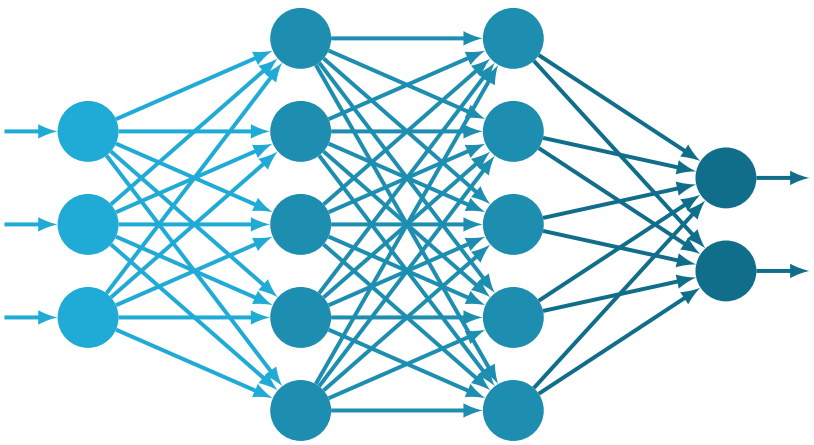
KU LEUVEN

Artificial Neural Networks & Deep Learning

Exercise Session 1

Prof. Dr. ir. Johan A. K. Suykens

Assistants
Sonny Achten
Bram De Cooman
David Winant
Xinjie Zeng



2023-2024

SESSION 1

Supervised Learning and Generalization

Note: There is no need to repeat the information provided during the guided session in your report. Answer the questions as accurate as possible. Use graphical and/or tabular presentation. The *page limit* for this exercise is *4 printed pages*, including figures. You will need the notebook `session1.ipynb` that you can find on toledo to answer the questions, or you can use the colab link. The first two parts are introductory and do not have to be included in your report. Your report should be based on section 1.3 and section 1.4 only.

This assignment will cover the use of neural networks as general models for the tasks of regression and classification. From their simplest form, the perceptron, to more complex architectures, you will explore how the networks are trained and the relationship between training and generalization. For all exercises and even if not asked explicitly, you should always pay attention to how fast the computations are, to the generalization capabilities of the model; this will help you to understand the motivations, consequences and limitations of each model or learning scheme. As a general tip: always try to get an insight of the bigger picture.

1.1 INTRODUCTION: THE PERCEPTRON AND BEYOND

This is an introductory section. You will not have to include this in your report.

1. In the case of a linear regression, the relation between an d-dimensional input and a univariate output datapoint is modelled by a linear relation of the form $y_i = w_1 x_{1,i} + w_2 x_{2,i} + \ldots + w_d x_{d,i} + \beta$. The parameters $\{w_j\}_{j=1}^{d}$ and β are found by minimizing the sum of squared errors

$$\sum_{i=1}^{N} \left(y_i - (w_1 x_{1,i} + w_2 x_{2,i} + \ldots + w_d x_{d,i} + \beta)\right)^2 .$$

   How can you perform a linear regression with a perceptron? Describe the link between those two models and the role of the activation function.

2. Consider a dataset $\{(x_i, y_i)\}_{i=1}^{N}$ where $\{x_i\}_{i=1}^{N}$ and $\{y_i\}_{i=1}^{N}$ are respectively the input and output set. Generate one corresponding to $y_i = -\sin(0.8 x_i)$.

   Plot the function. Will a linear model adequately capture the relationship between the input and output data? Can you relate this to over– or underfitting?

3. Train the perceptron using gradient descent. Decrease the learning rate value by 10, then 100. What do you observe ? What happens when the learning rate is set too high ? Remember to reset the weights of the network while doing your experiments.

4. Train now a neural network with one hidden layer on the data-set of the previous question. Does an epoch always lead to a decrease in loss function value with the proposed option `batch_size= 25` ? Why ?

1.2 INTRODUCTION: BACKPROPAGATION IN FEEDFORWARD MULTI-LAYER NETWORKS

This is an introductory section. You will not have to include this in your report.

Consider the neural network at Figure 1.1. Each hidden neuron uses a sigmoid activation function $\sigma(t) = 1/\left(1+\exp(-t)\right)$. The loss is the mean square error and the datapoints are $\left\{(x_1, x_2, y)_i\right\}_{i=1,2,3} = \{(1,1,-1),(0,-2,1),(-1,1,1)\}$. Input units are green, biases are yellow, hidden units blue and the output unit red. The weights can be found on the corresponding arrows. Compute the gradient through backpropagation analytically and don't forget to give the corresponding development. After one step of (simple, computed on the batch) gradient descent with learning rate 1, what are the new weights ? No need of a computer here, all can be done with pen and paper.

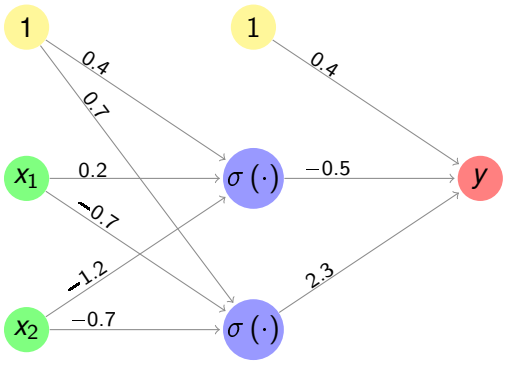


Figure 1.1: Neural network to perform the back-propagation on.

Hints. $\mathrm{d}(\sigma(t))/\mathrm{d}t = \sigma(t)(1-\sigma(t))$. Verifying that the new loss has indeed decreased is also a good hint.

1.3 IN THE WIDE JUNGLE OF THE TRAINING ALGORITHMS

Provide your answers to the questions in this section in your report.

Your goal in this exercise is to better understand the specificities, advantages and drawbacks of the many gradient-based optimization algorithms available to train neural networks. An implementation in `pytorch` is provided for you to play with.

A small model for a small dataset

We consider the surface given by the function

$$f(x,y) = \frac{\sin\left(0.8 * (x^2+y^2)\right)}{(x^2+y^2)^{0.9}}$$

The dataset consists of points sampled on this surface with or without noise, from a regular grid on the 2-dimensional plane. To answer the following questions, you can exploit the code provided in the notebook, and modify it to suit your needs. You will need to generate plots to support your findings.

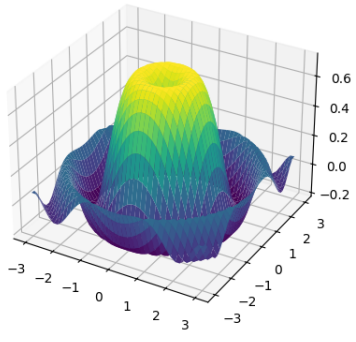


Figure 1.2: Visualization of the surface dataset

1. What is the impact of the noise (parameter `noise` in the notebook) with respect to the optimization process ?

2. How does (vanilla) gradient descent compare with respect to its stochastic and accelerated versions ?
3. How does the size of the network impact the choice of the optimizer ?
4. Discuss the difference between epochs and time to assess the speed of the algorithms. What can it mean to converge fast ?

A bigger model

We now consider the task of handwritten digits classification using the dataset MNIST, available in the `keras` package. We consider a convolutional neural network architecture.

5. How many parameters does the model have ?
6. Replace the ADAM optimizer by a SGD one. Can you still achieve excellent performances ? Try then the Adadelta optimizer. What is its particularity ?

1.4 A PERSONAL REGRESSION EXERCISE

Provide your answers to the questions in this section in your report.

In this exercise, the objective is to approximate a nonlinear function using a feedforward artificial neural network. The nonlinear function is unknown, but you are given a set of 13 600 datapoints uniformly sampled from it.
Everybody has to use a different dataset based on his or her student number. It will have to be built from 5 existing nonlinear functions. You can use the function `dataset_generation` from the notebook to build it from the `data.csv` file at your disposal.
The two first columns contain the input variables (in the domain $[0, 1] \times [0, 1]$). The five other columns, that we denote as T_1, T_2, T_3, T_4, T_5 are the 5 independent nonlinear functions evaluated at the corresponding points. The datapoints are noise free, i. e. the evaluation is exact. To construct your personal and unique data set, you have to build a new target T_{new}, which represents an individual nonlinear function to be approximated by your neural network. For this, consider the largest 5 digits of your student number in descending order, represented by d1, d2, d3, d4, d5 (with d1 the largest digit). T_{new} is built as follows:

$$T_{\text{new}} = \frac{\sum_{i=1}^{5} d_i T_i}{\sum_{i=1}^{5} d_i}.$$

1. Define your training and testing dataset using respectively 2000 and 1000 samples drawn independently. Explain the point of having different datasets for training and testing. Plot the surface associated to the training set.
2. Build and train your feedforward neural network. To that end, you must perform an adequate model selection on the training set. Investigate carefully the architecture of your model: number of layers, number of neurons, learning algorithm and transfer function. How do you validate your model ?
3. Evaluate the performance of your selected network on the test set. Plot the surface of the test set and the approximation given by the network. Explain why you cannot train further. Give the final MSE on the test set.
4. Describe the regularization strategy that you used to avoid overfitting. What other strategy can you think of ?