Large-Scale Machine Learning: Nearest Neighbor Methods

Thomas Bonald

2019 – 2020

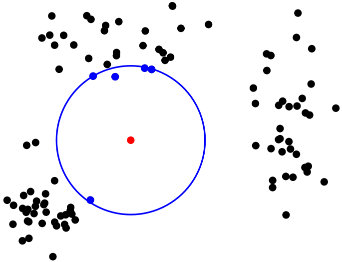
Nearest neighbor methods

Key approach to **large-scale** machine learning, useful for:

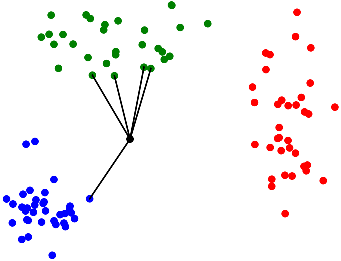
- Information retrieval   e.g., Shazam, Smartify
- Density estimation
- Classification
- Clustering
- Anomaly detection

Example

Density estimation

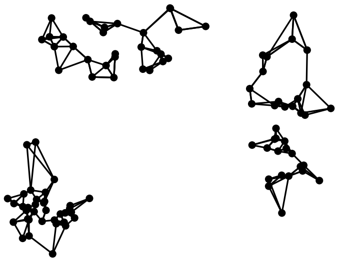


Classification

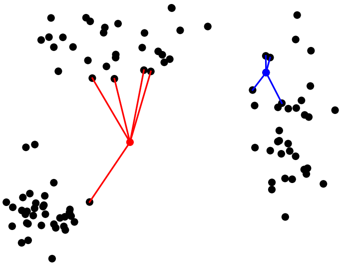


Example

Clustering



Anomaly detection



Nearest neighbors

A key appraoch to **large-scale** machine learning, useful for:

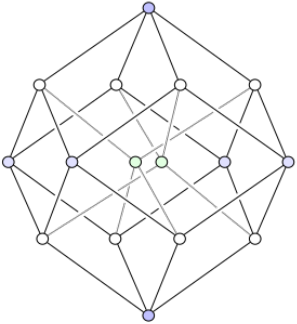
- Information retrieval e.g., Shazam, Smartify
- Density estimation
- Classification
- Clustering
- Anomaly detection

Challenges:

- Large number of samples nn search in $O(n)$
- High dimension

Curse of dimensionality

In high dimension, distances tend to be **similar** for all sample pairs (assuming independent features)



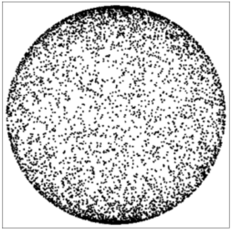
Example (real features)

- For $X, Y \sim \mathcal{N}(0, I_d)$,

$$||X - Y||^2 \sim 2\chi^2(d) \approx 2\mathcal{N}(d, 2d) \quad \text{when } d \to +\infty$$

- Coefficient of variation:

$$\text{cv} = O\left(\frac{1}{\sqrt{d}}\right)$$



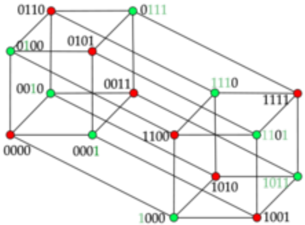
Example (binary features)

- For $X, Y \sim \mathcal{U}(\{0,1\}^d)$ and the Hamming distance,

$$d(X,Y) \sim \mathcal{B}(d, \frac{1}{2}) \approx \mathcal{N}(\frac{d}{2}, \frac{d}{2}) \quad \text{when } d \to +\infty$$

- Coefficient of variation:

$$\text{cv} = O\left(\frac{1}{\sqrt{d}}\right)$$



Outline

Part I

1. Metrics
2. Thinning
3. Searching
4. Clustering

Part II

5. Sketching
6. Embedding

Outline

1. **Metrics** $\rightarrow$ Minkowski, Jaccard, Hellinger
2. Thinning
3. Searching
4. Clustering
5. Sketching
6. Embedding

Euclidean distance

Let $x, y \in \mathbb{R}^d$

$$d(x, y) = ||x - y|| \; = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

Manhattan distance

Let $x, y \in \mathbb{R}^d$

$$d(x, y) = ||x - y||_1 \ = \sum_{i=1}^{d} |x_i - y_i|$$

Maximum distance

Let $x, y \in \mathbb{R}^d$

$$d(x, y) = ||x - y||_\infty \; = \max_{i=1,\ldots,d} |x_i - y_i|$$

Minkowski distance

Let $x, y \in \mathbb{R}^d$

$$d(x,y) = ||x-y||_p \ = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}} \quad p \geq 1$$

Particular cases:

- $p = 1$ $\rightarrow$ Manhattan distance
- $p = 2$ $\rightarrow$ Euclidean distance
- $p = +\infty$ $\rightarrow$ Maximum distance

Minkowski 1953

Cosine similarity

Let $x, y \in \mathbb{R}^d \setminus \{0\}$

$$s(x, y) = \cos(x, y) = \frac{x.y}{||x||||y||} \in [-1, 1]$$

Remark

$$d(\bar{x}, \bar{y}) = \sqrt{2(1 - s(x, y))}$$

with d the Euclidean distance,

$$\bar{x} = \frac{x}{||x||} \quad \bar{y} = \frac{y}{||y||}$$

Linear mapping

Let $x, y \in \mathbb{R}^d$

$$d(x,y) = ||Ax - Ay|| \ = \sqrt{(x-y)^T A^T A (x-y)}$$

Particular cases:

- A diagonal $\rightarrow$ scaling
- $A^T A = \text{Cov}^{-1}$ $\rightarrow$ Mahalanobis distance
- A rectangular $\rightarrow$ dimension reduction

Mahalanobis 1936

Hamming distance (binary features)

Let $x, y \in \{0,1\}^d$

$$d(x,y) = \sum_{i=1}^{d} |x_i - y_i|$$

$\rightarrow$ the Manhattan distance

Hamming 1950

Jaccard distance (binary features)

Let $x, y \in \{0, 1\}^d$

- Similarity:

$$s(x, y) = \frac{|A \cap B|}{|A \cup B|} \in [0, 1]$$

  with

$$A = \{i : x_i = 1\} \quad B = \{i : y_i = 1\}$$

- Distance:

$$d(x, y) = 1 - s(x, y) = \frac{|A \triangle B|}{|A \cup B|} = \frac{\sum_{i=1}^{d} |x_i - y_i|}{\sum_{i=1}^{d} \max(x_i, y_i)}$$

Jaccard 1901

Hellinger distance (positive features)

Let $x, y \in \mathbb{R}_+^d$

- Probability measures after normalization:

$$p = \frac{x}{\sum_{i=1}^d x_i} \quad q = \frac{y}{\sum_{i=1}^d y_i}$$

- Distance:

$$d(x,y) = \frac{1}{\sqrt{2}} ||\sqrt{p} - \sqrt{q}|| \in [0,1]$$

- Similarity:

$$d(x,y) = \sqrt{1 - s(x,y)} \quad \text{with} \quad s(x,y) = \sum_{i=1}^d \sqrt{p_i q_i}$$

Hellinger 1909

Bhattacharyya similarity

Let $x, y \in \mathbb{R}_+^d$

- Probability measures after normalization:

$$p = \frac{x}{\sum_{i=1}^{d} x_i} \quad q = \frac{y}{\sum_{i=1}^{d} y_i}$$

- Similarity:

$$s(x, y) = \sum_{i=1}^{d} \sqrt{p_i q_i}$$

- For binary vectors,

$$s(x, y) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2 \sum_{i=1}^{d} y_i^2}} = \cos(x, y)$$

Bhattacharyya 1943

Jensen-Shannon distance (positive features)

Let $x, y \in \mathbb{R}_+^d$

- Probability measures after normalization:

$$p = \frac{x}{\sum_{i=1}^d x_i} \quad q = \frac{y}{\sum_{i=1}^d y_i}$$

- Distance:

$$d(x, y) = \sqrt{H(\frac{1}{2}(p+q)) - \frac{1}{2}(H(p) + H(q))} \in [0, 1]$$

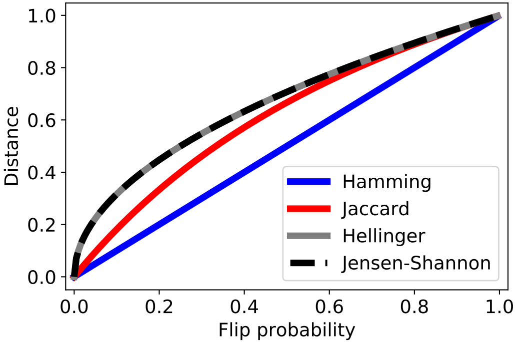
with

$$H(p) = -\sum_{i=1}^{d} p_i \log_2 p_i$$

Example

Average distance between binary vectors $x, y \in \{0,1\}^{100}$:

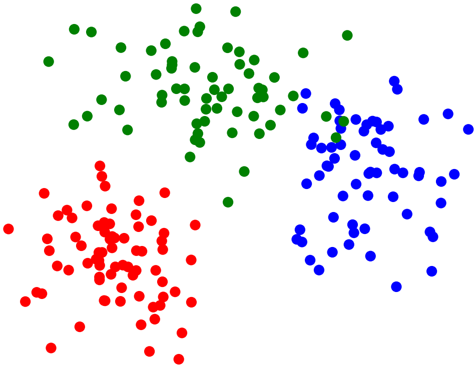
- $x = (1, \ldots, 1, 0, \ldots, 0)$
- $y = x$ with i.i.d. bit flips



Outline

1. Metrics
2. **Thinning** $\rightarrow$ Condensed nearest neighbors
3. Searching
4. Clustering
5. Sketching
6. Embedding

Idea: selection of prototypes



Condensed nearest neighbors

Data $X \in \mathbb{R}^{n \times d}$ with labels $\ell_1, \dots, \ell_n$ (training set)

Hart algorithm

$S \leftarrow \emptyset$
$A \leftarrow$ random set with one sample per label
While $A \neq \emptyset$:

- $S \leftarrow S \cup A$
- $A \leftarrow \{i \notin S : \min_{j \in S} d(X_i, X_j) < \min_{j \in S, \ell_j = \ell_i} d(X_i, X_j)\}$

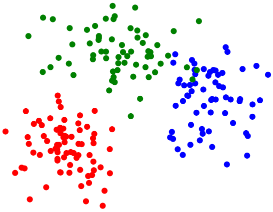
Return S

Note: Consistent algorithm (no error on the training set)
Can be adapted to k-nearest neighbors

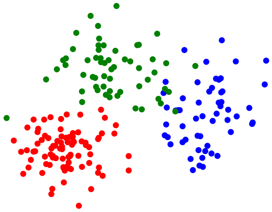
Hart 1968

Example: Gaussian mixture model

Train set (200 samples)



Test set (200 samples)

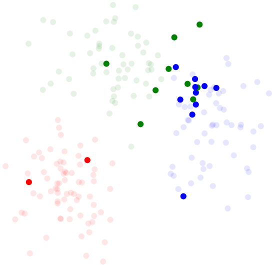


Precision of k-nn classification (test set):

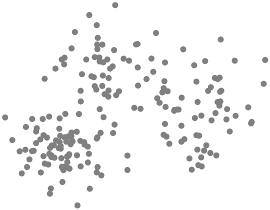
- 95% for $k = 1$
- 96% for $k = 3$

Example: Gaussian mixture model

Train set (200 samples)



Test set (200 samples)

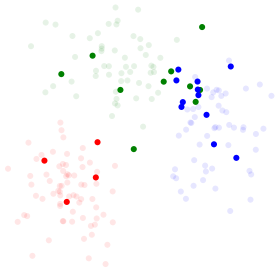


Condensed 1-nn $\sim$ 20 prototypes

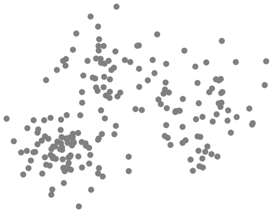
Precision of 1-nn classification $\rightarrow$ 94%

Example: Gaussian mixture model

Train set (200 samples)



Test set (200 samples)

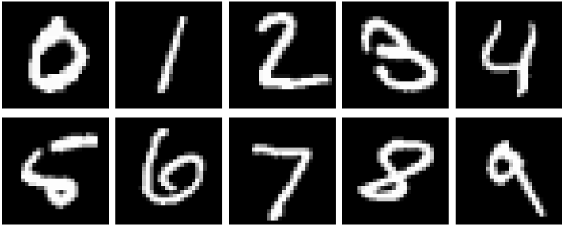


Condensed 3-nn $\sim$ 20 prototypes

Precision of 3-nn classification $\to$ 95%

Example: MNIST dataset

60,000 images (train set) of size 28×28
10,000 images (test set)



Precision of 3-nn classification (test set):

- 97% using the 60,000 samples of the train set
- 87% using 600 prototypes of the train set $\times$100 speed-up
- 81% using 600 random prototypes

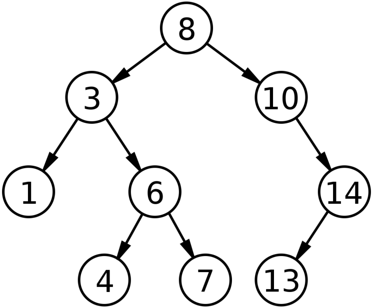
Outline

1. Metrics
2. Thinning
3. **Searching** $\rightarrow$ kd-trees, ball trees
4. Clustering
5. Sketching
6. Embedding

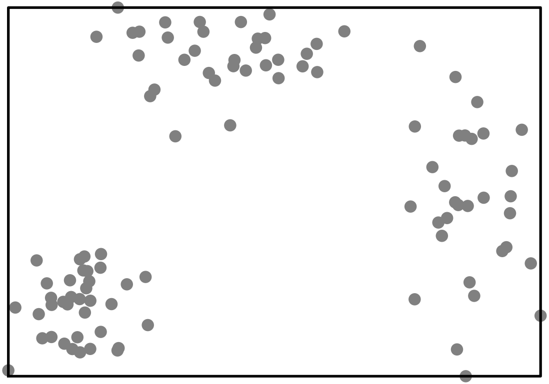
Binary tree search

For 1-d data, e.g.,

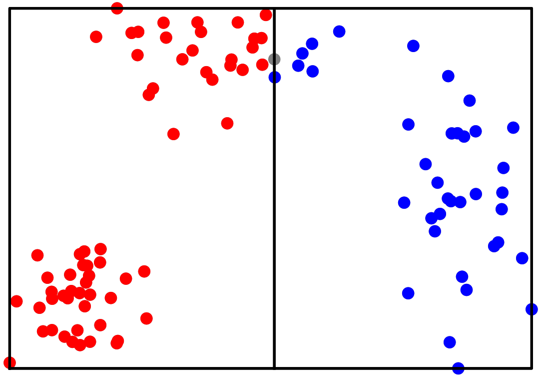
$$\{6, 10, 8, 1, 3, 13, 4, 14, 7\}$$



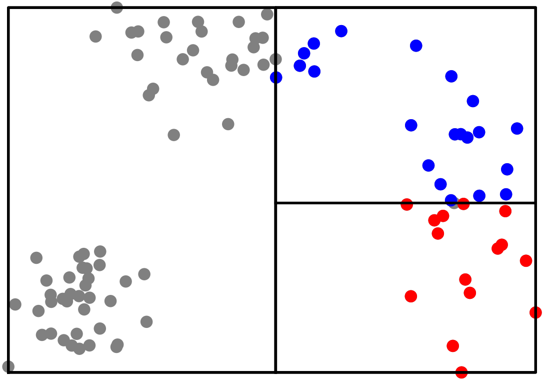
k-d tree



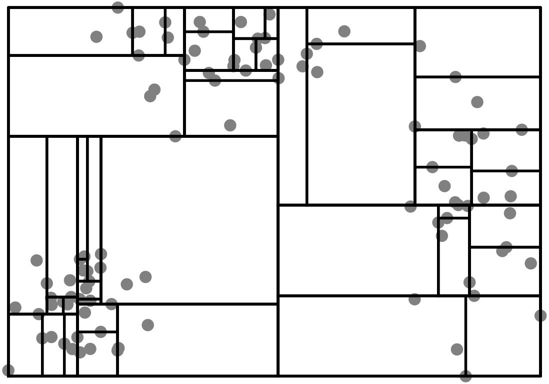
k-d tree



k-d tree



k-d tree



Construction

Data $X \in \mathbb{R}^{n \times d}$

kd_tree(S)

tree $\leftarrow$ new_tree

If $S \neq \emptyset$:

- $i, k \leftarrow$ split(S)
- tree.pivot, tree.direction $\leftarrow i, k$
- S_left $\leftarrow \{j \in S \setminus \{i\} : X_{jk} \leq X_{ik}\}$
- S_right $\leftarrow \{j \in S \setminus \{i\} : X_{jk} > X_{ik}\}$
- tree.left $\leftarrow$ kd_tree(S_left)
- tree.right $\leftarrow$ kd_tree(S_right)

Return tree

Splitting strategies

Highest variance

$k \leftarrow$ direction of **highest variance**
$i \leftarrow$ **median** sample in direction k

Note: Balanced trees but heterogeneous bins

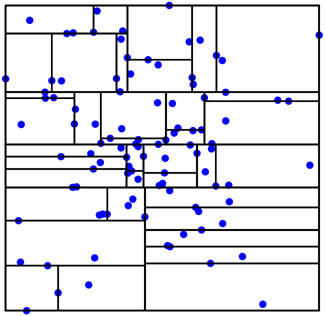
Max spread

$k \leftarrow$ direction of **maximum spread**
$i \leftarrow$ sample closest to the **middle** of the spread range in direction k

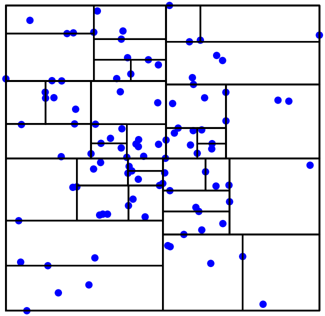
Note: Homogeneous bins but unbalanced trees

Example

Highest variance



Max spread



Pruning

No need for a tree structure for few samples (e.g., 10)

kd_tree(S)

tree $\leftarrow$ new_tree

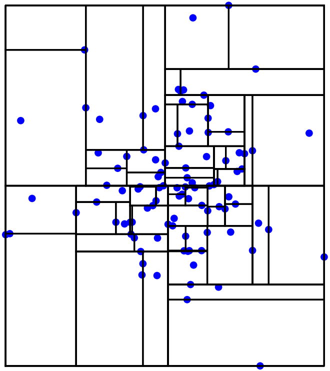
If $|S| >$ leaf_size:

- $i, k \leftarrow$ split(S)
- tree.pivot, tree.direction $\leftarrow i, k$
- S_left $\leftarrow \{j \in S \setminus \{i\} : X_{jk} \leq X_{ik}\}$
- S_right $\leftarrow \{j \in S \setminus \{i\} : X_{jk} > X_{ik}\}$
- tree.left $\leftarrow$ kd_tree(S_left)
- tree.right $\leftarrow$ kd_tree(S_right)

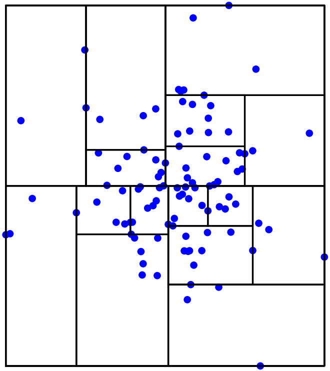
Return tree

Example

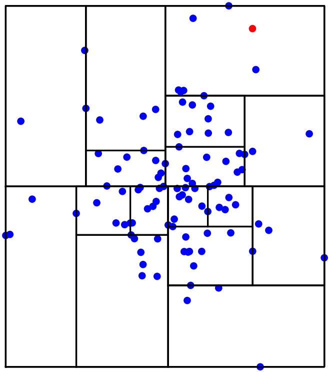
Leave size = 1 (full tree)



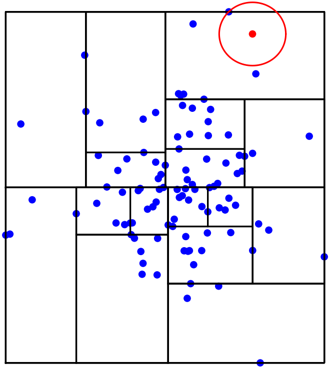
Leaf size = 10



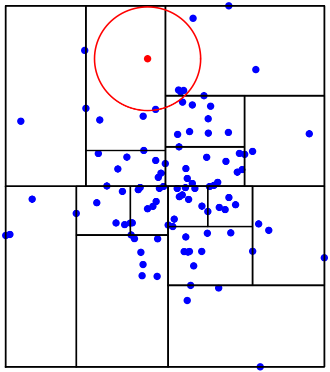
Nearest neighbor



Nearest neighbor



Nearest neighbor



Data structure

kd_tree(S, rectangle, ancestor)

tree $\leftarrow$ new_tree
tree.index $\leftarrow$ S
tree.rectangle $\leftarrow$ rectangle
tree.ancestor $\leftarrow$ ancestor

If $|S| >$ leaf_size:

- ...
- rectangle_left $\leftarrow$ rectangle.cut_left(k, X_{ik})
- rectangle_right $\leftarrow$ rectangle.cut_right(k, X_{ik})
- tree.left $\leftarrow$ kd_tree(S_left, rectangle_left, tree)
- tree.right $\leftarrow$ kd_tree(S_right, rectangle_right, tree)

Return tree

Nearest neighbor search

Let $x \in \mathbb{R}^d$ be the target

nn_search(x, kd_tree)

node $\leftarrow$ search(x, kd_tree) (leaf)
$i \leftarrow$ closest(x, node.index)
dist $\leftarrow d(x, X_i)$

While node.ancestor:

- previous, node $\leftarrow$ node, node.ancestor
- If previous $=$ node.left: tree $\leftarrow$ subtree(node.right) else ...
- If $d(x, X$_node.pivot$) <$ dist:
  $i \leftarrow$ node.pivot, dist $\leftarrow d(x, X_i)$
- If $d(x,$ tree.rectangle$) <$ dist:
  i, dist $\leftarrow$update(i, nn_search(x, tree))

Return i

k-nearest neighbor search

Let $x \in \mathbb{R}^d$ be the target

knn_search(x, kd_tree, k)

node $\leftarrow$ search(x, kd_tree)
$i_1, \ldots, i_k \leftarrow$ closest(x,node.index, k)
dist $\leftarrow \max(d(x, X_{i_1}), \ldots, d(x, X_{i_k}))$

While node.ancestor:

- ...
- If $d(x, X\_\text{node.pivot}) <$ dist:
  $i_1, \ldots i_k$, dist $\leftarrow$update($i_1, \ldots i_k$, node.pivot)
- If $d(x$, tree.rectangle$) <$ dist:
  $i_1, \ldots i_k$, dist $\leftarrow$update($i_1, \ldots i_k$, knn_search(x, tree, k))

Return $i_1, \ldots, i_k$

Near neighbor search

Let $x \in \mathbb{R}^d$ be the target, and $r > 0$ the target distance

nn_search(x, kd_tree, r)

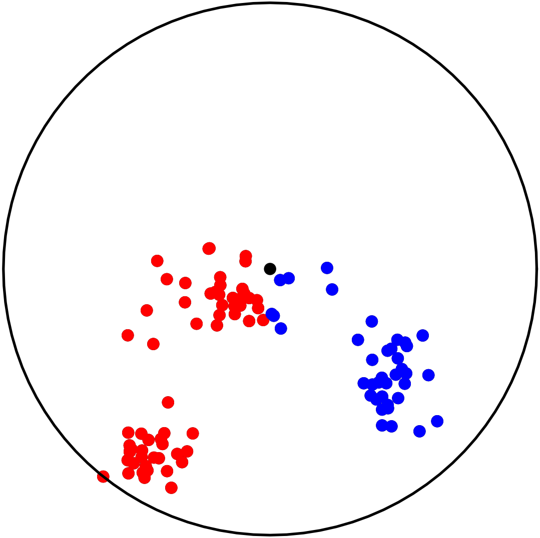
node $\leftarrow$ search(x, kd_tree)
nn_list $\leftarrow \{i \in \text{node.index} : d(x, X_i) < r\}$

While node.ancestor:

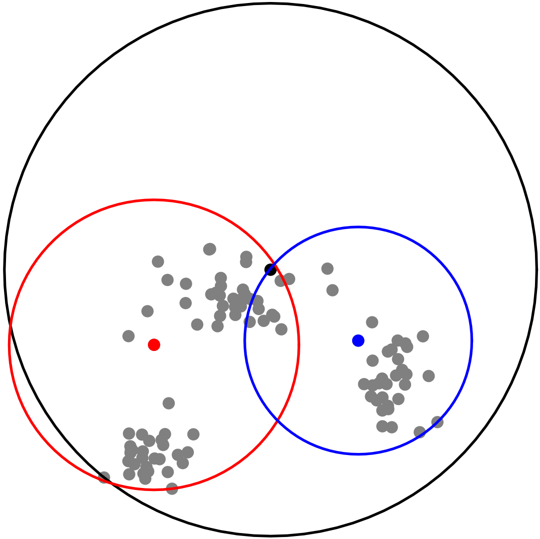
- ...
- If $d(x, X\_\text{node.pivot}) < \text{r}$:
  nn_list.add(node.pivot)
- If $d(x, \text{tree.rectangle}) < \text{r}$:
  nn_list.add(nn_search(x, tree, r))

Return nn_list

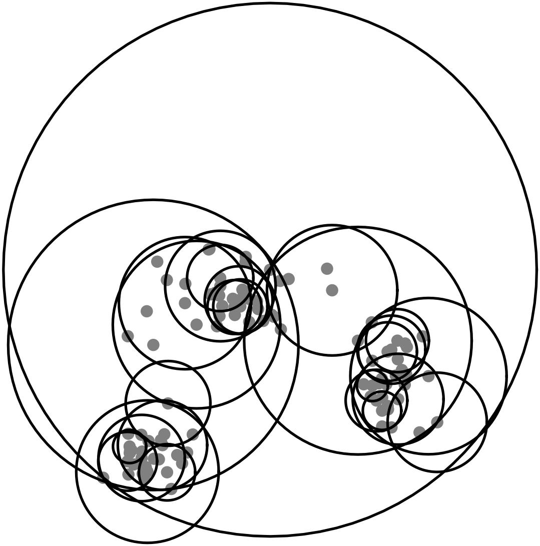
Ball tree



Ball tree



Ball tree



Construction

ball_tree(S)

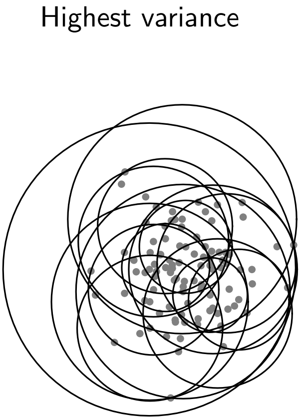
tree $\leftarrow$ new_tree

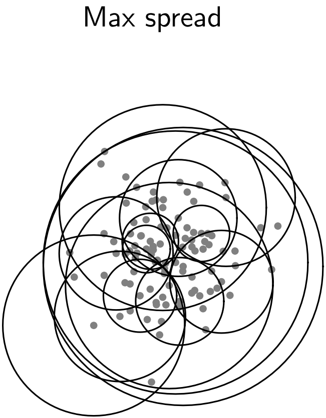
If $|S| >$ leaf_size:

- $i, k \leftarrow$ split(S)
- tree.pivot, tree.direction $\leftarrow i, k$
- S_left $\leftarrow \{j \in S \setminus \{i\} : X_{jk} \leq X_{ik}\}$
- S_right $\leftarrow \{j \in S \setminus \{i\} : X_{jk} > X_{ik}\}$
- tree.left $\leftarrow$ ball_tree(S_left)
- tree.right $\leftarrow$ ball_tree(S_right)

Return tree

Example





Data structure

ball_tree(S, ancestor)

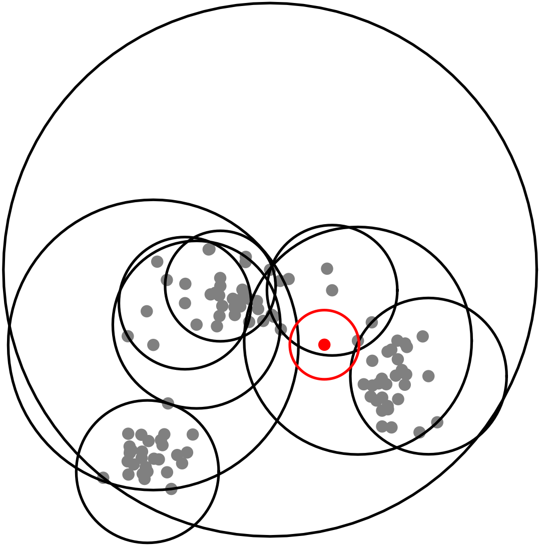
tree $\leftarrow$ new_tree
tree.index $\leftarrow$ S
tree.ancestor $\leftarrow$ ancestor

If $|S| >$ leaf_size:

- $i, k \leftarrow$ split(S)
- tree.pivot, tree.direction $\leftarrow$ i, k
- tree.radius $\leftarrow$ max_distance(tree.pivot, tree.index)
- ...
- tree.left $\leftarrow$ ball_tree(S_left, tree)
- tree.right $\leftarrow$ ball_tree(S_right, tree)

Return tree

Nearest neighbor search



Nearest neighbor search

Let $x \in \mathbb{R}^d$ be the target

nn_search(x, ball_tree)

node $\leftarrow$ search(x, ball_tree) (leaf)
$i \leftarrow$ closest(x, node.index)
dist $\leftarrow d(x, X_i)$

While node.ancestor:

- ...
- If $d(x, X\_\text{node.pivot}) <$ dist:
  $i \leftarrow$ node.pivot, dist $\leftarrow d(x, X_i)$
- If $d(x, X\_\text{tree.pivot}) <$ dist $+$ tree.radius:
  $i,$ dist $\leftarrow$update(i, nn_search(x, tree))

Return i

Complexity

Construction

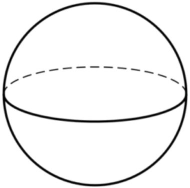
- $O(n \log n)$ for both kd-trees and ball-trees

Query

- $O(n)$ with brute force
- $O(\log n)$ with ball-trees
- $O(\log n)$ (low dimension) up to $O(n)$ (high dimension) for kd-trees

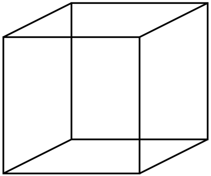
Hyperspheres vs Hypercubes

Unit hypersphere



$$\text{volume} = \begin{cases} \frac{\pi^p}{p!} & d = 2p \\ \frac{2^{p+1}\pi^p}{(2p+1)!!} & d = 2p+1 \end{cases}$$

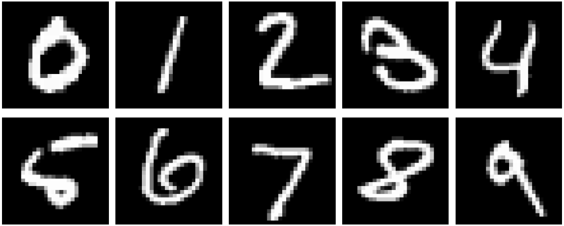
Unit hypercube



$$\text{volume} = 1$$

Example: MNIST dataset

60,000 images (train set) of size 28×28
10,000 images (test set)



Speed-up on nearest-neighbor search (test set):

- $\times 5$ with kd-trees
- $\times 10$ with ball-trees

with leaf size $= 10$ for both

Outline

1. Metrics
2. Thinning
3. Searching
4. **Clustering** $\rightarrow$ nn-graph
5. Sketching
6. Embedding

k-nearest neighbor graph

Data $X \in \mathbb{R}^{n \times d}$

k-nn graph

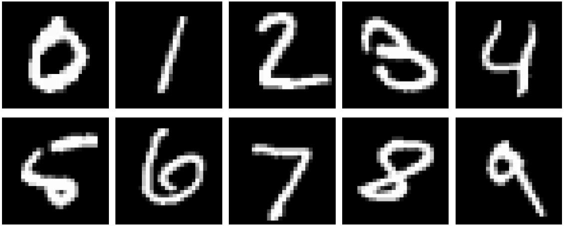
Directed graph

Link $i \to j$ if j is one of the k-nearest neighbors of i

Complexity: $O(n \log n)$ using a kd tree or a ball tree,
possibly after dimension reduction

Example: MNIST dataset

60,000 images (train set) of size 28×28
10,000 images (test set)

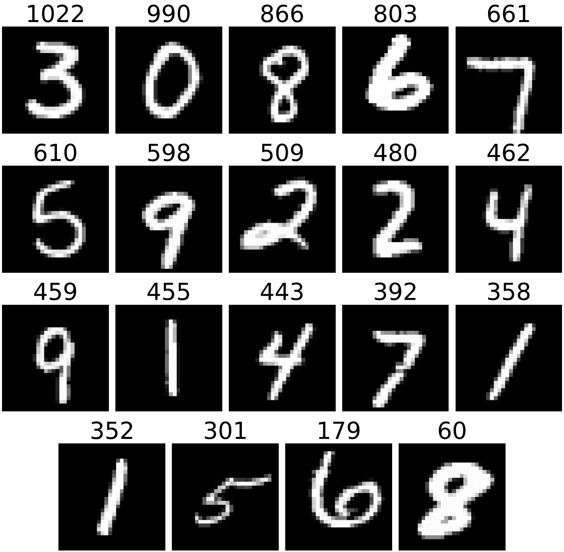


Clustering of test set embedded in dimension 64 (SVD)

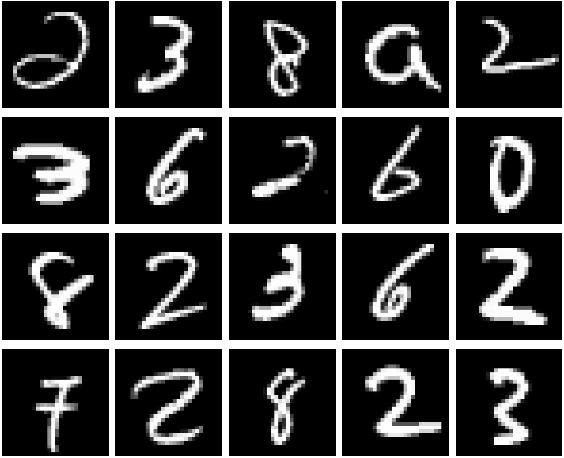
Precision after alignment:

- 59% using k-means ($k = 10$)
- 74% using k-means ($k = 20$)
- 94% using 5-nn graph (Louvain clustering) $\rightarrow$ 19 clusters

Clusters of the 5-nn graph



Some outliers



Summary

Part I

- Metrics
- Thinning $\rightarrow$ CNN
- Searching $\rightarrow$ kd trees*, ball trees*
- Clustering $\rightarrow$ nn-graph*

Part II

- Sketching $\rightarrow$ MinHash, random projection
- Embedding $\rightarrow$ PCA*, Halko, NCA*

* Available in `scikit-learn`