Introduction to Data Science 2018
Assignment 5-6

François Lauze, Thomas Hamelryck

Assignment 5-6 is a double assignment, i.e., it will have double weight with respect to the four others. You have also more time to complete it. Your solution to Assignment 5-6 must be uploaded to Absalon no later than April 15th, 23.59 (at the end of the exam week).

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.

This assignment consists of 10 exercises, each of the exercises is 10 points worth. You will be noted to the best of 8 exercises. The total amount of points for this assignment is 80 points. Note that if you exceed 80 points, this will be taken into account positively!

Bayesian Statistics

In this first part of the assignment, you are asked a few things about Bayesian Statistic. There are simple, theoretical questions, which do not require any computation.

Exercise 1. Consider the following linear model.

$$y = ax + b + \varepsilon \tag{1}$$

where ε is a Gaussian noise term, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, i.e. with mean 0 and standard deviation σ^2.

a) Write explicitly the probability density function of ε.

b) From it write the conditional probability distribution of y knowing a, x and b, i.e., the probability of observing y knowing a, x and b.

c) What are the parameters of the linear model (1)?

d) In a Bayesian treatment of the linear model (1), how many priors are needed, and over which parameters. Can you use Gaussian priors for all parameters?

Deliverables. A short answer to each question.

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 2 (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, call your 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 3 (Evaluating Linear Regression)**.**

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

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

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 $f(\mathbf{x}_i, \mathbf{w})$ 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 2
Deliverables. a) Code, b) the RMSE, c) the RMSE and a short description.

Random forest

Exercise 4 (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 5 (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 6 (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 7 (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 8 (Logistic regression loss-gradient (this is exercise 3.7 of Abu-Mostafa et al. [2012]))**.**

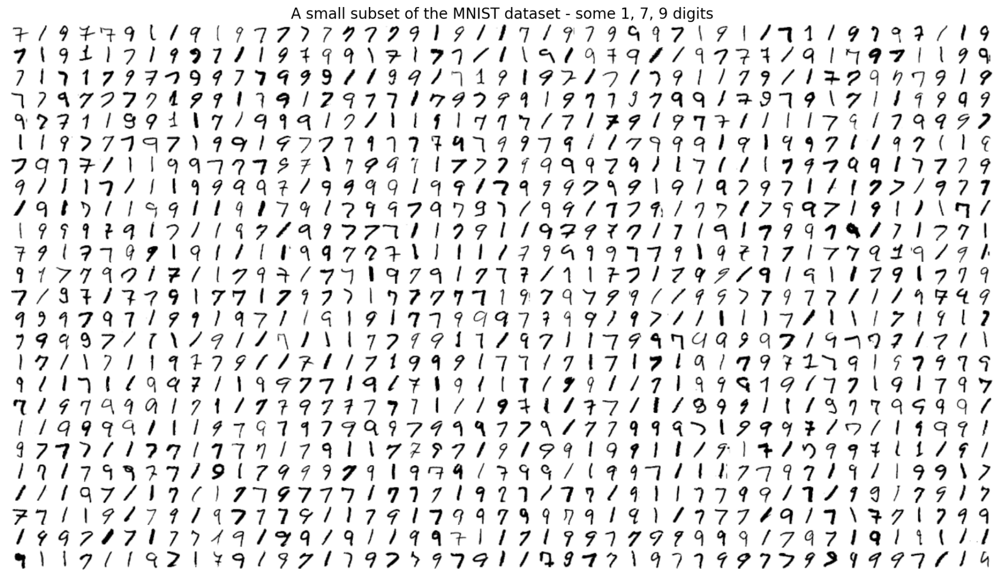
1. Show that the gradient of the in-sample function $E_{in}(\mathbf{w})$, as defined in the lecture, is given by

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

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

Dimensionality Reduction, Classification and Clustering

In the following questions, you will reuse some of your previous work to explore a (small) amount of the MNIST dataset, where images of hand written digits 1, 7, and 9 have been selected. A detailed description is provided in the appendix. The goal of this exercise is to compare performances of clustering and classification without and with dimensionality reduction.



Exercise 9 (Clustering and classification I.)**.** In this exercise, you will use some of the algorithms you used in previous assignments for clustering and classifying the dataset.

a) Run the k-Means algorithm with $k = 3$ to cluster the dataset. Are the results meaningful? As the k-means algorithm will return labels 0, 1 and 2, you cannot compare them directly with the 1, 7 and 9 from the label file. Count instead the proportion of 1s, 7s and 9s in each cluster. For that you may want to prepend the label data to the image data, but **not use it** in the k-means algorithm, e.g., by fitting all but the first column.

b) Classification. Train a k-NN classifier on the dataset, using $n-$fold validation to obtain the optimal k. Report test accuracy. Note that if you use `KNeighborsClassifier`, you can pass it the true labels as labels.

Deliverables. a) Description of the software. 3 cluster centers as images, percentage of each digit per cluster and a one liner describing the results. b) Test accuracy for k-best.

Exercise 10 (Clustering and classification after dimensionality reduction.)**.** In the second part, you will use dimensionality reduction (i.e. PCA here) to first reduce the dimensionality, and then perform clustering and classification.

a) Compute the PCA of the *training* data. Plot the cumulative variance (in %) w.r.t. the principal components.

b) Run the k-Means algorithm, again with $k = 3$ to cluster the data projected on the first principal components. Experiment with 20 and 200 components. Count the proportion of 1s, 7s and 9s in each cluster. Again, you should prepend the labels to the *projected data on the principal components* as first column, but take care of not using it in k-means. Compare with the non-PCA clustering result in the previous exercise.

c) Classification. Train a k-NN classifier on the dataset:

   - Do it using the first 20 principal components. Use n-fold validation to obtain the optimal k. Report test accuracy.
   - Do it again with the first 200 principal components. Use n-fold validation to obtain the optimal k. Report test accuracy.

Deliverables. a). A short description of the software a plot of the cumulative percentage of variance w.r.t. components, a one liner about it. b) 3 cluster centers as images: *this means recreating images from their projections on principal components!* percentage of each digit per cluster and a one liner describing the results. c) Test accuracy for best k and a one liner about comparison of performances in Ex.9b) and Ex.10c)

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.

The MNIST dataset

This is too a very used one in Computer Vision and Machine Learning. The dataset has been described in `http://yann.lecun.com/exdb/mnist/` (see Y. LeCun and Haffner). It consists of a training set and a test set of images of hand written digits. All images have been preprocessed to remove noise and well separate foreground from background and have the same dimension $28 \times 28 = 784$ pixels.

The training set contains 60,000 images, while the test set contains 10,000 of them. Ground truth labels from 0 to 9 are available for the data. For the purpose of this assignment, 900 training images and 225 test images from the MNIST dataset were collected and randomly mixed. The dataset `MNIST_179_digits.txt` is available as a text file readable by `numpy.loadtxt()`, it should return a (1125,784) array. True labels are available in the `MNIST_179_labels.txt`, also readable by `numpy.loadtxt()`, would should return a (1125,) array.

The reduced train set is provided as a numpy readable textfile `train_1_7_9images.txt`. It has 900 lines with 784 entries each and should be loaded with `numpy.loadtxt`. To visualize a sample as a digit, you need first to re-dimension the array containing the digit image values to an array of shape (28,28).

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.

Y. B. Y. LeCun, L. Bottou and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324.