Scalable Non-linear Beta Process Factor Analysis

Kai Fan
Duke University
kai.fan@stat.duke.edu

Katherine Heller
Duke University
kheller@stat.duke.com

Abstract

We propose a non-linear extension of the factor analysis with beta process priors. This non-linear Beta process factor analysis (nBPFA) model allows the data vector to be represented by processing an invertible non-linear transformation on standard factor decomposition. We develop a scalable variational inference framework for large scale datasets, which benefits from the idea of generalizing variational auto-encoder (VAE) by introducing another latent binary variable with sparse constraint. This framework also allows the inputs to be real valued, binary or count vector data. We empirically test our algorithm on images and document datasets, and demonstrates competitive results, especially showing better performance preventing overfitting.

1 Preliminary

We skip the definition of the beta process as a Lévy process, since sampling directly from the infinite beta process is difficult and not relevant to our analysis. We briefly review a marginalized approach based on [4, 7], which takes two scalar parameters for the Beta process and can be derived in the similar way as Chinese restaurant process. Denote the two parameter beta process as $\text{BP}(a, b, \mathcal{H}_0)$, where $a, b > 0$ and $\mathcal{H}_0$ is the base measure. Thus the BP can be represented as follows.

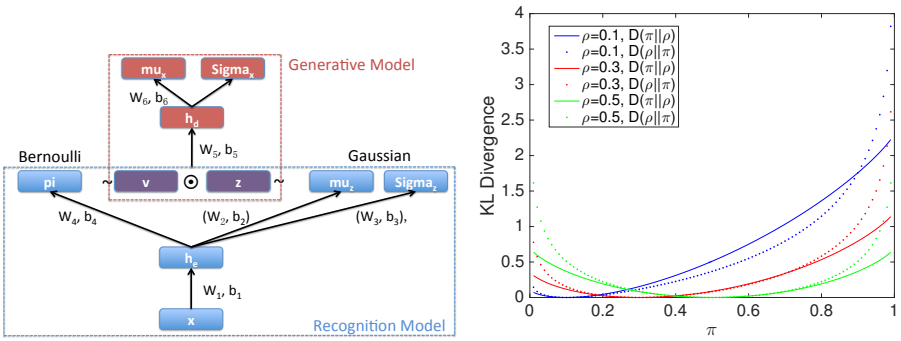
$$\mathcal{H}(v) = \sum_{k=1}^{K} \xi_k \delta_{v_k}(v), \quad \xi_k \sim \text{Beta}(a/K, b(K-1)/K), \quad v_k \sim \mathcal{H}_0 \tag{1}$$

with a well-defined base measure when $K \to \infty$. Then the generative process of a binary matrix with a BP prior can follow two steps similar to the Indian Buffet Process when $b = 1$ [5].

As in to (1), a finite approximation to the beta process can be derived by simply setting K as a large but finite number, which is similar to the finite approximation of Dirichlet process. In this case, the nice conjugacy property of this representation means posterior computation is analytical. Thus, a BPFA model can be naturally defined and readily inferred. Given the data vector $\mathbf{x}_i \in \mathbb{R}^D$, the latent Gaussian variable and binary variable $\mathbf{z}_i, \mathbf{v}_i \in \mathbb{R}^K$, and the weight matrix $\mathbf{W} \in \mathbb{R}^{D \times K}$, we have the following factor analysis model.

$$\mathbf{x}_i = \mathbf{W}(\mathbf{z}_i \odot \mathbf{v}_i) + \boldsymbol{\epsilon}_i, \quad \boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \gamma^{-1}\mathbf{I}_D) \tag{2}$$

where $\mathbf{z}_i$ follows a zero-mean Gaussian distribution with an isotropic covariance matrix represented as $\sigma_z^2\mathbf{I}$. Each element of the weight matrix may also follow a zero-mean Gaussian distribution. The prior for each element of $\mathbf{v}_i$ is a finite approximation of beta process, i.e. $\mathbf{v}_{ik} \sim \text{Ber}(\xi_k), \xi_k \sim \text{Beta}(a/K, b(K-1)/K)$. Notice that in the limit $K \to \infty$, the number of elements of $\mathbf{v}_i$ that are non-zero is a random variable drawn from distribution $\text{Poisson}(a/b)$.



(a) (b)

Figure 1: (a) Unrolled VSAE for binary input: $\sim$ means sampling and $\rightarrow$ means nonlinear or linear mapping, where $\mathbf{v} \sim \text{Ber}(\boldsymbol{\pi})$ and $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$. (b) Comparison between different sparse penalties.

2 Non-linear BPFA

In this section, we generalize BPFA to a non-linear Beta Process Factor Analysis model by simply imposing a non-linear transformation f on the linear combination, i.e.

$$\mathbf{x}_i = f(\mathbf{W}(\mathbf{z}_i \odot \mathbf{v}_i)) + \boldsymbol{\epsilon}_i \tag{3}$$

where f is an invertible mapping, e.g., sigmoid or hyperbolic tangent function or their composition. This invertibility assumption can be relaxed in practice, e.g, multi-layer perceptron (MLP) is usually used, but MLP may not be invertible when applying rectifier activation, which can be smoothly approximated by an invertible function $\log(1+e^x)$. Notice that [4] developed an efficient variational inference EM algorithm due to the conjugacy of BPFA, whereas the non-linear extension is usually non-conjugate. Thus we slightly modify the hierarchical generative model to develop the stochastic variational inference framework by jointly learning the parameters of both generative and variational models.

2.1 Scalable Variational Inference

We consider that $\mathbf{x}_i \sim \mathcal{N}\left(f(\mathbf{W}(\mathbf{z}_i \odot \mathbf{v}_i)), \text{diag}(\boldsymbol{\sigma}_x)\right)$ by absorbing the error term into the distribution. Unlike traditional BPFA, the covariance matrix we assumed is not necessarily isotropic. By writing $\boldsymbol{\xi} = (\xi_1, \xi_2, ..., \xi_K)$, we consider the probabilistic model $p(\mathbf{x}, \mathbf{z}, \mathbf{v}, \boldsymbol{\xi})$, where the other notation used here is exactly consistent with the previous section. Therefore the hierarchical generative process is quite similar, except for a non-linear mapping of the result of original BPFA. We then construct our stochastic variational inference based on the framework of a variational auto-encoder (VAE) [2].

Using Bayes' rule, we have the exact factorized form of generative model $p(\boldsymbol{\xi}|\mathbf{v})p(\mathbf{x}|\mathbf{z}, \mathbf{v})p(\mathbf{z})p(\mathbf{v})$. Thus we also propose a variational latent variable posterior or recognition model $q(\mathbf{z}, \mathbf{v}, \boldsymbol{\xi}|\mathbf{x})$ with structured form $q(\mathbf{z}|\mathbf{x})q(\boldsymbol{\xi}|\mathbf{v})q(\mathbf{v}|\mathbf{x})$. If we further assume $q(\boldsymbol{\xi}|\mathbf{v})$ takes the unknown true posterior $p(\boldsymbol{\xi}|\mathbf{v})$, we can compute the lower bound as follows.

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}, \mathbf{v}, \boldsymbol{\xi}) - \log q(\mathbf{z}, \mathbf{v}, \boldsymbol{\xi}|\mathbf{x})] \\ &= \mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z}, \mathbf{v})] - D(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) - D(q(\mathbf{v}|\mathbf{x})||p(\mathbf{v}))\end{aligned}$$

where $p(\mathbf{v}) = \int p(\mathbf{v}|\boldsymbol{\xi})p(\boldsymbol{\xi})\text{d}\boldsymbol{\xi} = \frac{\text{Beta}(a/K+\mathbf{v}, b(K-1)/K+1-\mathbf{v})}{\text{Beta}(a/K, b(K-1)/K)}$, i.e., $p(\mathbf{v}) \sim \text{Ber}\left(\frac{a}{a+b(K-1)}\right)$. Thus, we can safely remove $\boldsymbol{\xi}$ and denote $\rho = \frac{a}{a+b(K-1)}$. The $\text{Ber}(\rho)$ is a sparse prior imposed to each element of $\mathbf{v}$. Smaller ρ favors a sparser prior. It means that given a finite but large K, we can control the sparsity by tuning the hyper-parameter a, b.

In our proposed recognition model, the variational distribution is restricted to belong to a family of distributions of simpler form than true posterior, but preferably flexible enough to contain the true posterior as a solution. The approximate posterior $q(\mathbf{z}|\mathbf{x})$ follows $\mathcal{N}(\boldsymbol{\mu}_z, \text{diag}(\boldsymbol{\sigma}_z))$ rather than the zero-mean isotropic Gaussian distribution in standard BPFA; $q(\mathbf{v}|\mathbf{x})$ follows $\text{Ber}(\boldsymbol{\pi}_e)$ (in the sense of element-wise). The above assumption guarantees that the true posterior belongs to the same family.

To induce more flexibility, we use the trick of embedded parameters via a multi-layer perceptron, i.e., $\boldsymbol{\mu}_z = \text{MLP}_1(\mathbf{x})$, $\boldsymbol{\sigma}_z = \text{MLP}_2(\mathbf{x})$ and $\boldsymbol{\pi}_e = \text{MLP}_3(\mathbf{x})$. This idea follows VAE [2] except for another latent Bernoulli variable with the BP prior. This is actually an *encoder process*. Recall the generative model $\mathcal{N}\left(f(\mathbf{W}(\mathbf{z}_i \odot \mathbf{v}_i)), \text{diag}(\boldsymbol{\sigma}_x)\right)$, we can define the *decoder process* by simply imposing f as an MLP. To infer the error variance, we also apply a similar trick $\boldsymbol{\sigma}_x = \text{MLP}(\mathbf{z})$. We call this framework variational sparse auto-encoder (VSAE).

Compared with standard VAE, the difference of a feed-forward process for recognition model merely needs two extra steps (full derivation in Appendix).

$$\begin{aligned}\boldsymbol{\pi}_e &= \text{sigmoid}(W\mathbf{h}_e + b)\\ \mathbf{z} &\leftarrow \mathbf{z} \odot \mathbf{v}, \text{ where } \mathbf{v} \sim \text{Ber}(\boldsymbol{\pi}_e)\end{aligned}$$

where $\boldsymbol{\pi}_e$ is the Bernoulli parameter for $\mathbf{v}$ in recognition model. The unrolled VSAE model can be approximately represented as a deep neural network shown in Fig. 1. The input and the output layers have D nodes, and the middle two layers have K nodes. Notice that for real valued input, we have two output layers for $\boldsymbol{\mu}_x$ and $\boldsymbol{\sigma}_x$ respectively. The number of nodes for hidden encoder and decoder layers can be empirically set. For the backpropagation process associated with different lower bounds, we notice the difference by using KL divergence for factorized variational distribution, i.e.

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

where the second term is imposed to control the sparsity of latent variable $\mathbf{v}$. Notice that the KL divergence of a Bernoulli reaches its minimum of 0 if $\pi_i = \rho$. Explicitly, when the parameter of sparse prior ρ becomes smaller, more sparsity will be brought to the model. In fact, the sparsity constraint in standard auto-encoders can actually be formulated as $D_{KL}(p(\mathbf{v})||q_{\boldsymbol{\psi}}(\mathbf{v}|\mathbf{x}))$. However, when π approaches 0 or 1, this penalty may blow up to infinity, thus requiring a further tuning weight, λ, to control the magnitude of the KL divergence at the same time. Instead, our KL divergence term in the lower bound can reach its maximum either at $\log(1/\rho)$ or $\log(1/(1-\rho))$. An additional weight parameter is not necessary here (see Appendix for the difference in stochastic backpropagation).

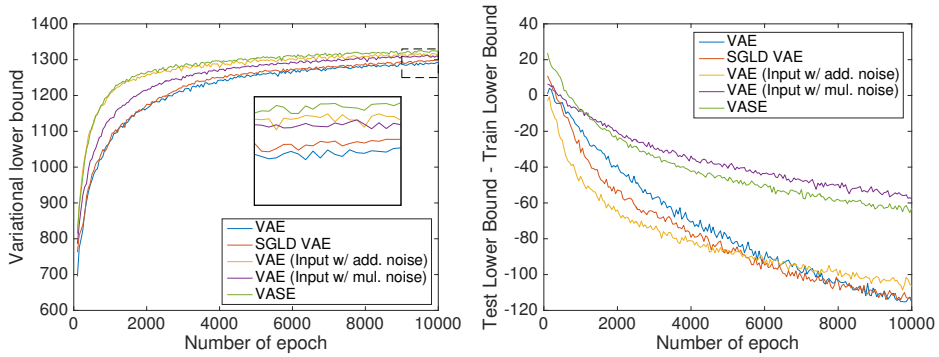
2.2 Discussion

Any stochastic optimization for variational inference is allowed in this framework so the algorithm is scalable for large datasets. Weight uncertainty considered in BPFA is omitted here for simplicity. In fact, only the weight matrix associated with $\mathbf{z} \odot \mathbf{v}$ in the deep network needs uncertainty, e.g., $\mathbf{W}_5$ in Fig 1(a). However, the technique in [1] of inferring weight uncertainty in neural networks can straightforwardly apply to all weight matrices. In addition, we can simplify VSAE by removing the hidden encoder and decoder layers of VSAE, thus more intuitively explaining the non-linear BPFA.

3 Experiments

We first train the VSAE on the dataset Frey face, corresponding to real value inputs. The structure is 560-200-(200,200)-200-560 where $\rho = 0.1$ is set to be $K = 20$ as in standard VAE. For comparison, we also train a standard VAE (560-200-20-200-560). The results are shown in Fig 2. Though the VSAE model has more parameters, the training results of VSAE can achieve both higher test lower bound with less overfitting, while the other four training schemes on standard VAE will suffer at least one problem.

We also train the simplified VSAE (sVSAE) on Frey Face, with $K = 200$ and $\rho = 0.1$. However, it is not surprising that sVSAE (lower bound is 1061 in Frey face dataset) will perform worse than VSAE (1325), since the model can only be unrolled to three layers (roughly equivalent to 560-20-560). We also find most of the activated π_i is close to 0, and it is easy to visualize the result (Fig. 3). The dictionary size is 200 which is the number of latent Gaussian variables. However, because of the sparse prior, only ten latent nodes have significant weights (≥ 0.5 in our experiment). Thus, the face can be represented as weighted sum of the ten activated dictionaries faces and the bias face, with a nonlinear transformation (e.g. sigmoid). From the perspective of generative models, each



(a) Test LowerBound (b) Overfitting

Figure 2: Performance Comparison

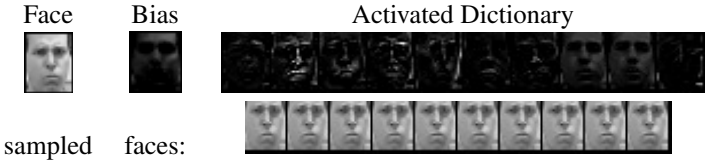


Figure 3: Sparse Representation

face can be represented as a linear combination of a few 1D Gaussian distributions. The second row of Fig. 3 shows a few sampled faces. The are generated by first drawing some random samples from latent 1D distributions and processing a nonlinear transformation.

Besides the above description, a simple application of sVSAE is topic modeling. The input $\mathbf{x} \in \mathbb{Z}_+^V$ is count vector data, to represent the frequency of each word, where V is the vocabulary size. The sparse binary latent variable $\mathbf{v}$ is the stochastic hidden topics. The output layer $\mathbf{y}$ is a multinomial distributed layer and can be seen as directed counterparts of the Replicated Softmax model. Thus we have the 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 softmax activation function, it is trivial to derive backpropagation (see Appendix).

In the variational inference framework, the perplexity is usually estimated by a lower bound on $e^{\left(-\frac{1}{N}\sum_{i=1}^{N}\frac{1}{L_i}p(\mathbf{x}_i)\right)}$. We first train sVSAE with equivalent 25 latent nodes as VAE, i.e., $K\rho = 25$. The perplexity result on 20 NEWS is shown in Table 1 (Except our result, other values are from [3, 6]). The VSAE ($\rho = 0.25$ for all Ks) trained as in the previous section for topic modeling can achieve competitive results for two hidden layers model. Notice the result reported by the Gibbs sampling algorithm is the predictive posterior, which is usually better than its upper bound estimation.

Table 1: 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

4 Conclusion

In this paper, we generalize the standard BPFA to a non-linear extension, while allows various input data types. Meanwhile, a scalable inference framework based on variational sparse auto-encoder is developed for this model and achieves competitive performance on some benchmark datasets. The possible area of future research may apply this model to more practical implementation, such as image denoising or inpainting.

References

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

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

[3] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1791–1799, 2014.

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

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

[6] R. Henao D. Carlson Z. Gan, C. Chen and L. Carin. Scalable deep poisson factor analysis for topic modeling. In *ICML*, 2015.

[7] Mingyuan Zhou, Haojun Chen, Lu Ren, Guillermo Sapiro, Lawrence Carin, and John W Paisley. Non-parametric bayesian dictionary learning for sparse image representations. In *Advances in neural information processing systems*, pages 2295–2303, 2009.

Appendix

Variational Sparse Auto-Encoding

We derive the feedforward and backpropagation for binary input, where the continuos input is a straightforward generalization. The feed forward process is as follows.

$$\begin{aligned}
\mathbf{h}_e &= \tanh(W_1\mathbf{x} + b_1) \\
\boldsymbol{\mu}_e &= W_2\mathbf{h}_e + b_2 \\
\log \boldsymbol{\sigma}_e &= 0.5 * (W_3\mathbf{h}_e + b_3) \\
\boldsymbol{\pi} &= \text{sigmoid}(W_4\mathbf{h}_e + b_4) \\
\boldsymbol{\epsilon} &\sim \mathcal{N}(0, \mathbf{I}_{d_z}) \\
\mathbf{z} &= \boldsymbol{\mu}_e + \boldsymbol{\sigma}_e \odot \boldsymbol{\epsilon} \\
\mathbf{v} &\sim \text{Ber}(\boldsymbol{\pi}) \\
\mathbf{h}_d &= \tanh(W_5(\mathbf{z} \odot \mathbf{v}) + b_5) \\
\mathbf{y} &= \text{sigmoid}(W_6\mathbf{h}_d + b_6).
\end{aligned}$$

We develop the practical stochastic backpropagation as follows.

$$\begin{aligned}
\boldsymbol{\delta}_6 &= \mathbf{x} \odot (1-\mathbf{y}) - (1-\mathbf{x}) \odot \mathbf{y} \\
\nabla_{W_6} &= \boldsymbol{\delta}_6 \mathbf{h}_d^\top, \quad \nabla_{b_6} = \boldsymbol{\delta}_6 \\
\boldsymbol{\delta}_5 &= (W_6^\top \boldsymbol{\delta}_6) \odot (\mathbf{1} - \mathbf{h}_d \odot \mathbf{h}_d) \\
\nabla_{W_5} &= \boldsymbol{\delta}_5 (\mathbf{z} \odot \mathbf{v})^\top, \quad \nabla_{b_5} = \boldsymbol{\delta}_5 \\
\boldsymbol{\delta}_4 &= (W_5^\top \boldsymbol{\delta}_5 \odot \mathbf{z} - \delta\mathcal{S}) \odot \boldsymbol{\pi} \odot (1 - \boldsymbol{\pi}) \\
\nabla_{W_4} &= \boldsymbol{\delta}_4 \mathbf{h}_e^\top, \quad \nabla_{b_4} = \boldsymbol{\delta}_4 \\
\boldsymbol{\delta}_3 &= 0.5 * [(W_5^\top \boldsymbol{\delta}_5) \odot (\mathbf{z} - \boldsymbol{\mu}_e) \odot \mathbf{v} + \mathbf{1} - \boldsymbol{\sigma}_e^2] \\
\nabla_{W_3} &= \boldsymbol{\delta}_3 \mathbf{h}_e^\top, \quad \nabla_{b_3} = \boldsymbol{\delta}_3 \\
\boldsymbol{\delta}_2 &= W_5^\top \boldsymbol{\delta}_5 \odot \mathbf{v} - \boldsymbol{\mu}_e \\
\nabla_{W_2} &= \boldsymbol{\delta}_2 \mathbf{h}_e^\top, \quad \nabla_{b_2} = \boldsymbol{\delta}_2 \\
\boldsymbol{\delta}_1 &= (W_2^\top \boldsymbol{\delta}_2 + W_3^\top \boldsymbol{\delta}_3 + W_4^\top \boldsymbol{\delta}_4) \odot (\mathbf{1} - \mathbf{h}_e \odot \mathbf{h}_e) \\
\nabla_{W_1} &= \boldsymbol{\delta}_1 \mathbf{x}^\top, \quad \nabla_{b_1} = \boldsymbol{\delta}_1.
\end{aligned}$$

where $\delta\mathcal{S} = \log \frac{(1-\rho)\boldsymbol{\pi}}{\rho(1-\boldsymbol{\pi})}$. If we consider the traditional sparse penalty $\lambda D_{KL}(p(\mathbf{v})||q(\mathbf{v}|\mathbf{x}))$ in auto-encoder, $\delta\mathcal{S} = \lambda\left(\frac{1-\rho}{1-\boldsymbol{\pi}} - \frac{\rho}{\boldsymbol{\pi}}\right)$.

Apply VSAE to Topic Modeling

Input $\mathbf{x} \in \mathbb{N}^V$ is count data, where V is the number of vocabulary. After the feed forward process, the output

$$\mathbf{y} = \text{softmax}(W_6\mathbf{h}_d + b_6)$$

where $\sum y_i = 1$ is the probability vector. The log likelihood is

$$\log p(\mathbf{x}|\mathbf{z}, \mathbf{v}) = \sum_{i=1}^{V} x_i \log y_i + C$$

Thus, we need modify $\boldsymbol{\delta}_6$ in the backpropagation, where the other part is the same.

$$\boldsymbol{\delta}_6 = \mathbf{x} - \mathbf{y}\left(\sum_{i=1}^{V} x_i\right)$$