# Convolutional Factor Analysis with Fast Gibbs Sampling Algorithms

**Zhe Gan(zg27)**
Department of ECE, Duke University
zhe.gan@duke.edu

## Abstract

This project aims to implement the convolutional factor-analysis model proposed in [1]. The model parameters are learned within a Bayesian setting, employing Gibbs sampling, while the number of factors is inferred via the beta process. Preliminary experimental results on MNIST and Caltech 101 datasets demonstrate that the proposed model can achieve good performance.

## 1 Introduction

There has been significant recent interest in multi-layered or "deep" models for representation of general data, with a particular focus on imagery. Popular deep learning models include: Deep Belief Network [4], Deep Boltzmann Machine [7], Convolutional Neural Networks [5] and Deconvolutional Networks [9], etc. Among these, deep covolutional networks have demonstrated excellent performance on image classification tasks. There are at least two key components of this model: 1) convolution operator, which considers all possible shifts of canonical filters; 2) a deep architecture, in which the features of a given layer serve as the inputs to the next layer above.

However, the training of a convolutional neural network is typically imposed as an optimization problem. In this course project, I will focus on developing Bayesian generative model for deep convolutional dictionary learning, based on the idea proposed in [1]. As stated in the paper, some advantages of the proposed model are: (*i*) the number of factors needed is inferred from the data by an Indian Buffet Process(IBP) [3] or Beta Process (BP) [6] construction; (*ii*) fast computations are performed using Gibbs sampling, where the convolution operation is exploited directly within the update equations; and (*iii*) sparseness is imposed on the factor loadings and factor scores, via a Bayesian generalization of the L1 regularizer.

Recently, I am focusing my research on designing efficient and scalable Bayesian inference algorithms for deep learning models. Therefore, the course project will be helpful for my research.

## 2 Model Formulation

The proposed model is applicable to general data, for which a convolutional dictionary representation is appropriate. One may, for example, apply the model to one-dimensional signals such as audio, or to two-dimensional imagery. In this project, we focus on imagery, and hence assume two-dimensional signals and convolutions.

Assume $N$ images $\{\mathbf{X}_n\}_{n=1,...,N}$, with $\mathbf{X}_n \in R^{n_y \times n_x \times K_c}$, where $K_c$ is the number of color channels (*e.g.*, for gray-scale images $K_c = 1$, while for RGB images $K_c = 3$); the images are analyzed jointly to learn the convolutional dictionary $\mathcal{D} = \{\boldsymbol{d}_k\}_{k=1,...,K}$. In practice, the number of dictionary elements $K$ is made large, and we wish to infer the subset of $\mathcal{D}$ that is actually needed to represent $\mathbf{X}_n$. The dictionary elements are designed to capture local structure within $\mathbf{X}_n$, and all possible two-dimensional (spatial) shifts of the dictionary elements are considered for representation

of $\mathbf{X}_n$. Specifically, consider the model

$$\mathbf{X}_n = \sum_{k=1}^{K} b_{nk} \mathbf{W}_{nk} * \boldsymbol{d}_k + \boldsymbol{\epsilon}_n, \tag{1}$$

where $*$ is the convolution operator, and $b_{nk} \in \{0, 1\}$ indicates whether $\boldsymbol{d}_k$ is used to represent $\mathbf{X}_n$, and $\boldsymbol{\epsilon}_n$ represents the residual. Considering $\boldsymbol{d}_k \in R^{n'_y \times n'_x \times K_c}$ (typically $n'_x \ll n_x$ and $n'_y \ll n_y$), the corresponding weights $\mathbf{W}_{nk}$ are of size $(n_y - n'_y + 1) \times (n_x - n'_x + 1)$, allowing for all possible shifts. We impose within the model that the $\{w_{nki}\}_{i \in \mathcal{S}}$ are sparse or nearly sparse, such that most $w_{nki}$ are sufficiently small to be discarded without significantly affecting the reconstruction of $\mathbf{X}_n$, where the set $\mathcal{S}$ contains all possible indexes for dictionary shifts.

Within a Bayesian construction, the priors for the model may be represented as

$$b_{nk} \sim \text{Bernoulli}(\pi_k), \qquad \pi_k \sim \text{Beta}(1/K, b), \tag{2}$$

$$w_{nki} \sim N(0, 1/\alpha_{nki}), \qquad \boldsymbol{d}_k \sim \prod_{j=1}^{J} \mathcal{N}(0, 1/\beta_j), \qquad \epsilon_n \sim \mathcal{N}(0, \mathbf{I}_P \gamma_n^{-1}), \tag{3}$$

$$\alpha_{nki} \sim \text{Ga}(e, f), \qquad \gamma_n \sim \text{Ga}(c, d), \qquad \beta_j \sim \text{Ga}(g, h) \tag{4}$$

where $J$ denotes the number of pixels in the dictionary elements $\boldsymbol{d}_k$ and $d_{kj}$ is the $j$th component of $\boldsymbol{d}_k$. $\text{Ga}(\cdot)$ denotes the gamma distribution. The integer $P$ denotes the number of pixels in $\mathbf{X}_n$, and $\mathbf{I}_P$ represents a $P \times P$ identity matrix. $\{b, c, d, e, f, g, h\}$ are hyperparameters. In the limit $K \to \infty$, and upon marginalizing out $\{\pi_k\}_{k=1,\ldots,K}$, the above model corresponds to the IBP [2] [8]. While the model may look somewhat complicated, local conjugacy admits Gibbs sampling or variational Bayes inference.

## 3 Inference

The local conditional posterior distribution for all parameters of the model is manifested in closed form, yielding efficient Gibbs sampling algorithm. The FFT is leveraged to accelerate computation of the covolution operations.

For each MCMC iteration, the samples are drawn from the following conditional distributions:

**(1) For** $b_{nk}$ and $\mathbf{W}_{nk}$ (*i.e.* $\{w_{nki}\}_{i \in \mathcal{S}}$) : we have $p(b_{nk} = 1|-) = \tilde{\pi}_{nk}$, and $p(w_{nki}, i \in \mathcal{S}|-) = (1 - b_{nk})\mathcal{N}(0, \alpha_{nki}^{-1}) + b_{nk}\mathcal{N}(\mu_{nki}, \Sigma_{nki})$, where $\Sigma_{nki} = (\boldsymbol{d}_{ki}^\top \boldsymbol{d}_{ki} \gamma_n + \alpha_{nki})^{-1}$, $\mu_{nki} = \Sigma_{nki} \gamma_n \mathbf{X}_{nki}^\top \boldsymbol{d}_{ki}$, with $\mathbf{X}_{nki} = \mathbf{X}_{-n} + b_{nk} \boldsymbol{d}_{ki} w_{nki}$, and

$$\frac{\tilde{\pi}_{nk}}{1 - \tilde{\pi}_{nk}} = \frac{\pi_k}{1 - \pi_k} \cdot \frac{\mathcal{N}(\mathbf{X}_{nk}|\mathbf{W}_{nk} * \boldsymbol{d}_k, \gamma_n^{-1}\mathbf{I}_P)}{\mathcal{N}(\mathbf{X}_{nk}|0, \gamma_n^{-1}\mathbf{I}_P)} \tag{5}$$

Here $\mathbf{X}_{nk} = \mathbf{X}_{-n} + \mathbf{W}_{nk} * \boldsymbol{d}_k$, $\mathbf{X}_{-n} = \mathbf{X}_n - \sum_{k=1}^{K} b_{nk} \mathbf{W}_{nk} * \boldsymbol{d}_k$ and $b_{nk}$ is the most recent sample. Taking advantage of the convolution property, we simultaneously update the posterior mean and covariance of the coefficients for all the shifted versions of one dictionary element. Consequently,

$$\boldsymbol{\Sigma}_{nk} = 1 \oslash (\gamma_n ||\boldsymbol{d}_k||_2^2 b_{nk} + \boldsymbol{\alpha}_{nk}), \tag{6}$$

$$\boldsymbol{\mu}_{nk} = b_{nk} \gamma_n \boldsymbol{\Sigma}_{nk} \odot (\mathbf{X}_{-n} * \boldsymbol{d}_k + ||\boldsymbol{d}_k||_2^2 \mathbf{W}_{nk}) \tag{7}$$

where both of $\boldsymbol{\Sigma}_{nk}$ and $\boldsymbol{\mu}_{nk}$ have the same size with $\mathbf{W}_{nk}$. The symbol $\odot$ is the element-wise product operator and $\oslash$ the element-wise division operator.

**(2) For** $\boldsymbol{d}_k$: we have $p(d_{kj}|-) = \mathcal{N}(\xi_{kj}, \Lambda_{kj})$, where $\xi_{kj}$ and $\Lambda_{kj}$ is the $j$th element of the following vectors $\boldsymbol{\xi}_k$ and $\boldsymbol{\Lambda}_k$, respectively.

$$\boldsymbol{\Lambda}_k = 1 \oslash \left( \sum_{n=1}^{N} \gamma_n b_{nk} ||\mathbf{W}_{nk}||_2^2 + \beta_k \right), \tag{8}$$

$$\boldsymbol{\xi}_k = \boldsymbol{\Lambda}_k \odot \left( \sum_{n=1}^{N} b_{nk} \gamma_n (\mathbf{X}_{-n} * \mathbf{W}_{nk} + \boldsymbol{d}_k ||\mathbf{W}_{nk}||_2^2) \right) \tag{9}$$

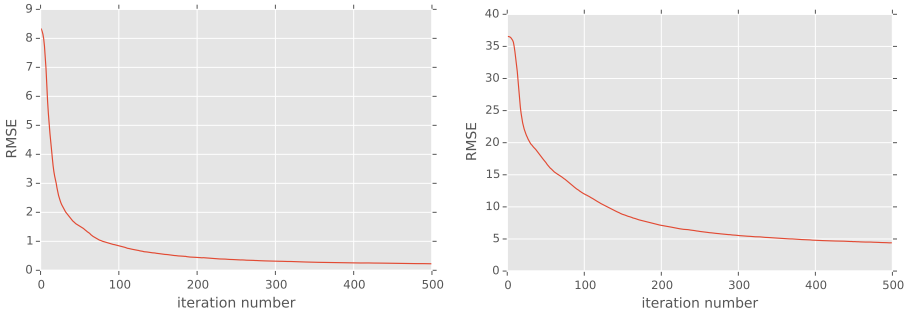


Figure 1: Root mean squared error as a function of iteration numbers. (Left) MNIST dataset. (Right) Caltech101 face dateset.

The remaining Gibbs updates are relatively standard, and are provided below. (*i*) Sampling $\pi_k$: $p(\pi_k|-) = \text{Beta}(\tilde{a}, \tilde{b})$, where $\tilde{a} = \sum_{n=1}^{N} b_{nk} + 1/K$, $\tilde{b} = N + b - \sum_{n=1}^{N} b_{nk}$; (*ii*) Sampling $\gamma_n$: $p(\gamma_n|-) = \text{Ga}(\tilde{c}, \tilde{d})$, where $\tilde{c} = c + 1/2$, $\tilde{d} = d + ||\mathbf{X}_n||_2^2/2$; (*iii*) Sampling $\alpha_{nki}$: $p(\alpha_{nki}|-) = \text{Ga}(\tilde{e}, \tilde{f})$, where $\tilde{e} = e + 1/2$, $\tilde{f} = f + w_{nki}^2/2$; (*iv*) Sampling $\beta_{kj}$: $p(\beta_{kj}|-) = \text{Ga}(\tilde{g}, \tilde{h})$, where $\tilde{g} = g + 1/2$, $\tilde{h} = f + d_{kj}^2/2$.

During the inference, we perform efficient block Gibbs sampling by sampling each dictionary element and corresponding hidden units given other parameters fixed. Throughout the inference procedure, no matrix inversion is needed since the dictionary and hidden units (factor scores) have been factorized. In practice, we typically do not need a large number of iterations to achieve useful results (we focus on finding an MAP solution, rather than a full posterior).

## 4 Experiments

We present the performance of the proposed model on a small subset of the two widely used datasets: MNIST and Caltech 101. In all experiments, we set $b = e = g = 1$, $c = d = h = 10^{-6}$, and $f = 10^{-3}$. The truncation level of the number of factors are set to 36. Since we only consider gray-scale images, and therefore $K_c = 1$. We collect 300 samples, after first computing and discarding 200 burn-in samples.

### 4.1 MNIST Examples

We first consider the widely studied MNIST dataset [1], in which we perform analysis on 1000 images, each $28 \times 28$, for digits 0 through 9 (for each digit, we randomly select 10 images). The dictionary $\boldsymbol{d}_k$ are of size $7 \times 7$.

Figure 2: Performance on the MNIST dataset. (Left) Training data. (Middle) Reconstructed digits. (Right) Learned dictioinaries.

[1] http://yann.lecun.com/exdb/mnist/

The root mean squared error (RMSE) is shown in Figure 1. After 500 iterations, we can achieve an RMSE of 0.23. As can be seen in Figure 2, the convolutional factor analysis can reconstruct the digits almost perfectly. However, using a model like PCA can make the reconstructed images blurry.

### 4.2 Caltech 101 Face Examples

We next consider the Caltech 101 dataset [2], in which we perform analysis on 128 face images, and each image is resized to $32 \times 32$. The dictionary $\boldsymbol{d}_k$ size is also set to $7 \times 7$.

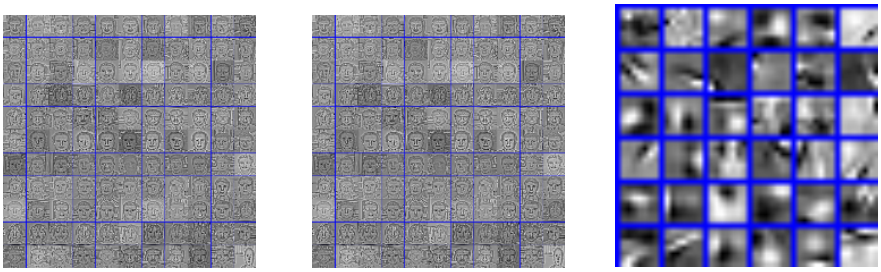

Figure 3: Performance on the Caltech 101 face dataset. (Left) Training data. (Middle) Reconstructed digits. (Right) Learned dictioinaries.

The root mean squared error (RMSE) is shown in Figure 1. After 500 iterations, we can achieve an RMSE of 4.40. As can be seen in Figure 3, the faces are almost perfectly reconstructed.

Now, we compare our implemented convolutional factor analysis (CFA) algorithm with simple PCA. For fair comparison, we select the first $K = 36$ principal components of PCA to recover the input images. The results are summarized in Table 1. As can be seen, the performance of CFA is much better compared with simple PCA.

| Dataset | CFA | PCA |
|---|---|---|
| MNIST | 0.23 | 2.47 |
| Caltech 101 Face | 4.40 | 23.86 |

Table 1: Comparison between CFA and PCA

## 5 Conclusion

In this course project, we developed the convolutional factor-analysis model and implemented the Gibbs sampling algorithm for posterior inference. Experimental results on MNIST and Caltech 101 Face datasets show that the CFA model can achieve good performance.

We tried to implement the deep extension of the CFA model, but failed to achieve good performances. This can be an interesting future work.

## References


[1] Bo Chen, Gungor Polatkan, Guillermo Sapiro, Lawrence Carin, and David B Dunson. The hierarchical beta process for convolutional factor analysis and deep learning. *ICML*, 2011.

[2] Thomas L. Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. *NIPS*, 2005.

[3] Thomas L Griffiths and Zoubin Ghahramani. The indian buffet process: An introduction and review. *JMLR*, 2011.


[2] http://www.vision.caltech.edu/ImageDatasets/Caltech101/

[4] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 2006.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[6] John Paisley and Lawrence Carin. Nonparametric factor analysis with beta process priors. *ICML*, 2009.

[7] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. *AISTATS*, 2009.

[8] Romain Thibaux and Michael I. Jordan. Hierarchical beta processes and the indian buffet process. *AISTATS*, 2007.

[9] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Robert Fergus. Deconvolutional networks. *CVPR*, 2010.