Homework06 / Restricted Boltzmann Machine with Contrastive Divergence

李豪韋 (HW-Lee) ID 103061527

Overview

Restricted Boltzmann Machine is a system that is able to learn data without knowing their corresponding labels, so it is used for pre-training, which is a kind of unsupervised learning to make the features easier to be found by classifiers. In this project, I tried to teach a RBM system with hand-written digits and verify its feasibility in pre-training. In Boltzmann Machine, the objective is to maximize the sum of log-likelihood of training instances, to reconstruct a testing instance so that the reconstructed instance is more 'similar' to training instances than before. Optimistically, the reconstructed input will be more robust to be classified and the variance within a class will be decreased. Finally, my system seems to successfully learn some features after feeding lots of instances, but the final weights look very different from those shown in Hinton's slides.

The more detailed information of the code is publicly available on https : //github.com/HW-Lee/2015-NN-Homeworks/tree/master/HW06.

Implementation

1. **System Structure**

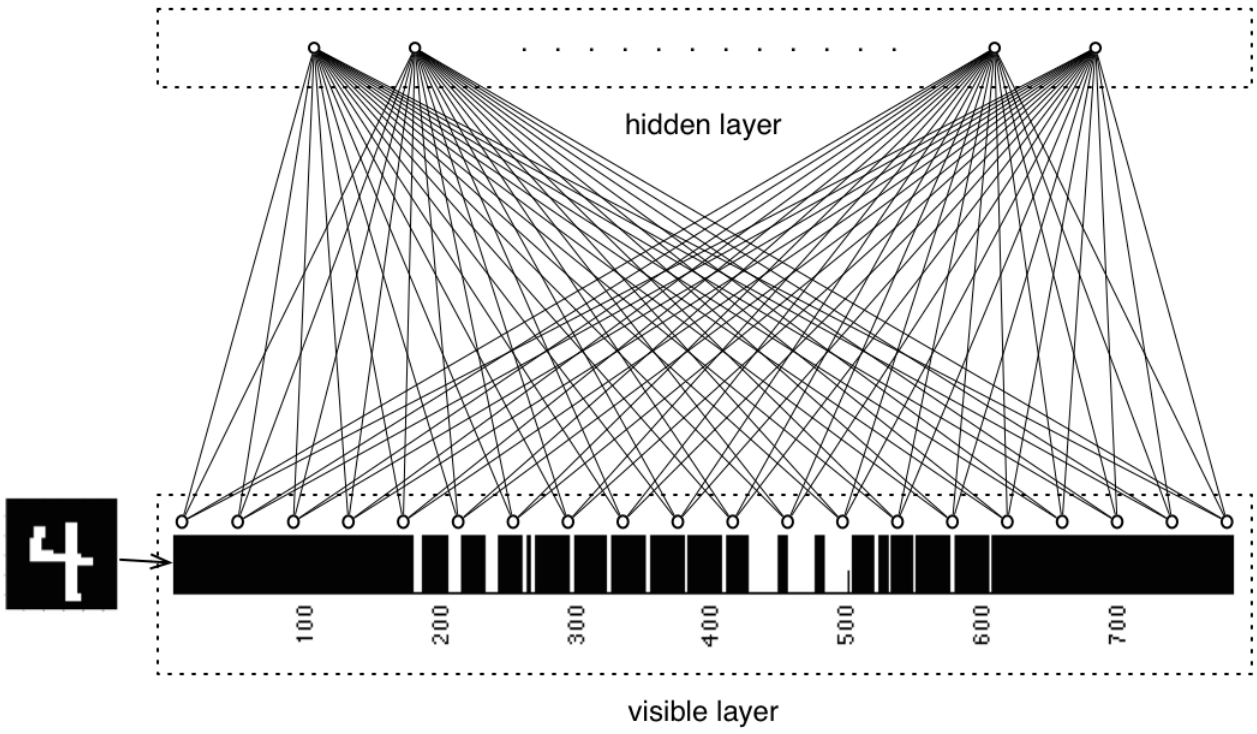
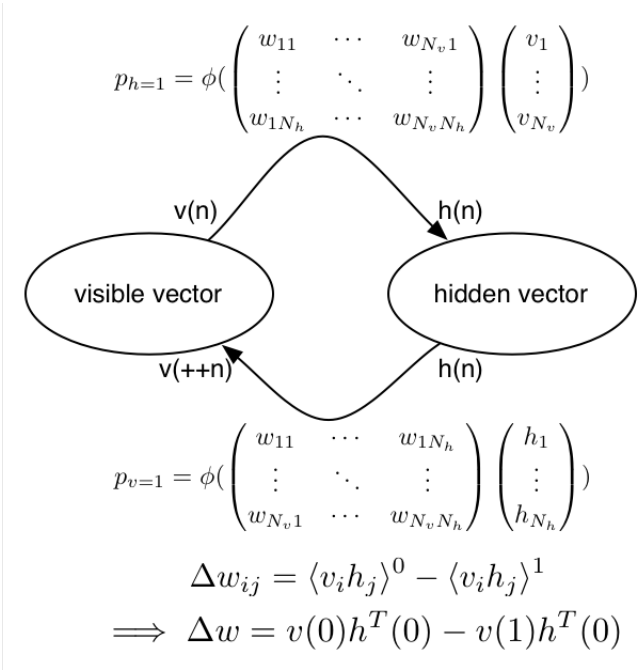


Figure 1: The net consists of 28×28 visible neurons and 450 hidden neurons

2. **Algorithm**



Contrastive Divergence:

1. $v(0) := \text{instance}$
2. $p_h := \phi\big(W^T v(0)\big)$
3. $h(0) := \text{Bernoulli}(p_h)$
4. $p_v := \phi\big(W h(0)\big)$
5. $v(1) := \text{Bernoulli}(p_v)$
7. $W := W + \eta \Delta W$

Reconstruction Iteration:

1. $v(0) := \text{instance}$
2. $n := 0$
3. $p_h := \phi\big(W^T v(n)\big)$
4. $h(n) := \text{Bernoulli}(p_h)$
5. $p_v := \phi\big(W h(n)\big)$
6. $v(n+1) := \text{Bernoulli}(p_v)$
7. $n := n + 1$
8. goto 3. until convergence

Figure 2: Training and reconstruction algorithm

Results

Successfully Reconstructed Result: with digit-7

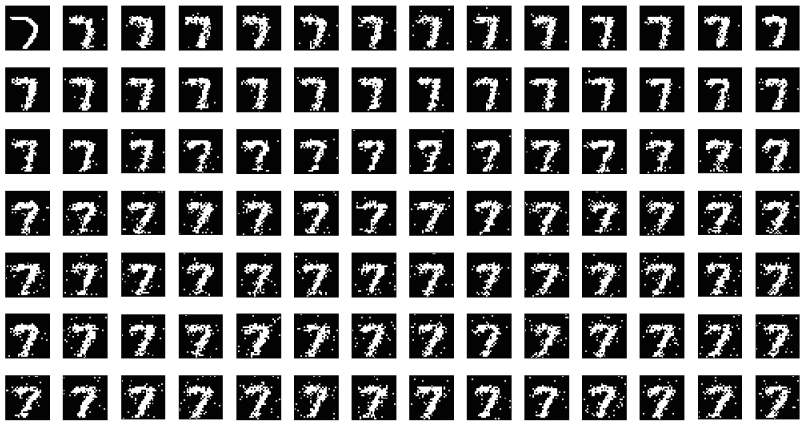


Figure 3: The visible neurons at each iteration, (left-to-right, top-to-bottom)

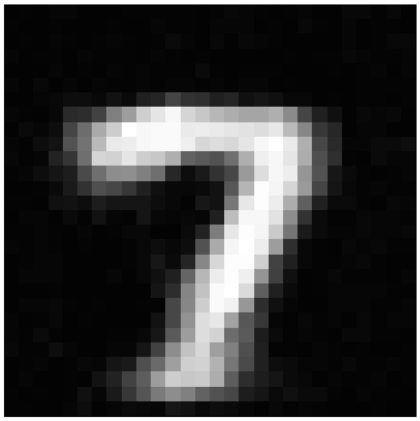


Figure 4: Averaged visible neurons

The results are publicly available on
https : //github.com/HW-Lee/2015-NN-Homeworks/tree/master/HW06/res.

Unsuccessfully Reconstructed Result: with digit-6

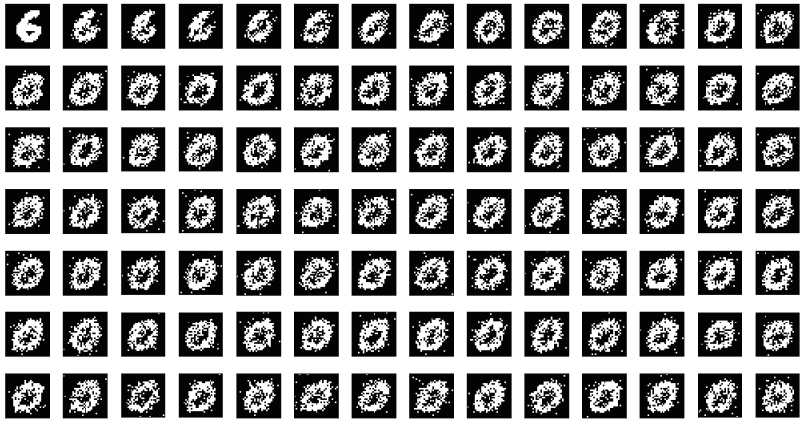


Figure 5: The visible neurons at each iteration, (left-to-right, top-to-bottom)

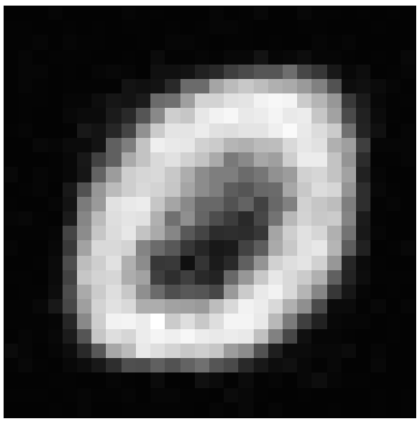


Figure 6: Averaged visible neurons

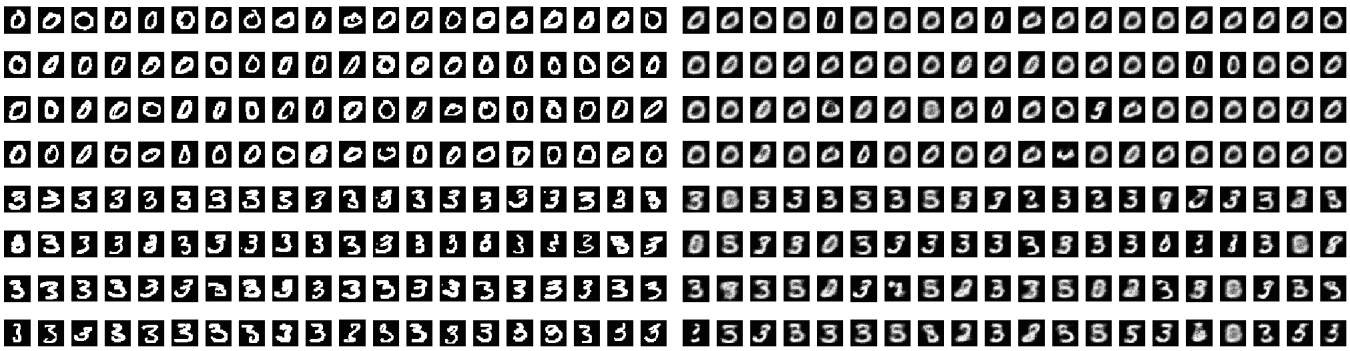


Figure 7: Better reconstructed results (left: original, right: reconstructed)

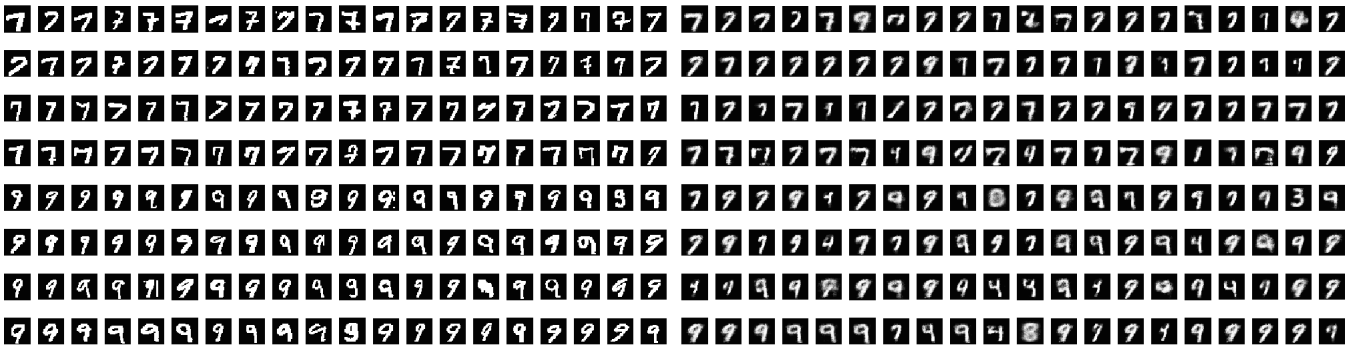


Figure 8: Worse reconstructed results (left: original, right: reconstructed)

Discussion

1. How to initialize the weights?

   The weights can not be arbitrarily randomized because every structure has different number of visible/hidden neurons. For adaptivity of any determined structure, I use the strategy that the random variance depends on the number of visible/hidden neurons. In this project, I simply set the variance to N_v^{-1}

2. How does the learning rate η affect the results?

   In my experience, different value of learning rate will not affect relative relation between weights in a hidden neuron too much, i.e. a large learning rate can lift weights in a hidden neuron while the ratio between them roughly remains the same, such that they look similar when they are visualized with imagesc function in MATLAB. As a result, the most critical difference is that the system will get stable more quickly because all weights are at the higher level, such that the probability will be closer to either 1 or 0.

3. What have been really learned by the system?

   According to the final results shown below, the system seems to learn the shape of digits rather than local features, which is mentioned in the Hinton's lecture.

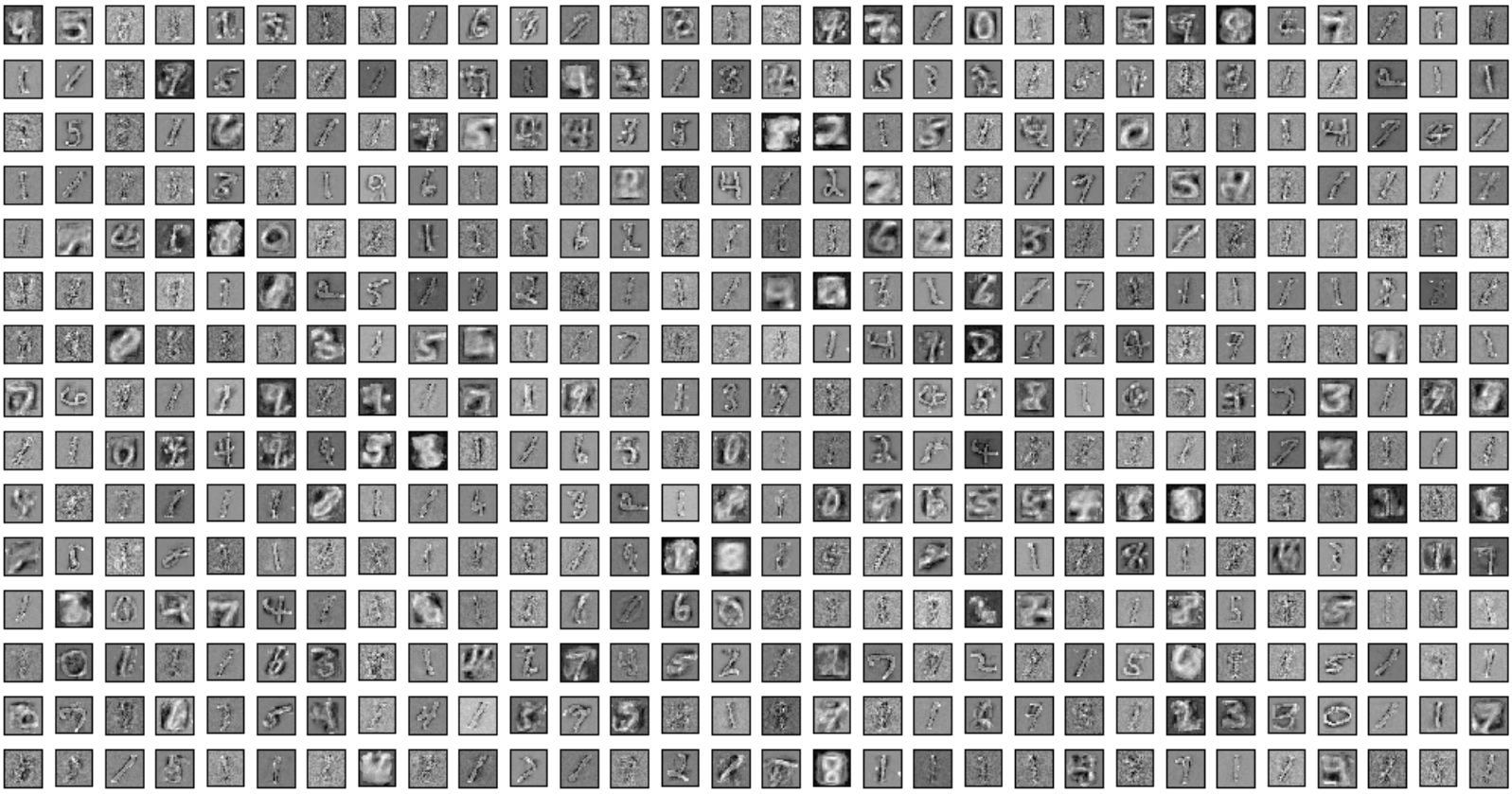


Figure 9: All final weights