) To illustrate this further, we consider \(n = O(m)\) and vary \(m\) (shown in Fig. 4.3c). 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 \(\tilde{O}(m+n)\), i.e., \(\tilde{O}(m)\). (3) However, there is no free lunch. The speed-up comes at cost of increased memory (for storing data-structures). For example, when \(n = 32\) mil, \(m = 4096\), and \(d = 1024\), a mere \(2\times\) increase in memory gets us a speed up of \(150\times\).

4.3 HANDLING VARIABLE STATE SPACE

The speed up techniques developed in previous sections and many of the strategies in prior literature assume the state space size to be fixed, which are well suited for parametric models like latent Dirichlet allocation (LDA). The key behind these speed-up techniques is to identify parameters that change relatively slowly during sampling (e.g. the location of cluster centres, the definition of topics) and appear as decoupled terms in the posterior with remaining portion being sparse. In this scenario, fast-to-sample proposals, leveraging sparsity and slowly varying terms, can be constructed that are very similar to the target distribution.

However, in many applications, it is desirable for the model complexity to be able to adapt to data size. In this regard, Bayesian nonparametrics have become popular, for which scalable inference more complex. First, the model does not decompose into local sparse terms and global, slowly varying terms. 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, as it is necessary for proposal in rejection or Metropolis-Hastings samplers to have a bigger support than the target target distribution. We explore how to extend techniques developed for speeding up can be extended to Bayesian nonparametrics like hierarchical Dirichlet process (HDP) Y. W. Teh et al., 2006.
4.3.1 Log Structured Alias Sampling

The resemblance of inference on HDP using the sampling by direct assignment (Y. W. Teh et al., 2006) 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.
34 scalability

Figure 4.4: Log structured alias table

\[
\Pr(z_{ji} = k|\text{rest}) \propto \begin{cases} 
\sum_{h} s_{hi}^{-ji} f_k^{-ji}(x_{ji}) + \alpha_0 \theta_0 f_k^{-ji}(x_{ji}) & \text{if } k \text{ in use} \\
\alpha_0 \theta_0 f_k^{-ji}(x_{ji}) & \text{if } k \text{ is new}
\end{cases}
\] (4.11)

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. As it is costly 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 sampling, inspired from the BigTables paper 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 4.4. Then once in a while 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 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.

4.3.2 Exploiting Sparsity

In case of Sampling by Direct Assignment, there is a partition of the posterior into a sparse local term and slowly varying global term similar to LDA. Whereas a faster mixing inference called Sampling by Table Configuration has been proposed in Chen, Du, and Buntine, 2011, but it has no straightforward decoupling. The probability of sitting at an existing table in the restaurant is sparse and depends only on counts within restaurants. However, the probability of starting a new table depends both on local and global counts and must be evaluated for all active topics, the number of which can be large. Specifically, the culprit is:

\[
\Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) \propto \frac{s_{m_{kj}}^{n_{kj}+1}}{s_{n_{kj}}^{n_{kj}+1}} \frac{m_{kj} + 1}{n_{kj} + 1} \frac{\alpha_0 m_k^2}{(m_k + 1)(m_k + \gamma)} f_k(x_{ji})
\] (4.12)
Observe that for topics which have no representation in the current restaurant, i.e., \( n_{kj} = 0, m_{kj} = 0 \), the probability becomes slowly varying. For topics which are already represented in the current restaurant, i.e., \( n_{kj} > 0, m_{kj} > 0 \) we were able to come up with a new decoupling of the sparse and slowly varying terms. For which we needed to first prove the following:

**Lemma 3** Stirling number of first kind satisfy the following inequality for all \( 0 \leq m \leq n \):

\[
\frac{m + 1}{n + 1} \frac{S_{n+1}^{m+1}}{S_{m+1}^{n+1}} \geq 1
\]  

(4.13)

Thus, with careful mathematical manipulation, even in this formulation of HDP the sparse and slowly varying terms are decoupled as desired.

### 4.3.3 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.

- Draw auxiliary variables \( b_{lj} \sim \text{Bernoulli} \left( \frac{\alpha_0 \theta_k}{\alpha_0 \theta + l - 1} \right) \) for \( l \in \{1, \ldots, n_{kj}\} \).
- A sample \( m \) from the Antoniak distribution can be computed by \( m = \sum_l b_{lj} \).

**Lemma 4** Denote by \( \mathcal{A}(\theta, \theta) \) the Antoniak distribution. That is, it denotes the distribution where \( p(x = m) \propto S_m^n \theta^m \) for \( m \in \{1, \ldots, m\} \) and where \( S_m^n \) is the Stirling number of the first kind. Moreover, assume that we have \( m \) Bernoulli random variables \( b_l \), drawn from \( b_l \sim \text{Bin} \left( \frac{\theta}{\theta + l - 1} \right) \) for \( l \in \{1, \ldots, n\} \). Then \( x = \sum_{l=1}^n b_l \) satisfies \( x \sim \mathcal{A}(\theta, \theta) \). Hence sampling from \( \mathcal{A}(\theta, \theta) \) can be accomplished in \( O(n) \) time and \( O(1) \) space.

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 \( S_m^n \) 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. But such implementation can be slow. Fortunately, as identified in Buntine and Hutter, 2010 for sampling of hierarchical models we only need the Stirling numbers as consecutive ratios. At every sampling instance of HDP/HPDP we can change \( n \) or \( m \) by at most \( \pm 1 \). So we can cache a few rows of Stirling numbers around the present value. If during sampling we move towards the boundary we can re-compute some more rows and discard values at the other end. The other ratios can be readily expressed in terms of \( U_m^n \).

### 4.3.4 Preliminary Experiments

With all these paraphernalia in place, we evaluate our improved inference methods for HDP.

**Software & Hardware** All algorithms are implemented in C++11. We implement multithreaded parallelization using the work-stealing Fork/Join framework. We implement log structured alias stables with the array doubling trick.

**Datasets** We employ two datasets: PubMed abstracts (141,043 vocabulary words, 8.2 million documents, 737 million tokens), and a smaller Enron (28,102 words, 40 thousand documents, and 6 million tokens).
Figure 4.5: On left two figures we show evolution of log likelihood on Pubmed dataset over number of iterations and time. Existing methods: SDA (Sampling by Direct Assignment), STC (Sampling Table Configuration). Proposed Methods: Alias HDP and SSTC HDP. On right, we compare the log likelihood achieved by HDP with LDA by sweeping the number of topic. We see HDP produces log-likelihood as good as the best LDA model.

**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. The hyper parameters are set as $\alpha = 50/K$ and $\beta = 0.1$. The results are in Figure 4.5.

### 4.4 Escaping Saddle Points

In the case of learning deep representations, optimization algorithms based on first-order information, like the popular stochastic gradient descent (Sgd), are typically favored as they are relatively inexpensive and scale seamlessly with respect to number of data points. A major obstacle in its scalability over modern computational resources consisting of multi-gpu and multi-machine is inability of first order methods to perform well under large mini-batch setting for nonconvex problems. 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, e.g. the cubic regularization (CR) method Nesterov and Polyak, 2006. However, most works on second-order methods rely extensively on expensive Hessian-based computations, making them impractical in large-scale settings. Another work Ge et al., 2015 shows a noisy variant of Sgd can converge close to a local minimum, under some conditions. 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.

To tackle this challenge, we want to explore a strategy that focuses on using Hessian information (or its structure) whenever the method gets stuck at stationary points that are not second-order critical, else utilize only first order information. The idea is illustrated in Fig. 4.6.

#### 4.4.1 Problem Setup

We study nonconvex finite-sum problems of the form

$$
\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x),
$$

(4.14)
We propose a generic framework for escaping saddle points while solving nonconvex problems of form (4.14). To evade saddle 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 focuses on using gradient information for decreasing the function. On its own, the Gradient-Focused-Optimizer might not converge to a second order critical point since it can get stuck at a saddle point. Hence, we require the subroutine Hessian-Focused-Optimizer to help avoid saddle points. We design a procedure that interleave these subroutines to obtain a second-order critical point, which not only provides meaningful theoretical guarantees, but also translates into strong empirical gains. Algorithm 2 provides pseudocode of our framework. Observe that the algorithm is still abstract, and 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 \(g : \mathbb{N} \times \mathbb{R}^+ \rightarrow \mathbb{R}^+\), such that

Figure 4.6: 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.
Algorithm 2 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 \(u^t\) with probability \(p\) and \(z^t\) with probability \(1 - p\)
5: \((x^{t+1}, z^{t+1}) = \text{Hessian-Focused-Optimizer}(u^t, \epsilon, \gamma)\) (refer to H.1 and H.2)
6: if \(t^{t+1} = \emptyset\) then
7: Output set \(\{x^{t+1}\}\)
8: end if
9: end for
10: Output set \(\{y^1, ..., y^T\}\)

G.1 \(\mathbb{E}[f(y)] \leq f(x)\),

G.2 \(\mathbb{E}[\|\nabla f(y)\|^2] \leq \frac{1}{g(n, \epsilon)} \mathbb{E}[f(x) - f(z)]\).

Here the outputs \(y, z \in \mathbb{R}^d\). 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 2. 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 \(y \in \mathbb{R}^d\) and \(\tau \in \{\emptyset, \diamond\}\). If \(\tau = \emptyset\), then \(y\) is a \((\epsilon, \gamma)\)-second order critical point with probability at least \(1 - q\). Otherwise if \(\tau = \diamond\), then \(y\) satisfies the following condition:

\(\text{H.1 } \mathbb{E}[f(y)] \leq f(x),\)

\(\text{H.2 } \mathbb{E}[f(y)] \leq f(x) - h(n, \epsilon, \gamma)\) when \(\lambda_{\text{min}}(\nabla^2 f(x)) \leq -\gamma\) for \(h : \mathbb{N} \times \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+\).

Here the expectation is over any randomness in subroutine. The two conditions ensure that the objective function value, in expectation, never increases and decreases with a certain rate when \(\lambda_{\text{min}}(\nabla^2 f(x)) \leq -\gamma\). 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 2 and hence, needs to be invoked judiciously.

4.4.3 Convergence Analysis

Theorem 7 Let \(\Delta = f(x^0) - B\) and \(\theta = \min((1 - p)\epsilon^2 g(n, \epsilon), ph(n, \epsilon, \gamma))\). Also, let set \(\Gamma\) be the output of Algorithm 2 with **Gradient-Focused-Optimizer** satisfying G.1 and G.2 and **Hessian-Focused-Optimizer** satisfying H.1 and H.2. Furthermore, \(T\) be such that \(T > \Delta/\theta\). Suppose the multiset \(S = \{i_1, ..., i_k\}\) are \(k\) indices selected independently and uniformly randomly from \(\{1, ..., |\Gamma|\}\). Then the following holds for the indices in \(S\):

1. \(y^t\), where \(t \in \{i_1, ..., i_k\}\), is a \((\epsilon, \gamma)\)-critical point with probability at least \(1 - \max(\Delta/(T\theta), q)\).

2. If \(k = O((\log(1/\zeta)/\min(\log(\Delta/(T\theta)), \log(1/q)))\), with at least probability \(1 - \zeta\), at least one iterate \(y^t\) where \(t \in \{i_1, ..., i_k\}\) is a \((\epsilon, \gamma)\)-critical point.

The key point regarding the above result is that the overall convergence rate depends on the magnitude of both functions \(g\) and \(h\). It shows that the slowest amongst the subroutines **Gradient-Focused-Optimizer** and **Hessian-Focused-Optimizer** governs the overall rate of Algorithm 2. Thus, it is important to ensure that both these procedures have good conver-
Figure 4.7: Comparison of various methods on CURVES and MNIST Deep Autoencoder. Our mix approach converges faster than the baseline methods and uses relatively few ISO calls, which are practically relatively expensive to IFO calls, in comparison to APPROXCUBICDEScent.

Figure 4.7: Comparison of various methods on CURVES and MNIST Deep Autoencoder. Our mix approach converges faster than the baseline methods and uses relatively few ISO calls, which are practically relatively expensive to IFO calls, in comparison to APPROXCUBICDEScent.

gence. Also, note that the optimal setting for $p$ based on the result above satisfies $1/p = 1/e^2g(n, \epsilon) + 1/h(n, \epsilon, \gamma)$.

For example if we use SVRG as the GRADIENT-FOCUSED-OPTIMIZER and HESSIANDESCRIPT as the HESSIAN-FOCUSED-OPTIMIZER, we can show the following result:

**Theorem 8 [Informal]** The overall running time of Algorithm 2 to find a $(\epsilon, \sqrt{\epsilon})$-second order critical point, with SVRG is used as GRADIENT-FOCUSED-OPTIMIZER and HESSIANDESCRIPT is used as HESSIAN-FOCUSED-OPTIMIZER, is $O(nd/e^{3/2} + n^{3/4}d/e^{7/4} + n^{2/3}d/e^2)$.

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 et al., 2016; Carmon et al., 2016 in their respective settings, but with a much simpler algorithm and analysis. We also note that our algorithm is faster than Jin et al., 2017, which has a time complexity of $O(nd/e^2)$.

**Practical considerations** For HESSIAN-FOCUSED-OPTIMIZER, we found stochastic, adaptive and inexact approaches for solving HESSIANDESCRIPT and CUBICDESCRIPT work well in practice. Furthermore, 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.

### 4.4.4 Preliminary Experiments

To investigate the practical performance of the framework, we applied it to two deep autoencoder optimization problems from G. E. Hinton and Salakhutdinov, 2006 called “CURVES” and “MNIST”. Due to their high difficulty, performance on these problems has become a standard benchmark for neural network optimization methods, e.g. Martens, 2010; Martens and Grosse, 2015; Sutskever et al., 2013; Vinyals and Povey, 2012.¹

As an instantiation of our framework, we use a mix of ADAM and APPROXCUBICDESCRIPT. 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 results are presented in Fig. 4.7, which shows 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.

¹ Data available at: [www.cs.toronto.edu/~jmartens/digs3pts_1.mat, mnist_all.mat](http://www.cs.toronto.edu/~jmartens/digs3pts_1.mat, mnist_all.mat)
My Other Research

Besides representation learning, I have also worked on a number of other topics such as random graphs, small sample estimation, spectral methods, reinforcement learning, which are not part of my thesis. Here, I briefly describe my contributions in these areas.

5.1 RANDOM GRAPHS

Preferential attachment models for random graphs are successful in capturing many characteristics of real networks such as power law behavior. At the same time they lack flexibility to take vertex to vertex affinities into account, a feature that is commonly used in many link recommendation algorithms. We propose a random graph model based on both node attributes and preferential attachment. This approach overcomes the limitation of existing models on expressing vertex affinity and on reflecting properties of different subgraphs. We analytically prove that our model preserves the power law behavior in the degree distribution as expressed by natural graphs and we show that it satisfies the small world property. Experiments show that our model provides an excellent fit of many natural graph statistics.

5.2 SMALL SAMPLE ESTIMATION

Today’s VLSI circuits are designed via a multi-stage flow. At each stage, simulation or measurement data are collected to validate the circuit design, before moving to the next stage. These measurements are extremely time consuming and expensive, hence only a few samples at each stage can be afforded. Traditional performance modeling techniques rely on the data collected at a single stage only and completely ignore the data that are generated at other stages. The key idea proposed was to reuse the early-stage data when fitting a late-stage performance model. As such, the performance modeling cost can be substantially reduced. Mathematically, the proposed method is casted as a hierarchical Bayesian model with transfer learning. This approach results in requiring very few late-stage samples points to produce reliable performance models. On many tasks, our approach demonstrates up to 9× reduction in validation time which can be translated to significant reduction of cost.

5.3 SPECTRAL METHODS

Nonparametric models are versatile, albeit computationally expensive, tool for modeling mixture models. We introduce spectral methods for the two most popular nonparametric models: the Indian Buffet Process (IBP) and the Hierarchical Dirichlet Process (HDP). We show proposed methods for the inference of nonparametric models are computationally and statistically efficient. In particular, we derive the lower-order moments, propose spectral algorithms for both models, and provide reconstruction guarantees for the algorithms. For the HDP, we further show that applying hierarchical models on dataset with hierarchical structure, which can be solved with the generalized spectral HDP, produces better solutions than flat models.
Publications


Zaheer, Manzil, Jean-Baptiste Tristan, et al. (2016). “Learning a Static Analyzer: A Case Study on a Toy Language.” In:


Bisk, Yonatan et al. (2016). “Evaluating Induced CCG Parsers on Grounded Semantic Parsing.” In: EMNLP.


Bordes, Antoine et al. (2015). “Large-scale Simple Question Answering with Memory Networks.” In: CoRR.


Deng, Jia et al. (2009). “Imagenet: A large-scale hierarchical image database.” In: CVPR.


Miller, Alexander H. et al. (2016). “Key-Value Memory Networks for Directly Reading Documents.” In: In EMNLP.


Sukhbaatar, Sainbayar et al. (2015). “End-To-End Memory Networks.” In: *NIPS.*


Wang, Limin et al. (2016). “Temporal segment networks: Towards good practices for deep action recognition.” In: *ECCV.*


Zhang, Bowen et al. (2016). “Real-time action recognition with enhanced motion vector CNNs.” In: *CVPR.*