Embeddings

Stephen Boyd Aqib Syed

ENGR108
Stanford University

October 16, 2025

Embedding

- an **embedding** is a function ϕ that maps objects into vectors
- objects can be words, text, documents, computer code, images, songs, video, DNA sequences, zip codes, ...
- if a is an object then $x = \phi(a)$ is an n-vector
- typical embedding dimensions range from 2 or 3 (for visualization) to 1000 or more (for AI tasks)
- often $x = \phi(a)$ has mean $\approx$ zero, RMS value $\approx$ one.

Fidelity

basic goal of an embedding is **fidelity**:

embedded vectors $\phi(a)$ and $\phi(a')$ are **close**, *i.e.*, $\|\phi(a) - \phi(a')\|$ is small

if and only if

objects a and a' are **similar**

- the first concept is **quantitative** (distance between vectors)
- the second concept is **subjective** (judged by humans)

Variation: Cosine similarity

- use angle $\theta = \angle(\phi(a), \phi(a'))$ instead of distance $d = \|\phi(a) - \phi(a')\|$
- **cosine similarity** is defined as

$$\cos\theta = \frac{\phi(a)^T \phi(a')}{\|\phi(a)\|\|\phi(a')\|}$$

- ranges between -1 and $+1$
- fidelity means objects a and a' are similar if and only if cosine similarity is large, *i.e.*, θ is near one
- when $\phi(a)$ all have same norm, *e.g.*, one, then $d^2 = 2 - 2\cos\theta$ so

$$\theta = (1/2)\cos^{-1}\left(2 - d^2\right)$$

- as $d \to 0$, $\theta \to 1$, so large cosine similarity is the same as small distance

Illustration: Embedding words into 2-vectors

lawyer
doctor
teacher
professor

blue
red
green
yellow

china
beijing
shanghai
shenzhen

Simple versus complex embeddings

- **simple embeddings** don't involve much computation, *e.g.*,
  - standarized feature vector associated with object
  - word count histogram of a document
- components of simple embedding often have meaning

- **complex embeddings** a.k.a. **AI embeddings**
  - involve much computation
  - developed using huge data sets, *e.g.*, all images on the internet
- often used for complex objects like unstructured text
- components of complex embedding typically have no meaning

Complex (AI) embeddings

- often have neural network structure
- and many, many parameters (*e.g.*, billions)
- the parameters are chosen ('trained') based on huge data sets
  - everything written in English available on the internet
  - all images available on the internet
  - all Python code in public repositories
- surrogate is used for 'similar', *e.g.*, two words are 'similar' if they appear in the same sentence, paragraph, or chapter
- some groups open source publish the embedding function (structure and parameters), so anyone can use it

Example: GloVe 6B word embedding

- embeds words (really, tokens) into 300 dimensions
- has 6 billion parameters (!)
- from it we derive an embedding into 2 dimensions
- we take 10000 most common words in its training data set

Some nearest neighbors in GloVe 6B

Word	NN	2-NN	3-NN	4-NN	5-NN
lawyer	attorney	counsel	prosecutor	judge	businessman
university	professor	graduate	college	harvard	faculty
car	vehicle	truck	driver	driving	vehicles
guitar	bass	piano	guitarist	acoustic	violin
cat	dog	pet	monkey	horse	rabbit
happy	glad	pleased	really	everyone	everybody

Julia code: load data, find word index

```
julia> words, emb300, emb2 = word_embeddings_data();
julia> words[99] # 99th word in list of words
"under"
julia> idx = findfirst(==("lawyer"), words) # find index
1140
```

Julia code: retrieve word embedding

```
julia> emb300[idx] # get 300D embedding for "lawyer"
300-element Vector{Float32}:
-0.06121889
 0.01856082
-0.06817669
 ⋮
 0.00238545
-0.06792316
 0.08270543
```

Julia code: indices of 5 nearest neighbors

```
julia> nn_indices, _ = VMLS.k_nearest_neighbors(emb300,
       emb300[idx], 6);
julia> nn_indices
6-element Vector{Int64}:
1140
1009
3372
2197
780
3499
```

Julia code: 5 nearest neighbor words

```
julia> nn_words = [words[i] for i in nn_indices]
6-element Vector{String}:
"lawyer"
"attorney"
"counsel"
"prosecutor"
"judge"
"businessman"
```

Summary

- embeddings maps objects into vectors
- 'similar' objects map to close vectors
- allows us to use methods of linear algebra on objects, *e.g.*,
  - find nearest neighbors
  - k-means clustering
  - and other methods we'll see later in the course