TOOL

Similarity Metrics

Similarity metrics are often used in machine learning and natural language processing algorithms to compare two objects. This numerical measurement of *likeness* is used in algorithms for identifying related documents from a collection of documents, comparing sentences, and finding synonyms or antonyms of words.

There are many different ways to calculate *similarity*, and deciding which metric to use depends on the specific situation. Typically, we choose more standardized metrics for human interpretability and less standardized metrics for algorithmic implementation to speed up the repetitive computations. Many of these metrics differ in the degree of standardization and in what they measure (either angle between two vectors $x, y \in \mathbb{R}^p$ or the distance between them).

This tool will help you distinguish the three popular similarity metrics discussed in this course.

Metric

Dot product (similarity)

$$x \cdot y = x'y = \sum_{i=1}^{p} x_i y_i$$

In Python

```
x@y
np.matmul(x, y)
np.dot(x, y)
np.inner(x, y)
sum(x*y)
sum([i*j for i,j in zip(x, y)])
sum(map(operator.mul, x, y))
more_itertools.dotproduct(x, y)
sum(map(lambda i,j: i*j, x, y))
```

Description

Computed as a sum of values from element-wise multiplication (i.e., sum product) of vectors $x, y \in \mathbb{R}^p$. Since same-signed elements in two corresponding positions contribute to the sum and opposite-signed values take away from the sum, higher value indicates greater similarity between two vectors.

Pro — Can be used to pick the "closest" candidate to a target by comparing dot products of candidate vectors with the target's vector; cheap to compute.

Con — Is a non-standardized metric that can be difficult to interpret on a non-relative basis.

Metric

Cosine similarity

$$\frac{x \cdot y}{\sqrt{(x \cdot x)(y \cdot y)}}$$

In Python

```
np.dot(x/(x@x)**.5, y/(y@y)**.5)
1 - cosine(x, y)
np.dot(x,y)/(np.norm(x)*np.norm(y))
cosine_similarity(x, y)
```

Description

Rescales the two input vectors to unit length (without changing their directions) then applies a dot product.

To rescale a vector, we divide it by any non-zero real value, which is done element-wise. In this case, we rescale it by its length; i.e., its norm or magnitude. Because a non-zero vector's length is positive, such rescaling does not change the direction of a vector. For example, dividing or multiplying a vector by a scalar -1 would flip it 180 degrees so that it stays on the same line but completely changes its orientation.

Its output is always in [-1, 1] range by design, where 1 indicates 0 angle between two vectors, -1 indicates a 180-degree angle, and 0 indicates orthogonal (or perpendicular) vectors with a 90-degree angle between them.

Note: The two vectors with a cosine similarity of 1 may not be the same. For example, [0,1] and [0,2] are on the same vertical axis with an angle zero, but the second vector is twice the length of the first. Once we rescale these, they both become [0,1] or unit length with dot product [0,1] [0,1]=1, but the original dot product is [0,1] [0,2]=2. Take a pen and paper and try drawing some vectors and computing their similarities.

Pro — Is a popular standardized version of dot product with an output between [-1,1] that corresponds with the cosine of the angle between vectors x,y.

Con — Slower to compute compared to dot product.

Metric

Correlation (similarity)

$$\frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2\sum(y_i-\bar{y})^2}}\text{ , where }\bar{x}:=\frac{1}{p}\sum_{i=1}^{p}x_i$$

In Python

```
np.corrcoef(x, y)[0,1]
scipy.stats.pearsonr(x, y)
```

Description

Same as the cosine similarity for centered vectors.

Pro — Is a standardized metric that has output between [-1, 1]; has strong statistical guarantees; well studied in many textbooks.

Con — Centering vectors makes computation even slower than that of cosine similarity.

Metric

L2 (Euclidean) distance

$$\|x-y\|_2=\sqrt{\sum_{i=1}^{p}(x_i-y_i)^2}$$

In Python

```
euclidean(x, y)
sum((x - y)**2)**0.5
cdist(XA, XB, ‘euclidean’)
```

Description

The length of a straight line between two points.

Aka L2 distance, Euclidean, or Euclidean-2 distance.

Pro — Interpretability; attractive statistical and mathematical guarantees; sensitive to effect of outlier elements.

Con — Not informative at high dimensions.

Metric

L1 (Manhattan) distance

$$\|x - y\|_1 = \sum_{i=i}^{p} |x_i - y_i|$$

In Python

```
sum(abs(x - y))
np.abs(x - y).sum()
cdist(x, y, ‘cityblock’)
```

Description

The sum of the absolute elements of the difference between the two vectors. It measures the path between two distances with 90-degree turns.

Aka Manhattan distance, cityblock distance, taxicab distance, Euclidean-1 distance.

Pro — Interpretability and attractive statistical and mathematical guarantees; less sensitive to effect of outlier elements.

Con — Not informative at high dimensions.