Scalable Variational Inference for Non-linear Beta Process Factor Analysis

Abstract

We propose a non-linear extension to factor analysis with beta process priors for improved data representation ability. This non-linear Beta Process Factor Analysis (nBPFA) allows data to be represented as a non-linear transformation of a standard sparse factor decomposition. We develop a scalable variational inference framework, which builds upon the ideas of the variational auto-encoder, by allowing latent variables of the model to be sparse. Our framework can be readily used for real-valued, binary and count data. We show theoretically and with experiments that our training scheme, with additive or multiplicative noise on observations, improves performance and prevents overfitting. We benchmark our algorithms on image and text datasets. We demonstrate faster convergence rates and competitive performance compared to standard gradient-based approaches.

1. Introduction

Sparse factor analysis is a well-known statistical method used to describe the correlation structure multivariate data, in terms of a potentially small set of K latent factors, such that each observation is allowed to explained by a combination of up to K of these factors. Beta Process Factor Analysis (BPFA) is a Bayesian nonparametric approach to sparse factor analysis, where factor memberships are modeled via a beta process prior (Paisley & Carin, 2009). The Beta processes (Hjort, 1990), and related priors for mixed membership, such as the Indian Buffet Process (IBP) (Ghahramani & Griffiths, 2005), are appealing because they provide a principled way of estimating K, which is usually unknown. The Beta process is also a fully-Bayesian conjugate prior, which in the case of BPFA, allows for analytic posterior calculation and fast variational inference (Paisley et al., 2010; Carin et al., 2011).

Though BPFA avoids the need to set K *a priori* and allows for efficient inference, it is limited to second-oder statistics, *i.e.*, linear combinations of Gaussian distributions (factors). We aim to improve the representation power of BPFA by allowing non-linear combinations of Gaussian factors. In principle, *non-linear BPFA* will be more flexible in its ability to explain the data because unlike traditional BPFA, is not restricted to only capture linear correlations.

A key problem of non-linear factor analysis models in general is that they are usually non-conjugate, which makes inference difficult and computationally expensive, if not prohibitive. Consequently, fast inference approaches for BPFA such as variational EM (Paisley & Carin, 2009), variational Bayes (Carin et al., 2011) and stochastic variational inference (Shah et al., 2015), can no longer be applied. The Variational Auto-Encoder (VAE) is a scalable stochastic variational inference and learning framework that efficiently approximates the posterior of the latent variables of a model using a *recognition model* (Kingma & Welling, 2013). In this framework, a Stochastic Gradient Variational Bayes (SGVB) estimator is used to optimize the recognition model, by minimizing the KL divergence between the prior imposed on the latent variables and the recognition model. Here, we extend VAE to allow latent variables to have a spike and slab prior, represented as an element-wise product of a continuous variable (Gaussian) and a binary vector drawn from a beta process prior. The new framework, termed *Variational Sparse Auto-Encoder* (VSAE), allows us to specify a non-linear version of BPFA in which both, the generative and the recognition models are built via multi-layer perceptrons (MLPs) (Hornik et al., 1989).

We also propose a more robust SGVB estimator, specifically designed for denoising and imputation, two tasks traditionally addressed by factor analysis models. It is understood that the VAE framework is advantageous for imputation purposes (Rezende et al., 2014). Here we go one step further by adding stochasticity to the gradient estimation in SGVB, via Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011), in order to improve denoising ability (Bengio et al., 2013). In particular, we leverage the *data corruption* technique of Maaten et al. (2013); Wager et al. (2013); Chen et al. (2014), which we demonstrate is equivalent to impose noise on the gradients in the SGVB formulation. Further, we justify the asymptotic properties of the robust SGVB for noise corrupted inputs.

The contributions of our work are: (i) We propose a non-

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

linear version of BPFA; (*ii*) We introduce a variational sparse auto-encoder framework for sparse latent variables; (*iii*) We propose a robust SGVB estimator based on data corruption techniques, with theoretical justification; (*iv*) Experiments on real image and text data demonstrate fast convergence rates and competitive performance *w.r.t* standard gradient-based approaches.

2. Background

Sparse factor analysis explains a data vector, $\mathbf{x}_i \in \mathbb{R}^D$, in terms of Gaussian latent variable, $\mathbf{z}_i, \mathbf{v}_i \in \mathbb{R}^K$, binary factor indicators, $\mathbf{v}_i \in \{0,1\}^K$, and weight matrix (factor loadings), $\mathbf{W} \in \mathbb{R}^{D\times K}$, with K factors as

$$\mathbf{x}_i = \mathbf{W}(\mathbf{z}_i \odot \mathbf{v}_i) + \boldsymbol{\epsilon}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \gamma^{-1}\mathbf{I}_D)\,, \tag{1}$$

where $\odot$ is represents the element-wise product, $\mathbf{z}_i$ follows a zero-mean Gaussian distribution with isotropic covariance matrix, denoted $\sigma_z^2\mathbf{I}_K$. Elements of the weight matrix, $\mathbf{W}$, are Gaussian with zero-mean. Observation noise, $\boldsymbol{\epsilon}_i$, is zero-mean Gaussian with covariance $\gamma^{-1}\mathbf{I}_D$. The prior distribution for each element of $\mathbf{v}$ is $v_{ik} \sim \text{Ber}(\xi_k)$, Bernoulli distributed, with $\xi_k \sim \text{Beta}(a/K, b(K-1)/K)$. This specification for sparse factor analysis has the advantage of being fully local-conjugate, thus conditional posteriors can be evaluated analytically (Paisley & Carin, 2009).

In the limit, when $K \to \infty$, the number of non-zero elements of $\mathbf{v}_i$ is a random variable with distribution Poisson(a/b), and $\mathbf{v}_i$ follows a beta process. Under these conditions, the model in (1) is called Beta Process Factor Analysis (BPFA) (Paisley & Carin, 2009).

Denote the two-parameter beta process as BP$(a, b, \mathcal{H}_0)$, where $a, b > 0$ and $\mathcal{H}_0$ is the base measure. The BP can be represented as follows (Zhou et al., 2009),

$$\begin{aligned} \mathcal{H}(v) &= \sum_{k=1}^{K} \xi_k \delta_{v_k}(v)\,, \\ \xi_k &= \text{Beta}(a/K, b(K-1)/K)\,, \quad v_k \sim \mathcal{H}_0\,, \end{aligned} \tag{2}$$

where $\mathcal{H}(v)$ is a well-defined probability measure for the vector $\mathbf{v}$, of length K, as $K \to \infty$ (Hjort, 1990). The generative process in (2) can be extended to binary matrices using a two-step process akin to the Indian buffet process (Thibaux & Jordan, 2007).

Performing inference directly from the infinite beta process is difficult and computationally expensive. In practice, a finite approximation is used instead, where K is set to a large but finite number, similar to truncated approximations for Dirichlet processes (Blei & Jordan, 2004). In such case, the local conjugacy of the specification in (1) can be still leveraged, and efficient inference can be carried out via variational methods (Paisley et al., 2010; Carin et al., 2011).

3. Non-linear BPFA

In this section, we propose a non-linear version of BPFA by imposing a non-linear transformation $f(\cdot)$, on the linear decomposition in (1) as

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \text{diag}(\boldsymbol{\sigma}_x))\,, \tag{3}$$

where $\boldsymbol{\mu}_x = f(\mathbf{W}(\mathbf{z} \odot \mathbf{v}))$ and $f(\cdot)$ is an invertible mapping, *e.g.*, sigmoid, hyperbolic tangent or their composition. The invertibility assumption can be relaxed in practice. Multi-layer perceptrons (MLPs) are commonly used non-linear mappings, however, they may not be invertible for some activations such as the rectified-linear function $\max\{0, u\}$. An invertible alternative to the rectified-linear used in MLPs is $\log(1 + e^u)$. Note that we have marginalized out $\boldsymbol{\epsilon}_i$ from (1) and we have replaced the isotropic covariance, $\gamma^{-1}\mathbf{I}_D$ from traditional BPFA, by a diagonal with elements specified in $\boldsymbol{\sigma}_x$. Details of $\boldsymbol{\mu}_x$ and $\boldsymbol{\sigma}_x$ are provided in the next section.

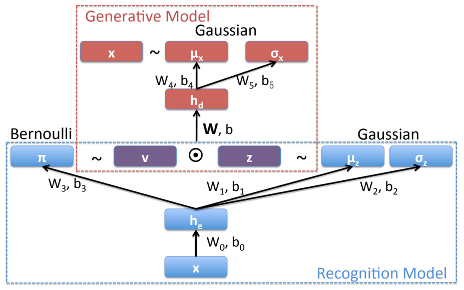
The efficient variational EM algorithm of Paisley & Carin (2009) cannot be used for (3) because the conditional posterior of $\mathbf{W}$ and $\mathbf{z}$ are no longer conjugate; at least for most choices of the function $f(\cdot)$. However, we can still leverage the finite approximation of the beta process to introduce a recognition model. In the following, we develop a stochastic variational inference framework to jointly learn the parameters of the generative model in (3) and a recognition network for latent variables $\mathbf{z}$ and $\mathbf{v}$, provided that $\mathbf{v}$ has a beta prior.

3.1. Scalable Variational Inference

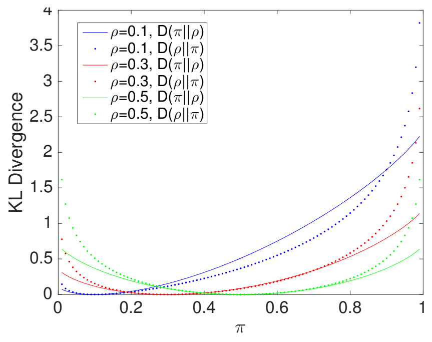
We construct our stochastic variational inference framework based upon ideas from the variational auto-encoder (VAE). We start by considering the generative model with joint distribution written concisely as $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, \mathbf{v}, \boldsymbol{\xi})$, where $\boldsymbol{\xi} = (\xi_1, \xi_2, ..., \xi_K)$ is the truncated beta process in (2). The generative model is parameterized then by $\boldsymbol{\theta}$ (including $\mathbf{W}$) from (3). Using Bayes' rule, we can factorize the joint of the generative model as $p(\boldsymbol{\xi}|\mathbf{v})p(\mathbf{x}|\mathbf{z}, \mathbf{v})p(\mathbf{z})p(\mathbf{v})$. We also propose a recognition model in the form of a variational distribution for the posterior of the latent variables as $q_{\boldsymbol{\psi}}(\mathbf{z}, \mathbf{v}, \boldsymbol{\xi}|\mathbf{x}) = q(\mathbf{z}|\mathbf{x})q(\boldsymbol{\xi}|\mathbf{v})q(\mathbf{v}|\mathbf{x})$, parameterized by $\boldsymbol{\psi}$. If we further assume that $q(\boldsymbol{\xi}|\mathbf{v})$ takes the form of the (unknown) true posterior $p(\boldsymbol{\xi}|\mathbf{v})$, the lower bound, $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}; \mathbf{x})$, can be simplified as follows

$$\begin{aligned} \log p(\mathbf{x}) &\geq \mathbb{E}_{q_{\boldsymbol{\psi}}}\left[\frac{\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, \mathbf{v}, \boldsymbol{\xi})}{\log q_{\boldsymbol{\psi}}(\mathbf{z}, \mathbf{v}, \boldsymbol{\xi}|\mathbf{x})}\right] \triangleq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}; \mathbf{x}) \\ &\geq \mathbb{E}_{q_{\boldsymbol{\psi}}}[\log p(\mathbf{x}|\mathbf{z}, \mathbf{v})] \\ &\quad - D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) - D_{KL}(q(\mathbf{v}|\mathbf{x})||p(\mathbf{v}))\,, \end{aligned}$$

where $p(v_k) \sim \text{Ber}\left(\frac{a}{a+b(K-1)}\right)$, after marginalizing out $\boldsymbol{\xi}$, with $p(v_k|\xi_k) \sim \text{Ber}(\xi_k)$ and ξ_k as in (2). We can further



(a)



(b)

Figure 1. (a) VSAE network for continuous input: $\sim$ means sampling, where $\mathbf{v} \sim \text{Ber}(\boldsymbol{\pi})$, $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_z, \text{diag}(\boldsymbol{\sigma}_z^2))$ and $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \text{diag}(\boldsymbol{\sigma}_x^2))$; each $\to$ means an element-wise mapping $g(W_i\mathbf{s} + \mathbf{b}_i)$, where g can be either nonlinear or linear and $\mathbf{s}$ is the source. $\mathbf{h}_e$ and $\mathbf{h}_d$ are the shared hidden layers in generative mode and recognition model respectively. (b) Comparison between different sparse penalties.

denote $\rho = \frac{a}{a+b(K-1)}$. The sparse prior imposed on $\mathbf{v}$, via $v_k \sim \text{Ber}(\rho)$, is controlled by tuning parameters a and b.

For the recognition model, the proposed variational distribution is restricted to be in a family of distributions of simpler form than true posterior, but preferably flexible enough to contain the true posterior as an instance. In particular, we let $q(\mathbf{z}|\mathbf{x})$ follow a $\mathcal{N}(\boldsymbol{\mu}_z, \text{diag}(\boldsymbol{\sigma}_z))$, rather than the zero-mean isotropic Gaussian distribution as in standard BPFA. Further, $q(\mathbf{v}|\mathbf{x})$ follows $\text{Ber}(\boldsymbol{\pi}_e)$, in a slight abuse of notation, meaning a product of K independent Bernoulli distributions with parameters in $\boldsymbol{\pi}_e$. These specifications guarantee that that the true posterior belongs to the same family as the variational distribution. To allow for flexibility, we use the trick of embedding parameters via multi-layer perceptrons, *i.e.*, $\boldsymbol{\mu}_z = \text{MLP}_1(\mathbf{x})$, $\boldsymbol{\sigma}_z = \text{MLP}_2(\mathbf{x})$ and $\boldsymbol{\pi}_e = \text{MLP}_3(\mathbf{x})$, where

$$\text{MLP}_j(\mathbf{x}) = g(\mathbf{W}_j\mathbf{h}_e + \mathbf{b}_j),\ \mathbf{h}_e = \tanh(\mathbf{W}_0\mathbf{x} + \mathbf{b}_0), \quad (4)$$

where $\mathbf{h}_e$ is a shared hidden layer for $\text{MLP}_{\{1,2,3\}}$. Note that the map $\mathbf{x} \to (\mathbf{v}, \mathbf{z})$ as specified above defines an *encoding process*, whereas (3) defines a *decoding process*. In fact, we can use a MLP formulation for (3), particularly, we let $\boldsymbol{\mu}_x = \text{MLP}_4(\mathbf{z} \odot \mathbf{v})$ and $\boldsymbol{\sigma}_x = \text{MLP}_5(\mathbf{z} \odot \mathbf{v})$, both with a shared hidden decoding layer, $\mathbf{h}_d = \tanh(\mathbf{W}(\mathbf{z} \odot \mathbf{v}) + \mathbf{b})$. As a result, parameters $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$ are weights and biases of all MLPs, $\{1, 2, 3\}$ for the encoder and $\{4, 5\}$ for the decoder. We call this framework variational sparse auto-encoder (VSAE).

Compared to standard VAE (Kingma & Welling, 2013), our recognition model has two additional components

$$\boldsymbol{\pi}_e = \text{sigmoid}(\mathbf{W}_3\mathbf{h}_e + \mathbf{b}_3)\,, \quad \mathbf{v} \sim \text{Ber}(\boldsymbol{\pi}_e)\,,$$

where $\mathbf{h}_e$ is shared and defined in (4) and recall that $\mathbf{z} \leftarrow \mathbf{z} \odot \mathbf{v}$. The compete model, which can be seen as a deep neural network, is illustrated in Figure 1. The input and the output layers have D nodes since $\mathbf{x}_i \in \mathbb{R}^D$, and the middle layer latent layer has two variables $\mathbf{v}$ and $\mathbf{z}$, with K nodes each. For real-valued inputs, we have two output layers as $\boldsymbol{\mu}_x$ and $\boldsymbol{\sigma}_x$, respectively. The number of nodes on hidden encoder and decoder layers is set empirically. For backpropagation purposes, the difference between VAE and VSAE lower bounds, using a factorized variational distribution for $\mathbf{v}$ and $\mathbf{z}$ is

$$\begin{aligned} D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) &+ D_{KL}(q(\mathbf{v}|\mathbf{x})||p(\mathbf{v})) \\ &= \tfrac{1}{2}\textstyle\sum_{k=1}^K(\mu_k^2 + \sigma_k^2 - 1 - 2\log\sigma_k) \\ &+ \textstyle\sum_{k=1}^K \left(\pi_k \log\frac{\pi_k}{\rho} + (1-\pi_k)\log\frac{1-\pi_k}{1-\rho}\right)\,, \end{aligned}$$

where μ_k, σ_k and π_k are elements of $\boldsymbol{\mu}_x$, $\boldsymbol{\sigma}_x$ and $\boldsymbol{\pi}_e$, respectively. The second KL divergence term is imposed to control the sparsity of latent variable $\mathbf{v}$. Note that the KL divergence of a Bernoulli distribution reaches its minimum of 0 if $\pi_i = \rho$. As a result, when the parameter of the sparse prior, ρ, becomes smaller, the model will become sparser. In fact, the sparsity constraint in standard auto-encoders can be formulated as $D_{KL}(p(\mathbf{v})||q_{\boldsymbol{\psi}}(\mathbf{v}|\mathbf{x}))$ (Ngiam et al., 2011). However, when π approaches 0 or 1, this regularization term may diverge to infinity (see Figure 1(b)), thus requiring an additional tuning weight, λ, to act as a trade-off between the magnitude of the KL divergence and the log-likelihood expectation, $\mathbb{E}_{q_{\boldsymbol{\psi}}}[\log p(\mathbf{x}|\mathbf{z}, \mathbf{v})]$. Instead, the KL divergence term in our lower bound can reach its maximum either at $\log(1/\rho)$ or $\log(1/(1-\rho))$. As a result, the additional tuning parameter is not necessary.

3.2. Continuous Approximation

In back-propagation inference, Monte Carlo integration is required to calculate the expectation $\mathbb{E}_{q_{\boldsymbol{\psi}}}[\log p(\mathbf{x}|\mathbf{z}, \mathbf{v})]$. Due to the nice property of location scale-families, the variance of integration with respect to a latent Gaussian variable, $\mathbf{z}$, can be reduced by the reparameterization trick (Kingma & Welling, 2013). In fact, even one draw from the standard Gaussian distribution is sufficient to approximate the expectation with an exponential decay error, under some mild conditions (Fan et al., 2015). However, as to the latent Bernoulli variable, $\mathbf{v}$, one draw from the Bernoulli distribution using Monte Carlo integration may result in a very large bias, which in turn, will decrease the convergence rate. The reason is, $\mathbb{E}[f(\mathbf{v})] = \sum_{\mathbf{v}\in\{0,1\}^K}\prod_{k=1}^{K} p_k^{v_k}(1-p_k)^{1-v_k} f(\mathbf{v})$ includes the summation of 2^K terms, and one draw is only one of them. Mnih & Gregor (2014) applied the control variate trick of reinforcement learning to reduce the variance. In this paper, we can circumvent this variance reduction need, by directly setting $\boldsymbol{\pi}_e \in [0,1]^K$, as the latent variable during stochastic back-propagation, in order to avoid drawing too many samples to achieve the intended accuracy. This continuous approximation for binary variables does not harm the regularization term or the KL divergence in the objective lower bound. Meanwhile, it reduces $\mathbf{v}$ to deterministic back-propagation while other latent variables remain the same. Consequently, significant less computation is required compared with the control variate trick or multi-sample Monte Carlo integration.

Any stochastic optimization for variational inference is allowed in our framework, so the algorithm is scalable for large datasets. Uncertainty of $\mathbf{W}$ in (3) accounted for in BPFA is omitted here for simplicity. Correspondingly, only the weight matrix associated with $\mathbf{z}\odot\mathbf{v}$ in our model needs uncertainty, *e.g.*, $\mathbf{W}_5$ in Figure 1(a). However, the technique in (Blundell et al., 2015; Kingma et al., 2015) of inferring weight uncertainty in neural networks can straightforwardly applied to all weight matrices, which is not the specific topic of our paper. In addition, we can simplify VSAE by removing the hidden encoder and decoder layers of VSAE, $\mathbf{h}_e$ and $\mathbf{h}_d$, thus yielding a simpler version of non-linear BPFA. Further, in the next section, we will discuss how we implement VSAE with a noisy training method to make the factor analysis model faster and more robust for denoising and imputation tasks.

4. Noisy VSAE

4.1. Data Corruption

Unlike traditional VAE, one of the advantages of VSAE is that it controls the activation of latent variables for different inputs. In principle, this setting will make the model more flexible to represent more data. In this section, we design a noisy training scheme to further strengthen this ability. In a nutshell, our stochastic optimization algorithm resembles stochastic gradient Langevin dynamics (SGLD).

4.1.1. Stochasticity from Data Noise

Maximizing the the lower bound of an auto-encoder is equivalent to optimize a function $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\psi};\mathbf{x},\mathbf{t})$, where the target $\mathbf{t}$ equals input $\mathbf{x}$. If we consider $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\psi};\tilde{\mathbf{x}},\mathbf{x})$, where the input is noised but the target remains unchanged, we can analyze how to impose stochasticity to the estimated gradient from corrupted input. Particularly, we consider two simple *corruption* settings: additive and multiplicative Gaussian noise, which is a fairly common practice in recent machine learning literature (Srivastava et al., 2014; Kingma et al., 2015). We will show that data noise is equivalent to adding a preconditioned adjustment to the gradient of the parameters.

Denote $\boldsymbol{\phi} = \{\boldsymbol{\theta},\boldsymbol{\psi}\}$ and $\mathbf{g}(\boldsymbol{\phi},\mathbf{x},\mathbf{t})$, as the first order derivative of the lower bound *w.r.t.* $\boldsymbol{\phi}$. For additive noise, we assume $\Delta\mathbf{x}\sim\mathcal{N}(\mathbf{0},\alpha\mathbf{I}_D)$. For multiplicative noise, the corrupted input $\tilde{\mathbf{x}} = \mathbf{x}\odot\boldsymbol{\sigma}$, where $\boldsymbol{\sigma}\sim\mathcal{N}(\mathbf{1},\alpha\mathbf{I}_D)$. Thus, $\Delta\mathbf{x} = \mathbf{x}\odot(\boldsymbol{\sigma}-\mathbf{1})$, and $\Delta\mathbf{x}\sim\mathcal{N}(\mathbf{0},\alpha\boldsymbol{\Sigma}_{\mathbf{x}})$, where $\boldsymbol{\Sigma}_{\mathbf{x}} = \mathrm{diag}(\mathbf{x}\odot\mathbf{x})$. In the dropout setting (Srivastava et al., 2014) with drop rate β, α can be set to $\beta/(1-\beta)$.

Consider the Taylor expansion of vector-valued the function $\mathbf{g}:\mathbb{R}^D\to\mathbb{R}^P$, *w.r.t.* input variable $\mathbf{x}$,

$$\mathbf{g}(\boldsymbol{\phi},\mathbf{x}+\Delta\mathbf{x},\mathbf{t}) = \mathbf{g}(\boldsymbol{\phi},\mathbf{x},\mathbf{t}) + \mathbf{J}(\mathbf{x})\Delta\mathbf{x} + o(\|\Delta\mathbf{x}\|), \quad (5)$$

where $\mathbf{J}(\mathbf{x})$ is Fréchet derivative or Jacobian matrix with dimension $P\times D$, $\|\cdot\|$ represents the Euclidean norm and $o(\cdot)$ is an infinitesimal. In fact, the linear map described by $\mathbf{J}$ is the best linear approximation of $\mathbf{g}$ near the point $\mathbf{x}$. For both additive and multiplicative noise, the resulting noise imposed on the gradient by a linear transformation is also Gaussian noise, but with a new covariance matrix $\boldsymbol{\Sigma} = \alpha\mathbf{J}\mathbf{J}^\top$ or $\alpha\mathbf{J}\boldsymbol{\Sigma}_{\mathbf{x}}\mathbf{J}^\top$. However, like back-propagation for deep neural networks, the error between the output and the target has a less significant influence on the parameter gradient of lower layers; similarly, the corruption introduced here will decrease and have less impact on the top layers. Thus, dropout is usually applied to each layer separately.

4.1.2. Connection to SGLD

Equation (5) indicates data noise can be transferred to gradient noise. However, the purpose of SGLD is to follow the Markov chain Monte Carlo (MCMC) approach to capture uncertainty, by directly adding a zero-mean Gaussian noise with decreasing variance, to a standard SGD update: $\Delta\boldsymbol{\phi}_t = \frac{\epsilon_t}{2}\left(\nabla\log p(\boldsymbol{\phi}) + \frac{N}{B}\sum_{i=1}^{B}\nabla\log p(\mathbf{x}_{t_i}|\boldsymbol{\phi})\right) + \boldsymbol{\eta}_t$ where $\boldsymbol{\eta}_t\sim\mathcal{N}(\mathbf{0},\epsilon_t\mathbf{I}_P)$ and B is the mini-batch size. Note that another two sources of stochasticity exist: one is the rescaled gradient estimated from the mini-batch data; the other one is the Langevin dynamics from injected Gaussian noise. In practice, a variety of adaptive learning rate optimization methods have been proposed, especially in the optimization community (Duchi et al., 2011; Kingma & Ba, 2014). For the rationale of the magnitude of additive Gaussian noise, an intensive analysis of the consistency and fluctuations of SGLD under verifiable assumptions implies that its asymptotic bias and variance can be rigorously characterized by an explicit function of the learning rate sequence (Teh et al., 2014). The lower bound $\mathcal{L}(\boldsymbol{\phi}; \mathbf{x}, \mathbf{t})$ of VSAE can also be decomposed to two parts. The first term is the expectation of $\log p(\mathbf{t}|\mathbf{z}, \mathbf{v}, \boldsymbol{\phi})$ dependent on the target. The second one is the KL divergence term between the posterior and prior of latent variables. Like in SLGD, we can also add the parameter prior $p(\boldsymbol{\phi})$. Therefore, we will provide a unifying framework to demonstrate how the three sources of stochasticity can affect the gradient based algorithm.

4.1.3. Consistency and Local-Optimality

As previously discussed, multiplicative noise can be essentially considered as additive noise with adaptive variance, dependent on each evaluated data point. Thus, we only explore the convergence properties in the case of additive noise. In principle, each element function of $\mathbf{g}$ is required to belong to a 2nd order smoothness function C_L^2 (Van Der Vaart & Wellner, 1996) *w.r.t.* $\mathbf{x}$, where the class of C_L^2 has uniformly L-bounded partial derivatives up to order 2, namely, the 2nd order Hölder continuous partial derivatives. For clarity, denote the average negative lower bound as $\mathcal{L}_{an}(\boldsymbol{\phi}) = -\frac{1}{N}\left(\log p(\boldsymbol{\phi}) + \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{\phi}; \mathbf{x}_i, \mathbf{t}_i)\right)$. One can minimize this objective function to obtain an estimate of $\boldsymbol{\phi}$. For $\mathcal{L}_{an}$, we only need to assume it is locally convex (*i.e.*, convex in a bounded region, $\{\boldsymbol{\phi} : \|\boldsymbol{\phi} - \boldsymbol{\phi}_1\|^2 \leq 2R\}$) to guarantee the exist of local optima. In fact, for 1st order gradient based algorithms, it is not necessary to assume that $\mathcal{L}_{an}$ is twice differentiable; instead we merely consider a weaker condition: $\nabla_{\boldsymbol{\phi}} \mathcal{L}_{an}$ is γ-Lipschitz, *i.e.*, $\mathcal{L}_{an}$ is γ-smoothness.

Theorem 1 (Local Convergence Rate). *Assume $\mathcal{L}_{an}(\boldsymbol{\phi})$ is γ-smoothness local convex function w.r.t $\boldsymbol{\phi}$; $\mathbf{g}(\boldsymbol{\phi}, \mathbf{x}, \mathbf{t}) \in C_L^2$ w.r.t $\mathbf{x}$; stochastic gradient oracle on each data point variance is bounded by C. Then, by setting input $\mathbf{x}$ corrupted with additive Gaussian noise $\mathcal{N}(\mathbf{0}, \alpha \mathbf{I}_D)$, mini-batch size B and step-size $\epsilon_t = \frac{4}{N} \frac{1}{\gamma + \sqrt{\frac{t(C + L^2 D + o(D\alpha) + 2B/N)}{2RB}}}$, SGLD update satisfies*

$$\mathbb{E}\left[\mathcal{L}_{an}\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\phi}_{t+1}\right)\right] \leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\mathcal{L}_{an}\left(\boldsymbol{\phi}_{t+1}\right)\right]$$
$$\leq \mathcal{L}_{an}(\boldsymbol{\phi}^*) + o\left(\sqrt{D\alpha}\right)\left|\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\boldsymbol{\phi}_t - \boldsymbol{\phi}^*]\right| \qquad (6)$$
$$+ \frac{R\gamma}{T} + \sqrt{\frac{2R(C + L^2 D + o(D\alpha) + 2B/N)}{TB}}\,.$$

where $\boldsymbol{\phi}^*$ is the local optimum.

Intuitively, Theorem 1 (proof and relevant definitions provided in the supplements) characterizes the behavior when the algorithm has already moved into a local mode, and indicates that the model bias induced by averaging the sampled parameter can be bounded by averaging models, and further bounded by three monotonically decreasing terms *w.r.t* sample size. The first term is associated with the bias of parameter posterior mean scaled by an infinitesimal $o\left(\sqrt{D\alpha}\right)$, which is controlled by the standard deviation of data corruption noise. Note that if no data corruption is introduced in the algorithm, this term will vanish completely, thus degenerating to the convergence rate of standard SGLD. In addition, the second and third terms diminish with order $\mathcal{O}\left(\frac{1}{T}\right)$ and $\mathcal{O}\left(\frac{1}{\sqrt{TB}}\right)$, respectively.

In the case of multiplicative noise, we can obtain a similar convergence bound, by defining an additive noise with bounded variance, $\sqrt{2R}\alpha$, and substituting for the α within inequality (6). If we reformulate to the gradient update in SGLD, it can be viewed as adding a zero-mean Gaussian noise, $\mathcal{N}\left(\mathbf{0}, \epsilon_t \mathbf{I} + \frac{N\epsilon_t}{2B}\sum_{i=1}^{B}\mathbf{J}(\mathbf{x}_{ti})\boldsymbol{\Sigma}_{\mathbf{x}_{ti}}\mathbf{J}(\mathbf{x}_{ti})^\top\right)$. However, it is unnecessary to add this noise in the gradient update, since the covariance matrix is not diagonal. Sampling from such a distribution with less numerical error usually requires Cholesky decomposition, while directly adding noise to input data takes much less computation.

4.1.4. Consistency with Moving Average

In addition to modifying the standard SGLD, in this section we further discuss how the momentum method (Polyak, 1964; Nesterov, 1983) is influenced with noisy training by exploring its theoretical properties, which is scarcely discussed in previous literature, to the best of our knowledge. We can readily formulate a standard momentum (SM) or Nesterov's Acceleration (NA) update for SGLD as follows,

$$\text{SM} \;:\mathbf{p}_{t+1} = \mu_t \mathbf{p}_t - \epsilon'_t \nabla \mathcal{L}_{an}(\boldsymbol{\phi}_t) + \boldsymbol{\eta}_t\,, \qquad (7)$$
$$\text{NA} \;:\mathbf{p}_{t+1} = \mu_t \mathbf{p}_t - \epsilon'_t \nabla \mathcal{L}_{an}(\boldsymbol{\phi}_t + \mu_t \mathbf{p}_t) + \boldsymbol{\eta}_t\,, \qquad (8)$$
$$\boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}_t + \mathbf{p}_{t+1}\,, \qquad (9)$$

where $\epsilon'_t = N\epsilon_t/2$. From the perspective of implementation, the two-step update is convenient. However, we can rewrite as a one-step second order moving average. Note that the second order here refers to the parlance in autoregressive (AR) models literature. The motivation for introducing momentum is to average previous samples with minor weights. More concretely, we have

$$\boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}_t + \mu_t(\boldsymbol{\phi}_t - \boldsymbol{\phi}_{t-1}) - \epsilon'_t \nabla \mathcal{L}_{an}(\boldsymbol{\phi}_t) + \boldsymbol{\eta}_t\,, \quad (10)$$

For NA, we only need to change $\nabla \mathcal{L}_{an}(\boldsymbol{\phi}_t)$ to $\nabla \mathcal{L}_{an}(\boldsymbol{\phi}_t + \mu_t \mathbf{p}_t)$. Intuitively, the momentum method plays the role of the *inertia* in the Newton's laws of motion. The next direction for gradient descent takes the acceleration velocity between previous two steps into account, *i.e.*, $\frac{\boldsymbol{\phi}_{t+\delta t} - \boldsymbol{\phi}_t}{\delta t}$, where the time interval $\delta t = 1$ in our case. This means the

Algorithm 1 Training with raw and noisy data.

```
1: for $t = 1, 2, \ldots$ do
2:     $\mathbf{x}_{b=1}^{B/2} \leftarrow$ Randomly draw $B/2$ data points.
3:     $\tilde{\mathbf{x}}_{b=1}^{B/2} \leftarrow \mathbf{x}_{b=1}^{B/2}$ with injected noise.
4:     Set input $\mathbf{x}_{b=1}^{B/2}, \tilde{\mathbf{x}}_{b=1}^{B/2}$ and target $\mathbf{x}_{b=1}^{B/2}, \mathbf{x}_{b=1}^{B/2}$
5:     Optimize the lower bound of the VSAE.
6: end for
```

newly sampled parameters depend on three key factors: the original direction computed from negative gradient, the inertia direction from previous history, and the random direction from injected Gaussian noise. Geometrically, from the vertex ϕ_t, we can follow a parallelogram law with the first two directions of the length ϵ_t and μ_t respectively; subsequently, a small perturbation is added. We show a figure in the supplements that intuitively illustrates that the Langevin dynamics can explore larger parameter space, thus potentially mitigating the overfitting problem, which is one of most appealing advantages of noisy training. Under minor modification of the assumptions in Theorem 1, we obtain a similar result.

Theorem 2. *Under the same assumptions of Theorem 1, except for relaxing the smoothness condition to adaptive smoothness, i.e., for the current ϕ_t, the function satisfies $\frac{1}{\epsilon_t'}$-smoothness in the region $\{\phi : \|\phi - \phi_t\| \le \epsilon_t'\}$, SGLD with update (8) and (9) satisfies*

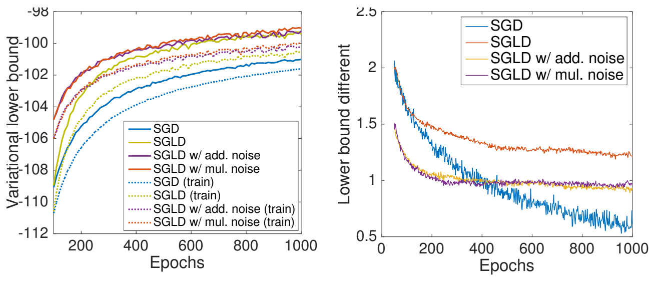
$$\mathbb{E}\left[\mathcal{L}_{an}\left(\phi_T\right)\right] \le \mathcal{L}_{an}(\phi^*) + \frac{4R\sup_{t<T}\gamma_t}{T^2} + \sqrt{\frac{8R(C+2B/N)\sup_{t<T}\gamma_t}{TB}}\left(\frac{4}{3} + \frac{1}{2T} + \frac{1}{6T^2}\right).$$

where $\epsilon_t = \frac{2}{N}\frac{1}{\gamma_t + \sqrt{\frac{t^3(C+2B/N)}{2RB\sup_{i<t}\gamma_t}}}$.

Note that R can be replaced by $\|\phi_1 - \phi^*\|^2/2$. Due to the stochasticity from the mini-batch and Langevin dynamics, the convergence rate is deteriorated by a term $\mathcal{O}\left(\frac{1}{\sqrt{TB}}\right)$; however, this result is comparable with the optimization algorithm, ADAM (Kingma & Ba, 2014), whose convergence rate is proved to be $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$, within the online learning framework and thus equivalent to our result without mini-batch size in the denominator. Further, we include the analysis connecting to HMC (Neal et al., 2011) in supplementary materials.

5. Experiments

We explore how different training approaches can affect the performance of convergence rates, overfitting and denoising. In addition, we apply the VSAE framework to topic modeling.



(a) Lower Bound (b) Underfitting

Figure 2. Performance comparison on MNIST data. In (a), solid and dashed lines are test and train lower bounds, respectively. (b) illustrates testing and training lower bound differences.

5.1. Practical Algorithm

We first specify a training algorithm for noisy data. The idea is to double the training set size using raw and noise corrupted data. In each iteration, we sample a mini-batch of data points $\mathbf{X}$ with size $B/2$, and inject some appropriate noise to obtain $\tilde{\mathbf{X}}$. Then, we feed the input $\{\mathbf{X}, \tilde{\mathbf{X}}\}$ to the auto-encoder using $\{\mathbf{X}, \mathbf{X}\}$ as target. This is similar to the approach discussed in the previous section, except that input data points are not all corrupted. This training framework is illustrated in Algorithm 1.

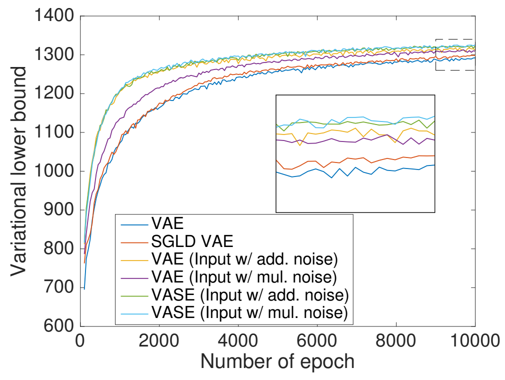
Different SGLD algorithms are considered. For comparison, we implemented algorithms in (Kingma & Welling, 2013; Fan et al., 2015), referred here as SGD and Hessian-free SVI (HFSVI). Results on two well-known image datasets, MNIST and Frey faces, are shown in Table 1 and Figure 2 (all algorithms are tuned for performance with and without momentum update). Since HFSVI converges quickly (within 50 epochs), we do not include it in the convergence rate comparison of 1st order algorithms. However, is clear that HFSVI suffers from severe overfitting, even when we use a mini-batch of size 1000 (a mini-batch of size 100 performs worse). Note that 2nd order algorithms often require large batch size (Martens, 2010). For all 1st order algorithms, we use a mini-batch size of 100. The settings for MNIST and Frey faces data are $d_h = 400$ and $d_h = 200$, respectively, while $d_z = 200$ and $\rho = 0.1$ in both cases.

For MNIST, our algorithms perform better than SGD in both testing and training lower bound. In particular, SGLD with data corrupted with multiplicative noise $\mathcal{N}(1, 0.4^2)$ converges faster. This behavior may be counterintuitive to the theoretical analysis in Theorem 1. However, as we discussed, one possible reason is data noise can be translated to gradient noise by the Jacobian matrix, then the appropriate amount of noise may lead to a larger parameter searching space. Since MNIST has a relatively large size, we see underfitting on all 1st order algorithms. It seems that

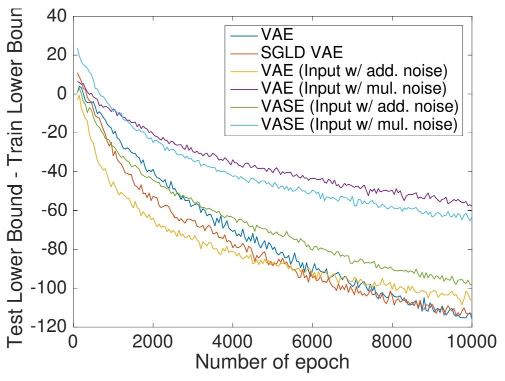
Table 1. Lower Bound Report

Dataset	SGD		SGLD		SGLD add.noise		SGLD mul.noise		HFSVI	
	train	**test**	**train**	**test**	**train**	**test**	**train**	**test**	**train**	**test**
MNIST	-101.5	-100.9	-100.6	-99.4	-100.1	-99.2	-99.9	-99.0	-109.0	-102.5
FreyFace	1406	1289	1411	1298	1419	1312	1367	1309	1437	1287

SGD suffers less from underfitting, however, its test lower bound is smaller than the smallest training lower bound of other algorithms. For Frey faces, SGLD with data corrupted by either additive noise $\mathcal{N}(0, 0.1^2)$ or multiplicative noise $\mathcal{N}(1, 0.3^2)$ achieves faster convergence, while vanilla SGLD shows no significant improvement over traditional SGD.



(a) Test LowerBound



(b) Overfitting

Figure 3. Results on Frey faces dataset.

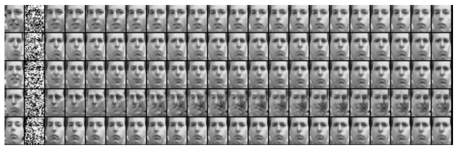
5.2. Preventing Overfitting

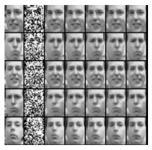
We train the model on a small dataset of approximately 2000 images, Frey faces, with real-valued inputs. Model structure from bottom to top as described in Figure 1(a) is 560-200-(200,200)-200-(560,560) where $\rho = 0.1$, is set to be equivalent to standard VAE with 20 latent variables (10% sparsity). For comparison, we also train a standard VAE 560-200-(20,20)-200-(560,560). Results are shown in Figure 3. Various training schemes are considered. Basically, adding noise to the input data while training improves the performance on the final lower bound. Testing performance with additive noise during training shows higher lower bounds than with multiplicative noise, while additive noise training results in larger gaps between testing and training lower bounds, *i.e.*, an even higher training lower bound. However, this interesting phenomenon indicates that better testing performances may be observed even if the model is overfitting. Additionally, we see that though the VSAE model has more parameters, results of VSAE can yield higher test lower bounds with less overfitting, since sparsity reduces model complexity but keeps the flexibility.

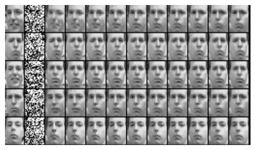
We also train the simplified version VSAE (sVSAE) on Frey faces without the two hidden layers, $\mathbf{h}_d$ and $\mathbf{h}_e$, and by setting $K = 200$ and $\rho = 0.1$. However, not surprisingly, sVSAE (the lower bound is 1061) performs worse than VSAE (1325), since the model only has three layers, roughly equivalent to 560-(20,20)-560, approximately ρK active units. We also find most of the active π_i are close to 0, which we illustrate in Figure 5. The *dictionary size* is 200, *i.e.*, the number of latent Gaussian variables. However, because of the sparse prior, only 10 latent nodes have significant weights ($v_k \geq 0.5$, in our experiment). Thus, the face can be represented as a weighted sum of the ten activated dictionaries faces and the bias face, with a nonlinear transformation (we used sigmoid). From the perspective of a generative model, each face can be encoded as a linear combination of a few 1D Gaussian distributions. The second row of Figure 5 shows some sampled faces. Each image is generated by sampling $p(\mathbf{x}|\mathbf{v}, \mathbf{z})$ after sampling from $p(\mathbf{v}, \mathbf{z}|\mathbf{x})$. This example highlights the flexibility of VSAE, since different data points will activate different latent variables.

5.3. Denoising

We demonstrate the ability of the noisy data model to impute missing data or denoise images one mini-batch at a time. The test images are first corrupted with different scales of additive Gaussian noise. The VSAE model is trained by SGLD with additive noise $\mathcal{N}(0, 0.1^2)$; standard VAE is also considered. The denoising process iterates by using the output of the last iteration as input for the current one. Figure 4(a) shows that standard VAE fails on $+\mathcal{N}(0, 0.4^2)$ noise after running 15 iterations. However,


(a) $+\mathcal{N}(0, 0.4^2)$ noise fails traditional VAE


(b) $\sigma^2 = 0.4^2$


(c) $\sigma^2 = 0.6^2$

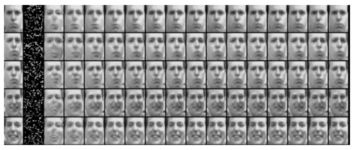

(d) $*(1 - \text{Ber}(0.8))$

Figure 4. Denoise and Imputation: Images of (a-c) are corrupted by additive Gaussian noise, and denoised on SGLD trained VSAE, while (d) is corrupted by Bernoulli noise. (a) VAE trained by standard SGLD; (b-d) VSAE trained by SGLD with additive noise $\mathcal{N}(0, 0.1^2)$.

Table 2. Perplexity on 20 News Test Data

Dim	RepSoftmax	SBN	fDRAN	DocNADE	LDA	sVSAE		
50	953	909	917	896	1091	948		
Dim	DPFA-RBM BCDF	DPFA-SBN SGNHT		DPFA-SBN Gibbs	LDA Gibbs	VSAE $K = 256$	VSAE $K = 128$	VSAE $K = 64$
128-64	893	896		851	893	875	877	885

the noisy trained VSAE can recover the images corrupted by $+\mathcal{N}(0, 0.6^2)$ within 10 iterations. In addition, even with multiplicative Bernoulli noise (equivalent to set 80% of the image pixels to missing), VSAE can successfully impute the missing pixels. It has been demonstrated that the standard VAE has the ability to impute Bernoulli noise (Rezende et al., 2014). We show empirically that training with additive noise can be beneficial to both denoising and imputation. To quantify denoising performance, average pixel-to-pixel (rescaled to interval $[0, 1]$) sum of squared errors (SSEs) between the recovered and original faces are calculated. Standard VAE achieves 4.446, while noisy trained VSAE yields 2.7216.

5.4. Topic Modeling

We present a simple application of VSAE to topic modeling. The input $\mathbf{x} \in \mathbb{Z}_+^V$, consists of count vector data, that represents the frequency of each word in a vocabulary of size V. The sparse binary latent variable $\mathbf{v}$ are the stochastic latent topic distributions. The output layer $\mathbf{y}$, is a multinomial distributed layer, that can be seen as a directed counterpart of the Replicated Softmax model (Hinton & Salakhutdinov, 2009). The corresponding log-likelihood $\log p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z}, \mathbf{v}) = \sum_{i=1}^{V} x_i \log y_i + \mathcal{C}$, where $\mathcal{C}$ is a constant independent of latent variables. Since $\mathbf{y}$ is simulated by a softmax activation function, it is straightforward to derive back-propagation algorithm.

In the variational inference framework, the perplexity is usually estimated by a lower bound on $\exp\left\{\left(-\frac{1}{N}\sum_{i=1}^{N}\frac{1}{L_i}p(\mathbf{x}_i)\right)\right\}$ (Mnih & Gregor, 2014). We first train sVSAE with a number of latent nodes equivalent to 25 in VAE, *i.e.*, $K\rho = 25$. Perplexity results on 20 News are shown in Table 2. Result for sigmoid belief net (SBN), DocNADE, and DPFA are from (Mnih & Gregor, 2014; **?**; Gan et al., 2015). We can see that sVSAE has less power to modeling topic data. However, the VSAE ($\rho = 0.25$ for all values of K) is trained as previously described but for topic modeling can achieve competitive results for two hidden layers models. Note that the results reported by the Gibbs sampling algorithm are from the predictive posterior, which are usually better than its upper bound estimation.

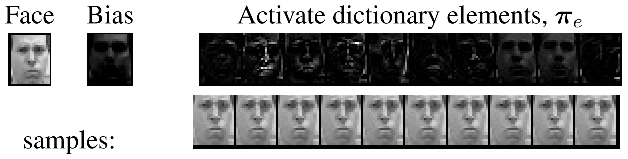


Figure 5. Sparse Representation learned by sVSAE on Frey faces data. Samples generated from one observation, top-left image.

6. Conclusion and Future Work

In this paper, we generalize standard BPFA to the non-linear case, while allowing different input data types. Meanwhile, a scalable inference framework based on variational sparse auto-encoders is developed, that achieves competitive performance on benchmark datasets. Since the VAE framework can be generalized and applied to a large class of complex models (non-conjugate), one possible area of future research can be to explore other non-parametric Bayesian models that may fit our framework. Another possibility may be to establish theoretical guarantees for corrupting hidden layers within the variational auto-encoder framework.

References

Bengio, Yoshua, Yao, Li, Alain, Guillaume, and Vincent, Pascal. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907, 2013.

Blei, David M and Jordan, Michael I. Variational methods for the dirichlet process. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 12. ACM, 2004.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Carin, Lawrence, Blei, David M, and Paisley, John W. Variational inference for stick-breaking beta process priors. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 889–896, 2011.

Chen, Ning, Zhu, Jun, Chen, Jianfei, and Zhang, Bo. Dropout training for support vector machines. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Fan, Kai, Wang, Ziteng, Beck, Jeff, Kwok, James, and Heller, Katherine. Fast second-order stochastic backpropagation for variational inference. In *Advances in Neural Information Processing Systems*, 2015.

Gan, Zhe, Chen, Changyou, Henao, Ricardo, Carlson, David, and Carin, Lawrence. Scalable deep poisson factor analysis for topic modeling. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1823–1832, 2015.

Ghahramani, Zoubin and Griffiths, Thomas L. Infinite latent feature models and the indian buffet process. In *Advances in neural information processing systems*, pp. 475–482, 2005.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pp. 1607–1614, 2009.

Hjort, Nils Lid. Nonparametric bayes estimators based on beta processes in models for life history data. *The Annals of Statistics*, pp. 1259–1294, 1990.

Hornik, Kurt, Stinchcombe, Maxwell, and White, Halbert. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, Diederik P, Salimans, Tim, and Welling, Max. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015.

Larochelle, Hugo and Lauly, Stanislas. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*, pp. 2708–2716, 2012.

Maaten, Laurens, Chen, Minmin, Tyree, Stephen, and Weinberger, Kilian Q. Learning with marginalized corrupted features. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 410–418, 2013.

Martens, James. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 735–742, 2010.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 1791–1799, 2014.

Neal, Radford M et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162, 2011.

Nesterov, Yurii. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pp. 372–376, 1983.

Ngiam, Jiquan, Coates, Adam, Lahiri, Ahbik, Prochnow, Bobby, Le, Quoc V, and Ng, Andrew Y. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 265–272, 2011.

Paisley, John and Carin, Lawrence. Nonparametric factor analysis with beta process priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 777–784. ACM, 2009.

Paisley, John W, Zaas, Aimee K, Woods, Christopher W, Ginsburg, Geoffrey S, and Carin, Lawrence. A stick-breaking construction of the beta process. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 847–854, 2010.

Polyak, Boris Teodorovich. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Rezende, Danilo J, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1278–1286, 2014.

Shah, Amar, Knowles, David, and Ghahramani, Zoubin. An empirical study of stochastic variational inference algorithms for the beta bernoulli process. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1594–1603, 2015.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Teh, Yee Whye, Thiéry, Alexandre, and Vollmer, Sebastian. Consistency and fluctuations for stochastic gradient langevin dynamics. *arXiv preprint arXiv:1409.0578*, 2014.

Thibaux, Romain and Jordan, Michael I. Hierarchical beta processes and the indian buffet process. In *International conference on artificial intelligence and statistics*, pp. 564–571, 2007.

Van Der Vaart, Aad W and Wellner, Jon A. *Weak Convergence*. Springer, 1996.

Wager, Stefan, Wang, Sida, and Liang, Percy S. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pp. 351–359, 2013.

Welling, Max and Teh, Yee W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.

Zhou, Mingyuan, Chen, Haojun, Ren, Lu, Sapiro, Guillermo, Carin, Lawrence, and Paisley, John W. Nonparametric bayesian dictionary learning for sparse image representations. In *Advances in neural information processing systems*, pp. 2295–2303, 2009.