Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

RBM and DBN

Ravi Bir—ravib@kth.se
Bharat Sharma—bsharma@kth.se
Qiao Ren—qiaor@kth.se

April 28, 2020

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- Implement RBM and investigate the effects of hidden units.
- Implement DBN by stacking and greedy training of RBMs. Investigate recognition and generation of images using DBN.
- Implement fine tuning of DBN using wake-sleep algorithm. Investigate its effects on performance.

2 Methods

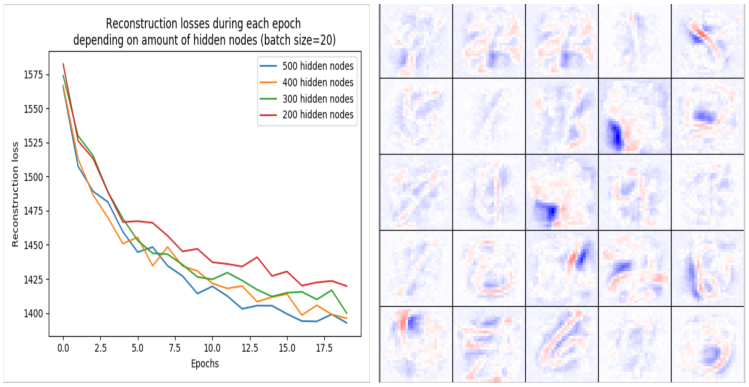
The lab was done in python using the provided code skeleton.

3 Tasks

3.1 RBM for Reconstructing Images

Convergence or stability can be monitored by following the same approach as other neural networks like early stopping. Training is stopped when reconstruction loss between the original and reconstructed image do not decrease by a significant amount. Another way to measure convergence is to see if $\triangle W \rightarrow 0$.

We noticed that reconstruction loss was higher for 300 and 200 hidden nodes when compared to 500 and 400 nodes and kept loss kept decreasing for all number of nodes with increasing epochs.



(a) Learning curves (b) Receptive fields

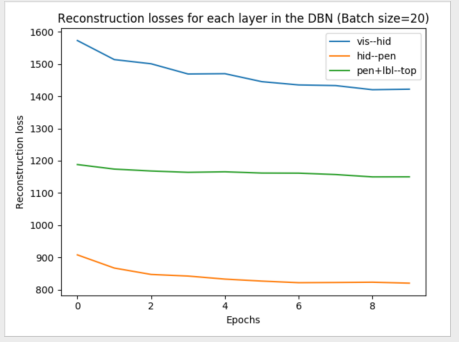
Figur 1: VIS-HID RBM

Receptive fields seen above have a resemblance to multiple numbers jumbled together.

3.2 Deep Networks - Greedy Layerwise Pre-training

We extended the single-hidden layer network to a deeper architecture by following the idea of greedy layer-wise pretraining.

Using the RBM machinery in the previous section, we built a network with two RBMs in the stack, trained greedily one layer after another, with CD algorithm. It is based on the 500-hidden-unit architecture. Figure 2 shows the reconstruction loss at the different layers.



Figur 2: Reconstruction loss at different layers

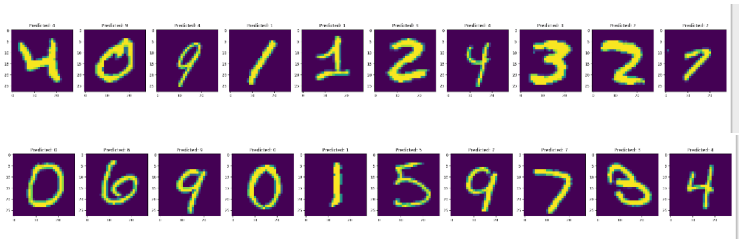
Image Recognition

Next we pretrained a DBN to perform image recognition. The topmost RBM was trained with CD algorithm. The label units were clamped correspondingly to the representations of digit images. Feedforward propagation of the input images through hidden layers of DBN was performed, followed by multiple iterations of Gibbs sampling. The accuracies that we managed to achieve can be found in Table 1.

Dataset	Average Accuracy
Train	83.76
Test	83.43

Tabell 1: Average Accuracies

Figure 3 shows some samples and their estimated labels for training(left) and test(right) labels . As you can see some of them are incorrect but majority of them are correct.



Figur 3: Samples and Estimated Labels for training(top) and test(bottom) data

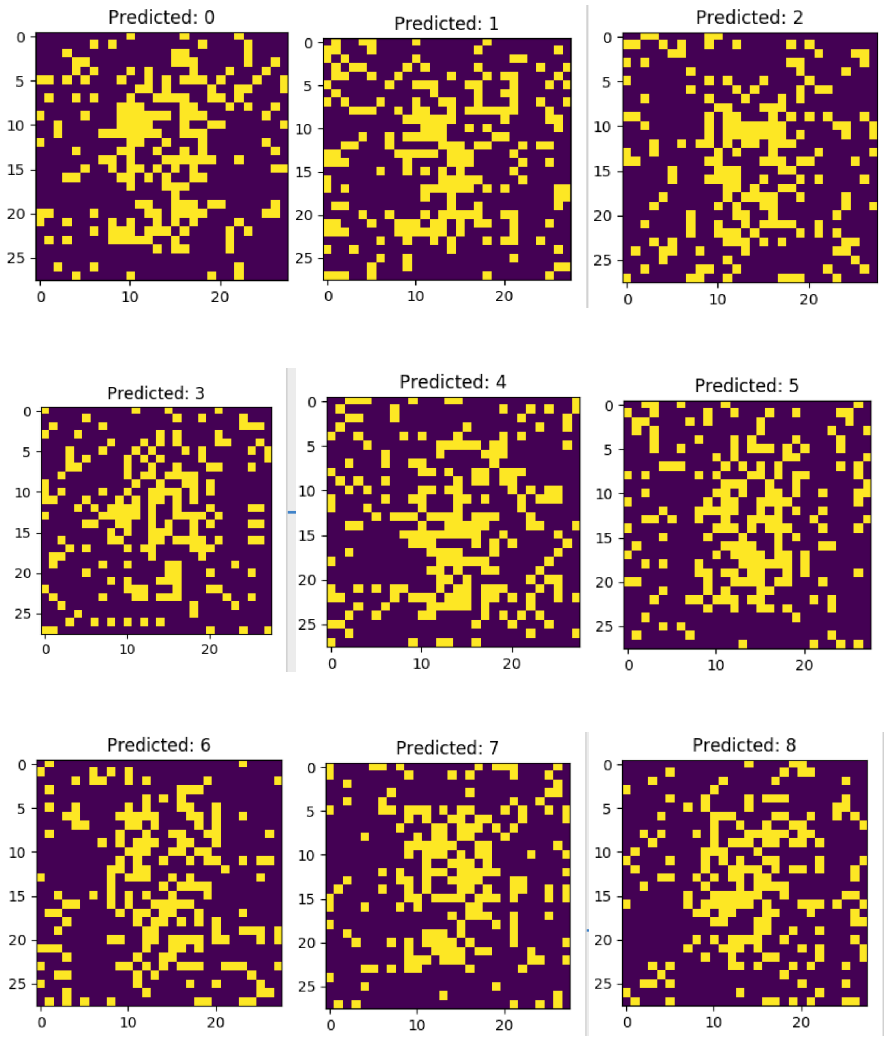
Image Generation

Next we used the pretrained DBN in a generative mode. The label units are clamped and Gibbs sampling is performed. A probability of activation is obtained for each unit in the concatenated layer of the top RBM. Then we can sample binary values from these probabilities across the hidden layer, propagate them down to the hidden layer of the lower RBM where we obtain another set of probabilities, which can in turn be used for sampling the binary activity states in this layer. Finally, these binary activities in the hidden layer of the bottom RBM can be further projected down to the input (visible) layer and represent the generated images.

Since we obtain probabilities in the top RBM, the sampling process and propagation of activity down in the network to the input layer can be done multiple times and so for the same clamped label we can generate numerous samples. The results are shown below in Figure 4. Most of them are not so good.

The batch size and number of epochs in the RBM training, as well as the number of iterations of Gibbs sampling, all determine the quality of the generate

image. More epochs and iterations of Gibbs sampling means better images are generated.



Figur 4: Image Generation before fine tuning

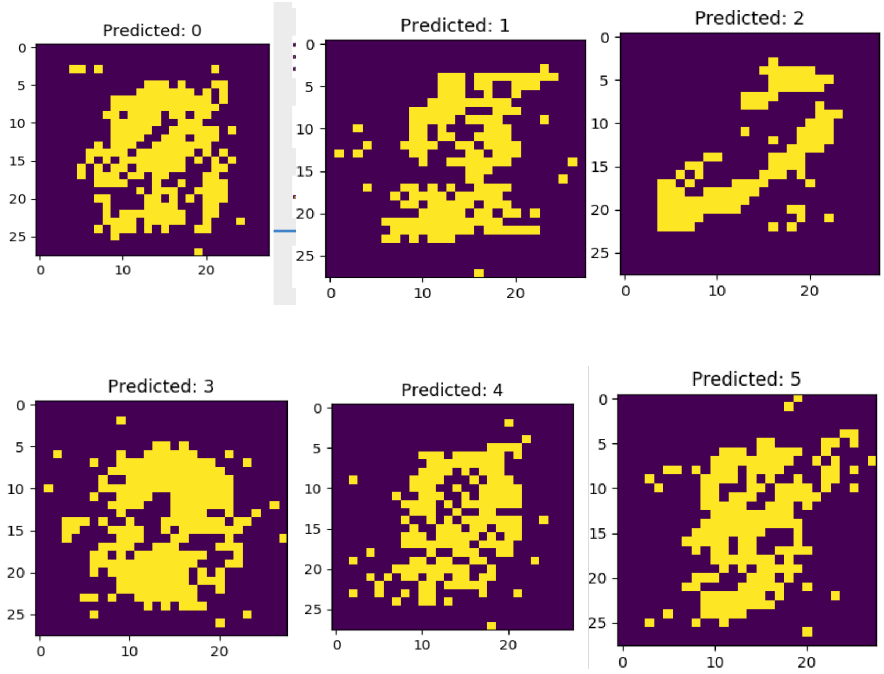
3.3 Supervised fine-tuning of the DBN

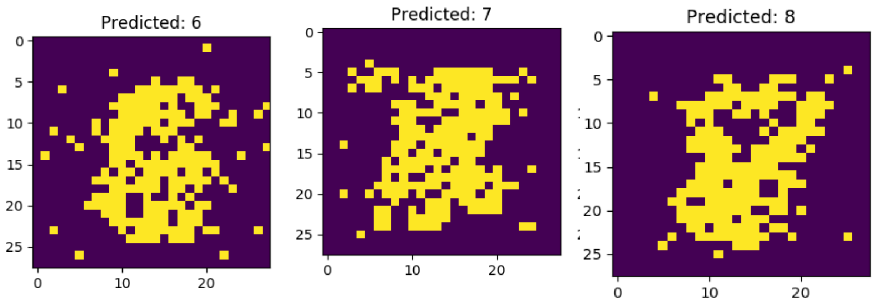
Building on the pretrained DBN in the previous part, we trained the network by implementing the wake-sleep approach that exploits the information about the target labels for input samples. We then compared the performance of image recognition and generation both before and after fine tuning. The accuracies that we managed to achieve for image recognition after fine tuning can be found in Table 2. The accuracies are slightly lower than before fine tuning.

Dataset	Average Accuracy
Train	79.34
Test	79.03

Tabell 2: Average Accuracies after fine tuning

The images that were produced after fine tuning can be seen in figure 5. The quality of these generations has significantly increased when compared to before fine tuning. So from our results, we can see that fine tuning significantly increases the performance of image generation and very slightly lowers the performance of image recognition.





Figur 5: Image Generation after fine tuning

4 Final Remarks

This was a useful assignment to get deeper knowledge about RBMs and DBNs. However, we found this lab to be the hardest of all others and one of the main challenges of this assignment was the run-time. Also, the theory as well as the implementation of DBNs was quite difficult to grasp. The provided code skeleton helped us in that regard.