M5MS10
Machine Learning

Spring 2018

Lecture 4

Dr Ben Calderhead

b.calderhead@imperial.ac.uk

M5MS10
Machine Learning

Spring 2018

Lecture 4

Dr Ben Calderhead

b.calderhead@imperial.ac.uk

IS10
Learning

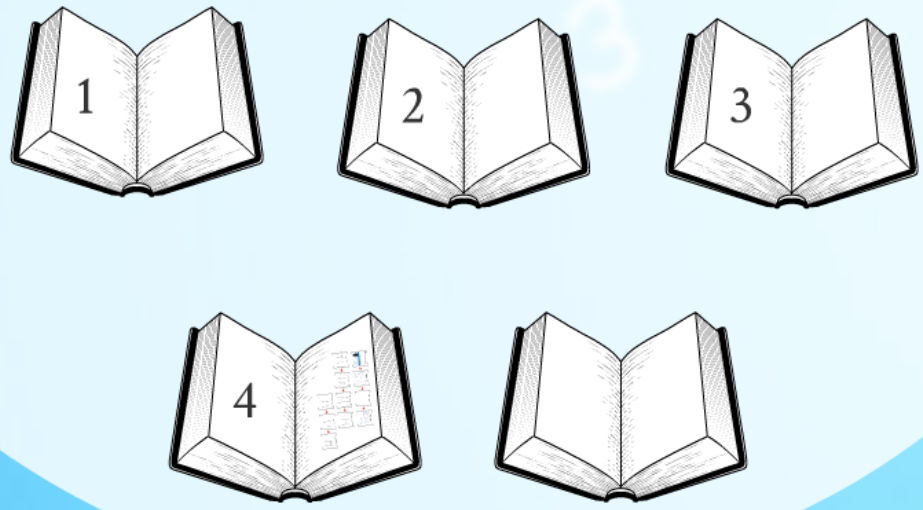
2018

re 4

lderhead

mperial.ac.uk

Parametric Methods



Classification

Classification Performance

In computer lab 2 we investigate the effect that increasing model complexity has on the performance of a classifier. In order to do so we must define an appropriate measure by which to determine performance.

We can consider the classification rate as one such performance measure, defined as the percentage of correct classifications using the testing data, having learned the model parameters using the training set.

We can classify new data points based on whether the posterior is greater or less than 0.5. Let's consider a more general loss function approach for making classification decisions.

Loss Matrices

A $(K \times K)$ loss matrix Λ with elements λ_{ij} defines the loss incurred when a pattern in class i is classified as belonging to class j, with $i, j = 1, \ldots, K$.

When $K = 2$ we therefore have the loss matrix

$$\begin{pmatrix} 0 & \lambda_{12} \\ \lambda_{21} & 0 \end{pmatrix}$$

In the case where, for example, C_1 is *healthy* and C_2 is *diseased*, we might take $\lambda_{21} > \lambda_{12}$, if the consequences of wrongly diagnosing a healthy subject as being diseased are particularly great.

We can then perform classification by incorporating our class posterior probability with a loss function.

Loss Functions

The conditional loss (or risk) incurred when $\mathbf{x}$ is assigned to C_k is

$$r(C_k|\mathbf{x}) = \sum_{i=1}^{K} \lambda_{ik} p(C_i|\mathbf{x})$$

As an example, consider λ_{ij} is the loss incurred for deciding C_i when the truth is C_j.

The conditional risks are,

$$r(C_1|\mathbf{x}) = \lambda_{11} p(C_1|\mathbf{x}) + \lambda_{21} p(C_2|\mathbf{x})$$

and

$$r(C_2|\mathbf{x}) = \lambda_{12} p(C_1|\mathbf{x}) + \lambda_{22} p(C_2|\mathbf{x})$$

The rule says that we should decide upon C_1 if

$$r(C_1|\mathbf{x}) < r(C_2|\mathbf{x})$$

Zero-One Loss

As a special case, consider the zero-one loss matrix, with elements $\lambda_{ik} = 0$ if $i = k$, otherwise $\lambda_{ik} = 1$.

We choose to assign a variable to class k by minimising

$$\sum_{i=1}^{K} \lambda_{ik} p(C_i|\mathbf{x}) = \sum_{i \neq k} \lambda_{ik} p(C_i|\mathbf{x}) = 1 - p(C_k|\mathbf{x})$$

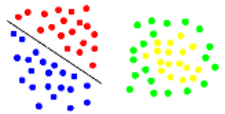
In other words, if we employ this zero-one loss we should choose the class k for which posterior $p(C_k|\mathbf{x})$ is a maximum.

Linear Separability

Suppose we use a linear classifier

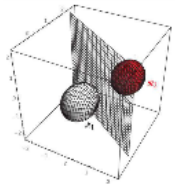
$$g(\mathbf{x}) = \theta_1^T \mathbf{x} + \theta_0$$

If we can find a set of parameters such that all the samples are correctly separated, the samples are said to be linearly separable.



Linear Separability

In the case where we have a 3 dimensional input variable, the decision boundary is a 2 dimensional hyperplane.



Classification Performance

In computer lab 2 we investigate the effect that increasing model complexity has on the performance of a classifier. In order to do so we must define an appropriate measure by which to determine performance.

We can consider the classification rate as one such performance measure, defined as the percentage of correct classifications using the testing data, having learned the model parameters using the training set.

We can classify new data points based on whether the posterior is greater or less than 0.5. Let's consider a more general loss function approach for making classification decisions.

Loss Matrices

A $(K \times K)$ loss matrix Λ with elements λ_{ij} defines the loss incurred when a pattern in class i is classified as belonging to class j, with $i, j = 1, \ldots, K$.

When $K = 2$ we therefore have the loss matrix

$$\begin{pmatrix} 0 & \lambda_{12} \\ \lambda_{21} & 0 \end{pmatrix}$$

In the case where, for example, C_1 is *healthy* and C_2 is *diseased*, we might take $\lambda_{21} > \lambda_{12}$, if the consequences of wrongly diagnosing a healthy subject as being diseased are particularly great.

We can then perform classification by incorporating our class posterior probability with a loss function.

Loss Functions

The conditional loss (or risk) incurred when $\mathbf{x}$ is assigned to C_k is

$$r(C_k|\mathbf{x}) = \sum_{i=1}^{K} \lambda_{ik} p(C_i|\mathbf{x})$$

As an example, consider λ_{ij} is the loss incurred for deciding C_i when the truth is C_j.

The conditional risks are,

$$r(C_1|\mathbf{x}) = \lambda_{11} p(C_1|\mathbf{x}) + \lambda_{21} p(C_2|\mathbf{x})$$

and

$$r(C_2|\mathbf{x}) = \lambda_{12} p(C_1|\mathbf{x}) + \lambda_{22} p(C_2|\mathbf{x})$$

The rule says that we should decide upon C_1 if

$$r(C_1|\mathbf{x}) < r(C_2|\mathbf{x})$$

Zero-One Loss

As a special case, consider the zero-one loss matrix, with elements $\lambda_{ik} = 0$ if $i = k$, otherwise $\lambda_{ik} = 1$.

We choose to assign a variable to class k by minimising

$$\sum_{i=1}^{K} \lambda_{ik} p(C_i|\mathbf{x}) = \sum_{i \neq k} \lambda_{ik} p(C_i|\mathbf{x}) = 1 - p(C_k|\mathbf{x})$$

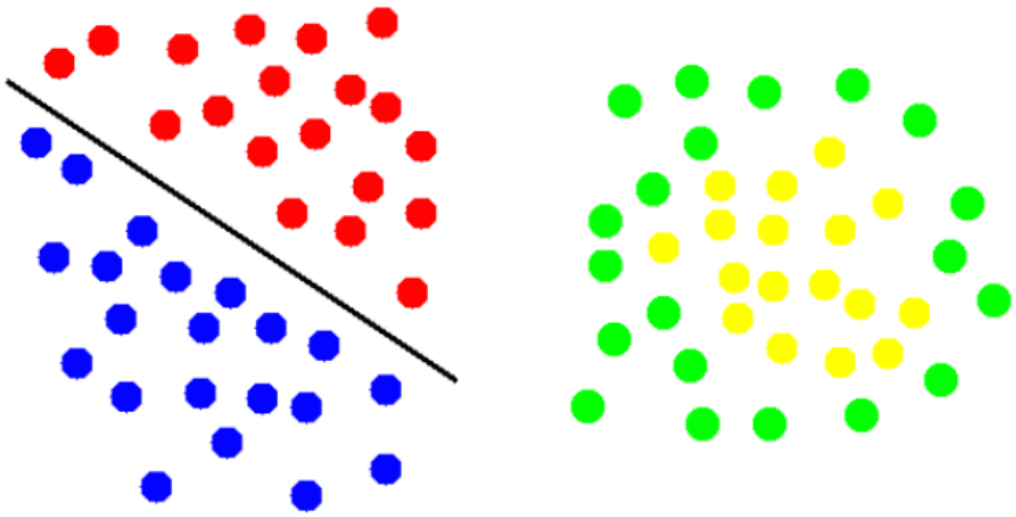
In other words, if we employ this zero-one loss we should choose the class k for which posterior $p(C_k|\mathbf{x})$ is a maximum.

Linear Separability

Suppose we use a linear classifier

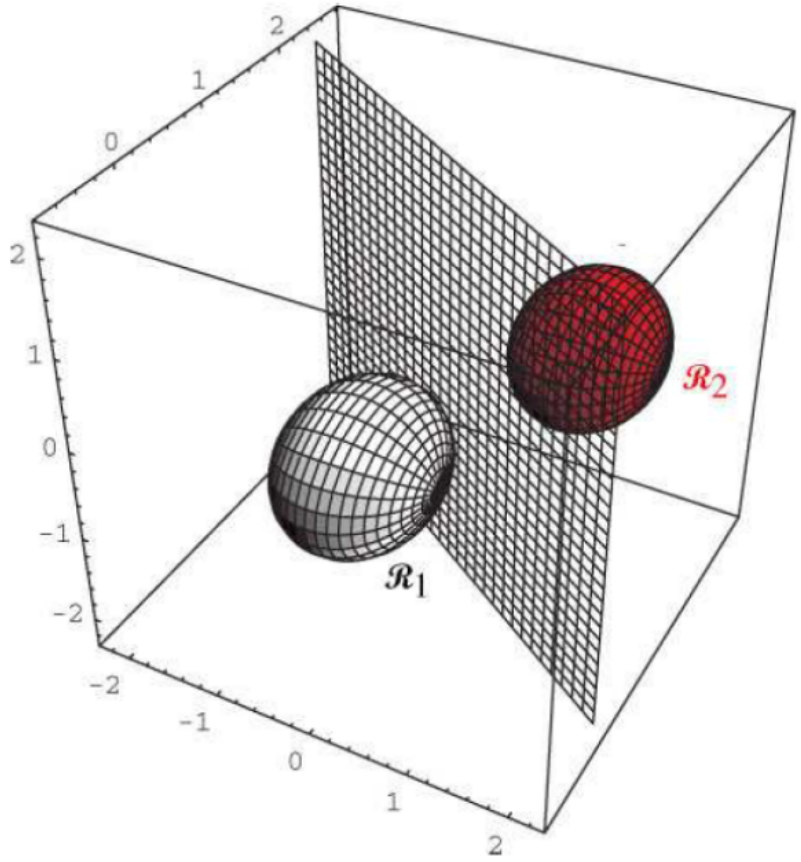
$$g(\mathbf{x}) = \boldsymbol{\theta}_1^T \mathbf{x} + \theta_0$$

If we can find a set of parameters such that all the samples are correctly separated, the samples are said to be linearly separable.



Linear Separability

In the case where we have a 3 dimensional input variable, the decision boundary is a 2 dimensional hyperplane.



Classification

Classification Performance

In computer lab 2 we investigate the effect that increasing model complexity has on the performance of a classifier. In order to do so we must define an appropriate measure by which to determine performance.

We can consider the classification rate as one such performance measure, defined as the percentage of correct classifications using the testing data, having learned the model parameters using the training set.

We can classify new data points based on whether the posterior is greater or less than 0.5. Let's consider a more general loss function approach for making classification decisions.

Loss Matrices

A $(K \times K)$ loss matrix Λ with elements λ_{ij} defines the loss incurred when a pattern in class i is classified as belonging to class j, with $i, j = 1, \ldots, K$.

When $K = 2$ we therefore have the loss matrix

$$\begin{pmatrix} 0 & \lambda_{12} \\ \lambda_{21} & 0 \end{pmatrix}$$

In the case where, for example, C_1 is *healthy* and C_2 is *diseased*, we might take $\lambda_{21} > \lambda_{12}$, if the consequences of wrongly diagnosing a healthy subject as being diseased are particularly great.

We can then perform classification by incorporating our class posterior probability with a loss function.

Loss Functions

The conditional loss (or risk) incurred when $\mathbf{x}$ is assigned to C_k is

$$r(C_k|\mathbf{x}) = \sum_{i=1}^{K} \lambda_{ik} p(C_i|\mathbf{x})$$

As an example, consider λ_{ij} is the loss incurred for deciding C_i when the truth is C_j.

The conditional risks are,

$$r(C_1|\mathbf{x}) = \lambda_{11} p(C_1|\mathbf{x}) + \lambda_{21} p(C_2|\mathbf{x})$$

and

$$r(C_2|\mathbf{x}) = \lambda_{12} p(C_1|\mathbf{x}) + \lambda_{22} p(C_2|\mathbf{x})$$

The rule says that we should decide upon C_1 if

$$r(C_1|\mathbf{x}) < r(C_2|\mathbf{x})$$

Zero-One Loss

As a special case, consider the zero-one loss matrix, with elements $\lambda_{ik} = 0$ if $i = k$, otherwise $\lambda_{ik} = 1$.

We choose to assign a variable to class k by minimising

$$\sum_{i=1}^{K} \lambda_{ik} p(C_i|\mathbf{x}) = \sum_{i \neq k} \lambda_{ik} p(C_i|\mathbf{x}) = 1 - p(C_k|\mathbf{x})$$

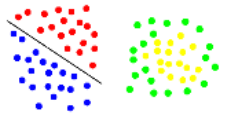
In other words, if we employ this zero-one loss we should choose the class k for which posterior $p(C_k|\mathbf{x})$ is a maximum.

Linear Separability

Suppose we use a linear classifier

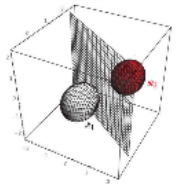
$$g(\mathbf{x}) = \theta_1^T \mathbf{x} + \theta_0$$

If we can find a set of parameters such that all the samples are correctly separated, the samples are said to be linearly separable.



Linear Separability

In the case where we have a 3 dimensional input variable, the decision boundary is a 2 dimensional hyperplane.



Generative Classification

Classification Approaches

We have already seen how we may construct a classification algorithm by considering the discriminant function.

We may adopt a discriminative approach by modelling it directly using some linear model,

$$\log \frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} = \boldsymbol{\theta}^T \phi(\mathbf{x})$$

Alternatively, using a generative approach we may define the discriminant function by directly estimating the posterior ratio from the likelihood and prior terms,

$$\log \frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} = \log \frac{p(\mathbf{x}|C=1)p(C=1)}{p(\mathbf{x}|C=0)p(C=0)}$$

Defining the Prior

The main challenge we now face is how should we define our prior distributions?

Given our data set $(\mathbf{x}, \mathbf{y})$ we could simply estimate this distribution based on the data we have. i.e. we could count the number of instances in one class, then the other class, and normalise by dividing by the total.

$$\hat{p}(C=k) = \frac{1}{N}\sum_{n=1}^{N} \delta(y_n, k)$$

where the "hat" notation indicates we are using a finite sample to estimate the underlying probability of class membership.

Defining the Likelihood

We now need an expression for the likelihood $p(\mathbf{x}|C)$, i.e. a probability distribution over the input data given the class. Here we are trying to describe how the data was generated.

The task of density estimation falls under the category of *unsupervised learning*, which we shall consider later in this course.

For now, let's have a look at an approach where we assume a particular parametric form for this likelihood function.

Gaussian Approximation

Let's assume for now that our likelihoods might be well represented using multivariate Gaussian distributions, such that

$$p(\mathbf{x}|C=k) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_k|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

It's clear that the next task is to estimate the mean vector, $\boldsymbol{\mu}_k$, and covariance matrix, $\boldsymbol{\Sigma}_k$, of this estimate of the class conditional likelihood, $\hat{p}(\mathbf{x}|C=k)$, which we can then use to construct a discriminant function.

Gaussian Approximation

Making this Gaussian assumption, we can rewrite the discriminant function with a bit of straightforward algebra,

$$\begin{aligned}\log \frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} &= \log \frac{p(\mathbf{x}|C=1)}{p(\mathbf{x}|C=0)} + \log \frac{p(C=1)}{p(C=0)} \\ &= \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} + \boldsymbol{\theta}^T \mathbf{x} + b_0\end{aligned}$$

where $\mathbf{A} = \boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1}$ and $\boldsymbol{\theta} = \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0$, and

$$b_0 = \log \frac{p(C=1)}{p(C=0)} + \frac{1}{2}\log \frac{|\boldsymbol{\Sigma}_0|}{|\boldsymbol{\Sigma}_1|} + \frac{1}{2}(\boldsymbol{\mu}_0^T \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0^T - \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1^T)$$

The discriminant function that we get as a result is a quadratic function of the inputs, $\mathbf{x}$, with a quadratic decision boundary.

Gaussian Approximation

We need to decide how to estimate the covariance matrix for our Gaussian approximation.

- Case I: $\boldsymbol{\Sigma}_j = \sigma^2 I$
  - All classes have the *same* variance, i.e. a *constant* diagonal covariance matrix.
  - The resulting classifier is linear.
- Case II: $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}$
  - All classes share the *same* covariance matrix.
  - The resulting classifier is linear.
- Case III: $\boldsymbol{\Sigma}_j$
  - Each class has its *own* covariance matrix.
  - The resulting classifier is quadratic.

Classification Approaches

We have already seen how we may construct a classification algorithm by considering the discriminant function.

We may adopt a discriminative approach by modelling it directly using some linear model,

$$\log \frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} = \boldsymbol{\theta}^T \phi(\mathbf{x})$$

Alternatively, using a generative approach we may define the discriminant function by directly estimating the posterior ratio from the likelihood and prior terms,

$$\log \frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} = \log \frac{p(\mathbf{x}|C=1)p(C=1)}{p(\mathbf{x}|C=0)p(C=0)}$$

Defining the Prior

The main challenge we now face is how should we define our prior distributions?

Given our data set $(\mathbf{x}, \mathbf{y})$ we could simply estimate this distribution based on the data we have. i.e. we could count the number of instances in one class, then the other class, and normalise by dividing by the total.

$$\hat{p}(C = k) = \frac{1}{N} \sum_{n=1}^{N} \delta(y_n, k)$$

where the "hat" notation indicates we are using a finite sample to estimate the underlying probability of class membership.

Defining the Likelihood

We now need an expression for the likelihood $p(\mathbf{x}|C)$, i.e. a probability distribution over the input data given the class. Here we are trying to describe how the data was generated.

The task of density estimation falls under the category of *unsupervised learning*, which we shall consider later in this course.

For now, let's have a look at an approach where we assume a particular parametric form for this likelihood function.

Gaussian Approximation

Let's assume for now that our likelihoods might be well represented using multivariate Gaussian distributions, such that

$$p(\mathbf{x}|C=k) = \frac{1}{\sqrt{(2\pi)^D|\mathbf{\Sigma}_k|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

It's clear that the next task is to estimate the mean vector, $\boldsymbol{\mu}_k$, and covariance matrix, $\mathbf{\Sigma}_k$, of this estimate of the class conditional likelihood, $\hat{p}(\mathbf{x}|C=k)$, which we can then use to construct a discriminant function.

Gaussian Approximation

Making this Gaussian assumption, we can rewrite the discriminant function with a bit of straightforward algebra,

$$\begin{aligned}\log \frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} &= \log \frac{p(\mathbf{x}|C=1)}{p(\mathbf{x}|C=0)} + \log \frac{p(C=1)}{p(C=0)} \\ &= \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \boldsymbol{\theta}^T\mathbf{x} + b_0\end{aligned}$$

where $\mathbf{A} = \Sigma_0^{-1} - \Sigma_1^{-1}$ and $\boldsymbol{\theta} = \Sigma_1^{-1}\boldsymbol{\mu}_1 - \Sigma_0^{-1}\boldsymbol{\mu}_0$, and

$$b_0 = \log \frac{p(C=1)}{p(C=0)} + \frac{1}{2}\log \frac{|\Sigma_0|}{|\Sigma_1|} + \frac{1}{2}(\boldsymbol{\mu}_0^T\Sigma_0^{-1}\boldsymbol{\mu}_0^T - \boldsymbol{\mu}_1^T\Sigma_1^{-1}\boldsymbol{\mu}_1^T)$$

The discriminant function that we get as a result is a quadratic function of the inputs, $\mathbf{x}$, with a quadratic decision boundary.

Gaussian Approximation

We need to decide how to estimate the covariance matrix for our Gaussian approximation.

- Case I: $\Sigma_i = \sigma^2 I$
  - All classes have the *same* variance, i.e. a *constant* diagonal covariance matrix.
  - The resulting classifier is linear.
- Case II: $\Sigma_i = \Sigma$
  - All classes share the *same* covariance matrix.
  - The resulting classifier is linear.
- Case III: Σ_i
  - Each class has its *own* covariance matrix.
  - The resulting classifier is quadratic.

Generative Classification

Classification Approaches

We have already seen how we may construct a classification algorithm by considering the discriminant function.

We may adopt a discriminative approach by modelling it directly using some linear model,

$$\log \frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} = \boldsymbol{\theta}^T \phi(\mathbf{x})$$

Alternatively, using a generative approach we may define the discriminant function by directly estimating the posterior ratio from the likelihood and prior terms,

$$\log \frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} = \log \frac{p(\mathbf{x}|C=1)p(C=1)}{p(\mathbf{x}|C=0)p(C=0)}$$

Defining the Prior

The main challenge we now face is how should we define our prior distributions?

Given our data set $(\mathbf{x}, \mathbf{y})$ we could simply estimate this distribution based on the data we have. i.e. we could count the number of instances in one class, then the other class, and normalise by dividing by the total.

$$\hat{p}(C=k) = \frac{1}{N}\sum_{n=1}^{N} \delta(y_n, k)$$

where the "hat" notation indicates we are using a finite sample to estimate the underlying probability of class membership.

Defining the Likelihood

We now need an expression for the likelihood $p(\mathbf{x}|C)$, i.e. a probability distribution over the input data given the class. Here we are trying to describe how the data was generated.

The task of density estimation falls under the category of *unsupervised learning*, which we shall consider later in this course.

For now, let's have a look at an approach where we assume a particular parametric form for this likelihood function.

Gaussian Approximation

Let's assume for now that our likelihoods might be well represented using multivariate Gaussian distributions, such that

$$p(\mathbf{x}|C=k) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_k|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

It's clear that the next task is to estimate the mean vector, $\boldsymbol{\mu}_k$, and covariance matrix, $\boldsymbol{\Sigma}_k$, of this estimate of the class conditional likelihood, $\hat{p}(\mathbf{x}|C=k)$, which we can then use to construct a discriminant function.

Gaussian Approximation

Making this Gaussian assumption, we can rewrite the discriminant function with a bit of straightforward algebra,

$$\begin{aligned}\log \frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} &= \log \frac{p(\mathbf{x}|C=1)}{p(\mathbf{x}|C=0)} + \log \frac{p(C=1)}{p(C=0)} \\ &= \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} + \boldsymbol{\theta}^T \mathbf{x} + b_0\end{aligned}$$

where $\mathbf{A} = \boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1}$ and $\boldsymbol{\theta} = \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0$, and

$$b_0 = \log \frac{p(C=1)}{p(C=0)} + \frac{1}{2}\log \frac{|\boldsymbol{\Sigma}_0|}{|\boldsymbol{\Sigma}_1|} + \frac{1}{2}(\boldsymbol{\mu}_0^T \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0^T - \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1^T)$$

The discriminant function that we get as a result is a quadratic function of the inputs, $\mathbf{x}$, with a quadratic decision boundary.

Gaussian Approximation

We need to decide how to estimate the covariance matrix for our Gaussian approximation.

- Case I: $\boldsymbol{\Sigma}_j = \sigma^2 I$
  - All classes have the *same* variance, i.e. a *constant* diagonal covariance matrix.
  - The resulting classifier is linear.
- Case II: $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}$
  - All classes share the *same* covariance matrix.
  - The resulting classifier is linear.
- Case III: $\boldsymbol{\Sigma}_j$
  - Each class has its *own* covariance matrix.
  - The resulting classifier is quadratic.

Gaussian Approximation

Case I

Case I: $\Sigma_i = \sigma^2 I$

When $\Sigma_i = \sigma^2 I$ for all classes, all data points within each class are independent and form equal-sized hyperspherical clusters.

Under this assumption it is clear that,

$$|\Sigma_i| = \sigma^{2p} \qquad \Sigma^{-1} = \frac{1}{\sigma^2} I$$

We can write out the full expression, ignoring any constants of proportionality.

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}-\mu_i)^T(\mathbf{x}-\mu_i) + \log p(C_i)$$
$$= -\frac{1}{2\sigma^2}(\mathbf{x}^T\mathbf{x} - 2\mu_i^T\mathbf{x} + \mu_i^T\mu_i) + \log p(C_i)$$

Note that the quadratic term in red is an additive constant since it does not depend on i

Case I

Case I: $\Sigma_i = \sigma^2 I$

The resulting classifier is linear, because it has the form

$$g_i(\mathbf{x}) = \theta_i^T \mathbf{x} + b_{i0}$$

with parameters

$$\theta_i = \frac{1}{\sigma^2}\mu_i$$

and

$$b_{i0} = -\frac{1}{2\sigma^2}\mu_i^T\mu_i + \log p(C_i)$$

The decision surfaces are hyperplanes defined by

$$g_i(\mathbf{x}) = g_j(\mathbf{x})$$

Case I

Case I: $\Sigma_i = \sigma^2 I$

The linear equation $g_i(\mathbf{x}) = g_j(\mathbf{x})$ can be written as

$$\theta^T(\mathbf{x} - \mathbf{x}_0) = 0$$

where

$$\theta = (\mu_i - \mu_j)$$

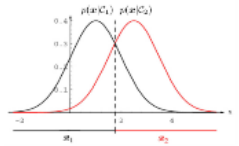
and

$$\mathbf{x}_0 = \frac{1}{2}(\mu_i - \mu_j) - \frac{\sigma^2}{||\mu_i - \mu_j||^2}\log\frac{p(C_i)}{p(C_j)}(\mu_i - \mu_j)$$

The equation defines a hyperplane passing through the point x_0 and orthogonal to θ.

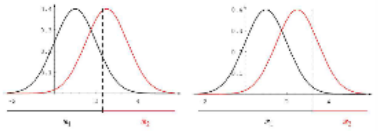
Since θ is in the same direction of $\mu_i - \mu_j$, the hyperplane is orthogonal to the line separating the two means.

Case I: Example



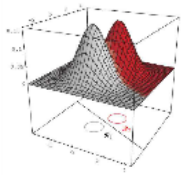
1 dimensional with equal priors (p(C) = 0.5)

Case I: Example



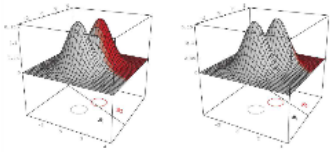
1 dimensional with different priors.
p(C1) = 0.7 (right) and p(C1) = 0.9 (right).

Case I: Example



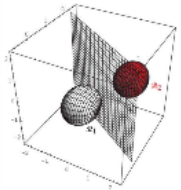
2 dimensional with equal priors.

Case I: Example



2 dimensional with different priors.
p(C1) = 0.8 (right) and p(C1) = 0.99 (right).

Case I: Example



3 dimensional with equal priors.

Case I

Case I: $\mathbf{\Sigma}_i = \sigma^2 I$

When $\mathbf{\Sigma}_i = \sigma^2 I$ for all classes, all data points within each class are independent and form equal-sized hyperspherical clusters.

Under this assumption it is clear that,

$$|\mathbf{\Sigma}_i| = \sigma^{2p} \qquad \mathbf{\Sigma}^{-1} = \frac{1}{\sigma^2} I$$

We can write out the full expression, ignoring any constants of proportionality,

$$\begin{aligned} g_i(\mathbf{x}) &= -\frac{1}{2\sigma^2}(\mathbf{x} - \mu_i)^T(\mathbf{x} - \mu_i) + \log p(C_i) \\ &= -\frac{1}{2\sigma^2}(\mathbf{x}^T\mathbf{x} - 2\mu_i^T\mathbf{x} + \mu_i^T\mu_i) + \log p(C_i) \end{aligned}$$

Note that the quadratic term in red is an additive constant since it does not depend on i.

Case I

Case I: $\boldsymbol{\Sigma}_i = \sigma^2 I$

The resulting classifier is linear, because it has the form

$$g_i(\mathbf{x}) = \boldsymbol{\theta}_i^T \mathbf{x} + b_{0i}$$

with parameters

$$\boldsymbol{\theta}_i = \frac{1}{\sigma^2} \boldsymbol{\mu}_i$$

and

$$b_{0i} = -\frac{1}{2\sigma^2} \boldsymbol{\mu}_i^T \boldsymbol{\mu}_i + \log p(C_i)$$

The decision surfaces are hyperplanes defined by

$$g_i(\mathbf{x}) = g_j(\mathbf{x})$$

Case I

Case I: $\boldsymbol{\Sigma}_i = \sigma^2 I$

The linear equation $g_i(\mathbf{x}) = g_j(\mathbf{x})$ can be written as

$$\boldsymbol{\theta}^T(\mathbf{x} - \mathbf{x}_0) = 0$$

where

$$\boldsymbol{\theta} = (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$$

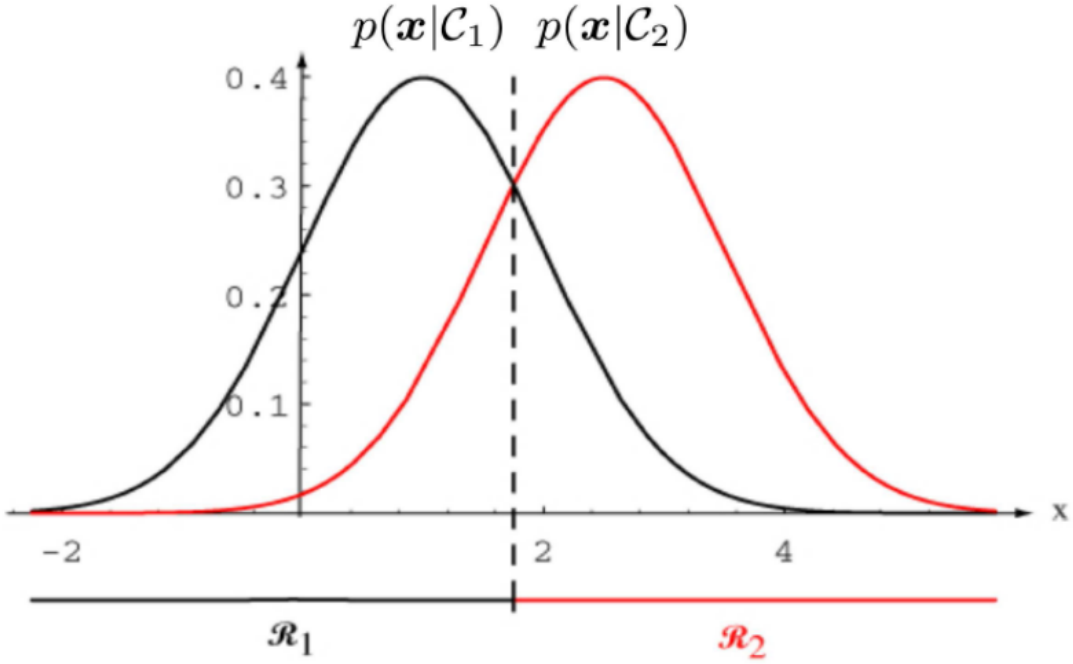
and

$$\mathbf{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) - \frac{\sigma^2}{||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||^2} \log \frac{p(C_i)}{p(C_j)}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$$

The equation defines a hyperplane passing through the point x_0 and orthogonal to $\boldsymbol{\theta}$.

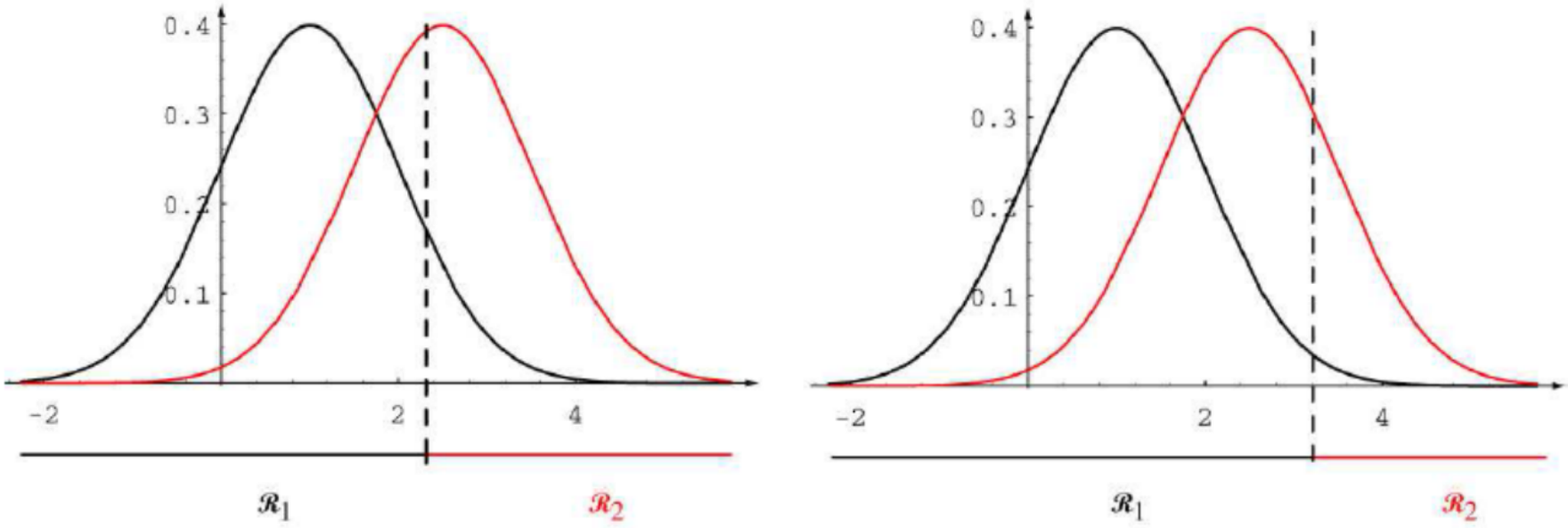
Since $\boldsymbol{\theta}$ is in the same direction of $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j$, the hyperplane is orthogonal to the line separating the two means.

Case I: Example



1 dimensional with equal priors (p(C) = 0.5)

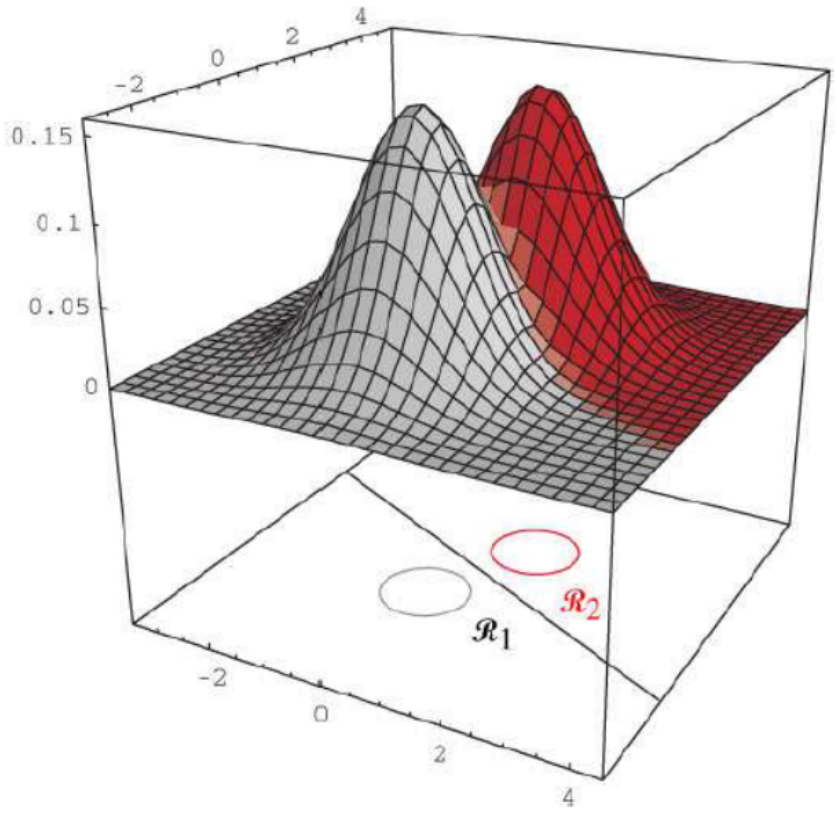
Case I: Example



1 dimensional with different priors.

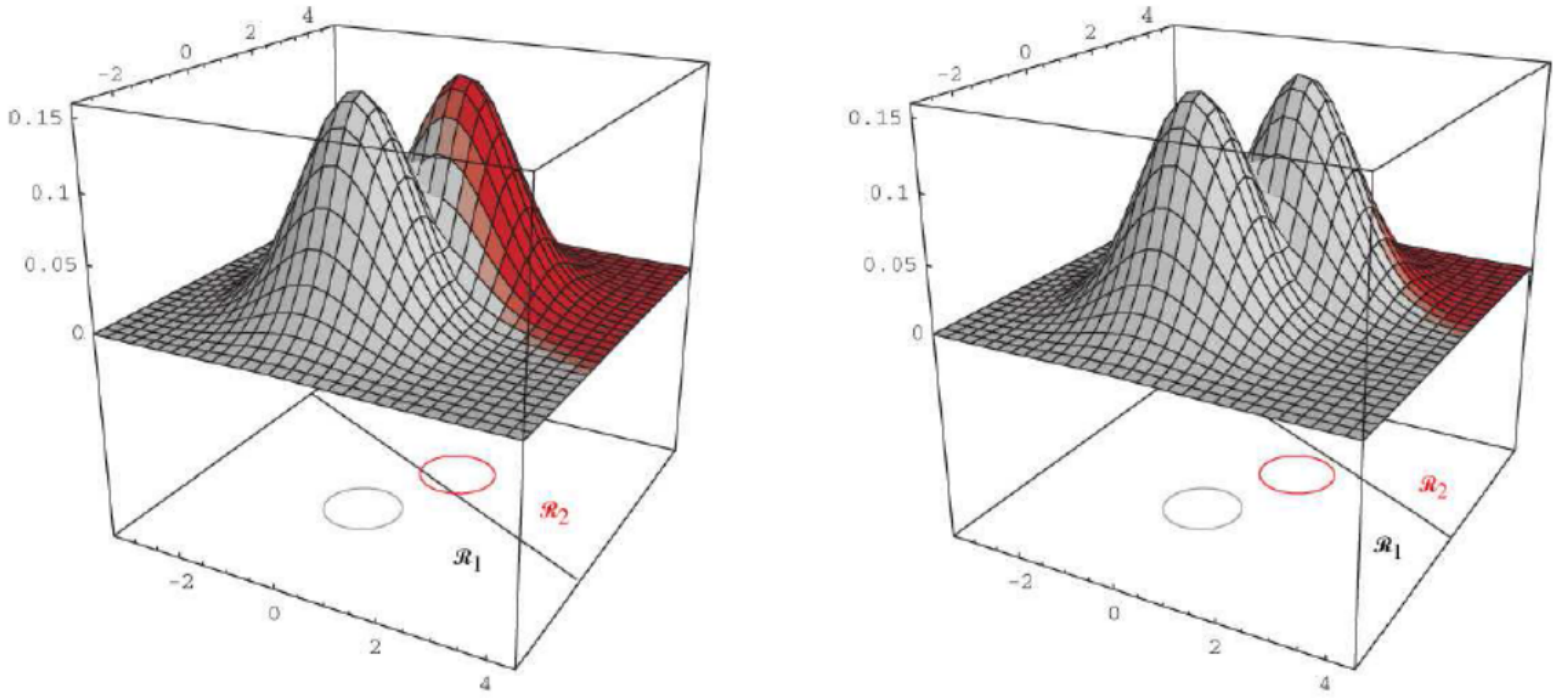
p(C1) = 0.7 (right) and p(C1) = 0.9 (right).

Case I: Example



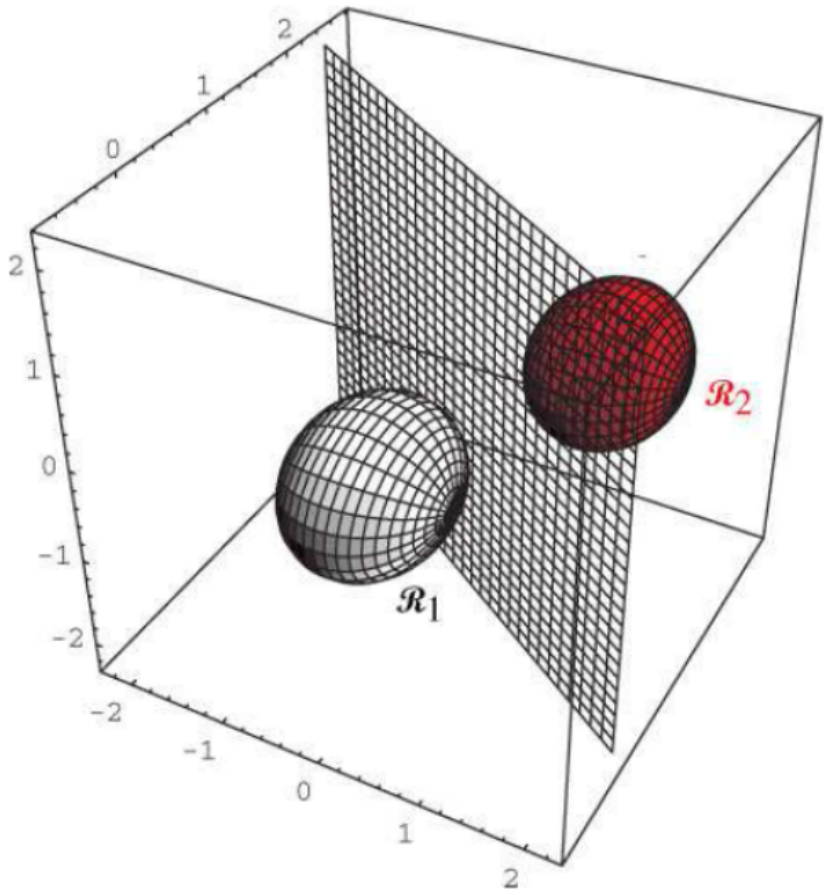
2 dimensional with equal priors.

Case I: Example



2 dimensional with different priors.
p(C1) = 0.8 (right) and p(C1) = 0.99 (right).

Case I: Example



3 dimensional with equal priors.

Gaussian Approximation

Case I

Case I: $\Sigma_i = \sigma^2 I$

When $\Sigma_i = \sigma^2 I$ for all classes, all data points within each class are independent and form equal-sized hyperspherical clusters.

Under this assumption it is clear that,

$$|\Sigma_i| = \sigma^{2p} \qquad \Sigma^{-1} = \frac{1}{\sigma^2} I$$

We can write out the full expression, ignoring any constants of proportionality.

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}-\mu_i)^T(\mathbf{x}-\mu_i) + \log p(C_i)$$
$$= -\frac{1}{2\sigma^2}(\mathbf{x}^T\mathbf{x} - 2\mu_i^T\mathbf{x} + \mu_i^T\mu_i) + \log p(C_i)$$

Note that the quadratic term in red is an additive constant since it does not depend on i

Case I

Case I: $\Sigma_i = \sigma^2 I$

The resulting classifier is linear, because it has the form

$$g_i(\mathbf{x}) = \theta_i^T \mathbf{x} + b_{i0}$$

with parameters

$$\theta_i = \frac{1}{\sigma^2}\mu_i$$

and

$$b_{i0} = -\frac{1}{2\sigma^2}\mu_i^T\mu_i + \log p(C_i)$$

The decision surfaces are hyperplanes defined by

$$g_i(\mathbf{x}) = g_j(\mathbf{x})$$

Case I

Case I: $\Sigma_i = \sigma^2 I$

The linear equation $g_i(\mathbf{x}) = g_j(\mathbf{x})$ can be written as

$$\theta^T(\mathbf{x} - \mathbf{x}_0) = 0$$

where

$$\theta = (\mu_i - \mu_j)$$

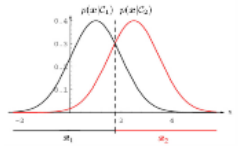
and

$$\mathbf{x}_0 = \frac{1}{2}(\mu_i - \mu_j) - \frac{\sigma^2}{||\mu_i - \mu_j||^2}\log\frac{p(C_i)}{p(C_j)}(\mu_i - \mu_j)$$

The equation defines a hyperplane passing through the point x_0 and orthogonal to θ.

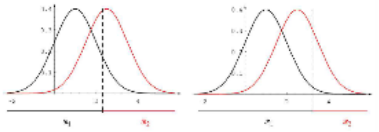
Since θ is in the same direction of $\mu_i - \mu_j$, the hyperplane is orthogonal to the line separating the two means.

Case I: Example



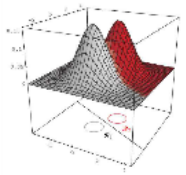
1 dimensional with equal priors (p(C) = 0.5)

Case I: Example



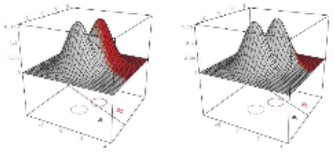
1 dimensional with different priors.
p(C1) = 0.7 (right) and p(C1) = 0.9 (right).

Case I: Example



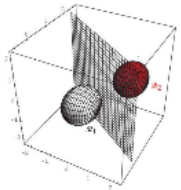
2 dimensional with equal priors.

Case I: Example



2 dimensional with different priors.
p(C1) = 0.8 (right) and p(C1) = 0.99 (right).

Case I: Example



3 dimensional with equal priors.

Gaussian Approximation

Case II

Case II: $\Sigma_i = \Sigma$
Again ignoring any terms not dependent on i,

$$g_i(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma^{-1}(\mathbf{x} - \mu_i) + \log p(C_i) \quad (1)$$

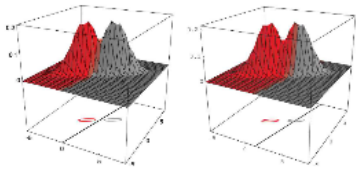
Expanding this equation, the quadratic term $\mathbf{x}^T \Sigma \mathbf{x}$ is independent of i, and we obtain a linear discriminant function

$$g_i(\mathbf{x}) = \theta_i^T \mathbf{x} + b_{i0} \quad (2)$$

where

$$\theta_i = \Sigma^{-1}\mu_i \qquad b_{i0} = -\frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i + \log p(C_i) \quad (3)$$

Case II: Example



2 dimensional example. The separating line is not perpendicular to the line separating the means, however the intersect is half-way through the means if the priors are equal

Case II

Note that when the prior probabilities are the same for all classes, the term $\log p(C_i)$ becomes an additive constant that can be ignored.

The resulting classification rule for these linear classifiers can then be described very simply:

- Measure the Euclidean distance between each $\mathbf{x}$ and each mean vector μ_i.
- Assign $\mathbf{x}$ to the class having the nearest mean.

This is called a *minimal distance classifier*.

Case II

In many cases, the data is not linearly separable and require more complex decision regions that can be obtained by moving away from linearity.

Adding terms involving the product of pairs of components of $\mathbf{x}$ gives a quadratic discriminant function.

$$g(\mathbf{x}) = \theta_0 + \sum_{i=1}^{p} \theta_i x_i + \sum_{i=1}^{p}\sum_{j=1}^{p} \theta_{ij} x_i x_j$$

where we can assume that $\theta_{ij} = \theta_{ji}$.

Quadratic discriminant functions can produce a rich family of decision boundaries - the equation $g(x) = 0$ defines a second degree or *hyperquadratic surface* - however there are an additional $p(p+1)/2$ parameters.

Case II

Case II: $\Sigma_i = \Sigma$

Again ignoring any terms not dependent on i,

$$g_i(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + \log p(C_i) \tag{1}$$

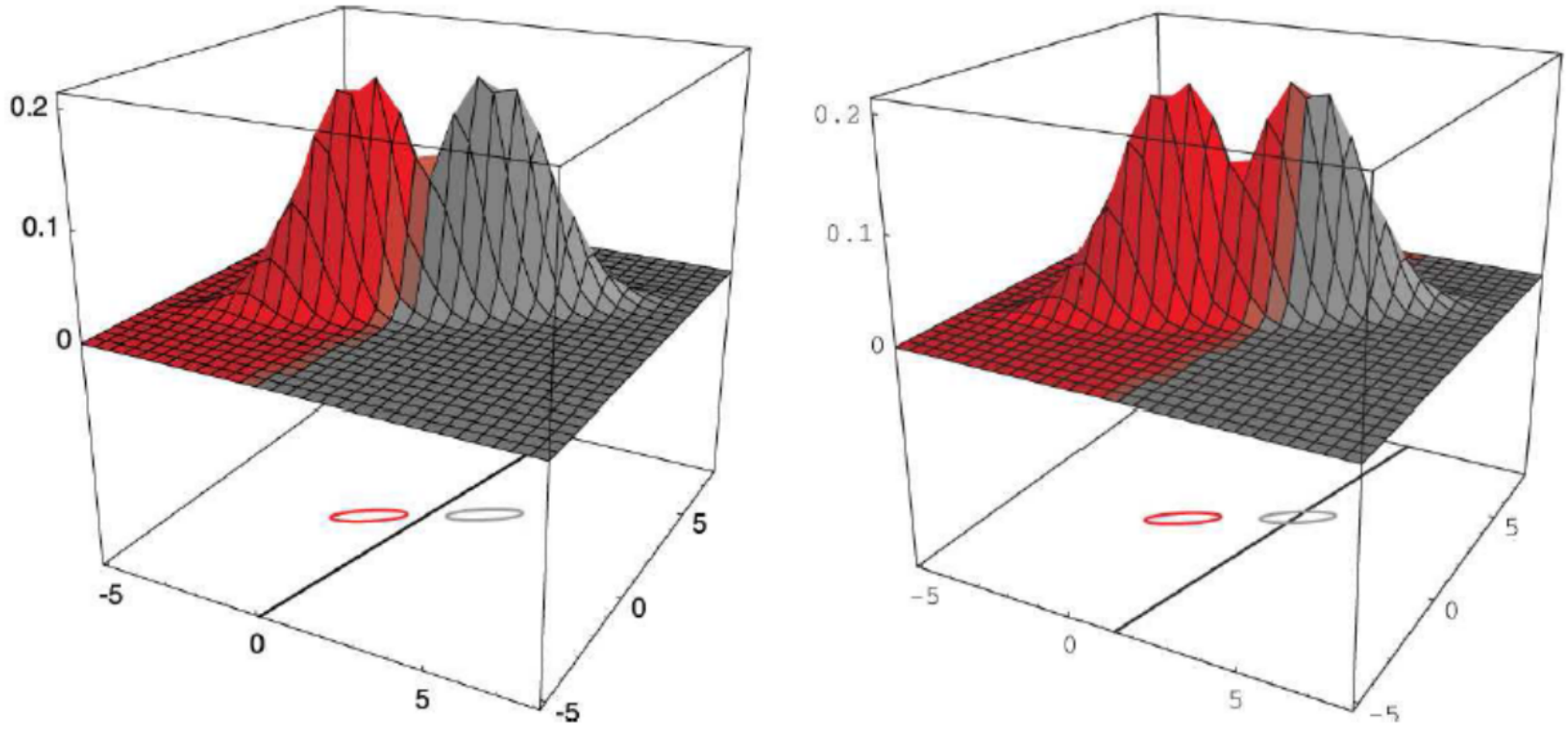
Expanding this equation, the quadratic term $\mathbf{x}^T \Sigma \mathbf{x}$ is independent of i, and we obtain a linear discriminant function

$$g_i(\mathbf{x}) = \boldsymbol{\theta}_i^T \mathbf{x} + b_{0i} \tag{2}$$

where

$$\boldsymbol{\theta}_i = \Sigma^{-1} \boldsymbol{\mu}_i \qquad b_{0i} = -\frac{1}{2} \boldsymbol{\mu}_i^T \Sigma^{-1} \boldsymbol{\mu}_i + \log p(C_i) \tag{3}$$

Case II: Example



2 dimensional example. The separating line is not perpendicular to the line separating the means, however the intersect is half-way through the means if the priors are equal

Case II

Note that when the prior probabilities are the same for all classes, the term $\log p(C_i)$ becomes an additive constant that can be ignored.

The resulting classification rule for these linear classifiers can then be described very simply:

- Measure the Euclidean distance between each $\mathbf{x}$ and each mean vector $\boldsymbol{\mu}_i$.
- Assign $\mathbf{x}$ to the class having the nearest mean.

This is called a *minimal distance classifier*.

Case II

In many cases, the data is not linearly separable and require more complex decision regions that can be obtained by moving away from linearity.

Adding terms involving the product of pairs of components of $\mathbf{x}$ gives a quadratic discriminant function.

$$g(\mathbf{x}) = \theta_0 + \sum_{i=1}^{p} \theta_i x_i + \sum_{i=1}^{p} \sum_{j=1}^{p} \theta_{ij} x_i x_j$$

where we can assume that $\theta_{ij} = \theta_{ji}$.

Quadratic discriminant functions can produce a rich family of decision boundaries - the equation $g(x) = 0$ defines a second degree or *hyperquadratic surface* - however there are an additional $p(p+1)/2$ parameters.

Gaussian Approximation

Case II

Case II: $\Sigma_i = \Sigma$
Again ignoring any terms not dependent on i,

$$g_i(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma^{-1}(\mathbf{x} - \mu_i) + \log p(C_i) \quad (1)$$

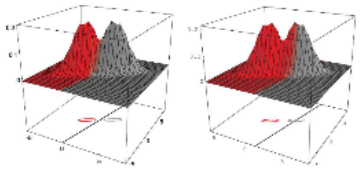
Expanding this equation, the quadratic term $\mathbf{x}^T \Sigma \mathbf{x}$ is independent of i, and we obtain a linear discriminant function

$$g_i(\mathbf{x}) = \theta_i^T \mathbf{x} + b_{i0} \quad (2)$$

where

$$\theta_i = \Sigma^{-1}\mu_i \qquad b_{i0} = -\frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i + \log p(C_i) \quad (3)$$

Case II: Example



2 dimensional example. The separating line is not perpendicular to the line separating the means, however the intersect is half-way through the means if the priors are equal

Case II

Note that when the prior probabilities are the same for all classes, the term $\log p(C_i)$ becomes an additive constant that can be ignored.

The resulting classification rule for these linear classifiers can then be described very simply:

- Measure the Euclidean distance between each $\mathbf{x}$ and each mean vector μ_i.
- Assign $\mathbf{x}$ to the class having the nearest mean.

This is called a *minimal distance classifier*.

Case II

In many cases, the data is not linearly separable and require more complex decision regions that can be obtained by moving away from linearity.

Adding terms involving the product of pairs of components of $\mathbf{x}$ gives a quadratic discriminant function.

$$g(\mathbf{x}) = \theta_0 + \sum_{i=1}^{p} \theta_i x_i + \sum_{i=1}^{p}\sum_{j=1}^{p} \theta_{ij} x_i x_j$$

where we can assume that $\theta_{ij} = \theta_{ji}$.

Quadratic discriminant functions can produce a rich family of decision boundaries - the equation $g(x) = 0$ defines a second degree or *hyperquadratic surface* - however there are an additional $p(p+1)/2$ parameters.

Gaussian Approximation

Case III

Case III: Σ_i

We model each class conditional density with a multivariate Gaussian distribution with its own covariance matrix.

Ignoring the constant term in the Gaussian, we obtain

$$g_i(\mathbf{x}) = \mathbf{x}^T \Theta_i \mathbf{x} + \theta_i^T \mathbf{x} + b_{i0}$$

where

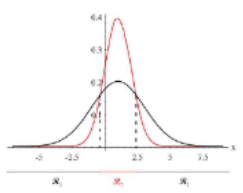
$$\Theta = -\frac{1}{2}\Sigma_i^{-1} \qquad \theta_i = \Sigma_i^{-1}\mu_i \tag{4}$$

and

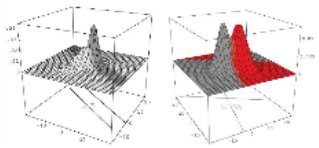
$$b_{i0} = \frac{1}{2}\mu_i^T \Sigma_i^{-1}\mu_i - \frac{1}{2}\log|\Sigma_i| + \log p(C_i) \tag{5}$$

In the two-class case, the decision surfaces are hyperquadratics and can assume general geometric forms.

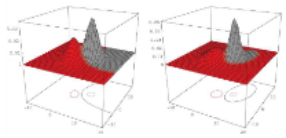
Case III: Example



Case III: Example



Case III: Example



Case III

Case III: Σ_i

We model each class conditional density with a multivariate Gaussian distribution with its own covariance matrix.

Ignoring the constant term in the Gaussian, we obtain

$$g_i(\mathbf{x}) = \mathbf{x}^T \Theta_i \mathbf{x} + \boldsymbol{\theta}_i^T \mathbf{x} + b_{0i}$$

where

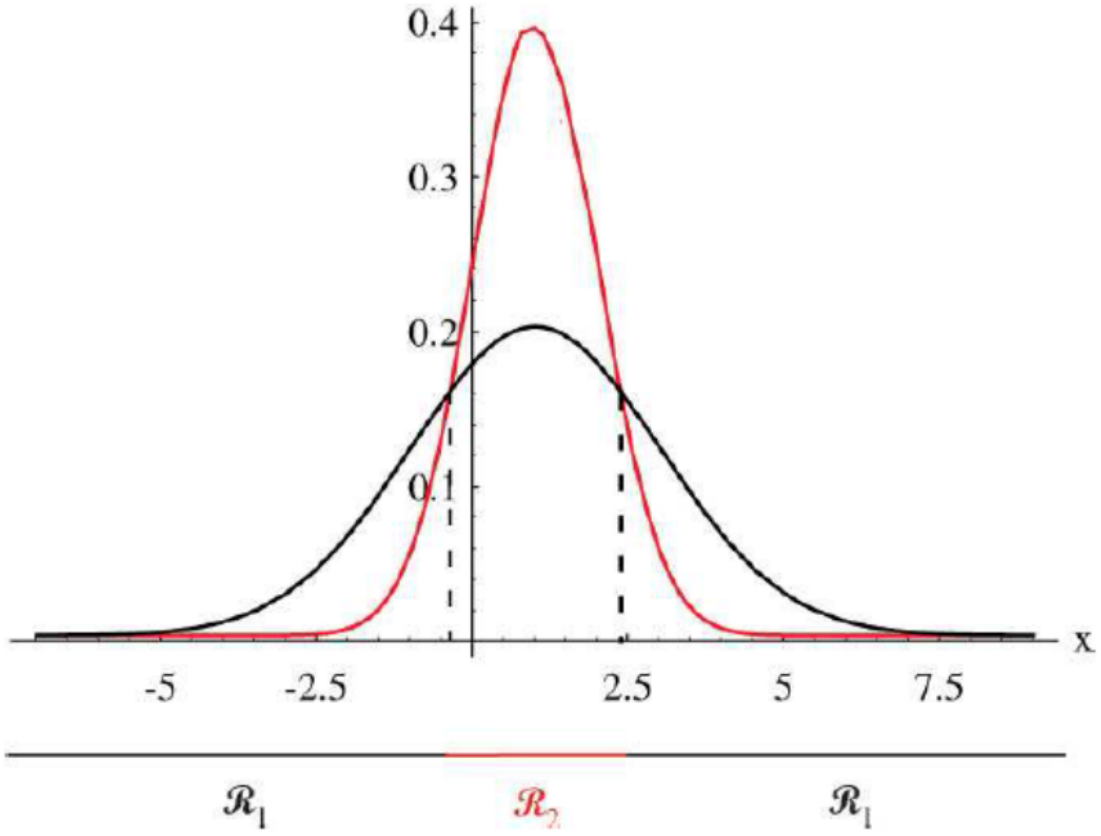
$$\Theta = -\frac{1}{2}\Sigma_i^{-1} \qquad \boldsymbol{\theta}_i = \Sigma_i^{-1}\boldsymbol{\mu}_i \tag{4}$$

and

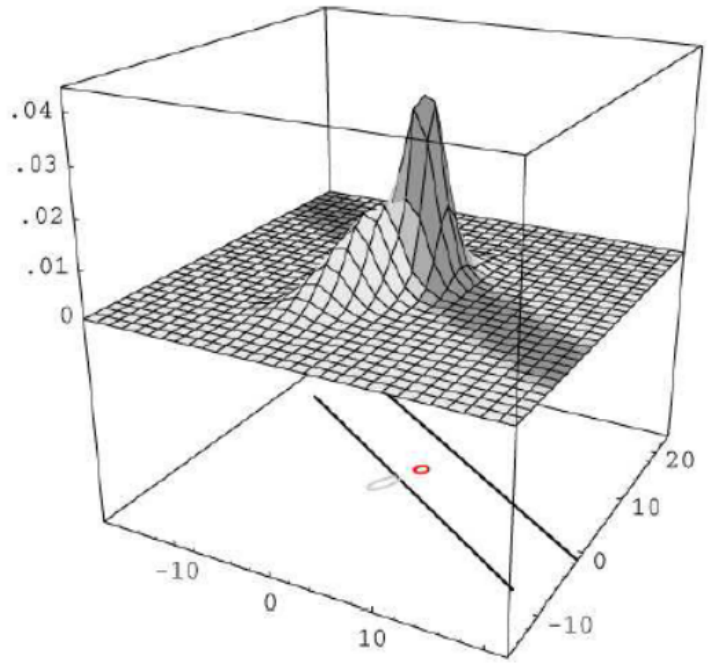
$$b_{0i} = \frac{1}{2}\boldsymbol{\mu}_i^T \Sigma_i^{-1} \boldsymbol{\mu}_i - \frac{1}{2}\log|\Sigma_i| + \log p(C_i) \tag{5}$$

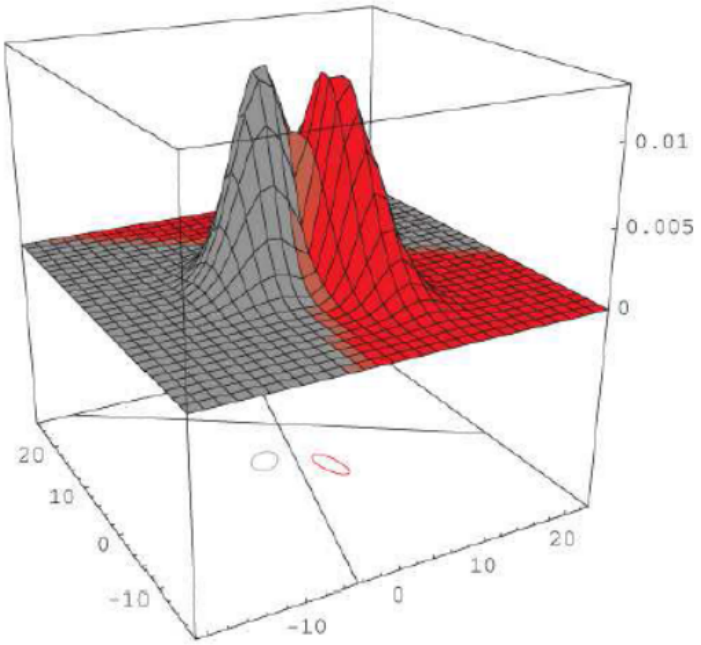
In the two-class case, the decision surfaces are hyperquadratics and can assume general geometric forms.

Case III: Example

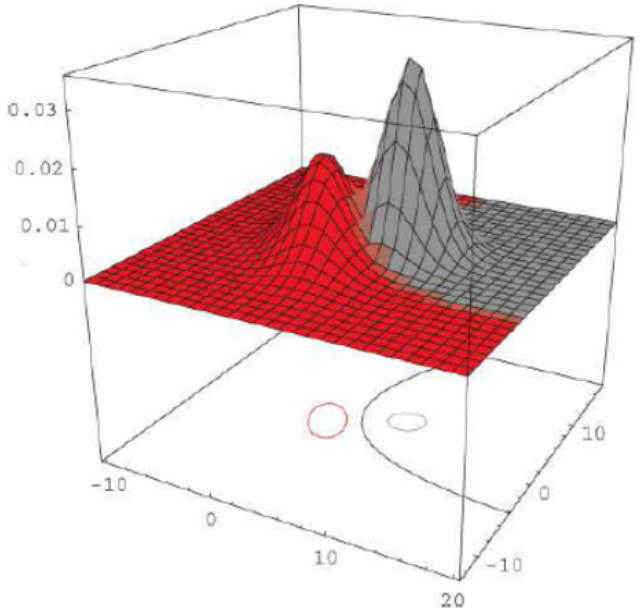


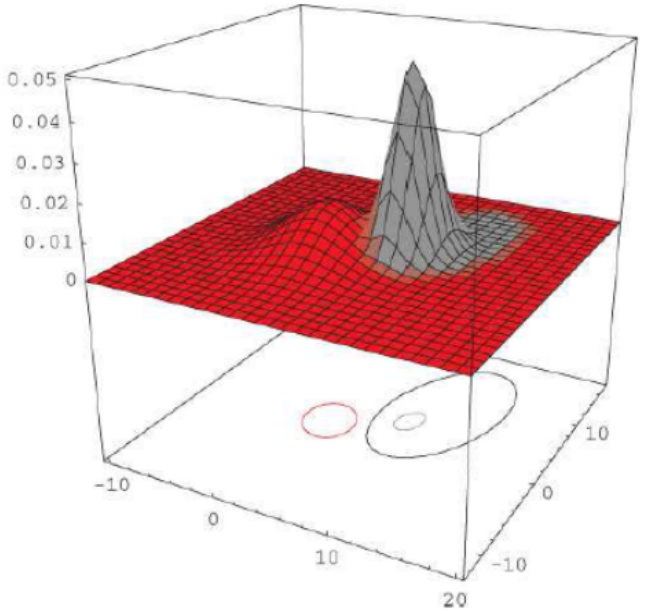
Case III: Example





Case III: Example





Gaussian Approximation

Case III

Case III: Σ_i

We model each class conditional density with a multivariate Gaussian distribution with its own covariance matrix.

Ignoring the constant term in the Gaussian, we obtain

$$g_i(\mathbf{x}) = \mathbf{x}^T \Theta_i \mathbf{x} + \theta_i^T \mathbf{x} + b_{i0}$$

where

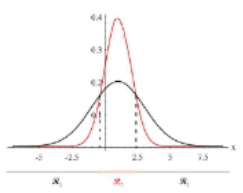
$$\Theta = -\frac{1}{2}\Sigma_i^{-1} \qquad \theta_i = \Sigma_i^{-1}\mu_i \tag{4}$$

and

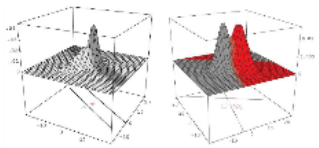
$$b_{i0} = \frac{1}{2}\mu_i^T \Sigma_i^{-1}\mu_i - \frac{1}{2}\log|\Sigma_i| + \log p(C_i) \tag{5}$$

In the two-class case, the decision surfaces are hyperquadratics and can assume general geometric forms.

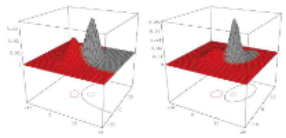
Case III: Example



Case III: Example



Case III: Example



Summary

Summary

In summary, we can employ a generative approach by directly defining the class priors and class conditional likelihoods, however this means we must decide on what the most appropriate approximations are for these distributions.

Ultimately we are really only interested in the discriminant function for making classification decisions, and so it could be argued that it is better to focus on estimating a functional form the discriminant *directly* by employing a discriminative approach.

Intuitively, it is not clear that making decisions about the parameter values for the underlying generative distributions based on input data for each class may necessarily help in defining the discriminant function.

Indeed covariance estimation may be very difficult when the dimensionality of the input space is very large!

Real World Example

Consider this real world example from the field of *Bioinformatics*.

Microarray data may be used to build a classifier that tries to distinguish between *cancerous* and *healthy* tissue samples.

For each sample is defined by the amount of mRNA that a large number of genes express in either healthy or diseased conditions. The challenging aspect of this application is that there may be in excess of $30,000$ genes, resulting in a $30,000$ dimensional feature vector, $\mathbf{x}$.

Making a (simplifying!) Gaussian assumption for the levels of mRNA still means that we need to estimate a $30,000$ by $30,000$ dimensional covariance matrix, based only on a small (usually < 50) samples.

Naive Bayes

The significant challenges here mean we have to make additional simplifications.

Despite the fact that there will likely be correlations between the features we observe, it is completely impractical to attempt to estimate the full covariance matrix. We are going to make the naive assumption that the covariance matrix is diagonal...

$$\boldsymbol{\Sigma}_k = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_D^2)$$

In this case the multivariate Gaussian reduces to a product of univariate Gaussians, such that

$$p(\mathbf{x}|C=k) = \prod_{d=1}^{D} p(x_d|C=k) = \prod_{d=1}^{D} \mathcal{N}_{x_d}(\mu_d, \sigma_d^2)$$

In fact, this approach works surprisingly well in many cases!

Summary

In summary, we can employ a generative approach by directly defining the class priors and class conditional likelihoods, however this means we must decide on what the most appropriate approximations are for these distributions.

Ultimately we are really only interested in the discriminant function for making classification decisions, and so it could be argued that it is better to focus on estimating a functional form the discriminant *directly* by employing a discriminative approach.

Intuitively, it is not clear that making decisions about the parameter values for the underlying generative distributions based on input data for each class may necessarily help in defining the discriminant function.

Indeed covariance estimation may be very difficult when the dimensionality of the input space is very large!

Real World Example

Consider this real world example from the field of *Bioinformatics*.

Microarray data may be used to build a classifier that tries to distinguish between *cancerous* and *healthy* tissue samples.

For each sample is defined by the amount of mRNA that a large number of genes express in either healthy or diseased conditions. The challenging aspect of this application is that there may be in excess of $30,000$ genes, resulting in a $30,000$ dimensional feature vector, $\mathbf{x}$.

Making a (simplifying!) Gaussian assumption for the levels of mRNA still means that we need to estimate a $30,000$ by $30,000$ dimensional covariance matrix, based only on a small (usually $<$ 50) samples.

Naive Bayes

The significant challenges here mean we have to make additional simplifications.

Despite the fact that there will likely be correlations between the features we observe, it is completely impractical to attempt to estimate the full covariance matrix. We are going to make the naive assumption that the covariance matrix is diagonal...

$$\mathbf{\Sigma}_k = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_D^2)$$

In this case the multivariate Gaussian reduces to a product of univariate Gaussians, such that

$$p(\mathbf{x}|C = k) = \prod_{d=1}^{D} p(x_d|C = k) = \prod_{d=1}^{D} \mathcal{N}_{x_d}(\mu_d, \sigma_d^2)$$

In fact, this approach works surprisingly well in many cases!

Summary

Summary

In summary, we can employ a generative approach by directly defining the class priors and class conditional likelihoods, however this means we must decide on what the most appropriate approximations are for these distributions.

Ultimately we are really only interested in the discriminant function for making classification decisions, and so it could be argued that it is better to focus on estimating a functional form the discriminant *directly* by employing a discriminative approach.

Intuitively, it is not clear that making decisions about the parameter values for the underlying generative distributions based on input data for each class may necessarily help in defining the discriminant function.

Indeed covariance estimation may be very difficult when the dimensionality of the input space is very large!

Real World Example

Consider this real world example from the field of *Bioinformatics*.

Microarray data may be used to build a classifier that tries to distinguish between *cancerous* and *healthy* tissue samples.

For each sample is defined by the amount of mRNA that a large number of genes express in either healthy or diseased conditions. The challenging aspect of this application is that there may be in excess of $30,000$ genes, resulting in a $30,000$ dimensional feature vector, $\mathbf{x}$.

Making a (simplifying!) Gaussian assumption for the levels of mRNA still means that we need to estimate a $30,000$ by $30,000$ dimensional covariance matrix, based only on a small (usually < 50) samples.

Naive Bayes

The significant challenges here mean we have to make additional simplifications.

Despite the fact that there will likely be correlations between the features we observe, it is completely impractical to attempt to estimate the full covariance matrix. We are going to make the naive assumption that the covariance matrix is diagonal...

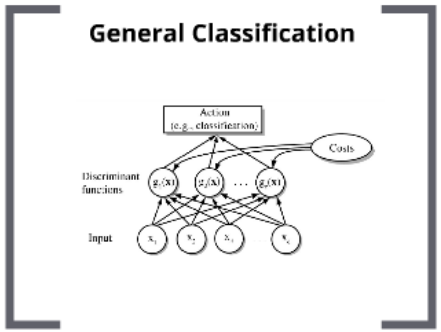
$$\boldsymbol{\Sigma}_k = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_D^2)$$

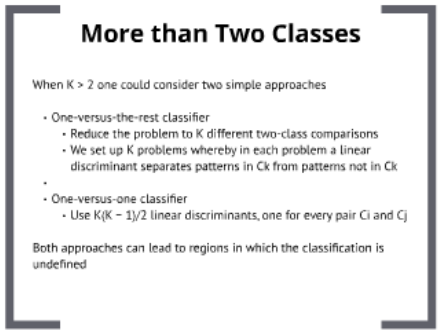
In this case the multivariate Gaussian reduces to a product of univariate Gaussians, such that

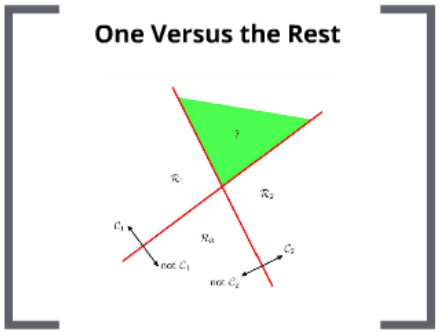
$$p(\mathbf{x}|C=k) = \prod_{d=1}^{D} p(x_d|C=k) = \prod_{d=1}^{D} \mathcal{N}_{x_d}(\mu_d, \sigma_d^2)$$

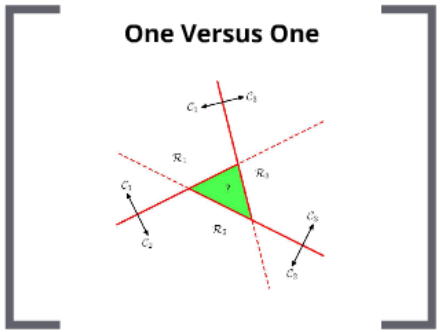
In fact, this approach works surprisingly well in many cases!

General Classification

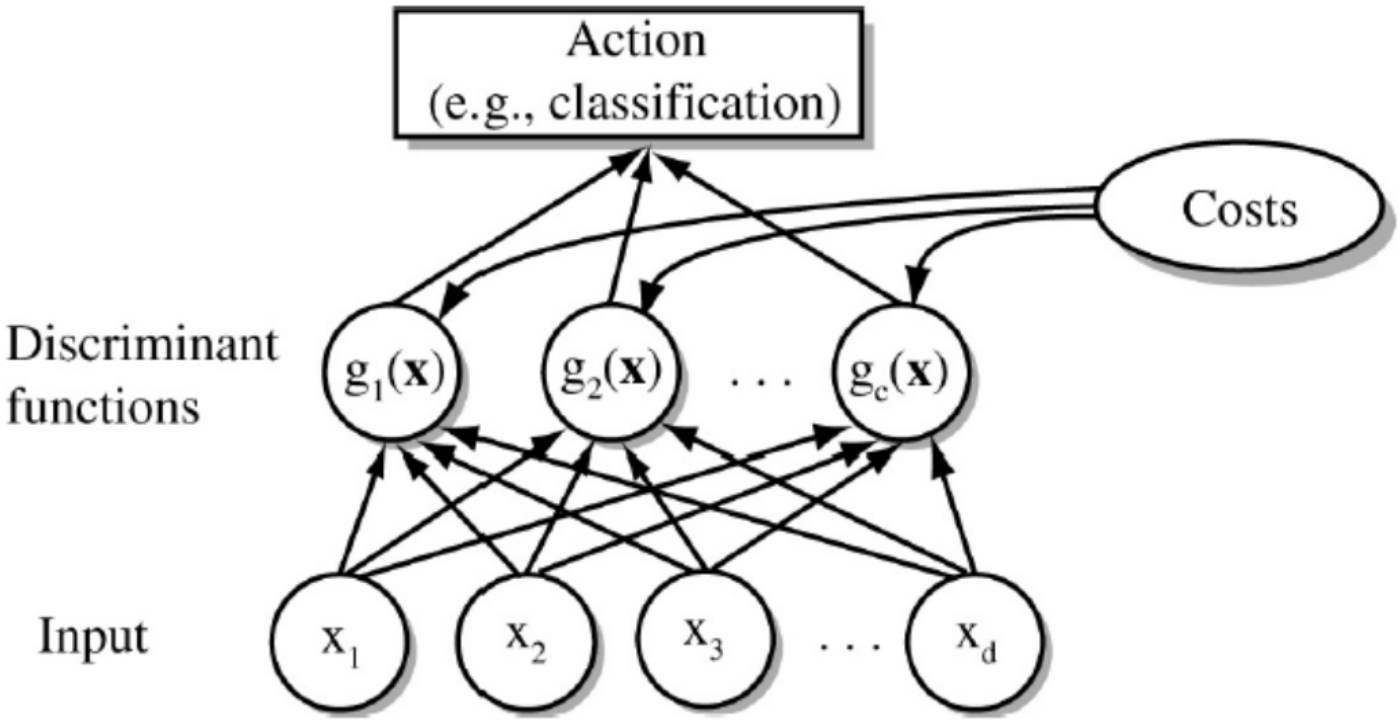








General Classification



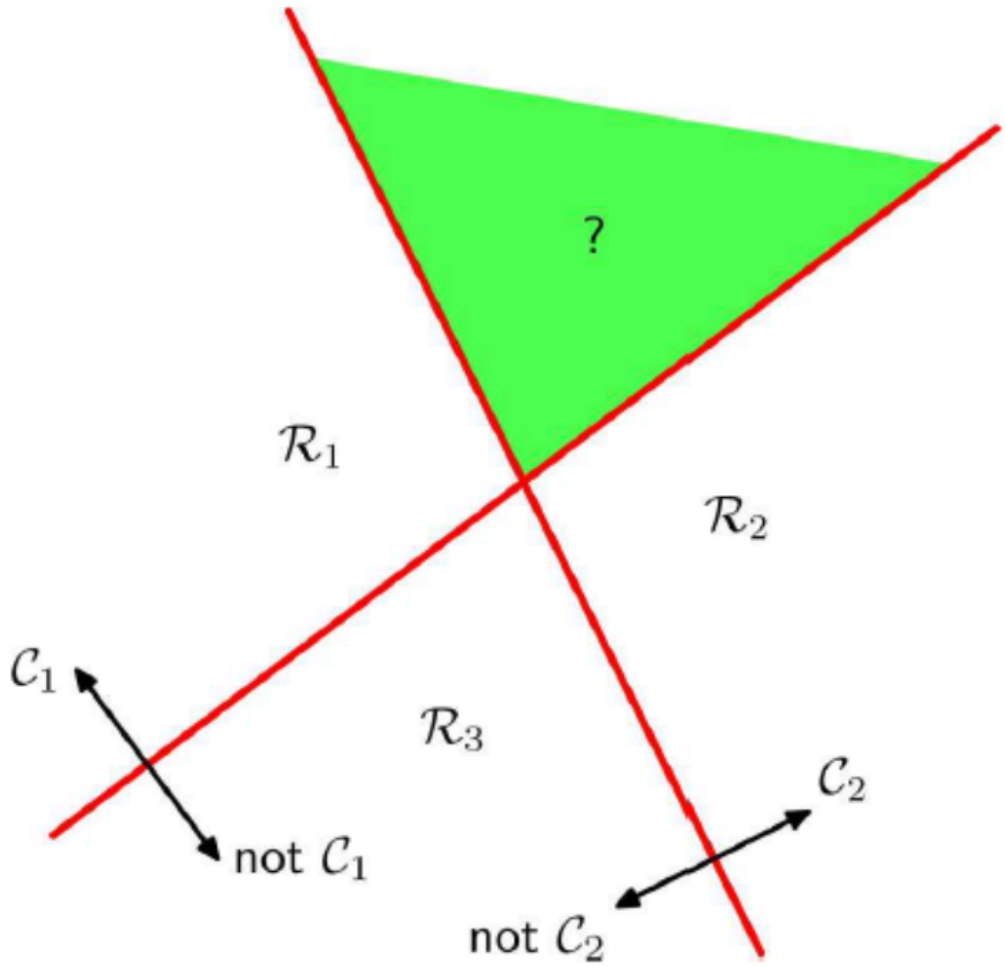
More than Two Classes

When K > 2 one could consider two simple approaches

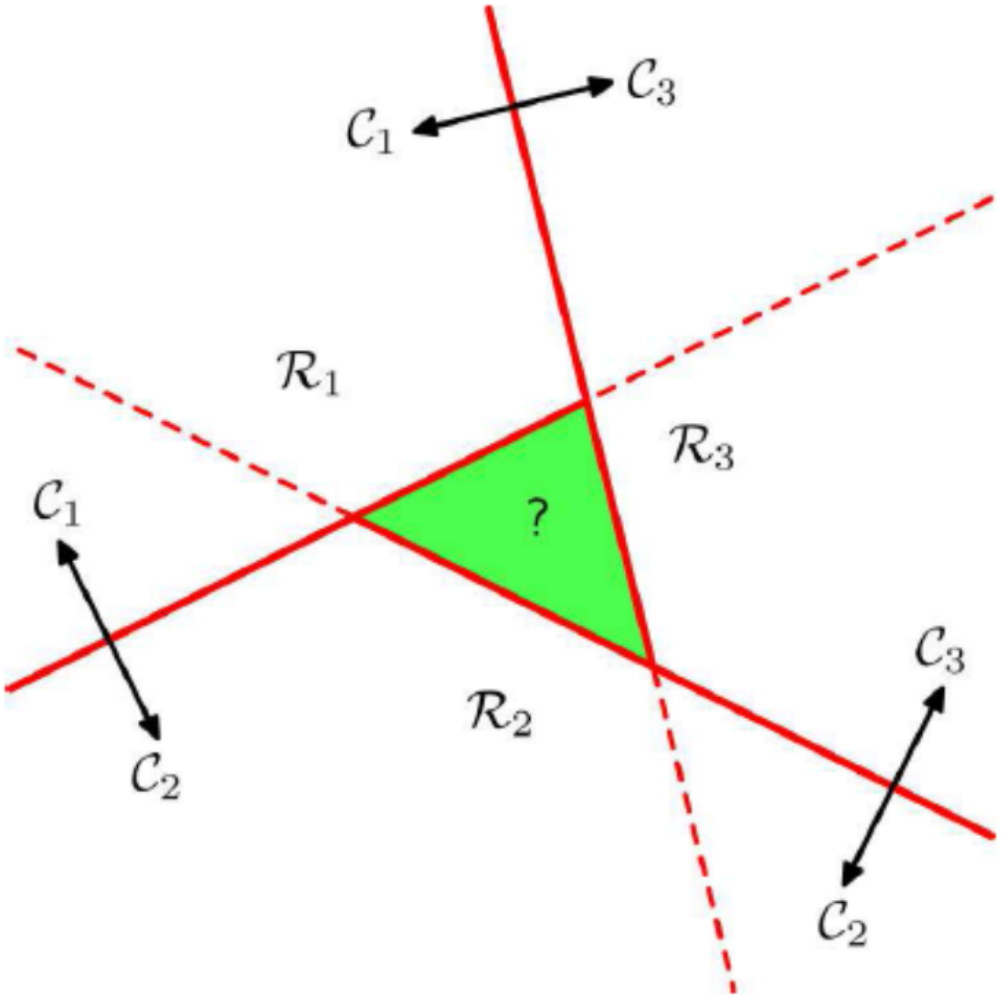
- One-versus-the-rest classifier
  - Reduce the problem to K different two-class comparisons
  - We set up K problems whereby in each problem a linear discriminant separates patterns in Ck from patterns not in Ck
-
- One-versus-one classifier
  - Use K(K − 1)/2 linear discriminants, one for every pair Ci and Cj

Both approaches can lead to regions in which the classification is undefined

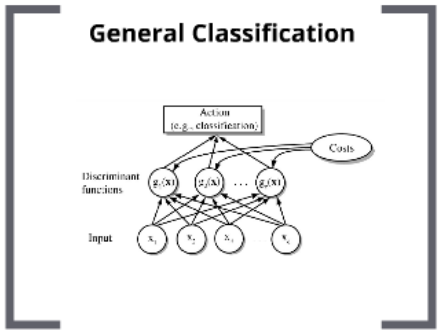
One Versus the Rest

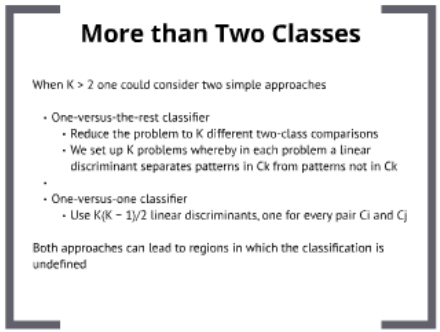


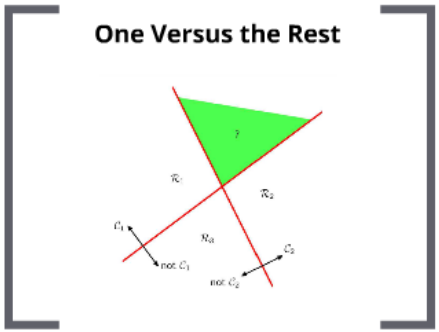
One Versus One

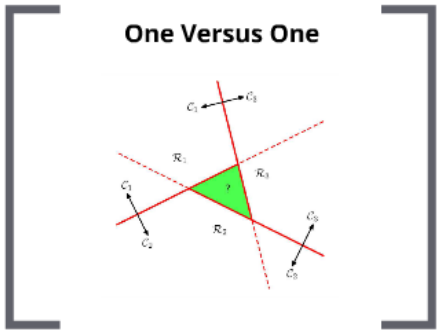


General Classification

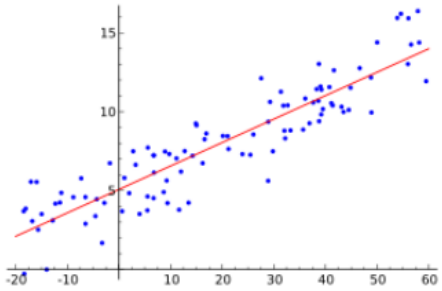




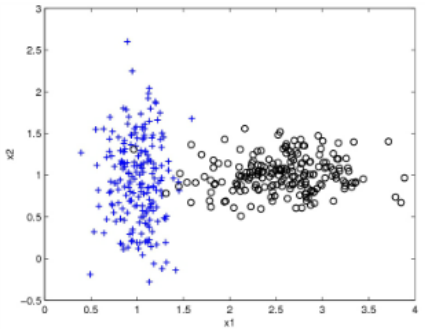




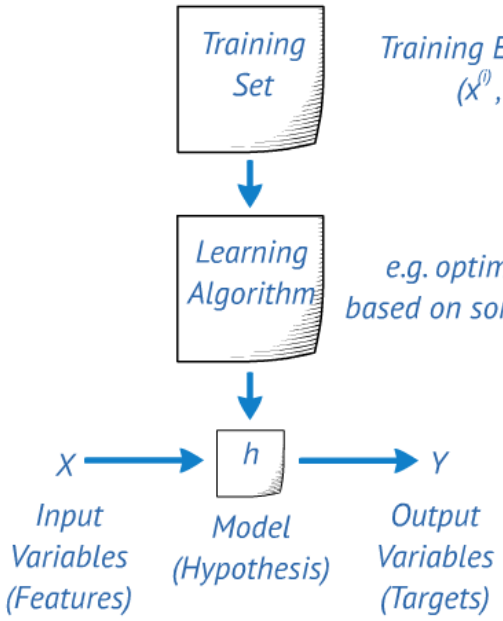
Supervised Learning Recap



Regression - supervised learning in which the labels are continuous values.



Classification - supervised learning in which the labels are from a discrete set of values.



Given a model and some data, we can define a loss function measuring the mismatch between the model output and the observed labels.

We can learn the parameters of the model by minimising the loss function.

Cross validation gives us an approach for choosing the best predictive model.

Computer Lab 1

Training data: *The data points used to estimate the parameters.*

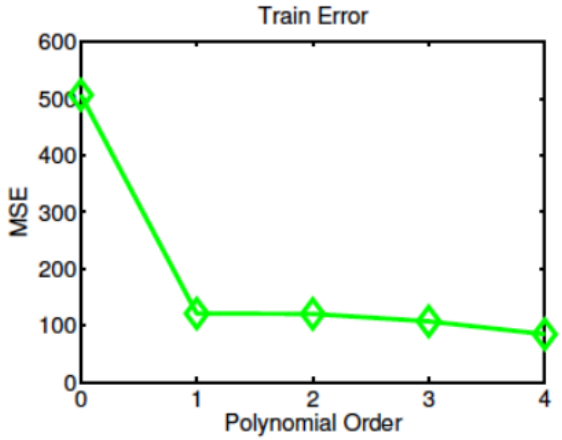
Testing data: *The data points used for calculating the testing error (measured using MSE).*

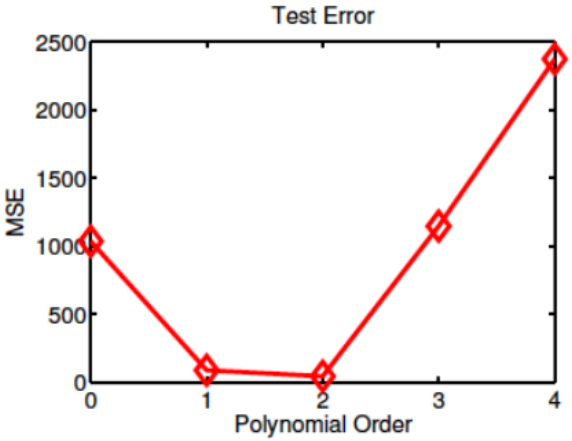
The exercises in the lab were based on splitting up the data into a training set and a test set.

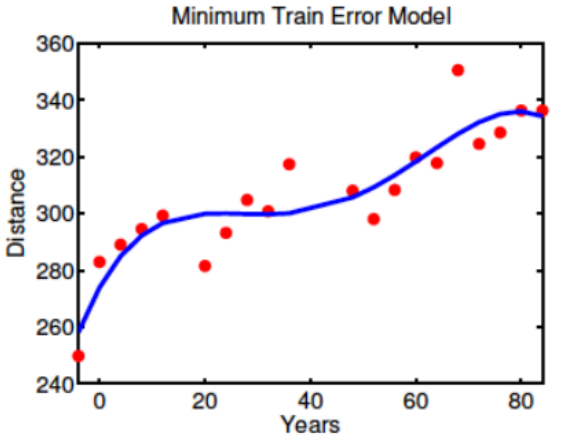
Some key observations:

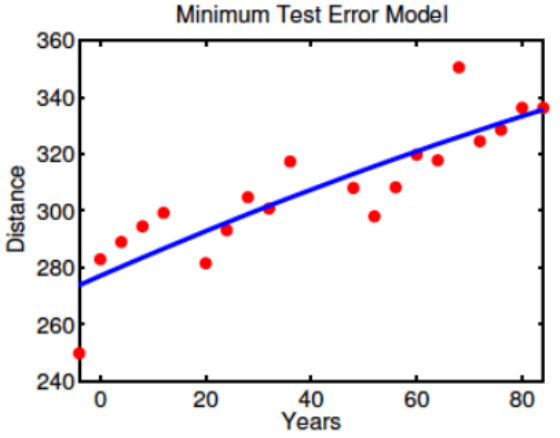
- *Increasing model complexity decreases the MSE calculated using the **training** data.*
- *Increasing model complexity may increase the **testing** error using the MSE.*

Computer Lab 1







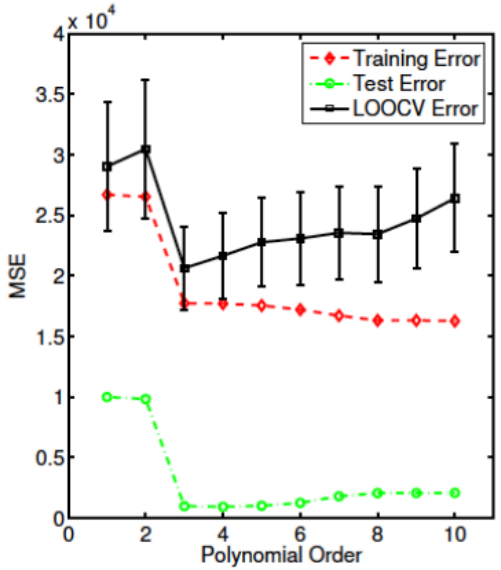


Computer Lab 1

Employing too simple a model results in poor predictions.

However, employing too complex a model may also result in poor predictions due to overfitting.

We should take care about what performance measure we optimise for finding the appropriate parameters for our model, so that the model can generalise its performance beyond the data used for training.



Cross validation provides a better measure of performance by holding back subsets of the data to be used for testing.

M5MS10
Machine Learning

Spring 2018

Lecture 4

Dr Ben Calderhead

b.calderhead@imperial.ac.uk