2IIG0 Data mining and machine learning

Homework 2

Group 7
Ruichen Hu - 1674544
Simin Sun - 1692933
Mingzhe Shi - 1665073
Yasen Gao - 1723294

Declarations
All the team members worked on the homework individually and discussed the answers together.
We all contributed to the homework.

2021 - 2022

Contents

1 Question 1

The correct answer is C, which calculated by the code.

A. The prediction f(cars per hours, wind speed, wind direction) is most susceptible to noise in the feature wind direction.

Check the code and plots.

B.The predicted model suffers from overfitting.

When the degree is low, the model is not overfitting.

C.The MSE on the test set is 0.327 (rounded).

This is correct. See the code.

D.The value of the (rounded) intercept $\beta 0$ is 0.368.

β_0 is 1.629, $\beta 1$ is 0.368, so the answer D is incorrect.

2 Question 2

The correct answer is C

The MSR for different degrees are shown as Table 1

degree	dataset	MSR
2	train	0.298163391
	test	0.295737195
3	train	0.277615493
	test	0.351582655
4	train	0.265845469
	test	0.319130634
5	train	0.242390682
	test	0.596229594

Table 1: The MSR for different degrees

3 Question 3

The answer we get shown as below and check the implement for detailed computation:

$bias^2 = 4.0$

$variance = 0.2$

4 Question 4

The correct answer is B

The neutral network implemented with only linear units can only lead to a linear classification. That means, no matter how many layers and neurons there are, if we only use linear activations, the result from the output layer

can also be derived from the linear function of the first layer. Thus, it makes no sense whether we have one or multiple layers.

The plot explicitly illustrates that the data is not linearly separable. Therefore, we cannot use a linear activation function, and both option A, C are incorrect.

As it has been mentioned in the lecture that any continuous function can be arbitrarily well approximated by a single layer of neutral network with non-linear activations, we conclude that the data in this question can be classified by a MLP with one hidden layer and non-linear activation.

Thus, option D is wrong and we choose option B as the answer for this question.

Actually, we can verify our conclusions by applying MLPClassifier function of sklearn.neural_network. We generated the dataset manuelly and chose different actiation functions with varied layers to figure out how MLP works. The code included in the implement and the we attach the result in Figure 1

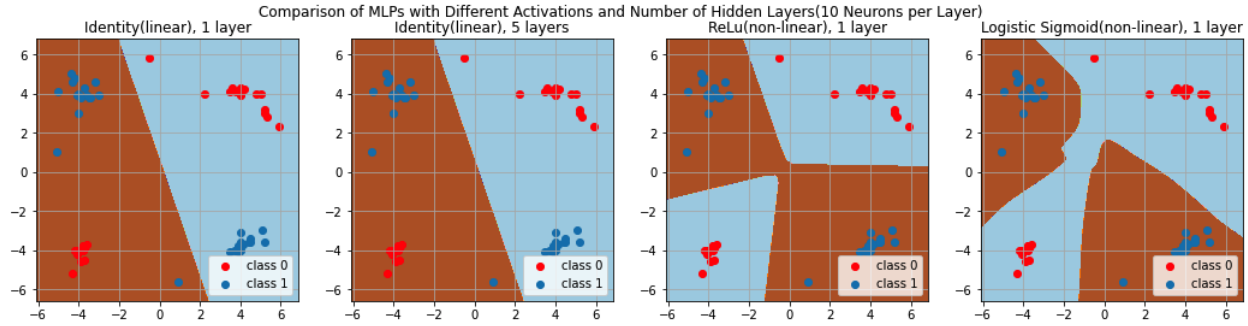

Figure 1: q4-implement

5 Question 5

The correct answer is C

A. The width of the kernel in layer S2 is 3.

> The kernel in S2 cannot be 3 as no padding is used. 2 × 2 kernel and stride 2 will give us exact 31 × 31 feature maps (62 - 2) / 2 + 1 = 31. If the kernel is 3 × 3, then the feature size will be floor((62 - 3) / 2) + 1 = 30, which is incorrect.

B. The number of the feature maps in layer S2 is 3.

> The number of feature layers in S2 is 6.

C. The number of parameters (weights, bias) in layer F3 is 23068.

> The number of parameters in layer F3 is (6 × 31 × 31 + 1)× 4 = 23068

D. The total number of network parameters (weights, bias) is 23136.

> The first layer: (3 * 3 * 1 + 1)*6 = 60
>
> The second layer: 0 (max pooling)
>
> The third layer: (6 * 31 * 31 + 1) * 4 = 23068
>
> The forth layer: 2 * 4 + 1 * 2 = 10
>
> Total : 23138

6 Question 6

We fit this data using MLP network. The network consists of 2 hidden layers. Each layer consists of 10 neurons.

We implement two MLP networks and both of them achieve accuracy of 97% . The difference between the two models is, the first networks(MLP I) works like the 2- layer model introduced in the lecture. The bias term treated like the extra input feature and controlled by the weight, as shown in Figure 2, we compute the value by $f([b:x]w)$. The second network(MLP II) we implement the bias as separate vector and we calculate the predicated value by $f(wx+b)$. Both networks trained using back-propagation and mini-batches Stochastic Gradient Descent (SGD).

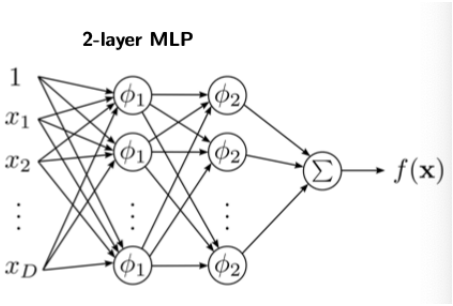

Figure 2: MLP I

6.1 b

(b) (1 points) Report the formula that you used in the report and discuss your choice. Include in your report the first 4 samples of the training data after transformation.

MLP I use the Min-Max normalization since we want our data range from 0 to 1.

MLP II use the Zero-min-unit-variance normalization.

The left is for MLP I and the right figure is MLP II:

```
after Min-Max normalization, first 4 sample
[[0.29148412 0.32829963]
 [0.18776047 0.07576615]
 [0.41972327 0.15359537]
 [0.81887582 0.98425828]]
```

```
Zero-Mean-Unit-Variance, first 4 sample:
[[-0.6924 -0.5403]
 [-1.0523 -1.3452]
 [-0.2475 -1.0972]
 [ 1.1374  1.5503]]
```

Figure 3: Answer for sub-question b

6.2 c

(c) (1 points) Determine the size of the input and output layers. Report your answer.

MLP I: input layer is 3, for m samples, it's m*3, output layer is 1,for m samples, it's m*1

MLP II: input layer is 2, for m samples, it's m*2, output layer is 1,for m samples, it's m*1

6.3 d

(d) (2 points) Choose a suitable loss function and activation function for the first and second hidden layers $\phi 1$, $\phi 2$. Report and justify your choice.

For both models, we use *Sigmoid* as activation function and *Binary Cross Entropy* as loss function.

If we plot the training data, we can find the two classes cannot be separated linearly. Thus, we need to choose a non-linear activation function. Here, we use *Sigmoid* for activation, because its output ranges from 0 to 1, which corresonds to our binary classification question that every target value is either 0 or 1. We did not consider *Softmax*, as it is mainly used for multi-label classification.

In terms of loss function, there are also many choices. After doing some research, we did not use *MSE* as the loss function, because the combination of *Sigmoid* and *corss entropy* causes learning slowdown when the final output value is nearly 1 or 0.[1]

More specifially, let L denote loss function(here, *MSE*), σ denote activation function(here, *Sigmoid*), a denote the result of activation function and z denote the value before activation at each neuron. Then, we calculate gradient descent:[1]

$$z = wx + b$$

$$\sigma = \frac{1}{1+e^{-z}}$$

$$L = \frac{(y-a)^2}{2}$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w} = (a-y)\sigma^{'}(z)x = a\sigma^{'}(z)$$

It explicitly shows that the gradient depends on the derivative of *Sigmoid*. When it converges to 1 or 0, the gradient is almost 0.

Instead of *MSE*, we use *cross entropy* method to calculate the loss. As there is only one output, we adapted a simplified version, i.e. binary cross entropy. The main reason of this selection is it can avoid learning slowdown. For binary cross entropy:[1]

$$L = yln(a) + (1-y)ln(1-a)$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w} = (\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)})\sigma^{'}(z)x = (\sigma(z)-y)x$$

This formula means the derivative of *Sigmoid* $\sigma^{'}(z)$ has no impact on the gradient this time. Therefore, even though the activations are close to 0 or 1, w and b can still be optimized.

6.4 e

(e) (2 points) Choose an initialization of the parameter values. Report and discuss your choice.

Initially, biases and weights in MLP are initialized both randomly by *Numpy np.random.randn* function to generate Gaussian distributions with mean 0 and standard deviation 1. However, this way of initialization is also a factor resulting in the instability of the model. In MLP II, the number of epochs when MLP reports a high accuracy varies a lot due to the uncertainty brought by this initialization.

To solve this problem, we then selected another method, which is an improvement based on *Numpy np.random.randn* function.

$$w = \frac{np.random.randn(m,n)}{np.sqrt(n))}$$

$$b = np.random.randn(m,1)$$

We use the above two formulas to generate the initial weights and biases with the dimension m x n and m x 1 respectively. The reason for this modification is that if we use the pure Gaussian distribution, the sum of $wx+b$ distributes broadly. As a result, the output from the last hidden layer can be to 1 or 0, and changing the value of weight or bias can hardly have impact on the performance of the model. By dividing the wights with a squared term, the distribution is more sharp and narrow. It leads to less probability of approaching to 0 or 1 at the last layer.[1]

6.5 g

(g) (10 points) Report the best values for each hyperparameter.

For MLP I, we try different combinations for several batch sizes, both models can achieve similar accuracy and only different in the learning rate and epochs, below Figure 4 we show the results of our first models, all results can been checked by running corresponding code blocks:

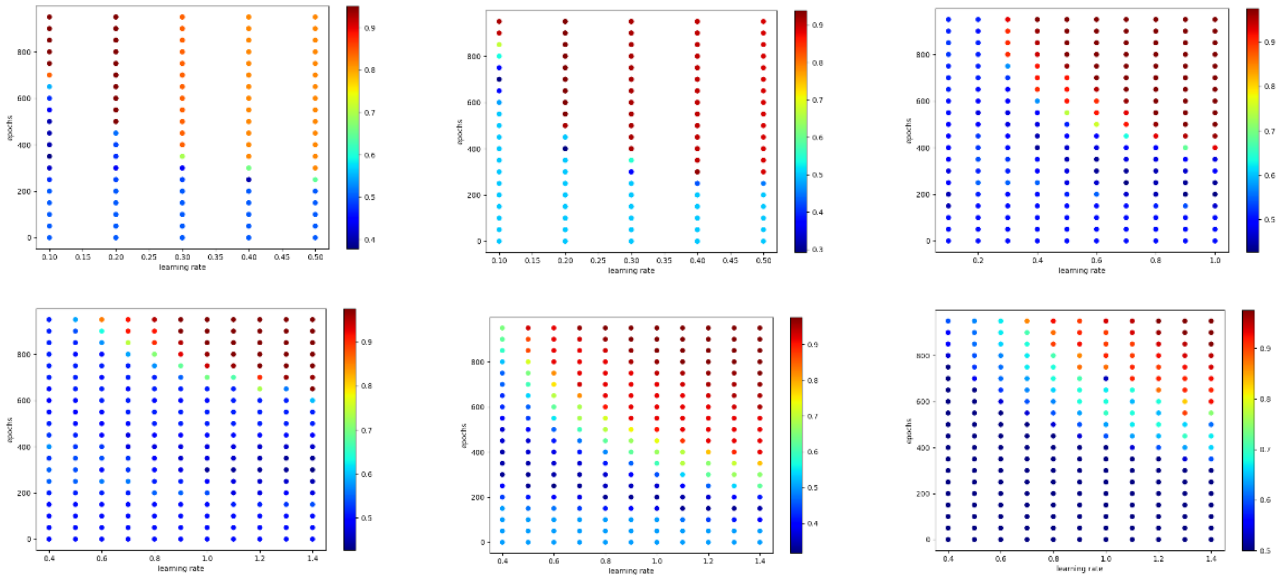

Figure 4: MLP I: Different hyper parameters based on batch sizes

The highest accuracy was obtained by the following combinations of learning rate and epochs Table 2. Note, for batch 11 and 21 we might can achieve the best accuracy if we try more combinations :

batch_size	learning rate	num_epochs	accuracy
11	0.1	750	0.9512195121951219
21	0.2	550	0.9390243902439024
41	0.4	950	0.975609756097561
82	0.9	950	0.975609756097561
140	0.8	950	0.975609756097561
210	1.2	950	0.975609756097561

Table 2: MLP I: Highest accuracy for each batch sizes

The hyperparameters in MLP II are mainly selected by trying different combinations of learning rate and batch size. The choose for number of epochs is related to the stopping criteria, which is discussed in question h. The best values for each parameters are listed in Table 3

batch_size	learning rate	num_epochs	training_accuracy	validation_accuracy
32	0.6	50	0.97560976	0.97560976

Table 3: MLP II: Highest accuracy for each batch sizes

6.6 h

(h) (5 points) Plot the loss computed over the training set and over the validation set. In addition, plot the classification accuracy computed over the two sets. Choose a stopping criteria for your training. State and justify your stopping criteria. Include your answer and the plots in your report.

The stopping criteria for MLP I is $num\_iterations > num\_epochs$ or $loss < allowed\_loss$. The loss computed over the training set shown as Figure 5

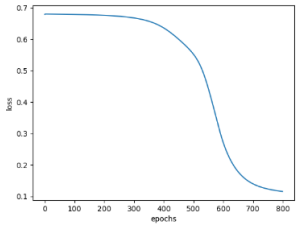

Figure 5: MLP I: loss change

As for MLP II, the learning is stopped when the number of epochs the model has taken reaches the threshold value. Specifically, we apply early stopping strategy.

The epoch number is chosen based mainly on the plots of correlation between number of epochs and accuracy, because the curve in the correlation between number of epochs and loss is relative smooth and decreases monotonically(Figure 6).

As the initial weights and biases are generated randomly, the number of epochs the model takes before accuracy stops increasing varies a bit, even though we take the improved way of initialization. According to Figure 6, we decide to choose 50 as the number of epochs for stopping. This guarantees that the accuracy of both training set and validation set exceed 0.9 for most initializations of weights and biases, and at the same time avoids overfitting

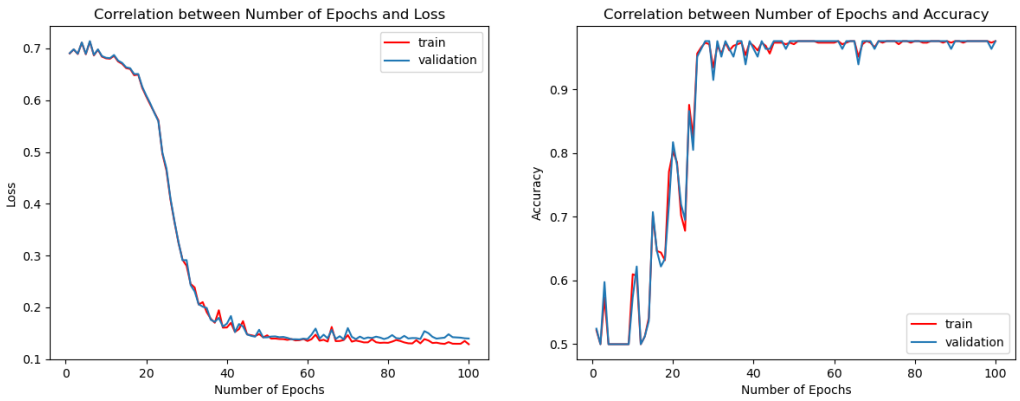

Figure 6: MLP II: Loss and classification accuracy over the training set and validation set with respect to epoch

6.7 i

(i) (5 points) Report the final accuracy obtained and the confusion matrix on the training and validation sets.

The final accuracy and the confusion matrix obtained by two MLPs are shown as Figure 7.

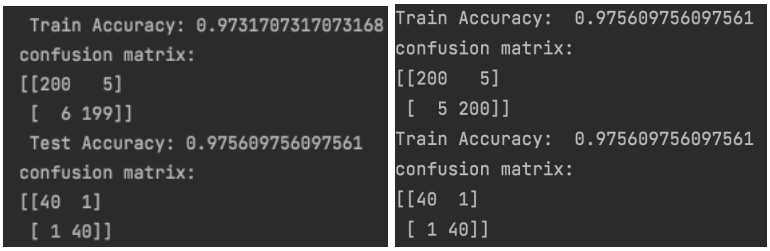

Figure 7: Accuracy and confusion matrix of MLP I(left) and MLP II(right)

References

[1] M.A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: `https://books.google.nl/books?id=STDBswEACAAJ`.