Lab 7: Cosine and Euclidean Distance

This is an INDIVIDUAL assignment. Due date is as indicated on BeachBoard. Follow ALL instructions otherwise you will lose points. In this lab, you will be classifying data using cosine distance, Euclidean distance, and Manhattan distance.

Required Knowledge:

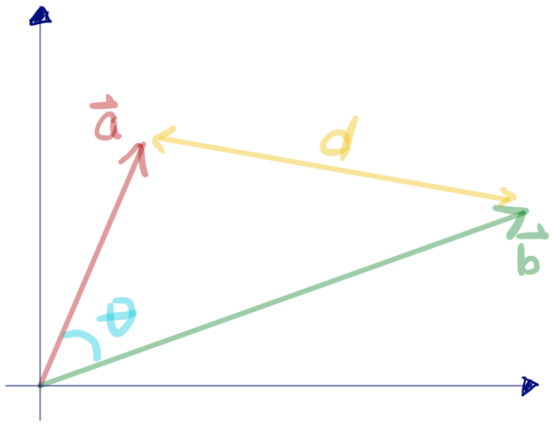
You have already learned about the cosine formula:

$$cos\theta = \frac{a \cdot b}{|a||b|}$$

You can find the angle created by two vectors by using this formula.

You have also learned how to find the distance between two vectors:

$$d = \sqrt{(a-b)\cdot(a-b)}$$



Background:

The idea is to classify data based on these distance metrics. For example, if you have two vectors that have a relatively small angle between them, they could be similar.

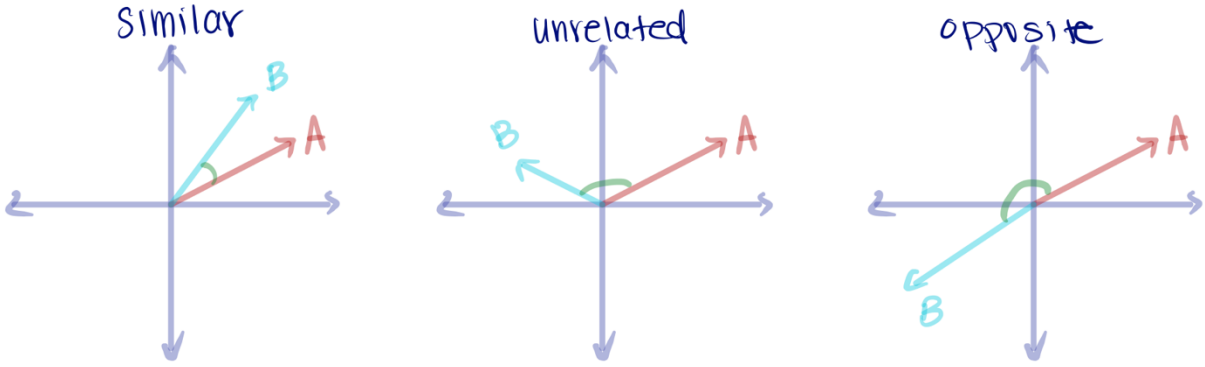


Figure 1: using cosine similarity

Another example: two vectors can be measured using traditional distance to see how far apart the two vectors are. If they are far apart (shown in green), then they could be unrelated. If the vectors are close, then they are more likely to be similar.

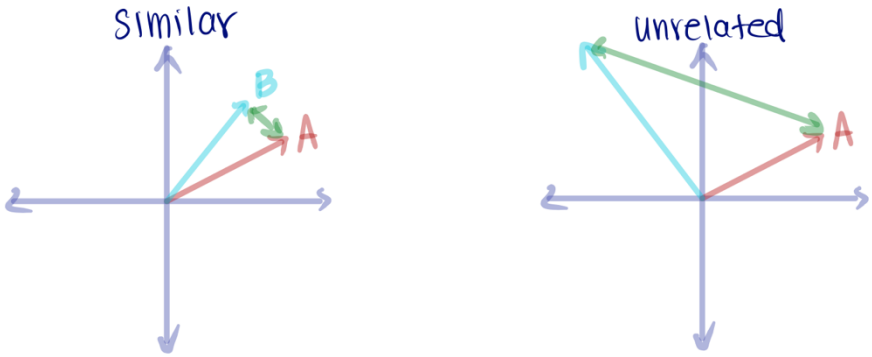


Figure 2: using Euclidean distance metric

Another distance metric is the Manhattan distance. This is a little different from Euclidean distance because it is not a straight-line distance. Manhattan distance is calculated by finding the difference in each dimension and adding it all together. The larger the Manhattan distance is, the less likely it is that the two vectors are similar. Similarly, the smaller the Manhattan distance is, the more likely it is that the two vectors are similar.

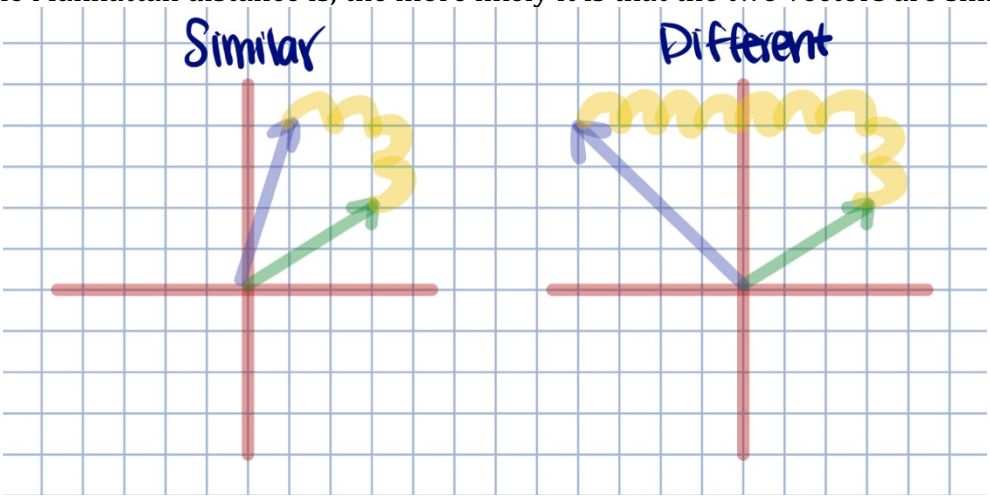


Figure 3: using Manhattan distance metric

Walkthrough:

You are given a csv file of the famous iris dataset. I have also attached a copy of the dataset on BeachBoard. Please note that there is no header in the csv file. Based on this data, you are going to use the three metrics (cosine similarity, Euclidean similarity, and Manhattan similarity) to predict which iris type a provided test vector should be classified as.
To do this, you need an anchor or a reference vector that will represent each class. You will need to find the average vector of all vectors in the dataset for each iris class. These averages will represent your "center" for all vectors. The center that has the smallest distance metric from the test vector will be the best match when classifying the iris.

You will need to find the average of all iris-setosa rows, all iris-versicolor rows, and all iris-virginica rows. This will need to be returned as a dataframe. See image below.

	0	1	2	3	4
0	5.936	2.770	4.260	1.326	Iris-versicolor
1	5.006	3.418	1.464	0.244	Iris-setosa
2	6.588	2.974	5.552	2.026	Iris-virginica

Now that you have this, you can try to classify your data using various data metrics. Let's look at the first test vector:

`test_vec1 = [5.1,3.4,1.5,0.2]`

The goal is to correctly classify what kind of iris this is. Is this an iris-versicolor, an iris-setosa, or an iris-virginica.

The first similarity metric is **cosine similarity**:

$$cos\theta = \frac{a \cdot b}{|a||b|}$$

We can find the angle formed between `test_vec1` and each center-vector. The center-vector that forms the smallest angle is the most likely iris class.

Iris-versicolor	Iris-setosa	Iris-virginica
b = [5.936,2.770,4.260, 1.326] $cos\theta = \frac{test\_vec1 \cdot b}{\lvert test\_vec1\rvert\lvert b\rvert}$ $\theta = 22.145°$	b = [5.006,3.418,1.464, 0.244] $cos\theta = \frac{test\_vec1 \cdot b}{\lvert test\_vec1\rvert\lvert b\rvert}$ $\theta = 0.768°$	b = [6.588,2.974,5.552, 2.026] $cos\theta = \frac{test\_vec1 \cdot b}{\lvert test\_vec1\rvert\lvert b\rvert}$ $\theta = 27.169°$

Since the iris-setosa center creates the smallest angle, we can assume that `test_vec1` is an iris-setosa

The second similarity metric is **Euclidean distance similarity**:

$$d = \sqrt{(a-b)\cdot(a-b)}$$

We can find the Euclidean distance between `test_vec1` and each center-vector. The center-vector that is the closest to `test_vec1` most likely iris class.

Iris-versicolor	Iris-setosa	Iris-virginica
b = [5.936,2.770,4.260, 1.326] $d = \sqrt{(test\_vec1-b)\cdot(test\_vec1-b)}$ $d = 3.159$	b = [5.006,3.418,1.464, 0.244] $d = \sqrt{(test\_vec1-b)\cdot(test\_vec1-b)}$ $d = 0.111$	b = [6.588,2.974,5.552, 2.026] $d = \sqrt{(test\_vec1-b)\cdot(test\_vec1-b)}$ $d = 4.706$

Since the iris-setosa center is the closest to `test_vec1` using Euclidean distance, we can assume that `test_vec1` is an iris-setosa.

The third similarity metric is **Manhattan distance similarity**:

$$d = \sum_{i}^{n} |a[i] - b[i]|$$

We can find the Euclidean distance between `test_vec1` and each center-vector. The center-vector that is the closest to `test_vec1` most likely iris class.

Iris-versicolor	Iris-setosa	Iris-virginica
`b = [5.936,2.770,4.260, 1.326]` $d = \sum_{i}^{4} \lvert test\_vec1[i] - b[i] \rvert$ $d = 5.352$	`b = [5.006,3.418,1.464, 0.244]` $d = \sum_{i}^{4} \lvert test\_vec1[i] - b[i] \rvert$ $d = 0.192$	`b = [6.588,2.974,5.552, 2.026]` $d = \sum_{i}^{4} \lvert test\_vec1[i] - b[i] \rvert$ $d = 7.792$

Since the iris-setosa center is the closest to `test_vec1` using Manhattan distance, we can assume that `test_vec1` is an iris-setosa.

Task:

1. Take a close look at the `metric_similarity.py` file. There are four functions that you need to fill in: `find_class_averages()` and `most_similar_cosine()` and `most_similar_euclid()` and `most_similar_manhattan()`. Read through all of their descriptions carefully.

`find_class_averages()` : Returns a pandas dataframe with no headers where each row represents the average values of the class. The last column should indicate the class. Round each value to three decimal places.
`most_similar_cosine()` : Find the class that best matches the test input using cosine similarity . Whichever vector in the class list has the smallest angle from the test vector is the class that you want to return.
`most_similar_euclid()` : Find the class that best matches the test input using Euclid's distance. Whichever vector in the class list has the smallest distance from the test vector is the class that you want to return.
`most_similar_manhattan()` : Find the class that best matches the test input using Manhattan distance. Whichever vector in the class list has the smallest distance from the test vector is the class that you want to return.

Remember, you will lose points if you do not follow the instructions. We are using a grading script. Some important notes:

- Though you are using `iris.csv` for your tests, you CANNOT assume that we will be using the same file for other test cases. We can use a file with a different number of columns and/or rows. The classes may also be different. However, you can assume that each row of the csv file is a data vector for a class. You may also assume that the last column by the class (as a str). The remaining columns will be the data (numeric values). Note that the csv file does not have headers
- Do NOT use library functions that will compute the distance metrics for you! Do NOT import other libraries such as scipy or sklearn. You will get an automatic zero! If you are unsure about a function, please ask me.

2. Your job is to implement `most_similar_cosine()` and `most_similar_euclid()` and `most_similar_manhattan()` so that it passes any test case. There are nine sample test cases provided for you, but these are not the only cases that we will test. We will be testing other test cases in the same way the test cases are presented.

3. After completing these functions, comment out the test cases (or delete them) or else the grading script will pick it up and mark your program as incorrect.

4. Convert your `metric_similarity.py` file to a `.txt` file. Submit your `metric_similarity.py` file and your `.txt` file on BeachBoard. Do NOT submit it in compressed folder.

Some helpful functions (feel free to google these functions to get more details)

Function name	What it does
`df_name.loc[condition]`	Returns all rows that fulfills `condition`. This results in a series. `Sub_df = df.loc[df[4] == 'hello'] =>` `Sub_df` is a sub-dataframe that contains all rows from `df` where column `4` contains '`hello`'
`df_name.mean()`	Returns the average of all rows in the dataframe
`df_name.round(num)`	Rounds all values in `df_name` to `num` decimal places
`df_name.append(series_name, ignore_index=True)`	Appends `series_name` onto `df_name`
`df_name.insert(index, col_name, values)`	Inserts a new column consisting of `values` with a column name of `col_name` at `index`
`np.linalg.norm(arr)`	Calculates the norm/length of `arr`

`math.acos(val)`	Calculates arccos (val) or $cos^{-1}(val)$. Note that this returns a value in radians and not degrees!
`math.sqrt(val)`	Calculates $\sqrt{val}$
`np.abs(arr)`	Takes absolute value of array `np.abs([-1, 2, -3]) -> [1, 2, 3]`

Grading rubric:

To achieve any points, your submission must have the following. Anything missing from this list will result in an automatic zero. NO EXCEPTIONS!

- Submit everything: py file, txt file
- Program has no errors (infinite loops, syntax errors, logical errors, etc.) that terminates the program

Please note that if you change the function headers or if you do not return the proper outputs according to the function requirements, you risk losing all points for those test cases.

Points	Requirement
10	Implemented `find_class_averages()` correctly
10	Implemented `most_similar_cosine()` correctly
10	Implemented `most_similar_euclid()` correctly
10	Implemented `most_similar_manhattan()` correctly
TOTAL: 40	

** Note that there are no points for passing the original test cases. Because this is an extra credit assignment, your points will come purely from hidden test cases.