Progress Report
18-28 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 10 days, we continued our works on neural networks built from unreliable components and constructing feature extracting methods suitable for our learning algorithm. Regarding the former case, we have studied some papers on the analysis of LDPC decoders built from faulty hardware as well as some natural phenomenon that is similar to our observation that some amount of noise actually improve the performance of the system in correcting external errors.

As for the feature extraction methods, we started reading about Non-Linear PCA (NLPCA) methods which extract non-linear structures from input data. We then applied some ideas borrowed from such approach to our learning algorithm to see if it improves the learning results (which did). Furthermore, we tested different approach to perform clustering for the recall phase the details of which can be found in the sequel.

I have also met with the two students conducting master semester projects and we are working on implementing various feature extraction methods as well as possibly building one of our own.

2 Fault tolerant neural networks

In the last 10 days, we got back to the neural networks built from unreliable components which we proposed in [1]. We performed some extra simulations to further investigate the phenomenon which we had seen previously, namely, the fact that some internal noise actually improves the performance of the network in dealing with external noise. To this end, we considered the effect of internal noise on the number of iterations it takes the algorithm to correct external noise. The idea was that for small amounts of external noise, where both noiseless and noisy neural networks can correct external errors, the noiseless decoder should spend less amount of time to accomplish the task. Our simulation results, given in [2], shows this trend to some extent. However, there have been cases that introduction of internal noise also reduces the number of iterations to perform the decoding task. Therefore, we are currently conducting more extensive simulations to first verify this phenomenon. Then we should work on theoretical aspects of the problems and see if we can prove this phenomenon analytically.

We have also spend some time reading two papers related to our works. In [3] the authors address the phenomenon known as **Stochastic Resonance** (SR) in various physical systems. SR refers to situations in which a fair amount of noise could actually improve the system's performance in one way or another. A good example is enhancing the ability of sensory neurons in picking up extremely weak signals. In this sense, the phenomenon is very similar to what we see so it would be interesting to further explore their connections.

We also read another paper on noisy Gallgher B decoders [4] in which the authors investigate the performance of non-binary Gallagher B decoders built from unreliable hardware. They are interested in finding out how the final Symbol Err Rate (SER) depends on the internal nodes at pattern and variable nodes, respectively. To this end, they use analytical tools, like Density Evolution, as well as simulations and hardware emulation. They show that the final SER is more sensitive to noise at variable nodes' side and derive the theoretical expression for the SER as a function of internal noise parameters of the model.

Both [3] and [4] have some nice figures and measures to investigate the performance of any algorithm related to this field, which we will use to evaluate our algorithm.Furthermore, we are also conducting experiments to investigate connections of our work to the experiments performed

on the effect of heat on the performance of human brain [5].

3 Nonlinear PCA

We have also spent some time on studying Non-Linear PCA (NLPCA) approaches. These approaches tend to extend PCA to extract nonlinear structures from input data.

In [7] the author considers theoretical aspects of the Non-Linear Principle Component Analysis (NLPCA) algorithm that has been proposed earlier. Their goal is to provide theoretical analysis for a phenomenon observed experimentally, namely, the fact that in some cases the non-linearity helps the algorithm find independent components in the data. From another viewpoint, this can be interpreted as the capability of the algorithm to do signal separation.

The author provides sufficient conditions on the non-linearity function and the probability distribution of the data so that the convergence of the algorithm to independent components is guaranteed. More specifically, he shows that for sub-Gaussian distributions, one can use a tanh(.) activation function while for super-Gaussian distributions, one should use polynomial activation functions.

In [6] the authors propose a novel method to extract non-linear data structures from data. The proposed approach is based on using neural networks to implement *Exploratory Projection Pursuit*, or EPP. EPP is closely related to Non-Linear PCA (NLPCA) in the sense that they both aim to find "interesting" directions in the data. From that regard, these approaches are similar to PCA as well. However, in PCA the interesting directions correspond to the directions with maximum variance while in EPP and NLPCA, the directions tend to maximize higher moments or even an arbitrary (but suitably) defined "index" which identifies interesting trends in the data.

Since a normal function is rather *uninteresting*, the authors define appropriate indices to determine how interesting each direction is by measuring its deviation from the normal distribution. The authors also propose a neural network capable of performing EPP. They discuss applications of various projection pursuit indices and use numerical simulations to investigate the performance of the proposed method in practice.

The above approach has been extended in [8] where the authors propose a novel method for performing Non-Linear Principle Component Analysis (NLPCA). One of the main differences between the proposed approach and the previous work lies in the fact that the derive principle components here have the an ordering similar to what the standard PCA returns, i.e. the first principle component is in the direction of the maximum variance, then follows the second one and so on. The authors have also prepared a nice MATLAB package to perform the NLPCA algorithm [9].

4 Simulations over CIFAR-10 dataset

We have also continued performing simulations of the CIFAR-10 dataset. sThis time, we have tested two different methods to investigate if there are any structures in the data: applying NLPCA and calculating minimum distance.

4.1 Applying NLPCA to CIFAR-10 dataset

The NLPCA package [9] helps us visualize the first three principle components to see if there are any (nonlinear) structure within the data. We have applied this toolbox to the class 1 of the

CIFAR-10 dataset before and after applying *Sparse Filtering*.

Learning results: So so. In both cases, we did not witness meaningful structures, at least within the first three nonlinear principle components (unlike the figures given in [8]). However, we will continue doing research on NLPCA techniques.

4.2 Evaluating the minimum distance of CIFAR database

We have also calculated the minimum distance between the patterns in a given class, say class 1, of the CIFAR database to see if the reason behind failure of our "scheduling" recall algorithm (see Section 6) is because of low minimum distance between patterns.

Learning results: failure It turns out that the minimum distance between the patterns is reasonable. And the reason behind the failure of that algorithm is that **one or two flips gives us a pattern lying in the subspace of the patterns**. Unfortunately, this pattern is not necessarily the pattern of interest. So we must find a representation such that two columns of the weight matrix do not add up to zero.

4.3 Learning with non-liearity and thresholds

Another important idea that we tested was to consider using NLMCA (Non-Linear Minor Component Analysis). To this end, we have used a $\tanh(W \cdot x)$ activation function for constraint neurons (instead of $y = W \cdot x$). Furthermore, we have included an update threshold τ for the neurons which *is learned* as well. Thus, the final update threshold is

$$y = \tanh(W \cdot x + \tau) \tag{1}$$

Please note that since we are interested in minimizing y, on paper we should find the weight vectors W that $W \cdot x = -\tau$, which is still linear but the $\tanh(.)$ function may make the convergence a bit more smooth. Furthermore, the introduction of the not-necessarily-zero threshold τ make the algorithm more general and enables us to find constraints that we couldn't have found with a zero update threshold.

Learning results: So so (mostly success!). We have tested the algorithm over the *Sparse Filtered* version of the class 1 of the CIFAR-10 database. We had 576 pattern neurons each values 0 or 1 (after quantization). We then generated random clusters and told the algorithm to learn 4-5 constraints in each clusters. We then built a global weight matrix W_g that captured the connectivity of the whole graph. The **learning phase was a success** in the sense that

1. We learned around 2200 constraints in total.
2. The minimum degree of a pattern neuron was 11.
3. The maximum degree was around 60.

So overall, the learning matrix was nice. However, the **recall phase was a bit more tricky** because we now have to divide the graph W_g into sub-graphs by picking a number of constraint neurons to put in each clusters. By doing so, we make the minimum and average degree of pattern

neurons in each cluster very small, which in turn create difficulties for the recall algorithm. To this end, we could either go back to the clustered version of learning algorithm (i.e. do the clustering before the learning and force the learning algorithm to learn $20-30$ constraints in each cluster) or find a way to do the clustering more intelligently.

With respect to the latter option, we are currently trying to form clusters in which

1. No two columns in the weight matrix have the same non-zero positions. This is equivalent to making sure that no two pattern neurons have exactly the same neighborhood in constraints side.

2. The number of pattern neurons with (local) degree at least two is larger than a minimum threshold.

3. Each pattern neuron is member of at least two clusters in which it has degree two or more.

Furthermore, we slightly change the recall algorithm to only allow pattern neurons with (local) degree two or more to update their states. Thus, the last two options are to make sure that the noise in pattern neurons get corrected.

We could easily find 250 clusters with such properties. However, the pattern error rates are yet to be in the acceptable region (either because the errors do not get corrected or the algorithm is very slow in correcting them). We will continue working on them to see what happens.

4.4 Classification between 5 classes of CIFAR algorithm using MCA

Another approach that we tested was to see if our algorithm is capable of classifying patterns in the first 5 classes of the CIFAR-10 database. To this end, we applied the standard learning algorithm to the training dataset for each class according to the procedure discussed in Section 4.3. The simulation parameters were $\alpha_0 = 0.95$, $\beta_0 = 0.75$ and $\theta_0 = 0.008$ (or 0.011).

Once finished, we had 5 matrices $W^{(\ell)}$, $\ell = 1, \ldots, 5$ with 576 columns and 100-200 rows (constraints). We then used these matrices to classify the patterns with the classification metric $argmin_\ell \mathbb{E}(W^{(\ell)} \cdot x)$. We performed the following steps:

1. Pick a pattern randomly from the training dataset of class c and see if it is classified correctly.

2. Pick a pattern randomly from the test dataset of class c and see if it is classified correctly.

Classification result for step 1: Success. This step was successful almost 100% of the times. In other words, the system was able to classify the pattern it had seen (learned) correctly in almost all instances. This means that the subspace that patterns in class c form is substantially different from that of other classes.

This is good news because even this step could not be done previously.

Classification result for step 2: So so (mostly failed!). However, the overall success rate for this step was around 45% for the best case and around 0% for the worst case (which happened to be for class 1). This means that the learned dual basis for the training dataset of class c is not necessarily orthogonal to the patterns in test dataset of class c. Furthermore, the projection of patterns in the test dataset of class c is on average smaller on the learned matrix for some other class c'.

We calculated the minimum distance between a pattern in the test dataset of class c to the patterns in the training dataset of class c and other classes to see if the misclassification is because of our algorithm or possibly for some other reason. Interestingly, we found out that the minimum distance between a pattern in test dataset of class $c = 1$ happens with respect to a pattern in some other class $c' \neq 1$. This is some serious problem and we should think about how to deal with it.

5 Clustering based on pattern neurons

So far, our clustering method has been based on putting some constraint neurons and all their neighbors on the pattern neurons side in a cluster. This makes sense because a cluster should be self-contained in the sense that even if it is cut from the other neurons, the constraints should be satisfied when there is no external noise.

Or should it? We could for instance think of performing the clustering algorithm based on pattern neurons, i.e. select a number pattern neurons and put them in a cluster together with all the constraints they are connected to. Then, at any instance of time in the global recall algorithm, only the pattern neurons in a given cluster are allowed to update their value. However, for this approach to work, the other pattern neurons should freeze at their current state (which is in contrast to the previous clustering method where the other pattern neurons sent 0 values).

Results: failure. This approach failed for a very simple reason: even when there is no noise in the pattern neurons corresponding to a given cluster, there could be a lot of noise in the "frozen" pattern neurons outside of the cluster. So the constraints in the cluster will not be satisfied.

6 Scheduling-based recall algorithm

We also tested another idea for the recall phase to alleviate the difficulties we had encountered in clustering the graph. To this end, we devise a scheduling system in which at any given instance of time, only a single pattern neuron can update its value. Therefore, we will not have any clustering (or we could think of this method as having clusters of size 1). We considered two different scheduling methods:

1. One in which pattern neurons update their value sequentially one by one.
2. One in which pattern neurons update their value sequentially one by one but we start from the patterns with larger degree.

Results: failure. This approach failed not because it did not converge to a pattern that is orthogonal to the learned matrix, but because it converged to a wrong pattern. Interestingly, **one or two flips lead the algorithm to a pattern orthogonal to the learned matrix** which was not the original pattern.

7 Conclusion and future work

In the upcoming weeks, we should work on theoretical aspect of the faulty neural networks to improve the upper bound on the probability of error which is currently not very tight, specially when the amount of internal noise is not very small.

For the clustering techniques, we will test various new ideas or current ideas on some of the weight matrices recently learned with new methods.

We will also test one of our old ideas (i.e. to force the network to learn tens of constraints in each cluster of rather small size) with the new learning algorithm we have to see if it improves the learning results.

In both cases above, the learned matrices will also be tested for classification purposes.

Finally, Mr. Amin Karbasi mentioned a new idea based on learning "union of subspaces" instead of just a single subspace. We will hopefully study this approach in more details later on.

References

[1] A. Karbasi, A. H. Salavati, A. Shokrollahi, L. R. Varshney, *Neural networks built from unreliable components*, Submitted to ISIT 2013.

[2] A. Karbasi, A. H. Salavati, A. Shokrollahi, and L. R. Varshney, Neural networks built from unreliable components, arXiv, Jan. 2013.

[3] K. Wiesenfeld, F. Moss, *Stochastic resonance and the benefits of noise: from ice ages to crayfish and SQUIDs*, Nature Vol. 373, 195, pp. 33–36

[4] S. M. S. Tabatabaei Yazdi, H. Cho, L. Dolecek, *Gallager B Decoder on Noisy Hardware*, To appear in IEEE Trans. Comm.

[5] N. H. Mackworth, *Effects of heat on wireless telegraphy operators hearing and recording morse messages*, British Journal of Industrial Medicine, Vol. 3, No. 3, 1946, pp. 143–158

[6] C Fyfe, R. Baddeley, *Non-linear data structure extraction using simple Hebbian networks*, Biol. Cyber., Vol. 72, No. 6, 1995, pp. 533–541.

[7] E. Oja, *The nonlinear PCA learning rule in independent component analysis*, Neurocomputing, Vol. 17, No. 1, 1997, pp. 25–45.

[8] M. Scholz, M. Fraunholz, J. Selbig, *Nonlinear principal component analysis: neural network models and applications*, Principal manifolds for data visualization and dimension reduction, Lecture notes in computational science and enginee, Vol. 58, 2008, pp 44–67.

[9] M. Scholz, *Nonlinear PCA toolbox for MATLAB*, Available at `http://www.nlpca.org/matlab.html`.