MAD 2018/2019 Exam

Jonas Peters, Fabian Gieseke, Aasa Feragen

14.01.2019 – 20.1.2019

You have to submit your individual solution of the exam electronically via the **Digital Exam** system. The submission deadline is **20 January 2019, 23:59**. The exam must be solved **individually**, i.e., you are **not allowed to work in teams or to discuss the exam questions with other students**. Your solution should consist of

1. a pdf file `answers.pdf` containing your answers, and
2. a `code.zip` file containing the associated code.

The correction will be done in an anonymous fashion; **hence you should not mention your name somewhere**. Instead, please **add your exam number** at the beginning of your `answers.pdf`.

WARNING: The goal of this exam is to evaluate your individual skills. We have to report any suspicion of cheating, in particular collaboration with other students, to the head of studies. Note that, if proven guilty, you may be expelled from the university. Please: Do not put yourself and your fellow students at this risk.

Feel free to ask questions via Absalon, but make sure that you do not reveal any significant parts of the solution. In doubt, just approach us directly via e-mail! Any additional hint given by us will be made available to all of you via Absalon. Some further comments:

1. *You* ***are allowed*** *to reuse the code that was made available to you via Absalon as well as the code you have developed in the course of the assignments. If you reuse code from the lectures or from the assignments, make sure to put a reference to this in your code, and if your code was developed as part of an assignment, in collaboration with a fellow student, add a corresponding comment to your answers, although keeping anonymity (i.e., just mention which parts stem from team work). In case you reuse code snippets you have found on the internet, please make sure that you provide a reference to this external source as well.*
2. *In case you notice any inaccuracies in the problem descriptions below, please let us know. If needed, we will provide updates and additional comments via Absalon. Thus, make sure that you check Absalon for announcements and discussions regularly!*
3. *For the coding tasks, you are given Python files/Jupyter notebook templates. You are supposed to complete these files and notebooks. Note that you are allowed to import additional Python packages and to make use of the functions provided by, e.g., the Numpy package. However, you should not use built-in functions for the task at hand (e.g., a single `kmeans` function from some other package that implements the K-means clustering approach). If in doubt, please ask us via e-mail!*
4. *All templates and data can be found in the file `template.zip`.*
5. *Good luck! :-)*

Exercise 1 (Continuous Random Variables, 3 points). *a) Let X be a continuous random variable with probability density function (pdf)*

$$f(x) = \begin{cases} c \cdot (5-x) & \text{if } 0 \le x \le 5 \\ 0 & \text{otherwise.} \end{cases}$$

Compute the correct value of $c \in \mathbb{R}$.

b) Let V be a continous random variable with cumulative distribution function (cdf)

$$F(v) = \begin{cases} 0 & \text{if} \quad v < 0 \\ v^5 & \text{if} \quad 0 \le v \le 1 \\ 1 & \text{if} \quad v > 1. \end{cases}$$

Compute the pdf f of V and its mean $\mathbb{E}V$.

c) Let $Y_1, Y_2, \ldots$ and $Z_1, Z_2, \ldots$ be continuous random variables such that $Y_n \xrightarrow{\mathbb{P}} Y$ and $Z_n \xrightarrow{\mathbb{P}} Z$. Prove that $Y_n + Z_n \xrightarrow{\mathbb{P}} Y + Z$. (Hint: you may use the triangle inequality: for all $a, b \in \mathbb{R}$, we have $|a+b| \le |a| + |b|$. You can look up the definitions of $\xrightarrow{\mathbb{P}}$ in the essentials.)

Deliverables. *a) real value (with argument) b) pdf (with argument), mean (with argument) c) proof*

Exercise 2 (Statistics, 3 points). *a) Let X_1, X_2 be i.i.d. random variables with pdf f_β that depends on a parameter β (that we want to estimate):*

$$f_\beta(x) = \begin{cases} \frac{2}{\beta^2} \cdot (\beta - x) & \text{if } 0 \le x \le \beta \\ 0 & \text{otherwise.} \end{cases}$$

Given the two observations $X_1 = x_1$ and $X_2 = x_2$, write down the likelihood (which should be a function of β, x_1, and x_2.) Compute the maximum likelihood estimator for $x_1 = 3$ and $x_2 = 4$.

b) Lisa bought a machine that records a person's hand movement when performing a coin flip. Using a complicated video analysis, it then predicts whether the outcome will be "heads" or "tails". The advertisement claimed that more than 50% of the machine's predictions are correct. At home, Lisa wants to check this claim using a statistical hypothesis test. Let X be the number of coin flip results that were predicted correctly by the machine (out of $n = 20$ trials). We assume that $X \sim \mathcal{B}in(n, \theta)$ with $n = 20$. As null and alternative hypothesis we use

$$H_0 : \theta = 0.5 \quad \text{and} \quad H_1 : \theta \neq 0.5.$$

Suppose that Lisa observes $X = 13$. Perform the corresponding test to the level 0.05 (Hint: use the "six steps" from the essentials. The rejection region $\mathcal{R}$ should be of the form $\mathcal{R} = \{0, \ldots, a\} \cup \{20 - a, \ldots, 20\}$ for some $a \in \mathbb{N}$.)

Deliverables. *a) likelihood function, value for beta (with argument) b) six steps*

Exercise 3 (Clustering, 3 points). *In this exercise, you will apply the K-means clustering approach to two datasets. For the tasks outlined below, please extend the Jupyter notebook `K_Means.ipynb` that is provided to you. Note that this notebook already contains an implementation for K-means (which you should not modify).*

a) Apply K-means to the well-known Old Faithful geyser *dataset [2] that is given in the file `old_faithful.csv`. Follow the instructions provided in the notebook and generate a 2D scatter plot showing the data (there are two features per instance). Apply the clustering approach to find two clusters; initialize the model via `KMeans(n_clusters=2, max_iter=30, seed=0)`. What are the cluster means computed by the approach?*

b) Next, you will use K-means to do image segmentation. Load and plot the image `copenhagen_tiny.jpg`, which is shown in Figure 1. Afterwards, consider each pixel as a point in $\mathbb{R}^3$ (for each pixel, one is given three color values) and apply K-means using `KMeans(n_clusters=5, max_iter=5, seed=0)`. What are the induced cluster means? Generate and plot the segmented image (similar to slide 21 of lecture L13).

c) Finally, segment the slightly larger image `copenhagen.jpg` in a similar fashion. The implementation provided is not very efficient from a computational perspective, and dealing with this image already takes quite some time! To accelerate the computations, you are supposed to apply K-means on a random subset of only 5000 points to identify suitable cluster means for this bigger image (you can use the Numpy function `numpy.random.choice` to draw a uniform random subset of indices without replacement). Use `n_clusters=16` to search for 16 clusters this time; keep the other parameter assignments. What are the cluster means obtained via your code? Generate and plot the segmented image!

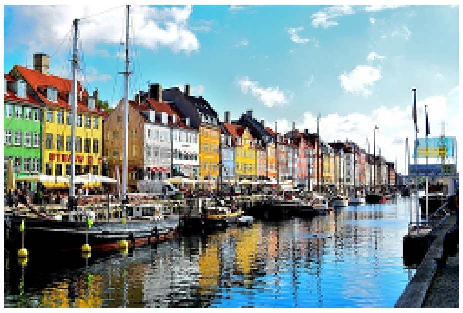

Figure 1: `copenhagen_tiny.jpg`

Deliverables. *All your source code (as a filled-out Jupyter notebook). Add to your report: a) The 2D scatter plot and the cluster means. b) The images generated and the five cluster means. c) The images generated and the 16 cluster means obtained via your code.*

Exercise 4 (Regression, 3 points). *We have seen how one can obtain non-linear regression models by extending the data matrix. An alternative to this approach works as follows: Given a training set* $T = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\} \subset \mathbb{R}^D \times \mathbb{R}$ *of points* $\mathbf{x}_i$ *with associated targets* t_i*, one can compute, for* ***each*** *new point* $\bar{\mathbf{x}} \in \mathbb{R}^D$*, the following model:*

$$\hat{\mathbf{w}}(\bar{\mathbf{x}}) = \operatorname{argmin}_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} h_\sigma(\bar{\mathbf{x}}, \mathbf{x}_n) \left(\mathbf{w}^T \mathbf{x}_n - t_n\right)^2 \tag{1}$$

A standard choice for the function $h_\sigma(\bar{\mathbf{x}}, \mathbf{x}_n)$ *is* $h_\sigma(\bar{\mathbf{x}}, \mathbf{x}_n) = \exp\left(-\frac{\|\bar{\mathbf{x}}-\mathbf{x}_n\|^2}{2\sigma^2}\right)$ *with so-called bandwith parameter* σ*.*

a) *Explain the influence of the values computed by the function* h*. In particular, what happens if* $\|\bar{\mathbf{x}} - \mathbf{x}_n\|$ *is very small and what happens if* $\|\bar{\mathbf{x}} - \mathbf{x}_n\|$ *is very large? What is the influence of the parameter* σ*?*

b) *Extend the Jupyter notebook* `Regression.ipynb` *that is provided to you (along with the* `linreg.py` *file that was already made available on Absalon). The first part generates a simple toy dataset and applies a linear regression model to the data. Your task is to extend the template and to implement the non-linear approach described above. In particular, you are supposed to generate corresponding plots for the data at hand using the following three bandwidth assignments:* $\sigma = 0.1$, $\sigma = 1.0$, *and* $\sigma = 10.0$*. Comment on the outcome. What happens if you further increase the value for* σ*?*

Comment: Smaller values for σ *might lead to numerical problems; you don't have to fix those.*

Deliverables. *All your source code (as a filled-out Jupyter notebook). Add to your report: a) Short explanation (3-4 lines). b) Three plots along with a short discussion (3-4 lines).*

Exercise 5 (PCA and Classification, 6 points). *In this exercise, you will perform PCA and classification on the DD dataset found in the file* `DD.tar.gz`*. Please see the appendix for a description of the dataset. You should write your code using the supplied Jupyter notebook template* `PCA and classification.ipynb`*.*

Again, you are allowed to reuse code from the lectures, including PCA and logistic regression. If you do so, remember to put a reference to it in comments in your code.

a) *Before we get started on the classification task, run PCA on the training set given in* `DD_train.txt` *and visualize the dataset by projecting the data onto the first two principal components and making a scatter plot of the result. Usin the class labels in* `DD_train_labels.txt`*, give distinct colors or shapes to the points belonging to different classes. What do you see?*

b) *Implement a* k*-nearest neighbor classifier (*k*-NN) using the supplied template function* `knn.py`*, which takes as input a training set, a test set, a set of training labels, a set of test labels, and a parameter* k*, and outputs a vector of labels and an accuracy for the test set.*

c) *Apply your* k*-NN classifier using* `DD_train.txt` *as training data and report the accuracy (the proportion of correct predictions) of the corresponding classifier on both the training data and the test data in the* `DD_test.txt` *dataset for* $k = 1, 3, 5$*. What do you observe for the two datasets? What do you observe as you vary* k*?*

d) Here, you will implement 10-fold cross validation for selecting an optimal k from $\{1, 3, 5, \dots, 25\}$, in the following way:

- ***Randomly** divide your training set into 10 folds, and use 10-fold cross validation, to estimate the performance of the k-NN classifier for every k. Output the k with the highest average accuracy, which we will call k_{best} in the following.* **NB!** *Only use the* training data *in the cross validation process.*
- *Please report the resulting optimal parameter k_{best}.*
- *After you have determined k_{best}, use it to estimate the performance of the resulting classifier on both the training set and the test set. Please report the two accuracies.*

e) Train the logistic regression on `DD_train.txt` and run it on both `DD_train.txt` and `DD_test.txt`. Threshold the returned probabilities at 0.5 *to obtain a binary classification, and compute the resulting accuracies. What do you see? Comparing to your cross-validated kNN classifier from the previous exercise, which classifier seems to work best on this dataset?*

f) Please implement data centering and normalization as a preprocessing step. Make sure that you estimate both the mean and coordinate-wise standard deviations using the training set only*, to avoid using the test set for anything but computing a test accuracy. Rerunning your classification from exercises c) and e) above with the preprocessed data, please report the accuracies and comment on what you observe.*

Deliverables. *a) Uploaded code both as snippet in the report and a filled out Jupyter template; the resulting plot; a short description/comment; b) Uploaded code both as a snippet in the report and a filled out Jupyter template; c) Training and test results of your k-NN classifier for $k = 1, 3, 5$;* 2-3 *lines of discussion of the results; d) Uploaded code both as a snippet in the report and a filled out Jupyter template; the optimal k, namely k_{best}, and the two final accuracies; e) Uploaded code both as snippet in the report and a filled out Jupyter template, the two accuracies, and a comment; f) Uploaded code both as snippet in the report and a filled out Jupyter template, resulting accuracies, and a comment.*

Appendix

The D&D dataset

The D&D dataset consists of 1178 protein structures [1] represented as graphs, from which *degree distributions* are extracted. Each protein is represented by a graph whose nodes are amino acids, which are connected by an edge if they are less than 6 Ångstrøms apart. Some of the proteins are enzymes and some are not; we shall train a classifier to predict whether or not an unseen protein is an enzyme.

The degree distribution, which we take as a vector representation of each graph, is generated in the following way: For each node in each graph, record its *degree* – that is, how many edges connect that node to other nodes in the graph. For each graph G, generate a count vector $g \in \mathbb{R}^d$, where the coordinate g_i counts how many nodes of degree i there were in the graph G.

You are given the *D&D* dataset split into a *training* set with 700 proteins, and a *test* set with 478 elements, along with their corresponding enzyme/non-enzyme classification labels.

References

[1] P.D. Dobson and A.J. Doig, *Distinguishing enzyme structures from non-enzymes without alignments.*, J. Mol. Biol., 330(4):771-783, 2003.

[2] A. Azzalini and A. W. Bowman. *A look at some data on the Old Faithful geyser*, Applied Statistics, 39, 357–365, 1990