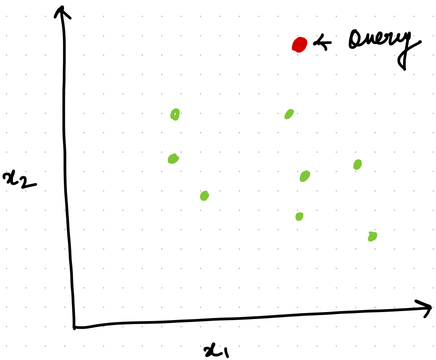
K-Nearest Neighbors Approximation

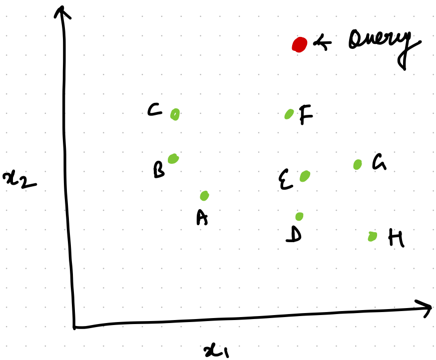
Nipun Batra

IIT Gandhinagar

August 2, 2025

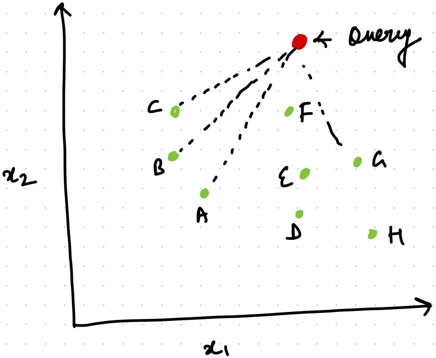


Find 1- NN for query point 'q' $\in R^D$



Find 1-NN for query point 'q' $\in R^D$

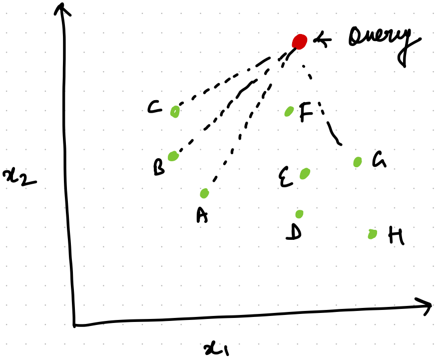
TRAIN SET is $X \in R^{N \times D}$



	DISTANCE
q - A	..
q - B	..
	..
	..
	..
	..
	..
	..

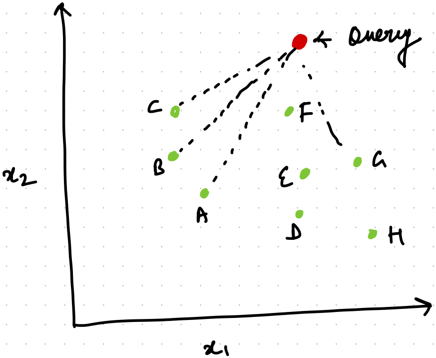
Find 1- NN for query point 'q' $\in R^D$

TRAIN SET is $X \in R^{N \times D}$



	DISTANCE
q - A	..
q - B	..
	..
	..
	..
	..
	..
	..

STEPS FOR FINDING $D(q, A)$



	DISTANCE
q - A	..
q - B	..
	..
	..
	..
	..
	..
	..

STEPS FOR FINDING $D(q, A)$

$$D(q, A) = \sqrt{(q_1 - A_1)^2 + \dots (q_D - A_D)^2}$$

STEPS FOR FINDING $D(q, A)$

$$D(q, A) = \sqrt{(q_1 - A_1)^2 + \dots (q_D - A_D)^2}$$

\# SUBTRACTIONS

\# MULTIPLICATIONS

\# ADDITIONS

\# SQRT

STEPS FOR FINDING $D(q, A)$

$$D(q, A) = \sqrt{(q_1 - A_1)^2 + \dots (q_D - A_D)^2}$$

# SUBTRACTIONS	D
# MULTIPLICATIONS	D
# ADDITIONS	D
# SQRT	1

STEPS FOR FINDING $D(q, A)$

$$D(q,A) = \sqrt{(q_1 - A_1)^2 + \dots (q_D - A_D)^2}$$

# SUBTRACTIONS	D
# MULTIPLICATIONS	D
# ADDITIONS	D
# SQRT	1

Total time for $D(q, A)$ or DISTANCE B/W 1 PAIR $= O(D)$

Q) Time required for creating table

	DISTANCE
q-A	..
q-B	..
	..
	..
	..
	..
	..
	..

Q) Time required for creating table

$$O(\underbrace{N}_{\#\text{ samples}}\ \underbrace{D}_{\text{Dimensionality}})$$

	DISTANCE
q-A	..
q-B	..
	..
	..
	..
	..
	..
	..

Q) Time required for finding 1- NN?

	DISTANCE
q-A	20
q-B	30
.	2
.	8
.	9
:	:
	:
	34

Q) Time required for finding 1-NN?

$O(N)$: linear search

	DISTANCE
q-A	20
q-B	30
.	2
.	8
.	9
.	..
	34

Q) Overall time complexity

$O(ND)$

Linear in # samples (sometimes millions of samples)

	DISTANCE
q-A	20
q-B	30
.	2
.	8
.	9
:	:
	.
	34

Goal:

Reduce $O(ND)$

Target

How?

Goal:

Reduce $O(ND)$

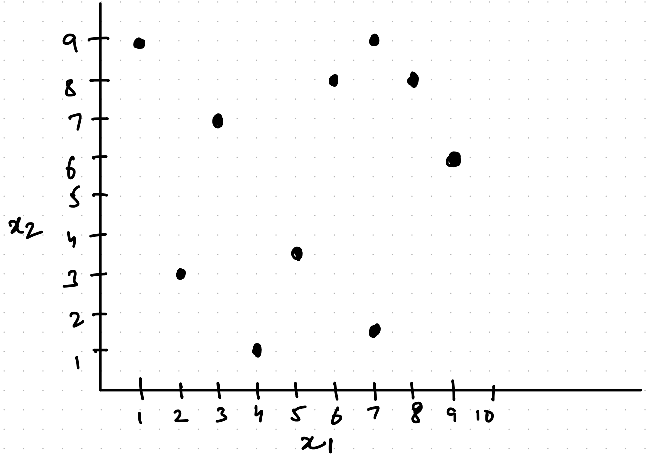
Target

How? (Hints)

- Decision trees
- Search for subset of examples
- current algorithm does <u>nothing</u> at training time.

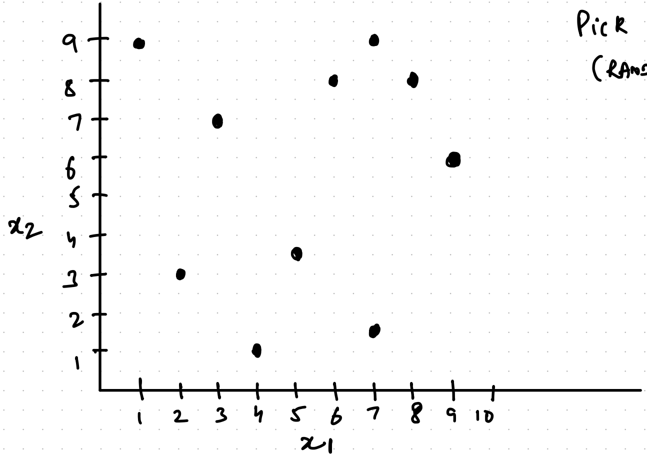
K-D trees (Victor Lavarenko slides)

- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$



K-D trees (Victor Lavrenko slides)

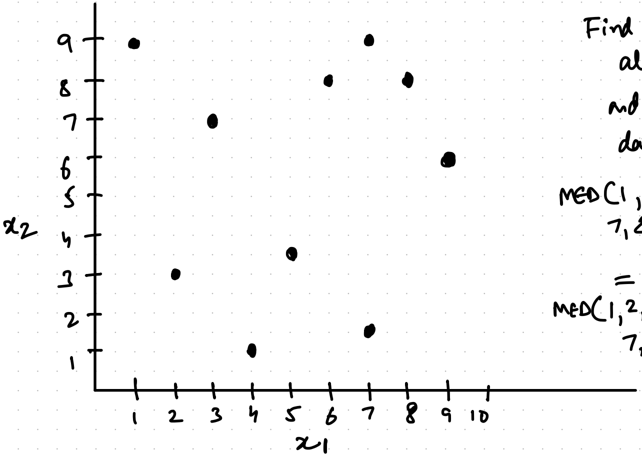
- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$



Pick dim x_1
(RANDOM$\{x_1, x_2\}$)

K-D trees (Victor Lavrenko slides)

- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$



Find median along x_1 and split data

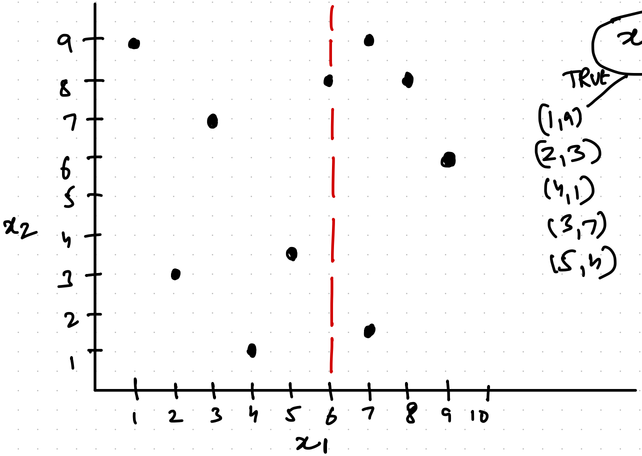
MED(1, 2, 4, 3, 5, 6, 7, 8, 7, 9)

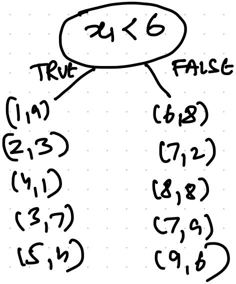
=

MED(1, 2, 3, 4, 5, <u>6</u>, 7, 7, 8, 9)

K-D trees (Victor Lavrenko slides)

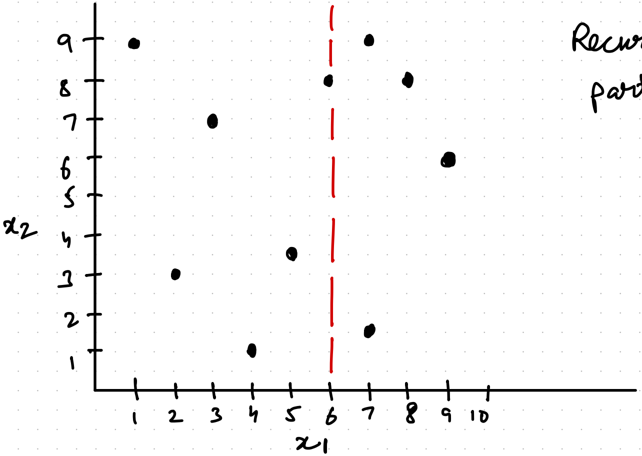
- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$





K-D trees (Victor Lavrenko slides)

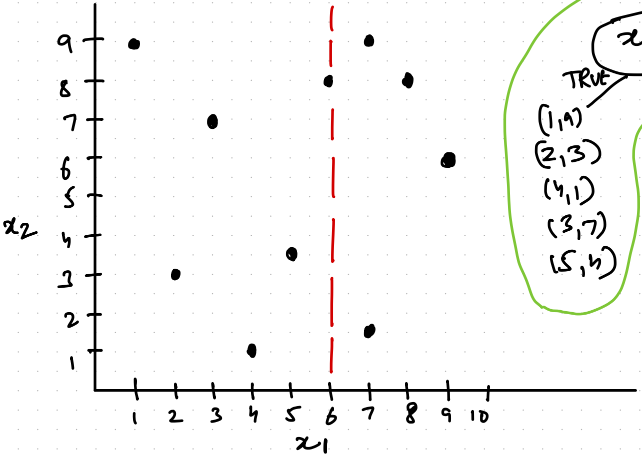
- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$



Recursively partition

K-D trees (Victor Lavrenko slides)

- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$



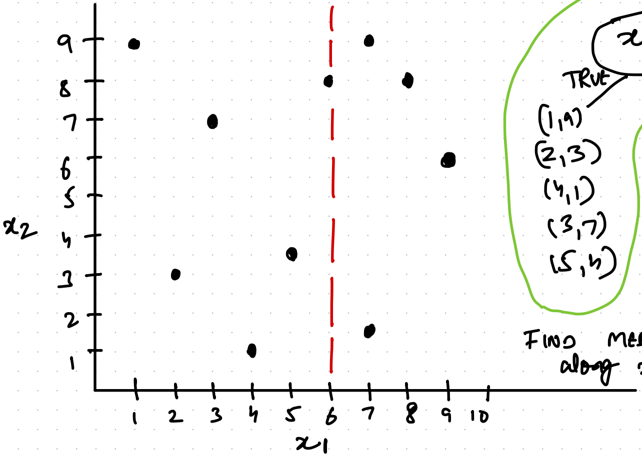
$x_1 < 6$

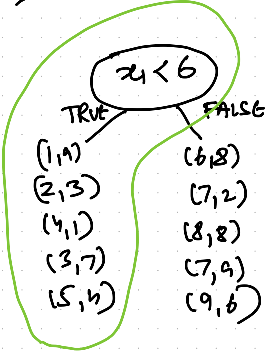
TRUE:
(1,9)
(2,3)
(4,1)
(3,7)
(5,4)

FALSE:
(6,8)
(7,2)
(8,8)
(7,9)
(9,6)

K-D trees (Victor Lavrenko slides)

- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$

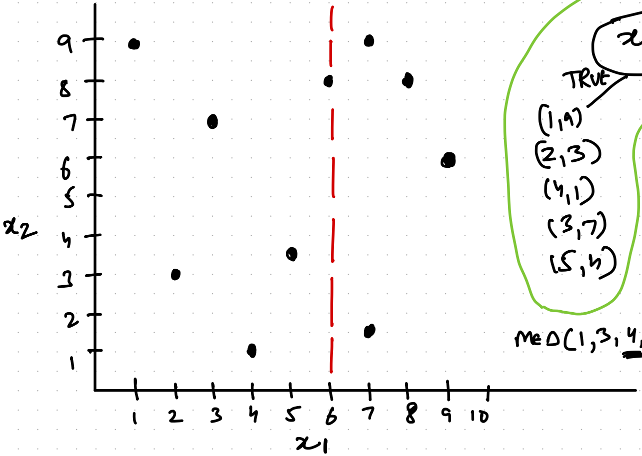


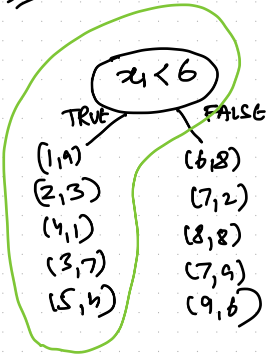


FIND MEDIAN along x_2

K-D trees (Victor Lavrenko slides)

- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$

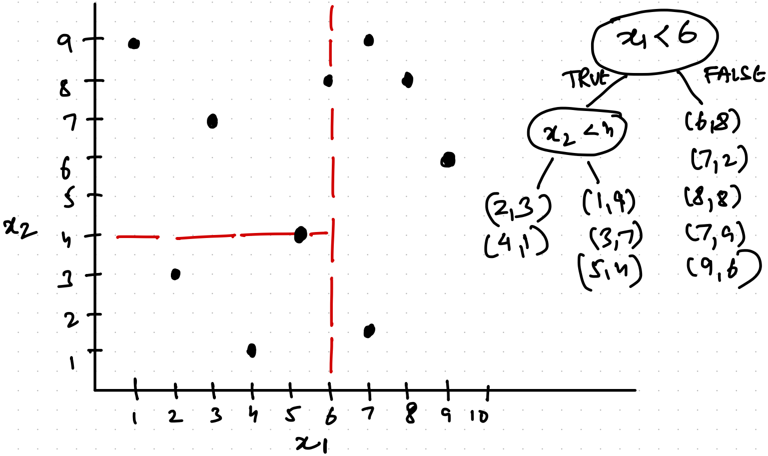




MED(1,3,4,7,9)

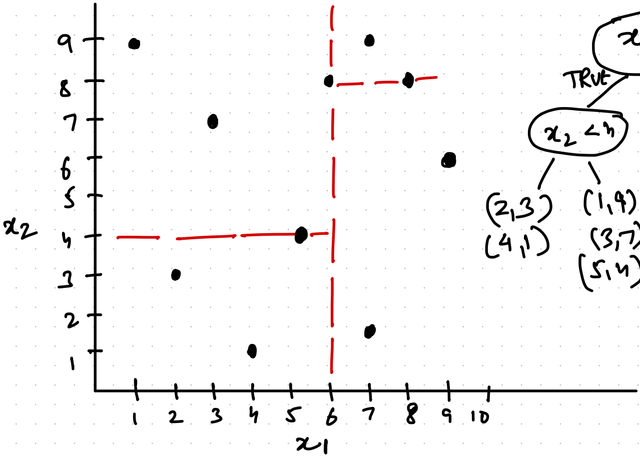
K-D trees (Victor Lavrenko slides)

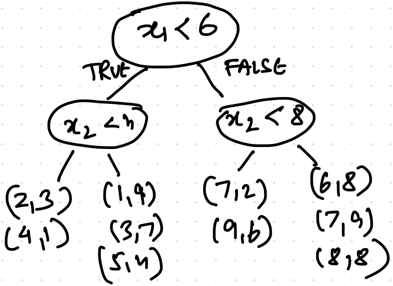
- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$



K-D trees (Victor Lavrenko slides)

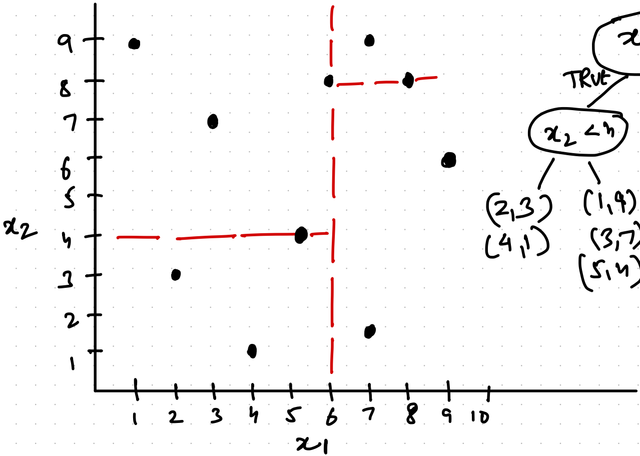
- $X = \{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)\}$

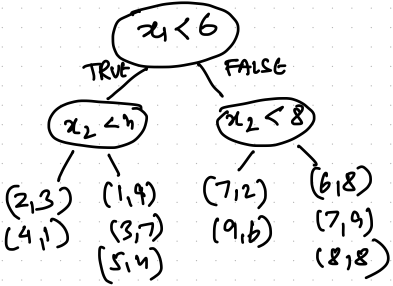




K-D trees (Victor Lavrenko slides)

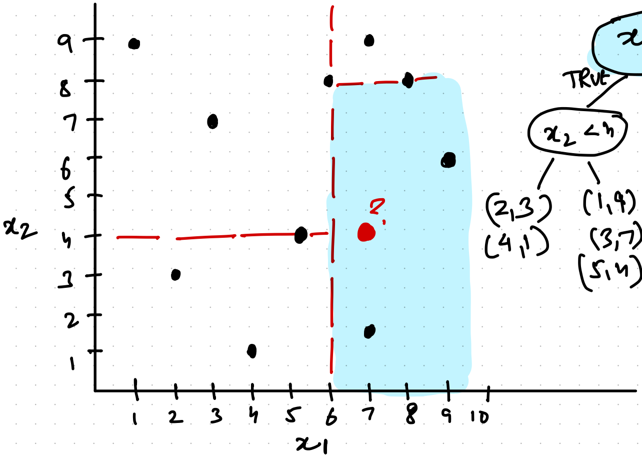
- query pt (7,4)

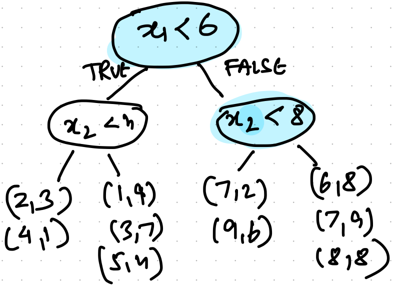




K-D trees (Victor Lavrenko slides)

- query pt (7, 4)

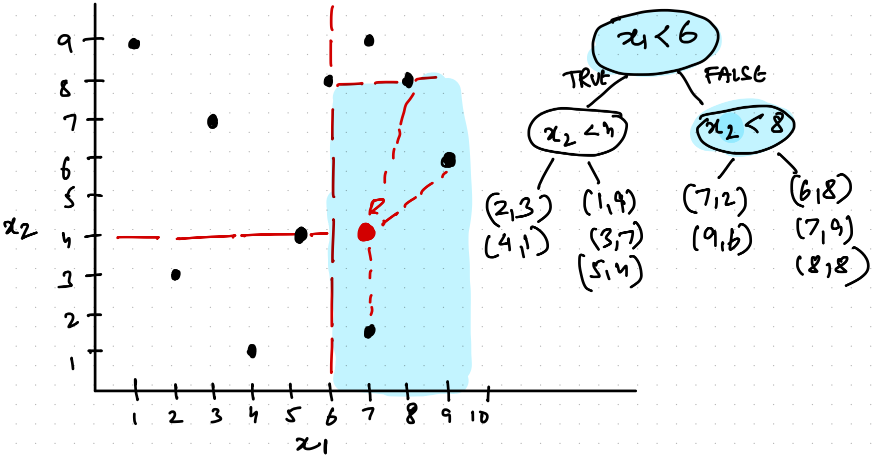




K-D trees (Victor Lavrenko slides)

- query pt (7,4)

FOR finding NNs, look in subspace

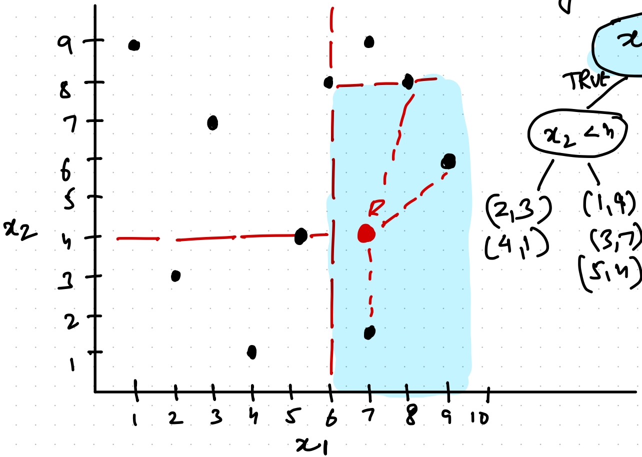


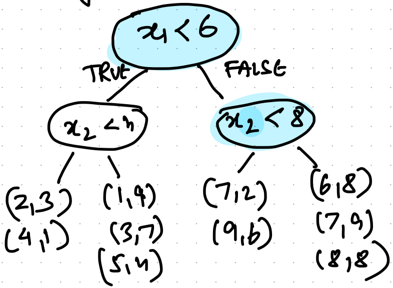
K-D trees (Victor Lavrenko slides)

- query pt (7,4)

For finding NNs, look in subspace

✓ Way lesser comparisons





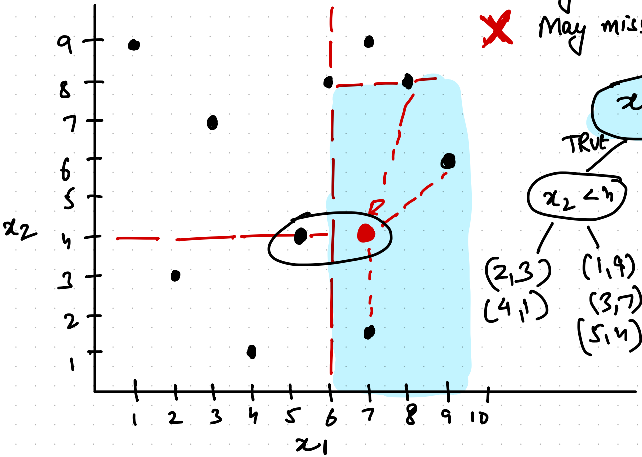
K-D trees (Victor Lavrenko slides)

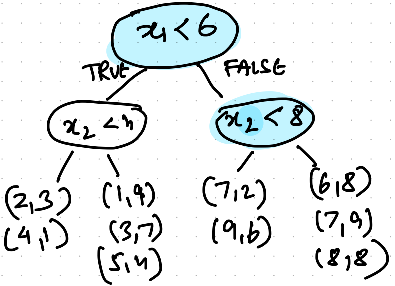
- query pt (7,4)

For finding NNs, look in subspace

✓ Way lesser comparisons

✗ May miss closest neighbors





KD trees (time complexity)

TRAINING Time (Ref. Wikipedia KD trees)

KD trees (time complexity)

TRAINING Time

samples in subtrees

KD trees (time complexity)

TRAINING Time

samples in subtrees

Depth 0

|

N

KD trees (time complexity)

TRAINING Time

samples in subtrees

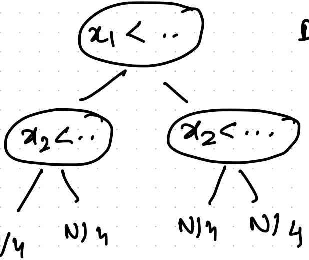
$x_1 < \ldots$ Depth)

$\frac{N}{2}$ $\frac{N}{2}$

KD trees (time complexity)

TRAINING Time

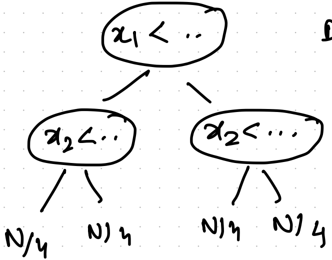
samples in subtrees



Depth 2

KD trees (time complexity)

TRAINING Time



Depth 2

Depth $O(\log_2 N)$

KD trees (time complexity)

TRAINING Time

- we have $O(\log_2 N)$ levels.
- For each level,
  - sort "N" examples to find median.
  - $O(N \log_2 N)$ time

- Total time to build datastructure $O(N \log^2 N)$

KD trees (time complexity)

* let's assume 'S' (≈ 40) is leaf size
(# samples at which
we stop partitioning)

* what is time complexity of query?

KD trees (time complexity)

* let's assume 'S' (≈ 40) is leaf size
(# samples at which we stop partitioning)

* what is time complexity of query?
— How many levels to reach leaf?

KD trees (time complexity)

* let's assume 'S' (≈ 40) is leaf size
(# samples at which we stop partitioning)

* what is time complexity of query?
  - How many levels to reach leaf?
    - $O(\log N)$
  - How many computat's per level?

KD trees (time complexity)

* let's assume 'S' (≈ 40) is leaf size
(# samples at which we stop partitioning)

* what is time complexity of query?
  - How many levels to reach leaf?
    - $O(\log N)$
  - How many computat's per level?
    - one 1-D comparison (e.g. $x_1 < t$)

KD trees (time complexity)

* let's assume 'S' (≈ 40) is leaf size
(# samples at which we stop partitioning)

* what is time complexity of query?
  - How many levels to reach leaf?
    - $O(\log N)$
  - How many computat's per level?
    - one 1-D comparison (e.g. $x_1 < t$)
  - How many computat's at leaf?

KD trees (time complexity)

* let's assume 'S' (≈ 40) is leaf size
(# samples at which we stop partitioning)

* what is time complexity of query?

  - How many levels to reach leaf?
    - $O(\log N)$
  - How many computat's per level?
    - one 1-D comparison (e.g. $x_1 < \theta$)
  - How many computat's at leaf?
    - $O(D * S)$

KD trees (time complexity)

* what is time complexity of query?

- How many levels to reach leaf?
  - $O(\log N)$
- How many computat's per level?
  - one 1-D comparison (e.g. $x_1 < \theta$)
- How many computat's at leaf?
  - $O(D * S)$

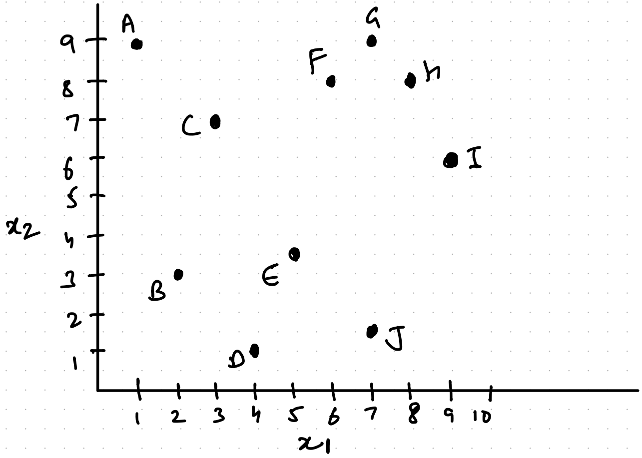
TOTAL QUERY $O(\log N + D * S)$

if $D << N$ and S is small

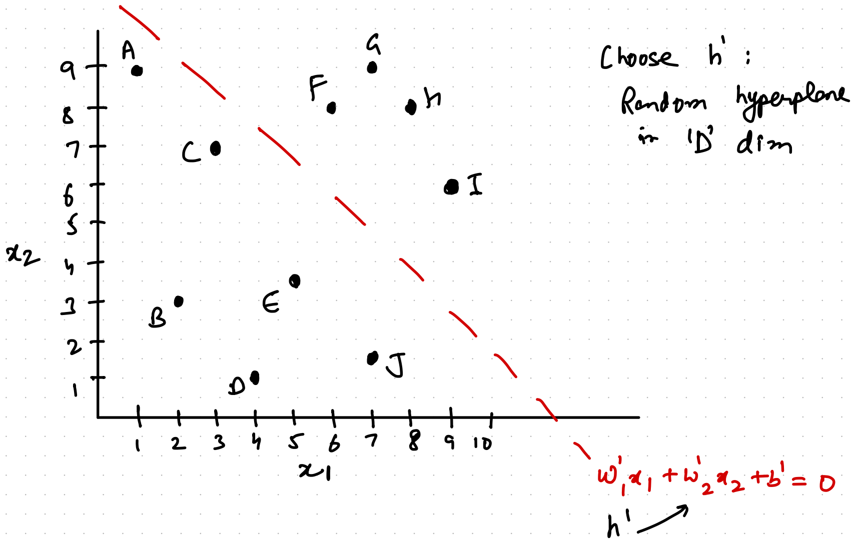
$O(\log N)$

Locality Sensitive Hashing (LSH) w/ Random Projection

(MachineLearningInterview.com)

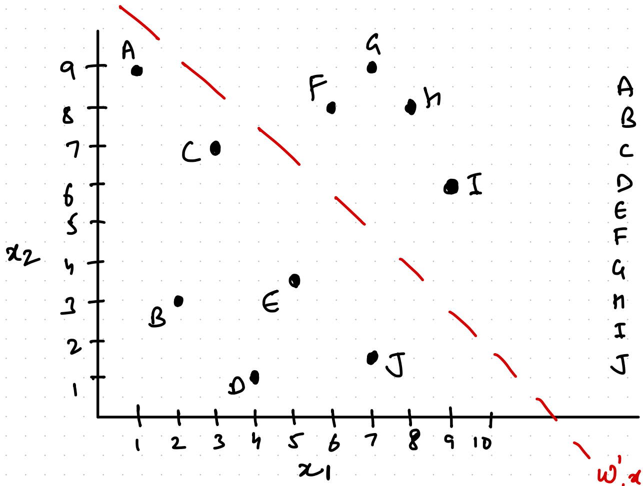


Locality Sensitive Hashing (LSH) w/ Random Projection



Choose h':
Random hyperplane in 'D' dim

Locality Sensitive Hashing (LSH) w/ Random Projection

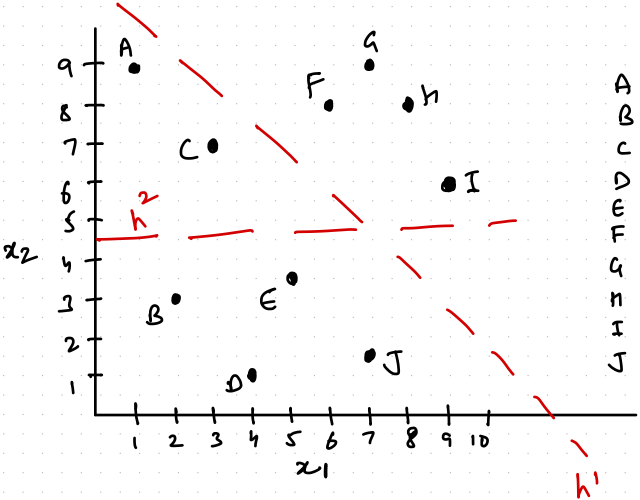


	h'
A	0
B	0
C	0
D	0
E	0
F	1
G	1
H	1
I	1
J	0

$$w'_1 x_1 + w'_2 x_2 + b' = 0$$

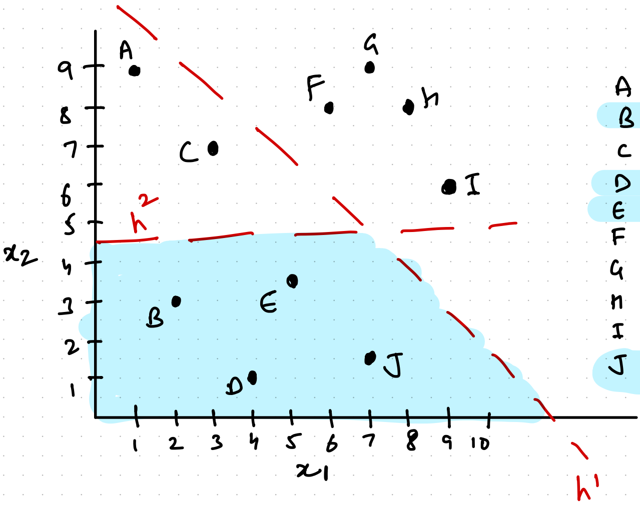
h'

Locality Sensitive Hashing (LSH) w/ Random Projection



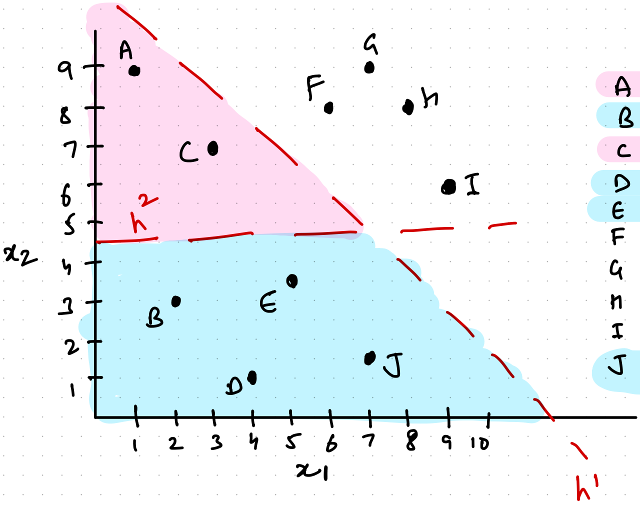
	h^2	h^1
A	1	0
B	0	0
C	1	0
D	0	0
E	0	0
F	1	1
G	1	1
H	1	1
I	1	1
J	0	0

Locality Sensitive Hashing (LSH) w/ Random Projection



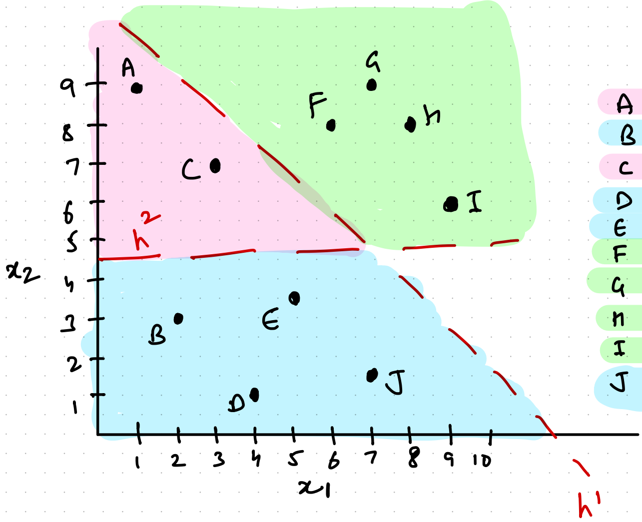
	h^2	h^1
A	1	0
B	0	0
C	1	0
D	0	0
E	0	0
F	1	1
G	1	1
H	1	1
I	1	1
J	0	0

Locality Sensitive Hashing (LSH) w/ Random Projection



	h^2	h^1
A	1	0
B	0	0
C	1	0
D	0	0
E	0	0
F	1	1
G	1	1
H	1	1
I	1	1
J	0	0

Locality Sensitive Hashing (LSH) w/ Random Projection



	h^2	h^1
A	1	0
B	0	0
C	1	0
D	0	0
E	0	0
F	1	1
G	1	1
H	1	1
I	1	1
J	0	0

Practical implementat^n

* How to sample hyperplane?

$i = 1 \ldots D$: sampled from

$w_i \sim N(.., ...)$

$b \sim N(..., ...)$

Practical implementatⁿ

* How to get $h^i(x) = 0$ or 1

$$h^i(x) = \text{SIGN}\left(b + w_1^i x_1 + w_2^i x_2 + \dots\dots\right)$$

Practical implementatⁿ

* How to vectorize finding Hash Table

$$X = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$$

Practical implementat^n

* How to vectorize finding Hash Table

$$X = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}_{N \times D}$$

$$X' = \begin{bmatrix} 1 & \\ 1 & \\ \vdots & \\ 1 & \end{bmatrix}_{N \times D+1}$$

Practical implementat^n

* How to vectorize finding Hash Table

$$x' = \begin{bmatrix} 1 & & \\ 1 & & \\ \vdots & & \\ 1 & & \end{bmatrix}_{N \times D+1}$$

let # planes be 'p'

$$W = \begin{bmatrix} - \; - \; - \; - \\ N(\ldots, \ldots) \\ - \; - \\ - \; - \quad - \; - \end{bmatrix}_{D+1 \times p}$$

Practical implementat^n

* How to vectorize finding Hash Table

$$x' = \begin{bmatrix} 1 & & \\ 1 & & \\ \vdots & & \\ 1 & & \end{bmatrix}_{N \times D+1} \qquad H = \operatorname{SGN}(x' W)_{N \times P}$$

let # planes be 'p'

$$W = \begin{bmatrix} - & - & - & - \\ & N(\ldots, \ldots) & & \\ - & - & & \\ - & - & - & - \end{bmatrix}_{D+1 \times P}$$

Time complexity

- Assuming one (1) set of hash functions

$\rightarrow$ $H_{N \times P} = SGN(X_{N \times D} \cdot R_{D \times P})$

Time $= O(N \times D \times P)$

Time to generate $R \approx O(D \times P)$

Thus, at training time: $O(N \times D \times P)$

At testing time,

→ Computing hash on $q_{1 \times D}$ takes $O(DP)$

→ Assuming 'T' points in bucket:

$O(T * D)$ to find closest neighbour

At testing time,

→ Computing hash on $q_{1 \times D}$ takes $O(DP)$

→ Assuming 'T' points in bucket:

$O(T * D)$ to find closest neighbour

→ What is T in terms of N and P (on average)

At testing time,

→ Computing hash on $q_{1 \times D}$ takes $O(DP)$

→ Assuming 'T' points in bucket:

$O(T * D)$ to find closest neighbour

→ What is T in terms of N and P (on average)

$$T \approx \frac{N}{2^P}$$

At testing time,

→ Computing hash on $q_{1 \times D}$ takes $O(DP)$

→ Assuming 'T' points in bucket:

$O(T * D)$ to find closest neighbour

→ What is T in terms of N and P (on average)

$$T \approx \frac{N}{2^P}$$

→ Test Time $= O\left(DP + \frac{DN}{2^P}\right)$