Data Mining

Practical Machine Learning Tools and Techniques

Slides for Sections 4.7 and 6.4
Instance Based Learning
(Skip kD-trees, Ball trees, generalized exemplars and generalized distance functions)

Instance-based learning

- General strategy that can be used for classification or regression.
- Determine "closest" member of training data – distance function needed
- Most instance-based schemes use *Euclidean distance*:

$$\sqrt{(a_1^{(1)}-a_1^{(2)})^2+(a_2^{(1)}-a_2^{(2)})^2+\ldots(a_k^{(1)}-a_k^{(2)})^2}$$

$\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$: two instances with k attributes

- Taking the square root is not required when comparing distances

Normalization and other issues

- Different attributes are measured on different scales ⇒ need to be *normalized*:

$$a_i=\frac{v_i-\min v_i}{\max v_i-\min v_i}$$

v_i: the actual value of attribute i

- Nominal attributes: distance either 0 or 1
- Common policy for missing values: assumed to be maximally distant (given normalized attributes)

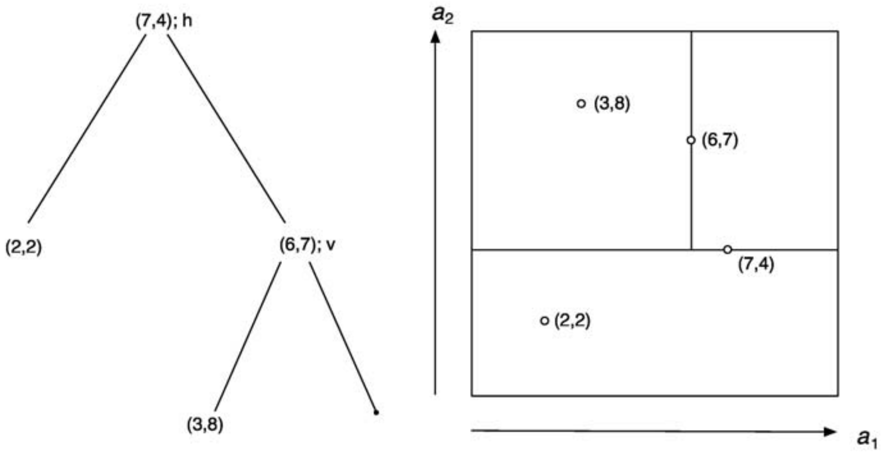
Finding nearest neighbors efficiently

- Simplest way of finding nearest neighbour: linear scan of the data
  - Classification takes time proportional to the product of the number of instances in training and test sets
- Nearest-neighbor search can be done more efficiently using appropriate data structures
- More elaborate methods exist:

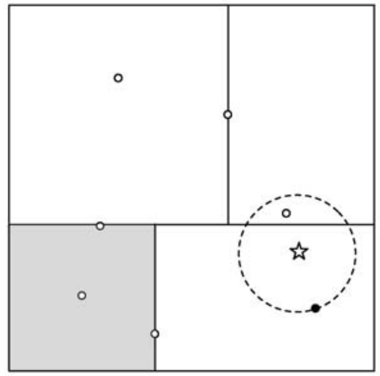
kD-trees and *ball trees*

- *Skip kD-trees and ball trees in book*

*k*D-tree example



Using *k*D-trees: example



Discussion of nearest-neighbor learning

- Often very accurate
- Assumes all attributes are equally important
  - Remedy: attribute selection or weights
- Sensitive to noisy instances:
  - Remedy: Take a majority vote over the k nearest neighbors
- Statisticians have used k-NN since early 1950s
  - If $n \to \infty$ and $k/n \to 0$, error approaches minimum

Difficulties

- Practical problems of 1-NN scheme:
  - Slow for large dimensional data
    - Remedy: remove irrelevant data
  - Noise (but: k -NN copes quite well with noise)
    - Remedy: remove noisy instances
  - All attributes deemed equally important
    - Remedy: weight attributes (or simply select)
  - Doesn’t perform explicit generalization
    - Remedy: rule-based NN approach

Speed up, combat noise

- IB3: Instance-Based Learner Version 3
  - Track the performance of each training example and discard instances that don't perform well
  - Compute confidence intervals for
    - 1. Each instance's success rate
    - 2. Default accuracy of its class
  - Accept/reject instances
    - Accept if lower limit of 1 exceeds upper limit of 2
    - Reject if upper limit of 1 is below lower limit of 2

Weight attributes

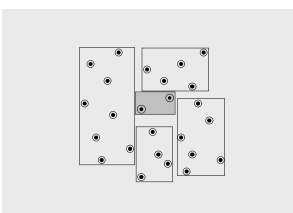
- Some attributes are less important than others – dynamically learn importance and adjust weights.
- IB4: weight each attribute (weights can be class-specific)
- Weighted Euclidean distance:

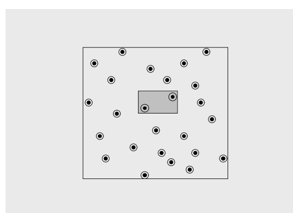
$$\sqrt{(w_1^2(x_1-y_1)^2+\ldots+w_n^2(x_n-y_n)^2)}$$

- Update weights based on nearest neighbor
  - Class correct: increase weight
  - Class incorrect: decrease weight
  - Amount of change for *i* th attribute depends on $|x_i - y_i|$

Rectangular generalizations

Organize instances into hyper-rectangles





- Nearest-neighbor rule is used outside rectangles
- Rectangles are rules! (But they can be more conservative than "normal" rules.)
- Nested rectangles are rules with exceptions

K-Nearest Neighbors

- Determine k-nearest neighbors.
- Given the k neighbors, weigh each closest neighbor according to its distance from query. Take weighted average for regression.
- For classification, choose the label that achieves the maximum weight among the k neighbors.