**LEBANESE AMERICAN UNIVESRITY**

**COMPUTER SCIENCE AND MATH DEPARTMENT, BYBLOS**

**CSC615 MACHINE LEARNING**

**ASSIGNMENT #2  
  
  
Ghadi El Hasbani, LAU I.D.: 201903122**

**Danielle Antoun Azar, PhD**

1. (60 points) Consider the problem of representing the Boolean function that is the disjunction of 5 boolean variables (x1vx2x3vx4vx5).
   1. Show all instances of the problem
   2. Implement in Java or Python the backpropagation algorithm for training a neural network with a single hidden layer. The code should allow for different numbers of neurons in the hidden layer. Have it prompt the user for the number of neurons. **Train your network on the data set you formed in Part a of this question.**

Note: You have to write the code from scratch. I mind a lot that you re-use code in this question.

* 1. In the previous assignment, you chose and used a data set to construct decision trees using C5. Train the neural network and test it on the same data you used in that assignment. Use the same data set up i.e. use the same folds you created in that assignment. If you did not save them, perform 10-fold cross-validation again. Plot the training and the validation error and how it changes with the number of iterations. Train and validate the network with several parameters (you find out what these are) before you choose the best ones and move on to the next part of the question.
  2. Choose the ANN parameters that gave the best results on the validation folds (best average). Retrain the model on the entire Set A and use it to classify the data in set C. This is your testing error.
  3. Generate the testing results for all metrics (accuracy, F1-measure, precision, recall, AUC – all those you generated in Assignment 1. Compare them to those obtained with C5.

**Note:** You can easily find code which implements neural networks online. However, unless you implement them yourselves you will not really understand how they work and then, there is only one person to blame when comes the exam. Guess who that would be…

1. (10 points) In this question, you should use the SVM library implemented in Python. Use the same data set you used in Question 1 and the previous assignment to train and test SVM. Tabulate and compare the testing results to those obtained with ANN and C5. Comment on the findings.
2. (5 points) In this question, you should use the ANN library implemented in Python. Repeat Q 1 with the library. Compare the results to those you obtained with your implementation. Add these results to the table you formed in the previous questions.

With relu activation, 1 hidden unit, a learning rate of 0.25, and a threshold of 0.5, my ANN achieves an accuracy of 100% at the 3rd epoch and converges at the 72nd epoch for the problem of representing the function that is the disjunction of 5 booleans. Nevertheless, probably because it does not contain bias units, the predicted probability for the negative instance (0,0,0,0,0) is 0.5 whereas all other predicted probabilities for positive instances are close to 1. Note that when sigmoid activation is used, my ANN trains on 0,1 labels, not 0.1,0.9 which could lead to an overflow in the exponential function. Moreover, MLPC minimizes cross entropy and uses a low regularization parameter in contrast to my ANN minimizing the squared difference using no regularization parameter.

For SVM, the model was tuned with a mistake in the scoring function whereby binary labels and not probabilities were used. This step could nevertheless not be repeated due to time-constraints.

Both my ANN and MLPC were tuned to have 2 hidden layers with differing numbers of units. Actually, MLPC selected double the number of units in the layers as compared to my ANN. The learning rate selected was comparable, but MLPC momentum was slightly higher. Batch size for MLPC was also drastically lower than my ANN, and activation was selected to be sigmoid as compared to relu for my ANN. For SVM, the regularization parameter C was set to around 27 . The chosen kernel was a radial basis function with a gamma of 0.68. This indicates the model selected is slightly high in complexity.

It seems that the testing metrics for my ANN indicate that it has comparable accuracy (0.59 ±0.09) and AUC (0.6 ±0.05) to SVM (Accuracy and AUC: 0.58 ±0.00) but lower than MLPC (Accuracy: 0.64 ±0.01, AUC: 0.63 ±0.01) and C5 (Accuracy: 0.63 ±0.01, AUC: 0.66 ±0.01) on the Steinmetz dataset. On the other hand, precision was comparable for my ANN (0.68 ±0.01), MLPC (0.68 ±0.01), and C5 (0.68 ±0.06) with a slightly lower precision for SVM (0.64 ±0.00). Nevertheless, my ANN had the lowest recall (0.52 ±0.28) and the best specificity (0.67 ±0.18). The f1-score for my ANN (0.67 ±0.01) was also close behind C5 which has the second-best f1-score (0.68 ±0.03) after MLPC (0.69 ±0.02). These metrics indicate that the main issue with my ANN is false negatives. Perhaps a different threshold than .5 is needed to tune for these metrics and lower the false negative rate.

The sklearn MLPC performs comparably to C5 on all metrics, indicating that a simpler, more interpretable model such as C5 might be more suitable in this case. This is especially given the performance of other models. My ANN performs comparably to or worse than MLPC on all metrics except specificity. This indicates that my ANN has a low false positive rate compared to MLPC. SVM performs worse than all other models on all metrics, but has almost comparable precision and recall to C5 and MLPC and higher recall than my ANN. Nevertheless, standard deviations indicate that the SVM model has much less variability compared to the other two models which is surprising given the complexity of the model. The training metrics could hint to us if SVM is overfitting. Nevertheless, variability of MLPC and C5 remain low whereas some metrics for my ANN have high variability indicating possible overfitting for this model. The results of these testing metrics mostly indicate that if the priority is to avoid falsely detecting NoGo trials, my ANN is more suitable. In contrast, if the priority is to avoid falsely detecting Go trials, MLPC or a, less complex, C5 model are more suitable.

The mean error on both training and validation sets, however, is much lower for my ANN (train: 668.77 ±67.24, val: 110.83 ±34.63) as compared to either MLPC (train: 2033.08 ±64.55, val: 225.82 ±7.33) or SVM (train: 1911.17 ±14.38, val: 206.57 ±1.87) which have comparable errors between both training and validation sets. Nevertheless, the error for SVM seems to have less variability than MLPC and my ANN which is consistent with the testing metrics. This also indicates my ANN is possibly overfitting, especially since the variability of the validation error is higher than that of MLPC and SVM. SVM seems to be the robust choice in this case.

The mean normalized training error for my ANN decreases drastically in the first few epochs then exhibits a plateau, and the standard deviations behaves similarly whereby it increases then plateaus. In contrast, the mean normalized validation error and its standard deviation mostly increase in the first few iterations and then exhibit a plateau. This indicates that the model might be experiencing overfitting, and the stopping criterion should maybe be based on the validation error instead of the training error. Perhaps more sampling in RandomSearchCV would have also yielded a lower learning rate. Nevertheless, time-constraints prevented thorough random sampling of the search space. Finally, a regularization parameter could also prevent overfitting.

1. (25 points) Search the web for any article on SVM or ANN published in one of the conferences or journals you located in your previous assignment. The article should date back to 2018 or later. Summarize the article. Give your thoughts about ways to improve on what’s been done. Prepare a power point presentation of the work presented in it.

ANNs are powerful on their own, but a combination of them in an ensemble can improve predictive performance and reduce variability. Nevertheless, it is also useful for this ensemble to be dynamic, especially when the model response is as dynamic as time-series. In the problem of time-series forecasting using ensemble methods, research has mostly focused on model aggregation. Nevertheless, another important step in ensemble fitting is model pruning, which is a combinatorial search problem. Previous research has indicated that performance and diversity of base models are both important in the model pruning selection step. Nevertheless, with the dynamic nature of time-series, it could be necessary to perform model pruning many times to update selected models at different points. This aspect requires that model pruning approaches be time-efficient. Saadallah et al. (2022) propose a method called Online Ensemble Pruning using Regions of Competence or OEP-ROC which, using interpretable saliency maps in the form of RoCs indicating model specificity to a time interval, performs drift-informed model pruning updates in a way to promote performance and diversity of underlying base models.

Base models chosen are CNN-based with some having an LSTM layer. The different parameters of this base architecture is varied to create a diverse ensemble of models. The drift concepts used are concept and model performance drift. The former indicates when drift occurs in the time-series by monitoring its mean, whereas the latter indicates model performance drift by monitoring its RoC similarity to the time-series as compared to the best model in this regard. User-specified thresholds are used to decide when such drifts have occurred. At the first forecast or whenever drift is detected, the model pruning occurs. Primarily, RoCs are computed and used to enrich previous RoCs if this forecast is not the first. Then, models are clustered using Euclidean distance and only cluster representatives are kept. Representatives in this step are chosen based on performance indicated as the similarity of RoC to the current pattern. Next, the errors for the ensemble are calculated that take into account the weighted average, ambiguity (variance), RoC similarity to pattern, and RoC diversity (distance) compared to other RoCs. The top-M models are finally selected through ranking based on a combination of those measures.

The model outperforms most state-of-the-art methods in a pairwise comparison. Moreover, variants of OEP-ROC were built by omitting or exclusively using one of the original steps. These variants were not found to improve on the model’s performance indicating the importance of all these steps together. Moreover, the variant built without drift-aware pruning although randomly updates the pruning more than the drift-aware method does not improve performance. This indicates that drift-aware pruning promotes computational efficiency without decreasing performance. Finally, model aggregation methods were found to have improved performance when combined with OEP-ROC as compared to aggregation methods alone.

Although this method provides interpretable results, it is specific to ANNs due to its reliance on saliency maps. A suggestion to generalize the method to other base models could include using representational dissimilarity matrix of the base model forecast as its input representation and the true time-series input. In this way, this method could be adapted to base models that output time-series forecasts. Moreover, although neural networks can be easily adapted to multi-target problems, it could be useful for target forecasts to inform each other’s forecasts at following timesteps in the problem of mutli-target forecasting. A recurrent connection could be added to feed back the forecasts at t-1 as inputs to the network at forecast t. This method is similar to extended Stacked-Single Target and Regressor Chains described in Spyromitros-Xioufis et al. (2016). In the case of extended Stacked-Single Target, a first stage of models is built for each forecast and outputs of all other forecasts are added to the second stage model predicting a forecast A. In neural networks, we could add all forecast outputs at t-1 to forecast of all outputs at t since multiple outputs can be accommodated easily. Nevertheless this does not test the relationship between different forecasts at t. In the case of extended Regressor Chains, a chain of models is built by forecasting one target and using this forecast as input in the subsequent model forecasting a different target. Because of the ordered nature of this process, it is repeated with many permutations of targets so that many orderings are considered. This would mean that, combined with OEP-ROC, this would result in an ensemble of ensembles. For this reason, RoCs of subensembles can be averaged to give 1 RoC per subensemble.

**References**

Saadallah, A., Jakobs, M., & Morik, K. (2022). Explainable online ensemble of deep neural network pruning for time series forecasting. *Machine Learning*, *111*(9), 3459-3487.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., & Vlahavas, I. (2016). Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, *104*(1), 55-98.