k-Nearest Neighbor Algorithm

SCMA292 Mathematical Modeling : Machine Learning

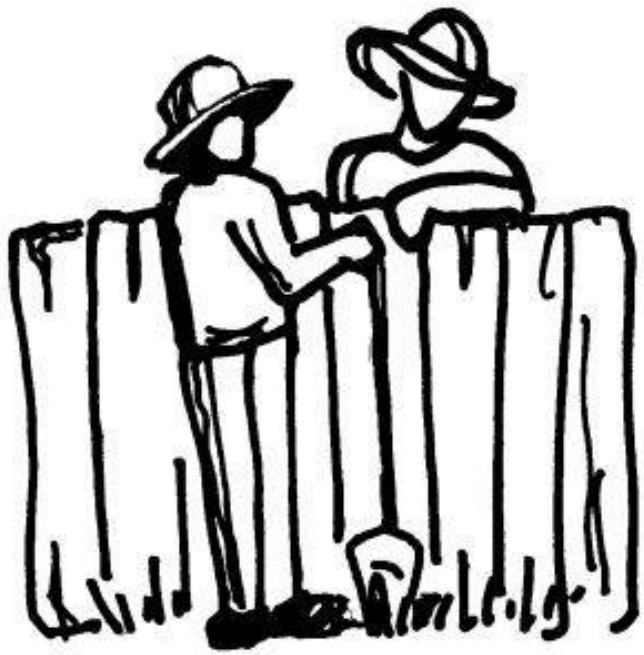
Krikamol Muandet

Department of Mathematics

Faculty of Science, Mahidol University

April 27, 2016

Your Neighbors Know It Best



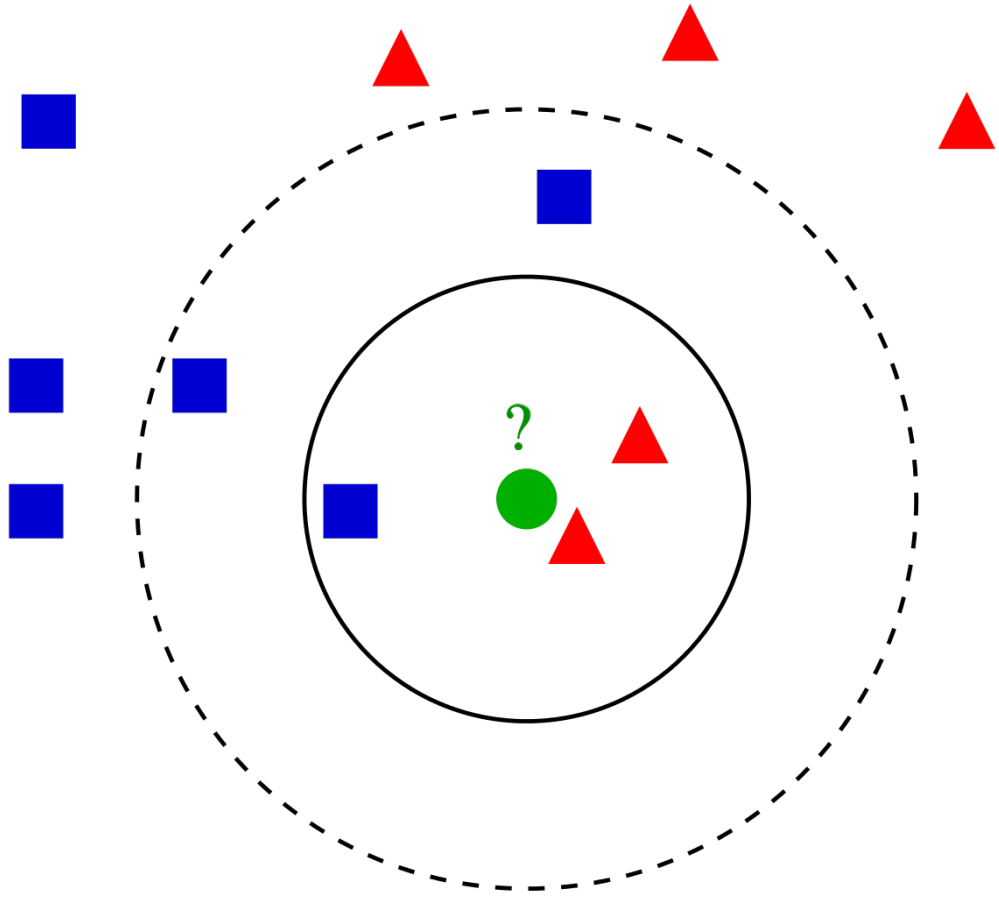
k-Nearest Neighbor Algorithm

- Lazy and memory-based algorithm : store all training examples
- Prediction : classify a new example x by finding the training example (x_i, y_i) that are *closest* to x
- Predict the class $y = y_i$
- The parameter k specifies the number of neighbors. If $k > 1$, we may use

$$y = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

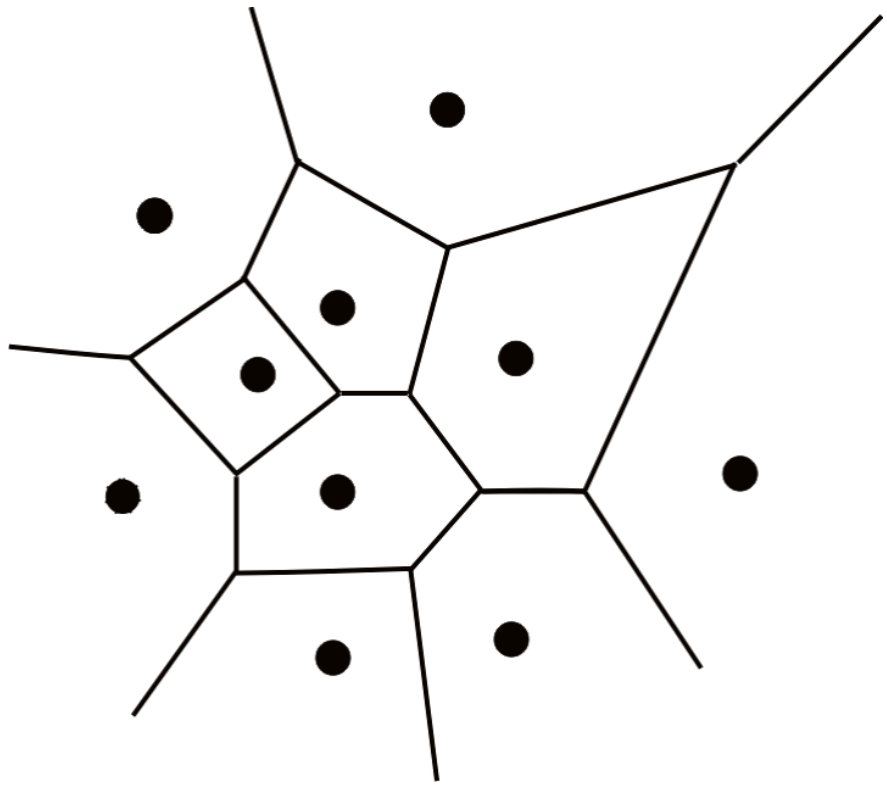
where $N_k(x)$ is the neighborhood of x, i.e., k closest points x_i in the training sample.

k-Nearest Neighbor Algorithm



Decision Boundaries: The Voronoi Diagram

A decision surface of 1-NN classifier



Given a set of points, a *Voronoi diagram* describes the areas that are nearest to any given point.

Linear Model

Examples : k=1

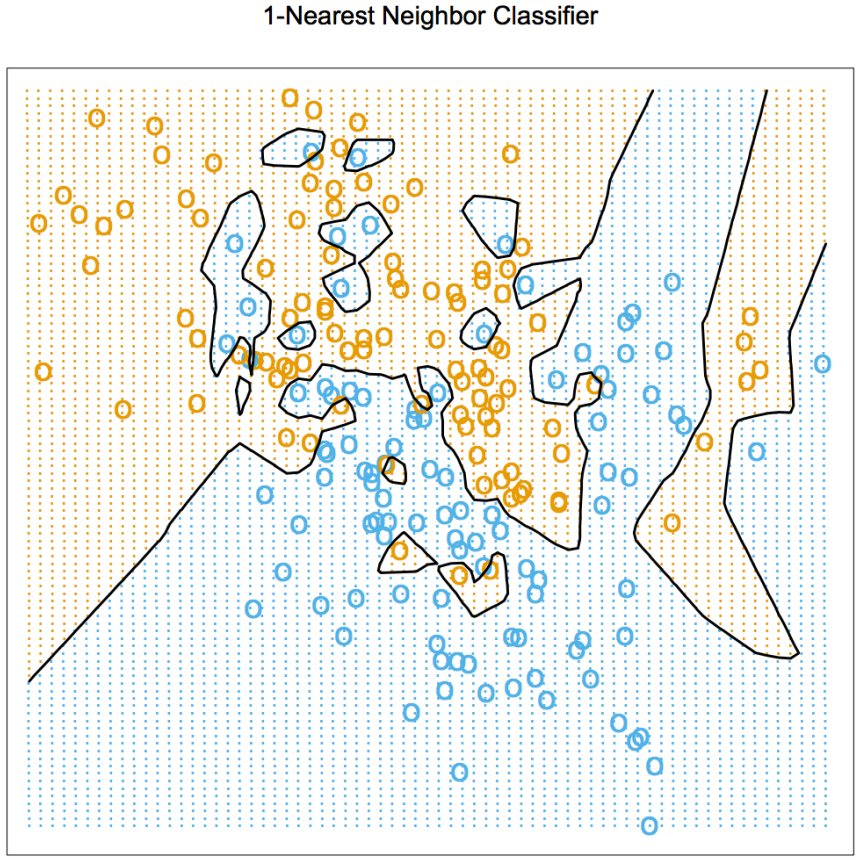


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

Examples : k=15

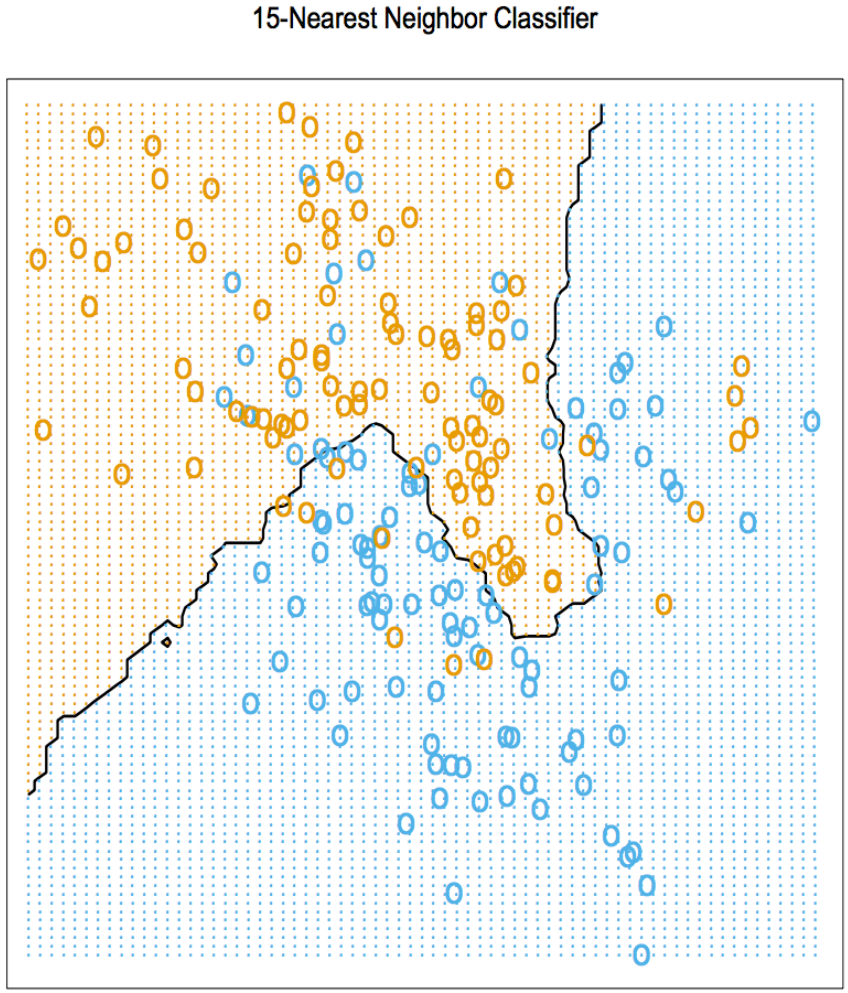


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

Examples : Bayes Classifier

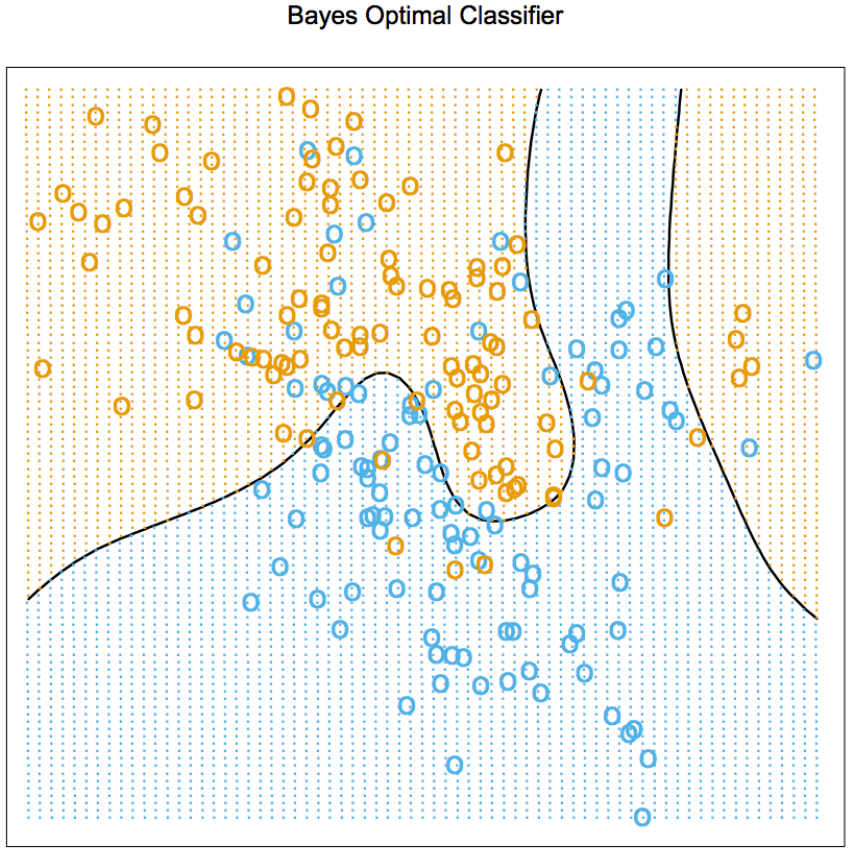


FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

Distances

- Euclidean distance $d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$
- Minkowski distance $d(x_i, x_j) = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^q\right)^{1/q}$
- Mahalanobis distance $d(x_i, x_j) = \sqrt{(x_i - x_j)^\top \Sigma^{-1} (x_i - x_j)}$
- Manhattan distance $d(x_i, x_j) = \sum_{k=1}^{d} |x_{ik} - x_{jk}|$

Euclidean Distance

- For vectors x_i, x_j in $\mathbb{R}^d$

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

- Most common distance metric
- Euclidean distance treats each feature as equally important
- In practice, some features may be much more discriminative than other features
- Mahalanobis distance : $d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} \frac{(x_{ik} - x_{jk})^2}{s_k^2}}$

Feature Normalization/Standardization

- Features can be on different scales : difficult to compare
  - One feature may take values between 1 and 10
  - Others may take values between 1000 and 10,000
- Trick: normalize all features to be on the same scale, e.g., $[0, 1]$
- Linear scaling :

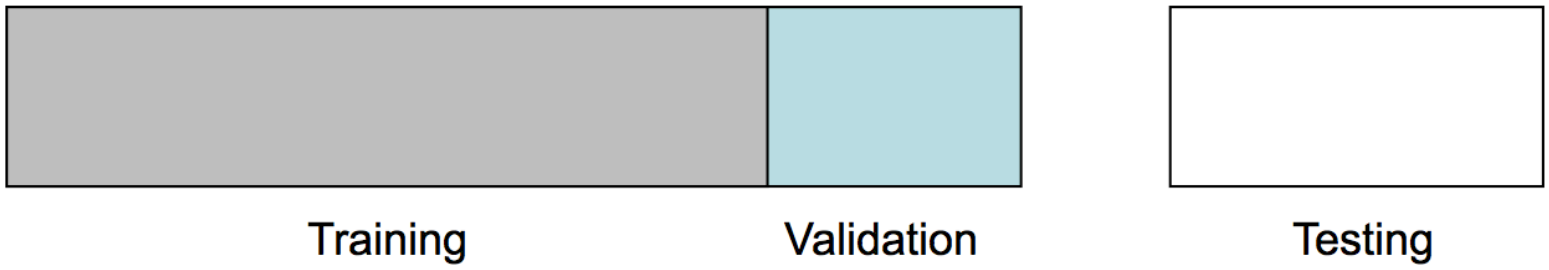
$$f_{\text{new}} = \frac{f_{\text{old}} - f_{\text{old}}^{\min}}{f_{\text{old}}^{\max} - f_{\text{old}}^{\min}}$$

- Standardization : zero mean and unit variance

$$f_{\text{new}} = \frac{f_{\text{old}} - \mu}{\sigma}$$

How to Choose k

- The number of neighbors, k, is the only parameter of k-NN
- Like many other models, we can perform *model selection* to choose the best value of k
  - If k is too small, the model can overfit (sensitive to noise).
  - If k is too large, the model can underfit.
- K-fold cross validation procedure : choose k that minimizes *validation error*:



How to Choose k

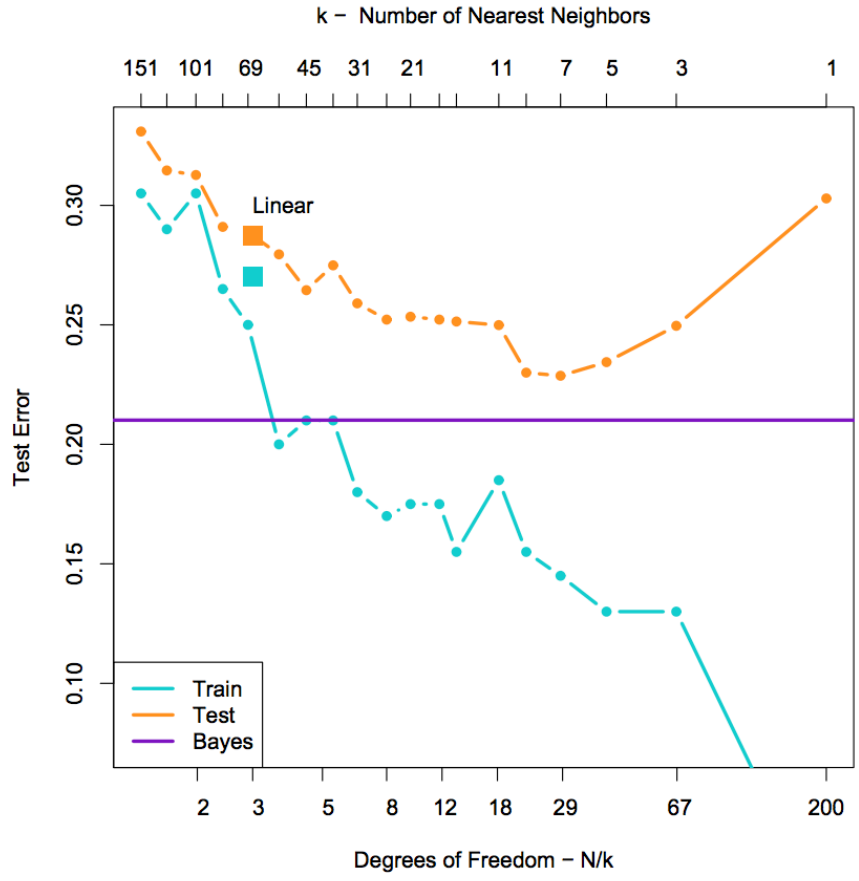


FIGURE 2.4. *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* $10,000$. *The orange curves are test and the blue are training error for k-nearest-neighbor classification. The results for linear regression are the bigger orange and blue squares at three degrees of freedom. The purple line is the optimal Bayes error rate.*

Curse of Dimensionality

- In high-dimensional space, the "neighborhood" becomes very large.
- Since points are very far away from each other, the notion of "nearest neighbor" does not really make sense anymore.
- Example : we want to find 5-NN for 5,000 points uniformly distributed in the unit hypercube
  - 1D : we must go a distance of $5/5000 = 0.001$ on average to capture the 5 nearest neighbors
  - 2D : we must go $\sqrt{0.001}$ to get a square that contain 0.001 of the volume
  - D : we must go $(0.001)^{1/D}$

Advantages and Disadvantages

Advantages

- Easy to build
- No training required
- Good empirical performance

Disadvantages

- Curse of dimensionality
- Computationally expensive (slow at test time)
- Susceptible to irrelevant features

Summary : k-Nearest Neighbor Algorithm

Given a test sample x:

1. Pick the number of neighbors k
2. Choose the distance metric $d(\cdot, \cdot)$
3. Find the k nearest neighbors $N_k(x)$ of x in the data set $\mathcal{D}$ using distance metric d
4. Output y from the aggregation of outputs y_i from $N_k(x)$