10907 Pattern Recognition

Lecturers
Prof. Dr. Thomas Vetter ⟨thomas.vetter@unibas.ch⟩

Assistants
Dr. Adam Kortylewski ⟨adam.kortylewski@unibas.ch⟩
Dennis Madsen ⟨dennis.madsen@unibas.ch⟩
Dana Rahbani ⟨dana.rahbani@unibas.ch⟩

Exercise 6 — Neural Networks

Introduction	28.11
Questions	03.12
Deadline	**11.12** Upload code to Courses.
	10.12+11.12 Group presentations, U1.001, see schedule online

In this exercise, you will compute by hand one iteration of the backpropagation algorithm. In addition, you will implement the backpropagation algorithm in pyTorch. Finally, you will study a neural network at a simple classification task.
You can download the data needed for this exercise and a code skeleton from the following repository: `https://bitbucket.org/gravis-unibas/pattern-recognition-2018`.
A number of helper functions are provided - your task is to implement the changes to the code as explained in Sections 1 & 2.

Remember to upload your code to courses in a ZIP file. Do NOT include the data folder. Only the python files you edited should be included + a file containing the group members names. Only 1 person from the team should upload!

Data:

Each `npy`-file contains the training and testing sets for a problem. Use it for developing the algorithms and studying the behaviour of your neural networks. The files are named `(train|test)_(inputs|targets)`.

Backpropagation (1 iteration - by hand)

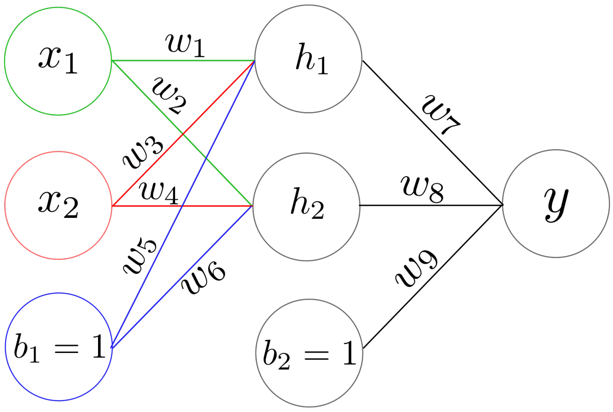

Figure 1: Fully connected Neural Network with three layers.

Update the parameters of the neural network in Figure 1 with the backpropagation algorithm based on one data point $X = [1, 1]$ with label $y = 1$. Consider the following network parameters for your calculations:

- Weights: $w_1 = 0.5$, $w_2 = 0.3$, $w_3 = 0.3$, $w_4 = 0.1$,$w_5 = 0.8$, $w_6 = 0.3$,$w_7 = 0.5$,$w_8 = 0.9$, $w_9 = 0.2$

- The hidden neurons use a sigmoid activation function:

$$a(t) = \frac{1}{1 + \exp^{-t}}$$

- The output neuron is computed with a simple linear activation function.

(a) **Forward Pass**
Calculate the values of the hidden neurons h_1, h_2 and the output neuron y and evaluate the error E of the prediction result. Use the mean squared error.

(b) **Backward Pass and Parameter Update**
Calculate the partial derivative of the prediction error E w.r.t. each weight $\frac{\partial E}{\partial w_i}$ and update the weights via: $w_i^* = w_i - \eta \frac{\partial E}{\partial w_i}$ with $\eta = 0.2$. Use the chain rule for obtaining the partial derivatives. Be aware that the derivative of the sigmoid function is: $a(t)' = a(t) * (1 - a(t))$. Verify your result by testing if the prediction error decreased after the weight update.

1 Automatic Differentiation with PyTorch

In this exercise, you have to implement the neural network depicted in Figure 1 in pyTorch and verify your backpropagation calculations from the previous section. The helper script for this task is provided in `ex6_NN_1_Toy.py`. Implement the forward computation as scalar products according to toy example presented in the lecture on neural networks. Invoke the automatic differentiation by calling the ".backward()"-function on your error variable. Are the computed gradients the same as the ones you obtained in your backpropagation implemention results?

2 Classification with a Multi-Layer Perceptron

In this exercise, you must classify the data provided in the `data` file with a neural network in pyTorch. We provide you with an implementation of a neural network with a single hidden layer. The model is defined in the file `mlp.py`. The Trainer class in the file `trainer.py` already implements all functionalities needed to train and test a neural network as well as for visualizing the decision function. Your tasks are as follows:

- **Basics**
  - Run the code by executing `ex6_NN_2_Parabola.py`. What is the validation accuracy after 500 epochs of training? How many parameters does the neural network have?
  - Increase the number of neurons in the hidden layer to 6 and then to 10. What phenomena do you observe?
  - Define a neural network with two hidden layers and three neurons per layer. How many parameters does the network have? Train the network and compare the outcome to the models you have trained before.
- **Regularization**
  - Introduce L2 regularization on the weights by increasing the `weight_decay` parameter. What phenomena can you observe when studying the training and validation loss, as well as the evolution of the decision function?