Coursework - 2: Sentiment Analysis

CM50265: Machine Learning 2

Group ID: 32

Deadline: April 29, 2022

Contents

List of Figures

1 SVM classification

1.1 Introduction

For the first task in this assignment, we utilise SVMs (Support Vector Machines) for the problem of solving an NLP (Natural Language Processing) task of sentiment analysis.

We have been provided a portion of the IMDB (Needham, 1990) dataset, consisting of 1500 movie reviews. Each movie review has an associated 'positive' or 'negative' sentiment, which our models will predict.

In order for our SVM model to achieve high accuracy with non-linear data, we explore a number of different kernels for transforming the data to a different dimension, for separating the two sentiment classes via a hyperplane. We have used kernels such as, the RBF kernel, Sigmoid kernel and our own custom kernel implementation, the Gram kernel.

1.2 Evaluation and comparison of 3 kernels on the train and test dataset

The training data first passes through the pre-processing stage, where they are converted into data-points for training, called the bag-of-words. The bag-of-words features are reduced to only include the useful information, where the data-points are filtered through 1 of the 3 kernels during the training of our SVM model, to find a hyperplane that separates the now higher-dimensional data-points.

The first in-built kernel that we used was the popular RBF (Radial basis function) kernel (Pedregosa et al., 2011). This function resembles a Gaussian distribution and has the advantages of the K-nearest neighborhood algorithm, suitable for non-linear datasets. Prior to using cross-validation, the RBF kernel with our SVM model attains an accuracy of 84.26% on the test dataset. On the training dataset, the model obtains a greater accuracy of 96.98%, indicating large overfitting.

For cross-validation, we optimise the regularisation parameter "c-value", where the higher the "c-value" the greater the tolerance of misclassified data-points. The second hyper-parameter we optimise is "coef0", which performs offsetting of the data. After cross-validation, we obtained the optimised parameter "c-value" = 1. However, the parameter "coef0" showed no significant performance changes. Our model achieves an accuracy of 83.9% on the test dataset and 95.08% on the training dataset. The optimisation of our hyper-parameters for the RBF kernel from cross-validation, indicate overfitting of the hyperplane and there is also a decrease in both the testing and training accuracies.

We then applied the Sigmoid kernel, the second in-build kernel function (Pedregosa et al., 2011). This kernel function is equivalent to the Sigmoid function used as an activation function for neural networks, a 2-layer perceptron. With this kernel, our model achieves an accuracy of 74.3% on the test data and an accuracy of 78.54% on the training data before cross-validation, with slight overfitting.

The sigmoid kernel has the hyper-parameters "c-value" and "coef0" similar to the RBF kernel, where "coef0" did not affect the performance. After performing the cross-validation and with the optimised hyper-parameters of "c-value" = 0.5, our model achieves an accuracy of 82.13% on the testing accuracy and 83.86% on the training accuracy, indicating a negligible amount of overfitting. Compared to before cross-validation, the model obtains a higher accuracy after cross-validation with optimised hyper-parameters, resulting in an improved fit of the hyperplane for separating the sentiments.

When comparing the accuracy results of the 2 in-built kernels, the RBF kernel outperforms the Sigmoid kernel both before and after cross-validation. The RBF kernel transforms the data to a dimension with a greater linear separation than the Sigmoid function. This allowed our model to produce a better hyperplane in both cases, resulting in a better fit for the training data, showing better performance.

However, there is large overfitting with the RBF kernel shown by the training accuracy being much higher than the testing accuracy.

For our custom kernel, we have implemented a precomputed kernel called the Gram matrix (Pedregosa et al., 2011). With this custom kernel our model achieves an accuracy of 82.4% without cross-validation on the testing data, and 100% accuracy on the training data. This indicates that using the Gram matrix results in significant overfitting. With regards to the hyper-parameters, we use cross-validation for the Gram matrix, however, there are no hyper-parameters to tune. After cross-validation, the custom kernel obtains 82.13% for testing accuracy and 83.86% training accuracy. Having found training and validation data suitable for our model, there is a significant decrease in overfitting, now with only slight overfitting that is negligible. The RBF kernel outperforms the other two kernels in terms of accuracy, but with strong overfitting; our custom kernel, after cross-validation, has the same testing and training accuracy as the Sigmoid kernel.

We implemented cross-validation through using a n-fold validation algorithm, splitting the training data n-folds for an improved evaluation of our model and selecting the best hyper-parameters for the 3 kernels. 10 is typically the number of folds that is used, however, due to time constraints, we chose 5. Based on the validation results from the split, we determine the optimal parameters for the kernels. These optimised hyper-parameters of the kernels are then used to get the best performance on the test data.

1.3 Analysis of mis-classified test examples

The Figures for the confusion matrices of the different kernels can be found in Appendix A.

The RBF Confusion Matrix is shown in Appendix A.1, where the False Positive $>$ False Negative, indicating that the classifier is weaker at classifying positive reviews. However, the disparity between False Positive and False Negative is so small that this is negligible.

The Sigmoid Confusion Matrix is shown in Appendix A.2, where False Positive $>$ False Negative. Again, this means that the classifier is weaker at classifying positive reviews, where the disparity between False Positive and False Negative is so small that this is negligible.

Our custom kernel's Confusion Matrix is shown in Appendix A.3. The False Positive $<$ False Negative, which signifies that the classifier is weaker at classifying negative reviews.

- Avg length of reviews: 120.30
- Avg length of misclassified reviews of RBF: 125.21
- Avg length of misclassified review of Sigmoid: 113.04
- Avg length of misclassified review of precomputed: 112.74

From the results above, for the RBF kernel, the length of misclassified reviews is longer than the average length of reviews. As a result, lengthier reviews are misclassified more frequently, implying that the classifier is weaker with respect to longer reviews.

For the Sigmoid kernel, it has a shorter average length of misclassified reviews than the average length of reviews. Shorter reviews get misclassified more often, suggesting that the classifier is weaker when it comes to shorter reviews.

Similar to the Sigmoid kernel, our custom kernel is weaker with respect to shorter reviews, as it misclassifies short reviews more frequently.

2 Boosting

2.1 Introduction

For the second task of this assignment, we are tasked with implementing an ensemble modelling technique called boosting, to again solve the binary classification task of sentiment analysis. This ensemble modelling technique creates a strong classifier by training and aggregating several weak learners and learning from their mistakes, to produce a more accurate result.

We chose to use random forests to implement the boosting algorithm, which is a combination of bagging and decision trees (Pedregosa et al., 2011). Our choice of boosting algorithm is Ada-boost (Freund, 2001), also known as Adaptive Boosting, which is suitable for the use of random forests.

2.2 Evaluation and analysis of various parameters of the boosting algorithm

For the Adaboost algorithm, we chose to optimise the hyper-parameters "Criterion", "Splitter", number of classifiers and the depth of the classifiers.

"Criterion" can either choose between Gini impurity or entropy to assess the quality of a split. "Splitter" refers to the strategy for selecting a split at each node, either by best or by random.

The depth of each weak classifier has a direct impact on our model's accuracy; typically a depth of 1 is employed. However, we conducted several tests in which a depth of 7 resulted in an over complex model that caused overfitting and a depth of 5 showed improved accuracy. As a result, we picked this value range for cross-validation ([1, 2, 3, 4, 5, 6]).

Finally, the hyper-parameter of the number of classifiers to train. We performed a number of experiments, where 8 classifiers showed poor performance. Therefore, we chose this set of parameters for cross-validation ([8, 9, 10, 11, 12, 13, 14, 15]). We discover that as we increase the number of classifiers, the performance of the model increases, but with diminishing returns. In addition, overfitting does not change significantly as the number of classifiers increase.

A series of pre-processing steps are first performed on the IMDB dataset before being used as training data-points for our SVM and boosting models. This pre-processing stage is used to reduce the number of Bag-of-words features extracted, so that it contains only useful information from the reviews, resulting in improved learning and training speed. If this reduction is not performed, the Bag-of-words will include a huge number of features, significantly increasing the time of training the models. This Bag-of-words features then serve as the training data-points for training our models. The pre-processing steps are as follows:

- Tokenisation - The review sentences are first chopped into individual words and punctuation.
- Lowercase - Making all letters lowercase, such that there won't be recurrences of the same word with just different letter cases.
- Punctuation - Removal of punctuation, removal of useless information.
- Spelling - Correcting the spelling of words, so that different spellings of words are not repeatedly added to the Bag-of-words.
- Stop words - The removal of common words that don't provide much information e.g. 'a', 'an', 'the'.
- Stemming - Removes the last few characters of the word, which don't provide much information.

- Lemminisation - Mapping words to its base form with the same meaning. Therefore, different words with the same meaning can be referenced as a single word.

For the words ending with ”e”, we use Lemminisation, otherwise we use stemming in accordance with the English language.

Before cross-validation, our boosting algorithm shows slight overfitting, achieving a testing accuracy of 73.5% and 79.7% for the training accuracy, showing slight overfitting.

We simplified the challenge for cross-validation on our boosting algorithm, choosing to optimise the hyper-parameters for a single classifier first. Otherwise considering all the hyper-parameters at once would take too long. We optimise the hyper-parameters, ”max-depth”, ”Criterion” and ”Splitter” type first.

After performing cross-validation to get the optimal hyper-parameters for one classifier, we use those optimised hyper-parameters to determine the optimal number of classifiers to use. The optimised hyper-parameters we found to be ”max_depth” = 5, ”criterion” = gini, ”split_type” = best, ”number of classifiers” = 15. With these values, the model achieves a testing accuracy of 77.93% and a training accuracy of 85.52%. This implies that our boosting model has slight overfitting and also has a lower accuracy than all three of the other kernels from the SVM model.

2.3 Analysis of mis-classified test examples

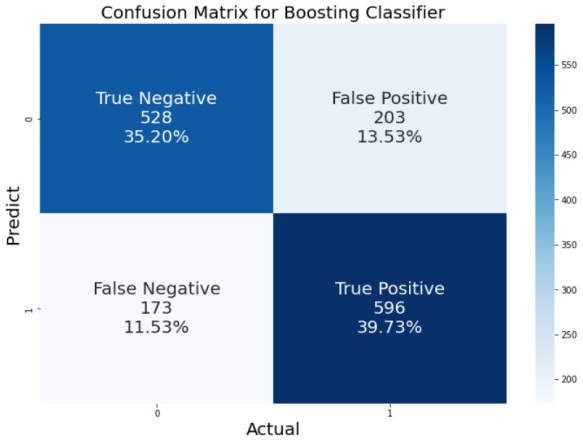

Figure 1: Confusion Matrix for Boosting Classifier

In Figure 1 above, the Confusion Matrix for the Adaboost classifier is displayed, where False Positives $>$ False Negatives. This indicates that the classifier is weaker at classifying positive reviews. Additionally, this classifier shows the biggest disparity between False Positives and False Negatives.

- Avg length of reviews: 120.30
- Avg length of misclassified reviews: 121.65

From the results above, longer IMDB reviews are very slightly getting incorrectly labelled more frequently. However, this is negligible.

3 Bibliography

References

Freund, Y. (2001), 'An adaptive version of the boost by majority algorithm', *Machine learning* **43**(3), 293–318.

Needham, C. (1990), 'Ratings, reviews, and where to watch the best movies amp; tv shows'. **URL:** *https://www.imdb.com/*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

4 Appendix

Appendix A Confusion Matrices

A.1 RBF Confusion Matrix

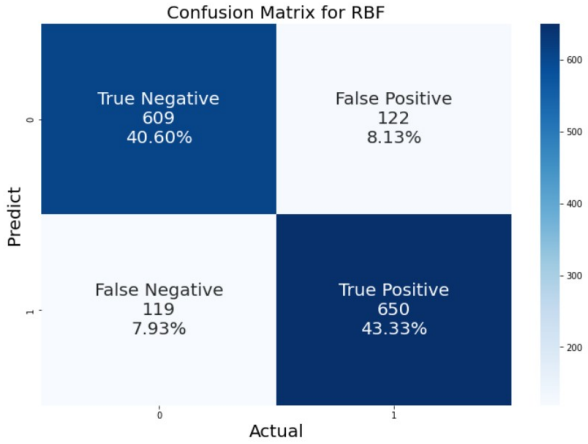

Figure 2: The RBF Confusion Matrix

A.2 Sigmoid Confusion Matrix

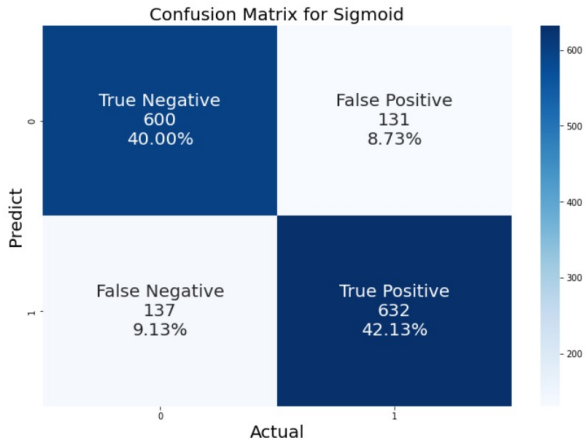

Figure 3: The Sigmoid Confusion Matrix

A.3 Custom kernel Confusion Matrix

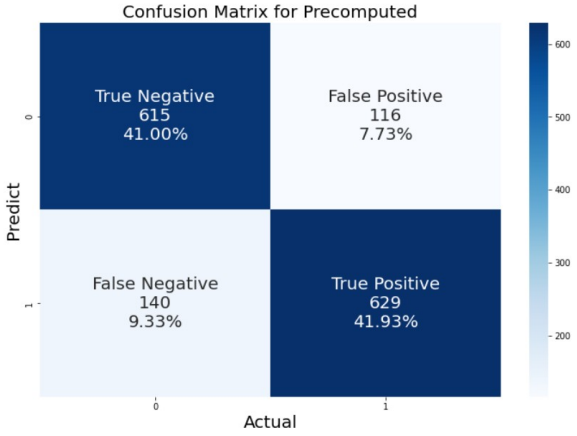

Figure 4: The Custom kernel Confusion Matrix