Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

RBM and DBN

Erik Molitor, Johanna Norén and Kidus Getahun

October 13, 2021

1 Main objectives and scope of the assignment

Our major goals in the assignment were:

- Get familiar with key ingredients of deep neural network architectures, and specifically for this lab RBM's (Restricted Boltzmann Machines) and DBN's (Deep Belief Nets).
- Train and test the different networks in order to generate and recognise images (MNIST images in particular).

2 Methods

In this lab we've worked in python, since the given start code for this lab has been written in python. We've used libraries that we found were necessary and useful.

3 Results and discussion

3.1 RBM for recognising MNIST images

Measuring convergence and stability in behaviour of units with the RBM can be done by plotting the reconstruction losses for different number of hidden units in the RBM.

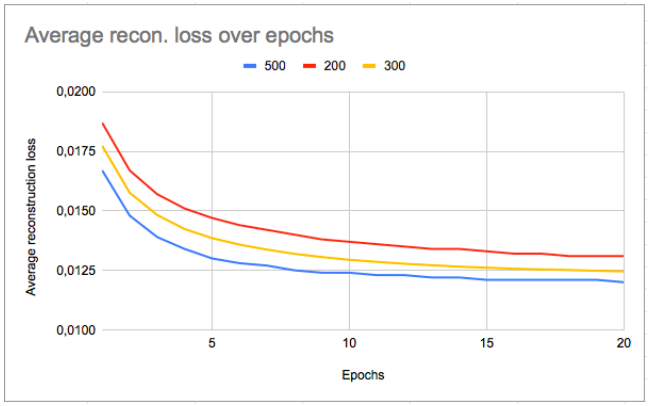

Figure 1: Average reconstructions losses for different hidden nodes.

In the plot above it can be seen that the reconstruction losses are decreasing, meaning they are converging and becoming stable as we iterate more with the alternating Gibbs sampling (can be shown as number of epochs). We can also tell from the plot that the more hidden units we have in the RBM, the performance is also more efficient in learning the data.

After training and learning the RBM we looked at how well it could recognise the outcome, which in this case are MNIST images.

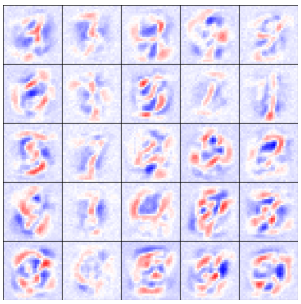

Figure 2: Plot of outcome (MNIST images) at iteration 0.

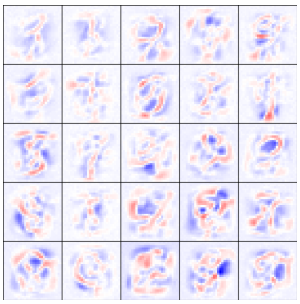

Figure 3: Plot of outcome (MNIST images) at iteration 5.

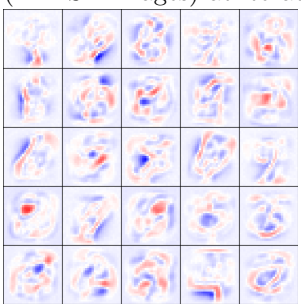

Figure 4: Plot of outcome (MNIST images) at iteration 10.

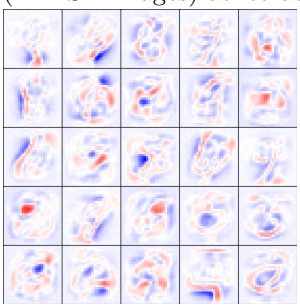

Figure 5: Plot of outcome (MNIST images) at iteration 15.

Above are figures of the receptive fields as images, where it is possible to interpret some of the numbers in the image. Comparing the images with the reconstruction losses we can see that there is a distinct difference between the images of iteration 0 and iteration 5 but not too much difference between the rest of the images. That goes hand in hand with the convergence we have seen in the plots of the reconstruction losses.

3.2 Towards deep networks - greedy layer-wise pretraining

As we did in the first task, we plotted the reconstruction losses for the now stacked RBM's. In the plot below are the results for the different layers, where blue line corresponds to the visible layer to hidden layer, red line corresponds to hidden layer to pen ultimate and lastly, yellow line corresponds to penultimate (with added labels) to the top layer.

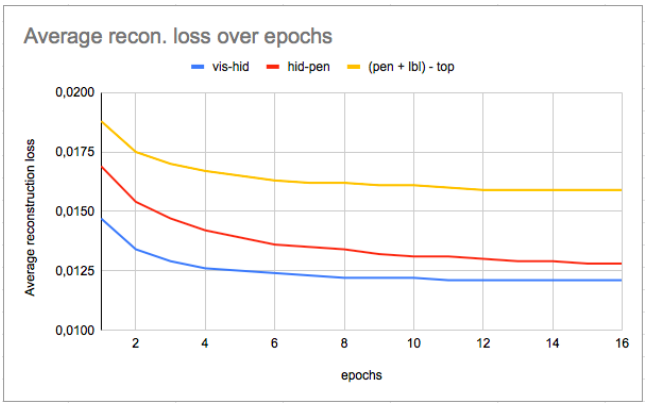

Figure 6: Average reconstructions losses for stacked RBM's.

Also here, the reconstruction losses are decreasing, hence converging and stabilizing like in the previous task.

	Run 1	Run 2
Accuracy, training	88,9 %	88,78 %
Accuracy, testing	88,54 %	89,66 %

To be completely sure about the stability, we decided to train and test the stacked RBM's two times. Above are the results of the accuracy presented and there are almost no difference in the accuracy between the two runs.

Through comparisons between the training and generated images we came to the conclusion that the generated images could have been improved. The key factors for quality was more or less avoiding misclassification and using better training data. Many of the numbers could look alike although they were different for example 6 and 4. This is because they were handwritten which led to many numbers looking alike. If we were to use more similar data then the generated images would more likely be better. The conclusion we therefore have drawn is that The algorithm worked well in labeling but could be improved in generating.

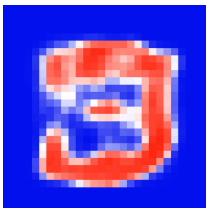

Figure 7: Generated image

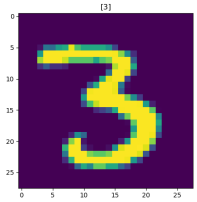

Figure 8: Training image

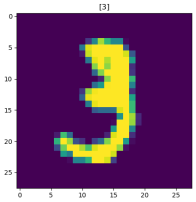

Figure 9: Training image

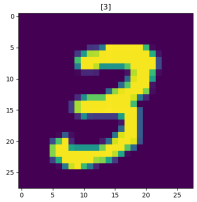

Figure 10: Training image

4 Final remarks

It was interesting lab to see how it could both classify and generate from quite crude data. It was also interesting to work with algorithms that take so long time to train. Makes one ready double check all values before running the code.