Learning Projections for Hierarchical Sparse Coding

Chaitu Ekanadham, David Ho, Daniel Wagner
CS229 Final Project
2006 December 15

Abstract

Olshausen and Field (1996) suggest a method for learning a set of Gabor-like bases that activate sparsely to reconstruct natural images. We extended this approach by learning a sparse representation of these filter activations using an adapted version of the sparse coding algorithm (TISC). We trained our models on a simplified dataset with three shapes, and then on a subset of the Caltech 101 dataset. We found that for the shapes data, SVM classification accuracy improved when using either the first or second layer responses compared to the original images, while for the Caltech data, only the first layer yielded improvement. We also attempted to learn a projection of the first layer responses for the Caltech data to use for learning a second layer, but the algorithm failed to converge. We discuss reasons why this may have occurred, and also evaluate all representations that we learn by applying SVM classification.

Background

Sparse Coding

Sparse coding theory has met with much success as a functional model for neurons in primary visual cortex. The theory stems from the landmark result in Olhausen & Field's paper (1996) demonstrating that the response properties of V1 simple cells can be learned by training a model that represents a static natural image as a linear combination of some set of basis images. The objective function to be minimized is:

$$f = ||X - B_1 S_1||_2^2 - ||S_1||_1$$

where X is a vector containing the original image data, B_1 is the matrix containing the basis images as columns, and S_1 is a vector containing the coefficient values (activations) for each basis. Thus, $B_1 S_1$ is the model's reconstruction of the image X, and so the first term in f represents reconstruction error, while the second term measures the sparseness of the coefficient values S_1 by taking its L^1 norm.

Hierarchical Sparse Coding

We focus on the problem of extracting higher-level features of images by learning a second layer of sparse-coding bases on top of the first-layer activations. One desirable property of high-level features is their invariance to small transformations, such as rotation or scaling. (Ultimately, of course, it would also be nice if this hierarchical sparse-coding could more accurately model higher levels of biological visual processing, or improve SVM classification accuracy over first-layer sparse coding alone.) Unfortunately, we can expect that a naïve strategy – using the first layer activations directly as input to a second layer of sparse coding – will not allow us to achieve this kind of invariance. Because of the sparse nature of the first-layer activations, slight transformations of the original image are likely to activate completely different bases in the first layer. Thus, the first-layer representations of the original image and the slightly-transformed image might appear mostly uncorrelated to the second-layer algorithm, making it hard to learn a "transformation-invariant" second-layer representation for both images.

Distance Metric Learning

Our approach to this problem is to *transform* the first layer activations using some linear projection, before applying the second-layer sparse coding algorithm on the transformed data. Here, the objective function would look like:

$$f = ||X - B_1 S_1||_2^2 - ||S_1||_1 + ||WS_1 - WB_2 S_2||_2^2 - ||S_2||_1$$

where X,B_1, and S_1 are as above, B_2 is the matrix with second layer bases as columns, S_2 are the coefficients of the second layer bases, and W represents some projection matrix. Note that the third term represents the squared difference between the projections (by matrix W) of the first layer outputs (S_1) and the second layer reconstruction of these outputs (B_2S_2), which can also be written as $\|S_1 - B_2S_2\|_W^2$. Intuitively, we would like to learn a projection W that maps images of the same object to be "close together," and images of different objects to be "far apart". Presumably, a good projection might allow us to achieve some degree of invariance with the second layer of sparse coding, as well as better image classification with both layers.

Conveniently, Eric Xing (2003) suggests a supervised-learning algorithm to learn such a projection. Given a matrix of similarity and dissimilarity judgments, this "distance metric learning" algorithm can learn a "distance metric" W that assigns small distances to pairs of images a human would judge as "similar", and large distances to pairs of images a human would judge as "dissimilar." (For classification tasks, we can simply label all images in the same category as similar to each other, and dissimilar to images in different categories.)

The algorithm is summarized in **Error! Reference source not found.**. It takes an initial guess for the distance metric, and then iteratively projects it to ensure that it satisfies both constraints (intuitively, the two constraints are: maximizing the distance for dissimilar pairs and keeping the distance for similar pairs under 1, and keeping the distance metric in the set of positive semi-definite matrices).

Table 1: Distance metric learning algorithm of Xing et al. (2003).

Optimization Problem	Algorithm

$\max_A \; g(A) = \sum_{(x_i,x_j)\in\mathcal{D}} \|\|x_i, x_j\|\|_A$
s.t. $\; f(A) = \sum_{(x_i,x_j)\in\mathcal{S}} \|\|x_i, x_j\|\|_A^2 \leq 1$
$A \succeq 0.$
	Iterate
	  Iterate
	  $A := \arg\min_{A'}\{\|\|A' - A\|\|_F : A' \in C_1\}$
	  $A := \arg\min_{A'}\{\|\|A' - A\|\|_F : A' \in C_2\}$
	  until A converges
	  $A := A + \alpha(\nabla_A g(A))_{\perp \nabla_A f}$
	until convergence

Key: A: distance metric; S: set of all similar pairs of data points; D: set of all dissimilar pairs of data points
x_i: data point; ||.||F: Frobenius norm;
$C2$: positive semi-definite matrices; *: convolution operator
$S(i,j)$: array with the activations of basis j to image i at each point.
$C1$: $\{A : \sum_{(x_i,x_j)\in\mathcal{S}} \|\|x_i - x_j\|\|_A^2 \leq 1\}$

Indeed, in his paper, Xing demonstrates that the supervised distance metric learning algorithm can be used as a preprocessing step to increase the effectiveness of unsupervised clustering algorithms. In our experiments, we address the question of whether distance metric learning can be applied to learn an appropriate projection matrix that can be coupled with sparse coding to yield representations that lend themselves easily to accurate classification.

Experiments and Results

Methods

We ran our experiments on two datasets. The first (87 images) consisted of 29 rotations of each of three hand-drawn shapes: a circle, a triangle, and a rectangle. The images from each class were rotated in 0.5-degree increments, from -7° to 7°. We started with this dataset because it made it easier to understand the features extracted by the second layer bases, and because we wanted to see whether the projection matrix and second layer could achieve some degree of rotational invariance (and therefore higher SVM classification accuracy). Our second dataset (60 images) consisted of 20 images from each of three randomly-chosen categories from the Caltech 101: faces, motorbikes, and starfish. Sample images from each of these datasets are shown below in Figure 1. To evaluate our results quantitatively, we fed the representations of the images at each stage into a linear-kernel support vector machine, and reported the classification error with leave-one-out cross validation.

Figure 1: original data from shapes dataset (left) and Caltech dataset (right)

Since we knew that first-layer basis learning produces bases very similar to Gabor filters (Honglak Lee, personal correspondence), and since our true aim was to evaluate the projection matrix and the second layer, we chose not to learn the first-layer bases from scratch. Instead, we used six Gabor filters as reasonable approximations for the first-layer bases that would have been learned (Figure 2). We then used the translation-invariant sparse coding algorithm developed by Roger Grosse to generate a reconstruction of the images using these bases (Figure 3). Translation-invariant sparse coding is similar to the vanilla sparse coding algorithm, except that it constructs an image using all possible translations of the basis set. The modified objective is shown in Equation 1.

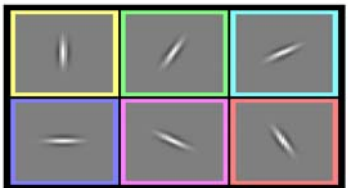

Figure 2: Gabor filters for 1st layer

Figure 3: First layer reconstructions of shapes (1st row) and Caltech images (2nd row)

$$\sum_{i=1}^{m} \|X^{(i)} - \sum_{j=1}^{n} A^{(j)} * S^{(i,j)}\|_2^2 + \beta \sum_{i,j} \|S^{(i,j)}\|_1,$$
$$\text{subject to } \|A^{(j)}\|_2^2 \leq 1, \qquad \text{for all } i, j.$$

Equation 1: TISC objective function

Shapes

To establish a baseline for the distance metric and second-layer sparse coding, we first learned the second-layer bases directly on top of the first-layer activations. Figure 4 shows the second-layer bases for the geometric data set. Each colored pixel represents a corresponding first-layer basis at that location; the intensity of the color represents the coefficient for that first-layer basis. The bases for the shape data clearly show some high-level features that are closely related to the original pictures, such as circular shapes and edges.

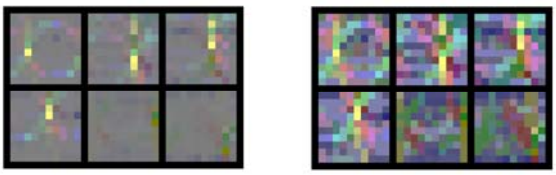

Figure 4: 2nd-layer bases for shapes dataset

Interestingly, on the shape data, we found that the classification error was already quite low on the raw images (16%), and it dropped to zero after the first layer, with or without the distance metric (Table 2). While this validated the utility of first-layer sparse coding, it did not shed light on the usefulness of the distance metric and second layer. Moreover, the distance metric learned on the first layer responses was very close to the identity matrix.

The fact that distance metric learning returned a matrix close to *I*, combined with the fact that classification performance hit ceiling so early, suggested to us that we should consider more complex data: perhaps the distance metric learning didn't need to project the first-layer representations very far, simply because they already so well clustered. Thus, we turned to the Caltech image set.

Caltech

Again, we learned the second-layer bases directly on top of the first layer, as a baseline. The bases for the Caltech data (Figure 5) are, of course, more difficult to interpret, so we moved on to the quantitative analysis (Figure 8).

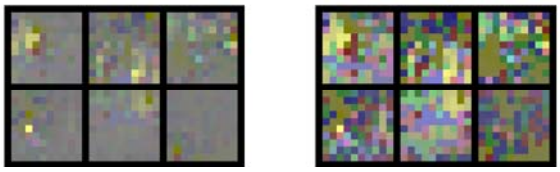

Figure 5: 2nd-layer bases for Caltech data

Again, compared to the raw images, using the first layer responses yielded a huge decrease in classification error (from 0.87 to 0.15). On this dataset, though, running second-layer sparse coding directly on the first layer actually *decreased* performance. Even so, we still hoped that the distance-metric learning could improve the accuracy.

Unfortunately, learning a distance metric on the first layer activations of the Caltech images proved to be a fragile process. The step in the algorithm that projects a guess of the distance metric onto the constraint sets C_1 and C_2 failed to converge in a reasonable number of iterations. As a result, the learned metric was again very similar to the initial guess. In the end, after the first layer, very few of the other manipulations seemed to help at all. This is likely to be due to the fact that the distance metric algorithm did not converge properly on the first-layer data.

Analysis

We verified our result in several ways:

- To ensure that the algorithm was failing to learn, (and that the identity matrix was not, in fact, the "best answer"), we initialized the algorithm once with the identity matrix and once with a random positive semi-definite matrix. In both cases, the learned metric was very similar to the initial guess.
- Furthermore, to rule out the possibility of a bug in our implementation, we tested it on hand-generated Gaussian data in two dimensions. It produced a reasonable result, and moreover, the objective function increased with each iteration of the outermost loop (Table 1). With our actual data, we had been unable to see whether the objective function increased with each iteration of the outermost loop, because the algorithm's convergence criterion is based on the difference between the guess for the distance metric on the previous iteration, and the new guess after iteratively projecting to satisfy constraint sets C_1 and C_2. Since the algorithm repeatedly failed to find a suitable projection simultaneously satisfying both constraints, the final result was very close to the original result, and the algorithm halted.

Therefore, we propose a few possible reasons for the algorithm's failure in our case:

1) The high dimensionality of the data (D = 1734 for the first layer responses), which could make a very high number of training examples necessary. In fact, the metric for which we are optimizing has size D^2. Indeed, Xing et al. (2003) had only tested the algorithm on data sets with dimensionality of up to thirty.
2) The robustness of the sparse coding algorithm, which could potentially output activations which represent significantly different data in significantly different ways under a high variety of simple transformations. Perhaps the distance metric learning code was unable to find appropriate projections for sparse, high-dimensional data.

Despite the fact that the distance metric learning failed to converge, we used the result to learn a second layer of bases. To transform the first-layer activations, we multiply by a matrix A such that A^TA is the learned distance metric. The results are shown below for both initializations (Figure 6 and Figure 7).

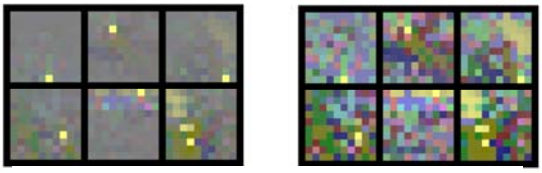

Figure 6: 2nd-layer bases on Caltech data, with distance metric initialized to *I*

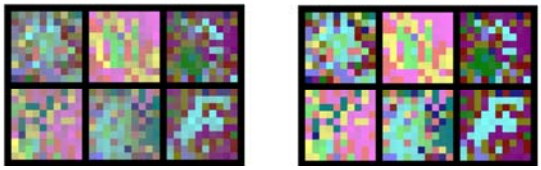

Figure 7: 2nd-layer bases on Caltech data, with distance metric initialized to random PSD matrix

Classification results

We fed the following as input to a standard SVM classifier:

1. original images (both datasets)
2. first layer responses (both datasets)
3. second layer responses without distance metric applied (both datasets)
4. first layer responses with distance metric applied (Caltech 101 only)
5. second layer responses learned with distance metric (Caltech 101 only)

For parts (4) and (5) we used 4-fold cross validation. The results for the shapes data is shown in Table 2. Also shown are the results on the Caltech data (Table 3 and Figure 8).

Table 2: Shape classification errors

	Triangles	*Rectangles*	*Circles*	*Overall*
Baseline	0.4828	0	0	0.1609
First layer	0	0	0	0
First layer with distance metric	0	0	0	0
Second layer	0	0	0	0

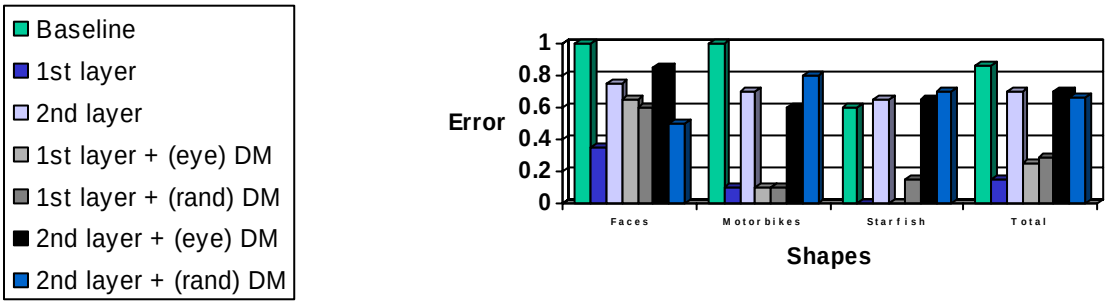

Figure 8: Caltech classification errors

Conclusions

Our initial reaction was that naïvely applying the sparse coding algorithm to the first-layer activations would not be useful. While this turned out to be true, this is likely to be a testament to the effectiveness of the first layer of sparse coding rather than to our ingenuity and insight. All our tests indicate that the first layer is so effective than any post-processing only detracts from classification performance. At the very least, we can say that neither the simple-minded approach nor the distance-metric learning approach outlined here are appropriate next steps. We hold out hope, however. Sparse coding is a linear transformation of data, and so is applying a distance metric. Combining two linear transformations yields another linear transformation, and this may perhaps explain the uselessness of using the metric we learned. Yet there may be some nonlinear post-processing step that could allow a second layer of sparse coding to learn a more interesting set of features, and ultimately improve our understanding of vision.

References

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

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning with application to clustering with side information. In S. Thrun S. Becker and K. Obermayer, editors, Advances in Neural Information Processing Systems 15, pages 505–512, Cambridge, MA, 2003. MIT Press.

Yang, L. (2006) PhD thesis. Distance Metric Learning: A Comprehensive Survey. Department of Computer Science and Engineering, Michigan State University.

Appendix: Caltech classification errors

Table 3: Caltech classification errors

	Faces	*Motorbikes*	*Starfish*	*Overall*
Baseline	1	1	0.6	0.8667
1st layer	0.35	0.1	0	0.15
2nd layer	0.75	0.7	0.65	0.7
1st layer + (eye) DM	0.65	0.1	0	0.25
1st layer + (rand) DM	0.6	0.1	0.15	0.2833
2nd layer + (eye) DM	0.85	0.6	0.65	0.7
2nd layer + (rand) DM	0.5	0.8	0.7	0.6667