Progress Report
1-15 February 2013

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)

1 Summary

In the past two weeks I was mainly busy with organizing some of the stuff we had done in the past (the MATLAB codes, reports, etc.). We also finally finished writing the journal paper for our ITW work and submitted it to IEEE Transaction on Neural Networks and Learning Systems.

We also re-initiated our simulations on the natural images dataset. We are testing some new ideas, some of which is explained in the sequel. We are going to host two master students as well for their semester projects to help us with feature extraction algorithms (both on the theoretical side and implementations).

Finally, I have written the notes for two of the papers I had read (on sparse filtering and non-binary neural networks), which is given towards the end of this report.

2 Simulations over CIFAR-10 dataset

In Section 3, we described the *Sparse Filtering* approach [1]. To reinitate the simulations over a dataset of natural images, we have selected the CIFAR-10 dataset and applied the Sparse Filtering methods to extract features from the dataset.

This time, however, our objective is to apply the learning algorithm to the whole pattern (of length 576) and then for the recall algorithm, we divide the learned network into clusters if necessary.

However, before considering the *Associative Memory* application, we will consider using our algorithm as a ***Classifier***. To this end, we performed the following steps:

1. Apply *Sparse Filtering* to the gray-scale version of the CIFAR-10 dataset. We extracted $24 \times 24 = 576$ features in the second layer.

2. Normalize the extracted features to have a norm equal to 1.

3. Quantizing the dataset by applying a threshold function: we put any values below 0.05 in the normalized dataset to zero. Values above 0.05 and below -0.05 are set to $+1$ and -1, respectively.

4. Apply the learning algorithm proposed in our journal paper [4] to the quantized dataset: We learn one matrix W^i for each class i in the dataset we are interested in.

5. Once finished, use the patterns in the "test dataset" and classify based on the projection of the patterns upon each of $W^{(i)}$'s.

More specifically, in order to classify, once we have finished learning the matrices $W(i)$, we pick a pattern from the "test dataset" of class, say, j. Then, for each pattern in the test dataset, we calculate its projection over each $W^{(i)}$ and find the one with minimum projection to classify the pattern into that class. If we had $i = j$, we declare success and, otherwise, failure.

Learning results: So so. So far, the learning algorithm has finished for 5 of the 10 clusters in the CIFAR dataset. On average, for each class we have learned around 150 constraints, which in fact is a rather good result. Another positive point is that the learning speed is fast (note that this results couldn't be achieved on the pixel-level images).

However, on the down side, there are still a lot of pattern neurons that are not connected to any constraint node at all, which is NOT desirable for the associative memory application.

We should try different values of the sparsity threshold to see if this situation can be dealt with.

2.1 Modifying learning algorithm to deal with maximum degree

A very important issue during the learning algorithm is to limit the degree of each pattern neuron to make sure that there is no subset of pattern neurons that are connected to all of the constraint neurons. So far, one approach was to measure the degree of each pattern neuron at the beginning of the procedure for learning a new constraint and only pick those pattern neurons that have a degree less than some $d_{\max}$. However, this approach usually results in failure after learning several constraints since the new constraint should only depend on a s limited set of pattern neurons, which might not be possible in general.

Recently, we have come up with a different approach: instead of strictly enforcing the $d_{\max}$ upper limit, introduce it as a penalty during the optimization problem so that the neurons themselves decide not to get involved in any constraint when their degree becomes larger than $d_{\max}$. To this end, there is yet much to be done, but at the moment we are considering the following penalty term:

$$\sum_{j=1}^{N} |\text{sign}(w_j)|(1 + tanh(\tau(d_j - d_{\max}))), \tag{1}$$

where N is the number of neurons, w is the constraint to be learned, d_j is the current degree of node j and τ is a rather large constant. If we furhter approximate $|\text{sign}(w_j)|$ by $\tanh(\sigma w_j^2)$, with σ being a large positive constant, this choice results in the following update equation for the weight vector

$$\begin{aligned} w_j(t+1) &= w_j(t) - \alpha_t y(t) \left(x - \frac{y(t) w_j(t)}{\|w(t)\|^2} \right) \\ &- \beta_t \left(1 - \left(\tanh(\sigma w_j(t)^2) \right)^2 \right) \\ &- \lambda_t \left(1 - \left(\tanh(\sigma w_j(t)^2) \right)^2 \right) ((1 + \tanh(\tau(d_j - d_{\max}))), \end{aligned} \tag{2}$$

where x is the current pattern from the learning dataset, $y(t) = x \cdot w(t)$ is the projection of x over $w(t)$ and α_t, β_t and λ_t are update coefficients.

This modification seems to work much better than our previous approach to enforce the constraint strictly.

Classification results: So so. On the positive side, the algorithm is capable of correctly classifying the patterns it has already seen, i.e. in the "training dataset", with probability very close to 1. This is good news because previously, even this couldn't be achieved.

However, as for the "unseen" patterns in the test dataset, the performance drops to 40% or even less. But note that so far, we have only prepared two test datasets and one of them should be re-processed. So this finding is still not rigorous enough and had to be verified in the coming weeks.

3 Notes on the paper "Sparse Filtering"

In [1] the authors propose an efficient unsupervised way of extracting features from the patterns. Currently, there are many unsupervised feature extraction methods but, generally speaking, they are not very fast and they have a lot of hyper-parameters to tune. In contrast, the proposed model here, called *Sparse Filtering*, has only one tunable hyper-parameter: the number of extracted features.

The main idea is to select sparse features among all possible features explaining a given pattern. We have sparsity in two different aspects:

1. Sparsity of features for each pattern: features should be sparse themselves (this is called *population sparsity*).

2. Sparsity of features across patterns: basically, we require features to be such that we can easily distinguish patterns [This is equivalent to having a large minimum distance between features], which can be achieved by making each feature to only be active for a few patterns. This property is called *lifetime sparsity*.

Furthermore, the authors require features to have uniform activity. In other words, there should not be a single feature that shows higher overall activity than the others. For instance, PCA does not have this property as the coefficients corresponding to the largest eigenvalues are always active. This property is called high dispersal.

The authors propose an efficient way of extracting sparse features which is fast and scalable to high-dimensional data. More specifically, suppose we have a linear feature extraction which returns a feature vector $f^{(\mu)}$ for any pattern $x^{(\mu)}$, i.e.

$$f^{(\mu)} = W \cdot x^{(\mu)} \tag{3}$$

Now suppose we have put all the features into a big matrix f in which rows represent features and columns represent the patterns. Then, a *Sparse Filter* can be obtained by

1. first normalizing the rows to get the matrix $\tilde{f}$, i.e. we normalize features to have an identical activity first.

2. secondly, normalizing each column of $\tilde{f}$ to get the matrix $\hat{f}$, i.e. we normalize features for each pattern.

3. and finally, minimizing the ℓ_1-norm of the corresponding matrix, i.e. solving the following equation

$$\min \sum_{\mu=1}^{M} \|\hat{f}^{(\mu)}\|_1 = \sum_{\mu=1}^{M} \left\| \frac{\tilde{f}^{(\mu)}}{\|\tilde{f}^{(\mu)}\|_2} \right\|_1 , \tag{4}$$

   where M is the total number of patterns.

The authors mention that their result show that optimization for population sparsity and enforcing high dispersal result in lifetime sparsity as well.

The experimental results show that their approach is competitive with state-of-the-art methods. In the experiments, to solve the optimization problem (4), they use off-the-shelf L-BFGS package [2].

4 Notes on the paper "Non-binary Neural Networks"

In [3] the authors introduce a novel neural architecture that is based on non-binary neurons to store a set of *random* patterns. This model can be considered as an extension to the famous binary model in which neurons can have values of $0/1$ or $+1/-1$.

The authors propose a simple thresholding mechanism to decide the state of a neuron in each iteration and also extend the delta rule to learn the set of network weights. Their experimental results show that the capacity of a non-binary neural network can be considerably increased compared to a binary case, albeit it is still less than n, the size of the network.

However, it must be mentioned that they do not consider a standard recall phase as they require the learned pattern be the attractor of the neural model. Nevertheless, since if a pattern is not an attractor neither itself nor a corrupted version can be recalled by the network, their result suggest that having a non-binary neural network could increase the storage capacity of neural associative memories.

5 Conclusion and future work

So far, we have realized that our results are not suitable for the associative memory application because there are lots of pattern neurons that are not connected to any constraint nodes. Therefore, we should try different values of the sparsity threshold to see if this situation can be dealt with. This is one of the topics we will investigate in the coming weeks.

As for the classification application, although we have made one step forward by being able to correctly classify already seen patterns, we are still not in a good position to classify unseen patterns in the dataset. This is the topic we should clearly spent some time on in the coming weeks as well.

References

[1] J. Ngiam, P. W. Koh, Z. Chen, S. Bhaskar, A. Y. Ng, *Sparse Filtering*, Proc. NIPS, 2011, pp. 1125–1133.

[2] M. Schmidt, *minFunc*, 2005, Available at *http://www.cs.ubc.ca/ schmidtm/Software/minFunc.html.*

[3] D. Prados, S. Kak, *Non-binary neural networks*, Lecture Notes in Control and Information Sciences, Springer, Vol. 130, 1989, pp. 97–104.

[4] A. H. Salavati, R. K. Kumar, A. Shokrollahi, *A non-binary associative memory with exponential pattern retrieval capacity and iterative learning*, Submitted to the IEEE Trans. Neur. Net. and Learn. Sys., Feb. 2013, Preprint available at `http://arxiv.org/abs/1302.1156`