Predicting Tennis Match Outcomes With Neural Networks

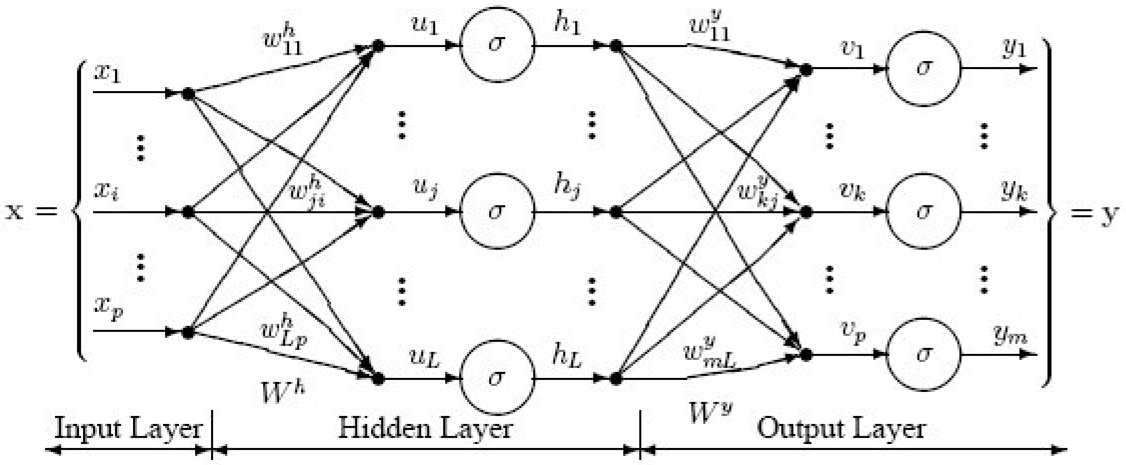
John Cook

Abstract

This paper draws from the models created by Amornchai Somboonphokkaphan, Suphakant Phimoltares, and Chidchanok Lursinsap in their paper Tennis Winner Prediction based on Time-Series History with Neural Modeling. This paper will present three neural networks of increasing complexity. The networks will accurately predict match outcomes by using match and player statistics.

Introduction

Tennis is a great sport for prediction by neural networks. As an individual sport we can use in-depth statistics for the players without drastically increasing the number of inputs. In addition the sport is organized by tournaments each consisting of many rounds. Thus, we will have more games to draw statistics from than other individual sports such as golf. Due to the popularity of Tennis, match results and associated betting lines are well documented. Betting lines will be used as a metric to evaluate the best model, beyond traditional error metrics. In-depth match statistics were found at "https://github.com/JeffSackmann". The availability of these in-depth statistics is crucial to the final model. Each neural network model proposed in this paper is a Multi-Layer Perceptron, meaning that each layer is fully connected to the next one.



For each model a single hidden layer model is tested. For the final 2 models a double hidden layer model is also tested for comparison purposes. For model standardization, each models inputs are calculated from 2011-2014 data. The models are then trained and evaluated on 2015 matches, randomly split into a training and a test set. The inputs are assigned either to the first player, the second player, or the court type. First player statistics are inputed first, followed by second player statistics and finally by the court type. The models output a single number, where a value greater than .5 is interpreted as the first player winning, and a value less than .5 as the second player winning. The models will be trained(and evaluated) using one of two error functions, mean squared error(sse for training)* or cross-entropy. While MSE is the most interpretable metric for this application a model's cross-entropy score can be useful in future applications. The error metrics are discussed further in the next session.

*note: the network is trained according to the sum squared error. Mean squared error is chosen as an error metric as it is more interpretable. Mean squared error is the sum squared error divided by the number of observations.

The Error Metrics

Each model will be trained with a certain amount of hidden layers and with a sum squared error or cross-entropy error function. The models will then be evaluated using mean squared error and cross-entropy. The squared error metrics are the most interpretable. The MSE statistic represents the proportion of wrong classifications from the testing set. This is the most important metric for this paper as each output is classified based on being above or below .5. However, one could classify by any number of rules. A possible classifier could be only classifying the matches with output away from .5, such as discarding any matches with an output between .4 and .6. In this case training the neural network on sum squared error may not provide the optimal network. The cross-entropy error metric accounts for how far away the predicted value is from the true value. Thus, if the true value is 0, a prediction of .9 would be worse than a prediction of .6. By minimizing cross-entropy we can hope to keep outputs for mis-classifications around .5 and thus, we may adjust our classification metric to ignore values close to .5. Further exploration into cross-entropy and different classification techniques are beyond the scope of this paper.

Model 1: Basic Model

The first model consists of eight inputs. For each player the inputs are: number of wins, number of losses, games played and win percentage. Four single layer network models were tested. While the MSE of the models were within .05 of each other the cross-entropy error metric ranged dramatically. The table of results are below:

```
##
## Model 1 Results
## ============================================
## model hiddenlayers errorfct  MSE     ce
## --------------------------------------------
## 1          10        sse    0.390  648.230
## 1          10         ce    0.370  328.570
## 1          20        sse    0.370 1,808.180
## 1          20         ce    0.420  786.970
## --------------------------------------------
```

Model 2: Court Model

The second model builds upon the first. It consists of fifteen inputs: the eight from the basic model, games played on court type and winning percentage on court type for each player, as well as three binary classifications for the three possible court types(clay, grass, hard). Thus for any given match one of the inputs of clay, grass and hard will be 1 and the other two will be zero. Model 2 provides a clear inprovement both in MSE and ce(despite the one ce score of 328 from Model 1). Thus, it is clear that our knew inputs lend predictive power to our model. In addition, further manipulation of the hidden layers is needed to determine if they have a positive effect. The results are below:

```
##
## Model 2 Results
## ==========================================
## model hiddenlayers errorfct  MSE    ce
## ----------------------------------------
## 2          10       sse    0.200 324.340
## 2          10        ce    0.190 285.820
## 2          20       sse    0.140 432.550
## 2          20        ce    0.150 455.680
## 2         10,5      sse    0.180 395.190
## 2         10,5       ce    0.150   605
## 2         20,5      sse    0.140 338.850
## 2         20,5       ce    0.140 572.490
## ----------------------------------------
```

Model 3: Advanced Stat Model

The third model builds upon the second. For each player, four in-depth statistics were added: winning % on first serve, winning % on second serve, winning % on return serve, winning % on break point. The accompanying file CalculatingStats.docx explains in detail the process and assumptions needed to calculate the statistics. Thus, there were 23 inputs. While most of the model 3 results do not appear significantly better than Model 2's results, the double hidden layer of 30 and 10 model has the lowest MSE as well as very low cross-entropy scores. Due to my classification method(spliting by over/under .5), I choose the model with a .11 MSE for additional testing(which is described in the section below). The results for Model 3 are shown below:

```
##
## Model 3 Results
## ========================================
## model hiddenlayers errorfct  MSE    ce
## ----------------------------------------
## 3          10       sse    0.160 321.540
## 3          10        ce    0.180 474.640
## 3          20       sse    0.150 305.730
## 3          20        ce    0.170 493.310
## 3          30       sse    0.180 389.300
## 3          30        ce    0.140 365.330
## 3         30,10     sse    0.110 330.120
## 3         30,10      ce    0.120 253.690
## ----------------------------------------
```

Additional Testing

While using mean squared error and cross-entropy can help us choose the most accurate model they cannot tell us how much "new information" is being gained by using the neural network. In order to test whether or not the final model has substantial predictive ability it can be tested against tennis betting lines from sportsbetting companies. If the model can make a positive net profit pseudo-betting, then the model can be classified as a very good predictor as it is able to outpredict most sports gamblers. For the test all input statistics were calculated from the data between 2011-2014. The Model 3 neural net with 2 hidden layers of 30 and 10 nodes respectively was trained on 2015 match data occurring before August 31, 2015, the day the US Open began. Although 127 matches were played, the neural network only had full information(all inputs available) on 35 matches. Of the 35 matches 31 were correctly predicted. If $100 was placed on each of the 35 matches the resulting net profit would have been $1,289.66. Suprisingly, the neural network correctly predicted every situation in which the underdog(not the favorite) won. The neural network collected the vast majority of the final profit from correctly predicting 3 large underdogs, an 8 to 1(bet $1 and get $9 back), a 5 to 1, and a 2.5 to 1 underdog. Its ability to correctly predict large underdogs lends credence to the idea that the model has better predictive power than most people involved in and around the sport of tennis.

Conclusion

Overall, each consecutive model improved the mean squared error and cross-entropy scores compared to its simpler counterparts. This paper also provides evidence that multiple hidden layers improve the results of sophisticated neural networks more than simpler ones. However, more testing must be done to substantiate this claim. The final model was able to achieve a 89% classification rate. It was also able to gain a 10x profit pseudo-betting on the US Open. Potentially, additional tuning and more inputs(such as statistics from a players last ten matches) could improve the models performance. Future projects could analyze the effect of different classification techniques in order to achieve the highest MSE but also the greatest profit when testing the model by pseudo-betting. In this case, the model would achieve both a low MSE and a low cross-entropy, as a low cross-entropy score would mean misclassifications are both rare and their predicted values are close to .5. For summation, the results for all the models are displayed below:

```
##
## Complete Results
## ===========================================
## model hiddenlayers errorfct  MSE     ce
## -------------------------------------------
## 1          10       sse    0.390  648.230
## 1          10        ce    0.370  328.570
## 1          20       sse    0.370 1,808.180
## 1          20        ce    0.420  786.970
## 2          10       sse    0.200  324.340
## 2          10        ce    0.190  285.820
## 2          20       sse    0.140  432.550
## 2          20        ce    0.150  455.680
## 2         10,5      sse    0.180  395.190
## 2         10,5       ce    0.150    605
## 2         20,5      sse    0.140  338.850
## 2         20,5       ce    0.140  572.490
## 3          10       sse    0.160  321.540
## 3          10        ce    0.180  474.640
## 3          20       sse    0.150  305.730
## 3          20        ce    0.170  493.310
## 3          30       sse    0.180  389.300
## 3          30        ce    0.140  365.330
## 3        30,10      sse    0.110  330.120
## 3        30,10       ce    0.120  253.690
## -------------------------------------------
```