A SUBSPACE APPROACH TO INVARIANT PATTERN RECOGNITION USING HOPFIELD NETWORKS

A. H. Gee, S. V. B. Aiyer & R. W. Prager

CUED/F-INFENG/TR 62

May 21, 1992

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

Email: ahg/svb10/rwp @eng.cam.ac.uk

A Subspace Approach to Invariant Pattern Recognition using Hopfield Networks

A. H. Gee, S. V. B. Aiyer & R. W. Prager

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

Email: ahg/svb10/rwp @eng.cam.ac.uk

May 21, 1992

Abstract

This work is concerned with a pattern recognition system which uses a method of subspace projection to compare n-point template and unknown patterns. The system is intrinsically invariant to linear transformations, though dependent on the relative ordering of the points within the template and unknown. However, invariance to point ordering may be added through the use of a Hopfield network as an optimization tool. Finding the correct point ordering is formulated as a combinatorial optimization problem, and then mapped onto a modified Hopfield network for solution. The overall pattern recognition system is successfully used to recognize instances of the ten handwritten digits. The results confirm that the system is invariant to both linear transformations and point ordering.

Introduction

In 1985, Hopfield and Tank [1] showed how the Hopfield network could be used to solve combinatorial optimization problems of the Travelling Salesman type. Their approach involved specifying two quadratic energy functions of the network output; the first was minimized when the network output represented a valid solution to the problem, the second was a monotonic function of the 'cost' of the solution. The sum of both of these formed the problem specific energy function. They then showed how this function could be associated with the Liapunov function of the Hopfield network; by setting the network parameters appropriately, the minimization of the network Liapunov function would effectively solve the optimization problem. However, it was found that the network frequently failed to find valid solutions, let alone high quality ones [2].

More recent work has revealed how best to modify the original Hopfield network to produce both valid and high quality solutions to combinatorial optimization problems. One approach has been to use a multi-state encoding scheme to explicitly enforce one of the validity constraints, coupled with a mean field annealing process [3]; this technique has been particularly successful when applied to Graph Partitioning type problems [4]. Another quite distinct approach evolved through an eigenvector and subspace analysis of the original Hopfield network's behaviour [5]. This lead to the development of a modified network, proposed in [6], which rigidly enforces all the validity constraints through a direct subspace projection, leaving the original network dynamics free to minimize the cost function alone. By decoupling the two parts of the problem specific energy function, the network operates with an increased efficiency and 100% reliability. A mathematical framework, drawing heavily on the use of Kronecker products (also known as Tensor products)

[7], has been developed to map arbitrary combinatorial optimization problems onto the modified network [6]. The modified network has been particularly successful when applied to Travelling Salesman type problems [6].

In this paper we use the modified network, as well as the Kronecker product framework, to solve a combinatorial optimization problem associated with a particular method of invariant pattern recognition. The pattern recognition scheme uses a method of subspace projection to compare template and unknown patterns, each of which are represented by n points placed evenly around their outlines. The recognition scheme demands that the n points be ordered in the same manner for both the template and unknown pattern being compared; the problem of finding the correct ordering may be formulated as a combinatorial optimization problem. We illustrate how the Kronecker product framework may be used to map the problem onto the modified network. Finally, we present results of the network being used in a pattern recognition system attempting to classify handwritten examples of the ten digits. Whilst the results are encouraging, more interesting are the intermediate states which the network passes through as it solves the optimization problem. These give a valuable insight into how the network solves the problem, an insight which will prove useful when attempting to improve the network's performance, or derive efficient mappings for other optimization problems.

Kronecker Product Notation and Matrix Identities

Let $\mathbf{A}^T$ denote the transpose of $\mathbf{A}$.
Let $[\mathbf{A}]_{ij}$ refer to the element in the i^{th} row and j^{th} column of the matrix $\mathbf{A}$.
Similarly, let $[\mathbf{a}]_i$ refer to the i^{th} element of the vector $\mathbf{a}$.
Sometimes, where the above notation would appear clumsy, and there is no danger of ambiguity, the same elements will be alternatively denoted A_{ij} and a_i.
Let $\mathbf{A} \otimes \mathbf{B}$ denote the Kronecker product of two matrices. If $\mathbf{A}$ is an $n \times n$ matrix, and $\mathbf{B}$ is an $m \times m$ matrix, then $\mathbf{A} \otimes \mathbf{B}$ is an $nm \times nm$ matrix given by:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}\mathbf{B} & A_{12}\mathbf{B} & \dots & A_{1n}\mathbf{B} \\ A_{21}\mathbf{B} & A_{22}\mathbf{B} & \dots & A_{2n}\mathbf{B} \\ \dots & \dots & \dots & \dots \\ A_{n1}\mathbf{B} & A_{n2}\mathbf{B} & \dots & A_{nn}\mathbf{B} \end{bmatrix} \tag{1}$$

Let $\text{vec}(\mathbf{A})$ be the function which maps the $n \times m$ matrix $\mathbf{A}$ onto the nm-element vector $\mathbf{a}$. This function is defined by:

$$\mathbf{a} = \text{vec}(\mathbf{A}) = [A_{11}, A_{21}, \dots, A_{n1}, A_{12}, A_{22}, \dots, A_{n2}, \dots, A_{1m}, A_{2m}, \dots, A_{nm}]^T \tag{2}$$

Throughout this paper we make use of the following identities (see [7] for proofs):

$$\text{trace}(\mathbf{AB}) = \text{trace}(\mathbf{BA}) \quad (\text{for } \mathbf{A} \text{ and } \mathbf{B} \text{ both } n \times n) \tag{3}$$

$$\text{trace}(\mathbf{AB}) = \left[\text{vec}(\mathbf{A}^T)\right]^T \text{vec}(\mathbf{B}) \quad (\text{for } \mathbf{A} \text{ and } \mathbf{B} \text{ both } n \times n) \tag{4}$$

$$\text{vec}(\mathbf{AYB}) = (\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{Y}) \tag{5}$$

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{X} \otimes \mathbf{Y}) = (\mathbf{AX} \otimes \mathbf{BY}) \tag{6}$$

Other Notation and Definitions

Let $\mathbf{I}^n$ be the $n \times n$ identity matrix.
Let $\mathbf{o}^n$ be the n-element column vector of ones:

$$[\mathbf{o}^n]_i = 1 \quad i \in \{1, \dots, n\} \tag{7}$$

Let $\mathbf{O}^n$ be the $n \times n$ matrix of ones:

$$[\mathbf{O}^n]_{ij} = 1 \quad i, j \in \{1, \dots, n\} \tag{8}$$

Let $\mathbf{R}^n$ be the $n \times n$ matrix given by

$$\mathbf{R}^n = \mathbf{I}^n - \frac{1}{n}\mathbf{O}^n \tag{9}$$

Multiplication by $\mathbf{R}^n$ has the effect of setting the column sums of a matrix to zero:

$$\begin{aligned} \sum_{i=1}^{n}[\mathbf{R}^n \mathbf{a}]_i &= \sum_{i=1}^{n}[\mathbf{a} - \frac{1}{n}\mathbf{O}^n \mathbf{a}]_i \\ &= \mathbf{o}^{nT}[\mathbf{a} - \frac{1}{n}\mathbf{O}^n \mathbf{a}] \\ &= \mathbf{o}^{nT}\mathbf{a} - \frac{1}{n}(\mathbf{o}^{nT}\mathbf{o}^n \mathbf{o}^{nT})\mathbf{a} \\ &= \mathbf{o}^{nT}\mathbf{a} - \mathbf{o}^{nT}\mathbf{a} \\ \Leftrightarrow \sum_{i=1}^{n}[\mathbf{R}^n \mathbf{a}]_i &= 0 \end{aligned} \tag{10}$$

Let $\mathcal{P}$ be the set of n coordinate pairs which define a 2-dimensional pattern:

$$\mathcal{P} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \tag{11}$$

Let $\mathbf{x}$ be the n-element column vector given by

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \tag{12}$$

Let $\mathbf{y}$ be the n-element column vector given by

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]^T \tag{13}$$

Let $\hat{\mathbf{x}}$ be a unit vector parallel to $\mathbf{x}$:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{|\mathbf{x}|} \tag{14}$$

Let $\hat{\mathbf{y}}$ be a unit vector parallel to $\mathbf{y}$:

$$\hat{\mathbf{y}} = \frac{\mathbf{y}}{|\mathbf{y}|} \tag{15}$$

Let $\mathbf{C}$ be the $n \times 2$ matrix whose columns are $\mathbf{x}$ and $\mathbf{y}$:

$$\mathbf{C} = [\mathbf{x}\ \mathbf{y}] \tag{16}$$

Let $\hat{\mathbf{C}}$ be the $n \times 2$ matrix whose columns are $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$:

$$\hat{\mathbf{C}} = [\hat{\mathbf{x}}\ \hat{\mathbf{y}}] \tag{17}$$

In the context of a pattern recognition system, in which any particular pattern is either a template or an unknown, let $\mathbf{x}^{\mathrm{M}}$, $\mathbf{y}^{\mathrm{M}}$, $\hat{\mathbf{x}}^{\mathrm{M}}$, $\hat{\mathbf{y}}^{\mathrm{M}}$, $\mathbf{C}^{\mathrm{M}}$ and $\hat{\mathbf{C}}^{\mathrm{M}}$ be the above quantities for an n-point template pattern, $\mathcal{P}^{\mathrm{M}}$. Also, let $\mathbf{x}^{\mathrm{U}}$, $\mathbf{y}^{\mathrm{U}}$, $\hat{\mathbf{x}}^{\mathrm{U}}$, $\hat{\mathbf{y}}^{\mathrm{U}}$, $\mathbf{C}^{\mathrm{U}}$ and $\hat{\mathbf{C}}^{\mathrm{U}}$ be the analogous quantities for an n-point unknown pattern, $\mathcal{P}^{\mathrm{U}}$.

Invariant Pattern Recognition by Subspace Projection

In this section we construct the framework of a pattern recognition system which is invariant to a wide variety of linear transformations. First, consider any new pattern, $\mathcal{P}^{\mathrm{M}'}$, obtained by transforming a template pattern as follows:

$$\begin{bmatrix} x_i^{\mathrm{M}'} \\ y_i^{\mathrm{M}'} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_i^{\mathrm{M}} \\ y_i^{\mathrm{M}} \end{bmatrix} \quad i \in \{1, \ldots, n\} \tag{18}$$

If we define $\mathbf{x}^{\mathrm{M}'}$ and $\mathbf{y}^{\mathrm{M}'}$ as

$$\begin{aligned} \mathbf{x}^{\mathrm{M}'} &= \left[x_1^{\mathrm{M}'}, x_2^{\mathrm{M}'}, \ldots, x_n^{\mathrm{M}'}\right]^T && (19) \\ \mathbf{y}^{\mathrm{M}'} &= \left[y_1^{\mathrm{M}'}, y_2^{\mathrm{M}'}, \ldots, y_n^{\mathrm{M}'}\right]^T && (20) \end{aligned}$$

then equation (18) is equivalent to

$$\begin{aligned} \mathbf{x}^{\mathrm{M}'} &= a\mathbf{x}^{\mathrm{M}} + b\mathbf{y}^{\mathrm{M}} && (21) \\ \mathbf{y}^{\mathrm{M}'} &= c\mathbf{x}^{\mathrm{M}} + d\mathbf{y}^{\mathrm{M}} && (22) \end{aligned}$$

Equations (21) & (22) tell us that the vectors $\mathbf{x}^{\mathrm{M}'}$ and $\mathbf{y}^{\mathrm{M}'}$ lie in the 2-dimensional subspace[1] spanned by $\mathbf{x}^{\mathrm{M}}$ and $\mathbf{y}^{\mathrm{M}}$. We can construct a projection matrix, $\mathbf{T}^{\mathrm{M}}$, for this subspace as follows:

$$\mathbf{T}^{\mathrm{M}} = \mathbf{C}^{\mathrm{M}}(\mathbf{C}^{\mathrm{M}^T}\mathbf{C}^{\mathrm{M}})^{-1}\mathbf{C}^{\mathrm{M}^T} \tag{23}$$

For any $\mathbf{r}$, the vector $\mathbf{T}^{\mathrm{M}}\mathbf{r}$ is the orthogonal projection of $\mathbf{r}$ onto the subspace spanned by $\mathbf{x}^{\mathrm{M}}$ and $\mathbf{y}^{\mathrm{M}}$.

For an unknown pattern $\mathcal{P}^{\mathrm{U}}$, a scalar "quality factor" q may be calculated which measures how well the vectors $\mathbf{x}^{\mathrm{U}}$ and $\mathbf{y}^{\mathrm{U}}$ fit in the subspace spanned by $\mathbf{x}^{\mathrm{M}}$ and $\mathbf{y}^{\mathrm{M}}$:

$$\begin{aligned} q &= \hat{\mathbf{x}}^{\mathrm{U}^T}\mathbf{T}^{\mathrm{M}}\hat{\mathbf{x}}^{\mathrm{U}} + \hat{\mathbf{y}}^{\mathrm{U}^T}\mathbf{T}^{\mathrm{M}}\hat{\mathbf{y}}^{\mathrm{U}} \\ &= \mathrm{trace}\left(\hat{\mathbf{C}}^{\mathrm{U}^T}\mathbf{T}^{\mathrm{M}}\hat{\mathbf{C}}^{\mathrm{U}}\right) && (24) \end{aligned}$$

Note that q is bounded by $0 \leq q \leq 2$, and reaches its maximum value only when both $\mathbf{x}^{\mathrm{U}}$ and $\mathbf{y}^{\mathrm{U}}$ lie fully in the subspace spanned by $\mathbf{x}^{\mathrm{M}}$ and $\mathbf{y}^{\mathrm{M}}$.

Equation (24) may be used as the basis of an invariant pattern recognition system. For any unknown pattern, $\mathcal{P}^{\mathrm{U}}$, q is used to give a measure of how well the unknown $\mathcal{P}^{\mathrm{U}}$ resembles a pattern obtained by a linear transformation of the template $\mathcal{P}^{\mathrm{M}}$. By comparing against a number of possible templates, the unknown pattern may be classified as belonging to the same class as the template with which it scores the maximum quality factor q.

Any $\mathcal{P}^{\mathrm{U}}$ obtained from $\mathcal{P}^{\mathrm{M}}$ by a transformation of the form given in equation (18) will lie wholly within the subspace spanned by $\mathbf{x}^{\mathrm{M}}$ and $\mathbf{y}^{\mathrm{M}}$, and will therefore score a maximum quality factor of 2. The classification scheme is therefore invariant to all transformations of the form given in equation (18). If all patterns are initially shifted such that their centres of gravity lie at the origin of coordinates, ie:

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i = 0 \tag{25}$$

then the transformations represented by equation (18) include rotations, enlargements and linear shears. The operation of shifting to the origin may be achieved by the simple matrix multiplication $\mathbf{C}^{\mathrm{s}} = \mathbf{R}^n\mathbf{C}$, where $\mathbf{C}^{\mathrm{s}}$ represents the pattern with its centre of gravity shifted to the origin. We apply this shifting operation to all template and unknown patterns, and use the superscript $^{\mathrm{s}}$ to denote quantities related to such shifted patterns.

This pattern recognition scheme demands that the sets of points representing the template and unknown patterns have the following properties:

1. Each set contains the same number, n, of point coordinates.
2. The point coordinates in the unknown and template coordinate sets are ordered in the same, preordained manner.

[1] There is a close link here with the notion of the *invariant subspace* of a transformation: see, for example, [8].

The first condition is simply a matter of representation, and is therefore easily fulfilled. To illustrate the importance of the second condition, consider the case where the unknown pattern $\mathcal{P}^{\mathrm{U}}$ is in fact identical to the template pattern $\mathcal{P}^{\mathrm{M}}$. Then we would expect the quality factor, q, as calculated using equation (24), to be exactly 2. However, if the ordering of the coordinates within $\mathcal{P}^{\mathrm{U}}$ is not the same as the ordering of the coordinates within $\mathcal{P}^{\mathrm{M}}$, then the quality factor will not, in general, be 2.

Point Ordering

For the pattern recognition system to be of any use, it is necessary to ensure that the template and unknown points are ordered in the same manner before equation (24) is used to calculate the quality of the match. A reordering of the points within a pattern may be defined by means of the n-element vector $\mathbf{p}$: let $[\mathbf{p}]_i = j$ if the point initially in position j is to be reordered into position i. We must also enforce

$$[\mathbf{p}]_i \neq [\mathbf{p}]_j \quad \text{for } i, j \in \{1, \ldots, n\}, i \neq j \tag{26}$$

if all the points in the original pattern are to be present in the reordered pattern. The reordering may be achieved directly by means of an $n \times n$ matrix operator $\mathbf{V}(\mathbf{p})$, which is constructed as follows:

$$[\mathbf{V}(\mathbf{p})]_{ij} = \begin{cases} 1 & \text{if } [\mathbf{p}]_i = j \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

With the above definition of $\mathbf{V}(\mathbf{p})$, the $n \times 2$ matrix $\mathbf{C}' = \mathbf{V}(\mathbf{p})\mathbf{C}$ contains the same point coordinates as $\mathbf{C}$, but reordered in the manner defined by $\mathbf{p}$. Note that $\mathbf{V}(\mathbf{p})$ is a permutation matrix: its elements are all either 1's or 0's, and each row and column contains only a single 1. Expressed mathematically, this means:

$$\begin{aligned} [\mathbf{V}(\mathbf{p})]_{ij} &\in \{1, 0\} && (28) \\ \sum_{i=1}^{n} [\mathbf{V}(\mathbf{p})]_{ij} &= 1 \quad j \in \{1, \ldots, n\} && (29) \\ \sum_{j=1}^{n} [\mathbf{V}(\mathbf{p})]_{ij} &= 1 \quad i \in \{1, \ldots, n\} && (30) \end{aligned}$$

By making use of the property of the matrix $\mathbf{R}^n$ given in equation (10), these conditions may be more neatly expressed as:

$$\begin{aligned} [\mathbf{V}(\mathbf{p})]_{ij} &\in \{1, 0\} && (31) \\ \mathbf{V}(\mathbf{p}) &= \mathbf{R}^n \mathbf{V}(\mathbf{p}) \mathbf{R}^n + \mathbf{S} && (32) \\ \text{where } \mathbf{S} &= \frac{1}{n} \mathbf{O}^n && (33) \end{aligned}$$

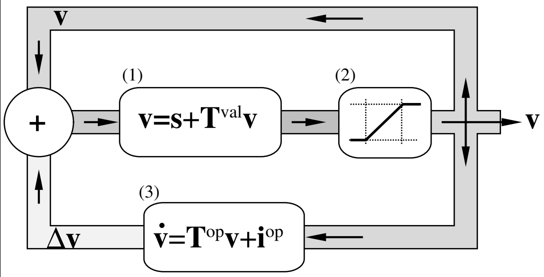
In equation (32), the row and column sums of the matrix $\mathbf{R}^n \mathbf{V}(\mathbf{p}) \mathbf{R}^n$ are both zero, and the row and column sums of $\mathbf{S}$ are both one, so the row and column sums of $\mathbf{V}(\mathbf{p})$ are also both one; hence the two summation conditions (29) and (30) are satisfied.

One approach to point ordering may be to impose something like a Hamilton Path ordering on the points, but this suffers in that it is not robust against the kind of deformations which we would expect the system to cope with; small deformations of a template might lead to significantly different Hamilton paths, resulting in a poor match quality, q. Another approach is to impose no *pre*ordering on the points at all; instead, each time the unknown is compared with a template, the points in the unknown are reordered to give the maximum possible match quality $q(\mathbf{p})$. This can be expressed as a combinatorial optimization problem as follows:

$$\max_{\mathbf{p}} q(\mathbf{p}) \quad \text{where } q(\mathbf{p}) = \operatorname{trace}\left[(\mathbf{V}(\mathbf{p}) \hat{\mathbf{C}}^{\mathrm{Us}})^T \mathbf{T}^{\mathrm{Ms}} (\mathbf{V}(\mathbf{p}) \hat{\mathbf{C}}^{\mathrm{Us}}) \right] \tag{34}$$

Expressed in words, the optimization problem is to find the permutation matrix $\mathbf{V}(\mathbf{p})$, which, when used to reorder the points in $\mathcal{P}^{\mathrm{U}}$, gives the maximum match quality, $q(\mathbf{p})$, with the template $\mathcal{P}^{\mathrm{M}}$.

The Modified Hopfield Network



(1) Projection of $\mathbf{v}$ onto valid subspace

(2) Nonlinear 'symmetric ramp' threshold functions constraining $\mathbf{v}$ to the unit hypercube.

(3) Change in $\mathbf{v}$ given by the gradient of the optimization energy term E^{op}, with $\Delta\mathbf{v} = \Delta t\dot{\mathbf{v}}$

Figure 1: Schematic diagram of modified network implementation.

In [6], a modified Hopfield network with increased efficiency is proposed — see Figure 1. The top loop contains a projection operator (1) which directly confines the network state vector, $\mathbf{v}$, to a particular affine subspace. The equation of this affine subspace, usually referred to as the "valid subspace", is

$$\mathbf{v} = \mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s} \tag{35}$$

The nonlinear operator (2) ensures that $\mathbf{v}$ stays within the unit hypercube, by applying a "symmetric ramp" threshold function to each of the elements of $\mathbf{v}$:

$$[\mathbf{v}]_i \quad \rightarrow \quad g([\mathbf{v}]_i) \tag{36}$$

$$\text{where } g([\mathbf{v}]_i) \quad = \quad \begin{cases} 1 & \text{if } [\mathbf{v}]_i > 1 \\ [\mathbf{v}]_i & \text{if } 0 \leq [\mathbf{v}]_i \leq 1 \\ 0 & \text{if } [\mathbf{v}]_i < 0 \end{cases} \tag{37}$$

We can view the nonlinear operator (2) as imposing a set of inequality constraints, where the i^{th} constraint is *active* if $[\mathbf{v}]_i < 0$ or $[\mathbf{v}]_i > 1$, and *inactive* if $0 \leq [\mathbf{v}]_i \leq 1$. Operation (3) in the bottom loop updates $\mathbf{v}$ using the dynamic equation

$$\dot{\mathbf{v}} = \mathbf{T}^{\mathrm{op}}\mathbf{v} + \mathbf{i}^{\mathrm{op}} \tag{38}$$

The dynamics expressed in (38) act so as to minimize E^{op}, where

$$E^{\mathrm{op}} = -\frac{1}{2}\mathbf{v}^T\mathbf{T}^{\mathrm{op}}\mathbf{v} - \mathbf{i}^{\mathrm{op}T}\mathbf{v} \tag{39}$$

$\mathbf{T}^{\mathrm{op}}$ is further expressed as

$$\mathbf{T}^{\mathrm{op}} = \mathbf{T}^{\mathrm{pq}} + \beta\mathbf{I}^n \tag{40}$$

$\mathbf{T}^{\mathrm{pq}}$ is intimately related to the cost function we wish to minimize in the combinatorial optimization problem. If $\mathbf{v}^{\mathrm{c}}$ is the converged network state vector $\mathbf{v}$, and E^{pq} is the cost function we desire to minimize, then

$$E^{\mathrm{pq}} = -\frac{1}{2}\mathbf{v}^{\mathrm{c}T}\mathbf{T}^{\mathrm{pq}}\mathbf{v}^{\mathrm{c}} - \mathbf{i}^{\mathrm{op}T}\mathbf{v}^{\mathrm{c}} \tag{41}$$

The term $\beta\mathbf{I}^n$ is included in equation (40) in order to allow implementation of the Matrix Graduated Non-Convexity (MGNC) algorithm described in [6]. By gradually increasing β as the network converges, the state vector $\mathbf{v}$ is forced into a hypercube corner, so that the converged state vector $\mathbf{v}^{\mathrm{c}}$ satisfies $[\mathbf{v}^{\mathrm{c}}]_i \in \{1,0\}$. The MGNC algorithm also helps prevent the network from becoming trapped in sub-optimal local minima of the cost function, E^{pq}. Precise implementation details relating to the modified network can be found in [6]. Broadly speaking, however, the mode of operation is that a single traversal of the bottom loop, (3), is followed by several traversals of the top loop, (1) & (2); the whole cycle is repeated continually whilst slowly increasing β to force convergence to a hypercube corner.

Mapping the Optimization onto the Hopfield Network

The modified Hopfield network may be used to perform the combinatorial optimization (34). The converged state vector of this network, $\mathbf{v}^{\mathrm{c}}$, is used to represent the desired permutation matrix, $\mathbf{V}(\mathbf{p})$, by way of the vec function (see [7]):

$$\mathbf{v}^{\mathrm{c}} = \mathbf{v}(\mathbf{p}) = \mathrm{vec}(\mathbf{V}(\mathbf{p})^T) \tag{42}$$

Hence, if the patterns are represented by n points, a network with n^2 units is required to find the reordering $\mathbf{p}$. We must also define a matrix, $\mathbf{V}$, which is the matrix equivalent of the unconverged state vector $\mathbf{v}$:

$$\mathbf{v} = \mathrm{vec}(\mathbf{V}^T) \tag{43}$$

Consider what happens if we confine $\mathbf{V}$ so that

$$\mathbf{V} = \mathbf{R}^n \mathbf{V} \mathbf{R}^n + \mathbf{S} \tag{44}$$

If we subsequently force $\mathbf{v}$ into a hypercube corner, ie. if we enforce $[\mathbf{V}]_{ij} \in \{1,0\}$, then it is clear that $\mathbf{V}$ will satisfy the conditions for $\mathbf{V}(\mathbf{p})$ in equations (31) and (32), and will therefore represent a valid solution to the combinatorial optimization problem.

The projection loop of the modified network, described by equation (35), may be used to enforce (44). The projection parameters $\mathbf{T}^{\mathrm{val}}$ and $\mathbf{s}$ can be derived as follows:

$$\begin{array}{llll}
 & \mathbf{V} & = & \mathbf{R}^n \mathbf{V} \mathbf{R}^n + \mathbf{S} \\
\Leftrightarrow & \mathbf{V}^T & = & \mathbf{R}^n \mathbf{V}^T \mathbf{R}^n + \mathbf{S} \\
\Leftrightarrow & \mathrm{vec}(\mathbf{V}^T) & = & \mathrm{vec}(\mathbf{R}^n \mathbf{V}^T \mathbf{R}^n) + \mathrm{vec}(\mathbf{S}) \\
\Leftrightarrow & \mathrm{vec}(\mathbf{V}^T) & = & (\mathbf{R}^n \otimes \mathbf{R}^n)\mathrm{vec}(\mathbf{V}^T) + \mathrm{vec}(\mathbf{S}) \quad \text{(using (5))} \\
\Leftrightarrow & \mathbf{v} & = & (\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{v} + \mathrm{vec}(\mathbf{S}) \quad \text{(using (43))}
\end{array}$$

Hence, by comparison with (35), we have

$$\mathbf{T}^{\mathrm{val}} = \mathbf{R}^n \otimes \mathbf{R}^n \tag{45}$$

$$\mathbf{s} = \mathrm{vec}(\mathbf{S}) = \frac{1}{n}(\mathbf{o}^n \otimes \mathbf{o}^n) \tag{46}$$

For this particular problem, $\mathbf{T}^{\mathrm{val}}$ is a projection matrix which projects $\mathbf{v}$ onto a subspace in which the row and column sums of $\mathbf{V}$ are zero; this subspace is termed the "zero-sum subspace".

In this application we are aiming to maximize $q(\mathbf{p})$, the match quality, with respect to the point ordering $\mathbf{p}$. The Hopfield network minimizes E^{pq}, so if we set $E^{\mathrm{pq}} = -q(\mathbf{p})$, then the Hopfield network will carry out the desired optimization. We are now in a position to derive the form of the matrix $\mathbf{T}^{\mathrm{pq}}$:

$$\begin{aligned}
E^{\mathrm{pq}} &= -q(\mathbf{p}) \\
&= -\mathrm{trace}\left[(\mathbf{V}(\mathbf{p})\hat{\mathbf{C}}^{\mathrm{Us}})^T \mathbf{T}^{\mathrm{Ms}} (\mathbf{V}(\mathbf{p})\hat{\mathbf{C}}^{\mathrm{Us}})\right] \\
&= -\mathrm{trace}\left[(\mathbf{V}(\mathbf{p})\hat{\mathbf{C}}^{\mathrm{Us}})(\mathbf{V}(\mathbf{p})\hat{\mathbf{C}}^{\mathrm{Us}})^T \mathbf{T}^{\mathrm{Ms}}\right] \qquad \text{(using (3))}
\end{aligned}$$

$$
\begin{aligned}
\Leftrightarrow E^{\mathrm{pq}} &= -\mathrm{trace}\left[\mathbf{V}(\mathbf{p})\hat{\mathbf{C}}^{\mathrm{Us}}\hat{\mathbf{C}}^{\mathrm{Us}^T}\mathbf{V}(\mathbf{p})^T\mathbf{T}^{\mathrm{Ms}}\right] \\
&= -\left[\mathrm{vec}(\mathbf{V}(\mathbf{p})^T)\right]^T \mathrm{vec}(\hat{\mathbf{C}}^{\mathrm{Us}}\hat{\mathbf{C}}^{\mathrm{Us}^T}\mathbf{V}(\mathbf{p})^T\mathbf{T}^{\mathrm{Ms}}) && \text{(using (4))} \\
&= -\left[\mathrm{vec}(\mathbf{V}(\mathbf{p})^T)\right]^T (\mathbf{T}^{\mathrm{Ms}} \otimes \hat{\mathbf{C}}^{\mathrm{Us}}\hat{\mathbf{C}}^{\mathrm{Us}^T})\mathrm{vec}(\mathbf{V}(\mathbf{p})^T) && \text{(using (5))} \\
&= -\mathbf{v}(\mathbf{p})^T(\mathbf{T}^{\mathrm{Ms}} \otimes \hat{\mathbf{C}}^{\mathrm{Us}}\hat{\mathbf{C}}^{\mathrm{Us}^T})\mathbf{v}(\mathbf{p}) && \text{(using (42))}
\end{aligned}
$$

By comparison with (41), we see that

$$
\begin{aligned}
\mathbf{T}^{\mathrm{pq}} &= 2(\mathbf{P} \otimes \mathbf{Q}) && (47) \\
\mathbf{P} &= \mathbf{T}^{\mathrm{Ms}} && (48) \\
\mathbf{Q} &= \hat{\mathbf{C}}^{\mathrm{Us}}\hat{\mathbf{C}}^{\mathrm{Us}^T} && (49) \\
\mathbf{i}^{\mathrm{op}} &= \mathbf{0} && (50)
\end{aligned}
$$

Results and Discussion

As a small illustrative experiment, the network was used in a pattern recognition system to recognize instances of the ten handwritten digits. The template and unknown digits used in the experiment are shown in Figure 2. A point positioning algorithm was used to place 20 points around the outlines of the digits; these 20 points represent the digits in the recognition scheme. The results are displayed in Table 1. Figure 3 shows the point reordering the network found for the unknown '3' compared with the template '3'.

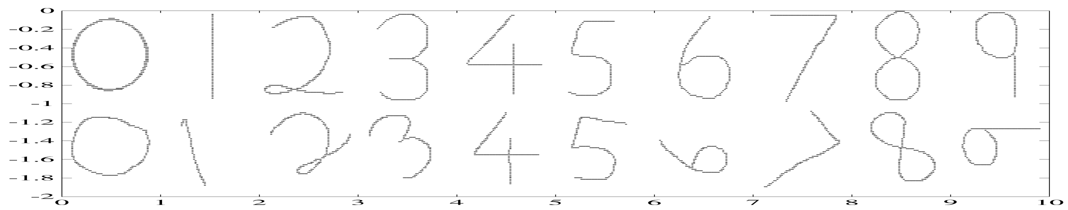


Figure 2: Template (top) and unknown (bottom) digits used in the experiment.

Unknown	**Template Digit**									
Digit	0	1	2	3	4	5	6	7	8	9
0	**1.98**	1.53	1.81	1.87	1.73	1.87	1.83	1.60	1.90	1.84
1	1.98	**1.99**	1.97	1.98	1.96	1.96	1.98	1.99	1.98	1.98
2	1.83	1.66	**1.93**	1.76	1.80	1.78	1.84	1.82	1.81	1.81
3	1.80	1.82	1.86	**1.94**	1.83	1.87	1.88	1.82	1.86	1.84
4	1.72	1.76	1.77	1.87	**1.88**	1.83	1.76	1.63	1.82	1.74
5	1.82	1.65	1.82	1.88	1.85	**1.91**	1.90	1.65	1.88	1.84
6	1.81	1.79	1.91	1.93	1.82	1.89	1.96	1.86	1.84	**1.97**
7	1.70	1.70	1.82	1.81	1.89	1.73	1.87	**1.98**	1.72	1.90
8	1.93	1.84	1.86	1.94	1.78	1.90	1.88	1.70	**1.96**	1.85
9	1.81	1.73	1.84	1.87	1.82	1.85	1.93	1.85	1.80	**1.96**

Table 1: Match qualities for all templates against all unknowns; winning scores are shown in bold type.

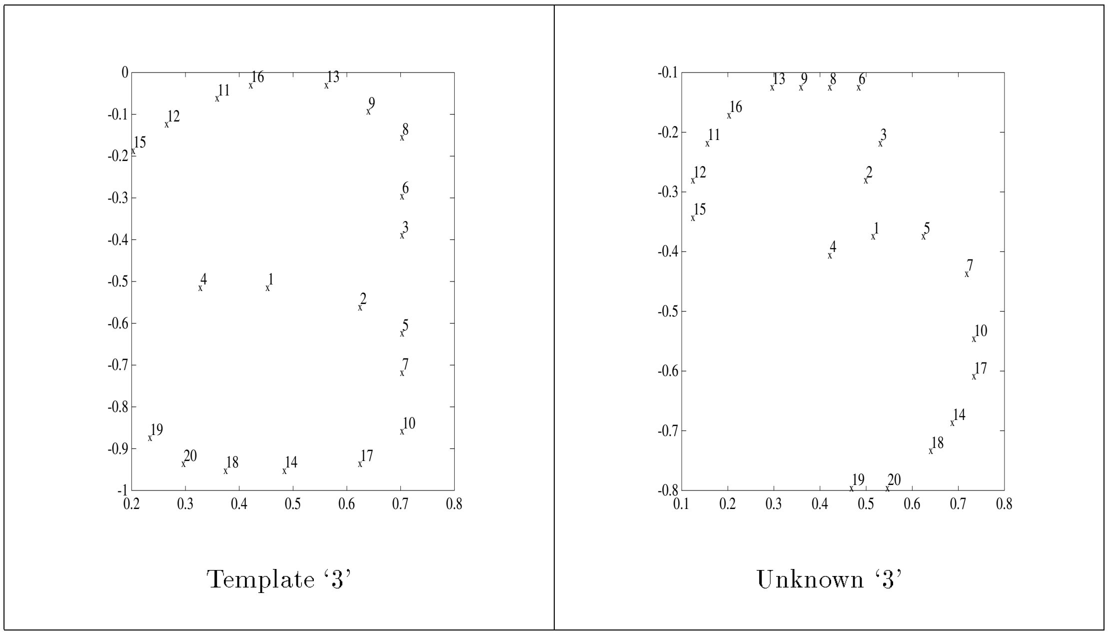


Figure 3: Point ordering found by the network in attempting to match an unknown ‘3’ to a template ‘3’.

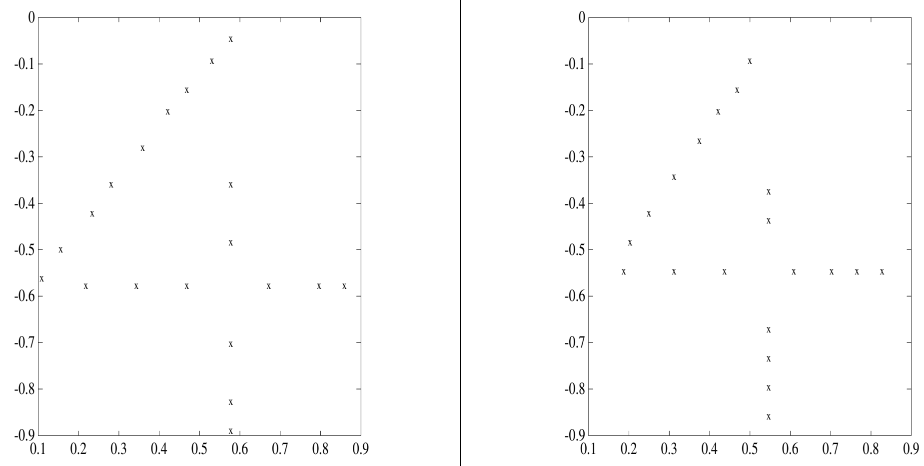
Not surprisingly, 6’s and 9’s are easily confused, a result of the rotational invariance of the scheme. 1’s have a high match quality against all templates, because a linear transformation of the form $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ will map any template onto a straight line of points. The lower match qualities for digits of the same class occurred when the point positioning algorithm placed different numbers of points on similar features of the templates and unknowns: for an example of this, see Figure 4.

More interesting than the results themselves is the way in which the Hopfield network performs the optimization. Figure 5 shows the output of the network during various stages of convergence as it attempts to reorder the points in the unknown ‘3’ to give a good match against the ‘5’ template. The quantities $q_{\rm I}$ and $\hat{q}_{\rm I}$, given in the figure, are the intermediate match quality and normalized intermediate match quality respectively:

$$q_{\rm I} = (\mathbf{V}\hat{\mathbf{x}}^{\rm Us})^T \mathbf{T}^{\rm M} (\mathbf{V}\hat{\mathbf{x}}^{\rm Us}) + (\mathbf{V}\hat{\mathbf{y}}^{\rm Us})^T \mathbf{T}^{\rm M} (\mathbf{V}\hat{\mathbf{y}}^{\rm Us}) \tag{51}$$

$$\hat{q}_{\rm I} = \frac{(\mathbf{V}\hat{\mathbf{x}}^{\rm Us})^T \mathbf{T}^{\rm M} (\mathbf{V}\hat{\mathbf{x}}^{\rm Us})}{\left|\mathbf{V}\hat{\mathbf{x}}^{\rm Us}\right|^2} + \frac{(\mathbf{V}\hat{\mathbf{y}}^{\rm Us})^T \mathbf{T}^{\rm M} (\mathbf{V}\hat{\mathbf{y}}^{\rm Us})}{\left|\mathbf{V}\hat{\mathbf{y}}^{\rm Us}\right|^2} \tag{52}$$

Before any of the nonlinear constraints (36) become active, the strategy is to find a linear combination of the 3’s coordinates which forms a straight line of points; such a pattern, as mentioned above, scores very well against all templates. As some nonlinearities become active, the network is no longer able to do this, and instead succeeds in using the 3’s coordinates to produce a pattern reminiscent of the ‘5’ template, again with a high normalized match quality. As $\mathbf{v}$ is forced into a hypercube corner, the network is likewise forced to make do with the coordinates it has got (instead of linear combinations of them), and simply reorders them to give the best match quality against the template. The intermediate behaviour of the network gives a valuable insight into how the network performs the optimization; such knowledge will be useful when attempting to improve the performance of the network, or derive efficient mappings for other optimization problems.



Template ‘4’ — five points on vertical stroke

Unknown ‘4’ — six points on vertical stroke

Figure 4: Template and unknown ‘4’, showing different numbers of points on the vertical strokes, resulting in a poor match quality.

Conclusions

A pattern recognition scheme, which uses a method of subspace projection to compare templates with unknowns, has been described. The scheme is invariant to a wide variety of linear transformations, but requires that the points representing the template and unknown be ordered in the same manner. However, invariance to point ordering has been added through the use of a modified Hopfield network as an optimization tool. The process of mapping the particular combinatorial optimization problem onto the network has been illustrated. The results of the invariant pattern recognition scheme are encouraging, though more interesting is the intermediate behaviour of the network as it solves the optimization problem. Knowledge of this behaviour will prove valuable when attempting to improve the performance of the network, or derive efficient mappings for other optimization problems.

References

[1] Hopfield, J. J. & Tank, D. W. *“Neural” Computation of Decisions in Optimization Problems*, Biological Cybernetics 52, 141-152, 1985.

[2] Wilson, V. & Pawley, G. S. *On the Stability of the TSP Problem Algorithm of Hopfield and Tank*, Biological Cybernetics 58, 63-70, 1988.

[3] Peterson, C. & Söderberg, B. *A new method for mapping optimization problems onto neural networks*, International Journal of Neural Systems, Vol. 1, No. 1, 1989.

[4] Van den Bout, D. E. & Miller III, T. K. *Graph Partitioning using Annealed Neural Networks*, IEEE Transactions on Neural Networks, Vol. 1, No. 2, June 1990.

[5] Aiyer, S. V. B., Niranjan, M. & Fallside, F. *A Theoretical Investigation into the Performance of the Hopfield Model*, IEEE Transactions on Neural Networks, Vol. 1, No. 2, June 1990.

[6] Aiyer, S. V. B. & Fallside, F. *A Subspace Approach to Solving Combinatorial Optimization Problems with Hopfield Networks*, Cambridge University Engineering Department Technical Report no. CUED/F-INFENG/TR 55, December 1990.

[7] Graham, A. *Kronecker Products and Matrix Calculus: with Applications*, Ellis Horwood Ltd, Chichester 1981.

[8] Gohberg, I., Lancaster, P. & Rodman, L. *Invariant Subspaces of Matrices with Applications*, Canadian Mathematical Society of Monographs and Advanced Texts, John Wiley and Sons, New York 1986.

[9] Rosenbrook, H. H. & Storey, C. *Mathematics of Dynamical Systems*, Thomas Nelson and Sons, London 1970.

Appendix

Improving the Speed of Convergence

In this appendix we carefully analyze the operation of the network in the early stages of convergence. In particular, we consider the operating region in which $\mathbf{v}$ is contained within the bounds of the unit hypercube, and has not yet come to any of the hypercube's faces: ie. before any of the inequality constraints (36) have become active. We start by deriving an expression for $\dot{\mathbf{v}}^{\mathrm{zs}}$, the component of $\dot{\mathbf{v}}$ which lies in the zero-sum subspace.

$$\begin{aligned} \dot{\mathbf{v}}^{\mathrm{zs}} &= \mathbf{T}^{\mathrm{val}}\dot{\mathbf{v}} = \mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{op}}\mathbf{v}) \\ &= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}(\mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s}) \end{aligned} \tag{53}$$

Hence we see that $\dot{\mathbf{v}}^{\mathrm{zs}}$ is made up of two parts, a constant term, $\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{s}$, and a term which depends on $\mathbf{v}$, $\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}}\mathbf{v}$. Let us consider the constant term first:

$$\begin{aligned} \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{s} &= \mathbf{T}^{\mathrm{val}}(-\mathbf{T}^{\mathrm{pq}} + \beta\mathbf{I}^{n})\mathbf{s} \\ &= -\mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{Ms}} \otimes \hat{\mathbf{C}}^{\mathrm{Us}}\hat{\mathbf{C}}^{\mathrm{Us}^T})\mathbf{s} + \frac{\beta}{n}(\mathbf{R}^{n} \otimes \mathbf{R}^{n})(\mathbf{o}^{n} \otimes \mathbf{o}^{n}) \\ &= -\frac{1}{n}\mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{Ms}} \otimes \hat{\mathbf{C}}^{\mathrm{Us}}\hat{\mathbf{C}}^{\mathrm{Us}^T})(\mathbf{o}^{n} \otimes \mathbf{o}^{n}) \\ &= -\frac{1}{n}\mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{Ms}}\mathbf{o}^{n} \otimes \hat{\mathbf{C}}^{\mathrm{Us}}\hat{\mathbf{C}}^{\mathrm{Us}^T}\mathbf{o}^{n}) \quad \text{(using (6))} \end{aligned} \tag{54}$$

But

$$\hat{\mathbf{C}}^{\mathrm{Us}} = \frac{\mathbf{R}^{n}\mathbf{C}^{\mathrm{U}}}{\left|\mathbf{R}^{n}\mathbf{C}^{\mathrm{U}}\right|} \tag{55}$$

and so

$$\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{s} = -\frac{1}{n\left|\mathbf{R}^{n}\mathbf{C}^{\mathrm{U}}\right|^{2}}\mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{Ms}}\mathbf{o}^{n} \otimes \mathbf{R}^{n}\mathbf{C}^{\mathrm{U}}\mathbf{C}^{\mathrm{U}^T}\mathbf{R}^{n}\mathbf{o}^{n}) = 0 \tag{56}$$

So we see that the constant term in $\dot{\mathbf{v}}^{\mathrm{zs}}$ is zero, and we are left with

$$\dot{\mathbf{v}}^{\mathrm{zs}} = \mathbf{A}\mathbf{v} = \mathbf{A}\mathbf{v}^{\mathrm{zs}} \tag{57}$$

$$\text{where } \mathbf{A} = \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}} \tag{58}$$

The general solution of (57) can be expressed by means of the matrix exponential (see [9]):

$$\mathbf{v}^{\mathrm{zs}}(t) = e^{\mathbf{A}t}\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}} = \sum_{k=0}^{\infty}\frac{t^{k}}{k!}\mathbf{A}^{k}\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}} \tag{59}$$

where $\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}}$ is the initial value of $\mathbf{v}^{\mathrm{zs}}$. When using the Hopfield Network as an optimization tool, $\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}}$ is usually set to be a small, random vector.

Suppose $\mathbf{A}$ has eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{n^2}$, with associated eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n^2}$. Let $\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}}$ be decomposed along $\mathbf{A}$'s eigenvectors as follows:

$$\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}} = \sum_{i=1}^{n^2} c_i \mathbf{u}_i \tag{60}$$

Then we have

$$\begin{aligned} \mathbf{A}^k \mathbf{v}_{\mathrm{o}}^{\mathrm{zs}} &= \sum_{i=1}^{n^2} c_i \lambda_i^k \mathbf{u}_i & (61) \\ \Leftrightarrow \mathbf{v}^{\mathrm{zs}}(t) &= \sum_{k=0}^{\infty} \frac{t^k}{k!} \sum_{i=1}^{n^2} c_i \lambda_i^k \mathbf{u}_i = \sum_{i=1}^{n^2} e^{\lambda_i t} c_i \mathbf{u}_i & (62) \end{aligned}$$

Equation (62) tells us that $\mathbf{v}^{\mathrm{zs}}(t)$ tends to get lined up with the eigenvector of $\mathbf{A}$ which has the corresponding largest positive eigenvalue: let us call this eigenvector the "dominant eigenvector", and the corresponding eigenvalue the "dominant eigenvalue". In this application, the dominant eigenvalue is degenerate[2], with a corresponding 2-dimensional dominant eigenspace. If we start the network running with an initial random $\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}}$, which will have a random component in the dominant eigenspace, then the network dynamics will tend to emphasize this random component, and hence the network is unlikely to head towards the global optimum in the linear phase of convergence. Although the network is quite capable of making substantial corrections once some of the nonlinear constraints (36) have become active, both a faster convergence and a better quality solution can be obtained by ensuring that the network heads in the right direction from the outset.

We can break the degeneracy of the dominant eigenvalue by using a pseudo-projection matrix, $\mathbf{T}^{\mathrm{Ms}'}$, instead of $\mathbf{T}^{\mathrm{Ms}}$, in the formulation of $\mathbf{T}^{\mathrm{pq}}$:

$$\mathbf{T}^{\mathrm{Ms}'} = \hat{\mathbf{C}}^{\mathrm{M}} \hat{\mathbf{C}}^{\mathrm{M}^T} \tag{63}$$

Using $\mathbf{T}^{\mathrm{Ms}'}$ instead of $\mathbf{T}^{\mathrm{Ms}}$ results in $\mathbf{A}$ having a non-degenerate dominant eigenvalue[3], and also gives a cost function E^{pq} which is very similar to the cost function obtained using $\mathbf{T}^{\mathrm{Ms}}$. In particular, the global optimum obtained with $\mathbf{T}^{\mathrm{Ms}'}$ consistently lies at a hypercube corner close (in terms of Hamming distance) to that obtained with $\mathbf{T}^{\mathrm{Ms}}$. As soon as any of the constraints (36) become active, the above analysis is no longer valid, and the true projection matrix $\mathbf{T}^{\mathrm{Ms}}$ can be substituted for $\mathbf{T}^{\mathrm{Ms}'}$ in the formulation of $\mathbf{T}^{\mathrm{pq}}$. There is still a small problem in that the component of $\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}}$ parallel to the dominant eigenvector may be positive or negative, so $\mathbf{v}$ can still go off randomly in one of two directions. Hence, to ensure the network finds the global optimum, we have to run the network twice, starting the first run with $\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}}$, and the second with $-\mathbf{v}_{\mathrm{o}}^{\mathrm{zs}}$. We are then free to choose the best solution of the two by direct evaluation of the solution quality, $q(\mathbf{p})$.

[2] The proof of this is rather protracted, but suffice it to say that the obvious degeneracy of the nonzero eigenvalue of the projection matrix, $\mathbf{T}^{\mathrm{Ms}}$, carries through to the dominant eigenvalue of $\mathbf{A}$.

[3] Again, the proof is protracted, but notice that $\mathbf{T}^{\mathrm{Ms}'}$ has no nonzero degenerate eigenvalues, which results in $\mathbf{A}$ having a non-degenerate dominant eigenvalue.

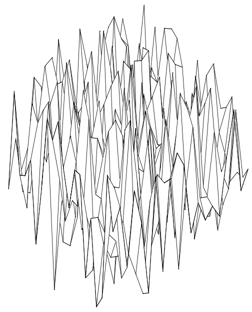
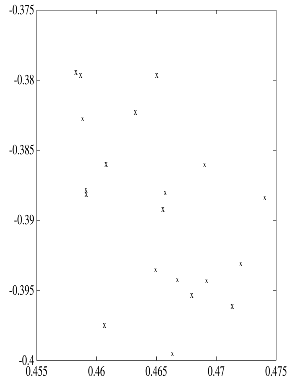
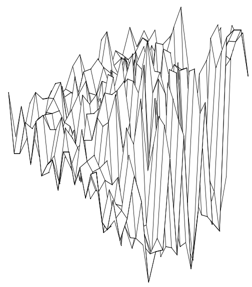
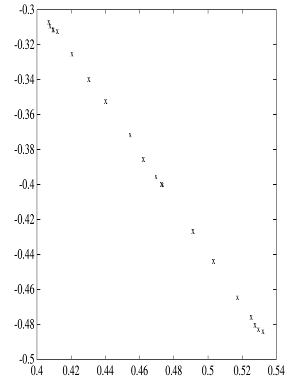
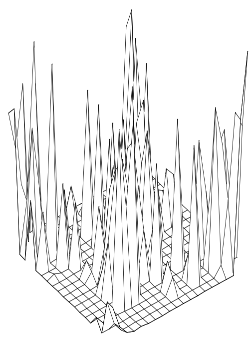
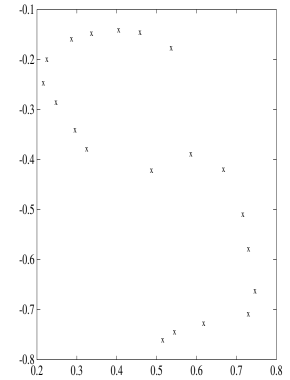
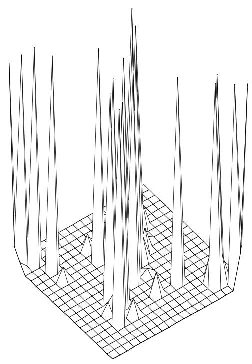
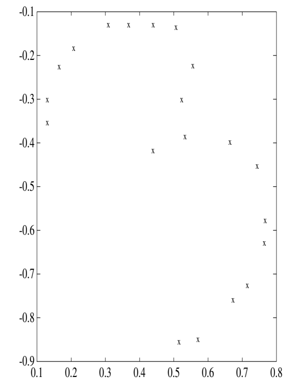
Iteration Parameters	Mesh plot of $\mathbf{V}$	$(\mathbf{V}\mathbf{x}^{\mathrm{U}})$ against $(\mathbf{V}\mathbf{y}^{\mathrm{U}})$
Iteration 1 $\beta = -0.995$ $q_{\mathrm{I}} = 0.0003$ $\hat{q}_{\mathrm{I}} = 0.435$		
Iteration 300 $\beta = -0.995$ $q_{\mathrm{I}} = 0.135$ $\hat{q}_{\mathrm{I}} = 1.999$		
Iteration 900 $\beta = -0.867$ $q_{\mathrm{I}} = 1.808$ $\hat{q}_{\mathrm{I}} = 1.993$		
Iteration 3100 $\beta = 1.241$ $q_{\mathrm{I}} = 2.052$ $\hat{q}_{\mathrm{I}} = 1.869$		

Figure 5: Intermediate behaviour of the network as it attempts to match an unknown '3' to a template '5'.