Unsupervised and reinforcement learning in neural networks

1st mini project

Lecturer: Marc-Oliver Gewaltig

Assistants: Berat Denizdurduran
Ihor Kuras
Sharbatanu Chatterjee
Carl Henrik Rolf Aslund

1 Kohonen maps and Hebbian Learning

Briefly explain following questions;

- How many layers are there in a Kohonen Map and what do they do?
- The self-organising process have 4 components; Initialization, Competition, Co-operation and Adaptation. How are all these components implemented in the Kohonen Map?
- Describe how a Kohonen Map performs dimensionality reduction.
- Kohonen maps are also described as Topographic maps, what is meant by Topographic maps and give an example of their possible presence in the human brain.
- Which one of the Hebbian type learning rule corresponds to the weight update in the Kohonen maps (Reminder of the examples of Hebbian type rules: Have a look at the Lecture 2, slide number 9). Show how can you obtain the learning rule of the Kohonen map from the corresponding Hebbian rule.

2 Kohonen maps on hand-written digits

Your task is to implement the Kohonen algorithm (or self-organizing map) and apply it to a data set of hand-written digits. The advantage of the data set is that it is easy to visualize, so that you can easily control the performance of the algorithm by visual inspection.

- Download the data (`miniproject1_data.zip`) from the moodle website. Unzip the archive which contains the data (`data.txt`) and the labels (`labels.txt`) that tell you which digit each of the data vectors represents. The data are provided in vectors of length 28x28=784. You can reshape them to 28x28 matrices for visualization. The data are taken from the MNIST database, a standard benchmark data set in pattern recognition.

  Use the provided MATLAB or Python script `name2digit` that takes a string of your name (or that of your partner) and returns a set of four digits. Use only the data corresponding to these four digits for the following analysis.

- Start with a Kohonen network of 6x6 neurons that are arranged on a square grid with unit distance and use a Gaussian neighborhood function with (constant) standard deviation $\sigma = 3$. Implement the Kohonen algorithm and apply it to the data in `data.txt`. Choose a small (constant) learning rate and report how you decide when your algorithm has converged.

- Visualize your prototypes. Describe your results.
- Using the information in (`labels.txt`), find a way to automatically assign one digit to each prototype that is represented best by it. Visualize and interpret your results.
- Explore different sizes of the Kohonen map (try at least 3 different sizes, not less than 36 units). Explore different widths of the neighborhood function (try at least $\sigma = 1$, 3, and 5). Describe the role of the width of the neighborhood function. Does the optimal width depend on the size of the Kohonen map?
- Until now, the width of the neighborhood function has been constant. Now, start with a large σ and decrease it over the runtime of the algorithm. Does it improve your result?

3 Report

Your report must be handed in by **Friday, May 4th 2018, at 23:55**. You can still upload your report after the deadline but be aware that **the late submissions will be penalized** proportional to the time after the deadline. You may work in teams of two persons but you should both upload your report **(As a team, you should write a single report together but each member of the team must upload that report individually)**. Submission takes place via the "moodle" web page. You should upload a zip file (**named after your last name, e.g.** `yourlastname.zip`) containing a PDF of the report and the source code of the programs. The source code itself is not part of the written report. The report **must not exceed 6 pages for single column or 4 pages for double column** (otherwise penalized proportionally) and it should contain:

1. your answers for the theoretical questions,
2. your choice of the learning rate and a description how you determine convergence,
3. visualization and description of the learnt prototypes,
4. a description and visualization of your method of assigning the digit that is represented by a prototype,
5. your results of the exploration of different network sizes and widths of the neighborhood function with a discussion of the results, and
6. your results and discussion of varying the width of the neighborhood function over time.

Feel free to add any comments or additional observations that you had while working on the mini-project.
You may use the functions provided (MATLAB and Python) for your implementation. All programming languages are accepted but necessary supports will be provided only for MATLAB and Python implementations.
Note that for this mini-project and the following one, after the corresponding deadline, there will be a 5-minute session per student during which you will very briefly explain your own source code and implementation of the mini-project. **This session will take place during one of the exercise sessions and is mandatory.**