

Generative Classifiers

LDQ, QDA, Naive Bayes

DS 6030 | Fall 2022

gen-classifiers.pdf

Contents

1	Classification and Pattern Recognition	2
1.1	Binary Classification	2
1.2	Two-Class Example	3
1.3	Discriminative Models	3
2	Generative Classification Models	5
2.1	From Discriminative to Generative, and Back Again	6
3	Linear/Quadratic Discriminant Analysis (LDA/QDA)	9
3.1	Estimation	9
3.2	LDA/QDA in Action	12
3.3	Connections: LDA, QDA, and Logistic Regression	12
4	Kernel Discriminant Analysis (KDA)	14
4.1	KDA with R	14
5	Naive Bayes	15
5.1	Gaussian Naive Bayes	15
5.2	Kernel Naive Bayes	18
6	Connections: Generalized Additive Models (GAM)	19

1 Classification and Pattern Recognition

- The outcome variable is categorical and denoted $G \in \mathcal{G}$
 - Default Credit Card Example: $\mathcal{G} = \{\text{"Yes"}, \text{"No"}\}$
 - Medical Diagnosis Example: $\mathcal{G} = \{\text{"stroke"}, \text{"heart attack"}, \text{"drug overdose"}, \text{"vertigo"}\}$
- The training data is $D = \{(X_1, G_1), (X_2, G_2), \dots, (X_n, G_n)\}$
- The optimal decision/classification is often based on the posterior probability $\Pr(G = g \mid \mathbf{X} = \mathbf{x})$

1.1 Binary Classification

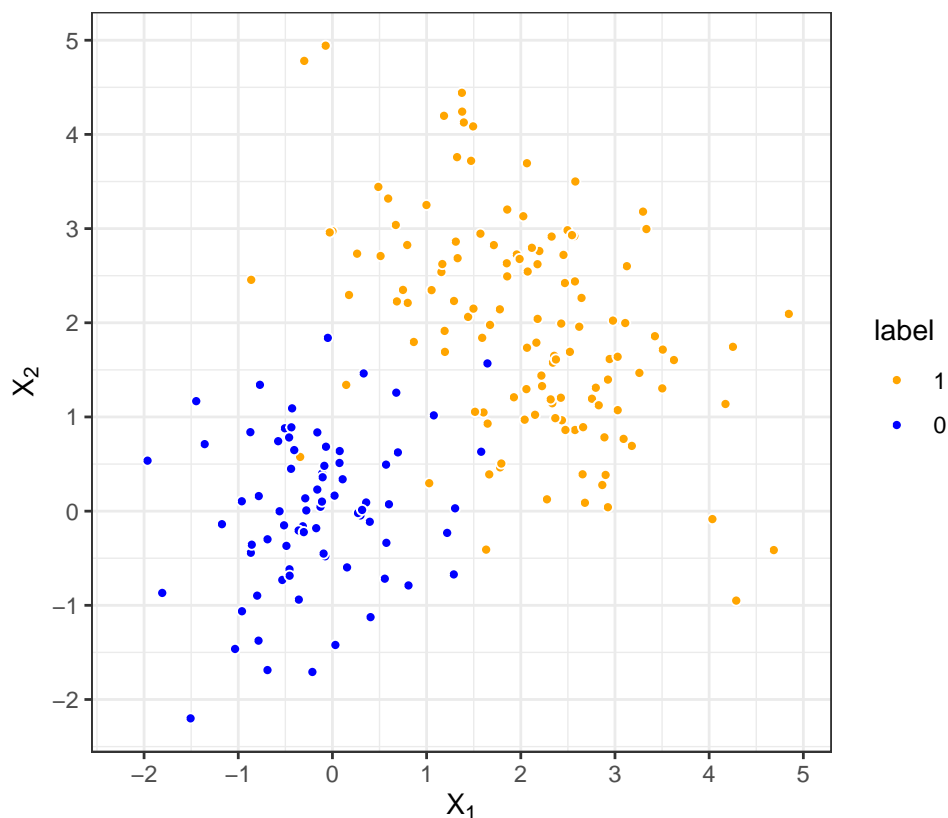
- Classification is simplified when there are only 2 classes.
 - Many multi-class problems can be addressed by solving a set of binary classification problems (e.g., [one-vs-rest](#)).
- It is often convenient to transform the outcome variable to a binary $\{0, 1\}$ variable:

$$Y_i = \begin{cases} 1 & G_i = \mathcal{G}_1 \\ 0 & G_i = \mathcal{G}_2 \end{cases} \quad (\text{outcome of interest})$$

- Or, like with SVM, as a $\{-1, +1\}$ variable:

$$Y_i = \begin{cases} +1 & G_i = \mathcal{G}_1 \\ -1 & G_i = \mathcal{G}_2 \end{cases} \quad (\text{outcome of interest})$$

1.2 Two-Class Example



1.3 Discriminative Models

- The models we have covered in this course so far (Linear Regression, Logistic Regression, SVM, and KNN) can be considered *discriminative* models.
- Their goal is to directly estimate $\Pr(Y = 1 \mid X = x)$ **conditional on** $X = x$.

$$p(x) = \Pr(Y = 1 \mid X = x)$$

a. Linear Regression (for binary outcomes)

$$\hat{p}(x) = \hat{\beta}^\top x$$

b. Logistic Regression

$$\log\left(\frac{\hat{p}(x)}{1 - \hat{p}(x)}\right) = \hat{\beta}^\top x$$

and thus,

$$\begin{aligned}\hat{p}(x) &= \frac{e^{\hat{\beta}^\top x}}{1 + e^{\hat{\beta}^\top x}} \\ &= \left(1 + e^{-\hat{\beta}^\top x}\right)^{-1}\end{aligned}$$

c. kNN (for binary outcomes)

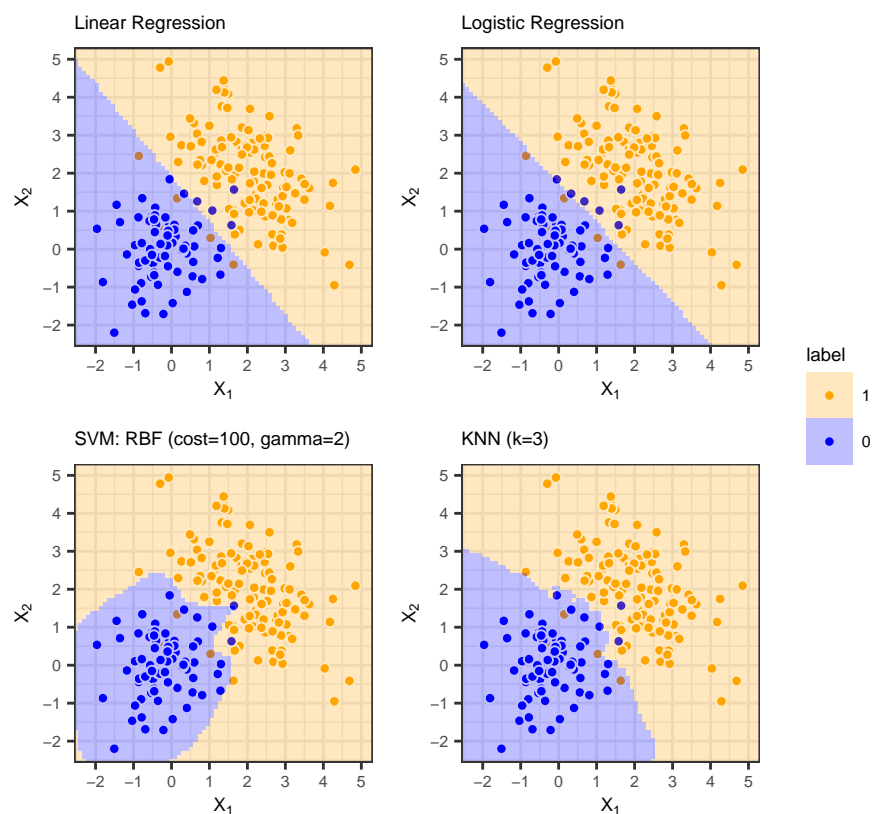
$$\begin{aligned}\hat{p}(x; k) &= \frac{1}{k} \sum_{i: x_i \in N_k(x)} y_i \\ &= \text{Avg}(y_i \mid x_i \in N_k(x))\end{aligned}$$

- $N_k(x)$ are the set of k nearest neighbors

d. Support Vector Machines (SVM)

$$\hat{g}(x) = \hat{\beta}_0 + \sum_{i=1}^n \hat{\alpha}_i y_i K(x, x_i)$$

- Decide $\hat{Y} = 1$ if $\hat{g}(x) > 0$
- Or calibrated probability: $\log \frac{\hat{p}(x)}{1-\hat{p}(x)} = \hat{\alpha}_0 + \hat{\alpha}_1 \hat{g}(x)$
 - I.e., using logistic regression with $\hat{g}(x)$ as the predictor.



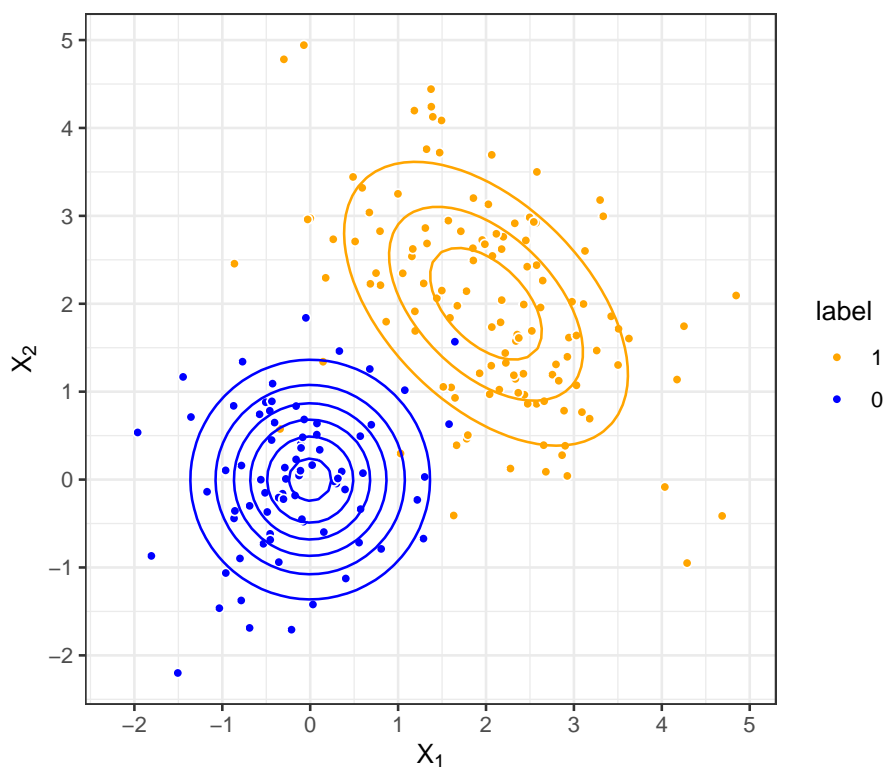
2 Generative Classification Models

Consider how the data $D = \{(X_1, G_1), (X_2, G_2), \dots, (X_n, G_n)\}$ could be generated.

1. First, the class label is selected according to the *prior probabilities* $\pi = [\pi_1, \dots, \pi_K]$.
 - That is, $\Pr(G_i = k) = \pi_k$
2. Given the class is k , the X value is generated $X | G = k \sim f_k$
 - Let $f_k(\mathbf{x})$ be the (pdf/pmf/mixed) of the predictors from class k .
3. Repeat n times

Example

- Two classes, $k \in \{0, 1\}$
 - $\pi_1 = 0.6, \pi_0 = 0.4$
 - I expect 60% of the observations to be from **class 1**.
- If $G_i = 1$, then $X \sim N\left(\mu_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\right)$
- If $G_i = 0$, then $X \sim N\left(\mu_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \Sigma_0 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\right)$



2.1 From Discriminative to Generative, and Back Again

- The models we have discussed in this course so far are considered *discriminative* and focused on estimating the **conditional** probability $\Pr(Y = k \mid X = x)$
- But there is another class of models termed *generative* which try to directly estimate the **joint** probability $\Pr(Y = k, X = x) = \Pr(X = x \mid Y = k) \Pr(Y = k)$

2.1.1 The Bayes Breakdown (Binary Classification)

Bayes Theorem

$$\begin{aligned} p(x) = \Pr(Y = 1 \mid X = x) &= \frac{\Pr(X = x \mid Y = 1) \Pr(Y = 1)}{\Pr(X = x)} \\ &= \frac{f_1(x)\pi}{f_1(x)\pi + f_0(x)(1 - \pi)} \end{aligned}$$

- $f_k(x)$ is the *class conditional density*
- $0 \leq \pi_k \leq 1$ are the *prior class probabilities*
- $\pi_0 + \pi_1 = 1$
- X is distributed as a finite mixture model

$$f(x) = \sum_j f_j(x)\pi_j$$

Recall our notation for the log-odds:

$$\gamma(x) = \log \frac{p(x)}{1-p(x)}$$

The log-odds reduces to a combination of prior odds and density ratios

$$\begin{aligned} \gamma(x) &= \log \left(\frac{p(x)}{1-p(x)} \right) \\ &= \log \left(\frac{f_1(x)\pi}{f_0(x)(1-\pi)} \right) \\ &= \underbrace{\log \left(\frac{\pi}{1-\pi} \right)}_{\text{log prior odds}} + \underbrace{\log \left(\frac{f_1(x)}{f_0(x)} \right)}_{\text{log density ratio}} \end{aligned}$$

2.1.2 Decision-Making (Hard Classification)

- We know that the optimal decision can be based on the density ratios

Choose $\hat{G}(x) = 1$ if:

$$\hat{\gamma}(x) > \log \left(\frac{C_{\text{FP}}}{C_{\text{FN}}} \right)$$

$$\log \left(\frac{1 - \hat{\pi}}{\hat{\pi}} \right) + \log \left(\frac{\hat{f}_1(x)}{\hat{f}_0(x)} \right) > \log \left(\frac{C_{\text{FP}}}{C_{\text{FN}}} \right)$$

$$\log \left(\frac{\hat{f}_1(x)}{\hat{f}_0(x)} \right) > \log \left(\frac{1 - \hat{\pi}}{\hat{\pi}} \right) + \log \left(\frac{C_{\text{FP}}}{C_{\text{FN}}} \right)$$

2.1.3 Estimation

- $\hat{\pi}_k = n_k/n$ is a natural estimate for the class priors if we think the testing data will have the same proportions as the training data
- The other term to estimate is the log density ratio: $\log \left(\frac{\widehat{f_1(x)}}{\widehat{f_0(x)}} \right)$
- Generative Models estimate this term by

$$\log \left(\frac{\widehat{f_1(x)}}{\widehat{f_0(x)}} \right) = \log \left(\frac{\hat{f}_1(x)}{\hat{f}_0(x)} \right)$$

- That is, generative models estimate the class conditional densities $\{f_k(\cdot)\}$
- The different generative models take different approaches to estimate these component densities

Generative Models

Generative Classification Models use *density estimation* to make predictions!

2.1.3.1 Linear/Quadratic Discriminant Analysis (LDA/QDA)

- Both LDA and QDA model the class conditional densities $f_k(x)$ with a *Gaussian* density
 - Thus, they model the observations as coming from a *Gaussian mixture model*
 - Each class has its own mean vector μ_k
 - The difference between LDA and QDA is what they use for their covariance matrix

- **LDA**

$$f_k(x) = (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_k)^\top \Sigma^{-1} (\mathbf{x} - \mu_k) \right\}$$

- $\Sigma_k = \Sigma \quad \forall k$ (uses the same variance-covariance for all classes)

- **QDA**

$$f_k(x) = (2\pi)^{-p/2} |\Sigma_k|^{-1/2} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_k)^\top \Sigma_k^{-1} (\mathbf{x} - \mu_k) \right\}$$

- Σ_k is different for each classes

2.1.3.2 Kernel Discriminant Analysis (KDA)

- Model the class conditional densities $f_k(x)$ with a multivariate *kernel density estimate (KDE)*

$$\hat{f}_k(x) = \frac{1}{n_k} \sum_{i:g_i=k} K(x - x_i; H)$$

where H is the $p \times p$ bandwidth matrix.

2.1.3.3 Mixture Discriminant Analysis (MDA)

- Model the class conditional densities $f_k(x)$ with a *finite mixture model*

$$\hat{f}_k(x) = \frac{1}{J} \sum_{j=1}^J \pi_j g_j(x; \theta_j)$$

where $\sum_{j=1}^J \pi_j = 1$ and $g_j(x)$ is a density function (e.g., Gaussian).

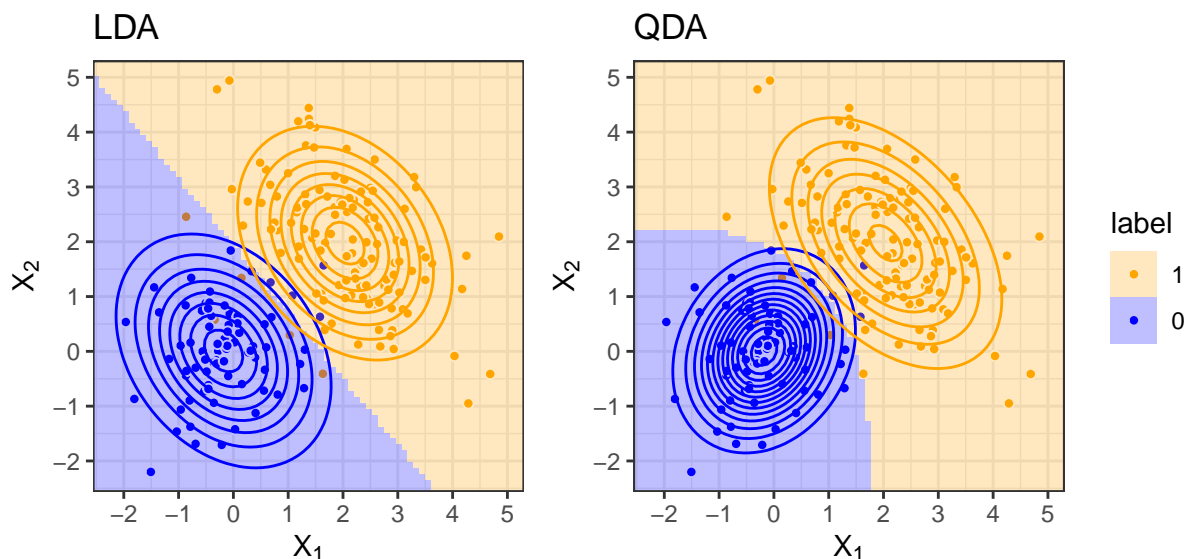
2.1.3.4 Naive Bayes

- **Naive Bayes** ignores potential associations between predictors and estimates the density of each predictor variable independently.

$$\hat{f}_k(x) = \prod_{j=1}^p \hat{f}_{jk}(x_j)$$

- This greatly simplifies the estimation
- You will often find $\hat{f}_{jk}(u) = \mathcal{N}(u; \hat{\mu}_{jk}, \hat{\sigma}_{jk})$
- But KDE is a great approach $\hat{f}_{jk}(u) = \frac{1}{n_k} \sum_{\{i: G_i=k\}} K_h(u - x_{ij})$
- And mix continuous and discrete variables is very easily

3 Linear/Quadratic Discriminant Analysis (LDA/QDA)



- *Linear Discriminant Analysis (LDA)* finds *linear* boundaries between classes
- *Quadratic Discriminant Analysis (QDA)* finds *quadratic* boundaries between classes
- Setup: $K = |\mathcal{G}|$ classes in the training data, $D = \{(\mathbf{X}_i, G_i)\}_{i=1}^n$
 - where $\mathbf{X}_i \in \mathbf{R}^p$, $G_i \in \mathcal{G}$
- The posterior probability of class g , given $X = x$,

$$\begin{aligned} \Pr(G = g \mid \mathbf{X} = \mathbf{x}) &= \frac{f(x \mid G = g) \Pr(G = g)}{f(x)} \\ &= \frac{f_g(x) \pi_g}{\sum_{k=1}^K f_k(x) \pi_k} \end{aligned}$$

- $f_k(x)$ is the *class conditional density*
- $0 \leq \pi_k \leq 1$ are the *prior class probabilities*; $\sum_{k=1}^K \pi_k = 1$

3.1 Estimation

- Both LDA and QDA model the class conditional densities $f_k(x)$ with *Gaussians*
 - Thus, they model the observations as coming from a K component *Gaussian mixture model*
 - Each class has its own mean vector μ_k
 - The difference between LDA and QDA is what they use for their covariance matrix

$$f_k(x) = \mathcal{N}(x; \mu_k, \Sigma_k)$$

- LDA: $\hat{\Sigma}_1 = \hat{\Sigma}_2 = \dots = \hat{\Sigma}_K = \hat{\Sigma}$ Common covariance
- QDA: $\hat{\Sigma}_1 \neq \hat{\Sigma}_2 \neq \dots \neq \hat{\Sigma}_K$ Different covariances

• LDA

$$f_k(x) = (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_k)^\top \Sigma^{-1} (\mathbf{x} - \mu_k) \right\}$$

- $\Sigma_k = \Sigma \quad \forall k$ (uses the same variance-covariance for all classes)

- QDA

$$f_k(x) = (2\pi)^{-p/2} |\Sigma_k|^{-1/2} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_k)^\top \Sigma_k^{-1} (\mathbf{x} - \mu_k) \right\}$$

– Σ_k is *different* for each classes

Your Turn #1 : Model Complexity

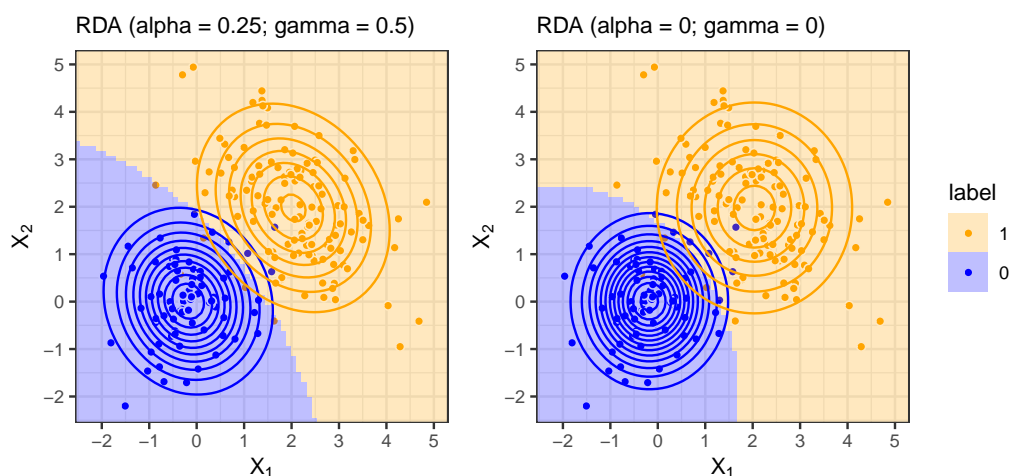
The LDA model uses a common covariance matrix while QDA allows each class to have a different covariance (which permits quadratic boundaries). But this flexibility comes at a cost.

1. How many parameters have to be estimated in an LDA model with K classes and p dimensions?

2. How many parameters have to be estimated in an QDA model with K classes and p dimensions?

- There are a few methods to maintain some flexibility, yet protect the model from high variance
- One is to use a *regularized covariance matrix* (see ESL 4.3.1). Called Regularized Discriminant Analysis (RDA)

$$\hat{\Sigma}_k(\alpha, \gamma) = \alpha \hat{\Sigma}_k + (1 - \alpha) \{ \gamma \hat{\Sigma} + (1 - \gamma) \hat{\sigma}^2 I_p \}$$



- A special case of above using diagonal covariance matrices only ($\hat{\Sigma}_k(\alpha = 0, \gamma = 0)$). This covariance

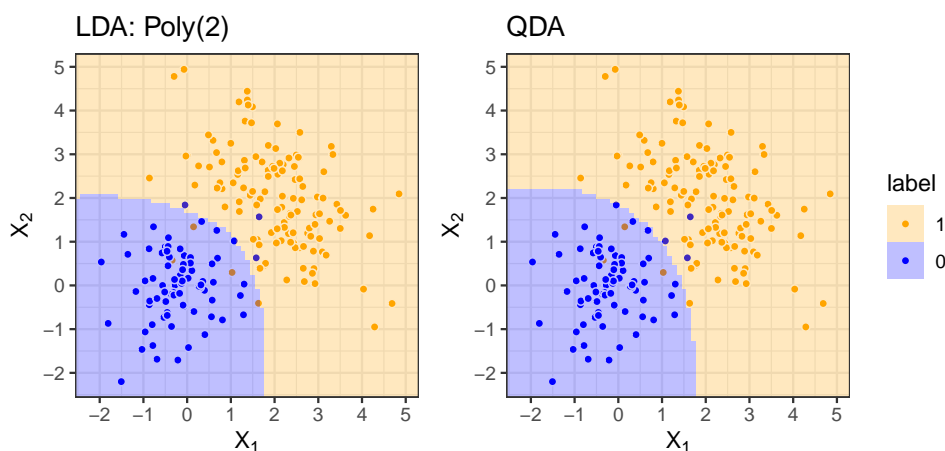
matrix has all off-diagonal terms set to 0.

$$\begin{aligned}\hat{\Sigma}_k &= \text{diag}(\hat{\sigma}_1^2, \hat{\sigma}_2^2, \dots, \hat{\sigma}_p^2) \\ &= \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_p^2 \end{bmatrix}\end{aligned}$$

- This treats predictors/features as uncorrelated/independent.
- It is a special case of *Naive Bayes*!
- A more restrictive (less complex) model specifies that variance in all dimensions are equal

$$\begin{aligned}\hat{\Sigma}_k &= \hat{\sigma}^2 I_p \\ &= \begin{bmatrix} \sigma^2 & 0 & \dots & 0 \\ 0 & \sigma^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma^2 \end{bmatrix}\end{aligned}$$

- This treats predictors/features as uncorrelated/independent.
- It is a special case of *Naive Bayes*!
- Models all variances as equal.
- In some settings (large K , small p), edf could be reduced by fitting an LDA model in an *enlarged feature space*
 - E.g., for $p = 2$ dimensions, use $X_1, X_2, X_1 \cdot X_2, X_1^2, X_2^2$ instead of QDA in X_1, X_2 .
 - Think basis expansion like what we did with polynomial regression or B-splines
 - Or kernels with SVM



Mahalanobis Distance

Notice that a multivariate normal density is a function of the squared *Mahalanobis* distance from x to the mean.

$$\begin{aligned}f(\mathbf{x}) &= (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu) \right\} \\ &= (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} D_m^2(x) \right\}\end{aligned}$$

where

$$D_m(x) = \sqrt{(\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu)}$$

is the Mahalanobis distance.

3.2 LDA/QDA in Action

- In **R**, LDA and QDA can be implemented with the `lda()` and `qda()` functions from the MASS package.
- See ISLR 4.7 for details
- Warning: the MASS package has a `select()` functions that conflicts with dplyr's `select()`. If you use tidyverse, I suggest you use `MASS::lda()` and `MASS::qda()` instead of loading the MASS package.

3.3 Connections: LDA, QDA, and Logistic Regression

ISL 4.5 and ESL 4.4.5 show more details about the parametric form LDA and QDA take.

Recall the notation for generative models:

$$\begin{aligned}\hat{\gamma}(x) &= \log \left(\frac{\hat{p}(x)}{1 - \hat{p}(x)} \right) \\ &= \log \left(\frac{\hat{\pi}}{1 - \hat{\pi}} \right) + \log \left(\frac{\hat{f}_1(x)}{\hat{f}_0(x)} \right)\end{aligned}$$

Logistic Regression

$$\hat{\gamma}(x) = \hat{\beta}_0 + \sum_{j=1}^p \hat{\beta}_j x_j$$

LDA

$$\hat{\gamma}(x) = \hat{\alpha}_0 + \sum_{j=1}^p \hat{\alpha}_j x_j$$

$$\hat{\alpha}_0 = \log \frac{\hat{\pi}}{1 - \hat{\pi}} - \frac{1}{2} (\hat{\mu}_1 - \hat{\mu}_0)^\top \hat{\Sigma}^{-1} (\hat{\mu}_1 - \hat{\mu}_0)$$

$$\hat{\alpha}_j = \text{the } j\text{th element of } \hat{\Sigma}^{-1} (\hat{\mu}_1 - \hat{\mu}_0)$$

QDA

$$\hat{\gamma}(x) = \hat{\alpha}_0 + \sum_{j=1}^p \hat{\alpha}_j x_j + \sum_{j=1}^p \sum_{k=1}^p \hat{a}_{jk} x_j x_k$$

$$\hat{\alpha}_0 = \log \frac{\hat{\pi}}{1 - \hat{\pi}} - \frac{1}{2} \log \frac{|\hat{\Sigma}_1|}{|\hat{\Sigma}_0|} - \frac{1}{2} (\hat{\mu}_1^\top \Sigma_1^{-1} - \hat{\mu}_0^\top \Sigma_0^{-1})$$

$$\hat{\alpha}_j = \text{the } j\text{th element of } \hat{\Sigma}_1^{-1} \hat{\mu}_1 - \hat{\Sigma}_0^{-1} \hat{\mu}_0$$

$$\hat{a}_{jk} = \text{the } (j, k)\text{th element of } (\hat{\Sigma}_0^{-1} - \hat{\Sigma}_1^{-1})/2$$

3.3.1 Estimation

LDA and QDA estimates model parameters by maximizing the *joint* likelihood:

$$\begin{aligned}\hat{\alpha} &= \arg \max_{\alpha} \Pr(X, Y) \\ &= \arg \max_{\alpha} \Pr(X | Y) \Pr(Y) \\ &= \arg \max_{\alpha} \Pr(Y | X) \Pr(X)\end{aligned}$$

Logistic Regression estimates model parameters by maximizing the *conditional* likelihood

$$\hat{\beta} = \arg \max_{\beta} \Pr(Y | X)$$

4 Kernel Discriminant Analysis (KDA)

- Model the class conditional densities $f_k(x)$ with a multivariate *kernel density estimate (KDE)*

$$f_k(x) = \frac{1}{n_k} \sum_{i:g_i=k} K(x - x_i; H)$$

where H is the $p \times p$ bandwidth matrix.

There are three primary approaches to multivariate (p dimensional) KDE:

1. Multivariate kernels

- e.g., $K(u) = N(\mathbf{0}, H)$:

$$\hat{f}(x) = \frac{1}{(2\pi)^{d/2} |H|^{1/2} n} \sum_{i=1}^n \exp \left(-\frac{1}{2} (x - x_i)^T H^{-1} (x - x_i) \right)$$

2. Product Kernels

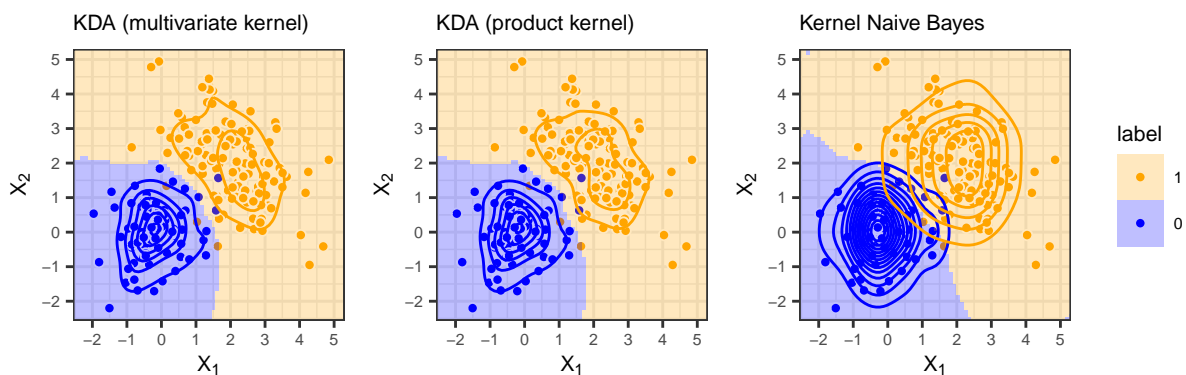
- $H = \text{diag}(h_1, h_2, \dots, h_p)$

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^n \left(\prod_{j=1}^p K(x_j - x_{ij}; h_j) \right)$$

3. Independence

- This is a special case of *Naive Bayes* (Kernel Naive Bayes)!

$$\begin{aligned} \hat{f}(x) &= \prod_{j=1}^p \hat{f}_j(x) \\ &= \prod_{j=1}^p \left(\frac{1}{n} \sum_{i=1}^n K(x_j - x_{ij}; h_j) \right) \end{aligned}$$



4.1 KDA with R

- In R, the `ks::kda()` function (ks package) implements Kernel Discriminant Analysis.

5 Naive Bayes

Naive Bayes is a generative model that ignores potential associations between predictors and estimates the density of each predictor variable independently.

$$\hat{f}_k(x) = \prod_{j=1}^p \hat{f}_{kj}(x_j)$$

- This greatly simplifies the estimation
- The densities do *not* have to be Gaussian (e.g., KDE is a good option)
- Categorical densities (i.e., pmfs) can be thrown in the mix without a problem
- Because of the independence, this is easy to implement in parallel (and thus can be fast)
- The decision function becomes:

$$\begin{aligned}\hat{\gamma}(x) &= \log \left(\frac{\hat{p}(x)}{1 - \hat{p}(x)} \right) \\ &= \log \left(\frac{\hat{\pi}}{1 - \hat{\pi}} \right) + \log \left(\frac{\hat{f}_1(x)}{\hat{f}_0(x)} \right) \\ &= \log \left(\frac{\hat{\pi}}{1 - \hat{\pi}} \right) + \log \left(\frac{\prod_{j=1}^p \hat{f}_{1j}(x_j)}{\prod_{j=1}^p \hat{f}_{0j}(x_j)} \right) \\ &= \log \left(\frac{\hat{\pi}}{1 - \hat{\pi}} \right) + \log \left(\prod_{j=1}^p \frac{\hat{f}_{1j}(x_j)}{\hat{f}_{0j}(x_j)} \right) \\ &= \log \left(\frac{\hat{\pi}}{1 - \hat{\pi}} \right) + \sum_{j=1}^p \log \left(\frac{\hat{f}_{1j}(x_j)}{\hat{f}_{0j}(x_j)} \right)\end{aligned}$$

5.1 Gaussian Naive Bayes

- Recall in LDA/QDA, the class conditional densities were estimated as Gaussians:

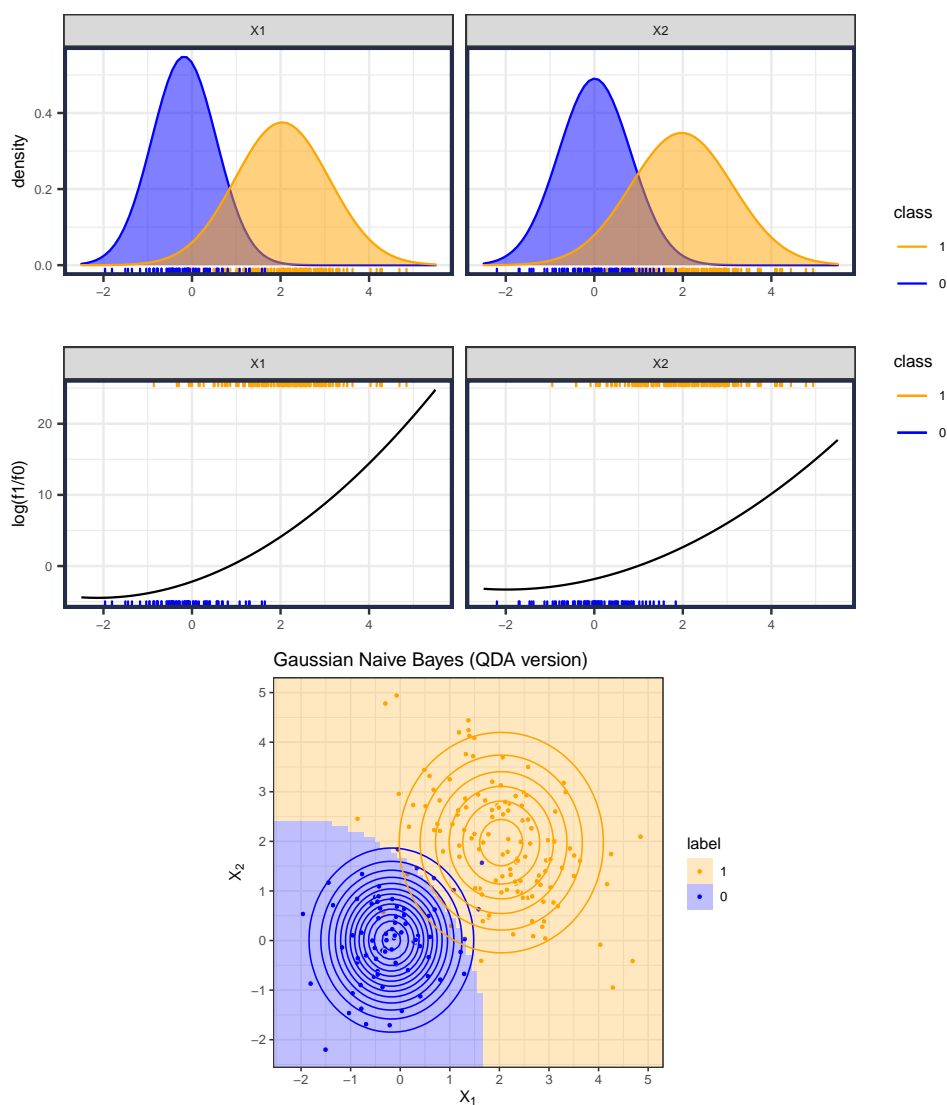
$$\hat{f}_k(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \hat{\mu}_k, \hat{\Sigma}_k)$$

- But when the dimensionality of \mathbf{x} gets large or there is high correlation, estimation of $\hat{\Sigma}_k$ can be poor
- If we force $\hat{\Sigma}_k$ to be *diagonal* then the densities are product of univariate Gaussians (called Gaussian Naive Bayes)

$$\hat{f}_k(\mathbf{x}) = \prod_{j=1}^p \mathcal{N}(x_j; \mu_{kj}, \sigma_{kj})$$

- Even if the data are not independent, this may give better estimates by reducing the variance (at the expense of a bit of bias)
- This is a special case of QDA, where we restrict the off-diagonal terms in the variance-covariance to be 0.

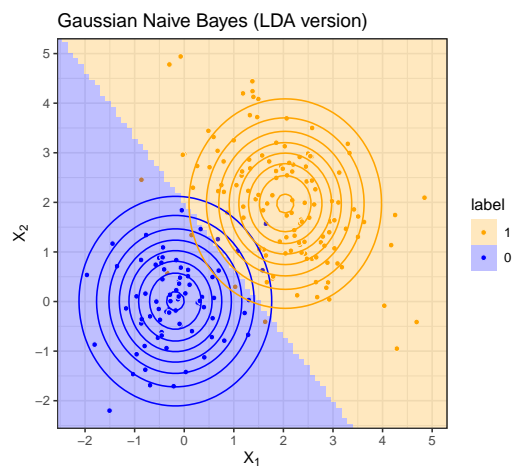
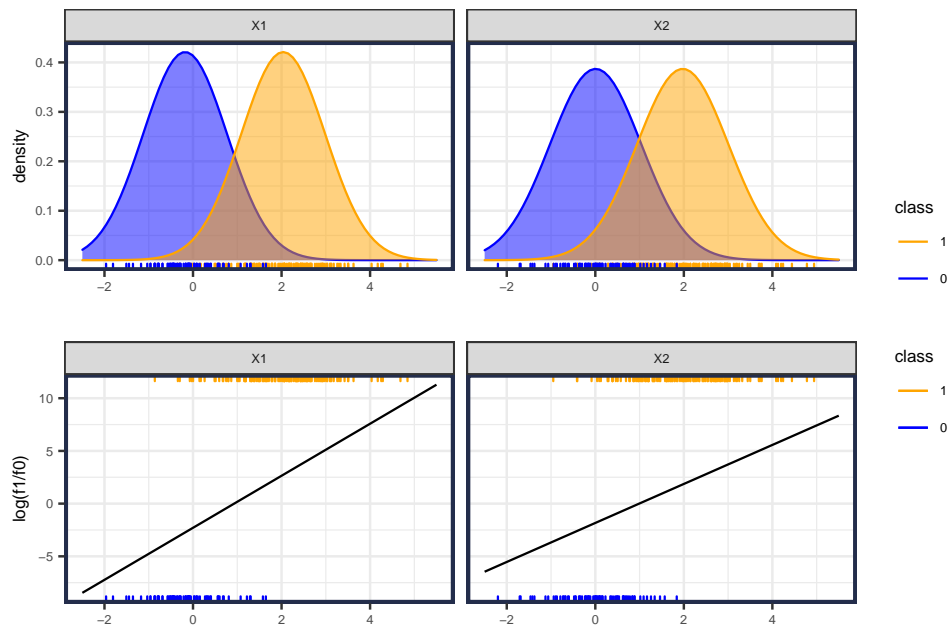
class	predictor	mu	sd	density
0	X1	-0.18	0.73	N(mu = -0.18, sd = 0.73)
0	X2	0.01	0.81	N(mu = 0.01, sd = 0.81)
1	X1	2.04	1.06	N(mu = 2.04, sd = 1.06)
1	X2	1.97	1.15	N(mu = 1.97, sd = 1.15)



- A simpler model (less complexity/edf) forces a common standard deviation for all class (special case of LDA)

$$\hat{f}_k(\mathbf{x}) = \prod_{j=1}^p \mathcal{N}(x_j; \mu_{kj}, \sigma_j)$$

class	predictor	mu	sd
0	X1	-0.18	0.95
0	X2	0.01	1.03
1	X1	2.04	0.95
1	X2	1.97	1.03



5.2 Kernel Naive Bayes

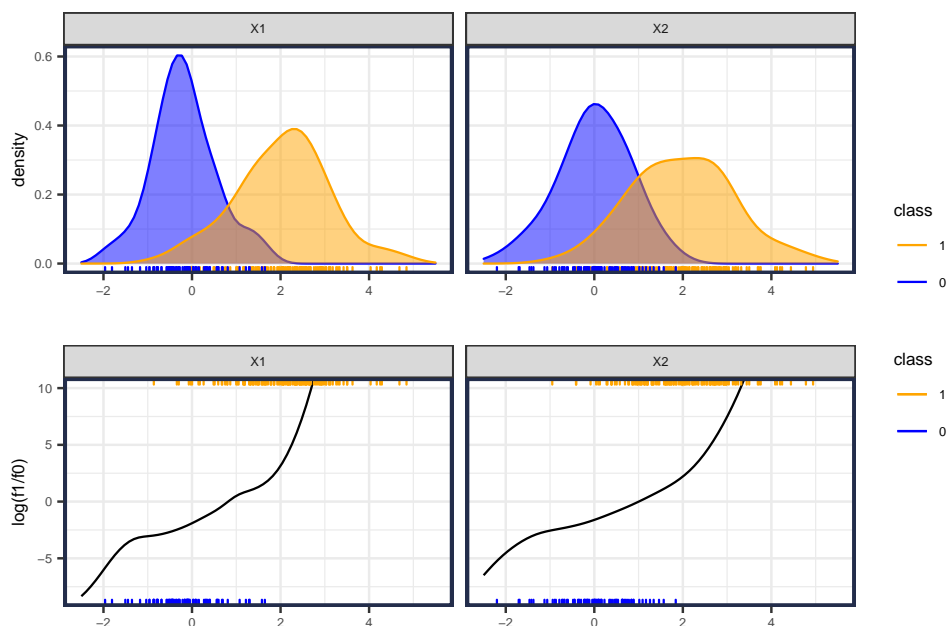
In *kernel density* Naive Bayes, use Kernel Density Estimation (KDE) to estimate each component density:

$$\hat{f}_{kj}(x_j) = \frac{1}{n_k} \sum_{i:g_i=k} K(x_j - x_{ij}; h_{kj})$$

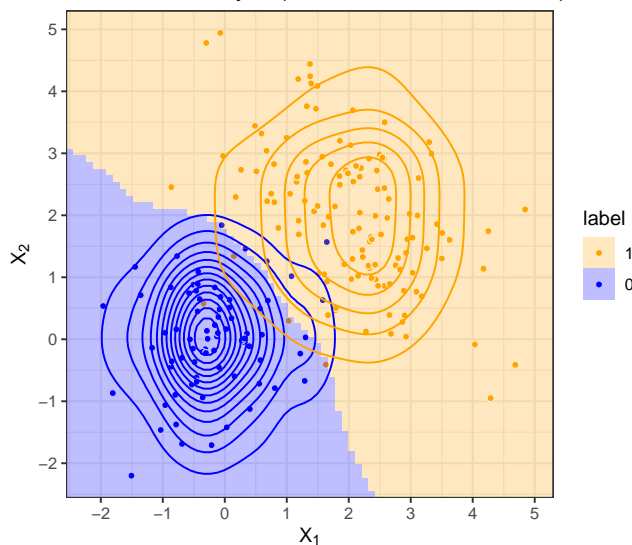
with bandwidth parameter h_{kj} .

The density ratio becomes

$$\frac{\hat{f}_{1j}(x_j)}{\hat{f}_{0j}(x_j)} = \frac{\frac{1}{n_1} \sum_{i:g_i=1} K(x_j - x_{ij}; h_{1j})}{\frac{1}{n_0} \sum_{i:g_i=0} K(x_j - x_{ij}; h_{0j})}$$



Kernel Naive Bayes (different class bandwidths)



- for less complex models, use same bandwidth parameter for each class.

Note: this gives a different solution than using KDE with a *product kernel*! (which is not a naive bayes model)

$$\hat{f}_k(\mathbf{x}) = \frac{1}{n_k} \sum_{i: g_i=k} \prod_{j=1}^p K(x_j - x_{ij}; h_{kj})$$

6 Connections: Generalized Additive Models (GAM)

It turns out that there is a close connection between Logistic Regression, Naive Bayes, and LDA. To help see this, notice that all three methods can be written:

$$\begin{aligned} \gamma(x) &= \log \left(\frac{\pi}{1 - \pi} \right) + \log \left(\frac{f_1(x)}{f_0(x)} \right) \\ &= \alpha_0 + \sum_{j=1}^p \alpha_j S_j \end{aligned}$$

- **Logistic Regression**

$$\hat{\alpha}_0 = \hat{\beta}_0$$

$$\hat{\alpha}_j = \hat{\beta}_j$$

$$\hat{S}_j = x_j$$

- **LDA**

$$\hat{\alpha}_0 = \log \frac{\hat{\pi}}{1 - \hat{\pi}} - \frac{1}{2} (\hat{\mu}_1 + \hat{\mu}_0)^T \hat{\Sigma}^{-1} (\hat{\mu}_1 - \hat{\mu}_0)$$

$$\hat{\alpha}_j = \hat{\Sigma}^{-1} (\hat{\mu}_1 - \hat{\mu}_0)$$

$$\hat{S}_j = x_j$$

- **Naive Bayes**

$$\hat{\alpha}_0 = \log \frac{\hat{\pi}}{1 - \hat{\pi}}$$

$$\hat{\alpha}_j = 1$$

$$\hat{S}_j = \log \frac{\hat{f}_{1j}(x_j)}{\hat{f}_{0j}(x_j)}$$

- **Generalized Additive Models (GAM)**

- GAM models are made to directly estimate models of this form.

$$\hat{\gamma}(x) = \hat{\alpha} + \sum_{j=1}^p \hat{g}_j(x_j)$$

- $g_j(x_j)$ is non-linear (usually based on penalized splines)
- In **R**, the `mgcv` package is worth becoming familiar with to implement GAM.
- See ESL 9.1 for more details