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 $\mathbf{x}$
• Find the $K$ nearest training inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$ to $\mathbf{x}$ given the distance metric $D(\mathbf{x}, \mathbf{x}_i)$

• Denote these points as $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_K, y_K)$

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

- Classification:
  \[
  \hat{y} = \text{most common class in weighted set}
  \}
  \{D(x, x_1) y_1, ..., D(x, x_K) y_K \}
  \]

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

- Use N fold cross validation
  - Pick values that minimize the 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