Exercise Set 3

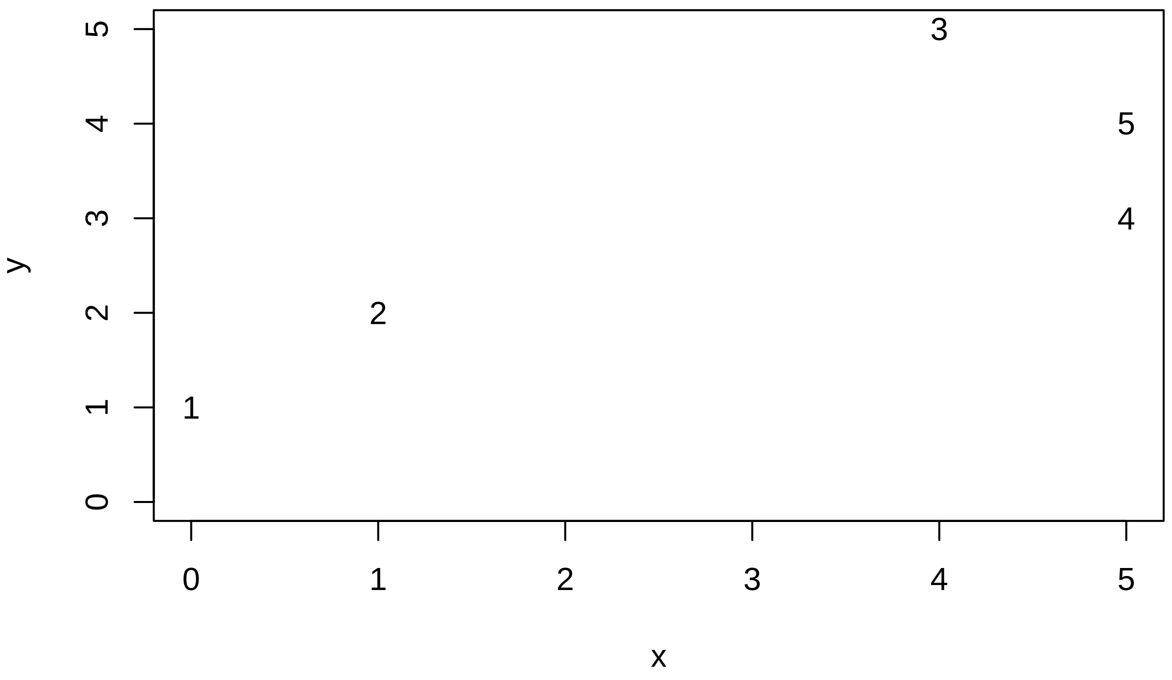
- Submit the answer via Moodle *before* 14 December 2021 at 23:59.
- You can answer anonymously or write your name to the answer sheet, at your choice.
- The assignment should be completed by one person, but discussions and joint solving sessions with others are encouraged. Your final solution must however be your own. You are not not allowed to copy ready-made solutions or solutions made by other students. You are allowed to use external sources, web searches included.
- The problems can be discussed in the 13 December exercise session.
- Your answer will be peer-reviewed by you and randomly selected other students.
- The language of the assignments is English.
- The submitted report should be in a single Portable Document Format (pdf) file.
- Answer to the problems in the correct order.
- Read the general instructions and grading criteria in Moodle before starting with the problems.
- Main source material: James et al., Chapter 12. Please feel to use other resources as well. "James et al." refers to: James, Witten, Hastie, Tibshirani, 2021. An Introduction to Statistical Learning with applications in R, 2nd edition. Springer.
- Notice that you can submit your answer to the Moodle well before deadline and revise it until the deadline. Therefore: please submit your answer well in advance, already after you have solved some problems! Due to peer review process we cannot grant extensions to the deadline. Even though the Moodle submission may occasionally remain open for a while after 23:59, the submission system will eventually close. If you try to submit your answers late you will not get any points (including peer-review points) from this Exercise Set. You have been warned.
- Please double-check that the submitted pdf is formatted properly and, e.g., contains all figures. It seems to be especially difficult to produce properly formatted pdf files with Jupyter Notebooks: remember to check the produced pdf or consider using R Markdown. .

Problem 16

[10 points]

Objectives: k-means loss and Lloyd's algorithm

```r
data <- data.frame(x=c(0,1,4,5,5),y=c(1,2,5,3,4))
```



You should be able to do this problem with a pen and paper.

Task a

For what kinds of tasks can we use Lloyd's (k-means) algorithm? Explain what the *inputs* and *outputs* of the algorithm are. How to interpret the results?

Task b

Define the objective (or cost) function that Lloyd's algorithm tries to minimize. What can be said about the value of the objective function during the two stages of each iteration of Lloyd's algorithm?

Task c

Consider the following set of data points in $\mathbb{R}^2$: $x_1 = (0, 1)$, $x_2 = (1, 2)$, $x_3 = (4, 5)$, $x_4 = (5, 3)$, $x_5 = (5, 4)$. Sketch the run of the Lloyd's algorithm using $K = 2$ and initial prototype (mean) vectors $\mu_1 = (0, 2)$ and $\mu_2 = (2, 0)$. Draw the data points, cluster prototype vectors, and cluster boundary after each iteration until convergence.

Also, write down the calculation procedure and the cluster memberships as well as prototype vectors after each iteration.

Problem 17

[10 points]

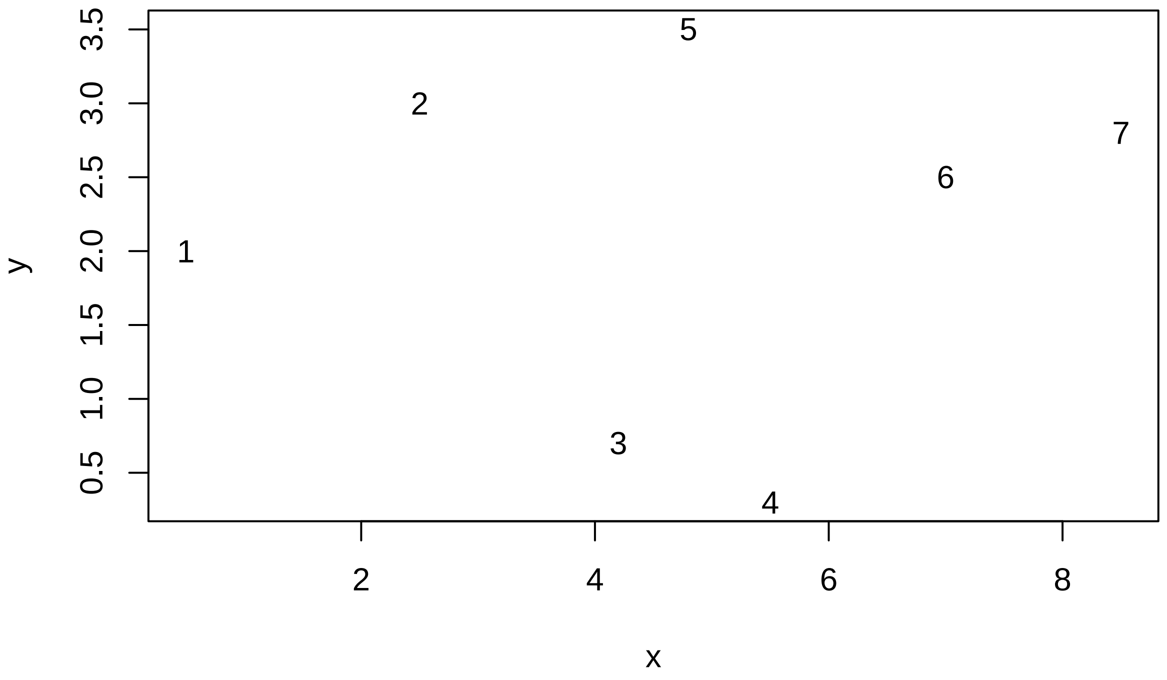
Objectives: understanding hierarchical clustering algorithms

We consider *hierarchical clustering* on a toy data set $D = \{(x_i, y_i)\}_{i=1}^{7}$, where $(x_i, y_i) \in \mathbb{R}^2$. You should be able to do this problem with a pen and paper.

The data points are shown in the table and figure below.

```
data <- data.frame(x=c(0.5,2.5,4.2,5.5,4.8,7.0,8.5),
                   y=c(2.0,3.0,0.7,0.3,3.5,2.5,2.8))
```

	x	y
1	0.5	2.0
2	2.5	3.0
3	4.2	0.7
4	5.5	0.3
5	4.8	3.5
6	7.0	2.5
7	8.5	2.8



The matrix of Euclidean distances between the data points is given below.

	1	2	3	4	5	6	7
1	0.00	2.24	3.92	5.28	4.55	6.52	8.04
2	2.24	0.00	2.86	4.04	2.35	4.53	6.00
3	3.92	2.86	0.00	1.36	2.86	3.33	4.79
4	5.28	4.04	1.36	0.00	3.28	2.66	3.91
5	4.55	2.35	2.86	3.28	0.00	2.42	3.77
6	6.52	4.53	3.33	2.66	2.42	0.00	1.53
7	8.04	6.00	4.79	3.91	3.77	1.53	0.00

Task a

Sketch a run of the basic agglomerative hierarchical clustering algorithm to this data using the single link (min) notion of dissimilarity between clusters. Visualise the result as a dendrogram. Draw a sketch of the above figure in your answer sheet and indicate in this how the algorithm forms clusters step-by-step. It is not necessary to show detailed calculations, but indicate the cost of the join that you select.

Task b

Repeat the task a, but using complete link (max) dissimilarity. Compare the results.

Problem 18

[9 points]

Objectives: practical application of k-means and hierarchical clustering Note the hint at the end of the problem.

For this problem you can use the term project dataset in file `npf_train.csv`. Your task is to cluster the rows of the data matrix, where the rows are given by the days and columns by various observations during that day. In this task you need only the class variable (`class4`) and the real valued mean measurements (variable names ending with `.mean`). You should end up with a dataset of 50 real valued variables and one class variable as columns. The R code to produce the data is given below for reference (in the last line we strip out the redundant `.mean` from the variable names!).

```r
npf <- read.csv("npf_train.csv")
## choose variables whose names end with ".mean" together with "class4"
vars <- colnames(npf)[sapply(colnames(npf),
                             function(s) nchar(s)>5 && substr(s,nchar(s)-4,nchar(s))==".mean")]
npf <- npf[,c(vars,"class4")]
## strip the trailing ".mean" to make the variable names prettier
colnames(npf)[1:length(vars)] <- sapply(colnames(npf)[1:length(vars)],
                                        function(s) substr(s,1,nchar(s)-5))
vars <- colnames(npf)[1:length(vars)]
```

Unless otherwise instructed, please scale the variables to zero mean and unit variance.

Task a

Cluster the rows of the data matrix and plot the k-means loss as a function of the number of clusters from 1 to 20. Should you normalise the columns? What effect does the normalization of the columns have? You can use a library function such as `kmeans` in R or `KMeans` in sklearn.

Task b

By using Lloyd's algorithm and scaled variables, cluster the rows into four clusters and make a confusion matrix (contingency table where the rows are the true classes, columns the cluster indices, and entries the counts of rows). Order the rows and columns so that the sum of diagonal entries in your contingency table is maximized. Where are most "errors" made, if you would use your clusters as a rudimentary classifier (i.e., if you would associate each cluster with a class)? (See the hint below!)

Task c

Repeat the clustering of task b above with 1000 different random initialisations (for example, picking at random 4 data vectors for your initial centroids in Lloyd's algorithm). Make a histogram of the losses. What is the minimum and maximum k-means losses for your 1000 random initialisations? How many initialisations would you expect to need to have to obtain one reasonably good loss (say, a solution with a loss within 1% of the best loss out of your 1000 losses), for this dataset and number of clusters? If you use `KMeans` from `sklearn.cluster` kmeans++ is the default initialisation; use here, e.g., `init="random"`.

Then try how using some smart initialisation such as kmeans++ affects your results.

Task d

Try clustering the same data with agglomerative hierarchical clustering with at least 2 different linkage functions. Produce a dendrogram and corresponding flat clustering (e.g., by splitting the dendrogram with `cutree`) and compare their properties (e.g., comparing sizes of clusters, by looking at confusion matrices). Find and report at least one interesting feature or reproduce some of the properties of hierarchical clustering (e.g., differences between the linkage functions) discussed in the lecture. (Hint: See Sect. 12.5.3 of James et al. for examples in R.)

Instructions and hints

You can use, e.g., `kmeans` to run Lloyd's algorithm. Please use Lloyd's algorithm, e.g., as follows (the code below uses Lloyd's algorithm with one random initialisation on scaled dataset).

```
cl <- kmeans(scale(npf[,vars]),
             centers=4,algorithm="Lloyd",nstart=1,iter.max=100)
```

In task b you are asked to combine the class variables and the cluster indices and in task d you are asked to compare clusterings. Because all permutations of cluster indices are equally good it is useful to be able to, e.g., find the best match between the know classes (`class4`) and the cluster indices. One way to do this is the Hungarian algorithm which you can use to find a permutation of cluster indices such that the sum of entries in the diagonal of the confusion matrix is maximised. An example in R is given below, where the implementation `solve_LSAP` of Hungarian algorithm in the `clue` library is used. In Scipy, the Hungarian algorithm is provided, e.g., by `linear_sum_assignment` from `scipy.optimize`.

```
library(clue)
## Create confusion matrix between the known classes (class 4) and
## cluster indices.
tt <- table(npf$class4,cl$cluster)
## Find a permutation of cluster indices such that they
## best match the classes in class4.
tt <- tt[,solve_LSAP(tt,maximum=TRUE)]
```

If you use R, you can use the following function to use the kmeans++ algorithm to find the initial cluster centroids. You can give the resulting centroids directly as a parameter to `kmeans` function (parameter `centers`). If you use SciPy, `KMeans` from `sklearn.cluster` kmeans++ is the default initialisation.

```
# kmeansppcenters - Finds initial kmeans++ centroids
# Arguments:
# x         numeric matrix, rows correspond to data items
# k         integer, number of clusters
# Value:
# centers   a matrix of cluster centroids, can be fed to kmeans
#
# Reference: Arthur & Vassilivitskii (2007) k-means++: the
# advantages of careful seeding. In Proc SODA '07, 1027-1035.
kmeansppcenters <- function(x,k) {
  x <- as.matrix(x)
  n <- nrow(x)
  centers <- matrix(NA,k,ncol(x))
  p <- rep(1/n,n)
  for(i in 1:k) {
    centers[i,] <- x[sample.int(n,size=1,prob=p),]
    dd <- rowSums((x-(rep(1,n) %o% centers[i,]))^2)
    d <- if(i>1) pmin(d,dd) else dd
    if(max(d)>0) { p <- d/sum(d) }
  }
  centers
}

cl <- kmeans(scale(npf[,vars]),centers=kmeansppcenters(scale(npf[,vars]),4),
             algorithm="Lloyd",iter.max=100)
```

Problem 19

[9 points]

Objectives: uses of PCA

Continue with the same dataset as in the problem above.

Task a

Make a PCA projection of the data into two dimensions. Indicate the class index (`class4`), e.g., by color and/or the shape of the glyph. Be sure to indicate which color/shape corresponds to which class, e.g., by `legend`.

Task b

Compute and plot the proportion of variance explained and the cumulative variances explained for the principal components. Study the effects of different normalisations - at least compare the difference if you do not scale the data at all vs. if you normalize each variable to zero mean and unit variance! Why does it seem for unnormalized data that fewer components explain a large proportion of the variance, as compared to the normalized data? (Hint: See Sect. 12.2.4 of James et al.)

Task c

Pick one classification algorithm that is implemented in R or SciPy or in your other favourite environment that would work with this data and choose one of the challenge performance measures (binary accuracy, multiclass accuracy, or perplexity). Split the data at random into training and validation sets of equal sizes. Train your classification algorithm first without the dimensionality reduction on the training set and report the performance (=your chosen performance measure) on the validation set. Do the same on the data where the dimensionality has been reduced by PCA (see task b above). How does the performance of your classifier vary with the (reduced) dimensionality and is there an "optimal" dimensionality which gives you the best performance on the validation set?

Hint: Notice that you can do the PCA on the combined training and validation sets. This is a simple form of semi-supervised learning: this way you can utilise the structure of the validation/test set even if you don't know the class labels on the validation/test set!

Task d (optional)

This is a bonus tasks for which no points are awarded.

Repeat task a above, but this time with isometric multidimensional scaling (MDS) and t-distributed stochastic neighbor embedding (t-SNE).

If you use R, you can use `isoMDS` from library `MASS` and `tsne` from library `tsne`. Notice that neither MDS nor t-SNE algorithms are guaranteed to converge to a local optimum. Therefore, a good initial position - one good choice is the PCA solution - is essential.

```r
library(MASS)
library(tsne)
d <- dist(scale(npf[,vars]))
## cmdscale essentially does PCA. Both MDS and t-SNE are sensitive to initial
## configuration and the PCA initialisation generally leads to reasonable convergence.
## (Even though R isoMDS and tsne can handle bad initial config fine,
## we are being explicit here.)
y <- cmdscale(d,2)
## isometric MDS embedding coordinates
x.mds <- isoMDS(d,y=y)$points
```

```r
## t-SNE embedding coordinates
x.tsne <- tsne(d,initial_config=y,k=2)
```

Problem 20

[2 points]

Objectives: self-reflection, giving feedback of the course

Tasks

- Write a learning diary of the topics of lectures 9-12 and this exercise set.

Instructions

The length of your reply should be 1-3 paragraphs of text. Guiding questions: What did I learn? What did I not understand? Was there something relevant for other studies or (future) work?

Machine Learning Guest Lectures

10 December 2021 at 10:15-12 via the regular Zoom link.

Andreas Henelius: Data Science in Finance - Machine learning for overdue invoice prediction

Abstract: Invoice financing is a concept in banking enabling short-term borrowing by the bank's customers against the amounts due from the invoices held by the customer. Such borrowing can help the customer, e.g., with cash flow management. For this process to be feasible, it must be automated, and the bank needs to estimate the net present value of invoices, which depends on the expected payback time. In this presentation I talk about how to leverage machine learning to predict if invoices will be overdue.

About the speaker: D.Sc. (Tech.) Andreas Henelius is a data scientist at the OP Financial Group. Previously he worked at the University of Helsinki, Aalto University and the Finnish Institute of Occupational Health.

Antti Ukkonen: From voice to meaning: Machine learning for spoken language understanding

Abstract: I will discuss some of the many technical challenges we face at Speechly (www.speechly.com) when building a SaaS -solution using which developers with no prior ML or speech recognition experience can integrate voice functionality to their mobile or web applications. I will talk about deep learning, transfer learning, neural language models, perhaps a bit about reinforcement learning, and how to put these together in an autonomous cloud-based model training infrastructure.

About the speaker: Antti works currently as the Chief Product Officer at Speechly. In this role he is mainly heading the day-to-day R&D and product design efforts on the basis of customer feedback, but occasionally he also hacks away at finite state transducers or tries to make a big deep learning model run in real-time on a Raspberry Pi. Prior to joining Speechly in summer 2019, Antti spent almost 15 years in industry and academia as a data mining researcher. He has held positions at Yahoo! Research, Helsinki Institute for Information Technology HIIT, Finnish Institute for Occupational Health. Until December 2019, he was an Academy Research Fellow at University of Helsinki. Antti got his doctoral degree from Aalto University in 2008.

Exercise Set 3 – schedule

In this document, you can find a suggested schedule to do the Exercise Set 3 (E3). The purpose of this document is to help you plan your studies.

It is assumed that you follow the lectures and that you read the corresponding textbook sections before each of the lectures.

It is not possible to grant extensions to the E3 answer deadline. If you do not submit your answer in time, you will lose not only the answer points but also the peer review points. Therefore, please submit your solutions after you have already solved some problems. It is possible to update your submission in Moodle until the deadline. This way you will not lose all points due to some last-minute internet or pdf conversion problems

If there are tasks that are difficult and take lots of time, I suggest that you first solve the easier problems and only then attempt to do the more time-consuming ones. You can ask for advice in Slack or in the exercise sessions. Also remember that you can get full exercise points even if you do not solve all problems or tasks, so it is not necessary to have perfect solutions to all tasks. Often a non-perfect solution is better than no solution at all.

Wednesday 1 December. Lecture on clustering.

After 1 December and before 3 December lecture: do **Problems 16-18** on clustering.

Friday 3 December. Lecture on dimensionality reduction.

After 3 December and before 8 December lecture: do **Problem 19** on dimensionality reduction.

(You can attend the 7 December exercise session which is mainly about peer review of E2, but you can also ask of E3.)

Wednesday 8 December. Recap lecture.

(During this week you can focus on the term project.)

Friday 10 December. ML guest lectures.

Monday 13 December. Exercise session 5 (solving E3). Participate to the exercise session, where you can discuss any remaining unclear problems and/or the topics of the Block 3 in general. Do Problem 20 (learning diary) and submit your solutions to Moodle.

Wednesday December – Sunday 19 December. Try to grade the problems at Moodle.

Wednesday 15 December. Participate to the exercise session, ask if anything in the solutions of E3 is unclear for you, finish the grading, and submit your gradings to Moodle at the end of the exercise session.