Distance Metrics

Clustering methods rely very heavily on our definition of distance. Our choice of Distance Metric will be extremely important when discussing our clustering algorithms and for clustering success.

Each metric has strengths and most appropriate use cases, but sometimes choosing a distance metric is also based on empirical evaluation to determine which metric works best to achieve our goals.

These are the most common distance metrics:

Euclidean Distance

This one is the most intuitive distance metric, and that we use in K-means, another name for this is the L2 distance. You probably remember from your trigonometry classes.

We calculate (d) by taking the square root of the square of each of this changes (values). We can move this to higher dimensions for example 3 dimensions, 4 dimensions etc. In general, for an n-dimensional space, the distance is:

$$d(p,q) = \sqrt{(P_1 - q_I)^2 + (P_2 - q_2)^2 + \cdots (p_i + q_i)^2 + \cdots (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (P_1 - q_i)^2}$$

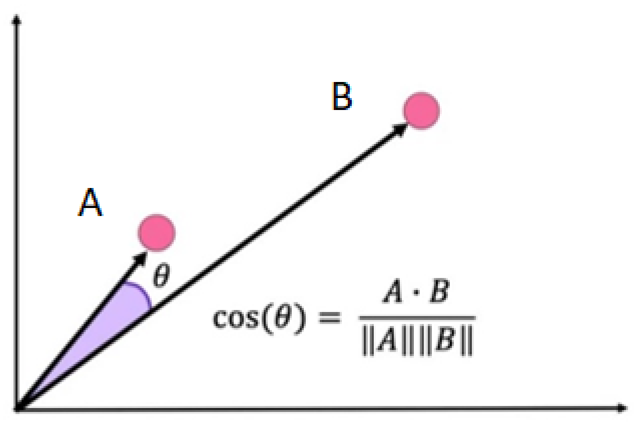
Manhattan Distance (L1 or City Block)

Another distance metric is the L1 distance or the Manhattan distance, and instead of squaring each term we are adding up the absolute value of each term. It will always be larger than the L2 distance, unless they lie on the same axis. We use this in business cases where there is very high dimensionality.

As high dimensionality often leads to difficulty in distinguishing distances between one point and the other, the L1 score does better than the L2 score in distinguishing these different distances once we move into a higher dimensional space.

Cosine Distance

This is a bit less intuitive than the distance metric. What we really care about the Cosine Distance is the angle between 2 points, for example, for two given points A and B:



This metric gives us the cosine of the angle between the two vectors defined from the origin to two given points in a two-dimensional space. To translate this definition into higher dimensions, we take the dot product of the vectors and divide it by the norm of each point.

The key to the Cosine distance is that it will remain insensitive to the scaling with respect to the origin, which means that we can move some of the points along the same line and the distance will remain the same. So, any two points on that same array, passing through the origin will have a distance of zero from one another.

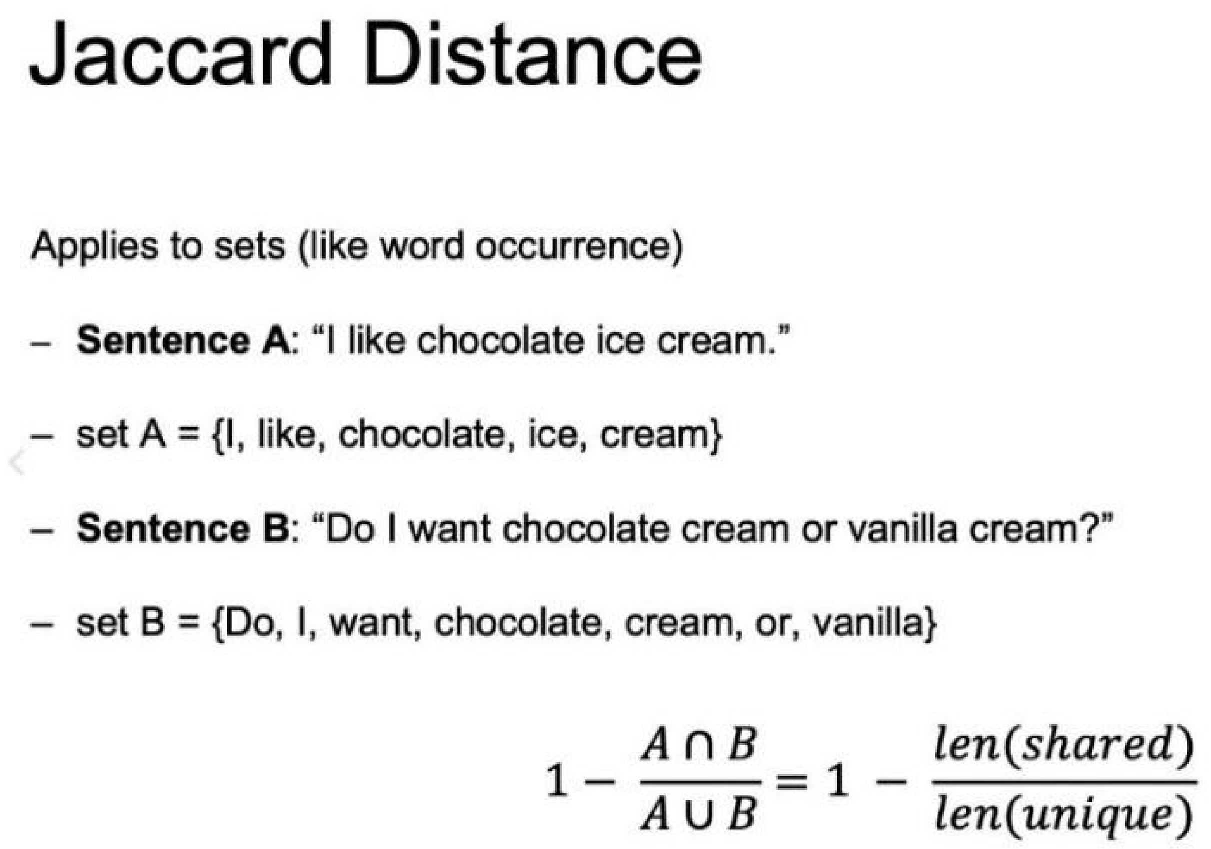
<u>Euclidean VS Cosine distances</u>

- Euclidean distance is useful for coordinate based measurements.
- Euclidean distance is more sensitive to curse of dimensionality.
- Cosine is better for data such as text where location of occurrence is less important.

Jaccard Distance

This distance is useful for texts and is often used in word occurrence.

Consider the following example:



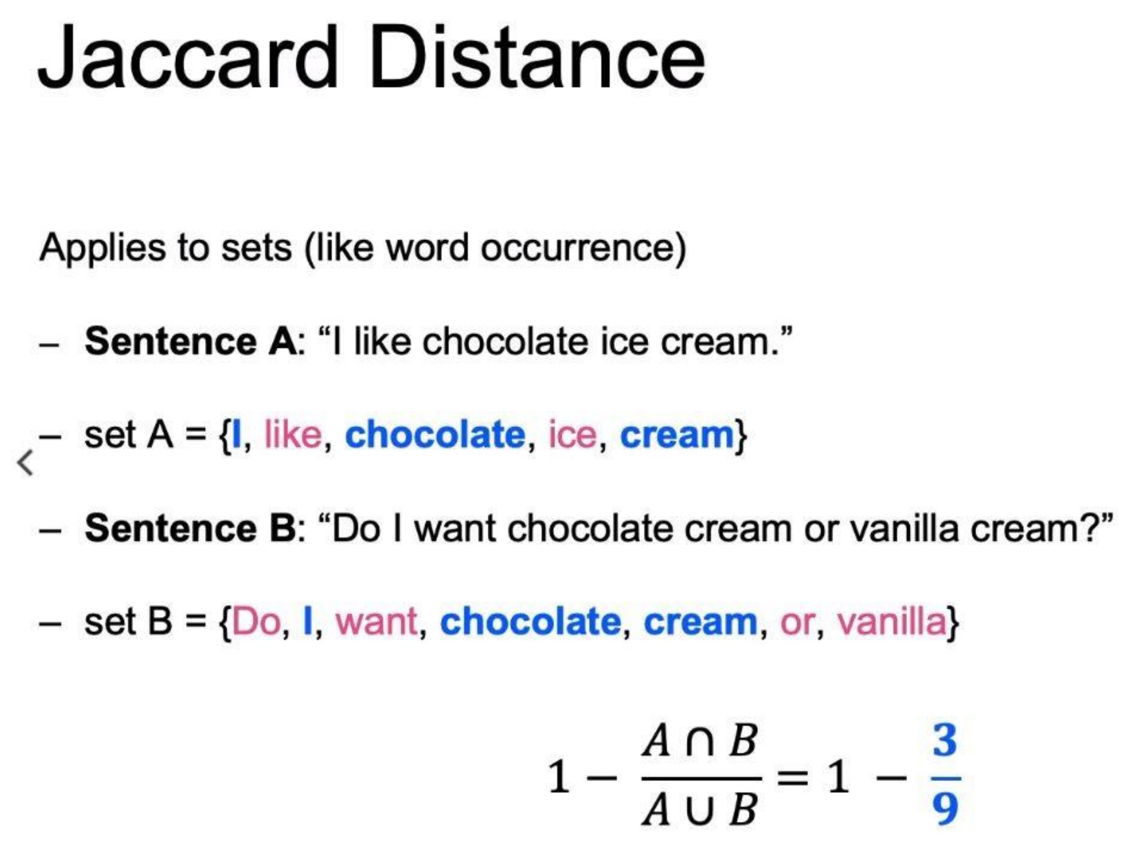
Jaccard Distance

Applies to sets (like word occurrence)

- **Sentence A**: “I like chocolate ice cream.”
- set A = {I, like, chocolate, ice, cream}
- **Sentence B**: “Do I want chocolate cream or vanilla cream?”
- set B = {Do, I, want, chocolate, cream, or, vanilla}

$$1 - \frac{A \cap B}{A \cup B} = 1 - \frac{len(shared)}{len(unique)}$$

In this case, the Jaccard Distance is going to be one minus the amount of value shared. So, the intersection over that union. This intersection means, the shared values of the two sentences over the length of the total unique values between sentences A and B.



Jaccard Distance

Applies to sets (like word occurrence)

– **Sentence A**: “I like chocolate ice cream.”

– set A = {**I**, like, **chocolate**, ice, **cream**}

– **Sentence B**: “Do I want chocolate cream or vanilla cream?”

– set B = {Do, **I**, want, **chocolate**, **cream**, or, vanilla}

$$1 - \frac{A \cap B}{A \cup B} = 1 - \frac{\mathbf{3}}{\mathbf{9}}$$

It can be useful in cases you have text documents and you want to group similar topics together.