SCALABLE BAYESIAN NONPARAMETRIC DICTIONARY LEARNING

Sarper Sertoglu

Department of Computer Science
Columbia University

John Paisley

Department of Electrical Engineering
Columbia University

ABSTRACT

We derive a stochastic EM algorithm for scalable dictionary learning with the beta-Bernoulli process, a Bayesian nonparametric prior that learns the dictionary size in addition to the sparse coding of each signal. The core EM algorithm provides a new way for doing inference in nonparametric dictionary learning models and has a close similarity to other sparse coding methods such as K-SVD. Our stochastic extension for handling large data sets is closely related to stochastic variational inference, with the stochastic update for one parameter exactly that found using SVI. We show our algorithm compares well with K-SVD and total variation minimization on a denoising problem using several images.

1. INTRODUCTION

Sparse coding models decompose a signal into a linear combination of a small subset of signal patterns selected from a larger dictionary of patterns. The goal of dictionary learning is to (1) learn the patterns of this dictionary, and (2) learn the sparse representation of each signal. Since both objectives are performed on data, the learning task is set up such that both goals are satisfied [1]. Bayesian nonparametric dictionary learning adds a third goal: (3) learn the size of the dictionary in conjunction with the first two goals. This can be achieved using the beta-Bernoulli process [2].

The beta-Bernoulli process gives a nonparametric prior distribution for dictionary learning. It first uses a beta process to define a measure on an infinite set of randomly generated dictionary elements. The measure for each element is a "coin-flip" probability, with the beta process ensuring that the sum of these infinite number of biases is finite. The Bernoulli process corresponds to the infinite process of coin-flips, which decides which elements are used for a patch – i.e., performs the sparse coding. The Bernoulli process has the property that the total number of unique dictionary elements used by all patches (from among the infinite set) is finite and growing proportional to the log of the number of patches.

The nonparametric beta-Bernoulli sparse coding model has proven effective in tasks such as denoising, inpainting [3] and compressed sensing for MRI [4]. Inference for the beta-Bernoulli process has focused on a variational approach [2] and MCMC sampling [3, 4]. Scalability was not considered in both cases. We develop a new EM-based algorithm for beta process factor analysis (BPFA) [2] that we then scale to large data sets along the lines of other scalable dictionary learning extensions [5]. We will show that our new EM-based algorithm has features very similar to the OMP step used by K-SVD [1], while our scalable extension can be viewed as an EM special case of stochastic variational inference [6]. The result is a more efficient sparse coding dictionary learning model based on Bayesian nonparametrics.

In the remainder of the paper, we first introduce the BPFA model in Section 2. We present a new EM algorithm in Section 3, which includes a "selective E-step" to perform sparse coding, giving an algorithm similar to OMP. In Section 4 we show experimental results on a denoising problem using several standard grayscale images, comparing with K-SVD and total variation denoising.

2. BETA PROCESS FACTOR ANALYSIS

Beta process factor analysis (BPFA) is a dictionary learning model for vectors $x \in \mathbb{R}^d$ that: (1) shrinks the dictionary to a number considered "suitable" according to the posterior distribution, and (2) performs sparse coding using the remaining dictionary elements [2]. BPFA is defined using an approximation to the beta process. For a fixed, large integer K, it first generates global variables

$$w_k \sim N(0, \eta^{-1} I), \quad \pi_k \sim \text{Beta}\left(\alpha \tfrac{\gamma}{K}, \alpha(1 - \tfrac{\gamma}{K})\right) \qquad (1)$$

for $k = 1, \ldots, K$. In the limit $K \to \infty$, the random measure $H_K = \sum_{k=1}^{K} \pi_k \delta_{w_k}$ constructed from these random variables converges to a beta process [7]. The larger the value of K, the more accurate the approximation; in practice, "large" values can be on the order of hundreds.

Sparse coding and weighting of a patch then follows

$$c_n \sim N(0, \lambda^{-1} I), \quad z_{nk} \sim \text{Bern}(\pi_k). \qquad (2)$$

Defining $Z_n = \text{diag}(z_{n1}, \ldots, z_{nK})$ and $W = [w_1, \ldots, w_K]$. the nth observation, x_n, is then

$$x_n \sim N(W Z_n c_n, \sigma^2 I), \qquad (3)$$

Nonparametric dictionary learning is enforced by the beta prior on each π_k. The prior on π_k encourages $z_{nk} = 0$ for

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

each n over many values of k. Sparse coding results from the values of k for which π_k is substantial, but still switches on and off factors. Using a Poisson process analysis of the beta process (in which $K \to \infty$), it can be shown that

$$\#\{k : \textstyle\sum_{n=1}^{N} z_{nk} > 0\} \sim \text{Pois}(\sum_{n=1}^{N} \frac{\alpha\gamma}{\alpha+n-1}), \quad (4)$$

while marginally, $\sum_k z_{nk} \sim \text{Pois}(\gamma)$ for each n. The parameters $\alpha, \gamma > 0$ can be used to control these values approximately for the finite prior above when $\alpha\gamma \ll K$ and $\gamma \ll K$.

3. EM FOR BPFA

In this section, we derive a new MAP-EM algorithm for BPFA. This is in contrast to the fully variational algorithm of [2], and fixes sensitivity issues of that algorithm which make it dependent on a good initialization. The current algorithm will be shown to be similar to K-SVD, which helps frame BPFA as a Bayesian nonparametric version of it. We then present a scalable extension of the inference algorithm using stochastic optimization.

3.1. An EM algorithm

Our goal is to maximize $p(\boldsymbol{x}, \boldsymbol{z}, W)$, the marginal joint likelihood over the sparse coding vectors z_n and the dictionary W, with integration over c_n and π_k. We perform EM using the variables c_n and π_k as hidden data. The variable c is essential for a fast closed form update of W, but can be a nuisance when optimizing z. We therefore would like to select dimensions of c to integrate over when optimizing z. To this end, we present a "selective EM" algorithm similar to those found useful in other models needing selective marginalization [8].

Notation: For an index set $\mathcal{A} \subset \{1, \ldots, K\}$, we write $c_{\mathcal{A}}$ to denote the subvector of c formed by the dimensions indexed by $\mathcal{A}$ and $W_{\mathcal{A}}$ to denote submatrix of W formed by selecting the columns of W indexed by $\mathcal{A}$. We again let Z_n be the matrix formed by putting the vector z_n on the diagonal. The identity matrix I is always assumed to be the appropriate size, as defined by the other matrices in the equation.

3.1.1. Sparse coding EM steps

The sparse coding step entails updating each z_n with c_n serving as the hidden data in EM. The EM algorithm is flexible in that we can selectively integrate over the "inconvenient" subsets of dimensions of c_n when performing the E-step. This can have computational benefits as we will show, and results in an algorithm similar to OMP [1].

Given W and $q(\pi_k)$, the update of each z_n is independent. Therefore, to make the indexing more clear we focus on a particular observation x_n in this section and ignore the index n. The following sparse coding procedure is independently run for each n. For a particular x and value of W, the sparse coding EM steps update z as described in Algorithm 1.

Algorithm 1 Sparse coding EM steps

Input: All $q(\pi_k) = p(\pi_k|z_1, \ldots, z_n) = \text{Beta}(a_k, b_k)$, where

$$a_k = \alpha\tfrac{\gamma}{K} + \textstyle\sum_n z_{nk}, \;\; b_k = \alpha(1 - \tfrac{\gamma}{K}) + \sum_n (1 - z_{nk}).$$

for each patch **do steps 1–3 below**

1. Set $z = 0$ and index set $\mathcal{A} = \emptyset$.
2. For all j, set $\xi_j^+ = \ln p(x|W, z_j = 1) + \mathbb{E}_q[\ln \pi_j]$,
   $\xi_j^- = \ln p(x|W, z_j = 0) + \mathbb{E}_q[\ln(1 - \pi_j)]$.
3. **while** $\max_j \; \xi_j^+ - \xi_j^- > 0$ **iterate (a)–(d) below**
   (a) Set $j' = \arg\max_j \; \xi_j^+ - \xi_j^-$. (See Eq. 12)
   (b) Augment $\mathcal{A} \leftarrow \mathcal{A} \cup \{j'\}$. Set $z_{j'} = 1$, $\xi_{j'}^+ = -\infty$.
   (c) Set $q(c_{\mathcal{A}}) = p(c_{\mathcal{A}}|x, z, W) = N(c_{\mathcal{A}}|\mu_{\mathcal{A}}, \Sigma_{\mathcal{A}})$, where
   $$\Sigma_{\mathcal{A}} = (\lambda I + W_{\mathcal{A}}^T W_{\mathcal{A}}/\sigma^2)^{-1}, \;\; \mu_{\mathcal{A}} = \Sigma_{\mathcal{A}} W_{\mathcal{A}}^T x/\sigma^2.$$
   (d) For all $j \notin \mathcal{A}$, set
   $$\xi_j^+ = \mathbb{E}_q[\ln p(x|c_{\mathcal{A}}, W, z_j = 1)] + \mathbb{E}_q[\ln \pi_j],$$
   $$\xi_j^- = \mathbb{E}_q[\ln p(x|c_{\mathcal{A}}, W, z_j = 0)] + \mathbb{E}_q[\ln(1 - \pi_j)].$$

After sparse coding each patch, move to Section 3.1.2.

The hidden data c and π_k each involve their own conditional posterior q distributions. As Algorithm 1 shows, we start with $z = 0$ and sequentially incorporate new dimensions similar to OMP. Given an index set of active dictionary elements $\mathcal{A}$, we form the E-step by calculating

$$Q(z, z_{\mathcal{A}}) = \mathbb{E}_q[\ln p(x, z|c_{\mathcal{A}}, W, \pi_{1:K})]. \quad (5)$$

The q distribution is on $\{\pi_{1:K}, c_{\mathcal{A}}\}$. Since these variables are independent given W, q automatically factorizes in the same way as their conditional posterior,

$$q(\pi_{1:K}, c_{\mathcal{A}}) = q(c_{\mathcal{A}}) \textstyle\prod_k q(\pi_k), \quad (6)$$

where $q(\pi_k) = \text{Beta}(a_k, b_k)$ and $q(c_{\mathcal{A}}) = N(\mu_{\mathcal{A}}, \Sigma_{\mathcal{A}})$ as indicated in Algorithm 1.

Similar to OMP, we sequentially add dimensions of c, scoring each dimension to determine which (if any) to add. The score for dimension j is denoted $\xi_j^+ - \xi_j^-$ in Algorithm 1. This is the difference of the expected log likelihoods of an "on" or "off" setting. A value > 0 indicates that $Q(z, z_{\mathcal{A}})$, and therefore the marginal likelihood, can be increased by setting $z_j = 1$. We select the maximum over j to greedily maximize $Q(z, z_{\mathcal{A}})$, as similarly done with OMP.

The expectations over q required for these scores are:

$$\mathbb{E}_q[\ln \pi_j] = \psi(a_j) - \psi(a_j + b_j), \quad (7)$$
$$\mathbb{E}_q[\ln(1 - \pi_j)] = \psi(b_j) - \psi(a_j + b_j), \quad (8)$$

where $\psi(\cdot)$ indicates the digamma function. These are computed using the conditional posterior of π_j from the previous

iteration, and not after updating each z. The likelihoods used in these scores are

$$p(x|c_{\mathcal{A}}, W, z_j = 1) = N(x|W_{\mathcal{A}}c_{\mathcal{A}}, \sigma^2 I + \lambda^{-1} w_j w_j^T), \quad (9)$$

$$p(x|c_{\mathcal{A}}, W, z_j = 0) = N(x|W_{\mathcal{A}}c_{\mathcal{A}}, \sigma^2 I). \quad (10)$$

Using the matrix inversion equality,

$$(\sigma^2 I + \lambda^{-1} w_j w_j^T)^{-1} = \sigma^{-2} I - \frac{w_j w_j^T / \sigma^2}{\lambda\sigma^2 + w_j^T w_j}, \quad (11)$$

we can simplify $\xi_j^+ - \xi_j^-$. Using the current $q(\pi_j)$ and $q(c_{\mathcal{A}})$, define the residual as $r_{\mathcal{A}} = x - W_{\mathcal{A}}\mu_{\mathcal{A}}$. Then

$$\xi_j^+ - \xi_j^- = \frac{1}{2\sigma^2}\frac{(r_{\mathcal{A}}^T w_j)^2}{\lambda\sigma^2 + w_j^T w_j} + \frac{1}{2\sigma^2}\frac{w_j^T W_{\mathcal{A}} \Sigma_{\mathcal{A}} W_{\mathcal{A}}^T w_j}{\lambda\sigma^2 + w_j^T w_j} - \frac{1}{2}\ln\left(1 + \frac{w_j^T w_j}{\lambda\sigma^2}\right) + \psi(a_j) - \psi(b_j). \quad (12)$$

We observe that the first term is very similar to the correlation score used by OMP, which greedily selects the next vector to minimize the residual. The following two terms in Eq. (12) relate to the probabilistic structure of EM. The final difference of digamma functions corresponds to the (approximate) nonparametric beta process penalty. This value will be very negative for low-probability dictionary elements, as learned through inference. It therefore eliminates these dictionary elements from the model by shrinking scores to negative values.

3.1.2. Dictionary EM steps

With sparse coding EM, we recalculate the E-step using a new $q(c_{\mathcal{A}})$ after turning on a dictionary element. EM for the dictionary W is more straightforward in that it involves one E and one M step. (Below, recall that $Z_n = \text{diag}(\vec{z}_n)$.)

E-Step: The full conditional posterior of the entire vector c_n is $q(c_n) = \mathcal{N}(c_n|\mu_n, \Sigma_n)$, where

$$\begin{aligned} \Sigma_n^{-1} &= \lambda I + \sigma^{-2} Z_n W^T W Z_n, \\ \mu_n &= \Sigma_n (W Z_n)^T x_n / \sigma^2. \end{aligned} \quad (13)$$

We use these posteriors to calculate the expectation of the complete-data log likelihood,

$$Q(W, W_{\text{old}}) = \sum_{n=1}^{N} \mathbb{E}_q[\ln p(x_n, W|c_n, z_n)].$$

We observe that this part does not involve $q(\pi_k)$. This expectation is equal to

$$Q(W, W_{\text{old}}) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N} \|x_n - W Z_n \mu_n\|^2 - \frac{1}{2\sigma^2}\sum_{n=1}^{N} \text{trace}(\Sigma_n Z_n W^T W Z_n) - \frac{\eta}{2}\text{trace}(W^T W) + \text{const.} \quad (14)$$

M-Step: We update W by maximizing Q in Eq. (14),

$$W = \left[\sum_{n=1}^{N} x_n \mu_n^T Z_n\right]\left[\eta\sigma^2 I + \sum_{n=1}^{N} Z_n(\mu_n\mu_n^T + \Sigma_n)Z_n\right]^{-1} \quad (15)$$

We then return to sparse coding EM to update each z_n and iterate until convergence.

3.2. Scalable EM

When there are many vectors to sparsely code, this procedure can take a long time. Therefore, we extend our inference algorithm to the stochastic setting to make it scalable. Using dummy variables, stochastic optimization takes an objective of the form

$$\mathcal{L} = \ln p(\beta) + \sum_{n=1}^{N} \ln p(x_n, \phi_n|\beta) \quad (16)$$

and at step t selects a subset of (x_n, ϕ_n) indexed by C_t to create a temporary objective function,

$$\mathcal{L}_t = \ln p(\beta) + \frac{N}{|C_t|}\sum_{n\in C_t} \ln p(x_n, \phi_n|\beta). \quad (17)$$

The local variables ϕ_n are then optimized for $n \in C_t$ (ϕ corresponds to z and the parameters of $q(c)$ here), and the global variables β are updated with a stochastic gradient step taken over $\mathcal{L}_t$ (β corresponds to W and the parameters of $q(\pi)$ here). Using a new random subset C_t for each t, this method is proven to converge to a local optimum of $\mathcal{L}$ under step size conditions discussed at the end of the section [9].

3.2.1. Stochastic learning for W

The update of W in Eq. (15) can be framed as a gradient step on $Q(W, W_{\text{old}})$ of the form

$$W^{(t+1)} = W^{(t)} + \rho_t(-\nabla_W^2 Q)^{-1}\nabla_W Q \quad (18)$$

in which $\rho_t = 1$. This results from the Gaussian form of W, and so the full Newton step moves directly to the solution. Stochastic optimization for W involves a stochastic gradient step. Using the standard setup of stochastic optimization [9], and selecting the preconditioning matrix to be the negative inverse Hessian calculated only over the subset C_t, we have the convenient update

$$\begin{aligned} B_t &= \eta\sigma^2\tfrac{|C_t|}{N} I + \textstyle\sum_{n\in C_t} Z_n(\mu_n\mu_n^T + \Sigma_n)Z_n, \\ W_t' &= \left(\textstyle\sum_{n\in C_t} x_n \mu_n^T Z_n\right) B_t^{-1}, \\ W^{(t+1)} &= (1-\rho_t)W^{(t)} + \rho_t W_t'. \end{aligned} \quad (19)$$

In other words, first find the optimal update of W restricted to subset C_t (using the scaling factor $N/|C_t|$), then perform a weighted average of this update with the current values.

3.2.2. *Stochastic learning for π_k*

Stochastic inference for each π_k follows precisely SVI [6]. This results from the fact that variational inference reduces to EM when the variables integrated out of the model are conditionally independent. We thus use the natural gradient of parameters (a_k, b_k) to update $q(\pi_k) = \text{Beta}(a_k, b_k)$ exactly as derived using SVI. At step t, for $k = 1, \ldots, K$ first set

$$\begin{aligned} a'_k &= \tfrac{\alpha\gamma}{K} + \tfrac{N}{|C_t|}\textstyle\sum_{n\in C_t} z_{nk}, \\ b'_k &= \alpha(1 - \tfrac{\gamma}{K}) + \tfrac{N}{|C_t|}\textstyle\sum_{n\in C_t}(1 - z_{nk}). \end{aligned} \tag{20}$$

This focuses on sparse coding of the data in C_t. Then set

$$\begin{aligned} a_k^{(t+1)} &= (1-\rho_t)a_k^{(t)} + \rho_t a'_k\,, \\ b_k^{(t+1)} &= (1-\rho_t)b_k^{(t)} + \rho_t b'_k\,. \end{aligned} \tag{21}$$

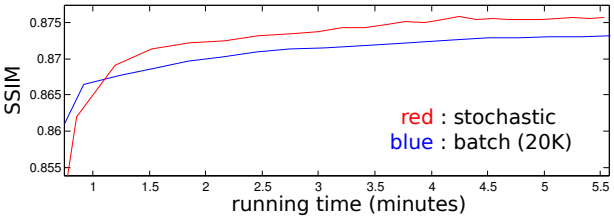
To ensure convergence, the stochastic updates to W and $q(\pi_k)$ require that $\sum_{t=1}^{\infty}\rho_t = \infty$ and $\sum_{t=1}^{\infty}\rho_t^2 < \infty$ [9]. We set $\rho_t = (t_0 + t)^{-\kappa}$, where $t_0 \geq 0$ and $\kappa \in (\frac{1}{2}, 1]$.

4. EXPERIMENTS

For our experimental evaluation we consider a denoising problem using five images: "peppers," "Lena," "house," "boat" and "Barbara." To each image we add white Gaussian noise with standard deviations $\sigma \in \{5, 10, 15, 20, 25, 50, 100\}$. For BPFA and K-SVD, we extracted 8×8 patches from each image using shifts of one pixel. We ran stochastic BPFA using the following settings: $\eta = 1/255^2$, $\lambda = 1/10$, $\alpha = 1$, $\gamma = 1$, $K = 256$, $|C_t| = 1000$, $t_0 = 10$, $\kappa = 0.75$ for a total of 300 iterations. We use the algorithm described in [11] to set the value of the noise variance σ^2.

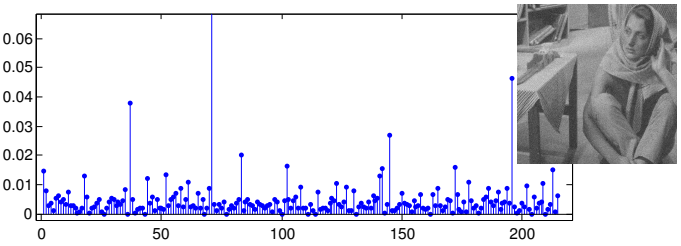
We compare our algorithm with K-SVD [1] and total variation denoising [10]. For K-SVD we use the standard settings of the code provided by [1] and use [11] to set the noise parameter ($K = 256$). For total variation, we consider both the isotropic and anisotropic versions. To set the regularization parameter, we adaptively modify its value until the resulting denoised image has empirical noise variance equal to the value found using [11]. Therefore, all algorithms are compared under the same noise assumption (which is close to the ground truth).

In Table 2 we show quantitative results using the SSIM and PSNR performance measures [12]. For the first four images, we also show these values using the noisy image as a baseline. The performance of BPFA is competitive with K-SVD, and significantly better than total variation denoising. We notice that in several cases (e.g., "Lena"), the PSNR of K-SVD is better, while the SSIM of the same reconstruction favors BPFA, suggesting that the added regularization of the BPFA prior can improve visual quality. Dictionary learning using BPFA took approximately 3.5 minutes per image (taking 300 steps with $|C_t| = 1000$), followed by an iteration over the full image for reconstruction.



(a) SSIM vs running time for "house" ($\sigma = 15$).

(b) "Barbara" denoising ($\sigma = 25$). Left to right: BPFA, K-SVD, TV (iso)



(c) BPFA dictionary probabilities for Fig. 1(b). (With noisy image inset)

Fig. 1. Some example results.

In Figure 1(a) we show an example of SSIM vs running time over 500 steps of stochastic inference. For comparison, we subsampled 20,000 patches and ran batch inference for dictionary learning on the same image. (Both functions show the SSIM after running an iteration over all patches in the image, which wasn't factored into the running time.) We see a computational advantage to stochastic inference, which can converge on a reconstruction faster than batch inference.

In Figure 1(b) and 1(c) we show an example reconstruction and distribution on the learned dictionary (the "coin-flip" biases). The model pruned 41 of the 256 dictionary elements and learned a distribution that promotes sparsity on the remaining elements. The visual results of BPFA and K-SVD are similar, but there is a noticeable visual improvement over TV denoising. In Table 1 we show the average number of dictionary elements per patch for BPFA and K-SVD for two images (discounting a shift element always used by BPFA). Sparsity is similar, which is not surprising given the similarity of OMP and our sparse coding EM step.

Table 1. Dictionary elements per patch for BPFA/K-SVD

image	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$
pepp.	6.5/6.2	3.3/2.7	1.7/1.5	1.2/1.0	0.9/0.7
boat	7.3/6.7	3.3/2.6	1.6/1.4	1.0/0.9	0.7/0.7

Table 2. SSIM | PSNR for five different grayscale images as a function of noise standard deviation.

PEPPERS	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$	$\sigma = 50$	$\sigma = 100$
BPFA	0.947 | 37.40	0.922 | 34.15	0.900 | 32.14	0.881 | 30.83	0.860 | 29.72	0.782 | 26.48	0.667 | 23.15
K-SVD	0.949 | 37.73	0.923 | 34.20	0.900 | 32.18	0.879 | 30.79	0.856 | 29.63	0.766 | 26.14	0.613 | 21.93
TV (aniso)	0.938 | 35.73	0.903 | 32.40	0.872 | 30.44	0.850 | 29.25	0.828 | 28.26	0.744 | 25.37	0.636 | 22.78
TV (iso)	0.938 | 35.85	0.905 | 32.56	0.875 | 30.59	0.853 | 29.42	0.832 | 28.40	0.751 | 25.48	0.649 | 22.89
Baseline	0.885 | 34.16	0.718 | 28.15	0.584 | 24.61	0.485 | 22.09	0.411 | 20.17	0.215 | 14.12	0.086 | 8.170
LENA	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$	$\sigma = 50$	$\sigma = 100$
BPFA	0.940 | 38.27	0.910 | 35.37	0.884 | 33.58	0.863 | 32.27	0.845 | 31.28	0.759 | 27.97	0.634 | 24.46
K-SVD	0.937 | 38.27	0.906 | 35.39	0.881 | 33.69	0.859 | 32.40	0.840 | 31.40	0.748 | 27.99	0.600 | 24.39
TV (aniso)	0.917 | 35.92	0.874 | 32.71	0.841 | 30.96	0.816 | 29.84	0.793 | 28.87	0.725 | 26.47	0.650 | 24.36
TV (iso)	0.917 | 35.95	0.874 | 32.78	0.843 | 31.04	0.818 | 29.93	0.796 | 28.98	0.731 | 26.57	0.661 | 24.50
Baseline	0.855 | 34.15	0.646 | 28.14	0.493 | 24.61	0.390 | 22.12	0.318 | 20.19	0.145 | 14.14	0.052 | 8.141
HOUSE	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$	$\sigma = 50$	$\sigma = 100$
BPFA	0.934 | 38.16	0.906 | 35.81	0.875 | 34.16	0.853 | 33.16	0.837 | 32.01	0.767 | 28.53	0.621 | 24.03
K-SVD	0.948 | 39.41	0.902 | 35.98	0.870 | 34.27	0.850 | 33.21	0.833 | 32.04	0.739 | 28.15	0.559 | 23.63
TV (aniso)	0.916 | 36.86	0.874 | 33.76	0.847 | 31.89	0.831 | 30.76	0.817 | 29.91	0.753 | 27.04	0.638 | 23.90
TV (iso)	0.915 | 36.82	0.873 | 33.72	0.846 | 31.85	0.831 | 30.73	0.819 | 29.96	0.762 | 27.12	0.654 | 24.01
Baseline	0.841 | 34.16	0.627 | 28.12	0.481 | 24.63	0.387 | 22.15	0.319 | 20.13	0.161 | 14.13	0.062 | 8.110
BARBRA	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$	$\sigma = 50$	$\sigma = 100$
BPFA	0.949 | 37.15	0.929 | 34.32	0.907 | 32.40	0.887 | 30.95	0.861 | 29.71	0.732 | 25.76	0.528 | 21.75
K-SVD	0.957 | 38.09	0.930 | 34.45	0.908 | 32.47	0.882 | 30.94	0.855 | 29.70	0.722 | 25.68	0.524 | 21.88
TV (aniso)	0.936 | 34.17	0.877 | 29.77	0.820 | 27.49	0.770 | 26.00	0.728 | 25.07	0.609 | 22.96	0.525 | 21.62
TV (iso)	0.936 | 34.18	0.877 | 29.77	0.822 | 27.50	0.773 | 26.01	0.734 | 25.12	0.618 | 23.02	0.535 | 21.70
Baseline	0.894 | 34.14	0.739 | 28.13	0.614 | 24.63	0.518 | 22.13	0.444 | 20.18	0.227 | 14.15	0.086 | 8.142
BOAT	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$	$\sigma = 50$	$\sigma = 100$
BPFA	0.934 | 36.76	0.887 | 33.54	0.850 | 31.71	0.818 | 30.39	0.787 | 29.36	0.670 | 26.13	0.519 | 22.88
K-SVD	0.935 | 37.11	0.881 | 33.53	0.841 | 31.68	0.805 | 30.37	0.773 | 29.31	0.658 | 26.08	0.501 | 22.83
TV (aniso)	0.910 | 35.19	0.849 | 31.64	0.802 | 29.79	0.762 | 28.50	0.730 | 27.57	0.634 | 25.17	0.525 | 22.81
TV (iso)	0.909 | 35.15	0.847 | 31.57	0.799 | 29.71	0.759 | 28.43	0.728 | 27.51	0.637 | 25.18	0.531 | 22.83

REFERENCES

[1] M. Aharon, M. Elad, A. Bruckstein and Y. Katz. K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, vol. 54, pp. 4311-4322, 2006.

[2] J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. *International Conference on Machine Learning*, 2009.

[3] 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 Trans. Image Process.*, vol. 21, no. 1, pp. 130-144, 2012.

[4] Y. Huang, J. Paisley, Q. Lin, X. Ding, X. Fu and X. Zhang. Bayesian nonparametric dictionary learning for compressed sensing MRI. *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5007-5019, 2014.

[5] J. Mairal, F. Bach, J. Ponce and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research,* vol. 11, pp. 19-60, 2010.

[6] M. Hoffman, D. Blei, C. Wang and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, vol. 14, pp. 1005-1031, 2013.

[7] J. Paisley and M. Jordan. A constructive definition of the beta process. Technical Report, 2015. (in preparation) www.columbia.edu/∼jwp2128/PaisleyJordan2015.pdf

[8] Z. Ghahramani and G. Hinton. The EM algorithm for mixtures of factor analyzers. *Technical Report CRG-TR-96-1*, 1996.

[9] L. Bottou. Online learning and stochastic approximations. In *Online Learning in Neural Networks*, CUP, Cambridge, 1998.

[10] T. Goldstein and S. Osher. The split Bregman method for L1-regularized problems. *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323-343, 2009.

[11] X. Liu, M. Tanaka and M. Okutomi. Single-image noise level estimation for blind denoising. *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5226-5237, 2013.

[12] Z. Wang, A. Bovik, H. Sheikh and E Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. on Image Processing*, vol. 13, no. 4, pp. 600-612, 2004.