Progress Report
19-31 October 2012

Amin Karbasi
E-mail: amin,karbasi@epfl.ch
Amir Hesam Salavati
E-mail: hesam.salavati@epfl.ch

Supervisor: Prof. Amin Shokrollahi
E-mail: amin.shokrollahi@epfl.ch

Algorithmics Laboratory (ALGO)
Ecole Polytechnique Federale de Lausanne (EPFL)

November 8, 2012

1 Summary

In the past days, we have continued working on applying our method to a dataset of natural images. We have managed to explore different approaches, some with success, some with failure and some promising avenues to explore further. We will discuss these techniques in more details in this report. To this end, we have also read the paper "Sparse Filtering" which addresses a novel way of feature extraction for image classification purposes. We will use their approach extensively in our simulations.

On another topic, we read a very nice survey on some connections between compressed sensing and brain activities. The paper, entitled "Compressed Sensing, Sparsity, and Dimensionality in Neuronal Information Processing and Data Analysis", summarizes compressed sensing achievements first. Then, it discusses applications of compressed sensing in analyzing brain activities and improving imaging methods. Finally, the authors consider various compressed sensing techniques performed by the brain. This last section is very interesting and provides many nice ideas for future research activities on using compressed sensing ideas to design more efficient neural networks.

2 Continuing our endeavors on reversible feature extraction models

Following the research lines of past few weeks, we continued looking for feature extracting models that allow us to reconstruct the image from its features while the features form a subspace as well. We tried a couple of different approaches, which are summarized below together with the outcome.

2.1 Check Olshausen model once more

Given that previous attempts on learning the sparse features returned by the Sparse Coding method [2] had failed, we tried to see if the features for the neighboring patches are similar at all. Because if they are, one can easily devise a method to take this similarity into account and enforce linear constraints.

Result: Failed. The feature coefficients were not similar. Note that the features correspond to applying the Sparse Coding methods to a set of preprocessed natural images who belong to one call of the STL-10 dataset. We also applied this method twice, i.e. add an additional layer of feature extraction. However, the result was the same and the learning algorithm could not find a dual space.

2.2 Keep only dominant features after applying PCA

Another approach that we have tried is the following:

1. Pre-process the set of images according to the method mentioned in the previous report.
2. Divide each image into smaller *non-overlapping* patches.

3. Apply Principle Component Analysis (to each matrix containing a specific patch of all images in the database).[1]

4. In each pattern, only keep the *coefficients* that are greater than a defined threshold. This way, we express *each* pattern in terms of the dominating principle components for that pattern.

5. Store these dominating coefficients as the new patterns (or as an alternative approach, reconstruct the original images sing only the dominating coefficients).

6. Divide these new patterns into *overlapping* clusters and apply our learning algorithm to each cluster.

Result: Failed. Overall, our learning algorithm had difficulties learning the dual vectors for the new patterns.

2.3 Projecting input patterns to a subspace

Another trick that one might use is to project the input onto a subspace defined by the dominating principle components of the data.

Result: Failure for now, promising for future! This approach is not suitable for our current setting since it projects everything on a subspace, even input noise or patterns not belonging to the set of memorized patterns. Therefore, there is a big chance that our learning algorithm learns the dual basis of the projection matrix and return the all-zero vector for any input pattern! So this approach will fail if the goal is to identify and eliminate **input noise**. However, the same approach might be successful if the goal is to denoise **internal noise of** the neurons. We will investigate this topic later.

3 Irreversible feature extraction models

Given that our approaches on learning reversible feature extraction methods did not succeed, we switched our attention towards models in which features are extracted for the purpose of object recognition/classification. These methods differ from the previous approaches in that using the features one can not reconstruct the input patterns with all their details but could use this information for the purpose of object classification. For the feature extraction technique, we have used the *Sparse Filtering* technique proposed in [1]. In short, here are the steps done in [1] for pattern classification over the STL-10 dataset:

1. [*optinal*]: Pre-process images according to the method mentioned in [3].[2]

[1] Note that one might think of applying PCA to the original image, instead of each smaller patch. However, this approach will not work in case the number of images are smaller that the set of pixels. For instance, if we have 500 images of size 100×100 pixels, then in the correlation matrix we will have 500 non-zero eigenvalues and the rest of 9500 eigenvalues will be zero. Therefore, this approach will not give us any additional information. However, if we consider patches of size 10×10, we will see some of the 100 eigenvalues to be close to zero, which is a very useful piece of information.

[2] As mentioned in [1], this step is not necessary for *computer vision* applications. But in general, such *whitenning* steps might be helpful.

2. From each input image of size $N \times N$, extract patches of size $w \times w$. This can be done by considering a $w \times w$ window which spans the image horizontally and vertically, and in each step, we shift it by s pixels (s is usually called *stride*). So if $s > w$, we will have non-overlapping features. This way, we get $N - w + 1 \times N - w + 1$ patches.

    - In [1] the authors propose to pick s very small so that we have dense feature extraction. Specifically, they choose $s = 1$.

3. For each patch, learn k features. The MATLAB code provided by the authors can be used for this purpose.

4. Gather the features for each image in the following way: let us label the feature ℓ corresponding to the patch centered at position (i, j) of the image by f_{ij}^{ℓ}. Then, for each image, we can construct k 2D arrays of size $N - w + 1 \times N - w + 1$, where array ℓ is formed by replacing patch (i, j) of the image by its feature f_{ij}^{ℓ}. Now to pool the features, we could

5. Pool the learned features. Here is the way the authors in [1] do it: divide each of the k 2D arrays above into four quadrants. In each quadrant, calculate the sum of all features within the quadrant and assign it as the value representing the quadrant. Therefore, at this point we have k arrays of 2×2, or $4k$ features for each image.

6. The authors then train the classifier based on these $4k$ features for each image.

Now we adopt a very similar procedure to what explained above and use the result to learn the dual matrix of the features. We could either do the learning on the results of *step 4* or *step 5* above. So far, we have used the result of **step 5** in our simulations since it has fewer parameters. More specifically, here is the process we have performed before applying our learning algorithm.[3]

1. **Do not** perform preprocessing over the classes of STL-10 dataset except for mapping them to gray-scale.

2. Extract patches of size 12×12 from 96×96 images (i.e. $N = 96$ and $w = 12$) with *stride* of $s = 4$.

3. Put the features in a big matrix with size $144 \times C$, where C is the number of images in the dataset.

4. Apply the *Sparse Coding* technique using the MATLAB code provided by the authors of [1] (available here).

5. Pool the features of the second feature extracting layer by dividing each of k 2D feature arrays into 5×5 regions (instead of the 4 quadrants done in [1]).

6. At this point, for each of C images in the database we have $k = 144$ arrays of size 5×5. We could merge these arrays into a 60×60 array by replacing each pixel of the 5×5 array by an image of size 12×12, i.e. by first reshaping the $k = 144$ features into a 2D array and then put this array in the appropriate position within the 5×5 pooled array of features. At this point, we will have a C feature arrays of size 60×60, which we could reshape to get a big matrix of size $C \times 3600$ as our new learning database.

[3] The same list can be found in this link for checking the steps.

7. Divide each new pattern into *overlapping clusters* of size 6×6, with *stride* of 2, and apply our learning algorithm to these clusters.

    - Before applying the learning algorithm, one might think about quantizing the patterns and apply the learning algorithm either on the quantized values or on the quantization indices.

8. In parallel, train a classifier on the features returned by step 6.

9. For the classification phase, once give the **noisy** input patterns o the classifier and report its performance. Afterwards, first give the noisy patterns to the dual learned matrices for denoising and then give the output of this algorithm to the classifier. If the accuracy is changed then we have been successful.

Note that one can apply the learning algorithm directly at step 4 of the above procedures. We can think of two different ways of doing this:

- Either think of each of the k 2D arrays as separate images and then consider overlapping clusters within each image,

- Or put all of the $144 \times 22 \times 22$ features in a big array ad apply random clustering methods to determine the overlapping clusters for the purpose of learning the dual vectors.

However, we choose to apply the learning algorithm to the result of pooled features since we will have fewer parameters to learn.

Result: mixture of Failure and Sucess: This approach has been successful for memorization purposes in the sense that the learning algorithm was able to learn both the quantized version of the dataset as well as quantization indices. However, it was a failure to some extent for classification purposes in the sense that once the dual learning algorithm is trained over the training dataset, it can not distinuish patterns from the test dataset for the same class. In other words, patetrns from the test dataset of the same class yield non-zero output at the end of the dual learned connectivity matrices.[4]

Note that the above result still does not rule out the application of the discussed approach for classification purposes because the recall procedure might still become handy during the classification of the patterns in the test dataset if they can be mapped to a pattern from the training set. In other words, the denoising algorithm might think of patterns in the training dataset and try to denoise them. If this turns out to be successful, then we can think about using this technique for classification.

4 Conclusions and future works

For the upcoming weeks, we will continue working on the last approach discussed above. In particular, we try to see if we could improve classification results in noisy environments using our technique.

[4] More detailed summary of the success status of different learning approaches can be found here.

Another related topic which we will pursue in parallel would be to consider the above technique as an associative memory memory for images. More specifically, we focus on memorizing pattern features instead of all the details within a pattern, like an image. This way, we might be able to show that using our technique and these real world patterns, we can accomplish large pattern retrieval capacities.

Finally, another avenue which we will hopefully explore further in future would be to consider our neural scheme as a mechanism to address neurons' internal noise, rather than handling external input noise to the nervous system. We might be able to show that the whole subspace mechanism may provide a nice tool for reliable computation in noisy environments.

References

[1] J. Ngiam, P. Koh, Z. Chen, S. Bhaskar, A.Y. Ng., *Sparse filtering*, NIPS 2011.

[2] B. Olshausen, D. Field, *Emergence of simple-cell receptive eld properties by learning a sparse code for natural images*. Nature, 199

[3] B. A. Olshausen BA, D. J. Field, *Sparse coding with an overcomplete basis set: a strategy employed by V1?*, Vision Research, Vol. 37, 1997, pp. 3311-3325.

[4] S. Ganguli, H. Sompolinsky, *Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis*, Annual Review of Neuroscience, Vol. 35, 2012, pp. 485-508.