Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

Restricted Boltzmann Machines and Deep Belief Nets

Tristan Perrot, Romain Darous and Mathis Pernin

March, 2024

Please be aware of the constraints for this document. The main intention here is that you learn how to select and organise the most relevant information into a concise and coherent report. The upper limit for the number of pages is 6 with fonts and margins comparable to those in this template and no appendices are allowed.
These short reports should be submitted to Canvas by the authors as a team before the lab presentation is made. To claim bonus points the authors should uploaded their short report a day before the bonus point deadline. The report can serve as a support for your lab presentation, though you may put emphasis on different aspects in your oral demonstration in the lab. Below you find some extra instructions in italics. Please remove them and use normal font for your text.

1 Main objectives and scope of the assignment

List here a concise list of your major intended goals, what you planned to do and what you wanted to learn/what problems you were set to address or investigate, e.g.
Our major goals in the assignment were to :

- Explain key ideas underlying the learning process of RBMs,
- Apply basic algorithms for unsupervised greedy pre-training of RBM layers and supervised fine-tuning of the resultant DBN,
- Design multi-layer neural network architectures based on RBM layers for classification problems,
- Study the functionality of DBNs including generative aspects

2 Methods

Mention here in just a couple of sentences what tools you have used, e.g. programming/scripting environment, toolboxes. If you use some unconventional method or introduce a clearly different performance measure, you can briefly mention or define it here.

In this lab, we based our model on Python with the some code that was given to us. We used the **MNIST** dataset which is a well known dataset for image-based machine learning. We also use **matplotlib.pyplot** to plot some graphs.

3 Tasks and Questions - Restricted Boltzmann Machines and Deep Belief Nets

*Make effort to be **concise and to the point** in your story of what you have done, what you have observed and demonstrated, and in your responses to specific questions in the assignment. You should skip less important details and explanations. In addition, you are requested to add a **discussion** about your interpretations/predictions or other thoughts concerned with specific tasks in the assignment. This can boil down to just a few bullet points or a couple of sentences for each section of your results. Overall, structure each Results section as you like. Analogously, feel free to group and combine answers to the questions, even between different experiments, e.g. with noise-free and noisy function approximation, if it makes your story easier to convey.*

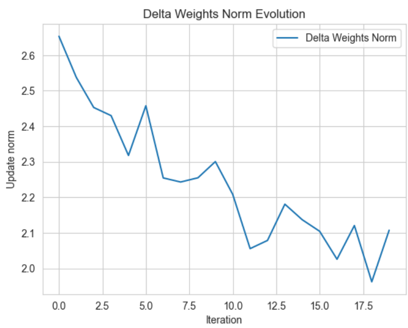
*Plan your sections and consider making combined figures with subplots rather than a set of separate figures. **Figures** have to condense information, e.g. there is no point showing a separate plot for generated data and then for a decision boundary, this information can be contained in a single plot. Always carefully describe the axes, legends and add meaningful captions. Keep in mind that figures serve as a support for your description of the key findings (it is like storytelling but in technical format and academic style.*

*Similarly, use **tables** to group relevant results for easier communication but focus on key aspects, do not overdo it. All figures and tables attached in your report must be accompanied by captions and referred to in the text, e.g. "in Fig.X or Table Y one can see ....".*

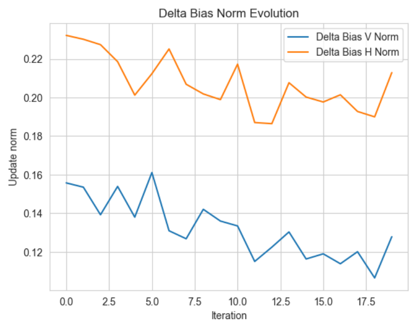
*When you report quantities such as errors or other performance measures, round numbers to a reasonable number of decimal digits (usually 2 or 3 max). Apart from the estimated mean values, obtained as a result of averaging over multiple simulations, always include also **the second*** *__moment__, e.g. standard deviation (S.D.). The same applies to some selected plots where __error bars__ would provide valuable information, especially where conclusive comparisons are drawn.*

3.1 RBM for recognising MNIST images

After initializing the weight matrix (including hidden and visible biases) with small random values. We implemented the Contrastive Divergence k=1 algorithm with **20 epochs** and a **batch size of 20** and a **learning rate of 0.01**. To monitor the convergence stability, we plotted the norm of every delta.

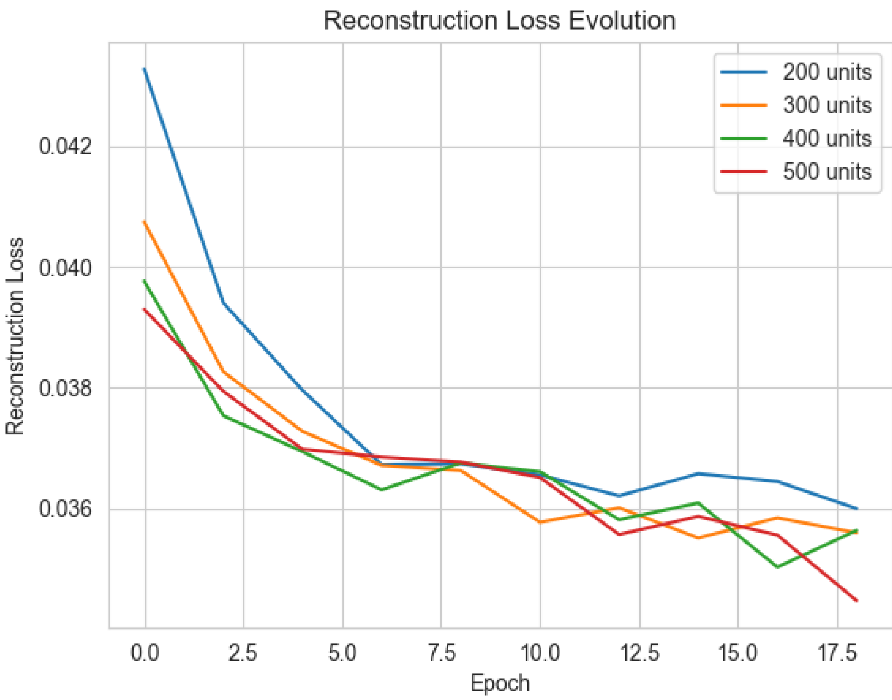


(a) Delta Weight Norm vs Iterations



(b) Delta Bias Norm vs Iterations

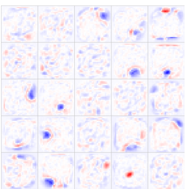
Here we can see that there is the trend that the update norm is decreasing even though there is still some noise to the decreasing. Then, we computed the average reconstruction loss for different hidden units numbers.



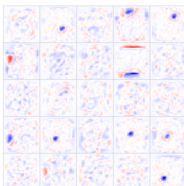
Figur 2: RBM losses evolution vs number of hidden units

We can see here that the more there is hidden units, the less the loss is but adding more hidden units add time for the calculus that are sometime not worth compared to the gain of loss.

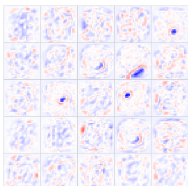
Then, we wanted to analyze the receptive fields and therefore we plotted the weights to the visible layer for the hidden units of interest.



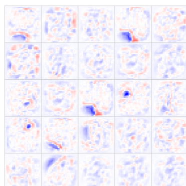
(a) 200 hidden units



(b) 300 hidden units



(c) 400 hidden units



(d) 500 hidden units

Figur 3: RBM weights

Here, what is important to notice is that the more hidden units we have, the less active zone we get. That's easily explained by the fact that the more hidden units we have the more we can spread the knowledge and therefore have a sparser weight matrix.

Figur 4: Reconstruction of some images

Also, we tried to plot for some images and see reconstructed images. Here, as expected we get a better reconstruction with less noise with more hidden units but it's not necessary to have to much hidden units because, for example here, with **400 hidden units** we with great result, great loss (cf 2) and less computational costs.

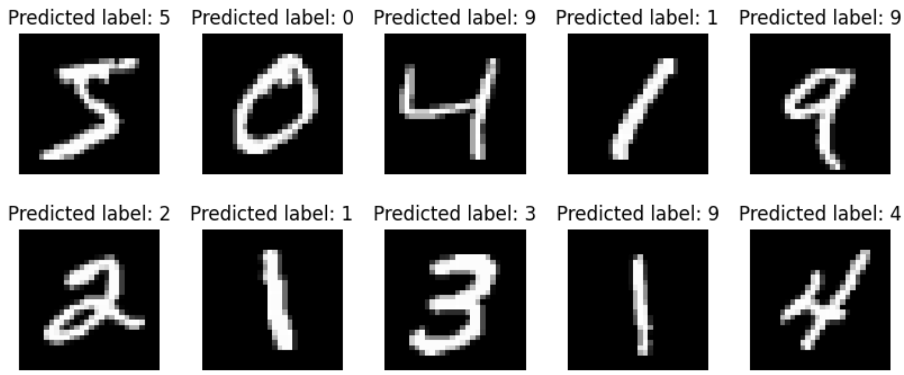
3.2 Towards deep networks - greedy layer-wise pre-training

3.2.1 DBN with two RBMs

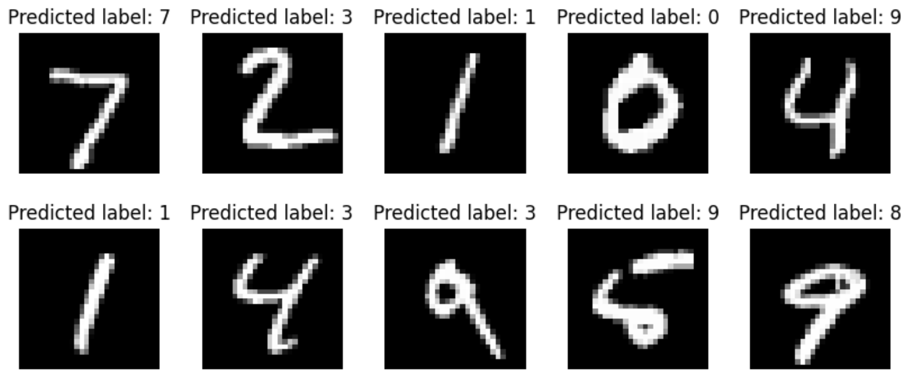
We started by training a DBN with two RBMs with the same amount of hidden units (500). We observed that, when training the two RBMs with the same number iterations, the reconsctruction loss was higher after adding the second RBM (0.904) than when training only one RBM (0.342). The running time is also significantly higher, which makes a single RBM better at the reconstruction task.

3.2.2 Classifying a DBN with three RBMs

Now, we add one last RBM that will be used for the generation / classification task. First, we trained the three RBMs with the same number of iterations. We get a classification accuracy of 69.58% for the training images and 70.26% for the test images. Even though the scores are not extraordinary high, the classification tasks shows rather good results and the accuracy doesn't drop when testing on unseen data. Here is an sample of the classification results for both train sets and test sets :



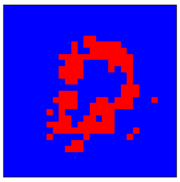
Figur 5: Classification of training images



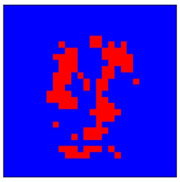
Figur 6: Classification of test images

3.2.3 Generating images a DBN with three RBMs

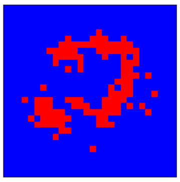
We also tried to generate images with the model, by imposing a label to the visible layer of the last RBM. However, the results are a bit disappointing :



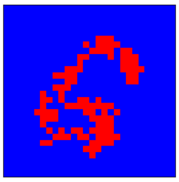
(a) 0



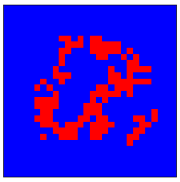
(b) 1



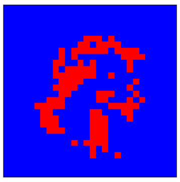
(c) 2



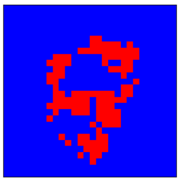
(d) 3



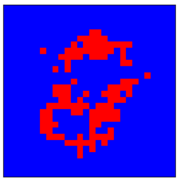
(e) 4



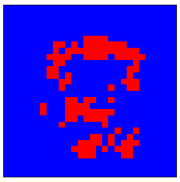
(f) 5



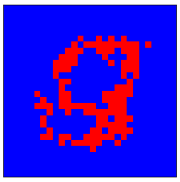
(g) 6



(h) 7



(i) 8



(j) 9

We can recognize guess the numbers 0, 2 and 9, and the generation of the label 3 leads to something similar to a 6.
However, the generation is not consistent enough. We still could identify some important parameters that have a role on the generation precision :

- The pre-training of the RBMs (iteration number, size of the epochs, learning rate),
- The number of iterations in the Gibbs Sampling step.

Thus, it seems like per-training is a good way to get a quite good classifier, but fails giving consistent result for image generation.

4 Final remarks *(max 0.5 page)*

In the first part of this lab, we implemented the learning algorithm for Restricted Boltzmann Machines. We studied the stability of the updates deltas and we have also compared the values of the loss compared to the number of hidden units. We have seen that there is a gap where it's no more useful to add hidden units to our RBM. Moreover, we compared the weights distribution for different hidden units number and see that the more hidden units there is, the sparser the weights are. At the end, we tried to reconstruct some of images and we have seen that there is also a gap in reconstruction and it is not useful to add too much hidden units.

In the second part of the lab, we tried to implement a classifier and a digit generator using three RBMs. Without fine-tuning, our model did well concerning classification task, but failed in the generation part. Still, it was very interesting to understand how RBMs and Gibbs sampling were use to perform those tasks, which is a different approach from usual MLPs for classification, for instance.