Farrukh Jabeen
COM 581 Mathematical Methods In AI

Assignment # 4

This problem asks you to analyze a K-WTA network (K winner take all). First read the K-WTA reference article, then answer the questions below. Read the Hopfield reference for a more general introduction to the topic. The power point presentation is also helpful as an introduction.

K-WTA neural networks are relatively simple. N neurons, with all the connection strengths equal to -1 (with no self-connections).

1.Define the connection strength matrix W.

Answer:

w_{ij} = connection strength between neuron i and neuron j.
Connection strengths are symmetric: $w_{ij} = w_{ji}$.
$w_{ij} = -1$ if i is not equal to j
$w_{ii} = 0$ (no self-connections)
W is an nxn symmetric matrix.

In java we can write like that

```
for ( i=0;  i < this.m_numberOfNeurons; i++)
            {
                  for (int j=0;  j < this.m_numberOfNeurons; j++)
                  {
                        // Setting initial Weights
                        if (i==j)
                        {
                              this.m_weights[i][j] = 0;
                        }
                        else
                        {
                              this.m_weights [i][j] = -1;
                        }

                  }
```

2. What are the eigenvalues and eigenvectors of W. (Work out the cases for N=2 and N=3, then try to generalize the result to arbitrary N).

Eigenvectors exist only for square matrices. An eigenvector of a square matrix has the defining property that multiplication by this matrix is equivalent to multiplication by a scalar(i.e. the length of the vector changes, its direction does not). If v is an eigenvector of W,

$Wv = \lambda v$

Where λ is a scalar called the eigenvalue associated with the eigenvector v.

For N=2

$$\mathbf{W=\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}}$$

Computing Eigenvalues and Eigenvectors

We can rewrite the condition $Wv=\lambda v$ as

$(W-\lambda I)v=0$.

where I is the $n \times n$ identity matrix

the determinant of $W-\lambda I$ must equal 0. We call $p(\lambda)=\det(W-\lambda I)$ the **characteristic polynomial** of W. The eigenvalues of W are simply the roots of the characteristic polynomial of W.

$p(\lambda)=$

$$\det \begin{bmatrix} 0-\lambda & -1 \\ -1 & 0-\lambda \end{bmatrix}$$

$= \lambda^2 - 1 = 0$

$= \lambda^2 = 1$

$= \lambda = \pm 1$

Thus, $\lambda 1 = 1$ and $\lambda 2 = -1$ are the eigenvalues of W

Now we find the eigenvectors corresponding to $\lambda 1 = 1$. Let $v=\begin{bmatrix} v1 \\ v2 \end{bmatrix}$ Then

(W- I)v=0 (here $\lambda=1$)

$$\begin{bmatrix} 0-1 & -1 \\ -1 & 0-1 \end{bmatrix} \begin{bmatrix} V1 \\ v2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

from which we obtain the equations

$-v1-v2 = 0$

$-v1- v2 = 0$

If we let v1=t, then v2=−t. All eigenvectors corresponding to $\lambda 1=1$ are multiples of $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

and thus the eigenspace corresponding to $\lambda 1=1$ is given by the span of $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$. That is, $\left\{\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right\}$ is a **basis** of the eigenspace corresponding to $\lambda 1=1$.

Similarly we can find the eigenvectors corresponding to $\lambda 2 = -1$

All eigenvectors corresponding to $\lambda 2=-1$ are multiples of $\begin{bmatrix}1\\1\end{bmatrix}$

and thus the eigenspace corresponding to $\lambda 2=-1$ is given by the span of $\begin{bmatrix}1\\1\end{bmatrix}$. That is, $\left\{\begin{bmatrix}1\\1\end{bmatrix}\right\}$ is a **basis** of the eigenspace corresponding to $\lambda 2=-1$.

For N=3

$$W=\begin{bmatrix}0 & -1 & -1\\ -1 & 0 & -1\\ -1 & -1 & 0\end{bmatrix}$$

Similarly we can find $p(\lambda)=\det(W-\lambda I)$ the **characteristic polynomial** of W. The eigenvalues of W are simply the roots of the characteristic polynomial of W

$P(\lambda)$

$$=\det\begin{bmatrix}0-\lambda & -1 & -1\\ -1 & 0-\lambda & -1\\ -1 & -1 & 0-\lambda\end{bmatrix}$$

$$=\begin{vmatrix}-\lambda & -1 & -1\\ -1 & -\lambda & -1\\ -1 & -1 & -\lambda\end{vmatrix}$$

Expand the determinant and we will get the values of $\lambda = -2,1,1$

$\lambda 1=-2$, $\lambda 2= 1$ and $\lambda 3= 1$ are the eigenvalues of W .

Now we find the eigenvectors corresponding to $\lambda 1=-2$. Let $\mathbf{v}=\begin{bmatrix}v1\\v2\\v3\end{bmatrix}$ then

$(W- (-2)I)v=0$
$= (W +2I)v=0$

$W + 2I =$

$$\begin{bmatrix}0 & -1 & -1\\ -1 & 0 & -1\\ -1 & -1 & 0\end{bmatrix}+\begin{bmatrix}2 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 2\end{bmatrix}$$

Now $(W+ 2I)\, v=0$

$$\begin{bmatrix}2 & -1 & -1\\ -1 & 2 & -1\\ -1 & -1 & 2\end{bmatrix}\begin{bmatrix}V1\\V2\\v3\end{bmatrix}$$

$$=\begin{bmatrix}0\\0\\0\end{bmatrix}$$

$2v1 - v2 - v3=0$
$-v1 + 2v2 - v3=0$
$-v1 - v2 + 2v3=0$

If we let v3=t, then v1=t and v2=t . All eigenvectors corresponding to $\lambda 1=-2$ are multiples of $\begin{bmatrix}1\\1\\1\end{bmatrix}$

and thus the eigenspace corresponding to $\lambda 1=-2$ is given by the span of $\begin{bmatrix}1\\1\\1\end{bmatrix}$ That is, $\left\{\begin{bmatrix}1\\1\\1\end{bmatrix}\right\}$ is a **basis** of the eigenspace corresponding to $\lambda 1=-2$.

Similarly we can find the eigenvectors corresponding to $\lambda 2= 1$

All eigenvectors corresponding to $\lambda 2=1$ are multiples of $\begin{bmatrix}-1\\1\\0\end{bmatrix}$

and thus the eigenspace corresponding to $\lambda 2=1$ is given by the span of $\begin{bmatrix}-1\\1\\0\end{bmatrix}$. That is, $\left\{\begin{bmatrix}-1\\1\\0\end{bmatrix}\right\}$ is a **basis** of the eigenspace corresponding to $\lambda 2=1$.

And the eigenvectors corresponding to $\lambda 3=1$ will be same as above

- Eigenvalues and eigenvectors can be complex-valued as well as real-valued.
- The dimension of the eigenspace corresponding to an eigenvalue is less than or equal to the multiplicity of that eigenvalue.
- The techniques used here are practical for 2×2 and 3×3 matrices. Eigenvalues and eigenvectors of larger matrices are often found using other techniques, such as iterative methods.

Let W be an $n \times n$ matrix. The eigenvalues of W are the roots of the characteristic polynomial $p(\lambda)=\det(W-\lambda I)$.

For each eigenvalue λ, we find eigenvectors $v=\begin{bmatrix}v1\\v2\\..\\.\\vn\end{bmatrix}$ by solving the linear system

$$(W-\lambda I)v=0.$$

The set of all vectors v satisfying $Wv=\lambda v$ is called the **eigenspace** of W corresponding to λ.

3. Show that all K-WTA hyperplanes are parallel.

Consider a symmetric network which bounded continuous activation values
$(m \leq a_i(t) \leq M, i = 1, \cdots, N).$
We will call K winner network if it satisfies the following conditions.
It always converges to a final state consisting of K units with maximum activation value and N-k units with minimum activation value.
The units at the maximum values are those units that had larger initial states.

The permissible exceptions are when there are ties between the winners.
The initial state of a unit is defined to be the value of
$a_i(0) + e_i$
Where e_i is the external input. Here we assume that external inputs are constant with respect to time but can be different for each unit.
k-winner corners of the N dimensional hypercube formed by the M and m boundaries.
Now let
$\{b_j: j = 1, \cdots, K_N\}$
The set of K winner corners for a given K and N. If $b_j = [b_{j,1}, \cdots, b_{j,N}]$
Then exactly K of $b_{j,i}$'s are equal to M and the rest are equal to m. So for a given choice of K all the b_j's lie on a common N-1 dimensional hyperplane perpendicular to the vector [1……,]. Thus , all k-winner hyperplanes are parallel.

4. Show that the k-winner states are stable equilibriums when the external input satisfies (assume m=0 and M=1):

k-1 < ext < k.

Hint: K-winner states are the corners of the hypercube corresponding to k neurons at maximum activation and the rest at minimum activation. To show that these states are stable equilibriums you must imagine the activation vector as being very close to, but not on, the hypercube corner. Then show that the dynamics of the network will "drive" the activation vector into the corner, as opposed to away from the corner. That is, if activation is near the maximum, then the update will add a positive amount, and if an activation is near the minimum the update will add a negative amount.

Here we consider

maximum neuron activation = 1
minimum neuron activation = 0

K-winner states are the corners of the hypercube corresponding to k neurons at maximum activation and the rest at minimum activation. To show that these states are stable equilibriums we must imagine the activation vector as being very close to, but not on, the hypercube corner. Then we show that the dynamics of the network will "drive" the activation vector into the corner, as opposed to away from the corner. That is, if activation is near the maximum, then the update will add a positive amount, and if an activation is near the minimum the update will add a negative amount k-winner states are at a stable equilibrium when the neuron activations are no longer changing from one iteration to the next. Neuron activations get updated through the following:

(1) ai(t+1) = ai(t) + step * (M - ai(t)) * (ai(t) - m) * neti(t)
where 'M' is the maximum activation, 'm' is the minimum activation, step is a constant that determines how fast the system converges to a stable equilibrium, and 'net' is the net input to a neuron.
The system enters into stable equilibrium when ai(t+1) = ai(t) for all neurons. In our system, this occurs when all neurons have an activation of either 'M' or 'm'. When this occurs, (M - ai(t)) * (ai(t) - m) = 0 and then ai(t+1) = ai(t) for all neurons.
As the activation vector approaches a hypercube corner, (M - ai(t)) * (ai(t) - m) approaches a minimum (0 in our case), causing the neurons to get closer to either 'm' or 'M' by increasingly smaller increments.
However, in order for the neuron's activations to get closer to the hypercube corner, the neti(t) factor must be positive for the neurons approaching 'M' (winners) and negative for the neurons approaching 'm ' (losers).
(2a) for k winners: neti(t) > 0
(2b) for N-k losers: neti(t) < 0
neti(t) is calculated through:
(3) neti(t) = [**Wa(t)** + **e(t)**]i
where **W** is an NxN matrix of connection strengths (wij) populated with 0 where i = j and -1 everywhere else and **e** is our external input. In the hypercube corner where k neurons have 'M' activation (1 in our case) and N-k neurons have 'm' activation (0 in our case), the following is true:
(4a) for k winners: neti(t) = - (k - 1) + ei
(4b) for N-k losers: neti(t) = - k + ei
However, as the system approaches the hypercube corner, the winners aren't quite all at 'M' and the losers aren't quite all at 'm' so,
(5a) for k winners: neti(t) < - (k - 1) + ei
(5b) for N-k losers: neti(t) > - k + ei
With our assumptions regarding the sign of neti(t) from (2), this becomes:
(6a) for k winners: 0 < - (k - 1) + e
(6b) for N-k losers: 0 > - k + e
Solving for e yields:
(7) k - 1 < e < k

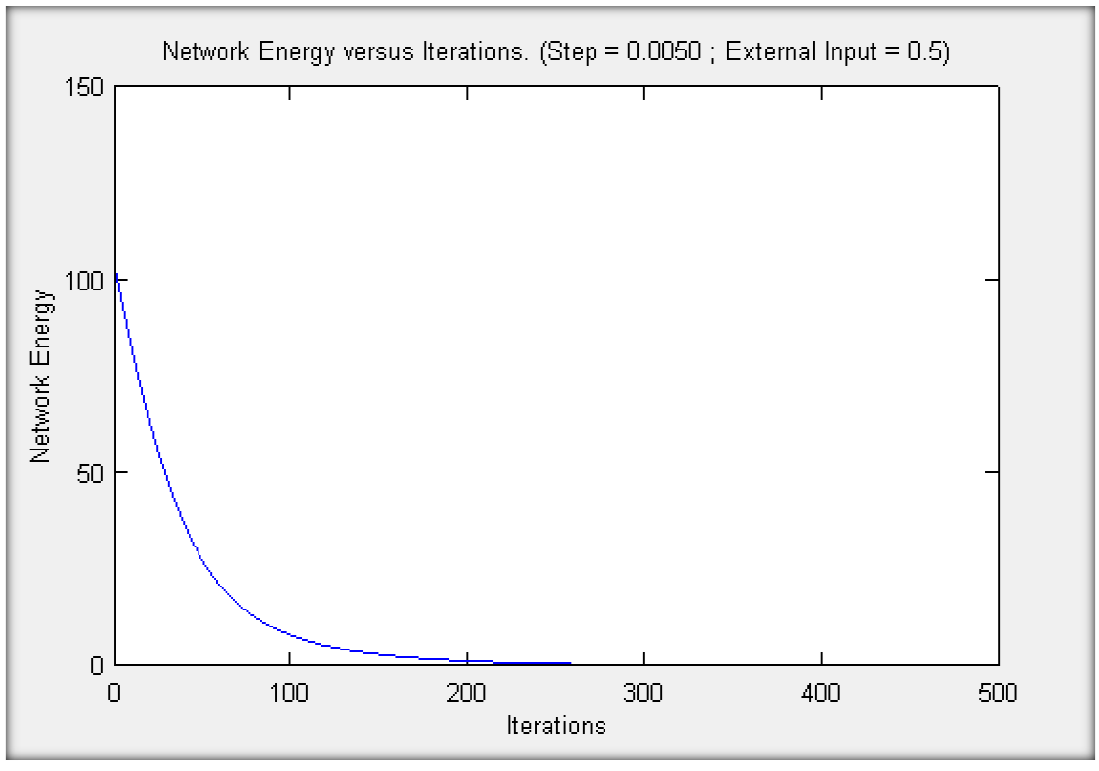
5. Show that the dynamics defined by da/dt = net is a gradient descent on the energy function.

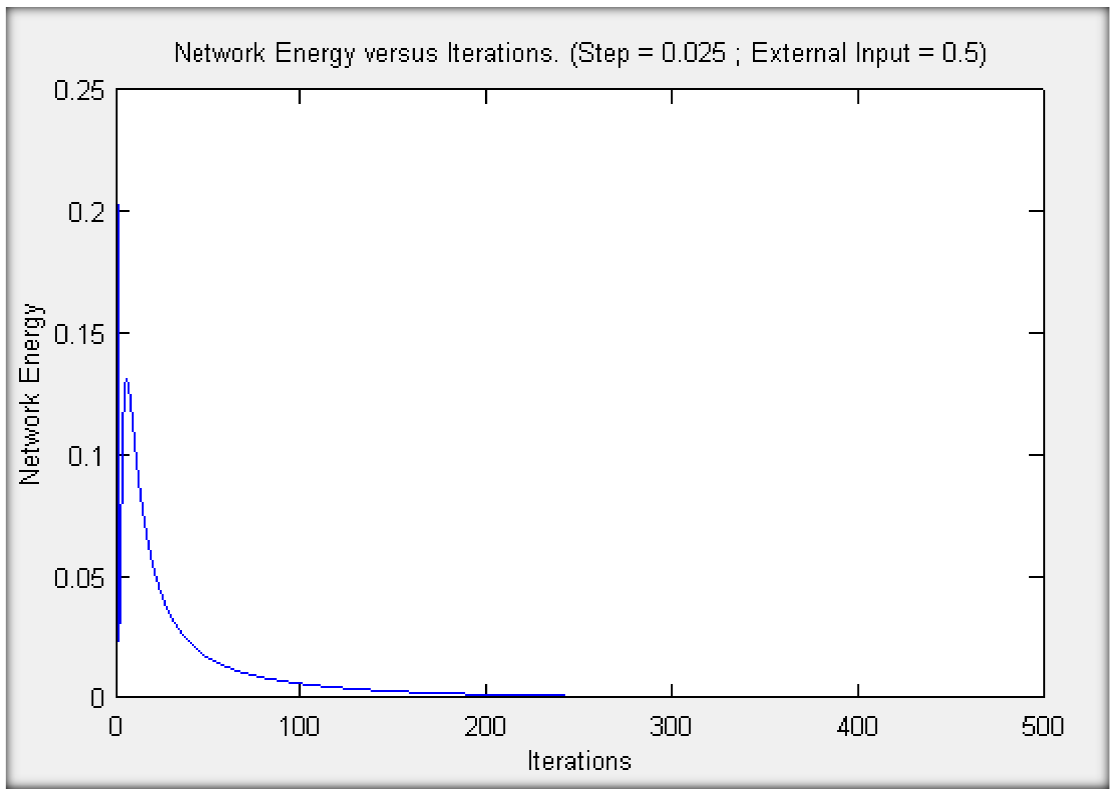
Gradient descent is a first-order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or the approximate gradient) of the function at the current point.

Suppose in the k winner network if we have

number of neurons = 20
maximum neuron activation = 1
minimum neuron activation = 0
number of iterations =500

The following graphs show that energy decreases for each iteration.
Here “Octave” is used to plot the results





Gradient descent with respect to an energy function, implemented by $\mathbf{Net}_i(t)$ calculations. The energy of the network is defined to be

$$\mathbf{Energy} = -1/2 * \sum_i \sum_j a_i * a_j * w_{ij} - \sum_i a_i * e_i.$$

A corner seeking part implemented $[M - a_i(t)]$ and $[a_i(t) - m]$.

So interactive activation can be viewed geometrically as a two step process

Step 1: For each unit calculate the negative gradient via $\mathbf{Net}_i(t)$. The vector $\mathbf{Net}(t)$ points into particular orthant of N- dimensional space. That orthant is consistent with a particular corner of the N-dimensional hypercube, call it corner(t). From the vector difference between the current activations a(t) and this corner vector: corner(t) – a(t).

Step2:

The multiplying $\beta * \mathrm{Net}_i(t) * |\mathrm{corner}_i(t) - a_i(t)|$ gives the components of the vector that the dynamics will follow $(\Delta a(t))$. That is $a_i(t+1) = a_i(t) + \Delta a_i(t)$, where
$\Delta a_i(t) = \beta * \mathrm{Net}_i(t) * |\mathrm{corner}_i(t) - a_i(t)|$.

From this geometric point of view we can say that that the dynamics follow a vector that points into an orthant that is on the positive side of the gradient descent; thus , for sufficiently small step size , the energy of network decreases at each iteration.

References

Dr.Wolfe class notes

http://en.wikipedia.org/wiki/Gradient_descent
http://books.google.com/books?id=ZEID3w9STOUC&pg=RA2-PA41&lpg=RA2-PA41&dq=eigenvalues+neural+network&source=bl&ots=svMyC_UB_N&sig=OCRpobxfriavqpL1B5hG_5FdhqU&hl=en&ei=R1LZSv3DKoP8sQPSrbGSBg&sa=X&oi=book_result&ct=result&resnum=5&ved=0CB4Q6AEwBDgU#v=onepage&q=eigenvalues%20neural%20network&f=false
http://www.math.hmc.edu/calculus/tutorials/eigenstuff/
http://www.freewebs.com/farrukhjabeen/cs572assignment1.htm