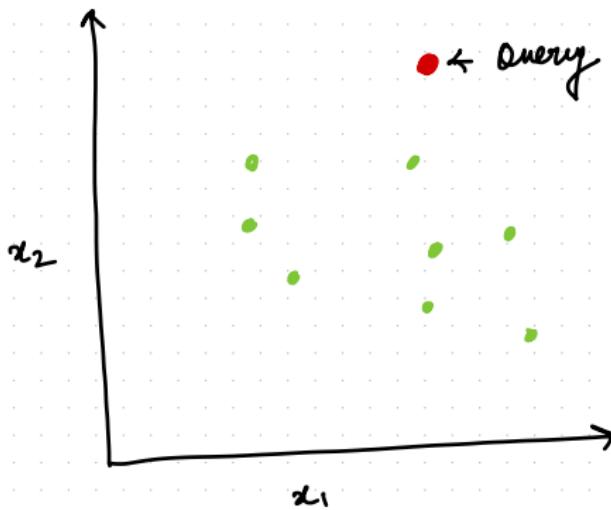


K-Nearest Neighbors Approximation

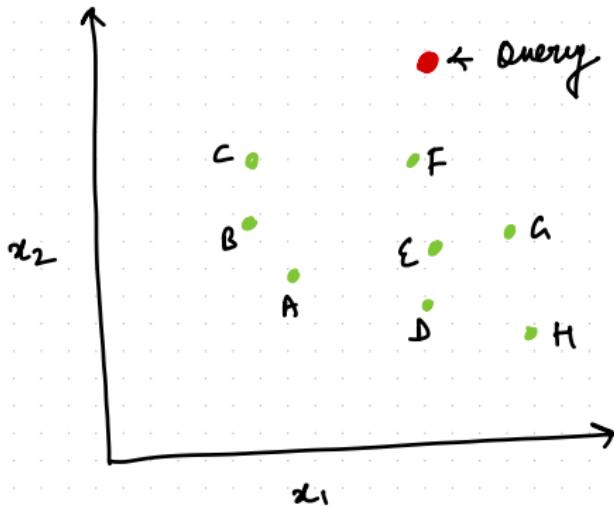
Nipun Batra

IIT Gandhinagar

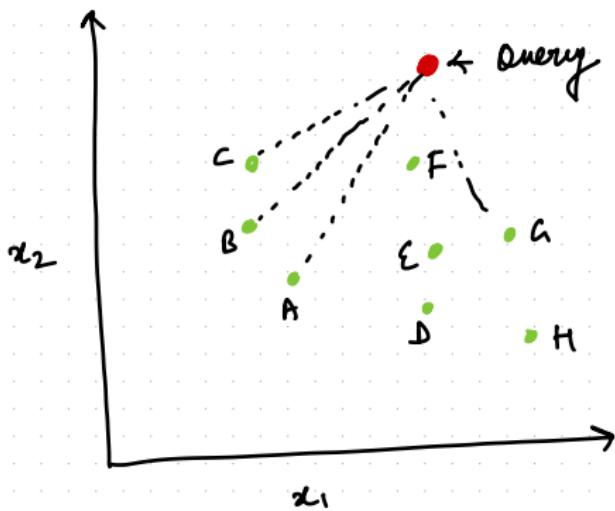
August 30, 2025



Find 1-NN for query point $\vec{q} \in \mathbb{R}^D$

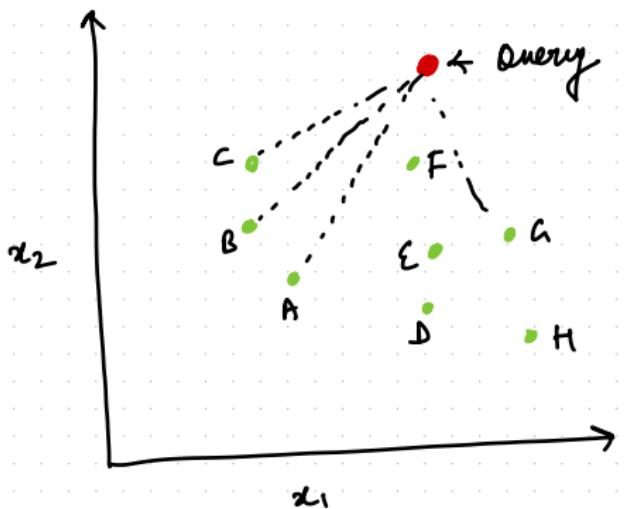


Find 1-NN for query point $\vec{q} \in \mathbb{R}^D$
 TRAIN SET is $X \in \mathbb{R}^{N \times D}$



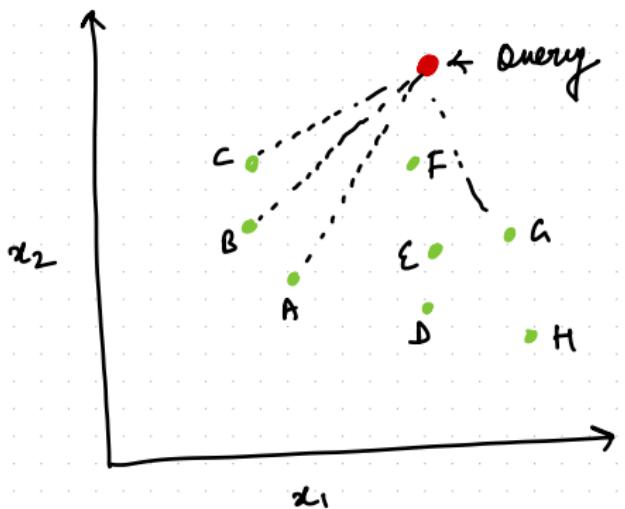
DISTANCE	
$q_r - A$	"
$q_r - B$	"
$q_r - C$	"
$q_r - D$	"
$q_r - E$	"
$q_r - F$	"
$q_r - G$	"
$q_r - H$	"

Find 1-NN for query point $\vec{q} \in \mathbb{R}^D$
 TRAIN SET is $X \in \mathbb{R}^{N \times D}$



DISTANCE	
$q_v - A$	"
$q_v - B$	"
$q_v - C$	"
$q_v - D$	"
$q_v - E$	"
$q_v - F$	"
$q_v - G$	"
$q_v - H$	"

STEPS FOR FINDING $D(q_v, A)$



DISTANCE

$q_V - A$	"
$q_V - B$	"
$q_V - C$	"
$q_V - D$	"
$q_V - E$	"
$q_V - F$	"
$q_V - G$	"
$q_V - H$	"

STEPS FOR FINDING $D(q_V, A)$

$$D(q_V, A) = \sqrt{(q_{V1} - A_1)^2 + \dots + (q_{VD} - A_D)^2}$$

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 $D(q_1, A)$

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

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

Q) Time required for creating table

Q) Time required for
finding 1-NN?

DISTANCE

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

Q) Time required for
finding 1-NN?

$O(n)$: linear search

DISTANCE

	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

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

Goal:

Reduce $O(N^D)$
↑
Target

How?

Goal:

Reduce $O(N^D)$
↑
Target

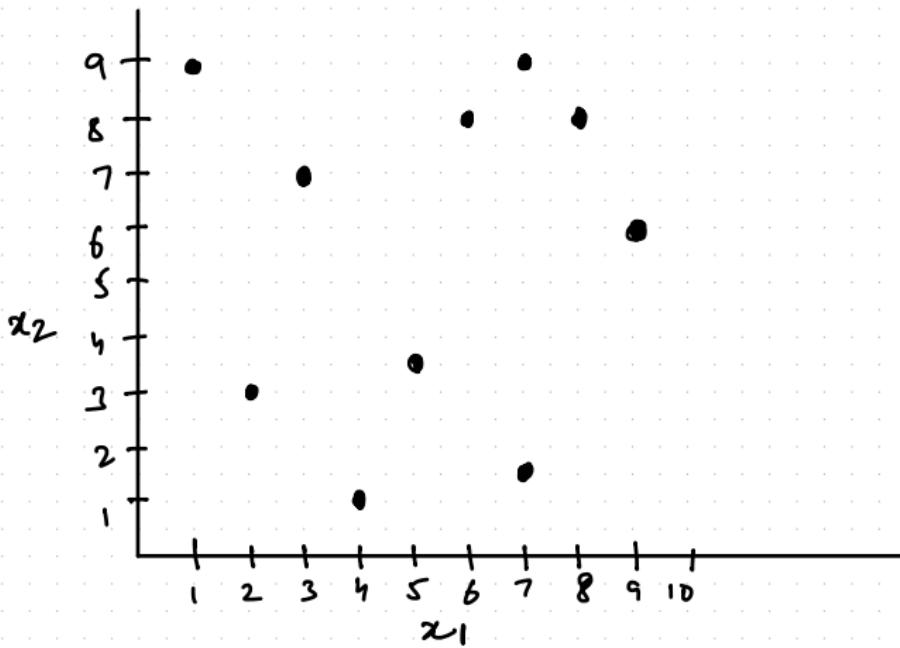
How? (Hints)

- Decision trees
- Search for subset of examples
- Current algorithm does nothing at training time.

K-D trees

(Victor Lawrence's slides)

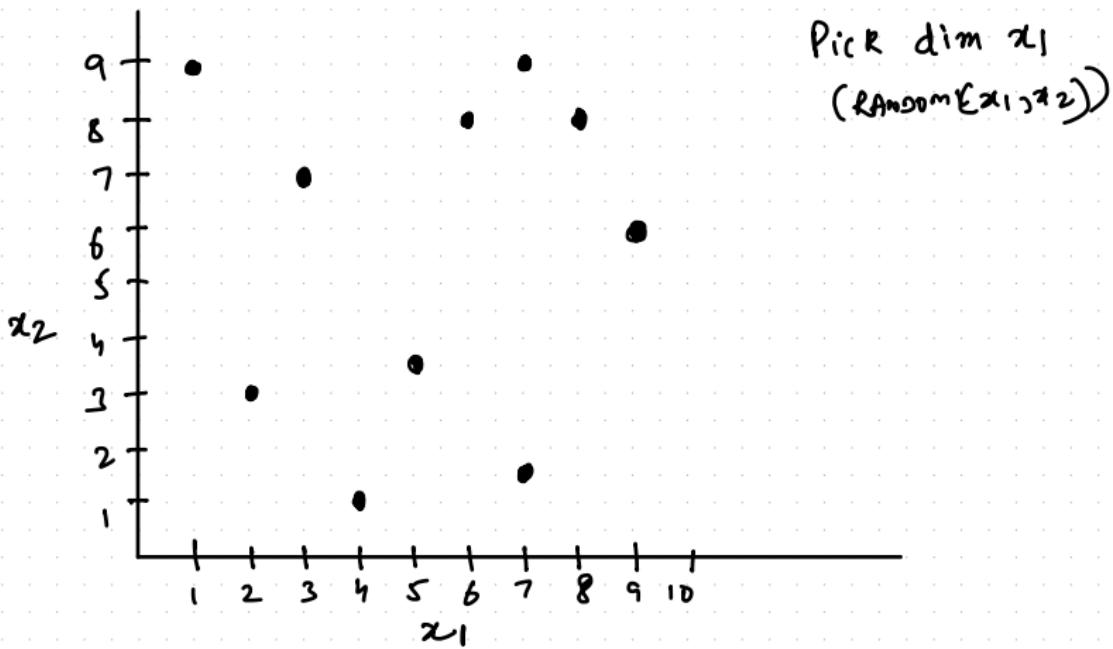
- $X = \{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$



K-D trees

(Victor Lawrence's slides)

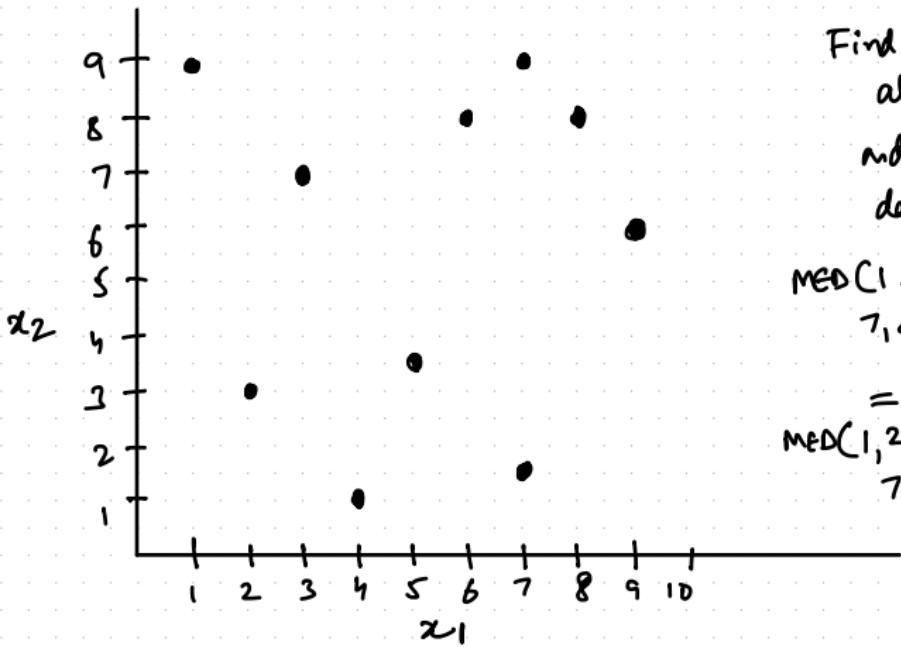
- $X = \{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$



K-D trees

(Victor Lamarceneo 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

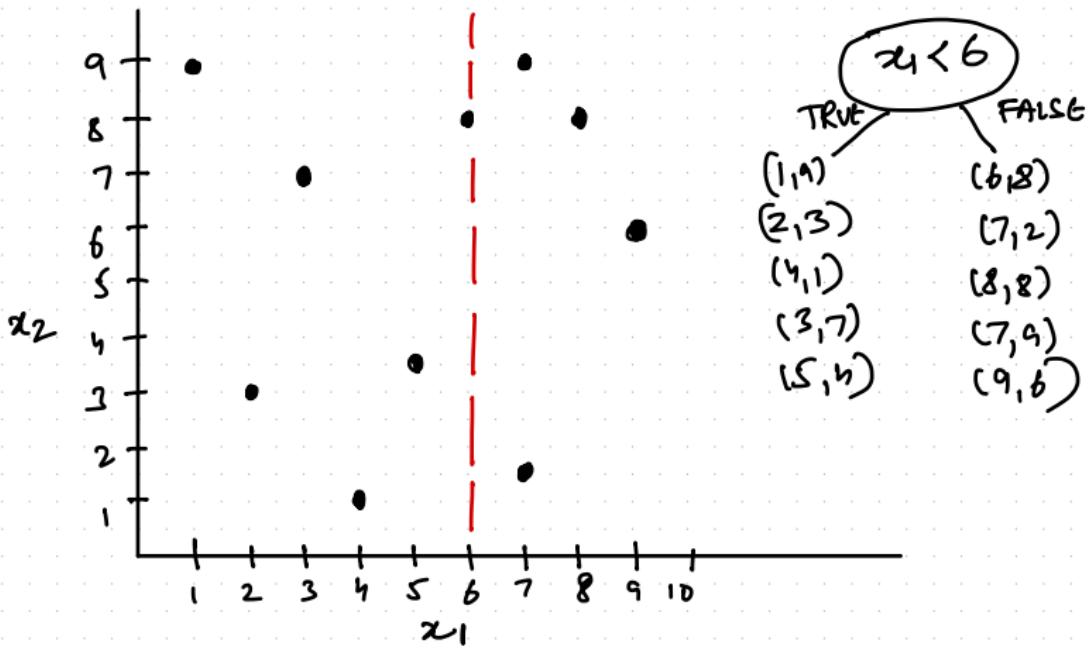
$$\text{MED}(1, 2, 4, 3, 5, 6, 7, 8, 7, 9)$$

$$= \\ \text{MED}(1, 2, 3, 4, 5, \underline{6}, 7, 7, 8, 9)$$

K-D trees

(Victor Lawrence's slides)

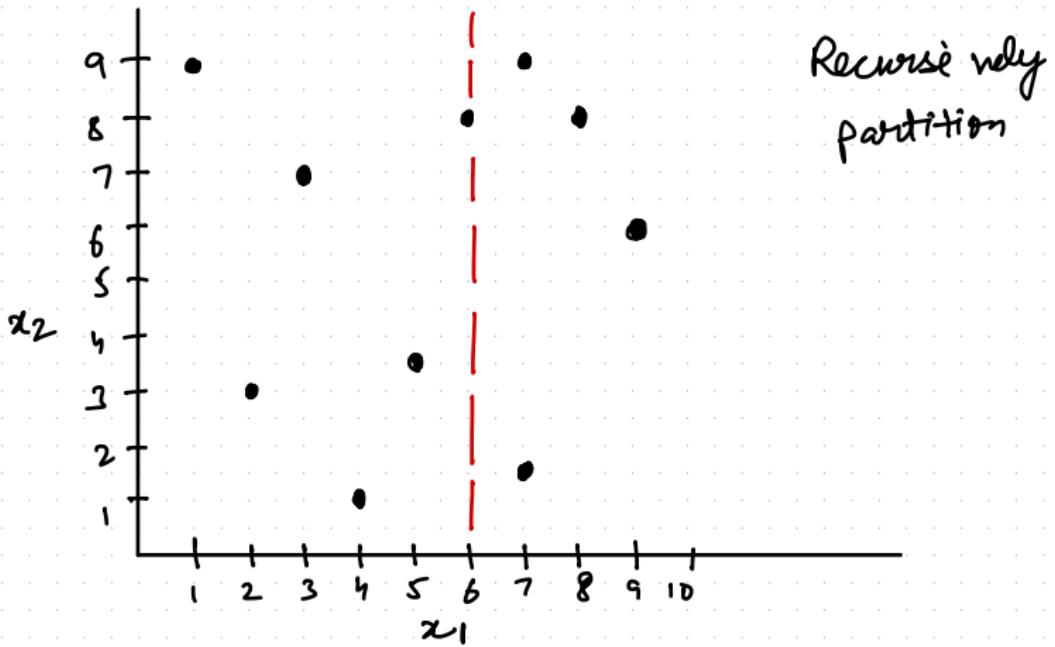
- $X = \{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$



K-D trees

(Victor Lawrence's slides)

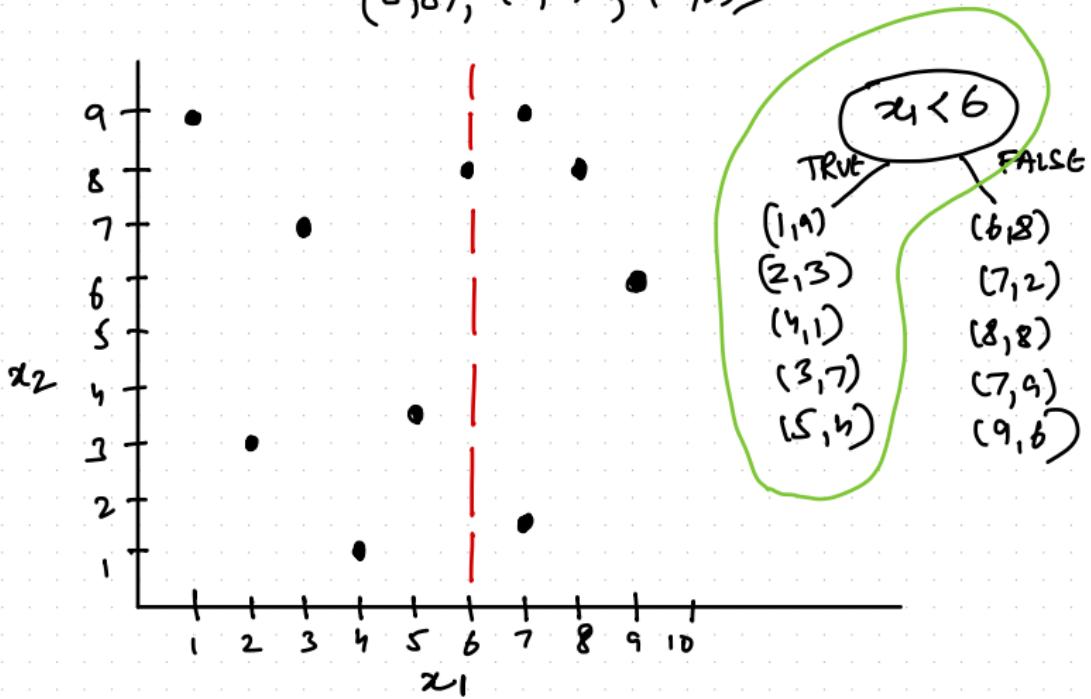
- $X = \{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$



K-D trees

(Victor Lamarcqo slides)

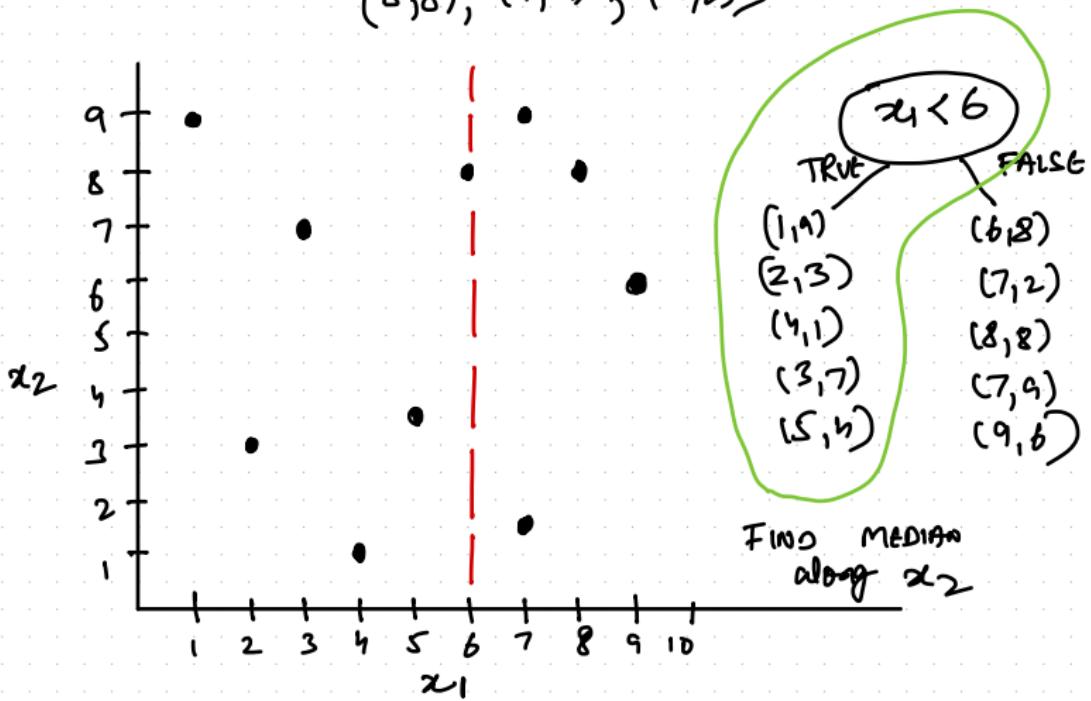
- $X = \{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$



K-D trees

(Victor Lamarcqo slides)

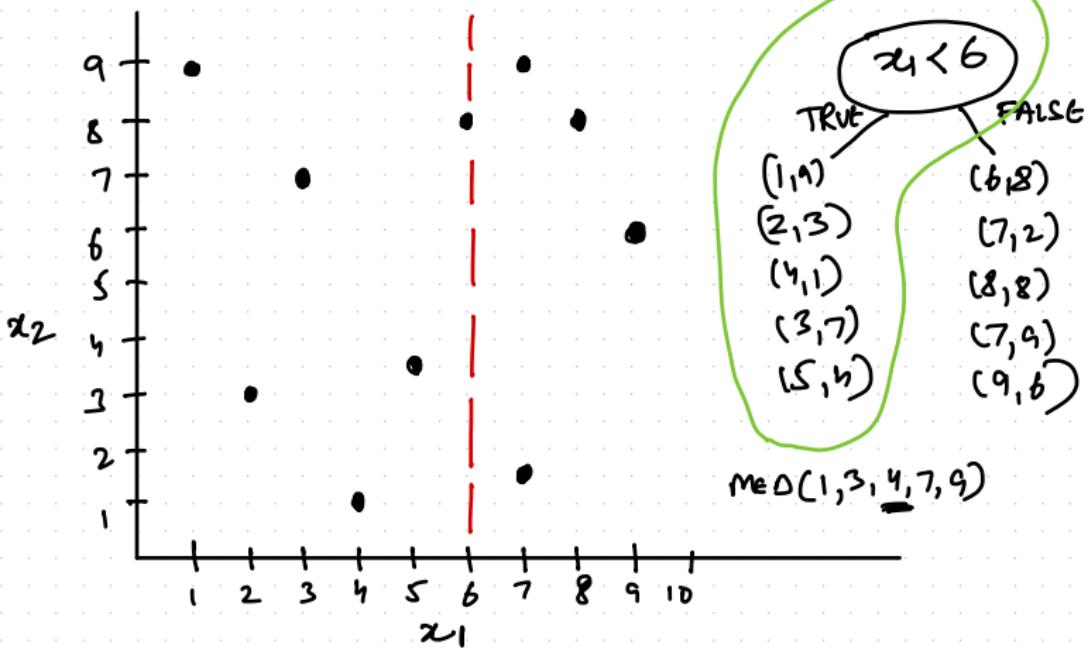
- $X = \{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$



K-D trees

(Victor Lamarcqo slides)

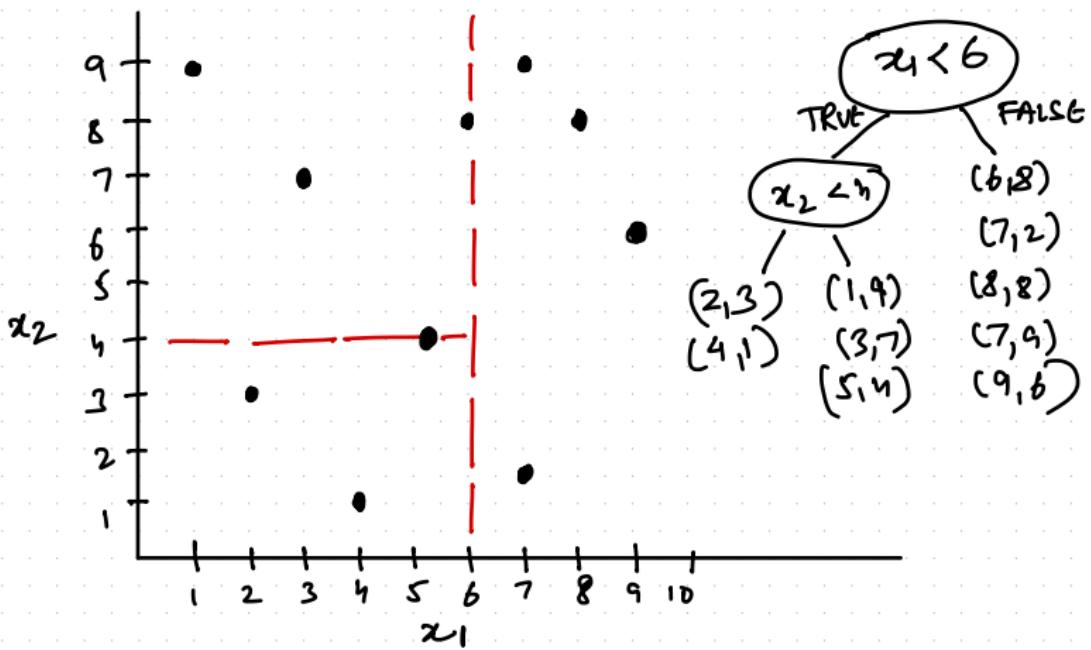
- $X = \{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$



K-D trees

(Victor Lamarcqo slides)

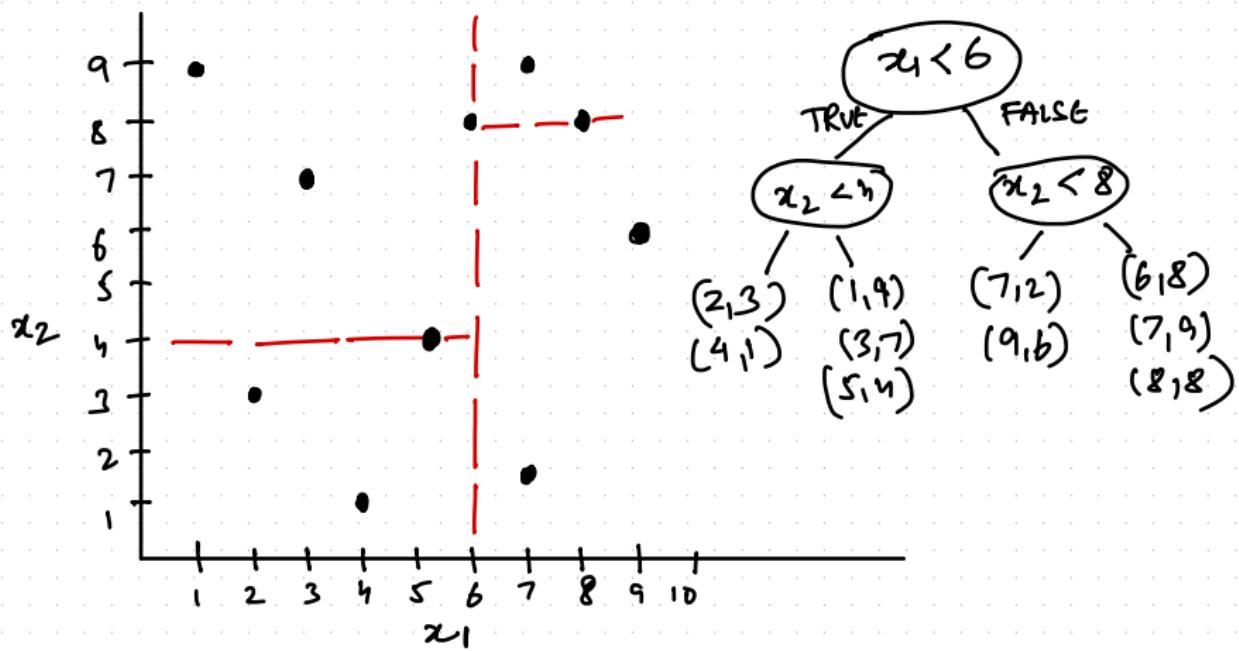
- $X = \{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$



K-D trees

(Victor Lamarceneo slides)

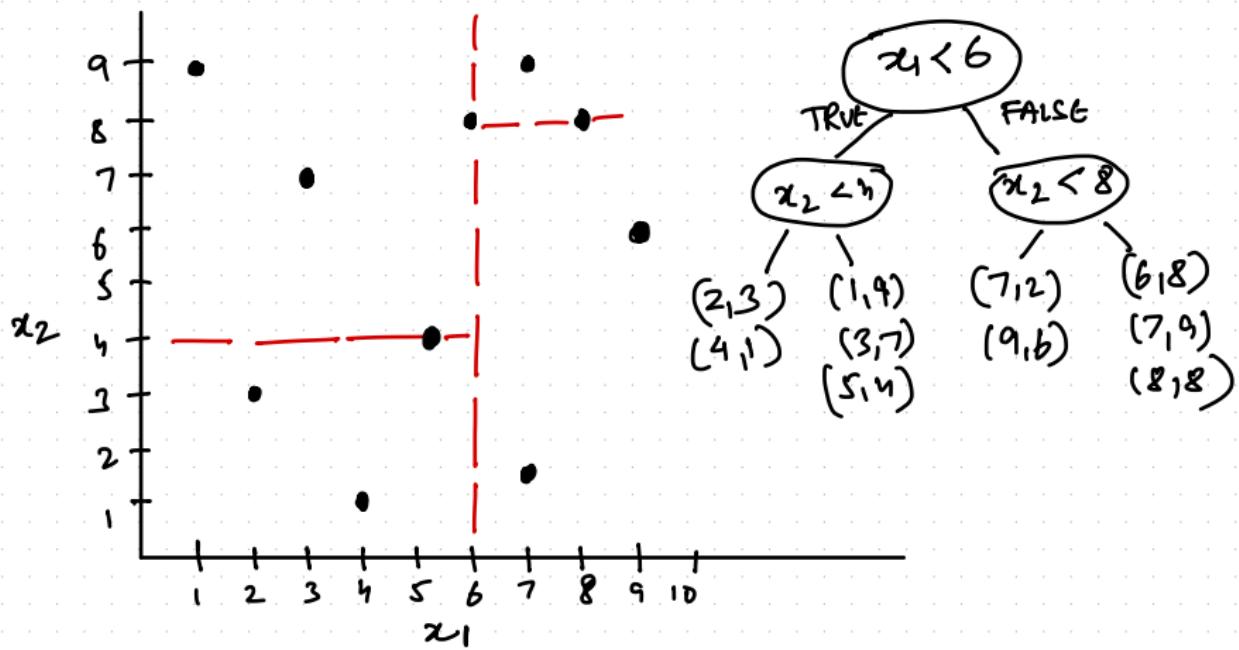
- $X = \{(1, 9), (2, 3), (4, 1), (3, 7), (5, 4), (6, 8), (7, 2), (8, 8), (7, 9), (9, 6)\}$



K-D trees

(Victor Lamarcq's slides)

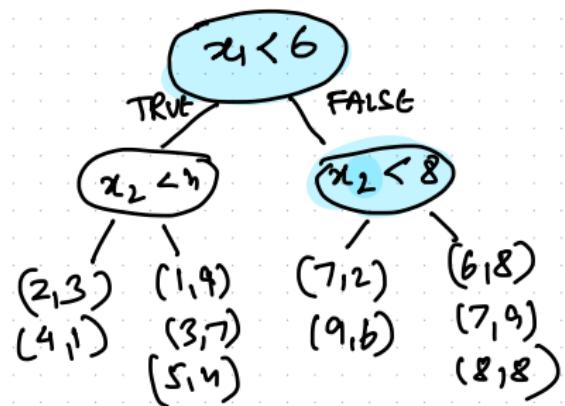
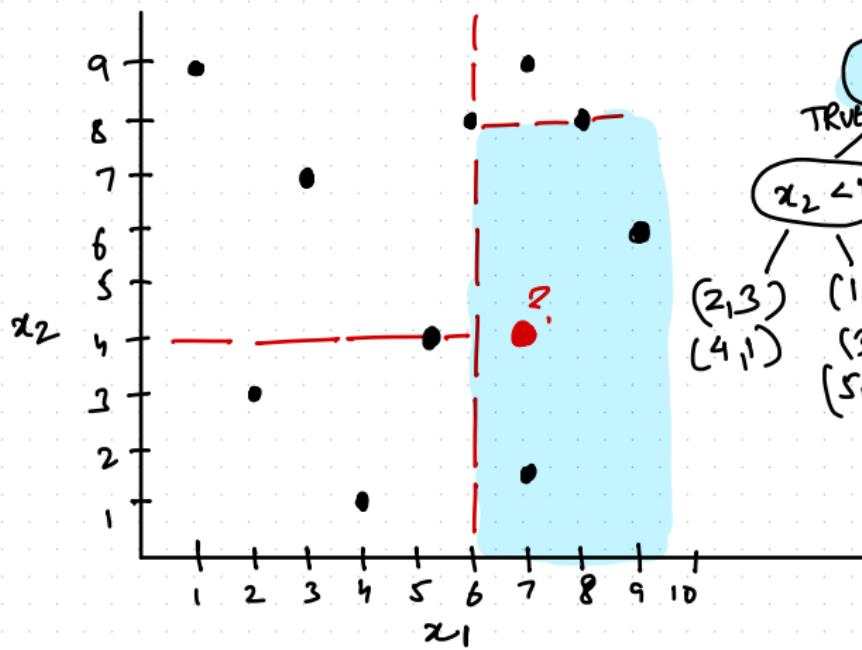
- query pt (7, 4)



K-D trees

(Victor Lamarcq's slides)

- query pt (7, 4)

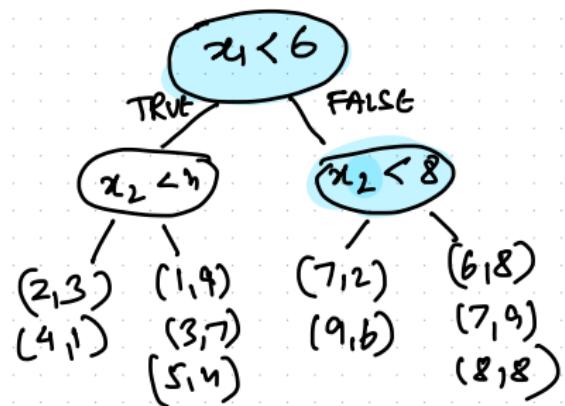
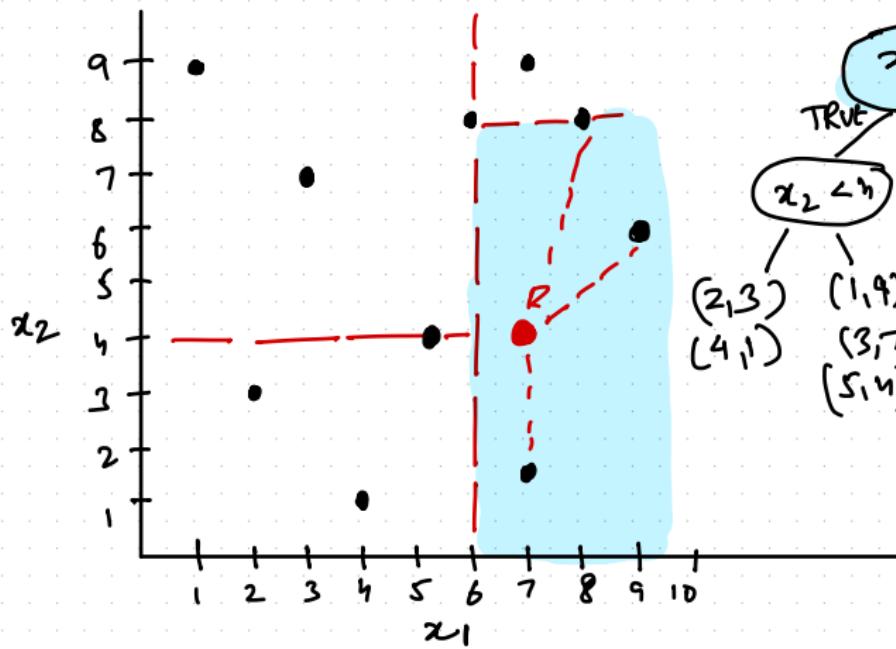


K-D trees

(Victor Lamarcq's slides)

- query pt (7, 4)

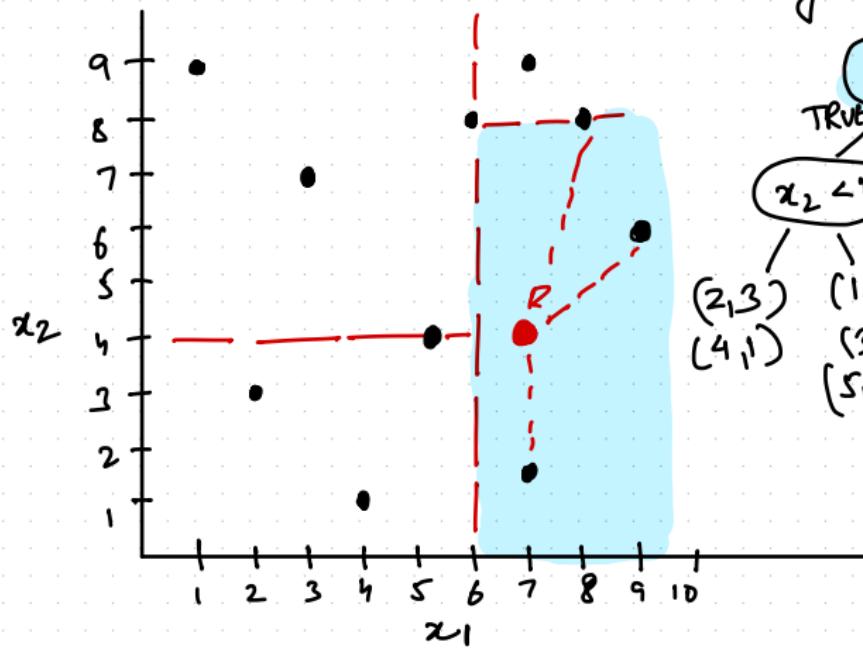
FOR Finding NBS, look
in subspace



K-D trees

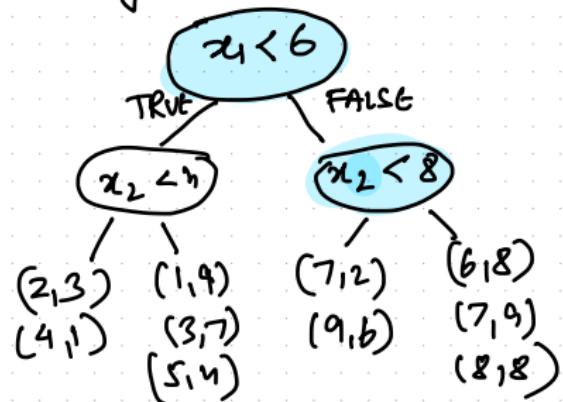
(Victor Lamarcq's slides)

- query pt (7, 4)



FOR Finding NNS, LOOK
in subspace

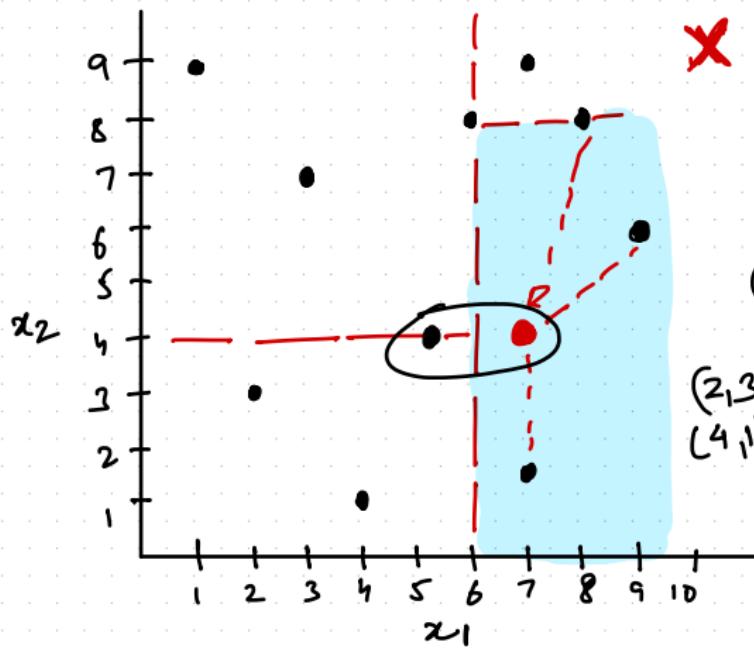
✓ Way lesser comparisons



K-D trees

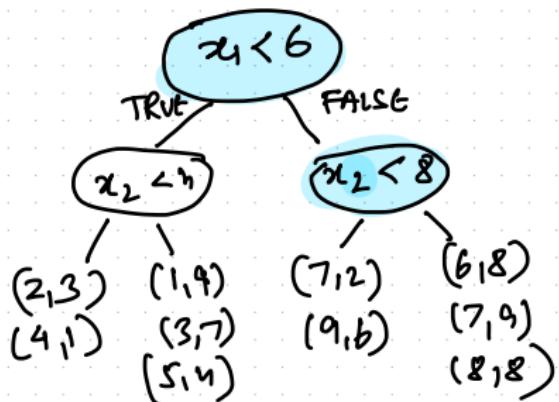
(Victor Lamarcq's slides)

- query pt (7, 4)



FOR Finding NNS, LOOK
in subspace

- ✓ Way fewer 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

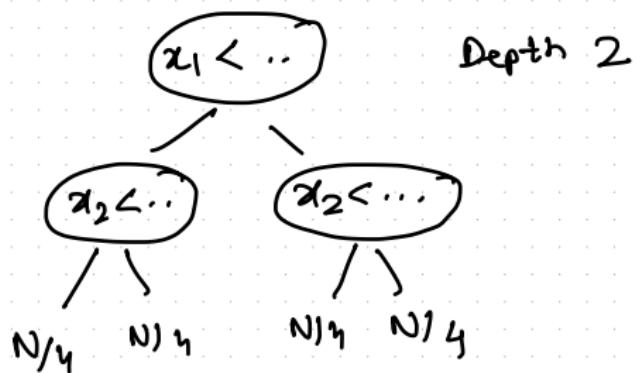


Depth 1

KD trees (time complexity)

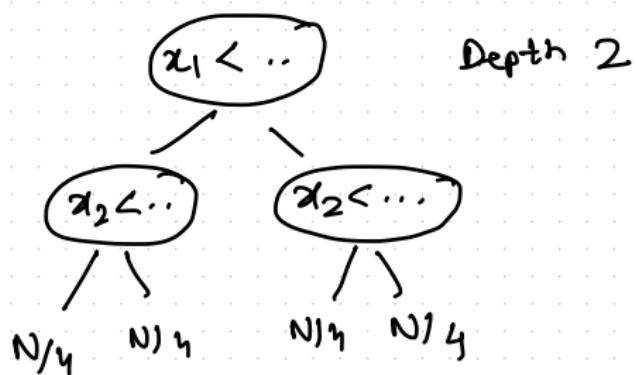
TRAINING TIME

samples in subtrees



KD trees (time complexity)

TRAINING TIME



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 computations 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 computations per level?
 - One $\rightarrow \Delta$ comparison (e.g. $x_1 < b$)

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 comparat's per level?
 - One $\rightarrow \Delta$ comparison (e.g. $x_1 < b$)
 - How many comparat'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 $\rightarrow \Delta$ comparison (e.g. $x_1 < b$)
 - 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 \rightarrow comparison (e.g., $x_1 < t$)

- How many computat's at leaf?

$$- O(D+S)$$

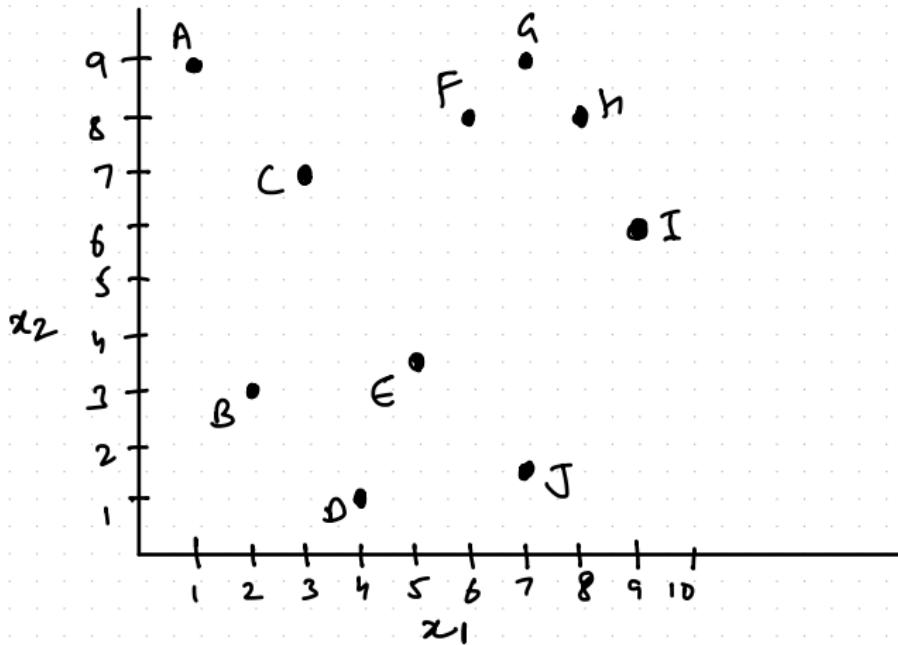
TOTAL QUERY $O(\log N + D+S)$

if $D \ll N$ and S is small

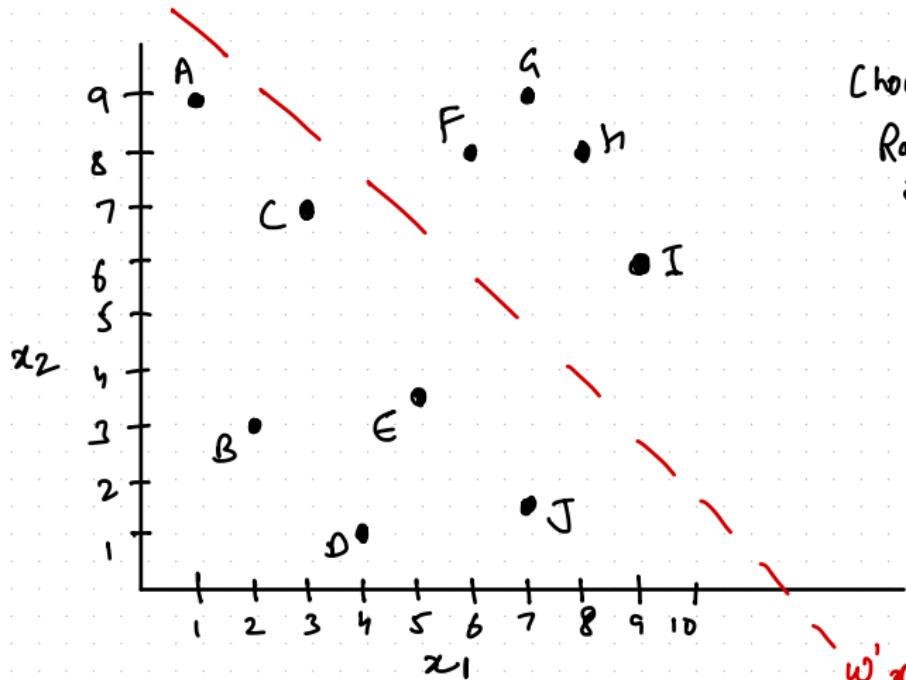
$$O(\log N)$$

Locality Sensitive Hashing (LSH) w/ Random Projection

(MachineLearningInterview.com)



Locality Sensitive Hashing (LSH) w/ Random Projection

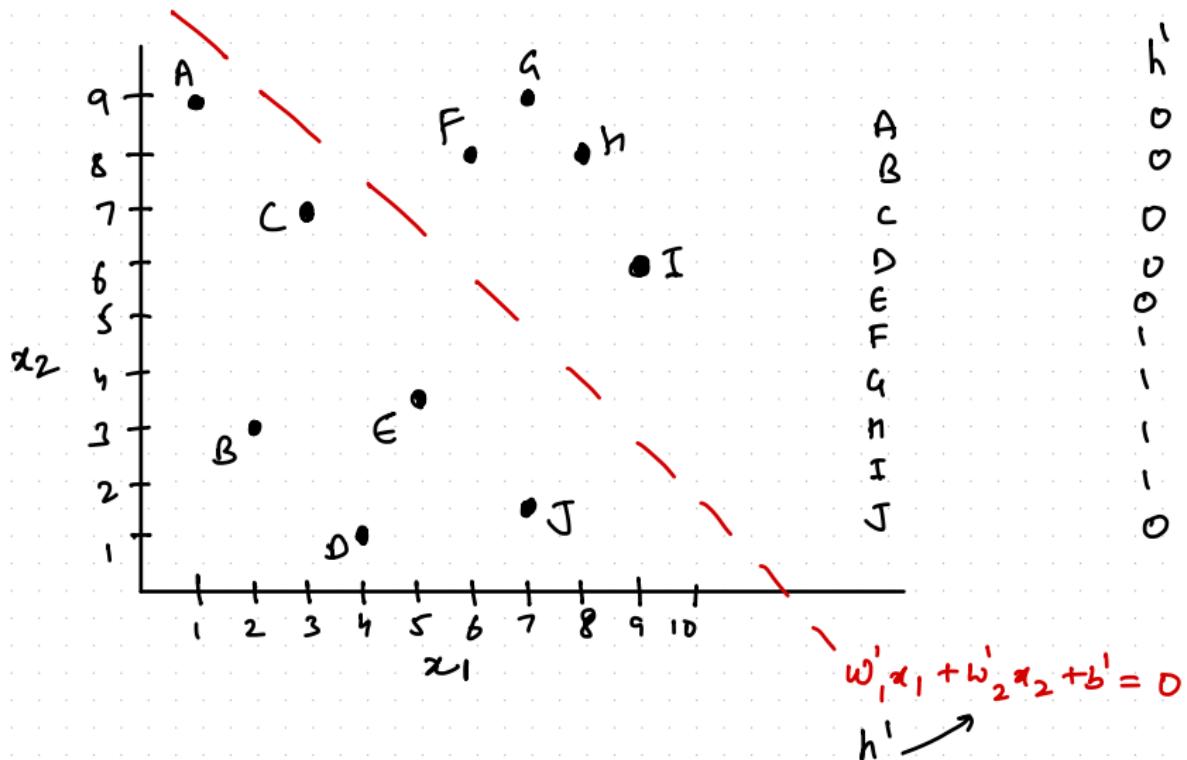


Choose h' :
Random hyperplane
in $1D$ dim

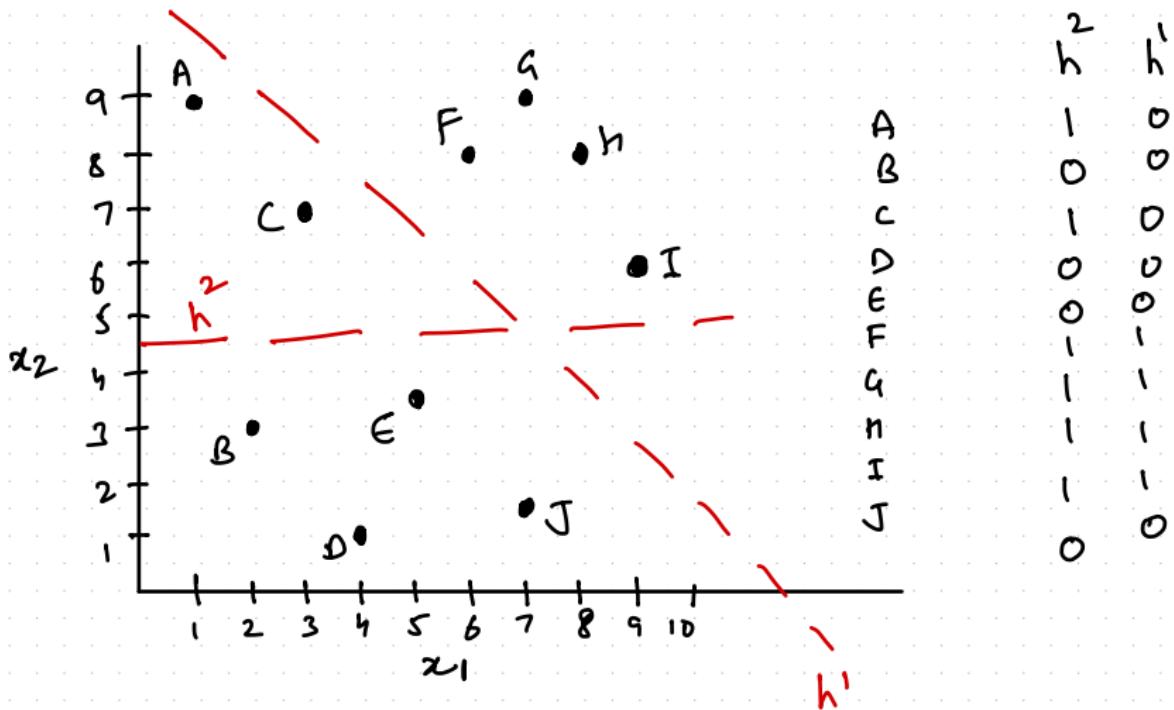
$$w'_1 x_1 + w'_2 x_2 + b' = 0$$

$h' \nearrow$

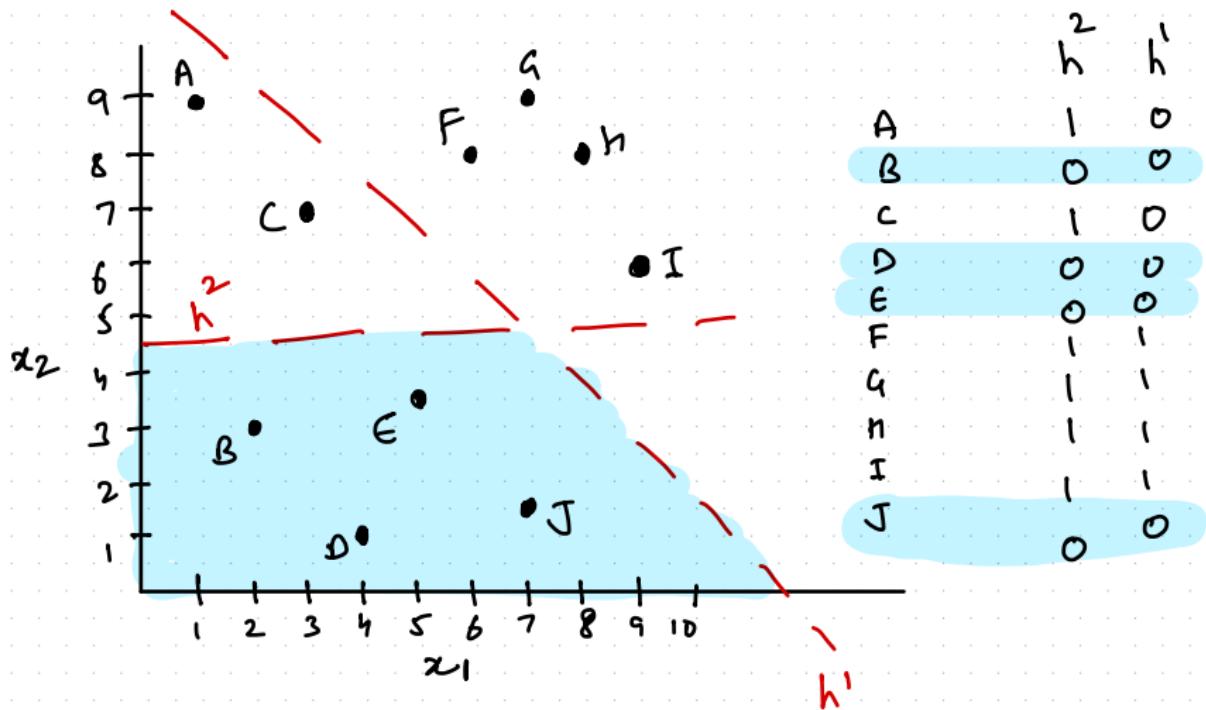
Locality Sensitive Hashing (LSH) w/ Random Projection



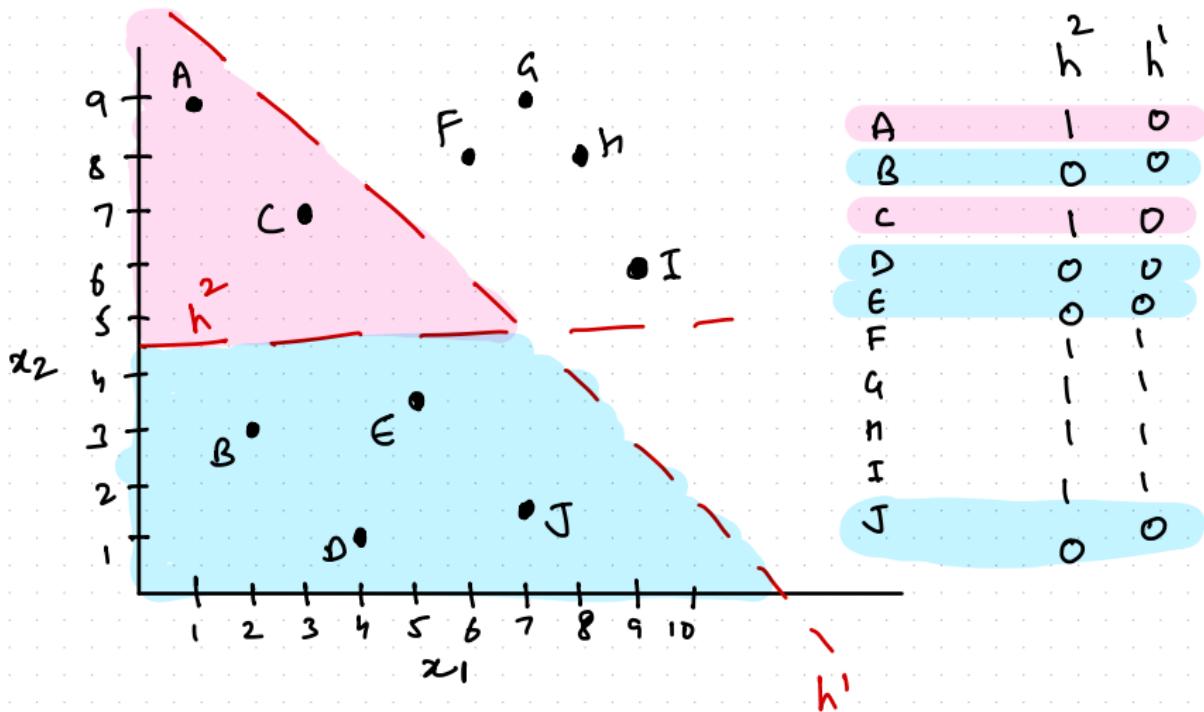
Locality Sensitive Hashing (LSH) w/ Random Projection



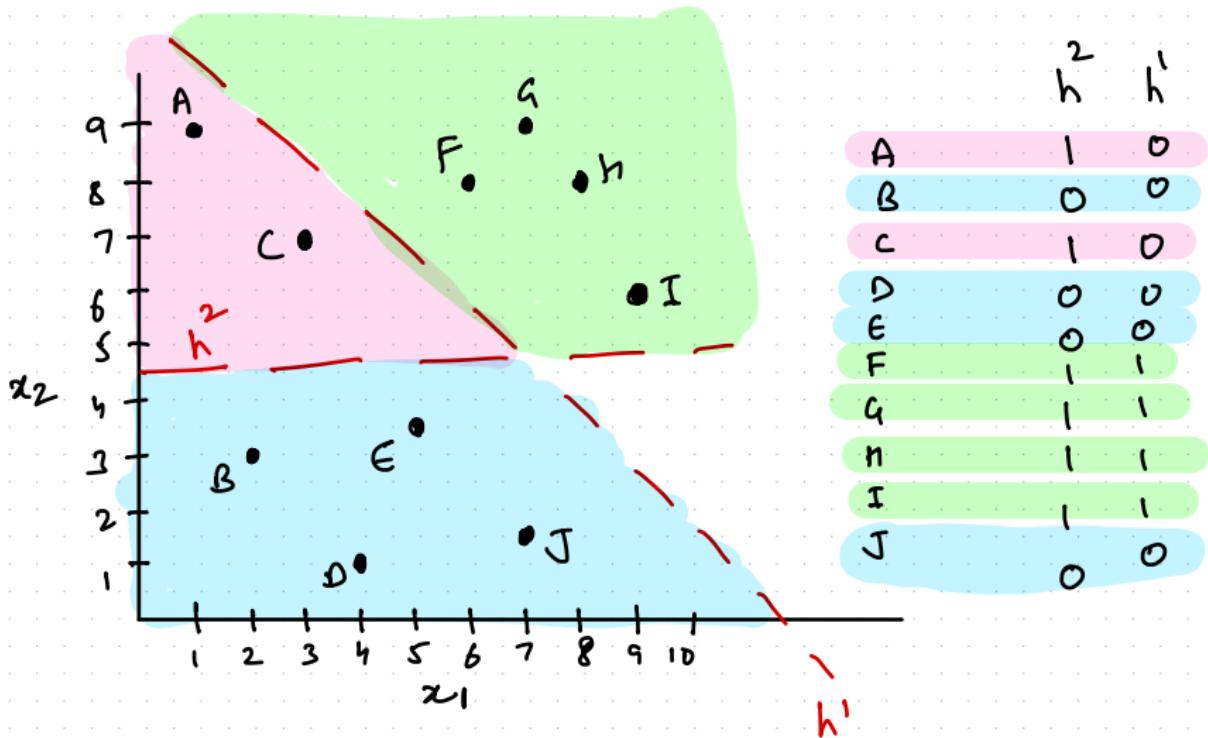
Locality Sensitive Hashing (LSH) w/ Random Projection



Locality Sensitive Hashing (LSH) w/ Random Projection



Locality Sensitive Hashing (LSH) w/ Random Projection



Practical implementation

* How to sample hyperplane?

$i = 1 \dots D :$ sampled from

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

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

Practical implementation

* How to get $h^i(x) = 0 \text{ or } 1$

$$h^i(x) = \text{SIGN}(b + \hat{w}_1^i x_1 + w_2^i x_2 + \dots)$$

Practical implementation

- » How to vectorize finding Hash Table

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

Practical implementation

- How to vectorize finding Hash Table

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

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

Practical implementation

- How to vectorize finding Hash Table

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

let # planes be 'p'

$$w = \begin{bmatrix} \dots & \dots & \dots \\ \vdots & \ddots & \vdots \\ - & - & - \\ - & - & - \end{bmatrix}_{D+1 \times p}$$

Practical implementation

- How to vectorize finding Hash Table

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

$$H = \text{sgn}(x^t w)_{N \times P}$$

let # planes be 'p'

$$w = \begin{bmatrix} \dots & \dots & \dots \\ N(\dots, \dots) & & \\ \dots & \dots & \dots \end{bmatrix}_{D+1 \times P}$$

Time complexity

Assuming one (1) set of hash functions

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

$$\text{Time} = O(N \times D \times P)$$

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

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)

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

$$\rightarrow \text{Test Time} = O(DP + \frac{DN}{2^P})$$