Inference & Introspection in Deep Generative Models of Sparse Data

Rahul G. Krishnan
New York University
rahul@cs.nyu.edu

Matthew Hoffman
Adobe Research
mathoffm@adobe.com

Abstract

Deep generative models such as deep latent Gaussian models (DLGMs) are powerful and popular density estimators. However, they have been applied almost exclusively to dense data such as images; DLGMs are rarely applied to sparse, high-dimensional integer data such as word counts or product ratings. One reason is that the standard training procedures find poor local optima when applied to such data. We propose two techniques that alleviate this problem, significantly improving our ability to fit DLGMs to sparse, high-dimensional data. Having fit these models, we are faced with another challenge: how to use and interpret the representation that we have learned? To that end, we propose a method that extracts distributed representations of features via a simple linearization of the model.

1 Introduction

Deep latent Gaussian models (DLGMs, a.k.a. variational autoencoders; Rezende *et al.* , 2014; Kingma *et al.* , 2014) have led a resurgence in the use of deep generative models for density estimation. DLGMs assume that observed vectors x are generated by applying a nonlinear transformation (defined by a neural network with parameters θ) to a vector of Gaussian random variables z.

Learning in DLGMs proceeds by approximately maximizing the average marginal likelihood $p(x) \equiv \int_z p(z)p(x|z)dz$ of the observations x. Computing the true marginal likelihood is intractable, so we resort to variational expectation-maximization (Bishop, 2006), an approximation to maximum-likelihood estimation. To learn the parameters θ of the generative model, the procedure needs to find a distribution $q(z|x)$ that approximates the posterior distribution $p(z|x)$ of the latent vector z given the observations x. In the past, such q distributions were fit using iterative optimization procedures (e.g., Hoffman *et al.* , 2013). But Rezende *et al.* (2014) and Kingma *et al.* (2014) showed that $q(z|x)$ can be parameterized by a feedforward "inference network" with parameters ϕ, speeding up learning. Embedded within this procedure, however, lies a potential problem: both the inference network and the generative model are initialized randomly. Early on in learning, the inference network's $q(z|x)$ distributions will be poor approximations to the true posterior $p(z|x)$, and the gradients used to update the parameters of the generative model will therefore be poor approximations to the gradients of the true log-likelihood $\log p(x)$. Previous stochastic variational inference methods (Hoffman *et al.* , 2013) had for every data-point, a set of variational parameters that were optimized within the inner loop of learning. Here, we investigate blending the two methodologies for learning models of sparse data. In particular, we use the parameters predicted by the inference network as an initialization and optimize them further during learning. When modeling high-dimensional sparse data, we show that updating the local variational parameters yields generative models with better held-out likelihood, particularly for deeper generative models.

We consider a simple method to interpret *what* is being learned by generative models such as DLGMs whose conditional probabilities are parameterized by deep neural networks. We use the Jacobian

AABI Workshop at the 29th Conference on Neural Information Processing Systems (NIPS 2016), Spain.

of the conditional distribution with respect to latent variables in the Bayesian network to form embeddings (or Jacobian vectors) of the observations.

2 Background

Generative Model: We instantiate our graphical model where our observations are bag-of-words documents. We observe a set of D word count vectors $x_{1:D}$, where x_{dv} denotes the number of times that word index $v \in \{1, \ldots, V\}$ appears in document d. We assume we are given the total number of words per document $N_d \equiv \sum_v x_{dv}$, and that x was generated via the following generative process:

$$z_d \sim \mathcal{N}(0, I); \quad \gamma(z) \equiv \text{MLP}(z; \theta); \quad \mu(z) \equiv \frac{\exp\{\gamma(z)\}}{\sum_v \exp\{\gamma(z)_v\}}; \quad x_d \sim \text{Multinomial}(\mu(z_d), N_d). \tag{1}$$

That is, we draw a Gaussian random vector, pass it through a multilayer perceptron (MLP) with parameters θ, pass the resulting vector through the softmax (a.k.a. multinomial logistic) function, and sample N_d times from the resulting distribution over the vocabulary.[1]

Variational Learning: We need to approximate the intractable posterior distribution $p(z|x)$ during learning. Using the well-known variational principle, we can obtain the lower bound on the log marginal likelihood of the data (or $\mathcal{L}(x; \theta, \phi)$). We leverage an *inference network* or *recognition network* (Hinton *et al.* , 1995), a neural network which approximates the intractable posterior, during learning. With a normal distribution as our variational approximation we have that $q_\phi(z|x) \sim \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$. $\mu_\phi(x), \Sigma_\phi(x)$ are functions of the observation x, and we denote by $\psi(x) := \{\mu_\phi(x), \Sigma_\phi(x)\}$ the local variational parameters predicted by the inference network.

3 Methodology

Inference with Global Information: The simplest way to incorporate global first order statistics across the training data into the inferential process is to condition on tf-idf (Baeza-Yates *et al.* , 1999) features instead of the raw-counts. tf-idf is one of the most widely used techniques in information retrieval. The key idea behind it is to re-weight features in a manner that increases the influence of rarer words while decreasing the influence of common words that appear in all documents. We define the tf-idf-transformed word-count vector $\tilde{x}_d$ as

$$\tilde{x}_{dv} \equiv x_{dv} \log \tfrac{D}{\sum_{d'} \min\{x_{d'v}, 1\}}. \tag{2}$$

After applying this tf-idf transform, the resulting vector $\tilde{x}$ is normalized by its L2 norm.

Optimizing Local Variational Parameters: The predictions of the inference network early in optimization are suboptimal variational parameters used to derive gradients of the parameters of the generative model. This induces noise and bias to the gradients used to update the parameters of the generative model. To avoid these issues, we only use the local variational parameters $\psi(x)$ predicted by the inference network to initialize an iterative optimizer that maximizes the ELBO with respect to ψ (we denote by M the number of optimization steps performed); we use the optimized variational parameters $\hat{\psi(x)}$ to derive gradients for the generative model. We then train the inference network using stochastic backpropagation and gradient descent, holding the parameters of the generative model θ fixed. We refer the reader to Algorithm 1 in the supplementary material for an overview of the learning algorithm.

Introspection: In DLGMs, the relationship between latent variables z and observations x cannot be quickly read off of the parameters θ. But we can still ask what happens if we perturb z by some small dz—this is simply the directional derivative $\frac{\partial \mathbb{E}[x|z]}{\partial z} dz$. We can interpret this Jacobian matrix in much the same way we would a factor loading matrix (Spearman, 1904).

Jacobian Vectors: We evaluate the Jacobian matrix as: $\mathcal{J}(z)^{\log} = \frac{\partial \log \mu(z)}{\partial z}$ For any z, $\mathcal{J}(z)^{\log} \in \mathbb{R}^{V \times K}$ where K is the latent dimension and V is the dimensionality of the observations. It is

[1]In keeping with common practice, we neglect the multinomial base measure term $\frac{N!}{x_1! \cdots x_V!}$, which amounts to assuming that the words are observed in a particular order.

this matrix that we use to form embeddings. $\mathcal{J}(z)$ is a function of z leaving open the choice of where to evaluate this function. The semantics of our generative model suggest a natural choice: $\mathcal{J}^{\text{log}}_{\textbf{mean}} := \mathbb{E}_{p(z)}[\mathcal{J}(z)^{\text{log}}]$. This set of embeddings captures the variation in the output distribution with respect to the latent state across the prior distribution of the generative model. For implementations of generative models in frameworks that support automatic differentiation (Theano Development Team, 2016), $\mathcal{J}(z)$ is readily available and we estimate $\mathcal{J}^{\text{log}}_{\textbf{mean}}$ via Monte-Carlo sampling from the prior.

4 Related Work

Learning in Deep Generative Models: For DLGMs, Hjelm *et al.* (2016) also consider the optimization of the local variational parameters, though their exposition focuses on deriving an importance-sampling-based bound to use during learning in deep generative models with discrete latent variables. Their experimental results suggest the procedure does not improve performance much on the binarized MNIST dataset. This is consistent with our experience—we found that our secondary optimization procedure helped more when modeling sparse, high-dimensional count data.

Introspection: In the context of discriminative modeling, (Erhan *et al.* , 2009) use gradient information to study the patterns with which neurons are activated in a deep neural networks while (Wang *et al.* , 2016) use the spectra of the Jacobian to study the complexity of the functions learned by neural networks. Miao *et al.* (2016) learn a shallow log-linear model on text data and obtain embeddings for words from the weight matrix that parameterize their generative model.

5 Evaluation

We study the effect of further optimization of the variational parameters and inference with tf-idf features on the two datasets of varying size: the smaller 20Newsgroups (Lang, 2008) and the larger RCV2 (Lewis *et al.* , 2004) dataset.

Training Procedure: On all datasets, we train shallow log-linear models ($\gamma(z) = Wz + b$) and deeper three-layer DLGMs ($\gamma(z) = \text{MLP}(z;\theta)$). We vary the number of secondary optimization steps $M = 1, 200$ to study the effect of optimization on $\psi(x)$ with ADAM (Kingma & Ba, 2015). We use a mini-batch size of 500, a learning rate of 0.01 for $\psi(x)$ and 0.0008 for θ, ϕ. The inference network was fixed to a two-layer MLP whose intermediate hidden layer $h(x)$ was used to parameterize the mean and diagonal log-variance $\mu(x), \log\Sigma(x)$. To evaluate the quality of the learned generative models, we report an upper bound on perplexity (Mnih & Gregor, 2014). The notation 3-M100-tfidf indicates a model where the MLP parameterizing $\gamma(z)$ has three hidden layers, the local variational parameters are updated 100 times before an update of θ and tf-idf features were used in inference.

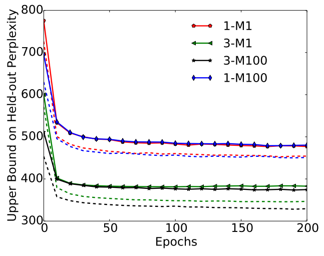
Improving Learning: Table 1 depicts our results on 20newsgroups and RCV2. On the smaller dataset, we find that the deeper models overfit quickly and are outperformed by shallow generative models. On the larger datasets, the deeper models' capacity is more readily utilized yielding better generalization. The use of tf-idf features helps learning on smaller datasets though on large datasets, the benefits are smaller when we also optimize $\psi(x)$. Finally, the optimization of the local variational parameters appears to help most on the larger datasets. To investigate how this occurs, we plot the held-out likelihood versus epochs, on models trained on the larger RCV2 (Figure 1a) and Wikipedia (Figure 1b) datasets. In the presence of large amounts of data, the larger deep generative models appear to converge to better solutions through the additional optimization of $\psi(x)$.

Jacobian Vectors: The Jacobian matrix can be used to visualize how much of the latent dimension is being utilized by the generative models learned using the inferential procedure we propose since it precisely encodes how sensitive the outputs are with respect to the inputs. The singular value spectrum of this matrix therefore, directly captures the amount of variance in the data explained by the latent space. In Figure 1c, 1d, we see that for the deeper models continuing to optimize the variational parameters allows us to learn models that use many more of the available latent dimensions. This suggests that, when fit to text data, DLGMs may be particularly susceptible to the overpruning phenomenon noted by Burda *et al.* (2015). In Figure 1, the lower held-out perplexity and the increased utilization of the latent space suggest that the continued optimization of the variational parameters yields more powerful generative models.

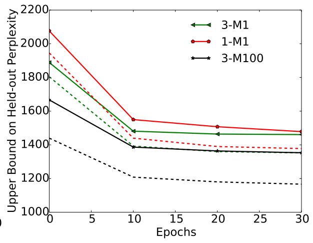
Table 1: **Test Perplexity: Left: Baselines** Results on the 20newsgroups and RCV1-v2 dataset Legend: LDA (Blei *et al.* , 2003), Replicated Softmax (RSM) (Hinton & Salakhutdinov, 2009), Sigmoid Belief Networks (SBN) and Deep Autoregressive Networks (DARN) (Mnih & Gregor, 2014), Neural Variational Document Model (Miao *et al.* , 2016). K denotes the latent dimension in our notation. **Right:** DLGMs on text data with $K = 100$. We vary the features presented to the inference network $q_\phi(z|x)$ during learning between: normalized count vectors ($\frac{x}{\sum_{i=1}^{V} x_i}$, denoted "norm") and normalized tf-idf (denoted "tf-idf") features.

Model	K	**20News**	**RCV1-v2**
LDA	50	1091	1437
LDA	200	1058	1142
RSM	50	953	988
SBN	50	909	784
fDARN	50	917	724
fDARN	200	—	598
NVDM	50	836	563
NVDM	200	852	550

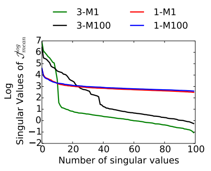
DLGM	**20News**		**RCV1-v2**	
	M1	M100	M1	M11
1-M1-norm	964	816	498	479
1-M100-norm	1182	831	485	453
3-M1-norm	1040	866	408	360
3-M100-norm	1341	894	378	329
1-M1-tfidf	895	**785**	475	453
1-M100-tfidf	917	792	480	451
3-M1-tfidf	1027	852	391	346
3-M100-tfidf	1029	833	377	**327**



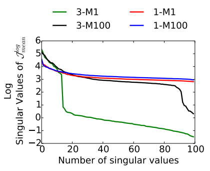
(a) RCV2



(b) Wikipedia



(c) RCV2



(d) Wikipedia

Figure 1: **Mechanics of Learning:** *Validation Perplexity and Log-singular Values of* $\mathcal{J}^{\log}_{\textbf{mean}}$*:* Best viewed in color. For the RCV2 and Wikipedia (large) datasets, we visualize the validation perplexity as a function of epochs. The solid lines indicate the validation perplexity for $M = 1$ and the dotted lines the indicate $M = 100$. The x-axis is **not** directly comparable on running times since larger values of M take longer during training. We find that learning with $M = 100$ takes approximately 15 times as long per mini-batch of size 500 on the text datasets. Figure 1c, 1d depict the sorted log singular values of $\mathcal{J}^{\log}_{\textbf{mean}}$.

Table 2: **Word Embeddings (Nearest Neighbors):** We visualize nearest neighbors of word embeddings. We exclude plurals of the query and other words in the neighborhood.

Query	**Neighborhood**
intelligence	espionage, secrecy, interrogation, counterterrorism
zen	dharma, buddhism, buddhas, meditation,yoga
artificial	artificially, molecules, synthetic, soluble
military	civilian, armys, commanders, infantry

We investigate how the Jacobian matrix may be used for model introspection by studying the qualitative properties of $\mathcal{J}^{\log}_{\textbf{mean}}$ on DLGMs (of type "3-M100-tfidf") trained on Wikipedia. We form a Monte Carlo estimate of $\mathcal{J}^{\log}_{\textbf{mean}}$ using 400 samples. The cosine distance is used to define neighbors of words in the embedding space of the Jacobian. In Table 2, we visualize some of the nearest neighbors of words using $\mathcal{J}^{\log}_{\textbf{mean}}$ obtained from models trained on the Wikipedia dataset. The neighbors are semantically sensible indicating that the method holds promise to derive distributed representations from trained variational auto-encoders.

References

Baeza-Yates, Ricardo, Ribeiro-Neto, Berthier, *et al.* . 1999. *Modern information retrieval*. Vol. 463. ACM press New York.

Bishop, C. 2006. *Pattern Recognition and Machine Learning*. Springer New York.

Blei, David M, Ng, Andrew Y, & Jordan, Michael I. 2003. Latent dirichlet allocation. *JMLR*.

Burda, Yuri, Grosse, Roger, & Salakhutdinov, Ruslan. 2015. Importance weighted autoencoders. *In: ICLR*.

Erhan, Dumitru, Bengio, Yoshua, Courville, Aaron, & Vincent, Pascal. 2009. Visualizing higher-layer features of a deep network.

Hinton, Geoffrey E, & Salakhutdinov, Ruslan R. 2009. Replicated softmax: an undirected topic model. *In: NIPS*.

Hinton, Geoffrey E, Dayan, Peter, Frey, Brendan J, & Neal, Radford M. 1995. The" wake-sleep" algorithm for unsupervised neural networks. *Science*.

Hjelm, R Devon, Cho, Kyunghyun, Chung, Junyoung, Salakhutdinov, Russ, Calhoun, Vince, & Jojic, Nebojsa. 2016. Iterative Refinement of Approximate Posterior for Training Directed Belief Networks. *In: NIPS*.

Hoffman, Matthew D, Blei, David M, Wang, Chong, & Paisley, John William. 2013. Stochastic variational inference. *JMLR*.

Kingma, Diederik, & Ba, Jimmy. 2015. Adam: A method for stochastic optimization. *In: ICLR*.

Kingma, Diederik P, Mohamed, Shakir, Rezende, Danilo Jimenez, & Welling, Max. 2014. Semi-supervised learning with deep generative models. *In: NIPS*.

Lang, Ken. 2008. *The 20 newsgroups data set*.

Lewis, David D, Yang, Yiming, Rose, Tony G, & Li, Fan. 2004. RCV1: A new benchmark collection for text categorization research. *JMLR*.

Miao, Yishu, Yu, Lei, & Blunsom, Phil. 2016. Neural Variational Inference for Text Processing. *In: ICML*.

Mnih, Andriy, & Gregor, Karol. 2014. Neural variational inference and learning in belief networks. *In: ICML*.

Rezende, Danilo Jimenez, Mohamed, Shakir, & Wierstra, Daan. 2014. Stochastic backpropagation and approximate inference in deep generative models. *In: ICML*.

Spearman, Charles. 1904. " General Intelligence," objectively determined and measured. *The American Journal of Psychology*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*.

Wang, Shengjie, Plilipose, Matthai, Richardson, COM Matthew, Geras, COM Krzysztof, Urban, Gregor, & Aslan, EDU Ozlem. 2016. Analysis of Deep Neural Networks with the Extended Data Jacobian Matrix. *In: ICML*.

Supplementary Material

Algorithm 1 presents the learning algorithm outlined in the main paper.

Algorithm 1 **Pseudocode for Learning:** We evaluate expectations in $\mathcal{L}(x)$ using a single sample from the variational distribution and aggregate gradients across mini-batches. $M = 1$ corresponds to performing no additional optimization to the variational parameters We update $\theta, \psi(x), \phi$ using stochastic gradient descent with adaptive learning rates $\eta_\theta, \eta_{\psi(x)}, \eta_\phi$ obtained via ADAM (Kingma & Ba, 2015)

```
Inputs: Dataset $\mathcal{D} := [x_1, \ldots, x_D]$, Inference Model: $q_\phi(z|x)$, Generative Model: $p_\theta(x|z), p(z)$
while notConverged() do
  1. Sample datapoint: $x \sim \mathcal{D}$
  2. Estimate local variational parameters $\psi(x)_1$ using $q_\phi(z|x)$
  3. Estimate $\psi(x)_M \approx \hat{\psi(x)} = \arg\max_{\psi(x)} \mathcal{L}(x; \theta; \psi(x))$ via SGD as:
       $m = 1, \ldots, M, \psi(x)_{m+1} = \psi(x)_m + \eta_{\psi(x)} \frac{\partial \mathcal{L}(x; \theta, \psi(x)_m)}{\partial \psi(x)_m}$
  4. Update $\theta$ as: $\theta \leftarrow \theta + \eta_\theta \nabla_\theta \mathcal{L}(x; \theta, \psi(x)_M)$
  5. Update $\phi$ as: $\phi \leftarrow \phi + \eta_\phi \nabla_\phi \mathcal{L}(x; \theta, \psi(x)_1)$
end while
```