# Locality Sensitive Hashing

Debapriyo Majumdar

Indian Statistical Institute Kolkata

# High Dimensional Data: Finding Similar Items

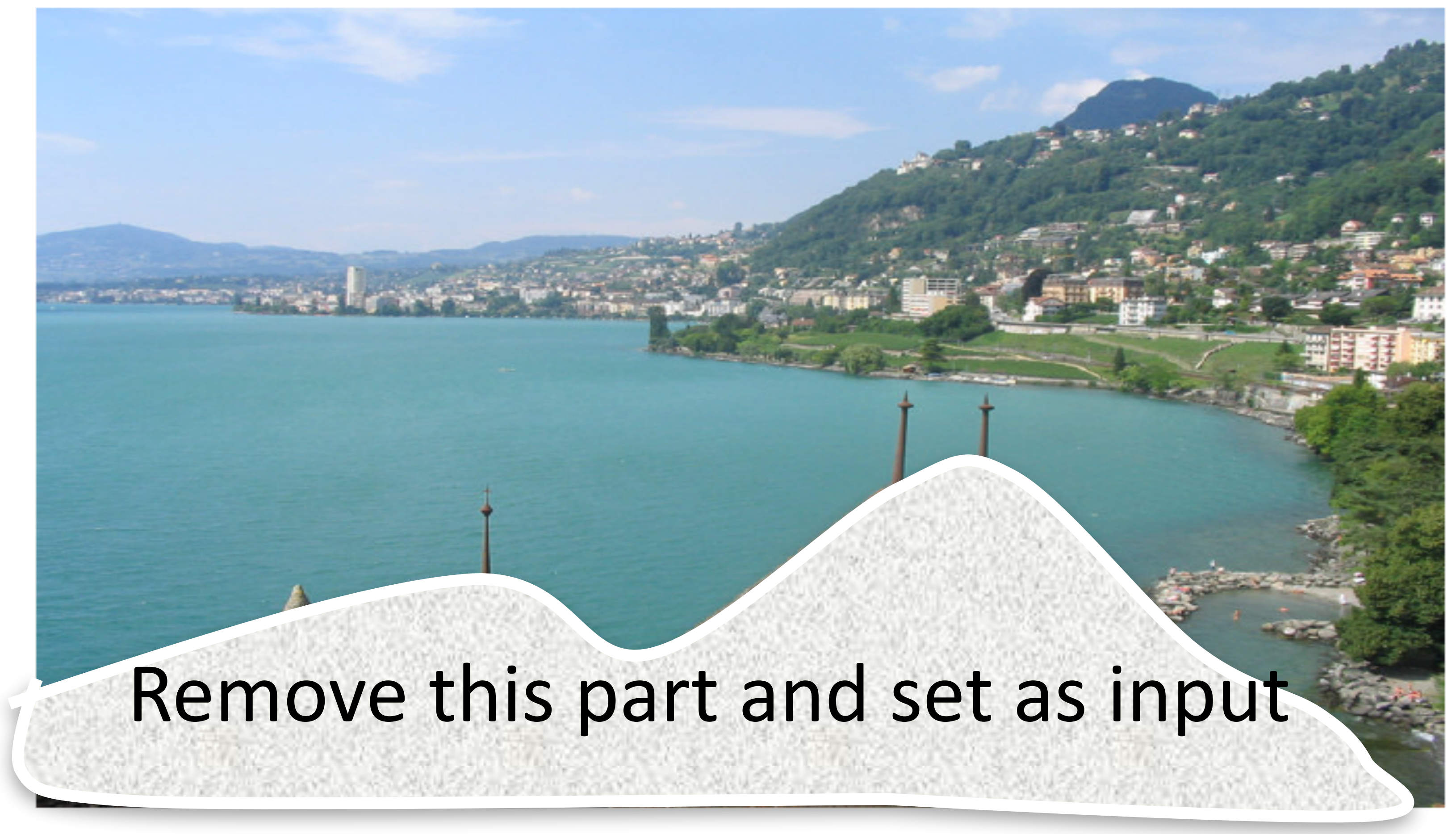
- Product recommendations
  - Millions of products, millions of customers
  - Recommend products bought by ***similar*** customers
- Online advertising
  - Billions of websites, billions of customer actions
  - Advertise pages to users who visited ***similar*** webpages
- Text search, clustering
  - Billions of documents, millions of terms
  - Documents with ***similar*** terms
- Graphics
  - Huge number of image features
  - ***Similar*** image search, scene completion

Key Problem:
**Find similar items**

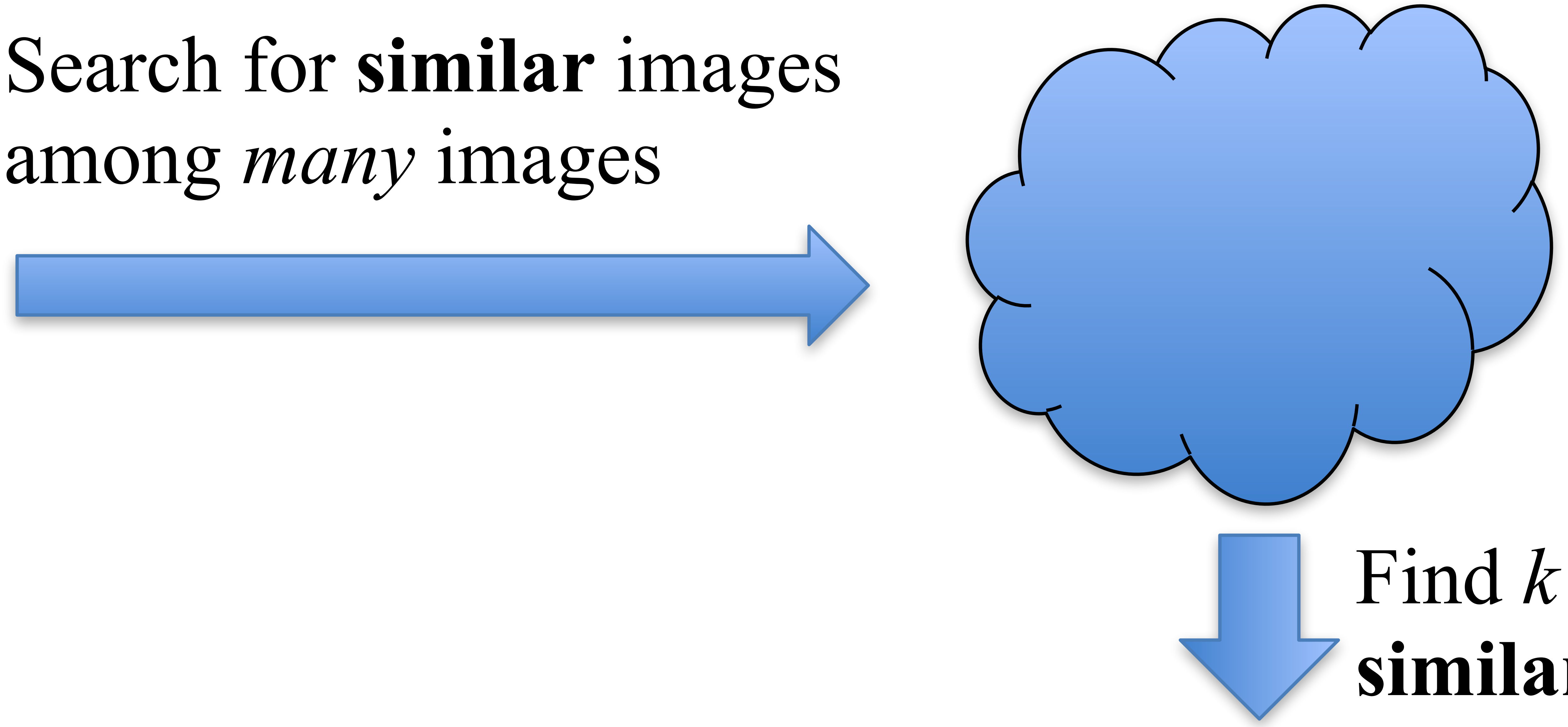
Very high dimensional data

Most algorithms become computationally challenging, or infeasible in very high dimension

## Application: Scene Completion: Hyes and Efros (2007)

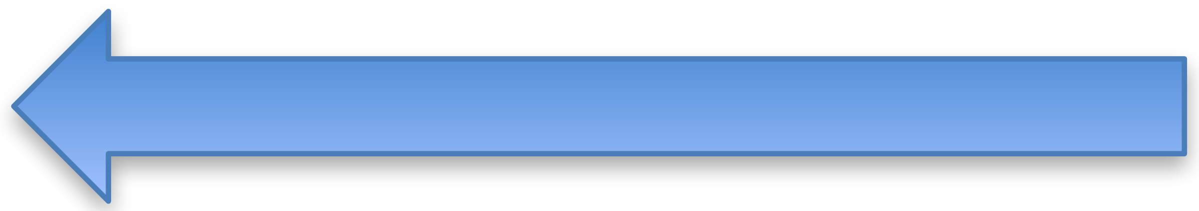


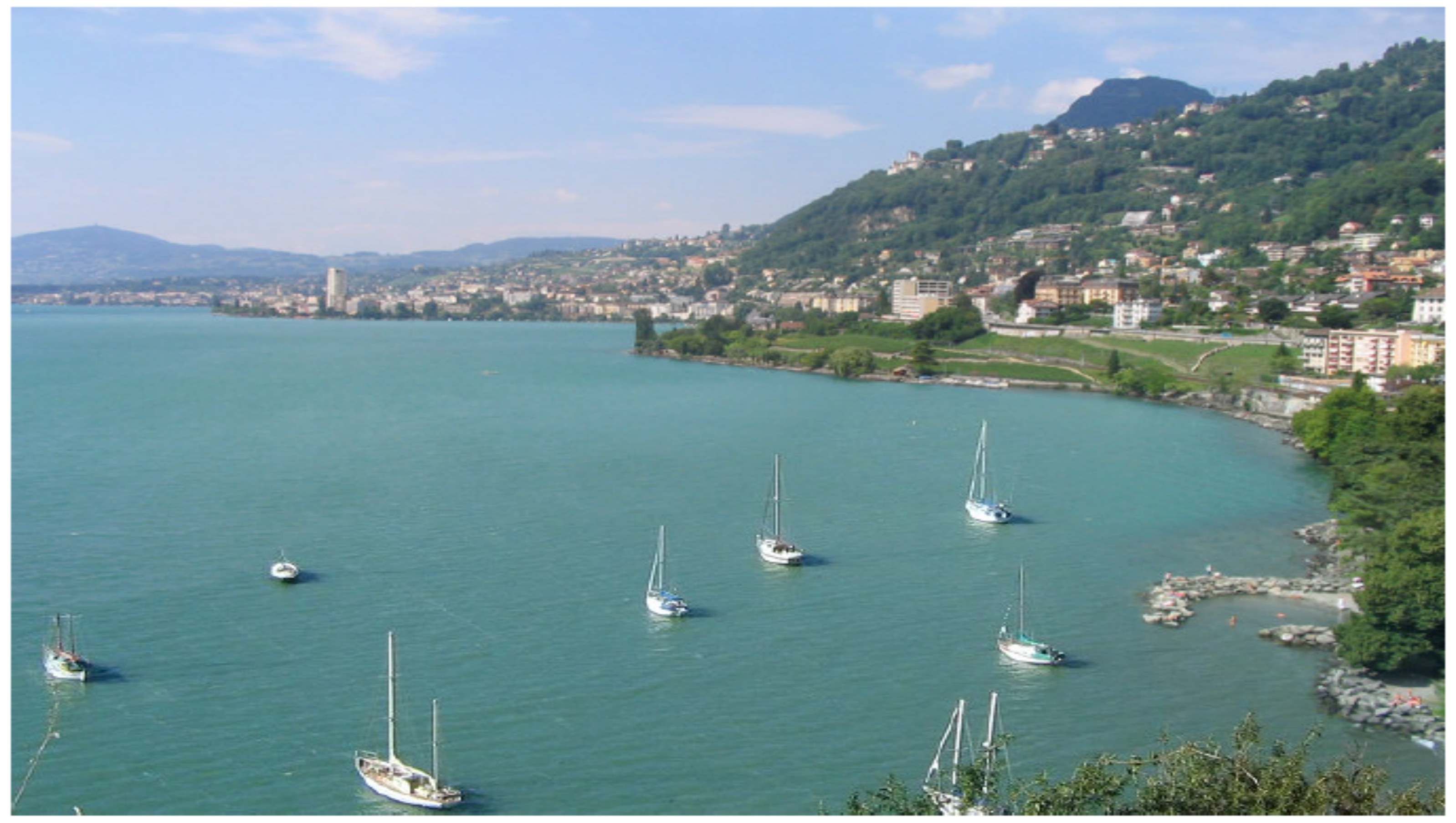
Search for **similar** images among *many* images



Find *k* most **similar** images

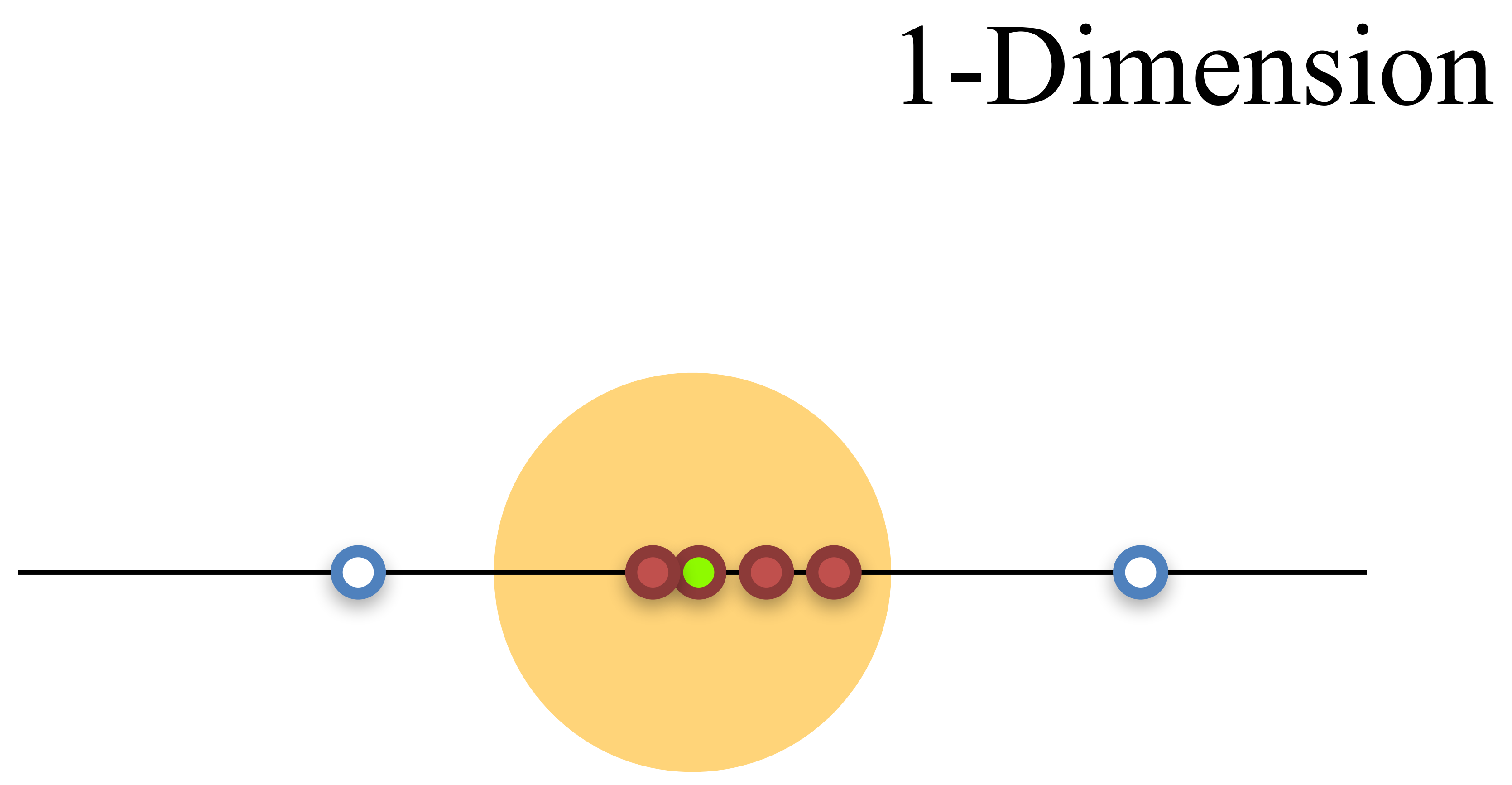
Reconstruct the missing part of the image

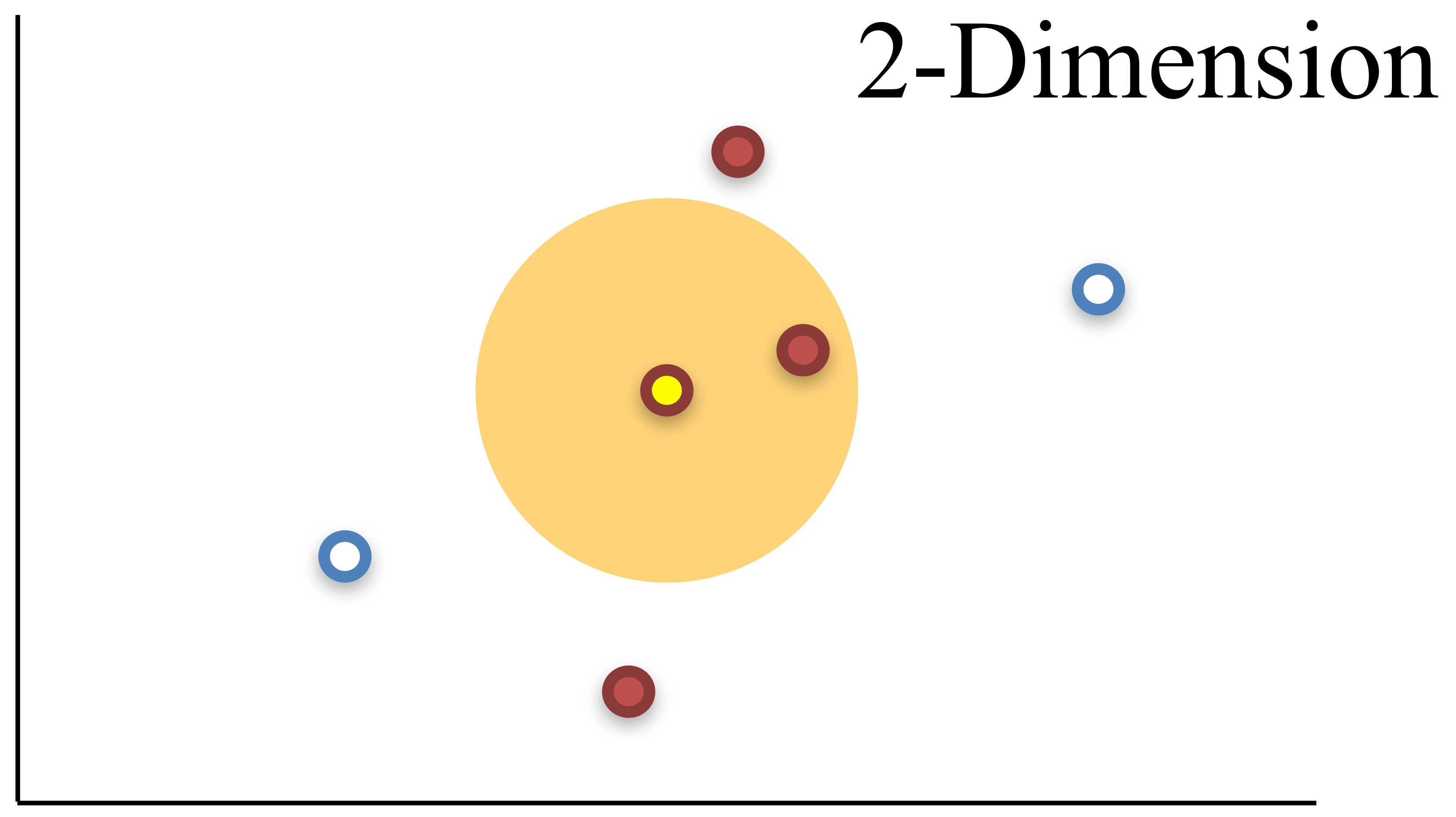




Picture source: http://www.eecs.berkeley.edu/~efros

# The Story of Dimensions





As dimension increases

- The average *distance* between points increases
- Less number of neighbors in the same radius → **Sparse data**

- Intuition:
  - When there is only one choice, all customers buy the same product
  - When there are many choices, different customers tend to differ in their choices

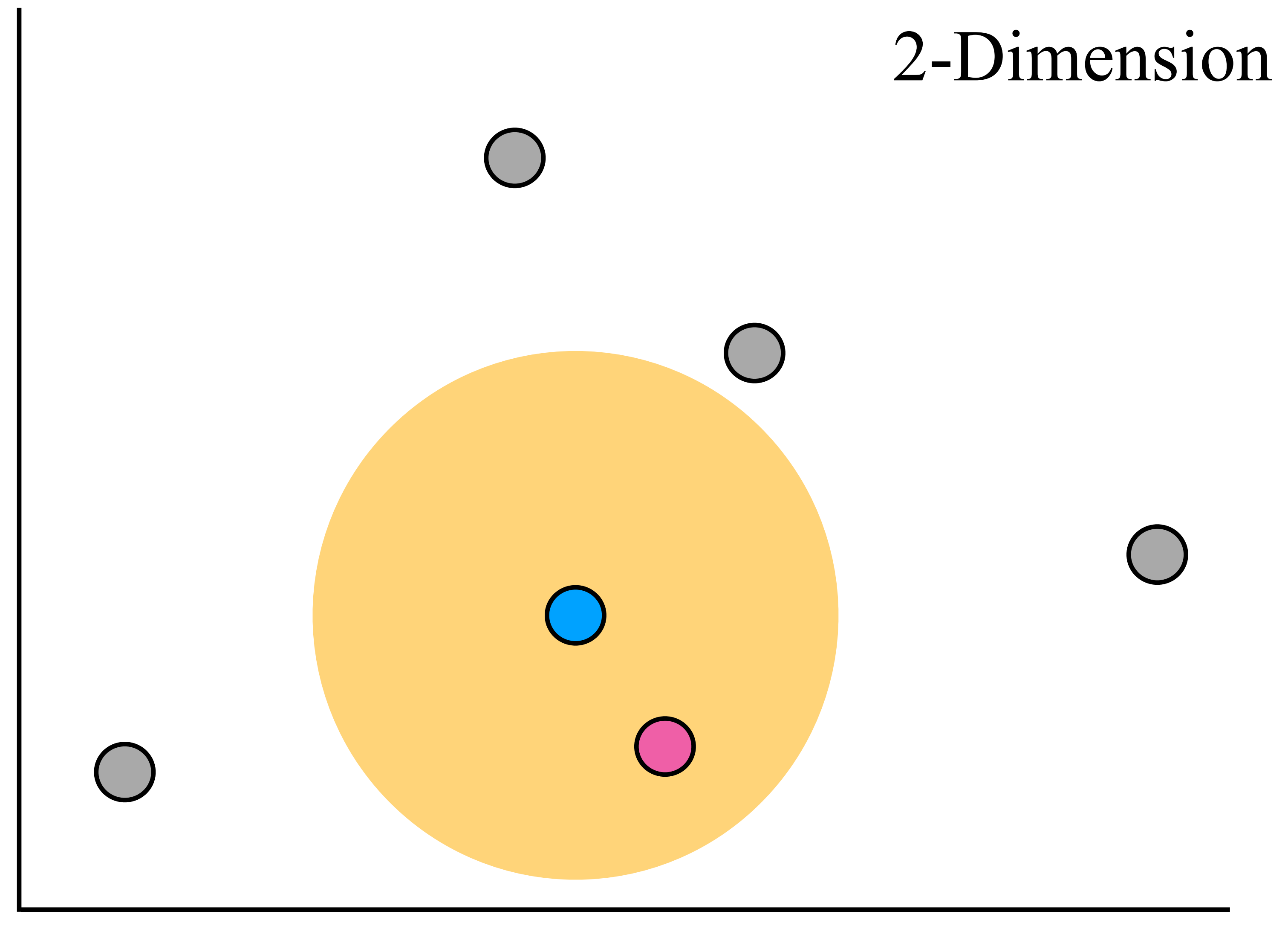
# Data Sparseness

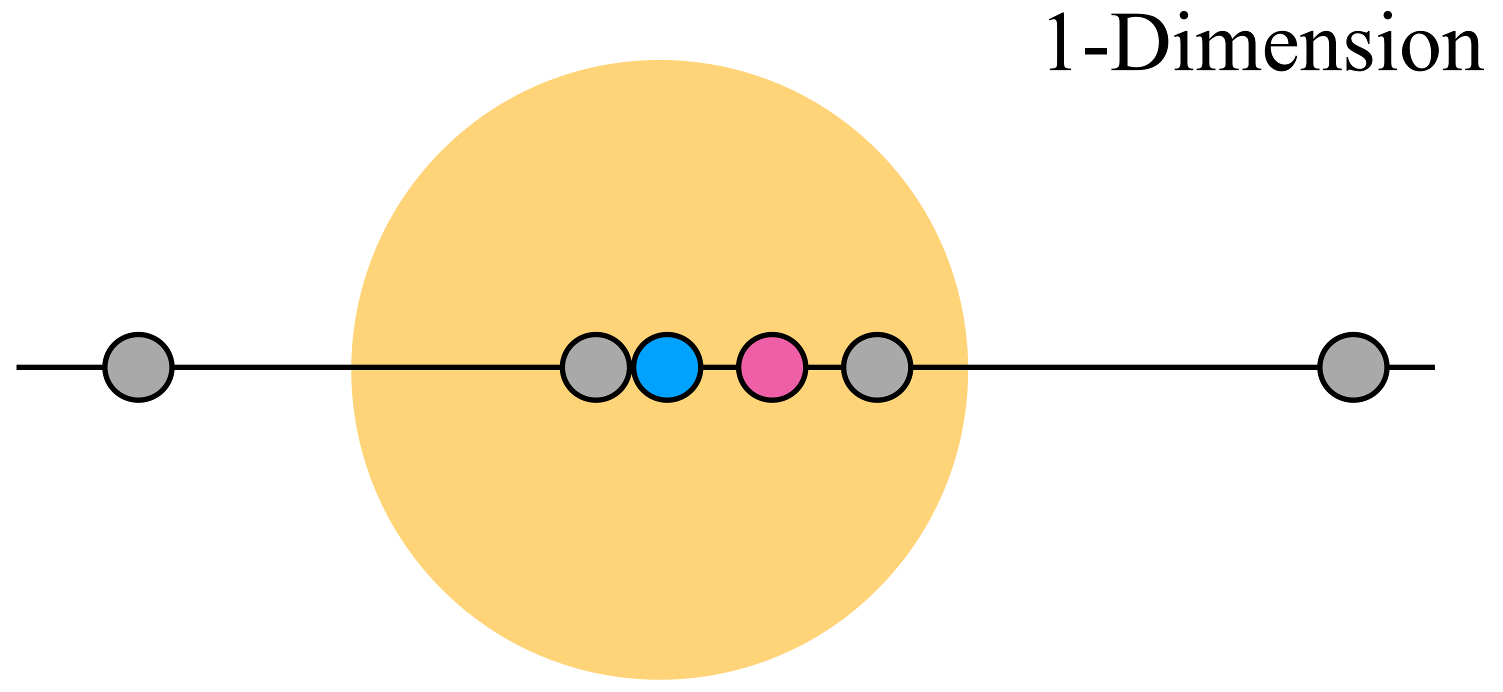
- Product recommendation
  - Most customers do not buy most products
- Online advertising
  - Most users do not visit most pages
- Web search
  - Most terms are not present in most documents
- Graphics
  - Most images do not contain most features

Recall — Key Problem:
**Find similar items**

In high dimension, not only computation becomes harder, but data sparseness poses another challenge for capturing similarity between items

## Preserving distance





- Mapping from higher dimension to lower dimension
  - If two points were near to each other, they remain near to each other
  - If two points were far from each other, the may or may not remain far from each other
- Can we reduce dimensionality in a way so that the distances are preserved?

## Problem setup: find similar documents

- Given a text document, find other documents which are very similar
  - Very similar set of words, or
  - **Several sequences of words overlapping**

- Applications
  - Clustering (grouping) search results, news articles
  - Web spam detection

- Syntactic Clustering of the Web: Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, Geoffrey Zweig (WWW 2007)

## Shingling of Documents

- A document
  - A sequence of words, a canonical sequence of tokens (ignoring formatting, html tags, case)
  - Every document $D$ is a set of subsequences or tokens $S(D,w)$
- Shingle: a contiguous subsequence contained in $D$
- For a document $D$, define its *$w$-shingling* $S(D, w)$ as the set of all unique shingles of size $w$ contained in $D$
  - Example: the 4-shingling of (a,car,is,a,car,is,a,car) is the set { (a,car,is,a), (car,is,a,car), (is,a,car,is) }

# Boolean Matrix Representation of Data

- Columns = documents
- Rows = dimensions / shingles
- Entry ($r$, $c$)

  = 1 if shingle $r$ is in document $c$,

  = 0 if not
- Assume matrix is *sparse* (almost all entries are 0s)

- Every document can be viewed as a set
  - In each dimension, it has a 1 or 0

Note:

The techniques we will see here can be applied to other settings with Boolean representation of data

Customers as rows, products as columns
Features as rows, images as columns

# Jaccard Similarity of Sets

- *Jaccard Similarity* of sets $A$ and $B$

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

- Jaccard similarity is a *metric*

A function $d : X \times X \to [0,\infty)$ such that for all $x, y, z \in X$,

1. $d(x,y) = 0 \iff x = y$
2. $d(x,y) = d(y,x)$
3. $d(x,y) + d(y,z) \geq d(x,z)$

- Examples:
  - $sim(\{0, 1, 5, 8\}, \{0, 5, 11\}) =$
  - $sim(\{0, 1, 5, 8\}, \{2, 7\}) =$
  - $sim(\{0, 1, 5, 8\}, \{0, 1, 5, 8\}) =$

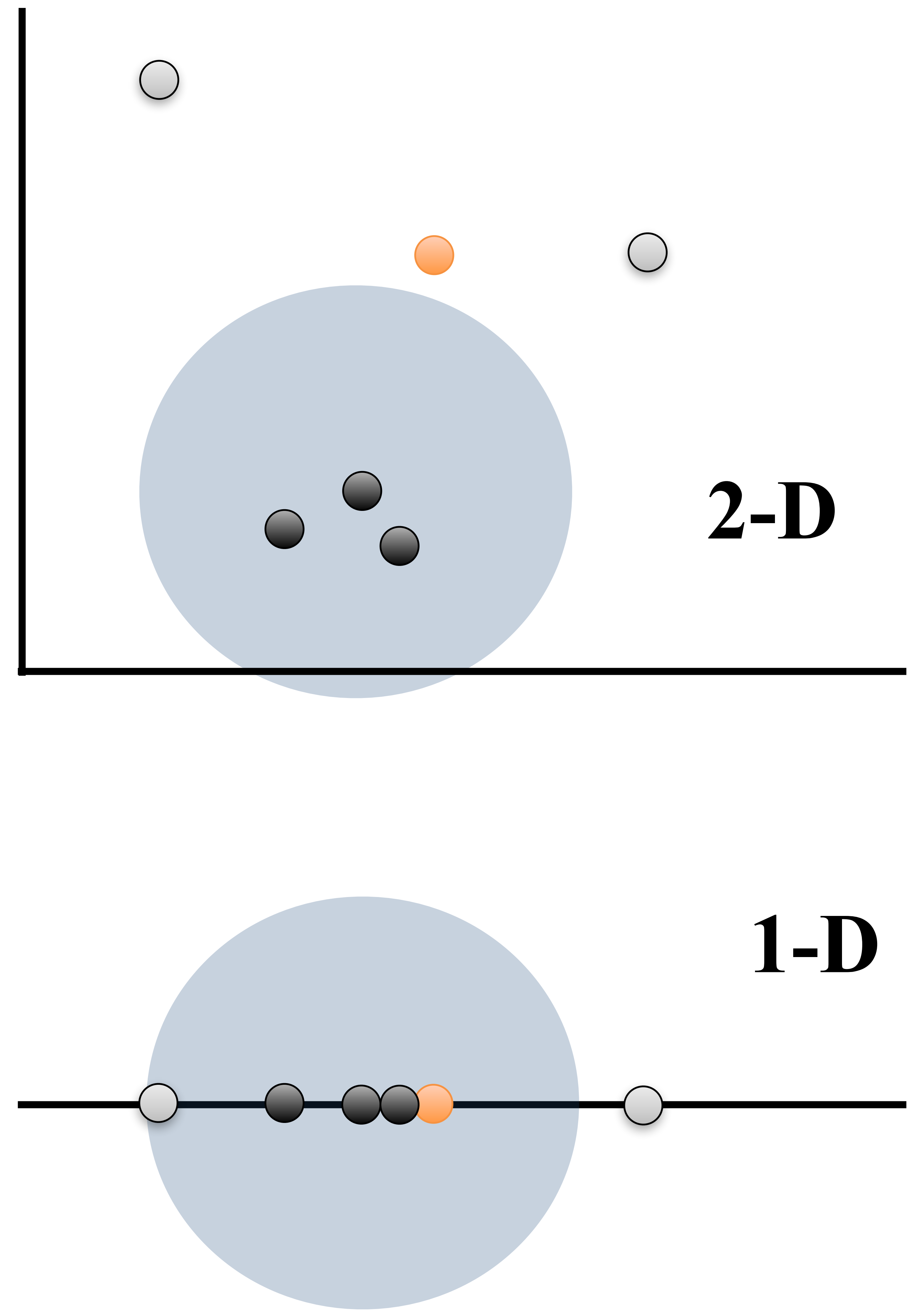
## Brute Force Method

- Problem: Compute pairwise similar documents

- We have: $N$ documents, similarity / distance metric

- Finding similar documents in brute force method is expensive
  - Finding similar documents for one given document: $O(N)$ pairwise computations
  - Finding pairwise similarities for all pairs: $O(N^2)$ pairwise computation

## Locality Sensitive Hashing (LSH): Intuition



- Two points are close to each other in a high dimensional space → They remain close to each other after a "projection" (map)
- If two points are not close to each other in a high dimensional space, they may come close after the mapping
- However, it is quite likely that two points that are far apart in the high dimensional space will preserve some distance after the mapping also

# LSH for Similar Document Search

- Documents are represented as sets of shingles (features)
  - Documents $D_1$ and $D_2$ are points at a (very) high dimensional space (many shingles)
  - Documents as columns (vectors), the set of all documents as a matrix
  - Each row corresponds to a shingle (feature / dimension)
  - The matrix is *very* sparse
- Need a hash function $h$, such that
  - If $d(D_1, D_2)$ is high, then $dist(h(D_1), h(D_2))$ is high, with high probability
  - If $d(D_1, D_2)$ is low, then $dist(h(D_1), h(D_2))$ is low, with high probability
- Then, we can apply $h$ on all documents, put them into hash buckets
- Compare only documents in the same bucket and check if they are actually very similar
- **Note: not a new way to measure similarity; just reducing the number of candidate pairs**

The distance function $d$ in the higher dimension is different from the distance $dist$ in the lower dimension

## Min-Hashing

- Defining the hash function $h$ as:

1. Choose a random permutation $\sigma$ of $m$ = number of dimensions
2. Permute all rows by $\sigma$
3. Then, for an document $D$, $h(D)$ = index of the first row in which $\sigma(D)$ has 1

| | D1 | D2 | D3 | D4 | D5 |
|---|---|---|---|---|---|
| S1 | 0 | 1 | 1 | 1 | 0 |
| S2 | 0 | 0 | 0 | 0 | 1 |
| S3 | 1 | 0 | 0 | 0 | 0 |
| S4 | 0 | 0 | 1 | 0 | 0 |
| S5 | 0 | 0 | 0 | 1 | 0 |
| S6 | 0 | 1 | 1 | 0 | 0 |
| S7 | 1 | 0 | 0 | 0 | 0 |
| S8 | 1 | 0 | 0 | 0 | 1 |
| S9 | 0 | 1 | 1 | 0 | 0 |
| S10 | 0 | 0 | 1 | 0 | 0 |

| $\sigma$ |
|---|
| 3 |
| 1 |
| 7 |
| 10 |
| 6 |
| 2 |
| 5 |
| 9 |
| 8 |
| 4 |

| | D1 | D2 | D3 | D4 | D5 |
|---|---|---|---|---|---|
| S2 | 0 | 0 | 0 | 0 | 1 |
| S6 | 0 | 1 | 1 | 0 | 0 |
| S1 | 0 | 1 | 1 | 1 | 0 |
| S10 | 0 | 0 | 1 | 0 | 0 |
| S7 | 1 | 0 | 0 | 0 | 0 |
| S5 | 0 | 0 | 0 | 1 | 0 |
| S3 | 1 | 0 | 0 | 0 | 0 |
| S9 | 0 | 1 | 1 | 0 | 0 |
| S8 | 1 | 0 | 0 | 0 | 1 |
| S4 | 0 | 0 | 1 | 0 | 0 |

***h*(*D*)**

| D1 | D2 | D3 | D4 | D5 |
|---|---|---|---|---|
| 5 | 2 | 2 | 3 | 1 |

## Property of Min-hash

- How does Min-Hashing help us?
- Do we retain some important information after hashing high dimensional vectors to one dimension?

- Property of MinHash
- The probability that $D_1$ and $D_2$ are hashed to the same value is same as the *Jaccard similarity* of $D_1$ and $D_2$
- In other words, $P[h(D_1) = h(D_2)] = sim(D_1, D_2)$

# Proof

- Consider any two documents $D_1$ and $D_2$
- There are four types of rows
- Let $n_x$ be the number of rows of type $x \in$ {11, 01, 10, 00}
- Note: $sim(D_1, D_2) = \frac{n_{11}}{n_{01} + n_{10} + n_{11}}$
- Now, let $\sigma$ be a *random* permutation. Consider $\sigma(D_1)$ and $\sigma(D_2)$
- Let $j$ be the index of the first 1 either in $\sigma(D_1)$ or in $\sigma(D_2)$ or in both
- Let $x_j$ be the type of the $j$-th row
- Observe: If $h(D_1) = h(D_2) = j$ then $x_j = 11$
- Also, If $x_j = 11$, then $h(D_1) = h(D_2) = j$
- Also, $x_j \neq 00$ in any case
- So: $P[x_j = 11] = \frac{n_{11}}{n_{01} + n_{10} + n_{11}} = sim(D_1, D_2)$

| | D1 | D2 |
|---|---|---|
| Type 11 | 1 | 1 |
| Type 10 | 1 | 0 |
| Type 01 | 0 | 1 |
| Type 00 | 0 | 0 |

$x_j = 11$

| D1 | D2 |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 1 | 1 |
| . | . |
| . | . |

$x_j \neq 11$

| D1 | D2 |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 0 | 1 |
| . | . |
| . | . |
| . | . |
| . | . |

## Using one min-hash function

- *High* similarity documents go to same bucket with *high* probability
- Task: Given $D_1$, find similar documents with at least 75% similarity
- Apply min-hash:
  - Documents which are 75% similar to $D_1$ fall in the same bucket with $D_1$ with 75% probability
  - Those documents do not fall in the same bucket with about 25% probability
  - Missing similar documents and false positives

- We can do better by combining multiple min-hash functions

# Min-hash Signature

**Hundreds, but still less than the number of dimensions**

- Create a signature for a document $D$ using *many* independent min-hash functions
- Compute similarity of columns by the similarity in their signatures
- Observe:
  $E[sim_{SIG}(D_i, D_j)] = sim(D_i, D_j)$
  for any 0 < $i, j$ < $N$ (#documents)

**Signature matrix**

| | | D1 | D2 | D3 | D4 | D5 |
|---|---|---|---|---|---|---|
| SIG(1) | $h_1$ | 5 | 2 | 2 | 3 | 1 |
| SIG(2) | $h_2$ | 3 | 1 | 1 | 5 | 2 |
| SIG(3) | $h_3$ | 1 | 4 | 4 | 1 | 3 |
| | … | … | … | … | … | … |
| SIG($n$) | $h_n$ | … | … | … | … | … |

Example (considering only 3 signatures):

$\text{Sim}_{\text{SIG}}(D_2,D_3) = 1$
$\text{Sim}_{\text{SIG}}(D_1,D_4) = 1/3$

One form of dimension reduction

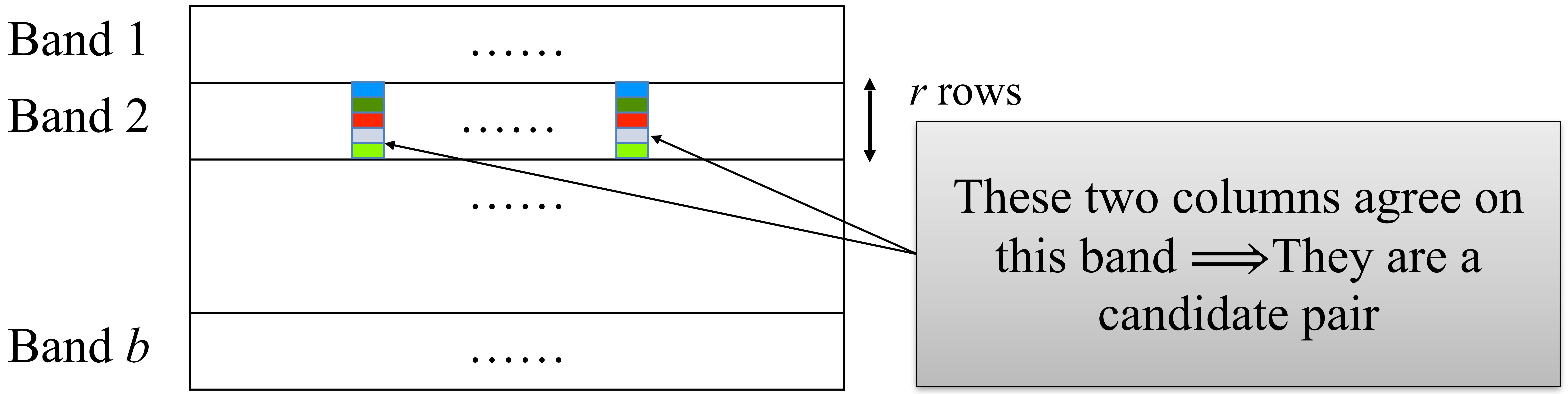
## Locality Sensitive Hashing

- Suppose there is a hashing scheme such that
  - Each hash function: similar documents are *likely* to fall into same bucket, dissimilar documents are *less likely* to fall into same bucket
- Main idea
  - Several hash functions: Dissimilar documents are *very unlikely* to fall into the same bucket for *several hash functions*
  - Two documents fall into the same bucket for several times $\Longrightarrow$ They are likely to be similar
  - Candidate pair: a pair of documents which goes to the same bucket for at least some number of hash functions $\rightarrow$ Compute actual similarity for those pairs

# Locality Sensitive Hashing

- Banding of hash functions: $b$ bands, $r$ rows each
- Signature matrix with $br$ rows (total $br$ hash functions)
- Approach: Two columns agree on at least one band
  $\Longrightarrow$ the corresponding pair of documents will be considered a candidate pair

For example:
$b = 10,\ r = 5,\ n = br = 50$


Band 1
Band 2
Band $b$
……
$r$ rows
These two columns agree on this band $\Longrightarrow$ They are a candidate pair

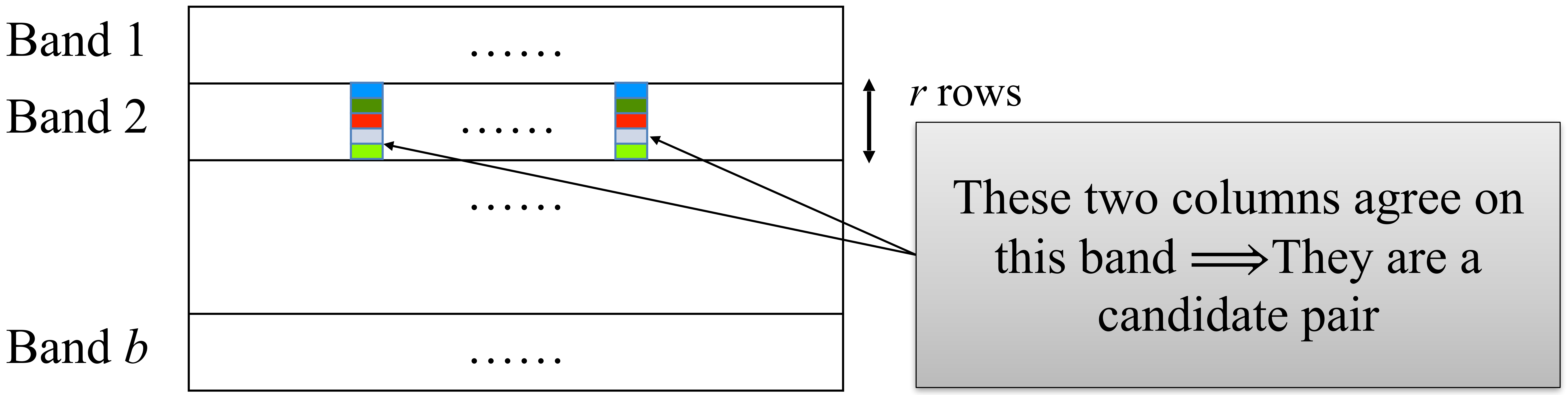
# Locality Sensitive Hashing

A band (portion) of two columns hashing to same bucket

→ High probability that those bands of those columns are identical

→ Signature of two documents matching significantly

→ Candidate pairs


Band 1
Band 2
Band $b$
……
$r$ rows
These two columns agree on this band $\Longrightarrow$ They are a candidate pair

## Computational Challenge

- Computing signature matrix of a large matrix is expensive
  - Picking a random permutation of millions or billions of rows is expensive
  - Sorting / permuting rows explicitly is not feasible
  - However, accessing the data in the order it is available is efficient and feasible
- Solution:
  - Pick a hash function $h : \{1,2,\ldots,m\} \rightarrow \{0,1,\ldots,m-1\}$
  - Some pairs of integers will be hashed to the same value (bucket), some values (buckets) will remain empty
  - Example: $m = 10, h : k \mapsto (3k+1) \mod 10$
  - Maps (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) to (4, 7, 0, 3, 6, 9, 2, 5, 8, 1)
  - *Almost* equivalent to a permutation

# Computing a Signature Matrix

- Pick $n$ different hash functions on the rows (not permutations): $h_1, h_2, \dots, h_n$
- Let SIG($i$,$j$) = the ($i$,$j$)-th entry of the signature matrix ($i$-th hash function, $j$-th document)

```
For each row $r$ BEGIN
  Compute $h_1(r), h_2(r), \dots, h_n(r)$
  For each column $j$ BEGIN
    If the $j$-th column has 1 in row $r$
      For each $i = 1, 2, \dots, n$ BEGIN
        set SIG($i$,$j$) = min{SIG($i$,$j$), $h_i(r)$}
      END
    END IF
  END
END
```

Let $n = 2$

$h_1(r) = (3r + 1) \mod 10$

$h_2(r) = (7r + 1) \mod 10$

| | D1 | D2 | D3 | D4 | D5 |
|---|---|---|---|---|---|
| S1 | 0 | 1 | 1 | 1 | 0 |
| S2 | 0 | 0 | 0 | 0 | 1 |
| S3 | 1 | 0 | 0 | 0 | 0 |
| S4 | 0 | 0 | 1 | 0 | 0 |
| S5 | 0 | 0 | 0 | 1 | 0 |
| S6 | 0 | 1 | 1 | 0 | 0 |
| S7 | 1 | 0 | 0 | 0 | 0 |
| S8 | 1 | 0 | 0 | 0 | 1 |
| S9 | 0 | 1 | 1 | 0 | 0 |
| S10 | 0 | 0 | 1 | 0 | 0 |

| r | h1(r) | h2(r) |
|---|---|---|
| 1 | 4 | 8 |
| 2 | 7 | 5 |
| 3 | 0 | 2 |
| 4 | 3 | 9 |
| 5 | 6 | 6 |
| 6 | 9 | 3 |
| 7 | 2 | 0 |
| 8 | 5 | 7 |
| 9 | 8 | 4 |
| 10 | 1 | 1 |

Exercise: do the computation

| | D1 | D2 | D3 | D4 | D5 |
|---|---|---|---|---|---|
| h1 | | | | | |
| h2 | | | | | |

## Analysis of LSH

**Signature: $n$ (hash functions)$\times N$, $b$ bands, $r$ rows per band**

For two documents, let the similarity be $s$

P[Signature agree in all rows of one particular band]

$= s^r$

P[Signature don't agree in at least one row of one particular band]

$= 1 - s^r$

P[For each band, signature don't agree in at least one row]

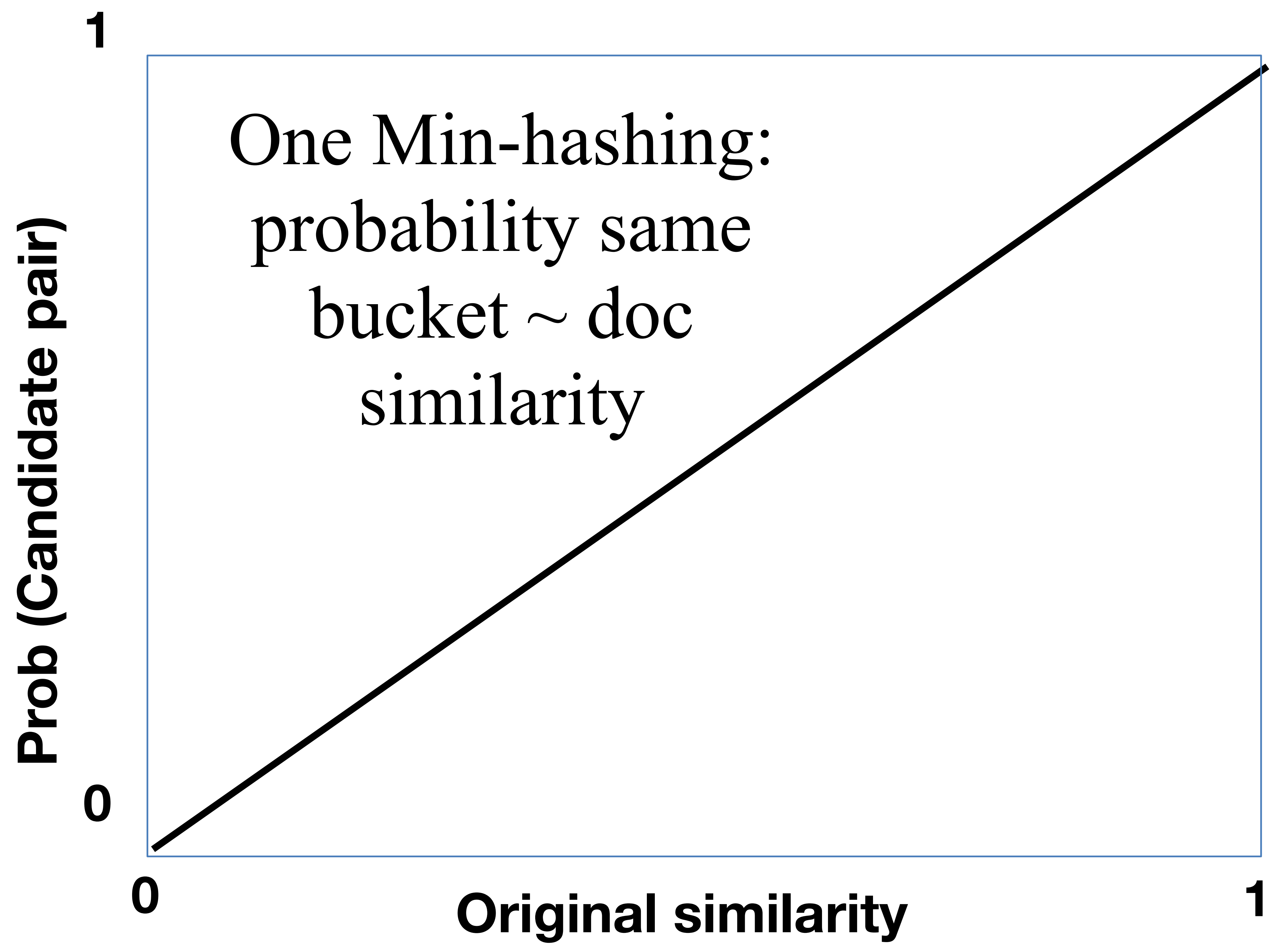
$= (1 - s^r)^b$

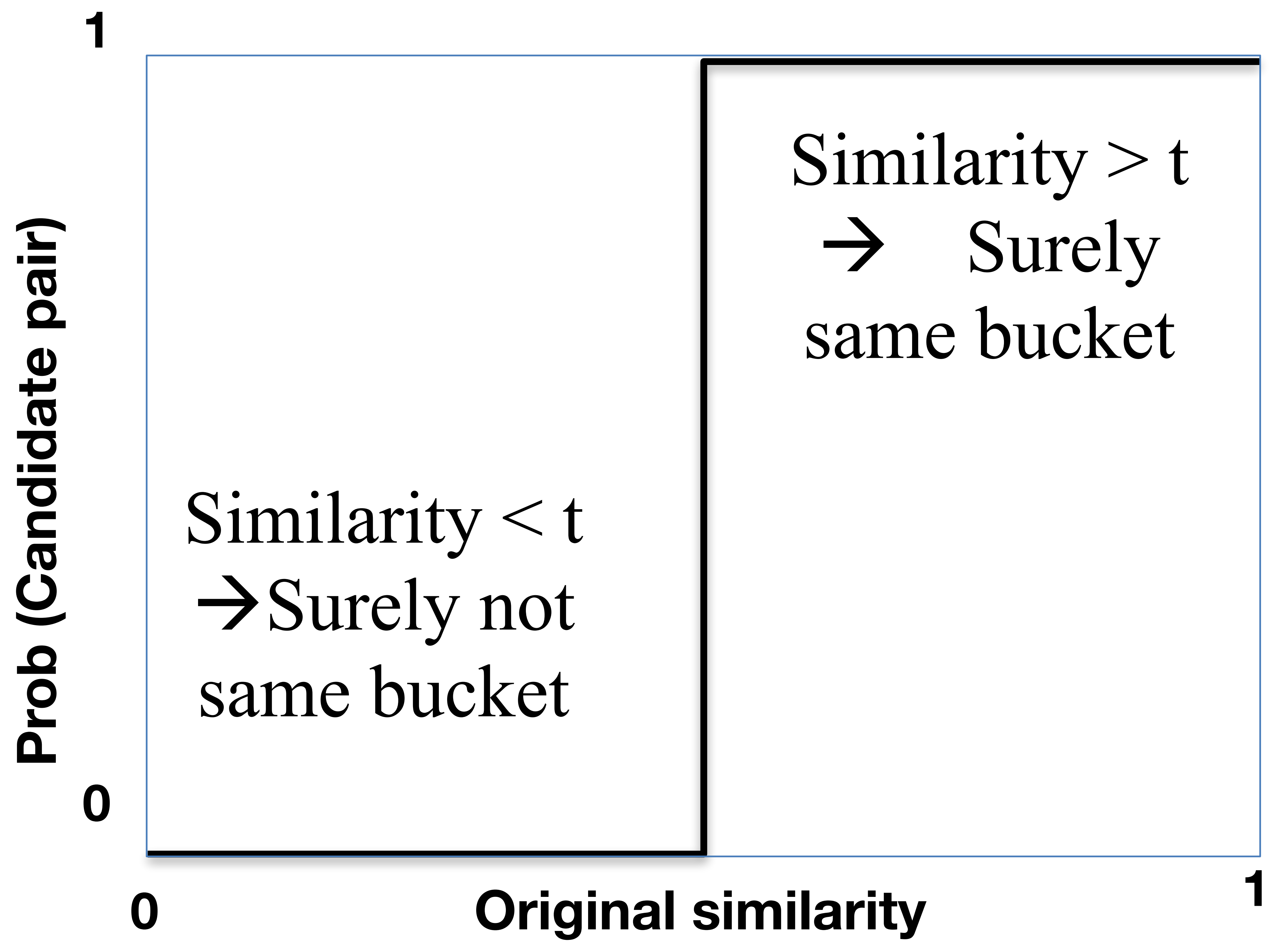
P[Signature agree in all rows of at least one band]

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

# What we want

Tune $r$ and $b$ to get the desired step function

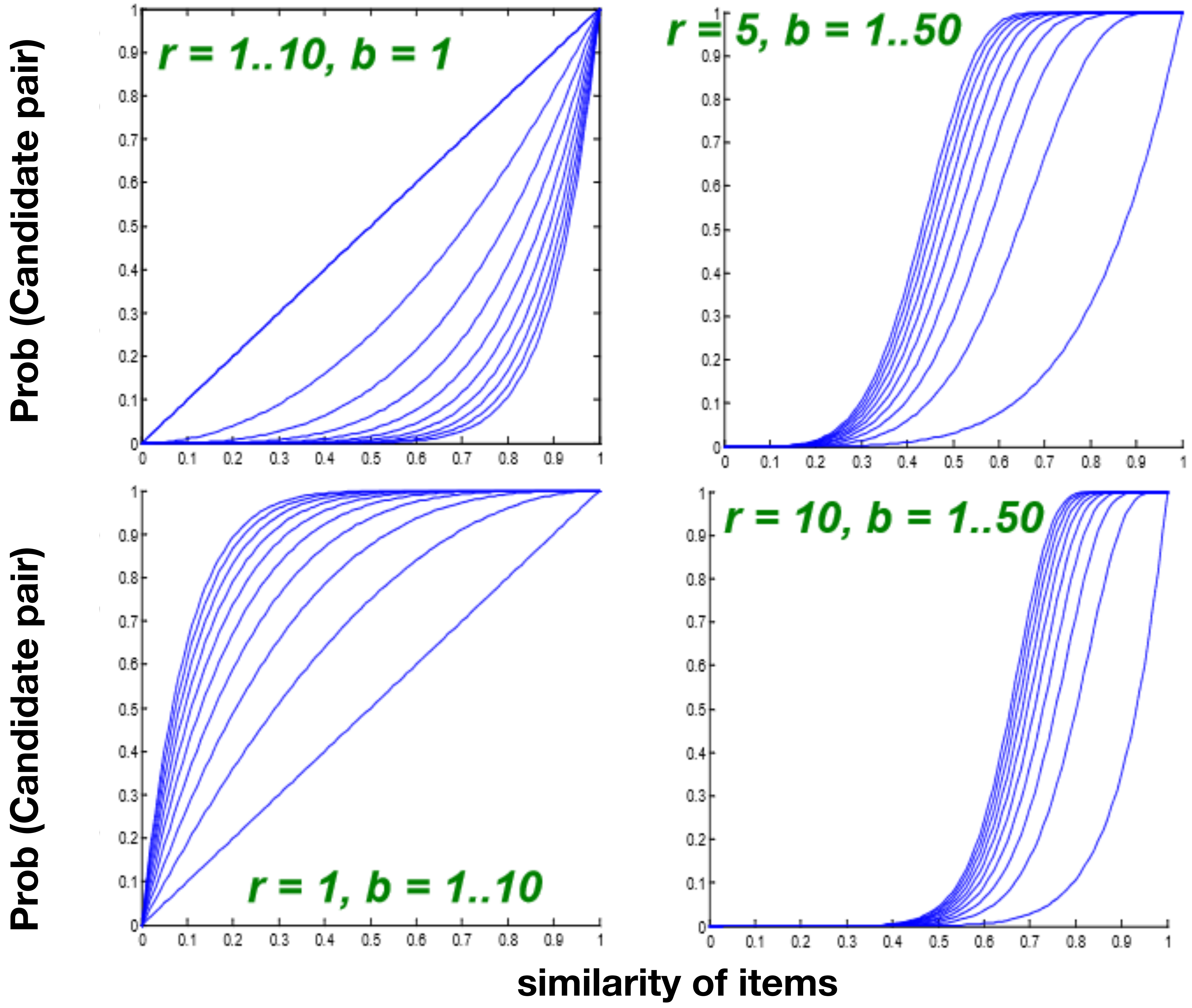




Credit for the plots: Jure Leskovec, Stanford University

## Tuning *b* and *r*

**By tuning *b* and *r* we can get a desired step function**



# Generalization: LSH Family of Functions

- Conditions for the family of functions

1. Declares closer pairs as candidate pairs with higher probability than a pair that are not close to each other
2. Statistically independent: product rule for independent events can be used
3. Efficient in identifying candidate pairs much faster than exhaustive pairwise computation
4. Efficient in combining for avoiding false positives and false negatives

## General Definition

- Let $d_1 < d_2$ be two distances (say between two pairs of points)
- A family $\mathscr{F}$ of functions is said to be $(d_1, d_2, p_1, p_2)$-sensitive, if for every $f \in \mathscr{F}$, and for some $0 \leq p_1, p_2 \leq 1$ we have:

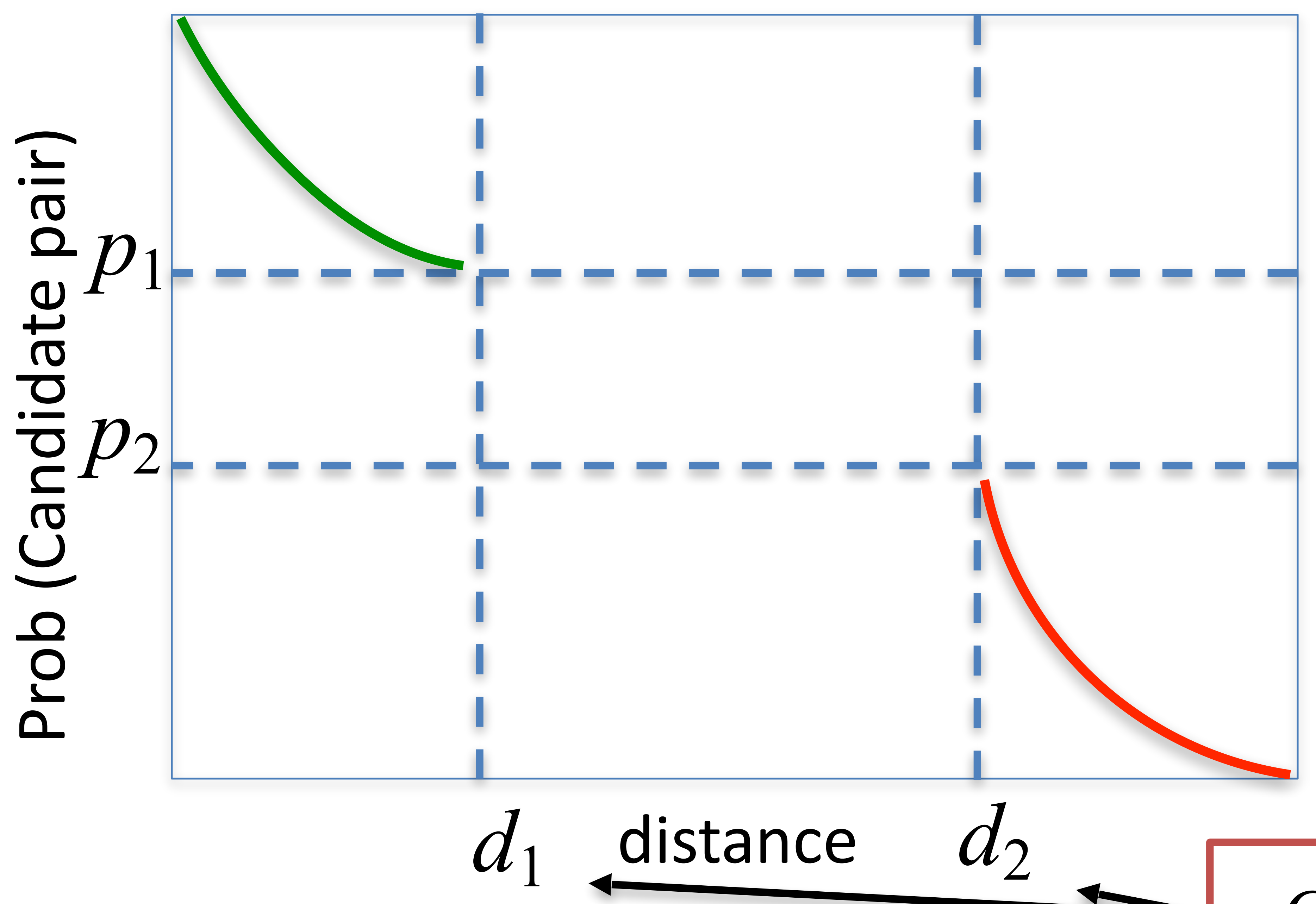
1. If $d(x,y) \leq d_1$ then $P[f(x) = f(y)] \geq p_1$
2. If $d(x,y) \geq d_2$ then $P[f(x) = f(y)] \leq p_2$

If two points are close enough, then probability that they are mapped to the same value is high enough

If two points are far enough, then probability that they are mapped to the same value is small enough

Like to make $p_1$ and $p_2$ far apart

Cannot say anything about some $d$ if $0 \leq d_1 \leq d \leq d_2 \leq 1$



Given

## LSH families of functions

- The family of min-hash functions is $(d_1, d_2, 1 - d_1, 1 - d_2)$-sensitive for any $d_1$ and $d_2$ such that $0 \leq d_1 \leq d_2 \leq 1$
- Hamming distance between two bit-vectors: the number of dimensions (bits) in which they *differ*
  - Define a function $f_i$ on vectors of $m$-dimension: $v \mapsto v_i$ (the $i$-th bit of the vector)
  - Then, $f_i(v) = f_i(w)$ for two bit-vectors $v$ and $w$ if they agree on the $i$-th bit
  - If $i$ is randomly chosen, then $P[f_i(v) = f_i(w)] = 1 - h(v, w)/m$ where $h$ is the hamming distance
  - The family of functions $\mathscr{F} = \{f_1, \dots, f_m\}$ is $(d_1, d_2, 1 - d_1/m, 1 - d_2/m)$ sensitive for any $d_1$ and $d_2$ such that $0 \leq d_1 \leq d_2 \leq 1$

# Amplifying an LSH Family

- Suppose $\mathscr{F}$ is a family $(d_1, d_2, p_1, p_2)$-sensitive functions
- Then another family $\mathscr{F}'$ of functions can be constructed from $\mathscr{F}$ such that $\mathscr{F}'$ is $(d_1, d_2, p_1^r, p_2^r)$ sensitive for some integer $r > 0$
- *AND-Construction*:

Fix any $0 < r < | \mathscr{F} |$

Define each $f \in \mathscr{F}'$ such that for some set of $r$ indices $i = 1, \ldots, r$:

$$f(x) = f(y) \Longleftrightarrow f_i(x) = f_i(y) \text{ for all } i = 1, \ldots, r$$

Now:

1. If $d(x,y) \leq d_1$ then $P[f_i(x) = f_i(y)] \geq p_1 \forall i \implies P[f(x) = f(y)] \geq p_1^r$
2. If $d(x,y) \geq d_2$ then $P[f_i(x) = f_i(y)] \leq p_2 \forall i \implies P[f(x) = f(y)] \leq p_2^r$

Since the functions $f_i$ are independent

## Amplifying

- The AND Construction is the effect of combining $r$ rows into a single band
  - Two documents form a candidate pair if and only if they are hashed to the same bucket in all rows of the band
- Similarly, an OR-Construction gives us a $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-sensitive family
  - The effect of $b$ bands
  - Two documents form a candidate pair if and only if they are hashed to the same bucket in at least one band
- The AND-construction lowers probabilities, OR-construction increases probabilities
- With carefully chosen $r$ and $b$
  - For AND, push $p_2$ very close to 0, keep $p_1$ significantly higher
  - For OR, push $p_1$ very close to 1, keep $p_2$ significantly lower

# References and further reading

- General reference: "Mining of Massive Datasets" by Leskovec, Rajaraman and Ullman
- Other citations are given in the respective slides

- Further reading: Sections 3.7 and 3.8 of the book