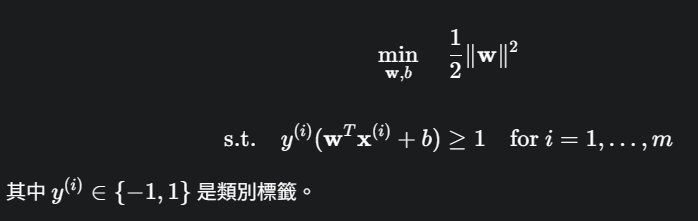
**1. SVM (Support Vector Machine) 的訓練過程**

SVM 的核心目標是找到一個**最大邊界超平面**，將不同類別的數據點分開。這是一個**二次規劃 (Quadratic Programming)** 問題。

**找出最佳的W 和 b 的數學推導**

**A. 原始優化問題 (Primal Problem)**

對於線性可分 (Hard Margin) 的數據，目標是最大化邊界 ，等價於最小化。



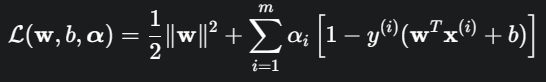
****

**B. 轉換為對偶問題 (Dual Problem)**

為了解決帶有不等式約束的優化問題，我們使用**拉格朗日乘數法**引入乘子

。

拉格朗日函數：

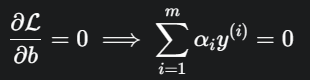


求解 w 的條件：

對 w 求偏導並令其為零，得出 w 的表達式：

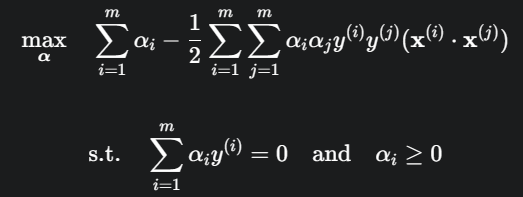
求解 b 的條件：

對 b 求偏導並令其為零，得出 α 的約束：

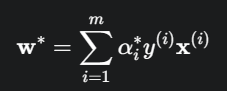


對偶目標函數 (Dual Objective Function)：

將 w∗ 和 b 的條件代回 L，形成一個只關於 α 的優化問題：



**C. 結論**

1. 通過求解 α 的二次規劃問題（通常使用 **SMO 算法**），得到最佳乘子 α\*。
2. 最佳權重 w∗：由支持向量（即 αi∗​>0 的樣本）計算得到：
3. **最佳偏置 b\***：由任一支持向量滿足邊界條件 解出。

**2. MLP (Multi-Layer Perceptron) 的訓練過程**

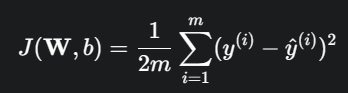
MLP 的訓練是通過**反向傳播算法 (Backpropagation)** 結合**梯度下降法 (Gradient Descent)** 來迭代調整權重 $\mathbf{W}$ 和偏置 $b$。

**找出最佳的 W的數學推導**

**A. 最小化目標函數**

MLP 訓練的目標是最小化**損失函數 (Loss Function)** J(W,b)。

損失函數範例 (均方誤差 MSE)：



其中 y^​ 是網絡的預測輸出。

**目標：** 找到 W\* 和 b\* 使得 J(W,b) 最小。

前向傳遞 (Forward Propagation) 流程

**B. 訓練核心流程：前向傳遞與反向傳播**

**B.1. 前向傳遞 (Forward Propagation) 流程**

前向傳遞計算網絡的輸出 y^。假設第 $l$ 層的激活函數為 ，前一層的激活輸出為 ，輸入層。

計算加權和 (Net Input)



計算激活輸出 (Activation)

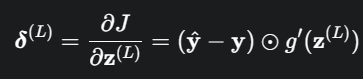


**預測值：**

**B. 梯度計算：反向傳播 (Backpropagation)**

使用**鏈式法則**計算損失函數 J 對於每一層權重 的梯度 。

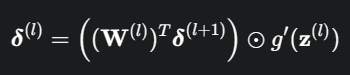
1. 輸出層誤差 δ(L)：



（以 MSE 和 Sigmoid 輸出為例， z(L) 是輸出層的加權和， g′ 是激活函數的導數）。

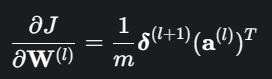
1. 隱藏層誤差 δ(l) 的遞歸：

誤差從後一層向前傳播：



權重梯度 ∂W(l)∂J​：

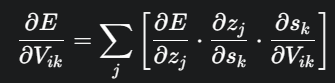
利用誤差項和前一層的激活值 a(l) 計算梯度：



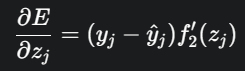
隱藏層到輸入層的權重梯度

假設符號：E為損失，為輸出層加權和， 為隱藏層加權和，為輸入到隱藏層 $k$ 的權重， 為輸入。

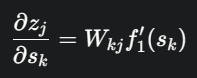
對於單一樣本，梯度計算為：



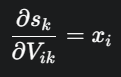
**輸出層誤差 ：** (即 )



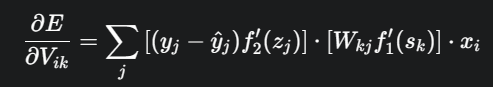
鏈接項



鏈接項

****

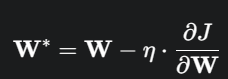
綜合梯度 (對於單一樣本)



**C. 權重更新 (梯度下降)**

計算出梯度後，利用梯度下降法迭代更新權重，以逼近最佳解 W\*：

權重更新規則：



**程式中如何實現:**

無論是 MLP 的梯度下降，還是 SVM 的 SGD 變體，程式中權重的迭代更新都遵循以下模式：

| **概念** | **數學表達式** | **程式實現 (偽代碼)** |
| --- | --- | --- |
| **當前權重** | W | w |
| **更新量** |  | delta\_w = -learning\_rate \* gradient\_J\_w |
| **學習率** |  | learning\_rate |
| **梯度** |  | gradient\_J\_w |
| **更新步驟** |  | w = w + delta\_w |

**程式碼範例 (Python/NumPy 概念):**

Python

# 1. 計算梯度 (通過反向傳播)

gradient\_J\_w = calculate\_gradient(data, w)

# 2. 設定學習率

learning\_rate = 0.01

# 3. 計算更新量 (Delta W)

delta\_w = -learning\_rate \* gradient\_J\_w

# 4. 更新權重 (W\* = W + Delta W)

w = w + delta\_w

# 此時的 w 即為下一輪的 w\*