Practical 1: ANN Training and Prediction

Yushuo Chen
2020101918
MAM

October 24, 2023

Task1

In this task, we need to use perceptron weight learning algorithm then find 3 weights about *Study Hrs per Week, Sleep Hrs per Week, Quiz Marks* and a bias by given by data set. Set the bias input and 3 variable input to be w_0, w_1, w_2 and w_3. Set 3 variables to be $x^{(1)}$, $x^{(2)}$, $x^{(2)}$, and exam marks to be y. And we use sigmoid function as activation funtion which is $\sigma(x) = \frac{1}{1+e^{-x}}$, with its derivative $\sigma'(x) = \sigma(x) \cdot [1 - \sigma(x)]$. Moreover, set learning rate and initial weights to be

$$\eta = 0.1,$$
$$w_0 = 0.5,\ w_1 = w_2 = w_3 = 0$$

The most important thing is, we must normalize the data set.

1.1 Solution step by step

Using the formula:

$$for\ each\ x_i: \quad o(x_i) = \sigma(\sum_{j=0}^{3} w_j \cdot x_i^{(j)}),$$
$$err = (y_i - o(x_i)),$$
$$for\ each\ w_j: \quad w_j = w_j + \eta \cdot err \cdot \sigma'(\sum_{j=0}^{3} w_j \cdot x_i^{(j)}) \cdot x_i^{(j)}$$

Similar to calculation in slides, we can execute epoch 1.

Epoch 1:

- The derivative of 6 instances is $\sigma'(x) = [0.235, 0.244, 0.231, 0.234, 0.238, 0.237]$.
- Sigmoid value is $\sigma(x) = [0.622, 0.623, 0.635, 0.622, 0.609, 0.612]$.

Then for x_1:

$$\begin{aligned}
&o(x_1) = \sigma[0.5 + 0 \times (-0.201) + 0 \times (-0.096) + 0 \times 1.058] = 0.622,\\
&w_0 \leftarrow 0.5 + 0.1 \times (1 - 0.622) \times 0.235 = 0.509,\\
&w_1 \leftarrow 0 + 0.1 \times (1 - 0.622) \times 0.235 \times (-0.201) = -0.00179,\\
&w_2 \leftarrow 0 + 0.1 \times (1 - 0.622) \times 0.235 \times (-0.096) = -0.00085,\\
&w_3 \leftarrow 0 + 0.1 \times (1 - 0.622) \times 0.235 \times 1.058 = 0.00940.
\end{aligned}$$

Then for x_2:

$$\begin{aligned}
&o(x_1) = \sigma[0.509 + (-0.00179) \times (-0.856) + (-0.00085) \times 0.861 + 0.00940 \times (-0.442)] = 0.624,\\
&w_0 \leftarrow 0.509 + 0.1 \times (0 - 0.624) \times 0.234,\\
&w_1 \leftarrow (-0.00179) + 0.1 \times (0 - 0.624) \times 0.234 \times (-0.856),\\
&w_2 \leftarrow (-0.00085) + 0.1 \times (0 - 0.624) \times 0.234 \times 0.861,\\
&w_3 \leftarrow 0.0094 + 0.1 \times (0 - 0.624) \times 0.234 \times (-0.442).
\end{aligned}$$

Then we can repeat the process for the rest of 4 instances, and update w_i every time. After training by 6 examples, learning step goes to epoch 2. And again, repeat the process above for all learning set. When finishing 3 times of epoch, we get the final predictor model.
Predictor model is

$$0.446 + 0.126 \cdot x_i^{(1)} + (-0.058) \cdot x_i^{(2)} + 0.184 \cdot x_i^{(3)} = y_i.$$

Then we can calculate the MSE which is $MSE = 0.0823$.

1.2 Further Analysis

We can find that 4 input are not converged so that we cannot get exact or approximate results. More epochs we operate, more different results we get. This may because that data set is such a small set, and number of epochs are too small or learning rate is not suitable. If we want to get a more accurate result, we need to collect more training set and execute more epochs.

Task2

In this task, we need to build ANN and then predict churn rate of customer using it. According to trained model in the demo, we can call the tensorflow function directly and solve the problems.

2.1 Prediction for One Customer

Plug in the information of the customer, and then predict the leaving probability which is 3.92%. So it can be concluded that the customer will stay in with the bank.
With a threshold of 0.5, since

$$0.0392 < 0.5,$$

so the output will be 0.

2.2 Confusion Matrix and Accuracy

Because of the previous processing of split into test data set, predict for test set and get the confusion matrix and accuracy.

Confusion matrix	$\begin{pmatrix} 1614 & 81 \\ 200 & 205 \end{pmatrix}$
Accuracy	85%

Confusion matrix shows that about 1819 customers' real behaviour is the same as prediction. And about 81 customers' behaviour differs to our prediction. Accuracy indicates that the possibility to predict correctly for every prediction is 85%.

2.3 Further Analysis

I think that the normalization of data is important. That is because normalization could make the process input to be same scale, and this may improve the accuracy and generality of model. Also, normalization would eliminate the influence due to difference in data units. If we run model without normalization, some data in great scale may become the main factors which lead to wrong results.

Codes

Key codes for task1:

```
X=np.asarray([[24,56,78],[5,66,52],[92,42,82],[31,63,67],[15,70,44],[12,45,35]])
Y=np.asarray([1,0,1,1,0,0])
for epo in range(1,4):
    for m in range(0,6):
        ip[m,0]=w0+w1*X[m,0]+w2*X[m,1]+w3*X[m,2]
        g=1/(1+(math.exp(-ip[m,0])))
        g_d=g*(1-g)
        err[m,0]=Y[m]-g
        w0=w0+lr*err[m,0]*g_d
        w1=w1+lr*err[m,0]*g_d*X[m,0]
        w2=w2+lr*err[m,0]*g_d*X[m,1]
        w3=w3+lr*err[m,0]*g_d*X[m,2]
        m=m+1
    # print("Output for epoch%d is %.4f, %.4f, %.4f, %.4f" %(epo, w0, w1, w2, w3))
    epo=epo+1
```

Key codes for task2:

```
#predict the customer
customer_infor=np.array([[600, "France", "Male",40, 3, 60000, 2, 1, 1, 50000]])
customer_infor[:,2]=le.transform(customer_infor[:,2])
customer_infor=ct.transform(customer_infor)
customer_infor=sc.transform(customer_infor)
customer_pred=ann.predict(customer_infor)
#Predicting the Test set results with threshold 0.5
y_pred_r=ann.predict(X_test)
y_pred=(y_pred_r>0.5)
cm=confusion_matrix(y_test,y_pred)
accuracy=accuracy_score(y_test,y_pred)
accuracy=accuracy*100
if customer_pred > 0.5:
    print("The customer will leave the bank. The probability of leave about the
         customer is %f." %customer_pred)
else:
    print("The customer will stay in the bank. The probability of leave about the
         customer is %f." %customer_pred)
print(cm)
print("Accuracy equals to %d%%" %accuracy)
```