Nearest Neighbor Learning

Greg Grudic
(Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Nearest Neighbor Algorithm

- Given training data \((x_1, y_1), \ldots, (x_N, y_N)\)
- Define a distance metric between points in inputs space. Common measures are:
  - Euclidean (squared) \(D(x, x_i) = \sum_{j=1}^{d} (x_j - x_{i,j})^2\)
  - Weighted Euclidean \(w_j \geq 0\) \(D(x, x_i) = \sum_{j=1}^{d} w_j (x_j - x_{i,j})^2\)

K-Nearest Neighbor Model

- Given test point \(x\)
- Find the \(K\) nearest training inputs \(x_1, \ldots, x_N\) to \(x\) given the distance metric \(D(x, x_i)\)
- Denote these points as \((x_1, y_1), \ldots, (x_K, y_K)\)

K-Nearest Neighbor Model

- Regression:
  \(\hat{y} = \frac{1}{K} \sum_{k=1}^{K} y_k\)
- Classification:
  \(\hat{y} = \text{most common class in set } \{y_1, \ldots, y_K\}\)
K-Nearest Neighbor Model: Weighted by Distance

• Regression:
  \[ \hat{y} = \frac{\sum_{k=1}^{K} w_k D(x, x_k) y_k}{\sum_{k=1}^{K} w_k D(x, x_k)} \]

• Classification:
  \[ \hat{y} = \text{most common class in weighted set} \]

\[ \left\{ \frac{1}{D(x, x_1)} y_1, \ldots, \frac{1}{D(x, x_K)} y_K \right\} \]

Picking K and \( w_1, \ldots, w_d \)

• Use N fold cross validation
  – Pick values that minimize cross validation error

Class Decision Boundaries: The Voronoi Diagram

Each line segment is equidistance between points in opposite classes. The more points, the more complex the boundaries.

K-Nearest Neighbor Algorithm Characteristics

• Universal Approximator
  – Can model any many to one mapping arbitrarily well

• Curse of Dimensionality: Can be easily fooled in high dimensional spaces
  – Dimensionality reduction techniques are often used

• Model can be slow to evaluate for large training sets
  – kd-trees can help
  – Selectively storing data points also helps
kd-trees

More Recent Optimized NN Searches

• Cover Trees
  – http://hunch.net/~jl/projects/cover_tree/cover_tree.html
• Fast for large d…