COMP9318: Project Report
Group: Vivid_Sydeny

Support Vector Machine Classifiers

Abstract

A linear SVM classifier was trained from sets of paragraph samples of two classes using “sklearn” library in Python3. Bag of words model, which mapped words to their frequency in the class, was used to train the classifier. For each word in the sample of the test data provided, the classifier mapped the word to its weight it had learnt during training. The weights of the words in the sample of test data was used to modify each sample paragraph by 20 tokens to fool a hidden SVM “target-classifier” that had been trained using more samples of each classes.

Introduction

A Support Vector Machine (SVM) is a machine learning algorithm that maps training data to a higher dimension. Then, SVM finds the hyperplane that separates the sets of data in that higher dimension with the help of support vectors. SVM’s are very accurate to model complex hyperplanes that are farthest from training data points and are also less susceptible to overfitting, however, they do not have the best time complexity (HK00 Pg.337).

COMP9318: Project Report
Group: Vivid_Sydeny

In our project, a linear SVM classifier was used to classify text based training data to two classes and the weights obtained from the trained SVM classifier was used to modify the samples from the test data to fool the hidden classifier.

Methodology and Justifications

Firstly, a bag of words model was used to map each unique feature in a class to its total number of occurrences inside that class. The bag of words for each class was stored in a python3 dictionary. After computing the bag of words for each sample, they (dictionary) were collectively transformed using the "DictVectorizer" function which was imported from "sklearn.feature_extraction " library. Likewise, the corresponding list of classes for each test sample' s dictionary was transformed using "LabelEncoder()" from "sklearn.preprocessing" library. Those two transformed data sets were then used to train our linear SVM classifier. Thereafter, a dictionary was created that mapped each feature from both the training sample texts to its

corresponding weight. The weights were obtained from the "coef_" attribute from our trained classifier.

Again, a bag of words model was used for each text sample from the test data, where each feature in the sample was mapped to its corresponding weight which was obtained from our training classifier. Then, the dictionary of bag of words was sorted in decreasing order of the weights for modification.

Since, $y=\vec{w}^T\vec{x}+b$, where, the weight vector, $\vec{w}=\sum \alpha_i y_i \vec{x}_i$, in which non-zero α_i values represent support vectors $\vec{x}_i$. We know that higher weighted words are classified into class 1, thus, we deleted the highest 20 positive weighted words in each sample to fool the target classifier. If the number of highest positive weighted words in a sample was less than 20, then we added the required number of lowest negative weighted words obtained from training data to complete 20 tokens.

Results and Conclusion

Finally, the target-classifier misclassified 89.5% of the modified samples as class 0. Therefore, our trained linear SVM classifier, was fairly accurate to fool the

target-classifier misclassify sample texts that would’ ve initially been classified as class 1. This also shows that the support vectors found by the training classifier was precise enough to correctly classify samples of test data. We might have achieved a higher success rate had we used the “Kernel Trick” that expands the feature space.

Works Cited:

(HK00) Han, Jiawei, and Micheline Kamber. Data Mining: Concepts and Techniques. 2nd ed., Elsevier/Morgan Kaufmann, 2012.