Progress Report
8-16 August 2011

Raj K. Kumar, Amir Hesam Salavati
E-mail: raj.kumar@epfl.ch, hesam.salavati@epfl.ch

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

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

August 29, 2011

1 Summary

In the past week, we continued reading some papers to help us find a way to solve the problem of learning the constraint matrix from a set of given patterns. Given that our approach on enforcing constraints is very similar to higher order neural networks, we read a bit about these type of neural networks. Interestingly, it seems that one can use higher-order neurons to learn the constraint matrix. The only major drawback is that the this approach has a very high computational cost.

Another interesting subject close to our interests is the problem of learning subspaces from sample data. In this problem, we are given a bunch of patterns that belong to a subspace. We are asked to learn the basis vectors using a neural network from these patterns. As we will see, this problem is closely related to our problem of interest in enforcing constraints to neural patterns. However, ensuring sparse connectivity matrix still remains a big issue.

Finally, as mentioned in previous reports, expander-based compressive sensing approaches are very similar to the method suggested in our ITW paper [6]. In both cases, expander graphs are used to recover sparse signals from linear measurements. The only difference is that we have more restrictions on the algorithm in neural networks as it must be simple enough to be implemented neurally. Because of these similarities, we have been also reading a few papers on compressive sensing methods and the required restrictions on the measurement matrix in order to ensure the correct recovery.

2 The subspace learning problem

In [1] the authors address the problem of classification of input patterns into classes, where each class is identified by a subspace. More specifically, the problem is that given a set of patterns, how we can identify classes as subspaces among the set of given data. In brief, we have K classes $\mathcal{C}^1, \dots, \mathcal{C}^K$. Class i has p_i basis vectors and these vectors are orthonormal but the basis vectors of different classes are not necessarily orthogonal.

Two schemes are overviewed by the authors, namely, Learning Subspace Method (LSM) [2] and Averaged Learning Subspace Method (ALSM) [3], which was previously introduced by the authors themselves. Both approaches belong to supervised learning.

LSM employs a similarity score for each of the given pattern in the training phase to adjust and rotate subspace basis vectors such that the misclassified patterns are rejected and correctly classified patterns are strengthened in that class. The main idea to find the classes is that, after starting with a set of classes, we project training patterns over the classes. In each iteration, if a pattern x is classified in the wrong class, we*rotate* the class subspace by multiplying the basis vectors by a correcting matrix, *which reduces the projection of the vector x in the wrong classes and increases the projection in the right class.*

ALSM is based on using conditional correlation matrices, that is a matrix that compute correlations between misclassified patterns. Main ideas introduced in ALSM is to use *conditional correlation matrices*, that is a matrix that compute correlations between *misclassified* patterns. In other words, the following conditional correlation matrix calculates the correlation between the patterns that a wrongly classified in class $\mathcal{C}^i$ and patterns that belong to class $\mathcal{C}^i$ and are classified in other classes:

$$\mathcal{S}^{(i,j)} = \sum_x \{xx^T | x \in \mathcal{C}^i, x \mapsto \mathcal{C}^j\} \tag{1}$$

where $\mapsto$ denotes "gets classified into". These matrices are then used to do exactly the same as what LSM does, i.e. rejecting the misclassified patterns and strengthening correctly classified patterns.

The mentioned approaches are applied to some large databases in order to investigate the overall performance.

3 Higher order associative memories

In [4] the authors consider extending the concept of Hopfield associative memory to neural networks that utilize higher-order neurons. The state of these higher-order neurons not only depend on the state of their neighbors, but also on their second and higher order correlations. The authors claim the assumption of having higher-order neurons is biologically meaningful as the brain is highly packed with neurons in close vicinities of each others and in such conditions, neighboring neurons can affect the the synaptic weight of other connections in addition of that of their own.

More specifically, a network of higher-order neurons, or a *multiconnected network* as called in this paper, is a neural network in which the weight w_{ij} between neurons i and j may also depend of the state of neuron k, denoted by s_k. Such dependecies is captured by

$$w_{ij} = w_{ij} + w_{ijk}s_k$$

As a result, the simple update rule of normal neural networks will become equivalent to that of higher-order neurons:

$$s_i = f(\theta + \sum_j w_{ij}s_j + \sum_{j,k} w_{ijk}s_j s_k + \dots) \tag{2}$$

where $f(.)$ is non-linear function and θ is the firing threshold.

The overall problem is the same as that of Hopfield associative memory: we have M binary patterns of length N, denoted by $X^\mu = \{x_i^\mu\}$, which we would like to memorize using multiconnected neural networks. The update rule is given by equation (2). Here, the authors have assumed that the absolute value of the higher-order connection weights scaled with second-order ones, i.e.

$$|w_{ijk}| \simeq \gamma |w_{ij}| \tag{3}$$

With the assumption of $\theta = 0$, the learning rule for the weights is an extended version of the Hopfield learning rule:

$$\begin{aligned} w_{ij} &= < s_i^\mu s_j^\mu > \\ w_{ijk} &= \gamma < s_i^\mu s_j^\mu s_k^\mu > \\ \dots & \\ w_{ijk\dots p} &= \gamma^{p-2} < s_i^\mu s_j^\mu s_k^\mu \dots s_p^\mu > \end{aligned} \tag{4}$$

where $< . >$ denotes averaging over all patterns in the training set.

Using an analysis similar to that of Hopfield, the authors breaks down the linear input sum of each neuron into a desired term and an interference term. Assuming the patterns to be stochastically independent, the interference term will become a Gaussian random variable as a consequence of central limit theorem. Therefore, the stability condition reduces to the probability of a Gaussian random variable being smaller than the desired term. Limiting this probability to be small will give us the maximum number of patterns than can be stored.

Using such a model, the authors show that the storage capacity of Hopfield networks can be improved. More specifically, the authors obtain storage capacities that are polynomial in N with exponents linear in p, the order of the network. In other words, $M = O(N^{p-2})$. However, note that the number of stored patterns is still proportional to the number of synapses. For a network of size N, we can get more synapses if we consider higher-order correlations.

4 Expander-based compressive sensing

In [5] the authors address the recovery of a sparse non-negative signal using a sparse measurement matrix. More specifically, the problem is to find a good sparse measurement matrix A that allows the recovery of a k-sparse non-negative signal x from the measurements $y = Ax$. Here, there are two main differences from the traditional compressive sensing problem, namely, Non-zero elements of x are assumed to be positive and A is sparse.

The main technique used to find such matrices is to focus on the complete rank of a matrix, denoted by $\mathcal{H}(A)$, that is the maximum integer r_0 such that any subset of r_0 columns of A are linearly independent. The authors find matrices that their complete rank is linear in n, the size of input signal. Having such matrices is equivalent to the *possibility* of recovering signals that have more non-zero elements. The main idea to find such matrices is that if a bipartite graph with 0/1 adjacency matrix A satisfies Hall's property, then by perturbing non-zero elements of A one can obtain another matrix $\tilde{A}$ such that $\mathcal{H}(\tilde{A}) \geq r_0$. So if we manage to find bipartite expanders with the complete rank r_0 proportional to n, then accodring to we will be able to recover a $\lceil r_0/d - 1 \rceil$-sparse signal using compressive sensing.

The authors show that ***there exists*** such expanders and give bounds on the properties of such graphs, namely, the left degree and the number of right nodes, such that the graph is an expander with high probability. They also come up with bounds on the recovery thresholds, i.e. the degree of sparsity that can be recovered using sparse matrices, and propose a fast algorithm to perform the recovery task. In brief, they find out that if a left d-regular matrix A is used in compressive sensing algorithms to recover any k-sparse signal, then it must be a $(k, 1 - 1/d)$ expander.

The authors also show that using expander graphs with slight perturbations results in better recovery thresholds. Furthermore, the degree of expansion required in their proposed method is much less that that of similar approaches. In fact the authors show that error correction is possible using an expander with coefficient $1 - \epsilon > 1/2$ (as compared to $1 - \epsilon > 3/4$). They show that the minimal expansion used in the constructions is actually necessary for any matrix that works for compressive sensing.

While it is shown that such *good* expander graphs exist, they do not discuss how to construct one. However, the error rate performance of the recovering mechanism show that random bipartite graphs with perturbation can also be used to achieve desirable recovery rates.

5 Conclusions and future works

We have read a few papers to give us more ideas about how to solve the problem of learning constraints from the set of training patterns. Based on the mentioned papers, there are some promising ideas which might worth pursuing further:

- The idea of using higher-orger neurons as mentioned in [4]. The advantage is that it works. The disadvantage is the high computational complexity of such approaches.
- Using ideas mentioned in [1] in order to find the subspace basis vectors. Instead of using the dual space specified by the parity check matrix H, one might be able to use $I - GG^{\dagger}$ matrix and to find the basis vectors in this subspace. They basically do the same job. The advantage is that it will probably work as well. The disadvantage would be the sparsity.
- Recovery requirements of [5] will help us analyze our proposed approach analytically. But on the down side, no specific construction method is given in their paper to help us find suitable expander graphs with the desired properties.

References

[1] E. Oja, T. Kohonen, *The subspace learning algorithm as a formalism for pattern recognition and neural networks*, Neural Networks, Vol. 1, 1988, pp. 277-284.

[2] T. Kohonen,M. J. Riittinen, E. Reuhkala, S. Haltsonen, *A 1000 word recognition system based on the learning subspace method and redundant hash addressing*, Proc. Pattern Recognition, 1980, pp. 168-166.

[3] E. Oja, M. Kuusela, *The ALSM algorithm - an improved subspace method of classification*, Pattern Rec., Vol. 16, No. 4, 1983, pp. 421-427.

[4] P. Peretto, J. J. Niez, *Long term memory storage capacity of multiconnected neural networks*, Biological Cybernetics, Vol. 54, No. 1, 1986, pp. 53-63.

[5] M. A. Khajehnejad, A. G. Dimakis, W. Xu, B. Hassibi, *Sparse recovery of nonnegative signals with minimal expansion*, IEEE Transactions on Signal Processing, Vol. 59, No. 1, 2011, pp. 196-208.

[6] R. Kumar, A. H. Salavati, M. A. Shokrollahi, *Exponential pattern retrieval capacity with non-binary associative memory*, Inf. Theory Workshop, 2011.