Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

Restricted Boltzmann Machines and Deep Belief Nets

Etienne Jodry, Frano Rajic, Ivan Stresec

October 9, 2020

1 Main objectives and scope of the assignment

Our major goals in the assignment were:

- Gain insight into implementation details of RBMs and DBNs
- Applying basic algorithms for unsupervised greedy pre-training of RBM layers
- Analyze the abilities of RBMs and DBNs in the context of the MNIST dataset
- Derive conclusions about the behaviour of RBMs and DBNs

2 Methods

We used Python 3.8 with some of its built-in libraries along with Matplotlib and NumPy for all of our experiments. Our implementation was built on the code framework for RBMs and DBNs attached to the lab assignment. We also often utilized Jupyter Notebooks and Google Sheets for result visualization and analysis.

3 Theoretical questions

3.1 Restricted Boltzmann machine convergence

The probabilities of the RBM's state are guaranteed to converge given an infinite number of alternating Gibbs sampling steps, as Gibbs sampling is a Markov

Chain Monte Carlo method and the created samples constitute a Markov chain. Markov chains always converge to a stationary distribution. This stationary distribution is the Boltzmann distribution (a.k.a. Gibbs distribution) because the RBM is a restricted version of the Boltzmann machine. A Boltzmann machine can be proven to always converge to a stable state modeled by the joint distribution of its undirected connections. This stable state of BM's state probabilities is the Boltzmann distribution, which is why the machine is called the Boltzmann Machine.

3.2 Pretraining deep networks with greedy layer-by-layer training

Stacking RBMs makes the originally undirected connections directed because in this case the updates are only going one way: from bottom to top the previous RBM layer is fed as visible of next network and conversely during the backward pass. More precisely, this untying brings about the possibility to change the probability distributions separately for generative and recognition purposes, which can, for example, be used for supervised fine-tuning of a DBN that has used unsupervised greedy pre-training.

The top RBM, however, retains its undirected connections as they represent infinitely many layers with directed connections and are used to converge during Gibbs sampling to the joint distribution modeled by the symmetric connections. Note that an RBM converges to a Boltzmann distribution given an infinite number of Gibbs sampling steps, though a finite number of steps is used to get an approximation that is faster to determine but still effective.

4 Results and discussion

4.1 RBM for recognising MNIST images

We used the code framework attached to the lab assignment to implement unsupervised learning of RBMs based on the contrastive divergence with one step. We loaded the MNIST dataset and used it to learn networks of architectures $784-500$ and $784-200$. Besides the number of hidden nodes, other hyperparameters controlled number of epochs, batch size, weight decay, learning rate and momentum. We did not use extensive testing of configuration, but run tests to observe specific differences.

To monitor the development of the learning process, we used the reconstruction loss defined as the difference of the visible layer before and after one step of Gibbs sampling was made with training dataset initially set as the visible layer. This gave us a brief idea, but should not be trusted as much as the reconstruction loss does not necessarily mean the learning process converged to some weights. Therefore, we also measured the sum of absolute value of weight updates where we could notice smaller and smaller weight updates with the number of iterations. When the sum gets drawn closer and closer to zero, we can consider that the learning process reached a certain amount of stability (Fig. 1 illustrates this).

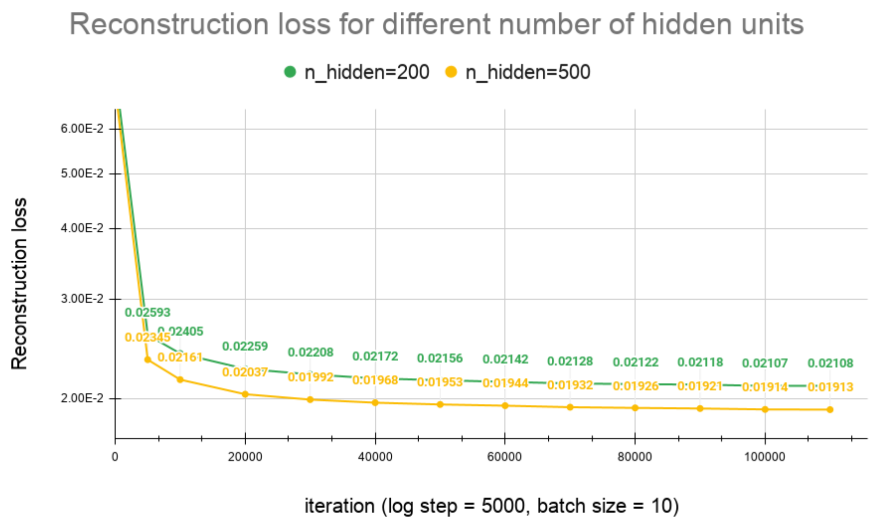


Figure 1: Reconstruction loss for the two architectures, $784 - 500$ and $784 - 200$

In figure 2 we can see the impact of a different number of nodes in the hidden layer on the reconstruction loss. More nodes allow for a better reconstruction.

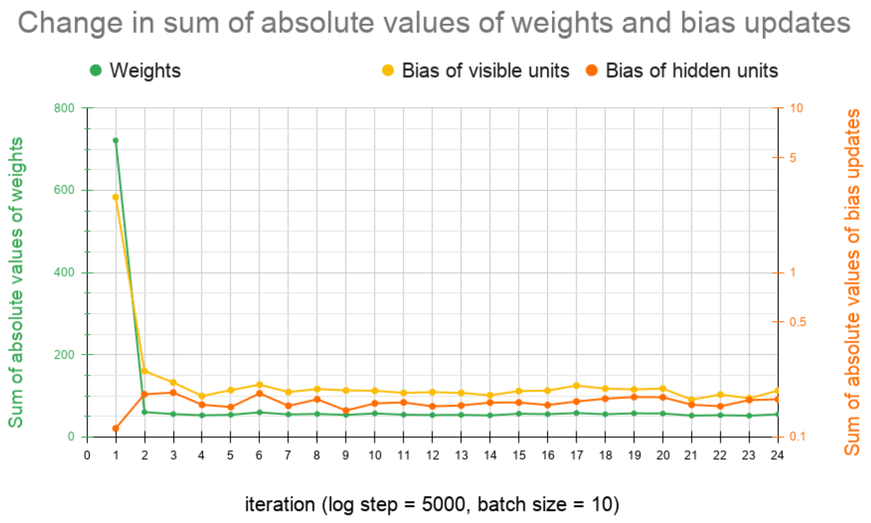


Figure 2: Change in sum of absolute values of weights and biases for a RBM of 500 hidden nodes with momentum and no weight decay

We and plotted receptive fields corresponding to the weights from a particular hidden node to the visible layer. We observed that some nodes have prominent parts that are specialized for the detection of curves and lines, as seen in the first row in figure 3. However, most of the receptive fields were hard to explain, like those in row 2 of figure 3. That is because all of the hidden nodes hold abstract and latent representations of the visible input, in this case forcefully compressed from 784 units into 500 and 200 units.

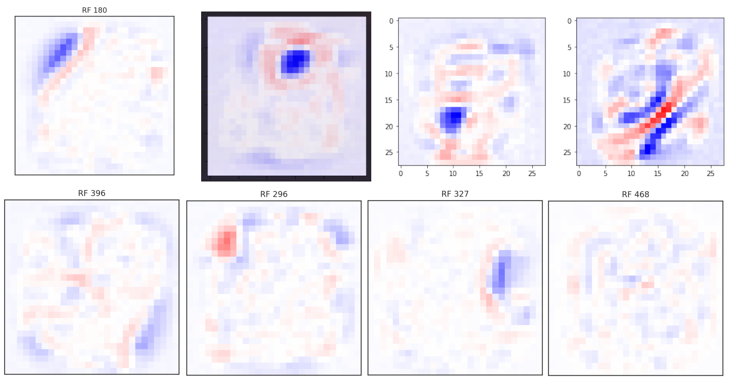


Figure 3: Receptive fields of a few hidden nodes of a 784 − 500 RBM

4.2 Towards deep networks - greedy layer-wise pretraining

For this part we turned our attention to a deep architecture called Deep Belief Networks (DBNs), essentially stacking RBMs together. We have implemented the greedy layer-wise pre-training, a method which uses unsupervised learning greedily, for each layer starting from the bottom, firstly using images, and then a trained layer's hidden representation as the visible units of the next layer RBM.

Firstly we made a 3-layer stack of two RBMs (784-500-500 architecture) and observed their respective reconstruction losses. We noticed similar behaviour in reconstruction loss convergence, with topmost RBM having a slightly higher error than the RBM below (Figure 4).

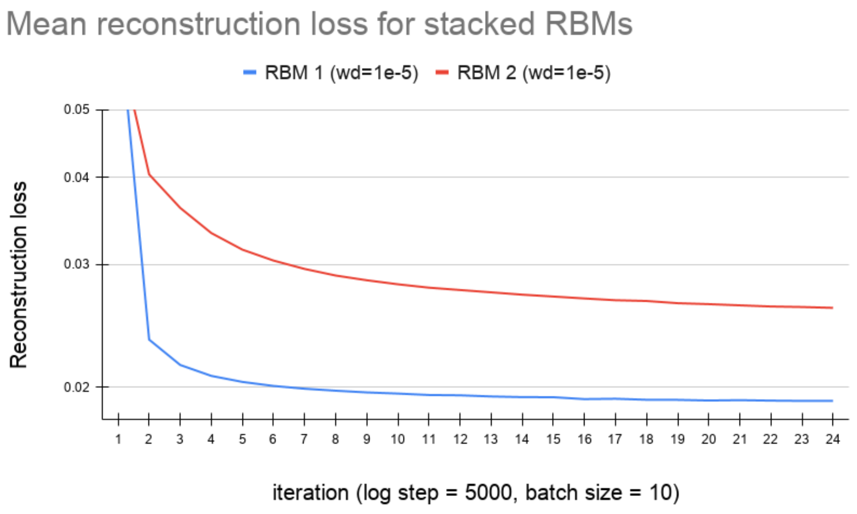


Figure 4: Mean reconstruction loss for stacked RBMs

Secondly, we added an additional RBM at the top of our previous network. The top RBM has 510 visible units (500 taken from the hidden layer of the RBM beneath it and 10 additional units used for digit labels) and 2000 hidden units. The top RBM is trained in the same way as the RBMs beneath with the exception of using training labels as our starting values of the 10 additional visible units. One-hot encoding is used for the conversion of a digit label to 10 units. This kind of network could now be considered a DBN and can be used for both image recognition and image generation purposes.

Observing the convergence and the recognition accuracy when classifying images, we can see the benefit of using around 15 Gibbs sampling iterations in the top layer in figure 5. In the network run shown in the figure, the accuracy achieved was 82.94%, but during various runs, with changes in the under-the-hood network architecture details like when to use probabilities instead of samples during greedy pre-training and how to make the upward network pass during picture recognition, we managed to achieve recognition accuracy results reaching 94.5% (Fig. 6).

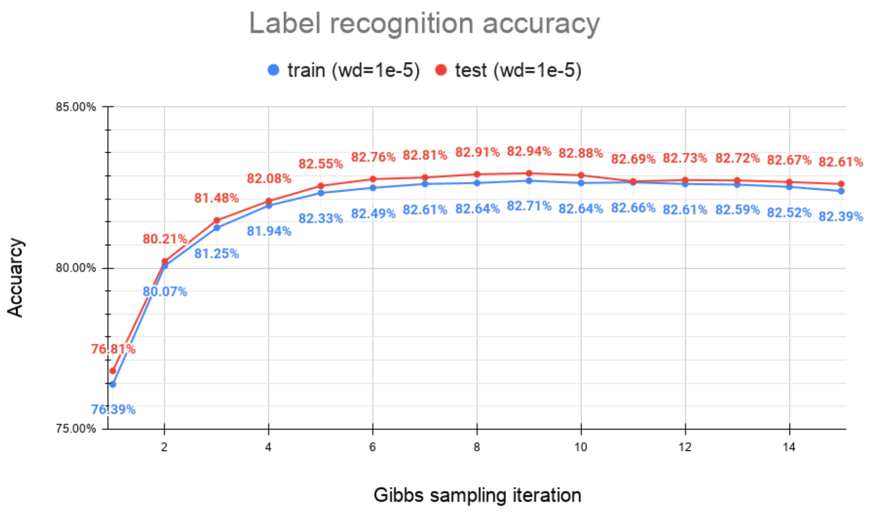


Figure 5: Label recognition accuracy

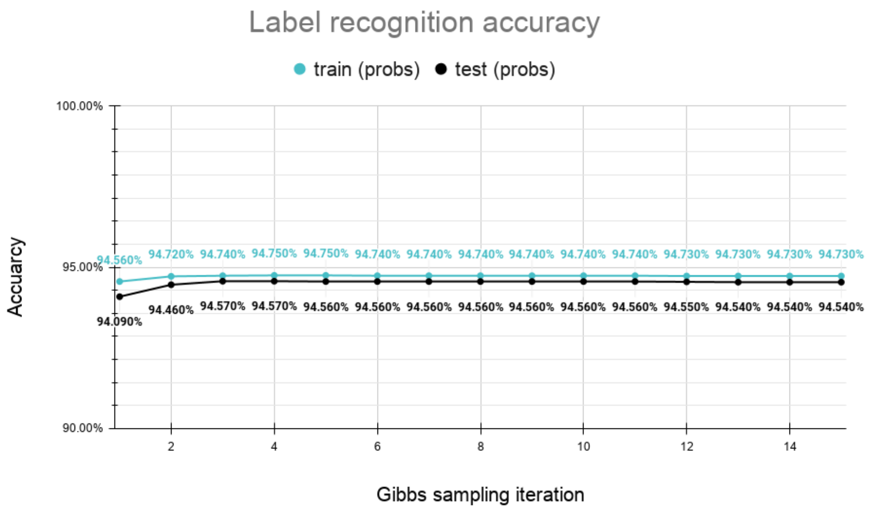


Figure 6: Label recognition accuracy is very high when passing probabilities from one RBM to the other and when using only probabilities during Gibbs sampling in topmost RBM for recognition classification. Having this modified network gave very bad results in picture generation

We also observed the behaviour of the probabilities in the 10 units responsible for digit classification in the visible layer of the topmost RBM. In figure 7 we show two examples of correct digit classification and one example of an incorrect classification.

Finally, we tested the generative abilities of our network. The network did not perform particularly well, but there were times when intuitive and expected patterns came up, such as those shown in Figure 8. These were the result of a couple of networks, trained with varying number of epochs, weight decay coefficients, and momentum coefficients. Better results could most certainly be acquired using supervised fine-tuning of the DBN.

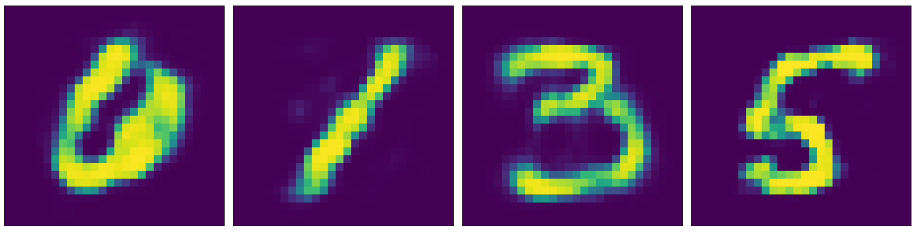
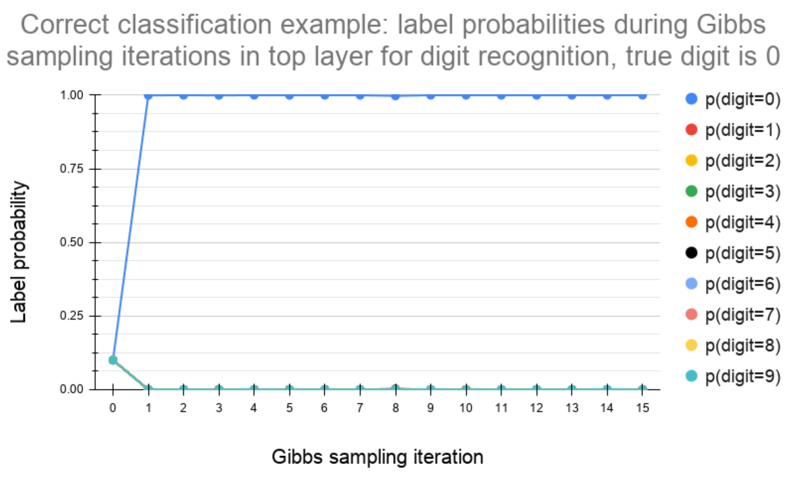
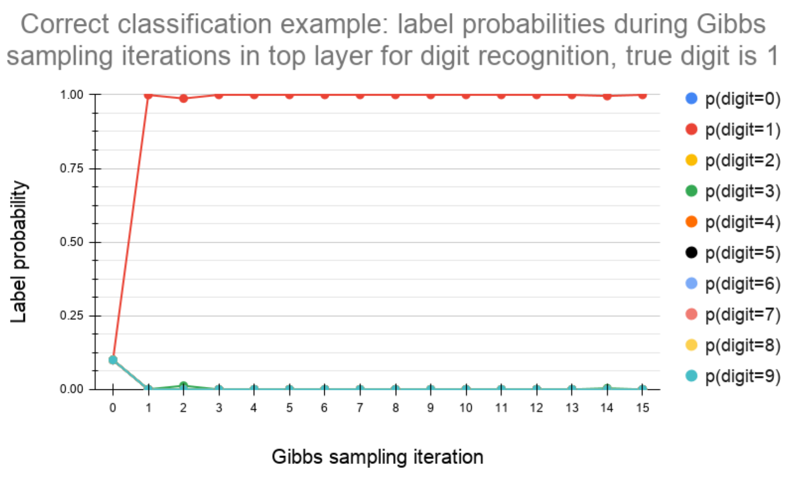


Figure 8: Mean reconstruction loss for stacked RBMs

5 Final remarks

The topic of the assignment was very interesting, but the instructions and materials could have been slightly clearer. It is easy to get lost in deciding when to use probabilities and when to use binary sampling; this could have been more clearly elaborated in terms of describing the intuition behind and letting the students know what exactly should they reproduce. This is important because the length of the training does not allow for many tests and it is hard to get intuition based on just a couple of tests. However, the exploratory part of the lab, even though perhaps too unguided, was interesting and advantageous in some learning aspects.





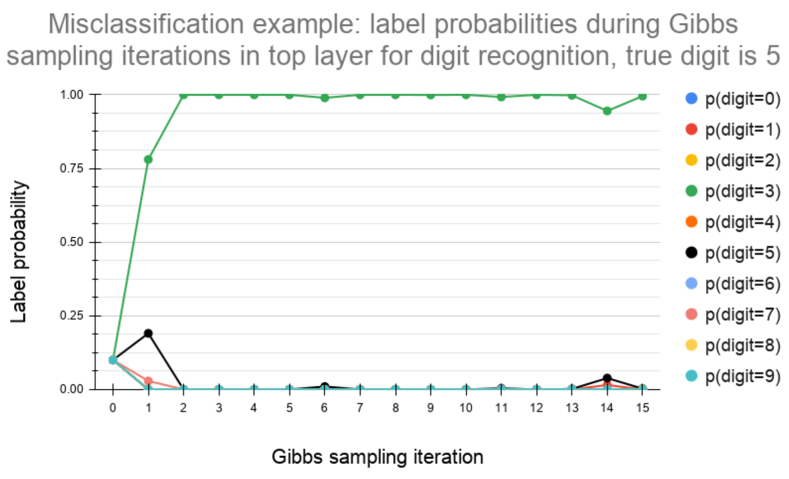


Figure 7: Examples of correct classification in first two pictures and of incorrect classification in the last picture.