Machine Learning Report of Proj3

Haorui Dong
UBIT number: 50291149
Department of Computer Science and Engineering
University at Buffalo
haoruido@buffalo.edu

Abstract

In this project, we learn about classification. We have lots of digit image with a digit number in the middle of these image. We need use four classifiers to do classification. And at last, we need combine the result of these four classifiers to have a final result. We will use logistic regression, SVM, neural network and random forest. I implement logistic regression with python. Using sklearn to finish SVM and random forest. And using keras to set up a neural network.

1 Introduction of MNIST and USPS Data

1.1 MNIST

MNIST is a data set for a large database of handwritten digits that is commonly used for training various image processing systems. It often used for new hand to machine learning like us. The data contains black and white digitals in a 28x28 pixel from 0 to 9. We use part of them to train out model and a little part to do test. Because there are 10 sorts(0 to 9), this is a more than two classification problem. Thus we need use softmax instead of logistic regression.

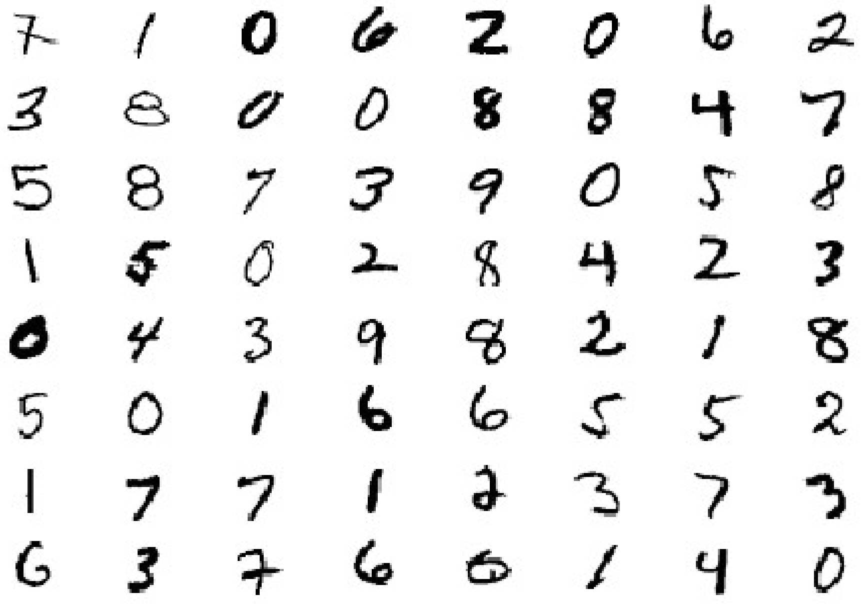


Figure 1: A random selection of MNIST digits

In this project, the teacher offers us an archive file of MNIST. The offed code has already get the data of train, validation and test. It is convenient for us use them.

1.2 USPS data

In this project, we just use USPS data to test our model which trained by MNIST data.

Figure 2: Sample image from USPS

2 Introduction of logistic regression with softmax

2.1 Logistic regression

First, we talk about logistic regression function:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Figure 3**:** Sigmoid Function

This function is activation function of logistic regression. This function maps real numbers to the [0,1] interval.

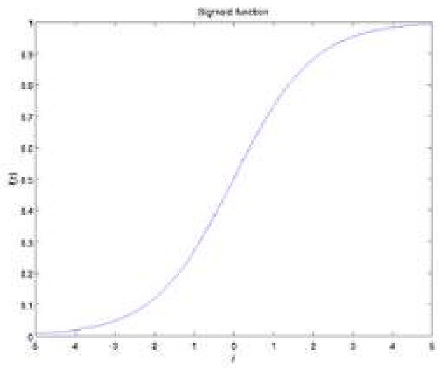


Figure 4: Curve of Sigmoid Function

Logistic regression with sigmoid function could do two classification problem. But we need more classifications, so we use softmax function.

2.2 Softmax regression

To solve more than two classification problem, we use softmax. The function of softmax is:

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1|x^{(i)};\theta) \\ p(y^{(i)} = 2|x^{(i)};\theta) \\ \vdots \\ p(y^{(i)} = k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

Figure 5: Softmax Function

And the loss of this function is:

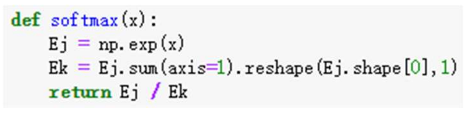
$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

Figure 6: Loss function of Softmax

We use gradient descent to train our regression model.

2.3 Implement softmax with python

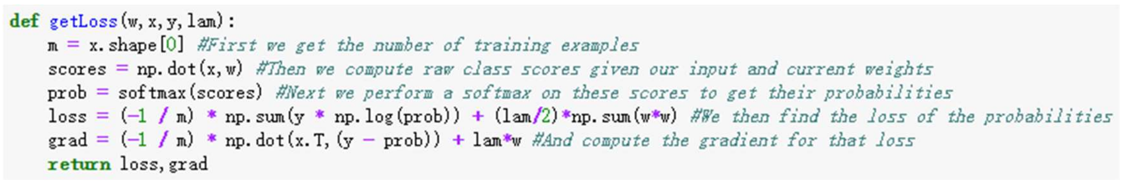
First, I implement the softmax function:



```
def softmax(x):
    Ej = np.exp(x)
    Ek = Ej.sum(axis=1).reshape(Ej.shape[0],1)
    return Ej / Ek
```

This function will return the softmax result of input. And will be used in other functions.

Then I implement the loss function and also gradient.



```
def getLoss(w,x,y,lam):
    m = x.shape[0] #First we get the number of training examples
    scores = np.dot(x,w) #Then we compute raw class scores given our input and current weights
    prob = softmax(scores) #Next we perform a softmax on these scores to get their probabilities
    loss = (-1 / m) * np.sum(y * np.log(prob)) + (lam/2)*np.sum(w*w) #We then find the loss of the probabilities
    grad = (-1 / m) * np.dot(x.T,(y - prob)) + lam*w #And compute the gradient for that loss
    return loss,grad
```

This function return the loss and gradient of the input.

The first input w is shape of weight. The second input x is training data. The third input y is training label after one-hot encode. And the last input is lambda.

2.4 The Result of Softmax

During training, the curve of training cost is:

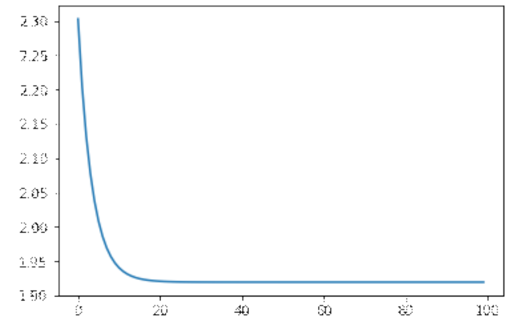


Figure 7: Cost

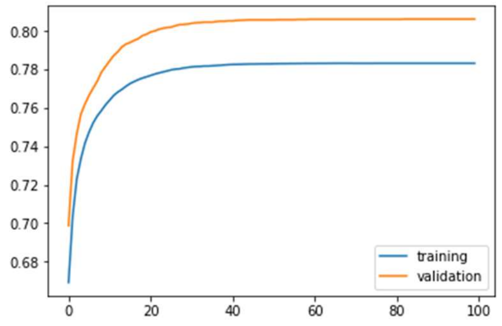


Figure 8: Accuracy

And I print the figure of the weight after training, for example, I print the ‘8’ and ‘4’’s weight:

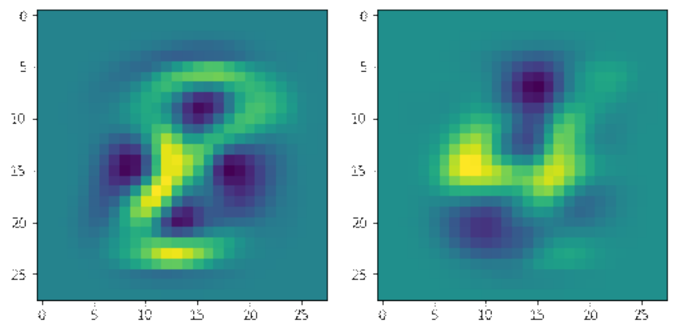
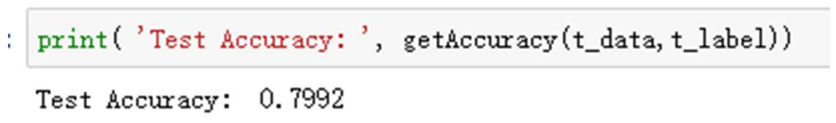


Figure 9: Image of weight

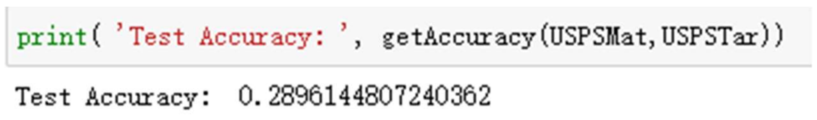
The accuracy of test data is 0.7992. To compare, I use sklearn to set up a model of logistic regression with softmax. And that get an accuracy: 0.9199. It is obviously higher than my classifier. I think there may be some function is not very precise to be set up by my code. And these cause the lower accuracy.



```
print('Test Accuracy: ', getAccuracy(t_data,t_label))

Test Accuracy:  0.7992
```

Then I test the USPS data, it just got 0.2896 of accuracy.



```
print('Test Accuracy: ', getAccuracy(USPSMat,USPSTar))

Test Accuracy:  0.2896144807240362
```

3 Neural network

In this project, we allowed to use package from Keras. So I use Keras to set up my neural network.

First, I choose the model named “Sequential”. Then I add two layers to my model. The first layer has 32 units and the activation function is sigmoid. The input shape is (748,). Then the second layer has 10 units and its activation function is softmax.

Then, before training, I need do compile method to my model. It will set the optimizer, loss function and metrics. I use Stochastic gradient descent(SGD) as my optimizer and “categorical crossentropy” as my loss function. And metrics usually will be set as ‘accuracy’.

After these, we can train our model with fit method. I set batch size as 128 and epochs as 50.

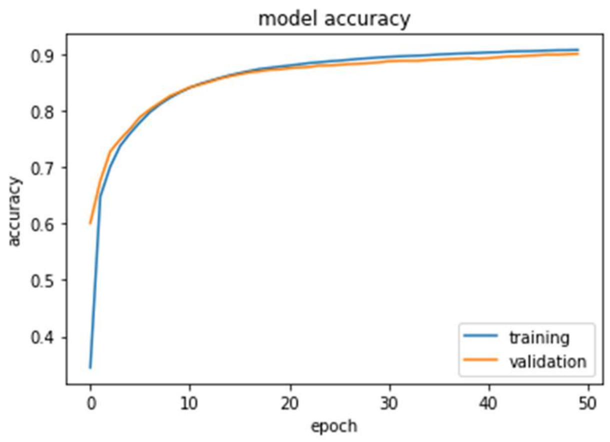
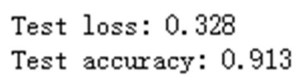


Figure 10: Accuracy of Training and Validation

And the test loss is 0.328, accuracy of test is 0.913.



```
Test loss: 0.328
Test accuracy: 0.913
```

Then I change some training parameters. First, I change the loss function of my model, I use ‘mean_squared_error’. And I set the batch size as 100. At beginning, I set the epochs as 50, but it does not reach the final result of the model. So I set it be 100, 200, 400. I find when it is 400, it almost reach the highest result.

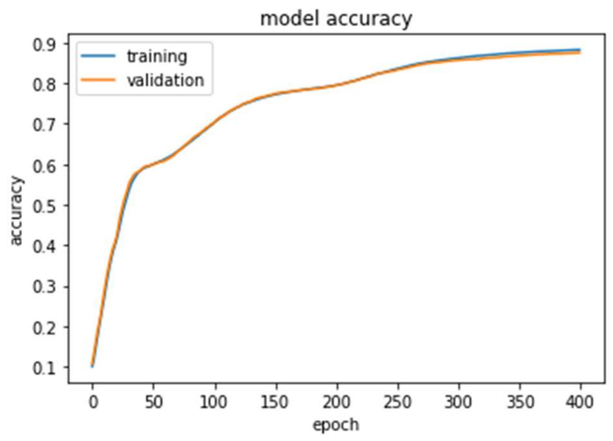
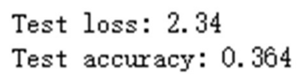


Figure 11: Accuracy of Training and Validation

The final loss is 0.0204 and accuracy is 0.893. It is little lower than the first model. But still got a worse model than I expected. Because TA said the neural network could have 0.98+ accuracy.

Then I test the data from USPS. The loss is 2.34, and the accuracy is just 0.364.



```
Test loss: 2.34
Test accuracy: 0.364
```

The neural network get better performance than logistic regression.

4 Support Vector Machine(SVM)

4.1 Mathematical Derivation of SVM

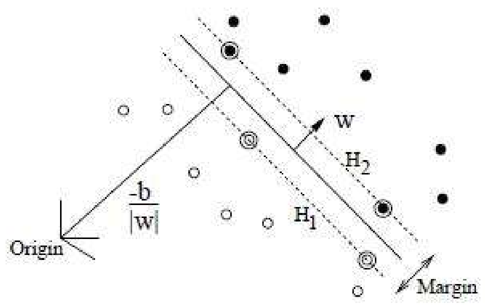


Figure 11: SVM

First, there are some training data, we set them as $\{x_i, y_i\}, \quad i = 1, \cdots, l, \quad y_i \in \{-1, 1\}, x_i \in R^d$

And we set the boundary as $w \cdot x + b = 0$, then we can do classification to these data. And two support vectors H1 and H2 as:

$$w \cdot x + b = 1$$
$$w \cdot x + b = -1,$$

Then all the data should be outside of H1 and H2, so the function transfer to:

$$w \cdot x_i + b \geq 1 \quad for \quad y_i = 1$$
$$w \cdot x_i + b \leq -1 \quad for \quad y_i = -1$$

And the margin is:

$$Margin = 2/\|w\|$$

So what we need to do is find the minimum of ||w||,

$$\begin{cases} \quad \min \|w\|^2/2 \\ s.t. \quad y_i\left(w \cdot x_i + b\right) - 1 \geq 0 \end{cases}$$

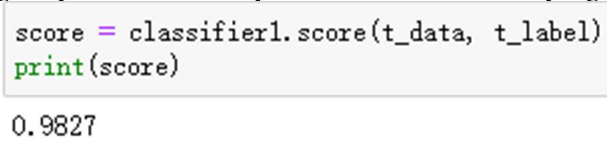
Then we will use Lagrangian theorem, actually I have not totally understand the following part of calculation, it’s very complex. And the final result of calculation is:

$$\begin{aligned} f(x) &= w^T x + b \\ &= \sum_{i=1}^{m} \alpha_i y_i x_i{}^T + b \end{aligned}$$

We use the SMO algorithm to find α and use support vector to find β.

4.2 Implement SVM with sklearn

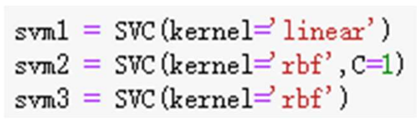
Sklearn is a very convenient package for machine learning. I just import SVC from sklearn.svm and the set my model as SVC(kernel=’rbf’,c=2,gamma=0.05), then it take a very very long time to train my model. Finally, I got my result as accuracy=0.9827. It has a obviously high performance.



```
score = classifier1.score(t_data, t_label)
print(score)
```

```
0.9827
```

Then I following the description of project, change some parameters of my model:



```
svm1 = SVC(kernel=’linear’)
svm2 = SVC(kernel=’rbf’,C=1)
svm3 = SVC(kernel=’rbf’)
```

Then these three model do not reach the accuracy of the first one. And the last two has the same accuracy.

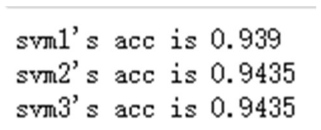
I find the manual of SVM in sklearn to learn the parameters.

Kernel is a core funcion, it could be ‘rbf’, ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’.

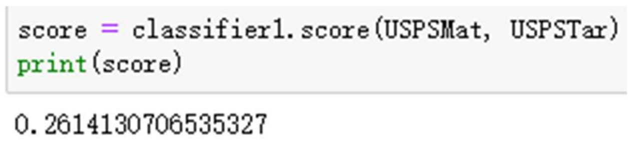
Gamma is parameters which works when kernel is ‘rbf’,’poly’,’sigmoid’, default is ‘auto’.

C is penalty parameter when C become higher, the loss will more influence the model.

The final accuracy of these three models is:

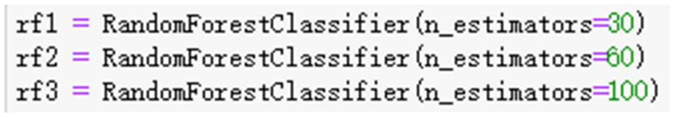

```
svm1's acc is 0.939
svm2's acc is 0.9435
svm3's acc is 0.9435
```

Then I test the data from USPS, it just got accuracy as 0.261.


```
score = classifier1.score(USPSMat, USPSTar)
print(score)
```
```
0.2614130706535327
```

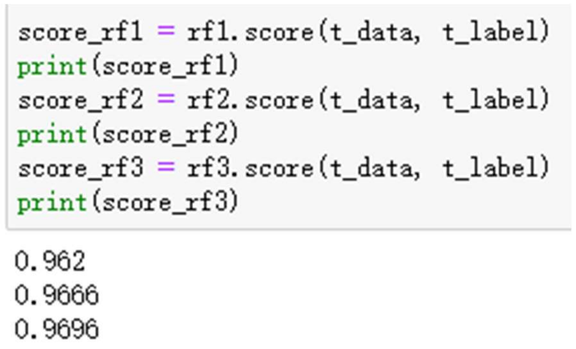
5 Random Forest

I used sklearn package to implement random forest. Following description of project I modify the tree number of my models.

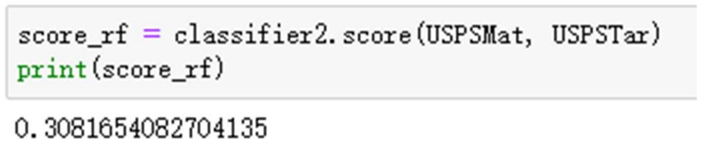

```
rf1 = RandomForestClassifier(n_estimators=30)
rf2 = RandomForestClassifier(n_estimators=60)
rf3 = RandomForestClassifier(n_estimators=100)
```

After training, the higher tree number get the higher accuracy, but their difference is not particularly big.

They are 0.962, 0.9666, 0.9696.


```
score_rf1 = rf1.score(t_data, t_label)
print(score_rf1)
score_rf2 = rf2.score(t_data, t_label)
print(score_rf2)
score_rf3 = rf3.score(t_data, t_label)
print(score_rf3)
```
```
0.962
0.9666
0.9696
```

Then I test the data from USPS, it just got accuracy as 0.308.


```
score_rf = classifier2.score(USPSMat, USPSTar)
print(score_rf)
```
```
0.3081654082704135
```

6 ensemble the classifiers

To combine all these classifiers together, sklearn also offer a package. But the logistic regression of my model does not convenient to use that function of skearn. So I just implement a voting with python.

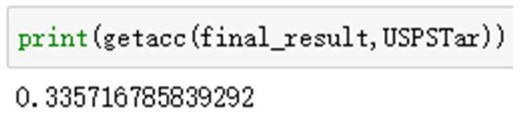
I used a function from collections.counter. It will find the element which appears most time in a list.

So I put each element which has same index from four classifiers into a list, and find the most time one. It will be append into the result list.

Finally, I got my reuslt list. And I implement a function to calculate accuracy of my list.

I got the accuracy=0.9338.

Then I test the data from USPS, I use the four model which have already trained before, it got accuracy as 0.3357.



```
print(getacc(final_result,USPSTar))

0.335716785839292
```

7 Answer questions

1) Yes, my results support the “No Free Lunch” theorem. I got very high test accuracy with MNIST dataset. But when I test my model with USPS dataset, it just got a very low result.

2)

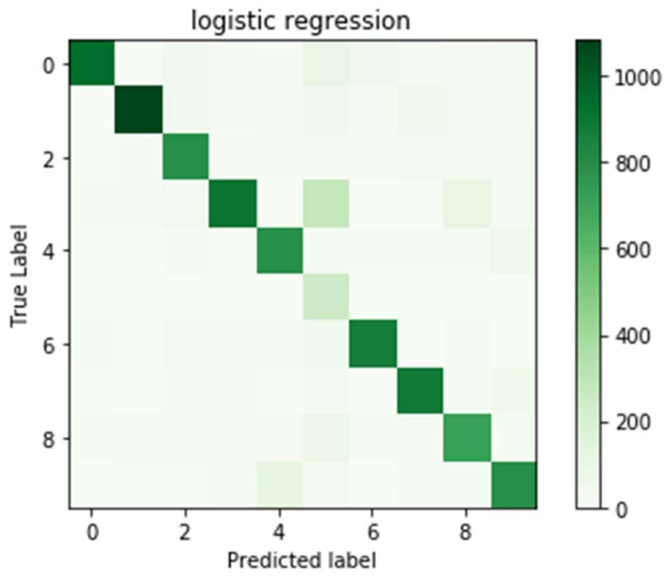


Figure 12: Confusion Matrix of Logistic Regression

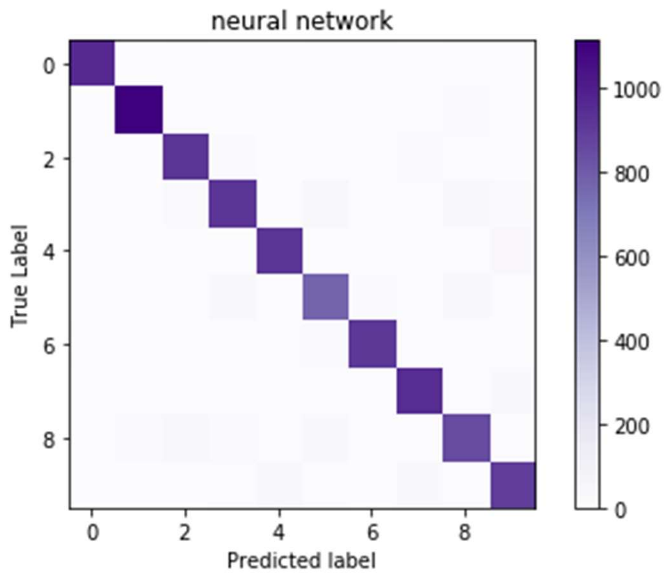


Figure 13: Neural Network Confusion Matrix

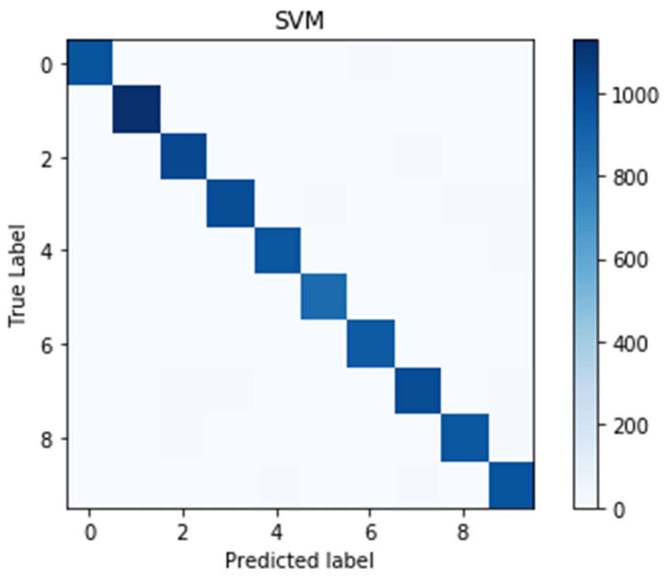


Figure 14: Confusion Matrix of SVM

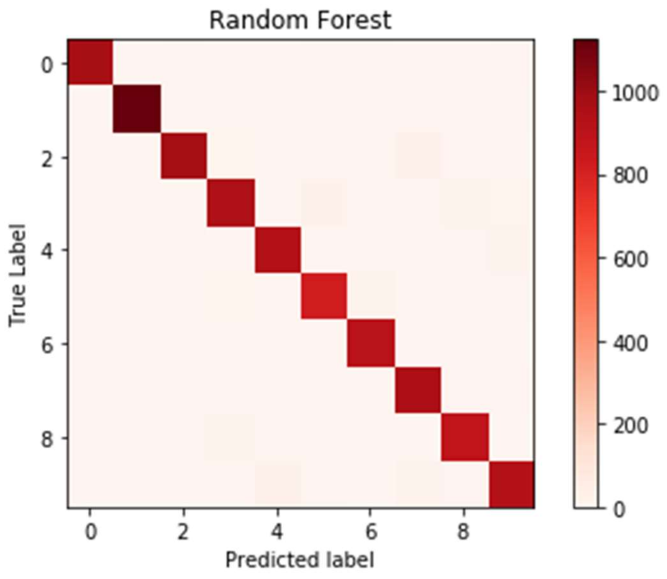


Figure 15: Confusion Matrix of Random Forest

SVM has the best performance.

3) Cause my neural network and logistic regression accuracy have not reach the expected value, the combined performance is worse than SVM and random forest accuracy.