Big Data Analytics & Applications

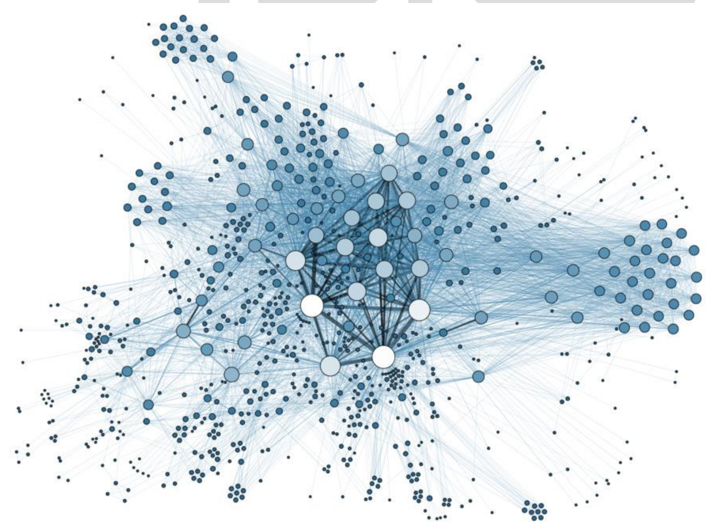
Bin Li

School of Computer Science

Fudan University

High-dimensional & Sparse Data

- In real-world BDA scenarios, 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
  - Network data: Spare adjacent matrix representation



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

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

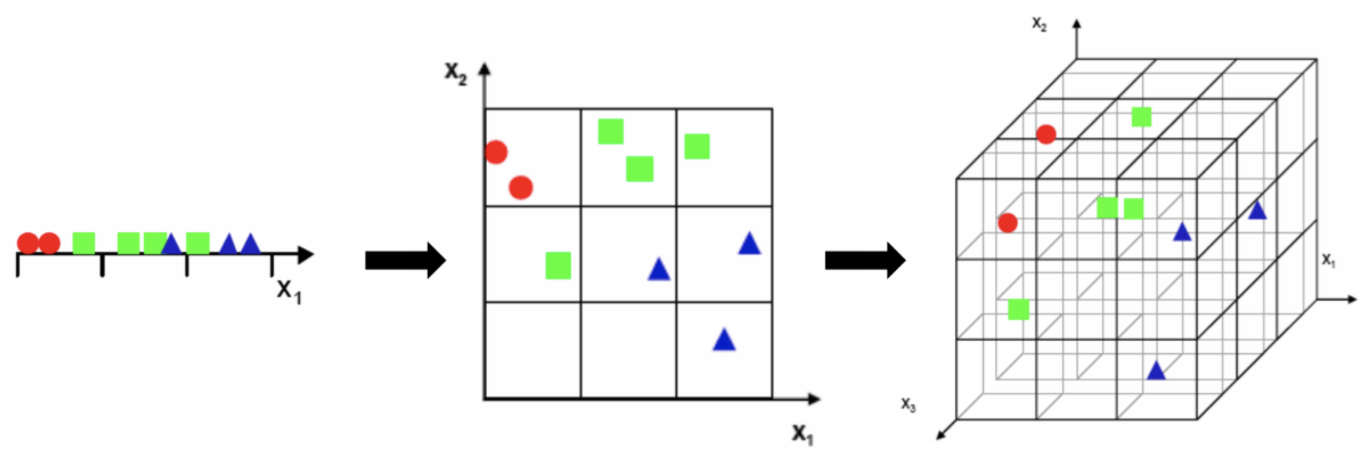
$$\Rightarrow w = \frac{1}{\lambda N}\sum_{n=1}^{N}(y_n - w^\top x_n)x_n = \sum_{n=1}^{N}\alpha_n x_n$$

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

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

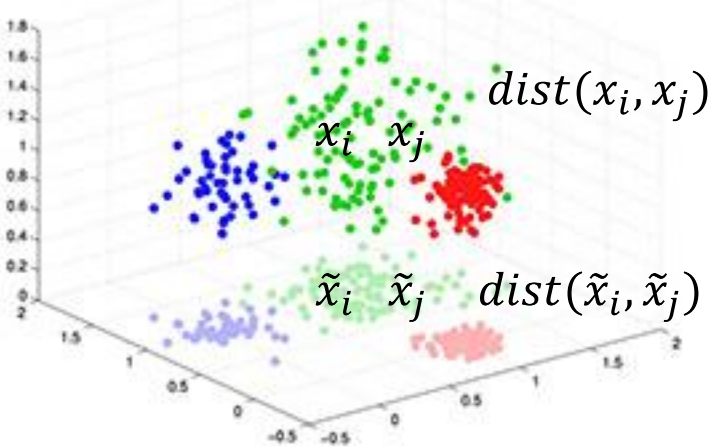
Challenges with High-dimensional Data

- 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".

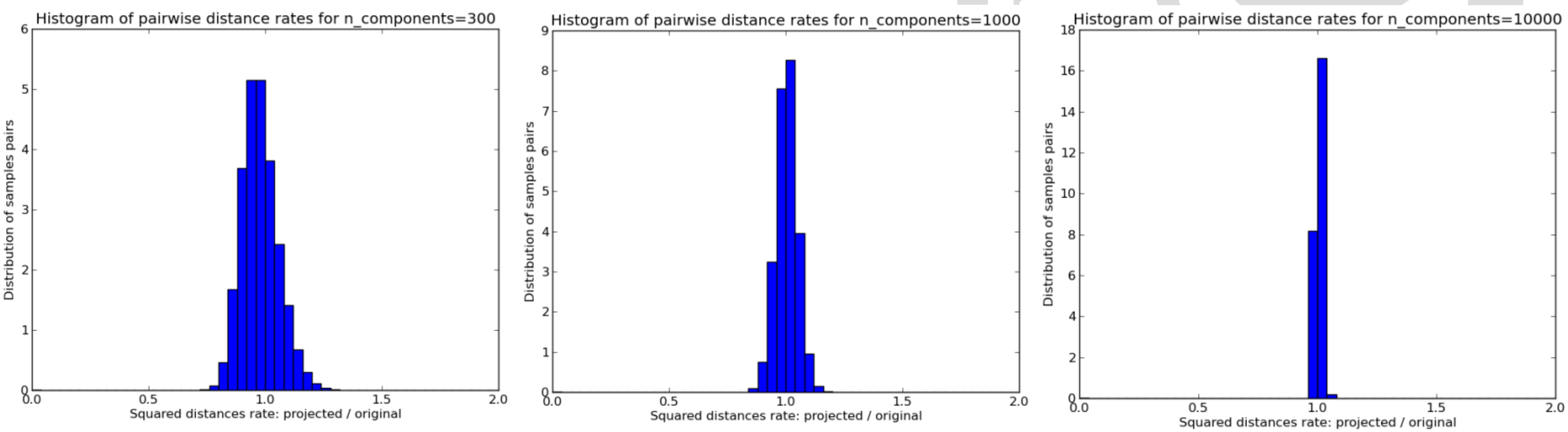
The Random Projection Algorithm

- 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'
  - Project a data point 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

Variants of Random Projection

- Some sparse variations of random projection are more efficient than the canonical one. The entries in the random projection matrix can be generated as

$$H_{i,j} = \begin{cases} +1 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/2 \end{cases}$$

$$H_{i,j} = \begin{cases} N(0,1/p) & \text{with probability } p \\ -1 & \text{with probability } 1-p \end{cases}$$

$$H_{i,j} = \begin{cases} +1 & \text{with probability } 1/6 \\ -1 & \text{with probability } 1/6 \\ 0 & \text{with probability } 2/3 \end{cases}$$

$$H_{i,j} = \begin{cases} +1 & \text{with probability } p \\ -1 & \text{with probability } p \\ 0 & \text{with probability } 1-2p \end{cases}$$

- Random projection can be applied to almost all typical machine learning algorithms as an approximate solution.

[1] Ailon and Chazelle (2006), Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform.
[2] Achlioptas (2003), Database-friendly random projections: Johnson-Lindenstrauss with binary coins.
[3] Matousek (2008), On variants of the Johnson–Lindenstrauss lemma.

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 jth 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.

Feature Hashing

- Instead of maintaining a growing dictionary, feature hashing can be used to build a vector of a pre-defined length by applying two hash functions to the features

function feature_hashing(S: BoW represented features, d: integer):
  x := **new** vector[d]
  for $f \in S$:
    h := hash(f)
    idx := h mod d
    if $\xi(f)$ == 1:
      x[idx] += 1
    else:
      x[idx] -= 1
  return x

[1] Weinberger, et al. (2009), Feature Hashing for Large Scale Multitask Learning.
[2] Attenberg, et al. (2009), Collaborative spam filtering with the hashing trick.

Feature Hashing

- Feature hashing can also be viewed as random projection
  - Only $+1/0/-1$ in H where $+1/-1$ has same probability
  - Each column of H only has one nonzero entry – each feature (column) can only be assigned to one bucket (row)

	He	studies	machine	learning	is	interesting	supports	big	data
Bucket1	0	-1	0	1	0	0	0	0	0
Bucket2	0	0	0	0	0	-1	0	1	0
Bucket3	1	0	0	0	-1	0	0	0	1
Bucket4	0	0	1	0	0	0	-1	0	0

- The inner products in the hashed space are unbiased

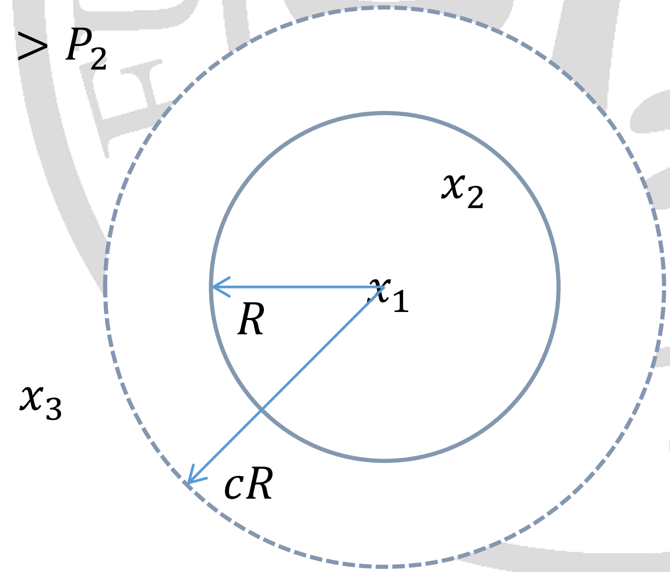
$$E\left[\langle Hx_i, Hx_j\rangle\right] = \langle x_i, x_j\rangle$$

Locality-Sensitive Hashing

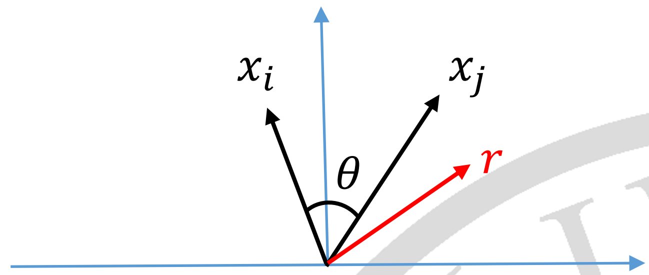
- Locality-Sensitive Hashing (LSH) hashes input items so that similar items map to the same "buckets" with high probability
  - If $d(x_1, x_2) \leq R$, then $h(x_1) = h(x_2)$ with high probability at least P_1
  - If $d(x_2, x_3) \geq cR$, then $h(x_2) = h(x_3)$ with low probability at most P_2
  - 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 $h(x_i) = \mathrm{sgn}(r^\top x_i)$
- 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 $\frac{\theta(x_i, x_j)}{\pi}$

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

[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

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

- 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

Machine
Leaning
Conference

Journal
Machine
Leaning
Research

$\hbar_1(S_1) = \mathbf{1}$	$\hbar_2(S_1) = 1$	$\hbar_3(S_1) = 2$	$\hbar_4(S_1) = 2$
Machine Learning Journal Conference Research	Conference Learning Machine Research Journal	Journal Machine Learning Research Conference	Research Conference Machine Learning Journal
$\hbar_1(S_2) = \mathbf{1}$	$\hbar_2(S_2) = 2$	$\hbar_3(S_2) = 1$	$\hbar_4(S_2) = 1$

$$J(S_1, S_2) = \frac{\mathbf{1}}{4}$$

Summary

- Random Projection → $E\left[\langle Hx_i, Hx_j\rangle\right] = \langle x_i, x_j\rangle$
  - Johnson-Lindenstrauss transform
  - Gaussian random projection
  - Sparse random projection
  - etc.

- Locality-Sensitive Hashing → $E_h\left[h(x_i) = h\left(x_j\right)\right] = sim(x_i, x_j)$
  - Feature hashing
  - SimHash
  - MinHash
  - etc.

Kernel Methods

- We can efficiently embed high-dimensional data satisfying $E\left[\langle Hx_i, Hx_j\rangle\right] = \langle x_i, x_j\rangle$ or $E_h\left[h(x_i) = h(x_j)\right] = sim(x_i, x_j)$ 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 $x^\top x$
  - The inner product can be replaced by any (approximate) similarity measure

Kernel Methods

- Replace inner product $x_n^\top x$ by kernel function $\kappa(x_n, x)$
  - 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: $\kappa(x_i, x_j) = \langle Hx_i, Hx_j \rangle$
  - Feature hashing: $\kappa(x_i, x_j) = \langle Hx_i, Hx_j \rangle$
  - 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

Project: Document Classification

- Dataset:
  - Public available text classification datasets (e.g., UCI Machine Learning Repository https://archive.ics.uci.edu)
  - Or text data collected by yourself
- Method:
  - Use random projection or locality-sensitive hashing techniques to deal with high-dimensional text data
  - And use supervised learning model (e.g., logistic regression or SVM) to train and predict the test data
- Experiments:
  - Compare results on hashed and non-hashed text data
  - And discuss the observations from the experimental results

Thanks

Email: libin@fudan.edu.cn