Lecture Notes: Vector Semantics and Word Embeddings

CS375: NLP / Williams College / Spring 2023

1 Vectors review

Let's start with a review of vectors and vector notation.

We'll begin with 2D vectors (and we'll make everything genertic to more than two dimensions eventually). Let's call a generic vector $\vec{v} = [a, b]$ where a and b are real numbers where vector connects the origin, $[0, 0]$ to that point.

Recall, all vectors have a **direction** and **magnitude**. For example $\vec{v}_1 = [2, 2]$ and $\vec{v}_2 = [4, 4]$ have the same directions but not the same magnitudes. In our example, $\vec{v}_2$ has a larger magnitude.

The magnitude comes from the Pythagorean thereom

$$a^2 + b^2 = c^2 \tag{1}$$

and in the case of a 2D vector, the hypotenuse of the right triangle (c) is the magnitude of our vector. Let $||\vec{v}||$ denote the magnitude of the vector

$$||\vec{v}|| = \sqrt{a^2 + b^2} \tag{2}$$

Notice, this looks similar to the vector dot producted with itself

$$\vec{v} \cdot \vec{v} = a * a + b * b \tag{3}$$

Thus,

$$||\vec{v}|| = \sqrt{a^2 + b^2} \tag{4}$$

$$= \sqrt{\vec{v} \cdot \vec{v}} \tag{5}$$

This extends for higher dimensions as well. Let $\vec{v} = [x_1, x_2, x_3, \ldots, x_k]$ then

$$||\vec{v}|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_k^2} \tag{6}$$

$$= \sqrt{\sum_{i=1}^{k} x_i^2} \tag{7}$$

---

2 Similarity/distance metrics

We have two main options for similarity metrics: minimizing Euclidean distance and maximizing cosine similarity (minimizing cosine distance).

Recall, all our vectors are positive so we're only working in the first quadrant (in 2D space).

The **Euclidean distance**[1] between vectors $\vec{v}_1$ and $\vec{v}_2$ is defined as

$$||\vec{v}_1 - \vec{v}_2|| = \sqrt{(\vec{v}_1 - \vec{v}_2) \cdot (\vec{v}_1 - \vec{v}_2)} \tag{8}$$

[1] This is called "Euclidean" because its the geometry Euclid developed. This is the same kind of geometry you probably learned in high school.

We can interpret this as the magnitude of the line that connects the two vectors at their end points.

The **cosine similarity**[2] between two vectors is the value of the cosine at the angle between them

$$\text{CosSim}(\vec{v}_1, \vec{v}_2) = \cos \angle_{v_1 v_2} = \frac{\vec{v}_1 \cdot \vec{v}_2}{||\vec{v}_1|| ||\vec{v}_2||} \tag{9}$$

Example: Try finding the cosine similarity beteween $\vec{v}_1 = [1, 0]$ and $\vec{v}_2 = [0, 1]$.

We can change this to **cosine distance** via

$$\text{CosDist}(\vec{v}_1, \vec{v}_2) = 1 - \text{CosSim}(\vec{v}_1, \vec{v}_2) \tag{10}$$

Two questions you might have:

1. How are cosine distance and Euclidean distance the same/different?
2. When should you use one versus the other?

Euclidean distance can act as a similarity metric because it will tend to be low (more similar) when vectors have large values in the same dimensions. However, (without normalizing first), Euclidean distance has the issue that the frequency of words matter. This is where we might prefer cosine similarity.

For taks of **ranking** the most similar vector, we'll show that cosine distance and Euclidean distance produce the same results when vectors are normalized.[3]

First, we'll normalize two vectors of interest, x' and y', by dividing by their norm/magnitude:

$$x = \frac{x'}{||x'||} \tag{11}$$

and

$$y = \frac{y'}{||y'||}.$$

So we have $||x'|| = ||y'|| = 1$.

[2] Proof invloves Law of Cosines: https://proofwiki.org/wiki/Cosine_Formula_for_Dot_Product.
[3] Proof from Manning and Schuetze section 8.5.

Then, the squared Euclidean distance between these two vectors gives us

$$\begin{aligned}
(\text{EuclDist}(x,y))^2 &= (||x-y||)^2 \\
&= (\sqrt{(x-y)\cdot(x-y)})^2 \\
&= (x-y)\cdot(x-y) \\
&= \sum_{i=1,k}(x_i-y_i)^2 \\
&= \sum_{i=1,k}x_i^2 - 2\sum_{i=1,k}x_iy_i + \sum_{i=1,k}y_i^2 \\
&= (||x||)^2 - 2\sum_{i=1,k}x_iy_i + (||y||)^2 \\
&= 1 - 2\sum_{i=1,k}x_iy_i + 1 \\
&= 2 - 2x\cdot y \\
&= 2(1-x\cdot y) \\
&\propto 1 - \text{CosSim}(x,y) \\
&\propto \text{CosineDistance}(x,y)
\end{aligned}$$

Thus if we normalize vectors (make their magnitude equal to 1), we can use Euclidean distance in place of cosine distance/similarity to find ranked neighbors. In certain situations, one of these metrics may be more computationally efficient than the other.

---

3 word2vec

Let's derive how word2vec skip-gram negative sampling (SGNG) works to learn "dense" embeddings from a corpus. We follow Goldberg and Levy 2014.[4]

Step 1: Initialization. Randomly initialize two sets of real-valued vectors, each vector with dimensionality k. The first are our embeddings of our "target" words, $w \in V$, which we call $\vec{v}_w \in R^k$. The second are the embeddings of context words, $c \in V$, which we call $\vec{v}_c \in R^k$. The goal of this model is to learn "good" vectors for these words.

Step 2: Save context words – "positive" training set. Given a context window, L, find context words c that exist in the $\pm L$ conext window of the target word w, for all $w \in V$. Save this set as $(w,c) \in \mathcal{D}$.

Step 3: Randomly sample "negative" training set. Create a set of "negative" pairs, which we call $\mathcal{D}'$. We create this set by sampling words c' proportional to their unigram distribution

$$\hat{P}(c) = \frac{\text{Count(word } c\text{)}}{\text{Count(all tokens in corpus)}} \tag{12}$$

that *never* occur in the context of the target word w.

Step 4: Loss function. Our goal is to learn embeddings that maximizes the true occurring pairs of words, $\mathcal{D}$ and minimizes the randomly sampled (never occurring) pairs of words, $\mathcal{D}'$. To change this to a binary prediction task, label the former as $d = 1$ and the later as $d = 0$.

[4] https://arxiv.org/pdf/1402.3722.pdf

First, recall that cosine similarity is our target intrinsic property for these embeddings. So for $(w, c) \in \mathcal{D}$ we want to maximimze

$$\text{CosineSim}(\vec{v}_w, \vec{v}_c) = \frac{\vec{v}_w \cdot \vec{v}_c}{||\vec{v}_w||||\vec{v}_c||} \tag{13}$$

$$\propto \vec{v}_w \cdot \vec{v}_c \tag{14}$$

Now we need to turn this into a probability. We'll use the same logistic function we used for Logistic regression. We predict the probability words (w, c) came from the true context windows $d = 1$ or not $d = 0$. Then

$$P(d = 1|\vec{v}_w, \vec{v}_c) = \sigma(\vec{v}_w \cdot \vec{v}_c) = \frac{1}{1 + e^{-\vec{v}_w \cdot \vec{v}_c}} \tag{15}$$

Our goal is thus

$$\underset{\vec{v}_w, \vec{v}_c}{\text{argmax}} \prod_{(w,c) \in \mathcal{D}} P(d = 1|\vec{v}_w, \vec{v}_c) \prod_{(w,c') \in \mathcal{D}'} P(d = 0|\vec{v}_w, \vec{v}_{c'}) \tag{16}$$

Changing this into a loss function that we want to minimize, we have

$$\underset{\vec{v}_w, \vec{v}_c}{\text{argmin}} \Big(- \prod_{(w,c) \in \mathcal{D}} P(d = 1|\vec{v}_w, \vec{v}_c) \prod_{(w,c') \in \mathcal{D}'} P(d = 0|\vec{v}_w, \vec{v}_{c'}) \Big) \tag{17}$$

The vectors themselves, $\vec{v}_w$ and $\vec{v}_c$ for all $w, c \in V$, are the model parameters (analogous to θ as the model parameter for Logistic Regression). We use gradient descent to find these parameters.[5]

[5] From Goldberg and Levy: "If we fix the words representation and learn only the contexts representation, or fix the contexts representation and learn only the word representations, the model reduces to logistic regression, and is convex. However, in this model the words and contexts representations are learned jointly, making the model non-convex.