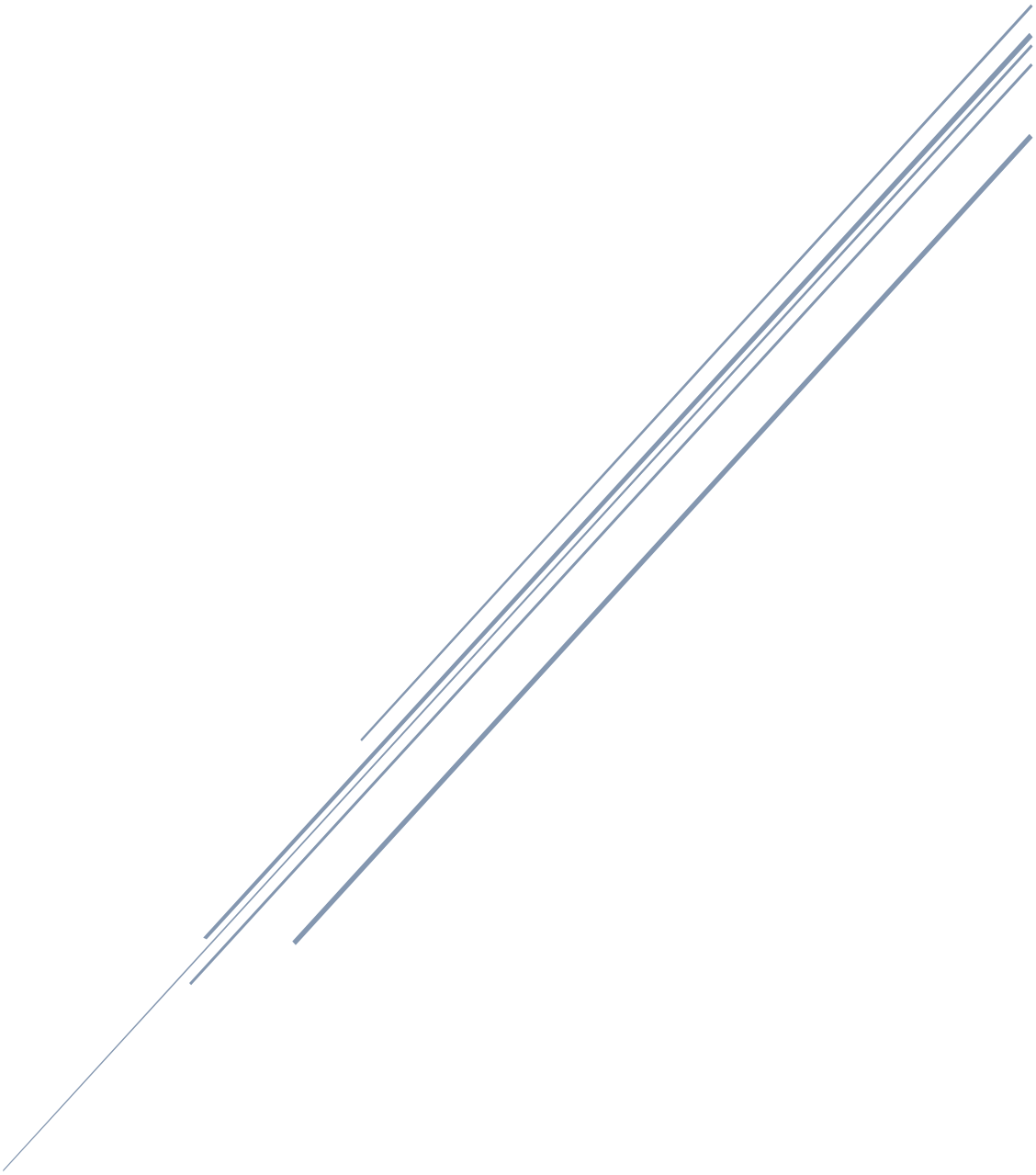
COURSEWORK - 2: SENTIMENT ANALYSIS



University of Bath
Machine Learning 2

Contents

Task 1: SVM Classification task analysis

For the SVM classification, we chose a linear kernel as it can solve many practical problems and is less computationally intensive than other kernels like RBF. The idea of linear SVM is straightforward, as the algorithm creates a line or a maximised hyperplane which separates the data into classes.

The Radial Basis Function Kernel is popular because of its similarity to the K-Nearest Neighbour Algorithm. It has the advantages of K-NN, but overcomes the space complexity problem, as RBF Kernel Support Vector Machines just need to store the support vectors during training and not the entire dataset.

For the custom kernel, the choice of function was motivated by running through various trial and error tests to obtain the best accuracy on the data.

Previously, the pre-processing was done by the clean string function, where the punctuations and stop words are filtered out and lemmatization was applied. Words less than 3 characters were also removed from the reviews and our built-in bag of word extraction function was used, which shuffles certain sections of the reviews as data augmentation to create more reviews. However, the accuracy result did not improve and it was later taken out of the algorithm. The training and the testing sentiment data are both converted from positive and negative to +1 and -1.

A range between 83 to 88% was obtained using the method mentioned above and it was later found that our built-in pre-process function was not needed due to TF-IDF Vectorizer having an inbuilt pre-processor for text. The default processing selects tokens of 2 or more alphanumeric characters and removes all punctuation and HTML tags. We used an n-gram range of (1,2), which has been commonly shown to improve accuracy. Sublinear term frequency scaling was also used in place of standard term frequency, as we saw an accuracy boost. This replaces the term frequency with 1 + log(term frequency).

The accuracy of the dataset using linear SVM also rose to 90% by applying both vectorizer and grid search, which was later implemented.

All of the SVM methods use a range of C values from 0.1 to 4 and a grid search was applied on these parameters, where:

1. Linear SVM - Both hinge and square of the hinge are tested for linear SVM loss. For penalty, both l1 and l2 regularization are used
2. RBF SVM - Gamma was set to both auto and scaled, and shrinking was set with both True and False
3. Custom SVM - The equation of the custom kernel follows:

$$\log(ax \cdot x^T) + b$$

Best SVM hyperparameters

For the linear kernel, the optimal hyperparameters were found to be C = 1.4, with an L2RB penalty and a hinge loss. With the RBF kernel, C = 1.83, gamma = ‘scale’ and the use of shrinking gave the highest accuracy. In our custom kernel, C = 4, a = 2.2 and b = 5 were found to be optimal.

Linear kernel results

The LinearSVC model was able to achieve an accuracy of 90% (0.89866). The confusion matrix below shows that the vast majority of predictions were correct. There were 660 true negatives and 71 false positives, representing a 9.7% error. The error was slightly higher for false negatives, namely 10.5%. Considering the simplicity of the function, the fact that the linear kernel gave such a high accuracy suggests that there was a linear decision boundary in the data. This is supported by research [1], where it is recommended to use linear kernels for text classification. As the text has a lot of features, complicated higher dimensional kernels don't give significant performance gains. The linear kernel was exceptionally faster to run than RBF and the custom kernel. There were also fewer parameters to optimise (no gamma), hence a linear kernel is the most suitable function for sentiment analysis.

RBF kernel results

The RBF kernel reached 89% accuracy (0.88733). This is slightly worse than the linear kernel and took a substantial amount of time to run, as expected by the higher dimensionality. The confusion matrix reveals a 9.3% false positive and a 13.1% false-negative rate.

Custom kernel results

Surprisingly, the custom kernel achieved the highest accuracy of over 90% (0.90066). There was a 9.6% false-positive and 10.3% false-negative rate, the lowest of all the kernels. Although this kernel was the most successful in terms of accuracy, its runtime was unfortunately very long. Hence, in terms of performance, the linear kernel is still the most suitable for sentiment analysis.

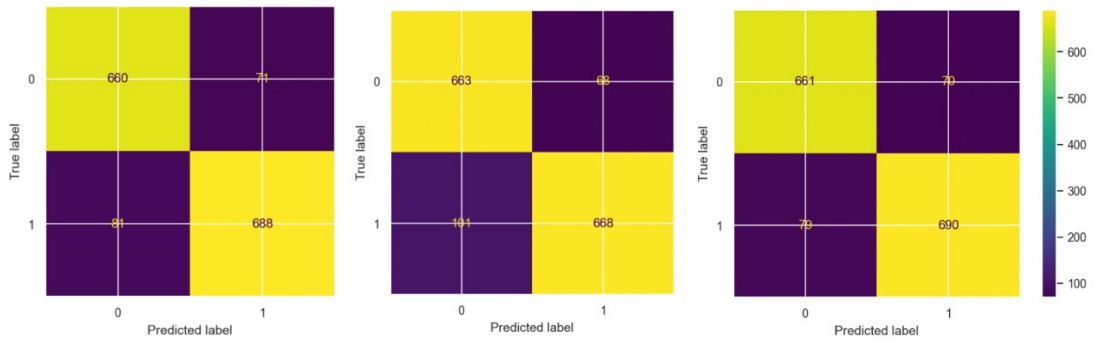


Figure 1:Linear SVM confusion matrix(left), RBF SVM confusion matrix(middle), Custom SVM confusion matrix(right)

Task 2: Boosting task analysis

Ensemble learning combines several base models to form one optimized model. Instead of relying on the accuracy of a single model, ensemble methods take multiple different weak learners and aggregate them into one strong predictor. They are often used to decrease the variance and bias and to give better generalized predictive models. The two groups of ensemble methods are:

- Sequential Learners – Different models are generated sequentially and the mistakes made by previous models are learned by successors, which exploits the dependency between models by assigning mislabelled examples higher weights (AdaBoost).
- Parallel Learners – Base models are generated in parallel, which exploits the independence between models by averaging out their mistakes (Random Forests).

An AdaBoost (Adaptive Boosting) [2] classifier begins by fitting the DecisionTreeClassifier on the original dataset, then fits additional copies of the classifier and dataset, modifying the weights of incorrectly classified instances such that subsequent classifiers focus on more difficult cases. This was chosen to reduce the penalty for our model in the case of incorrectly classified labels. Adjustable parameters in our custom class are num_estimators and max_depth, which both control the number of estimators generated for AdaBoost and the maximum depth of the DecisionTreeClassifier.

Boosting results

With the parameter setting at 1200 estimators and a maximum depth of 4, accuracy of around 84.6% was achieved on the boosting classifier. On a relatively fast computer, this took over more than 1 hour to train (dependent on the computing power available) which is an order of magnitude longer than the SVM training. The confusion matrix below shows a 16% false-positive and a 15% false-negative rate. The results in the booster classifier show that the SVM methods are far more effective than Boosting algorithms for sentiment analysis. Not only did SVM achieve higher accuracy, but it also performed better with a lower runtime. This was surprising because research from [3] suggested that random forests (similar to boosting algorithms) were the best for classification tasks and were expected to run faster. SVM is however more suited to binary classification problems and is better at handling sparse data, hence SVM's victory over Boosting may be limited to the niche scenario of sentiment analysis.

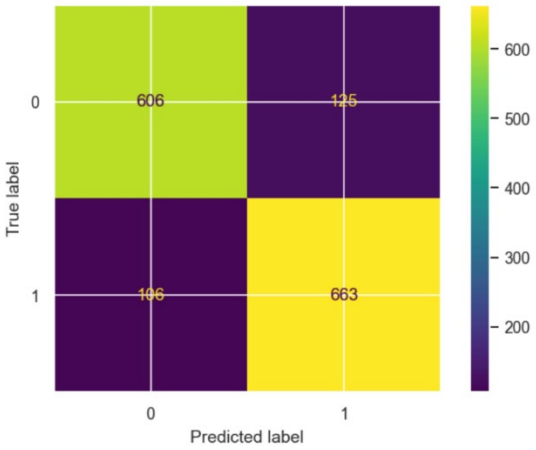


Figure 2: Boosting confusion matrix (1200 estimators, maximum depth of 4)

With the grid search implemented in the classifier, a range of large estimators and maximum depth values can be tested. While using more estimators and depth values can marginally increase the accuracy of the classifier, the code would take too long to run. Therefore, to decrease the runtime while maintaining an acceptable accuracy, smaller estimates are used. With the number of estimators limited to 100 and a maximum depth of 4, the accuracy of the classifier is still maintained at just over 81% to 82%, with a significantly reduced runtime of 5 minutes. It has a false positive rate of 18% and a false-negative rate of 18%. Unless using a greater number of estimators and a greater maximum depth, this accuracy is still far less accurate in comparison to the SVM classifier in task 1.

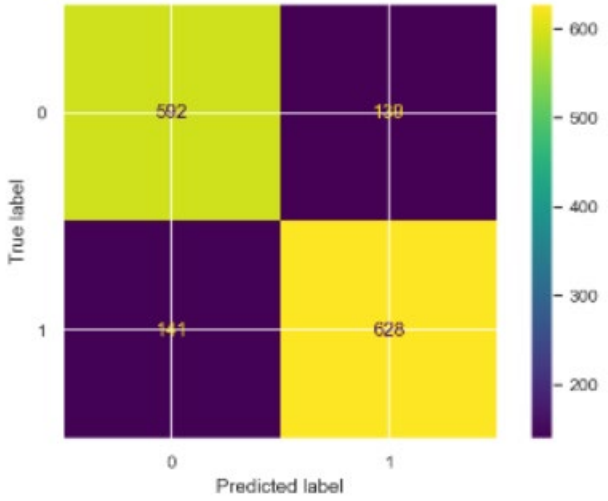


Figure 3: Boosting confusion matrix (100 estimators, maximum depth of 4)

For the final parameters, it is believed that setting the estimators to be around 1000 and a reduced value of maximum depth of 1 is an appropriate choice for higher accuracy and a decent amount of runtime.

References

[1] Kowalczyk A. Support vector machines succinctly. Syncfusion Inc 2017.

[2] Freund Y, Schapire RE. Experiments with a new boosting algorithm. icml, vol. 96, Citeseer; 1996, p. 148–56.

[3] Fernández-Delgado M, Cernadas E, Barro S, Amorim D. Do we need hundreds of classifiers to solve real world classification problems? The Journal of Machine Learning Research 2014;15:3133–81.