The Hopfield Associative Memory Network: Improving Performance with the Kernel "Trick"

Cristina García and José Alí Moreno

Laboratorio de Computación Emergente,
Facultades de Ciencias e Ingeniería,
Universidad Central de Venezuela
{cgarcia, jose}@neurona.ciens.ucv.ve
http://neurona.ciens.ucv.ve/laboratorio

Abstract. This paper provides a new insight into the training of the Hopfield associative memory neural network by using the kernel theory drawn from the work on kernel learning machines and related algorithms. The kernel "trick" is used to define an embedding of memory patterns into (higher or infinite dimensional) memory feature vectors allowing the training of the network to be carried out in the feature space without ever explicitly representing it. The introduction of a kernel function in the training phase of the network considerably improves the performance of the network. This enhanced performance is experimentally investigated on sets of random memory patterns.

1 Introduction

Without doubt one of the most profusely studied neural network models has been the Hopfield associative memory neural net (HNN) [1, 2]. Indeed, from a superficial examination of the literature it becomes obvious that very much effort has been invested in its characterization. The interest has mainly been focused on the capacity of the network, on its learning rules and learning dynamics as well as on the dynamics of pattern retrieval [3–11]. The Hopfield neural model consists on a network of N two state ($S_i = \pm 1$) neurons or processing units of the McCoulloch and Pitts type, fully connected to each other through synaptic weights w_{ij}. The neural network conforms a dynamical system that evolves in discrete time according to a relaxation dynamics. These neural models are commonly used as autoassociative content addressable memories. That is, they are able to retrieve a previously learnt memory pattern from an example which is similar to, or a noisy version of one of the memory patterns. A pattern is represented by associating each of its components to one of the processing units of the HNN. Memory recall proceeds by initially clamping the neurons to a starting pattern $\boldsymbol{S}(t = 0)$ and evolving in time the state of the network according to the dynamical rule towards the nearest attractor state. In consequence for the network to perform properly as an autoassociative memory, the activity pattern of the attractor state must correspond to a memory pattern. When this is the case

C. Lemaître, C.A. Reyes, and J.A. Gonzalez (Eds.): IBERAMIA 2004, LNAI 3315, pp. 871–880, 2004.

it is said that $\boldsymbol{S}(t=0)$ lies in the basin of attraction of the corresponding memory pattern. Hence the purpose of a learning algorithm for the HNN is to find a synaptic weight matrix w_{ij} such that given p memory patterns, they become fixed points of the dynamics and that patterns resulting from small deviations of the memory patterns be in their basin of attraction.

The necessary condition for the memories to be fixed points of the dynamic is equivalent to stating that each processing unit of the network, acting as a linear binary classifier, solves a classification problem [5, 6]. The classification problem on each processing unit is defined by the set of memory patterns, acting as inputs with class labels equal to the associated memory component. In this respect most of the advanced learning rules proposed in the literature for the HNN [4, 5, 6] are sort of variants of the old perceptron learning rule for linear classifiers. On the other hand the condition that the basins of attraction be as large as possible is taken into account by training the processing units so as to maximize a stability operative criterion. The idea is that the output activity of the processing units be robust against small deviations of the input patterns from the memory patterns used in training.

The Adatron learning rule, an algorithm proposed in the statistical mechanics literature [6], implements this idea. From the point of view of linear classifiers the Adatron algorithm is an alternative formulation of the linear maximum margin classifier [10]. In summary the training of an optimal HNN is equivalent to the training of N linear maximum margin classifiers, one for each processing unit of the network. Recently [11] the Adatron algorithm has been generalized with the incorporation of the kernel trick. In this way the implementation simplicity of the Adatron is enhanced with the capability of working in a nonlinear feature space as support vector machines do. The result is a kernel machine that maximizes the margin in a feature space, which is equivalent to nonlinear decision boundaries in input space. The basic idea behind the "kernel trick" is that under certain conditions the kernel function can be interpreted as an inner product in a high dimensional Hilbert space (feature space). A fact that has been used extensively in generating non-linear versions of conventional linear supervised and unsupervised learning algorithms [10].

In the present case, the application of this procedure to the HNN defines an embedding of memory patterns into (higher or infinite dimensional) memory feature vectors allowing the training of the network to be carried out in the feature space without ever explicitly representing it. We refer to this enhanced neural model a kernel-HNN. The result is that the performance of the network is improved in two aspects. First, the condition that arbitrary memory patterns be attractors of the network dynamics is readily met by the selection of an adequate kernel. That is the classification problem at each processing unit has always a solution. Second, the basin of attraction of the memory patterns are increased improving the capacity of recall of the network. Experimental simulation runs on a kernel-HNN trained with a set of uncorrelated random memory patterns for different memory loads and several kernel functions are carried out. The experimental results show a significant performance gain in the pattern retrieval

dynamics with respect to the linear HNN model. Qualitative measures of the increase of the basins of attraction are reported. The organization of the paper is as follows: in section 2 an introduction to the the formalism of the HNN model is presented; in section 3, the Kernel extension of the Adatron learning algorithm is described and in section 4 the kernel-HNN model introduced; in section 5 the experiments and results are discussed and, in section 6, the conclusions are presented.

2 The Hopfield Associative Memory Network

The HNN consists on a network of N McCoulloch and Pitts type processing units fully connected to each other through a matrix of synaptic weights w_{ij}, with no self coupling ($w_{ii} = 0$). The exclusion of the self coupling terms enlarge the basins of attraction [8]. The dynamics of such a system is taken to be a simple zero-temperature Monte Carlo process

$$S_i(t+1) = sign(\sum_j w_{ij} S_j(t)) = sign\left(\langle \boldsymbol{w_i} \cdot \boldsymbol{S}(t) \rangle\right) \tag{1}$$

where $\boldsymbol{S}(t)$ is the vector of activities of the network at time t ($S_i = \pm 1$), $\boldsymbol{w_i}$ the weight vector of the ith processing unit and the brackets denotes dot product. The update of the network is performed asynchronously. The configuration of synaptic weights is generally adjusted by a learning algorithm such that, the given p memory patterns

$$\boldsymbol{Y}^\nu = (y_1^\nu, \ldots, y_N^\nu)^T, \quad y_i^\nu = \pm 1, \quad \nu = 1, \ldots, p \tag{2}$$

become locally stable fix points of the dynamics. That is, the set of inequalities

$$y_i^\nu h_i^\nu = y_i^\nu \sum_j w_{ij}\ y_j^\nu \geq c > 0 \tag{3}$$

must be satisfied for every memory pattern ν and processing unit i. Since the diagonal elements of the weight matrix are null, the above set of inequalities decouple and each processing unit can be treated independently. We can then consider the case for one processing unit only, say $i = 1$, and omit the index i in the following. The condition for the stability of the memories is then written

$$y^\nu h^\nu = y^\nu \sum_j w_j\ y_j^\nu = \gamma^\nu \geq c > 0 \tag{4}$$

where w_j are the components of the weight vector of a linear processing unit with component $w_i = 0$ (i is the index of the processing unit). This condition denotes the positiveness of the functional margins γ^ν, of the input vectors $\boldsymbol{Y}^\nu$ with labels y^ν (ith component of the memory pattern) with respect to the linear classifier defined by the weight vector $\boldsymbol{w}$. This is the separability condition of the set of memory patterns and corresponding labels by the linear classifier on

the ith processing unit [10]. The associative memory network will be properly trained when all classification problems are simultaneously separable. In another case the recall dynamics will be unstable and converge to a pattern of activities different from a memory.

Fulfillment of inequality (4) is a necessary condition for the HNN to operate as an associative memory but in addition large basins of attraction are desirable for each memory pattern. Thus the fix points should be stable against many component inversions of the activity patterns of the network in its time evolution. As a sufficient condition it has been argued [5, 6] that the functional margins γ^ν in (4) should be large relative to the magnitudes of the synaptic weights. A normalized measure of stability for the processing units is then defined as:

$$\hat{\gamma} = \min_{\nu}(\hat{\gamma}^\nu) \quad \hat{\gamma}^\nu = \frac{\gamma^\nu}{||\boldsymbol{w}||} = \frac{y^\nu \sum_j w_j \; y_j^\nu}{||\boldsymbol{w}||} \tag{5}$$

expression which is equivalent to the geometrical margin [10] of the set of input memories with respect to the linear classifier defined by the weight vector $\boldsymbol{w}$.

The Adatron algorithm is a learning algorithm which attains optimal stability, i.e. a configuration of weights with maximum geometrical margin for a set of arbitrary memory patterns. It is an alternative formulation of the maximum margin classifier algorithm well known in the machine learning literature. An enhanced version of the algorithm with the application of the kernel trick has been presented [11]. In this way the learning proceeds in a high-dimensional memory feature space providing a non-linear version of the conventional linear perceptronlike learning algorithm. By introducing Kernels into the algorithm the measure of stability of the processing units is maximized in memory feature space, that is the stability conditions for the memories, inequalities in (4), become nonlinear in input memory space. This allows that, with a suitable kernel choice, sets of memory patterns for which the conditions (4) could not be fulfilled in input memory space, non separable set, do satisfy them in memory feature space. Furthermore as we experimentally show the basin of attraction of the memories are increased and the information content of the network is enhanced.

3 The Kernel Trick in the Adatron Algorithm

The basic idea behind the Adatron algorithm of Anlauf and Biehl [6] is to induce a configuration of synaptic weights that maximize the stability of the processing unit (5). They assume a scale for the weight vectors where the positive constant c in relation (4) is unity. In this way the algorithm is derived from the problem of minimizing the norm of the weight vector subject to the restrictions that the functional margins of the memory patterns be greater or equal to unity:

$$minimize \; \frac{1}{2}||\boldsymbol{w}||^2 \quad s.t. \; \gamma^\nu = y^\nu \sum_j w_j \; y_j^\nu \geq 1 \; \forall \nu = 1, \ldots, p \tag{6}$$

In the dual formulation of this optimization problem they show that the solution has the form:

$$\boldsymbol{w} = \frac{1}{N} \sum_{\nu=1}^{p} \alpha_\nu \; y^\nu \boldsymbol{Y}^\nu \tag{7}$$

where the α_ν are positive Lagrange multipliers which can be interpreted as a measure of the information contribution that each memory pattern does to the synaptic weights. These dual variables are learned, from a *tabula rasa* initialization, by an iterative procedure where the memory patterns are presented repeatedly and the following corrections are applied for each memory:

$$\delta\alpha_\nu = \max\{-\alpha_\nu, \lambda(1-\gamma^\nu)\} \tag{8}$$

where λ is a learning rate. The learning is done when the minimum functional margin equals unity. This algorithm is presented with theoretical guaranties of convergence to the optimal solution and of a rate of convergence exponentially fast in the number of iterations, provided that a solution exists.

It is interesting to note that the above optimization problem can be entirely formulated in terms of dot products hence it can be solved in an arbitrary feature feature space induced by a kernel function. A symmetric function $K(\boldsymbol{x}, \boldsymbol{y})$ is a kernel if it fulfills Mercer's condition, i.e. the function K is (semi) positive definite. When this is the case there exists a mapping ϕ such that it is possible to write $K(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{y}) \rangle$. That is the kernel represents a dot product on a different space F called feature space into which the original vectors are mapped. With the introduction of a suitable kernel function the above learning procedure of the Adatron can be carried out in an arbitrary feature space where on the one hand the linear separability of the problem be guaranteed for every processing unit and on the other, the high dimensionality of the feature space produces an increment on the capacity of the network.

Summarizing, in [11] is shown that the learning rule given in expression (8) remains invariant when considering learning in feature space and the form of the resulting kernel classifier on each processing unit of the network is

$$S_i(t+1) = sign\left(\sum_{\nu=1}^{p} \alpha_\nu \; y_i^\nu K(\boldsymbol{Y}^\nu, \boldsymbol{S}(t)) \right) \tag{9}$$

$K(\boldsymbol{Y}^\nu, \boldsymbol{S})$ is the kernel function and the α_ν are the Lagrange multipliers resulting from the solution of the optimization problem (6) in its dual representation. They are obtained in the training procedure. In the literature [10] the most common used kernels are the linear, polynomial, and Gaussian kernels shown in relation (10). The linear kernel is the simple dot product in input space whereas the other kernel functions represent dot products in arbitrary feature space.

$$\begin{aligned} K_l(\boldsymbol{Y}, \boldsymbol{S}) &= \langle \boldsymbol{Y} \cdot \boldsymbol{S} \rangle \\ K_p(\boldsymbol{Y}, \boldsymbol{S}) &= (\langle \boldsymbol{Y} \cdot \boldsymbol{S} \rangle + b)^n \\ K_G(\boldsymbol{Y}, \boldsymbol{S}) &= \exp(-||\boldsymbol{Y} - \boldsymbol{S}||^2/\sigma^2) \end{aligned} \tag{10}$$

In this work, experiments with the three of the kernel functions in (10) are carried out.

4 Kernel Hopfield Network

Expression (9) is a generalization of the dynamic equation in (1) defining the dual representation of the dynamic of the HNN in feature space:

$$S_i(t+1) = sign\left(\langle \boldsymbol{w_i} \cdot \boldsymbol{S}(t)\rangle\right) = sign\left(\sum_{\nu=1}^{p} \alpha_\nu \; y_i^\nu K(\boldsymbol{Y}^\nu, \boldsymbol{S}(t))\right) \tag{11}$$

It is important to note that the described formalism does not provide an equation for the synaptic weights similar to relation (7), as a linear combination of the transforms of the memory patterns in feature space. The reason for this is simply that the explicit mapping of the input memories into feature space is not known. Nevertheless rewriting the definitions (10) of the kernel functions in terms of the products of components of the involved vectors in input space an expression equivalent to (7) for the weight vectors can be given as a generalized product of functions of the memory components. It is shown in [10] that for the polynomial kernel this is readily done. As an example let us consider a simple polynomial kernel with $b = 0$ and $n = 2$. Equation (11) can be written

$$S_i(t+1) = sign\left(\langle \boldsymbol{w_i} \cdot \boldsymbol{S}(t)\rangle\right) = sign\left(\sum_{\nu=1}^{p} \alpha_\nu \; y_i^\nu \langle \boldsymbol{Y}^\nu \cdot \boldsymbol{S}(t)\rangle^2\right) \tag{12}$$

since

$$\langle \boldsymbol{Y}^\nu \cdot \boldsymbol{S}(t)\rangle^2 = \sum_{k=1}^{N} \sum_{j=1}^{N} y_k^\nu \, y_j^\nu \, S_k \, S_j \tag{13}$$

Inserting (13) in (12)

$$S_i(t+1) = sign\left(\sum_{(k,j)=(1,1)}^{(N,N)} S_k \, S_j \sum_{\nu=1}^{p} y_i^\nu \, \alpha_\nu \, y_k^\nu \, y_j^\nu\right) \tag{14}$$

The argument of the function *sign* in expression (14) is a generalized inner product of feature vectors with components $S_k S_j$ and $y_k^\nu y_j^\nu$. It can be recognized that the summatory on the memory patterns $y_k^\nu y_j^\nu$ is of the same form as expression (7). The writing of an equivalent expression for the Gaussian kernel is a more involved procedure that takes us out of the scope of this paper, but in general one such expression can be written.

In conclusion the kernel trick allows a straightforward generalization of the Hopfield network to a to higher dimensional feature space. The important advantage of this procedure is that in principle all processing units can be trained to optimal stability. Such a network of fully optimal processing units show several important improvements that can be experimentally shown: larger memory capacity, increased memory recall capacity and bigger basins of attraction.

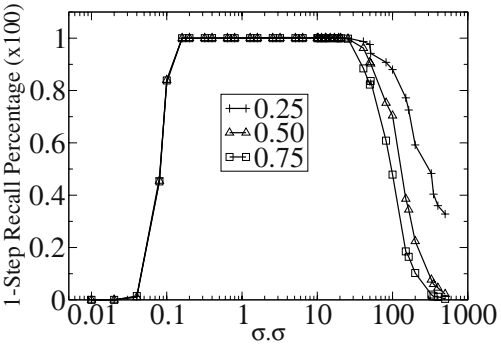

Fig. 1. Fraction of correctly recognized patterns in one dynamical step as a function of the parameter of the Gaussian kernel, for three values of the memory loading parameter: 0.25, 0.5 and 0.75

5 Experiments and Results

In order to investigate the kernel-HNN in a systematic fashion a statistical analysis of several simulations were carried out. Networks were trained for different memory loads. For every trained network, experiments on the recovery of each memory pattern contaminated with a given percentage of random noise were repeated 50 times. The reported results are the values of the fraction of correctly recovered patterns. A $N = 100$ processing unit Hopfield network was used throughout. Three kinds of kernel functions were investigated: linear, polynomial with $b = 0.1$ and $n = 2$; and Gaussian kernel. The parameter value of the Gaussian was established experimentally. The dynamic of the network is synchronous in the one step memory recall experiment and asynchronous in the others.

5.1 One Step Memory Recall

In these experiments the fraction of correctly recovered memory patterns, in one dynamical step, are measured as a function of the kernel parameter for various values of the memory loading parameter $\alpha = p/N$ $(0.25, 0.5, 0.75)$. The experiments measure a sort of direct basin of attraction of the memories. It was carried out only on the network with Gaussian kernel in order to establish an adequate value of the kernel parameter. The kernel parameter is swept over several orders of magnitude. The results are shown in Figure 1.

It can be appreciated that the direct basins of attraction suffer a slight decrease with memory loading but remain essentially invariant over a broad range of the kernel parameter values. Since greater parameter values produce sparser dual representations, the results suggest that a reasonable good choice for the value of this parameter is $\sigma^2 = 25$. This value is used in the rest of the experiments.

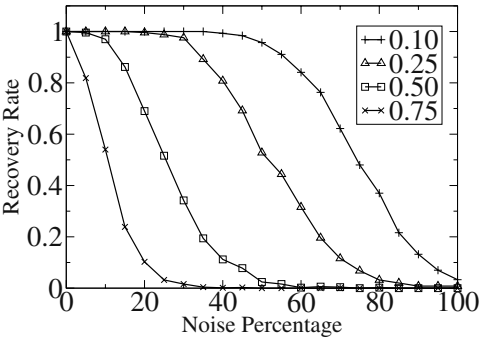

Fig. 2. Fraction of correctly recognized patterns in a linear kernel network as a function of the initial random noise contamination, for four values of the memory loading parameter: 0.10, 0.25, 0.5 and 0.75

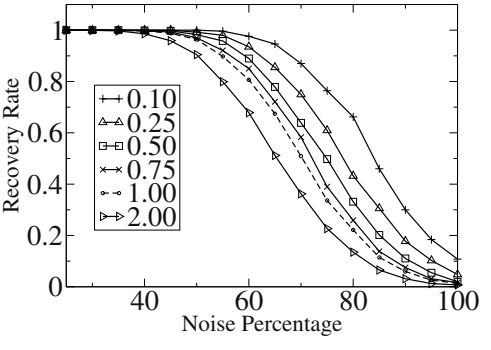

Fig. 3. Fraction of correctly recognized patterns in a polynomial kernel network as a function of the initial random noise contamination, for six values of the memory loading parameter: 0.10, 0.25, 0.5, 0.75, 1.0 and 2.0

5.2 Experiments on the Capacity of Memory Recall

In these experiments the fraction of correctly recovered memory patterns are measured as a function of the magnitude of the initial distortion of the memories (noise contamination) for various values of the loading parameter. The experiments were carried out for the three kernel types. In figure 2 the results for the linear kernel are shown.

It can be appreciated that the basins of attraction shrink with increasing value of the loading parameter. Already for loading parameter 0.75 the network does not support any contamination of the input stimulus. This is the typical behavior of standard Hopfield Networks [8]. In Figure 3 the results for a network with polynomial kernel are presented.

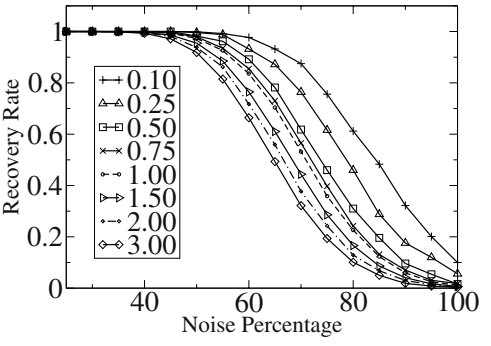

Fig. 4. Fraction of correctly recognized patterns in a Gaussian kernel network, $\sigma^2 = 25$ as a function of the initial random noise contamination, for eight values of the memory loading parameter: 0.10, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0 and 3.0

In this case the basins of attraction do decrease in a moderate way with increasing values of memory loading but remain robust for loading parameter values as high as 2. The network is tolerant to input contaminations as high as 40% for the highest memory load of 2.0. This is indicative of the presence of large basins of attraction. Finally in Figure 4 the results for a network with Gaussian kernel are shown.

It can be appreciated that the basins of attraction again with this kernel decrease in a moderate way with increasing values of memory loading but remain robust for loading parameter values as high as 3. The network again evidence the presence of large basins of attraction.

It is clear from this series of experiments that the kernel-HNN model is superior to the standard linear model in several aspects: increased memory capacity, larger basins of attraction and hence a great robustness to noise contamination of the stimuli patterns.

6 Conclusions

We have presented a generalization of the Hopfield auto-associative memory network taking advantage of the kernel trick already well known in the Machine Learning literature. It is experimentally shown that the kernel-Hopfield networks are clearly superior in performance to the standard linear model. Important observables of these models like memory capacity, recall capacity, noise tolerance and size of the basins of attraction are clearly improved by the introduction of the kernel formalism. That is, an implicit mapping of the network to a higher dimensional feature space. In this generalization procedure the learning algorithm of the network remains in its simple form as it is applied to linear networks. This means that the training complexity is by no means affected by the generalization. Nevertheless, the training of an N processing units kernel-HNN

model implies the calculation of N kernel-Adatron maximal margin classifiers over the set of memory patterns. Resulting in a considerable computational burden as N increases one or two orders of magnitude. It is clear that further work, both theoretical as computational is necessary in the model. Computationally is necessary to investigate the possibility of implementing larger networks ($1000 - 10000$ processing units) in order to make the model attractive for real life applications. We are engaged in the parallel implementation of these models on clusters of workstations. Finally more theoretical work is also needed in the formalization of this generalized neural network model.

References

1. Muller, B., Reinhardt, J.: Neural Networks. An Introduction. Springer - Verlag, Berlin (1990)
2. Hertz, J., Krogh, A., Palmer, R.: Introduction to the Theory of Neural Computation. Addison-Wesley, Redwood City, CA (1991)
3. Personnaz, L., Guyon, I., Dreyfus, J.: Collective computational properties of neural networks: New learning mechanisms. J. Physique Lett. **16** (1985) 1359
4. Diederich, S., Opper, M.: Learning of correlated patterns in spin-glass networks by local learning rules. Phys. Rev. Lett. **58** (1987) 949
5. Krauth, W., Mezard, M.: Learning algorithms with optimal stability in neural networks. J. Phys. A **20** (1987) 1745
6. Anlauf, J., Biehl, M.: The adatron - an adaptive perceptron algorithm. Europhysics Letters **10** (1989) 687–692
7. Opper, M.: Learning times of neural networks: Exact solution for a perceptron algorithm. Phys. Rev. A **38** (1988) 3824
8. Gardner, E.: The space of interactions in neural network models. J. Phys. A **21** (1988) 257
9. Gardner, E., Derrida, B.: Optimal storage properties of neural network models. J. Phys. A **21** (1988) 271
10. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press (2000)
11. Friest, T., Campbell, C., Cristianini, N.: The kernel-adatron: A fast and simple learning procedure for support vector machines. In: Proceedings of the Fifteenth International Conference on Machine Learning. Morgan - Kaufmann, San Francisco, CA (1998)