Introduction

This report describes what has been done to solve exercises in Home Assignment 2 and what were the results. To run code, execute 'main.m' file. Code repository: https://gitlab.cs.ttu.ee/totahv/iti8565

Exercise 1. k-NN

Implemented k-NN classifier. Wrote data generator for 4 clusters in 2D and 3D. Split data 70:30 for training and testing. Implemented Minkowski distance function for Manhattan, Euclidean and Chebyshev. Wrote loop that calculates accuracy of different k values and performance of distance functions. For final classification it chooses k and distance function according to best accuracy. Wrote scatterplot for 2D and 3D that marks correct and incorrect predictions. Wrote unit test to compare implemented k-NN against built-in function, test run 100 times and predicted labels were the same ~80% of the time. Figure 1 shows k-NN performance using different distance functions and k values for 2D and 3D.

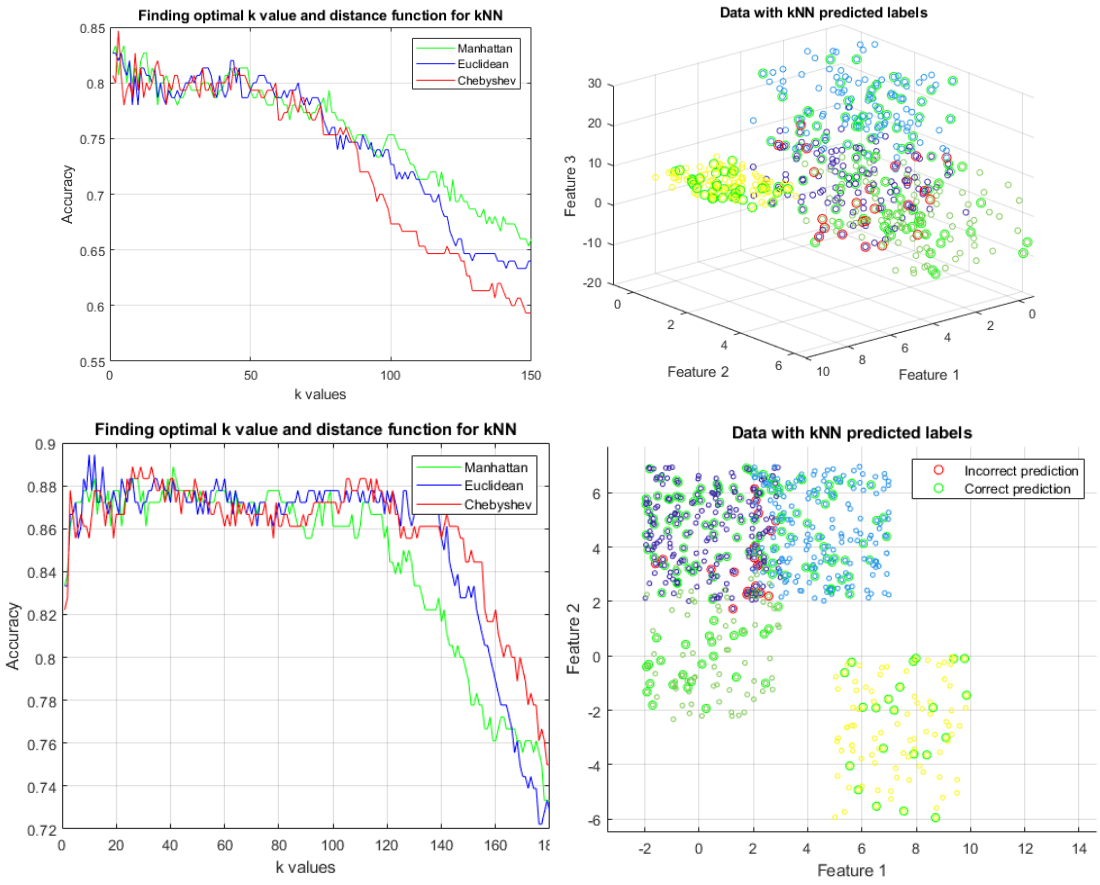

Figure 1 k-NN performance using different distance functions and k values for 2D and 3D.

Exercise 2. Decision trees

Implemented decision tree ID3 algorithm. It uses datasets from Lecture 5 and two other websites that had decision tree tutorials. It calculates entropy using frequency table of one attribute: $E(T) = \sum_{i=1}^{n} -p_i log_2 p_i$. Entropy of two attributes: $E(T,X) = \sum_{c \in X} P(c)E(c)$. Finally, it computes information gain $G(T,X) = E(T) - E(T,X)$. Attribute with highest information gain will be parent, unique values of this attribute will be children and process repeated for children too. Leaf branch is attribute with zero entropy and target decision is made there. Implemented decision tree supports depth up to 4, because function is not recursive and needs refactoring. There is no tree visualization. Tree is built using Matlab struct array, it contains root node, computed children and it can be seen in Matlab variables workspace. Manually tested written tree against results on tutorial website (https://www.saedsayad.com/decision_tree.htm) and final tree was the same.

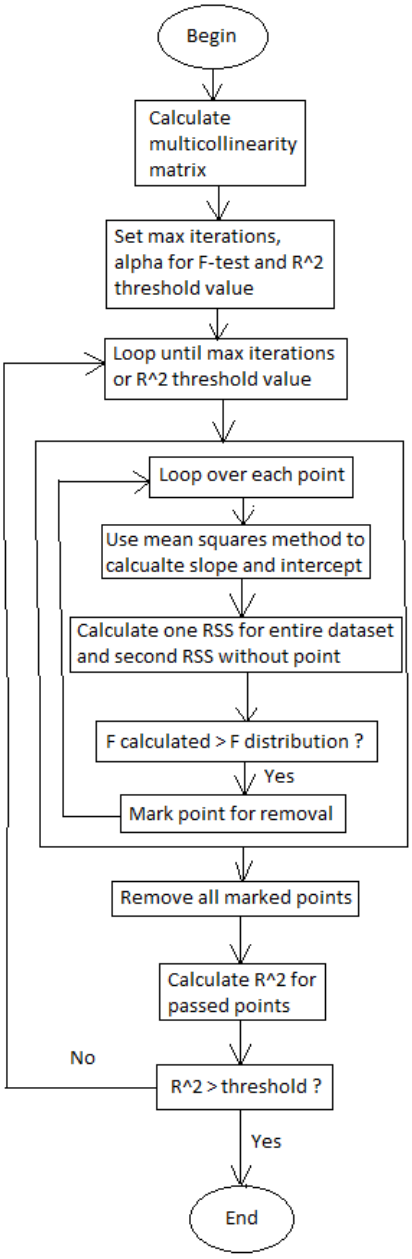

Figure 2 Stepwise regression

Exercise 3. Regression

Implemented stepwise linear regression. Implemented mean squares method for calculating slope and intercept. Figure 2 shows decisions made while creating model. Implemented function for building multicollinearity matrix. In model, it just serves informative purpose, columns are not removed based on correlation. Before training, need to define alpha for F-test, R^2 threshold value and maximum iterations count. Main loop iterates until maximum iterations or R^2 values are reached. Inside main loop, there is loop that iterates over each point. Mean squares calculates slope and intercept. One RSS is calculated for entire dataset, second RSS calculated without selected point. If F-test not passed, then point is marked for removal. Points that have passed F-test move on and are chosen for R^2 calculation. If R^2 is above threshold or maximum iterations is reached, then final slope and intercept parameters are returned. Figure 3 shows learning rate curve and result for 3D dataset. Removed points are not shown on plot. Wrote unit tests to compare correlation coefficient, mean squares and R^2 values against built-in ones and results were comparable.

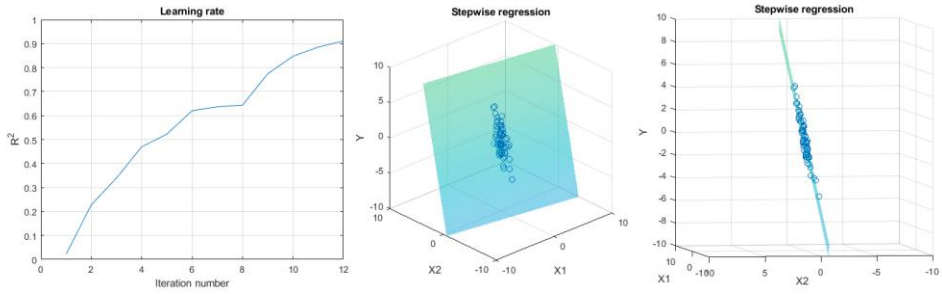

Figure 3 Learning rate and scatterplot result for 3D model (removed points not shown).

Exercise 4. Gradient descent

Implemented gradient descent for linear function. Generated normally distributed data points without significant noise. Used mean squared error function $\frac{1}{n}\sum_{i=0}^{n}(y_i - (ax_i + b))^2$ to be minimized. Final formulas for updating slope a and intersect b: $\begin{cases} a = a - t \cdot (-\frac{2}{n}\sum_{i=0}^{n} x_i(y_i - (ax_i + b))) \\ b = b - t \cdot (-\frac{2}{n}\sum_{i=0}^{n}(y_i - (ax_i + b))) \end{cases}$, where t is step.

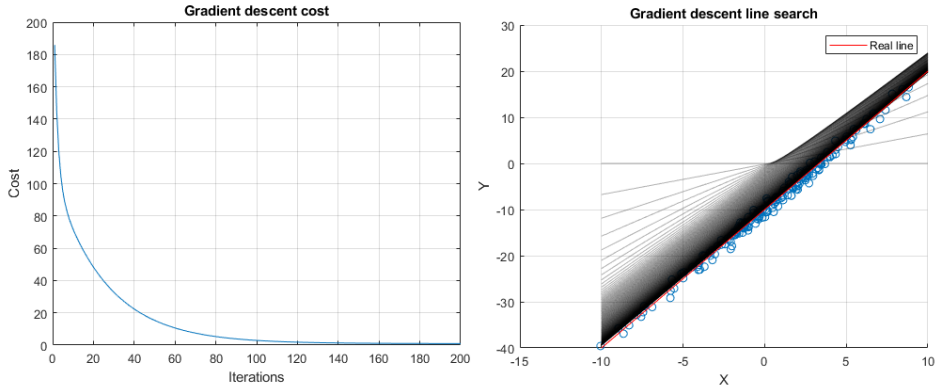

Figure 4 Gradient descent cost function and line search.

Figure 4 visualization is created by saving slope, intersect and cost values into array after each iteration. Red line is real line that gradient descent is trying to approximate.

Conclusion

Implemented k-NN classifier, decision tree, stepwise linear regression and gradient descent for linear model.