Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4
Restricted Boltzmann Machines and Deep Belief Nets

Benjelloun Hamza, Benchekroun Hamza and Ait lahmouch Nadir

February 20, 2021

1. Main objectives and scope of the assignment

- **Restricted Boltzmann Machines** : Implement RBMs and see the effect of some hyperparameters on their results.
- **Deep belief network** : implement DBN using unsupervised greedy pretraining of RBM layers and supervised fine-tuning.

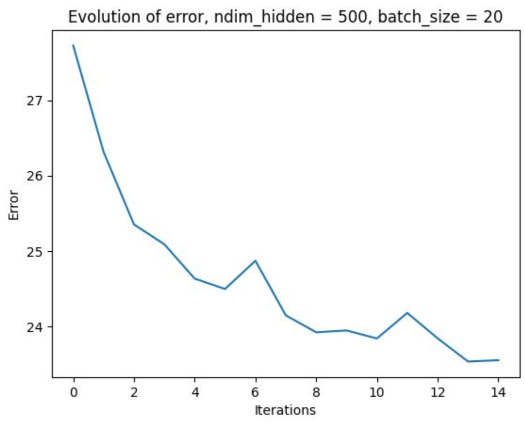
2. Methods

We use the provided skeleton for this assignment. The language used is Python, with the help of libraries such as struct and matplotlib. The code provided and developed gives a great flexibility and allows to experiment freely with the parameters.

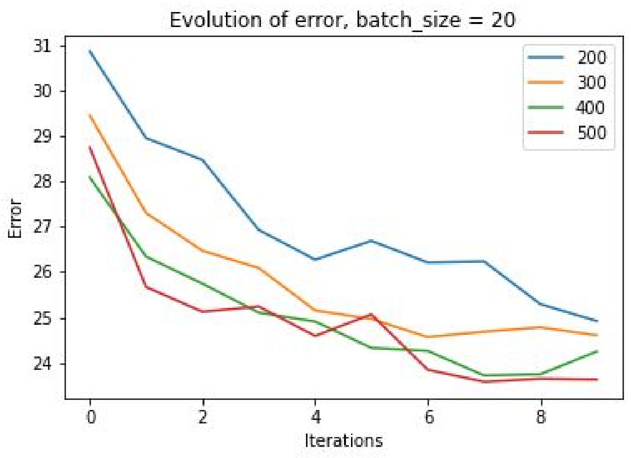
3. Tasks and questions

3.1 RBM for recognising MNIST images

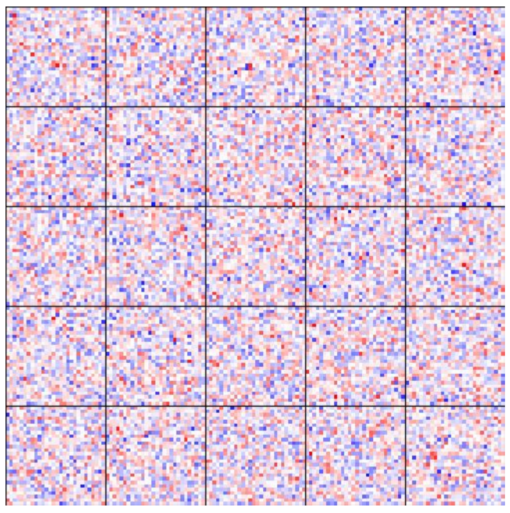
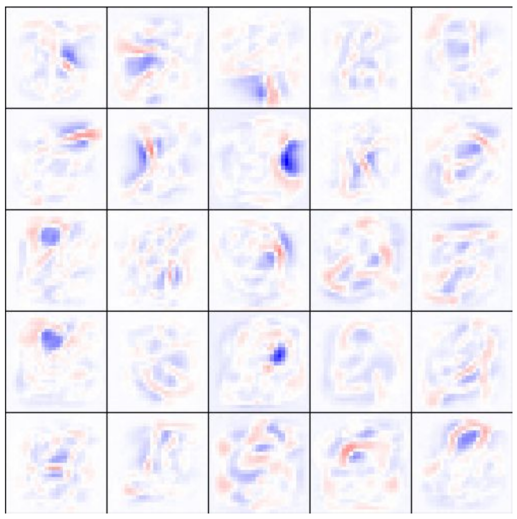
We start by verifying that the error (MSE) decreases steadily with the number of epochs, using the recommended 500 units in the hidden layer, a batch size of 20, and up to 15 epochs, we have :



To study the effect of the number of units in the hidden layer, we experiment with a number of units ranging from 200 to 500. We can safely conclude that a bigger number of hidden nodes yields better error using the same number of epochs, It is however more computationally expensive. We can also see that in this case, the nets with 400 and 500 hidden units are basically equivalent.



To investigate the receptive fields (weights), we plot these at each of 15 iterations using 500 hidden units:

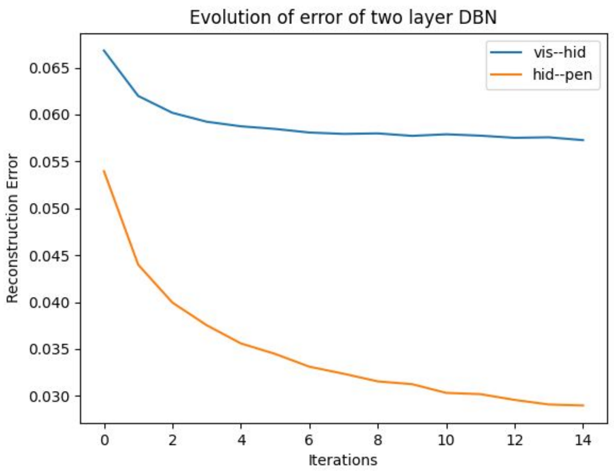
Visualization of the receptive fields before (left) and after (right) 15 iterations.

We can see that after one iteration only, the fields shape up to recognize the handwritten digits' characteristics: shapes, curves, etc...

3.2 Towards deep networks - greedy layer-wise pretraining

3.2.1 DBN with two RBMs

After building a DBN with two RBMs stacked one on top of another, we plotted the reconstruction loss of each one of them. Here's the result after 15 iterations with the configuration given:

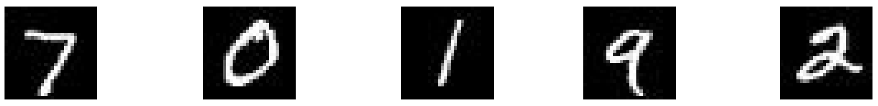


It seems like the top RBM is performing better than the bottom one.

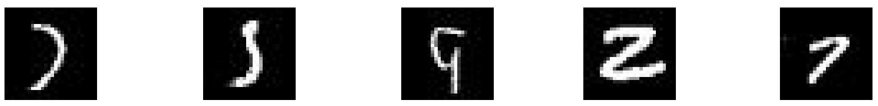
3.2.2 DBN for image recognition

For the purpose of image recognition, we implemented a 784 - 500 - 500 - 2000 which is three RBMs stacked one on top of another with the last one receiving labels as well. With a mini batch size of 20 and after 30 iterations, our best accuracy were:

- For training set: **87.06%**
- For test set: **86.84%**



Examples of samples rightly classified



Classified as: 9 3 4 0 9

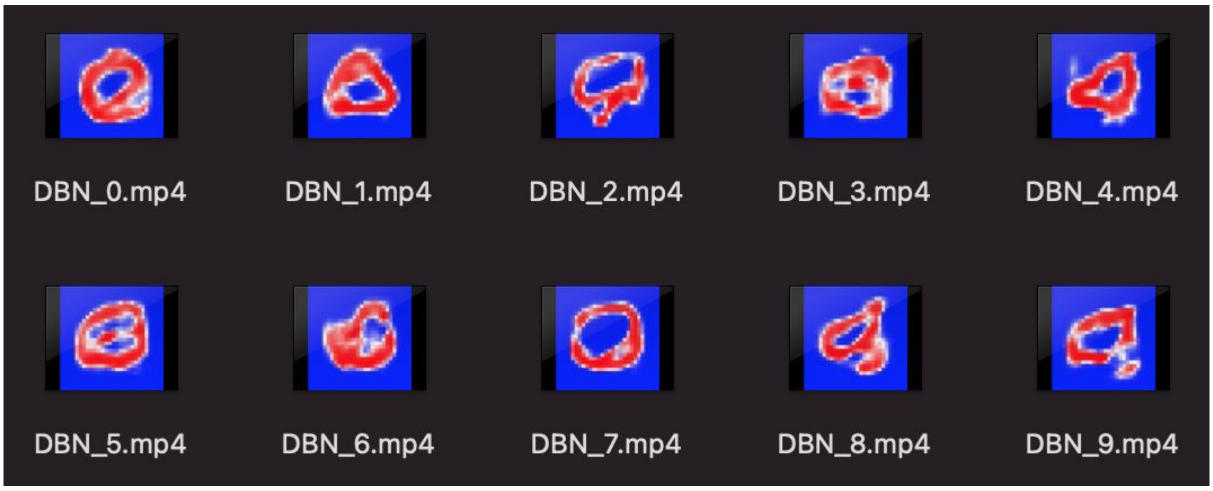
Examples of samples wrongly classified

The above images show some of the samples that were misclassified (second row) and rightly classified (top row). We can see that unlike the good classified ones, the misclassified samples are ambiguous and sometimes unrecognizable even for humans.

Note: Training with only 20 iterations as asked in the lecture yielded 76% of accuracy on the test set.

3.2.3 DBN for image generation

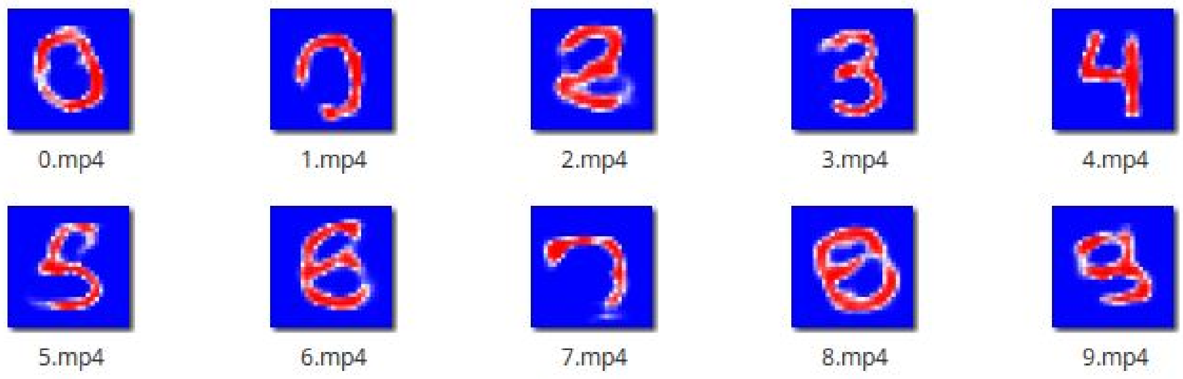
Using the same networks and same hyperparameters of the section above, we try to generate the digits. We noticed that the generated images do not look like the targeted digits. The network doesn't really seem to take into account the given label. The reason for that would be that the DBN was not well trained, because the RBM pre-training and the number of iterations for Gibbs-Sampling are what determine the quality of the generated images. Also, the weights were updated to decrease the recognition error but nothing was really done for generation.



The generated images

3.3 Fine-tuning deep networks

We trained our DBN with a wake-sleep algorithm using 30 epochs and a batch of size 100.



As we see in the above picture. The quality of generation looks better than the previous part. Decoupling the weights of recognition and generation helped to create a model which is both discriminative and a generative.And by comparing the directed weights matrices it seems that they had different distributions.

4. Final remarks

RBM is an efficient way to reconstruct images in a reasonable time.

DBM with unsupervised greedy pretrained RBM layers is good for image classification but it takes a lot of time for training (more than 40 min gives an accuracy around 85%)

DBM with wake-sleep training gives a good accuracy for image classification and a good image reconstruction. It is hard to implement and takes a lot of time in training, therefore it's difficult to experiment with a lot of configurations.