MAD 2018-19, Assignment 3

Jonas Peters, Fabian Gieseke, Aasa Feragen

hand in until: 18.12.2018 at 23:59

General comments: The assignments in MAD must be completed and written individually. You are allowed (and encouraged) to discuss the exercises in small groups. If you do so, you are required to list your group partners in the submission. The report must be written completely by yourself. In order to pass the assignment, you will need to get at least 40% of the available points. The data needed for the assignment can be found in the assignment folder that you download from Absalon.

Submission instructions: Submit your report as a PDF, not zipped up with the rest. Please add your source code to the submission, both as executable files and as part of your report. To include it in your report, you can use the `lstlisting` environment in LaTeX, or you can include a "print to pdf" output in your pdf report.

Exercise 1 (**Estimating House Prices I, 4 points).** *The Boston Housing dataset contains prices for various houses in the area of Boston (Massachusetts, USA) around the year 1978. Your task is to build models that can be used to estimate these prices given information such as the crime rate, the average number of rooms per apartment, average distances to employment centers, and other attributes. You can find a description of the dataset in Appendix A.*

The `boston.zip` *file that is available on Absalon contains* `boston_train.csv` *and* `boston_test.csv`*, which, in turn, contain the training and the test instances, respectively. You are supposed to make use of the training instances to "build" a model; the test instances are used afterwards to evaluate the "quality" of the model.*[1]

a) *A naive approach to obtain a price estimate for a new house is to resort to the average price of all the houses given in the training set. Implement this simple model, i.e., compute the mean of the house prices given in the* ***training set****. Extend the* `housing_1.py` *file—also available on Absalon—that already contains some code to parse both the training and the test dataset.*
   Hint: Make use of the routines and functions for arrays provided by the Numpy package.

b) *Next, evaluate the quality of this model by implementing and using a function* `rmse(t,tp)` *that computes the root-mean-square error*

$$RMSE(\mathbf{t}, \mathbf{t}') = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|t_i - t'_i\|^2}$$

   for two N-dimensional vectors $\mathbf{t}$ *and* $\mathbf{t}'$ *(here,* `t` *and* `tp` *are lists/arrays containing the values* t_i *and* t'_i*). Use this function to compute the RMSE between the true house prices and the estimates obtained via this simple 'mean' model for the instances in the* ***test set****.*

c) *Visualize the results by generating a 2D scatter plot ("true house prices" vs. "estimates") for all instances in the* ***test set*** *(e.g., via the* `matplotlib.scatter` *function of the* `matplotlib` *package).*

Deliverables. *The adapted* `housing_1.py` *file containing your source code and a) the mean, b) the RMSE, and c) the 2D scatter plot.*

Exercise 2 (**Estimating House Prices II, 4 points).** *Next, you will make use of the multivariate linear regression implementation that was discussed during the lecture. More specifically, you are going to fit a linear regression model of the form*

$$t = f(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D = \mathbf{w}^T \mathbf{x},$$

where t *is the predicted output variable (predicted house price),* $\mathbf{x} = (1, x_1, x_2, \ldots, x_D)^T$ *is the (augmented) vector-valued input variable, and* $\mathbf{w} = (w_0, w_1, \ldots, w_D)$ *are the free parameters. The parameters* w_i define *the regression model, and once they have been estimated, the model can be used to predict outputs for a new input vector* $\mathbf{x}'$*. For the following tasks, make use of the content discussed during the lecture to obtain estimates for these coefficients (i.e., linear modelling via least-squares).*

[1] Note that the test instances must not be used during the training phase. This mimics a realistic scenario where you do not have access to the target values you aim to predict.

a) *We have discussed and developed how to implement linear regression with Python/Numpy. Make use of the corresponding code and (re-)implement linear regression using the supplied template files* `housing_2.py` *and* `linreg.py`*: The former one contains some template code and instructions related to parsing the data and addressing the subtasks detailed below. The latter file contains a Python class* `LinearRegression` *with—so far—incomplete* `fit` *and* `predict` *functions. Start by implementing the* `fit` *function, which shall compute optimal coefficients* $\hat{\mathbf{w}}$ *(remember the offset parameter* w_0*). Store the computed weights in the local variable* `self.w`*; see the template code as well as the code discussed during the lecture for details.*

Note: You should not use a built-in function for computing the optimal weights/fitting the linear regression model. The goal is to make use of standard vector/matrix operations provided by the Numpy package to compute the optimal weights.

b) *Use the code developed in a) to fit a linear regression model on a version of the* ***training set*** *that only contains* the first feature *(CRIM). What are the two weights? What can you learn from them?*

c) *Next, fit a model on* all the features *in the* ***training set****. What are the computed weights?*

d) *Finally, implement the* `predict` *function in* `linreg.py`*, which shall compute, for an array of input instances, the associated predictions. Use the* `predict` *function to compute predictions for the* ***test instances*** *using (i) the model that is based on the first feature only and (ii) the model that is based on all features? What are the two induced RMSE values? Also, generate a 2D scatter plot for each of the two cases ("true house prices" vs. "estimates") based on the test instances.*

Deliverables. *Both adapted files* `housing_2.py` *and* `linreg.py` *containing your source code and b) weights* $\hat{w}_0, \hat{w}_1$ *and a two-liner, c) weights* $\hat{w}_i$*, and d) the RMSE for both cases along with two scatter plots.*

Exercise 3 (**Total Training Loss, 4 points, Exercise 1.10 in Rogers & Girolami).** *Derive the optimal least squares parameter values* $\hat{\mathbf{w}}$ *for the total training loss*

$$\mathcal{L} = \sum_{n=1}^{N} \left(f(\mathbf{x}_n; \mathbf{w}) - t_n\right)^2 = \sum_{n=1}^{N} \left(\mathbf{w}^T \mathbf{x}_n - t_n\right)^2$$

How does the expression compare with that derived from the average loss?

Deliverables. *Mathematical derivation for the optimal least squares parameter* $\hat{\mathbf{w}}$*.*

Exercise 4 (**Weighted Average Loss, 4 points, Exercise 1.11 in Rogers & Girolami).** *The following expression is known as the* ***weighted*** *average loss:*

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \alpha_n \left(f(\mathbf{x}_n; \mathbf{w}) - t_n\right)^2 = \frac{1}{N} \sum_{n=1}^{N} \alpha_n \left(\mathbf{w}^T \mathbf{x}_n - t_n\right)^2$$

where the influence of each data point is controlled by its associated $\alpha_n > 0$ *parameter.*

1. *Assuming that each* α_n *is fixed, derive the optimal least squares parameter value* $\hat{\mathbf{w}}$*.*
   Hint: Make use of a diagonal matrix $\mathbf{A}$ *that contains the weights* $\alpha_1, \ldots, \alpha_N$*.*

2. *Similar to Exercise 2, implement a corresponding regression model in Python (e.g., implement a new* `linweighreg.py` *module). Afterwards, fit a model on all the features in the training set* `boston_train.csv` *using* $\alpha_n = t_n^2$*. Compute corresponding predictions for the test instances given in* `boston_test.csv` *and generate a scatter plot as in Exercise 2. What do you expect to happen? What do you observe? Do the additional weights have an influence on the outcome?*

Deliverables. *(a) Mathematical derivation for the optimal least squares parameter* $\hat{\mathbf{w}}$*. (b) Implementation of the weighted version, scatter plot, and short answers to the questions raised.*

Appendix

A The Boston Housing Dataset

A detailed description of the dataset can be found here: `https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html`. For the sake of illustration, the main characteristics are given below. On Absalon, you can find a zip file `boston.zip` that contains the following two files:

- `boston_train.csv`: Training set containing 253 instances. Each row contains 14 values separated by commas. The first 13 values correspond to the features; the last value (MEDV) is the target (house price).

- `boston_test.csv`: Corresponding test set containing different 253 instances.

Each row contains the following pieces of information:

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's