David Vetrano CS229

Learning Unsupervised Features from Objects

Unsupervised feature learning can be a powerful technique for some machine learning problems. This project explores the use of a convolution deep belief network (CDBN) to learn unsupervised features from images. Deep belief networks (DBN) and CDBNs have previously been applied to many problems, from learning to generate facial expressions (Susskind et al., 2009) to unsupervised feature learning for audio classification (Lee et al., 2009b).

Here, a CDBN is used to learn unsupervised features from two categories in the Caltech-256 dataset (Griffin et al., 2007). After learning features, a simple linear SVM is used to predict object category using supervised information. The results are compared against tiny versions of the images as a baseline and histogram of oriented gradients (HOG) features, which have successful for instance in person detection (Dalal and Triggs, 2004).

DBNs share many commonalities with the CDBN. Both algorithms have a hierarchical structure where each layer learns to model statistical dependencies between variables. In a DBN, this 'layer' is called a Restricted Boltzmann Machine (RBM). An RBM is a graphical model with two disjoint sets of units, called visible and hidden units. Hidden units model unobserved “causes” of the visible unit activations. Explicitly computing the gradient of its parameters is extremely expensive and infeasible in practice. However, the RBM can be trained efficiently because the layers are conditionally independent on one another. Using Gibbs sampling we can alternatingly sample from the conditional distributions of one layer given the other. Particularly, using Contrastive Divergence (CD) we sample very few times; in practice, even one up-down-up pass can be quite effective (Carreira-Perpinan and Hinton, 2005).

CDBNs, as proposed by Lee et al., introduce two significant modifications to the DBN (2009a). First, instead of learning weights for all positions in the image, we instead learn a bank of filters that are convolved with a given image, taking advantage of the implicit structure of an image. Second, to encourage additional invariance, a pooling layer is added above each CRBM. To allow for a fully generative model, Lee et al. propose a new method of pooling, which they call probabilistic max-pooling (2009a).

Method

Matlab code for a simple CRBM, provided by Honglak Lee at <http://ai.stanford.edu/~hllee/softwares/icml09.htm> was used as a starting point. Several important changes were implemented. First, code was adapted to aid in exploring the effects of different parameters. Code was written to walk

through parameter space and save results at each step. In practice, running several instances simultaneously took around 120 hours before a suitable set of parameters for the first layer were learned. Second, following Lee et al. (2008), a sparsity penalty was added to encourage the learned bases to be more sparse as

$$\begin{aligned} S(W_{i,j}^k, b_k) &= \lambda \sum_{i,j} |p - \frac{1}{m} \sum_{l=1}^{m} E(H_{i,j}^{k,l} | v^l)|^2 \\ &= \lambda \sum_{i,j} |p - \frac{1}{m} \sum_{l=1}^{m} U_{i,j}^{k,l})|^2 \end{aligned}$$

Using CD, the CRBM already approximates derivatives for its parameters. However, we can easily compute the exact gradient of the sparsity term. During one epoch, several mini-batches were first run. Afterward, one step of gradient descent was taken on the sparsity penalty term. Following Lee et al. (2008), only the partial derivatives of the bias parameters were computed and used for gradient descent.

Two other interesting options were implemented in the CRBM. First, an option to add a zero-border of a certain size to the image was added, following a suggestion by Krizkevsky. This feature was found to be necessary to learn first layer bases. Second, a debug mode was implemented to watch the visible data, reconstructed visible data, hidden unit activations, and weight changes to the CRBM during training. This was found to be a useful tool when exploring different parameter choices manually.

After CRBM code was written, bases were learned. Following Lee et al. (2009a), the first layer bases were learned from the Kyoto natural images dataset. Two forms of preprocessing were used. First, images were whitened to flatten the power spectrum. Afterward, image data was transformed to have mean 0 and variance 0.1.

To explore the effectiveness of a CDBN in learning unsupervised features, the Caltech-256 dataset was used to learn second-layer bases from the “tshirt” and “billiards” categories. Due to time restrictions, the parameter space could not be adequately explored and these results are tentative. The results indicate that these bases were moderately effective in classification.

Results

First layer bases were difficult to learn using only the CRBM with sparsity penalty. Typically, the learned bases focused on the corners and did a poor job of modeling the visible data (see Figure 1). Following Krizhevsky, a zero-border half the width of the bases was added to the image after rescaling. This modification was quite helpful (see Figure 1). These features are generally centered

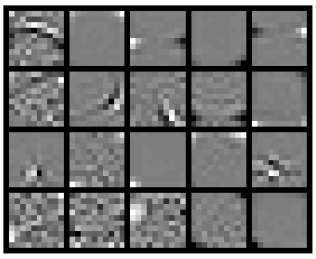
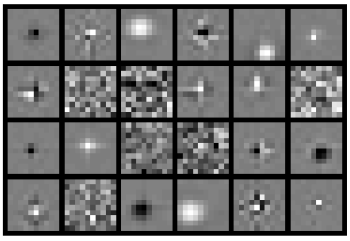


Figure 1. 20 First-layer bases learned without a zero-border



24 First-layer bases after a zero-border was added

and are closer to the "expected" result. However, they fail to capture orientation information and mostly exhibit radial symmetry. In addition, the CRBM did not take advantage of all of the bases effectively.

To address this issue, whitening was performed as a preprocessing step. Whitening significantly improved the results, as can be seen in Figure 2. The bases learned capture much more interesting information. For example, many bases are orientation-dependent.

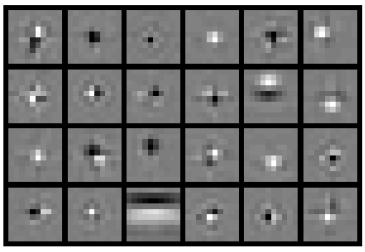


Figure 2: First-layer bases after whitening

Finally, layer two bases are presented in Figure 3. These bases were learned using a pooling factor of C=2 and were trained on both categories. The first row contains hand-selected "interesting" features and the rest were selected randomly from the 480 layer two bases. Although some bases are quite noisy, the CDBN does learn Gabor-like features as well as interesting contours.

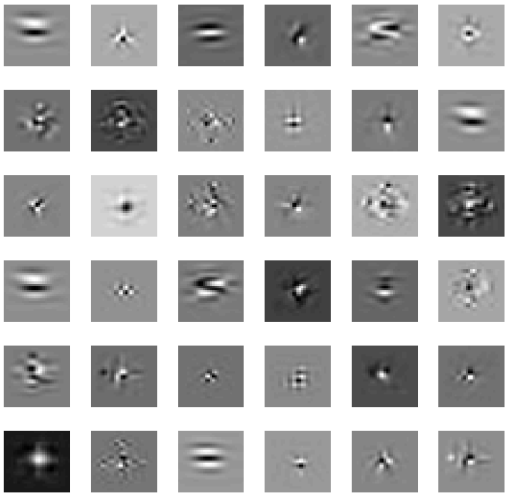


Figure 3: 36 second-layer bases learned from objects

After learning features from the data, the features were evaluated by their performance in object classification using an SVM (libsvm). Features were constructed by convolving the learned bases over each input image and then scaling the result. Except where noted, all results are cross-validated. These results use only a linear kernel. Although other kernels could be used, the linear kernel appears to work fairly well. In addition, since the feature space is highly-dimensional for certain evaluations and we have few examples, using for instance an RBF kernel may not be a panacea. Finally, for simplicity, the dataset contains exactly 278 images, the minimum, from each of the categories "tshirt" and "billiards." Thus, chance performance is 50%.

As a baseline, images from each category were simply scaled and used as input to the SVM. We can imagine, for instance, that “billiards” images tend to be more illuminated on the bottom half while “tshirt” images are illuminated more uniformly. In this case, the raw image data could be quite successful at classifying the images. Images of size 6x6 were found to offer the best performance of tested sizes, achieving 68.52% accuracy (see Figure 4 for all results).

Next, the first-layer features were used as input to the SVM as described. Rescaling the result to 6x6 again produced the best result, achieving 74.12% accuracy. This gives interesting insight. First, the features learned on natural images do perform fairly well on classification of objects. Second, since we find significantly better performance than using the raw image data, these features capture more valuable local information about the images than simply the means for image regions (see Figure 4).

As a comparison, simple HOG features were used as input to an SVM. In this experiment, histogram features were computed globally for the entire image. The best performance was achieved using 16 HOG features, correctly classifying 84.55% of the images. In a sense, these features are extremely local as they operate on only several adjacent pixels. However, the relative success of this method shows that this information is quite useful at classifying the images.

Finally, the learned layer 2 features were evaluated. One problem in testing these features is that the resulting input to the SVM is high-dimensional. For example, using all bases with even 6x6 scaled feature data results in 17280-dimensional data. As a simple experiment, the data for each feature was scaled to 1x1 (71.23% accuracy) and 2x2 (72.31% accuracy). Compressing down the response from each filter performs a bit worse than the first layer features. Larger data sizes were also tried, with limited effectiveness. With only 556 total examples, 501 of which are in the training set, the SVM could not effectively learn to classify the images.

To address this issue, two tricks were tested. First, only a small subset of randomly-selected bases were used. Using only 50 bases of the total 480 and crushing down the responses to 4x4 resulted in 70.49% accuracy, which is worse than achieved with the first-layer bases and close to raw image performance. Other selections for the number of bases and scaled response size were approximately equivalent and did not outperform even the 2x2 scaled responses. PCA was also performed to find linear combinations of features that had the greatest variance over all images. However, the results are underwhelming, resulting in comparable to worse performance (not included).

Name	Cross-validated?	Total dimensionality	Performance
Plain images, 2x2	Y	4	59.18%
Plain images, 4x4	Y	16	65.30%
Plain images, 6x6	Y	36	**68.52%**
Plain images, 8x8	Y	64	66.03%
Plain images, 16x16	Y	256	60.63%
Plain images, 32x32	Y	1024	61.34%
First layer bases, 4x4	Y	384	73.21%
First layer bases, 6x6	Y	864	**74.12%**
First layer bases, 8x8	Y	1536	74.12%
HOG, 8 features	Y	8	82.39%
HOG, 12 features	Y	12	84.37%
HOG, 16 features	Y	16	**84.55%**
HOG, 32 features	N	32	83.64%
HOG, 64 features	N	64	82.14%
Second layer bases, 1x1	Y	480	71.23%
Second layer bases, 2x2	Y	1920	**72.31%**
Random 50 second-layer bases, 4x4	Y	800	70.49%

Figure 4: Results of all tests

Discussion

While deep belief networks can be quite powerful, getting all of the parameters right in a system can be quite difficult. For this problem, features from the first layer significantly outperformed raw image data. The second-layer bases performed poorly, although it is suspected that this performance is mostly due to the difficulty for the linear SVM to separate the examples. A simple implementation of HOG outperformed all other features. I would like to continue exploring this type of problem more thoroughly and using a GPU to make the computational times more tolerable.

Works Cited

Carreira-Perpignan, M. and Hinton, G. “On contrastive divergence learning.” Artificial intelligence and statistics, pages 33-41. Fort Lauderdale, 2005. Society for Artificial Intelligence and Statistics.

Dalal, N. and Triggs, W. “Histograms of Oriented Gradients for Human Detection.” 2005 IEEE CVPR05. Vol 1(3), p. 886-93. 2004.

Griffin, Gregory and Holub, Alex and Perona, Pietro. “Caltech-256 Object Category Dataset.” California Institute of Technology. 2007.

Krizhevsky, A. Convolutional Deep Belief Networks on CIFAR-10. Unpublished manuscript, 2010.

Lee, H., Grosse, R., Ranganath, R., Ng, A.Y. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchial Representations,” *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009a.

Lee, H., Largman, Y., Pham, P., Ng, A. “Unsupervised feature learning for audio classification using convolutional deep belief networks,” Advances in neural information processing systems [1049-5258] 2009b vol:22 pg:1096.

Susskind, J.M., Hinton, G.E., Movellan, J.R., Anderson, A.K. “Generating facial expressions with deep belief nets,” Affective Computing, Emotional Modelling, Synthesis and Recognition, ARS Publishers, pp. 421-40, 2008.