Student Number:198688 Kaggle Team Name:Slade

1. APPROACH

My solution to the competition will use a multi-layered perceptron to perform the classifications on test data.

The additional incomplete data is completed using mean imputation. Confidence ratings then replace the outputs for the data that has them, This works well for the MLP regressor approach I am using and adds a type of weighting to part of the training data. The data is then scaled to a maximum of 1 using a sci-kit learn module made to handle sparse data [1] the test data is transformed with the same scale as the training data. Domain adaption is then performed on the data. This is my own development based upon previous work in domain adaption [2]. Once the data has been standardised and domain adapted, it is used to train an MLP regressor. The MLP regressor then classifies the test data before the proportions are equalised to match the testing set proportions before the regressor classifications are converted to binary 1/0 outputs to be tested on the Kaggle site.

The reason I intend to use a multi-layered perceptron is because my feature selection will be used to reduce dissimilarity between the training domain and test domain [2], so I still intend to have high dimensionality to my data. MLP handle high dimensional data well [3]. The reason I am using a regressor is because it allows me to utilise the annotator confidence and test set proportions easily. MLP regressors performed best in preliminary tests of classifiers.

The main assumption of this approach is reducing features for the sake of domain adaption as a focus rather than the focus being on accuracy of cross validation. This should in theory reduce the variance of my approach. Because of this being the focus of feature selection, it results in using very high dimensional feature sets

2. METHODOLOGY

My classifier was trained on the complete and incomplete training data. To handle and refine the missing data in the additional data, I will use the technique of Imputation, where I replace the missing value with the mean of that value for that class, this method is shown to improve results classifier [4].

I performed scaling on the data where all data is converted with a maximum scale of 1. This approach is recommended for sparse datasets [1]. Analysis of the given training set shows a high amount of 0s adding to the sparsity. Scaling high dimensional data is shown to improve the performance of an MLP regressor [5].

Part of the data had its outputs replaced with its ‘confidence rating’ negative rating for negative classifications and positive ratings for positive classifications. This use of the annotator confidence rating allows me to further a part of the training data. This method works well with a regressor and increased the performance of the classifier.

The use of the test proportions helped increase the accuracy of the classifier. Because of the continuous nature of the regressor outputs, I was able to subtract or add 0.001 to each classification iteratively until the proportions of the test data predictions matched the proportions of the test data. This helped increase accuracy and justify use of a system that can produce continuous outputs as opposed to binary classifications. The data is then converted to binary classifications once this process has been performed. This methodology has been shown to produce successful results in previous Kaggle competitions [6].

Feature selection was performed for domain adaption. The method of domain adaption is a simplified approach inspired by previous work on the topic [2]. I was able to simply the approach with the use of the provided test class proportions. The training data is split by class with the mean value for each feature calculated. The expected mean for the test set is calculated by multiplying the training class means by the test proportions. Any feature in the testing set outside of 1 standard deviation * x is then removed from both the training set and testing set. With this approach x is a free variable that is tuned by hand. This ensures the classifier is not trained on features that deviate to greatly from the norm of between data sets.

The hyper parameters are tuned using cross validation. I use k-fold cross validation with k of 5. K of 5 is possible due to the use of the additional training data due to the increased size of the training set.

3. RESULTS AND DISCUSSION

3.1 Implementation Features

Feature added:	Accuracy:
V1	0.62926
V2 Annotator Confidence	0.63494
V3 test proportions	0.67471
V4 additional data	0.71875
V5 scaling	0.69744
V6 domain adaption	0.70738
V7 hyper params tuned	0.7159

Table 1: Accuracy results per feature added.

Testing of features was done using the accuracy score generated from the Kaggle competition. Table 1 shows the total accuracy after each part was added.

Each version generally increased in accuracy, however scaling showed a decrease, this leads me to believe that more work is needed in determing effective stadnardisation and normalisation of the data.

3.2 Hyper Parameter Tuning

Solver	Accuracy
lbfg	0.6637
sgd	0.5467
adam	0.6714

Table 2: Hyper parameter tests for 'solver'

Hyper parameters where tuned using the same standard seed for each classifier to ensure results where comparable. K-fold 5 was used for the cross validation, this ensures the is a large average for comparison, while keeping the data size comparable to the amount of data needed to ensure high accuracy established in the variance/bias testing.In total 8 hyper parameters where tuned.

Through the tuning of hyper parameters, I was able to increase the overall accuracy in cross validation tests from 0.6714 to 0.6783. While this increase is not large, this is partly due to the default hyper parameters being well optimised. This is shown through an example of the hyper parameter test on the solver shown in table 2, where the default solver is adam.

Table 2 shows how the default solver had the best performance.

This trend showed through most of the hyper parameters with the biggest increase coming from changing the default to not shuffling the training data. Shown in table 3

Shuffle	Accuracy
TRUE	0.6733
FALSE	0.6783

Table 3: Hyper parameter tests for 'shuffle'

3.3 Bias Variance Analysis

Figure 1 shows the analysis of my solutions bias/variance when starting with very small training sets.

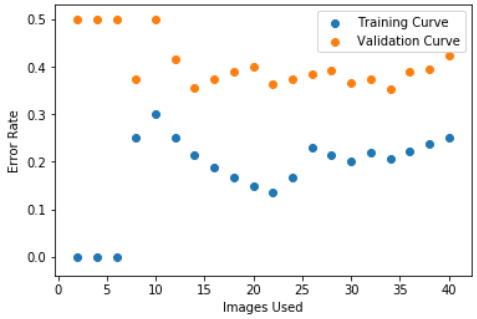

Figure 1: Training/Validation curve when trained on small sets of data

However when larger amounts are used, the difference between training and validation quickly becomes less noticeable. Figure 2 shows the same tests when larger sets are used to train the classifiers.

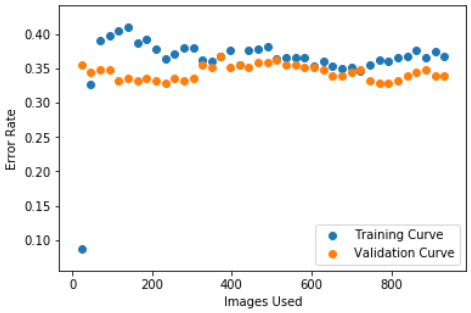

Figure 2: Training/validation curve when trained on larger sets of data

While there is not a high variance, it is noticeable with room for improvement. The reason for an average variance rate is likely due to a reduction in the model complexity through removing the domain dependant

features and through the mean imputation used on the missing data.

The classifier I used for my final implementation comes with a hyper parameter for tuning the regularisation values, due to the close nature of the training and validation curve, I am not optimising this parameter to control variance/bias to decrease overfitting.

3.4 Discussion

My overall goal for this project was to build a classifier and find ways to apply my knowledge in order to improve on its effectiveness. I feel that while my overall goal was achieved there are parts that could have been done differently.

Looking through the accuracy gained from the adaptions I implemented. I believe that I would have achieved better results if I had used a library to perform feature weighting for the domain adaption, spent more time on accuracy-based feature selection and tested the effect of more standardisation and normalisation techniques.

One drawback to the investigation is that my tests on the different version of the implementation where not done on a standardised seed for the network, unlike with parameter tests. This has made version comparison harder to definitively say and is something I would do differently in further work.

Student Number:198688 Kaggle Team Name:Slade

Bibliography

[1] "sklearn.preprocessing.MaxAbsScaler," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MaxAbsScaler.html. [Accessed 15 5 2018].

[2] Y. Mansour, M. Mohri and A. Rostamizadeh, "Domain Adaptation: Learning Bounds and Algorithms," in *COLT 2009 : The 22nd Annual Conference on Learning Theory*, Montreal, Canada, 2009.

[3] S. B. Kotsiantis, Emerging Artificial Intelligence Applications In Computer Engineering, 1 ed., Amsterdam;Washington: IOS Press, 2007.

[4] T. G. Stewart, D. Zeng and M. C. Wu, "Constructing support vector machines with missing data," *WIREs Computational Statistics*, vol. 10, no. 4, 2018.

[5] C. C. Aggarwal, A. Hinnerburg and D. A. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Spaces," in *8th International Conference on Database Theory*, Berlin, 2001.