Technical Report TR-2010-04

University of British Columbia

Sparsity priors and boosting for learning localized distributed feature representations

Bo Chen Kevin Swersky Ben Marlin Nando de Freitas

March 29, 2010

Department of Computer Science, University of British Columbia
2366 Main Mall, Vancouver, BC Canada V6T 1Z4

Abstract

This technical report presents a study of methods for learning sparse codes and localized features from data. In the context of this study, we propose a new prior for generating sparse image codes with low-energy, localized features. The experiments show that with this prior, it is possible to encode the model with significantly fewer bits without affecting accuracy. The report also introduces a boosting method for learning the structure and parameters of sparse coding models. The new methods are compared to several existing sparse coding techniques on two tasks: reconstruction of natural image patches and self taught learning. The experiments examine the effect of structural choices, priors and dataset size on model size and performance. Interestingly, we discover that, for sparse coding, it is possible to obtain more compact models without incurring reconstruction errors by simply increasing the dataset size.

1 Introduction

In their seminal work, Olshausen and Field (1996, 1997) proposed a method for learning sparse linear codes from natural images that automatically generated a set of localized, oriented, bandpass filters similar to the receptive fields of simple cells found in the mammalian primary visual cortex. We will refer to these learned filters as features or bases. Proponents of sparse coding argue that sparseness is a desirable property because it leads to simple, compact descriptions of the structures in natural scenes in terms of a small number of basis vectors (Garrigues and Olshausen, 2008; Karklin and Lewicki, 2009). Compact features also provide a convenient way of forming associations at later stages of processing (Olshausen, 2001). Underlying this line of work, is the ecological-statistical hypothesis that the visual system is adapted to the statistical structure of natural images, where locality and hierarchical structure play a prominent role (Hyvarinen et al., 2009).

It has been argued that the design of hierarchical representations in terms of local features is key to understanding intelligence and, in particular, to the design of parsimonious memory systems (Hawkins and George, 2006). Much progress has also been attained recently with deep-layer representations, where the features tend to be sparse (Hinton and Salakhutdinov, 2006; Hinton et al., 2006; Bengio et al., 2007; Bengio and Le Cun, 2007; Larochelle et al., 2009; Salakhutdinov and Hinton, 2009). Many of the deep-layer models consist of stacks of restricted Boltzmann machines (RBMs). These RBMs are trained in a greedy fashion layer-by-layer. At the end of the greedy training, the parameters of the entire network are refined with a back-propagation step. In much earlier work on Boltzmann machines, Kappen (1995) advocated the use of sparse representations as a way of simplifying computation in these models.

More recently, Kavukcuoglu et al. (2008a) state that learning sparse representations can be advantageous because the resulting features are more likely to be linearly separable in high dimensions and, hence, more robust to noise. Learning sparse representations from large datasets in an unsupervised manner has also been shown to be very suitable for complex multi-task learning problems (R. Raina and Ng., 2007). The enormous success of engineered local feature methods in computer vision, such as SIFT (Lowe, 2004), is also a significant motivating factor for designing algorithms that will learn these local, invariant bases automatically.

It is therefore not surprising that the problem of learning sparse codes and compact features automatically from data has taken a central position in the machine learning field (Bell and Sejnowski, 1997; Hoyer, 2004;

Kavukcuoglu et al., 2008a; Lee et al., 2008). The most popular strategy is to place an L_1 prior on the code coefficients and an L_2 prior on the bases; see among many others (Kavukcuoglu et al., 2008a; R. Raina and Ng., 2007). Garrigues and Olshausen (2008) propose a Bayesian version of this strategy, consisting of a mixture prior with a Dirac function at zero and a normal distribution. In the context of learning deep restricted Boltzmann machines (RBMs) and convolutional networks, Lee et al. (2008, 2009) introduce a simple prior that modifies the coefficients of the hidden units to attain sparser RBMs.

In this paper, we examine the mechanisms used to obtain compact feature representations. In doing so, we propose a simple prior for sparse coding and obtain very compact, localized features without affecting the reconstruction performance of the model. We observe that the prior leads to models with comparable approximating power but with less storage requirements (an important factor in embodied systems). We also propose a boosting algorithm to learn the features and structure of sparse coding models automatically. This algorithm guarantees that the right number of hidden units is chosen according to an early stopping cross-validation test. Finally, we examine the effect of other variables such as model structure and dataset size on obtaining sparse localized features with sparse coding. We find that by simply increasing the dataset size, we obtain more compact feature representations.

2 Continuous Coding Schemes

We begin with a review of several linear coding schemes. Here, the data matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$ (of size N data cases by D dimensions) is approximated by a product of two matrices $\mathbf{B}$ and $\mathbf{C}$. $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_j, \ldots, \mathbf{b}_M] \in \mathbb{R}^{D \times M}$ is a matrix of M feature vectors $\mathbf{b}_j \in \mathbb{R}^D$ and $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_n, \ldots, \mathbf{c}_N] \in \mathbb{R}^{M \times N}$ is a matrix of N code vectors $\mathbf{c}_n \in \mathbb{R}^M$. (We use subscripts and superscripts to denote columns and rows of a matrix: b_j^i denotes the element of $\mathbf{B}$ at the i-th row and j-th column. Matrices are upper-case bold and vectors are lower-case bold. Finally, $\mathbf{B}_{1:j}$ denotes the first j columns of $\mathbf{B}$.)

Existing linear coding methods differ in terms of the constraints imposed on $\mathbf{B}$ and/or $\mathbf{C}$. One of the most popular approaches is principal component analysis (PCA), which constrains the bases to be orthonormal: $\mathbf{b}_j^T \mathbf{b}_l = 0, \forall j \neq l$ and $||\mathbf{b}_j||_2^2 = 1, \forall j$. In certain application domains, it might be desirable for $\mathbf{B}$ and $\mathbf{C}$ to be non-negative. This gives rise to Non-negative Matrix Factorization (NMF) (Paatero and Tapper, 1994; Lee and Seung, 1999), where the optimal parameters $\mathbf{B}^*, \mathbf{C}^*$ are given by:

$$\begin{aligned} \mathbf{B}^*, \mathbf{C}^* &= \arg\min_{\mathbf{B},\mathbf{C}} ||\mathbf{X} - \mathbf{BC}||_2^2 \\ s.t. &\quad b_j^i \geq 0,\ c_n^j \geq 0,\ \ \forall i, j, n \end{aligned}$$

In limited cases, NMF induces sparse representations (Lee and Seung, 1999), but in general sparsity has to be enforced through explicit addition of appropriate regularizers (Hoyer, 2004; Li et al., 2001).

Independent Component Analysis (ICA) is a popular linear approach for obtaining sparse representations by maximizing statistical independence. It produces the same type of results as sparse coding, discussed in detail below, so we do not pursue it further in this paper. We refer the reader to (Hyvarinen et al., 2009) for a comprehensive treatment of ICA.

Classical sparse coding (Olshausen and Field, 1996, 1997) uses an overcomplete set of features and imposes a sparsity penalty on the activation matrix $\mathbf{C}$. Formally,

$$\begin{aligned} \mathbf{B}^*, \mathbf{C}^* &= \arg\min_{\mathbf{B},\mathbf{C}} ||\mathbf{X} - \mathbf{BC}||_2^2 + \lambda \left\|\mathbf{C}\right\|_1 \\ s.t. &\quad ||\mathbf{b}_j||_2^2 = 1,\ \forall j. \end{aligned}$$

This objective function is denoted $C_{SC}(\mathbf{B}, \mathbf{C})$. Note that the basis vectors are normalized to have unit length in order to avoid the trivial solution of shifting all the energy from $\mathbf{C}$ to $\mathbf{B}$. The goal of sparse coding is to learn both the bases $\mathbf{B}$ shared among all data instances and the codes $\mathbf{c}_n$ for each n-th data instance. Although given $\mathbf{B}$ the problem is convex in $\mathbf{C}$ and vice versa, it is not jointly convex in both $\mathbf{B}$ and $\mathbf{C}$. Hence optimization typically involves alternating between minimizing over $\mathbf{B}$ and $\mathbf{C}$; see for example (Kavukcuoglu et al., 2008a).

To understand how the L_1 prior on $\mathbf{C}$ leads to the emergence of sparse (local) features $\mathbf{b}_j$, consider the example of discriminating among images of 3's and 8's shown in Figure 1. There are two local features that are sufficient for discriminating the two classes (even under significant occlusion and noise in the rest of the image). One way these features can be learned is if the corresponding coefficients are non-zero for an 8 and precisely zero for a 3. The sparseness prior on $\mathbf{C}$ helps with this process by encouraging many of the feature coefficients to go to zero.

Once the feature set $\mathbf{B}$ is learned, inference for new data involves finding the optimal $\mathbf{C}$ given $\mathbf{B}$. This amounts to solving an L_1 regularized linear

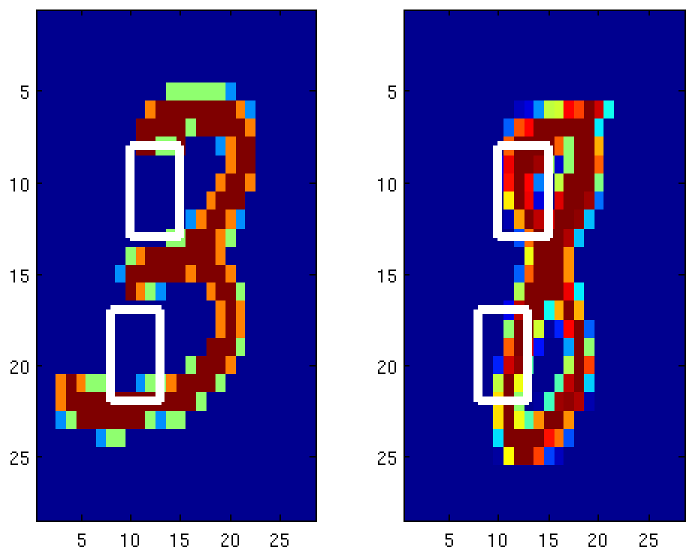

Figure 1: Local features to discriminate 3's from 8's.

regression, which can be inefficient at test time. To avoid this expensive inference procedure, (Kavukcuoglu et al., 2008b) approximate the mapping from the input to the code by a feed-forward neural network. During training, the optimization problem is:

$$\begin{aligned} \arg\min_{\mathbf{B},\mathbf{C},\mathbf{W},\mathbf{G}} & \; ||\mathbf{X}-\mathbf{BC}||_2^2 + \lambda\,\|\mathbf{C}\|_1 + \beta\,\|\mathbf{C}-\mathbf{G}\tanh(\mathbf{WX})\|_2^2 \\ s.t. & \qquad ||\mathbf{b}_j||_2^2 = 1,\ \forall j, \end{aligned}$$

where $\beta \in (0,1)$ weights the sparse coding objective $C_{SC}(\mathbf{B},\mathbf{C})$ against the sigmoidal approximation objective $C_{Approx}(\mathbf{C},\mathbf{W},\mathbf{G}) = \|\mathbf{C}-\mathbf{G}\tanh(\mathbf{WX})\|_2^2$, where the tanh neural network is parameterized by $\mathbf{W}$, an $M \times D$ weight matrix, and $\mathbf{G}$, a $D \times D$ (diagonal) gain matrix. The choice of tanh nonlinearity and gain matrix allows the input to take real values greater than 1. The resulting network can reasonably approximate the optimal $\mathbf{C}$ while expediting inference. At test time, approximate codes can be generated by a simple feed-forward computation. We refer to this as the sigmoidal (approximation) method.

2.1 Basis Interaction Priors

There has been a significant amount of work in the literature of modeling lateral inhibition between neurons in neural architectures. See for example (Garrigues and Olshausen, 2008) and (Rozell et al., 2008). Here we model this inhibition between features with the goal of obtaining sparse models with less redundant features, while not increasing the approximation error. This has the potential to result in models requiring fewer bits to describe, leading to memory savings.

Let the similarity between two feature vectors be computed via a kernel $\mathcal{K} : \mathbb{R}^D \times \mathbb{R}^D \longrightarrow \mathbb{R}$. Then we can model lateral inhibition by adding a prior that penalizes the interaction between all pairs of distinct features. We refer to this prior as a Basis Interaction Prior (BIP) and define it as follows:

$$P(\mathbf{B}) \propto \exp\left(-\sum_{j=1}^{M}\sum_{l\neq j}\mathcal{K}(\mathbf{b}_j, \mathbf{b}_l)\right).$$

From an optimization perspective, we incorporate this prior by adding the following term to the cost function of sparse coding:

$$C_{BIP}(\mathbf{B}) = \sum_{j=1}^{M}\sum_{l\neq j}\mathcal{K}(\mathbf{b}_j, \mathbf{b}_l). \tag{1}$$

Using $\mathcal{K}(B_j, B_l) = \sum_{i=1}^{D} b_{ij}b_{il}$ would not rule out the possibility of two feature vectors with complementary regions of opposite signs cancelling out in the inner product computation. Instead, we propose to compute the similarity between fully-rectified feature vectors. More formally, we define the following cost:

$$C_{BIP}(\mathbf{B}) \equiv \sum_{i=1}^{D}\sum_{j=1}^{M}\sum_{l\neq j}|b_j^i||b_l^i|.$$

This penalty penalizes features for sharing any overlapping regions. As we will see in the experiments, combining this prior with sparse coding will produce extremely localized features.

The overall cost function $C(\mathbf{B}, \mathbf{Z})$ we minimize, in what we refer to as the BIP method, is

$$C_{SC}(\mathbf{B}, \mathbf{C}) + \alpha C_{BIP}(\mathbf{B}) + \beta C_{approx}(\mathbf{C}, \mathbf{W}, \mathbf{G}), \tag{2}$$

where $\alpha \in (0,1)$ balances the strength of the BIP prior with the other terms. When α is large, overlapping features are heavily discouraged, hence the features are extremely localized. As the effect of BIP weakens and vanishes eventually, the basis vectors reduce to those of ordinary sparse coding. In section 4, we analyze in more detail the smooth transition between these two extreme cases.

BIP is closely related to a technique developed to obtain localized part-based representations called Local NMF (Li et al., 2001). Local NMF minimizes redundancy in the representation by penalizing pair-wise correlations between feature vectors. It also encourages the activation pattern of each feature across the entire data set to have large magnitude in order to only retain important feature vectors. Mathematically,

$$\begin{aligned} \min_{\mathbf{B},\mathbf{C}} \quad & D(\mathbf{X}||\mathbf{BC}) + \lambda_1 \sum_{l,j} \mathbf{b}_l^T \mathbf{b}_j - \lambda_2 \sum_{j=1}^{M} ||\mathbf{c}^j||_2 \\ s.t. \quad & b_j^i \geq 0,\ c_n^j \geq 0,\ \forall i,j,n, \end{aligned}$$

where λ_1 and λ_2 are positive constants and the divergence between the data and reconstruction is defined as: $D(\mathbf{P}||\mathbf{Q}) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}} - p_{ij} + q_{ij}$. Since $\mathbf{B}$ is required to have positive entries, the feature interaction terms of BIP and local NMF are equivalent up to a constant. However, BIP is more general in that it allows for negative feature elements and coefficients.

2.2 Boosted Sparse Coding

Inspired by the work of Buhlmann and Yu (2005), who show that boosting and variable selection with an L_1 penalty have striking similarities, we develop an L_2-boosting algorithm for sparse coding in this section. The idea is to grow the model incrementally and only add those features that are needed to fit the data. Cross-validation is used to determine the final structure of the model.

We reformulate the objective in Equation (2) in a Boosting framework:

$$C(\mathbf{B},\mathbf{C},\mathbf{W},\mathbf{G}) = ||\mathbf{X} - \sum_{j=1}^{M} \mathbf{b}_j \mathbf{c}^j)||_2^2 + \Phi(\mathbf{B},\mathbf{C}), \tag{3}$$

where $\Phi(\mathbf{B},\mathbf{C}) =$ includes the sparsity and approximation components of the objective function. In this equation, $\mathbf{BC} = \sum_{j=1}^{M} \mathbf{b}_j \mathbf{c}^j$ takes the form of an additive model that can be solved incrementally via L_2-Boosting (Friedman, 2001; Lutz et al., 2008). In the view of boosting, each $\mathbf{c}^j$ is a weak learner weighted by the coefficient vector $\mathbf{b}_j$.

Formally, assume that $j-1$ weak learners $\mathbf{C}^{1:j-1}$ and coefficients $\mathbf{B}_{1:j-1}$ have been included in the model, we can solve for the increment that maximally reduces the objective:

$$\begin{aligned}
\{\mathbf{b}_j, \mathbf{c}^j\} &= \arg \min_{\mathbf{c},\mathbf{b}:||\mathbf{b}||_2=1} ||\mathbf{X} - \mathbf{B}_{1:j-1}\mathbf{C}^{1:j-1} - \mathbf{b}\mathbf{c})||_2^2 \\
&+ \Phi(\mathbf{B}_{1:j-1}, \mathbf{C}_{1:j-1}, \mathbf{b}, \mathbf{c}) \\
&= \arg \min_{\mathbf{c},\mathbf{b}:||\mathbf{b}||_2=1} ||\mathbf{X} - \mathbf{B}_{1:j-1}\mathbf{C}^{1:j-1} - \mathbf{b}\mathbf{c}||_2^2 \\
+ \lambda||\mathbf{c}||_1 + \alpha C_{BIP}(\mathbf{B}_{1:j-1}, \mathbf{b}) &+ \beta C_{approx}(\mathbf{c}, \mathbf{W}_j, \mathbf{G}_j),
\end{aligned}$$

where the parameters $\{\mathbf{G}_j, \mathbf{W}_j\}$ are learned at the same time as $\mathbf{c}$. This formulation is valid because the L_1 norm, L_2 norm and the sigmoidal approximation are all separable. However, in our particular choice of the interaction kernel, the jth feature depends on all the other features, including those not yet added to the model. In the L_2-Boosting view, the coefficients of all the weak learners, except of those included in the model, are zero. Therefore we consider only the interaction between $\mathbf{b}$ and $\mathbf{B}_{1:j-1}$.

L_2-Boosting allows us to construct an incremental solution to the feature set, of which the optimal size is determined via cross-validation. In addition, L_2-Boosting itself has a variable selection effect: Lin et al. (July 2008) demonstrate that L_2-Boosting is an approximation to L_0 penalty on the weak learners and, as mentioned earlier, Buhlmann and Yu (2005) show that boosting and variable selection with an L_1 penalty have striking similarities. Therefore, the benefit of L_2-Boosting lies not only in automatic model selection, but also in minimizing the number of feature vectors that are needed to represent the data. A closely-related paper is (Bradley and Bagnell, 2009), where Boosting is also employed to learn sparse coding bases. The distinction is that that work relies on an additional, structural sparsity prior on the activations, and it does not seem to yield sparse basis vectors.

3 Discrete Coding Schemes

Coding with discrete variables is an alternative to the previous regularized continuous coding schemes. Binary Restricted Boltzmann machines are a popular example of this approach (Hinton and Salakhutdinov, 2006; Hinton et al., 2006). An RBM is an undirected probabilistic graphical model with symmetric weights b_j^i between visible units x^i, $i \in 1, ..., D$ and discrete hidden units $c^j \in \{0,1\}$, $j \in 1, ..., M$. (Note that most papers on RBMs use v for the visible units, h for the hidden units and W for the weights. We adopt a different notation with the intention of drawing a closer connection between these models and the continuous sparse coding methods presented in the previous section.). Henceforth, we use Gaussian RBM and RBM interchangeably.

The negative log-probability for an RBM with Gaussian-distributed visible units and binary hidden units is given below where $\mathbf{r}$ are the visible unit bias parameters and $\mathbf{s}$ are the hidden unit bias parameters:

$$\begin{aligned} -\log P(\mathbf{x}_n, \mathbf{c}_n) = \log Z(\mathbf{B}, \mathbf{r}, \mathbf{s}) + \frac{1}{2}\sum_{i=1}^{D} \frac{(x_n^i - r_i)^2}{\sigma^2} \\ - \sum_{i=1}^{D}\sum_{j=1}^{M} \frac{x_n^i b_j^i c_n^j}{\sigma^2} - \sum_{j=1}^{M} \frac{s_j c_n^j}{\sigma^2}, \end{aligned}$$

where $Z(\mathbf{B}, \mathbf{r}, \mathbf{s})$ is the partition function, which depends on the weights and bias parameters. In an RBM, the visible units are conditionally independent given the hidden units and the hidden units are conditionally independent given the visible units. We can easily sample from the conditional distributions:

$$P(\mathbf{x}|\mathbf{c}) \sim \prod_{d=1}^{D} \mathcal{N}\left(r_i + \sum_{j=1}^{M} b_j^i c^j, \sigma^2\right) \tag{4}$$

$$P(\mathbf{c}|\mathbf{x}) \sim \prod_{j=1}^{M} \mathit{logistic}\left(\frac{1}{\sigma^2}\left(s_j + \sum_{i=1}^{D} b_j^i x^i\right)\right) \tag{5}$$

and apply stochastic approximation algorithms to estimate the parameters. Lee et al. (2008) propose a method for encouraging further sparsity in the hidden unit activations by adding a regularization term to the log-likelihood of the form:

$$\lambda \sum_{j=1}^{M} \left(\frac{1}{N}\sum_{n=1}^{N} E[c^j|\mathbf{x}_n] - p\right)^2, \tag{6}$$

where $\mathbf{x}_n$ is the n^{th} data case, λ is the strength of the regularization term, and p is a small constant indicating the proportion over which the hidden units should be active. A closely related prior was proposed earlier in (Kappen, 1995).

4 Experiments

Our experiments will show that (i) BIP leads to very sparse features and gains in computational efficiency, (ii) that for the same measure of reconstruction error, it requires less bits to represent the feature matrix (it is more compact/sparse), (iii) that the proposed L_2-boosting procedure results in compact features at multiple frequencies, (iv) that the approximating sigmoidal term of Kavukcuoglu et al. (2008a), adopted to expedited inference, does not affect reconstruction performance and (v) that it is possible to obtain sparser features by simply increasing the dataset size. The experiments will also assess the generalization performance of the models with BIP and trained with L2 Boosting on a self-taught learning task.

4.1 Reconstruction Against Model Size

We investigate the relationship among reconstruction error, feature sparsity and code sparsity. For this experiment, we use the dataset of $10,000$ 14×14 natural image patches from Lee et al. (2007). To learn the sparse coding models, we adopt the projected gradient method of Schmidt et al. (2007). As in Kavukcuoglu et al. (2008a), we divide the training set into mini-batches of size 10. Learning consists of first computing the optimal code $\mathbf{C}$ for a given batch, followed by updating the basis vectors $\mathbf{B}$ with a single step of stochastic gradient descent and then a normalization step to make sure each $\mathbf{b}_j$ has unit L_2-norm. We use an adaptive learning rate of $\frac{1}{\tau+10}$ where τ denotes the epoch number (number of passes over the entire data set). We terminate training when either the reconstruction errors between adjacent epochs differs by less than 1% or a maximum of 20 epochs has been reached.

To learn the Sparse Gaussian RBM (SRBM) models, we use code provided by Lee et al. (2008). We vary the number of hidden units, training set size, and sparsity penalty, including a sparsity penalty of 0 corresponding to an ordinary Gaussian RBM. Each model is trained using Contrastive Divergence (Hinton et al., 2006) by running 500 steps of Stochastic Gradient Descent on mini-batches of 100 examples, which means 150,000 parameter updates on the 30,000 data. However, we replace the sampling of x^i with the mean field values $E[x^i|\mathbf{c}]$.

The estimated receptive fields Figure 2 indicates that BIP is able to learn extremely local features. As in sparse coding, these features seem to correspond to edges, although each filter is responsible for modeling only a small region of the input image. By varying the balancing parameter α, we can create a smooth transition from extremely localized filters to regular

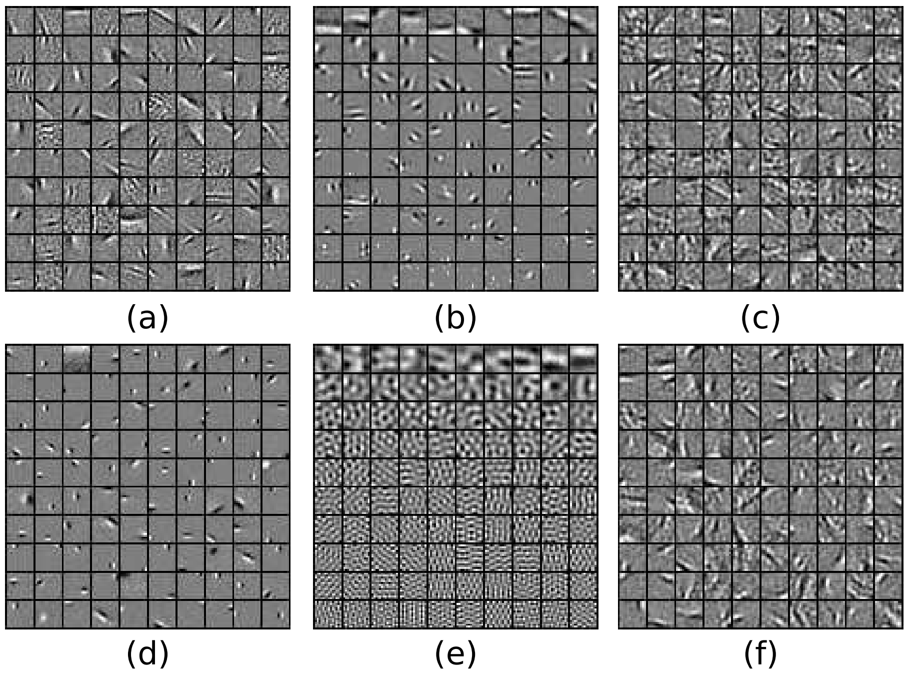

Figure 2: Learned features with (a) sparse coding, (b) L_2 boosting with BIP, (c) Gaussian RBM, (d) sparse coding with BIP, (e) PCA, and (f) Sparse Gaussian RBM. BIP features are extremely localized and selective to orientation. The features learned with PCA and boosting incrementally model higher spatial frequencies. Yet, unlike PCA, the boosting features and codes are sparse.

sparse coding filters.

We can also use BIP to warm-start ordinary sparse coding. Since BIP imposes additional constraints in the solution space, it converges faster to a local optimum. From there the constraints can be dropped or annealed to recover the normal sparse coding setting. In the experiment, we first run the optimization algorithm with $\alpha = 0.01$ for two epochs and then run it with $\alpha = 0$ for another two epochs. This produces identical features, both in terms of appearance and reconstruction error, to the ones obtained by standard sparse coding after 20 epochs.

For the boosting procedure we had to consider some modifications. First, the modeling power of one basis as a weak learner is often insufficient so we use a batch of several, say 10, feature vectors at each boosting iteration. This corresponds to performing block coordinate descent in function space (Rudin et al., 2004). Second, when optimizing within a new batch of features, we ignore the interaction penalty imposed by the existing features to encourage spatial exploration of the new features. This relaxation maintains a good approximation to the original BIP penalty when the number of features is small, and is less affected by the excessively greedy nature of boosting as the number of features increases. These modifications turn out to be important to prevent trivial solutions. Finally, cross-validation errors are monitored to determine the optimal stopping point of the boosting procedure. In the experiments, the optimal number of features was found to be 120.

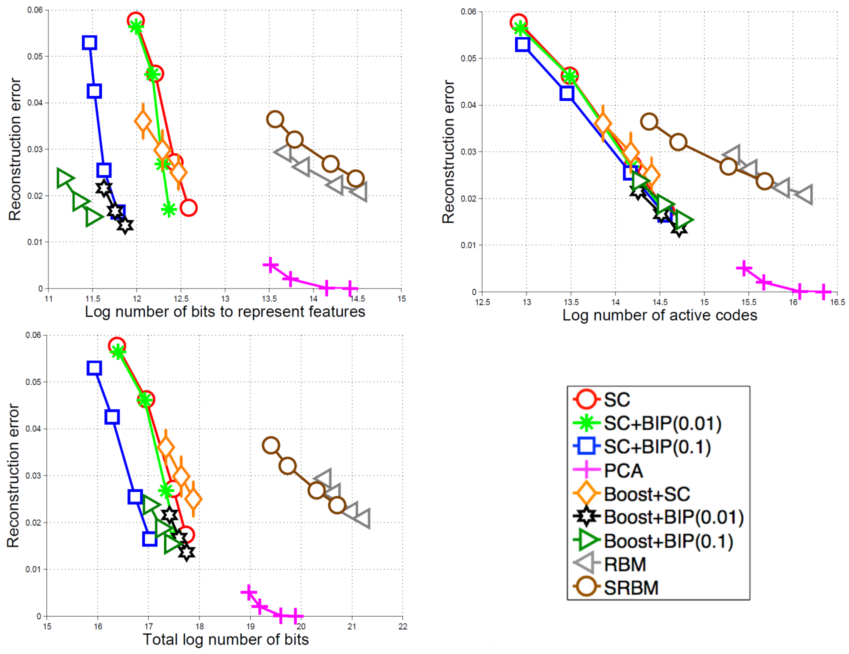

Figure 3: Number of bits needed to encode the models vs their reconstruction error on natural image patches.

The boosted features are similar to PCA features in that each batch of features explains a higher frequency band of the data. This is due to the greedy stepwise nature of boosting: each batch is given the myopic task of reconstructing the entirety of the residual as if there were no more batches coming up in the future. As a consequence, the initial features account for low frequency components of the data and leave the higher frequency components for the subsequent batches.

While it is interesting that some of the methods can produce local features, it is not immediately clear whether these features capture meaningful aspects of the data. To assess this, we compare the methods on the task of reconstructing natural image patches. We use the sigmoidal approximation to determine whether the features learned by BIP can be approximated by a feed-forward mapping. Before learning the bases, we randomly split the data set into a training set of $9,000$ images, and a test set of $1,000$ images for evaluation. For each method, we vary M between $80, 100, 150$ and 200. The exception is the boosting methods, for which $M = 120$ is optimal. For these methods, we also tried $M = 80$ and 100 to show the trend. We repeat each experiment for 10 random training/test set splits and report the error bars showing the mean $\pm$ and standard deviation.

All the methods reconstruct the image patches very well with 150 feature vectors (images not shown for lack of space). To provide quantitative results, we consider the reconstruction error of each method as a function of the number of bits needed to represent the features, the number of non-zero (active) codes and the total number of bits to encode the model (number of bits taken by the active codes and their respective features). The results appear in Figure 3.

The performance of PCA is optimal, but the bases are extremely dense. The boosting methods gradually become both sparse and accurate as BIP is strengthened by increasing α. Sparse coding with BIP delivers more compact feature representations without sacrificing reconstruction performance.

Finally, in our system, the inference time of standard sparse coding is 50 seconds for $N = 1000$ and $M = 200$, while the one for the methods with the sigmoidal approximation is approximately 0.1 seconds.

Although the SRBM and RBM are similar, the RBM seems capable of providing a lower reconstruction error. Neither are as accurate as Sparse Coding when enough hidden units are used. This is reasonable, since they are not directly optimizing a reconstruction objective, only indirectly through Contrastive Divergence. The SRBMs provide sparser models than the Gaussian RBM, but they are both far less sparse than the Sparse Coding models. This is due to the fact that the hidden units in an RBM are computed via a feedforward operation, rather than an exact optimization. A similar effect was observed in the model used in Kavukcuoglu et al. (2008a).

4.2 The Effect of Dataset Size on Feature Sparsity

We investigate how the number of training samples affects feature sparsity with both sparse coding and Gaussian RBMs. We train sparse coding with $M = 100, 150, 200$ and 300 hidden units, and in each case use $N = 5,000, 10,000$ and $30,000$ training samples. We repeat each experiment five times and report the mean and standard deviation of the number of bits required to encode the features, the number of active codes (non-zero coefficients) and the reconstruction error. The number of bits to represent bases is computed as 64 (the number of bits in a double precision floating point number) times the number of bases elements whose magnitude is greater then 0.1. The SC, BIP, Boosting and PCA bases are normalized during training, and lowering the threshold will not change the trend in the plots. In the case of RBMs, the bases are not normalized during training, and we normalize them for diagnostics. We use a threshold of 0.01 for the RBM models. The number of active codes is the number of non-zero code units. We count the number of code units whose magnitude is greater than machine epsilon for SC, BIP, Boosting and PCA. We use a threshold of 0.01 for SRBM models. Prior to conducting all experiments, a significant effort was devoted to obtaining good configurations for the learning rates, momentum terms, iterate averaging and mini-batches for online learning.

Figure 4 shows that the features become sparser as the number of training samples increases. *As the dataset increases, the size of the parts (patterns) that are common to all data instances decreases. As a result, the features become sparser.*

Note also that the reconstruction error does not appear to increase. Moreover, there is an inverse relationship between the reconstruction error and the original number of code bits M. This implies that it is possible to obtain models that require less memory storage and have better reconstruction properties by starting with a larger number of hidden units and increasing the dataset size. We ran this experiment on the RBM as well and the trend is identical to 4, larger amounts of data correspond to sparser models. In fact, as the dataset size becomes very large the need for priors such as BIP disappears.

With $M = 300$ and $N = 30,000$, the training times for sparse coding and Gaussian RBM models are typically one day and 30 minutes respectively. This is a crucial computational comparison between these two approaches. It might seem counterintuitive because sparse coding algorithms are deemed to be efficient in continuous optimization. Yet, it seems like in this realm, the naive stochastic approximation algorithms with binary codes are much more

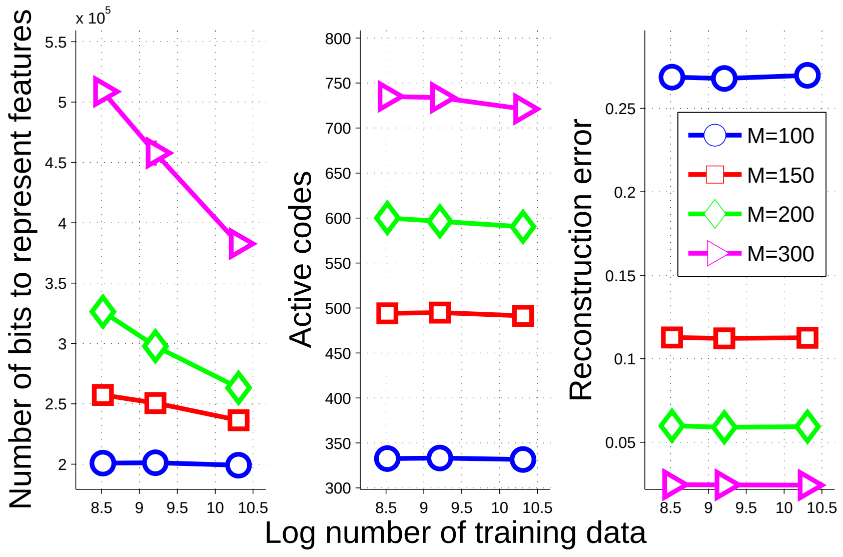

Figure 4: Number of bits to encode features, code size and reconstruction error as a function of dataset size (in log-scale) for sparse coding

efficient. From this computational standpoint, RBMs with binary codes are to be preferred.

4.3 Self Taught Learning Experiment

R. Raina and Ng. (2007) applied sparse coding in their "self-taught learning" framework. This framework consists of an unsupervised learning stage, where features are extracted from large volumes of unlabeled data, and a supervised learning stage, where the self-taught features are used to form parsimonious data representations for different supervised tasks. The main novelty of this framework is that no assumption is made about the distri-

Methods	Baseline	Boosted BIP	SC	BIP
Accuracy(%)	16	36.33	43.39	43.33
Variance($\pm$%)	—	0.2	0.6	0.8
Feature size	—	2339	4607	3712

Table 1: Classification accuracy and feature size (in bytes). The Baseline method is Fei-Fei et al. (2004).

bution of the unlabeled data. Despite having less dependence on expensive labeled data, self-taught learning replies on the feature extraction technique to produce transferable, discriminative representations. To understand the effect of BIP on the generalization properties of the features, we compare it against ordinary sparse coding in a classification task. More specifically, we directly apply the features learned in the previous experiments to classify images in Caltech101 (Fei-Fei et al., 2004).

The experiment protocol is as follows: Each image is resized and subdivided into a 4×4 grid of 14×14 non-overlapping patches. Each patch is standardized and encoded using one of the sparse coding methods to obtain a sparse code vector. Finally, all vectors within a grid are concatenated to form the representation of the image. This representation is normalized and fed into a SVM classifier with a Gaussian kernel. We randomly choose 15 images from each category as the training set and use the remaining as the test set. The 102-way classification is repeated 10 times for each method. Table 1 shows the classification accuracy.

The classification performance of BIP is almost identical to that of sparse coding, but BIP uses fewer feature elements. This result demonstrates that we can afford to enhance feature sparsity (hence energy efficiency/less storage) without compromising generalization performance.

5 Conclusions

An important direction for future work is the issue of invariant features. One could incorporate the ideas of topographic maps, convolution and pooling into our proposed methods to address this issue. Another important direction for future work is the application of these methods to the problem of learn deep representations. It is clear that the proposed methods, using a sigmoidal approximation, can be naturally extended to deep architectures. Following the strategy outlined in this paper, boosting methods could be devised to learn the structure of deep networks of composite features.

Finally, the discovery that larger datasets can result in more parsimonious feature representations deserves further investigation. If we are indeed able to learn more compact representations by increasing dataset sizes while not affecting performance, it strongly suggests the need to place more emphasis on the problem of learning to extract patterns efficiently from massive datasets.

6 Acknowledgements

This work was supported by CIFAR's Neural Computation and Adaptive Perception Program and NSERC.

References

A. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37:3327–3338, 1997.

Y. Bengio and Y. Le Cun. Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*, 2007.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, 2007.

D. Bradley and J. A. Bagnell. Convex coding. Technical Report CMU-RI-TR-09-22, Robotics Institute, Pittsburgh, PA, May 2009.

P. Buhlmann and B. Yu. Boosting, model selection, lasso and nonnegative garrote. Technical report, UC Berkeley, 2005.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. 2004.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

P. Garrigues and B. A. Olshausen. Learning horizontal connections in a sparse coding model of natural images. *Advances in Neural Information Processing Systems*, 2008.

J. Hawkins and D. George. Hierarchical temporal memory: Concepts, theory and terminology. Technical report, Numenta, 2006.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res.*, 5:1457–1469, 2004.

A. Hyvarinen, J. Hurri, and P.O. Hoyer. *Natural Image Statistics*. Springer, 2009.

H. J. Kappen. Deterministic learning rules for Boltzmann machines. *Neural Networks*, 8(4):537–548, 1995.

Y. Karklin and M. S. Lewicki. Emergence of complex cell properties by learning to generalize in natural scenes. *Nature*, pages 83–86, 2009.

K. Kavukcuoglu, M.A. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical Report CBLL-TR-2008-12-01, Computational and Biological Learning Lab, Courant Institute, NYU, 2008a.

Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU, 2008b. Tech Report CBLL-TR-2008-12-01.

H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring Strategies for Training Deep Neural Networks. *Journal of Machine Learning Research*, 1:1–40, 2009.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. *Advances in neural information processing systems*, 19:801, 2007.

H. Lee, C. Ekanadham, and A. Ng. Sparse deep belief net model for visual area V2. *Advances in Neural Information Processing Systems*, 2008.

H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*, 2009.

S. Z. Li, X. Hou, H. Zhang, and Q. Cheng. Learning spatially localized, parts-based representation. pages 207–212, 2001.

D. Lin, E. Pitler, D. P. Foster, and L. H. Ungar. In defense of L0. *Sparse Optimization and Variable Selection, Workshop, ICML/COLT/UAI*, July 2008.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004.

R. W. Lutz, M. Kalisch, and P. Buhlmann. Robustified L2 boosting. *Computational Statistics & Data Analysis*, 52(7):3331–3341, March 2008.

B. A. Olshausen. Sparse codes and spikes. In *Probabilistic Models of the Brain: Perception and Neural Function*, pages 257–272, 2001.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Res*, 37(23):3311–3325, 1997.

P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

H. Lee B. Packer R. Raina, A. Battle and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. *International Conference on Machine Learning*, 2007.

C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10):2526–2563, 2008.

C. Rudin, I. Daubechies, and R. E. Schapire. On the dynamics of boosting. In *Advances in Neural Information Processing Systems*, 2004.

R. Salakhutdinov and G. E. Hinton. Deep Boltzmann Machines. In *Artificial Intelligence and Statistics*, 2009.

M. Schmidt, G. Fung, and R. Rosales. Fast optimization methods for L1 regularization: A comparative study and two new approaches. In *European Conference on Machine Learning*, pages 286–297, 2007.