1. Define a loss-function for SVM

在機器學習中，鉸鏈損失是一個用於訓練分類器的損失函數。鉸鏈損失被用於「最大間格分類」，因此非常適合用於支持向量機(SVM)。對於一個預期輸出 $t=\pm 1$，分類結果y的鉸鏈損失定義為 $\ell(y)=max(0,1-t\cdot y)$

特別注意：以上式子的y應該使用分類器的「原始輸出」，而非預測標籤。例如，在線性支持向量機當中，$y=w\cdot x+B$，其中 (w,B) 是超平面參數，x 是輸入資料點。

當t和y同號（意即分類器的輸出y是正確的分類），且 $|y|\geq 1$ 時，鉸鏈損失$\ell(y)=0$。但是，當它們異號（意即分類器的輸出y是錯誤的分類）時，$\ell(y)$隨y線性增長。套用相似的想法，如果 $|y|<1$，即使t和y同號（意即分類器的分類正確，但是間隔不足），此時仍然會有損失。

我們將優化模型 $\mathbf{y}=\mathbf{x}*w$ 通過調整參數 w從而使 所有樣本的 均方誤差 (MSE) 最小化。最小化函數也稱為 損失（或成本）函數 。

均方誤差定義為 $\xi=\frac{1}{N}\sum_{i=1}^{N}\|t_i-y_i\|^2$， 和 N訓練集中的樣本數。這對應於 輸出和相應目標之間的平均 歐幾里得距離 。因此優化目標是： $\underset{w}{\operatorname{argmin}}\frac{1}{N}\sum_{i=1}^{N}\|t_i-y_i\|^2$.

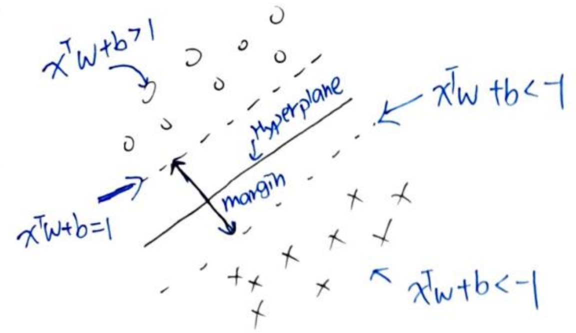
請注意，我們取所有樣本的誤差均值，這稱為批量訓練。我們還可以一次基於一個樣本更新參數，這稱為在線訓練。

這個變量的損失函數 w如下圖所示。價值$w=2$ 是損失函數的最小值（拋物線的底部），這個值與我們選擇的斜率相同 $f(x)$. 請注意，此函數是 凸 函數 ，並且只有一個最小值：全局最小值。雖然線性回歸的每個平方誤差損失函數都是凸的，但其他模型和其他損失函數並非如此。

神經網絡模型在 `nn(x, w)` 函數中實現，損失函數在函數中實現 `loss(y, t)` 。

2. Derivation of optiomal $\vec{w}$ for a binary SVM

2.



margin $= \rho = \frac{2}{\|w\|}$

$\max \rho \Leftrightarrow \max \rho^2 \Leftrightarrow \min \frac{1}{2}\|w\|^2$

$x_i^T w + b \geq +1, \; y_i = +1$

$x_i^T w + b \leq -1, \; y_i = -1$

$\min J(w) = \min \frac{1}{2}\|w\|^2 \quad y_i(x_i^T w + b) \geq 1, \; i = 1, 2, 3, \ldots n.$

$y_i(x_i^T w + b) = 1$

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

$\max L(w, b, \alpha) = +\infty$

$\max L(w, b, \alpha) = J(w) = \frac{1}{2}\|w\|^2$

$\min \max L(w, b, \alpha)$

$\nabla_w L(w, b, \alpha) = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i$

$\nabla_b L(w, b, \alpha) = -\sum_{i=1}^{n} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$

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

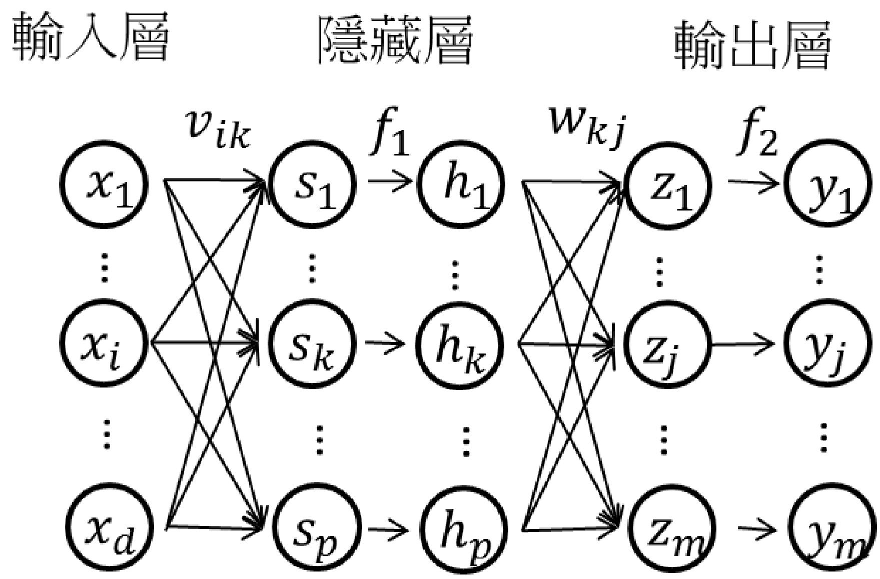
$= \frac{1}{2}\sum_{i=1}^{n} \alpha_i y_i x_i^T \cdot \sum_{j=1}^{n} \alpha_j y_j x_j - \sum_{i=1}^{n} \alpha_i y_i x_i^T \cdot \sum_{j=1}^{n} \alpha_j y_j x_j - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i.$

$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$

$\min L(w, b, \alpha) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \; \alpha_i \geq 0, 1, 2, \ldots, n$

$\min \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n} \alpha_i$

3. Derivation of optimal $\vec{w}'$s for a 3-layers MLP



假設有個 MLP 的結構，共有 n 筆樣本，每個樣本對應 m 個輸出值。隱藏層只有一層設定為 p 個 hidden node。

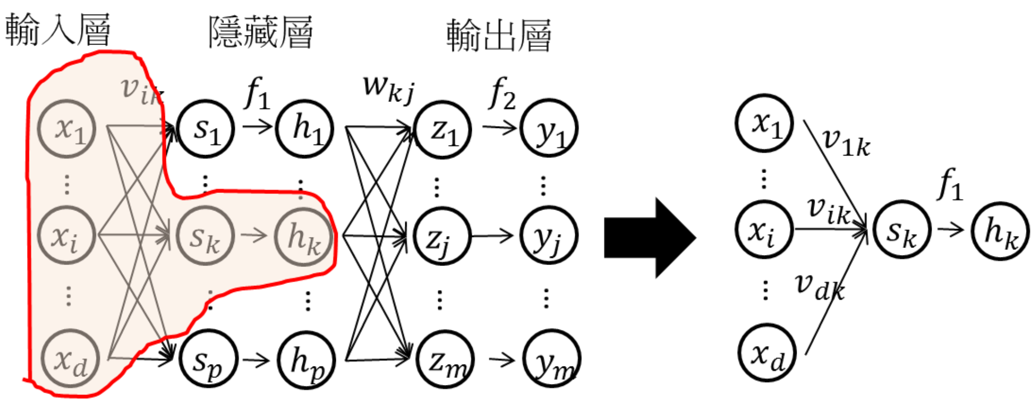
$\{(x^{(i)}, y^{(i)})\}, i = 1, \ldots, n, \quad x_i \in R^d, y_i \in R^m$

前向傳遞(Forward propagation): 較簡單 (只有線性合成，和非線性轉換)

反向傳遞 (Backward propagation): 較複雜 (因為多微分方程)

前向傳遞 (Forward propagation)

輸入層到隱藏層



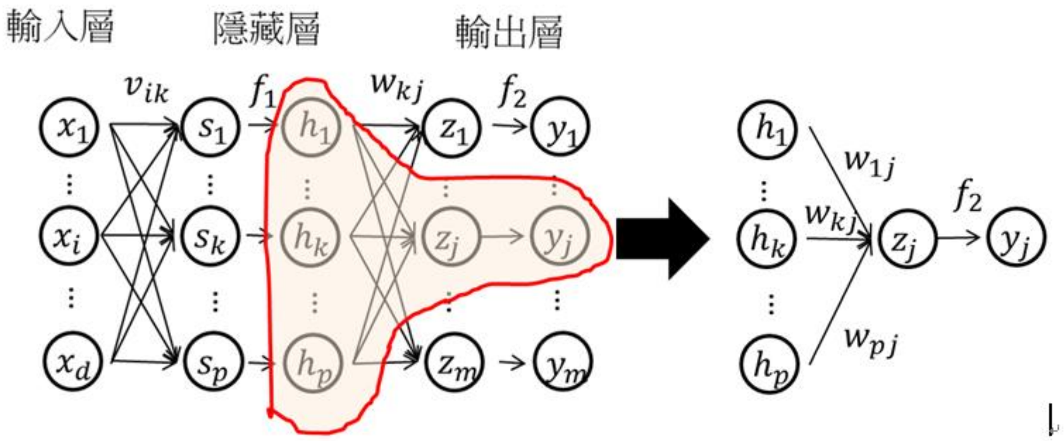
輸入層到隱藏層的值為 sk,k=1,…,p，為輸入訊號的加權線性和(vik 為第 i 個輸入到第 k 個 hidden node 的權重)。

$$s_k = \sum_{i=0}^{d} v_{ik} x_i$$

經過 非線性轉換/激活函數(activation function，f*1*)後，得到 hidden node 的輸出

$$h_k = f_1(s_k)$$

隱藏層到輸出層



隱藏層到輸出層的值為 *zj,j=1,...,m*，為 hidden node 輸出的加權線性和(w*kj* 為第 k 個 hidden node 輸出到第 j 個輸出值的權重)

$$z_j = \sum_{k=1}^{p} w_{kj} h_k$$

經過 非線性轉換/激活函數(activation function，*f*2)後，得到推估的輸出值

$$\hat{y}_j = f_2(z_j)$$

反向傳遞 (Backward propagation)

反向傳遞的目的就是利用最後的目標函數(loss/cost function)來進行參數的更新，一般來說都是用誤差均方和(mean square error)當作目標函數。如果誤差值越大，代表參數學得不好，所以需要繼續學習，直到參數或是誤差值收斂。

x^(i)為第 i 筆資料的輸入值，其輸出值為

$$y^{(i)} = \begin{bmatrix} y_1^{(i)} \\ y_j^{(i)} \\ y_m^{(i)} \end{bmatrix}$$

其目標的誤差為

$$E^{(i)} = \frac{1}{2}\sum_{j=0}^{m}\left(\hat{y}_j^{(i)} - y_j^{(i)}\right)^2$$

所有樣本的誤差和當作目標函數

$$E = \sum_{i=0}^{n} E^{(i)}$$

最佳化的目的就是讓「所有樣本的誤差均方和」越小越好，所以目標是

$$\min_{w_{kj}, v_{ik}} \{E\}$$

所以要找到最佳參數解(參數只有 wkj 和 vik)，最簡單的方式就是微分方程式等於 0 找解

$$\frac{\partial E}{\partial w_{kj}} = 0, \ \frac{\partial E}{\partial v_{ik}} = 0$$

但參數量多無法直接找到唯一解(後面公式有偏微分後的結果，很難直接找到唯一解)，所以還是需要依賴 gradient descent 找最佳解。

假設讀者對 gradient descent 有基本認識。

利用 gradient descent 找最佳參數解(參數只有 wkj 和 vik)

$$w_{kj} = w_{kj} - \eta \Delta w_{kj}$$

$$v_{ik} = v_{ik} - \eta \Delta v_{ik}$$

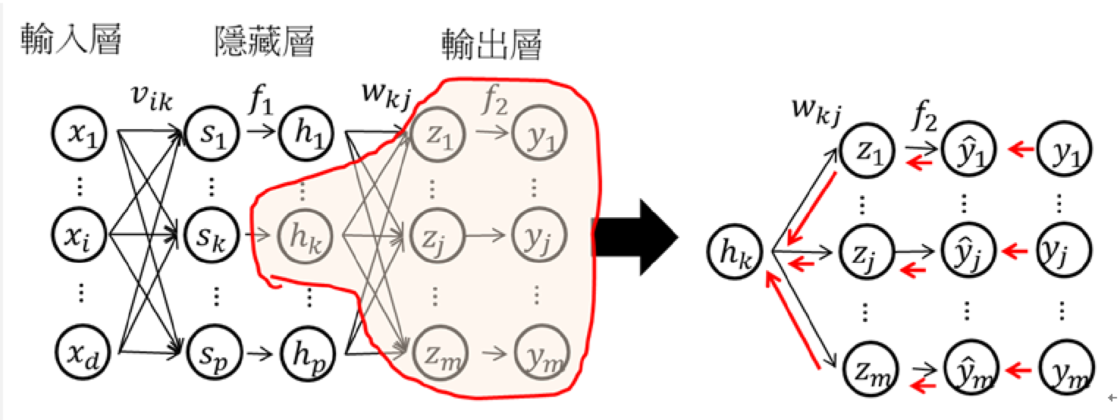
其中 η 為學習率(learning rate)，

$$\Delta w_{kj} = \frac{\partial E}{\partial w_{kj}} = \frac{\partial \sum_{i=0}^{n} E^{(i)}}{\partial w_{kj}} = \sum_{i=0}^{n} \frac{\partial E^{(i)}}{\partial w_{kj}} \text{ (輸出到隱藏層)}$$

$$\Delta v_{ik} = \frac{\partial E}{\partial v_{ik}} = \frac{\partial \sum_{i=0}^{n} E^{(i)}}{\partial v_{ik}} = \sum_{i=0}^{n} \frac{\partial E^{(i)}}{\partial v_{ik}} \text{ (隱藏到輸入層)}$$

基本上微分解無法直接算出，因此用 chain rule 方式，可以更有效得到解，以下針對不同層別的連結算倒傳遞 (只針對一個樣本去計算)

輸出到隱藏層(wkj)



chain rule:

所以$\frac{\partial E^{(i)}}{\partial w_{kj}}$可以拆成兩項($\frac{\partial E^{(i)}}{\partial z_j}$和 $\frac{\partial z_j}{\partial w_{kj}}$)

第一項($\frac{\partial E^{(i)}}{\partial z_j}$):

$$\frac{\partial E^{(i)}}{\partial z_j} = \frac{\partial \frac{1}{2}\sum_{j=0}^{m}\left(\hat{y}_j^{(i)} - y_j^{(i)}\right)^2}{\partial z_j} = \frac{\partial \frac{1}{2}\sum_{j=0}^{m}\left(f_2(z_j^{(i)}) - y_j^{(i)}\right)^2}{\partial z_j}$$

$$= \sum_{j=0}^{m}(\hat{y}_j^{(i)} - y_j^{(i)})\, f_2'(z_j^{(i)})$$

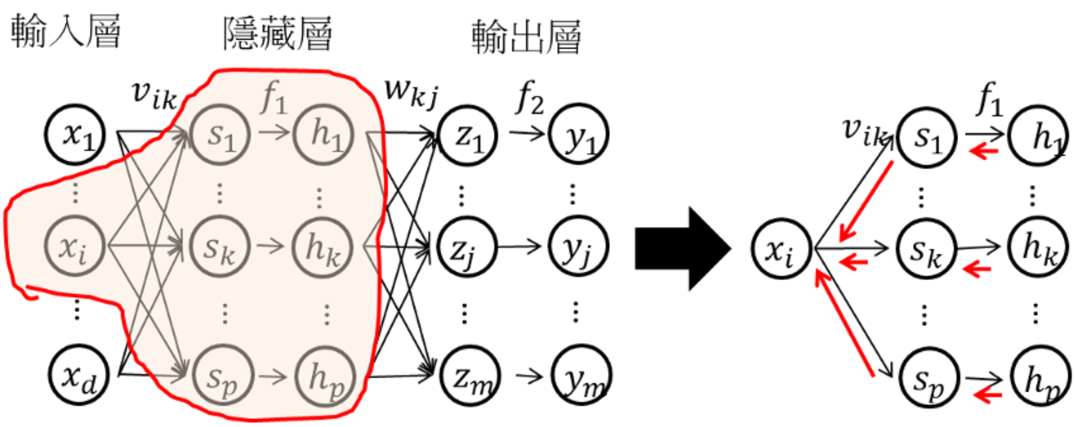
第二項($\frac{\partial z_j}{\partial w_{kj}}$):

$$\frac{\partial z_j}{\partial w_{kj}} = \frac{\partial \sum_{i=0}^{q} w_{kj} h_k}{\partial w_{kj}} = h_k$$

所以

$$\frac{\partial E^{(i)}}{\partial w_{kj}} = \sum_{j=0}^{m}(\hat{y}_j^{(i)} - y_j^{(i)})\, f_2'(z_j^{(i)}) \cdot h_k$$

隱藏層到輸入層(vik)



chain rule:

$$\frac{\partial E^{(i)}}{\partial v_{ik}}=\frac{\partial E^{(i)}}{\partial s_k}\cdot\frac{\partial s_k}{\partial v_{ik}}$$

所以$\frac{\partial E^{(i)}}{\partial v_{ik}}$可以拆成兩項($\frac{\partial E^{(i)}}{\partial s_k}$和 $\frac{\partial s_k}{\partial v_{ik}}$)。

其中$\frac{\partial E^{(i)}}{\partial s_k}$用 chain rule 又可以拆成兩項($\frac{\partial E^{(i)}}{\partial z_j}$和$\frac{\partial z_j}{\partial s_k}$)。

$$\frac{\partial E^{(i)}}{\partial s_k}=\frac{\partial E^{(i)}}{\partial z_j}\cdot\frac{\partial z_j}{\partial s_k}$$

所以$\frac{\partial E^{(i)}}{\partial v_{ik}}$最後可以拆成三項($\frac{\partial E^{(i)}}{\partial z_j}$、$\frac{\partial z_j}{\partial s_k}$和 $\frac{\partial s_k}{\partial v_{ik}}$)

$$\frac{\partial E^{(i)}}{\partial v_{ik}}=\frac{\partial E^{(i)}}{\partial z_j}\cdot\frac{\partial z_j}{\partial s_k}\cdot\frac{\partial s_k}{\partial v_{ik}}$$

2020/08/20 修改所以 E/V 可以拆成兩項的筆誤

第一項($\frac{\partial E^{(i)}}{\partial z_j}$):

$\frac{\partial E^{(i)}}{\partial z_j}$前面有推導了

第二項($\frac{\partial z_j}{\partial s_k}$):

$$\frac{\partial z_j}{\partial s_k}=\frac{\partial\sum_{i=0}^{q}w_{kj}f_1(s_k)}{\partial s_k}=w_{kj}f_1'(s_k^{(i)})$$

第三項($\frac{\partial s_k}{\partial v_{ik}}$):

$$\frac{\partial s_k}{\partial v_{ik}}=\frac{\partial\sum_{i=0}^{d}v_{ik}x_i}{\partial v_{ik}}=x_i$$

所以

$$\frac{\partial E^{(i)}}{\partial v_{ik}}=\frac{\partial E^{(i)}}{\partial z_j}\cdot\frac{\partial z_j}{\partial s_k}\cdot\frac{\partial s_k}{\partial v_{ik}}=\sum_{j=0}^{m}(\hat{y}_j^{(i)}-y_j^{(i)})f_2'(z_j^{(i)})\cdot w_{kj}f_1'(s_k^{(i)})\cdot x_i$$

最後把 n 個樣本所有 gradient 加起來得到參數的 update

$$\Delta w_{kj}=\sum_{i=0}^{n}\frac{\partial E^{(i)}}{\partial w_{kj}}=\sum_{i=0}^{n}\sum_{j=0}^{m}(\hat{y}_j^{(i)}-y_j^{(i)})f_2'(z_j^{(i)})\cdot h_k \quad \text{(輸出到隱藏層)}$$

$$\Delta v_{ik}=\frac{\partial\sum_{i=0}^{n}E^{(i)}}{\partial v_{ik}}=\sum_{i=0}^{n}\sum_{j=0}^{m}(\hat{y}_j^{(i)}-y_j^{(i)})f_2'(z_j^{(i)})\cdot w_{kj}f_1'(s_k^{(i)})\cdot x_i \quad \text{(隱藏到輸入層)}$$

到這邊倒傳遞也推導完成了，看完有沒有覺得很簡單(這邊只是符號多了一點而已，基本上用到的數學應該沒有很多)。

PS:裡面有一個重點，非線性轉換/激活函數(*f1*,*f2*)在倒傳遞時都有微分，所以在選擇激活函數時**必須**要選擇可微分函數。

結論

MLP 神經網路只是在利用 gradient descent 找最佳參數解

$$w_{kj} = w_{kj} - \eta \Delta w_{kj}$$

$$v_{ik} = v_{ik} - \eta \Delta v_{ik}$$

最後帶入 MLP 內的前向傳遞 (Forward propagation)即可得到最後的預測值。