Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4
Restricted Boltzmann Machines and Deep Belief Nets

Chiachen Ho, Angelos Stais and Ruxue Zeng

October 12, 2021

1 Main objectives and scope of the assignment

Our major goal in the assignment is to study the Restricted Boltzmann machines and deep belief nets. Our objectives are:

- Understand the learning process of RBMs
- Apply basic algorithms for unsupervised greedy pretraining of RBM layers and supervised greedy pretraining of DBN
- Design multi-layer neural network architectures based on RBM layers for classification problems
- Study the functionality of DBNs including generative aspects

2 Methods

We have used Python in this lab, using the semi-completed code of the professor, with libraries below:

- Numpy, Math, Random, Matplotlib, Spicy

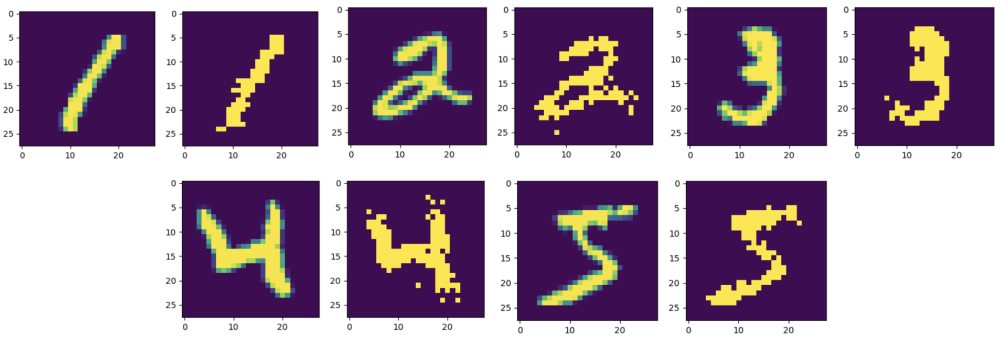

Figure 1 - Comparison between initial patterns and reconstruction after 30 iterations with 500 hidden nodes

3.1 RBM for recognizing MNIST images

- **Contrastive divergence learning**

We initialized the weight matrix with small random values (normally distributed: N(0,0.01)) and iterated the training process (CD) for 30-50 epochs. The size of mini-batches is 20. To monitor convergence or stability, we can stop training when MSE between the original and reconstructed images or W is not larger than a threshold. The Average Reconstruction Loss converges at around 20-30 epochs.

- **Reconstruction loss**

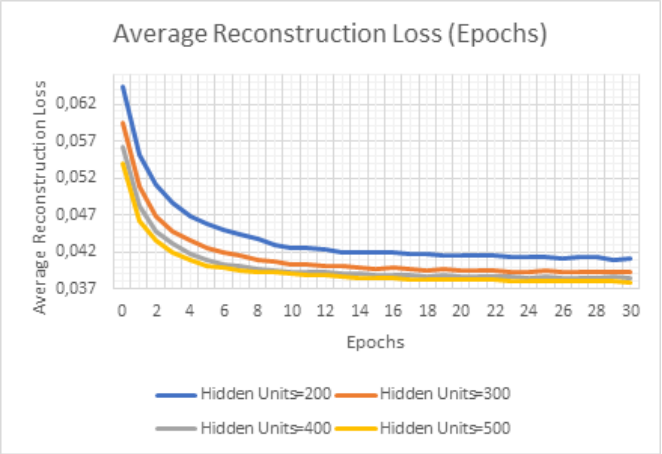

Figure 2: Average Reconstruction Loss for 200-500 hidden units.

From Figure 2, we can see that decreasing the number of hidden units increases the average reconstruction loss. For 500 Hidden Units the reconstruction loss is around 0.038 and for 200 Hidden Units around 0.041.

- **Receptive fields**

After learning, we want to examine the outcomes. Since the training has been conducting without labels, the evaluation boils down to examining the fidelity of the reconstructed images as well as the nature of the receptive fields that each unit develops throughout the learning process. So we decide to plot the weights to the visible layer to be visualized for the hidden units of interest. Each square corresponds to the 784 weights of one hidden neuron to the 28 x 28 = 784 visible neurons. The squares are ordered according to the probabilities of the corresponding hidden units to be 1 given the training set in decreasing order.

When we are on the first iteration, we can even recognize digits in them, the learned filters are rather complex. When we are on the 30th iteration, the receptive fields get more localized and show stroke-like features. For example in the first square, the weight is visualized at the position of the blue and red area, where blue represents negative weight, and red represents positive weight.

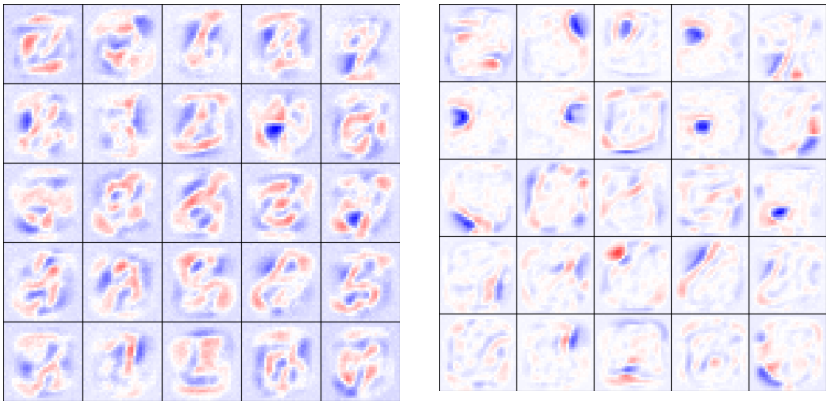

Figure 3 - Image representations of the hidden weights in the RBM in 1st iteration (L) and 30th iteration (R)

3.2 Towards deep networks-greedy layer-wise pretraining

- **Two RBMs in the stack**

Now, we want to use the RBM machinery in the previous section to build a network with two RBMs in the stack, trained greedily one layer after another, with contrastive divergence learning. We obtain the reconstruction losses below:

	Reconstruction loss of First RBM	**Reconstruction loss of Second RBM**
First iteration	0.0597	0.001
Convergence iteration	0.0366 (50th iteration)	0.000 (5th iteration)

We observe that the second RBM generates a very small reconstruction loss, approximately 0. That's why we can stack more and more RBMs on top. We assume this separate learning of each RBM contributes to a globally useful representation of the true data.

- **DBN with Gibbs sampling**

As specified in the pdf file, we constructed a three-layer DBN to perform the image recognition task. The architecture now becomes 784-500-500-2000 and the top stack can receive labels.

After 20 epochs of training:

- **Batch size (20)**
  Training accuracy: 83.38%
  Testing accuracy: 83.35%
- **Batch size (100)**
  Training accuracy: 85.5%
  Testing accuracy: 85.95

Image classification

- **Misclassification on figures shared some similar features**

Figure 4 - Misclassification on figures shared some similar features

- **Misclassification on hard cases**

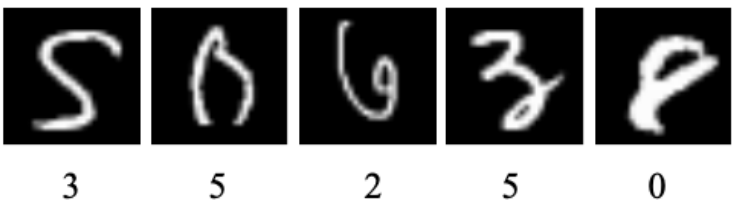

Figure 5 - Misclassification of hard cases

The numbers placed below the images are the prediction of the DBN network corresponding to each image shown in fig 4. (The order of predictions of the numbers and images are the same.) We can see that some misclassifications are due to the similarity in some strokes shared between different numbers. For example, in the first image, 2's curve looks like 7 without the bottom horizontal stroke. 4 and 7 both have this feature where the left part is a long stroke. As for the hard cases, those pictures are not so recognizable even for humans.

- **DBN in a generative mode**

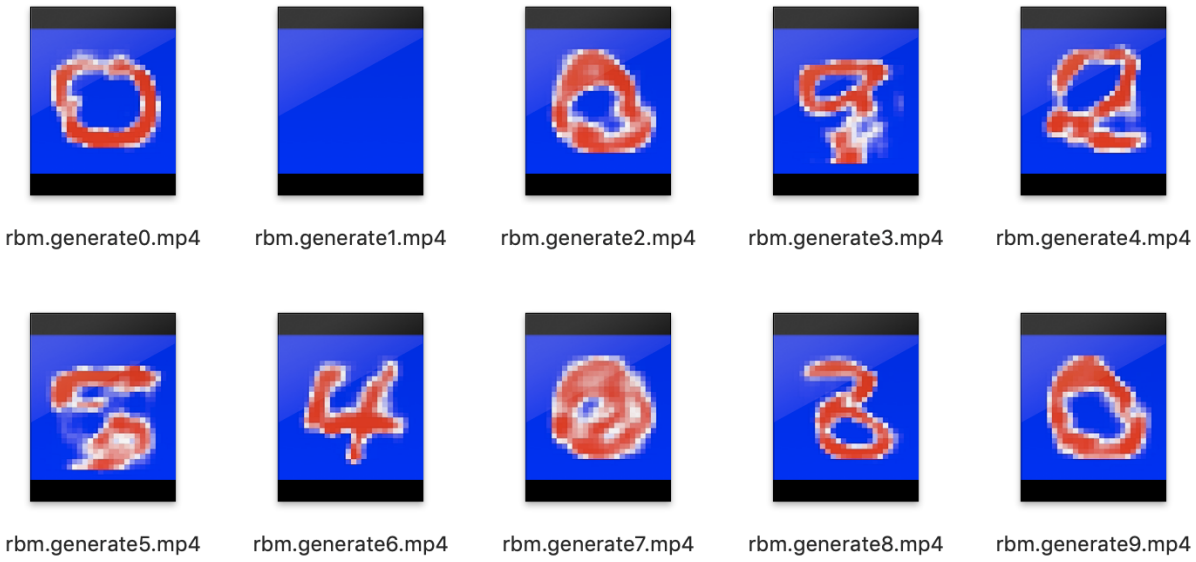

Figure 6 - Image generation by DBN

The results of the image generation depend on the training of RBM (such as batch size, epochs). Another factor is the number of iteration for Gibbs-sampling between the top and penultimate+label layer. Less interaction would yield images with more noise. As shown above, some generated images do look like their corresponding numbers such as 0,5, and 8.

As for numbers 4 and 6, the corresponding image looks completely like other numbers. This suggests that DBN does not take labels into consideration well. The reason behind this might be that the update rule for weights is designed to decrease the reconstruction error and it does not take classification into account when trying to minimize the loss.

4. Final remarks

In this lab, we get familiar with the learning process of RBMs, we use it for reconstruction and classification. We applied the basic algorithms for unsupervised greedy pretraining of RBM layers and supervised greedy pretraining of DBN.