Prof. Dr. Jörg Müller
Arthur Fleig

AI VIII – Serious Games

UNIVERSITÄT BAYREUTH

Exercises for **Programming, Data Analysis, and Deep Learning in Python** (SoSe 2021)

Exercise Sheet no. 10, *Deadline*: Monday, June 28, 10:15

Notes

- Pay attention to the notes on the previous sheet.

Exercise 29 Linear Regression from a Neural Networks Perspective (12 points)

a) Give the three types of Machine Learning mentioned in the lecture.

b) In the case of linear regression, what are the "task", the "experience", and the "performance measure"?

c) What is the difference between test and training data?

d) Why should one differentiate between training and test data? How was the test data chosen in the lecture?

e) How many parameters do you need when fitting a line to the data and what do these parameters represent, respectively?

f) Give two reasons why a quadratic loss function was used in the lecture.

g) Consider the case of linear regression from the lecture. Write down the corresponding minimization problem. Explain all variables and functions appearing in that problem.

h) Explain the purpose of the "learning rate" α in the gradient descent algorithm and why it should be chosen carefully.

i) Can one expect the cost (or loss) function J to always go to zero (or close to zero, i.e., a value of 0.001)? Why or why not?

j) Is the error always lower on the test data or always lower on the training data or neither? Briefly explain.

k) Consider the following hypothesis function

$$h_w(x) := w_2 x^2 + w_1 x + w_0,$$

where w_0, w_1, and w_2 are scalars and $w \in \mathbb{R}^3$ contains all these three components: $w = (w_0, w_1, w_2)$. (We have replaced the "bias" b by w_0 and added an additional "weight" w_2.) Calculate the gradient of the cost function

$$J(w) := \frac{1}{2} \sum_{i=1}^{m} \left((h_w(x^{(i)}) - y^{(i)}\right)^2.$$

(It is enough to compute the partial derivatives with respect to w_0, w_1, and w_2.)

Exercise 30 Gradient Descent (programming exercise) (12 points)

Before continuing with the exercise, have a look at ***http://fa.bianp.net/teaching/2018/eecs227at/gradient_descent.html*** *and observe the effects of modifying the step size (learning rate) α for various examples.*

Use the Gradient Descent method from the lecture to minimize the function

$$f(w, b) := (1 - w)^2 + 100(b - w^2)^2$$

with respect to w and b. To do so, download the associated Python script from E-Learning and proceed as follows:

a) Implement the function `f(w, b)`.

b) Take the `propagate` function from the lecture notes. Remove the parameters and variables `X`, `y`, `m`, `z`, and `a`. In the "forward propagation" part the cost should be given by `f(w, b)`. In the "backward propagation" part you need to calculate the partial derivatives of f with respect to w and b, as in the lecture (update the formulas for `dw` and `db`). The function should return the gradients and the cost.

c) Implement the function `optimize(w, b, num_iter, learning_rate)` analogously to the lecture, but with one change: The cost should be appended to a `costs` list in **every** step.

d) Find the parameters w and b that minimize $f(w, b)$ and print these parameters together with the corresponding function value. Set $w = 0$, $b = 0$ as the initial parameters. Use 7500 iterations in the `optimize` function. Plot the solution (the parameters w and b) as a single red point (or red star 'r*') in the contour plot provided by the Python script. Illustrate the costs in a new plot.
Hints: The function f is poorly conditioned and requires a good choice of the step size (learning rate). Try a learning rate of 0.001.

e) Perform task d) for different learning rates: $0.001 \cdot i$, $i = 1, ..., 9$. Out of these 9 learning rates, which learning rates work well, which do not? How can you see this in the corresponding costs plots?