CSC2515 HW3

Bingzhang Zhu
(Code Collaborators: Zhimao Lin)

November 25, 2021

1 Backpropagation

1.1 (a)

The dimension of $W^{(1)}$ is d $\times$ d
The dimension of $W^{(2)}$ is 1 $\times$ d
The dimension of Z_1 is d $\times$ 1
The dimension of Z_2 is d $\times$ 1

1.2 (b)

The number of parameters is: $d^2 + d$

1.3 (c)

$$\bar{y} = \frac{\partial L}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(y-t)^2 = \frac{1}{2}\,2\,(y-t)\cdot\frac{\partial y}{\partial y} = y-t$$

$$\overline{W^{(2)}} = \frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial W^{(2)}} = \bar{y}\,z_2^T = (y-t)\,z_2^T$$

$$\bar{z}_2 = \frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_2} = \bar{y}\,W^{(2)T} = (y-t)\,W^{(2)T}$$

$$\bar{h} = \frac{\partial L}{\partial h} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial h} = \bar{z}_2 = (y-t)\,W^{(2)T}$$

$$\bar{z}_1 = \frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial h}\frac{\partial h}{\partial z_1} = \bar{h}\circ\alpha'(z_1) = (y-t)W^{(2)T}\circ\alpha'(z_1)$$

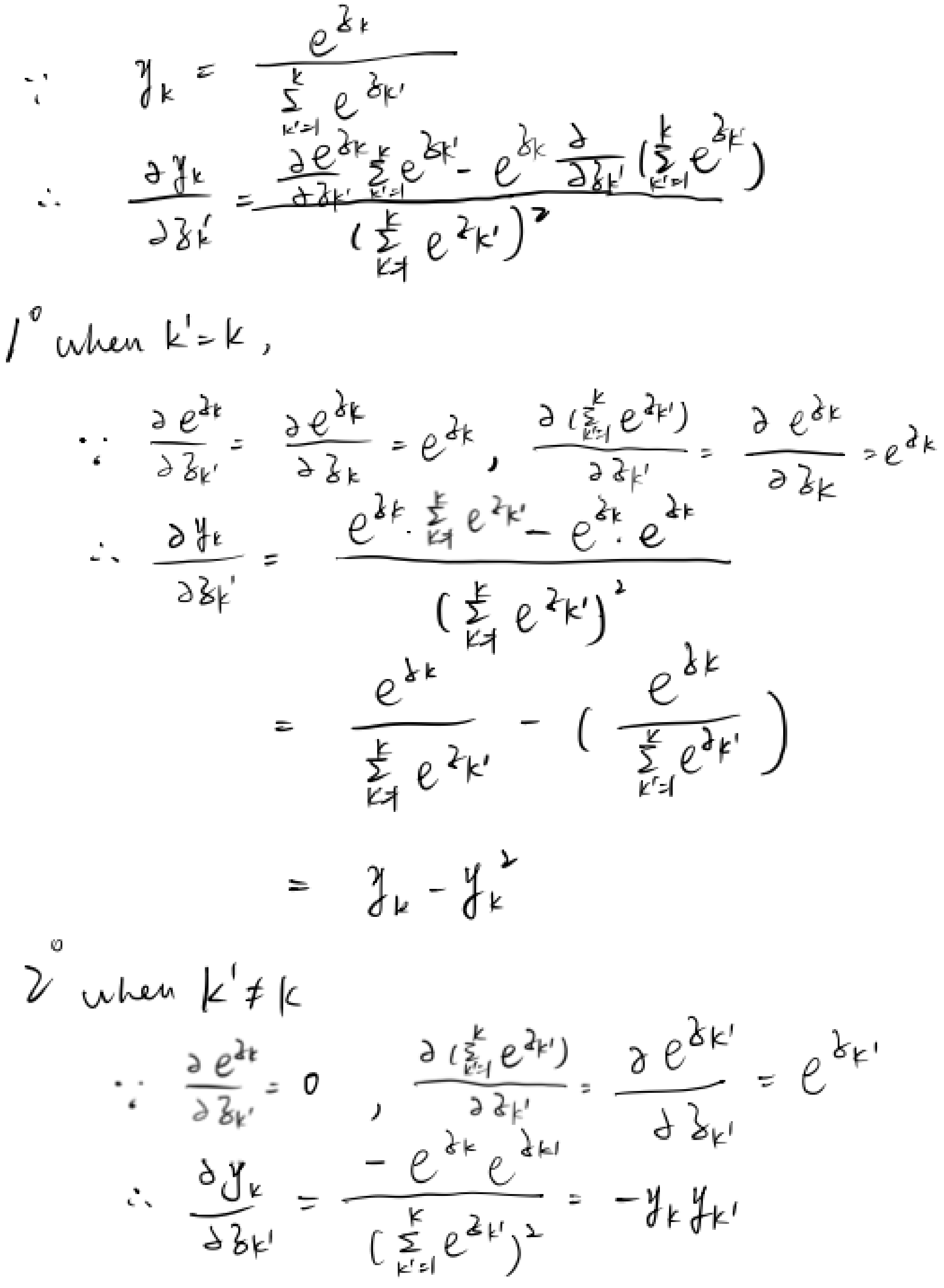
$$\overline{W^{(1)}} = \frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial h}\frac{\partial h}{\partial z_1}\frac{\partial z_1}{\partial W^{(1)}}$$
$$= \bar{z}_1\,x^T = (y-t)\,W^{(2)T}\circ\alpha'(z_1)\,x^T$$

$$\bar{x} = \frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial x} = \bar{z}_2\left(\frac{\partial h}{\partial x} + I\right)$$
$$= \bar{z}_2\left(\frac{\partial h}{\partial z_1}\frac{\partial z_1}{\partial x} + I\right) = \bar{z}_2\left(W_1^{(1)T}\alpha'(z_1) + I\right)$$
$$= W^{(1)T}(y-t)W^{(2)T}\circ\alpha'(z_1) + (y-t)\,W^{(2)T}$$

Figure 1:

2 Multi-Class Logistic Regression

2.1 (a)



$$\because \quad y_k = \frac{e^{z_k}}{\sum_{k'=1}^{K} e^{z_{k'}}}$$

$$\therefore \quad \frac{\partial y_k}{\partial z_{k'}} = \frac{\frac{\partial e^{z_k}}{\partial z_{k'}} \sum_{k'=1}^{K} e^{z_{k'}} - e^{z_k} \frac{\partial}{\partial z_{k'}} \left(\sum_{k'=1}^{K} e^{z_{k'}} \right)}{\left(\sum_{k=1}^{K} e^{z_{k'}} \right)^2}$$

1° when $k' = k$,

$$\because \quad \frac{\partial e^{z_k}}{\partial z_{k'}} = \frac{\partial e^{z_k}}{\partial z_k} = e^{z_k}, \quad \frac{\partial \left(\sum_{k'=1}^{K} e^{z_{k'}} \right)}{\partial z_{k'}} = \frac{\partial e^{z_k}}{\partial z_k} = e^{z_k}$$

$$\therefore \quad \frac{\partial y_k}{\partial z_{k'}} = \frac{e^{z_k} \cdot \sum_{k=1}^{K} e^{z_{k'}} - e^{z_k} \cdot e^{z_k}}{\left(\sum_{k=1}^{K} e^{z_{k'}} \right)^2}$$

$$= \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_{k'}}} - \left(\frac{e^{z_k}}{\sum_{k'=1}^{K} e^{z_{k'}}} \right)$$

$$= y_k - y_k^2$$

2° when $k' \neq k$

$$\because \quad \frac{\partial e^{z_k}}{\partial z_{k'}} = 0, \quad \frac{\partial \left(\sum_{k'=1}^{K} e^{z_{k'}} \right)}{\partial z_{k'}} = \frac{\partial e^{z_{k'}}}{\partial z_{k'}} = e^{z_{k'}}$$

$$\therefore \quad \frac{\partial y_k}{\partial z_{k'}} = \frac{- e^{z_k} e^{z_{k'}}}{\left(\sum_{k'=1}^{K} e^{z_{k'}} \right)^2} = -y_k y_{k'}$$

Figure 2:

2.2 (b)

$$\frac{\partial L_{CE}(t, y(x, W))}{\partial W_k} = \sum_{n=1}^{K} \frac{\partial L_{CE}}{\partial y_n} \cdot \frac{\partial y_n}{\partial z_k} \cdot \frac{\partial z_k}{\partial W_k}$$

$$\because \begin{cases} \dfrac{\partial L_{CE}}{\partial y_n} = \dfrac{\partial\left(-\sum_{n=1}^{K} t_n \log y_n\right)}{\partial y_n} = \dfrac{\partial(-t_n \log y_n)}{\partial y_n} = \dfrac{-t_n}{y_n}, n \in [1, K] \\ \dfrac{\partial y_n}{\partial z_k} = \begin{cases} y_k - y_k^2, & n = k \\ -y_n y_k, & n \neq k \end{cases}, \text{ from } Q2(a) \\ \dfrac{\partial z_k}{\partial W_k} = x \end{cases}$$

$$\therefore \frac{\partial L_{CE}(t, y(x, W))}{\partial W_k} = \left[\underbrace{\sum_{n=1 \wedge n \neq k}^{K} \left[\left(\frac{-t_n}{y_n}\right) \cdot (-y_n y_k)\right]}_{n \neq k} + \underbrace{\left(-\frac{t_k}{y_k}\right) \cdot (y_k - y_k^2)}_{n = k}\right] \cdot x$$

$$= \left[\sum_{n=1 \wedge n \neq k}^{K} (t_n) y_k - t_k (1 - y_k)\right] \cdot x$$

$\because t$ is a one hot encoding of the output

$$\therefore \sum_{n=1}^{K} t_n = 1 \Rightarrow \sum_{n=1 \wedge n \neq k}^{K} t_n = 1 - t_k$$

$$\therefore \frac{\partial L_{CE}(t, y(x, W))}{\partial W_k} = \left[(1 - t_k) y_k - t_k + y_k t_k\right] \cdot x$$

$$= (y_k - t_k) x$$

Figure 3:

3 Handwritten Digit Classification

3.1 Load the data and plot the means for each of the digit classes in the training data (include these in your report).

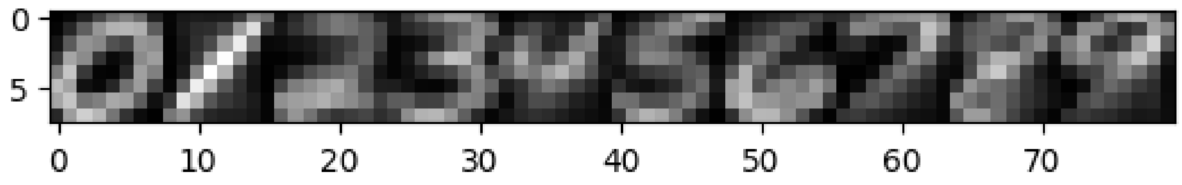


Figure 4:

3.2 K-NN Classifier

1.
The train classification accuracy of [K = 1] is [1.0].
The test classification accuracy of [K = 1] is [0.96875].
The train classification accuracy of [K = 15] is [0.9594285714285714].
The test classification accuracy of [K = 15] is [0.9585].

2.
If K-NN encounters ties and K is less than 10, we can increase K to K+1.
If K-NN encounters ties and K is not less than 10, we randomly assign it to one of the class.

3.
The optimal K is [k = 1]
The train classification accuracy is [1.0]
The average accuracy across folds is [0.9648571428571427]
The test accuracy of the best K is [0.96875]

3.3 Classifiers comparison

3.3.1 MLP - Neural Network Classifier

We used *GridSearchCV*() with cv = 3 to find the optimal parameter set from the sets:

```
params= {'hidden_layer_sizes': [(64, 64), (100, 100), (64, 64, 64, 64) (64, 64, 64)],
'activation':["logistic", "tanh", "relu"], 'learning_rate': ["constant", "invscaling", "adaptive"]}
```

Under 3-fold Cross Validation, we know that the optimal parameter set for MLP is:

```
params= {'hidden_layer_sizes': [(64, 64, 64,)], 'activation':["tanh"], 'learning_rate': ["adaptive"]}
```

And we used this parameter set for the model design. The overfitting problem has been taken into consideration while doing the Cross Validation and choosing best parameters.

3.3.2 SVM classifier

We used *GridSearchCV*() with cv = 3 to find the optimal parameter set from the sets:

```
params= {'kernel': ["rbf", "poly", "sigmoid"], 'gamma':[0.1, 0.01, 0.001], 'C':[100, 10, 1, 0.1, 0.01]}
```

Under 3-fold Cross Validation, we know that the optimal parameter set for SVM Classifier is:

```
params= {'kernel': ["rbf"], 'gamma':[0.1], 'C':[10]}
```

And we used this parameter set for the model design.

3.3.3 AdaBoost Classifier

We used *DecisionTreeClassifier*() as the *base_estimator* and *GridSearchCV*() with cv = 3 to find the optimal parameter set from the sets:

```
params= {"base_estimator__min_samples_leaf" : [2, 5], "base_estimator__max_depth": [5, 7],
'n_estimators': [50, 200, 300], 'learning_rate':[1, 0.1, 0.01, 0.001]}
```

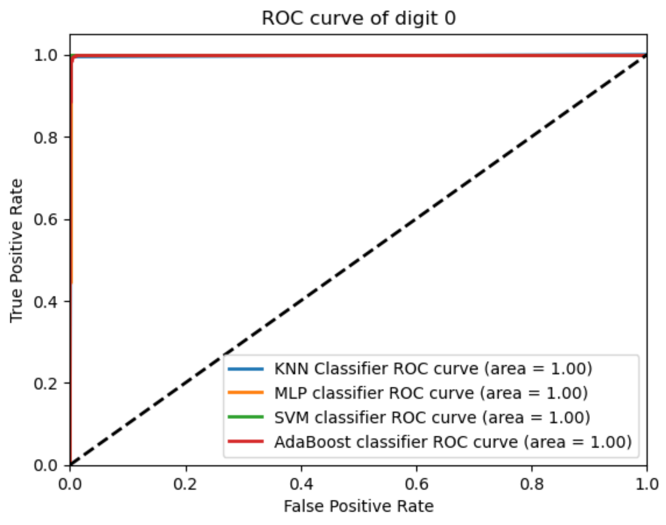
Under 3-fold Cross Validation, we know that the optimal parameter set for SVM Classifier is:

```
params= {"base_estimator__min_samples_leaf" : [2], "base_estimator__max_depth": [7],
'n_estimators': [300], 'learning_rate':[1]}
```

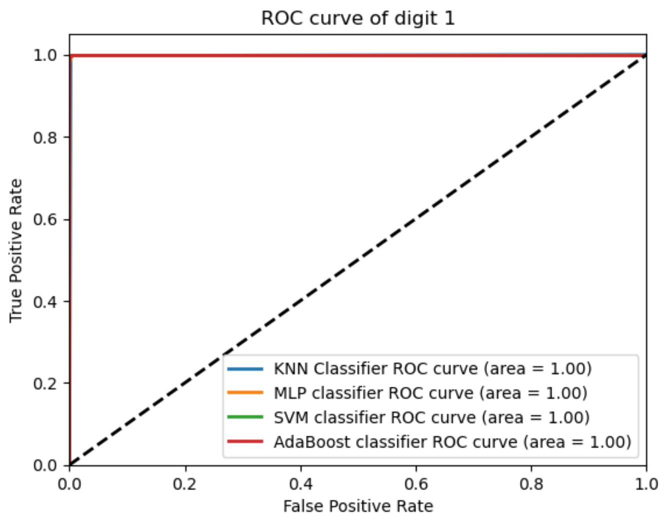
And we used this parameter set for the model design.

3.4 Model Comparison

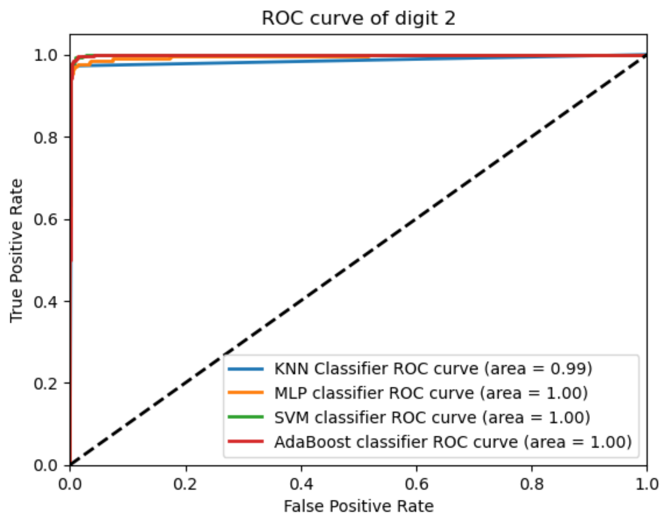
3.4.1 ROC (Receiver Operating Characteristics) curve



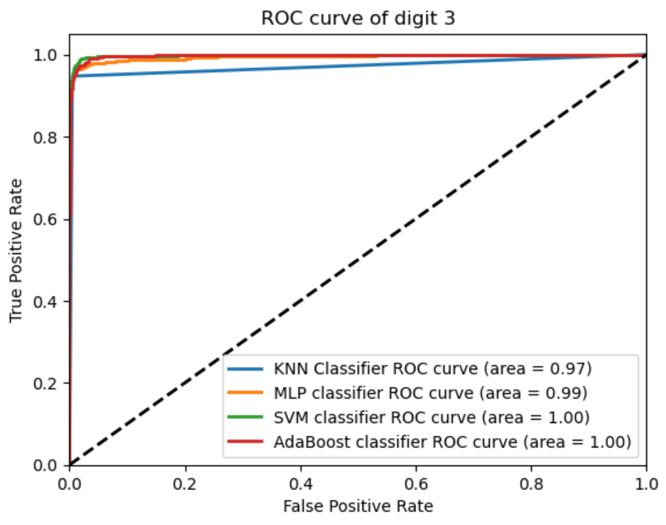
(a) digit 0



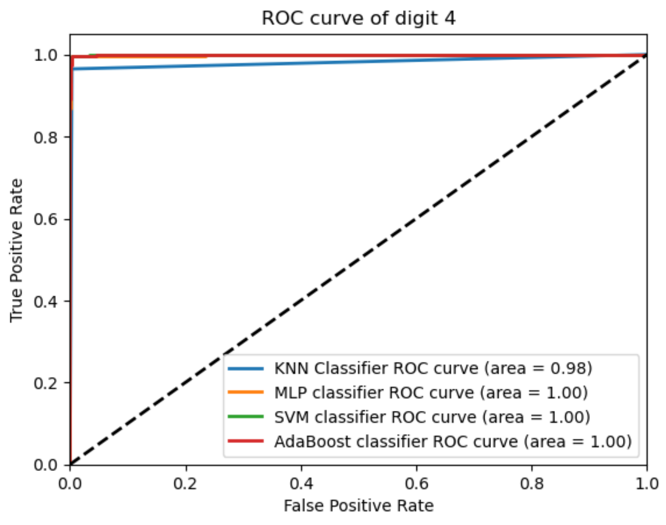
(b) digit 1



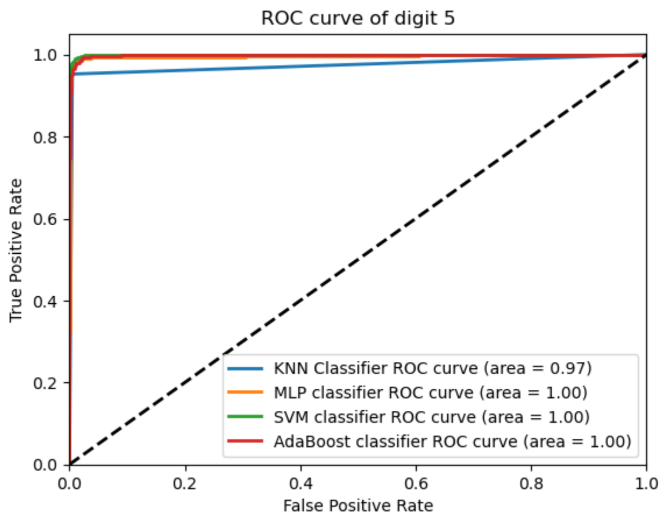
(c) digit 2



(d) digit 3

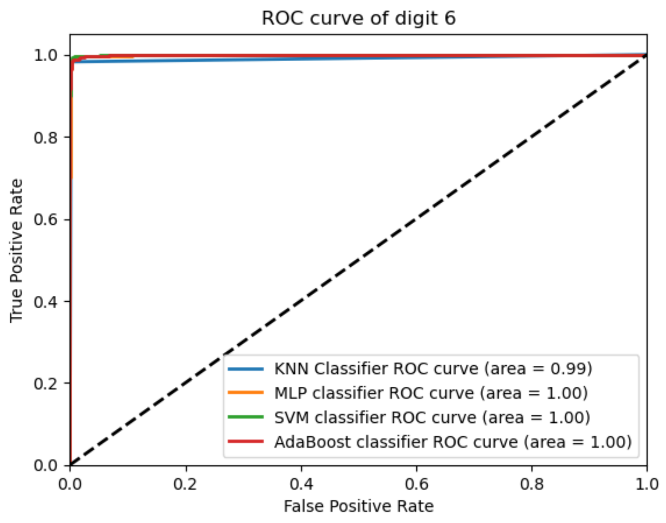


(e) digit 4

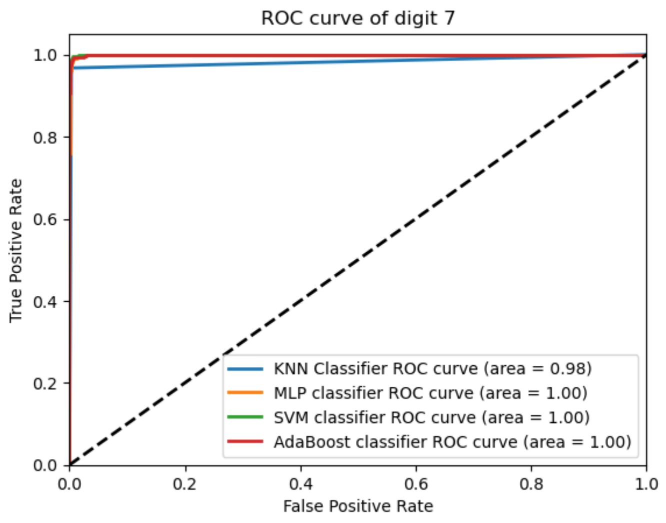


(f) digit 5

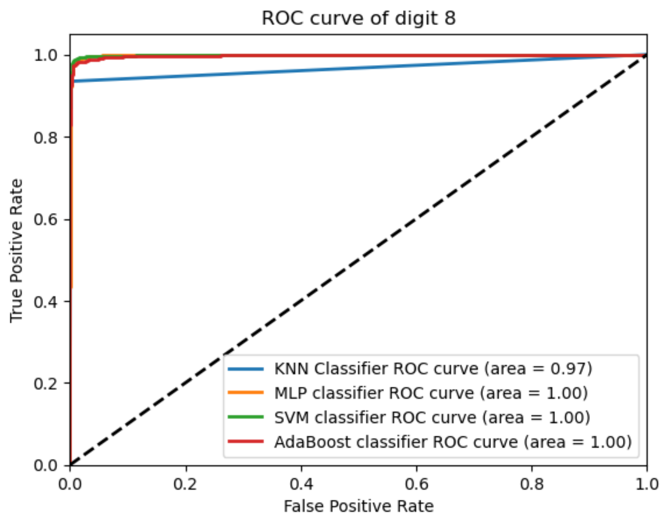
Figure 5: This is the ROC curve for the first six digit classes



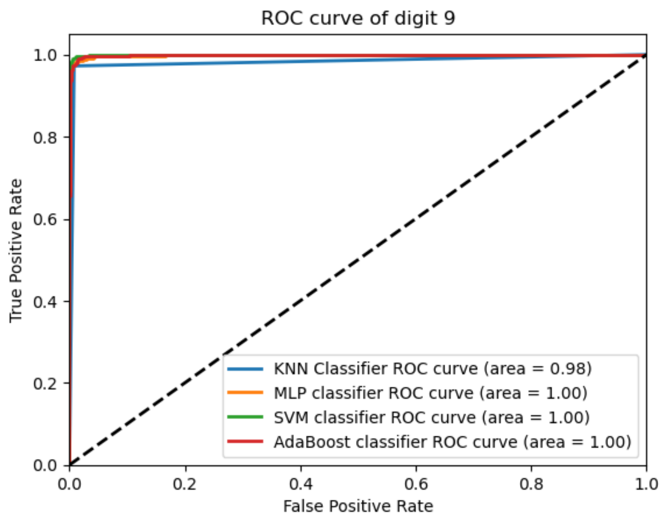
(a) digit 6



(b) digit 7



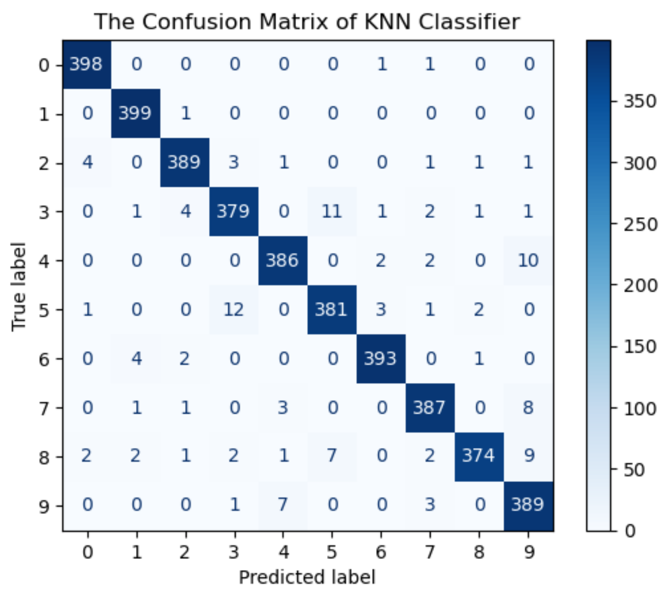
(c) digit 8



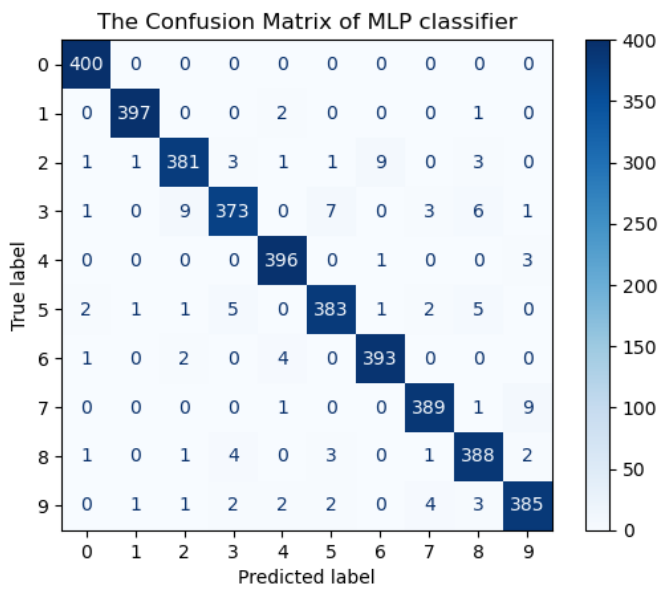
(d) digit 9

Figure 6: This is the ROC curve for the remaining four digit classes

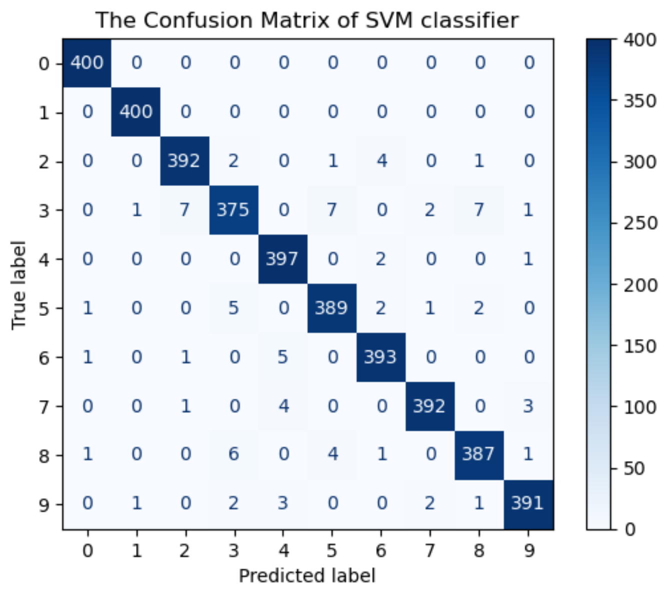
3.4.2 Confusion Matrix



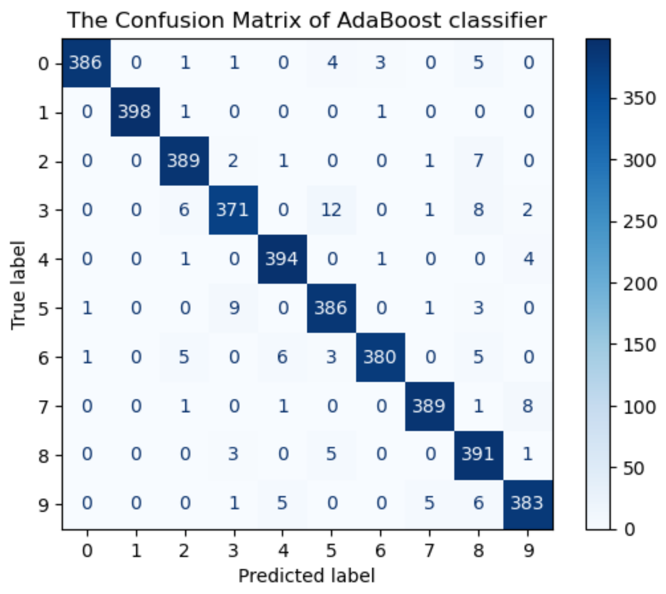
(a) KNN



(b) MLP



(c) SVM



(d) AdaBoost

Figure 7: Confusion Matrix for different models

3.4.3 Accuracy

```
The Accuracy of KNN Classifier is: 0.96875
The Accuracy of MLP classifier is: 0.97125
The Accuracy of SVM classifier is: 0.979
The Accuracy of AdaBoost classifier is: 0.96675
```

3.4.4 Precision

The Precision of KNN Classifier is:

```
The precision of class=[Digit 0] is [0.9827160493827161].
The precision of class=[Digit 1] is [0.9803439803439803].
The precision of class=[Digit 2] is [0.9773869346733668].
The precision of class=[Digit 3] is [0.9546599496221663].
The precision of class=[Digit 4] is [0.9698492462311558].
The precision of class=[Digit 5] is [0.9548872180451128].
The precision of class=[Digit 6] is [0.9825].
The precision of class=[Digit 7] is [0.9699248120300752].
The precision of class=[Digit 8] is [0.9868073878627969].
The precision of class=[Digit 9] is [0.930622009569378].
The average precision across all classes is [0.9689697587760747].
```

The Precision of MLP classifier is:

```
The precision of class=[Digit 0] is [0.9852216748768473].
The precision of class=[Digit 1] is [0.9925].
The precision of class=[Digit 2] is [0.9645569620253165].
The precision of class=[Digit 3] is [0.9638242894056848].
The precision of class=[Digit 4] is [0.9753694581280788].
The precision of class=[Digit 5] is [0.9671717171717171].
The precision of class=[Digit 6] is [0.9727722772277227].
The precision of class=[Digit 7] is [0.974937343358396].
The precision of class=[Digit 8] is [0.9533169533169533].
The precision of class=[Digit 9] is [0.9625].
The average precision across all classes is [0.9712170675510716].
```

The Precision of SVM classifier is:

```
The precision of class=[Digit 0] is [0.9925558312655087].
The precision of class=[Digit 1] is [0.9950248756218906].
The precision of class=[Digit 2] is [0.9775561097256857].
The precision of class=[Digit 3] is [0.9615384615384616].
The precision of class=[Digit 4] is [0.9706601466992665].
The precision of class=[Digit 5] is [0.970074812967581].
The precision of class=[Digit 6] is [0.9776119402985075].
The precision of class=[Digit 7] is [0.9874055415617129].
The precision of class=[Digit 8] is [0.9723618090452262].
The precision of class=[Digit 9] is [0.9848866498740554].
The average precision across all classes is [0.9789676178597897].
```

The Precision of AdaBoost classifier is:

```
The precision of class=[Digit 0] is [0.9948453608247423].
The precision of class=[Digit 1] is [1.0].
The precision of class=[Digit 2] is [0.9628712871287128].
The precision of class=[Digit 3] is [0.958656330749354].
The precision of class=[Digit 4] is [0.9680589680589681].
The precision of class=[Digit 5] is [0.9414634146341463].
The precision of class=[Digit 6] is [0.987012987012987].
The precision of class=[Digit 7] is [0.9798488664987406].
The precision of class=[Digit 8] is [0.9178403755868545].
The precision of class=[Digit 9] is [0.9623115577889447].
The average precision across all classes is [0.9672909148283451].
```

3.4.5 Recall

The Recall of KNN Classifier is:

```
The recall of class=[Digit 0] is [0.995].
The recall of class=[Digit 1] is [0.9975].
The recall of class=[Digit 2] is [0.9725].
The recall of class=[Digit 3] is [0.9475].
The recall of class=[Digit 4] is [0.965].
The recall of class=[Digit 5] is [0.9525].
The recall of class=[Digit 6] is [0.9825].
The recall of class=[Digit 7] is [0.9675].
The recall of class=[Digit 8] is [0.935].
The recall of class=[Digit 9] is [0.9725].
```

The average recall across all classes is [0.96875].

The Recall of MLP classifier is:

```
The recall of class=[Digit 0] is [1.0].
The recall of class=[Digit 1] is [0.9925].
The recall of class=[Digit 2] is [0.9525].
The recall of class=[Digit 3] is [0.9325].
The recall of class=[Digit 4] is [0.99].
The recall of class=[Digit 5] is [0.9575].
The recall of class=[Digit 6] is [0.9825].
The recall of class=[Digit 7] is [0.9725].
The recall of class=[Digit 8] is [0.97].
The recall of class=[Digit 9] is [0.9625].
```

The average recall across all classes is [0.9712500000000001].

The Recall of SVM classifier is:

```
The recall of class=[Digit 0] is [1.0].
The recall of class=[Digit 1] is [1.0].
The recall of class=[Digit 2] is [0.98].
The recall of class=[Digit 3] is [0.9375].
The recall of class=[Digit 4] is [0.9925].
The recall of class=[Digit 5] is [0.9725].
The recall of class=[Digit 6] is [0.9825].
The recall of class=[Digit 7] is [0.98].
The recall of class=[Digit 8] is [0.9675].
The recall of class=[Digit 9] is [0.9775].
```

The average recall across all classes is [0.9789999999999999].

The Recall of AdaBoost classifier is:

```
The recall of class=[Digit 0] is [0.965].
The recall of class=[Digit 1] is [0.995].
The recall of class=[Digit 2] is [0.9725].
The recall of class=[Digit 3] is [0.9275].
The recall of class=[Digit 4] is [0.985].
The recall of class=[Digit 5] is [0.965].
The recall of class=[Digit 6] is [0.95].
The recall of class=[Digit 7] is [0.9725].
The recall of class=[Digit 8] is [0.9775].
The recall of class=[Digit 9] is [0.9575].
```

The average recall across all classes is [0.96675].

3.4.6 Briefly summarize the performance of each model

1. KNN
From previous question, we know that with 10-fold Cross Validation the best K is 1. For 1-NN Classifier, we know that it performs the worst when we look at the ROC Curve and AUC(Area under the curve). However, it performs the second worst when we look into Accuracy, average Precision and Recall across all classes, with its accuracy = 0.96875, average Precision = 0.96897, average Recall = 0.96875.

2. MLP
For MLP Classifier, we know that it performs the second worst when we look at the ROC Curve and AUC(Area under the curve). In digit 3, its AUC=0.99, while for all the other digits it has AUC=1.00. However, it performs the second best when we look into Accuracy, average Precision and Recall across all classes, with its accuracy = 0.97125, average Precision = 0.97122, average Recall = 0.97125.

3. SVM
For SVM Classifier, we know that it performs the best for all the metrics. When we look at the ROC Curve, it has all AUC(Area under the curve) equals to 1.00. It also performs the best when we look into Accuracy, average Precision and Recall across all classes, with its accuracy = 0.979, average Precision = 0.9790, average Recall = 0.9790.

4. AdaBoost
For AdaBoost Classifier, we know that it performs the best when we look at the ROC Curve and AUC(Area under the curve). It has all AUC(Area under the curve) equals to 1.00. However, it performs the worst least when we look into Accuracy, average Precision and Recall across all classes, with its accuracy = 0.96675, average Precision = 0.9673, average Recall = 0.96675.

3.4.7 Which classifier performed best? Which performed worst? Did this match your expectations? The most important is to show good understanding of why a certain classifier did better in the provided dataset.

In this homework, the SVM Classifier performs the best and the AdaBoost Classifier performs the worst. And this is not exactly what I expect. I expected the MLP has the best performance and the KNN has the worst peformance, because I think MLP has multiple layers and it could has more flexibility and could learn more complex model so that it could has the best predictions and KNN is such a simple model it might has the worst performance. I think the reason for getting this result could be:

First, the data in this classification problem is not complex with only 8*8 pixels, so that simple model like KNN could have good performance on it.

Secondly, the size of the dataset is not big, with only 700 instance for each class in training set. Under this circumstance, the MLP has a very high chance to be over-fitting and has a bad performance. To avoid over-fitting we could not has so many layers and units and this may cause it is not the best on the dataset.

Thirdly, the SVM Classifier shows the best performance because we chose the right kernal Function. Kernel Function transforms the training set of data so that a non-linear decision surface is able to transformed to a linear equation in a higher number of dimension spaces. In this not big dataset, it could transform data in the right way which helps it develop the best prediction performance.

Finally, why the Adaboost Classifier performs the worst could might be the base_estimator we chose it not that suitable for this dataset. Even though it took the longest time to build the model, it still cannot learn the training set very well and end up with the worst performance.