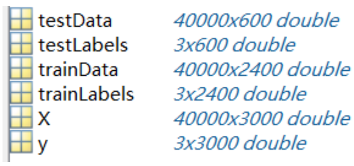
2018/2019 COMP1037 Coursework 2 – Machine Learning

Report

FAI Coursework 2 – Neural Network Classification

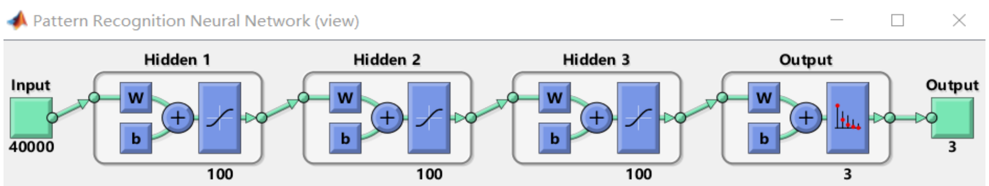
1. dataset structure & network configuration

Dataset structure:

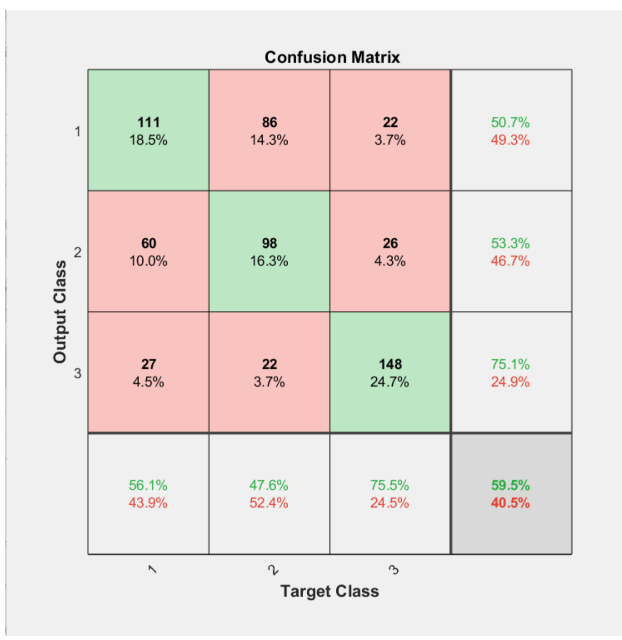


testData	40000x600 double
testLabels	3x600 double
trainData	40000x2400 double
trainLabels	3x2400 double
X	40000x3000 double
y	3x3000 double

Network configuration:



2. Confusion Matrix



The row of the confusion matrix represents 'predict value'. The column of the confusion matrix represents 'true value'. The first three row shows examples which are predicted by nn model as Class1, Class2 and Class3 respectively. The first three column shows examples whose true values are Class1, Class2 and Class3 respectively. The column on the rightmost of the matrix shows the correct and incorrect percentages of all the examples predicted to belong to each class. The bottom row shows the correct and incorrect percentages of all the examples belonging to each class. And the last grid in the matrix demonstrates the total accuracy of nn's prediction.

For example, the first grid in the matrix means: The model predicts 111 data as a 'Cat' of which the true values are 'Cat'. The second grid means: The model predicts 86 data as a 'Cat' of which the true values are actually 'Dog'. Therefore, the green grids represent all correct predictions, and red grids show wrong predictions. The final grid, which show 59.50%, demonstrates the total accuracy of the trained NN model.

As showed in the matrix, the total accuracy of my trained NN model is 59.50%. It does better in identifying the panda, which is probably because dogs and cats have more similar appearance. On the other hand, pandas have more different characteristics compared with the other two animals.

To conclude, 59.50% predictions are correct, 40.50% predictions are incorrect. At this stage, the NN_model does not perform accurately.

3. Parameter Settings

To control the test dataset and parameters as same as possible (for every time of tests), I use the same rand_indices for each independent test. I also use 'control variate method' to test different parameters' impact on performance.

1. **Test for hidden layers**

   size of images: 40*40

patternnet	Min accuracy	Max accuracy	Test times	Average accuracy
25	47.50	52.50	20	49.92
100	49.17	54.00	20	52.72
[100,100,100]	51.00	53.83	20	53.13
[300,300,300]	53.33	59.33	20	56.03
[500,500,500]	53.33	57.83	20	55.47
[100,100,100,100,100]	54.00	58.00	20	56.07
[100,100,100,100,100,100,100]	52.00	56.33	20	54.67

More hidden layers and more neurons on each layer can improve the accuracy. However, excessive layers and neurons are not necessary. Overmuch neurons have little influence on model performance, sometimes may even cause overfitting and reduce the accuracy. Also, implementing more hidden layers and neurons may require longer time for running which decreases the efficiency.

2. **Test for learning rate**

   size of images: 40*40; hidden layer: 100,100,100

learning rate	Min accuracy	Max accuracy	Test times	Average accuracy

0.5	55.83	58.67	20	57.86
0.1	56.00	59.83	20	58.08
0.01	56.17	58.5	20	57.63
0.05	56.33	59.67	20	57.98

In my test result, average accuracy are very closed. This kind of slight fluctuate may be caused by the randomness, which is not enough to prove the relationship between learning rate and performance. Therefore No obvious evidence shows learning rate can affect the performance of the trained model in my experiment. More times of test should be done to make an accurate conclusion.

3. **Test for image size**

Hidden layer: 100,100,100

Image size	Min accuracy	Max accuracy	Test times	Average accuracy
1*1	40.33	48.83	30	43.36
40*40	51.17	59.83	20	56.42
100*100	50.33	59.50	20	55.84
150*150	49.50	58.89	20	53.18
200*200	49.83	58.17	20	52.42

My test result shows that smaller image size can lead to a more accurate model. Smaller image size which has smaller dimension, also has a faster training speed. However, a too small image size will perform worse. It may because of that computer can not identify the image clearly.

4. **Performance function**

For ‘mse’ and ‘msereg’ ‘mae’, ‘mse’, ‘sae’, ‘sse’, my test result didn’t show obvious evidence that those two different functions have observable impact on accuracy.

5. **Gradient function**

size of images: 40*40; hidden layer: 100,100,100;

gradient function	Min accuracy	Max accuracy	Test times	Average accuracy
'trainoss'	51.87	59.17	20	55.45
'trainscg'	51.33	58.83	20	54.67

No obvious result show close relationship between gradient functions and the performance.

4. Efficiency

1.Data Normalization

If the original data vary widely, Normalization can reduce the data difference, and all data will be normalized to the same level of magnitude. The running time slightly decreases after implementing the normalization.

```
X1 = mapminmax(X, 0, 1);
```

2. PCA

The thought of PCA is to gain main features of samples, and to project the sample data into a new space which has a lower dimension to realize the dimension reduction.

It converts multiple data into a few comprehensive data and calculates the least dimension to preserve the most important features.

In this way, PCA can help decrease the running time and denoise the data which may even increase the accuracy.

Here is my code:

The PCA code is in the comment of nn_animals.m. If you want to run it, you can just delete all the sign '%'.

```
%% PCA
X2=X';
[COEFF,SCORE,latent] = pca(X2);
total_sum=0;

for i= 1:length(latent)
total_sum=total_sum+latent(i);
end
sum_k=0;i=1;
while ((sum_k/total_sum)<0.9)
    sum_k = sum_k + latent(i);
    i=i+1;
end
P=COEFF(:,1:i-1);
X3=X2*P;
X=X3';
```

6. CNN

I have put a file named cnn inside the CW file.