Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

Restricted Boltzmann Machines and Deep Belief Nets

QIU Danny, SAMAILLE Noe

October 8, 2019

1 Main objectives and scope of the assignment

- Restricted Boltzmann Machines: contrastive divergence and analysis on the MNIST dataset.
- Deep Belief Networks: Greedy layer-wise pre-training and supervised fine-tuning.

2 Methods

We used the framework proposed by the lab to implement the different trainings.

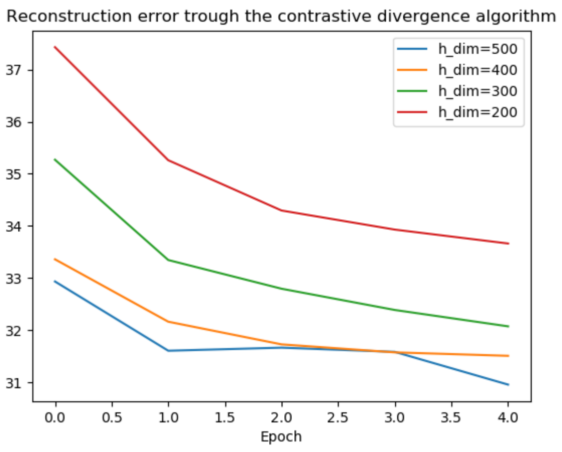
3 Results and discussion

3.1 RBM for recognising MNIST images

3.1.1 Convergence of the training

When training RBMs, we monitored the convergence and stability of this learning in the hidden units by taking a look at the average reconstruction loss.

Figure 1 shows the reconstruction loss through the contrastive divergence for RBMs with a number of hidden units varying from 500 to 200.



(a) Receptive fields initialized

Figure 1: Reconstruction loss for different numbers of hidden units

- The decreasing loss implies that the training converges and is stable.
- The performance of the RBM increases with the number of hidden units, meaning that the more units there is in the hidden layer, the more efficient the RBM is to learn the data distribution.

3.1.2 Units analysis

In order to understand how hidden units process the inputs, we trained an RBM over one iteration and studied the hidden units activation for a batch of ten images.

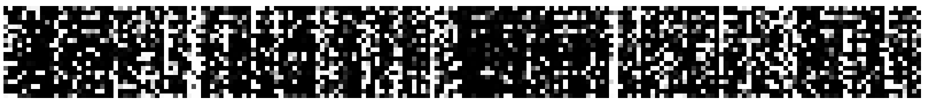
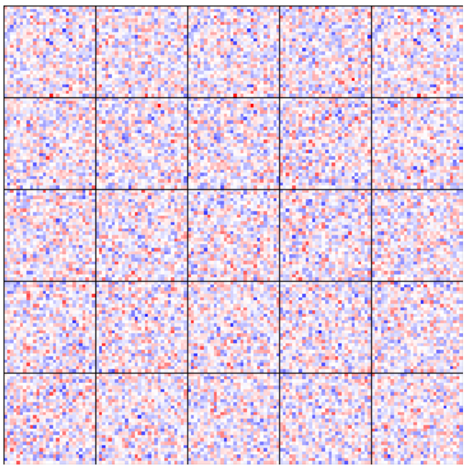


Figure 2: RBM hidden activation on a batch of MNIST test set. Each line represents the activations of the hidden units for one image of the batch. The batch is made of ten images.

Figure 2 shows that overall, hidden units activate on most of the images of the batch. A few vertical white lines shows units that almost never activate. This kind of activation pattern may show that units are specialized in recognizing some features.

To confirm the specialization of hidden units, we displayed the receptive fields of some hidden units.


(a) Receptive fields initialized

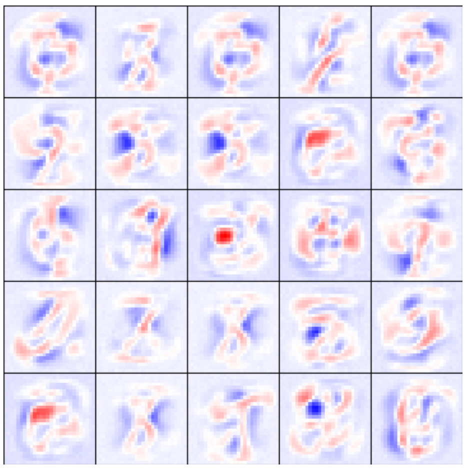

(b) Receptive fields after one iteration

Figure 3: Receptive fields of an RBM trained on one iteration

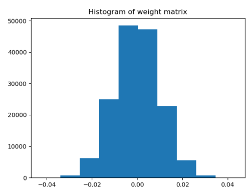
Figure 3 shows that after one epoch, the receptive field of hidden units quickly changes to look like components of handwritten digits. This confirms the specialization of the units in the recognition of certain patterns: numbers, curves, lines... Units do not isolate a single digit, but a combination of their features.

3.1.3 Weight analysis

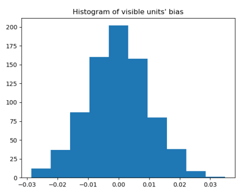
When taking a look at the histogram of weights of the RBM, (Figure 4), we see after two epochs an increase of weights with values close to zero and a few weights with higher values. RBM training has the same effect as regularization on the weights and biases.

3.1.4 Reconstruction

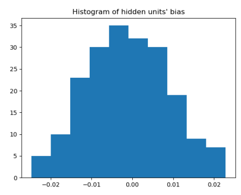
After one training, figure 5 shows the recall is quite accurate, the digit being a bit more noisy compared to the original images. RBM performs well at generalizing reconstruction.



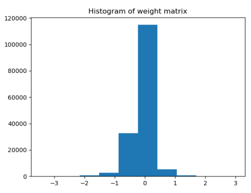
(a) Iteration 0



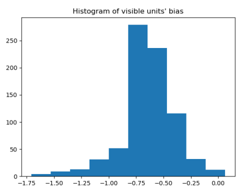
(b) Iteration 0



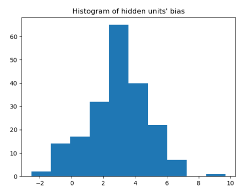
(c) Iteration 0



(d) Iteration 2



(e) Iteration 2



(f) Iteration 2

Figure 4: Histogram of weights and biases before training and after two epochs

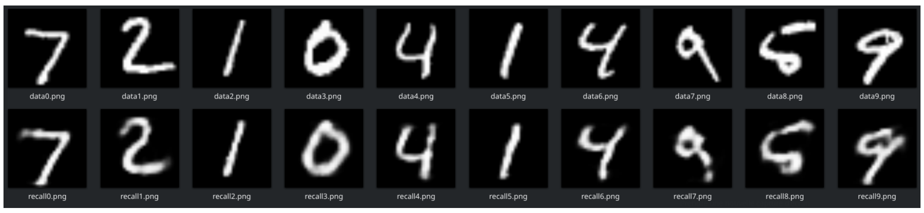


Figure 5: RBM recall of 10 handwritten digits from MNIST test set

3.2 Towards deep networks - greedy layer-wise pretraining

3.2.1 DBN with two RBMs

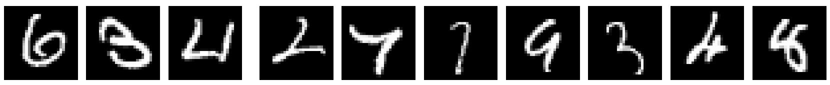
We compared the reconstruction loss of a single RBM with a DBN with two RBMs. The loss is summed over all the batches. On one training, we obtained a loss of 125994 for a DBN with two RBMs and of 100920 for one RBM.

Although the single RBM has a smaller error, there is not much difference when looking at the reconstructed images. We conclude a single RBM is more efficient in reconstruction and time.

3.2.2 DBN with three RBMs

We implemented a $784 - 500 - 500 - 2000$ DBN with three RBMs, where the top layer can also receive labels. The architecture is the one proposed in the lab code.

- Using batches of size 20 over 20 epochs, our best accuracy after pre-training is 76.90% on the test set.
- Using batches of size 100 over 20 epochs for each RBM, Our classification is good after pre-training with 86% accuracy.



(a) 0 (b) 8 (c) 0 (d) 6 (e) 4 (f) 1 (g) 4 (h) 2 (i) 6 (j) 4

Figure 6: Wrongly classified samples and there estimated labels

Figure 6 shows some of the samples that where misclassified by our Deep Belief Network after the greedy layer-wise pre-training for batch size 100. We can see that most of these samples are either ambiguous combination of numbers (e.g. 9 and 4) or simply unrecognizable ones, even for humans.

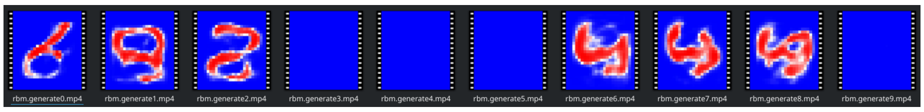


Figure 7: RBM generation of the 10 labels

For generation, the key factor that determine the quality of generated images is the RBM pre-training (batch size, epochs) and the number of iterations for Gibbs-Sampling. More iterations give less noisy generations.

Generated images do not look like the original images. The DBN does not seem to take into account the label. Sometimes, it generates a distinct number 3 or 8 (which does not match the input label), or an overlap of several digits. This can be explained because greedy training propagates the inputs upwards, and trains the weights to decrease the recognition error, but nothing is done for generation. Another reason could be the lack of accuracy.

3.3 Supervised fine-tuning of the DBN

When trained on 100 epochs with batches of size 100, the accuracy on the test set went from 76.60% to 85.14%.

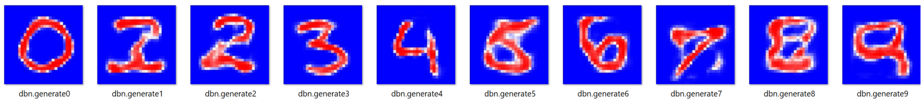


Figure 8: RBM generation of the 10 labels after fine-tuning

For generation, we can see a significant increase in the quality of generations (figure 8). Labels are taken into account for generation most of the time.

In terms of architecture, we compared the histogram of weights of the DBN before and after fine-tuning. We could see that weight matrix *h_to_v* and *v_to_h* had different distribution, showing they are different from the original undirected matrix. There is no evident change in weight distribution.

3.3.1 Comparison with one hidden layer removed

The simpler network has three layers (two RBMs) with the architecture $784, 500 + 10, 2000$.

To compare with the previous network, the training was made over 10 epochs of pre-training and fine-tuning for both, with a mini-batch of size 100.

On the test set, the simple network has an accuracy of 85.30% after pre-training and 56.75% after fine-tuning.

In comparison, the full network had 86% before and after fine-tuning.

The optimal architecture seem to be with three RBMs.

4 Final remarks

The RBM is an efficient network for reconstructing a given image, with reasonable computational time. The simple architecture allows us to easily look at the hidden units and their way of learning features.

DBN after a pre-training phase is capable of classifying images, the accuracy depending on the time spent on computation (for 86%, it took us approximately 30 minutes of training).

Fine-tuning was more difficult to implement, the steps of the wake-sleep algorithm were well-guided in the code but whether to use probabilities or activations was hard to understand from litterature. Time computation was also an obstacle to experimenting different configurations.