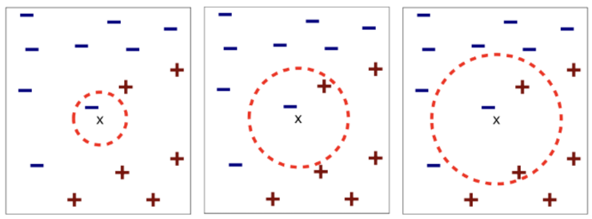
☆ k Nearest-Neighbors Classification

- To classify an example x, select k data points of the training set that are most similar to x.

- Assign the most frequent class among the k selected.

>> Instance-based methods

- Simplest form of learning, training dataset is the model itself
- Traing dataset is searched for the instances that are more similar to the unlabeled instance
- Similarity function defines what's learned
- Lazy learning: nothing happens until new unlabeled instance must be classified
- Methods: "Rote Learning", "Case Base Reasoning", "k-Nearest Neighbors"



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

- How many neighbors?

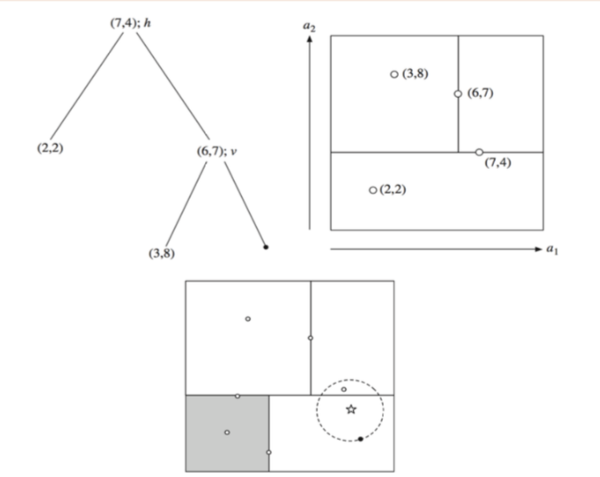
  If k too small → classification may be sensitive to noise

  If k too large → neighborhood may include dissimilar instances

- What similarity measures? → same ones we applied for clustering =)
  → like clustering, we must apply normalization when needed.

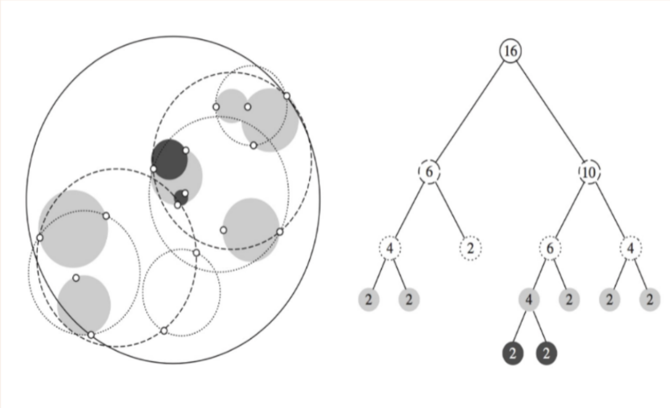
- Basic Approach:
  - Linear scan of the data
  - Classification time for single distance depends on number of data points and number of variables. $O(nd)$ → huge if training set large!
  - Nearest Neighbor seach can be speed up using: KD-Trees, Ball-Trees

- KD-Trees:



- Split the space hierarchically using a three generated by the data
- Add points iteratively to three
- Each point fall in a leaf and splits region around it based on value of one of its attributes
- Search for nearest neighbors: Navigate tree to reach leaf and check
  Backtrack up tree to check nearby regions
  Until all k-nearest neighbors found
- Search complexity: $O(\log n)$ for balanced tree
- To build a good tree;
  - Find a good split point and split direction
  - Possible split direction: greatest variance, possible split point: median value along that direction
  - If data is skewed; use value closest to the mean (rather than median)

- Ball Trees:



- Corners in high-dim space may mean query ball intersects with many regions → limitation for KD-tree
- Can use hyper spheres (balls) instead of hyper rectangles
- Ball tree organizes the data into a tree of k-dimensional hyperspheres
- Balls may allow for a better fit to the data and thus more efficient search

- KNN Regression:

1) Given a data set $(x_1, y_1), \ldots, (x_n, y_n)$

2) Given a query point x_q

3) The value y_q associated to x_q is computed as a local interpolation of the targets associated to neighbor points

- Prediction can use average (or weighted average) of k-nearest targets
- Prediction can use kernel functions that take the distance as input and return a weight

- Discussion:

- KNN is often very accurate but slow since it scans entire training data to derive prediction
- Assumes all attributes are equally important → may need attribute selection or weights
- For noisy instances: Majority vote over k-nearest neighbors
  Remove noisy instances from dataset (difficult =c)