

Nearest Neighbor Methods

Introduction to Machine Learning (ICP363)

Paulo Mann · Instituto de Computação, UFRJ

Prepared based on the teaching material of **Sebastian Raschka**¹ (STAT 451: Intro to Machine Learning, Fall 2020)

¹ https://sebastianraschka.com/pdf/lecture-notes/stat451fs20/02-knn_slides.pdf

About these slides

These slides were prepared based on the lecture material of **Sebastian Raschka**

STAT 451: Introduction to Machine Learning
University of Wisconsin–Madison, Fall 2020
Nearest Neighbor Methods

Order, concepts and examples follow the original lecture notes; the visual style was redesigned and the notation follows the course conventions. All credit for the content goes to him.



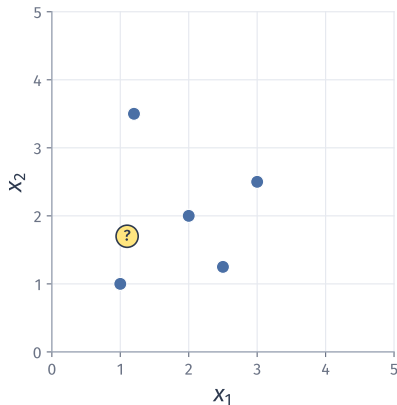
Topics

- 1. Intro to nearest neighbor models**
2. Nearest neighbor decision boundary
3. K-nearest neighbors
4. Naive Nearest Neighbor Search
5. Nearest neighbor search with a priority queue

1-Nearest Neighbor

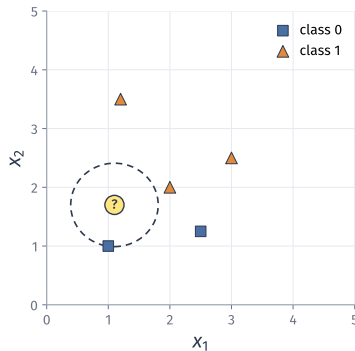
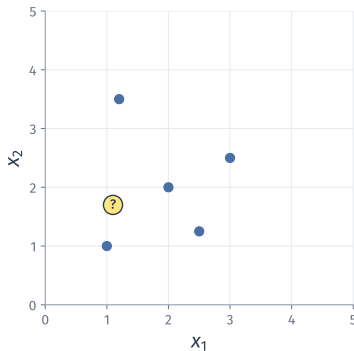
1-Nearest Neighbor

Task: predict the target / label of a new data point



1-Nearest Neighbor

Task: predict the target / label of a new data point



How? Look at the most “similar” data point in the training set

1-Nearest Neighbor Training Step

$$(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{D} \quad (|\mathcal{D}| = n)$$

How do we “train” the 1-NN model?

1-Nearest Neighbor Training Step

$$(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{D} \quad (|\mathcal{D}| = n)$$

To train the 1-NN model, we simply “remember” the training dataset

1-Nearest Neighbor Prediction Step

Given

$(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{D} \quad (|\mathcal{D}| = n)$

$(\mathbf{x}, ???)$

Predict

$h(\mathbf{x})$

Algorithm →

```
closest_point := None
closest_distance :=  $\infty$ 
for  $i = 1, \dots, n$ :
    current_distance :=  $d(\mathbf{x}^{(i)}, \mathbf{x})$ 
    if  $\text{current\_distance} < \text{closest\_distance}$ :
        closest_distance := current_distance
        closest_point :=  $\mathbf{x}^{(i)}$ 
return label  $y$  of closest_point
```

query point

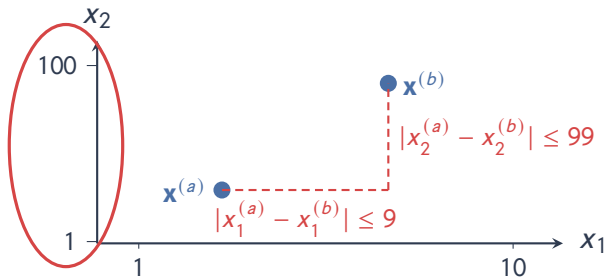


Commonly used: Euclidean Distance (L^2)

$$d(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = \sqrt{\sum_{j=1}^p (x_j^{(a)} - x_j^{(b)})^2}$$

Commonly used: Euclidean Distance (L^2)

$$\underline{d(\mathbf{x}^{(a)}, \mathbf{x}^{(b)})} = \sqrt{\sum_{j=1}^p \underline{(x_j^{(a)} - x_j^{(b)})^2}}$$



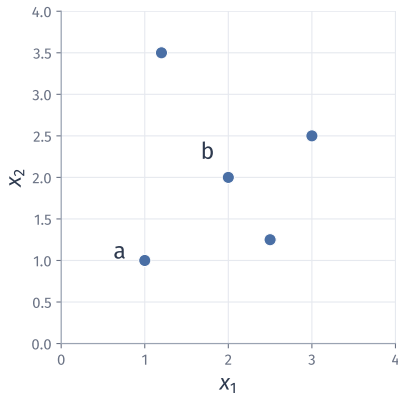
$x_2 \in [1, 100]$ vs. $x_1 \in [1, 10] \Rightarrow x_2$ dominates the distance

Topics

1. Intro to nearest neighbor models
- 2. Nearest neighbor decision boundary**
3. K-nearest neighbors
4. Naive Nearest Neighbor Search
5. Nearest neighbor search with a priority queue

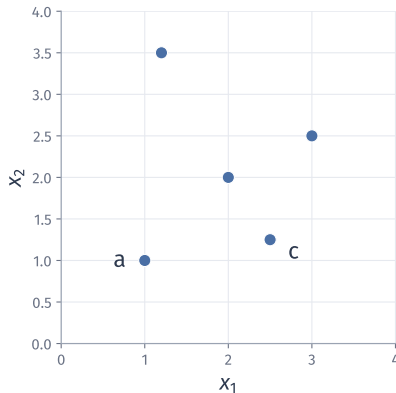
Nearest Neighbor Decision Boundary

Decision Boundary Between (a) and (b)



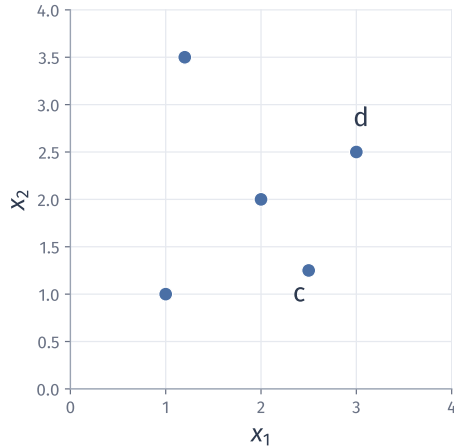
How does it look like?

Decision Boundary Between (a) and (c)

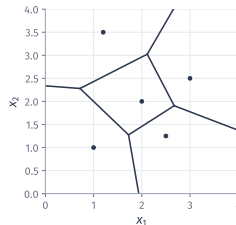
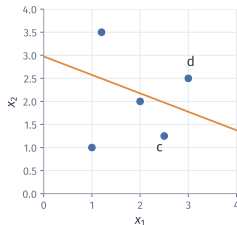
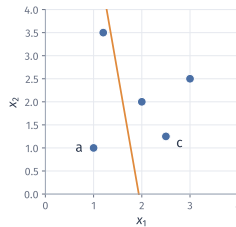
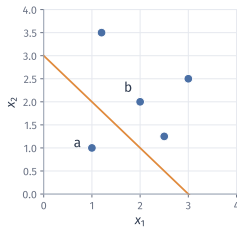


How does it look like?

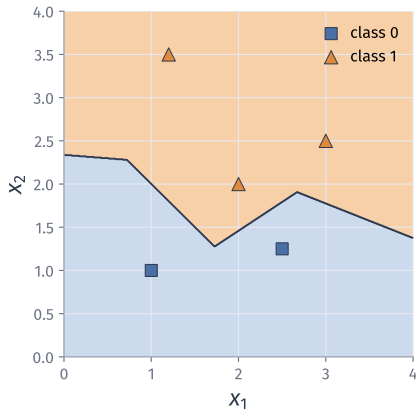
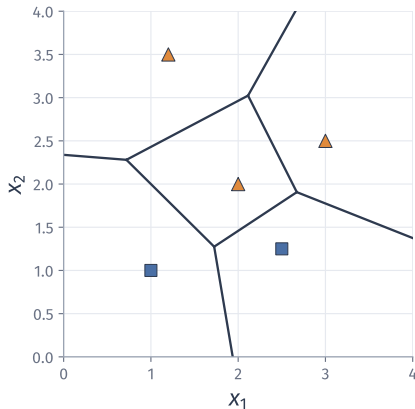
Decision Boundary Between (c) and (d)



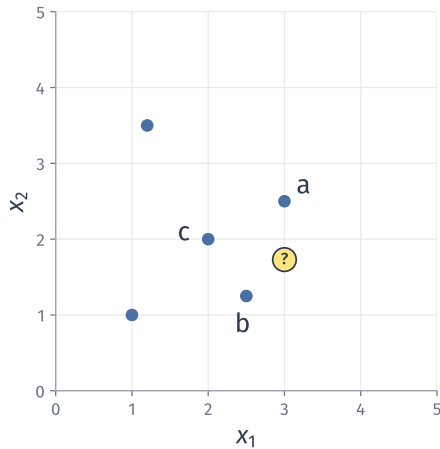
Decision Boundary of 1-NN



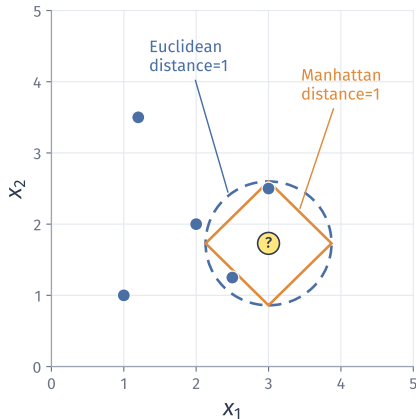
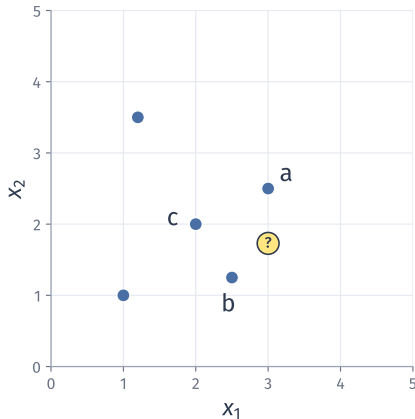
Decision Boundary of 1-NN



Which Point is Closest to ? ?



Depends on the Distance Measure!



Some Common Continuous Distance Measures

Euclidean

$$d(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = \sqrt{\sum_{j=1}^p (x_j^{(a)} - x_j^{(b)})^2} = \|\mathbf{x}^{(a)} - \mathbf{x}^{(b)}\|_2$$

Manhattan

$$d(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = \sum_{j=1}^p |x_j^{(a)} - x_j^{(b)}| = \|\mathbf{x}^{(a)} - \mathbf{x}^{(b)}\|_1$$

Minkowski · Mahalanobis · Cosine similarity · ...

Minkowski Distance

$$d(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = \left[\sum_{j=1}^p |x_j^{(a)} - x_j^{(b)}|^q \right]^{\frac{1}{q}} = \|\mathbf{x}^{(a)} - \mathbf{x}^{(b)}\|_q$$

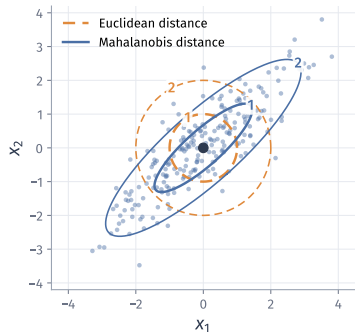
$q = 1 \Rightarrow$ Manhattan

$q = 2 \Rightarrow$ Euclidean

Mahalanobis Distance

$$d_M(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = \sqrt{(\mathbf{x}^{(a)} - \mathbf{x}^{(b)})^\top \Sigma^{-1} (\mathbf{x}^{(a)} - \mathbf{x}^{(b)})}$$

- Σ : feature covariance matrix, estimated from the training data (as in LDA)
- Differences measured **in standard deviations**: features with large spread count less
- Accounts for **correlations** between features (off-diagonal entries of Σ)
- $\Sigma = \mathbf{I}$ (variances 1, covariances 0) $\Rightarrow$ **Euclidean distance**
- $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_p^2) \Rightarrow$ Euclidean on standardized features

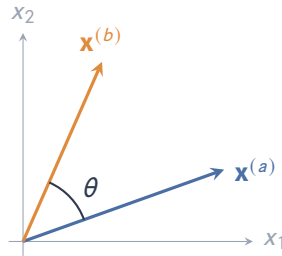


Cosine Similarity

$$\cos(\theta) = \frac{\mathbf{x}^{(a)\top} \mathbf{x}^{(b)}}{\|\mathbf{x}^{(a)}\|_2 \|\mathbf{x}^{(b)}\|_2} = \frac{\sum_{j=1}^p x_j^{(a)} x_j^{(b)}}{\sqrt{\sum_{j=1}^p (x_j^{(a)})^2} \sqrt{\sum_{j=1}^p (x_j^{(b)})^2}}$$

Cosine distance: $d(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = 1 - \cos(\theta)$

- Depends only on the **angle** between the vectors, not on their length
- $\cos(\theta) \in [-1, 1]$: 1 = same direction, 0 = orthogonal



Discrete Distance: Hamming

$$d(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = \sum_{j=1}^p |x_j^{(a)} - x_j^{(b)}| \quad \text{where } x_j \in \{0, 1\}$$

counts the positions where the two binary vectors differ

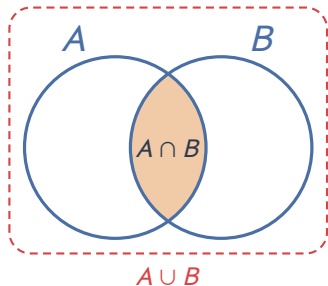
	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$
$\mathbf{x}^{(a)}$	0	1	0	1	0	0	1
$\mathbf{x}^{(b)}$	0	1	0	0	1	0	1
$ x_j^{(a)} - x_j^{(b)} $	0	0	0	1	1	0	0

$\Rightarrow d = 1 + 1 = 2$

Discrete Similarity: Jaccard / Tanimoto

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

- A, B : **sets** (e.g., the positions where a binary vector is 1)
- $J \in [0, 1]$: 0 = nothing in common, 1 = identical sets
- Jaccard *distance*: $d_J(A, B) = 1 - J(A, B)$



From Binary Feature Vectors to Sets

$$A = \{j : x_j^{(a)} = 1\} \quad B = \{j : x_j^{(b)} = 1\}$$

the *indices* of the features equal to 1 (e.g., $x_j = 1$: document contains word j)

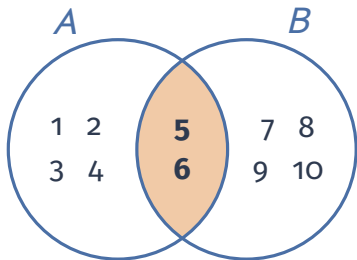
	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$	$j=10$	
$\mathbf{x}^{(a)}$	1	1	1	1	1	1	0	0	0	0	$\Rightarrow A = \{1, 2, 3, 4, 5, 6\}$
$\mathbf{x}^{(b)}$	0	0	0	0	1	1	1	1	1	1	$\Rightarrow B = \{5, 6, 7, 8, 9, 10\}$

both 1: $A \cap B = \{5, 6\}$

at least one 1: $A \cup B = \{1, \dots, 10\}$

Positions where both vectors are 0 do not enter $A \cup B$: Jaccard ignores shared absences (unlike Hamming).

Jaccard / Tanimoto: Example



$$A = \{1, 2, 3, 4, 5, 6\}, \quad |A| = 6$$

$$B = \{5, 6, 7, 8, 9, 10\}, \quad |B| = 6$$

$$A \cap B = \{5, 6\}, \quad |A \cap B| = 2$$

$$|A \cup B| = 6 + 6 - 2 = 10$$

$$J(A, B) = \frac{2}{10} = 0.2$$

$$d_J(A, B) = 1 - 0.2 = 0.8$$

Discrete Similarity: Dice

$$D(A, B) = \frac{2|A \cap B|}{|A| + |B|}$$

- Like Jaccard, but the intersection counts **twice**
- $D \in [0, 1]$, and $D \geq J$ for the same pair of sets
- Relation: $D = \frac{2J}{1 + J}$

Same example:

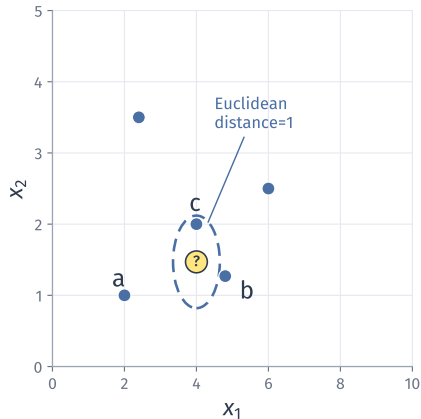
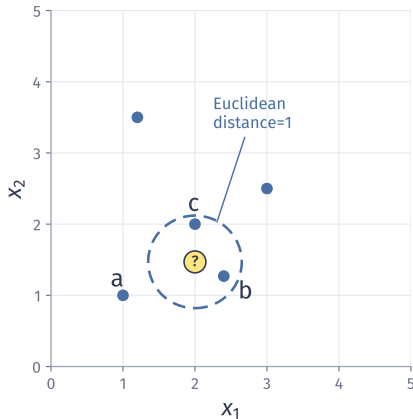
$$A = \{1, \dots, 6\}, \quad B = \{5, \dots, 10\}$$

$$|A \cap B| = 2, \quad |A| + |B| = 12$$

$$D(A, B) = \frac{2 \cdot 2}{12} = 0.33$$

$$\text{vs. } J(A, B) = 0.2; \quad \frac{2 \cdot 0.2}{1.2} = 0.33 \checkmark$$

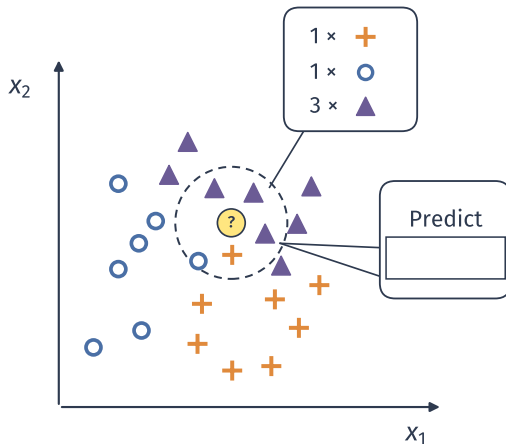
Feature Scaling



Topics

1. Intro to nearest neighbor models
2. Nearest neighbor decision boundary
- 3. K-nearest neighbors**
4. Naive Nearest Neighbor Search
5. Nearest neighbor search with a priority queue

k -Nearest Neighbors



Majority vs. Plurality Vote

A y : 

Majority vote: 

Plurality vote: 

B y : 

Majority vote: None

Plurality vote: 

k NN for Classification

$$\mathcal{D}_k = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(k)}, y^{(k)})\} \quad \mathcal{D}_k \subseteq \mathcal{D}$$

$$h(\mathbf{x}) = \arg \max_{\ell \in \{1, \dots, g\}} \sum_{i=1}^k \mathbb{I}(y^{(i)} = \ell) \quad \mathbb{I}(A) = \begin{cases} 1, & \text{if } A \text{ is true,} \\ 0, & \text{otherwise.} \end{cases}$$

$$h(\mathbf{x}) = \text{mode}(\{y^{(1)}, \dots, y^{(k)}\})$$

k NN for Regression

$$\mathcal{D}_k = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(k)}, y^{(k)})\} \quad \mathcal{D}_k \subseteq \mathcal{D}$$

$$\hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^k y^{(i)}$$

Topics

1. Intro to nearest neighbor models
2. Nearest neighbor decision boundary
3. K-nearest neighbors
- 4. Naive Nearest Neighbor Search**
5. Nearest neighbor search with a priority queue

Naive Nearest Neighbor Search

Variant A

```
 $\mathcal{D}_k := \{\}$   
while  $|\mathcal{D}_k| < k$ :  
    closest_distance :=  $\infty$   
    for  $i = 1, \dots, n, \quad \forall i \notin \mathcal{D}_k$ :  
        current_distance :=  $d(\mathbf{x}^{(i)}, \mathbf{x})$   
        if current_distance < closest_distance:  
            closest_distance := current_distance  
            closest_point :=  $\mathbf{x}^{(i)}$   
    add closest_point to  $\mathcal{D}_k$ 
```

Naive Nearest Neighbor Search

Variant A

```
 $\mathcal{D}_k := \{\}$   
while  $|\mathcal{D}_k| < k$ :  
    closest_distance :=  $\infty$   
    for  $i = 1, \dots, n, \quad \forall i \notin \mathcal{D}_k$ :  
        current_distance :=  $d(\mathbf{x}^{(i)}, \mathbf{x})$   
        if current_distance < closest_distance:  
            closest_distance := current_distance  
            closest_point :=  $\mathbf{x}^{(i)}$   
    add closest_point to  $\mathcal{D}_k$ 
```

$$O(k \cdot n)$$

while: k iterations

for: n points each

each distance costs $O(p)$

Naive Nearest Neighbor Search

Variant B

$\mathcal{D}_k := \mathcal{D}$

while $|\mathcal{D}_k| > k$:

 largest_distance := 0

for $i = 1, \dots, n, \quad \forall i \in \mathcal{D}_k$:

 current_distance := $d(\mathbf{x}^{(i)}, \mathbf{x})$

if current_distance > largest_distance:

 largest_distance := current_distance

 farthest_point := $\mathbf{x}^{(i)}$

 remove farthest_point from $\mathcal{D}_k$

Naive Nearest Neighbor Search

Variant B

$\mathcal{D}_k := \mathcal{D}$

while $|\mathcal{D}_k| > k$:

 largest_distance := 0

for $i = 1, \dots, n, \quad \forall i \in \mathcal{D}_k$:

 current_distance := $d(\mathbf{x}^{(i)}, \mathbf{x})$

if current_distance > largest_distance:

 largest_distance := current_distance

 farthest_point := $\mathbf{x}^{(i)}$

 remove farthest_point from $\mathcal{D}_k$

$$O((n - k) \cdot n)$$

$$= O(n^2) \text{ for } k \ll n$$

while: $n - k$ iterations

for: up to n points each

each distance costs $O(p)$

Topics

1. Intro to nearest neighbor models
2. Nearest neighbor decision boundary
3. K-nearest neighbors
4. Naive Nearest Neighbor Search
- 5. Nearest neighbor search with a priority queue**

Nearest Neighbor Search with a Priority Queue

Idea: keep the k closest points seen so far in a **max-heap**, ordered by distance to $\mathbf{x}$ (farthest on top)

$H :=$ heap with $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)}$ and their distances $d(\mathbf{x}^{(i)}, \mathbf{x})$

for $i = k + 1, \dots, n$:

$\text{current_distance} := d(\mathbf{x}^{(i)}, \mathbf{x})$

if $\text{current_distance} < \text{largest distance in } H$:

 remove the farthest point from H

 insert $\mathbf{x}^{(i)}$ into H

$\mathcal{D}_k :=$ points in H

Nearest Neighbor Search with a Priority Queue

Idea: keep the k closest points seen so far in a **max-heap**, ordered by distance to $\mathbf{x}$ (farthest on top)

$H :=$ heap with $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)}$ and their distances $d(\mathbf{x}^{(i)}, \mathbf{x})$

for $i = k + 1, \dots, n$:

$\text{current_distance} := d(\mathbf{x}^{(i)}, \mathbf{x})$

if $\text{current_distance} < \text{largest distance in } H$:

 remove the farthest point from H

 insert $\mathbf{x}^{(i)}$ into H

$\mathcal{D}_k :=$ points in H

$$O(n \log k)$$

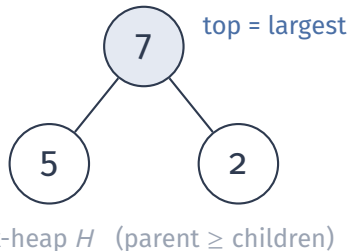
for: one pass over n points

heap: remove / insert in $O(\log k)$

each distance costs $O(p)$

Priority Queue: Example with $k = 3$

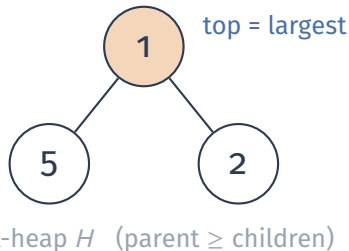
i	1	2	3	4	5	6	7
$d(\mathbf{x}^{(i)}, \mathbf{x})$	5	2	7	1	4	6	3



Initialize H with the first $k = 3$ points: the largest distance, 7, goes on top

Priority Queue: Example with $k = 3$

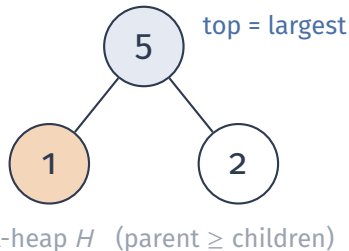
i	1	2	3	4	5	6	7
$d(\mathbf{x}^{(i)}, \mathbf{x})$	5	2	7	1	4	6	3



$i = 4$: $1 < 7$ (top) $\Rightarrow$ remove 7 and put 1 on top

Priority Queue: Example with $k = 3$

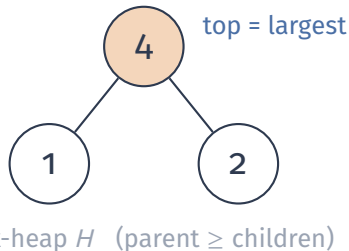
i	1	2	3	4	5	6	7
$d(\mathbf{x}^{(i)}, \mathbf{x})$	5	2	7	1	4	6	3



Restore the heap: swap 1 with its larger child 5 (cost $O(\log k)$)

Priority Queue: Example with $k = 3$

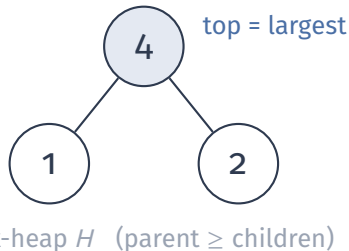
i	1	2	3	4	5	6	7
$d(\mathbf{x}^{(i)}, \mathbf{x})$	5	2	7	1	4	6	3



$i = 5$: $4 < 5$ (top) $\Rightarrow$ remove 5 and put 4 on top (no swap needed)

Priority Queue: Example with $k = 3$

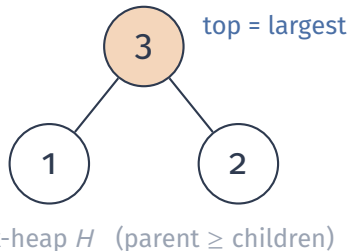
i	1	2	3	4	5	6	7
$d(\mathbf{x}^{(i)}, \mathbf{x})$	5	2	7	1	4	6	3



$i = 6$: $6 > 4$ (top) $\Rightarrow$ not among the 3 nearest: **skip**

Priority Queue: Example with $k = 3$

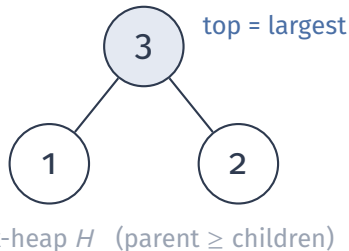
i	1	2	3	4	5	6	7
$d(\mathbf{x}^{(i)}, \mathbf{x})$	5	2	7	1	4	6	3



$i = 7$: $3 < 4$ (top) $\Rightarrow$ remove 4 and put 3 on top

Priority Queue: Example with $k = 3$

i	1	2	3	4	5	6	7
$d(\mathbf{x}^{(i)}, \mathbf{x})$	5	2	7	1	4	6	3



Done: H holds the distances $\{3, 2, 1\} \Rightarrow$ the 3 nearest neighbors

Nearest Neighbor Search: Runtime

Method	Time
Variant A	$O(k \cdot n)$
Variant B	$O((n - k) \cdot n)$
Priority queue	$O(n \log k)$