International Journal of Computers & Technology
Volume 4 No. 1, Jan - Feb, 2013

A COMPARATIVE STUDY OF VARIOUS METHODS OF ANN FOR SOLVING TSP PROBLEM

Prof. Sharadindu Roy
University of Calcutta

Prof Samer Sen Sarma
University of Calcutta

Soumyadip chakravorty
University of Calcutta

Suvodip Maity
University of Calcutta

Abstract

This paper represents TSP (Travelling Salesman Problem) by using Artificial Neural Networks. A comparative study of various methods of ANN is shown here for solving TSP problem.The Travelling Salesman Problem is a classical combinational optimization problem, which is a simple to state but very difficult to solve. This problem is to find the shortest possible tour through a set of N vertices so that each vertex is visited exactly once. TSP can be solved by Hopfield Network, Self-organization Map, and Simultaneous Recurrent Network. Hopfield net is a fully connected network, where every vertex is connected with each other forwardly and backwardly. So starting the walk from a vertex we can travel all the other vertex exactly once and return to starting vertex again.

Key Word

TSP : Travelling Salesman Problem

ANN : Artificial Neural Network

HN : Hopfield Network

EN : Elastic Net

SRN : Simultaneous Recurrent Network

Introduction

A traveling sales person must visit a number of cities, each only once. Moving from one city to other have a cast e.g. the intercity distance associated the cost/distance traveled must be minimized. The salespersons have returned the starting point.

An implementation of an algorithm in neural network for an approximate solution for Traveling Salesman's Problem. TSP is a classical example of optimization and constrain satisfaction problem which falls under the family of NP-complete of problems. Here used continuous Hopfield network to find the solution for given problem. Hopfield Network is constructed from artificial neuron.

The traveling salesman problem (TSP) is well known in optimization. The TSP problem is NP-complete problem. There is no algorithm for this problem, which gives a perfect solution. Thus any algorithm for this problem is going to be impractical with certain examples.

Here we assume that we are given n cities, and a non-negative integer distance Dij between any two cities I and j. We try to find the tour for the salesman that best fit the above mentioned criterion. There are various neural Network algorithm that can be used to try to solve such constrain satisfaction problems. Most solution have used one of the following method:

Hopfield Network

Elastic Net

Simultaneous Recurrent Network

1.Hopfield Network

In 1983, a physicist John Hopfield published the famous paper "Neural network and physical system with emergent collective computational abilities". Along with the rediscovery of Backpropagation and the introduction of cheap computing power, this helped to reignite the dormant world of Neural Networks once again.

A Hopfield artificial neural network is a type of recurrural network artificial network that is used to store one or more stable vectors. These stable vectors can be viewed as memories that the network recalls when provided with similar vectors that can act as a cue to the network memory. The binary units only take two different values for their states that are determined by whether or not the unit input exceeds their threshold. Binary unit can take either values of 1 or -1, or values of 1 or 0. Consequently there are two possible definitions for binary unit activation.

Basic Operation

Hopfield was to add feedback connection to the network(the outputs are fed back into the inputs) and show that with these connections the network are capable of interesting behaviors which we might not expected of them, in particular they can holds memories. Networks with such connections are called "feedback" or "recurrent" networks.

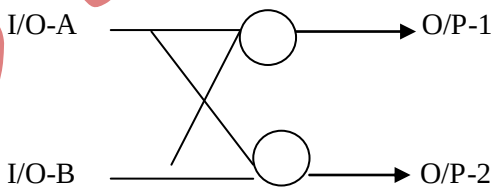


Fig: Free Forward Network

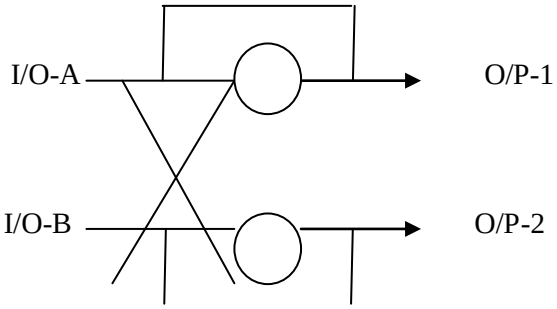


Fig: Free Forward Network with Feedback connection

Network with such connection are called "feedback" or "recurrent" networks.

Hopfield networks are constructed from artificial neurons. These artificial neurons have N inputs. With each input i there is a weight Wi associated. They also have an output. The state of the output is maintained until the neuron is updated. Updating the neuron entails the following operations:

The value of each input, Xi is determined and the weighted sum of all inputs, Σi Wi Xi , is calculated.

The output state of the neuron is set to +1 if the weighted input sum is larger or equal to 0. It is set to -1 if the weighted sum is smaller than 0.

A neuron retains its output state until it is updated again.

Written as a formula:

$O = 1 : \Sigma i\ Wi\ Xi \geq 0$

$: \Sigma i\ Wi\ Xi < 0$

A Hopfield network is a network of N such artificial neurons, which are fully connected. The connection weight from neuron j to neuron I is given by a number wij. The collection of such number is represented by the weighted matrix W, whose components are Wij.

Now given the weighted matrix and the updating rule for neurons the dynamics of the network is defined if we tell in which order we update the neurons. There are two ways of updating them:

Asynchronous: One picks one neuron, calculates the weighted input sum and update immediately. This can be done in a fixed order, or neurons can be picked at random, which is called asynchronous random updating.

Synchronous: The weighted input sums of all neurons are calculated without updating the neurons. Then all neurons are set to their new value, according to the value of their weight input sum.

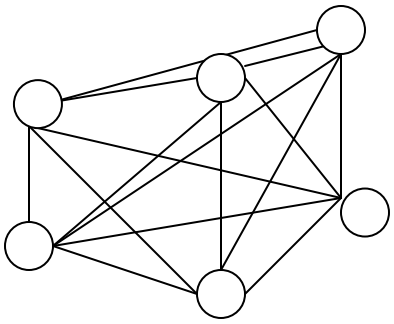


Figure : A Hopfield network

A Hopfield Network as an autoassociator.

There are two type of Hopfield Network:

1. The Discrete Hopfield Model
2. The Continuous Hopfield-Tank Model.

The Discrete Hopfield Model:

The original Hopfield neural network model is a fully inter connected network of binary units with symmetric connection weights between the units. The connections weights are not learned but are defined a priori from problem data (the inter city distances in a TSP context).Starting from some arbitrarily chosen initial configuration, either feasible or infeasible, the Hopfield network evolves by updating the activation of each unit in turn (i.e. an activated unit can be turned off, and an inactivated unit can be turned on). The update rule of any given unit involves the activation of the units it is connected to as well as the weights on the connections. Via this update process, various configurations. In this final state, all units are stable according to the update rule and do not change their activation status.

The dynamics of the Hopfield network can be described formally in mathematical terms. To this end, the activation levels of the binary units are set to zero and one for "off" and "on" respectively. Starting from some initial configuration {vi}i=1,…….L. where L is the number of units and Vi is the activation level of unit i ,the network relaxes to a stable configuration according to the following rule

set Vi to 0 if $\Sigma\ TijVi < \theta i$

set Vi to 1 if $\Sigma\ TijVi > \theta i$

do not change if $\Sigma\ TijVi = \theta i$

where Tij is the connection weight between units i and j, and θi is the threshold of unit i.

The behavior of the network can be characterized by an appropriate energy function. The energy E depends only on the activation level Vi (the weight Tij thresholds θi are fixed and derived from problem data), and is such that it can only decrease as the network evolves over time.

The energy functionis given by:

$E=-1/2\ \Sigma i\Sigma jTijViVj+\Sigma i\theta iVi$ ……..1 Since the connection weights Tij are symmetric, each term TijViVj appears twice within the double summation of equ(1). Hence, this double summation is divided by 2.

It is easy to show that a unit changes its activation level if and only if the energy of the network decreases by doing so. In order to prove that statement, we must consider the

contribution Ei of a given unit i to the overall energy E, that is,

Ei= - ΣjTijViVj+θiVi

consequently.

If Vi to 1 then Ei= -ΣjTijViVj+θiVi

If Vi to 0 then Ei= 0

Hence, the change in energy due to a change ΔVi in the activation level of unit i is:

ΔE=-ΔVi(ΣjTijVi- θi).

Now, ΔVi is one if unit I changed its activation level from zero to one, and such a change can only occur if the expression between the parentheses is positive. As a consequence, ΔEi is negative and the energy decreases. This same line of reasoning can be applied when a unit i changes its activation level from one to zero(i.e.,ΔVi=-1). Since the energy can only decreases over time and the number of configuration is finite, the network must necessarily converge to a stable state (but not necessarily the minimum energy state). In the next section, a natural extension of this model to units with continuous activation levels is described.

The Continuous Hopfield-Tank Model:

In Hopfield and Tank extended the original to a fully inter connected network of nonlinear analog units, where the activation level of each unit is a value in the interval [0, 1].

The main motivation of Hopfield and Tank for extending the discrete network to a continuous one was to provide a model that be easily implemented using simple analog hardware. However, it seems that continuous dynamics also feasibility convergence.

The evolution of the units over time is now characterized by the following differential equations (usually called "equations of motion")

dui/dt=ΣjTijVi+Ii-Ui,i=1,…..L.(2) where Ui ,Ii and Vi are the input, input bias, and activation level of unit i, respectively. The activation level of unit I is a function of its input, manly

Vi = g(Ui) = 1/2 (tan(h) Ui/Uo)

=1/(1+e^-2Ui/Uo) ,…….(3).

The activation function g is the well-known sigmoidal function.

3.The Eleastic Net

Durbin and Willshaw's (1987) elastic net algorithm for solving the traveling salesperson problem (TSP) is based on a method for developing topology preserving maps between the eye and brain (lateral geniculate nucleus and cortex for mammals) due to von der Malsburg and Willshaw(1977) and Willshaw and von Malsburg(1979).

The elastic net approach is fundamentally different from the approach of Hopfield and Tank. It is a geometrically inspired algorithm, and is closely related to the self organizing map.

3.Sumultaneous Recurrent Network

A learning based recurrent neural search algorithm is expected to offer significant performance improvements over a non- learning based algorithm. One such neural paradigm, the Simultaneous Recurrent Neural Network (SRN) incorporates powerful features: it is a recurrent algorithm with relaxation search capability, while also begin trainable. The simultaneous Recurrent network has the potential to develop, through " learning", the ability to address the computationally challenging task of large-scale static optimization. This forms a very important first step towards eventually addressing dynamic optimization problems through algorithm that can formulate learning based solution.

A Simultaneous Recurrent (SRN) is an artificial neural network [werbos 1994; Pang and werbos,1997]with the graphical representation as in figure.

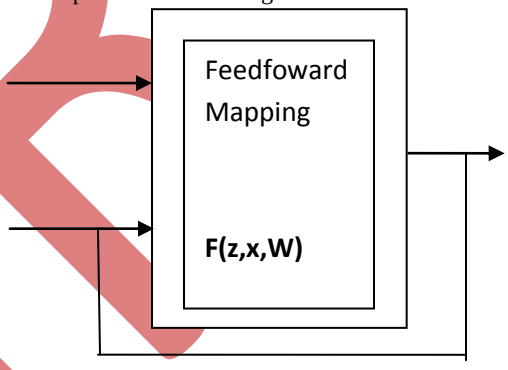


fig: Simultaneous Recurrent Network graphical representation

The system has external inputs in the form of a vector x, a feed-forward vector function F. (any feed-forward network, including the multi-layer perceptron is appropriate), outputs in the form of a vector z, and a feedback path which copies the output to the input without a time delay.

The feed-forward network F will also induce a wait matrix w, which represent the inter connection topology of the network. The network starting from an initial state as indicated by initial value of the output vector, will iterate until the output vector z stabilizes and converges to a stable point given that one exists. In other terms, an SRN is based on a feed-forward network with simultaneous feedback from output of the network to its input. An SRN exhibits complex temporal behavior: it follows a trajectory in the state.

Space to relax to a fixed point. One "relaxation" of the network consists of one or more Iterations of output computation and propagation along the feed-forward and feedback paths until the outputs converge to a stable equilibrium value. A more formal description of SRN is formulated in [Werbos; 1992] who defines an SRN as a mapping

Z=F(x,W)……..(1a)

where x and W are the external inputs and weights, respectively and z is the equilibrium value of e.g. :

z=lim za……..(1b).

which can be computed by the following iteration

zn-1 =F(zn ,x.W)……..(1c).

where F is a feedforward network and n is the iteration index with very fast computation cycle compare to feedback delays found in the time lagged recurrent networks.

The network is provided with the external inputs and initial outputs, which are typically assumed randomly in the absence of a priori information. The output of previous iteration is fed back to the network along with the external inputs to compute the output of next iteration.

The network is allowed to iterate until it reaches a stable equilibrium point, assuming at least one exists. External inputs are applied throughout the complete relaxation cycle. When a stable equilibrium point is reached, the outputs stop changing (i.e. the output value of Z(n+1) is most equal to or very close to Z(n)). It is important to note that the feedback from the output layer to the input in SRN is not delayed: the feedback is, theoretically speaking, simultaneous.

Solution Methodology
1.Hopfield Network

Here used continuous Hopfield network to find the solution for given problem. Hopfield Network is constructed from artificial neuron.

These artificial neurons haven inputs. With each input i there is a weight wi associated. They also have an output. The state of the output is maintained, until the neuron is updated. Updating the neuron entails the following operations:

The value of each input, xi is determined and the weight sum of all input, ∑wi xi is calculated.

The output state of the neuron is set to +1 if the weighted input sum is larger or equal to 0. It is set to -1 if the weighted input sum is smaller than 0.

A neuron retains its output state until it is updated again.

Written as formula:

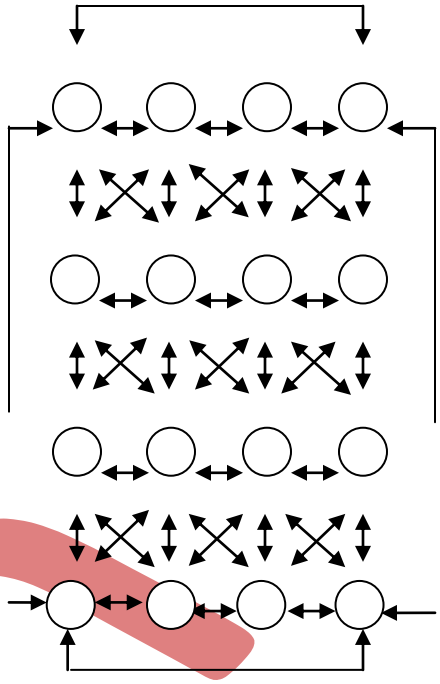
O = 1 : wixi >=0

: wixi <=0

TSP of Hopfield Network

Hopfield network is a dynamic network, which iterates to converge from an arbitrary input state. The Hopfield Network as minimizing an energy function.

The Hopfield net is fully connected network. It is a weighted network where the output of the network is fed back and there are weights to each of this link. The fully connected is shown in following figure:



Here we use n2 neuron in the network, where n is the total number of cities. The neurons here have a threshold and step function. The inputs are given to the weighted input node. The network then calculates the output and then based on Energy function and weight update function, converges to the stable solution after few iteration. The most important task on hand is to find an appropriate connection weight. It should be such that invalid tour should be prevented and valid tour should be preferred.

The output result of TSP can be represented as following. The example here is for 4 cities. The 4 cities TSP need 16 neurons.

	1	2	3	4
A	0	1	0	0
B	1	0	0	0
C	0	0	0	1
D	0	0	1	0

Figure: Tour Matrix obtained as the output of the network.

The corresponding visiting route, in the above example is City B→City A→City D→City C→City B.

So the total distance is: D=AB+BC+CD+DA

Network Input

The inputs to this network are chosen arbitrarily. The initial state of the network is thus not fixed and is not biased against any particular route. If as a consequence of the choice of the inputs, the activation works out to give outputs that add up to the number of cities, and initial solution for the problem, a legal tour will result. Also there are inputs that are taken from user. The user is asked to input the number of cities user want to travel and the distance between those cities which are used to generate the distance matrix.

The distance matrix in n*n square matrix whose principal diagonal is zero. The figure below shows a typical distance matrix is shown below:

	1	2	3	4
A	0	15	13	17
B	15	0	14	2 7
C	13	14	0	25
D	17	27	25	0

Fig: *distance matrix generated after getting inputs from user*

Here the distance between city A and city C is 13 and distance between a city itself is zero.

Energy Function

The Hopfield network for the application of the neural network can be best understood by the energy function. The energy function that is developed by Hopfield and Tank is used for the project. The energy function has various hollows that represent the patterns stored in the network. An unknown input pattern represent a particular point in the energy landscape and the pattern iterates its way to a solution, the point moves through the landscape towards one of the hollows. The iteration is carried on till some fixed number of time or till the stable state in reached.

The energy function used should satisfy the following criterions:

The energy function should be able to lead to a stable combination matrix.

The energy function should lead to the shorter traveling path.

The energy function used for the Hopfield neural network is

$$E=A/2\sum nx=1\sum ni=1\sum nj=1,j\neq I \quad vxivxj$$
$$+B/2\sum ni=1\sum nx=1\sum ny\neq xvxivyi$$

$$+C/2(\sum nx=1\sum ni=1vxi-n)2$$

$$+D/2\sum nx=1\sum ny=1,y\neq x\sum ni=1dxyvxi(vy,i+1+vy,i-1)$$

A,B,C,D are the constants, to be determined by trial and error

If all vxi approach either 0 or 1,and if those with vxi $\approx$1 represents a Hamiltonian circuit, then

dxyvxi(vy,i+1+vy,i-1)=dxy (if city y is either before or after city x in the circuit)

= 0 otherwise.

Row Term

(A/2∑nx=1∑ni=1∑nj=1,j≠I vxivxj) in the energy function the first triple sum is zero if and only if there is only one "1" in each order column. Thus this takes care that no two or more cities are in same travel order. i.e. no two cities are visited simultaneously.

Column Term

(B/2∑ni=1∑nx=1∑ny≠xvxivyi) in the energy function the 2nd triple sum is zero if and only if there is only one city appears in each order column. Thus this takes care that each city is visited only once.

Total number of "1" term

(C/2(∑nx=1∑ni=1vxi-n)2) the third triple sum is zero if and only N number of 1 appearing in the whole n*n matrix. Thus this takes into care that all cities are visited.

The first three summation are up to satisfy the condition 1, which is necessary to produce a legal traveling path.

Shortest distance term

(D/2∑nx=1∑ny=1,y≠x∑ni=1dxyvxi(vy,i+1+vy,i-1)) the fourth triple summation provides the term for the shortest path. D is the distance between city i and city j. the value of this term is minimum when the total distance traveled shortest.

THE HOPFIELD TSP ALGORITHM

Steps1: Initialization n is the number of cities, D[]n×n is the distance matrix, and I[]n×n is the identity matrix. tour=0.

Steps2: Using identity matrix we create i number of tour matrix where tour matrix is T[]n×n. And the total tour matrix is s=n!.P[s]=T[]n×n .

Steps3: for j=1 upto s

Steps4: P[s]

Steps5: for x=1 upto n

Steps6: for y=1 upto n

Set i=y

Steps7: if T[x][1] equal to 1 then

Set s1=x;

Steps8: if T[x][n] equal to 1 then

Set s1=x;

Steps9: if i-1 equal to 0 then

Set i-1=s1

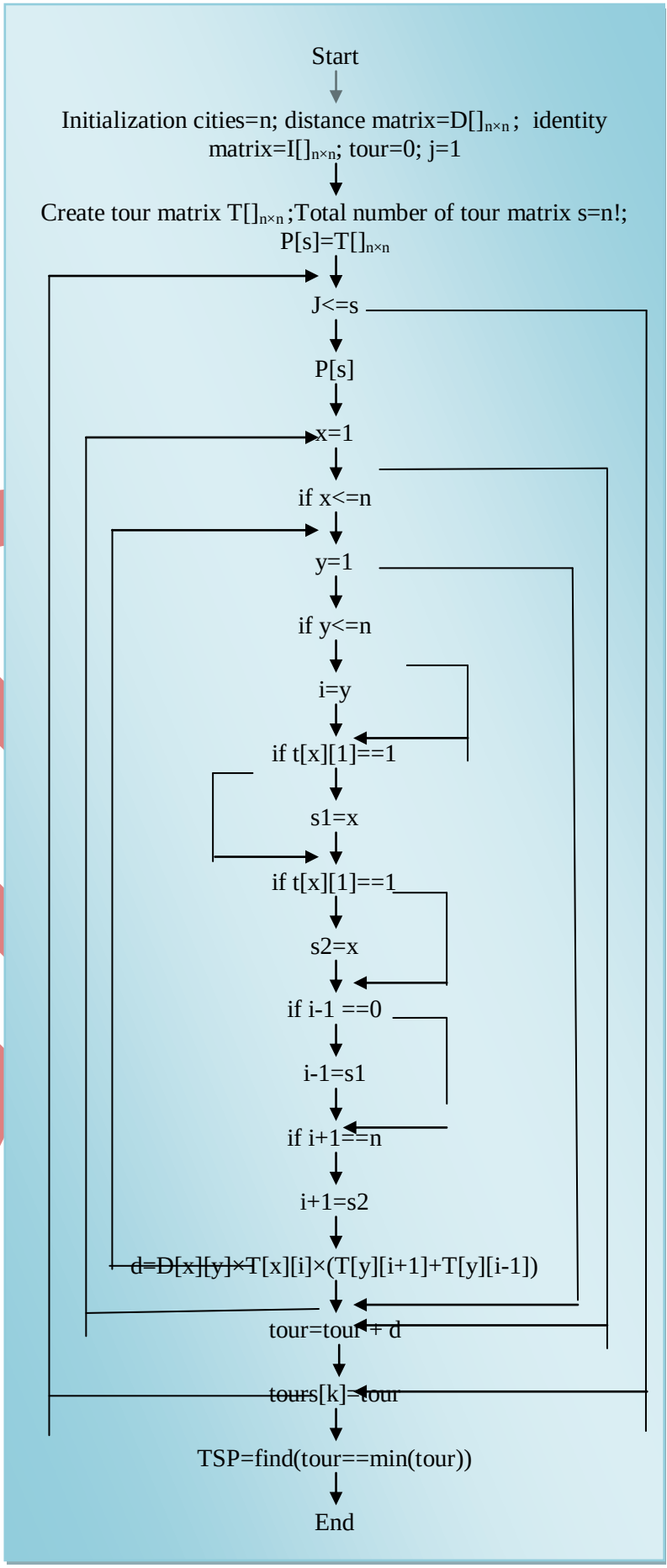
Steps10: if i+1 garter then n then

Set i+1=s2

Steps11:d=D[x][y]×T[x][i]×(T[y][i+1]+T[y][i-1])

Steps12: tour=tour + d

Steps13: end



Steps14: end

Steps15: tours[k]=tour

Steps16: end

Steps17: set TSP=find(tour==min(tour))

Steps18: end

2.The Elastic Network

The Elastic Net algorithm is an iterative procedure when M points , with M larger than the number of cities N, are like on a circular ring or " rubber band" originally located at the center of the cities. The rubber band is gradually elongated until it passes sufficiently near each city to define a tour. During that process two forces apply:

One for minimizing the length of the ring.

The other one for minimizing the distance between the cities and points on the ring.

This forces are gradually adjusted as procedure evolves.

The solution can be done by the following equation:

$\Delta Yj=\alpha\sum iWij(xi-yj)+\beta K(yj+1+yj-1-2yj)$

j=1,……M

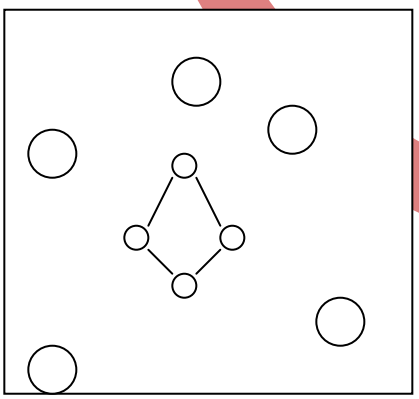
$Wij=\Phi(dxiyj,k)/\sum k\Phi(dxiyk,k)$

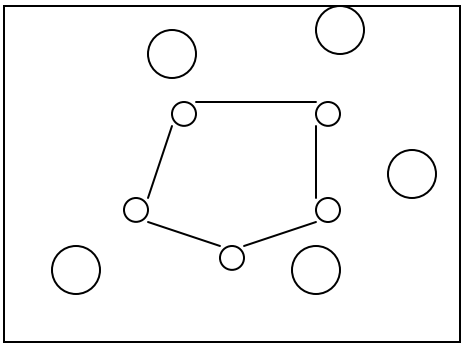
α and β are the constant parameter.

K is the "scale" parameter.

Wij is a measure of the "influence" of city I on point j.

The α term drives the points on the ring towards the cities, so that for each city I there is at least one ring point j within distance k. the β term is aimed at keepingneighboring points together so as to produce a short tour.





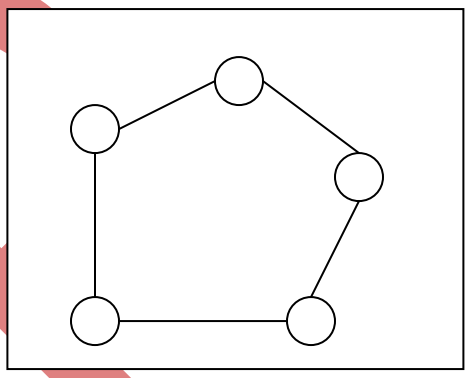


Fig: Evolution of elastic net over time and final tour.

The energy function $E=k(dE/dy_j)$

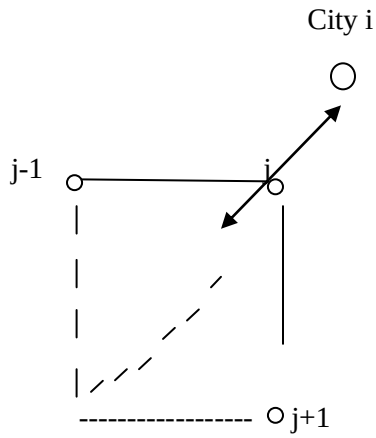


fig: forces that impact ring point j

Here, in contrast to the TSP problem, the circular form of rings is predefined. It is convenient to represent the net geometrically not as a set of nodes. But as a continuous curve in order to keep the circular form. Since there are no preferences on the ring parameters, the first evolution rule can be formulated as :

*the net has no internal

Forces.

In case of special preferences on the ring parameters(such as elliptical form) additional constraints can be easily introduced.

The second distinction from TSP is that the desired ring does not pass through all hits but only through a maximum possible number of them. Therefore the problem of single ring finding is divided into two steps: recognition of ring hits and further reconstruction of ring parameters.

At the recognition step a special energy function is used to separate ring hits from others.

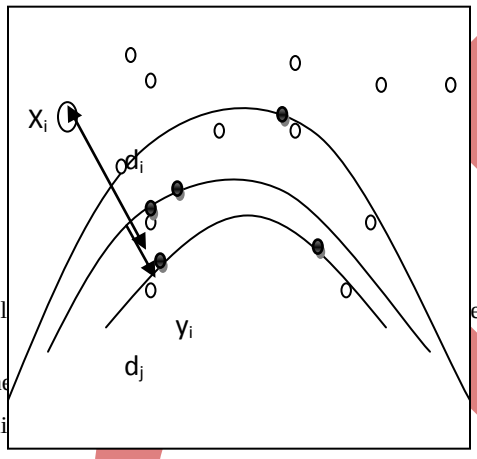


Fig: sel e ith hit.

yi is the

di =(xi

Algorithm:

Start

Initialize with M points, where M>N.(N is the number of cities)

Imagine a circular ring at the center of the cities.

The circle is elongated until all cities are covered.

Updated the ring points in parallel.

Computing the output.

Stop

4.Sumultaneous Recurrent Network

The traveling sales problem (TSP) was chosen as the benchmark for the performance evaluation of the SRN because it is representative of NP-hard optimization problems. In the TSP, a salesman spends his time visiting N cities (or nodes) cyclically. In one tour he visits each city just once, and concludes where he starts. The goal is to find the order he should visit the cities to minimize the total distance traveled. Selection of the TSP as the benchmark is appropriate because almost all non-learning neural search algorithms fail to deliver acceptable quality solutions with reasonable computational cost and time for large-scale variants of this problem.

SRN Topology for the TSP

N-city TSP is represented by an N×N array, where each row represents a different city and each column represents the possible positions of the cities in the path.

One set of N×N neurons in the input layer represents the distance between cities and an additional N×N neurons are used to receive the simultaneous feedback from N×N output layer neurons. Exploratory simulation studies indicated that only one hidden layer with four neurons is sufficient for satisfactory training of the network, which most likely represents the lower limit of neurons. The matrix of N×N neurons in the output layer represents the path of the travelling salesman. An active node at row r and column c in the output layer array represents that the path from city r to city c is included in the travelling salesman's itinerary represent an N-city TSP using the SRN.

In the tour matrix there is only one "1"in each row or column. And diagonally there are zeros in the matrix for the distance from one city to itself is zero. .In order to represent an N-city TSP using the SRN, the feed forward network F in the SRN consists of two layers. The output layer is an N×N matrix of nodes, with each row representing a city and each column representing a possible position in the path. Additionally, there is a single layer of hidden nodes. In the TSP, the inputs to the problem are the distances between the cities, represented by the cost matrix. As presented in the next section, the cost matrix is used in the calculation of the error function of the training algorithm. Therefore, in the case of the TSP the external inputs x in Figure are not applied to the network. The SRN for the TSP consists of a two-layer network with a relatively small number of hidden nodes and an N×N array of output nodes, with a recurrent connection between the nodes in the output layer and the hidden layer. Since no external inputs exist, the network is simply initialized with small random values for the weights and the outputs z and allowed to relax.After the outputs of the network converge to a fixed point, the outputs can be compared against problem-specific error function and the weights modified using a suitable learning algorithm. the architecture of the SRN for the TSP, where the feed forward network F consists of one hidden layer and one output layer. The SRN has trainable connections from each neuron of the hidden layer to each neuron of the output layer, which is represented with the forward weight matrix P. The simultaneous recurrent connections, which are also trainable, exist from each node in the output layer to each node in the hidden layer, and are represented by the backward weight matrix V. There are

no connections among the nodes in either the hidden layer or the output layer. The outputs z of the network are taken from each output layer neuron. As discussed above, the external inputs x are not necessary.

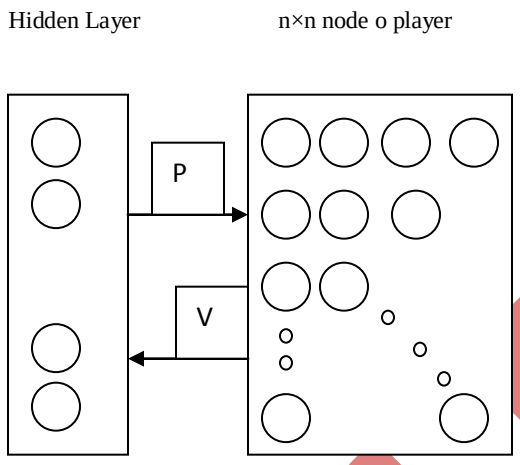


Fig: SRN architecture for TSP

Training SRN with RBP

In order to train the SRN, it is necessary to define a measure of the error for the output neurons. The error function needs to ensure valid solutions, as well as a minimum path length. Given the problem representation presented above, each row and column in the N×N output array must have exactly one neuron active: the output value of an active neuron should be close .

The traveling salesman should travel all cities at least and only once. Thus, when network converges to a solution, there should be exactly one node active per each row and per each column. This constraint can be implemented using inhibition among the nodes in a given row and column. The error term for the column constraint is defined by

Ecol= gcol∑Ni=1∑Nj=1[1- ∑Nm=1zmj(∞)2 ……(a)

where i and j are the indices for rows and columns, respectively, m is the index for rows of the network, (∞)mj z is the stable value of mj-th neuron output upon convergence to a fixed point, and gcol is a positive real weight parameter. When each column of the output matrix has exactly one active neuron, this error term will be zero. The first summation over the indexing variable i is included because the error function needs to be defined for each neuron in the output layer similarly for the row constrain

Erow=grow∑Ni=1∑Nj=1[1- ∑Nn=11znj(∞)]2 ……..(b)

where, i and j are the indices for rows and columns, respectively, of the network, n is the index for columns and g row is a positive real weight parameter. This error term will have a value of zero when each row of the output matrix has exactly one active neuron. Again, the second summation over the index variable j is included since the error function needs to be defined for every ij-th neuron in the output layer an error term is also introduced that force the neuron output to limiting the value as

Ebin= gbin∑Ni=1∑Nj=1[-(zij(∞)-α2+β] ……(c)

The error term associated with the distance between the cities can be formulated as

Edis=gdis∑Ni=1∑Nj=1∑Nm=1zij(∞)zm(j+1)(∞)dim ……(d)

Where dim is the cost associated with the path from city i to city m and gdis is the positive real weight parameter for the constraint. For each neuron Zij the index m search each neuron in the (j+1)st column, indicated by the Zm(j+1) term. If both neurons are active, the distance from city m,dim will be included in this error term, where the minimum value is achieved if the total distance of the path is minimum.

The total function is the sum of each individual error term defined by

E = Ecol+ Erow + Ebin + Edis...(e)

In order to converge on a valid and good (with minimum total distance) solution for the TSP, the state space portrait of the SRN must be changed by moving the fixed points towards preferably good solutions of the TSP.

The full derivation of the RBP algorithm can be found in [Pineda, 1987; Werbos, 1988]. The RBP training algorithm requires an adjoint network, which is topologically identical to the SRN except all signal directions reversed, to be set up and relaxed to compute updates for the weights of the SRN. The adjoint network accepts the error, which is computed using the stable values of neurons in the output layer of the SRN upon convergence to a fixed point, as external inputs to neurons in the input layer, which is the output layer for the SRN.

As we know that there is only one visiting city in a walk in a column, so in the error function the portion of the Ecol will be error free. Similarly in case of row, there is only one visiting city in a row, so the portion of the Erow will be error free. For this reason the third portion Ebin will be error free. So by calculating the fourth and the last portion Edis we can find the minimum distance of the traveling path.

Result & Discussion

In case of Hopfield network there are four constrain in the energy function. The first three constrain of them will be zero as per there is only one "1" in each row and column in a valid tour. So to get the minimum traveled path or the shortest path to cover all the city once at a time we have to use the fourth constrain as :(D/2 ∑nx=1∑ny=1,y≠x∑ni=1dxyvxi(vy,i+1+vy,i-1)), Where D/2 is a constant, vxi should be in the form of 0 or 1,

because it indicates the city is present in path or not. dxy is the distance from a city to another. Here by taking 4 city we can get 24 iteration using MATLAB 12, taking the values from the tour matrix i the minimum tour is: dDA + dAB + dBC + dCD=17+15+ 14+25=71.

In case of SRN here we have four error constrain. If there is only one "1" in a row or column in a valid tour then the first thee error constrain term will be error free its means the constrain term will be zero. So by just putting value in the last or fourth term: Edis=gdis∑Ni=1∑Nj=1∑Nm=1zij(∞)zm(j+1)(∞)dim We get the shortest or minimum path covering all the city once at a time. Where dim is the cost associated with the path from city i to city m and gdis is the positive real weight parameter for the constraint. For each neuron Zij the index m search each neuron in the (j+1)st column, indicated by the Zm(j+1) term. If both neurons are active, the distance from city m,dim will be included in this error term, where the minimum value is achieved if the total distance of the path is minimum. By using MATLAB , and taking 4 city in the tour we get 24 iteration to solve the TSP problem. Among these tour the minimum tour matrix is So the total minimum distance is EDA + DAB + DBC + DCD = 17 + 14+ 15 + 25 = 71.

In Elastic network firstly considering a neuron ring among all the cities. Here the ring work as a rubber band which increase to cover all the cities. So there is only one result in this method to solve the traveling salesman problem. So here we calculate the only the outer most path to calculate the minimum distance 71.

Comparing these three different process of solving the Traveling Salesman Problem (TSP) we can say that the Hopfield Network is the process by which the total distance to cover all the city in a tour exactly once, will be the minimum than Simultaneous Recurrent Network and The Elastic Network. in case of HN and SRN taking 4 city in tour we can get 24 different iteration by using MATLAB to reach the starting city by covering all the city exactly once in the tour. Using MATLAB we can also get the minimum tour among these 24 tours. In SRN we calculate distance from the error term. If the row, column, bin constrain is zero as for the there is only one 1 in column or row. In this two process there is energy function and error calculation respectively so there is difficult to get the result. There is no energy function in EN to calculate the minimum distance to TSP problem. There is only one way to get the distance, by adding the outer distance from each city to another. So this process is quite easy to solve TSP. But in HN we get the least distance.

Conclusion

In this comparative study on traveling salesman problem(TSP) we use Hopfield network, elastic net, simultaneous recurrent network to reach to optimal tour with less total distance. In developing the solution we can't get the exact optimal solution, it always be near the optimal solution. Few changes in energy function or update on them may create a better result. If any city is added or deleted from the tour we have to start from the starting city. As the number of city will increase then the number of iteration will also increase. In case of EN, it does not produce unfeasible solutions. There is no need of any energy function parameter to solve tsp problem. It produce sub optimal solutions that gives better results, it scales with problem size. In case of SRN is easy to compute and to get good quality result. Here neuron-optimizer does not require any weights as HN. Results are encouraging since the network was able to locate near optimal solutions. The SRN is able to identify solution with average intercity distance.

References:

Hopfield, J.J and Tank, D.W., neural computation of Decision in optimization problem, Big cybren.

Andrzej. K., & Zbigniew. N. modified Hopfield neural network for traveling salesman problem.

Kohonen T. the self organization map. Proc IEEE 1990

Peterson C. parallel distributed approaches to combinational optimization problem. Bio cyber 1985

E.H.L Aarts and J.Korts, " Boltzman machine for traveling problem" 1989

Hritsev_the ANN book

L.B.Almedia. " a learning Rupe for Asynchronous Perceptron with Feedback in a combinational Environment," proceeding of IEEE ICNN 1987

G. Laporte, A. Asef-Vaziri, and C. Sriskandarajah, "Some applications of the generalized travelling salesman problem." J. Oper Res Soc.47, 1461-1467(1996)

"Neural Network : A comprehensive foundation" by Simon S. Haykin,(1999), prentice hall

"Element of artificial neural network" ,by Kishan Mehrotra, Chilukuri K. Mohan and Sanjay Ranka,(1996), MIT press.