Week 2 - ML

Start Date	@10 November 2025

⭐ Chapter 3

Choosing Classification Algorithm

- Takes practice and time

> "No free lunch theorem" - David H. Wolpert

- There isn't one classifier that works best for all scenarios
- Choose a handful of models, then select one that works → model training very dependant on data over training model
- **5 steps fro supervised training:**
    1. Select features + collect labelled training data
    2. Choose performance metric
    3. Choose learning algorithm + training model
    4. Evaluate performance of model
    5. Fine-Tune model
- Feature selection, pre-processing, etc.. is important too but covered later in the book

Scikit-Learn - Perceptron

- User-friendly + optimised implementation of several classification algorithms
- `train_test_split` → split data into test + training data
- Many machine learning + optimisation algorithms → require feature scaling for optimal performance (e.g. Gradient Descent)

Logistic Regression + Conditional Probabilities

> ⭐ Logistic Regression → Classification model → performs well on linearly separable classes

- **One of the most widely used algorithms for classification** → easy to implement and scale
- Linear model for binary classification
- Logistic Regression for multiple classes → **Multinomial Logistic Regression or Softmax regression**

Main mechanics

Odds

Odds in favour of particular event.

$$\frac{p}{(1-p)}$$

p = probability of positive event

- positive event = event we want to predict. Doesn't necessarily mean that it's positive

Logit Function

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

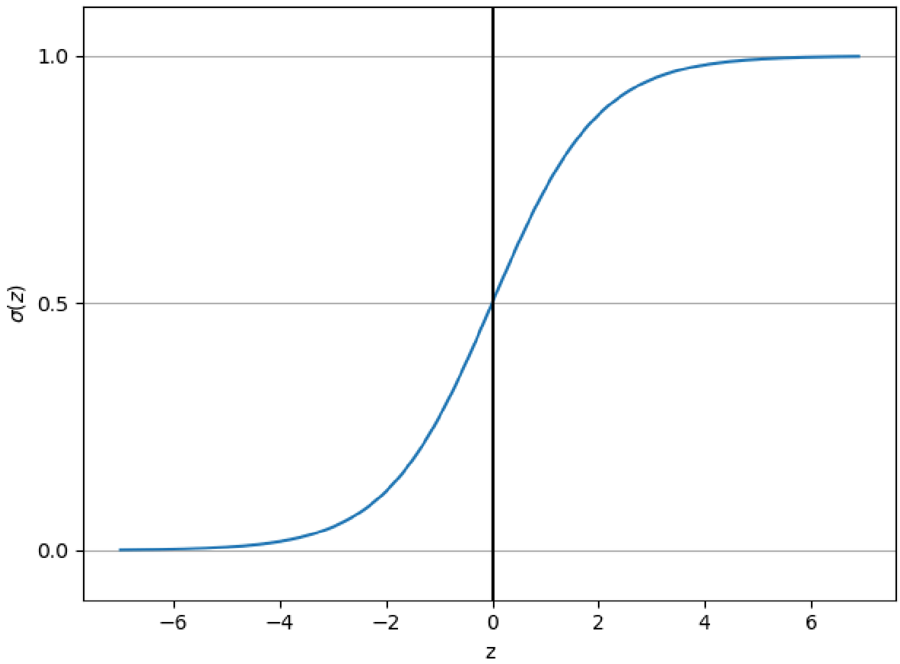
- log → Natural log
- Takes input value in the range of 0 to 1

$$logit(p) = w_1x_1 + ...w_mx_m + b = \sum_{i=j} w_jx_j + b = w^Tx + b$$

- Logit function maps probability to real-number range → consider inverse of function to map back to [0,1] range

Inverse of Logit

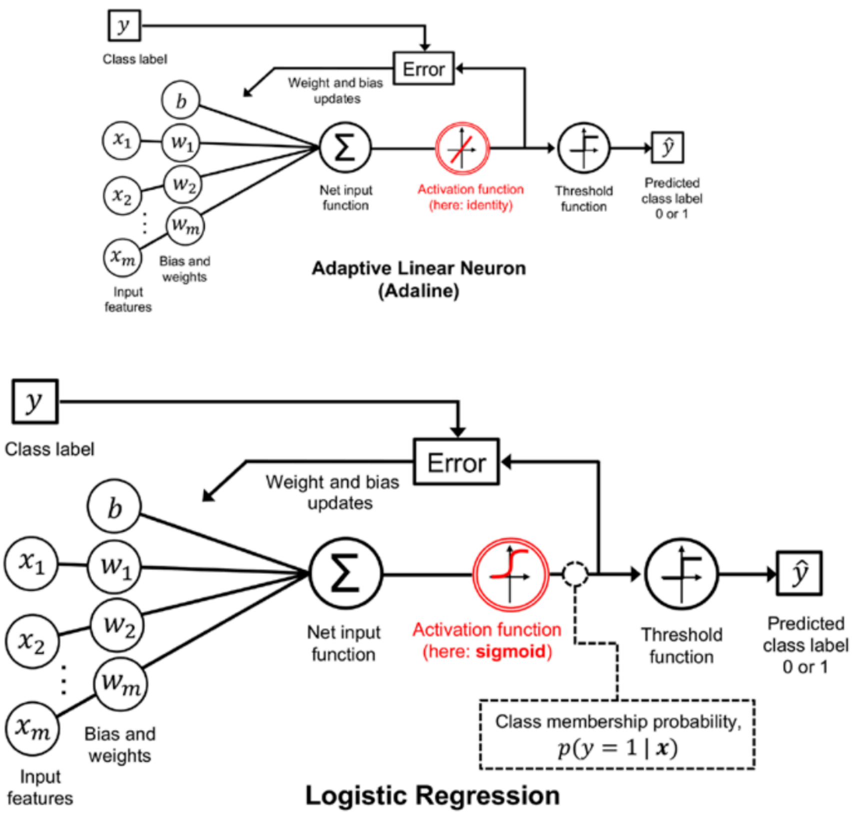
- Logistic Sigmoid Function → abb. Sigmoid Function → S shape



Logistic Sigmoid Function

$$\sigma(z) = \frac{1}{1 + e^{-2}}$$

- z = net input → $z = w^T x + b$



Logistic Regression compared to Adaline

- $\sigma(z) \rightarrow 1$ if $z \rightarrow \infty$ because e^{-2} becomes very small for large values
- $\sigma(z) \rightarrow 0$ if $z \rightarrow \infty$ as a result of an increasingly large denominator
- Conclusion is that Sigmoid takes real-number values as input and transforms them into values in range 0 to 1. Intercept $\sigma(0) = 0.5$

> 💡 Logistic Regression used in weather forecasting and also popular in medicine

Model Weights Via Logistic Loss Function

- Minimise MSE to learn parameters for Adaline classification model

$$L(w, b|x = p(y|x, w, b) = \prod_{i=1}^{n} p(y^{(i)}; w, b) = \prod_{i=1}^{n} (\sigma(z^{(i)}))^{y^{(i)}} (1 - \sigma(z^{(i)}))^{1-y^{(i)}}$$

In practice → we would use log-likelihood

$$L(w, b|x) = LogL(y|x; w, b) = \sum_{i=1} [y^{(i)} log(\sigma(z^{(i)}) + (1 + y^{(i)}) log(1 - \sigma(z^{(i)}))]$$

▼ Logistic Loss Function

$$L(\theta) = \prod_{i=1}^{n} P(y^{(i)}|x^{(i)}; \theta)$$

Log-Likelihood

$$logL(\theta) = \sum_{i=1}^{n} logP(y^{(i)}|x^{(i)}; \theta)$$

1. Applying Log Function reduces potential numeric underflow
   - Working with probability in range 0-1
   - Sometimes probabilities can get really close to 0 or 1.
   - $log(p)$ **makes very tiny probabilities manageable + numerically stable**
     - Improves reliability for model training process
2. Convert product of factors into summation of factors → **Easier to compute derivatives**

- e.g. $log(a \times b \times c) = log(a) + log(b) + log(c)$
- Taking log of products allows you to write the loss as a sum, making gradient calculation for weights simpler + stable

⭐ **Foundation of Logistic Regression → Likelihood Function**

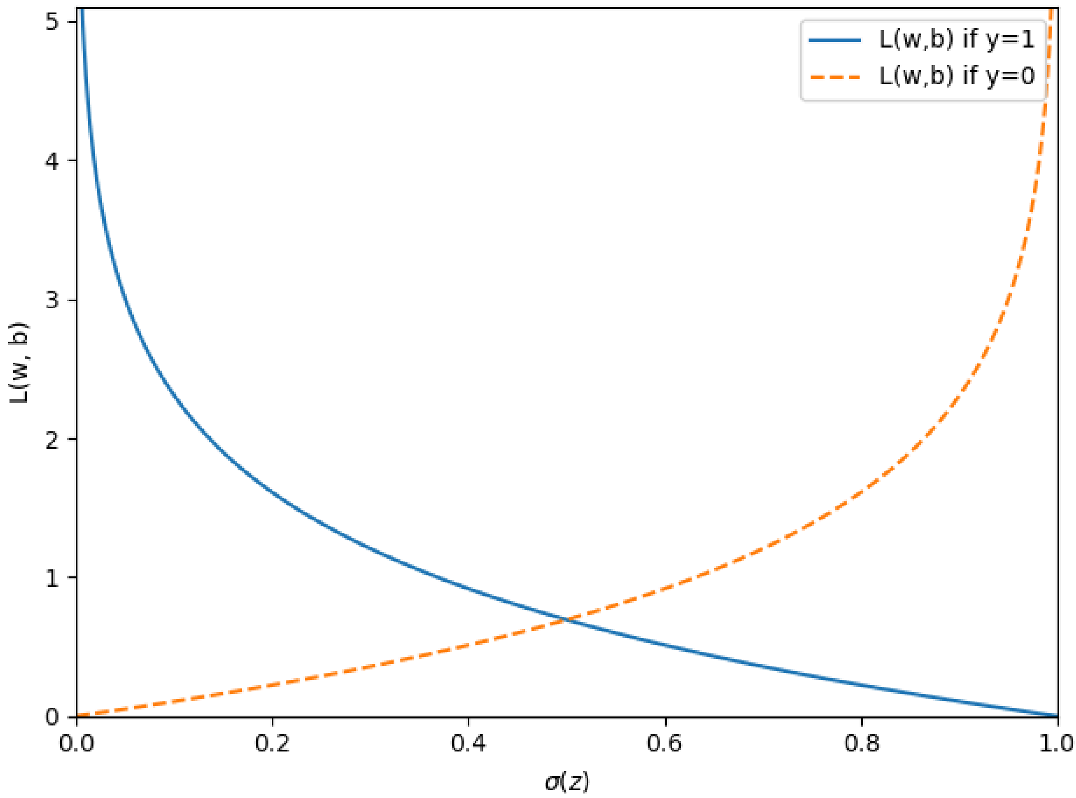
- Goal → Maximise the likelihood → Done with Gradient Ascent
  - **However**, can use gradient descent with negative log-likelihood

💡 Machine Learning frameworks are designed for minimising → it's common to switch to negative and would be the better option too.

$$L(w, b) = \sum_{i=1}[-y^{(i)}log(\sigma(z^{(i)})) + (1 + y^{(i)})log(1 - \sigma(z^{(i)}))]$$

$$L(\sigma(z), y; w, b) = -ylog(\sigma(z)) - (1 - y)log(1 - \sigma(z))$$

$$L(\sigma(z), y; w, b) = \begin{cases} -\log(\sigma(z)) & \text{if } y = 1 \\ -\log(1 - \sigma(z)) & \text{if } y = 0 \end{cases}$$



A plot of the loss function used in logistic regression

Main Point → We penalise wrong predictions with increasingly larger loss

Converting Adaline Implementation Into Algorithm For Logistic Regression

- Compute loss of classifying per epoch
- Swap linear activation with Sigmoid
    - Switch the first two and it becomes Logistic Regression
- Fit Logistic Regression model → Only works for binary classification tasks

Gradient Descent Learning Algorithm For Logistic Regression

- Adaline, weights + bias didn't change for Logistic Regression → Gradient Descent similar for Logistic Regression
- Partial derivative of log-likelihood with respect to jth term:

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial a}\frac{da}{dz}\frac{\partial z}{\partial w_j} \quad \text{WHERE } a(z) = \frac{1}{1+e^{-z}}$$

$$\frac{\partial L}{\partial a} = \frac{a-y}{a\cdot(1+a)}$$

$$\frac{da}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} = a\cdot(1+a)$$

$$\frac{\partial L}{dz} = a-y$$

$$\frac{\partial z}{w_j} = x_j$$

$$\frac{\partial L}{\partial w_j} = (a-y)x_j = -(y-a)x_j$$

→ TAKE STEP OPPOSITE OF GRADIENT → FLIP $\frac{\partial L}{w_j} = -(y-a)x_j$ + LEARNING RATE η

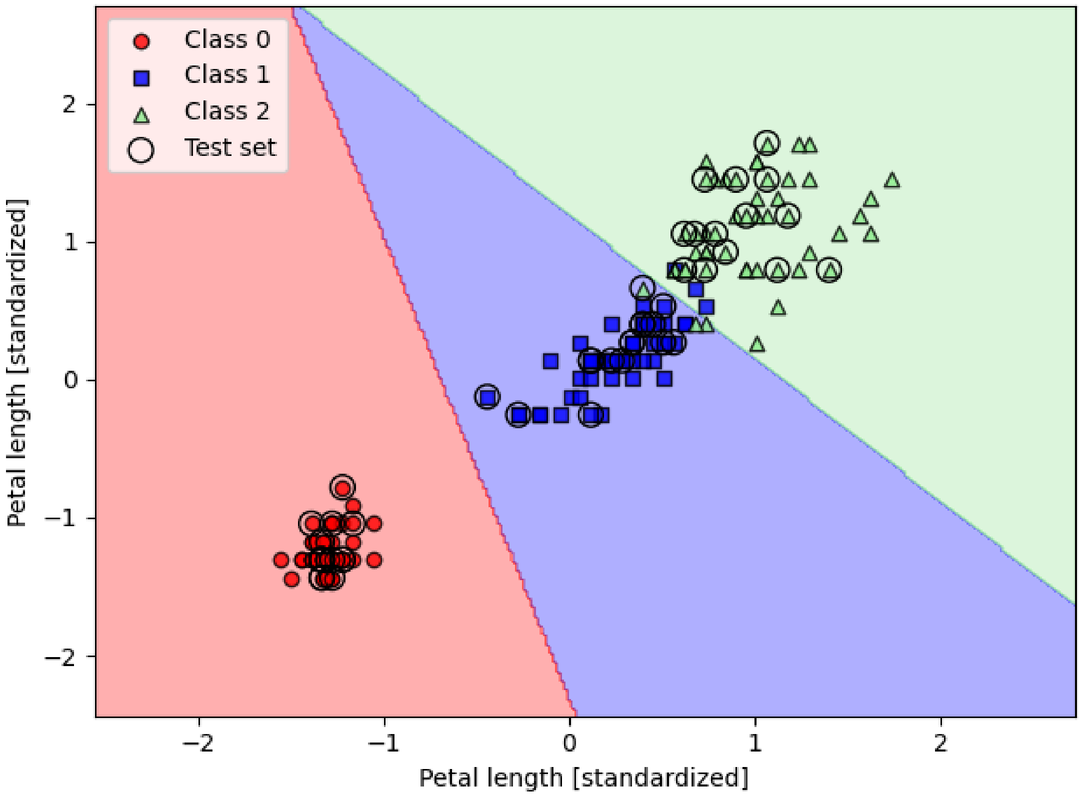
$$w_j := w_j + \eta(y-a)x_j$$

$$b := b + \eta(y-a)$$

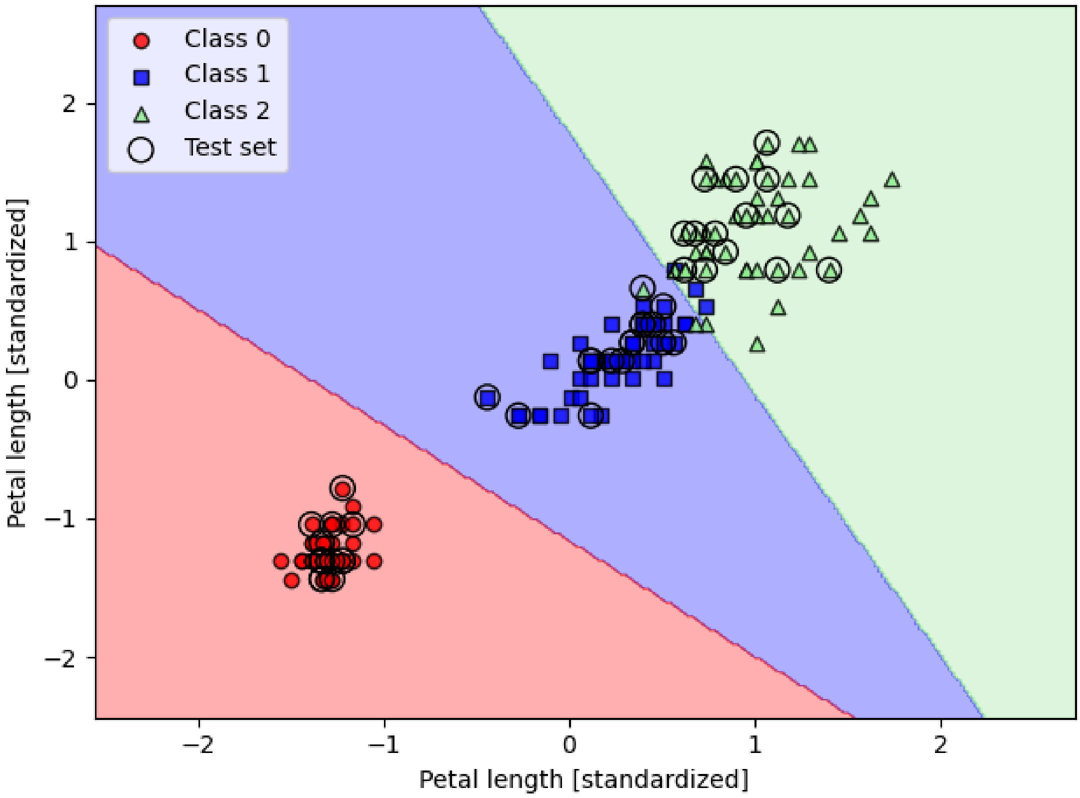
WEIGHT + BIAS = ADALINE WEIGHT + BIAS

Logistic Regression with Scikit-Learn

- Supports multi-class Logistic Regression off the shelf



Multi-class = OvR

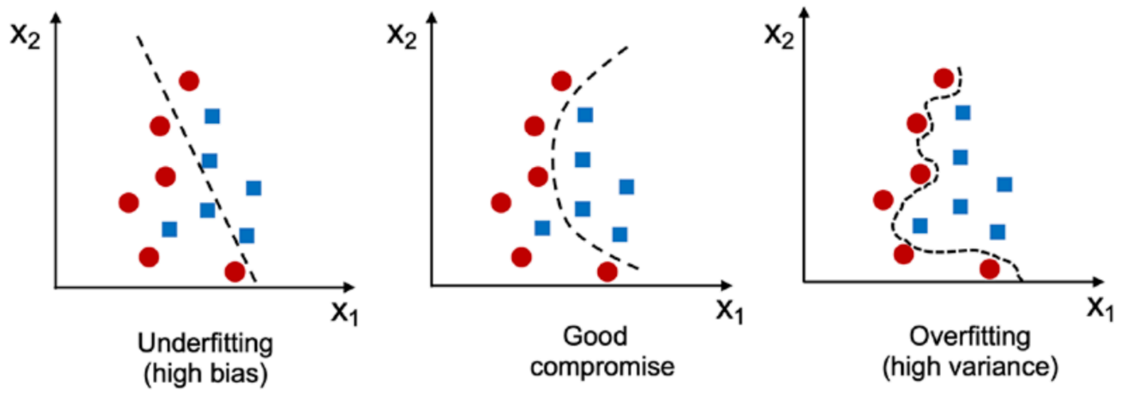


Multi-class = Multinomial (Default for Scikit-learn Logistic Regression)

Convex Optimisation → minimising convex loss functions (logistic regression) → **recommended to use more advanced algorithm over SGD**

Overfitting via Regularisation

- Common problem for machine learning → Doesn't generalise well with unseen data
- Overfitting → High variance model → caused by too many parameteres (too complex)
- Underfitting → High bias → model not complex enough



Examples of underfitted, well-fitted, and overfitted models

Bias-Variance Trade-Off

What is Bias?

- Measures how far off the predictions are from the correct values
- High-bias model makes strong assumptions about data but underfits - can't figure out the underlying pattern well

What is Variance

- Measures consistency of model prediction
- High-variance → model is sensitive to specific training data (fits noise + small data fluctuation)

The Trade-Off

- Gotta find the sweet spot between the two
- One way to find good bias-variance trade-off → tune complexity of model via regularisation

> ⭐ Regularization → Very useful for handling collinearity, diltering out noise from data + eventually presenting overfitting

- Add additional information to penalise extreme parameter values → most common → **L2-Regularization**
- Regularisation → feature scaling and standardisation → are all important
    - For this to work, features must be on comparable scales

$$\frac{\lambda}{2n} = \frac{\lambda}{2n}\sum_{m}^{j=1} w_j^2$$

- λ = Regularisation parameter
- 2 = Scaling factor → cancels computing loss function
- n = Sample size added to scale regularisation term similar to loss

Loss function for logistic regression + regularisation:

$$L(w,b) = \sum_{i=1}[-y^{(i)}log(\sigma(z^{(i)})) + (1+y^{(i)})log(1-\sigma(z^{(i)}))] + \frac{\lambda}{2n}||w||^2$$

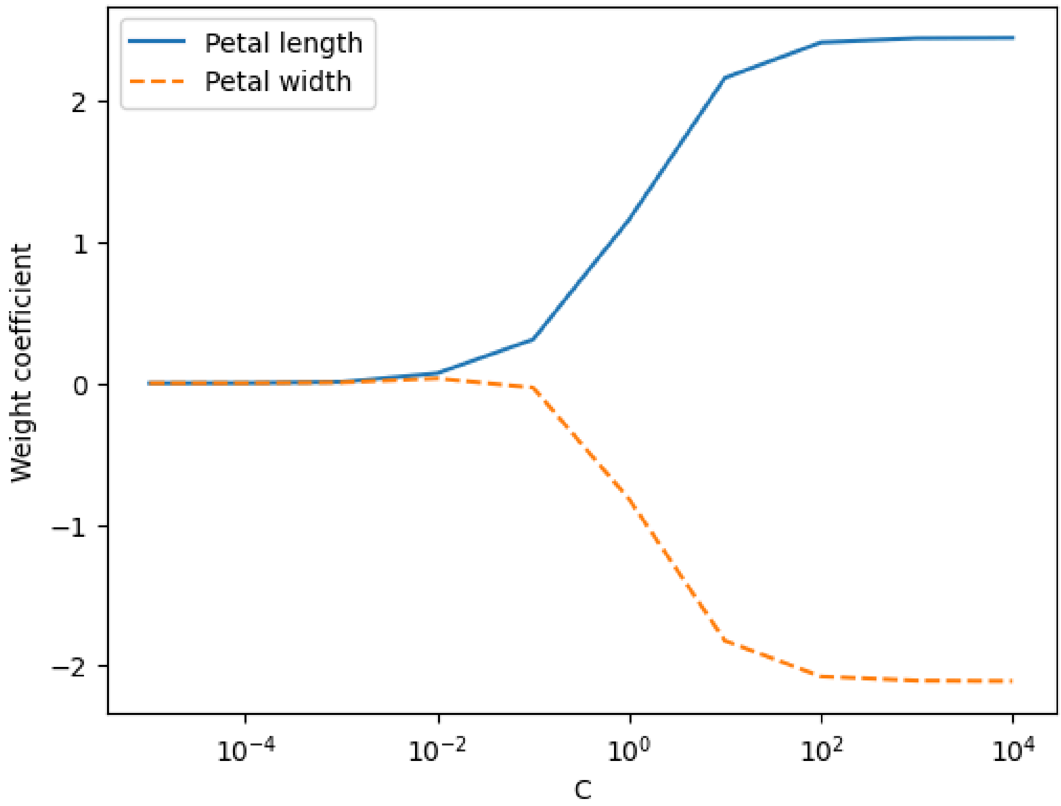
Partial derivative of unregularised loss:

$$\frac{\partial L(w,b)}{\partial w_j} = \left(\frac{1}{n}\sum_{i=1}^{n}(\sigma(w^T x^{(i)}) - y^{(i)})x_j^2)\right)$$

Adding regularisation to derivative of loss:

$$\frac{\partial L(w,b)}{\partial w_j} = \left(\frac{1}{n}\sum_{i=1}^{n}(\sigma(w^T x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{n}w_j$$

📌 We can control how closely we fit the data with λ → If we increase λ, iuncrease regularisation strength.

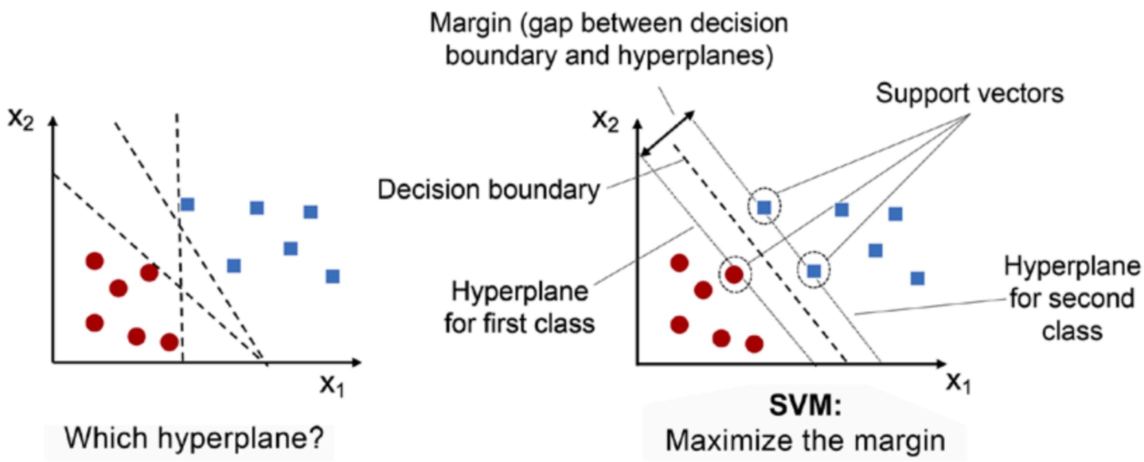


The impact of the inverse regularization strength parameter C on L2 regularized model results

> ⚠️ **Why do we not strongly regularie all models if it reduces overfitting?**
> If regularisation is too high → weights coefficient approaches 0 and therefore our model will underfit. (look at fig above)

Maximum Margin Classification With Support Vector Machines (SVM)

- Powerful + widely used algorithm → **Support Vector Machine (SVM)** → **considered extension of perceptron**
- SVM → maximise margin → Distance between the **separating hyperplane (decision boundary)** and **training examples closest to hyperplane (support vectors)**



SVM maximizes the margin between the decision boundary and training data points

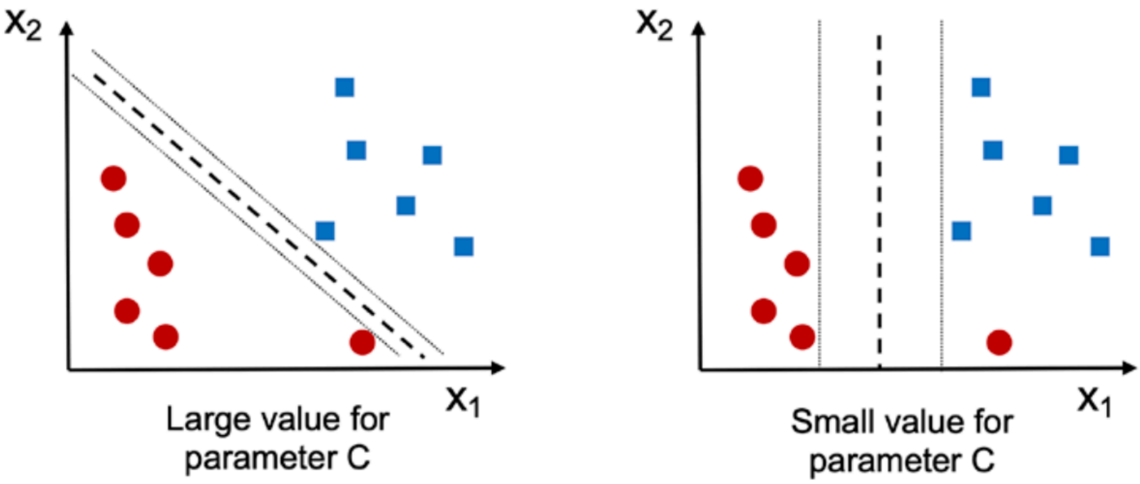
Maximum Margin Intuition

- Larger margins → tend to have a lower generalization error

> ⚠️ SVM looks simple → math is complex → need to know **constrained optimisation**

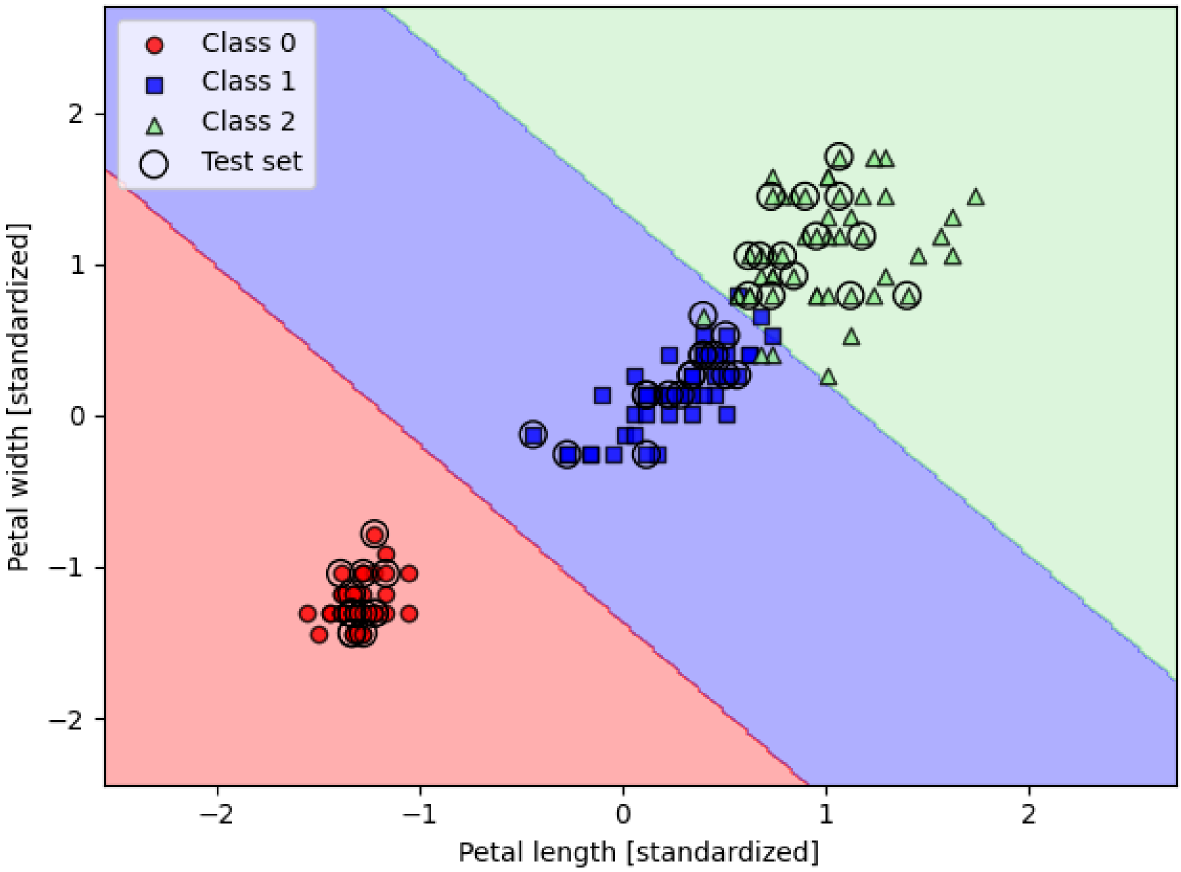
Slack Variable

- Introduced by Vladimir Vapnik in 1995 → this led to soft-margin classification
- Strict constraints → this force the SVM to find a hard boundary. This fails if data isn't cleanly separated
- Relaxing constraints → this would be adding slack variables → some points can be misclassified, but will get penalised for each mistake
- SVM optimisation seeks balance → wide margin (good separation) + few misclassification (low slack penalty)
- **To use slack variables** → introduce variable C → hyperparameter for controlling penalty for misclassification



The impact of large and small values of the inverse regularization strength C on classification

- Large C → the larger the error penalties
- Smaller C → less strict about misclassification error
- C controls width of margin, therefore toning bias-variance tradeoff



SVM's decision regions

Logistic Regression vs. SVM

- Logistic Regression and SVM in practical classification yield similar results
- Logistic Regression → Maximises the conditional likelihood
- SVM → Cares about points closest to decision boundary (support vectors)

> ⭐ Logistic Regression is the simpler model, it’s easier to implement and is easily updatable with newer data.