Machine Learning in Engineering Science

N96084094 彭巧緣

Homework 4

1. Concept and Derivation

(a)

$$u^{(1)} = \begin{bmatrix} 1 \times 0.1 + 2 \times 0.3 \\ 1 \times 0.2 + 2 \times 0.4 \end{bmatrix} = \begin{bmatrix} \mathbf{0.7} \\ \mathbf{1} \end{bmatrix}$$

$$z^{(1)} = \begin{bmatrix} 1 \\ \tanh(0.7) \\ \tanh(1) \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0.604} \\ \mathbf{0.762} \end{bmatrix}$$

$$u^{(2)} = 1 \times 0.2 + 0.604 \times 1 + 0.762 \times (-3) = \mathbf{-1.482}$$

$$z^{(2)} = \begin{bmatrix} 1 \\ \tanh(-1.482) \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \mathbf{-0.902} \end{bmatrix}$$

$$u^{(3)} = 1 \times 1 + (-0.902) \times 2 = -0.804$$

$$y^{(3)} = \tanh(-0.804) = \mathbf{-0.666}$$

(b)

※ **Derivative of** $\tanh(x)$ = $1 - \tanh^2(x)$

※ **Derivative of half of the sum square** $\mathrm{E}(x)$ = $x - y$

$$\delta^{(3)} = \frac{\partial E}{\partial u^{(3)}} = \frac{\partial E}{\partial y^{(3)}}\frac{\partial y^{(3)}}{\partial u^{(3)}} = \left(y^{(3)} - 1\right)\left(1 - \tanh^2\left(u^{(3)}\right)\right)$$
$$= (-0.666 - 1)(1 - \tanh^2(-0.8)) = \mathbf{-0.928}$$

$$\delta^{(2)} = \frac{\partial E}{\partial u^{(2)}} = \delta^{(3)} \times W_{21}^{(3)} \times \left(1 - \tanh^2\left(u^{(2)}\right)\right)$$
$$= (-0.928)(2)(1 - \tanh^2(-1.48)) = \mathbf{-0.348}$$

$$\delta^{(1)} = \frac{\partial E}{\partial u^{(1)}} = \begin{bmatrix} \delta^{(2)} \times W_{21}^{(2)} \times \left(1 - \tanh^2\left(u_{11}^{(1)}\right)\right) \\ \delta^{(2)} \times W_{31}^{(2)} \times \left(1 - \tanh^2\left(u_{21}^{(1)}\right)\right) \end{bmatrix}$$
$$= \begin{bmatrix} -0.348 \times 1 \times (1 - \tanh^2(0.7)) \\ -0.348 \times (-3) \times (1 - \tanh^2(1)) \end{bmatrix} = \begin{bmatrix} \mathbf{-0.221} \\ \mathbf{0.438} \end{bmatrix}$$

(c)

$$\frac{\partial E}{\partial W^{(1)}} = x^{(0)}\left(\delta^{(1)}\right)^T = \begin{bmatrix} 1 \\ 2 \end{bmatrix} [-0.221 \quad 0.438] = \begin{bmatrix} \mathbf{-0.221} & \mathbf{0.438} \\ \mathbf{-0.442} & \mathbf{0.876} \end{bmatrix}$$

$$\frac{\partial E}{\partial W^{(2)}} = z^{(1)}\left(\delta^{(2)}\right)^T = \begin{bmatrix} 1 \\ 0.604 \\ 0.762 \end{bmatrix} [-0.348] = \begin{bmatrix} \mathbf{-0.348} \\ \mathbf{-0.21} \\ \mathbf{-0.265} \end{bmatrix}$$

$$\frac{\partial E}{\partial W^{(3)}} = z^{(2)}(\delta^{(3)})^T = \begin{bmatrix} 1 \\ -0.902 \end{bmatrix} [-0.928] = \begin{bmatrix} \mathbf{-0.928} \\ \mathbf{0.837} \end{bmatrix}$$

(d)

※ $\boldsymbol{W \leftarrow W - \eta \nabla E(W)}$

$$W^{(1)} \leftarrow W^{(1)} - \eta \nabla E\left(W^{(1)}\right)$$

$$\Rightarrow \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix} - 0.5 \begin{bmatrix} -0.221 & 0.438 \\ -0.442 & 0.876 \end{bmatrix} = \begin{bmatrix} \mathbf{0.211} & \mathbf{-0.019} \\ \mathbf{0.521} & \mathbf{-0.038} \end{bmatrix}$$

$$W^{(2)} \leftarrow W^{(2)} - \eta \nabla E\left(W^{(2)}\right) \Rightarrow \begin{bmatrix} 0.2 \\ 1 \\ -3 \end{bmatrix} - 0.5 \begin{bmatrix} -0.348 \\ -0.21 \\ -0.265 \end{bmatrix} = \begin{bmatrix} \mathbf{0.374} \\ \mathbf{1.105} \\ \mathbf{-2.868} \end{bmatrix}$$

$$W^{(3)} \leftarrow W^{(3)} - \eta \nabla E\left(W^{(3)}\right) \Rightarrow \begin{bmatrix} 1 \\ 2 \end{bmatrix} - 0.5 \begin{bmatrix} -0.928 \\ 0.837 \end{bmatrix} = \begin{bmatrix} \mathbf{1.464} \\ \mathbf{1.5815} \end{bmatrix}$$

$$u^{(1)} = \begin{bmatrix} 1 \times 0.211 + 2 \times (0.521) \\ 1 \times (-0.019) + 2 \times (-0.038) \end{bmatrix} = \begin{bmatrix} \mathbf{1.253} \\ \mathbf{-0.1} \end{bmatrix}$$

$$z^{(1)} = \begin{bmatrix} 1 \\ \tanh(1.253) \\ \tanh(-0.1) \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0.849} \\ \mathbf{-0.1} \end{bmatrix}$$

$$u^{(2)} = 1 \times 0.374 + 0.849 \times 1.105 + (-0.1) \times (-2.868) = \mathbf{1.599}$$

$$z^{(2)} = \begin{bmatrix} 1 \\ \tanh(1.599) \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0.922} \end{bmatrix}$$

$$u^{(3)} = 1 \times 1.464 + (0.922) \times 1.5815 = 2.922$$

$$y^{(3)} = \tanh(2.922) = \mathbf{0.994}$$

2. Programming

(a)

```
from sklearn.metrics import mean_squared_error
y_val_pred = model.predict(X_val) # predict validation
print('MSE : ',mean_squared_error(y_val,y_val_pred)) # calculate MSE
```

```
MSE :  993.5947013815668
```

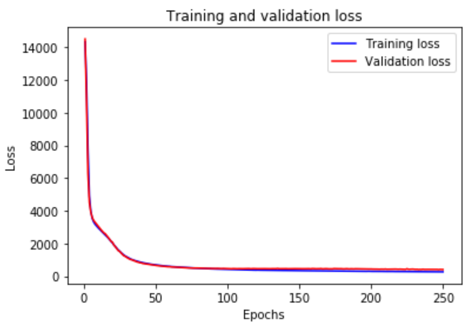
未經 StandardScaler 轉換

```
from sklearn.metrics import mean_squared_error
y_val_pred = model.predict(X_val) # predict validation
print('MSE : ',mean_squared_error(y_val_scaler,y_val_pred)) # calculate MSE
```

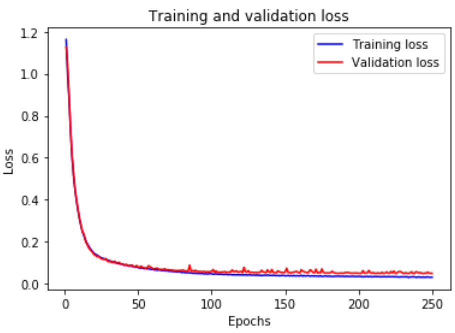
```
MSE :  0.09160529844185669
```

經 StandardScaler 轉換

(b) Plot the training history of Fully Connected (Dense) Hidden Layers



未經 StandardScaler 轉換



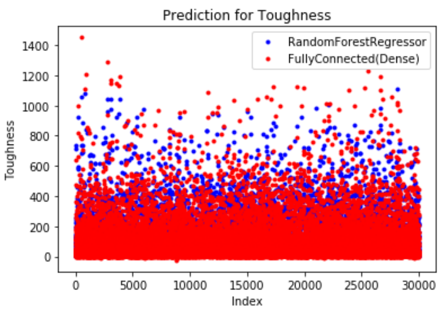
經 StandardScaler 轉換

(c) Essay

機器學習(Random Forest Regressor)是拿數據的特徵做模型的建置，而深度學習(Fully Connected (Dense) Hidden Layers)是利用數據去建置一個會自行判斷拿取特定特徵的模型。

將 label 數據做標準化後(StandardScaler)，數值成常態分佈而較容易訓練，因此可發現 loss 明顯下降且深度學習的 loss 較少。另外，可發現 RandomForestRegressor 的 loss 已約為最小值，而深度學習仍可以透過超參數的調整使 loss 更下降，但同時也可發現不管是用深度學習還是機器學習的 model 做訓練，其預測結果與分布皆相似，如下圖。

因此，我們可以歸納出若數據(回歸問題)較簡單時，可以使用機器學習去做預測，但較複雜的數據且須高準確性的話則是使用深度學習較適合。




```
import statistics

print("The mean of toughness with RandomForestRegressor = {:.5f}".format(statistics.mean(data_rfr.Toughness)))
print("The mean of toughness with FullyConnected(Dense) = {:.5f}\n".format(statistics.mean(data_fc.Toughness)))

print("The standard deviation of toughness with RandomForestRegressor = {:.5f}".format(statistics.stdev(data_rfr.Toughness)))
print("The standard deviation of toughness with FullyConnected(Dense) = {:.5f}".format(statistics.stdev(data_fc.Toughness)))
```

```
The mean of toughness with RandomForestRegressor = 66.39821
The mean of toughness with FullyConnected(Dense) = 64.96423

The standard deviation of toughness with RandomForestRegressor = 102.42544
The standard deviation of toughness with FullyConnected(Dense) = 103.92932
```