Measures of similarity, summary statistics and probabilities with R

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.

R Help: R help for a function is obtained by typing `?<function name>` in the R command prompt. In practice, the fastest and easiest way to get help in R is often to simply Google your problem. For instance: `"How to add legends to a plot in R"` 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.

Piazza discussion forum: You can get help by asking questions on Piazza:
https://piazza.com/dtu.dk/fall2020/02450

Software installation: Extract the R toolbox from DTU Inside. Start R and go to the `<base-dir>/02450Toolbox_R/` directory by setting the working directory through `setwd(<base-dir>/02450Toolbox_R/)` and run `source('setup.R')`. 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_R/Scripts/`
Representation of data in R:

	R var.	Type	Size	Description
	`X`	Numeric	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	Cell array	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	Numeric	Scalar	Number of data objects.
	`M`	Numeric	Scalar	Number of attributes.
Classification	`y`	Numeric	$N \times 1$	Class index: For each data object, `y` contains a class index, $\mathtt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	Cell array	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	Numeric	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.R`. We will use the package "tm" [?, ?], which can be installed in R by typing `install.packages("tm")`. Make sure that your R version is new, as the package "tm" needs an R version at least 2.14.0. Use the package to generate a document-term matrix and to convert it into the format described in the beginning of the exercise (Representation of data in R). You will also need the package `Snowball` to make the option `lang` in tm work. This option indicates the language of the analyzed text. Install it using install.packages("Snowball").

Script details:

- *Refer to the manual [?] to get help with using the package.*
- *The text documents are stored in the files* Data/textDocs1.txt, Data/textDocs2.txt, *. . ., Data/textDocs5.txt in the directory "Data".*
- *The command Corpus constructs a collection of documents given a source of documents.*
- *The function DirSource creates a source of documents for use in Corpus. The function DirSource takes as its first argument a folder. Another argument, "patterns", can be passed to specify the pattern that the filenames we want in the source folder must match.*
- *Punctuation is removed from documents through the function tm_map by passing the corpus of documents as the first argument, and the function name "removePunctuation" as the second argument. The output is the corpus of documents given as input, but without punctuation.*
- *The function DocumentTermMatrix takes a corpus as input, and generates a document-term matrix.*

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.R

Script details:

- *Type ?tm_map to learn how to apply a function to all documents in a corpus.*
- *The function removeWords removes the words given in the following argument.*
- *To remove the stopwords in the vector "mystopwords", use tm_map(docs, removeWords, mystopwords).*

- *A list of stop words is stored in the file* `Data/stopWords.txt`*.*
- *To read the text file* `stopWords.txt` *into the vector "mystopwords", use* `mystopwords=scan("./Data/stopWords.txt", character(0))`*.*
- *After removing stopwords, make a new document-term matrix, based on the documents with removed stopwords. To ensure that stopwords are not included in the generation of the document term matrix, pass the argument* `control=list(stopwords=TRUE)` *to* `DocumentTermMatrix`*.*

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:

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

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.R`.

Script details:

- *The function* `tm_map` *is also used to stem documents in a corpus. When used for stemming, the function name to be passed as the second argument is "stemDocument".*
- *A third argument, "language" may be passed. However, the default for this argument is English (language="english"). Since we are using English documents, it is not necessary to pass this argument, but may be done to enhance code readability.*

Inspect the document-term matrix: How does it compare to your original matrix? Can you get to the same result as shown below?

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, $\boldsymbol{X}$:

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

$$\boldsymbol{q} = \left[\begin{array}{ccccccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right]$$

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.R`, and show that Document 4 is most similar to the query.

Script details:

- *The function* `cosine`*, which computes the cosine measure, is included in the package "lsa".*
- *You can extract a document (row of the* `X` *matrix) using the command* `inspect(X)[i,]` *where* `i` *is the index of the document.*

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

3.2 Summary Statistics

3.2.1 Consult the script `ex3_2_1.R`. 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`, `sd`, `median`, *and* `range`

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:

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

where $\mathrm{cov}(\boldsymbol{x},\boldsymbol{y})$ denotes the covariance between $\boldsymbol{x}$ and $\boldsymbol{y}$ and $\mathrm{std}(\boldsymbol{x})$ 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

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

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.R`. 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.R`

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

Script details:

- *Open* `similarity.R` *to learn about the R 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 function cor.R. Specifically address the questions:

- Include basic summary statistics of the attributes.

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.