15-853:Algorithms in the Real World

Announcement:

- HW3 due tomorrow (Nov. 20) 11:59pm
- There is recitation this week:
  - HW3 solution discussion and a few problems
- Scribe volunteer
- Exam: Nov. 26
  - 5-pages of cheat sheet allowed
  - Need not use all 5 pages of course!
  - At least one question from each of the 5 modules
    - Will test high level concepts learned

15-853:Algorithms in the Real World

Announcements: Project report (reminder):

- Style file available on the course webpage:
  - 5 page, single column
  - Appendices (might not read them)
  - References (no limit)
- Write carefully so that it is understandable. This carries weight.
- Same format even for surveys: you need to distill what you read, compare across papers and bring out the commonalities and differences, etc.
- For a research project, in case you don't have any new results, mention what all you tried even if it didn’t work out.

15-853:Algorithms in the Real World

Hashing:

- Concentration bounds
- Load balancing: balls and bins
- Hash functions
- Data streaming model
- Hashing for finding similarity (cont)

Dimensionality Reduction:

- Johnson-Lindenstrauss
- Principal Component Analysis

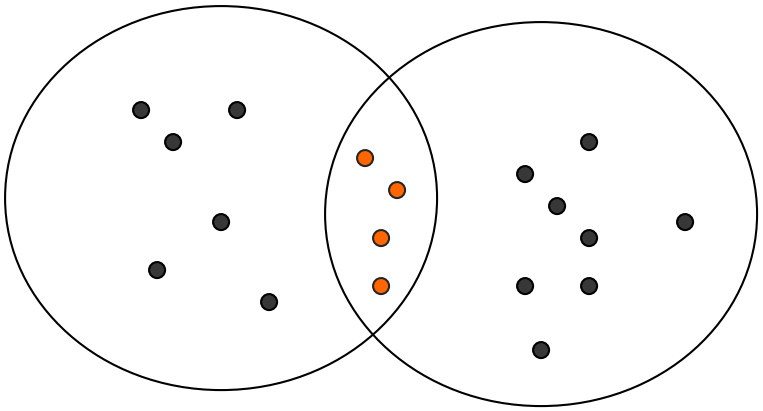
Recap: Defining Similarity of Sets

Many ways to define similarity.

One similarity metric, “distance”, for sets

Jaccard similarity

$$\mathrm{SIM}(A, B) := \frac{A \cap B}{A \cup B}$$



4 common
18 total

SIM(A,B) = 4/18 = 2/9

Jaccard distance is 1 – SIM(A, B)

Recap: Characteristic Matrix of Sets

Element num	Set1	Set2	Set3	Set4
0	1	0	0	1
1	0	0	1	0
2	0	1	0	1
3	1	0	1	1
4	0	0	1	0
…				

Stored as a sparse matrix in practice.

Example from "Mining of Massive Datasets" book by Rajaraman and Ullman

Recap: Minhashing

Minhash(π) of a set is the number of the row (element) with first non-zero in the **permuted order π**.

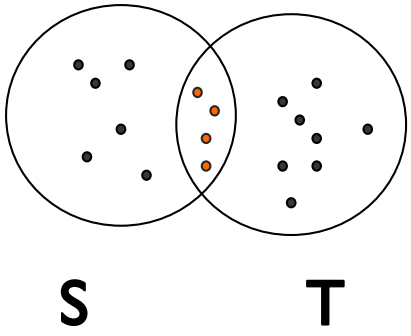
Element num	Set1	Set2	Set3	Set4
1	0	0	1	0
4	0	0	1	0
0	1	0	0	1
3	1	0	1	1
2	0	1	0	1
…				

Π=(1,4,0,3,2)

Recap: Minhash and Jaccard similarity

Theorem:

P(minhash(S) = minhash(T)) = SIM(S,T)



Representing collection of sets: Minhash signature

Let h_1, h_2, …, h_n be different minhash functions (i.e., independent permutations).

Then **signature** for set S is:

SIG(S) = [h_1(S), h_2(S), …, h_n(S)]

Recap: Minhash signature

Signature for set S is:

$SIG(S) = [h_1(S), h_2(S), \ldots, h_n(S)]$

Signature matrix:

Rows are minhash functions

Columns are sets

SIM(S,T) ≈ fraction of coordinates where SIG(S) and SIG(T) are the same

Recap: LSH requirements

A good LSH hash function will divide input into large number of **buckets.**

To find nearest neighbors for a query item q, we want to only compare with items in the bucket hash(q): "**candidates**".

If two A and B are similar, we want the probability that hash(A) = hash(B) be high.

- *False positives*: sets that are not similar, but are hashed into same bucket.
- *False negatives*: sets that are similar, but hashed into different buckets.

Recap: LSH based on minhash

We will consider a specific form of LSH designed for documents represented by shingle-sets and minhahsed to short signatures.

Idea:

divide the signature matrix rows into **b** bands of **r** rows

hash the columns in each band with a basic hash-function

→ each band divided to buckets [i.e., a hashtable for each band]

Recap: LSH based on minhash

Idea:

divide the signature matrix rows into **b** bands of **r** rows

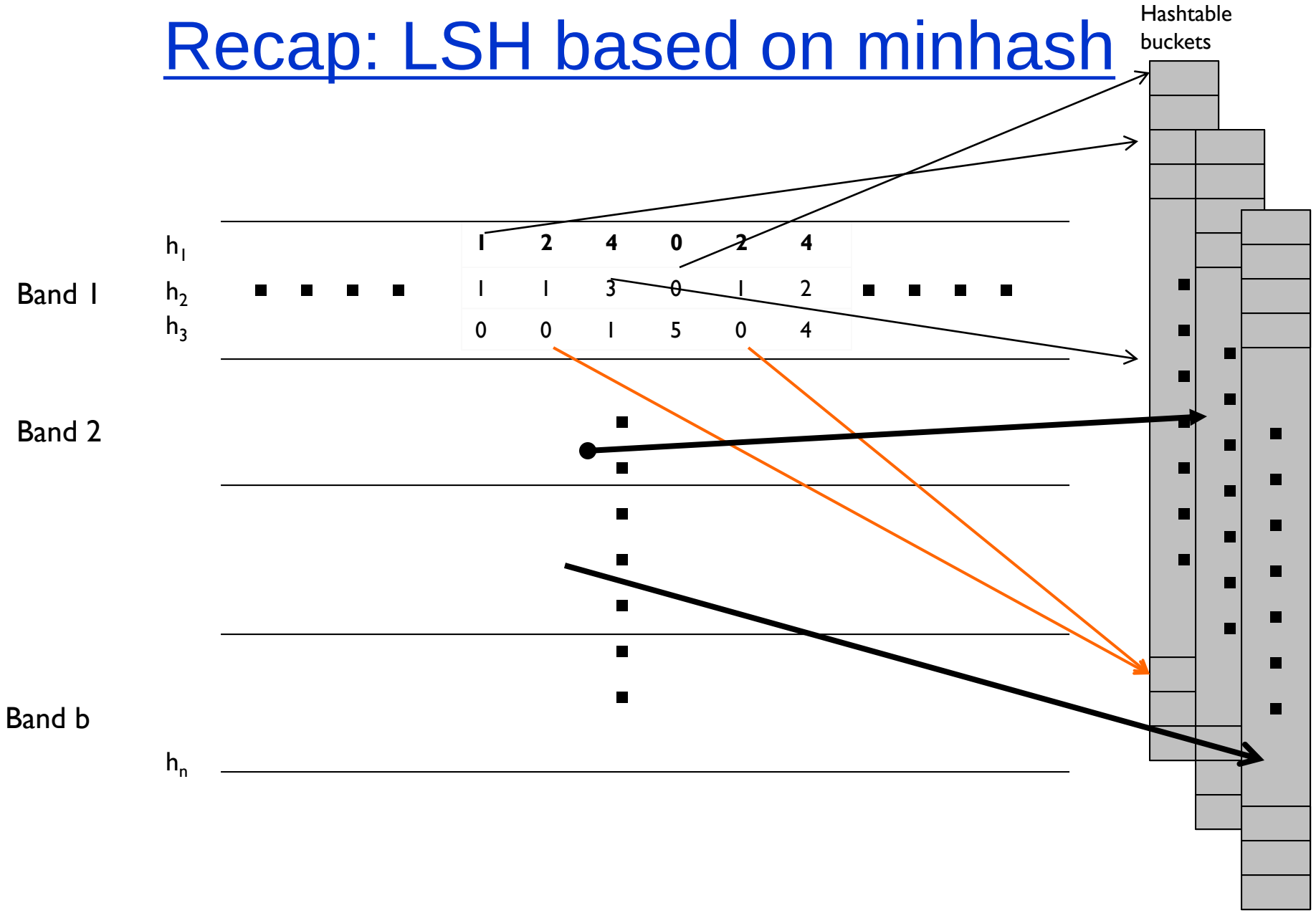
hash the columns in each band with a basic hash-function

→ each band divided to buckets [i.e., a hashtable for each band]

If sets S and T have same values in a band,
they will be hashed into the same bucket in that band.

For nearest-neighbor, the candidates are
the items in the same bucket as query item, in each band.

Recap: LSH based on minhash



Analysis

Consider the probability that we find T with query document Q

Let

s = SIM(Q,T) = P{ $h_i(Q) = h_i(T)$ }

b = # of bands

r = # rows in one band

What is the probability that rows of signature matrix agree for columns Q and T in one band?

Analysis

s = SIM(Q,T)
b = # of bands
r = # rows in one band

Probability that Q and T agree on all rows in a band

s^r

Probability that disagree on at least one row

$1 - s^r$

Probability that signatures do not agree on any of the bands:

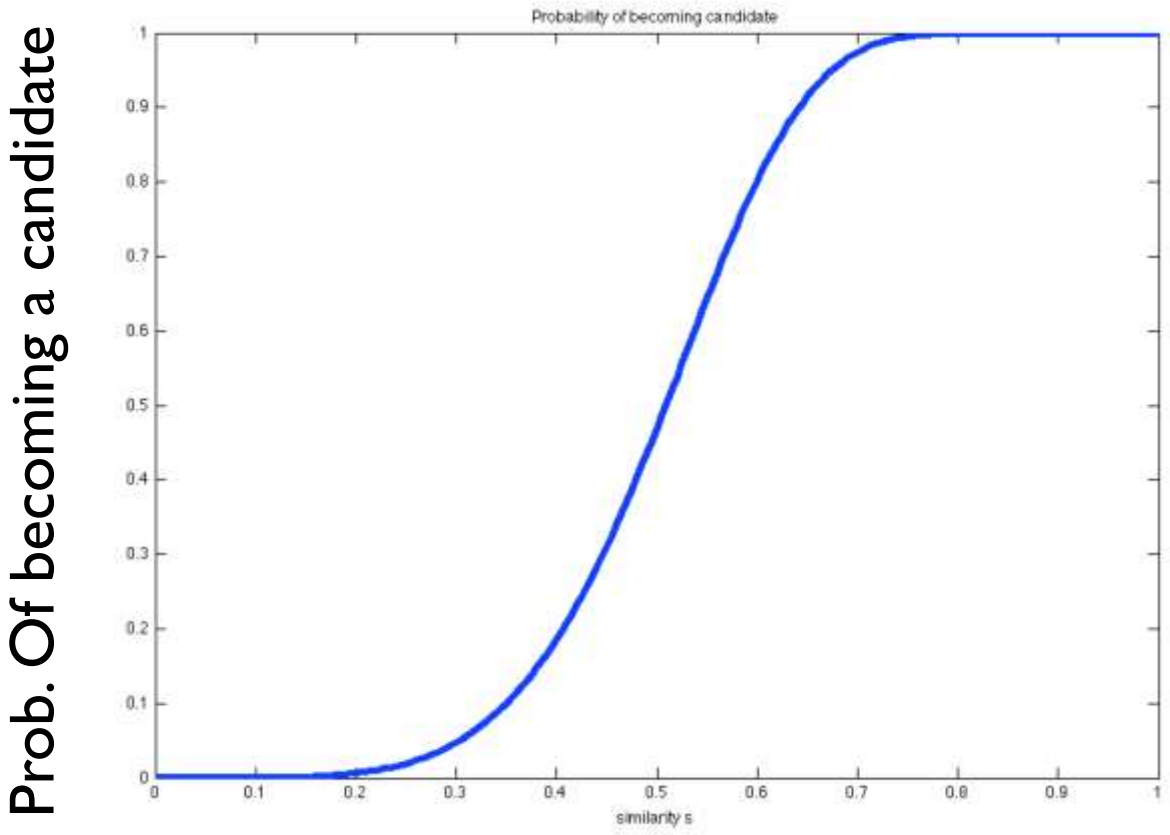
$(1 - s^r)^b$

Probability that T will be chosen as candidate: ____

$1- (1 - s^r)^b$

S-curve

r = 5
b = 20



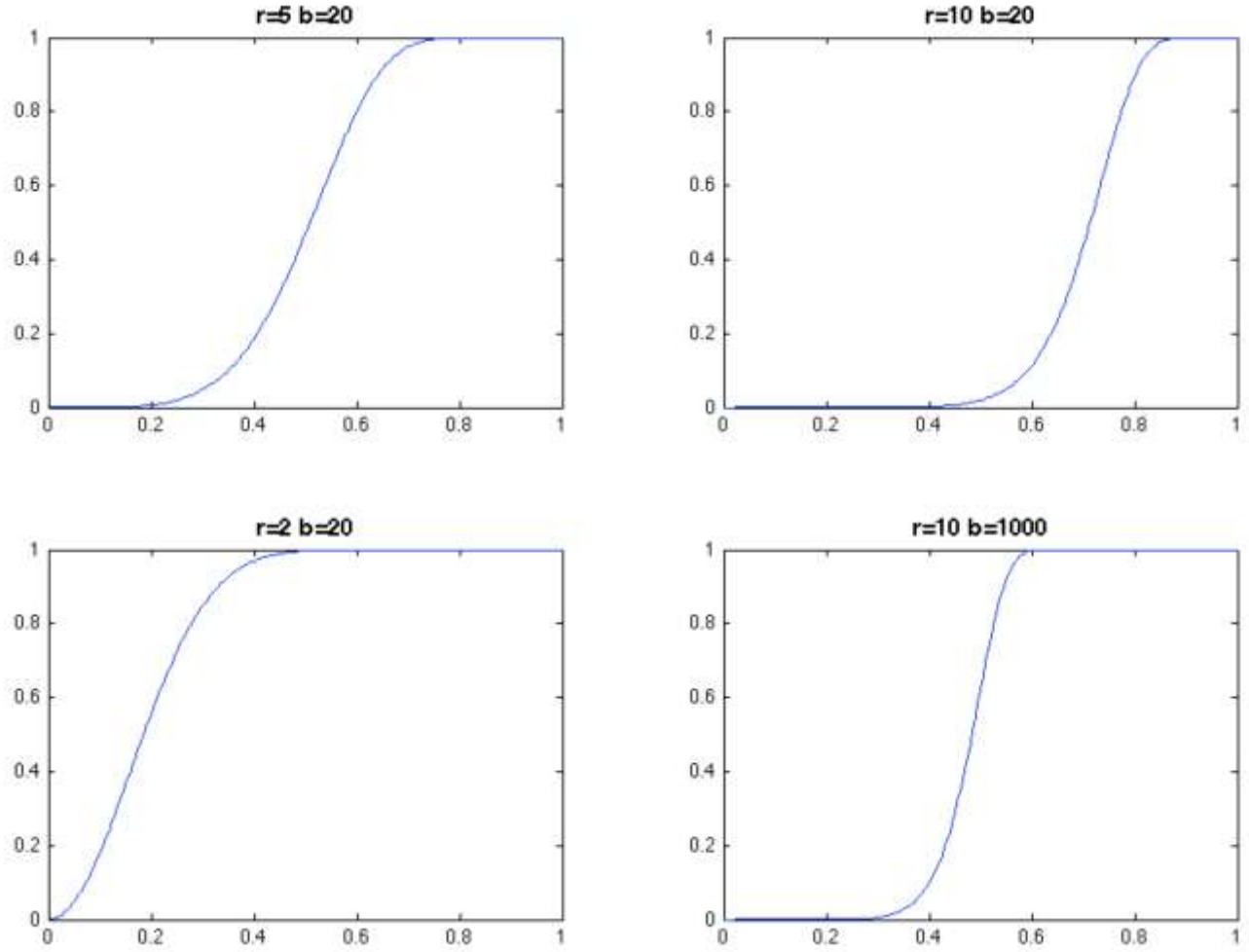
Prob. Of becoming a candidate

Jaccard similarity

Approx. value of the threshold: (1/b)^{1/r}

S-curves

r and b are parameters of the system: trade-offs?



Summary

To build a system that quickly finds similar documents from a corpus:

1. Pick a value of k to represent each document in terms of k-shingles
2. Generate minhash signature matrix for the corpus
3. Pick a threshold t for similarity; choose b and r using this threshold such that b*r = n (length of minhash signatures)
4. Divide signature matrix into bands
5. Store each band-column into a hashtable
6. To find similar documents, compare to candidate documents for each band only in the same bucket (using minhash signatures or the docs themselves) .

More About Locality Sensitive Hashing

Has been an active research area.

Different distance metrics and compatible locality sensitive hash functions:

- Euclidean distance
- Cosine distance
- Edit distance (strings)
- Hamming distance
- Jaccard distance (= 1 – Jaccard similarity)

More About Locality Sensitive Hashing

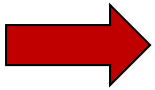
Leskovec, Rajaraman, Ullman:
Mining of Massive Datasets (available for download)

CACM technical survey article by Andoni and Indyk
and an implementation by Alex Andoni.

15-853:Algorithms in the Real World

Hashing:

- Concentration bounds
- Load balancing: balls and bins
- Hash functions
- Data streaming model
- Hashing for finding similarity



Dimensionality Reduction:

- Johnson-Lindenstrauss Transform
- Principal Component Analysis

High dimensional vectors

Common in many real-world applications

E.g.,: Documents, Movie or product ratings by users, gene expression data

Often face the “curse of dimensionality”

Dimension reduction: Transform the vectors into lower dimension **while retaining useful properties**

Today we will study two techniques: (1) Johnson-Lindenstrauss Transform, (2) Principal Component Analysis

Johnson-Lindenstrauss Transform

- Linear transformation
  - Specifically, multiple vectors with a specially chosen matrix
- Preserves pairwise distances (L2) between the data points

JL Lemma:

Let $\varepsilon \in (0, 1/2)$. Given any set of points $X = \{x1, x2, \ldots, xn\}$ in RD, there exists a map $S:RD \rightarrow Rk$ with $k = O(\varepsilon^{-2}\log n)$ s.t

$1-\varepsilon \leq \|Sxi - Sxj\|2 \leq 1+\varepsilon. \|xi - xj\|2$

Observations:

- The final dimension after reduction (i.e. k is independent of the original dimension D)
- It is dependent only on the number of points n and the accuracy parameter ε

Johnson-Lindenstrauss Transform

Construction:

Let M be a $k \times D$ matrix, such that every entry of M is filled with an i.i.d. draw from a standard Normal N(0,1) distribution (a.k.a. the Gaussian distribution)

Define the transformation matrix $S := \frac{1}{\sqrt{k}}M$.

Transformation: The point $x \in R^D$ is mapped to Sx

- I.e.: Just multiply with a Gaussian matrix and scale with $\frac{1}{\sqrt{k}}$
- The construction does not even look at the set of points X

Johnson-Lindenstrauss Transform

Proof for JL Lemma:

We will assume the following Lemma (without proof).

Lemma 2:

Let $\varepsilon \in (0, 1/2)$. If S is constructed as above with $k = O(\varepsilon^{-2} \log \delta^{-1})$,

and $x \in RD$ is a unit vector (i.e., $\|x\|_2 = 1$), then $\Pr[\|Sx\|_2 \in (1 \pm \varepsilon)] \geq 1 - \delta$.

Q: Why are we done if this Lemma holds true?

Johnson-Lindenstrauss Transform

Q: Why are we done if this Lemma holds true?

Set $\delta = 1/n2$, and hence $k = O(\varepsilon^{-2} \log n)$.

Now for each $x_i, x_j \in X$ we get that the squared length of the unit vector $x_i - x_j$ is maintained to within $1 \pm \varepsilon$ with probability at least $1 - 1/n2$.

Since the map is linear, we know that $S(\alpha x) = \alpha S x$, and hence the squared length of the non-unit vector $x_i - x_j$ is in $(1 \pm \varepsilon)\|x_i - x_j\|2$ with probability $1/n2$

Next by a union bound, all nChoose2 pairs of squared lengths in XChoose2 are maintained with probability at least 1 − nChoose2 *1/n^2 ≥ ½

Shows that a randomized construction works with constant prob!

Johnson-Lindenstrauss Extensions

Lot of research on this topic.

- Instead of the entries of the $k \times D$ matrix M being Gaussians, we could have chosen them to be unbiased $\{-1, +1\}$ r.v.s. The claim in Lemma 2 goes through almost unchanged!

- Sparse variations for reducing computation time

Principal Component Analysis

In JL Transform, we did not assume any structure in the data points. Oblivious to the dataset. Cannot exploit any structure.

What is the dataset is well-approximated by a low-dimensional affine subspace?

That is for some small k, there are vectors u1, u2, . . . , uk ∈ RD such that every xi is close to the span of u1, u2, . . . , uk.

Applications

- Analysis of genome data and gene expression levels in the field of bioinformatics
  - Gene microarray data: Microarrays measure activity levels of a large number of genes, say D = 10, 000 genes. After testing m individuals, one obtains m vectors in RD.
  - In practice it is found that this gene expression data is low-dimensional (some biological phenomenon that activates multiple genes at a time).

- Denoising of stock market signals

Principal Component Analysis

The goal of PCA is to find k (orthonormal) vectors such that the points in the datasets have a good approximation in the subspace generated these vectors

Good approximation: in the L2 sense, that is the L2 distance (a.k.a. mean squared error) between the given points and their closest approximation in the low-dimensional subspace obtained is minimized

We look for orthonormal vectors: since we want the basis vectors for the low-dimensional space

Principal Component Analysis: Preprocessing

PCA is very sensitive to scaling

Data needs to be preprocessed before performing PCA

- Data needs to be mean zero
  - Achieved by subtracting by sample mean
- Each coordinate needs to be scaled appropriately so that they are comparable
  - Empirically, dividing each coordinate (column) by sample standard deviation has been found to perform well

Principal Component Analysis

Minimizing l2 error of approximation = maximizing the projected distances

(draw picture for 1 dimensional case)

(can easily see why scaling of dimensions matter)

That is, PCA **maximizes the variance of the projected points**

Let us first go through the 1 dimensional case for intuition

PCA: 1-dimensional case

Given a unit vector u and a point x, the length of the projection of x onto u is given by $x^T u$

To maximize the projected distances:

$$\sum_{i=1}^{n} (x_i^T u)^2 = \sum_{i=1}^{n} u^T x_i x_i^T u$$

$$= u^T \left(\sum_{i=1}^{n} x_i x_i^T \right) u = u^T M u$$

(M is symmetric)

$$\underset{u \in \mathbb{R}^D, \|u\|_2 = 1}{\operatorname{argmax}} \; u^T M u = \text{principal eigen vector of } M$$

(Linear algebra)

PCA: k-dimensional case

Let $M = \sum_{i=1}^{n} x_i x_i^T$ (DxD)

EVD of $M = V \Lambda V^T$ where $\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \end{bmatrix}$

Let $\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots$

Let $v_1, v_2, v_3 \cdots v_k$ be the first k columns of V

Then $v_1, v_2, v_3 \cdots v_k$ are the k orthonormal vectors that maximize the projected distances (i.e variance)

PCA Algorithm

Preprocess the data

Compute the "covariance matrix" M = $\sum_{i=1}^{n} x_i x_i^T$

Find Eigen value decomposition of M = $V \Lambda V^T$

where $\Lambda = \text{diag}(\lambda_1, \lambda_2 \dots)$ and $\lambda_1 \geq \lambda_2 \geq \lambda_3 \dots$

$V = [v_1 \; v_2 \; \dots \; v_D]$

Set the linear transformation matrix as $S = \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix}$

When PCA does not work?

- PCA finds a linear approximation. If the low dimensionality of the data is due to non-linear relationships then PCA cannot find it. E.g. (x, y) with y = x^2

- If normalization is not done correctly