

10-605/805 – ML for Large Datasets

Lecture 10: Randomized Algorithms

Henry Chai

9/29/22

Front Matter

- HW3 released 9/23, due 10/4 at 11:59 PM
- Midterm exam on 10/11, two weeks from today!
 - Lecture on 10/6 (next Thursday) will be an *optional* practice exam - these will not be collected/graded
 - Recitation on 10/7 (next Friday) will go through the solutions
 - Both the practice exam and the solutions will be released after the Recitation
- ✗ • Lecture on 10/4 (next Tuesday) will be an AWS tutorial; please bring your laptops to class on that day

Two Specific Applications of Feature Hashing

1. *Count-min sketch* for logistic regression
2. *Locality sensitive hashing* for nearest neighbors/clustering

Two Specific Applications of Feature Hashing

1. *Count-min sketch* for logistic regression
2. *Locality sensitive hashing* for nearest neighbors/clustering

Recall: Count-min Sketch

- Data structure used to estimate the frequency of items in some stream of inputs
 - Running example: finding “viral” terms in Google searches (<https://trends.google.com/trends/?geo=US>)
 - Naïve approach: just keep an array with an index for every possible search term and add one to that index when the term appears
 - Massive array
 - Could be dynamic/need to grow if unseen search terms arrive

Count-min Sketch: Connection to Logistic Regression

- If \mathbf{x} is a (sparse) one-hot encoded vector, then we can efficiently compute $\mathbf{w}^{(t)^T} \mathbf{x}$ as

$$\sum_{j : x_j \neq 0} w_j^{(t)} x_j \approx \sum_{j : x_j \neq 0} \left(\min_i C[i, h_i(j)] \right) x_j$$

1	2	3	...	m
$C[1,1]$	$C[1,2]$	$C[1,3]$...	$C[1, m]$
\vdots	\vdots	\vdots	\ddots	\vdots
$C[r, 1]$	$C[r, 2]$	$C[r, 3]$...	$C[r, m]$

- This matrix holds a compact representation of $\mathbf{w}^{(t)}$ at each iteration

Count-min Sketch: Algorithm

- Input: r independent hash functions h_1, \dots, h_r that each map to m buckets
1. Initialize an $r \times m$ count matrix, C , to all zeros
 2. For each item, s , in some stream of data:
 - a. For $i \in \{1, \dots, r\}$:
 - i. $C[i, h_i(s)] += 1$
 3. For any item s , return $\hat{c}_s = \min_i(C[i, h_i(s)])$ as an approximation c_s , the true number of occurrences of s

Observation:
Count-min
sketch can only
overestimate
counts!

But by how
much?

- Input: r independent hash functions h_1, \dots, h_r that each map to m buckets
1. Initialize an $r \times m$ count matrix, C , to all zeros
 2. For each item, s , in some stream of data:
 - a. For $i \in \{1, \dots, r\}$:
 - i. $C[i, h_i(s)] += 1$
 3. For any item s , return $\hat{c}_s = \min_i(C[i, h_i(s)])$ as an approximation c_s , the true number of occurrences of s

Count-min Sketch: Theoretical Guarantees

- Given $r > \log_2 \frac{1}{\delta}$ hash functions that each map to $m > \frac{2}{\epsilon}$ buckets, then $\forall s$

$$P(\hat{c}_s \geq c_s + \epsilon \|c\|_1) \leq \delta$$

where $\|c\|_1$ is the total number of items in the stream.

- Key assumptions:

1. All hash functions are independent :

$$P(h_i(s) \neq h_j(s')) = P(h_i(s)) \quad \forall i, j, s, s'$$

2. Each individual hash function $\xrightarrow{\text{pairwise-independent}}$

$$P(h_i(s) = b \cap h_i(s') = b') = \frac{1}{m^2}$$

$$\Rightarrow P(h_i(s) = h_i(s')) = \frac{1}{m} \quad \forall i, s, s', b, b'$$

Count-min Sketch: Theoretical Guarantees Proof

- Given $r > \log_2 \frac{1}{\delta}$ hash functions that each map to $m > \frac{2}{\epsilon}$ buckets, then $\forall s$

$$P(\hat{c}_s \geq c_s + \epsilon \|c\|_1) \leq \delta$$

where $\|c\|_1$ is the total number of items in the stream.

- Proof:

For an arbitrary hash function h_i w/ item s :

$$E[C[i, h_i(s)]] = c_s + \sum_{s' \neq s} \zeta_{s'} (P(h_i(s) = h_i(s')))$$

(by pairwise independence) $= c_s + \sum_{s' \neq s} \frac{1}{m} c_{s'}$

(by the definition of m) $\leq c_s + \frac{1}{(2/\epsilon)} \sum_{s' \neq s} c_{s'}$
 $\leq c_s + \frac{\epsilon}{2} \|c\|_1$

Count-min Sketch: Theoretical Guarantees Proof (Cont.)

- Given $r > \log_2 \frac{1}{\delta}$ hash functions that each map to $m > \frac{2}{\epsilon}$ buckets, then $\forall s$

$$P(\hat{c}_s \geq c_s + \epsilon \|c\|_1) \leq \delta$$

where $\|c\|_1$ is the total number of items in the stream.

- Proof (Continued):

$$\begin{aligned} P(\underbrace{C[i, h_i(s)]}_{\text{(by Markov's inequality)}} \geq c_s + \epsilon \|c\|_1) &= P(\underbrace{C[i, h_i(s)] - c_s}_{\mathbb{E}[C[i, h_i(s)]] - c_s} \geq \epsilon \|c\|_1) \\ &\leq \frac{\epsilon \|c\|_1}{\epsilon \|c\|_1} \\ &\leq \frac{c_s + \frac{\epsilon}{2} \|c\|_1 - c_s}{\epsilon \|c\|_1} \\ &= \frac{1}{2} \end{aligned}$$

(from the previous slide)

Count-min Sketch: Theoretical Guarantees Proof (Cont.)

- Given $r > \log_2 1/\delta$ hash functions that each map to $m > 2/\epsilon$ buckets, then $\forall s$

$$P(\hat{c}_s \geq c_s + \epsilon \|c\|_1) \leq \delta$$

where $\|c\|_1$ is the total number of items in the stream.

- Proof (Continued):

$$\begin{aligned} P(\hat{c}_s \geq c_s + \epsilon \|c\|_1) &= P\left(\min_i C[i, h_i(s)] \geq c_s + \epsilon \|c\|_1\right) \\ &= P\left(\bigcap_i C[i, h_i(s)] \geq c_s + \epsilon \|c\|_1\right) \\ &= \prod_{i=1}^r P(C[i, h_i(s)] \geq c_s + \epsilon \|c\|_1) \\ &\leq \frac{1}{2^r} \leq \delta \end{aligned}$$

(by independence)

(by definition of r)

Count-min Sketch: Summary

- Key idea: use multiple (independent) hash functions to negate the effect of collisions in any single hash function
- Nice theoretical guarantees
- Can address overestimation bias to make unbiased count estimators (see HW3)
- Can be extended to handle weighted updates as in SGD logistic regression → efficient model updates in high-dimensional settings (large k)
 - For more details see:
<https://arxiv.org/pdf/1711.02305.pdf>

Two Specific Applications of Feature Hashing

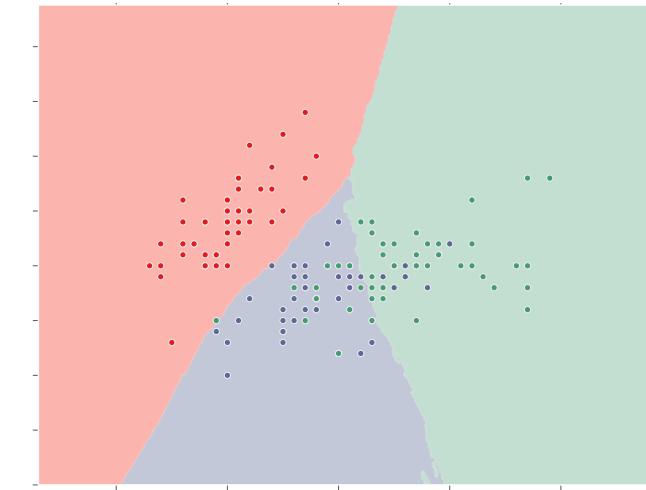
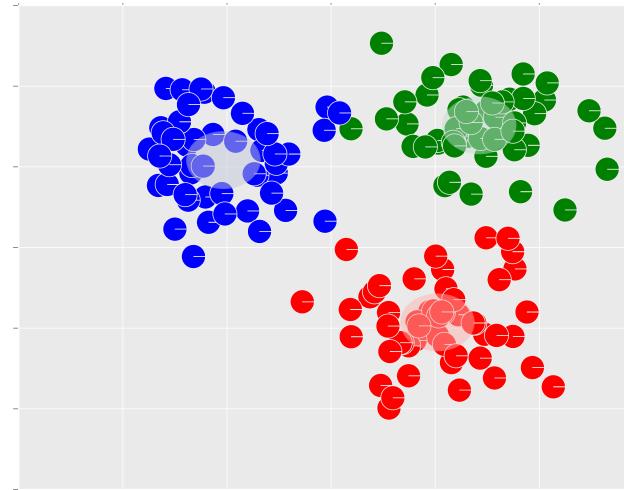
1. *Count-min sketch* for logistic regression
2. *Locality sensitive hashing* for nearest neighbors/clustering

Two Specific Applications of Feature Hashing

1. *Count-min sketch* for logistic regression
2. *Locality sensitive hashing* for nearest neighbors/clustering

Locality-Sensitive Hashing

- Key idea: use hash functions that map “similar” items to the same ~~function~~ *bucket*
 - Plot twist: collisions are the goal!
- Enables efficient comparison between points
- In the context of large-scale machine learning, can be used for clustering or nearest neighbor models in high-dimensional space



Locality-Sensitive Hashing: Formal Definition

- A *family* of hash functions \mathcal{F} that map $\mathbb{R}^k \rightarrow \{0, \dots, m\}$ is $(\epsilon, c\epsilon, \delta_1, \delta_2)$ -sensitive with respect to some distance metric d iff $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^k$, a hash function h chosen uniformly at random from \mathcal{F} has the property that
 - if $d(\mathbf{x}, \mathbf{y}) \leq \epsilon$, then $P(h(\mathbf{x}) = h(\mathbf{y})) \geq \delta_1$
 - “Points close to each other (according to some distance metric d) are likely to collide”
 - if $d(\mathbf{x}, \mathbf{y}) \geq c\epsilon$, then $P(h(\mathbf{x}) = h(\mathbf{y})) \leq \delta_2$
 - “Points far apart (according to some distance metric d) are unlikely to collide”

Locality-Sensitive Hashing: Example

- Random projection:
 - $m = 1$ so our hash functions map \mathbb{R}^k to $\{0,1\}$
 - Let \mathcal{F} be the set of all linear decision boundaries:

$$\mathcal{F} = \left\{ h : h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases} \right\}$$

- $d(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$, the *cosine distance*

Recall: $\mathbf{x}^T \mathbf{y} = \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos(\theta)$ where
 θ is the angle between
vectors \mathbf{x} and \mathbf{y}

Locality-Sensitive Hashing: Example

- Random projection:
 - $m = 1$ so our hash functions map \mathbb{R}^k to $\{0,1\}$
 - Let \mathcal{F} be the set of all linear decision boundaries:

$$\mathcal{F} = \left\{ h : h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases} \right\}$$

- $d(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$, the *cosine distance*
- For a random weight vector \mathbf{w} (e.g., weights sampled independently from a standard Gaussian) ?

$$\begin{aligned} P(h_{\mathbf{w}}(\mathbf{x}) = h_{\mathbf{w}}(\mathbf{y})) &= 1 - \frac{\arccos(1 - d(\mathbf{x}, \mathbf{y}))}{\pi} \\ &= 1 - \frac{\theta}{\pi} \end{aligned}$$

Locality-Sensitive Hashing: Example

- Random projection:
 - $m = 1$ so our hash functions map \mathbb{R}^k to $\{0,1\}$
 - Let \mathcal{F} be the set of all linear decision boundaries:

$$\mathcal{F} = \left\{ h : h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases} \right\}$$

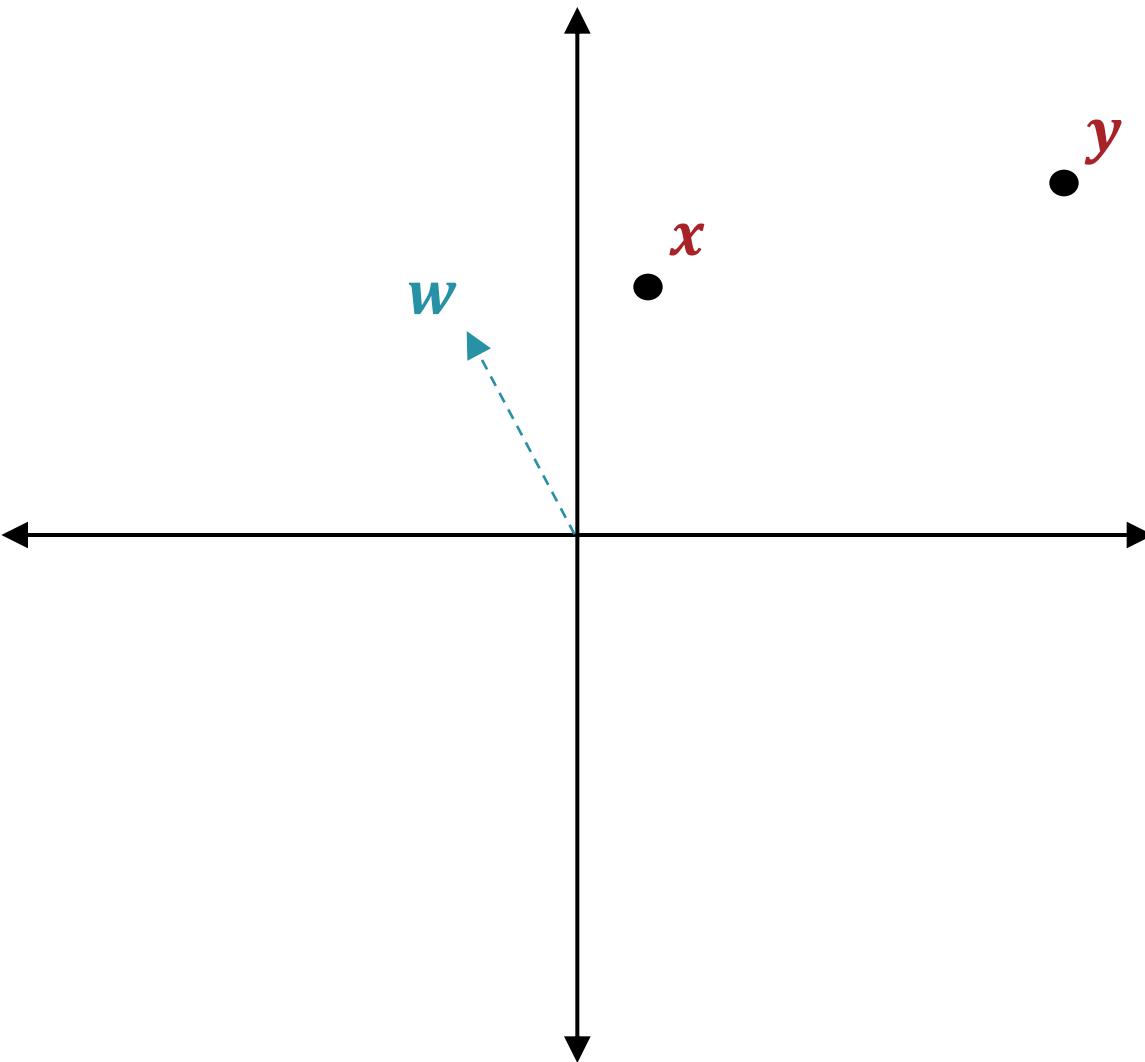
- $d(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$, the *cosine distance*
- For a random weight vector \mathbf{w} (e.g., weights sampled independently from a standard Gaussian)

$$P(h_{\mathbf{w}}(\mathbf{x}) = h_{\mathbf{w}}(\mathbf{y})) = 1 - \frac{\theta}{\pi}$$

- \mathcal{F} is $\left(\epsilon, c\epsilon, 1 - \frac{\arccos 1-\epsilon}{\pi}, 1 - \frac{\arccos 1-c\epsilon}{\pi}\right)$ -sensitive

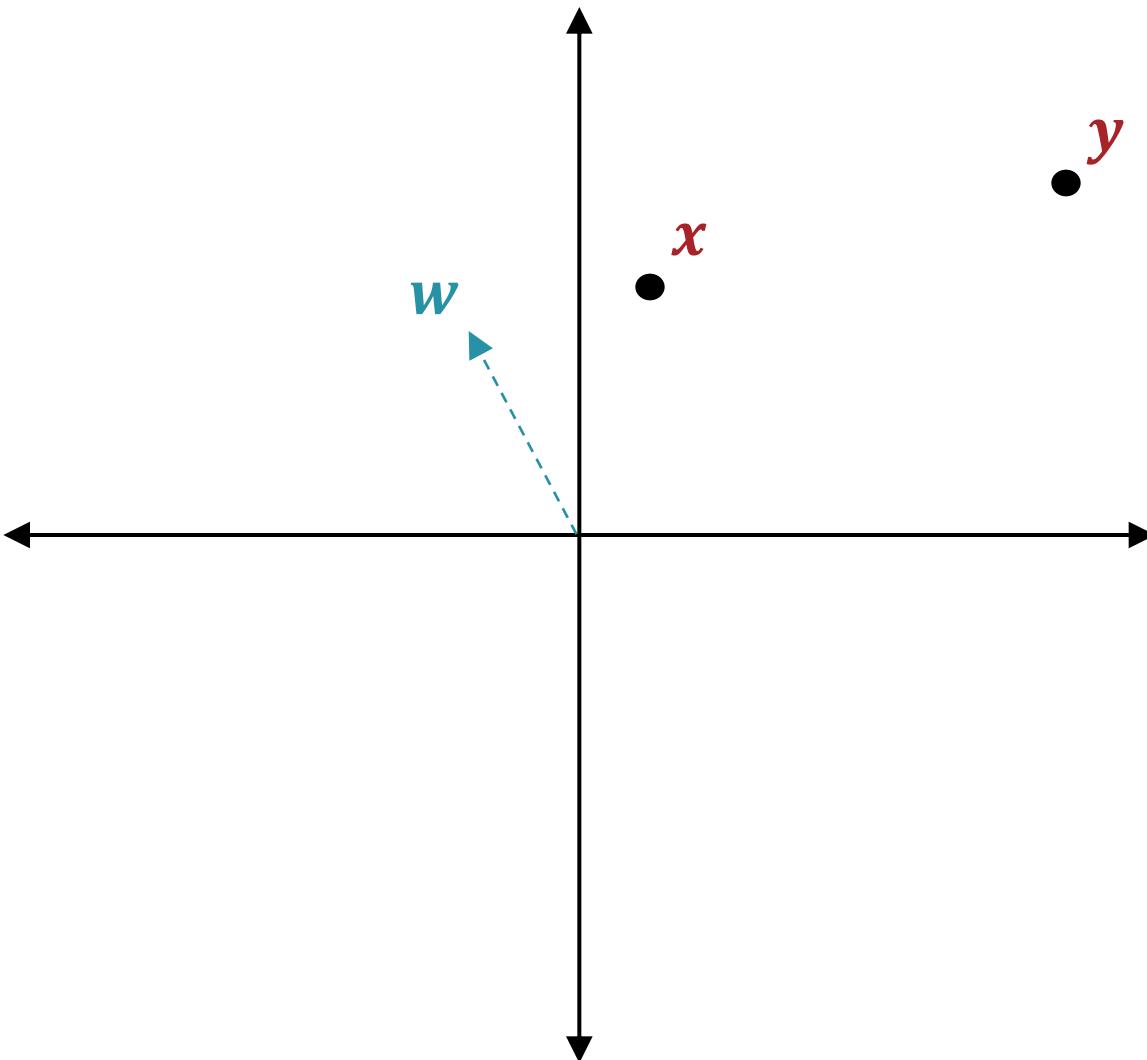
Locality-Sensitive Hashing: Example

- Random projection:



What does the decision boundary corresponding to w look like?

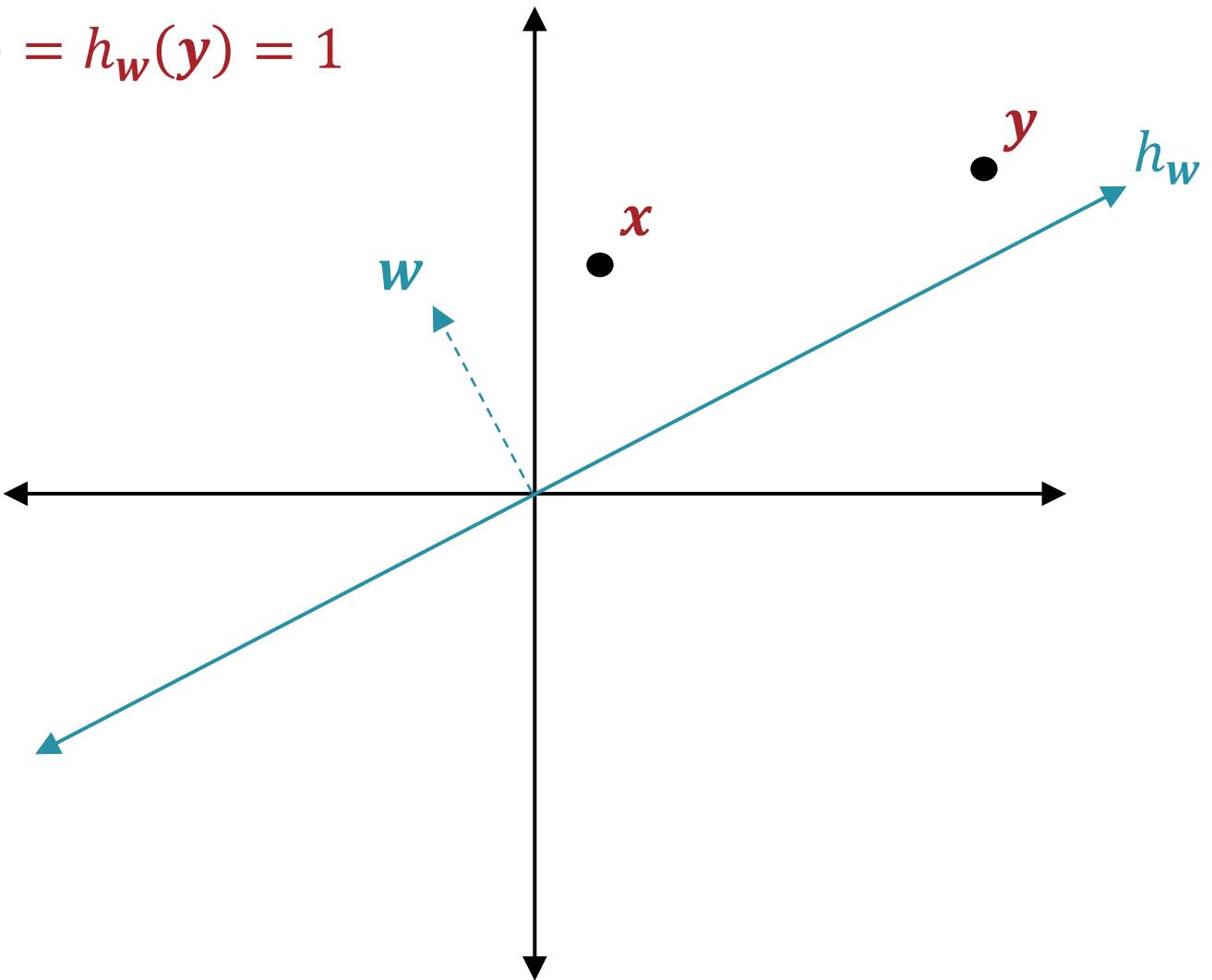
- Random projection:



Locality-Sensitive Hashing: Example

- Random projection:

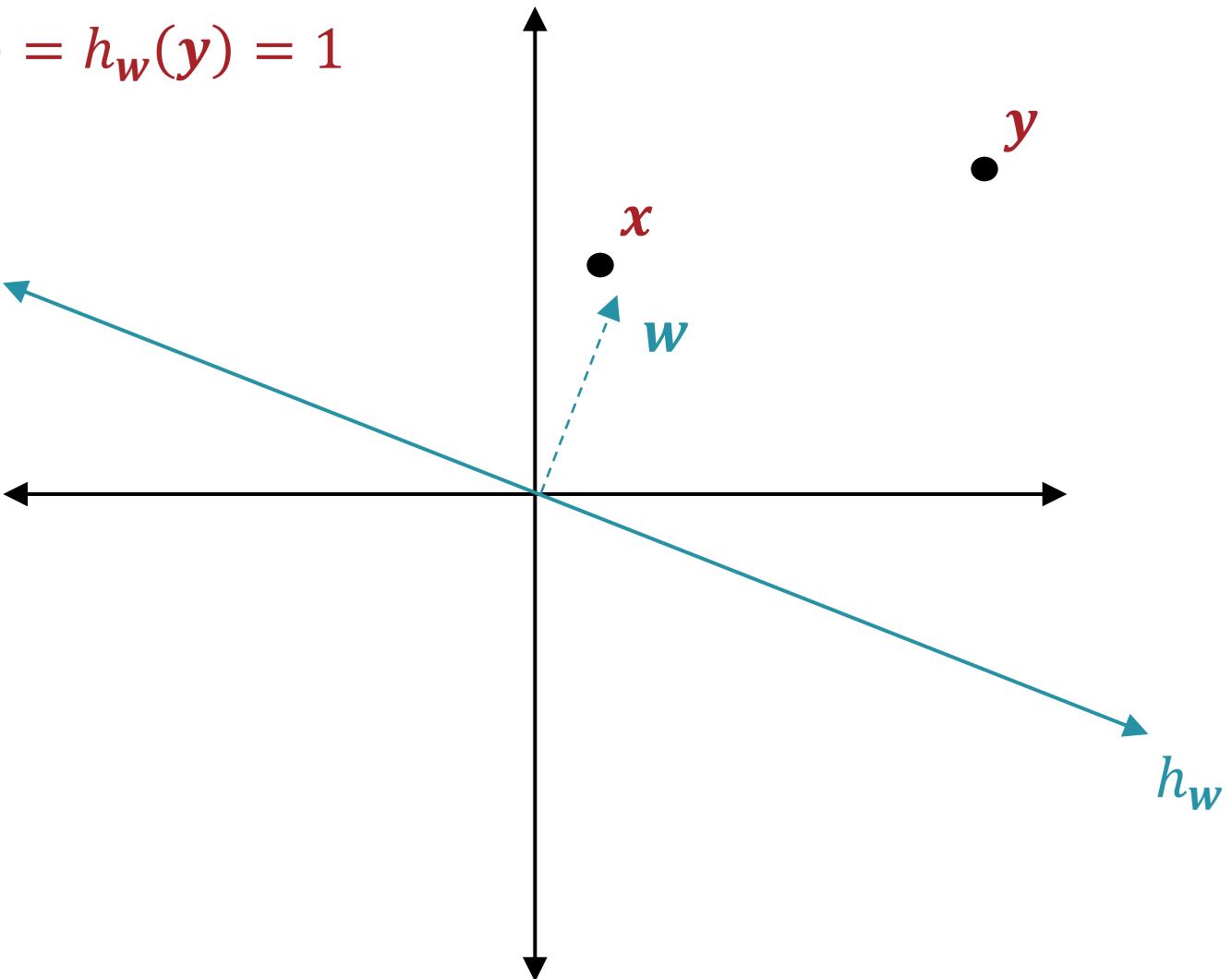
$$h_w(x) = h_w(y) = 1$$



Locality-Sensitive Hashing: Example

- Random projection:

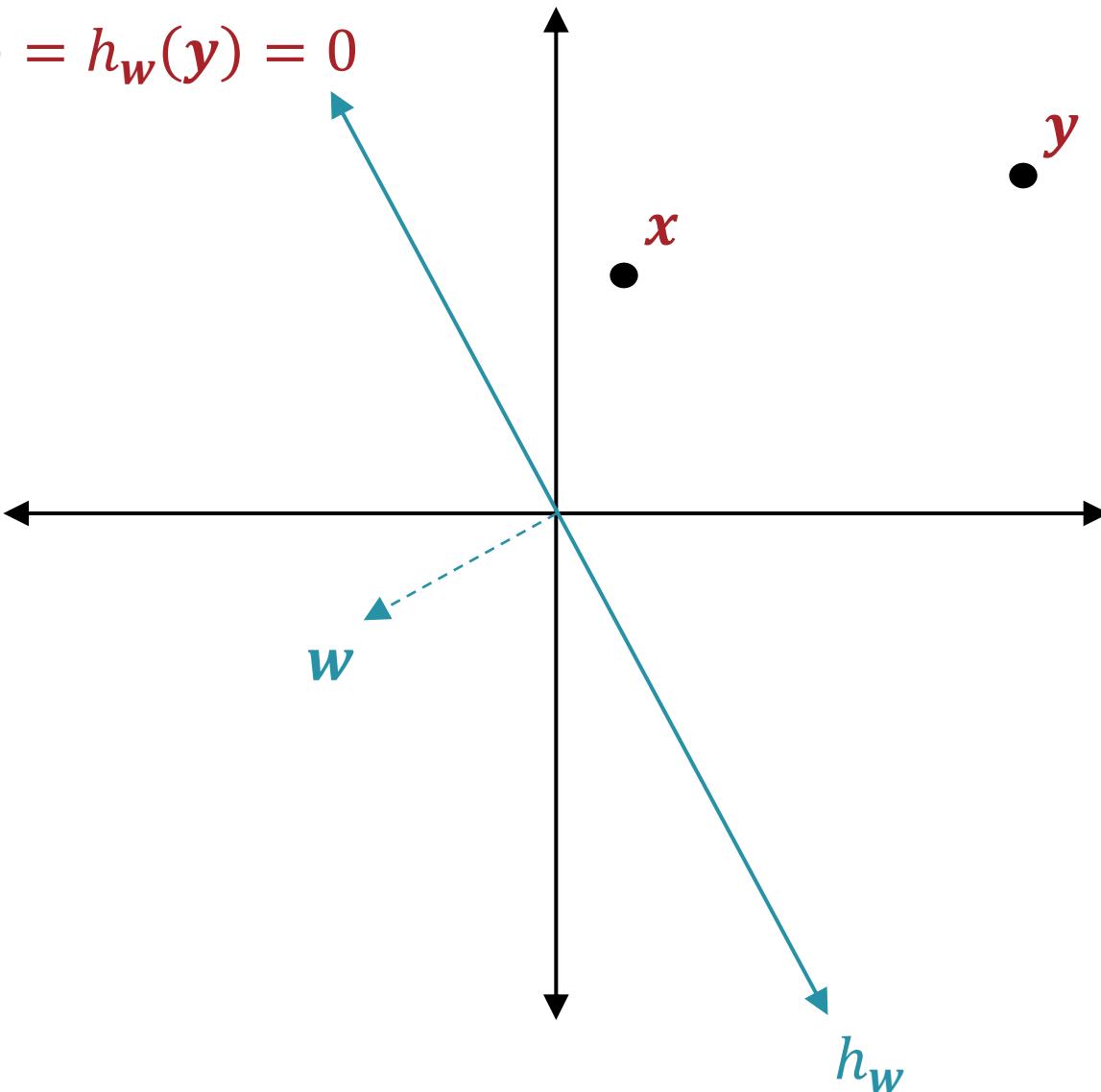
$$h_w(x) = h_w(y) = 1$$



Locality-Sensitive Hashing: Example

- Random projection:

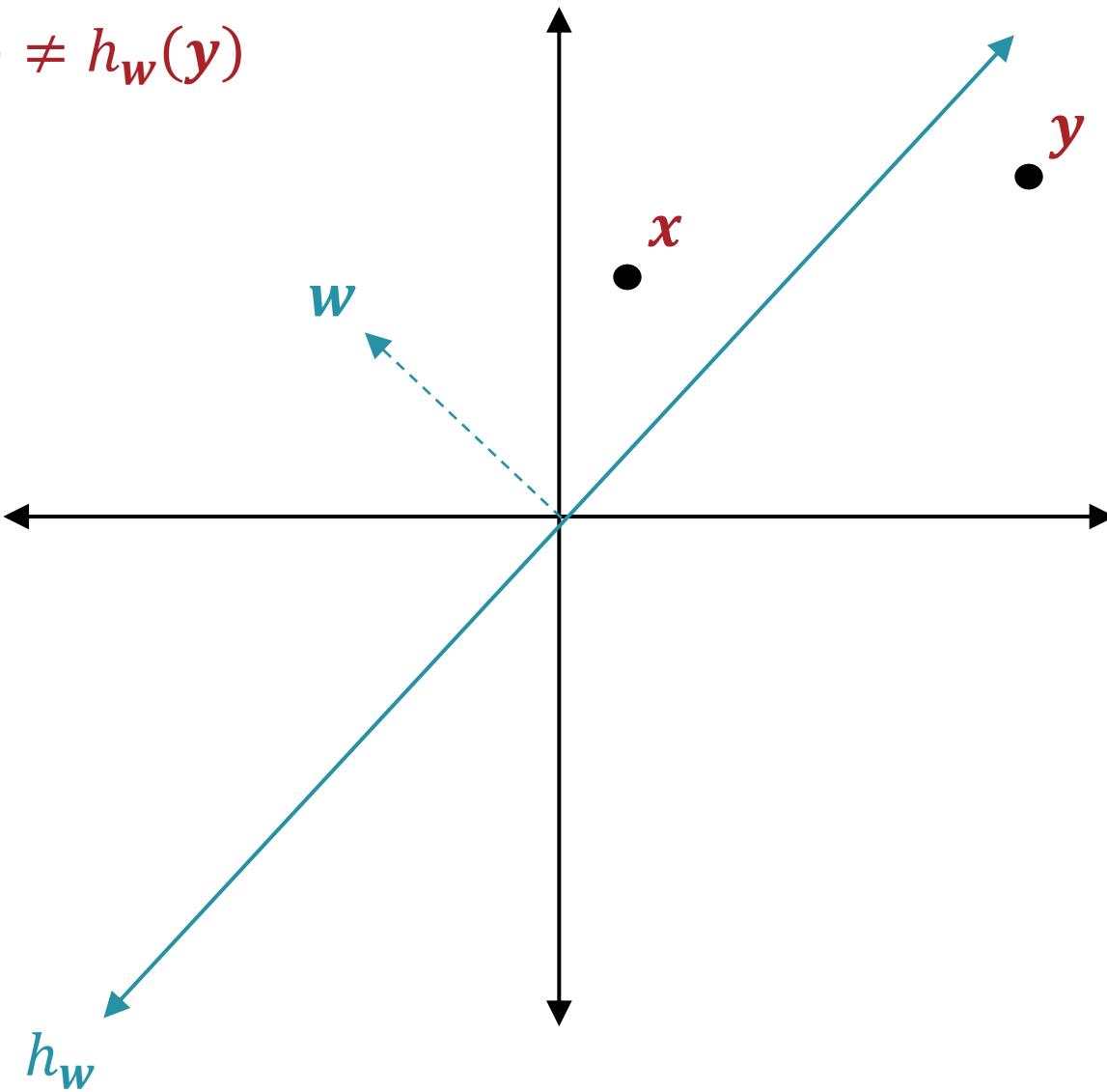
$$h_w(x) = h_w(y) = 0$$



Locality-Sensitive Hashing: Example

- Random projection:

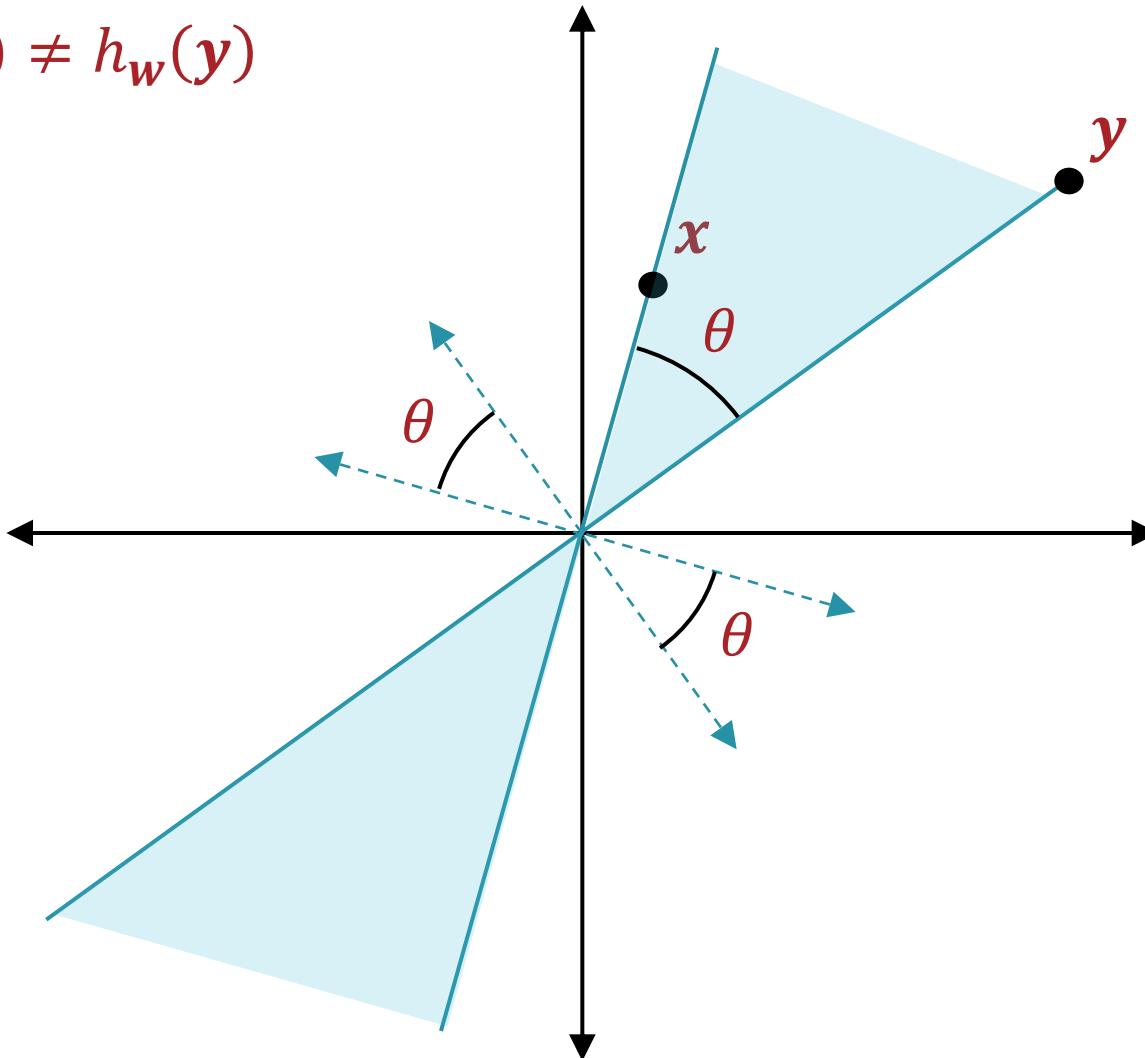
$$h_w(x) \neq h_w(y)$$



Locality-Sensitive Hashing: Example

- Random projection:

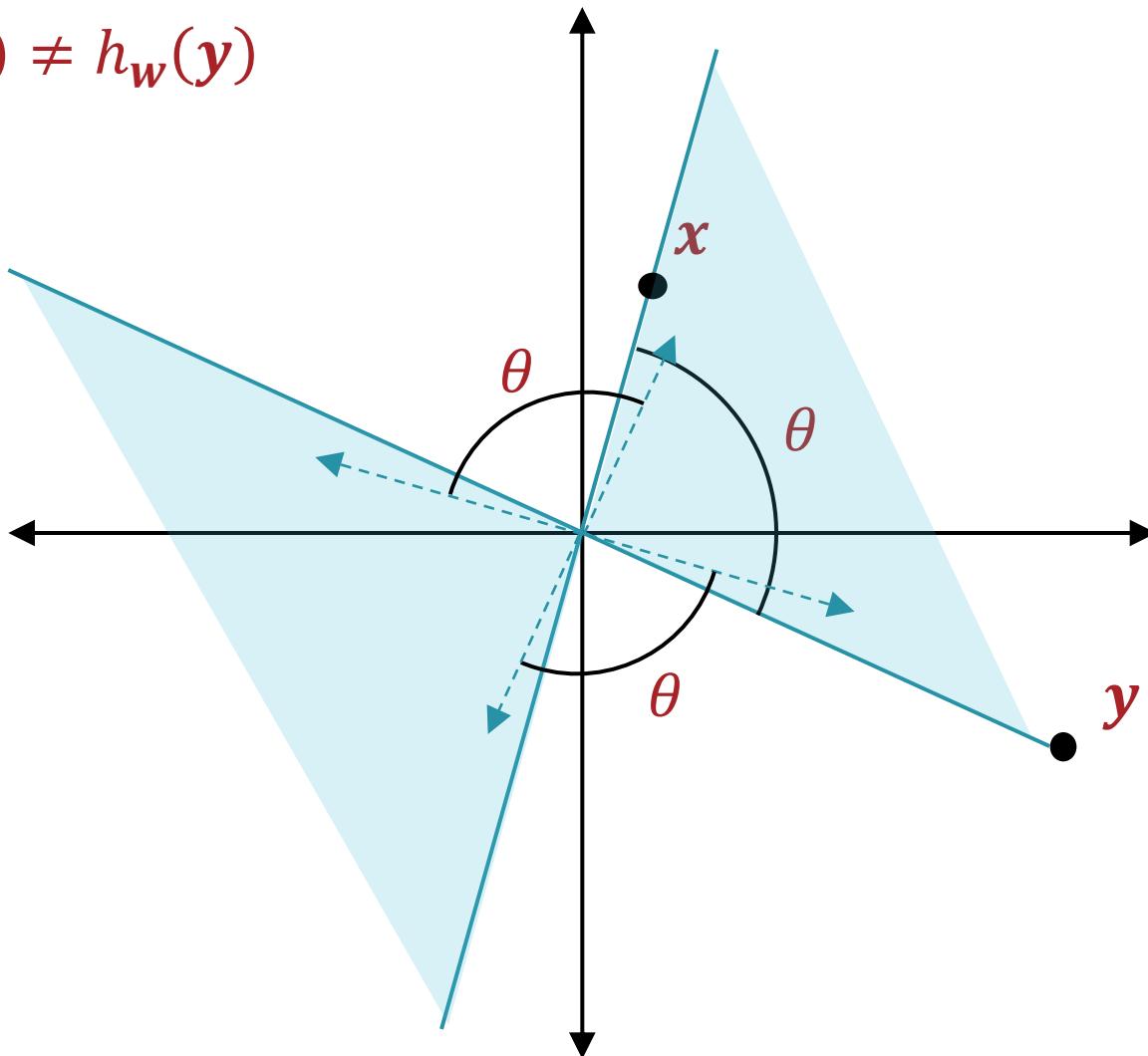
$$h_w(x) \neq h_w(y)$$



Locality-Sensitive Hashing: Example

- Random projection:

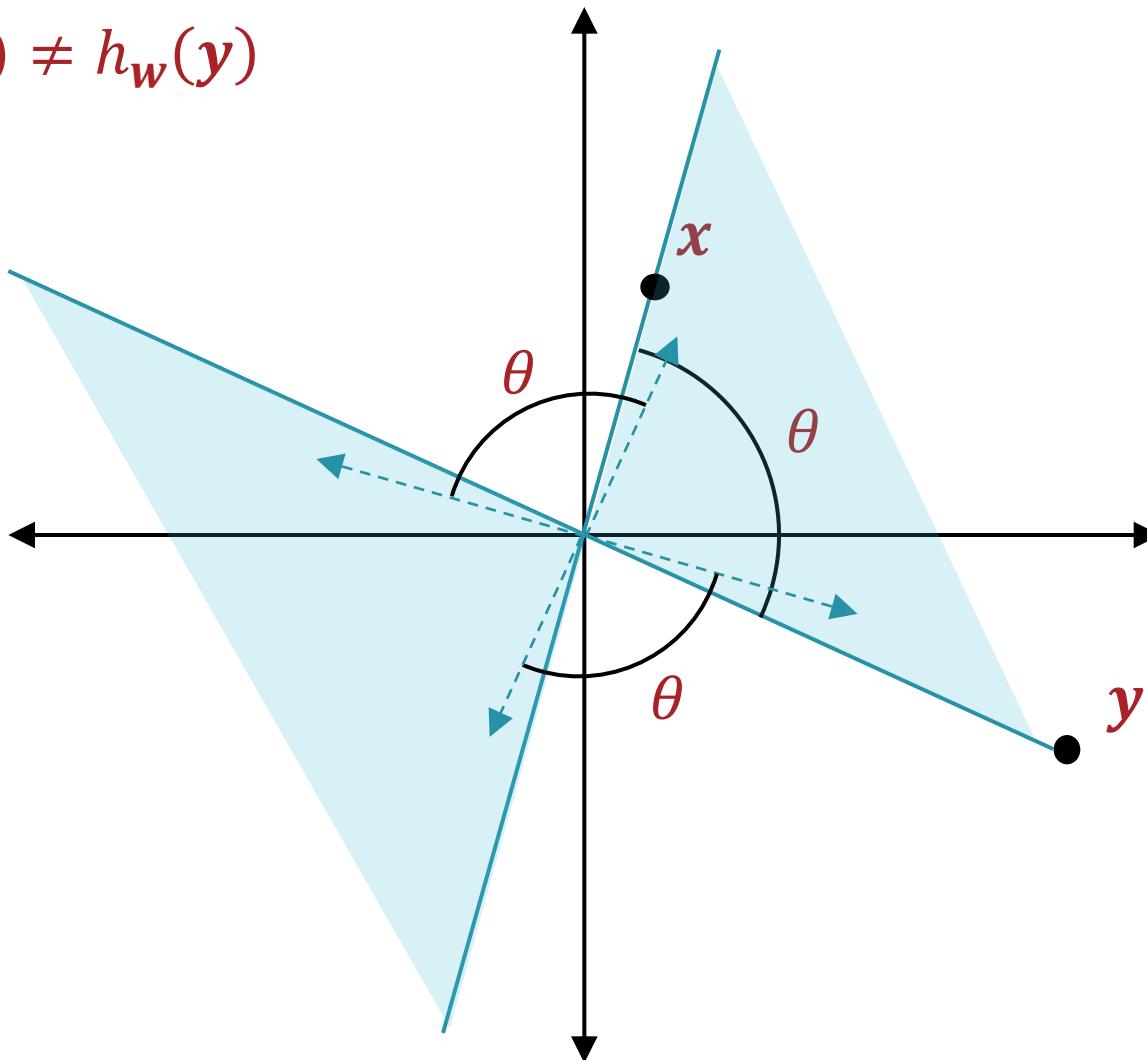
$$h_w(x) \neq h_w(y)$$



But why stop
at just one
weight vector?

- Random projection:

$$h_w(x) \neq h_w(y)$$



Locality-Sensitive Hashing: Amplification

- Random projections:
 - $m = 1$ so our hash functions map \mathbb{R}^k to $\{0,1\}$
 - Let \mathcal{F} be the set of all linear decision boundaries:

$$\mathcal{F} = \left\{ h : h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases} \right\}$$

- $d(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$, the *cosine distance*
- Sample r weight vectors $\mathbf{w}_1, \dots, \mathbf{w}_r$ and define a new *bit vector* representation for \mathbf{x} as

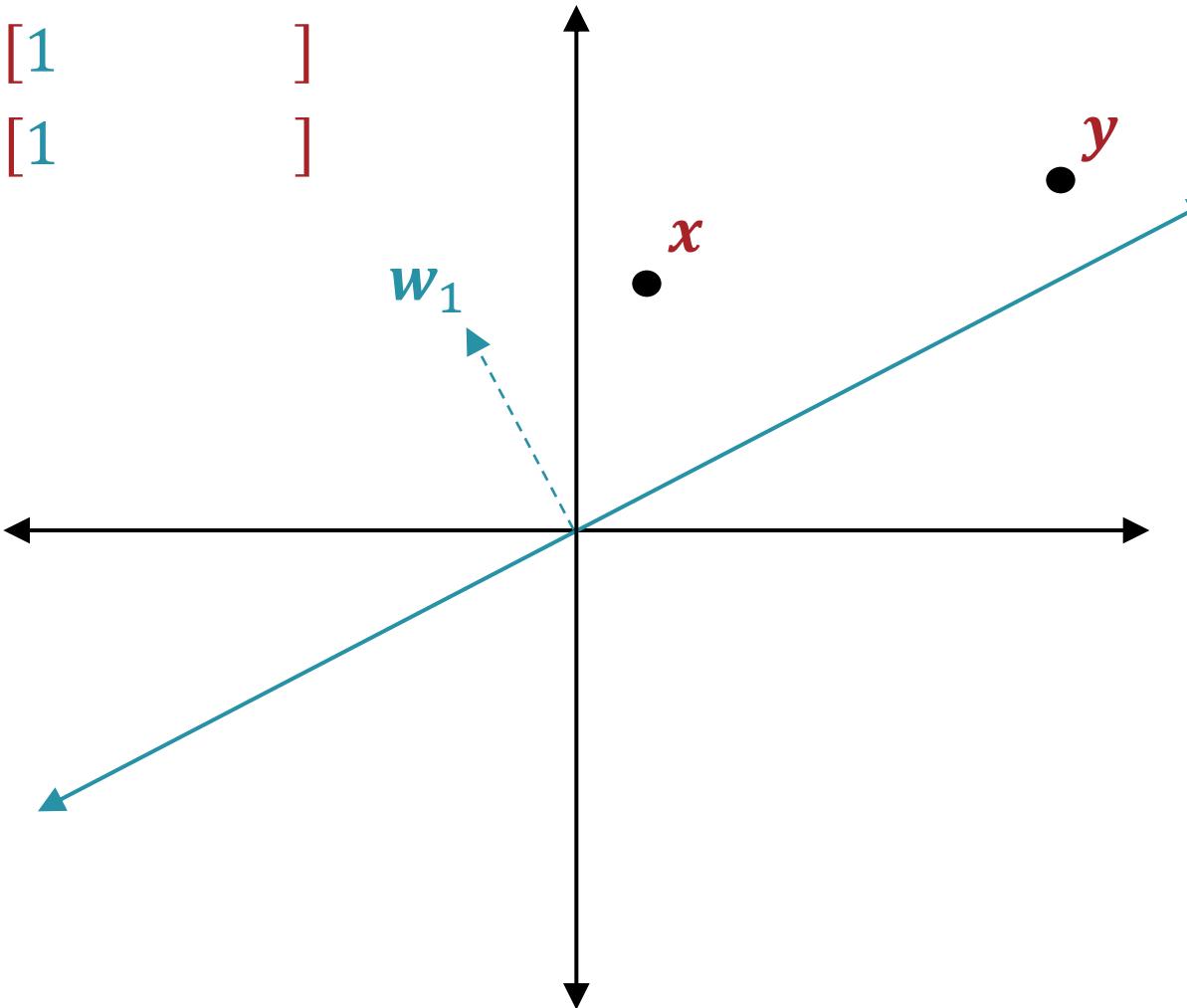
$$\mathbf{x}' = [h_{\mathbf{w}_1}(\mathbf{x}), h_{\mathbf{w}_2}(\mathbf{x}), \dots, h_{\mathbf{w}_r}(\mathbf{x})]$$

Locality-Sensitive Hashing: Amplification

- Random projections:

$$\mathbf{x}' = [1 \quad \quad \quad]$$

$$\mathbf{y}' = [1 \quad \quad \quad]$$

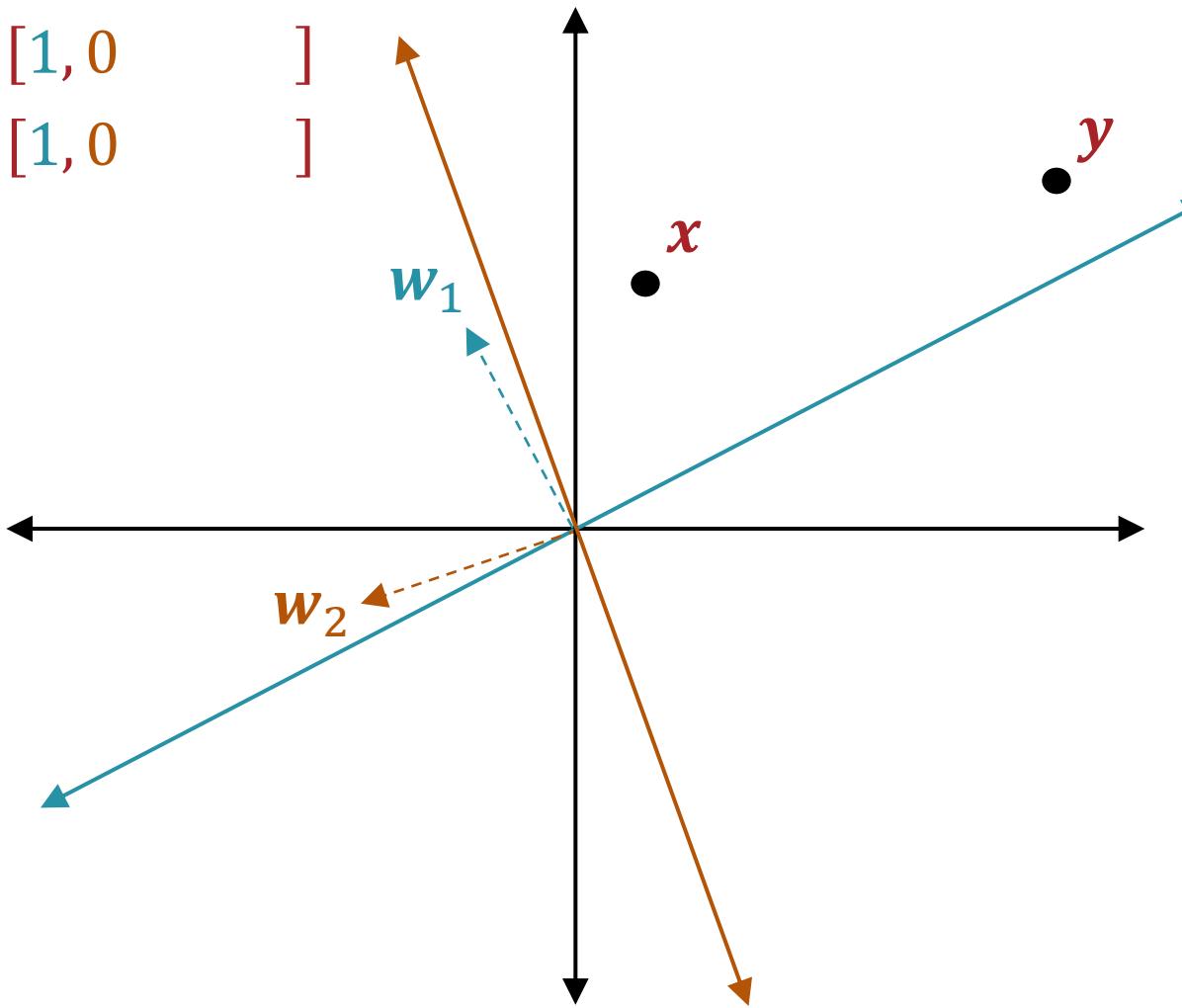


Locality-Sensitive Hashing: Amplification

- Random projections:

$$\mathbf{x}' = [1, 0]$$

$$\mathbf{y}' = [1, 0]$$

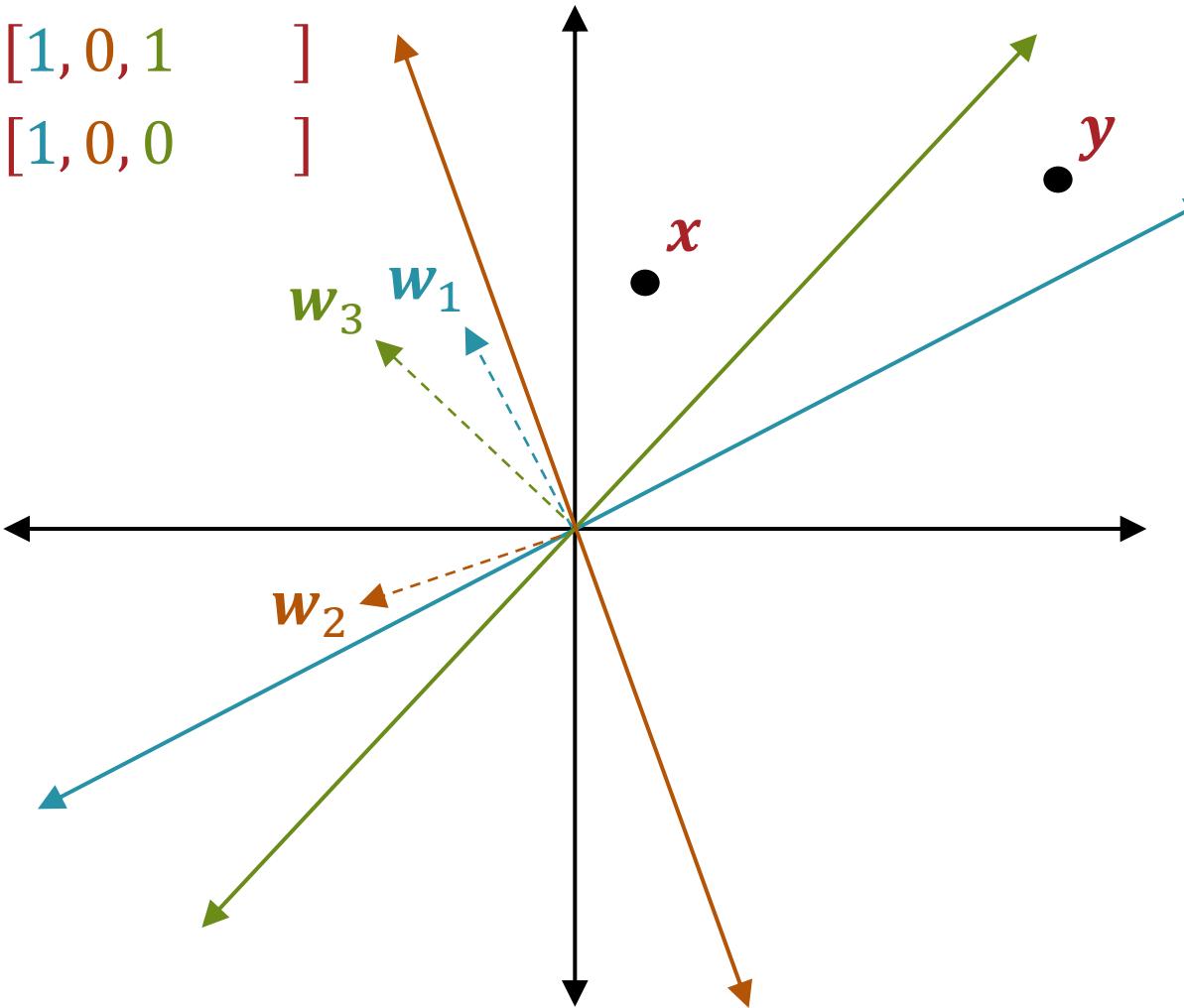


Locality-Sensitive Hashing: Amplification

- Random projections:

$$\mathbf{x}' = [1, 0, 1]$$

$$\mathbf{y}' = [1, 0, 0]$$

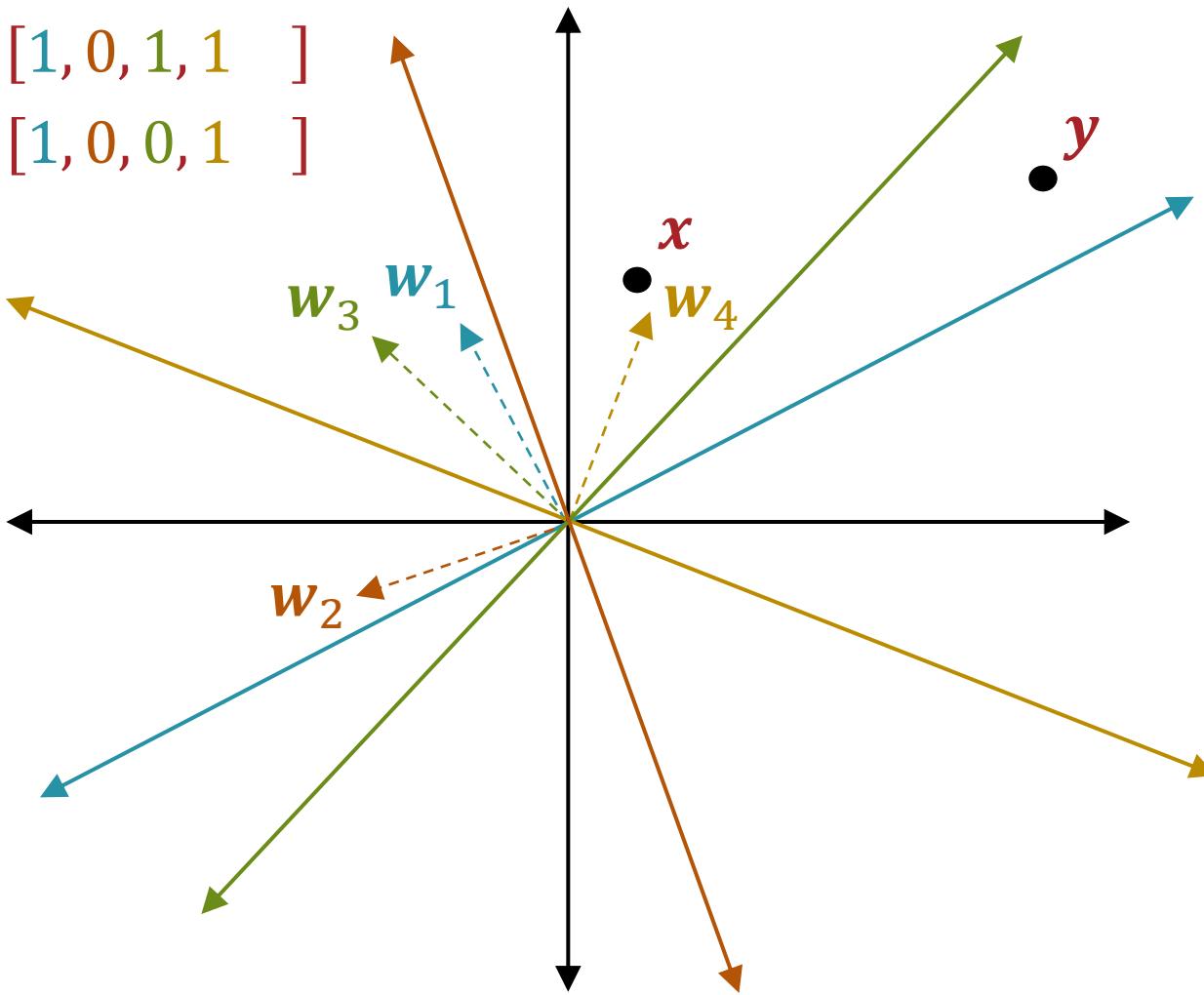


Locality-Sensitive Hashing: Amplification

- Random projections:

$$\mathbf{x}' = [1, 0, 1, 1]$$

$$\mathbf{y}' = [1, 0, 0, 1]$$

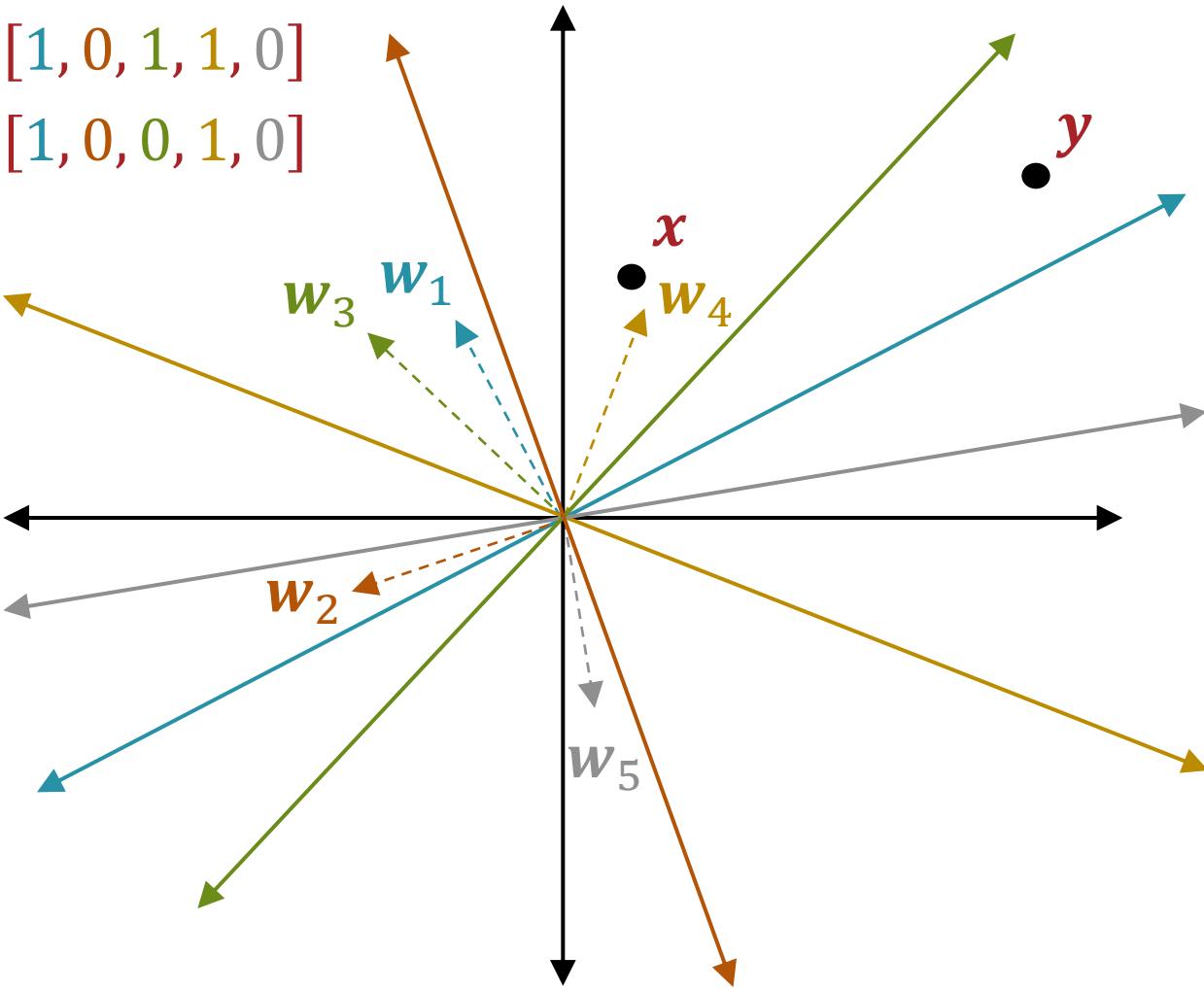


Locality-Sensitive Hashing: Amplification

- Random projections:

$$\mathbf{x}' = [1, 0, 1, 1, 0]$$

$$\mathbf{y}' = [1, 0, 0, 1, 0]$$



Locality-Sensitive Hashing: Amplification

- Random projections:
 - $m = 1$ so our hash functions map \mathbb{R}^k to $\{0,1\}$
 - Let \mathcal{F} be the set of all linear decision boundaries:

$$\mathcal{F} = \left\{ h : h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases} \right\}$$

- $d(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$, the *cosine distance*
- Sample r weight vectors $\mathbf{w}_1, \dots, \mathbf{w}_r$ and define a new *bit vector* representation for \mathbf{x} as

$$\mathbf{x}' = [h_{\mathbf{w}_1}(\mathbf{x}), h_{\mathbf{w}_2}(\mathbf{x}), \dots, h_{\mathbf{w}_r}(\mathbf{x})]$$

- The Hamming distance between two bit vectors is

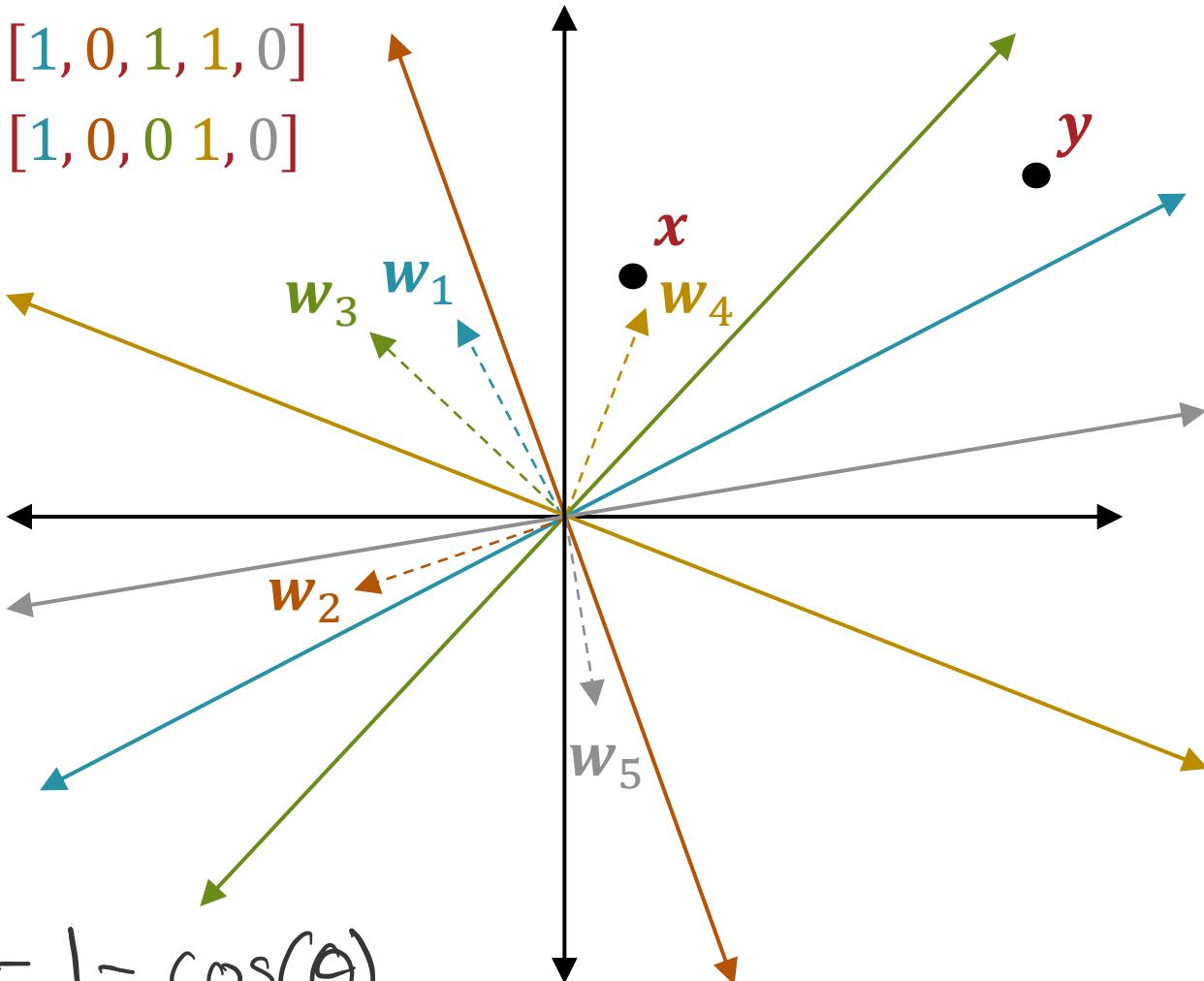
$$d_h(\mathbf{x}', \mathbf{y}') = \sum_{i=1}^r \mathbb{1}(x'_i \neq y'_i) = \|\mathbf{x}' - \mathbf{y}'\|_1$$

Locality-Sensitive Hashing: Amplification

- Random projections:

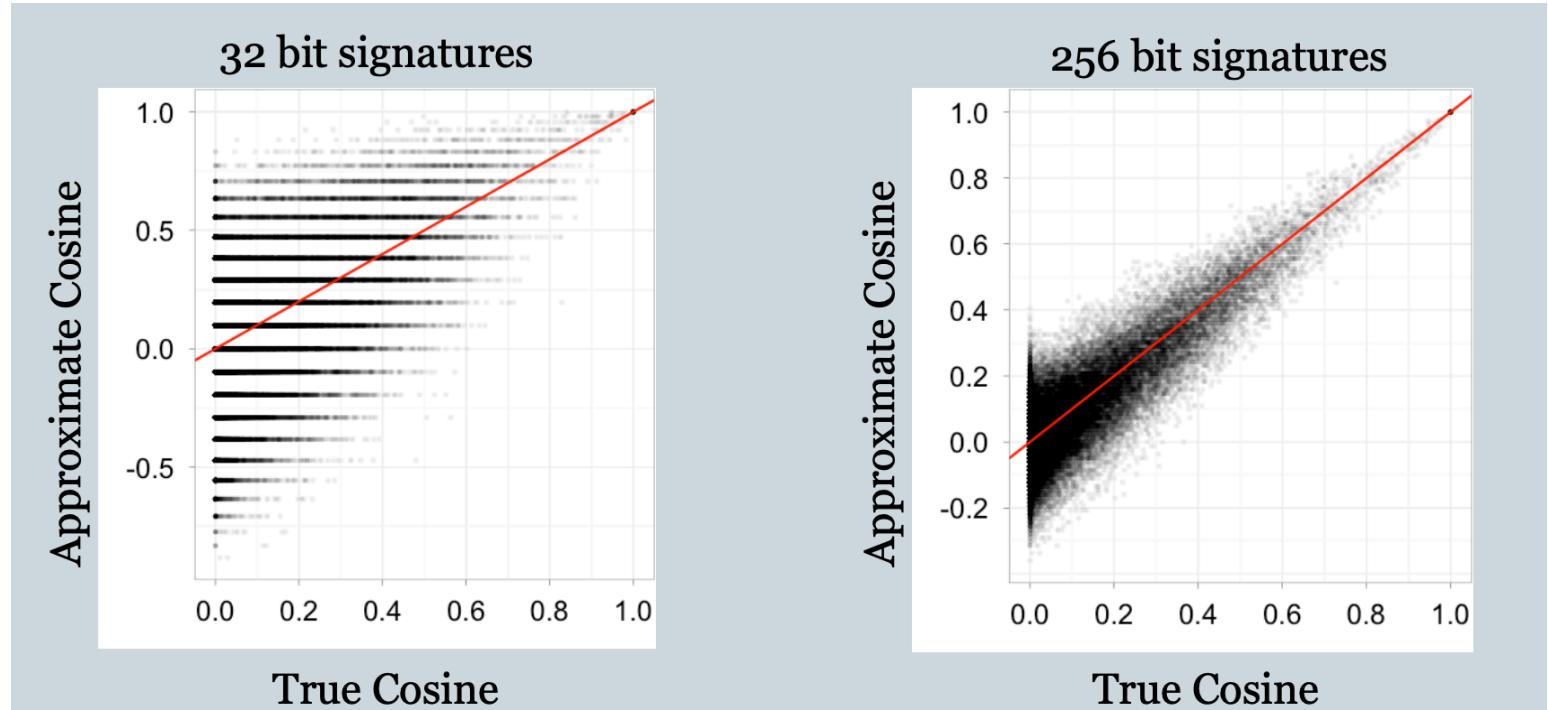
$$\mathbf{x}' = [1, 0, 1, 1, 0]$$

$$\mathbf{y}' = [1, 0, 0, 1, 0]$$



$$d(x, y) = 1 - \cos(\theta)$$
$$\approx 1 - \cos\left(\frac{d_h(x', y')}{r}\right)$$

Locality-Sensitive Hashing: Evaluation



Cheaper, less accurate approximation

Expensive, more accurate approximation

Locality-Sensitive Hashing: Summary

- Key idea: use hash functions that map “similar” items to the same function
 - Plot twist: collisions are the goal!
- Random projections can be used to efficiently approximate the cosine distance between points
 - Reduces to computing the Hamming distance between bit vectors consisting of multiple hash functions
- Can be used to efficiently cluster data and compute the nearest neighbors to a query point

Distributed k -NN

1. Distribute training dataset across some number of worker machines
2. Communicate query point \mathbf{x} to each worker machine
3. Map: compute distance between \mathbf{x} and each training data point
 - If the data is high-dimensional, first compute random projection bit vectors to approximate distances
4. Reduce: return the k nearest neighbors from each worker (and their distances)
5. Driver computes the global k nearest neighbors

Key Takeaways

- Non-numeric data can be a challenge for many machine learning models
- One-hot encodings are impractical for large datasets
 - Feature hashing is a reasonable alternative
 - Hash kernels are unbiased estimators of dot products that preserve distances (with high probability)
- For algorithms where updates are weighted counts of feature values, we can use a weighted version of count-min sketch, an application of hashing with nice theoretical guarantees and good empirical performance
- For algorithms where distances between points is relevant, locality sensitive hashing can be used to efficiently approximate certain distance metrics

Looking
unreasonably
far ahead...

- Let's talk about HW4!
- Main goals:
 - Give you all hands-on experience working with cloud computing, specifically Amazon Web Services
 - Work with an actual large dataset in Spark (the entire Million Song Dataset ~280 GB)
 - Execute an end-to-end machine learning pipeline in a distributed manner
- Warning: this homework will be more open-ended than previous ones

Looking
unreasonably
far ahead...

- Let's talk about HW4!
- Main goals:
 - Give you all hands-on experience working with **cloud computing**, specifically **Amazon Web Services**
 - Work with an actual large dataset in Spark (the entire Million Song Dataset ~280 GB)
 - Execute an end-to-end machine learning pipeline in a distributed manner
- Warning: this homework will be more open-ended than previous ones

Recall: Cloud Computing

- Enables distributed processing by democratizing access to storage and computational resources
- Similar to a public utility, you can access as much or as little of it as you need



Recall: Cloud Computing

- Cloud computing relies on sharing of resources to achieve coherence and typically using a "pay-as-you-go" model which can help in reducing capital expenses but may also lead to unexpected operating expenses for users.



Data Centers: Amazon Data Center in Ashburn, Virginia



Data Centers: Facebook Data Center in Prineville, Oregon



Data Centers: Amazon Data Center in Brétigny-sur- Orge, France



Cloud Computing: Pros and Cons

- Pros:
 - On-demand computing/storage allows for scalability and elasticity (only pay for what you use)
 - Eliminates infrastructure and **maintenance** costs
- Cons:

Cloud Computing: Maintenance

Florida Data Centers Brace for Powerful Hurricane Ian

BY RICH MILLER - SEPTEMBER 28, 2022 — LEAVE A COMMENT



Data Center Fire: Google Suffers 'Electrical Incident,' 3 Injured

News of yet another data center fire in just two years: OVH in 2021 and now Google. Here's how to make sure you're not next.

Apple Recovers From Massive Outage

TECH • FACEBOOK

Yet another example
rely heavily on the cl

Facebook's outage cost the company nearly \$100 million in revenue

BY CHRIS MORRIS

Jeff Baumgartner | Mar 22, 2021 October 4, 2021 4:30 PM EDT
Updated October 5, 2021 11:25 AM EDT

Source: <https://www.datacenterknowledge.com/google-alphabet/data-center-fire-google-suffers-electrical-incident-3-injured>

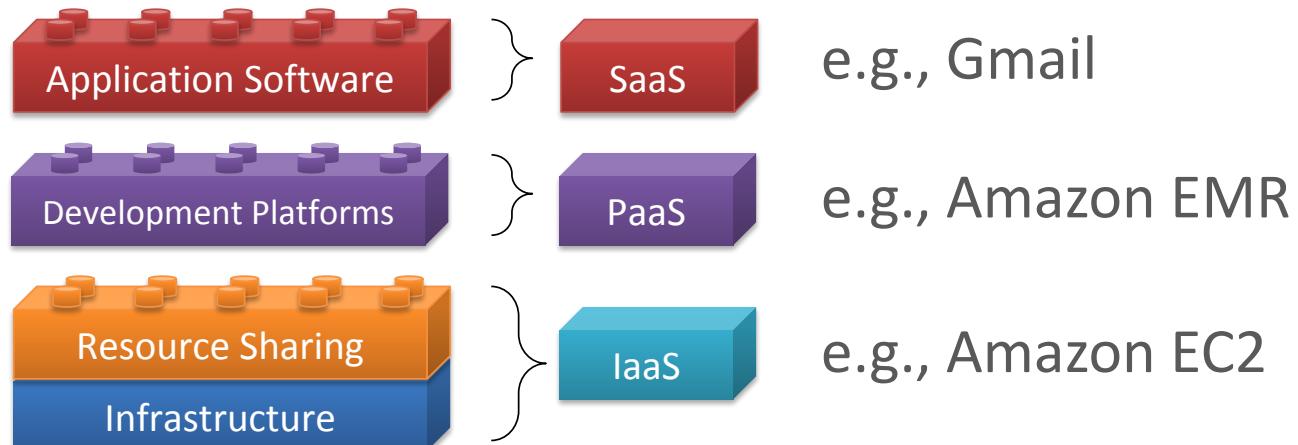
Source: <https://www.yahoo.com/now/facebook-outage-already-cost-company-203040137.html>

Cloud Computing: Pros and Cons

- Pros:
 - On-demand computing/storage allows for scalability and elasticity (only pay for what you use)
 - Eliminates infrastructure and **maintenance** costs
- Cons:
 - Depending on usage, overall (monetary) costs may be higher than being “on-perm”, i.e., buying your own compute resources
 - Data gravity – difficult to switch providers once a lot of resources have been committed to a single one
 - Data privacy – do you really trust Amazon?
 - Environmental impacts – densely packed computing resources require a lot of energy to run and **cool**

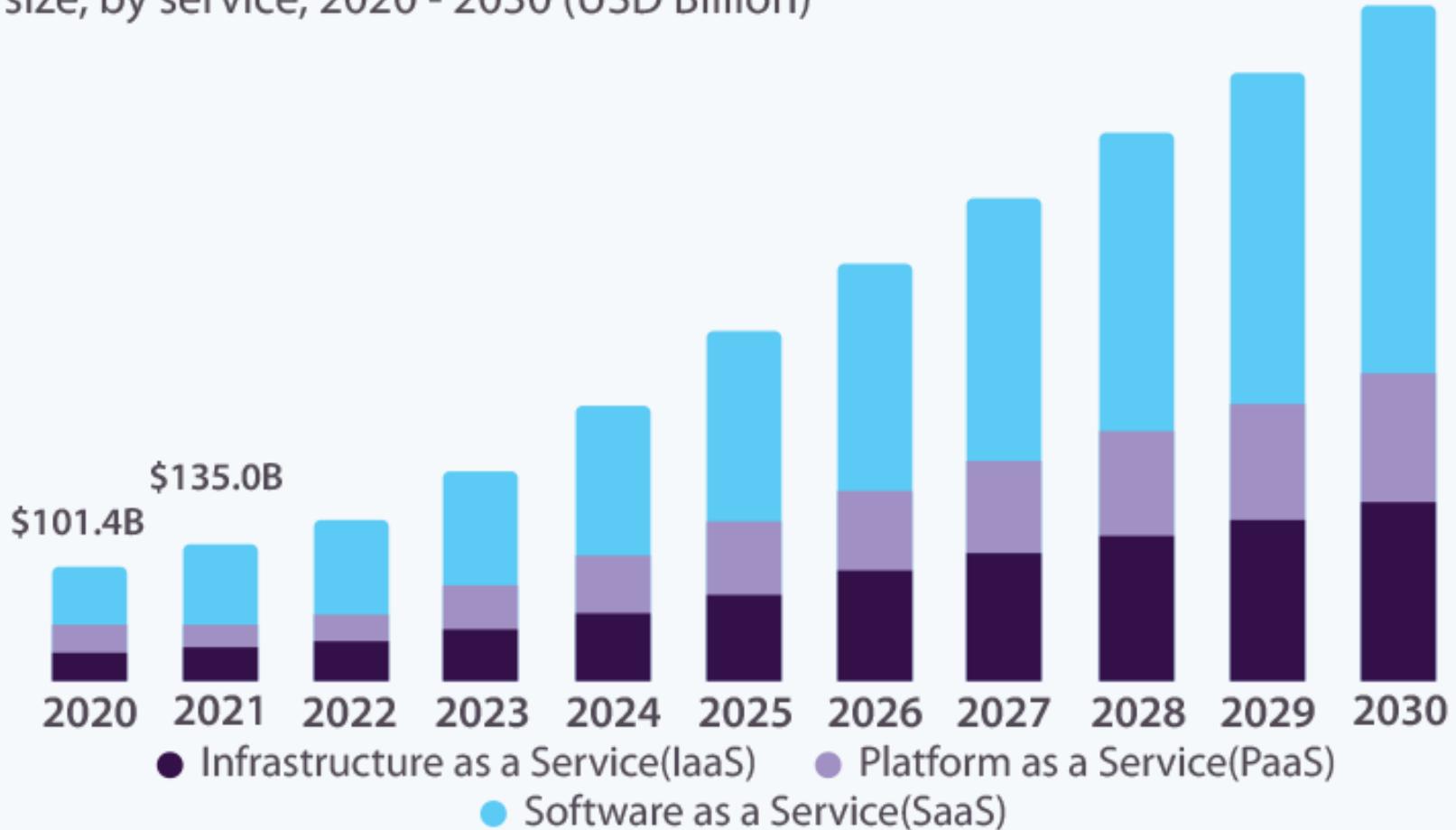
Cloud Computing: Services

- Cloud computing services can be thought of as different levels of abstraction:
 - Software as a service (SaaS)
 - Platform as a service (PaaS)
 - Infrastructure as a service (IaaS)



Cloud Computing: Market

U.S. Cloud Computing Market size, by service, 2020 - 2030 (USD Billion)



Cloud Computing: Market

