Applied Machine Learning (AML)

Introduction to Classification

Oisin Mac Aodha • Siddharth N.

Classification

Classification Overview

- In supervised learning, we are tasked with predicting an output y, given an input feature vector x
- For **classification** problems, the output space is a set of mutually exclusive 'classes' (also commonly referred to as 'labels')

Binary versus Multiclass Classification

- In **binary classification** we have two possibilities, e.g. dog versus cat. Thus, $y \in \{0, 1\}$, $y \in \{1, 2\}$, $y \in \{-1, +1\}$, ...
- In **multiclass classification** we can have C possible options, e.g. different breeds of dog. Thus, $y \in \{1, ..., C\}$, where C is the number of classes of interest

Example Classification Problems

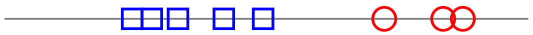
- Spam filtering
- Determining the object present in an image, i.e. image classification
- Fraudulent transaction detection
- Music genre classification
- Medical diagnostic tests
- ...

Example 1D Classification Problem

- We have collected a dataset containing the measurements of the petal lengths (in cm) of plants from two different species: species A and species B
- Thus, we have a one dimensional (1D) continuous measurement $x \in \mathbb{R}$ and a binary class label $y \in \{0, 1\}$

Example 1D Classification Problem

- We have collected a dataset consisting of the measurements of the petal length (in cm) of two different species of plants: species A and species B
- Thus, we have a one dimensional (1D) continuous measurement $x \in \mathbb{R}$ and a binary class label $y \in \{0, 1\}$
- For species A, we have five measurements $\{1.8, 2.1, 2.5, 3.2, 3.8\}$ and for species B we have three $\{5.8, 6.7, 7.0\}$
- We can write our dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N} =$ $\{(1.8, 0), (2.1, 0), (2.5, 0), (3.2, 0), (3.8, 0), (5.8, 1), (6.7, 1), (7.0, 1)\}$



The Generative Approach

- Given a new observation x, can we predict which of the two classes it most likely belongs to?
- To do this, one approach is to fit a **model** to our already observed data
- We can then use this model to make predictions about unobserved (i.e. *new*) data
- For *continuous* features, one obvious choice is the **Gaussian** distribution

Univariate Gaussian Distribution

- The Gaussian (normal) distribution is a very widely used distribution for real-valued random variables, i.e. $x \in \mathbb{R}$
- The probability density function of the Gaussian is defined as

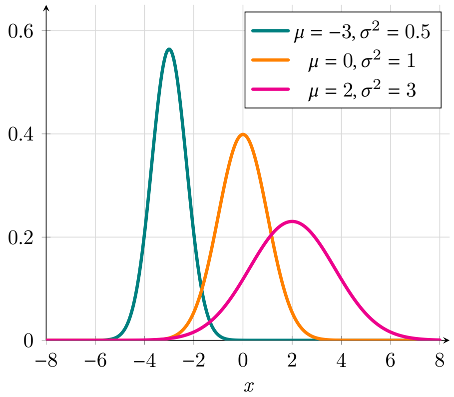
$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

- There are two parameters, the **mean** μ which controls where the distribution is centred and the **variance** σ^2 which controls how wide it is

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

Parameters of the Univariate Gaussian Distribution

- The **mean** μ controls where the distribution is centred and the **variance** σ^2 controls how wide it is



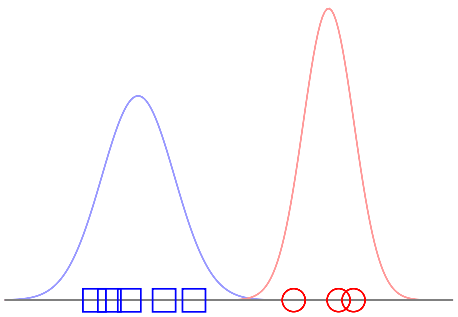
Generative Classifier

- For binary classification, we begin by defining a model for each of our two classes
- We will make the *assumption* that, conditioned on the class, the data is Gaussian distributed

- For data from class 0, we will assume that it is generated from $x|y = 0 \sim \mathcal{N}(x|\mu_0, \sigma_0^2)$
- For data from class 1, we will assume that it is generated from $x|y = 1 \sim \mathcal{N}(x|\mu_1, \sigma_1^2)$

Revisiting the 1D Example

- We can fit our two per-class Gaussians to our dataset $\mathcal{D} = \{(1.8, 0), (2.1, 0), (2.5, 0), (3.2, 0), (3.8, 0), (5.8, 1), (6.7, 1), (7.0, 1)\}$



Generative Classifier - Making Predictions

- Now that we have a model for each class, and assuming that we have estimated the parameters for them (more on this later), we can use them to make predictions
- For a new test datapoint x we can simply assign it to the class with the *largest* output

$$\hat{y} = \arg\max_c \mathcal{N}(x|\mu_c, \sigma_c^2)$$

- We may also want to know how 'likely' it is that a test datapoint is from a given class, e.g. from class 1

$$\hat{p}_1 = \frac{\mathcal{N}(x|\mu_1, \sigma_1^2)}{\mathcal{N}(x|\mu_0, \sigma_0^2) + \mathcal{N}(x|\mu_1, \sigma_1^2)}$$

where $\hat{p}_1 \in [0, 1]$

informatics

Adding 'Prior' Knowledge

- In many cases, we made have prior knowledge that is relevant to our classification problem
- For example, we may have many more observations from one class than another
- We can encode this information as a weighting factor for each class, ϕ_0 and ϕ_1, where $\phi_1, \phi_0 \in [0, 1]$
- In the binary case $\phi_1 = 1 - \phi_0$, i.e. $\phi_0 + \phi_1 = 1$
- We can then combine this with the expression from the previous slide to obtain

$$\hat{p}_1 = \frac{\mathcal{N}(x|\mu_1, \sigma_1^2)\phi_1}{\mathcal{N}(x|\mu_0, \sigma_0^2)\phi_0 + \mathcal{N}(x|\mu_1, \sigma_1^2)\phi_1}$$

informatics

Bayes Classifier

- We came up with the following expression for making predictions for new data

$$\hat{p}_1 = \frac{\mathcal{N}(x|\mu_1, \sigma_1^2)\phi_1}{\mathcal{N}(x|\mu_0, \sigma_0^2)\phi_0 + \mathcal{N}(x|\mu_1, \sigma_1^2)\phi_1}$$

- It turns out that this is just a restatement of **Bayes' rule**

$$p(y = c|x) = \frac{p(x|y = c)p(y = c)}{\sum_{c'} p(x|y = c')p(y = c')} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

- Note, here we have omitted the dependence on the parameters for simplicity

informatics

Bayes' Rule

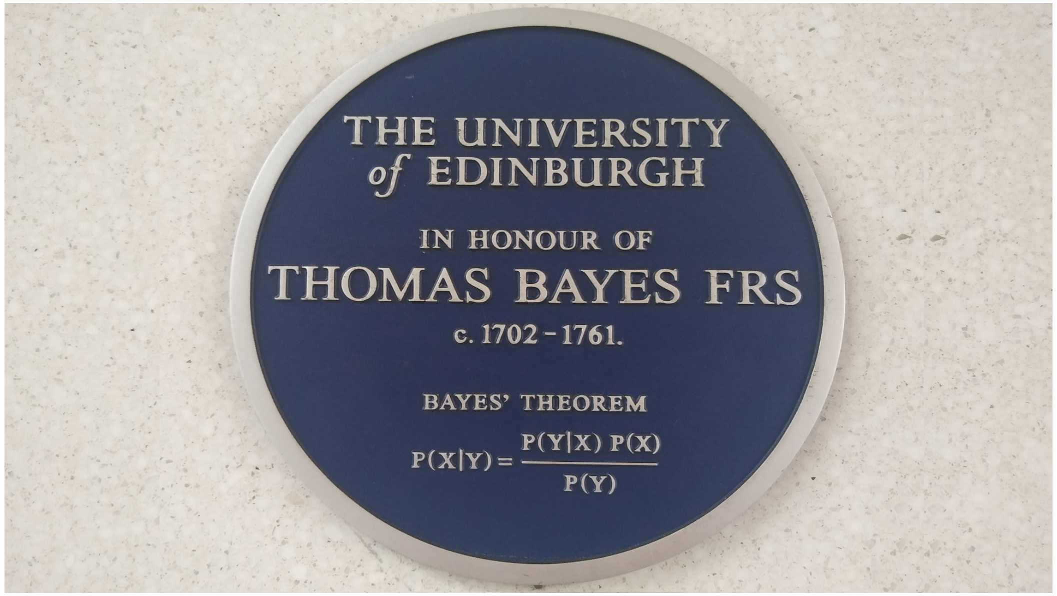
- Bayes' rule can be derived though application of the *product rule*, i.e.

$p(x, y) = p(x|y)p(y) = p(y|x)p(x)$

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

- $p(y|x)$ is the **posterior** distribution of y, conditioned on x
- $p(x|y)$ is the **likelihood** of x, conditioned on y
- $p(y)$ is the **prior** distribution over y, i.e. what we know about y before seeing any data
- $p(x)$ is the **evidence**, which can be computed by marginalising over the unknown y, i.e. $\sum_y p(x|y)p(y)$

informatics



Maximum Likelihood Estimation

Maximum Likelihood Estimation

- In binary classification we have a set of $N_{\mathcal{D}}$ pairs of observations, where $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N_{\mathcal{D}}}$
- The process of learning the model parameters $\boldsymbol{\theta}$ from our dataset $\mathcal{D}$ is called **model fitting** or **training**

- One common approach for fitting a model to data, is called **Maximum Likelihood Estimation** (MLE)
- Here we aim to find the parameters that assign the highest *likelihood* to our data given our model, i.e. the ones that maximise the likelihood

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$$

Independence Assumption

- For convenience, we typically assume that the training data are *independent and identically* sampled from the same distribution, i.e. the **iid assumption**

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N_{\mathcal{D}}} p(x_n, y_n; \boldsymbol{\theta})$$

Log Likelihood

- Taking the product of many terms can introduce numerical issues. To overcome this, we take the log which will not impact where the maximum of the function is

$$\begin{aligned}\mathrm{LL}(\boldsymbol{\theta}) &= \log p(\mathcal{D}|\boldsymbol{\theta}) \\ &= \log \prod_{n=1}^{N_{\mathcal{D}}} p(x_n, y_n; \boldsymbol{\theta}) \\ &= \sum_{n=1}^{N_{\mathcal{D}}} \log p(x_n, y_n; \boldsymbol{\theta})\end{aligned}$$

- Recall that the log of a product equals the sum of the logs, i.e. $\log(ab) = \log(a) + \log(b)$

Negative Log Likelihood

- Many optimisation algorithms are designed to **minimise** functions. We can instead write the log likelihood (LL) as the **Negative Log Likelihood** (NLL)

$$\mathrm{NLL}(\boldsymbol{\theta}) = -\sum_{n=1}^{N_{\mathcal{D}}} \log p(x_n, y_n; \boldsymbol{\theta})$$

- Maximising the LL is equivalent to minimising the NLL

$$\hat{\boldsymbol{\theta}}_{\mathrm{MLE}} = \arg\min_{\boldsymbol{\theta}} \mathrm{NLL}(\boldsymbol{\theta})$$

Negative Log Likelihood

- We can rewrite our expression for the NLL as

$$\begin{aligned}\mathrm{NLL}(\boldsymbol{\theta}) &= -\sum_{n=1}^{N_{\mathcal{D}}} \log p(x_n, y_n; \boldsymbol{\theta}) \\ &= -\sum_{n=1}^{N_{\mathcal{D}}} \log\left[p(y_n; \boldsymbol{\theta}_b) p(x_n|y_n; \boldsymbol{\theta}_g)\right] \\ &= \underbrace{-\left[\sum_{n=1}^{N_{\mathcal{D}}} \log p(y_n; \boldsymbol{\theta}_b)\right]}_{\text{Bernoulli NLL of labels}} \underbrace{-\left[\sum_{n=1}^{N_{\mathcal{D}}} \log p(x_n|y_n; \boldsymbol{\theta}_g)\right]}_{\text{Guassian NLL of features}}\end{aligned}$$

- These two terms depend on different sets of parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_b, \boldsymbol{\theta}_g\}$, so they can be optimised independently

Bernoulli Distribution

- In the case of the binary label data $y \in \{0, 1\}$, we can use a Bernoulli prior
- The probability mass function with the parameter ϕ of the Bernoulli is defined as

$$\mathrm{Ber}(y|\phi) = \begin{cases} 1-\phi & \text{if } y = 0 \\ \phi & \text{if } y = 1 \end{cases}$$

- We can rewrite this as

$$\mathrm{Ber}(y|\phi) = \phi^y (1-\phi)^{(1-y)}$$

MLE for the Bernoulli Distribution

- We can compute the NLL for the Bernoulli with $\boldsymbol{\theta}_b = \{\phi\}$ as follows

$$\begin{aligned}\text{NLL}(\phi) &= -\sum_{n=1}^{N_{\mathcal{D}}} \log p(y_n; \boldsymbol{\theta}_b) \\ &= -\sum_{n=1}^{N_{\mathcal{D}}} \log \left[\phi^{y_n}(1-\phi)^{(1-y_n)}\right] \\ &= -N_1 \log(\phi) - N_0 \log(1-\phi)\end{aligned}$$

- The MLE can be found by solving $\frac{\partial}{\partial \phi}\text{NLL}(\phi) = 0$
- Which results in

$$\hat{\phi} = \frac{N_1}{N_0 + N_1}$$

Gaussian Likelihood

- For the Gaussian NLL we need to solve for the parameters $\boldsymbol{\theta}_g = \{\mu_0, \sigma_0^2, \mu_1, \sigma_1^2\}$, i.e. the parameters for both Gaussians (one for each class)

$$\begin{aligned}\text{NLL}(\mu_0, \sigma_0^2, \mu_1, \sigma_1^2) &= -\sum_{n=1}^{N_{\mathcal{D}}} \log p(x_n|y_n; \boldsymbol{\theta}_g) \\ &= -\sum_{n=1}^{N_{\mathcal{D}}} \log\left[\mathcal{N}(x_n|\mu_0, \sigma_0^2)^{(1-y_n)}\mathcal{N}(x_n|\mu_1, \sigma_1^2)^{(y_n)}\right] \\ &= -\sum_{n=1}^{N_{\mathcal{D}}}(1-y_n)\log\left[\mathcal{N}(x_n|\mu_0, \sigma_0^2)\right] - \sum_{n=1}^{N_{\mathcal{D}}} y_n \log\left[\mathcal{N}(x_n|\mu_1, \sigma_1^2)\right]\end{aligned}$$

Splitting the Data

- For convenience we will split the data into two subsets $\mathcal{D}_0$ and $\mathcal{D}_1$, where $N_0 = |\mathcal{D}_0|$ and $N_1 = |\mathcal{D}_1|$
- Here, $\mathcal{D}_0 \subset \mathcal{D}$ is the subset of data where $y_n = 0$, and $\mathcal{D}_1$ is the subset where $y_n = 1$
- We can then find the maximum likelihood estimate for each set separately

- Our expression for the Guassian NLL now becomes

$$\text{NLL}(\boldsymbol{\theta}_g) = -\sum_{x_n \in \mathcal{D}_0} \log \mathcal{N}(x_n|\mu_0, \sigma_0^2) \quad - \sum_{x_n \in \mathcal{D}_1} \log \mathcal{N}(x_n|\mu_1, \sigma_1^2)$$

MLE for Univariate Gaussians

- Here, we will just focus on one of the Gaussians, i.e. the case where $y_n = 0$

$$\begin{aligned}\text{NLL}(\mu_0, \sigma_0^2) &= -\sum_{x_n \in \mathcal{D}_0} \log \mathcal{N}(x_n|\mu_0, \sigma_0^2) \\ &= -\sum_{x_n \in \mathcal{D}_0} \log\left[\frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{1}{2\sigma_0^2}(x_n - \mu_0)^2\right)\right] \\ &= \frac{N_0}{2}\log(2\pi) + \frac{N_0}{2}\log(\sigma_0^2) + \sum_{x_n \in \mathcal{D}_0} \frac{(x_n - \mu_0)^2}{2\sigma_0^2}\end{aligned}$$

- The minimum of the NLL must satisfy the following conditions

$$\frac{\partial}{\partial \mu_0}\text{NLL}(\mu_0, \sigma_0^2) = 0, \qquad \frac{\partial}{\partial \sigma_0^2}\text{NLL}(\mu_0, \sigma_0^2) = 0$$

MLE Solution for Univariate Gaussians

- Solving for the MLE for both classes we get the following expressions for the **means**

$$\hat{\mu_0} = \frac{1}{N_0} \sum_{x_n \in \mathcal{D}_0} x_n, \qquad \hat{\mu_1} = \frac{1}{N_1} \sum_{x_n \in \mathcal{D}_1} x_n$$

- With the following for the **variances**

$$\hat{\sigma_0}^2 = \frac{1}{N_0} \sum_{x_n \in \mathcal{D}_0} (x_n - \hat{\mu_0})^2, \qquad \hat{\sigma_1}^2 = \frac{1}{N_1} \sum_{x_n \in \mathcal{D}_1} (x_n - \hat{\mu_1})^2$$

Bringing it all Together

- We have solved for the parameters $\boldsymbol{\theta} = \{\phi, \mu_0, \sigma_0^2, \mu_1, \sigma_1^2\}$ of our model using MLE
- Which we can use in our Bayes classifier

$$p(y = 1|x) = \frac{p(x|y = 1)p(y = 1)}{p(x|y = 0)p(y = 0) + p(x|y = 1)p(y = 1)}$$

- Which in the case of our binary classification model, is equivalent to

$$p(y = 1|x) = \frac{\mathcal{N}(x|\mu_1, \sigma_1^2)\phi}{\mathcal{N}(x|\mu_0, \sigma_0^2)(1 - \phi) + \mathcal{N}(x|\mu_1, \sigma_1^2)\phi}$$

Multivariate Classification

Multivariate Data

- Previously we discussed the case where the input feature was a one dimensional continuous value, i.e. $x \in \mathbb{R}$
- In practice, most datasets will be multivariate, i.e. $\boldsymbol{x} \in \mathbb{R}^D$
- We need to define model for multivariate data

Multivariate Gaussian

- The probability density function (PDF) of the **multivariate Gaussian** is given by

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{(D/2)}|\Sigma|^{1/2}} \exp\left(-0.5(\boldsymbol{x}-\boldsymbol{\mu})^\intercal \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Here, $\boldsymbol{\mu} \in \mathbb{R}^D$ is the **mean vector** and $\Sigma \in \mathbb{R}^{D\times D}$ is the **covariance matrix**
- The *univariate* Gaussian is a special case of this PDF

MLE for Multivariate Gaussian

- The maximum likelihood estimate of the **mean** vector is defined as

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$$

- The maximum likelihood estimate of the **covariance matrix** is defined as

$$\hat{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}})^\intercal$$

Properties of the Covariance Matrix

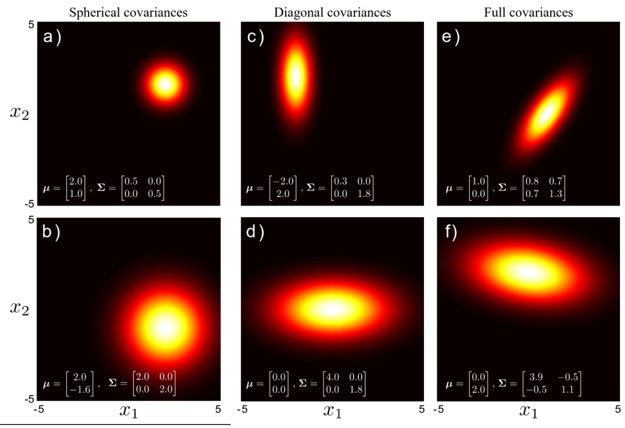
- It is a square matrix ($D \times D$) specifying the covariance between each pair of elements of a given random vector
- Intuitively, it generalises the notion of variance to *multiple dimensions*
- The main diagonal contains variances, i.e. the covariance of each dimension with itself
- The covariance matrix is **symmetric**, i.e. $\Sigma = \Sigma^\intercal$ and $\Sigma^{-1} = (\Sigma^{-1})^\intercal$
- It is positive semi-definite, i.e. $\boldsymbol{x}^\intercal \Sigma \boldsymbol{x} \geq 0$ and $\boldsymbol{x}^\intercal \Sigma^{-1} \boldsymbol{x} \geq 0$
- The full covariance matric has D(D+1)/2 free parameters

Types of Covariance Matrices

- There are three types of covariance matrix
- Here, we show some 2D examples

$$\Sigma_{\text{spher}} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \quad \Sigma_{\text{diag}} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \quad \Sigma_{\text{full}} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 \end{bmatrix}$$

Types of Covariance Matrices



Simon Prince - Computer Vision Models (Book)

Classification With Multivariate Gaussians

- We can use the same generative classification model as before

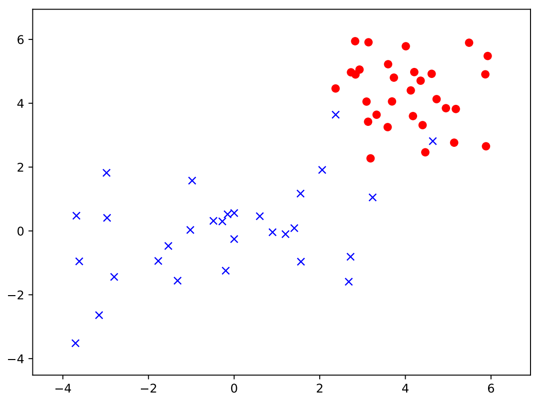
$$p(y = c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = c)p(y = c)}{\sum_{c'} p(\boldsymbol{x}|y = c')p(y = c')}$$

- In the multivariate case, we use a multivariate Gaussian for the class conditional density

$$p(\boldsymbol{x}|y = c) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \Sigma_c)$$

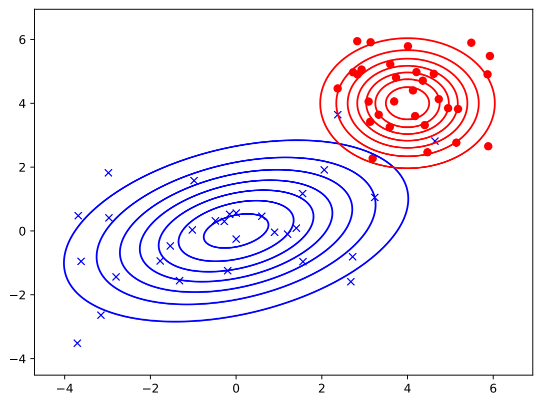
Gaussian Discriminant Analysis - 2D Example

- In this example we have two dimensional data from two different classes, blue and red



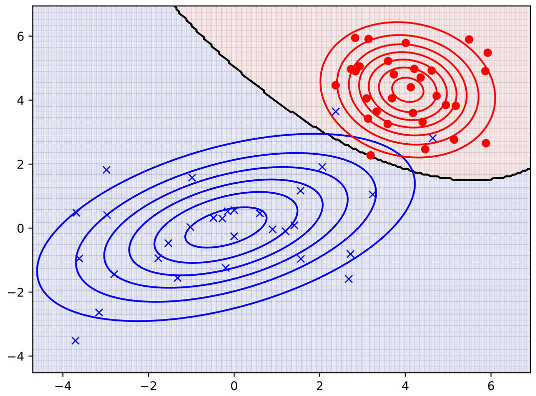
Gaussian Discriminant Analysis - 2D Example

- Here we visualise the underlying Gaussian distributions that generated the observed data



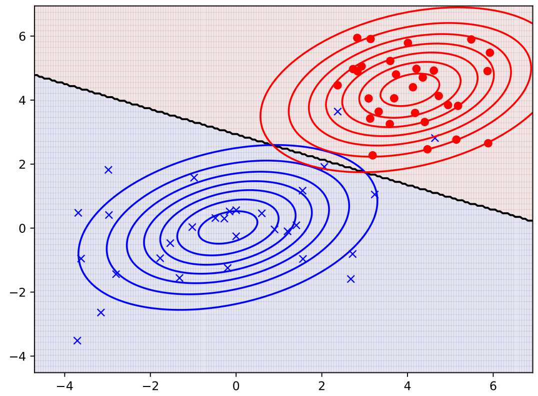
Quadratic Discriminant Analysis - 2D Example

- If we estimate a separate covariance matrix for each class (i.e. Σ_0 and Σ_1) and fit our classifier we get a **quadratic** decision boundary



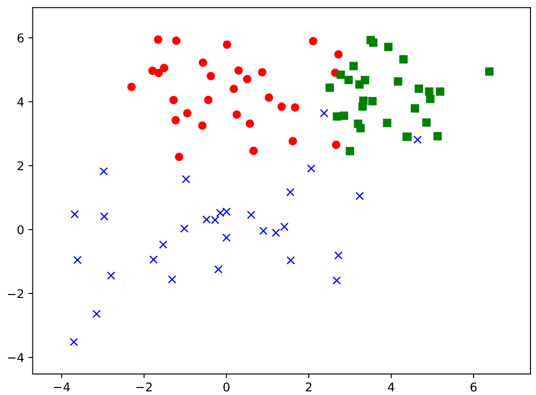
Linear Discriminant Analysis - 2D Example

- If instead, we assume that both classes share the same covariance matrix (i.e. $\Sigma_0 = \Sigma_1$) and fit our classifier we get a **linear** decision boundary



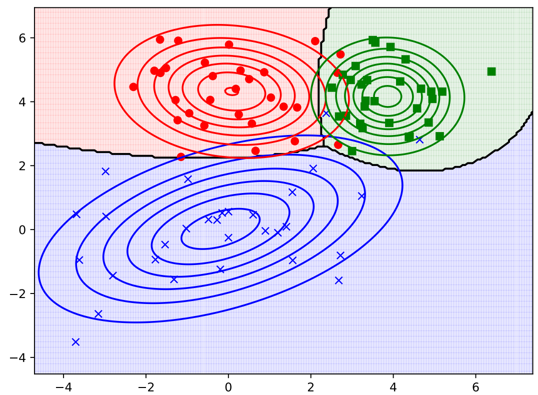
Multiclass Classification

- We can apply the same model in the multiclass case, i.e. where $y \in \{1, ..., C\}$ and $C > 2$, by simply defining a class conditional model $p(\boldsymbol{x}|y = c)$ for each class



Multiclass Classification

- We can apply the same model in the multiclass case, i.e. where $y \in \{1, ..., C\}$ and $C > 2$, by simply defining a class conditional model $p(\boldsymbol{x}|y = c)$ for each class



Summary

- We introduced the problem of supervised classification
- We showed that simple Guassian based models can be used for classification with continuous data through the application of Bayes' rule
- The parameters of these models are estimated using maximum likelihood estimation
- These models can be used for both single or vector input data and for binary or multiclass outputs