CAP 5625: Computational Foundations for Artificial Intelligence

Discriminant analysis

Our second statistical (machine) learning method

In this lecture, we will introduce the linear regression classifier, Bayes classifier, and discriminant analysis.

Notation

Input X: X is often multidimensional. Each dimension of X is referred to as a feature, predictor, or independent variable

Output Y: The response, or dependent variable

Categorization

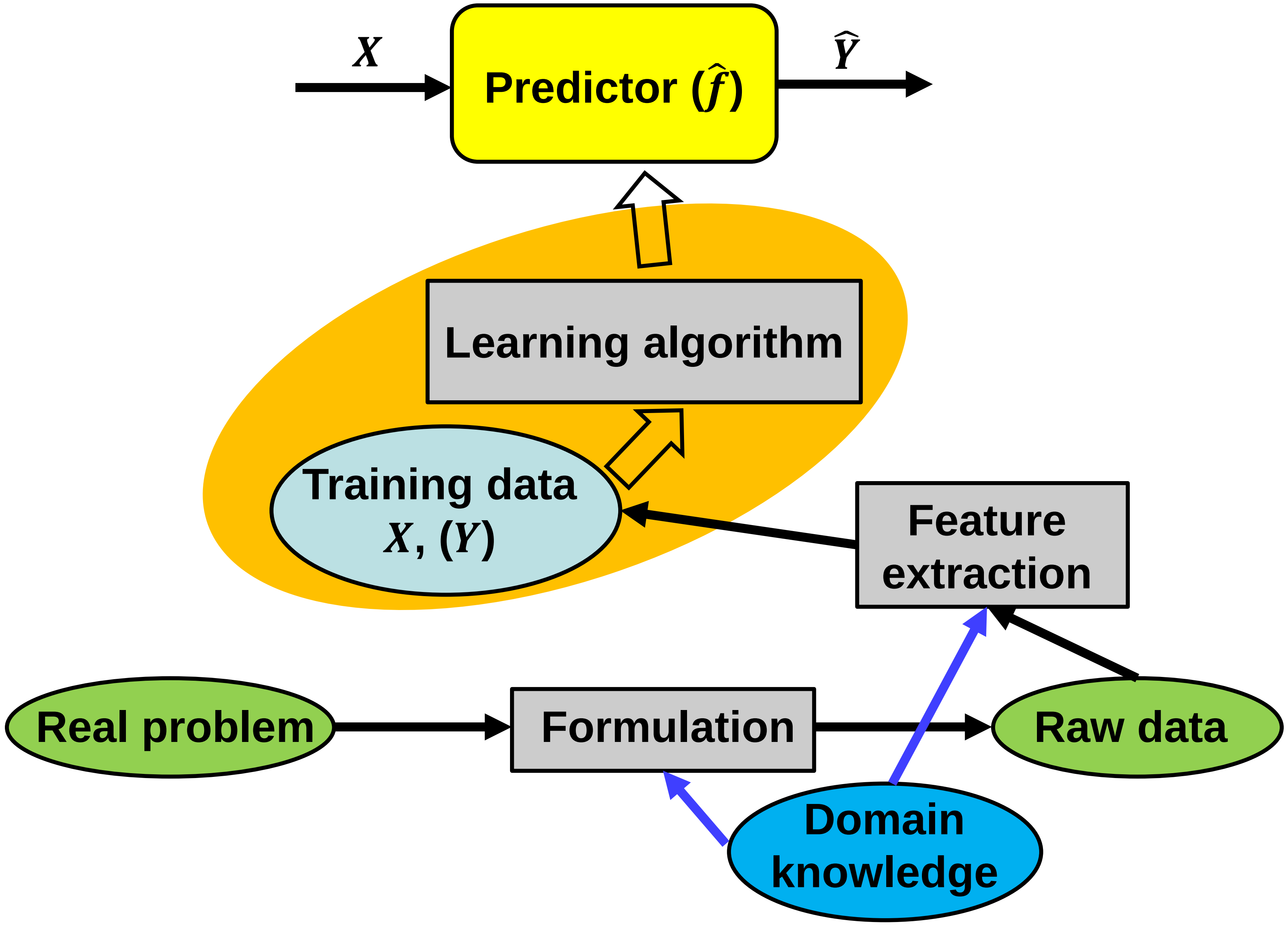
Supervised learning vs. unsupervised learning

Is Y available in the training data?

Regression vs. classification

Is Y quantitative or qualitative?

Statistical learning is simply function (f) estimation



Classification problems and qualitative outputs

So far we have considered regression problems, in which the response Y was quantitative.

However, what if the response Y was qualitative?

For example, maybe the response is categorical with $K = 2$ (dichotomous) categories representing yes/no, true/false, win/lose, male/female.

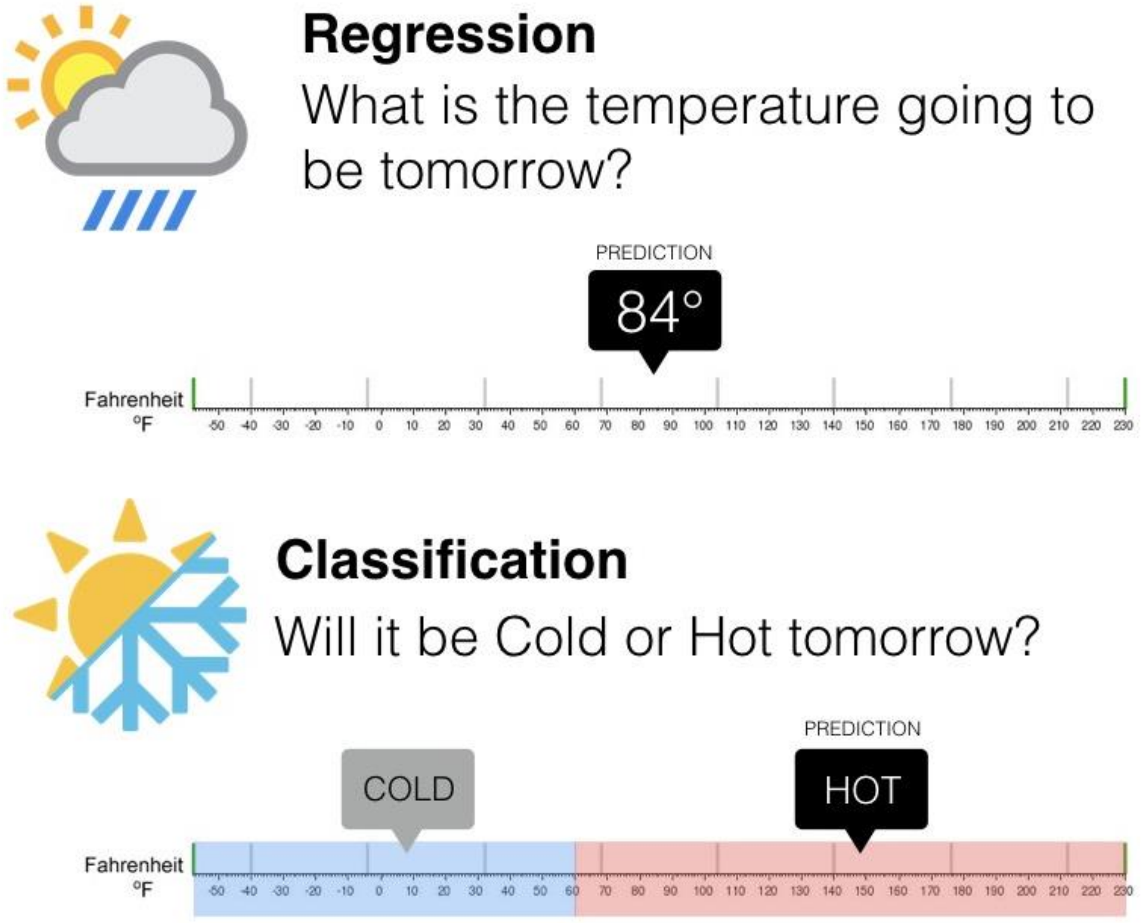
Or maybe it is a categorical variable with $K > 2$ categories, such as small/medium/large or cold/cool/warm/hot.

Or potentially we have K categories that are numeric ($e.g.$, 1, 2, 3, 4, and 5), but these values are arbitrary.

Difference between regression and classification

Regression: Response Y is **quantitative** (**numerical**), and so predications are numbers.

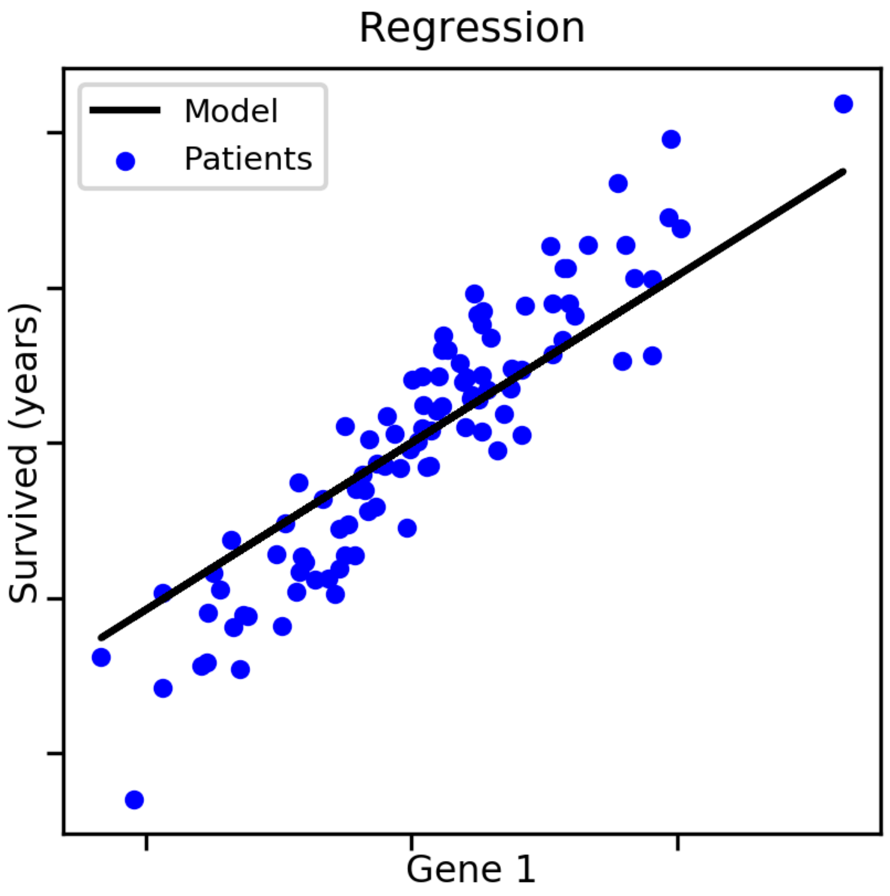
Classification: Response Y is **qualitative** (**categorical**), and so predictions are classes (which could be represented as numbers).

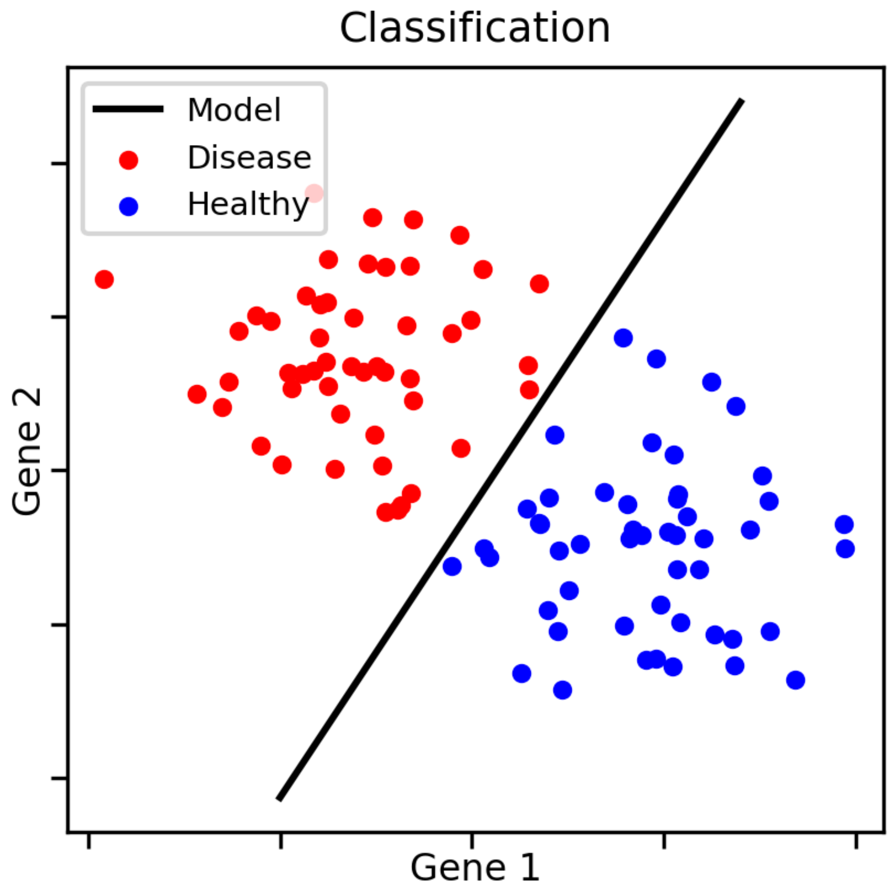


Visualizing the difference with regression

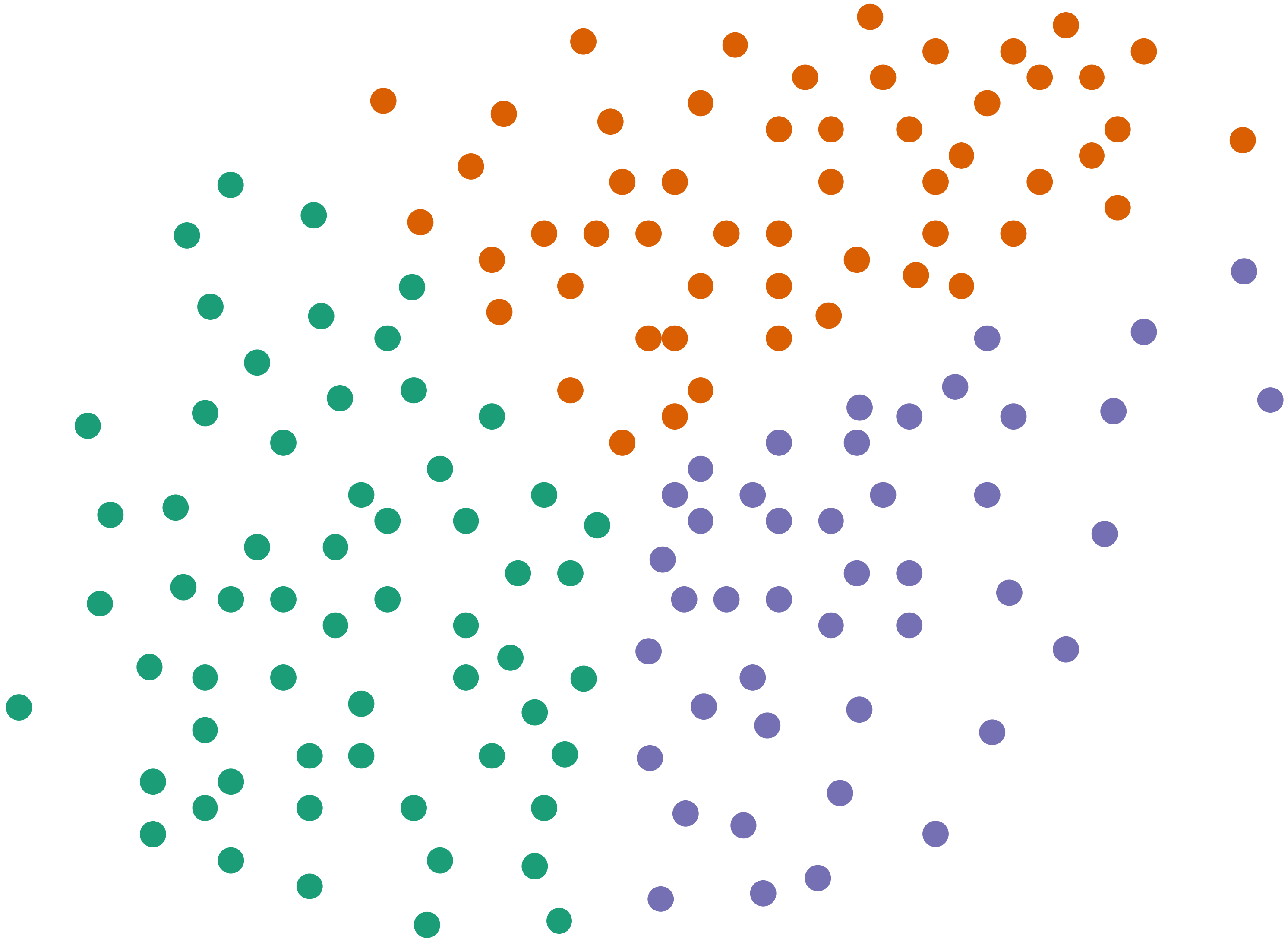
Regression: Predict a **quantitative** response by **fitting** the data.

Classification: Predict a **qualitative** response by **splitting** the data.

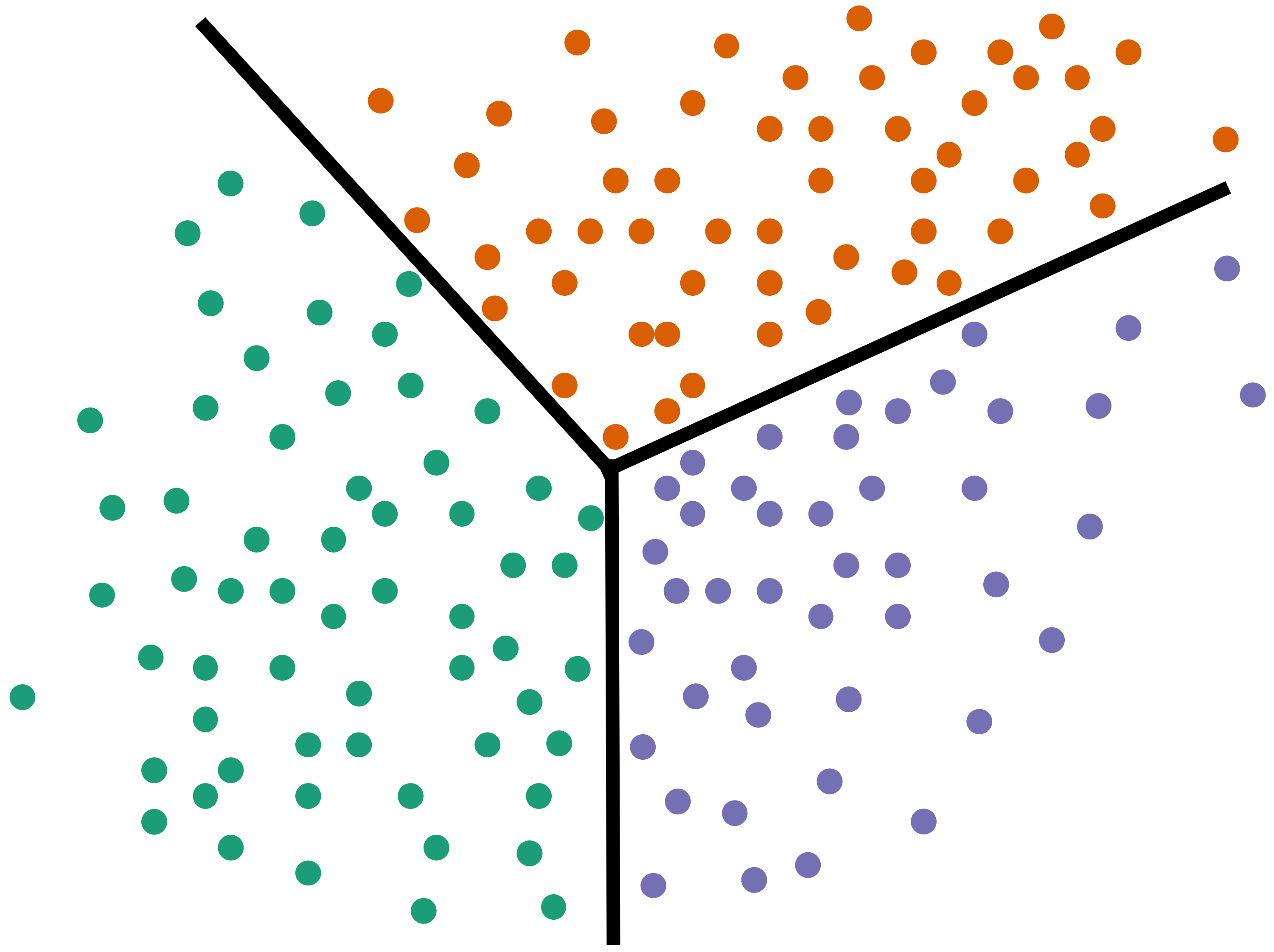




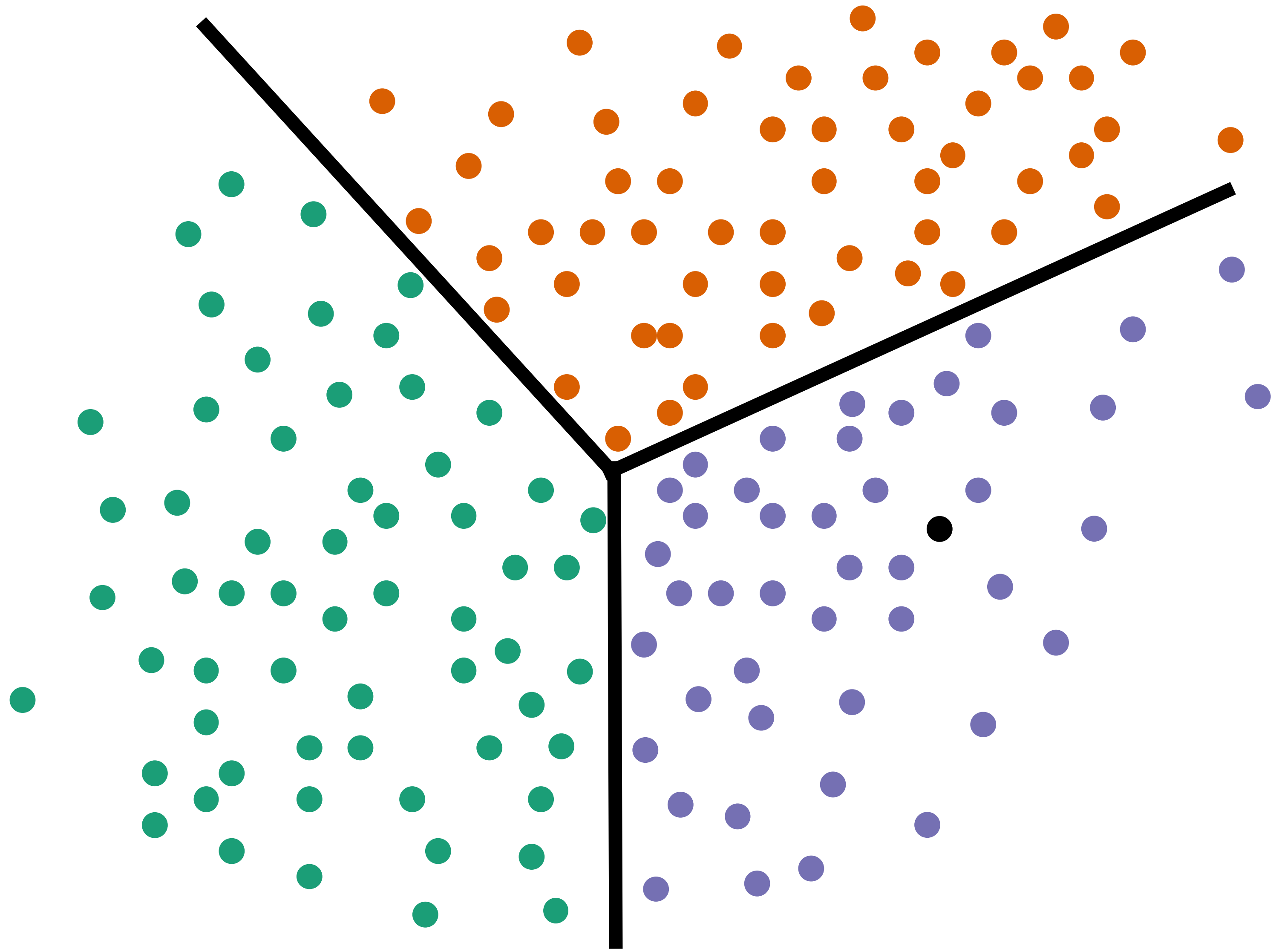
Training data of different class labels (colors)



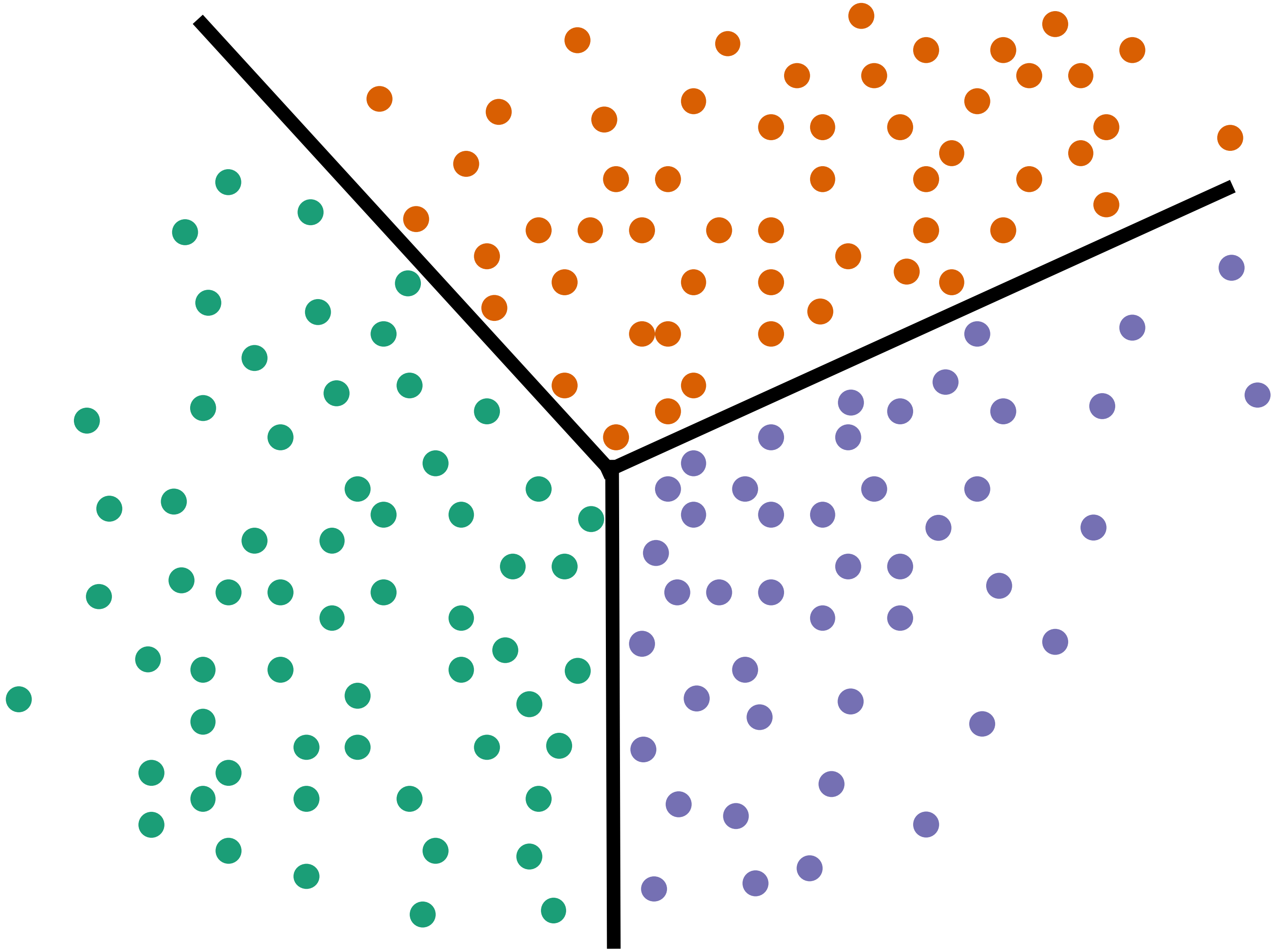
Identify decision boundaries between classes



Classify a test observation (black)



Test observation is likely purple



Example classification problem

Example: Handwritten digit recognition

Goal: identify the digits ($0, 1, \ldots, 9$) associated with handwritten numbers ($K = 10$ classes).

Raw data: images that are scaled segments from five digit ZIP codes.

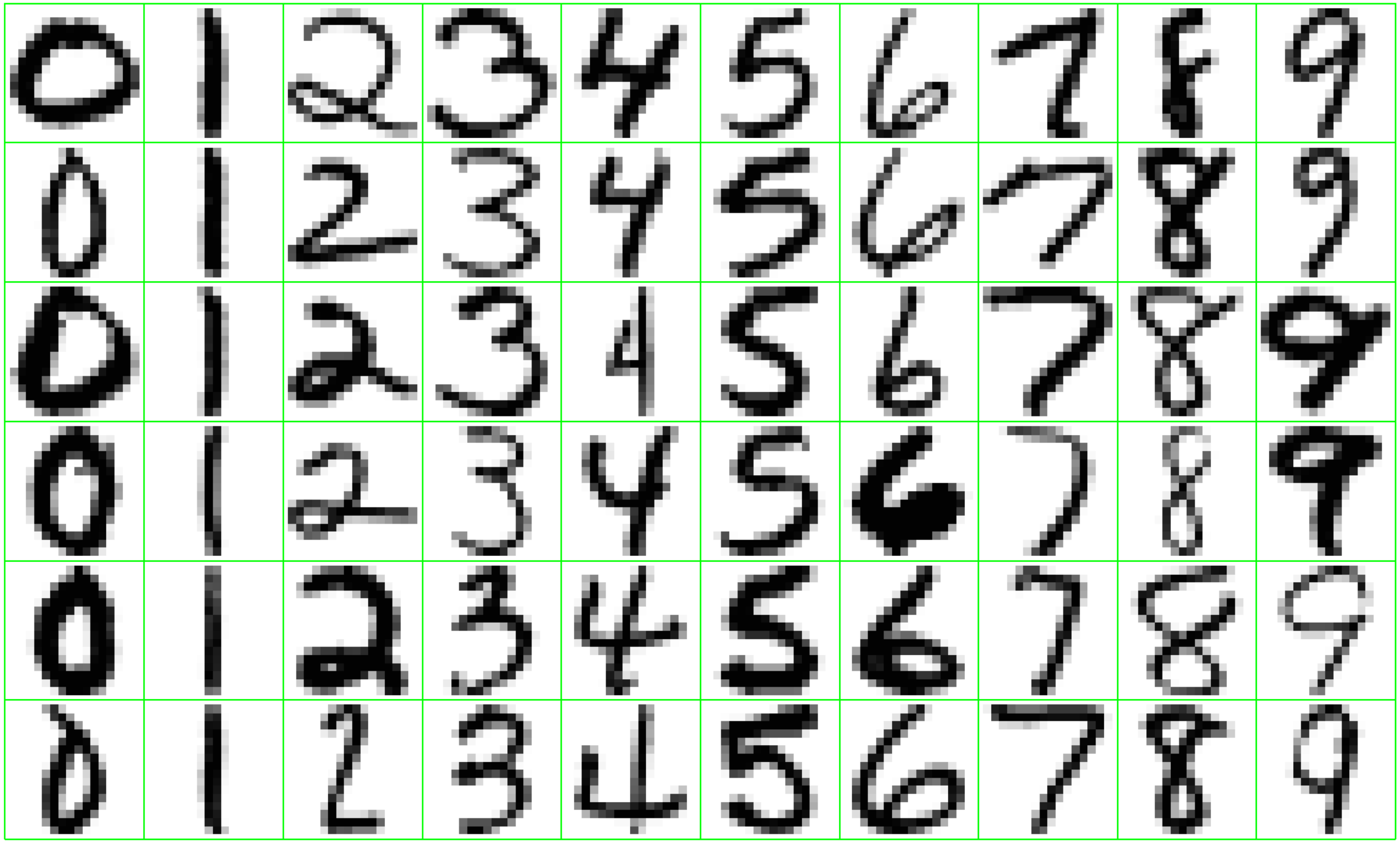
16×16 eight-bit grayscale maps

Pixel intensities range from 0 (black) to 1 (white)

Input data: a $p = 256$-dimensional vector, or feature vectors with lower dimensions.

This is a supervised learning classification problem

Identify the digits ($0, 1, \ldots, 9$) associated with handwritten numbers



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Using linear regression to predict binary response

Suppose we have a dataset with p features $X = [X_1, X_2, \ldots, X_p]$.

Consider we wish to predict a binary (*e.g.*, yes or no, true or false, 0 or 1) response Y.

Let's code one class as $Y = 0$ and the other class as $Y = 1$.

To predict the class, we can perform linear regression of the form

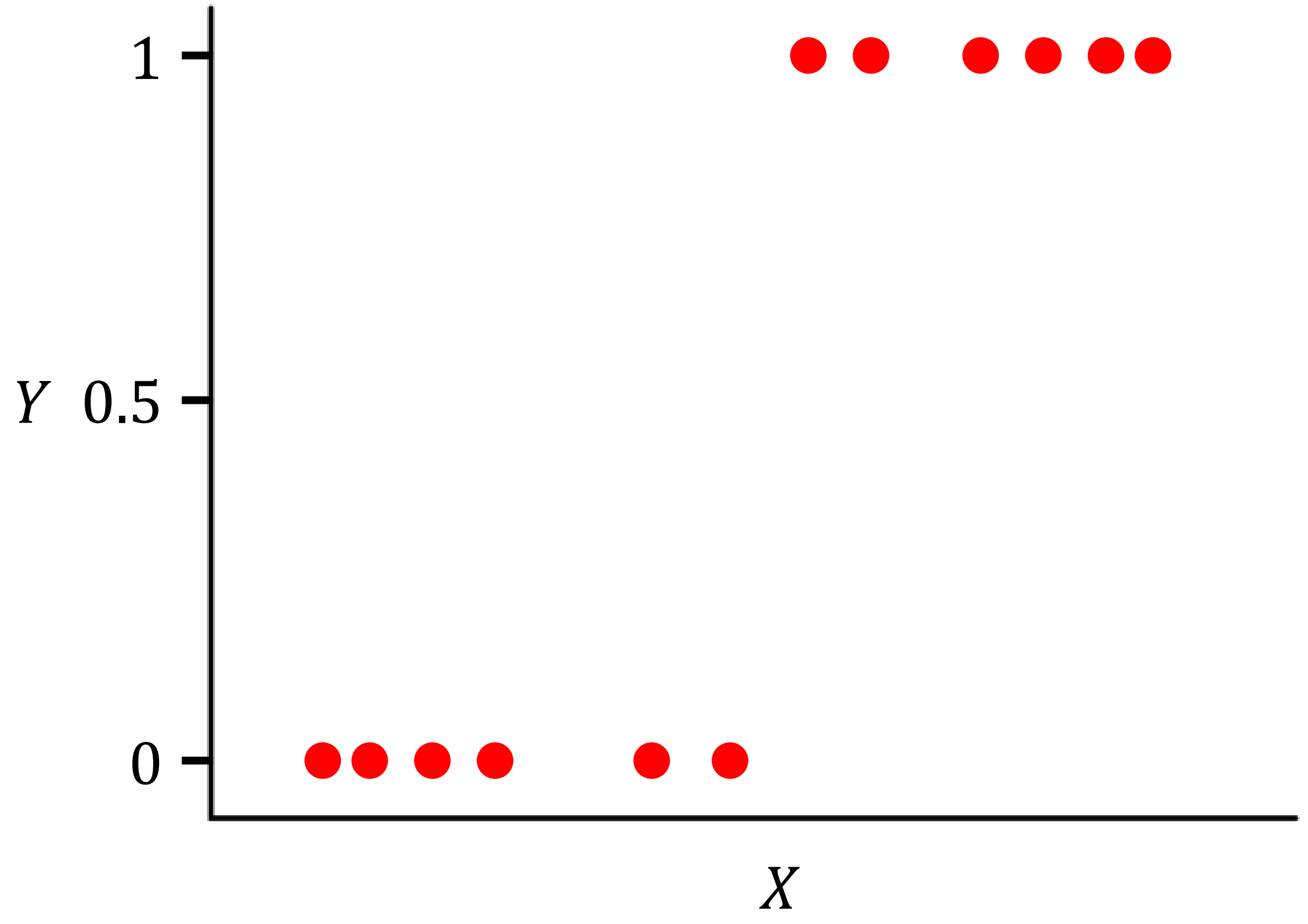
$$Y = f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

We choose the first class if $\hat{Y} < 0.5$ and the second class if $\hat{Y} > 0.5$, with $\hat{Y} = 0.5$ being the **decision boundary**.

Using linear regression to predict binary response

Assuming we have a single feature ($p = 1$) denoted by X, let's examine what classification with linear regression would look like.

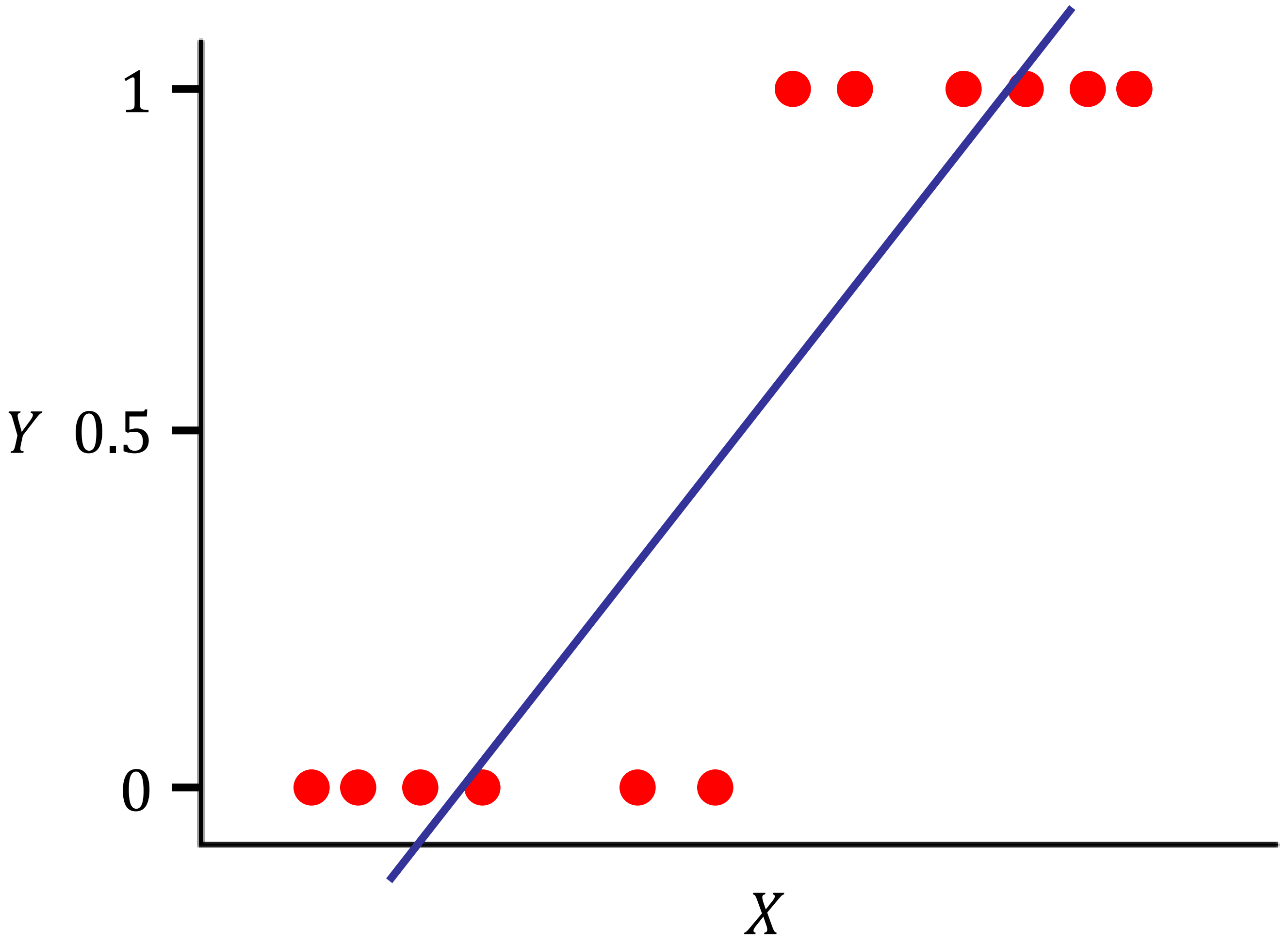
$$Y = \beta_0 + \beta_1 X$$



Using linear regression to predict binary response

Assuming we have a single feature ($p = 1$) denoted by X, let's examine what classification with linear regression would look like.

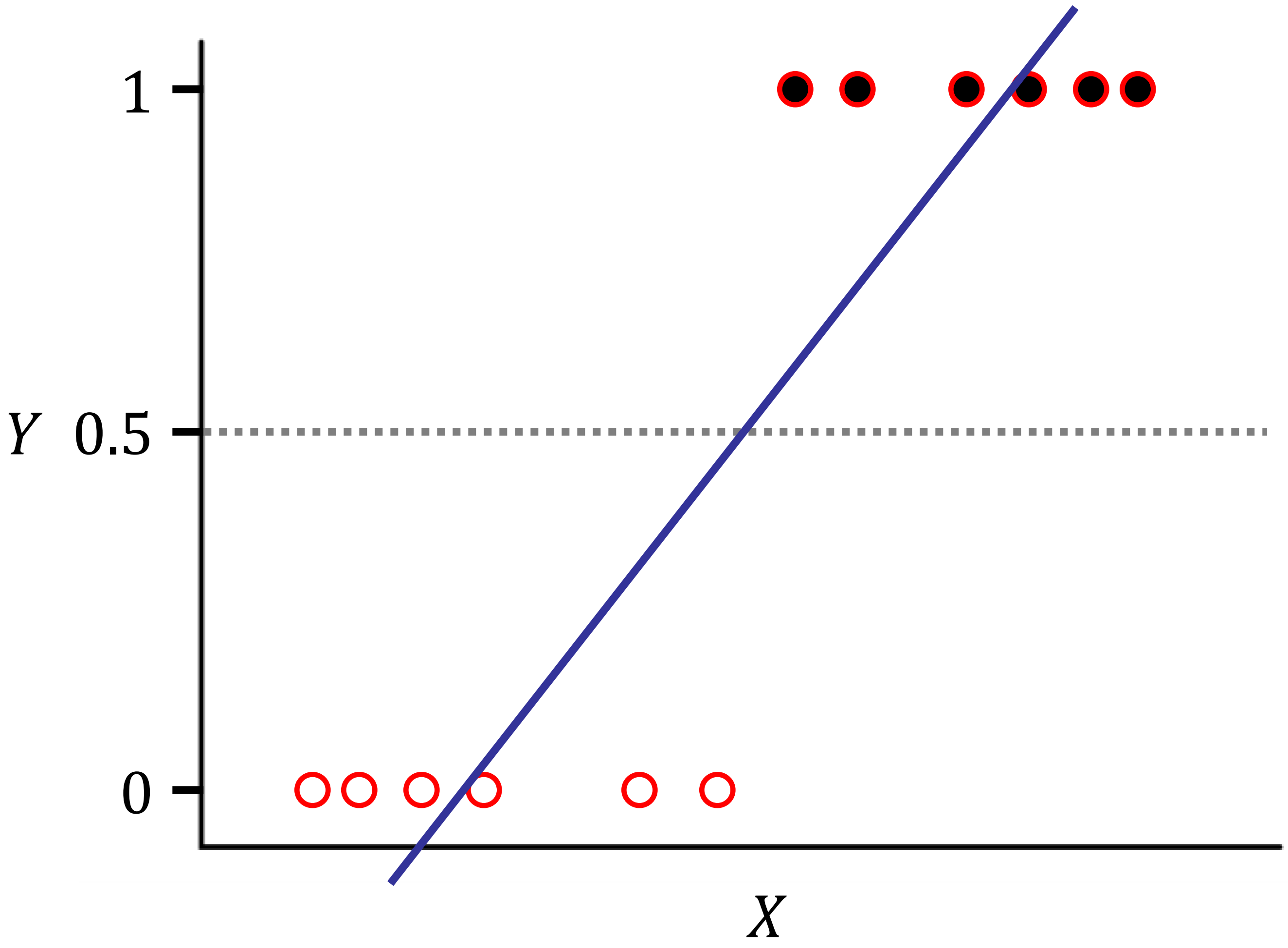
$$Y = \beta_0 + \beta_1 X$$



Using linear regression to predict binary response

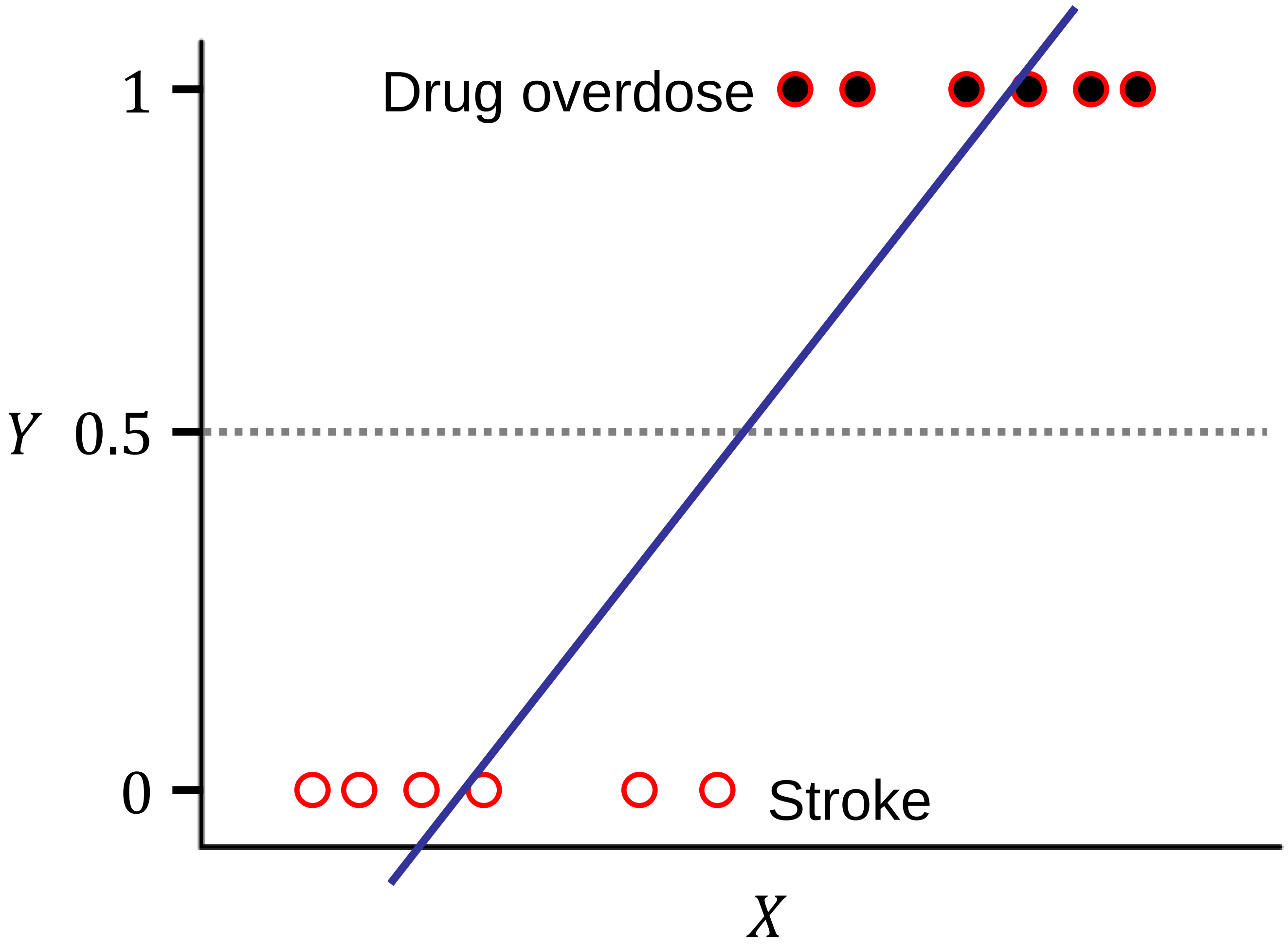
Assuming we have a single feature ($p = 1$) denoted by X, let's examine what classification with linear regression would look like.

$$Y = \beta_0 + \beta_1 X$$



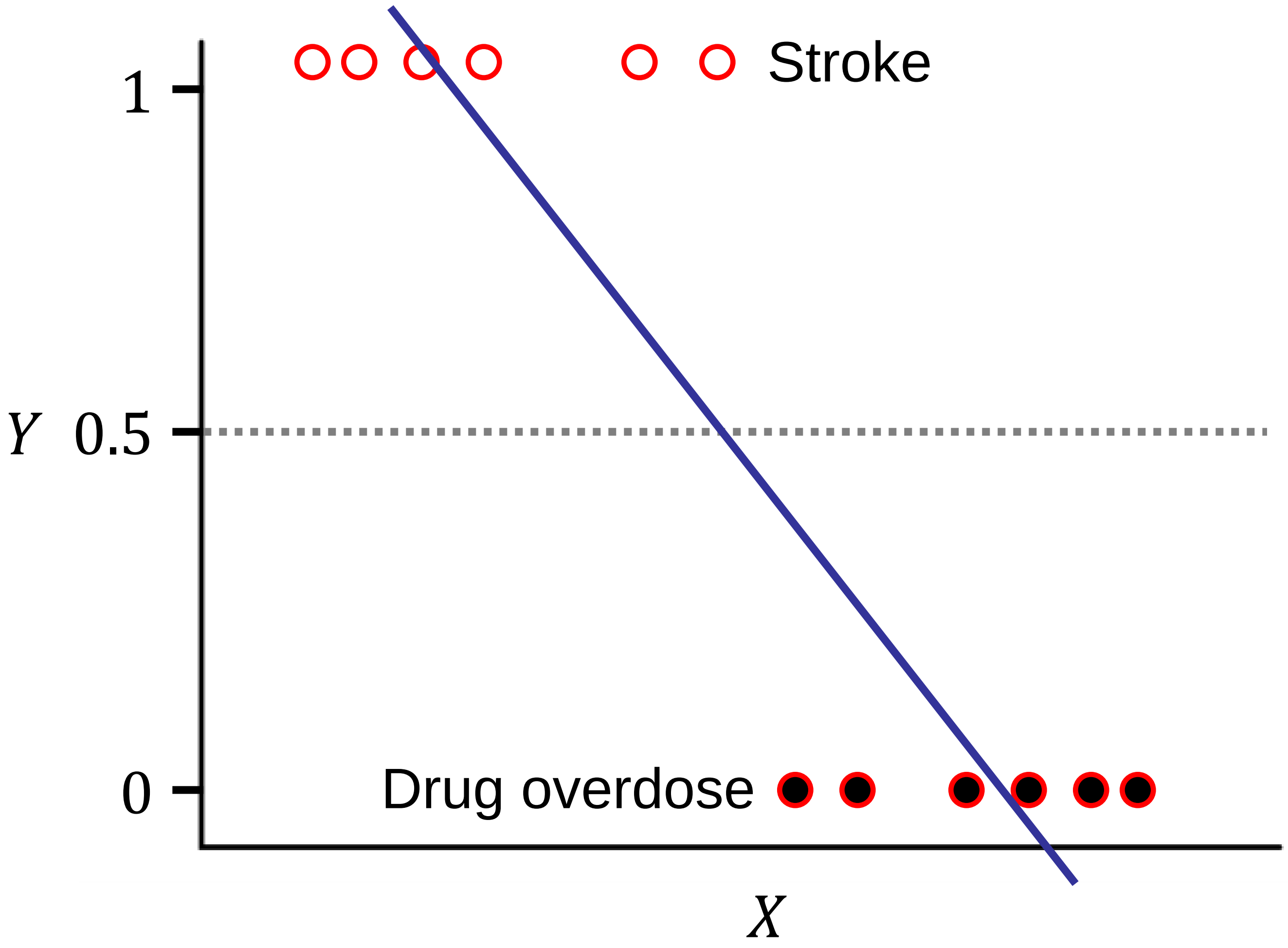
Using linear regression to predict binary response

Supposing the coded Y represents a condition taking on values of "stroke" ($Y = 0$) and "drug overdose" ($Y = 1$), observations where $\hat{Y} > 0.5$ would be assigned to "drug overdose", and to "stroke" otherwise.



Changing coding does not alter classification

If we instead change the coding with “stroke” ($Y = 1$) and “drug overdose” ($Y = 0$), then observations where $\hat{Y} > 0.5$ would be assigned to “stroke”, and to “drug overdose” otherwise.



Linear regression to predict > 2 category response

Suppose we have a dataset with p features $X = [X_1, X_2, \dots, X_p]$.

Consider predicting a categorical response Y with three categories (*e*.*g*., “stroke”, “drug overdose”, or “epileptic seizure”).

Let’s code one class as $Y = 0$, another class as $Y = 1$, and the final class as $Y = 2$, with for example response coded as

$$Y = \begin{cases} 0 & \text{if stroke} \\ 1 & \text{if drug overdose} \\ 2 & \text{if epileptic seizure} \end{cases}$$

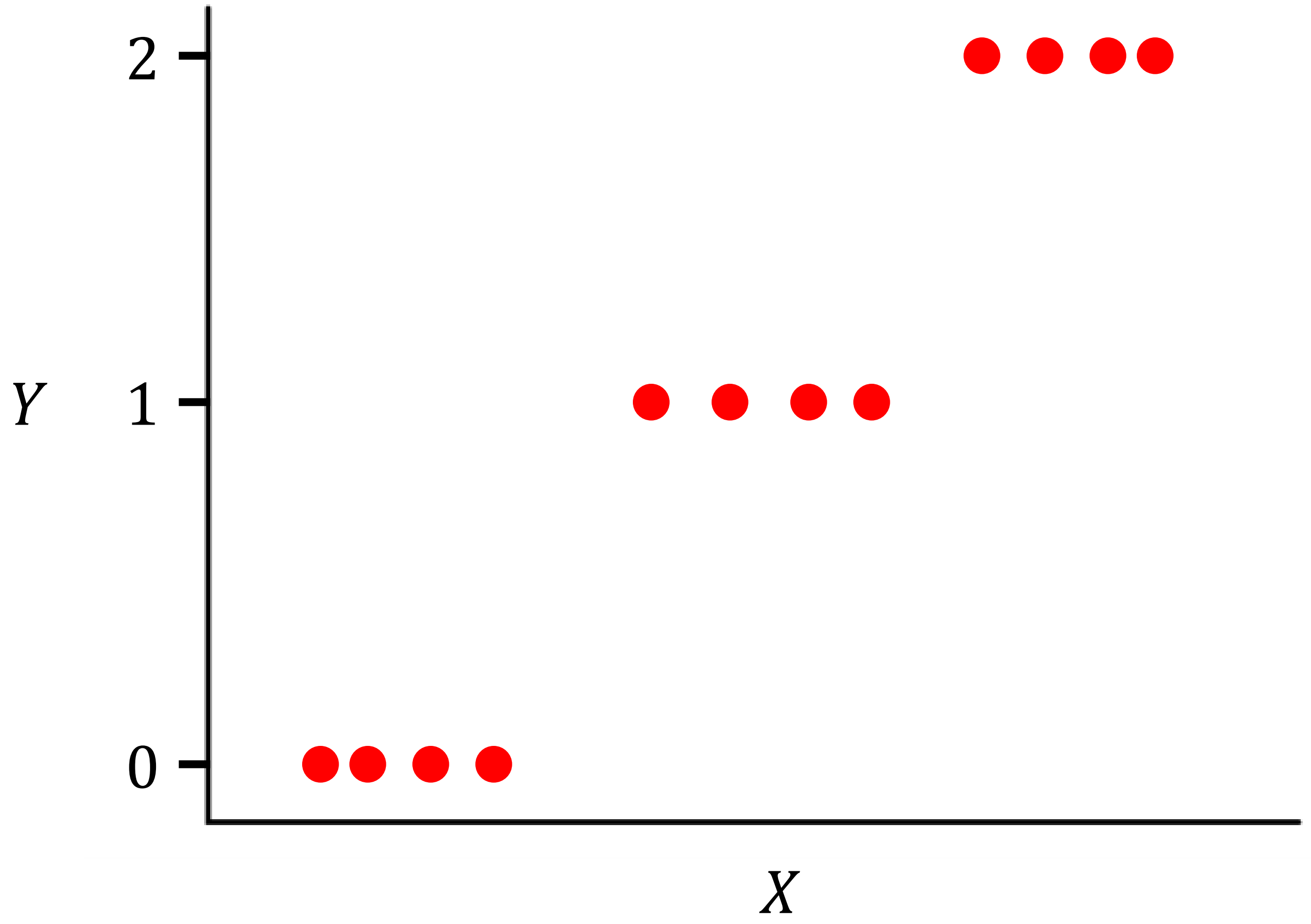
To predict the class, we can perform linear regression of form

$$Y = f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Linear regression to predict > 2 category response

Assuming we have a single feature ($p = 1$) denoted by X, let's examine what classification with linear regression would look like.

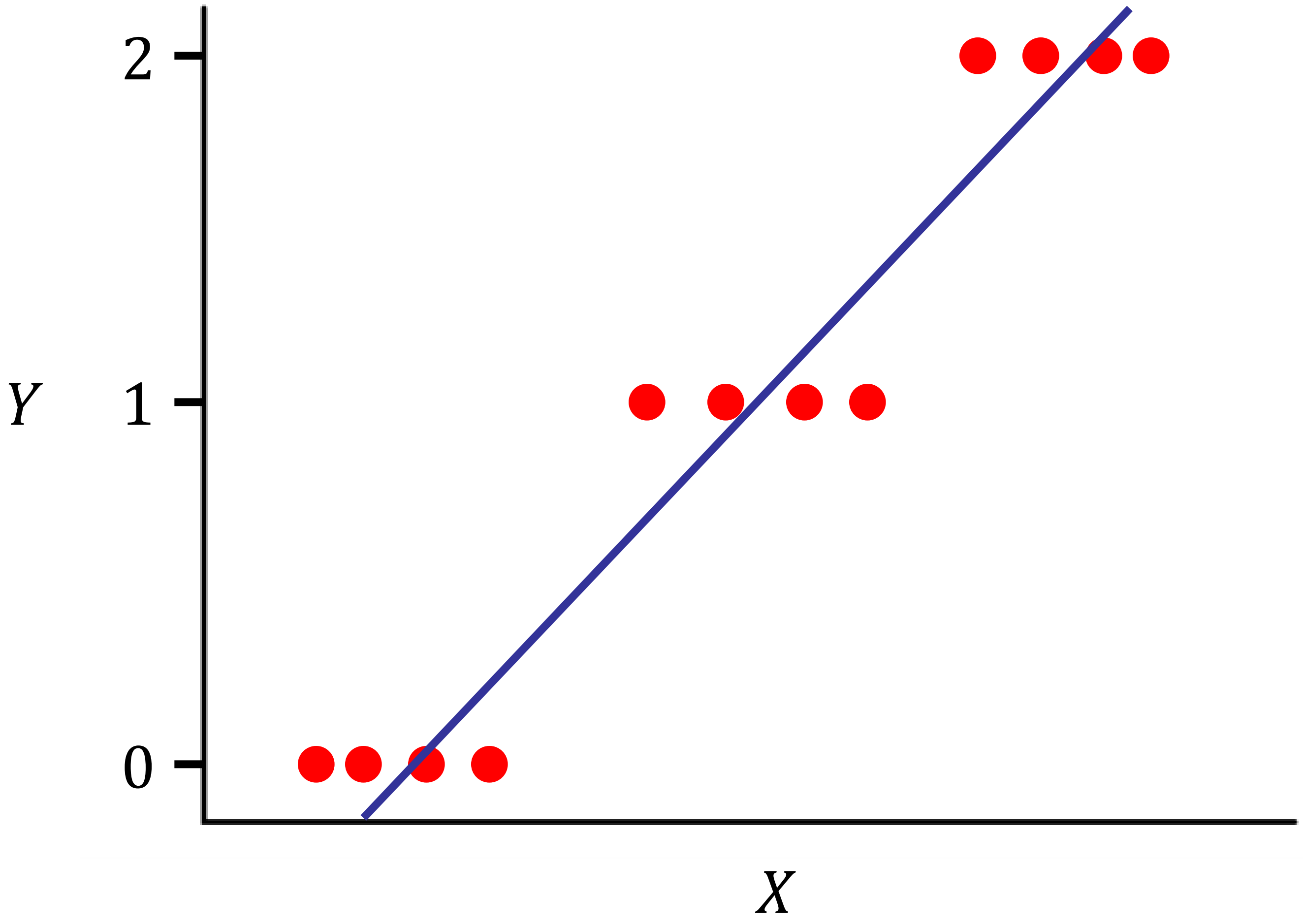
$$Y = \beta_0 + \beta_1 X$$



Linear regression to predict > 2 category response

Assuming we have a single feature ($p = 1$) denoted by X, let's examine what classification with linear regression would look like.

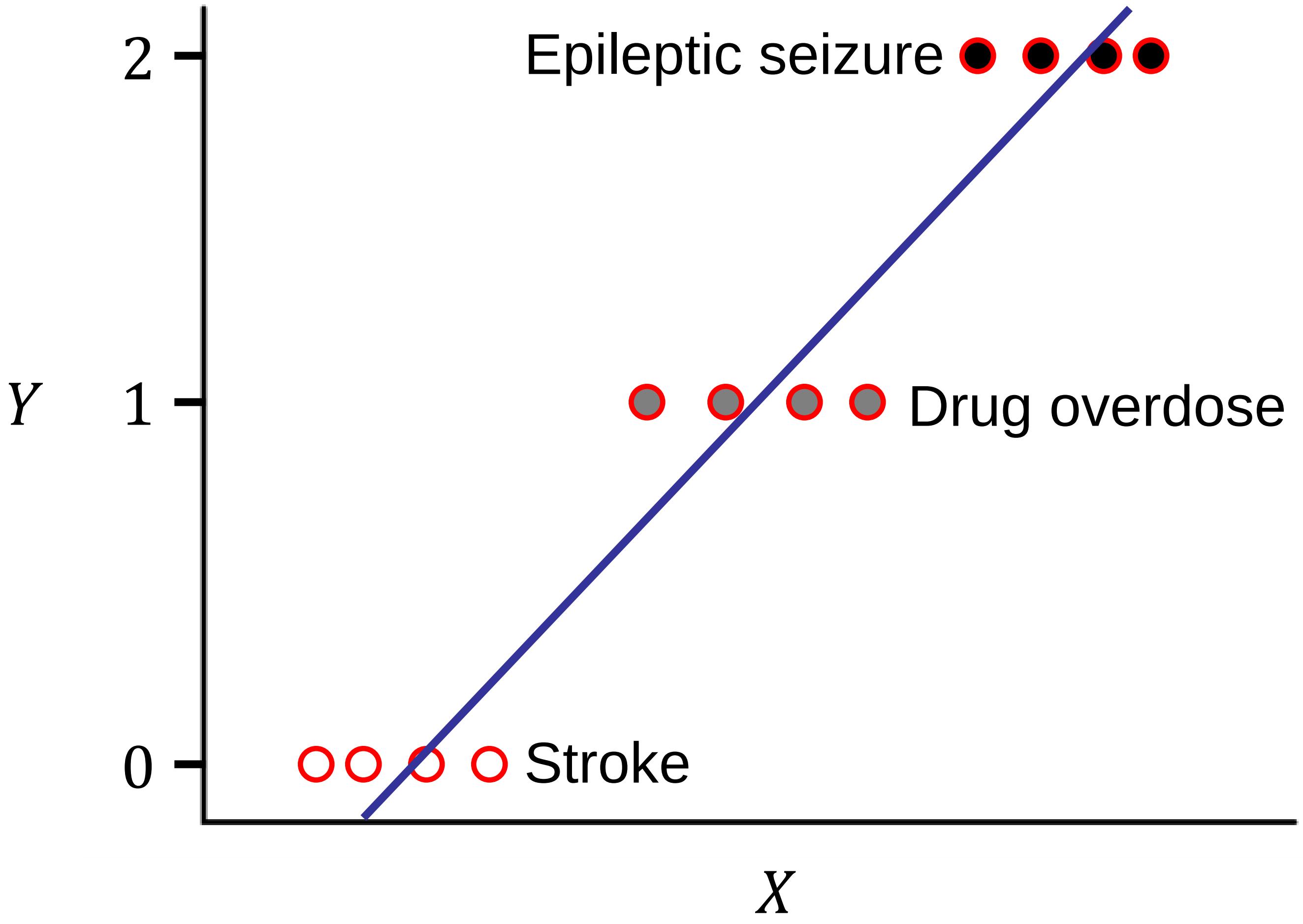
$$Y = \beta_0 + \beta_1 X$$



Linear regression to predict > 2 category response

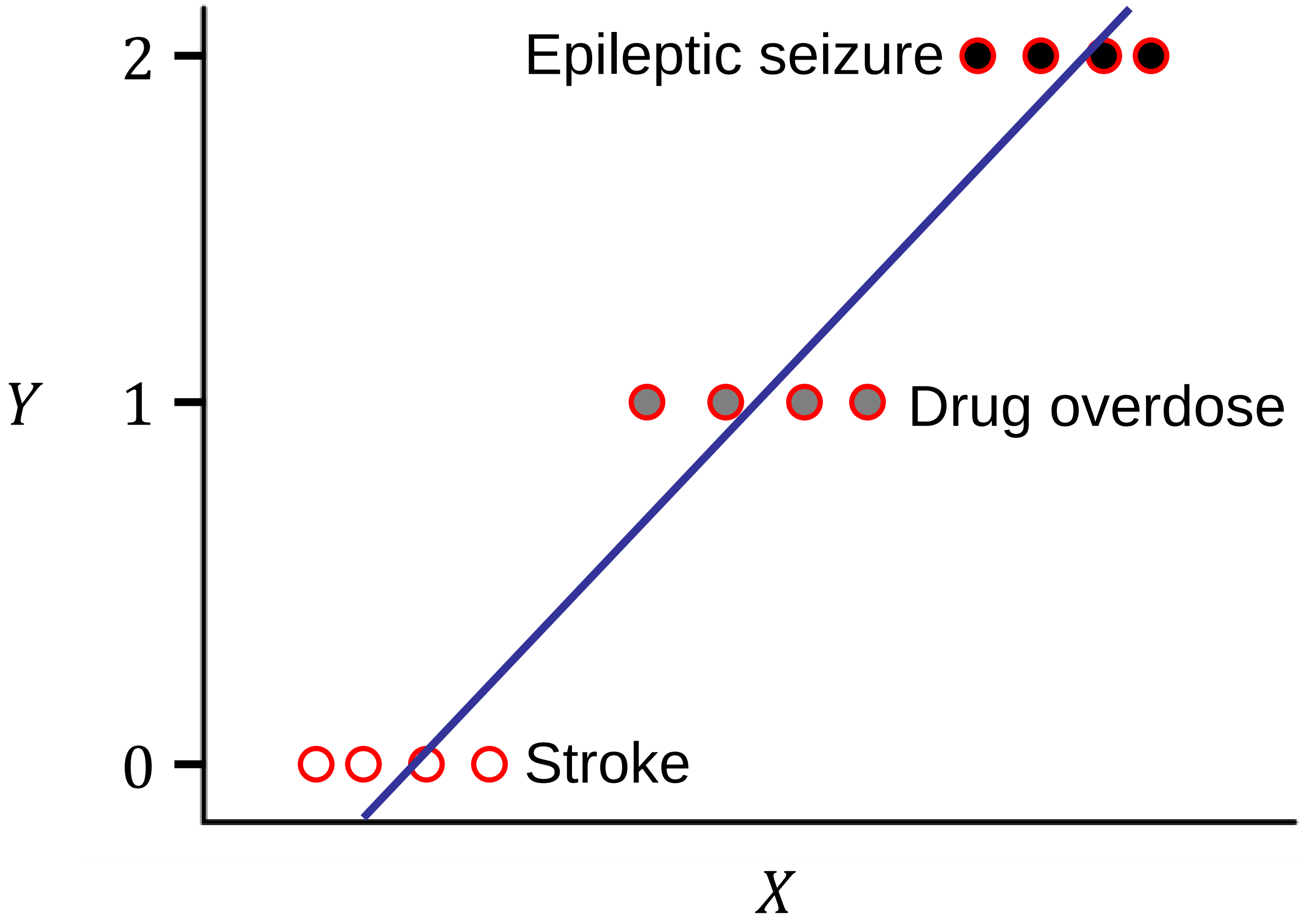
Assuming we have a single feature ($p = 1$) denoted by X, let's examine what classification with linear regression would look like.

$$Y = \beta_0 + \beta_1 X$$



Linear regression to predict > 2 category response

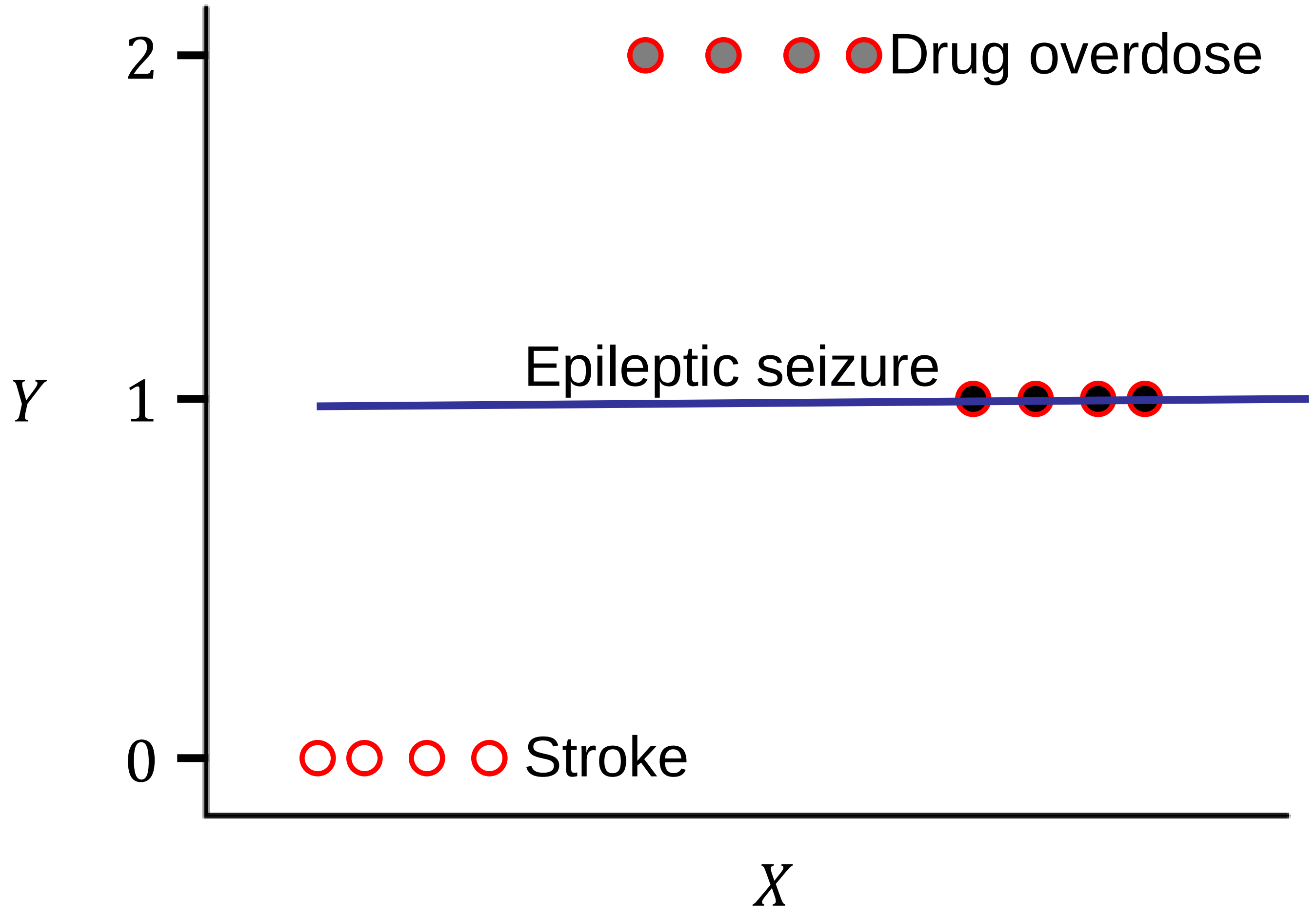
But why is this coding better than an alternative coding?



Linear regression to predict > 2 category response

But why is this coding better than an alternative coding?

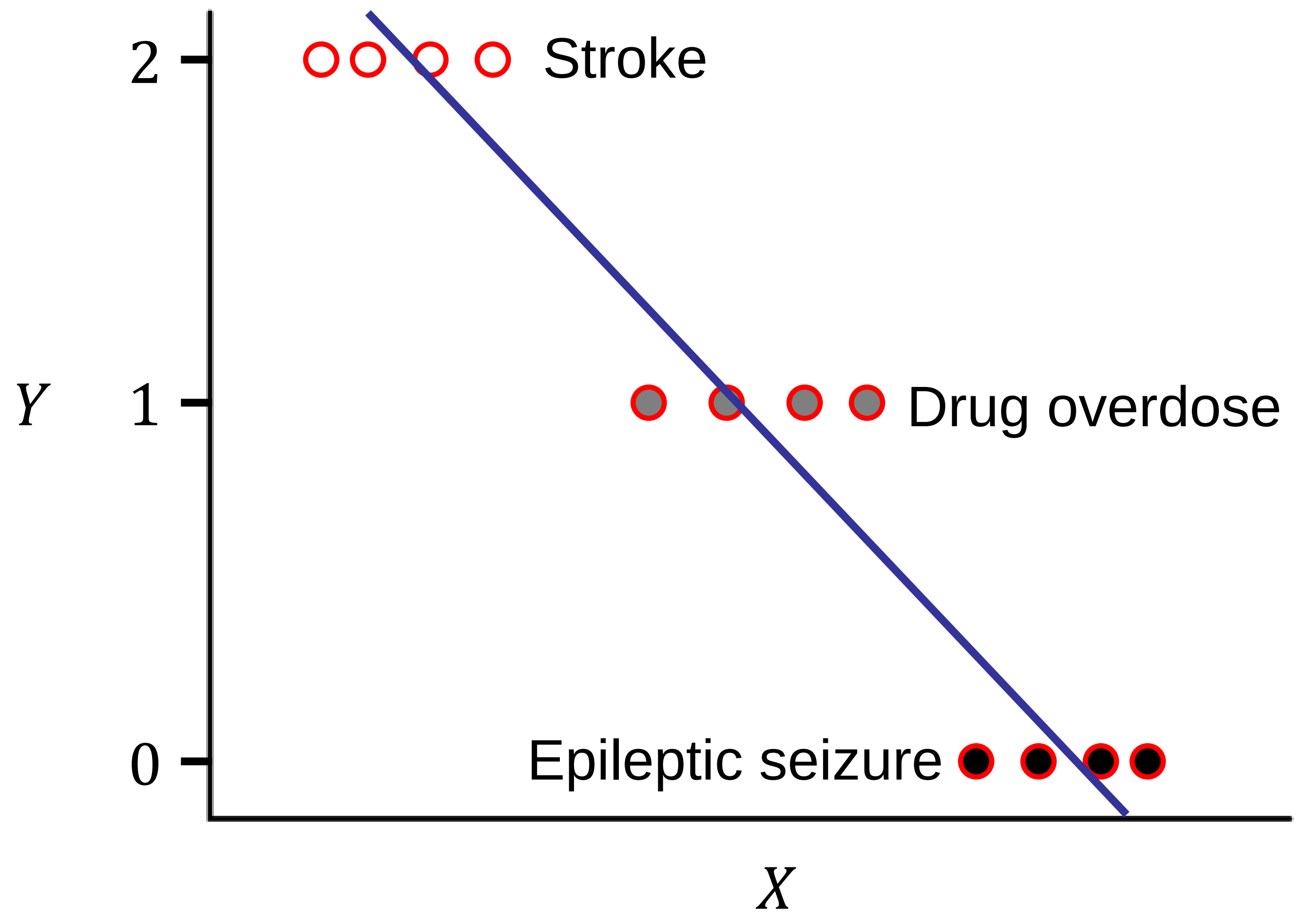
Such as this coding.



Linear regression to predict > 2 category response

But why is this coding better than an alternative coding?

Or this one?



How to use linear regression with >2 categories?

Recall when we discussed using qualitative features, we said that we could break a qualitative feature into multiple binary features called dummy variables.

The same can be done here, and to illustrate, consider again the original coding for three categories (classes)

$$Y = \begin{cases} 0 & \text{if stroke} \\ 1 & \text{if drug overdose} \\ 2 & \text{if epileptic seizure} \end{cases}$$

and performing linear regression of the form

$$Y = f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

For three classes, we can perform three separate linear regressions instead.

How to use linear regression with > 2 categories?

$$Y = \begin{cases} 0 & \text{if stroke} \\ 1 & \text{if drug overdose} \\ 2 & \text{if epileptic seizure} \end{cases}$$

$$Y = f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Instead of the above coding, let the first “stroke” class be associated with response Y_1, the second “drug overdose” class be associated with Y_2, and the third “epileptic seizure” class be associated with Y_3.

That is, we have a separate binary response for each class of form

$$Y_1 = \begin{cases} 1 & \text{stroke} \\ 0 & \text{not stroke} \end{cases} \quad Y_2 = \begin{cases} 1 & \text{overdose} \\ 0 & \text{not overdose} \end{cases} \quad Y_3 = \begin{cases} 1 & \text{seizure} \\ 0 & \text{not seizure} \end{cases}$$

How to use linear regression with > 2 categories?

$$Y_1 = \begin{cases} 1 & \text{stroke} \\ 0 & \text{not stroke} \end{cases} \quad Y_2 = \begin{cases} 1 & \text{overdose} \\ 0 & \text{not overdose} \end{cases} \quad Y_3 = \begin{cases} 1 & \text{seizure} \\ 0 & \text{not seizure} \end{cases}$$

For each class, we will have a set of $p + 1$ regression coefficients since we are fitting a separate regression model.

That is, for the first “stroke” class, we will have coefficients

$$\beta_1 = [\beta_{10}, \beta_{11}, \beta_{12}, \dots, \beta_{1p}]$$

for the second “drug overdose” class, we will have coefficients

$$\beta_2 = [\beta_{20}, \beta_{21}, \beta_{22}, \dots, \beta_{2p}]$$

and for the third “epileptic seizure” class, we will have coefficients

$$\beta_3 = [\beta_{30}, \beta_{31}, \beta_{32}, \dots, \beta_{3p}]$$

How to use linear regression with > 2 categories?

$$Y_1 = \begin{cases} 1 & \text{stroke} \\ 0 & \text{not stroke} \end{cases} \quad Y_2 = \begin{cases} 1 & \text{overdose} \\ 0 & \text{not overdose} \end{cases} \quad Y_3 = \begin{cases} 1 & \text{seizure} \\ 0 & \text{not seizure} \end{cases}$$

$$\beta_1 = [\beta_{10}, \beta_{11}, \beta_{12}, \ldots, \beta_{1p}]$$

$$\beta_2 = [\beta_{20}, \beta_{21}, \beta_{22}, \ldots, \beta_{2p}]$$

$$\beta_3 = [\beta_{30}, \beta_{31}, \beta_{32}, \ldots, \beta_{3p}]$$

We then perform three separate regressions of the form

$$Y_1 = f_1(X) = \beta_{10} + \beta_{11}X_1 + \beta_{12}X_2 + \cdots + \beta_{1p}X_p$$

$$Y_2 = f_2(X) = \beta_{20} + \beta_{21}X_1 + \beta_{22}X_2 + \cdots + \beta_{2p}X_p$$

$$Y_3 = f_3(X) = \beta_{30} + \beta_{31}X_1 + \beta_{32}X_2 + \cdots + \beta_{3p}X_p$$

Choose class 1 if $\hat{Y}_1$ is largest, 2 if $\hat{Y}_2$ is largest, and 3 if $\hat{Y}_3$ is largest.

General setup for using linear regression with K classes

In general, suppose we have $X = [X_1, X_2, \dots, X_p]$ features and K classes.

We first create K dummy variable responses, so that the response for class k, $k = 1,2,\dots,K$, is

$$Y_k = \begin{cases} 1 & \text{if class } k \\ 0 & \text{if not class } k \end{cases}$$

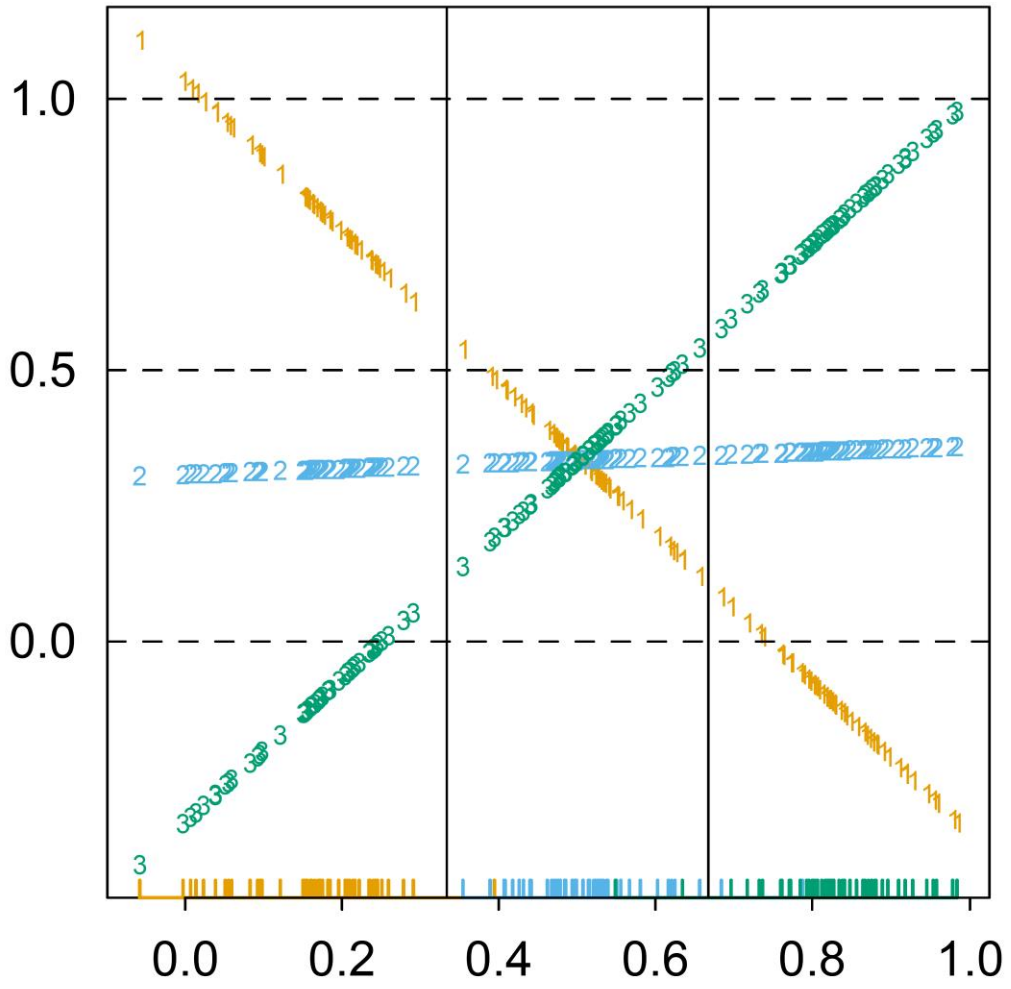
For each class k, $k = 1,2,\dots,K$, perform separate regression of form

$$Y_k = f_k(X) = \beta_{k0} + \beta_{k1}X_1 + \beta_{k2}X_2 + \cdots + \beta_{kp}X_p$$

Choose class k if $\hat{Y}_k$ is largest of all K classes.

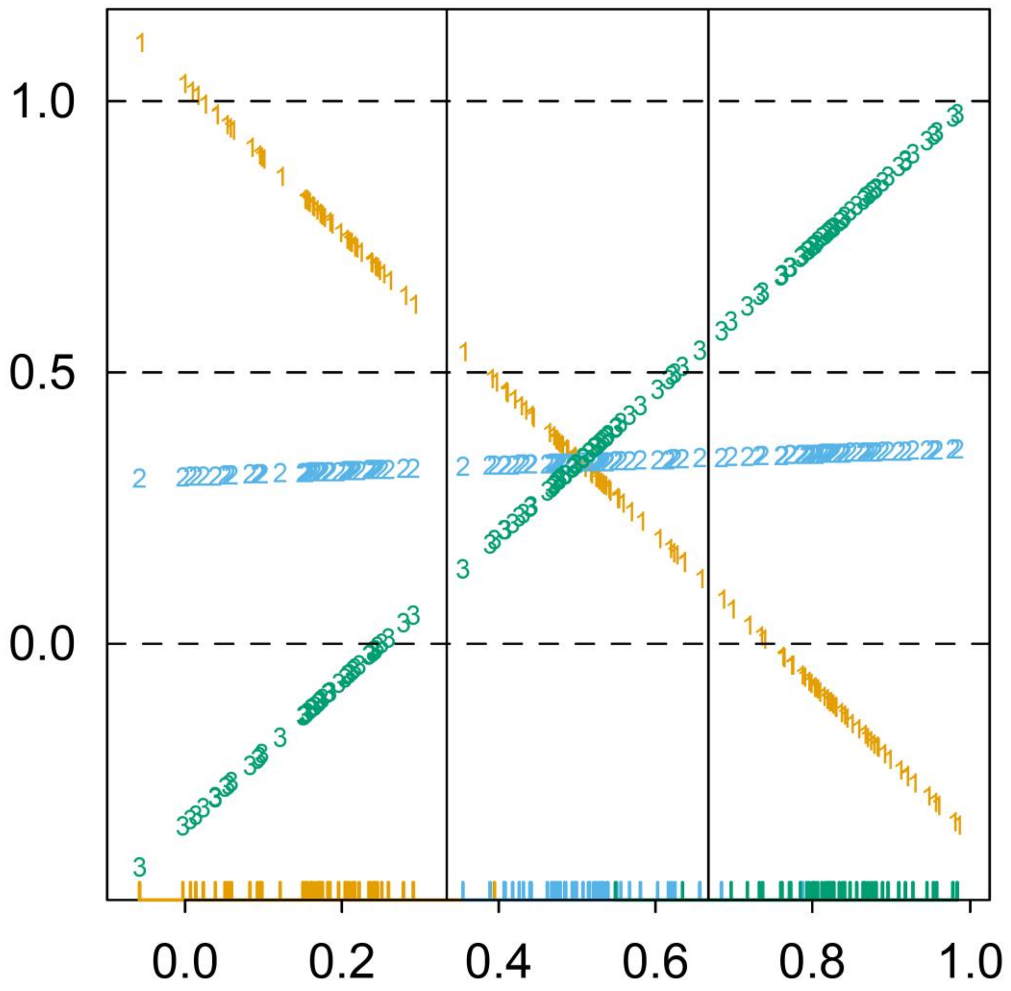
Example with $K = 3$ illustrating the effect of **masking**

Model with $K = 3$ classes (colors) and 1 feature X with the y-axis displaying the fit value of each function $\hat{f}_1(X)$, $\hat{f}_2(X)$, and $\hat{f}_3(X)$.



Example with $K = 3$ illustrating the effect of **masking**

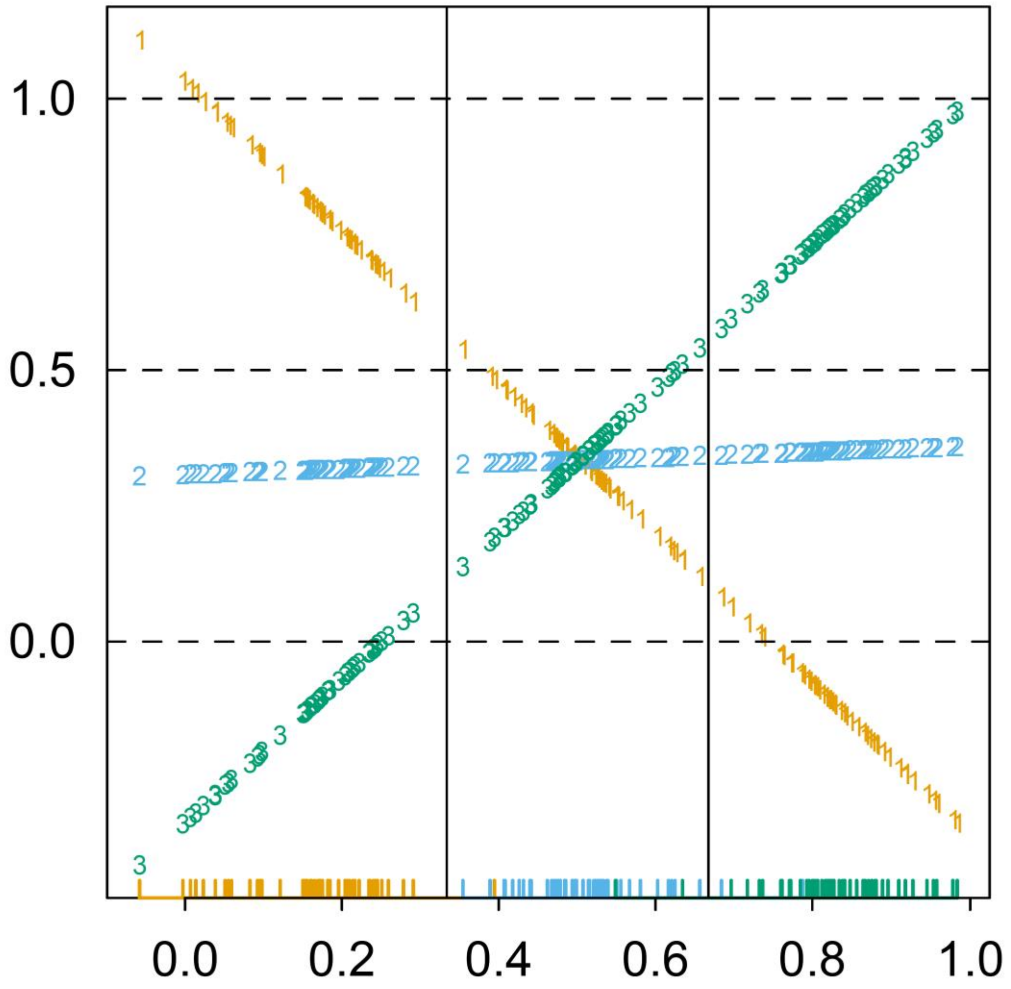
There is no region in this space where class 2 (blue) has highest predicted value, even though there are clearly 3 classes.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Example with $K = 3$ illustrating the effect of **masking**

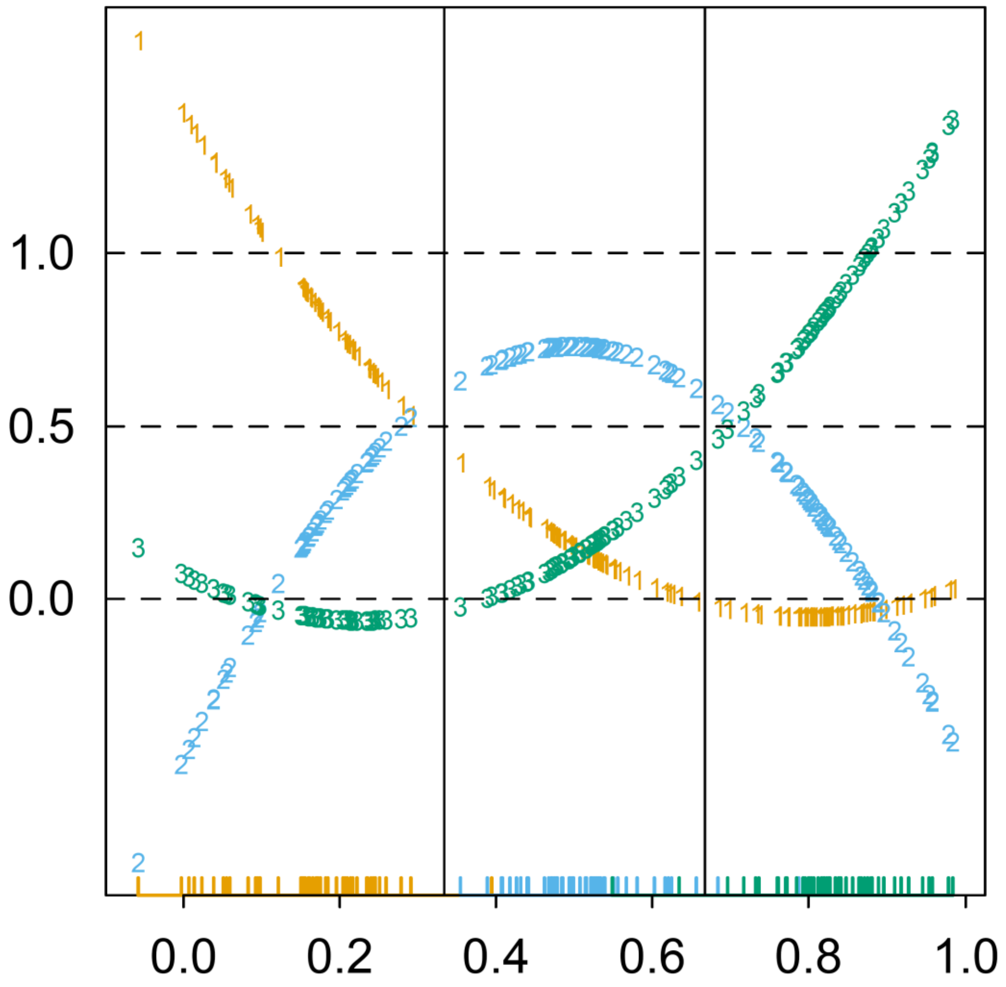
This effect is known as **masking**, and it can be particularly problematic when performing linear regression with $K > 2$ classes.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Masking can be rescued by transforming data

The issue of masking can be potentially alleviated by transforming the data, here **expanding model dimension** to a degree 2 polynomial.



That is, instead of modeling three regressions of the form

$$Y_1 = f_1(X) = \beta_{10} + \beta_{11}X$$
$$Y_2 = f_2(X) = \beta_{20} + \beta_{21}X$$
$$Y_3 = f_3(X) = \beta_{30} + \beta_{31}X$$

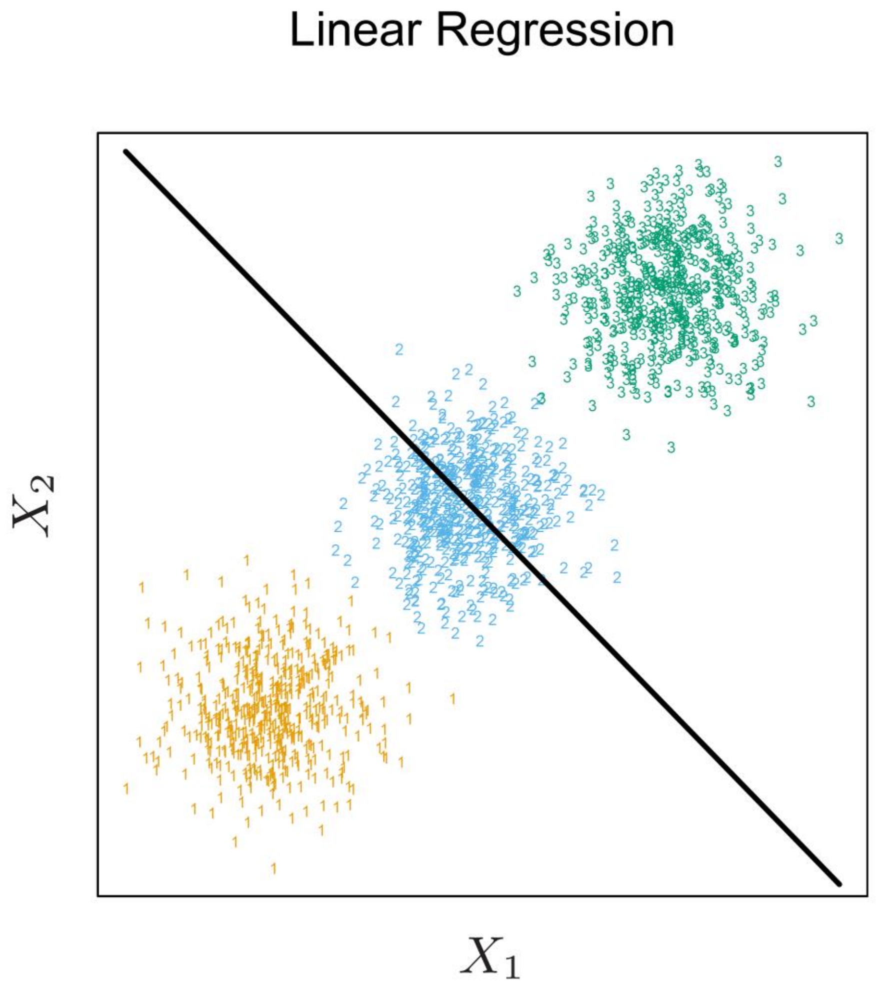
we model

$$Y_1 = f_1(X) = \beta_{10} + \beta_{11}X + \beta_{12}X^2$$
$$Y_2 = f_2(X) = \beta_{20} + \beta_{21}X + \beta_{22}X^2$$
$$Y_3 = f_3(X) = \beta_{30} + \beta_{31}X + \beta_{32}X^2$$

Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Additional masking example, with 2-dimensional input

Model with $K = 3$ classes (colors) and 2 features X_1 and X_2, with decision boundary as line, indicating blue class masked.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

General classification setup

Linear regression is not ideal for classification beyond $K = 2$ classes, requiring more appropriate method need to be constructed.

We will assume that we have a set of p features $X = [X_1, X_2, \dots, X_p]$ and a categorical response $Y = 1,2, \dots, K$, termed a class label.

Training data is the set of tuples (x_i, y_i), $i = 1,2, \dots, N$, where x_i is a list of p feature values and where y_i is a categorical response.

We want to form a predictor (classifier) $f(X)$ that can predict class Y from input X.

$f(X)$ will divide the input space into a collection of regions, where each region corresponds to one class label.

Improving upon the linear regression classifier

Suppose we have trained a classifier.

Given a new test observations with $X = [X_1, X_2, \dots, X_p]$ with p feature measurements, we wish to predict its true class Y, that can take on K possible values.

The observation is classified correctly if $\hat{Y} = Y$ and incorrectly if $\hat{Y} \neq Y$.

The ideal classifier is the one that on average makes the smallest number of errors.

It turns out that such an ideal classifier assigns each test observation to the **most likely class given its feature values** $X = [X_1, X_2, \dots, X_p]$.

Bayes classifier

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

The above equation is computing the probability of class Y given input feature values X.

Recall that the ideal classifier assigns each test observation to the most likely (most probable) class *given* its feature values.

Therefore, the ideal classifier applies Bayes' rule, and is known as the **Bayes classifier**.

That is, we use Bayes' rule to calculate the probability of each class conditional on the set of input feature values of a test observation, and then choose the class with the highest probability.

Bayes classifier with two classes

Consider a scenario with $K = 2$ classes.

The Bayes classifier corresponds to the following predictions

Class 1 if $P(Y = 1|X) > 0.5$

Class 2 if $P(Y = 1|X) < 0.5$

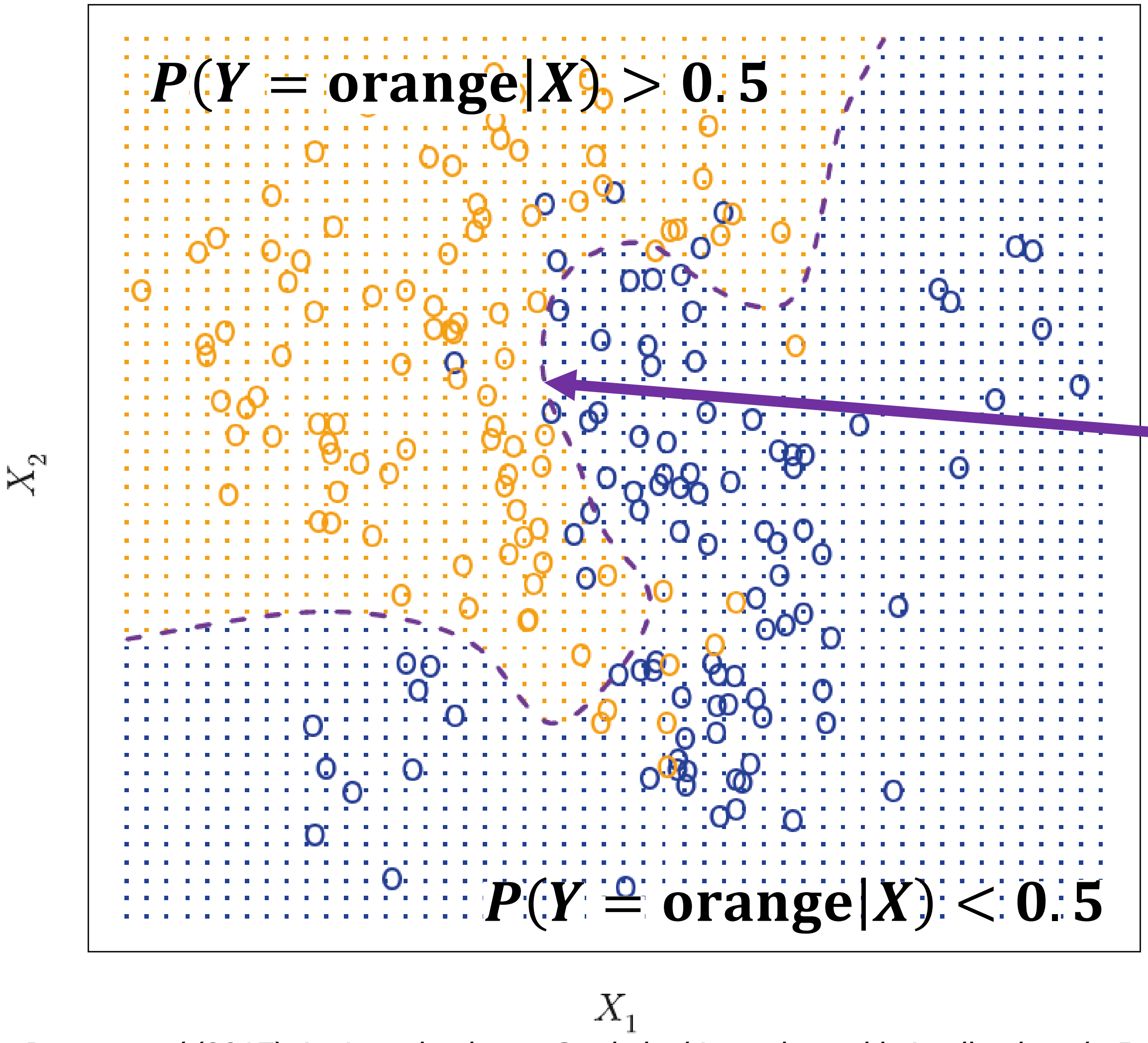
Note that this is the same as $P(Y = 2|X) = 1 - P(Y = 1|X)$, then the above two predictions are the same as the following predictions

Class 1 if $P(Y = 2|X) < 0.5$

Class 2 if $P(Y = 2|X) > 0.5$

Bayes classifier with two classes

Simulated dataset of $N = 100$ observations on $p = 2$ features $X = [X_1, X_2]$, with $K = 2$ classes (orange and blue).

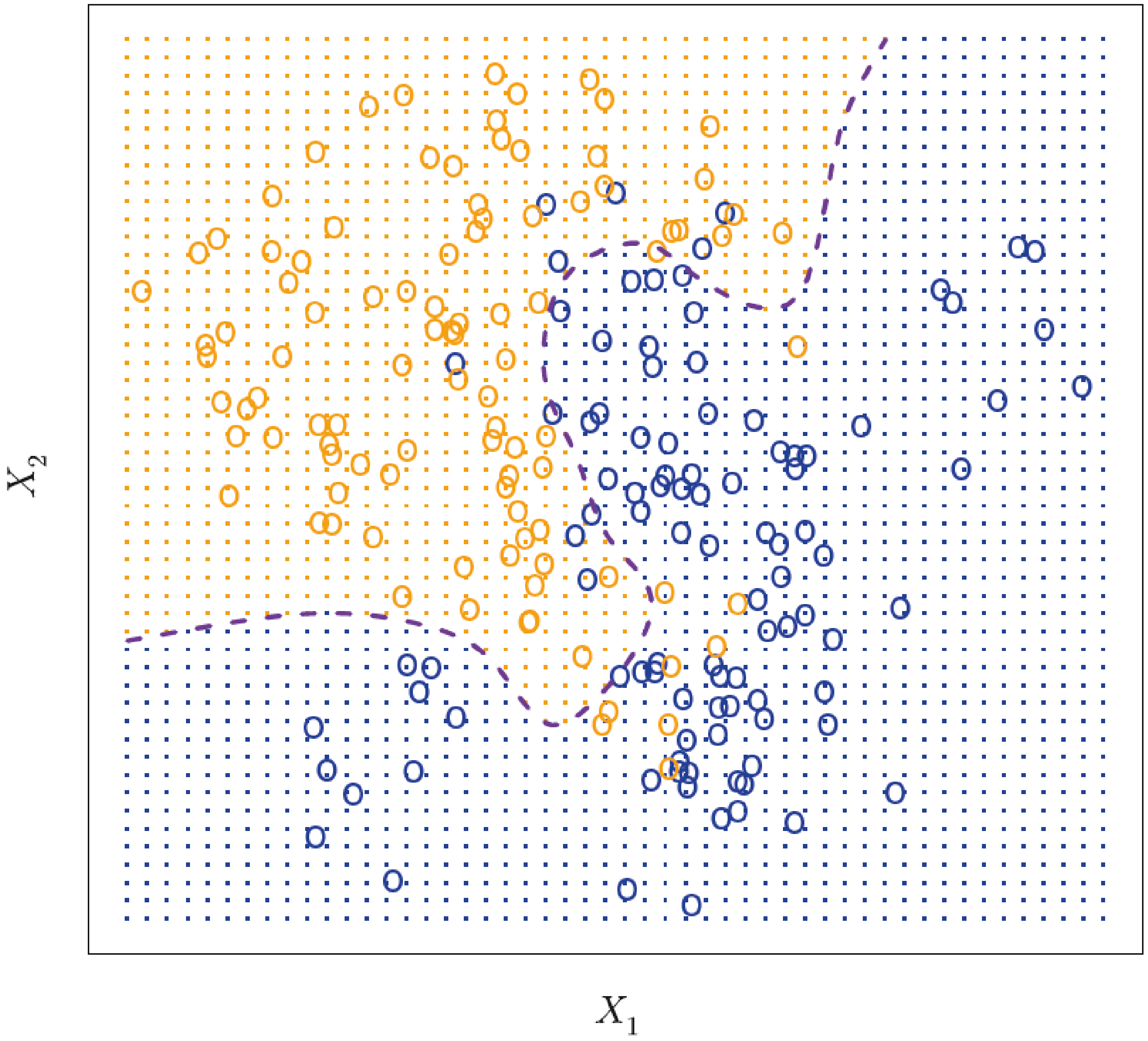


Bayes decision boundary
$P(Y = \text{orange}|X) = 0.5$

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Bayes classifier is "gold standard" classifier

This is because it has the lowest possible test error rate of any other classifier, with this test error known as the **Bayes error rate**.



In this example, the Bayes error rate is 0.1304.

The reason the error rate is greater than zero is due to the classes overlapping in the true population.

Therefore, in this example, it is impossible to decrease the error rate further.

James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

The issue with the Bayes classifier

It is impossible to achieve higher accuracy (lower error rate) than with the Bayes classifier

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

However, it is also impossible to apply Bayes classifier to real data.

The reason is that we cannot compute $P(Y|X)$ from real data because we do not know the conditional distribution of class Y given input feature set X.

We can make assumptions about the form of this distribution, but we never know its true form unless we are directly generating simulated data and therefore have control over the distribution.

So why do we care about the Bayes classifier?

Though the Bayes classifier cannot be applied to real data, it does not mean that it is not useful.

The Bayes classifier is highly useful for assessing the test accuracy (error) of other classifiers, because it provides a lower bound for the test error rate of any classifier.

To evaluate the accuracy of a new classifier, we can apply both the new classifier and the Bayes classifier to simulated data, for which we control (know) the conditional distribution $P(Y|X)$.

We then compare the error rates of the new and Bayes classifiers to determine how well the new classifier performs relative to the best-case scenario.

Extensive set of other classifiers

There are a number methods for performing classification, each with its strengths and weaknesses.

That is, none are uniformly better than the others.

- Discriminant analysis
- Logistic regression
- K-nearest neighbors
- Bagging
- Random forests
- Boosting
- Support vector machines
- Neural networks

Linear discriminant analysis (LDA)

We now move to our first approach that is specifically tailored for classification, known as **linear discriminant analysis** (**LDA**).

We are going to consider an approach that models the probability of the class label Y given input data X. That is, we are going to model

$$P(Y|X)$$

Recall Bayes' rule from the first lecture (Introduction), which gives

$$P(Y=k|X=x) = \frac{P(X=x|Y=k)P(Y=k)}{\sum_{\ell=1}^{K} P(X=x|Y=\ell)P(Y=\ell)}$$

Here we will write $p_k(x) = P(Y=k|X=x)$, $f_k(x) = P(X=x|Y=k)$, and $\pi_k = P(Y=k)$, which is the prior of class k.

Linear discriminant analysis (LDA)

Rewriting gives

$$p_k(x) = \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell}$$

We can see the probability density $f_k(x)$ of x given class label k is somewhat of a proxy for the probability we are after.

If we can compute the quantities on the right hand side, then we can assign the class label for an observation X as

$$Y(X) = \underset{k \in \{1,2,\dots,K\}}{\text{argmax}} p_k(X)$$

So how do we compute these quantities?

Modeling input for each class label

Under LDA, we will assume that input X are drawn from a multivariate Gaussian (normal) distribution with mean $\mu_k \in \mathbb{R}^p$ and covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{p \times p}$ for class k.

That is, we assume each class has the same covariance structure, but different means (**centroids**).

We write $X \sim N(\mu_k, \mathbf{\Sigma})$, with probability density defined as

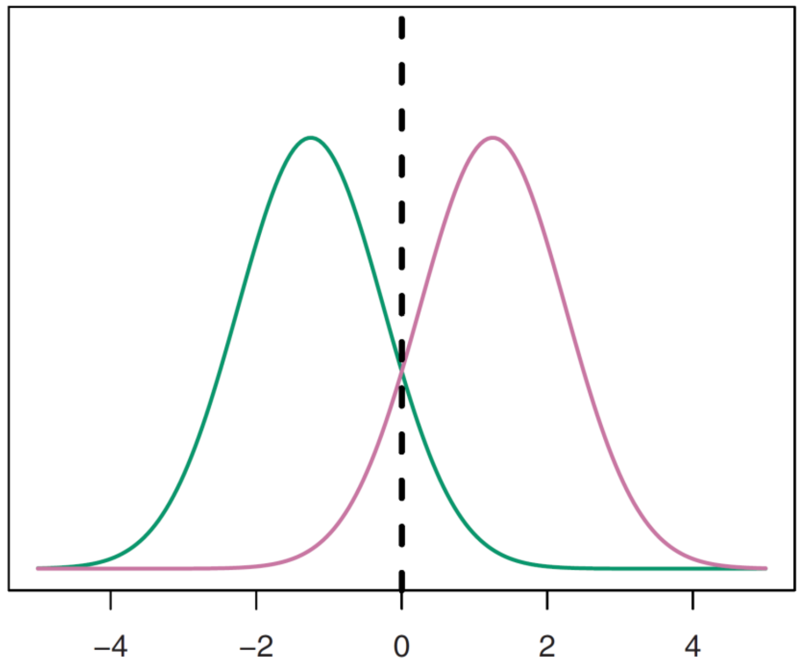
$$f_k(x) = f(x; \mu_k, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{p/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} (x - \mu_k)^T \mathbf{\Sigma}^{-1} (x - \mu_k) \right)$$

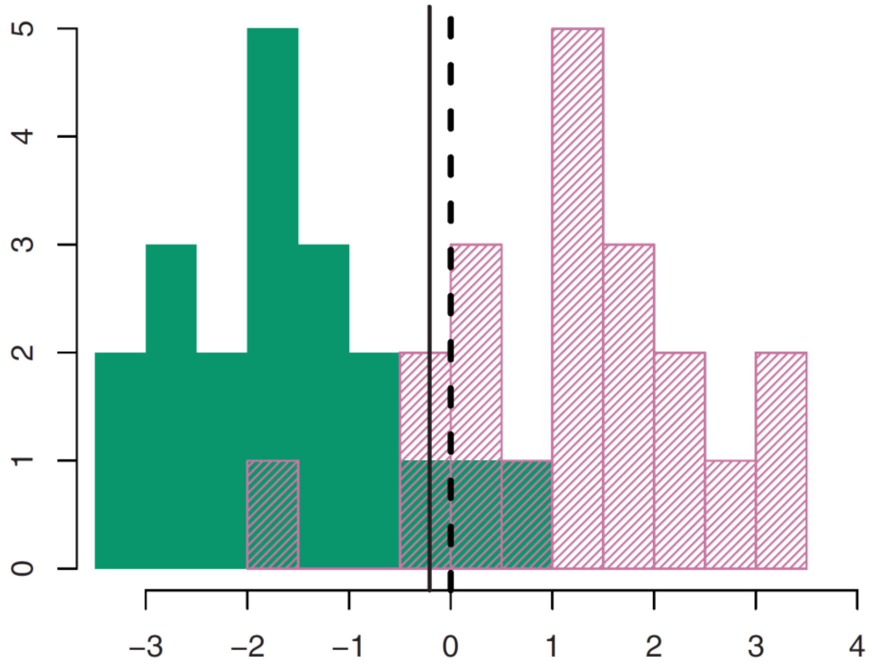
where $|\mathbf{\Sigma}|$ denotes the determinant of $\mathbf{\Sigma}$.

Modeling 2 classes with 2 Gaussian distributions

Left displays 2 Gaussian distributions with identical variances but different means, with distribution a mixture of the 2.

Right displays histograms of 20 observations drawn from each class (LDA decision boundary as vertical black line)

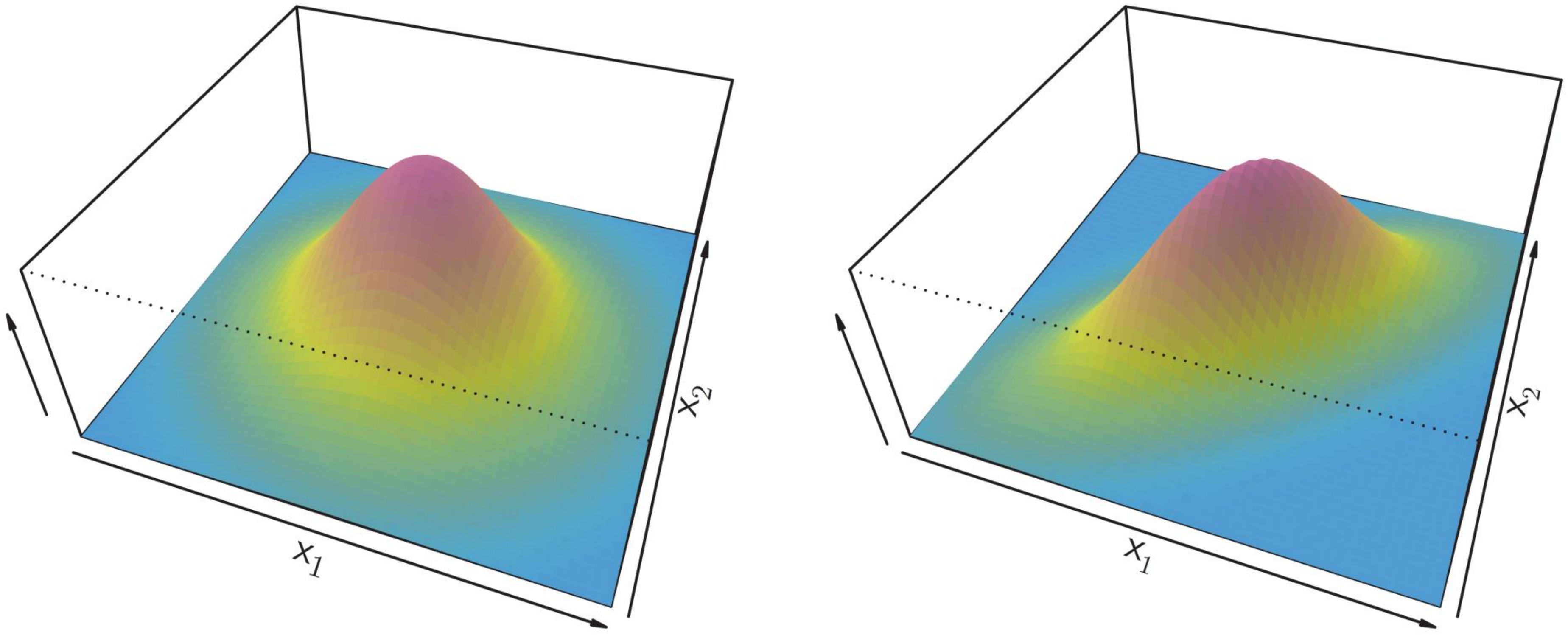




Example bivariate Gaussian distributions

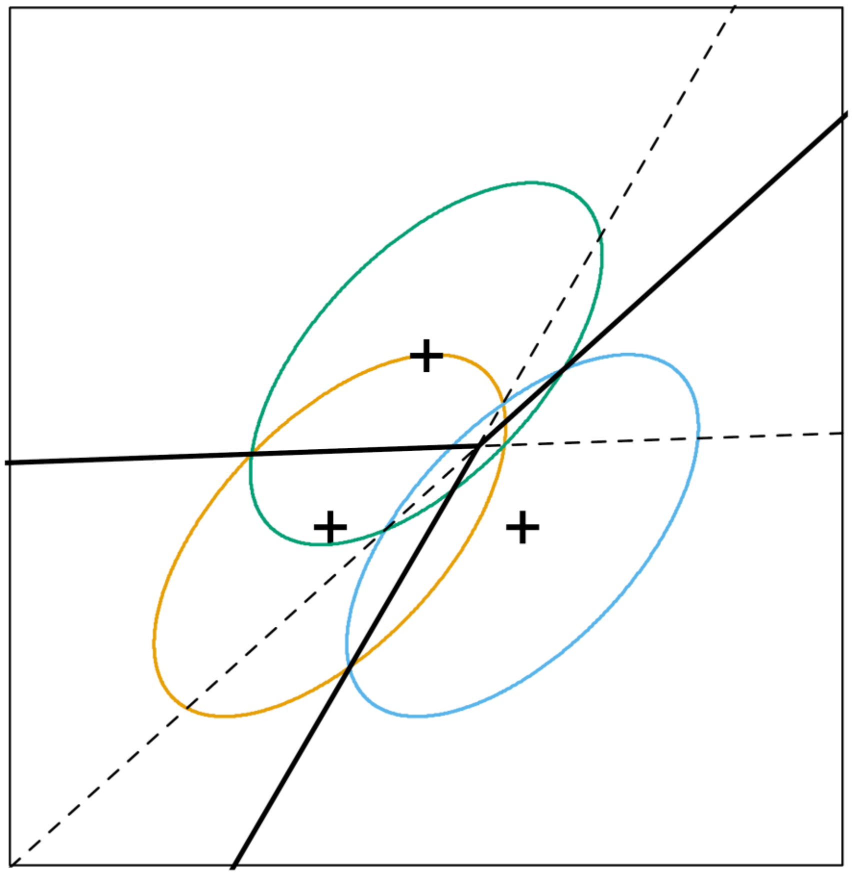
Left displays bivariate Gaussian distribution with uncorrelated features.

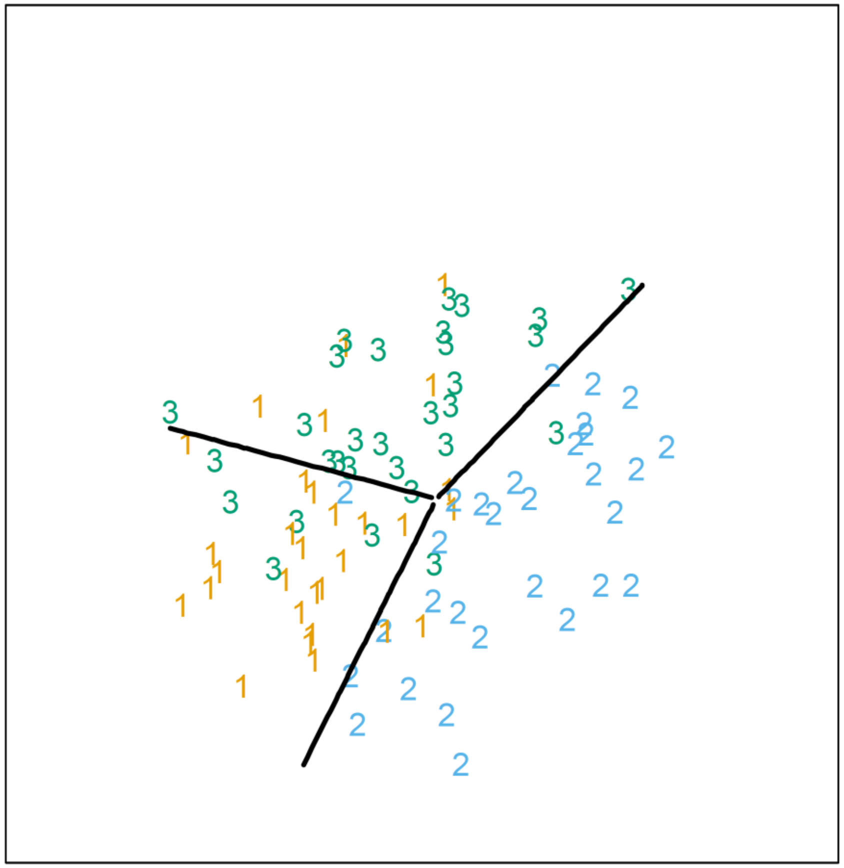
Right displays bivariate Gaussian distribution with features correlated with coefficient 0.7.



Modeling 3 classes with 3 Gaussian distributions

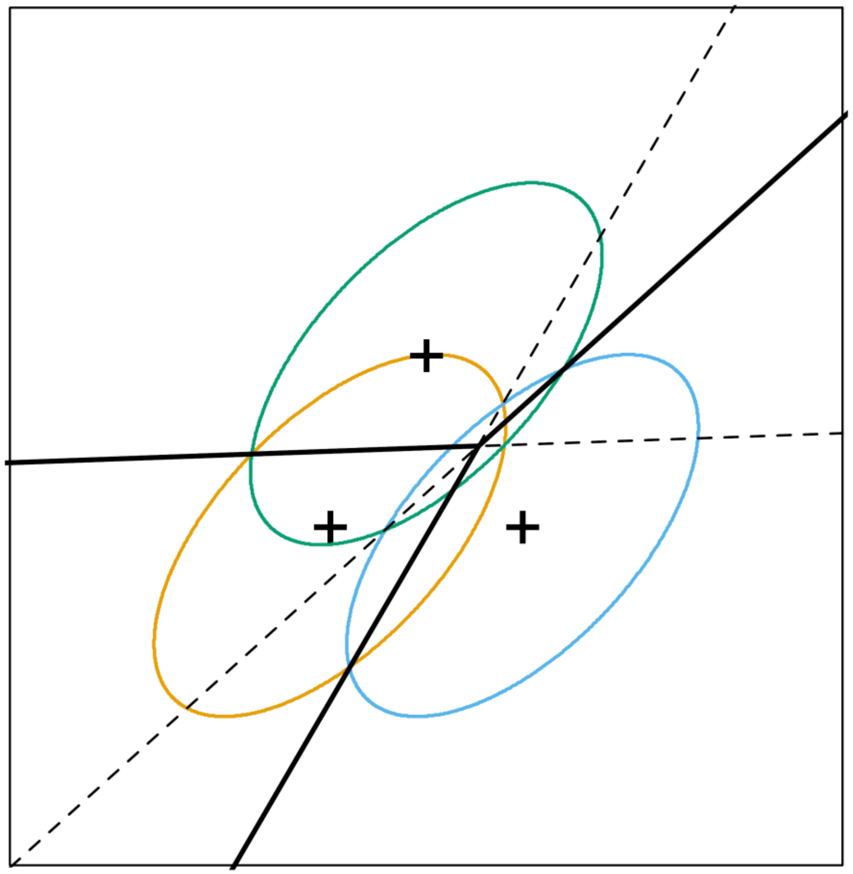
Left displays 3 bivariate Gaussian distributions (95% of density in each contour) with identical covariance matrix but different means (centroids) indicated by crosses, with distribution a mixture of the 3.

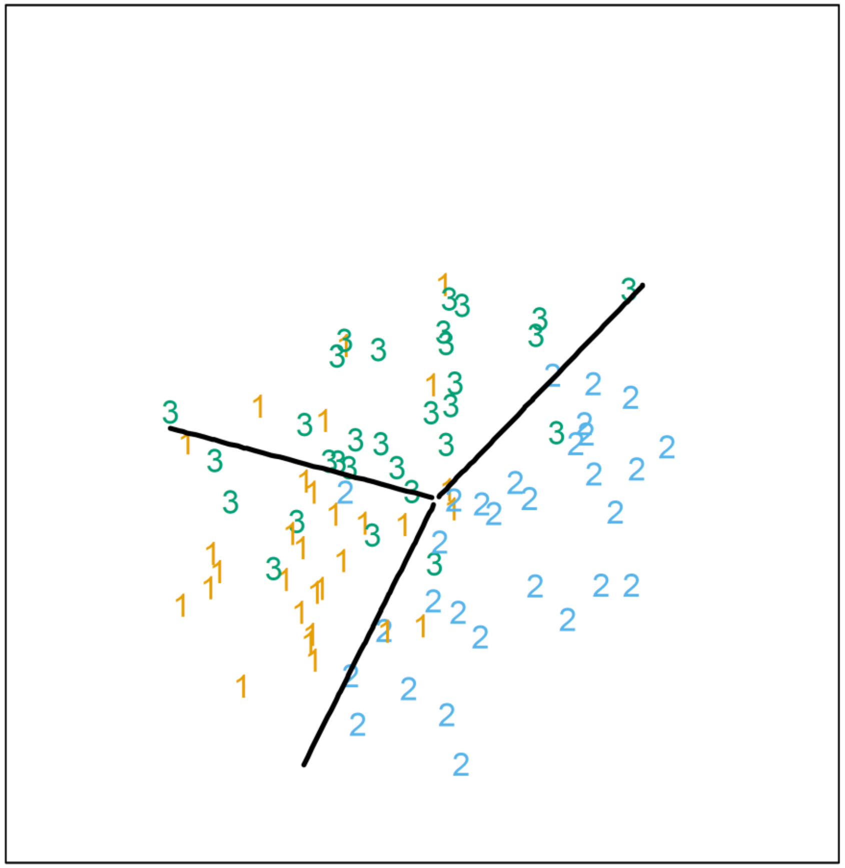




Modeling 3 classes with 3 Gaussian distributions

Right displays a scatter plot of 30 observations drawn from each class (LDA decision as black lines)





How do we choose the best class label?

Recall that we need to identify k such that $p_k(x)$ is maximized, or equivalently find k such that $\log p_k(x)$ is maximized, which is

$$\begin{aligned}\log p_k(x) &= \log \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell} \\ &= \log f_k(x) + \log \pi_k - \log\left(\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell\right)\end{aligned}$$

Computing $\log f_k(x)$, we have

$$\begin{aligned}\log f_k(x) &= \log \frac{1}{(2\pi)^{p/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}^{-1}(x-\mu_k)\right) \\ &= -\frac{1}{2}\log\big((2\pi)^p |\mathbf{\Sigma}|\big) - \frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}^{-1}(x-\mu_k)\end{aligned}$$

How do we choose the best class label?

Note that $(x-\mu_k)^T\boldsymbol{\Sigma}^{-1}(x-\mu_k)$ can be expanded as

$$\begin{aligned}(x-\mu_k)^T\boldsymbol{\Sigma}^{-1}(x-\mu_k) &= \left(x^T-\mu_k^T\right)\boldsymbol{\Sigma}^{-1}(x-\mu_k)\\ &= x^T\boldsymbol{\Sigma}^{-1}x - x^T\boldsymbol{\Sigma}^{-1}\mu_k - \mu_k^T\boldsymbol{\Sigma}^{-1}x + \mu_k^T\boldsymbol{\Sigma}^{-1}\mu_k\\ &= x^T\boldsymbol{\Sigma}^{-1}x - 2x^T\boldsymbol{\Sigma}^{-1}\mu_k + \mu_k^T\boldsymbol{\Sigma}^{-1}\mu_k\end{aligned}$$

because

$$\mu_k^T\boldsymbol{\Sigma}^{-1}x = \left(\mu_k^T\boldsymbol{\Sigma}^{-1}x\right)^T = x^T\boldsymbol{\Sigma}^{-1}\mu_k$$

Therefore

$$\begin{aligned}\log f_k(x) &= \log\frac{1}{(2\pi)^{p/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_k)^T\boldsymbol{\Sigma}^{-1}(x-\mu_k)\right)\\ &= -\frac{1}{2}\log\left((2\pi)^p|\boldsymbol{\Sigma}|\right) - \frac{1}{2}x^T\boldsymbol{\Sigma}^{-1}x + x^T\boldsymbol{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\boldsymbol{\Sigma}^{-1}\mu_k\end{aligned}$$

How do we choose the best class label?

Putting it all together, we have

$$\begin{aligned}\log p_k(x) &= x^T\mathbf{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}^{-1}\mu_k + \log\pi_k \\ &\quad -\frac{1}{2}x^T\mathbf{\Sigma}^{-1}x - \frac{1}{2}\log\left((2\pi)^p|\mathbf{\Sigma}|\right) - \log\left(\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell\right) \\ &= \delta_k(x) - C(x)\end{aligned}$$

where

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

$$C(x) = \frac{1}{2}x^T\mathbf{\Sigma}^{-1}x + \frac{1}{2}\log\left((2\pi)^p|\mathbf{\Sigma}|\right) + \log\left(\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell\right)$$

Because $C(x)$ is the same regardless of class label, we only need to find k that maximizes $\delta_k(x)$.

How do we choose the best class label?

Linear discriminant analysis therefore assigns class label to observation X as

$$Y(X) = \underset{k \in \{1,2,\dots,K\}}{\operatorname{argmax}} \delta_k(X)$$

where

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

is known as the **linear discriminant function** for class k, and is linear in the input x, and the linear decision boundary between classes k and ℓ is the log odds

$$\log \frac{P(Y = k|X = x)}{P(Y = \ell|X = x)} = \log \frac{p_k(x)}{p_\ell(x)} = \log p_k(x) - \log p_\ell(x)$$

$$= \delta_k(x) - C(x) - \ [\delta_\ell(x) - C(x)] \ = \delta_k(x) - \delta_\ell(x)$$

with the set of points where this log odds equals 0 forming a hyperplane separating classes k and ℓ.

How do we train the model?

So far we have discussed how to choose the optimal class given the linear discriminant function $\delta_k(x)$ for each class k.

However, there are numerous unknown parameters in this model, including

- K π_k parameters
- K μ_k parameters
- $\frac{p(p+1)}{2}$ $\boldsymbol{\Sigma}$ parameters, with p variance terms and $\binom{p}{2} = \frac{p(p-1)}{2}$ covariance terms for different features.

Estimate all these parameters from the N training observations.

Estimating model parameters from training data

Denote the number of training observations in class k by N_k, such that $N = N_1 + N_2 + \cdots + N_K$.

The prior probability $\pi_k = P(Y = k)$ of class k can be estimated by the sample proportion, or the fraction of training observations from class k. That is

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

We estimate the mean of class k by the mean input value of all training observations from class k. That is

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i=1:\, y_i=k}^{N} x_i$$

Estimating model parameters from training data

Because the covariance matrix is the same for each class, we use the entirety of the training dataset to estimate the covariance.

This **pooled covariance matrix** is computed as

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{i=1:\, y_i=k}^{N} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

where the input values for observations from class k are centered with respect to the sample mean of class k, and where

$$aa^T = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_p \end{bmatrix} = \begin{bmatrix} a_1^2 & a_1 a_2 & \cdots & a_1 a_p \\ a_2 a_1 & a_2^2 & \cdots & a_2 a_p \\ \vdots & \vdots & \ddots & \vdots \\ a_p a_1 & a_p a_2 & \cdots & a_p^2 \end{bmatrix}$$

with $a \in \mathbb{R}^p$ and assuming $a = (x_i - \hat{\mu}_k)$.

Predictions based on fitted model

Once we have estimated these parameters, we can compute the linear discriminant function for each class k as

$$\hat{\delta}_k(x) = x^T \hat{\boldsymbol{\Sigma}}^{-1} \hat{\mu}_k - \frac{1}{2} \hat{\mu}_k^T \hat{\boldsymbol{\Sigma}}^{-1} \hat{\mu}_k + \log \hat{\pi}_k$$

and the class label for observation X is assigned as

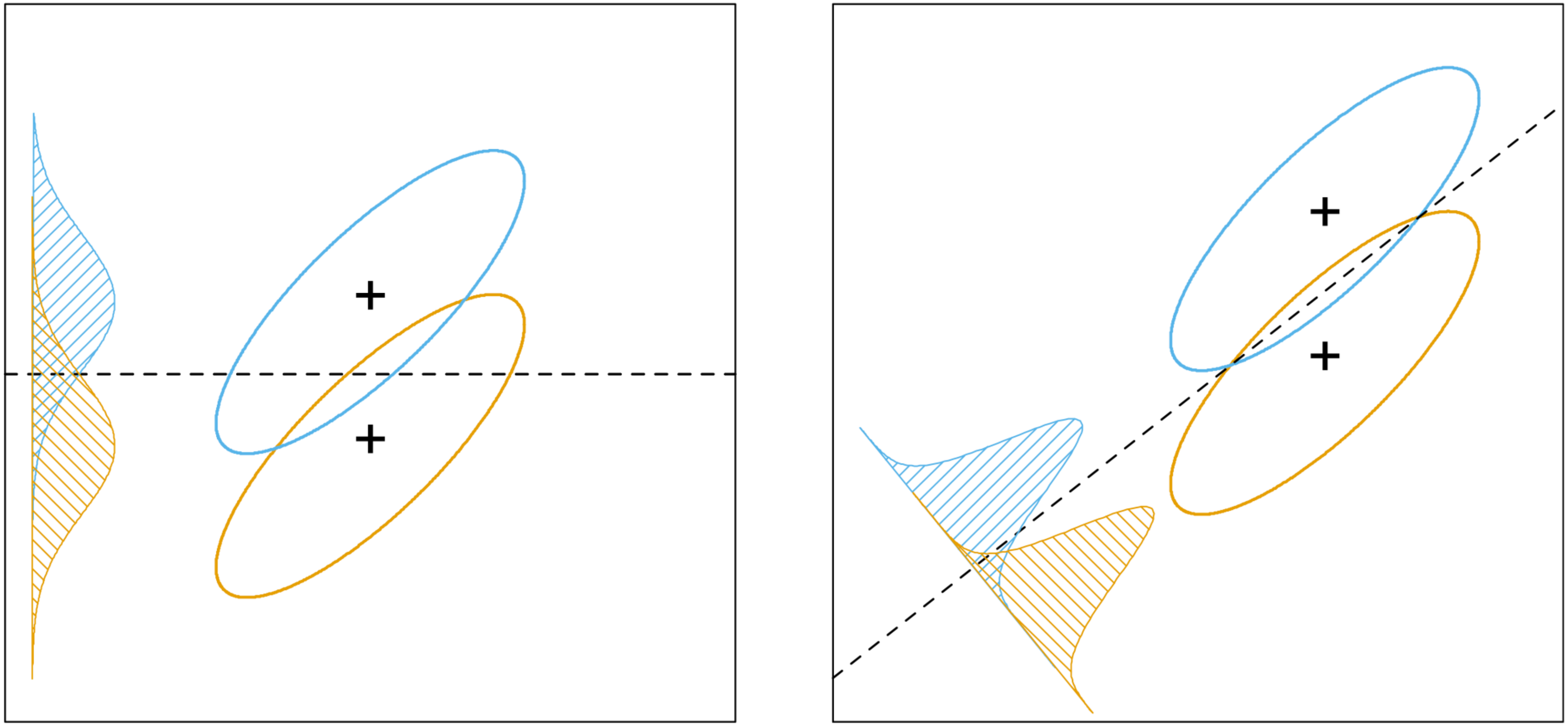
$$\hat{Y}(X) = \underset{k \in \{1,2,\ldots,K\}}{\operatorname{argmax}} \hat{\delta}_k(X)$$

and the linear decision boundary between classes k and ℓ is

$$\hat{\delta}_k(x) = \hat{\delta}_\ell(x)$$

Discriminant function minimizes overlap for Gaussians

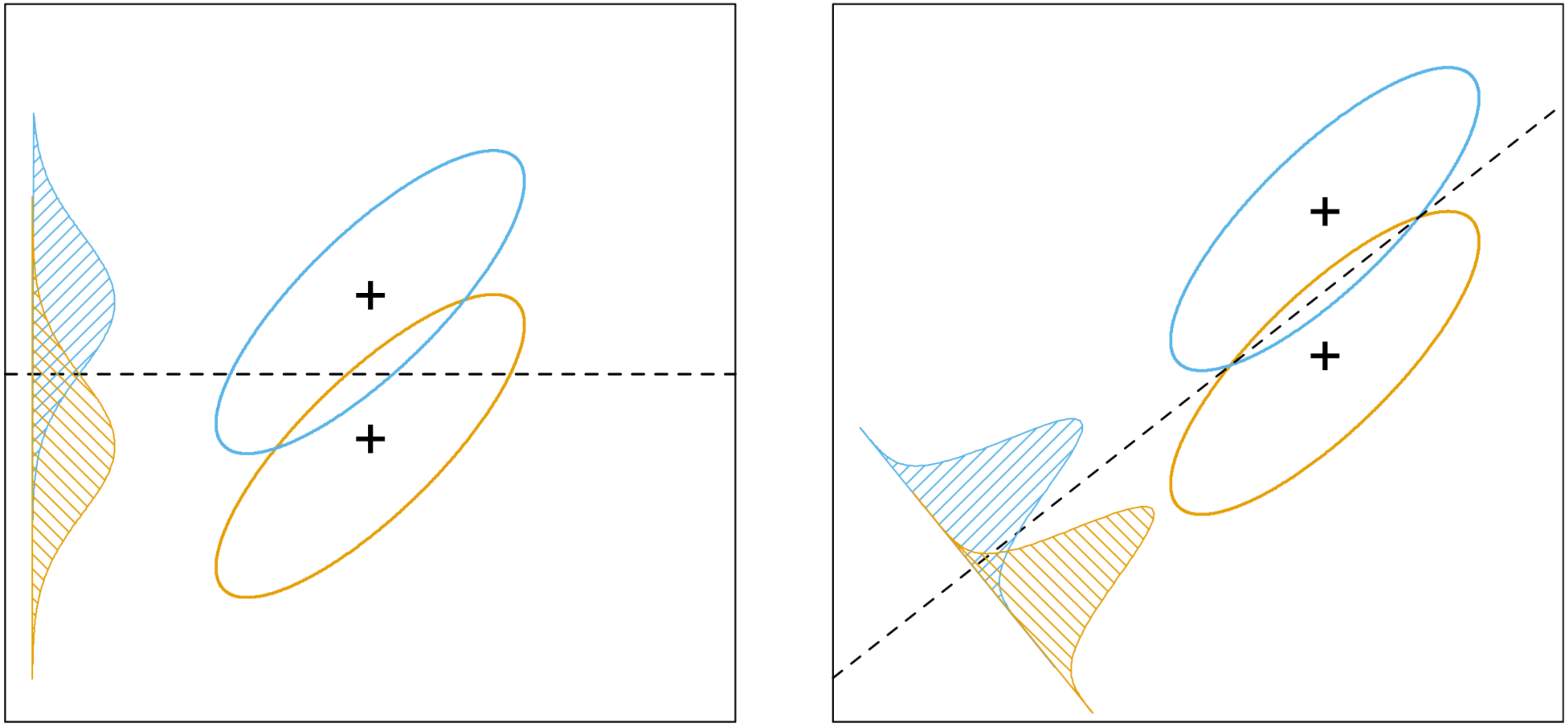
Left displays that the direction of greatest centroid spread (vertical) leads to projected data with a large overlap in their distributions, due to the covariance structure of the classes.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

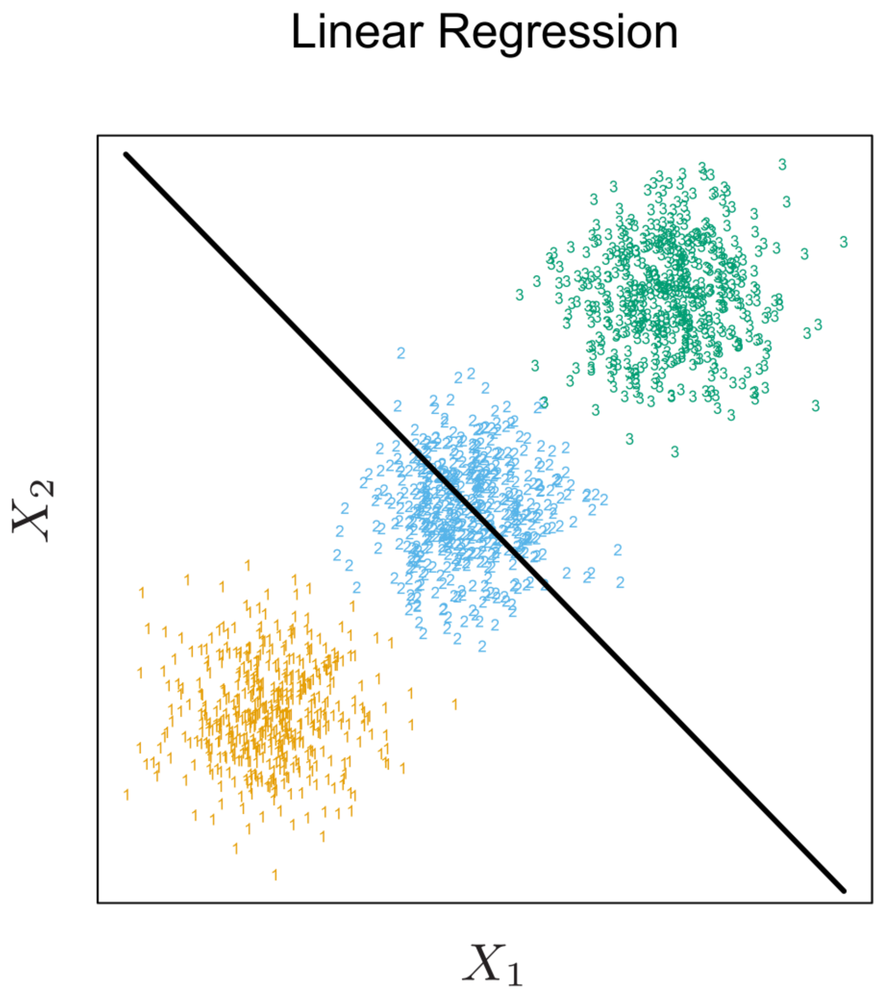
Discriminant function minimizes overlap for Gaussians

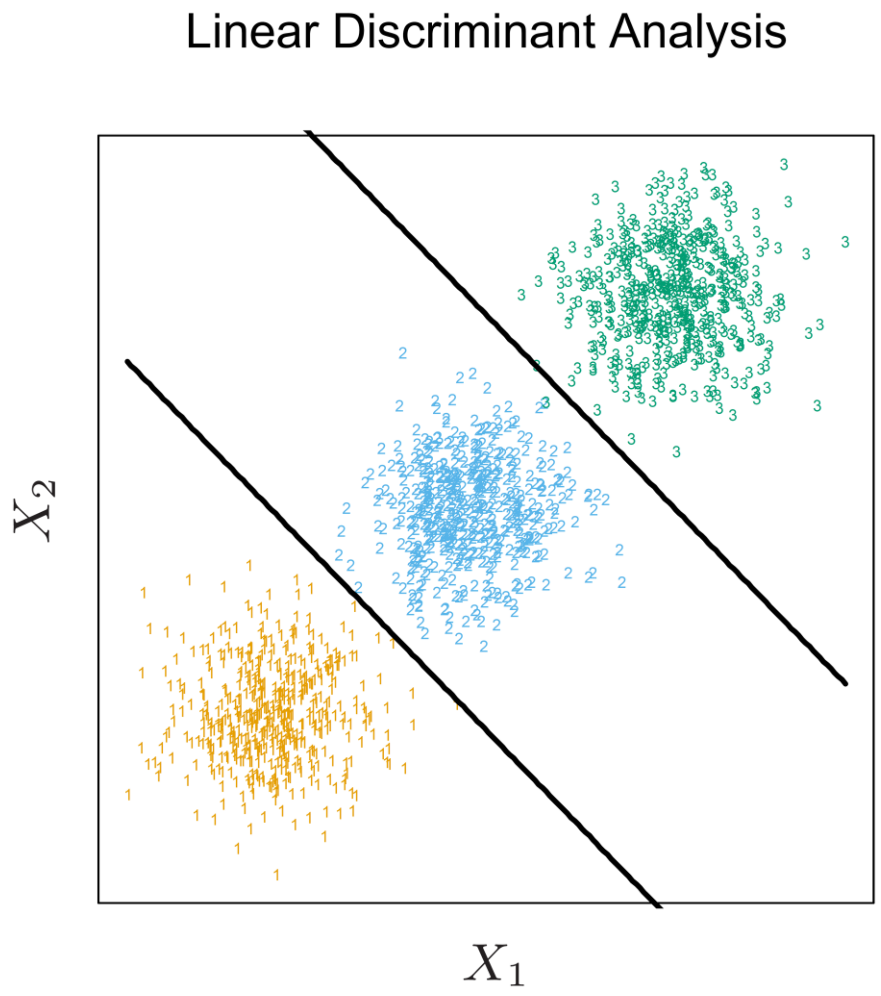
Right displays that the discriminant function (dotted line) minimizes this overlap for the Gaussian data.



Masking not a problem for LDA

Model with $K = 3$ classes (colors) and 2 features X_1 and X_2, with decision boundaries as lines.

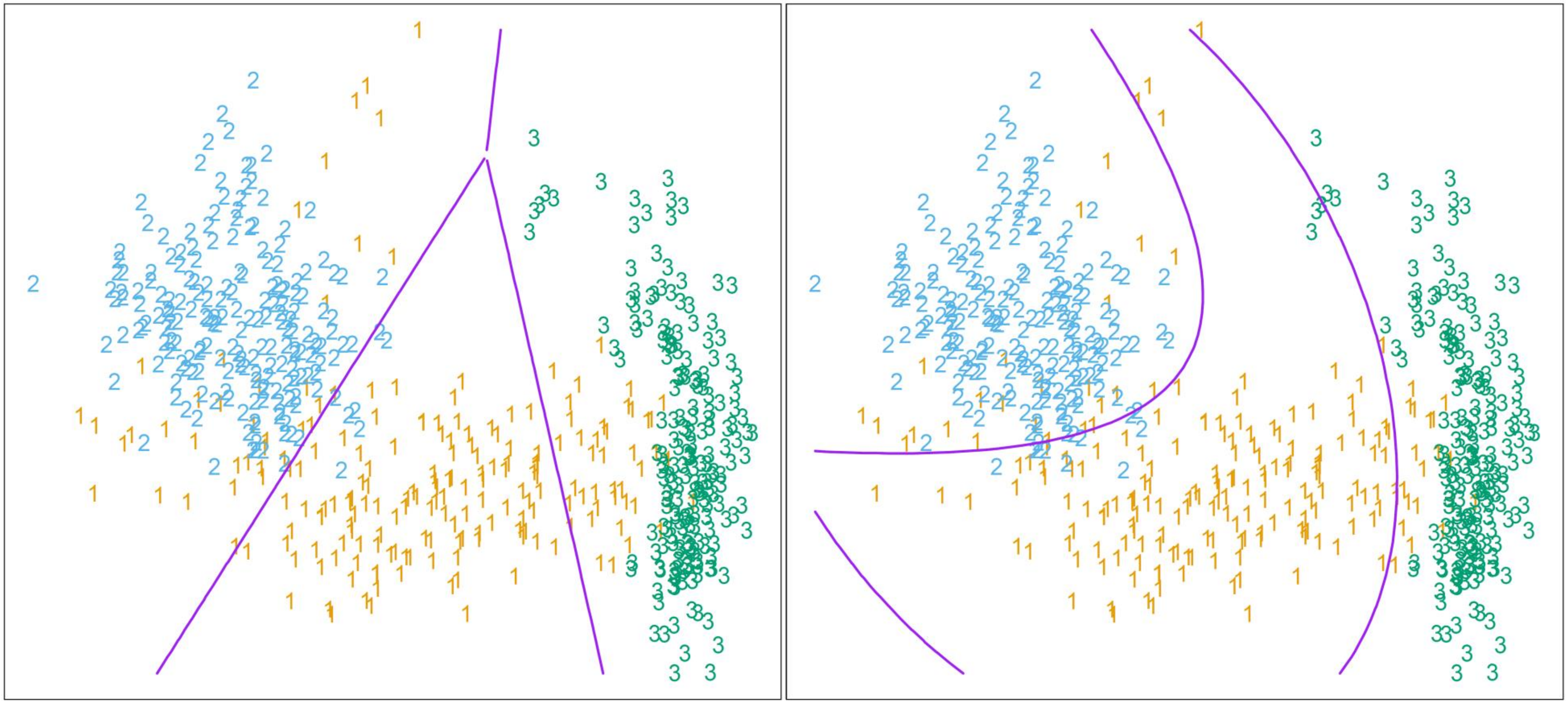




Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Expanding the feature dimension can help

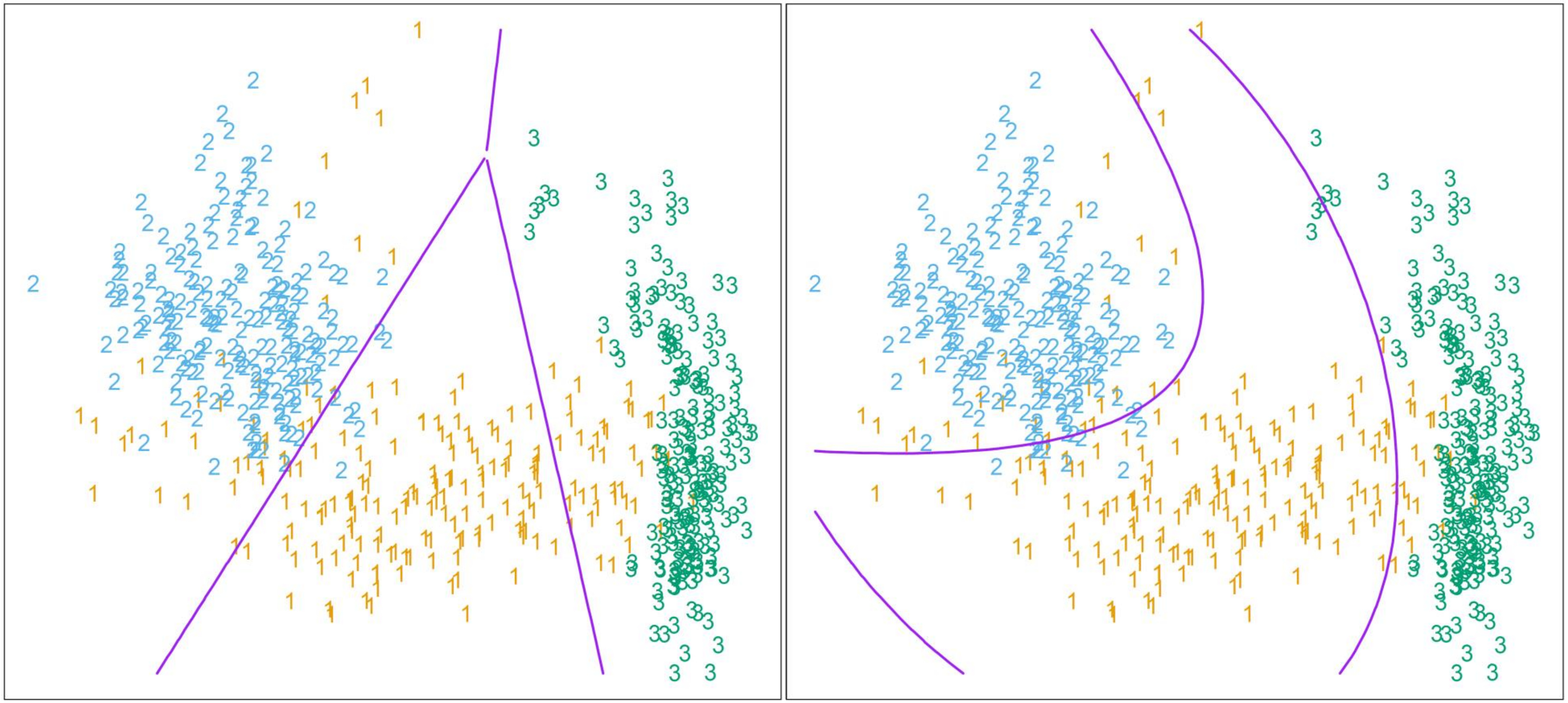
Left displays the linear decision boundaries from LDA applied to training data with 2 features X_1 and X_2, highlighting substantial misclassification of class 1 (orange).



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

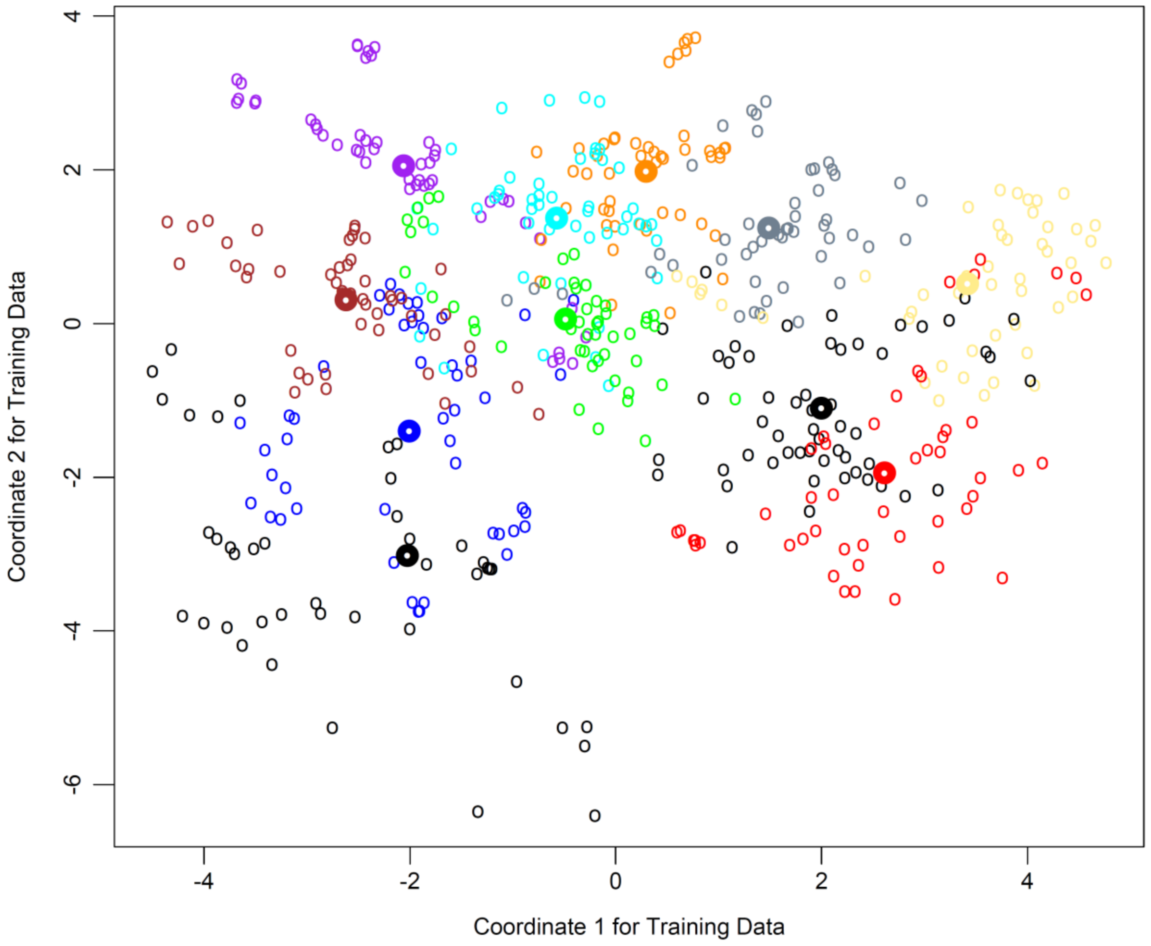
Expanding the feature dimension can help

Right displays the linear decision boundaries from LDA applied to training data by expanding the features to a 5-dimensional space of X_1, X_2, X_1X_2, X_1^2, and X_2^2, highlighting much less misclassification.



Example complicated training dataset

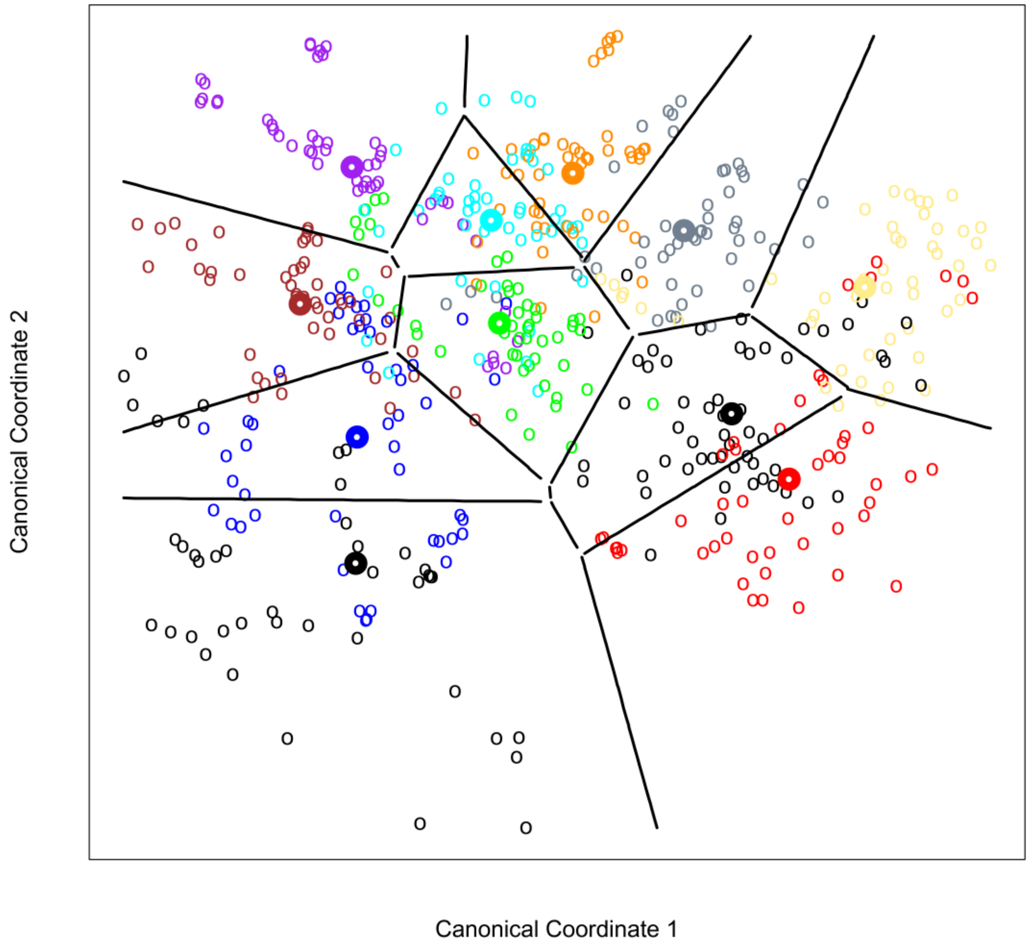
Data contains $K = 11$ classes (colors) with $p = 10$ features, of which 3 of the features account for 90% of the variance (as computed by PCA). There is considerable class overlap.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Decision boundaries for LDA in two dimensions

Data contains $K = 11$ classes (colors) with $p = 10$ features, of which 3 of the features account for 90% of the variance (as computed by PCA). LDA was applied to the two displayed coordinate dimensions.

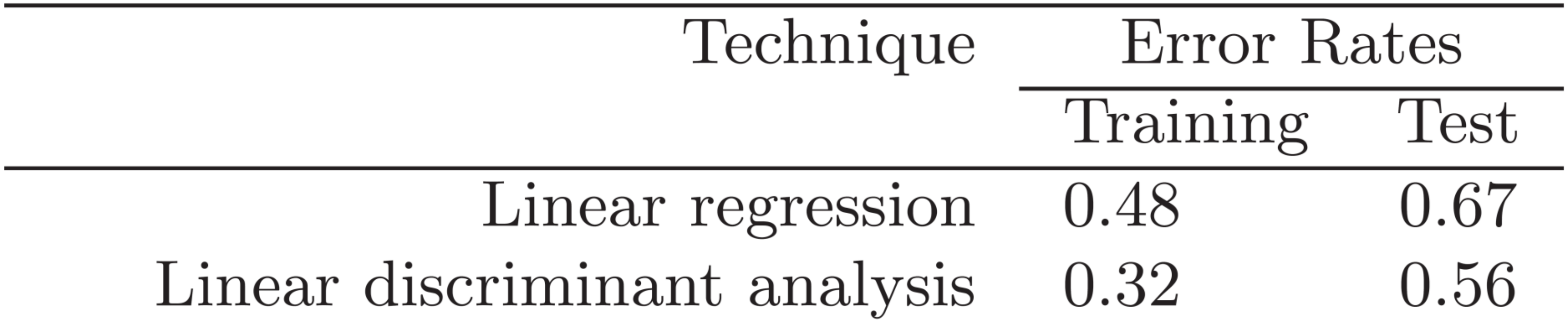


Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Error rates for indicator linear regression and LDA

Data contains $K = 11$ classes with $p = 10$ features, of which 3 of the features account for 90% of the variance (as computed by PCA).

Indicator linear regression is affected by masking, leading to large increases in both training and testing errors relative to LDA.



Technique	Error Rates	
	Training	Test
Linear regression	0.48	0.67
Linear discriminant analysis	0.32	0.56

Adapted from Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Relaxing the equal covariance assumption

The assumption of identical covariance structure $\mathbf{\Sigma}$ across classes is unlikely to be realistic.

We can then consider developing a classifier that permits separate covariance matrices $\mathbf{\Sigma}_k$ for each class k.

Let us assume that observations drawn from class k follow a multivariate normal distribution with mean μ_k and covariance $\mathbf{\Sigma}_k$, and we write $X \sim N(\mu_k, \mathbf{\Sigma}_k)$ with probability density

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

Choose the best class under this relaxed model

We can simply swap $\mathbf{\Sigma}_k$ for $\mathbf{\Sigma}$ in $\log p_k(x)$ to get

$$\log p_k(x) = \log \pi_k - \frac{1}{2}(x-\mu_k)^T {\mathbf{\Sigma}_k}^{-1}(x-\mu_k) \; - \frac{1}{2}\log|\mathbf{\Sigma}_k|$$

$$-\frac{p}{2}\log 2\pi - \log\left(\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell\right)$$

$$= \delta_k(x) - D(x)$$

where

$$\delta_k(x) = \log \pi_k - \frac{1}{2}(x-\mu_k)^T {\mathbf{\Sigma}_k}^{-1}(x-\mu_k) \; - \frac{1}{2}\log|\mathbf{\Sigma}_k|$$

$$D(x) = \frac{p}{2}\log 2\pi + \log\left(\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell\right)$$

Because $D(x)$ does not depend on the class label k, we only need to find k that maximizes $\delta_k(x)$.

How do we choose the best class label?

This classifier is known as **quadratic discriminant analysis** (**QDA**) and assigns class label to observation X as

$$Y(X) = \underset{k \in \{1,2,\ldots,K\}}{\operatorname{argmax}} \delta_k(X)$$

where

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

is known as the **quadratic discriminant function** for class k, and is quadratic in the input x.

How to we fit the model?

Just like with LDA, we can estimate the model parameters in QDA from a set of N training observations, but the number of parameters will be larger as we are now estimating K covariance matrices instead of 1.

There are numerous unknown parameters in this model, including

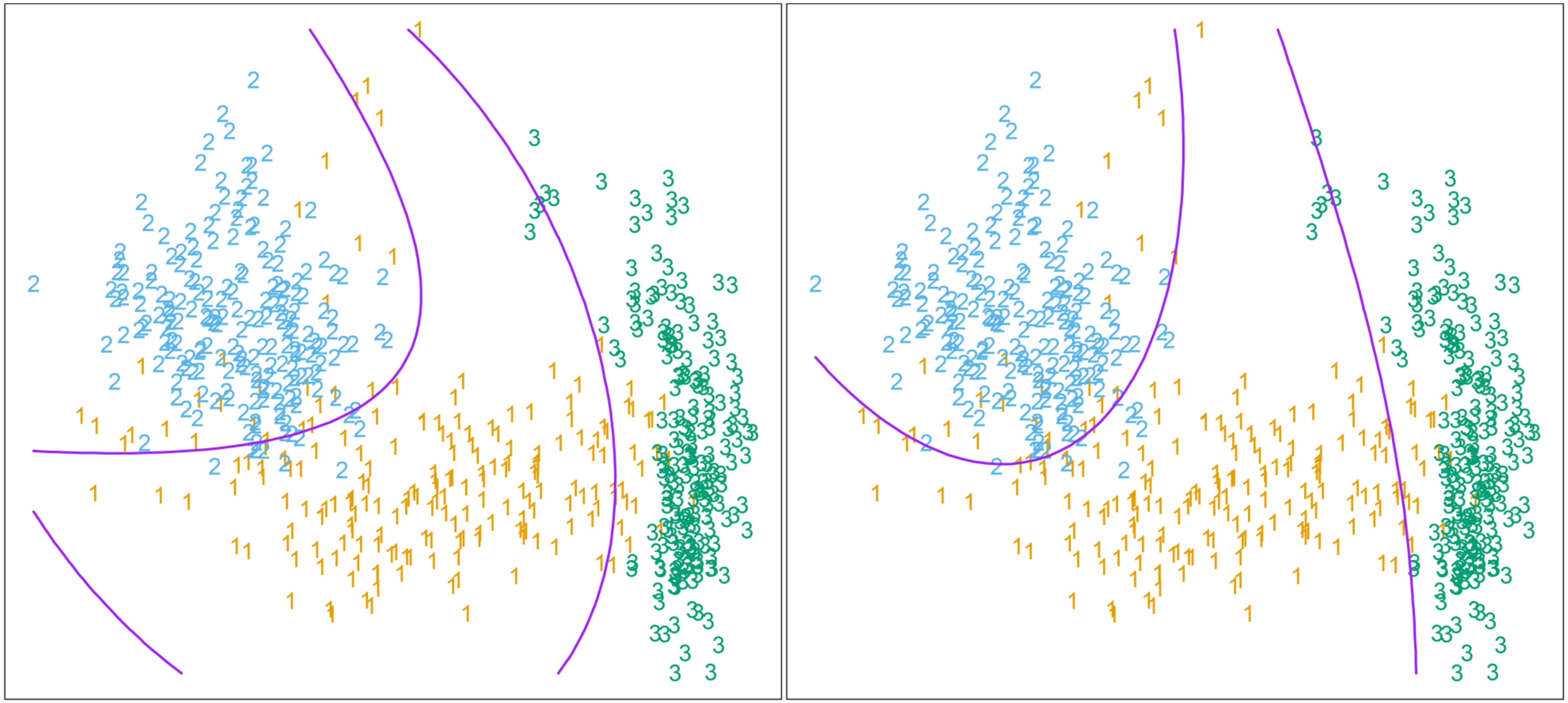
K π_k parameters

K μ_k parameters

$K \frac{p(p+1)}{2}$ $\mathbf{\Sigma}_k$ parameters, with p variance terms and $\binom{p}{2} = \frac{p(p-1)}{2}$ covariance terms for different features.

QDA behaves similar to LDA with expanded dimension

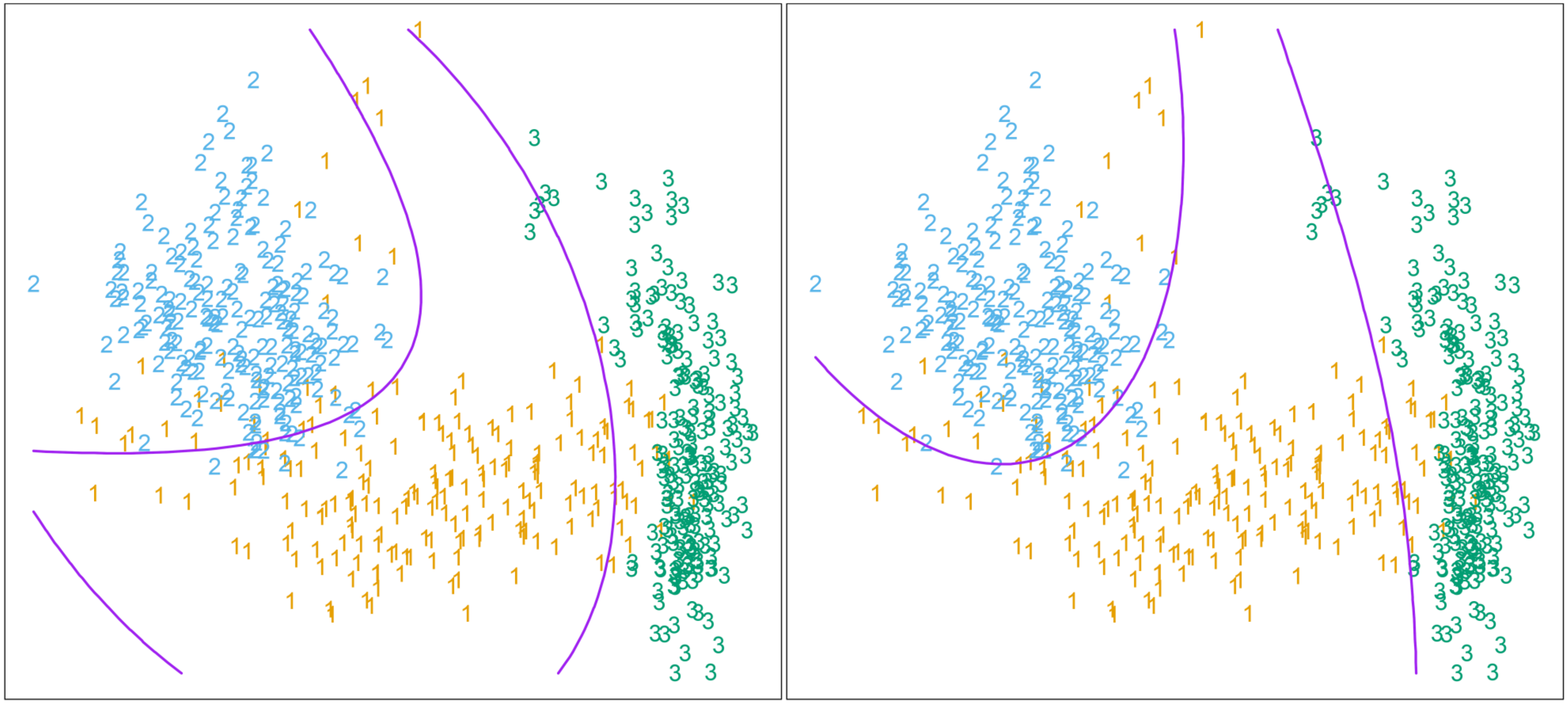
Left displays the linear decision boundaries from LDA applied to training data by expanding the features to a 5-dimensional space of X_1, X_2, X_1X_2, X_1^2, and X_2^2.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Expanding the feature dimension can help

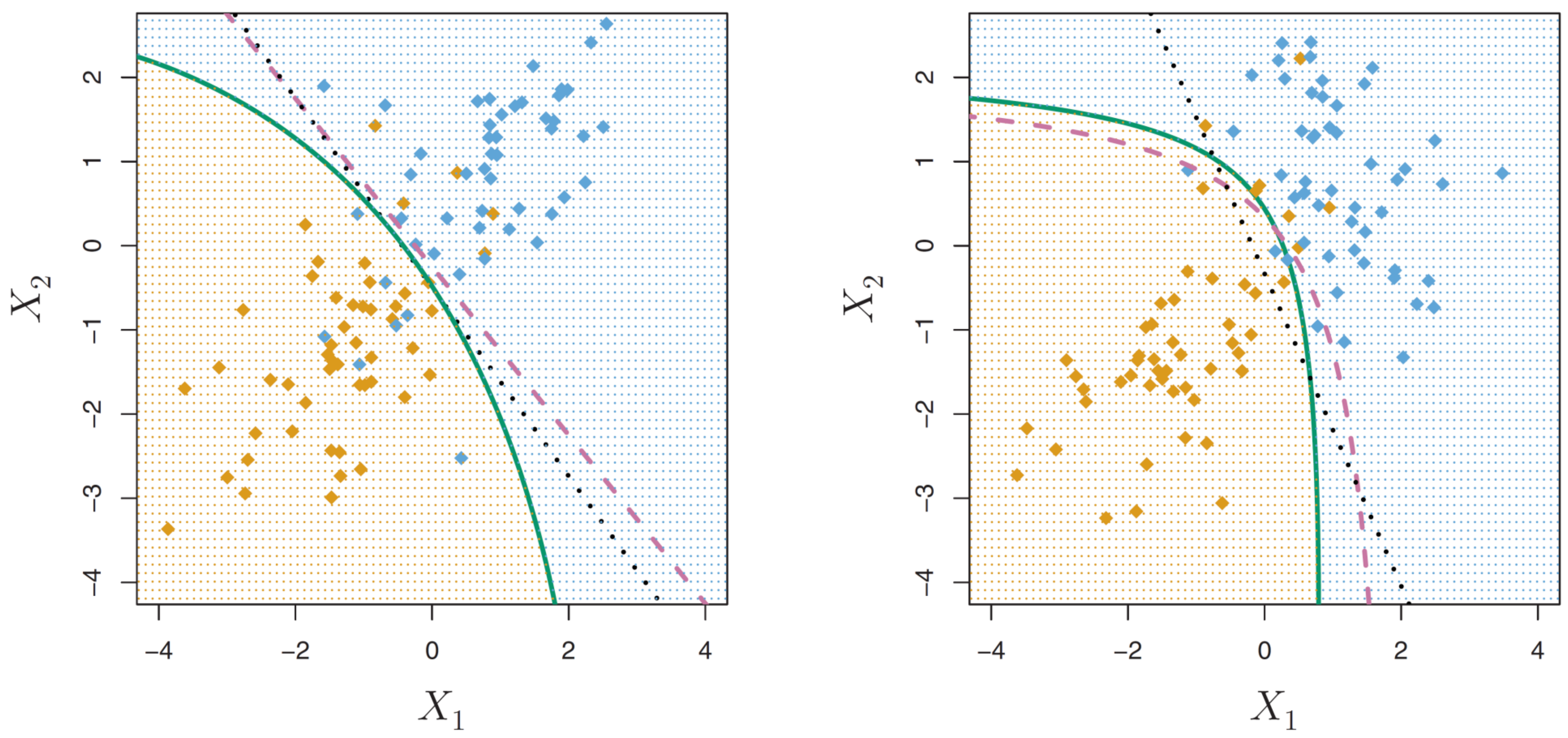
Right displays the quadratic decision boundaries from QDA applied to training data with 2 features X_1 and X_2, revealing similar classification regions as LDA applied to features expanded to higher dimensions.



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Side-by-side comparison of LDA and QDA

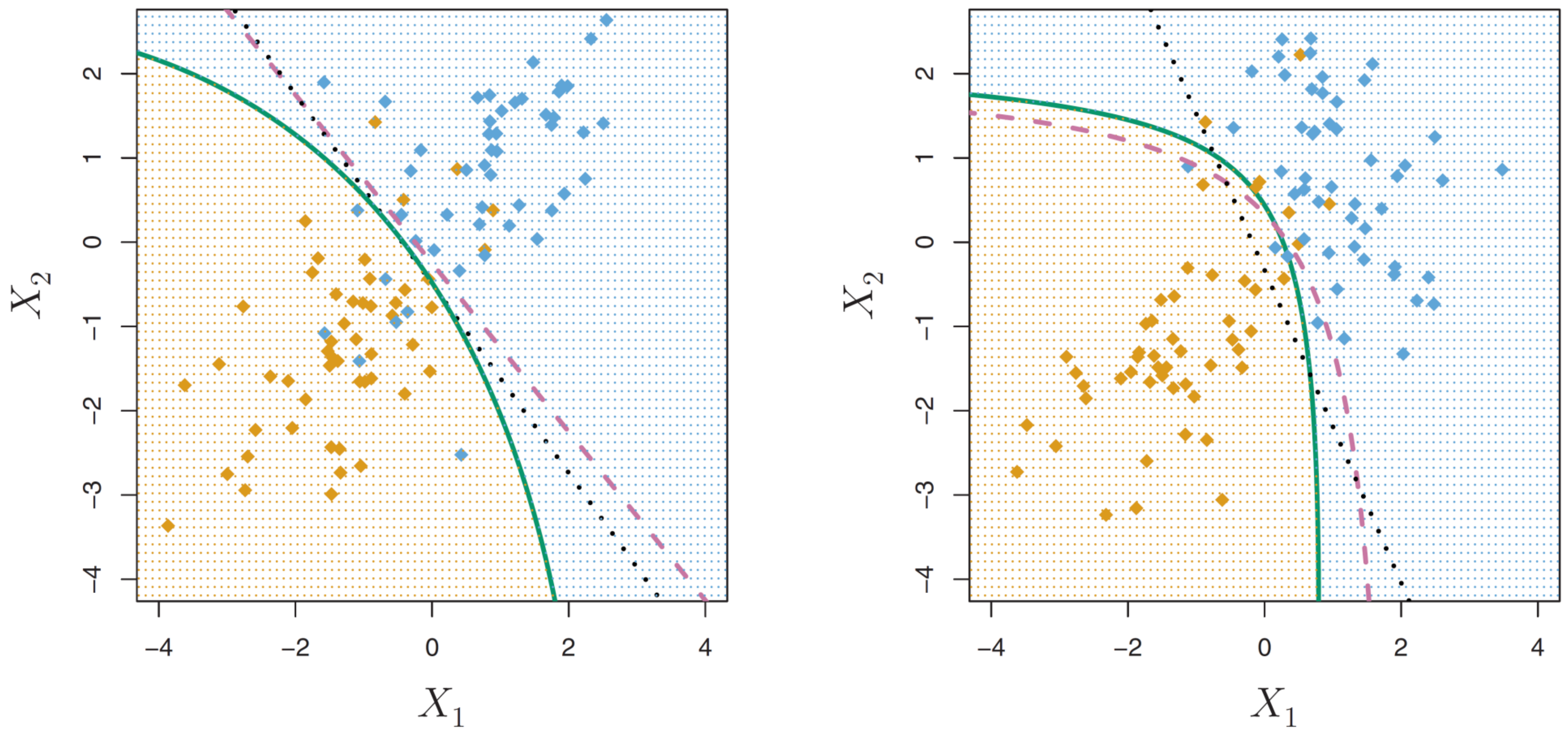
Left displays LDA (black dotted line) and QDA (green curve) decision boundaries for 2 classes in which $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$. That is, the true decision boundary would be linear.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Side-by-side comparison of LDA and QDA

Right displays LDA (black dotted line) and QDA (green curve) decision boundaries for 2 classes in which $\mathbf{\Sigma}_1 \neq \mathbf{\Sigma}_2$. That is, the true decision boundary would be non-linear.



Error rates for indicator linear regression, LDA, and QDA

Data contains $K = 11$ classes with $p = 10$ features, of which 3 of the features account for 90% of the variance (as computed by PCA).

Indicator linear regression is affected by masking, leading to large increases in both training and testing errors relative to LDA and QDA.

QDA is highly flexible leading to small training error rates, but not substantial improvement relative to LDA for test error rates.

Technique	Error Rates	
	Training	Test
Linear regression	0.48	0.67
Linear discriminant analysis	0.32	0.56
Quadratic discriminant analysis	0.01	0.53

Adapted from Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Choosing between LDA and QDA

How do we choose between LDA and QDA?

If the classes have equal covariance matrices ($\mathbf{\Sigma}_k = \mathbf{\Sigma}_\ell$ for $k \neq \ell$), then LDA would be ideal as it makes this assumption and has $(K - 1)p(p+1)/2$ fewer parameters to estimate. This model is constrained relative to QDA and has greater bias but less variance.

If the classes have unequal variances ($\mathbf{\Sigma}_k \neq \mathbf{\Sigma}_\ell$ for $k \neq \ell$), then QDA may be the better option, as LDA would require expanding the input dimension to obtain accurate classifications. This model is more flexible relative to LDA and has less bias but greater variance.

It would be better if we could somehow automatically tradeoff between the high bias-low variance of LDA and the low bias-high variance of QDA.

Regularized discriminant analysis (QDA-LDA tradeoff)

We can make a tradeoff between LDA and QDA by introducing a tuning parameter $\alpha \in [0,1]$ that represents the proportion of the model deriving from QDA.

That is, we model the covariance as a function of α, with estimated covariance matrix for class k as

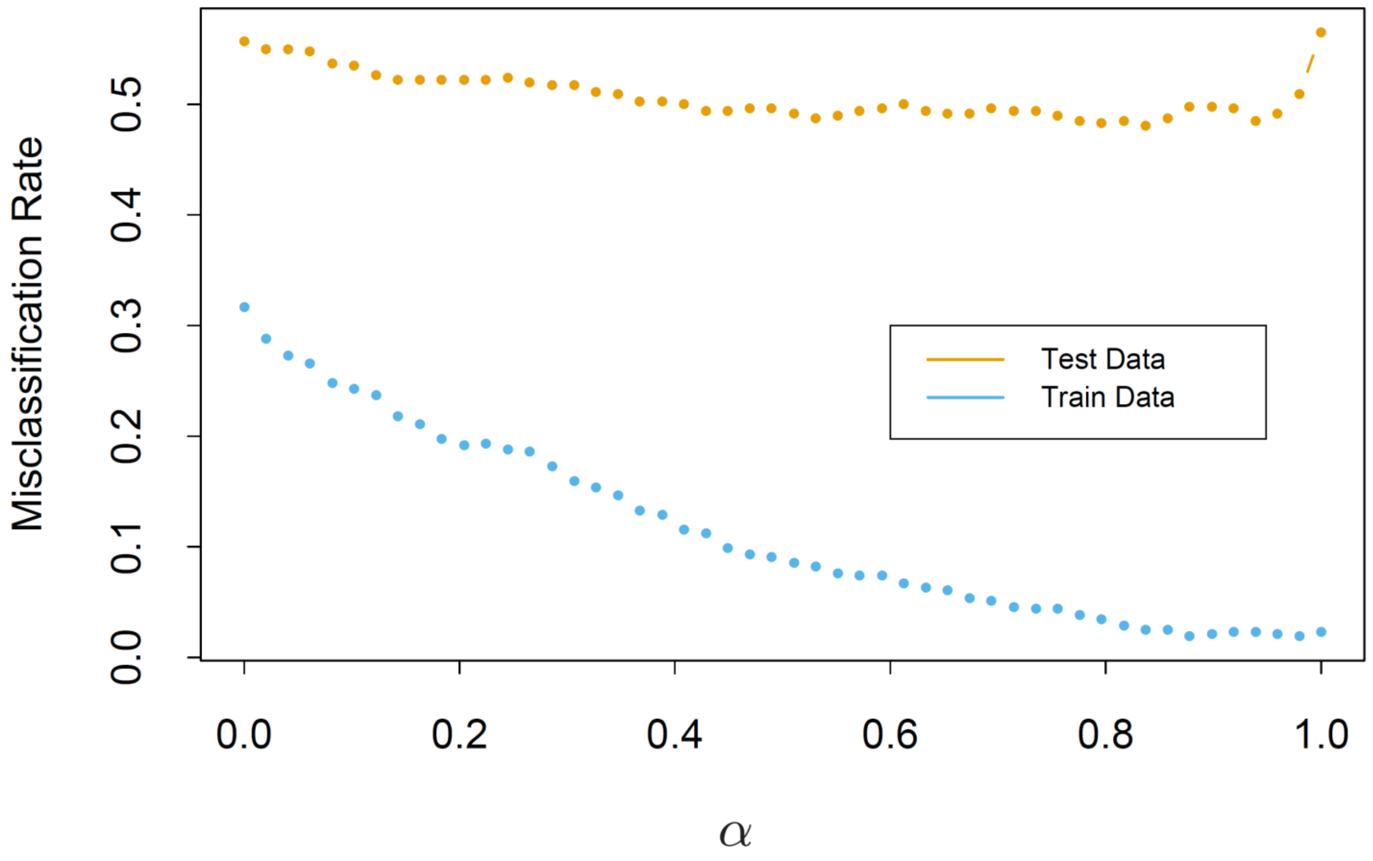
$$\hat{\boldsymbol{\Sigma}}_k(\alpha) = \alpha\hat{\boldsymbol{\Sigma}}_k + (1-\alpha)\hat{\boldsymbol{\Sigma}}$$

Here, when $\alpha = 1$, we have $\hat{\boldsymbol{\Sigma}}_k(1) = \hat{\boldsymbol{\Sigma}}_k$, which is the QDA solution, providing increased model flexibility with reduced bias. In contrast, when $\alpha = 0$, we have $\hat{\boldsymbol{\Sigma}}_k(0) = \hat{\boldsymbol{\Sigma}}$, which is the LDA solution, providing decreased model flexibility with reduced variance.

The best model likely lies at intermediate α.

Best model may be intermediate between LDA and QDA

Data contains $K = 11$ classes (colors) with $p = 10$ features, of which 3 of the features account for 90% of the variance (as computed by PCA). Regularized discriminant analysis was applied to this dataset, and the optimal α is approximately $\alpha = 0.9$ (close to QDA).



Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Simulation examples relating LDA, QDA, and RDA

Unequal highly ellipsoidal covariance matrices (*i.e.*, $\mathbf{\Sigma}_k \neq \mathbf{\Sigma}$) across classes $k \in \{1,2,3\}$ with nonzero mean differences.

RDA outperforms LDA and QDA for moderate and large covariance matrices, with QDA performing slightly better than RDA for small covariance matrices.

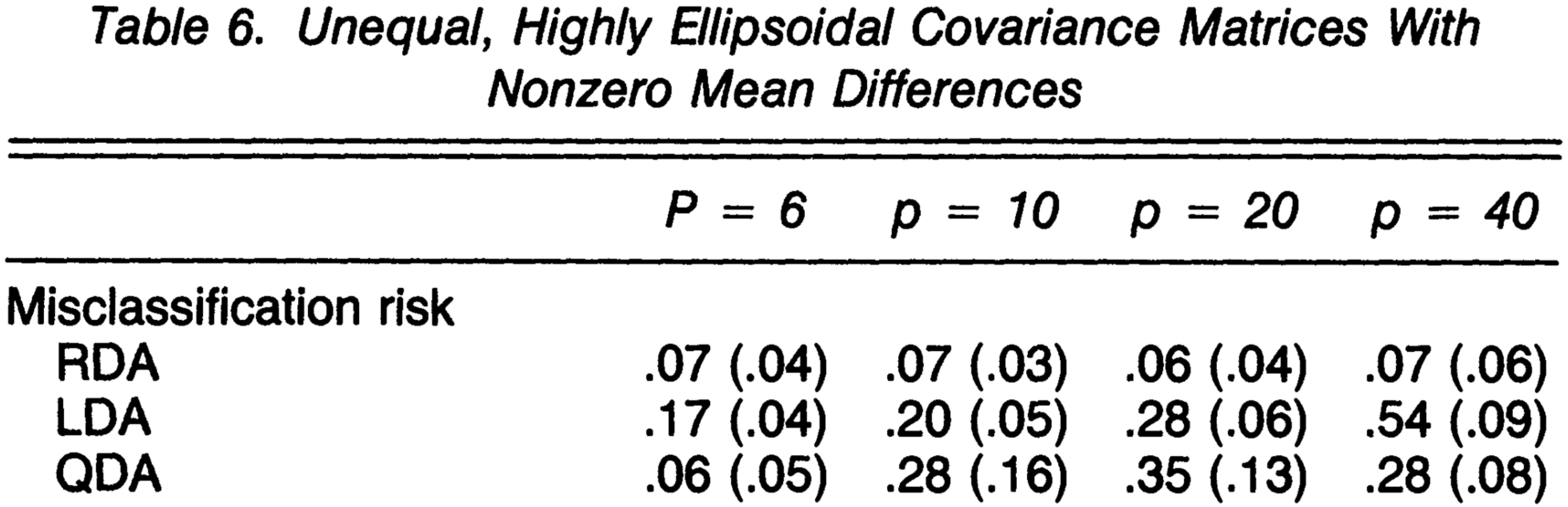


Table 6. Unequal, Highly Ellipsoidal Covariance Matrices With Nonzero Mean Differences

	$P = 6$	$p = 10$	$p = 20$	$p = 40$
Misclassification risk				
RDA	.07 (.04)	.07 (.03)	.06 (.04)	.07 (.06)
LDA	.17 (.04)	.20 (.05)	.28 (.06)	.54 (.09)
QDA	.06 (.05)	.28 (.16)	.35 (.13)	.28 (.08)

Adapted from Friedman (1989) *J Am Stat Assoc* 84:165-175.

Some important points about discriminant analysis

The observations are assumed to be normally (Gaussian) distributed, which can lead to increased classification accuracy over other methods if this assumption is true, but can also lead to decreased accuracy if this assumption is violated.

The assumption of a particular probability distribution can be helpful if the number N of training observations is small.

Because the input is assumed Gaussian, the features are considered quantitative, and the use of qualitative features (*e*.*g*., dummy variables) is not compatible with this model formulation.

However, the literature shows that LDA and QDA can still perform well when including such variables.

Chance of overfitting increases with increasing p/N ratio (regularize).