COMP9318 Project Report

z5146418 Yuchen Yan

1. Introduction.

In this project, we should implement a algorithm in order to fool a classifier called target classifier. As a result the classifier should misclassified the test data from class 1 to class 0. In this project I learnt a lot of related knowladge including the concept about support vector machine, the mathematic background about svm, and also the python implementation about the svm. I want to talk about them one by one in this project report.

2. Background knowledge

(1) What is svm?
There can have a lot of solutions for one classifier, but there is only one optimal hyperplane. So the vector support machine is basically use the support vectors to find the optimal classifier. The optimal classifier can maximizes the distance between the hyperplane and the "difficult points" close to decision boundary.

(2) Linear SVM:
Linear svm is the most common and simple SVM, it means that the datasets given can be separable by a line. According to the formula provided by lecture (f(x) = sum(ysv * asv * <Xsv, X> +b). We only need to calculate the inner product between the test point x and the support vectors xi. And then we have the value f(x), according to the f(x) we can find out the target about the classifier. This fairly simple.

(3) Soft margin classification:
In this case, in order to solve the problem that there are some points be moved to where they belong. If the training data is not linearly separable, slack variables ξi can be added to allow misclassification of difficult or noisy examples. But the main job of this classification is the same as the linear SVM. The only difference of the process of linear classifier and soft margin classification is the training process, a slack variables ξi needed be added. But finally the function got have the same format with the linear SVM.

(4) Non-linear SVMs:
This classifier deal with the situation that the datasets are not separable. Sometimes the datasets can't be separated by a line. In this situation, we can map the data into a higher dimensional space, which the data can be separable. Most of time it is very hard to find out the mapping function of the data. So in this situation we can use the 'kernel tick'. k(xsv, x) = xTx. We can input the vectors and it will give us the inner product of the vectors after the 'mapping'. As a result, we don't need to bother to find the mapping functions, which is very hard. The common kernels that we have. Already have are linear, polynomial, radial basis function and also sigmoid.

Radial basis function:
It is a function that the value base on the distance between the two input vectors. It is very important concept which can Map the original features into infinite dimensional space.

3. Implementation

In this part I will introduce the process that I implement this project using python.

(1) Structure of my project:

Firstly, I input the class0 and class1 data for model training. I find out the vectorizer of each samples and its target. And then I use the grid search method to find out the best parameters for the SVM. It should find out which kernel function should use. And also the corresponding parameters about the kernel. Then I use the parameter just find out and the training data to train the model.

(2) The library I used:

numpy:

Here I use the numpy to created the array in order to contain the training data sets and test data sets.

CountVectorizer:

This function also belong to sklearn.feature_extraction.text, I input a list of test data samples and output a matrix which has the share of (n_samples, n_features). It contains the feature frequency of each samples. Corresponding code are below:

```
    class_0 = [' '.join(i) for i in strategy_instance.class0]
    class_1 = [' '.join(i) for i in strategy_instance.class1]
    class_all = class_0 + class_1
    vectorizer = CountVectorizer()
    count = vectorizer.fit_transform(class_all)
```

train_test_split:

This function belong to sklearn.cross_validation. It can split the data into test data and also training data. Corresponding code are below:

```
     X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.5, random_state=0)
```

GridSearchCV:

This function belongs to sklearn.cross_validation. We can input the parameters of each kernels in a dictionary and also the training datas. Then the function can help us to find the best parameters of all. Corresponding code are below:

```
     param_grid = [{'kernel': ['rbf'], 'gamma': [0.01],'C': [1, 10, 100,
1000]},{'kernel': ['linear'], 'C':[1,10,100,1000]}]

    grid_search = GridSearchCV(svm.SVC(),param_grid,cv=5)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.5, random_state=0)
    grid_search.fit(X_train,y_train)
```

(3) Finally, I changed the features of the test_data. In order to let the classifier to misclassify the test_data from class1 to class0.

I tried several different method to change the features of the test data. For example I delete the highest 15 frequency of each sample, and add 5 different features by 10 times for each one.

(4) I test the mis classification-rate of the data in my computer use the classifier that I predicted first, then I use the submission system for the project to evaluate.