Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

Restricted Baltzmann Machines and Deep Belief Nets

Shuyuan Zhang, Xinxing Guo, Wenqin Chen

October 6, 2019

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to understand the learning process of RBMs
- to implement basic algorithms for unsupervised greedy pre-training of RBM layers and supervised fine-tuning of the constructed DBN
- to design multi-layer neural networks for classification and generation tasks based on RBM
- to investigate the functionality and generative aspects of DBNs

2 Methods

We worked with Python as our programming language in this project, numpy/matplotlib libraries in Python were adopted.

3 Theoretical Questions

Why is there a guarantee to converge after sufficient number of alternating Gibbs sampling?

According to Law of Large Numbers(LLN), if performing the same experiment for many times, the average of the results obtained from a large number of trials should be close to the expected value, and it would tend to be closer with more and more trials. Therefore, in this case, there should be a guartee to converge after sufficient number of altering Gibbs sampling.

Why does stacking RBMs make the originally undirected connections directed? And why does the top RBM retain its undirected connections?

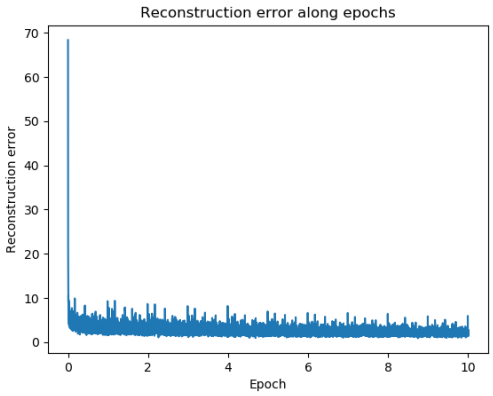
Because after we stack a new RBM on the top of our network, the lower RBM's weights needs to be untied and divided into recognition and generation parameters. The training process for them a little bit different in the wake-sleep algorithm so they need to be untied and updated individually. While we still need to do Gibbs sampling on the top RBM, that is, we are intrersted in joint

probability distribution over hidden and visible units. That's why the undirected connections are retained.

4 Results and discussion

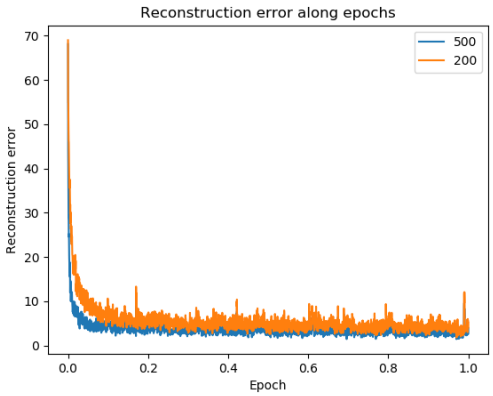
4.1 RBM for recognising MNIST images

In this part, we focused on training a single RBM with the MNIST data set and monitoring its activities.



Figur 1: Reconstruction error along epochs

We monitored and measured such stability by calculating the reconstruction error in this case. As figure 1 shows, it is clear that the reconstruction error on the mini-batch training set falls rapidly at the beginning of the learning process and then more slowly. Overall, the reconstruction error curve converges although it oscillates slightly in the later epochs.

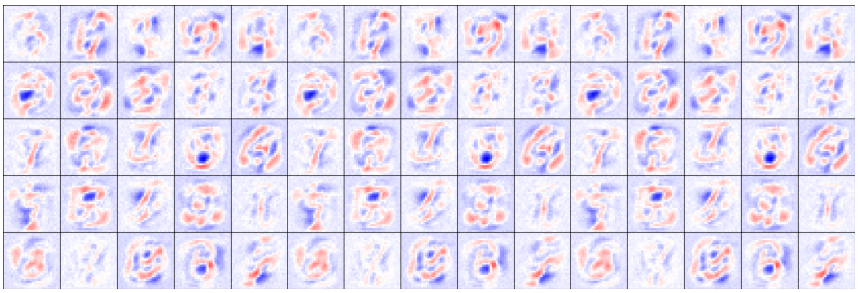


Figur 2: Reconstruction error along epochs with 500 and 200 hidden units

According to Figure 2 and Table 1,with the number of hidden units decreasing from 500 down to 200, the average reconstruction loss falls slower. Also, the final reconstruction loss of 200 hidden units will be a little higher at the end of the training process.

error / hidden units / iteration	500	200
0	66.1196	69.0853
500	5.4131	5.8240
100	3.1511	4.8330
1500	3.1440	3.7567
2000	2.3512	3.4364
2500	2.9253	2.8979

Tabell 1: Reconstruction error of 500 and 200 hidden units



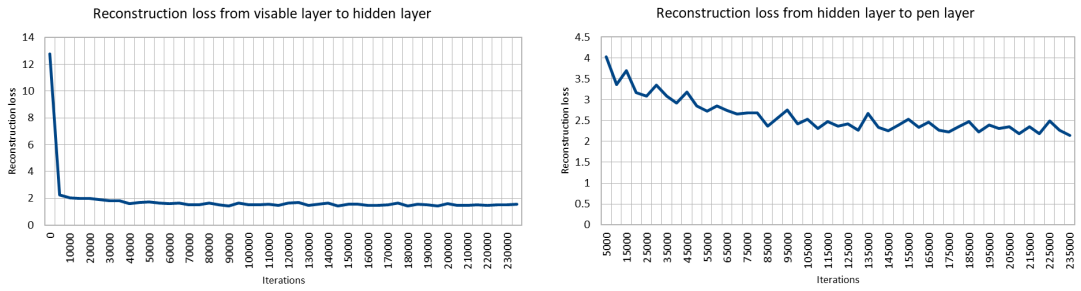
(a) 1000 iteration (b) 2000 iteration (c) 3000 iteration

Figur 3: Receptive fields of weight matrix

To study receptive fields, we reshape the weight vector for one hidden node as a matrix and plot it as an image for some randomly selected units in the hidden layer. As you can see from Figure 3, the weights have some certain patterns which are similar to some parts of the original images. In other words, the weights are trying to capture crucial features of the input image and different units are detecting different special features in the RBM.

4.2 Towards deep networks - greedy layer-wise pretraining

In this section, when building a network with two RBMs in the stack, we trained greedily one layer after another with CD algorithm. We designed the network based on the 500-hidden-unit architecture proposed by Hinton, and figure 4a and figure 4b show the reconstruction losses for the bottom RBM and the middle RBM in the stack, respectively.



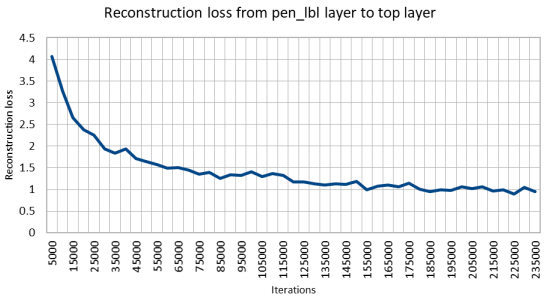
(a) RBM 1 (b) RBM 2

Figur 4: Reconstruction loss in the first two RBMs

We found that the reconstruction loss of the bottom RBM converges quickly, while the reconstruction loss of the middle RBM oscillates. It might be caused by the size differences in these two

RBMs.

In the second part of task 4.2, we were required to pretrain a DBN and use it to perform tasks such as image classification and generation. Hence we trained the topmost RBM (with 500 output units from the previous layer and 10 label units concatenated as the visible layer, and the top 2000 hidden units as the hidden layer) using CD algorithm. Figure 5 shows the reconstruction loss results after multiple iterations of the CD algorithm.



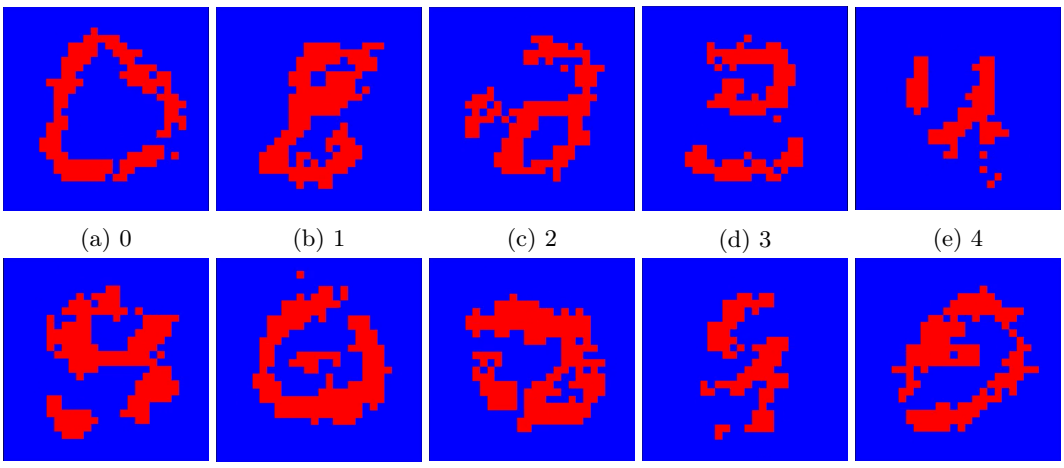
Figur 5: Reconstruction loss in *pen_lbl* layer to top layer

We trained the network on the training set for 20 epochs, and its recognition performance is shown in table below.

Data set	training set	test set
Accuracy	83.94%	83.82%

Tabell 2: Recognition accuracy of our pre-trained DBN

In the third section of task 4.2, we tested the generation capability of our DBN, the generated images of each class are shown in figure below:



(a) 0 (b) 1 (c) 2 (d) 3 (e) 4

(f) 5 (g) 6 (h) 7 (i) 8 (j) 9

Figur 6: Generated figures

We sampled random binary values at the pen- layer, concatenate it with a real label, then performed Gibbs sampling at the top RBM with the label fixed. Then we propagated them down to the hidden layer of lower RBMs, which can in turn be used to sampling the binary activity states of the visible

layer in lower RBMs. The binary activity in the hidden layer of the bottom RBM can be projected down to the visible layer and represent the generated images.

The generation performance is not so satisfactory, but still, we can see that the network is trying to generate images corresponding to the input label.

The key factors that may affect the quality of generated images are numbers of epochs, learning rate, and the labels. In this section, we found that softmax is a good choice to decide the final labels. Moreover, we found that the features of generated images are not so clear when compared to the original input patterns.

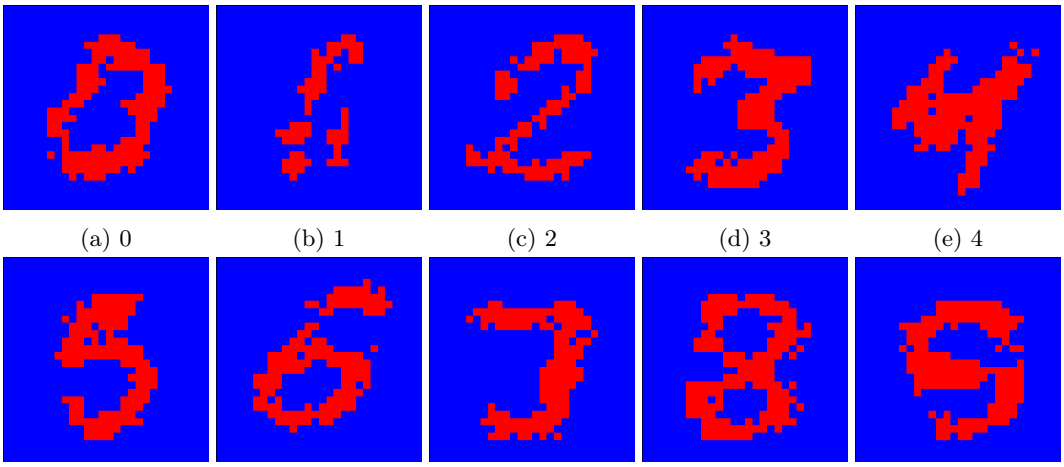
4.3 Supervised fine-tuning of the DBN

In this section, we compared the classification performance of the fine-tuned DBN with the pretrained DBN. After applying the wake-sleep learning rule ,the classification accuracy of the fine-tuned DBN grow to 84.76%, which is better than the pretrained DBN. Results are shown in table below:

Data set	training set	test set
Accuracy	84.43%	84.76%

Tabell 3: Recognition accuracy of our fine-tuned DBN

And the generated figures of the fine-tuned DBN are shown in figure below:



(a) 0 (b) 1 (c) 2 (d) 3 (e) 4

(f) 5 (g) 6 (h) 7 (i) 8 (j) 9

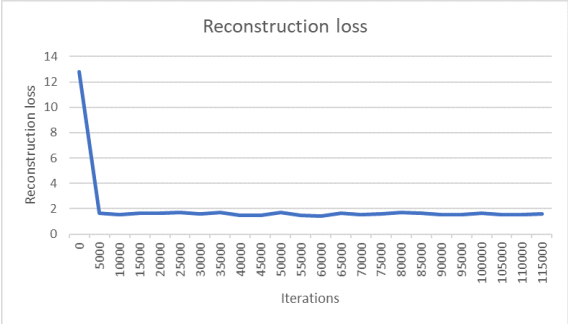
Figur 7: Samples of numbers

In this case, the generative model performs better after fine-tuning since the generative weights as well as weights in the top RBM are fine-tuned by the wake-sleep algorithm.

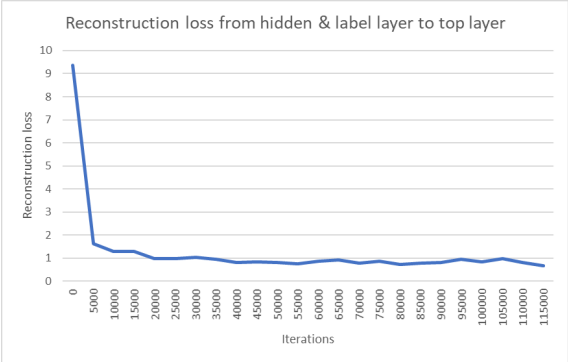
Next, We designed a simpler network with one hidden layer removed. The network now consists of two RBMs, one 784-500, another 500+10-2000.

The reconstruction error curves of the two RBMs are shown in figure 8 and Figure 9.

Also, to enable direct comparison of this network to the 3-layer network we used before, we trained this architecture on the dataset for 20 epochs and fine-tuned it for 10 epochs.



Figur 8: Reconstruction loss in visible layer to hidden layer



Figur 9: Reconstruction loss in hidden label layer to hidden layer

The generalization performance(classification accuracy reported on test data) obtained with the pretrained vs fine-tuned DBNs is shown in table4.

dataset	2 layer pre-trained	2 layer fine-tuned	3 layer pre-trained	3 layer fine-tuned
train	77.60%	79.08%	83.94%	84.43%
test	77.42%	78.95%	83.82%	84.76%

Tabell 4: Accuracy of two and three layers network

It is clear that the accuracy of three-layer network is higher than two-layer network. But the 2-layer network also reached a relatively high recognition accuracy. After fine-tuning, the accuracy improved a little bit for both architectures. The most important thing in this table is that the generalization performance of both DBNs are quite good. The performance on the training set and the test set seems similar. This is quite different from some deep feed-forward architectures, which may suffer from over-fitting, going into local minimum and thus a worse generalization performance.

5 Final remarks

In lab4, we apply basic algorithms for unsupervised greedy pretraining of RBM layers and supervised fine-tuning of the resultant DBN, design multi-layer neural network architectures based on RBM layers for classification problems, and study the functionality of DBNs including generative aspects