POLITECNICO
MILANO 1863

The Bayes Classifier (and KNN)

Matteo Matteucci, PhD (matteo.matteucci@polimi.it)

Artificial Intelligence and Robotics Laboratory
Politecnico di Milano

AIRLAB
ARTIFICIAL INTELLIGENCE AND ROBOTICS LAB

What about Statistical Learning Theory for Classification?

For a classification problem we use the error rate for valuation

$$Error\ Rate = \sum_{i=1}^{n} I(y_i \neq \hat{y}_i)/n$$

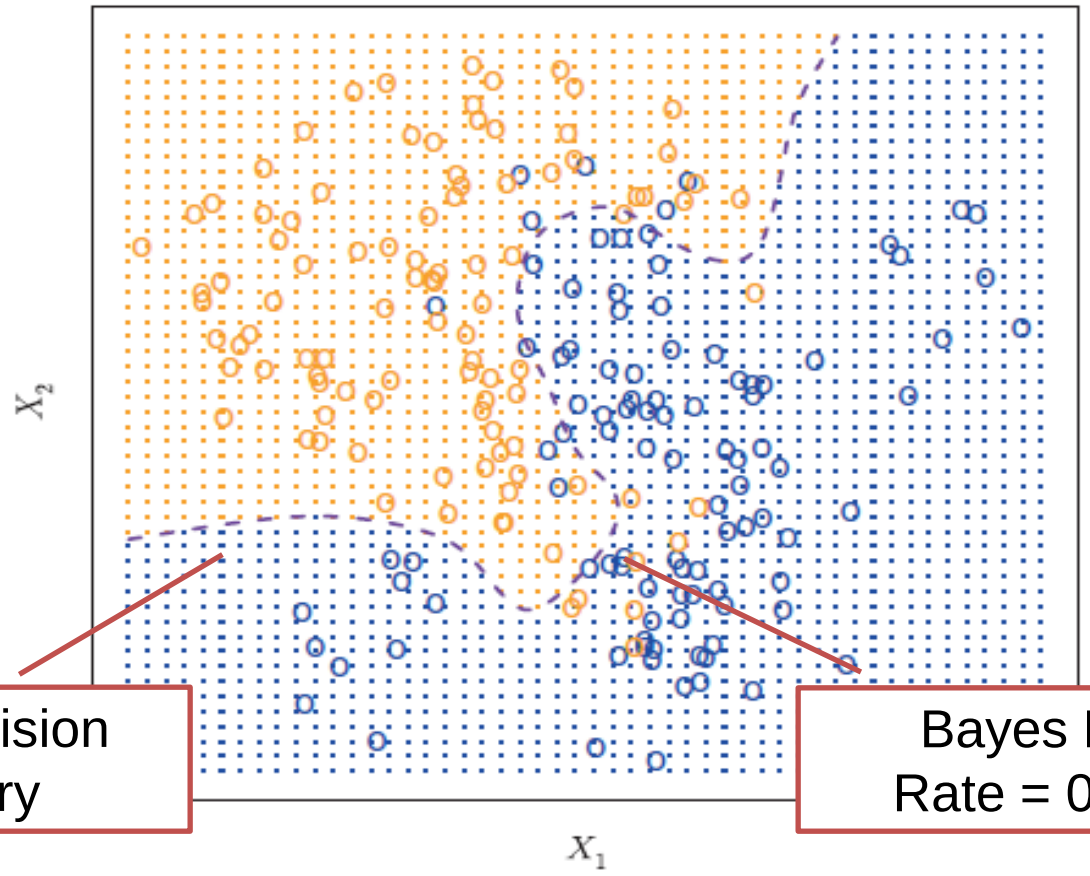
- Where $I(y_i \neq \hat{y}_i)$ is an indicator function, which will give 1 if the condition $(y_i \neq \hat{y}_i)$ is correct, otherwise it gives a 0.
- Represents the fraction of incorrect classifications, or misclassifications

The Bayes Classifier minimizes the Average Test Error Rate

$$\max_j P(Y = j \mid X = x_0)$$

The Bayes error rate refers to the lowest possible Error Rate achievable knowing the "true" distribution of the data: $1 - E\left(\max_j \Pr(Y = j|X)\right)$

Bayes Classifier



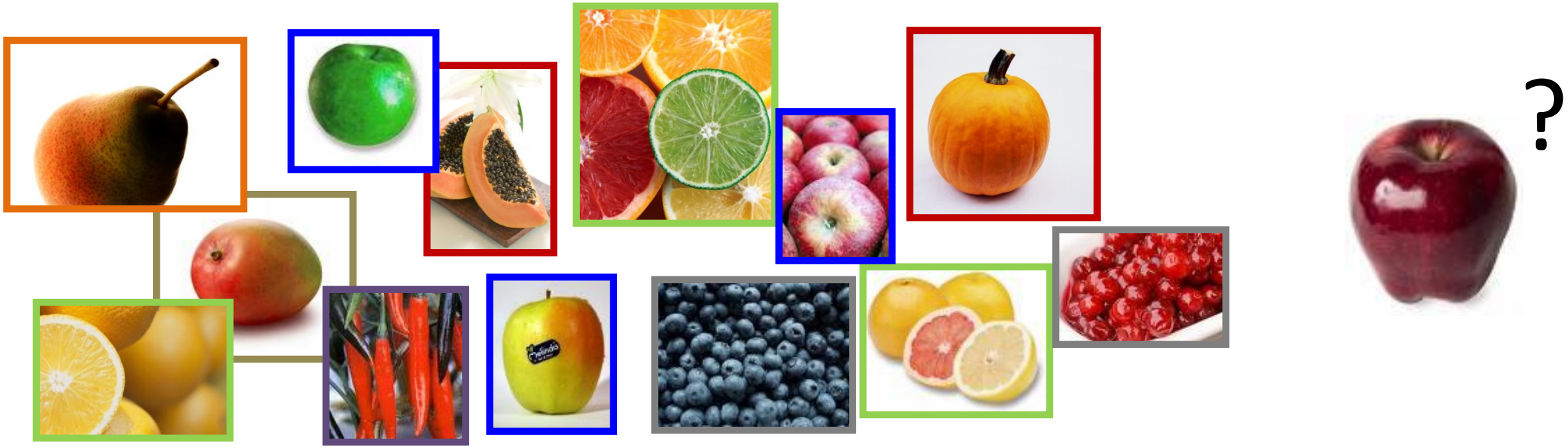
Bayes Decision Boundary

Bayes Error Rate = 0.1304

FIGURE 2.13. *A simulated data set consisting of* 100 *observations in each of two groups, indicated in blue and in orange. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which a test observation will be assigned to the orange class, and the blue background grid indicates the region in which a test observation will be assigned to the blue class.*

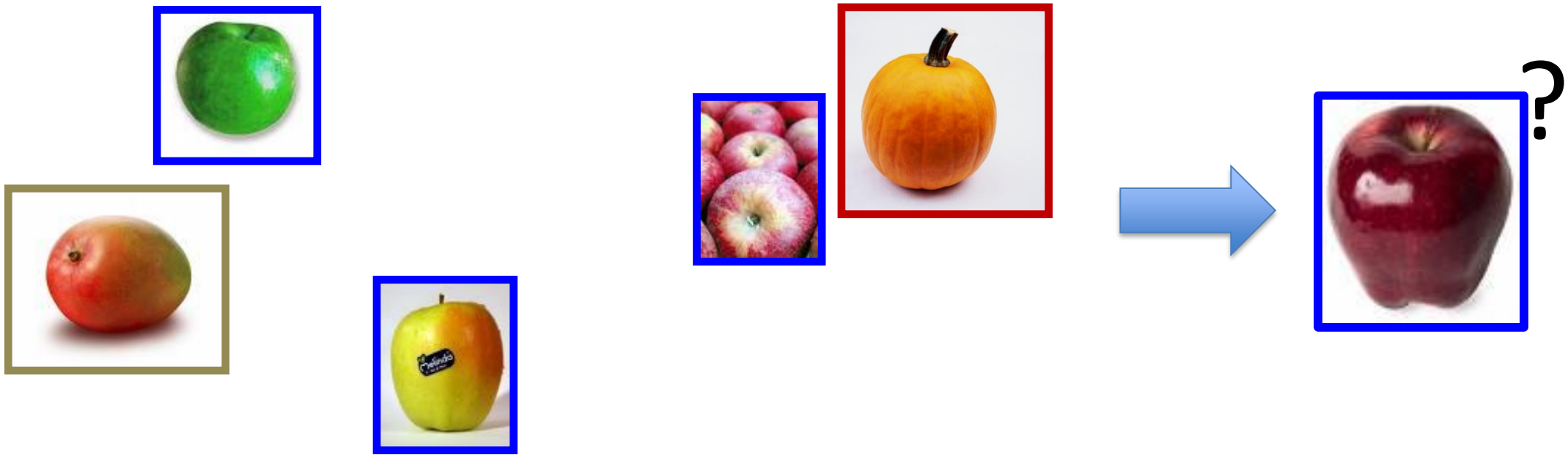
Nearest Neighbors Classifier

To decide the label for an unseen example, consider the k examples (5) from the training data that are more similar to the unknown one



Nearest Neighbors Classifier

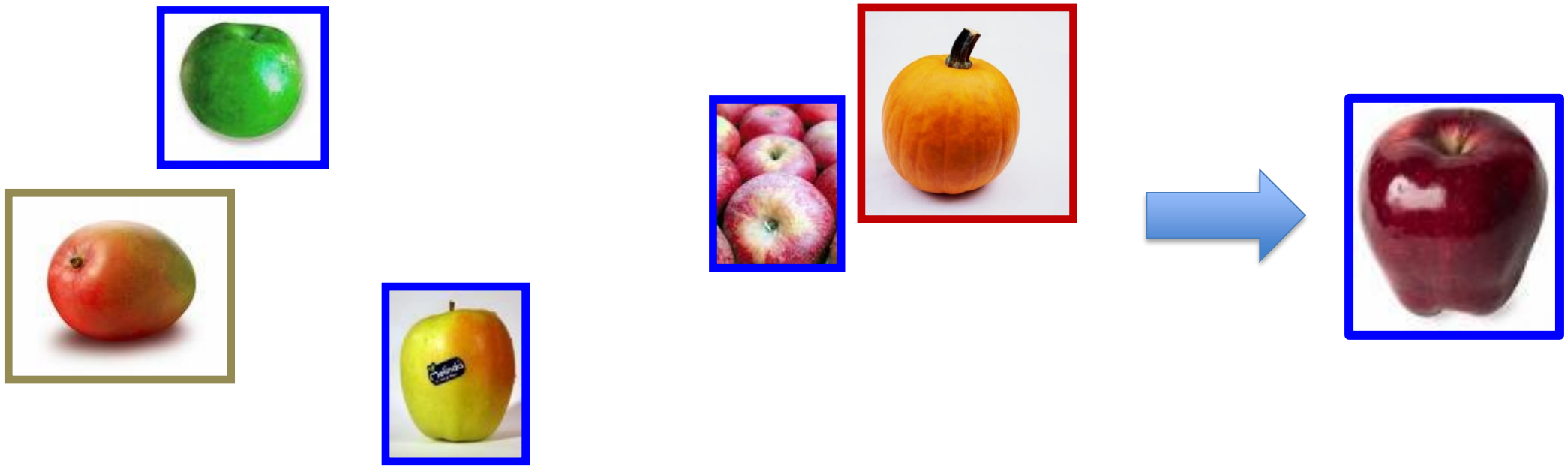
To decide the label for an unseen example, consider the k examples (5) from the training data that are more similar to the unknown one



Classify the unknown example using the most frequent class

Nearest Neighbors Classifier

To decide the label for an unseen example, consider the k examples (5) from the training data that are more similar to the unknown one



Classify the unknown example using the most frequent class

K-Nearest Neighbors (KNN)

The k Nearest Neighbors method is a non parametric model often used to approximate the Bayes Classifier

- For any given X we find the k closest neighbors to X in the training data, and examine their corresponding Y
- If the majority of the Y's are orange we predict orange otherwise guess blue.

Some notes about such a simple classifier …

- The smaller the k, the more flexible the method will be
- KNN has "zero" training time, some cost at runtime to find the k closest neighbors thus requires indexing
- KNN has problems in higher dimensional spaces (needs approximate methods)

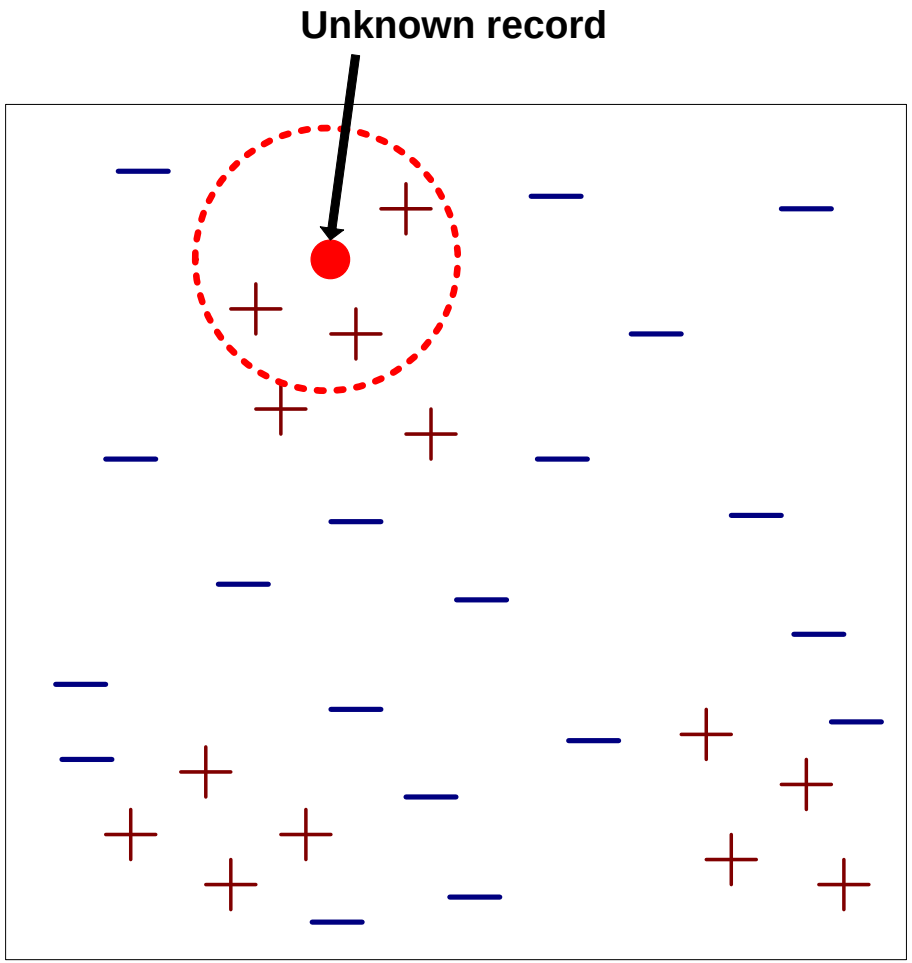
Nearest Neighbors Classifier

To decide the label for an unseen example via Neirest Neighbors classification you need:

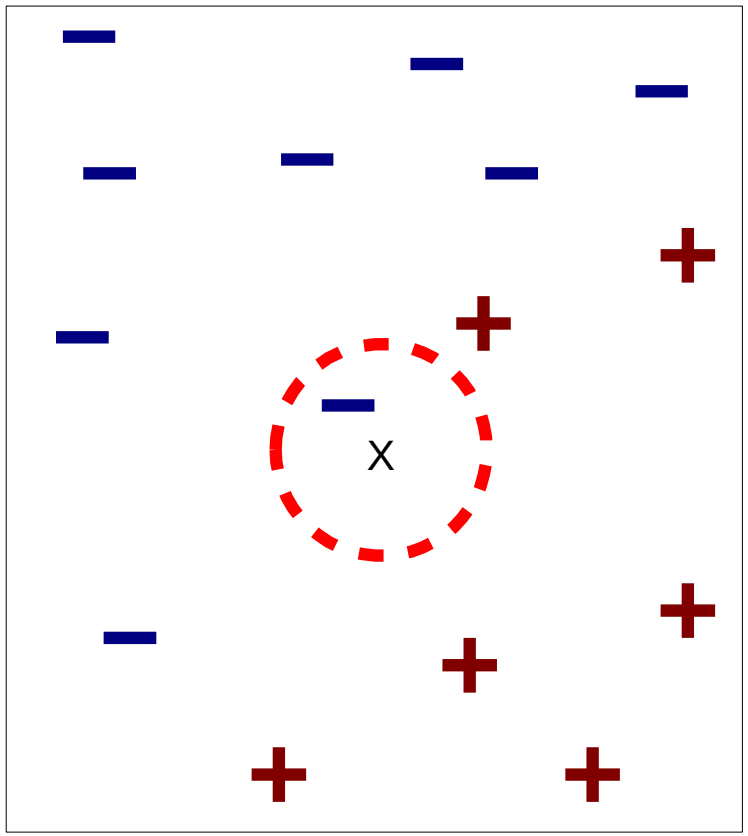
- The training dataset
- Similarity function (or distance metric)
- The value of k, of nearest neighbors to retrieve

Classification is then:

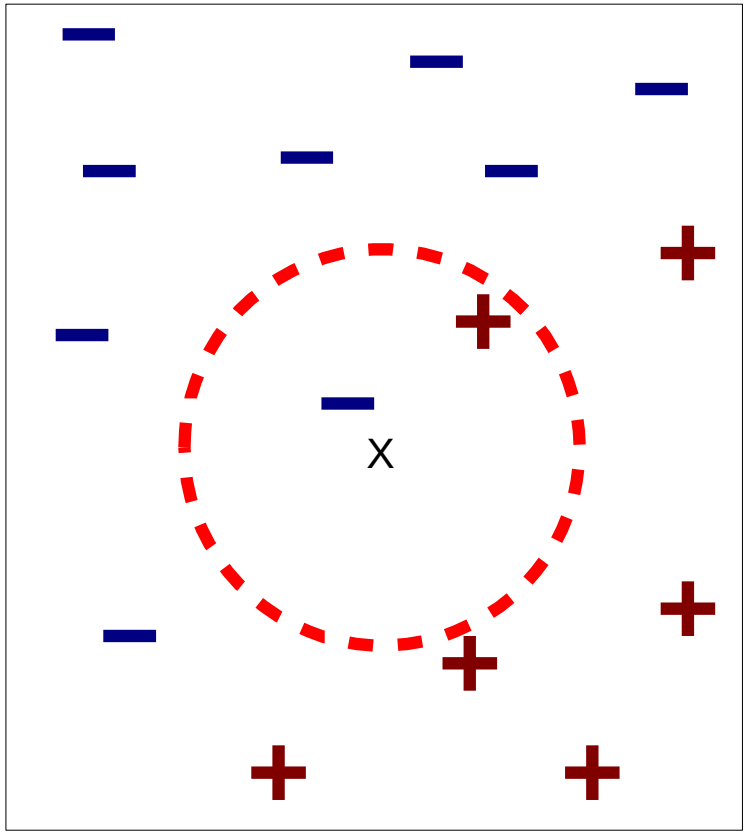
- Compute distance to other training records
- Identify the k nearest neighbors
- Use labels of nearest neighbors to determine the class of unknown record (e.g., by majority vote)



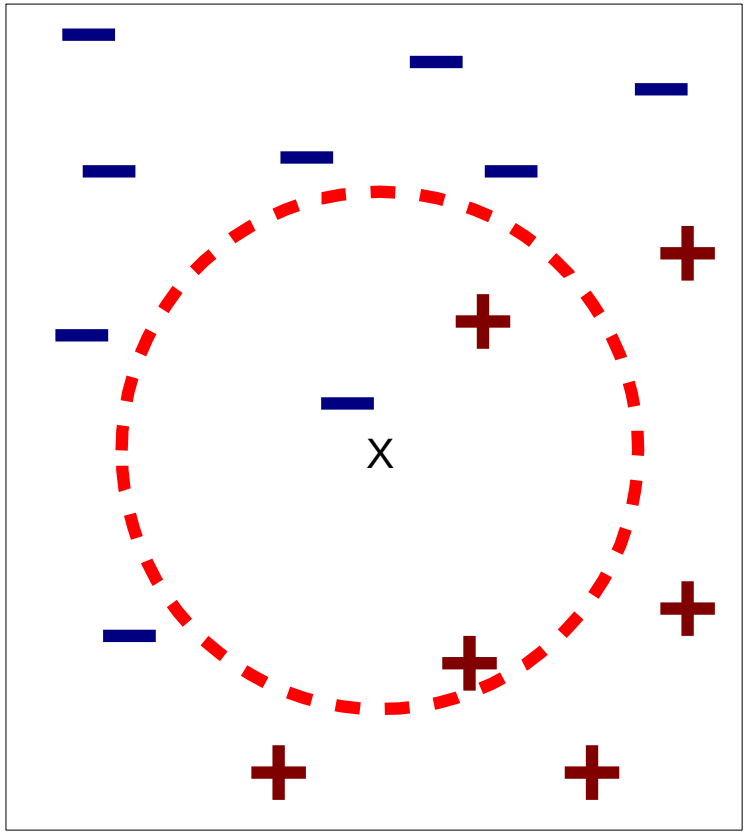
K-Nearest Neighbors Classifier



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

KNN Example with k = 3

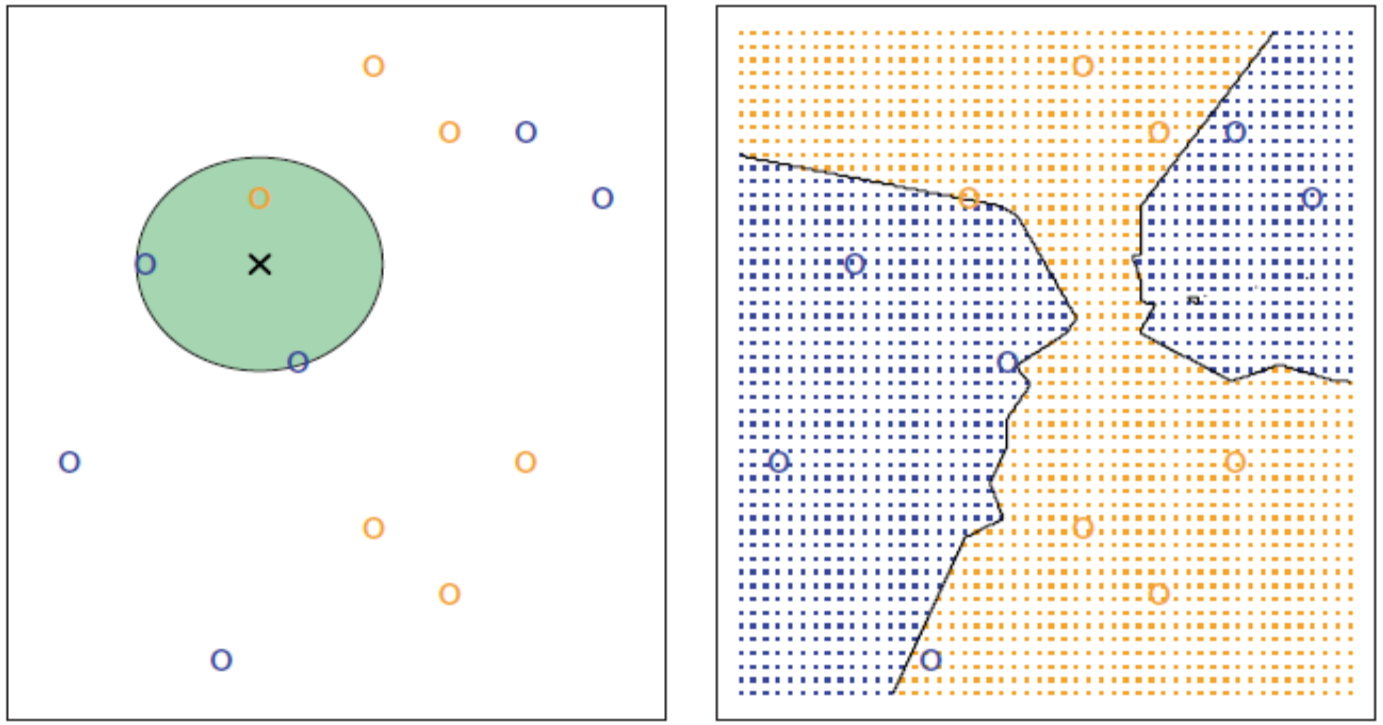


FIGURE 2.14. *The KNN approach, using $K = 3$, is illustrated in a simple situation with six blue observations and six orange observations.* Left: *a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue.* Right: *The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.*

Simulated Data: K = 10

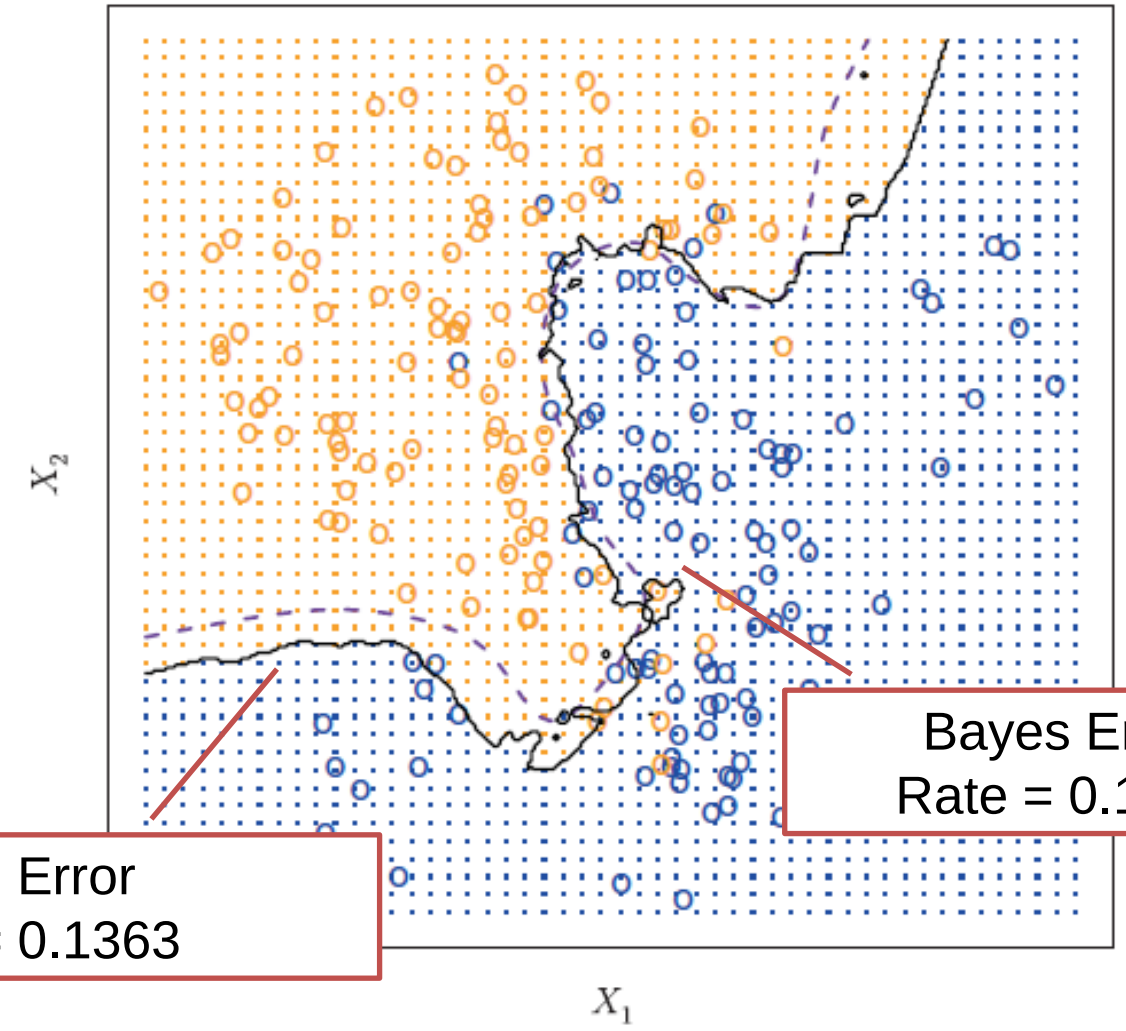


FIGURE 2.15. *The black curve indicates the KNN decision boundary on the data from Figure 2.13, using $K = 10$. The Bayes decision boundary is shown as a purple dashed line. The KNN and Bayes decision boundaries are very similar.*

K = 1 and K = 100

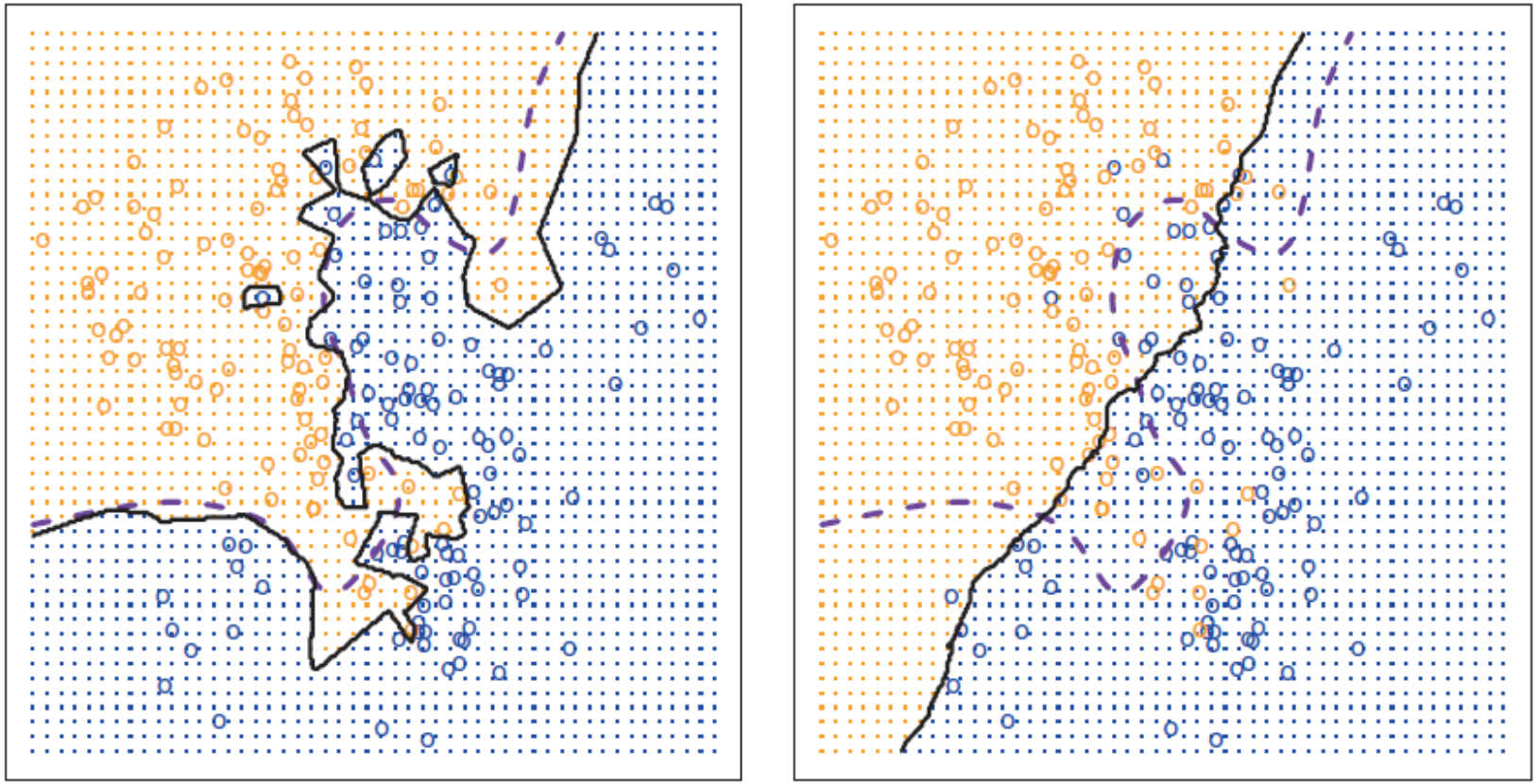


FIGURE 2.16. *A comparison of the KNN decision boundaries (solid black curves) obtained using $K = 1$ and $K = 100$ on the data from Figure 2.13. With $K = 1$, the decision boundary is overly flexible, while with $K = 100$ it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.*

Training vs. Test Error Rates

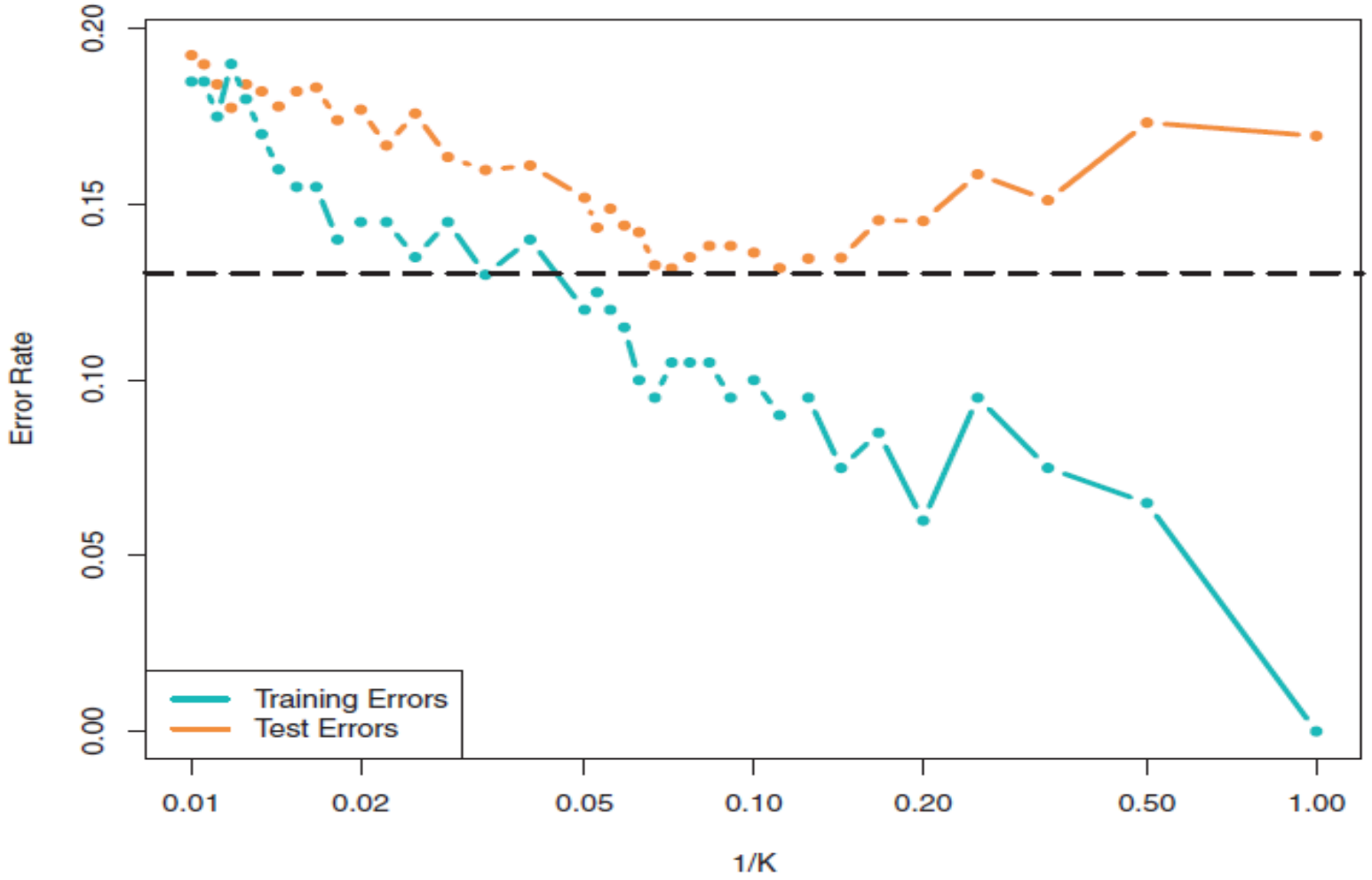


FIGURE 2.17. *The KNN training error rate (blue, 200 observations) and test error rate (orange, 5,000 observations) on the data from Figure 2.13, as the level of flexibility (assessed using $1/K$) increases, or equivalently as the number of neighbors K decreases. The black dashed line indicates the Bayes error rate.*

POLITECNICO
MILANO 1863

Linear Classifiers (Generative)

Matteo Matteucci, PhD (matteo.matteucci@polimi.it)

Artificial Intelligence and Robotics Laboratory
Politecnico di Milano

AIRLAB
ARTIFICIAL INTELLIGENCE AND ROBOTICS LAB

Example: Default dataset

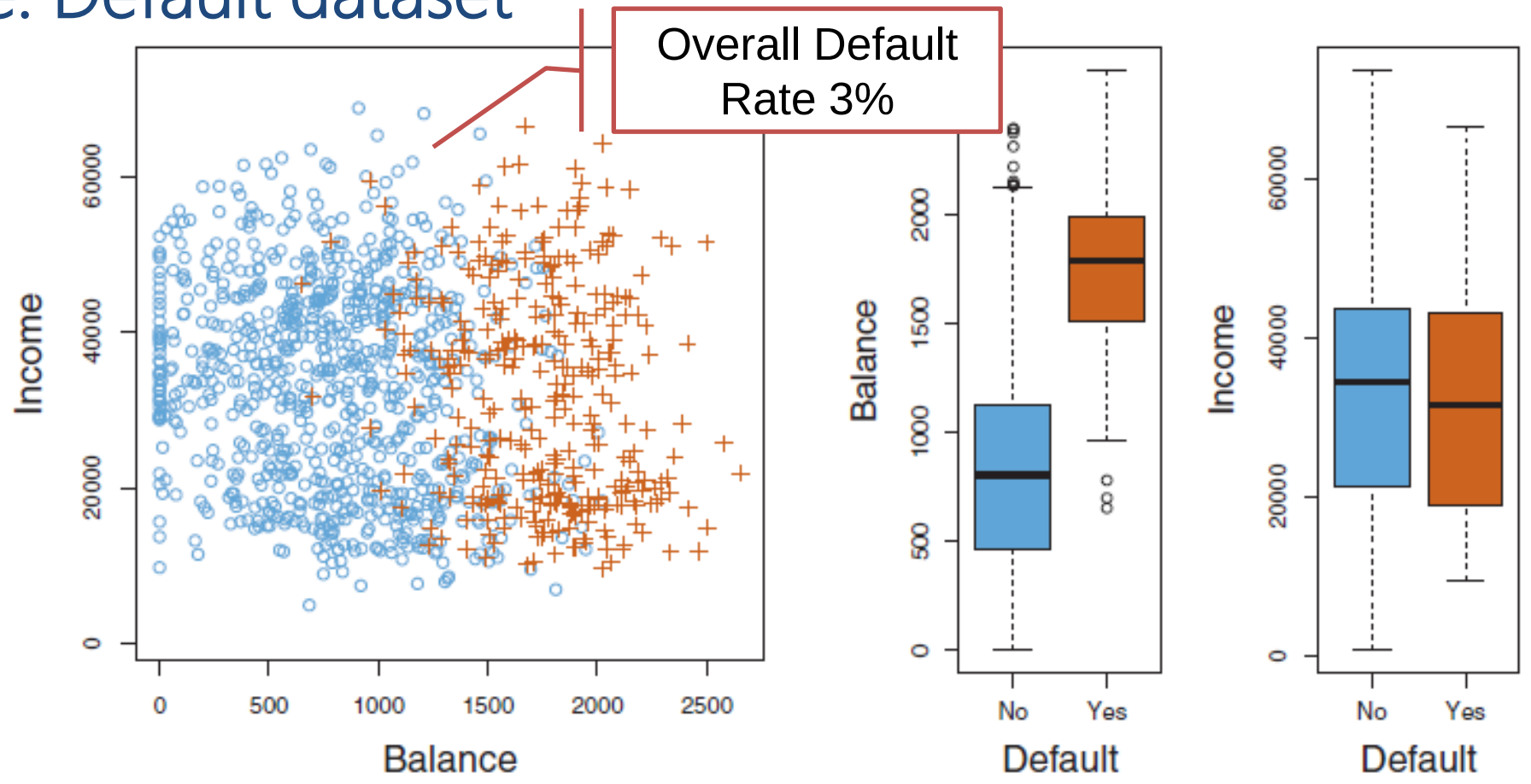


FIGURE 4.1. *The* `Default` *data set.* Left: *The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.* Center: *Boxplots of* `balance` *as a function of* `default` *status.* Right: *Boxplots of* `income` *as a function of* `default` *status.*

Recall Statistical Learning Theory for Classification

For a classification problem we use the error rate for valuation

$$Error\ Rate = \sum_{i=1}^{n} I(y_i \neq \hat{y}_i) / n$$

- Where $I(y_i \neq \hat{y}_i)$ is an indicator function, which will g… 1 if th… condition $(y_i \neq \hat{y}_i)$ is correct, otherwise it gives a 0.
- Represents the fraction of incorrect classifications, or m…

The best classifier possible estimates the class posterior probability!!

The Bayes Classifier minimizes the Average Test Error Rate

$$\max_j P(Y = j \mid X = x_0)$$

The Bayes error rate refers to the lowest possible Error Rate achievable knowing the "true" distribution of the data: $1 - E\left(\max_j \Pr(Y = j|X)\right)$

Logistic Regression

I know it should be written as P(Y|X) instead of p(X) ☹

We want to model the probability of the class given the input

$$p(X) = \Pr(Y = 1|X)$$

$$p(X) = \beta_0 + \beta_1 X$$

but a this naïve model has some serious drawbacks

Example: Default data & Linear Regression

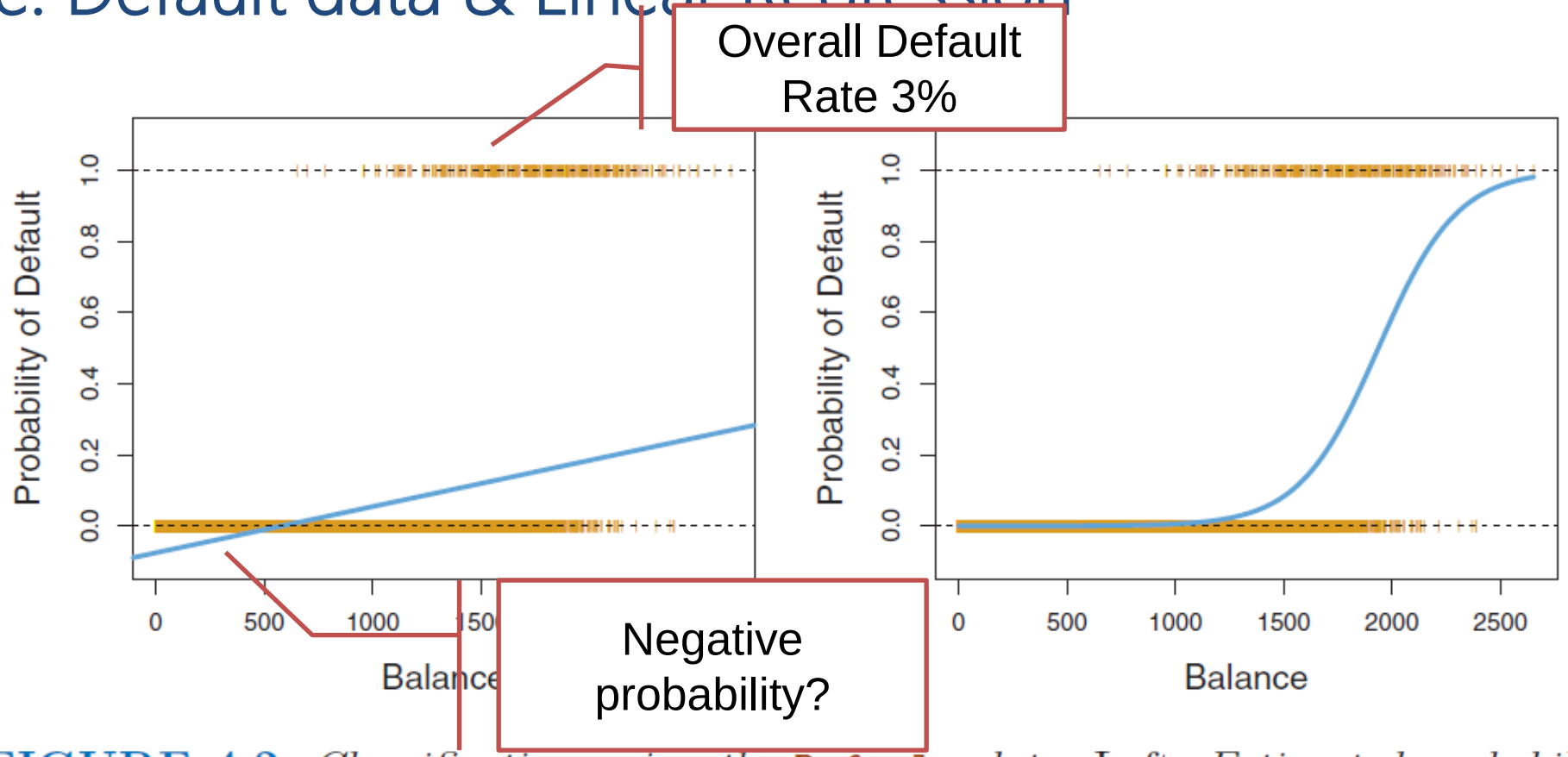


FIGURE 4.2. *Classification using the* `Default` *data.* Left: *Estimated probability of* `default` *using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for* `default` *(*`No` *or* `Yes`*).* Right: *Predicted probabilities of* `default` *using logistic regression. All probabilities lie between* 0 *and* 1.

Logistic Regression

We want to model the probability of the class given the input

$$p(X) = \Pr(Y = 1|X)$$

$$p(X) = \beta_0 + \beta_1 X$$

but a this naïve model has some serious drawbacks

Logistic regression solves the negative probability (and other issues as well) by regressing the logistic function

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

Example: Default data & Logistic Regression

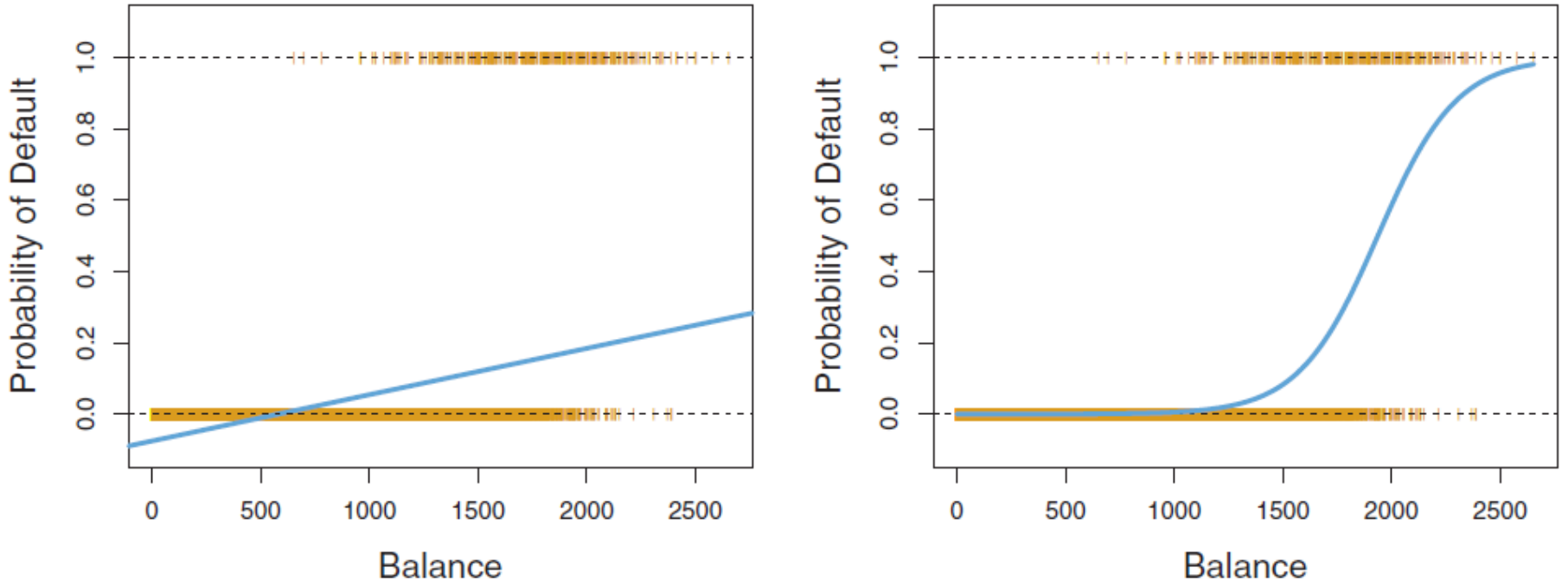


FIGURE 4.2. *Classification using the* `Default` *data.* Left: *Estimated probability of* `default` *using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for* `default` *(*`No` *or* `Yes`*).* Right: *Predicted probabilities of* `default` *using logistic regression. All probabilities lie between* 0 *and* 1.

Logistic Regression

Logistic regression solves the negative probability (and other issues as well) by regressing the logistic function

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

Logistic Regression

from this we derive

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X}$$

This is called *odds*

and taking logarithms

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

This is called
log-odds or logit

Coefficient interpretation

Interpreting what β_1 means is not very easy with logistic regression, simply because we are predicting P(Y) and not Y.

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

- If $\beta_1 = 0$, this means that there is no relationship between Y and X
- If $\beta_1 > 0$, this means that when X gets larger so does the probability that Y = 1
- If $\beta_1 < 0$, this means that when X gets larger, the probability Y = 1 gets smaller.

But how much bigger or smaller depends on where we are on the slope, i.e., it is not linear

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

Training Logistic Regression (1/4)

For the basic logistic regression we need two parameters

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

In principle we could use (non linear) Least Squares fitting on the observed data the corresponding model

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

But a more principled approach for training in classification problems is based on Maximum Likelihood

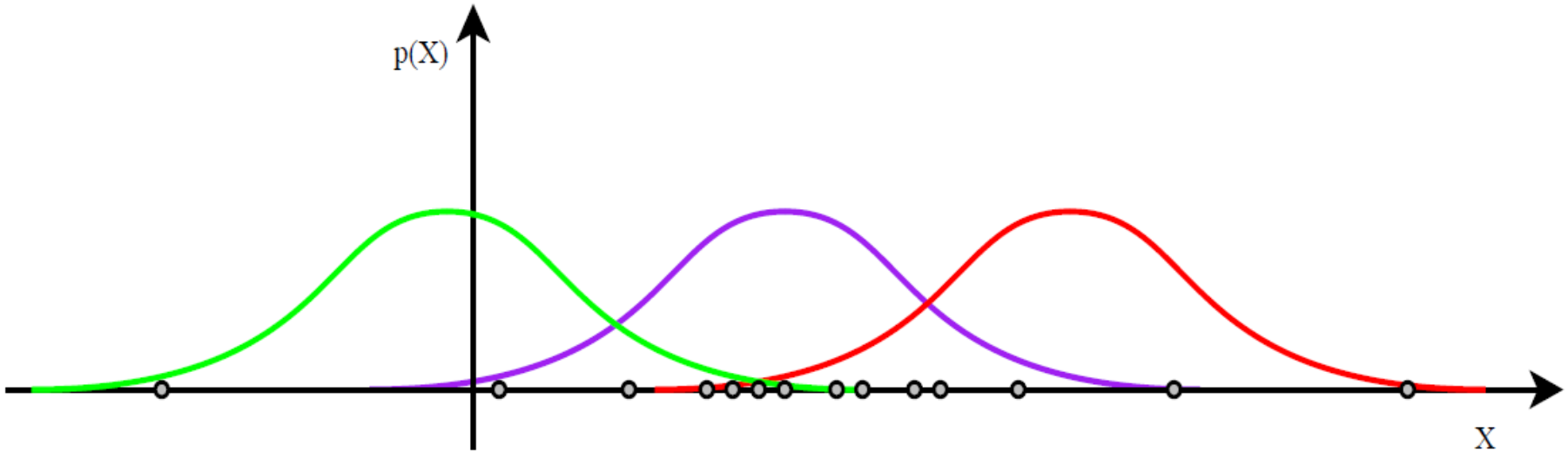
- We want to find the parameters which maximize the likelihood function

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1-p(x_{i'}))$$

Maximum Likelihood flash-back (1/6)

Suppose we observe some i.i.d. samples coming from a Gaussian distribution with known variance:

$$x_1, x_2, \ldots, x_K \sim N(\mu, \sigma^2) \qquad p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$



Which distribution do you prefer?

Maximum Likelihood flash-back (2/6)

There is a simple recipe for Maximum Likelihood estimation

1. Write the likelihood $L = P(Data|\theta)$ for the data
2. (Take the logarithm of likelihood $\mathcal{L} = \log P(Data|\theta)$)
3. Work out $\partial L/\partial\theta$ or $\partial\mathcal{L}/\partial\theta$ using high-school calculus
4. Solve the set of simultaneous equations $\partial\mathcal{L}/\partial\theta_i = 0$
5. Check that θ^{mle} is a maximum

Let's try to apply it to our example

$$x_1, x_2, \ldots, x_K \sim N(\mu, \sigma^2) \qquad p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

Maximum Likelihood flash-back (3/6)

Let's try to apply it to our example

$$x_1, x_2, \ldots, x_K \sim N(\mu, \sigma^2) \qquad p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

1. Write le likelihood for the data

$$L(\mu) = p(x_1, x_2, \ldots, x_N|\mu, \sigma^2) = \prod_{n=1}^{N} p(x_n|\mu, \sigma^2)$$

$$= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_n-\mu)^2}{2\sigma^2}}$$

Maximum Likelihood flash-back (4/6)

Let's try to apply it to our example

$$x_1, x_2, \ldots, x_K \sim N(\mu, \sigma^2) \qquad p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

2. (Take the logarithm of the likelihood -> log-likelihood)

$$\begin{aligned}
\mathcal{L} &= log \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\
&= \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x_n-\mu)^2}{2\sigma^2}) \\
&= N(\log \frac{1}{\sqrt{2\pi}\sigma}) - \frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2
\end{aligned}$$

Maximum Likelihood flash-back (5/6)

Let's try to apply it to our example

$$x_1, x_2, \ldots, x_K \sim N(\mu, \sigma^2) \qquad p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

3. Work out the derivatives using high-school calculus

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mu} &= \frac{\partial}{\partial \mu} N(\log \frac{1}{\sqrt{2\pi}\sigma}) - \frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2 \\
&= -\frac{1}{2\sigma^2} \frac{\partial}{\partial \mu} \sum_{n=1}^{N} (x_n - \mu)^2 = \\
&= \frac{1}{2\sigma^2} \sum_{n=1}^{N} 2(x_n - \mu)
\end{aligned}$$

Maximum Likelihood flash-back (6/6)

Let's try to apply it to our example

$$x_1, x_2, \ldots, x_K \sim N(\mu, \sigma^2) \qquad p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

4. Solve the unconstrained equations $\partial \mathcal{L} / \partial \theta_i = 0$

$$\frac{1}{2\sigma^2} \sum_{n=1}^{N} 2(x_n - \mu) = 0$$

$$\sum_{n=1}^{N} (x_n - \mu) = 0$$

$$\sum_{n=1}^{N} x_n = \sum_{n=1}^{N} \mu$$

$$\mu_{MLE} = \frac{1}{N} \sum x_n$$

Training Logistic Regression (1/4)

For the basic logistic regression we need two parameters

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

In principle we could use (non linear) Least Squares fitting on the observed data the corresponding model

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

But a more principled approach for training in classification problems is based on Maximum Likelihood

- We want to find the parameters which maximize the likelihood function

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1-p(x_{i'}))$$

Training Logistic Regression (2/4)

Let's find the parameters which maximize the likelihood function

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

If we compute the log-likelihood for N observations

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta)$$

Taken from ESL

where $p_k(x_i; \theta) = \Pr(G = k | X = x_i; \theta)$

Can you derive it?

We obtain a log-likelihood in the form of

$$\begin{aligned} \ell(\beta) &= \sum_{i=1}^{N} \Big\{ y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta)) \Big\} \\ &= \sum_{i=1}^{N} \Big\{ y_i \beta^T x_i - \log(1 + e^{\beta^T x_i}) \Big\} . \end{aligned}$$

Training Logistic Regression (3/4)

Let's find the parameters which maximize the likelihood function

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

- Z-statistics has the same role of the regression t-statistics, a large value means the parameter is not null
- Intercept does not have a particular meaning is used to adjust the probability to class proportions

	Coefficient	Std. error	Z-statistic	P-value
`Intercept`	−10.6513	0.3612	−29.5	<0.0001
`balance`	0.0055	0.0002	24.9	<0.0001

TABLE 4.1. *For the* `Default` *data, estimated coefficients of the logistic regression model that predicts the probability of* `default` *using* `balance`*. A one-unit increase in* `balance` *is associated with an increase in the log odds of* `default` *by* 0.0055 *units.*

Training Logistic Regression (4/4)

Let's find the parameters which maximize the likelihood function

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

- We can train the model using qualitative variables through the use of binary (dummy) variables

	Coefficient	Std. error	Z-statistic	P-value
`Intercept`	−3.5041	0.0707	−49.55	<0.0001
`student[Yes]`	0.4049	0.1150	3.52	0.0004

TABLE 4.2. *For the* `Default` *data, estimated coefficients of the logistic regression model that predicts the probability of* `default` *using student status. Student status is encoded as a dummy variable, with a value of* 1 *for a student and a value of* 0 *for a non-student, and represented by the variable* `student[Yes]` *in the table.*

Multiclass Logistic Regression

Logistic Regression extends naturally to multiclass problems by computing the log-odds w.r.t. the K[th] class

$$\log \frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)} = \beta_{10} + \beta_1^T x$$

$$\log \frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)} = \beta_{20} + \beta_2^T x$$

$$\vdots$$

$$\log \frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)} = \beta_{(K-1)0} + \beta_{K-1}^T x$$

Comes from ESL, but it's worth knowing!!!

Notation different because it comes from ESL

This is equivalent to

$$\Pr(G=k|X=x) = \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \beta_\ell^T x)}, \quad k = 1, \ldots, K-1,$$

$$\Pr(G=K|X=x) = \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \beta_\ell^T x)}, \qquad (4.18)$$

Making predictions with Logistic Regression

Once we have the model parameters we can predict the class, and it probability, e.g., the Default probability having 1000$ balance is <1%

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1{,}000}}{1+e^{-10.6513+0.0055\times 1{,}000}} = 0.00576$$

while with a balance of 2000$ this becomes 58.6%

With qualitative variables, i.e., dummy variables, we get that being a students results in

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431$$

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292$$

Multiple Logistic Regression

We can extend the approach to multiple regressors

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

	Coefficient	Std. error	Z-statistic	P-value
`Intercept`	−10.8690	0.4923	−22.08	<0.0001
`balance`	0.0057	0.0002	24.74	<0.0001
`income`	0.0030	0.0082	0.37	0.7115
`student[Yes]`	−0.6468	0.2362	−2.74	0.0062

TABLE 4.3. *For the* `Default` *data, e… the logistic regression model that predicts the probability … `balance`, `income`, and student status. Student status is encoded as a dummy variable* `student[Yes]`*, with a value of 1 for a student and a value of 0 for a non-student. In fitting this model,* `income` *was measured in thousands of dollars.*

What about this?

Example: South African Heart Disease

Taken from ESL

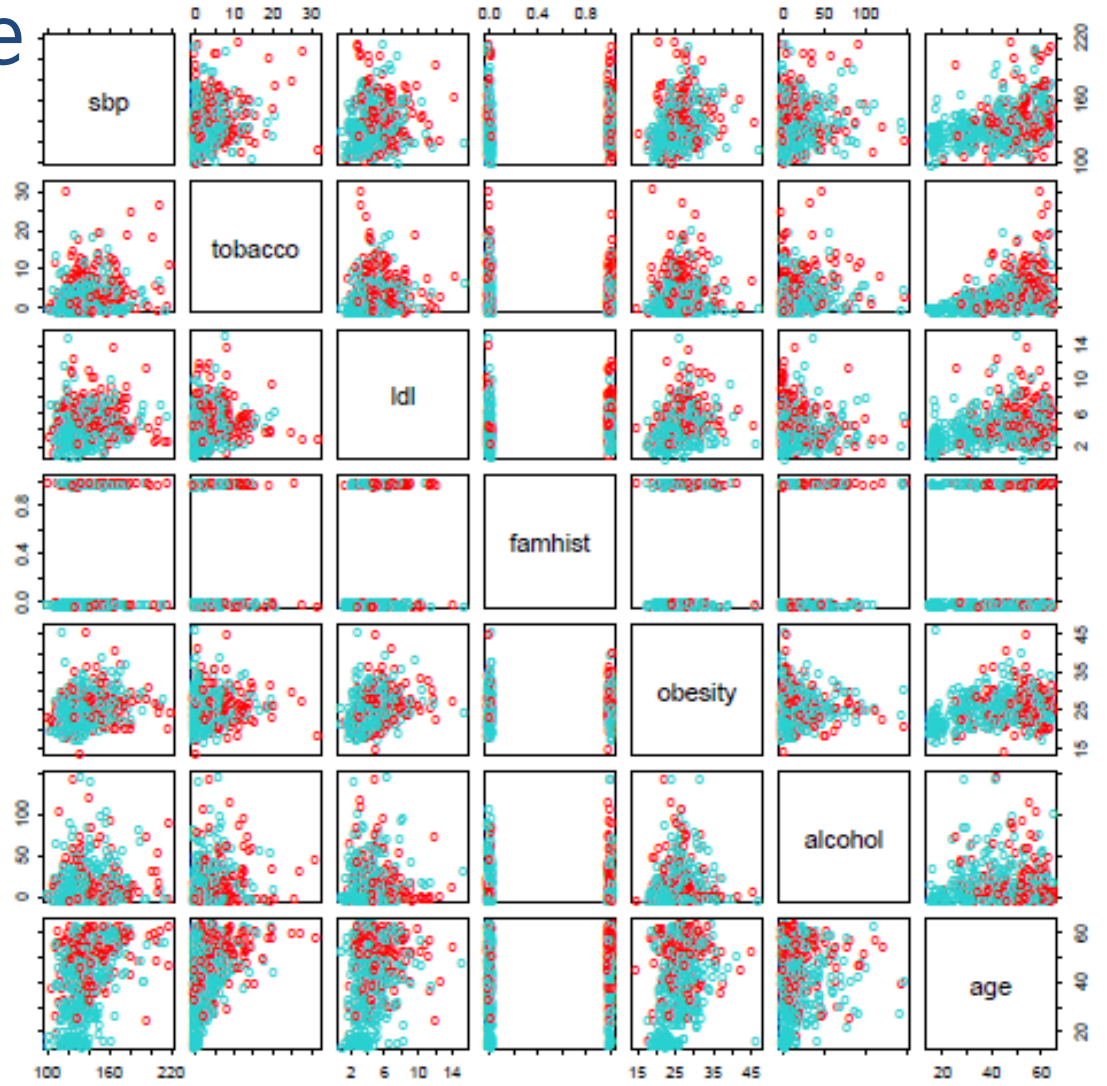


FIGURE 4.12. *A scatterplot matrix of the South African heart disease data. Each plot shows a pair of risk factors, and the cases and controls are color coded (red is a case). The variable* family history of heart disease (`famhist`) *is binary (yes or no).*

Logistic Regression for Feature Selection

If we fit the complete model on these data we get

Taken from ESL

TABLE 4.2. *Results from a logistic regression fit to the South African heart disease data.*

	Coefficient	Std. Error	Z Score
(Intercept)	−4.130	0.964	−4.285
sbp	0.006	0.006	1.023
tobacco	0.080	0.026	3.034
ldl	0.185	0.057	3.219
famhist	0.93		
obesity	-0.03		
alcohol	0.00		
age	0.04		

Taken from ESL

While if we use stepwise Logistic Regression

TABLE 4.3. *Results from stepwise logistic regression fit to South African heart disease data.*

	Coefficient	Std. Error	Z score
(Intercept)	−4.204	0.498	−8.45
tobacco	0.081	0.026	3.16
ldl	0.168	0.054	3.09
famhist	0.924	0.223	4.14
age	0.044	0.010	4.52

Logistic Regression parameters meaning

Regression parameters represent the increment on the logit of probability given by a unitary increment of a variable

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

The increase of *tobacco* consumption in life of 1Kg counts for an increase in log-odds of exp(0.081)=1.084 which means an overall increase of 8.4%

With a 95% confidence interval

$$\exp(0.081 \pm 2 \times 0.026) = (1.03, 1.14)$$

TABLE 4.3. *Results from stepwise logistic regression fit to South African heart disease data.*

	Coefficient	Std. Error	Z Score
(Intercept)	−4.2		
tobacco	0.081	0.026	3.16
ldl	0.168	0.054	3.09
famhist	0.924	0.223	4.14
age	0.044	0.010	4.52

Taken from ESL

Regularized Logistic Regression

As for Linear Regression we can compute a "Lasso" version

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

Regularized Logistic Regression

As for Linear Regression we can compute a "Lasso" version

Taken from ESL

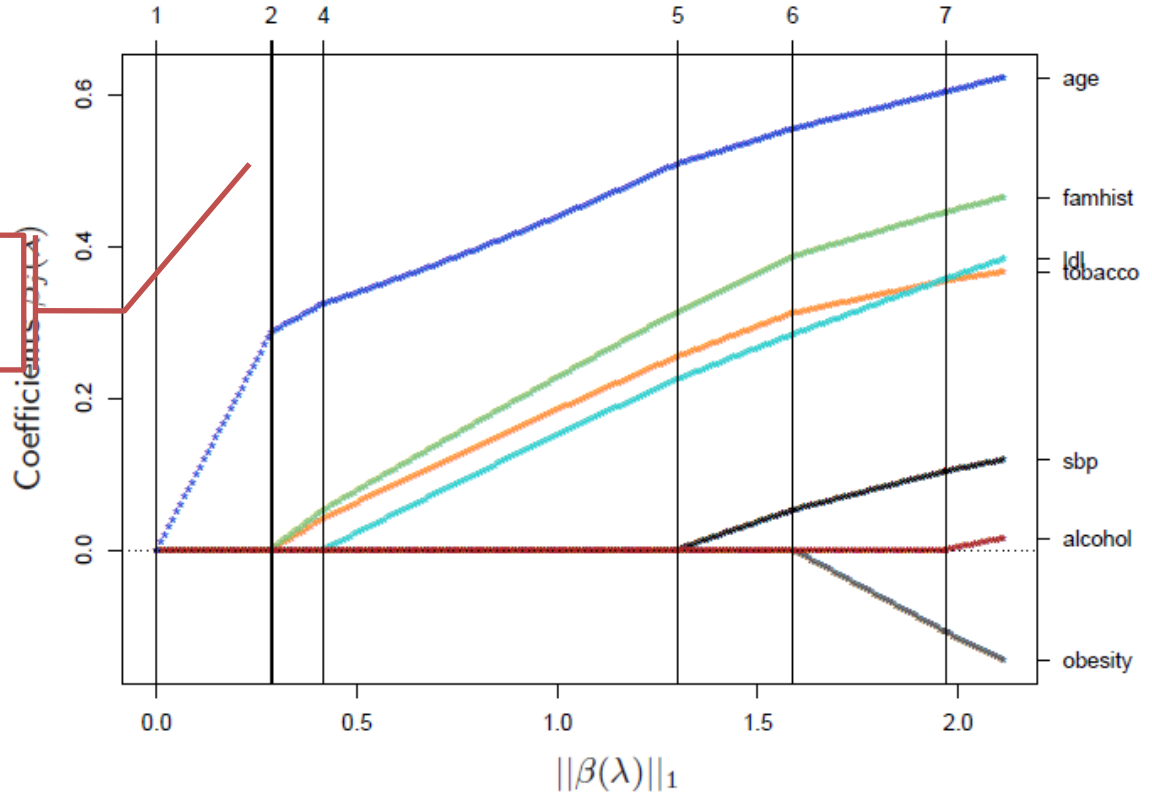


FIGURE 4.13. *L_1 regularized logistic regression coefficients for the South African heart disease data, plotted as a function of the L_1 norm. The variables were all standardized to have unit variance. The profiles are computed exactly at each of the plotted points.*

Wrap-up on Logistic Regression

We model the log-odds as a linear regression model

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

This means the posterior probability becomes

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

Parameters represent log-odds increase per variable unit increment keeping fixed the others

We can use it to perform feature selection using z-scores and forward stepwise selection

The class decision boundary is linear, but points close to the boundary count more … this will be discussed later

Prove it!

Beyond Logistic Regression …

Logistic Regression models directly class posterior probability

$$\Pr(Y = k|X = x)$$

Linear Discriminant Analysis uses the Bayes Theorem

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

What improvements come with this model?

- Parameter learning unstable in Logistic Regression for well separated classes
- With little data and normal predictor distribution LDA is more stable
- A very popular algorithm with more than 2 response classes

Linear Discriminant Analysis (1/3)

Suppose we want to discriminate among K>2 classes, each with a prior probability π_k

Given the class we model the density function of predictors as

$$f_k(X) \equiv \Pr(X = x|Y = k)$$

Using the Bayes Theorem we obtain

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

- Prior probability π_k is relatively simple to learn
- Likelihood $f_k(X)$might be more tricky and we need some assumptions

If we correctly estimate likelihood $f_k(X)$ we obtain the *Bayes Classifier*!!!

Linear Discriminant Analysis (2/3)

Let assume p=1 and use a Gaussian distribution

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)$$

Let assume all classes have the same covariance

$$\sigma_1^2 = \ldots = \sigma_K^2$$

The posteriors probability as computed by LDA becomes

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}$$

The selected class is the one with the highest posterior which the one with highest discriminating function

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Can you derive this?

Linear discriminant function in x

Linear Discriminant Analysis (3/3)

With 2 classes having the same prior probability $\pi_1 = \pi_2$ we decide the class according to the inequality

$$2x\,(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$$

The Bayes decision boundary corresponds to

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2}$$

Training is as simple as estimating the model parameters

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

$$\hat{\pi}_k = n_k/n$$

LDA Simple Example (with p=1)

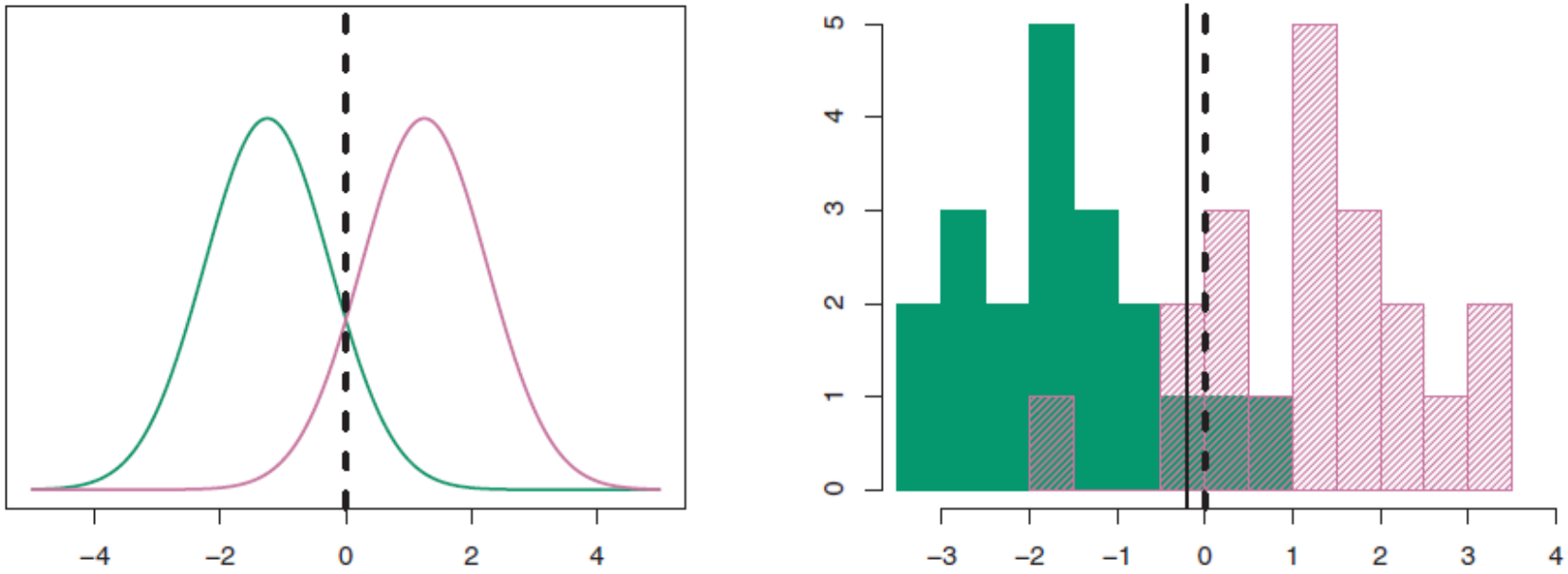


FIGURE 4.4. Left: *Two one-dimensional normal density functions are shown. The dashed vertical line represents the Bayes decision boundary.* Right: *20 observations were drawn from each of the two classes, and are shown as histograms. The Bayes decision boundary is again shown as a dashed vertical line. The solid vertical line represents the LDA decision boundary estimated from the training data.*

Linear Discriminant Analysis with p>1 (1/3)

In case p>1 we assume $X = (X_1, X_2, \ldots, X_p)$ comes from

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

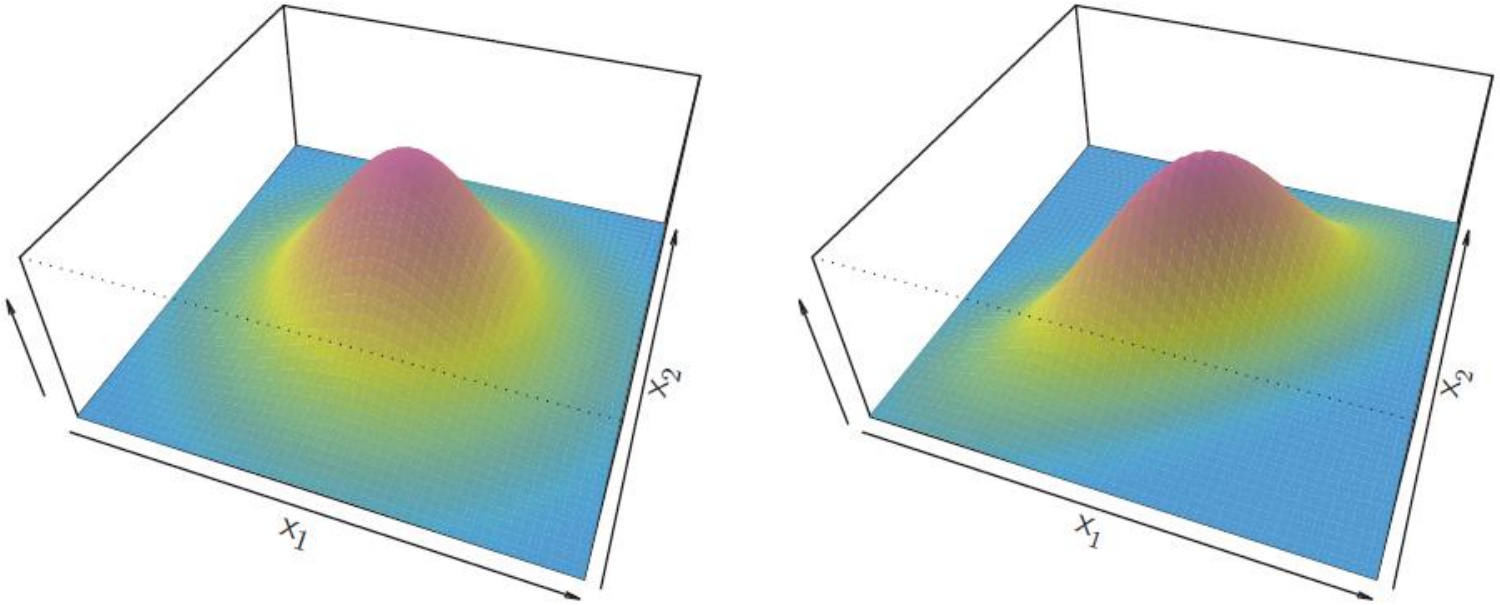


FIGURE 4.5. *Two multivariate Gaussian density functions are shown, with* $p = 2$. Left: *The two predictors are uncorrelated.* Right: *The two variables have a correlation of* 0.7.

Linear Discriminant Analysis with p>1 (2/3)

In the case of p>1 the LDA classifier assumes

- Observations from the k-th class are drawn from $N(\mu_k, \mathbf{\Sigma})$
- The covariance structure is common to all classes

The Bayes discriminating function becomes

Still linear in x!!!

Can you derive this?

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

From this we can compute the boundary between each class (considering the two classes having the same prior probability)

$$x^T \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k^T \mathbf{\Sigma}^{-1} \mu_k = x^T \mathbf{\Sigma}^{-1} \mu_l - \frac{1}{2} \mu_l^T \mathbf{\Sigma}^{-1} \mu_l$$

Training formulas for the LDA parameters are similar to the case of p=1 …

Linear Discriminant Analysis Example

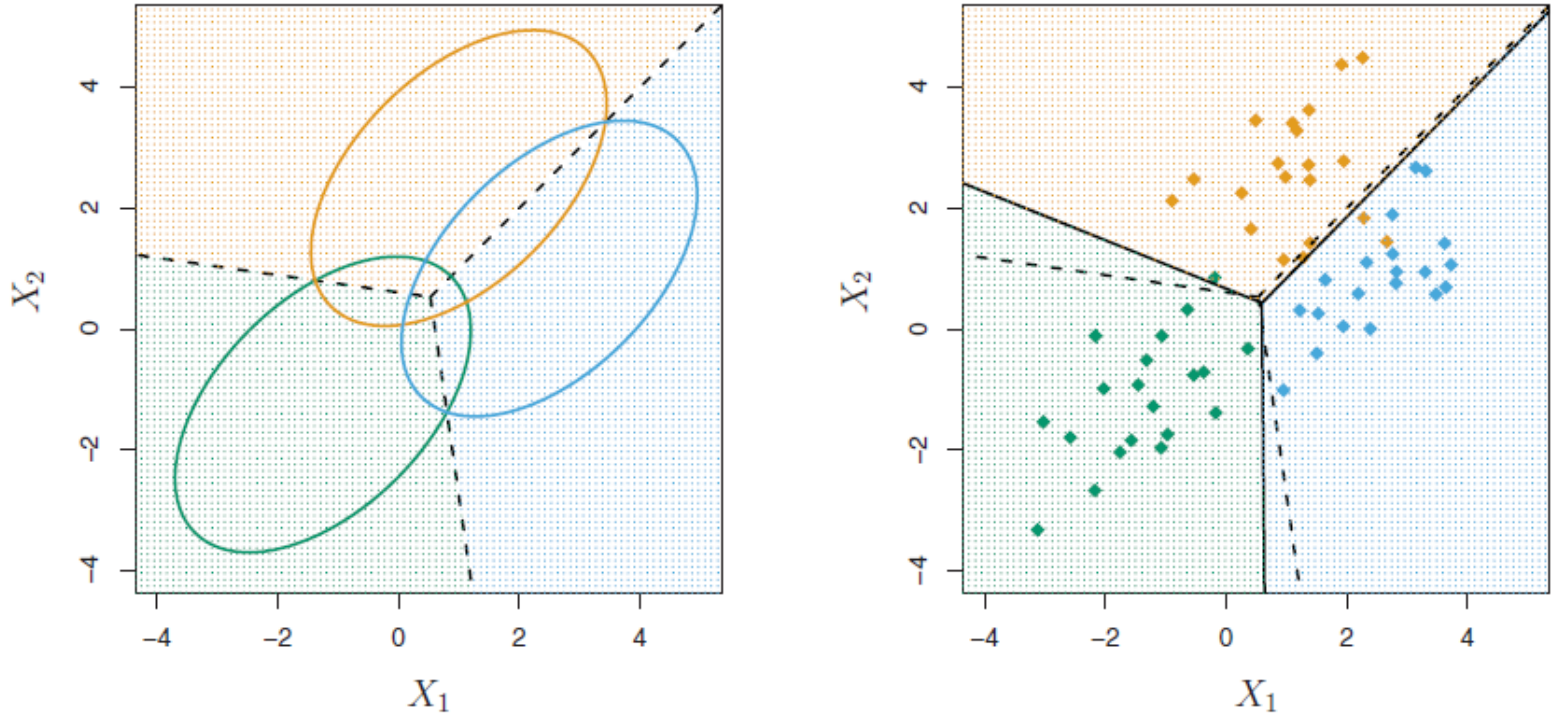


FIGURE 4.6. *An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with $p = 2$, with a class-specific mean vector and a common covariance matrix.* Left: *Ellipses that contain 95 % of the probability for each of the three classes are shown. The dashed lines are the Bayes decision boundaries.* Right: 20 *observations were generated from each class, and the corresponding LDA decision boundaries are indicated using solid black lines. The Bayes decision boundaries are once again shown as dashed lines.*

Quadratic Discriminant Analysis

Linear Discriminant Analysis assumes all classes with common covariance

Quadratic Discriminant Analysis assumes different covariances

$$X \sim N(\mu_k, \mathbf{\Sigma}_k)$$

Under this hypothesis the Bayes discriminant function becomes

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

Can you derive this?

Quadratic function

The decision LDA vs. QDA boils down to bias-variance trade-off

- QDA requires $Kp(p+1)/2$ parameters while LDA only Kp

QDA vs LDA Example

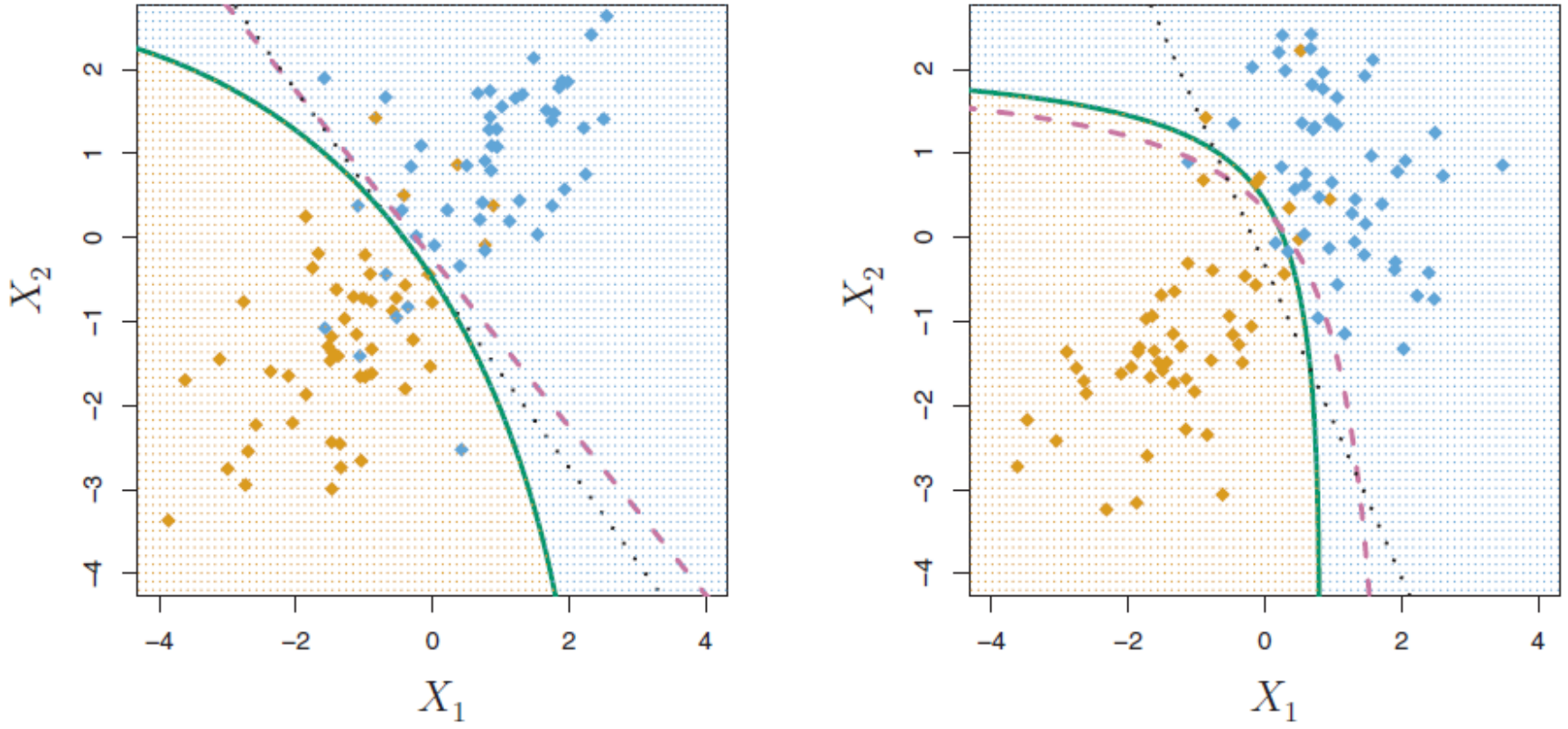


FIGURE 4.9. Left: *The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with* $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$. *The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA.* Right: *Details are as given in the left-hand panel, except that* $\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2$. *Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.*

Which classifier is better?

Let consider 2 classes and 1 predictor

- It can be seen that for LDA the log odds is given by

$$\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = \log\left(\frac{p_1(x)}{p_2(x)}\right) = c_0 + c_1 x$$

- While for Logistic Regression the log odds is

$$\log\left(\frac{p_1}{1-p_1}\right) = \beta_0 + \beta_1 x$$

- Both linear functions but learning procedures are different …

Linear Discriminant Analysis is the Optimal Bayes if its hypothesis holds otherwise Logistic Regression can outperforms it!

Quadratic Discriminant Analysis is to be preferred if the class covariances are different and we have a non linear boundary

Some scenarios tested …

Linear Boundary Scenarios

1. Samples from 2 uncorrelated normal distributions
2. Samples from 2 slightly correlated normal distributions
3. Samples from t-student distributed classes

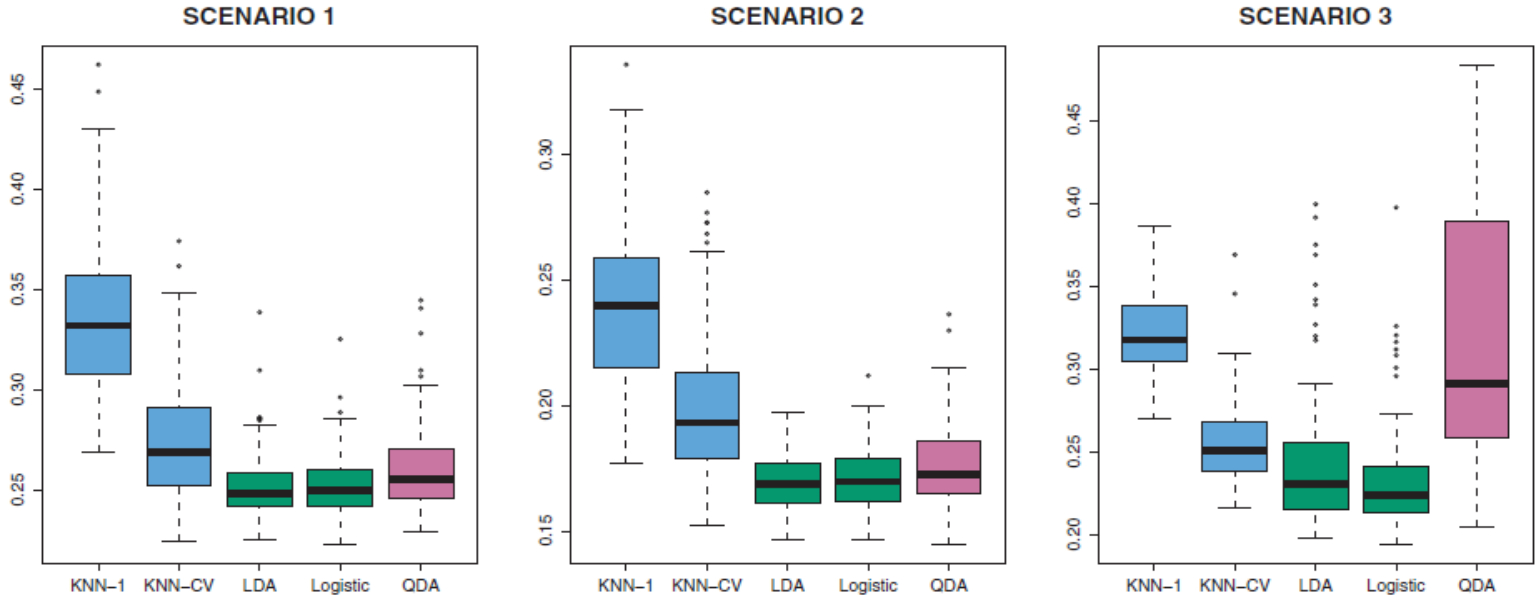


FIGURE 4.10. *Boxplots of the test error rates for each of the linear scenarios described in the main text.*

Some other scenarios tested …

Non Linear Boundary Scenarios

4. Samples from 2 normal distribution with different correlation
5. Samples from 2 normals, predictors are quadratic functions
6. As previous but with a more complicated function

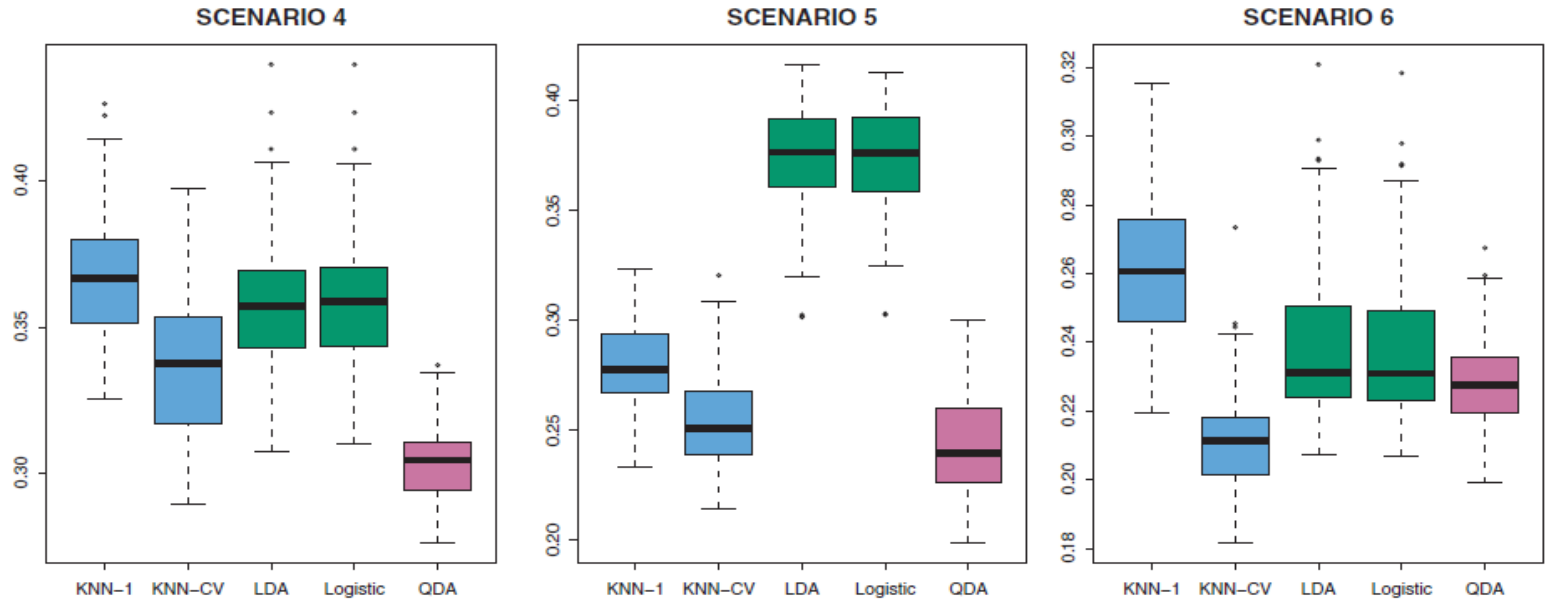


FIGURE 4.11. *Boxplots of the test error rates for each of the non-linear scenarios described in the main text.*

Overall conclusion on the comparison

No method is better than all the others!

- If the decision boundary is linear then
  LDA and Logistic Regression are those performing better

- When the decision boundary is moderately non linear
  QDA may give better results

- For much complex decision boundaries non parametric approaches such as KNN perform better, but the right level of smoothness has to be chosen

POLITECNICO
MILANO 1863

Classifiers Evaluation

Matteo Matteucci, PhD (matteo.matteucci@polimi.it)

Artificial Intelligence and Robotics Laboratory
Politecnico di Milano

AIRLAB
ARTIFICIAL INTELLIGENCE AND ROBOTICS LAB

Example: LDA on the Default Dataset

LDA on the Default dataset gets 2.75% training error rate

- Having 10000 records and p=3 we do not expect much overfitting … by the way how many parameters we have?
- Being 3.33% the defaulters a dummy classifier would get similar error rate

$252/333 = 75.7\,\%$

		True default status		
		No	Yes	Total
Predicted default status	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000

TABLE 4.4. *A confusion matrix compares the LDA predictions to the true default statuses for the 10,000 training observations in the* `Default` *data set. Elements on the diagonal of the matrix represent individuals whose default statuses were correctly predicted, while off-diagonal elements represent individuals that were misclassified. LDA made incorrect predictions for 23 individuals who did not default and for 252 individuals who did default.*

On the performance of LDA

Errors in classification are often reported as a Confusion Matrix

		True default status		
		No	Yes	Total
Predicted default status	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000

99.8 %

24.3 %

- Sensitivity: percentage of true defaulters
- Specificity: percentage of non-defaulters correctly identified

The Bayes classifier optimize the overall error rate independently from the class they belong to an it does this by thresholding

$$\Pr(\texttt{default} = \texttt{Yes}|X = x) > 0.5$$

Can we do better?

Example: Increasing LDA Sensitivity

We might want to improve classifier sensitivity with respect to a given class because we consider it more "critical"

$$P(\texttt{default} = \texttt{Yes} | X = x) > 0.2$$

- Reduced "Default" error rate from 75.7% to 41.4%
- Increased overall error of 3.73% (but it is worth)

		True default status		
		No	Yes	Total
Predicted default status	No	9,432	138	9,570
	Yes	235	195	430
	Total	9,667	333	10,000

TABLE 4.5. *A confusion matrix compares the LDA predictions to the true default statuses for the* 10,000 *training observations in the* `Default` *data set, using a modified threshold value that predicts default for any individuals whose posterior default probability exceeds* 20 %.

Tweaking LDA sensitivity

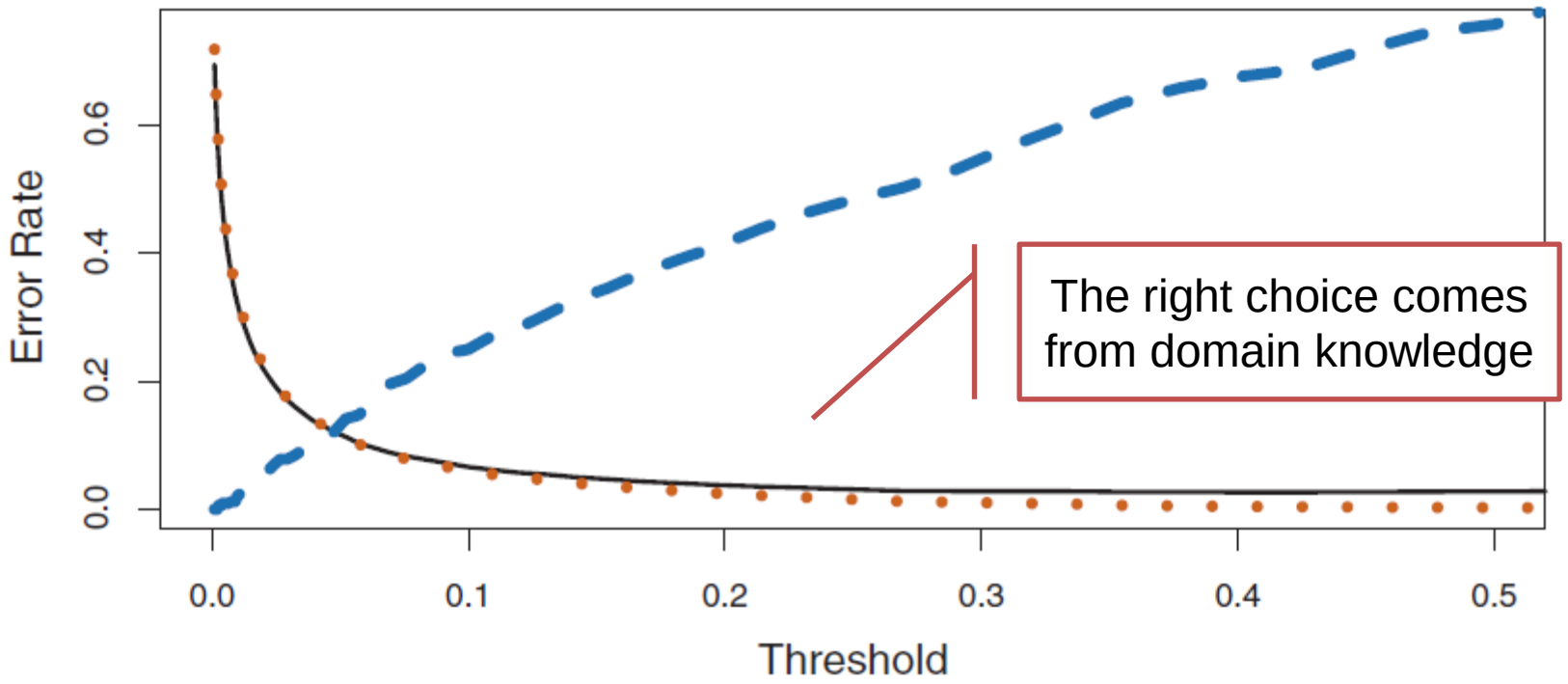


FIGURE 4.7. *For the* `Default` *data set, error rates are shown as a function of the threshold value for the posterior probability that is used to perform the assignment. The black solid line displays the overall error rate. The blue dashed line represents the fraction of defaulting customers that are incorrectly classified, and the orange dotted line indicates the fraction of errors among the non-defaulting customers.*

ROC Curve

The ROC (Receiver Operating Characteristics) summarizes false positive and false negative errors

Obtained by testing all possible thresholds

- Overall performance given by Area Under the ROC Curve
- A classifier which randomly guesses (with two classes) has AUC = 0.5 a perfect classifier has AUC = 1

ROC curve considers true positive and false positive rates

- Sensitivity is equivalent to true positive rate
- Specificity is equivalent to 1 – false positive rate

ROC Curve for LDA on Default data

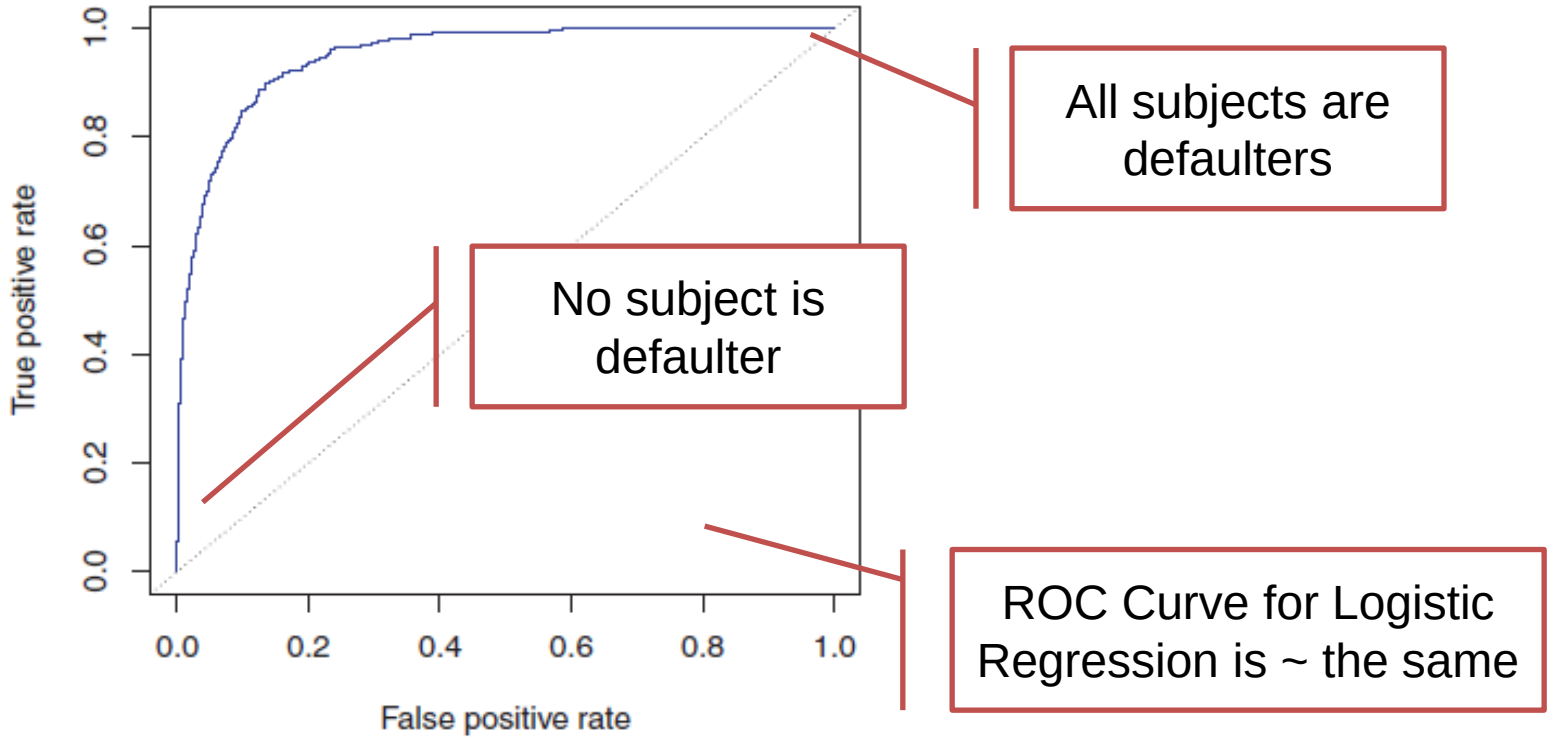


FIGURE 4.8. *A ROC curve for the LDA classifier on the* `Default` *data. It traces out two types of error as we vary the threshold value for the posterior probability of default. The actual thresholds are not shown. The true positive rate is the sensitivity: the fraction of defaulters that are correctly identified, using a given threshold value. The false positive rate is 1-specificity: the fraction of non-defaulters that we classify incorrectly as defaulters, using that same threshold value. The ideal ROC curve hugs the top left corner, indicating a high true positive rate and a low false positive rate. The dotted line represents the "no information" classifier; this is what we would expect if student status and credit card balance are not associated with probability of default.*

Clearing out terminology

When applying a classifier we can obtain

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

TABLE 4.6. *Possible results when applying a classifier or diagnostic test to a population.*

Out of this we can define the following

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N*	

TABLE 4.7. *Important measures for classification and diagnostic testing, derived from quantities in Table 4.6.*

Matthews Correlation Coefficient

https://en.wikipedia.org/wiki/Matthews_correlation_coefficient

POLITECNICO
MILANO 1863

Linear Classifiers (Discriminative)

Matteo Matteucci, PhD (matteo.matteucci@polimi.it)

Artificial Intelligence and Robotics Laboratory
Politecnico di Milano

AIRLAB
ARTIFICIAL INTELLIGENCE AND ROBOTICS LAB

Discriminative vs. Generative Approaches

Generative approach: we derived the classifier from some generative hypothesis about the way data have been generated

- Linearity of the log odds for posteriors (Logistic Regression)
- Multivariate Gaussian given the class for the likelihood (LDA)

Discriminative approach: find the prescribed boundary (e.g., a linear separating boundary) able to reduce the classifier error

- Define a discriminating function and optimize it instead of ma assumptions on the data distribution

From ESL

Example: find the separating hyperplane which separates the data points in the best way (e.g., with the minimum error rate)

$$\{x : \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0\}$$

Perceptron

A perceptron computes the value of a weighted sum and returns its sign (name dates back to '50s literature on neural networks)

$$\{x : \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0\}$$

Basically a perceptron is a linear classifier for which:

- We do not assume any particular probabilistic model generating the data
- We learn the parameters using some optimization technique so to minimize an error function (e.g., the error rate)
- We cannot infer the role of the single variables in the model from the values of the weights

Inference in discriminative models is quite complex and usually it is not the goal, instead, they usually well perform in prediction.

An Example: Simulated Data

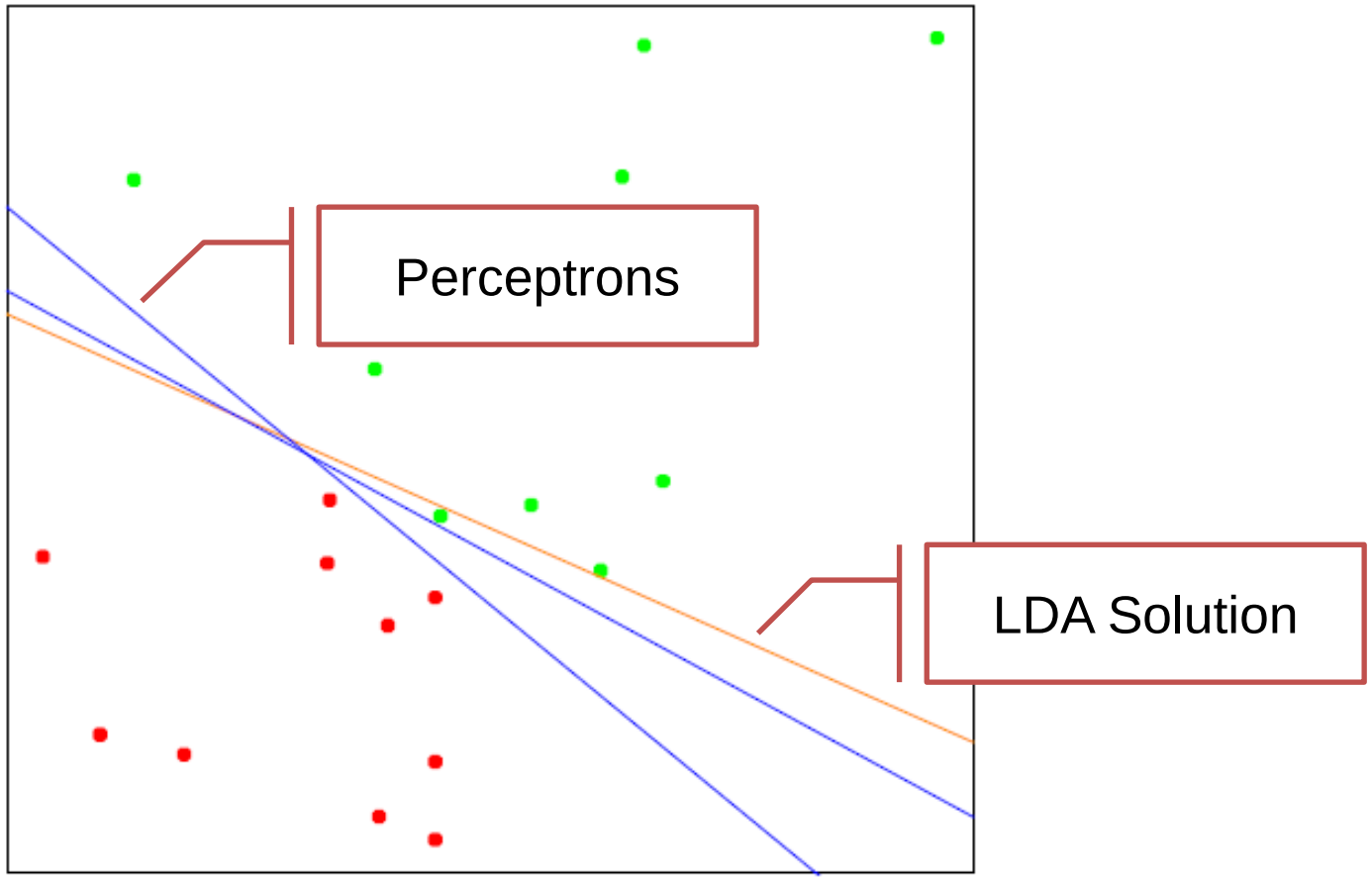


FIGURE 4.14. *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

Hyperplanes Linear Algebra

Let consider the hyperplane (affine set) L in $\mathbb{R}^2$

$$f(x) = \beta_0 + \beta^T x = 0$$

- Any two points x_1 and x_2 on L have

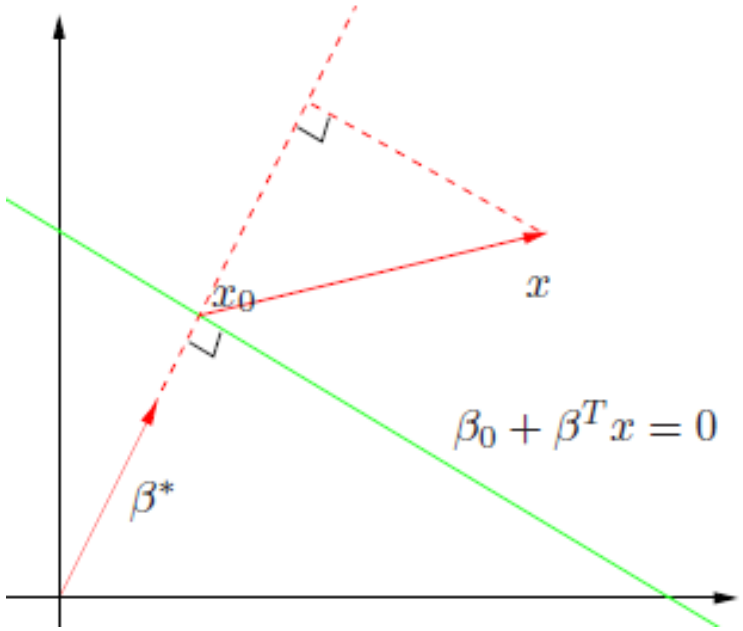
$$\beta^T(x_1 - x_2) = 0$$

- The vector normal to the surface L is

$$\beta^* = \beta/||\beta||$$

- For any point x_0 in L we have

$$\beta^T x_0 = -\beta_0$$



- The signed distance of any point x to L is defined by

$$\begin{aligned} \beta^{*T}(x - x_0) &= \frac{1}{\|\beta\|}(\beta^T x + \beta_0) \\ &= \frac{1}{||f'(x)||} f(x). \end{aligned}$$

f(x) proportional to the distance of x from the plane defined by *f(x)=0*

Perceptron Learning Algorithm (1/2)

The error function for the perceptron learning is the distance of misclassified points from the decision boundary

$$x_i^T \beta + \beta_0 < 0$$

- The output is coded with +1/-1
- If an output which should be +1 is misclassified
- For an output with -1 we have the opposite

The goal becomes minimizing

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^T \beta + \beta_0)$$

Set of points misclassified

- non negative and proportional to the distance of the misclassified points from

$$\beta^T x + \beta_0 = 0$$

Perceptron Learning Algorithm (2/2)

Minimize by stochastic gradient descend the error function

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^T \beta + \beta_0)$$

- The gradients with respect to the model parameters are

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i x_i,$$

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i.$$

- Stochastic gradient descent applies for each misclassified point

$$\begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} \leftarrow \begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} + \rho \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix}$$

Learning rate

If data are linearly separable, it converges to a separating hyperplane

An Example: Simulated Data

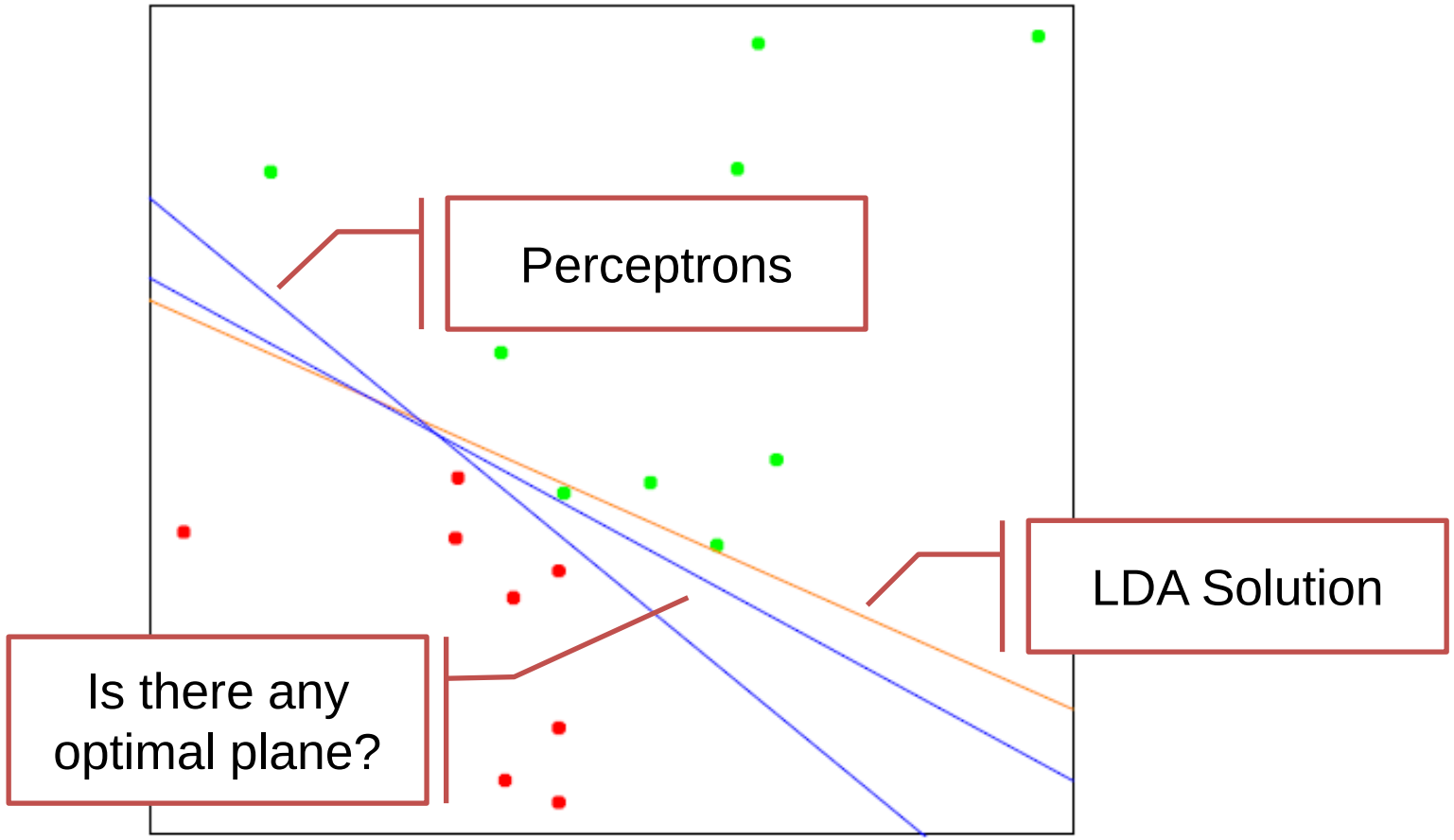


FIGURE 4.14. *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

An Example: Simulated data

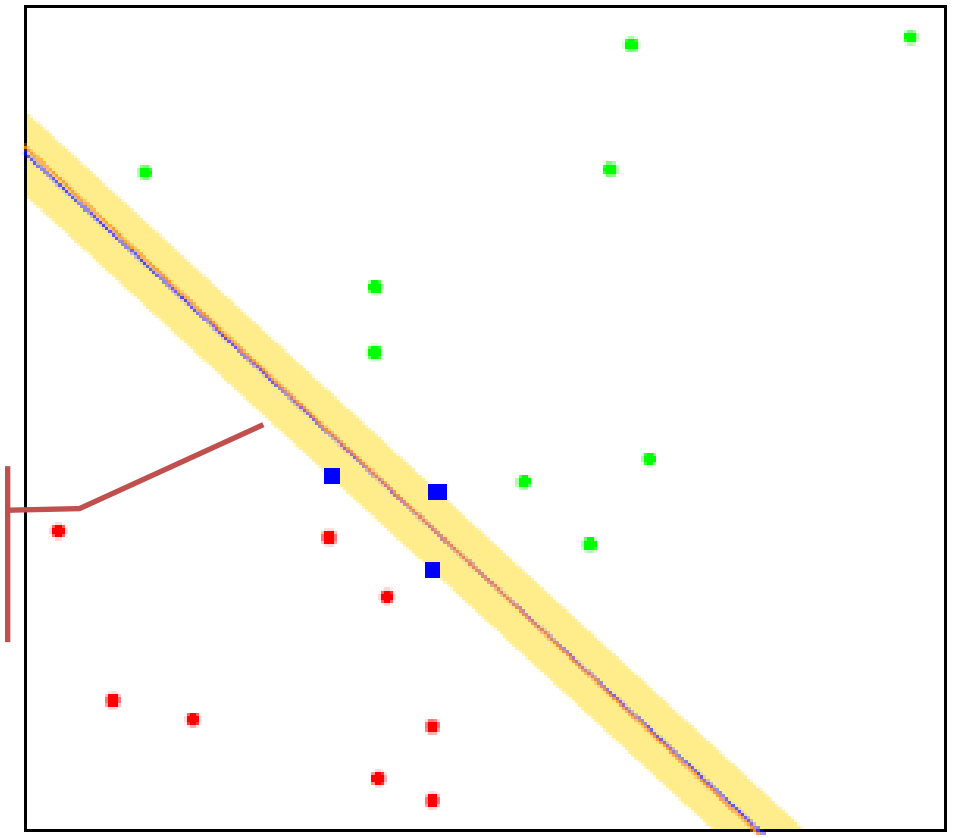


FIGURE 4.16. *The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

Separable Case Formulation (1)

From ESL

Maximize the margin M

$$\max_{\beta,\beta_0,||\beta||=1} M$$

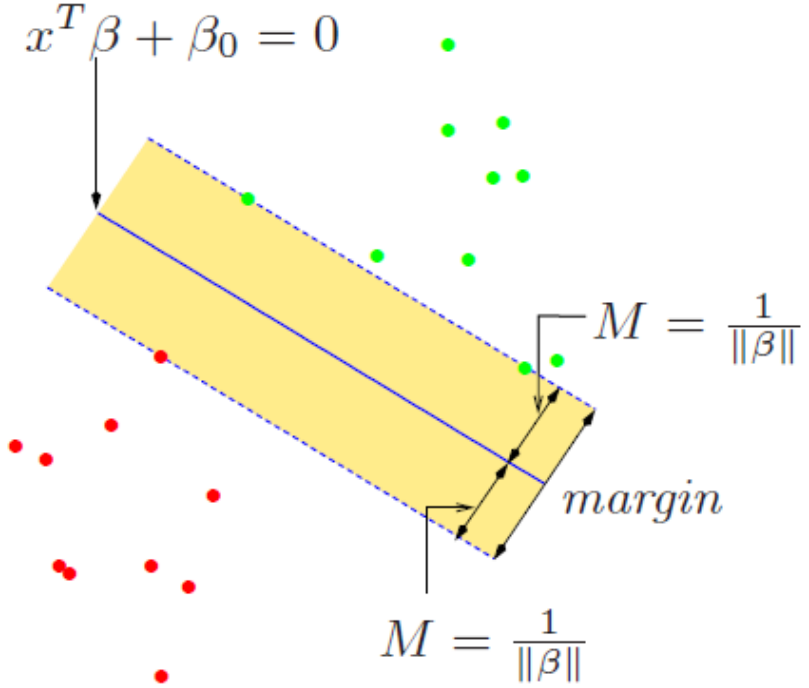
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M,\ i = 1,\ldots,N$$

The two constraints respectively

- Select one of the possible hyper planes

$$k(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) = 0$$

- Each point is on the right side of the margin



Separable Case Formulation (2)

Remove the constraint on parameter norm changing margin constraint

$$\frac{1}{||\beta||} y_i(x_i^T \beta + \beta_0) \geq M$$

- which becomes

$$y_i(x_i^T \beta + \beta_0) \geq M||\beta||$$

If we redefine $M = 1/||\beta||$ we obtain the equivalent problem

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2$$

$$\text{subject to } y_i(x_i^T \beta + \beta_0) \geq 1,\ i = 1, \ldots, N$$

Which is a "simple" convex problem (quadratic with linear constraints)

Separable Case Solution (1)

We can solve the constrained quadratic problem by Lagrange multipliers

$$L_P = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i [y_i(x_i^T \beta + \beta_0) - 1]$$

- Setting the derivatives to zero we have

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i,$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i,$$

- And by substitution the so-called Wolf dual

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

$$\text{subject to} \quad \alpha_i \geq 0.$$

Check ESL book for derivation …

Separable Case Solution (2)

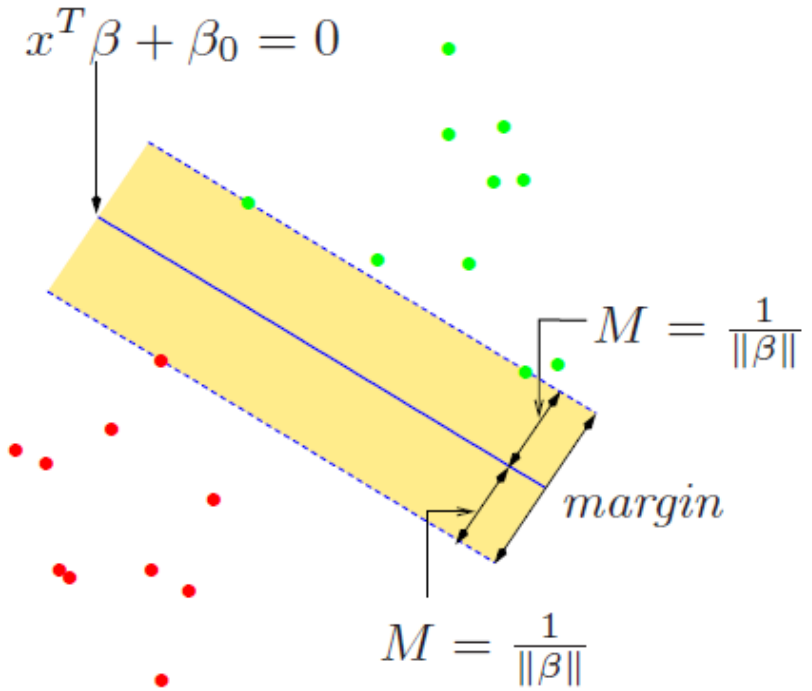
The Karush-Kuhn-Tucker conditions must hold too

$$\alpha_i[y_i(x_i^T\beta + \beta_0) - 1] = 0 \; \forall i.$$

- if $\alpha_i > 0$ then we have $y_i(x_i^T\beta + \beta_0) = 1$
- If $y_i(x_i^T\beta + \beta_0) > 1$ this means $\alpha_i = 0$

The final output comes from:

$$\hat{G}(x) = \text{sign}\hat{f}(x)$$



Non Separable Case Formulation (1)

Maximize the margin M

$$\max_{\beta,\beta_0,||\beta||=1} M$$

$$\text{subject to } y_i(x_i^T\beta+\beta_0) \geq M,\ i=1,\ldots,N$$

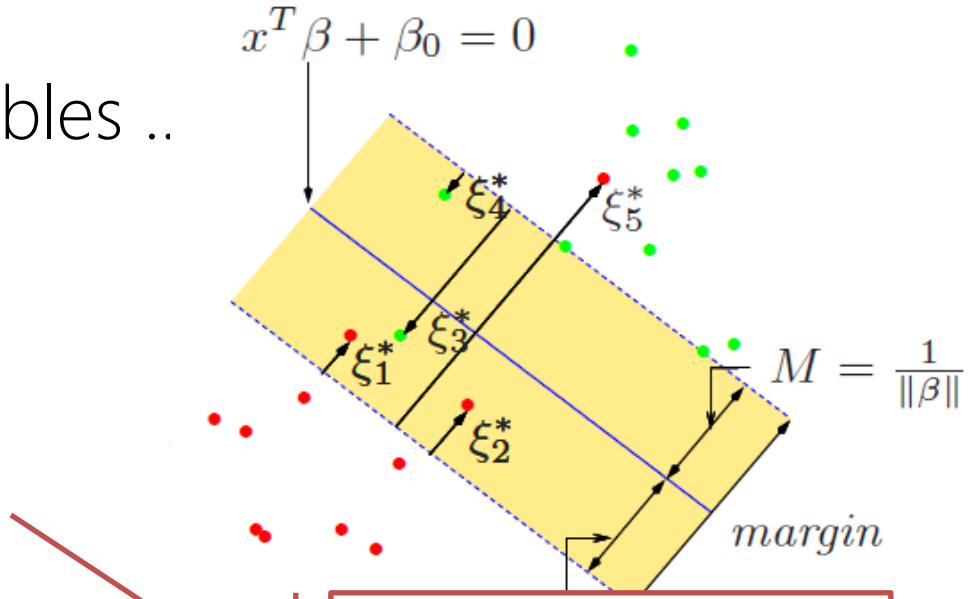
To account for "errors" we use extra variables ..

$$y_i(x_i^T\beta+\beta_0) \geq M-\xi_i,$$

or

$$y_i(x_i^T\beta+\beta_0) \geq M(1-\xi_i),$$

$$\forall i,\ \xi_i \geq 0,\ \sum_{i=1}^{N}\xi_i \leq \text{constant}$$



This gives a convex optimization problem

Non Separable Case Formulation (2)

We can remove the constraint on the parameter norm obtaining

$$\min \|\beta\| \quad \text{subject to} \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \ \forall i, \\ \xi_i \geq 0, \ \sum \xi_i \leq \text{constant}. \end{cases}$$

From ESL

This can be rewritten as

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N} \xi_i$$

$$\text{subject to} \quad \xi_i \geq 0, \ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \ \forall i,$$

Used defined cost

Infinite corresponds to separable case …

Solved by Lagrange multipliers as for the separable case …

Non Separable Case Solution (1)

The primal Lagrange function is

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i,$$

- Setting the derivatives to zero gives

$$\begin{aligned}\beta &= \sum_{i=1}^{N}\alpha_i y_i x_i,\\ 0 &= \sum_{i=1}^{N}\alpha_i y_i,\\ \alpha_i &= C - \mu_i,\ \forall i,\end{aligned}$$

- With $\alpha_i,\ \mu_i,\ \xi_i \geq 0\ \forall i$

By substitution we obtain the Lagrangian (Wolf) dual function …

Non Separable Case Solution (2)

For derivation check the ESL book …

The dual Lagrange function is

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'},$$

- Subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$
- Having the Karush-Kuhn-Tucker conditions

$$\begin{aligned} \alpha_i [y_i (x_i^T \beta + \beta_0) - (1 - \xi_i)] &= 0, \\ \mu_i \xi_i &= 0, \\ y_i (x_i^T \beta + \beta_0) - (1 - \xi_i) &\geq 0, \end{aligned}$$

Solving the optimization problem we have

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i,$$

Computed using the support vectors only

Non Separable Case Solution (3)

The solution of the Dual optimization problem provides

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i,$$

Karush-Kuhn-Tucker conditions imply

$$\begin{aligned} \alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] &= 0, \\ \mu_i \xi_i &= 0, \\ y_i(x_i^T\beta + \beta_0) - (1 - \xi_i) &\geq 0, \end{aligned}$$

- $\hat{\alpha}_i$ is non zero only for support vectors
- If $\hat{\xi}_i = 0$ the support is on the margin and $0 < \hat{\alpha}_i < C$
- If $\hat{\xi}_i > 0$ we have $\hat{\alpha}_i = C$
- Using the margin points $0 < \hat{\alpha}_i,\ \hat{\xi}_i = 0$ we can solve for β_0

Separable Data vs Non Separable Data

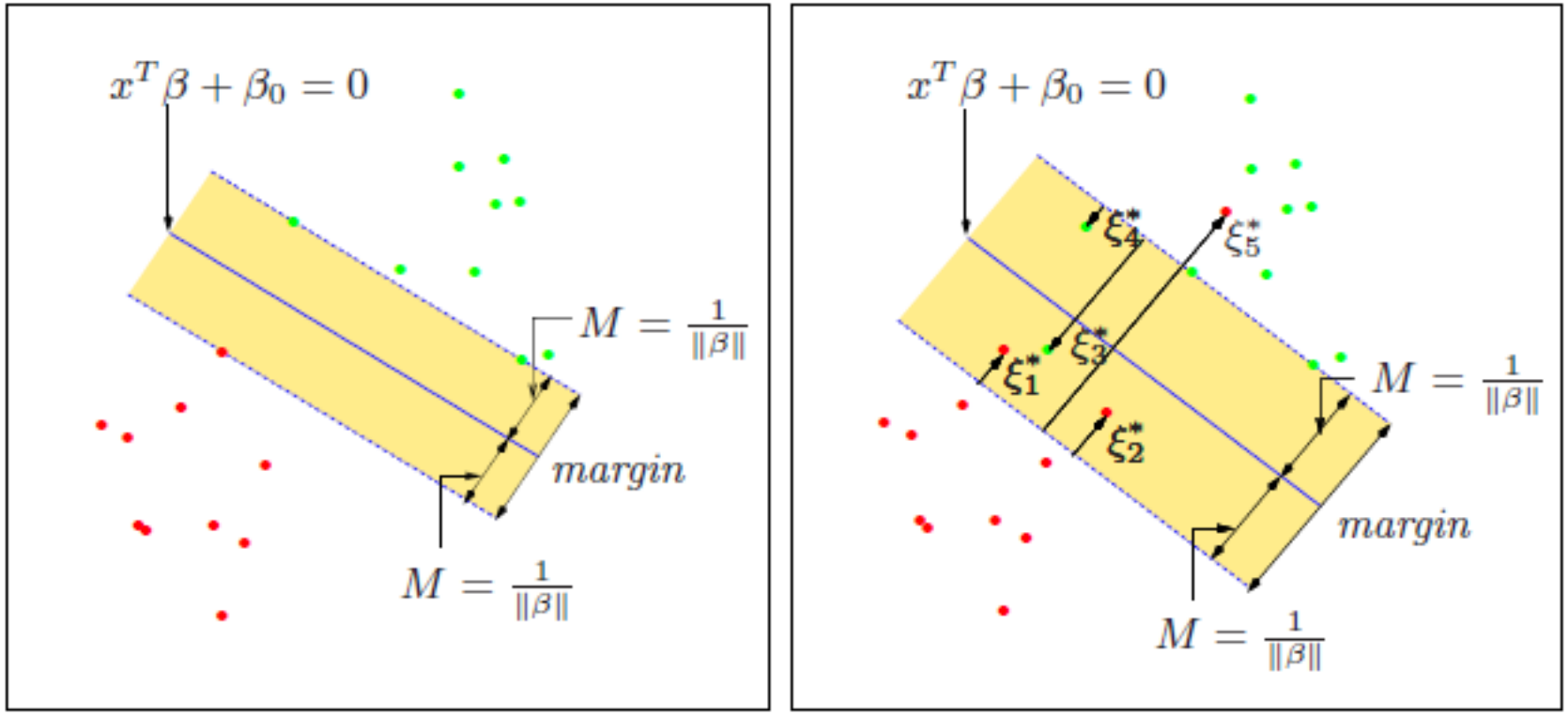


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

A Synthetic Example

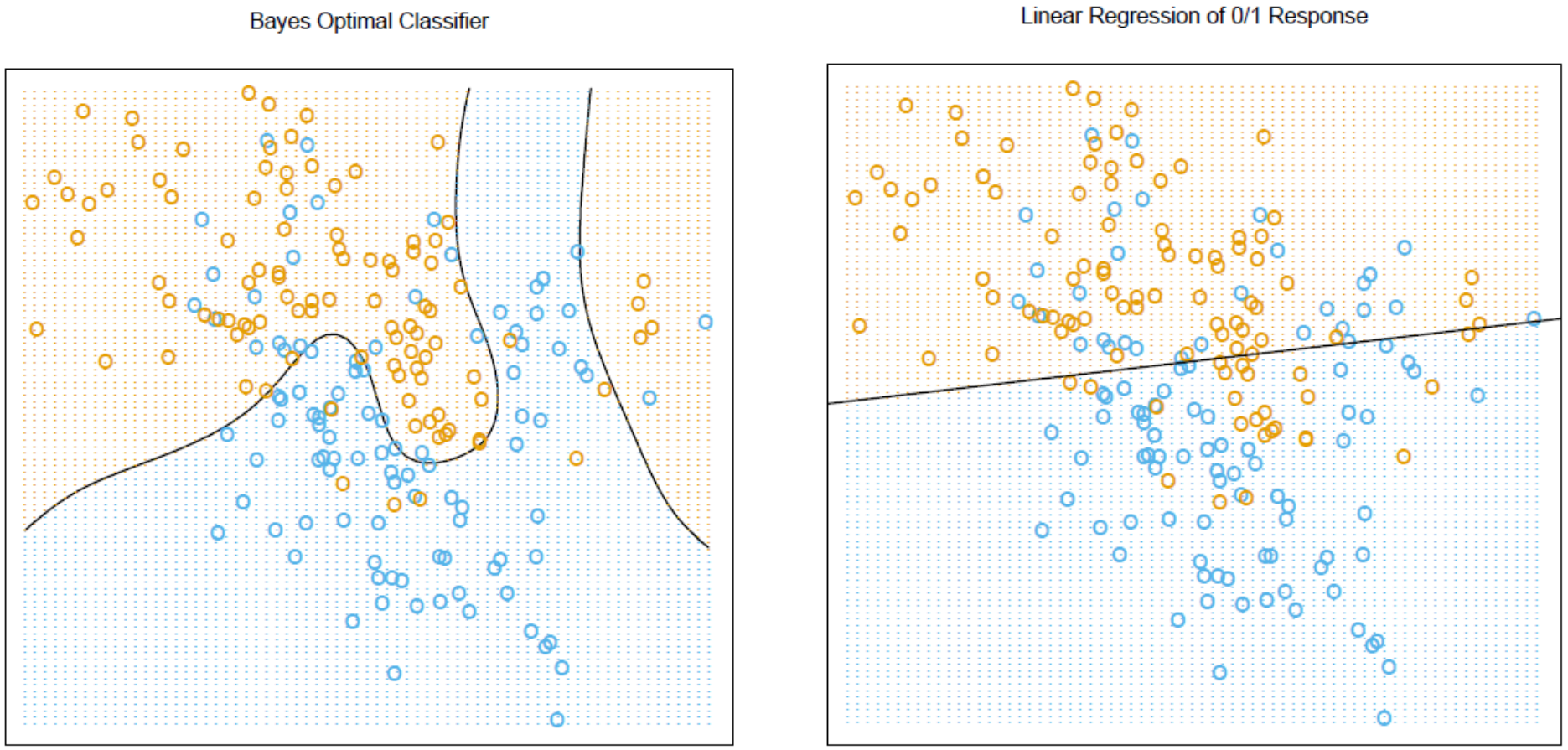
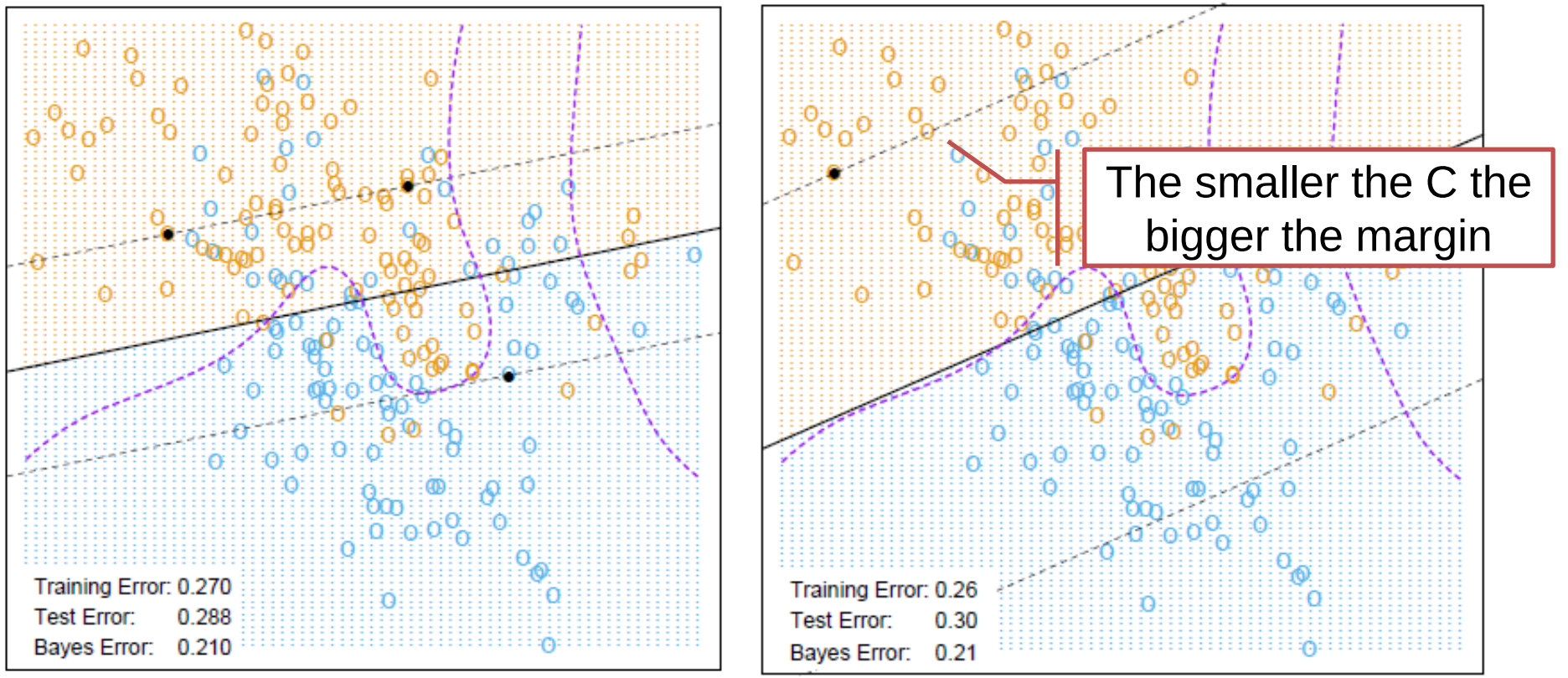


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$. *The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.*

A Synthetic Example



$C = 10000$ $C = 0.01$

FIGURE 12.2. *The linear support vector boundary for the mixture data example with two overlapping classes, for two different values of C. The broken lines indicate the margins, where $f(x) = \pm 1$. The support points ($\alpha_i > 0$) are all the points on the wrong side of their margin. The black solid dots are those support points falling exactly on the margin ($\xi_i = 0$, $\alpha_i > 0$). In the upper panel 62% of the observations are support points, while in the lower panel 85% are. The broken purple curve in the background is the Bayes decision boundary.*

Support Vector Machines (1)

Learning the classifier involves only the scalar product of features

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle$$

$$h_m(x),\ m = 1, \ldots, M$$

Scalar product

We can compute this scalar product in a new feature space

$$h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i)),\ i = 1, \ldots, N,$$

$$\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0$$

$$\hat{G}(x) = \text{sign}(\hat{f}(x))$$

The feature space can grow up to infinity with SVM …

$$\begin{aligned} f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0 \end{aligned}$$

Scalar product

The Kernel Trick

For learning and prediction we need the result of the scalar product only

$$\begin{aligned} f(x) &= h(x)^T\beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i)\rangle + \beta_0 \end{aligned}$$

In some cases this scalar product can be written as a Kernel

$$K(x, x') = \langle h(x), h(x')\rangle$$

Popular choices are

$$\begin{aligned} d\text{th-Degree polynomial: } & K(x, x') = (1 + \langle x, x'\rangle)^d, \\ \text{Radial basis: } & K(x, x') = \exp(-\gamma\|x - x'\|^2), \\ \text{Neural network: } & K(x, x') = \tanh(\kappa_1\langle x, x'\rangle + \kappa_2). \end{aligned}$$

The Kernel Trick Example

Consider an Input space with 2 variables and a polynomial kernel of degree d=2

$$\begin{aligned} K(X, X') &= (1 + \langle X, X' \rangle)^2 \\ &= (1 + X_1 X_1' + X_2 X_2')^2 \\ &= 1 + 2X_1 X_1' + 2X_2 X_2' + (X_1 X_1')^2 + (X_2 X_2')^2 + 2X_1 X_1' X_2 X_2'. \end{aligned}$$

It turns out the that M=6 and if we chose

$$\begin{aligned} h_1(X) &= 1 & h_4(X) &= X_1^2 \\ h_2(X) &= \sqrt{2} X_1 & h_5(X) &= X_2^2 \\ h_3(X) &= \sqrt{2} X_2 & h_6(X) &= \sqrt{2} X_1 X_2 \end{aligned}$$

We obtain $K(X, X') = \langle h(X), h(X') \rangle$

A Synthetic Example (Non Linear)

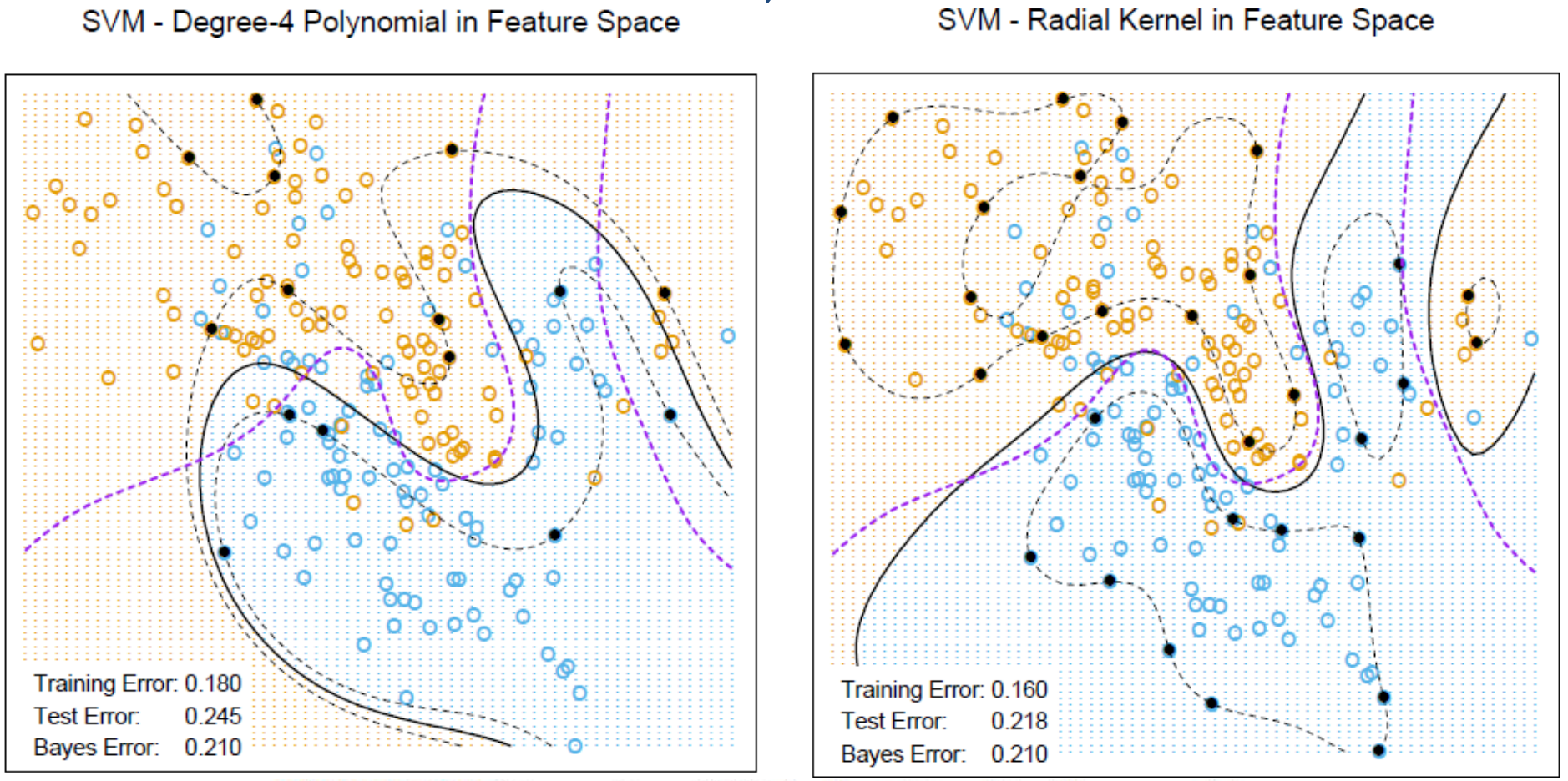


FIGURE 12.3. *Two nonlinear SVMs for the mixture data. The upper plot uses a 4th degree polynomial kernel, the lower a radial basis kernel (with $\gamma = 1$). In each case C was tuned to approximately achieve the best test error performance, and $C = 1$ worked well in both cases. The radial basis kernel performs the best (close to Bayes optimal), as might be expected given the data arise from mixtures of Gaussians. The broken purple curve in the background is the Bayes decision boundary.*