ONLINE BAYESIAN DICTIONARY LEARNING FOR LARGE DATASETS

Lingbo Li, Jorge Silva, Mingyuan Zhou and Lawrence Carin

Department of Electrical and Computer Engineering, Duke University, Durham NC 27708, USA

ABSTRACT

The problem of learning a data-adaptive dictionary for a very large collection of signals is addressed. This paper proposes a statistical model and associated variational Bayesian (VB) inference for simultaneously learning the dictionary and performing sparse coding of the signals. The model builds upon beta process factor analysis (BPFA), with the number of factors automatically inferred, and posterior distributions are estimated for both the dictionary and the signals. Crucially, an *online* learning procedure is employed, allowing scalability to very large datasets which would be beyond the capabilities of existing batch methods. State-of-the-art performance is demonstrated by experiments with large natural images containing tens of millions of pixels.

Index Terms— Dictionary learning, online learning, variational Bayes, factor analysis

1. INTRODUCTION

It has been known that learning a dictionary adapted to a specific set of signals can yield significantly better performance than using an off-the-shelf dictionary or basis, such as the discrete cosine transform (DCT) or wavelet basis [1]. Typically, a good dictionary is "similar" to the data in the sense that each signal can be well approximated by a linear combination of a few dictionary atoms. Finding such a representation, given a dictionary, is the goal of *sparse coding* [2, 3]. Dictionary learning and sparse coding are often jointly employed for many applications, such as image denoising [1] and interpolation/inpainting [4]. Numerous methods are available, for instance [1, 5, 4, 6, 7]. However, the majority of these methods are based on *batch* learning, *i.e.*, they require the entire data to be loaded into memory, and hence do not scale well to very large datasets of, say, millions of signals. Such datasets require *online* learning, in which portions the data are processed sequentially.

Recent work [8] has shown the ability of online methods to successfully perform dictionary learning and sparse coding for very large images. However, [8] is an optimization-based approach which does not learn a statistical model for the signals. In this paper, we propose a statistical model based on beta process factor analysis (BPFA) and associated online variational Bayesian (VB) [9] inference algorithm, which yields posterior distributions rather than point estimates for the dictionary and signals. The BPFA is related to the beta process (BP) [10, 11, 6, 7], which is a nonparametric Bayesian model allowing automatic determination of the number of useful dictionary atoms.

Similar to [12], who learn topic models for very large text collections with online VB inference, we sequentially update the posterior parameters by natural gradient descent. Under appropriate choices for the learning rate, this approach is guaranteed to converge [13]. Our online BPFA algorithm is able to efficiently compute posterior distributions for datasets of tens of millions of signals, which would be infeasible using batch inference methods, *e.g.*, Gibbs sampling as in [6, 7]. This is demonstrated by inpainting (*i.e.*, filling missing pixels in) natural images with 12 Mpixels, similarly to [8].

The remainder of the paper is organized as follows. Section 2 briefly reviews the BPFA model applied to images, Section 3 describes our proposed online VB inference algorithm, and Section 4 contains experimental results. The paper is concluded in Section 5 with a discussion of the results and final comments.

2. BPFA FOR DICTIONARY LEARNING IN IMAGES

The problem of dictionary learning for the dataset $\mathbf{X} = \{\boldsymbol{x}_i\}_{i=1,\dots,N}$ can be cast as factor analysis with $\boldsymbol{x}_i = \mathbf{D}\boldsymbol{w}_i + \boldsymbol{\epsilon}_i$, where $\mathbf{D} \in \mathbb{R}^{P\times K}$ is a dictionary with K atoms (factor loadings in a factor-analysis formulation), $\boldsymbol{w}_i \in \mathbb{R}^K$ are vectors of coefficients (factor scores) and $\boldsymbol{\epsilon}_i$ is a residual term that encompasses noise and deviation from the linear factor model; this deviation is assumed small. As an example, we may process images in blocks (patches) of size $B \times B$, represented as vectors $\boldsymbol{x}_i \in \mathbb{R}^P$, where $P = B^2$ is the number of pixels in each patch, and $i = 1, \dots, N$ with N equal to the number of patches.

Following the original BPFA model [6, 7], it is assumed that the vectors $\boldsymbol{w}_i$ are sparse. This is enforced by placing a beta-Bernoulli prior on $\boldsymbol{w}_i$. Specifically, define variables $\boldsymbol{z}_i \sim \prod_{k=1}^K \text{Bernoulli}(\pi_k)$ and $\boldsymbol{\pi} \sim \prod_{k=1}^K \text{Beta}(a_0/K, b_0(K-1)/K)$ where π_k is the k-th component of $\boldsymbol{\pi}$. It can be shown that this construction favors sparse binary vectors $\boldsymbol{z}_i$, as K grows large. The role of $\boldsymbol{z}_i$ is to select a subset of the columns of $\mathbf{D}$ for representing $\boldsymbol{x}_i$. The full model is

$$
\begin{aligned}
&\boldsymbol{x}_i = \mathbf{D}\boldsymbol{w}_i + \boldsymbol{\epsilon}_i && \boldsymbol{w}_i = \boldsymbol{z}_i \odot \boldsymbol{s}_i \\
&\boldsymbol{d}_k \sim \mathcal{N}(0, P^{-1}\mathbf{I}_P) && \boldsymbol{s}_i \sim \mathcal{N}(0, \gamma_s^{-1}\mathbf{I}_K) \\
&\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \gamma_\epsilon^{-1}\mathbf{I}_P) && \pi_k \sim \text{Beta}(\frac{a_0}{K}, \frac{b_0(K-1)}{K}) \\
&\gamma_s \sim \text{Gamma}(c_0, d_0) && \gamma_\epsilon \sim \text{Gamma}(e_0, f_0)
\end{aligned}
\tag{1}
$$

where $\boldsymbol{d}_k$ represents the kth column (atom) of $\mathbf{D}$, $\odot$ represents the elementwise or Hadamard vector product, $\mathbf{I}_P$ ($\mathbf{I}_K$) represents a $P \times P$ ($K \times K$) identity matrix, and $\{\boldsymbol{z}_i\}_{i=1,N}$ are drawn as described above. The priors are independent for all i and k, and each $\boldsymbol{s}_i$ has its own precision γ_s. The constants $\{a_0, b_0, c_0, d_0, e_0, f_0\} = \boldsymbol{\Gamma}$ are hyperparameters, which we collect in vector $\boldsymbol{\Gamma}$. The construction in (1), with the beta-Bernoulli prior for $\{\boldsymbol{z}_i\}_{i=1,\dots,N}$, is henceforth referred to as the beta process factor analysis (BPFA) model. Inference can be performed using variational Bayes (VB) or Markov chain Monte Carlo methods such as Gibbs sampling. Typically, batch implementations (as in [7]) are used; however, as discussed above, batch algorithms do not scale to very large datasets. Below, we describe an online VB method which obviates this problem.

3. ONLINE VARIATIONAL BAYES ALGORITHM

Variational Bayes (VB) inference [9, 11] for the aforementioned BPFA model, given training data $\mathbf{X}$ and hyperparameters $\mathbf{\Gamma}$, is based on approximating the true posterior $p(\mathbf{D}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{\pi}, \gamma_s, \gamma_\epsilon | \mathbf{X}, \mathbf{\Gamma})$ by members of a tractable class of distributions. In our case, this is chosen to be the class of fully factorized distributions of the form $q(\mathbf{D}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{\pi}, \gamma_s, \gamma_\epsilon) = q(\mathbf{D})q(\boldsymbol{z})q(\boldsymbol{s})q(\boldsymbol{\pi})q(\gamma_s)q(\gamma_\epsilon)$. Denoting the latent variables we wish to infer as $\mathbf{H} = \{\mathbf{D}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{\pi}, \gamma_s, \gamma_\epsilon\}$, VB inference maximizes the Evidence Lower BOund (ELBO):

$$\begin{aligned}\log p(\mathbf{X}|\mathbf{\Gamma}) \geq \mathcal{L}(\mathbf{X}, \mathbf{H}) =& \mathbb{E}_q[\log p(\boldsymbol{x}, \mathbf{D}, \boldsymbol{s}, \boldsymbol{z}, \boldsymbol{\pi}, \gamma_s, \gamma_\epsilon)] \\ &- \mathbb{E}_q[\log q(\mathbf{D}, \boldsymbol{s}, \boldsymbol{z}, \boldsymbol{\pi}, \gamma_s, \gamma_\epsilon)]\end{aligned} \tag{2}$$

with respect to $q(\mathbf{H})$. Maximizing the ELBO is equivalent to minimizing the Kullback-Leibler (KL) divergence between $q(\mathbf{H})$ and the true posterior $p(\mathbf{H}|\mathbf{X}, \mathbf{\Gamma})$.

Before further specifying how to optimize $q(\mathbf{H})$, we describe how to adapt the standard VB formulation to the online setting. We carry out online learning by stochastic coordinate ascent, in a manner analogous to [12]. The dataset $\mathbf{X}$ is randomly partitioned into T subsets (called *minibatches*) of vectors with N_t vectors per minibatch t. We use constant N_t for all t. In the limit, we can process one patch at a time by setting $T = N$ and $N_t = 1$, or revert to a batch algorithm by setting $T = 1$ and $N_t = N$. The advantages of this procedure are two-fold: (i) it is only necessary to store N_t vectors at a time in memory, independently of N; (ii) by processing the data in a random sequence, we gain robustness (albeit not immunity) to local optima and maintain convergence guarantees [13]. We now write $q(\mathbf{H})$ taking into account the minibatch index t:

$$\begin{aligned} q(\boldsymbol{d}_k) &= \mathcal{N}(\boldsymbol{\mu}_k, \mathbf{\Sigma}_k) & q(s_{tik}) &= \mathcal{N}(\nu_{tik}, \Omega_{tik}) \\ q(z_{tik}) &= \text{Bernoulli}(\eta_{tik}) & q(\pi_k) &= \text{Beta}(\tau_{1k}, \tau_{2k}) \\ q(\gamma_s) &= \text{Gamma}(c', d') & q(\gamma_\epsilon) &= \text{Gamma}(e', f'). \end{aligned} \tag{3}$$

In the above distributions, $s_{tik}, \nu_{tik}, \Omega_{tik}, z_{tik}$ and η_{tik} all refer to the i–th patch in minibatch t and to the k-th dictionary atom. The next step is to derive estimates for the posterior parameters $\boldsymbol{\mu}_k, \mathbf{\Sigma}_k, \nu_{tik}, \Omega_{tik}, \eta_{tik}, \tau_{1k}, \tau_{2k}, c', d', e', f'$. The bound (2) expands to

$$\begin{aligned}\mathcal{L}(\mathbf{X}, \mathbf{H}) =& \sum_t \sum_i \{\mathbb{E}_q[\log p(\boldsymbol{x}_{ti}|\mathbf{D}, \boldsymbol{z}_{ti}, \boldsymbol{s}_{ti}, \gamma_\epsilon)] \\ &+ \sum_{k=1}^{K} \{\mathbb{E}_q[\log p(s_{tik}|\gamma_s)] \\ &- \mathbb{E}_q[\log q(s_{tik})] + \mathbb{E}_q[\log p(z_{tik}|\pi_k)] - \mathbb{E}_q[\log q(z_{tik})]\} \\ &+ \sum_k \{\mathbb{E}_q[\log p(\boldsymbol{d}_k|0, \frac{1}{P}\mathbf{I}_P)] - \mathbb{E}_q[\log q(\boldsymbol{d}_k)] \\ &+ \mathbb{E}_q[\log p(\pi_k|a_0, b_0)] - \mathbb{E}_q[\log q(\pi_k)]\}/T \\ &+ \big(\mathbb{E}_q[\log p(\gamma_s|c_0, d_0)] - \mathbb{E}_q[\log q(\gamma_s)] \\ &+ \mathbb{E}_q[\log p(\gamma_\epsilon|e_0, f_0)] - \mathbb{E}_q[\log q(\gamma_\epsilon)]\big)/T\} \\ =& \sum_t \ell(\boldsymbol{x}_t, \boldsymbol{s}_t, \boldsymbol{z}_t, \mathbf{D}, \boldsymbol{\pi}, \gamma_s, \gamma_\epsilon)\end{aligned} \tag{4}$$

where $\ell(\boldsymbol{x}_t, \boldsymbol{s}_t, \boldsymbol{z}_t, \mathbf{D}, \boldsymbol{\pi}, \gamma_s, \gamma_\epsilon)$ denotes the contribution of minibatch t to the ELBO. As in [12], the ELBO is written as a sum over t, thereby enabling an online inference method.

The expectations involved are taken under the variational distribution q:

$$\mathbb{E}_q[\log \gamma_s] = \psi(c') - \log d', \qquad \mathbb{E}_q[\log \gamma_\epsilon] = \psi(e') - \log f' \tag{5}$$

$$\mathbb{E}_q[\ln(\pi_k)] = \psi(\frac{a_0}{K} + \frac{N}{N_t}\langle n_{tk}\rangle) - \psi(\frac{a_0 + b_0(K-1)}{K} + N)$$

$$\begin{aligned}\mathbb{E}_q[\ln(1-\pi_k)] =& \psi(\frac{b_0(K-1)}{K} + N - \frac{N}{N_t}\langle n_{tk}\rangle) \\ &- \psi(\frac{a_0 + b_0(K-1)}{K} + N)\end{aligned}$$

$$\mathbb{E}_q[s_{tik}^2] = \nu_{tik}^2 + \Omega_{tik}, \qquad \mathbb{E}_q[\boldsymbol{d}_k^T \boldsymbol{d}_k] = \boldsymbol{\mu}_k^T \boldsymbol{\mu}_k + \mathbf{\Sigma}_k$$

$$\begin{aligned}&\mathbb{E}_q[(\boldsymbol{s}_{ti} \odot \boldsymbol{z}_{ti})\mathbf{D}^T\mathbf{D}(\boldsymbol{s}_{ti} \odot \boldsymbol{z}_{ti})^T] = \\ &\sum_{k=1}^{K}\sum_{k\neq k'} \nu_{tik}\eta_{tik}\nu_{tik'}\eta_{tik'}\boldsymbol{\mu}_k^T\boldsymbol{\mu}_{k'} + \sum_{k=1}^{K}\eta_{tik}\mathbb{E}_q[s_{tik}^2]\mathbb{E}_q[\boldsymbol{d}_k^T\boldsymbol{d}_{k'}]\end{aligned}$$

where $\langle n_{tk}\rangle = \sum_{i=1}^{N_t}\eta_{tik}$ and $\psi(.)$ represents the digamma function. Maximizing $\ell(\cdot)$ for minibatch t, with respect to the parameters of q, leads to the following estimates.

3.1. Per-minibatch parameter estimates

For $q(\boldsymbol{d}_k)$,

$$\begin{aligned}\mathbf{\Sigma}_k &= (P\mathbf{I}_P + \gamma_\epsilon \frac{N}{N_t}\sum_{i=1}^{N_t}\mathbb{E}_q[s_{tik}^2]\eta_{tik})^{-1} \\ \boldsymbol{\mu}_k &= \gamma_\epsilon \mathbf{\Sigma}_k \frac{N}{N_t}\sum_{i=1}^{N_t}\nu_{tik}\eta_{tik}\mathbb{E}_q[X_{ti}^{-k}],\end{aligned} \tag{6}$$

where X_{ti}^{-k} is the reconstruction of the i-th vector in minibatch t using all but the k–th atom. For $q(s_{itk})$,

$$\begin{aligned}\Omega_{tik} &= (\gamma_s + \gamma_\epsilon \mathbb{E}_q[\boldsymbol{d}_k^T\boldsymbol{d}_k]\eta_{tik})^{-1} \\ \nu_{tik} &= \gamma_\epsilon \Omega_{tik}\boldsymbol{\mu}_k^T\eta_{tik}\mathbb{E}_q[X_{ti}^{-k}].\end{aligned} \tag{7}$$

The estimate for $q(z_{tik})$ is

$$\begin{aligned}q(z_{tik}=1) &\propto \exp\big[\mathbb{E}_q[\log \pi_k]\big) \times \\ &\exp\big(-\frac{\gamma_\epsilon}{2}\{\mathbb{E}_q[s_{tik}^2]\mathbb{E}_q[\boldsymbol{d}_k^T\boldsymbol{d}_k] - 2\boldsymbol{\mu}_k^T\nu_{tik}\mathbb{E}_q[X_{ti}^{-k}]\}\big) \\ q(z_{tik}=0) &\propto \exp\big(\mathbb{E}_q[1-\log \pi_k]\big),\end{aligned}$$

and evaluating the expected value yields

$$\mathbb{E}_q[z_{tik}] = \eta_{tik} = \frac{q(z_{tik}=1)}{q(z_{tik}=1) + q(z_{tik}=0)}. \tag{8}$$

We let $z_{tik} = 1$ if $\eta_{tik} \geq 0.5$ and $z_{tik} = 0$ otherwise. For $q(\boldsymbol{\pi})$ the estimates are

$$\tau_{1k} = \frac{a_0}{K} + \frac{N}{N_t}\langle n_{tk}\rangle, \qquad \tau_{2k} = \frac{b_0(K-1)}{K} + N - \frac{N}{N_t}\langle n_{tk}\rangle, \tag{9}$$

and for $q(\gamma_s)$ and $q(\gamma_\epsilon)$ we have

$$c' = c_0 + \frac{NK}{2} \quad d' = d_0 + \frac{N}{2N_t}\sum_{i=1}^{N_t}\boldsymbol{\nu}_{ti}^T\boldsymbol{\nu}_{ti} \quad e' = e_0 + \frac{NP}{2}$$

$$\begin{aligned}f' = f_0 + \frac{N}{2N_t}\sum_{i=1}^{N_t}\{&\boldsymbol{x}_{ti}\boldsymbol{x}_{ti}^T - 2\sum_{k=1}^{K}\eta_{tik}\nu_{tik}\boldsymbol{\mu}_k^T\boldsymbol{x}_{ti} \\ &+ \mathbb{E}_q[(\boldsymbol{s}_{ti} \odot \boldsymbol{z}_{ti})\mathbf{D}^T\mathbf{D}(\boldsymbol{s}_{ti} \odot \boldsymbol{z}_{ti})^T]\}\end{aligned} \tag{10}$$

3.2. Global parameter updates

Let $\mathbf{\Lambda} = \{\boldsymbol{\mu}_k, \mathbf{\Sigma}_k, \tau_{1k}, \tau_{2k}, c', d', e', f'\}$ be a vector containing the parameters of q that are not specific to the minibatch and data indices t and i. Define also $\tilde{\mathbf{\Lambda}}$ as the estimates of $\mathbf{\Lambda}$ based solely on the most recent minibatch, as computed in Section 3.1. We call $\mathbf{\Lambda}$ the *global* parameters. These are sequentially updated by computing a weighted average between their previous value and $\tilde{\mathbf{\Lambda}}$:

$$\mathbf{\Lambda} \leftarrow (1 - \rho_t)\mathbf{\Lambda} + \rho_t \tilde{\mathbf{\Lambda}}. \tag{11}$$

where ρ_t is the weight given to each new minibatch. Following [12], in the above estimates the contribution of each datum i is weighted by N/N_t, so as to make $\tilde{\mathbf{\Lambda}}$ equal to the batch parameter estimate we would obtain if the latest minibatch constituted the entirety of the dataset. The weight ρ_t, also called the *learning step*, is chosen according to the schedule $\rho_t = (\tau_0 + t)^{-\kappa}$, where $\kappa \in (0.5, 1]$ controls the rate of decay of the contribution from old minibatches and $\tau_0 \geq 0$ serves to slow down the initial iterations. It is shown in [12, 13] that this procedure converges and is equivalent to stochastic *natural gradient* ascent. The natural gradient is obtained by multiplying the standard gradient of the objective function by an appropriate metric matrix (in our case, the inverse Fisher information matrix for q). The overall method is summarized in Algorithm 1.

Algorithm 1 Online variational Bayes for Dictionary Learning

Define $\rho_t = (\tau_0 + t)^{-\kappa}$
Initialize $\mathbf{\Lambda}^{(1)}$ by MCMC on a small random subset of $\mathbf{X}$
for $t = 2$ to ∞ **do**
  while Stopping criterion is not met **do**
    for $k = 1$ to K **do**
      Estimate $\boldsymbol{\mu}_k$ and $\mathbf{\Sigma}_k$
      For $i = 1, \ldots, N_t$: Estimate η_{tik}, Ω_{tik} and ν_{tik}
      Estimate τ_{1k} and τ_{2k}
    end for
    Update c', d', e', f'
  end while
  Compute $\tilde{\mathbf{\Lambda}}$
  $\mathbf{\Lambda}^{(t)} = (1 - \rho_t)\mathbf{\Lambda}^{(t-1)} + \rho_t \tilde{\mathbf{\Lambda}}$
end for

4. EXPERIMENTS

We demonstrate the performance of our algorithm in two image inpainting tasks, where we do not observe all the pixels in each patch. Hence, for patch i we do not directly observe $\boldsymbol{x}_i$, but rather random projections $\boldsymbol{y}_i = \mathbf{A}_i \boldsymbol{x}_i$, where $\mathbf{A}_i \in \mathbb{R}^{m_i \times P}$ is a concatenation of random rows of the $P \times P$ identity matrix $\mathbf{I}_P$ and we observe m_i pixels. The BPFA statistical model is well suited to this missing data problem [7]. For all experiments, we used $K = 256$ atoms and set the hyperparameters to $a_0 = b_0 = 1, c_0 = d_0 = e_0 = f_0 = 10^{-6}$, which are standard "flat" values. The online learning parameters were set to $\tau_0 = 1000$ and $\kappa = 0.5$, and we divided the dataset in $T = 10,000$ minibatches of size $N_t \approx 1200$, which we found to be a good compromise between computational efficiency and convergence stability. For the first minibatch only, we initialize the model using the Gibbs sampling algorithm in [7].

The first experiment consists of taking the well-known "castle" RGB image of size 321×481, removing 50% of the pixels uniformly at random and then reconstructing. The result is shown in Figure 1, and we obtain a peak signal-to-noise ratio (PSNR) very similar to that of the batch Gibbs sampling algorithm from [7] on the same image.

Table 1. Peak signal-to-noise (PSNR) for the reconstruction with varying percentage of missing-at-random pixels and for text-damaged image

	90%	80%	70%	60%	50%	text
PSNR(dB)	34.53	40.24	43.31	45.24	46.72	45.73

The second experiment is a difficult inpainting task, using the same RGB "bird" image of size 3000×4000 (12 Mpixel) as in [8]. Due to the use of overlapping patches, N is slightly under 12 million. We restore the image from two types of damage: superimposed text, and a varying percentage of missing-at-random pixels (which are set to zero). The patches are of size $8 \times 8 \times 3$. Unlike [8], we simultaneously learn a dictionary and reconstruct the image. A non-parallelized MATLAB implementation of our Algorithm 1 takes approximately 18 hours for one full pass through the dataset (one epoch). The PSNR results can be seen in Table 1, and we illustrate the reconstruction in Figure 2, for the cases of superimposed text and 90% missing pixels. For the latter, we also show the reconstruction using nearest-neighbor (NN) interpolation with neighborhood size five. Our approach achieves better PSNR (34.53 dB vs. 16.65 dB) and avoids the multiple artifacts present in the kNN version. Note that, while [8] do not report PSNR, their reconstruction from superimposed text is visually indistinguishable from ours. The posterior mean of the learned dictionary is shown in Figure 3, and only 41 out of 256 atoms are used among all patches; individual patches use significantly less than 41 atoms. We also obtain the posterior variance, which is shown in Figure 2. Note that the pixels situated in edges and high-complexity textures have higher variance than those located in smooth regions.

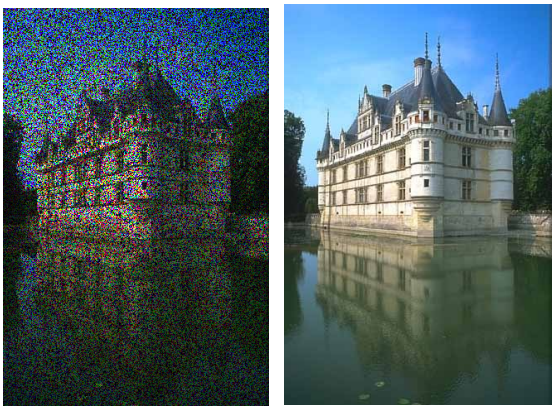

Fig. 1. Inpainting example on a medium-sized image with 50% missing pixels (left). The reconstructed image (right) has PSNR=36.53 dB, virtually identical to that of a batch implementation using Gibbs sampling [7] (PSNR=36.45).

5. CONCLUSION

We have presented an online variational Bayes algorithm for analysis of very large datasets, applied to image inpainting. The algorithm infers a BPFA statistical model, enjoys converge guarantees and can be interpreted as natural gradient optimization. State-of-the-art performance is achieved in inpainting problems with very large natural images. This is one of very few methods capable of simultaneously

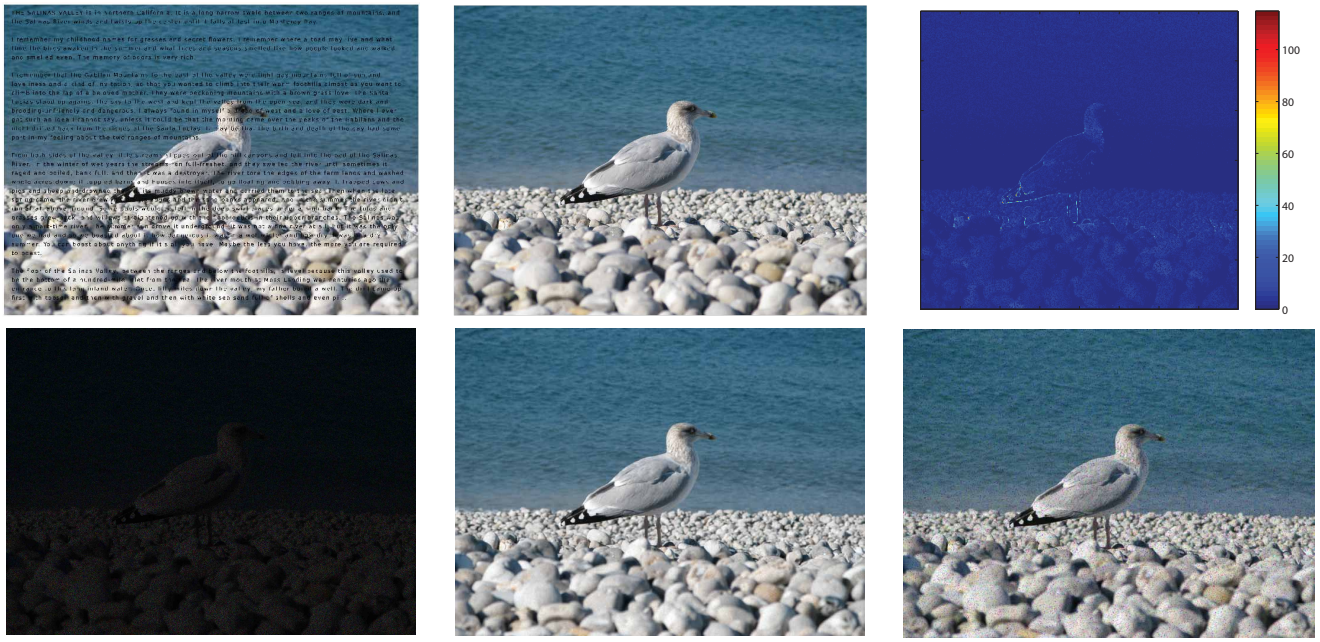

Fig. 2. Inpainting example on a 12-Mpixel image. Top left: Image damaged by text; Top middle: reconstruction (PSNR=45.73 dB); Top right: variance. Bottom left: Image with 90% missing pixels (shown as black); Bottom middle: reconstruction (PSNR=34.53 dB); Bottom right: reconstruction using NN interpolation with neighborhood size five (PSNR=16.65 dB, multiple artifacts). Best viewed in color with electronic zooming.

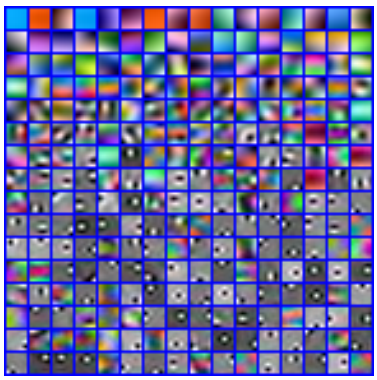

Fig. 3. Posterior mean of the learned dictionary for the 12 Mpixel "bird" image. There are 256 atoms, but only 41 are used.

performing dictionary learning and sparse coding for images of tens of millions of pixels while also inferring the number of factors. Additionally, and unlike other methods, (approximate) full posterior distributions are learned rather than point estimates. This enables further analysis and allows other problems to be considered, such as topic modeling and active learning.

6. REFERENCES

[1] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE TIP*, 2006.

[2] B.A. Olshausen and D.J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?," *Vision Research*, 1997.

[3] S.S. Chen, D.L. Donoho, and M.A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, 1999.

[4] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEE TIP*, 2008.

[5] H. Lee, A. Battle, R. Raina, and A.Y. Ng, "Efficient sparse coding algorithms," *NIPS*, 2007.

[6] M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin, "Non-parametric Bayesian dictionary learning for sparse image representations," in *NIPS*, 2009.

[7] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin, "Nonparametric Bayesian dictionary learning for analysis of noisy and incomplete images," *IEEE TIP*, 2010.

[8] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *JMLR*, 2010.

[9] H. Attias, "A variational Bayesian framework for graphical models," *NIPS*, 2000.

[10] R. Thibaux and M.I. Jordan, "Hierarchical beta processes and the Indian buffet process," in *AISTATS*, 2007.

[11] J. Paisley and L. Carin, "Nonparametric factor analysis with beta process priors," in *ICML*, 2009.

[12] M.D. Hoffman, D.M. Blei, and F. Bach, "Online learning for latent Dirichlet allocation," *NIPS*, 2010.

[13] M.A. Sato, "Online model selection based on the variational Bayes," *Neural Computation*, 2001.