Classification of Multiclass Overlapping Data

Renato J. Rodriguez Nunez
Department of Mechanical Engineering, Temple University
Renato.Rodriguez@temple.edu

Introduction: This research involves the classification of a multiclass (3-class) dataset with significant class overlap. Class overlap is considered one of the most difficult problems in classification [4]. The problem arises from different classes sharing similar characteristics which results in class boundaries that are not clearly defined, and thus not linearly separable [2]. Current approaches from the literature use modified version of popular classifiers or implement a feature engineering approach to minimize the negative effects of the class overlap. In this study a multilayer perceptron (MLP) developed in the TensorFlow learning platform, and a random forest classifier (RFC) developed in the scikit-learn learning platform were used to tackle the classification problem of overlapping data. These classifiers were selected based on intuition gained from visualizing the data, refer to Figure-1, test classification scores, refer to Table-1, and information from the literature [1],[2],[3],[4]. Note that the test error rates were achieved via implementation of the respective classifiers with default hyperparameter settings. Also, note that the feature engineering approach was also implemented, with the goal of reducing the negative impact of overlapping via the implementation of different linear and nonlinear transformations, as well as the introduction of new features. These results however are not presented in this report as they did not yield performance improvements (0.5% to 1% decrease in performance). The rest of the paper is organized as follows. Section I presents a detailed summary of the multilayer perceptron approach and the process used to fine tune the model. Section II presents a summary of the random forest classifier and how it was tuned. Section III shows the classification results achieved by the selected classification methods, MLP and RFC. Lastly, Section IV presents the summary and conclusion.

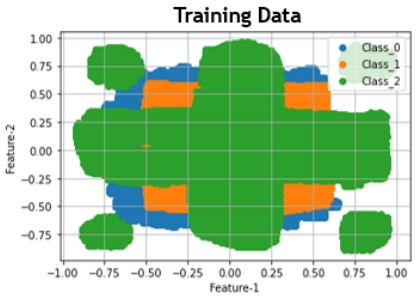

Figure-2: Visualization of Training Data

	Data Set	
Algorithm	**Train**	**Dev**
AdaBoost	36.92%	37.08%
KNN	N/A	30.75
MLP	28.15%	31.97%
RFC	32.66%	32.81%
SVM-RBF	30.86%	30.99%

Table-1: Comparison of Test Error Rates

Multilayer Perceptron Approach: The MLP classifier is a fully connected feedforward artificial neural network. The MLP used in this study was developed in the TensorFlow learning Platform. This classification method has many hyperparameters (hp) that define its structure or topology, as well as its predictive performance. These hyperparameters include the number of hidden layers, the number of neurons per layer, the activation functions, the number of epochs, and in some cases can also include the loss function and optimizer. It is easy to see that properly tunning a system with this many variables is a complex problem, and that manually tunning via a trail and error approach can be extremely time consuming and often unfruitful. This complex problem was approached via an automated hyperparameter tunning process. This process works by implementing variations of hyperparameter sets to the MLP model and looks for the set within defined hyperparameter range that maximizes validation-accuracy. It is important to note that the hyperparameters are tuned based on validation results, this is distinctly different than the weights which are tuned during the training process. Tunning the hyperparameters on the validation-accuracy helps prevent over-fitting during the training process, meaning that the learned system is more generalizable and will perform well on unseen data, such as the development and evaluation data sets.

To implement this automated tunning approached the development of a dynamic MLP model along with a hyperparameter testing range are needed. The dynamic MLP model replaces static (constant) hyperparameter inputs with variables. These variable inputs are used by the tunning process to implement and test different variations of the hyperparameter sets. These sets can include numerical values for parameters such as the number of hidden layers, string values for parameters such as the activation functions, and Booleans for activating and deactivating processes such as pooling and dropout. The hyperparameter sets are bounded by the range specified for each of the parameters being turned. For numerical values, this range is defined by a minimum, a maximum and a step size. For string and Boolean values, this range is defined by a list of options (strings/Booleans) that the process can iterate through. This automated tunning process was used to develop the MLP classifier used in this research, refer to Table-2. The classification results obtained via this method are discussed in Section III Results.

Hyperparameters	Value
Number of HL	**9** (+2 in/out)
Number of Neurons	**5:90** (per HL)
Activation Function	**ReLU**, **Tanh**, etc. (9 diff)
Optimizer	**Adam**
Loss	Sparse-Categorical Crossentropy
Number of Epochs	50

Table-2: MLP Hyperparameters

Random Forrest Classifier Approach: The RFC is an ensemble-based meta-estimator that uses decision trees to fit various subsets of training data. Like other ensemble methods, it uses averaging to improve the predictive accuracy of the learned model and to prevent over-fitting. For tunning of the RFC's hyper parameters, such as number of estimators, max depth, and max samples, a tunning process akin to the method used for tunning the MLP was used. Since the MLP approach was developed for the TensorFlow platform only, a similar but simplified approach was implemented for the scikit-learn RFC classification method. The main distinction here is that while the previous approach implemented and tested a set of hyperparameters at a time, this simplified approach iterates over individual hyperparameters. Another key distinction is that this tunning process seeks to obtain the hyperparameters that optimize training-accuracy, while the previous method optimized validation-accuracy. This is an important distinction because optimizing the training-accuracy can lead to overfitting, as is discussed in Section III Results.

	Data Set		
Algorithm	**Train**	**Dev Test**	**Eval**
RFC	00.18%	29.03%	63.36%
MLP	27.44%	27.40%	61.59%

Table-3. Comparison of Final Error Rates.

Results: The RFC and MLP classification methods used in this study were tunning via automated hyperparameter tunning processes as describe in Section I and Section II. The developed classifiers were used to generated (predict) class assignments for the training, development, and evaluation data sets. These results are summarized in Table-3. From Table-3 we can conclude that both methods performed well on all data sets, but the RFC had the best performance on the training data set, while the MLP method had the best performance on the development and evaluation data sets. This is an important observation because it means that the RFC severely overtrained on the training data set reaching a marginal error rate of 0.18%, which lead to poor generalization that corresponds to the significantly higher error rates achieved on the unseen data sets. Also, from Table-3 we can determine that the MLP approach did not suffer as much from

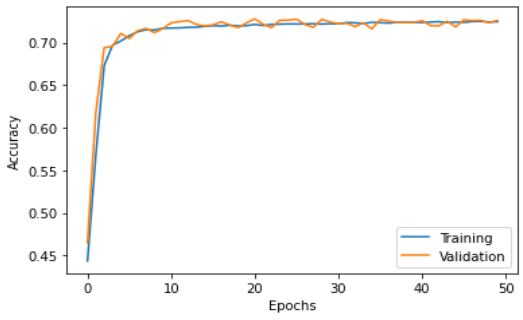

Figure-2: MLP Accuracy vs Epochs

this issue as it achieved essentially the same score on both the training and development data sets. Further evidence of the generalizability of the MLP method is presented in Figure-2, where both the training and validation accuracy ($1 - error\ rate$) converge to approximately the same value.

Conclusions: In this study two classifications methods were used to tackle the complex multiclass classification problem of a dataset with significant class overlap. These methods include the random forest classifier (RFC), and the Multilayer Perceptron (MLP). Both methods were fine-tuned via automated hyperparameter tunning processes. The MLP tunning process works by implementing variations of hyperparameter sets to the MLP model and looks for the set within defined hyperparameter range that maximizes validation-accuracy. As detailed in Section III Results, the MLP method tuned via this automated approach yielded good predictive performance and generalizability. But it did not reach the expected classification performance, consisting of matching or improving upon the provided baseline results. Considering this issue, further work is needed to refine this approach and ensure that the optimal hyperparameter set is found.

Citations:

[1] Lecture Slides ECE 8527 Introduction to Machine Learning and Pattern Recognition
[2] Sáez, J. A., Galar, M., & Krawczyk, B. (2019). Addressing the overlapping data problem in classification using the one-vs-one decomposition strategy. IEEE Access, 7, 83396-83411.
[3] Tang, W., Mao, K. Z., Mak, L. O., & Ng, G. W. (2010, July). Classification for overlapping classes using optimized overlapping region detection and soft decision. In 2010 13th International Conference on Information Fusion (pp. 1-8). IEEE.
[4] Xiong, H., Wu, J., & Liu, L. (2010, December). Classification with classoverlapping: A systematic study. In Proceedings of the 1st International Conference on E-Business Intelligence (ICEBI2010),. Atlantis Press.
[5] https://scikit-learn.org/
[6] https://keras.io/