A Weighted Voting Model of Associative Memory: Theoretical Analysis

Xiaoyan Mu
Department of Electrical and Computer Engineering
Rose-Hullman Institute of Technology
Terre Haute, IN 47803
mu@Rose-Hulman.edu

Paul Watta
Dept. Electrical and Computer Engineering
University of Michigan-Dearborn
Dearborn, MI 48128
watta@umich.edu

Mohamad H. Hassoun
Dept. Electrical and Computer Engineering
Wayne State University
Detroit, MI 48202
hassoun@eng.wayne.edu

Abstract

In this paper we investigate a RAM-based associative memory that uses a weighted voting scheme. We adopt the testing protocols commonly used in the area of face recognition, and propose that the capacity of the system be measured by the results of an identification test (ability to properly recognize known information) and a watch-list test (ability to properly reject inputs that should not be matched with any of the memory set patterns). For the case of binary and random memory sets, we are able to derive theoretical expressions characterizing the performance of the weighted voting memory on both of these tests.

I. Introduction

The associative memory problem is stated as follows: We are given a *fundamental memory set* of desired associations: $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$, where $\mathbf{x}_i \in X \subset R^N$ and $\mathbf{y}_i \in Y \subset R^L$, $i = 1, 2, \ldots, m$. The task is to design a system which robustly stores the fundamental associations [3], such that

(1) When presented with $\mathbf{x}_i$ as input, the system should produce $\mathbf{y}_i$ at the output.

(2) When presented with a *noisy* (corrupted, distorted, or incomplete) version of $\mathbf{x}_i$ at the input, the system should also produce $\mathbf{y}_i$ at the output.

(3) When presented with input $\mathbf{x}$ that is *not sufficiently similar* (application dependent) to any of the inputs in the memory set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, the system should reject the input.

In Ikeda, Watta, Artiklar, and Hassoun, (2001), a voting-based model of associative memory was proposed and analyzed. This voting memory was shown to have a number of advantages over other neural net-based associative memories. In this paper, we generalize the voting scheme by allowing for each local processor to cast not just a single vote, but a set of weighted votes. The resulting memory model will be referred to as the *weighted voting memory.* In this paper, a theoretical analysis of the weighted voting memory is given.

The innovations of the voting memory developed here were inspired by the application of human face recognition [7, 8]. In this context, the fundamental memory set consists of a database of face images, where the input $\mathbf{x}_i$ is an image of the *i*th person, and the associated output $\mathbf{y}_i$ is the corresponding name of the person. The face recognition research community has developed protocols for assessing system performance [7, 8]. There are 2 main tests commonly used: *identification test* and *watch list test*. In both cases, we are given a database of face images. In the identification test, the system is tested with (new) images of known people. For a given input image, the task is to identify which database person it is. In this case, no rejection state is needed. The measure of merit here is the *identification rate* (IR), which is the probability that a given input image will be matched with the correct stored memory.

In the watch list test, the system must have a rejection mechanism. Here, two test sets are required: TS_G, which contains images of the known people, and TS_N, which contains images of strangers (people not in the database). For the TS_G test set, there are two measures of merit: the *detection and identification rate* (DIR), which is the percentage of images that are correctly matched with the known individuals, and the *false rejection rate* (FRR), which is the percentage of images that are rejected by the system [8]. For the TS_N test set, there is only one measure of interest: the *false acceptance rate* (FAR), which gives the percentage of imposter images that are incorrectly matched with someone in the database.

Of course there is a trade-off between the detection and identification rate and the false acceptance rate. Typically, face recognition systems are designed with a tunable parameter or threshold T which allows one to control the trade-off between DIR and FAR. A *receiver operating characteristic* (ROC) curve can be constructed which shows how DIR and FAR vary as a function of T.

We propose that researchers in neural associative memories adopt both the identification *and* watchlist testing methodology. In this paper, for memory sets consisting of random binary patterns, we are able to derive theoretical expressions for the retrieval rate, detection and identification rate, and the false acceptance rate for the proposed weighted voting model.

II. The Voting Associative Memory

In the voting associative memory, the N-dimensional input $\mathbf{x}$ and each memory pattern $\mathbf{x}_k$ are partitioned into a collection of non-overlapping windows of size n. For notional simplicity, we will assume that n divides N, hence N/n is an integer.

For the input (memory key), let $\mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[N/n]$ denote the data in each window. That is, $\mathbf{x}[i]$ is the portion of $\mathbf{x}$ contained in the ith window, etc. The database patterns are partitioned in the same way: $\mathbf{x}_k[1], \mathbf{x}_k[2], \ldots, \mathbf{x}_k[N/n]$, $k = 1, 2, \ldots, m$. The partitioned database patterns can be stored in a RAM-type network [5], where the ith RAM holds all the database patterns associated with the ith window: $\mathbf{x}_1[i], \mathbf{x}_2[i], \ldots, \mathbf{x}_M[i]$. Figure 1 shows the architecture of a RAM network with 9 windows arranged in a 3×3 structure.

The voting network requires a distance measure to be computed locally at each window. Let d be a distance measure between two n-dimensional vectors. Any suitable distance function can be used. For example, for binary memory patterns, the Hamming distance can be used; for real-valued patterns, the city-block distance can be used. In either case, the (local) distance between $\mathbf{x}_k[i]$ and $\mathbf{x}[i]$ is given by:

$$d(\mathbf{x}_k[i], \mathbf{x}[i]) = \sum_{j=1}^{n} |\mathbf{x}_k[i]_j - \mathbf{x}[i]_j|$$

where $\mathbf{x}_k[i]_j$ and $\mathbf{x}[i]_j$ denote the jth component of $\mathbf{x}_k[i]$ and $\mathbf{x}[i]$, respectively.

At each window, we compute a *local distance* $d_1, \ldots, d_m$ between the input key and all the memory patterns. The smallest distance is found, say d_k, and then the local window casts a vote for memory pattern k. The decision network examines the votes of all the windows and chooses the memory pattern that received the most votes.

It is easy to introduce a rejection mechanism in the voting model. We simply use a threshold T to indicate whether the number of votes received by the best matching pattern is sufficiently large. In the case that the number of votes received is less than T, then the input is rejected.

Note that the network structure of the voting memory is similar to the WISARD system of Aleksander [5]. However, the present system is formulated to work with general types of data, not just binary data. In addition, the present system uses regular connections between the input and the RAM units and not random connections. Finally, the voting-based method of information retrieval is not present in the WISARD system.

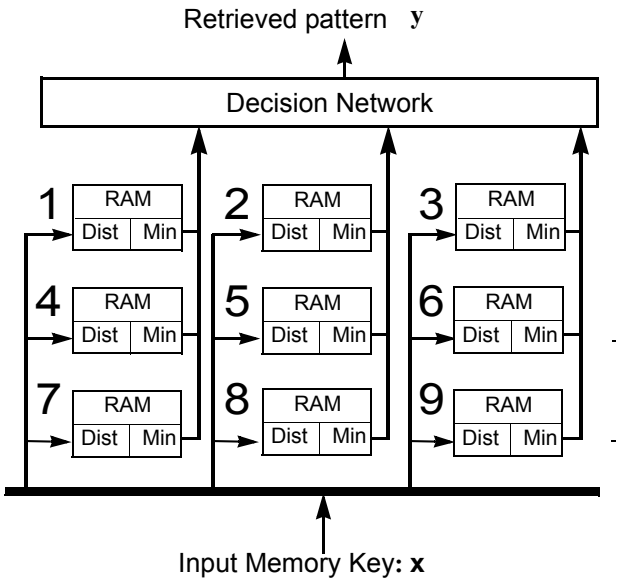

Figure 1. Structure of the voting associative memory.

It is interesting to note the 2-level structure of this associative memory network. The RAMs are low-level processors which operate on just a portion of the image. The decision network is a higher level computation which integrates and makes sense of the low-level information. Of course the problem of understanding the connection between low-level processing and high-level decision-making has long been an area of interest in both neurobiology and artificial intelligence [1, 2].

III. The Weighted Voting Memory

At the local level, the voting memory works in an all-or-nothing fashion [4]. That is, the memory pattern that has the smallest (local) distance gets a vote and all the other memory patterns get nothing. It possible, though, that for a noisy memory key, the target pattern may not appear first on the list of best matching patterns. In this case, the worst happens: the window casts a vote for a non-target pattern and the target pattern gets nothing.

The weighted voting model operates as follows. As before, we compute local distance measures at each window. But instead of just choosing the smallest distance and assigning a vote to the corresponding memory pattern, we sort all the distances and assign a rank to each. Let the memory set pattern that has the smallest (local) distance be assigned $rank = 1$. The pattern with the next smallest distance will have $rank = 2$, etc. The distance computations and ranking of memory set patterns are done independently by each local window. Hence the weighted voting model has the same parallel structure as the voting model shown in Figure 1. The only difference is that each window now requires a sorting operation and not a simple min select. After all the rankings have been computed, they are routed to the decision network. The decision network examines the rankings for each memory set pattern and then computes an appropriate output.

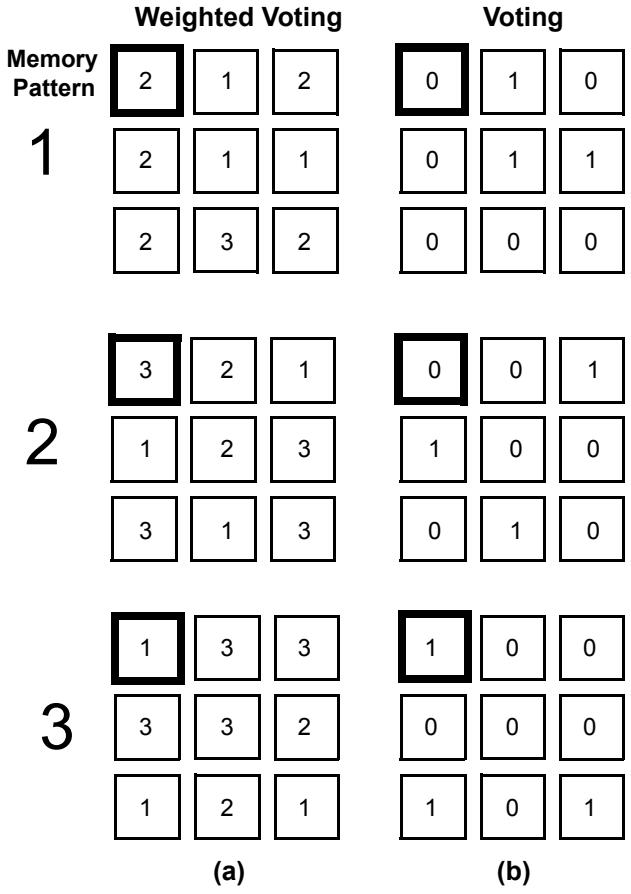

Figure 2. Example of a memory set with 3 patterns and a set of 3×3 local classifiers. The rankings assigned to each memory pattern by the weighted voting model are shown.

A simple example will help clarify the concepts. Suppose we have a memory set consisting of $m = 3$ patterns and suppose the patterns are partitioned into $N/n = 9$ windows (in a 3×3 arrangement). Suppose that for a given memory key, the local distances are computed and sorted, and the resulting rankings are as shown in Figure 2(a). For example, for the first window (highlighted in the Figure), the local distances d_1, d_2, and d_3 were found to satisfy $d_3 < d_1 < d_2$. Hence, for this window, memory set pattern #1 is assigned $rank = 2$, pattern #2 is assigned $rank = 3$, and pattern #3 is assigned $rank = 1$. The ranking from all the other windows are shown, as well.

The voting network can be seen as a special case of weighted voting, where we only consider the $rank = 1$ information, as shown in Figure 2(b). In the given example, though, each memory set pattern receives 3 votes, and hence the voting network cannot adequately discriminate among the memory set patterns. Clearly in this example, the second and third place rankings give additional information that would lead us to prefer memory set pattern #1 over the other two.

Now how does the decision network of the weighted voting memory operate? First, let's tally up the votes. Let $N_{\mathbf{x}_k}^{(1)}$ be the number of windows that were found to have rank 1, $N_{\mathbf{x}_k}^{(2)}$ the number of windows that have rank 2, etc. In general, let $N_{\mathbf{x}_k}^{(r)}$ denote the number of windows that have rank r, $r = 1, 2, \ldots, m$. We propose that for the weighted voting memory, the total number of votes $N_{\mathbf{x}_k}$ assigned to pattern $\mathbf{x}_k$ be a weighted sum of the number of windows at each ranking $N_{\mathbf{x}_k}^{(1)}, N_{\mathbf{x}_k}^{(2)}, \ldots, N_{\mathbf{x}_k}^{(m)}$ that $\mathbf{x}_k$ received:

$$N_{\mathbf{x}_k} = \alpha_1 N_{\mathbf{x}_k}^{(1)} + \alpha_2 N_{\mathbf{x}_k}^{(2)} + \ldots + \alpha_m N_{\mathbf{x}_k}^{(m)} \qquad (1)$$

where the weights $\alpha_1, \alpha_2, \ldots, \alpha_m$ are used to adjust the relative importance of each ranking. Note that the simple voting memory can be seen as a special case of weighted voting with $\alpha_1 = 1$ and all other weights set to zero: $\alpha_2 = \alpha_3 = \ldots = \alpha_m = 0$. Although there are many possible ways of choosing proper weights, we propose that the weights be set as follows:

$$\alpha_r = P(\omega_t | rank = r), \; r = 1, 2, \ldots, m \qquad (2)$$

That is, given the fact that we know that a memory pattern is (locally) ranked 1, α_1 is the probability that it is, in fact, the target pattern. The target pattern does not always locally get ranked 1, though. And, given the fact that a memory pattern locally receives a rank of r, α_r is the probability that said memory pattern is the target. For notational convenience, let $P^{(r)} = P(\omega_t | rank = r)$. Hence, the total number of votes received by pattern $\mathbf{x}_k$ is given by:

$$N_{\mathbf{x}_k} = P^{(1)} N_{\mathbf{x}_k}^{(1)} + P^{(2)} N_{\mathbf{x}_k}^{(2)} + \ldots + P^{(m)} N_{\mathbf{x}_k}^{(m)} \qquad (3)$$

For the special case of binary and random memory patterns, it is possible to derive theoretical expressions for $P^{(r)}$ and $N_{\mathbf{x}_k}^{(r)}$ as a function of pattern dimension, window

size, and noise level. In fact, the derivation is given next (due to space limitations, we will omit the derivation of the weights $P^{(r)}$). For more practical memory sets, the weights can be determined empirically using a training phase and an additional set of training data.

IV. Assumptions for the Theoretical Analysis

Both the voting memory and the weighted voting memory can store real-valued (or integer-valued) and heteroassociative memory sets. For the theoretical analysis that follows, though, we assume that the memory set is binary-valued: $\mathbf{x}_i \in \{0, 1\}^N$, and random: each memory pattern is generated randomly and independently. We will assume that each component of the fundamental memory patterns has a 50% chance of being 1 and 50% chance of being 0.

We start by considering the identification task. In this case, we test the system with noisy versions of the memory set patterns and see how well the system can retrieve the correct pattern. To create a suitable memory key, we proceed as follows. We first select one of the memory set patterns—call it the *target memory pattern*, or simply *target*. The memory key is formed by corrupting the target memory pattern with an amount ρ of uniform random noise; that is, with probability ρ, each component of the target pattern is flipped from its original value. Each of the remaining $m - 1$ fundamental memories will be called *non-target* memory patterns.

We want to compute how many votes are received by each memory pattern. There are really only 2 cases to consider: the number of votes received by the target memory pattern: N_t, and the number of votes received by the *i*th non-target memory pattern: N_i. In the weighted voting scheme, Equation 3 is used to compute N_t and N_i. To use this equation, we must derive the expected number of windows at each rank *r* for the target pattern: $N_t^{(r)}$, as well as the *i*th non-target pattern: $N_i^{(r)}$.

V. Distribution for $N_t^{(r)}$ and $N_i^{(r)}$

In [1], the authors showed (for the basic voting model) that it is possible to derive an expression for the probability P_t that a local window votes for the target pattern and the probability P_i that a local window votes for one of the non-target patterns. Using a similar analysis, it is possible to derive, for the weighted voting memory, the probability that a local window will receive a rank of 1, 2, …, *m* (the details for this analysis can be found in [6]). Let $P_t^{(r)}$ denote the probability that a single a single local window of the target image is ranked *r*. Similarly, let $P_i^{(r)}$ denote the probability that a single local window of the *i*th non-target memory pattern is ranked *r*.

For the target memory pattern, each window of the RAM network performs a Bernoulli experiment with probability $P_t^{(r)}$ of success (getting a vote) and probability $1 - P_t^{(r)}$ of failure (not getting a vote). If the experiment is repeated over all N/n windows, the probability that there will be *j* successes follows a binomial distribution:

$$P(N_t^{(r)} = j) = \binom{N/n}{j} [P_t^{(r)}]^j [1 - P_t^{(r)}]^{\frac{N}{n} - j} \quad (4)$$

Similarly, for the non-target patterns, the distribution for the number of windows at each rank *r* is given by:

$$P(N_i^{(r)} = j) = \binom{N/n}{j} [P_i^{(r)}]^j [1 - P_i^{(r)}]^{\frac{N}{n} - j} \quad (5)$$

VI. Distribution for N_t, N_i, and N_i^{max}

Now we have computed the number of windows $N_{\mathbf{x}_k}^{(r)}$ for each rank *r* and for each memory pattern $\mathbf{x}_k$. Actually, we computed $N_{\mathbf{x}_k}^{(r)}$ for the target pattern: $N_t^{(r)}$ and for the *i*th non-target pattern: $N_i^{(r)}$. We can now compute the total number of votes received by each memory set pattern. The total number of votes received by the target pattern is given by:

$$N_t = \sum_{r=1}^{m} P^{(r)} N_t^{(r)} \quad (6)$$

The probability distribution for N_t can be computed by exhaustively enumerating all possible rankings of the N/n votes. For example, we can construct a table which lists all possible values for $N_t^{(1)}, N_t^{(2)}, \ldots, N_t^{(m)}$. Each row of the table will occur with a certain probability (given by the product of the individual probabilities). The probability that N_t has value, say *z*, can be obtained by summing all rows which have probability *z*. Hence, an analytic expression for the discrete probability density function N_t can be written as:

$$P(N_t = z) = \sum_{i_1 = 0}^{N/n} \cdots \sum_{i_m = 0}^{N/n} \left[\prod_{r=1}^{m} P(N_t^{(r)} = i_r) \right] \times C_1 \times C_2 \quad (7)$$

The term C_1 is used to enforce the constraint that the sum of all the $N_t^{(r)}$ values must be the total number of windows:

$$C_1 = \delta\left(\sum_{r=1}^{m} i_r - \frac{N}{n} \right) \quad (8)$$

where δ is the delta function: $\delta(s) = 1$ if $s = 0$ and $\delta(s) = 0$ otherwise. The term C_2 is used to ensure that we sum only those rows that have probability *z*.

$$C_2 = \delta\left(\sum_{r=1}^{m} P^{(r)} i_r - z \right) \quad (9)$$

In general, for the weighted voting memory, the weights are non-negative real numbers. Hence the number of votes N_t is not an integer (nor is z in Equation 7). However, there are only a finite number of possibilities for z, and hence N_t can be described by the discrete distribution given in (7).

Similarly, the discrete probability density function of N_i can be written as

$$P(N_i = z) = \sum_{i_1=0}^{N/n} \cdots \sum_{i_m=0}^{N/n} \left[\prod_{r=1}^{m} P(N_i^{(r)} = i_r) \right] \times C_1 \times C_2$$

Here, the constraints C_1 and C_2 are the same as those given in Equations 8 and 9, respectively.

We now have probability density functions for N_t, the number of votes received by the target pattern, and N_i, the number of votes received by the ith non-target pattern. The system will retrieve the correct pattern when N_t is larger than N_i for each and every one of the $m-1$ non-target memory patterns. Of these $m-1$ non-target patterns, we need only concern ourselves with the one that received the maximum number of votes.

Let N_i^{max} be a random variable that gives the maximum of the $m-1$ random variables $\{N_i\}$. The distribution for N_i^{max} can be derived by accounting for all possible ties among the weighted votes received by the non-target patterns. For example, suppose the maximum number of votes among the $m-1$ non-target patterns is j. The probability that a single non-target memory set pattern received j votes (and all the other $m-2$ non-target patterns received less) is $Prob(N_i = j)Prob(N_i < j)^{m-2}$. The probability that precisely k of the non-target patterns achieve j number of votes (a k-way tie at the top) is given by: $\binom{m-1}{k} Prob(N_i = j)^k Prob(N_i < j)^{m-1-k}$. To account for all possible ties, we sum over all possible values of k:

$$P(N_i^{max} = z) = \sum_{k=1}^{m-1} \binom{m-1}{k} [P(N_i = z)]^k [P(N_i < z)]^{m-1-k}$$

VII. Identification Test

For the identification problem, the probability of correct retrieval is simply the probability that the number of votes received by the target image exceeds that received by the maximum non-target image:

$$P_{id} = Prob(N_t > N_i^{max}) \quad (10)$$

Since N_t is an integer that varies from 0 to N/n, we have:

$$P(N_t > N_i^{max}) = \sum_{j=1}^{N/n} P(N_t > N_i^{max} \,|\, N_t = j) P(N_t = j)$$

Assuming N_t is independent of N_i^{max}, the distribution for the probability of correct retrieval (identification rate) can be written as:

$$P_{id} = P(N_t > N_i^{max}) = \sum_{j=1}^{N/n} P(N_t = j) \sum_{p=0}^{j-1} P(N_i^{max} = p) \quad (11)$$

VIII. Watch List Test

Here, we use a threshold and reject patterns that do not get at least T votes. The detection and identification rate (DIR) is the probability

$$P_{dir} = Prob(N_t > N_i^{max} \cap N_t > T) \quad (12)$$

To compute P_{dir}, we use Equation 12, but now the smallest allowable value for N_t is T:

$$P_{dir}(n, N, \rho, m) = \sum_{j=T}^{N/n} P(N_t = j) \sum_{p=0}^{j-1} P(N_i^{max} = p) \quad (13)$$

For the false positive test, the memory key is created by generating a completely random input. Here there is no target pattern in the memory set, and all m memories are non-target patterns. It is possible to derive the probability $P_{i'}^{(r)}$ that the i' th (non-target) memory has rank r [6].

At a single window, $P_{i'}^{(r)}$ gives the probability that the i' th non-target memory pattern will be ranked r, and $1 - P_t^{(r)}$ gives the probability that it will not be ranked r. If the experiment is repeated over all N/n windows, the probability that there will be j successes follows a binomial distribution:

$$P(N_{i'}^{(r)} = j) = \binom{N/n}{j} [P_{i'}^{(r)}]^j [1 - P_{i'}^{(r)}]^{\frac{N}{n} - j} \quad (14)$$

The total number of votes received by the i' th non-target memory pattern is:

$$N_{i'} = \sum_{i=1}^{m} P^{(r)} N_{i'}^{(r)} \quad (15)$$

Similar to Equations 7 and 8, the discrete form of the probability density functions for $N_{i'}$ can be computed by going through all possible combinations of r:

$$P(N_{i'} = z) = \sum_{i_1 = 0}^{N/n} \cdots \sum_{i_m = 0}^{N/n} \left[\prod_{r=1}^{m} P(N_{i'}^{(r)} = i_r) \right] \times C_1 \times C_2 \quad (16)$$

Let $N_{i'}^{max}$ denote the maximum among $\{N_{i'}\}$. The distribution for $N_{i'}^{max}$ is given by:

$$P(N_{i'}^{max} = j) = \sum_{k=1}^{m} \binom{m}{k} [P(N_{i'} = j)]^k [P(N_{i'} < j)]^{m-k} \quad (17)$$

The probability of false acceptance is the probability that one of the memory patterns receives more than T votes:

$$P_{far} = P(N_{i'}^{max} \geq T) \quad (18)$$

Using the distribution in (17), this can be computed as:

$$P_{far}(n, N, \rho, m) = \sum_{j=T}^{N/n} P(N_{i'}^{max} = j) \quad (19)$$

By varying the threshold T, we can achieve different values of P_{dir} and P_{far}. An ROC curve can be plotted to show the trade-off between the two as a function of the threshold.

IX. Conclusions

The goal of associative memory research is to design systems that can reliably store and retrieve information. For reliable performance, associative memory systems must have a rejection mechanism whereby very noisy or unwanted input patterns can be rejected by the system. Consider the implication of not having such a rejection mechanism when an associative memory is used in the context of human face recognition. In this case, images of strangers (people not in the fundamental memory set) will be (mis)matched with one of the known people. Such a system—even if it was able to achieve 100% identification rates on images of known people—is useless because its output could not be trusted.

In this paper, we proposed a way to study the capacity of associative memory systems that have a rejection capability. A high performance associative memory must maximize the identification rate P_{dir} and, simultaneously, minimize the false acceptance rate P_{far}. A flexible design would allow the user to adjust system parameters (for example, a threshold) to achieve the desired balance of P_{dir} and P_{far}.

The weighted voting associative memory model proposed in this paper has the following desirable properties:

•Can operate in autoassociative or heteroassociative mode

•Can store binary, integer, or real-valued data (though the theoretical analysis here pertains only to binary and random patterns).

•Has a rejection mechanism and a tunable threshold T which allows the user to adjust P_{dir} and P_{far}.

•Never produces a spurious memory.

In this paper, we have extended the analysis of the voting associative memory proposed in Ikeda et al, 2001. In addition, we have proposed a generalization of the voting memory where rather than cast a single vote for the best matching memory set pattern, each window casts a set of weighted votes. For the case of random and binary memory set patterns, we were able to derive expressions for the retrieval rate, detection and identification rate, and false acceptance rate for both the voting memory and the weighted voting

References

[1] Altmann, C., Bülthoff, H., and Kourtzi, Z. (2003). "Perceptual Organization of Local Elements into Global Shapes in the Human Visual Cortex, *Current Biology*, 13, 342-349.

[2] Burton, A., Bruce, V., Hancock, P., (1999). "From Pixels to People: A Model of Familiar Face Recognition," *Cognitive Science*, 23(1), 1-31.

[3] Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks*, MIT Press, Cambridge, Mass.

[4] Ikeda, N., Watta, P., Artiklar, M., and Hassoun, M. (2001). "Generalizations of the Hamming Net for High Performance Associate Memory," *Neural Networks*, 14(9), 1189-1200.

[5] Lockwood G., and Aleksander, I. (2003). "Predicting the behaviour of G-RAM networks," *Neural Networks,* 16, 91-100.

[6] Mu, X., (2004). "Automated Face Recognition: A Weighted Voting Method," Ph.D. Dissertation, Dept. Electrical and Computer Engineering, Wayne State University, Detroit, MI, 48202.

[7] Phillips, P., Moon, H., Rizvi, S., and Rauss, P. (2000). "The FERET Evaluation Methodology for Face-Recognition Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(10), 1090-1104.

[8] Phillips, P. Wechsler, H., Huang, J., and Rauss, P. (1998). "The FERET database and evaluation procedure for face recognition algorithms," *Image and Vision Computing J*, 16(5), 295-306.