On the challenges of learning with inference networks on sparse, high-dimensional data

Rahul G. Krishnan
MIT

Dawen Liang
Netflix

Matthew D. Hoffman
Google

Abstract

We study parameter estimation in Nonlinear Factor Analysis (NFA) where the generative model is parameterized by a deep neural network. Recent work has focused on learning such models using inference (or recognition) networks; we identify a crucial problem when modeling large, sparse, high-dimensional datasets – underfitting. We study the extent of underfitting, highlighting that its severity increases with the sparsity of the data. We propose methods to tackle it via iterative optimization inspired by stochastic variational inference [Hoffman et al., 2013] and improvements in the data representation used for inference. The proposed techniques drastically improve the ability of these powerful models to fit sparse data, achieving state-of-the-art results on a benchmark text-count dataset and excellent results on the task of top-N recommendation.

1 Introduction

Factor analysis [FA, Spearman, 1904] is a widely used latent variable model in the applied sciences. The generative model assumes data, x, is a linear function of independent normally distributed latent variables, z. The assumption of linearity has been relaxed in nonlinear factor analysis (NFA) [Gibson, 1960] and extended across a variety of domains such as economics [Jones, 2006], signal processing [Jutten and Karhunen, 2003], and machine learning [Valpola and Karhunen, 2002, Lawrence, 2003]. NFA assumes the joint distribution factorizes as $p(x, z; \theta) = p(z)p(x|z; \theta)$. When the nonlinear conditional distribution is parameterized by deep neural networks [e.g., Valpola and Karhunen, 2002], the graphical model is referred to as a deep generative model. When paired with an inference network [Hinton et al., 1995], a parametric function that approximates posterior distribution $p(z|x)$ (local variational parameters) from data, such models go by the name of variational autoencoders [VAEs, Kingma and Welling, 2014, Rezende et al., 2014]. We study NFA for the density estimation of *sparse*, high-dimensional data.

Sparse, high-dimensional data is ubiquitous; it arises naturally in survey and demographic data, bag-of-words representations of text, mobile app usage logs, recommender systems, genomics, and electronic health records. Such data are characterized by few frequently occurring features and a long-tail of rare features. When directly learning VAEs on sparse data, a problem we run into is that the standard learning algorithm results in *underfitting* and fails to utilize the model's full capacity. The inability to fully utilize the capacity of complex models in the large data regime is cause for concern since it limits the applicability of this class of models to problems in the aforementioned domains.

The contributions of this work are as follows. We identify a problem with standard VAE training when applied to sparse, high-dimensional data—underfitting. We investigate the underlying causes of this phenomenon, and propose modifications to the learning algorithm to address these causes. We combine inference networks with an iterative optimization scheme inspired by Stochastic Variational Inference (SVI) [Hoffman et al., 2013]. The proposed learning algorithm dramatically improves the quality of the estimated parameters. We empirically study various factors that govern the severity of underfitting and how the techniques we propose mitigate it. A practical ramification of our work is that improvements in learning NFA on recommender system data translate to more accurate predictions and better recommendations. In contrast, standard VAE training fails to outperform the simple shallow linear models that still largely dominate the collaborative filtering domain [Sedhain et al., 2016].

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

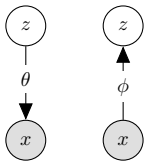


Figure 1: **Nonlinear Factor Analysis:** [**Left**] The generative model contains a single latent variable z. The conditional probability $p(x|z;\theta)$ parameterized by a deep neural network. [**Right**] The inference network $q_\phi(z|x)$ is used for inference at train and test time.

2 Background

Generative Model: We study learning in generative models of the form shown in Figure 1. We introduce it in the context of performing maximum likelihood estimation over a corpus of documents. We observe D word-count[1] vectors $x_{1:D}$, where x_{dv} denotes the number of times that word index $v \in \{1, \ldots, V\}$ appears in document d. Given the number of words per document $N_d \equiv \sum_v x_{dv}$, x_d is generated as:

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

That is, we draw a Gaussian random vector, pass it through a multilayer perceptron (MLP) parameterized by θ, pass the resulting vector through the softmax (a.k.a. multinomial logistic) function, and sample N_d times from the resulting distribution over V.[2]

Variational Learning: We drop the subscript on x_d when referring to a single data point. Jensen's inequality yields the following lower bound on the log marginal likelihood of the data:

$$\log p_\theta(x) \geq \underbrace{\mathbb{E}_{q(z;\psi)}[\log p_\theta(x \mid z)] - \text{KL}(\, q(z;\psi) \,||\, p(z)\,)}_{\mathcal{L}(x;\theta,\psi)}. \tag{2}$$

$q(z;\psi)$ is a tractable "variational" distribution meant to approximate the intractable posterior distribution $p(z \mid x)$; it is controlled by some parameters ψ. For example, if q is Gaussian, then we might have $\psi = \{\mu, \Sigma\}$, $q(z;\psi) = \mathcal{N}(z; \mu, \Sigma)$. We are free to choose ψ; ideally we would choose the ψ that makes the bound in equation 2 as tight as possible: $\psi^* \triangleq \arg\max_\psi \mathcal{L}(x;\theta,\psi)$.

Hoffman et al. [2013] proposed finding ψ^* using iterative optimization, starting from a random initialization. This is effective, but can be costly. More recently, Kingma and Welling [2014] and Rezende et al. [2014] proposed training a feedforward *inference network* [Hinton et al., 1995] to find good variational parameters $\psi(x)$ for a given x, where $\psi(x)$ is the output of a neural network with parameters ϕ that are trained to maximize $\mathcal{L}(x;\theta,\psi(x))$. Often it is much cheaper to compute $\psi(x)$ than to obtain an optimal ψ^* using iterative optimization. But there is no guarantee that $\psi(x)$ produces optimal variational parameters—it may yield a much looser lower bound than ψ^* if the inference network is either not sufficiently powerful or its parameters ϕ are not well tuned.

We will use $\psi(x)$ to denote an inference network that implicitly depends on some parameters ϕ, and ψ^* to denote a set of variational parameters obtained by applying an iterative optimization algorithm to equation 2. Following common convention, we will sometimes use $q_\phi(z \mid x)$ as shorthand for $q(z;\psi(x))$.

3 Mitigating Underfitting

Tight Lower Bounds on $\log p(x)$: Why might a learning algorithm for VAEs be underfitting? There are two sources of error in variational parameter estimation with inference networks:

The first is the distributional error accrued due to learning with a tractable-but-approximate family of distributions $q_\phi(z|x)$ instead of the true posterior distribution $p(z|x)$. Although difficult to compute in practice, it is easy to show that this error is exactly $\text{KL}(q_\phi(z|x)\|p(z|x))$. We restrict ourselves to working with normally distributed variational approximations and do not aim to overcome this source of error.

The second source of error comes from the suboptimality of the variational parameters ψ used in Eq. 2. We are guaranteed that $\mathcal{L}(x;\theta,\psi(x))$ is a valid lower bound on $\log p(x)$ for *any* output of $q_\phi(z|x)$ but within the same family of variational distributions, there exists an optimal choice of variational parameters $\psi^* = \{\mu^*, \Sigma^*\}$ realizing the tightest variational bound for a data point x.

$$\log p(x) \geq \tag{3}$$
$$\underbrace{\mathbb{E}_{\mathcal{N}(\mu^*;\Sigma^*)}[\log p(x|z;\theta)] - \text{KL}(\mathcal{N}(\mu^*, \Sigma^*)\|p(z))}_{\mathcal{L}(x;\theta,\psi^*)}$$
$$\geq \mathcal{L}(x;\theta,\psi(x))$$

Where we define: $\psi^* := \{\mu^*, \Sigma^*\}$
$= \arg\max_{\mu,\Sigma} \mathbb{E}_{\mathcal{N}(\mu,\Sigma)}[\log p(x|z;\theta)] - \text{KL}(\mathcal{N}(\mu, \Sigma)\|p(z))$.

Fig. 2 illustrates this double bound.

Standard learning algorithms for VAEs update θ, ϕ jointly based on $\mathcal{L}(x;\theta,\psi(x))$ (see Alg. 1 for pseudocode). That is, they directly use $\psi(x)$ (as output by $q_\phi(z|x)$) to estimate Equation 2.

[1] We use word-count in document for the sake of concreteness. Our methodology is generally applicable to other types of discrete high-dimensional data.

[2] 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.

In contrast, older stochastic variational inference methods [Hoffman et al., 2013] update θ based on gradients of $\mathcal{L}(x;\theta,\psi^*)$ by updating randomly initialized variational parameters for each example. ψ^* is obtained by maximizing $\mathcal{L}(x;\theta,\psi)$ with respect to ψ. This maximization is performed by M gradient ascent steps yielding $\psi_M \approx \psi^*$. (see Alg. 2).

Algorithm 1 **Learning with Inference Networks** [Kingma et al., 2014]

Inputs: $\mathcal{D} := [x_1, \ldots, x_D]$,
Model: $q_\phi(z|x)$, $p_\theta(x|z)$, $p(z)$;
for k = 1...K **do**
  Sample: $x \sim \mathcal{D}$, $\psi(x) = q_\phi(z|x)$, update θ, ϕ:
    $\theta^{k+1} \leftarrow \theta^k + \eta_\theta \nabla_{\theta^k} \mathcal{L}(x;\theta^k,\psi(x))$
    $\phi^{k+1} \leftarrow \phi^k + \eta_\phi \nabla_{\phi^k} \mathcal{L}(x;\theta^k,\psi(x))$
end for

Algorithm 2 **Learning with Stochastic Variational Inference**: M: number of gradient updates to ψ.

Inputs: $\mathcal{D} := [x_1, \ldots, x_D]$,
Model: $p_\theta(x|z)$, $p(z)$;
for k = 1...K **do**
  1. Sample: $x \sim \mathcal{D}$ and initialize: $\psi_0 = \mu_0, \Sigma_0$
  2. Approx. $\psi_M \approx \psi^* = \arg\max_\psi \mathcal{L}(x;\theta;\psi)$:
    For $m = 0, \ldots, M-1$:
      $\psi_{m+1} = \psi_m + \eta_\psi \frac{\partial \mathcal{L}(x;\theta^k,\psi_m)}{\partial \psi_m}$
  3. Update θ: $\theta^{k+1} \leftarrow \theta^k + \eta_\theta \nabla_{\theta^k} \mathcal{L}(x;\theta^k,\psi_M)$
end for

Limitations of Joint Parameter Updates: Alg. (1) updates θ, ϕ jointly. During training, the inference network learns to approximate the posterior, and the generative model improves itself using local variational parameters $\psi(x)$ output by $q_\phi(z|x)$. If the variational parameters $\psi(x)$ output by the inference network are close to the optimal variational parameters ψ^* (Eq. 3), then the updates for θ are based on a relatively tight lower bound on $\log p(x)$. But in practice $\psi(x)$ may not be a good approximation to ψ^*.

Both the inference network and generative model are initialized randomly. At the start of learning, $\psi(x)$ is the output of a randomly initialized neural network, and will therefore be a poor approximation to the optimal parameters ψ^*. So the gradients used to update θ will be based on a very loose lower bound on $\log p(x)$. These gradients may push the generative model towards a poor local minimum— previous work has argued that deep neural networks (which form the conditional probability distributions $p_\theta(x|z)$) are often sensitive to initialization [Glorot and Bengio, 2010, Larochelle et al., 2009]. Later in learning, $\psi(x)$ may yield suboptimal gradients for θ if the inference network cannot find optimal variational parameters for all data points.

Learning in the original SVI scheme does not suffer from this problem, since the variational parameters are optimized within the inner loop of learning before updating to θ (i.e. in Alg. (2); $\partial\theta$ is effectively derived using $\mathcal{L}(x;\theta,\psi^*)$). However, this method requires potentially an expensive iterative optimization.

This motivates blending the two methodologies for parameter estimation. Rather than relying on the output of the inference network, $\mathcal{L}(x;\theta,\psi(x))$ in Figure 2, to train the generative model we use its output to "warm-start" an SVI-style optimization that yields higher-quality variational parameters ψ^*. Taking gradients of the lower bound under ψ^*, $\mathcal{L}(x;\theta,\psi^*)$, should yield more meaningful gradients for θ.

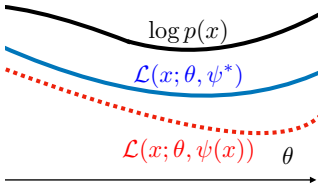


Figure 2: **Lower Bounds in Variational Learning:** To learn θ by maximizing a lower bound on $\log p(x;\theta)$. $\mathcal{L}(x;\theta,\psi(x))$ denotes the standard training objective used by VAEs. The tightness of this bound (relative to $\mathcal{L}(x;\theta,\psi^*)$ depends on the inference network. The x-axis is θ.

3.1 Optimizing Local Variational Parameters

We use the local variational parameters $\psi = \psi(x)$ predicted by the inference network to initialize an iterative optimizer. As in Alg. 2, we perform gradient ascent to maximize $\mathcal{L}(x;\theta,\psi)$ with respect to ψ. The resulting ψ_M approximates the optimal variational parameters: $\psi_M \approx \psi^*$. Since NFA is a continuous latent variable model, these updates can be achieved via the reparameterization gradient [Kingma and Welling, 2014]. We use ψ^* to derive gradients for θ under $\mathcal{L}(x;\theta,\psi^*)$. Finally, the parameters of the inference network (ϕ) are update using stochastic backpropagation and gradient descent, holding fixed the parameters of the generative model (θ). Our procedure is detailed in Alg. 3.

3.2 Representations for Inference Networks

The inference network must learn to regress to the optimal variational parameters for any combination of features, but in sparse datasets, many words appear only rarely. To provide more global context about rare words, we provide to the inference network (but not the generative network) TF-IDF [Baeza-Yates et al., 1999] features instead of raw counts. These give the inference network a hint that rare words are likely to be highly informative. TF-IDF is a popular technique in informa-

Algorithm 3 Maximum Likelihood Estimation of θ with Optimized Local Variational Parameters: Expectations in $\mathcal{L}(x, \theta, \psi^*)$ (see Eq. 3) are evaluated with a single sample from the optimized variational distribution. M is the number of updates to the variational parameters ($M = 0$ implies no additional optimization). $\theta, \psi(x), \phi$ are updated using stochastic gradient descent with learning rates $\eta_\theta, \eta_\psi, \eta_\phi$ obtained via ADAM [Kingma and Ba, 2015]. In step 4, we update ϕ separately from θ.

Inputs: $\mathcal{D} := [x_1, \ldots, x_D]$,
Inference Model: $q_\phi(z|x)$,
Generative Model: $p_\theta(x|z), p(z)$,
for k = 1...K **do**
1. Sample: $x \sim \mathcal{D}$ and set $\psi_0 = \psi(x)$
2. Approx. $\psi_M \approx \psi^* = \arg\max_\psi \mathcal{L}(x; \theta^k; \psi)$,
For $m = 0, \ldots, M-1$:
$$\psi_{m+1} = \psi_m + \eta_\psi \frac{\partial \mathcal{L}(x; \theta^k, \psi_m)}{\partial \psi_m}$$
3. Update θ,
$\theta^{k+1} \leftarrow \theta^k + \eta_\theta \nabla_{\theta^k} \mathcal{L}(x; \theta^k, \psi_M)$
4. Update ϕ,
$\phi^{k+1} \leftarrow \phi^k + \eta_\phi \nabla_{\phi^k} \mathcal{L}(x; \theta^{k+1}, \psi(x))$
end for

tion retrieval that re-weights features to increase the influence of rarer features while decreasing the influence of common features. The transformed feature-count vector is $\tilde{x}_{dv} \equiv x_{dv} \log \frac{D}{\sum_{d'} \min\{x_{d'v}, 1\}}$. The resulting vector $\tilde{x}$ is then normalized by its L2 norm.

We use TF-IDF features as an additional way to ease the job of the inference network. On its own the transformation represents a normalized linear scaling of the input, it doesn't change the power of the encoder; all it can do is help the encoder find a better local optimum using a representation that might otherwise be challenging to find via gradient descent.

4 Related Work

Salakhutdinov and Larochelle [2010] optimize local mean-field parameters from an inference network in the context of learning deep Boltzmann machines. Salimans et al. [2015] explore warm starting MCMC with the output of an inference network. Hjelm et al. [2016] explore a similar idea as ours to derive an importance-sampling-based bound for learning deep generative models with discrete latent variables. They find that learning with ψ^* does not improve results on binarized MNIST. This is consistent with our experience—we find that our secondary optimization procedure helped more when learning models of sparse data. Miao et al. [2016] learn log-linear models [multinomial-logistic PCA, Collins et al., 2001] of documents using inference networks. We show that mitigating underfitting in deeper models yields better results on the benchmark RCV1 data. Our use of the spectra of the Jacobian matrix of $\log p(x|z)$ to inspect learned models is inspired by Wang et al. [2016].

Previous work has studied the failure modes of learning VAEs. They can be broadly categorized into two classes. The first aims to improves the utilization of latent variables using a richer posterior distribution [Burda et al., 2015, Rezende and Mohamed, 2015]. However, for sparse data, the limits of learning with a normally distributed $q_\phi(z|x)$ have barely been pushed – our goal is to do so in this work. Further gains may indeed be obtained with a richer posterior distribution but the techniques herein can inform work along this vein. The second class of methods studies ways to alleviate the underutilization of latent dimensions due to an overtly expressive choice of models for $p(x|z;\theta)$ such as a Recurrent Neural Network [Bowman et al., 2016, Chen et al., 2017]. This too, is not the scenario we are in; underfitting of VAEs on sparse data occurs even when $p(x|z;\theta)$ is an MLP.

Our study here exposes a third failure mode; one in which learning is challenging not just because of the learning objective but also because of the *characteristics* of the data being modeled.

5 Evaluation

We first confirm our hypothesis empirically that underfitting is an issue when learning VAEs on high dimensional sparse datasets. We quantify the gains (at training and test time) obtained by the use of TF-IDF features and the continued optimization of $\psi(x)$ on two different types of high-dimensional sparse data—text and movie ratings. In Section 5.2, we learn VAEs on two large scale bag-of-words datasets. We study (1) *where* the proposed methods might have the most impact and (2) present evidence for *why* the learning algorithm (Alg. 3) works. In Section 5.3, we show that improved inference is crucial to building deep generative models that can tackle problems in top-N recommender systems. We conclude with a discussion.

5.1 Setup

Notation: $\psi(x)$ denotes learning with Alg. 1 and ψ^* denotes the results of learning with Alg. 3. $M = 100$ (number of updates to the local variational parameters) on the bag-of-words text data and $M = 50$ on the recommender systems task. M was chosen based on the number of steps it takes for $\mathcal{L}(x; \theta, \psi_m)$ (Step 2 in Alg. 3) to converge on training data. 3-ψ^*-norm denotes a model where the MLP parameterizing $\gamma(z)$ has three layers: two hidden layers and one output layer, ψ^* is used to derive an update of θ and normalized count features are conditioned on by the inference network.

In tables, we display evaluation metrics obtained under both $\psi(x)$ (output of the inference network) and ψ^* (optimized variational parameters). In figures, we display metrics obtained under ψ^* (even if the model was trained with $\psi(x)$) since $\mathcal{L}(x;\theta,\psi^*)$ always forms a tighter bound to $\log p(x)$. If left unspecified TF-IDF features are used as input to the inference network.

Training and Evaluation: We update θ using ADAM [Kingma and Ba, 2015] (with a batch size of 500), The inference network's intermediate hidden layer $h(x) = \text{MLP}(x;\phi_0)$ (we use a two-layer MLP in the inference network for all experiments) are used to parameterize the mean and diagonal log-variance as: $\mu(x) = W_\mu h(x), \log\Sigma(x) = W_{\log\Sigma} h(x)$ where $\phi = \{W_\mu, W_{\log\Sigma}, \phi_0\}$. Code is available at `github.com/rahulk90/vae_sparse`.

5.2 Bag-of-words text data

Datasets and Metrics: We study two large text datasets. (1) RCV1 [Lewis et al., 2004] dataset (train/valid/test: 789,414/5,000/10,000, V: 10,000). We follow the preprocessing procedure in Miao et al. [2016]. (2) The Wikipedia corpus used in Huang et al. [2012] (train/test: 1,104,937/100,000 and V:20,000) for which we set all words to lowercase, ignore numbers and restrict the dataset to the top $20,000$ frequently occurring words. We report an upper bound on perplexity [Mnih and Gregor, 2014] given by $\exp(-\frac{1}{N}\sum_i \frac{1}{N_i}\log p(x_i))$ where $\log p(x_i)$ is replaced by Eq 2. To study the utilization of the latent dimension obtained by various training methods, we compute the Jacobian $\mathcal{J}(z)$ matrix (as $\nabla_z \log p(x|z)$). The singular value spectrum of the Jacobian directly measures the *utilization* of the latent dimensions in the model. We justify this choice in the appendix.

Reducing Underfitting: Is underfitting a problem and does optimizing $\psi(x)$ with the use of TF-IDF features help? Table 1 confirms both statements.

We make the following observations: (1) between "norm" and "tfidf" (comparing first four rows and second four rows), we find that the use of TF-IDF features almost always improves parameter estimation; (2) optimizing $\psi(x)$ at test time (comparing column ψ^* with $\psi(x)$) always yields a tighter bound on $\log p(x)$, often by a wide margin. Even after extensive training the inference network can fail to tightly approximate $\mathcal{L}(x;\theta,\psi^*)$, suggesting that there may be limitations to the power of generic amortized inference; (3) optimizing $\psi(x)$ during training ameliorates under-fitting and yields significantly better generative models on the RCV1 dataset. We find that the degree of underfitting and subsequently the improvements from training with ψ^* are significantly more pronounced on the larger and sparser Wikipedia dataset (Fig. 3a and 3b).

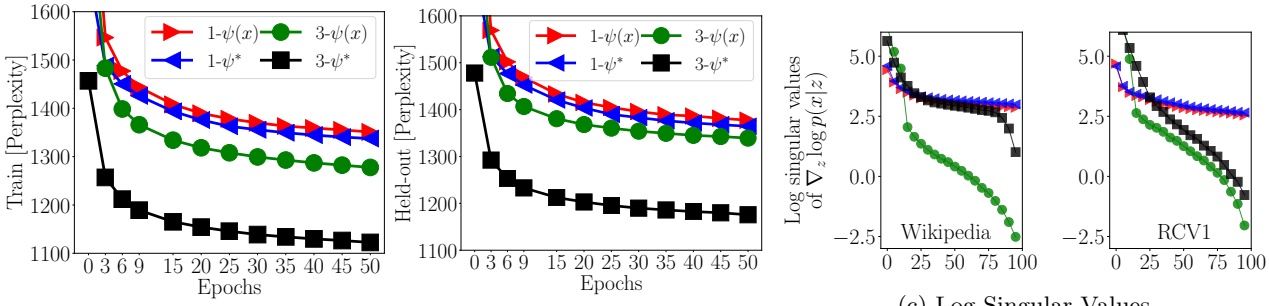
Effect of optimizing $\psi(x)$: How does learning with ψ^* affect the rate of convergence the learning algorithm? We plot the upper bound on perplexity versus epochs on the Wikipedia (Fig. 3a, 3b) datasets. As in Table 1, the additional optimization does not appear to help much when the generative model is linear. On the deeper three-layer model, learning with ψ^* dramatically improves the model allowing it to fully utilize its potential for density estimation. Models learned with ψ^* quickly converge to a better local minimum early on (as reflected in the perplexity evaluated on the training data and held-out data). We experimented with continuing to train 3-$\psi(x)$ beyond 150 epochs, where it reached a validation perplexity of approximately 1330, worse than that obtained by 3-ψ^* at epoch 10 suggesting that longer training is unsufficient to overcome local minima issues afflicting VAEs.

Overpruning of latent dimensions: One cause of underfitting is due to overpruning of the latent dimensions in the model. If the variational distributions for a subset of the latent dimensions of z are set to the prior, this effectively reduces the model's capacity. If the KL-divergence in Eq. 2 encourages the approximate posterior to remain close to the prior early in training, *and* if the gradient signals from the likelihood term are weak or inconsistent, the KL may dominate and prune out latent dimensions before the model can use them.

In Fig. 3c, we plot the log-spectrum of the Jacobian matrices for different training methods and models. For the deeper models, optimizing $\psi(x)$ is crucial to utilizing its capacity, particularly on the sparser Wikipedia data. Without it, only about ten latent dimensions are used, and the model severely underfits the data. Opti-

Table 1: **Test Perplexity on RCV1: Left: Baselines** Legend: LDA [Blei et al., 2003], Replicated Softmax (RSM) [Hinton and Salakhutdinov, 2009], Sigmoid Belief Networks (SBN) and Deep Autoregressive Networks (DARN) [Mnih and Gregor, 2014], Neural Variational Document Model (NVDM) [Miao et al., 2016]. K denotes the latent dimension in our notation. **Right:** NFA 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

Model	K	**RCV1**	**NFA**	$\psi(x)$	ψ^*
LDA	50	1437	1-$\psi(x)$-norm	501	481
LDA	200	1142	1-ψ^*-norm	488	454
RSM	50	988	3-$\psi(x)$-norm	396	355
SBN	50	784	3-ψ^*-norm	378	**331**
fDARN	50	724	1-$\psi(x)$-tfidf	480	456
fDARN	200	598	1-ψ^*-tfidf	482	454
NVDM	50	563	3-$\psi(x)$-tfidf	384	344
NVDM	200	550	3-ψ^*-tfidf	376	**331**



(a) Wikipedia - Training (b) Wikipedia - Evaluation (c) Log-Singular Values

Figure 3: **Mechanics of Learning:** Best viewed in color. **(Left and Middle)** For the Wikipedia dataset, we visualize upper bounds on training and held-out perplexity (*evaluated* with ψ^*) viewed as a function of epochs. Items in the legend corresponds to choices of training method. **(Right)** Sorted log-singular values of $\nabla_z \log p(x|z)$ on Wikipedia (left) on RCV1 (right) for different training methods. The x-axis is latent dimension. The legend is identical to that in Fig. 3a.

mizing $\psi(x)$ iteratively likely limits overpruning since the variational parameters (ψ^*) don't solely focus on minimizing the KL-divergence but also on maximizing the likelihood of the data (the first term in Eq. 2).

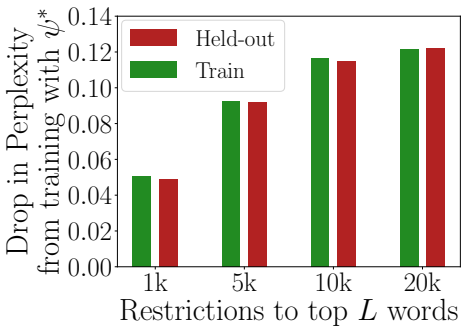


Figure 4: **Decrease in Perplexity versus Sparsity:** We plot the relative drop in perplexity obtained by training with ψ^* instead of $\psi(x)$ against varying levels of sparsity in the Wikipedia data. On the y-axis, we plot $\frac{P_{\psi(x)} - P_{\psi^*}}{P_{\psi(x)}}$; P denotes the bound on perplexity (evaluated with ψ^*) and the subscript denotes variational parameters used for training. The x-axis is a restriction of the dataset to the top L most frequently occurring words.

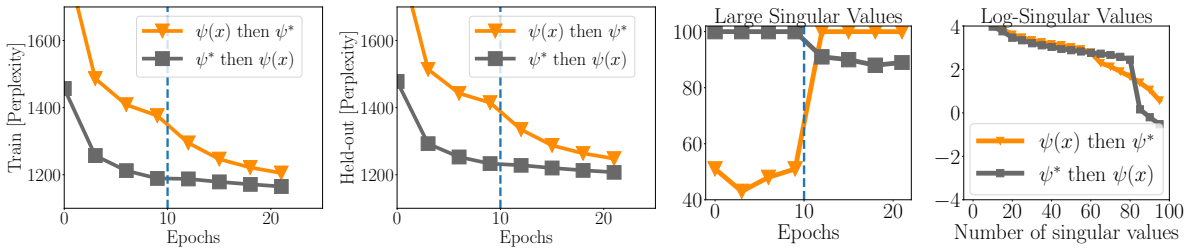
Sparse data is challenging: What is the relationship between data sparsity and how well inference networks work? We hold fixed the number of training samples and vary the sparsity of the data. We do so by restricting the Wikipedia dataset to the top L most frequently occurring words. We train three layer generative models on the different subsets. On training and held-out data, we computed the difference between the perplexity when the model is trained with (denoted P_{ψ^*}) and without optimization of $\psi(x)$ (denoted $P_{\psi(x)}$). We plot the relative decrease in perplexity obtained by training with ψ^* in Fig. 4.

Learning with ψ^* helps *more* as the data dimensionality increases. Data sparsity, therefore, poses a significant challenge to inference networks. A possible explanation is that many of the tokens in the dataset are rare, and the inference network therefore needs many sweeps over the dataset to learn to properly interpret these rare words; during this time, the generative model is receiving essentially random learning signals that drive it to a poor local optimum.

When should $\psi(x)$ be optimized: *When* are the gains obtained from learning with ψ^* accrued? We learn three-layer models on Wikipedia under two settings: (a) we train for 10 epochs using ψ^* and then 10 epochs using $\psi(x)$. and (b) we do the opposite.

Fig. 5 depicts the results of this experiment. We find that: (1) much of the gain from optimizing $\psi(x)$ comes from the early epochs, (2) somewhat surprisingly using ϕ^* instead of $\phi(x)$ later on in learning also helps (as witnessed by the sharp drop in perplexity after epoch 10 and the number of large singular values in Fig. 5c [Left]). This suggests that even after seeing the data for several passes, the inference network is unable to find $\psi(x)$ that explain the data well. Finally, (3) for a fixed computational budget, one is better off optimizing $\psi(x)$ sooner than later – the curve that optimizes $\psi(x)$ later on does not catch up to the one that optimizes $\psi(x)$ early in learning. This suggests that learning early with ψ^*, even for a few epochs, may alleviate underfitting.

Rare words and loose lower bounds: Fig. 4 suggests that data sparsity presents a problem for inference networks at an aggregate level. We now ask which *data points* benefit from the optimization of $\psi(x)$? We sample 20000 training and held-out data points; we compute $\mathrm{KL}(\psi(x) \| \psi^*)$ (both are Normal distributions and the KL is analytic) and the number of rare words in each document (where a word is classified as being rare if it occurs in less than 5% of training documents). We visualize them in Fig. 6. We also display the Spearman



(a) Training Data (b) Held-out Data (c) Log-singular Values

Figure 5: **Late versus Early optimization of $\psi(x)$:** Fig. 5a (5b) denote the train (held-out) perplexity for three-layered models trained on the Wikipedia data in the following scenarios: ψ^* is used for training for the first ten epochs following which $\psi(x)$ is used (denoted "ψ^* then $\psi(x)$") and vice versa (denoted "$\psi(x)$ then ψ^*"). Fig. 5c (Left) depicts the number of singular values of the Jacobian matrix $\nabla_z \log p(x|z)$ with value greater than 1 as a function of training epochs for each of the two aforementioned methodologies. Fig. 5c (Right) plots the sorted log-singular values of the Jacobian matrix corresponding to the final model under each training strategy.

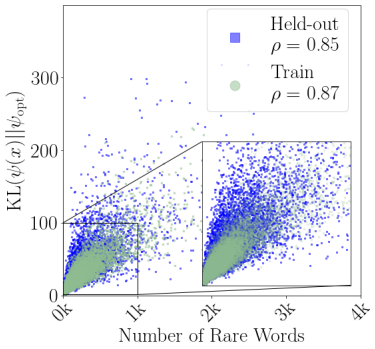


Figure 6: **Raw KL and Rare Word Counts:** We plot the raw values of $\mathrm{KL}(\psi(x)\|\psi^*)$ versus the number of rare words. We zoom into the plot and reduce the opacity of the train points to better see the held-out points. The Spearman ρ correlation coefficient is computed between the two across 20,000 points. We find a positive correlation.

ρ correlation between the two values in Fig. 6. There exists a positive correlation (about 0.88 on the training data) between the two values suggesting that the gains in perplexity that we observe empirically in Table 1 and Fig. 3 are due to being able to better model the likelihood of documents with rare words in them.

Putting it all together: Our work provides a narrative of how underfitting occurs when learning VAEs on sparse data. The rare words in sparse, high-dimensional data are difficult to map into local variational parameters that model the term $\mathbb{E}[\log p(x|z)]$ well (Fig. 4,6); $q_\phi(z|x)$ therefore focuses on the less noisy (the KL is evaluated analytically) signal of minimizing $\mathrm{KL}(q_\phi(z|x)||p(z))$. Doing so prunes out many latent dimensions early on resulting in underfitting (Fig. 5c [Left]). By using ψ^*, the inadequacies of the inference network are decoupled from the variational parameters used to derive gradients to θ. The tighter variational bound $\mathcal{L}(x;\theta,\psi^*)$ achieves a better tradeoff between $\mathbb{E}[\log p(x|z)]$ and $\mathrm{KL}(q_\phi(z|x)||p(z))$ (evidenced by the number of large singular values of $\nabla_z \log p(x|z)$ when optimizing ψ^* in Fig. 5c). The gradient updates with respect to this tighter bound better utilize θ.

How well would such an approach work on other kinds of data – e.g. images? When learning VAEs on MNIST Hjelm et al. [2016] [3] report no gains from optimizing $\psi(x)$ during training. Coupled with our findings, this suggests that the structure in the data plays an important role in the mechanics of how well deep generative models can be learned using inference networks.

5.3 Collaborative filtering

We study the top-N recommendation performance of NFA under strong generalization [Marlin and Zemel, 2009]. We condition on TF-IDF features in the inference network; since user feedback data has similar power-law behavior to bag-of-words text data, and rarely consumed items may be highly informative about user preferences.

Datasets: We study two large user-item rating datasets: MovieLens-20M (ML-20M) [Harper and Konstan, 2015] and Netflix[4]. We binarize the explicit rating data, keeping ratings of four or higher and interpret them as implicit feedback [Hu et al., 2008]; we keep users who have positively rated at least five movies. We train with users' binary implicit feedback as x_d; the vocabulary is the set of all movies. The number of training/validation/test users is 116,677/10,000/10,000 for ML-20M (V: 20,108) and 383,435/40,000/40,000 for Netflix (V: 17,769).

Evaluation and Metrics: We train with complete feedback history from training users, and evaluate on held-out validation/test users. For held-out users, we

[3] Page 10 ; Table 2; Compare Column 2, Line 2 (VAE) and 3 (VAE optimizing ψ in training)

[4] `http://www.netflixprize.com/`

randomly select 80% of the feedback as the input to the inference network and see how the other 20% of the positively rated items are ranked based on $\mu(z)$. We report two ranking-based metrics averaged over all held-out users: Recall@N and truncated normalized discounted cumulative gain (NDCG@N) [Järvelin and Kekäläinen, 2002]. For each user, both metrics compare the predicted rank of unobserved items with their true rank. We select model architecture (MLP with 0, 1, 2 hidden layers) based on validation NDCG@100 and report metrics on held-out test users.

Define π as a ranking over all the items where $\pi(v)$ indicates the v-th ranked item, $\mathbb{I}\{\cdot\}$ is the indicator function, and $d(\pi(v))$ returns 1 if user d has positively rated item $\pi(v)$. Recall@N and NDCG@N for user d are, respectively,

$$\text{Recall@}N(d,\pi) := \sum_{v=1}^{N} \frac{d(\pi(v))}{\min(N, \sum_{v'}^{V} d(\pi(v')))},$$

$$\text{DCG@}N(d,\pi) := \sum_{v=1}^{N} \frac{2^{\mathbb{I}\{d(\pi(v))=1\}} - 1}{\log(v+1)}.$$

As baselines, we consider:

Weighted matrix factorization (WMF) [Hu et al., 2008]: a linear low-rank factor model. We train WMF with alternating least squares; this generally leads to better performance than with SGD.

SLIM [Ning and Karypis, 2011]: a linear model which learns a sparse item-to-item similarity matrix by solving a constrained ℓ_1-regularized optimization problem.

Collaborative denoising autoencoder (CDAE) [Wu et al., 2016]: An autoencoder achitecture specifically designed for top-N recommendation. It augments a denoising autoencoder [Vincent et al., 2008] by adding a per-user latent vector to the input, inspired by standard linear matrix-factorization approaches. Among the baselines, CDAE is most akin to NFA.

Table 2 summarizes the results of NFA under different settings. For WMF and CDAE, we set the latent dimension to 100, the same as the latent dimension of z. We found that optimizing $\psi(x)$ helps both at train and test time and that TF-IDF features consistently improve performance. Crucially, the standard training procedure for VAEs realizes a poorly trained model that underperforms every baseline. The improved training techniques we recommend generalize across different kinds of sparse data. With them, *the same generative model*, outperforms CDAE and WMF on both datasets, and marginally outperforms SLIM on ML-20M while achieving nearly state-of-the-art results on Netflix. In terms of runtimes, we found that learning NFA (with ψ^*) to be approximately two-three times faster than SLIM. This highlights the importance of inference, showing that NFA, when properly fit, can outperform popular linear factorization approaches.

Table 2: **Recall and NDCG on Recommender Systems:** "2-ψ^*-tfidf" denotes a two-layer (one hidden layer and one output layer) generative model. Standard errors are around 0.002 for ML-20M and 0.001 for Netflix. **Runtime:** WMF takes on the order of minutes [ML-20M & Netflix]; CDAE and NFA ($\psi(x)$) take 8 hours [ML-20M] and 32.5 hours [Netflix] for 150 epochs; NFA (ψ^*) takes 36 hours [ML-20M] and 72 hours [Netflix]; SLIM takes 84 hours [ML-20M] and 336 hours [Netflix].

ML-20M	Recall@50		NDCG@100	
NFA	$\psi(x)$	ψ^*	$\psi(x)$	ψ^*
2-$\psi(x)$-norm	0.475	0.484	0.371	0.377
2-ψ^*-norm	0.483	0.508	0.376	0.396
2-$\psi(x)$-tfidf	0.499	0.505	0.389	0.396
2-ψ^*-tfidf	0.509	**0.515**	0.395	**0.404**
WMF	0.498		0.386	
SLIM	0.495		0.401	
CDAE	0.512		0.402	

Netflix	Recall@50		NDCG@100	
NFA	$\psi(x)$	ψ^*	$\psi(x)$	ψ^*
2-$\psi(x)$-norm	0.388	0.393	0.333	0.337
2-ψ^*-norm	0.404	0.415	0.347	0.358
2-$\psi(x)$-tfidf	0.404	0.409	0.348	0.353
2-ψ^*-tfidf	0.417	0.424	0.359	0.367
WMF	0.404		0.351	
SLIM	**0.427**		**0.378**	
CDAE	0.417		0.360	

6 Discussion

Studying the failures of learning is an important step to designing more robust architectures for inference networks. A question for future work is *why* inference networks have a harder time turning sparse data into variational parameters compared to images? One hypothesis is that the redundant correlations that exist among pixels (but occur less frequently in features found in sparse data) are more easily transformed into local variational parameters $\psi(x)$ that are, in practice, often reasonably close to ψ^* during learning.

Acknowledgements

The authors are grateful to David Sontag, Fredrik Johansson, Matthew Johnson, and Ardavan Saeedi for helpful suggestions regarding this work. Part of this work was done with RGK was an intern at Adobe.

References

R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 2003.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In *CoNLL*, 2016.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In *ICLR*, 2015.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. *ICLR*, 2017.

M. Collins, S. Dasgupta, and R. E. Schapire. A generalization of principal component analysis to the exponential family. In *NIPS*, 2001.

W. Gibson. Nonlinear factors in two dimensions. *Psychometrika*, 1960.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

F. M. Harper and J. A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2015.

G. E. Hinton and R. R. Salakhutdinov. Replicated softmax: an undirected topic model. In *NIPS*, 2009.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 1995.

R. D. Hjelm, K. Cho, J. Chung, R. Salakhutdinov, V. Calhoun, and N. Jojic. Iterative refinement of approximate posterior for training directed belief networks. In *NIPS*, 2016.

M. D. Hoffman, D. M. Blei, C. Wang, and J. W. Paisley. Stochastic variational inference. *JMLR*, 2013.

Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, 2008.

E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. Improving Word Representations via Global Context and Multiple Word Prototypes. In *ACL*, 2012.

K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 2002.

C. S. Jones. A nonlinear factor analysis of s&p 500 index option returns. *The Journal of Finance*, 2006.

C. Jutten and J. Karhunen. Advances in nonlinear blind source separation. In *ICA*, 2003.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. *JMLR*, 2009.

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *NIPS*, 2003.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *JMLR*, 2004.

B. M. Marlin and R. S. Zemel. Collaborative prediction and ranking with non-random missing data. In *RecSys*, 2009.

Y. Miao, L. Yu, and P. Blunsom. Neural variational inference for text processing. In *ICML*, 2016.

A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. In *ICML*, 2014.

X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In *ICDM*, 2011.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

R. Salakhutdinov and H. Larochelle. Efficient learning of deep boltzmann machines. In *AISTATS*, 2010.

T. Salimans, D. Kingma, and M. Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*, 2015.

S. Sedhain, A. K. Menon, S. Sanner, and D. Braziunas. On the effectiveness of linear models for one-class collaborative filtering. In *AAAI*, 2016.

C. Spearman. "general intelligence," objectively determined and measured. *The American Journal of Psychology*, 1904.

H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 2002.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

S. Wang, M. Plilipose, M. Richardson, K. Geras, G. Urban, and O. Aslan. Analysis of deep neural networks with the extended data jacobian matrix. In *ICML*, 2016.

Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*, 2016.