Backpropagation for MLPs

舉個例子，二分類的 r 層 MLP 是：

$$\mathrm{MLP}(x) = \mathrm{MM}_{W^{[r]},b^{[r]}}(\sigma(\mathrm{MM}_{W^{[r-1]},b^{[r-1]}}(\sigma(\cdots \mathrm{MM}_{W^{[1]},b^{[1]}}(x)))))$$

它的 logistic loss 可以被寫成一系列的模組 M 的疊加，具體來說是：

$$z^{[1]} = \mathrm{MM}_{W^{[1]},b^{[1]}}(x)$$

$$a^{[1]} = \sigma(z^{[1]})$$

$$z^{[2]} = \mathrm{MM}_{W^{[2]},b^{[2]}}(a^{[1]})$$

$$a^{[2]} = \sigma(z^{[2]})$$

$$\vdots$$

$$z^{[r]} = \mathrm{MM}_{W^{[r]},b^{[r]}}(a^{[r-1]})$$

$$J = l_{\text{logistic}}(z^{[r]},\ y)$$

而參數都在 $z^{[i]}$ 裡面。開始 backpropagation，首先是一連串求 $\partial J/\partial z^{[i]}$ 的連鎖率：

$$\frac{\partial J}{\partial z^{[r]}} = \mathcal{B}[l_{\text{logistic}},\ z^{[r]}]\left(\frac{\partial J}{\partial J}\right) = \mathcal{B}[l_{\text{logistic}},\ z^{[r]}](1)$$

$$\frac{\partial J}{\partial a^{[r-1]}} = \mathcal{B}[\mathrm{MM}_{W^{[r]},b^{[r]}},\ a^{[r-1]}]\left(\frac{\partial J}{\partial z^{[r]}}\right),\ \frac{\partial J}{\partial z^{[r-1]}} = \mathcal{B}[\sigma,\ z^{[r-1]}]\left(\frac{\partial J}{\partial a^{[r-1]}}\right)$$

$$\vdots$$

$$\frac{\partial J}{\partial a^{[1]}} = \mathcal{B}[\mathrm{MM}_{W^{[2]},b^{[2]}},\ a^{[1]}]\left(\frac{\partial J}{\partial z^{[2]}}\right),\ \frac{\partial J}{\partial z^{[1]}} = \mathcal{B}[\sigma,\ z^{[1]}]\left(\frac{\partial J}{\partial a^{[1]}}\right)$$

再來是一連串對 $\partial J/\partial W^{[i]}$, $\partial J/\partial b^{[i]}$ 的連鎖率：

$$\frac{\partial J}{\partial W^{[r]}} = \mathcal{B}[\mathrm{MM}_{W^{[r]},b^{[r]}},\ W^{[r]}]\left(\frac{\partial J}{\partial z^{[r]}}\right),\ \frac{\partial J}{\partial b^{[r]}} = \mathcal{B}[\mathrm{MM}_{W^{[r]},b^{[r]}},\ b^{[r]}]\left(\frac{\partial J}{\partial z^{[r]}}\right)$$

$$\vdots$$

$$\frac{\partial J}{\partial W^{[1]}} = \mathcal{B}[\mathrm{MM}_{W^{[1]},b^{[1]}},\ W^{[1]}]\left(\frac{\partial J}{\partial z^{[1]}}\right),\ \frac{\partial J}{\partial b^{[1]}} = \mathcal{B}[\mathrm{MM}_{W^{[1]},b^{[1]}},\ b^{[1]}]\left(\frac{\partial J}{\partial z^{[1]}}\right)$$

接著，分別討論關於 loss function, MM 和 activation 這些模組的 Jacobian 轉置。

(1) loss function 的 Jacobian 轉置。由於 $\partial J/\partial t$ ($t = \theta^T x \in \mathbb{R}^n$), n 是最後隱藏層的神經元數量) 是一個純量（以下都是向量式）：

$$\mathcal{B}[l_{\text{logistic}},\ t](v) = \frac{\partial l_{\text{logistic}}(t,\ y)}{\partial t} \cdot v = \left(\frac{1}{1+\exp(-t)} - y\right) \cdot v$$

(2) MM(z)：

$$\mathcal{B}[\text{MM}, z](v) = \begin{bmatrix} \frac{\partial (Wz+b)_1}{\partial z_1} & \cdots & \frac{\partial (Wz+b)_n}{\partial z_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial (Wz+b)_1}{\partial z_m} & \cdots & \frac{\partial (Wz+b)_n}{\partial z_m} \end{bmatrix} v$$

由於：

$$\forall i \in \mathbb{R}^m,\ j \in \mathbb{R}^n,\ \frac{\partial (Wz+b)_j}{\partial z_i} = \frac{\partial b_j + \sum_{k=1}^m W_{jk} z_k}{\partial z_i} = W_{ji}$$

因此（如果忘記，回去看 W 的定義）：

$$\mathcal{B}[\text{MM}, z](v) = W^T v$$

MM(W)：為了規避對矩陣求導的複雜性（Jacobian 轉置會變成一個四維張量），能把 $W \in \mathbb{R}^{n\times m}$ 攤平成向量 $W = (W_{11},\ W_{12},\ \cdots,\ W_{1m},\ \cdots,\ W_{n1},\ W_{n2},\ \cdots,\ W_{nm}) \in \mathbb{R}^{nm}$。通過目測求導，Jacobian 轉置為：

$$\mathcal{B}[\text{MM}, W] = \begin{bmatrix} \frac{\partial (Wz+b)_1}{\partial W_{11}} & \frac{\partial (Wz+b)_2}{\partial W_{11}} & \cdots & \frac{\partial (Wz+b)_n}{\partial W_{11}} \\ \frac{\partial (Wz+b)_1}{\partial W_{12}} & \frac{\partial (Wz+b)_2}{\partial W_{12}} & \cdots & \frac{\partial (Wz+b)_n}{\partial W_{12}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial (Wz+b)_1}{\partial W_{1m}} & \frac{\partial (Wz+b)_2}{\partial W_{1m}} & \cdots & \frac{\partial (Wz+b)_n}{\partial W_{1m}} \\ \frac{\partial (Wz+b)_1}{\partial W_{21}} & \frac{\partial (Wz+b)_2}{\partial W_{21}} & \cdots & \frac{\partial (Wz+b)_n}{\partial W_{21}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial (Wz+b)_1}{\partial W_{nm}} & \frac{\partial (Wz+b)_2}{\partial W_{nm}} & \cdots & \frac{\partial (Wz+b)_n}{\partial W_{nm}} \end{bmatrix} = \begin{bmatrix} z_1 & 0 & \cdots & 0 \\ z_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ z_m & 0 & \cdots & 0 \\ 0 & z_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & z_m \end{bmatrix} \in \mathbb{R}^{nm\times n}$$

則：

$$\mathcal{B}[\mathrm{MM}, W](v) = \begin{bmatrix} v_1 z_1 \\ \vdots \\ v_1 z_m \\ \vdots \\ v_n z_1 \\ \vdots \\ v_n z_m \end{bmatrix}$$

將此式以 row 優先順序重塑為 $n \times m$ 矩陣，得：

$$\mathcal{B}[\mathrm{MM}, W](v) = vz^T$$

以上是教科書的解釋，但我覺得這個「重塑」說明得不清楚，同時也找到了更好的解釋如下。

根據 chain rule：

$$\frac{\partial J}{\partial W_{ij}} = \sum_{k=1}^{n} \frac{\partial J}{\partial y_k} \cdot \frac{\partial y_k}{\partial W_{ij}}$$

計算 $\partial y_k / \partial W_{ij}$：

$$y_k = \sum_{l=1}^{m} W_{kl} z_l + b_k$$

所以，當 $k \neq i$，$\partial y_k / \partial W_{ij} = 0$；當 $k = i$，$\partial y_k / \partial W_{ij} = z_j$。於是，chain rule 變成：

$$\frac{\partial J}{\partial W_{ij}} = \sum_{k=1}^{n} \frac{\partial J}{\partial y_k} \cdot \frac{\partial y_k}{\partial W_{ij}} = \frac{\partial J}{\partial y_i} \cdot z_j$$

所以，整個 $\partial J / \partial W$ 是：

$$\left[\frac{\partial J}{\partial W}\right]_{ij} = \left[\frac{\partial J}{\partial y}\right]_i \cdot z_j \;\Rightarrow\; \frac{\partial J}{\partial W} = \left(\frac{\partial J}{\partial y}\right) z^T$$

MM(b)：

$$\mathcal{B}[\mathrm{MM}, b](v) = \begin{bmatrix} \frac{\partial (Wz+b)_1}{\partial b_1} & \cdots & \frac{\partial (Wz+b)_n}{\partial b_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial (Wz+b)_1}{\partial b_n} & \cdots & \frac{\partial (Wz+b)_n}{\partial b_n} \end{bmatrix} v = v$$

(3) activation：

$$\begin{aligned} \mathcal{B}[\sigma, z](v) &= \begin{bmatrix} \frac{\partial \sigma(z_1)}{\partial z_1} & \cdots & \frac{\partial \sigma(z_m)}{\partial z_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \sigma(z_1)}{\partial z_m} & \cdots & \frac{\partial \sigma(z_m)}{\partial z_m} \end{bmatrix} v = \mathrm{diag}(\sigma'(z_1), \cdots, \sigma'(z_m))\, v \\ &= \sigma'(z) \odot v \end{aligned}$$

這裡的 $\odot$ 符號表示逐元素乘法 (element-wise product)。

Vectorization over Training Examples

實務上，在 numpy 環境中，我們會將所有輸入數據組合成一個陣列進行訓練。
假設有三筆輸入數據，為：

$$\begin{aligned} z^{[1](1)} &= W^{[1]} x^{(1)} + b^{[1]} \\ z^{[1](2)} &= W^{[1]} x^{(2)} + b^{[1]} \\ z^{[1](3)} &= W^{[1]} x^{(3)} + b^{[1]} \end{aligned}$$

定義：

$$X = \begin{bmatrix} | & | & | \\ x^{(1)} & x^{(2)} & x^{(3)} \\ | & | & | \end{bmatrix} \in \mathbb{R}^{d \times 3}$$

有：

$$Z^{[1]} = \begin{bmatrix} | & | & | \\ z^{[1](1)} & z^{[1](2)} & z^{[1](3)} \\ | & | & | \end{bmatrix} = W^{[1]}X + b^{[1]}$$

我們會注意到 b 的維度是不正確的。但是 numpy 有一個叫做“broadcast”的功能，會用既有的值自動補齊 b 的維度。這個部分不需要擔心。

一個更重要的問題是關於深度學習軟體的 row major convention。理論上（例如教科書或學術論文），通常將每個數據點表示為一個 column vector，如果有多個數據點，它們會被排列在一個矩陣的列中。但是實務上，大多數深度學習庫（如 TensorFlow、PyTorch 等）和實際程式碼實現中，更常見的習慣是將數據點放在數據矩陣的 row 中。為了在理論符號和實際實現之間進行轉換，需要將列向量變成行向量、行向量變成列向量、所有的矩陣都需要轉置，同時矩陣乘法的順序需要翻轉。

即：

$$Z^{[1]} = XW^{[1]} + b^{[1]} \in \mathbb{R}^{3\times m}$$