Unsupervised Dictionary Learning for Large-Scale Natural Image Classification

Rui Zhang, Yang Yang

Advisor: Ph.D Kihyuk Sohn, Professor Honglak Lee
Department of CSE, University of Michigan

1 Introduction

Recent work in machine learning has been proven successful on object recognition task. For instance, best digit-classification accuracy on MNIST dataset rivals that of human-beings (Cireşan et al., 2012); (Coates et al., 2011) has achieved state-of-art performance on both CIFAR and NORB benchmarks; breakthrough on scalable visual recognition has been made by (Krizhevsky et al., 2012) via efficient implementation of deep convolutional networks.

However, there are still some limitations we can observe. Most progress was made with human-labeled dataset. The size of training set and test set of MNIST is 60000 and 10000 respectively (LeCun et al., 1998). Moreover, convolutional network shows impressive performance on large-scale task, but the supervise-styled training entails dealing with millions with parameters and multi-layer deep architecture.

In this work, we are interested in classifying natural images with unsupervised learning algorithms. We construct an efficient yet simple feature learning system trained on millions of images cropped from Youtube video. The system utilized clustering algorithms to train two layers of dictionaries which are then used for feature mapping and classification. Testing on dataset derived from Labeled Face in the Wild, we can achieve best AUC of PR curve as much as 95.79%.

2 Algorithm and Architecture

We make essential use of several dictionary learning and clustering algorithms in our system. In section 2.1 to section 2.3, we introduce several building block algorithms, and finally in section 2.4, the whole training pipeline together with the classifier is delineated.

2.1 Orthogonal Matching Pursuit

Given a set of basis functions and a data vector, Orthogonal Matching Pursuit (Pati et al., 1993, Szlam et al, 2010) is a greedy coding algorithm to find coding representation of data. During each iteration, OMP selects the most correlated basis function, which is a column of dictionary matrix in our representation, according to the activation for the current residual. The coding vector is then updated to reflect the selected dictionary and its corresponding activation. With fixed number of iteration k, OMP-k algorithm tries to minimize following objective function:

$$f(z) = \parallel Dz - x \parallel^2,$$

where $D \in \mathbb{R}^{n\times d}$is the dictionary, $x \in \mathbb{R}^n$is data vector, and $z \in \mathbb{R}^d$is coding subject to $\parallel z \parallel_0 \leq k$.

Algorithm 1 Orthogonal Matching Pursuit-k

```
1: Input:Data vector of dimension n: $x \in \mathbb{R}^n$, Dictionary matrix with d filters:$D \in \mathbb{R}^{n\times d}$
2: Output: Coding of data: $z \in \mathbb{R}^d$
3: Objective function: $\min_z \parallel Dz - x \parallel^2$, where $z \in \mathbb{R}^d$, $\parallel z \parallel_0 \leq k$
4: Intialize: $R^0 = x$, $m = 0$, $z = \overrightarrow{0}$
5: while $m < k$ do
6:     $a = \max_j \parallel D_j^T x \parallel$
7:     $i = \arg\max_j \parallel D_j^T x \parallel$
8:     $R^{m+1} = R^m - aD_i$
9:     $z_j = z_j + a$
10:    $m = m + 1$
11: end while
```

2.2 Dictionary Learning

OMP-k is used to code the data with prepared dictionary. If data only is provided, it turns out an alternating procedure is available to learn the dictionary and code in the same time. When k is chosen to be 1, the OMP learning degenerates to one iteration, and coding is equivalent to clustering assigning of data vectors based on absolute value of activation. To update a filter, we take the sum of data assigned to this cluster, weighted by the response. Then we keep updating codes and dictionaries for a fixed times of steps. This dictionary learning procedure can then be summarized as follows.

Algorithm 2 Dictionary Learning with OMP-1

```
1: Input: $m$ $n$-dimension data vectors: $X \in \mathbb{R}^{n\times m}$
2: Output: dictionary matrix with d filters: $D \in \mathbb{R}^{n\times d}$
3: Initialize: $D = \text{norm}(\text{rand}(n,d))$
4:         rand(): random number genrator
5:         norm(): rescale columns to be unit length
6:
7: Repeat for fixed number of times:
8:         Apply Algorithm 1 on $x^i, \forall i$
9:         $\forall j$, retrieve all coding $z^i$ with $z_j^i \neq 0$ and corresponing data vector $x^i$
10:        Update $D_j = \text{norm}(\sum_i z_j^i x^i)$
```

It is interesting how the procedure resembles standard K-means clustering algorithms. In K-means, data points are assigned to the closest centroid in terms of Euclidean distance, and centroids are update by taking average of data points with the same assignment. The proposed dictionary learning algorithm above can be regarded as modified K-means with different clustering criterion and updating rule.

2.3 Max Pooling and Affinity Propagation

Pooling is a frequently applied operation on mapped features. During max pooling process, the features are grouped into different regions within which only the maximum feature is selected. With fewer features representing the data, pooling is essential to avoid intensive computation and over-fitting issues (UFLDL Tutorial).

In our case, we construct pooling regions by grouping features mapped from similar filters. (Coates et al., 2012) used single-link agglomerative clustering as grouping algorithm. Filters within the pre-selected Euclidean distance are clustered together, as long as the diameter of the group does not exceed certain threshold. In practice, we find it difficult to find appropriate parameters: smaller thresholds will produce too many groups, while larger ones will allow filters of wide difference in the same group.

To cluster the filters, we use "affinity propagation" (Frey et al., 2007) to approach the problem. Rather than focusing only on learned filters alone, affinity propagation takes as input the response to filters of the data. Messages are transmitted until a good set of clusters are formed. We use off-the-shelf implementation of the algorithms, which can cluster the filters into exactly certain number of groups.

2.4 Learning Architecture and Classifier

Now we are ready to present the learning architecture and our classifier. We adopt model similar as that in (Coates et al., 2012). The system is built of three layers of hierarchy: two of which are dictionary learning with max-pooling layer in between. After the pipeline is finished, we will have two sets of dictionary, and a grouping assignment to the first set. The learned dictionary is afterwards available to map from given images to their feature representations, which then is directly used for classification.

Given a dataset of 32-by-32 pixel images, data vectors are constructed by dividing images into 16 non-overlapping 8-by-8 patches. The first layer is connected to those 8-by-8 data patches, on which dictionary learning algorithm described above is applied. The output of first layer is a set of k_1 8-by-8 clusters. First layer responses are constructed by taking the absolute value of inner product of data points and filters. Receiving those responses, the second layer then utilizes affinity propagation algorithm to partition k_1 clusters into exactly G groups. According to grouping assignment, responses are max-pooled within each group. At this point, we have 16 data vectors making up the original 32-by-32 pixel images, and each data vector is mapped to G max-pooled responses. We then reshape the responses of the same image to $16G$-dimension data vectors. This forms input to the third layer, where the same dictionary learning procedure proceeds to find k_2 filters.

After going through aforementioned pipeline, we are prepared with two sets of dictionary and a grouping policy for the first-layer features. Armed with filters and groups, it is then convenient to map a given 32-by-32 image to its feature representation. The process is similar as training procedure. The image is first divided into 16 64-dimension data vectors. Each data point will have $k_1$responses, which is absolute value of dot product with each filter in first set of dictionary. According to grouping assignment, first layer responses is max-pooled to produce G responses out of $k_1$features. Those G responses for 16 data vectors are reshaped to $16G$-dimension feature as an entirety. The response to second set of dictionary is simply dot product, without taking absolute value.

Finally, the classifier will be the third layer filter itself. Although it is standard to train a top level classifier using softmax regression model or linear SVM, it is not feasible in our case because the natural images are not labeled. Instead, every filter in the dictionary can be a classifier itself: the image with higher response will be considered with greater possibility to be in a particular category.

3 Experiments

With the preceding algorithms and system, we train our dictionary matrix using natural images downloaded from Youtube video (Coates et al., 2012). Since we are particularly interested in identifying faces in the image, we conduct a spectrum of experiments with various positive example proportions and sizes. Each third layer filter will be tested as a potential face detector against our testing dataset, which consists of randomly selected negative examples and Labeled Face in the Wild images (Huang et al., 2007) in seven different scales and positions. We succeed in finding impressive face detectors in different experiments. The best one reaches 95.79% AUC of the PR curve.

Datasets

To construct our training dataset, we cropped images from Youtube dataset. The raw data set contains 1.2 millions random video images of various sizes (128-by-96 is a typical one). We manages to cut off 0.2 million 64-by-64 face images as our positive example pool. Figure 1 shows five such samples. Negative examples are randomly selected 32-by-32 images. An array of experiments take as input the dataset with different number and portion of positive examples.

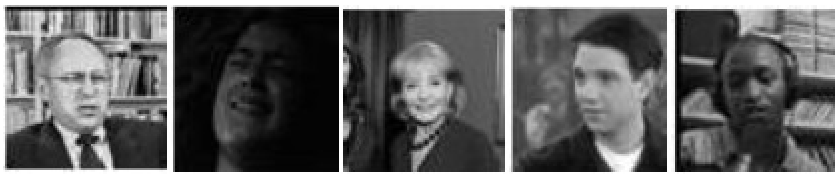

Figure 1: 64-x-64 face images from Youtube dataset

To artificially augment dataset, (Krizhevsky et al., 2012) manully divided 64-by-64 images into five parts (four corner patches and the center patch). In our experiments, we choose to sample 32-by-32 images from our pool every five iterations during training of our first layer dictionary. This trick ensures that the dictionary sees an enriched set of faces, while keeping dataset small to save computation expense. Thereafter, as mentioned before, each 32-by-32 images are transformed to 16 64-dimension data points. Before fed into network, the data vectors goes through preprocessing including mean-removing, normalizing, and ZCA whitening.

We test our filters against Labeled Face in the Wild (LFW). LFW consists of 13233 250-by-250 images centered at faces. For each example, we produce 7 32-by-32 images with different scales and positions, abundantly enlarge our testing dataset. Figure2 shows an example of original face images, and its derived 7 images. Another five times non-face images are added into testing dataset as negative examples.

Figure 2: LFW dataset

Results

We perform eight experiments with different parameters summarized in Table 1. The most successful trains 640 first layer filters (partitioned into 128 groups) and 15000 second layer filters on 1 millions training images with one fifth positive examples, achieving best AUC as 95.79% on our LFW testing dataset. In the most difficult setting, we can still learn good dictionaries with 87.94% AUC, where only 20000 face images are in the training set.

ID	Data size(Face:Non-Face)	Number of filters	Max AUC
1	1 million(1:4)	640 15000	95.79%
2	1 million(1:4)	1280 30000	28.91%
3	1 million(1:4)	3200 75000	31.84%
4	2 millions(1:10)	640 15000	23.41%
5	2 millions(1:10)	640 15000	91.76%
6	4 millions(1:20)	640 15000	26.35%
7	2 millions(1:50)	640 15000	66.00%
8	2 millions(1:100)	640 15000	87.94%

Table 1: My caption

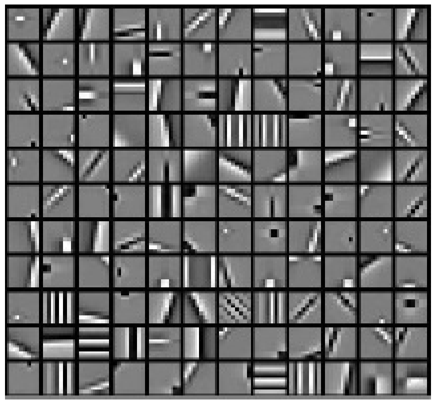

Figure 3: Lower level dictionary

As the table shows, the success of learning is highly sensitive to the parameters we are using. First, more positive examples in the training set help improve the performance. The best filter

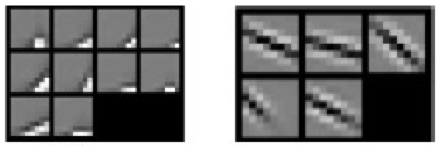

Figure 4: Group of first layer filters

sees largest number of faces images, 0.2 million faces out of 1 million training dataset. It is not surprising because if we present more faces in the training set, it is easier for the clustering algorithm to catch the pattern. Moreover, appropriate numbers of filters is essential to ensure excellent filters. If the number of filters is doubled or even larger, we can no longer find good face detectors as demonstrated in comparison among Experiment 1,2,3. Simply increasing the number of filters might not be a good choice, in that it is unlikely to “over-fit” the objects.

To verify our algorithms and classifiers, we further make statistics and visualization on Experiment 1 in different ways. When the best filter can reach AUC as high as nearly 96%, many other good face detectors are equally impressive. As estimated, there are 198 filters with AUC higher than 90%. Besides, to find filters specialized for faces with certain position and scale, we categorized top 100 stimuli for each detector into the seven types (described in Dataset section). Table 2 summarized the statistics for seven most specialized filters for each type of face.

AUC Rank	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7	None Face
105	69	0	3	14	13	0	1	0
576	1	58	0	4	10	1	7	19
622	0	0	83	1	0	7	1	8
256	2	0	3	82	0	8	2	3
214	0	0	2	3	73	5	9	8
640	0	0	5	1	3	74	0	17
510	0	11	12	0	4	5	54	14

Table 2: Top 100 stimuli for seven specialized filters

We then visualize randomly selected 132 first layer filters, and 2 groups generated by affinity propagation clustering. As presented in Figure 3, the dictionary contains a wide range of features, and similar ones successfully merge into one group.

For the higher level dictionary, we cannot display them directly because they are not connected to input images. Rather, we visualize them by finding their highest-activating images in the testing dataset. Figure 4 shows the top 100 stimuli for the filters specialized in type 2 face. Figure 5 shows the mean of top hundred stimuli for seven specialized filters.

Acknowledgements

We would like to give our great thankfulness to Kihyuk Sohn and Honglak Lee, for their support in terms of both theoretical guidance and computing environment access.

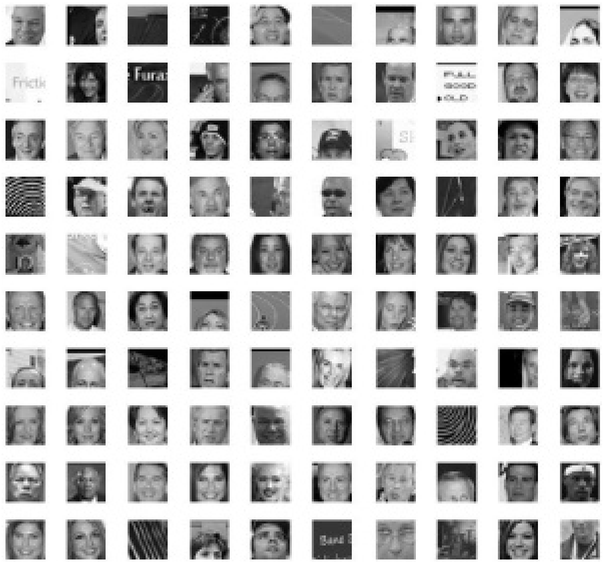

Figure 5: Top 100 stimuli of filters specialized for type 2 face

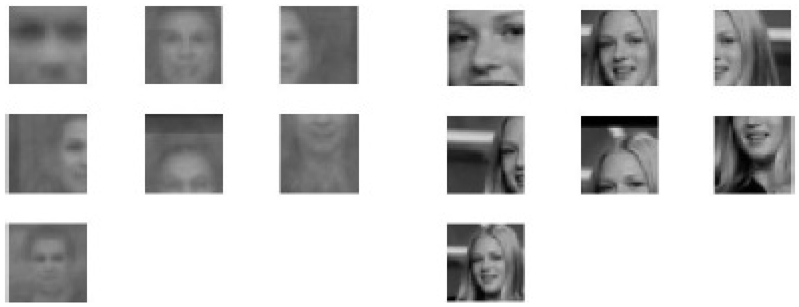

Figure 6: Mean of top simuli for seven specialized filters

References

G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. University of Massachusetts, Amherst, Technical Report 07-49, October, 2007.

A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning*, pages 921-928,2011.

A. Coates, A. Karpathy, and A. Y. Ng. Emergence of Object-Selective Features in Unsupervised Feature Learning. In NIPS, 2012

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In NIPS, 2012.

A. Szlam, K. Kavukcuoglu, and Y. LeCun. Convolutional Matching Pursuit and Dictionary Learning. arXiv:1010.0422, 2010.

//M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. arXiv 1311.2901, 2013.

D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. In CVPR, 2012.

Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Application to Wavelet Decomposition. In *Annual Asilomar Conference on Signals Systems and Computers*, 1993

B. J. Frey and D. Dueck. Clustering by Passing Messages Between Data Points. In *Science* 315, Feb 2007

A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen. UFLDL Tutorial. http://ufldl.stanford.edu/wiki/index.php.

Y. LeCun, L. Bottou, Y. Bengion, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.