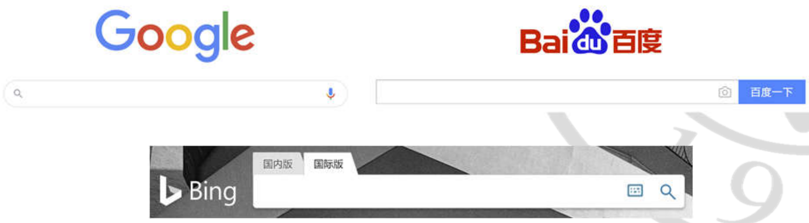
# 智能系统原理与开发
# 第07章 文本检索与分类

李斌
复旦大学 计算机科学技术学院

## High-dimensional & Sparse Data

- In real-world intelligent systems, data are usually high-dimensional and very sparse
  - Text data: Bag-of-words representation
  - Image data: Visual dictionary representation
  - Rating data: Spare adjacent matrix representation
  - etc.



## An Example of Text Data Representation

- Each text (e.g., document, paper, webpage, etc.) can be represented by a set of $n$-grams
- For example, a sentence "This is a short sentence"
  - $n = 1$: { "This", "is", "a", "short", "sentence" }
  - $n = 2$: { "This is", "is a", "a short", "short sentence" }
  - $n = 3$: { "This is a", "is a short", "a short sentence" }
  - In practice, it is common to adopt $n \geq 5$
- Using $n$-grams will lead to extremely high-dimensional feature vectors: $D = \left(10^5\right)^5 = 10^{25} = 2^{83}$
- In current practice, $D = 2^{64}$ seems sufficient

## Computation of Similarity

- Computation of data distance is essential in machine learning and thus big data analytics
  - Nearest Neighbor (NN) search for retrieval

$$x^* = \arg\min_{x_n \in S} ||x_q - x_n||^2$$

  - Similarity (Distance) based clustering

$$\min \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \, ||x_n - \mu_k||^2$$

- Computation of inner product is common in linear kernel methods
  - Support Vector Machine
  - Gaussian Process
  - Kernel PCA

# Linear Kernel Methods

- Recall the objective function for a linear regression

$$\min_{w} \frac{1}{N}\sum_{n=1}^{N}(y_n - w^\top x_n)^2 + \lambda||w||^2$$

- Set the derivatives of the objective w.r.t. $w$ to zero

$$\frac{\partial J(w)}{\partial w} = \frac{2}{N}\sum_{n=1}^{N}(w^\top x_n - y_n)x_n + 2\lambda w = 0$$

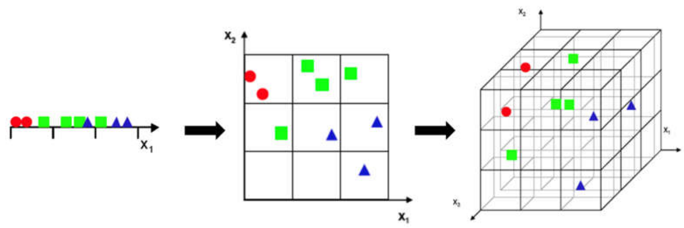
$$\Rightarrow w = \frac{1}{\lambda N}\sum_{n=1}^{N}(y_n - w^\top x_n)x_n = \sum_{n=1}^{N}\alpha_n x_n$$

- Substitute $w = \sum_{n=1}^{N}\alpha_n x_n$ into the objective function we can obtain the dual representation

$$\min_{\alpha}(Y - XX^\top\alpha)^\top(Y - XX^\top\alpha) + \lambda\alpha^\top XX^\top\alpha$$

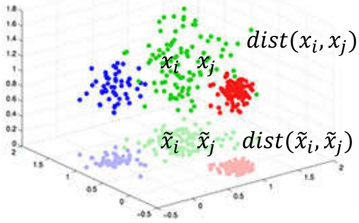
# Challenges with High-dimensional Data

- High computational cost: Increase matrix operations and increase search space
- High storage cost: Too large to store high-dimensional data and difficult to load the big model into memory
- The curse of dimensionality: The volume of the space increases so fast that the available data become sparse



# Random Projection

- Recall that dimensionality reduction methods (e.g., PCA) are themselves learning based
- Are there methods that is able to generate a projection matrix without learning and satisfies $dist(x_i, x_j) \approx dist(\tilde{x}_i, \tilde{x}_j)$
  - $\tilde{x}_n = H^\top x_n$, where $H \in R^{D \times d}$ ($d \ll D$) is a projection matrix
  - $dist(\cdot,\cdot)$ is a distance function (or similarity measure)




# Random Projection: JL Lemma

- **Johnson-Lindenstrauss Lemma**[1]: Given $0 < \epsilon < 1$, a set $X$ of $N$ points in $R^D$, and a number $d > 8 \ln N / \epsilon^2$, there exists a linear mapping $H: R^D \rightarrow R^d$ such that for all $x_i, x_j \in X$

$$(1-\epsilon)||x_i - x_j||^2 \leq ||Hx_i - Hx_j||^2 \leq (1+\epsilon)||x_i - x_j||^2$$

- The JL-lemma states that a small set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved.

- The JL-Lemma also holds for dot products

$$x_i^\top x_j - \epsilon \leq (Hx_i)^\top Hx_j \leq x_i^\top x_j + \epsilon$$

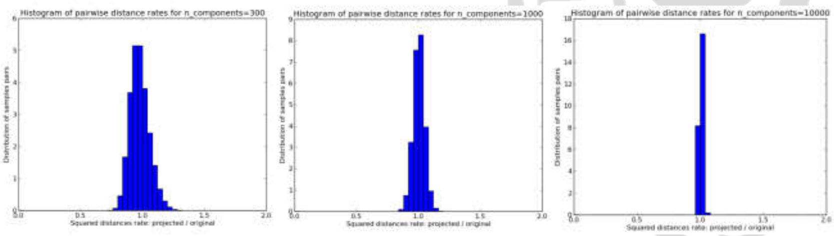
[1] Johnson & Lindenstrauss (1984). "Extensions of Lipschitz mappings into a Hilbert space".

# The Random Projection Algorithm

- Random Projection: Fast, efficient and distance-preserving dimensionality reduction technique!
- The core idea behind random projection is given in the Johnson-Lindenstrauss lemma
- Canonical (Gaussian) Random Projection:
  - Construct a random matrix $H' \in R^{D \times d}$ ($d \ll D$) by picking the entries from a univariate Gaussian $N(0, \sigma^2)$
  - Orthonormalize the rows of $H'$ to get $H$
  - Project a data point $x_n$ in the original $D$-dimensional space into the new $d$-dimensional space: $\tilde{x}_n = H^\top x_n$

# Random Projection Example

- Apply Gaussian random projection to the 20-Newsgroups dataset with different configuration of dimensions
  - From 100.000 features to 300 (0.3%)
  - From 100.000 features to 1.000 (1%)
  - From 100.000 features to 10.000 (10%)




[1] https://www.cs.toronto.edu/~duvenaud/talks/random-kitchen-sinks-tutorial.pdf

## Document Classification

- In a document classification task, the input to the machine learning algorithm is raw text, represented by a bag of words (BoW) representation:
  - The individual tokens are extracted and counted, and each distinct token in the training set defines a feature.
  - The BoW for a set of documents is regarded as a term-document matrix where each row is a single document, and each column is a single feature (word).
  - The entry $i, j$ in such a matrix captures the frequency (or weight) of the $j$th term of the vocabulary in document $i$.
  - The common approach is to construct a dictionary of the training set, and use that to map words to indices.

## Document Classification

- An example of BoW representation of documents
  - Document 1: "He studies machine learning".
  - Document 2: "Machine learning is interesting".
  - Document 3: "Machine learning supports big data".

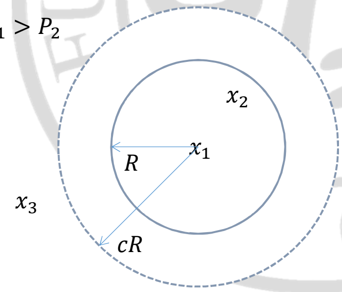
| | Dictionary | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | He | studies | machine | learning | is | interesting | supports | big | data |
| Doc1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Doc2 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| Doc3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

- The problem with this process is that such dictionaries take up a large amount of storage space and grow in size as the training set grows.

# Locality-Sensitive Hashing

- Locality-Sensitive Hashing (LSH) hashes input items so that similar items map to the same "buckets" with high probability
  - If $d(x_1, x_2) \leq R$, then $h(x_1) = h(x_2)$ with high probability at least $P_1$
  - If $d(x_2, x_3) \geq cR$, then $h(x_2) = h(x_3)$ with low probability at most $P_2$
  - An LSH family is interesting only $P_1 > P_2$
- Alternative definition of LSH

$$E_h[h(x_i) = h(x_j)] = sim(x_i, x_j)$$




# LSH: SimHash

- SimHash is designed to approximate the cosine similarity $\cos(\theta(x_i, x_j))$ between vectors $x_i$ and $x_j$.
- SimHash is used by the Google Crawler to find near duplicate pages
- Given an input vector $x_i$ and a random hyperplane specified by a normal unit vector $r$, the SimHash function is defined as $h(x_i) = \operatorname{sgn}(r^\top x_i)$
- Randomly choose multiple hyperplanes and the limit of the collision ratio equals to the probability of hyperplane falling in the angle between the two vectors $\frac{\theta(x_i, x_j)}{\pi}$

$$\frac{1}{N}\sum_k 1(h_k(x_i) = h_k(x_j)) \xRightarrow{k\to\infty} E_h[h(x_i) = h(x_j)] = 1 - \frac{\theta(x_i, x_j)}{\pi}$$

[1] Charikar (2002), Similarity Estimation Techniques from Rounding Algorithms.

# LSH: MinHash

- MinHash is designed to approximate the Jaccard similarity $J(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$ between sets $S_i$ and $S_j$.
- MinHash is the first position of element after a random permutation $h(S_i) = \min(\pi(S_i))$.
- Property of MinHash

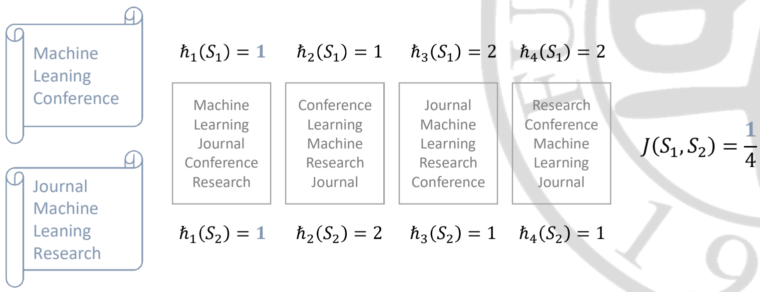
$$J(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = E_\pi[1(\min(\pi(S_i)) = \min(\pi(S_j)))]$$

- Empirically use $K$ independent random permutations for approximation

$$J(S_i, S_j) \approx \frac{1}{K} \sum_{k=1}^{K} 1(\min(\pi_k(S_i)) = \min(\pi_k(S_j)))$$

# MinHash Example

- MinHash based text classification
  - Represent each text as a bag-of-words
  - Compute pairwise Jaccard similarities based on MinHashes
  - Construct the kernel matrix



## Summary

- Random Projection → $E[\langle Hx_i, Hx_j\rangle] = \langle x_i, x_j\rangle$
  - Johnson-Lindenstrauss transform
  - Gaussian random projection
  - etc.
- Locality-Sensitive Hashing → $E_h[h(x_i) = h(x_j)] = sim(x_i, x_j)$
  - SimHash
  - MinHash
  - etc.

## Kernel Methods

- We can efficiently embed high-dimensional data satisfying $E[\langle Hx_i, Hx_j\rangle] = \langle x_i, x_j\rangle$ or $E_h[h(x_i) = h(x_j)] = sim(x_i, x_j)$ now.
- Recall the linear kernel methods introduced before
  - The classifier $f(x) = w^\top x = \sum_{n=1}^{N} \alpha_n x_n^\top x$
  - The objective $\min_\alpha (Y - XX^\top\alpha)^\top(Y - XX^\top\alpha) + \lambda\alpha^\top XX^\top\alpha$
- Only the inner product appears in both the objective function and the classifier
  - No need to know the exact form of $x$
  - The inner product can be replaced by any (approximate) similarity measure

# Kernel Methods

- Replace the inner product by the kernel function
  - The classifier becomes $f(x) = w^\top x = \sum_{n=1}^{N} \alpha_n \kappa(x_n, x)$
  - The objective becomes $\min_{\alpha} (Y - K\alpha)^\top (Y - K\alpha) + \lambda \alpha^\top K \alpha$, where $K_{i,j} = \kappa(x_i, x_j)$
- Now we only need to let the kernel function be
  - Random projection: $\kappa(x_i, x_j) = \langle H x_i, H x_j \rangle$
  - SimHash: $\kappa(x_i, x_j) = \frac{1}{K} \sum_{k=1}^{K} 1(h_k(x_i) = h_k(x_j))$
  - MinHash: $\kappa(x_i, x_j) = \frac{1}{K} \sum_{k=1}^{K} 1(\min(\pi_k(S_i)) = \min(\pi_k(S_j)))$
  - etc.
- A valid kernel – The kernel matrix $K$ must be positive semi-definite

# Thanks

Email: libin@fudan.edu.cn