Locality-sensitive hashing

2IMW30 - Foundations of data mining
TU Eindhoven, Quartile 3, 2016

Anne Driemel

Overview of this lecture

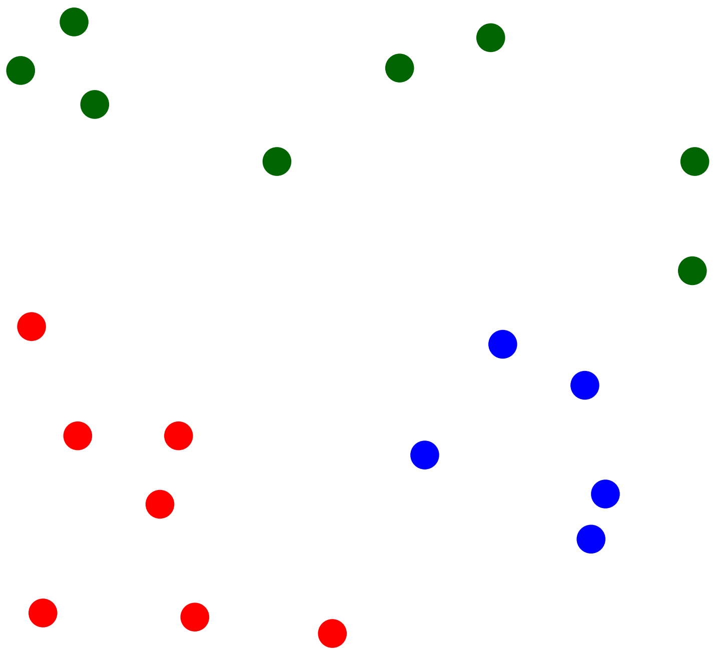
- Nearest-Neighbor rule
- Locality sensitive hashing
- Cosine distance
- Euclidean distance
- Jaccard Similarity
- Minhashing
- Banding
- Amplification

Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.

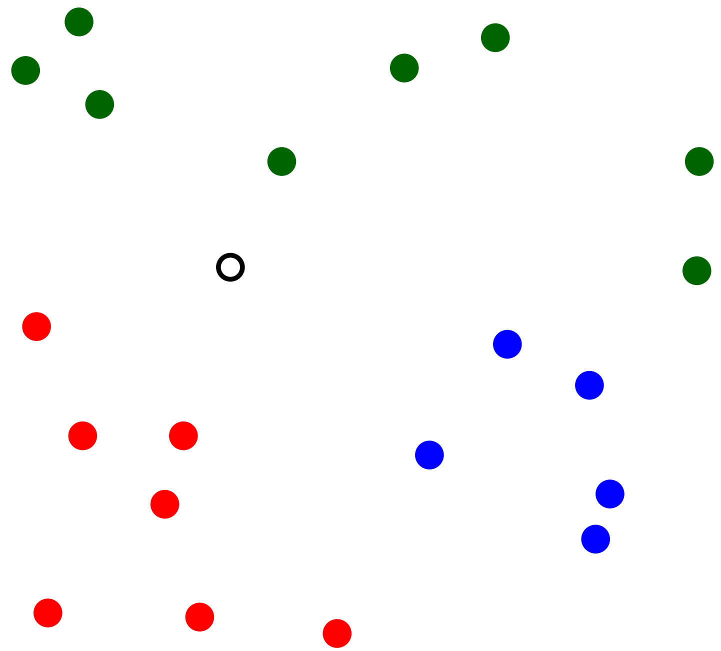
Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.



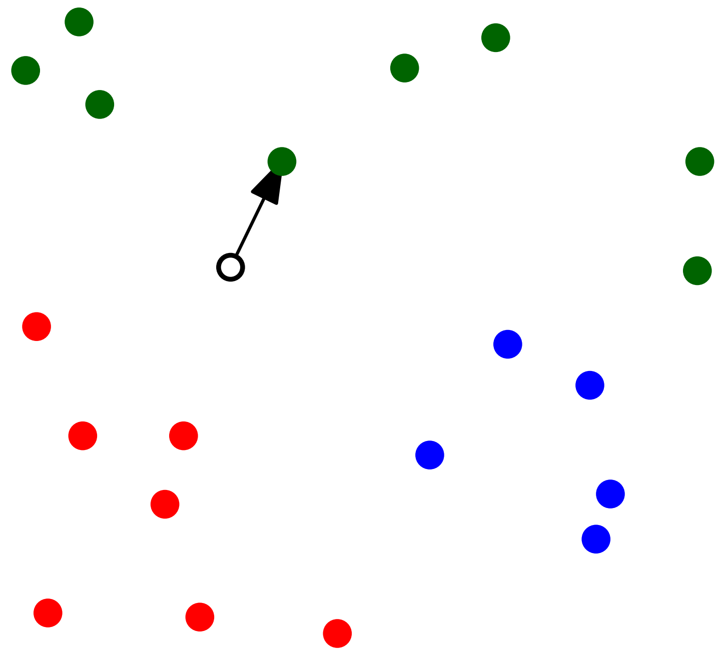
Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.



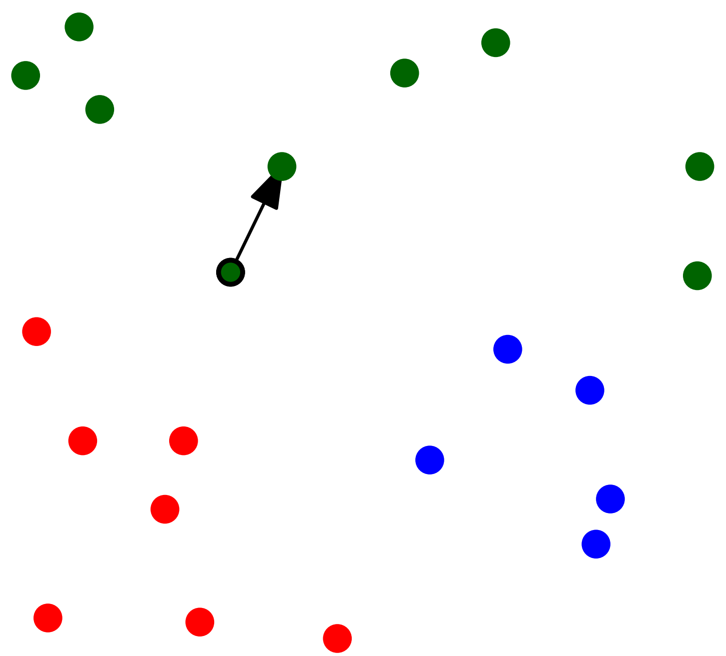
Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.



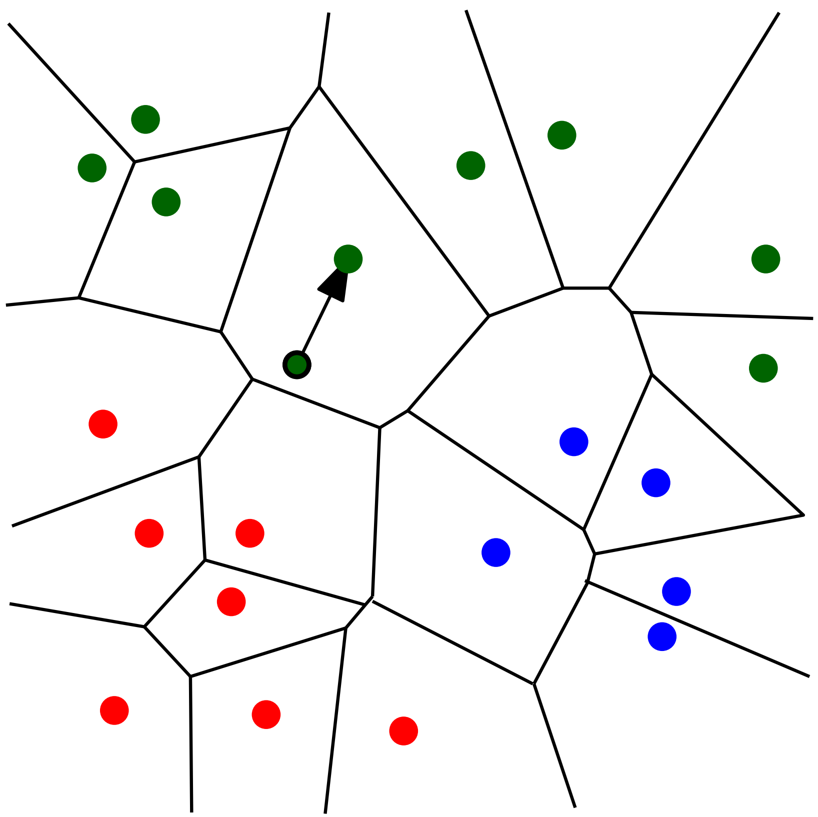
Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.



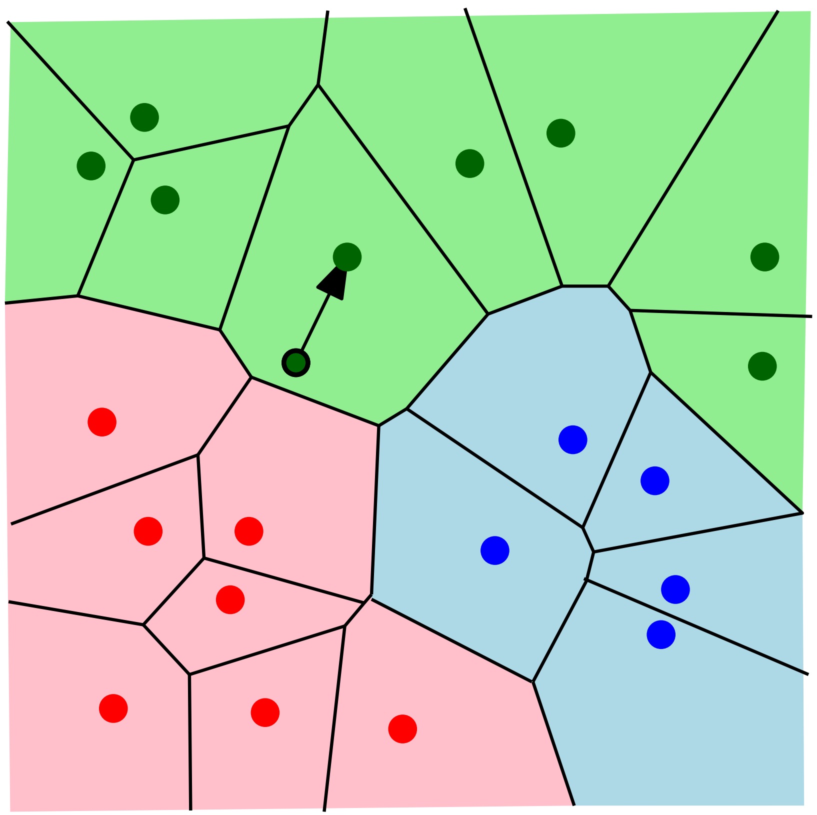
Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.



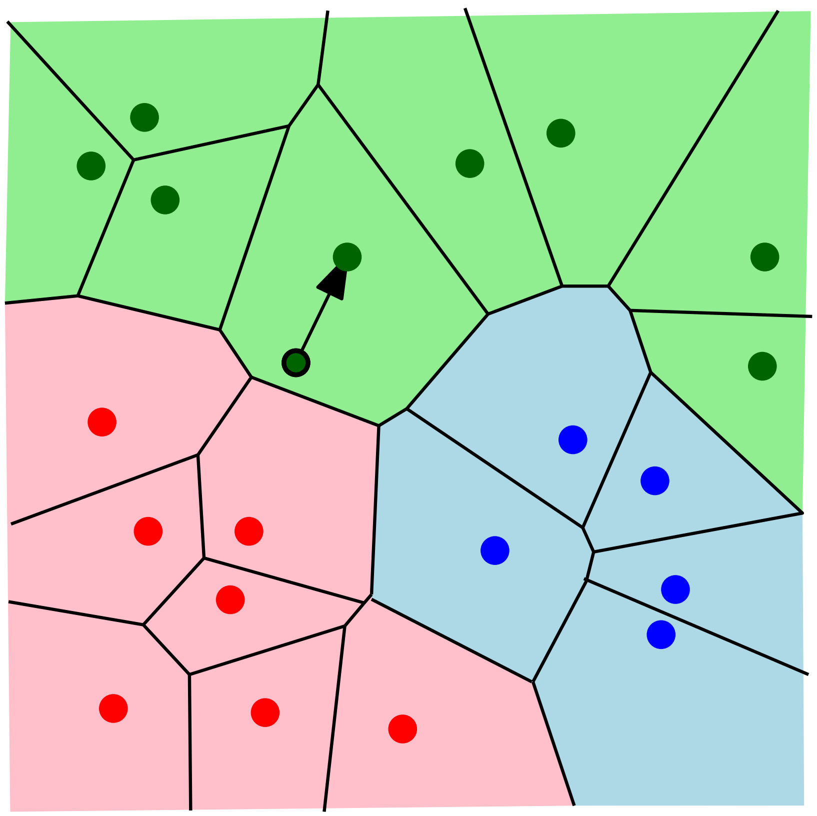
Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.



Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.

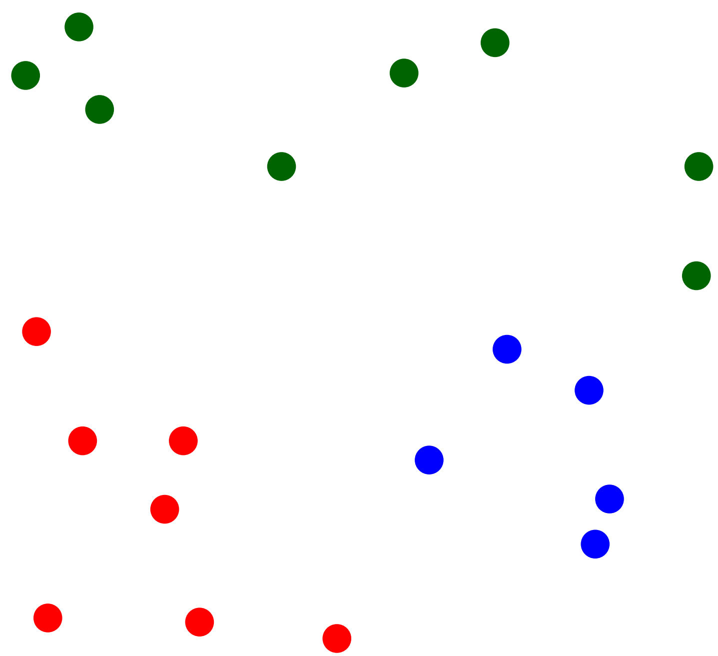


This induces a Voronoi partition with exponential growth in complexity

Can we use a random partition instead?

Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.

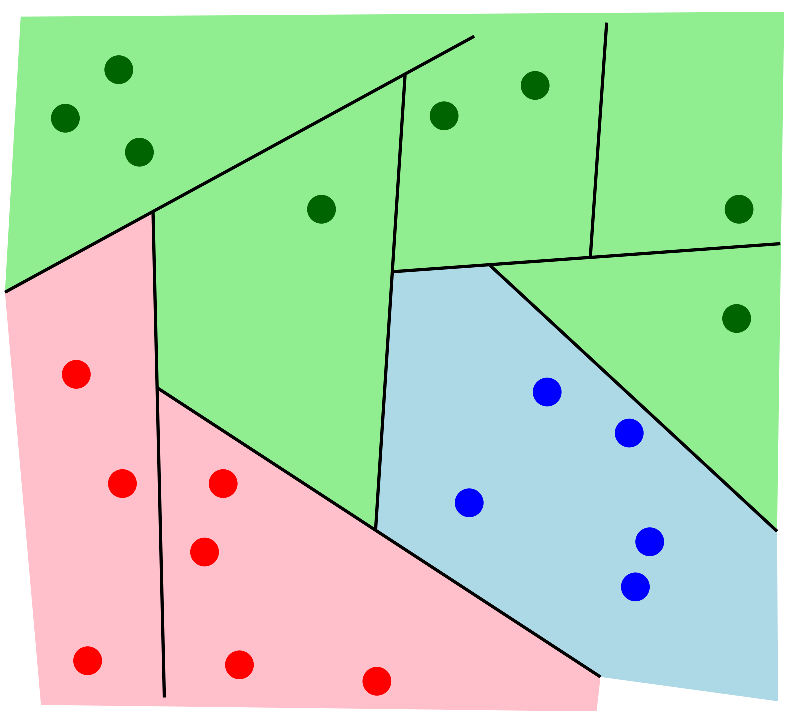


This induces a Voronoi partition with exponential growth in complexity

Can we use a random partition instead?

Random Partitions

Nearest-Neighbor-rule: Search among all labelled input elements for the one that minimizes a distance function (i.e., the *nearest neighbor*) and use this label as an estimator.



This induces a Voronoi partition with exponential growth in complexity

Can we use a random partition instead?

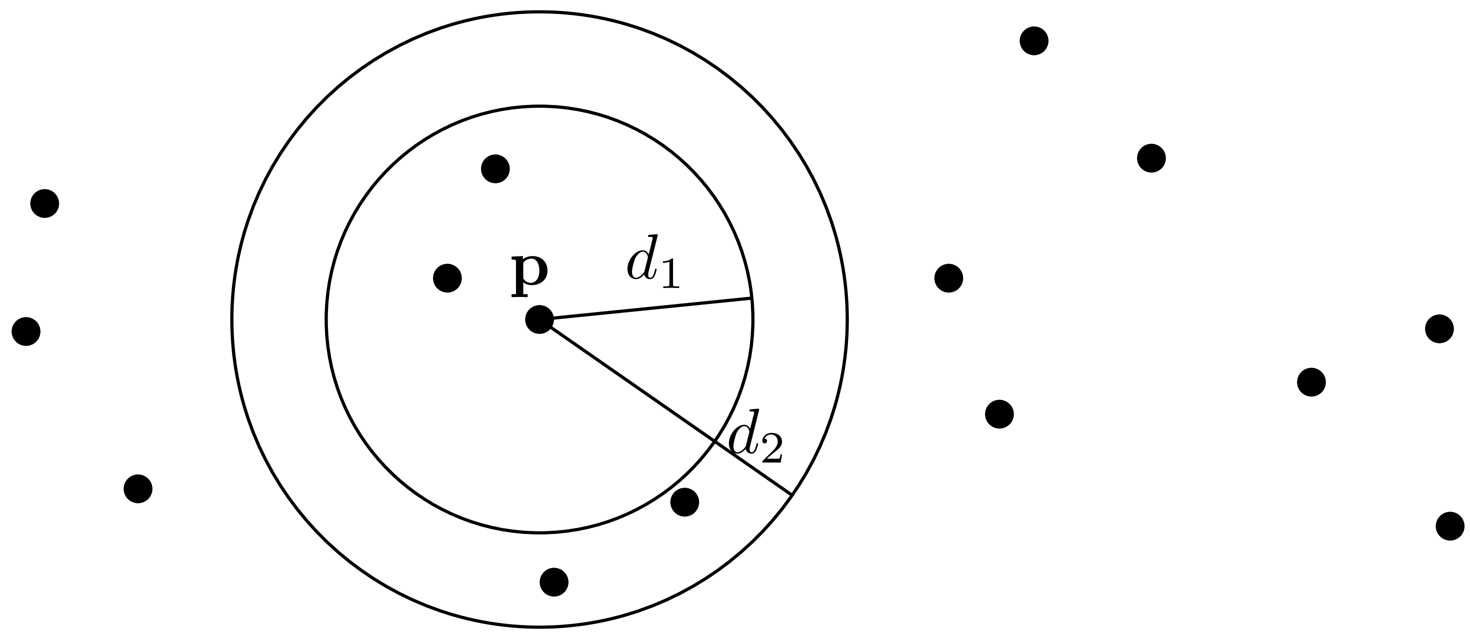
Locality-sensitive hashing (LSH)

Definition:
A family of hash functions H is called (d_1, d_2, p_1, p_2)-locality-sensitive if for $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$:

$$(a) \quad \text{if } \mathrm{d}(\mathbf{p}, \mathbf{q}) \leq d_1 \quad \text{then} \quad \Pr[h(\mathbf{p}) = h(\mathbf{q})] \geq p_1$$

$$(b) \quad \text{if } \mathrm{d}(\mathbf{p}, \mathbf{q}) \geq d_2 \quad \text{then} \quad \Pr[h(\mathbf{p}) = h(\mathbf{q})] \leq p_2$$

In general, we want $d_1 < d_2$ and $p_1 > p_2$



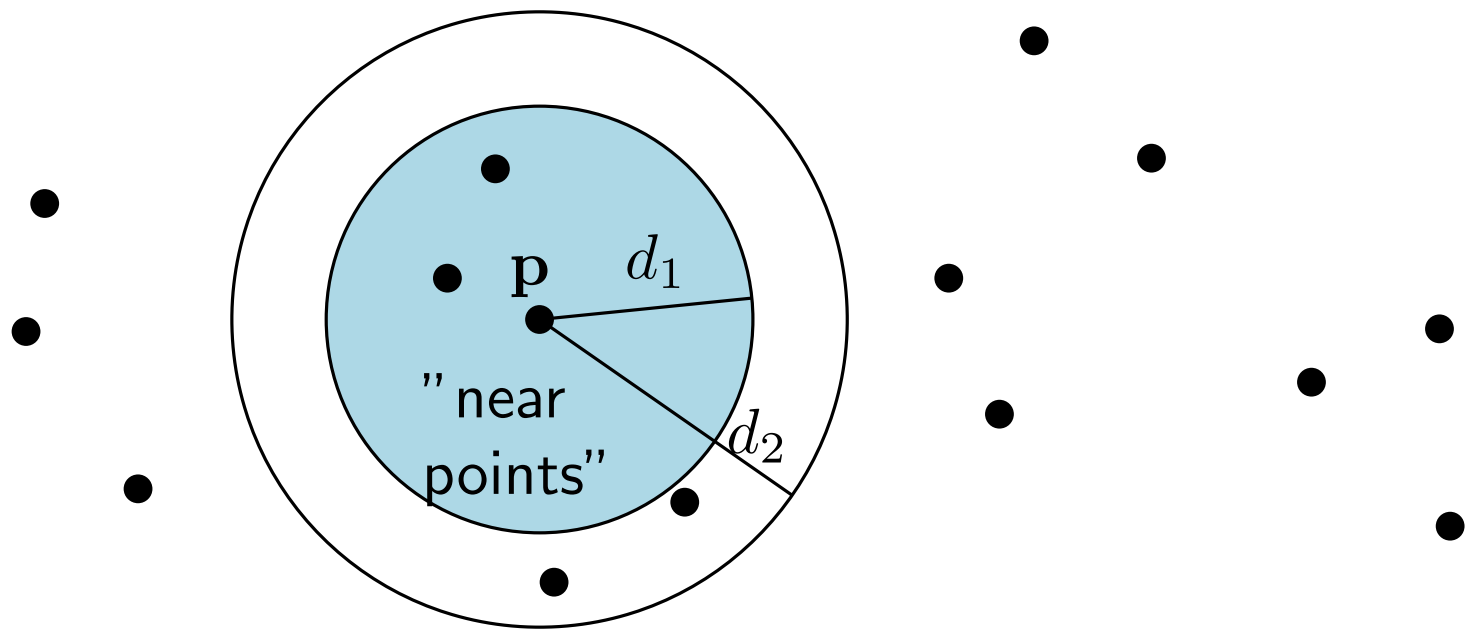
Locality-sensitive hashing (LSH)

Definition:
A family of hash functions H is called (d_1, d_2, p_1, p_2)-locality-sensitive if for $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$:

$$(a) \quad \text{if } \mathrm{d}(\mathbf{p}, \mathbf{q}) \leq d_1 \quad \text{then} \quad \Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] \geq p_1$$

$$(b) \quad \text{if } \mathrm{d}(\mathbf{p}, \mathbf{q}) \geq d_2 \quad \text{then} \quad \Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] \leq p_2$$

In general, we want $d_1 < d_2$ and $p_1 > p_2$



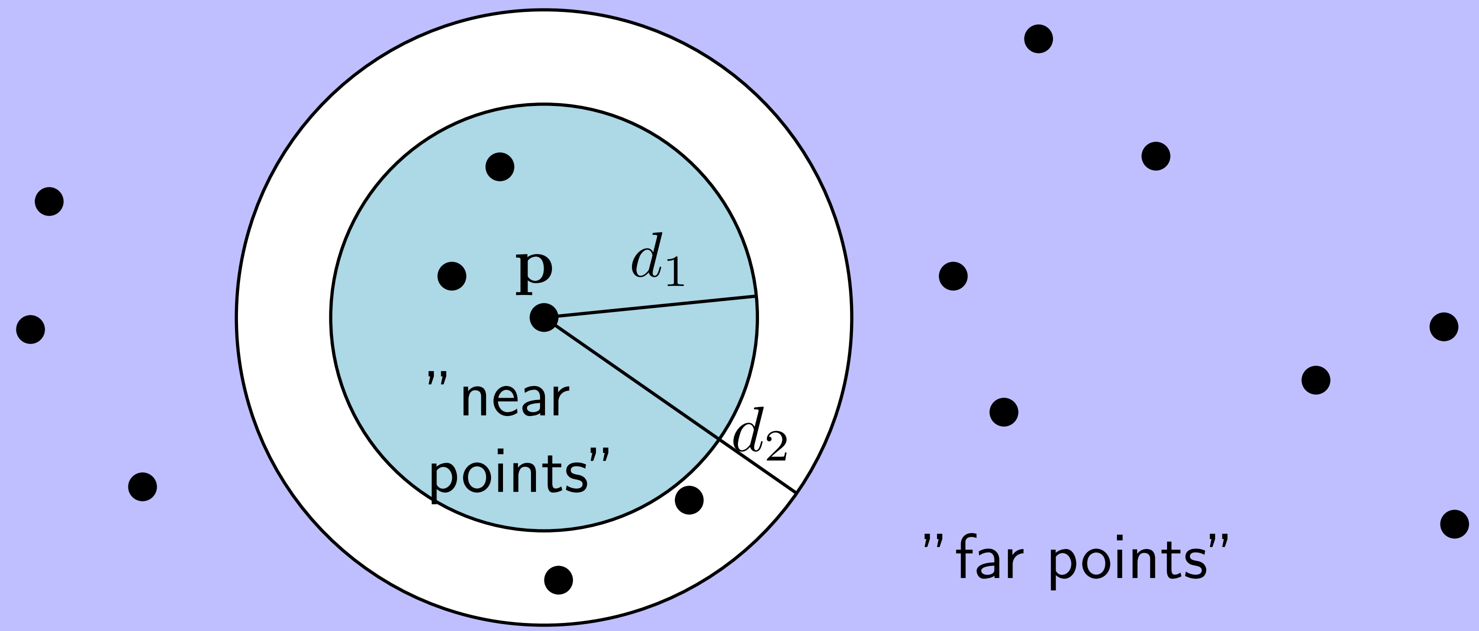
Locality-sensitive hashing (LSH)

Definition:
A family of hash functions H is called (d_1, d_2, p_1, p_2)-locality-sensitive if for $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$:

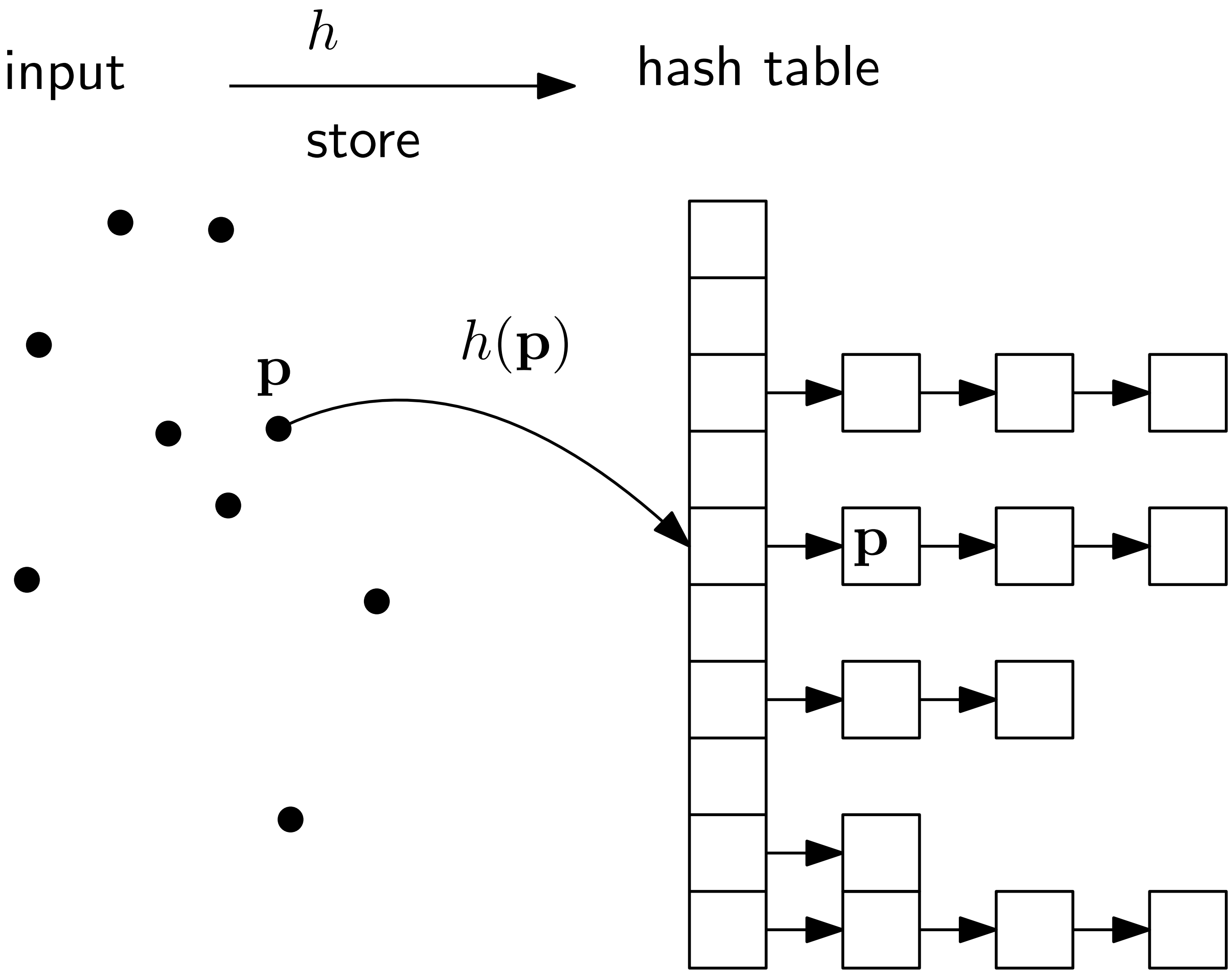
$$(a) \quad \text{if } \mathrm{d}(\mathbf{p}, \mathbf{q}) \leq d_1 \quad \text{then} \quad \Pr[h(\mathbf{p}) = h(\mathbf{q})] \geq p_1$$

$$(b) \quad \text{if } \mathrm{d}(\mathbf{p}, \mathbf{q}) \geq d_2 \quad \text{then} \quad \Pr[h(\mathbf{p}) = h(\mathbf{q})] \leq p_2$$

In general, we want $d_1 < d_2$ and $p_1 > p_2$

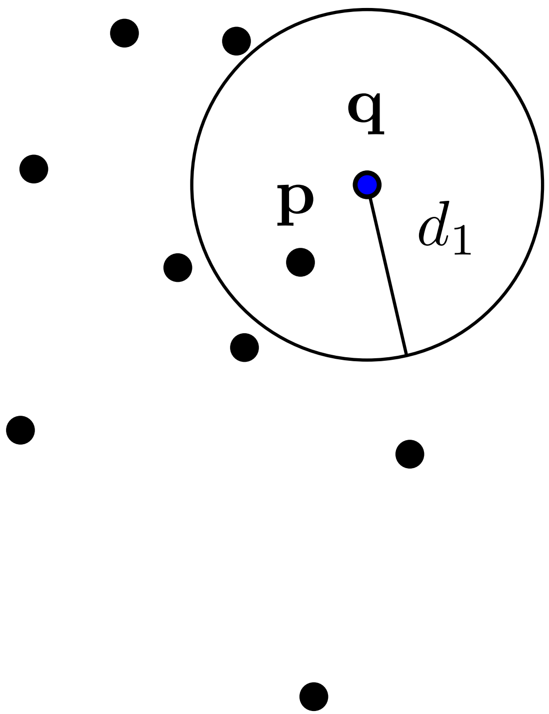


Locality-sensitive hashing (LSH)

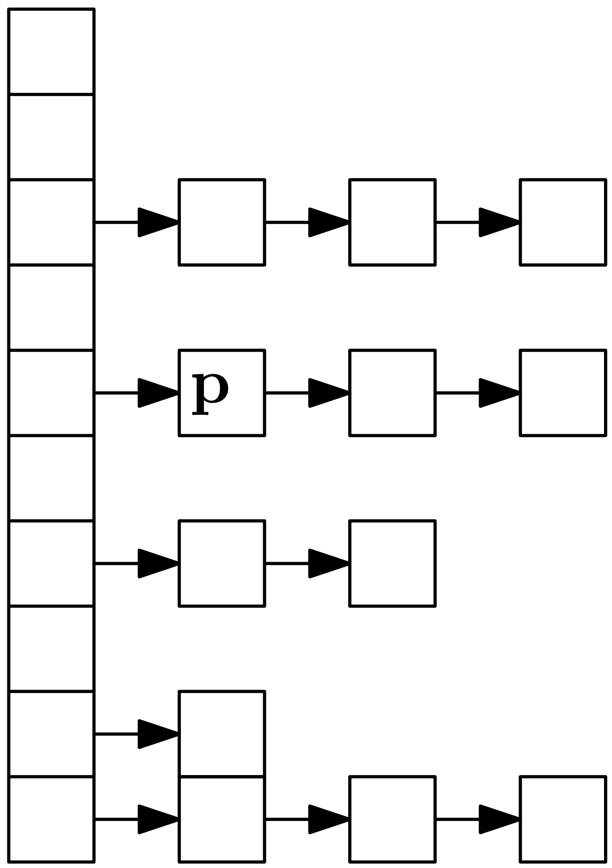


Locality-sensitive hashing (LSH)

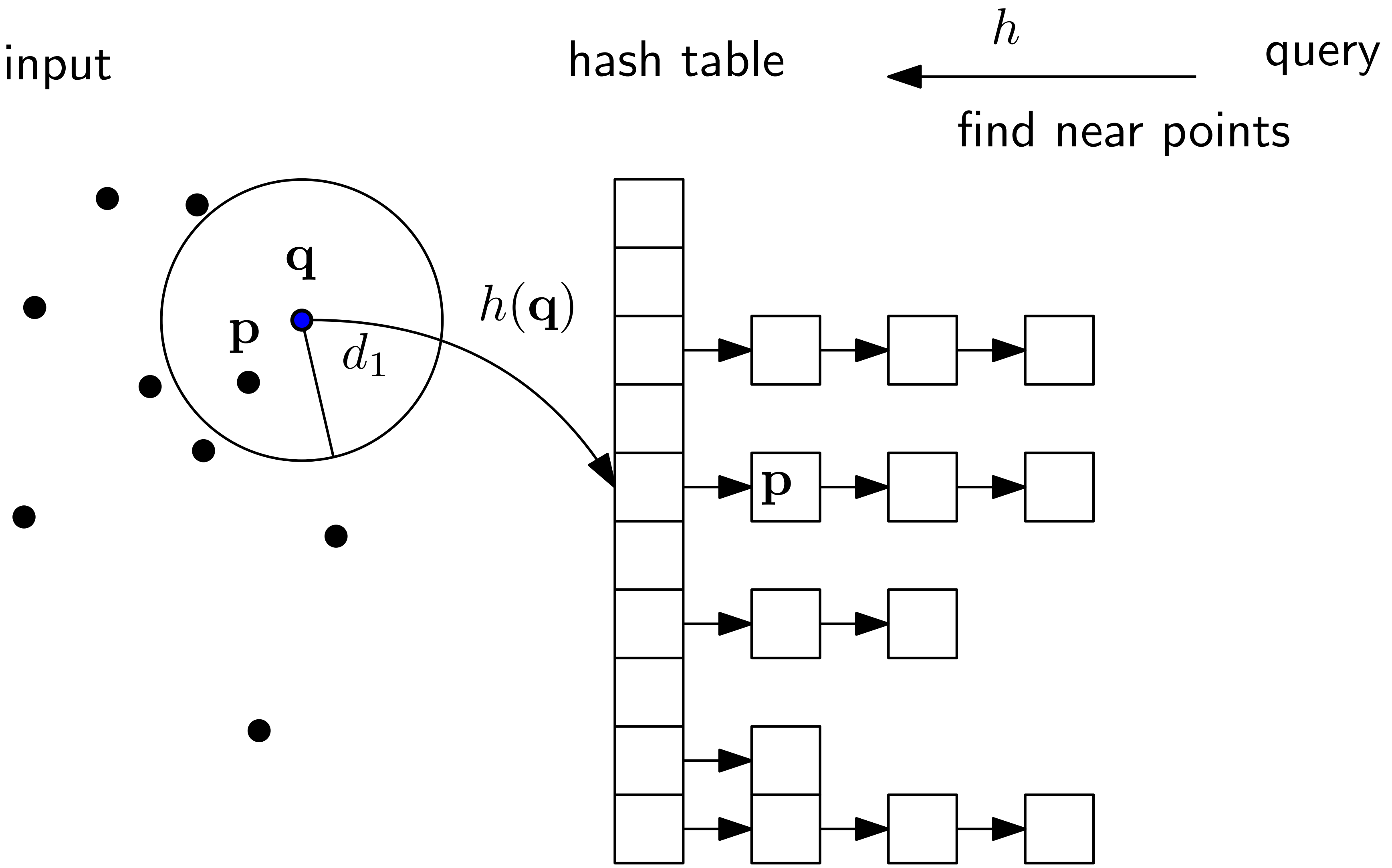
input



hash table



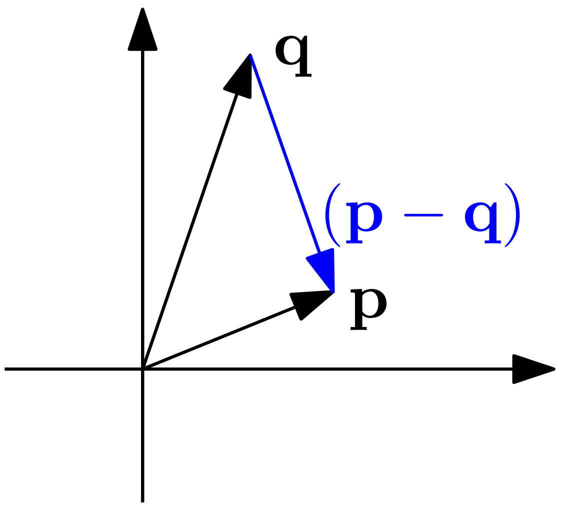
Locality-sensitive hashing (LSH)



Commonly used distance functions

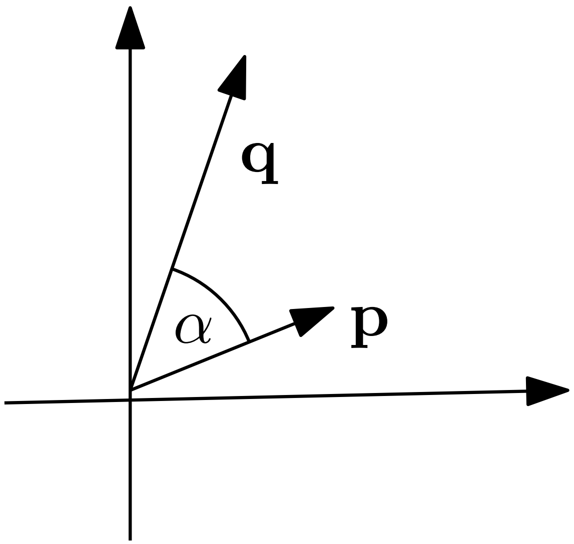
Euclidean distance:

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) := \|\mathbf{p} - \mathbf{q}\|$$

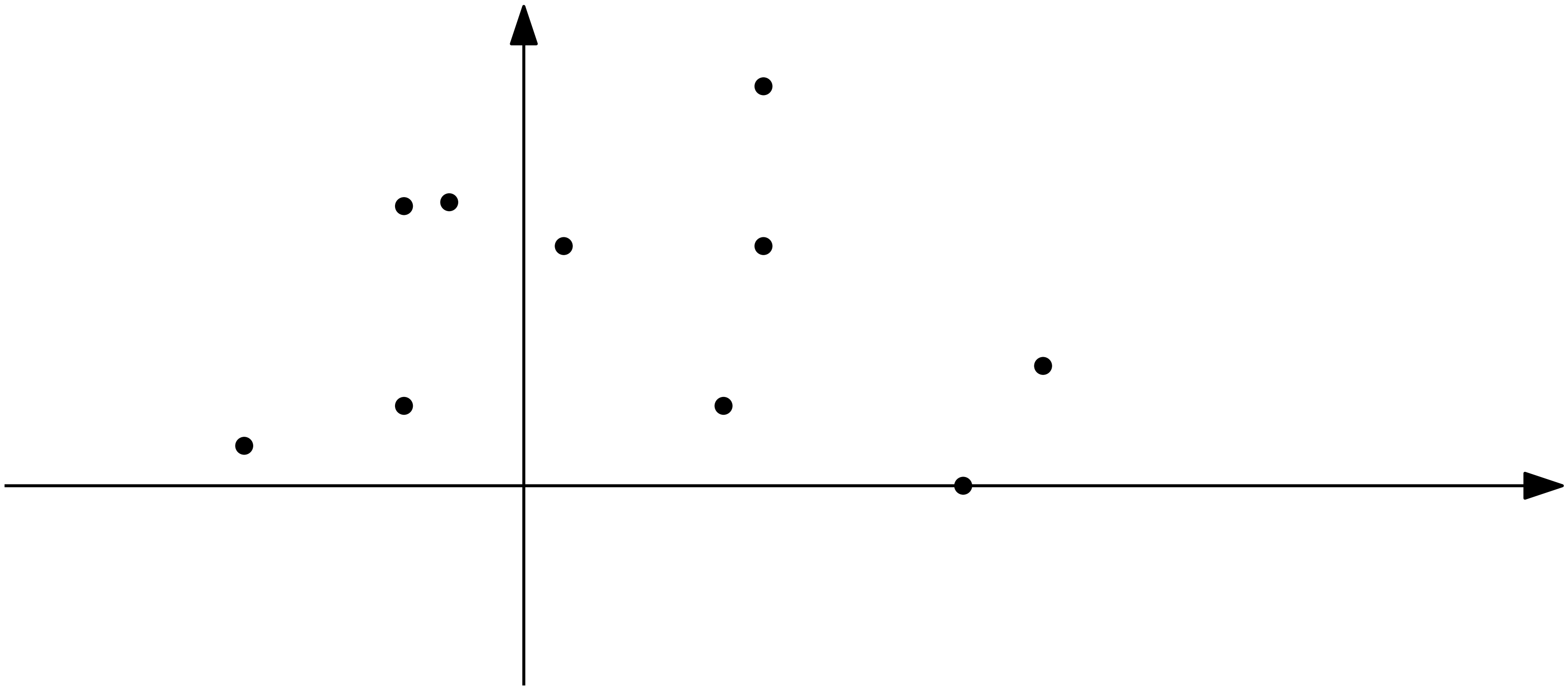


Arccos distance:

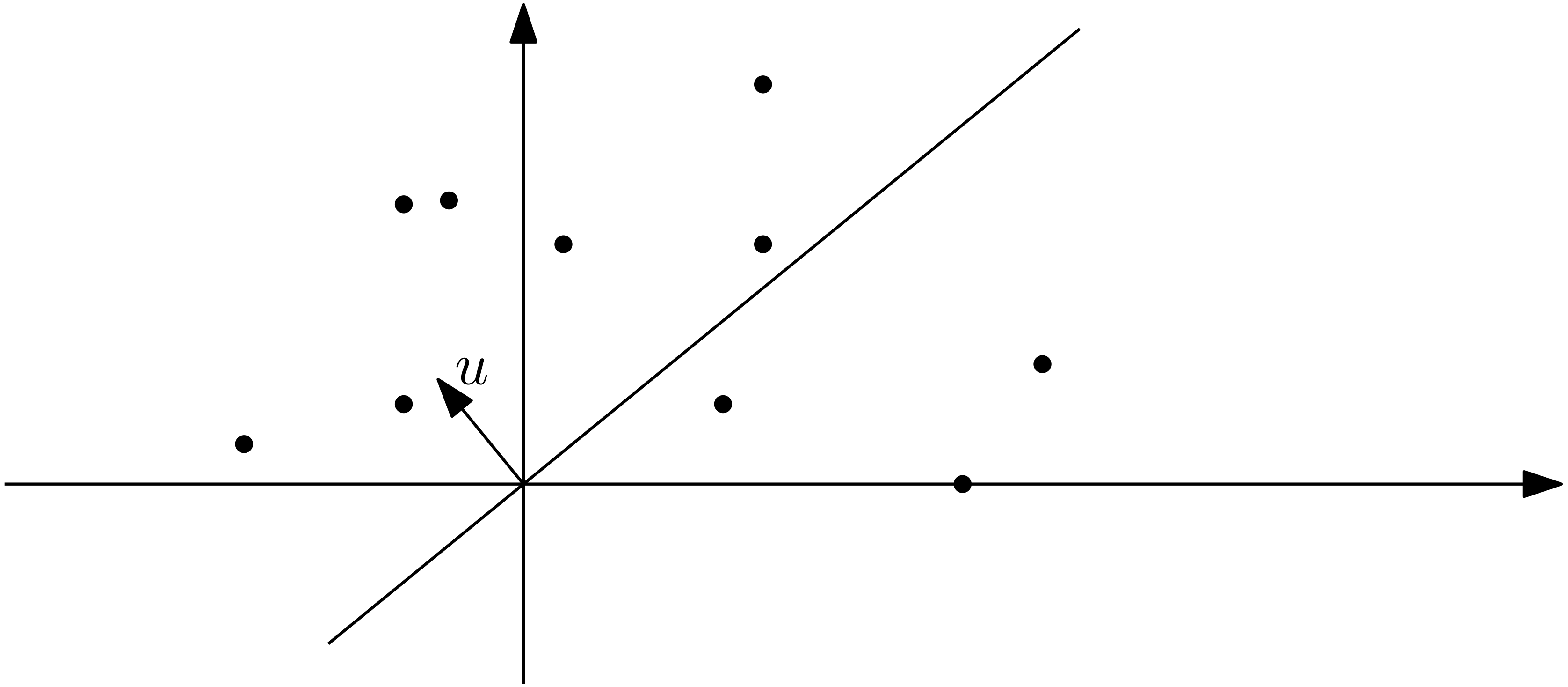
$$\mathrm{d}(\mathbf{p}, \mathbf{q}) := \arccos\left(\frac{\langle \mathbf{p}, \mathbf{q} \rangle}{\|\mathbf{p}\| \; \|\mathbf{q}\|}\right)$$



Locality sensitive hashing: Arccos distance

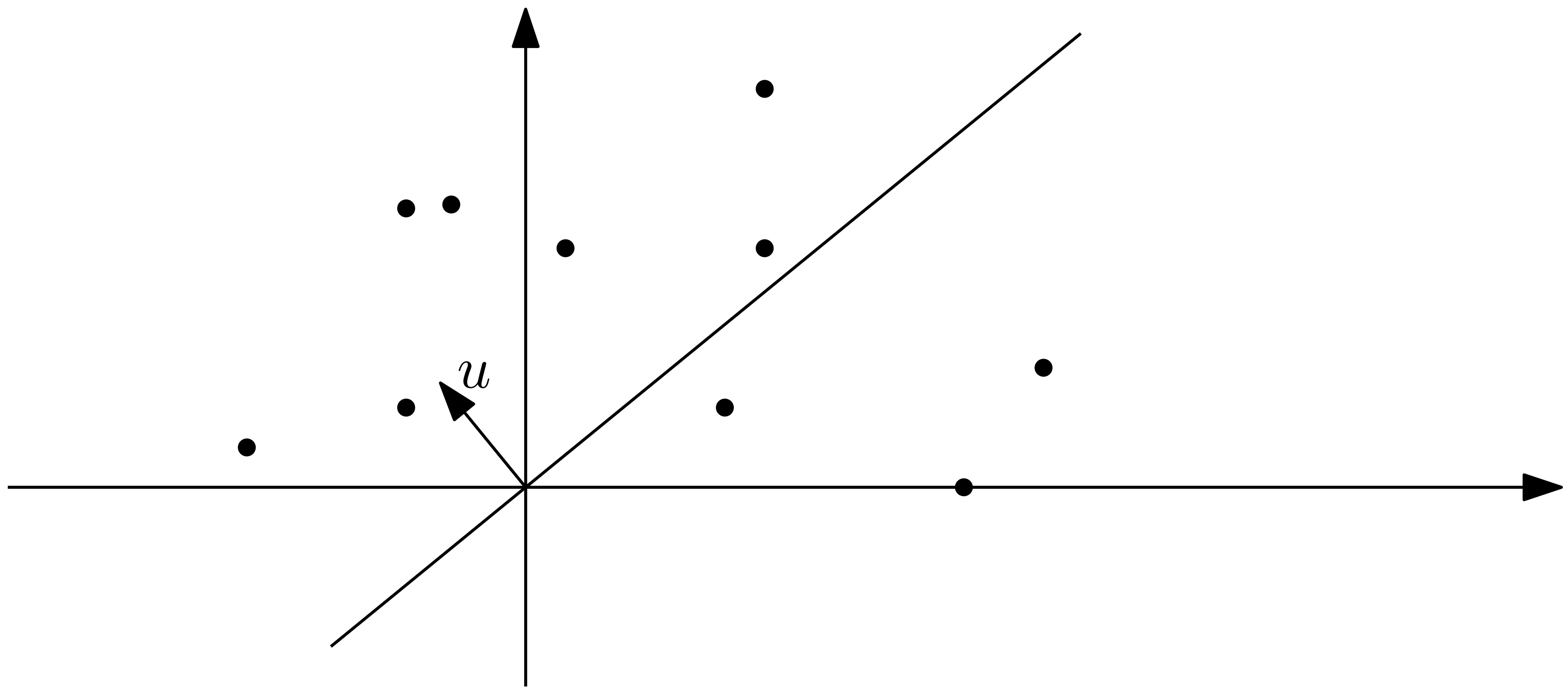


Locality sensitive hashing: Arccos distance



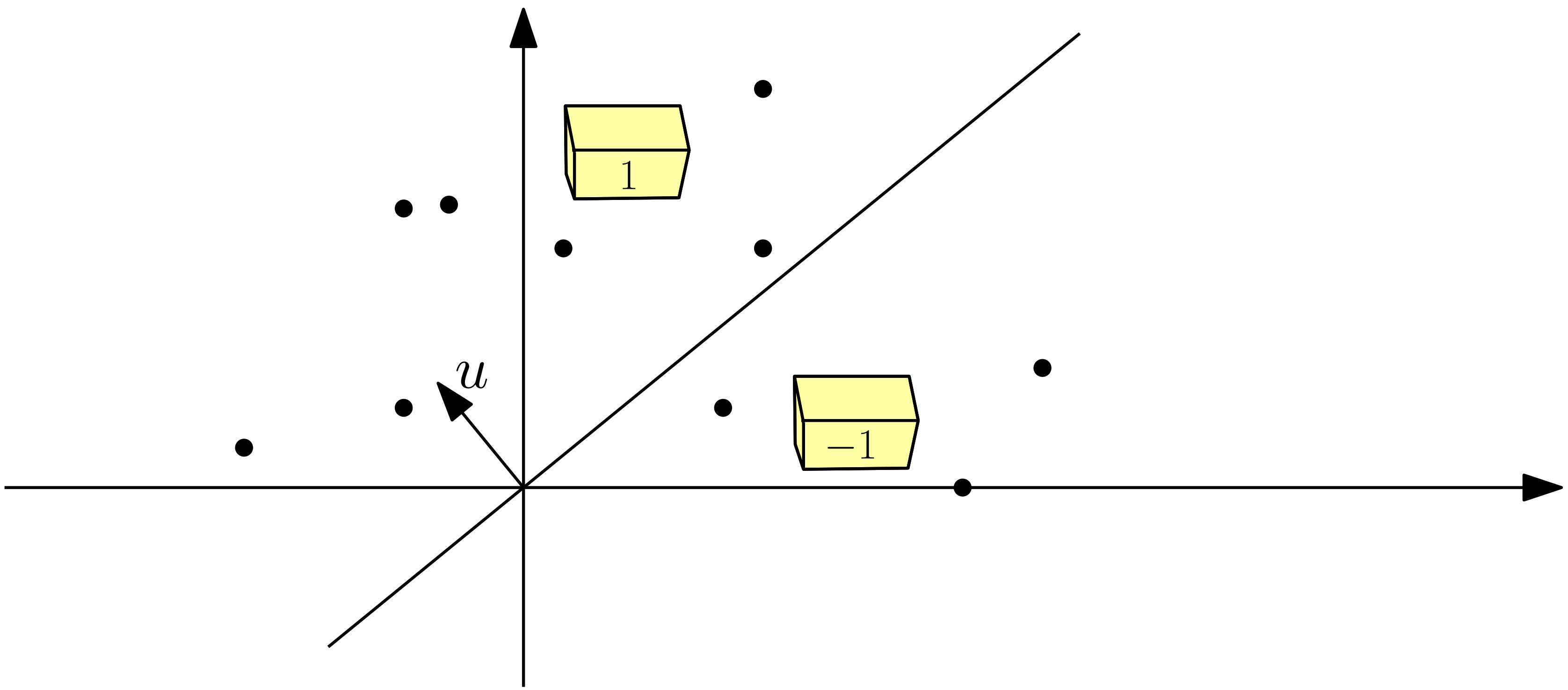
- randomly sample a hyperplane by choosing a normal vector u

Locality sensitive hashing: Arccos distance



- randomly sample a hyperplane by choosing a normal vector u
- to hash $\mathbf{p}$ compute the sign of $\langle \mathbf{p}, u \rangle$ to find the side of the hyperplane thay $\mathbf{p}$ lies on

Locality sensitive hashing: Arccos distance



- randomly sample a hyperplane by choosing a normal vector u
- to hash $\mathbf{p}$ compute the sign of $\langle \mathbf{p}, u \rangle$ to find the side of the hyperplane thay $\mathbf{p}$ lies on
- $h(\mathbf{p}) = \mathrm{sign}(\langle \mathbf{p}, u \rangle)$

Locality sensitive hashing: Arccos distance

Claim:

For any $\mathbf{p}, \mathbf{q}$, it holds that

$$\Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] = \frac{2\pi - \alpha}{2\pi}$$

where $\alpha = \arccos\left(\frac{\langle \mathbf{p}, \mathbf{q} \rangle}{\|\mathbf{p}\| \ \|\mathbf{q}\|}\right)$.

Locality sensitive hashing: Arccos distance

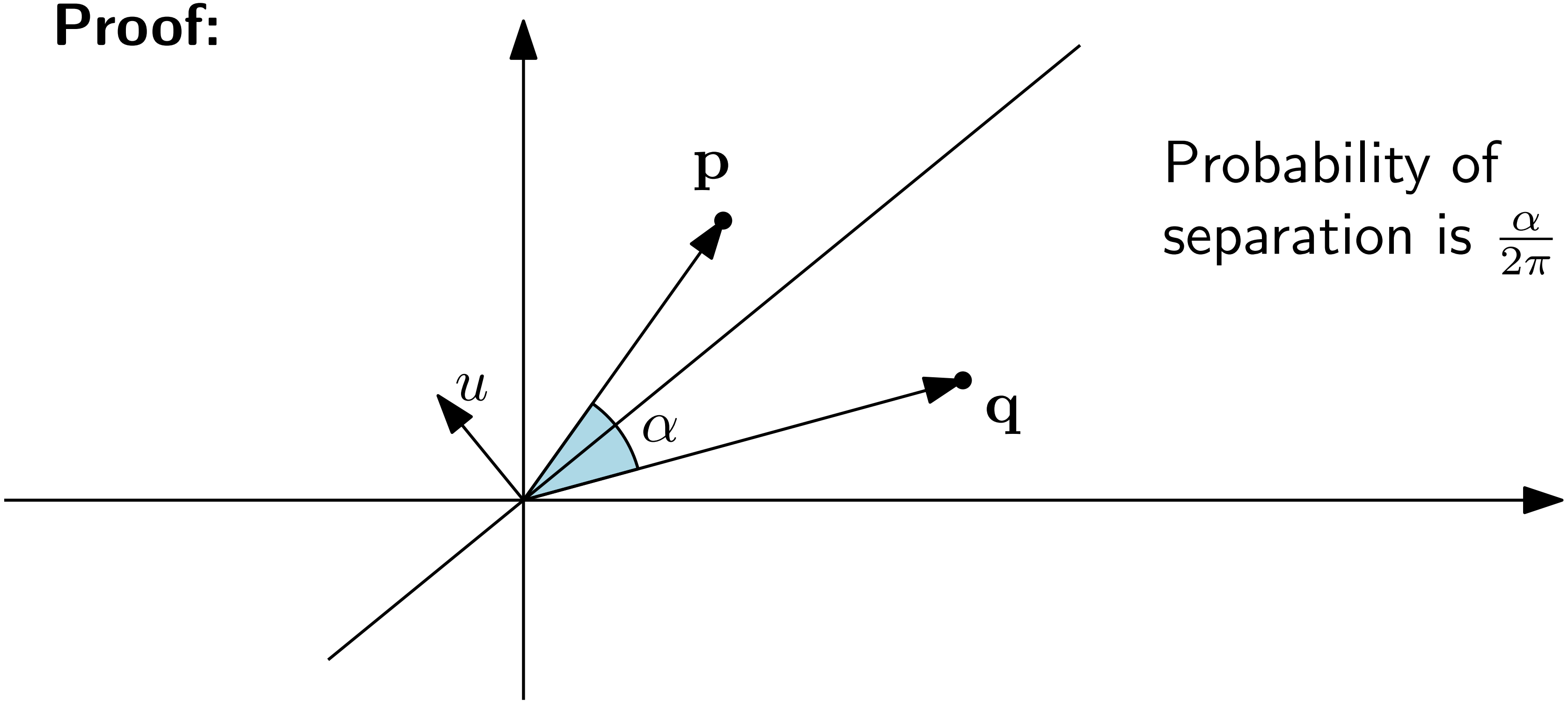
Claim:

For any $\mathbf{p}, \mathbf{q}$, it holds that

$$\Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] = \frac{2\pi - \alpha}{2\pi}$$

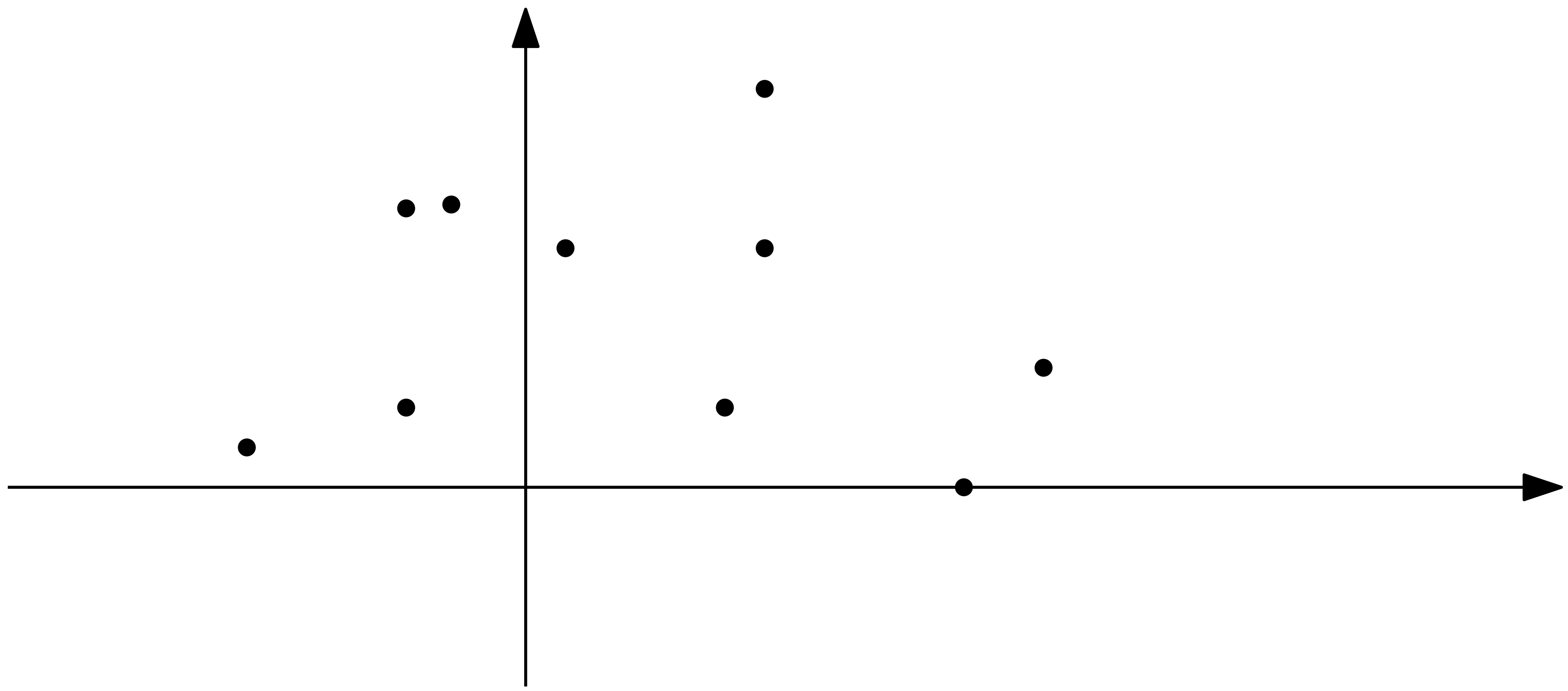
where $\alpha = \arccos\left(\frac{\langle \mathbf{p}, \mathbf{q} \rangle}{\|\mathbf{p}\| \ \|\mathbf{q}\|}\right)$.

Proof:

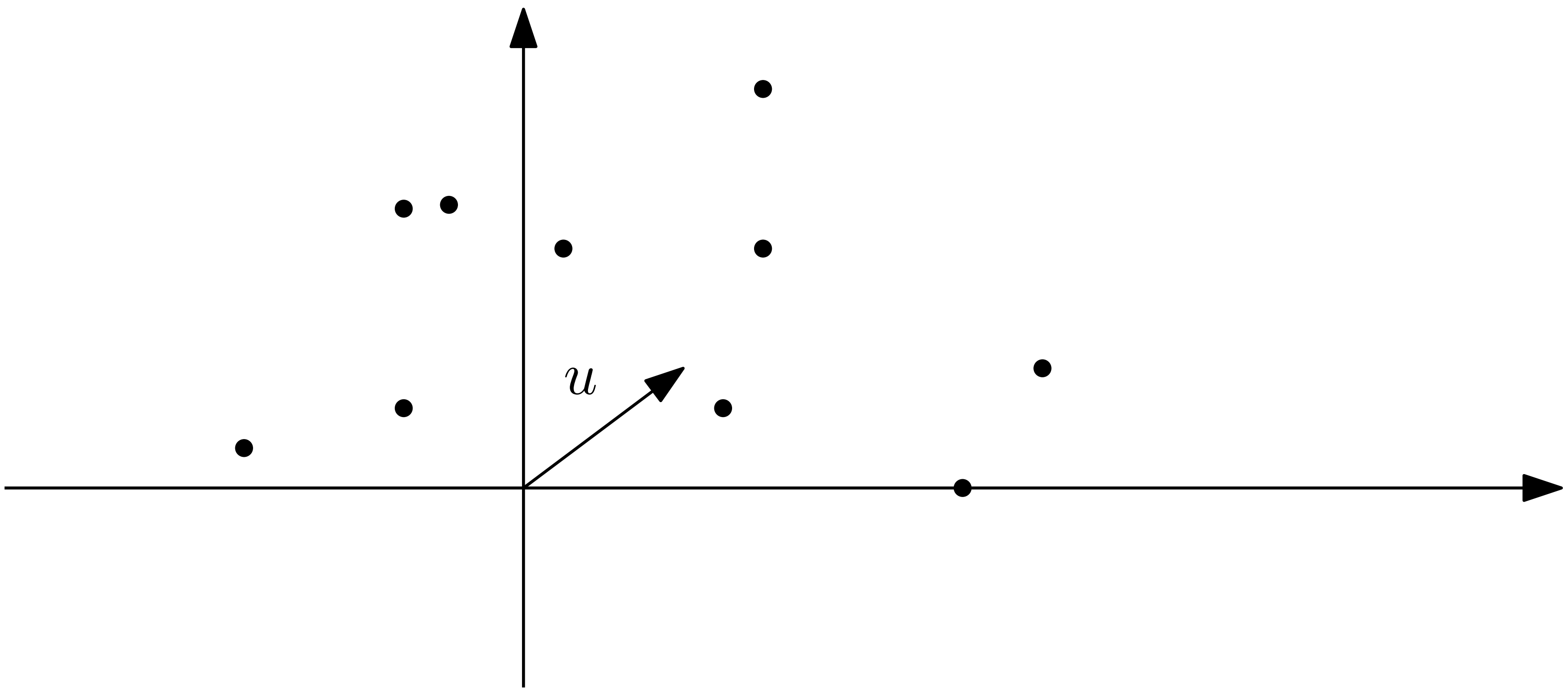


Probability of separation is $\frac{\alpha}{2\pi}$

Locality-sensitive hashing: Euclidean distance

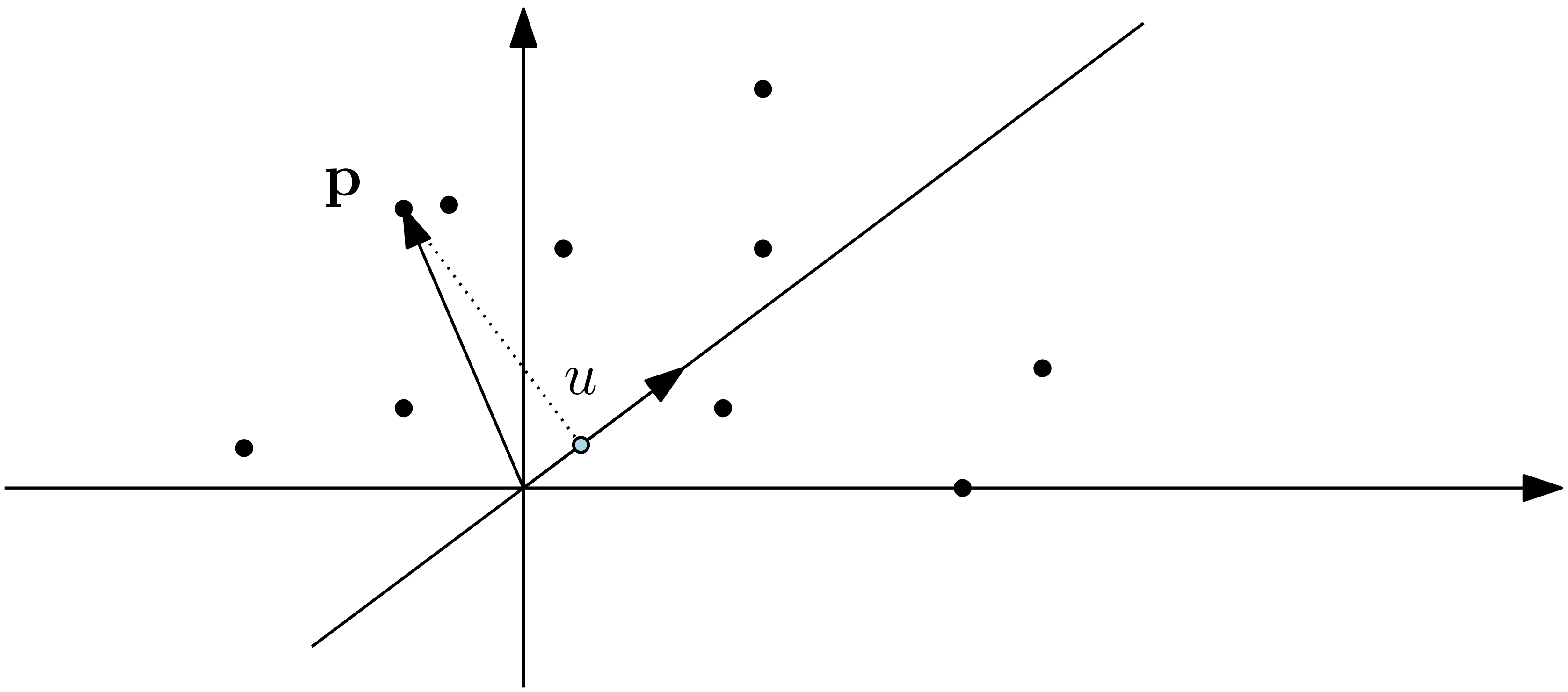


Locality-sensitive hashing: Euclidean distance



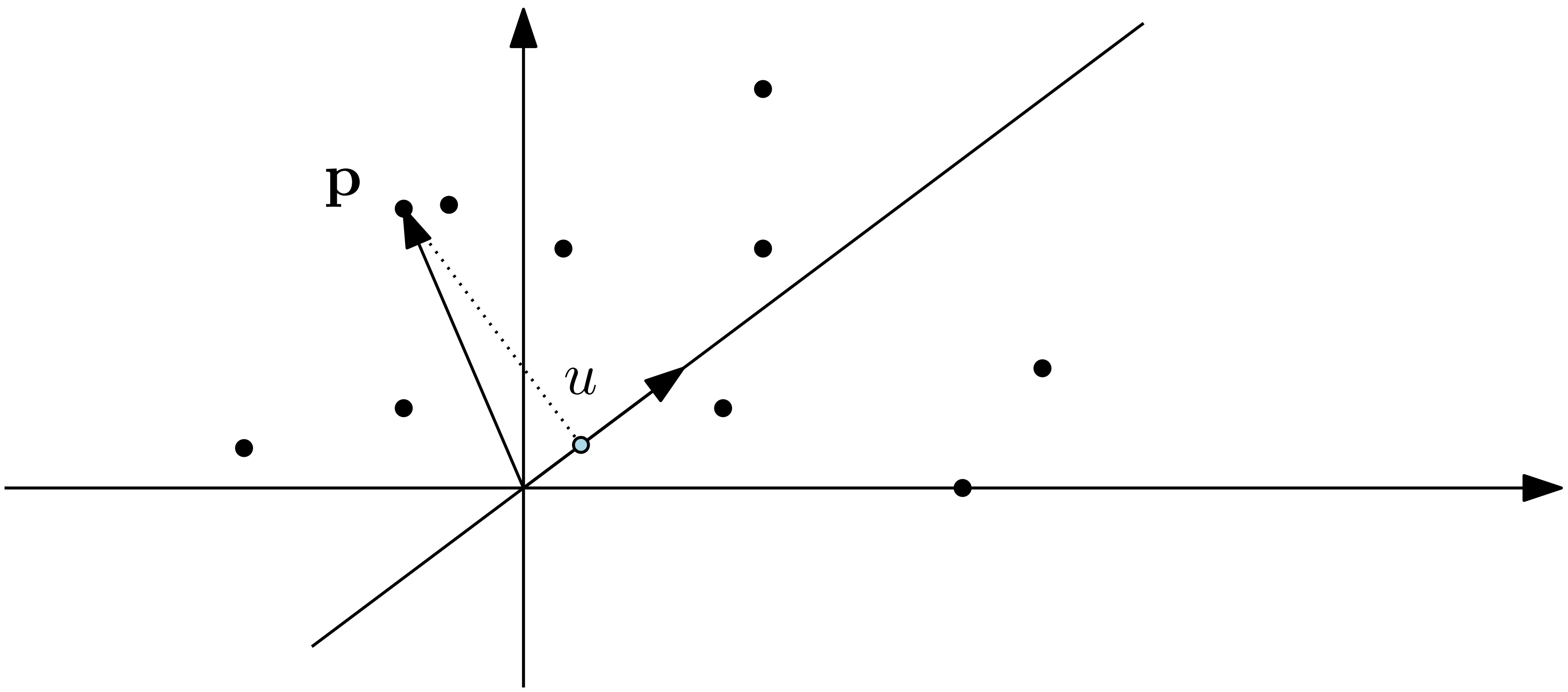
- randomly sample a unit vector u

Locality-sensitive hashing: Euclidean distance



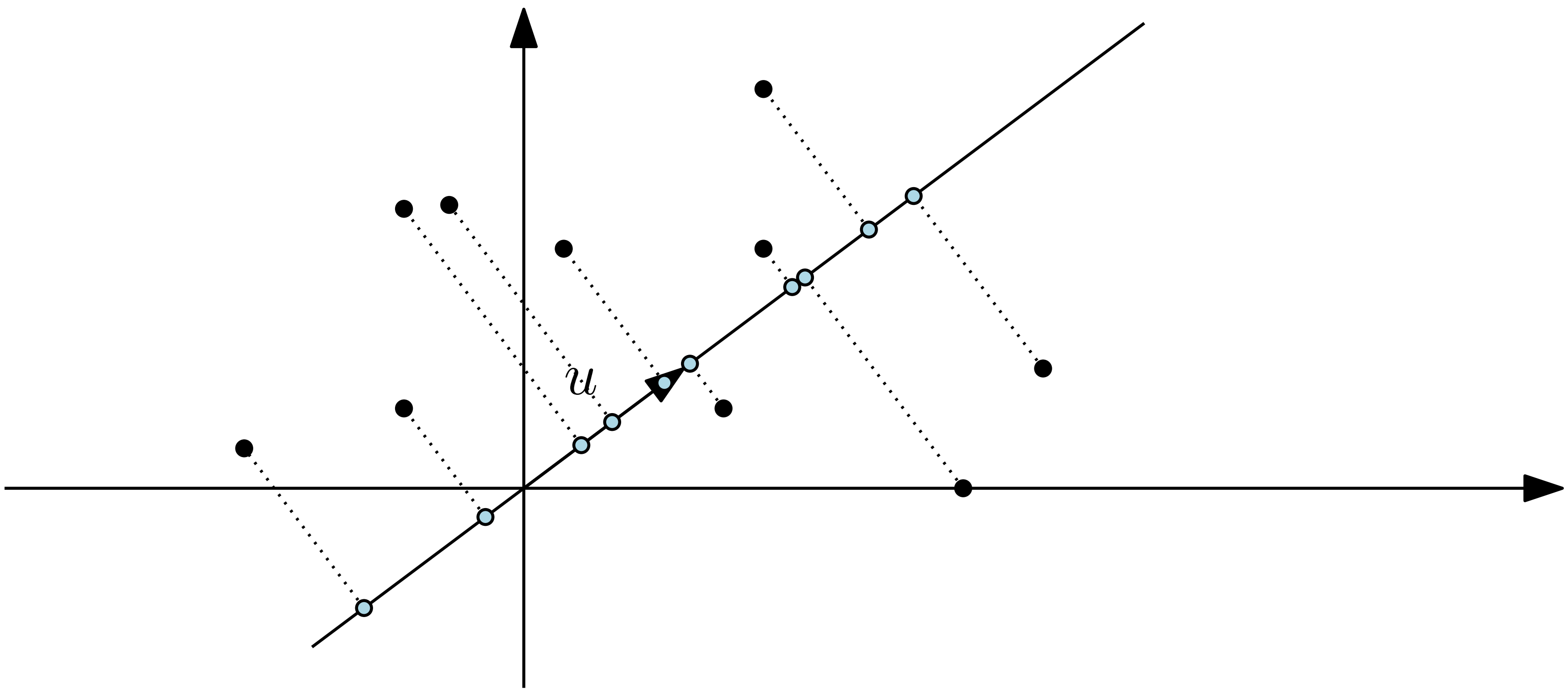
- randomly sample a unit vector u
- project onto u by computing the dot product $\langle \mathbf{p}, u \rangle$

Locality-sensitive hashing: Euclidean distance



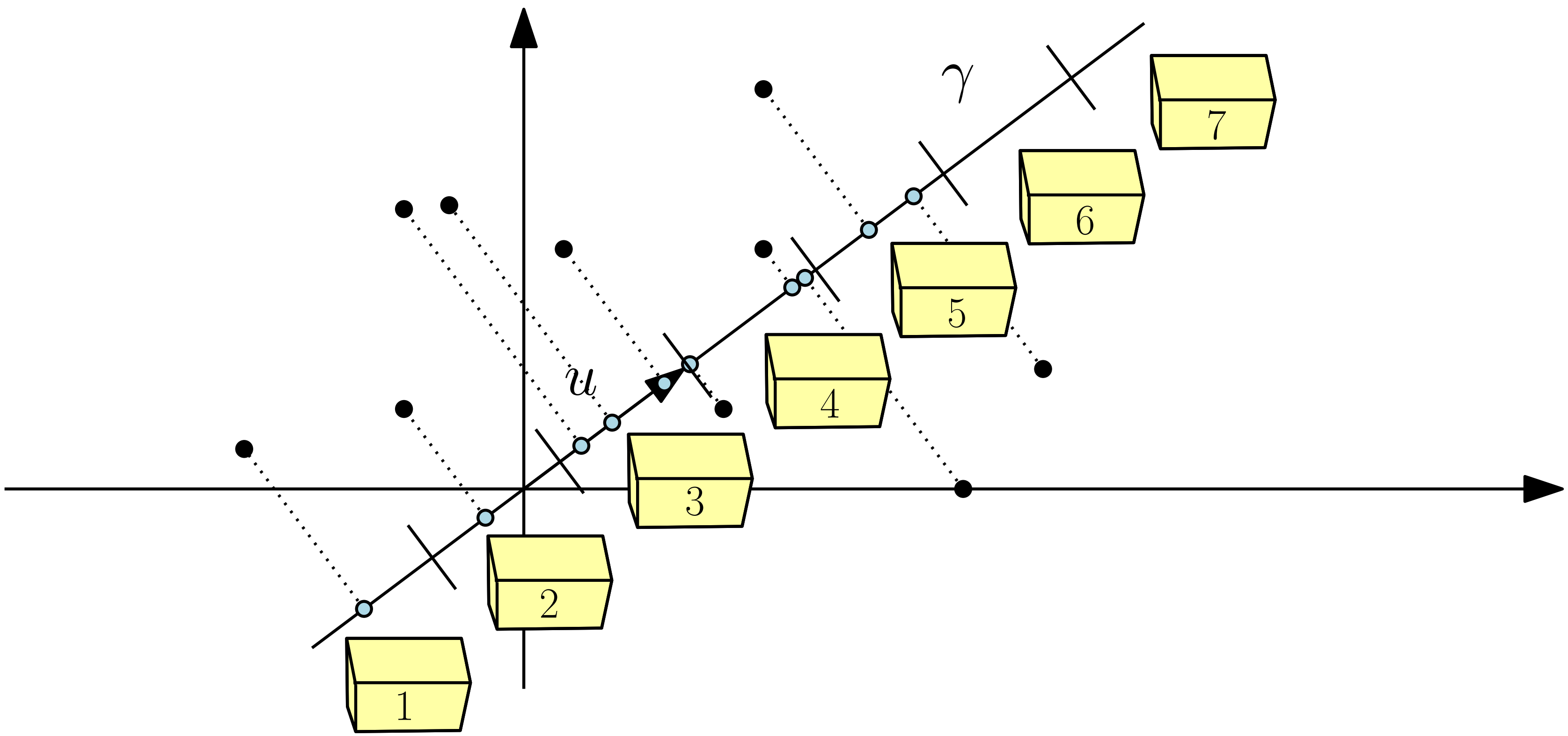
- randomly sample a unit vector u
- project onto u by computing the dot product $\langle \mathbf{p}, u \rangle$

Locality-sensitive hashing: Euclidean distance



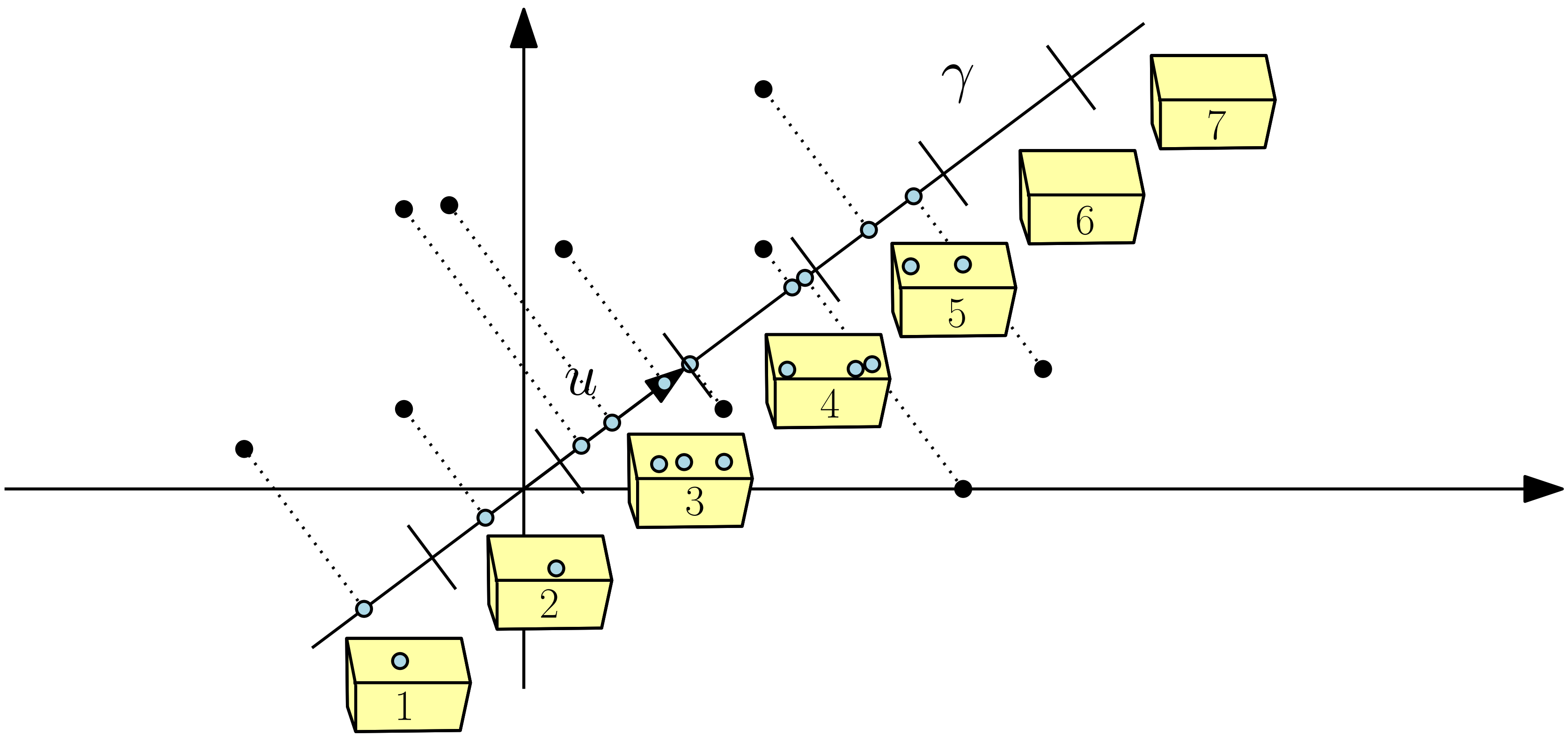
- randomly sample a unit vector u
- project onto u by computing the dot product $\langle \mathbf{p}, u \rangle$

Locality-sensitive hashing: Euclidean distance



- randomly sample a unit vector u
- project onto u by computing the dot product $\langle \mathbf{p}, u \rangle$
- create bins of size γ in $\mathbb{R}^1$ with random shift in $[0, \gamma)$

Locality-sensitive hashing: Euclidean distance



- randomly sample a unit vector u
- project onto u by computing the dot product $\langle \mathbf{p}, u \rangle$
- create bins of size γ in $\mathbb{R}^1$ with random shift in $[0, \gamma)$
- $h(\mathbf{p})$ = index of the bin that $\mathbf{p}$ is projected into

Locality-sensitive hashing: Euclidean distance

Claim:

This hashing scheme is $(\frac{\gamma}{2}, 2\gamma, \frac{1}{2}, \frac{1}{3})$-locality-sensitive, that is:

(a) if $\|\mathbf{p} - \mathbf{q}\| \leq \frac{\gamma}{2}$ then $\Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] \geq \frac{1}{2}$, and

(b) if $\|\mathbf{p} - \mathbf{q}\| \geq 2\gamma$ then $\Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] \leq \frac{1}{3}$

Locality-sensitive hashing: Euclidean distance

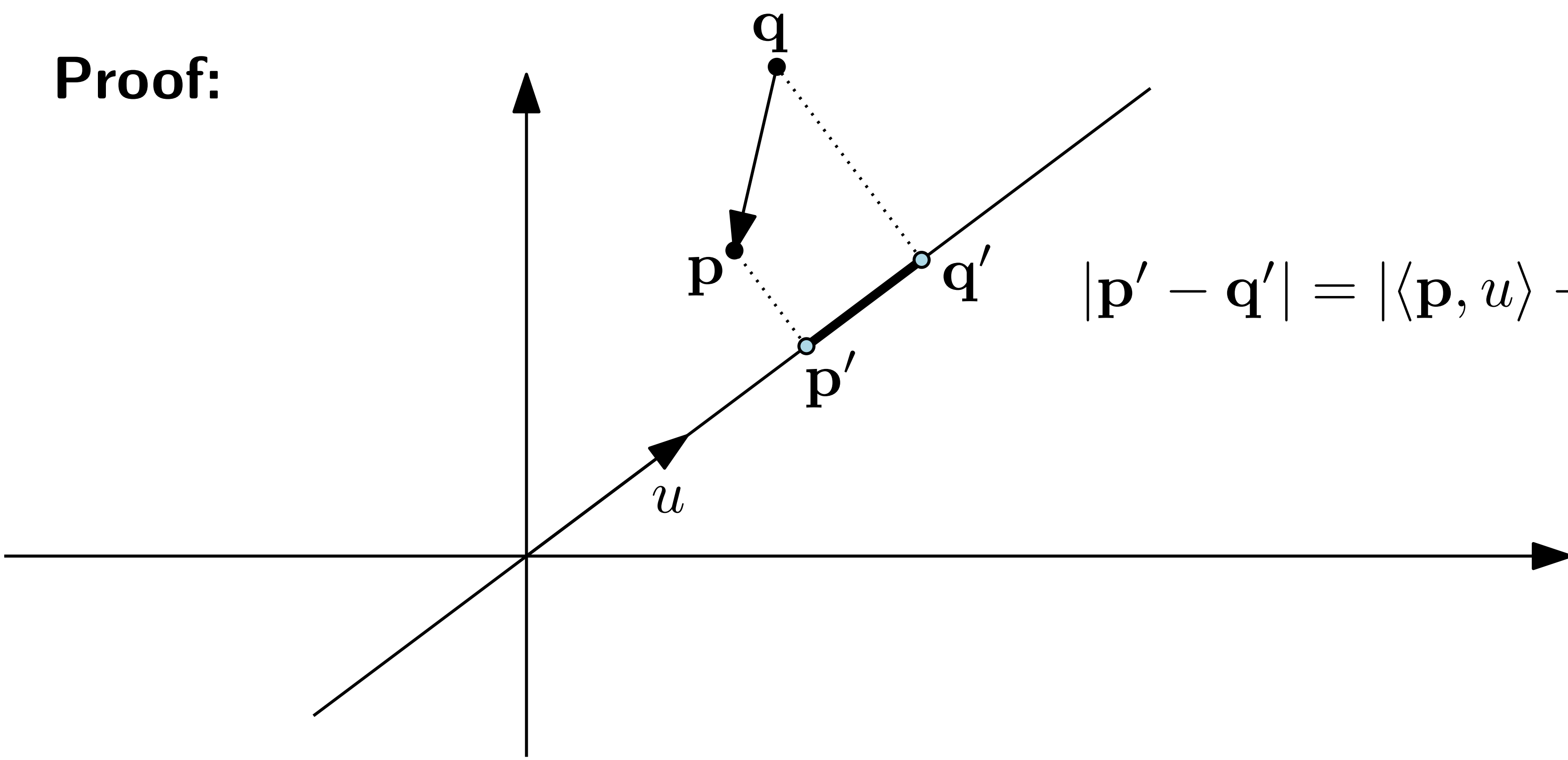
Claim:

This hashing scheme is $(\frac{\gamma}{2}, 2\gamma, \frac{1}{2}, \frac{1}{3})$-locality-sensitive, that is:

(a) if $\|\mathbf{p}-\mathbf{q}\| \leq \frac{\gamma}{2}$ then $\Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] \geq \frac{1}{2}$, and

(b) if $\|\mathbf{p}-\mathbf{q}\| \geq 2\gamma$ then $\Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] \leq \frac{1}{3}$

Proof:



$$|\mathbf{p}'-\mathbf{q}'| = |\langle \mathbf{p}, u\rangle - \langle \mathbf{q}, u\rangle|$$

Locality-sensitive hashing: Euclidean distance

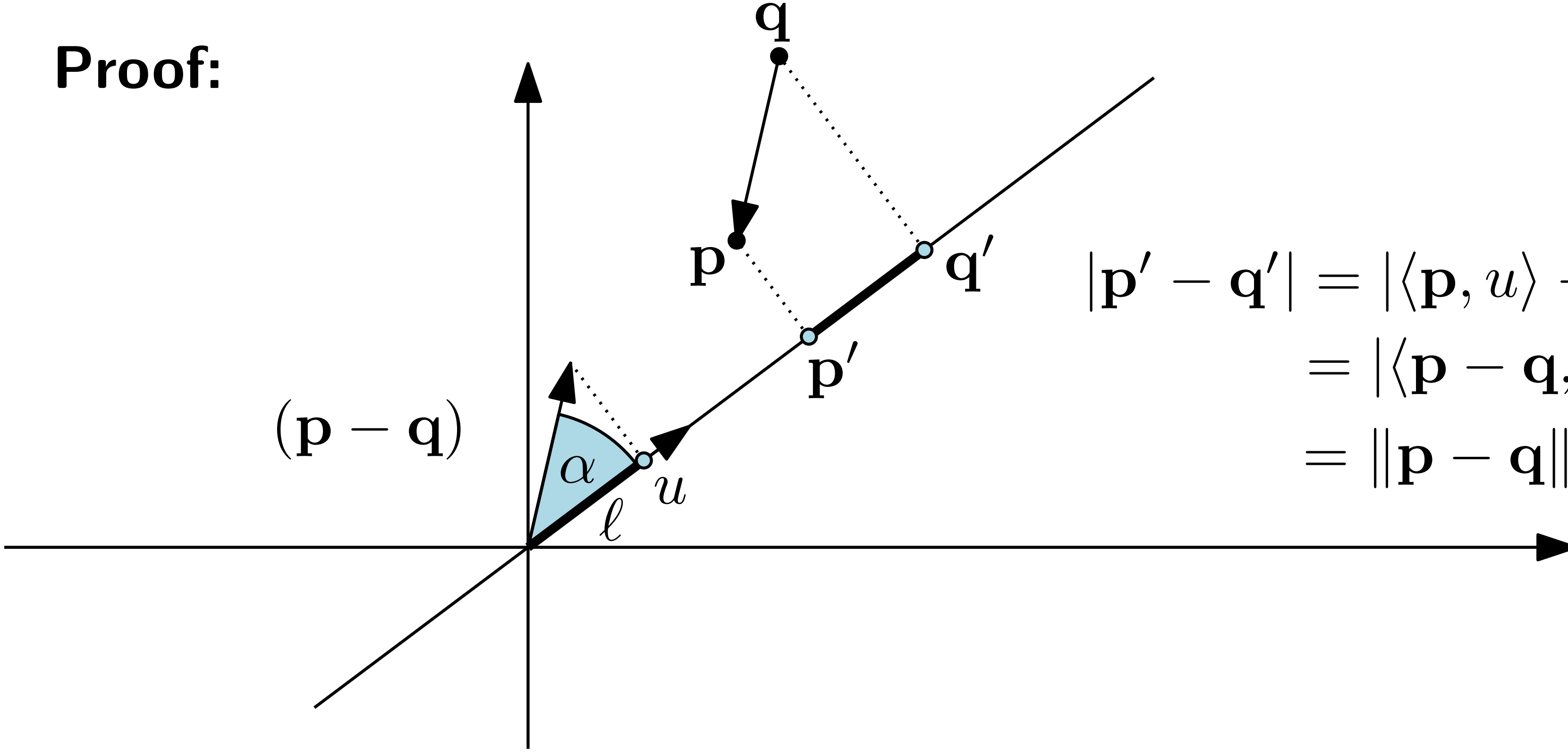
Claim:

This hashing scheme is $(\frac{\gamma}{2}, 2\gamma, \frac{1}{2}, \frac{1}{3})$-locality-sensitive, that is:

(a) if $\|\mathbf{p} - \mathbf{q}\| \leq \frac{\gamma}{2}$ then $\Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] \geq \frac{1}{2}$, and

(b) if $\|\mathbf{p} - \mathbf{q}\| \geq 2\gamma$ then $\Pr\left[h(\mathbf{p}) = h(\mathbf{q})\right] \leq \frac{1}{3}$

Proof:



$$\begin{aligned} |\mathbf{p}' - \mathbf{q}'| &= |\langle \mathbf{p}, u\rangle - \langle \mathbf{q}, u\rangle| \\ &= |\langle \mathbf{p} - \mathbf{q}, u\rangle| \\ &= \|\mathbf{p} - \mathbf{q}\| \cdot |\cos(\alpha)| \end{aligned}$$

Locality-sensitive hashing: Euclidean distance

(Continued)

Proof of (a)   "near points have higher collision probability"

The probability of separation is

$$\Pr\left[h(\mathbf{p}) \neq h(\mathbf{q})\right] = \frac{|\mathbf{p}' - \mathbf{q}'|}{\gamma}$$

Locality-sensitive hashing: Euclidean distance

(Continued)

Proof of (a) "near points have higher collision probability"

The probability of separation is

$$\Pr\left[h(\mathbf{p}) \neq h(\mathbf{q})\right] = \frac{|\mathbf{p}' - \mathbf{q}'|}{\gamma}$$
$$= \frac{\|\mathbf{p} - \mathbf{q}\| |\cos\alpha|}{\gamma}$$

Locality-sensitive hashing: Euclidean distance

(Continued)

Proof of (a) "near points have higher collision probability"

The probability of separation is

$$\Pr\left[h(\mathbf{p}) \neq h(\mathbf{q})\right] = \frac{|\mathbf{p}' - \mathbf{q}'|}{\gamma}$$
$$= \frac{\|\mathbf{p} - \mathbf{q}\| |\cos\alpha|}{\gamma}$$
$$\leq \frac{\|\mathbf{p} - \mathbf{q}\|}{\gamma} \qquad \text{(since } |\cos\alpha| \leq 1\text{)}$$

Locality-sensitive hashing: Euclidean distance

(Continued)

Proof of (a) "near points have higher collision probability"

The probability of separation is

$$\Pr\left[h(\mathbf{p}) \neq h(\mathbf{q})\right] = \frac{|\mathbf{p}' - \mathbf{q}'|}{\gamma}$$

$$= \frac{\|\mathbf{p} - \mathbf{q}\| |\cos \alpha|}{\gamma}$$

$$\leq \frac{\|\mathbf{p} - \mathbf{q}\|}{\gamma} \qquad \text{(since } |\cos\alpha| \leq 1\text{)}$$

$$\leq \frac{\gamma/2}{\gamma} = \frac{1}{2}$$

Locality-sensitive hashing: Euclidean distance

(Continued)

Proof of (b) "far points have lower collision probability"

If $h(\mathbf{p}) = h(\mathbf{q})$ then

$$\gamma \geq |\mathbf{p}' - \mathbf{q}'|$$

Locality-sensitive hashing: Euclidean distance

(Continued)

Proof of (b) "far points have lower collision probability"

If $h(\mathbf{p}) = h(\mathbf{q})$ then

$$\gamma \geq |\mathbf{p}' - \mathbf{q}'| \; = \|\mathbf{p} - \mathbf{q}\| |\cos\alpha|$$

Locality-sensitive hashing: Euclidean distance

(Continued)

Proof of (b) "far points have lower collision probability"

If $h(\mathbf{p}) = h(\mathbf{q})$ then

$$\gamma \geq |\mathbf{p}' - \mathbf{q}'| = \|\mathbf{p} - \mathbf{q}\| |\cos\alpha| \geq 2\gamma |\cos\alpha|$$

Locality-sensitive hashing: Euclidean distance

(Continued)

Proof of (b) "far points have lower collision probability"

If $h(\mathbf{p}) = h(\mathbf{q})$ then

$$\gamma \geq |\mathbf{p}' - \mathbf{q}'| = \|\mathbf{p} - \mathbf{q}\| |\cos\alpha| \quad \geq 2\gamma |\cos\alpha|$$

This implies

$$|\cos\alpha| \leq \tfrac{1}{2}$$

Locality-sensitive hashing: Euclidean distance

(Continued)

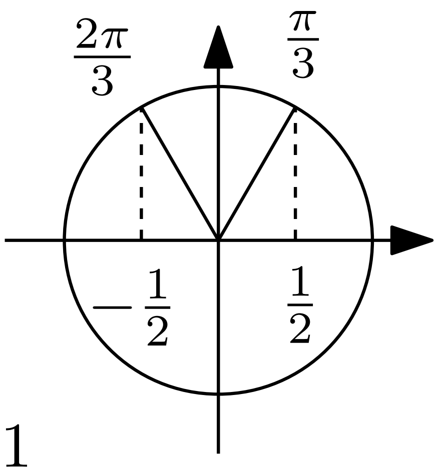
Proof of (b) "far points have lower collision probability"

If $h(\mathbf{p}) = h(\mathbf{q})$ then

$$\gamma \geq |\mathbf{p}' - \mathbf{q}'| = \|\mathbf{p} - \mathbf{q}\| |\cos\alpha| \quad \geq 2\gamma |\cos\alpha|$$

This implies

$$|\cos\alpha| \leq \tfrac{1}{2}$$



Since α is uniformly random in $(0, \pi)$, we have

$$\Pr\left[|\cos\alpha| \leq \frac{1}{2}\right] = \Pr\left[\alpha \in \left[\frac{\pi}{3}, \frac{2\pi}{3}\right]\right] = \frac{1}{3}$$

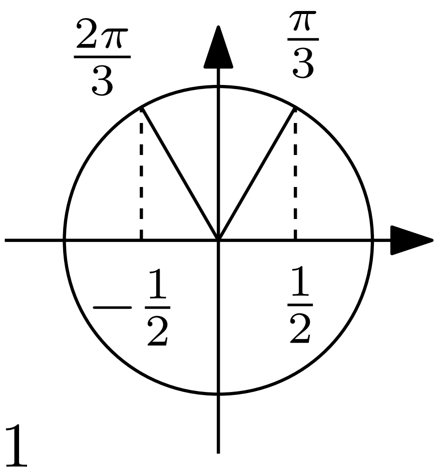
Locality-sensitive hashing: Euclidean distance

(Continued)

Proof of (b) "far points have lower collision probability"

If $h(\mathbf{p}) = h(\mathbf{q})$ then

$$\gamma \geq |\mathbf{p}' - \mathbf{q}'| = \|\mathbf{p} - \mathbf{q}\| |\cos\alpha| \geq 2\gamma |\cos\alpha|$$



This implies

$$|\cos\alpha| \leq \tfrac{1}{2}$$

Since α is uniformly random in $(0, \pi)$, we have

$$\Pr\left[|\cos\alpha| \leq \frac{1}{2}\right] = \Pr\left[\alpha \in \left[\frac{\pi}{3}, \frac{2\pi}{3}\right]\right] = \frac{1}{3}$$

Caveat: This argument only works for $\mathbf{p}, \mathbf{q} \in \mathbb{R}^2$

Locality-sensitive hashing

Definition:
A family of hash functions H is called (d_1, d_2, p_1, p_2)-locality-sensitive if for $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$:

$$(a) \quad \text{if } \mathrm{d}(\mathbf{p}, \mathbf{q}) \le d_1 \quad \text{then} \quad \Pr[h(\mathbf{p}) = h(\mathbf{q})] \ge p_1$$

$$(b) \quad \text{if } \mathrm{d}(\mathbf{p}, \mathbf{q}) \ge d_2 \quad \text{then} \quad \Pr[h(\mathbf{p}) = h(\mathbf{q})] \le p_2$$

In general, we want $d_1 < d_2$ and $p_1 > p_2$

We saw

- $(\frac{\gamma}{2}, 2\gamma, \frac{1}{2}, \frac{1}{3})$-locality-sensitive hashing scheme for the Euclidean distance
- $(\alpha_1, \alpha_2, \frac{2\pi - \alpha_1}{2\pi}, \frac{2\pi - \alpha_2}{2\pi})$-locality-sensitive hashing scheme for the Arccos distance

Locality-sensitive hashing

Definition:
A family of hash functions H is called (d_1, d_2, p_1, p_2)-locality-sensitive if for $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$:

(a) if $\mathrm{d}(\mathbf{p}, \mathbf{q}) \leq d_1$ then $\Pr[h(\mathbf{p}) = h(\mathbf{q})] \geq p_1$

(b) if $\mathrm{d}(\mathbf{p}, \mathbf{q}) \geq d_2$ then $\Pr[h(\mathbf{p}) = h(\mathbf{q})] \leq p_2$

In general, we want $d_1 < d_2$ and $p_1 > p_2$

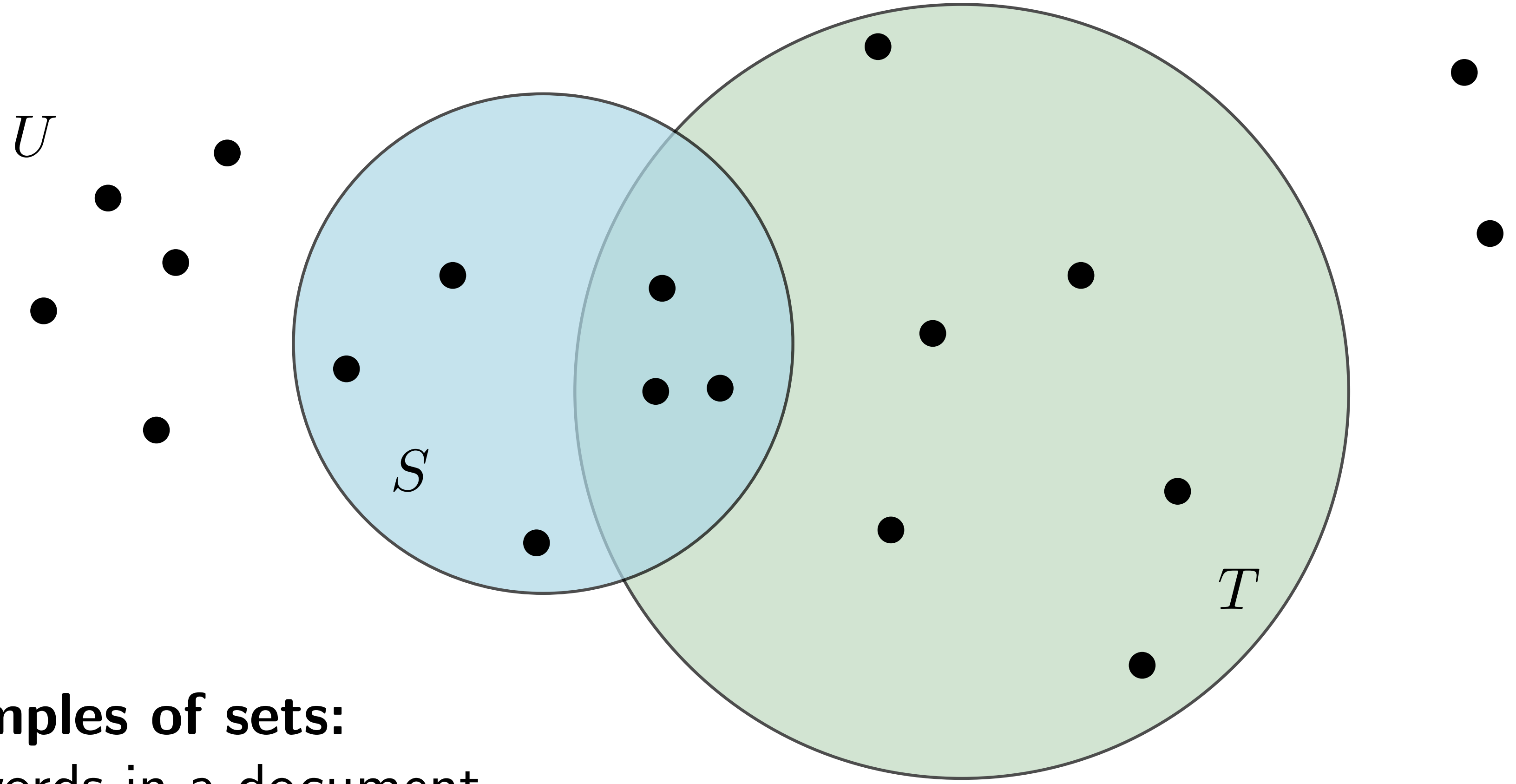
We saw

- $(\frac{\gamma}{2}, 2\gamma, \frac{1}{2}, \frac{1}{3})$-locality-sensitive hashing scheme for the Euclidean distance
- $(\alpha_1, \alpha_2, \frac{2\pi-\alpha_1}{2\pi}, \frac{2\pi-\alpha_2}{2\pi})$-locality-sensitive hashing scheme for the Arccos distance

What about other distance measures?

Jaccard Similarity

Similarity function to compare sets.



Examples of sets:

- words in a document
- products in a shopping basket
- movies liked by a person

Definition:

$$\mathrm{sim}_{\mathcal{J}}(S,T) := \frac{|S \cap T|}{|S \cup T|}$$

Jaccard Similarity

We represent the sets $S, T \subseteq U$ using indicator vectors.

Example

- Given set of documents $D_1, \ldots, D_n$.
- Let S_i be the set of words contained in D_i
- Indicator vector for S_i is a $(0,1)$-vector over the dictionary U

$$v_i = (\ldots, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, \ldots)$$

'cat'

'catastrophy'

'caterpillar' $\in S_i$

'category' $\notin S_i$

Jaccard Similarity – Minhashing

Minhashing for estimating the Jaccard similarity

Characteristic matrix with indicator vectors as columns

	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Jaccard Similarity – Minhashing

Minhashing for estimating the Jaccard similarity

Characteristic matrix with indicator vectors as columns

	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

randomly permute rows

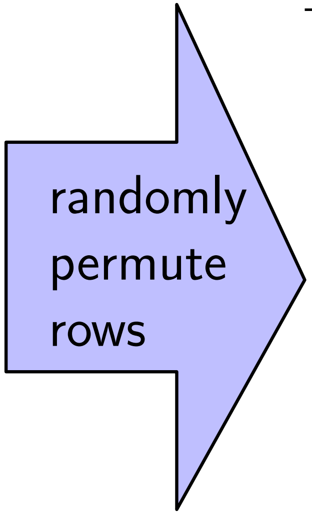
	S_1	S_2	S_3	S_4
b	0	0	1	0
a	1	0	0	1
c	0	1	0	1
e	0	0	1	0
d	1	0	1	1

Jaccard Similarity – Minhashing

Minhashing for estimating the Jaccard similarity

Characteristic matrix with indicator vectors as columns

	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0



	S_1	S_2	S_3	S_4
b	0	0	$\boxed{1}$	0
a	$\boxed{1}$	0	0	$\boxed{1}$
c	0	$\boxed{1}$	0	1
e	0	0	1	0
d	1	0	1	1

Minhash $h(S_i)$ is the index of first row from the top which has a 1

Jaccard Similarity – Minhashing

Minhashing for estimating the Jaccard similarity

Characteristic matrix with indicator vectors as columns

	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

randomly permute rows

	S_1	S_2	S_3	S_4
b	0	0	$\boxed{1}$	0
a	$\boxed{1}$	0	0	$\boxed{1}$
c	0	$\boxed{1}$	0	1
e	0	0	1	0
d	1	0	1	1

Minhash $h(S_i)$ is the index of first row from the top which has a 1

Claim:

$$\Pr\left[h(S_i) = h(S_j)\right] = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$$

Jaccard Similarity – Minhashing

Claim: $\Pr\left[h(S_i) = h(S_j)\right] = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

	S_1	S_2	S_3	S_4
b	0	0	1	0
a	1	0	0	1
c	0	1	0	1
e	0	0	1	0
d	1	0	1	1

Jaccard Similarity – Minhashing

Claim: $\Pr\left[h(S_i) = h(S_j)\right] = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Is it true for S_1 and S_2?

	S_1	S_2	S_3	S_4
b	0	0	1	0
a	1	0	0	1
c	0	1	0	1
e	0	0	1	0
d	1	0	1	1

Jaccard Similarity – Minhashing

Claim: $\Pr\left[h(S_i) = h(S_j)\right] = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Is it true for S_1 and S_2?

$\mathrm{sim}_{\mathcal{J}}(S_1, S_2) = 0$

$\Pr\left[h(S_1) = h(S_2)\right] = 0$

	S_1	S_2	S_3	S_4
b	0	0	1	0
a	1	0	0	1
c	0	1	0	1
e	0	0	1	0
d	1	0	1	1

Jaccard Similarity – Minhashing

Claim: $\Pr[h(S_i) = h(S_j)] = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Is it true for S_1 and S_2?

$\mathrm{sim}_{\mathcal{J}}(S_1, S_2) = 0$

$\Pr[h(S_1) = h(S_2)] = 0$

Is it true for S_3 and S_4?

	S_1	S_2	S_3	S_4
b	0	0	1	0
a	1	0	0	1
c	0	1	0	1
e	0	0	1	0
d	1	0	1	1

Jaccard Similarity – Minhashing

Claim: $\Pr[h(S_i) = h(S_j)] = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Is it true for S_1 and S_2?

$\mathrm{sim}_{\mathcal{J}}(S_1, S_2) = 0$
$\Pr[h(S_1) = h(S_2)] = 0$

Is it true for S_3 and S_4?

$\mathrm{sim}_{\mathcal{J}}(S_3, S_4) = \frac{1}{5}$
$\Pr[h(S_3) = h(S_4)] = \frac{1}{5}$

	S_1	S_2	S_3	S_4
b	0	0	1	0
a	1	0	0	1
c	0	1	0	1
e	0	0	1	0
d	1	0	1	1

Jaccard Similarity – Minhashing

Claim: $\Pr\left[h(S_i) = h(S_j)\right] = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Is it true for S_1 and S_2?

$\mathrm{sim}_{\mathcal{J}}(S_1, S_2) = 0$
$\Pr\left[h(S_1) = h(S_2)\right] = 0$

Is it true for S_3 and S_4?

$\mathrm{sim}_{\mathcal{J}}(S_3, S_4) = \frac{1}{5}$
$\Pr\left[h(S_3) = h(S_4)\right] = \frac{1}{5}$

	S_1	S_2	S_3	S_4
b	0	0	1	0
a	1	0	0	1
c	0	1	0	1
e	0	0	1	0
d	1	0	1	1

Proof:

$x := |S_i \cap S_j|$ (i.e., number of $(1, 1)$ rows)

Jaccard Similarity – Minhashing

Claim: $\Pr\left[h(S_i) = h(S_j)\right] = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Is it true for S_1 and S_2?

$\mathrm{sim}_{\mathcal{J}}(S_1, S_2) = 0$
$\Pr\left[h(S_1) = h(S_2)\right] = 0$

Is it true for S_3 and S_4?

$\mathrm{sim}_{\mathcal{J}}(S_3, S_4) = \frac{1}{5}$
$\Pr\left[h(S_3) = h(S_4)\right] = \frac{1}{5}$

	S_1	S_2	S_3	S_4
b	0	0	1	0
a	1	0	0	1
c	0	1	0	1
e	0	0	1	0
d	1	0	1	1

Proof:

$x := |S_i \cap S_j|$ (i.e., number of $(1, 1)$ rows)

$y := |S_i \cup S_j| - |S_i \cap S_j|$ (i.e., number of $(0, 1)$ and $(1, 0)$ rows)

Jaccard Similarity – Minhashing

Claim: $\Pr[h(S_i) = h(S_j)] = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Is it true for S_1 and S_2?

$\mathrm{sim}_{\mathcal{J}}(S_1, S_2) = 0$
$\Pr[h(S_1) = h(S_2)] = 0$

Is it true for S_3 and S_4?

$\mathrm{sim}_{\mathcal{J}}(S_3, S_4) = \frac{1}{5}$
$\Pr[h(S_3) = h(S_4)] = \frac{1}{5}$

	S_1	S_2	S_3	S_4
b	0	0	1	0
a	1	0	0	1
c	0	1	0	1
e	0	0	1	0
d	1	0	1	1

Proof:

$x := |S_i \cap S_j|$ (i.e., number of $(1,1)$ rows)
$y := |S_i \cup S_j| - |S_i \cap S_j|$ (i.e., number of $(0,1)$ and $(1,0)$ rows)

$$\mathrm{sim}_{\mathcal{J}}(S_i, S_j) = \frac{x}{x+y} = Pr[h(S_i) = h(S_j)]$$

$\square$

Jaccard Similarity – Signature Matrix

Now repeat and create hash functions $h_1, h_2, \ldots, h_m$

	S_1	S_2	S_3	S_4	h_1	h_2	$\ldots$
0	1	0	0	1	1	1	$\vdots$
1	0	0	1	0	2	4	
2	0	1	0	1	3	2	
3	1	0	1	1	4	0	
4	0	0	1	0	0	3	

For each set we obtain a **minhash signature**:

	S_1	S_2	S_3	S_4
h_1 :				
h_2 :				
$\vdots$				

Jaccard Similarity – Signature Matrix

Now repeat and create hash functions $h_1, h_2, \ldots, h_m$

	S_1	S_2	S_3	S_4	h_1	h_2	$\ldots$
0	$\boxed{1}$	0	0	$\boxed{1}$	1	1	$\vdots$
1	0	0	1	0	2	4	
2	0	$\boxed{1}$	0	1	3	2	
3	1	0	1	1	4	0	
4	0	0	$\boxed{1}$	0	0	3	

For each set we obtain a **minhash signature**:

	S_1	S_2	S_3	S_4
h_1 :	1	3	0	1
h_2 :				
$\vdots$				

Jaccard Similarity – Signature Matrix

Now repeat and create hash functions $h_1, h_2, \ldots, h_m$

	S_1	S_2	S_3	S_4	h_1	h_2	$\ldots$
0	1	0	0	1	1	1	$\vdots$
1	0	0	1	0	2	4	
2	0	1	0	1	3	2	
3	1	0	1	1	4	0	
4	0	0	1	0	0	3	

For each set we obtain a **minhash signature**:

	S_1	S_2	S_3	S_4
h_1 :	1	3	0	1
h_2 :				
$\vdots$				

Jaccard Similarity – Signature Matrix

Now repeat and create hash functions $h_1, h_2, \dots, h_m$

	S_1	S_2	S_3	S_4	h_1	h_2	$\dots$
0	1	0	0	1	1	1	$\vdots$
1	0	0	1	0	2	4	
2	0	1	0	1	3	2	
3	1	0	1	1	4	0	
4	0	0	1	0	0	3	

For each set we obtain a **minhash signature**:

	S_1	S_2	S_3	S_4
h_1 :	1	3	0	1
h_2 :	0	2	0	0
$\vdots$				

Jaccard Similarity – Signature Matrix

Now repeat and create hash functions $h_1, h_2, \ldots, h_m$

	S_1	S_2	S_3	S_4	h_1	h_2	$\ldots$
0	1	0	0	1	1	1	$\vdots$
1	0	0	1	0	2	4	
2	0	1	0	1	3	2	
3	1	0	1	1	4	0	
4	0	0	1	0	0	3	

For each set we obtain a **minhash signature**:

	S_1	S_2	S_3	S_4
h_1 :	1	3	0	1
h_2 :	0	2	0	0
$\vdots$				

Jaccard Similarity – Signature Matrix

Now repeat and create hash functions $h_1, h_2, \ldots, h_m$

	S_1	S_2	S_3	S_4	h_1	h_2	$\ldots$
0	1	0	0	1	1	1	$\vdots$
1	0	0	1	0	2	4	
2	0	1	0	1	3	2	
3	1	0	1	1	4	0	
4	0	0	1	0	0	3	

For each set we obtain a **minhash signature**:

	S_1	S_2	S_3	S_4
h_1 :	1	3	0	1
h_2 :	0	2	0	0
$\vdots$				

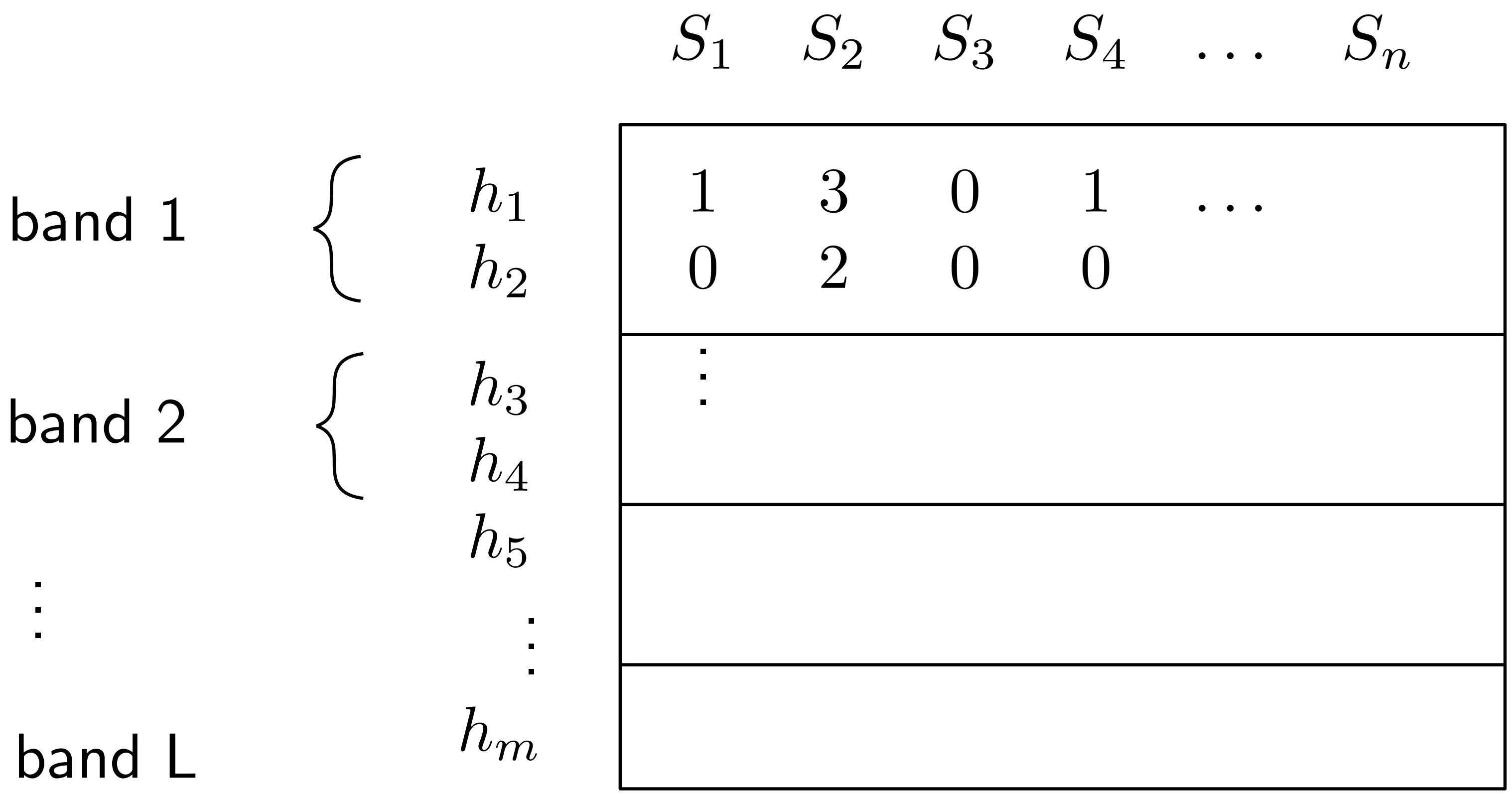
minhash signature of S_2 is $(3, 2)$

Jaccard Similarity – Banding technique

		S_1	S_2	S_3	S_4	$\dots$	S_n
band 1	h_1	1	3	0	1	$\dots$	
	h_2	0	2	0	0		
		$\vdots$					
band 2	h_3						
	h_4						
	h_5						
$\vdots$	$\vdots$						
band L	h_m						

Divide the rows of the signature matrix into bands of size k

Jaccard Similarity – Banding technique



Divide the rows of the signature matrix into bands of size k

If S_i and S_j have equal minhash signature within some band, we consider them as **candidates**

Jaccard Similarity – Banding technique

If S_i and S_j have equal minhash signature within some band, we consider them as **candidates**

What is the probability of two sets becoming candidates?

Jaccard Similarity – Banding technique

If S_i and S_j have equal minhash signature within some band, we consider them as **candidates**

What is the probability of two sets becoming candidates?

Let $s = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Event	**Probability**
They agree in all rows of a particular band:	
They do not agree in a particular band:	
They do not agree in any of the bands:	
They become candidates:	

Jaccard Similarity – Banding technique

If S_i and S_j have equal minhash signature within some band, we consider them as **candidates**

What is the probability of two sets becoming candidates?

Let $s = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Event	**Probability**
They agree in all rows of a particular band:	s^k
They do not agree in a particular band:	
They do not agree in any of the bands:	
They become candidates:	

Jaccard Similarity – Banding technique

If S_i and S_j have equal minhash signature within some band, we consider them as **candidates**

What is the probability of two sets becoming candidates?

Let $s = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Event	**Probability**
They agree in all rows of a particular band:	s^k
They do not agree in a particular band:	$1 - s^k$
They do not agree in any of the bands:	
They become candidates:	

Jaccard Similarity – Banding technique

If S_i and S_j have equal minhash signature within some band, we consider them as **candidates**

What is the probability of two sets becoming candidates?

Let $s = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Event	**Probability**
They agree in all rows of a particular band:	s^k
They do not agree in a particular band:	$1 - s^k$
They do not agree in any of the bands:	$(1 - s^k)^L$
They become candidates:	

Jaccard Similarity – Banding technique

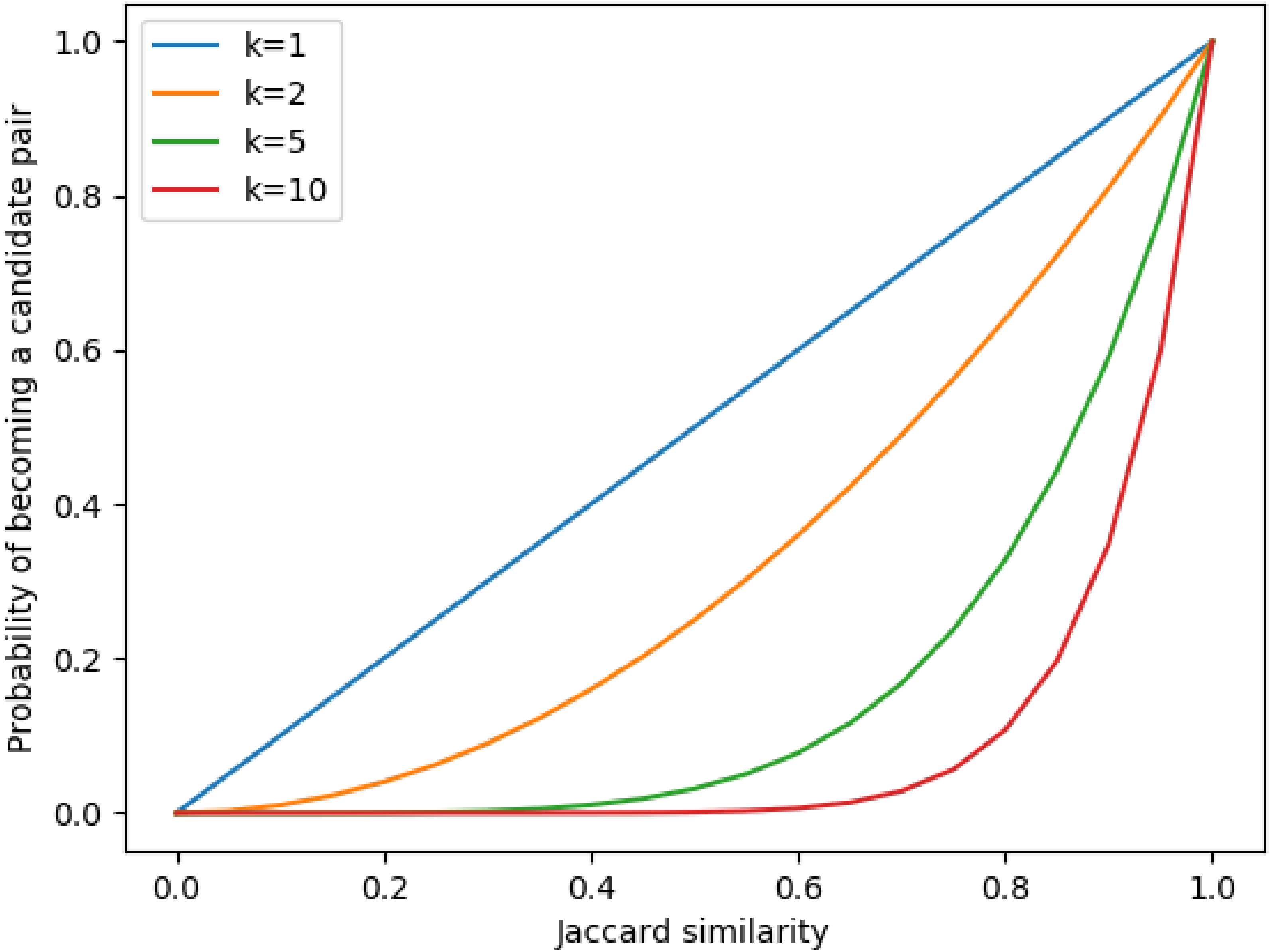
If S_i and S_j have equal minhash signature within some band, we consider them as **candidates**

What is the probability of two sets becoming candidates?

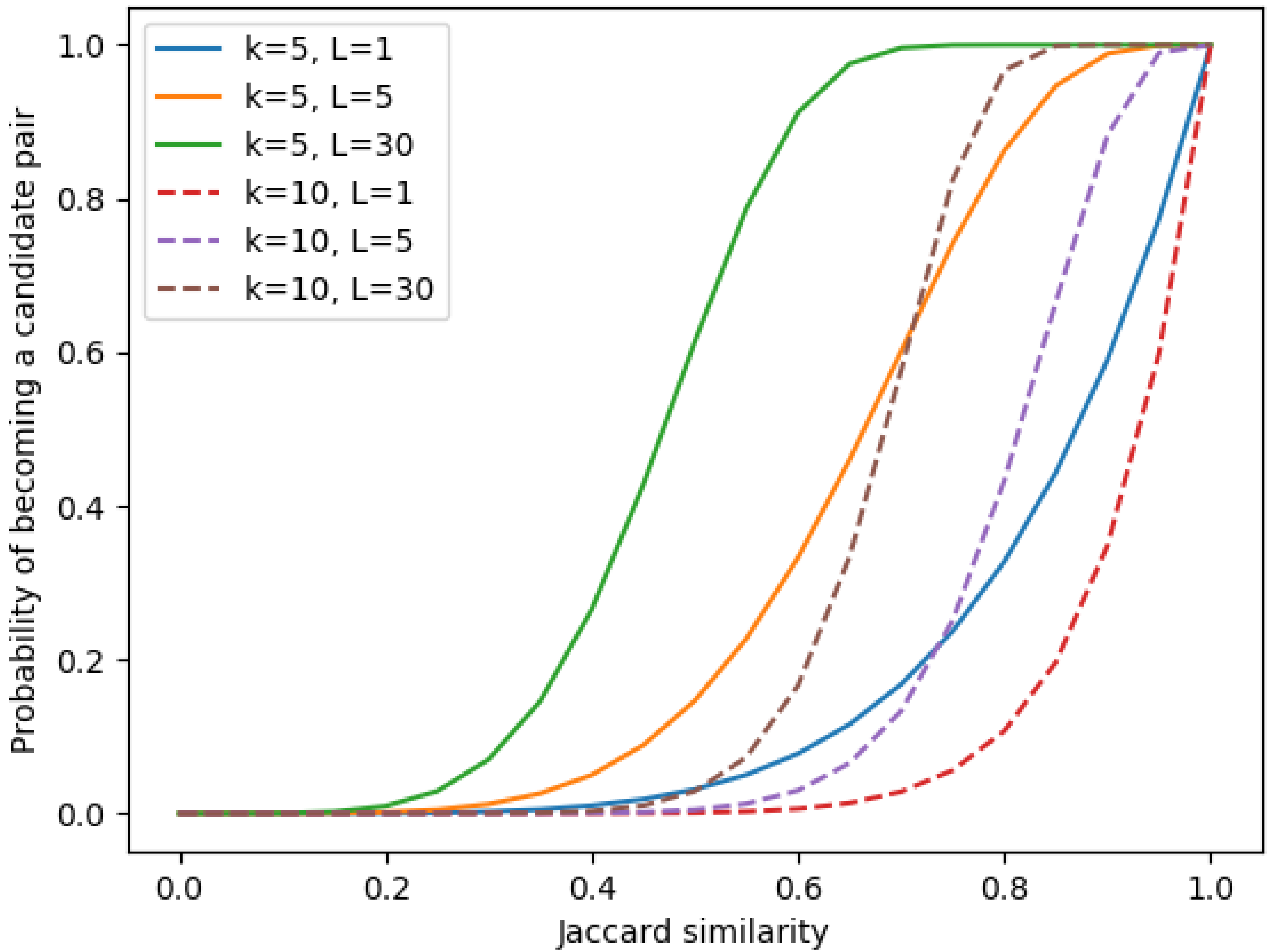
Let $s = \mathrm{sim}_{\mathcal{J}}(S_i, S_j)$

Event	**Probability**
They agree in all rows of a particular band:	s^k
They do not agree in a particular band:	$1 - s^k$
They do not agree in any of the bands:	$(1 - s^k)^L$
They become candidates:	$1 - (1 - s^k)^L$

Jaccard Similarity – Banding technique



Jaccard Similarity – Banding technique



Amplification of an LSH

In general, this process is called **amplification** (we "amplify" the success probabilities)

Let H be a (d_1, d_2, p_1, p_2)-sensitive family of hash functions

AND-construction:

$g(\mathbf{p}) = g(\mathbf{q})$ if and only if $h_i(\mathbf{p}) = h_i(\mathbf{q})$ for all $1 \leq i \leq r$

yields a (d_1, d_2, p_1^r, p_2^r)-sensitive family

OR-construction:

$g(\mathbf{p}) = g(\mathbf{q})$ if and only if $h_i(\mathbf{p}) = h_i(\mathbf{q})$ for some $1 \leq i \leq L$

yields a $(d_1, d_2, 1 - (1 - p_1)^L, 1 - (1 - p_2)^L)$-sensitive family

Summary

- Nearest-Neighbor rule
- Locality sensitive hashing
- Cosine distance
- Euclidean distance
- Jaccard Similarity
- Minhashing
- Banding
- Amplification

References

- Lescovec, Rajaraman, and Ullman
  Mining of Massive Datasets
- Sariel Har-Peled, Piotr Indyk, Rajeev Motwani
  "Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality" Theory of Computing 8(2012),pp. 321-350
- Alexandr Andoni, Nearest Neighbor Search: the Old, the New, and the Impossible (Dissertation) 2009,