C111112130 林廉閎 機器學習與實作HW1

SVM 數學推導

以**線性 SVM（hard-margin）**為例，目標是最大化分類間隔（margin）→ 等價於最小化權重的二次範數：

Primal problem（hard-margin）：

$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$\text{s.t.} \quad y_i(w^\top x_i + b) \geq 1, \quad i = 1, \ldots, N$$

引入拉格朗日乘子 $\alpha_i \geq 0$，建 Lagrangian：

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \big(y_i(w^\top x_i + b) - 1\big)$$

對 w 與 b 求偏導並令其為 0 (KKT 條件)：

- $\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i.$
- $\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0.$

把 w 代回 Lagrangian 得到 **Dual** 問題（凸二次規劃）：

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j$$
$$\text{s.t.} \quad \alpha_i \geq 0, \quad \sum_{i=1}^{N} \alpha_i y_i = 0.$$

對於 **soft-margin**（允許錯分），引入鬆弛變數 ξ_i 與常數 C：
Primal:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C \sum_i \xi_i, \quad \text{s.t. } y_i(w^\top x_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0$$

Dual 的條件會變成 $0 \leq \alpha_i \leq C$。

得到最終解：

- 計算出最佳 α^* 後，線性權重為

$$w^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i$$

- 偏差 b 可由任一支援向量（support vector）求得（取 $0 < \alpha_i < C$ 的那些）：

$$b = y_i - \sum_j \alpha_j y_j x_j^\top x_i.$$

- 若使用 kernel，一律把 $x_i^\top x_j$ 換成 $K(x_i, x_j)$。

重點：**SVM 有明確的凸優化形式（QP）**，可用 QP 求解器或捷徑（hinge loss + SGD）來近似求解。

SVM訓練過程

SVM目標在於找到一條線(2維)或一個平面(3維)來將不同類型的資料做區隔，以線來說，雖然有很多條都能達到這個目的，但SVM想找那條**最中間、最穩定**的線。而最中間就是看這條線到「最近的點」的距離，這個距離叫 **margin**。

1.margin 大 → 分類更穩定
2.margin 小 → 分類容易錯誤，一點點就誤判

然後最靠近的點叫**支援向量**，用來決定所要求的線。而最佳 w其實就是這些支援向量加起來，算出來的方向向量。也就是說找最佳w的過程，其實就是一個挑出支援向量 → 算出中間線的過程。

我們要「最大化間隔 (margin)」，等價於「最小化權重的長度」：

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

條件是：

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i$$

這就是 SVM 的核心最佳化問題。

接著，首先我們的目標如下:

找一條線（或超平面）：

$$w \cdot x + b = 0$$

可以 完美區分類別，而且兩邊的距離（margin）最大。

其中訓練過程就如下方所示:

1. **設定目標函數：**
   我們要讓 margin 最大化 → 相當於要讓 $\|w\|$ 最小化。

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

   條件是：

$$y_i(w \cdot x_i + b) \geq 1$$

2. **引入拉格朗日乘子 (Lagrange Multipliers)：**
   把限制條件放進去，建立拉格朗日函數：

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i [y_i(w \cdot x_i + b) - 1]$$

3. **對 w, b 求偏導，讓它等於 0：**

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_i \alpha_i y_i x_i$$

   代表 w 是由各樣本的線性組合組成（但只有支援向量的 $\alpha_i > 0$）。

4. **把它帶回去 → 解對偶問題 (Dual Problem)：**

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

   用數學最佳化演算法（像 SMO）解出最佳 α_i。

5. **算出最終結果：**

$$w^* = \sum_i \alpha_i y_i x_i$$

$$b^* = y_k - w^* \cdot x_k \quad \text{(k 為支援向量之一)}$$

透過這樣的方式我們可以求出所要找的最佳w跟能夠將資料完美分開的那條線。

MLP數學推導

設輸入 $x \in \mathbb{R}^D$、隱藏層維度 H、輸出類別 K。權重與偏差：

- 第一層：$W^{(1)} \in \mathbb{R}^{H \times D}$, $b^{(1)} \in \mathbb{R}^H$
- 第二層：$W^{(2)} \in \mathbb{R}^{K \times H}$, $b^{(2)} \in \mathbb{R}^K$

forward pass：

$$
\begin{aligned}
z^{(1)} &= W^{(1)} x + b^{(1)} \\
a^{(1)} &= \phi(z^{(1)}) \quad \text{(activation，例如 ReLU / sigmoid / tanh)} \\
z^{(2)} &= W^{(2)} a^{(1)} + b^{(2)} \\
\hat{y} &= \mathrm{softmax}(z^{(2)})
\end{aligned}
$$

以cross-entropy為 loss（單一樣本）：

$$
L = -\sum_{k=1}^{K} y_k \log \hat{y}_k
$$

其中 y 為 one-hot 標籤。

反向傳播（關鍵推導）：

- 對輸出層 $z^{(2)}$：對於 softmax + cross-entropy，有漂亮的化簡

$$
\delta^{(2)} \equiv \frac{\partial L}{\partial z^{(2)}} = \hat{y} - y
$$

- 輸出層權重與偏差梯度：

$$
\frac{\partial L}{\partial W^{(2)}} = \delta^{(2)} (a^{(1)})^\top, \quad \frac{\partial L}{\partial b^{(2)}} = \delta^{(2)}
$$

- 傳回到隱藏層：

$$
\delta^{(1)} = (W^{(2)})^\top \delta^{(2)} \odot \phi'(z^{(1)})
$$

$$
\frac{\partial L}{\partial W^{(1)}} = \delta^{(1)} x^\top, \quad \frac{\partial L}{\partial b^{(1)}} = \delta^{(1)}
$$

其中 $\odot$ 是 element-wise 乘法，ϕ' 為 activation 的導數（ReLU 的導數在 $z > 0$ 為 1，否則 0）。

參數更新（梯度下降）：

$$
W \leftarrow W - \eta \frac{\partial L}{\partial W}, \quad b \leftarrow b - \eta \frac{\partial L}{\partial b}
$$

通常採 mini-batch + Adam 或 SGD + momentum。

結論：MLP 的最佳 W 由反覆的梯度更新而來，無閉式解。

在推導過程中提到的反向傳播意思和方法如下:

反向傳播（Backpropagation） 是一種用來計算**梯度（Gradient）的方法，它告訴我們該如何調整每個權重與偏差**，讓模型表現變好。

簡單來說，**反向傳播的目標是讓「損失（Loss）」變小**，它的核心步驟如下：

1. **前向傳播（Forward Pass）：**

2. 1. 輸入資料 x 經過神經網路，產生預測值 y。
   2. 計算預測值和真實值 y之間的誤差（Loss）。

2. **反向傳播（Backward Pass）：**

3. 1. 透過鏈鎖律（Chain Rule），計算損失對每個權重的偏導數（梯度）。
   2. 使用 梯度下降（Gradient Descent） 來更新權重，讓模型預測得更準確。

MLP訓練過程

MLP的一開始會有一個輸入X，而我們希望它的預測輸出值能夠盡可能接近真實輸出，但一開始的權重W、b都是隨機的，需用損失函數計算預測和實際相差多少，再藉由反向傳播算出權重對損失的影響，最後再用梯度下降法來更新權重使預測輸出逐漸接近實際輸出，有點類似於**疊代調整**的概念。

其中MLP的目標就是要找出一組權重，讓神經網路的輸出值盡可能接近實際結果，而下面就是其訓練過程。

訓練過程（找最佳 W）

1. **初始化權重：**
   一開始所有權重 W 都是隨機的。
2. **正向傳遞 (Forward Propagation)：**
   資料從輸入層 → 隱藏層 → 輸出層流動，
   每層都做：

$$z = W \cdot x + b$$

   再經過非線性函數（ReLU、Sigmoid、Tanh）得到輸出。
3. **計算誤差 (Loss)：**
   比較預測結果 y_{pred} 與真實值 y：

$$\text{Loss} = \frac{1}{2}(y - y_{pred})^2$$

4. **反向傳播 (Backpropagation)：**
   根據誤差計算每個權重對誤差的影響（梯度）。
   例如：

$$\frac{\partial \text{Loss}}{\partial W}$$

   表示如果改變這個權重，Loss 會變多少。
5. **梯度下降 (Gradient Descent)：**
   按照「誤差下降的方向」去調整權重：

$$W_{\text{new}} = W_{\text{old}} - \eta \frac{\partial \text{Loss}}{\partial W}$$

   其中 η 是學習率（控制更新的步伐）。
6. **重複步驟 2–5：**
   一直更新權重，直到誤差夠小或達到迭代上限。

透過上面的過程中不斷更新誤差及權重，最後可以得出所要求的最佳w。