HW4

Xiang Pan 潘翔
1900941520
hover@hust.edu.cn

Part 1: Why the update rule is doing gradient descent?

1 Proof

$$\Delta E = 2(-v_r^{new} + v_r^{old}) \sum_{i \neq r} v_i w_{i,r}$$
$$= 2(-sign(\sum_{i \neq r} v_i w_{i,r}) + (-1)sign(\sum_{i \neq r} v_i w_{i,r})) \sum_{i \neq r} v_i w_{i,r}$$

$$\text{if sign}\left(\sum_{i!=r} v_i w_{i,r}\right) > 0: \Delta E<0$$
$$\text{if sign}\left(\sum_{i!=r} v_i w_{i,r}\right) \leq 0: \Delta E=0$$
$$\Delta E<=0$$

2 Pattern

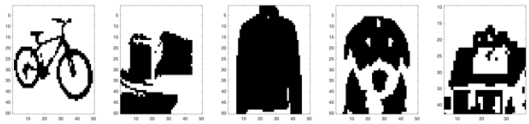

Image 1.1 Pattern from original Image

Image 1.2 Corrupt and the Image Pattern after Hopnet

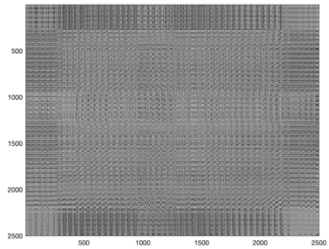

Image 1.3 Connection Map of Neuron

Does the network always settle down to the global minimum?

There are three situations

X(m) is the state of Neuron, because Hopfield neural network is a finite neural network

1. Stable(A)

$$X(m+1) = X(m)$$

2. Loop(B)

$$X(m) = X(i)$$

When the iteration end, the final state will be one of the loop states, but it is usually not the global minimum

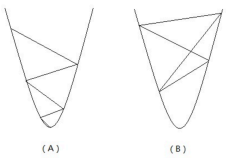

Image 1.4 The Energy Space of Hopfield NN

Part 2: How many patterns can be stored and retrieved?

Image 2.1 Pattern Learning By Hebbian rule, Corrupted

Image 2.2 Pattern Learning By oja rule, Corrupted

With enough iteration numbers, the result of oja rule and Hebbian rule is almost no difference.

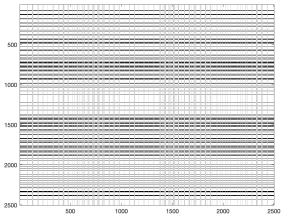

Image 2.4 Connection Map of Neuron(pattern=1)

We can notice that the connection is more sparse than more pattern numbers. Hence, we can use the connection map to measure the capacity of the NN which means that if the connection map is dense, then we add more patterns, the network may miss some information of other pattern.

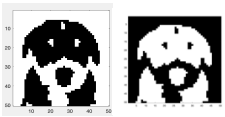

Image 2.5 Connection Map of Neuron(Noise=10000)

Noise is add randomly, so it may cover the blank area of the picture, the network may memory the pattern outline, but with the reverse of point distribution.

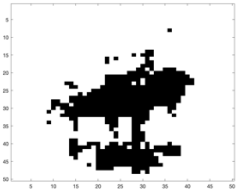

Image 2.5 Pattern not in Network Memory

Part 3: What is the algorithm implemented and why it helps?

Code 3.1 Learning Alternately

```
for epoch=1:100
    for iter=1:numPatterns
        xcur = mypatterns(:,:,1:iter);
        xcur = reshape(xcur,2500,iter);
        ycur = xcur'*T';
        deltaT = lr*(xcur - T*ycur')*ycur;
        T=T+deltaT;
    end
end
```

The method implemented is just like batch train. Comparing to use one sample at one training step, the batch method provides more stable and can make the NN memory more share pattern and will not overfit 1 pattern.