Similarity Search 2

Presentation partially based on slides from:

http://infolab.stanford.edu/~ullman/mining/2009/index.html

Anand Rajaraman & Jeff Ullman

Agenda

- Cosine Similarity and Random Projections (sketches)
- Euclidean distance and Random Projections
- An application of LSH to searching similar fingerprints
- *(Optional) General theory of LSH*
  - *LSH families*
  - *AND & OR constructs*

Cosine Similarity, Cosine Distance

- In many applications, objects are represented not by sets but as vectors in R^d *(or in $\{0,1\}^d$ or in N^d , …)*

Study
Chapter 1!

- **Examples:**
  - a document can be represented by a vector of frequencies (or ***TF.IDF*** *values*) of words that occur in the document
  - an image can be represented by a vector of pixel intensities (e.g., 256 counts, if 1 byte per pixel)
  - a sound (segment) can be represented by its frequency spectrum (e.g., FFT -> 128 floats)
  - … a vector of ratings given by a user to movies he/she watched (Netflix Challenge: ~17.000 numbers)

- **Cosine Distance between two vectors, v_1 and v_2, is defined as *the angle* between v_1 and v_2**

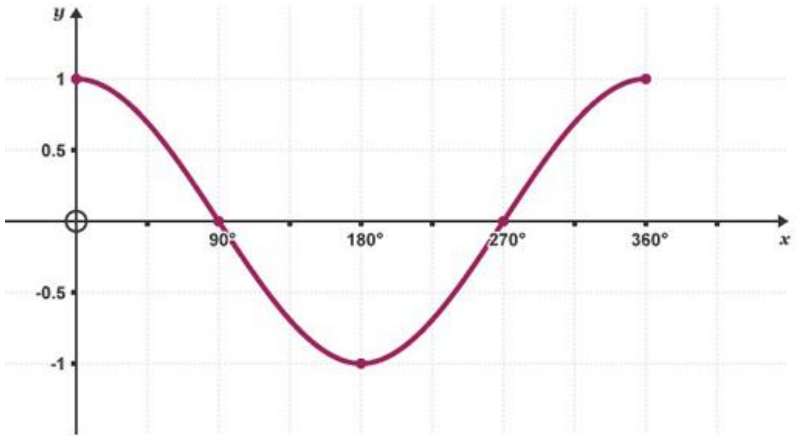
  Thus, we don't care about the lengths of vectors, only about the angle between them.
  The smaller the angle the more similar the vectors are.

Cosine Distance: the θ

- Think of a point as a vector from the origin (0,0,...,0) to its location.
- Two points' vectors make an angle, whose cosine is the normalized dot-product of the vectors: $\mathbf{\cos(\theta) = p_1 \cdot p_2 / |p_1||p_2|}$.
  - Example: $p_1 = 00111$; $p_2 = 10011$.
  - $p_1 \cdot p_2 = 2$; $|p_1| = |p_2| = \sqrt{3}$.
  - $\cos(\theta) = 2/3$; θ **is about 48 degrees**.

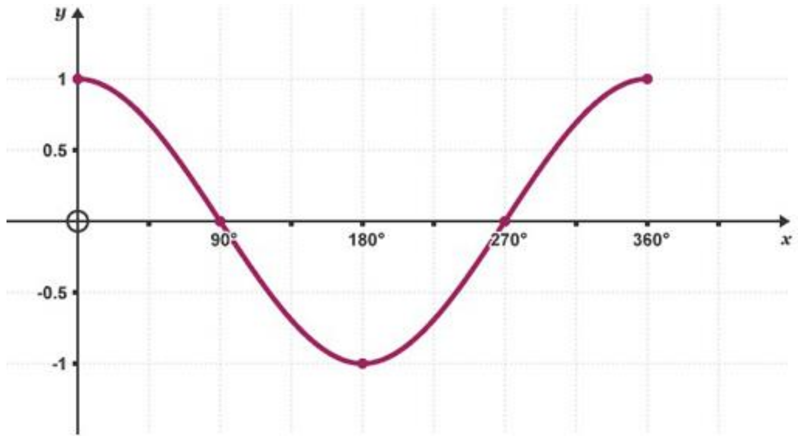
The $\cos(\theta) = p_1 \cdot p_2 / |p_2||p_1|$ formula

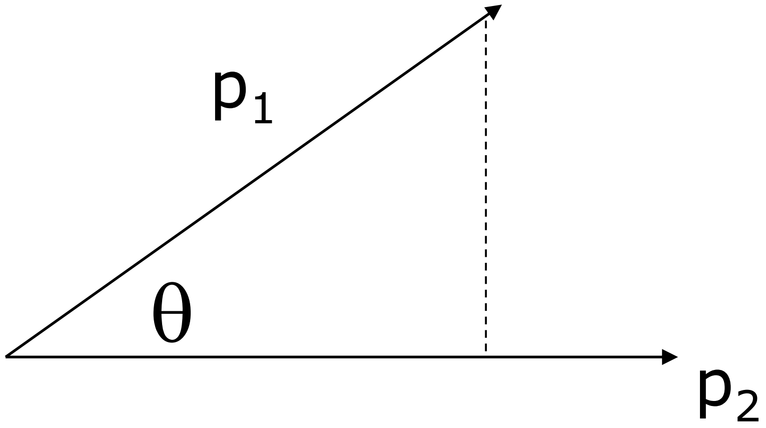
- $p_1 \cdot p_2$ is a vector dot product: $p_1 \cdot p_2 = \sum_{n=1}^{dim}(p_{1,n} \cdot p_{2,n})$
- $|p_2| \cdot |p_1|$ is the product of lengths of p_1 and p_2
- There are many proofs of this equation, e.g.,:
  - http://www.mit.edu/~hlb/StantonGrant/18.02/details/tex/lec1snip2-dotprod.pdf
  - https://www.youtube.com/watch?v=PnJoKGynu_U
- Note that $p_1 \cdot p_2$ might be negative!
- $\cos(\theta)$ takes values between -1 and 1.
- If components of p_1 and p_2 are non-negative (as in A2) then $\cos(\theta)$ is also non-negative.



Cosine Distance and Cosine Similarity

cosine distance: $d(p_1, p_2) = \theta = \mathit{arccos}(p_1.p_2/|p_2||p_1|)$





Wikipedia: cosine similarity(p_1, p_2) = ***$\cos(\theta)=p_1.p_2/|p_2||p_1|$***; *values in [-1,1]*

BUT: *<u>in the textbook*</u>*: ***cosine similarity(p_1, p_2) = $1-\theta/180$***; *values in [0,1]*

**Actually, the term "cosine similarity" is not used in the textbook at all!*

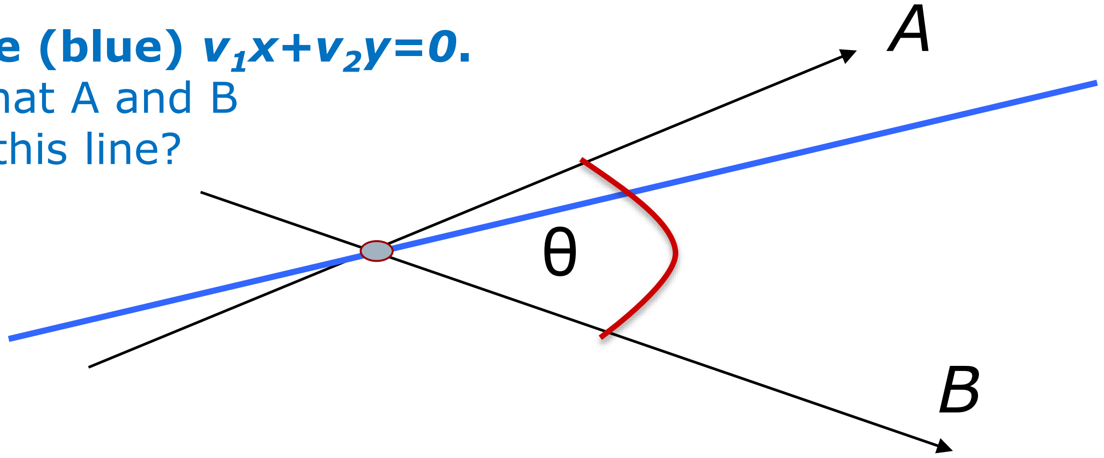
Why C.D. Is a Distance Measure?

- d(x,x) = 0 because arccos(1) = 0.
- d(x,y) = d(y,x) by symmetry.
- d(x,y) $\geq$ 0 because angles are chosen to be in the range 0 to 180 degrees.
- Triangle inequality: physical reasoning. If I rotate an angle from *x* to *z* and then from *z* to *y*, I can't rotate less than from *x* to *y*.
- *Cosine similarity ranges between 0 and 1 (Textbook!)*

Estimating θ in 2 dimensions

Consider two vectors *A* and *B with angle $\theta < 180^o$*.

Select at random a line (blue) $v_1x+v_2y=0$.
What is the probability that A and B are on the same side of this line?



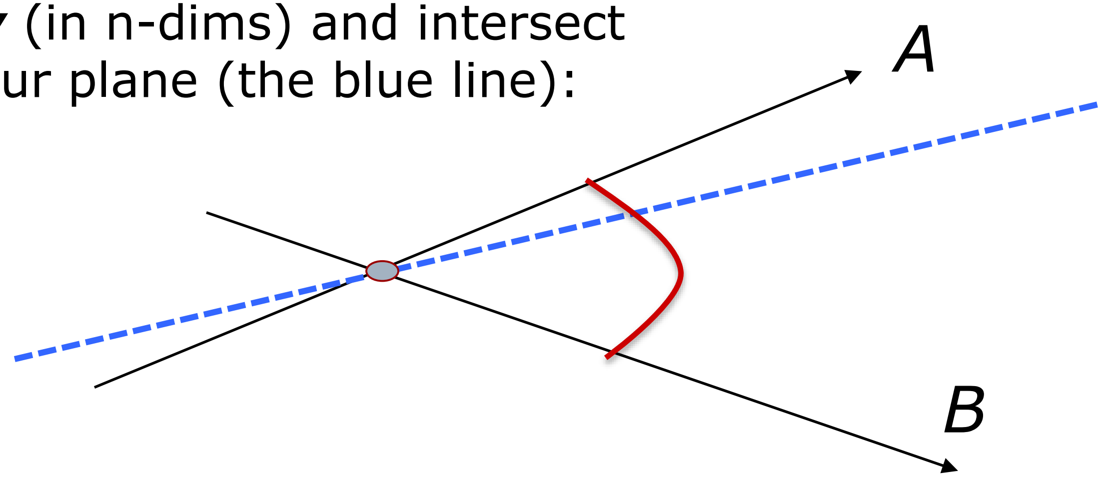
Prob[A and B on different sides of the line] = θ/180
Prob[A and B on the same side of the line] = 1-θ/180

Let v a normal vector that defines the line. Then v.B>0 ("x is above") and v.A<0 ("y is below")

Textbook Cosine Similarity

Estimating θ in n-dimensions …

Consider two vectors *A* and *B in n-dimensions with angle* $\theta < 180^o$.
These 2 vectors define (or span) a 2-dimensional plane!
Let's plot this plane with both vectors (see below) …
Now take a random vector $\boldsymbol{v}$ (in n-dims) and intersect the hyperplane $\boldsymbol{vx}=\boldsymbol{0}$ with our plane (the blue line):



Repeat the reasoning from the previous slide (+ some imagination: textbook page 118)

LSH for Cosine Similarity

- LSH is based on the idea of hash functions: “the chance that two objects will hash to the same value is the same as similarity of these objects”

- Instead of “minhashing” we now use “random projections”: hash functions which return 1 or -1 depending on whether x is below or above some randomly chosen hyperplanes.

Random Projections

- Pick a random vector *v*, which determines a hash function h_v with two buckets.
- $h_v(x) = +1$ if $v.x > 0$; -1 if $v.x < 0$.
- Claim: Prob[h(x)=h(y)] = cosine_sim(x,y)
- Proof: look at slide 7!

Signatures for Cosine Distance

- Pick some number of random vectors, and hash your data for each vector.

- The result is a signature (*sketch*) of +1's and –1's that can be used for LSH like the minhash signatures (with the banding trick) for the Jaccard similarity!

- *If dimensionality is high, it suffices to consider only vectors v consisting of +1 and –1 components (eliminates multiplications - we sum up "positives" and subtract "negatives"!)*

LSH for Euclidean Distance

- Key Idea: if two points are close to each other then no matter from which direction we look at them, they always seem to be close; if they are far apart, then it's difficult to find such a direction

- **Hash functions are defined by random directions (lines) that are partitioned into buckets (intervals) of size *a*.**

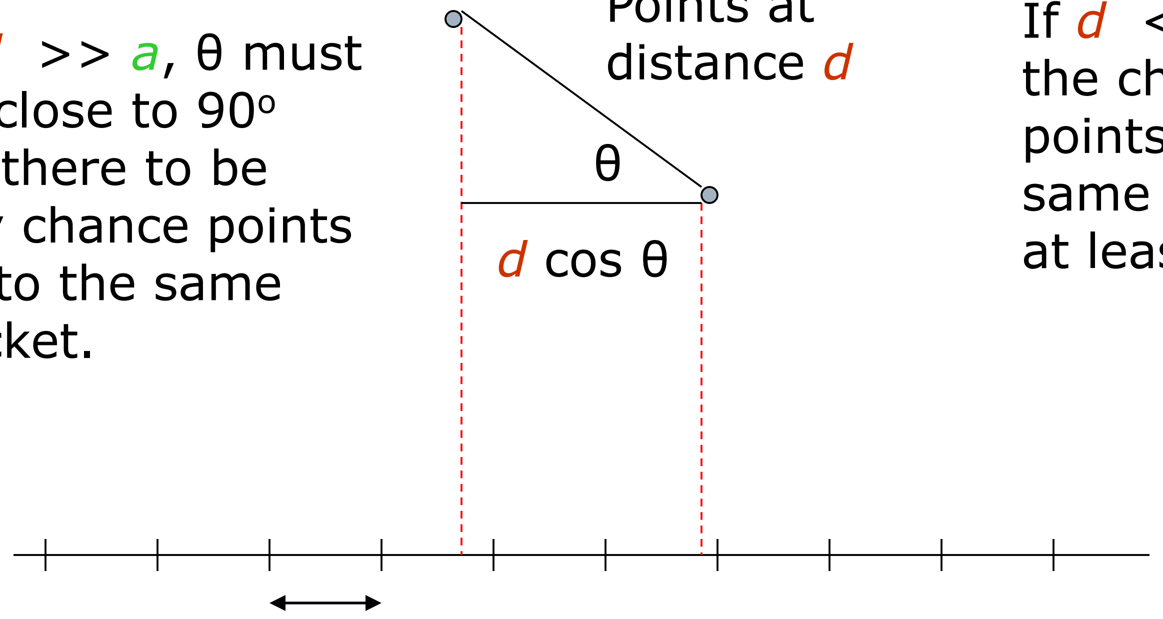
- Hash each point to the bucket containing its projection onto the line:
  $h_{w,a}(x)$ = the id of the bucket that contains w*x

- Nearby points are often close (in the same bucket); distant points are rarely in the same bucket => **probability of hashing x and y to the same bucket is proportional to the Euclidean similarity of x and y.**

- ***More details in the textbook (3.7.4)***

Projection of Points

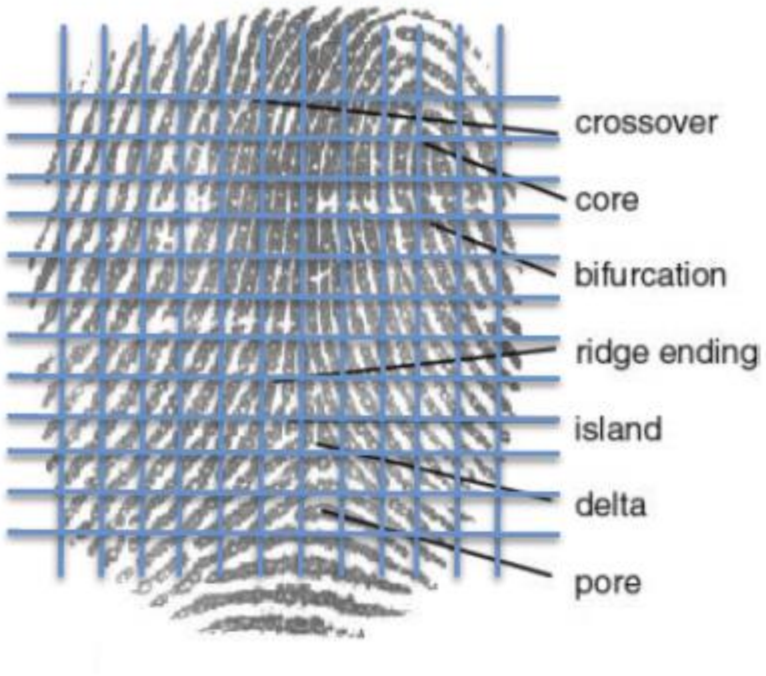
If d >> a, θ must be close to 90º for there to be any chance points go to the same bucket.



If d << a, then the chance the points are in the same bucket is at least $1 - d/a$.

Application of LSH: Fingerprint Matching

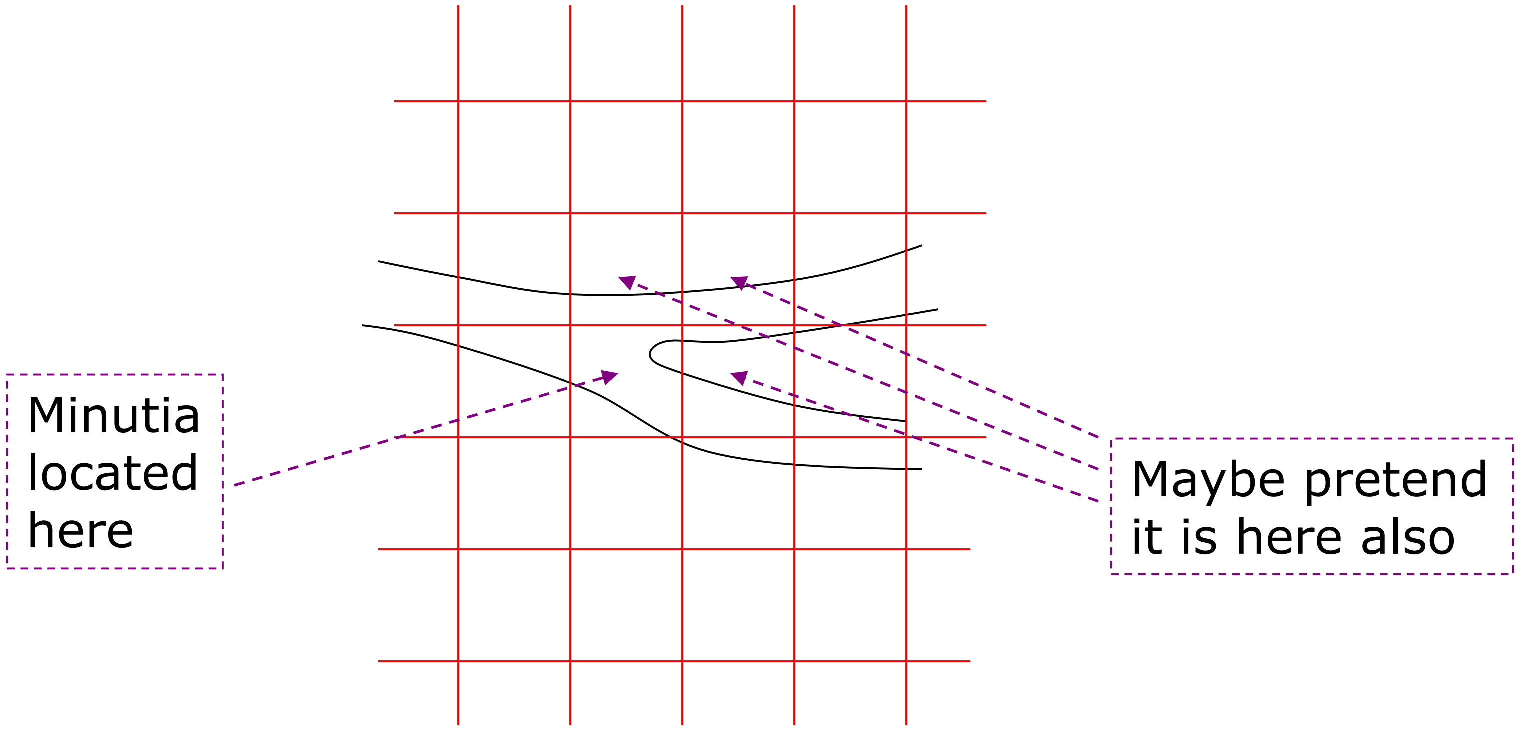
- Fingerprints are represented by the set of *minutiae* features.



LSH for Fingerprints

- Place a grid on a fingerprint.
  - Normalize so identical prints will overlap.
- Set of grid points where minutiae are located represents the fingerprint.
  - Possibly, treat minutiae near a grid boundary as if also present in adjacent grid points.

Discretizing Minutiae



Assumptions:

- Suppose typical fingerprints have minutiae in 20% of the grid points.
- Suppose fingerprints from the same finger agree in at least 80% of grid points.
- Suppose we have 1.000.000.000 fingerprints
- **Challenge:**
  speed up the process of searching this data base with help of LSH!

Applying LSH to Fingerprints

- Make a bit vector for each fingerprint's set of grid points with minutiae.
- We could minhash the bit vectors to obtain signatures.
- But since there probably aren't too many grid points, we can work from the bit-vectors directly.

LSH/Fingerprints (2)

- Pick 1024 (?) sets of 3 (?) grid points, randomly.
- For each set of points, prints with 1 for all three points are candidate pairs.
- Thus we have 1024 “hash functions” and every fingerprint is mapped into a string of 1024 bits - “signatures” of fingerprints (128 bytes long)
- Apply the “banding technique” with b=1024 (one row per band)…

LSH/Fingerprints (3)

- Typical fingerprints have minutiae in 20% of the grid points.
- Fingerprints from the same finger agree in at least 80% of grid points.
- Probability two random fingerprints each have 1 in all three points = $(0.2)^6 = .000064$.
- Why ^6? Both fingers should return 3x1 and they are coming from different persons

LSH/Fingerprints (4)

First image has 1 in a point

Second image of same finger also has 1.

- Probability two fingerprints from the same finger each have 1's in three given points = $(\boxed{(0.2)}(\boxed{0.8}))^3$ = .004096.

- Prob. for at least one of 1024 sets of three points = $1-(1-.004096)^{1024}$ = .985.

1.5% false negatives

- But for random fingerprints: $1-(1-.000064)^{1024}$ = .063.

6.3% false positives

Example: Reducing the false positives

- Build two systems with 1024 buckets each.
- For an input fingerprint return the intersections of sets returned by both systems

- Prob. of false negatives = $(1- 0.985^2)= 3\%$

- Prob. of false positives:
  $0.063^2= 0.4\%$ (16 times better!)