CS5014 P2 Report

200009419

Introduction

The project aims to apply machine learning on recent histograms for object detection problem. The project uses SVM (Support Vector Machine) and NN (Neural Network) approaches to deal with the problem. For the dataset, there are two groups of label needs to be classified—color and texture. Each group of labels are dealt with separate trained model. The training process uses cross validation as its evaluation method. During the training process, there are different models trained with different hyperparameters. The model with highest accuracy is used for prediction of test data.
It offers two versions, the complete one—main.py will try parameter optimising and requires approximately 30 minutes to run. The quick-run version — quick_run.py already selected parameters with relatively good performance and could complete the training process within 2 minutes. The report is based on the result retrieved from the complete one.

Data Preparation

For convenience of fitting, the original data are split into record and labels for colors and textures. To create clean data, arbitrary data such as "id", "size" columns are dropped. Empty records are filled with default value 0. Columns of labels for "color" and "texture" are extracted into individual files. For convenience of training, the original value is mapped to integers. For instance: 'white' is represented by "1", brown is represented by "2".
To sum up, there are four .csv files for pure training records (train.csv), pure labels of color for training records (color_train.csv), pure labels of texture for training records (texture_train.csv), pure testing records (test.csv), and two .txt files for color (color-key.txt) and texture key-value (texture-key.txt) dictionaries.
Before applying the classifier, the data are scaled with MinMaxScaler known as normalisation, that rescales the data set such that all feature values are in the range [0, 1]. Scaling can be important that it speeds up gradient descent by making it require fewer iterations to get to a good solution. It is also important to ensure that all the data are appeared in form of float. Thus, data in string form must be encoded.

Training Process

SVM

A support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification. It is effective in high dimensional spaces. The task in the project is multi-class classification. SVC in the project implement the "one-versus-one" approach for multi-class classification. In total, n_classes * (n_classes - 1) / 2 classifiers are constructed and each one trains data from two classes.

The programme uses Radial Basis Function (RBF) kernel SVM and tries different gamma and c values to find best model. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. While the c parameter trades off correct classification of training examples against maximization of the decision function's margin. It behaves as a regularization parameter in the SVM. For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy.

The programme will use cross validation as a technique for evaluating ML models. In detail, it will divide the training set into ten sections. Each validation process will pick 1/10 of the data as evaluation data and train with the rest. The performance of the model in each validation process is measured by accuracy.

The programme uses hyperparameter tuning to optimise the model. It is implemented in a brute-force approach. In the programme, there is a list to store the candidate parameter for gamma and another one for c. Every combination of gamma and c will form a case of trained model. The programme will pick the model with the best performance i.e., the model with highest mean accuracy score in cross validation. Thus, the programme would pick the parameter pairs with best performance.

NN

The programme uses a Multi-layer Perceptron bases model to train the NN classifier. The network consists of an input layer with the number of nodes equals to the number of features, an output layer with the number of nodes equals to the number of classes. Between the input and output layers, there can be one or more non-linear layers, called hidden layers that can be modified. The module contains the public attributes coefs_ and intercepts_. coefs_ is a list of weight matrices, where weight matrix at index i represents the weights between layer i and layer i+1. intercepts_ is a list of bias vectors, where the vector at index i represents the bias values added to layer i+1. MLP is capable of learning non-linear models and dealing with complicated classification problems.

The training process for NN approach is similar to SVC including hyperparameter tuning and cross validation. It uses ReLU as its activation function in default while parameters hidden layer size and learning rate are modified. The model will pick the parameter pairs with the best accuracy as what is implemented in SVC.

The learning rate may be the most important hyperparameter when configuring neural network. The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.

Layer size is also important for training. Increasing layer size could probably increasing the accuracy of the model. However, increasing the number of hidden layers much more than the sufficient number of layers will cause accuracy in the test set to decrease by introducing overfitting that it will learn the training data, but it won't be able to generalize to new unseen data.

Output

The programme will print the predicted result for both color and texture with model trained by SVM and NN separately. The predicted result is shown in key and requires further transform to become values. The dictionaries are provided in .txt files. The output .csv files can be found in "multiclass" folder. The first row is the index for testing data, the second row is predicted result from SVM while the third row is predicted result from NN.

Evaluation

Accuracy

The table below indicates both SVM and NN model works on input training set. The accuracy after training by the whole training set is much higher than mean accuracy for cross-validation. It may indicate that there are too few training records for cross-validation. The performance of both models may be improved if a larger training set is available.

	Total Accuracy for Color	Total Accuracy for Texture	Mean Cross-Validation Accuracy for Color	Mean Cross-Validation Accuracy for Texture
SVM	0.8299	0.7071	0.5064	0.5333
NN	0.7154	0.7963	0.4434	0.4652

Table1: Training Result

Issues Requires Further Researches

The accuracy for the model needs to be improved. Several factors need to be considered to obtain a more precise model. Firstly, there are relatively large classes with more than ten labels to be classified. Compare to binary classification problems, it is much more complicated to be dealt with. Secondly, the records for each class is severely imbalanced. For instance, there are more than 400 records for "white" while less than 40 for "pink" in training set. It is important to treat imbalanced data.

How Parameters Affect Accuracy

The figure below illustrates the relationship between gamma, c value and mean cross-validation accuracy for SVM. It indicates that a small gamma value could improve the accuracy while c value may have small effect in this model.

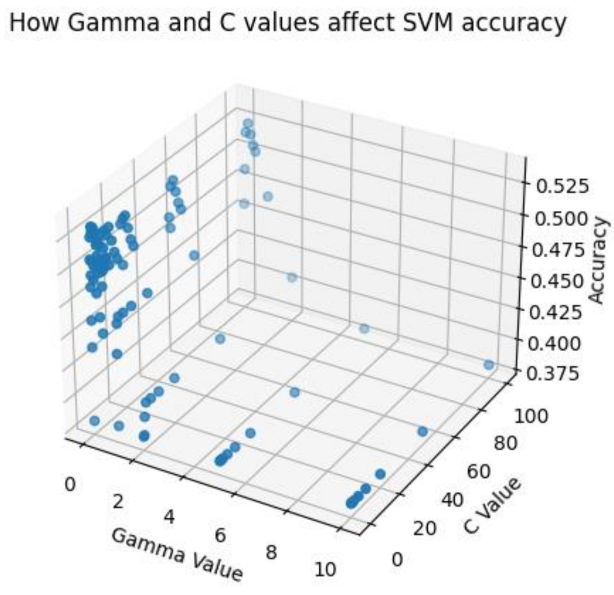

Figure1: How Parameters Affect SVM

The figure below demonstrates the relationship between layer size, learning rate and accuracy for NN. It indicates that a layer number between 25 to 100 is suitable for the model. A number of layers above 100 is too large for the model. In addition, a low learning rate is preferred for the model

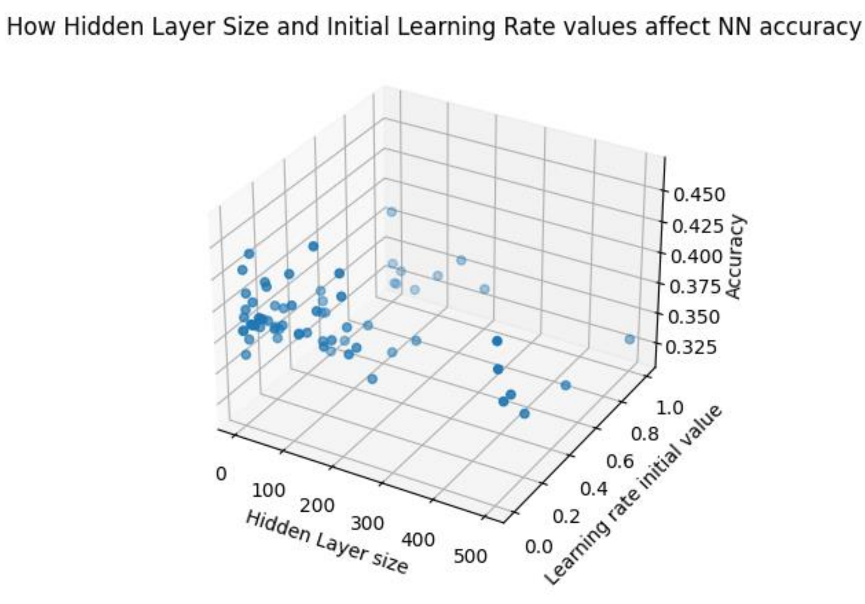

Figure2: How Parameters Affect NN

Testing Screenshot

Loading data

```
Reading Multiclass data...
[[0.85885474 0.87636946 0.74306731 ... 0.73595022 0.40873006 0.59973717]
 [0.46969481 0.60644349 0.586488   ... 0.48949047 0.5943794  0.48692472]
 [0.64271217 0.61451781 0.62412946 ... 0.56290543 0.57487351 0.5904085 ]
 ...
 [0.52061803 0.23428057 0.31195553 ... 0.56662353 0.48496293 0.61164693]
 [0.25061368 0.14056064 0.16250678 ... 0.63861675 0.64996056 0.62888097]
 [0.74082457 0.56123785 0.45424958 ... 0.5353737  0.83458652 0.66234729]]
      0
0     1
1     1
2     2
3     2
4     1
...  ..
1330  1
1331  3
1332  3
1333  2
1334  4

[1335 rows x 1 columns]
[[1.78229607e-01 7.76228039e-01 4.12106164e-01 ... 5.94266458e-01
  4.92783216e-01 5.81549768e-01]
 [1.86509935e-02 1.38836510e-04 3.09937976e-02 ... 5.88030935e-01
  8.03346808e-01 6.80748682e-01]
 [4.32853530e-01 3.45135750e-01 5.14200641e-01 ... 6.27395209e-01
  4.74637857e-01 5.99444328e-01]
```

Scoring final picked model

```
Scoring NN...
Total score on trained data for color:
Score: 0.7153558052434457
Cross validated score of the classifier for color:
All scores:
[0.39552239 0.49253731 0.47014925 0.43283582 0.45522388 0.46616541
 0.42857143 0.44360902 0.42857143 0.42105263]
Average score:
0.44342385815284474
Total score on trained data for texture:
Score: 0.7962546816479401
Cross validated score of the classifier for texture:
All scores:
[0.46268657 0.48507463 0.41044776 0.47014925 0.42537313 0.45864662
 0.4887218  0.5112782  0.46616541 0.47368421]
Average score:
0.4652227583885086
============================================
```

Prediction result

```
Making predictions...
Making predictions for color
NN Predicted:
[3 6 3 3 3 3 3 3 3 1 1 1 2 3 6 1 3 3 3 1 3 3 3 3 3 2 2 3 1 2 1 1 3 3 1 1 1
 2 3 3 3 3 3 2 6 1 3 1 3 6 3 3 3 1 1 3 1 3 1 1 3 3 3 1 3 6 3 2 1 1 1 3 1 2
 3 1 1 1 1 3 3 1 1 3 3 1 2 1 3 5 3 5 3 1 3 3 1 2 3 3 2 3 3 3 3 6 2 1 3 3 3
 1 3 1 3 1 3 6 3 3 1 3 1 3 2 2 1 3 3 3 3 1 2 3 1 3 2 3 3 2 1 3 7 3 1 3 3 1
 1 1 3 3 2 1 3 3 3 1 2 1 3 1 1 6 3 3 3 3 3 3 4 3 1 3 1 3 1 3 3 3 3 2 3 1 1
 2 3 1 3 3 3 1 3 3 1 3 3 1 3 1 3 3 1 3 3 1 3 3 3 2 1 3 3 3 3 2 3 3 1 3 3 2
 1 1 3 5 3 3 3 9 6 1 1 3 3 3 3 3 1 1 3 1 3 1 8 3 3 3 3 2 2 3 3 3 1 1 3 2 3
 1 3 3 3 1 3 1 1 3 3 3 3 1 1 3 2 1 1 2 3 3 1 1 1 3 1 1 1 3 3 1 1 1 3 6 3 2
 3 3 1 3 2 3 1 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 6 2 3 1 3 1 3 1 3 1 3 1 3 3 1
 3]
SVC Predicted:
[1 1 1 3 1 1 3 3 1 1 1 1 1 1 1 1 3 1 3 1 1 1 1 1 3 1 1 1 1 1 1 1 1 3 1 1 1
 1 1 1 1 1 3 1 1 1 1 1 3 1 1 1 3 1 1 3 1 1 1 1 3 2 3 1 3 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 3 1 1 1 1 3 1 1 3 1 1 1 1 1 1 3 1
 1 1 1 3 1 3 1 1 1 1 3 1 3 3 1 1 1 1 1 1 1 1 1 1 3 1 3 1 1 1 1 1 1 1 1 1 1
 1 1 3 1 1 1 3 3 1 1 3 1 1 1 1 1 1 1 1 1 2 1 1 1 1 3 1 3 1 3 1 3 3 1 1 1 1
 1 1 1 3 3 1 1 3 1 1 1 3 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 3 5 1 1 3 1 2 1 1
 1 1 1 1 3 3 1 1 1 1 1 1 1 1 3 1 1 1 3 1 1 1 1 3 1 3 1 1 1 1 1 1 1 1 1 1 3
 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 3 3 1 1 1 3 1 1 1
 1 1 1 1 3 1 1 1 3 1 1 1 1 1 3 1 3 1 1 1 1 1 1 1 3 1 1 1 1 1 3 1 1 1 1 1 1
 1]
Making predictions for texture
NN Predicted:
[2 1 2 2 1 2 2 2 4 3 1 6 1 2 8 1 2 2 2 1 2 2 3 8 2 1 6 2 1 6 1 2 2 2 1 1 2
 6 2 8 8 2 2 6 1 1 2 2 2 1 2 2 2 1 2 2 1 2 1 1 2 2 2 2 2 1 4 2 1 2 1 2 1 1
 2 1 2 1 2 2 2 2 3 2 2 1 2 1 2 8 2 8 2 2 2 2 1 1 8 2 6 2 3 8 2 1 6 1 1 2 8
 1 2 1 2 1 2 1 1 3 1 2 2 2 2 6 2 2 4 3 2 1 1 2 1 2 6 2 8 2 1 6 3 2 1 2 2 1
 1 1 2 2 1 1 2 2 2 1 1 2 2 8 3 1 2 2 2 2 6 2 4 2 1 2 2 2 1 2 2 2 2 1 2 1 1
 6 2 1 2 2 2 2 2 1 1 2 2 1 2 1 2 2 1 2 2 4 2 2 1 2 1 2 2 2 8 2 2 2 1 6 2 4
```