Simple Classifier / Logistic Regression
B083040029 邱品諺

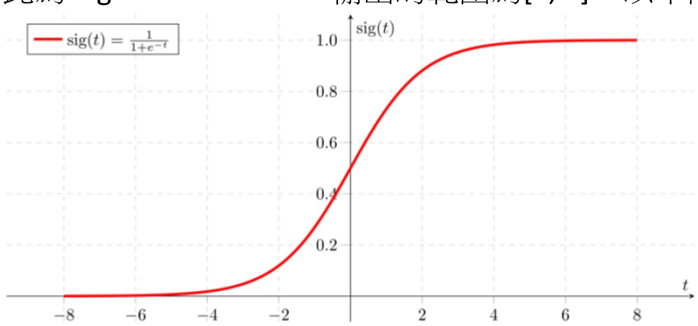
本次作業利用 Logistic Regression 預測房價高低，資料集來自於 HousingPrice dataset，此資料集 1460 筆資料，每筆資料有 81 個特徵。

0. Dataloading and Data Preprocessing
   載入資料的部分，training, validation, testing 資料的比例分別為 60%, 20%, 20%。以資料集的 GrLivArea 欄位為每一筆資料的訓練特徵，以資料集的 SalePrice 欄位作為每一筆資料的目標預測欄位，並將每筆數值資料進行正規化（Normalization），再以 np.stack 函式將資料轉換成符合矩陣運算的 shape。最後，此部分最後處理完的資料集僅保留了部分資料，因其以 ground truth 的第 30 百分位數及第 70 百分位數為基準，將大於第 70 百分位數的 ground truth 設為 1、將小於第 30 百分位數的 ground truth 設為 0，並刪去其他的資料，另外，此問題也變為一個 Binary Classification 的問題。

1. Set up a Classifier Model
   此為 Logistic Regression 的 Model，可以看作 Linear Regression 再加上一個非線性的 Sigmoid Function。

$$\hat{y} = \sigma(\mathrm{X} \cdot w)$$

   此為 Sigmoid Function，輸出的範圍為[0, 1]。以本問題而言，0 為低價、1 為高價。



$$\sigma(t) = \frac{1}{1+e^{-t}}$$

2. Loss: Binary Cross Entropy

$$BCE(y, \hat{y}) = -y \cdot log(\hat{y}) - (1 - y) \cdot log(1 - \hat{y})$$

   由於此問題為 Binary Classification，因此挑選 BCE 為 Loss Function。Loss Function 計算出的數值越低，表示預測值與 ground truth 愈接近，誤差值越小，準確率越高，反之亦然。另外，由於 BCE 是一個 non-convex function，所以它沒有 closed-form 解可以直接將解算出來，需要利用 numeric methods 算出，像是接下來將使用的 Gradient Descent。

3. Backpropagation

$$\frac{\partial L(w)}{\partial w} = \frac{\partial L(w)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w}$$

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial \sigma(s)}{\partial w} = \frac{\partial \sigma(s)}{\partial s} \cdot \frac{\partial s}{\partial w}$$

Backpropagation 的主要目的為透過計算 Loss Function 對 Network 中各個 weight 的 Gradient，以得知每個參數需要調整的方向。因為如果想要得到某個 Function 的極值，在微積分中我們可以利用微分=0 求得，此處即使用此概念。

而 Loss Function 對 weight 的微分藉由 Chain Rule 其實可表示為 Loss Function 對 Prediction Value 的 Gradient 乘上 Prediction Value 對 weight 的 Gradient，其中 Prediction Value 對 weight 的 Gradient 又可利用 Chain Rule 推為 Sigmoid Function 對 $s = X \cdot w$的 Gradient 乘上$s = X \cdot w$對 weight 的 Gradient。

以下為 Loss 對 weight 微分經過 Chain Rule 拆解的結果：

$$\frac{\partial L(w)}{\partial \hat{y}} = (-\frac{y}{\hat{y}} - \frac{1-y}{-(1-\hat{y})})$$
$$\frac{\partial \hat{y}}{\partial w} = \sigma(s) \cdot \big(1-\sigma(s)\big) \cdot X$$
$$\frac{\partial L(w)}{\partial w} = (-\frac{y}{\hat{y}} - \frac{1-y}{-(1-\hat{y})}) \cdot \sigma(s) \cdot \big(1-\sigma(s)\big) \cdot X$$

4. Optimizer and Gradient Descent

定義好 Loss Function 後，接著需要 Optimizer 一步一步的更新 Model 中的參數，而更新的方法為 Gradient Descent，其透過向 Gradient 的反方向持續向低谷（最小值）前進，直到 Converge 亦或是達到規定的訓練次數。

$$w^{(n+1)} = w^{(n)} - \alpha \cdot \frac{dL}{dw}$$

Gradient Descent 步驟：

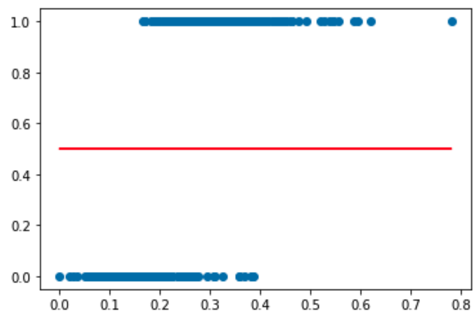
(1) 隨機初始化 Network 中的 weight
(2) 計算 Loss Function
(3) 計算 Loss Function 對各 weight 的 Gradient
(4) 依據 Gradient 更新 weight
(5) 持續執行 2~4 直到訓練完畢

5. Training & Solver

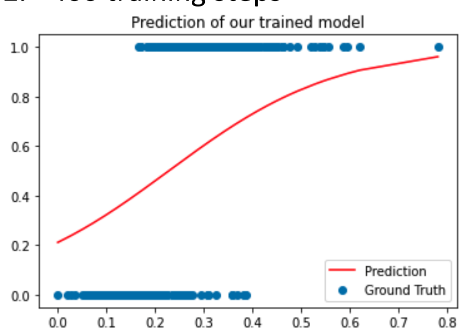
訓練中除了要執行上述 Gradient Descent 每個 epoch 更新 weights 的步驟外，訓練之前需要先決定幾個超參數（Hyperparameter），像是 Learning rate、epochs (steps)、Batch size 等等，而現今多數的訓練方式都一個一個 batch 下去訓練並計算

平均的 Gradient 以更新 Weights，直到訓練至設定的 epochs 數模型才訓練完畢。以下為本問題不同的訓練次數所訓練出 Model 結果（藍點為資料點、紅線為預測回歸線）：

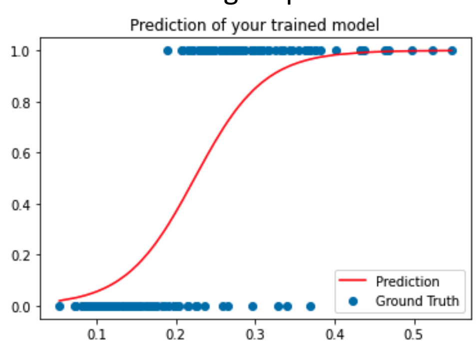
1. 無訓練



2. 400 training steps



3. 25000 training steps



我們可以看到訓練到一定的次數，Model 會學得較好、較貼近資料的曲線，Loss 也會比較低，也表示準確率會比較高。