Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Data Mining
Learning from Large Data Sets

Lecture 3 – Locality Sensitive Hashing

263-5200-00L

Andreas Krause

Announcement

- No class next week

Review:

Fast near neighbor search in high dimensions

Locality sensitive hashing

- **Idea**: Create hash function that maps "similar" items to same bucket

Hashtable

0	**1**	**2**	**3**

- **Key problem**: Is it possible to construct such hash functions??
  - Depends on the distance function
  - Possible for Jaccard distance!! ☺
  - Some other distance functions work as well

Recall: Shingle Matrix

documents

shingles

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

$$\mathrm{Sim}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Min-hashing

- Simple hash function, constructed in the following way:

- Use random permutation π to reorder the rows of the matrix
  - Must use same permutation for all columns C!!
- *h*(*C*) = minimum row number in which permuted column contains a 1

$$h(C) = h_\pi(C) = \min_{i:C(i)=1} \pi(i)$$

Min-hashing property

- Want that similar documents (columns) have same value of hash function (with high probability)

- Turns out it holds that

$$\underset{h}{\Pr}[h(C_1) = h(C_2)] = \mathrm{Sim}(C_1, C_2)$$

- Need to control false positives and misses.

Min-hash signatures

1	4	3
3	2	4
7	1	7
6	3	6
2	6	1
5	7	2
4	5	5

Input matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

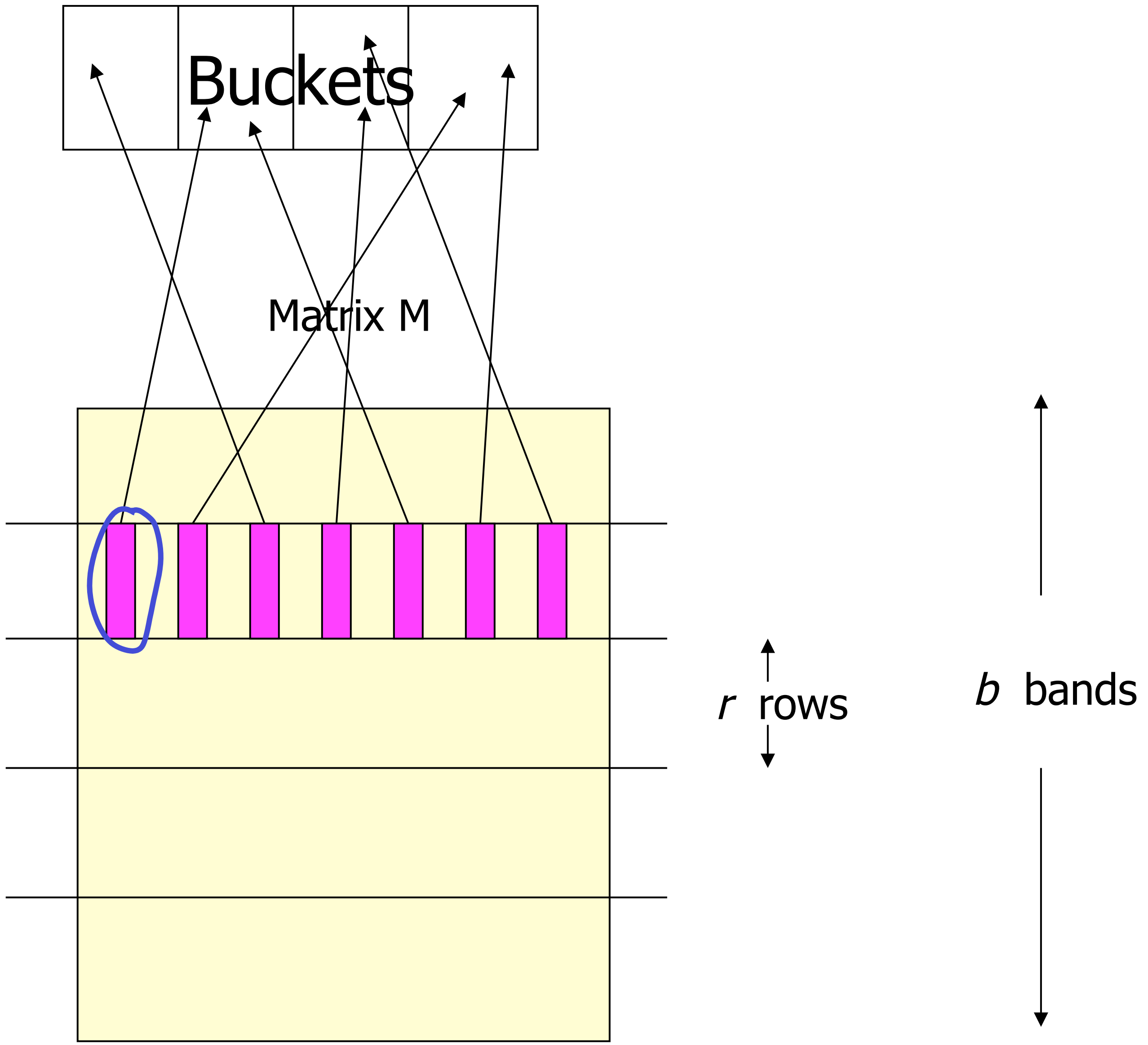
Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

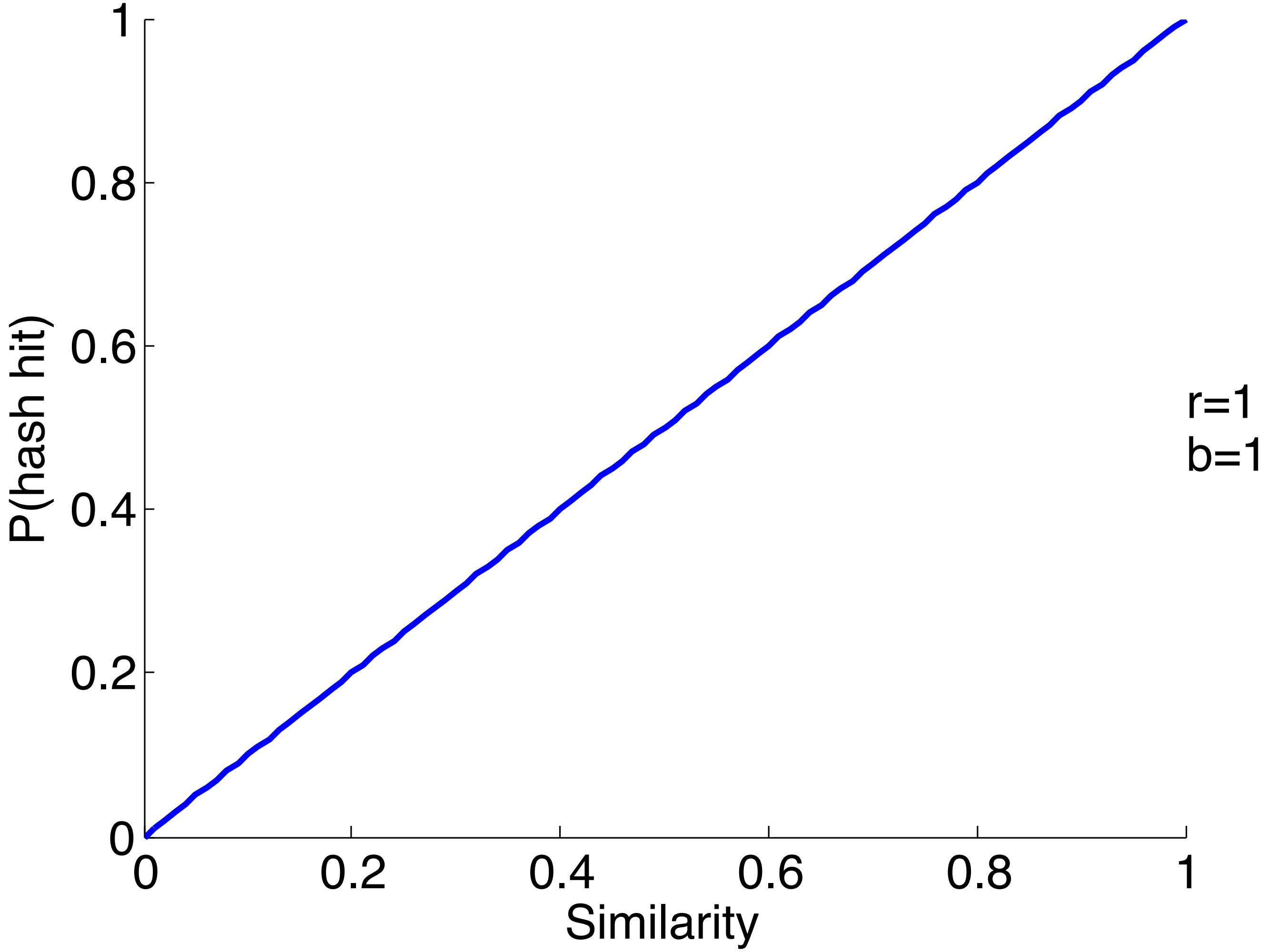
Similarities:

	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

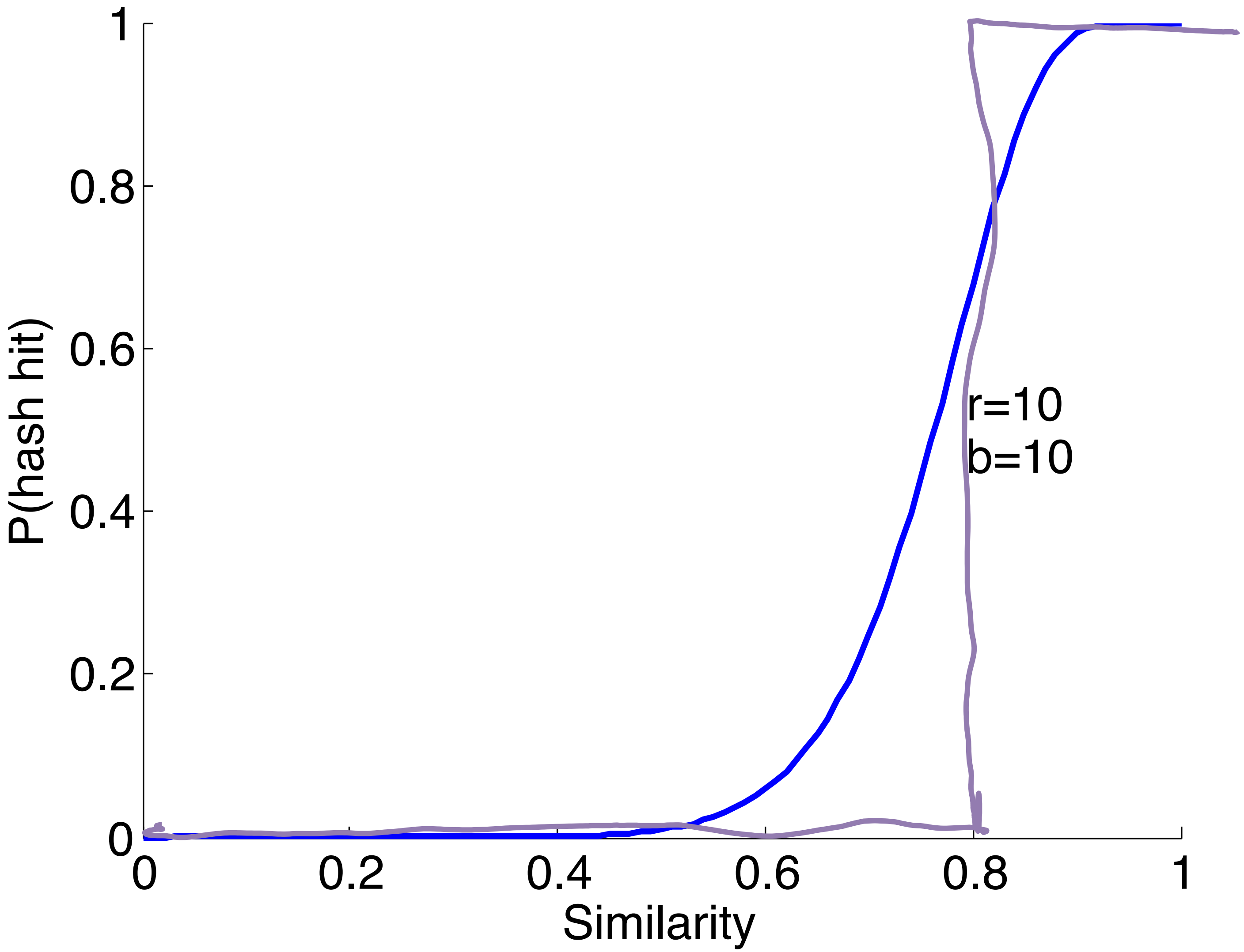
Hashing bands of M



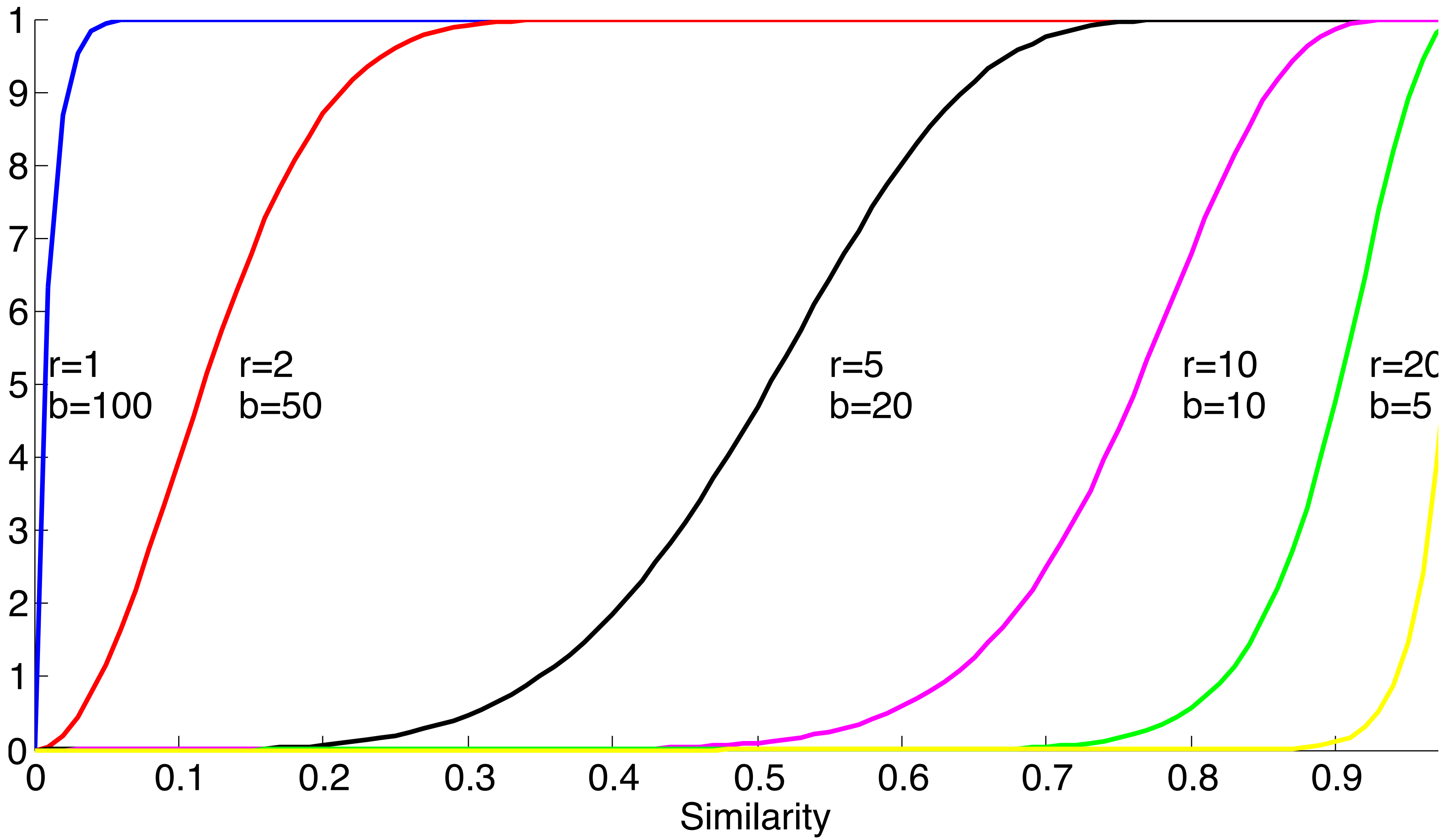
One hash function



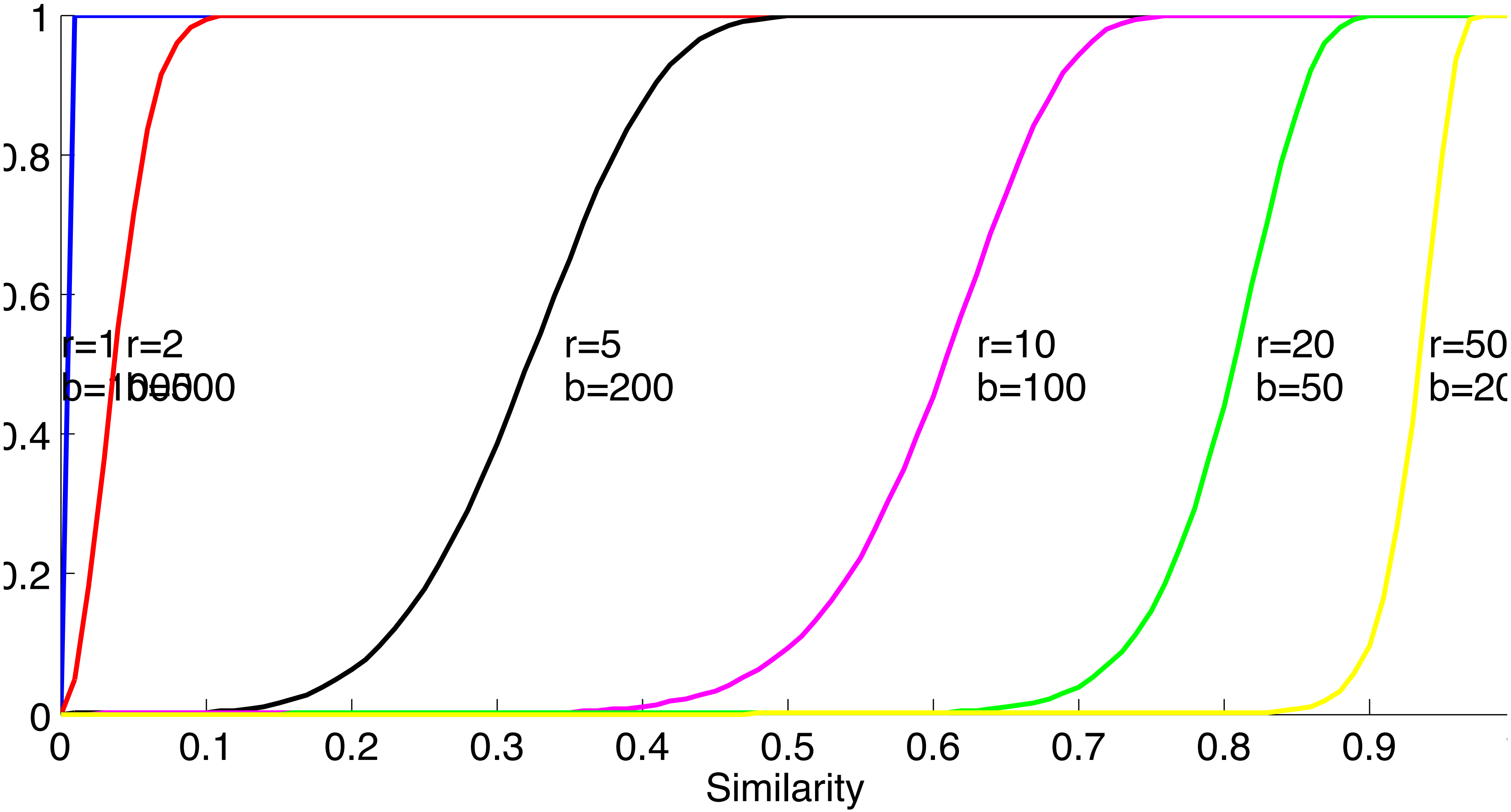
100 hash functions



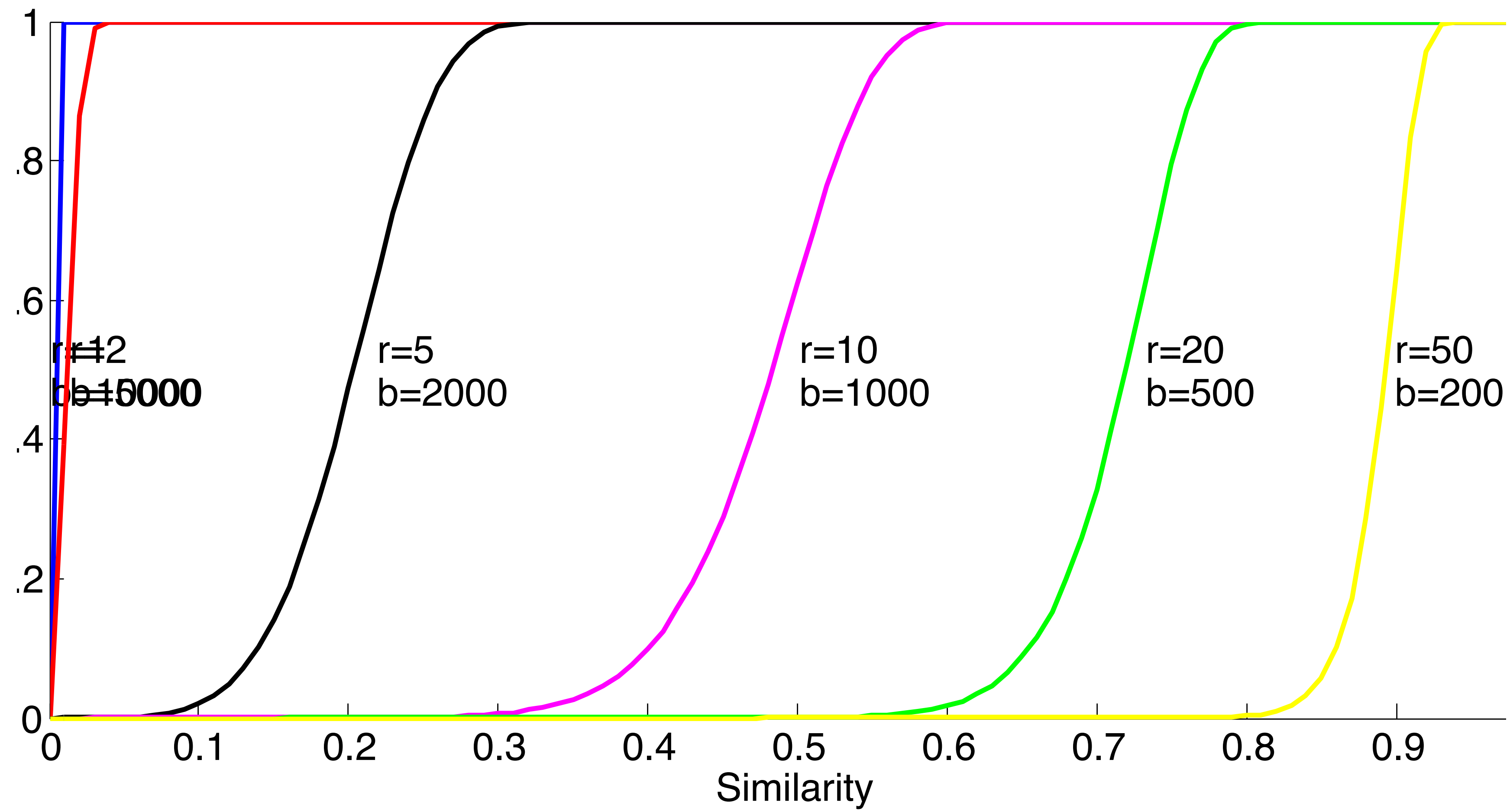
100 hash functions



1000 hash functions



10000 hash functions

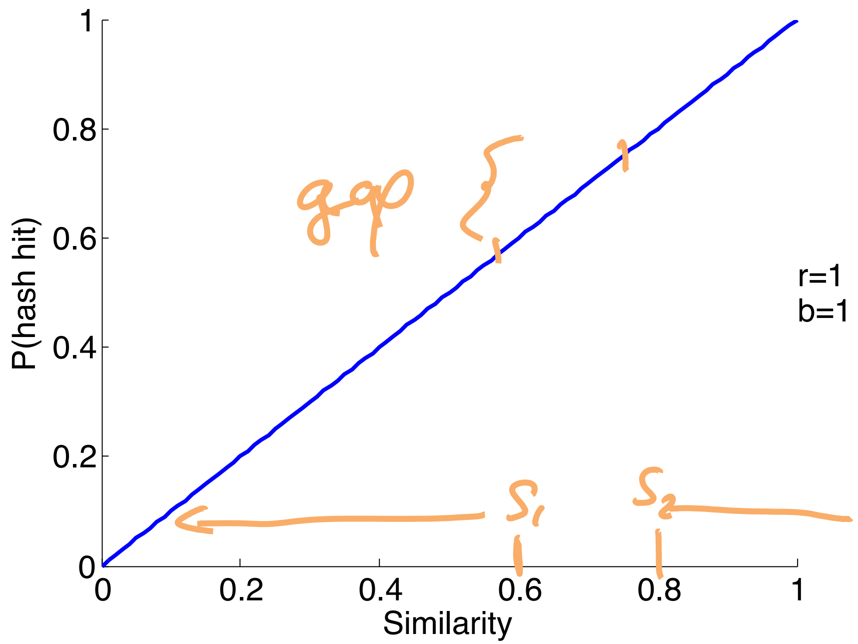


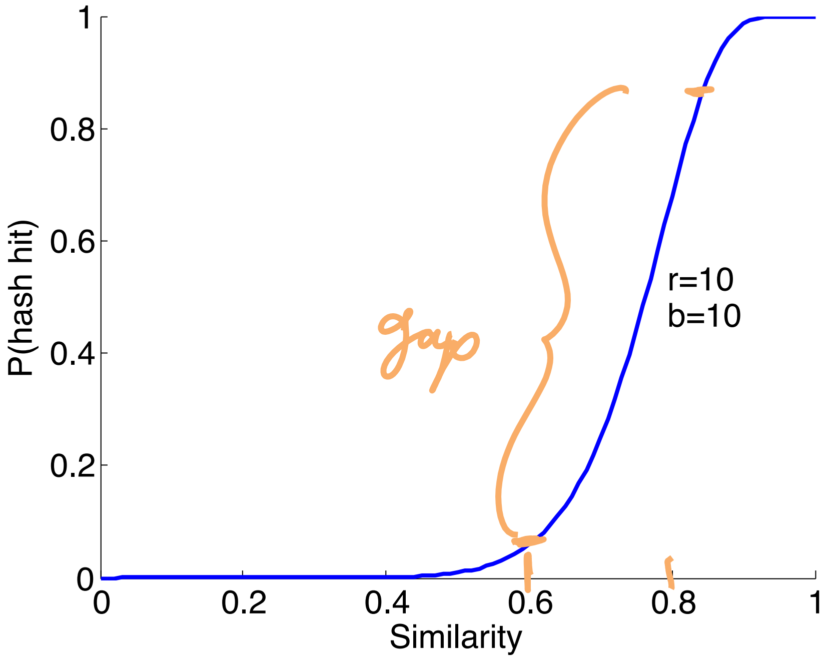
LSH more generally

- So far we have considered
  - Min-hashing for computing compact document signatures representing Jaccard similarity
  - Locality Sensitive Hashing (LSH) for decreasing false negatives and false positives
  - Let's us do duplicate detection without requiring pairwise comparisons!
- Can we generalize what we learned?
  - Other data types (e.g., real vectors → images)
  - Other distance functions (Euclidean? Cosine?)

Key insight behind LSH

- LSH allows to boost the gap between similar (Sim(C1,C2)>s) non-similar (Sim(C1,C2)<s' for s' < s) pairs





LSH more generally

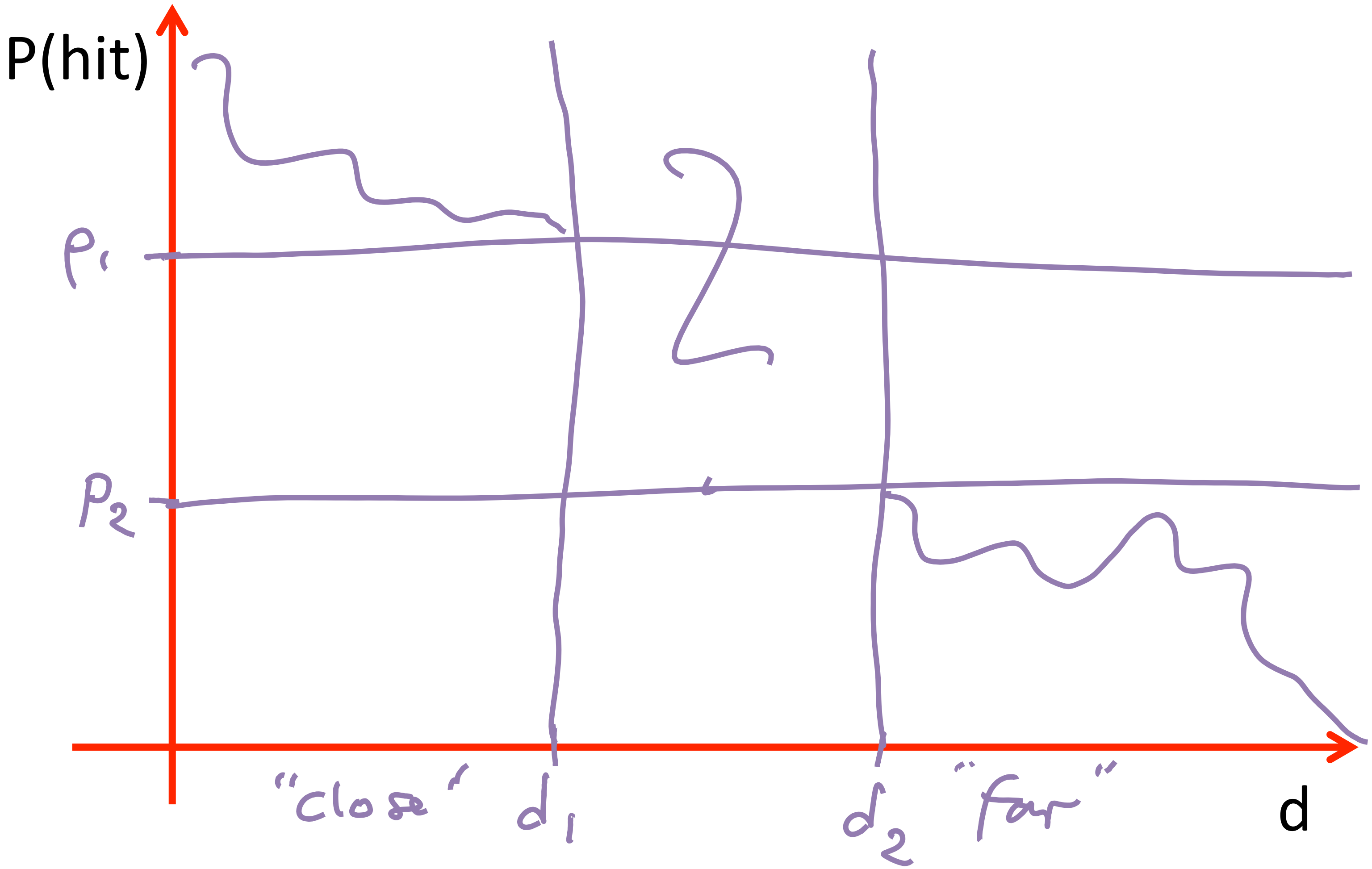
- Consider a metric space *(S,d)*, and a family *F* of hash functions *h: S→B* $= \{1 \dots n\}$
- *F* is called *(d_1, d_2, p_1, p_2)-sensitive* if

$$\forall x, y \in S : d(x, y) \leq d_1 \Rightarrow \Pr_h[h(x) = h(y)] \geq p_1$$

$$\forall x, y \in S : d(x, y) \geq d_2 \Rightarrow \Pr[h(x) = h(y)] \leq p_2$$

h drawn uniformly at random from F

Example



Example: Jaccard-distance

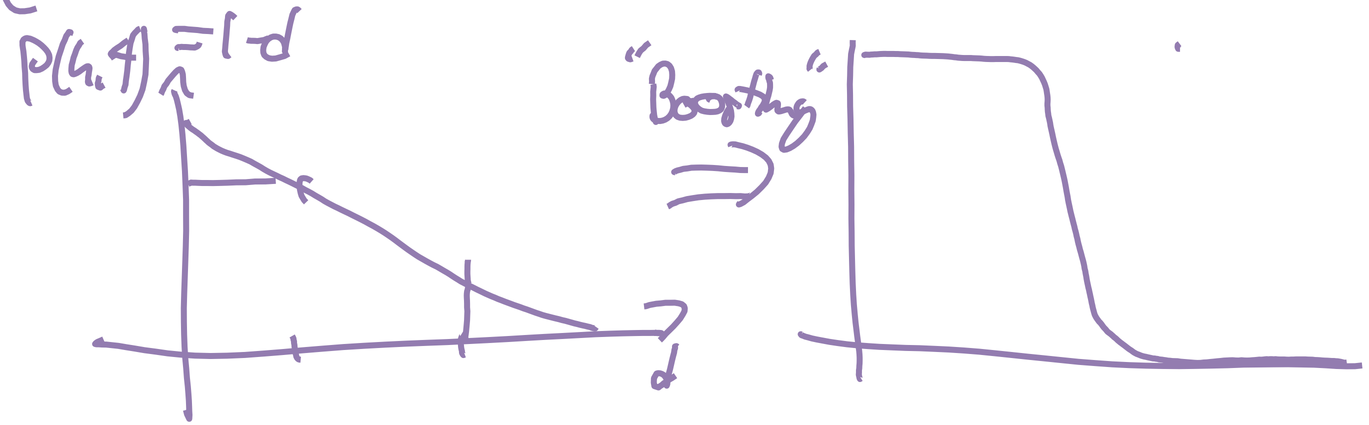
Recall, we want:

$$\forall x, y \in S : d(x,y) \le d_1 \Rightarrow \Pr[h(x) = h(y)] \ge p_1$$

$$\forall x, y \in S : d(x,y) \ge d_2 \Rightarrow \Pr[h(x) = h(y)] \le p_2$$

$F = \{ \pi$ permutations used in minhashing $\}$

is $(d_1, d_2, 1-d_1, 1-d_2)$-sensitive



Boosting a LS hash family

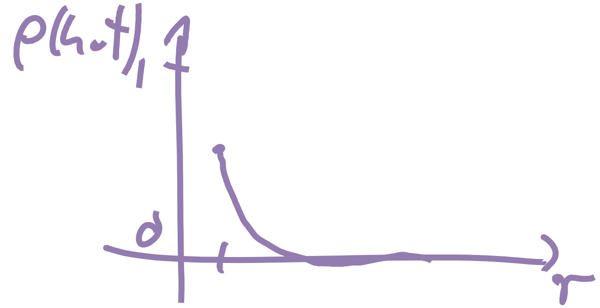
- Can we reduce false positives and false negatives (create "S-curve effect") for arbitrary LS hash functions??

- Can apply same partitioning technique!
- AND/OR construction

r-way AND of hash function

- **Goal**: Decrease false positives
- Convert hash family *F* to new family *F'*
- Each member of *F'* consists of a "vector" of r hash functions from *F*
- For $h = [h_1,\ldots,h_r]$ in ***F'***, $h(x)=h(y) \Leftrightarrow h_i(x)=h_i(y)$ for *all i*.

- Theorem: Suppose ***F*** is (d_1,d_2,p_1,p_2)-sensitive.
  Then ***F'*** is (d_1 , d_2 , p_1^r , p_2^r)-sensitive

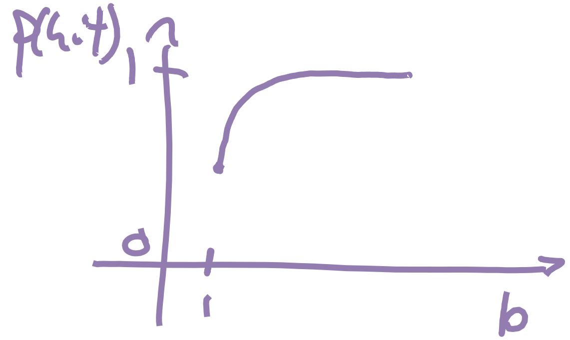


b-way OR of hash function

- **Goal**: Decrease false negatives
- Convert hash family *F* to new family *F'*
- Each member of *F'* consists of a "vector" of b hash functions from *F*
- For $h = [h_1,\ldots,h_r]$ in ***F'***, $h(x)=h(y) \Leftrightarrow h_i(x)=h_i(y)$ for *some i.*

collision

- Theorem: Suppose ***F*** is (d_1,d_2,p_1,p_2)-sensitive.
  Then ***F'*** is $(\ d_1\ ,\ d_2\ ,\ 1-(1-p_1)^b,\ 1-(1-p_2)^b\ )$-sensitive

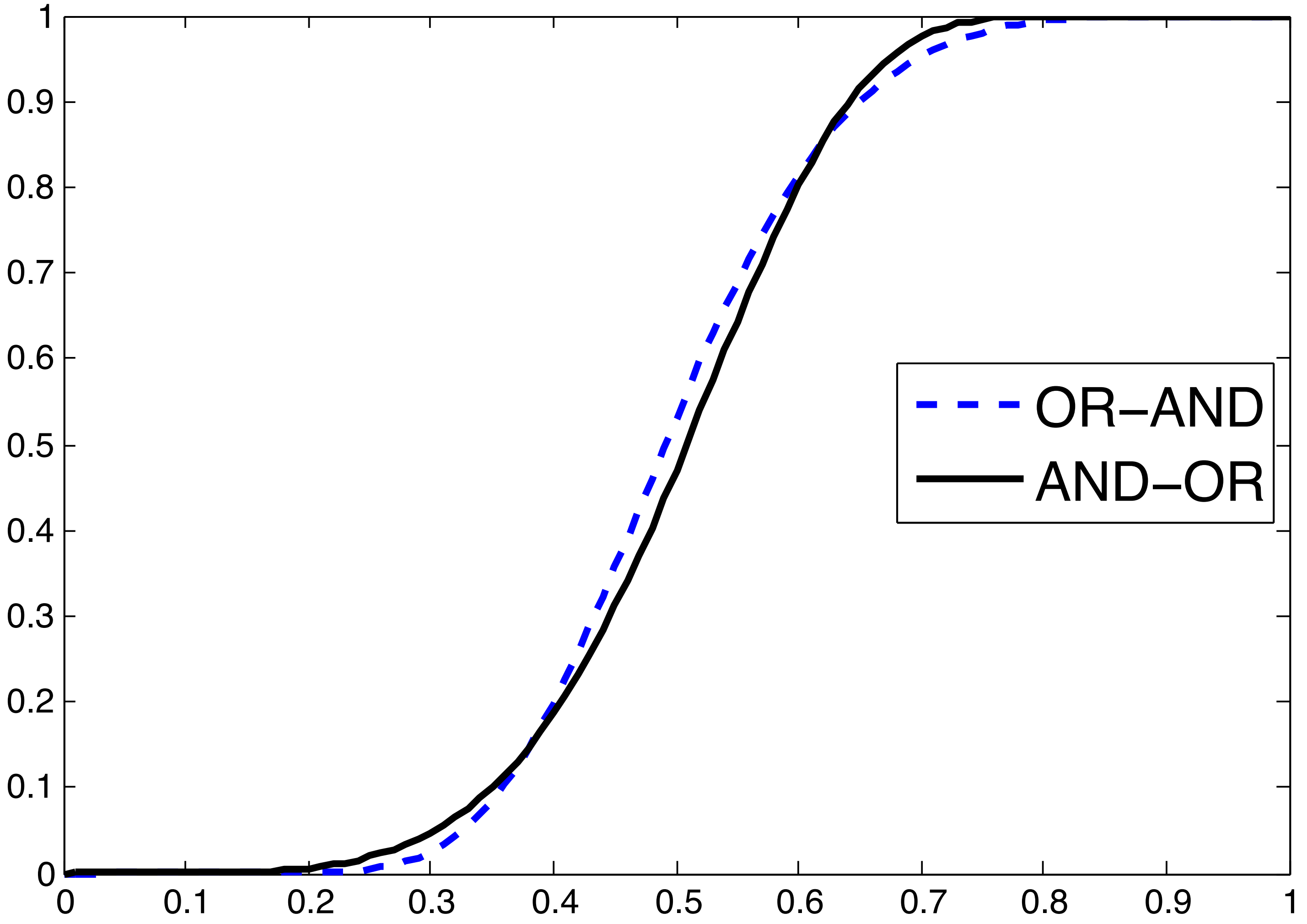


Composing AND and OR

- Suppose we start with a (d_1,d_2,p_1,p_2)-sensitive F
- First apply r-way AND, then b-way OR
- This results in $(d_1,d_2, 1-(1-p_1^r)^b, 1-(1-p_2^r)^b)$ sensitive F'

- Can also reverse order of AND and OR
- This results in $(d_1,d_2, (1-(1-p_1)^b)^r, (1-(1-p_2)^b)^r)$ sensitive F'

Example



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
OR–AND
AND–OR

Cascading constructions

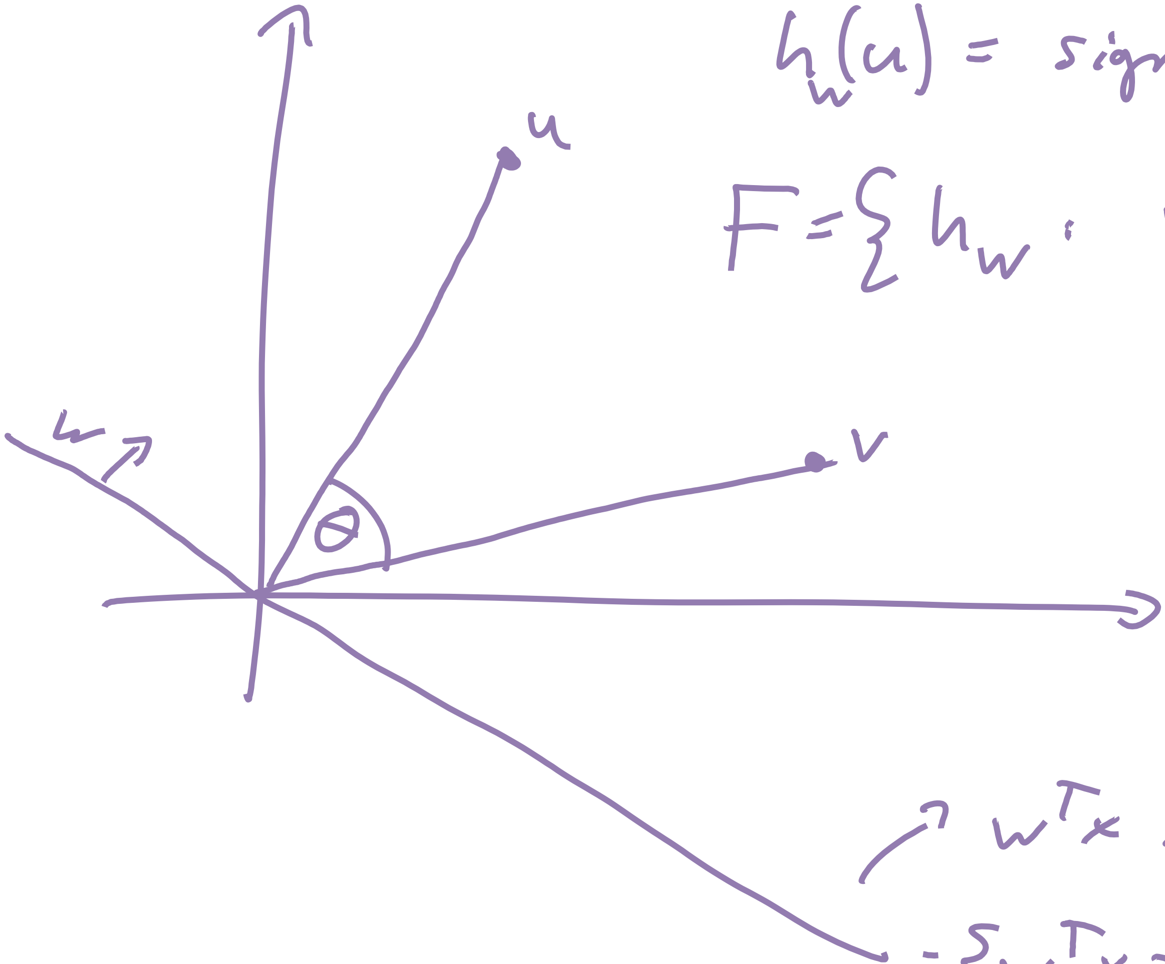
- Can also combine all previous constructions
- For example, first apply (4,4) OR-AND construction followed by a (4,4) AND-OR construction.
- Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9999996,.0008715)-sensitive family!
- How many hash functions are used?

$4^4 = 256$

Other examples of LS families

- *So far*: Jaccard distance has a LS hash family
- Several other distance functions do too
  - Cosine distance
  - Euclidean distance

LSH for Cosine Distance



$$h_w(u) = \text{sign}(w^T u)$$

$$F = \{ h_w : w \in \mathbb{R}^D, \|w\|_2 = 1 \}$$

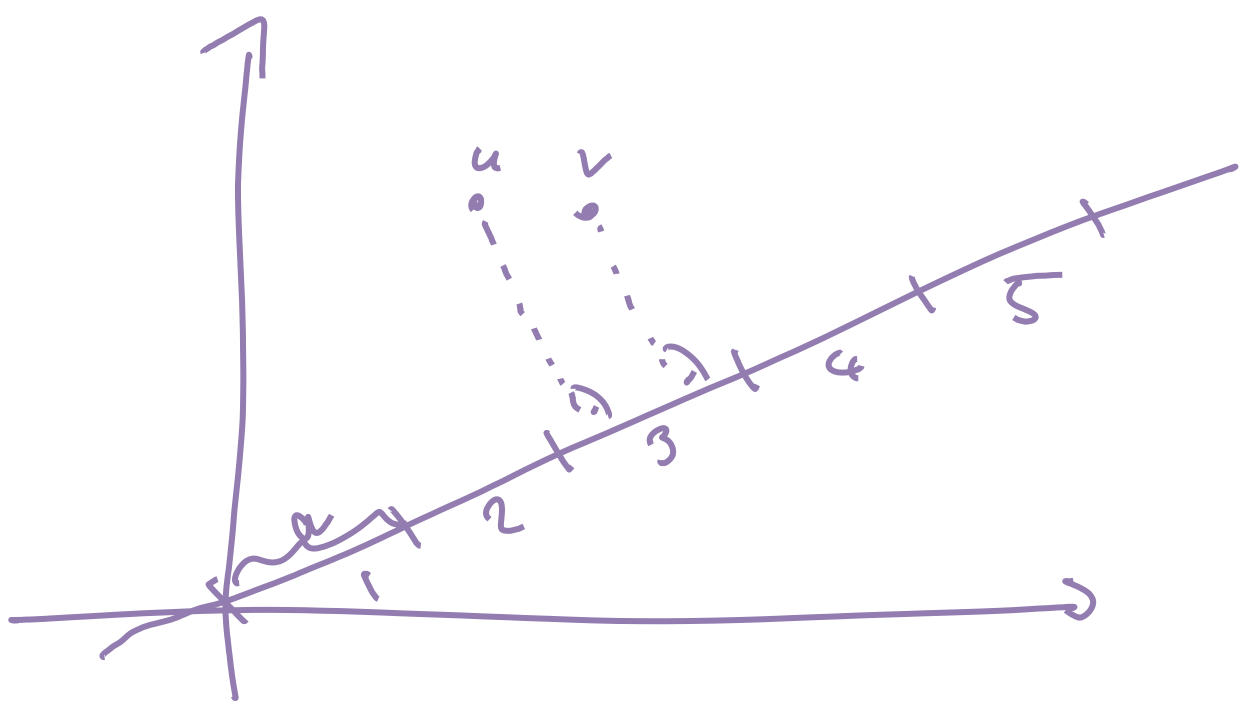
$w^T x > 0$

$= \{ w^T x = 0 \}$

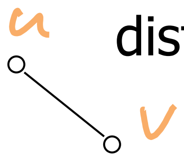
$w^T x < 0$

LSH for Euclidean distance

- **Key idea**: Map points to random line
- Here, let's consider 2 dimensions (but generalizes)

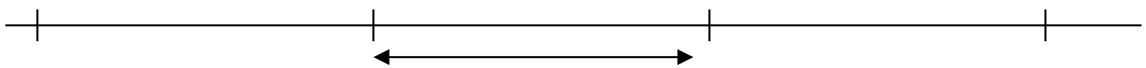


LSH for Euclidean distance



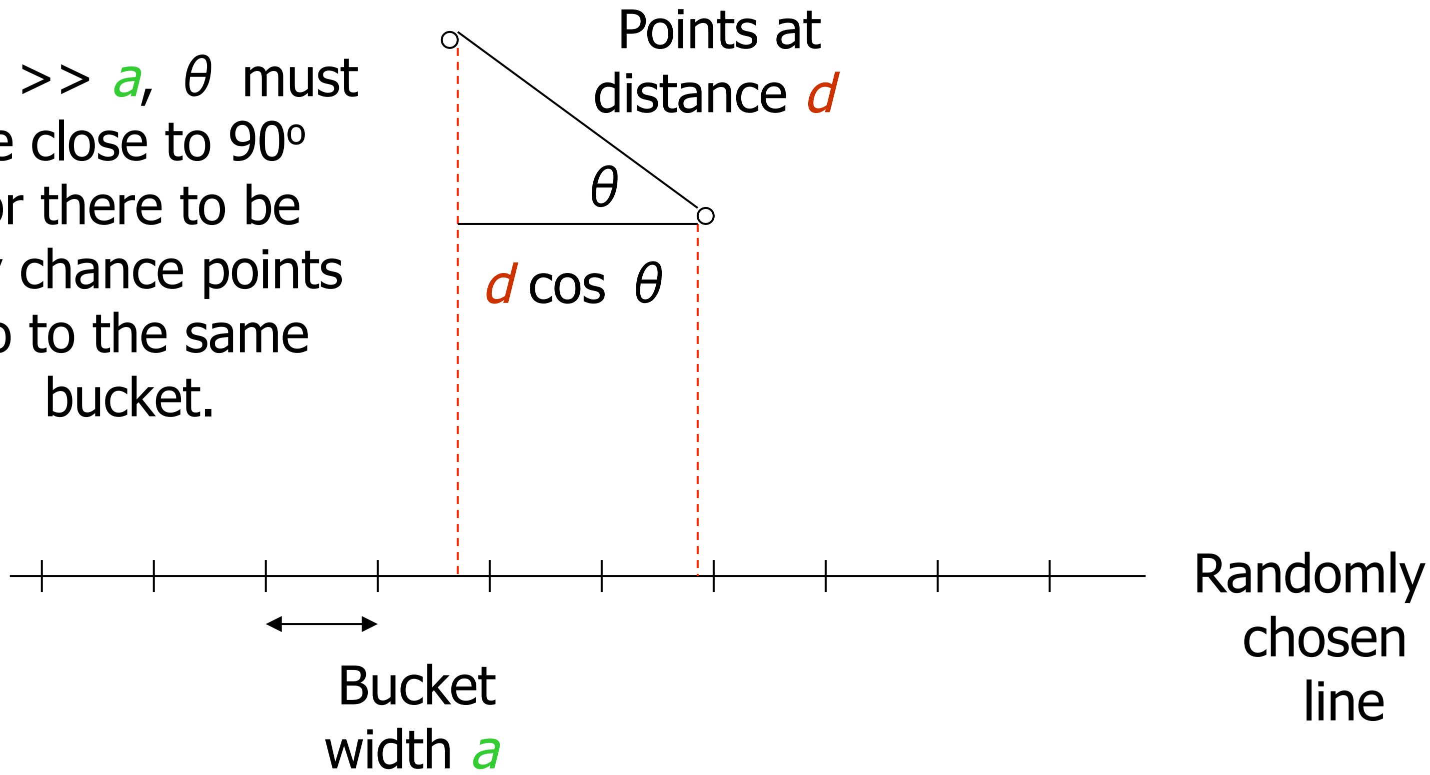
If $d < a$, then the chance the points are in the same bucket is

$1 - \frac{d}{a}$



LSH for Euclidean distance

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



LSH for Euclidean distance

- If distance $d \leq a/2$, P(same bucket) ≥ 1- d/a = 1/2
- If distance $d \geq 2a$, then they can end up in the same bucket only if d cos θ ≤ a
  - cos θ ≤ ½
  - 60 ≤ θ ≤ 90
  - This event has probability at most 1/3.
- Yields a (a/2, 2a, 1/2, 1/3)-sensitive family of hash functions for any a.
- Can boost using AND-OR constructions

LSH in MapReduce?

- LSH is well suited for MapReduce style computation!
- You'll find out how in the homework ☺

LSH for nearest neighbor search

- So far we discussed the problem of finding near duplicates.
- How do we implement nearest neighbor search?

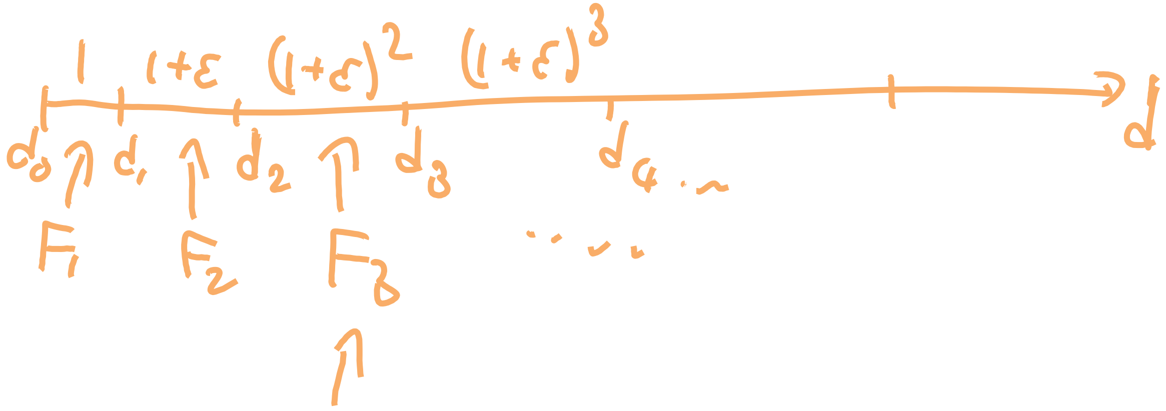
Approximate near-neighbor search

- Consider slightly different problem: *approximate near neighbor search*
  - Want to find any point in data set that has distance at most *r* from query
  - Don't want to return points of distance more than *(1+ε) r*

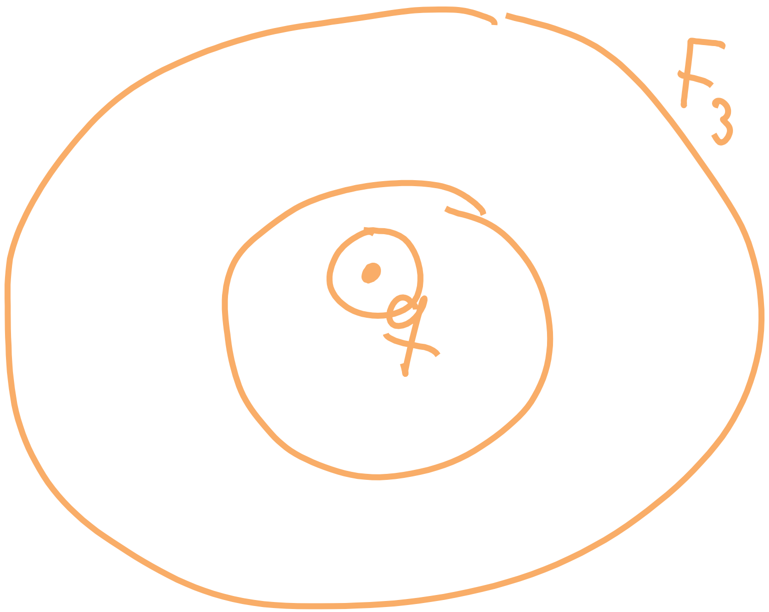
  - Pick *(r, (1+ε) r, p,q)-sensitive* hash family
- **Preprocessing**: Hash data set as in duplicate detection
- **Query**: Hash query in the same way
- Retrieve all candidate pairs (perhaps pick closest)

Approximate nearest neighbor search

- Can we use approximate near-neighbor search for (approximate) nearest-neighbor search?



F_3 is (d_2, d_3, p, q) - sensitive



use binary search to find
approximate nearest neighbor distance

Course organization

- **Retrieval**
  - Given a query, find "most similar" item in a large data set
  - *Applications*: GoogleGoggles, Shazam, …
- **Supervised learning** (Classification, Regression)
  - Learn a concept (function mapping queries to labels)
  - *Applications*: Spam filtering, predicting price changes, …
- **Unsupervised learning** (Clustering, dimension reduction)
  - Identify clusters, "common patterns"; anomaly detection
  - *Applications*: Recommender systems, fraud detection, …
- **Learning with limited feedback**
  - Learn to optimize a function that's expensive to evaluate
  - *Applications*: Online advertising, opt. UI, learning rankings, …

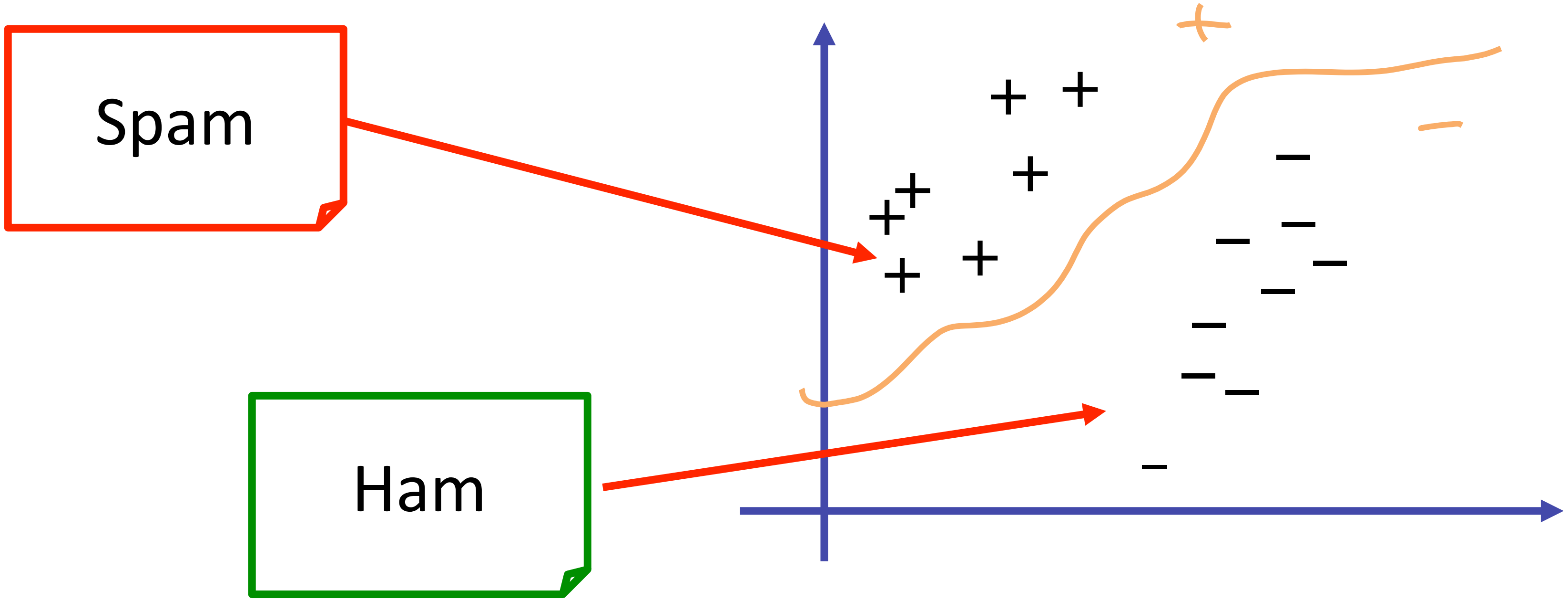
Classification (intuitively)

- Want to assign data points
  - Documents
  - Queries
  - Images
  - Audio
  - …

  a label (spam/not-spam; topic such as sports, politics, entertainment, etc.)

- **Goal**:
  - extract rules (hypotheses) based on training examples.
  - Hope that those rules generalize to previously unseen data

Document classification



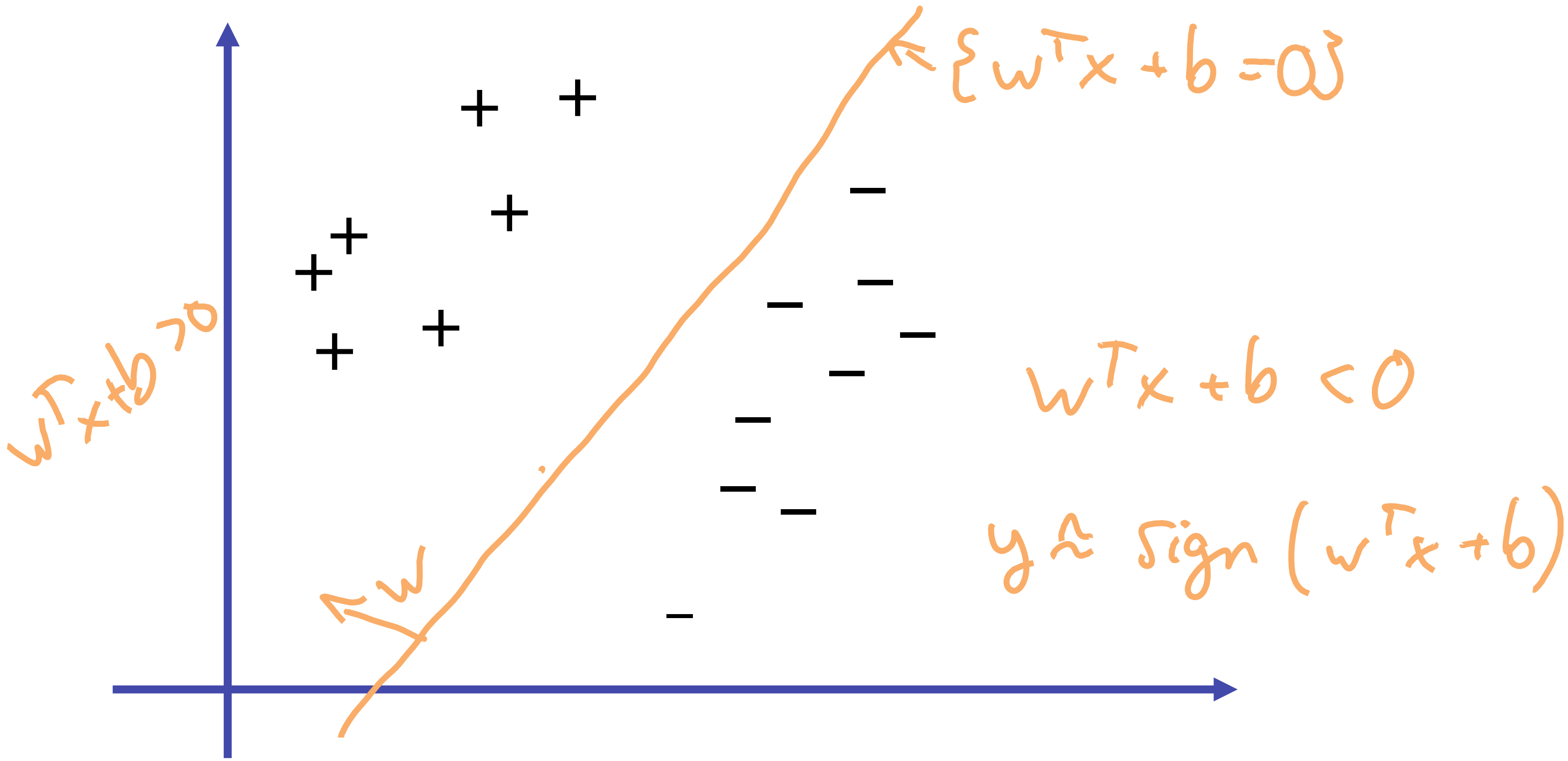
- **Input**: Labeled data set (e.g., rep. bag-of-words) with positive (+) and negative (-) examples
- **Output**: Decision rule (hypothesis)

Linear classifiers

- Data set

$$(x_1, y_1), \ldots, (x_n, y_n)$$

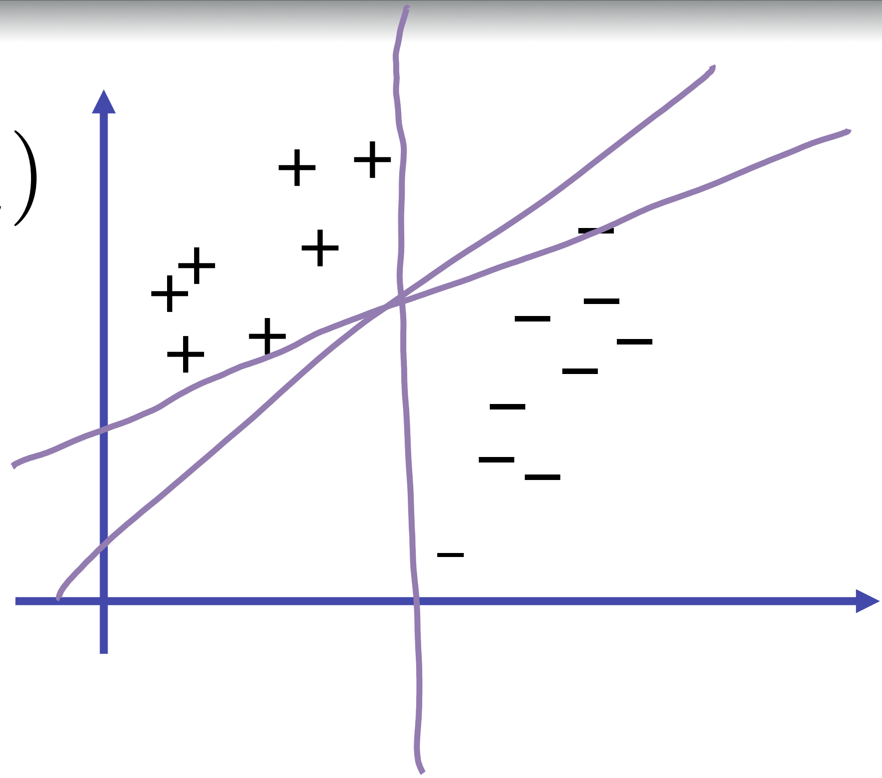
, $x_i \in \mathbb{R}^D$, $y_i \in \{+1, -1\}$



Which linear classifier is the best one?

- Data set

$$(x_1, y_1), \ldots, (x_n, y_n)$$



- Linear classifier:

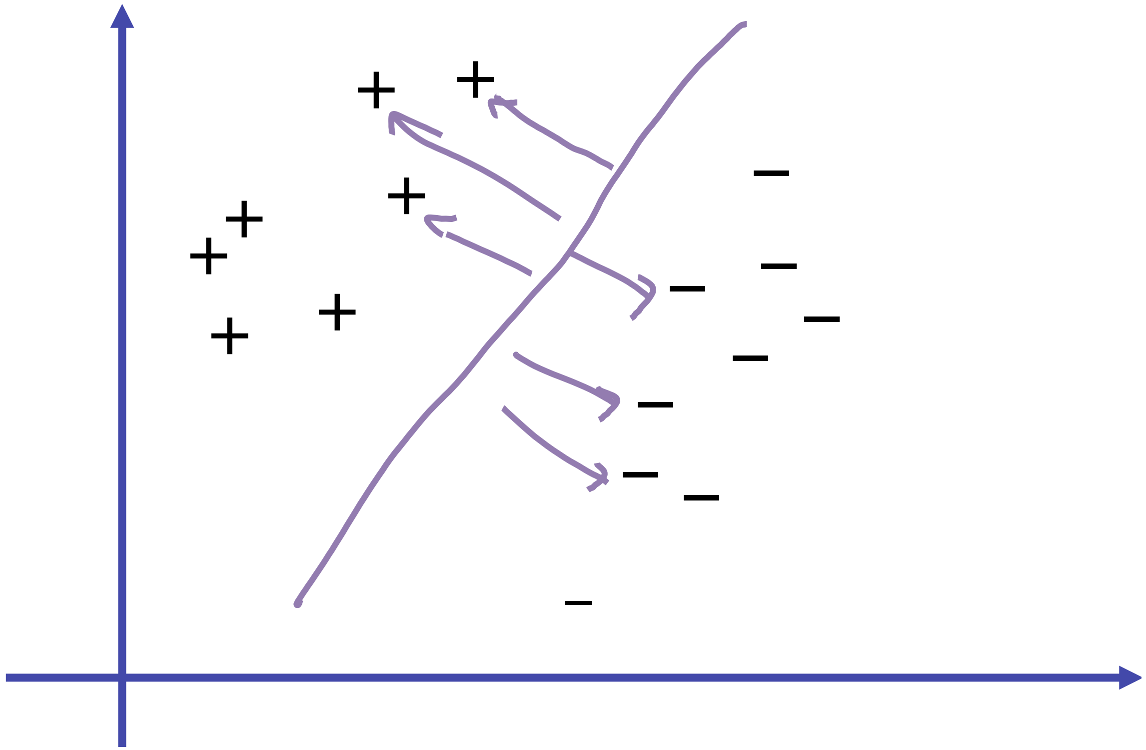
$$\text{sign}(w^T x + b)$$

Large margin classification

- Margin of confidence:

$$y_i (w^T x_i + b) \triangleq \gamma_i$$

$$\max_{w,b} \min_i \gamma_i$$

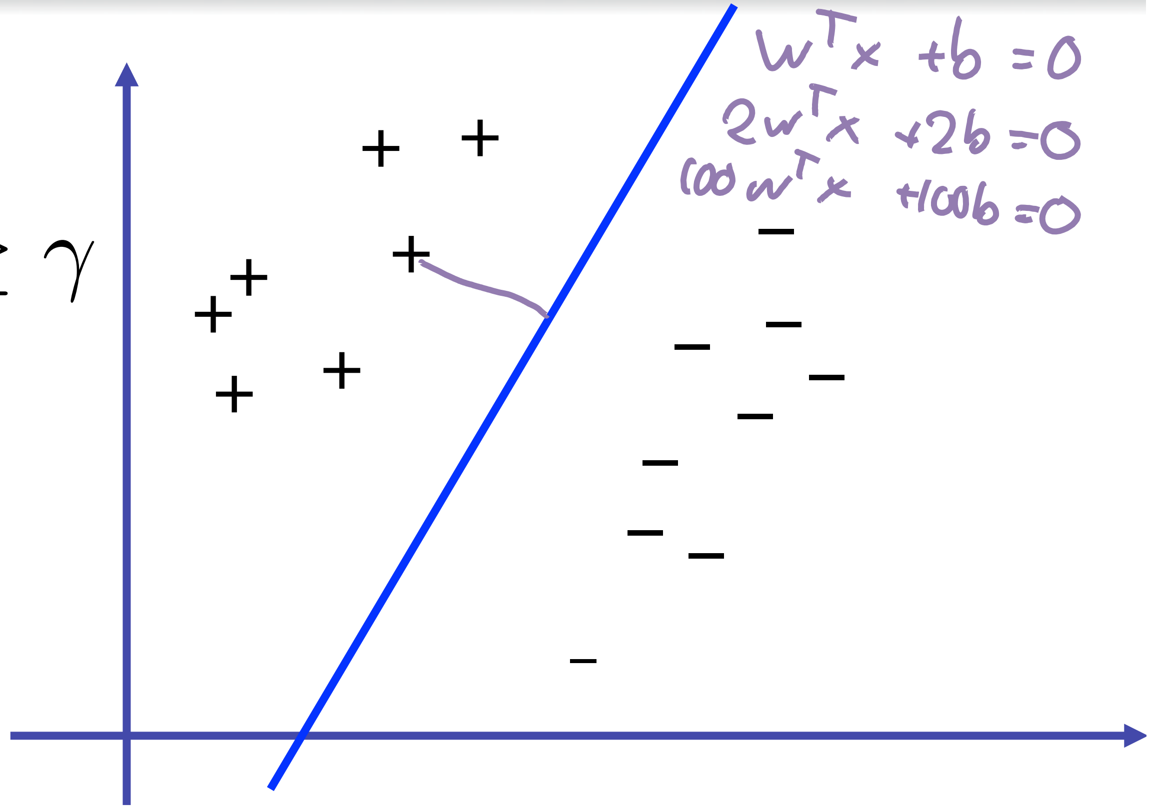


- Want to maximize confidence in our prediction!
- Turns out to be the “right” thing to do
  - Larger margin → Better generalization

Nonuniqueness

$$\max_{w,b,\gamma} \gamma \simeq \infty$$

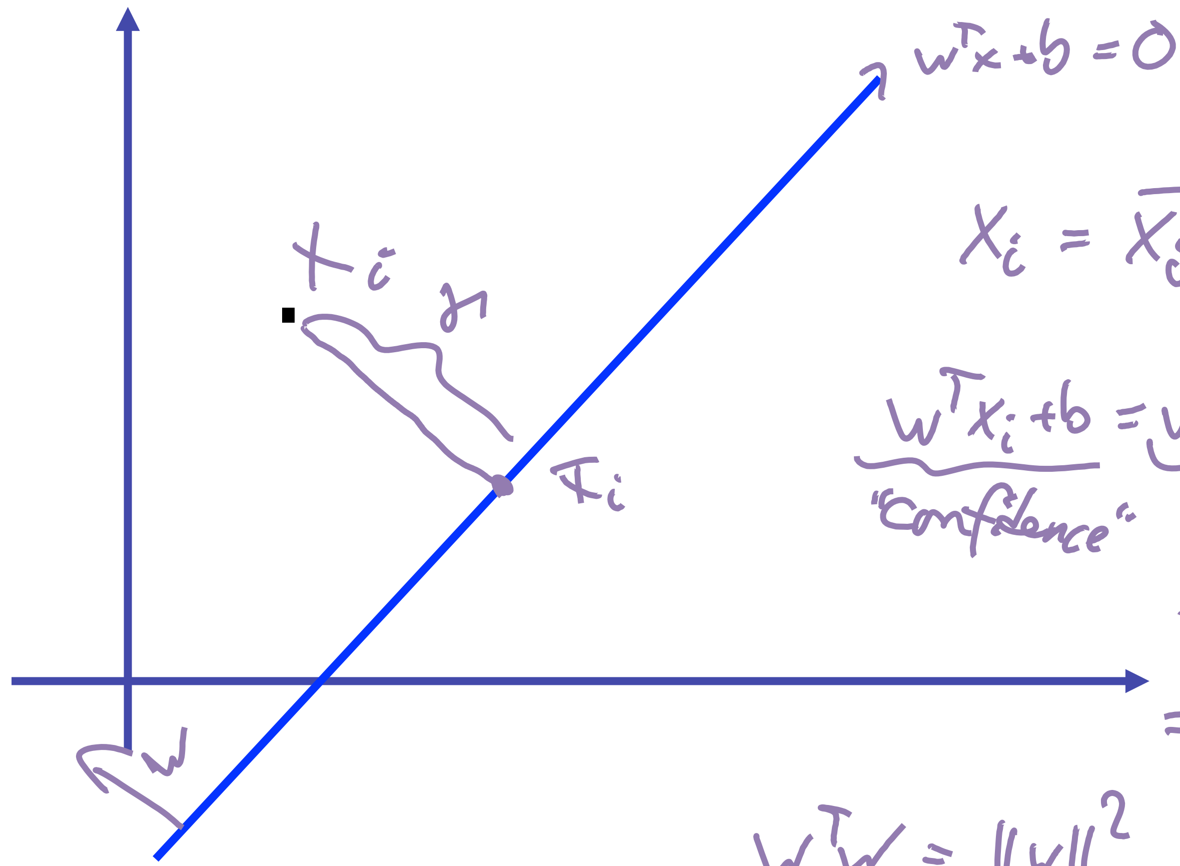
$$\text{s.t.} \underline{(w^T x_i + b) y_i} \geq \gamma$$



So far, our notion of confidence is not yet well defined!

Review: Projection on a plane

$\gamma = \|x_i - \bar{x}_i\|_2$



$$x_i = \bar{x}_i + \gamma \frac{w}{\|w\|}$$

$$\underbrace{w^T x_i + b}_{\text{"confidence"}} = \underbrace{w^T \bar{x}_i + b}_{0} + \gamma \frac{w^T w}{\|w\|}$$

$$= \gamma \underbrace{\frac{w^T w}{\|w\|}}$$

$$= \gamma \|w\|$$

$$w^T w = \|w\|_2^2$$

Canonical hyperplanes

For all positive examples

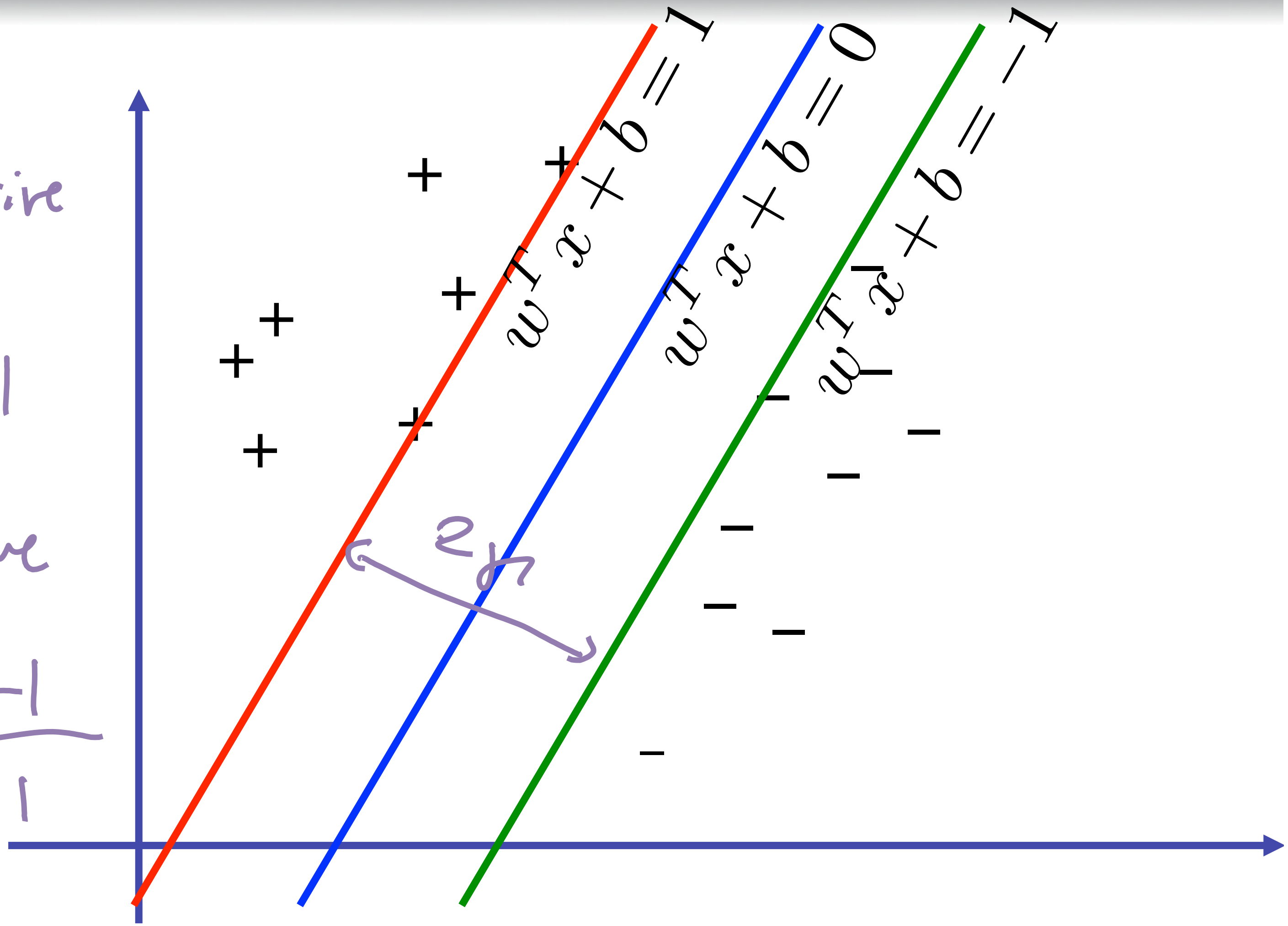
$w^Tx + b \geq 1$

For all negative examples

$w^Tx + b \leq -1$

$y(w^Tx + b) \geq 1$

$\gamma = \frac{1}{\|w\|}$

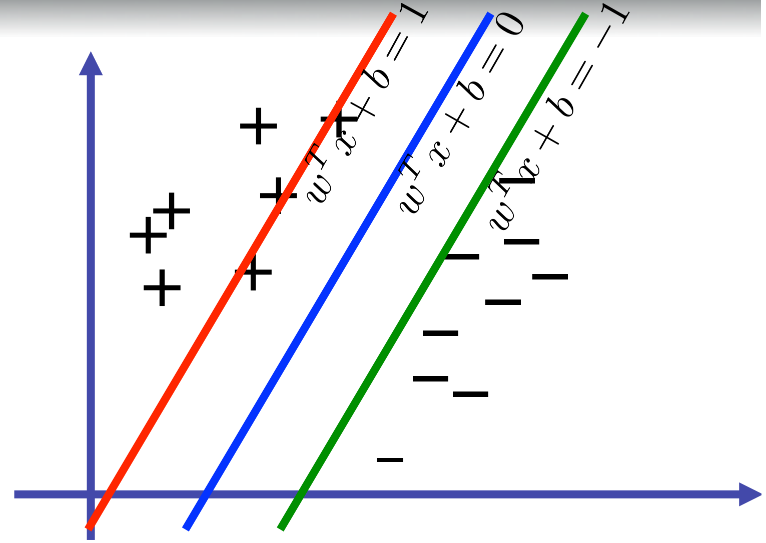


Maximizing the normalized margin

$$\max_{w,b,\gamma} \gamma$$

$$\text{s.t.} \underbrace{(w^T x_i + b) y_i}_{} \geq \cancel{\gamma}\ 1$$

$$\gamma = \frac{1}{\|w\|}$$



$$\equiv \min_{w,b} \|w\|_2$$
$$\text{s.t. } (w^T x_i + b) y_i \geq 1 \quad \forall i$$

$$\equiv \boxed{\begin{array}{l} \min_{w,b} \ w^T w \\ \text{s.t. } (w^T x_i + b) y_i \geq 1 \ \forall i \end{array}}$$

Support Vector Machines

Acknowledgments

- Several slides adapted from the material accompanying the textbook (Anand Rajaraman, Stanford)