Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

Restricted Boltzmann Machines and Deep Belief Nets

Isabella Rositi, Gustav Thorén and Nicolas Guy Albert Wittmann

23 Febraury, 2023

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- understand the key concepts of training RBMs and apply basic algorithms for unsupervised greedy pre-training of RBM layers
- study the functionality of DBNs

We first applied RBM on the MNIST data set by implementing contrastive divergence. Later we focused on DBN and we designed a multi-layer neural network architecture based on RBM layers in order to perform classification by implementing unsupervised greedy layer-wise pre-training.

2 Methods

The code was implemented in Python with the use of the *Numpy* and *Matplotlib* libraries starting from the code framework provided.

3 Results and discussion - Part I: RBM for Recognising MNIST images

We implemented an RBM with binary stochastic units and we trained it with a unsupervised algorithm, the contrastive divergence algorithm, in order to learn data representations. The architecture employed has 500 units in the hidden layer and takes a flattened picture of 28 x 28 pixels as input.

3.1 Convergence of the Training Process

We first of all investigated the average reconstruction loss throughout the iterations of the algorithm and studied its behaviour as the number of hidden units decreases from 500 to 200. Figure 1 we can see that the general behaviour is the same for all values of hidden units, with a strong decay between the beginning and the first 400 iterations and a less steep decay throughout the rest of the training process. As the number of hidden units increases, the error decreases slightly.

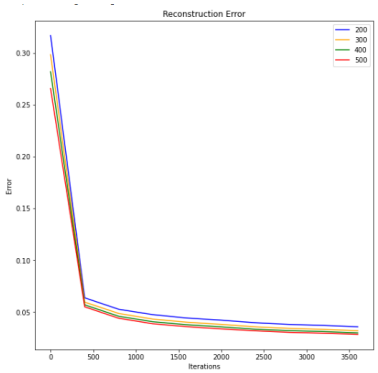

Figure 1: Reconstruction error throughout 4000 epochs for different numbers of hidden units, going from 500 to 200.

Moreover, to study the convergence of the training process, we also analysed the behaviour of the weights and bias updates throughout the iterations. As shown in figure 2 we observe a similar behaviour as for the reconstruction loss, meaning that as the training progresses, the difference between the updated weights and biases decreases. For the first 400 iterations the decay is very steep (especially for the weights) and it is less steep throughout the rest of the training process.

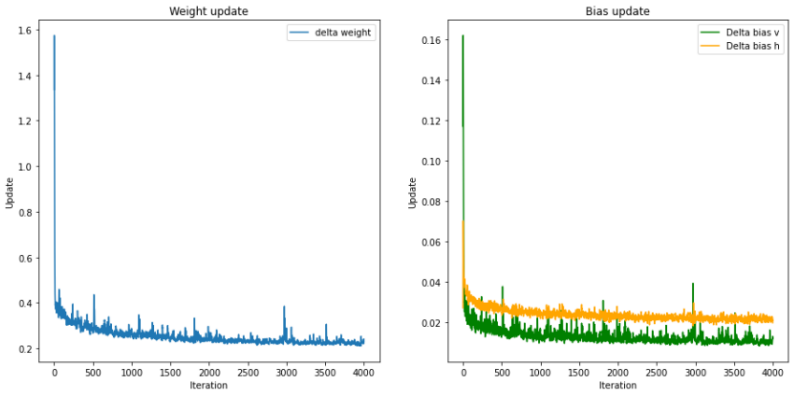

Figure 2: Weights and bias updates (normalised) for 4000 iterations.

3.2 Digits Reconstruction

In Figure 3 we can see the reconstruction of different digits compared to the original test images. The resemblance is evident, and we can note how numbers with loops tend to be more difficult to reconstruct. For example the 3 presents almost a close lower loop in the reconstruction, even if it was clearly separated in the image, as well as the 8 which results to be pretty noisy and undefined.

Figure 3: Original image and reconstruction for 10 different numbers after 400 iterations.

3.3 Receptive Fields

From Figure 4 we can see that throughout the training the receptive fields of hidden units change and tend to look like components of handwritten digits. This highlights the fact that each unit specializes in recognising specific patterns like curves and lines. Indeed, units do not identify single digits, but a combination of their features. We can see that the patterns are discernible after 400 iterations already, but after the whole training process these are more defined.

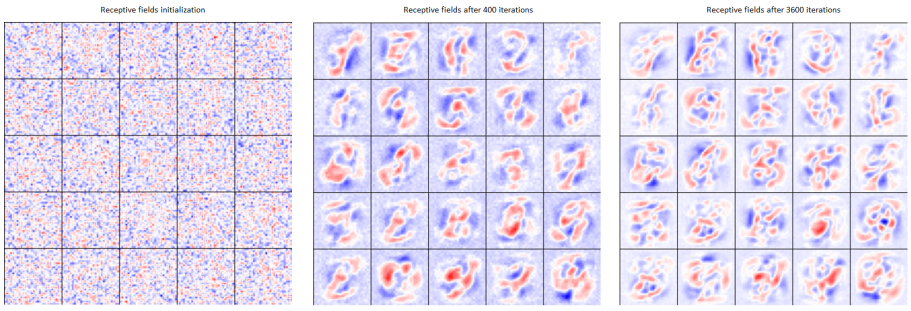

Figure 4: Receptive fields of an RBM before training, after 400 epochs and after 3600 epochs.

4 Results and discussion - Part II: Towards deep networks: greedy layer-wise pre-training

A DBN utilizing Hinton et al.'s (2006) architecture with three stacked RBNs was used. The top layer was concatenated with the 10 labels in the visible layer

of the RBN.

4.1 Training the Network

The network was pre-trained using an unsupervised greedy layer-wise training. After each layer had finished training it was converted to a directed network and had its weights locked. The layer was converted to a directed network in order to facilitate fine tuning if desired. After locking the layer a new layer was added on top such that the new layers' visible part became the old layers' hidden parts. For the top layer the training was done in a supervised manner by concatenating the corresponding labels to the visible layer.

4.2 Recognition

We can use the trained network to recognize digits. To do so we fix the image on the visible layer and process it bottom-up. In the top layer, we use alternating Gibbs sampling and then read the output from the label nodes. The network had an accuracy of 89.86% on the training set and an accuracy of 90.02% on the test set. This result was from no fine tuning of the layers after the pre-training, with fine tuning the performance could likely be improved.

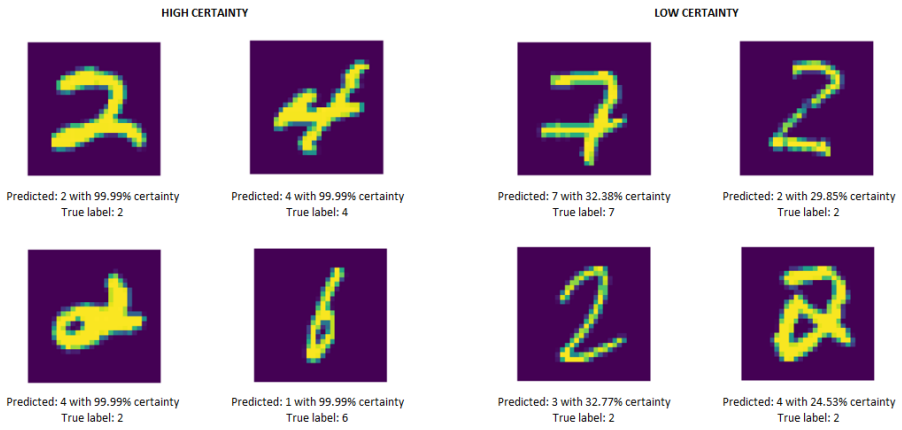

Figure 5: Recognitions that the network gave the highest and lowest certainty to. The top row where correctly identified and the bottom row incorrectly classified digits.

Fig. 5 shows the the certainty of the network when recognizing digits. The least certain guesses in the data set was the two incorrectly classified digits in the bottom left of the figure. The figure shows how the networks predicts some labels incorrectly with a very high certainty, when we as humans can clearly see that it is wrong or are uncertain. On the other hand, it also does the opposite, it has a low certainty for the digit in the top left while we humans can clearly tell that it is a two.

4.3 Generation

To generate images from a label, we first produce a random image we process bottom-up in the network, then we clamp the chosen label in the *pen+lbl* level, and use alternating Gibbs sampling while clamping the desired label at each step. We then sample top to bottom in the network to generate an image. Fig. 6 shows the results of generating the digits 0-9 10 times each. The only digits which can clearly be recognized is the sevens, all other digits are unidentifiable, but we can still recognize some patterns. The poor performance is likely due to the network not fine tuning after the pre-training is done.

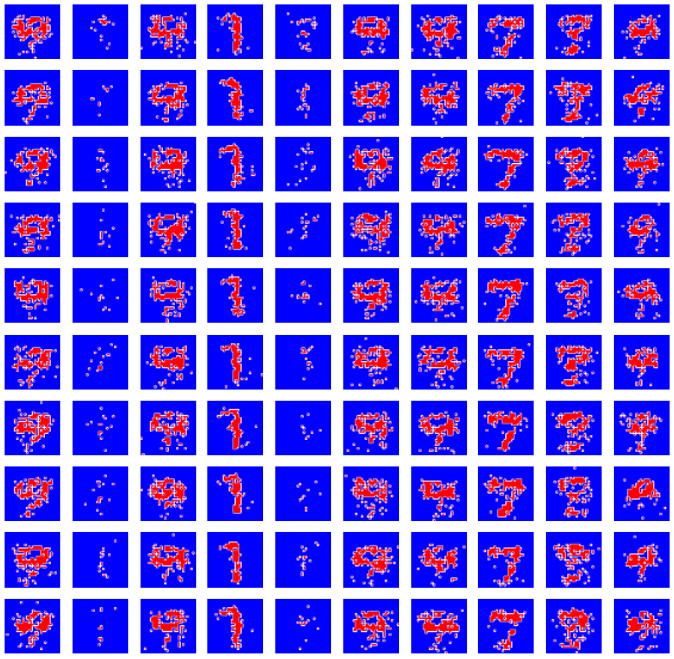

Figure 6: Generation of 10 images per digit using Deep Belief Nets

As can be seen in section 3 the RBM is not the best at reconstructing images. In the simplest case of a single layer there is already a lot of noise, see Fig. 3, adding more layers only increases the noise and too generate 784 pixels from a single label is difficult without fine tuning the network.

5 Final remarks

RBMs are efficient networks to reconstruct images, whose simple architecture allow us to investigate the way that the hidden layer learns by looking at the receptive fields. Moreover, by tracking the values of the weight and bias updates, we can monitor the convergence of the training process.

DBNs are a powerful network that combines unsupervised and supervised techniques in order to both recognize and generate images. They are computationally more expensive than RBMs since they are composed of many of these networks stacked one on top of the other. The networks should ideally be fine tuned after the pre-training.