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://piazza.com/dtu.dk/spring2023/02450

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.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`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.

google
matrix
model
internet
nonzero
link
webpage
rank
eigenvalue
problem
england
fifa
done
solving

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, $\boldsymbol{X}$:

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

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

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`

The script computes the difference LHS-RHS. If the value is 0, then the expressions are equivalent.

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

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*?

K=6

	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.