Dharmit Prajapati
1219597132

Fundamentals of Statistical Learning Project2 Report

The dataset has 50 different categories/classes into the which the data samples can be categorised. Each sample (in training and test set) has 3 feature matrices (X1,X2,X3) and corresponding category Y (we have flattened the labels in the project). The training set has 4786 samples whereas testing set has 1833 samples.

Step0)

Part1:

In this part of step0 we have to train SVM classifier with each of the feature separately (ie X1,X2,X3 separately). We then use the trained model to predict the categories of test data (again by using each feature separately). We used the svm_train to train the model and svm_predict to predict using the trained model.

For this part, the penalty parameter is set to ‘c 10’ and linear kernel ‘-t 0’

c parameter indicates how much misclassification should be avoided for each training sample. Large values of c will choose a small margin hyperplane if that hyperplane does a better job of classifying all training points correctly. Small values of c will choose a larger margin even if the classifier misclassifies the points.

t 0 indicates linear kernel

Model training example:

model_step0_que1_case1 = svm_train(y_train_flatten,x1_train, '-c 10 -t 0')

Model Prediction example:

p_label_step0_que1_case1, p_acc_step0_que1_case1, p_val_step0_que1_case1 = svm_predict(y_test_flatten,x1_test, model_step0_que1_case1)

Accuracy(~8-17%):

```
Accuracy for Step 0 part 1 and feature vector X1:11.364843335103558
Accuracy for Step 0 part 1 and feature vector X2:17.525225703664365
Accuracy for Step 0 part 1 and feature vector X3:8.603292618162508
```

Part2:

In this part of step0 we have to train SVM classifier with each of the feature separately (ie X1,X2,X3 separately). We then use the trained model to predict the categories of test data (again by using each feature separately). We used the svm_train to train the model and svm_predict to predict using the trained model.

For this part, the penalty parameter is set to ‘c 10’ , linear kernel '-t 0’ and ‘-b 1’ option.
By using the ‘-b 1’ option we can obtain the posterior probability $p_k(w_i|\mathbf{x})$ (k indicates the Feature vector ie X1 or X2 or X3 and w_i indicates the class/category)

Model training example:

model_step0_que2_case1 = svm_train(y_train_flatten,x1_train, '-c 10 -t 0 -b 1')

Model Prediction example:

p_label_step0_que2_case1, p_acc_step0_que2_case1, p_val_step0_que2_case1 = svm_predict(y_test_flatten,x1_test, model_step0_que2_case1,'-b 1')

Accuracy(~27-29):

```
Accuracy for Step 0 part 2 and feature vector X1:28.624535315985128
Accuracy for Step 0 part 2 and feature vector X2:27.82793414763675
Accuracy for Step 0 part 2 and feature vector X3:28.73074880509825
```

Step1)

From Step0 part2 we get the probabilities $p_k(w_i|\mathbf{x})$ which indicate the probabilities (50 probabilities as we have 50 classes) of each sample of feature vector $\mathbf{X}_k$.

The task was to take the average of $p_k(w_i|\mathbf{x})$ for $(1 \le k \le 3)$ for each sample $\mathbf{X}_k$.
This will give us 50 probabilities for each sample and we have to choose the max of these 50 probabilities.

Logic:

length1 = len(p_val_step0_que2_case1) #no of test samples =1833
length2 = len(p_val_step0_que2_case1[0]) #no of classes=50
predicted_class = []
for i in range(length1):
avg=[] #initialize avg as empty list for each sample
for j in range(length2):

avg.append((p_val_step0_que2_case1[i][j]+p_val_step0_que2_case2[i][j]+p_val_step0_que2_case3[i][j])/3) //ca
max_class = np.argmax(avg) #take the index of the class with max probability
predicted_class.append(max_class+1) #add the class with max probability (classes are labelled as 1,2…50 hence max_class+1)

Then we compare the predicted_class with test_labels and find the accuracy

Accuracy(~45):

```
Accuracy for Step1: 45.72490706319702
```

We can see that the accuracy is much better than the accuracy of step0 part2. We can observe that taking the maximum probability gives us better accuracy for the given dataset.

Step2)

In this step, instead of taking each feature separately we concatenate them and now each sample can be considered in a 3-d feature. We then train(using svm_train) the model on this concatenated data and use the same model to predict(using svm_predict)

For this part, the penalty parameter is set to ‘c 10’ and linear kernel ‘-t 0’

Model training example:

model_step2 = svm_train(y_train_flatten,concat_train_x, '-c 10 -t 0')

Model Prediction example:

p_label_step2, p_acc_step2, p_val_step2 = svm_predict(y_test_flatten,concat_test_x, model_step2)

Accuracy (~39):

```
Accuracy for Step2: 39.19277748274031
```

The above accuracy is more than any of the 3 accuracies calculated in step0 part1.We can observe that the accuracy has increased with increase in dimensionality for the given dataset.