Associative Memory Design Using Support Vector Machines

Daniele Casali, Giovanni Costantini, Renzo Perfetti, and Elisa Ricci

***Abstract*—The relation existing between support vector machines (SVMs) and recurrent associative memories is investigated. The design of associative memories based on the generalized brain-state-in-a-box (GBSB) neural model is formulated as a set of independent classification tasks which can be efficiently solved by standard software packages for SVM learning. Some properties of the networks designed in this way are evidenced, like the fact that surprisingly they follow a generalized Hebb's law. The performance of the SVM approach is compared to existing methods with nonsymmetric connections, by some design examples.**

***Index Terms*—Associative memories, brain-state-in-a-box neural model, support vector machines (SVMs).**

I. Introduction

SINCE the seminal paper of Hopfield [1], a lot of synthesis methods have been proposed for the design of associative memories (AMs) based on single-layer recurrent neural networks. The problem consists in the storage of a given set of prototype patterns as asymptotically stable states of a recurrent neural network [2]. A review of some methods for AM design can be found in [3]. In this paper, we adopt the generalized brain-state-in-a-box (GBSB) neural model introduced by Hui and Zak [4]. This model was analysed by several authors and its usefulness for AM design is known [5]–[7]. In particular, we consider the design of GBSB neural networks with nonsymmetric connection weights. This design problem was addressed in [5], where the weights are computed using the pseudoinverse of a matrix built with the desired prototype patterns. Moreover, in [5] a method to determine the bias vector is suggested in order to reduce the number of spurious stable states. Chan and Zak [7], inspired by a previous work [8], proposed a "designer" neural network capable of synthesizing in real time the connection weights of the GBSB model. Chan and Zak reformulated the design of the AM as a constrained optimization problem; then mapped the design problem onto an analog circuit consisting of two interconnected subnetworks. The first subnetwork implements the inequality constraints, giving penalty voltages when constraints are violated. The second subnetwork is made of integrators whose outputs represent the connection weights of the AM.

Over the past few years, support vector machines (SVMs) have aroused the interest of many researchers as an attractive alternative to multilayer feedforward neural networks for data classification and regression [9]. The basic formulation of SVM learning for classification minimizes the norm of a weight vector that satisfies a set of linear inequality constraints. This optimization problem can easily be adapted to incorporate the conditions for asymptotic stability of the states of a recurrent neural network. So, it seems useful to exploit the relation between these two paradigms in order to take advantage of some peculiar properties of SVMs: The "optimal" margin of separation, the robustness of the solution, the availability of efficient computational tools. In fact, SVM learning can be recast as a convex quadratic optimization problem without nonglobal solutions; it can be solved by standard routines for quadratic programming (QP) or, in the case of large amounts of data, by some fast specialized solvers.

The paper is organized as follows. In Section II, we briefly review linear SVMs. In Section III, we examine the relation existing between the design of a recurrent associative memory based on the GBSB model and support vector machines and propose a new design method. In Section IV we compare the proposed approach to existing techniques, through some design examples and simulation results. Finally, in Section V some conclusions are given.

II. Support Vector Machines

Let $(\mathbf{x}^k, y_k), k = 1, \dots, m$ represent the training examples for the classification problem; each vector $\mathbf{x}^k \in R^n$ belongs to the class $y_k \in \{-1, +1\}$. Assuming linearly separable classes, there exists a separating hyperplane such that

$$y_k(\mathbf{w}^T\mathbf{x}^k + b) > 0, \qquad k = 1, \dots, m. \tag{1}$$

The distance between the separating hyperplane and the closest data points is the *margin of separation*. The goal of a SVM is to maximize this margin. We can rescale the weights $\mathbf{w}$ and the bias b so that the constraints (1) can be rewritten as [10]

$$y_k(\mathbf{w}^T\mathbf{x}^k + b) \geqslant 1, \qquad k = 1, \dots, m \tag{2}$$

As a consequence, the margin of separation is $1/\|\mathbf{w}\|$ and the maximization of the margin is equivalent to the minimization of the Euclidean norm of the weight vector $\mathbf{w}$.[1] The corresponding weights and bias represent the *optimal separating hyperplane*

[1]For convenience, we consider the margin between the closest data points and the separating hyperplane $(1/\|\mathbf{w}\|)$ instead of the usual margin of separation between classes $(2/\|\mathbf{w}\|)$.

Manuscript received May 24, 2005; revised October 27, 2005.

D. Casali and G. Costantini are with the Department of Electronic Engineering, University of Rome "Tor Vergata," 00100 Rome, Italy (e-mail: daniele.casali@uniroma2.it; costantini@uniroma2.it).

R. Perfetti and E. Ricci are with the Department of Electronic and Information Engineering, University of Perugia, I-06125 Perugia, Italy (e-mail: perfetti@diei.unipg.it; elisa.ricci@diei.unipg.it).

Digital Object Identifier 10.1109/TNN.2006.877539

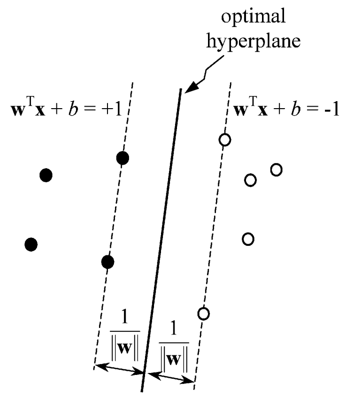

optimal hyperplane
$\mathbf{w}^T\mathbf{x} + b = +1$
$\mathbf{w}^T\mathbf{x} + b = -1$
$\frac{1}{\|\mathbf{w}\|}$
$\frac{1}{\|\mathbf{w}\|}$

Fig. 1. Optimal separating hyperplane corresponding to the SVM solution. The SVs lie on the dashed lines.

(Fig. 1). The data points $\mathbf{x}^k$ for which the constraints (2) are satisfied with the equality sign are called *support vectors*. Introducing the Lagrange multipliers $\alpha_1 \ldots \alpha_m$, the minimization of $\|\mathbf{w}\|^2$ under constraints (2) can be recast in the following dual form [10]: Find the minimum of

$$J(\boldsymbol{\alpha}) = -\sum_{k=1}^{m} \alpha_k + \frac{1}{2}\sum_{k=1}^{m}\sum_{j=1}^{m} \alpha_k \alpha_j y_k y_j (\mathbf{x}^k)^T \mathbf{x}^j \tag{3}$$

subject to the linear constraints

$$\sum_{k=1}^{m} \alpha_k y_k = 0 \tag{4}$$

$$\alpha_k \geqslant 0, \qquad k = 1, \ldots, m. \tag{5}$$

Solving this QP problem, we obtain the optimum Lagrange multipliers α_k, one for each data point. Using the Lagrange multipliers we can compute the optimum weight vector $\mathbf{w}$ as follows:

$$\mathbf{w} = \sum_{k=1}^{m} \alpha_k y_k \mathbf{x}^k. \tag{6}$$

Only the Lagrange multipliers corresponding to support vectors are greater than zero, so the optimum weights depend uniquely on the support vectors (SVs).

For an SV $\mathbf{x}^i$, it is

$$\mathbf{w}^T \mathbf{x}^i + b = y_i$$

from which the optimum bias b can be computed. In practice, it is better to average the values obtained by considering the set of all support vectors

$$b = \frac{1}{\#\mathrm{SV}} \sum_{i \in \mathrm{SV}} \left(y_i - \sum_{k=1}^{m} \alpha_k y_k (\mathbf{x}^k)^T \mathbf{x}^i \right) \tag{7}$$

where expression (6) has been used.

SVM formulation can be extended to the case of nonseparable classes, introducing the slack variables ξ_k; optimal class separation can be obtained by

$$\min \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{k=1}^{m} \xi_k \tag{8}$$

$$\text{s.t.} \quad y_k(\mathbf{w}^T\mathbf{x}^k + b) \geqslant 1 - \xi_k, \qquad k = 1, \ldots, m \tag{9a}$$

$$\xi_k \geqslant 0 \quad k = 1, \ldots, m. \tag{9b}$$

In this case, the margin of separation is said to be *soft*. The constant $C > 0$ is user-defined and controls the tradeoff between the maximization of the margin and the minimization of classification errors on the training set. The dual formulation of the QP is the same as (3)–(5) with the only difference in the bound constraints, which now become:

$$0 \leqslant \alpha_k \leqslant C, \qquad k = 1, \ldots, m. \tag{10}$$

In this case, for the optimum Lagrange multipliers we have: $0 < \alpha_k < C$ for support vectors ($\xi_k = 0$); $\alpha_k = 0$ for points correctly classified with $\xi_k = 0$; $\alpha_k = C$ for points with $\xi_k > 0$ (are misclassified if $\xi_k > 1$). In the soft margin case (7) still holds, but the average is taken over all $\mathbf{x}^i$ such that $\alpha_i > 0$ (generalized support vectors).

Finally, the threshold b could be chosen *a priori*; in this case, the SVM can be trained by finding the minimum of

$$J(\boldsymbol{\alpha}) = -\sum_{k=1}^{m} (1-b)\alpha_k + \frac{1}{2}\sum_{k=1}^{m}\sum_{j=1}^{m} \alpha_k \alpha_j y_k y_j (\mathbf{x}^k)^T \mathbf{x}^j \tag{11}$$

subject to the bound constraints

$$0 \leqslant \alpha_k \leqslant C, \qquad k = 1, \ldots, m.$$

With respect to the general formulation, the equality constraint (4) disappears and it is easier to find the solution. From the optimum Lagrange multipliers, we obtain the weight vector $\mathbf{w}$ using (6). Note that even if the classes are linearly separable, it is not guaranteed that they can be separated by the hyperplane with the threshold chosen *a priori*.

III. SVM Design of GBSB Neural Networks

Let consider the generalized brain-state-in-a-box (GBSB) neural model described by the following difference equation[2]:

$$\mathbf{x}(t+1) = \mathbf{g}[\mathbf{x}(t) + \alpha(\mathbf{W}\mathbf{x}(t) + \mathbf{b})], \quad t = 0, 1, 2 \ldots \tag{12}$$

$\mathbf{x}(t) = [x_1(t) \ldots\ldots x_n(t)] \in [-1, +1]^n$ is the state vector; n is the number of neurons; $\mathbf{W} = [w_{ij}] \in \Re^{nxn}$ is the weight matrix assumed *nonsymmetric*; $\mathbf{b} \in \Re^n$ is the bias vector; α is

[2]We use the same symbol for the gain α in (12) and the Lagrange multipliers introduced in Section II, complying with the literature on GBSB networks and on SVMs.

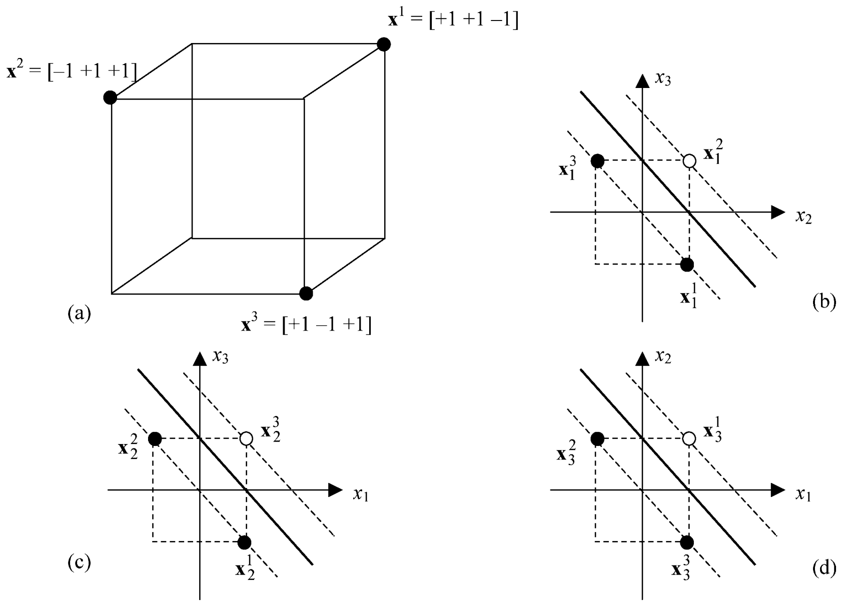


Fig. 2. Three bipolar patterns to be stored in the $\mathrm{AM}(\mathbf{x}^1 = [+1+1-1], \mathbf{x}^2 = [-1+1+1], \mathbf{x}^3 = [+1-1+1])$ and corresponding reduced vectors in $\Re^2$. The optimal separating hyperplane is shown.

a scalar; $\mathbf{g}$ is a vector valued function, whose ith component is defined as

$$\begin{aligned}(\mathbf{g}(\mathbf{y}))_i &= 1, \quad \text{if } y_i \geqslant 1\\ (\mathbf{g}(\mathbf{y}))_i &= y_i, \quad \text{if } -1 < y_i < +1\\ (\mathbf{g}(\mathbf{y}))_i &= -1, \quad \text{if } y_i \leqslant -1.\end{aligned}$$

Let $B = \{-1, +1\}$. The design of a bipolar associative memory, based on model (12), can be formulated as follows.

Synthesis Problem

Find the connection matrix $\mathbf{W}$ and bias vector $\mathbf{b}$ such that

1) a given set of bipolar vectors $\mathbf{x}^1 \ldots \mathbf{x}^m \in B^n$ are stored as asymptotically stable equilibrium states of system (12);
2) the basins of attraction of the desired equilibria are as large as possible;
3) the number of not desired (spurious) stable equilibria and the number of oscillatory solutions are as small as possible.

Several analysis results are available for GBSB neural networks. In the following, we review only the most significant of them (the proofs can be found in [6] in the case $\mathbf{b} = 0$; the extension to the present case is straightforward).

a) If $w_{ii} \geqslant 0$ for $i = 1, \ldots, n$, only the vertices of $[-1, +1]^n$ can be asymptotically stable equilibria of system (12).
b) $\mathbf{x} \in B^n$ is an asymptotically stable equilibrium point of system (12) if and only if

$$x_i \left(\sum_{j=1}^{n} w_{ij} x_j + b_i \right) > 0, \qquad i = 1, \ldots, n \qquad (13)$$

c) Assume that $\mathbf{x} \in B^n$ is an asymptotically stable equilibrium point of system (12). Then, no equilibrium point exists at Hamming distance one from $\mathbf{x}$ if $w_{ii} = 0$, for $i = 1, \ldots, n$.
d) Let $\mathbf{x} \in B^n$ denote an asymptotically stable equilibrium point of (12). Let $\mathbf{x}^* \in B^n$ differ from $\mathbf{x}$ in the components $x_i^*, i \in \{j_1 \ldots j_k\} \subset \{1 \ldots n\}$. If

$$x_i \left(\sum_{j=1}^{n} w_{ij} x_j^* + b_i \right) > 0 \qquad (14)$$

for at least one integer $i \in \{j_1 \ldots j_k\}$ then $\mathbf{x}^*$ is not an equilibrium point.

Constraints (13) represent existence and stability conditions for a given binary equilibrium point $\mathbf{x}$. We can design the associative memory by solving (13) with respect to the weights and bias for a given set of bipolar equilibrium points $\mathbf{x}^1 \ldots \mathbf{x}^m$. From properties a and c it follows that a zero-diagonal connection matrix guarantees the absence of equilibria with nonbinary components, and the absence of equilibria at Hamming distance one from the desired patterns. Taking into account the stability conditions (13) and the zero-diagonal constraint, the synthesis problem for the ith neuron can be recast as follows: Find $\mathbf{w}_i$ and $b_i, i = 1, \ldots, n$ such that

$$x_i^k \left(\boldsymbol{w}_i^T \mathbf{x}_i^k + b_i \right) > 0, \qquad k = 1, \ldots, m \qquad (15)$$

where $\mathbf{x}_i^k \in B^{n-1}$ is equal to $\mathbf{x}^k$ with the ith entry eliminated and $\mathbf{w}_i \in \Re^{n-1}$ is the ith row of matrix $\mathbf{W}$ with the ith entry eliminated.

Comparing expressions (1) and (15), the design of the ith neuron can be interpreted as a *classification* problem with training set given by $(\mathbf{x}_i^k, x_i^k), k = 1, \ldots, m$, where each reduced vector $\mathbf{x}_i^k$ is void of the component x_i^k and belongs to the class represented by $x_i^k \in \{-1, +1\}$. The synthesis problem can be solved by a set of n independent SVMs, where $\mathrm{SVM}(i), i = 1, \ldots, n$, computes the weight vector $\mathbf{w}_i$ and bias b_i for ith neuron. The method is illustrated in Fig. 2 in the

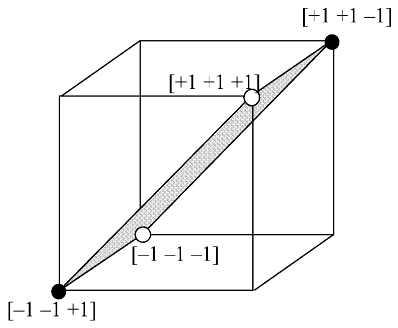


Fig. 3. Reduced vectors in $\Re^3$ which are not linearly separable.

three-dimensional case. The reduced vectors live in $\Re^2$. In this simple example all three classification problems admit the same margin and all patterns are support vectors for each machine.

A similar viewpoint was adopted by Liu and Lu [11] to design a continuous-time AM by n perceptrons trained by Rosenblatt's algorithm. Using SVMs we have a maximization of the margin of separation.

Some useful relations can be found between the properties of SVMs in Section II and the design of model (12).

Property 1: Given m vectors $\mathbf{x}^k \in B^n$, they can be stored as asymptotically stable states of system (12) with a zero-diagonal $\mathbf{W}$ if and only if, for every i, the corresponding reduced vectors $\mathbf{x}_i^k$ are linearly separable in $\Re^{n-1}$, where the classes are determined according to the ith element of the desired patterns. In particular, any two vectors $\mathbf{x}^r, \mathbf{x}^s$ must differ in at least two components.

Proof: The design constraints (15) can be interpreted as finding a separating hyperplane in $\Re^{n-1}$; so the given set of patterns admits a solution if and only if the reduced vectors are linearly separable. Moreover, if two prototypes $\mathbf{x}^r, \mathbf{x}^s$ differ in only one component, say the ith, the corresponding reduced vectors $\mathbf{x}_i^r, \mathbf{x}_i^s$ are identical but belong to different classes. In this case, we cannot separate them. □

Property 1 is illustrated by an example. Consider the following four patterns in $\Re^4$: $[+1+1-1+1], [+1+1+1-1], [-1-1-1-1], [-1-1+1+1]$, which differ in at least two positions. Deleting the last component we obtain the coplanar vectors $[+1+1-1], [+1+1+1], [-1-1-1], [-1-1+1]$ shown in Fig. 3, which cannot be linearly separated. Exploiting the geometrical properties introduced by Cover [12], we can say that $m \leqslant n$ bipolar vectors with n components can be stored with a zero-diagonal $\mathbf{W}$, apart from pathological cases (the reduced vectors lie on an hyperplane in $\Re^{n-1}$).

In general, there are no simple criteria to predict the linear separability of the given patterns. However, using SVMs this condition can be simply checked. It is sufficient to solve the QP with a very large value for C: If $\alpha_i < C$ for every i the given patterns are linearly separable. If at least one α_i saturates, linear separation is not possible; in this last case, we can store a subset of the given patterns using the soft margin approach, as explained in Section IV.

Property 2: The weights of the GBSB neural network computed using SVMs follow a *generalized Hebb's law*.

Proof: Using (6) with $y_k = x_i^k$ we have the following expression for the weights:

$$w_{ij} = \sum_{k=1}^{m} \alpha_i^k x_i^k x_j^k, \qquad i \neq j \tag{16}$$

where α_i^k is the Lagrange multiplier corresponding to the ith neuron and the kth prototype vector. Expression (16) corresponds to the Hebb's law where each product $x_i^k x_j^k$ is multiplied by a real constant $\alpha_i^k \geqslant 0$. Note that a subset of the Lagrange multipliers will be zero (corresponding to non-support vectors), hence not all the products $x_i^k x_j^k$ will be present in (16). □

Often the design of model (12) is formulated as follows [6], [7]: Find $\mathbf{w}_i, b_i, i = 1, \ldots, n$ such that

$$x_i^k \left(w_i^T \mathbf{x}_i^k + b_i \right) \geqslant \delta > 0, \qquad k = 1, \ldots, m \tag{17}$$

where δ is called *margin of stability*. Since the weight vector $\mathbf{w}_i$ can be rescaled to obtain any value for δ, a limitation or normalization of the weights is required in this formulation (e.g., $\|\mathbf{w}_i\| = 1$). Increasing δ we generally obtain less spurious stable states with larger basins of attraction. This property can be justified exploiting property d of the GBSB model. Assume $\mathbf{x} \in B^n$ is a stable equilibrium point, and $\mathbf{x}^*$ a random bipolar pattern with the constraint that it differs from $\mathbf{x}$ in ith component. Assuming $w_{ii} = 0$, condition (14) can be rewritten as follows:

$$\sum_{j \neq i} w_{ij} x_i x_j^* + b_i x_i = \sum_{j \neq i} w_{ij} x_i x_j + b_i x_i + \sum_{j \neq i} w_{ij} x_i \Delta x_j > 0 \tag{18}$$

where $\Delta x_j (j \neq i)$ are random variables with values 0, +2 and −2. From ith condition (13), we have

$$\sum_{j \neq i} w_{ij} x_i x_j + b_i x_i = \left| \mathbf{w}_i^T \mathbf{x} + b_i \right| .$$

Hence, (18) becomes

$$\sum_{j \neq i} w_{ij} x_i \Delta x_j > - \left| \mathbf{w}_i^T \mathbf{x} + b_i \right| . \tag{19}$$

Let us define $P_i(\mathbf{x})$ as the probability that (19) is true

$$P_i(\mathbf{x}) = \text{prob} \left\{ y_i > - \left| \mathbf{w}_i^T \mathbf{x} + b_i \right| \right\}$$

where $y_i = \sum_{j \neq i} w_{ij} x_i \Delta x_j$.

Assume $\Delta x_j (j \neq i)$ are identically distributed independent random variables, with zero mean and variance σ^2. If n is large the random variable y_i becomes normally distributed with zero mean and variance

$$\sigma_i^2 = \sigma^2 \sum_{j \neq i} w_{ij}^2 .$$

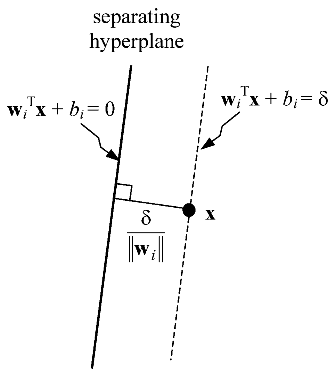

separating hyperplane
$\mathbf{w}_i^T\mathbf{x} + b_i = 0$
$\mathbf{w}_i^T\mathbf{x} + b_i = \delta$
$\frac{\delta}{\|\mathbf{w}_i\|}$
$\mathbf{x}$

Fig. 4. Relation between margin of stability and distance from the separating hyperplane.

Thus, we can write

$$P_i(\mathbf{x}) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{|\mathbf{w}_i^T\mathbf{x} + b_i|}{\|\mathbf{w}_i\|\sigma\sqrt{2}}\right)\right] \quad (20)$$

where erf denotes the error function. Increasing the ratio $|\mathbf{w}_i^T\mathbf{x} + b_i|/\|\mathbf{w}_i\|$, the probability that points different from the desired patterns are unstable will increase; this fact favours larger basins of attraction improving the error correction ability.

Expression (20) has a simple geometrical interpretation from the classification viewpoint. The ratio $|\mathbf{w}_i^T\mathbf{x} + b_i|/\|\mathbf{w}_i\|$ represents the distance between $\mathbf{x}$ and the ith separating hyperplane [10]; thus, $\delta/\|\mathbf{w}_i\|$ is the distance between the ith separating hyperplane and the closest point (the *margin of separation*) (Fig. 4). Using SVMs we minimize $\|\mathbf{w}_i\|$ s.t.

$$x_i^k\left(\mathbf{w}_i^T\mathbf{x}_i^k + b_i\right) \geqslant 1, \qquad i = 1, \ldots, n. \quad (21)$$

Hence, we can state the following property.

Property 3: The SVM solution of the synthesis problem corresponds to the maximum margin of separation with normalized (unitary) stability margin and minimum weight vector norm. □

Remark 1: Using (21) we can easily take into account the effects of parameter quantizations. Assume the weight vector $\mathbf{w}_i$ is affected by a rounding error $\Delta\mathbf{w}_i$; in order to preserve the stability of $\mathbf{x}_i^k$ condition (13) becomes

$$x_i^k\left((\mathbf{w}_i + \Delta\mathbf{w}_i)^T\mathbf{x}_i^k + b_i\right) > 0$$

i.e.

$$x_i^k\left(\mathbf{w}_i^T\mathbf{x}_i^k + b_i\right) > -x_i^k\Delta\mathbf{w}_i^T\mathbf{x}_i^k.$$

Using the worst case approach and taking into account (21), we have

$$(n-1)|\Delta w_{ij}|_{\max} < 1.$$

Assume that the weights are represented in digital form with $B+1$ bits, the previous condition becomes

$$(n-1)2^{-(B+1)} < 1.$$

The required number of bits to represent the weights is easily obtained

$$B + 1 > \log_2(n-1).$$

It is worth noting that this number depends only on the memory size and *not on the stored patterns*.

Remark 2: A second application of (21) is a preselection of the bias b_i. Using SVMs, as explained above, some of the negatives of the stored patterns $-\mathbf{x}^1 \ldots -\mathbf{x}^m$ could be stable states (attractors) of the AM. To avoid this problem, the values b_i can be computed as follows. The vector $-\mathbf{x}^k$ cannot be a stable state if $-x_i^k(-\mathbf{w}_i^T\mathbf{x}_i^k + b_i) < 0$, for *at least one neuron i.* Assume that $\mathbf{x}_i^k$ is an SV for the ith SVM, hence

$$x_i^k\left(\mathbf{w}_i^T\,\mathbf{x}_i^k + b_i\right) = 1 \quad (22)$$

Replacing $\mathbf{x}^k$ with $-\mathbf{x}^k$ in (22) gives

$$\begin{aligned}-x_i^k\left(-\mathbf{w}_i^T\mathbf{x}_i^k + b_i\right) &= x_i^k\left(\mathbf{w}_i^T\mathbf{x}_i^k + b_i\right) - 2x_i^k b_i \\ &= 1 - 2x_i^k b_i.\end{aligned}$$

To destabilize $-\mathbf{x}^k$, it must be

$$-x_i^k\left(-\mathbf{w}_i^T\mathbf{x}_i^k + b_i\right) < 0$$

hence it is sufficient that $b_i > 1/2$ if $x_i^k = +1$ or $b_i < -1/2$ if $x_i^k = -1$. We suggest the following expression for the bias:

$$b_i = (0.5 + \varepsilon)\operatorname{sign}\left(\sum_{\ell=1}^{m} x_i^\ell\right) \quad (23)$$

where ε is a small positive number. With this bias value $-\mathbf{x}^k$ will be unstable for neuron i if

$$x_i^k = \operatorname{sign}\left(\sum_{\ell=1}^{m} x_i^\ell\right). \quad (24)$$

Using (23) the sign of b_i is selected according to the majority of $+1$ or -1 in position i. In this way, we maximize the number of prototypes whose negatives are destabilized. This method does not guarantee that all the negatives are unstable, since it may happen that $\mathbf{x}_i^k$ is an SV but condition (24) is not fulfilled. However, as evidenced by computer experiments, if we use n SVMs to store $m \leqslant n$ vectors, each $\mathbf{x}^k$ corresponds to a support vector for most of the machines. So the probability is high that the negative is destabilized for some i. Note that expression (23) is in accordance with the method suggested in [5], [7] to bias the trajectory toward one of the desired patterns.

Remark 3: No convergence results are known for model (12) with nonsymmetric matrix $\mathbf{W}$. However, the simulation results show that there is a strict relation between the value of α in (12) and the magnitude of the weights. If we increase the magnitude of the weights, a correspondingly lower value of α should be used in order to have a convergent dynamics for most initial conditions. This behavior is well known for symmetric weight matrices; in this case convergence is guaranteed if $\alpha < 2/|\lambda_{\min}(\mathbf{W})|$ where $\lambda_{\min}(\mathbf{W})$ represents the minimum real eigenvalue of $\mathbf{W}$; since the diagonal elements of $\mathbf{W}$ are zero, $\lambda_{\min}$ is negative and its magnitude increases with the norm of $\mathbf{W}$. Using SVMs, we obtain the minimum norm weight vector $\mathbf{w}_i$ satisfying the constraints, for every i; this property allows a larger α for a convergent behaviour.

As concerns the practical implementation of the proposed design method, we can use one of the computationally efficient packages available to solve the SVM learning problem, e.g., SVM$^{\text{light}}$ [13] and Libsvm [14]. In particular, Libsvm, based on Platt's sequential minimal optimization (SMO) [15], can successfully handle classification problems with large amounts of data points (10^3–10^5). An evaluation of the run time for the SMO algorithm can be found in [15], where some benchmark problems are solved using a 266-MHz Pentium II processor. For linear SVMs, as those considered in this paper, the computation time is of the order of few seconds for problems with less than 5000 data points. Moreover, the computation time scales between $\sim m$ and $\sim m^{2.2}$ with the number m of data points. Finally, also some specialized digital hardware has been realized [16].

IV. Experimental Results

In this section, we present some design examples and simulation results in order to compare the SVM design with two existing methods: The perceptron learning algorithm [11] and the designer neural network [7]. Among the many existing techniques for AM design, we have chosen these for three reasons: They give nonsymmetric $\mathbf{W}$ matrices, as the present method; they are based on a constraint satisfaction formulation of the synthesis problem, as the present method; their performance is good as concerns capacity and error correction ability.

In all the experiments presented here, SVM training was carried out using SVM$^{\text{light}}$ [13] with a Matlab interface. The run time was negligible since m (the number of desired patterns) is quite small. In Examples 1 and 2 the desired patterns are linearly separable; the SVM solution was obtained using $C = 10^2$ and the values of the Lagrange multipliers were comprised between 0 and 2. In Example 3 we considered different values for C. As concerns the simulation of (12), in all cases the value $\alpha = 0.3$ was used, both for sake of comparison with [7] and because the networks synthesized by the perceptron learning algorithm often oscillate for larger values of α.

Example 1: In this example, we compare the SVM solution with the design obtained using a perceptron trained by Rosenblatt's algorithm [11]. With this algorithm we must select a learning rate η which affects the performance of the designed AM. We tried different values for η; the margins of separation are comparable for $0.5 < \eta \leqslant 1$; for $\eta \leqslant 0.5$ the margin can be very small for some patterns, worsening the error correction ability. So, in the following, we present only the results obtained with $\eta = 1$. This choice presents the additional advantage of integer connection weights [11].

For this test, ten training sets were randomly generated; each set is composed of $m = 5$ patterns with size $n = 10$ ($m = 0.5n$). The minimum Hamming distance between patterns is two. For each set, we found the weight matrix and bias vector using the SVMs and the perceptron learning algorithm. Both methods always stored all the five vectors as asymptotically stable states. The average number of spurious stable states was 4 with the SVMs and 5.7 with the perceptron learning algorithm. Then we simulated the dynamic behavior of the GBSB network starting from initial states with Hamming distance H from each of the stored prototype vectors. In Table I the results are summarized; they are based on ensemble averages over ten training sets. The first row represents the total number of vectors at Hamming distance H from a stored pattern, given by $\binom{10}{H}$. The remaining rows represent the average number of initial states at Hamming distance H from one of the stored patterns $\mathbf{x}^k$, converging to $\mathbf{x}^k$.

TABLE I
Example 1: Basins of Attraction

	H = 1	H = 2	H = 3	H = 4
# vectors at distance H	10	45	120	210
# recalled vectors (SVM)	9.9	38.5	63.6	46.3
# recalled vectors(perceptron)	8.4	26	44.6	39

TABLE II
Example 1: Convergence Test for 1024 Initial States

convergence to	SVM	perceptron
closest stored pattern	809	556
stored pattern (not the closest)	64	173
spurious stable state	144	250
not convergent	7	45

To gain a deeper insight into the behavior of the designed associative memory, we simulated the dynamic evolution starting from all the 1024 bipolar initial states. Then we counted: The number of initial states converging to the closest stored pattern (in Hamming sense); the number of initial states converging to one of the stored patterns but not to the closest one; the number of initial states converging to binary spurious states and, finally, those corresponding to oscillatory solutions (convergence is not guaranteed since $\mathbf{W}$ is nonsymmetric). These results are shown in Table II. Using SVMs, we obtain less trajectories converging to spurious stable states or not converging at all.

We examined also the capacity measure proposed in [18], based on the concept of *recall accuracy* (RA) for a given *Hamming radius* ρ. $\mathrm{RA}(\rho)$ represents the percentage of initial states at Hamming distance ρn $(0 < \rho < 1)$ from a stored pattern, converging to it. For example, in Table I $H = 1$ corresponds to $\rho = 0.1$; since in B^{10} there are ten vectors with $H = 1$ from a given prototype, the recall accuracy is 99% with the SVM and 84% with the perceptron. The value $H = 2$ corresponds to $\rho = 0.2$; taking into account that there are 45 vectors at $H = 2$, the recall accuracy is 86% with the SVM and 58% with the perceptron. For sake of comparison we consider the results in [17], based on the Ho–Kashyap method [18]: They are $\mathrm{RA} > 95\%$ for $\rho = 0.1$ and $m < 0.3n$; $\mathrm{RA} > 80\%$ for $\rho = 0.2$ and $m < 0.25n$.

Example 2: Here, we consider a specific design example and compare four different design strategies: SVM standard, SVM with fixed threshold according to (23), perceptron, and the designer neural network of Chan and Zak. Let us consider the five patterns with $n = 10$ shown at the bottom of the page, to be stored in the associative memory (it is the same example considered in [6] and [7]).

The weight matrix and bias vector are given in (25), as shown at the bottom of the page. All the prototype vectors are stored, along with five spurious states. The same problem was solved computing the bias by (23) with $\varepsilon = 0.01$. In this case, all the vectors were stored, with no spurious states. The weight matrix and bias vector are given in (26), as shown at the bottom of the page. Then we solved the design problem using the perceptron learning algorithm, with $\eta = 1$: all the prototypes were stored, with seven spurious stable states. The solution is shown in (27)

$$\begin{array}{rrrrrrrrrr}
0 & -1 & -3 & -3 & -3 & 1 & 1 & -1 & -1 & 1\\
-2 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 4 & 2\\
-2 & 0 & 0 & 2 & -4 & -2 & 0 & 0 & 0 & -2\\
-3 & 1 & 1 & 0 & -1 & 1 & -1 & 1 & 1 & -1\\
-3 & -1 & -3 & -3 & 0 & 1 & 1 & -1 & -1 & 1\\
1 & -3 & -3 & 1 & 1 & 0 & -3 & 3 & -3 & -1\\
2 & 0 & 0 & -2 & 0 & -2 & 0 & -4 & 0 & -2\\
-2 & 0 & 0 & 2 & 0 & 2 & -4 & 0 & 0 & 2\\
-2 & 4 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 2\\
0 & 2 & -2 & 0 & 2 & 0 & -2 & 2 & 2 & 0
\end{array} \tag{27a}$$

$$\begin{array}{rrrrrrrrrr}
1 & 2 & -2 & 1 & 1 & 1 & 2 & -2 & 2 & 0.
\end{array} \tag{27b}$$

The designer neural network stores the desired vectors without spurious states (**W** and **b** in [7, p. 366]).

$$\begin{aligned}
\mathbf{x}^1 &= [-1 \quad +1 \quad -1 \quad +1 \quad +1 \quad +1 \quad -1 \quad +1 \quad +1 \quad +1]^T\\
\mathbf{x}^2 &= [+1 \quad +1 \quad -1 \quad -1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1 \quad +1]^T\\
\mathbf{x}^3 &= [-1 \quad +1 \quad +1 \quad +1 \quad -1 \quad -1 \quad +1 \quad -1 \quad +1 \quad -1]^T\\
\mathbf{x}^4 &= [+1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1 \quad +1 \quad +1]^T\\
\mathbf{x}^5 &= [+1 \quad -1 \quad -1 \quad -1 \quad +1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1]^T.
\end{aligned}$$

$$\begin{array}{llllllllll}
0 & -0.205 & -0.692 & -0.821 & -1.000 & 0.077 & 0.128 & -0.128 & -0.205 & 0.487\\
-0.080 & 0 & 0.050 & 0.110 & -0.080 & -0.310 & -0.060 & 0.060 & 0.370 & 0.320\\
-0.262 & 0.048 & 0 & 0.194 & -0.262 & -0.185 & 0.136 & -0.136 & 0.049 & -0.282\\
-0.344 & 0.118 & 0.215 & 0 & -0.344 & 0.140 & -0.258 & 0.258 & 0.118 & -0.097\\
-1.000 & -0.205 & -0.692 & -0.821 & 0 & 0.077 & 0.128 & -0.128 & -0.205 & 0.487\\
0.041 & -0.419 & -0.257 & 0.178 & 0.041 & 0 & -0.432 & 0.432 & -0.419 & -0.162\\
0.051 & -0.061 & 0.141 & -0.242 & 0.051 & -0.323 & 0 & -0.384 & -0.061 & -0.202\\
-0.051 & 0.061 & -0.141 & 0.242 & -0.051 & 0.323 & -0.384 & 0 & 0.061 & 0.202\\
-0.080 & 0.370 & 0.050 & 0.110 & -0.080 & -0.310 & -0.060 & 0.060 & 0 & 0.320\\
0.250 & 0.421 & -0.382 & -0.118 & 0.250 & -0.158 & -0.263 & 0.263 & 0.421 & 0
\end{array} \tag{25a}$$

$$\begin{array}{llllllllll}
0.231 & 0.440 & -0.553 & 0.419 & 0.231 & 0.703 & 0.414 & -0.414 & 0.440 & -0.474
\end{array} \tag{25b}$$

$$\begin{array}{llllllllll}
0 & -0.308 & -0.553 & -0.895 & -1.000 & -0.035 & 0.035 & -0.035 & -0.308 & 0.553\\
-0.085 & 0 & 0.085 & 0.085 & -0.085 & -0.333 & -0.085 & 0.085 & 0.333 & 0.333\\
-0.274 & 0.028 & 0 & 0.180 & -0.274 & -0.216 & 0.119 & -0.119 & 0.028 & -0.272\\
-0.340 & 0.075 & 0.258 & 0 & -0.340 & 0.089 & -0.286 & 0.286 & 0.075 & -0.061\\
-1.000 & -0.308 & -0.553 & -0.895 & 0 & -0.035 & 0.035 & -0.035 & -0.308 & 0.553\\
0.067 & -0.366 & -0.423 & 0.290 & 0.067 & 0 & -0.389 & 0.389 & -0.366 & -0.267\\
0.044 & -0.100 & 0.178 & -0.267 & 0.044 & -0.345 & 0 & -0.345 & -0.100 & -0.178\\
-0.044 & 0.100 & -0.178 & 0.267 & -0.044 & 0.345 & -0.345 & 0 & 0.100 & 0.178\\
-0.085 & 0.333 & 0.085 & 0.085 & -0.085 & -0.333 & -0.085 & 0.085 & 0 & 0.333\\
0.250 & 0.333 & -0.250 & -0.250 & 0.250 & -0.333 & -0.755 & 0.755 & 0.333 & 0
\end{array} \tag{26a}$$

$$\begin{array}{llllllllll}
0.51 & 0.51 & -0.51 & 0.51 & 0.51 & 0.51 & 0.51 & -0.51 & 0.51 & 0.51
\end{array} \tag{26b}$$

TABLE III
EXAMPLE 2: BASINS OF ATTRACTION—STANDARD SVM

	H = 1	H = 2	H = 3	H = 4
$\mathbf{x}^1$	8	30	46	30
$\mathbf{x}^2$	10	39	72	42
$\mathbf{x}^3$	10	41	72	71
$\mathbf{x}^4$	10	36	43	33
$\mathbf{x}^5$	10	41	75	71
TOT	48	187	308	247

TABLE IV
EXAMPLE 2: BASINS OF ATTRACTION—SVM WITH BIAS GIVEN BY (23)

	H = 1	H = 2	H = 3	H = 4
$\mathbf{x}^1$	9	32	52	53
$\mathbf{x}^2$	10	40	69	67
$\mathbf{x}^3$	10	42	74	54
$\mathbf{x}^4$	9	40	72	48
$\mathbf{x}^5$	10	42	84	70
TOT	48	196	351	292

TABLE V
EXAMPLE 2: BASINS OF ATTRACTION—PERCEPTRON

	H = 1	H = 2	H = 3	H = 4
$\mathbf{x}^1$	9	31	39	29
$\mathbf{x}^2$	10	38	63	52
$\mathbf{x}^3$	10	38	59	42
$\mathbf{x}^4$	7	9	9	8
$\mathbf{x}^5$	10	44	89	86
TOT	46	160	259	217

TABLE VI
EXAMPLE 2: BASINS OF ATTRACTION—DESIGNER NEURAL NETWORK

	H = 1	H = 2	H = 3	H = 4
$\mathbf{x}^1$	9	32	62	62
$\mathbf{x}^2$	10	39	81	79
$\mathbf{x}^3$	10	44	80	39
$\mathbf{x}^4$	8	31	60	60
$\mathbf{x}^5$	10	41	59	40
TOT	47	187	342	280

Then, we simulated the dynamic behaviour of the GBSB network starting from the initial states with Hamming distance $1 \leqslant H \leqslant 4$, from each of the stored prototype vectors. The results are shown in Tables III–VI, where the entry at intersection of row $\mathbf{x}^k$ and column H represents the number of initial states at Hamming distance H from $\mathbf{x}^k$ converging to $\mathbf{x}^k$. Finally, we simulated the dynamic evolution starting from all the 1024 bipolar initial states. These results are shown in Table VII.

From the simulation results, we see that standard SVMs give larger basins of attraction and less spurious states with respect to the perceptron learning algorithm. Using SVMs with the bias computed by (23) we obtain results comparable to those of Chan and Zak [7]. As concerns the convergence speed, the data in Table VII confirm the comments in Remark 3 of Section III. The networks synthesized using SVMs have a minimum norm weight vector; as a consequence the dynamic evolution is slower but the chance of oscillation is rare.

An important advantage of the proposed approach is that standard software routines for SVMs can be used without the need for special-purpose analog circuits, as those proposed in [7], [8]. Finally, there are no design parameters to be tuned. This is a major difference with respect to both the perceptron, where the learning rate η can influence the results, and to the designer neural network, where some circuit parameters must be selected (input voltage source, slope of the piecewise-linear amplifiers, stability margin).

Example 3: In this example, we consider $m > n$ vectors and try to store them in the AM. Even when a solution to the problem does not exist, using SVMs we can store a subset of the given prototype vectors, trading off the number of successfully stored patterns and their recall accuracy. This is possible by using the soft margin constraints (10). Varying the penalty parameter C we can find several solutions with different stored patterns and margin of separation. To verify this property, we consider the following 12 vectors of size $n = 10$

$$\begin{array}{l}
[-1\ \ -1\ \ 1\ \ -1\ \ -1\ \ 1\ \ -1\ \ -1\ \ -1\ \ 1] \\
[-1\ \ -1\ \ -1\ \ -1\ \ -1\ \ 1\ \ -1\ \ -1\ \ 1\ \ -1] \\
[-1\ \ 1\ \ 1\ \ 1\ \ -1\ \ -1\ \ -1\ \ 1\ \ 1\ \ 1] \\
[\ 1\ \ -1\ \ -1\ \ 1\ \ -1\ \ 1\ \ -1\ \ -1\ \ -1\ \ -1] \\
[-1\ \ 1\ \ 1\ \ -1\ \ -1\ \ 1\ \ -1\ \ 1\ \ -1\ \ -1] \\
[-1\ \ 1\ \ 1\ \ -1\ \ 1\ \ 1\ \ -1\ \ -1\ \ 1\ \ 1] \\
[-1\ \ 1\ \ -1\ \ 1\ \ -1\ \ -1\ \ -1\ \ 1\ \ -1\ \ -1] \\
[-1\ \ -1\ \ -1\ \ -1\ \ -1\ \ -1\ \ 1\ \ -1\ \ 1\ \ -1] \\
[-1\ \ -1\ \ -1\ \ 1\ \ 1\ \ 1\ \ -1\ \ -1\ \ 1\ \ 1] \\
[-1\ \ 1\ \ 1\ \ 1\ \ 1\ \ -1\ \ 1\ \ -1\ \ 1\ \ -1] \\
[\ 1\ \ -1\ \ 1\ \ -1\ \ -1\ \ -1\ \ 1\ \ 1\ \ 1\ \ 1] \\
[\ 1\ \ -1\ \ -1\ \ 1\ \ 1\ \ -1\ \ -1\ \ -1\ \ -1\ \ 1].
\end{array} \tag{28}$$

The minimum Hamming distance between patterns is two. We computed the weight matrix and bias vector using several values of C (the same C is used for each SVM). In Table VIII, some design examples are summarized. They correspond to the largest number of stored patterns; for the values of C not shown in Table VIII we obtained less than seven stored vectors out of twelve. Table VIII shows: The number of successfully stored patterns, the RA for $H = 1$, the number of spurious stable states. The value $C = \inf$ denotes the positive infinity in Matlab; it corresponds to unbounded values for the Lagrange multipliers, as in (5).

Note that the twelve vectors can simultaneously be stored as stable states, i.e., they can be linearly separated by each neuron. This is obtained by using $C = \inf$. However this solution corresponds to the minimum recall accuracy. Note that RA decreases with C. This behaviour can easily be explained observing the QP (8) and (9). Larger values of C correspond to SVM solutions with reduced margin of separation, i.e., smaller attraction basins.

TABLE VII
EXAMPLE 2: CONVERGENCE TEST FOR 1024 INITIAL STATES

convergence to	standard SVM	SVM using exprn.(23)	perceptron	designer neural network
closest stored pattern	778	871	660	849
stored pattern (not the closest)	101	153	82	129
spurious stable state	145	0	238	0
not convergent	0	0	44	46
average number of cycles for convergence	28.7	31.45	6.67	17.38

TABLE VIII
EXAMPLE 3: VECTORS (28)

C	#stored vectors	RA (H=1)	#spurious states
inf	12	57%	8
$4\text{x}10^5$	7	60%	6
10^5	10	66%	5
10^4	10	67%	5
1	8	67%	6

TABLE IX
EXAMPLE 3: VECTORS (29)

C	#stored vectors	RA (H=1)	#spurious states
inf	9	55%	4
$2\text{x}10^5$	10	67%	8
10^5	9	71%	6
10^4	9	73%	7
1	8	87.5%	8

As a further example, we consider the following 12 vectors of size $n = 10$:

$$\begin{array}{l}
[-1\ \ 1\ \ 1\ -1\ -1\ -1\ \ 1\ -1\ \ 1\ \ 1]\\
[-1\ -1\ -1\ \ 1\ \ 1\ -1\ -1\ -1\ \ 1\ \ 1]\\
[\ 1\ \ 1\ -1\ -1\ \ 1\ \ 1\ \ 1\ -1\ -1\ \ 1]\\
[-1\ -1\ \ 1\ \ 1\ -1\ -1\ \ 1\ -1\ -1\ -1]\\
[\ 1\ \ 1\ -1\ -1\ \ 1\ -1\ -1\ \ 1\ \ 1\ \ 1]\\
[\ 1\ \ 1\ \ 1\ -1\ \ 1\ -1\ -1\ \ 1\ -1\ -1]\\
[\ 1\ \ 1\ -1\ \ 1\ -1\ -1\ \ 1\ \ 1\ -1\ \ 1]\\
[\ 1\ -1\ -1\ -1\ \ 1\ \ 1\ \ 1\ \ 1\ -1\ -1]\\
[\ 1\ \ 1\ \ 1\ -1\ -1\ -1\ \ 1\ -1\ \ 1\ -1]\\
[-1\ \ 1\ -1\ \ 1\ -1\ -1\ -1\ -1\ \ 1\ \ 1]\\
[\ 1\ -1\ -1\ -1\ -1\ \ 1\ \ 1\ -1\ -1\ \ 1]\\
[\ 1\ \ 1\ -1\ -1\ -1\ \ 1\ -1\ -1\ -1\ -1].
\end{array} \quad (29)$$

The minimum Hamming distance is two. In this case the twelve vectors cannot be simultaneously stored in the AM. At most ten vectors can be stored, with $C = 2 \times 10^5$. The results for some values of C are summarized in Table IX.

V. COMMENTS AND CONCLUSIONS

We showed how the design of a recurrent associative memory can be recast as a classification problem and efficiently solved using a pool of SVMs. We pointed out some properties of the associative memories designed in this way, as the generalized Hebb's law. The proposed approach presents some interesting features. First of all, if the design problem admits a solution, the proposed approach gives the "optimal" one, i.e., that with the largest margin of separation. As a consequence of the maximization of the margin, the recall accuracy of the stored patterns compares well with existing techniques for associative memory design. Moreover, the SVM solution allows a simple method to control the effect of parameter quantization and to avoid, with high probability, the storage of the negatives of the desired patterns. Further, the minimization of weight vector norm, implicit in the SVM solution, helps to obtain a convergent dynamic behaviour in the recall phase, even if the connection matrix is nonsymmetric. Another practical advantage of the proposed method is the absence of design parameters, at least in the usual case when a given set of (linearly separable) patterns must be stored. However, the SVM approach gives a further benefit: It allows to tradeoff capacity and recall accuracy in a simple way, by varying the design parameter C.

REFERENCES

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Natl. Acad. Sci.*, 1982, vol. 79, pp. 2554–2558.

[2] J. M. Zurada, *Introduction to Artificial Neural Systems*. St. Paul, MN: West, 1992.

[3] A. N. Michel and J. A. Farrell, "Associative memories via artificial neural networks," *IEEE Control Syst. Mag.*, vol. 10, no. 3, pp. 6–17, Apr. 1990.

[4] S. Hui and S. H. Zak, "Dynamical analysis of the brain-state-in-a-box (BSB) neural models," *IEEE Trans. Neural Netw.*, vol. 3, no. 1, pp. 86–100, Jan. 1992.

[5] W. E. Lillo, D. C. Miller, S. Hui, and S. H. Zak, "Synthesis of brain-state-in-a-box (BSB) based associative memories," *IEEE Trans. Neural Netw.*, vol. 5, no. 5, pp. 730–737, Sep. 1994.

[6] R. Perfetti, "A synthesis procedure for brain-state-in-a-box neural networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 5, pp. 1071–1080, Sep. 1995.

[7] H. Y. Chan and S. H. Zak, "On neural networks that design neural associative memories," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 360–372, Mar. 1997.

[8] R. Perfetti, "A neural network to design neural networks," *IEEE Trans. Circuits Syst.*, vol. 38, no. 9, pp. 1099–1103, Sep. 1991.

[9] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," in *Data Mining and Knowledge Discovery*. Norwell, MA: Kluwer, 1998, vol. 2, pp. 121–167.

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[11] D. Liu and Z. Lu, "A new synthesis approach for feedback neural networks based on the perceptron training algorithm," *IEEE Trans. Neural Netw.*, vol. 8, no. 6, pp. 1468–1482, Nov. 1997.

[12] T. M. Cover, “Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition,” *IEEE Trans. Electron. Comput.*, vol. EC-14, no. 3, pp. 326–334, Jun. 1965.

[13] T. Joachims, “Making large scale SVM learning practical,” in *Advances in Kernel Methods-Support Vector Learning*, B. Scholkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 169–184 [Online]. Available: http://svmlight.joachims.org/

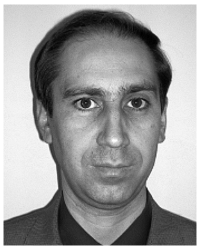
[14] C.-C. Chang and C.-J. Lin, LIBSVM—A library for support vector machines [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[15] J. C. Platt, “Fast training of support vector machines using sequential minimal optimization,” in *Advances in Kernel Methods—Support Vector Learning*, B. Scholkopf, C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 185–208.

[16] R. Genov and G. Cauwenberghs, “Kerneltron: Support vector machine in silicon,” *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1426–1434, Sep. 2003.

[17] M. H. Hassoun and A. M. Youssef, “Associative neural memory capacity and dynamics,” in *Proc. Int. J. Conf. Neural Networks*, San Diego, CA, 1990, vol. 1, pp. 736–769.

[18] M. H. Hassoun and A. M. Youssef, “High performance recording algorithm for Hopfield model associative memories,” *Opt. Eng.*, vol. 28, no. 1, pp. 46–54, 1989.



Giovanni Costantini was born in Italy in 1965. He received the electronic engineering Laurea degree from the University of Rome “La Sapienza,” Italy, and the Ph.D. degree in telecommunication and microelectronics from the University of Rome “Tor Vergata,” Italy, in 1991 and 1999, respectively. He also graduated in piano and electronic music from the Music Conservatory, Italy.

Currently, he is an Assistant Professor at the University of Rome, “Tor Vergata.” His primary research interests are in the fields of neural networks, pattern recognition, algorithms, and systems for audio and musical signal processing.

Renzo Perfetti was born in Italy in 1958. He received the Laurea degree (with honors) in electronic engineering from the University of Ancona, Italy, and the Ph.D. degree in information and communication engineering from the University of Rome, “La Sapienza,” Italy, in 1982 and 1992, respectively.

From 1983 to 1987, he was with the radar division of Selenia, Rome, Italy, where he worked on radar systems design and simulation. From 1987 to 1992, he was with the radiocommunication division of Fondazione Bordoni, Rome, Italy. In 1992, he joined the Department of Electronic and Information Engineering of the University of Perugia, Italy, where he is currently a Full Professor of electrical engineering. His current research interests include artificial neural networks, pattern recognition, machine learning, and digital signal processing.

Daniele Casali was born in Italy in 1974. He received the M.S. degree in electronic engineering and the Ph.D. degree in sensory and learning systems, both from the University of Rome “Tor Vergata,” Italy, in 2000 and 2004, respectively.

Currently, he is a Postdoctoral Research Assistant at the University of Rome, and has teaching assignments at the Faculties of Engineering and Science at the same university. His research interests include neural networks, image, and audio signal processing.

Elisa Ricci received the M.S. degree in electronic engineering from the University of Perugia, Italy, in 2003. She is currently working toward the Ph.D. degree in the Department of Electronic and Information Engineering, University of Perugia.

Her research interests include machine learning, neural networks, and image processing.