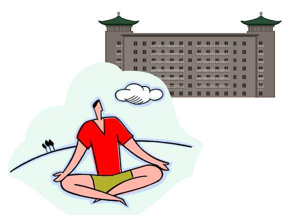
Associative memory and Hopfield Neural Network

Content of this lecture

- What is an associative memory?
- What is a Hopfield neural network (HNN)?
- Energy function and property of HNN.
- Method for storing and re-calling patterns using HNN.
- A brief introduction to the Boltzmann machine (BM).

What is associative memory ?

- Conventional memory in computers
  – Address based memory
- Associative memory
  – Content based memory
  – Keyword based memory
  – Pattern based memory

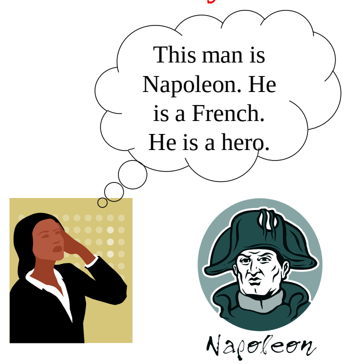


Auto-associative memory

- key → key
- De-noising
- Reconstruction of patterns based on similarity.

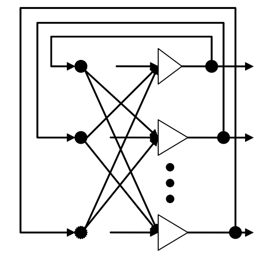
Hetero-associative memory

- key → content
- Recognition of patterns based on hints or features.



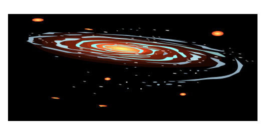
The Hopfield neural network

- Hopfield neural network (HNN) is a model of auto-associative memory.
- The structure is shown in the right figure.
- It is a single layer neural network with feedbacks.



Some terminologies

- The output of each neuron is a binary number in {-1,1}.
- The output vector is the ***state vector.***
- Starting from an initial state (given as the input vector), the state of the network changes from one to another like an automaton.
- If the state converges, the point to which it converges is called the ***attractor.***



The state-transition mechanism

Suppose that the current state of the network is

$$\mathbf{v}^k = [v_1^k, v_2^k, \ldots, v_n^k]$$

then, the next state can be calculted by

$$v_i^{k+1} = \mathrm{sgn}(net_i) = \mathrm{sgn}(\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} v_j^k + \theta_i)$$

where θ_i is the threshold of the ith neuron

- Note that the update is asynchronous. That is, one neuron is updated each time, and the update order is random.

- Note also that $w_{ij} = w_{ji}$ and $w_{ii} = 0$ for all i.

The energy function

- We can define an energy function for each HNN.
- For the network shown before, the energy function E is defined by

$$\boldsymbol{E} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_{ij} v_i v_j + \sum_{i=1}^{n}\theta_i v_i = -\frac{1}{2}\boldsymbol{v}^T \boldsymbol{W}\boldsymbol{v} + \Theta^T \boldsymbol{v}$$

Feature of the energy function

- The energy function E is a function of the state vector $\boldsymbol{v}$.
- For an HNN, its energy function never increases during state transition.
- That is, the value of E at the initial state is always larger than that at the attractor.

Increment of *E*

By updating the state, we get an increment

$$\Delta \boldsymbol{v} = \boldsymbol{v}^{k+1} - \boldsymbol{v}^{k}$$

Using Taylor extension, we can find the increment of $\boldsymbol{E}$ by

$$\Delta \boldsymbol{E} = \boldsymbol{E}^{k+1} - \boldsymbol{E}^{k} = (\nabla \boldsymbol{E})^T \Delta \boldsymbol{v} + \varepsilon$$

where ε is the 1st order approximation error, and can be omitted.

Increment of E (cont.)

The gradient of $\boldsymbol{E}$ can be calculated by

$$\nabla \boldsymbol{E} = -\frac{1}{2}(\boldsymbol{W}^T + \boldsymbol{W})\boldsymbol{v} + \Theta$$

The weight matrix $\boldsymbol{W}$ is supposed to be symmetric, and thus we have

$$\nabla \boldsymbol{E} = -\boldsymbol{W}\boldsymbol{v} + \Theta$$

Thus, the increment of $\boldsymbol{E}$ becomes

$$\Delta \boldsymbol{E} = -(\boldsymbol{W}\boldsymbol{v} - \Theta)^T \cdot \Delta \boldsymbol{v}$$

Increment of E (cont.)

Since the state transition is asynchronous, only the i-th (i is random) output is changed each time. The state increment is actually given by

$$\Delta \boldsymbol{v} = (0,\ldots,\Delta v_i,\ldots,0)^T$$

Therefore,

$$\Delta E = -(\sum_{j=1}^{n} w_{ij} v_j - \theta_i)\Delta v_i = -net_i \cdot \Delta v_i$$

Increment of E (cont.)

We have the following relations :

$$\text{when } net_i < 0, \Delta v_i \le 0$$
$$\text{when } net_i > 0, \Delta v_i \ge 0$$

Thus, $\Delta E = -net_i \Delta v_i$ is always negative, and thus E will never be increased by state transition.

Why Hopfield neural network is an associative memory ?

- Starting from any initial state, the HNN will change its state until the energy function approaches to the minimum.
- The minimum point is called the ***attractor.***
- Patterns can be stored in the network in the form of attractors.
- The initial state is given as the input, and the state after convergence is the output.

How to store the patterns ?

Suppose that we have p patterns to be stored. We can calculate the weight matrix as follows :

$$W = \sum_{m=1}^{p} s^m (s^m)^T - pI$$

where s^m is the m-th pattern (a column vector), and I is the unit matrix. The thresholds of all neurons are set to zeros.

How to store the patterns (cont.)?

If the patterns take value from {-1,1}, the weights are given by

$$w_{ij} = (1-\delta_{ij})\sum_{m=1}^{p} s_i^m s_j^m$$

where δ_{ij} is the Kronecker function defined by

$$\delta_{ij} = \begin{cases} 1 & i=j \\ 0 & \text{otherwise} \end{cases}$$

If the patterns take value from {0,1}, we have

$$w_{ij} = (1-\delta_{ij})\sum_{m=1}^{p} (2s_i^m - 1)(2s_j^m - 1)$$

How to use a HNN ?

- Phase 1: Store all patterns into the network by finding the weight matrix as above.
- Phase 2: Recall a pattern when an input is given as the initial state.

Phase 1: Storage

- Step 1 Initialization: ***W=0***
- Step 2 Store the *m-th* pattern $s^{(m)}$ by
  $$\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{s}^{(m)}(\boldsymbol{s}^{(m)})^t$$
- Step 2 is repeated for all patterns.
- After all patterns are stored, set $w_{ii}=0$ for $i=0,1,\ldots,n$.

Program for Storage

There is NO feedback from one neuron to itself

```
Store_Pattern(){
 int i,j,k;

 for(i=0; i<n_neuron; i++){
   for(j=0; j<n_neuron; j++){
     w[i][j]=0;
     if(i==j) continue;

     for(k=0;k<n_pattern;k++)
         w[i][j] += s[k][i]*s[k][j];
     w[i][j]=w[i][j]/n_pattern;
}}}
```

Store the patterns into the network using previous equation

Normalize the weights

Phase 2: Recall

- Step 1: The keyword is given as the initial state.
- Step 2: Select one of the neurons at random, find the output using the current outputs.
- Step 3: If for many neurons, the new outputs are the same as the old ones, the network converges, and the current output is the pattern to be recalled.
- Step 4: If the new output is different from the old one, return to Step 2.

Program for recall

Select a neuron to update at random or one by one

Find the new output of the i0-th neuron

Stop if there is no change after many iterations

```
Recall_Pattern(int v[ ]){
  int   i,i0,j,k, n_update, net, vnew;

  for(k=1,n_update=0; ;k++){
    for(i=0; i<n_neuron; i++){
      i0=rand()%n_neuron;

      for(j=0,net=0; ; j<n_neuron; j++) net+=w[i0][j]*v[j];
      vnew=(net >= 0)? 1:-1;

      if(vnew != v[i0]) { n_update++; v[i0]=vnew; }
   }
   if(n_update==0) break;
} }
```


Example of associate memory

```
********
********
**    **
**    **
**    **
**    **
**    **
**    **
********
********
```

```

********
********
      **
      **
********
********
**
********
********
```

```
   **
  ***
 ****
** **
**  **
*********
*********
    **
    **
    **
```

```

*********
*********
**
*********
*********
**      **
**      **
*********
*********
```

- We have four patterns given on the left.
- The purpose is to save them into a Hopfield neural network, and then recall them when they are corrupted by noises.

An example of results

```
*******
**********
***  * **
**   **
* *   ***
**   **
**   **
**    *
********
********
```

```
********
**********
**    **
*     **
***   ***
**     **
**     **
**     **
********
********
```

```
********
********
**     **
*      **
***   ***
**     **
**     **
**     **
********
********
```

```
********
********
**     **
**     **
**     **
**     **
**     **
**     **
********
********
```

How to see the usefulness of the Hopfield neural network?

- As an associative memory, an HNN can only store *0.14n* patterns, where *n* is the number of neurons.
- The HNN, however, can be useful for signal or image restoration.
- It may also be useful for solving combinatorial optimization problems.

Boltzmann machine

- Boltzmann machine (BM) is another type of neural network, proposed by G. Hinton and T. Sejnowski (1985).
- The structure is the same as HNN with the following restrictions:
  - There is no feedback from a neuron to itself.
  - The connection matrix is symmetric.
- The energy function can also be defined in the same way as in HNN.

BM is a stochastic network

- In a BM, a neuron fires according with a probability given by

$$p(v_k = 1 \mid \boldsymbol{net}_k) = \frac{1}{1 + \exp(-\frac{\boldsymbol{net}_k}{\tau})}$$

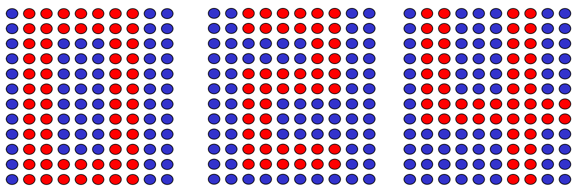
  where τ is called the temperature.
- That is, the neuron may not fire even if the effective input is larger than the threshold.
- The property might be useful to get the global optimal solution when we use the network to solve combinatorial problems.

Property of the BM

- If we run BM by repeatedly choosing a neuron and finding its next state with the above mentioned formula, the probability of the state will become stationary, and will follow the Boltzmann distribution.
- To guarantee the convergence of the state, we should set a relatively high temperature, and reduce it gradually.
- This process is called ***simulated annealing.***

Team Project III

- Suppose that a Hopfield neural network contains 12x10 neurons. We want to store some patterns given as follows (not necessarily the same) into the network, and recall any of them with a pattern corrupted by noise.



Team Project III (cont.)

- Write a computer program to implement the algorithm.
- Try to recall the patterns with the noise level being 0%, 10% or 15%.
- We say a pixel is a noise if its value is changed from 1 (or 0) to 0 (or 1).