深度學習
Lab Assingment 2

1. 目的

- 撰寫 Logistic Regression 演算法
- 理解機器學習的訓練/測試階段

2. 課程章節與實驗環境

2.1. 課程章節

Weeks 5&6: Logistic Regression

2.2. 實驗環境

限定使用 Python 程式語言

3. 實驗進行步驟及結果

3.1. 自行撰寫 Logistic Regression 程式，不可使用套裝軟體現成程式

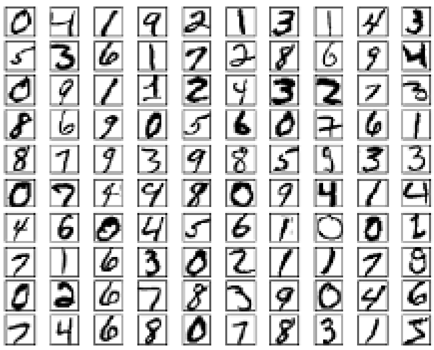
- 激活函數(Activation Function)為 Sigmoid Function $\sigma(n) = \frac{1}{1+e^{-n}}$
- 損失函數(Loss Function)為**二元**交叉熵(Binary Cross-Entropy):

$$E(w_1, w_2, b) = -(y \ln \hat{y} + (1-y) \ln (1-\hat{y}))$$

其中 y (= 0 或 1)為標籤，$\hat{y} \in (0, 1)$為神經元輸出值

3.2. 輸入資料

本實驗採用 **MNIST** 數字數據集，僅使用部分資料，辨識 2, 5 兩個數字。輸入資料為 784 維，助教準備具有類別資料 4000 筆，提供同學訓練及驗證，另有 1000 筆無類別測試資料，請同學預測。



- 使用讀檔方式讀取訓練資料(train.csv)，測試資料(test.csv)
- train.csv 共含 4000 筆具有兩類別 2, 5 數字
- test.csv 共含 1000 筆無類別測試資料

3.3. 輸出資料

當程序停止時，顯示停止條件，如世代(epoch)數或符合容忍誤差、學習率、訓練準確率、測試資料預測結果等。

本次作業分數，含括測試準確率高低；即測試準確率高，作業分數較高

※ **請依照助教指示之輸入/輸出格式要求**

4. 說明

4.1. Linear Regression 與 Logistic Regression 演算法過程相似

- 可以按照 Linear Regression 過程實施 Logistic Regression。 但是 Linear Regression 輸出值為 $\hat{y} = n \in$ R；而 Logistic Regression 輸出值為 $\sigma(n)$。 即，$\hat{y} = \sigma(n)$，並且 $0 < \hat{y} < 1$。
- 請注意 Logistic Regression 訓練過程 $\hat{y} \in (0, 1)$，是一個機率值，介於 0 與 1 之間。計算訓練準確率，及預測測試資料的分類標籤時，才需要將機率轉換為類別標籤。換言之，如果 $\hat{y} \geq 0.5$，則預測為 1； 否則為 0。

4.2. Linear Regression 與 Logistic Regression 比較

	Activation Function	Error function (one training example)	Gradient Descent Algorithm
Linear Regression	**Linear** function $\hat{y} = f(n) = n$	**Half of the squared error** $\frac{1}{2}(\hat{y} - y)^2$	$\mathbf{w} \leftarrow \mathbf{w} - \eta(\hat{y} - y)\mathbf{x}$ $b \leftarrow b - \eta(\hat{y} - y)$ where $\hat{y} = f(n) = n$ $= b + w_1x_1 + \ldots + w_Mx_M$
Logistic Regression	**Logistic** function $\hat{y} = \sigma(n)$ $= \dfrac{1}{1 + e^{-n}}$	**Cross-entropy error** $-(y \ln\hat{y} + (1 - y) \ln(1 - \hat{y}))$	$\mathbf{w} \leftarrow \mathbf{w} - \eta(\hat{y} - y)\mathbf{x}$ $b \leftarrow b - \eta(\hat{y} - y)$ where $\hat{y} = \sigma(n) = \frac{1}{1+e^{-n}}$ $= \dfrac{1}{1 + e^{-(b + w_1x_1 + \cdots + w_Mx_M)}}$

4.3. 本程式作業可採用 **Stochastic version** 或 **Batch version**.

- **Stochastic version:** 檢查每筆訓練資料後，立即更新權重和偏差值

Logistic Regression with Stochastic Gradient Descent Algorithm

Step 1 Input training data set:

$D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ // N training examples, M attributes

Step 2 Initialize $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}$ and b to random values; Choose a learning rate η.

Step 3 UNTIL a termination condition is met, DO

Step 4 FOR each training example $\mathbf{x} \in D$

Step 4.1 // Generate the estimated value for a single example

$$n = b + \mathbf{w}^T\mathbf{x} = b + [w_1, w_2, \ldots, w_M]\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$$

$$\hat{y} = \sigma(n) = \frac{1}{1+e^{-n}}$$

Step 4.2 // Update the weight vector $\mathbf{w}$ and the bias b

$\mathbf{w} \leftarrow \mathbf{w} - \eta(\hat{y} - y)\mathbf{x}$ // Update the weight vector once per example

$b \leftarrow w - \eta(\hat{y} - y)$ // Update the bias once per example

– **Batch version:** 檢查所有訓練資料後，才更新權重和偏差值

Logistic Regression with Batch Gradient Descent Algorithm

Step 1 Input training data set:

$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ // N training examples, M attributes

$$= \left\{ \left(\begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{M1} \end{bmatrix}, y_1 \right), \left(\begin{bmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{M2} \end{bmatrix}, y_2 \right), \cdots, \left(\begin{bmatrix} x_{1N} \\ x_{2N} \\ \vdots \\ x_{MN} \end{bmatrix}, y_N \right) \right\}$$

Let $\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix}$, and $\mathbf{y} = [y_1, y_2, \ldots, y_N]$

Step 2 Initialize $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}$ and b to random values; Choose a learning rate η.

Step 3 UNTIL a termination condition is met, DO

Step 3.1 // Generate estimated values for all examples in D

// $\hat{\mathbf{y}} = [\widehat{y_1}, \widehat{y_2}, \ldots, \widehat{y_N}]$

$$// = \sigma\left(\left([w_1, w_2, \ldots, w_M] \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} \right)_{1\times N} + ([b, b, \ldots, b])_{1\times N} \right)$$

$\hat{\mathbf{y}} = \sigma(\mathbf{w}^T\mathbf{X} + b)$ // **Broadcasting in Numpy**

Step 3.2 // Compute the average of gradients

$$// \ \mathbf{dw} = \frac{1}{N}\left(\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} \begin{bmatrix} \widehat{y_1} - y_1 \\ \widehat{y_2} - y_2 \\ \vdots \\ \widehat{y_N} - y_N \end{bmatrix} \right)_{M\times 1}$$

$$// \ \mathbf{dw} = \frac{1}{N}\left(\begin{bmatrix} x_{11}(\widehat{y_1} - y_1) + x_{12}(\widehat{y_2} - y_2) + \cdots + x_{1N}(\widehat{y_N} - y_N) \\ x_{21}(\widehat{y_1} - y_1) + x_{22}(\widehat{y_2} - y_2) + \cdots + x_{2N}(\widehat{y_N} - y_N) \\ \vdots \\ x_{M1}(\widehat{y_1} - y_1) + x_{M2}(\widehat{y_2} - y_2) + \cdots + x_{MN}(\widehat{y_N} - y_N) \end{bmatrix} \right)_{M\times 1}$$

$$\mathbf{dw} = \frac{1}{N}(\mathbf{X}(\hat{\mathbf{y}} - \mathbf{y})^T)_{M\times 1}$$

$$// \ \mathrm{d}b = \frac{1}{N}\big((\widehat{y_1} - y_1) + (\widehat{y_2} - y_2) + \cdots + (\widehat{y_N} - y_N)\big)_{1\times 1}$$

$$\mathrm{d}b = \frac{1}{N}\sum_{k=1}^{N}(\widehat{y_k} - y_k)$$

Step 3.3 // Update the weight vector $\mathbf{w}$ and the bias b

$\mathbf{w} \leftarrow \mathbf{w} - \eta\,\mathbf{dw}$ // Update the weight vector once per epoch

$b \leftarrow b - \eta\,\mathrm{d}b$ // Update the bias once per epoch

4.4. 梯度下降(Gradient descent)演算法的停止準則通常包括：

(a) 最大世代(Epoch)數

(b) 當訓練資料的某種誤差度量足夠小時停止，比如設定容忍誤差，$\tau = 10^{-2}$。若y_k為訓練資料輸出屬性值，$\widehat{y_k}$為估計值，常見誤差度量如下:

(i) Mean Squared Error (MSE) = $\frac{1}{N}\sum_{k=1}^{N}(y_k - \widehat{y_k})^2 < \tau$

(ii) Root Mean Squared Error (RMSE) = $\sqrt{\text{MSE}} < \tau$

(iii) Mean Absolute Error (MAE) = $\frac{1}{N}\sum_{k=1}^{N}|y_k - \widehat{y_k}| < \tau$

(iv) Average of the cross-entropy loss:

$$-\frac{1}{N}\sum_{k=1}^{N}(y_k \, ln \, \widehat{y_k} + (1 - y_k) \, ln \, (1 - \widehat{y_k})) < \tau$$

(c) 在連續若干次世代迭代中，訓練沒有任何改進時，可提早終止，例如誤差度量停滯不再下降時。

(d) 當觀察到 overfitting 時，則提早終止，停止訓練。

- 停止條件 (a)為強迫停止。
- 停止條件(b)是因為某種誤差度量足夠小而停止，代表訓練模型佳，即權重 **w** 及偏差值 b 對訓練資料產生不錯的估計值，因此對預測測試資料的正確性較有信心。
- 停止條件(c)是認為再繼續訓練，不會有較佳結果，因此提早終止。
- 停止條件(d)是因為擔心過度訓練，反而降低測試資料的預測正確性。世代數的增加，一般可以提升訓練資料的準確率，可是未必提高測試準確率。偵測是否 overfitting，可保留一部分訓練資料，作為驗證資料，驗證資料不參與訓練。當世代數增加，訓練資料準確率雖然提高，但驗證資料準確率卻反而降低時，即產生所謂的 overfitting，此時應選擇停止訓練。

※ 本次程式作業至少需同時考慮(a)及(b)其中一種誤差度量。