COMP 9318 Data Warehousing and Data Mining 18S1

Final Project Report

Group Name: ChenAndWu

Group Members:

Zeshi Wu z5102175

Keshi Chen z5142821

Project Description:

In this project we need to design and implement an algorithm to "fool" (i.e. let the target classifier mis-classifies class "1" to class "0") the target classifier which belongs to SVM family by modifying the test data, with exactly 20 distinct tokens each sample.

We are given two training data sets (360 samples of class "0" and 180 samples of class "1") and one testing data set with 200 samples of class "1" in it. The data sets consist of short paragraphs of news. So we use "bag-of-words" model to represent them. The bag of words has 5720 features in total.

Our algorithm is based on selecting the best features in class "1" by finding top 20 features assigned with highest weights (w_i in **w** in SVM classifier) in each test-sample and delete them.

The score of success %-age is 85%.

Characteristics of SVM Classifier:

Support Vector Machine (SVM) maximizes the margin around the hyperplane which seperates samples into different classes.

The function of SVM classifier can be represented as follow:

$$f(\mathbf{x_i}) = \text{sign}(\mathbf{w}^T\mathbf{x_i} + b)$$

The functional margin of $\mathbf{x_i}$ is:

$$y_i (\mathbf{w}^T\mathbf{x}_i + b)$$

The distance from example to seperater:

$$r_i = y_i (\mathbf{w}^T\mathbf{x}_i + b) / ||\mathbf{w}||$$

where $\mathbf{w}$ is the normal vector of decision hyperplane, $\mathbf{x_i}$ is data point i, y_i is the class of data point i, which can be either +1 or -1.

And $\mathbf{w}^T\mathbf{x} = \sum_{i=1} w_i x_i$.

It can be easily seen that the distance r is decided by the cumulative sum of weights of features multiplied by the value of corresponding feature. Since, when predicting, the weights are derived from trained knowledge, while feature values are given by test data, we can simply modify best-selected feature values (in our case, we set them to zero) to move the data point toward the other side of the hyperplane, leading to mis-classification.

Our Algorithm:

fool_classifier(test_data):

Input: test_data

Output: modified_data

(1) X_train = build_bag_of_words(training_data_0 + training_data_1)

(2) X_train = count_vectorizer(X_train)

(3) parameters = GridSearch(svc, C=[1...200, step=5]) . best_params

(4) clf = train_svm(parameters, X_train, y_train)

(5) tokens = tokenizer(test_data)

(6) **weights = clf.coef_** // To get weights of each features in bag_of_words

(7) tokens_weights = Find weights for each token in tokens, if token is not in bag_of_words, set its weight to 0

(8) Sort tokens' weights, in descending order // Features spport class "1" have positive weights, so we sort them out

(9) **threshold = tokens_weights[21] // We only delete features with weights larger than the threshold, in order to restrict the modification times to be exactly 20**

(10) for line in tokens:

for token in lines:

if token not in bag_of_word or tokens_weights[token] < threshold:

modified_data += token

modified_data += " "

else:

skip the token, don't add it to modified_data, which means **delete** it from test_data

modified_counter += 1

if modified_counter >= 20:

break

If 20 distinct features modified, skip rest of features in the current line, then check the next line

while modified_counter < 20:

word = "abcdefg"

number = 2018

while word+str(number) in line:

number += 1

add word+str(number) to the sample

modified_counter +=1

number += 1

(11) return modified_data

Kernel and Parameters Chosen:

In this project, we use linear kernel as it is commonly used in bi-class text classification tasks and performs well.

We search for best value of parameter "C" by using grid search, from the range 1 to 200 with step-space 5.

Performance:

The score of success %-age we get from feedback is 85%, which means our approach is fairly efficient.