STA 221: LECTURE 4

KRISHNA BALASUBRAMANIAN

(University of California, Davis)

Principal Component Analysis

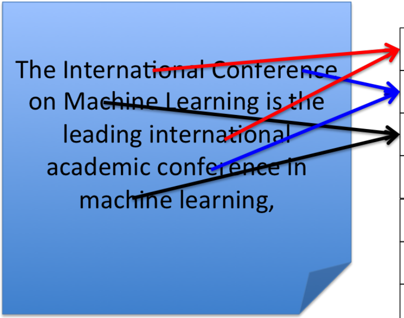
- Consider data sample $X^{(i)} in \mathbb{R}^d$.
- In linear PCA, we try to represent each data sample as linear combination of some fixed basis vector.

$$X^{(i)} = \sum_{j=1}^{p} \mathbf{v}_j \beta_j^{(i)} = \mathbf{V}\beta^{(i)}$$

  where $\mathbf{V} \in \mathbb{R}^{d \times p}$ is the matrix with the basis vectors $\mathbf{v}_j$ as its columns and $\beta_j^{(i)}$ denote the j^{th} coordinate of the vector $\beta^{(i)}$.
- Note that the basis vectors are fixed and does not change with i and the coefficients change with i.
- The reasoning behind this is that, you want to represent each sample ($X^{(i)} \in \mathbb{R}^d$) as a different linear combination ($\beta^{(i)} \in \mathbb{R}^p$) of the same basis vector.

Principal Component Analysis (PCA)

- Data matrix can be big.
- Example: bag-of-word model
- Each document is represented by a d-dimensional vector $\boldsymbol{x}$, where x_i is number of occurrences of word i.



(international)	2
(conference)	2
(machine)	2
(train)	0
(learning)	2
(leading)	1
(totoro)	0

number of features $=$ number of potential words $\approx$ 10,000

Feature generation for documents

- Bag of n-gram features ($n = 2$):

The International Conference on Machine Learning is the leading international academic conference in machine learning,

(international)	2
(conference)	2
(machine)	2
(train)	0
(learning)	2
(leading)	1
(totoro)	0

(international conference)	1
(machine learning)	2
(leading international)	1
(totoro tiger)	0
(tiger woods)	0
(international academic)	1
(international academic)	1

10,000 words $\Rightarrow 10,000^2$ potential features

Data Matrix (document)

- Use the bag-of-word matrix or the normalized version (TF-IDF) for a dataset (denoted by D):

$$\mathrm{tfidf}(\mathrm{doc}, \mathrm{word}, \mathrm{D}) = \mathit{tf}(\mathrm{doc}, \mathrm{word}) \cdot \mathit{idf}(\mathrm{word}, \mathrm{D})$$

  - tf(doc, word): term frequency

$$\frac{\text{word count in the document)}}{\text{(total number of terms in the document)}}$$

  - idf(word, Dataset): inverse document frequency

$$\frac{\text{log((Number of documents)}}{\text{(Number of documents with this word))}}$$

Data Matrix (document)

	angeles	los	new	post	times	york
d1	0	0	1	0	1	1
d2	0	0	1	1	0	1
d3	1	1	0	0	1	0

tf-idf

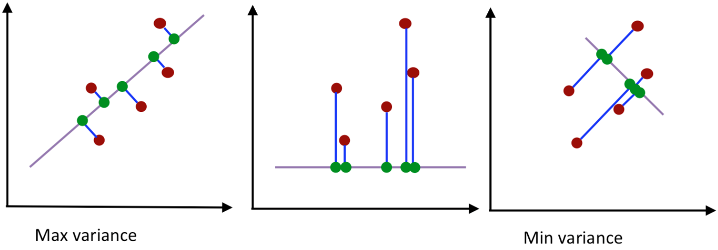
	angeles	los	new	post	times	york
d1	0	0	0.584	0	0.584	0.584
d2	0	0	0.584	1.584	0	0.584
d3	1.584	1.584	0	0	0.584	0

PCA: Motivation

- Data can have huge dimensionality:
  - Reuters text collection (rcv1): 677,399 documents, 47,236 features (words)
  - Pubmed abstract collection: 8,200,000 documents, 141,043 features (words)
- Can we find a low-dimensional representation for each document?
  - Enable many learning algorithms to run efficiently
  - Sometimes achieve better prediction performance (de-noising)
  - Visualize the data

PCA: Motivation

- Orthogonal projection of data onto lower-dimensional linear space that:
  - Maximize variance of projected data
    (preserve as much information as possible)
  - Minimize reconstruction error



PCA: Formulation

- Given (mean-zero) data $X^{(1)}, \cdots, X^{(n)} \in \mathbb{R}^d$, compute the principal vector $\mathbf{v}_1$ by:

$$\mathbf{v}_1 = \arg\max_{\|\mathbf{v}\|_2=1} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{v}^T X^{(i)})^2 \quad = \arg\max_{\|\mathbf{v}\|=1} \frac{1}{n} \mathbf{v}^T \widehat{X} \widehat{X}^T \mathbf{v}$$

  where each column of $\widehat{X}$ is $X^{(i)}$
- The first principal component $\mathbf{w}$ is the leading eigenvector of $\frac{1}{n}\widehat{X}\widehat{X}^T$ (eigenvector corresponding to the largest eigenvalue)

Algorithm 1 Stochastic/Online Algorithm for PCA

Input: Unit-vector $\mathbf{v}^{(0)} \in \mathbb{R}^d$ (Randomly generated) and step-size β.

for $t = 1, \dots, T$ **do**

Sample $X^{(t)}$ from the distribution P(X).

Let $\mathbf{v}^{(t)} = \mathbf{v}^{(t-1)} + \beta X^{(t)} X^{(t)\top} \mathbf{v}^{(t-1)}$

$\mathbf{v}^{(t)} = \frac{\mathbf{v}^{(t)}}{\|\mathbf{v}^{(t)}\|_2}$

end for

PCA: Computation

- PCA: top-p eigenvectors of $\widehat{X}\widehat{X}^T$
- Assume $\widehat{X} = U\Sigma V^T$, then principal components are U_p (top-p singular vectors of $\widehat{X}$)
- Projection of $\widehat{X}$ to U_p:

$$U_p^T \widehat{X} = \Sigma_p V_p^T \quad (p \text{ by } n \text{ matrix})$$

Each column is the p-dimensional features for an observation

Word Representation

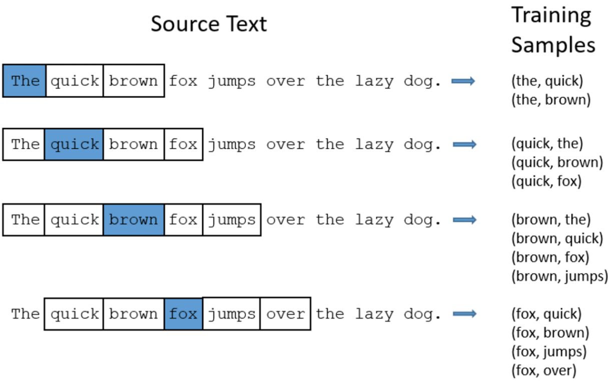
- Goal: understand the meaning of a word
- Given a large text corpus, how to learn low-dimensional features to represent a word?
- Skip-gram model:

  For each word w_i, define the "contexts" of the word as the words surrounding it in an L-sized window:

$$w_{i-L-2}, w_{i-L-1}, \underbrace{w_{i-L}, \cdots, w_{i-1}}_{\text{contexts of } w_i}, w_i, \underbrace{w_{i+1}, \cdots, w_{i+L}}_{\text{contexts of } w_i}, w_{i+L+1}, \cdots$$

- Get a collection of (word, context) pairs, denoted by D.

Skip-gram model



(Figure from `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/`)

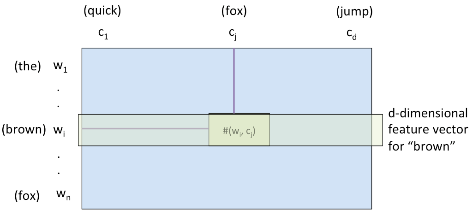
Use bag-of-word model

- Idea 1: Use the bag-of-word model to "describe" each word
- Assume we have context words $c_1, \cdots, c_d$ in the corpus, compute

$$\#(w, c_i) := \text{ number of times the pair } (w, c_i) \text{ appears in } D$$

- For each word w, form a d-dimensional (sparse) vector to describe w

$$\#(w, c_1), \cdots, \#(w, c_d),$$



PMI/PPMI Representation

- Similar to TF-IDF: Need to consider the frequency for each word and each context
- Instead of using co-ocurrent count $\#(w, c)$, we can define pointwise mutual information:

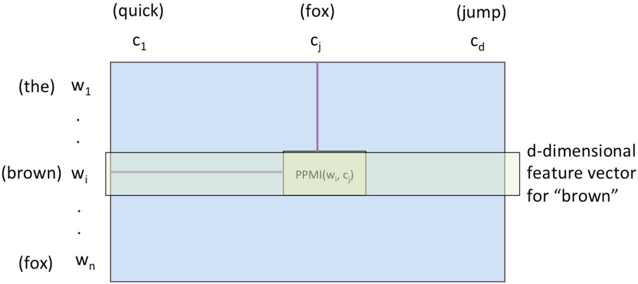
$$\text{PMI}(w, c) = \log\left(\frac{\widehat{P}(w, c)}{\widehat{P}(w)\widehat{P}(c)}\right) = \log \frac{\#(w, c)|D|}{\#(w)\#(c)},$$

  $\#(w) = \sum_c \#(w, c)$: number of times word w occurred in D
  $\#(c) = \sum_w \#(w, c)$: number of times context c occurred in D
  $|D|$: number of pairs in D
- Positive PMI (PPMI) usually achieves better performance:

$$\text{PPMI}(w, c) = \max(\text{PMI}(w, c), 0)$$

- M^{PPMI}: a n by d word feature matrix, each row is a word and each column is a context

PPMI Matrix


(quick)
(fox)
(jump)
c_1
c_j
c_d
(the) w_1
(brown) w_i
(fox) w_n
PPMI(w_i, c_j)
d-dimensional
feature vector
for "brown"

- Advantages to extracting low-dimensional dense representations:
  - Improve computational efficiency for end applications
  - Better visualization
  - Better performance (?)
- Perform PCA/SVD on the sparse feature matrix:

$$M^{\mathrm{PPMI}} \approx U_k \Sigma_k V_k^T$$

  Then $W^{\mathrm{SVD}} = U_k \Sigma_k$ is the context representation of each word
  (Each row is a k-dimensional feature for a word)
- This is one of the word2vec algorithm.

Generalized Low-rank Embedding

- SVD basis will minimize

$$\min_{W,V} \|M^{\mathrm{PPMI}} - WV^T\|_F^2$$

- Extensions (Glove, Google W2V, ...):

  Use different loss function (instead of $\|\cdot\|_F$)
  Negative sampling (less weights to 0s in M^{PPMI})
  Adding bias term:

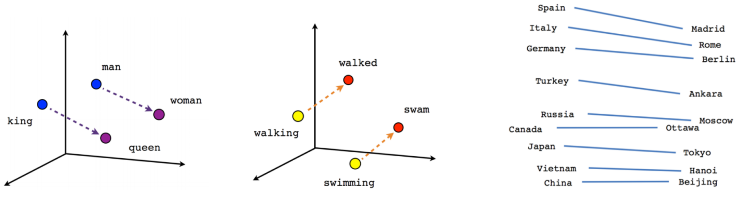
$$M^{\mathrm{PPMI}} \approx WV^T + \mathbf{b}_w\mathbf{e}^T + \mathbf{e}\mathbf{b}_c^T$$

- Details and comparisons:

  "Improving Distributional Similarity with Lessons Learned from Word Embeddings", Levy et al., ACL 2015.
  "Glove: Global Vectors for Word Representation", Pennington et al., EMNLP 2014.

Results

The low-dimensional embeddings are (often) meaningful:



Male-Female Verb tense Country-Capital

(Figure from
`https://www.tensorflow.org/tutorials/word2vec`)