CSE-428

Image Processing

Assignment-II

Name : Udoy Saha

ID : 23341134

Section : 02

Submitted to :

Saiful Bari Iftu [SDQ]

Submission date :

5 - October - 2024

Ans to the ques no: 1

Dataset:

Living Area (feet^2) x_1	# Bedrooms x_2	Price ($\times 1000\$$) y
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	516

Let the prediction, $\hat{y} = m_1 x_1 + m_2 x_2 + c$

So, trainable parameters are m_1, m_2 and c.

Lets initialize with, $m_1 = 0$, $m_2 = 0$, $c = 0$

Let the loss function be MSE.

So, $\frac{\partial MSE}{\partial m_1} = \frac{1}{n} \sum_{i=1}^{n} x_1 \cdot (m_1 x_{1i} + m_2 x_{2i} + c - y_i)$

$= \frac{1}{5} \sum_{i=1}^{5} x_{1i} \times (-y_i)$ $[\because m_1 = m_2 = c = 0]$

$=$ ~~-862840~~ -826342

Similarly, $\frac{\partial MSE}{\partial m_2} = \frac{1}{n} \sum_{i=1}^{5} x_{2i} \times (-y_i)$

$= -1165$

And, $\frac{\partial MSE}{\partial c} = \frac{1}{n} \sum_{i=1}^{5} (-y_i)$

$= -369.4$

Let the learning rate be 10^{-6}.

So, after first iteration,

$$m_1 = 0 - 10^{-6} \times (-826342)$$
$$= 0.826$$

$$m_2 = 0 - 10^{-6} \times (-1165)$$
$$= 0.0012$$

$$c = 0 - 10^{-6} \times (-369.4)$$
$$= 0.00037$$

This way multiple iterations needed to be run and optimal values will be achieved eventually.

Till iteration 1, if a house features 4 bedrooms and 2200 $feet^2$ of living area, the house would cost

$$= 0.826 \times 2200 + 0.0012 \times 4 + 0.00037$$

$$= 1817.21 \ \$$$

(Ans)

More iterations will make the prediction more accurate.

—— o —— x —— o ——

Ans to the ques no:-2

Dataset:

Living Area (feet^2) x	Price Level	Encoded label y
1000	Low	0
1500	High	1
2000	High	1

Let the prediction, $\hat{y} = \dfrac{1}{1 + e^{-(m_1 x + c)}}$

Let's initialize, $m_1 = 0$, $c = 0$

And the cost function is binary-crossentropy.

First iteration ⇒

$\hat{y}$	y	Loss
0.5	0	0.693
0.5	1	0.693
0.5	1	0.693

So, mean of all errors $= \frac{1}{3}(0.693 + 0.693 + 0.693)$

$= 0.693$

Let the learning rate be 0.01

So, updating the weights,

$m_1 = m_1 - 0.01 \times \dfrac{\partial \text{ Error}}{\partial m_1}$

$= 0.00213$

$c = c - 0.01 \times \dfrac{\partial \text{ Error}}{\partial c}$

$= 0.00213$

More iterations needed. (Ans)

Ans to the ques no:-3

a

Yes there needs to be the following preprocessings:-

1. <u>Feature scaling</u>:- x_1, x_2 and x_3 dont lie on the same range. So, we need to make sure to normalize those data, so that the model is not biased towards any feature.

2. <u>Train-Test split</u>:- To identify the accuracy.

b

Here the model would be the following:-

$$y' = m_1x_1 + m_2x_2 + m_3x_3 + c$$

where; y' is the prediction, m_1, m_2, m_3 and c are learnable parameters.

The cost function would be:-

$$\frac{1}{2n} \cdot \sum_{i=1}^{n} (y_i - y'_i)^2$$

$$= \frac{1}{2n} \cdot \sum_{i=1}^{n} (y_i - m_1x_{1i} - m_2x_{2i} - m_3x_{3i} - c)^2$$

d (c)

For (i), the prediction would be,

$0 \times 10 + 0.1 \times 124 + 0.008 \times (-139) + 0.001$

$= 11.289$

For (ii), the prediction would be,

0

(c)

For (i), the cost from the cost function will be $= \frac{1}{3} \sum_{i=1}^{3} (y_i - y'_i)^2$

$= 6.868$

For (ii), the cost from the cost function will be $= 23.1397$

As the cost of (i) is lower, (i) is the better model.

$\boxed{d}$

From the input layer to each of the neurons of the hidden layer, the matrix equation would be :-

Let, $X = [x_1 \ x_2 \ x_3]^T$

$A(x)$ is the activation fuction

W is the weights

b is the bias

$\therefore$ Output of a neuron,

$$m = A(W^T X + b)$$

$\therefore$ M is the output from 4 neurons, $M = [m_1 \ m_2 \ m_3 \ m_4]^T$

The output layer will contain only 1 neuron, because of being a binary classification problem.

Let W_1 be the new set of weights and b_1 is the bias.

$$\therefore Y = \text{softmax}(W_1^T M + b)$$

Now based on the value of Y, there will be a theshold and finally the output.

— o — x — o —

Ans to the ques no: 4

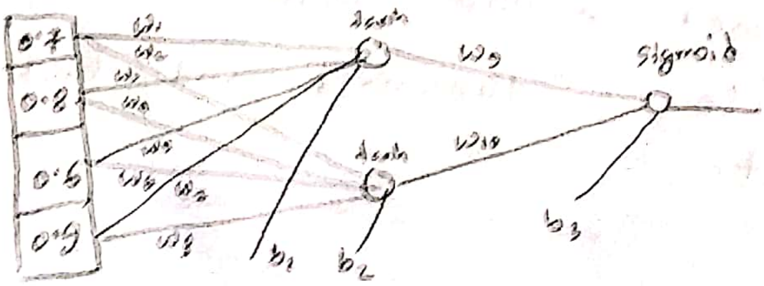
a

Activation functions are introduced to neural networks to implement non-linearity so that the model can be complex enough and capture complex relation among the features.

If we use linear activation function, there will be no need for neural networks, since two or more linear activation layers make the model nothing but linear. So traditional ML models and neural network will be equal then. That's why we avoid linear activation functions.

b

First, we have to flatten the image.
Here is the model architecture:



(a)

For the hidden layer,

$$output_{hidden} = \tanh\left(W_{input-hidden} \cdot X_{input} + b_{hidden} \right)$$

$$= \tanh\left(\begin{bmatrix} 0.1 & 0.5 & 0.3 & 0.1 \\ 0.2 & 0.4 & 0.09 & 0.3 \end{bmatrix} \begin{bmatrix} 0.7 \\ 0.8 \\ 0.6 \\ 0.9 \end{bmatrix} + \begin{bmatrix} -0.32 \\ -0.28 \end{bmatrix} \right)$$

$$= \tanh\left(\begin{bmatrix} 2.23 \\ 0.504 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0.97714 \\ 0.465257 \end{bmatrix}$$

For the output layer,

$$Output = \sigma\left(\begin{bmatrix} 0.5 & 0.9 \end{bmatrix} \begin{bmatrix} 0.97714 \\ 0.465257 \end{bmatrix} + \begin{bmatrix} 0.5 \end{bmatrix} \right)$$

$$= \sigma\left(\begin{bmatrix} 1.407 \end{bmatrix} \right)$$

$$= 0.8$$

Since, 0.8 is very close to 1, the output will be light (label: 1).

(Ans)

Ans to the ques no:-5

In scenario 1, lets choose a model with 2 hidden layers. For the data transformation, we may employ normalization of each feature.

The hyperparameters include the number of hidden layers, unit number in each, activation function, regularization technique and many more. Tuning these may improve performance.

In scenario 2, there is some data missing from the columns. We may either ~~drp~~ drop those rows, or fill it with somethings. Then we will check the heatmap of the dataset and exclude all the features that are not needed. Lastly the categorical variables need to be converted into a label.

After that correlation, bar chart etc will help understand the trend of the data. Also, data normalization is needed here.

A traditional decision tree based model would work fine in this case in my opinion.

— o — x — o —

Ans to the ques no:-6

In scenario 3, the test accuracy is ~~not~~ high. It is evident from the figure that the test loss is decreasing greatly. But the validation loss in increase over the epochs.

So, it ~~[illegible]~~ can be said that the model learned the training data too much and can not generalize. So, there is a overfit. Here regularizing techniques like dropout may address this issue.

In scenario 4, the training loss is increasing greatly over time. This clearly indicates underfitting. The model might not be complex enough. So, increasing the models complexity may solve this issue.

In scenario 5, the validation loss was decreasing and after a certain point, the validation loss is increasing. This means there is an overfitting. In this case, early stopping might be a good option. Also model complexity reduction will be another option.

In scenario 9, stochastic gradient descent might have been applied. The loss curve is fluctuating and this is a characteristic of SGD. To make the curve smoother, other optimization techniques like mini-batch gradient descent or ADAM optimizer can be used. The model is not necessarily a bad model. It may converge really soon, so seeing the curve it cant be said that the model was overfit or underfit. So, in my opinion the curve and model is ~~kind~~ quite good.

— · — ✕ — ○ —