Remarks on Homework 2

Through this assignment, you should solve tasks that include the following topics:

- Linear regression,
- Logistic regression and Gradient Descent (GD).

The overall **goal of the assignment** is to understand these topics, but also to understand terms such as the capacity of a model, underfitting and overfitting, training, validation and test errors, in other words, how do we decide what a good model is, and how to choose a model that fits data that we have at disposal.

These topics were covered in the lectures, and tutorials are provided for you (see TeachCenter). In order to solve the tasks, we strongly recommend going first through at least tutorials.

Tasks in this assignment include:

- **doing derivations** (Task 1.1 and Task 2.1). For a given cost function, derive the gradient of the cost function, set all partial derivatives to zero, and make sure that the expression that you get is the same as the expressions given in the assignment sheet, to be precise, Equ. (4) and Equ. (15) from the assignment sheet. There are also questions about the dimensions of matrices and vectors used in these formulas that you have to answer.

  One possible way to solve the Task 1.1.2 would be to reuse the derivation from *Practical 4, 1_Linear_regression.pdf, section 1.4.* You will have to add the part that represents the regularization term: rewrite it as a sum term first and include it in derivation steps.

- **programming part** (implementing parts of the code). These are intended for creating design matrix X, implementation of formulas such as analytical solution for linear regression, gradient descent update rule of parameters, and similar.
  These could be written as 5-10 lines of code in each TODO section.

  When writing the code for the analytical solution for linear regression, you should use the Numpy function *pinv* to calculate Moore-Penrose Pseudoinverse of X. See *Slide 5* of *3_Linear_regression_recap.pdf* in *Practical 4,* and be careful what you use as the argument of *pinv* function.

In Task 2.2, you will have to implement the cost function (Equ. (14)), the gradient of the cost function (Equ. (15)), and Gradient Descent (see the last slide of *Logistic_Regression_recap.pdf,* in *Practical 5*).

In fact, you might want to check IPython notebooks, because you might find useful parts of the code there (for example, creating design matrix, gradient descent).

- **writing a report.** Once done with the programming part, you will have to find an appropriate model and to discuss how you chose it. For example, in Task 1.2 trying out different degrees of the polynomial function, observing training, validation and test errors, and deciding which model is the best.
  In the bonus task (Task 1.3) instead of a polynomial function, a radial basis function should be used. This affects only the way you create design matrix X. Analytical solution will stay the same as in the previous task. Your task will be then to compare this model to the polynomial model.
  The design matrix for Task 2.2 (Equations (16)-(19)) is already provided for you.
  A remark on choosing an appropriate learning rate for Gradient Descent: see Fig. 3.

(Some remarks from the theoretical point of view that might be useful for you)

Basic concepts in Machine Learning

The central challenge in Machine Learning is the ability **to perform well on previously unseen (unobserved) examples**. In other words, the overall goal is to find a model that learns from data, but that is generalizable, which means, that it is able to perform well on new, unseen data.

The model that we choose is characterized by its **capacity** (complexity). The capacity of a model is defined as the ability of a model to fit a wide variety of functions. You can think of it as how capable the model is to fit the data and follow the trends present in the data.

To choose a good model, we need to train it, and to do that, we use a **training set**. Note that this training set is a proportion of the whole dataset that we have at disposal. The dataset that we use should be divided into three (sometimes 2) parts: training, validation and test sets (the validation set is sometimes skipped).

The training set is used to optimize (adjust) model parameters.

The validation set is used to find model complexity. This is the proportion of the whole dataset on which we “validate” our model during training (i.e., test the model while

training), in order to estimate how it performs on the examples that were not used for estimating the parameters (training). Once we decide which model is the best, we will test it using the test set.
The test set is used to estimate generalization error. It is used after the learning when parameters of the model are fixed.

If the model is not able to follow the trend of the data, that is an indicator that the model has low capacity. The model is not able to obtain a low error on the training data, and this is known as **underfitting**. If the model becomes too specialized to data points in the training set (obtaining low training error) but is not performing well on validation (and test) set when new, unseen examples are used, it is doing **overfitting**. The gap between the training and test error is large in the case of overfitting.

Which model is the best? The simplest one that performs well. How to choose it? See Fig. 1 and Fig. 2.

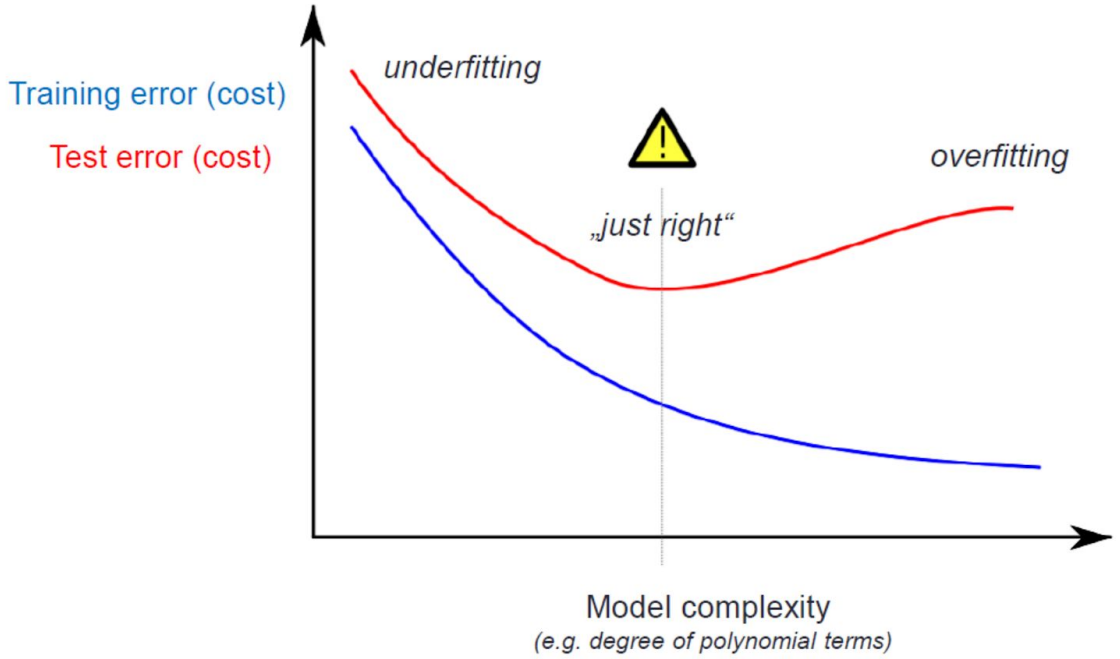

Figure 1. Underfitting and overfitting. Choose your model by observing the validation error (red line, here denoted as test error, because of *“testing while training”*).

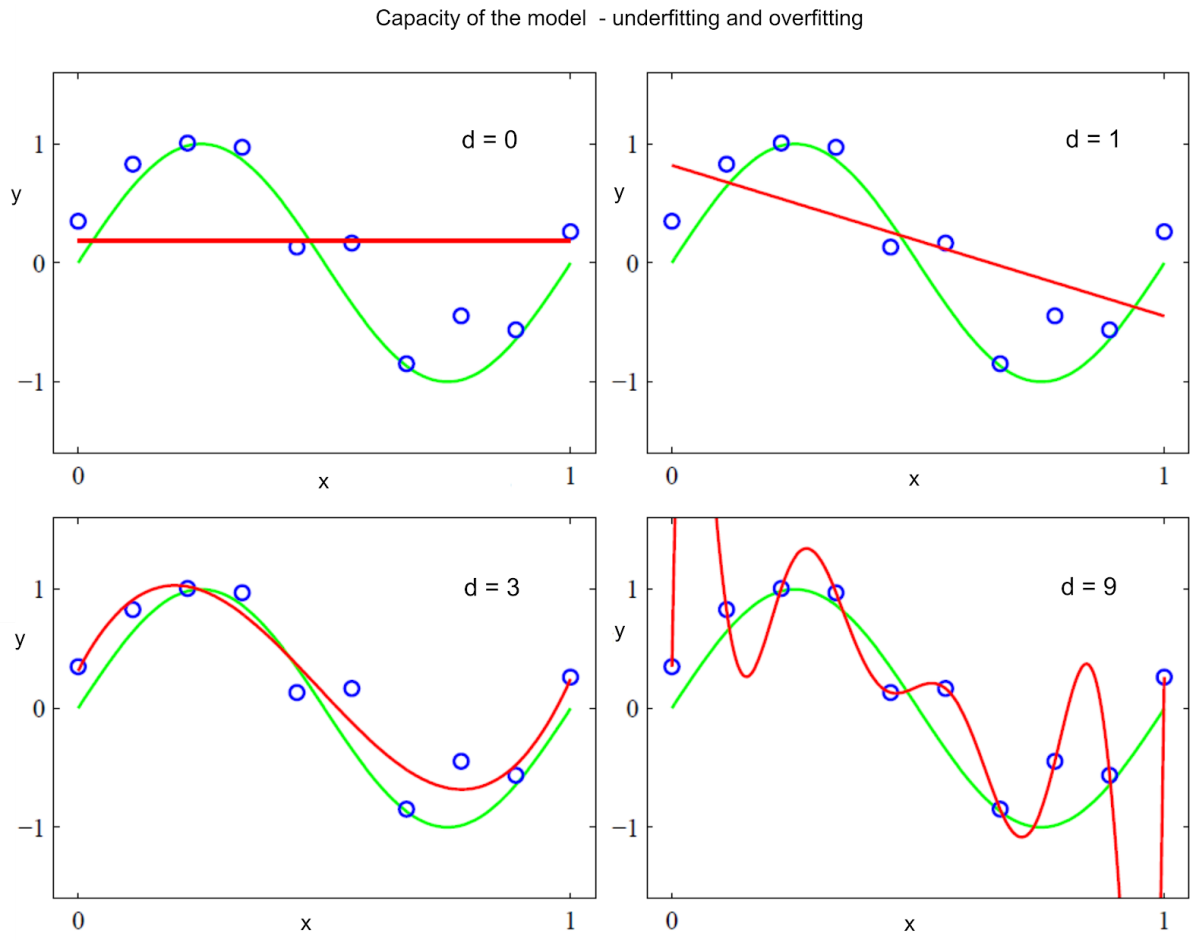

Figure 2. The capacity of a model with different degrees, where *d* denotes the degree of the polynomial function. The first two examples (the top row) are examples of underfitting. The model is not able to follow the trend of the data. The third example (the bottom row, on the left) is the appropriate model, whereas the fourth example (the bottom row, on the right) is an example where the model is doing overfitting. The model is more complex than it should be, and hence it is trying to fit all the examples individually. This will lead to a high test error (when testing on new examples).

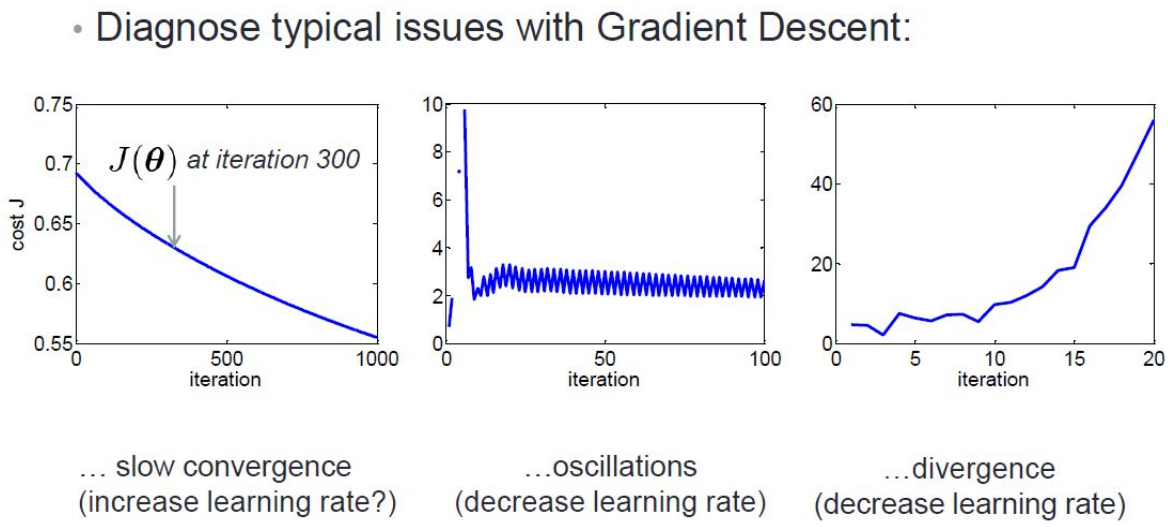

Figure 3. Gradient descent and how to choose an appropriate learning rate.

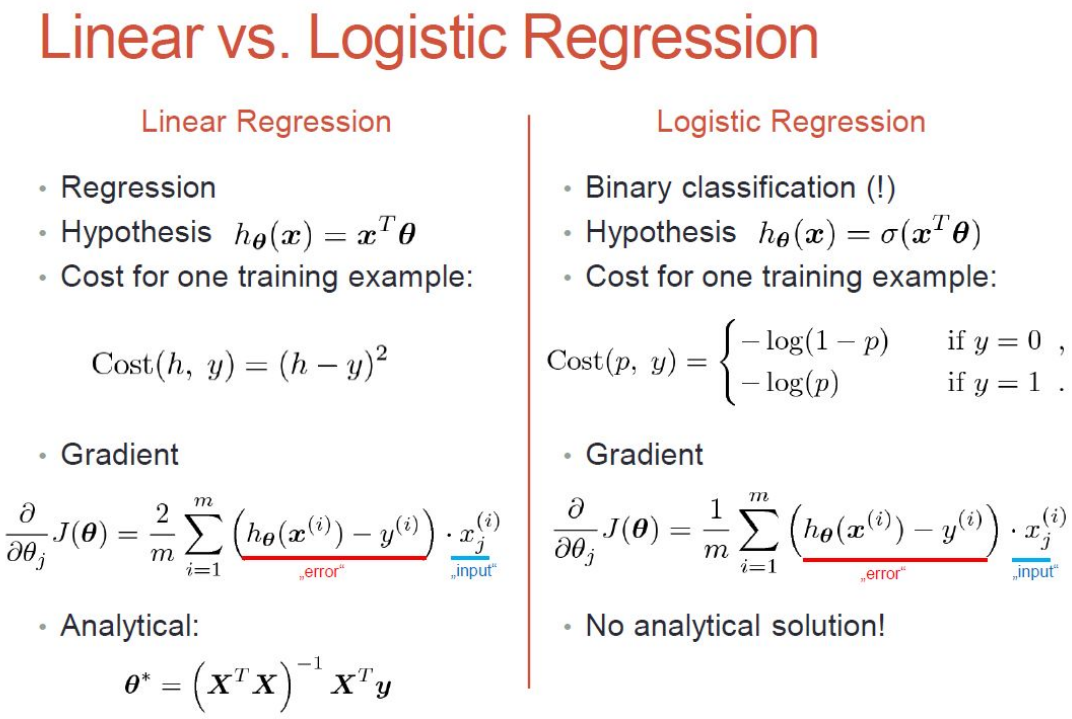

Figure 4. Recap of Linear and logistic regression.