Kaggle Competiton Report

ZHE TANG 746071

(All data included are in the Appendix on page 2.)

In Kaggle Competition, I tried several methods to decide which one I would use. I split the data into two parts: train set and test set. Firstly, based on the result of project 2(Table 1) and some methods included in our project 2, whose results shown by Table 2, the error rate of polynomial kernel is still the least. In this case, I predict the test set by polynomial kernel. After that, I have tried KNN and Random forest algorithm learned from the lecture and upload the result. However, the private score of these two are 0.94400 and 0.82000 separately. Moreover, I have tried CNN from keras package and OneVsRestClassifier. However, these methods cost too much time to train. Until epochs reach 15, the accuracy of CNN is only 0.78. I plan to upload my CNN code to a cloud server to run. In that case, I choose the polynomial kernel method.

The method that I use is C-Support Vector Classification with polynomial kernel to predict the test set. Firstly, I import svm package from sklearn. Then I use train set with label to train SVC(kernel='poly', degree=6, C=5, coef0 = 0) model. After training the model, I use the trained model to predict test set.

In this model, there are three main parameters. One is C, which is an error term's penalty parameter. Another is d, which is the degree. Also, coef0 is Independent term. I run two projects to decide the value C and coef0. The results of these two projects are total same, which are shown in Chat 1. From chart 1. Because the range of c is (1, 21, 2), the values of C and coef0 in the Chart do not affect the error rate. Also, I try the project to decide d. I launched three projects, whose value of d is 4, 5 and 6 respectively. The predicts of test set when d = 4 and 5 are almost same. Although the predicts of d = 5 and 6 are not the same, the score between these two are the same. Therefore, I use d = 6, C = 5 and coef0 = 0.

Appendix

Method(perceptron)	Train error	Heldout error
Linear Approach	0.00888888888889	0.04
Basis Expansion	0.0844444444444	0.106666666667
Linear kernel svc	0.146666666667	0.2
polynomial kernel svc	0.0222222222222	0.0266666666667
rbf kernel svc	0.466666666667	0.506666666667
MDS	0.146666666667	0.346666666667
PCA	0.12	0.306666666667

Table 1

Method(perceptron)	Train error	Heldout error
Linear Approach	0.232495238095	0.245028571429
Basis Expansion	0.900380952381	0.8976
Linear kernel svc	0.0	0.274971428571
polynomial kernel svc	0.0	0.0675428571429
rbf kernel svc	0.0	0.904457142857
DecisionTreeClassifier		0.23897143

Table 2

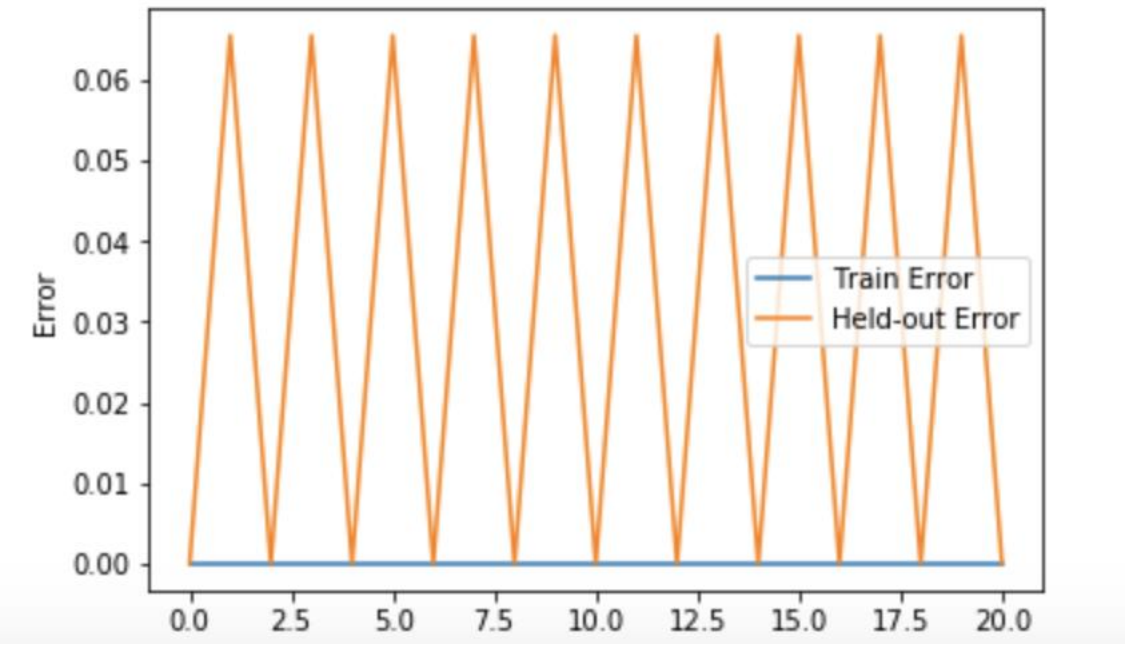

Chart 1