COMP9318 18 S1 Project Report

Haoxiang Zhao z5084093
Tianwei Zhu z5140081

Abstract

This is a report for project of COMP9318 18 S1. There are three parts in this report. First section is a introduction of this project (include demands and constraints), second part is our method to resolve problems and results derived from our program, last one is some conclusions of this project.

Introduction

In this project, we aim at designing a program to fool a binary classifier which belongs to SVM, in other words, the main target is miss-classifying the test-dataset after training. Thus, at beginning, we should train this binary classifier with two training sets named class-0.txt and class-1.txt, and then, modify each test sample in test-dataset twenty times by adding, deleting and replacing operations. There are 2 rules we need to follow: one is we must use training-datasets in training part and another is making sure twenty modifications exactly (no more and no less).

Methodology and results

Method:

There is a useful tool we should know before starting training, that is a build-in function in skrlearn-SVM called coef_. We can find an introduction of this from skrlearn website.

coef_ : array, shape = [n_class-1, n_features]

Weights assigned to the features (coefficients in the primal problem). This is only available in the case of a linear kernel.

coef_ is a readonly property derived from dual_coef_ and support_vectors_.

Recall that in linear SVM, the result is a hyperplane that separates the classes as best as possible. The weights from svm.coef_ represent this hyperplane, by giving the coefficients of a vector which is orthogonal to the hyperplane.

We will use this to modify file later.

Training:

So, we use linear as kernel function which is only affected by penalty parameter C of error term. Although the accuracy of training is higher with larger C, because generalization is decreasing when C is increasing, overfitting may happen in this case which means miss-classification would occur when we train with a high C, thus we choose C = 0.8 to make sure avoiding overfitting reasonably.

Testing and modifying:

The svm.coef_ contains direction which gives us the predicted class, so we can tell on which side the feature is. So, we could say that the absolute size of the coefficient relative to the other ones gives an indication of how important the feature was for the separation. With the help of coef_, we can get the weights of each features(words) which are plus or minus. For example, the word “world” has weight of 1.3, then it contributes classification to class 1.

After training, we can delete exactly 20 words for each line in test file, those words have top 20 positive weights in coef_.

The picture below is our classification results before and after twenty exact modifications:

```
gama:3, C:0.8, kernel:linear, degree:2, coef0:0.8
result:  0.43
[1 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 0 1 1 1 1 0 0 0
 0 0 1 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 1 0 1 1
 0 1 1 0 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 1 0
 0 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0 1
 1 0 0 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0]
Modifying file...
gama:3, C:0.8, kernel:linear, degree:2, coef0:0.8
result:  0.0
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

We can find that there were zero vector in class 1 and the final result is 0% after modifying file.

modified_data.txt|submission.py	2018-05-24 14:04:08	Success = 86.500 % (Plz make sure that you do not use test data for inference)

Conclusion

We finally fooled classification function to misclassify the test file with using weights of support vector. This project shows us how SVM works and we also learned practical skills on applying SVM.