02450 Introduction to Machine Learning and Data Mining

DTU Learn: If you are enrolled in the course you can access material and participate in the course through the DTU learn homepage

Lectures: The lectures will take place Tuesdays from 13:00-15:00. Due to coronavirus the lectures will be online in the first few weeks of the semester. Information about streaming and access will be available on DTU Learn.

Exercises: Exercises will take place after lectures Tuesdays from 15:00-17:00. Format will be subject to change as dictated by the coronavirus situation. As of right now, exercises will be held on Microsoft teams

- **Exercise team Turing, (Python):** Rune Dodensig Kjærsgaard: *rdokj@dtu.dk*, Jonas Søbro Christophersen: *s153232@student.dtu.dk*, Rasmus Hannibal Tirsgaard: *s174511@student.dtu.dk*
- **Exercise team Neumann, (Python):** Alison Marie Sandrine Pouplin: *alpu@dtu.dk*, Peter Mørch Groth: *s164049@student.dtu.dk*, Oldouz Majidi: *s163502@student.dtu.dk*
- **Exercise team Minsky, (R+Python):** Oliver Kinch Hermansen: *s183980@student.dtu.dk*, Qahir Siavash Yousefi: *s164180@student.dtu.dk*, Nikolaos Nakis: *nnak@dtu.dk*
- **Exercise team Bayes, (Matlab+Python):** Alvaro Carrera Cardeli: *s172254@student.dtu.dk*, Fabian Martin Mager: *s190212@student.dtu.dk*, David Ribberholt Ipsen: *s164522@student.dtu.dk*
- **Exercise team Rosenblatt, (Python):** Jorge Sintes Fernandez: *s202581@student.dtu.dk*, Michael Rahbek: *s164035@student.dtu.dk*

Reading material, lecture slides and exercises: The course will use lecture notes and other freely available material. Lecture notes, slides, course assignment instructions etc. is available on DTU learn.

Online Demos: We have developed several online demos which illustrates key concepts from the course. The topics discussed currently includes PCA, regression, classification and density estimation.

Discussion forum: For online help and discussion see the DTU Learn discussion forum.

Teachers :

Tue Herlau (TH): `tuhe@dtu.dk`

Lecture schedule:

No	Date		Subject	Reading	Homework
1	2 February, 2021	TH	Introduction	**C1**	
			Data: Feature extraction, and visualization		
2	9 February, 2021	TH	Data, feature extraction and PCA	**C2, C3**	P3.1, P2.1, P3.2
3	16 February, 2021	TH	Measures of similarity, summary statistics and probabilities	**C4, C5**	P4.1, P4.2, P4.3
4	23 February, 2021	TH	Probability densities and data visualization	**C6, C7**	P6.1, P6.2, P7.1
			Supervised learning: Classification and regression		
5	2 March, 2021	TH	Decision trees and linear regression	**C8, C9**	P9.1, P8.1, P8.2
6	9 March, 2021	TH	Overfitting, cross-validation and Nearest Neighbor **(Project 1 due before 13:00**	**C10, C12**	P10.1, P10.2, P12.1
7	16 March, 2021	TH	Performance evaluation, Bayes, and Naive Bayes	**C11, C13**	P13.1, 13.2, P12.2
8	23 March, 2021	TH	Artificial Neural Networks and Bias/Variance	**C14, C15**	P15.1, P15.2, P15.3
			Holiday		
9	6 April, 2021	TH	AUC and ensemble methods	**C16, C17**	P16.1, P16.2, P17.1
			Unsupervised learning: Clustering and density estimation		
10	13 April, 2021	TH	K-means and hierarchical clustering	**C18**	P18.1, P18.2, P18.3
11	20 April, 2021	TH	Mixture models and density estimation **(Project 2 due before 13:00**	**C19, C20**	P20.1, P19.1, P19.2
12	27 April, 2021	TH	Association mining	**C21**	P21.1, P18.2, P18.3
			Recap		
13	4 May, 2021	TH	Recap and discussion of the exam	**C1-C21**	

Cx refers to Chapter **x** of the course notes. **Px.y** refers to problem number **y** in chapter **x** of the course notes. The first listed problem will be that weeks discussion question at the exercises.

FAQ:

- **I got a question about a report/lecture topic. Can I send you an email/visit your office?**
  Questions about course content/reports are best asked on the discussion forum so all students have access to the same information.

- **I have a question relating to the course regarding a personal matter. Who should I contact?**
  Please send me an email at tuhe@dtu.dk

- **It is one week into the semester course but I would like to join it! Can you add me?**
  I can (and will!) add you to the group so you can see the material, but to join the course (and be at the exam) you have to enroll. You can do this through your DTU study planner. If the study planner does not work, you have to contact the study administration by email. My default position is to approve all new students. Always check your exam registration when they are published.

- **Can I get my reports trasferred from a previous semester?**
  Yes. Old reports are automatically transferred by default. See description for project 1. Please do not upload old reports anew unless you have changed the content

- **How do I know I get credit for my reports?**
  If you are missing a report, I will send you an email to your s123456@student.dtu.dk shortly after the exam. No email, no problem.

- **What is my grade in the report?**
  Reports are not graded but evaluated. By the DTU rules we cannot give you a numerical score. However, the feedback from TA's should provide a good indication.

- **When do I get feedback on the reports?**
  Under normal circumstances, feedback should be available about 3 weeks after you handed in project 1, and 2 weeks after you handed in project 2.

- **I still haven't gotten report feedback?**
  If feedback is delayed more than a day or two please send me an email.

- **Is my report "passed"/"approved"?**
  Reports are not passed/not passed. Even a very poor report is better than not handing in.

- **The report handin is tomorrow and I don't have a group/my other team members are not responding to emails. Can I get an extension?**
  No. Some students complete their projects on their own; if you don't have a group, or things are going south with your group, you have to fix it within the deadline.

- **How do I find a group?**
  Please use the discussion forum to find team members.

- **Can I do the projects alone?**
  Only after explicit permission from me due to resource constraints.

- **I think I will hand in my report an hour or two after the deadline. What could go wrong?**
  We are reasonable if you are handing in late due to extraordinary circumstances, but otherwise we take the deadline serious. It would not be fair to all those students who meet the deadline if we accepted late handins without consequences.

- **Can I bring a computer to the exam?**
  Yes, exam is "all aids allowed". See study handbook for details. We strongly recommend you bring a computer to the exam with the typical course software.

- **When is the re-exam?**
  In the spring semester there is a re-exam in August. For the fall semester the next exam is in the following spring semester.

- **I cannot log onto DTU Learn/inside or a specific functionality on the sites appear to be broken**
  If you encounter IT problems and it appears clear something is wrong with a page or your account please submit a ticket to the (usually very responsive) IT support: https://itservice.ait.dtu.dk

- **Can I be a TA?**
  Yes, after the course has completed, I will make a call for teaching assistants. Pay is more than 200DKR per hour and there is preperation time!

Introduction to Python

Objective: This exercise is an *optional* pre-exercise. You can use the exercise to check that you have the sufficient coding skills, as well as ensuring that you have your integrated development environment (IDE) setup.

The objective is to setup your Python IDE. Additionally, the objective is to get you acquainted with Python, as well as the exercise format of the course. Upon completing this exercise it is expected that you:

- Have setup your Python IDE.
- Can write and execute basic Python code.
- Understand how data can be represented as vectors and matrices in numerical Python (NumPy)
- Understand basic operations and plotting in Python.
- Understand the format of the exercises.

Material: For this exercise, you will need the 02450 toolbox. See the optional section in the end of the document for more material and tutorials on coding.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

0.1 How to do the exercises

The exercises in the course are structured such that you go through an exercise document like this one.

This exercise (Exercise 0) is an optional exercise that you can use to make sure have the sufficient coding skills and have setup the your IDE—that is: the exercise is not introducing any course material, and is meant as a help in preparing for starting with the course and ensuring that you fulfill the course requirements. If you encounter any problems in doing this exercise—in setting up your IDE or installing the 02450 Toolbox—make sure to get in touch with a teaching assistant after the first lecture at the exercises. The first real course exercise (Exercise 1) will be done after the first lecture. Note that if you have difficulties following the scripts, catching up on your

progamming skills are to be done as a self-study exercise (see optional material in the end of this document).

The exercises are centered around running and understanding a series of scripts provided in the 02450 Toolbox. The exercise descriptions guide you through the scripts and help you understand what is going on in them. You wont have time to code everything from scratch for the exercises, and so it is important that you get familar with this workflow centered around the existing scripts. The scripts that you go through in the exercises provide the basis for your work the project you will work on, where you will be able to re-use large parts of the code you have worked with in the exercises. However, for the reports you will have to code more yourself and tailor the scripts to your dataset and problem.

The exercises are structured as smaller numbered sections. When a certain section concerns a particular script, it will be stated and their number will match. For instance, the first script you will run (in a little while) is called `ex0_4_3.py` and corresponds to the section 0.4.3 in this document.

0.2 Getting started with Python

We strongly recommend installing the **Anaconda** Python distribution. To install Anaconda, go to `https://www.anaconda.com/download/` and download the most recent version for your operating system. If asked, ensure you select the latest python version in the Anaconda installer and accept the default settings.

Anaconda will amongst other things install Spyder, which offers many tools characteristic for mature IDEs, and thus will enhance your programming experience. These include: context help, text completion, syntax highlighting, enhanced editor, integrated interactive console, project navigator and more. Working with Spyder is quite similar to Matlab environment. See: `spyder-ide.org`. In the following it will be assumed Spyder is used to run python commands and edit Python files. To install other packages, note that you can open the **"Conda Command Prompt"** and use `easy_install <packagename>`.

0.3 Installing the 02450 Toolbox

The course will make use of several specialized scripts and toolboxes not included with Python. These are distributed as a toolbox which need to be installed.

0.3.1 Download and unzip the 02450 Toolbox for Python, `02450Toolbox_Python.zip`. It will be assumed the toolbox is unpacked to create the directories:

```
<base-dir>/02450Toolbox_Python/Tools/     # Misc. tools and packages
<base-dir>/02450Toolbox_Python/Data/      # Datasets directory
<base-dir>/02450Toolbox_Python/Scripts/   # Scripts for exercises
```

For the exercises, you should work on the example scripts in `<base-dir>/02450Toolbox_Python/Scripts/` (notice the scripts are labelled according to exercise number) and not try to write the scripts from the bottom up.

0.3.2 To finalize the installation you need to update your path. In Spyder, go to `Tools -> PYTHONPATH Manager` and press `Add path`. Navigate to `<base-dir>/02450Toolbox_Python/Tools/` and press `Select folder`. Restart Spyder for changes to take effect. Test your path by typing:

```
import toolbox_02450
print(toolbox_02450.__version__)
```

If a revision with a date is printed the path to the toolbox is correctly set up.

0.4 Basic operations in Python

0.4.1 *Console* The IDE provides both a console and an editor. The console can be used to execute code fast and interactively, and is often used when debugging your code. You can define and display variables, plot data, and even define functions on-the-fly using interactive console. It is useful when you debug your code or experiment with small chunks of code, mathematical expressions, etc. In Spyder, find the IPython Console (the console) and try typing: `a=9`. The variable `a` should then appear in your Variable explorer If you write `a` in the console, the value of `a` is displayed.

0.4.2 *Editor, scripts and functions* The editor is used to write longer scripts and functions. In Spyder, write the following lines in the script in the editor:

```
a = 3
b = 4
print(a*b)
```

and save the file as `myscript.py`. Run the script from the console by typing `import myscript` in the console and observe the results. The current file can be run in Spyder by pressing `F5` (or the green arrow), the current line or selection can be run by pressing `F9`. You can clear all variables by pressing eraser-like icon in the righthand corner of the console. You can interrupt a running script by pressing `CTRL+C`. The Python kernel can be restarted by pressing `CTRL + .` in the console (needed e.g. if it has crashed).

In many cases you will rather write scripts, and use console only as a supporting/debugging tool. Python script is a file with `.py` extension, which contains instructions, functions, class definitions. Try to write the following code in the Spyder editor, and save it as `mymodule.py` (notice that tabs are important after the first line):

```
def myfunction(n=int):
  """myfunction will return an array of values ranging
  from 1 to the input integer n."""
  x = range(1,n+1)
  return list(x)
```

Note that the triple quotes sign `"""` indicates multiline comments. To use the function, you import the function from the module. You can do this in your script, but also from the console like: `from mymodule import myfunction`. You can then use the function by writing `y = myfunction(9)`. Try writing `help(myfunction)` and see the result. Also, try calling the function a non-integer value and see the result (e.g. `'test'` or `3.1` instead of `9`).

You can get help in your Python interpreter by typing `help(obj)` or you can explore source code by typing `source(obj)`, where `obj` is replaced with the name of function, class or object.
Furthermore, you get context help in Spyder after typing function name or namespace of interest. In practice, the fastest and easiest way to get help in Python is often to simply Google your problem. For instance: `"How to add legends to a plot in Python"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

0.4.3 *Making sequences* We often need to use a sequence of numbers. Observe the various results obtained when running `ex0_4_3.py`.

0.4.4 *Indexing* We often need to retrieve or assign a value to a certain part of a vector or matrix. Inspect `ex0_4_4.py` to see how to do indexing in Python.

0.4.5 *Matrix operations* We will use various matrix operations extensively throughout the course. Inspect `ex0_4_5.py` to see how to do some common ones in Python.

0.5 Basic plotting

It is important to visualize the results you obtain. In this part of the exercise, we will make two plots, a simple, and a more elaborate example. Going forwards, try to keep the elements of the figure made in 1.5.2 in mind as a model for minimum required considerations when making a figure.

0.5.1 Inspect `ex0_5_1.py` to see how to do basic plotting in Python. Try modifying the script to display a cosine curve for values in the range $[0, \pi]$.

0.5.2 Now, let us consider a slightly more advanced plot. Inspect `ex0_5_2.py`. We simulate measurements from two sensors (sensor 1 and sensor 2) for a period of 10 seconds, and we say that the sensors output measurents in millivolt. Since the two measurements are done at the same time, we want to do plot them in the same figure. Furthermore, we want to make sure that the axis are labelled correctly both with a name and a designation of the unit of measurement. We also need to make sure that it is easy to see which curves comes from which sensor. Lastly, we ensure that the axis are readable (large enough), both when looking at them in the IDE, but also in an exported version that we might use in a report about the sensors.

0.6 OPTIONAL

The following online materials are recommended:

- `docs.python.org/tutorial` - Introduction into python environment, syntax and data structures. Recommended reading - sections 1, 2, 3, 4 and 5.
- `docs.scipy.org/doc/numpy-1.15.1/user/quickstart.html` - Tutorial introducing the scientific computing in Python, array and matrix operations, indexing and slicing matrices.

- docs.scipy.org/doc/numpy-1.15.0/user/numpy-for-matlab-users.html - Useful reference to scientific computing in Python if you have previous experience with Matlab programming.
- matplotlib.sourceforge.net - Documentation and examples related to matplotlib module, which we shall use extensively through the course to visualize data and results.
- spyder-ide.org - Overview of Spyder IDE for Python code editing/debugging.
- https://www.youtube.com/playlist?list=PL6gx4Cwl9DGAcbMi1sH6oAMk4JHw91mC_ - Series of video tutorials covering basics of Python programming.

Especially the Python tutorial (sections 1-5) and NumPy tutorial are recommended. The more you get acquainted with Python now, the easier it will be for you to solve machine learning problems in the following weeks. You will benefit from this course most if you try to implement the solutions on your own, before checking the correct answers. Nevertheless, if you run into problems, the guidelines and the correct scripts will be always provided for your reference.

References

Datasets and handling data in Python

Objective: The objective is to you acquainted with the exercise format of the course. Additionally, the aim is to familiarize yourself with the standard dataformat used in the course. Upon completing this exercise it is expected that you:

- Understand the format of the exercises and how the exercises are related to the reports.
- Can import data into Python and represent the data in course $\mathbf{X}$, $\mathbf{y}$ format.
- Can do common preprocessing steps for datasets.
- Have selected a proper dataset for use in the your project work (for the reports).

Material:

PYTHON Help: You can get help in your Python interpreter by typing `help(obj)` or you can explore source code by typing `source(obj)`, where `obj` is replaced with the name of function, class or object.
Furthermore, you get context help in Spyder after typing function name or namespace of interest. In practice, the fastest and easiest way to get help in Python is often to simply Google your problem. For instance: `"How to add legends to a plot in Python"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

1.1 How to do the exercises

During exercises in the course, you will go through an exercise document like this one. The exercises are centred around running and understanding a series of scripts provided in the 02450 Toolbox. The exercise descriptions guide you through the scripts and note that you won't have time to code everything from scratch. The scripts are also the basis for your work in the reports, where you will be able to re-use large parts of the code. However, for the reports, you will tailor the scripts to your dataset and problem.

The exercises are structured as smaller numbered sections. When a certain section concerns a particular script, it will be stated and their number will match. For instance, the first script you will run (in a little while) is called `ex1_5_1.py` and corresponds to the section 1.5.1 in this document.

1.2 Getting started with Python

We assume that you have a working Python IDE set up. If that is not the case, complete the pre-exercise (Exercise 0) before proceeding. If you have already done the optional Exercise 0, you can skip the next section ("Installing the 02450 Toolbox").

We will assume you have the most recent **Anaconda** Python distribution (Python 3.6). Additionally, in the following, it will be assumed Spyder is used to run python commands and edit Python files.

1.3 Installing the 02450 Toolbox

The course will make use of several specialized scripts and toolboxes not included with Python. These are distributed as a toolbox which need to be installed.

1.3.1 Download and unzip the 02450 Toolbox for Python, `02450Toolbox_Python.zip` . It will be assumed the toolbox is unpacked to create the directories:

```
<base-dir>/02450Toolbox_Python/Tools/     # Misc. tools and packages
<base-dir>/02450Toolbox_Python/Data/      # Datasets directory
<base-dir>/02450Toolbox_Python/Scripts/   # Scripts for exercises
```

For the exercises, you should work on the example scripts in `<base-dir>/02450Toolbox_Python/Scripts/` (notice the scripts are labelled according to exercise number) and not try to write the scripts from the bottom up.

1.3.2 To finalize the installation you need to update your path. In Spyder, go to `Tools -> PYTHONPATH Manager` and press `Add path`. Navigate to `<base-dir>/02450Toolbox_Python/Tools/` and press `Select folder`. Restart Spyder for changes to take effect. Test your path by typing:

```
import toolbox_02450
print(toolbox_02450.__version__)
```

If a revision with a date is printed the path to the toolbox is correctly set up.

1.4 Representation of data in Python

We will use a standard data representation throughout the course. Using this representation makes it easy to apply the various tools in the 02450 Toolbox on a new dataset. Once you have a given dataset in the standard format, the scripts will all be set up to work with it correctly.

An overview of the format is presented for Python in this table:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
Classification	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
Classification	`C`	integer	Scalar	Number of classes.

1.5 Loading data

Before we can begin to do machine learning, we need to load the data. Datasets are distributed as various types of files, and a few common ones will be shown here for future reference to be used once you have to load your own dataset.

Once we have loaded a dataset, we often need to process the data before the format fits our needs. For this course, in particular, this mostly means putting the dataset in the $\mathbf{X}$, $\mathbf{y}$-format shown above. The machine learning algorithms you will use needs the data to be in a numerical format, so we will also go through how to convert data which is in a text format into a numerical format.

Lastly, we will also go through a few tasks that often need to be handled before we can load some dataset.

For today's exercises you need to add a package to your Python environment. The additional package is for importing data from excel spreadsheets. Please make sure that you have installed the following packages (you can follow the guidelines at the corresponding websites):

- Excel file data extraction (xlrd package):
  `https://xlrd.readthedocs.io`

The websites provide documentation of the packages. Note if you use the Anaconda Python distribution these packages may already be added, use `conda list` in the terminal for a list of installed packages.

For illustrating loading data, we will consider the Iris flower dataset, which we will also return to later on. The Iris flower dataset or Fisher's Iris dataset is a multivariate dataset introduced by Sir Ronald Aylmer Fisher (1936) for the problem of classifying Iris flower types. It is sometimes called Anderson's Iris dataset because Edgar Anderson collected the data to quantify the geographic variation of Iris flowers in the Gaspé Peninsula. The dataset consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four variables were measured from each sample, they are the length and the width of sepal and petal, in centimetres. Based on the combination of the four variables, Fisher developed a linear discriminant model to distinguish the species from each other. It is used as a typical test for many other classification techniques (see also `http://en.wikipedia.org/wiki/Iris_flower_data_set`). The data has been downloaded from `http://archive.ics.uci.edu/ml/datasets/Iris`.

The perhaps most common and simple format of storing data is the comma-separated values-file format (or CSV). In such files, the data is stored such that a sample or an observation is a line in a text document, and the document then has as many lines (or rows) as there are samples. The attribute values for an observation is written within one line, separated by (usually) a comma or a tab-character in a consistent order. This order is usually defined in a header (the first line of the file), which has a designation of the variable name in some format.

1.5.1 Inspect the file

`<base-dir>/02450Toolbox_Python/Data/iris.csv`

using a simple text editor (e.g. for Windows "Notepad" or for MacOS "TextEdit"). Afterwards, inspect the script `ex1_5_1.py` to see how to load the Iris data from a CSV-file and put it into the standard format. Since the class label (the flower species) are stored as text (or strings), we convert them into a numerical value.

1.5.2 Sometimes datasets are distributed as Excel-files (`.xls(x)`). Inspect the script `ex1_5_2.py` to see how to load the same Iris data, when it has been stored as an Excel-file (open

`<base-dir>/02450Toolbox_Python/Data/iris.xls`

to have a look at the file).

1.5.3 Other times, and especially in this course, data is stored as MATLAB files (`.mat`). Inspect `ex1_5_3.py` to load the Iris data from

`<base-dir>/02450Toolbox_Python/Data/iris.mat` .

1.5.4 In the examples up until now, we have handled the data in the Iris dataset as if to solve a classification problem. We could say that the *primary* machine learning modelling aim is to classify the species of Iris flower based on the petal and sepal dimensions. However, we could also use the dataset to illustrate how to do regression without needing to use a whole different dataset. We would achieve this by e.g. trying to predict either of the petal (or sepal) dimensions based on the remaining dimensions, for instance. This changes how we define our $\mathbf{X}$,$\mathbf{y}$-format. Inspect `ex1_5_4.py` to see how to cast the Iris dataset into a regression problem. In the script, we will set up the $\mathbf{X}$,$\mathbf{y}$-format such that we are predicting the petal lengths form the other available information. Notice that we change how we use the information of the class label from before (the species information). Instead of storing it as a single variable, we have now used a "one-out-of-K encoding", since it is a categorical variable—we will return to various types of variables when we go through chapter 2 in the book (where one-out-of-K-encodings are described in section 2.4.1).

1.5.5 While the Iris dataset is a real dataset, it is a very clean and easy to work with dataset. Usually, data is a bit messier, and we will consider a toy dataset that has some common issues. Often, the description of "real-world" data is stored along with the data in some form of a text file. Have a look at the folder

`<base-dir>/02450Toolbox_Python/Data/messy_data/`,

and read more about the toy dataset—notice that there is a `README.txt` file.

Inspect the data in `messy_data.data` and try to identify some issues (use e.g. simple text editor as before). Afterwards, inspect `ex1_5_5.py` to see how the dataset can be stored in the desired representation.

1.6 Select a dataset for the reports

The course will include two written group reports. The two reports will cover the first two sections of the course:

- Data: Feature extraction, and visualization
- Supervised learning: Classification and regression

Each group will find one dataset they will use for the reports. This can either be your own dataset or a dataset you find yourself. Additional details about where you can find a dataset and formal requirements can be found in the **Finding a dataset for the reports** `.pdf` file on DTU Learn.

As a guideline, your dataset should have at least 60 observations and 5 attributes with at least two of the attributes being interval or ratio. Once you have found a dataset you need to have the dataset approved by one of the teaching assistants of the course.

As part of project 1, you will have to think about how various machine-learning tasks can be carried out for the dataset chosen and this exercise will briefly touch upon this. Discuss the following questions:

1.6.1 What is the problem of interest? (describe what your data is about in general terms, i.e. "to someone who knows nothing of machine learning")

1.6.2 Who made the data and why? Did they, or somebody else, work with the data and report results? If so, what were their results?

1.6.3 What is the *primary* machine learning modelling aim? (Is it *primarily* a classification, a regression, a clustering, an association mining, or an anomaly detection problem?)

1.6.4 Which attributes are relevant when carrying out a classification, a regression, a clustering, an association mining, and an anomaly detection? Specifically: Which attribute do you wish to explain in the regression based on which other attributes? Which class label will you predict based on which other attributes in the classification task?

1.6.5 Are there any data issues? Either directly reported in the accompanying dataset description or apparent by inspection of the data? (such as missing values or incorrect/corrupted values)

1.7 Tasks for the report

You are now able to address the following tasks for the report

1. **A description of your data set.**
   - Explain what your data is about. I.e. what is the overall problem of interest?
   - Provide a reference to where you obtained the data.
   - Summarize previous analysis of the data. (i.e. go through one or two of the original source papers and read what they did to the data and summarize their results).
   - You will be asked to apply (1) classification and (2) regression on your data in the next report. For now, we want you to consider how this should be done. Therefore:
     Explain, in the context of your problem of interest, what you hope to accomplish/learn from the data using these techniques?.
     Explain which attribute you wish to predict in the regression based on which other attributes? Which class label will you predict based on which other attributes in the classification task?
     If you need to transform the data in order to carry out these tasks, explain roughly how you plan to do this.

One of these tasks (1)–(5) is likely more relevant than the rest and will be denoted the **main machine learning aim** in the following. The purpose of the following questions, which asks you to describe/visualize the data, is to allow you to reflect on the feasibility of this task.

References

Principal Component Analysis with PYTHON

Objective: To get acquainted with how data can be filtered and visualized using principal component analysis (PCA). Upon completing this exercise it is expected that you:

- Can apply and interpret principal component analysis (PCA) for data visualization.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 2 folder available from Campusnet.

PYTHON Help: You can get help in your Python interpreter by typing `help(obj)` or you can explore source code by typing `source(obj)`, where `obj` is replaced with the name of function, class or object.
Furthermore, you get context help in Spyder after typing function name or namespace of interest. In practice, the fastest and easiest way to get help in Python is often to simply Google your problem. For instance: `"How to add legends to a plot in Python"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` For today's exercises you need to a package to your Python environment. The additional package is a machine learning toolkit for the last exercise (today optionally, but we shall need it in the following weeks). Please make sure that you have installed the following package (you can follow the guidelines at the corresponding websites):

- Machine learning toolkit (scikit-learn) - large package implementing various ML methods for supervised and unsupervised learning:
  `http://scikit-learn.org/stable/install.html`

The websites provide documentation of the packages. Note if you use the Anaconda Python distribution these packages may already be added, use `conda list` in the terminal for a list of installed packages.

Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
Classification	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
Classification	`C`	integer	Scalar	Number of classes.

2.1 PCA on the Nanose dataset

As an example dataset we will consider chemical sensor data obtained from the NanoNose [1] project, see also [2]. The data contains 8 sensors named by the letters *A–H* measuring different levels of concentration of Water, Ethanol, Acetone, Heptane and Pentanol injected into a small gas chamber. The data will be represented in matrix form such that each row contains the 8 sensors measurements (i.e. sensor A-H) of the various compounds injected into the gas chamber.

2.1.1 Inspect the file `<base-dir>/02450Toolbox_Python/Data/nanonose.xls` and make sure you understand how the data is stored in Excel. We will load the Nanose dataset from the file `<base-dir>/02450Toolbox_Python/Data/nanonose.xls` into into Python using the `xlrd` package, and get it into the standard data matrix form as we learnt how to do it in Exercise 1. See `ex2_1_1.py` for details. There are 90 data objects with 8 attributes each. Do you get the correct data matrix X of size 90×8?

2.1.2 The data resides in an 8 dimensional space where each dimension corresponds to each of the 8 NanoNose sensors. This makes visualization of the raw data difficult, because it is difficult to plot data in more than 2–3 dimensions.

Plot the two attributes A and B against each other in a scatter plot using `ex2_1_2.py`.

Script details:

- *You need to import* `matplotlib.pyplot` *package to use plotting functions in Python:*
  `from matplotlib.pyplot import *`
- *Use* `plot()` *function to plot data.*
- *The attributes A and B are the first and second columns of the matrix $\boldsymbol{X}$.'*
- *You can use indexing to get the columns out of the matrix, e.g.,* `x=X[:,1]` *or* `y = X[:,2]`
- *Notice that the third argument of the* `plot()` *command can be used to set a plot symbol. For example, the command* `plot(x,y,'o')` *plots a scatter plot with circles.*
- *Use* `show()` *function to render the plot.*
- *You can find extensive help and numerous examples on matplotlib website:* `http://matplotlib.sourceforge.net`

Try to change the dimensions that are plotted against each other.

We will use principal component analysis to reduce the dimensionality of the data. PCA is computed by subtracting the mean of the data, $\boldsymbol{Y} = \boldsymbol{X} - \mathbf{1}\boldsymbol{\mu}$ (where $\boldsymbol{\mu}$ is a (row) vector containing the mean value of each attribute and $\mathbf{1}$ is a N by 1 column vector of ones in all entries) and then calculating the singular value decomposition (SVD) of the zero mean data, i.e. $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$.

From PCA we can find out how much of the variation in the data each PCA component accounts for. This is given by

$$\rho_m = \frac{s_{mm}^2}{\sum_{m'=1}^{M} s_{m',m'}^2},$$

i.e. the squared singular value of the given component divided by the sum of all the squared singular values.

2.1.3 Compute the PCA of the NanoNose data and plot the percent of variance explained by the principal components as well as the cumulative variance explained using `ex2_1_3.py`.

Script details:

- *You can use the method* `mean()` *of array or matrix object to compute the mean of the data. You should compute the mean for each attribute(column), i.e., the vector of means should have M elements.*
- *You cannot directly subtract a vector from a matrix. One way to accomplish this is to subtract the product of vector of ones and vector of means:*
  `Y = X - np.ones((N,1))*X.mean(0)`
- *You can use the function* `numpy.linalg.svd()` *to compute the SVD.*
- *To extract the diagonal from a matrix, use the method* `diagonal()` *of an array object, or use* `np.diagonal()` *or* `np.diag()`.

Can you verify that more than 90% of the variation in the data is explained by the first 3 principal components? How many components would be needed for 95 %?

2.1.4 Plot principal component 1 and 2 against each other in a scatterplot, see the script `ex2_1_4.py` for details.

Script details:

- *Data can be projected onto the principal components using* `Z = Y@V` *or* `Z = np.dot(Y,V)`, *where* `Y` *is centered data.*
- *You learned how to make a scatter plot in Exercise 2.1.2.*

What are the benefits of visualizing the data by the projection given by PCA over plotting two of the original data dimensions against each other? Compare with the scatter plots of attributes you made in Exercise 2.1.2.

2.1.5 Interpret the principal directions ($\boldsymbol{V}$) obtained using the PCA. Consider the script `ex2_1_5.py`. Which of the original attributes does the second principal component mainly capture the variation of and what would cause an observation to have a large negative/positive projection onto the second principal component? (remember both the attributes and the prinpal component has a sign and a magnitude)

Script details:

- *The columns of* `V` *gives you the principal component directions*

· *The data is projected onto the second principal component by* `Y@V[:,1]`

We can correct for differences in scale by standardization. When doing PCA on data with attributes of different scales, it can be very important to standardize the dataset. We standardize a dataset by ensuring each attribute has a mean of zero (as before), but also has a variance of one (i.e. zero mean and unit variance).

In `ex2_1_5.py` you saw that we can interpret the principal directions by investigating the coefficients in the vectors of $\boldsymbol{V}$. Another way to approach intepreting the principal directions is to plot the coefficients as vectors in the principal component space. In the PC1/PC2-space, we can for instance interpret the relationship between PC1, PC2 and a given attribute by drawing a line form Origo to the coefficients in PC1 and PC2 corresponding to the attribute. The diretion and magnitude of such a vector defines how the data from that attribute is projected onto the PC1/PC2-space—e.g. if the vector points in positive direction of PC1, then positive values of that attribute contributes to a positive projection onto PC1. Since the vectors in $\boldsymbol{V}$ are unit-vectors, all coefficients will lie within the unit-circle.

2.1.6 Investigate the standard deviation of the NanoNose attributes and try to determine if some of the attributes have higher variance than the others using `ex2_1_6.py`. Which attribute has the highest standard deviation? Use the script to visualize the difference between either only subtracting the mean or both subtracting the mean and dividing by the standard deviation (visualize: the projection, attribute coefficients, and the variance explained of a PCA for the two). How did the attribute with the highest standard deviation change in terms of its direction and magnitude in the attribute coefficients? How did the variance explained change? Lastly, try multiplying one of the attributes with a factor 100 and see how that changes the PCA.

2.2 Structure in handwritten digits

The US Postal Service (USPS) wanted to automate the process of sorting letters based on their zip-codes. We will presently consider a dataset of USPS handwritten digits available at `http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html`, see also [3]. There are two datasets containing handwritten digits `testdata` and `traindata`.

2.2.1 Load the dataset. Inspect and run the script `ex2_2_1.py` to visualize the first digit of the traindata (the script uses `reshape` to turn a digit vector into an image and `imshow()` to display the image).

2.2.2 Inspect and run the script `ex2_2_2.py`. Show that it requires 22 PCA components to account for more than 90% of the variance in the data. Show that the first principal component is almost sufficient to separate zeros and ones. Examine the first principal component and discuss and understand what it captures.

2.2.3 Change the value of `K` and show that reconstruction accuracy improves when more principal components are used. How many principal components do you need to be able to see the different digits properly? What happens if you set `K=256`?

2.2.4 Try decomposing one digit at a time. Hint: Modify the variable `n` to contain only a single digit. Explain what happens to the principal components when only a single digit type is analyzed compared to when all digit types are analyzed at the same time.

2.3 Extra challenge

We will later in the course learn various methods for classification. Among the approaches we will learn is K-nearest neighbor (KNN) classification. For now we will consider the KNN classifier a black box that we will use to evaluate how well we can determine the digit class in the space given by the K first principal components, i.e. after filtering out the PCA components with smallest singular values which we consider components pertaining to noise.

2.3.1 Inspect and run the script `ex2_3_1.py` and see how well we are able to classify the digits when we use say `K=10` PCA components, `K=40` PCA components and the whole data, i.e `K=256` PCA components. Show that the classifier is best when using around 40–60 PCA components, and explain why that is so.

2.4 Tasks for the report

For the report, complete the following sections:

- **A detailed explanation of the attributes of the data.**

- Describe if the attributes are discrete/continuous, Nominal/Ordinal/Interval/Ratio,
- Give an account of whether there are data issues (i.e. missing values or corrupted data) and describe them if so.

If your data set contains many similar attributes, you may restrict yourself to describing a few representative features (apply common sense).

- **Tasks from PCA section**

There are three aspects that needs to be described when you carry out the PCA analysis for the report:

- The amount of variation explained as a function of the number of PCA components included,
- the principal directions of the considered PCA components (either find a way to plot them or interpret them in terms of the features),
- the data projected onto the considered principal components.

If your attributes have different scales you should include the step where the data is standardizes by the standard deviation prior to the PCA analysis.

1 Homework problems for this week

Problems

Question 1. Spring 2012 question 4: The first and second principal components directions of the data in the RAT dataset (considering only the attributes $x_1 - x_{13}$) in Table 1 are:

$$\boldsymbol{v}_1 = \begin{bmatrix} 0.0247 \\ -0.0388 \\ -0.3288 \\ -0.2131 \\ 0.0477 \\ -0.4584 \\ 0.2683 \\ -0.0838 \\ -0.5020 \\ -0.0200 \\ -0.3091 \\ -0.2588 \\ 0.3714 \end{bmatrix}, \quad \boldsymbol{v}_2 = \begin{bmatrix} -0.0764 \\ 0.5675 \\ -0.0550 \\ 0.2449 \\ 0.3115 \\ -0.1999 \\ 0.1738 \\ 0.3668 \\ -0.0737 \\ 0.2988 \\ 0.0628 \\ 0.4446 \\ 0.1051 \end{bmatrix}.$$

and in Figure 1 the data projected onto the first two principal components is plotted against the average consumer ratings (RAT). Which of the following statements is *correct*?

No.	Attribute description	Abbrev.
x_1	Type (0 = served cold, 1 = served hot)	TYPE
x_2	Calories per serving	CAL
x_3	Grams of protein	PROT
x_4	Grams of fat	FAT
x_5	Milligrams of sodium	SOD
x_6	Grams of dietary fiber	FIB
x_7	Grams of complex carbohydrates	CARB
x_8	Grams of sugars	SUG
x_9	Milligrams of potassium	POT
x_{10}	Vitamins and minerals in 0%, 25%, or 100% of FDA recommendations	VIT
x_{11}	Shelf position (1, 2, or 3, counting from the floor)	SHELF
x_{12}	Weight in ounces of one serving	WEIGHT
x_{13}	Number of cups in one serving	CUPS
x_{14}	Name of cereal brand	NAME
y	Average rating of the cereal (from 0 to 100)	RAT

Table 1: Attributes in a study of cereals (i.e. breakfast products, taken from http://lib.stat.cmu.edu/DASL/Datafiles/Cereals.html). The data we consider has 74 observations (i.e., the original data has 77 observations but three observations have been removed due to missing values). The data has 14 input attributes x_1–x_{14} and one output variable y which defines the average rating of the cereal product given by the consumers.

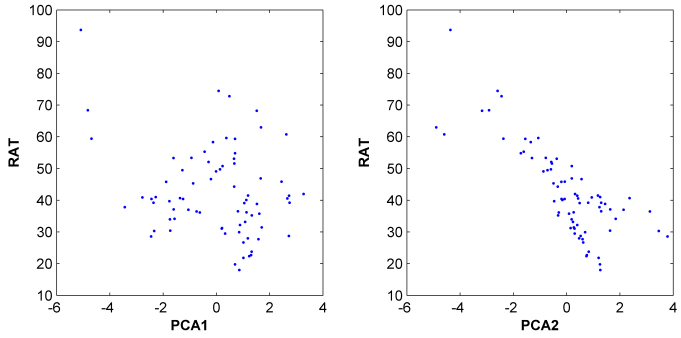

Figure 1: The output RAT plotted against the first and second principal component respectively.

A Relatively high values of CAL, PROT, FAT, FIB, SUG, POT, VIT, SHELF, and

WEIGHT and low values of TYPE, SOD, CARB, and CUPS will result in a negative projection onto the first principal component.

B PCA2 primarily discriminates between relatively low values of PROT and high values of SHELF.

C The data projected onto the second principal component (i.e., PCA2) is positively correlated with RAT.

D The principal component directions are not guaranteed to be orthogonal to each other since the data has been standardized.

E Don't know.

Question 2. Fall 2011 question 1: Consider the data set described in Table 2. Which statement about the attributes in the data set is *correct*?

No.	Attribute description	Abbrev.
x_1	Age of Mother in Whole Years	Age
x_2	Mothers Weight in Pounds	MW
x_3	Race (1 = Other, 0 = White)	Race
x_4	History of Hypertension (1 = Yes, 0 = No)	HT
x_5	Uterine Irritability (1 = Yes, 0 = No)	UI
x_6	Number of Physician Visits First Trimester	PV
y	Birth Weight in Kilo Grams	BW

Table 2: Attributes in a study on risk factors associated with giving birth to a low birth weight (less than 2.5 kg) baby [Hosmer and Lemeshow, Applied Logistic Regression, 1989]. The data we consider contains 189 observations, 6 input attributes x_1–x_6, and one output variable y.

A Race, HT and UI are ordinal.

B Age and PV are ratio.

C Age is continuous and ratio.

D MW is discrete whereas PV is continuous.

E Don't know.

Question 3. Fall 2011 question 2: Consider the data set described in Table 2. Each attribute in the data set is standardized, and we carry out a principal component analysis (PCA) on the standardized input data, x_1–x_6. The singular values obtained are: $\sigma_1 = 17.0$, $\sigma_2 = 15.2$, $\sigma_3 = 13.1$, $\sigma_4 = 13.0$, $\sigma_5 = 11.8$, $\sigma_6 = 11.3$. The first and second principal component directions are:

$$\boldsymbol{v}_1 = \begin{bmatrix} 0.5238 \\ 0.5237 \\ -0.3491 \\ 0.1981 \\ -0.3369 \\ 0.4204 \end{bmatrix}, \quad \boldsymbol{v}_2 = \begin{bmatrix} -0.2948 \\ 0.3452 \\ 0.3584 \\ 0.6808 \\ -0.3049 \\ -0.3302 \end{bmatrix}.$$

Which one of the following statements is *incorrect*?

A The first three principal component account for more than 90% of the variation in the data.

B Relatively heavy, old and white mothers that frequently goes to the physician and have a history of hypertension but do not have uterine irritability will have a positive projection onto the first principal component.

C Relatively young, heavy mothers that are not white and have a history of hypertension but infrequently goes to the physician and do not have a uterine irritability will have a positive projection onto the second principal component.

D Since the data is standardized we do not need to subtract the mean when performing the PCA but can directly carry out the singular value decomposition on the standardized data.

E Don't know.

References

[1] Nanonose project.

[2] Tommy S Alstrøm, Jan Larsen, Claus H Nielsen, and Niels B Larsen. Data-driven modeling of nano-nose gas sensor arrays. In *SPIE Defense, Security, and Sensing*, pages 76970U–76970U. International Society for Optics and Photonics, 2010.

[3] Jonathan J. Hull. A database for handwritten text recognition research. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(5):550–554, 1994.

Measures of similarity, summary statistics and probabilities with PYTHON

Objective: The overall objective is to get a basic understanding for measures of similarity as well as summary statistics. Upon completing this exercise it is expected that you:

- Understand the *bag of words* representation for text documents including filtering methods based on removal of stop words and stemming.
- Understand how to calculate summary statistics such as mean, variance, median, range, covariance and correlation.
- Understand the various measures of similarity such as Jaccard and Cosine similarity and apply similarity measures to query for similar observations.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 3 folder available from Campusnet.

PYTHON Help: You can get help in your Python interpreter by typing `help(obj)` or you can explore source code by typing `source(obj)`, where `obj` is replaced with the name of function, class or object.
Furthermore, you get context help in Spyder after typing function name or namespace of interest. In practice, the fastest and easiest way to get help in Python is often to simply Google your problem. For instance: `"How to add legends to a plot in Python"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
Classification	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
Classification	`C`	integer	Scalar	Number of classes.

3.1 The document-term matrix

An important area of research in machine learning and data mining is the analysis of text documents. Here, important tasks are to be able to search documents as well as group related documents together (clustering). In order to accomplish these tasks the text documents must be converted into a format suitable for data modeling. We will use the *bag of words* representation. Here, text documents are stored in a matrix $\mathbf{X}$ where x_{ij} indicate how many times word j occurred in document i.

Suppose that we have 5 text documents [2], each containing just a single sentence.

Document 1: The Google matrix P is a model of the internet.
Document 2: P_{ij} is nonzero if there is a link from webpage i to j.
Document 3: The Google matrix is used to rank all Web pages.
Document 4: The ranking is done by solving a matrix eigenvalue problem.
Document 5: England dropped out of the top 10 in the FIFA ranking.

3.1.1 Propose a suitable *bag of words* representation for these documents. You should choose approximately 10 key words in total defining the columns in the document-term matrix and the words are to be chosen such that each document at least contains 2 of your key words, i.e. the document-term matrix should have approximately 10 columns and each row of the matrix must at least contain 2 non-zero entries.

3.1.2 In practice, the above procedure is carried out automatically, see the script `ex3_1_2.py`. We will use an `sklearn` function from the feature extraction-module, called `CountVectorizer` to generate a document-term matrix and to convert it into the format described in the beginning of the exercise (Representation of data in Python).

Script details:

- *Make sure that you have the* `sklearn`*-package installed.*
- *Read more about the CountVectorizer using:*
  `help(CountVectorizer)`
  after it has been imported from `sklearn`*.*

Compare the generated document-term matrix to the one you generated yourself.

Stop words are words that one can find in virtually any document. Therefore, the occurrence of such a word in a document does not distinguish the document from other documents. The following is the beginning of one particular stop word list:

> a, a's, able, about, above, according, accordingly, across, actually, after, afterwards, again, against, ain't, all, allow, allov, ahnost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anyway, anyways, anywhere, apart, appear, appreciate, appropriate, are, around, as, aside,ask, ...

When forming the document-term it is common to remove these specified stop words.

3.1.3 The generated document-term matrix contains words that carry little information such as the word "the". We will remove these words as they can be interpreted as "noise" carrying no information about the content of the documents. Compute a new document-term matrix with stop words removed using `ex3_1_3.py`

Script details:

- *A list of stop words is stored in the file* `../Data/stopWords.txt`*.*
- *Once the stop words are loaded, they can be parsed to the* `CountVectorizer` *by parsing it using the keyword* `stop_words`*.*

Inspect the document-term matrix: How does it compare to your original matrix?

Stemming denotes the process for reducing inflected (or sometimes derived) words to their stem, base or root form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Clearly, from the point of view of information retrieval, no information is lost in the following stemming reduction:

$$\left.\begin{array}{l} \textit{computable} \\ \textit{computing} \\ \textit{computed} \\ \textit{computational} \\ \textit{computation} \end{array}\right\} \rightarrow \textit{comput}$$

3.1.4 Document 3, 4 and 5 have the word "rank" in common. However in document 4 and 5 this word is stored as a the separate word entry "ranking" in the document-term matrix whereas in document 3 it is stored as the word entry "rank" . As such, the document-term matrix does not indicate that document 3, 4 and 5 share the word "rank". By the use of stemming we can obtain a matrix that indicate that the word "rank" appears in all 3 documents. Enable stemming and compute a new document-term matrix using the script `ex3_1_4.py`.

Script details:

- *Make sure you have the* `nltk`*-package installed.*
- *You can stem verbs using the a* `PorterStemmer`*. Once the PorterStemmer is made, try writing:*
  `stemmer.stem('computational') == stemmer.stem('computable').`

Inspect the document-term matrix: How does it compare to your original matrix?

Based on our document-term representation we can now make simple searches (queries) in our documents based on some form of similarity measure between our query vector and document-term representation. Lets say we want to find all documents that are relevant to the query "**solving** for the **rank** of a **matrix**." This is represented by a query vector, $\boldsymbol{q}$, constructed in a way analogous to the document-term matrix, $\boldsymbol{X}$:

$$\boldsymbol{X} = \begin{array}{c} \begin{array}{cccccccccccccccc} \text{drop} & \text{eigenvalu} & \text{england} & \text{fifa} & \text{googl} & \text{internet} & \text{link} & \text{matrix} & \text{model} & \text{nonzero} & \text{page} & \text{problem} & \text{rank} & \text{solv} & \text{top} & \text{web} & \text{webpag} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$\boldsymbol{q} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

3.1.5 We will use the *cosine distance* as a measure of similarity between the i'th document $\boldsymbol{x}_i$ and the query vector $\boldsymbol{q}$, i.e. $\cos(\boldsymbol{q}, \boldsymbol{x}_i) = \frac{\boldsymbol{q}\boldsymbol{x}_i^\top}{\|\boldsymbol{q}\|\|\boldsymbol{x}_i\|}$. Compute the cosine similarity between each document and the query using `ex3_1_5.py`, and show that Document 4 is most similar to the query.

Script details:

- *You can extract a document (row of the* `X` *matrix) using the command* `x=X[i,:]` *where* `i` *is the index of the document.*
- *Numpy matrices and arrays can be transposed using notation* `m.T` *or* `m.transpose()`.
- *Dot products between two row vectors can be computed as* `numpy.dot(q,X.T)` *(or simply* `q@X.T`*).*
- *The norm of a vector can be computed using the function* `numpy.linalg.norm()`.

Explain what documents, according to our similarity measure, are most related to the query.

3.1.6 (OPTIONAL) If you find text processing exciting, read more about Natural Language Processing toolkit. Here is a good place to start: `http://www.nltk.org/book/`

3.2 Summary Statistics

3.2.1 Consult the script `ex3_2_1.py`. Calculate the (empirical) mean, standard deviation, median, and range of the following set of numbers:

$$\{-0.68, -2.11, 2.39, 0.26, 1.46, 1.33, 1.03, -0.41, -0.33, 0.47\}$$

Script details:

- *Look at the help page of the functions* `mean()`, `std()`, `median()`, `min()` *and* `max()` *of NumPy* `array` *class.*

3.3 Measures of similarity

We will use a subset of the data on wild faces described in [1] transformed to a total of 1000 gray scale images of size 40×40 pixels, we will attempt to find faces in the data base that are the most similar to a given query face. To measure similarity we will consider the following measures: SMC, Jaccard, Cosine, ExtendedJaccard, and Correlation. These measures of similarity are given by:

$$\begin{aligned}
\text{SMC}(\boldsymbol{x}, \boldsymbol{y}) &= \frac{\text{Number of matching attribute values}}{\text{Number of attributes}} \\
\text{Jaccard}(\boldsymbol{x}, \boldsymbol{y}) &= \frac{\text{Number of matching presences}}{\text{Number of attributes not involved in 00 matches}} \\
\text{Cosine}(\boldsymbol{x}, \boldsymbol{y}) &= \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|} \\
\text{ExtendedJaccard}(\boldsymbol{x}, \boldsymbol{y}) &= \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \boldsymbol{x}^\top \boldsymbol{y}} \\
\text{Correlation}(\boldsymbol{x}, \boldsymbol{y}) &= \frac{\text{cov}(\boldsymbol{x}, \boldsymbol{y})}{\text{std}(\boldsymbol{x})\text{std}(\boldsymbol{y})}
\end{aligned}$$

where $\text{cov}(\boldsymbol{x}, \boldsymbol{y})$ denotes the covariance between $\boldsymbol{x}$ and $\boldsymbol{y}$ and $\text{std}(\boldsymbol{x})$ denotes the standard deviation of $\boldsymbol{x}$.

Notice that the SMC and Jaccard similarity measures only are defined for binary data, i.e., data that takes values of $\{0, 1\}$. As the data we analyze is non-binary, we will transform the data to be binary when calculating these two measures of similarity by setting

$$x_i = \begin{cases} 0 & \text{if } x_i < \text{median}(\boldsymbol{x}) \\ 1 & \text{otherwise.} \end{cases}$$

Note that, depending on the situation, it can be incorrect to encode information in a single binary attribute—and this is true for binary attributes in general. If the meaning behind the value 0 is not specifically non-presence of an attribute, it can be erroneous. For instance, if male/female is encoded in one binary attribute (male: 0, female: 1), some measures will not model the information carried in being male, and a one-of-out-K encoding would be a proper representation.

For the next step, we will look at the USPS handwritten digit database. The digits dataset contains 9298 16x16 handwritten (single) digits images in greyscale.

3.3.1 Inspect and run the script `ex3_3_1.py`. The script loads the digits dataset, computes the similarity between a selected query image and all others, and display the query image, the 5 most similar images, and the 5 least similar images. The value of the used similarity measure is shown below each image. Try changing the query image and the similarity measure and see what happens.

3.3.2 We will investigate how scaling and translation impact the following three similarity measures: Cosine, ExtendedJaccard, and Correlation. Let α and β be two constants. Which of the following statements are correct? Check your answers with the script `ex3_3_2.py`

$$\begin{aligned}
\text{Cosine}(\boldsymbol{x}, \boldsymbol{y}) &= \text{Cosine}(\alpha\boldsymbol{x}, \boldsymbol{y}) \\
\text{ExtendedJaccard}(\boldsymbol{x}, \boldsymbol{y}) &= \text{ExtendedJaccard}(\alpha\boldsymbol{x}, \boldsymbol{y}) \\
\text{Correlation}(\boldsymbol{x}, \boldsymbol{y}) &= \text{Correlation}(\alpha\boldsymbol{x}, \boldsymbol{y}) \\
\text{Cosine}(\boldsymbol{x}, \boldsymbol{y}) &= \text{Cosine}(\beta + \boldsymbol{x}, \boldsymbol{y}) \\
\text{ExtendedJaccard}(\boldsymbol{x}, \boldsymbol{y}) &= \text{ExtendedJaccard}(\beta + \boldsymbol{x}, \boldsymbol{y}) \\
\text{Correlation}(\boldsymbol{x}, \boldsymbol{y}) &= \text{Correlation}(\beta + \boldsymbol{x}, \boldsymbol{y})
\end{aligned}$$

Script details:

- *Type* `help(similarity)` *to learn about the Python function that is used to compute the similarity measures.*
- *Even though a similarity measure is theoretically invariant e.g. to scaling, it might not be exactly invariant numerically.*

3.4 Tasks for the report

Provide the basic summary statistics of your attributes preferable in a table and consider if attributes are correlated, see also the functions numpy.cov() and numpy.corrcoef(). Specifically address the questions:

- Include basic summary statistics of the attributes.

1 Homework problems for this week

Problems

Question 1. Fall 2014 question 10:

In table 1 is given the pairwise cityblock distances between 8 observations along with a description of the dataset. What can be concluded about the similarity of observation o_1 and o_3?

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
o_1	0	4	7	9	5	5	5	6
o_2	4	0	7	7	7	3	7	8
o_3	7	7	0	10	6	6	4	9
o_4	9	7	10	0	8	6	10	9
o_5	5	7	6	8	0	8	6	7
o_6	5	3	6	6	8	0	8	11
o_7	5	7	4	10	6	8	0	7
o_8	6	8	9	9	7	11	7	0

Table 1: Pairwise Cityblock distance, i.e $d(o_i, o_i) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1 = \sum_{k=1}^{M} |x_{ik} - x_{jk}|$, between 8 observations. Each observation o_i corresponds to a $M = 15$ dimensional binary vector, $x_{ik} \in \{0, 1\}$. The blue observations $\{o_1, o_2, o_3, o_4\}$ belong to class C_1 and the black observations $\{o_5, o_6, o_7, o_8\}$ belong to class C_2.

A $\text{COS}(o_1, o_3) = 0.533$

B $\text{J}(o_1, o_3) = 0.533$

C $\text{SMC}(o_1, o_3) = 0.533$

D There is insufficient information to draw specific conclusions.

E Don't know.

Question 2. Spring 2013 question 18: We will let $J(A, B)$, $SMC(A, B)$, and $cos(A, B)$ denote the Jaccard Coefficient, Simple Matching Coefficient and Cosine Similarity respectively between observation A and B. We will consider the data in Table 2 containing 10 observations denoted NS1, NS2, NS3, NS4, NS5, AS1, AS2, AS3, AS4, and AS5 such that the first observation is given by NS1= $\{1, 0, 0, 1, 0, 1, 1, 0\}$. Which one of the following statements is *correct*?

	CD_Y	CD_N	AST_Y	AST_N	SI_Y	SI_N	HF_Y	HF_N
NS1	1	0	0	1	0	1	1	0
NS2	0	1	1	0	1	0	1	0
NS3	1	0	0	1	0	1	1	0
NS4	0	1	1	0	0	1	1	0
NS5	1	0	1	0	1	0	1	0
AS1	0	1	1	0	0	1	1	0
AS2	0	1	1	0	0	1	1	0
AS3	0	1	1	0	0	1	1	0
AS4	0	1	0	1	1	0	0	1
AS5	1	0	1	0	0	1	1	0

Table 2: Given are the first five subjects with normal semen (denoted NS1, NS2, ..., NS5) as well as the first five subjects with abnormal semen (denoted AS1, AS2, ..., AS5) including whether these subjects have had a childhood disease or not (CD_Y, CD_N), accident or serious trauma or not (AST_Y, AST_N), serious injury or not (SI_Y, SI_N), and high fever or not (HF_Y, HF_N).

A $J(\text{NS1,NS2}) = SMC(\text{NS1,NS2})$

B $cos(\text{NS4,NS5}) = \frac{1}{8}$

C $J(\text{NS5,AS5}) = SMC(\text{NS5,AS5})$

D $cos(\text{NS5,AS5}) = \frac{3}{4}$

E Don't know.

Question 3. Fall 2013 question 18: We will let $J(A, B)$, $SMC(A, B)$, and $cos(A, B)$ denote the Jaccard Coefficient, Simple Matching Coefficient and Cosine Similarity respectively between observation A and B. We will consider the data in Table 3 containing 10 observations denoted S1, S2, S3, S4, S5, NS1, NS2, NS3, NS4, and NS5 such that the first observation is given by S1= $\{1, 0, 1, 0, 1, 0\}$. Which one of the following statements is *correct*?

	YA_Y	YA_N	OA_Y	OA_N	PA_Y	PA_N
S1	1	0	1	0	1	0
S2	1	0	1	0	0	1
S3	0	1	0	1	1	0
S4	0	1	1	0	1	0
S5	0	1	1	0	1	0
NS1	0	1	1	0	1	0
NS2	0	1	0	1	1	0
NS3	1	0	0	1	0	1
NS4	0	1	1	0	1	0
NS5	0	1	1	0	1	0

Table 3: Given are five subjects that survived in Haberman's study (denoted S1, S2, ..., S5) as well as the five subjects that did not survive in Haberman's study (denoted NS1, NS2, ..., NS5) including whether these subjects are young or old (YA_Y, YA_N), were operated after 1960 or not (OA_Y, OA_N), and had positive axillary nodes or not (PA_Y, PA_N).

A Using the Jaccard coefficient S1 is more similar to S2 than to NS1, i.e. $J(S1, S2) > J(S1, NS1)$.

B Using the Simple Matching coefficient S1 is more similar to S2 than to NS1, i.e. $SMC(S1, S2) > SMC(S1, NS1)$.

C The Jaccard coefficient between S1 and S2 is identical to the Cosine Similarity between S1 and S2, i.e. $J(S1, S2) = cos(S1, S2)$.

D The Simple Matching coefficient between S1 and S2 is identical to the Cosine Similarity between S1 and S2, i.e. $SMC(S1, S2) = cos(S1, S2)$.

E Don't know.

References

[1] Tamara L Berg, Alexander C Berg, Jaety Edwards, and DA Forsyth. Who's in the picture. *Advances in neural information processing systems*, 17:137–144, 2005.

[2] Lars Eldén. *Matrix Methods in Data Mining and Pattern Recognition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

Probability densities and data visualization with PYTHON

Objective: Upon completing this exercise it is expected that you:

- Understand the univariate and multivariate normal distribution.
- Get an understanding of the many ways data can be visualized including histograms, boxplots, and scatter plots.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 4 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum: https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
Classification	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
Classification	`C`	integer	Scalar	Number of classes.

4.1 Understanding the Univariate and Multivariate Normal Distribution

The univariate and multivariate normal distribution is central to many machine-learning methods. In this exercise we will investigate the basic properties of these distributions in Python.

4.1.1 Inspect and run the script `ex4_1_1.py`. The script generates 200 samples from a Normal distribution with mean $\mu = 17$ and standard deviation $\sigma = 2$ and plot the samples as well as a histogram of the samples.

Script details:

- *Read about* `np.random.normal()` *function to learn how to generate Normal distributed random numbers in Python.*
- *The function* `plot()` *can be used to plot the samples.*
- *The function* `hist()` *can be used to plot a histogram.*
- *Check Matplolib tutorial for more examples:* `matplotlib.sourceforge.net/users/pyplot_tutorial.html`

Try running the code a few times and see how the data and histogram changes when new random numbers are generated from the same distribution.

The histogram is generated by counting how many of the samples fall within the range covered by each bin of the histogram. Try changing the number of bins in the histogram.

4.1.2 Compute the empirical mean and standard deviation of the generated samples. Show, that they are close but not equal to the theoretical values used to generate the random samples. See the script `ex4_1_2.py` for details.

Script details:

- *Take a look at the functions* `mean` *and* `std`.

Try running the code a few times and see how the empirical mean and standard deviation changes when new random numbers are generated from the same distribution.

In the next part we will see how the number of samples influence the resulting histogram.

4.1.3 Inspect and run the script `ex4_1_3.py` which generates random samples from the Normal distribution and plots the histogram as before. Also, the function plots the true probability density function (`scipy.stats.norm.pdf()`) of the Normal distribution.

Show that when the number of samples `N` is increased, the histogram approximates the pdf better and the empirical estimates of the mean and standard deviation improve.

So far we have been considered a 1-dimensional Normal distributed random variable. In the next step we will consider the multivariate Normal distribution in two dimensions.

The covariance matrix for a 2D Gaussian is described by

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & \text{cov}\,(x_1, x_2) \\ \text{cov}\,(x_2, x_1) & \sigma_2^2 \end{bmatrix},$$

where $\text{cov}(\cdot, \cdot)$ is covariance between two random variables. Note that $\text{cov}\,(x_1, x_2) = \text{cov}\,(x_2, x_1)$, i.e., the covariance matrix is symmetric, and $\sigma_n^2 = \text{cov}\,(x_n, x_n)$. We can write the covariance matrix in terms of Correlation, i.e.

$$\begin{aligned} \text{Correlation}\,(x_1, x_2) &= \frac{\text{cov}\,(x_1, x_2)}{\sqrt{\text{cov}\,(x_1, x_1)}\sqrt{\text{cov}\,(x_2, x_2)}} \Rightarrow \\ \text{cov}\,(x_1, x_2) &= \text{Correlation}\,(x_1, x_2)\sqrt{\text{cov}\,(x_1, x_1)}\sqrt{\text{cov}\,(x_2, x_2)}. \end{aligned}$$

4.1.4 Inspect and run the script `ex4_1_4.py`. The script generates 1000 samples from a 2-dimensional Normal distribution with mean

$$\boldsymbol{\mu} = \begin{bmatrix} 13 & 17 \end{bmatrix}$$

and covariance matrix

$$\mathbf{\Sigma} = \begin{bmatrix} 4 & 3 \\ 3 & 9 \end{bmatrix}.$$

Script details:

- *Look at the function* `np.random.multivariate_normal()` *to learn how you can generate multivariate Normal distributed random numbers in Python.*

4.1.5 Inspect and run the script `ex4_1_5.py` which generates 2-dimensional Normal distributed random samples and plots a scatter plot as well as a 2-dimensional histogram. In the script, the covariance matrix is constructed from the standard deviations and correlation as described above.

Show that when the correlation between x_1 and x_2 is zero, the scatter plot and 2-d histogram have the shape of an axis-aligned ellipse. Can you explain why?

Show that when the correlation between x_1 and x_2 is one, the values of x_1 and x_2 fall on a straight line. Can you explain why?

Try varying the number of samples, the mean, the standard deviations, the correlation and the number of histogram bins and see what happens.

In this part we will use the concepts and commands from the previous section in order to make some very simple statistical models of the digits data set (16×16 pixel images of hand written digits) we considered in last weeks exercise.

4.1.6 Inspect and run the script `ex4_1_6.py`. The script loads the digits data set, and computes the mean of each pixel, the standard deviation of each pixel, and the covariance matrix between the pixels. By default, only images of "ones" are included in the analysis.

Does the mean look like you expect? Can you explain why the standard deviation is highest along the edges of the digit one? Try to change the digit you analyze and get a felling of how different the individual digits are.

For each pixel we now have a mean and a standard deviation, i.e., 256 means and 256 corresponding standard deviations. So in essence we can make a a simple model of the digits with a Normal distributions for each individual pixel.

Since we know how to draw a new sample from a Normal distribution we can draw a sample for each individual pixel based on their respective 1D normal distribution (i.e. draw a total of 256 values). Combining these samples we end up with a new digit. The question is now how natural our newly generated/artificial samples are, and if they at all are possible to recognize as digits.

With our simple model above we argued that we had 256 different 1D Normal distributions; however, we could also look at the problems in terms of the multivariate Normal. Instead of assuming 256 independent 1D Gaussians we could formulate our

model for digits as a 256-dimensional multivariate Normal, which allows each pixel to depend on the other pixels. This dependency is described through the covariance matrix.

4.1.7 Inspect and run the script `ex4_1_7.py`. The script generates 10 new images with the same mean and standard deviation as the data using a 1-dimensional Normal distribution for each pixel. Next, the script generates 10 new images with the same mean and covariance as the data using a $16 \cdot 16 = 256$-dimensional multivariate Normal distribution.

Which model is best? Try changing the analyzed digits and see what happens.

4.1.8 (Extra challenge) Try to vary the number of observations of a given digit you use to estimate the mean and (co)variance. Does the number of observations used make a difference on the quality of the generated digits?

4.2 Visualizing Fisher's Iris data

In this exercise we will reproduce most of the figures in "Introduction to Data Mining." section 3.3 in Matlab using the Iris flower dataset that was also introduced in Exercise 1.

4.2.1 The Iris data set is available in the Excel file `Data/iris.xls`. Load the data into Python and generate all the variables described in *Representation of data in Python* using the script `ex4_2_1.py`.

Script details:

- *You can use the package* `xlrd` *which you have used before in exercise 2.2.*

4.2.2 Inspect and run `ex4_2_2.py`. The script plots a histogram of each of the four attributes in the Iris data.

Script details:

- *You can use the command* `hist` *to plot a histogram.*
- *Use indexing to extract each attribute. For example,* `X[:,m-1]` *extracts the* `m`*'th attribute.*
- *For multiple plots in one figure window you can use the command* `subplot(n,m,i)`.

Show on the graph that the petal length is either between 1 and 2 cm. or between 3 and 7 cm. but that no flowers in the data set have a petal length between 2 and 3 cm. Do you think this could be useful to discriminate between the different types of flowers?

4.2.3 Inspect and run `ex4_2_3.py`. The script produces a boxplot of the four attributes in the Iris data as shown in Figure 3.11 in the book.

Script details:

- *Take a look at the function* `boxplot`.
- *Type* `help(boxplot)`*" to see how you can adjust the boxplot and add labels.*

This boxplot shows the same information as the histogram in the previous exercise. Discuss the advantages and disadvantages of the two types of plots.

4.2.4 Inspect and run `ex4_2_4.py`. The scripts produces a boxplot for each attribute for each class as shown in Figure 3.12 in the book.

Script details:

- *Use the functions* `subplot()` *and* `boxplot()`.
- *The variable* `y` $\in \{0, 1, \ldots, \texttt{C} - 1\}$ *contains the class labels. To extract the data objects belonging to, say, class* `c`*, you can use* `y` *to index into* `X` *like this:* `X[(y==c), :]`
- *It is easier to compare the boxplots if they are all on the same axis. To do this, you can use the function* `ylim()`.

Show on the graph that all the Iris-setosa in this data set have a petal length between 1 and 2 cm.

4.2.5 Inspect and run `ex4_2_5.py`. The scripts produces a matrix of scatter plots of each combination of two attributes against each other as shown in Figure 3.16 in the book.

Script details:

- *To make a scatter plot, you can use the function* `plot(x,y,s)` *where* `x` *and* `y` *specify the coordinates and* `s` *is a string that specifies the line style and plot symbol, e.g.,* `s='.'` *to make dots.*
- *To extract the data values for the* `m`*'th attribute in the* `c`*'th class, you can write* `X[(y==c),m]`.
- *You can use the command* `hold all` *to plot multiple plots on top of each other.*

Say you want to discriminate between the three types of flowers using only the length and width of either sepal or petal. Show on the graph why it would be better to use petal length and width rather than sepal length and width.

4.2.6 Inspect and run `ex4_2_6.py`. The script produces a 3-dimensional scatter plot of three attributes as shown in Figure 3.17 in the book.

Script details:

- *Read more about plotting in 3 dimensions:* `matplotlib.sourceforge.net/mpl_toolkits/mplot3d/tutorial.html`
- *To plot in 3 dimensions you need to import* `matplotlib.pyplot` *as earlier, and additionally* `Axes3D` *from* `mpl_toolkits.mplot3d`.

Try rotating the data. Can you find an angle where the three types of flower are separated in the plot?

4.2.7 Use the script `ex4_2_7.py` to plot the data matrix as an image as shown in Figure 3.23 in the book. The data matrix should be standardized to have zero mean and unit standard deviation.

Script details:

- *You can use the function* `imagesc()` *to plot an image.*
- *By default, the image will be smoothed, what is not always desired when you look at the data. Use parameter* `interpolation='None'` *to display raw data.*
- *The function* `zscore()` *can be used to standardize the data matrix.*

You are welcome to try out other plotting methods for the data. Matplotlib online repository is a good source of inspiration:
`matplotlib.sourceforge.net/gallery.html`

4.3 Visualizing Wine Data

We will in this part of the exercise consider two datasets related to red and white variants of the Portuguese "Vinho Verde" wine [1], the data has been downloaded from `http://archive.ics.uci.edu/ml/datasets/Wine+Quality`. Only physicochemical and sensory attributes are available, i.e., there is no data about grape types, wine brand, wine selling price, etc. The data has the following attributes:

Attributes 1–11 are based on physicochemical tests and attribute 12 on human judging. Later in the course we will attempt to predict whether a wine is a red or

#	Attribute	Unit
1	Fixed acidity (tartaric)	g/dm^3
2	Volatile acidity (acetic)	g/dm^3
3	Citric acid	g/dm^3
4	Residual sugar	g/dm^3
5	Chlorides	g/dm^3
6	Free sulfur dioxide	mg/dm^3
7	Total sulfur dioxide	mg/dm^3
8	Density	g/cm^3
9	pH	pH
10	Sulphates	g/dm^3
11	Alcohol	% vol.
12	Quality score	0–10

white wine based on these attributes and we will also attempt to predict the wine quality. Unfortunately, the data set has many observations that can be considered outliers and in order to carry out analyses later it is important to remove the corrupt observations.

The aim of this exercise is to use visualization to identify outliers and remove these outliers from the data. It might be necessary to remove some outliers before other outlying observations become visible. Thus, the process of finding and removing outliers is often iterative. The wine data is stored in a Matlab file, `Data/wine.mat`.

4.3.1 Inspect and run the script `ex4_3_1.py`. The scripts loads the data into Python using the `scipy.io.loadmat()` function, as in previous exercises. This dataset contains many observations that can be considered outliers and the visualization tools you have worked with in the previous exercise is used to identify the outliers in the data set. How many outliers are identified by the script? How are the identified outliers removed from the data set?

Script details:

- *You can use your solutions to the previous exercise as a starting point for making your visualizations.*
- *Say you want to find all data objects for which the alcohol percentage (attribute number 11) is not greater than 100%. You can mask them simply as* `mask=(X[:,10]<=100)`.

- *You can use the mask to eliminate the outlier observations (rows of data matrix). For instance you can write* `X=X[mask,:]` *where* `mask` *indicates the data objects that should be maintained. Remember also to remove them from the class index vector,* `y=y[mask,1]` *and to recompute* `N`*.*

We will later in the course attempt to classify the type of wine (white or red) as well as predict the quality of wine based on the physicochemical tests. Visual inspection of the data can give an indication of the difficulty of these tasks.

4.3.2 Are there any of the measurements that seem to be well suited in order to discriminate between red and white wines? What plots are particular useful in order to investigate this?

4.3.3 Does any of the 12 attributes appear to correlate with each other? What plots are well suited to investigate this?

4.3.4 Can you identify any clear relationship between the various physicochemical measurements of the wines and the quality of the wines as rated by human judges?

4.4 Tasks for the report

After today, you can address the last questions for part one of the report. Note if you have categoric variables you can still analyze these by PCA and other modeling approaches that assumes interval or ratio data types by converting them to one-out-of-K coding. The function `categoric2numeric` converts a column vector of a categoric attribute to numeric using one-out-of-K coding, type help(categoric2numeric) to learn more.

- **Data visualization(s) based on suitable visualization techniques including a principal component analysis (PCA).**
  Touch upon the following subjects, use visualizations when it appears sensible. *Keep in mind the ACCENT principles and Tufte's guidelines when you visualize the data.*
  - Are there issues with outliers in the data,
  - do the attributes appear to be normal distributed,

  - are variables correlated,
  - does the primary machine learning modeling aim appear to be feasible based on your visualizations.

- **A discussion explaining what you have learned about the data.** Summarize here the most important things you have learned about the data and give also your thoughts on whether your primary machine learning aim appears to be feasible based on your visualization.

1 Homework problems for this week

Problems

Question 1. Fall 2014 question 8: A factory produces cars. We consider three properties of the cars produced by the factory and each property can only take two values:

- The *color* which can be either *red* or *blue*.
- The *weight* which can be either *heavy* or *light*.
- The *model* which can be either *2-doors* or *4-doors*.

There are thus $2^3 = 8$ possible car types such as *(red, heavy,2-doors)* or *(blue,light,4-doors)*.

Suppose you are given the following information about cars produced from the factory:

- The probability a car has four doors is 0.5
- The probability a car is heavy given it has four doors is 0.8
- The probability a car is heavy given it has two doors is 0.2
- The probability a car is heavy and red is 0.1

Given the above information, what is the probability a car is blue given it is heavy?

A 0.2

B 0.5

C 0.8

D 0.9

E Don't know.

Question 2. Fall 2013 question 15: Based on Haberman's Survival Data found in Table 1 it is found:

- 56 pct. of the subjects had positive axillary nodes detected.
- 36 pct. of the subjects that had positive axillary nodes detected survived.
- 14 pct. of the subjects that did not have positive axillary nodes survived.

What is the probability that a subject that has survived would have positive axillary nodes according to the study by Haberman?

No.	Attribute description	Abbrev.
x_1	Young (< 60 years), $x_1 = 0$ or Old (≥ 60 years), $x_1 = 1$	Age
x_2	Operated before, $x_2 = 0$ or after 1960, $x_2 = 1$	OpT
x_3	Positive axillary nodes detected No, $x_3 = 0$ or Yes, $x_3 = 1$	PAN
y	Lived after 5 years No, $y = 0$ or Yes, $y = 1$	Surv

Table 1: A modified version of Haberman's Survival Data taken from http://archive.ics.uci.edu/ml/machine-learning-databases/haberman/haberman.names. The attributes x_1-x_3 denoting the age, operation time and cancer size as well as the output denoting survival after five years are binary. The data contains a total of $N = 306$ observations.

A 20.2 %

B 36.0 %

C 56.0%

D 76.6%

E Don't know.

Question 3. Fall 2012 question 2: We will only use the attributes x_1–x_{10} as well as the output y in our modeling of the data. The attributes x_1–x_{10} are standardized (i.e., the mean has been subtracted each attribute and the attributes divided by their standard deviations). In Figure 1 a boxplot of the standardized data

is given. Which of the following statements is *correct*?

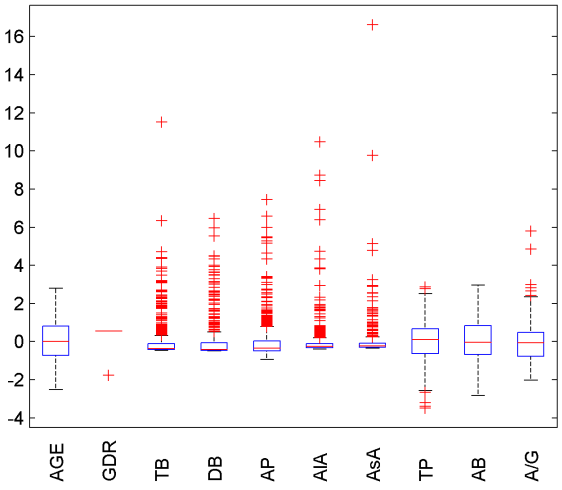

Figure 1: Boxplots of the 10 attributes x_1–x_{10} where the data has been standardized.

A The value of the 50th and 75th percentiles of the attribute DB coincides.

B Even though the distribution of AlA and AsA may have a similar shape this does not imply that the two attributes are correlated.

C The attribute TB is likely to be normal distributed.

D The attribute GDR has a clear outlier that should be removed.

E Don't know.

References

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

Decision trees and linear regression with PYTHON

Objective: The objective of this exercise is to become familiar with fitting decision trees and linear regression models in Python.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 5 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Make sure that you have `sklearn` package installed (we have already used it during the week 2 exercises). `Sklearn` package and documentation can be found here:
`http://www.scikit-learn.org/stable/install.html`
To visualize decision trees, we will need GraphViz editor, you can find it here: `http://www.graphviz.org/` Below you can see how graphviz is installed properly on your computer. Remember to restart Spyder before this takes into effect.

Linux

You just need to paste following snips in your terminal

```
sudo apt-get install graphviz
conda install graphviz
```

and you are done!

Mac

You need to install the python package. Do it from your terminal with following snip

```
conda install graphviz
```

and the graphviz executables also from your terminal

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Windows

You need to download graphviz from graphviz.org (clickable). Unzip the folder and note the path where you store it—you need to input the path in the script later.

Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.

5.1 Decision Trees

In this part of the exercise we will fit decision trees using the Python class `sklearn.tree.DecisionTreeClassifier`.

As a splitting criterion, the function uses one of the two impurity measures:

$$\text{gdi}(t) = 1 - \sum_{i=0}^{c-1} p(i|t)^2 \qquad \text{equivalent to Gini}(t),$$

$$\text{deviance}(t) = -2 \sum_{i=0}^{c-1} p(i|t) \log p(i|t) \quad \text{equivalent to Entropy}(t).$$

We will analyze the vertebrate data given in Table 4.1 of "Introduction to Data Mining" [1] as well as the wine data we have used in the previous exercise (this data is similar to the data introduced in the chapter on "Tree-based methods" in the book "Introduction to Machine Learning and Data Mining").

The vertebrate data set has the following attributes, which are all categorical:

Vertebrate dataset

#	Attribute	Unit
1	Body temperature	0:Cold blooded, 1:Warm blooded.
2	Skin cover	0:None, 1:Hair, 2:Scales, 3:Feathers, 4:Fur, 5:Quills
3	Gives birth	0:No, 1:Yes.
4	Aquatic creature	0:No, 1:Yes, 2:Semi.
5	Aerial creature	0:No, 1:Yes.
6	Has legs	0:No, 1:Yes.
7	Hibernates	0:No, 1:Yes.

The wine data set have the following attributes, where 1–11 are continuous and 12 is categorical:

Wine dataset

#	Attribute	Unit
1	Fixed acidity (tartaric)	g/dm^3
2	Volatile acidity (acetic)	g/dm^3
3	Citric acid	g/dm^3
4	Residual sugar	g/dm^3
5	Chlorides	g/dm^3
6	Free sulfur dioxide	mg/dm^3
7	Total sulfur dioxide	mg/dm^3
8	Density	g/cm^3
9	pH	pH
10	Sulphates	g/dm^3
11	Alcohol	% vol.
12	Quality score	0–10

5.1.1 Examine and run the script `ex5_1_1.py`, which contains the vertebrates data given in Table 4.1 of "Introduction to Data Mining".

5.1.2 Examine and run the script `ex5_1_2.py`. The script fits a decision tree to the vertebrates data using the Gini splitting criterion and stops splitting only when nodes are pure.

Script details:

- *Import module* `tree` *from* `sklearn` *and type* `help(tree.DecisionTreeClassifier)` *to learn how to fit a decision tree in Python.*
- *The parameter* `criterion` *can be used to choose the splitting criterion (Gini or Entropy).*
- *The parameters* `min_samples_split`, `min_samples_leaf`, *and* `max_depth` *influence the stopping criterion.*
- *After fitting the tree, export its structure to GraphViz format. The generated file (here:* `tree_gini.gvz`*) can be visualized externally with i.e.* `Graphviz` *editor or* `dot` *command-line tool or automatically rendered in Spyder as done in the script.*
- *The parameter* `feature_names` *can be used during the exporting to supply the names of the attributes.*

Did you get a similar result to the figure 1? Try to understand the information provided by the tree.

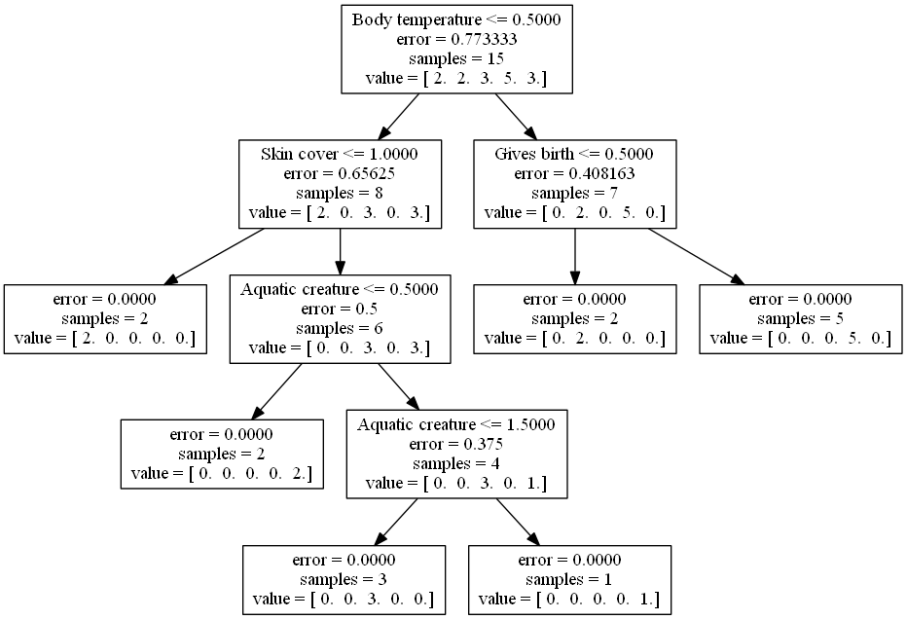

Figure 1: Decision tree of the vertebrates data

5.1.3 Examine and run the script `ex5_1_3.py`. The script shows how the tree changes when you use the Entropy (i.e. `deviance`) splitting criterion instead of the Gini (`gdi`).

Script details:

- *The parameter* `criterion` *can be used to choose the splitting criterion used.*

5.1.4 Use the script `ex5_1_4.py` to show that a dragon, which is cold-blooded, has scales, gives birth, is semi aquatic, is an aerial creature, has legs, and hibernates, will be classified as a reptile.

Script details:

- *After fitting decision tree, you can use the method* `predict()` *of the tree object to classify a new data object.*

5.1.5 Load the wine data set into Python using `ex5_1_5.py`.

The script removes outliers from the data set. Outliers are defined as data objects where Volatile acidity > 20 g/dm^3, Density > 10 g/cm^3, or Alcohol $> 200\%$.

The script also removes attribute 12, Quality score, from the data set, in order to use only the quantitative measurements for the prediction.

Script details:

- *You have worked with the wine data before in exercise 4.2.1 You can use your solution to that exercise as a starting point.*
- *To select a subset of data objects from the data matrix, you can write* `X=X[n,:]` *where* `n` *is either a vector of indices of the data objects to be selected, or a binary mask. Remember to update the class index vector accordingly* `y=y[n,:]`*, and to recompute* `N`*.*
- *Similarly you can select a subset of the attributes from the data matrix* `X=X[:,m]` *where* `m` *is either a vector of indices to the attributes, or a binary mask. Remember to update the list of attribute names accordingly, and to recompute the number of attributes* `M`*.*

You should end up with a data matrix of size $N \times M = 6304 \times 11$.

5.1.6 Inspect and run the script `ex5_1_6.py`. The script fits a decision tree to the wine data in order to estimate if the wine is red or white. The script uses the Gini splitting criterion and the following parameter value as stopping criterion `min_samples_split=100`.

Script details:

- *The solution to exercise 5.1.2 is used as a starting point.*

Explain what happens when you change the values of the parameter `min_samples_spli`

Observe the changes in the tree size after reducing the parameter.

5.1.7 Explain how the script `ex5_1_7.py` shows that a wine with the following attribute values would be classified as White.

#	Attribute	Value
1	Fixed acidity (tartaric)	6.9
2	Volatile acidity (acetic)	1.09
3	Citric acid	0.06
4	Residual sugar	2.1
5	Chlorides	0.0061
6	Free sulfur dioxide	12
7	Total sulfur dioxide	31
8	Density	0.99
9	pH	3.5
10	Sulphates	0.44
11	Alcohol	12

5.2 Linear and logistic regression

Regression is widely used in datamining and machine learning for predicting an output (dependent variable) for a data object given its attributes. Regression can be defined as the task of learning a target function that maps a set of attributes onto an output.

We will initially consider a linear regression problem where we will model the outputs, $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]^\top$, based on a single attribute for each data object, $\boldsymbol{X} = [x_1, x_2, \ldots, x_N]^\top$, using the model

$$y_n = w_0 + w_1 x_n + \epsilon_n,$$

where ϵ_n is residual noise, w_0 a constant offset and w_1 the linear coefficient for x_n. We can write an equation for each data object,

$$\begin{aligned} y_1 &= w_0 + w_1 x_1 + \epsilon_1 \\ y_2 &= w_0 + w_1 x_2 + \epsilon_2 \\ &\vdots \\ y_N &= w_0 + w_1 x_N + \epsilon_N. \end{aligned}$$

Introducing a vector of all ones $\mathbf{1} = [1, 1, \ldots, 1]^\top$ we can write the N equation compactly in matrix-vector form

$$\boldsymbol{y} = w_0 \mathbf{1} + w_1 \boldsymbol{x} + \boldsymbol{\epsilon} = \begin{bmatrix} \mathbf{1} & \boldsymbol{x} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} + \boldsymbol{\epsilon}.$$

In order to estimate w_0 and w_1 we minimize the least squares error function, i.e.

$$(w_0, w_1) = \arg\min_{w_0,w_1} \sum_{n=1}^{N} (y_n - (w_0 + w_1 x_n))^2 = \arg\min_{w_0,w_1} \left\| \boldsymbol{y} - \begin{bmatrix} \mathbf{1} & \boldsymbol{x} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \right\|_F^2 .$$

Based on our linear regression model we can predict the value of an unobserved output y^* from an observed measurement x^* as

$$y^* = w_0 + w_1 x^*.$$

5.2.1 Inspect and run the script `ex5_2_1.py`. The script generates 100 data objects according to the linear regression model,

$$y_n = w_0 + w_1 x_n + \epsilon_n,$$

Each of the n observations (having one attribute value x_n) should be set to $x_1 = 0, x_2 = 1, x_3 = 2, \ldots, x_N = N - 1$, and ϵ_n is generated at random from a Normal distribution with mean zero and standard deviation 0.1. Inspect the script and verify the parameters of the model are $w_0 = -0.5$ and $w_1 = 0.01$.

Script details:

- *You can use functions* `range()` *or* `linspace()` *to make a sequence of integer or real numbers.*
- *You can generate Normal distributed random numbers using the function* `random.randn()`.

Make a scatter plot of your data. Does it look like the plot in figure 2?

5.2.2 Inspect and run the script `ex5_2_2.py`. Explain how the script estimates the model parameters, w_0 and w_1, from the data you have generated. Show that the estimated parameters are close but not exactly equal to the parameters used to generate the data.

Notice how the script plots predictions of the model as well as the predictions you get using the parameters used to generate the data in the same graph as the scatter plot you made in exercise 5.2.1.

Script details:

- *You can use the class* `sklearn.linear_model.LinearRegression` *as your model. Set* `fit_intercept` *parameter to True, unless you have added column of "1"s to your data matrix X.*

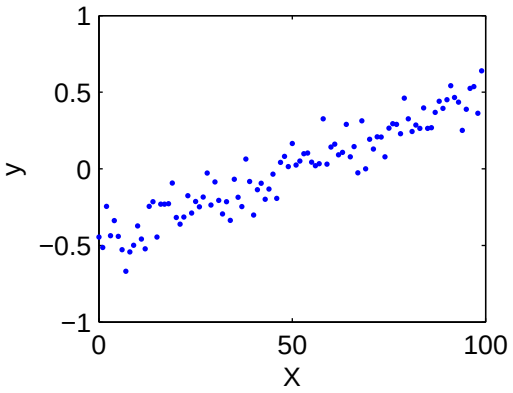

Figure 2: Scatter plot of data

- *Use method* `fit()` *to estimate parameters in the linear model, and method* `predict()` *to predict values for new data points.*

Show that the estimated model parameters get closer to the parameters used to generate the data if you include more data objects or reduce the noise variance.

Sometimes there are more complex relationships between the attributes and output. In order to capture more complex relationships we can include attributes that are specified transformations of existing attributes such as $\sqrt{x}$, $\log(x)$, x^2 and x^3. We will presently consider the model

$$y_n = w_0 + w_1 x_n + w_2 x_n^2 + \epsilon_n$$

The above model can again be written for all N observations simultaneously in terms of vectors and matrices, i.e.

$$\boldsymbol{y} = [\mathbf{1}\ \boldsymbol{x}_1\ \boldsymbol{x}_2] \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} + \boldsymbol{\epsilon} = [\mathbf{1}\ \boldsymbol{X}]\boldsymbol{w} + \boldsymbol{\epsilon}$$

where $\boldsymbol{x}_1 = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$, $\boldsymbol{x}_2 = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \vdots \\ x_N^2 \end{bmatrix}$ and $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$.

5.2.3 Inspect and run the script `ex5_2_3.py` which generates a data set according to the linear regression model including the squared term, fits the model

and makes a plot as in the previous exercise. The script allows you to specify separately how many terms to include (squared, cubed, etc.) for generating the data and fitting the model. Illustrate what happens when fitting a model with more higher order terms? How well is the model able to fit the data?

We will return to the wine data and attempt to predict the alcohol content using linear regression. The linear regression model we consider is

$$y_n = w_0 + \sum_k w_k x_{n,k} + \epsilon_n,$$

where y_n is the observed alcohol content, ϵ_n the model residual, w_k the value of the estimated model coefficient for the k^{th} attribute whereas $x_{n,k}$ denotes the n^{th} observation of the k^{th} attribute. As such, $x_{2,1}$ denotes the second observation of the first attribute (Fixed acidity) and $x_{5,2}$ the fifth observation of the second attribute (Volatile acidity).

5.2.4 Inspect and run the script `ex5_2_4.py`. The script loads the Wine data into Python using the solution to exercise 5.1.5. The script also fits a linear model that predicts the Alcohol attribute based on the 10 remaining attributes (again excluding Quality score) and makes a scatter plot of the true versus the predicted alcohol content. Explain how the visualizations show that the alcohol content can be predicted with an accuracy of approximately ± 1 percentage point.

Script details:

- *You can take similar approach as in exercises 5.2.2 and 5.2.3 to fit linear regression model and use it to predict values.*

Show from the estimated parameters of the linear regression model that wines with higher alcohol content in general have lower density.

5.2.5 Inspect and run the script `ex5_2_5.py`. The script introduces more variables in the regression by combining or transforming existing variables. Explain how that is done in the script and which transformations are used. Try to introduce your own transformations.

Script details:

- *For example you can include the square of the Fixed acidity in the model and the square of the Volatile as well as a multiplicative interaction between Fixed acidity and Volatile acidity. You can use* `numpy.power()` *and* `numpy.multiply()` *functions on appropriate columns of data matrix.*
- *Function* `numpy.bmat()` *is useful to concatenate matrices and/or vectors, for instance:* `X=np.bmat('X, Xfa2, Xva2, Xfava')`.
- *Plots of attributes versus the model residuals can reveal if there is structure in the output that can be explained by transformations of the attributes.*

How low can you get the mean squared error?

Until now we have discussed the most simple linear model based on the squared error cost function. In the generalized linear model, we can also use other cost functions as well as non-linear link functions. For instance, for outputs that are purely binary, $y \in \{0, 1\}$, the logistic function is very suitable, $\text{logistic}(s) = \frac{1}{1+exp(-s)}$, which maps the output in the regression analysis to the range (0,1). In figure 3 there is a plot of the logistic link function.

It can be very useful to limit the possible outputs to the range (0,1) if the outputs we wish to predict are binary. Predicting binary outputs is what we have previously referred to as classification, and generalized linear models can thus both be used for regression and classification. Classification with a binomial cost function and logistic link function is called *logistic regression*, and in Python is implemented in module: `sklearn.linear_model.logistic.LogisticRegression`

We will return to the wine data and predict the type of wine using logistic regression. The model we consider is

$$y_n = \text{logistic}\left(w_0 + \sum_k w_k x_{n,k}\right) + \epsilon_n$$

where y_n is the observed type of wine (Red: $y_n = 0$ and White: $y_n = 1$). We will say that a wine is White with probability p and Red with probability $1 - p$. where $p = \text{logistic}(w_0 + \sum_k w_k x_{n,k})$.

5.2.6 Inspect and run the script `ex5_2_6.py`. The script loads the Wine data into Python using the solution to exercise 5.1.5 and fits a logistic regression model that predicts the type of wine (Red or White) based on the attributes 1-11 (again excluding Quality score). Explain how this is done.

Show that the wine defined in exercise 5.1.7 would be classified as White with more than 90% probability.

Script details:

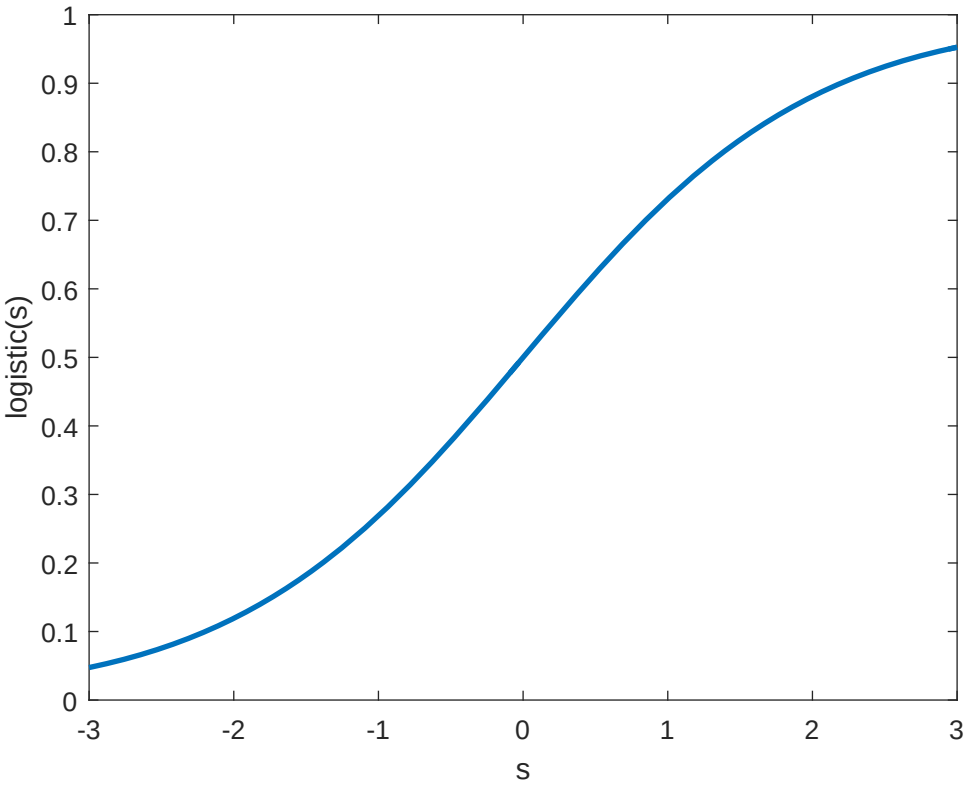

Figure 3: Plot of the logistic link function.

- *To estimate a logistic regression model, use the class:* `sklearn.linear_model.logistic.LogisticRegression`.
- *To make a prediction from the estimated model, use the function* `predict()`. *If you want to get probabilities of new data object belonging to given class use* `predict_proba()`.

5.3 Tasks for the report

The report will contain three sections, two about regression and one about classification. For the regression section, you will now be able to address this question:

Regression, part a: In this section, you are to solve a relevant regression problem for your data and statistically evaluate the result. We will begin by examining the most elementary model, namely linear regression.

- Explain what variable is predicted based on which other variables and what you hope to accomplish by the regression. Mention your feature transformation choices such as one-of-K coding. Since we will use regularization momentarily, apply a feature transformation to your data matrix $\mathbf{X}$ such that each column has mean 0 and standard deviation 1[1].

Meanwhile, for the **classification** section:

- Explain which classification problem you have chosen to solve. Is it a multi-class or binary classification problem?

1 Homework problems for this week

[1]We treat feature transformations and linear regression in a very condensed manner in this course. Note for real-life applications, it may be a good idea to consider interaction terms and the last category in a one-of-K coding is redundant (you can perhaps convince yourself why). We consider this out of the scope for this report

Problems

Question 1. Fall 2013 question 14: We consider a dataset on survival of breast cancer taken from `http://archive.ics.uci.edu/ml/machine-learning-databases/haberman/haberman.names`. The data has been binarized as outlined in Table 1. The dataset contain a total of 306 observations. In the dataset 81 survived after 5 years (i.e., $y = 1$) whereas 225 died (i.e., $y = 0$). We would like to build a decision tree and consider using positive axillary nodes detected (PAN) as an attribute condition at the root of the tree. We thereby split according to whether positive axillary nodes were detected and find:

- For the 170 subjects that had positive axillary nodes detected 62 survived.
- For the 136 subjects that did not have positive axillary nodes 19 survived.

What is the gain, Δ, of splitting according to whether a subject had positive axillary nodes (PAN) using the Gini as impurity measure $I(t)$, (i.e., $I(t) = 1 - \sum_{i=0}^{C-1} p(i|t)^2$)?

No.	Attribute description	Abbrev.
x_1	Young (< 60 years), $x_1 = 0$ or Old (≥ 60 years), $x_1 = 1$	Age
x_2	Operated before, $x_2 = 0$ or after 1960, $x_2 = 1$	OpT
x_3	Positive axillary nodes detected No, $x_3 = 0$ or Yes, $x_3 = 1$	PAN
y	Lived after 5 years No, $y = 0$ or Yes, $y = 1$	Surv

Table 1: A modified version of Haberman's Survival Data taken from http://archive.ics.uci.edu/ml/machine-learning-databases/haberman/haberman.names. The attributes x_1-x_3 denoting the age, operation time and cancer size as well as the output denoting survival after five years are binary. The data contains a total of $N = 306$ observations.

A -0.025

B 0

C 0.025

D 0.036

E Don't know.

Question 2. Spring 2013 question 12: We fit a linear regression model to the PM10 data shown in Table 2. The input attributes are standardized (i.e., we have subtracted the mean of each input attribute, x_1–x_7, and divided by their standard deviations) whereas the output logPM10 is kept in its original format. We obtain the following model:

$$f(\boldsymbol{x}) = 3.27 + 0.36x_1 - 0.01x_2 - 0.19x_3 + 0.01x_4 + 0.05x_7.$$

Which one of the following statements about the model is *incorrect*?

No.	Attribute description	Abbrev.
x_1	Logarithm of number of cars per hour	logCAR
x_2	Temperature 2 meter above ground (degree Celsius)	TEMP
x_3	Wind speed (meters/second)	WIND
x_4	Temperature difference between 25 and 2 meters (degree Celsius)	TEMPDIF
x_5	Wind direction (degrees between 0 and 360)	WINDDIR
x_6	Whole hour of the day	HOUR
x_7	Day number from October 1. 2001	DAY
y	Logarithm of PM10 concentration	logPM10

Table 2: The attributes of the PM10 data. The output is given by the hourly values of the logarithm of the concentration of PM10 particles (logPM10).

A According to the model WINDDIR and HOUR are not relevant for predicting the pollution level.

B According to the model fewer cars and more wind will result in lower pollution levels.

C According to the model it seems that pollution is decreasing over time.

D According to the model higher temperatures will result in lower pollution levels.

E Don't know.

Question 3. Spring 2014 question 5: We consider the Wholesale dataset shown in Table 3 and wish to predict whether a consumer is from Lisbon (y=0) or Oporto (y=1) by discarding observations from the Other region included in the wholesale data. After discarding the observations pertaining to the Other region we standardize the attributes x_1–x_6 (i.e., for each attribute subtract the mean and divide by the standard deviation) and fit a logistic regression model. We obtain the following model for the prediction of the origin of the consumer:

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = \text{logit}(-0.51 - 0.11x_1 - 0.36x_2 + 0.44x_3 + 0.39x_4 + 0.09x_5 - 0.28x_6),$$

where $\text{logit}(w) = \frac{1}{1+\exp(-w)}$ is the logit function. Which one of the following statements about the model is *correct*?

No.	Attribute description	Abbrev.
x_1	Fresh products	FRESH
x_2	Milk products	MILK
x_3	Grocery products	GROCERY
x_4	Frozen products	FROZEN
x_5	Detergents and paper products	PAPER
x_6	Delicatessen products	DELI
y	Region	REGION

Table 3: The six input attributes x_1–x_6 denoting the annual consumption in monetary units of customers as well as the output y denoting which of the three regions; Lisbon, Oporto, and one additional region denoted Other, the customers came from in the wholesale customer data.

A According to the model it seems that people in Lisbon buy more FRESH products, MILK products and DELI products than people in Oporto.

B According to the model if a costumer after the standardization has $x_1 = x_2 = x_3 = x_4 = x_5 = x_6 = 0$ the customer is more likely to come from Oporto than Lisbon.

C The logit function will return the probability a person is from Lisbon.

D From the model it can be seen that FRESH and PAPER are unimportant and should be removed in order to avoid overfitting.

E Don't know.

References

[1] Pang-Ning Tan et al. *Introduction to data mining*. Pearson Education India, 2006.

Overfitting, cross-validation and Nearest Neighbor with PYTHON

Objective: The objective of this exercise is to understand how cross-validation can be used to avoid overfitting as well as the k-nearest neighbor method.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 6 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`*_train`	—	—	Training data.
	`*_test`	—	—	Test data.

6.1 Decision tree pruning using cross-validation

In this exercise we will use cross-validation to prune a decision tree. When applying cross-validation the observed data is split into training and test sets, i.e., `X_train`, `y_train` and `X_test` and `y_test`. We train the model on the training data and evaluate the performance of the trained model on the test data.

6.1.1 Inspect and run the script `ex6_1_1.py`. The script load the `wine2.mat` file with wine data using the `loadmat()` function. In this version of the wine data, outliers have already been removed. Notice how the script divides the data into a training and a test data set. Now, we want to find optimally pruned decision tree, be modifying its maximum depth. For different values of parameter (depth from 2 to 20) explain how the script

fits the decision tree, and compute the classification error on the training and test set (holdout cross-validation). Notice how the script plot the training and test classification error as a function of the pruning level. What does this plot tell you?

Script details:

- *Take a look at the module* `sklearn.cross_validation` *and see how it can be used to partition the data into a training and a test set (holdout validation,* `train_test_split()` *function). Note, that the package contains also functions to partition data for K-fold cross-validation. Some of the functions can ensure that both training and test sets have roughly the same class proportions.*
- *Fit and train the classification tree similarly like in the previous week exercises, modify regularizing parameter in every iteration (here:* `max_depth`*)*

What appears to be the optimal tree depth? Do you get the same result when you run your code again, generating a new random split between training and test data? What other parameters of the tree could you optimize in cross-validation?

6.1.2 Inspect the script `ex6_1_2.py`. The script repeat the exercise above, using 10-fold cross-validation. To do this, the data set is divided into 10 random training and test folds. For each fold, a decision tree is fitted on the training set and it's performance is evaluated on the test set. Finally, the average classification error is computed across the 10 cross-validation folds.

Script details:

- *This time* `KFold()` *function from module* `sklearn.cross_validation` *can be used to partition the data into the 10 training and test partitions. It returns* `CV` *object through which you can iterate to obtain train/test indices at each fold.*

What appears to be the optimal tree depth? Do you get the same result when you run your code again, generating a new random split between training and test data? How about 100-fold cross-validation or leave-one-out cross-validation?

6.2 Variable selection in linear regression

In this exercise we consider cross-validation for variable selection and model performance evaluation in linear regression. We will try to predict the body-weight of a person based on a number of body measurements using linear regression with feature subset selection. The data is a subset of the data available at `http://www.sci.usq.`

edu.au/courses/STA3301/resources/Data/ described in [1]. To measure how well we can predict the body-weight, we will use the squared error between the true and estimated body-weight.

In our estimation we will use two levels of cross-validation: 1) On the outer level, we use 5-fold cross-validation to estimate the performance of our model, i.e., we compute the squared error averaged over 5 test sets. 2) On the inner level, we use 10-fold cross-validation to perform sequential feature selection (see figure 1).

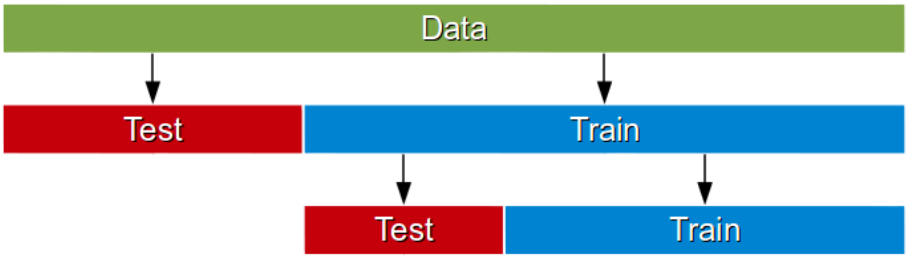

Figure 1: Multi-level cross validation

6.2.1 You can load the body data into Python with the command `loadmat('..\\Data\\body`
The data set contains data for the 23 attributes in the matrix `X` and the body-weight in `y`.

Inspect and run the script `ex6_2_1.py`. The script applies 5-fold cross-validation to the problem of fitting a linear regression model to estimate the body-weight based on the attributes. Explain how the script, when fitting the models, compares two methods: 1) using all 23 attributes, and 2) using 10-fold cross-validation to perform sequential feature selection, thus choosing a subset of the 23 attributes.

Explain how the script computes the 5-fold cross-validated training and test error with and without sequential feature selection. Explain how it can be seen that without feature selection, the model overfits. Explain how it can be seen the feature selection tends to choose features such as height and waist girth, and disregard features such as the wrist diameter, which seems reasonable when predicting body-weight.

Script details:

- *Again, you may use* `KFold()` *function to set up the crossvalidation partitions needed.*
- *To fit a linear regression model, use the* `sklearn.linear_model.LinearRegression` *class (methods* `fit()` *and* `predict()`*), as you did in the previous exercises.*
- *To perform sequential features selection with linear regression model and k-fold cross-validation you can use the function* `feature_selector_lr()` *from the 02450 toolbox. Type* `help(feature_selector_lr)` *to read how it works, or give a closer look at its implementation in* `toolbox_02450.py` *file.*

Optional: Try modifying the solution to use backward feature subset selection. Does it give the same result? If you are interested in other methods for feature selection, have a look at module `sklearn.feature_selection`.

6.3 K-nearest neighbor classification

In this exercise we will use the k-nearest neighbors (KNN) method for classification. First, we will consider 4 different synthetic datasets, that can be loaded into Python using the `loadmat` function. The data is stored in files `Data/synth1`, ..., `Data/synth4`.

6.3.1 Consider the script `ex6_3_1.py`. For each of the four synthetic datasets, do the following. Load the dataset into Python and examine it by making a scatter plot. Classify the test data `X_test` using a k-nearest neighbor classifier. Choose a distance measure (consider the following distance measures: `euclidean`, `cityblock`). Choose a suitable number of neighbors. Examine the accuracy and error rate.

Script details:

- *The Python class* `KNeighborsClassifier` *from* `sklearn\neighbors` *module can be used to perform k-nearest neighbors classification.*
- *To generate a confusion matrix, you can use the function* `confusion_matrix()` *function from module* `sklearn.metrics` *in the course toolbox. You can use* `imshow()` *function to plot the confusion matrix.*

Which distance measures worked best for the four problems? Can you explain why? How many neighbors were needed for the four problems? Can you give an example of when it would be good to use a large/small number of neighbors? Consider e.g. when clusters are well separated versus when they are overlapping.

In general we can use cross-validation to select the optimal distance metric and number of nearest neighbors k although this can be computationally expensive. We will return to the Iris data we have considered in previous exercises, and attempt to classify the Iris flowers using KNN.

6.3.2 Consider the script `ex6_3_2.py`. The script loads the Iris data into Python. Explain how the script uses leave-one-out crossvalidation to estimate the number of neighbors, k, for the k-nearest neighbors classifier and plots the crossvalidated average classification error as a function of k for $k = 1, \ldots, 40$.

Script details:

- *To load the Iris data, you can run your solution to exercise 4.1.1.*
- *Use* `LeaveOneOut` *crossvalidation from module* `sklearn.cross_validation`.
- *As before, use the* `KNeighborsClassifier` *class for k-nearest neighbors classification.*

6.3.3 Discussion: What are the benefits and drawbacks of K-nearest neighbor classification and regression compared to logistic regression, decision trees and linear regression? (Hint: There are two important aspects of classification and regression methods, how well the methods can *predict* unlabeled data and how well the method *describe* what aspects in the data causes the data to be classified a certain way .)

6.4 Task for the report

The report will make use of cross-validation, but in conjunction with methods we have not seen yet. Please see report description for more information.

1 Homework problems for this week

Problems

Question 1. Fall 2014 question 27: Alice is considering a linear regression model for a dataset comprised of $N = 1000$ observations. She wishes to both select the optimal regularization strength as well as estimate the generalization error of the model at the optimal regularization strength. To simplify the problem, she only considers the following 6 possible values of the regularization strength λ:

$$\lambda = 10^{-2},\ 10^{-1},\ 10^{0},\ 10^{1},\ 10^{2},\ 10^{3}.$$

Alice opts for a two-level strategy in which she uses the hold-out method to estimate the generalization error and cross-validation is used to select the optimal regularization strength, i.e. the dataset is first divided into a validation set $D_{\text{validation}}$, comprised of 20% of the full dataset, and the remainder D_{CV} is used for cross-validation. Alice uses standard $K = 10$ fold cross-validation to select the optimal regularization strength on D_{CV} and, having estimated the optimal regularization strength, uses the hold-out method on D_{CV} and $D_{\text{validation}}$ to estimate the generalization error.

Suppose for any fixed value of the regularization strength, the time taken to *train* the weights of the linear regression model on a dataset of size N_{train} is N_{train}^2 units of time and the time taken to *test* a trained model on a dataset of size N_{test} is $\frac{1}{2}N_{\text{test}}^2$ units of time. Suppose the duration of all other tasks is neglible, what is the total time taken for the entire procedure?

A $12.78 \cdot 10^6$ units of time.

B $15.98 \cdot 10^6$ units of time.

C $31.30 \cdot 10^6$ units of time.

D $31.96 \cdot 10^6$ units of time.

E Don't know.

Question 2. Spring 2013 question 13: We would like to fit an artificial neural network to the PM10 dataset shown in Table 2. It is decided that DAY should not be included in the model as this cannot be influenced by decision makers. We therefore only consider x_1, x_2, x_3 and x_4 corresponding to logCAR, TEMP, WIND and TEMPDIF respectively. An artificial neural network is applied to the data with these four attributes. The neural network has three hidden units and is trained using different combinations of the four attributes x_1, x_2, x_3 and x_4. Table 1 gives the training and test performance of the artificial neural network for different combinations of the four attributes. Which one of the following statements is *correct*?

Feature(s)	Training rmse	Test rmse
x_1	0.71	0.75
x_2	0.58	0.64
x_3	0.60	0.62
x_4	0.92	0.94
x_1 and x_2	0.60	0.69
x_1 and x_3	0.35	0.44
x_1 and x_4	0.52	0.66
x_2 and x_3	0.56	0.69
x_2 and x_4	0.45	0.52
x_3 and x_4	0.62	0.64
x_1 and x_2 and x_3	0.36	0.34
x_1 and x_2 and x_4	0.28	0.33
x_1 and x_3 and x_4	0.27	0.45
x_2 and x_3 and x_4	0.20	0.43
x_1 and x_2 and x_3 and x_4	0.10	0.35

Table 1: Root mean square error (rmse) for the training and test set when using an artificial neural network with three hidden units to predict the level of pollution (logPM10) based only on the first four attributes (x_1–x_4) using the hold-out method with 50 % of the observations hold-out for testing.

No.	Attribute description	Abbrev.
x_1	Logarithm of number of cars per hour	logCAR
x_2	Temperature 2 meter above ground (degree Celsius)	TEMP
x_3	Wind speed (meters/second)	WIND
x_4	Temperature difference between 25 and 2 meters (degree Celsius)	TEMPDIF
x_5	Wind direction (degrees between 0 and 360)	WINDDIR
x_6	Whole hour of the day	HOUR
x_7	Day number from October 1. 2001	DAY
y	Logarithm of PM10 concentration	logPM10

Table 2: The attributes of the PM10 data. The output is given by the hourly values of the logarithm of the concentration of PM10 particles (logPM10).

A Neither forward nor backward selection will identify the optimal feature combination for this problem.

B Backward selection will result in a better model being selected than using forward selection.

C Backward selection will use a model that include all the features x_1, x_2, x_3, and x_4.

D Forward selection will select the features x_1, x_2 and x_4.

E Don't know.

Question 3. Fall 2013 question 9: In order to predict if an observation corresponds to a relatively small or large island we will use a k-nearest neighbor (KNN) classifier based on the Euclidean distance between the eight observations given in Table 3. We will use leave-one-out cross-validation for the KNN in order to classify whether the eight considered observations constitute small islands (given in red, i.e. observation O1, O2, O3, O4) or large island (given in blue, i.e. observation O5, O6, O7, O8) using a three-nearest neighbor classifier, i.e. $K = 3$. The analysis will be based only on the data given in Table 3. Which one of the following statements is *correct*?

	O1	O2	O3	O4	O5	O6	O7	O8
O1	0	2.39	1.73	0.96	3.46	4.07	4.27	5.11
O2	2.39	0	1.15	1.76	2.66	5.36	3.54	4.79
O3	1.73	1.15	0	1.52	3.01	4.66	3.77	4.90
O4	0.96	1.76	1.52	0	2.84	4.25	3.80	4.74
O5	3.46	2.66	3.01	2.84	0	4.88	1.41	2.96
O6	4.07	5.36	4.66	4.25	4.88	0	5.47	5.16
O7	4.27	3.54	3.77	3.80	1.41	5.47	0	2.88
O8	5.11	4.79	4.90	4.74	2.96	5.16	2.88	0

Table 3: Pairwise Euclidean distance, i.e $d(Oa, Ob) = ||\boldsymbol{x}_a - \boldsymbol{x}_b||_2 = \sqrt{\sum_m (x_{am} - x_{bm})^2}$, between eight observations of the Galápagos data. Red observations (i.e., O1, O2, O3, and O4) correspond to the four smallest islands whereas blue observations (i.e., O5, O6, O7, and O8) correspond to the four largest islands.

A The error rate of the classifier will be 1/8

B The error rate of the classifier will be 1/4

C The error rate of the classifier will be 3/8

D The error rate of the classifier will be 1/2

E Don't know.

References

[1] Grete Heinz, Louis J Peterson, Roger W Johnson, and Carter J Kerk. Exploring relationships in body dimensions. *Journal of Statistics Education*, 11(2), 2003.

Performance evaluation, Bayes, and Naive Bayes with PYTHON

Objective: The objective of today's exercise is to estimate confidence intervals for the performance of both regression and classification models, to compare two regression and two classification models against each other, and finally to introduce the Naïve Bayes classification method.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 7 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	$\star$`_train`	—	—	Training data.
	$\star$`_test`	—	—	Test data.

7.1 Statistical evaluation of classifiers

In this part of the exercise, we will examine how to evaluate and compare classification methods. The tasks we will be concerned with solving is, firstly, how to determine a reasonable interval $[\theta_L, \theta_U]$ for the accuracy of a single classifier based on it's prediction on a training set and secondly, how to compare two classifiers by estimating reasonable bounds for the difference of their accuracy $\theta = \theta_A - \theta_L$.

For simplicity, we will focus on comparing two k-nearest neighbor methods on the Iris dataset. Since this dataset consist of only $N = 150$ observations, we will use leave-one-out cross validation to construct the $n = N$ model predictions.

7.1.1 In this problem, we will evaluate the performance of a single model. The models we will consider are k-nearest neighbor classification models for different values of k. Specifically, we will consider $k = 1$, $k = 20$ and $k = 80$ to obtain three models, $\mathcal{M}_A$, $\mathcal{M}_B$ and $\mathcal{M}_C$, with a reasonably different performance. As a first step, inspect and run `ex7_1_1.py`. Note it computes and saves the predictions of all three models as `yhat`. Modify the script to compute the accuracy of model $\mathcal{M}_A, \mathcal{M}_B$ and $\mathcal{M}_C$. What are your conclusions?

Script details:

- *This is simply a matter of comparing the predictions* `yhat` *with* `y_true` *and counting the fraction of times the model makes the right prediction.*
- *You should find that model $\mathcal{M}_C$ seems to perform relatively worse than $\mathcal{M}_A$ and $\mathcal{M}_B$.*

7.1.2 Inspect and run `ex7_1_2.py`. The script computes the Jeffrey interval of model $\mathcal{M}_A$ as defined in Box 11.3.1. Modify the script to compute the Jeffrey interval of all three models. What is your conclusion?

7.1.3 Try to change α to $\alpha = 0.1$ and $\alpha = 0.01$. What effect does this have on the Jeffrey interval? Explain how α influences $P(\theta \in [\theta_L, \theta_U])$.

Script details:

- *This is connected to the definition of the CDF. Under the assumption in the book leading to the definition of the Jeffrey interval it is easy to show*

$$P(\theta \in [\theta_L, \theta_U]) = 1 - P(\theta < \theta_L) - P(\theta > \theta_U)$$

and you should be able to relate the two quantities on the right-hand side to α.

7.1.4 Inspect and run `ex7_1_4.py`. The script attempts to estimate the difference in performance

$$\theta = \theta_A - \theta_B$$

between model $\mathcal{M}_A$ and $\mathcal{M}_B$ using McNemar's test (see lecture notes Box 11.3.2 for more details). Relate the quantities the script output to Box 11.3.2 in the lecture notes. You should find that $\hat{\theta} < 0$. Does this signify $\mathcal{M}_A$ is better than $\mathcal{M}_B$? How does the result relate to your conclusions in 7.1.2?

Script details:

- *You should find the confidence interval barely doesn't contain 0, which is weak evidence towards $\mathcal{M}_B$ having a relatively higher accuracy than $\mathcal{M}_A$. Meanwhile, the p-value is relatively high, indicating the result is likely due to chance. All in all the result is inconclusive and we should not conclude $\mathcal{M}_B$ is better than $\mathcal{M}_A$.*

7.1.5 Modify the script to compare $\mathcal{M}_A$ against $\mathcal{M}_C$. Based on these results, are there statistically good reasons to believe $\mathcal{M}_A$ is better than $\mathcal{M}_B$? Are there statistically good reasons to believe $\mathcal{M}_A$ is better than $\mathcal{M}_C$?

Script details:

- *As we saw in 7.1.1, there is a relatively higher difference in performance between $\mathcal{M}_A$ and $\mathcal{M}_C$, which is confirmed by McNemar's test. The performance difference θ is estimated to be between (approximately) 0.05 and 0.1. The confidence interval is therefore well clear of 0 and the low p-value ($p < 0.01$) indicates the result is not likely to be due to chance.*

7.1.6 Modify the script `ex7_1_1.py` to replace $\mathcal{M}_B$ with a classification tree model. Use McNemar's test to compare the decision tree against a KNN classifier with $k = 1$. Comment on the result of the evaluation. Is one model better than the other?

7.2 Statistical evaluation of a regression model

In this problem, we will statistically evaluate a regression model and compare two regression models. The problem will be based on the `wine` dataset, where the goal is to predict the alcohol content (in percent) from the other features. Since we do not have many candidate regression models to choose from at this point, we will compare the linear regression model against a regression tree. Please note regression trees are starred material in the course notes, and they can be treated as a black-box method for the purpose of this exercise.

7.2.1 Inspect and run the script `ex7_2_1.py`. Notice how the model use the hold-out method to generate a training/test split and then use the t-test to obtain a confidence interval for the regression model as in Box 11.3.3. Modify the script to obtain a confidence interval for the regression tree.

7.2.2 Continuing the above exercise, notice how the model compare the linear regression model and regression tree as described in Box 11.3.4. What is the confidence interval and p-value, and what would you conclude based on the results?

7.2.3 Modify the script to use $K = 10$-fold cross validation and compare the p-value and confidence interval to that obtained with the hold-out method. What changes do you expect to occur with the confidence interval when going from hold-out to K-fold to leave-one-out cross-validation? Re-do the exercise with L_1 loss as performance measure.

7.3 Statistical evaluation in **setup II** (Optional)

The past two problems have considered **setup I**, in which we are asking the narrow question which model perform better when trained on a particular data set denoted. That is, our conclusions only hold for the particular dataset set, and if we are therefore not justified in claiming our results hold for a new dataset drawn from the same statistical source. We refer the reader to section 11.1 which elaborates on this distinction. In this section, we will therefore consider a statistical evaluation which based on the correlation heuristic compares estimates of the true generalization error as described in section 11.4.

7.3.1 In this problem, we will once more consider the `wine` dataset, and attempt to compare a linear regression model and a regression tree. Inspect and run the script `ex7_3_1.py` which implements the method described in Box 11.4.1. Carefully examine how it computes the quantities that enter into the test. What is the p-value and confidence interval?

7.3.2 The script `ex7_3_1.py` can also compute the p-value and confidence interval using the methods in `ex7_2_1.py` by using K-fold cross-validation. Compare the p-values and confidence intervals with those you just obtained above, i.e. $J = K$. How do they differ and how do you explain this difference?

7.3.3 Try to alter the script to compare two classification models (you can use the KNN example above as inspiration).

7.4 Bayes and Naïve Bayes

In this part of the exercise we will classify names as female or male names `http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/corpora/names/`, for further details see also the readme_male_female.txt file in the Data folder. We have a database with 2943 male names and 5001 female names. We will only consider names that contain at least four letters resulting in a total of 2785 male and 4866 female names. As feature for the classification we will use the first and second letter as well as the second last and last letter of the names denoted respectively x_1, x_2, x_3 and x_4. Thus the name "Richard" will have $x_1 = r$, $x_2 = i$, $x_3 = r$, $x_4 = d$. Therefore, $x_i \in \{a, b, c, d, \ldots, z\}$ such that each feature can take the value of any of the 26 letters of the alphabet (from a to z) hence the attributes used are discrete/categorical. In Python, we code each letter as a numbers between 1 and 26 where 1 corresponds to a and 26 to z.

Suppose we let $y = 0$ corresponds to Male and $y = 1$ correspond to Female. In this case, according to Bayes rule, we get (see section 13.1.1):

$$p(y = c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = c)p(y = c)}{p(\boldsymbol{x}|y = 0)p(y = 0) + p(\boldsymbol{x}|y = 1)p(y = 1)}. \tag{1}$$

We will classify a name as a female name if $P(y = 1|x_1, x_2, x_3, x_4) > P(y = 0|x_1, x_2, x_3, x_4)$ and as a male name otherwise. We will split the data into a training and a test set and build our classifier using the training data, i.e.

$$\begin{aligned} P(x_1, x_2, x_3, x_4|y = 0) &= \frac{N_{\text{train}}^{\text{Female}}(x_1,x_2,x_3,x_4)}{N_{\text{train}}^{\text{Female}}} \\ P(x_1, x_2, x_3, x_4|y = 1) &= \frac{N_{\text{train}}^{\text{Male}}(x_1,x_2,x_3,x_4)}{N_{\text{train}}^{\text{Male}}}, \end{aligned}$$

where $N_{\text{train}}^{\text{Female}}(x_1, x_2, x_3, x_4)$ and $N_{\text{train}}^{\text{Male}}(x_1, x_2, x_3, x_4)$ denotes respectively the number of female and male names with first letter x_1, second letter x_2, second last letter x_3 and last letter x_4 in the training data. The prior terms $p(y)$ are estimated as described in section 13.1.1.

7.4.1 In order to classify names as male or female according to the above we need to count the number of times given letter combinations occurred in

the male and female names in the training data, i.e. how many times each of the letter combinations $(x_1, x_2, x_3, x_4) = (a, a, a, a), (x_1, x_2, x_3, x_4) = (a, a, a, b), \ldots, (x_1, x_2, x_3, x_4) = (z, z, z, z)$ occurred. How many different letter combinations do we have to evaluate?

7.4.2 How well do you think we can identify the probabilities $P(x_1, x_2, x_3, x_4|\text{Male})$ and $P(x_1, x_2, x_3, x_4|\text{Female})$ from the data at hand? Consider how likely it is that a given dataset will contain enough training samples to allow us to accurately estimate these probabilities.

Rather than finding the full joint distribution $P(\boldsymbol{X}|Y)$ we will use a Naïve Bayes classifier instead that assumes that each attribute (letter in a name) is independent, and thus we arrive at the expression:

$$p(y|x_1, x_2, \ldots, x_M) = \frac{p(x_1|y) \times \cdots \times p(x_M|y)p(y)}{\sum_{c=1}^{C} p(x_1|y=c) \times \cdots \times p(x_M|y=c)p(y=c)}.$$

From the training data we can obtain

$$\begin{aligned} P(X_i = x_t|y=0) &= \frac{N_{\text{train}}^{\text{Male}}(X_i = x_t)}{N_{\text{train}}^{\text{Male}}} \\ P(X_i = x_t|y=1) &= \frac{N_{\text{train}}^{\text{Female}}(X_i = x_t)}{N_{\text{train}}^{\text{Female}}} \end{aligned}$$

where $N^{\text{Male}}(X_i = x_t)$ is the number of times the letter x_t occurred in the i^{th} considered letter of the name. Using Naïve Bayes we only need to estimate the probability of observing each of the 26 letters for each of the four considered position in the spelling of the name separately. From more than one thousand male and female names we can quite accurately estimate these probabilities.

7.4.3 Inspect and run the script `ex7_4_3.py`. The script loads the male and female names, filters the text data and extracts the first two and last two letters of each name as feature.

7.4.4 Inspect and run the script `ex7_4_4.py`. The script is used to classify the names using a naïve Bayes classifier. Use a uniform prior, assuming male and female names are equally likely (i.e. $P(y = 0) = P(y = 1) = 0.5$). Compute the classification error using 10-fold crossvalidation.

Script details:

- *Type* `help(sklearn.naive_bayes.MultinomialNB)` *to learn how to use naive bayes framework in Python.*
- *To specify that the data is categorical we use the* `sklearn.preprocessing.OneHotEncoder`*. Otherwise* `MultinomialNB` *assumes that the input numbers are e.g. discrete counts of words or tokens. Without the encoding, the value 26 for a given element would signify 26 counts of a given tokens, whereas the encoding ensures that the value 26 is interpreted a level of a categorical variable corresponding to the letter "z".*
- *The multinomial distribution is used for describing discrete/categorical features, as we do with this name dataset. However, if we need to model continous data, we need to use a different distribution, such as a Gaussian, see* `GaussianNB` *or* `https://scikit-learn.org/stable/modules/naive_bayes.html` *for more information. An alternative to Gaussian Naïve Bayes is to discretize the input data (e.g. by binarizing based on median).*
- *As usual, use* `sklearn.cross_validation` *module to set up the crossvalidation partitions.*

Try classifying names based on only two of the letters in the name. You can do this by writing, e.g., `X=X[:,0:2]` to choose the first two letters. Are the first or last letters more useful for classifying names? Can you explain why?

7.4.5 Change the prior to `empirical` such that $P(y=1) = \frac{N_{\text{train}}^{\text{Female}}}{N_{\text{train}}^{\text{Female}}+N_{\text{train}}^{\text{Male}}}$, $P(y=0) = \frac{N_{\text{train}}^{\text{Male}}}{N_{\text{train}}^{\text{Female}}+N_{\text{train}}^{\text{Male}}} = 1 - P(\text{Female})$. Does this setting improve the classification and if so why?

7.5 Tasks for the report

Statistical evaluation will be important for the report, but we still need to introduce regularization and artificial neural networks to state the tasks. See report description for more information.

1 Homework problems for this week

Problems

Question 1. Fall 2015 question 16: Nine of the fifteen observations in Table 1 have chronic kidney disease (i.e., O_1–O_9 given in red) whereas six of the observations do not have chronic kidney disease (i.e., O_{10}–O_{15}) given in black). We would like to predict whether a subject has chronic kidney disease or not using the data in Table 1 and the attributes RBC, PC, DM, and CAD. We will apply a Naïve Bayes classifier that assumes independence between the four attributes. Given that a subject has these four attributes (i.e., $RBC = 1$, $PC = 1$, $DM = 1$, and $CAD = 1$) what is the probability that the person has chronic kidney disease, i.e., what is $P(CKD = 1|RBC = 1, PC = 1, DM = 1, CAD = 1)$ according to the Naïve Bayes classifier?

	RBC	PC	PCC	HTN	DM	CAD	PE
O_1	0	0	0	0	1	0	0
O_2	0	1	1	1	0	0	1
O_3	0	0	0	0	0	0	0
O_4	0	1	0	0	1	0	1
O_5	0	1	1	1	1	0	0
O_6	1	1	1	1	1	0	0
O_7	1	1	1	1	1	0	1
O_8	0	1	1	1	1	1	1
O_9	0	1	0	1	0	0	0
O_{10}	1	1	0	0	0	1	0
O_{11}	0	0	0	0	1	0	0
O_{12}	0	0	0	0	0	0	0
O_{13}	0	0	0	0	0	0	0
O_{14}	0	0	0	0	0	0	0
O_{15}	0	0	0	0	0	0	0

Table 1: For each observation there are $M = 7$ binary features and $N = 15$ observations $O_1, \ldots, O_{15}$ belonging to two categories (i.e., CKD=1 for $O_1, \ldots, O_9$ and CKD=0 for $O_{10}, \ldots, O_{15}$).

A 2.56 %

B 96.14 %

C 98.03 %

D 100 %

E Don't know.

Question 2. Fall 2011 question 2: Consider the data set described in Table 2. Each attribute in the data set is standardized, and we carry out a principal component analysis (PCA) on the standardized input data, x_1–x_6. The singular values obtained are: $\sigma_1 = 17.0$, $\sigma_2 = 15.2$, $\sigma_3 = 13.1$, $\sigma_4 = 13.0$, $\sigma_5 = 11.8$, $\sigma_6 = 11.3$. The first and second principal component directions are:

$$\boldsymbol{v}_1 = \begin{bmatrix} 0.5238 \\ 0.5237 \\ -0.3491 \\ 0.1981 \\ -0.3369 \\ 0.4204 \end{bmatrix}, \quad \boldsymbol{v}_2 = \begin{bmatrix} -0.2948 \\ 0.3452 \\ 0.3584 \\ 0.6808 \\ -0.3049 \\ -0.3302 \end{bmatrix}.$$

Which one of the following statements is *incorrect*?

No.	Attribute description	Abbrev.
x_1	Age of Mother in Whole Years	Age
x_2	Mothers Weight in Pounds	MW
x_3	Race (1 = Other, 0 = White)	Race
x_4	History of Hypertension (1 = Yes, 0 = No)	HT
x_5	Uterine Irritability (1 = Yes, 0 = No)	UI
x_6	Number of Physician Visits First Trimester	PV
y	Birth Weight in Kilo Grams	BW

Table 2: Attributes in a study on risk factors associated with giving birth to a low birth weight (less than 2.5 kg) baby [Hosmer and Lemeshow, Applied Logistic Regression, 1989]. The data we consider contains 189 observations, 6 input attributes x_1–x_6, and one output variable y.

A The first three principal component account for more than 90% of the variation in the data.

B Relatively heavy, old and white mothers that frequently goes to the physician and have a history of hypertension but do not have uterine irritability will have a positive projection onto the first principal component.

C Relatively young, heavy mothers that are not white and have a history of hypertension but infrequently goes to the physician and do not have a uterine irritability will have a positive projection onto the second principal component.

D Since the data is standardized we do not need to subtract the mean when performing the PCA but can directly carry out the singular value decomposition on the standardized data.

E Don't know.

Question 3. Fall 2014 question 23: Consider a two-dimensional data set consisting of $N = 7$ observations as shown in fig. 1. The dataset consist of two classes indicated by the black crosses (class 1) and red circles (class 2). In the figure, the decision boundary for four K-nearest neighbor classifier (KNN) is shown such that the lighter brown color indicates Class 2 (red circles) and the darker brown color indicates Class 1 (black crosses). Suppose K is restricted to the values $1, 3, 5, 7$, which of the following statements are true about values of k_1, k_2, k_3 and k_4?

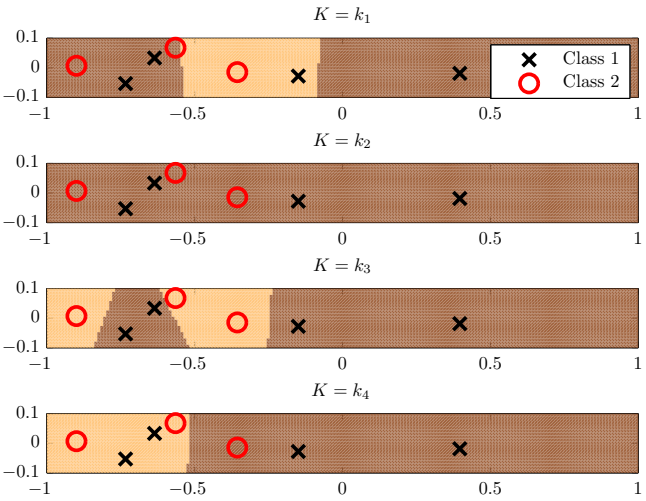

Figure 1: Decision boundaries for four KNN classifiers.

A $k_1 = 1, k_2 = 7, k_3 = 3, k_4 = 5$

B $k_1 = 1, k_2 = 5, k_3 = 7, k_4 = 3$

C $k_1 = 5, k_2 = 7, k_3 = 1, k_4 = 3$

D $k_1 = 3, k_2 = 7, k_3 = 1, k_4 = 5$

E Don't know.

References

Artificial Neural Networks and Bias/Variance with PYTHON

Objective: The objective of today's exercise is to understand (i) regularization in linear regression (regularized linear regression), (ii) Introduce artificial neural networks (ANNs) and get a feeling of how neural networks can be used for data modelling and the role of hidden units in neural networks (iii) understand how ANNs and multinomial regression (the generalization of logistic regression to multiple classes) can be applied to multi-class problems.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 8 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`⋆_train`	—	—	Training data.
	`⋆_test`	—	—	Test data.

8.1 Regularization

An approach to control for the complexity of the model is to regularize the parameters in the objective function. For instance for linear regression we can regularize the parameters $\boldsymbol{w}$ by the Frobenius norm, i.e.

$$C = \sum_n \|y_n - \boldsymbol{x}_n^\top \boldsymbol{w}\|_F^2 + \lambda \|\boldsymbol{w}\|_F^2 \quad (1)$$

$$= \sum_n (y_n - \boldsymbol{x}_n^\top \boldsymbol{w})^2 + \lambda \boldsymbol{w}^\top \boldsymbol{w} \quad (2)$$

Solving the equation $\frac{\partial C}{\partial \boldsymbol{w}} = 0$ we obtain $\boldsymbol{w} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$, see also the lecture slides for today.

When regularizing a model as above we apply the same regularization strength to all features, and so it is important to standardize the features. Note that if we include a bias term (offset/intercept) we do not regularize it.

In the two following scripts, we will see how to regularize a model for predicting a continous target variable (using *linear* regression), and a model for classification (predicting a binary class label with *logistic* regression) using the above regularization method (also called ridge regression, or L2-regularization).

8.1.1 *Linear* regression model: inspect and run the script `ex8_1_1.py`. The script investigates the body data with the regularized least squares regression method. In the inner loop of the script 10-fold cross-validation is used to select for the optimal value of regularization λ. In the outer loop 5-fold cross-validation is used to evaluate the performance of the optimal selected value for λ. What is plotted on each of the generated plots? How is the optimal value of the regularization (λ) chosen? What do you think may be the advantages and disadvantages of regularized linear regression compared to the sequential feature selection? Try to inspect the weights obtained when using a very high regularization strength. What has happend to the value of the coefficients and what does the bias now represent? (try to compare the mean of the target variable in the training set to the parameters that have been fitted)

8.1.2 *Logistic* regression model: inspect and run the script `ex8_1_2.py`. The script loads the wine data (`Data/wine2`). The script standardizes the data based on the training set feat means and standard deviations. The data is then used to train a model that predicts whether a data point is a red or a white wine (see Exercise 5.2.6, for the unregularized counterpart). A suitable regularization strength is determined using hold-out cross-validation. The Wine dataset is relatively large, so we train on only a small part of the dataset (10 percent) to illustrate the effect of regularization on small datasets. You can try to change the percent used for training (try e.g. 50 % and 99 %). Run the script multiple times to see how much the subset affects the performance when reducing the training set size. How is the magnitude of the fitted parameters affected by the regularization strength?

8.2 Artificial neural networks

In this part of the exercise we will use neural networks to classify data. We will consider networks with an input layer, one layer of hidden units and an output

layer. If there is one unit in the hidden and output layer and the transfer function of each layer is linear, i.e. $g^{(hidden)}(t) = t$ and $g^{(output)}(t) = t$ it can be shown that the output y^{est} of the neural network is equivalent to linear regression, i.e. $y^{est} = w_0 + \sum_k w_k x_k$, while if the transfer function of the hidden layer ($g^{(hidden)}$) is given by the sigmoid or tanh while the transfer function of the output layer ($g^{(output)}$) is linear the network is closely related to logistic regression. Artificial neural networks (ANN), or just neural networks, with more than one hidden units can be much more flexible than linear and logistic regression, however, ANN are not guaranteed to find the optimal solutions. Therefore, ANNs are normally trained multiple times with different random initialization and the trained network with best training error selected.

We can use ANNs for both classification and regression. When we train an ANN to do regression, we do not a apply a tranfer function to the output node(s), and we train the network using a mean-square-error loss. However, when we train classification problems, we use a transfer function for the output nodes and a different loss. For instance, when doing binary classification, we can use a sigmoidal tranfer function in the last layer, and we would use a binary cross entropy loss. When doing multi-class classification, we would use e.g. a softmax transfer function and apply a cross entropy loss.

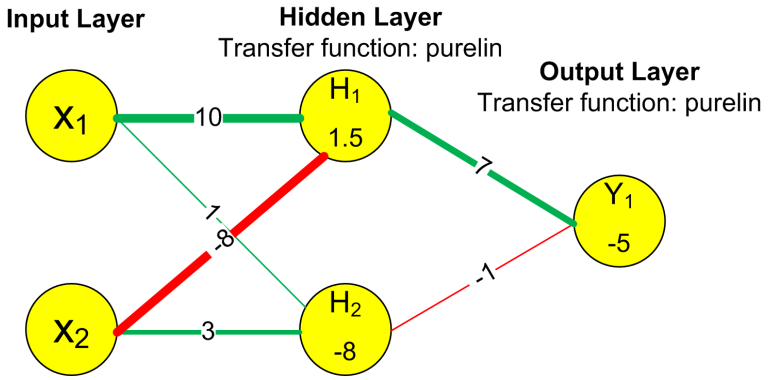

Figure 1: A learned network with two inputs, two hidden units and one outputs. Positive weights are indicated in green, negative in red. The bias, i.e. the constant for each unit is given by the numeric value inside the hidden and output unit. The transfer functions of the hidden layer and output layer are here linear.

In the following exercises we shall use an open source library offering neural networks functionality in Python called PyTorch. You can find the source code, installation hints, documentation and examples at: `https://pytorch.org/tutorials/`

We can install PyTorch with no GPU-support (no CUDA) by running the following command(s) or following the instructions on `https://pytorch.org/get-started/locally/`:

```
(Windows, using the Anaconda Prompt:)
>>>conda install pytorch-cpu -c pytorch
>>>pip3 install torchvision
(Mac:)
>>>conda install pytorch torchvision -c pytorch
(Linux:)
>>>conda install pytorch-cpu torchvision-cpu -c pytorch
```

8.2.1 In figure 1 a network with two input units, two hidden units and one output unit has been trained based on linear functions as transfer function, i.e. the output of the first hidden unit is $h_1 = g^{(hidden)}(1.5 + 10x_1 - 8x_2)$ and the output of the second hidden unit is $h_2 = g^{(hidden)}(-8 + 1x_1 + 3x_2)$ with $(g^{(hidden)}(t) = t)$. The final output of the neural network is given by $y^{est} = g^{(Output)}(-5 + 7h_1 - 1h_2)$ with $(g^{(output)}(t) = t)$. What is the output of the network if $x_1 = 1$ and $x_2 = 0$ and what is the output if $x_1 = 0.5$ and $x_2 = 0.5$?
Change the transfer function of the hidden layer to be the rectified linear function, i.e. $g^{(hidden)}(t) = \left\{ \begin{array}{l} t \text{ if } t > 0 \\ 0 \text{ otherwise} \end{array} \right\}$. This function is also called a rectifier, and when a node has this tranfer function we call it a rectified linear unit (ReLU). What is the output now of the network if $x_1 = 1$ and $x_2 = 0$ and what is the output if $x_1 = 0.5$ and $x_2 = 0.5$?

8.2.2 We will use a simple data set to demonstrate the neural network. Use the `loadmat()` function to load `Data/xor.mat` Matlab datafile into Python. After examining the loaded dataset, examine and run the script `ex8_2_2.py`, which fits a neural network and uses k-fold crossvalidation to estimate the classification error. Observe the decision boundaries and learning curves (loss as a function of training steps).
Script details:

- *We can import PyTorch using* `import torch`
- *To define a neural network we use* `torch.nn.Sequential`. *Each argument to this function is a part of a sequentially applied network. We can get a layer that maps M features to H hidden units by using* `torch.nn.Linear(M, H)`. *We can*

then apply a transfer function to the output of that layer by having the next argument be `torch.nn.Tanh()` *(applying a hyperbolic tangent function). To train a binary classifier, we need a suitable final transfer function, which is for instance* `torch.nn.Sigmoid()`.

- *By using a lambda function we can get a new copy of the network each time we call* `model()`*, which we need since we're doing K-fold cross-validation and multiple replicates within each fold.*
- *Neural networks are trained by iteratively reducing a cost function/loss. We can define a binary cross entropy loss suitable for binary classification problems by using* `torch.nn.BCELoss()`.
- *PyTorch operates on objects called tensors, which are simply multi-dimensional matrices, and the way they are defined within PyTorch enables e.g. processing on GPUs, which speeds up training of large networks. We wont be using GPUs for training, but we need to define the inputs to the neural network as tensors, which we do using* `torch.tensor()`*. We can define the inputs to be of type floats by specyfing the datatype:* `dtype=torch.float`.
- *Training the specified network is done by calling* `train_neural_net`*. Type* `help(train_neural_` *You should have a look at how the training is done by writing* `edit train_neural_net`*— this will take you to where the function is defined in the 02450 toolbox.*
- *The outputs of the network are the activation of the final unit passed through the sigmoidal tranfer function, resulting in a number between zero and one. To obtain class-estimates we can threshold the value at* 0.5.
- *Since PyTorch operates on PyTorch-tensors, we need to convert back to "regular" arrays before using e.g. numpy-operations, we can get a numpy version of a tensor* `a` *by* `b = a.data.numpy()`.
- *We can visualize the decision boundary of a classifier by using* `visualize_decision_boundary()` *The function needs a lambda-function, which is function that takes as input a 2-dimensional dataset and outputs predictions.*
- *We can visualize a diagram of a trained network similar to Figure 1 by calling* `draw_neural_net`*. This functions needs to be supplied with the weights, biases and transfer functions supplied, which we can get by extracting the values from the* `torch.nn.Sequential`*-object that is returned from* `train_neural_net`*. These diagrams quickly become of little use with increasing input features and increasing layer sizes (number of hidden units).*
- *We control how many replicates/restarts we do when training a network through* `n_replicates`*, and we can set the maximum number of training steps that is done during training by* `max_iter`.

Note that you will get quite different results every time you run the script. Can you explain why? Since there is no certainty a network will converge, you might need to rerun the script.
Usually, you will need to train multiple randomly initiated networks and

use crossvalidation to select the best ones, as well as to optimize hyper-parameters (i.e. learning rate, number of hidden nodes). This process can be computationally expensive, hence in practice it is often executed in parallel on multi-processor machines and computational grids.

8.2.3 Try increase the number of hidden units to `n_hidden_units = 2`. Does the classification performance improve? Can you explain why?

8.2.4 Try change the number of hidden units in the network to `n_hidden_units = 10`. What happens to the decision boundaries of the learned neural networks? What are the benefits and drawbacks of including many hidden units in the network?

8.2.5 Load the wine data (load `Data/wine2` for data with outliers removed) and try and classify wine as red or white using the neural network classifier. Use the script `ex8_2_5.py` as a reference. What is the performance when `n_hidden_units=2`? Can you interpret how wine is classified as red and white wine? Try run the model with `n_hidden_units=1`, can you now interpret how the network predicts the type of wine?

8.2.6 Neural networks can also be used for regression (i.e. continuous output). Examine and run the script `ex8_2_6.py`. Try to predict the alcohol content of wine for `n_hidden_units=2`. How well is the network able to predict the alcohol content of wine? Try and see if you can interpret how the trained networks determines the alcohol content of the wine? Try set the `n_hidden_units=5`, how well does the network predict the alcohol content of the wine? Script details:

- *When doing regression, we do not apply a transfer-function to the final layer. If we have a one-dimensional regression target, the last argument to* `torch.nn.Sequential` *when defining the model should be a* `torch.nn.Linear(n_hidden_units, n_out)`*, where* `n_out` *is equal to one.*
- *Similarly, when doing regression we do not train the network based on a cross entropy loss, but on a mean-square-error-loss (MSE loss). We can define such a loss function by using* `torch.nn.MSELoss()`*.*

8.3 Multiclass problems using ANNs and multinomial regression

We have previously used NaïveBayes, K-Nearest Neighbor and Decision trees for the classification of data with multiple classes. Logistic regression and artificial neural networks (ANN) can however also be extended to multiple classes by use

of the softmax function given by $f_c(\boldsymbol{o}) = \frac{\exp(o_c)}{\sum_{c'} \exp(o_{c'})}$. Thus, o_c corresponds to the predictions made for the c^{th} class and all predictions are tied together through the softmax function. When training the model such that the output of the function $f_c(\boldsymbol{o})$ can be interpreted as the probability that the observation belongs to the c^{th} class. The softmax link function is for two class problems equivalent to the logit link function and this particular extension of logistic regression to multi-class is called multinomial regression.

8.3.1 Examine and run the scripts `ex8_3_1.py`. The scripts loads a synthetic multi-class classification data set and fits a neural network to training data using the multi-class ANN (`ex8_3_1.py`). Next, it computes the outputs of the trained classifier on the test data and classifies it according to the class having the highest probability. The script then computes the error rate and plots the decision boundaries of the trained model. For evaluation, you can use one of the synthetic multiclass data sets as before, (`Data/synth1` ... `Data/synth4`).

Script details:

- *When we train a network for C-classes classification problem, we set the last layer of the network to have C output neurons, that we also call logits.*
- *The softmax-transfer function can be used by* `torch.nn.Softmax`. *When defining this, we need to specify which dimension to normalize/link over. If its the second dimension (as is the case for the script), we specify* `dim=1`*—if we had used the first dimesion, the function would have normalized not over classes but over the number of observations.*
- *The appropriate loss function for a multi-class problem is a cross-entropy loss,* `torch.nn.CrossEntropyLoss()`.

8.3.2 **Optional:** Use `ex8_3_2.py` to fit a multinomial regression model to the dataset. Multinomial regression models are based on tieing the outputs by the softmax function. Especially consider `Data/synth3`. Notice the multinomial regression model is able to solve this problem; look at the weights and explain *how* this is accomplished (n.b. notice the vertical dimension does not matter).

8.3.3 **Optional:** Just like we can regularize a linear and logistic regression, we can regularize multinomial regression. Use `ex8_3_3.py` to fit a regularized multinomial regression model. Investigate how a very high regularization strength affects the class label estimations (look at which class label is the mode in the training set and which label is estimated).

8.4 Tasks for the report

You are now in a position to address the remaining tasks for the report.
Regression, part a

1. Introduce a regularization parameter λ as discussed in chapter 14 of the lecture notes, and estimate the generalization error for different values of λ. Specifically, choose a reasonable range of values of λ (ideally one where the generalization error first drop and then increases), and for each value use $K = 10$ fold cross-validation (algorithm 5) to estimate the generalization error.

   Include a figure of the estimated generalization error as a function of λ in the report and briefly discuss the result.

2. Explain how a new data observation is predicted according to the linear model with the lowest generalization error as estimated in the previous question. I.e., what are the effects of the selected attributes in terms of determining the predicted class. Does the result make sense?

Regression, part b: In this section, we will compare three models: the regularized linear regression model from the previous section, an artificial neural network (ANN) and a baseline. We are interested in two questions: Is one model better than the other? Is either model better than a trivial baseline?. We will attempt to answer these questions with two-level cross-validation.

1. Implement two-level cross-validation (see algorithm 6 of the lecture notes). We will use 2-level cross-validation to compare the models with $K_1 = K_2 = 10$ folds[1]. As a baseline model, we will apply a linear regression model with no features, i.e. it computes the mean of y on the training data, and use this value to predict y on the test data.

   Make sure you can fit an ANN model to the data. As complexity-controlling parameter for the ANN, we will use the number of hidden units[2] h. Based on a few test-runs, select a reasonable range of values for h (which should include $h = 1$), and describe the range of values you will use for h and λ.

[1] If this is too time-consuming, use $K_1 = K_2 = 5$

[2] Note there are many things we could potentially tweak or select, such as regularization. If you wish to select another parameter to tweak feel free to do so.

Outer fold	ANN		Linear regression		baseline
i	h_i^*	E_i^{test}	λ_i^*	E_i^{test}	E_i^{test}
1	3	10.8	0.01	12.8	15.3
2	4	10.1	0.01	12.4	15.1
⋮	⋮	⋮	⋮	⋮	⋮
10	3	10.9	0.05	12.1	15.9

Table 1: Two-level cross-validation table used to compare the three models

2. Produce a table akin to Table 1 using two-level cross-validation (algorithm 6 in the lecture notes). The table shows, for each of the $K_1 = 10$ folds i, the optimal value of the number of hidden units and regularization strength (h_i^* and λ_i^* respectively) as found after each inner loop, as well as the estimated generalization errors E_i^{test} by evaluating on $\mathcal{D}_i^{\text{test}}$. It also includes the baseline test error, also evaluated on $\mathcal{D}_i^{\text{test}}$. Importantly, you must re-use the train/test splits $\mathcal{D}_i^{\text{par}}, \mathcal{D}_i^{\text{test}}$ for all three methods to allow statistical comparison (see next section).

   Note the error measure we use is the squared loss *per observation*, i.e. we divide by the number of observation in the test dataset:

$$E = \frac{1}{N^{\text{test}}} \sum_{i=1}^{N^{\text{test}}} (y_i - \hat{y}_i)^2$$

   Include a table similar to Table 1 in your report and briefly discuss what it tells you at a glance. Do you find the same value of λ^* as in the previous section?

3. Statistically evaluate if there is a significant performance difference between the fitted ANN, linear regression model and baseline using the methods described in chapter 11. These comparisons will be made pairwise (ANN vs. linear regression; ANN vs. baseline; linear regression vs. baseline). We will allow some freedom in what test to choose. Therefore, choose either:

   setup I (section 11.3): Use the paired t-test described in Box 11.3.4

   setup II (section 11.4): Use the method described in Box 11.4.1)

   Include p-values and confidence intervals for the three pairwise tests in your report and conclude on the results: Is one model better than the other? Are

Outer fold	Method 2		Logistic regression		baseline
i	x_i^*	E_i^{test}	λ_i^*	E_i^{test}	E_i^{test}
1	3	10.8	0.01	12.8	15.3
2	4	10.1	0.01	12.4	15.1
⋮	⋮	⋮	⋮	⋮	⋮
10	3	10.9	0.05	12.1	15.9

Table 2: Two-level cross-validation table used to compare the three models in the classification problem.

the two models better than the baseline? Are some of the models identical? What recommendations would you make based on what you've learned?

Classification: In this part of the report you are to solve a relevant classification problem for your data and statistically evaluate your result. The tasks will closely mirror what you just did in the last section. The three methods we will compare is a baseline, logistic regression, and **one** of the other four methods from below (referred to as *method 2*).

Logistic regression for classification. Once more, we can use a regularization parameter $\lambda \geq 0$ to control complexity

ANN Artificial neural networks for classification. Same complexity-controlling parameter as in the previous exercise

CT Classification trees. Same complexity-controlling parameter as for regression trees

KNN k-nearest neighbor classification, complexity controlling parameter $k = 1, 2 \ldots$

NB Naïve Bayes. As complexity-controlling parameter, we suggest the term $b \geq 0$ from section 11.2.1 of the lecture notes to estimate[3] $p(x = 1) = \frac{n^+ + b}{n^+ + n^- + 2b}$

[3] In Python, use the `alpha` parameter in `sklearn.naive_bayes` and in R, use the `laplacian` parameter to `naiveBayes`. We do *not* recommend NB for Matlab users, as the implementation is somewhat lacking.

1. We will compare logistic regression[4], *method 2* and a baseline. For logistic regression, we will once more use λ as a complexity-controlling parameter, and for *method 2* a relevant complexity controlling parameter and range of values. We recommend this choice is made based on a trial run, which you do not need to report. Describe which parameter you have chosen and the possible values of the parameters you will examine.

   The baseline will be a model which compute the largest class on the training data, and predict everything in the test-data as belonging to that class (corresponding to the optimal prediction by a logistic regression model with a bias term and no features).

2. Again use two-level cross-validation to create a table similar to Table 2, but now comparing the logistic regression, *method 2*, and baseline. The table should once more include the selected parameters, and as an error measure we will use the error rate:

$$E = \frac{\{\text{Number of misclassified observations}\}}{N^{\text{test}}}$$

   Once more, make sure to re-use the outer validation splits to admit statistical evaluation. Briefly discuss the result.

3. Perform a statistical evaluation of your three models similar to the previous section. That is, compare the three models pairwise. We will once more allow some freedom in what test to choose. Therefore, choose either:

   setup I (section 11.3): Use McNemera's test described in Box 11.3.2)

   setup II (section 11.4): Use the method described in Box 11.4.1)

   Include p-values and confidence intervals for the three pairwise tests in your report and conclude on the results: Is one model better than the other? Are the two models better than the baseline? Are some of the models identical? What recommendations would you make based on what you've learned?

4. Train a logistic regression model using a suitable value of λ (see previous exercise). Explain how the logistic regression model make a prediction. Are the same features deemed relevant as for the regression part of the report?

Discussion:

[4] in case of a multi-class problem, substitute logistic regression for multinomial regression

1. Include a discussion of what you have learned in the regression and classification part of the report.

2. If your data has been analyzed previously (which will be the case in nearly all instances), find a study which uses it for classification, regression or both. Discuss how your results relate to those obtained in the study. If your dataset has not been published before, or the articles are irrelevant/unobtainable, this question may be omitted but make sure you justify this is the case.

1 Homework problems for this week

Problems

Question 1. Fall 2013 question 23: Which one of the following statements pertaining to regression is *correct*?

A In regularized least squares regression the aim is to introduce more variance by reducing substantially the model's bias.

B In least squares regularized regression the regularization strength λ is chosen to be the value of λ that minimizes the term $\lambda \boldsymbol{w}^\top \boldsymbol{w}$.

C An artificial neural network with linear transfer functions ($q(t) = t$) can be written in terms of a linear regression model.

D For regression problems backward or forward selection can be used to define which part of the output that is relevant for modeling.

E Don't know.

Question 2. Fall 2014 question 5: Consider a feedforward neural network shown in fig. 2. The network has no bias weights.

Suppose the weights of the neural network are trained to be $w_{31} = 0.5$, $w_{41} = 0.4$, $w_{32} = -0.4$, $w_{42} = 0$, $w_{53} = -0.4$, $w_{54} = 0.1$ and the activation function of all five $n_1, \ldots, n_5$ nodes is the thresholded linear function

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Suppose the network is called evaluated on input $x_1 = 1, x_2 = 2$, what is the output?

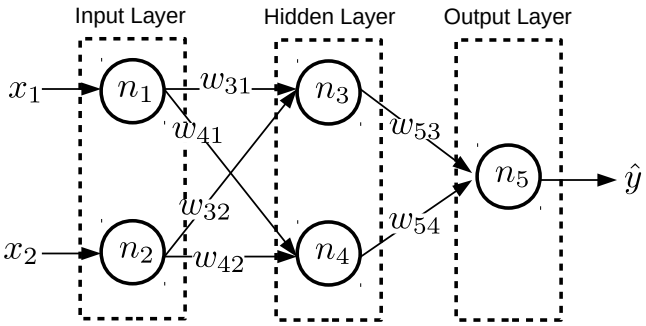

Figure 2: Simple neural network of 6 weights

A $\hat{y} = 0.04$

B $\hat{y} = 0.0$

C $\hat{y} = 1.0$

D $\hat{y} = 0.16$

E Don't know.

Question 3. Fall 2013 question 12: Consider the classification problem given in Figure 3. The problem is solved using a 1-nearest neighbor classifier, a decision tree, an artificial neural network with four hidden units and a logistic regression model. All the classifiers are only using the attributes x_1 and x_2. The decision boundaries are indicated in gray and white. We would like to know which classifier each of the four decision boundaries in Figure 3 correspond to. Which one of the following statements is *correct*?

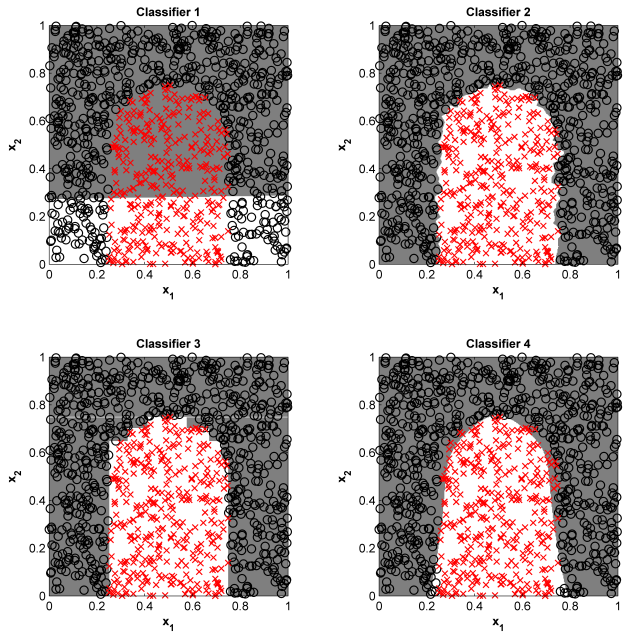

Figure 3: The decision boundaries given in white and gray of four different classifiers used to separate red crosses from black circles.

A Classifier 1 is the decision tree, Classifier 2 is the artificial neural network, Classifier 3 is the logistic regression model, and classifier 4 is the 1-nearest neighbor classifier.

B Classifier 1 is the artificial neural network , Classifier 2 is the 1-nearest neighbor, Classifier 3 is the decision tree, and classifier 4 is the logistic regression model.

C classifier 1 is the logistic regression model, Classifier 2 is the decision tree, Classifier 3 is the 1-nearest neighbor classifier, and classifier 4 is the artificial neural network classifier.

D Classifier 1 is the logistic regression model, Classifier 2 is the 1-nearest neighbor, Classifier 3 is the decision tree, and classifier 4 is the artificial neural network.

E Don't know.

References

AUC and ensemble methods with PYTHON

Objective: The objective of today's exercise is to understand (i) how the ensemble methods bagging and boosting is used to improve the performance of classifiers, (ii) how the receiver operating characteristic (ROC) is used to evaluate the performance of two-class classification problems, and (iii) how artificial neural network and logistic regression can be generalized to multi-class classification.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 9 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`⋆_train`	—	—	Training data.
	`⋆_test`	—	—	Test data.

9.1 Ensemble methods

In this part of the exercise we will consider ensemble methods to improve the classification performance. In particular we will consider bagging and boosting methods.

In bagging we randomly sample with replacement the same number of samples as the size of the training data. This is also denoted bootstrapping and it can be shown that on average each bootstrap sample will contain approximately 63% of the samples in the data.

In boosting we adaptively change the distribution that we sample the training examples from so that the classifiers will focus on examples that are hard to classify. A particularly well known boosting method is *"AdaBoost"* which is described in section 15.4 of the lecture notes *Introduction to Machine Learning and Data Mining*.

In this part of the exercise we will work with a synthetic data set which has two classes and two attributes, x_1 and x_2. This data set cannot be classified correctly with any linear classifier, such as logistic regression or an artificial neural network with one hidden unit, as long as only x_1 and x_2 are used as features.

9.1.1 Load the artificial dataset (`Data/synth5` file) with `loadmat` function. Inspect the data by making a scatterplot. Why can this data set not be classified correctly using logistic regression?

Inspect and run the script `ex9_2_1.py`. The script fits an ensemble of logistic regression models to the data, using the bootstrap aggregation (bagging) algorithm. Use $L = 100$ bootstrap samples. This requires creating 100 bootstrapped training sets, fit a logistic regression model to each, and combine the results of their outputs to make the final classification. Explain how the error rate is computed (on the training set) and how the decision boundary is plotted.

Script details:

- *To generate the bootstrap sample, you need to draw N data objects with replacement from the data set.*
- *You can use the function* `bootstrap()` *function from the toolbox to generate random numbers from a discrete distribution. You can write something like* `X_bs, y_bs = bootstrap(X, y, N, weights)` *to make a bootstrap sample.*
- *To make the final classification, take a majority vote amongst the classifiers in the ensemble. You can use simple class* `BinClassifierEnsemble` *from the toolbox.*
- *To plot the decision boundary, you can use the function* `dbplot()` *or* `dbprobplot()` *from the toolbox. It requires as an input an object that implements* `predict(X)` *and* `predict_proba(X)` *methods. You can call it e.g. like this:* `dbprobplot(fitted_classifier, X, y, 'auto', resolution=200)`

Bagging is known to be most effective for non-linear classifiers. Show that bagging only leads to a limited performance improvement for the logistic regression classifier.

Try also the data set in `Data/synth6` which is the] one used as an example in the lecture.

9.1.2 In the script `ex9_2_2.py`, the script `ex9_2_1.py` has been modified so that boosting is used instead of bagging. The script fits an ensemble of logistic regression models to the data, using the AdaBoost algorithm. Notice the script uses $L = 100$ rounds of boosting. This requires creating a

randomly chosen training set, fit a logistic regression model to it, evaluate its performance and update the weights accordingly, and compute a classifier importance. This process is repeated $L = 100$ times, and ultimately the trained ensemble of classifiers is combined to make a final classification. Compute the error rate (on the training set) and make a plot of the decision boundary.

Script details:

- *Read the hints to the previous exercise.*
- *Note that you will need two sets of weights in the algorithm. One is a weight for each of the N data objects, that is adapted when the boosting algorithm proceeds (we could call these* `weights`*). The other is the importance weights for the L trained classifiers (we could call these* `alpha`*).*
- *You can use the function* `bootstrap()` *to generate random numbers from a discrete distribution, as before. You will need to update the 'weights' parameter in according to AdaBoost algorithm, in every iteration.*
- *To make the final classification, take a weighted majority vote amongst the classifiers in the ensemble (weighted by* `alpha`*). Note that* `alpha` *needs to be normalized so that it sums to one. ii*

Show that, if you use enough rounds of boosting, the data set can be perfectly classified.

Try also the data set in `Data/synth6` which is the one used as an example in the lecture.

Let us return to the bagging algorithm, which we found to be of little use for logistic regression. Now, we will try the algorithm on a non-linear classifier: the decision tree. Bagging applied to decision trees is often called "random forests". The Python's package `sklearn.ensemble` has implemented bagging for random trees, see the class `RandomForestClassifier()`.

9.1.3 The data set `Data/synth5` we have used so far can trivially be fitted using a decision tree. Can you explain why? We will consider a different data set, which you can load from the file `Data/synth7`. Inspect and run the script `ex9_2_3.py`. Explore the data set and explain in what sense this is more challenging for a decision tree.

Notice the script fits a random forest (an ensemble of bagged decision trees) to the data using $L = 100$ rounds of bagging. Compute the error rate (on the training set) and make a plot of the decision boundary.

Script details:

- *Type* `help(sklearn.ensemble)` *to learn about the functions for fitting random forests with bagging.*

For comparison, you can try also to fit a regular decision tree (without bagging or pruning). Does bagging appear to improve on the classification, and if so, in what sense? Observe how the classification rate and decision boundaries decrease when you reduce number of bootstrap iterations.

Try the script on the other synthetic multi-class data sets you have studied before, (`Data/synth5` ... `Data/synth7`).

9.2 The receiver operating characteristic (ROC)

The receiver operating characteristic (ROC) is commonly used to evaluate and compare the performance of two-class classifiers, where one class is denoted "positive" and the other is denoted "negative." The ROC is a graphical approach for displaying the tradeoff between the true positive rate (y-axis) and the false positive rate (x-axis). For classifiers such as logistic regression and artificial neural networks that estimate the class labels by thresholding a continuous output variable, an ROC curve can be plotted by varying the threshold value.

9.2.1 Inspect and run the script `ex9_1_1.py`. The script loads the wine data (`Data/wine2`) into Python with the `loadmat` function. Notice how the data is divided into 50% for training and 50% for test using stratification such that training and test have roughly equal class proportions. Fit a logistic regression model to the training set to classifiy the wines as red or white. Consider the red wines as "positive" and white wines as "negative." Notice how the script makes a plot of the ROC curve showing that the AUC is around 0.99.

Script details:

- *You have fitted a logistic regression model to the wine data before in exercise 5.2.6.*
- *As before, use cross-validation. Use* `StratifiedKFold` *method to ensure that training and test sets have roughly equal class proportions (stratification).*
- *There is a function in the course toolbox called* `rocplot()` *that can make the ROC plot and compute the AUC score for you. Import it and type* `help(rocplot)` *to learn how to use it. Notice that the rocplot code assumes the score values are all distinct.*

9.2.2 Consult the script `ex9_1_2.py`. The experiment in exercise 9.1.1 is repeated, but this time it is examined how well the type of wine can be classified using only the "Alcohol" attribute. Explain how it can be seen that the alcohol contents of the wine is not very useful for classifying wine as red or white.

Discussion:

- ◇ You have showed that using only the single attribute "Alcohol" to classify the wine as red or white performs worse than using all attributes. When using logistic regression, is it always best to use as many attributes as possible for the classification?

9.3 Tasks for the report

Continue working on the tasks for the reports as described in the previous exercise note.

1 Homework problems for this week

Problems

Question 1. Fall 2016 question 18: Considering the data in Table 1, we will use x_1 to classify whether a subject has inflammation of urinary bladder ($y = 1$) or not ($y = 0$). We will quantify how useful x_1 is for this purpose by calculating the area under curve (AUC) of the receiver operator characteristic (ROC). Which one of the ROC curves given in Figure 1 corresponds to using the feature x_1 to determine if a subject has inflammation of urinary bladder?

	x_1	x_2	x_3	x_4	x_5	y
P1	1	1	1	1	0	1
P2	0	0	0	0	0	0
P3	1	1	0	1	0	0
P4	0	1	1	0	1	0
P5	1	1	1	1	1	1
P6	0	0	0	0	0	0
P7	1	1	0	1	0	0
P8	0	1	1	0	1	0
P9	1	1	1	1	0	1
P10	0	1	1	0	1	0
P11	0	0	0	0	0	0
P12	1	1	0	1	0	0
P13	0	1	1	0	1	0
P14	0	1	1	0	1	0

Table 1: Provided in the above table are the last 14 observations of the acute inflammation data.

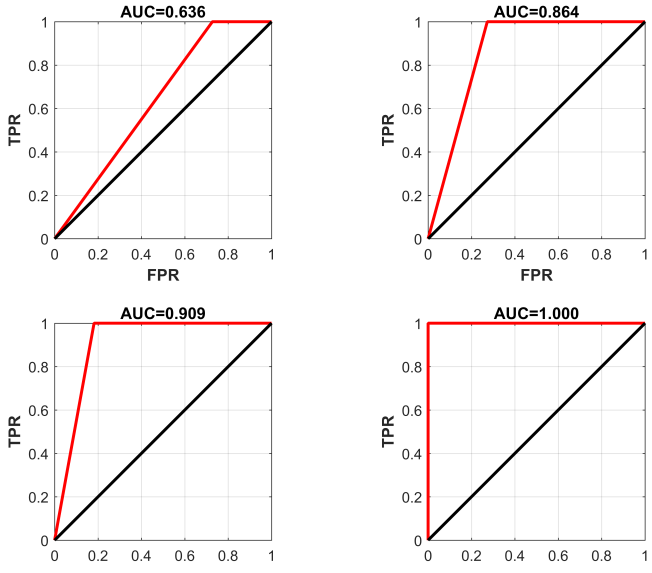

Figure 1: Four different receiver operating characteristic (ROC) curves and their corresponding area under the curve (AUC).

A The curve with AUC=0.636.

B The curve with AUC=0.864.

C The curve with AUC=0.909.

D The curve with AUC=1.000.

E Don't know.

Question 2. Fall 2018 question 13:

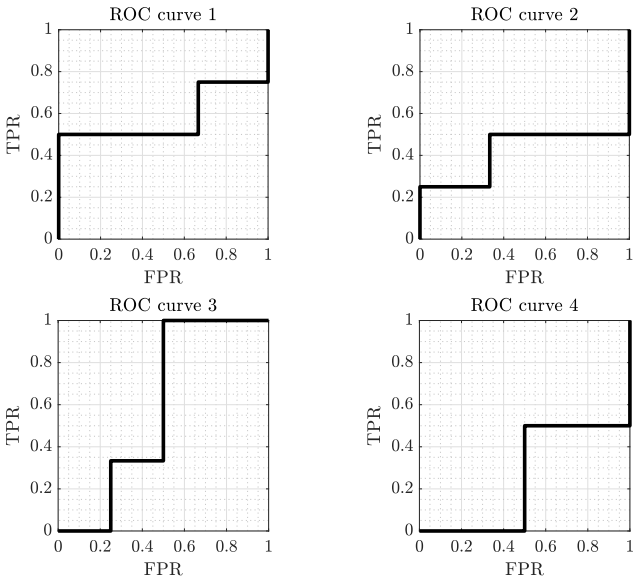

Figure 2: Proposed ROC curves for the logistic regression classifier in fig. 3.

To evaluate the classifier fig. 3, we will use the *area under curve* (AUC) of the *reciever operator characteristic* (ROC) curve as computed on the 7 observations in fig. 3. In fig. 2 is given four proposed ROC curves, which one of the curves corresponds to the classifier?

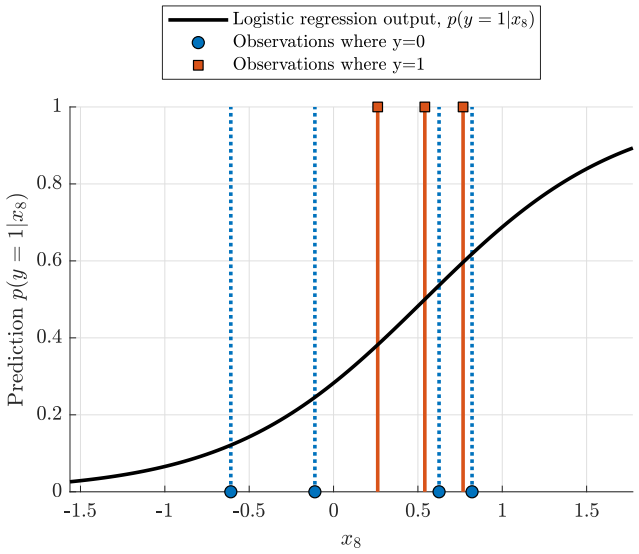

Figure 3: Output of a logistic regression classifier trained on 7 observations from the dataset.

A ROC curve 1

B ROC curve 2

C ROC curve 3

D ROC curve 4

E Don't know.

Question 3. Fall 2014 question 26: Suppose Jane wishes to apply a decision tree classifier to a binary classification problem of only $N = 4$ observations. Training and applying the decicion tree to the full dataset $\boldsymbol{X}$ and $y_1, \ldots, y_4$ gives predictions $\hat{y}_1, \ldots, \hat{y}_4$ shown in table 2.

y	$\hat{y}$
1	1
1	0
0	0
0	0

Table 2: True values y_j and predictions $\hat{y}_j$ for a decision tree classifier trained on the full data set with observed values $y_1, \ldots, y_4$.

To improve performance Jane decides to apply AdaBoost, however Jane implements AdaBoost such that instead of sampling the N elements of the training sets D_i *with* replacement, Jane samples the training sets *without* replacement, i.e. the training set D_i is simply the full dataset. Suppose Jane applies AdaBoost for $k = 1$ round of boosting, what is the resulting (approximate) value for the weights w?

A $w = \begin{bmatrix} 0.123 & 0.630 & 0.123 & 0.123 \end{bmatrix}$

B $w = \begin{bmatrix} 0.167 & 0.5 & 0.167 & 0.167 \end{bmatrix}$

C $w = \begin{bmatrix} 0.081 & 0.756 & 0.081 & 0.081 \end{bmatrix}$

D $w = \begin{bmatrix} 0.077 & 0.769 & 0.077 & 0.077 \end{bmatrix}$

E Don't know.

References

K-means and hierarchical clustering with PYTHON

Objective: The objective of today's exercise is to understand how the unsupervised learning methods k-means clustering and hierarchical clustering work. Upon completing the exercise you should also understand how the choice of number of clusters, distance metrics and linkage functions can impact the solutions obtained and further be able to interpret dendrograms and measures of cluster validity.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 10 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`*_train`	—	—	Training data.
	`*_test`	—	—	Test data.

In previous exercises we considered supervised learning, i.e., we were given both input data $\boldsymbol{X}$ and output values $\boldsymbol{y}$. In classification, the outputs were discrete variables, and in regression, the outputs were continuous.

We now move on to unsupervised learning where we are only provided input data $\boldsymbol{X}$. The aim is here to find common patterns in the data such as groups of observations that are similar in some sense. In this exercise we will consider two clustering approaches for unsupervised learning: k-means clustering and hierarchical clustering.

10.1 k-means clustering

In this part of the exercise we will investigate k-means clustering. In k-means each of the data points are assigned to the cluster in closest proximity according to some measure of distance between cluster centers and data points. When the distance is given by the squared Euclidean distance the centers are also denoted centroids. Once

the data points have been assigned, each cluster center is updated to be placed at the center of the data points that are assigned to the cluster. This continues iteratively, usually until the assignment of data points to centers no longer change or until a maximal number of iterations is reached.

10.1.1 You can load the `Data/synth1.mat` data set file into Python using the `loadmat` function. The script `ex10_1_1.py` clusters the data into $K = 4$ clusters using the k-means algorithm. Notice how the script makes a scatter plot of the data and the clustering using the `clusterplot` function in the toolbox.

Script details:

- *In Python, you can use the function* `k_means()` *from the package* `sklearn.cluster` *to compute k-means clustering. Import the function and type* `help(k_means)` *to learn how to use the function.*
- *The function can be called as* `centroids, clusters, inertia = k_means(X,K);` *where K is the number of clusters.*
- *Type* `clusterplot(X,clusters,centroids,y)` *to plot the data and the clustering.*
- *Type* `help(clusterplot)` *to learn more about how to use the clustering plot tool in the toolbox.*
- *To be more robust against different initial conditions, you can run multiple iterations of clustering with different initial centroid seeds, see the* `'n_init'` *parameter of* `k_means()`.

Does the clustering coincide with the true classes? Try running your code several times (with `n_init` set to 1) to show that the algorithm can fail if the initial conditions are poor. Try also the data sets `synth2`, `synth3`, and `synth4`.

For supervised learning we evaluated the model performance in terms of the error rate and accuracy. This however requires that we can match the estimated outcome to the true underlying classes, provided they are known. Even when we know the true underlying classes, we do not in general know which cluster corresponds to which class. Thus, by some means the true classes and the estimated clusters must be related to each other. One way of relating the two is to find out which cluster best matches each class such that each class is assigned one of the clusters and then calculate the error rate based on these clusters.

10.1.2 Is the above definition of the classification error rate for clustering reasonable if we extract the same number of clusters as classes in the data? Does the definition of the classification error make sense when the number of extracted clusters are different from the true underlying classes?

Rather than using the error rate we will consider the supervised measures of cluster validity, in particular Rand Statistic, Jaccard coefficient and normalized mutual information (NMI). Carefully review these measures in the book and make sure you understand how they are calculated.

10.1.3 The script `ex10_1_3.py` repeats exercise 10.1.1, but this time perform k-means clustering for $K = 1, \ldots, 10$ clusters. For each value of K the three cluster validity measures mentioned above are computed. Notice how the script plots the cluster validity measures as a function of K.

Script details:

- *It is a good exercise that write your own code for computing the cluster validity measures. You can look at simple example in the toolbox (function* `clusterval()`*).*
- *You can find many additional cluster validity measures in* `sklearn.metrics.cluster` *package. The script* `ex10_1_3.py` *shows how to use some of them (completeness score, homogeneity score, v-measure-score and others). You can find more details in* `sklearn` *package documentation.*

How can the cluster validity measures be used to select the best number of clusters?

What happens when more than four clusters are used to model the data?

10.1.4 For supervised learning we used cross-validation to evaluate performance and estimate the number of parameters in our models, i.e., the number of clusters.

Let us assume that we split the data into a training and a test set, train the k-means model on the training set and evaluate how well the model accounts for the test data. Consider evaluating the clustering by summing the distance of test points to the closest estimated cluster center obtained. What will happen with this training and test error as we increase the number of clusters?

K-means clustering has many different applications, one of which is data compression. A data set can be compressed by performing k-means clustering and then representing

each data object by its cluster center. Thus, the only data that need to be stored is the K cluster centers and the N cluster indices.

10.1.5 We will consider a subset of the wild faces data described in [2]. You can load the wildfaces data set from the `Data/wildfaces.mat` file with the `loadmat` function, see the script `ex10_1_5.py`. Each data object is a $40 \cdot 40 \cdot 3 = 4800$ dimensional vector, corresponding to a 3-color 40×40 pixels image. The script computes a k-means clustering of the data with $K = 10$ clusters. Plot a few random images from the data set as well as their corresponding cluster centroids to see how they are represented.

Script details:

- *You can plot an image by the command* `imshow(np.reshape(X[k,:],(c,x,y)).T)` *which reshapes an image vector to a 3-dimensional array and plots it. Similarly, you can plot the cluster centroids.*
- *Running k-means on a large data set can be slow. If you type* `k_means(X, K, verbose=True, max_iter=X, n_init=S)` *it will provide information about the iterations as they run.* `n_init` *as before constrains the number of initial repeated centroid seeds. Type* `help(k_means)` *to read more about these options.*

How well is the data represented by the cluster centroids? Are you able to recognize the faces in the compressed representation? What happens if you increase or decrease the number of clusters?

10.1.6 Modify the script `ex10_1_5.py` to repeat the previous exercise but with the digits data set. You can load the digits data set from the file `Data/digits`. Each data object is a $16 \cdot 16 = 256$ dimensional vector, corresponding to a gray scale 16×16 pixels image.

Script details:

- *You can change the color map to black-on-white grey-scale by adding paramter* `cmap=cm.binary` *to* `imshow()` *function.*

Why does running k-means with $K = 10$ not give you 10 clusters corresponding to the 10 digits 0–9? How many clusters do you need to visually represent the 10 different digits? Are there any digits that the clustering algorithm seems to confuse more than others?

10.2 Hierarchical clustering

We will in this part of the exercise consider hierarchical clustering based on the functions from the package `scipy.cluster.hierarchy`. Function `linkage()` forms a sample to sample distance matrix according to a given distance metric, and creates the linkages between data points forming the hierarchical cluster tree. Function `dendrogram` creates a plot of the generated tree. Function `fcluster` extracts the cluster from linkage matrix wrt given criterion. Use `help` for the three function and see how they are used and inspect what distance metrics and linkage functions are implemented.

10.2.1 Inspect and run the script `ex10_2_1.py`. The script loads the data set from the file `Data/synth1` and partitions dthe data using hierarchical clustering with single linkage using the Euclidean distance measure. Notice how the script is used to cluster the data into 4 clusters by cutting off dendrogram at a threshold and plots a dendrogram and a scatter plot of the clusters.

Script details:

- *The function* `linkage()` *computes the hierarchical clustering, resulting in a matrix representing the hierarchy of clusterings. Type* `help(linkage)` *to learn how to use it.*
- *The second parameter of* `linkage()` *can be used to select the method used in the hierarchical clustering procedure.*
- *The third parameter of* `linkage()` *can be used to change the distance measure.*
- *You can e.g. type* `Z = linkage(X, method='single', metric='euclidean)` *to use single linkage with the Euclidean distance measure.*
- *To compute a clustering, you can use the function* `fcluster()`*. For example, type* `cls = fcluster(Z, criterion='maxclust', t=4)` *to get a maximum of 4 clusters. Type* `help(fcluster)` *to learn more about what this function does.*
- *To plot a dendrogram, you can use the* `dendrogram()` *function.*
- *Again, you can use the function* `clusterplot()` *from the toolbox to plot a scatter plot of the clustering.*

Try changing the linkage method and see how it changes the dendrogram. Try running your code several times to see if it generates exactly the same dendrogram each time. Try also the data sets `synth2`, `synth3`, and `synth4`, and choose suitable distance measures for these data sets.

10.3 Old Faithful geyser data

Old Faithful is a famous geyser located in Yellow Stone national park in the US. The Old Faithful geyser dataset described in [1, 3] consists of $N = 272$ observations of

two variables, namely the duration of each eruption in minutes (duration) and the waiting time between eruptions also in minutes (waiting).

10.3.1 Load the Old Faithful data in the file `load Data/faithful.mat`. Analyze the data by `k-means` visually inspect the labeling of the data points.

What happens if you increase K beyond the obvious $K = 2$? Try to run the program a couple of times (resulting in different initial conditions). Do you see the same solution every time? The scaling of the variables can seriously affect the results we get in clustering. Discuss whether it is more reasonable to normalize the Old faithful data set? Try normalizing the data and see whether the results of running `k-means` change.

10.3.2 Run hierarchical clustering of the Old Faithful data using the "single" and "ward" linkage, with and without normalizing the data. Do you find support for a two cluster model from the structure of the dendrograms?

10.4 Extra Challenge

10.4.1 Try clustering some of the data sets you have analyzed in the previous exercises such as the Iris data and the wine data using k-means and hierarchical clustering.

1 Homework problems for this week

Problems

Question 1. Fall 2013 question 8: In Table 1 is given the pairwise distances between the four smallest and four largest islands in the Galápagos data. A hierarchical clustering is used to cluster these eight observations using single (i.e., minimum) linkage. Which one of the dendrograms given in Figure 1 corresponds to the clustering?

	O1	O2	O3	O4	O5	O6	O7	O8
O1	0	2.39	1.73	0.96	3.46	4.07	4.27	5.11
O2	2.39	0	1.15	1.76	2.66	5.36	3.54	4.79
O3	1.73	1.15	0	1.52	3.01	4.66	3.77	4.90
O4	0.96	1.76	1.52	0	2.84	4.25	3.80	4.74
O5	3.46	2.66	3.01	2.84	0	4.88	1.41	2.96
O6	4.07	5.36	4.66	4.25	4.88	0	5.47	5.16
O7	4.27	3.54	3.77	3.80	1.41	5.47	0	2.88
O8	5.11	4.79	4.90	4.74	2.96	5.16	2.88	0

Table 1: Pairwise Euclidean distance, i.e $d(Oa, Ob) = ||\boldsymbol{x}_a - \boldsymbol{x}_b||_2 = \sqrt{\sum_m (x_{am} - x_{bm})^2}$, between eight observations of the Galápagos data. Red observations (i.e., O1, O2, O3, and O4) correspond to the four smallest islands whereas blue observations (i.e., O5, O6, O7, and O8) correspond to the four largest islands.

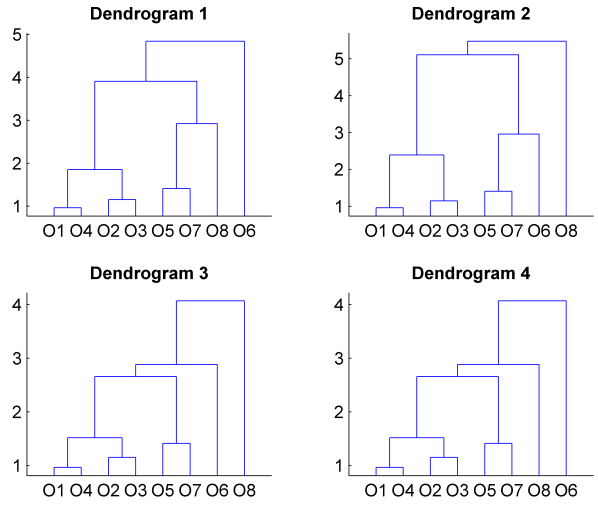

Figure 1: Hierarchical clustering of the eight observations considered in Table 1.

A Dendrogram 1.

B Dendrogram 2.

C Dendrogram 3.

D Dendrogram 4.

E Don't know.

Question 2. Fall 2014 question 20: Consider the simple 1-dimensional data set comprised of $N = 7$ observations as shown in table 2. Suppose we wish to apply K-means clustering to the dataset and the $K = 3$ one-dimensional cluster centers are initialized in $\mu_1 = 4$, $\mu_2 = 7$ and $\mu_3 = 14$. After terminating of the K-means clustering algorithm, what are the final (rounded) cluster centers μ_1, μ_2, μ_3?

X	3	6	7	9	10	11	14

Table 2: Simple 1-dimensional dataset comprised of $N = 7$ observations.

A $\mu_1 = 3.00, \mu_2 = 8.00, \mu_3 = 12.50$

B $\mu_1 = 3.00, \mu_2 = 7.33, \mu_3 = 11.67$

C $\mu_1 = 4.50, \mu_2 = 9.25, \mu_3 = 14.00$

D $\mu_1 = 5.33, \mu_2 = 10.00, \mu_3 = 14.00$

E Don't know.

Question 3. Fall 2014 question 11: In table 3 is given the pairwise cityblock distances between 8 observations. A hierarchical clustering is used to cluster these nine observations using *group average* linkage. Which of the dendrograms shown in fig. 2 corresponds to the clustering?

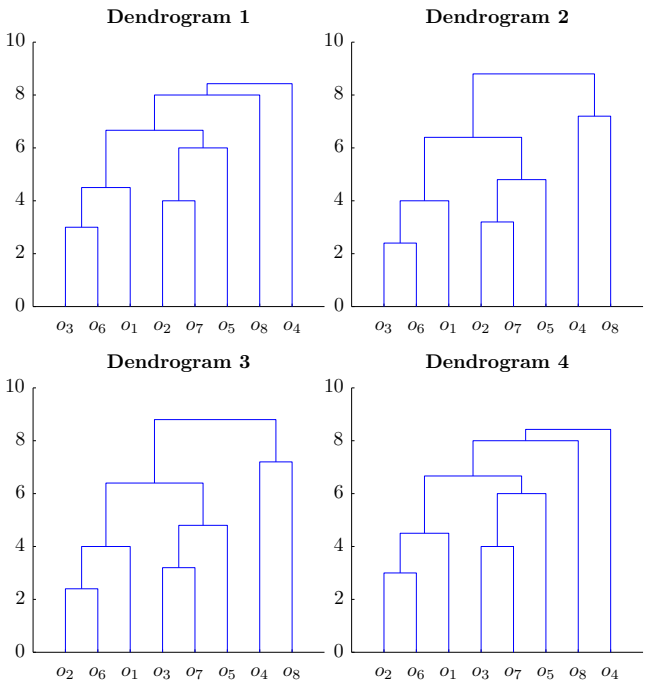

Figure 2: Hierarchical clustering of the 8 observations considereded in table 3

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
o_1	0	4	7	9	5	5	5	6
o_2	4	0	7	7	7	3	7	8
o_3	7	7	0	10	6	6	4	9
o_4	9	7	10	0	8	6	10	9
o_5	5	7	6	8	0	8	6	7
o_6	5	3	6	6	8	0	8	11
o_7	5	7	4	10	6	8	0	7
o_8	6	8	9	9	7	11	7	0

Table 3: Pairwise Cityblock distance, i.e $d(o_i, o_i) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1 = \sum_{k=1}^{M} |x_{ik} - x_{jk}|$, between 8 observations. Each observation o_i corresponds to a $M = 15$ dimensional binary vector, $x_{ik} \in \{0, 1\}$. The blue observations $\{o_1, o_2, o_3, o_4\}$ belong to class C_1 and the black observations $\{o_5, o_6, o_7, o_8\}$ belong to class C_2.

A Dendrogram 1.

B Dendrogram 2.

C Dendrogram 3.

D Dendrogram 4.

E Don't know.

References

[1] A Azzalini and AW Bowman. A look at some data on the old faithful geyser. *Applied Statistics*, pages 357–365, 1990.

[2] Tamara L Berg, Alexander C Berg, Jaety Edwards, and DA Forsyth. Who's in the picture. *Advances in neural information processing systems*, 17:137–144, 2005.

[3] Wolfgang Härdle. *Smoothing techniques: with implementation in S*. Springer, 1991.

Mixture models and density estimation with PYTHON

Objective:

To understand the Gaussian mixture model; how Expectation-Maximization is used to estimate the model parameters; and how information criteria and cross-validation is used to choose the number of clusters. How the density of data can be estimated by histograms, Gaussian mixture models, kernel density estimators and k-nearest neighbor density estimation, as well as understand density based and distance based outlier/anomaly detection methods.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 11 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`⋆_train`	—	—	Training data.
	`⋆_test`	—	—	Test data.

11.1 The Gaussian Mixture Model and the EM algorithm

In the exercise last week we considered k-means clustering and hierarchical clustering. Today we will consider clustering based on the Gaussian mixture model (GMM). We recall that the multivariate Gaussian distribution is given by

$$\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_{(k)}, \boldsymbol{\Sigma}_{(k)}) = \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma}_{(k)})}} \exp\big\{ -\tfrac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_{(k)})^\top \boldsymbol{\Sigma}_{(k)}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_{(k)})\big\}.$$

In the Gaussian mixture model we use a mixture of K multivariate Gaussians to model the data. We give each Gaussian a mixture coefficient w_k between zero and one such that all coefficients sum to one, $\sum_{k=1}^{K} w_k = 1$. The probability of a data

vector $\boldsymbol{x}_i$ is then modeled as a weighted sum of Gaussian distributions,

$$p(\boldsymbol{x}_i|\boldsymbol{w}, \{(\boldsymbol{\mu}_{(1)}, \boldsymbol{\Sigma}_{(1)}), \ldots, (\boldsymbol{\mu}_{(K)}, \boldsymbol{\Sigma}_{(K)})\}) = \sum_{k=1}^{K} w_k \cdot \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_{(k)}, \boldsymbol{\Sigma}_{(k)})$$

The parameters of the model, which comprises the mixture coefficients and the K means and covariance matrices, are found by the Expectation-Maximization (EM) algorithm that progress in the following way:

1. Initialize the mixture coefficient $\boldsymbol{w}$, mean and covariance of each Gaussian $\boldsymbol{\mu}_{(k)}$ and $\boldsymbol{\Sigma}_{(k)}$ by random.
2. (E-step) Calculate the expectation $P(k|\boldsymbol{x}_i, \boldsymbol{w}, \{(\boldsymbol{\mu}_{(1)}, \boldsymbol{\Sigma}_{(1)}), \ldots, (\boldsymbol{\mu}_{(K)}, \boldsymbol{\Sigma}_{(K)})\})$ for each data point.
3. (M-step) Optimize $\boldsymbol{w}$ and $\{(\boldsymbol{\mu}_{(1)}, \boldsymbol{\Sigma}_{(1)}), \ldots, (\boldsymbol{\mu}_{(K)}, \boldsymbol{\Sigma}_{(K)})\}$ by maximizing the expected likelihood.
4. Keep doing 2 and 3 until the clusters do not change or a maximum number of iterations have progressed.

The EM-algorithm is closely related to the k-means algorithm but rather than operating with hard assignment of each data point, each data point is assigned a given probability of belonging to each cluster based on Bayes rule,

$$P(k|\boldsymbol{x}_i, \boldsymbol{w}, \{(\boldsymbol{\mu}_{(1)}, \boldsymbol{\Sigma}_{(1)}), \ldots, (\boldsymbol{\mu}_{(K)}, \boldsymbol{\Sigma}_{(K)})\}) = \frac{w_k \cdot N(\boldsymbol{x}_i|\boldsymbol{\mu}_{(k)}, \boldsymbol{\Sigma}_{(k)})}{\sum_{k=1}^{K} w_k \cdot N(\boldsymbol{x}_i|\boldsymbol{\mu}_{(k)}, \boldsymbol{\Sigma}_{(k)})},$$

while each cluster is not only described by its center (mean) but also by its covariance. In order to get hard assignments of the data points from the results obtained by the above EM-algorithm we assign the observations to the clusters with highest probability.

An important property of the estimated Gaussian mixture model is that the GMM density integrates to one, i.e.

$$\int \sum_k w_k \cdot \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{(k)}, \boldsymbol{\Sigma}_{(k)}) d\boldsymbol{x} = \sum_k w_k \int \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{(k)}, \boldsymbol{\Sigma}_{(k)}) d\boldsymbol{x} = 1$$

This follows from the fact that each Gaussian integrates to one $\int \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{(k)}, \boldsymbol{\Sigma}_{(k)}) d\boldsymbol{x} = 1$ and the constraint $\sum_{k=1}^{K} w_k = 1$. An important consequence of this is that when additional clusters are introduced they have to take density mass from existing clusters.

11.1.1 Inspect and run the script `ex11_1_1.py`. The script loads the synth1 data from the file `Data/synth1` with the `loadmat` function and fits a $K = 4$ component Gaussian mixture model to the data. Notice how the script makes a scatter plot of the data and the clustering using the `clusterplot` function in the toolbox.

Script details:

- *To fit a Gaussian Mixture Model in Python you can use the class* `GaussianMixture` *from the package* `sklearn.mixture`*. You will need two methods:* `fit` *and* `predict`*. Type* `help(GaussianMixture)` *to learn how to use the class.*
- *When creating GaussianMixture object you can specify number of clusters, covariance type (full/diagonal), number of repetitions with different initial seeds.*
- *You can extract the fitted cluster means (centroids) by calling the method* `means_` *of* `GaussianMixture` *class object.*
- *Type* `clusterplot(X,clusterid,centroids,cov_matrices)` *to plot the data and clustering.*
- *Type* `help(clusterplot)` *to learn more about how to use the clustering plot tool in the toolbox. Note that you can specify* `covars` *parameter to plot ellipsoids of the gaussians corresponding to GMM.*

Sometimes the estimated models for the same value of K are not necessarily the same due to different initial placement of the centroids. In such cases you might try setting the parameter `n_init` to 10 or more. What will this do, and why should it solve the problem?

Try to change the structure of the covariance matrix to be a diagonal matrix by setting the parameter `covariance_type` to `'diagonal'`. How does this affect the generated clusters? What do you think might be benefits and drawbacks of restricting the covariance matrices?

11.1.2 **Discussion:** In k-means based on Euclidean distance observations are assigned the centroids they are the closest to. Is this also the case when clustering by the Gaussian mixture model or is it possible that points are assigned a cluster that is further away than other clusters in terms of Euclidean distance?

Can the scaling of the variables seriously affect the results we get when clustering by the GMM or is the model able to take the scaling of the data into account?

11.1.3 **Discussion:** For supervised learning we used cross-validation to evaluate performance and estimate the number of parameters in our models. In

last weeks exercise, we found that this approach could not be used in the k-means algorithm. What happens if we validate the number of clusters for Gaussian mixture model based on the EM algorithm using cross-validation, i.e., split the data into training and test data, train the model on the training data and evaluate how likely the test data points are based on the learned parameters, $\boldsymbol{w}$ and $\{\boldsymbol{\mu}_{(1)}, \boldsymbol{\Sigma}_{(1)}, \ldots, \boldsymbol{\mu}_{(K)}, \boldsymbol{\Sigma}_{(K)}\}$ using the logarithm of the likelihood of the test data,

$$\log L^{\text{test}} = \sum_{i=1}^{N_{\text{test}}} \log[p(\boldsymbol{x}_i^{(\text{test})} | \boldsymbol{w}, \{(\boldsymbol{\mu}_{(1)}, \boldsymbol{\Sigma}_{(1)}), \ldots, (\boldsymbol{\mu}_{(K)}, \boldsymbol{\Sigma}_{(K)})\})]?$$

Script details:

- *Remember, that in the GMM each cluster is represented by a mean vector and a covariance matrix. In one dimension, it is the well known bell-shaped probability distribution.*
- *If we have 100 data points and use 1 cluster, what would be the optimal solution for the GMM using the EM-algorithm? How well would this solution generalize to test data?*
- *If we have 100 data points and use 100 clusters, what would be the optimal solution? How well would this solution generalize?*
- *What if we use some intermediate number of clusters?*

Apart from cross-validation the optimal number of clusters are sometimes derived by penalizing model complexity based on the Bayesian Information Criteria (BIC) or Akaike's Information Criteria (AIC). The two information criteria are defined by

$$\text{BIC} = -2 \log L + p \log(N), \qquad \text{AIC} = -2 \log L + 2p$$

where p is the number of free parameters in the model, i.e., the total number of estimated variables in $\boldsymbol{w}$, $\{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K\}$ whereas N is the number of observations and $\log L$ is the log likelihood of observing the data, i.e.,

$$\log L = \sum_{i=1}^{N} \log[p(\boldsymbol{x}_i | \boldsymbol{w}, \{\boldsymbol{\mu}_{(1)}, \boldsymbol{\Sigma}_{(1)}, \ldots, \boldsymbol{\mu}_{(K)}, \boldsymbol{\Sigma}_{(K)}\})].$$

The two information criteria define a trade-off between modeling the data well, i.e., minimize $-2 \log L$, and penalizing complexity of the model, i.e., $M \log(N)$ and $2M$ respectively, such that the model with lowest AIC and BIC value indicates the model with best trade-off. Note, that AIC and BIC do not require splitting the data in test and training sets, but are computed directly on the whole training data.

11.1.4 **Discussion:** Which of the two information criteria BIC and AIC will in general penalize model complexity the most?

11.1.5 Inspect and run the script `ex11_1_5.py`. We will use BIC, AIC, and 10-fold crossvalidation to assess the best number of clusters for the synth1 data set. Use the script to compute the three measures for $K = 1, \ldots, 10$, and use 10 replicates to avoid bad solutions due to poor initial conditions.

Script details:

- *As before, use* `sklearn.mixture.GaussianMixture` *class to fit the models.*
- *As usual, use the module* `sklearn.model_selection` *to set up the crossvalidation folds.*
- `GaussianMixture` *class has* `bic` *and* `auc` *methods to compute BIC and AUC scores automatically. Type* `help(GaussianMixture)` *to read more.*
- *To evaluate the model fit on the test set in terms of negative log likelihood, you can use the method* `score` *(after fitting the model). For instance:*
  `NLOGL = -gmm.score(X_test).sum()`.

What are the benefits and drawbacks of AIC and BIC versus cross-validation?

11.1.6 **(Optional):** Use a Gaussian mixture model to cluster the Old Faithful data set, `Data/faithful`, using AIC, BIC, or crossvalidation to select the number of clusters.

11.1.7 **(Optional):** Use a Gaussian mixture model to cluster the Iris data set, `Data/iris.xls`, using AIC, BIC, or crossvalidation to select the number of clusters.

11.2 Density estimation

We will apart from the Gaussian mixture model (we just used) now consider the following two approaches to density estimation: kernel density estimation, and k-nearest neighbors density estimation.

Kernel density estimation is a non-parametric method of estimating the probability density function of a random variable. Inference about the population are made, based on a finite data sample. The estimated kernel density for a variable x is given by

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} K(x - x_i) \tag{1}$$

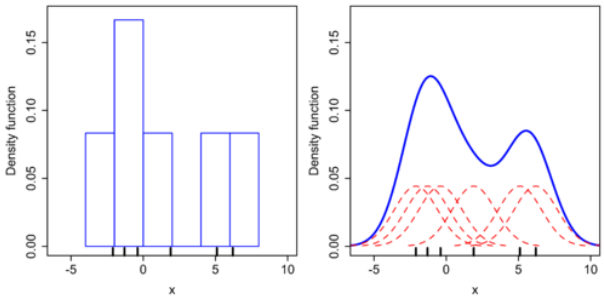

Figure 1: Estimation of the data density by histogram (left) and kernel density estimation (right). The Kernel specifies a shape around each data point (here a univariate normal distribution) and the density is estimated by summing over these Gaussians placed at each observation

where K is the kernel function that must integrate to one, N is the sample size, and $f(x)$ is the estimated density (see also figure 1.)

We will presently consider density estimation based on the kernel formed by the normal distribution [1] , i.e.,

$$K(x - x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(- \frac{(x - x_i)^2}{2\sigma^2} \Big).$$

We will initially consider an artificially generated one-dimensional data set formed by a mixture of three Gaussians defined as follows

$$p(x) = \frac{1}{3}\mathcal{N}(x|1, 1) + \frac{1}{3}\mathcal{N}(x|3, 0.5) + \frac{1}{3}\mathcal{N}(x|6, 2) \ . \tag{2}$$

11.2.1 Inspect and run the script `ex11_2_1.py`. Notice how the script generates 1000 data objects according to the model in equation (2), and plots a histogram of the data with 50 bins in the range -10 to 10.

Script details:

[1] Notice, that for this choice of kernel function the kernel density estimation can be considered a Gaussian mixture model where the number of clusters is the same as the number of observations, i.e., there is a Gaussian around each observation, while the mixing coefficient is fixed to be the same for all classes, $w_n = 1/N$.

- *You can generate data from the Gaussian mixture model in the following way: First, you choose one of the components by random according to the mixture coefficients. Then, you generate the data according to the Gaussian distribution for the chosen component.*
- *Type* `help(np.random.multinomial)` *to learn how to generate a random variable from a discrete distribution.*
- *You can use* `help(np.random.normal)` *function to draw samples from normal distribution (particular gaussian from the mixture).*
- *The function* `numpy.hist` *can be used to plot a histogram. Use* `help(numpy.hist)` *to get help.*
- *To define the bins at which the histogram should be evaluated, you can define a vector of values and pass it to* `hist`*. To get 50 bins evenly spaced between* -10 *and* 10*, you can use the command* `x=np.linspace(-10,10,50)`*.*

Discussion:

- ◇ Can you identify each component of the mixture in the plot and its associated parameters (mean, variance, mixture coefficient?)
- ◇ Try changing the parameters of the mixture to see the effect on the resulting density.
- ◇ Try varying the number of bins in the histogram.

11.2.2 Inspect and run the script `ex11_2_2.py`. Explain how the script estimates the density using a kernel density estimator with a Gaussian kernel and a kernel width of 1, and plot the density on the range -10 to 10.

Script details:

- *Use* `gaussian_kde` *from the module* `scipy.stats.kde` *to fit a kernel density estimator.*
- *To define at which x-values the KDE should be evaluated, you can define a vector of values and pass it to the* `evaluate` *method of the fitted KDE estimator. For example, to get 100 points evenly spaced between* -10 *and* 10*, you can use the command* `x=np.linspace(-10,10,100)`

Discussion:

- ◇ Compare the result to the histogram.
- ◇ Try varying the kernel width in the KDE.
- ◇ How could you select an optimal kernel width?

Consider the following measure of density of the i^{th} observation $\boldsymbol{x}_i$ given based on its k-nearest neighbors (KNN)

$$\text{density}_{\boldsymbol{X}_{\setminus i}}(\boldsymbol{x}_i, K) = \frac{1}{\frac{1}{K}\sum_{\boldsymbol{x}' \in N_{\boldsymbol{X}_{\setminus i}}(\boldsymbol{x}_i, K)} d(\boldsymbol{x}_i, \boldsymbol{x}')}, \tag{3}$$

where $N_{\boldsymbol{X}}(\boldsymbol{x}, K)$ is the K observations in $\boldsymbol{X}$ which are nearest to $\boldsymbol{x}$, and $\boldsymbol{X}_{\setminus i}$ is simply $\boldsymbol{X}$ with observation i removed: $\boldsymbol{X}_{\setminus i}^T = \begin{bmatrix}\boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots \boldsymbol{x}_{i-2} & \boldsymbol{x}_{i-1} & \boldsymbol{x}_{i+1} & \boldsymbol{x}_{i+2} & \cdots & \boldsymbol{x}_N\end{bmatrix}$. This estimates the density as the inverse of the average distance to the k nearest neighbors.

If the dataset contains regions of varying densities it can be useful to define a notion of density that is relative to the neighborhood of the object. The following average relative density is one such approach

$$\text{ard}_{\boldsymbol{X}}(\boldsymbol{x}_i, K) = \frac{\text{density}_{\boldsymbol{X}_{\setminus i}}(\boldsymbol{x}_i, K)}{\frac{1}{K}\sum_{\boldsymbol{x}_j \in N_{\boldsymbol{X}_{\setminus i}}(\boldsymbol{x}_i, K)} \text{density}_{\boldsymbol{X}_{\setminus j}}(\boldsymbol{x}_j, K)}. \tag{4}$$

11.2.3 Inspect and run `ex11_2_3.py` and see how the script can be used to estimate the density using equation (3) as well as the average relative density using equation (4) with the 200-nearest neighbors based on the Euclidean distance measure, and plot the density on the range -10 to 10.

Script details:

- *To define at which x-values the nearest neighbor density estimator should be evaluated, you can define a vector of values, e.g., to get 100 points evenly spaced between -10 and 10, you can use the command* `x=np.linspace(-10,10,100)`.
- *To find the nearest neighbors and the distances needed to compute equation (3), you can fit the* `NearestNeighbors` *model from the module* `sklearn.neighbors` *and then use the method* `kneighbors()` *of to extract nearest neighbors distances.*

Discussion:

- ◇ Try different values for the number of neighbors.
- ◇ Compare your results to the three previous exercises.

11.3 Outlier detection

The following two definitions of an outlier are common:

Hawkin's Definition of an Outlier. An outlier is an observation that differs so much from other observations as to arouse suspicion that it was generated by a different mechanism.

Probabilistic Definition of an Outlier. An outlier is an object that has a low probability with respect to a probability distribution model of the data.

Thus, in order to detect outliers we need a mechanism that can establish whether an observation differs so much from other observations as to be deemed an outlier. An important tool to evaluate this is to create a model of the data density and then evaluate how likely the observations are given the density.

11.3.1 Inspect and run the script `ex11_3_1.py`. Prior to running the script you need to execute the command `X[-1,0]=-10` to add an outliner at -10 to the data. The script uses a kernel density estimator (as in exercise 11.2.2) with a Gaussian kernel to estimate the density at the x-values in the data set. Verify that the outlier you have introduced has the lowest density. Notice how the bar plot corresponds to the density of the 20 lowest-density points.

Script details:

- *You originally had $N = 1000$ data objects. What is the index of the introduced outlier?*
- *To get the KDE for each data object in the data set, you have to fit KDE model first (class* `gaussian_kde()` *from the module* `scipy.stats.kde`*), and then use* `evaluate()` *method.*
- *To find the indices and values of the 20 lowest-density points, you can use the methods* `argsort()` *and* `sort()` *respectively, on the output from* `evaluate()` *method.*
- *To make a bar plot, use the function* `bar`*.*

Discussion:

- ◇ Can the outlier be detected from this plot?
- ◇ Try different values of the kernel width.
- ◇ What happens when the kernel width is too large / too small?

11.3.2 The function `gausKernelDensity()` in the toolbox implements density estimation by the gaussian kernel density estimator using a very efficient implementation of leave-one-out cross-validation. I.e. the density of each observation is estimated from all other observations not including the observation itself in the estimate. Inspect and run the script `ex11_3_2.py` where it is used. Use the script to estimate the optimal kernel width by evaluating the estimated densities for a range of different kernel widths. Plot the leave-one-out density of the 20 lowest-density points in a bar plot. Script details:

- *To get the leave-one-out KDE for each data object in the data set for a kernel width of 5, you can use the command* `f,log_f=gausKernelDensity(X,5)`.
- *The log-density for all observations is given by* `logP=log_f.sum()`. *The optimal kernel width is the width with highest* `logP`.
- *To find the 20 lowest-density points, you can again use the function* `sort` *to sort the output from* `ksdensity`.
- *To make a bar plot, use the function* `bar`.

11.3.3 **(Optional):** Repeat exercise 11.3.1 using the KNN-density estimator as well as the KNN-average relative density.

11.4 Hand written digits

We will in this part of the exercise investigate if some of the hand written digits of each class can be considered outliers. To do this, inspect and run the script `ex11_4_1.py`.

11.4.1 Load the hand written digits data set from the file `Data/digits.mat` with `loadmat()` function. Restrict your analysis to images of the digit "2" by the command `X=X[y==2,:]`.

1) Use the function `gausKernelDensity()` from toolbox that implements density estimation by the gaussian kernel density estimator using the very efficient implementation of leave-one-out density estimation (as in exercise 11.3.2). Estimate the optimal kernel width evaluating a range of different widths. Make a bar plot of the leave-one-out densities of the 20 lowest-density data objects in the data set based on the optimal kernel width.

2) Use the KNN density estimation method (as in exercise 11.2.4) based on the Euclidean distance measure and using $K = 5$ neighbors. Make a

bar plot of the densities of the 20 lowest-density data objects in the data set.

3) Use the KNN average relative density estimation method (as in exercise 11.2.4) also based on the Euclidean distance measure and using $K = 5$ neighbors. Make a bar plot of the densities of the 20 lowest-density data objects in the data set.

Plot the images of the 20 data objects with the lowest density / highest outlier score.

Script details:

- *The images are 16 by 16 pixels stored as 256 dimensional vectors. To plot an image, you must reshape the vector to a 16 by 16 matrix and plot it using the* `imshow()` *function. For example, to plot the* `i`*'th image in* `X`*, use the command:* `imshow(np.reshape(X[i-1,:], (16,16)).T, cmap=cm.binary)`.

Discussion:

- ◇ Does the three bar plots reveal any possible outliers in the data set?
- ◇ Can you identify any data objects that really are outliers?
- ◇ Do the three methods agree?
- ◇ Try plotting 20 random images of hand written 2's (for example the first 20 in the data set) for comparison. In what sense are the low-density 2's different from the randomly chosen ones?
- ◇ Try other digits than "2". Does the method in general identify digits that are outliers?
- ◇ Try the method on the whole data set with all digits (warning: this might take a long time to compute.)

1 Homework problems for this week

Problems

Question 1. Fall 2013 question 11:

	O1	O2	O3	O4	O5	O6	O7
O8	5.11	4.79	4.90	4.74	2.96	5.16	2.88

Table 1: Pairwise Euclidean distance, i.e $d(Oa, Ob) = ||\boldsymbol{x}_a - \boldsymbol{x}_b||_2 = \sqrt{\sum_m (x_{am} - x_{bm})^2}$, between observation O8 and observation O1–O7 given in Table 2.

We would like to quantify if O8 is an outlier using a Gaussian kernel density estimator where we use the seven observations O1, O2, ..., O7 to estimate the density at observation O8 based on the Euclidean distances given in Table 2 and reproduced in terms of observation O8 in Table 1. The Gaussian kernel density estimator is given by

$$p(x) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{(2\pi)^{M/2}\sqrt{|\sigma^2 \boldsymbol{I}|}} \exp\left(-\tfrac{1}{2}(\mathbf{x} - \mathbf{x}_n)^\top (\frac{1}{\sigma^2}\boldsymbol{I})(\mathbf{x} - \mathbf{x}_n)\right)$$

Thus, $N = 7$ and in our analysis we will use $\sigma = 1$ and as the dataset is 7-dimensional we have that M=7. We note that $|\sigma^2 \boldsymbol{I}|$ is the determinant of the diagonal matrix with σ^2 in the diagonal. For $\sigma = 1$ we have $|\sigma^2 \boldsymbol{I}| = 1$. What is the density at observation O8 using only observations O1-O7 in the above Gaussian kernel density estimator

	O1	O2	O3	O4	O5	O6	O7	O8
O1	0	2.39	1.73	0.96	3.46	4.07	4.27	5.11
O2	2.39	0	1.15	1.76	2.66	5.36	3.54	4.79
O3	1.73	1.15	0	1.52	3.01	4.66	3.77	4.90
O4	0.96	1.76	1.52	0	2.84	4.25	3.80	4.74
O5	3.46	2.66	3.01	2.84	0	4.88	1.41	2.96
O6	4.07	5.36	4.66	4.25	4.88	0	5.47	5.16
O7	4.27	3.54	3.77	3.80	1.41	5.47	0	2.88
O8	5.11	4.79	4.90	4.74	2.96	5.16	2.88	0

Table 2: Pairwise Euclidean distance, i.e $d(Oa, Ob) = ||\boldsymbol{x}_a - \boldsymbol{x}_b||_2 = \sqrt{\sum_m (x_{am} - x_{bm})^2}$, between eight observations of the Galápagos data. Red observations (i.e., O1, O2, O3, and O4) correspond to the four smallest islands whereas blue observations (i.e., O5, O6, O7, and O8) correspond to the four largest islands.

A $3.4 \cdot 10^{-32}$

B $1.3 \cdot 10^{-8}$

C $6.5 \cdot 10^{-6}$

D $5.1 \cdot 10^{-2}$

E Don't know.

Question 2. Fall 2013 question 22: Let

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right)$$

define the multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. In Figure 2 is given 5000 observations drawn from a density defined by a Gaussian Mixture Model (GMM) with three clusters.

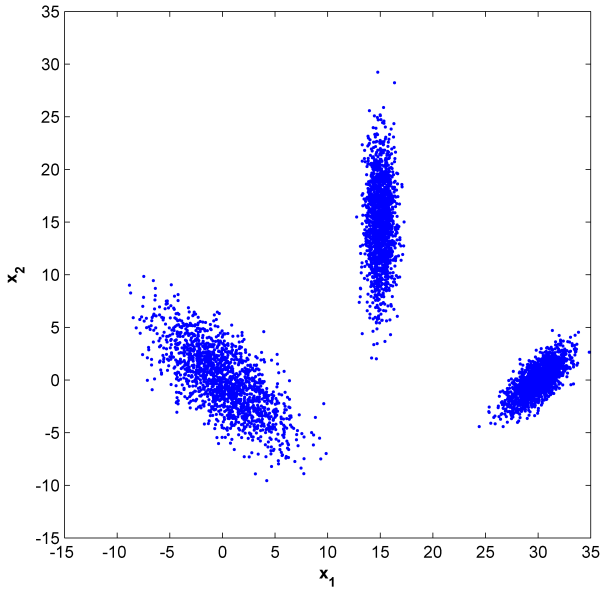

Figure 2: 5000 data observations drawn from a Gaussian Mixture Model (GMM) with three clusters.

Which one of the following GMM densities was used to generate the data?

A

$$p(x) = \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 15 \\ 15 \end{bmatrix}, \begin{bmatrix} 0.5 & 0 \\ 0 & 15 \end{bmatrix}) + \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 30 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & -1.4 \\ -1.4 & 2 \end{bmatrix}) + \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 10 & 7 \\ 7 & 10 \end{bmatrix})$$

B

$$p(x) = \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 15 \\ 15 \end{bmatrix}, \begin{bmatrix} 0.5 & 0 \\ 0 & 15 \end{bmatrix}) + \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 30 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 1.4 \\ 1.4 & 2 \end{bmatrix}) + \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 10 & -7 \\ -7 & 10 \end{bmatrix})$$

C

$$p(x) = \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 15 \\ 15 \end{bmatrix}, \begin{bmatrix} 0.5 & 0 \\ 0 & 15 \end{bmatrix}) + \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 30 \\ 0 \end{bmatrix}, \begin{bmatrix} 10 & -7 \\ -7 & 10 \end{bmatrix}) + \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 1.4 \\ 1.4 & 2 \end{bmatrix})$$

D

$$p(x) = \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 15 \\ 15 \end{bmatrix}, \begin{bmatrix} 15 & 0 \\ 0 & 0.5 \end{bmatrix}) + \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 30 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 1.4 \\ 1.4 & 2 \end{bmatrix}) + \frac{1}{3} \cdot \mathcal{N}(\boldsymbol{x}| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 10 & -7 \\ -7 & 10 \end{bmatrix})$$

E Don't know.

Question 3. Fall 2013 question 26: Which one of the following statements pertaining to clustering is *correct*?

A k-means and Gaussian Mixture Models are guaranteed to find the same solutions regardless of initialization.

B The level at which clusters merge in the dendrogram in hierarchical clustering using minimum/single-, maximum/complete- or group average linkage can be determined by the proximities between all the observations.

C In k-means the cluster centers are updated as the average of the observations belonging to the cluster regardless of the distance measure used.

D A Gaussian Mixture Model with diagonal covariance matrix has the same number of free parameters as k-means.

E Don't know.

References

Association mining with PYTHON

Objective: The objective of this exercise is to understand association mining, how frequent itemsets can be extracted by the Apriori algorithm and be able to calculate and interpret association rules in terms of support and confidence.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 12 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/60254/discussions/List

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`⋆_train`	—	—	Training data.
	`⋆_test`	—	—	Test data.

12.1 Association Analysis

In this last exercise we will focus on association analysis. Association analysis is widely used in data mining in order to identify important co-occurrence relationships. We will use the following definition of association rule discovery:

> **Association Rule Discovery.** Given a set of transactions T, find all the rules having support $\geq$ *minsup* and confidence $\geq$ *minconf*, where *minsup* and *minconf* are the corresponding support and confidence thresholds.

We have summarized the most important terms in table 1. We will use the Apriori algorithm to find all itemsets with support greater than $\geq$ supp. The Apriori algorithm is based on the following principle:

Apriori principle. If an itemset is frequent, then all of its subsets must also be frequent.

As a result of the Apriori principle we can start looking at frequent 1-itemsets. The frequent 2-itemsets can then only contain the items in the extracted 1-itemsets and so on and so forth. This greatly reduces the number of itemsets to check to find all frequent itemsets.

Term	**Meaning**
$I = \{i_1, i_2, \ldots, i_d\}$	The set of all items
$T = \{t_1, t_2, \ldots, t_N\}$	The set of all transactions
Transaction, t_i	A subset of items: What was bought by a customer
Transaction width	Number of items in transaction
Itemset	A set of items from the set I of all items
k-itemset	An itemset having k items
Support count, $\sigma(X)$	Number of transactions that contain a particular itemset $\sigma(X) = \lvert\{t\}\rvert$
Association rule	Implication expression of the form $X \leftarrow Y$ where $X \bigcap Y = \varnothing$
Support, $s(Y \leftarrow X)$	Strength of association rule, $s(Y \leftarrow X) = \frac{\sigma(X \bigcup Y)}{N} = P(X, Y)$
Confidence, $c(Y \leftarrow X)$	Frequency items in Y appear in transactions containing X, $c(Y \leftarrow X) = \frac{\sigma(X \bigcup Y)}{\sigma(X)} = \frac{P(X,Y)}{P(X)} = P(Y\lvert X)$
Support-based pruning	Pruning strategy based on the Apriori principle (formed by the anti-monotone property)
Anti-monotone property	The support for an itemset never exceeds the support for its subsets (Apriori principle)
F_k	The set of frequent k-itemsets

Table 1: Association mining nomenclature.

12.1.1 In table 2 some of the courses that 6 students completed during their studies are given. Find all itemsets with supp $\geq 80\%$.

12.1.2 What is the confidence of the rule 02457 $\leftarrow$ 02450?

	02322	02450	02451	02453	02454	02457	02459	02582
student 1	0	1	0	0	1	1	1	1
student 2	1	1	1	0	0	1	1	1
student 3	0	1	0	1	0	1	0	1
student 4	0	0	1	0	0	1	1	0
student 5	0	1	0	0	0	1	1	0
student 6	0	1	1	0	0	1	1	1

Table 2: Students that upon completing their engineering degree had taken various of the courses 02322, 02450, 02451, 02453, 02454, 02457, 02459 and 02582.

We will use the Apriori algorithm to automatically mine for associations[1]. To use the Apriori algorithm, simply install the package `apyori` using `conda install apyori` (or `pip install apyori` if you are not using Anaconda).

12.1.3 Inspect the file `Data/courses.txt` and the script `ex12_1_3.py`. The script loads the course data file from table 2. Make sure you understand how the data in table 2 is stored in the file and how the script transforms it

12.1.4 Inspect and run the script `ex12_1_4.py`. The script transforms the binary matrix, plus label information, into a set of transactions. Make sure you understand how this transformation performs and relate it to the notation of the lecture notes. Finally note how the Apriori algorithm is invoked to find association rules with supp $\geq 80\%$ and conf $\geq 100\%$ and print them. Inspect the print command to see how you can access each association rule programmatically. What are the generated association rules?

We will in this last part of the exercise mine for associations in the wine data [1] (`http://archive.ics.uci.edu/ml/datasets/Wine+Quality`) considered in the previous exercises. However, as this data is not binary we will need to convert it to a format suitable for association mining. We will thus binarize the data by dividing each attribute into given percentiles.

12.1.5 Inspect and run the script `ex12_1_5.py`. The script load the `Data/wine2.mat` data into Python (for how to load .mat files into Python see also exercise

[1] A high-performing version of the Apriori algorithm is also available from: `http://www.borgelt.net/apriori.html`.

4.2.1) and divide each of the attributes in the data into percentiles using the function `binarize2`.

The scripts convert the continuous attributes into a one-out-of-K coding based on percentiles. Note how the function also transforms the attribute names. Why do you think we don't just include the 50-100 percentiles of a variable? What are the benefits of including variables corresponding to the 0-50 percentile?

12.1.6 Inspect and run the script `ex12_1_6.py` to find association rules in the Wine dataset with supp $\geq$ 30% and conf $\geq$ 60%. Do these association rules make sense?

12.1.7 Often we are interested in rules with high confidence. Is it possible for itemsets to have very low support but still have a very high confidence?

12.1.8 (optional) Try find associations also in terms of the type of wine by adding two additional columns to the binary data corresponding to `1-y` and `y` (Note this is easiest done by adding new columns to the X-matrix and changing the `attributeNames` variable.)

1 Homework problems for this week

Problems

Question 1. Spring 2014 question 11: We consider the twelve costumers given in Table 3. We will consider this data set a market basket problem in which the twelve customers have various combinations of the six items denoted M_H, M_L, P_H, P_L, D_H, D_L. Which one of the proposed solutions below includes *all* the frequent itemsets with support of more than 40 %?

	M_H	M_L	P_H	P_L	D_H	D_L
LIS1	1	0	0	1	1	0
LIS2	1	0	1	0	1	0
LIS3	0	1	0	1	0	1
LIS4	0	1	0	1	0	1
LIS5	1	0	1	0	0	1
LIS6	1	0	1	0	1	0
OPO1	1	0	1	0	0	1
OPO2	0	1	0	1	1	0
OPO3	0	1	1	0	0	1
OPO4	0	1	0	1	0	1
OPO5	0	1	1	0	0	1
OPO6	1	0	1	0	1	0

Table 3: Given are the six first costumers of Lisbon and Oporto including whether these costumers spent more or less than the median consumption of MILK (M_H, M_L), PAPER (P_H, P_L), and DELI (D_H, D_L). Subscript H and L are thus used to respectively denote a relatively high and low level of consumption (i.e., above or below the median consumption).

A $\{M_L\}, \{M_H\}, \{P_H\}, \{P_L\}, \{D_H\}, \{D_L\}$.

B $\{M_L\}, \{M_H\}, \{P_H\}, \{P_L\}, \{D_H\}, \{D_L\}, \{M_H, P_H\}$.

C $\{M_L\}, \{M_H\}, \{P_H\}, \{P_L\}, \{D_H\}, \{D_L\}, \{M_H, P_H\}, \{M_L, D_L\}$.

D $\{M_L\}, \{M_H\}, \{P_H\}, \{P_L\}, \{D_H\}, \{D_L\}, \{M_H, P_H\}, \{M_H, D_H\}, \{M_L, P_L\}, \{M_L, D_L\}, \{P_L, D_L\}$.

E Don't know.

Question 2. Fall 2014 question 20: Consider the simple 1-dimensional data set comprised of $N = 7$ observations as shown in table 4. Suppose we wish to apply K-means clustering to the dataset and the $K = 3$ one-dimensional cluster centers are initialized in $\mu_1 = 4$, $\mu_2 = 7$ and $\mu_3 = 14$. After terminating of the K-means clustering algorithm, what are the final (rounded) cluster centers μ_1, μ_2, μ_3?

X	3	6	7	9	10	11	14

Table 4: Simple 1-dimensional dataset comprised of $N = 7$ observations.

A $\mu_1 = 3.00, \mu_2 = 8.00, \mu_3 = 12.50$

B $\mu_1 = 3.00, \mu_2 = 7.33, \mu_3 = 11.67$

C $\mu_1 = 4.50, \mu_2 = 9.25, \mu_3 = 14.00$

D $\mu_1 = 5.33, \mu_2 = 10.00, \mu_3 = 14.00$

E Don't know.

Question 3. Fall 2014 question 11: In table 5 is given the pairwise cityblock distances between 8 observations. A hierarchical clustering is used to cluster these nine observations using *group average* linkage. Which of the dendrograms shown in fig. 1 corresponds to the clustering?

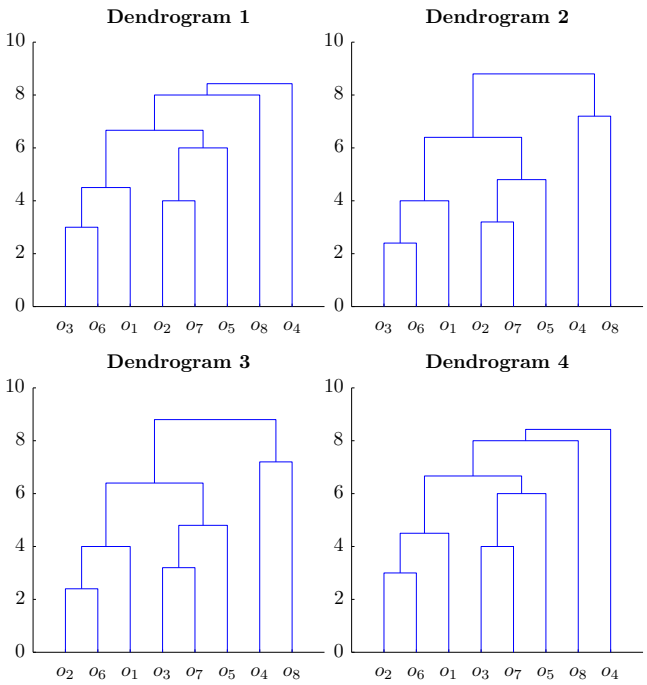

Figure 1: Hierarchical clustering of the 8 observations considereded in table 5

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
o_1	0	4	7	9	5	5	5	6
o_2	4	0	7	7	7	3	7	8
o_3	7	7	0	10	6	6	4	9
o_4	9	7	10	0	8	6	10	9
o_5	5	7	6	8	0	8	6	7
o_6	5	3	6	6	8	0	8	11
o_7	5	7	4	10	6	8	0	7
o_8	6	8	9	9	7	11	7	0

Table 5: Pairwise Cityblock distance, i.e $d(o_i, o_i) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1 = \sum_{k=1}^{M} |x_{ik} - x_{jk}|$, between 8 observations. Each observation o_i corresponds to a $M = 15$ dimensional binary vector, $x_{ik} \in \{0, 1\}$. The blue observations $\{o_1, o_2, o_3, o_4\}$ belong to class C_1 and the black observations $\{o_5, o_6, o_7, o_8\}$ belong to class C_2.

A Dendrogram 1.

B Dendrogram 2.

C Dendrogram 3.

D Dendrogram 4.

E Don't know.

References

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