MLP Regression

結構化機器學習第一次作業

林祐陞
國立中興大學 (統計研究所)
學號: 7107018017
email : 7107018017@smail.nchu.edu.tw

***Abstract*—這次使用的資料是美國 Boston 的房價預測資料，在 Sklearn 的 Sample Data 中可以找到。將結構化機器學習課堂中學到的 MNIST MLP model 進行修改，輸出為 Regression 數值而不再是類別，並將 cost function 更改為 SSE。**

I. Data Introduction

從 Sklearn 中可找到本次實驗用資料，為美國統計局所收集的 Boston 房價資料，目的為預測房價，總共有 13 個 variable 及 1 個 target。

Samples total	506
Dimensionality	13
Features	real, positive
Targets	real 5. - 50.

Fig. 1. Data introduction on sklearn

CRIM: 所在城市犯罪率
ZN: 佔地面積超過 25,000 平方呎的住宅用地比例
INDUS: 城鎮的非零售業務面積比例
CHAS: 虛擬變量，靠河岸為 1，其他為 0
NOX: 空氣氮氧化合汙染物濃度
RM: 每棟住宅的平均房間數量
AGE: 屋齡
DIS: 到波士頓五個就業中心的加權距離
RAD: 到高速公路的遠近指標
TAX: 每 $10,000 的地價稅
PTRATIO: 城鎮的學生與教師比例
B: $1000(Bk - 0.63)^2$, Bk 為非洲裔美國人後裔
LSTAT: 低階層人口比率
MEDV: 自住宅的房價中位數，單位為 $1000

其中的 MEDV 則是我們此次預測的 target。並將 Data 拆分為 70% 作為 training data，30% 作為 testing data。

II. Two model differences

在這次 Regression 模型中，不需要原本應用於 MNIST dataset 中的 softmax 層，輸出層的節點數為 1，直接 output 即為房價輸出。
Cost function 則更改為 SSE。

$$Cost = \sum_i \left\| \hat{y^{(i)}} - y^{(i)} \right\|^2$$

由於 sigmoid 在 ±4 之外斜率近乎為 0，會造成梯度消失問題，在這次房價預測中，我使用的是主流的 ReLu 的 Activation function，在 code 編寫上也簡單許多。

$$ReLu\,(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geqslant 0 \end{cases}$$

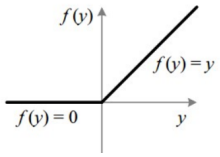


Fig. 2. ReLu Activation Function

III. Model structure

如同原先 MNIST MLP 的 model，設置為兩層 neural network，一層隱藏層，一層輸出層。隱藏層節點數目設置為 30，而輸出則將節點數目改為 1。
而在超參數上 epoch 我選擇的是 1000，學習率 η 則是選擇 0.00005，後面第 IV 章會解釋如何決定。

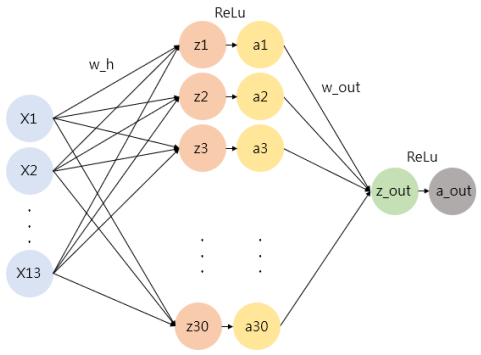


Fig. 3. Model structure

這次設計的 Neural Network，由 13 個 input 連結隱藏層的 30 個節點，再進入 ReLu 進行非線性轉換，再進入到 z_out，最後再做一次 ReLu 的轉換，當作輸出。
更新方式則是透過 Backpropagation，推導過程為

$$SSE = \sum (\hat{y} - y)^2 = (\hat{Y} - Y)^T (\hat{Y} - Y)$$

$$\frac{\partial SSE}{\partial W_{out}} = \frac{\partial (\hat{Y} - Y)^T (\hat{Y} - Y)}{\partial A_{out}} \frac{\partial A_{out}}{\partial Z_{out}} \frac{\partial Z_{out}}{\partial W_{out}}$$

可得 W_{out} 的梯度為:

$$\frac{\partial SSE}{\partial W_{out}} = (A_h)^T 2(\hat{Y} - Y)\frac{\partial ReLu(Z_{out})}{\partial Z_{out}}$$

另外，

$$\frac{\partial A_h}{\partial W_h} = \frac{\partial A_h}{\partial Z_h}\frac{\partial Z_h}{\partial W_h} = X^T\frac{\partial ReLu(Z_h)}{\partial Z_h}$$

$$\frac{\partial SSE}{\partial A_h} = \frac{\partial SSE}{\partial A_{out}}\frac{\partial A_{out}}{\partial Z_{out}}\frac{\partial Z_{out}}{\partial A_h}$$

$$= (W_{out})^T 2(\hat{Y} - Y)\frac{\partial ReLu(Z_{out})}{\partial Z_{out}}$$

$$= (W_{out})^T \delta_{out}\frac{\partial ReLu(Z_{out})}{\partial Z_{out}}$$

即可得 W_h 的梯度為:

$$\frac{\partial SSE}{\partial W_h} = \frac{\partial SSE}{\partial A_h}\frac{\partial A_h}{\partial Z_h}\frac{\partial Z_h}{\partial W_h}$$

$$= X^T(W_{out})^T 2(\hat{Y} - Y)\frac{\partial ReLu(Z_{out})}{\partial Z_{out}}\frac{\partial ReLu(Z_h)}{\partial Z_h}$$

$$= X^T(W_{out})^T \delta_{out}\frac{\partial ReLu(Z_h)}{\partial Z_h}$$

$$= X^T\delta_h\frac{\partial ReLu(Z_h)}{\partial Z_h}$$

將誤差 δ 層層傳遞回去更新 Weight 值，例如:

$$w_{x_1,z_1}{}' = w_{x_1,z_1} - \eta\delta_1\frac{\partial a_1}{\partial z_1}x_1$$

而在這之中 $\frac{\partial a_1}{\partial z_1}$ 對於 z_1 層中的正數來說即是 1，其餘為 0。

$$\frac{\partial a_1}{\partial z_1} = \begin{cases} 0 \; for \; z_1 < 0 \\ 1 \; for \; z_1 \geqslant 0 \end{cases}$$

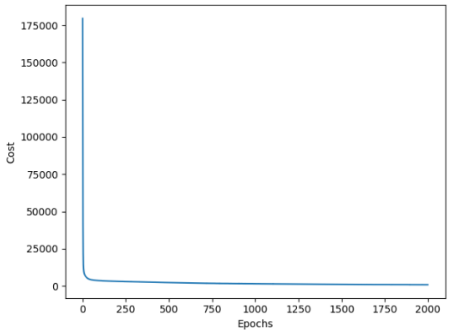


Fig. 4. Cost reduction

並以此梯度更新的方式進行參數的修正，逐漸得到更佳的 Weight 值，使 Cost 值逐漸下降。

IV. Accuracy Selection

在模型的評估上，我使用了 RMSE 和 R^2，由於 SSE 跟樣本數目有關，RMSE 比起原本 Cost 的 SSE 更能良好的評估模型，可以將訓練資料集和測試資料集一同作圖比較。

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}}$$

而 R^2 能夠說明模型對於資料的解釋力，若 R^2 越接近 1 說明模型越能夠解釋資料。

$$SS_{res} = \sum_i (y_i - f_i)^2, f_i = a_{out}$$

$$SS_{tot} = \sum_i (y_i - \bar{y})^2$$

$$R^2 \equiv 1 - SS_{res}/SS_{tot}$$

V. Hyper parametric decision

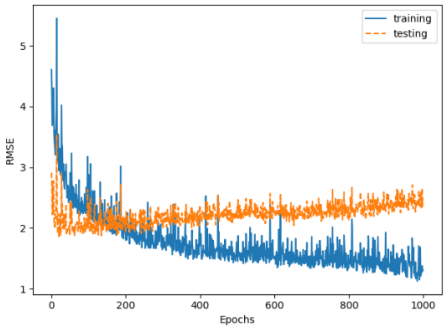


Fig. 5. eta = 0.001

在超參數決定上，發現若學習率 η 在 0.00005 以上，Cost 會激烈抖動，說明 η 過大，需要下修。若取值太小，學習過程會變得很緩慢。而在 epoch 上我先測試為 2000，

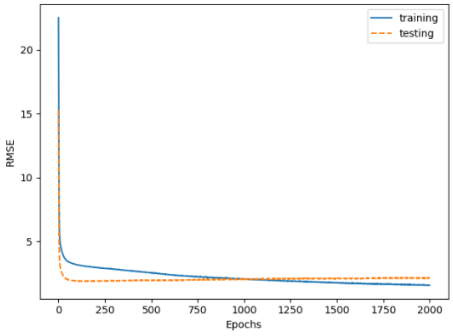
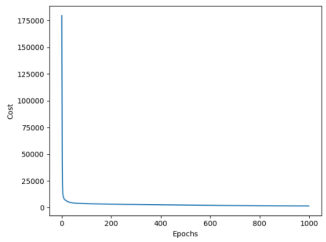


Fig. 6. RMSE for train and test data

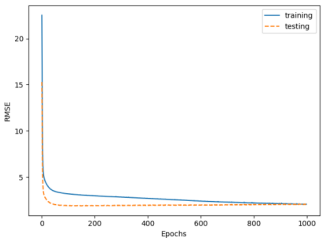
並將 train 跟 test 的 RMSE 都畫出來，發現在大約 1000 左右，發生了 train 持續下降，而 test 不再下降，甚至微微上升，發生了一次交叉。這也說明了在 epoch 為 1000 之後發生 Overfitting，因此選擇 epoh 值使用 1000。

VI. Model Accuracy

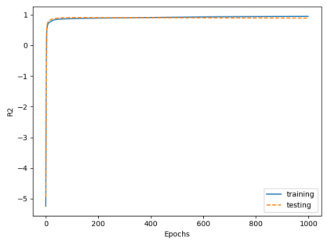
因此使用 epoch 為 1000，學習率 η 為 0.00005 的狀況，將數值和圖形印出來。



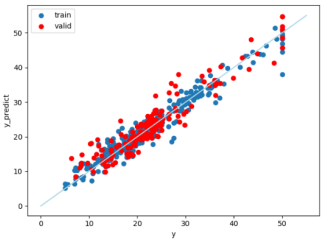
(a) Cost reduction



(b) RMSE for train & test



(c) R^2 for train & test



(d) Estimated with actual

Fig. 7. epoch:1000, η:0.00005

```
Final cost of train =  894.2511718858543
Final RMSE of train =  1.5893814060554636
Final RMSE of test  =  2.1393279732078643
Final R_2 of train =  0.9688857938144312
Final R_2 of test  =  0.8837050965540066
```

(a) epoch:2000

```
Final cost of train =  1507.25056491687
Final RMSE of train =  2.063436443749186
Final RMSE of test  =  2.077977770439038
Final R_2 of train =  0.9475573459397997
Final R_2 of test  =  0.8902795103076426
```

(b) epoch:1000

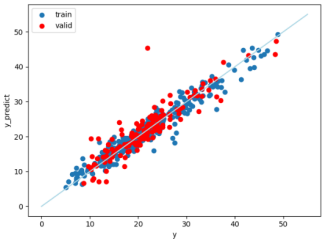
Fig. 8. Value differences

而在數值上我們可以看一下跟前一次的比較，將 epoch 下降為 1000 後，雖然 RMSE of train 上升，但 RMSE of test 卻是下降的，而且相對於 2000 epoch，1000 epoch 的 R_2 of test 較高，這代表 2000 epoch 確實造成了 Overfitting 的狀況，各方面來看 epoch 1000 才是更好的選擇。

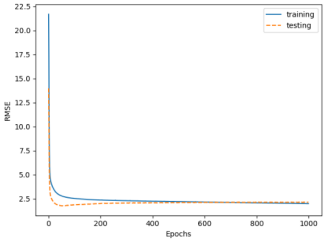
而 RMSE 在 train 跟 test 都很接近壓在 2.06 左右，代表預測準度還不錯，而 R^2 也在 0.9 左右，代表模型很高程度上的解釋了數據，說明是一個還不錯的模型。

VII. Additional discussion

可以觀察一下 Fig.7 的 (d)，為估計值 $\hat{y}$ 和實際值 y 作圖，可以發現在最右邊 y 為 50 的地方發生了一列離群值。回去檢查資料，發現數值最高的 50 很可能並非真的是 50，而是資料紀錄的問題，有可能房價超過 $50,000 即紀錄為 50。若真是如此會影響模型準確率，若我們將房價紀錄為 50 的先移除觀察一下。



(a) Estimated with actual



(b) RMSE for train & test

```
Final RMSE of train =  1.9795167307340453
Final RMSE of test  =  2.1284437485647754
```

(c) Value

Fig. 9. data without y=50

可以發現 train 的 RMSE 確實下降了一些，而預測值也更貼近實際值了，也就是說確實有資料紀錄上的問題，但 test 的 RMSE 反倒上升了一些，很可能是因為左上角那筆離群值的關係。

VIII. Conclusion

在這次 Boston 的房價預測資料中，學習到 Neural Network 在 Regression 上的使用，並透過簡單的調整超參數方式來達到更好的模型，也能夠大略的發現資料上的離群值，或許可以像 KFold 進行多次訓練，並平均模型參數的方式得以穩定。