Ordinal Classification Techniques Applied to Racehorse Ranking Prediction

Adam Gardiner-Hill MSci - Quantum Leap Solutions

May 11, 2022

Abstract

In this investigation we aim to build upon the work previously conducted in our studies on the application of machine learning algorithms to race horse ranking prediction. We introduce ordinal classification techniques to study whether they can improve overall network accuracy when compared to standard classification methodologies. The ordinal flat model achieved an exact prediction accuracy of 47.9% and an accuracy to within one rank of 95.4%, whilst the ordinal jumps model achieved an exact accuracy of 49.2% and an accuracy to within one rank of 95.8%. These results indicated that a slight improvement in network accuracy, especially to within one rank of error, was achieved with the introduction of ordinal techniques.

Investigation

Ordinal Classification Introduction

In our previous work, we outlined that 'rankings prediction' can be thought of as falling somewhere in between a classification and regression problem. The target variable exists in discrete groups, but unlike more common classification problems where each value is entirely independent of all others, the order of these groups relative to each other is important. This is because if the network predicts a rank 4 for a true rank 1, it is 'more wrong' than if it predicts a rank 2 or 3 for a true rank 1. Ordinal classification problems are so named because the 'order' of the target variable is also important.

It is common in classification networks to use loss functions that punish all incorrect assessments equally, such as the categorical cross-entropy function we used in our previous studies. Given by the equation,

$$Loss = -\sum_{i=1}^{n} y_i \cdot log(\hat{y}_i) \tag{1}$$

where n is the number of possible output classes, y_i is the target class probability, and $\hat{y}_i$ is corresponding output probability from the Softmax activation function. As can be inferred from equation 1, any incorrect classification group will have a term equal to zero in the loss function because of the y_i factor. This means that all incorrect classifications are punished equally by the loss function and relative order or 'closeness' is disregarded.

To remedy this issue, a dummy variable can be used to bin classification classes and make them correspond to a continuous output variable, which can be assigned as the target for a regression type network. For example, in our case of four output classes, we can normalise the values to between 0 and 1. Four bins are then created, with edges at 0.25, 0.5 and 0.75. Each rank 1-4, can then be assigned to a center of each bin, 1 to 0.125, 2 to 0.375, 3 to 0.625 and 4 to 0.875. This process is visualised in figure 1. This reassignment allows us to use a regression type network to try and predict a value between 0 and 1 based on the same input variables. Regression type networks commonly use loss functions such as the mean squared error to train models, gievn by the equation,

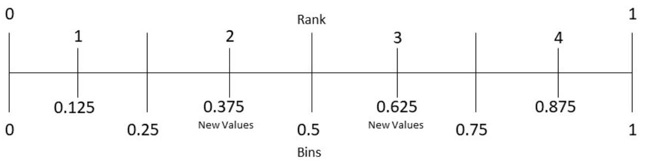

Figure 1: Illustration of the reassignment process to make classification data ordinal in nature. Each rank is reassigned to the center of a bin on a line normalised to between 0 and 1.

$$Loss = MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2}$$

where N is the number of data instances, y_i is the true value and $\hat{y}_i$ is the predicted value. The use of a loss function of this form will act to punish the network more severely for predictions that are further away from their true value than those that are closer, better accounting for the ordinal nature of the data. Any regression value predicted by the network for a runner instance can then be placed into its associated bin, and the corresponding, discrete 1-4 rank inferred.

Theoretically, this approach should offer improvements in network accuracy, especially to within one rank of error, due to the increased informational feedback that the network will receive from its loss function during training.

Data and Pre-Processing

The combined flat and jumps data sets used in our previous investigation were recalled. They were then reformatted as described to make the target 'rank' data ordinal in nature according to the process outlined the previous subsection. Illustrations of the data structure for the flat and jumps data can be seen in tables 1 and 2 respectively.

Reusing the exact data from our previous investigation allowed for easier direct comparison between ordinal and non-ordinal results, and also saved on computational workload as some results had already been obtained.

MoPR	Draw	Race P.	Stalls Pos.	Race Dis.	Strat.
...	...	...	...	...	...

Table 1: Table illustrating the variables used to train the network for all flat racing after the removal of the 'distance to first bend' variable

MoPR	Race Pace	Race Distance	Strategy
...	...	...	...

Table 2: Table illustrating the considered variables used to train the network for all jumps races.

Neural Networks

Two separate models were trained using the flat and jumps data from our previous investigation. Standard sequential neural networks from the Keras python package were used and consisted of one input layer, three dense layers and a regression type output layer. For the flat model, the network input layer had shape (6,), each of the subsequent dense layers had nodes 60, 42, 6, and the final regression layer had 1. For the jumps model, the input layer had shape (4,), the dense layers had nodes 40, 28, 4 and the regression layer had 1.

ReLu activation functions were used throughout the network architecture, and a mean squared error loss function was used to train the models.

Platform and Packages

The analysis was completed using open source Python packages and the Spyder IDE. These included NumPy, Pandas, Keras, Sci-Kit Learn, MatPlotLib and Seaborn. A TensorFlow virtual environment was created for this task, and all neural networks were trained on an NVidia GTX 1060 6GB GPU.

Results

Combined Flat Racing

The regression model trained on the combined flat racing data set achieved a mean squared error score on the test data of 0.0382, which resulted in a final classification accuracy of 47.9% and 95.4% to within one rank of error. The training and test accuracy of the model, as well as the confusion matrix for its predictions can be seen in figure 2

Jumps Racing

The network trained using jumps racing data achieved a mean squared error on the test data of 0.0373. This gave a final exact classification accuracy of 49.2% and an accuracy to within one rank of error of 95.8%. The training and test accuracy of the model, as well as the confusion matrix for its predictions can be seen in figure 3.

Discussion of Results

Comparison to Classification Approach

For combined flat races, the ordinal network achieved exact accuracy of 47.9%, and accuracy to within one of error of 95.4%. This is compared to the traditional classification approach that we took in our previous investigation, where the network achieved accuracies of 47.8% and 94.2%. The confusion matrices for the ordinal and non-ordinal approaches can be seen in figure 4. The improvement of 0.1% in exact accuracy is minimal and its difficult to know whether its due to stochastic processes associated with the use of machine learning algorithms. The improvement of 1.2% to within one rank of error is more substantial, and indicative that the introduction or ordinal techniques improved network performance in a meaningful way.

At a more granular level, the regression network exhibited a more aggressive tendency to predict rank 3s for true rank 2s and 4s. It identified 1% more leaders correctly than the classification network, but 3.2% and 7.2% rank 2s and 4s correctly, respectively.

For the jumps data, the regression network achieved an exact accuracy of 49.2% and an accuracy to within one rank of error of 95.8%. This is compared to the traditional classification network which achieved an exact accuracy of 49.4% and an accuracy to within one rank of error of 94.7%. A slight decrease of 0.2% in exact accuracy is compensated for by a 1.1% improvement in accuracy to within one rank.

Granular Effects

A key granular effect that emerged after the introduction of the ordinal techniques was the network's tendency to predict almost no rank 4s. This is likely another result of the challenges the network faces when trying to differentiate between ranks 2-4. It would seem that the network minimises its MSE loss function by predicting high values in the rank '3' range when it cannot clearly choose between rank 3 and 4. Whilst this may appear detrimental to network performance, for our purposes its not a substantial problem. Further investigation can be carried out to determine whether the network predicts true rank 4s as higher output values than true rank 3s, and bin edges moved to capture more rank 4s. If there is a correlation between higher rank 3 values and true rank 4s, then a hierarchy distribution for a real race can be populated by rank ordering the runners according to this.

This finding further reinforces the hypothesis that the input data used may not contain sufficient informational detail for a network to regularly distinguish between ranks 2-4.

Further Work

This investigation concludes our series of studies into the application of machine learning algorithms to race horse raking prediction. We have successfully demonstrated that these techniques can be used to model and predict all types of racing, across all courses with acceptable degrees of accuracy. Further work for Quantum Leap solutions in this area will focus on product implementation and optimisation over time. A good place to start would be studying the distribution of prediction values for ranks 3 and 4 to determine if a manually developed classification step could better distinguish between them.

There are many opportunities for us to apply the principles explored in these studies to other problems within race analysis and prediction. Future investigations will likely focus on statistical analyses of input variables, other feature prediction and developing deep insights into potential runner performance.

Conclusion

The introduction of ordinal regression techniques made little difference to the ability of the models to predict rankings exactly. For the combined flat model, exact accuracy improved by 0.1% compared to our previous classification type network, whilst for the jumps model it fell by 0.2%. Without conducting numerous repeat studies where all networks are retrained on the data from scratch, it is impossible to determine the degree to which this is due to stochastic processes associated with machine learning methods. To within one rank of error both models saw improvement, the flat model by 1.2% and the jumps model by 1.1%. These values are more substantial, and therefore more likely to indicate that the introduction of ordinal regression techniques led to some meaningful improvement in network predictive accuracy.

Going forward, ordinal type classification problems will likely be common for Quantum Leap given the nature of horse racing prediction and analysis. We have demonstrated in this study that ordinal regression techniques should be considered and tested whenever we tackle one of these problems in the future, as it may lead to material network accuracy improvements.

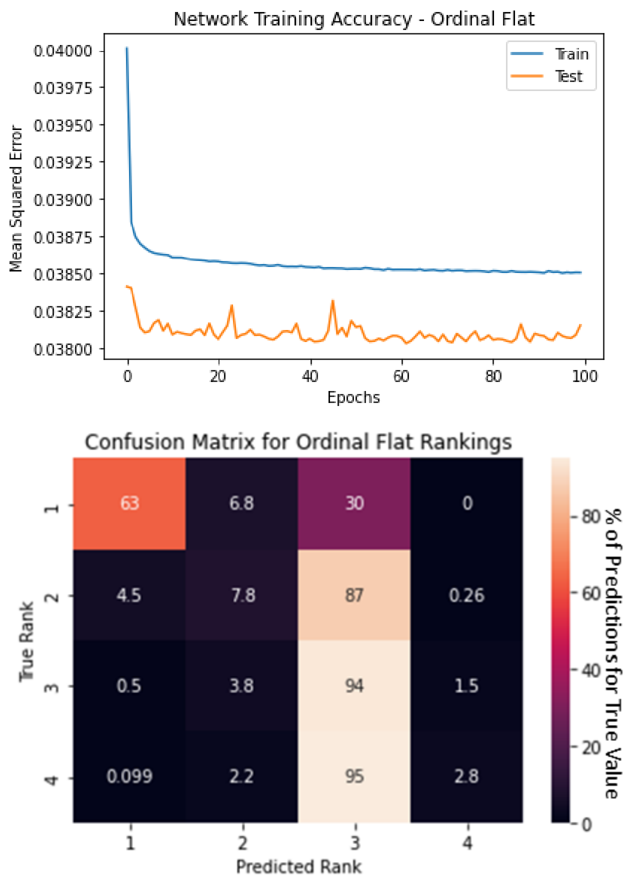

Figure 2: Training and test data accuracy of the regression network for flat racing, and the confusion matrix for its classified predictions. The difference between the value tended to by the training accuracy and the test accuracy is small, but indicates that the test set may have been slightly easier to predict than the training set.

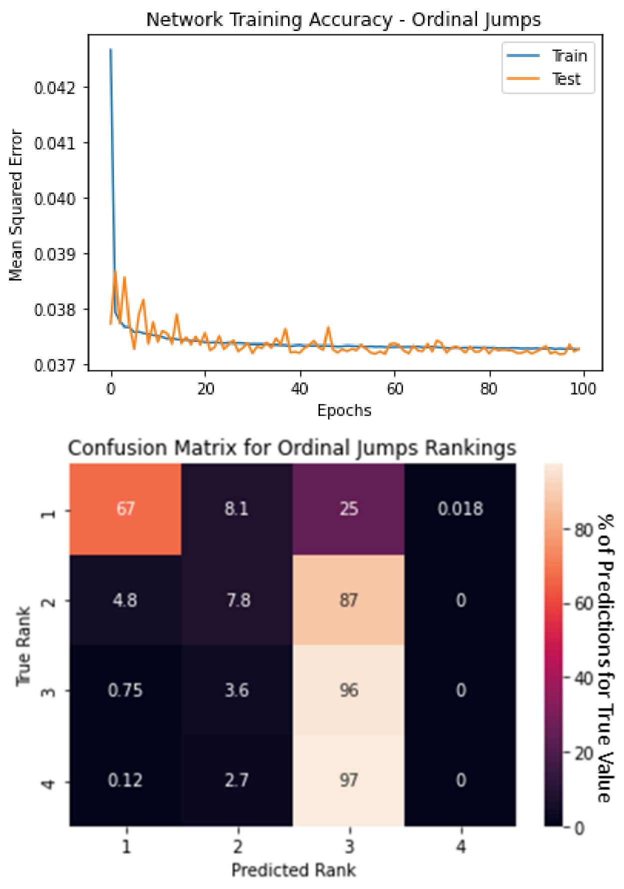

Figure 3: Training and test data accuracy of the regression network for jumps racing, and the confusion matrix for its classified predictions.

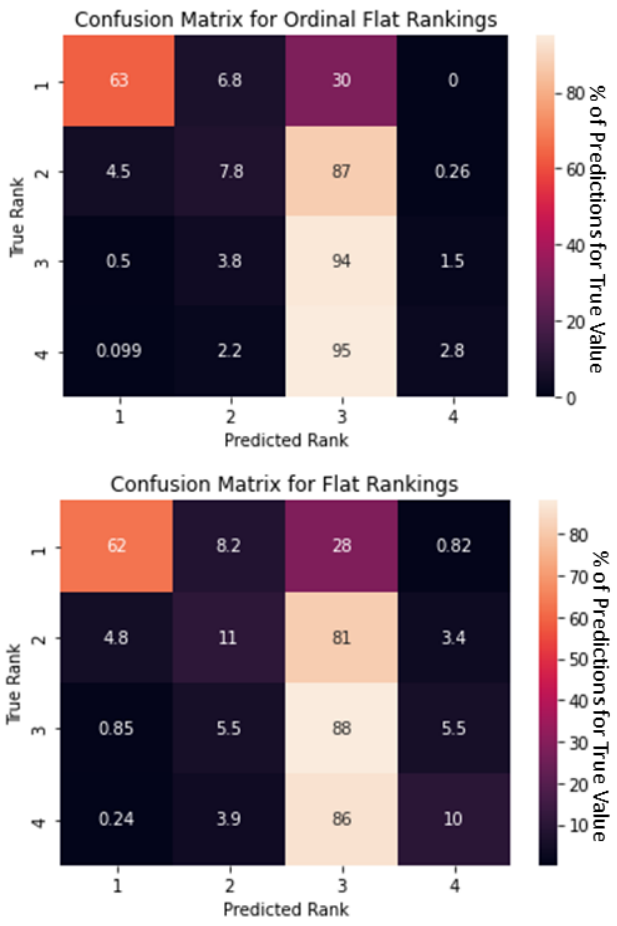

Figure 4: Confusion matrices for ordinal and non-ordinal flat models.

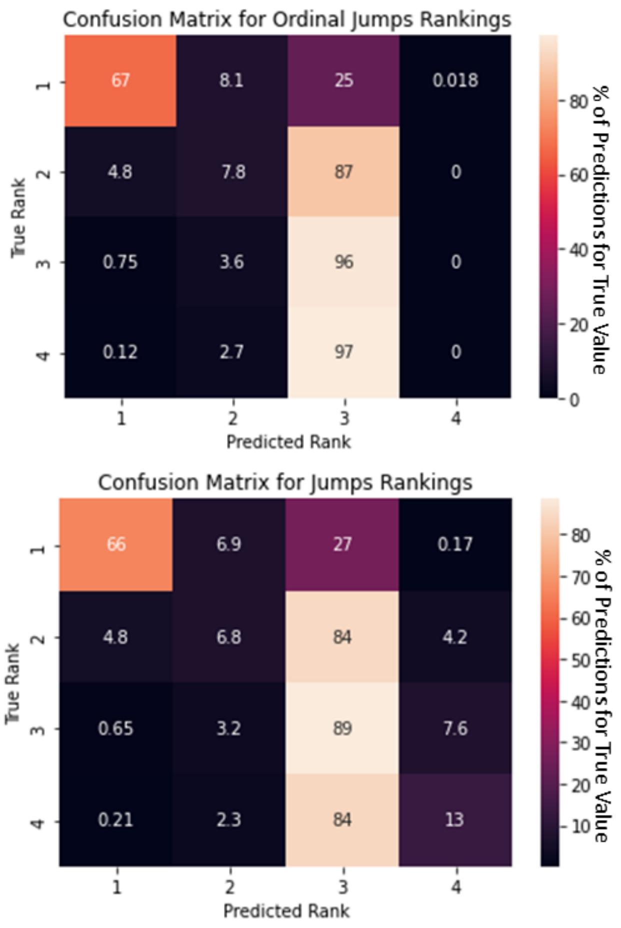

Figure 5: Confusion matrices for ordinal and non-ordinal jumps models