Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

Restricted Boltzmann Machines and Deep Belief Nets

Alex Norlin, Athanasios Charisoudis and Giovanni Castellano

March 4, 2022

1 Main objectives and scope of the assignment

The main objective of this lab was to study restricted Boltzmann machines and Deep Belief Networks (DBN). In particular, we implemented RBMs and used them to reconstruct images from the MNIST dataset. Afterwards, we stacked RBMs to implement DBNs and performed image classification and conditional image generation tasks.

2 Methods

We decided to use the programming language Python to implement and test the two networks. Thanks to the library Numpy we could easily perform mathematical operations efficiently. Moreover, we used matplotlib to plot results. Our code was, of course, based on the provided Python classes and functions in order to train and evaluate RBMs and DBNs.

3 Results and discussion

In what follows results of training and evaluating the aforementioned networks are given. Initially, a single RBM with varying number of hidden nodes is trained and its performance and learned features are presented. Subsequently, we stack two (2) RBMs to form a simple DBN, which we train using greedy layer-wise (unsupervised) pre-training, i.e. each RBM is trained independently using the hidden activations of the previous RBM. Finally, we stack three (3) RBMs two levels of feature abstractions, in a more powerful DBN. This is then trained in the same manner and its performance is evaluated.

3.1 RBM for recognising MNIST images

To begin, we trained a RBM using contrastive divergence on the MNIST dataset. We were interested in the reconstruction capabilities of the network; in order

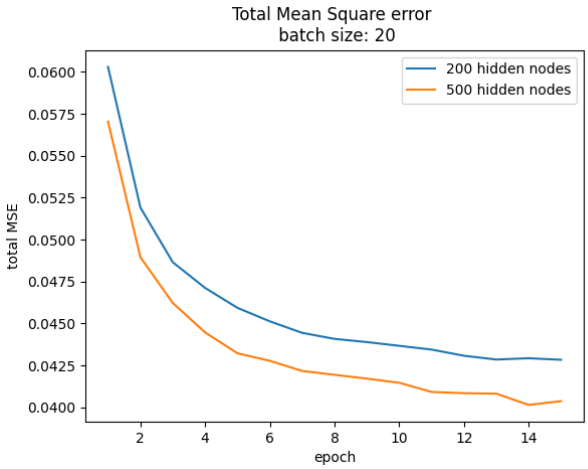

Figure 1: Training total MSE for different number of hidden nodes. Learning rate was set to 0.3 for both architectures.

to measure the similarity between the original data and the reconstructed data we used the total mean square error between the two matrices. Figure 1 illustrates how the mean square error changes during the training. In particular, we compared the performance of two different RBMs, the first one with 200 hidden nodes and the second one with 500 hidden nodes. As we can observe, we get better results using a higher number of hidden nodes. This basically means that the model with only 200 hidden nodes is not complex enough to learn the dataset.

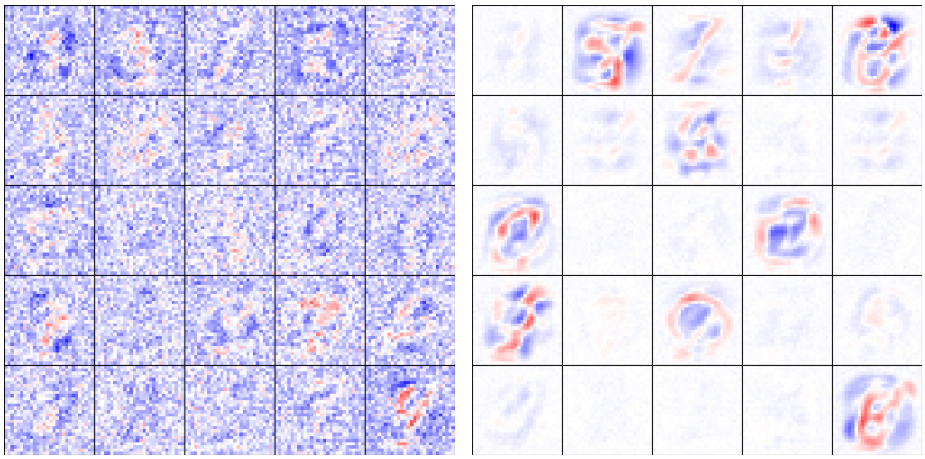

Figure 2: Receptive field of some hidden nodes after (a) 50 iterations (b) 3000 iterations

It is also interesting to plot the receptive field for some of the hidden nodes; we notice that as we proceed with the training such plots assume the shapes of digits. (Figure 2). Finally, in Figure 3, we give the reconstructions of 10 test

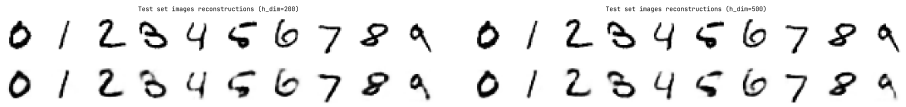

Figure 3: Reconstructions of 10 test set images from the RBM with 200 (left) and 500 (right) hidden nodes. Probabilities of visible layer were plotted rather than binary samples for better visualization.

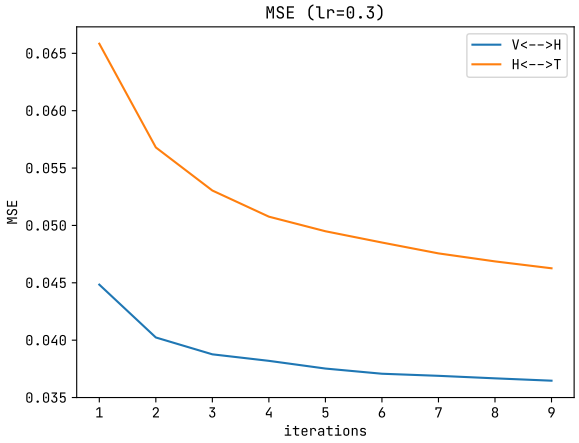

Figure 4: Training MSE for each RBM in the simple 2-hidden-layer DBN. Learning rate was set to 0.3 for both RBMs.

set images (one per training class) for number of hidden nodes in RBMs. As can be seen there, the RBM with 500 hidden nodes does a slightly better job at reconstructing the visible layer signal; it's like producing more sharp results. Test-set MSE was found to be **0.00396** (200 hidden nodes) and **0.00360** (500 hidden nodes) for the two RBMs respectively.

3.2 Towards deep networks - greedy layer-wise pretraining

In the first part of this section, we trained a 2-hidden-layer (i.e. comprising 2 RBMs) DBN with number of nodes from bottom (visible) to top (hidden) RBM being 784-500 and 500-500 respectively. The reconstruction losses during the (independent) layer-wise pre-training of each RBM are given in the Figure 4. As can be seen there, Visual-to-Hidden RBM exhibits lower training MSE than the Hidden-to-Hidden RBM. In Figure 5 (left) there are sample reconstructions of the test-set images from our simple 2-hidden-layer DBN, while the test MSE over the entire test set was found to be **0.00162**. As can be seen there and after comparing with reconstructions from single RBMs (Figure 3), our simple DBN manages to produce more detailed and rich reconstructions. Note: we treated the weights of the bottom RBM as directional while training the top one, whose weights are always undirectional.

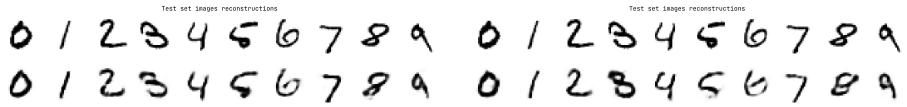

Figure 5: Reconstructions of 10 test set images from the 2-hidden-layer (left) and 3-hidden-layer + labels (right) DBN. Probabilities of visible layer were plotted rather than binary samples for better visualization.

Subsequently, we extend our simple 2-hidden-layer DBN by adding another RBM on top; thus having a 3-hidden-layer DBN with number of nodes being 784-500 (vis<–>hid RBM), 500-500 (hid<–>pen RBM), (500+10)-2000 (pen<–>top RBM) where in the visual layer of the top RBM we added 10 more nodes to account for one-hot-encoded label as a separate input. Via the latter addition, we enable **conditional generation** as well as image **classification** tasks to be facilitated with this DBN.

Image Recognition using 3-hidden-layer DBN

To begin with, in Figure 6 we provide the training MSEs for each of the RBMs that comprise the DBN, during the layer-wise pre-training with Contrastive Divergence. As can be seen the further we move from the actual visible layer of the DBN, the harder is for the RBM to train. Therefore, the top RBM exhibits x4 training MSE than the bottom one, while the middle RBM is almost at x1.5.

To perform image classification, we feed the bottom RBM with the image(s), get first hidden layer activation, feed it to the middle RBM, get its hidden layer activations, concatenate them with the initial label vector (0.1 for each label - uniform prior) and perform **10 Gibbs Sampling steps**. As a result, we have the predicted labels which we transform to 1-hot encoding and compare them with the corresponding 1-hot encodings of the true labels. Accuracy was **91.55%** for the training set and **91.50%** for the test set. In addition we plot the **average cross entropy** between the true and predicted labels **during those 10 Gibbs Sampling steps** in Figure 7, where the convergence of the labels during the recall steps is apparent.

(Conditional) Image Generation using 3-hidden-layer DBN

Firstly, we test the reconstructive performance of our 3-hidden-layer DBN. Reconstructions of test set images can be seen in Figure 5 (right), which exhibit higher expressitivity than those of the 2-hidden-layer DBN (left). Finally, in Figure 8, we provide conditional generation samples from our 3-hidden-layer DBN. As can be observed there, we got some digits like 0 and 9 in the correct place, and some other like 3 and 8 produced in different requested class labels. This should be attributed to the fact that we did not fine-tuned the entire DBN after pre-training. Note: we started from 1's in the visible nodes of the top RBM rather than the biases because it lead to slightly better results.

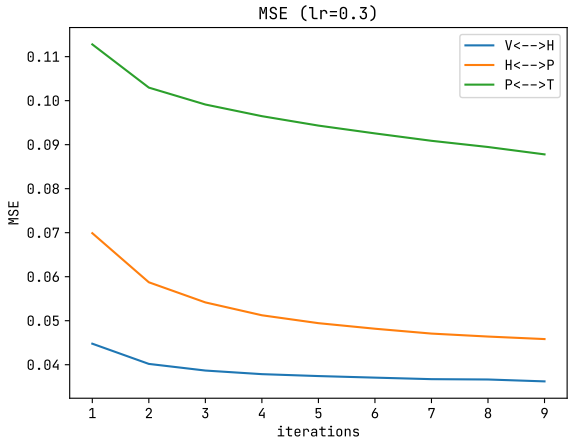

Figure 6: Training MSE for each RBM in our 3-hidden-layer DBN. Learning rate was set to 0.3 for top and RBMs and 0.2 for the middle one.

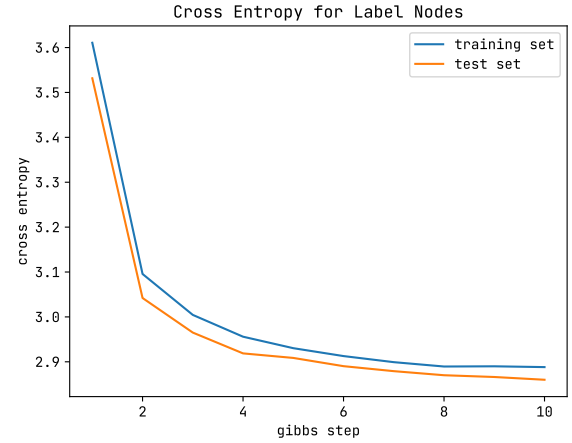

Figure 7: Cross entropy of the labels vs. true labels during Gibbs sampling steps of the recall process. The initial labels were given uniform probability and the resulting winners were more than 91% correct.

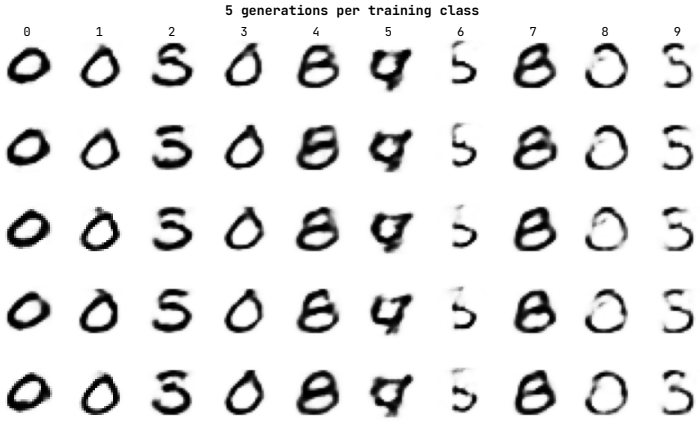

Figure 8: Conditional generation samples from our DBN trained with layer-wise pre-training only. Five (5) samples are given per training class.

4 Final remarks

This lab had some trouble some aspects, especially when it came to when we had to use probabilities vs. binary samples. Eventually, we stuck with the given instructions and suggestions which resulted in relatively good results. Overall, DBN have very good reconstruction capabilities however without fine-tuning after layer-wise pre-training our networks had very limited generative capabilities.