Principal Component Analysis
Principal Component Analysis (PCA)

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(international)</td>
<td>2</td>
</tr>
<tr>
<td>(conference)</td>
<td>2</td>
</tr>
<tr>
<td>(machine)</td>
<td>2</td>
</tr>
<tr>
<td>(train)</td>
<td>0</td>
</tr>
<tr>
<td>(learning)</td>
<td>2</td>
</tr>
<tr>
<td>(leading)</td>
<td>1</td>
</tr>
<tr>
<td>(totoro)</td>
<td>0</td>
</tr>
</tbody>
</table>

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,

<table>
<thead>
<tr>
<th>(international)</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(conference)</td>
<td>2</td>
</tr>
<tr>
<td>(machine)</td>
<td>2</td>
</tr>
<tr>
<td>(train)</td>
<td>0</td>
</tr>
<tr>
<td>(learning)</td>
<td>2</td>
</tr>
<tr>
<td>(leading)</td>
<td>1</td>
</tr>
<tr>
<td>(totoro)</td>
<td>0</td>
</tr>
</tbody>
</table>

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

10,000 words $\Rightarrow$ 10,000$^2$ potential features
Use the bag-of-word matrix or the normalized version (TF-IDF) for a dataset (denoted by $D$):

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

- $\text{tf}(\text{doc}, \text{word})$: term frequency (word count in the document)/(total number of terms in the document)
- $\text{idf}(\text{word}, \text{Dataset})$: inverse document frequency
  \[ \log \left( \frac{\text{Number of documents}}{\text{Number of documents with this word}} \right) \]
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

Max variance

Min variance
PCA: Formulation

- Given the data $x_1, \cdots, x_n \in \mathbb{R}^d$, compute the principal vector $w$ by:

  $$w = \arg \max_{\|w\|=1} \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - w^T \bar{x})^2$$

  where $\bar{x} = \sum_i x_i / n$ is the mean.

- First, shift data so that $\hat{x}_i = x_i - \bar{x}$, so

  $$w = \arg \max_{\|w\|=1} \frac{1}{n} \sum_{i=1}^{n} (w^T \hat{x}_i)^2 = \arg \max_{\|w\|=1} \frac{1}{n} w^T \hat{X} \hat{X}^T w$$

  where each column of $\hat{X}$ is $\hat{x}_i$.

- The first principal component $w$ is the leading eigenvector of $\hat{X} \hat{X}^T$ (eigenvector corresponding to the largest eigenvalue)
PCA: Formulation

- 2nd principal component $\mathbf{w}_2$:
  - Perpendicular to $\mathbf{w}_1$
  - Again, largest variance
  - Eigenvector corresponding to the second eigenvalue

- Top $k$ principal components:
  - $\mathbf{w}_1, \cdots, \mathbf{w}_k$
  - Top $k$ eigenvectors
  - The $k$-dimensional subspace with largest variance

$$\mathcal{W} = \arg \max_{\mathcal{W} \in \mathbb{R}^{d \times k}, \mathcal{W}^T \mathcal{W} = I} \left\{ \sum_{r=1}^{k} \frac{1}{n} \mathbf{w}_r^T \hat{\mathbf{X}} \hat{\mathbf{X}}^T \mathbf{w}_r \right\}$$
PCA: illustration
PCA: Computation

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

$$U_k^T \hat{X} = \Sigma_k V_k^T \quad (k \text{ by } n \text{ matrix})$$

Each column is the $k$-dimensional features for a example

- PCA can be computed in two ways:
  - Top-$k$ SVD of $\hat{X}$
  - Top-$k$ SVD of $\hat{X}\hat{X}^T$ (explicitly form the matrix only when $d$ is small)
- Need large scale SVD solver for dense or sparse matrices.
Word2vec: Learning Word Representations
Word2vec: Motivation

- 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:

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

- Get a collection of (word, context) pairs, denoted by $D$. 
Skip-gram model

Source Text

The quick brown fox jumps over the lazy dog.

Training Samples

(the, quick)
(the, brown)
(quick, the)
(quick, brown)
(quick, fox)
(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)
(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

(Figure from http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/)
Idea 1: Use the bag-of-word model to “describe” each word

Assume we have context words \( c_1, \ldots, c_d \) 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 \( w \)

\[
\#(w, c_1), \ldots, \#(w, c_d),
\]
PMI/PPMI Representation

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

\[
\text{PMI}(w, c) = \log \left( \frac{\hat{P}(w, c)}{\hat{P}(w) \hat{P}(c)} \right) = \log \frac{\#(w, c)|D|}{\#(w)\#(c)},
\]

- \(\#(w) = \sum_c \#(w, c)\): 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:

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

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

(dimensional feature vector for “brown”)
Low-dimensional embedding (Word2vec)

- 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:

\[ M^{PPMI} \approx U_k \Sigma_k V_k^T \]

Then \[ W^{SVD} = U_k \Sigma_k \] 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

\[
\min_{W,V} \| M_{PPMI} - WV^T \|_F^2
\]

- Extensions (Glove, Google W2V, …):
  - Use different loss function (instead of \( \| \cdot \|_F \))
  - Negative sampling (less weights to 0s in \( M_{PPMI} \))
  - Adding bias term:

\[
M_{PPMI} \approx WV^T + b_w e^T + e b_c^T
\]

- 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.
The low-dimensional embeddings are (often) meaningful:

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

- Tree-based algorithms

Questions?