SYSTEM REALIZATION USING ASSOCIATIVE MEMORY BUILDING BLOCKS

Mark Carlotto and David Izraelevitz
The Analytic Sciences Corporation
55 Walkers Brook Dr.
Reading, MA

ABSTRACT

A data-based model of associative memory is described which uses statistical inference techniques to estimate an output response from a set of inputs and a database of previously stored patterns. The model is easily scaled in terms of the number of patterns that can be stored in the database as well as the number of fields in a pattern. Other features include the ability to change the input and output fields, to adjust the amount of generalization performed by the associative memory, and to control the size of the database by pruning redundant or conflicting patterns. Applications of associative memories to a wide variety of problems are illustrated to motivate their use as general system building blocks. Implementations in hardware and software are discussed.

1. INTRODUCTION

Most intelligent systems perform some kind of "low level" numerically-intensive processing (e.g., as is required in early vision for shape recovery, object/background segmentation, and grouping) as well as "high-level" symbolic processing (e.g., for scene interpretation, natural language understanding, etc.). As a result, these systems tend to be divided into low-level, algorithm-based and high-level, knowledge-based modules. While connectionism provides a new perspective on what has been perceived as a dichotomy between low and high level processing, implementations of connectionist models using neural networks have proven to be unsatisfactory in some ways. Practical problems exist with respect to training, scaling, and in understanding and controlling the way neural networks perform generalization. The concept of memory-based reasoning[1] has been suggested as an alternative to neural networks which, instead of attempting to build associations via connection weights between neurons, computes an output response by comparing the input to a large database of previously stored patterns.

This paper expands the idea of memory-based reasoning to the more general paradigm of associative memory. Section 2 reviews some classical models of associative memory with particular attention to the number of patterns that can be correctly recalled and the ease with which the memory can be reconfigured (i.e., inputs and outputs changed). A new computational model, the data-based model of associative memory, is presented in Section 3. It too is also based on the idea of storing a large number of patterns or training vectors in a database, but uses statistical inference techniques rather than heuristics for estimating an output response. Means for adjusting the amount of generalization performed by the associative memory and for controlling the size of the database by pruning redundant or inconsistent patterns are also discussed. In Section 4, a number of problems are formulated in terms of the associative memory paradigm. Section 5 describes initial implementations on a Symbolics and a Connection Machine and outlines possible VLSI and optical implementations. The generality of the associative memory paradigm along with the potential for implementing practical associative memory processors in hardware and software motivates a new approach to system development. Section 6 discusses this idea and suggests future directions.

2. COMPUTATIONAL MODELS OF ASSOCIATIVE MEMORY

The basic model of associative memory is a "black box" with vector input $\underline{x}$ and vector output $\underline{y}$. Associations are created by simultaneously presenting input $\underline{x}$ and output $\underline{y}$ vectors in a "learning mode". In "recall mode", a partial pattern $\underline{x}^*$ is presented to the associative memory and used to access or compute the rest of the pattern $\underline{y}^*$. Two classical models of associative memory are now reviewed.

2.1 Correlation Matrix Model

The correlation matrix model[2] associates input vectors $\underline{x}_k$ with output vectors $\underline{y}_k$ via a correlation matrix $\underline{M} = \underline{Y}\,\underline{X}^T$ where $\underline{X}$ and $\underline{Y}$ are rectangular matrices having input vectors $\underline{x}_k$ and associated output vectors $\underline{y}_k$ as their columns respectively. If the $\{\underline{x}_k\}$ are orthonormal and the number of input vectors $K \leq N$, the number of elements in an input vector, K patterns can be stored in the associative memory without confusion. To increase the number of patterns to K', the size of the input vector must be increased to $N' \geq K'$ or cross-talk will occur, i.e., the output will contain contributions from more than one $\underline{y}_k$. Cross-talk will also occur if the $\underline{x}_k$ are not orthonormal; thus, the input vectors must be preprocessed, e.g., via a Gram-Schmit procedure. A disadvantage of the correlation matrix memory is that the input and output fields cannot be changed without recomputing $\underline{M}$.

2.2 Hopfield Model

The Hopfield model of associative memory[3] is based on a network of N neurons interconnected as $u_n = \mathrm{sgn}\,(\,\Sigma\, w_{nm}\, u_m - t_n\,)$ where $\underline{W} = \{w_{nm}\}$ is a matrix of weights and t_n are thresholds. The weight matrix is built by accumulating outer products of training vectors $\underline{W} = \Sigma\, \underline{u}_k\, \underline{u}_k{}^T$. Once the weight matrix has been computed, patterns can be recalled by clamping those fields of the state vector that are known (i.e., those elements that correspond to the input) and allowing the network to compute those that are unknown (i.e., those elements that correspond to the output). An advantage of the Hopfield model over the correlation matrix memory is that any subset of neurons can serve as the input field. The capacity of the Hopfield network is upper bounded by N patterns[4]; Hopfield[3] finds empirically that about 0.15N patterns can be stored without confusion. Like the correlation matrix memory, in order to increase the number of patterns, the size of the state vector $\underline{u}$ must be increased.

3. A DATA-BASED MODEL OF ASSOCIATIVE MEMORY

The data-based model of associative memory[5] is essentially an auto-associative scheme that involves storing a set of patterns $\{\underline{z}_k\}$ in a database and using statistical inference to recall the full pattern by estimating the output as a function of the input and the entire database. The capacity is equal to the size of the database, which can be easily increased without having to alter its structure. The associative memory is easily scaled in terms of the number of fields N, where any can be though of as inputs or outputs and can represent numerical or categorical data. For brevity, the following discussion is limited to numerical data. A more complete description of the data-based model can be found in Ref. 5.

3.1 Associative Recall by Statistical Inference

Assume that $\{\underline{z}_k\}$ is a random sampling of an underlying joint probability density $p(\underline{z})$ and consider the problem of determining a value y* for an output field from the values $\underline{x}^*$ of one or more input fields. (The problem of estimating a vector of outputs can be reduced to this case by determining each output field separately.) If the underlying density were known, it can be shown that the optimal estimate (in the least-squares sense) for y* given $\underline{x}^*$ is given by the conditional mean,

$$y^* = \int y\, p(y \mid \underline{x}^*)\, dy \;=\; \int y\, p(y, \underline{x}^*)\, dy \;/\; \int p(y, \underline{x}^*)\, dy. \qquad (1)$$

By approximating the underlying joint density as

$$p(\underline{z}) = \sum_k \Phi(\underline{z}_k, \underline{z}) \qquad (2)$$

where $\Phi(\underline{z}_k, \underline{z}) = (2\pi\sigma^2)^{-N/2} \exp\,[-\,(\underline{z}_k - \underline{z})^T(\underline{z}_k - \underline{z}) \,/\, 2\sigma^2\,]$, the optimal estimate can be written as:

$$y^* = \sum_k \Phi(\underline{x}_k, \underline{x}^*)\, y_k \,/ \sum_k \Phi(\underline{x}_k, \underline{x}^*). \tag{3}$$

The above estimate for y^* is a weighted sum of the $\{y_k\}$ divided by a normalizing factor that is equal to the sum of the weights. The kth weight depends on the distance between $\underline{x}^*$ and $\underline{x}_k$ and decreases as the distance increases.

Fig. 1 illustrates the problem of learning the non-linear mapping $y = x^2$ for $0 \le x < 256$. In the training mode, the associative memory was presented K input-output pairs $\{x_k, y_k=f(x_k)\}$, where the x_k were generated at random. The true and estimated values for y using K=10 and K=50 patterns are shown in (a) and (b). As one would expect, the accuracy of the estimate improves as the number of training vectors increases.

3.2 σ-Tuning

The estimated output depends on the contents of the database, the value of the input, and the smoothing factor σ. As σ decreases, y^* approaches the value y_k corresponding to the $\underline{x}_k$ that is nearest to $\underline{x}^*$. As σ increases, the estimates depend on y_k that are farther away. Thus σ controls the amount of generalization performed by the associative memory.

A value for σ can be computed using "leave-one-out" procedures[6]. Let y_k^* be the estimate of y_k obtained by leaving out the kth training vector $\underline{z}_k$,

$$y_k^* = \sum_{k' \neq k} \Phi(\underline{x}_{k'}, \underline{x}^*)\, y_{k'} \,/ \sum_{k' \neq k} \Phi(\underline{x}_{k'}, \underline{x}^*). \tag{4}$$

Clearly, if the $\underline{z}_k$ are kept, the estimate could only get worse as σ was increased. By not using it, the error $|\, y_k^* - y_k \,|$ depends only on $\{\underline{z}_{k'}\}$ $k' \neq k$ and will increase as σ either becomes too small or too large. A computational procedure for finding a suitable smoothing factor is to sweep σ logarithmically over some scale range using the value that minimizes

$$\varepsilon(\sigma) = \sum_k |\, y_k^* - y_k \,|. \tag{5}$$

The plot of $\varepsilon(\sigma)$ vs. σ for $1 < \sigma < 10$ is shown in Fig. 2 for the $y = x^2$ example presented earlier, for K=50 patterns. (The optimal value $\sigma^2 = 50$ was used in Fig. 1b.) Fig. 3 shows the dramatic effect the smoothing parameter can have on the estimate. Here, the results for K=10 patterns using the optimal value $\sigma^2 = 500$, and values 50 times smaller and larger, are shown for comparison. As the smoothing factor decreases the estimate becomes a staircase where the levels are equal to the nearest y_k. As the smoothing factor increases, the estimates approach the average of all the y_k.

3.3 Database Pruning

As the number of patterns in the database increases, the accuracy of the associative memory will generally improve. However it is a desirable feature to be able to remove vectors that are redundant (i.e., similar to other vectors in the database), or inconsistent (e.g., having the same input field values but different output field values). One procedure for pruning the database is to use the "leave-one-out" procedure discussed above to compute the error

$$\varepsilon(k) = \sum_{k'} |\, y_{k'k}^* - y_{k'} \,| \tag{6}$$

where $y_{k'k}^*$ is the estimate of $y_{k'}$ obtained by leaving out the kth training vector $\underline{z}_k$. The errors can then be used to rank the training vectors. Fig. 4 shows the $\varepsilon(k)$ plotted in ascending order for the K=50, $y=x^2$ example. The first 14 patterns were removed from the database (28% reduction in size) without any significant effect on the accuracy. The use of the principle of minimum description length[7] is being investigated for determining the optimal number of training vectors.

4. ASSOCIATIVE MEMORIES AS BUILDING BLOCKS

A large number of problems can be cast in an associative memory framework: learning input/output mappings and the solution of inverse problems, pattern classification, signal recovery from partial information, and rule-based systems, to name just a few. Several examples are presented below to motivate the idea of using associative memories as system building blocks.

4.1 Learning Input-Output Mappings

Suppose one would like to compute $y = f(x)$ or the inverse $x = f^{-1}(y)$ without knowing f( ). Examples include learning scene parameters[8] and modeling the dynamics of robot arms[9]. If f( ) is linear, then least-squares techniques can be used. However if the mapping is non-linear such as the $y = x^2$ problem presented earlier or is simply not known, and a large number of training vectors $\{x_k,y_k\}$ are available or can be generated, associative memories provide an attractive alternative. In addition, given $y = f(x)$, the inverse $x = f^{-1}(y)$ can be solved with an associative memory provided the transformation is one-to-one by training on $\{x_k,y_k\}$ as before, and reversing the sense of the inputs and outputs so that, in effect, one is estimating x^* from y^*. Fig. 5 shows the result of inverting the $y = x^2$ problem from Fig. 1b using the data-based associative memory with K=50 training vectors.

4.2 Pattern Classification

Associative memories can also be thought of as non-parametric pattern classifiers. Let $\underline{x} = \{x_n\}$ be an N-dimensional measurement vector and $\underline{y} = \{y_m\}$ be a vector of hypotheses each corresponding to one of M classes of interest. A pattern can be classified into one of the M classes by assigning the measurement to the class m corresponding to the largest y_m. For example, consider the problem of classifying points as lying inside or outside a circle of radius R. The input is the position of the point and the output is one if a point is inside the circle and zero otherwise. Fig. 6 shows the decision regions for K = 2, 10, and 50 training vectors. Results presented by Lippman[10] suggest that significantly more (~ 200) training vectors are required for a back-propagation classifier to learn circular decision regions. After training the data-based associative memory on 200 random points inside and outside the circle, 500 unknown points were classified with a correct classification rate greater than 98%.

4.3 Signal Reconstruction

Signal reconstruction from partial information (e.g., recognizing instances of known objects in the presence of occlusions) is essentially an auto-associative problem. Let $\underline{z}$ represent a discrete signal z(n). Consider the problem of recovering a class of signals that are symmetric about the center. A database was generated that contained 1000 six-dimensional vectors of the form ABCCBA where the first three elements were random numbers between -128 and 128. Once the database was loaded, the system was queried with partial vectors to see how it would fill in the missing information. Several examples are presented below:

```
[ 0  100  -100  ?    ?    ? ] ⇒ [ 0  100 -100 -106  91 .76 ]
[ 0   ?   -100  ?   100   ? ] ⇒ [ 0   91  -100 -106 100 .76 ]
[ ?  100  -100  ?    ?    ? ] ⇒ [ 30 100 -100 -104 100  30 ].
```

The associative memory is able to recover the full signal even in the last case with 60% of the information missing. Evidently the associative memory has also developed an implicit model of the symmetry reflected in the database.

4.4 Rule-Based Systems

Previous examples have shown how associative memories can perform numerical computation. An early motivation for the present work was the idea of memory-based reasoning[1] which involves drawing inferences directly from a large database of undigested facts and experiences. This is in contrast with traditional methods that use rules which must first be derived inductively, often with great effort, from experts. Rule-based systems can be cast in an associative memory framework by viewing the $\{\underline{z}_k\}$ as either data or rules. Patterns can be thought of as being divided into "if" and "then" fields, for convenience. For example, consider a subset of Winston's animal world[11] that deals with mammals and birds:

If (animal has hair) then (animal is mammal)
If (animal gives milk) then (animal is mammal)
If (animal has feathers) then (animal is bird)
If (animal flies) and (animal lays eggs) then (animal is bird)

Let the pattern vector $\underline{z}$ be divided into the following seven fields

z(1): "has-hair"	z(2): "is-mammal"	z(3): "gives-milk"
z(4): "has-feathers"	z(5): "is-bird"	z(6): "flies"
z(7): "lays-eggs"		

that can take on the values: +1, -1, and 0 if the assertion is true, false, or unknown. The above four rules can thus be represented by two patterns:

$\underline{z}_1 = [+1\ +1\ +1\ -1\ -1\ -1\ -1]$ and $\underline{z}_2 = [-1\ -1\ -1\ +1\ +1\ +1\ +1]$.

The following are example responses from the associative memory where ? denotes an output field:

$$[\,?\ +1\ \ ?\ \ ?\ \ ?\ \ ?\ \ ?\,] \Rightarrow [+1\ +1\ +1\ -1\ -1\ -1\ -1\,]$$
$$[\,?\ \ ?\ \ ?\ \ ?\ +1\ +1\ \ ?\,] \Rightarrow [\,-1\ -1\ -1\ +1\ +1\ +1\ +1\,]$$
$$[+1\ \ ?\ \ ?\ \ ?\ \ ?\ +1\ \ ?\,] \Rightarrow [+1\ \ 0\ \ 0\ \ 0\ \ 0\ +1\ \ 0\,].$$

The last example clearly represents a reasonable response to contradictory information that was not explicitly accounted for in the database. Finally it is noted that in associative memory realizations of rule-based expert systems, there do not have to be preferred inputs or outputs.

5. IMPLEMENTATIONS OF ASSOCIATIVE MEMORY

5.1 Software Implementations

The initial version of the data-based associative memory was implemented on a Symbolics in Lisp. Databases are implemented as arrays that are accessed through structures so that multiple associative memories can be defined, interconnected, and exercised simultaneously. An interactive environment was created to allow a user to create an instance of a database with an arbitrary number of fields (symbolic or numeric), to read/write a database from/to disk, to print the contents of a database, to merge multiple databases, to store and delete patterns in a database, to perform generalization, to prune a database, and finally to run and evaluate the performance of an associative memory.

To investigate larger problems, a data-parallel version of the associative memory was implemented on a Connection Machine[12]. Two virtual-processor sets within the machine are defined: one for the database and the other for the domain of interest, which are images. An output image is computed by accumulating the response of an input image(s) across each pattern in the database (in actuality, the numerator and denominator in Eq. 3 are accumulated separately and divided at the end). Fig. 7 is the result of using the Connection Machine based associative memory to detect roads in multispectral imagery. The input to the associative memory $\underline{x}(i,j)$ is a image of multispectral vectors and the output y(i,j) is equal to one if a road is present and zero otherwise. A database of training vectors was built over one area in the image using ground truth. The entire image was then processed, accumulating the results in place in time proportional to the number of training vectors. A decision rule was applied that assigned the pixel (i,j) to a road if $y(i,j) > 0.5$. The results over another area were compared to independent ground truth with a correct classification rate of 98%.

5.2 Hardware Implementations

Like VLSI implementations of classical content addressable associative memories[13], the structure of the data-based associative memory appears to be amenable to modular hardware implementations as well. With reference to Eq. 3, an associative memory and database can be partitioned into a number of smaller databases as:

$$y^* = \frac{\sum_{K_1} \Phi(\underline{x}_k, \underline{x}^*)\, y_k + \sum_{K_2} \Phi(\underline{x}_k, \underline{x}^*)\, y_k + \ldots}{\sum_{K_1} \Phi(\underline{x}_k, \underline{x}^*) + \sum_{K_2} \Phi(\underline{x}_k, \underline{x}^*) + \ldots.} \qquad (7)$$

Fig. 8 is a hardware block diagram of a proposed "data-slice" associative memory processor. Each slice is associative memory unit (AMU) having N interconnected sets of tri-state input/output lines and corresponding control lines that define the input and output fields. Additional chip interconnects are required to pass the partial sums in the numerator and denominator between AMUs. Each AMU would contain local memory for storing a portion of the database, an arithmetic unit for computing a contribution to the output estimate, and a controller for coordinating inter-AMU data transfers. Ways of partitioning the computation by field and word would need to be developed as well for truly general and extensible realizations in hardware.

Finally, the possibility of an optical realization of the data-based associative memory is considered. An optical processing architecture is depicted in Fig. 9. The input vector $\underline{x}^*$ is introduced at P1 using an array of light emitting diodes, for example. Optics spread the light horizontally while imaging the input vector vertically onto a mask at P_2 that contains the database vectors, $\{\underline{z}_k\}$. Optics between P_2 and P_3 integrate the light transmitted through the mask vertically while imaging horizontally. The intermediate result at P_3 can be thought of as a vector of weighting factors. Assume for the moment that a diffuser is placed at P_3 to uniformly scatter the incident light and that the imaging optics between P_1 and P_3 are replicated between P_3 and P_5. (Alternatively, a mirror could be placed behind the diffuser to reflect the $\{w_k\}$ back and allow the outputs to be read out at P_1). The result at P_5 is $\underline{y}^* = \underline{Y}\, \underline{X}^T \underline{x}^*$ where $\underline{X} = \{\underline{x}_k\}$ and $\underline{Y} = \{\underline{y}_k\}$. The architecture in Fig. 9 is thus an optical implementation of Kohonen's cross-correlation associative memory.

In order to implement the data-based associative memory, either some kind of non-linear optical device must be placed at P_3 or an iterative electro-optical processor (IOP)[14] be employed. The problem is in computing the $\exp[-(\underline{x}^* - \underline{x}_k)^T (\underline{x}^* - \underline{x}_k) / 2\sigma^2$ weighting factors. The IOP is an optical vector-matrix multiplier in an electronic feedback loop that contains a high-speed digital signal processor. The exponent is the sum of three terms:

$$(\underline{x}^* - \underline{x}_k)^T (\underline{x}^* - \underline{x}_k) = \underline{x}^{*T}\underline{x}^* - 2\, \underline{x}^{*T}\underline{x}_k + \underline{x}_k^T \underline{x}_k. \qquad (8)$$

For each new database, the third term can be computed once initially by cycling the $\underline{x}_k$ through the input and storing the $\underline{x}_k^T\underline{x}_k$ in the digital signal processor. Then, for each new input vector, $\underline{x}^{*T}\underline{x}^*$ can be computed electronically, and added to the prestored $\underline{x}_k^T\underline{x}_k$ along with the $\underline{x}^{*T}\underline{x}_k$ that are being computed optically during the first IOP cycle. In the second cycle, the previous result is fed back through the vector matrix multiplier a second time to form the final output.

6. SUMMARY

A number of new ideas have been discussed in this paper: a modular, data-based model of associative memory, the general utility of associative memories in performing algorithm- and knowledge-based computation, and the implementation of practical associative memory processors in hardware and software. It is our belief that the realization of large, reconfigurable, and flexible associative memory modules will provide a new perspective on system design and implementation. One could envision the implementation of intelligent systems having diverse functionalities in a homogeneous fashion using a common building block whose behavior is entirely data-dependent.

REFERENCES

1. C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Communications of the ACM*, Vol. 29, No. 12, pp 1213-1228, Dec. 1986.
2. T. Kohonen, E. Oja, and P. Lehtio, "Storage and processing of information in distributed associative memory systems," in Parallel Models of Associative Memory, G. Hinton and J. Anderson (eds.), Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1981.
3. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci., Vol. 79 (Biophysics), pp 2554-2558, April 1982.
4. Y. Abu-Mostafa and J. St. Jacques, "Information capacity of the Hopfield model," IEEE Trans. Inf. Theory, Vol. IT-31, No. 4, July 1985.
5. D. Izraelevitz and M. Carlotto, "A database model of associative memory based on statistical inference," Tech. Info. Memo. TP-8569-1, TASC (The Analytic Sciences Corp.), October, 1989.
6. K. Fukunaga and D. Hummels, "Leave-one-out procedures for nonparametric error estimates," IEEE Trans. PAMI, Vol. 11, No. 4, April, 1989.
7. J. Rissanen, "Universal coding, information, prediction, and estimation," IEEE Trans. Inf.Theory, Vol. IT-30, No. 4, July 1984.
8. D. Kersten, A. O'Toole, M. Sereno, D. Knill, and J. Anderson, "Associative learning of scene parameters from images," Applied Optics, Vol. 26, No. 23, 1 December, 1987.
9. C. Atkeson and D. Reinkensmeyer, "Using associative content addressable memories to control robots," Proc. 27th Conf. on Decision and Control, Austin, TX, December 1988.
10. R. Lippman, "An introduction to computing with neural nets," ASSP Magazine, Vol. 4, No. 2, April, 1987.
11. P. Winston, LISP, Addison-Wesley, Reading, MA, 1981.
12. W. Hillis, The Connection Machine, MIT Press, Cambridge, MA, 1985.
13. I. Scherson and S. Ilgen, "A reconfigurable fully parallel associative processor," J. Parallel and Dist. Computing, Vol. 6, pp 69-89, 1989.
14. M. Carlotto and D. Casasent, "Microprocessor-based fiber-optic iterative optical processor," Applied Optics, Vol. 21, No. 1, pp 147-152, 1 January 1982.

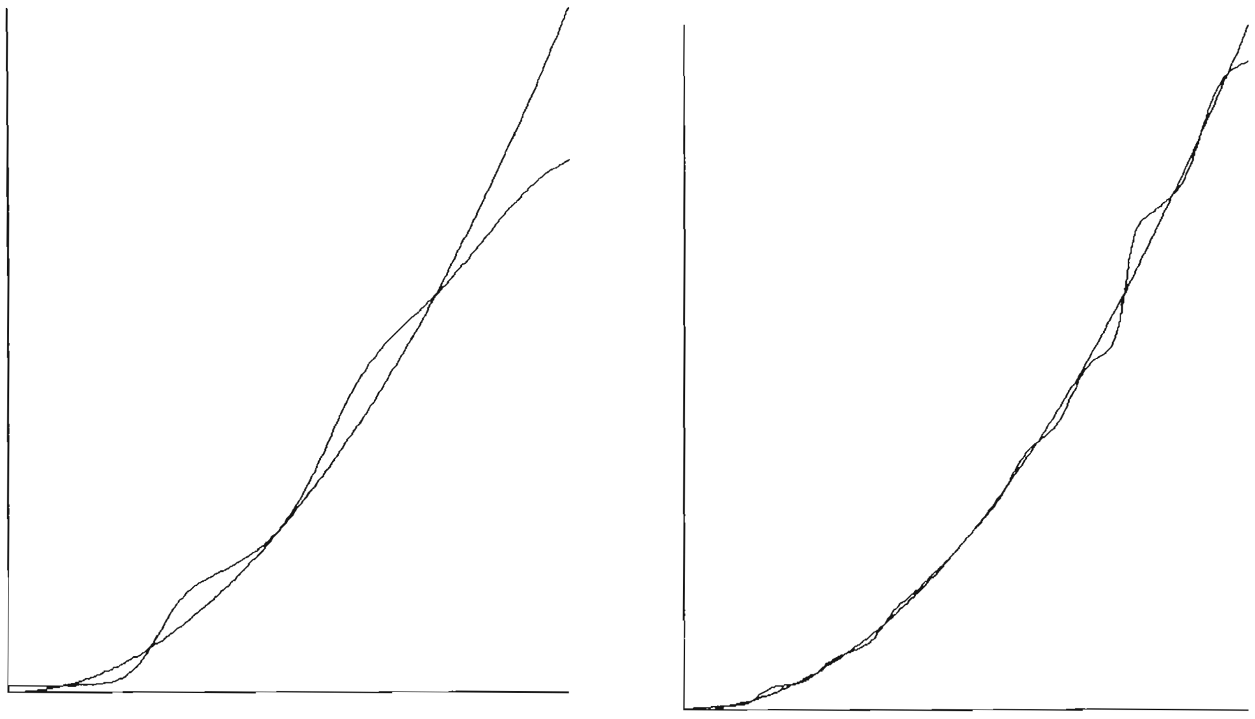

Fig.1 True and estimated values of $y = x^2$ for K=10 training vectors (a) left, and K=50 training vectors (b) right

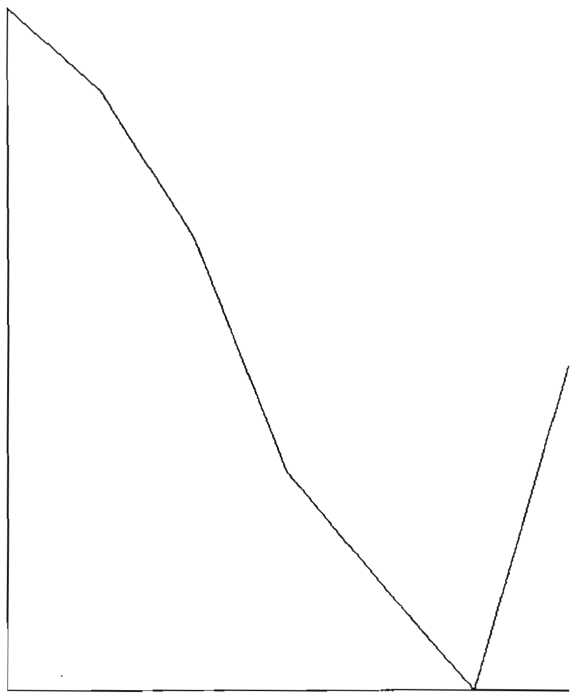

Fig.2 Plot of the $\varepsilon(\sigma)$ between $\sigma^2 = 1$ and $\sigma^2 = 100$ for the $y= x^2$ K=50 problem. The minimum value $\sigma^2 = 50$ was used in Fig. 1b.

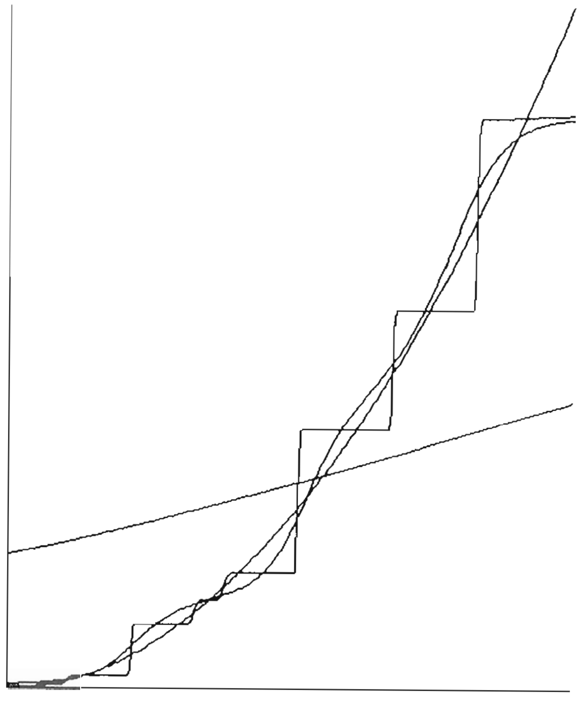

Fig. 3 Effect of the smoothing factor σ on the estimate for the $y= x^2$ K=10 problem. The optimal value $\sigma^2 = 500$ and values 50 X smaller and larger shown.

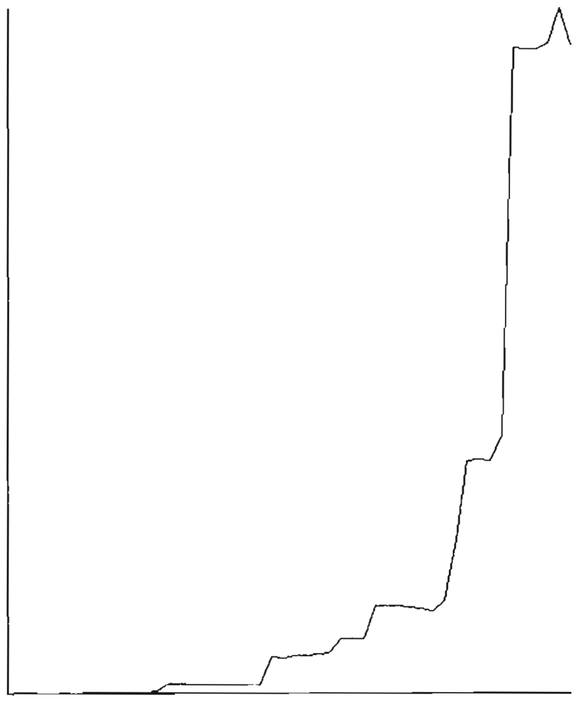

Fig. 4 Plot of ε(k) for the y= x^2 K=50 problem. Removing the first 14 training vectors reduces the database by 28% without affecting the accuracy.

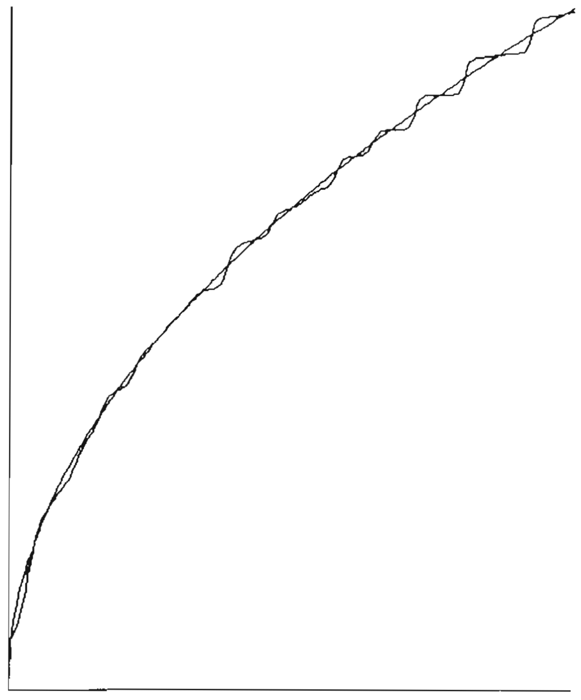

Fig. 5 True and estimated values for x = √y (K=50).

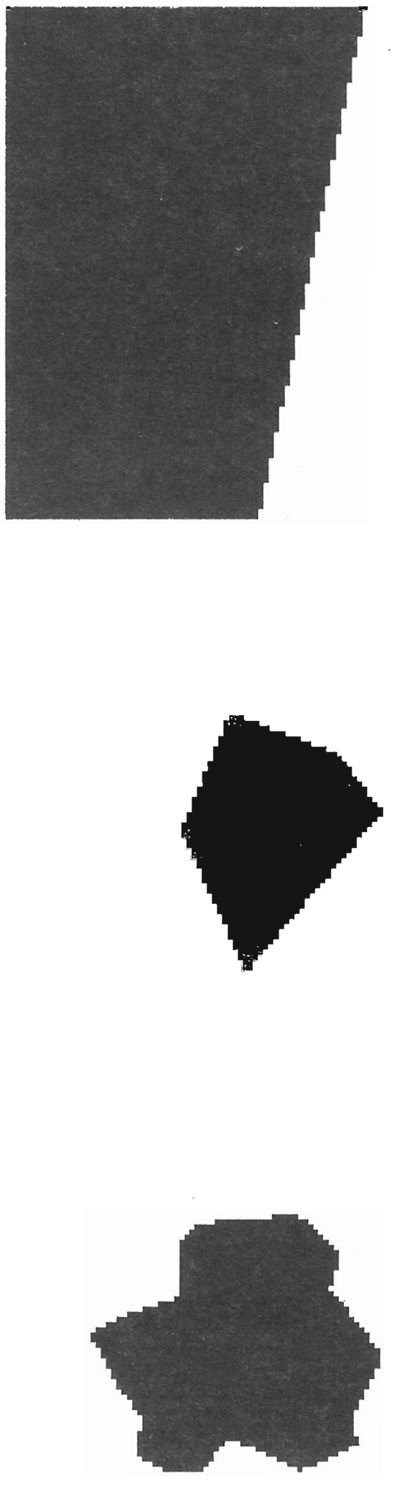

Fig. 6 Decision regions for the circle problem. Top to bottom: K=2, 10, and 50 training vectors.

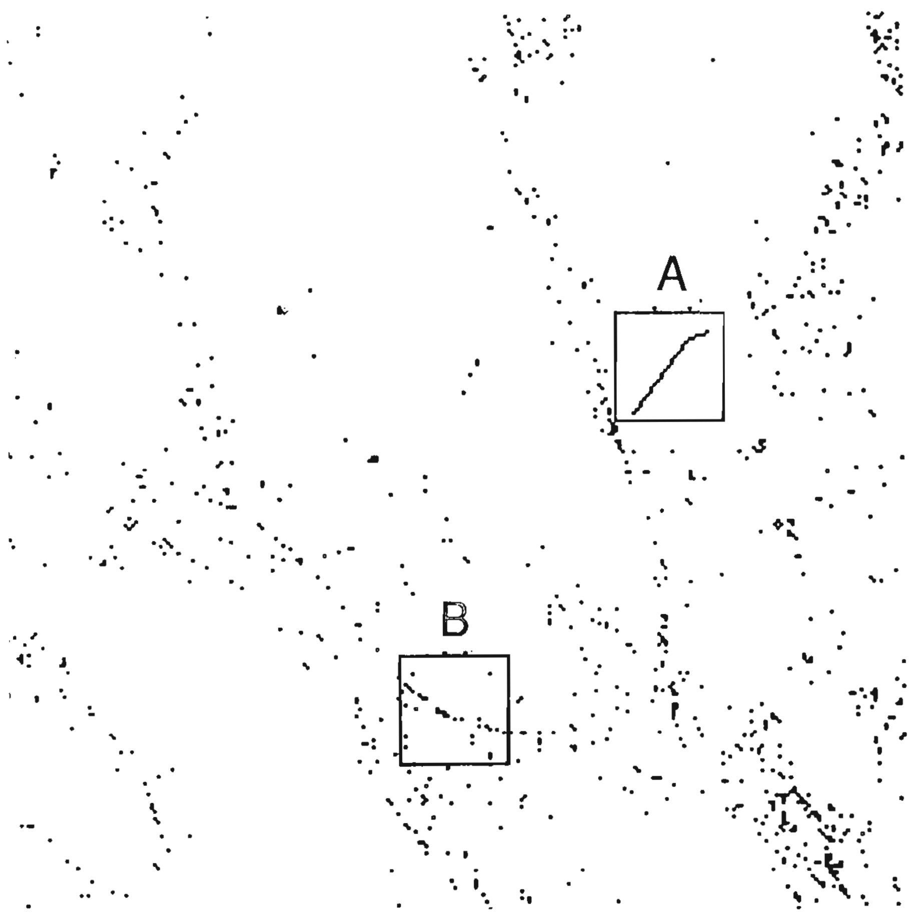

Fig. 7 Example of Connection Machine version of the data-based associative memory for detecting roads in Landsat TM imagery. The system was trained over area A. The associative memory and database were then used to classify the entire image. Results over area B were compared with ground truth and shown to be 98% correct.

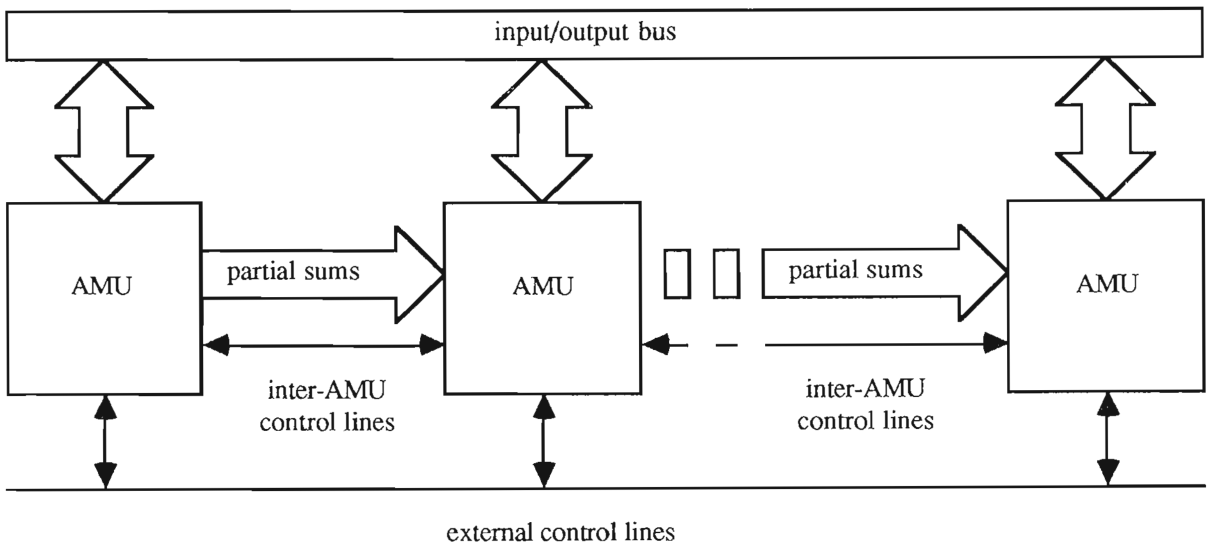

Fig. 8 Organization of a "data-slice" associative memory processor. Each associative memory unit (AMU) computes a part of the estimate.

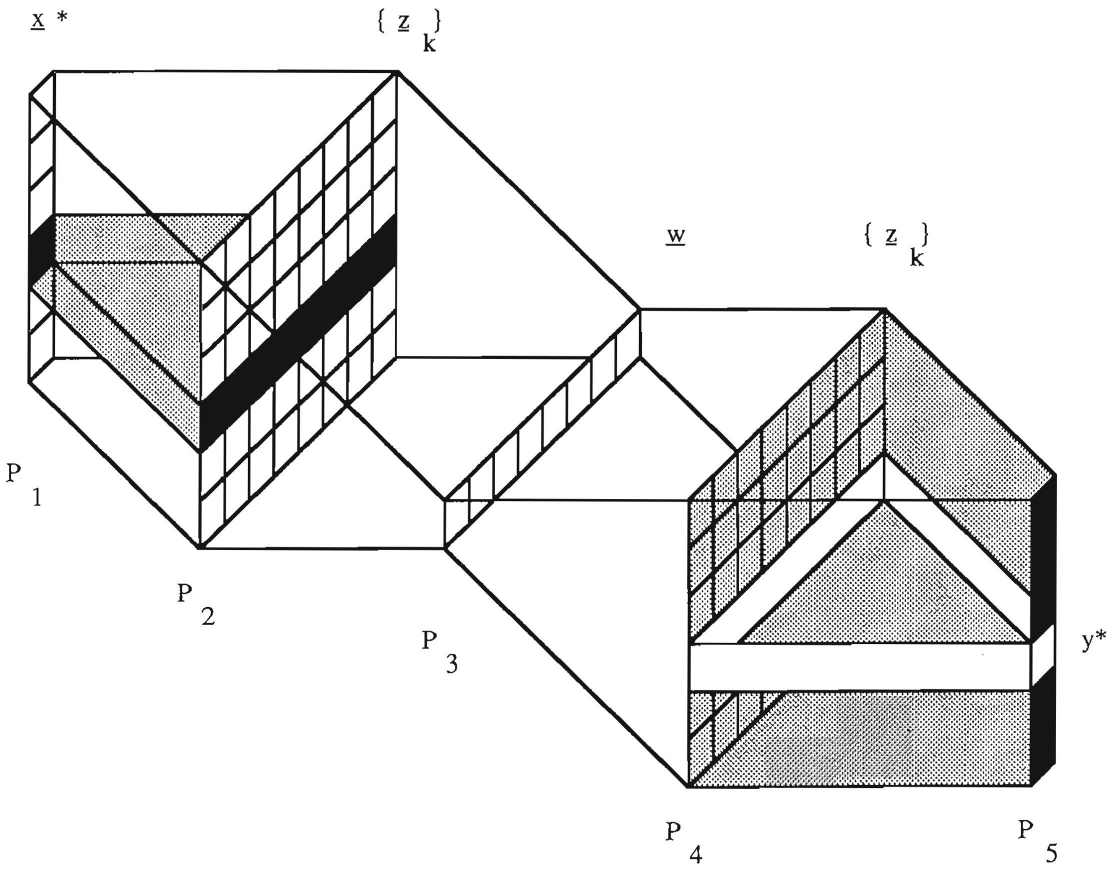

Fig. 9 Optical processor realization of associative memory (see text).