Introduction to Data Science 2017
Assignment 5

Aasa Feragen, François Lauze, Christian Igel

Your solution to Assignment 5 must be uploaded to Absalon no later than April 2nd, 11:59 am.

Guidelines for the assignment:

- **The assignments in IDS must be completed and written individually.** This means that your code and report must be written completely by yourself.
- Upload your report as a single PDF file (no Word) named `firstname.lastname.pdf`.
- Upload your Python code. If the code is in several files, upload them in a `.zip` archive.

Each of the seven exercises is 10 points worth, though the last one is a bonus one: the total amount fo points is 60+ for this assignment, meaning that you cannot loose points in exercsie 7, but you can gain some.

Linear regression

In this part of the assignment you will use and evaluate linear regression for predicting the quality of red wine from its physicochemical properties. See the Appendix below for a proper description of the *red wine* dataset.

Exercise 1 (Using Linear Regression)**.** Your task is to predict the quality score of red wine from its physicochemical properties using linear regression. You are going to learn a linear model

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

where t is the predicted output variable (quality score), $\mathbf{x} = (1, x_1, x_2, \ldots, x_D)^T$ is the vector-valued input variable (physicochemical properties), 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 new input values $\mathbf{x}_0$.

a) Implement linear regression using the supplied template function `multivarlinreg.py`. Your code should load the data matrix X containing the input variables, as well as the output vector t, and output an estimate of the free parameters in the model, that is the w_i in the form of the vector $\mathbf{w}$. Remember the offset parameter w_0. You should not use a built-in function for regression.
b) Run your regression function on a version of the training set that only contains the first feature (fixed acidity). Please report your weights. What can you learn from them?
c) Run your regression function on all the features in the training set and report your estimated parameters w_i. What do they tell you about how the different physiochemical properties relate to the wine quality?

NB! Take care to separate out the wine quality parameter from the data matrix before you learn the model!

Deliverables. a) Code, b) Parameters w_i and a short description, c) Parameters w_i and a short description.

Exercise 2 (Evaluating Linear Regression)**.**

a) Implement a function `rmse.py` using the supplied template, which computes the root-mean-square error

$$RMSE(\mathbf{w}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|y_i - h(\mathbf{x}_i, \mathbf{w})\|^2}$$

for a set of known input-output values $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. Here, y_i is the recorded output value associated to the i^{th} data point $\mathbf{x}_i$ in the dataset, and $f(\mathbf{x}_i, \mathbf{w})$ is the output value associated to the input $\mathbf{x}_i$ as predicted by your regression model. Your code should take as input the predicted values $h(\mathbf{x}_i, \mathbf{w})$ and ground truth output values y_i. You should not use a built-in function for RMSE.

b) Build the regression model for 1-dimensional input variables using the training set as in 1b). Use the first feature for the test set to compute the RMSE of your model.

c) Build the regression model for the full 11-dimensional input variables using the training set as in 1c). Use the full-dimensional test set to compute the RMSE of this model. How does it compare to your answer from b)?

In b) and c) it is OK to use a built-in function for regression if you have not completed exercise 1
Deliverables. a) Code, b) the RMSE, c) the RMSE and a short description.

Random forest

Exercise 3 (Random forest & normalization)**.** As discussed, normalizing each component to zero mean and variance one (measured on the training set) is a common preprocessing step, which can remove undesired biases due to different scaling.

Using this normalization affects different classification methods differently. How does it affect a random forest classifier?

Deliverables. Short discussion on how normalization of the features to zero mean and unit variance affects a random forest

Now we apply a random forest classifier to the data from the previous assignments.

```
from sklearn.ensemble import RandomForestClassifier
```

Does it beat the nearest neighbor classifier?

Exercise 4 (Applying random forrest)**.** Train a random forest with 50 trees on `IDSWeedCropTrain.csv` from the previous assignment(s). Test the resulting classifier using the data in `IDSWeedCropTest.csv`.

Deliverables. Source code; prediction accuracy of the random forest

Gradient descent

Exercise 5 (Gradient descent & learning rates)**.** Apply gradient descent to find the minimum of the function $f(x) = e^{-x/2} + 10x^2$.

Detailed instructions:

1. Compute the derivative of the function f.
2. Apply gradient descent with learning rates $\eta = 0.1, 0.01, 0.001, 0.0001$.
3. For each of the learning rates do the following:

(a) Take $x = 1$ as a starting point.

(b) Visualize the tangent lines and gradient descent steps for the first three iterations (produce 4 plots for your report corresponding to gradient descent with the four learning rates). The first tangent line should be at the initial point $x = 1$.

(c) Visualize gradient descent steps for the first 10 iterations (another 4 plots; no visualization of tangent lines in this case).

(d) Run the algorithm until the magnitude of the gradient falls below 10^{-10} or the algorithm has exceeded 10,000 iterations. Report the number of iterations it took the algorithm to converge and the function value at the final iteration (4×2 values corresponding to the four learning rates).

Logistic Regression

Exercise 6 (Logistic regression implementation)**.**
In this exercise you will implement, run and test logistic regression on two simple data sets, made of two classes each. They come from Fisher's Iris dataset and are described below.

1. Make a scatter plot of each dataset, with point color depending on classes. What do you observe?
2. Implement logistic regression as presented in the lectures. Your function should take as input the training set data matrix, the training set label vector, and the test set data matrix.
3. Apply logistic regression to each dataset described below. You are supposed to solve a binary classification task.
4. Report the training an test error as measured by the 0-1 loss. Furthermore, report the three parameters of the (affine) linear model.

The datasets we use are modified subsets of the Iris data set. Each dataset consists of two files, a training and a test set, The first pair is called `Iris2D1_train.txt` and `Iris2D1_test.txt`. The second pair is called `Iris2D2_train.txt` and `Iris2D2_test.txt`.

Each entry contains 2 numbers and a class label. Thus row n contains (x_{1n}, x_{2n}, y_n). Each dataset is binary, with classes labeled 0, 1. Both `Iris2D1_train.txt` and `Iris2D2_train.txt` have 70 entries, 40 of class 0 and 30 of class 1. Both `Iris2D1_test.txt` and `Iris2D2_test.txt` have 30 entries, 10 of class 0 and 20 of class 1.

The model parameter $\mathbf{w} = (w_1, w_2, w_0)$ and the affine (linear) model takes the form

$$(x_1, x_2) \mapsto w_0 + x_1 w_1 + x_2 w_2 = \mathbf{w}^T (1, x_1, x_2)^T.$$

This is the same format as used in the lecture's Jupyter Notebook.

Include all your code in the accompanying `.zip` file.

Exercise 7 (Bonus/Chocolate exercise: Logistic regression loss-gradient (this is exercise 3.7 of Abu–Mostafa et al. [2012]))**.**

1. Show that the gradient of the in-sample function $E_{in}(\mathbf{w})$, as defined in the lecture, is given by

$$\begin{aligned} \nabla E_{in}(\mathbf{w}) &= -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^\top \mathbf{x}_n}} \\ &= \frac{1}{N} \sum_{n=1}^{N} -y_n \mathbf{x}_n \theta\left(-y_n \mathbf{w}^\top \mathbf{x}_n\right). \end{aligned}$$

2. Argue that a 'misclassified' example contributes more to the gradient than a correctly classified one.

Appendix: Data material

The red wine dataset

The *red wine* dataset P. Cortez and Reis [2009], publicly available in the UCI database `https://archive.ics.uci.edu/ml/datasets/Wine+Quality`, contains a set of physicochemical features collected for 1599 wines along with a wine quality score in the form of an integer between 0 and 10, based on sensory data. The associated predictive task is to predict the quality score from the physicochemical data.

The physicochemical features are

1. fixed acidity
2. volatile acidity
3. citric acid
4. residual sugar
5. chlorides
6. free sulfur dioxide
7. total sulfur dioxide
8. density
9. pH
10. sulfates
11. alcohol

For the sake of this exercise, the dataset has been randomly divided into a *training set*, which you can find in `redwine_training.txt` and `redwine_testing.txt`, respectively. The two datasets are stored in the form of $N \times 12$ matrices, where the quality score is stored in the 12th column – remember to remove this from the data matrix prior to analysis.

You can read the data as usual, using the command

```
data_train = np.loadtxt('redwine_training.txt')
data_test = np.loadtxt('redwine_testing.txt')
```

The Iris dataset

The Iris dataset is a classical one, used by R. Fisher. From Wikipedia: "*The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.*"

For the assignment, `Iris2D2_train.txt` and `Iris2D2_test.txt` were made of the 100 first entries, keeping only features 0 and 1. `Iris2D1_train.txt` and `Iris2D1_test.txt` were made of the 100 last entries, keeping only features 0 and 2. Classes 1 and 2 for these last entries where remapped to 0 and 1. Each of the 100 entries where randomly permuted and divided into 70 training and 30 testing entries.

References

Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from data.* AMLbook, 2012.

F. A. T. M. P. Cortez, A. Cerdeira and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 4(47):547–553, 2009.