project 2:

least squares regression and nearest neighbor classifiers

solution(s) due:

Jan 7, 2016 at **12:00** via email to **bauckhag@bit.uni-bonn.de**

problem specification:

task 2.1: least squares regression for missing value prediction: Download the file `whData.dat`, remove the outliers and collect the remaining height and weight data in two vectors

$$\boldsymbol{x} = [x_1\ x_2\ \ldots\ x_n]^T \quad \text{and} \quad \boldsymbol{y} = [y_1\ y_2\ \ldots\ y_n]^T,$$

respectively.

Use the method of least squares to fit polynomial models

$$y(x) = \sum_{j=0}^{d} w_j x^j$$

to the data. In particular, fit models for $d \in \{1, 5, 10\}$ and plot your results. Use each of your resulting models to predict a weight value for the outliers (i.e. the data points in `whData.dat` where the weight value is -1).

note:

There are numerous ways of how this can be done. However, depending on how you implement your solution, you may run into numerical problems (you will recognize them when they occur). In this case, it is not acceptable to give up and simply claim that the method did not work. Rather, you should either double your efforts and try to come up with an implementation that works or provide an in-depth analysis as to why and where your code failed and point out possible solutions.

task 2.2: conditional expectation for missing value prediction: Fit a bi-variate Gaussian to the height and weight data in $\boldsymbol{x}$ and $\boldsymbol{y}$ to model the joint density $p(x, y)$ of heights and weights.

Given your fitted bi-variate Gaussian, use the idea of conditional expectation to predict the weight values for the outliers. That is, let x_o denote the available height data of an outlier and compute

$$\mathbb{E}\big[y \mid x_o\big] = \int y\, p\big(y \mid x_o\big)\, dy$$

Do this either analytically as discussed in the lecture or numerically and report your results.

task 2.3: Bayesian regression for missing value prediction: Use the method of Bayesian regression to fit a fifth degree polynomial

$$y(x) = \sum_{j=0}^{5} w_j x^j$$

to the height and weight data in $\boldsymbol{x}$ and $\boldsymbol{y}$. Assume a Gaussian prior

$$p(\boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}_0, \sigma_0^2\, \boldsymbol{I})$$

for the parameter vector $\boldsymbol{w}$ where $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\sigma_0^2 = 3$. Plot you resulting model and compare it to the corresponding model ($d = 5$) from task 2.1

task 2.4: nearest neighbor classifier: implement a function that realizes an n-nearest neighbor classifier, i.e. a function that decides the class label of a test data point from the majority of the labels among the n nearest training data.

Download the files `data2-train.dat` and `data2-test.dat` of labeled 2D data and determine the recognition accuracy (percentage of correctly classified data points) of your classifier for $n \in \{1, 3, 5\}$.

Determine the overall run time for computing the 1-nearest neighbor of every data in `data2-test.dat`.

task 2.5: computing a kD-tree: implement a function that computes *and plots* up to four different kinds of kD-trees of the data in `data2-train.dat` where $k = 2$.

Determine overall run time for computing the 1-nearest neighbor of every data in `data2-test.dat` by means of your kD-tree.

note:

Since `scipy` provides functions for kD-tree computation, you may use those for your run time evaluations. However, just using these functions is of course too easy. Therefore, implement your own function for building and plotting a tree (you may ignore the problem of searching the tree). Your function for building the tree should allow for choosing how to determine the dimension for splitting and for choosing how to determine the split point. In particular, you should enable the following two methods for selecting the splitting dimension:

1. alternate between the x and the y dimension
2. split the data along the dimension of higher variance

With respect to computing the split point, you should enable the following to ideas:

1. split at the midpoint of the data
2. split at the median of the data

Given the data in `data2-train.dat` create all four possible kD-trees and plot them. what do you observe?

general hints and remarks

- Send all your solutions (code, plots, slides) in a ZIP archive to

  bauckhag@bit.uni-bonn.de

- If you insist on using a language other than python, you have to figure out elementary functions/toolboxes in these languages by yourself. **Implementations in MATLAB will not be accepted.**

- Remember that you have to complete all practical projects (and the tasks therein) to be eligible to the written exam at the end of the semester. Your grades (and credits) for this course will be decided based on the exam only, but –once again– you have to succeed in the projects to get there.
- Not handing in a solution implies failing the course.
- Your project work needs to be *satisfactory* to count as a success. Your code and results will be checked and your presentation needs to be convincing.
- If your solutions meets the above requirements and you can demonstrate that they work in practice, it is a *satisfactory* solution.
- A *good* to *very good* solution requires additional efforts especially w.r.t. to elegance and readability of your code. If your code is neither commented nor well structured, your solution is not good! A very good solution requires additional efforts towards the quality of your project presentation in the colloquium. Your presentation should be well timed, consistent, and convincing. Striving for very good solutions should always be your goal!