Progress Report
16-30 November 2011

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)

January 10, 2012

1 Summary

In the last two weeks, I continued working on compressive sensing methods and their application in neural associative memories. I also worked on an iterative algorithm to learn a sparse vector that is orthogonal to the patterns that belong to a subspace. This algorithm corresponds to the learning phase of our neural network proposed in [5] which memorizes patterns that belong to a subspace. I tested some of the approaches for sparse estimation proposed in the "Approximate Message Passing (AMP) algorithm by Donoho et

Finally, I explored a couple of new ideas about a different architecture to memorize neural patterns.

In what follows, one can find the details of proposed learning algorithms as well as the sketch of the new ideas. In addition, I have brought the important points of the papers on compressed sensing that are most relevant to our work. Specially, one can find some useful remarks on the paper "Message-Passing Algorithms for Compressed Senesing" which proposes an elegant iterative algorithm for recovering sparse signals.

2 Learning algorithm and its proof of convergence

2.1 Overview of the proposed algorithm

The main idea of identifying the null basis is adopted from [1] with some modifications in order to make it realizable by our double layer network in a distributed fashion. The overall learning algorithm for one constraint node y with our modifications is given by equation (1):

$$y(\mu, t) = x^{\mu}.w(t) \tag{1a}$$

$$w(t+1) = w(t) - \alpha_t y(\mu, t)\left(x^{\mu} - \frac{y(\mu, t)w(t)}{\|w(t)\|^2}\right) - \lambda_t f(w(t)) \tag{1b}$$

In the above equations, t is the iteration index, x^{μ} and w are vecotrs of length n, x^{μ} is the data vector μ, α_t and λ_t are small positive constants. Finally, $f(w) : \mathcal{R}^n \to \mathcal{R}^n$ is the gradient of the penalty term for non-sparse solutions.

In words, y is the projection of x^{μ} on the basis vector w. If for a given data vector x^{μ}, y is equal to zero, namely, the data is orthogonal to the weight vector w, then according to equation (1b) the weight vector will not be updated. However, if the data vector x^{μ} has some projection over w^{μ} then the weight vector is updated towards the direction to reduce this projection.

Since we are interested in finding m orthogonal vectors, we have to do the above procedure m times in parallel.

We update λ_t according to the following equation:

$$\lambda_{t+1} = \lambda_t + \gamma(g(w(t)) - q) \tag{2}$$

where γ is a small positive real number and q determines degree of sparsity. For instance, if $g(w) = \|w\|_1$, then q determines the maximum amount of ℓ_1-norm that w can have.

Thus, overall we are using primal-dual methods to solve the following optimization problem:

$$\min \sum_{\mu=1}^{M} \|w.x^{\mu}\| \tag{3a}$$

subject to:

$$g(w) \leq q \tag{3b}$$

where $g(w)$ is a function to enforce sparsity and $f(w) = \nabla g(w)$.

2.2 Convergence analysis

We can rewrite equation (1b) as:

$$w(t+1) = w(t)\left(1 + \alpha(\frac{y(\mu,t)}{\|w(t)\|})^2\right) - \alpha y(\mu,t)x^\mu - \lambda_t f(w(t)) \tag{4}$$

Now multiplying both sides of the above equation with x^μ we will get:

$$y(\mu,t+1) = y(\mu,t)\left(1 + \alpha\left[(\frac{y(\mu,t)}{\|w(t)\|})^2 - \|x^\mu\|^2\right]\right) - \lambda_t x^\mu.f(w(t)) \tag{5}$$

As a result

$$\begin{aligned} |y(\mu,t+1)| &= |y(\mu,t)\left(1 + \alpha\left[(\frac{y(\mu,t)}{\|w(t)\|})^2 - \|x^\mu\|^2\right]\right) - \lambda_t x^\mu.f(w(t))| \\ &\leq |y(\mu,t)\left(1 + \alpha\left[(\frac{y(\mu,t)}{\|w(t)\|})^2 - \|x^\mu\|^2\right]\right)| + \lambda_t|x^\mu.f(w(t))| \end{aligned} \tag{6}$$

Now if we choose the function $f(w(t))$ such that $|f(w(t))| \leq |w(t)|$ and $sign(f(w(t))) = sign(w(t))$ for all values of $w(t)$, then we will have

$$|y(\mu,t+1)| \leq |y(\mu,t)|.|\left(1 + \lambda_t + \alpha\left[(\frac{y(\mu,t)}{\|w(t)\|})^2 - \|x^\mu\|^2\right]\right)| \tag{7}$$

Therefore, in order to have $|y(\mu,t+1)| \leq y(t)$, we must show pick the coefficients such that

$$|1 + \lambda_t + \alpha\left((\frac{y(\mu,t)}{\|w(t)\|})^2 - \|x^\mu\|^2\right)| < 1 \tag{8}$$

or equivalently (assuming $\alpha > 0$):

$$\frac{\lambda_t}{\alpha} < \|x^\mu\|^2 - (\frac{y(\mu,t)}{\|w(t)\|})^2 < \frac{2+\lambda_t}{\alpha} \tag{9}$$

Now noting that $y(\mu,t) = x^\mu.w(t) = \|x^m u\|\|w(t)\|\cos(\theta_{\mu,t})$, we can simplify the above inequality to:

$$\frac{\lambda_t}{\alpha} < \|x^\mu\|^2 \sin^2(\theta_{\mu,t}) < \frac{2+\lambda_t}{\alpha} \tag{10}$$

This last equation gives us the following inequalities for the value of α:

$$\alpha < \frac{2+\lambda_t}{\|x^\mu\|^2 \sin^2(\theta_{\mu,t})},\ \forall\mu \Rightarrow \alpha < \frac{2+\lambda_t}{\max(\|x^\mu\|^2)} \tag{11a}$$

and

$$\alpha > \frac{\lambda_t}{\|x^\mu\|^2 \sin^2(\theta_{\mu,t})},\ \forall\mu \tag{11b}$$

The above equations only prove that the algorithm convegres as long as there is a proper relationship between α and λ in every iteration. However, it does not tell us any thing about the degree of sparsity achieved in the end. Nevertheless, simulation results show that we will achieve some good degree of sparsity in the end. See the results section for more details.

2.3 Some Remarks

There some technical remarks about the approach discussed above:

1. **A Remark on the Sparsity Penalty Function**: Another remark is on the choice of the function $f(w)$. We can choose $g(w)$ such that it mimics ℓ_0-norm of w. We notice that $\|w\|_0 = \sum_{i=1}^{n} |sign(w_i)|$, where $sign(0) = 0$. Therefore, if we choose $g(w(t)) = \sum_{i=1}^{n} \tanh(\sigma_t w_i(t)^2)$, for large values of σ_t the function $\tanh(\sigma_t w_i^2)$ is a good approximation of $|sign(w_i(t))|$. Therefore, $g(w(t))$ approximates $\|w(t)\|_0$. And the approximation gets better as σ_t increases.

   Now, the i^{th} entry of the function $f(w(t)) = \nabla g(w(t))$ will be $f_i(w(t)) = \partial g(w(t))/\partial w_i(t) = 2\sigma_t w_i(t)(1 - \tanh^2(\sigma_t w_i(t)^2))$. This function has the interesting property that for very small values of $w_i(t)$, $f_i(w(t)) \simeq 2\sigma_t w_i(t)$ and for relatively larger values of $w_i(t)$, we get $f_i(w(t)) \simeq 0$. Therefore, by proper choice of λ_t, equation (1b) suppresses small values of$w(t)$ by pushing them towards zero. In other words, this function favors sparser results.

   Interestingly, the above choice for the function $f(w)$ looks very similar to the soft thresholding function (17) by [2] to perform iterative compressed sensing. The authors show that their choice of the sparsity function is very competetive in the sense that one can not get much better results by choosing other thresholding functions. However, one main difference between their work and that of ours is that we enforce the sparsity as a penalty in equation (1b) while they apply the soft thresholding function in equation (17) to the whole w, i.e. if the updated value of w is larger than a threshold, it is left intact while it will be put to zero otherwise.

2. **Proper choice of α and λ**: The above formulas assume that we fix a pattern and iteratively learn it (by increasing the iteration number t). Then, apply another pattern and learn that one (by increasing the pattern index μ). As we have shown above, if α and λ satisfy the set of inequalities (11), then we are pretty sure that $|y(\mu, t+1)| < |y(\mu, t)|$, i.e. the error term for the current pattern μ reduces. However, there is subtlety here as the update along one direction for one pattern may ruin the patterns that we have learned before. So one thing that we can do is to choose α rather small in the hope that update in one direction does not affect the patterns already learnt. However, since according to inequalities (11) α and λ are linked together we might want to enforce a bound on λ as well.

   One idea to avoid ruining other already learnt patterns is that we make sure that the norm of the update in each iteration is less than a small factor of the norm of the weight vector.Namely,

$$\Delta w_t = \|\alpha_t y(\mu, t) \left(x^\mu - \frac{y(\mu, t) w(t)}{\|w(t)\|^2} \right) \|^2 \leq \beta^2 \|w(t)\|^2 \tag{12}$$

   where β is a very small constant. In this way, we slow down the convergence but make sure we converge! Expanding the left hand side of inequality (12), we will get

$$\begin{aligned} \Delta w_t &= \|\alpha_t y(\mu, t) \left(x^\mu - \frac{y(\mu, t) w(t)}{\|w(t)\|^2} \right) \|^2 \\ &= \alpha_t^2 y^2(\mu, t) \left(\|x^\mu\|^2 - \frac{y^2(\mu, t)}{\|w(t)\|^2} \right) \\ &= \alpha_t^2 y^2(\mu, t) \|x^\mu\|^2 \sin^2(\theta_{\mu,t}) \\ &= \alpha_t^2 \|x^\mu\|^4 \|w(t)\|^2 \sin^2(\theta_{\mu,t}) \cos^2(\theta_{\mu,t}) \end{aligned} \tag{13}$$

Therefore, inequality (12) reduces to:

$$\alpha_t \|x^\mu\|^2 |\sin(2\theta_{\mu,t})| \leq 2\beta \tag{14}$$

Since $|\sin(2\theta_{\mu,t})| \leq 1$, if we manage to have $\alpha_t \|x^\mu\|^2 \leq 2\beta$ we will be safe. Therefore, in order to have small enough updates, it is sufficient to have:

$$\alpha_t \leq \frac{2\beta}{\max_\mu(\|x^\mu\|^2)} \tag{15}$$

Combining this equation with (11a), we notice that since $\lambda > 0$ and $\beta \ll 1$, we have $2\beta < 2 + \lambda_t$. Therefore, it is sufficient to choose α_t according to equation (15). However, equation (11b) will cause problems if $\frac{\lambda_t}{\|x^\mu\|^2 \sin^2(\theta_{\mu,t})} > 2\beta$ which we have to address in our next reports.

3. **A Remark about equation 11**: In equation (11b), note that as $\theta_{\mu,t} \simeq 0^\circ$, we need large α's to ensure convergence. However, in such cases, $w(t)$ does not get updated at all because $\Delta w_t \simeq 0$. Therefore, one have to be careful to pick an initial $w(0)$ such that its projection on x^μ is not large, i.e. $\theta_{\mu,0} \gg 0^\circ$. In that case, if we show that $\theta_{\mu,t}$ increases monotonically, we can prove convergence. We can use induction on t to show that the required conitions are satisfied because at $t = 0$ we have $\lambda_t = 0$. Thus, we always can find an α ensuring $|y(\mu, 1)| < |y(\mu, 0)|$, which intuitively translates into $\theta_{\mu,1} > \theta_{\mu,0}$.

4. **Practical choice of** α: Even if we can find a proper α for each pattern in each iteration, one notices that for each μ (and t) we must use a different α to have $\frac{\lambda_t}{\|x^\mu\|^2 \sin^2(\theta_{\mu,t})} < \alpha < \frac{2+\lambda_t}{\|x^\mu\|^2 \sin^2(\theta_{\mu,t})}$. Therefore, it is possible that for some cases we can not have a unique α that works for all patterns. However, simulations show that a relatively small constant α is usually sufficient to ensure convergence.

2.4 Simulation Results

We have simulated the proposed learning algorithm over three different network sizes $n = 90, 300, 600$ with $m = 60, 100, 200$ constraints, respectively. To execute the learning phase, we first generate M patterns that belong to a subpace with dimension $k = n - m$ by multiplyig M *binary* vectors of length k to a sparse randomly generated 0/1 matrix $G \in \mathcal{R}^{k \times n}$. We repeat the learning algorithm given by equation (1) until $y(\mu, t) \leq \epsilon$ for *all patterns* μ, where ϵ is a very small number. For avoiding numerical issues, we also normalized all the patterns. This does not affect the final result though. Furthermore, λ_t was bounded above so that it was always less than 0.5 to ensure that consecutive updates do not affect each other. Finally, in each iteration we put all entries less than 0.001 to zero manually.

Table 2.4 summarizes other simulation parameters.

Table 1: Simulation parameters

Parameter	Maximum Learn Iterations	Number of Learned Patterns	ϵ	σ_t	α_t	γ
Value	5000	1000	10^{-4}	$15t$	$0.8\frac{2+\lambda_t}{\|x^\mu\|^2 - \|y(\mu,t)\|^2/\|w(t)\|^2}$	$\frac{0.005}{n}$

Figure 1 shows the number of iterations executed before convergence is reached for different constraint nodes. We have investigated three different network sizes, i.e. $n = 90, 300, 600$. The corresponding number of constraints were $m = 60, 100, 200$, respectively.

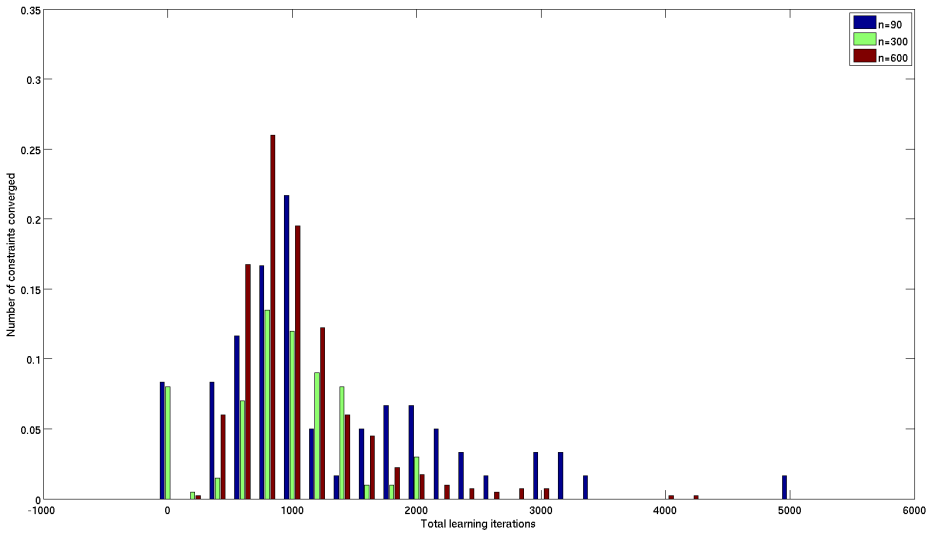

Figure 1: Convergence rate for the specified number of constraints and different values of network sizes: $n = 90, 300, 600$. The corresponding number of constraints were $m = 60, 100, 200$, respectively

Figure 2 illustrates the percentage of variable nodes with the specified sparsity measure defined as $\rho = \kappa/n$, where κ is the number of non-zero elements. From the figure one notices that as n increases, the weight vectors become sparser.

Finally, figure 3 shows the MSE, given by the $\sum_{\mu=1} : My(\mu, t)$ in equation (1a), in each iteration for a sample constraint node for three different values of n. Overall, the MSE is overall decreasing. However, small increases happens due to the addition of the sparsity constraint. The amplitude of fluctuations (and in fact the convergence of the algorithm) depends heavily on the choice of parameters α and λ.

3 Papers

3.1 Message Passing Algroithms for Compressed Sensing

In [2], the authors introduce an iterative algorithm which approximates belief propagation method for compressed sensing. The main advantage of the proposed approach compared to previous message passing methods is its ability to achieve reconstruction performances nearly identical to the optimal given by linear programming methods. The key point behind this success is the addition of a correction term to message passing equations to compensate for approximation. The authors find a one-dimensional recursive formula for the MSE of each iteration of the algorithm and show

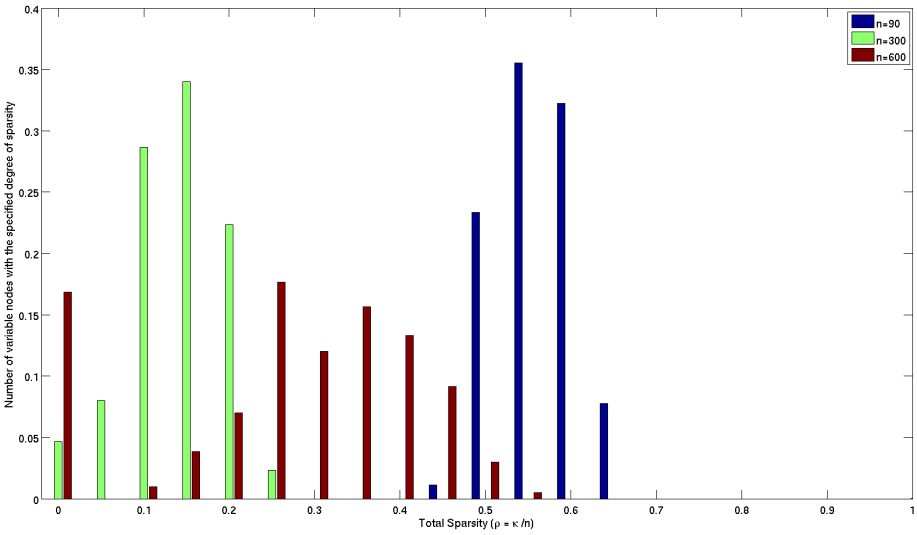

Figure 2: The percentage of variable nodes with the specified sparsity measure and different values of network sizes: $n = 90, 300, 600$. The sparsity measure is defined as $\rho = \kappa/n$, where κ is the number of non-zero elements.

that when the measurement matrix is selected randomly according to the Gaussian distribution, this equation has only one fixed point at 0 when the sparsity-undersampling properties of the system satisfy certain criteria. Furthermore, using analytical approaches and simulations the authors show that this criteria is almost identical to the optimal scenario, found by the LP-based methods. Extensive simulations show that the suggested analysis is also applicable to other ensembles of random matrices. Thus, the Gaussianity assumption of the measurement matrix might be relaxed.

More specifically, the goal is to reconstruct a vector x_0 from a set of linear measurements $y = Ax_0$, where $y \in \mathcal{R}^n$, $A \in \mathcal{R}^{n \times N}$ and $x \in \mathcal{R}^N$. For the purpose of analysis, the entries of A are assumed to be zero-mean iid Gaussian random variables with variance equal to $1/n$. Then, the proposed algorithm is as follows:

$$x(t+1) = \eta_t(A^T z(t) + x(t)) \tag{16a}$$

$$z(t) = y - Ax(t) + \frac{1}{\delta} z(t-1) \left\langle \eta'_{t-1}(A^T z(t-1) + x(t-1)) \right\rangle \tag{16b}$$

where $x(0) = 0$, A^T is the transpose of matrix A, $\eta'(.)$ is the derivative of the soft threshold function, given by equation (17), and $< u >$ for a *vector* u is defined to be $< u >= \frac{1}{m} \sum_{i=1}^{m} u_i$. Furthermore, the function $f = \eta_t(x)$ and its derivative are considered as vector functions, i.e. $f_i = \eta(x_i)$. Finally, δ is usually set to n/N.

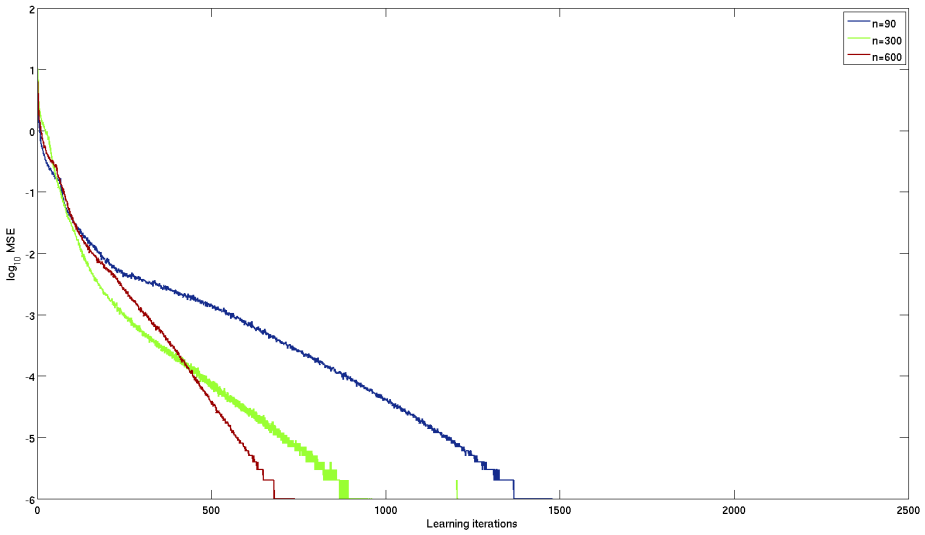

Figure 3: MSE as a function of iteration for a random constraint node and three different values of n.

$$\eta_t(x) = \begin{cases} x - \theta_t & \text{if } x > \theta_t; \\ x + \theta_t & \text{if } x < -\theta_t \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

where θ_t is the threshold at iteration t.

Equation (16a) and the first two terms of the equation (16b) are the same as other message passing methods for compressed sensing [3]. More specifically, for the original Message Passing (MP) approaches we have:

$$x(t+1) = \eta_t(A^T z(t) + x(t)) \tag{18a}$$

$$z(t) = y - Ax(t) \tag{18b}$$

The original Belief Propagation (BP) algorithms are based on the following equations:

$$x_{i\to a}^{t+1} = \eta_t \left(\sum_{b \in N_i \backslash a} A_{bi} z_{b\to i}^t \right) \tag{19a}$$

$$z_{a\to i}^{t+1} = \sum_{j \in N_b \backslash i} A_{aj} x_{j\to a}^t \tag{19b}$$

However, the above algorithm has a major drawback, which is its complexity because we are sending different messages over different links. The authors introduce equations (16) for AMP to approximate message passing equations. The term $\frac{1}{\delta} z(t-1) \left\langle \eta'_{t-1}(A^T z(t-1) + x(t-1)) \right\rangle$ in (16b) is the first order term to make this approximation more accurate.

3.1.1 Algorithm Intuition: standard Message Passing (MP) compressed sensing

To see the main rationale behind using message passing methods as explained above, consider the special case of $A^T = A{-}1$, which holds only for $n = N$. In this case, for an appropriate choice of the function $\eta_t(.)$ and disregarding the last term in equation (16b), we see that $x(1) = A^{-1}y = x_0$. Therefore, the algorithm converges in one iteration. The same analysis holds true for any matrix A with the property $A^TA = I$. To see why the approach might work for other matrices for which $A^TA \neq I$, consider the case where A is a matrix with iid Gaussian entries. Now if we assume that $x(t)$ is independent of the entries of A (which holds for $t = 0$ but not for $t > 0$ in general), then we have $x(t+1) = \eta_t(A^TAx_0 + x(t) - A^TAx(t)) = \eta_t\left(H.(x_0 - x(t)) + x_0\right)$, where $H = A^TA - I$. Now since we assumed $x(t)$ is independent of H, then the elements of $H.(x_0 - x(t))$ are iid Gaussian terms. Therefore, we can write $H(x_0 - x(t)) + x_0$ as $x_0 + g$, where g is a Gaussian noise term. Now it has been shown that soft-thresholding reduces Gaussian noise/interference [4]. Therefore, $|x(t+1) - x_0| < |x(t) - x_0|$.

3.1.2 One-dimensional Recursive Equation for Analysis

The authors use the intuition above to formulate the problem (18) as a recursive one-dimensional equation based on the mean square error in each iteration, i.e. $\sigma_t^2 = 1NE\{\|x^t - x_0\|^2\}$.

$$\sigma_{t+1}^2 = \Psi(\sigma_t^2) \tag{20a}$$

where

$$\Psi(\sigma_t^2) = E\left\{\left[\eta\left(X + \frac{\sigma}{\sqrt{\delta}}Z, \lambda\sigma\right) - X\right]^2\right\} \tag{20b}$$

In the above equations, Z $N(0,1)$ and X is a random variable whose distribution is identical to the empirical distribution of the entries of x_0.

Now if equation (20) has no fixed-point other than $\sigma = 0$, then we are sure that the algorithm converges to $x^t = x_0$ for some t large enough.

3.1.3 Some remarks

The thresholding function used in the proposed approach has a threshold which varies iteration-by-iteration. Usually, the threshold is bigger in the beginning, i.e. unless we are quite sure, we will not update variable nodes. However, the threshold reduces towards the end of the algorithms. The authors have shown that although one can choose different thresholding function $\eta_t(.)$, where SE is accurate one can hope to make much improvements over the results obtained by the thresholding functions chosen in this paper. And SE might be inaccurate when the threshold function does not result in a function $\Psi(x)$ which is **monotone** and **concave**.

3.2 Sparse Principal Component Analysis

In [6], the authors have applied ℓ_1 norm penalty, similar to LASSO, to the principle component analysis problem in order to find sparse principle components. The proposed approach is much more faster than the similar methods and yet yields better approximations in many cases, as shown by simulations.

To be more specific, we are interested in finding the first k largest Principle Components (PC) of a n data vectors of length p, stored in the form of $n \times p$ matrix X. However, we would like these PCs to be also sparse. To solve this problem, the authors formulate the requirements into an optimization problem which is a combination of PCA and lasso. To address this problem, the following optimization problem learns the first k PC's such that they are orthogonal to each other. The first term in (21a) is the information loss cost.

$$(\hat{A}, \hat{B}) = argmin_{A,B} \sum_{i=1}^{n} \|x_i A B^T x_i\|^2 + \lambda \sum_{j=1}^{k} \|\beta_j\|^2 \tag{21a}$$

subject to:

$$A^T A = I_{k \times k} \tag{21b}$$

Here, the j^{th} column of $\hat{B}$, denoted by $\hat{\beta}_j$ is proportional to the j^{th} principle component for any $\lambda > 0$.

Note that the above theorem will not necessary give us a sparse solution. Therefore, we need to have an ℓ_1 penalty term. This defines a new problem, given by equation (22).

$$(\hat{A}, \hat{B}) = argmin_{A,B} \sum_{i=1}^{n} \|x_i A B^T x_i\|^2 + \lambda \sum_{j=1}^{k} \|\beta_j\|^2 + \sum_{j=1}^{k} \lambda_j^1 \|\beta_j\|_1 \tag{22a}$$

subject to:

$$A^T A = I_{k \times k} \tag{22b}$$

To solve (22), the authors consider two steps:

1. **Find B given A**: In this case, we note that $\sum_{i=1}^{n} \|x_i A B^T x_i\|^2 = \|X - XBA^T\|^2$. Now since A is orthonormal, let $A_\perp$ be *any* orthonormal matrix such that the matrix $[A; A_\perp]$ is a $p \times p$ orthonormal matrix. Then:

$$\begin{aligned} \|X - XBA^T\|^2 &= \|XA_\perp\|^2 + \|XA - XB\|^2 \\ &= \|XA_\perp\|^2 + \sum_{j=1}^{k} \|Xa_j - X\beta_j\|^2 \end{aligned} \tag{23}$$

   Now since A is given, we only need to find the solution to the problem $argmin_B \sum_{j=1}^{k} \|Xa_j - X\beta_j\|^2 + \lambda \sum_{j=1}^{k} \|\beta_j\|^2 + \sum_{j=1}^{k} \lambda_j^1 \|\beta_j\|_1$. However, since this problem is independent for each j, one can solve the following problem to find B given A:

$$\hat{\beta}_j = argmin_{\beta_j} \|Xa_j - X\beta_j\|^2 + \lambda \|\beta_j\|^2 + \lambda_j^1 \|\beta_j\|_1 \tag{24}$$

   To solve the above problem, we can use the elastic net approach by [7].

2. **Find B given A**: When B is given, then we ignore all the sparsity-related parts and solve the following problem: $\min_A \|X - XBA^T\|^2$ subject to $A^T A = I_{k \times k}$. The solution to this problem is given by Theorem 4 and is $\hat{A} = UV^T$ where U and V come from SVD by $(X^T X)B = UV^T$.

Our work is closely related to the first step of the above algorithm because in our case we are not interested in A. More precisely, we can solve $\|0 - XB^T\|^2$ instead of $\|X - XB^T A\|^2$.

An advantage of the above algorithm in comparison to the one in [8] is its convergence speed. This rapidness enables us to try different values of λ_j^1 and pick a good one. The LARS-EN algorithm in [7] will give us good values for λ_j^1 anyway.

4 Future Works

There are a lot work needed to be done in future. For the proof of convergence of the learning algorithm (1), we have to do the followingsteps:

1. So far, we have shown that if we iteratively learn a single pattern, the MSE for this pattern in each iteration is less than the previous iteration, assuming certain conditions are satisfied. However, updating the weight vector in favor of one pattern might be against the patterns that have been already learnt. So we must find a way to ensure that one update does not ruin other patterns. It might be useful to use the proof by [1] in this case as a starting point. Another idea is to extend the one mentioned earlier in this report, i.e. pick α small enough so that the magnitude of update in each iteration is a fraction of $\|w(t)\|$. In this way, may be we can bound the amount of corruption caused by the update of pattern on the other patterns. Another idea is to consider $\sum_{mu} y(\mu, t)$ as the MSE and try to bound that.

2. For the journal paper, we must show that the algorithm does not converge to the case of $w = \mathbf{0}$, where $\mathbf{0}$ is the all-zero vector. This would be easy to show if we replace $f(w)$ in sparsity penalty wih $f(w - \alpha y(x - yw/\|w\|^2))$, i.e. first pursue the null basis direction and then if the amount of updated weight vector is very small, put it to zero. Although this is not exactly our proof here, it is an interesting option to consider. We can state this argument by saying that small weight vanish from one iteration to the other one. In fact, that might be an interesting idea! Instead of putting all small elements to zero, envision a sort of decaying function such that small weights vanish from one time slot to another. We can say we wait Δt seconds before introducing the next pattern so during this time all small weights are vanished!

3. Although simulation results show that the final solution is rather sparse, it would be nice to have thoretical results on the degree of sparsity as well.

4. It would be nice to show that $\theta_{\mu,t}$ increases monotnically instead of showing that $y(\mu, t)$ decreases monotnically.

In addition to the above items, Mr. Amin Karbasi and I are thinking to extend the approach proposed in [2] to do the learning phase of the neural algorithm we proposed in [5].

Another idea that we might address in our future works is that so far, we assume we are interested in learning *all* patterns in a subspace. What happens if we would like to memorize only sum of them? In other words, how to dinstinguish patterns that belong to a subpsapce and we have memorized from those that belong to the same subspace but we have not memorized?

References

[1] L. Xu, A. Krzyzak, E. Oja, Neural nets for dual subspace pattern recognition method, Int. J. Neur. Syst., Vol. 2, No. 3, 1991, pp. 169-184.

[2] D. L. Donoho, A. Maleki, A. Montanari, *Message passing algorithms for compressed sensing*, Proc. Nat. Acad. Sci., Vol. 106, 2009, pp. 1891418919.

[3] J. Tropp J, S. J. Wright, *Computational methods for sparse solution of linear inverse problems*, Proc. IEEE, Vol. 98, No. 6, 2010, pp. 948-958.

[4] D. L. Donoho DL, I. M. Johnstone, *Minimax risk over ℓ_p balls*, Prob. Theor. Rel. Fields., Vol. 99, No. 2, 1994, pp. 277303.

[5] K.R. Kumar, A.H. Salavati and A. Shokrollahi, *Exponential pattern retrieval capacity with non-binary associative memory*, Proc. IEEE Information Theory Workshop, 2011.

[6] H. Zou, T. Hastie, R. Tibshirani, *Sparse principal component analysis*, J. Comp. and Graphical Stat., Vol. 15, No. 2, 2006, pp. 265-286.

[7] H. Zou,T. Hastie, *Regularization and variable selection via the Elastic Net*, J. Royal Statistical Society, Series B, Vol. 67, 2005, pp. 301320.

[8] I. T. Jolliffe, N. T., Trendalov,M. Uddin, *A modied principal component technique based on the lasso*, J. Computational and Graphical Statistics, Vol. 12, 2003, pp. 531547.