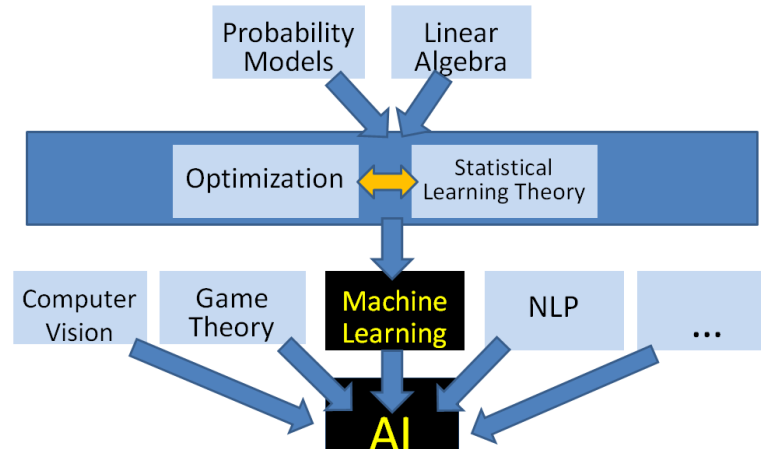


MSDS 422: Practical Machine Learning

Other Classification Models



NORTHWESTERN
UNIVERSITY

Dr. Nathan Bastian

References

- *An Introduction to Statistical Learning, with Applications in R* (2013), by G. James, D. Witten, T. Hastie, and R. Tibshirani.
- *The Elements of Statistical Learning* (2009), by T. Hastie, R. Tibshirani, and J. Friedman.

L₁ Regularized Logistic Regression

- For logistic regression, we would maximize a penalized version of the log-likelihood function:

$$\max_{\beta_0, \beta} \left\{ \sum_{i=1}^N \left[y_i(\beta_0 + \beta^T x_i) - \log(1 + e^{\beta_0 + \beta^T x_i}) \right] - \lambda \sum_{j=1}^p |\beta_j| \right\}$$

- As with the lasso method, we typically do not penalize the intercept term, and we standardize the predictors for the penalty to be meaningful.
- Because this optimization criterion is concave, a solution can be found using nonlinear programming methods or via quadratic approximations as used in the Newton-Raphson algorithm.

Multi-Class Logistic Regression

- We sometimes wish to classify a response variable that has more than two classes.
- The two-class logistic regression models have multiple class extensions.
- This method is often not used in practice because *discriminant analysis* is a more popular method for multi-class classification.

Multi-Class Logistic Regression (cont.)

- When the number of classes $K > 2$, the parameter β is a $(K - 1)(p + 1)$ vector:

- Also, let $\bar{\beta}_l = \begin{pmatrix} \beta_{10} \\ \beta_l \end{pmatrix}$

$$\beta = \begin{pmatrix} \beta_{10} \\ \beta_1 \\ \beta_{20} \\ \beta_2 \\ \vdots \\ \beta_{(K-1)0} \\ \beta_{K-1} \end{pmatrix} = \begin{pmatrix} \beta_{10} \\ \beta_{11} \\ \vdots \\ \beta_{1p} \\ \beta_{20} \\ \vdots \\ \beta_{2p} \\ \vdots \\ \beta_{(K-1)0} \\ \vdots \\ \beta_{(K-1)p} \end{pmatrix}$$

Multi-Class Logistic Regression (cont.)

- The log-likelihood function becomes:

$$\begin{aligned} l(\beta) &= \sum_{i=1}^N \log p_{g_i}(x_i; \beta) \\ &= \sum_{i=1}^N \log \left(\frac{e^{\bar{\beta}_{g_i}^T x_i}}{1 + \sum_{l=1}^{K-1} e^{\bar{\beta}_{g_i}^T x_i}} \right) \\ &= \sum_{i=1}^N \left[\bar{\beta}_{g_i}^T x_i - \log \left(1 + \sum_{l=1}^{K-1} e^{\bar{\beta}_{g_i}^T x_i} \right) \right] \end{aligned}$$

Multi-Class Logistic Regression (cont.)

- The indicator function $I(.)$ equals 1 when the argument is *true* and 0 *otherwise*.
- Thus, the first order derivatives of the log-likelihood function are:

$$\begin{aligned}\frac{\partial l(\beta)}{\partial \beta_{kj}} &= \sum_{i=1}^N \left[I(g_i = k) x_{ij} - \frac{e^{\bar{\beta}_k^T x_i} x_{ij}}{1 + \sum_{l=1}^{K-1} e^{\bar{\beta}_l^T x_i}} \right] \\ &= \sum_{i=1}^N x_{ij} (I(g_i = k) - p_k(x_i; \beta))\end{aligned}$$

Multi-Class Logistic Regression (cont.)

- The second order derivatives of the log-likelihood function are:

$$\begin{aligned}\frac{\partial^2 l(\beta)}{\partial \beta_{kj} \partial \beta_{mn}} &= \sum_{i=1}^N x_{ij} \cdot \frac{1}{(1 + \sum_{l=1}^{K-1} e^{\bar{\beta}_l^T x_i})^2} \cdot \left[-e^{\bar{\beta}_k^T x_i} I(k=m) x_{in} (1 + \sum_{l=1}^{K-1} e^{\bar{\beta}_l^T x_i}) + e^{\bar{\beta}_k^T x_i} x_{in} \right] \\ &= \sum_{i=1}^N x_{ij} x_{in} (-p_k(x_i; \beta) I(k=m) + p_k(x_i; \beta) p_m(x_i; \beta)) \\ &= - \sum_{i=1}^N x_{ij} x_{in} p_k(x_i; \beta) [I(k=m) - p_m(x_i; \beta)]\end{aligned}$$

- We next express the solution in matrix form because it is much more compact. First, we introduce several notations.

Multi-Class Logistic Regression (cont.)

- \mathbf{y} is the concatenated indicator vector, a column vector of dimension $N(K - 1)$.

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_{K-1} \end{pmatrix} \quad \mathbf{y}_k = \begin{pmatrix} I(g_1 = k) \\ I(g_2 = k) \\ \vdots \\ I(g_N = k) \end{pmatrix} \quad 1 \leq k \leq K - 1$$

- \mathbf{p} is the concatenated vector of fitted probabilities, a column vector of dimension $N(K - 1)$.

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_{K-1} \end{pmatrix} \quad \mathbf{p}_k = \begin{pmatrix} p_k(x_1; \beta) \\ p_k(x_2; \beta) \\ \vdots \\ p_k(x_N; \beta) \end{pmatrix} \quad 1 \leq k \leq K - 1$$

Multi-Class Logistic Regression (cont.)

- $\tilde{\mathbf{X}}$ is an $N(K - 1) \times (p + 1)(K - 1)$ matrix, where \mathbf{X} is the $N(p + 1)$ input matrix defined before:

$$\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X} & 0 & \dots & 0 \\ 0 & \mathbf{X} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \mathbf{X} \end{pmatrix}$$

- Matrix \mathbf{W} is an $N(K - 1) \times N(K - 1)$ square matrix:

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \dots & \mathbf{W}_{1(K-1)} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \dots & \mathbf{W}_{2(K-1)} \\ \dots & \dots & \dots & \dots \\ \mathbf{W}_{(K-1),1} & \mathbf{W}_{(K-1),2} & \dots & \mathbf{W}_{(K-1),(K-1)} \end{pmatrix}$$

Multi-Class Logistic Regression (cont.)

- Each submatrix \mathbf{W}_{km} , $1 \leq k, m \leq K - 1$, is an $N \times N$ diagonal matrix.
- When $k = m$, the i th diagonal element in \mathbf{W}_{kk} is $p_k(x_i; \beta^{old})(1 - p_k(x_i; \beta^{old}))$.
- – When $k \neq m$, the i th diagonal element in \mathbf{W}_{km} is $-p_k(x_i; \beta^{old})(1 - p_m(x_i; \beta^{old}))$.

- As with binary classification:

$$\frac{\partial l(\beta)}{\partial \beta} = \tilde{\mathbf{X}}^T (\mathbf{y} - \mathbf{p})$$
$$\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} = \tilde{\mathbf{X}}^T \mathbf{W} \tilde{\mathbf{X}}$$

Multi-Class Logistic Regression (cont.)

- The formula for updating β^{new} in the binary classification case holds for multi-class, except that the definitions of the matrices are more complicated.
- The formula itself may look simple because the complexity is wrapped in the definitions of the matrices.

$$\beta^{new} = (\tilde{\mathbf{X}}^T \mathbf{W} \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{W} \mathbf{z}$$

where $\mathbf{z} \triangleq \tilde{\mathbf{X}} \beta^{old} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})$ or simply:

$$\beta^{new} = \beta^{old} + (\tilde{\mathbf{X}}^T \mathbf{W} \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T (\mathbf{y} - \mathbf{p})$$

Discriminant Analysis

- Let the feature vector be X and the class labels be Y .
- The Bayes rule says that if you have the joint distribution of X and Y , and if X is given, under 0-1 loss, the optimal decision on Y is to choose a class with maximum posterior probability given X .
- *Discriminant analysis* belongs to the branch of classification methods called generative modeling, where we try to estimate the within class density of X given the class label.
- Combined with the prior probability (unconditioned probability) of classes, the posterior probability of Y can be obtained by the Bayes formula.

Discriminant Analysis (cont.)

- Assume the *prior* probability or the marginal probability mass function for class k is denoted as π_k , $\sum_{k=1}^K \pi_k = 1$
- π_k is the probability that a given observation is associated with the k th category of the response variable Y .
- Note that π_k is usually estimated simply by empirical frequencies of the training set:

$$\hat{\pi}_k = \frac{\text{\# of Samples in class } k}{\text{Total \# of samples}}$$

Discriminant Analysis (cont.)

- You have the training data set and you count what percentage of data come from a certain class.
- Then we need the class-conditional density of X .
- This is the *density function* of X conditioned on the class k , or class $G = k$ denoted by $f_k(x)$.
- Note that $f_k(x)$ is relatively large if there is a high probability that an observation in the k th class has $X \approx x$, and $f_k(x)$ is relatively small if it is very unlikely that an observation in the k th class has $X \approx x$.

Discriminant Analysis (cont.)

- According to the Bayes rule, we can now compute the *posterior probability* that an observation $X = x$ belongs to the k th class:

$$Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^K f_l(x)\pi_l}$$

- This is a *conditional* probability of class G (or denoted Y in the book) given X (the predictor value for that observation).

- By the Bayes rule for 0-1 loss:
$$\begin{aligned}\hat{G}(x) &= \arg \max_k Pr(G = k|X = x) \\ &= \arg \max_k f_k(x)\pi_k\end{aligned}$$

Discriminant Analysis (cont.)

- Notice that the denominator is identical no matter what class k you are using. Thus, for maximization, it does not make a difference in the choice of k .
- The *maximum a posteriori* rule (i.e. Bayes rule for 0-1 loss) is essentially trying to maximize π_k times $f_k(x)$.
- Note that the Bayes classifier, which classifies an observation to the class for which the posterior probability is largest, has the lowest possible error rate out of all classifiers.

Class Density Estimation

- Depending on which algorithms you use, you end up with different ways of *density estimation* within every class.
- In both **Linear Discriminant Analysis (LDA)** and **Quadratic Discriminant Analysis (QDA)**, we assume that every density within each class is a Gaussian distribution.
- In LDA we assume those Gaussian distributions for different classes share the same covariance structure. In QDA, however, we do not have such a constraint.

Class Density Estimation (cont.)

- You can also use **general nonparametric density estimates**, for instance kernel estimates and histograms.
- **Naive Bayes**: assume each of the class densities are products of marginal densities, that is, all the variables are independent.
- In the well-known Naive Bayes algorithm, you can separately estimate the density (or probability mass function for discrete values of X) for every dimension and then multiply them to get the joint density (or joint probability mass function).

Linear Discriminant Analysis

- LDA undertakes the same task as logistic regression in that it classifies data based on categorical variables.
- When the classes are well separated, the parameter estimates for the logistic regression model are surprisingly unstable, where LDA does not suffer from this problem.
- Under LDA, we assume that the density for X given every class k is following a Gaussian distribution.

Linear Discriminant Analysis (cont.)

- The following is the density formula for a multivariate Gaussian distribution:

$$f_k(x) = \frac{1}{(2\pi)^{p/2} |\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)}$$

where p is the dimension and Σ_k is the covariance matrix.

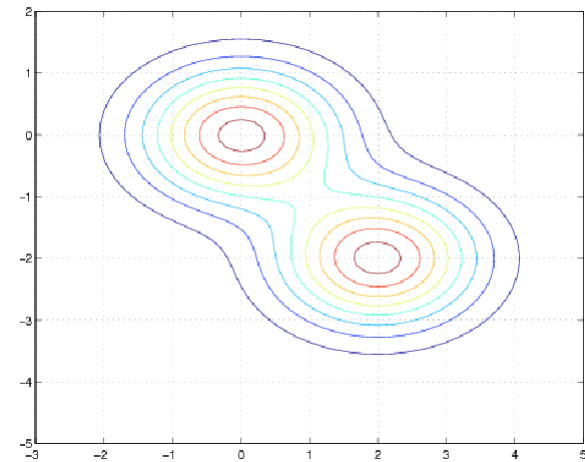
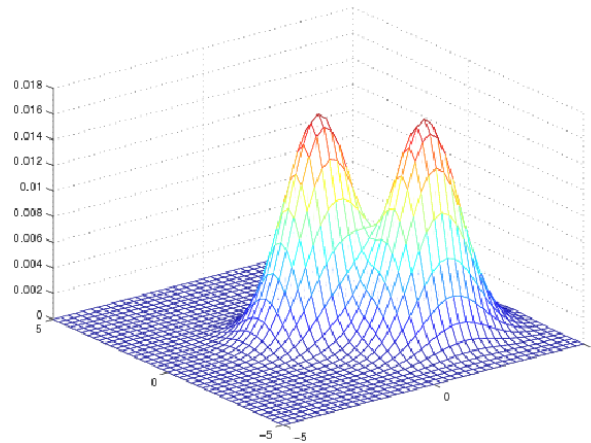
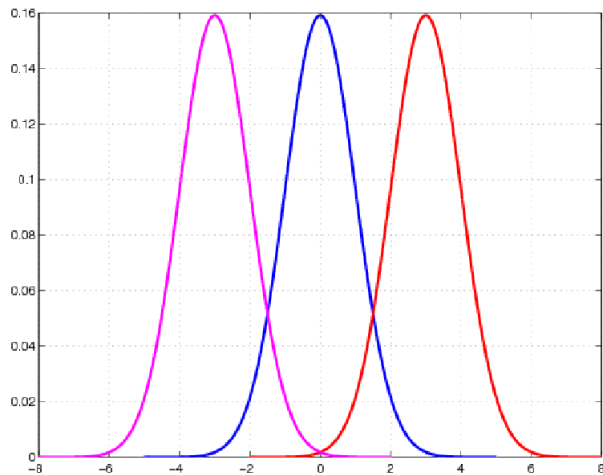
- This involves the square root of the determinant of this matrix. In this case, we are doing matrix multiplication.
- The vector x and the mean vector μ_k are both column vectors.

Linear Discriminant Analysis (cont.)

- For LDA, the covariance matrix $\Sigma_k = \Sigma \forall k$
- In LDA, you simply assume for different k that the covariance matrix is identical.
- By making this assumption, the classifier becomes linear. The only difference from QDA is that we do not assume that the covariance matrix is identical for different classes.
- For QDA, the decision boundary is determined by a quadratic function.

Linear Discriminant Analysis (cont.)

- Since the covariance matrix determines the shape of the Gaussian density, in LDA, the Gaussian densities for different classes have the same shape, but are shifted versions of each other (different mean vectors).



Linear Discriminant Analysis (cont.)

- For the moment, we will assume that we already have the covariance matrix for every class.
- In terms of optimal classification, Bayes rule says that we should pick a class that has the maximum posterior probability given the feature vector X .
- If we are using the generative modeling approach, this is equivalent to maximizing the product of the prior and the within class density.

Linear Discriminant Analysis (cont.)

- Since the log function is an increasing function, the maximization is equivalent because whatever gives you the maximum should also give you a maximum under a log function.
- Thus, we plug in the density of the Gaussian distribution assuming common covariance and then multiplying the prior probabilities:

$$\begin{aligned}\hat{G}(x) &= \arg \max_k Pr(G = k | X = x) \\ &= \arg \max_k f_k(x) \pi_k \\ &= \arg \max_k \log(f_k(x) \pi_k) \\ &= \arg \max_k \left[-\log((2\pi)^{p/2} |\Sigma|^{1/2}) - \frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log(\pi_k) \right] \\ &= \arg \max_k \left[-\frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log(\pi_k) \right]\end{aligned}$$

Linear Discriminant Analysis (cont.)

- Note that: $-\frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k - \frac{1}{2} x^T \Sigma^{-1} x$
- After simplification, we obtain the Bayes classifier:

$$\hat{G}(x) = \arg \max_k \left[x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k) \right]$$

- Given any x , you simply plug into this formula and see which k maximizes this.
- Usually the number of classes is pretty small, and very often only two classes, so an exhaustive search over the classes is effective.

Linear Discriminant Analysis (cont.)

- LDA provides a linear boundary because the quadratic term is dropped.

- Thus, we define the linear discriminant function as:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k)$$

$$\hat{G}(x) = \arg \max_k \delta_k(x)$$

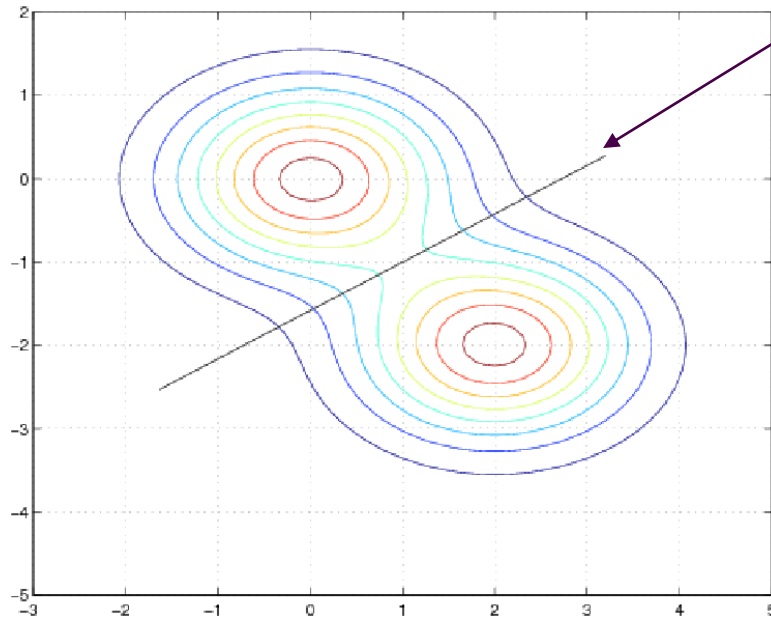
- The decision boundary between class k and l is $\{x : \delta_k(x) = \delta_l(x)\}$ or equivalently $\log \frac{\pi_k}{\pi_l} - \frac{1}{2} (\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) + x^T \Sigma^{-1} (\mu_k - \mu_l) = 0$

Linear Discriminant Analysis (cont.)

- In binary classification, for instance if we let $(k = 1, l = 2)$, then we would define constant a_0 , where π_1 and π_2 are prior probabilities for the two classes and μ_1 and μ_2 are mean vectors.
- Define: $a_0 = \log \frac{\pi_1}{\pi_2} - \frac{1}{2} (\mu_1 + \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$
- Define: $(a_1, a_2, \dots, a_p)^T = \Sigma^{-1} (\mu_1 - \mu_2)$
- Classify to class 1 if $a_0 + \sum_{j=1}^p a_j x_j > 0$ or to class 2 otherwise.

Linear Discriminant Analysis (cont.)

- For example:



Decision Boundary

$$*\pi_1 = \pi_2 = 0.5$$

$$*\mu_1 = (0, 0)^T, \mu_2 = (2, -2)^T$$

$$*\Sigma = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 0.5625 \end{pmatrix}$$

$$*\text{Decision boundary: } 5.56 - 2.00x_1 + 3.56x_2 = 0.0$$

- We have two classes and we know the within class density. The marginal density is simply the weighted sum of the within class densities, where the weights are the prior probabilities.

Linear Discriminant Analysis (cont.)

- Because we have equal weights and because the covariance matrix two classes are identical, we get these symmetric lines in the contour plot.
- The black diagonal line is the decision boundary for the two classes.
- If you are given an x , if it is above the line we would be classifying this x into the first-class. If it is below the line, it would be classified into the second class.

Linear Discriminant Analysis (cont.)

- In this example, we assumed that we have the prior probabilities for the classes and we also had the within class densities given to us.
- In practice, however, we do not have this. All we have is a set of training data.
- Thus, how do we find the π_k 's and the $f_k(x)$?
- We need to estimate the Gaussian distribution!!

Linear Discriminant Analysis (cont.)

- Here is the formula for estimating the π_k 's and the parameters in the Gaussian distributions.
- The formula below is actually the maximum likelihood estimator:

$$\hat{\pi}_k = N_k / N$$

where N_k is the number of class- k samples and N is the total number of points in the training data.

- To get the prior probabilities for class k , we simply count the frequency of data points in class k .

Linear Discriminant Analysis (cont.)

- The mean vector for every class is also simple. We take all of the data points in a given class and compute the sample mean:

$$\hat{\mu}_k = \sum_{g_i=k} x^{(i)} / N_k$$

- The covariance matrix formula looks slightly complicated. The reason is because we have to get a common covariance matrix for all of the classes.
- First, we divide the data points in two given classes according to the given labels.

Linear Discriminant Analysis (cont.)

- If we were looking at class k , for every point we subtract the corresponding mean which we computed earlier. Then multiply its transpose.
- Remember x is a column vector, therefore if we have a column vector multiplied by a row vector, we get a square matrix, which is what we need.

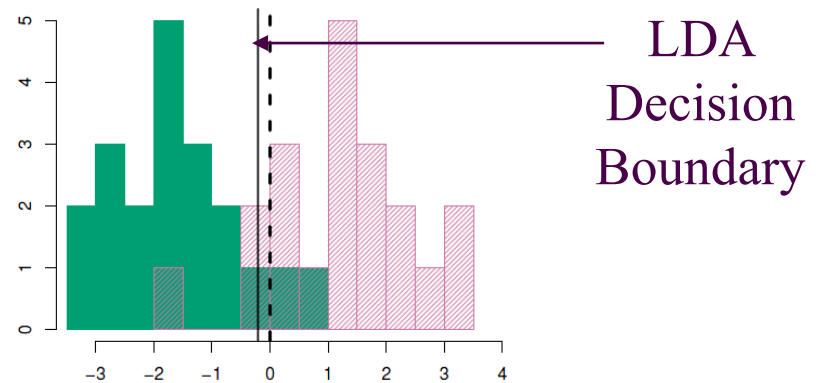
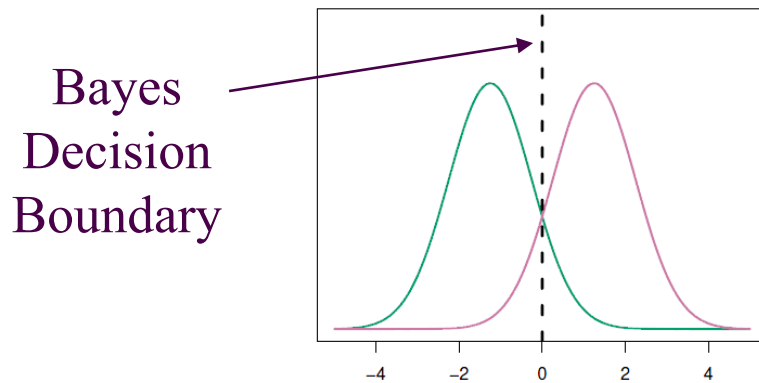
$$\hat{\Sigma} = \sum_{k=1}^K \sum_{g_i=k} \left(x^{(i)} - \hat{\mu}_k \right) \left(x^{(i)} - \hat{\mu}_k \right)^T / (N - K)$$

Linear Discriminant Analysis (cont.)

- First, we do the summation within every class k , then we have the sum over all of the classes. Next, we normalize by the scalar quantity, $N - K$.
- When we fit a MLE it should be divided by N , but if it is divided by $N - K$, we get an *unbiased estimator*.
- Remember, K is the number of classes. So, when N is large, the difference between N and $N - K$ is pretty small.
- Note that $x^{(i)}$ denotes the i th sample vector.

LDA: Example

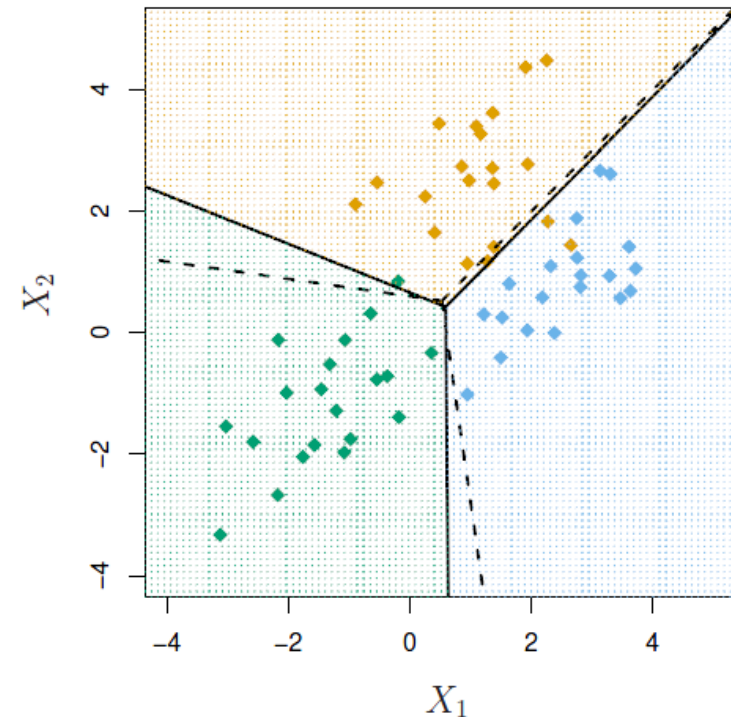
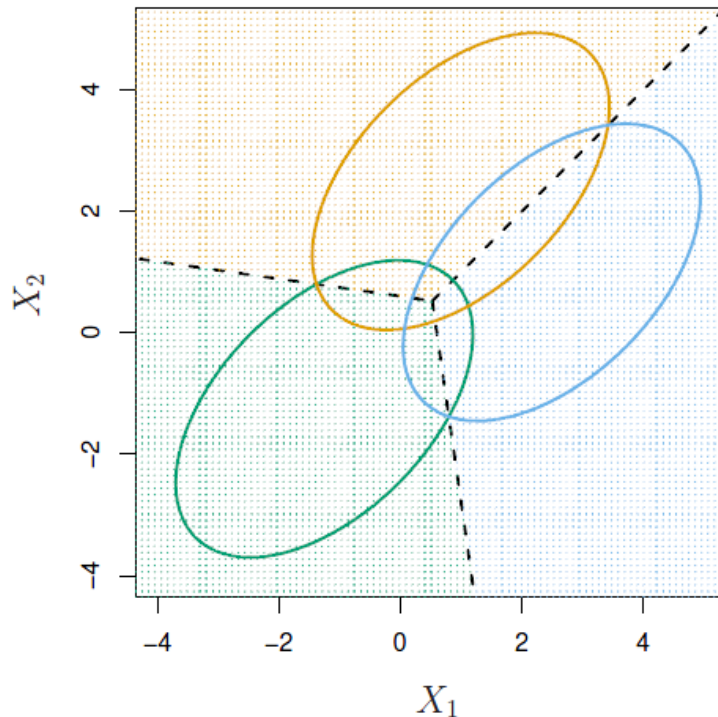
- Suppose we have only one predictor ($p = 1$), and two normal density functions $f_1(x)$ and $f_2(x)$, represent two distinct classes.
- The two density functions overlap, so there is some uncertainty about the class to which an observation with an unknown class belongs.



Example with $\mu_1 = -1.5$, $\mu_2 = 1.5$, $\pi_1 = \pi_2 = 0.5$, and $\sigma^2 = 1$.

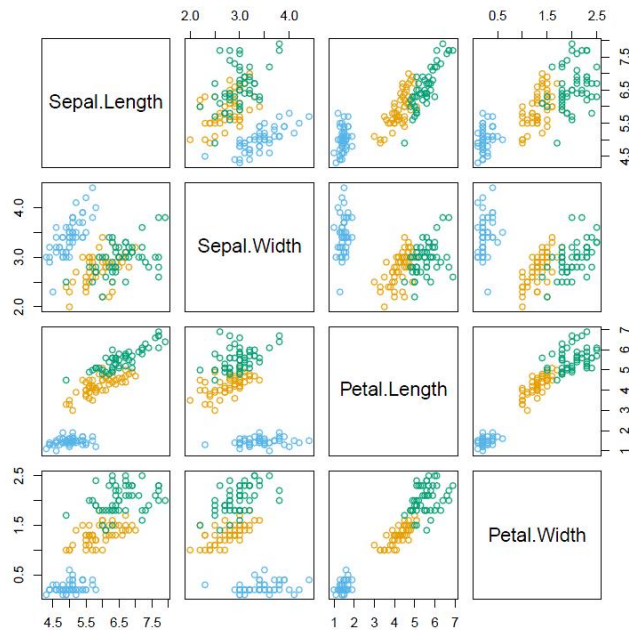
LDA: Example (cont.)

- If X is multidimensional where $p = 2$ and $K = 3$ classes:



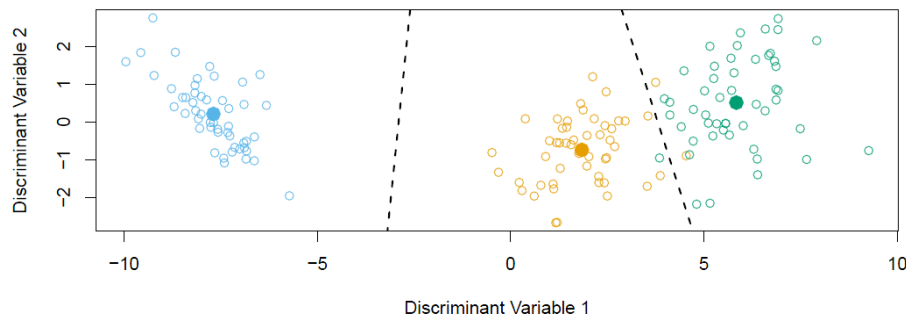
Here $\pi_1 = \pi_2 = \pi_3 = 1/3$.

LDA: Example (cont.)



- Using Fisher's Iris data set, we have:
 - 4 variables (sepal length, sepal width, petal length, petal width)
 - 3 species (setosa, versicolor, virginica)
 - 50 samples per class

- LDA correctly classifies all but 3 out of the 150 training samples.



Performance of a Classifier

- LDA is trying to approximate the Bayes classifier, which has the lowest total error rate out of all classifiers (if the Gaussian model is correct).
- In other words, the Bayes classifier will yield the smallest possible total number of misclassified observations, irrespective of which class the errors come from.

- Example of *confusion matrix*:

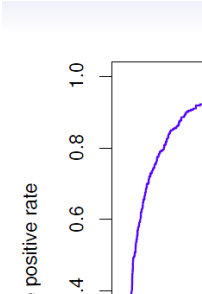
		<i>True Default Status</i>		
		No	Yes	Total
<i>Predicted Default Status</i>	No	9644	252	9896
	Yes	23	81	104
Total		9667	333	10000

Performance of a Classifier (cont.)

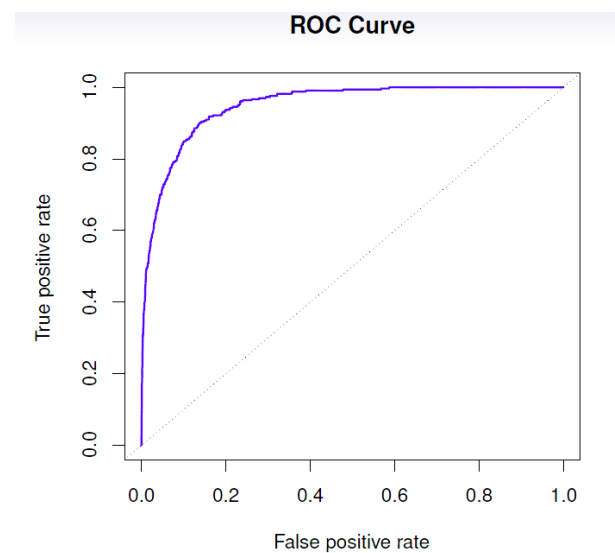
- Note that the results in the confusion matrix depend upon the establish *threshold* value, which is used to perform the classification based on the posterior probability.
- The *receiver operating characteristics* (ROC) curve is a popular graphic for simultaneously displaying the two types of errors for all possible thresholds.
- The overall performance of a classifier, summarized over all possible thresholds, is given by the *area under* the ROC curve.
 - The larger the area under the curve, the better the classifier!

Performance of a Classifier (cont.)

	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Test outcome	Test outcome positive	True positive	False positive (Type I error)	Positive predictive value (PPV, Precision) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Test outcome negative	False negative (Type II error)	True negative	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Positive likelihood ratio (LR+) = TPR/FPR	True positive rate (TPR, Sensitivity, Recall) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR, Fall-out) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
	Negative likelihood ratio (LR-) = FNR/TNR	False negative rate (FNR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR, Specificity, SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$		
	Diagnostic odds ratio (DOR) = LR+/LR-	http://en.wikipedia.org/wiki/Sensitivity_and_specificity			



A Receiver Operating Characteristic (ROC) curve is shown on the right side of the image. The y-axis is labeled 'positive rate' and ranges from 0.4 to 1.0. The x-axis represents the false positive rate, though it is not explicitly labeled. The curve is a blue line that starts at the bottom-left and curves towards the top-right, indicating a model's performance in distinguishing between classes.



Quadratic Discriminant Analysis

- QDA is not much different from LDA except that you assume that the covariance matrix can be different for each class.
- Thus, we will estimate the covariance matrix Σ_k separately for each class k , $k=1, 2, \dots, K$.
- We get the following quadratic discriminant function:

$$\delta_k(x) = -\frac{1}{2} \log|\Sigma_k| - \frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log \pi_k$$

Quadratic Discriminant Analysis (cont.)

- This quadratic discriminant function is very much like the linear discriminant function except that because Σ_k is not identical, you cannot throw away the quadratic terms.
- This discriminant function is a quadratic function and will contain second order terms.
- We get the following classification rule:

$$\hat{G}(x) = \arg \max_k \delta_k(x)$$

Quadratic Discriminant Analysis (cont.)

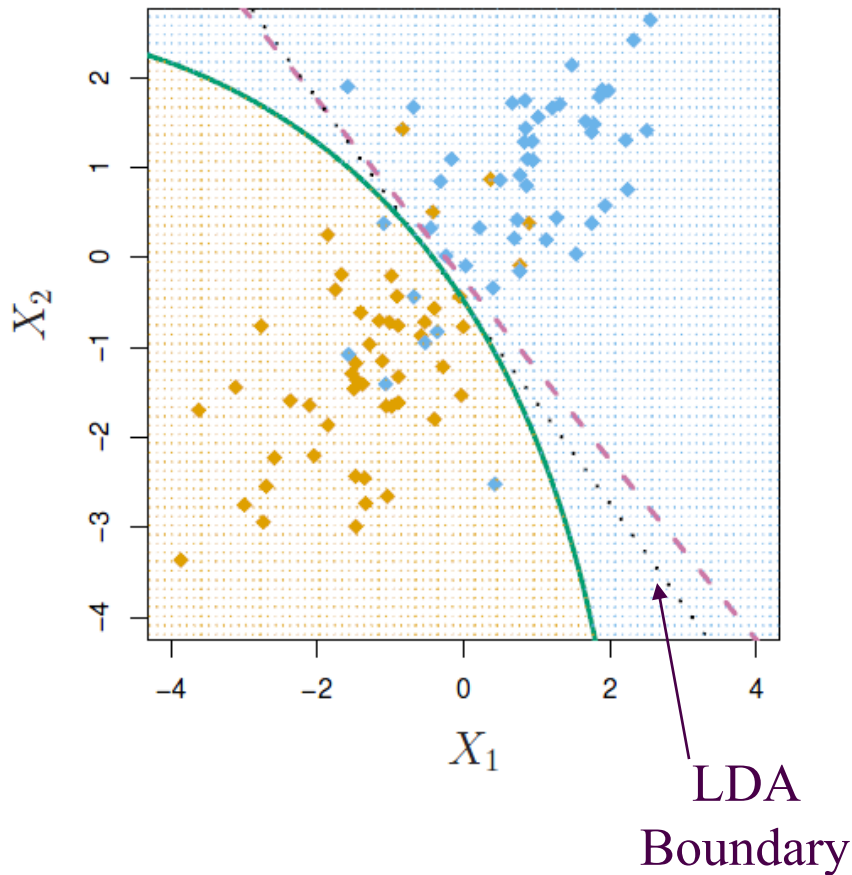
- For the classification rule, we find the class k which maximizes the quadratic discriminant function. The decision boundaries are quadratic equations in x .
- QDA, because it allows for more flexibility for the covariance matrix, tends to fit the data better than LDA, but then it has more parameters to estimate.
- The number of parameters increases significantly with QDA because of the separate covariance matrix for every class. If we have many classes and not so many sample points, this can be a problem.

Quadratic Discriminant Analysis (cont.)

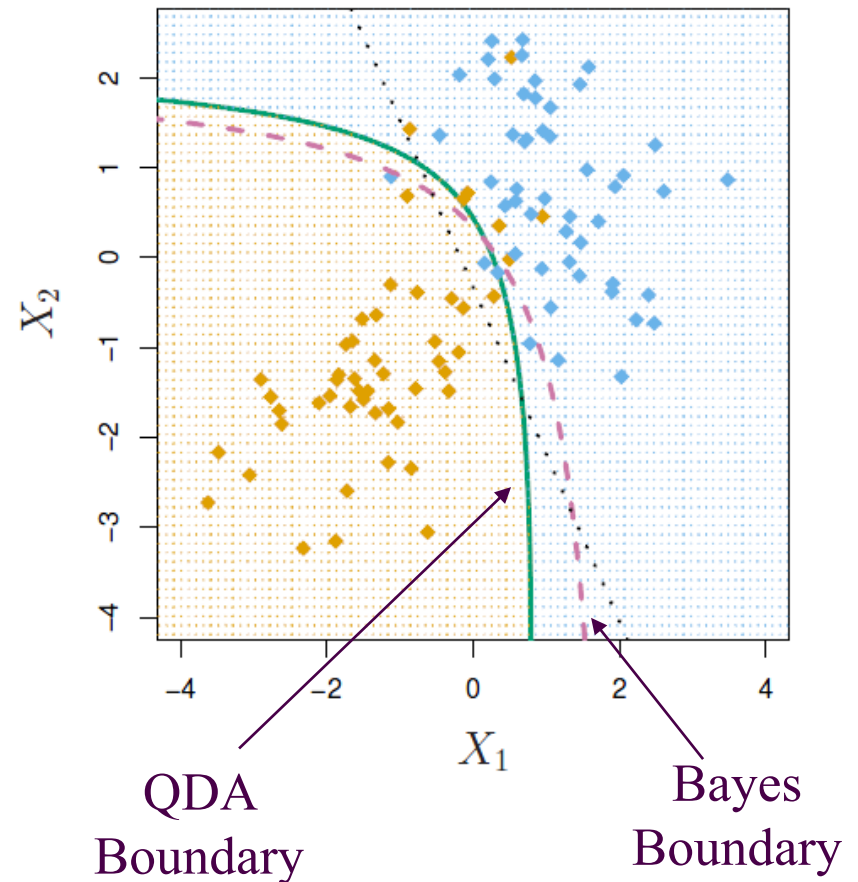
- As a result, there are trade-offs between fitting the training data well and having a simple model to work with.
- A simple model sometimes fits the data just as well as a complicated model.
- Even if the simple model does not fit the training data as well as a complex model, it still might be better on the test data because it is more robust.

LDA versus QDA

LDA is better!



QDA is better!



LDA and Logistic Regression

- Under the model of LDA, we can compute the log-odds:

$$\begin{aligned} & \log \frac{Pr(G = k | X = x)}{Pr(G = K | X = x)} \\ &= \log \frac{\pi_k}{\pi_K} - \frac{1}{2} (\mu_k + \mu_K)^T \Sigma^{-1} (\mu_k - \mu_K) \\ &= a_{k0} + a_k^T x \end{aligned}$$

- The model of LDA satisfies the assumption of the linear logistic model.

LDA and Logistic Regression (cont.)

- The difference between linear logistic regression and LDA is that the linear logistic model only specifies the conditional distribution $\Pr(G = k \mid X = x)$.
- No assumption is made about $\Pr(X)$, while the LDA model specifies the joint distribution of X and G .
- $\Pr(X)$ is a mixture of Gaussians (where ϕ is the Gaussian density function):

$$\Pr(X) = \sum_{k=1}^K \pi_k \phi(X; \mu_k, \Sigma)$$

LDA and Logistic Regression (cont.)

- Moreover, linear logistic regression is solved by maximizing the conditional likelihood of G given X : $\Pr(G = k \mid X = x)$, while LDA maximizes the joint likelihood of G and X : $\Pr(X = x, G = k)$.
- If the additional assumption made by LDA is appropriate, LDA tends to estimate the parameters more efficiently by using more information about the data.
- Another advantage of LDA is that samples without class labels can be used under the model of LDA. On the other hand, LDA is not robust to gross outliers.
- Because logistic regression relies on fewer assumptions, it seems to be more robust to non-Gaussian type of data.

K-Nearest Neighbors

- These classifiers are *memory-based* and require no model to be fit.
- Training data: $(g_i, x_i), i = 1, 2, \dots, N$
 1. Define distance on input x (e.g. Euclidian distance)
 2. Classify new instance by looking at the label of the single closest sample in the training set:

$$\hat{G}(x^*) = \operatorname{argmin}_i d(x_i, x^*)$$

***K*-Nearest Neighbors (cont.)**

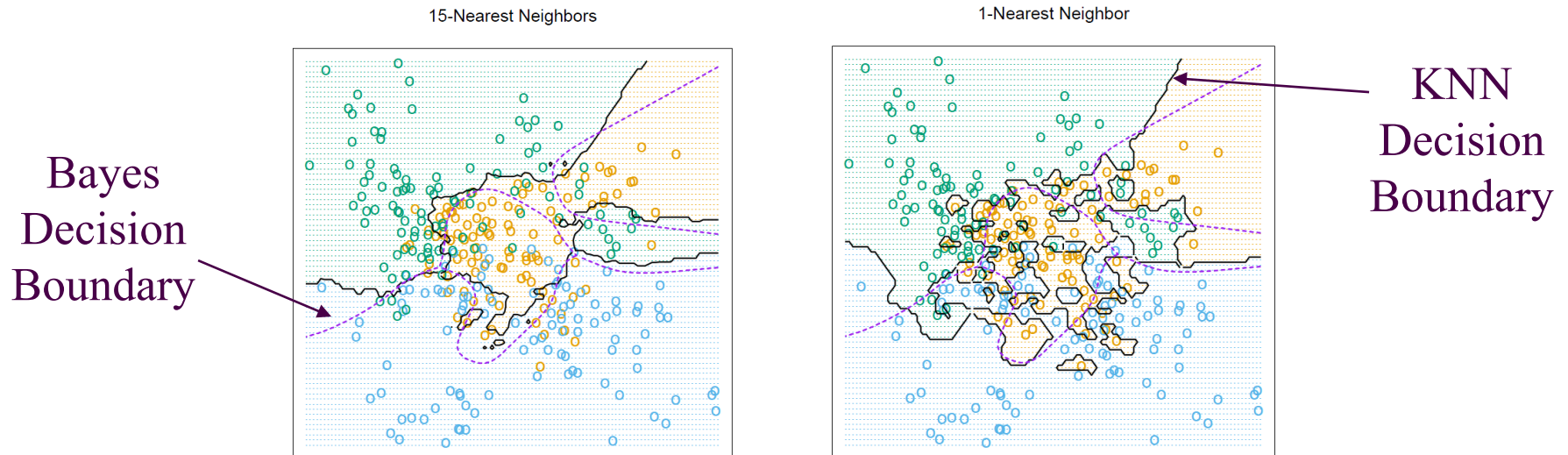
- By looking at only the closest sample, overfitting the data can be a huge problem.
- To prevent overfitting, we can smooth the decision boundary by K nearest neighbors instead of 1.
- Find the K training samples x_r , $r = 1, \dots, K$ closest in distance to x^* , and then classify using majority vote among the k neighbors.
- The amount of computation can be intense when the training data is large since the distance between a new data point and every training point has to be computed and sorted.

K-Nearest Neighbors (cont.)

- Feature *standardization* is often performed in pre-processing.
- Because standardization affects the distance, if one wants the features to play a similar role in determining the distance, standardization is recommended.
- However, whether to apply normalization is rather subjective.
- One has to decide on an individual basis for the problem in consideration.

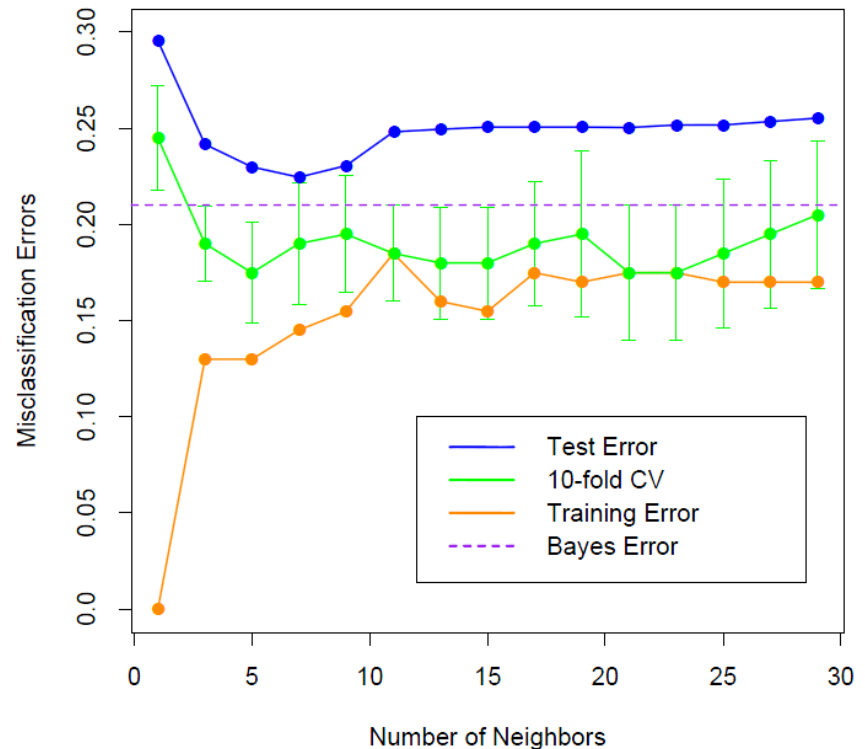
K-Nearest Neighbors (cont.)

- The only parameter that can adjust the complexity of KNN is the number of neighbors k .
- The larger k is, the smoother the classification boundary. Or we can think of the complexity of KNN as lower when k increases.



K -Nearest Neighbors (cont.)

- For another simulated data set, there are two classes. The error rates based on the training data, test data, and 10-fold cross validation are plotted against K , the number of neighbors.
- We can see that the training error rate tends to grow when k grows, which is not the case for the error rate based on a separate test data set or cross-validation.



Comparison of Classification Methods

- KNN is completely non-parametric, as there are no assumptions made about the shape of the decision boundary.
- We can expect KNN to dominate both LDA and logistic regression when the decision boundary is highly non-linear.
- On the other hand, KNN does not tell us which predictors are important; there is no table of coefficients.
- QDA serves as a compromise between the non-parametric KNN method and the linear LDA and logistic regression approaches.

Comparison of Classification Methods

- Logistic regression is very popular for classification, especially when $K = 2$.
- LDA is useful when n is small, the classes are well separated (and Gaussian assumptions are reasonable), and when $K > 2$.
- Finally, Naive Bayes is useful when p is very large.