Name: Muhao Chen
IIT ID Number: A20456889

CS 584: Machine Learning

Homework 3

Theory Questions:

T_Q1:

Answer:

From 5.59, we can get the formula:

$$\tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} \quad (1)$$

And, through the question, we can get:

$$\sigma(a) = \{1 + \exp(-a)\} - 1 \quad (2)$$

From (1) and (2):

$$\frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} = -1 + 2 * (1/(1 + e^{-2a})) = 2\sigma(2a) - 1$$

While using the $\tanh(a)$, we can get the formula:

$$\begin{aligned} yvalue\_tanha &= \tanh(a^1) * w^1 + \tanh(a^2) * w^2 + w^0 \\ &= (2\sigma(2a^1) - 1) * w^1 + (2\sigma(2a^2) - 1) * w^2 + w^0 \\ &= 2 * w^1 * (2a^1) + 2 * w^2 * \sigma(2a^2) - (w^1 + w^2) + w^0 \end{aligned}$$

While using the $LR(a)$, we can get the formula:

$$yvalue\_lra = \sigma(a^1) * w^1 + \sigma(a^2) * w^2 + w^0$$

The question said that, $yvalue\_tanha = yvalue\_lra$, Therefore:

$$2 * w_h^k = w_{lr}^k \quad (3)$$

$$(w^1 + w^2) + w_h^0 = w_{lr}^0 \quad (4)$$

In conclusion: there exist two equivalent networks.

T_Q2:

While we analyze the soft-max layer, we can easily get the formula:

$$\frac{\partial y_{kn}}{\partial a_i} = y(\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} - y_{in})$$

And if we compute the derivative of the parameter:

$$\frac{\partial E(w)}{\partial a_i} = \frac{\partial}{\partial a_i}(-1) * (\sum_{n=1}^{N} \sum_{k=1}^{K} \ln y_k(x_{n,w}))$$

if we cut the derivative of the parameter, we can get:

$$(-1) * (\sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn} \frac{\partial y_{kn}}{\partial a_i} = y(\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} - y_{in}))$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn} y_{in} + (-1) * (\sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn} \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix})$$

$$= \sum_{n=1}^{N} y_{in} - \sum_{n=1}^{N} t_{in}$$

Finally, if k is equal to i, and matrix is 1, the sum of the t_kn is 1.

Name: Muhao Chen
IIT ID Number: A20456889

Q1_ CIFAR10:

Design Issues:

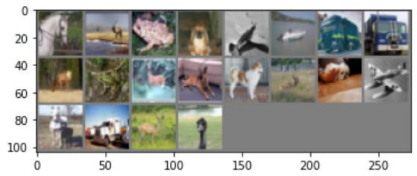
We have 50,000 pictures, with 3*32*32, and we want to use our algorithms to recognize the label of the pictures. It's a supervised machine learning issues. We can use fully connected neural network, and we also use CNN to compare whose performance is better.

Specific Problem:

1. How to use validation data set.
2. How to solve the large amount of data.
3. How to visualize the hidden layers.

A batch of outputs:

We convert the matrix data to the output.




```
the labels of one batch: ['horse', 'deer', 'frog', 'dog', 'plane', 'ship', 'truck', 'truck', 'deer', 'frog', 'deer', 'dog', 'dog', 'dee
r', 'dog', 'plane', 'horse', 'truck', 'deer', 'bird']
```

Q1_Fully_Connected_NN:

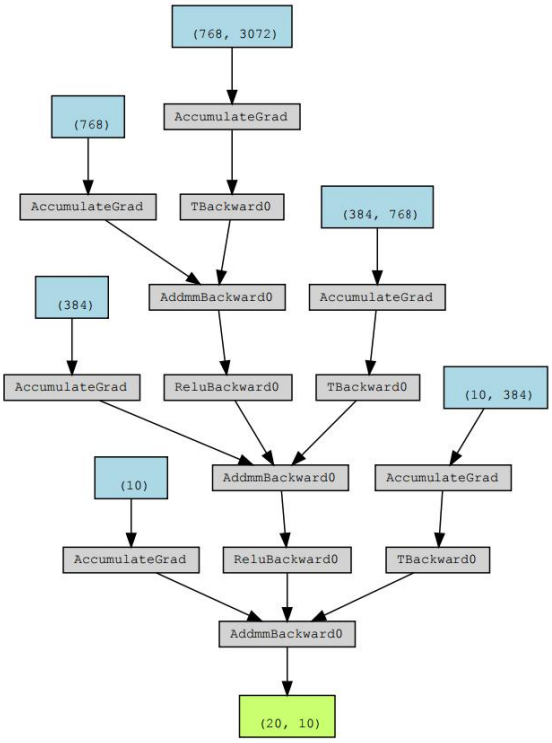
The neural network structure:

Firstly, we build a simple Neural Network, the structure is:

NN Model Information: NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=3072, out_features=768, bias=True)
    (1): ReLU()
    (2): Linear(in_features=768, out_features=384, bias=True)
    (3): ReLU()
    (4): Linear(in_features=384, out_features=10, bias=True)
)
)

Secondly, we visualize the structure:

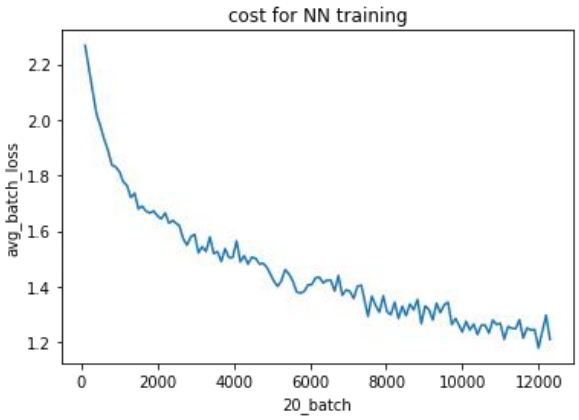
Name: Muhao Chen
IIT ID Number: A20456889



Training:

Because it takes too many time to train, so I set the epoch to 5, and each 100 times computation, I will record the loss and accuracy of training and validation.

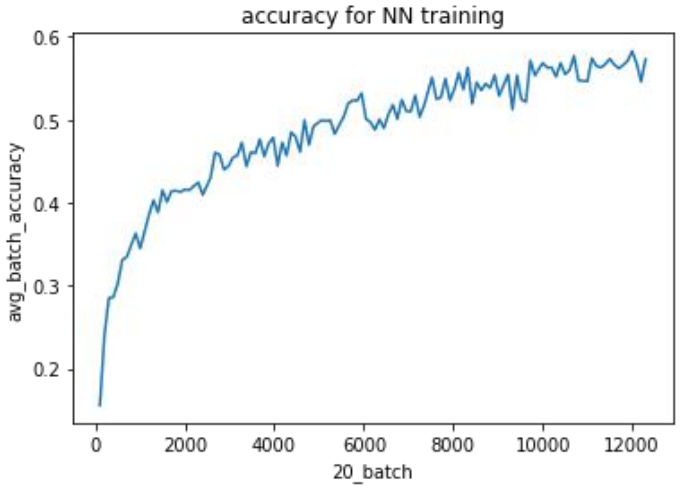
The cost of fully connected neural network while training:



It seems that it have significant effects.

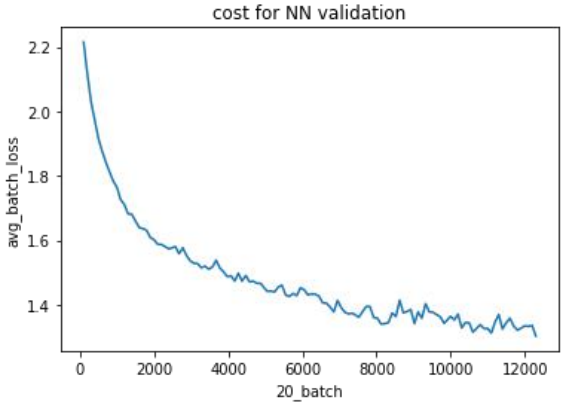
The accuracy of fully connected neural network while training:

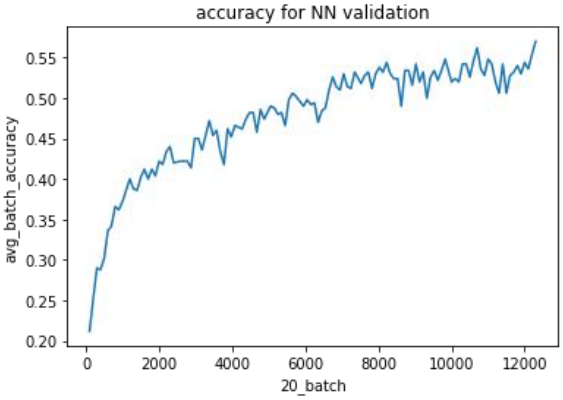
Name: Muhao Chen
IIT ID Number: A20456889



Initially, it ha a rapid improvement in the accuracy, and after 8000-20-batchs, slowly to improve the accuracy.

Validation:





Unfortunately, it dose not have a good effect as the training data. But it also have more than 50% accuracy.

Testing:

Accuracy of the network on the test images: 52 %, which could be better I consider. And we want to figure out which class have a more accurate prediction.

Therefore, we get:

Accuracy for class plane is: 57.3 %
Accuracy for class car is: 66.4 %
Accuracy for class bird is: 39.9 %
Accuracy for class cat is: 38.3 %
Accuracy for class deer is: 31.6 %
Accuracy for class dog is: 42.7 %
Accuracy for class frog is: 68.7 %
Accuracy for class horse is: 61.9 %
Accuracy for class ship is: 67.1 %

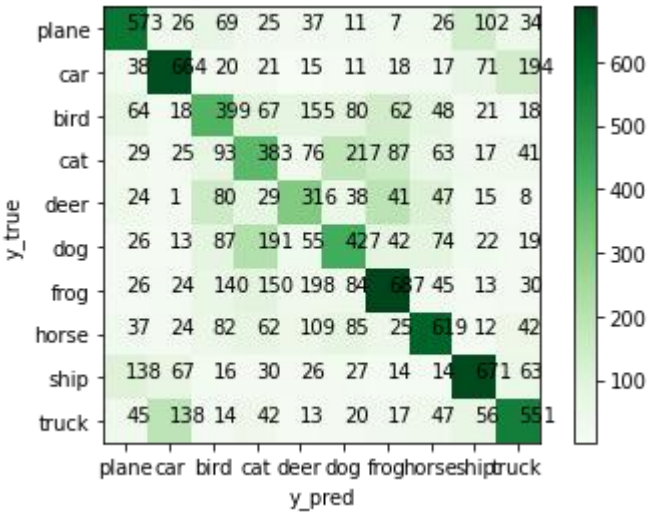
Name: Muhao Chen
IIT ID Number: A20456889

Accuracy for class truck is: 55.1 %

It's obvious that frog and car have a significant accuracy, and perhaps the style and color of these two classes are extremely different from the others.

And we use confusion matrix to see the result:

	precision	recall	f1-score	support
plane	0.63	0.57	0.60	1000
car	0.62	0.66	0.64	1000
bird	0.43	0.40	0.41	1000
cat	0.37	0.38	0.38	1000
deer	0.53	0.32	0.40	1000
dog	0.45	0.43	0.44	1000
frog	0.49	0.69	0.57	1000
horse	0.56	0.62	0.59	1000
ship	0.63	0.67	0.65	1000
truck	0.58	0.55	0.57	1000
accuracy			0.53	10000
macro avg	0.53	0.53	0.52	10000
weighted avg	0.53	0.53	0.52	10000



Q1_Fully_Connected_NN_Improved:

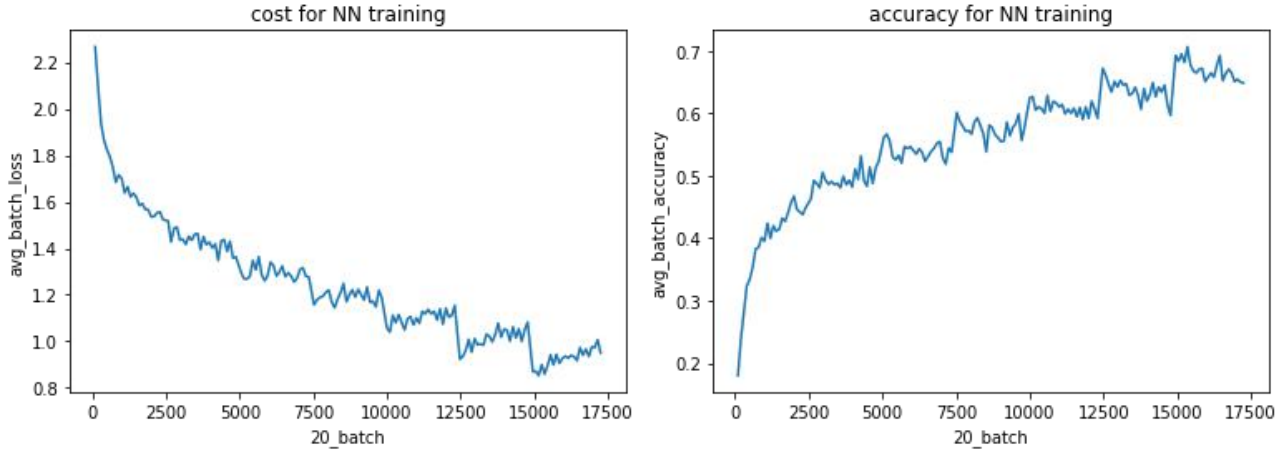
And we think whether we could use more deeper and different hyper parameters to make a better result. Therefore, we add a new layer:

NN Model Information: NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=3072, out_features=768, bias=True)
    (1): ReLU()
    (2): Linear(in_features=768, out_features=768, bias=True)
    (3): ReLU()
    (4): Linear(in_features=768, out_features=384, bias=True)
    (5): ReLU()
    (6): Linear(in_features=384, out_features=10, bias=True)
)
)

Name: Muhao Chen
IIT ID Number: A20456889

And change the learning rate from 0.001 to 0.005, which help us to learn more quicker, which solve the problem of under-fitting.

And it's obvious, on the training data, there are some steep line, which shows that the model learn something important.



Test:

Finally, Accuracy of the network on the test images: 55 %, which is better than the simple NN. And we see the accuracy of different classes:

Accuracy for class plane is: 69.5 %
Accuracy for class car is: 64.3 %
Accuracy for class bird is: 44.3 %
Accuracy for class cat is: 44.3 %
Accuracy for class deer is: 38.9 %
Accuracy for class dog is: 46.3 %
Accuracy for class frog is: 58.9 %
Accuracy for class horse is: 62.1 %
Accuracy for class ship is: 67.8 %
Accuracy for class truck is: 54.7 %

The images label plane or ship have a significant improvement, which means, the hidden layers learn something about the transportation.

Q1_CNN_Improved:

And we use the most popular model in the area of image recognization:

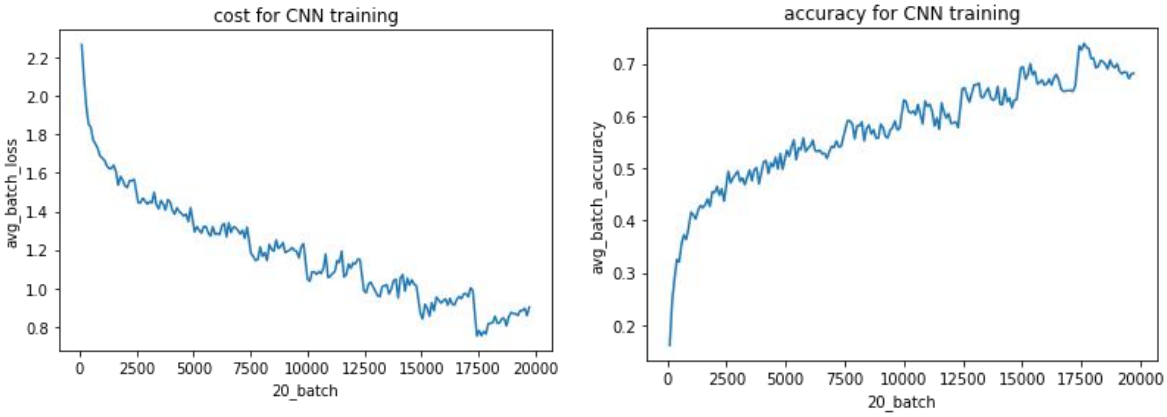
CNN Model Information: Net(
(features): Sequential(
(0): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
(1): ReLU(inplace=True)
(2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(3): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
(4): ReLU(inplace=True)
(5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)

Name: Muhao Chen
IIT ID Number: A20456889

```
  (classifier): Sequential(
    (0): Linear(in_features=400, out_features=120, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=120, out_features=84, bias=True)
    (4): ReLU(inplace=True)
    (5): Linear(in_features=84, out_features=10, bias=True)
  )
)
```

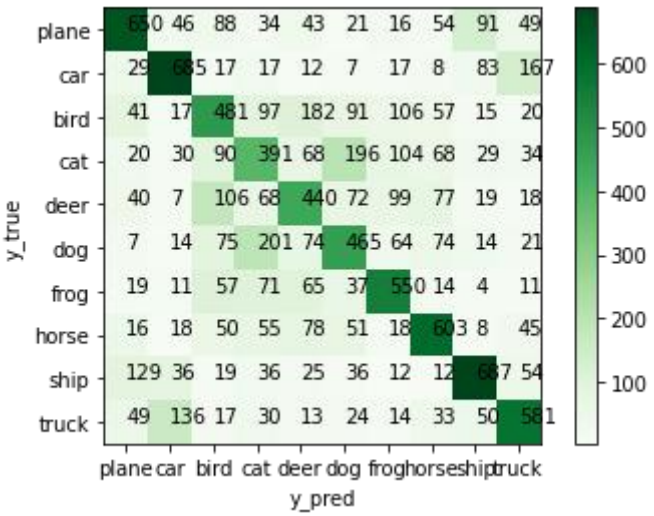
Training:

It's obvious that the CNN have learn something step by step.



Testing:

Accuracy of the network on the test images: 55 %. Which is same to the complex NN model, and actually, I only design a simple CNN, so there is a large space to improve the model.



Name: Muhao Chen
IIT ID Number: A20456889

Q2_ spam:

Design Issues:

We have 4,601 mails, with 57 features, and we want to classify whether the mail is spam. It's seems easy, and in early era, data scientist often used the NB algorithms. And today we use the NN model to analyze.

Specific Problem:

1. Normalize the data.
2. Put into the tensor, which is different from the CIFAR10.

Prepare Data:

Normalize:

we use the max - min to get the normalized data.

data = data.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))

Then, we put the data into the tensor.

```
# train loader
batch_size = 10
train_target = torch.tensor(data_train[57].values.astype(int))
train_target = train_target.type(torch.LongTensor)
train = torch.tensor(data_train.drop(57, axis = 1).values.astype(np.float32))
train_tensor = data_utils.TensorDataset(train, train_target)
train_loader = data_utils.DataLoader(dataset = train_tensor, batch_size = batch_size, shuffle = False)
# validation loader
vali_target = torch.tensor(data_vali[57].values.astype(int))
vali_target = vali_target.type(torch.LongTensor)
vali = torch.tensor(data_vali.drop(57, axis = 1).values.astype(np.float32))
vali_tensor = data_utils.TensorDataset(vali, vali_target)
vali_loader = data_utils.DataLoader(dataset = vali_tensor, batch_size = batch_size, shuffle = False)
# test loader
test_target = torch.tensor(data_test[57].values.astype(int))
test_target = test_target.type(torch.LongTensor)
test = torch.tensor(data_test.drop(57, axis = 1).values.astype(np.float32))
test_tensor = data_utils.TensorDataset(test, test_target)
test_loader = data_utils.DataLoader(dataset = test_tensor, batch_size = batch_size, shuffle = False)
```

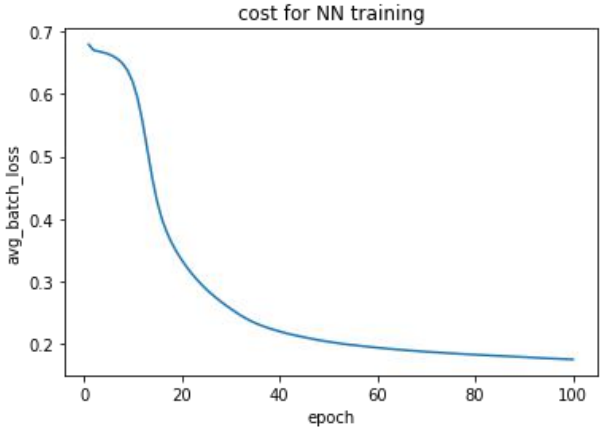
Training:

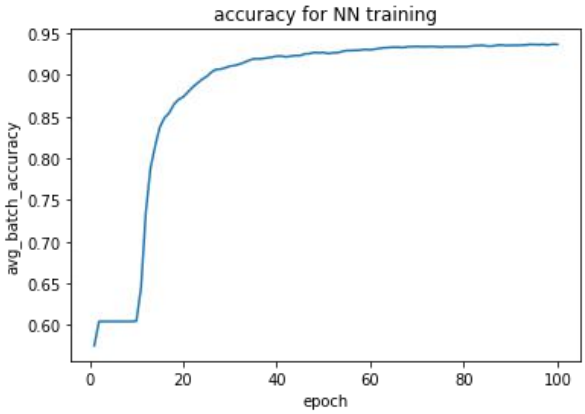
The structure of the NN, because we should think this problem is easy, not put so many hidden layers into the nn:

```
Model Information: NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=57, out_features=114, bias=True)
    (1): ReLU()
    (2): Linear(in_features=114, out_features=114, bias=True)
    (3): ReLU()
    (4): Linear(in_features=114, out_features=2, bias=True)
  )
)
```

Name: Muhao Chen
IIT ID Number: A20456889

It dose have a good performance on the training data set.





Test:

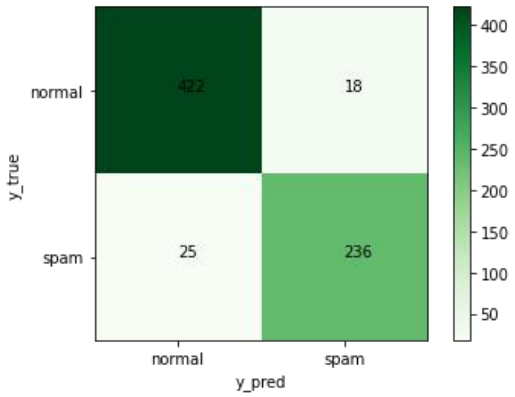
Accuracy of the network on the test images: 93 %.

Accuracy for class normal is: 94.4 %

Accuracy for class spam is: 92.9 %

Conclusion: we have a good performance to choose who is the spam email. And it's more likely easy to select the normal emails.

	precision	recall	f1-score	support
normal	0.96	0.94	0.95	447
spam	0.90	0.93	0.92	254
accuracy			0.94	701
macro avg	0.93	0.94	0.93	701
weighted avg	0.94	0.94	0.94	701



However, there is a problem, the precision of spam is only 90%, which means it has a bad performance while recognizing the spam email. And we always use ROC to judge the performance, these data have a bias of normal emails.

Name: Muhao Chen
IIT ID Number: A20456889

Q3_ CRIME:

Design Issues:

We have 1994 cases, with 127 features, and we want to classify whether it have a high probability to make crime. It's seems difficult, because there are some null data, and it only have 1994 cases, which have more than 127 features.

Specific Problem:

1. How to solve the over-fitting.
2. How to wash the data.
3. How to k-fold validation.

Prepare the data:

Firstly, we use "0" to replace all the null data.

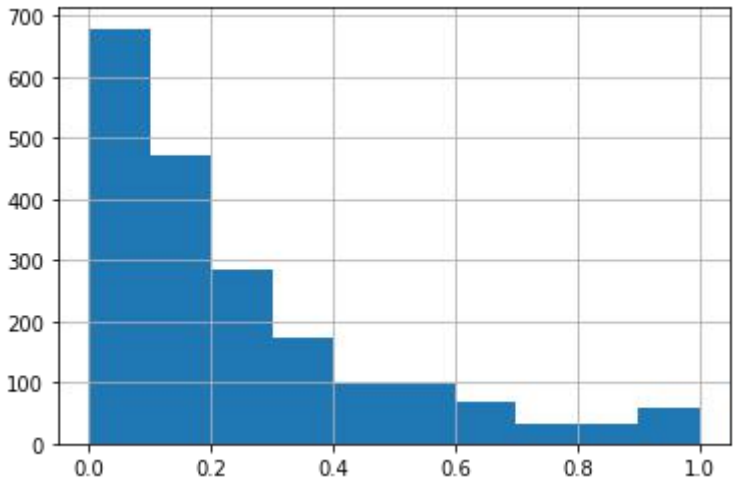
Secondly, we turn the class features, like the Lakewoodcity, Tukwilacity and so on. We use virtual variable to turn to them to the numeric value.

```
# fill with the 0 value
df = df.replace("?",0)
# trun to the np.float32 if possible
for col in df.columns:
    try:
        df[col] = df[col].astype(np.float32)
    except:
        continue
df_feature = df.iloc[:,:-1]
df_class = df.iloc[:,-1:]
tmp_list = df_feature[3].unique().tolist()
# generate the virtual variable
df_feature[3] = df.apply(lambda x:virtual(x[3],tmp_list),axis = 1)
df_feature
```

Label the Instance:

Because we want to classify, so we turn the numeric value to the int.

And at first, we should see the distribution of y.



Then ,we define the division function:

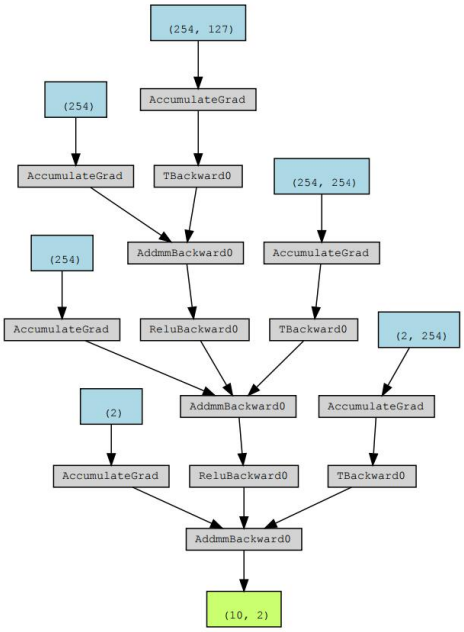
```
def y_classify_2(series):
    if   series > 0.15:
        return 1
    else:
        return 0
```

Name: Muhao Chen
IIT ID Number: A20456889

NN structure:

Because it's easy to get over-fitting, therefore, we design a simple NN model.

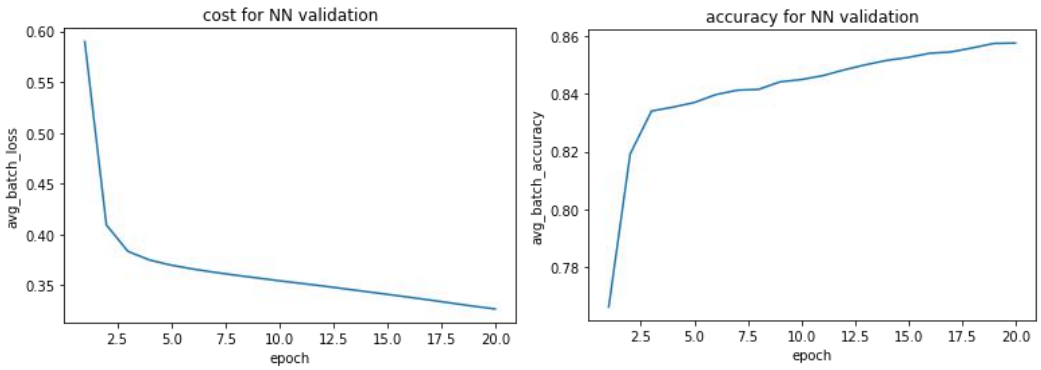
Model Information: NeuralNetwork(
(flatten): Flatten(start_dim=1, end_dim=-1)
(linear_relu_stack): Sequential(
(0): Linear(in_features=127, out_features=254, bias=True)
(1): ReLU()
(2): Linear(in_features=254, out_features=254, bias=True)
(3): ReLU()
(4): Linear(in_features=254, out_features=2, bias=True)
)
)



Training and Validation:

We use the 5-fold validation, through generating 5 loaders.

The result is :



Name: Muhao Chen
IIT ID Number: A20456889

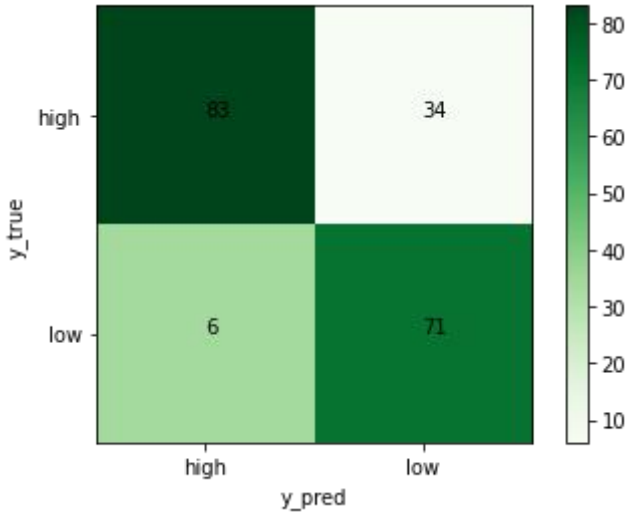
Testing:

Accuracy of the network on the test images: 79 %

Accuracy for class high is: 93.3 %

Accuracy for class low is: 67.6 %

	precision	recall	f1-score	support
high	0.71	0.93	0.81	89
low	0.92	0.68	0.78	105
accuracy			0.79	194
macro avg	0.82	0.80	0.79	194
weighted avg	0.82	0.79	0.79	194



Conclusion:

It's obvious that it has a good performance, which is 79%. And it's more accurate if the label is "high". We can add more test data to make more balance.