A Sparse Analog Associative Memory via L1-Regularization and Thresholding

Rakesh Chalasani and Jose C. Principe, *Fellow, IEEE*

***Abstract*—The CA3 region of the hippocampus acts as an auto-associative memory and is responsible for the consolidation of episodic memory. Two important characteristics of such a network is the sparsity of the stored patterns and the non-saturating firing rate dynamics. To construct such a network, here we use a maximum *a posteriori* based cost function, regularized with L1-norm, to change the internal state of the neurons. Then a linear thresholding function is used to obtain the desired output firing rate. We show how such a model leads to a more biologically reasonable dynamic model which can produce a sparse output and recalls with good accuracy when the network is presented with a corrupted input.**

I. Introduction

The CA3 subregion of the hippocampus is responsible for the storage of the long-term episodic memory [1]. It is believed that this part of the brain that can store information for a few hours to a few weeks is a recurrent network acting as an auto-associative memory and uses synaptic weights to store the data. One of the important characteristic of this network is the sparseness of the stored data. As argued in [1], the Dentrate Granuale cells preceding the CA3 region sparsify the data entering the network in order to increase the capacity of the memory. This recurrent feedback network when presented with an input, which can be a distorted version of previously memorized data, it can iteratively recover it using the information stored in the synaptic weights.

For the reasons described in [2, and references therein] most of the firing rate based auto-associative memory models deal with the binary values. But it is observed that most neurons do not work in saturation and they have a continuous value firing rates, which undermines the usefulness of the binary inputs to study such networks. To alleviate this problem Treves [2] has proposed a model to deal with the graded response neurons. But subsequently, it has been shown that the performance of the network deteriorates significantly even for a modest graded values with 10-levels [3].

More recently, [4, 5] have used a probabilistic approach for recall from the associative memory where a general maximum a posteriori based rule is used. Particularly, in case of [5] the rule naturally leads to memory that can deal with analog values. It has been shown that such a network can be compared with the linear firing rate model of a neuron [6]. But it does not consider two important characteristics essential for an associative memory model; viz., the sparseness of the memory patterns and the positive firing rates.

In their Bayesian-Optimal network Lengyel and Dayan [5] obtained the output by considering the one that is most likely to occur given the synaptic weight matrix and the distorted input. The prior probability density of the output is considered to be Gaussian distributed which does not lead to a sparse solution as required. In this paper we set the prior to be Laplacian distributed, in line with the sparse assumption on the data. As we will show, this will enhance the ability of the network to represent the data more accurately.

Also, we show that the cost function obtained with the MAP rule can be compared with a dynamical model of the system which are described by the ordinary differentiable equations that govern the internal states and the neurons' output firing rate [6]. More elaborately, the internal state (or total synaptic current) in a neuron is changed in the direction of the gradient to minimize the cost function and subsequently, the output firing rate is obtained by using a linear thresholding function which does not saturate. But to solve for the gradient of the cost function with an L1-norm, we use an interior point method to find the direction in which the internal state of the neuron (or node) changes. In the following sections we describe this algorithm in detail.

The paper is organized as follows: in section II we describe the probabilistic approach for recall and show its relationship with the non-linear firing rate equation. In section III we propose a method to solve the cost function using interior point methods with log-barrier function and in section IV we show the results obtained using this method. We conclude the papaer in section V discussing the results obtained and possible future work.

II. Bayesian Recall and Rate equations

A. Firing Rate Model for Recurrent Networks

Consider a recurrent network with two layers, where the input layer acts as feed forward layer with a persistent activity $\hat{\mathbf{x}}$ as the total input fed into the output layer, while the nodes in the output layer have synaptic interconnects without self feedback. In such a case, for the firing rate model the dynamic system equation is given by [6]

$$\begin{aligned} \tau_r \frac{d\mathbf{u}}{dt} &= \dot{\mathbf{u}} &= -\mathbf{u} + \mathbf{W} \,.\, T_\theta(\mathbf{u}) + \hat{\mathbf{x}} \\ \mathbf{x} &= T_\theta(\mathbf{u}) \end{aligned} \tag{1}$$

Rakesh Chalasani (phone: +1 352 2262344; email: rakeshch@ufl.edu).
Jose C. Principe (phone: +1 352 3922662; email: principe@cnel.ufl.edu).
Both are with the Computational NeuroEngineering Laboratory, Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida, USA.

This work is supported by ONR grant #N000141010375

where the $\mathbf{u} \in \Re^n$ is the total synaptic current in the neurons, $\mathbf{W}$ is the synaptic weight matrix between the neurons, i.e., i, jth entry w_{ij} is the synaptic strength between the ith and jth neuron in the output layer (with $w_{ii} = 0$) and $\mathbf{x} \in \Re^n$ is the output firing rates. The function $T_\theta(.)$ is called the activation function or the threshold function, which is defined as $T_\theta(\mathbf{u}) = g[\mathbf{u} - \theta]_+$ where g is the gain.

The synaptic weight matrix is constructed using the Hebbian learning rule. Here we have considered the simple covariance synaptic learning mechanism [7] which modifies the weight matrix depending on the covariance between the pre- and post- synaptic firing rates. Since, the covariance between two nodes is commutative, the weight matrix is fully connected and symmetric.

$$w_{ij} = \sum_{p}^{P} (\eta_i^p - < \eta >)(\eta_j^p - < \eta >) \tag{2}$$

where η_i is the firing rate of the ith component of one of the stored patterns, P is the total number of stored patterns and $< \eta >$ is the mean firing rate taken across a time window greater than that of the stored elements. We consider this to be constant throughout the learning and recall process and replace it with a constant 'a' which approximately equal to the mean firing rate of all the stored patterns.

B. Bayesian Recall

Ideally, when the weight matrix is constructed as above, any recalled elements should lie in the support of only the P stored elements. But correlation between the stored patterns, which happens both in biology and in computer models, leads to many other possibilities and hence, would require a generalized method to work on the whole space of possible outcomes.

In this context, according to the maximum *a posteriori* (MAP) rule, the optimal output ($\mathbf{x}$) in a recall is the one that maximizes $P(\mathbf{x}/\mathbf{W}, \hat{\mathbf{x}})$ where $\hat{\mathbf{x}}$ is a corrupt input [5]. Using the Bayes rule

$$\begin{aligned} P[\mathbf{x}/\mathbf{W}, \hat{\mathbf{x}}] &\propto P[\mathbf{x}, \mathbf{W}, \hat{\mathbf{x}}] \\ P[\mathbf{x}, \mathbf{W}, \hat{\mathbf{x}}] &= P[\mathbf{W}/\mathbf{x}, \hat{\mathbf{x}}] P[\hat{\mathbf{x}}/\mathbf{x}] P[\mathbf{x}] \end{aligned} \tag{3}$$

If $\hat{\mathbf{x}}$ is a corrupted version of the stored pattern $\mathbf{x}$, assuming ideal recall, it is reasonable to assume that the weight dependent term in (3) is independent of $\hat{\mathbf{x}}$. Moreover, the term $P[\mathbf{W}/\mathbf{x}]$ points to the error in constructing the weight matrix when only $\mathbf{x}$ is the stored pattern. Assuming this error to be Gaussian *i.i.d*,

$$P[\mathbf{W}/\mathbf{x}] = \prod_{i,j \neq i} G(w_{i,j}; \Omega(x_i, x_j), \sigma_w^2) \tag{4}$$

where $\Omega(x_i, x_j) = (x_i - \mu_x)(x_j - \mu_x)$. Similarly, with the assumption that the noise in the corrupted input $\hat{\mathbf{x}}$ is Gaussian *i.i.d*, we get

$$P[\hat{\mathbf{x}}/\mathbf{x}] = \prod_{i} G(\hat{\mathbf{x}}_i; \mathbf{x}, \sigma_n^2) \tag{5}$$

The prior in (3) determines the distribution of the output and is significant in regularizing the cost function to obtain the desired output. As argued before, storing sparse data significantly improves the capacity of the system, therefore the prior should reflect such an assumption. An appropriate prior to obtain a sparse output is of the form

$$P[\mathbf{x}] = \exp(-\tilde{\lambda}\phi(\mathbf{x})) \tag{6}$$

where $\phi(\mathbf{x}) = \|\mathbf{x}\|_1$ leads to Laplacian distribution.

Now, substituting the above equations into (3) and taking negative log likelihood gives the cost function that needs to be minimized to obtain the desired output.

$$\begin{aligned} \mathbf{x} &= \arg\min_{\mathbf{x}} E(\mathbf{x}) \\ &= \arg\min_{\mathbf{x}} \frac{1}{2} \sum_{i,j \neq i} \|w_{i,j} - \Omega(x_i, x_j)\|^2 \\ &\quad + \frac{\beta}{2} \sum_{i} \|\hat{x_i} - x_i\|^2 + \lambda\phi(\mathbf{x}) \end{aligned} \tag{7}$$

where the parameters σ_w, σ_n, $\tilde{\lambda}$ are combined to obtain β and λ.

C. Relation Between Bayesian Recall and Neural Dynamic Equation

The negative gradient of the cost function (7) can be written as

$$\begin{aligned} -\frac{\partial E}{\partial x_i} &= -[(\beta + \sigma^2)x_i + \lambda\phi_i^{'}(\mathbf{x})] \\ &\quad + [\sigma^2 - \sum_{j \neq i} w_{ij}]a + \sum_{j \neq i} w_{ij}x_j + \beta\hat{x}_i \end{aligned} \tag{8}$$

where $\phi_i^{'} = \frac{\partial \phi}{\partial x_i}$ and σ^2 is related to the variance of the prior distribution. Now, the above obtained gradient equation can be compared with the dynamic firing rate formulation described in (1) with the assumption that $\dot{u}_i \propto -\frac{\partial E}{\partial x_i}$, i.e, the rate of change of the internal state of the neuron is proportional to the direction of the change in the output firing rate. This is a reasonable assumption as long as the function mapping the internal state of the neuron u_i to the output firing rate x_i is a monotonically increasing function; such as $T_\theta(.)$.

Comparing the two equations, while neglecting the second term in (1) as a bias term constant to all the updates,

$$\begin{aligned} u_i &= (\beta + \sigma^2)x_i + \lambda\phi_i^{'}(\mathbf{x}) \\ \implies x_i = T_\theta(u_i) &= \frac{1}{(\beta + \sigma^2)}[u_i - \lambda\phi_i^{'}(\mathbf{x})] \end{aligned} \tag{9}$$

and the synaptic weight term and the input term are similar. This shows that minimizing the above cost function (7) is similar to using the dynamic rate equation.

But this optimization is not trivial as the cost function is not continuously differentiable. In the next section we use the interior point method to solve this cost function in order to change the internal state of the system as well as the output using thresholding.

III. Interior Point method with Barrier function

Firstly, the cost function in (7) is a convex function in $\mathbf{x}$ that is not continuously differentiable. In order to minimize this cost function, it can be written as a convex quadratic function, with linear inequality constraints by adding a variable $\mathbf{y}$ such that (7) can be written as [8, and ref. there in]

$$\begin{aligned} \min \quad & \sum_{i,j\neq i} \|w_{i,j} - \Omega(x_i, x_j)\|^2 + \beta \sum_i \|\hat{x_i} - x_i\|^2 \\ & + \lambda \sum_i y_i \\ s.t \quad & -y_i \leq x_i \leq y_i \quad \forall \quad i \in \{1, 2, ..., n\} \end{aligned} \tag{10}$$

Now, the above constrained optimization problem can be solved using the interior point method(IPM) by constructing a barrier function for the inequality constraints such that the cost function become an unconstrained optimization problem [9]. It goes as follows:

The bounded constraints in (10) are re-written in the form of log-barrier function such that

$$\begin{aligned} \Phi(\mathbf{x}, \mathbf{y}) &= -\sum_i log(y_i + x_i) - \sum_i log(y_i - x_i) \\ &= -\sum_i log(y_i^2 - x_i^2) \end{aligned} \tag{11}$$

Now, for the cost function

$$\begin{aligned} \arg\min_{\mathbf{x},\mathbf{y}} \Psi(\mathbf{x}, \mathbf{y}) = \arg\min_{\mathbf{x},\mathbf{y}} & \, t \sum_{i,j\neq i} \|w_{i,j} - \Omega(x_i, x_j)\|^2 \\ & + t\beta \sum_i \|\hat{x_i} - x_i\|^2 + t\lambda \sum_i y_i + \Phi(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{12}$$

as 't' varies from 0 to ∞ the minimizer of the above cost function traces through the central path that contains the unique optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ for (10). So, by gradually increasing the value of 't' one can update the values of $(\mathbf{x}, \mathbf{y})$ to get closer to the optimal solution. In fact, it can be shown that $(\mathbf{x}^*, \mathbf{y}^*)$ can not be more than $2n/t$-suboptimal [9].

As discussed above, (12) is an unconstrained optimization problem and the cost function can be minimized using Newton's method. The search directions $(\triangle\mathbf{x}, \triangle\mathbf{y})$ are obtained by using

$$H \begin{bmatrix} \triangle\mathbf{x} \\ \triangle\mathbf{y} \end{bmatrix} = -\mathbf{g} \tag{13}$$

where $g \in \Re^{2nX1}$ is the gradient matrix of the function $\Psi(\mathbf{x}, \mathbf{y})$ and is obtained as

$$\begin{aligned} \mathbf{g} &= \begin{bmatrix} \mathbf{g1} \\ \mathbf{g2} \end{bmatrix} \\ \mathbf{g1}(i) = \nabla\Psi_{\mathbf{x}}(i) &= -2t \sum_{j\neq i} w_{ij}(x_j - a) + 2t\sigma^2(x_i - a) \\ & \quad + 2t\beta(x_i - \hat{x_i}) + \frac{2x_i}{y_i^2 - x_i^2} \\ & \quad \forall \;\; i \in \{1, 2, ..., n\} \\ \mathbf{g2} = \nabla\Psi_{\mathbf{y}} &= t\lambda\mathbf{1} - \frac{2\mathbf{y}}{\mathbf{y}^2 - \mathbf{x}^2} \end{aligned} \tag{14}$$

And the Hessian matrix, $H \in \Re^{2nX2n}$ is

$$\begin{aligned} H &= \begin{bmatrix} A + D2 & D1 \\ D1 & D2 \end{bmatrix} \\ where \;\; A &= 2t(\sigma^2 + \beta)I \\ D1 &= \frac{-4\mathbf{xy}}{(\mathbf{y}^2 - \mathbf{x}^2)^2} \\ D2 &= \frac{2(\mathbf{x}^2 + \mathbf{y}^2)}{(\mathbf{y}^2 - \mathbf{x}^2)^2} \end{aligned} \tag{15}$$

But in accordance with the discussion in the previous section, instead of updating $\mathbf{x}$ we change the internal state, $\mathbf{u}$, using $\triangle\mathbf{x}$ and subsequently find the output $\mathbf{x}$ via thresholding. This is summarized in the algorithm 1:

Algorithm 1 IPM for Sparse Associative Memory

Require: : $\epsilon_{tol} > 0$.

Initialize: $\mathbf{x}, \mathbf{u} = 0$; $\mathbf{y} = \mathbf{1} \in \Re^n$, $t = 0.01$

while $2n/t < \epsilon_{tol}$ **do**

- Compute the search directions $(\triangle\mathbf{x}, \triangle\mathbf{y})$ using (13).
- Compute the step size 's' in these directions using backtracking line search.
- Update the values as: $(\mathbf{u}, \mathbf{y}) = (\mathbf{u}, \mathbf{y}) + s.(\triangle\mathbf{x}, \triangle\mathbf{y})$
- Update $\mathbf{x} = T_\theta(\mathbf{u})$.
- Update t

end while

The variable t is updated as $\mu t_0, \mu^2 t_{,0} \mu^3 t_0 ...$ per each iteration where $\mu \in [2, 50]$ and t_0 is the initial value. A more rigorous approach is to keep the value of the t constant until the gradient is small. But since we are using the back tracking line search, changing t in each iteration is still appropriate.

The back tracking line search is used to obtain the step size 's' in search direction, i.e., give the search direction $(\triangle\mathbf{x}, \triangle\mathbf{y})$ the step size is obtained as $s = b^k, k > 0$ such that

$$\begin{aligned} & \Psi(\mathbf{x} + b^k\triangle\mathbf{x}, \mathbf{y} + b^k\triangle\mathbf{y}) \\ & \qquad \leq \Psi(\mathbf{x}, \mathbf{y}) + \alpha b^k \mathbf{g}^T [\triangle\mathbf{x} \; \triangle\mathbf{y}] \end{aligned} \tag{16}$$

where $\alpha \in (0, 1/2)$ and $b \in (0, 1)$. For details about back tracking line search algorithm readers are referred to [9].

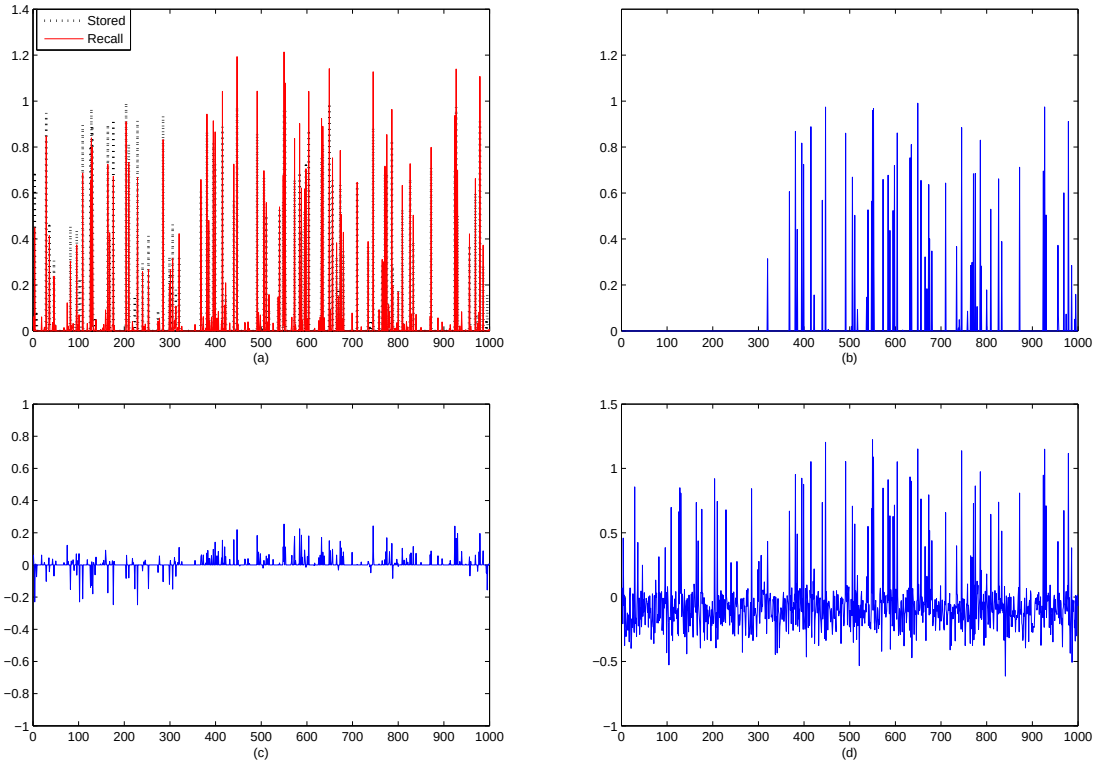

Fig. 1. Typical recall obtained when the input has missing values. (a) Stored elements in the memory which is corrupted to obtain the input compared with the recalled output. (b) Corrupted Input. (c) Point-wise error between the stored pattern and the output. (d) Internal State.

IV. Results

Most of the associative memory system deal with the inputs that have either binary or graded response. Since either cases are just special cases of an analog associative memory, we consider here only analog values to show the performance of the system.

In this section we show the results obtained when the proposed algorithm is used over sparse analog inputs. The memroy elements are obtained from a uniform distribution between $[0, 1]$ with most of the points zero, i.e., sparse data. We consider the case where some part of the signal is missing and this can be extended as well to cases where the input is corrupted with noise.

Fig.1 shows a typical recall obtained when a corrupted input cue (with missing values) is presented to the system. The size of the network is 1000 and the number of stored elements is 100 with density (or degree of sparseness) of each element equal to 0.1 (10% sparse). The parameters are set to the following values:

$$\lambda = 10; \quad \beta = 20; \quad \theta = 0.05.$$
$$\alpha = 0.0001; \quad b = 0.5; \quad \mu = 50 \tag{17}$$

The value of β is annealed (by a factor of 0.5) during each iteration until it is sufficiently small. This is in line with the cost function where β is dependent on the variance of the error between the input and the output. Also, ideally we would like to have the threshold,$\theta = 0$. But because of the jitters around the zeros values, mainly because of the approximation in the interior point method, the threshold is taken as a small positive value. The values of these parameters are kept constant for all the results obtained in this section; if only specified otherwise.

In Fig.1 (a) the recall obtained is compared with the stored pattern. In fact, the performance of the method can be quantified in terms of the correlation between the input, output and the stored pattern. In this case, the correlation between the input and stored pattern is 0.7553, while that between the output and the stored pattern is 0.9505.

The main factors that influence the performance of the system are the number of stored patterns and distortion induced in the inputs. Firstly, we consider the effect of the distortion on the performance. Fig. 2 shows a plot of the correlation between the input and the stored patterns versus the correlation between output and the stored pattern. The Fig.2 also indicates the $x = y$ line. A point on this line would effectively mean that the input cue is reproduced at the output without any additional information about the stored pattern. Any point above the line indicates that there is some information extracted from the memory that is not present in the cue (the farther the better) and any point below would indicate loss of information, i.e., the output is worse than the cue (the farther the worser). Thus, such a plot can tell whether the cue presented to the system has led the system

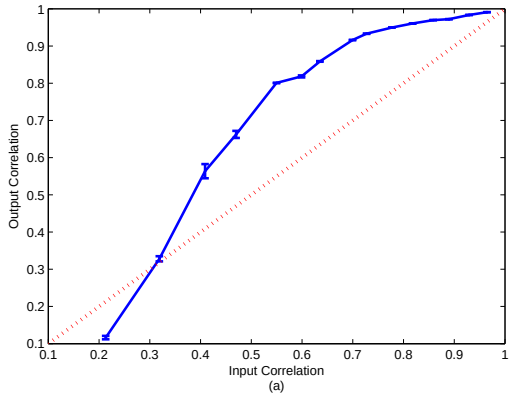

Fig. 2. The plots show the relation between the correlation between the input cue and the recalled output with the stored element. (a) In case the input cue has missing values. The straight line indicates equal correlation values ($x = y$).

into the right attractor basin or not. High correlation between the output and memorized pattern indicates that the output is in fact in the right basin of attraction and lower correlation indicates it is in the wrong basin of attraction.

From the plot in Fig. 2, it can be inferred that as long as the correlation is sufficiently high the output is heading towards the right attractor basin. It can also be observed that there is a sharp decrease in the performance of the system when the correlation of the input cue falls below ≈ 0.5. This indicates that when the cue is distorted more than a critical value it can no longer be recalled correctly from the memory. This is consistent with the theoretical results obtained shown in [1].

Another factor that influences the performance of the system is the number of elements stored in the memory. With the increase in the number of elements stored in the memory the information encoded per synapse increases leading to interference between the two stored data. Having sparse patterns reduces the covariance between the stored pattern and in turn, reduces the interference allowing larger storage capacity. But there will be a critical loading factor, the ratio of the number of memory elements stored divided by the number of units, beyond which the system fails to recall the contents of the memory; in other words the system no longer acts as a memory.

Fig. 3 shows the results obtained when the loading factor of the memory is increased. They are obtained for two cases, when the correlation between the input and the memorized pattern is set constant at 0.8 and 0.9. The plots indicate that with the increase in the number of stored patterns the recall performance decreases. In particular, there is a steep fall in performance when the loading factor is beyond 0.7-0.8. This indicates that the critical loading factor is somewhere between these values.

We also benchmark our algorithm with the method proposed by Treves [2]. This method is developed to work with the graded responses without any saturation and sparseness

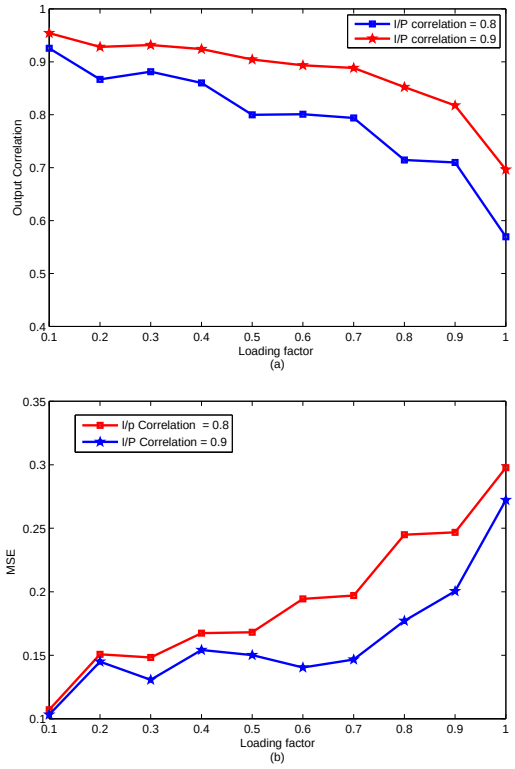

Fig. 3. The figure shows the variation in the performance of the system with the increase in the loading factor. (a) The plot between the correlation of the output with the memorized pattern versus the loading factor. (b) The plot between the mean squared error (MSE) between the output and the memorized pattern versus the loading factor. The size of the network is 100 and number of stored pattern vary from 10 to 100. The memory patterns are obtained from a uniform distribution with sparseness density 0.4. The parameters of the system are kept the same as indicated in (17). Each point in the curve is obtained by averaging over 10 recalls obtained from 10 different memories.

of the patterns stored is also taken into consideration. In this algorithm, the patterns are stored similar to covariance rule and the update rule is asynchronous. At each step, the 'local field' is calculated as $h_i = \sum_{j \neq i} w_{i,j} x_j - \kappa(a - \bar{\mathbf{x}})^3 + \hat{x_i}$, then the output is obtained by passing it through the threshold function $T_\theta()$. Here we consider $\theta = 0$ and κ is set by iterative search to find the optimal solution. Table. 1 summarizes the results obtained for both the methods where the stored patterns are obtained from the uniform sparse distribution with density 0.4 and each pattern has 100 samples. The performance of our method when compared with the Treves' network is much better. In fact, in most cases the recall obtained using the Treves' network is worse than the input itself. This is consistent with the results obtained in [3, 5] where is shown that the performance of the network deteriorates greatly with the use of higher levels of graded response or continuous values. It is also observed in our simulations that the performance of this network is very sensitive to its parameter κ.

TABLE I
COMPARING THE PERFORMANCE OF THE RECALL WHEN USE INTERIOR POINT METHOD (IPM) AND THE METHOD PROPOSED IN [2] (TREVES). FOR BOTH THE METHODS, THE CORRELATION OF THE RECALLED OUTPUT WITH THE MEMORIZED PATTERN THAT IS CORRUPTED TO OBTAIN THE INPUT IS SHOWN FOR TWO DIFFERENT LOADING FACTORS. THE ENTRIES SHOULD BE READ AS 'MEAN ± SD'.

Loading Factor	I/P corr.	O/P corr. (IPM)	O/P corr. (Treves)
0.2	0.89 ±0.02	0.95 ±0.020	0.89 ±0.170
	0.80 ±0.04	0.89 ±0.041	0.72 ±0.100
	0.69 ±0.07	0.83 ±0.072	0.59 ±0.010
	0.56 ±0.13	0.64 ±0.013	0.50 ±0.097
0.5	0.91 ±0.02	0.91 ±0.028	0.86 ±0.043
	0.79 ±0.06	0.80 ±0.060	0.74 ±0.060
	0.70 ±0.09	0.71 ±0.098	0.65 ±0.095
	0.54 ±0.09	0.55 ±0.098	0.48 ±0.080

V. CONCLUSION

We have shown in this paper the use of L1-norm regularization to obtain a sparse auto-associative memory. The cost function is obtained by considering a MAP based recall procedure that is used to update the internal state of the nodes. A regular thresholding function is used to obtain the output firing rate from the internal state of the nodes. The results show that this method can produce a sparse solution and is able to recall the desired 'memorized' pattern with good accuracy.

However, the use of the interior point method might not be biologically possible. But the motivation of this paper is to show that the cost function to be minimized is in fact related to the dynamic firing rate model of a neuron. Moreover, it has been shown that L1-norm regularized functions can be solved using thresholding functions [10], albeit in the case of simple regression case. We hope that the results obtained is a good motivation to use it and a biologically relevant way of solving the above cost function is being pursued.

APPENDIX

The negative gradient of the cost function (7) with respect to the variable x_i is:

$$\begin{aligned} -\frac{\partial E}{\partial x_i} &= \sum_{j\neq i}(w_{ij} - \Omega(x_i, x_j)) * (x_j - a) \\ &\quad +\beta(\hat{x_i} - x_i) - \lambda\phi_i^{'}(\mathbf{x}) \\ &= \sum_{j\neq i} w_{ij}(x_j - a) - (x_i - a)\sum_{j\neq i}(x_j - a)^2 \\ &\quad +\beta(\hat{x_i} - x_i) - \lambda\phi_i^{'}(\mathbf{x}) \end{aligned}$$

Now, in the second term in the above equation $\sum_{j\neq i}(x_j - a)^2$ can be considered in terms of the variance of the prior distribution when $(n-1)$ samples are considered. This implies

$$\sum_{j\neq i}(x_j - a)^2 = (n-1)\sigma_x^2 = \sigma^2$$

Substituting this back into the gradient equation and rearranging term,

$$\begin{aligned} -\frac{\partial E}{\partial x_i} &= -[(\beta + \sigma^2)x_i + \lambda\phi_i^{'}(\mathbf{x})] \\ &\quad +[\sigma^2 - \sum_{j\neq i} w_{ij}]a + \sum_{j\neq i} w_{ij}x_j + \beta\hat{x}_i \end{aligned}$$

REFERENCES

[1] E. Rolls, *Memory, Attention, and Decision-Making: A unifying computational neuroscience approach*, 1st ed. Oxford University Press, USA, August 2007.

[2] A. Treves, "Graded-response neurons and information encodings in autoassociative memories," *Physical Review A*, vol. 42, no. 4, pp. 2418–2430, Aug 1990.

[3] E. T. Rolls, A. Treves, C. P. Vicente, and D. Foster, "Simulation studies of the CA3 hippocampal subfield modelled as an attractor neural network," *Trans. Soc. Comput. Simul. Int.*, vol. 14, pp. 1559–1569, December 1997.

[4] F. T. Sommer and P. Dayan, "Bayesian retrieval in associative memories with storage errors," *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 705–713, July 1998.

[5] M. Lengyel and P. Dayan, "Rate- and Phase-coded Autoassociative Memory," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 769–776.

[6] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, 1st ed. The MIT Press, September 2005.

[7] T. J. Sejnowski and P. K. Stanton, *Covariance Storage in the Hippocampus*. New York: Academic Press, 1989, pp. 365–377.

[8] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An Interior-Point Method for Large-Scale L1-Regularized Least Squares," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, no. 4, pp. 606–617, 2007.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[10] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural Comput.*, vol. 20, pp. 2526–2563, October 2008.