Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

Restricted Boltzmann Machines and Deep Belief Nets

Mustafa Al-Janabi, Einar Lennelöv - Group 15

February 28, 2020

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- To implement the methods used in RBM and DBN networks.
- To examine the properties of the RBM and DBN for image recovery, classification, and generation.

We have focused on qualitative findings that help in the understanding of the algorithms, as opposed to more rigorous numerical results.

2 Methods

All of the code was written in Python. The data used is taken from the MNIST dataset.

3 Results and discussion

3.1 RBM for recognising MNIST images

Restricted Boltzman Machines, RBMs, have the capacity to store input data, and recover this stored data when given sufficiently. Since the recovery is done by sampling from a probability distribution, this means that the recovered image may not be identical to the stored image. The level of similarity is dependent on how certain the probability distribution is. One key aspect to investigate is thus how well the RBM can recover data. Figure **??** shows the reconstruction error for different numbers of nodes in the hidden layer. The error quickly drops after a few gibbs sampling iteration, and continues to drop to a low value, and that more nodes lead to better reconstruction.

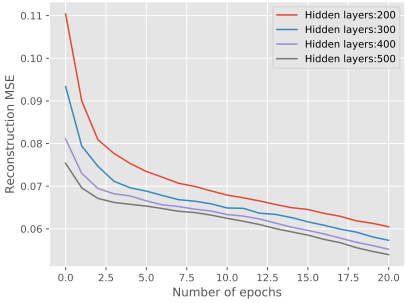


Figure 1: Convergence of reconstruction loss.

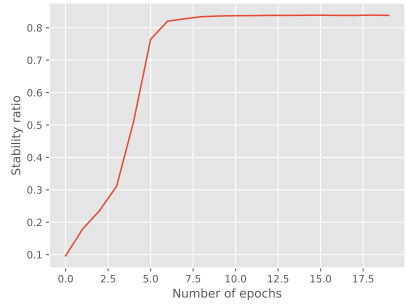


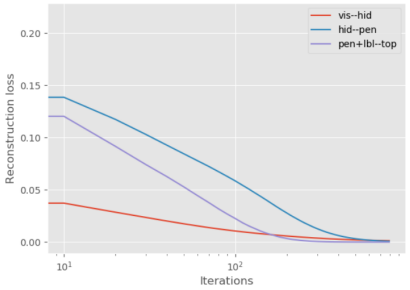
Figure 2: Stability analysis.

Although the reconstruction itself seems to converge, it is also interesting to see if the parameters themselves converge. We investigate this by calculating the percentage of weights that do not change by more than 0.001 in one step. The results are plotted in Figure 2. We can see that about 20% the weights change every for any given iteration.

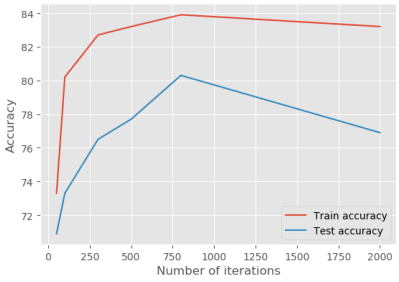
3.2 Toward deep networks - greedy-layerwise pretraining

Although RBM can be used for certain tasks such as denoising and image recovery, they are not immediately suitable for tasks such as image recognition and generation. One reason for this is that, unlike the feedforward networks we have worked with previously, there is no obvious built in capacity to handle targets or labels. A simple way to deal with this issue is to concatenate the image data with an encoding of the label while training. This causes the RBM to also memorize corresponding labels. One can then simulate "lost labels" by attaching a non-informative prior distribution to the label section of test data, and using Gibbs sampling to recover the label.

In order to further improve the performance of such a model is may be necessary to refine the inputs to the RBM. One way to accomplish this is to stack one or more RBM under the classifying RBM. The lower RBMs are then responsible

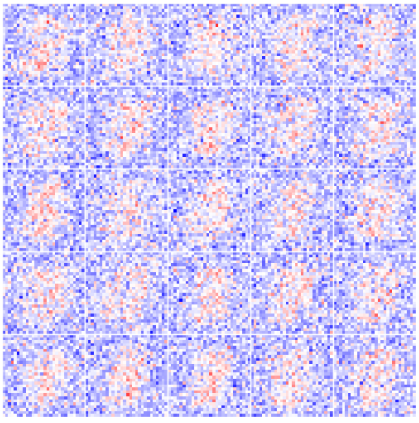


(a) Convergence of reconstruction loss.

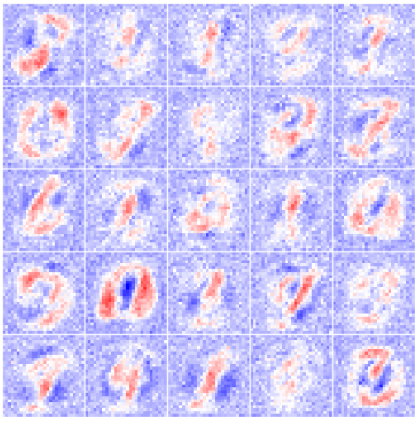


(b) Accuracy on train and test set.

Figure 3



(a) 50 iterations.



(b) 1500 iterations.

Figure 4: Weights into randomly selected nodes.

for abstracting the image data into features, which is then passed on to the classifying RBM.

Here we build such a model, a deep belief network (DBN), and train it using greedy layer-wise training. The first concern is the convergence of the individual layers. This should be straightforward since they are trained separately, but it is important each layer is given sufficient time to train, without also overfitting. In this case we have trained the RBMs on 600 samples using a momentum-based update rule. Figure 3a shows the reconstruction loss plotted against the epoch number. It is clear that by 800 iterations all of the layers are able to reconstruct their inputs nicely.

Since we would like to use the DBN for classification, we need to also investigate the classification accuracy. This time we train the DBN on 6000 samples, using a varying number of iterations. The test set contained 100 samples. Figure 3b shows the result on both the train and test sets. The plot suggests that there is a peak somewhere between 800 and 2000 iterations, and that too many iterations lead to overfitting. Overall we achieve a maximum of about 84% on the training set and 82% on the test set.

One of the main advantages of the DBN compared to a regular feedforward network is that we also can use it in a generative mode. Figure **??** shows the resulting images when labels are fed to the network and propagated down to the visible layer. The images plot the probabilities in the visible layer, for the final gibbs sample.

Lastly, we visually examine the features that the RBM learns, by plotting the weight matrices as images for a few randomly chosen nodes. After 50 iterations no distinct features have yet been found, but after 1500 iterations there are clear shapes that resemble pieces of digits.

3.3 Supervised fine-tuning of the DBN

After the network has been trained to recognize and generate handwritten letters based on the label in the previous section. Now comes the time to improve its performance using in a supervised manner. In contrast to greedy-layer-wise learning, in this section we employ a wake-sleep algorithm. The wake sleep algorithm consists of two parts. In the wake cycle we move bottom to top through the network. Each layer has its weights decoupled from the other. On each step up a layer, we perform a generative step and update the weights in a manner that is reminiscent of the preceptron learning, hence the supervised nature of the fine tuning process. On the top layer Gibbs sampling performed with clamped layers as in the previous section. The second part to the wake-sleep algorithm is the sleep phase. During this phase we move top to bottom through the network. At each layer we perform a recognition step and update the weights in a similar manner to the wake phase.

The results found in this part were somewhat surprising. Contrary to the belief that the wake-sleep algorithm should improve the performance of the network, we found that in some aspects it made the network substantially worse and in

others improvements where found. Initially the fine-tuned DBN was found to result in a remarkable decrease in recognition accuracy. Upon further, extensive this proved to not hold. In fact, it depended on many of the parameters of the network. One of the parameters we chose to study careful was the learning rate. In this study we use the model which was trained on 6000 samples with 800 iterations, from the previous section using greedy-layerwise pretraining.

Studying the learning rate, we found two different architectures that lead to two very different result. The first architecture will be referred to as the **recognition specialised** architecture and the second will be referred to as the **generation specialised** architecture.

The **recognition specialised** architecture is obtained by setting the learning rate to 0.001 and dividing by the total number of data points in the mini-batch during the weight updates in the *update_generate_params* and *update_recognize_params*.

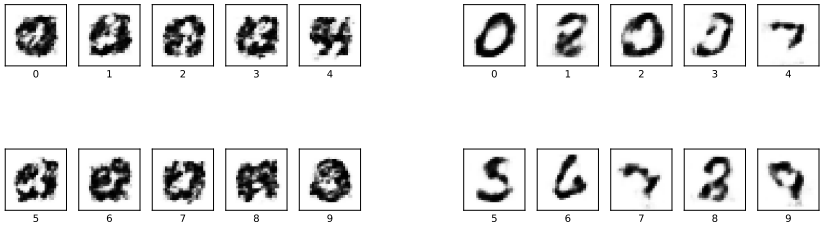
The **generation specialised** architecture is obtained by setting the learning rate to 0.01 and updating the weights in *update_generate_params* and *update_recognize_params* without any scaling by the number of samples.

The names for these two architectures come from the results obtained in Figures 5 and 6. Both images in Figure 5 are obtained by running the wake-sleep algorithm for 10 iterations. As we can see in Figure 5 on left the generated pictures are not recognisable as hand-written numbers. Meanwhile on the right the generation specialised model comes pretty close to a full sweep. The numbers $0, 5, 6, 7, 8, 9$ are recognisable indeed. In contrast to that, when studying Figure 6 we find the surprising fact that the architecture that has the best generation capabilities is also the one with lowest recognition accuracy. This is an unexpected result, since we expected that a layer that has a high label recognition capability also should be able to generate from labels well, furthermore the fine-turning algorithm seems to yield no improvement to the accuracy of the model that was pretrained using greedy-layerwise training.

The reason for this peculiar result, where the network is either good at recognition or generation with no improvements on the accuracy is not yet understood. Furthermore, it is very likely that the recognition specialised is also on a path of decreasing accuracy, but since we divide by the number of samples in the weight updates of the generation and recognition steps, the learning rate is too small for the effect in the generation specialised architecture to take place within 10 epochs. Despite the peculiarity the results obtained in Figure 5 are indeed fascinating.

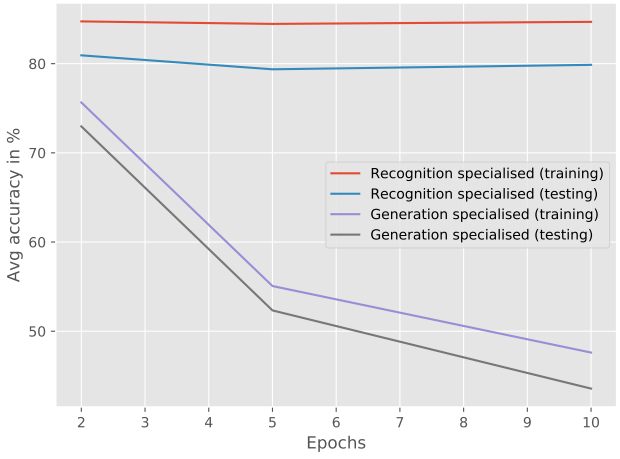
4 Final remarks

We liked that the assignment presented a challenge, and that the results were quite impressive (being able to generate digits). Although RBMs might not be used as much in practice anymore, it was interesting to go through the concepts involved.



(a) Recognition specialised (b) Generation specialised

Figure 5: The generated handwritten pictures of every label.



trimtrim

Figure 6: Accuracy as a function of the number of epochs for the two architectures described.