Johannes Brinkrolf Bielefeld University

Introduction to Machine Learning (WS 2018/19)
3. Assignment

Released: Thursday, 01.11.2018.
Due: Please solve the exercises in groups of three and present your solutions in the GZI V2-222 (student computer pool) on **Monday, 12.11.2018**, at the assigned time.

- Exercises that require a documentation are marked with **[DOC]**.
- The python toolbox needed for solving the practical exercises can be found at `http://scikit-learn.org/stable/`. It comes with an excellent online description and demos. You can also look at the given demo code.
- On the computer in the GZI you just need to enable *mltools* in the RCINFO Package Selector. You will have a new version of sklearn (0.20.0), updated Spyder 3.3.1 develoment environment, and our toolbox launch Python 3.5 automatically when you start *python* or *ipython* from command line.
- If you have any questions, please ask your tutor or write an email to intromachlearn@techfak.uni-bielefeld.de.

---

1 Your own implementation of the k-NN classifier

(*6 Points*)

In this exercise you will implement the k-NN classifier from scratch.
The file `my_knn_skeleton.py` contains an incomplete implementation of the k-NN classifier. Your task is to implement the missing functionalities marked by *TODO*.

(a) *(2 Points)* Compute the Euclidean distances between the current data point and all stored data points, i.e. all samples from training set.

(b) *(2 Points)* Given the distances to all points, find the k nearest neighbors of the current data point.

(c) *(2 Points)* Predict the label (majority voting) of the current data point based on the labels of the k nearest neighbors of (b).

Hints:

(a) The data is already loaded and split into a test and a training set.

(b) Samples are stored row-wise in the matrices, e.g. `X[i,:]` gives you the i-th data point from the matrix X.

(c) Take a look at the cheat-sheets on Python, Numpy and SciPy (you can find them in the ekVV, E-Learning Space).

(d) If you need help, ask your tutor and/or go to the additional tutorial on Tuesday.

2 Logistic Regression

(*9 Points*)

In this exercise you may use the *Logistic Regression*[1] implementation from scikit-learn.

(a) *(3 Points)* In the ekVV, you can find the file `data_3_logreg_a.npz` containing a 2-dimensional data set for binary classification as well as the two files `logistic_regression_a_skeleton.py` and `utils.py`. Implement the missing pieces in `logistic_regression_a_skeleton.py` marked by *TODO*.

  (a) *(1 Points)* Split the data into a training and a test set and fit a logistic regression model to the training set. Note, for your convenience the data is already loaded.

  (b) *(1 Points)* Compute the error on the training and test set separately.

  (c) *(1 Points)* Visualize the data set and the decision boundary of the fitted model (you might want to use the function `plot_2d_decisionboundary`).

[1] see `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

(b) *(6 Points)* In the ekVV, you can find the file `data_3_logreg_b.npz` containing a 1d data set for binary classification as well as the files `logistic_regression_b_skeleton.py`.
Implement the missing pieces in `logistic_regression_b_skeleton.py` marked by *TODO*.

(a) *(1 Points)* Split the data into a training and a test set and fit a logistic regression model to the 1-dim. data set `data_3_logreg_b.npz` (note that for your convenience the data is already loaded) and compute the test error.

(b) *(1 Points)* **[DOC]** Visualize the 1-dim. data set and explain the performance of logistic regression (you might want to use the function `plot_classification_dataset`).

(c) *(1 Points)* Transform the 1-dim. data set into a 2-dim. data set by mapping all data points x to $\vec{x}' = \begin{pmatrix} x \\ x^2 \end{pmatrix}$.

(d) *(1 Points)* Split the data and fit a logistic regression model, again, but use the new 2-dim. data set.

(e) *(1 Points)* Visualize the new 2-dim. data set and the decision boundary of the fitted model (you might want to use the function `plot_2d_decisionboundary`).

(f) *(1 Points)* **[DOC]** Explain the different performance of logistic regression between the original and the transformed data set.

Hints:

(a) The error of a given classifier and data set is the *number of wrongly classified samples* divided by the *total number of samples*, i.e. $\frac{\text{\#Wrongly classified points}}{\text{\#Data points}}$.

(b) The file `my_knn_skeleton.py` contains an example of using the functions `plot_classification_dataset` and `plot_2d_decisionboundary`.