Distributed word representations: Vector comparison

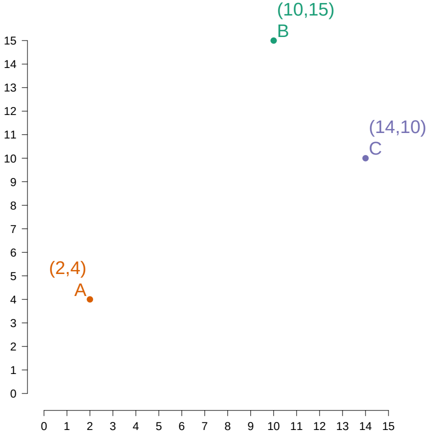
Christopher Potts

Stanford Linguistics

CS224u: Natural language understanding

Running example

	d_x	d_y
A	2	4
B	10	15
C	14	10



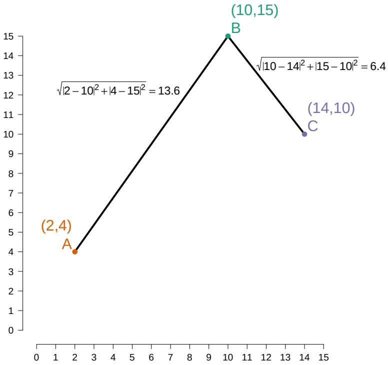
- Focus on distance measures
- Illustrations with row vectors

Euclidean

Between vectors *u* and *v* of dimension *n*:

$$\textbf{euclidean}(u, v) = \sqrt{\sum_{i=1}^{n} |u_i - v_i|^2}$$

	d_x	d_y
A	2	4
B	10	15
C	14	10



Length normalization

Given a vector u of dimension n, the L2-length of u is

$$\|u\|_2 = \sqrt{\sum_{i=1}^{n} u_i^2}$$

and the length normalization of u is

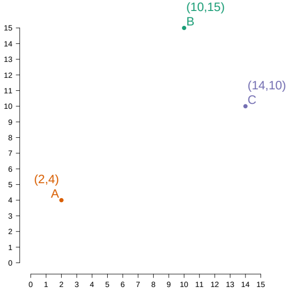
$$\left[\frac{u_1}{\|u\|_2}, \frac{u_2}{\|u\|_2}, \cdots, \frac{u_n}{\|u\|_2}\right]$$

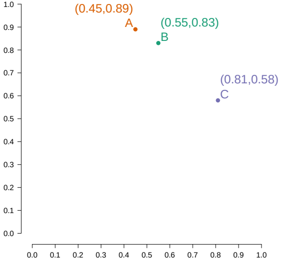
Length normalization

	d_x	d_y	$\|\|u\|\|_2$
A	2	4	4.47
B	10	15	18.03
C	14	10	17.20

row L2 norm $\Rightarrow$

	d_x	d_y
A	$\frac{2}{4.47}$	$\frac{4}{4.47}$
B	$\frac{10}{18.03}$	$\frac{15}{18.03}$
C	$\frac{14}{17.20}$	$\frac{10}{17.20}$





Cosine distance

Between vectors u and v of dimension n:

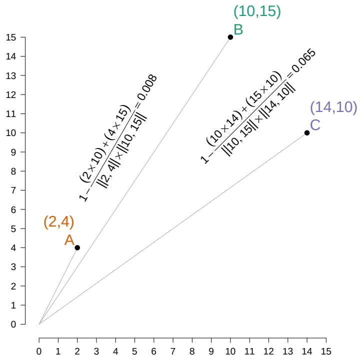
$$\mathbf{cosine}(u, v) = 1 - \frac{\sum_{i=1}^{n} u_i \times v_i}{\|u\|_2 \times \|v\|_2}$$

Cosine distance

Between vectors u and v of dimension n:

$$\textbf{cosine}(u, v) = 1 - \frac{\sum_{i=1}^{n} u_i \times v_i}{\|u\|_2 \times \|v\|_2}$$

	d_x	d_y
A	2	4
B	10	15
C	14	10

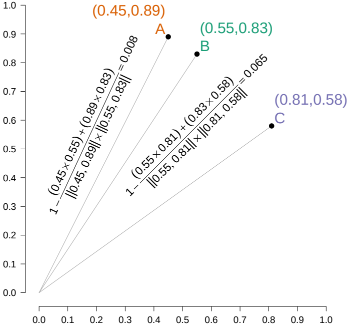


Cosine distance

Between vectors u and v of dimension n:

$$\mathbf{cosine}(u, v) = 1 - \frac{\sum_{i=1}^{n} u_i \times v_i}{\|u\|_2 \times \|v\|_2}$$

	d_x	d_y
A	2	4
B	10	15
C	14	10



Matching-based methods

Matching coefficient

$$\textbf{matching}(u, v) = \sum_{i=1}^{n} \min(u_i, v_i)$$

Jaccard distance

$$\textbf{jaccard}(u, v) = 1 - \frac{\textbf{matching}(u, v)}{\sum_{i=1}^{n} \max(u_i, v_i)}$$

Dice distance

$$\textbf{dice}(u, v) = 1 - \frac{2 \times \textbf{matching}(u, v)}{\sum_{i=1}^{n} u_i + v_i}$$

Overlap

$$\textbf{overlap}(u, v) = 1 - \frac{\textbf{matching}(u, v)}{\min\left(\sum_{i=1}^{n} u_i \,,\ \sum_{i=1}^{n} v_i\right)}$$

KL divergence and variants

KL divergence

Between probability distributions p and q:

$$D(p \parallel q) = \sum_{i=1}^{n} p_i \log\left(\frac{p_i}{q_i}\right)$$

p is the reference distribution. Before calculation, smooth by adding ϵ.

Symmetric KL

$$D(p \parallel q) + D(q \parallel p)$$

Jensen–Shannon distance

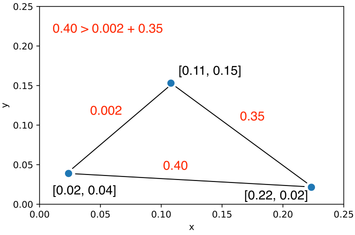
$$\sqrt{\frac{1}{2}D\left(p \parallel \frac{p+q}{2}\right) + \frac{1}{2}D\left(q \parallel \frac{p+q}{2}\right)}$$

Proper distance metric?

To qualify as a distance metric, a vector comparison method d has to be symmetric ($d(x,y) = d(y,x)$), assign 0 to identical vectors ($d(x,x) = 0$), and satisfy the **triangle inequality**:

$$d(x,z) \leqslant d(x,y) + d(y,z)$$

Cosine distance as I defined it doesn't satisfy this:



Distance metric?

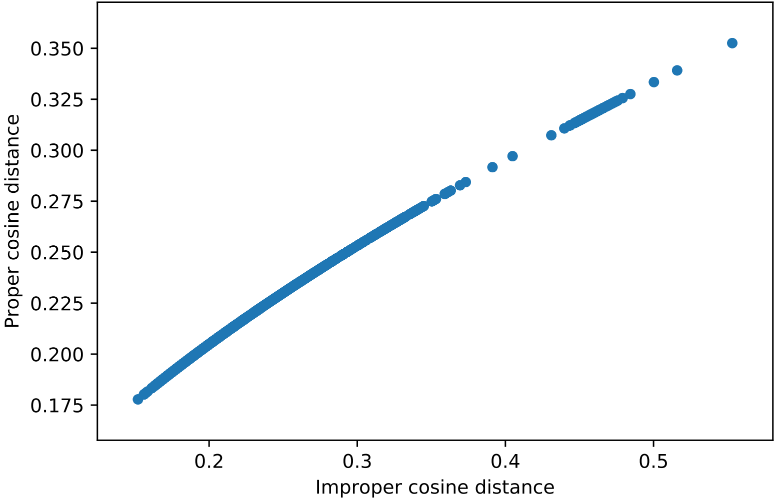
Yes: Euclidean, Jaccard for binary vectors, Jensen–Shannon, cosine as

$$\frac{\cos^{-1}\left(\frac{\sum_{i=1}^{n} u_i \times v_i}{\|u\|_2 \times \|v\|_2}\right)}{\pi}$$

No: Matching, Jaccard, Dice, Overlap, KL divergence, Symmetric KL

Comparing the two versions of cosine

Random sample of 100 vectors from our `giga20` count matrix. Correlation is 99.8.



Relationships and generalizations

1. Euclidean, Jaccard, and Dice with raw count vectors will tend to favor raw frequency over distributional patterns.

2. Euclidean with L2-normed vectors is equivalent to cosine w.r.t. ranking.

3. Jaccard and Dice are equivalent w.r.t. ranking.

4. Both L2-norms and probability distributions can obscure differences in the amount/strength of evidence, which can in turn have an effect on the reliability of cosine, normed-euclidean, and KL divergence. These shortcomings might be addressed through weighting schemes.

Code snippets

```
[1]: import os
     import pandas as pd
     import vsm

[2]: ABC = pd.DataFrame([
         [ 2.0,  4.0],
         [10.0, 15.0],
         [14.0, 10.0]], index=['A', 'B', 'C'], columns=['x', 'y'])

[3]: vsm.euclidean(ABC.loc['A'], ABC.loc['B'])

[3]: 13.601470508735444

[4]: vsm.vector_length(ABC.loc['A'])

[4]: 4.47213595499958

[5]: vsm.length_norm(ABC.loc['A']).values

[5]: array([0.4472136 , 0.89442719])

[6]: vsm.cosine(ABC.loc['A'], ABC.loc['B'])

[6]: 0.007722123286332261

[7]: vsm.matching(ABC.loc['A'], ABC.loc['B'])

[7]: 6.0

[8]: vsm.jaccard(ABC.loc['A'], ABC.loc['B'])

[8]: 0.76
```

Code snippets

```
[9]:  DATA_HOME = os.path.join('data', 'vsmdata')

      yelp5 = pd.read_csv(
          os.path.join(DATA_HOME, 'yelp_window5-scaled.csv.gz'), index_col=0)

[10]: vsm.cosine(yelp5.loc['good'], yelp5.loc['excellent'])

[10]: 0.1197421543700451

[11]: vsm.cosine(yelp5.loc['good'], yelp5.loc['bad'])

[11]: 0.14118253033888817

[12]: vsm.neighbors('bad', yelp5).head()

[12]: bad              0.000000
      unfortunately    0.116183
      memorable        0.120179
      …              0.122024
      obviously        0.123120
      dtype: float64

[13]: vsm.neighbors('bad', yelp5, distfunc=vsm.jaccard).head(3)

[13]: bad       0.000000
      ..        0.452427
      though    0.484269
      dtype: float64
```