Homework 1: Distance functions on vectors

Juliane Klatt
juliane.klatt@bsse.ethz.ch

Giulia Muzio
giulia.muzio@bsse.ethz.ch

Bastian Rieck
bastian.rieck@bsse.ethz.ch

Prof. Dr. Karsten Borgwardt
karsten.borgwardt@bsse.ethz.ch

Submission deadline: 14.10.2020 at 08:00

Objectives

The goals of this homework are:

1. To implement different similarity measures on vectors.
2. To apply similarity measures in the context of text mining on a standard machine learning dataset.
3. To explore theoretical aspects of similarity measures on vectors.

Problem overview

You will implement some of the similarity measures on vectors that were discussed in the lecture using `Python`. Your program will accept a series of command line arguments that will specify all execution parameters. It will receive data files as input (tab-separated text files) and will create output files with the results. Additionally, you will be asked to address some theoretical questions about similarity measures.

Dataset

You will work on a dataset of text documents, known as '20 Newsgroups Data', that has been extensively used in the data mining community [1, 2]. Each document is stored as a plain-text file and is categorised as belonging to one of twenty different newsgroups. In some cases, newsgroups have topics that are very related to each other, e.g. the groups `comp.graphics` and `comp.sys.mac.hardware` cover topics on computer graphics and hardware for Mac computers respectively. Other groups differ greatly in their contents, e.g. `rec.autos` and `talk.religion.misc` which discuss general concepts about cars and miscellaneous topics about religion, respectively.

Letting $\mathcal{U}$ denote the universe of words that occur in all documents. Each document D is represented as a vector $\mathbf{x} \in \mathbb{R}^{|\mathcal{U}|}$. The value of the ith element in $\mathbf{x}$ is the *term frequency-inverse document frequency* (tf-idf) of the ith word in $\mathcal{U}$ and $\mathbf{x}_i = 0$ if D does not contain that word. In general, the tf-idf of a word—or term—in a document is computed by two different measures:

- **tf** (term frequency) is the frequency of the word in the document. This value is higher if the word occurs many times in a document.
- **idf** (inverse document frequency) indicates how *rare* the word is across all documents in the dataset. This value is lower if the word occurs in most of the documents, thus giving the word little discriminating power.

In essence, tf-idf measures the importance of a word within a document with respect to the entire dataset. The input to your program consists of vectors $\mathbf{x}$ with tf-idf values, saved as text files, with one $\mathbf{x}$ per document D in the dataset. You will work on a subset of the '20 Newsgroups Data' that consists of 5 groups with 10 documents each. The groups are:

`comp.graphics`
`rec.autos`
`comp.sys.mac.hardware`
`talk.politics.guns`
`talk.religion.misc`

Table 1: Subset of the groups used in this assignment.

The documents in each group were randomly selected from the original data. The file `20news-bydate-train-subset.zip` contains all the input data for this assignment. It includes the following items:

1. The file `file_info.txt` that lists the documents and the groups to which they belong. Table 2 shows the layout of the file.
2. The original documents, e.g. the text file `54529`
3. The vectorised form of the documents, e.g. the text file `54529_vec.txt`

The original documents are only included for your reference. They are *not* required to solve this assignment. The main program processes the `*_vec.txt` files according to what is indicated in `file_info.txt`.

`file_name`	`group`
`X99`	`id`

Table 2: Format of the file `file_info.txt` file. Columns are tab-separated and `id` in the second column corresponds to a group name in Table 1.

Exercise 1

Exercise 1.a

Your main goal is to determine the distance of documents within the same newsgroup and between different newsgroups. For this purpose, you will extend the script `distance_fn.py` with additional implementations of the following distance functions:

- `manhattan_dist(v1, v2)`
- `hamming_dist(v1, v2)`
- `euclidean_dist(v1, v2)`
- `chebyshev_dist(v1, v2)`
- `minkowski_dist(v1, v2, d)`

Please note:

- The names of the functions and their parameters should match the above description. Do *not* change the provided skeleton script.
- The return value of all functions should be a single `float`.
- For the function `hamming_dist(v1, v2)`, you will need to binarise the vectors `v1` and `v2` inside the function. Any value greater than 0 should be set to 1. This is equivalent to saying that for document $\mathbf{x}$, if $\mathbf{x}_i = 1$ then the document contains *at least one* instance of the ith word in $\mathcal{U}$.
- For this homework, **do not use functions from external libraries that implement these distance functions**. For instance, `sklearn.metrics.pairwise_distances` implements all of these metrics, but using this function to solve the exercise is *not* permitted. The goal is for you to gain an understanding in implementing these functions for yourself.

Then, use the script `compute_distances.py` that is provided in the assignment. This is the main program and it invokes the functions that you have to create in `distance_fn.py`. In Python, as shown in line 29 of `compute_distances.py`, you can invoke the functions from a different script by including the following line:

```
1 import distance_fn as d
```

Later on, you can call the function as `d.manhattan_dist()`. In total, the main script performs two tasks:

1. For every pair of groups G_1 and G_2, it computes the average distance between all $v_1 \in G_1$ and all $v_2 \in G_2$. Because of symmetry of the distance functions, there is no need to compare G_1 vs. G_2 and G_2 vs. G_1
2. It iterates through all pairs of groups and all distance functions to create an output file as the one shown in Table 3. The file is named `output_distances.txt`. In this file, the last two columns correspond to calls to `minkowski_dist(v1, v2, d)` with `d=3` and `d=4`. All columns are tab-separated.

In this homework, the main program is given to you to ease your transition into Python. In this way, you can focus on the manipulation of the arrays to compute the distances and worry less about the input/output of data to the program. You are responsible

for extending `distance_fn.py` and of running `compute_distances.py` to generate the results.

Pair of newsgroups	Manhattan	Hamming	Euclidean	Chebyshev	Minkowski d=3	Minkowski d=4
comp.graphics:comp.graphics	99.99	99.99	99.99	99.99	99.99	99.99
comp.graphics:rec.autos	99.99	99.99	99.99	99.99	99.99	99.99
	⋮					⋮

Table 3: Format of output file `output_distances.txt` file. Columns are tab-separated. Values above are centred within columns just for illustration purposes.

Task: Write the script `distance_fn.py` and run `compute_distances.py` to generate the results, as described above.

Exercise 1.b

Report and discuss any abnormalities in the results. For example, do all distance functions report a lower average distance when comparing documents of the same group vs. documents from different groups?

Exercise 1.c

Which metric seems to provide, on average, the best separation between groups? Explain why this is the case.

Exercise 1.d

Another similarity similarity measure is also often suggested in the literature, viz.

$$s(\mathbf{x}, \mathbf{y}) := \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|},$$

where $\mathbf{x} \cdot \mathbf{y}$ refers to the dot product of the two vectors, and $\| \cdot \|$ denotes the usual Euclidean distance. Given that $\mathbf{x}$ and $\mathbf{y}$ are tf-idf vectors, what can you say about the range, i.e. the possible values, that s can attain? How does this change when $\mathbf{x}$ and $\mathbf{y}$ are arbitrary vectors? What happens (both in case $\mathbf{x}$ and $\mathbf{y}$ are tf-idf vectors, and for arbitrary vectors) if the dimensionality is increased?

Hint: Use the fact that the dot product between two vectors can also be defined using their *angle*.

Exercise 1.e

The Manhattan and Euclidean distances are also known as L_1 and L_2 norms, respectively. In general, what behaviour can we expect about the L_1 vs. L_2 norms as the dimensionality (i.e. the number of attributes) in the data increases? Is this behaviour observed in our dataset? If not, why not?

Exercise 2

Based on the conditions that must be met for a function to be a metric, answer the following questions:

Exercise 2.a

Which of the functions listed below are not metrics? Indicate what condition is not satisfied, if any. It is sufficient to provide a simple counterexample to prove that something is *not* a metric. By contrast, you do not have to prove that something *is* a metric, you merely have to state it.

i)	$x, y \in \mathbb{R}^n$,	$d(x,y) = \sum_{i=1}^{n} (x_i - y_i)^2$
ii)	$x, y \in \mathbb{R}^n$,	$d(x,y) = \sum_{i=1}^{n} x_i y_i (x_i - y_i)^2$
iii)	$x, y \in \mathbb{R}^n$,	$d(x,y) = \sum_{i=1}^{n} w_i \lvert x_i - y_i \rvert,\ w_i > 0\ \forall i$
iv)	$x, y \in \{z \in \mathbb{R}^n \mid \sum_{i=1}^{n} z_i = 1, z_i > 0\ \forall i\}$,	$d(x,y) = \sum_{i=1}^{n} x_i \log\left(\frac{x_i}{y_i}\right)$
v)	$x, y \in \mathbb{R}^n$,	$d(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$

Exercise 2.b

For the Minkowski distance, show that:

i)	$a \in \mathbb{R}$ and $x, y \in \mathbb{R}^n$,	$d(ax, ay) = \lvert a \rvert d(x,y)$
ii)	$x, y, z \in \mathbb{R}^n$,	$d(x+z, y+z) = d(x,y)$

Exercise 2.c

Property i) in Exercise 2.b is called *homogeneity*. Determine if homogeneity applies to function v) of Exercise 2.a.

Exercise 2.d

Property ii) in Exercise 2.b is called *translation invariance*. Determine if translation invariance applies to the following function:

$$x, y \in \mathbb{R}^n_+ : \quad d(x,y) = \frac{2}{\pi} \operatorname{acos}\left(\frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}\right)$$

Hint: What can you say about the angle between two points?

Command-line arguments

Your program will receive 2 (a third one is optional) command line arguments:

`--datadir path`: is the path to the directory where the input files are stored. The documents will be stored in vectorial form as described in Section: Dataset.

`--outdir path`: the path to the directory where the output file will be stored.

After unpacking the file `20news-bydate-train-subset.zip` in the directory in which you create your solution, a sample invocation for Exercise 1 is:

```
> python compute_distances.py --datadir 20news-bydate-train-subset \
                              --outdir  output
```

This will create the required output file in the directory `output`.

Grading and submission guidelines

This homework is worth a total of 100 points. Table 4 shows the points assigned to each exercise/question.

Table 4: Grading key for homework 1

50 pts.	**Exercise 1**	
	20 pts.	Exercise 1.a
	5 pts.	Exercise 1.b
	5 pts.	Exercise 1.c
	10 pts.	Exercise 1.d
	10 pts.	Exercise 1.e
50 pts.	**Exercise 2**	
	20 pts.	Exercise 2.a
	10 pts.	Exercise 2.b
	10 pts.	Exercise 2.c
	10 pts.	Exercise 2.d

Follow the submission guidelines posted on the Moodle webpage. Refer to the document titled 'General guidelines for homework sheets'.

References

[1] T. Joachims. A probabilistic analysis of the Rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3.

[2] K. Lang. NewsWeeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.