Approximate nearest neighbour search

Why do we need approximate k-NN?

1

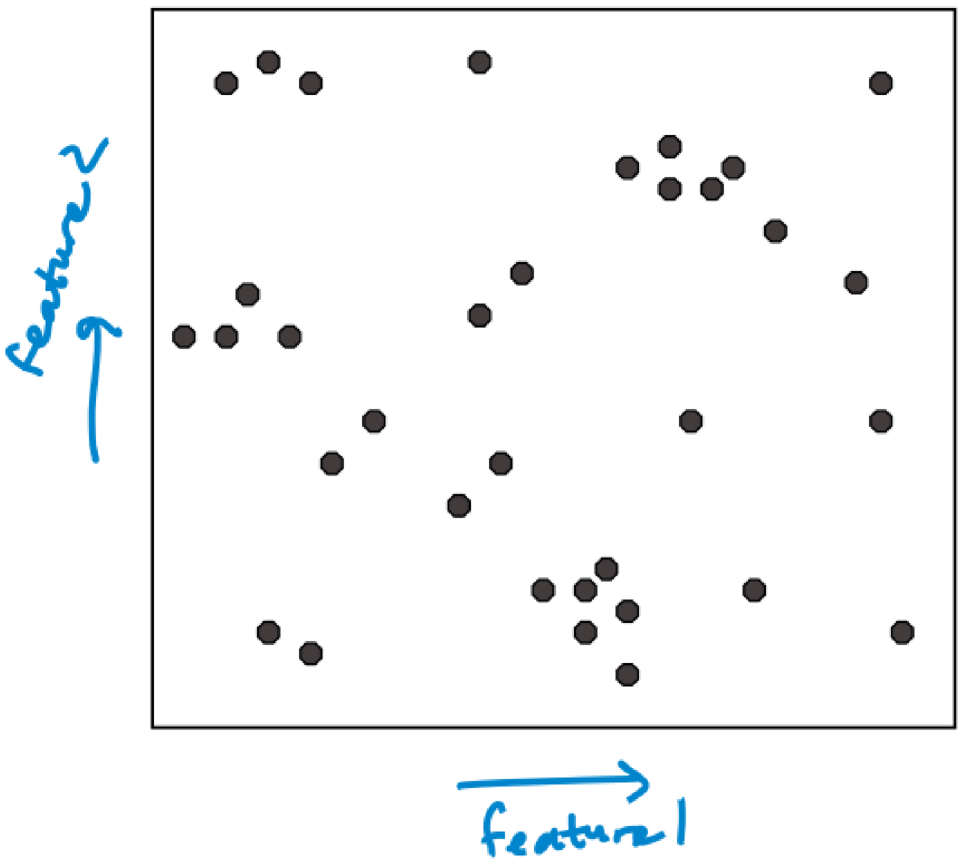
k-NN is slow. O(n) for high-dimensional data

2

We do not always need exact nearest neighbour

KD-trees

KD-tree construction



Start with a list of d-dimensional points.

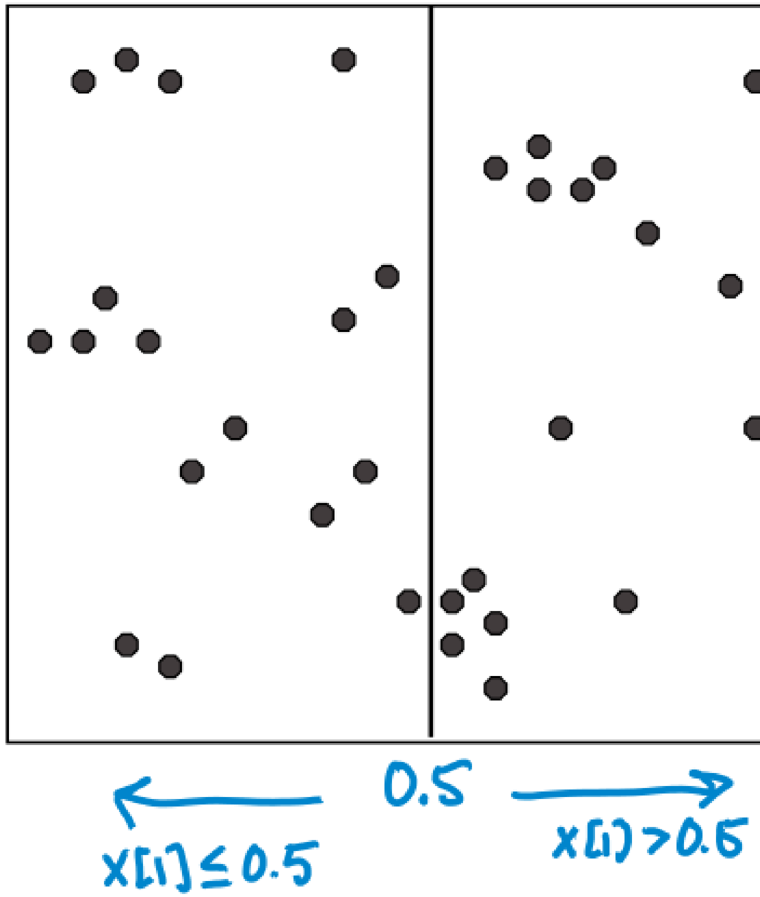
Pt	**x**[1]	**x**[2]
1	0.00	0.00
2	1.00	4.31
3	0.13	2.85
...	...	...

obs. indices

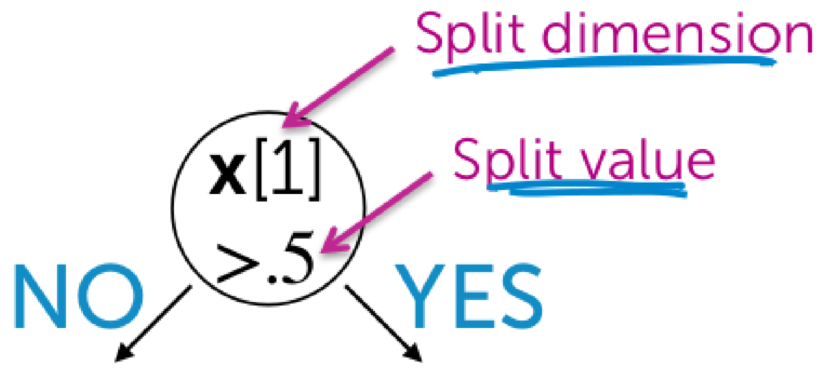
feat. 1 (word 1)

feat. 2 (word 2)

KD-tree construction



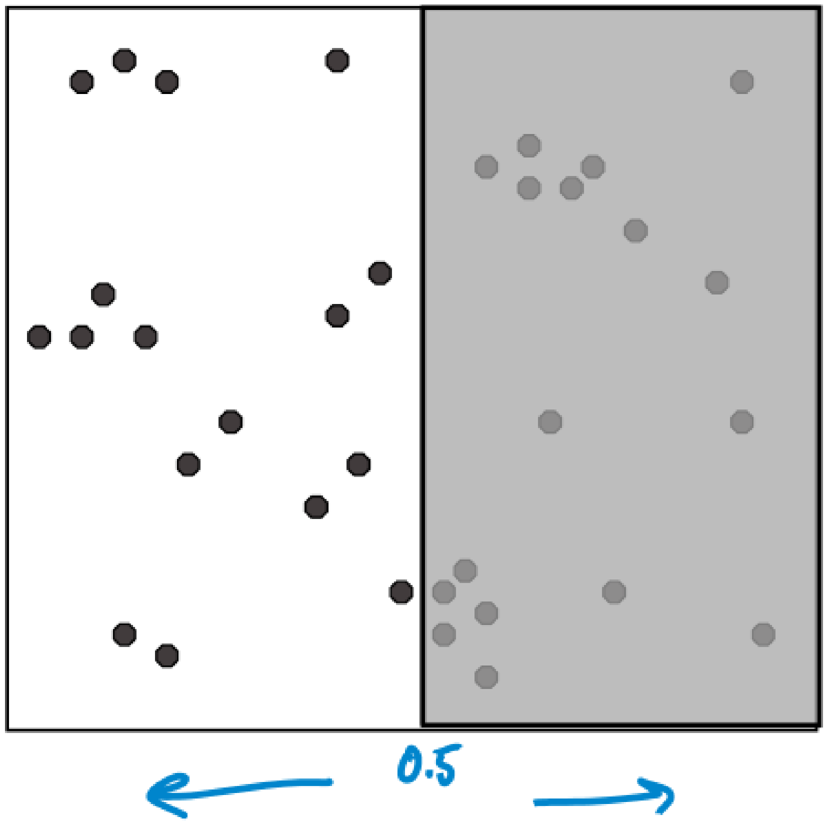
Split points into 2 groups



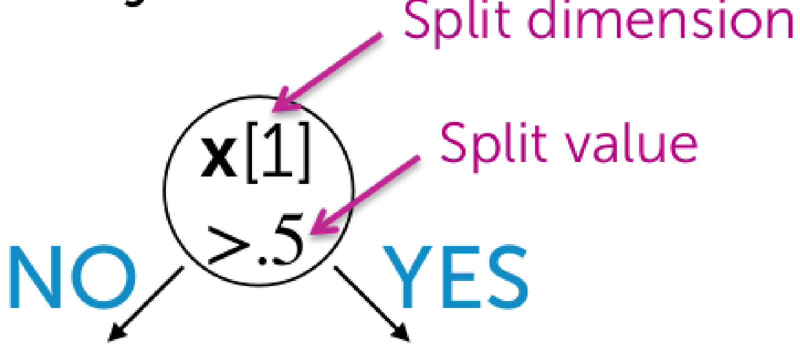
Pt	**x**[1]	**x**[2]
1	0.00	0.00
3	0.13	2.85
...	...	...

Pt	**x**[1]	**x**[2]
2	1.00	4.31
...	...	...

KD-tree construction



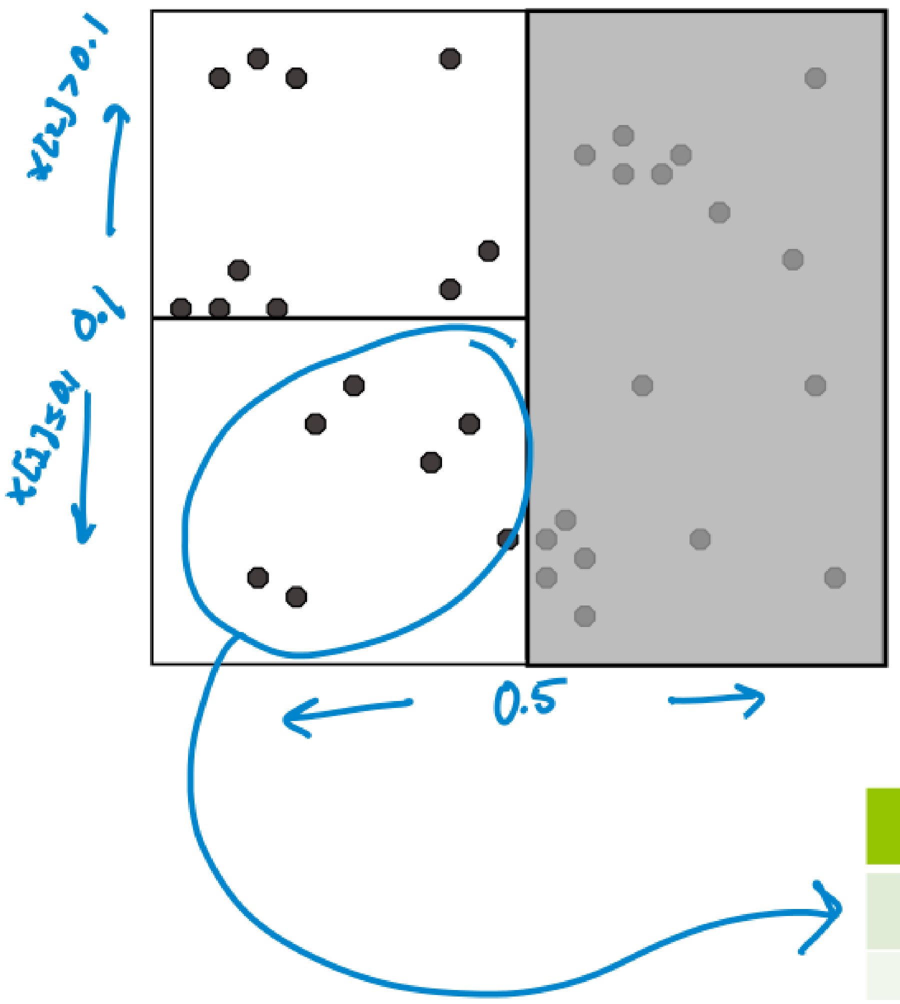
Recurse on each group separately



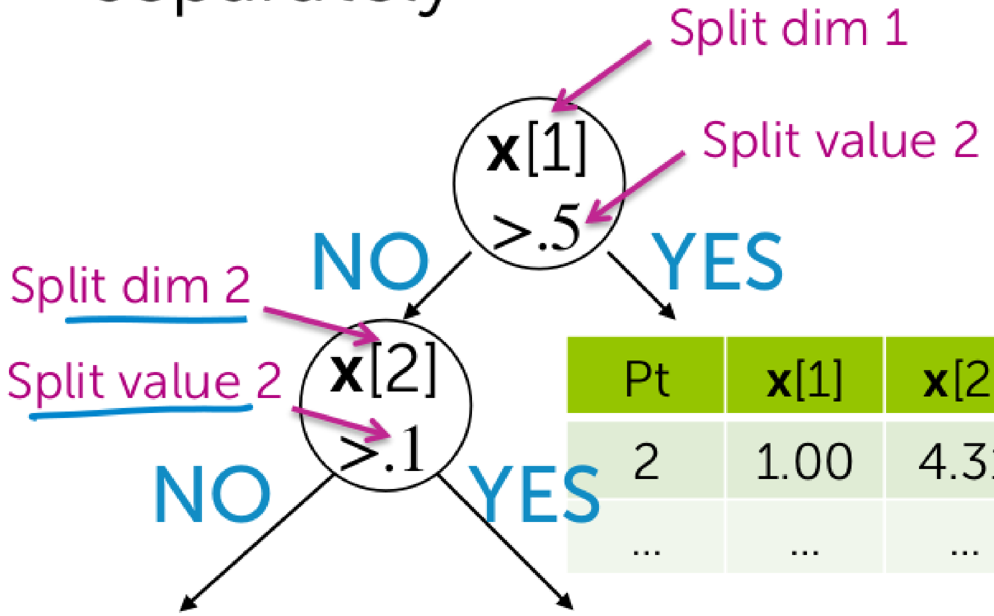
Pt	**x**[1]	**x**[2]
1	0.00	0.00
3	0.13	2.85
...	...	...

Pt	**x**[1]	**x**[2]
2	1.00	4.31
...	...	...

KD-tree construction



Recurse on each group separately

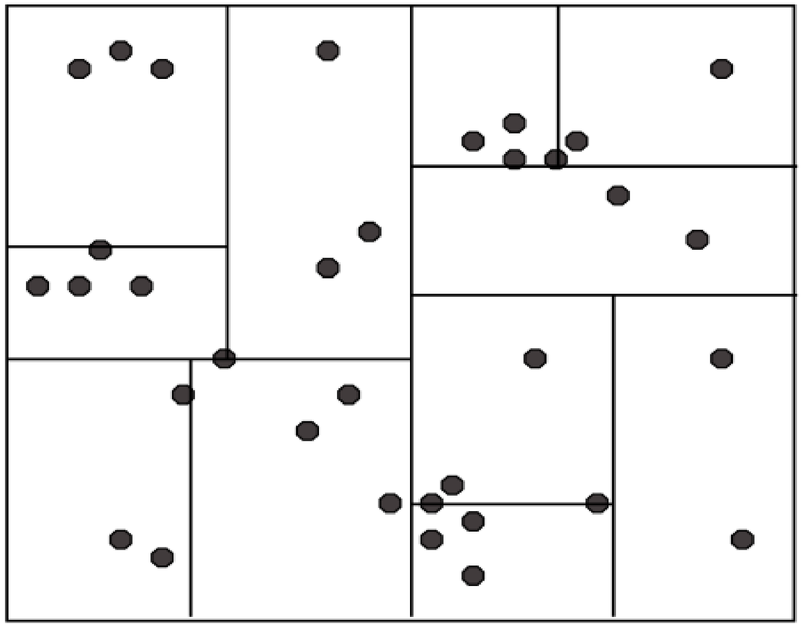


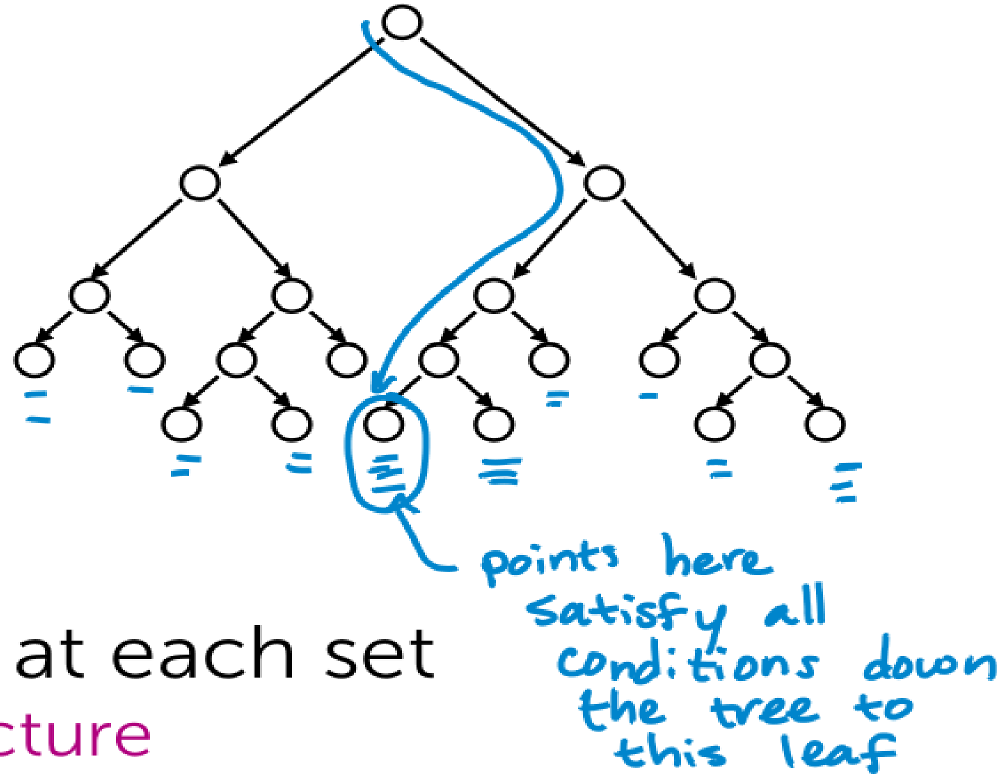
Pt	**x**[1]	**x**[2]
2	1.00	4.31
…	…	…

Pt	**x**[1]	**x**[2]
3	0.13	2.85

Pt	**x**[1]	**x**[2]
1	0.00	0.00

KD-tree construction

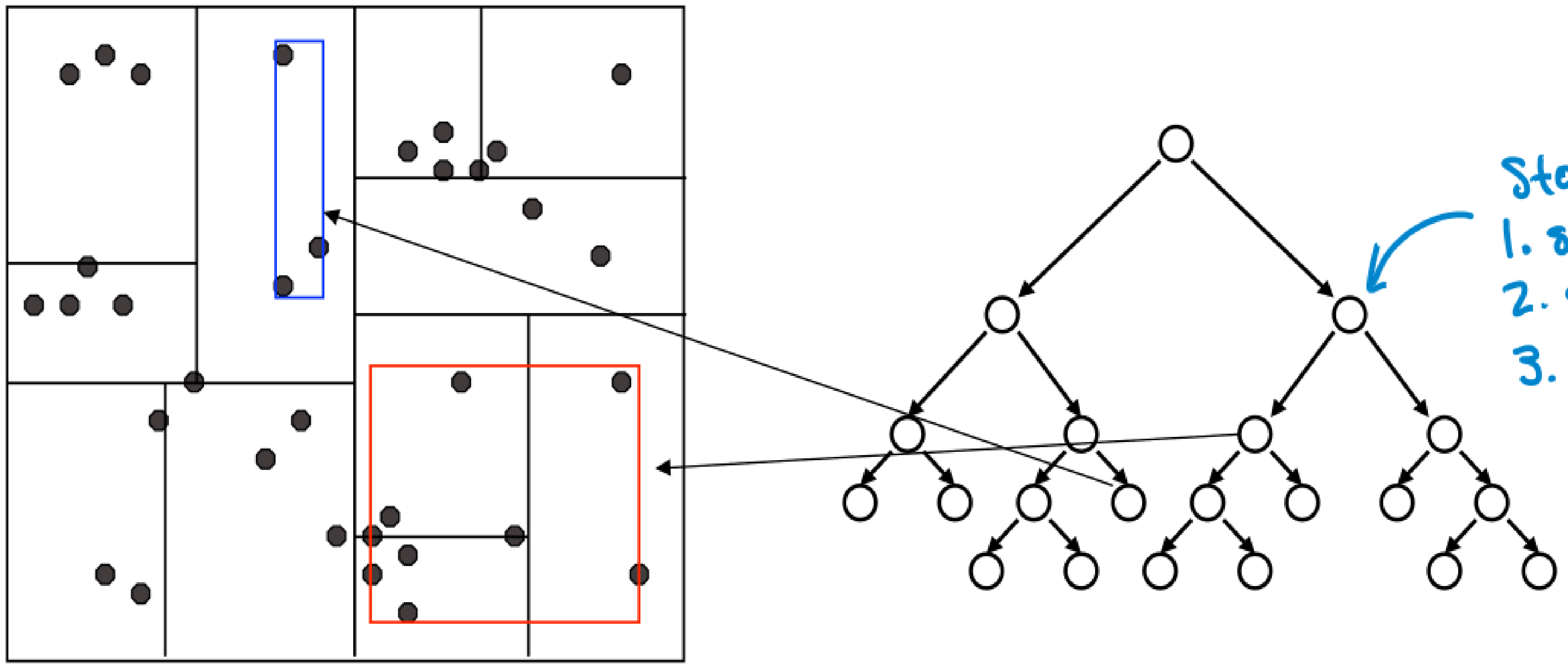




Continue splitting points at each set
- Creates a binary tree structure

Each leaf node contains a list of points

KD-tree construction



Store:
1. split dim?
2. split value
3. bounding box that is as small as possible while containing pts

KD-tree construction choice

Use heuristics to make splitting decisions:

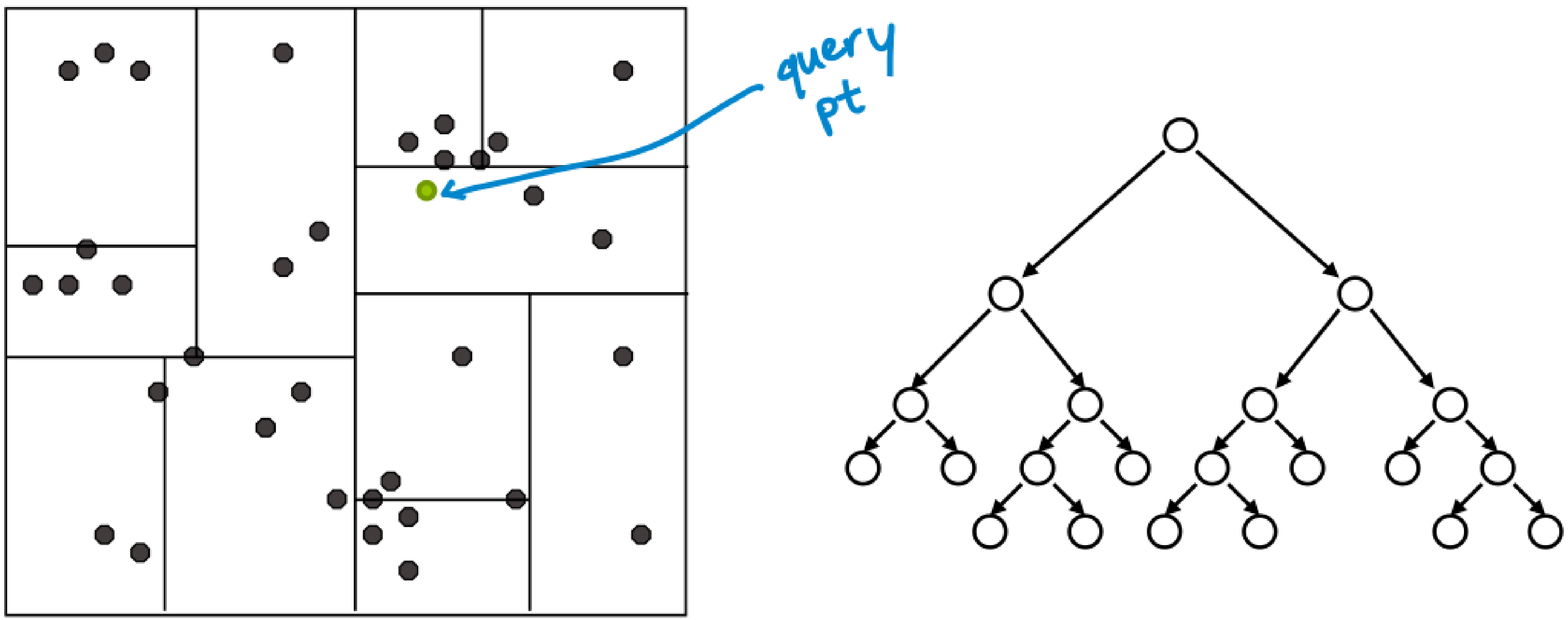
- Which dimension do we split along?

- Which value do we split at?

- When do we stop?

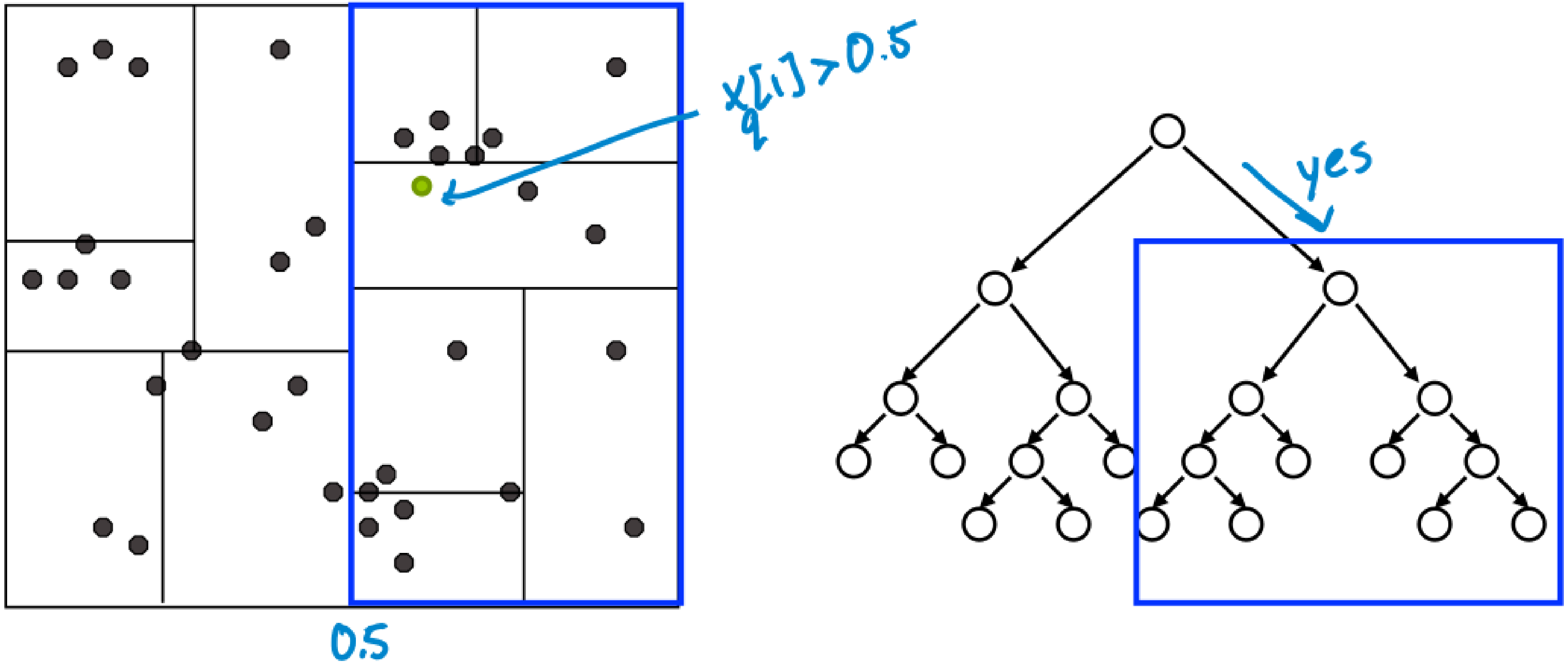
NN search with KD-trees

NN search with KD-trees



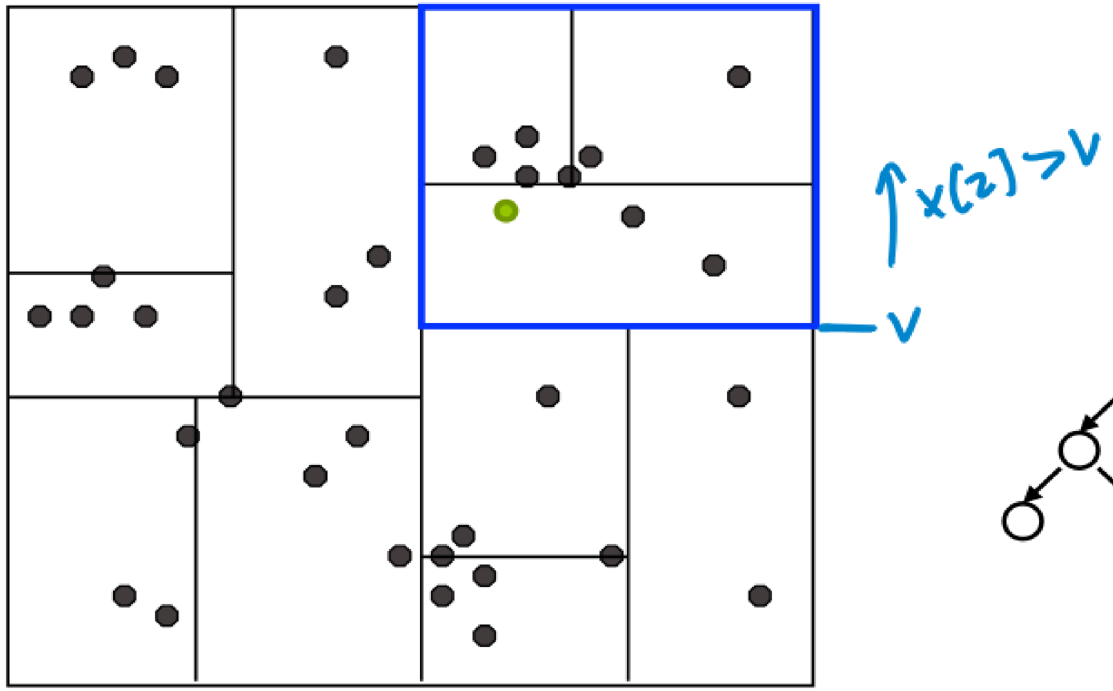
Traverse tree looking for nearest neighbor to query point

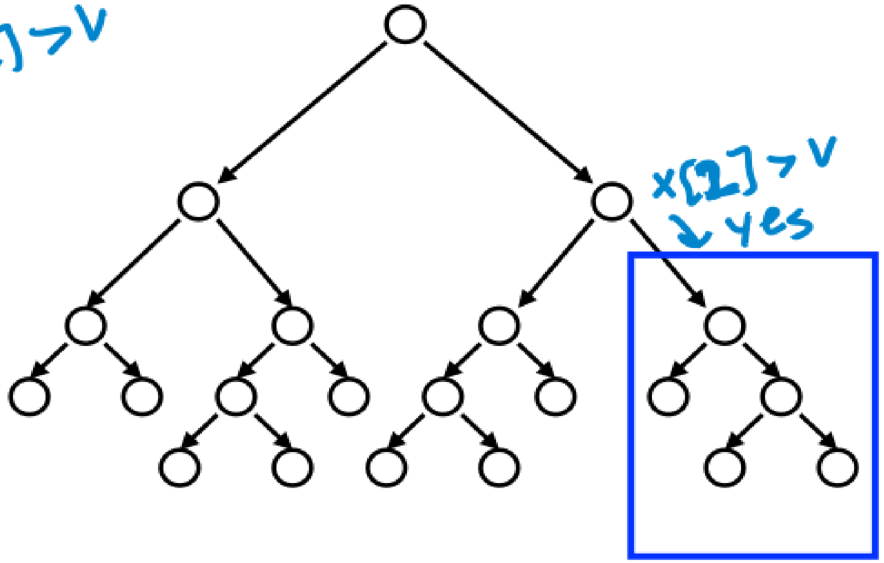
NN search with KD-trees



1. Start by exploring leaf node containing query point

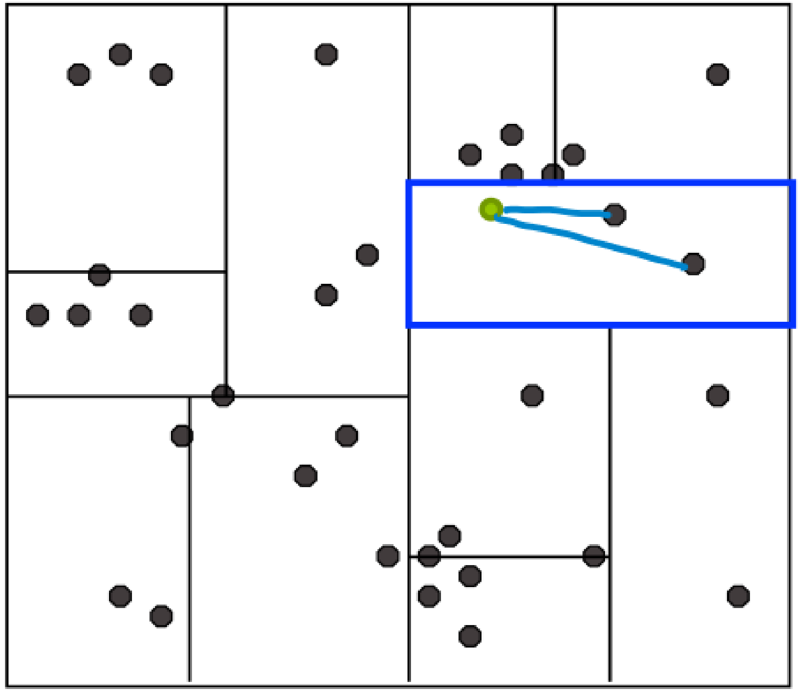
NN search with KD-trees

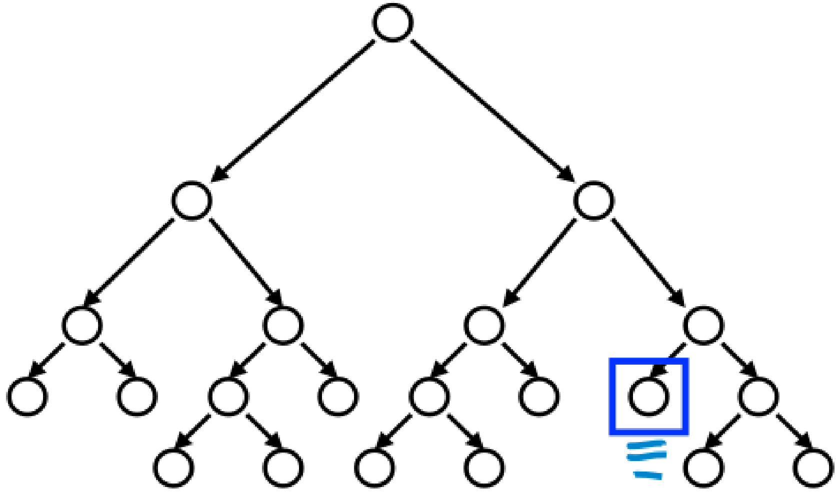




1. Start by exploring leaf node containing query point

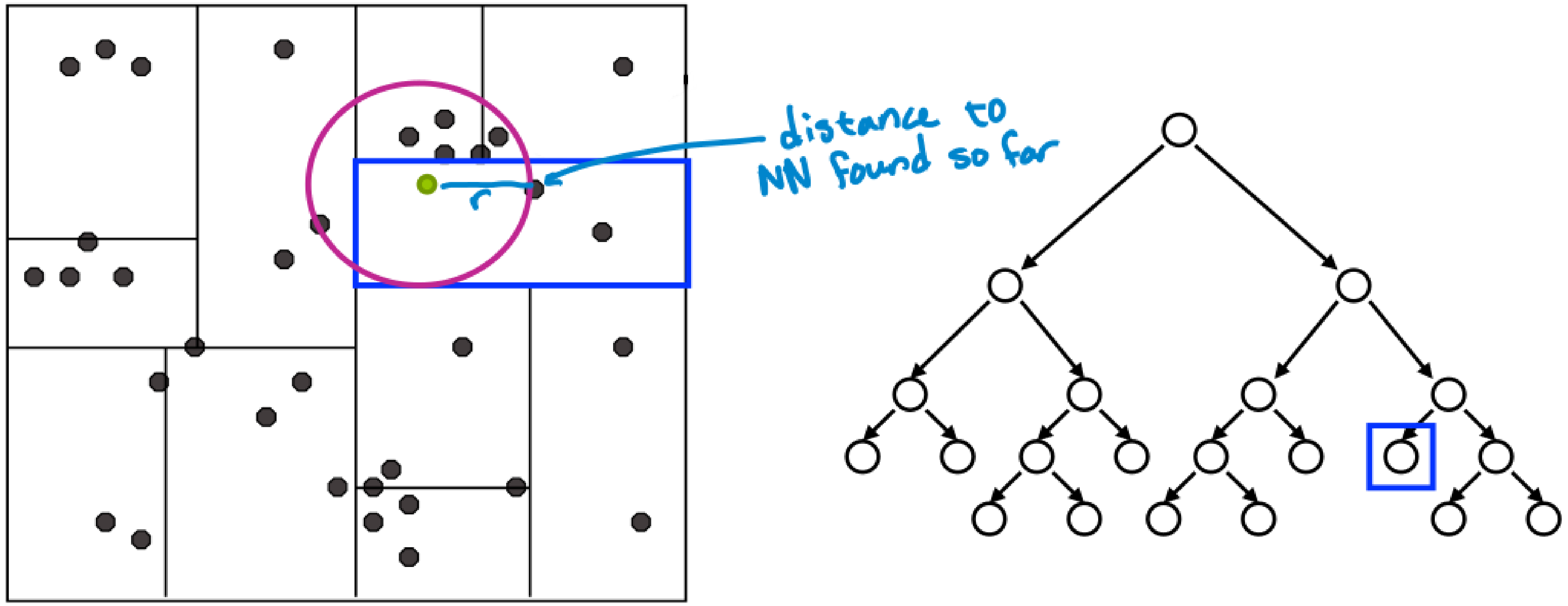
NN search with KD-trees





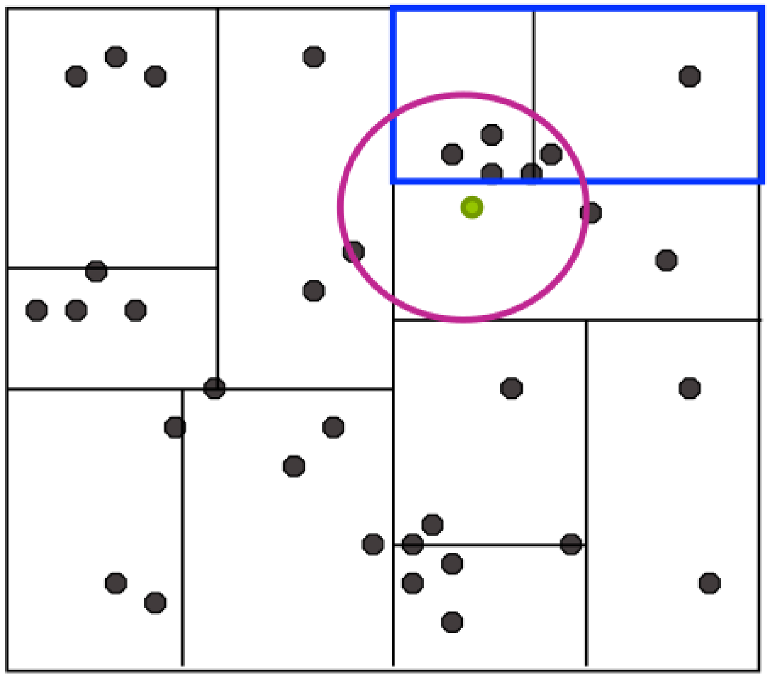
1. Start by exploring leaf node containing query point

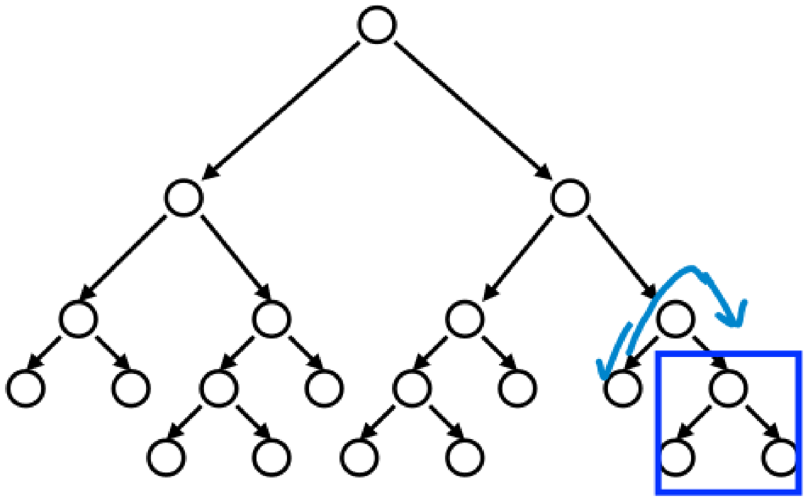
NN search with KD-trees



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

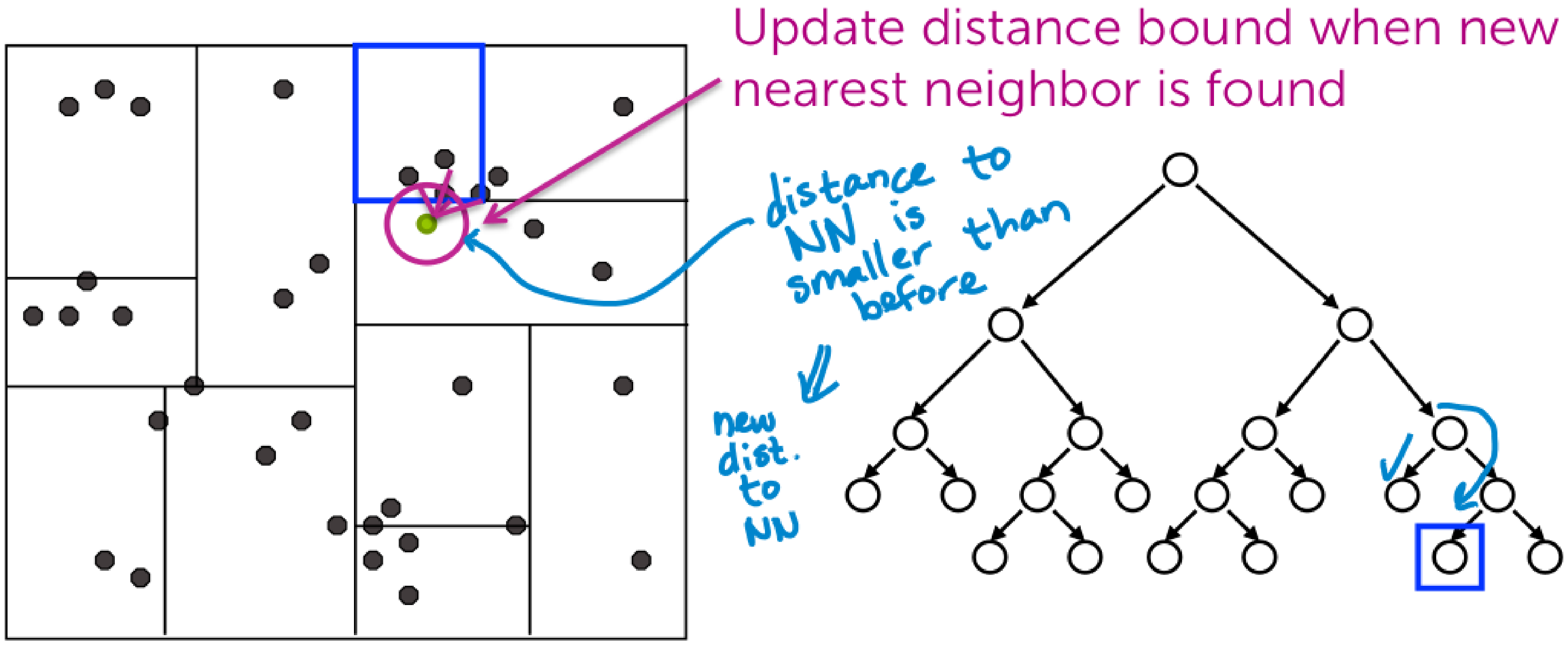
NN search with KD-trees





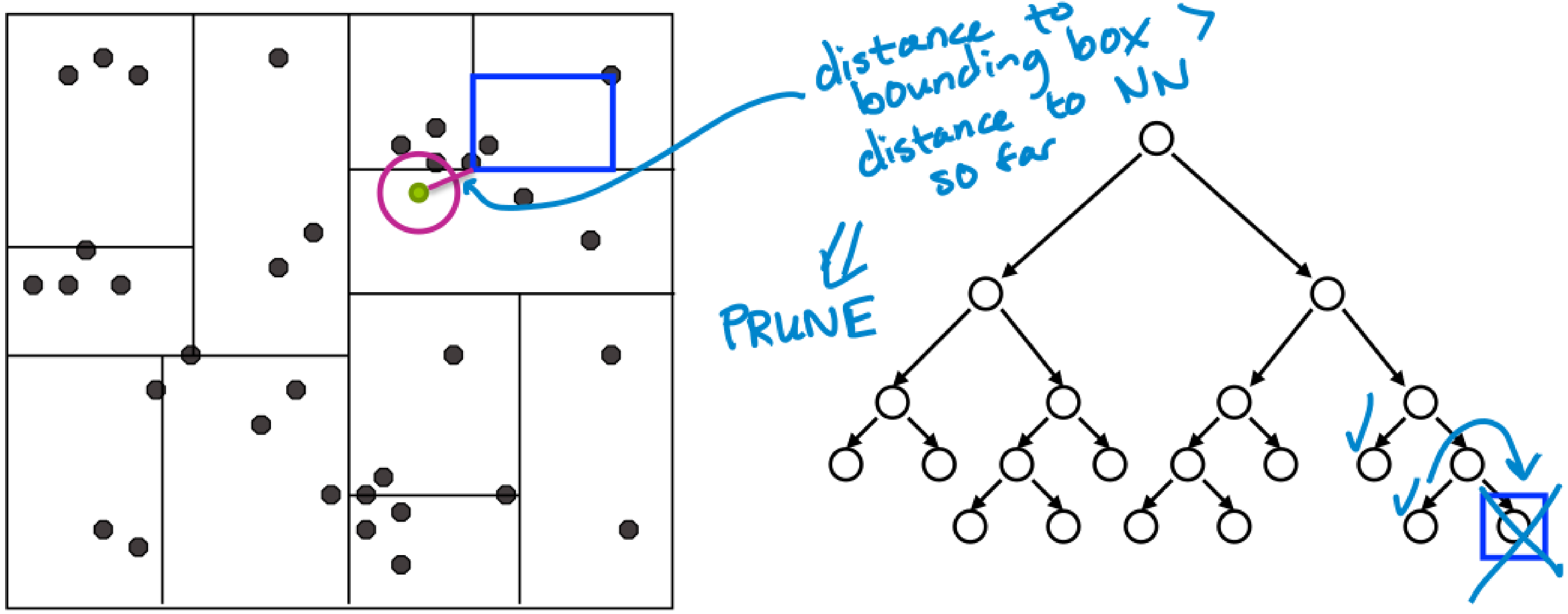
1. Start by exploring leaf node containing query point
2. Compute distance to each other point at leaf node
3. Backtrack and try other branch at each node visited

NN search with KD-trees



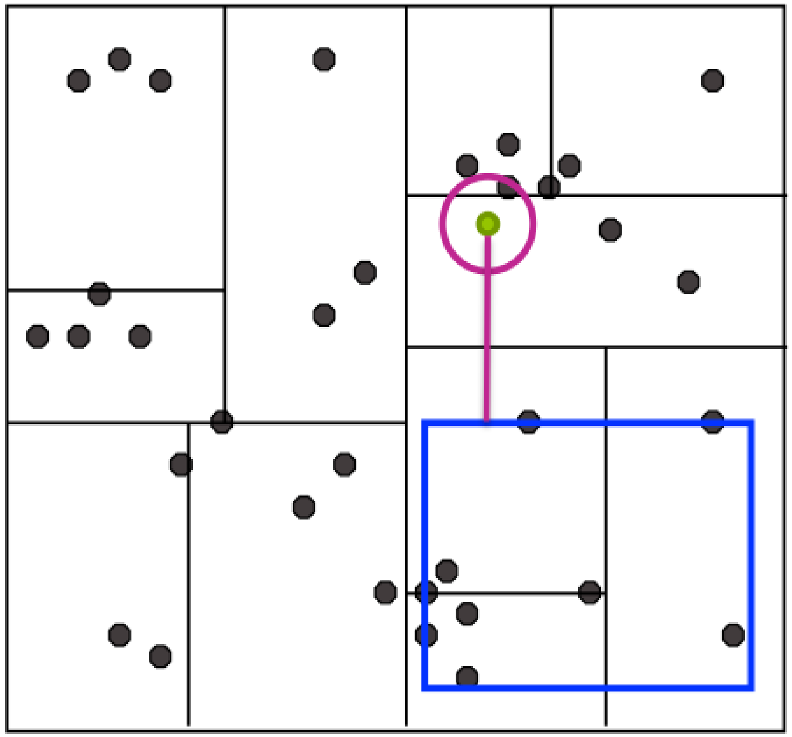
1. Start by exploring leaf node containing query point
2. Compute distance to each other point at leaf node
3. Backtrack and try other branch at each node visited

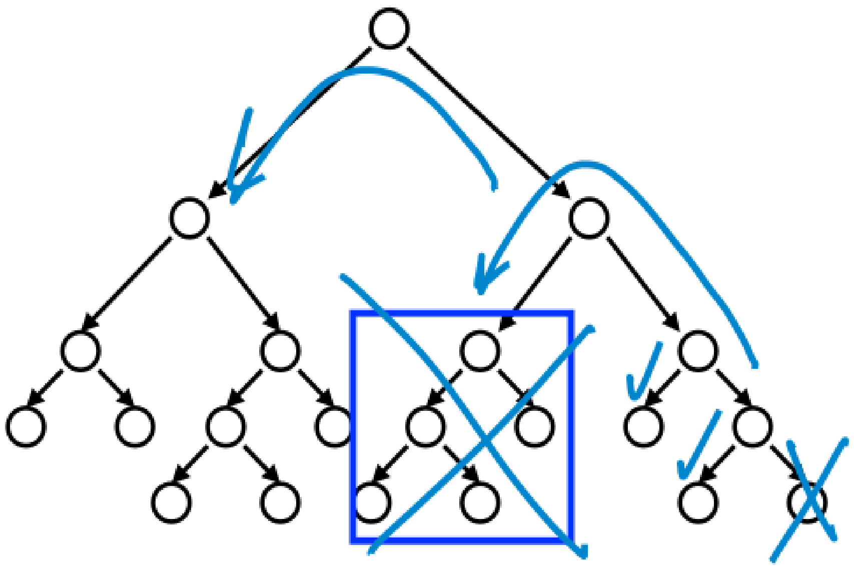
NN search with KD-trees



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

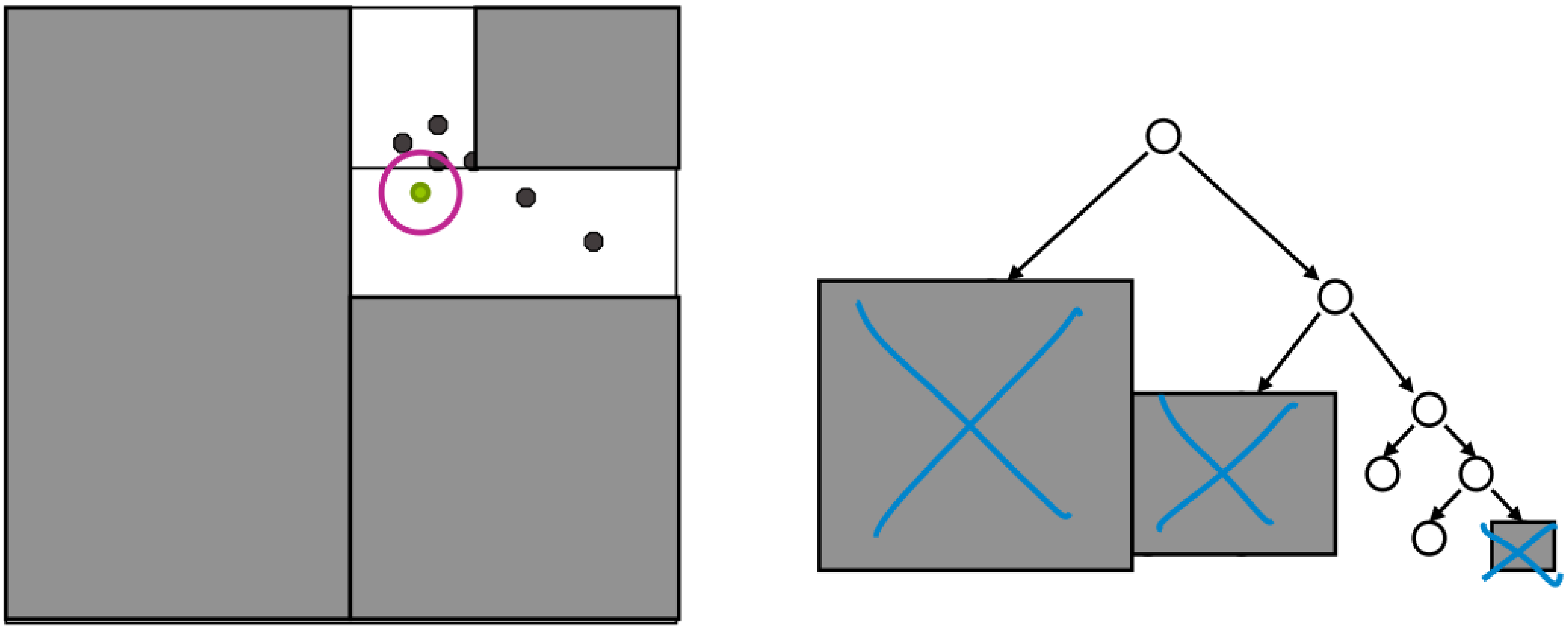
NN search with KD-trees





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

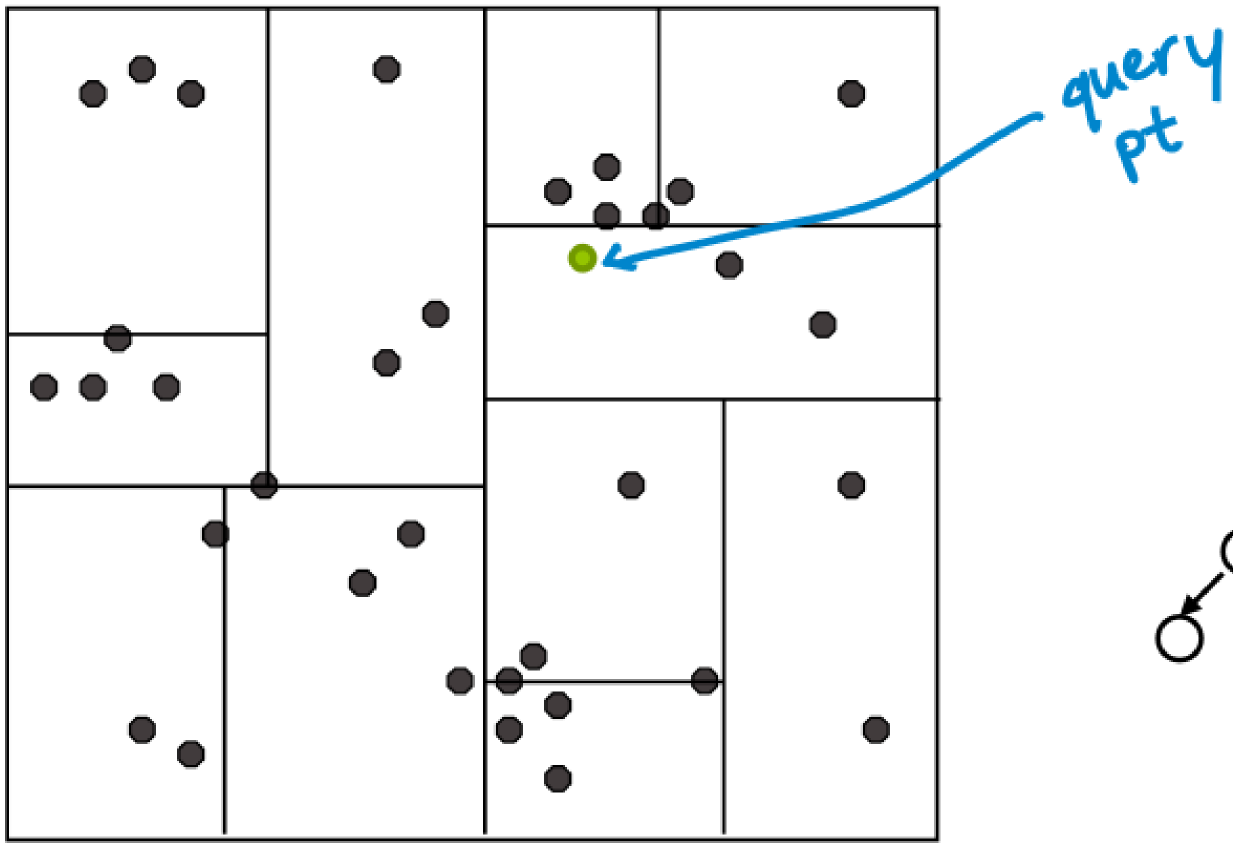
NN search with KD-trees

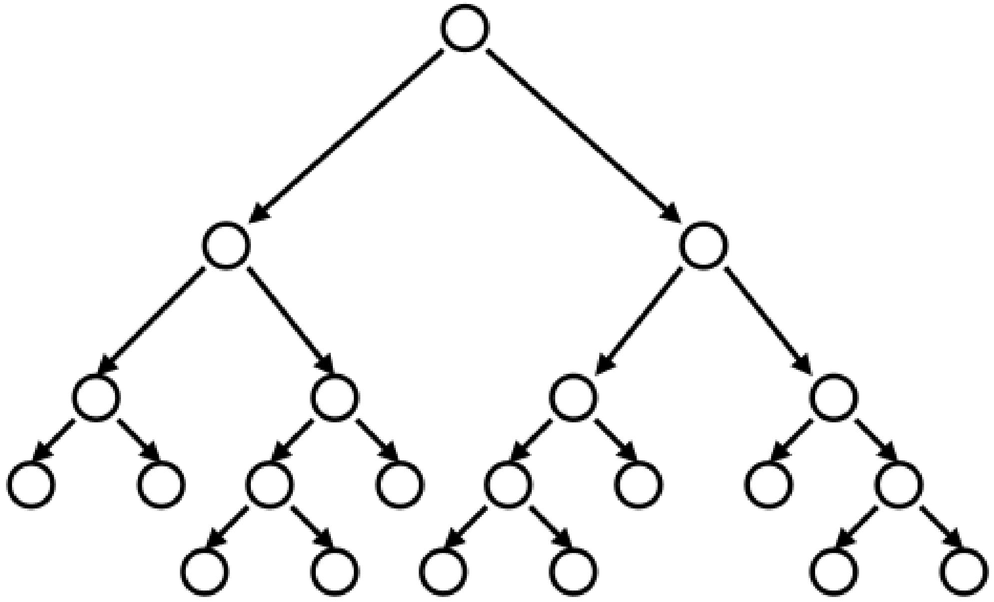


Use distance bound and bounding box of each node to prune parts of tree that cannot include nearest neighbor

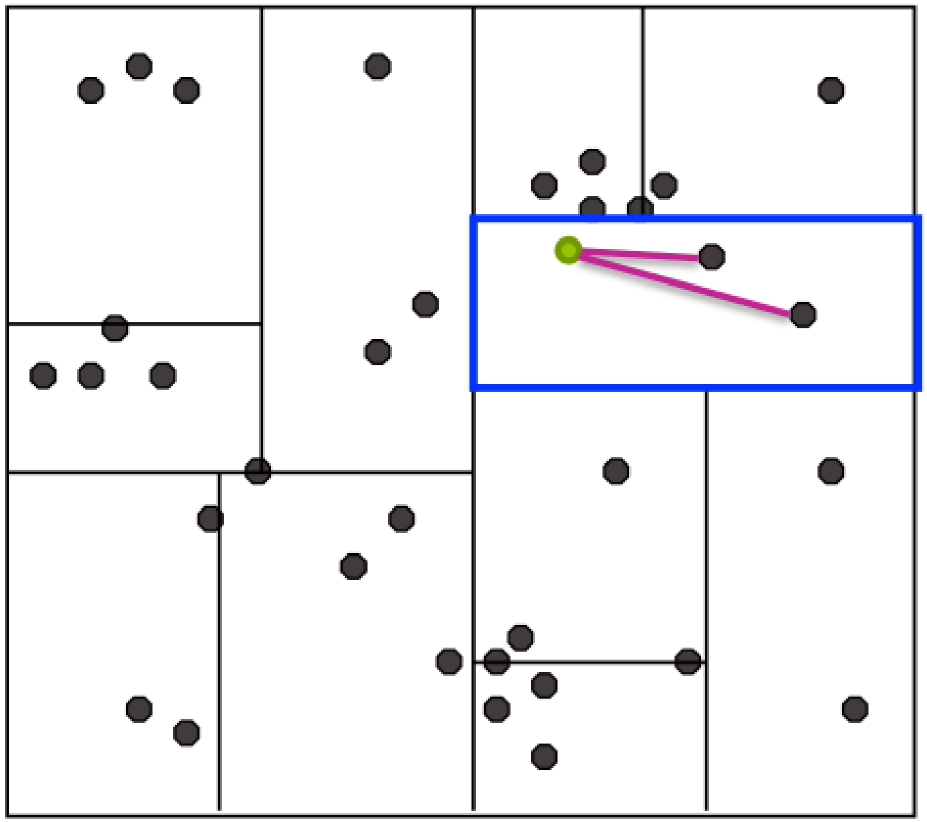
Approximate NN search with KD-trees

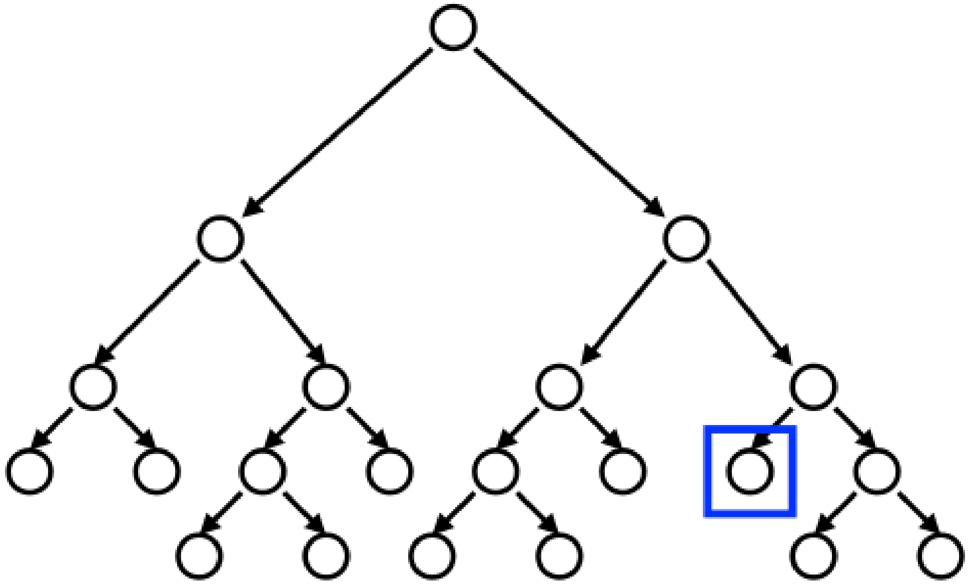
Approximate NN search with KD-trees



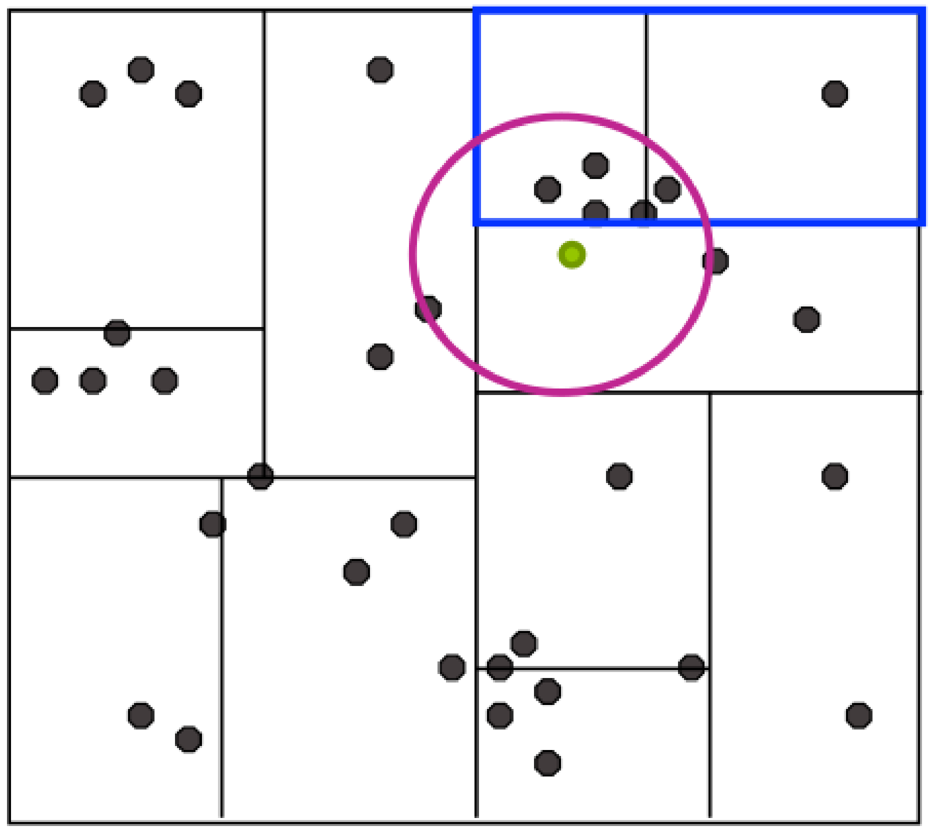


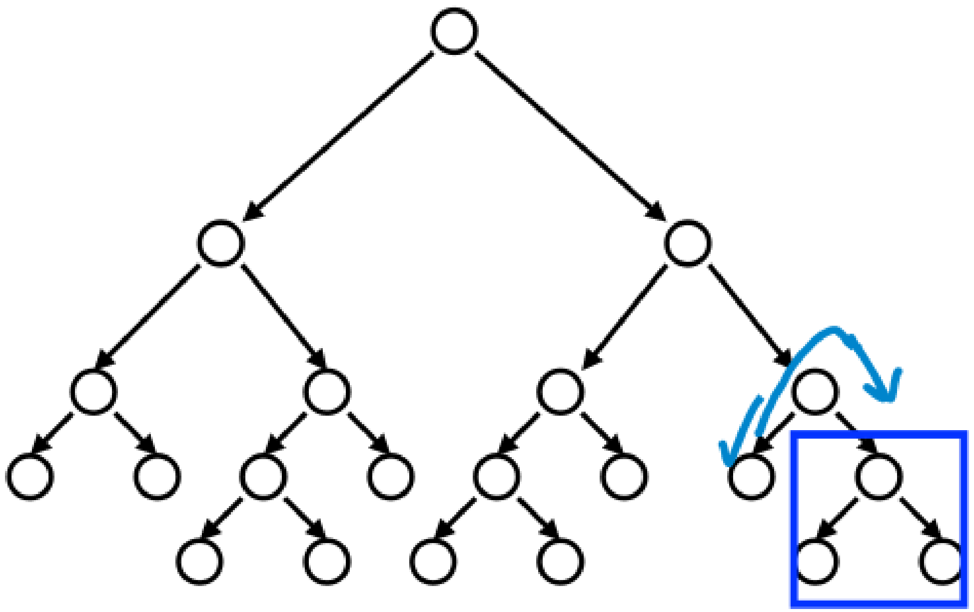
Approximate NN search with KD-trees





Approximate NN search with KD-trees

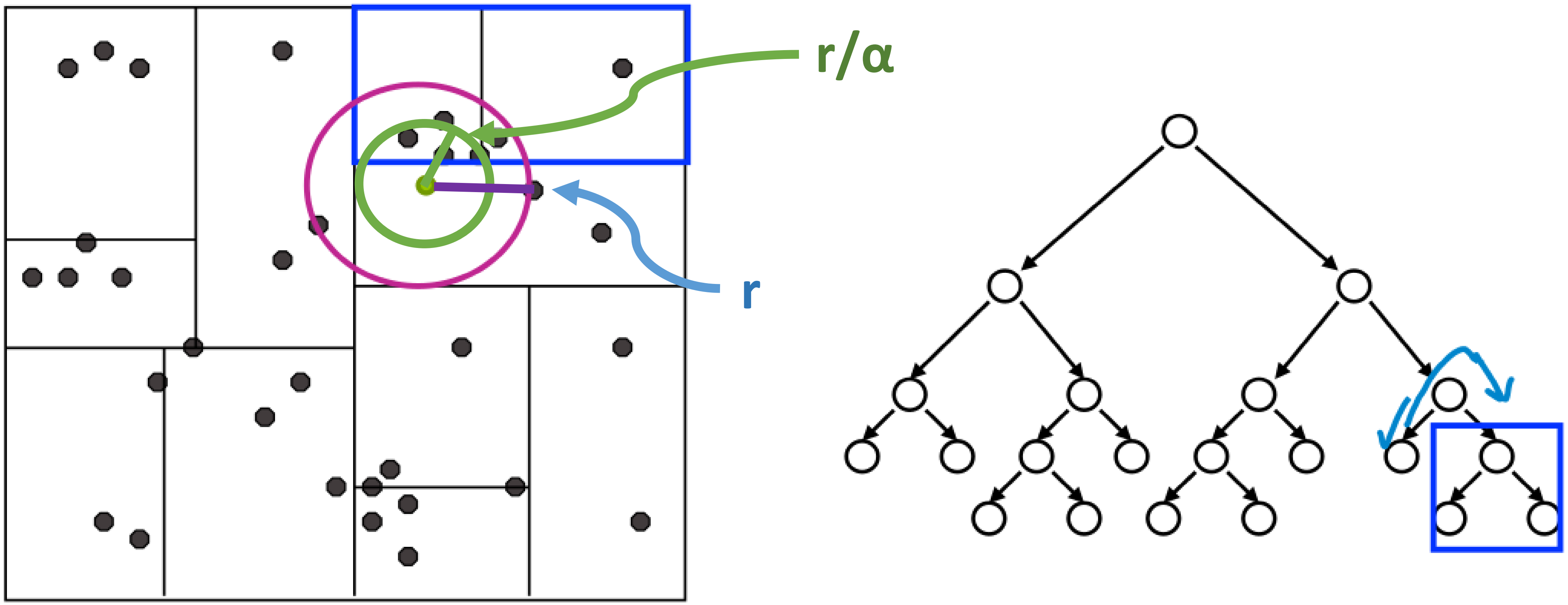




Approximate NN search with KD-trees

- is the distance to the nearest neighbour

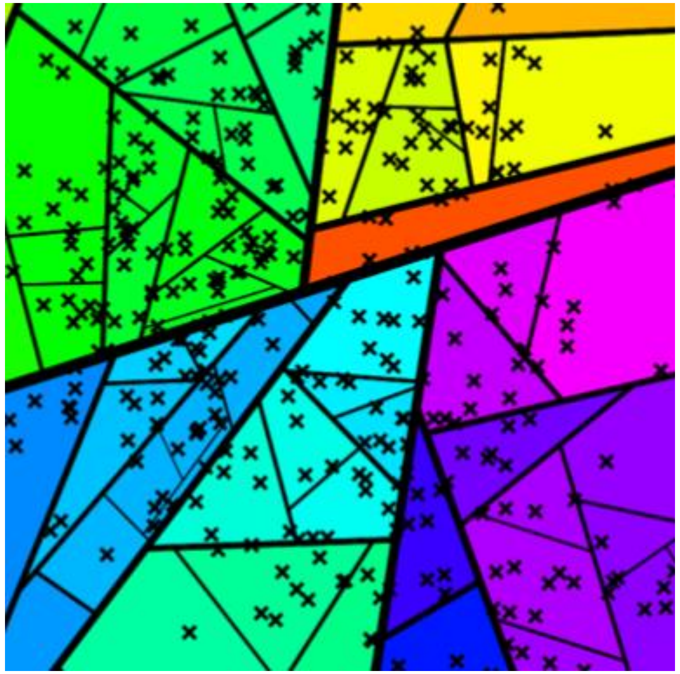
We will seek for a point with distance **≤ αr**

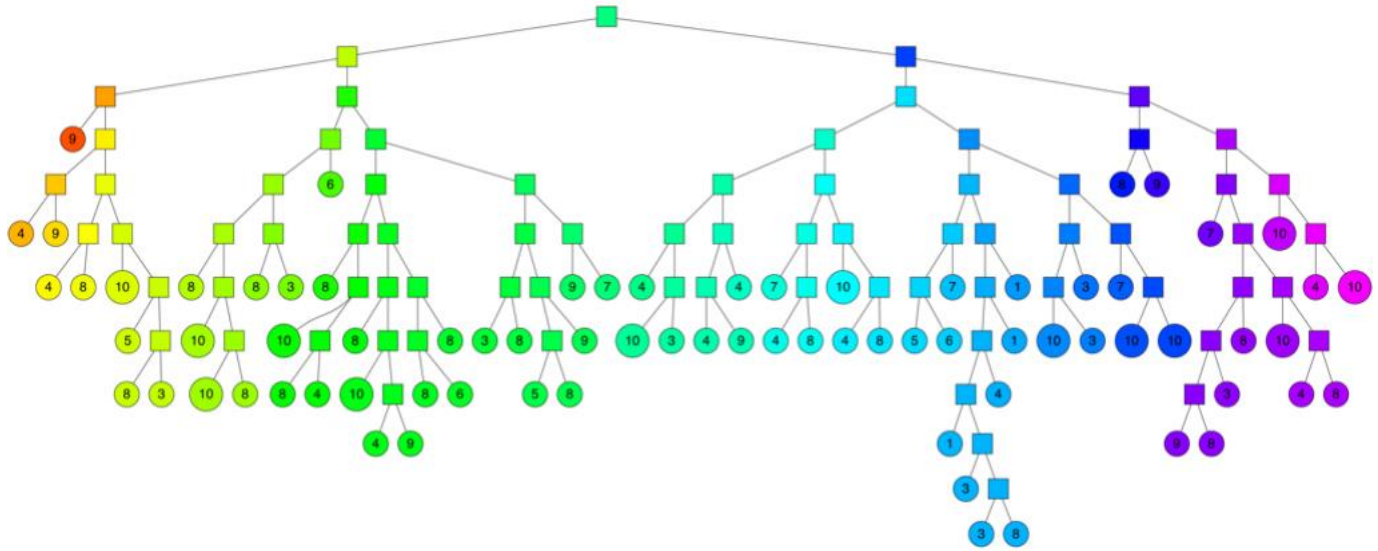


Approximate randomized KD-tree algorithm

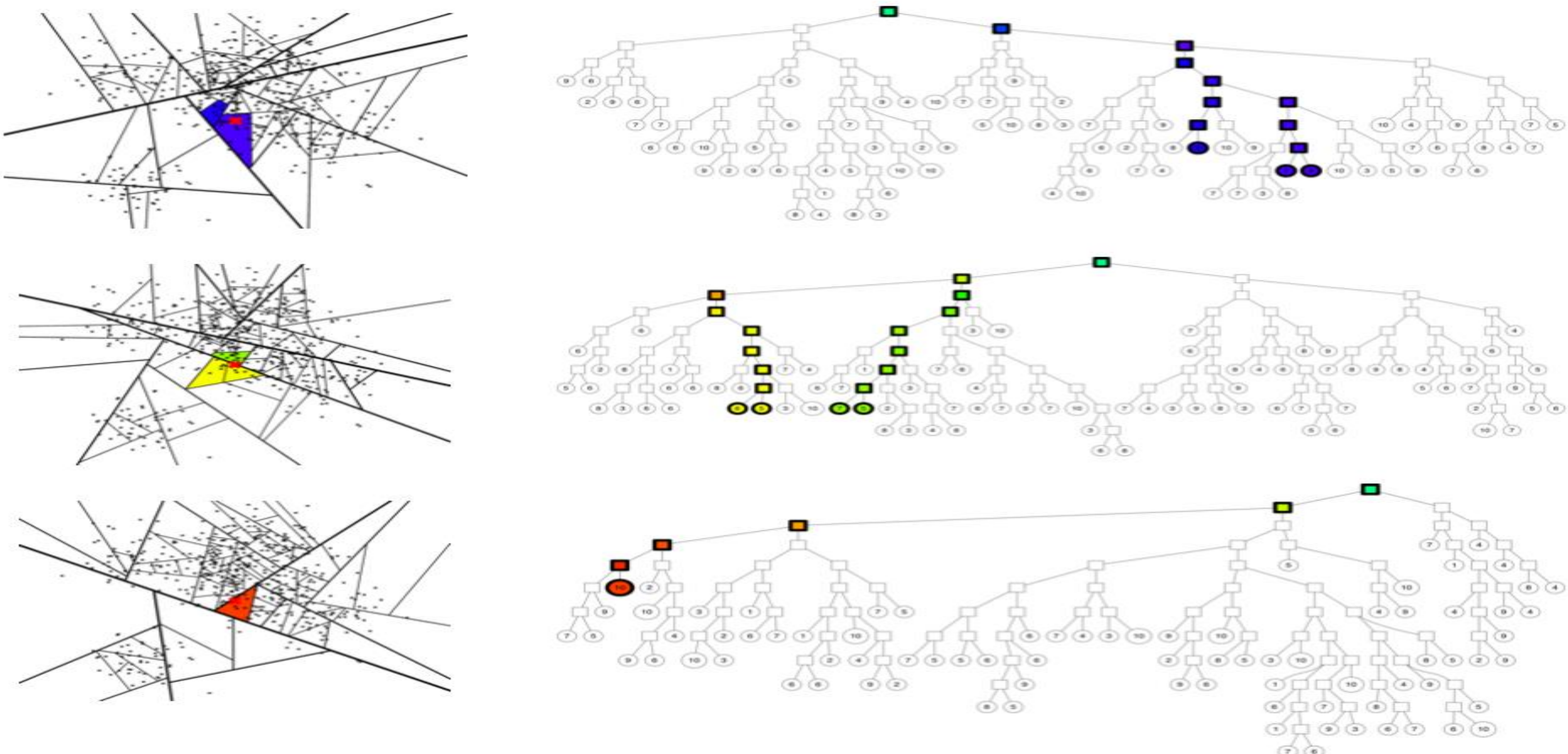
Approximate randomized KD-tree algorithm

- Choose the split dimension randomly from the first D dimensions on which data has the greatest variance and split it in half





Build a lot trees



Approximate randomized KD-tree algorithm

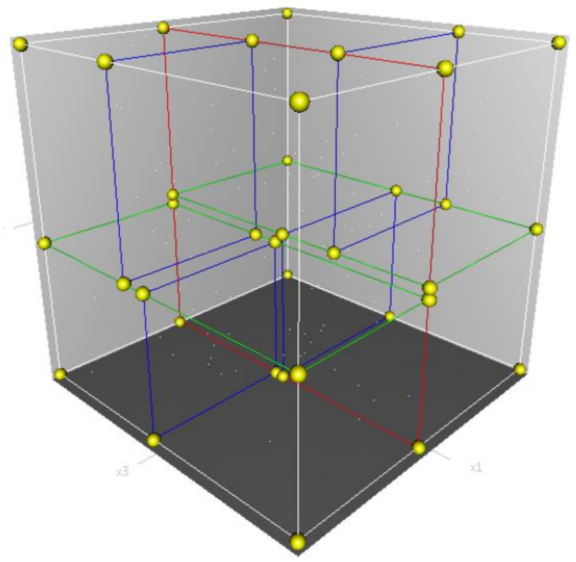
- When searching the trees, a single priority queue is maintained across all the randomized trees
- Search is ordered by increasing distance to each bin boundary
- The degree of approximation is determined by examining a fixed number of leaf nodes, at which point the search is terminated and the best candidates returned.

The hierarchical k-means tree algorithm

Why do we need another method when we have KD-trees?

KD-trees have following drawbacks

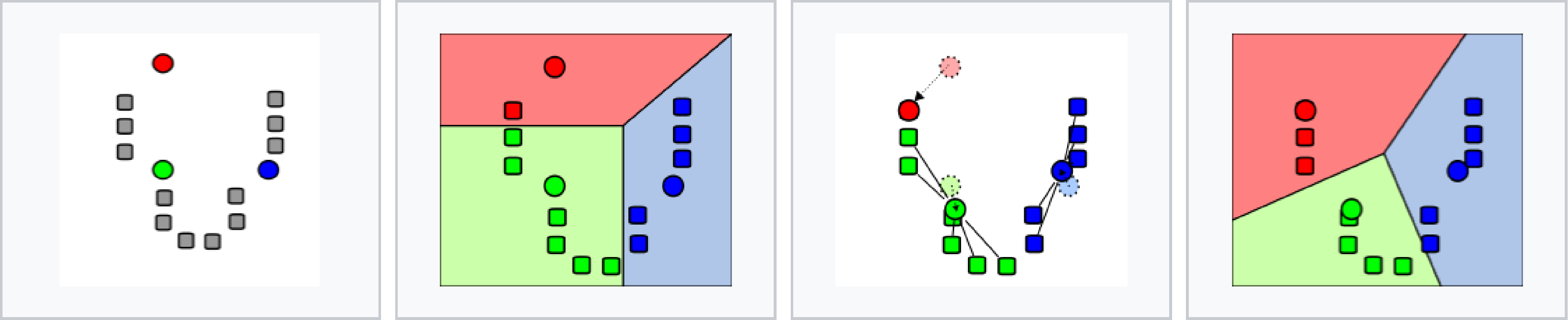
1. It is hard to construc them
2. Works slow on high-dimensional data



What is k-mean clustering?

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \longrightarrow \min_{\mu}$$

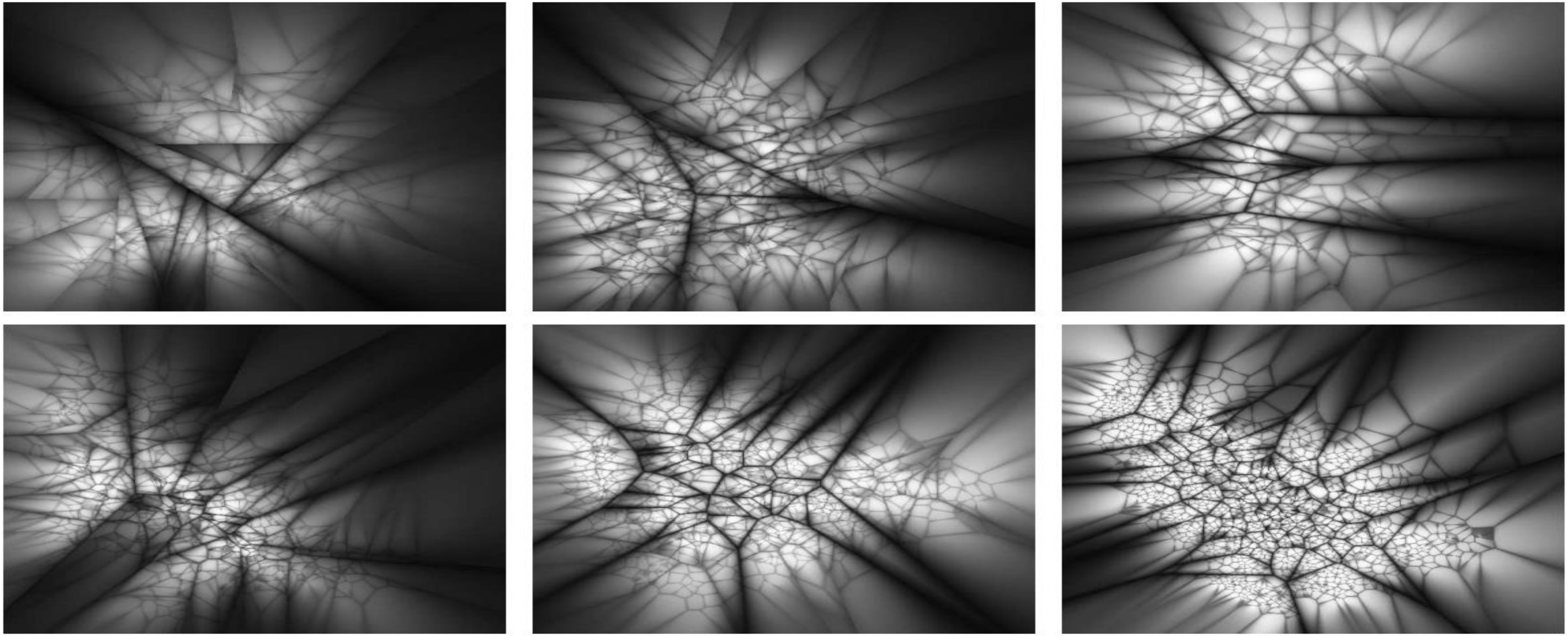
k– the number of clasters, S_i – clusters $i = 1, 2, \ldots$, μ_i - clusters centres $i = 1, 2, \ldots$



Constructing k-mean tree

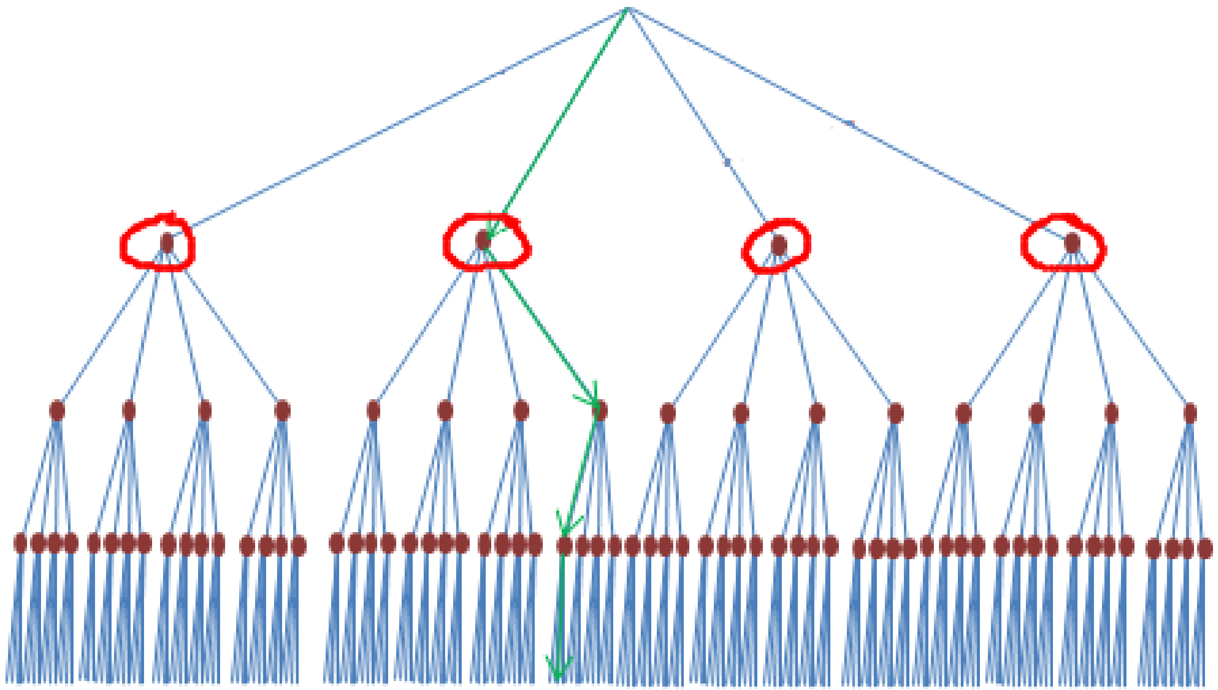
- The hierarchical k-means tree is constructed by splitting the data points at each level into K distinct regions using a k-means clustering, and then applying the same method recursively to the points in each region.
- We stop the recursion when the number of points in a region is smaller than K.

Hierarchical k-means tree examples



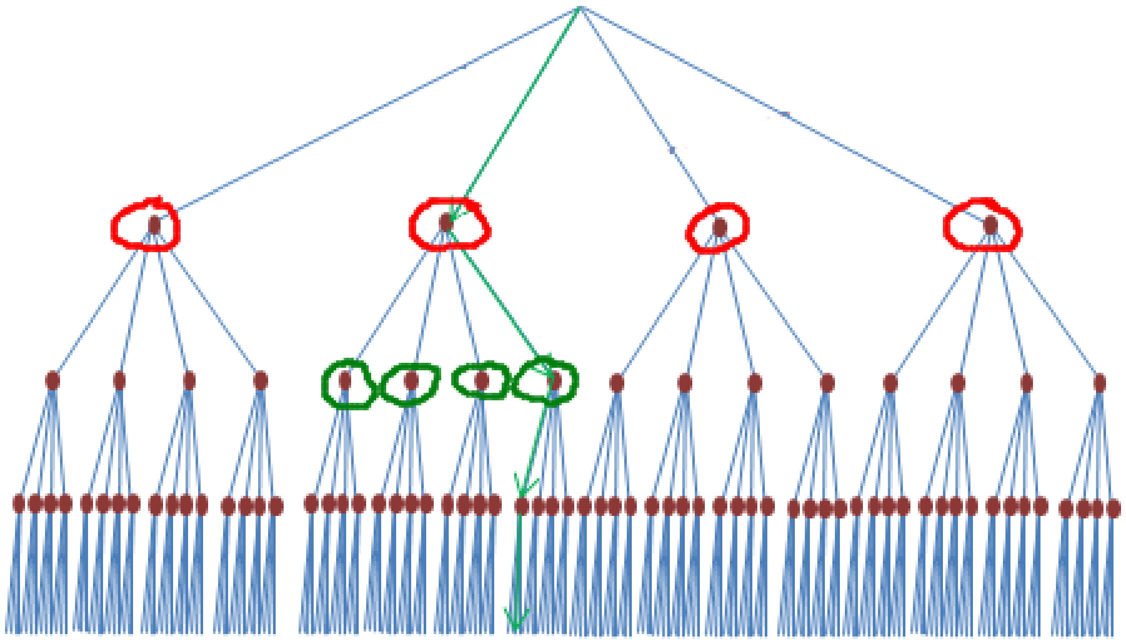
Search NN in hierarchical k-means tree

The algorithm initially put all branches in priority queue



Search NN in hierarchical k-means tree

Next, it extracts from the priority queue the branch that has the closest center to the query point and it restarts the tree traversal from that branch.



Search NN in hierarchical k-means tree

- In each traversal the algorithm keeps adding to the priority queue the unexplored branches along the path.
- The degree of approximation is specified in the same way as for the randomized kd-trees, by stopping the search early after a predetermined number of leaf nodes (dataset points) have been examined.

FANN method and FLANN library

FANN method quick introduction

- FANN method uses approximate randomized KD-trees and hierarchical k-means trees.
- It choose algorithm for use depending on data structure and precision requirements
- FLANN libriry implements FANN

FANN method

- By considering the algorithm itself as a parameter of a generic nearest neighbor search routine, the problem is reduced to determining the parameters that give the best solution

$$cost = \frac{s + w_b b}{(s + w_b b)_{\text{opt}}} + w_m m$$

s - the search time, *b* - build time

THE END

LINKS

- http://www.cs.ubc.ca/~lowe/papers/09muja.pdf
- https://www.coursera.org/learn/ml-clustering-and-retrieval/home/welcome
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.408.306&rep=rep1&type=pdf