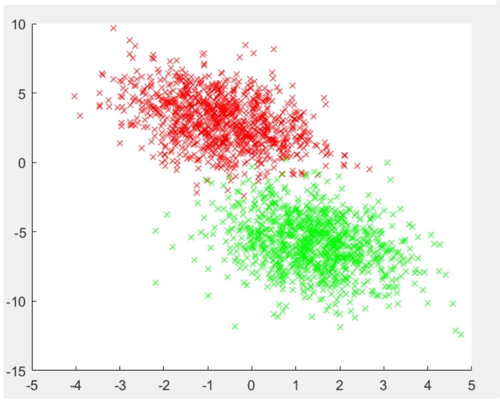
TBMI26 – Computer Assignment Report Supervised Learning

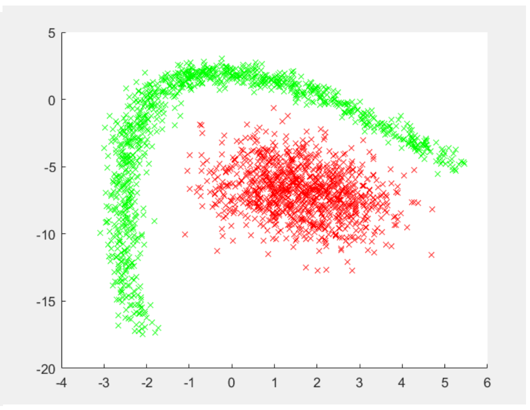
Deadline – March 14 2021

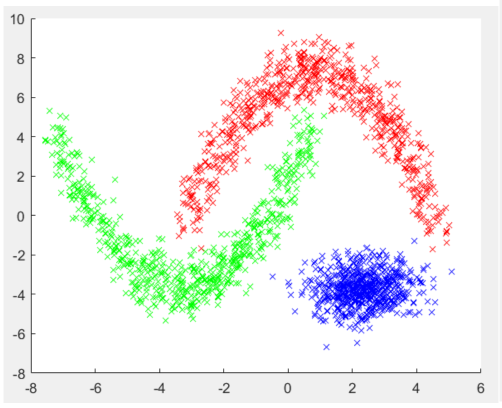
Author/-s:Wuhao Wang(wuhwa469)

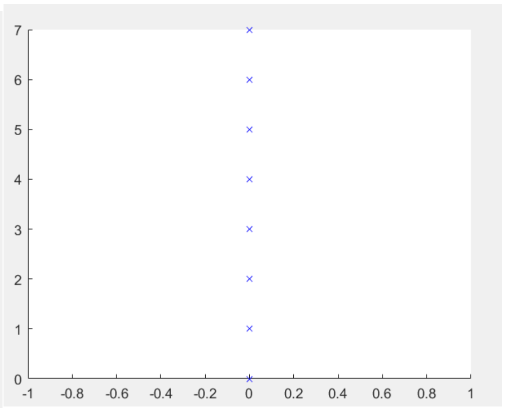
In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. Please upload the document in PDF format. **You will also need to upload all code in .m-file format**. We will correct the reports continuously so feel free to send them as soon as possible. If you meet the deadline you will have the lab part of the course reported in LADOK together with the exam. If not, you'll get the lab part reported during the re-exam period.

1. **Give an overview of the four datasets from a machine learning perspective. Consider if you need linear or non-linear classifiers etc.**


data1


data2


data3


OCR

Data 1 can be classified by liner model, where as data2/3 need a non-liner model to have a good result. It is hard to explore the OCR data from this picture. But the data is actually the similar matrix where each row is a sample with 64 features representing pixel value. In this case, I think a non-liner model is also needed for OCR data.

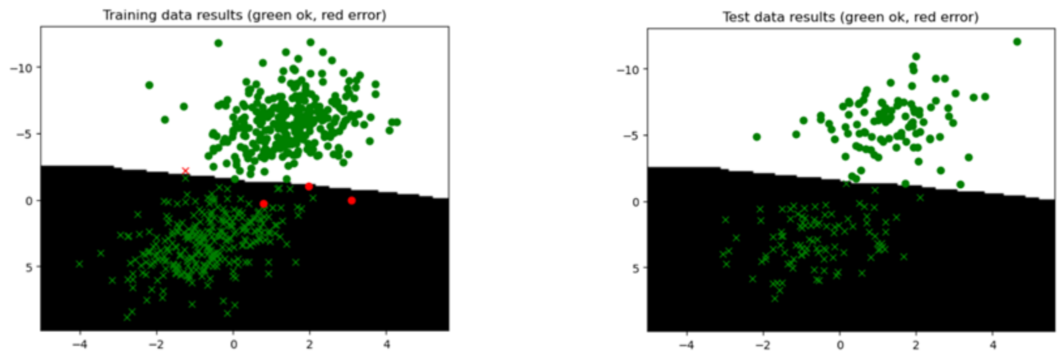
2. **Explain why the down sampling of the OCR data (done as pre-processing) result in a more robust feature representation. See http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits**
   In this method, they use a pixel value to represent a 4*4 block in the original picture which means the noises or outliers would be 'averaged'. In this case, those noise and outliers would be less 'important' in the modeling stage which makes a more robust feature.

3. **Give a short summary of how you implemented the kNN algorithm.**
   In each iteration(loop the input data):
   1 calculate the k minimum distance
   2 find the corresponding k points and their information(index,label)
   3 count each label in the k points
   4 if there is only one maximum value, then we set this label to input data
   5 if there is more than one maximum value, then for every group(corresponding to the label), we calculate the sum of distance between the the train points and input data.
   6 if there is only one minimum value, we set the corresponding label.
   7 if there is more than one labels(rarely happen), we just set it randomly.

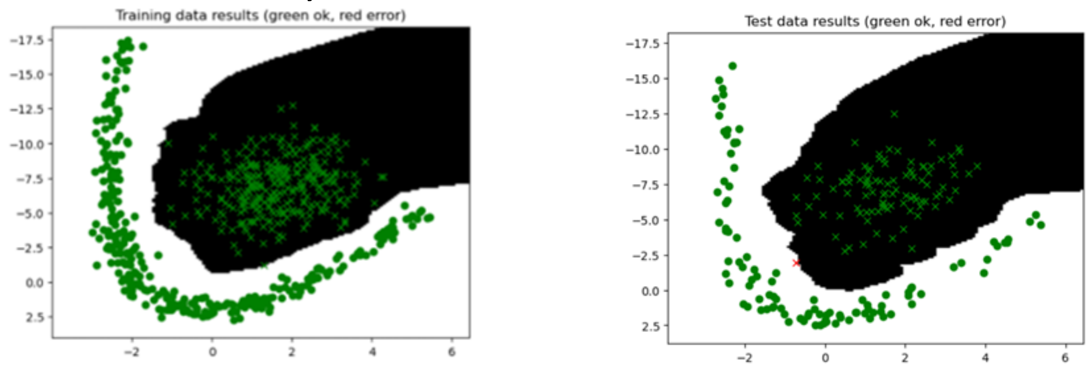
4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**
   According to the discussion in question3, we will first check which class has the maximum value. Here if there are only 2 neighbors and they have different class, we will calculate the distance and pick the shortest one. If the distance is same, we will pick it randomly.

5. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**
   I use 5-folds cross-validation (Five groups, four for training and one for validation). In each iteration, I select one group data and there will be 30 candidates(k from 1 to 30).After finishing cross-validation, there will be 5 results for each k. I pick the k with highest mean accuracy among these 5-folds.

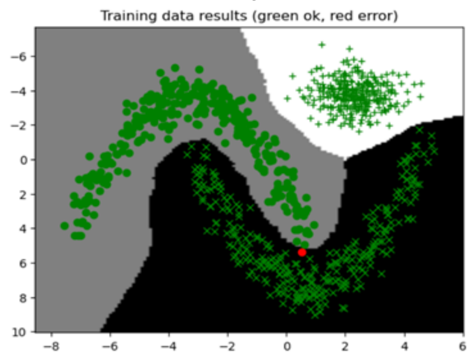
Data1:K = 29 Test Accuracy 1

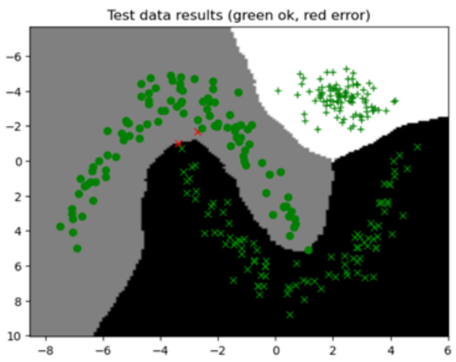


Data2:K = 1 Test Accuracy 0.995

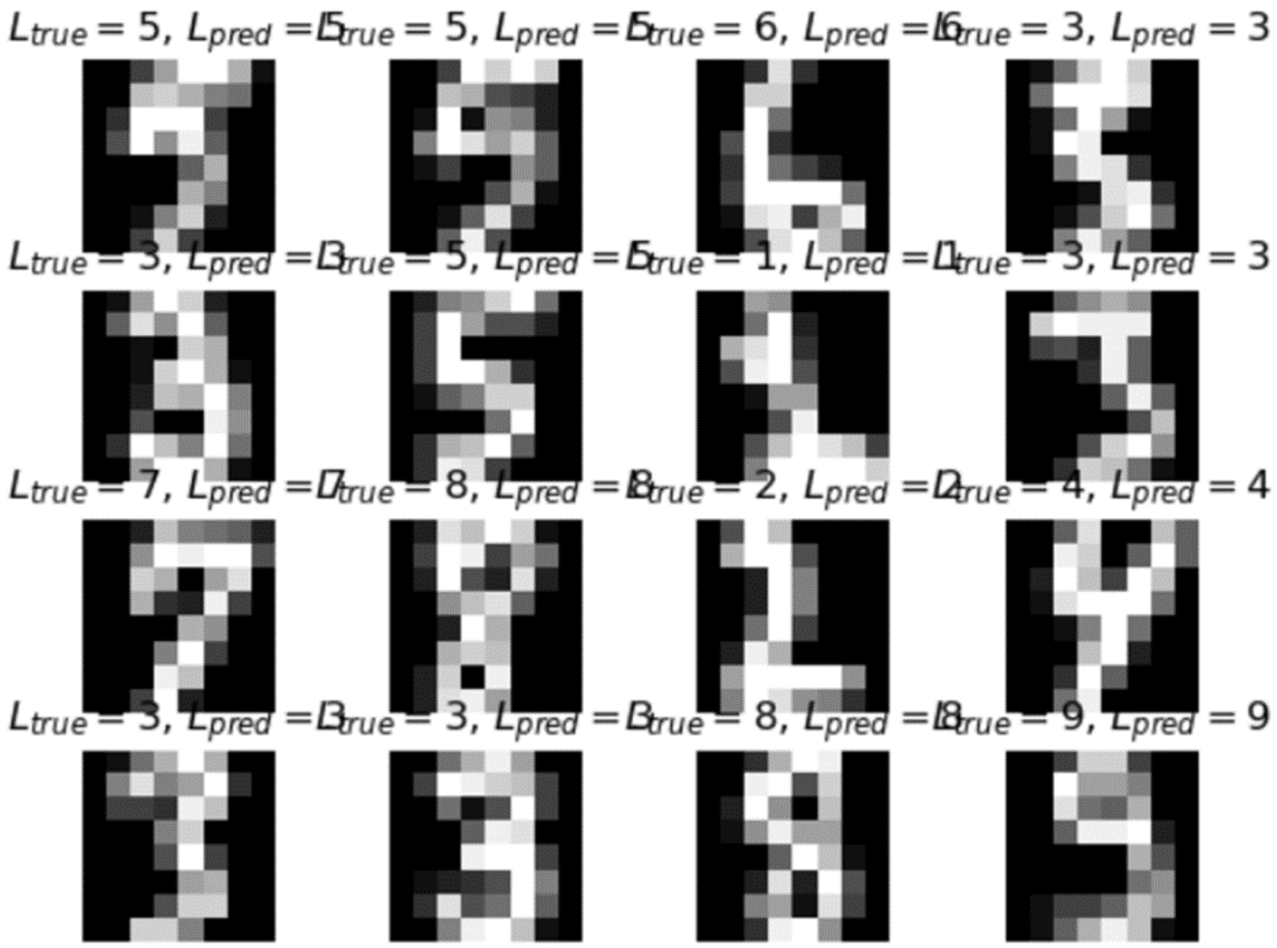


Data3: K = 9 Accuracy 0.9933





Data4: K = 1 Accuracy 0.984



6. **Give a short summary of your backprop implementations (single + multi). You do not need to derive the update rules.**

   Single:

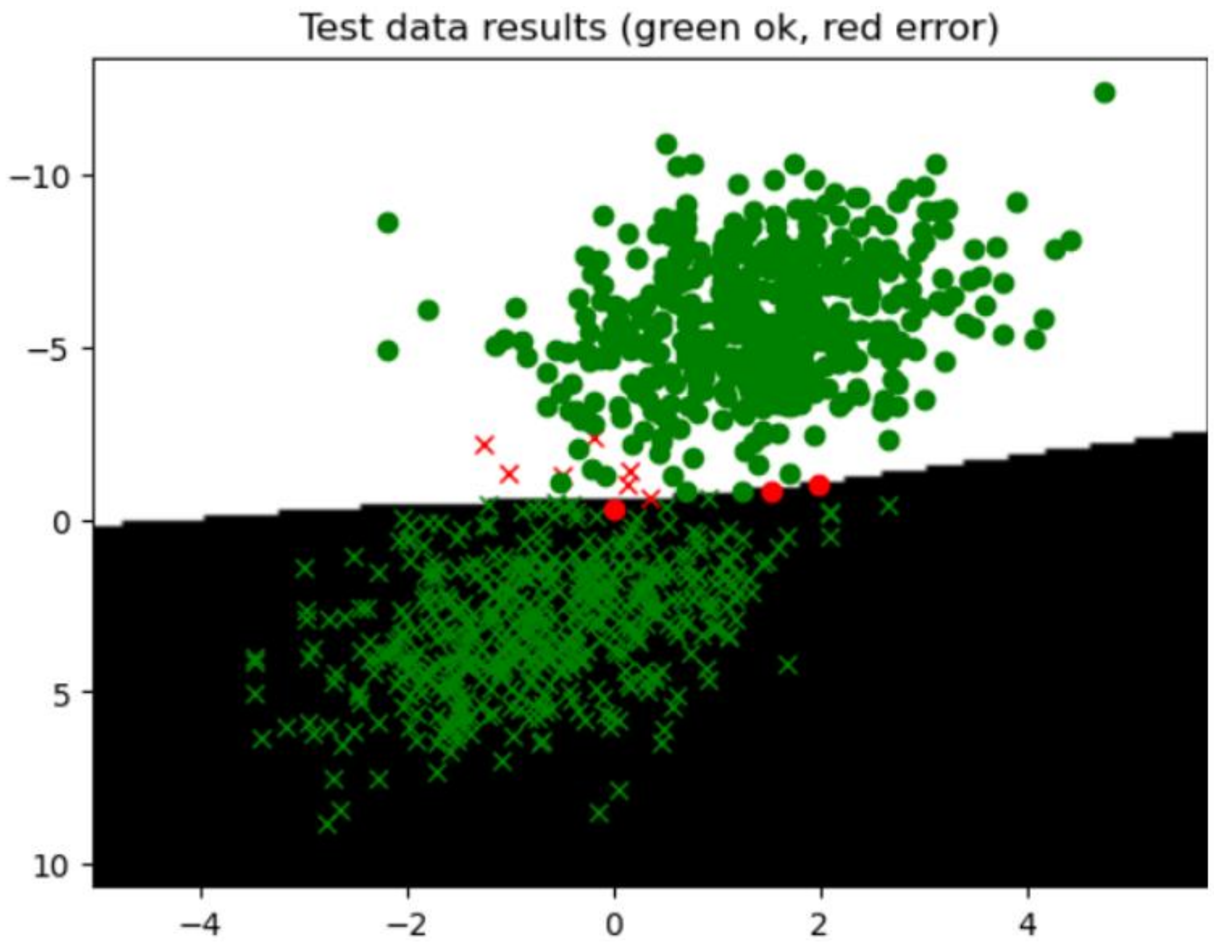
   1 we need an activation function at the end to get the classification, so, in the backpropagation part we need to add its derivation into the formula.

   Multi:

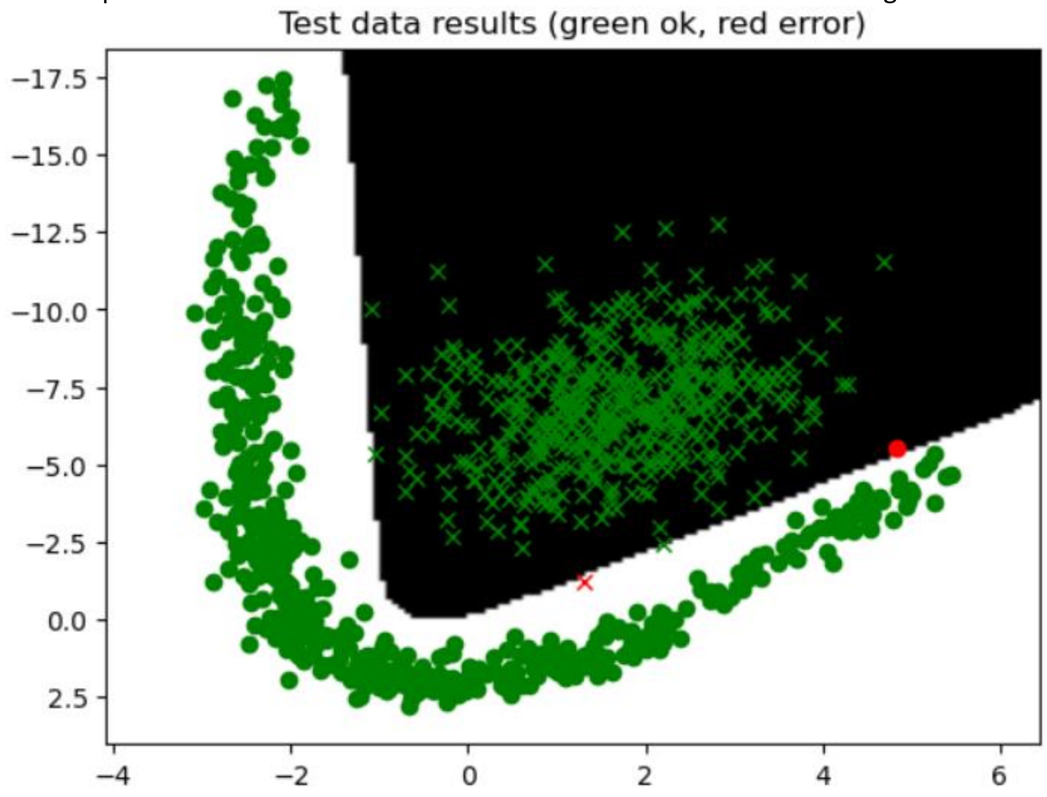
   1 There is a bias in the hidden layer output, it should be added after the activation function of hidden layer.

   2 During the backpropagation part, when the direction is from hidden layer to input layer, the influence of the bias(in the hidden layer) should be removed. This is done by reshape the hidden layer and the coefficient between hidden layer and output layer.

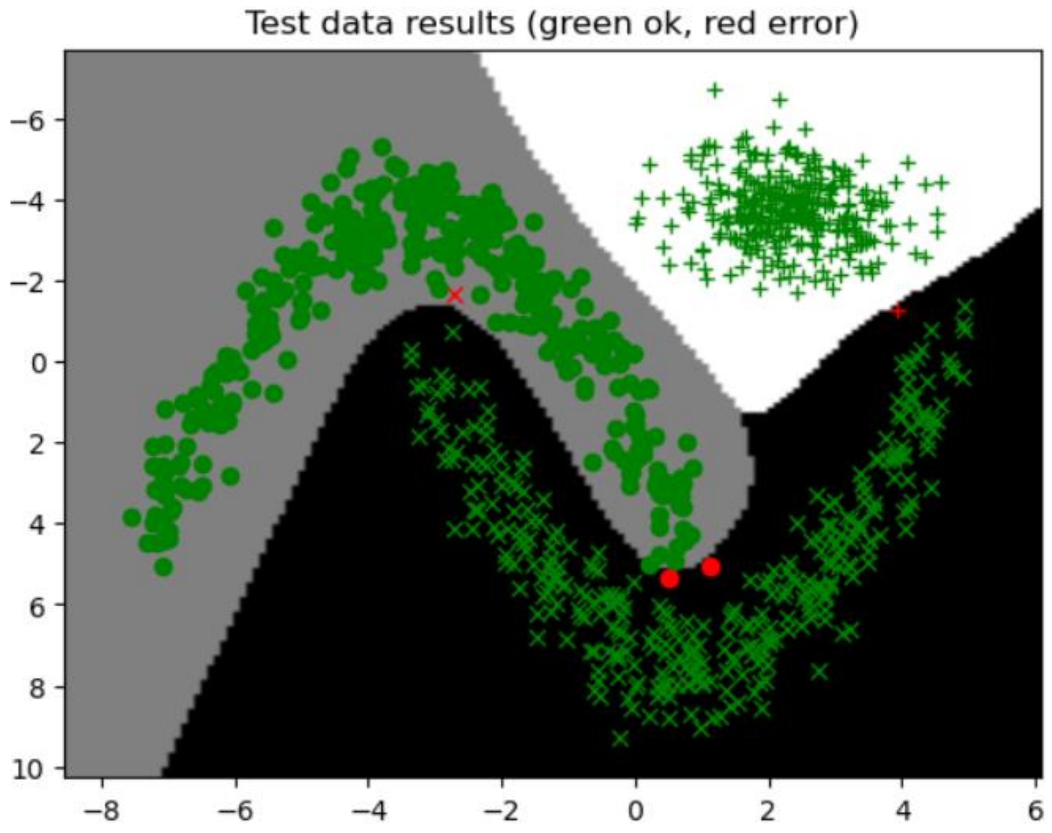
7. **Present the results from the neural network training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result for each dataset, including parameters etc.**
   data1: Accuracy:0.99
   parameters: numHidden = 3 numIterations =175 learningRate = 0.01



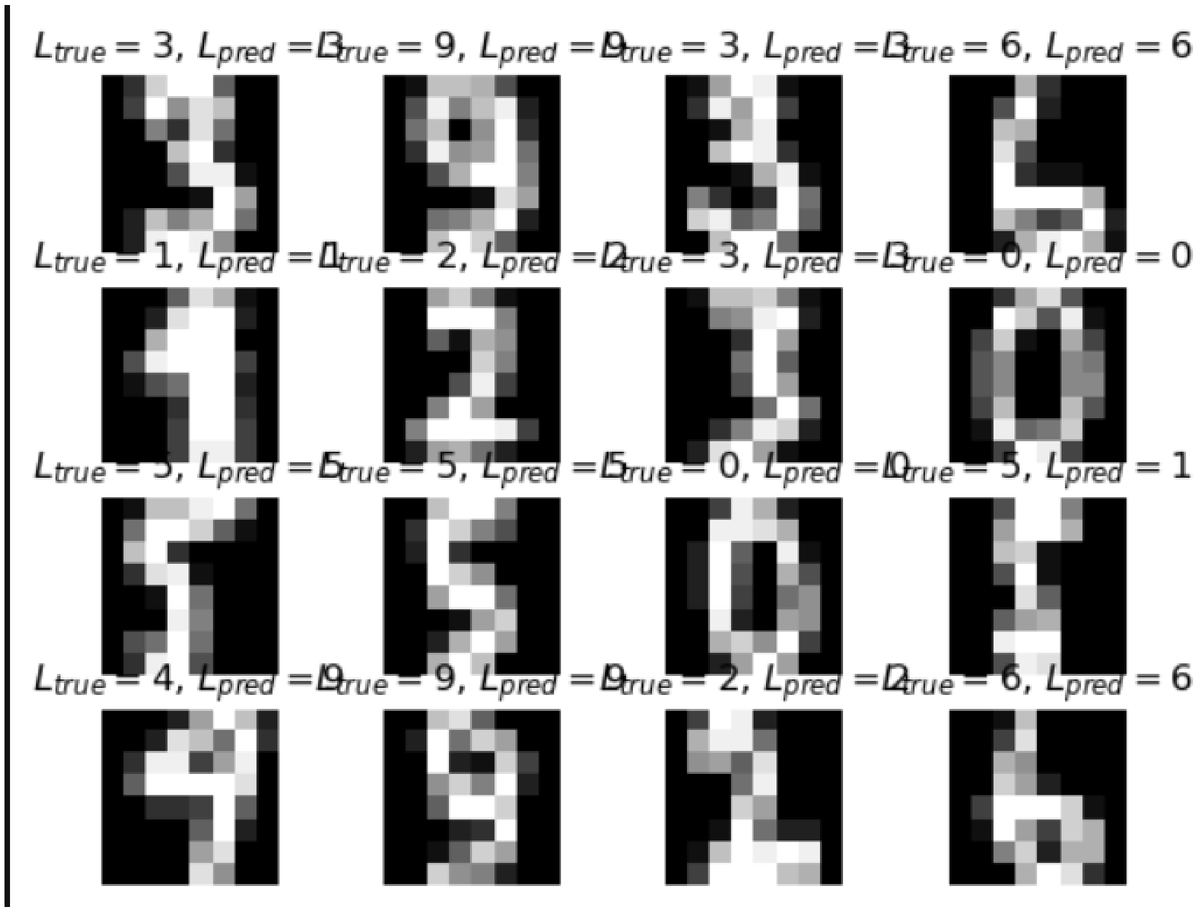
data2: Accuracy:0.9980
parameters: numHidden = 3 numIterations = 1500 learningRate = 0.05



data3: Accuracy:0.9960
parameters: numHidden = 5 numIterations = 6001 learningRate = 0.04



data4:	Accuracy:0.9630
parameters:	numHidden = 78	numIterations = 6100	learningRate = 0.065



For data1, the boundary looks like a liner one. So, the model should not be too complex. So, finally those parameters are picked.

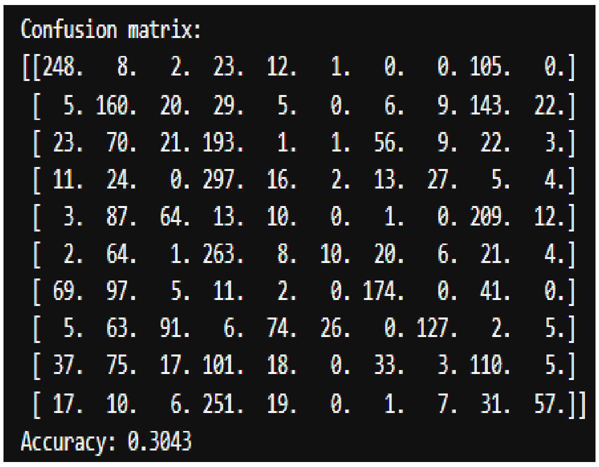
For data2, the boundary looks like a simple polyomino, so, I just use the parameters from data1 and larger the numiterations. It works pretty well.

For data3, the boundary is comparatively complex, so I start from 5 and find 5 is good enough. I copied numiterations from parameters in data2 and found it was underfitting, so the larger numiterations rate is applied, and it works well.

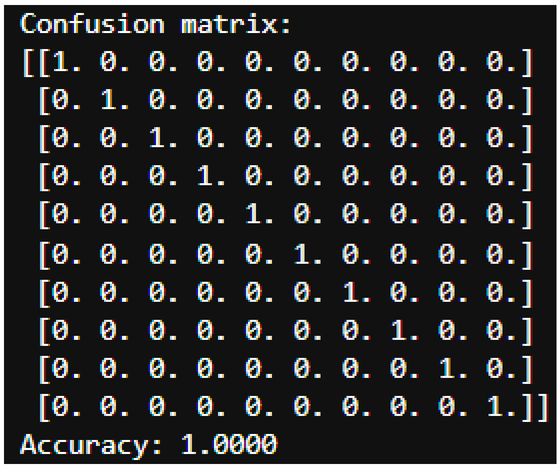
For data4, we cannot see the boundary explicitly, but it will not be simple. So, the start point was copied from data3 and finally found 78 could be the optimal one. Other parameters just copied from data3 and modified very little.

8. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**

for data4 parameters: numHidden = 20   numIterations = 6100   learningRate = 0.065
and we only use 10 samples for training.



```
Confusion matrix:
[[248.   8.   2.  23.  12.   1.   0.   0. 105.   0.]
 [  5. 160.  20.  29.   5.   0.   6.   9. 143.  22.]
 [ 23.  70.  21. 193.   1.   1.  56.   9.  22.   3.]
 [ 11.  24.   0. 297.  16.   2.  13.  27.   5.   4.]
 [  3.  87.  64.  13.  10.   0.   1.   0. 209.  12.]
 [  2.  64.   1. 263.   8.  10.  20.   6.  21.   4.]
 [ 69.  97.   5.  11.   2.   0. 174.   0.  41.   0.]
 [  5.  63.  91.   6.  74.  26.   0. 127.   2.   5.]
 [ 37.  75.  17. 101.  18.   0.  33.   3. 110.   5.]
 [ 17.  10.   6. 251.  19.   0.   1.   7.  31.  57.]]
Accuracy: 0.3043
```



```
Confusion matrix:
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
Accuracy: 1.0000
```

The left side is the confusion matrix for test data, the right side is for training data. From the classified results, it can be seen that the model works perfect on training data but performs very bad on test data which means this model is non-generizable.

9. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**

	Need train	Easy to implement	Depends on initial value	Is efficient on large data
Knn	N	Y	N	N
Single layer NN	Y	Y	Y	Y
Multi-layer NN	Y	N	Y	Y

Knn Pros: This algorithm does not have training stage, it can be use directly based on training data, and it is easy to implement .Knn Cons: But when it comes to big data, it will consume lots of computing resources since every new data should be compared with all the training data.

single layer and multi-layer pros: they are efficient when it comes to big training data since when doing the prediction on new data, they do not need to resort to training data. And multi-layer NN can stimulate most function if there is enough hidden layer. Cons: The initial weights are very important to these algorithms. If the initial weights are 'bad', those algorithms cannot work well which means these algorithms are easily trapped by local optimal value.

10. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**
    1 Maybe scale the data will have a better fitting result.
    2 In the Neural Net algorithms, maybe dynamic learning rate can be implemented. We can use continuous decreasing learning rate to get more accurate results.

11. **Optional task (but very recommended). Simple gradient decent like what you have implemented can work well, but in most cases we can improve the weight update by using more sophisticated algorithms. Some of the most common optimization algorithms are summarized nicely here:**
    https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6
    Implement one or a few different optimizers and compare the speed at which the training converges. A good starting point is to implement momentum gradient decent.