S&DS 365 / 565

Data Mining and Machine Learning

Word Embeddings

3. litoria

4. leptodactylidae

5. rana

7. eleutherodactylus

Yale

Shortcomings of word clusters

- No centroid in $\mathbb{R}^d$, or Euclidean structure
- Can't use vector space operations
- "One hot" representation of classes wasteful
- No features of words outside of the above
- These are addressed with *distributed representations*

Word embeddings

Instead of representing words as classes, represent them as vectors

Still use PMI-like scores

Using co-occurence/local information

Constructing embeddings

probability w_2 appear within context of w_1

word 1 is mapped to some feature space

$$p(w_2 \mid w_1) = \frac{\exp(\phi(w_2)^T \phi(w_1))}{\sum_w \exp(\phi(w)^T \phi(w_1))}.$$

multinomial model

2-word model

Log-likelihood on bigram model is then

feature vector of w_1

$$\ell(\phi) = \sum_{w_1, w_2} \log p(\phi_2 \mid \phi_1) p(w_2 \mid \phi_2)$$

distributed representation: vectors shared info across different features

Where the sum is just over all bi-grams

Contrast to class based bi-grams

discrete structures

$$\ell(c) = \sum_{w_1, w_2} \log p(c_2 \mid c_1) p(w_2 \mid c_2)$$

Constructing embeddings

Carry out stochastic gradient descent over the embedding vectors $\phi \in \mathbb{R}^d$ (where $d \approx 50$–100 is chosen by trial and error)

This is Mikolov et al. (2014, 2015) did at Google. With a couple of twists (heuristics).

"Distributed representations of words," (2014) "Efficient representations of words in vector space" (2015)

Skip-gram

The idea of skip-gram is to predict the probabilities of a word being a context (surrounding) word for the current target word.

- Suppose we have the sentence “The man passes the sentence should swing the sword”, by Ned Stark
- Suppose our sliding window size is 5
- Then, the target word “swing” in the above case produces four training samples:
  - (“swing”, “sentence”)
  - (“swing”, “should”)
  - (“swing”, “the”)
  - (“swing”, “sword”)

Constructing embeddings

Heuristics used:

- Skip-gram: predict surrounding words from current word (now it's no longer a likelihood, since you're looking ahead!)
- Use "negative sampling"

$$p(w_2 \mid w_1) \propto \sum_{w} \exp(\phi(w)^T \phi(w_1)) \qquad \text{gigantic sum}$$

$$\approx \exp(\phi(w_2)^T \phi(w_1)) + \sum_{\text{random } w} \exp(\phi(w)^T \phi(w_1))$$

most of sum

extra term

→ a bunch of random words

"Distributed representations of words," (2014) "Efficient representations of words in vector space" (2015)

Analogies

These heuristics enable training on very large text collections. Leads to vector representations of words with interesting properties.

For example, analogies:

`king` is to `man` as ? is to `woman`

Analogies

These heuristics enable training on very large text collections. Leads to vector representations of words with interesting properties.

For example, analogies:

`king` is to `man` as ? is to `woman`

`Bill Gates` is to `Microsoft` as ? is to `Google`

Analogies 类比

These heuristics enable training on very large text collections. Leads to vector representations of words with interesting properties.

For example, analogies:

$\texttt{king}$ is to $\texttt{man}$ as ? is to $\texttt{woman}$

$\texttt{Bill Gates}$ is to $\texttt{Microsoft}$ as ? is to $\texttt{Google}$

$$\phi(\texttt{king}) - \phi(\texttt{man}) \overset{?}{\approx} \phi(\texttt{queen}) - \phi(\texttt{woman})$$

word $$\widehat{w} = \arg\min_{w} \|\phi(\texttt{king}) - \phi(\texttt{man}) + \phi(\texttt{woman}) - \phi(w)\|^2$$

≈ 0

Does $\widehat{w} = \texttt{queen}$?

Learned Analogies

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Mikolov et al., “Distributed representations of words,” (2014); “Efficient representations of words in vector space” (2015)

Evaluation Analogies

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Mikolov et al., “Distributed representations of words,” (2014); “Efficient representations of words in vector space” (2015)

Bias

$v = \phi(she) - \phi(he)$

v is an interesting direction, with that direction we project all other words onto v

compute $\langle \phi(w), v \rangle$

Extreme *she* occupations

1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

Extreme *he* occupations

1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss

Figure 1: The most extreme occupations as projected on to the *she*−*he* gender direction on g2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded.

GloVe

Shortly after: Stanford group introduced a computational expedient (with attempt to give a "principled" motivation)

$$\mathcal{O}(\phi) = \sum_{w_1, w_2} f(c_{w_1,w_2}) \left(\phi(w_1)^T \phi(w_2) - \log c_{w_1,w_2} \right)^2$$

where $c_{w,w'}$ are cooccurrence counts.

- $c_{w,w'}$ is the number of times word w' occurs in the context of word w
- A type of regression estimator. Can interpret/relate this to other objectives.
- Main advantage is that SGD can be carried out much more efficiently

---

Pennington et al., "GloVe: Global vectors for word representation," (2015)

GloVe

$$\mathcal{O}(\phi) = \sum_{w_1, w_2} f(c_{w_1,w_2}) \left(\phi(w_1)^T \phi(w_2) - \log c_{w_1,w_2} \right)^2$$

where $c_{w,w'}$ are cooccurrence counts.

- Heuristic weighting function

$0 < \frac{x}{x_{max}} < 1 \qquad \alpha > 0$

$$f(x) = \left(\frac{x}{x_{\max}} \right)^{\alpha}$$

  where $\alpha = 3/4$ set empirically.

- So $10^{-4} \mapsto 10^{-3}$. Each order of magnitude down gets "boosted" by 1/4-magnitude.

Pennington et al., "GloVe: Global vectors for word representation," (2015)

GloVe site and code

HW7

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

Introduction

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Getting started (Code download)

- Download the code (licensed under the Apache License, Version 2.0)
- Unpack the files: unzip GloVe-1.2.zip
- Compile the source: cd GloVe-1.2 && make
- Run the demo script: ./demo.sh
- Consult the included README for further usage details, or ask a question
- The code is also available on GitHub

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Citing GloVe

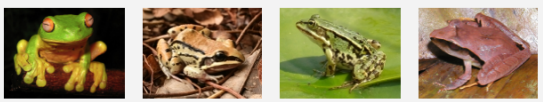
Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. [pdf] [bib]

Highlights

1. **Nearest neighbors**

   The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

   0. *frog*
   1. frogs
   2. toad
   3. litoria
   4. leptodactylidae
   5. rana
   6. lizard
   7. eleutherodactylus



3. litoria 4. leptodactylidae 5. rana 7. eleutherodactylus

2. **Linear substructures**

   The similarity metrics used for nearest neighbor evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, *man* may be regarded as similar to *woman* in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

Embedding Embeddings

- How can we visualize the embeddings? in 2D
- We're in a very high dimensional space (100)
- What have we learnt that might help? Dimension Reduction?

PCA — Linear relationship
t-SNE — non-linear

Dimension Reduction

- We could use PCA here
- The problem is that PCA is a linear method, whereas we expect our data to lie on a low-dimensional, non-linear manifold.
- Need a nonlinear method

t-SNE

Most popular is t-SNE: “Student-t Stochastic Neighborhood Embedding” Here’s the idea behind t-SNE: it minimizes the *divergence* between two distributions:

- a distribution that measures pairwise similarities of the word embedding vectors
- a distribution that measures pairwise similarities of the corresponding visualization vectors

Interpretation: if $\phi(w_i)$ is very close to $\phi(w_j)$ then y_i will be close to y_j. (long distances may be stretched further...)

Pronounced: **tee**-snee

Summary: word embeddings

- Word embeddings are vector representations of words, learned from cooccurrence statistics
- The models can be viewed in terms of multinomial logistic regression
- Surprising semantic relations are encoded in linear relations
- Various heuristics have been introduced to get scalability
- Embeddings improve with more data
- t-SNE is an algorithm for visualizing embeddings

t-SNE: Detailed algorithm

For each word w_i compute a language model

a conditional distribution model

Given word i, what probability word j is its neighbor

$$P_{j\,|\,i} \propto \exp\left(-\frac{\|\phi(w_i) - \phi(w_j)\|^2}{2h_i^2}\right)$$

different h for word i

That is: Gaussian kernel

$$P_{j\,|\,i} = \frac{\exp\left(-\frac{\|\phi(w_i) - \phi(w_j)\|^2}{2h_i^2}\right)}{\sum_k \exp\left(-\frac{\|\phi(w_i) - \phi(w_k)\|^2}{2h_i^2}\right)}$$

Choose the bandwith h_i so that the perplexity is, say, 10. This puts the probabilities all on the same scale.

$e^{H(j|i)} \approx 10$ paper: 5~50

entropy $H(j|i) = \sum_j p_{j|i} \log(p_{j|i})$

Pronounced: **tee**-snee

t-SNE: Detailed algorithm

For each word w_i compute a language model

$$P_{j\,|\,i} \propto \exp\left(-\frac{\|\phi(w_i) - \phi(w_j)\|^2}{2h_i^2}\right)$$

t-SNE: Detailed algorithm

original : Gaussian distribution

Swiss Roll → flatten out

For each word w_i compute a language model

yield non-linear ←

$$P_{j|i} \propto \exp\left(-\frac{\|\phi(w_i) - \phi(w_j)\|^2}{2h_i^2}\right)$$

Now form symmetralize matrix

$$P_{ij} = \frac{1}{2}\left(P_{j|i} + P_{i|j}\right)$$

as a simple way of symmetrizing.

Kernel PCA

① $P_{j|i}$ matrix

② SVD on $P_{j|i}$

First embed on high dimensional space then do SVD projection in infinite space

Refresh: t-distribution

$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})}\left(1+\frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}},$$

For t-distribution with 1 degree of freedom, this simplifies to

$$f(t) = \frac{1}{\pi}\left(1+t^2\right)^{-1},$$

e^{-t^2} (previous Gaussian)

which is also the Cauchy distribution.

t-SNE: Detailed algorithm

Now form Student-t distribution depending on the visualization vectors $y_i \in \mathbb{R}^2$:

$$Q_{ij} \propto \left(1 + \|y_i - y_j\|^2\right)^{-1}$$

t-SNE: Detailed algorithm

Heavy-tail

Now form Student-t distribution depending on the visualization vectors $y_i \in \mathbb{R}^2$: (visualize in 2D)

embedding vector

each entry of matrix is proportional to

$$Q_{ij} \propto \left(1 + \|y_i - y_j\|^2\right)^{-1}$$

→ encourage close more close, spread more spread

That is:

$$Q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq \ell} \left(1 + \|y_k - y_\ell\|^2\right)^{-1}}$$

Sum over row

This has fatter tails than a Gaussian

t-SNE: Detailed algorithm

Finally, run stochastic gradient descent (SGD) over the vectors y_i to optimize the Kullback-Leibler divergence between P and Q:

(loss) non-linear

$$\hat{y} = \arg\min \sum_{ij} P_{ij} \log P_{ij}/Q_{ij}$$

← influence by $\hat{y}$

P_{ij}: known

Q_{ij}: we are learning

$P_{ij} = \frac{1}{2}(P_{j|i} + P_{i|j})$

t-SNE

In summary:

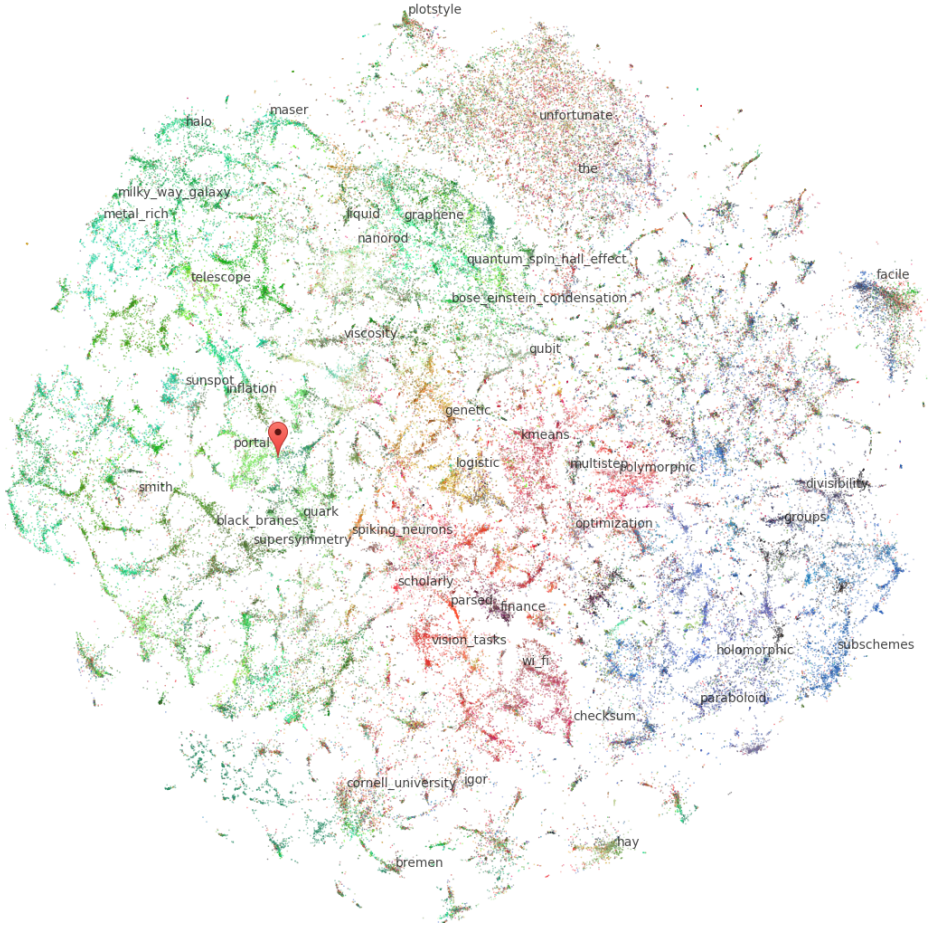
- Form a Gaussian kernel over vocabulary based on embedding vectors
- Scale and symmetrize, giving a matrix $P = [P_{ij}]$
- Represent word i by $y_i \in \mathbb{R}^2$. Use a heavy-tailed distribution (Student-t with one degree of freedom)
- Select y_i using stochastic gradient descent

t-SNE: More info and examples

https://lvdmaaten.github.io/tsne/
http://cs.stanford.edu/people/karpathy/tsnejs/

Note: This is just a visualization technique, to give intuition for the high dimensional embedding

t-SNE: Examples



http://www.cs.cornell.edu/~ginsparg/arxiv/gmaps2.html