A HOPFIELD NETWORK IMPLEMENTATION OF THE VITERBI ALGORITHM FOR HIDDEN MARKOV MODELS

S.V.B.Aiyer & F.Fallside

CUED/F-INFENG/TR 60

June 17, 1992

Cambridge University Engineering Department

Trumpington Street

Cambridge CB2 1PZ

England

Email: svb10 / fallside @dsl.eng.cam.ac.uk

Submitted to IJCNN-91-Seattle

Abstract

Treating the Viterbi algorithm as a form of combinatorial optimization, this paper shows how it can be implemented on a Hopfield network. The implementation uses a framework developed in our previous papers [1, 2] which ensures the network can achieve valid solutions for a much larger class of combinatorial optimization problem than previously considered. This class includes dynamic programming problems of the type represented by the Viterbi algorithm. The aim here is to present in detail the actual mapping required to implement the Viterbi algorithm on the Hopfield network, together with an analysis and justification of it. Finally, to confirm the theory, results are presented which show the Hopfield network achieving the same solution as a standard dynamic programming based Viterbi algorithm, for a recognition task based on a pre-trained 10 state Hidden Markov model.

Introduction and Outline

Hopfield and Tank's 1985 paper [3] established the Hopfield network, as an important alternative method of solving combinatorial optimization problems. Although many researchers have since proposed a wide variety of uses for the network, most of these applications have been to solve highly artificial combinatorial optimization problems, such as the Travelling Salesman and Graph Partitioning problems. Even in these cases it has been found that the network rarely achieves valid solutions to the problem it is being used to solve [4, 5]. This problem of reliability has been addressed in [1, 2] and a framework for using the network with guaranteed reliability has been developed. A consequence of this framework (see [2]) is the possibility of using the network to solve Dynamic Programming type combinatorial optimization problems like the Viterbi algorithm [6]. This algorithm has a direct application to Hidden Markov Models, specifically for recognition tasks such as those that occur in speech processing.

In this paper the possibility of using the Hopfield network to solve dynamic programming problems is developed into an actual implementation of the Viterbi algorithm. The description of the implementation is divided into three sections. The first introduces the key concepts and notation conventions relevant to the Hidden Markov Model and the Viterbi algorithm. The second section develops and justifies the expressions required to map the Viterbi algorithm onto the parameters of the Hopfield network. Finally the last section presents and discusses the experimental results.

The Hidden Markov Model and Viterbi Algorithm

Let t be a discrete time variable where: $t \in \{1, 2, \dots, M\}$.

Let $q_1, q_2, \dots, q_N$ be the states of an HMM with N states.

Let $\mathbf{A}$ be the $N \times N$ matrix of interstate transition probabilities: $A_{ij} = \Pr(q_j \text{ at } t{+}1 | q_i \text{ at } t)$

Let $\omega_1, \omega_2 \dots, \omega_K$ be a set of K output symbols.

Let $\mathbf{B}$ be the $N \times K$ output symbol probability matrix: $B_{ij} = \Pr[w_j \text{ at } t | q_i \text{ at } t]$

Let $\mathbf{O}$ be a M element vector which denotes a sequence of M output symbols: $O_t = k$ if the output symbol at time t is ω_k

Let $\mathbf{p}$ be a M element vector which denotes a sequence of M states: $p_t = i$ if at time t the HMM is in state q_i

The Viterbi algorithm seeks to find a sequence of states $\mathbf{p}$ which maximizes the joint likelihood that the HMM generated $\mathbf{O}$ with state sequence $\mathbf{p}$. Let $\mathcal{L}(\mathbf{O}, \mathbf{p}|\mathbf{A}, \mathbf{B})$ denote this joint likelihood. Using this expression the Viterbi algorithm reduces to:

$$\arg \max_{\mathbf{p}} \mathcal{L}(\mathbf{O}, \mathbf{p}|\mathbf{A}, \mathbf{B})$$

Note that by Bayes theorem, $\mathcal{L}(\mathbf{O}, \mathbf{p}|\mathbf{A}, \mathbf{B}) = \mathcal{L}(\mathbf{O}|\mathbf{A}, \mathbf{B}, \mathbf{p})\mathcal{L}(\mathbf{p}|\mathbf{A}, \mathbf{B})$, and that:

$$\mathcal{L}(\mathbf{O}|\mathbf{A}, \mathbf{B}, \mathbf{p}) = \prod_{t=1}^{M} B_{p_t o_t} \qquad (1)$$

$$\mathcal{L}(\mathbf{p}|\mathbf{A}, \mathbf{B}) = \prod_{t=1}^{M-1} A_{p_t p_{t+1}} \qquad (2)$$

Mapping of the Viterbi Algorithm for HMMs onto a Hopfield Network

This mapping has two basic requirements:

(i) A vector of ones and zeros (i.e a vector that corresponds to a hypercube corner) has to be found that can uniquely represent the state sequence $\mathbf{p}$. Let this vector be denoted $\mathbf{v}(\mathbf{p})$.

(ii) A quadratic Liapunov function of the network output vector $\mathbf{v}$ has to be found, which is of the form $E^{op}(\mathbf{v}) = -\frac{1}{2}\mathbf{v}^T\mathbf{T}^{op}\mathbf{v} - \mathbf{i}^{op\,T}\mathbf{v}$. The matrix $\mathbf{T}^{op}$ and vector $\mathbf{i}^{op}$ must be such that $E^{op}(\mathbf{v}(\mathbf{p}))$ is a monotonic function of $-\mathcal{L}(\mathbf{O},\mathbf{p}|\mathbf{A},\mathbf{B})$. In other words

$$\text{if}\quad \mathcal{L}(\mathbf{O},\mathbf{p}^1|\mathbf{A},\mathbf{B}) \geq \mathcal{L}(\mathbf{O},\mathbf{p}^2|\mathbf{A},\mathbf{B}) \quad\text{then}\quad E^{op}(\mathbf{v}(\mathbf{p}^1)) \leq E^{op}(\mathbf{v}(\mathbf{p}^2))$$

Assuming $\mathbf{T}^{op}$ is symmetric, it can be shown that the Hopfield network will gradually change $\mathbf{v}$ so as to minimize $E^{op}(\mathbf{v})$. Further, using the technique of valid subspace confinement developed in our previous paper [2], it can also be ensured that whenever a final solution is reached, $\mathbf{v}$ is of the form $\mathbf{v}(\mathbf{p})$. Minimizing $E^{op}(\mathbf{v})$ subject to $\mathbf{v} = \mathbf{v}(\mathbf{p})$ is equivalent to maximizing $\mathcal{L}(\mathbf{O},\mathbf{p}|\mathbf{A},\mathbf{B})$ over $\mathbf{p}$, hence it is possible for the Hopfield network to perform the same optimization operation as the Viterbi algorithm.

A general expression for $\mathbf{v}(\mathbf{p})$

Let $\boldsymbol{\delta}^p(N)$ be the N dimensional co-ordinate vector given by,

$$[\boldsymbol{\delta}^p(N)]_i = \delta_{pi} = \begin{cases} 1 & \text{if} \quad i = p \\ 0 & \text{if} \quad i \neq p \end{cases} \quad p \in \{1, 2, \ldots, N\} \tag{3}$$

In effect $\boldsymbol{\delta}^p(N)$ is the p^{th} column of the $N \times N$ identity matrix. **N.B.** The (N) part of the vector $\boldsymbol{\delta}^p(N)$, specifies the number of elements in this vector and the range of p. If this is obvious from the context of use, then it will be omitted, leaving just $\boldsymbol{\delta}^p$.

The NM dimensional vector $\mathbf{v}(\mathbf{p})$, which corresponds to the state sequence denoted by $\mathbf{p}$, can now be defined as follows:

$$\mathbf{v}(\mathbf{p}) = \begin{bmatrix} \boldsymbol{\delta}^{p_1}(N) \\ \boldsymbol{\delta}^{p_2}(N) \\ \cdots \\ \boldsymbol{\delta}^{p_M}(N) \end{bmatrix} \tag{4}$$

An alternative way of representing $\mathbf{p}$ is by a $N \times M$ matrix. Let such a matrix be denoted $\mathbf{V}(\mathbf{p})$, and be defined by:

$$\mathbf{V}(\mathbf{p}) = [\ \boldsymbol{\delta}^{p_1}(N) \quad \boldsymbol{\delta}^{p_2}(N) \quad \cdots \quad \boldsymbol{\delta}^{p_M}(N)\ ] \quad \text{or} \quad [\mathbf{V}(\mathbf{p})]_{ij} = [\boldsymbol{\delta}^{p_j}(N)]_i \tag{5}$$

For a 4 state HMM, with the sequence of states $q_1q_2q_3q_3q_4q_4$, the vectors $\mathbf{p}$ and $\mathbf{v}(\mathbf{p})$ plus the matrix $\mathbf{V}(\mathbf{p})$ would expand as follows,

$$\begin{array}{ccc} \mathbf{p}^T & \mathbf{v}(\mathbf{p})^T & \mathbf{V}(\mathbf{p}) \\ [1,2,3,3,4,4] & \begin{array}{cccccc} [1\,0\,0\,0 & 0\,1\,0\,0 & 0\,0\,1\,0 & 0\,0\,1\,0 & 0\,0\,0\,1 & 0\,0\,0\,1] \\ [\boldsymbol{\delta}^{p_1 T} & \boldsymbol{\delta}^{p_2 T} & \boldsymbol{\delta}^{p_3 T} & \boldsymbol{\delta}^{p_4 T} & \boldsymbol{\delta}^{p_3 T} & \boldsymbol{\delta}^{p_4 T}] \end{array} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

The function vec() (see [7]), which concatenates the columns of a matrix into one vector, can be used to map the $N \times M$ matrix $\mathbf{V}$ to a NM element vector $\mathbf{v}$:

$$\mathbf{v} = \text{vec}(\mathbf{V}) \Rightarrow v_k = V_{ij} \text{ where } k = N(j-1) + i$$

From the definitions of $\mathbf{v}(\mathbf{p})$ and $\mathbf{V}(\mathbf{p})$ given in (4)(5), it can be seen that $\mathbf{v}(\mathbf{p}) = \text{vec}(\mathbf{V}(\mathbf{p}))$.

Expressions for $\mathbf{T}^{op}$ and $\mathbf{i}^{op}$

The expressions for $\mathbf{T}^{op}$ and $\mathbf{i}^{op}$ rely on the use of the Kronecker product notation developed in [2] (see [7] for detailed definitions and proofs of the Kronecker product identities), with $\mathbf{T}^{op}$ being given by an expression of the form:

$$\mathbf{T}^{op} = \beta\mathbf{I} + \mathbf{T}^{pq} + \mathbf{T}^{pq^T} \quad \text{where} \quad \mathbf{T}^{pq} = (\mathbf{P}\otimes\mathbf{Q}) = \begin{bmatrix} P_{11}\mathbf{Q} & P_{12}\mathbf{Q} & \cdots & P_{1M}\mathbf{Q} \\ P_{21}\mathbf{Q} & P_{22}\mathbf{Q} & \cdots & P_{2M}\mathbf{Q} \\ \cdots & \cdots & \cdots & \cdots \\ P_{M1}\mathbf{Q} & P_{M2}\mathbf{Q} & \cdots & P_{MM}\mathbf{Q} \end{bmatrix}$$

and where $\mathbf{P}$ is a $M \times M$ matrix ,$\mathbf{Q}$ is a $N \times N$ matrix.
Let the NM element vector $\mathbf{u}$ and the $N \times M$ matrix $\mathbf{U}$ be related by: $\mathbf{u} = \text{vec}(\mathbf{U})$. From the identity (see [7]):

$$(\mathbf{P}\otimes\mathbf{Q})\mathbf{v} = \text{vec}(\mathbf{QVP}^T) \quad \text{where } \mathbf{v} = \text{vec}(\mathbf{V}) \tag{6}$$

it follows that if $\mathbf{U} = \mathbf{QV(p)P}^T$ then $\mathbf{u} = (\mathbf{P}\otimes\mathbf{Q})\mathbf{v(p)}$ since $\mathbf{v(p)} = \text{vec}(\mathbf{V(p)})$. Hence:

$$\mathbf{v(p)}^T\mathbf{T}^{pq}\mathbf{v(p)} = \mathbf{v(p)}^T\mathbf{u} = \sum_{i=1}^{NM}[\mathbf{v(p)}]_i[\mathbf{u}]_i = \sum_{i=1}^{N}\sum_{j=1}^{M}[\mathbf{V(p)}]_{ij}U_{ij} \tag{7}$$

Consider the value of $E^{pq} = \frac{1}{2}\mathbf{v(p)}^T\mathbf{T}^{pq}\mathbf{v(p)}$ for the case where:

$$P_{ji} = \begin{cases} 1 & \text{if } i-j=1 \\ 0 & \text{otherwise} \end{cases} \qquad i,j \in \{1,2,\ldots,M\} \tag{8}$$

$$Q_{xy} = \log(A_{xy}) \qquad x,y \in \{1,2,\ldots,N\} \tag{9}$$

Using (5) and (3) allows $\mathbf{U} = \mathbf{QV(p)P}^T$ to be written in component form:

$$U_{kl} = \sum_{i=1}^{N}\sum_{j=1}^{M}\mathbf{Q}_{ki}[\mathbf{V(p)}]_{ij}P_{lj} = \sum_{i=1}^{N}\sum_{j=1}^{M}\log(A_{ki})\boldsymbol{\delta}_{p_ji}P_{lj} = \sum_{j=1}^{M}\log(A_{kp_j})P_{lj}$$

Since $P_{lj} = 1$ only if $j = l+1$ and $P_{lj} = 0$ otherwise: $U_{kl} = \begin{cases} \log(A_{kp_{l+1}}) & \text{if } l = 1\ldots M-1 \\ 0 & \text{if } l = M \end{cases}$

But from (7) $E^{pq} = \sum_{i=1}^{N}\sum_{j=1}^{M}[\mathbf{V(p)}]_{ij}U_{ij}$, hence since $[\mathbf{V(p)}]_{ij} = \delta_{p_ji}$:

$$\begin{aligned} E^{pq} &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\begin{cases} [\mathbf{V(p)}]_{ij}\log(A_{ip_{j+1}}) & \text{if } j = 1\ldots M-1 \\ 0 & \text{if } j = M \end{cases} \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M-1}\delta_{p_ji}\log(A_{ip_{j+1}}) = \frac{1}{2}\sum_{j=1}^{M-1}\log(A_{p_jp_{j+1}}) \end{aligned}$$

From (2) $\mathcal{L}(\mathbf{p}|\mathbf{A},\mathbf{B}) = \prod_{t=1}^{M-1} A_{p_tp_{t+1}}$ hence $E^{pq} = \frac{1}{2}\log(\mathcal{L}(\mathbf{p}|\mathbf{A},\mathbf{B}))$. This means:

$$\begin{aligned} \tfrac{1}{2}\mathbf{v(p)}^T\mathbf{T}^{op}\mathbf{v(p)} &= \tfrac{1}{2}\mathbf{v(p)}^T(\beta\mathbf{I} + \mathbf{T}^{pq} + \mathbf{T}^{pq^T})\mathbf{v(p)} \\ &= \tfrac{1}{2}\beta|\mathbf{v(p)}| + \tfrac{1}{2}\mathbf{v(p)}^T\mathbf{T}^{pq}\mathbf{v(p)} + \tfrac{1}{2}(\mathbf{v(p)}^T\mathbf{T}^{pq}\mathbf{v(p)})^T \\ &= \beta|\mathbf{v(p)}| + \tfrac{1}{2}\log(\mathcal{L}(\mathbf{p}|\mathbf{A},\mathbf{B})) + \tfrac{1}{2}\log(\mathcal{L}(\mathbf{p}|\mathbf{A},\mathbf{B})) \end{aligned}$$

$$\Rightarrow \tfrac{1}{2}\mathbf{v(p)}^T\mathbf{T}^{op}\mathbf{v(p)} = \beta M + \log(\mathcal{L}(\mathbf{p}|\mathbf{A},\mathbf{B})) \tag{10}$$

Now let the NM element vector $\mathbf{i}^{op}$ be related to the $N \times M$ matrix $\mathbf{I}^{op}$ by: $\mathbf{i}^{op} = \text{vec}(\mathbf{I}^{op})$. Further let:

$$[\mathbf{I}^{op}]_{ij} = \log(B_{iO_j}) \tag{11}$$

Noting from (1) that $\mathcal{L}(\mathbf{O}|\mathbf{A}, \mathbf{B}, \mathbf{p}) = \prod_{j=1}^{M} B_{p_j O_j}$ it can now be seen that:

$$\mathbf{i}^{op\,T}\mathbf{v}(\mathbf{p}) = \text{vec}(\mathbf{I}^{op})^T \text{vec}(\mathbf{V}(\mathbf{p})) = \sum_{i=1}^{N}\sum_{j=1}^{M} \log(B_{iO_j})\delta_{p_j i} = \sum_{j=1}^{M} \log(B_{p_j O_j}) = \log(\mathcal{L}(\mathbf{O}|\mathbf{A}, \mathbf{B}, \mathbf{p})) \tag{12}$$

Putting (10) and (12) together:

$$\begin{aligned} E^{op}(\mathbf{v}(\mathbf{p})) &= -\tfrac{1}{2}\mathbf{v}(\mathbf{p})\mathbf{T}^{op}\mathbf{v}(\mathbf{p}) - \mathbf{i}^{op\,T}\mathbf{v}(\mathbf{p}) = -\tfrac{1}{2}\beta M - \log(\mathcal{L}(\mathbf{p}|\mathbf{A}, \mathbf{B})) - \log(\mathcal{L}(\mathbf{O}|\mathbf{A}, \mathbf{B}, \mathbf{p})) \\ \Rightarrow E^{op}(\mathbf{v}(\mathbf{p})) &= -\tfrac{1}{2}\beta M - \log(\mathcal{L}(\mathbf{O}, \mathbf{p}|\mathbf{A}, \mathbf{B})) \end{aligned}$$

Thus, as required, $E^{op}(\mathbf{v}(\mathbf{p}))$ is a monotonic function of $-\mathcal{L}(\mathbf{O}, \mathbf{p}|\mathbf{A}, \mathbf{B})$. (**N.B.** The β parameter is used to ensure convergence to hypercube corner. For a discussion of its role see [2])

Enforcement of $\mathbf{v} = \mathbf{v}(\mathbf{p})$ by confinement to the valid subspace

Following the analysis developed in [2], the enforcement of $\mathbf{v} = \mathbf{v}(\mathbf{p})$ is achieved by confining the network output $\mathbf{v}$ to a subspace - the valid subspace - such that if $\mathbf{v}$ is a hypercube corner that lies in the valid subspace, then it must be of the form $\mathbf{v}(\mathbf{p})$.

The general equation of the valid subspace is:

$$\mathbf{v} = \mathbf{s} + \mathbf{T}^{zs}\mathbf{v} \quad \text{with} \quad \mathbf{T}^{zs} = (\mathbf{I} \otimes \mathbf{R})$$

where $\mathbf{s}$ is a constant vector and $\mathbf{T}^{zs}$ is a projection matrix, which projects $\mathbf{v}$ onto the zerosum subspace, $\mathbf{I}$ is the $M \times M$ identity matrix and $\mathbf{R}$ is a $N \times N$ matrix given by: $R_{ij} = \delta_{ij} - \frac{1}{N}$.

Let $\mathbf{v}^{zs} = \mathbf{T}^{zs}\mathbf{v} = \text{vec}(\mathbf{V}^{zs})$ and $\mathbf{s} = \text{vec}(\mathbf{S})$ where $\mathbf{V}^{zs}$ and $\mathbf{S}$ are $N \times M$ matrices. Using the identity (6) it can be seen that:

$$\mathbf{v}^{zs} = \text{vec}(\mathbf{V}^{zs}) = \text{vec}(\mathbf{RV})$$

Multiplication by $\mathbf{R}$ has the effect of setting the column sums of a matrix to zero (for proof see [2]), hence the column sums of $\mathbf{V}^{zs}$ are always zero. Now let $\mathbf{S}$ be given by $S_{ij} = \frac{1}{N}$. If $\mathbf{v} = \mathbf{s} + \mathbf{v}^{zs}$ then $\mathbf{V} = \mathbf{S} + \mathbf{V}^{zs}$ hence if $\mathbf{v}$ satisfies the valid subspace equation then $\mathbf{V}$ must have column sums equal to $\sum_{i=1}^{N} S_{ij} = 1$. Clearly if $\mathbf{v}$ is a hypercube corner (i.e a vector of ones and zeros) and $\mathbf{V}$ has a column sum of one, then $\mathbf{V}$ must be of the form $\mathbf{V}(\mathbf{p})$ and hence $\mathbf{v} = \mathbf{v}(\mathbf{p})$.

Implementation on a Hopfield network and Results of a Simple Experiment

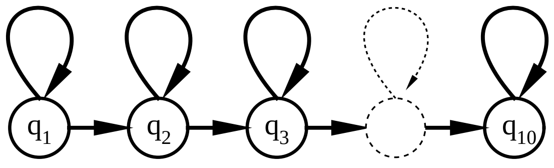
The simplest method of implementation is to set the connection matrix $\mathbf{T}$ and input bias $\mathbf{i}^b$ of the network as proposed in [3] according to equations (40) and (43) of [2]. Thus:

$$\mathbf{T} = \theta(\mathbf{T}^{zs} - \mathbf{I}) + \mathbf{T}^{op} \quad \text{and} \quad \mathbf{i}^b = \theta\mathbf{s} + \mathbf{i}^{op} \quad \text{where } \theta \text{ is variable positive parameter} \tag{13}$$

However, for reasons of efficiency it is better to use the modified network proposed in [2] and shown in Fig 2. This was the case for the simple experiment that was performed to confirm the theory behind the mapping of the Viterbi algorithm.

The actual experiment consisted of finding the optimum state sequence for a 10 state left to right HMM (shown in Fig 1) to generate a sequence of 20 output symbols.

As a simplification, the total number of possible output symbols was restricted to 20 and the output sequence was assumed to be $\omega_1\omega_2\cdots\omega_{20}$: hence $\mathbf{O} = [1, 2, \ldots, 20]^T$. The transition matrix $\mathbf{A}$ is shown in Fig 1 and the output probability matrix $\mathbf{B}$ was generated randomly subject to the constraint that the sum of each row was one (i.e the total output probability per state was one). In addition, to ensure that the HMM started in state q_1 and ended in state q_{10}, all the elements of the first and last rows and columns of $\mathbf{B}$ were set to zero, except for the element B_{11} and B_{NM} which where set to one. Fig 3 shows the evolution of the network output from an initial random state to the final solution.



$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0 & \cdots & 0 \\ 0 & 0.5 & 0.5 & \cdots & 0 \\ 0 & 0 & 0.5 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0.5 \end{bmatrix}$$

Figure 1: Diagram of left to right 10 state HMM with a 10×10 state transition matrix $\mathbf{A}$

Discussion and Conclusion

The output shown in Figure 3 clearly shows the network achieving a valid solution. Further the state sequence represented by this solution, $q_1q_2q_3q_4q_5q_6q_6q_6q_6q_6q_7q_7q_8q_8q_9q_9q_9q_9q_9q_{10}$ is in fact the optimum, since a standard dynamic programming based Viterbi algorithm gives exactly the same state sequence. This experiment provides a useful illustration of some of the features intrinsic to the Hopfield network approach. One of these is the ability of the network to keep active more than one solution. For example, at iteration 200 it can be seen that two partial solutions are active simultaneously, of which one eventually dominates. In addition, from (12) and (10) it can be seen that the mapping used neatly splits up the joint likelihood function into an output sequence part, $\mathcal{L}(\mathbf{O}|\mathbf{A},\mathbf{B},\mathbf{p})$, which is handled by the input bias term $\mathbf{i}^{op}$, and a state sequence part $\mathcal{L}(\mathbf{p}|\mathbf{A},\mathbf{B})$ which is handled by the quadratic term $\mathbf{v}(\mathbf{p})\mathbf{T}^{op}\mathbf{v}(\mathbf{p})$. Since the input bias term is an external input to the network, it is highly appropriate that it carries all the information about the external output sequence which the HMM is trying to recognise. Overall, these features, together with the obvious advantage of the network's inherent parallel structure, make this mapping of the Viterbi algorithm onto the Hopfield network not just a curiosity but a serious application with significant potential.

References

[1] Aiyer,S.V.B, Niranjan,M, Fallside,F, *A Theoretical Investigation into the performance of the Hopfield Model* IEEE Trans. Neural Networks, Vol NN-1, Issue 2, p204-215, **1990**.

[2] Aiyer,S.V.B, Niranjan,M, Fallside,F, *A Subspace Approach to Solving Combinatorial Optimization Problems using Hopfield Networks* Cambridge Univ. Engineering Dept. Tech. Report CUED/F-INFENG/TR-55 **1990**

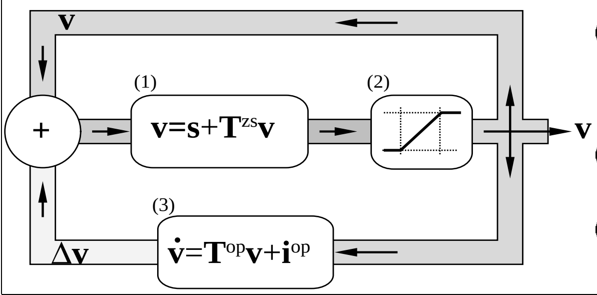
[3] Hopfield,J.J, Tank,D.W, *Neural Computation of Decisions in Optimization Problems* Biological Cybernetics. 52, 1-25, **1985**.

[4] Wilson,V, Pawley,G.S *On the Stability of the TSP Problem Algorithm of Hopfield and Tank* Biological Cybernetics 58, p63-70 **1988**.

[5] Kahng,A, *Traveling Salesman Heuristics and Embedding Dimension in the Hopfield Model* Proc. IJCNN 89, Vol I, p513-520, **1989**.

[6] Levinson,S, *Structural Methods in Automatic Speech Recognition* Proc. IEEE, Vol. 73, No. 11, Nov. **1985**.

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



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

$$\mathbf{v} = \mathbf{s} + \mathbf{T}^{zs}\mathbf{v} \text{ where } \mathbf{T}^{zs} = (\mathbf{I} \otimes \mathbf{R})$$

(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 2: Schematic diagram of modified network implementation

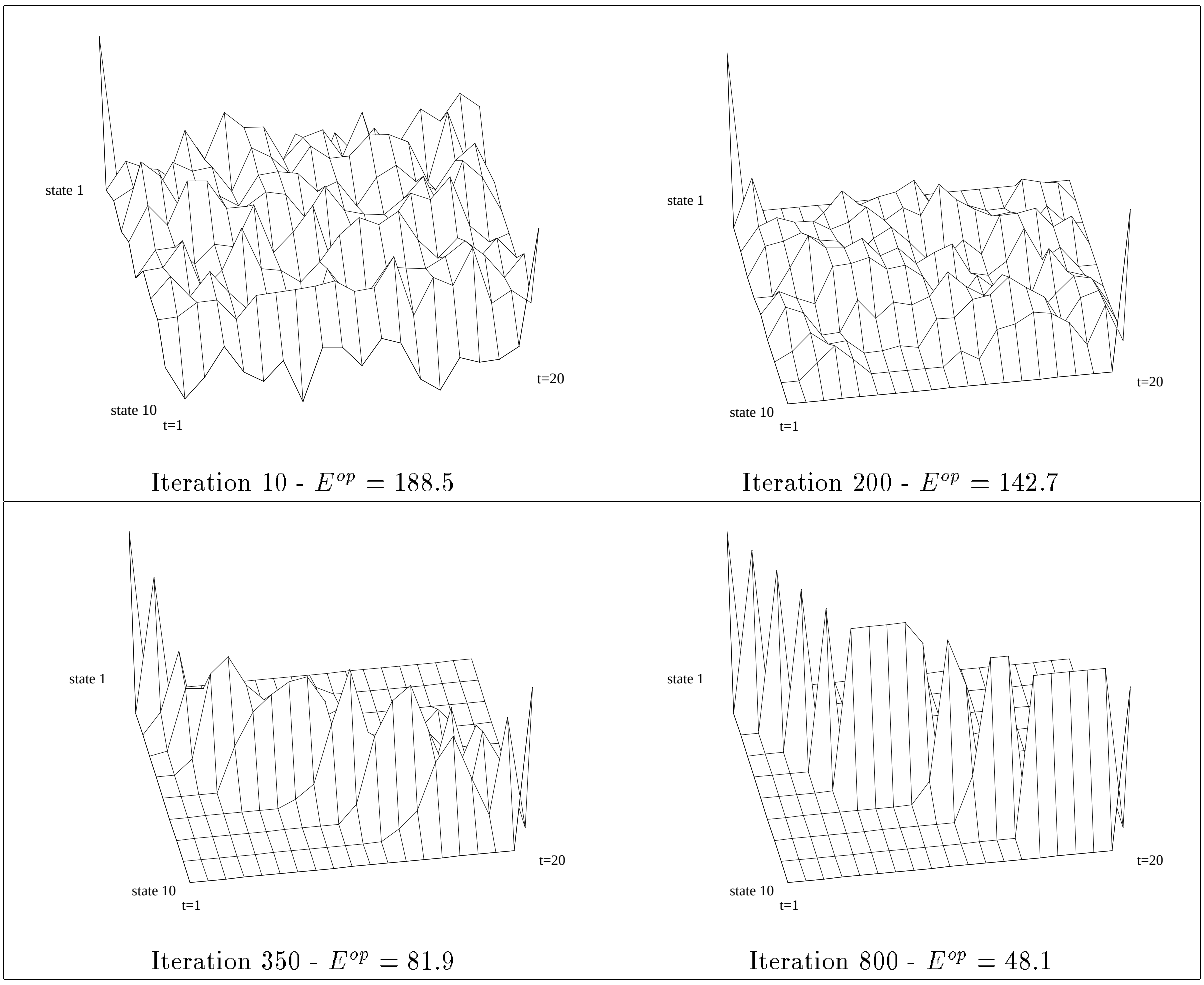


Figure 3: 3D Mesh Plots showing the evolution of $\mathbf{V}$ (the matrix representation of the network output vector $\mathbf{v}$, where $\mathrm{vec}(\mathbf{V}) = \mathbf{v}$) in a simulation of the modified network implementing the Viterbi algorithm for a 10 state HMM and sequence of 20 output symbols. The final solution at iteration 800 is identical to the solution obtained by dynamic programming, i.e. it is the global optimum. (N.B. The vertical axis scale changes for each mesh plot.)