

# DATA SCIENCE WITH MACHINE LEARNING: RETRIEVAL

This lecture is  
based on course by E. Fox and C. Guestrin, Univ of Washington

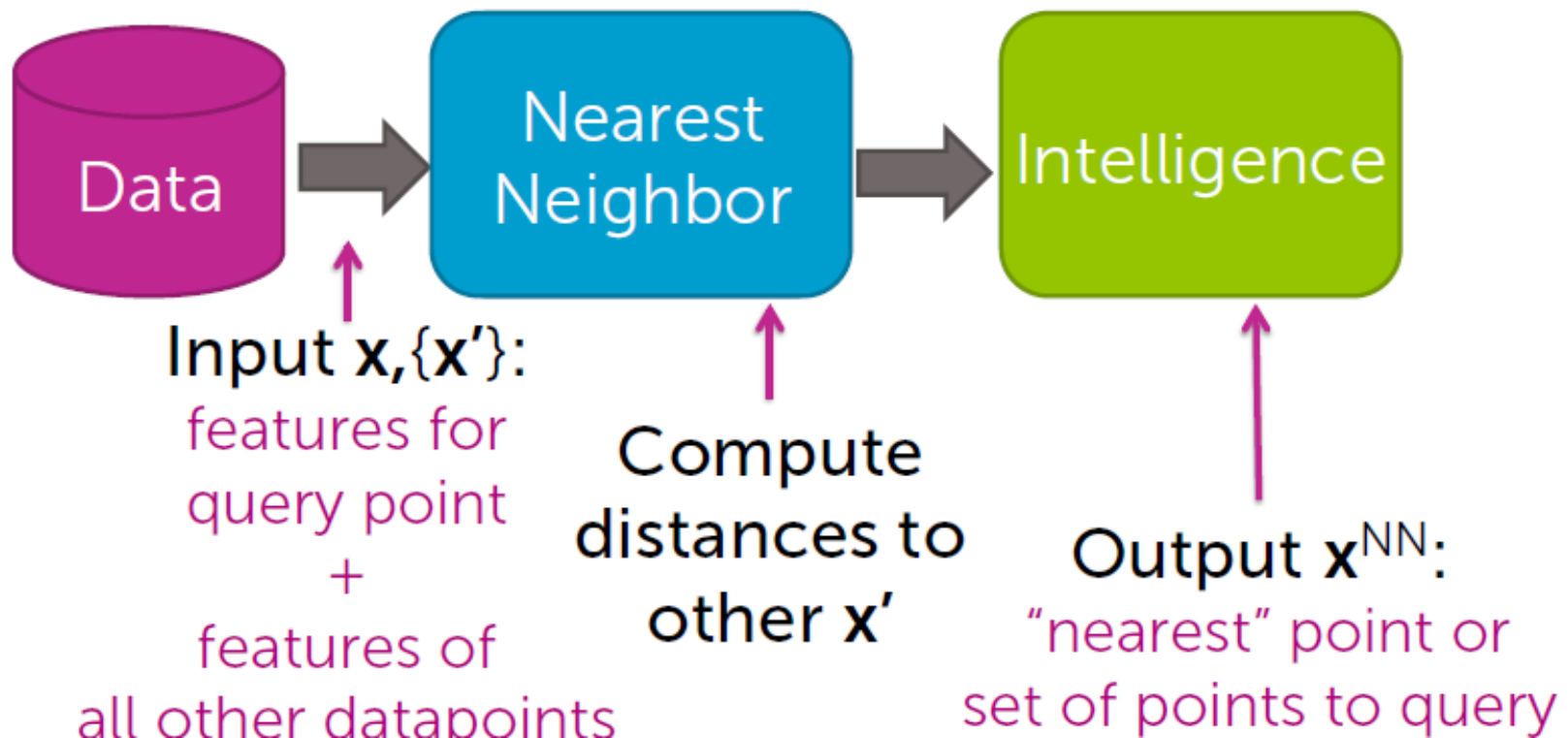
25/01/2022

WFAiS UJ, Informatyka Stosowana  
I stopień studiów

# What is retrieval?

2

Search for related items



# What is retrieval?

3

## Retrieve “nearest neighbor” article

Space of all articles,  
organized by similarity of text

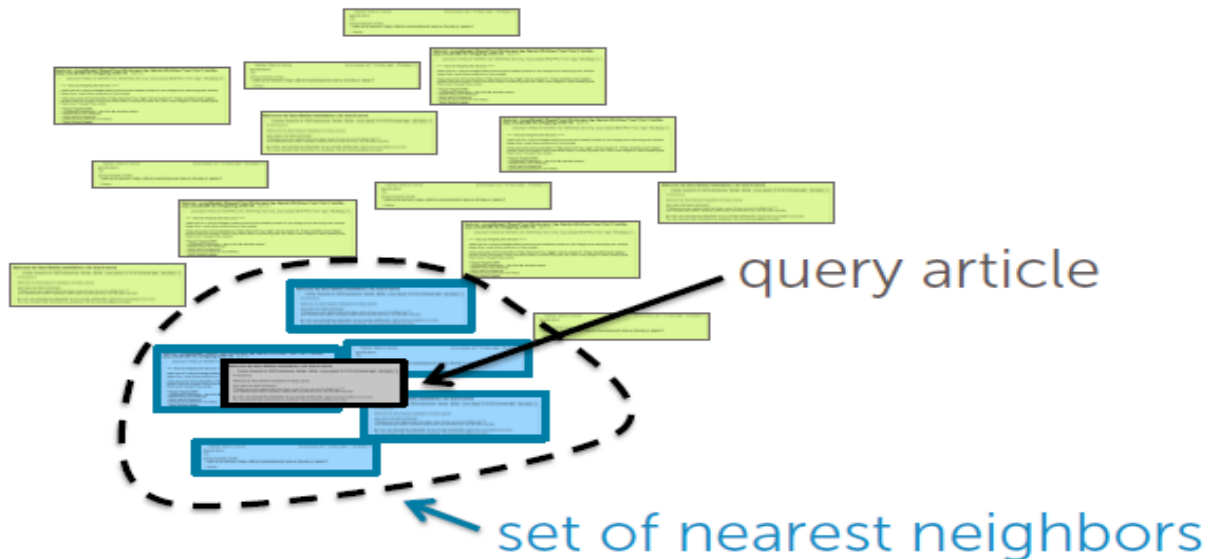


# What is retrieval?

4

## Or set of nearest neighbors

Space of all articles,  
organized by similarity of text



# Retrieval applications

5

Just about everything...

Images



Products



Streaming content:

- Songs
- Movies
- TV shows
- ...

News articles



Social networks

(people you might want to connect with)

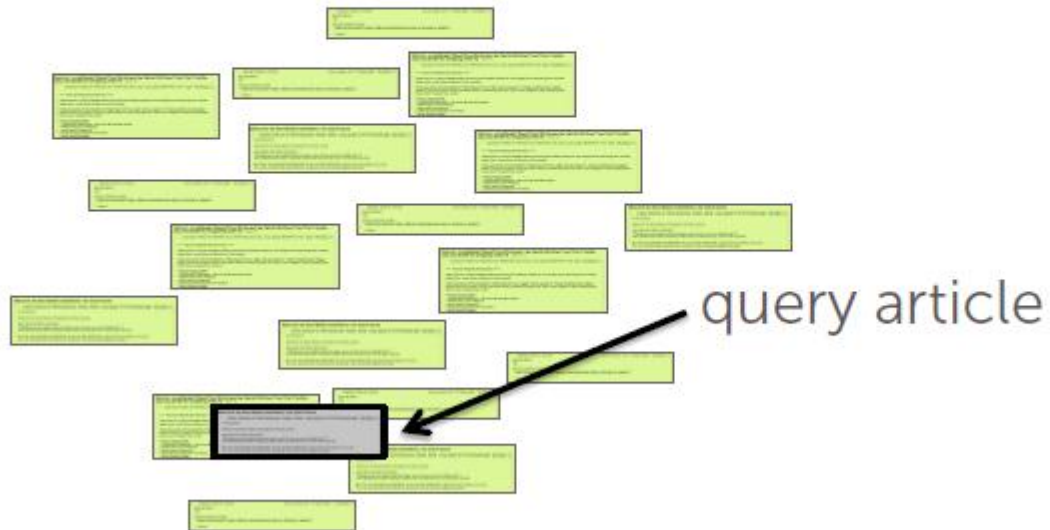


# Retrieval as k-nearest neighbor search

# 1-NN search for retrieval

7

Space of all articles,  
organized by similarity of text

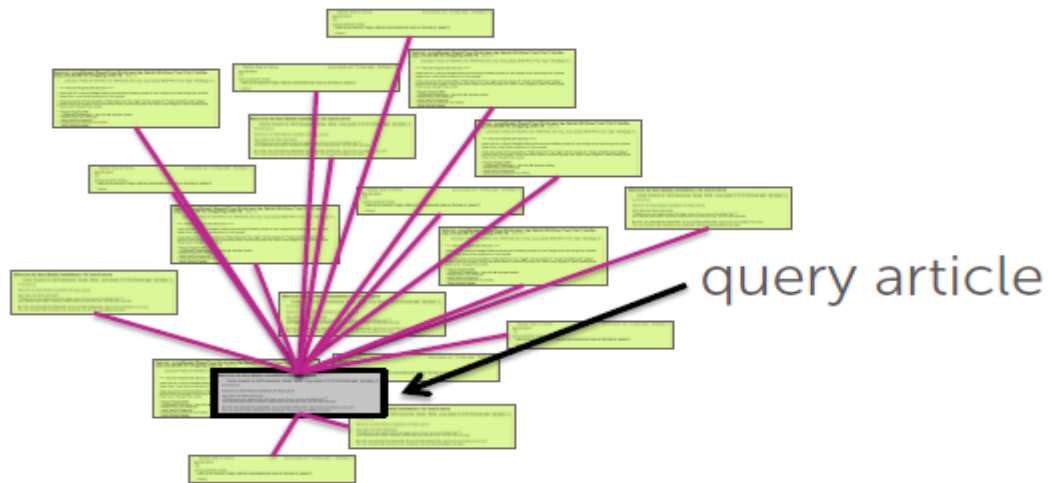


# 1-NN search for retrieval

8

## Compute distances to all docs

Space of all articles,  
organized by similarity of text



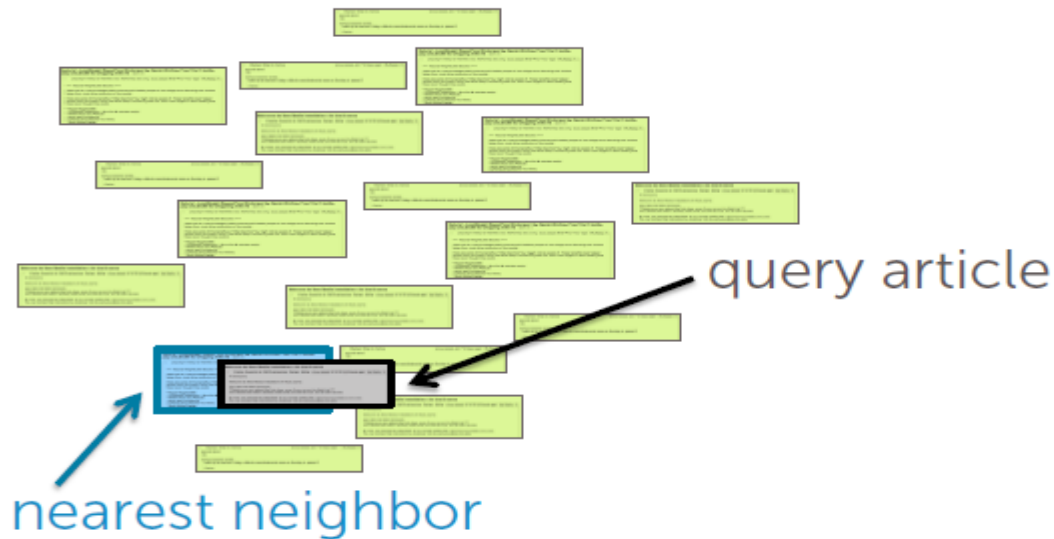


# 1-NN search for retrieval

9

## Retrieve “nearest neighbor”

Space of all articles,  
organized by similarity of text

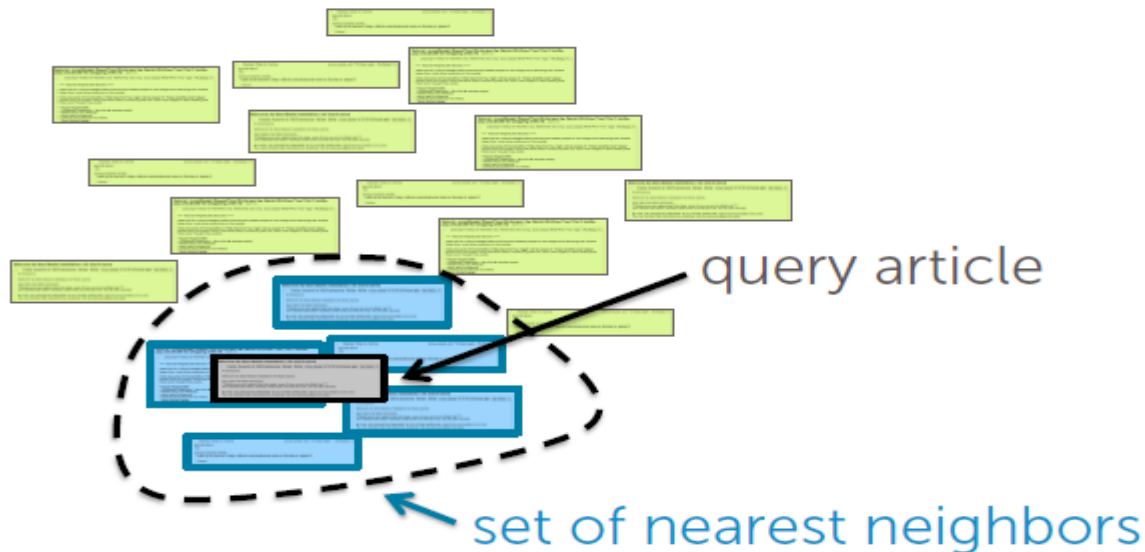


# 1-NN search for retrieval

10

## Or set of nearest neighbors

Space of all articles,  
organized by similarity of text



# 1-NN algorithm

11

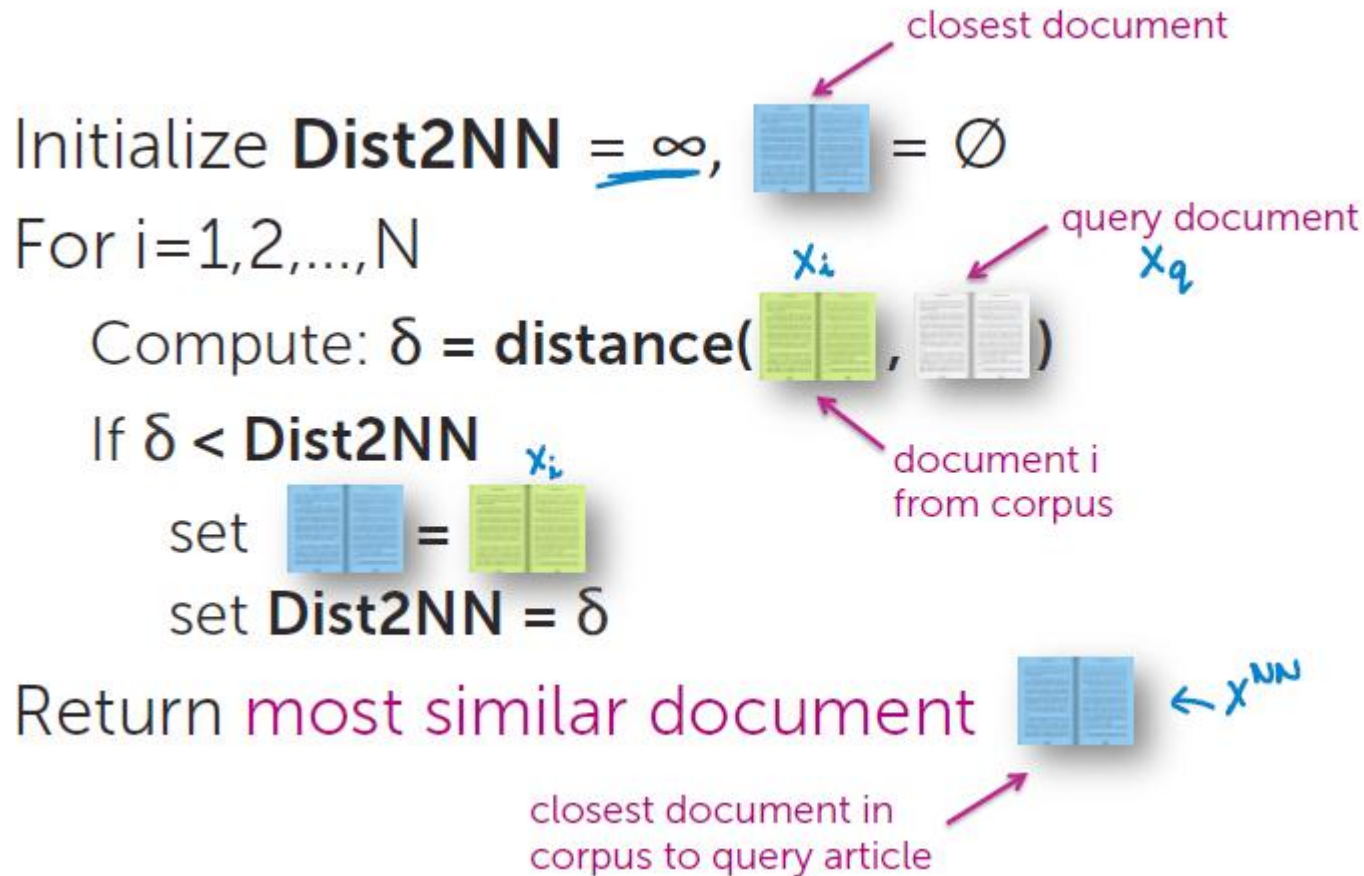
## 1 – Nearest neighbor

- **Input:** Query article  :  $\mathbf{x}_q$   
Corpus of documents (N docs)  
 :  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
- **Output:** *Most* similar article   $\leftarrow \mathbf{x}^{NN}$

Formally: 
$$\mathbf{x}^{NN} = \min_{x_i} \text{distance}(\mathbf{x}_q, \mathbf{x}_i)$$

# 1-NN algorithm

12



# k-NN algorithm

13

- **Input:** Query article  :  $\mathbf{x}_q$   
Corpus of documents  
 :  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
- **Output:** *List of k* similar articles



Formally:

$$X^{NN} = \{x^{NN_1}, \dots, x^{NN_k}\}$$

For all  $x_i$  not in  $X^{NN}$ ,  $\text{distance}(x_i, x_q) \geq \max_{x^{NN_j}, j=1 \dots k} \text{distance}(x^{NN_j}, x_q)$

# k-NN algorithm

14

Initialize **Dist2kNN** =  $\text{sort}(\delta_1, \dots, \delta_k)$  ← list of sorted distances  
 =  $\text{sort}(\dots, \text{distance}(\text{doc}_1, \text{query doc}), \dots, \text{distance}(\text{doc}_k, \text{query doc}))$  ← list of sorted docs

For  $i=k+1, \dots, N$

Compute:  $\delta = \text{distance}(\text{doc}_i, \text{query doc})$

If  $\delta < \text{Dist2kNN}[k]$  ← distance to  $k^{\text{th}}$  NN (furthest NN in set)

find  $j$  such that  $\delta > \text{Dist2kNN}[j-1]$  but  $\delta < \text{Dist2kNN}[j]$

remove furthest house and shift queue:

$\text{Dist2kNN}[1:k] = \text{Dist2kNN}[1:j-1]$

inserting new article

$\text{Dist2kNN}[j+1:k] = \text{Dist2kNN}[j:k-1]$

set  $\text{Dist2kNN}[j] = \delta$  and  $\text{Doc2kNN}[j] = \text{doc}_i$

closest k docs to query doc

Return  $k$  most similar articles



# Critical elements of NN search

15

Item (e.g., doc) representation

$\mathbf{x}_q \leftarrow$



Measure of distance between items:

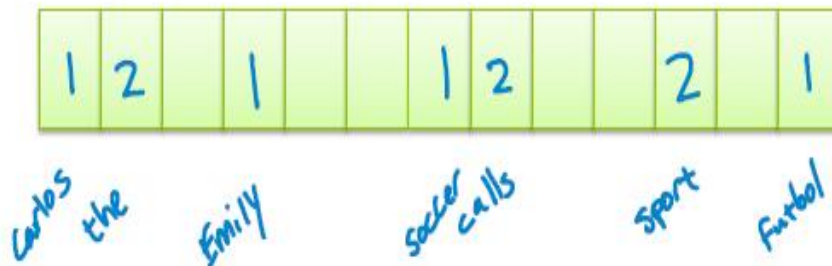
$$\delta = \text{distance}(\mathbf{x}_i, \mathbf{x}_q)$$

# Document representation

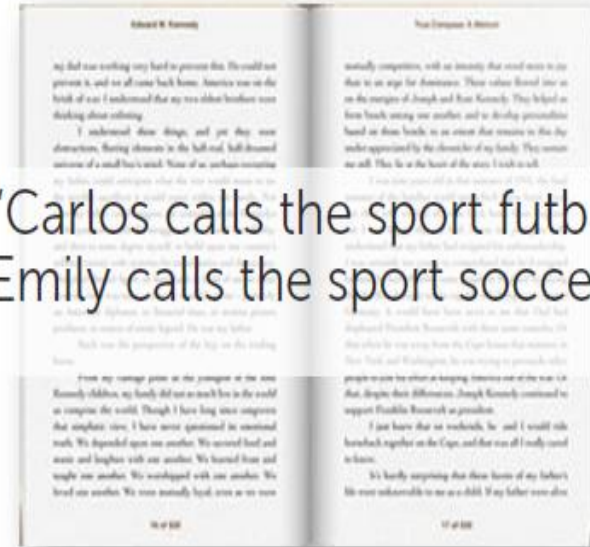
16

## Bag of words model

- Ignore order of words
- Count # of instances of each word in vocabulary



“Carlos calls the sport futbol.  
Emily calls the sport soccer.”

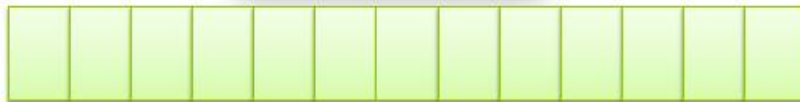




# Document representation

17

## Issues with word counts – Rare words



Common words in doc: "the", "player", "field", "goal"

Dominate rare words like: "futbol", "Messi"

# Document representation

18


## TF-IDF document representation

Emphasizes **important words**

- Appears frequently in document (**common locally**)

Term frequency = 

- Appears rarely in corpus (**rare globally**)

Inverse doc freq. = 



# Document representation

19

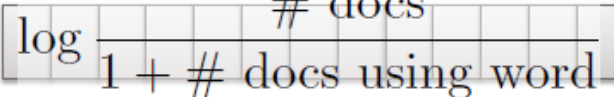
## TF-IDF document representation

Emphasizes important words

- Appears frequently in document (common locally)

Term frequency = 

- Appears rarely in corpus (rare globally)

Inverse doc freq. =  $\log \frac{\# \text{ docs}}{1 + \# \text{ docs using word}}$  

Trade off: local frequency vs. global rarity



tf \* idf

# Distance metrics:

20

## Distance metrics: Defining notion of “closest”

In 1D, just Euclidean distance:

$$\text{distance}(x_i, x_q) = |x_i - x_q|$$

In multiple dimensions:

- can define many interesting distance functions
- most straightforwardly, might want to weight different dimensions differently

# Distance metrics:

21

## Weighting different features

Reasons:

- Some features are more relevant than others



**# bedrooms**  
**# bathrooms**  
**sq.ft. living**  
sq.ft. lot  
floors  
**year built**  
year renovated  
waterfront



# Distance metrics:

## Weighting different features

Reasons:

- Some features are more relevant than others



**title**  
**abstract**  
main body  
**conclusion**



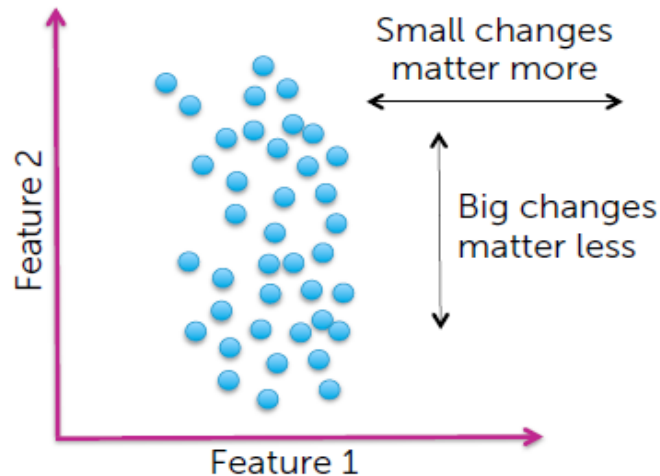
# Distance metrics:

23

## Weighting different features

Reasons:

- Some features are more relevant than others
- Some features vary more than others



Specify weights  
as a function of  
feature spread

For feature  $j$ :

$$\frac{1}{\max_i(\mathbf{x}_i[j]) - \min_i(\mathbf{x}_i[j])}$$

# Distance metrics:

24

## Scaled Euclidean distance

Formally, this is achieved via

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{a_1(\mathbf{x}_i[1] - \mathbf{x}_q[1])^2 + \dots + a_d(\mathbf{x}_i[d] - \mathbf{x}_q[d])^2}$$

weight on each feature  
(defining relative importance)



# Distance metrics:

25

## Effect of binary weights

distance( $\mathbf{x}_i, \mathbf{x}_q$ ) =

$$\sqrt{a_1(\mathbf{x}_i[1] - \mathbf{x}_q[1])^2 + \dots + a_d(\mathbf{x}_i[d] - \mathbf{x}_q[d])^2}$$

Setting weights as 0 or 1  
is equivalent to  
**feature selection**

Feature engineering/  
selection is  
**important, but hard**

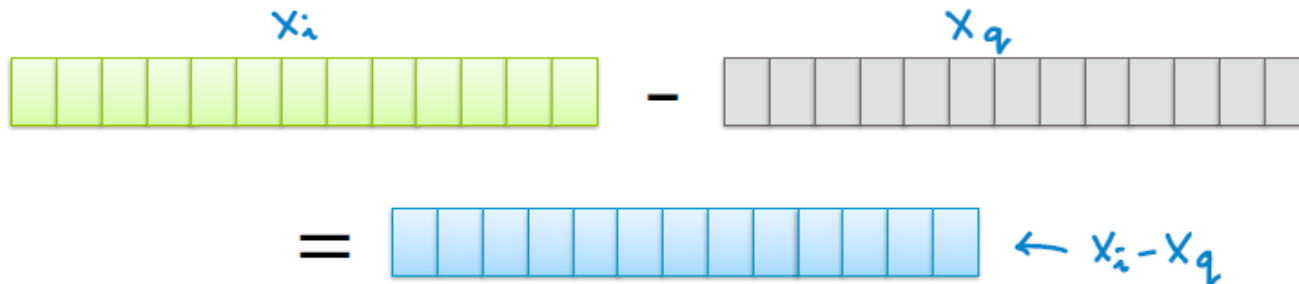
# Distance metrics:

26

## (non-scaled) Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T (\mathbf{x}_i - \mathbf{x}_q)}$$
$$\sqrt{(\mathbf{x}_i[1] - \mathbf{x}_q[1])^2 + \dots + (\mathbf{x}_i[d] - \mathbf{x}_q[d])^2}$$



# Distance metrics:

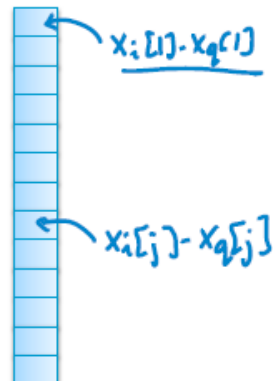
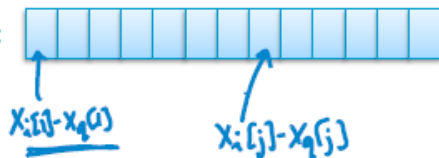
27

## (non-scaled) Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T (\mathbf{x}_i - \mathbf{x}_q)} \leftarrow$$
$$\sqrt{(x_i[1] - x_q[1])^2 + \dots + (x_i[d] - x_q[d])^2}$$

distance<sup>2</sup> =



take  
sq.r.t.

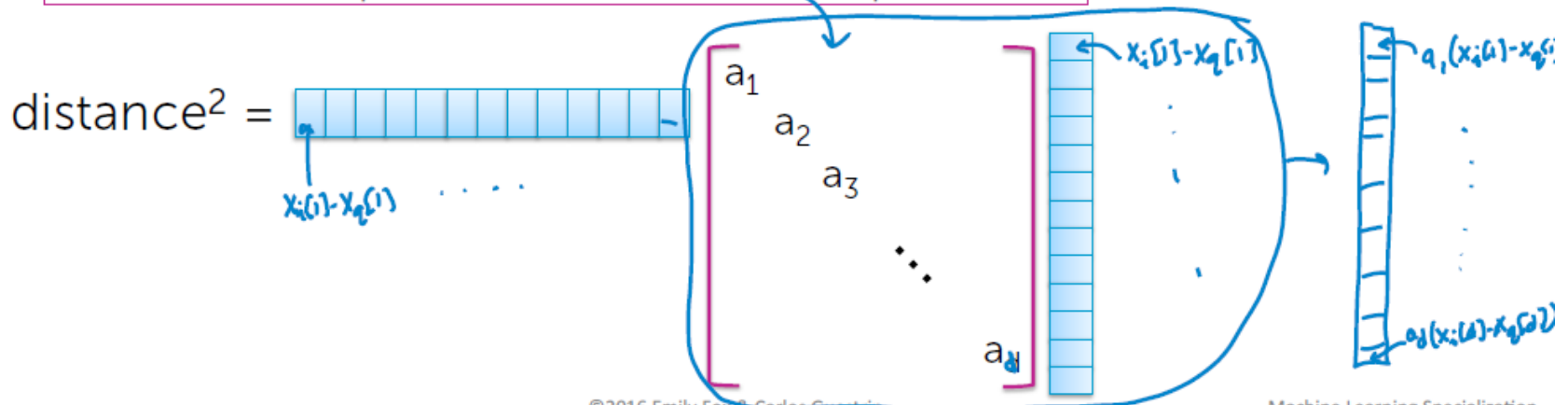
# Distance metrics:

28

## Scaled Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T \mathbf{A} (\mathbf{x}_i - \mathbf{x}_q)}$$
$$a_1 \sqrt{(\mathbf{x}_i[1] - \mathbf{x}_q[1])^2} + \dots + a_d \sqrt{(\mathbf{x}_i[d] - \mathbf{x}_q[d])^2}$$



# Distance metrics:

## Another natural inner product measure



$x_q$

1	0	0	0	5	3	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---



3	0	0	0	2	0	0	1	0	1	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---



$x_i$

**Similarity**

$$\begin{aligned} &= \mathbf{x}_i^T \mathbf{x}_q \\ &= \sum_{j=1}^d \mathbf{x}_i[j] \mathbf{x}_q[j] \\ &= 13 \end{aligned}$$

# Distance metrics:

30

## Another natural inner product measure



1	0	0	0	5	3	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---

Similarity

→ = 0

0	0	1	0	0	0	9	0	0	6	0	4	0
---	---	---	---	---	---	---	---	---	---	---	---	---



# Distance metrics

31

## Cosine similarity – normalize

$$\text{Similarity} = \frac{\sum_{j=1}^d \mathbf{x}_i[j] \mathbf{x}_q[j]}{\sqrt{\sum_{j=1}^d (\mathbf{x}_i[j])^2} \sqrt{\sum_{j=1}^d (\mathbf{x}_q[j])^2}}$$

$$\frac{\sum_{j=1}^d \mathbf{x}_i[j] \mathbf{x}_q[j]}{\sqrt{\sum_{j=1}^d (\mathbf{x}_i[j])^2} \sqrt{\sum_{j=1}^d (\mathbf{x}_q[j])^2}}$$

$$\mathbf{a}^T \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos(\theta)$$

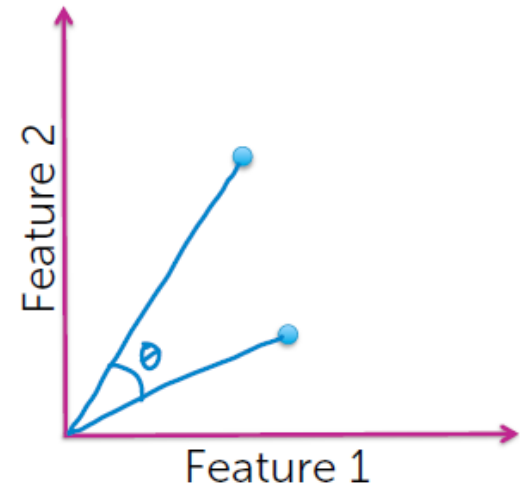
$$\mathbf{x}_i^T \mathbf{x}_q = \cos(\theta)$$

$$\frac{\mathbf{x}_i^T \mathbf{x}_q}{\|\mathbf{x}_i\| \|\mathbf{x}_q\|}$$

$$= \left( \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|} \right)^T \left( \frac{\mathbf{x}_q}{\|\mathbf{x}_q\|} \right)$$

First normalize

- Not a proper distance metric
- Efficient to compute for sparse vecs



# Distance metrics

32

## Normalize



1	0	0	0	5	3	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---

$\leftarrow x_i$

$$\sqrt{(1^2 + 5^2 + 3^2 + 1^2)} \leftarrow \|x_i\| = \sum_{j=1}^d x_i[j]^2$$

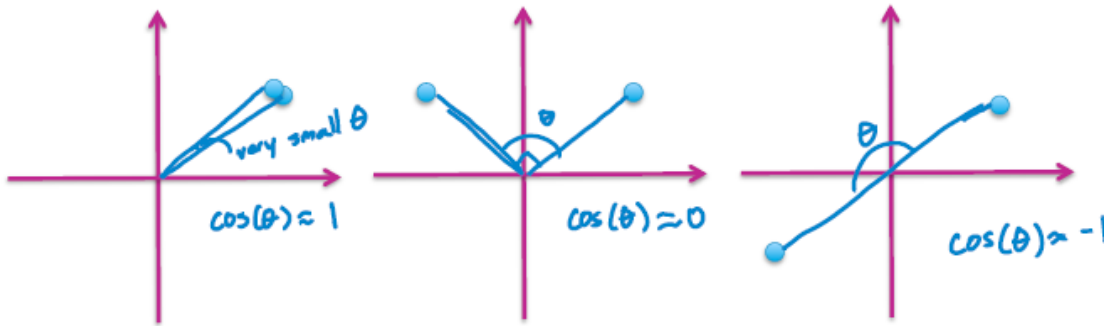
1				5	3			1				
/	0	0	0	/	/	0	0	/	0	0	0	0
6				6	6			6				



# Distance metrics

33

## Cosine similarity



In general,  $-1 < \text{similarity} < 1$

For positive features (like tf-idf)

$0 < \text{similarity} < 1$

} our focus



Define **distance = 1-similarity**

# Distance metrics

## To normalize or not?



1 0 0 0 5 3 0 0 1 0 0 0 0

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

Similarity = 13



2 0 0 0 10 6 0 0 2 0 0 0 0


6 2 0 0 4 0 0 2 0 2 0 0 0

Similarity = 52





# Distance metrics

## In the normalized case




1				5	3			1				
/	0	0	0	/	/	0	0	/	0	0	0	0
6				6	6			6				
3	1			1			1	1	1			
/	/	0	0	/	0	0	/	0	/	0	0	0
4	4			2			4	4				

Similarity = 13/24

1				5	3			1				
/	0	0	0	/	/	0	0	/	0	0	0	0
6				6	6			6				
3	1			1			1	1	1			
/	/	0	0	/	0	0	/	0	/	0	0	0
4	4			2			4	4				

Similarity = 13/24



# Distance metrics

36

## But not always desired...



long document

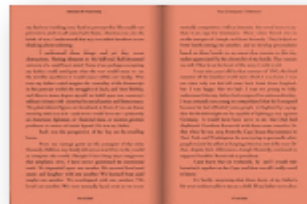


short tweet

Normalizing can  
make dissimilar  
objects appear  
more similar



long document



long document

**Common  
compromise:**  
Just cap maximum  
word counts

# Distance metrics

37

## Other distance metrics

- Mahalanobis
- rank-based
- correlation-based
- Manhattan
- Jaccard
- Hamming
- ...

# Combining distance metrics

38

## Example of document features:

1. Text of document
  - Distance metric: Cosine similarity
2. # of reads of doc
  - Distance metric: Euclidean distance

Add together with user-specified weights

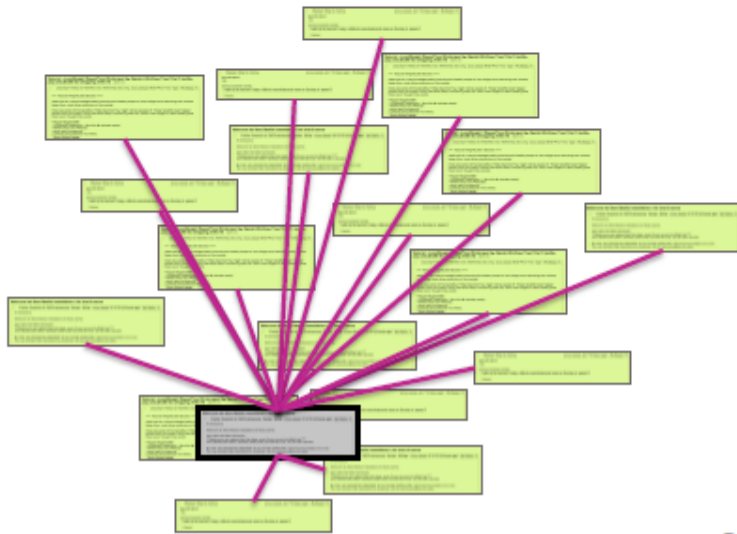
# Scaling up k-NN search by storing data in a KD-tree

# Complexity of brute-force search

40

Given a query point, scan through each point

- $O(N)$  distance computations per 1-NN query!
- $O(N \log k)$  per  $k$ -NN query!



What if  $N$  is huge??  
(and many queries)



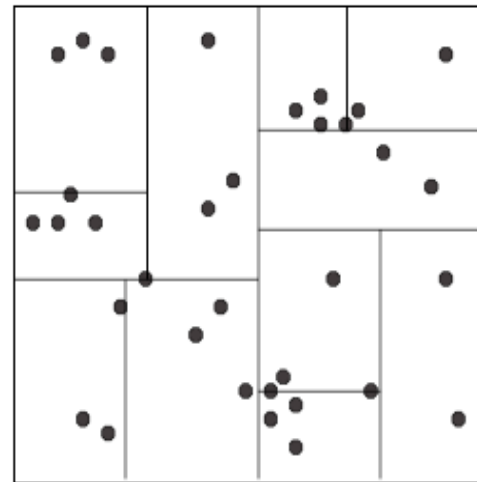
# KD-trees

41

## Structured organization of documents

- Recursively partitions points into axis aligned boxes.

Enables more efficient pruning of search space



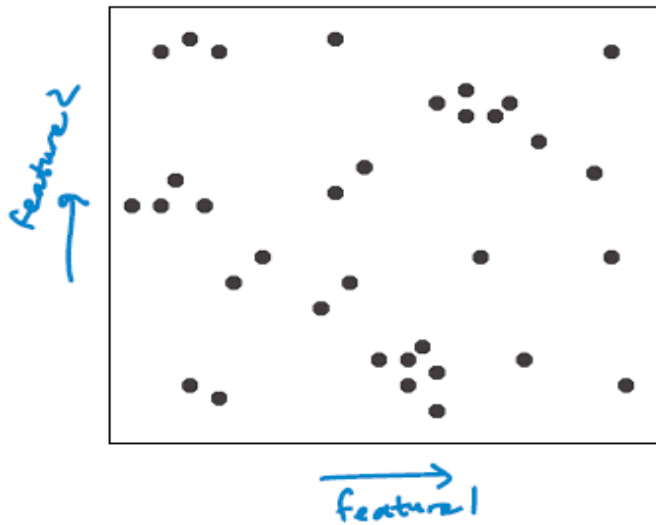
Works "well" in "low-medium" dimensions

- We'll get back to this...

# KD-trees

42

## KD-tree construction



Start with a list of d-dimensional points.

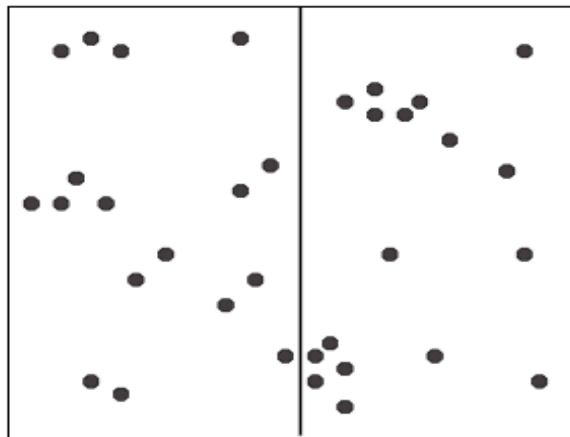
Pt	x[1]	x[2]
1	0.00	0.00
2	1.00	4.31
3	0.13	2.85
...	...	...

obs. indices  
↑  
Feat. 1 (word 1)  
↑  
Feat. 2 (word 2)

# KD-trees

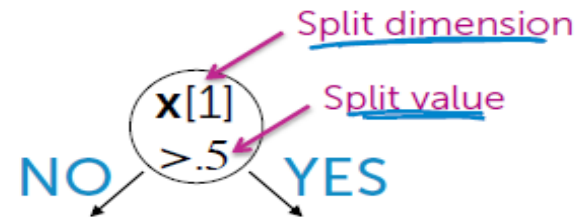
43

## KD-tree construction



← 0.5 →  
 $x[1] \leq 0.5$        $x[1] > 0.5$

Split points into 2 groups

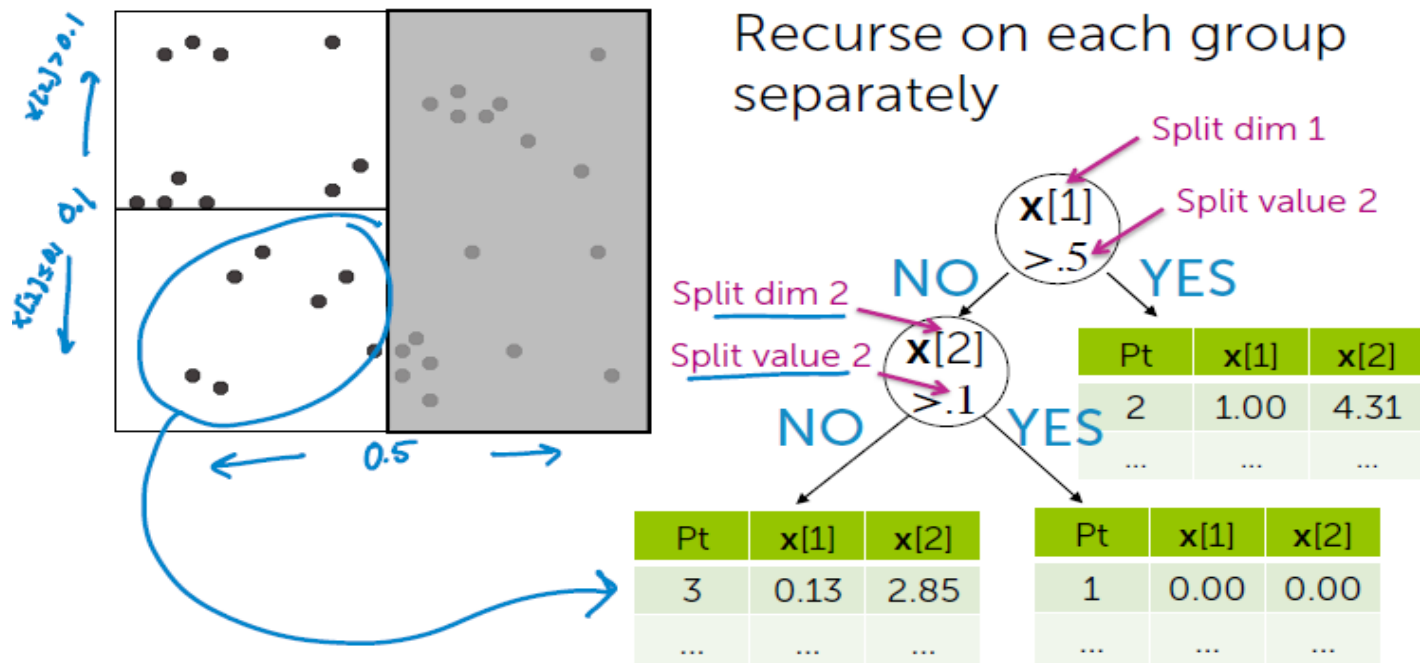


Pt	x[1]	x[2]	Pt	x[1]	x[2]
1	0.00	0.00	2	1.00	4.31
3	0.13	2.85	...	...	...
...	...	...	...	...	...

# KD-trees

44

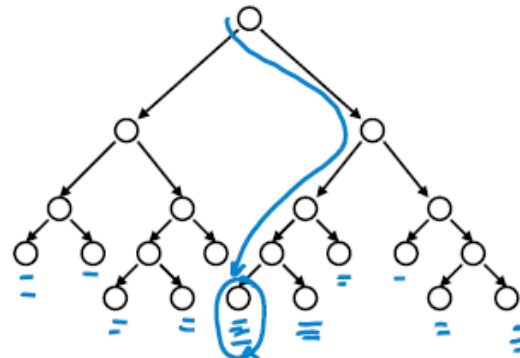
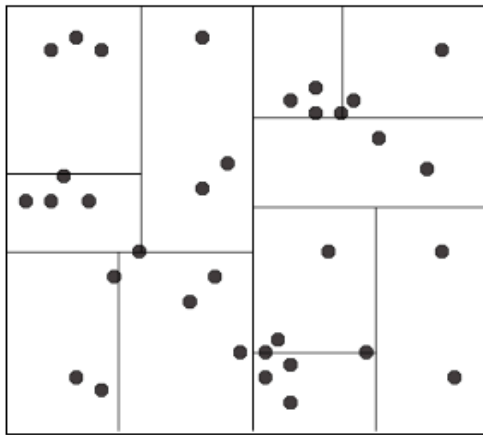
## KD-tree construction



# KD-trees

45

## KD-tree construction



points here  
satisfy all  
conditions down  
the tree to  
this leaf

Continue splitting points at each set

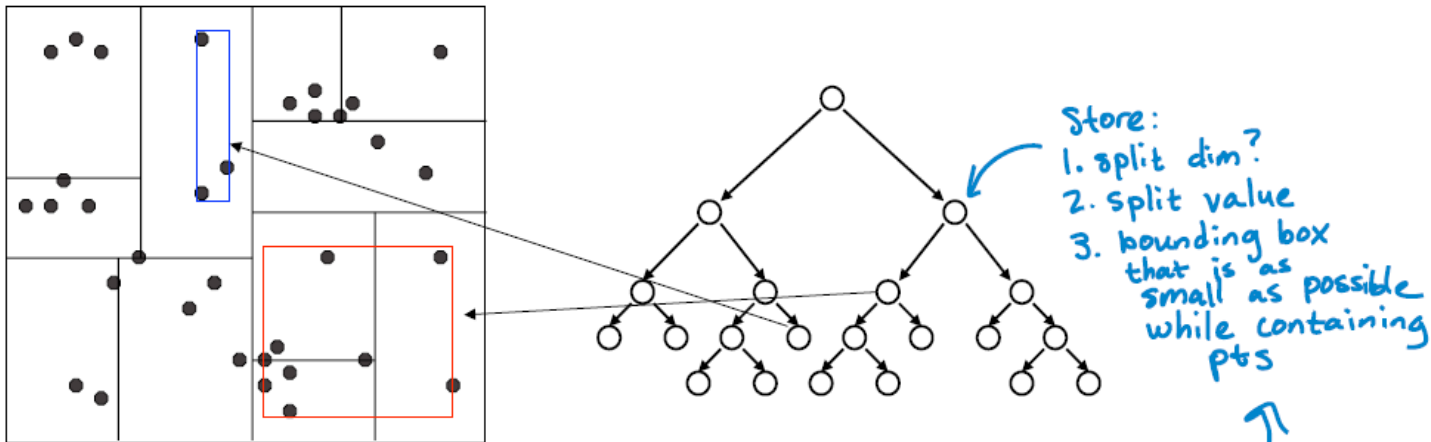
- Creates a binary tree structure

Each leaf node contains a list of points

# KD-trees

46

## KD-tree construction



Keep one additional piece of info at each node:

#3- The (tight) bounds of points at or below node

# KD-trees

47

## KD-tree construction choices

Use heuristics to make splitting decisions:

- Which dimension do we split along?

widest (or alternate)

- Which value do we split at?

median (or center point of box,  
ignoring data in box)

- When do we stop?

Fewer than  $m$  pts left

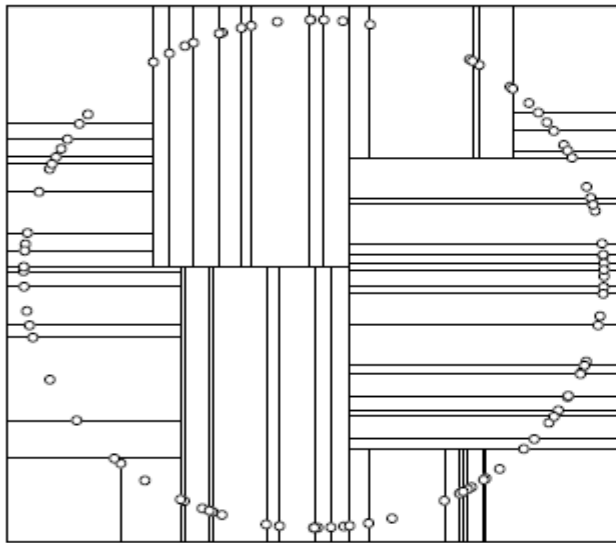
or

box hits minimum width

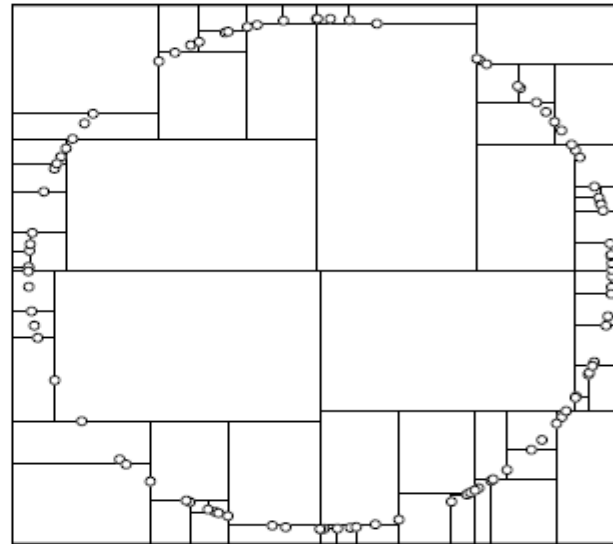
# KD-trees

48

## Many heuristics...



median heuristic

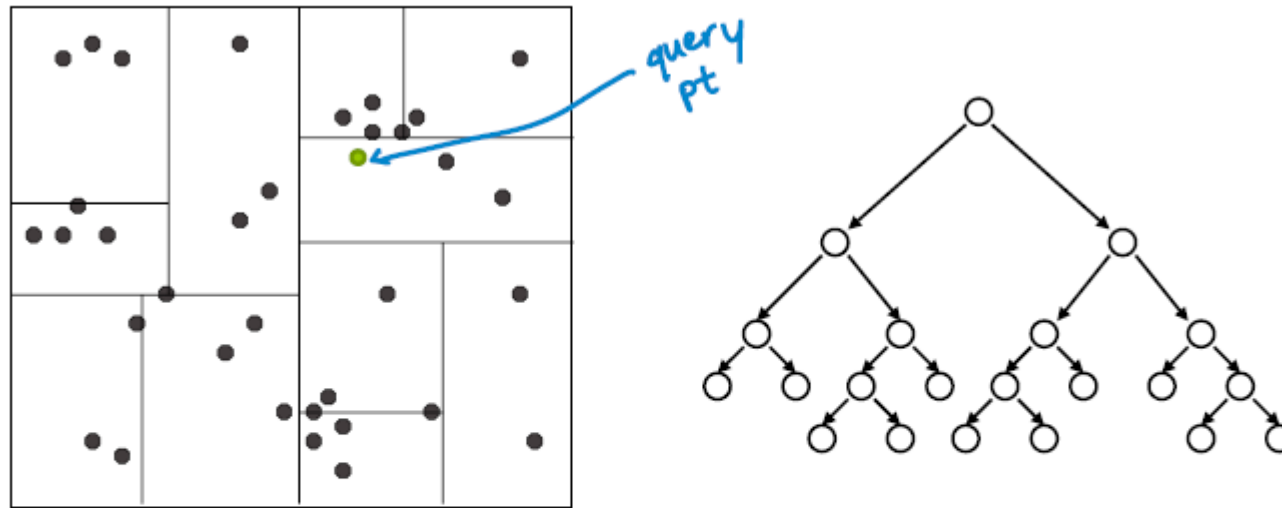


center-of-range  
heuristic



# Nearest neighbor with KD-trees

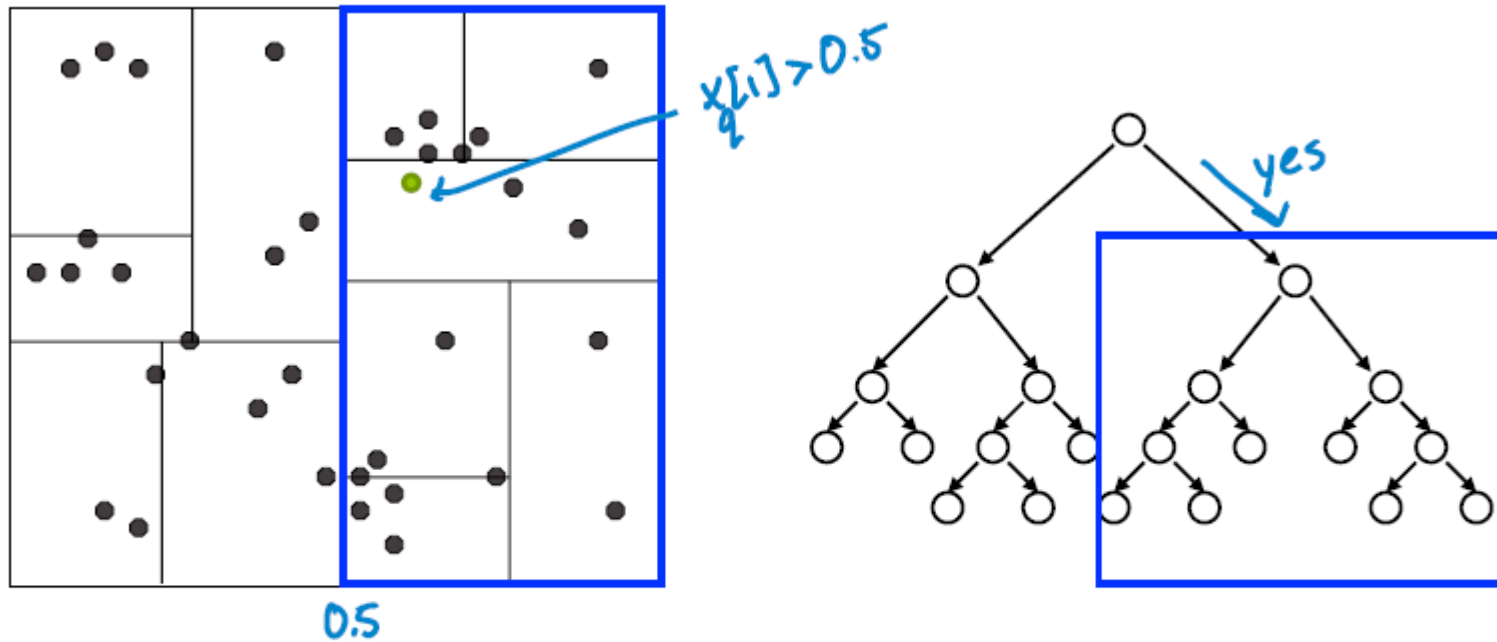
49



Traverse tree looking for nearest neighbor to query point

# Nearest neighbor with KD-trees

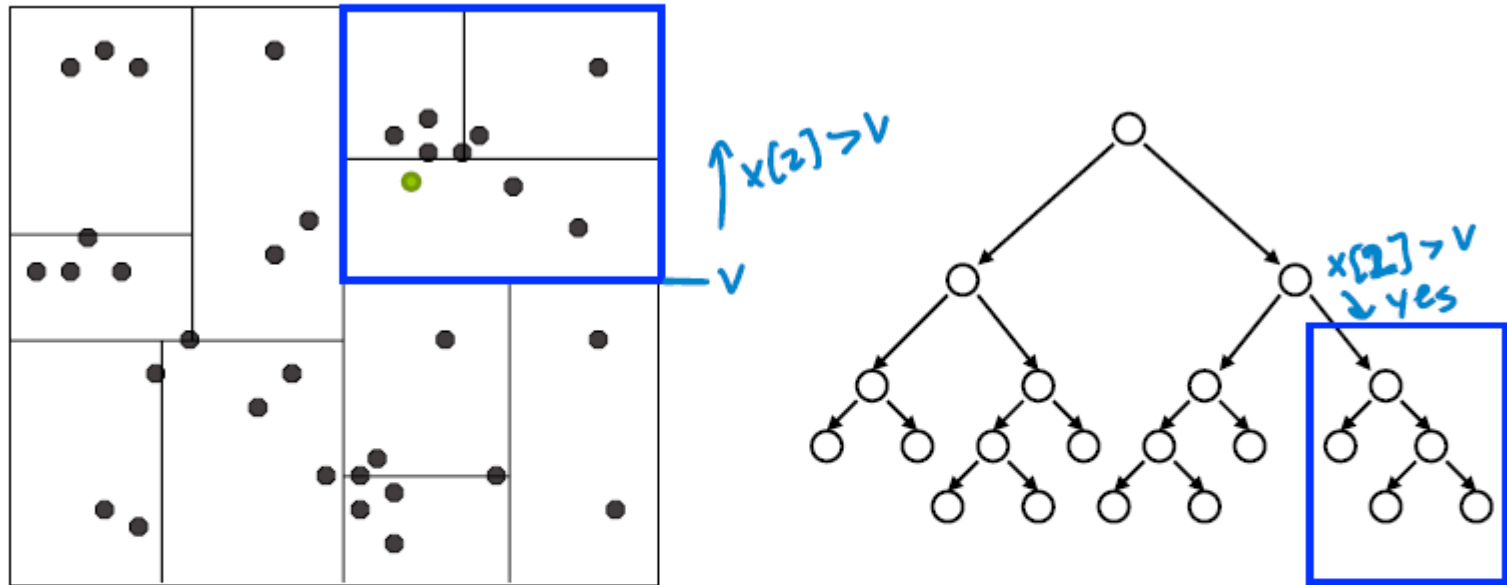
50



1. Start by exploring leaf node containing query point

# Nearest neighbor with KD-trees

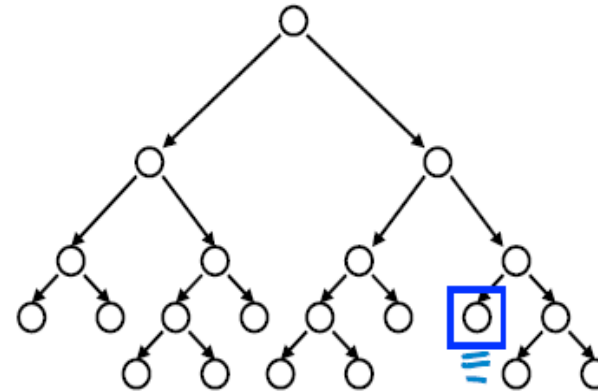
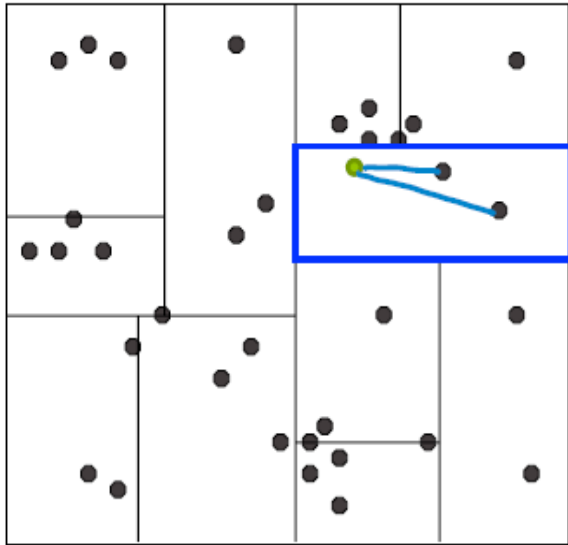
51



1. Start by exploring leaf node containing query point

# Nearest neighbor with KD-trees

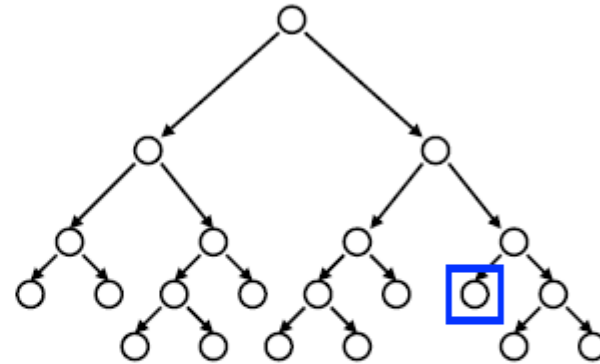
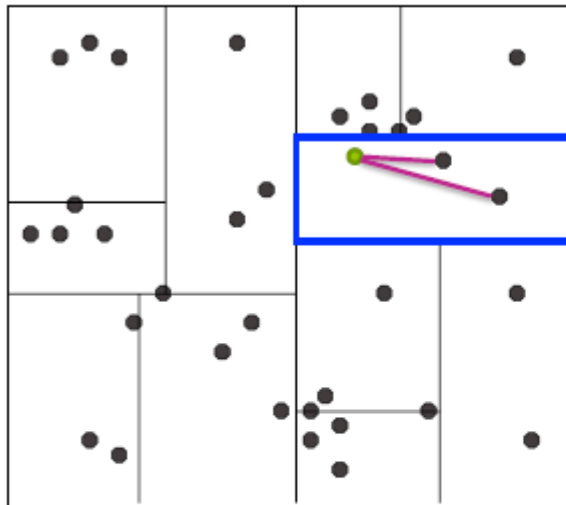
52



1. Start by exploring leaf node containing query point

# Nearest neighbor with KD-trees

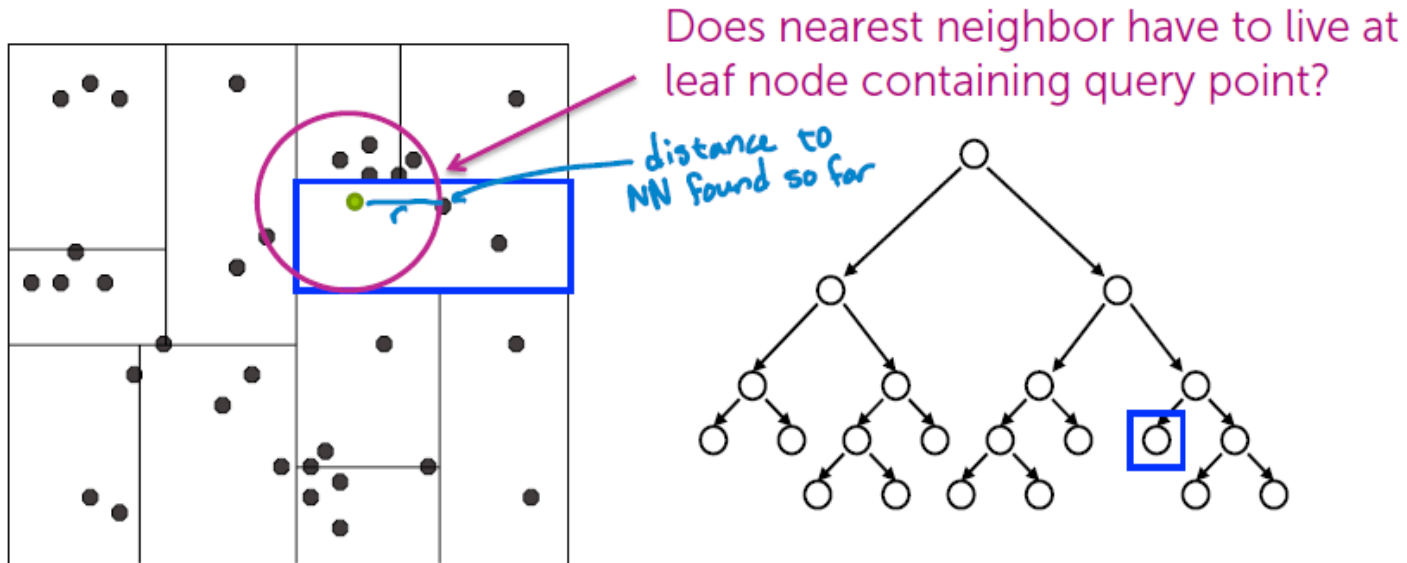
53



1. Start by exploring leaf node containing query point
2. Compute distance to each other point at leaf node

# Nearest neighbor with KD-trees

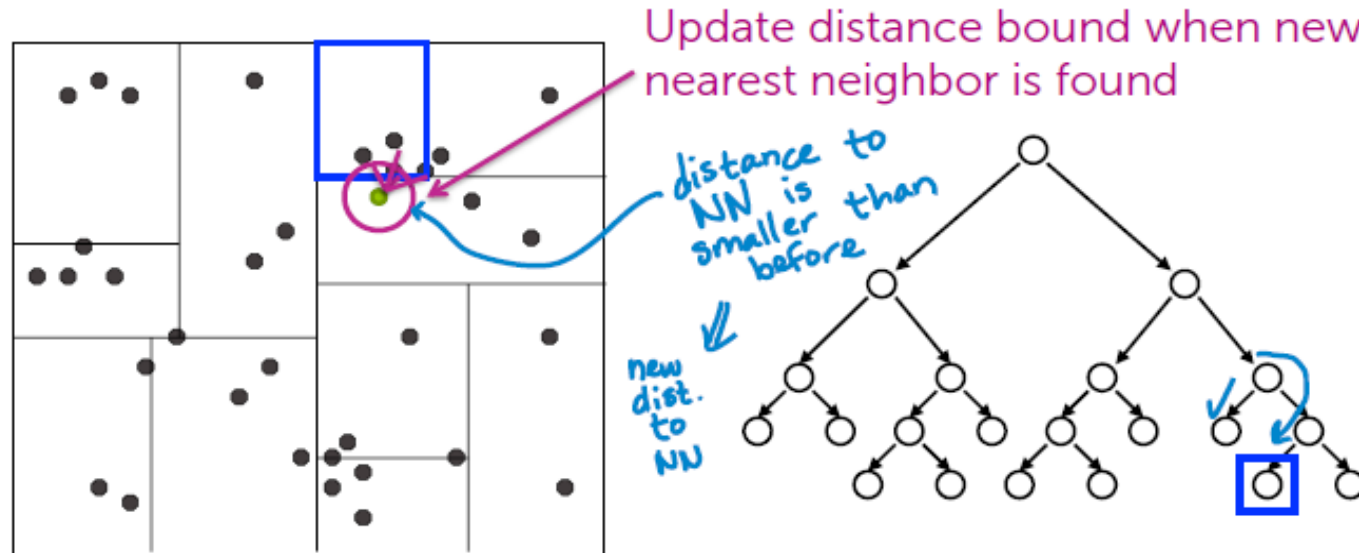
54



1. Start by exploring leaf node containing query point
2. Compute distance to each other point at leaf node

# Nearest neighbor with KD-trees

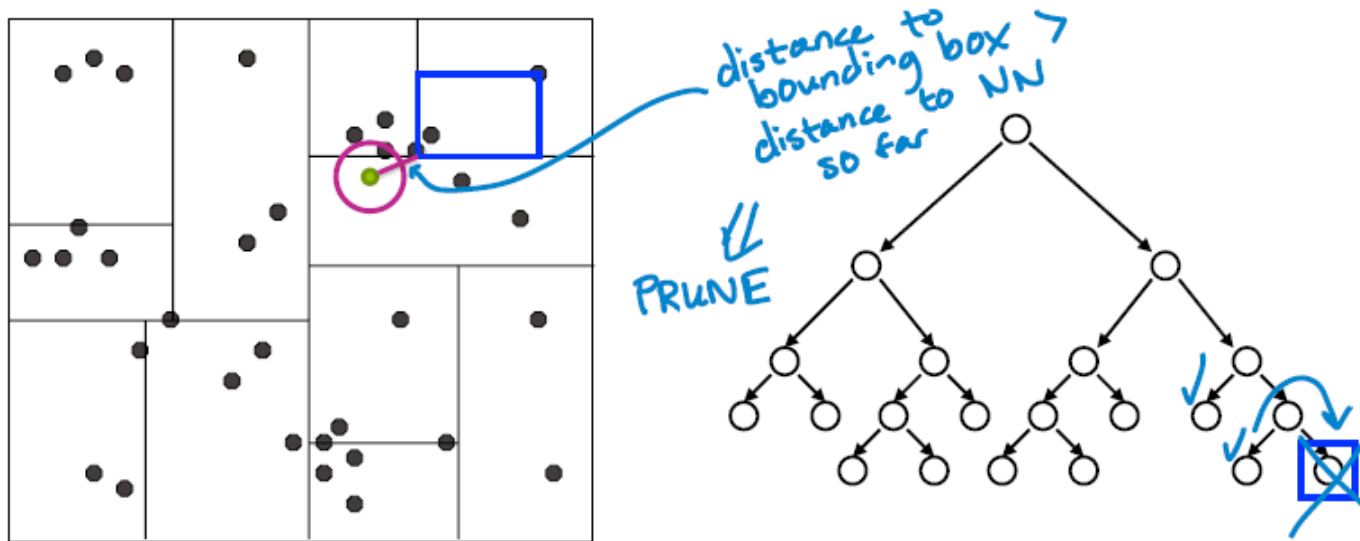
55



1. Start by exploring leaf node containing query point
2. Compute distance to each other point at leaf node
3. Backtrack and try other branch at each node visited

# Nearest neighbor with KD-trees

56

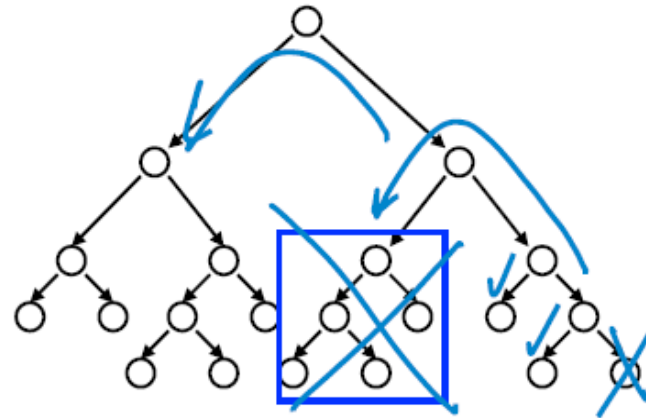
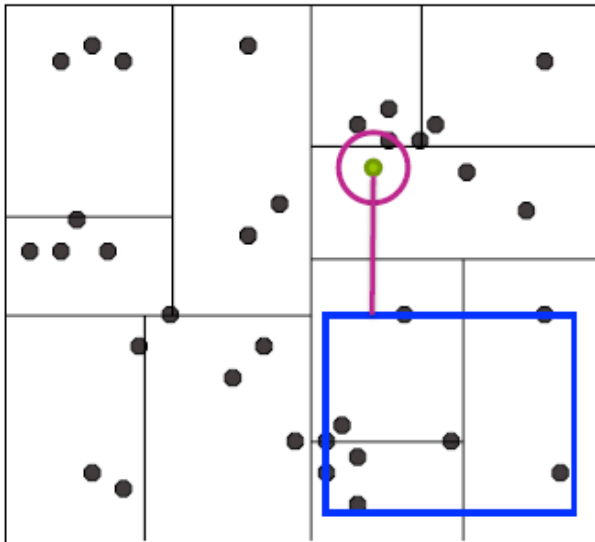


Use distance bound and bounding box of each node to **prune** parts of tree that **cannot include nearest neighbor**



# Nearest neighbor with KD-trees

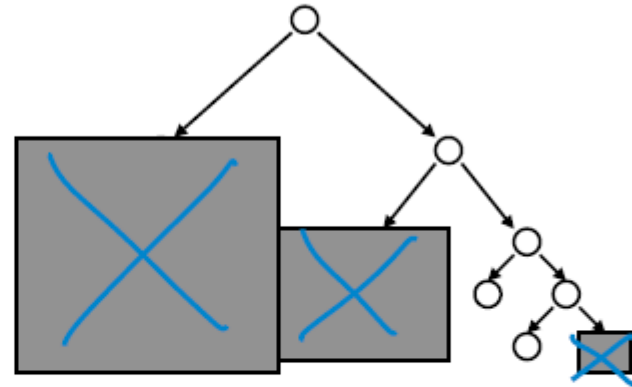
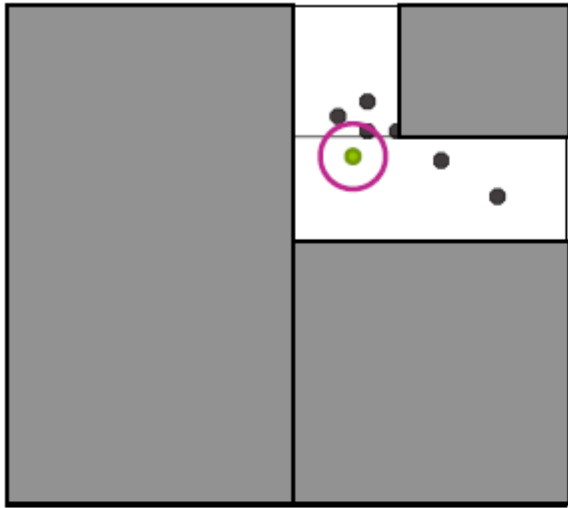
57



Use distance bound and bounding box of each node to **prune** parts of tree that **cannot include nearest neighbor**

# Nearest neighbor with KD-trees

58



Use distance bound and bounding box of each node to **prune** parts of tree that **cannot include nearest neighbor**

# Nearest neighbor with KD-trees

59

## Complexity



For (nearly) balanced, binary trees...

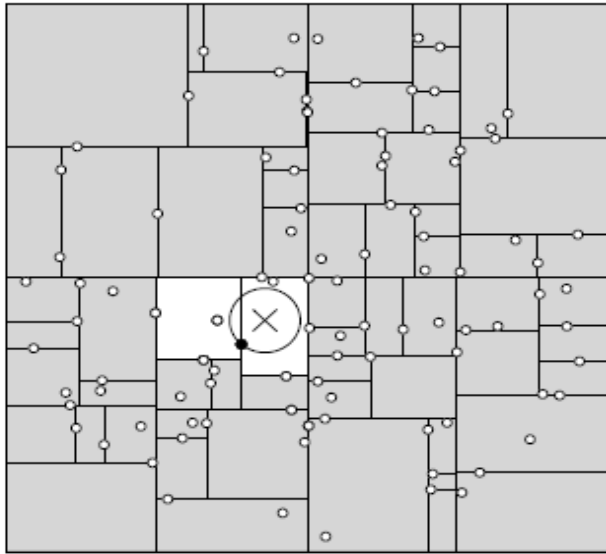
- Construction
  - Size:  $2N-1$  nodes if 1 datapoint at each leaf  $\rightarrow \underline{O(N)}$
  - Depth:  $O(\log N)$
  - Median + send points left right:  $O(N)$  at every level of the tree
  - Construction time:  $O(N \log N)$
- 1-NN query
  - Traverse down tree to starting point:  $O(\log N)$
  - Maximum backtrack and traverse:  $O(N)$  in worst case
  - Complexity range:  $O(\log N) \rightarrow O(N)$

Under some assumptions on distribution of points, we get  $O(\log N)$  but exponential in  $d$

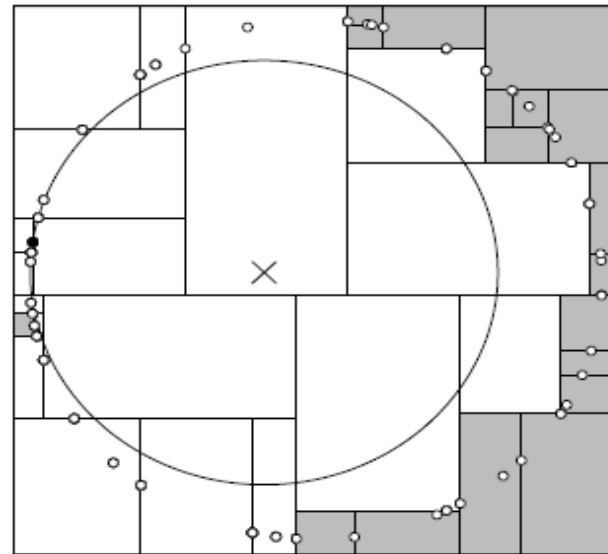
# Nearest neighbor with KD-trees

60

## Complexity



pruned many  
(closer to  $O(\log N)$ )



pruned few  
(closer to  $O(N)$ )

# Complexity for N queries

61

- Ask for nearest neighbor to each doc

*N queries*

- Brute force 1-NN:

*$O(N^2)$*

- kd-trees:

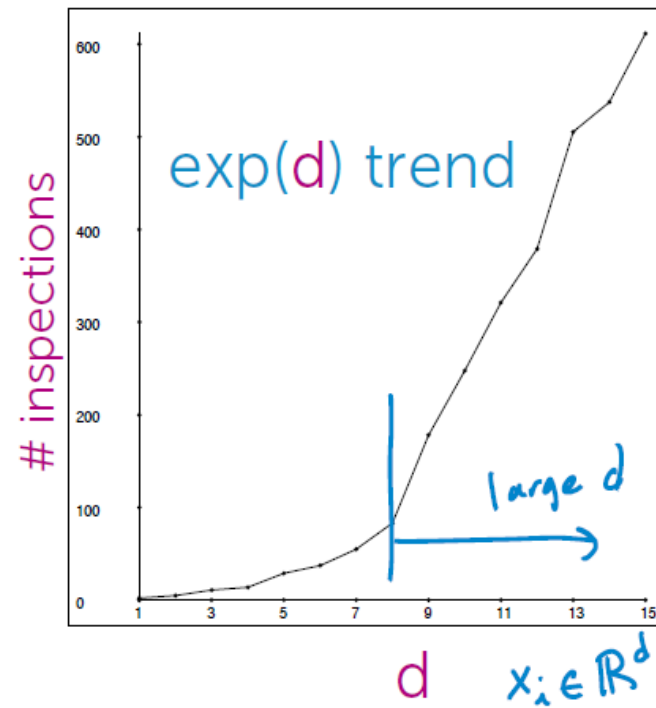
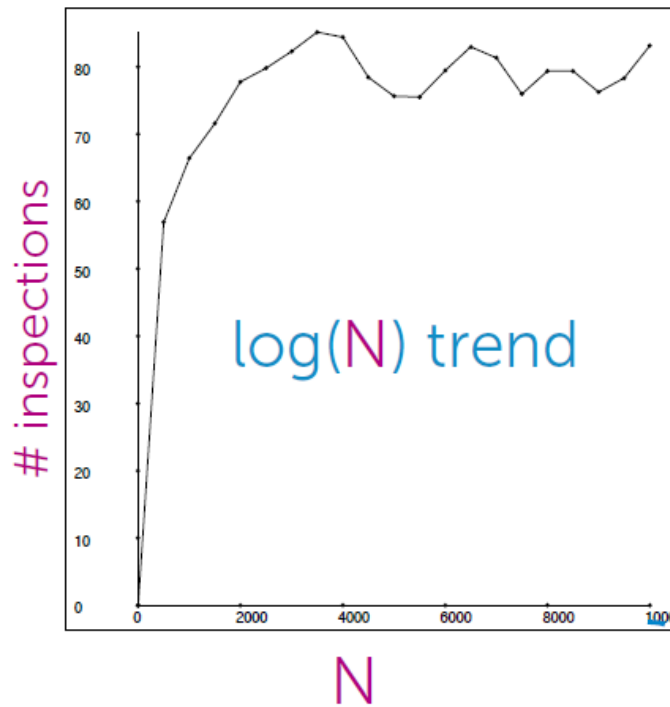
*$O(N \log N) \rightarrow O(N^2)$*

*↑  
potentially  
very large  
savings for  
large N!*

# Complexity for N queries

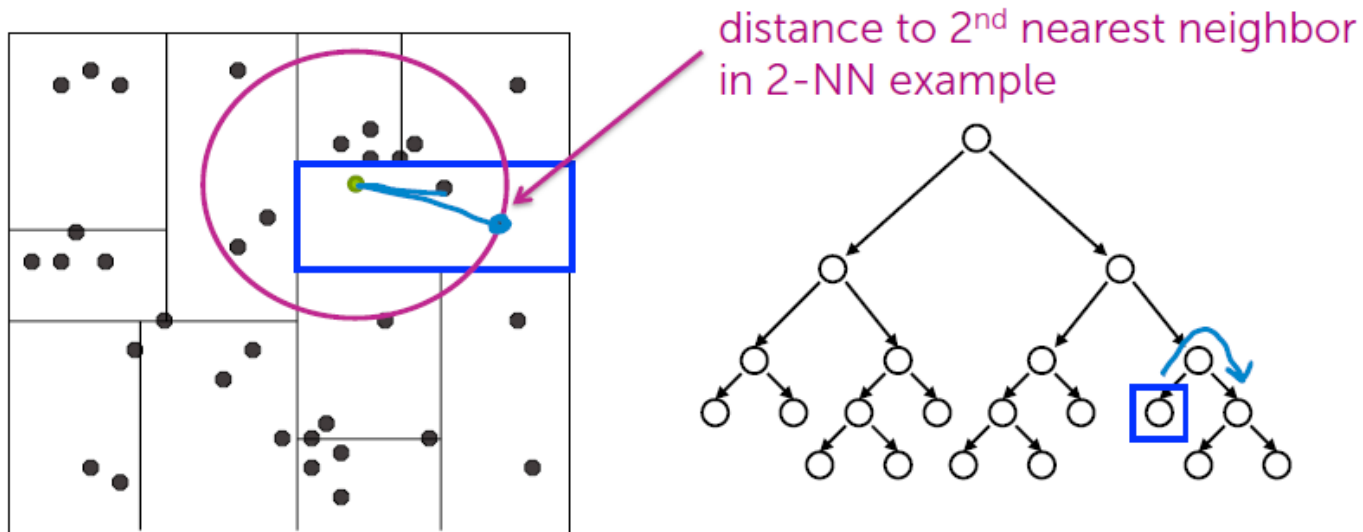
62

## Inspections vs. N and d



# k-NN with KD-trees

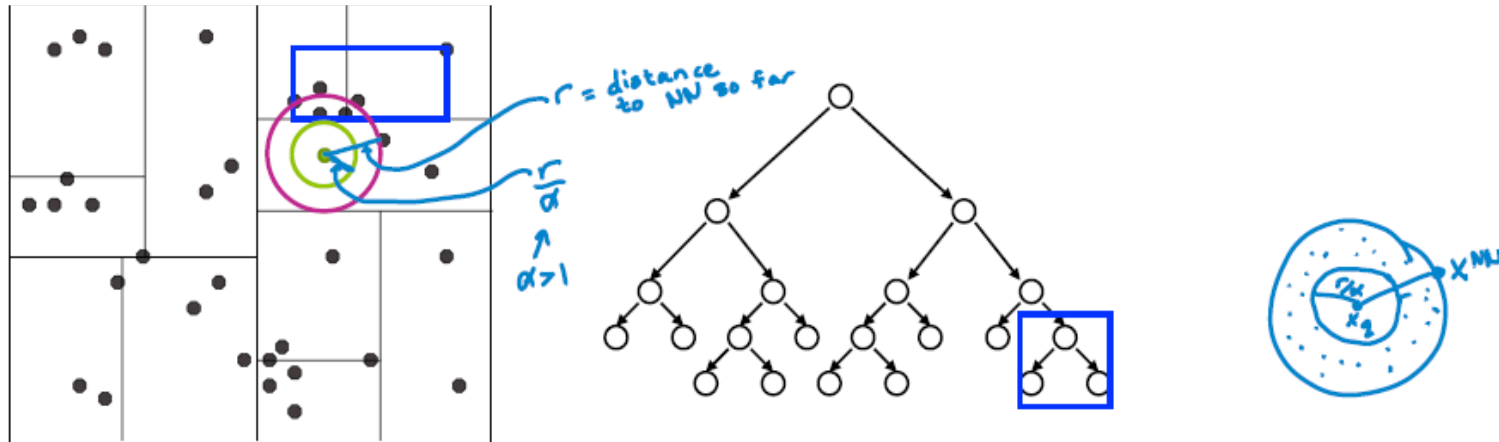
63



Exactly same algorithm, but maintain distance to  
furthest of current  $k$  nearest neighbors

# Approximate k-NN with KD-trees

64



**Before:** Prune when distance to bounding box  $> r$

**Now:** Prune when distance to bounding box  $> r/\alpha$

Prunes more than allowed, but can **guarantee** that if we return a neighbor at distance  $r$ , then there is **no neighbor closer** than  $r/\alpha$

← Bound loose...In practice, often closer to optimal.

Saves lots of search time at little cost in quality of NN!



# Closing remarks on KD-trees

65

## Tons of variants of kd-trees

- On construction of trees  
(heuristics for splitting, stopping, representing branches...)
- Other representational data structures for fast NN search  
(e.g., ball trees,...)

## Nearest Neighbor Search

- Distance metric and data representation crucial to answer returned

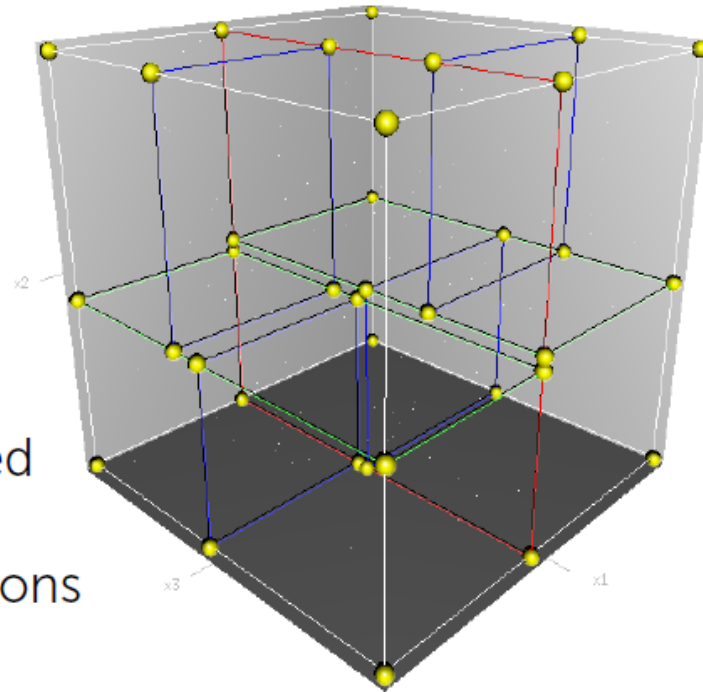
## For both, high-dim spaces are hard!

- Number of kd-tree searches can be exponential in dimension
  - Rule of thumb...  $N \gg 2^d$ ... Typically useless for large  $d$ .
- Distances sensitive to irrelevant features
  - Most dimensions are just noise  $\rightarrow$  everything is far away
  - Need technique to learn which features are important to given task

# KD-tree in high dimensions

66

- Unlikely to have any data points close to query point
- Once “nearby” point is found, the search radius is likely to intersect many hypercubes in at least one dim
- Not many nodes can be pruned
- Can show under some conditions that you visit at least  $2^d$  nodes



# Moving away from exact NN search

67

- Approximate neighbor finding...
  - Don't find exact neighbor, but that's okay for many applications



Out of millions of articles, do we need the closest article or just one that's pretty similar?

Do we even fully trust our measure of similarity???

- Focus on methods that provide good probabilistic guarantees on approximation

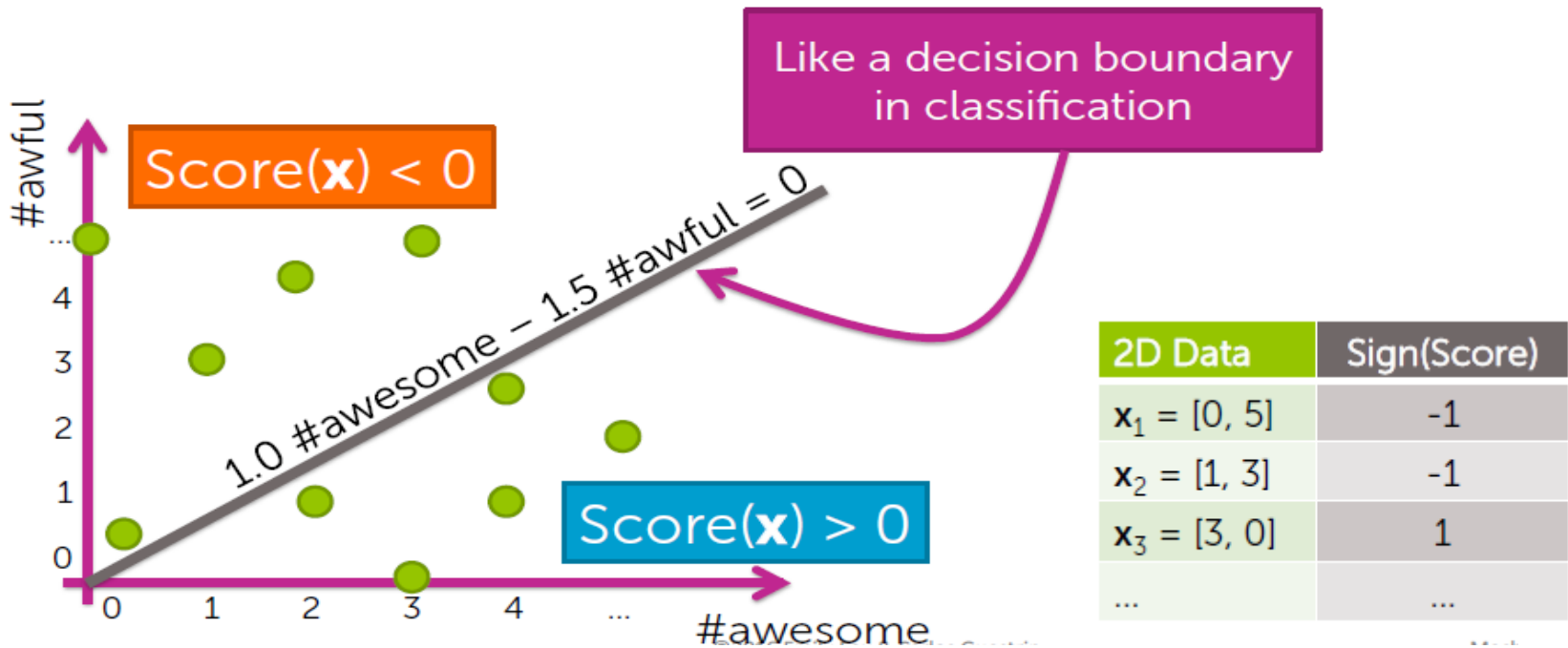
# Locality Sensitive Hashing (LHS) as alternative to KD-trees

# Locality sensitive hashing

69

## Simple "binning" of data into 2 bins

$$\text{Score}(\mathbf{x}) = 1.0 \text{ \#awesome} - 1.5 \text{ \#awful}$$



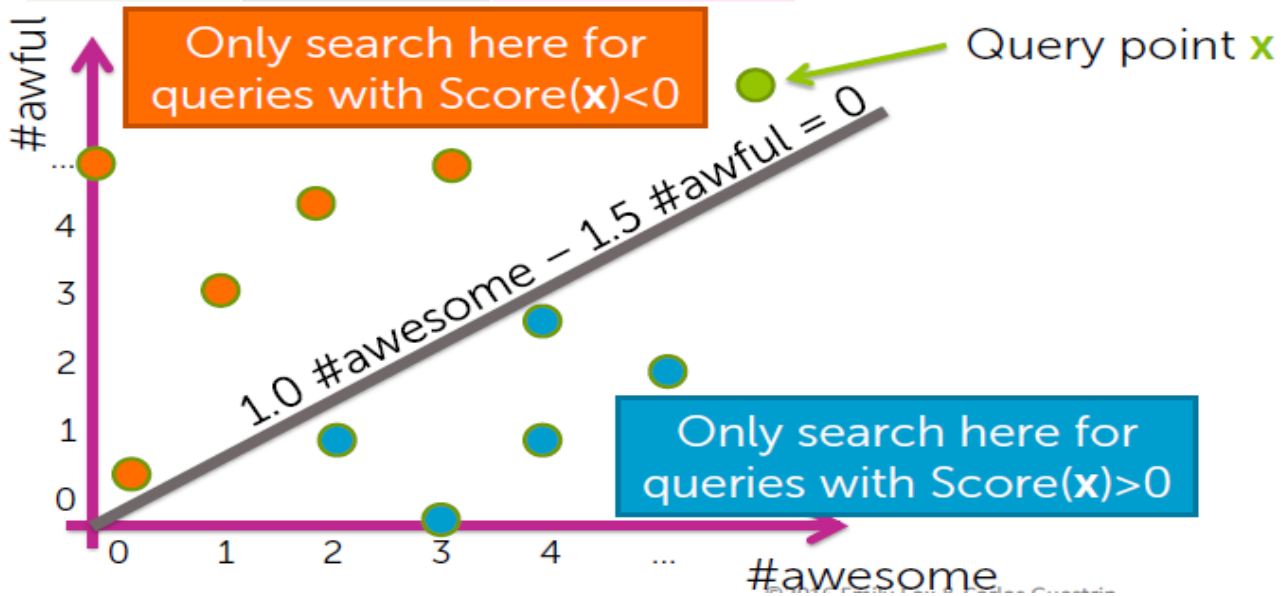
# Locality sensitive hashing

70

## Using bins for NN search

2D Data	Sign(Score)	Bin index
$x_1 = [0, 5]$	-1	0
$x_2 = [1, 3]$	-1	0
$x_3 = [3, 0]$	1	1
...	...	...

candidate neighbors if  $\text{Score}(x) < 0$



# Locality sensitive hashing

71

## Using score for NN search

2D Data	Sign(Score)	Bin index
$x_1 = [0, 5]$	-1	0
$x_2 = [1, 3]$	-1	0
$x_3 = [3, 0]$	1	1
...	...	...

candidate neighbors if  $\text{Score}(x) < 0$



Bin	0	1
List containing indices of datapoints:	{1,2,4,7,...}	{3,5,6,8,...}

HASH TABLE

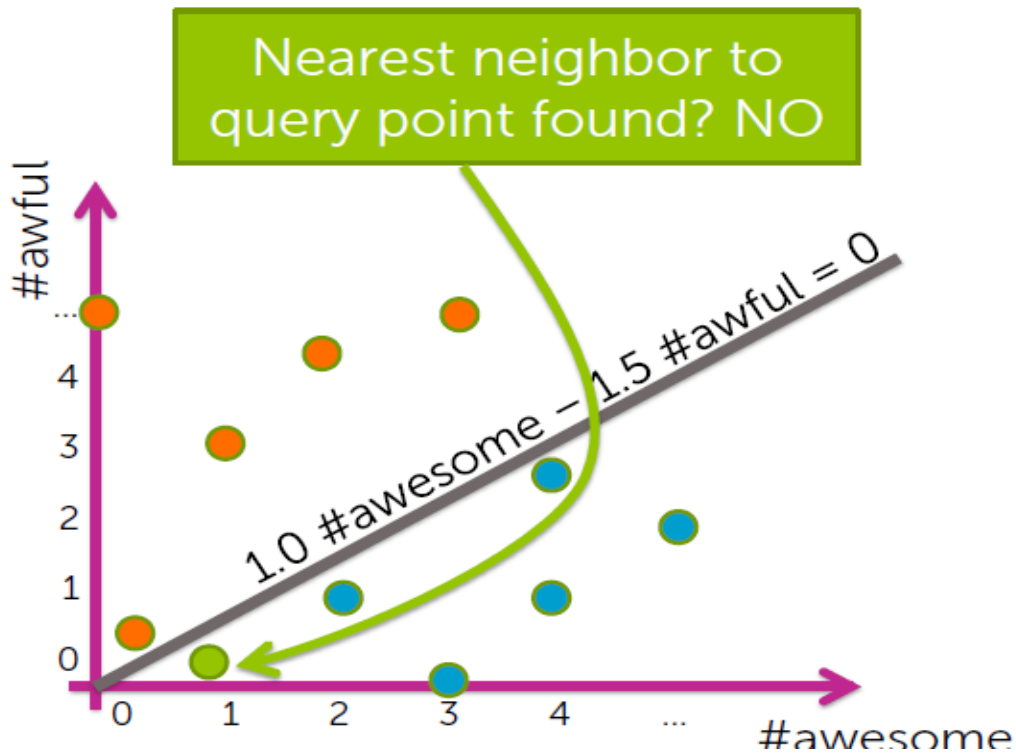
search for NN amongst this set



# Locality sensitive hashing

72

## Provides approximate NN





# Locality sensitive hashing

73

## Three potential issues with simple approach

1. Challenging to find good line
2. Poor quality solution:
  - Points close together get split into separate bins
3. Large computational cost:
  - Bins might contain many points, so still searching over large set for each NN query

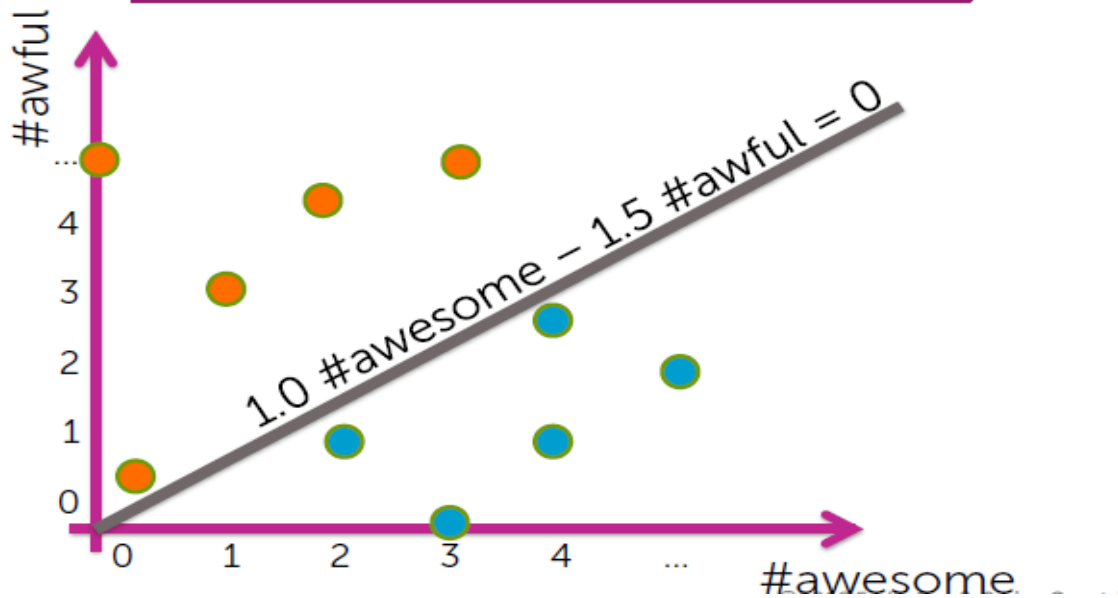
Bin	0	1
List containing indices of datapoints:	{1,2,4,7,...}	{3,5,6,8,...}

# Locality sensitive hashing

74

## How to define the line?

**Crazy idea:**  
Define line randomly!

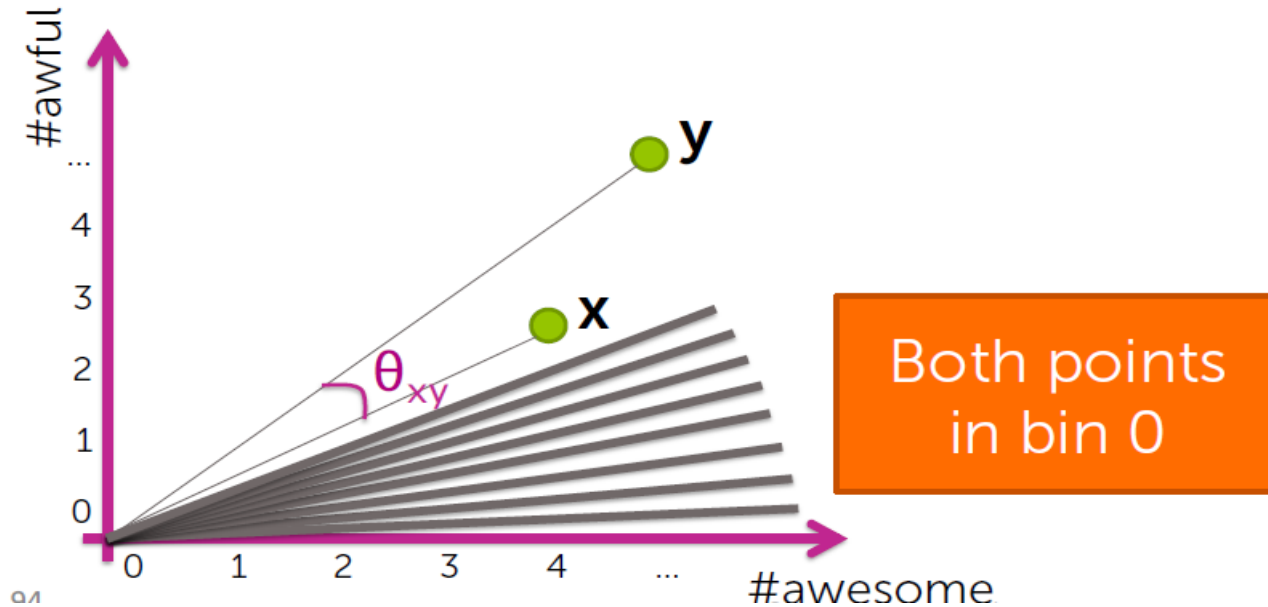


# Locality sensitive hashing

75

## How bad can a random line be?

**Goal:** If  $x, y$  are close (according to *cosine similarity*), want binned values to be the same.

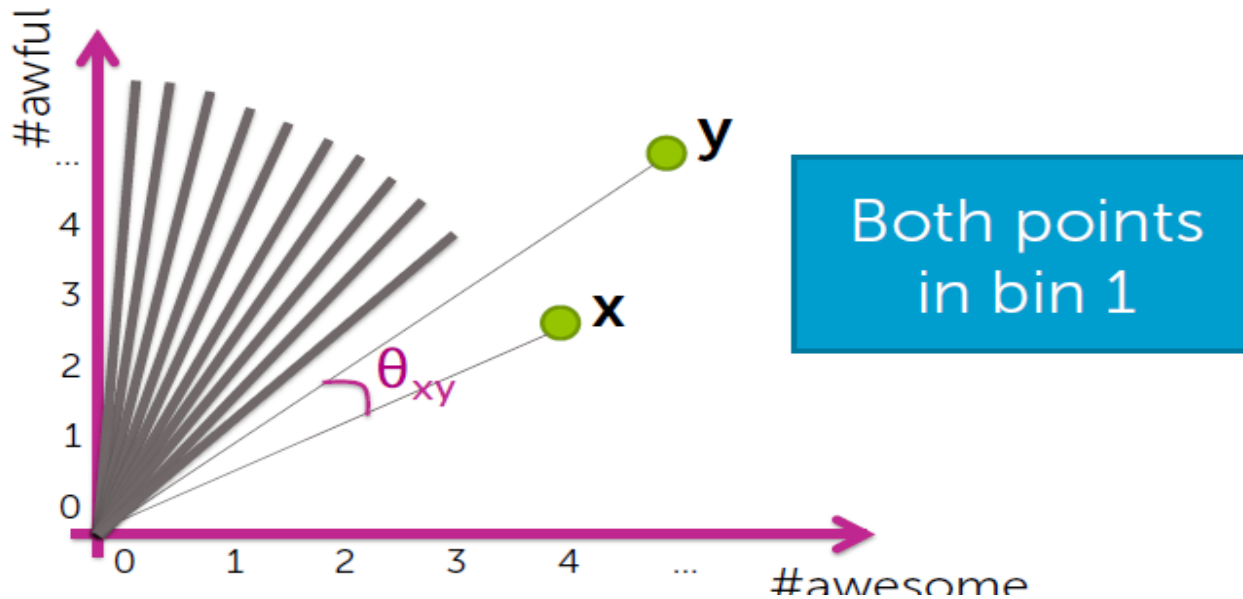


# Locality sensitive hashing

76

## How bad can a random line be?

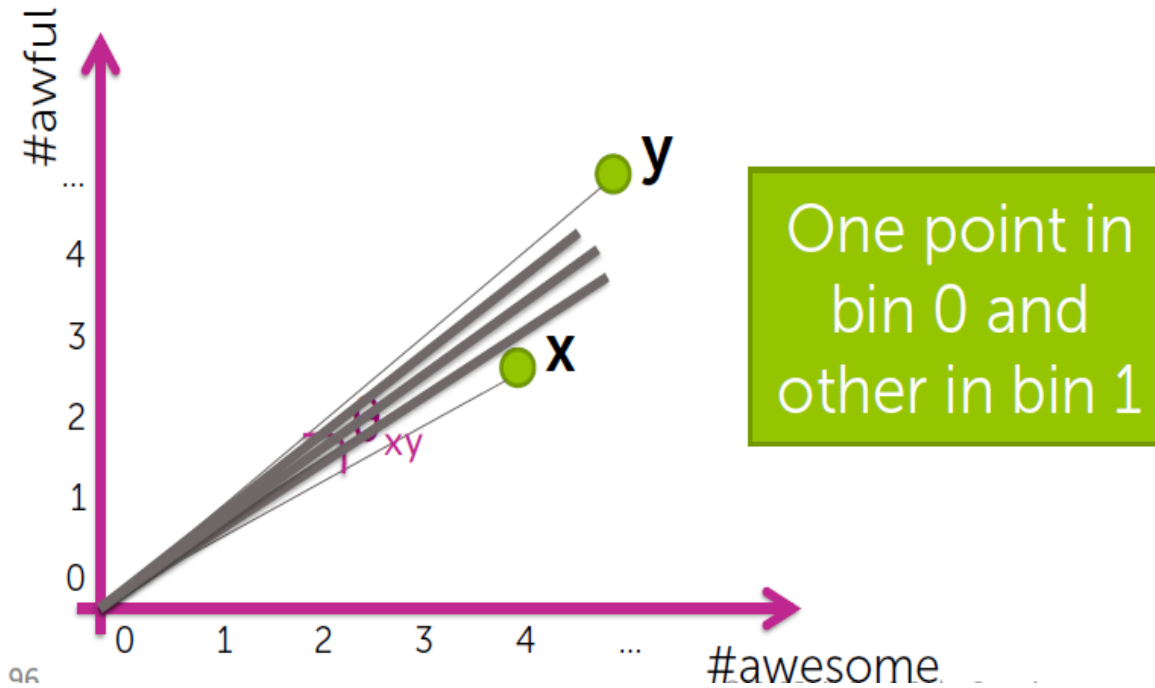
**Goal:** If  $x, y$  are close (according to cosine similarity), want binned values to be the same.



# Locality sensitive hashing

77

**Goal:** If  $x, y$  are close (according to cosine similarity), want binned values to be the same.

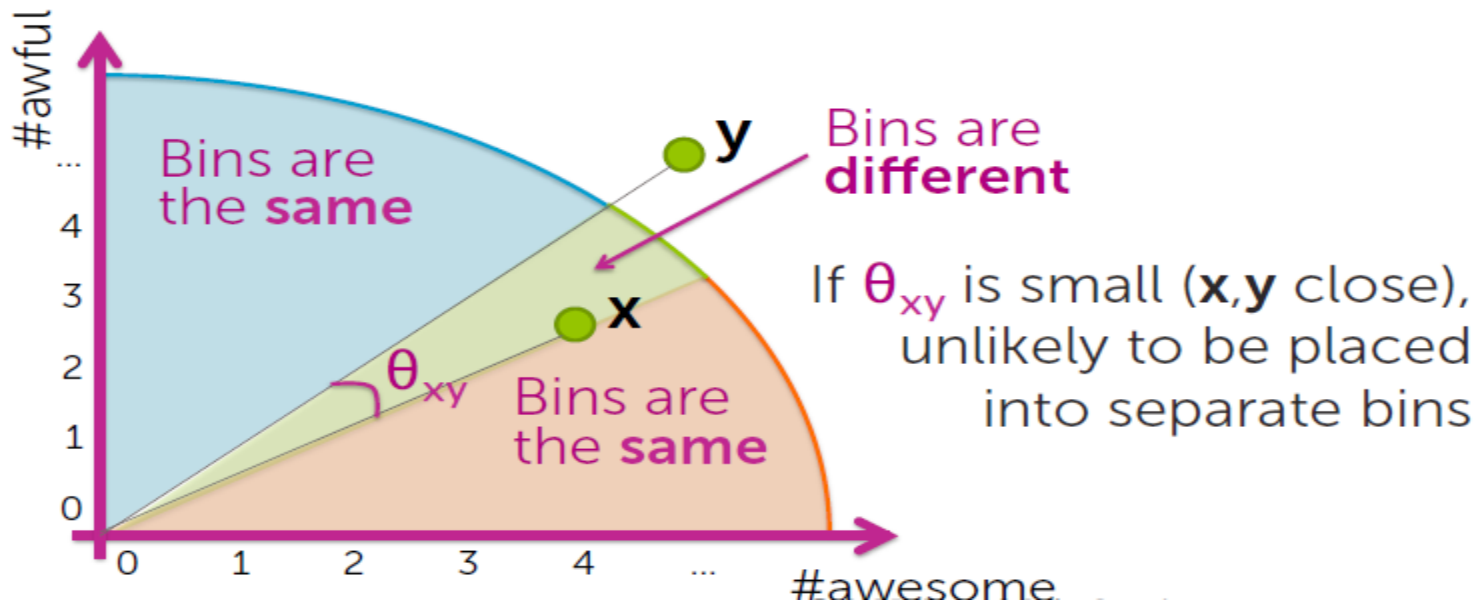


# Locality sensitive hashing

78

## How bad can a random line be?

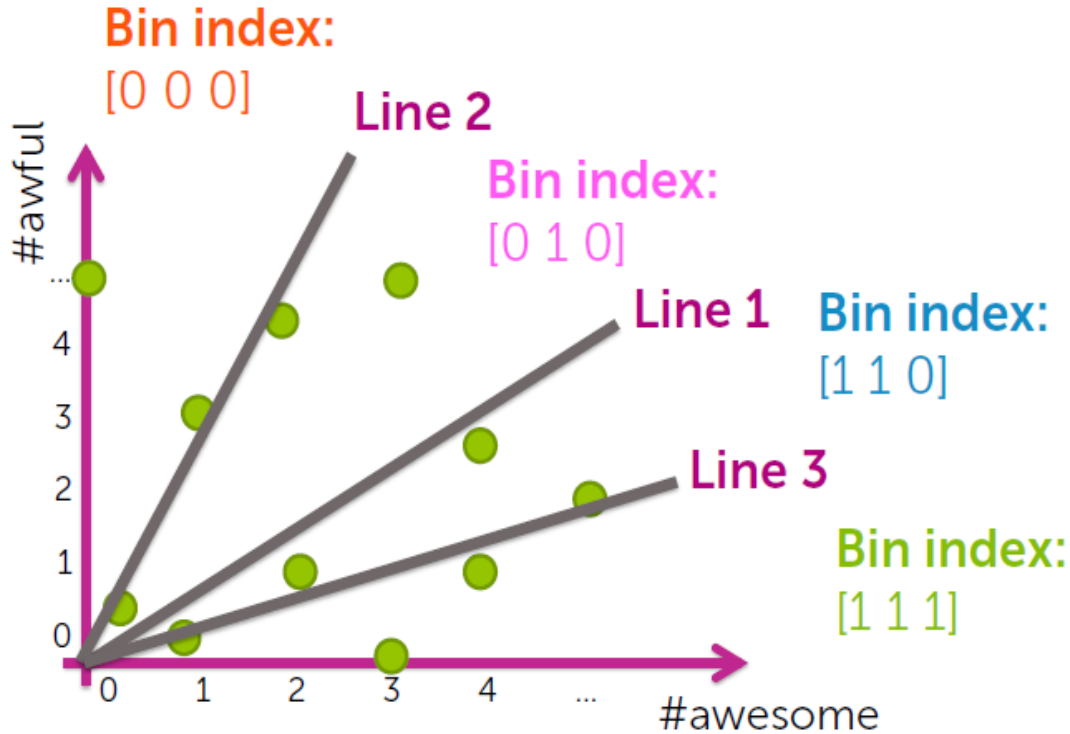
**Goal:** If  $\mathbf{x}, \mathbf{y}$  are close (according to cosine similarity), want binned values to be the same.



# LSH: improving efficiency

79

## Reducing search cost through more bins



# LSH: improving efficiency

80

## Using score for NN search

2D Data	Sign (Score <sub>1</sub> )	Bin 1 index	Sign (Score <sub>2</sub> )	Bin 2 index	Sign (Score <sub>3</sub> )	Bin 3 index
$x_1 = [0, 5]$	-1	0	-1	0	-1	0
$x_2 = [1, 3]$	-1	0	-1	0	-1	0
$x_3 = [3, 0]$	1	1	1	1	1	1
...	...	...	...	...	...	...

Bin	[0 0 0] = 0	[0 0 1] = 1	[0 1 0] = 2	[0 1 1] = 3	[1 0 0] = 4	[1 0 1] = 5	[1 1 0] = 6	[1 1 1] = 7
Data indices:	{1,2}	--	{4,8,11}	--	--	--	{7,9,10}	{3,5,6}

search for NN  
amongst this set



# LSH: improving efficiency

81

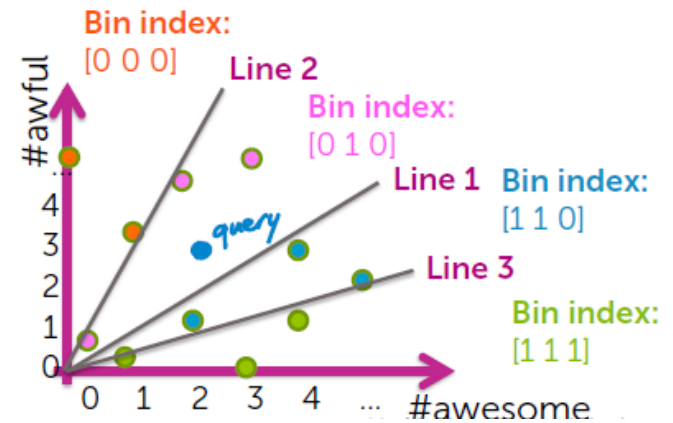
## Improving search quality by searching neighboring bins

Bin	[0 0 0] = 0	[0 0 1] = 1	[0 1 0] = 2	[0 1 1] = 3	[1 0 0] = 4	[1 0 1] = 5	[1 1 0] = 6	[1 1 1] = 7
Data indices:	{1,2}	--	{4,8,11}	--	--	--	{7,9,10}	{3,5,6}

Query point here, but is NN?

Not necessarily

Even worse than before...Each line can split pts. Sacrificing accuracy for speed



107

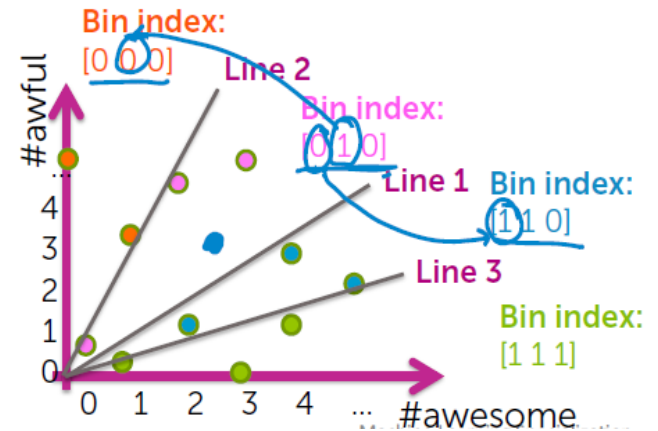
# LSH: improving efficiency

82

## Improving search quality by searching neighboring bins

Bin	[0 0 0] = 0	[0 0 1] = 1	[0 1 0] = 2	[0 1 1] = 3	[1 0 0] = 4	[1 0 1] = 5	[1 1 0] = 6	[1 1 1] = 7
Data indices:	<u>{1,2}</u>	--	<u>{4,8,11}</u>	--	--	--	<u>{7,9,10}</u>	{3,5,6}

Next closest bins  
(flip 1 bit)



# LSH: improving efficiency

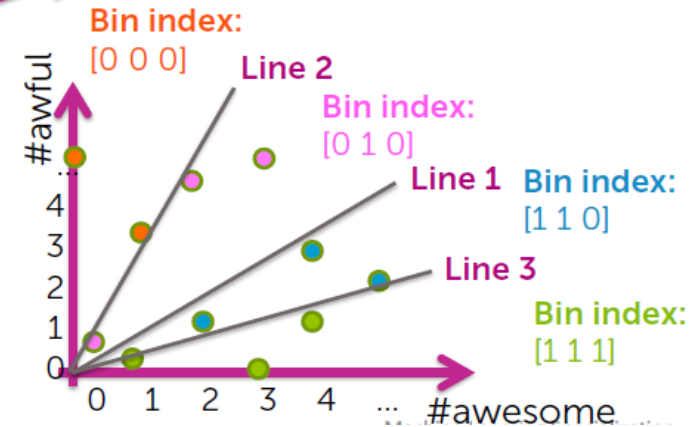
83

Improving search quality by searching neighboring bins

Bin	[0 0 0] = 0	[0 0 1] = 1	[0 1 0] = 2	[0 1 1] = 3	[1 0 0] = 4	[1 0 1] = 5	[1 1 0] = 6	[1 1 1] = 7
Data indices:	{1,2}	--	{4,8,11}	--	--	--	{7,9,10}	<u>{3,5,6}</u>



Further bin (flip 2 bits)



# LSH: improving efficiency

84

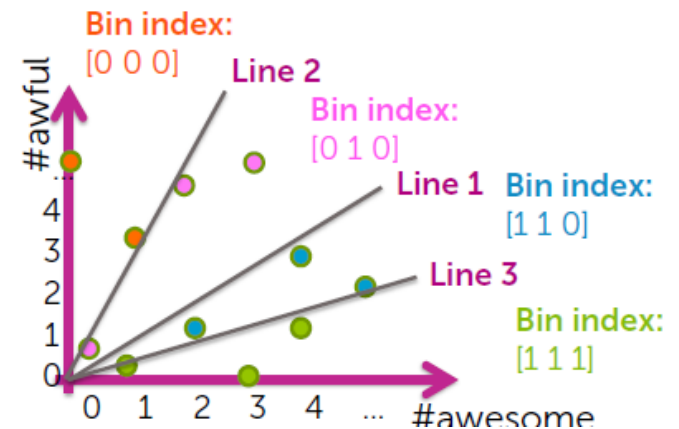
## Improving search quality by searching neighboring bins

Bin	$[0\ 0\ 0]$ = 0	$[0\ 0\ 1]$ = 1	$[0\ 1\ 0]$ = 2	$[0\ 1\ 1]$ = 3	$[1\ 0\ 0]$ = 4	$[1\ 0\ 1]$ = 5	$[1\ 1\ 0]$ = 6	$[1\ 1\ 1]$ = 7
Data indices:	{1,2}	--	{4,8,11}	--	--	--	{7,9,10}	{3,5,6}

Quality of retrieved NN can only improve with searching more bins

### Algorithm:

Continue searching until computational budget is reached or quality of NN good enough



105

# LSH recap

85

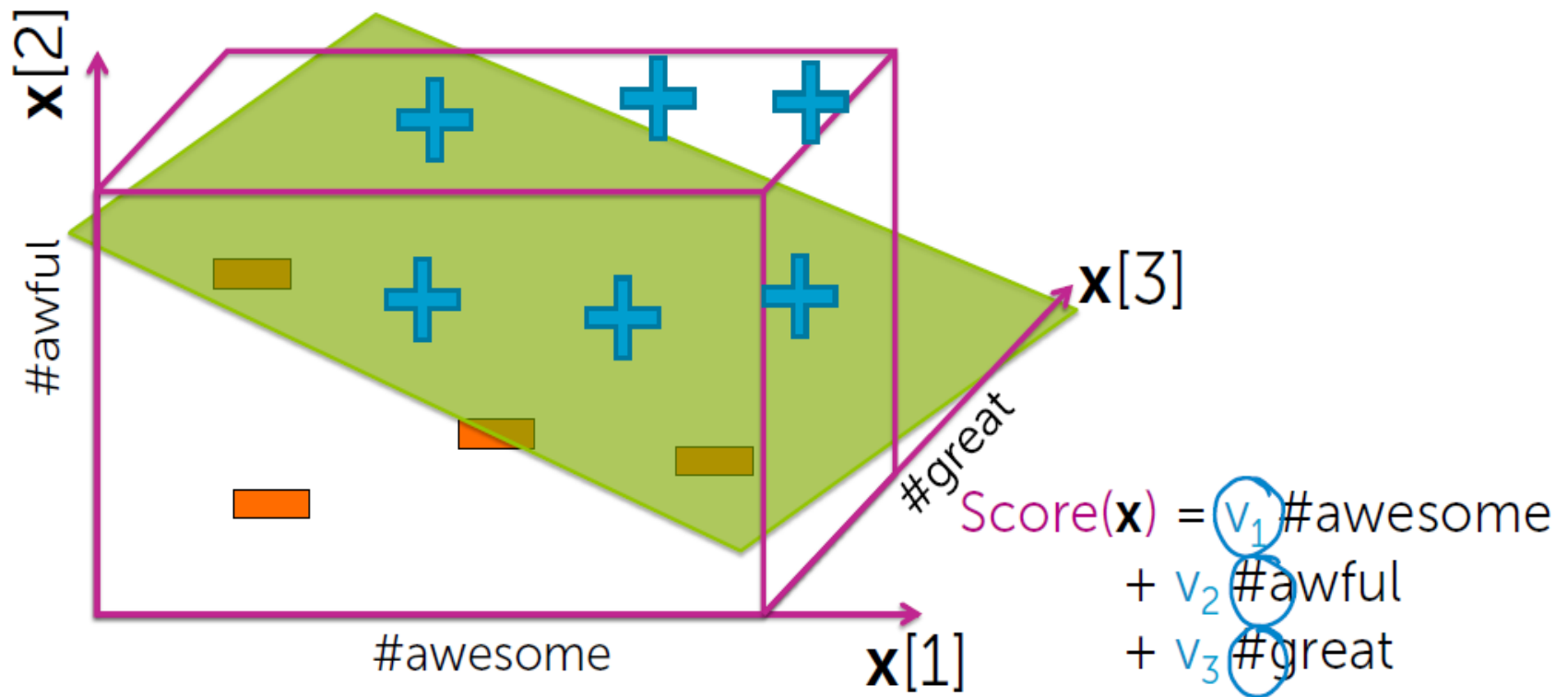
kd-tree competitor  
data structure

- Draw  $h$  random lines
  - Compute “score” for each point under each line and translate to binary index
  - Use  $h$ -bit binary vector per data point as bin index
  - Create hash table
- 
- For each query point  $\mathbf{x}$ , search  $\text{bin}(\mathbf{x})$ , then neighboring bins until time limit

# LSH: moving to higher dimensions $d$

86

## Draw random *planes*



# LSH: moving to higher dimensions $d$

87

## Cost of binning points in $d$ -dim

$$\text{Score}(\mathbf{x}) = v_1^{(i)} \# \text{awesome} \\ + v_2^{(i)} \# \text{awful} \\ + v_3^{(i)} \# \text{great}$$

*$i^{\text{th}}$  hyperplane*

Per data point,  
need  $d$  multiplies  
to determine bin  
index per plane

*In high-dim, (and some applications)  
this is often a sparse mult.*

One-time cost offset if many  
queries of fixed dataset

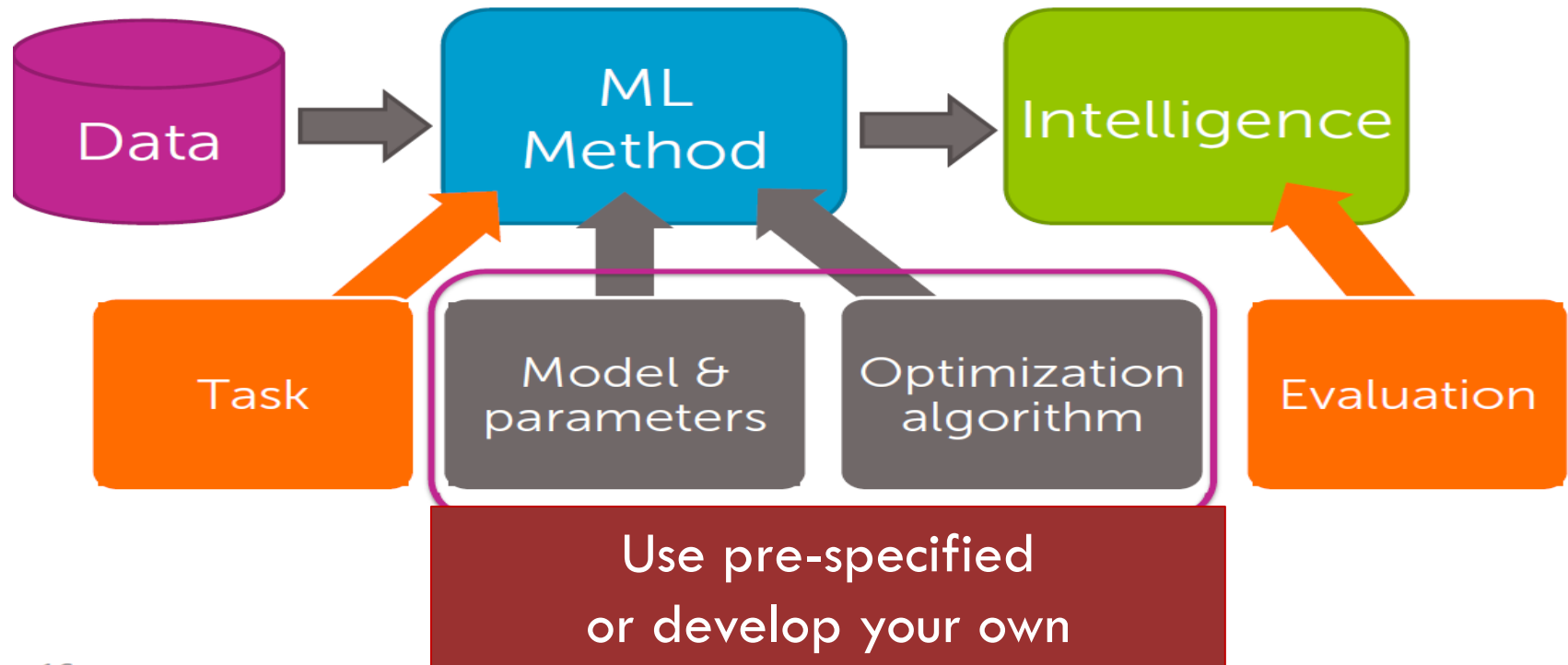
# Wrapping up



# Deploying intelligence module

89

**Case studied are about building, evaluating, deploying intelligence in data analysis.**



# Prediction: Predicting house prices

90

## Models

- Linear regression
- Regularization: Ridge (L2), Lasso (L1)

## Algorithms

- Gradient descent
- Coordinate descent

## Concepts

- Loss functions, bias-variance tradeoff, cross-validation, sparsity, overfitting, model selection

# Classification: Sentiment analysis

91

## Models

- Linear classifiers (logistic regression, SVMs, perceptron)
- Kernels
- Decision trees

## Algorithms

- Stochastic gradient descent
- Boosting

## Concepts

- Decision boundaries, MLE, ensemble methods, random forests, CART, online learning

# Clustering & Retrieval: Finding documents

92

## Models

- Nearest neighbors
- Clustering, mixtures of Gaussians
- Latent Dirichlet allocation (LDA)

## Algorithms

- KD-trees, locality-sensitive hashing (LSH)
- K-means
- Expectation-maximization (EM)

## Concepts

- Distance metrics, approximation algorithms, hashing, sampling algorithms, scaling up with map-reduce