Lect 19: K-d tree for fast Nearest Neighbor Search

In our previous lecture, we introduced the K-nearest neighbor decision rule. It has a Bayesian interpretation and an asymptotic error rate no more than twice the Bayesian error, as the number of training examples goes to infinity. This sounds promising. The drawback are

1, it needs to memorize all n-samples (n could be very large)

2, it needs to search the nearest neighbors on-line, complexity is O(n).

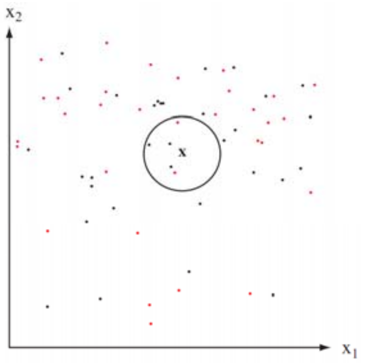
In this lecture, we introduce the Kd-tree technique for fast approximate nearest neighbor search:

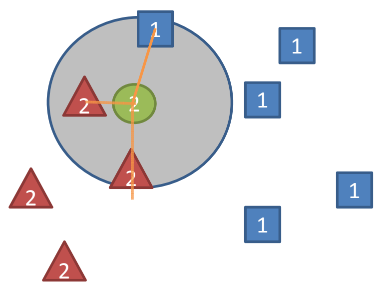
Kd-tree: finding K-nearest neighbors in a d-dimension feature space.

This lecture note is based on a short tutorial (a chapter from a Ph.D. thesis) and a conference paper. Both are provided at the course webpage.

k-Nearest Neighbor Decision Rule

$$\omega(x) = \operatorname{argmax}_i \{k_1, k_2, \ldots k_c\}$$





Nearest Neighbor Search is Time-Consuming

Database:

$X_1 = \{x_{11}, \ldots, x_{1d}\}$

$X_2 = \{x_{21}, \ldots, x_{2d}\}$

....

....

....

$X_n = \{x_{n1}, \ldots, x_{nd}\}$

Query word:

$X_q = \{x_{q1}, \ldots, x_{qd}\}$

Exactly nearest neighbor search is : $O(dn)$

If the dimension d is over 10 to 20, any methods for building search/index structures (including the Kd-trees) do no better than brute-force linear search. So people seek for approximate nearest neighbor. One may argue that approximate search is sufficient because the distance is often not precisely defined anyway.

What is k-d tree

- k-d tree is a multidimensional binary search tree [Bentley '75]
- Structuring the codebook(exemplar-set) more intelligently
- Nearest Neighbor Search : Given an input vector, find the r closest codeword in the codebook and output its index

Input vector

$X_q = \{x_{q1}, \ldots, x_{qk}\}$

Codebook

$X_1 = \{x_{11}, \ldots, x_{1k}\}$

$X_2 = \{x_{21}, \ldots, x_{2k}\}$

....

....

....

$X_n = \{x_{n1}, \ldots, x_{nk}\}$

$\Rightarrow$

r nearest neighbor

X_1'

.

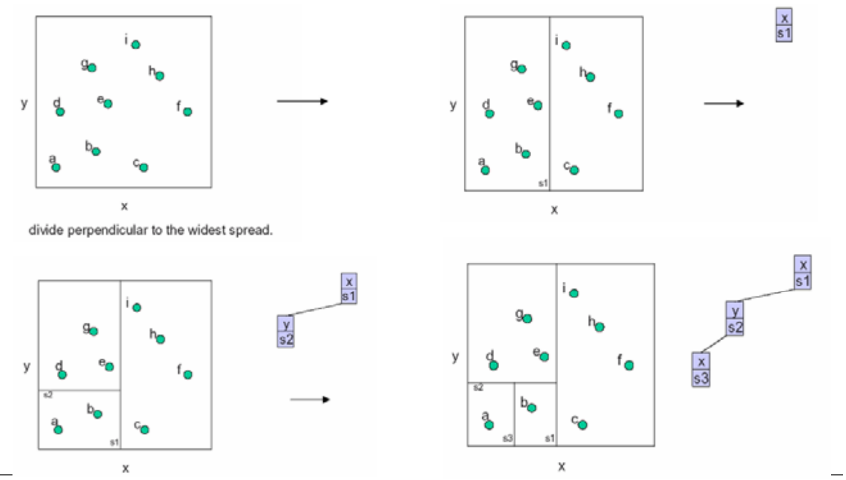
.

.

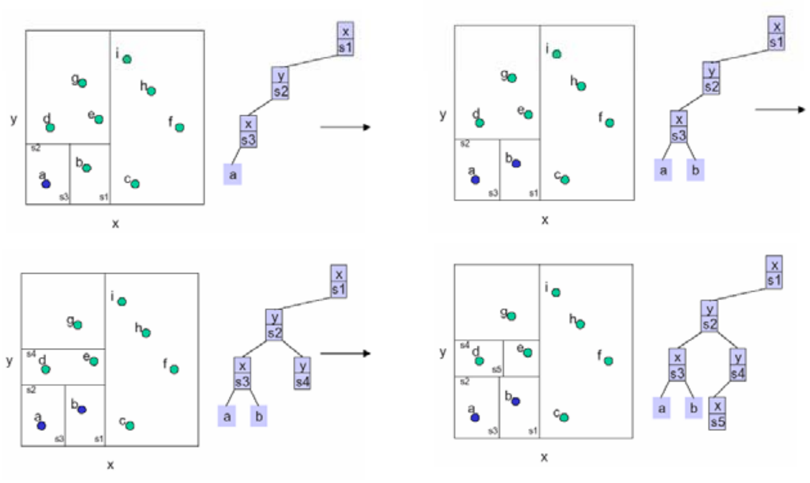
X_k'

K-d tree construction

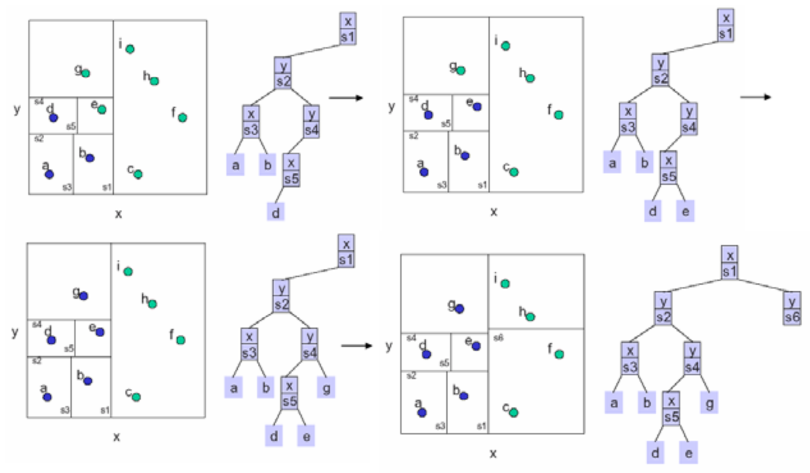
Each time, it chooses a dimension x that has the widest spread (variance), and divide the axis at the mean (or median) value.



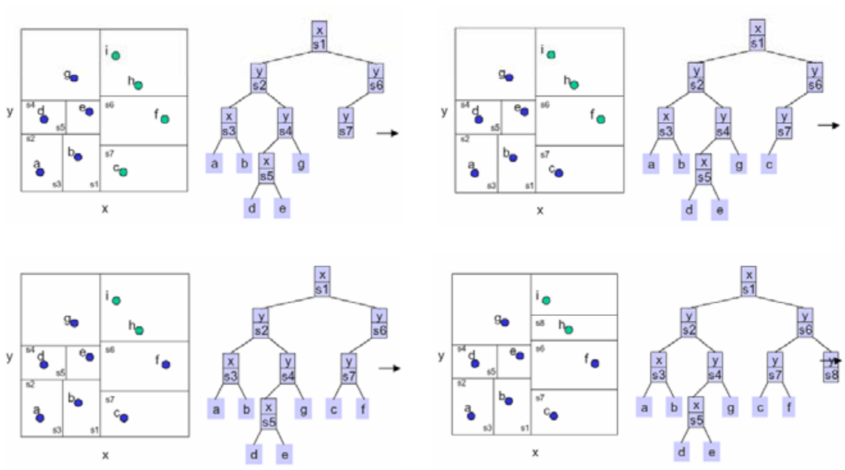
K-d tree construction



K-d tree construction

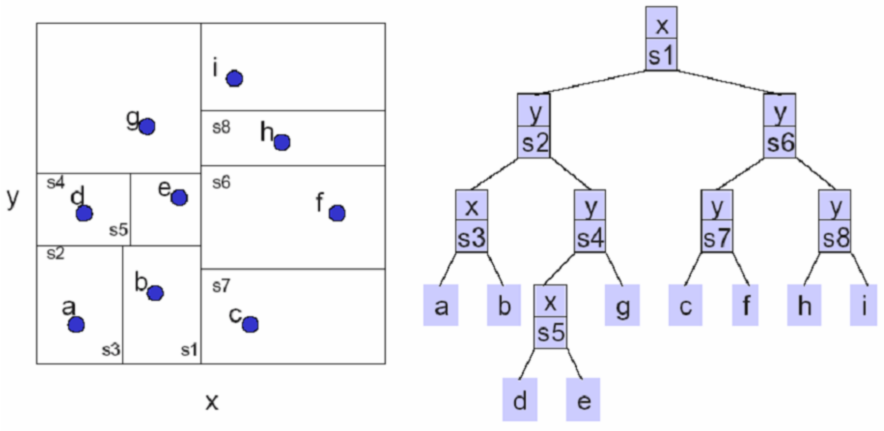


K-d tree construction



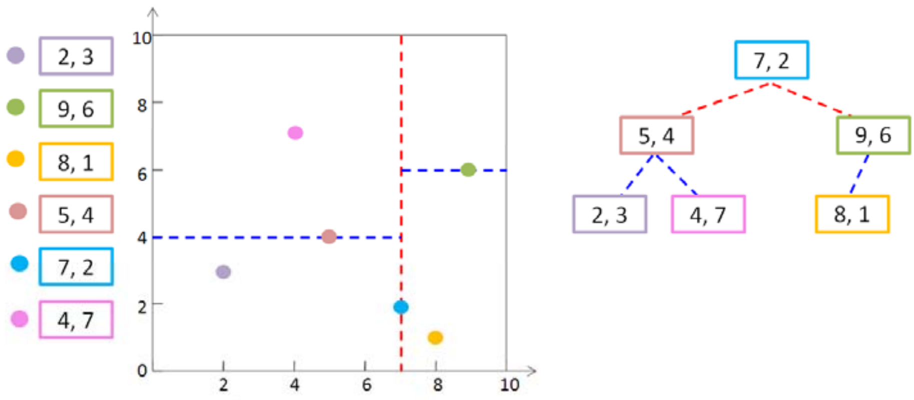
Final K-d tree constructed (d=2)

Bifurcation based on mean values, the samples are on the leaf nodes. The search follows the tree.



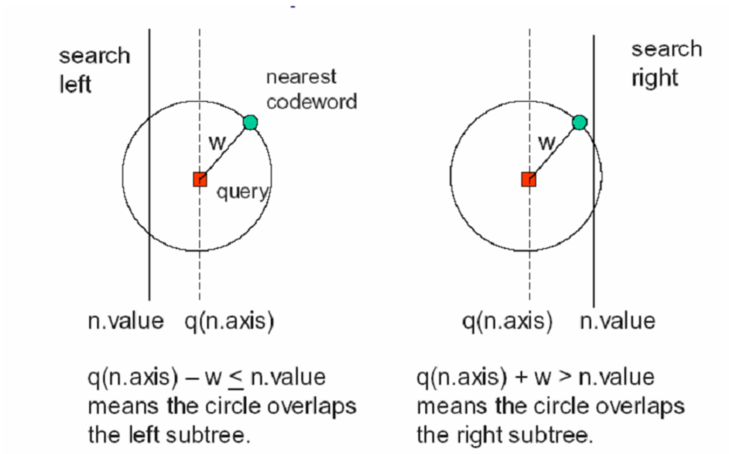
An example of Kd-tree based on median values

The inner nodes are also samples



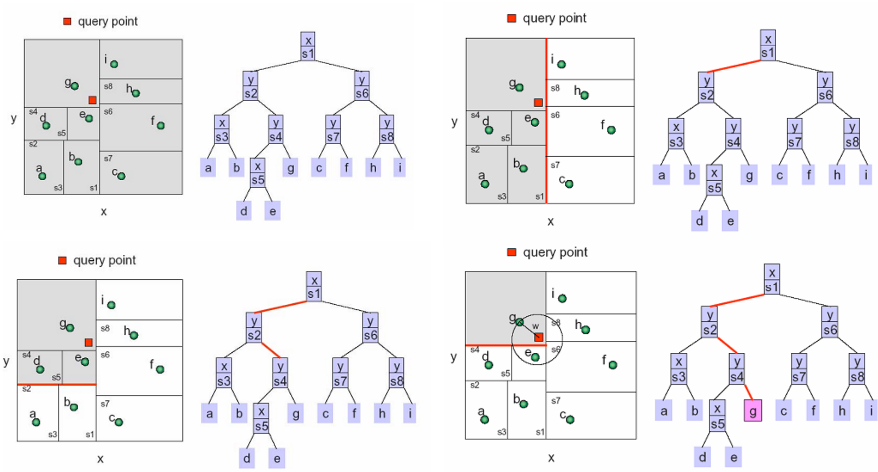
Approximate Nearest Neighbor Search

Using a circle with radius w to tolerate errors. If the query point is within w-distance to the boundary, then it will search for both sides.



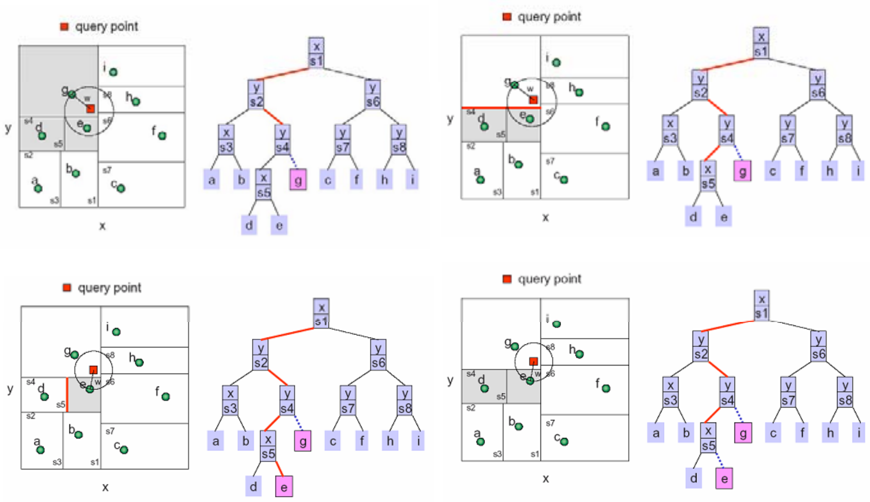
Approximate Nearest Neighbor Search (example)

This is a depth-first-search: it goes all the way down the tree to get a neighbor (leaf-node g) that contains the query point, and then it back-traces to other leaf-nodes.



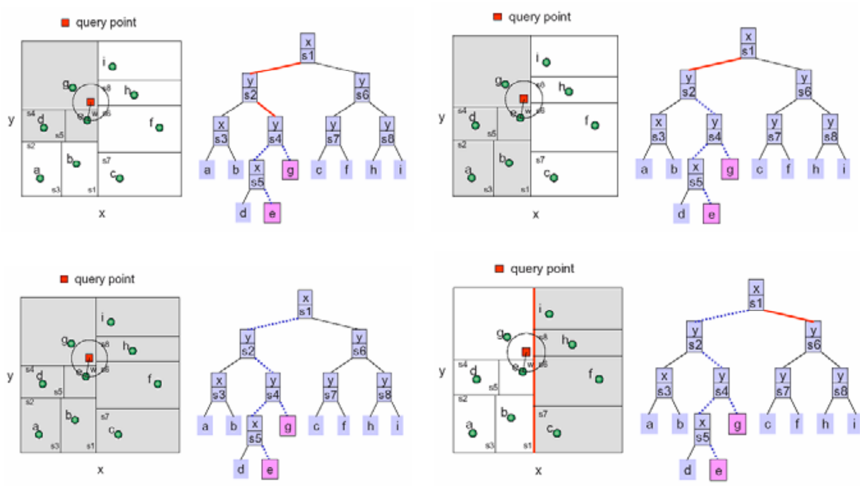
Approximate Nearest Neighbor Search

But the real nearest point is e not g. It searches all branches that overlaps the w-circle.



Approximate Nearest Neighbor Search

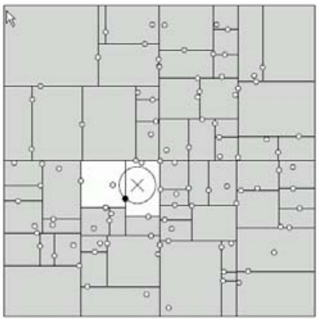
Trace to the left lobe of the tree because the w-circle touch the boundary at the root node.



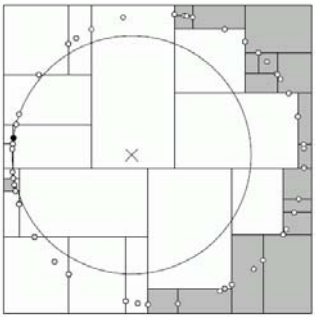
Complexity analysis

The number of searched nodes in the Kd-tree is between O(log n) and O(n).

$$O(\log n) \leq c \leq O(n)$$



A nice case: 2 leaf nodes are searched.

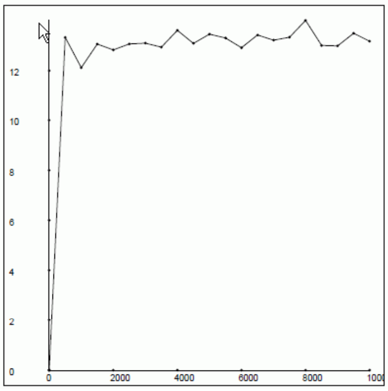


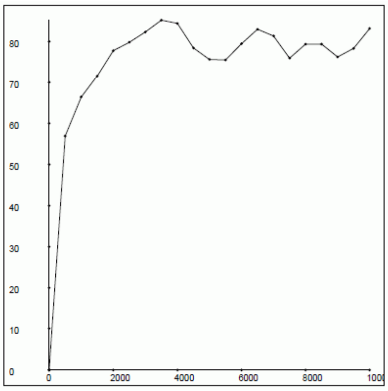
A bad case: majority nodes are searched.
High-dimensional data are just like this.

Examples illustrated by Andrew Moore, 1991 (see handout)

Complexity analysis: empirical results

Plot the number of leaf nodes inspected against the total leaf node number in the tree.

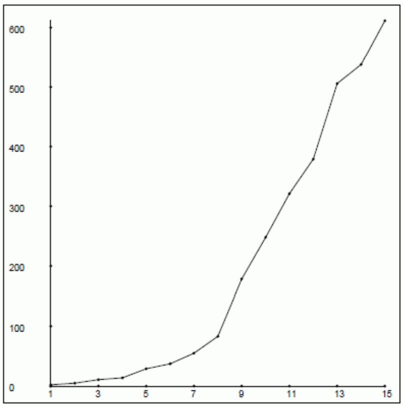




(Left) K=4 (vector dimension), but data distributed in a 3-dim space (intrinsic dimen).
(right) K =8 true dimension is 8.
Each point is averaged over 500 searches in the tree.

Complexity analysis: empirical results

Plot the number of leaf nodes inspected against the dimension K.



When dimension is over 10-20, it will become more and more like the bad cases.

Randomized k-d trees

The Kd-tree search does return all the nearest neighbors within a tolerance radius w.

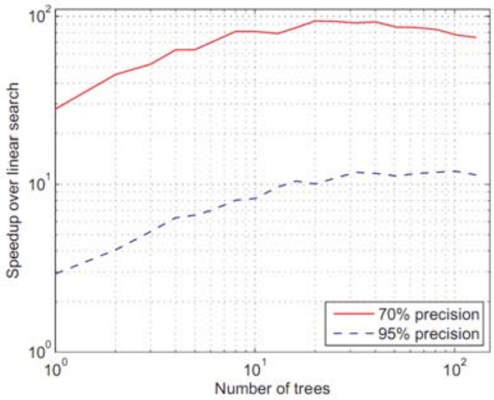
But such distance measure may not be robust. For example, if certain entry in the query vector has noise perturbation, then the search results can be largely affected.

To resolve this problem, people construct many Kd-trees by choosing different dimensions for bifurcations. During query, all these trees are searches.

Other techniques for randomized search: Hashing.

Multiple randomized k-d trees

Multiple trees are built, instead of only one in traditional k-d tree



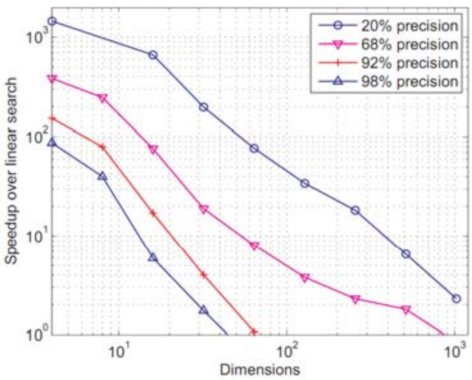
Query point is q,
True nearest neighbor is p*,
you only search it once and return p:

d(q, p) <= (1 + ε) d(q, p*)

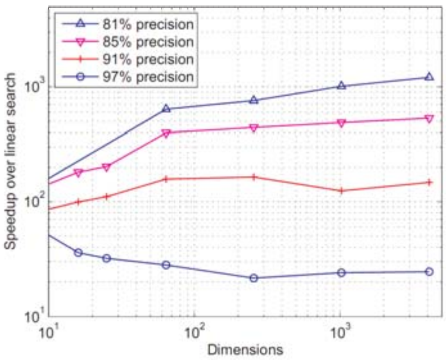
Marius Muja and David G. Lowe, **"Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration"**, in International Conference on Computer Vision Theory and Applications, 2009

k-d tree

- Experiments: Data dimensionality



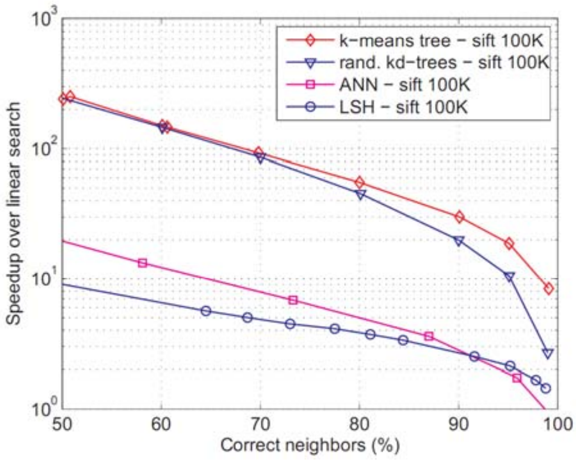
(a) Random vectors



(b) Image patches

k-d tree

- Experiments: Search precision



k-d tree

- Experiments: Search precision