MAD 2021/2022 Exam

Bulat Ibragimov, Kim Steenstrup Pedersen

Exam period: 17.01.2022 – 25.1.2022

This is the exam questions for the 8 day take-home exam on the course Modeling and Analysis of Data (MAD). 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**. You have to submit your solution of the exam electronically via the **Digital Exam**[1] system. The submission deadline is **25 January 2022 at 22:00**.
Your solution should consist of

1. a pdf file `answers.pdf` containing your answers, and
2. a `code.zip` file containing the associated code (python files and / or Python Jupyter notebooks).

The grading will be done anonymously; **hence you should not mention your name anywhere in the answers or code**. Instead, you must **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. Do not put yourself and your fellow students at risk.

You are allowed to ask questions via the Discussions board in Absalon, but make sure that you do not reveal any significant parts of the solution. In doubt, just contact Bulat or Kim directly via e-mail! Any additional hints 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, but you are allowed to convert a Jupyter notebook to a python file or the opposite. 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 if we ask you to implement a specific algorithm without using existing implementations (e.g., a single `kmeans` function from some other package that implements the K-means clustering approach). If in doubt, please ask us via the discussion board in Absalon!
4. All code templates and data can be found in the files `code.zip` and `data.zip`.
5. The deadline is hard (late submissions are not allowed), so make sure to submit in good time before the deadline. Up until the deadline it is possible to upload new versions of your solution several times. In the unlikely event that the Digital Exam system fails when you submit just at the deadline, then immediately send an e-mail to `uddannelse@diku.dk` and `bulat@di.ku.dk` with an explanation of what happened and attach your solution (the pdf and zip files) to the exam. Your solution will be assessed if you have a valid excuse and submitted on time.
6. Good luck! :-)

[1] `https://eksamen.ku.dk/`

Statistics

In this part, we will test your knowledge and skills in performing statistical analysis.

Question 1 (**Maximum Likelihood Estimation, 1 points).** Let X be a continuous random variable which is log-normal distributed, $X \sim \text{logn}(\mu, \sigma)$ with parameters $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}_+$ (the set of positive real numbers), with probability density function

$$f(x) = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} \cdot \frac{1}{x} \cdot \exp\left(-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2\right) & \text{for } x \in \mathbb{R}_+ \\ 0 & \text{otherwise.} \end{cases}$$

Assume we have a dataset $x_1, x_2, \ldots, x_n$ of n i.i.d. samples from X and that we know the parameter σ. Prove that the maximum likelihood estimate $\hat{\mu}$ for the parameter μ is given by

$$\hat{\mu} = \log\left(\sqrt[n]{x_1 \cdot x_2 \cdot \ldots \cdot x_n}\right) .$$

Deliverables. Provide the essential steps and argumentation in your proof.

Question 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 follows a binomial distribution, $X \sim \text{Bin}(n, \theta)$ 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 arguments) b) Provide the steps you follow in order to perform the test as well as your conclusion.

Principal component analysis

Question 3 (**Principal Component Analysis, 3 points).** The aim of this task is to test your understanding of the principal component analysis algorithm. You are given the following set of $N = 8$ 2-dimensional points:

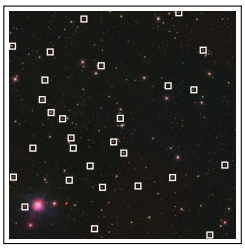
Points								
coordinate x	0.6	0.5	1.1	-0.5	0.8	0.2	-0.1	1
coordinate y	1.1	1	2	0.2	-0.1	-0.1	-1.5	2.5

Your task is to compute the principal component decomposition for these points. You can use either N or $N - 1$ in the denominator during calculation of the covariance matrix.

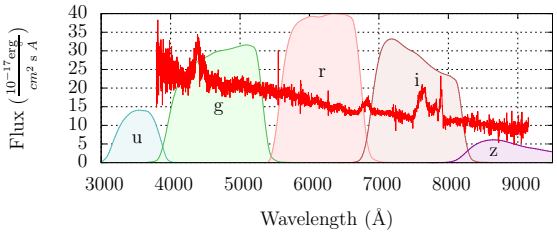
Deliverables. Step-by-step calculation of the principal components. Your report should include mathematical derivations of the following:

1. Mean point of the data.
2. Two eigenvalues.
3. Two eigenvectors.

(a) Observatory



(b) Image Data



(c) Spectra

Figure 1: The Apache Point Observatory shown in Figure (a) gathers both images and spectra. The image data are given in terms of multi-spectral images that are based on five filters covering different wavelength ranges, called the `u`, `g`, `r`, `i`, and `z` bands, see Figure (c). In Figure (b) an RGB image is shown that is based on such images of a particular region. For a small subset of detected objects (white squares), detailed follow-up observations in terms of spectra are available, see Figure (c). The dataset provided to you contains, for each such follow-up observations, the true redshift (obtained via the corresponding spectrum) as well as 10 features, which correspond to features extracted from the image data (so-called magnitudes). Your task is to build a regression model that can predict the redshift for all remaining objects in the image data.

Regression

Question 4 (Regression, 6 points). The aim of this task is to test your understanding of methods for regression. An important problem in astrophysics is to estimate the distance of objects to our Earth. Since one cannot directly measure these distances, one resorts to the light emitted by the objects and that reaches Earth. In particular, one considers the so-called *redshift* z of an object: The larger the redshift, the larger the distance is from an object to Earth. This redshift can be measured very accurately in case one has access to the detailed spectrum of the light for an object at hand. Unfortuntely, obtaining such spectra is very time-consuming and are, hence, only available for a small subset of the detected objects, see Figure 1. Instead, one has access to image data and one usually aims at estimating the redshift based on the image data only. **Note that the aforementioned physical details are not important for solving this exercise, but are provided to motivate the corresponding machine learning task.**

The task of computing the redshifts can be formalized as a regression problem for which you have access to a training set $\{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\}$ and a corresponding test set, which are stored in `galaxies_train.csv` and `galaxies_test.csv`, respectively. Each file contains, for each object (row), a target value $t_n \in \mathbb{R}$ (first column) and a vector $\mathbf{x}_n \in \mathbb{R}^{10}$ of $D = 10$ attributes (second to eleventh column). Your task is to build regression models that can be used to predict the target (redshift) for new, unseen objects!

1. *Nearest Neighbor Regression (4 points):* As discussed in L9, one can use nearest neighbours in the context of regression scenarios. More precisely, for a vector $\mathbf{x}$, the prediction is given via

$$f(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_n \in N_k(\mathbf{x})} t_n$$

where $N_k(\mathbf{x})$ denotes the set of the $k \geq 1$ nearest neighbors of $\mathbf{x}$ in the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of training points.

   (a) Extend the Jupyter notebook `Galaxies_kNN.ipynb` and implement nearest neighbor regression in Python for arbitrary numbers $k \geq 1$ of nearest neighbors; make use of the Eulidean distance $d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{D}(p_i - q_i)^2}$ for data points $\mathbf{p}, \mathbf{q} \in \mathbb{R}^D$. Use the training set to train the model using $k = 3$ neighbors. Afterwards, apply the model to the instances given in the test set. Compute the RMSE between the predictions made by the model for the test instances and the corresponding test labels. Also, visualize the model quality via a scatter plot ('true redshift' vs. 'predicted redshift').

   (b) As discussed during the lecture, one is not restricted to the Eulcidean distance only. For instance, one can make use of

$$d(\mathbf{p}, \mathbf{q}) = (\mathbf{p} - \mathbf{q})^T \mathbf{M} (\mathbf{p} - \mathbf{q})$$

   for points $\mathbf{p}, \mathbf{q} \in \mathbb{R}^D$, where $M \in \mathbb{R}^{D \times D}$ is a positive semidefinite matrix. Implement this variant for arbitrary positive semidefinite matrices $\mathbf{M}$. Afterwards, consider the following concrete example $\mathbf{M} = diag(c_i)$ with $c_i = 0.00001$ for $i = 1, \ldots, 8$ and $c_i = 1.0$ for $i = 9, 10$ (i.e., a diagonal matrix with values c_i on the diagonal). Again, train the model using $k = 3$ and apply it to the test instances afterwards. Generate another scatter plot and report the RMSE. Can you explain what this particular matrix M does?

2. *Non-Linear Least-Squares via Neighbors (2 points):* We have seen how one can obtain non-linear least-squares 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 $\mathbf{x}_n$ with associated targets t_n, one can compute, for **each** new point $\bar{\mathbf{x}} \in \mathbb{R}^D$, a prediction via the following three steps:

   - Step 1: Compute the k nearest neighbors $N_k(\bar{\mathbf{x}})$ of $\bar{\mathbf{x}}$ in T (make use of the Euclidean distance).
   - Step 2: Fit a regularized linear least-squares regression model using these k nearest neighbors and the associated targets, i.e., compute:

$$\hat{\mathbf{w}}(\bar{\mathbf{x}}) = \text{argmin}_{\mathbf{w}} \frac{1}{k} \sum_{\mathbf{x}_n \in N_k(\bar{\mathbf{x}})} (f(\mathbf{x}_n; \mathbf{w}) - t_n)^2 + \lambda \sum_{i=0}^{D} w_i^2$$

     Here, $f(\mathbf{z}; \mathbf{w}) = w_0 + w_1 z_1 + \ldots + w_D z_D$ depicts the linear model and $\lambda > 0$ a regularisation parameter.
   - Step 3: Use this linear model to obtain a prediction $f(\bar{\mathbf{x}}; \hat{\mathbf{w}}(\bar{\mathbf{x}}))$ for $\bar{\mathbf{x}}$.

   Extend the Jupyter notebook `Galaxies_LinRegNeighbors.ipynb` and implement this regression model in Python. Similarly to the the nearest neighbor model, make use of the training set to fit the model and apply it to the test set. Use $k = 15$ nearest neighbors and $\lambda = 1.0$. Again, visualize the model quality via a scatter plot. What is the induced RMSE value?

Comments: You are supposed to implement both parts on your own, i.e., you are not allowed to use, e.g., corresponding class/functions from the *Scikit-Learn* package. Note that you are allowed to make use of the *Numpy* package. In doubt, please get in touch with us.

Deliverables. All your source code used to solve this exercise (as a Jupyter notebook or Python script file(s)). Add to your report:

1. (a) RMSE value and scatter plot and (b) RMSE value, scatter plot, short explanation for what M does (2-3 lines).
2. RMSE value and scatter plot

Classification

In this part, we will test your knowledge and skills in performing classification of data.

Question 5 (Random Forests, 2 points). This question tests your understanding of random forest construction, in particular the selection of the optimal threshold value. Let's say you have a training set of the following samples:

Points												
feature	10	20	30	40	50	60	70	80	90	100	110	120
label	0	0	1	1	1	1	0	0	0	1	0	0

Each data sample is defined with one feature and has a binary label. Your aim is to compute the optimal threshold for separating this data into two subsets using a selected metric (you can select either information gain based on entropy or Gini index). You are asked to calculate all the steps manually and give explanations in words when appropriate:

a) Correct use of information gain based on entropy or Gini index
b) Correct identification of potential thresholds to consider
c) Calculating the metric values for the potential thresholds and finding the optimal threshold.

Deliverables. For a) and c), include your calculations. For b), write an explanation of which thresholds are worth considering. Do not forget for each threshold to explicitly write the metric values.

Question 6 (Classification & Validation, 4 points). This task aims to test your understanding on the training/validation data separation and classifier parameter tuning. Use training data from `accent-mfcc-data_shuffled_train.txt` and validation data from `accent-mfcc-data_shuffled_validation.txt`. The data corresponds to samples of human speech performed for individuals with different mother languages,

e.g. English, Spanish, German etc. Both files are in comma- separated format and can be read by modifying the `loaddata` function from `regression.py` to accommedate the columns of these files. Each line in the file represents one speech sample. The first value of a line is the mother language label. The remaining 12 values denoted X1-X12 are numerical features of the speech sample.

Use the implementation of Random forests from the `sklearn` Python package to do the following:

a) (1 point) Implement random forest training using the data from `accent-mfcc-data_shuffled_train.txt`. Use features X1-X12 to predict the "language" label. Report the accuracy of the random forest model on the training data. The accuracy is the number of correctly predicted languages divided to the total number of samples (accuracy lies between 0 to 1).

b) (2 points) Use the validation data to find the optimal set of random forest classifier parameters. The parameters to test: 1) criterion (entropy or gini); 2) maximal tree depth (values to test 2, 5, 7, 10, 15); 3) the number of features to consider for each split (*sqrt* and log_2 of the total number of features). To find the optimal set of parameters, perform an exhaustive search over all possible combinations of parameters from 1), 2) and 3). Calculate and print two metrics during the optimal parameter search. The first metric is the number of correctly classified validation samples. The second metric is the average probability assigned to the correct class for all validation samples. Use the first metric to select the optimal set of parameters. If two sets of parameters have the same value of correctly classified validation samples, i.e. the first metric is the same for both sets, the set with the highest second metric is considered more optimal. Add into report the code fragment that calculates both metrics and updates the optimal parameter set if these metrics are superior to the metrics for the previously found optimal set.

c) (1 point) Add into the report the accuracy improvements during the optimal parameter search. Every time you find a more optimal set of parameters, print a statement "criterion = ? ; max_depth = ? ; max_features = ? ; accuracy on validation data = ? ; number of correctly classified validation samples = ?". Replace question marks with the appropriate values.

Deliverables. a) Provide a code snippet in the report of the essential steps, b) code snippet in the report of the essential steps, c) provide the obtained results in the report as well as your reflections on these.

Clustering

In this part, we will test your knowledge and skills in performing clustering of data.

Question 7 (K-means Clustering & Principal Component Analysis, 7 points). This task aims to test your understanding of clustering and principal component analysis. The task includes the following:

a) (1 point) Read and normalize data from the comma-separated data file `seedsDataset.txt`. You can read the file by modifying the `loaddata` function from `regression.py` to accommodate the columns of this file. The dataset consist of 200 samples. Each line in the file represent one sample. Each sample is defined with 8 features X1-X8. Normalize the data using the mean and standard deviation values computed for each feature. That is, for the i'th sample, the normalized X1 feature value is $X1_i^{norm} = (X1_i - \bar{X1})/(\sigma_{X1})$, where $X1_i$ is the original X1 feature value for the i'th sample, $\bar{X1}$ is the mean value of feature X1 computed from all samples, and σ_{X1} is the standard deviation of feature X1 computed from all samples.

b) (2.5 points) Implement K-means clustering without using any existing implementation. Test your K-means implementation using the normalized data from a) and set number of clusters to $K = 3$. Run K-means 5 times using random samples from the dataset as the initial cluster centers. Select the solution with the smallest intra-cluster distance.

c) (0.5 point) Compute and print the number of samples in each cluster in the final solution you obtained in b).

d) (1 point) Compute the principal components of the data. You are allowed to use existing implementations of the principal component analysis.

e) (2 points) Transform the data using the two largest components, i.e. eigenvectors with the largest eigenvalues. Plot the clustering results including the transformed cluster centers and transformed data. Your plot should contain 200 colored dots representing data samples and 3 black dots representing cluster centers. The dot associated with i'th sample should have coordinates $(y, z) = PCA_2(X1_i, X2_i, ..., X8_i)$, where PCA_2 is the transformation defined by projection onto the two largest components, and the color defined by the cluster label of i'th sample obtained in b). In figure 2, you find an example of such a plot. Note that you should run clustering on the original 8-dimensional data, save the clustering labels in some array, transform the data to 2-dimensional space and then plot the data samples colored according to the saved clustering labels.

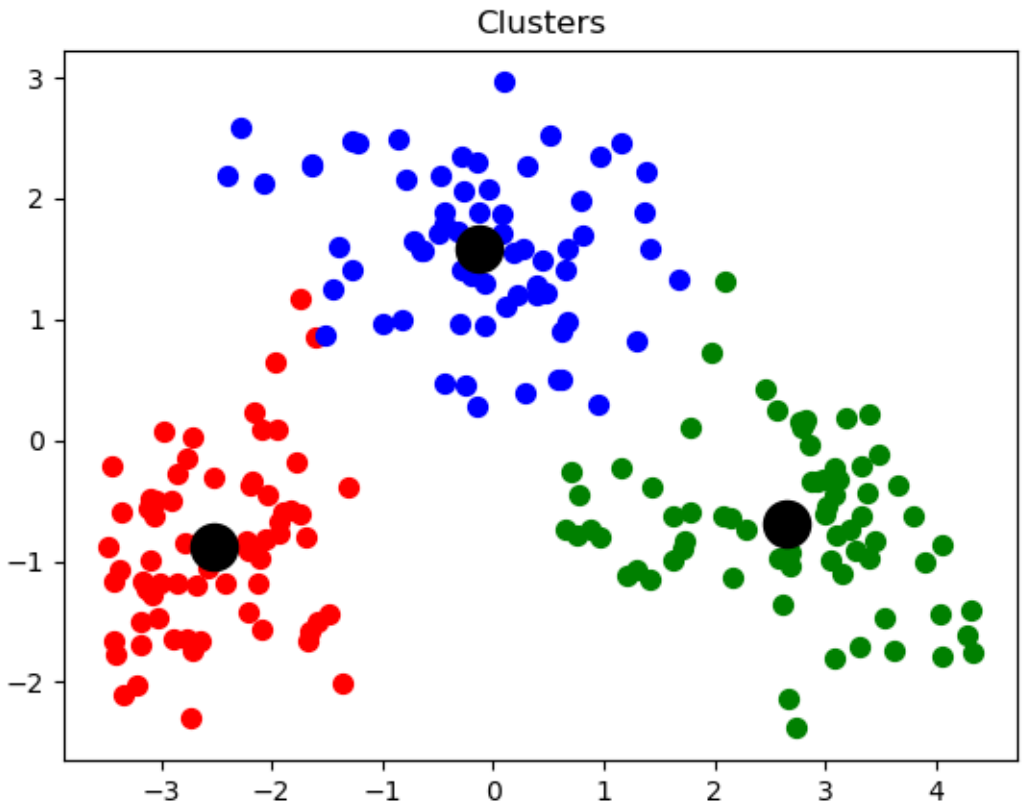


Figure 2: Example of plot we ask for in question 7(e). The black dots represents the cluster centers.

Deliverables. a) Provide a code snippet in the report showing how you perform the normalization, b) provide a code snippet in the report showing your K-means implementation, c) include the number of samples in each cluster in the report, d) provide a code snippet and explanation of what you did in the report, e) include a code snippet, the plot and your reflections on the results in the report.