STA 221: LECTURE 5

KRISHNA BALASUBRAMANIAN

(University of California, Davis)

Word Representation

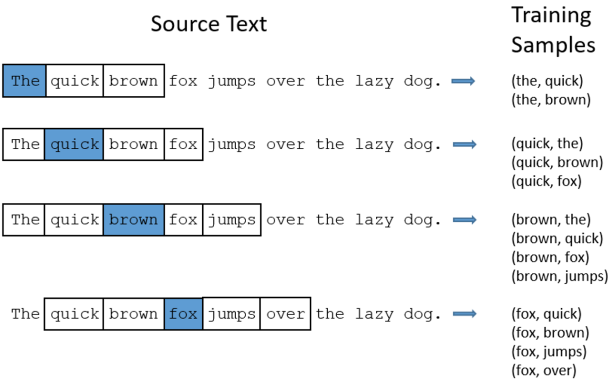
- Goal: understand the meaning of a word
- Given a large text corpus, how to learn low-dimensional features to represent a word?
- Skip-gram model:

  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/`)

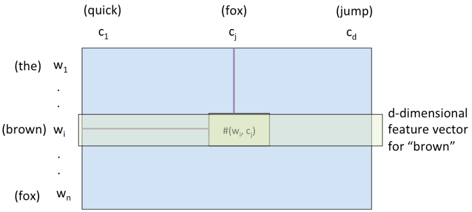
Use bag-of-word model

- 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:

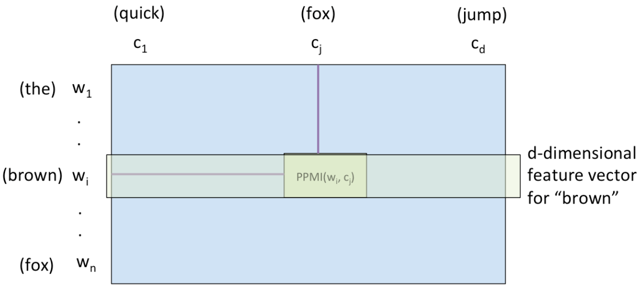
$$\mathrm{PMI}(w, c) = \log\left(\frac{\widehat{P}(w, c)}{\widehat{P}(w)\widehat{P}(c)}\right) = \log \frac{\#(w, c)|D|}{\#(w)\#(c)},$$

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

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

- 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:

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

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

- SVD basis will minimize

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

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

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

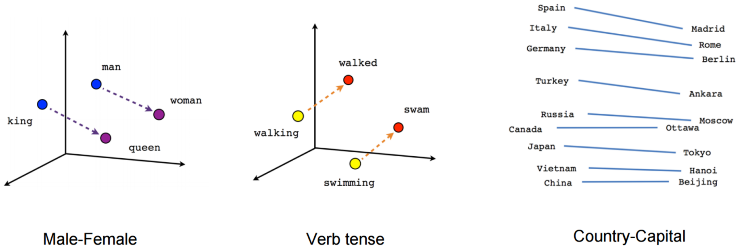
$$M^{\text{PPMI}} \approx WV^T + \mathbf{b}_w\mathbf{e}^T + \mathbf{e}\mathbf{b}_c^T$$

- 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.

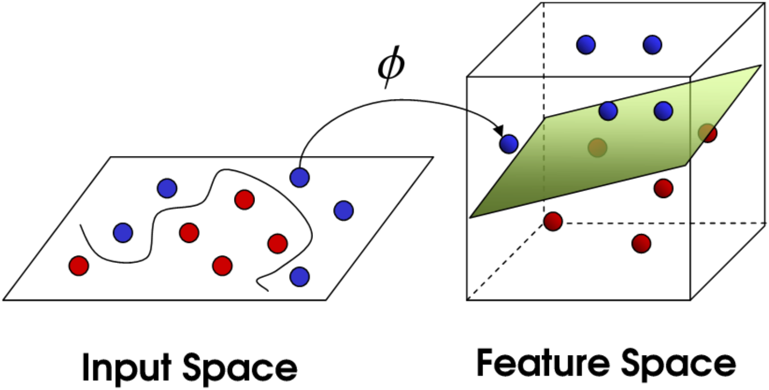
Results

The low-dimensional embeddings are (often) meaningful:



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

Kernel PCA


ϕ
Input Space
Feature Space

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

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

  where each column of $\phi(\widehat{X})$ is $\phi(X^{(i)})$
- The first principal component $\mathbf{v}_1$ is the leading eigenvector of $\frac{1}{n}\phi(\widehat{X})\phi(\widehat{X})^T$ (eigenvector corresponding to the largest eigenvalue)

Kernel PCA: Formulation

- It appears that we are actually lifting the data from already high-dimensional space to an even higher-dimensional space.

- But the so-called **kernel trick** comes to our rescue!

- Specifically, we have a kernel as follows:

$$K(X^{(i)}, X^{(j)}) = \phi(X^{(i)})^\top \phi(X^{(j)})$$

- The eigenvector computation from the previous slide could be all done with the help of this kernel trick without actually computing the mapping $\phi(X^{(i)})$ at all!

Widely used Kernel

- Gaussian kernel

$$K(X^{(i)}, X^{(j)}) = e^{-\|X^{(i)} - X^{(j)}\|^2/\gamma^2}$$

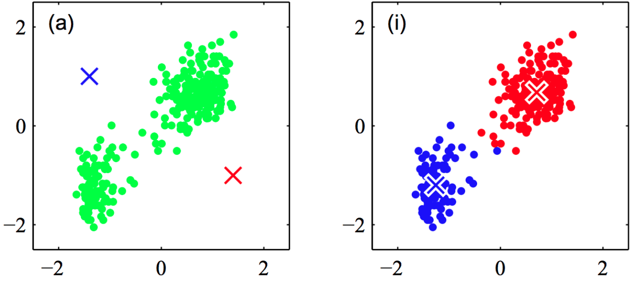
- In Lecture5.ipynb we have the command

$$KernelPCA(n\_components = 2, kernel =' rbf', gamma = 15)$$

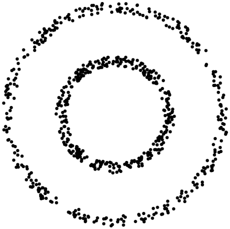
- $kernel =' rbf'$ means we are picking this kernel
- $gamma = 15$ means we are setting γ.

Clustering

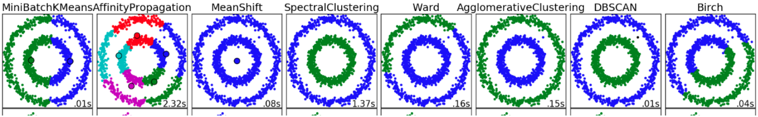
- Given $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and K (number of clusters)
- Output $A(\boldsymbol{x}_i) \in \{1, 2, \ldots, K\}$ (cluster membership)



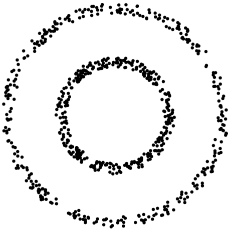
Concentric circles



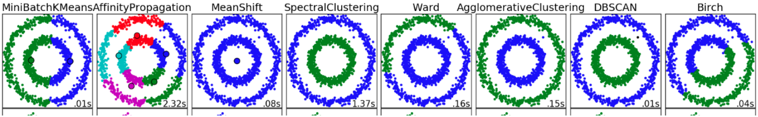
Can we split the data into two clusters?



Concentric circles



Can we split the data into two clusters?

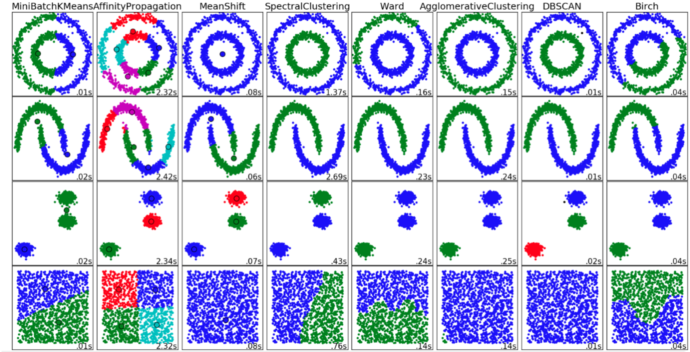


Clustering is Subjective

- Non-trivial to say one clustering is better than the other
- Each algorithm has two parts:

  Define the objective function

  Design an algorithm to minimize this objective function



K-means Objective Function

- Partition dataset into $C_1, C_2, \ldots, C_K$ to minimize the following objective:

$$J = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_k} \|\boldsymbol{x} - \boldsymbol{m}_k\|_2^2,$$

  where $\boldsymbol{m}_k$ is the mean of C_k.

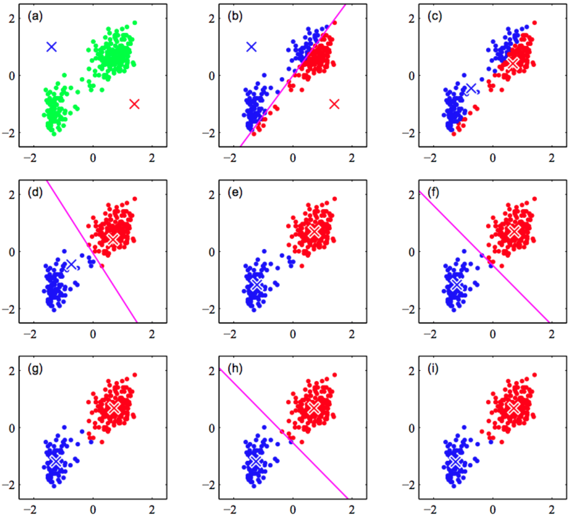
- Multiple ways to minimize this objective

  Hierarchical Agglomerative Clustering

  K-means Algorithm (Today)

  ...

K-means Algorithm



K-means Algorithm

- Re-write objective:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{m}_k\|_2^2,$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \text{ if and only if } \boldsymbol{x}_n \in C_k$$

- Alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{m}_k\}$

  Fix $\{\boldsymbol{m}_k\}$ and update $\{r_{nk}\}$

  Fix $\{r_{nk}\}$ and update $\{\boldsymbol{m}_k\}$

- Step 0: Initialize $\{\boldsymbol{m}_k\}$ to some values
- Step 1: Fix $\{\boldsymbol{m}_k\}$ and minimize over $\{r_{nk}\}$:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{m}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- Step 2: Fix $\{r_{nk}\}$ and minimize over $\{\boldsymbol{m}_k\}$:

$$\boldsymbol{m}_k = \frac{\sum_n r_{nk} \boldsymbol{x}_n}{\sum_n r_{nk}}$$

- Step 3: Return to step 1 unless stopping criterion is met

- ▷ Step 0: Initialize $\{\boldsymbol{m}_k\}$ to some values
- ▷ Step 1: Fix $\{\boldsymbol{m}_k\}$ and minimize over $\{r_{nk}\}$:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{m}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- ▷ Step 2: Fix $\{r_{nk}\}$ and minimize over $\{\boldsymbol{m}_k\}$:

$$\boldsymbol{m}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

- ▷ Step 3: Return to step 1 unless stopping criterion is met

- ▷ Step 0: Initialize $\{\boldsymbol{m}_k\}$ to some values
- ▷ Step 1: Fix $\{\boldsymbol{m}_k\}$ and minimize over $\{r_{nk}\}$:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{m}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- ▷ Step 2: Fix $\{r_{nk}\}$ and minimize over $\{\boldsymbol{m}_k\}$:

$$\boldsymbol{m}_k = \frac{\sum_n r_{nk} \boldsymbol{x}_n}{\sum_n r_{nk}}$$

- ▷ Step 3: Return to step 1 unless stopping criterion is met

K-means Algorithm

- ▷ Step 0: Initialize $\{\boldsymbol{m}_k\}$ to some values
- ▷ Step 1: Fix $\{\boldsymbol{m}_k\}$ and minimize over $\{r_{nk}\}$:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{m}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- ▷ Step 2: Fix $\{r_{nk}\}$ and minimize over $\{\boldsymbol{m}_k\}$:

$$\boldsymbol{m}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

- ▷ Step 3: Return to step 1 unless stopping criterion is met

Equivalent to the following procedure:

- ▷ Step 0: Initialize centers $\{\boldsymbol{m}_k\}$ to some values
- ▷ Step 1: Assign each $\boldsymbol{x}_n$ to the nearest center:

$$A(\boldsymbol{x}_n) = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{m}_j\|_2^2$$

  Update clusters:

$$C_k = \{\boldsymbol{x}_n : A(\boldsymbol{x}_n) = k\} \;\; \forall k = 1, \ldots, K$$

- ▷ Step 2: Calculate mean of each cluster C_k:

$$\boldsymbol{m}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x}_n \in C_k} \boldsymbol{x}_n$$

- ▷ Step 3: Return to step 1 unless stopping criterion is met

More on K-means Algorithm

- Always decrease the objective function for each update
- Objective function will keep unchanged when step 1 doesn't change cluster assignment $\Rightarrow$ Converged
- May not converge to global minimum

  Sensitive to initial values
- Kmeans++: A better way to initialize the clusters

More on K-means Algorithm

- Always decrease the objective function for each update
- Objective function will keep unchanged when step 1 doesn't change cluster assignment ⇒ Converged
- May not converge to global minimum

  Sensitive to initial values
- Kmeans++: A better way to initialize the clusters

More on K-means Algorithm

- Always decrease the objective function for each update
- Objective function will keep unchanged when step 1 doesn't change cluster assignment $\Rightarrow$ Converged
- May not converge to global minimum

  Sensitive to initial values
- Kmeans++: A better way to initialize the clusters