CS230

Tennis Match Predictions Using Neural Neworks

Mitchell Dumovic , Trevor Howarth
Stanford University
mdumovic@stanford.edu, thowarth@stanford.edu

Abstract

Accurate tennis match predictions have the potential to have a huge impact on sports betting markets. We use a deep neural network leveraging Long-Short-Term-Memory (LSTM) cells to predict the winners of tennis matches given a dataset of various statistics on every ATP men's singles match since 2000. Our goal was to use this deep network to match or improve upon the accuracy of existing cutting edge prediction methods. With the final network, we were able to predict the winner of tennis matches with approximately **69.6%** accuracy on the test set, a **4%** improvement over a naïve prediction which predicts the winner entirely based on who had the higher rank. This prediction accuracy matches the best results we found in literature on the subject, but does so without the use of hand engineered features.

1 Introduction

The projects authors are both very passionate about tennis, and wanted to use the tools that we have learned in CS230 to build a model that predicts the winners of tennis matches. Due to the large number of professional matches played each year, tennis betting makes up one of the largest betting markets in the world. Odds makers have used a wide variety of statistical models to predict the outcome of these matches, many based on metrics such as performance of players against common opponents[1] and stochastic models considering the probability that each player wins individual points. Although these traditional methods have been able to predict the outcomes of tennis matches with relatively high accuracy, the author's goal was to find out whether or not neural networks can be used to improve this performance. The simple statistical models used in the past are limited by their creator's beliefs about what decides a tennis match. By implementing a neural network for prediction, the network would be able to take advantage of the unexpected correlations neglected by traditional methods.

In order to make a prediction of the winner of a tennis match, our neural network architecture takes in basic information about the match being predicted (the ranks of both players, the surface on which the match is being played, etc.) as well as statistics such as points won and serving percentages from the last 50 matches played by each participant in the match. These statistics are fed into the neural network with LSTM, Dropout, and Dense fully connected layers in order to predict a binary classification (0 or 1) as to which of the two players won that match.

2 Related work

A previous CS229 final project[2] evaluated how well various machine learning algorithms predict the outcome of professional tennis matches. The machine learning tools that they evaluated included a random forest, neural network with a single hidden layer, SVM, and logistic regression. The best results for each of these approaches yielded $69.7, 65.2, 69.9$, and 69.9 percent test accuracy. While this group tried a number of machine learning approaches, the authors hypothesized that a more complex network could produce better results.

Somboonphokkaphan[3] also attempted to train a neural network to predict the winners of tennis matches, trying several different network architectures with different numbers of nodes and hidden layers. Their best performing network had three layers and 27 input nodes representing features for both the players and the matches, and it achieved an average accuracy of about 75% in predicting the outcomes of matches in the 2007 and 2008 Grand Slam tournaments.

Sipko[4] attempts to use neural networks to predict what bets should be made on tennis matches for maximum return on investment, as opposed to simply predicting the winners of matches. Here, Sipko achieves a ROI of approximately 4.35% when in competition with the betting market. Sipko attempts to use both logistic regression and a neural network in order to place bets on tennis matches, and finds that the neural network results in the best return on investment.

3 Dataset and Features

The ATP (Association of Tennis Professionals) provides a dataset containing information about every official ATP men's tennis match since 1968[5]. For each row in the dataset, which corresponds to one real tennis match, there are statistics such as the winner's and loser's age, rank, handedness, height, as well as per match statistics such as the score, percentage of first serves won, percentages of break points won, etc. These matches are sorted by date and contain information on the tournament and surface on which they were played as well.

This dataset required significant processing before we had something we could feed to our first pass neural network. We found that a large portion of the raw statistics were missing for the earlier matches in the dataset, so we ultimately restricted our model to only matches played after the year 2000, for which most matches had complete statistics.

In our data pipeline, we took each match in the database and generated a match specific feature vector as well as a series of feature vectors representing the 50 previous matches played by each participant in the match. The match specific feature vector contained information on the age, height, rank, and handedness of each player, as well as the surface on which the match was played (this was encoded as a 1-hot vector across the four possible surfaces, grass, clay, carpet, and hard). The feature

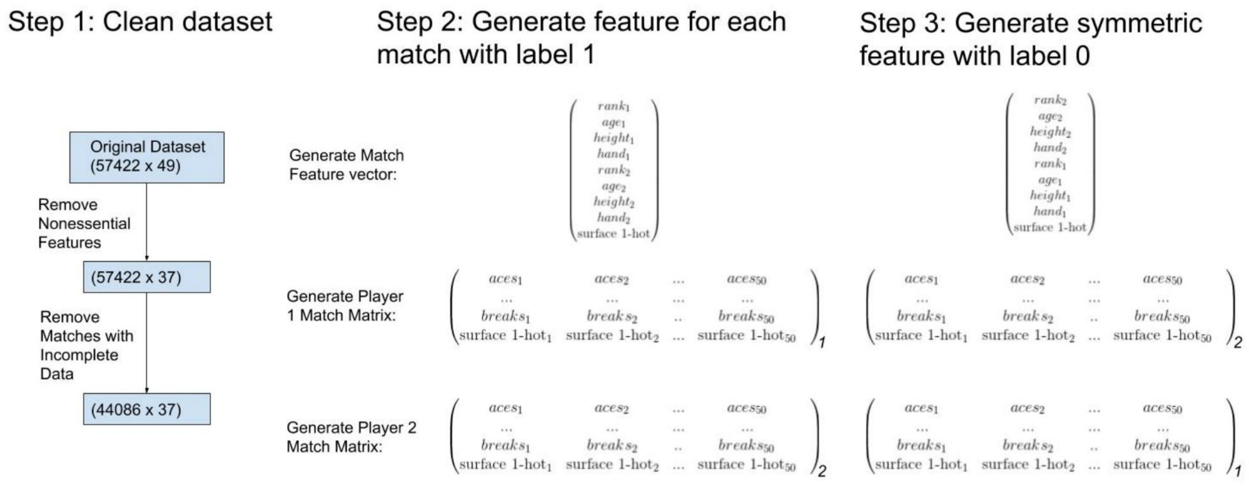
Step 1: Clean dataset
Step 2: Generate feature for each match with label 1
Step 3: Generate symmetric feature with label 0
Original Dataset (57422 x 49)
Remove Nonessential Features
(57422 x 37)
Remove Matches with Incomplete Data
(44086 x 37)
Generate Match Feature vector:
Generate Player 1 Match Matrix:
Generate Player 2 Match Matrix:
rank1 age1 height1 hand1 rank2 age2 height2 hand2 surface 1-hot
rank2 age2 height2 hand2 rank1 age1 height1 hand1 surface 1-hot
aces1 aces2 ... aces50
breaks1 breaks2 .. breaks50
surface 1-hot1 surface 1-hot2 ... surface 1-hot50

Figure 1: *The full data processing pipeline.*

vector representing the a previous match played by a given player contained match statistics (serving percentages, total points won, etc.) as well as whether the player won the match, which surface the match was played on (once again encoded in a 1-hot vector), and the amount time between the match and the current match being considered.

To improve the quality of our dataset, we only tried to predict matches where both participants had statistics available from at least 25 previous matches. If a player had at least 25 matches played, but not 50 matches played, then we padded the data with zeros as features for matches that were missing. We recorded information on no more than 50 previous matches in order to reduce the size of our dataset and reduce the time to train the network. We chose 50 matches as our cutoff because analysis of the dataset showed that players rarely had more than 50 previous matches played within the scope of our dataset. The previous match statistics were stored in a matrix for each player and these, along with the match feature vector, served as the features for each match. We labeled each of these matches with either a 1 (player 1 winning) or a 0 (player 1 losing). We created two symmetric data points for each match in our database, one labeled 1 and one labeled 0. Augmenting the data in this way allowed us to have an even mix of wins and losses in our dataset.

After we generated our feature csv files from the raw matches datasets, we divided our features into 95% training set (74202 examples), 2.5% dev set (1952 examples), and 2.5% test set (1952 examples) .

4 Methods

We initially approached the problem with a simple feed-forward neural network. This model was built in tensorflow and was used as a baseline against which to evaluate our other more complex models. This network had 2 hidden layers in the form LINEAR -> RELU -> LINEAR -> RELU -> LINEAR -> SIGMOID. The first hidden layer had 25 neurons, and the second had 12. For the cost function, we used cross entropy loss.

For this first pass network, we used the **averaged** statistics from the last 50 matches played by each player rather than statistics from individual matches. The averaged feature vectors for each player as well as the match feature vector (which contained information on age, height, rank, etc.) were concatenated and used as the input.

During training, we divided the train set into minibatches of size 32, and trained using a learning rate of 0.0001 for 100 total epochs. For our final predictions, we thresholded the output of the final sigmoid layer at .5 and said anything greater or equal to that corresponded to player 1 winning (a label of 1) and anything less than that corresponded to player 2 winning (a label of 0).

After implementing the baseline, we wanted to create a model that could be fed raw statistics from past matches rather than the averaged features we used for our baseline model. Averaging the statistics from all the past matches throws away a lot of potentially useful information, and we hoped to create a model that could glean more information from the past matches competed in by each player.

In our first attempt to avoid using averaged features, we simply concatenated the statistics from all of the past matches into one large input feature vector. Each past match was represented by 11 features and we used 50 past matches for each player, giving us a dense input vector with more than 1000 elements. We fed this to a 4 hidden layer feed forward neural network with the architecture: LINEAR -> RELU -> LINEAR -> RELU -> LINEAR -> RELU -> LINEAR -> RELU -> LINEAR -> SIGMOID. The first hidden layer had 200 hidden units, the second had 100 units, the third had 50 units and the fourth hidden layer had 10 units. We trained this model with minibatches of 32 and a learning rate of 0.0001 and used a prediction threshold of .5 for the output of the final sigmoid unit.

The four layer neural network described above ultimately performed significantly worse than our baseline algorithm and it became clear that a more nuanced approach was required. The authors hypothesized that a long short-term memory (LSTM) recurrent neural network could be used to generate an "encoding" of each player for the purpose of prediction. LSTM networks are useful for dealing with sequence data, especially data that occurs forward in time over unknown time intervals. These networks effectively keep a "memory" of previous inputs to the network, and the authors believe that this notion of memory is especially helpful for taking in match statistic data, where each LSTM network "remembers" the information from the output of the previous LSTM cell in the network and combines it with new input information to create an updated encoding.

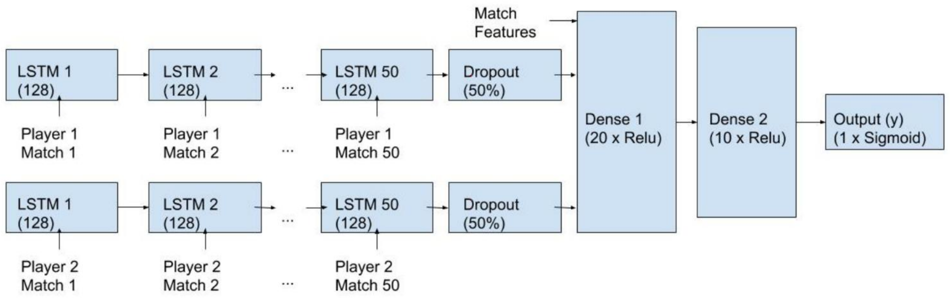

Figure 2: *The architecture of the LSTM network.*

A diagram of the LSTM model, our best performing neural network is illustrated in Figure 2. We start by feeding in data from the past 50 matches through an LSTM network with 128 output nodes. This layer outputs a length 128 vector that represents an "encoding" for a player. Both players in the to-be-predicted match are fed through the same LSTM layer, generating encodings for each of these players. We then take these two player encodings, concatenate them with the match-specific features described in section 3, and input this single concatenated vector into a fully connected layer that has 20 neurons and a 'ReLu' activation. This is in turn fed into a full connected layer with 10 neurons with a 'ReLu' activation, and finally goes through the output layer with a sigmoid activation in order to produce a single output between 0 and 1. If this output is greater than or equal to 0.5, a win is predicted. Otherwise, a loss is predicted. As with our other networks, the LSTM model was trained on an Adam optimizer with minibatches of 32 and a learning rate of 0.0001. Back-propagation was run through this entire network in order to train the weights in the network in order to minimize the binary crossentropy loss on the data:

$$\mathcal{L} = -\sum_{i=1}^{m} \hat{y_i}_i log(y_i) + (1 - \hat{y_i})(1 - log(y_i))$$

5 Results

A summary of the results for our various models can be seen in Table 1.

As a baseline metric for comparison for our dataset, we checked the accuracy in predicting a match's winner by predicting solely on which player had more ATP rank points (i.e. was ranked higher). This baseline achieved **65.6%** accuracy on our test set.

Model	Test Accuracy
Naïve Rank-Based Prediction	65.6%
Feed-forward network with averaged Match statistics	64.4%
Feed-forward network with concatenated match statistics	53.2%
LSTM (Figure 2)	69.6%

Table 1: Test accuracy of our three approaches alongside a Naïve rank-based prediction scheme

After 100 epochs of training on our training dataset, our baseline model achieved **64.4%** prediction accuracy on the held out test set.

Our first attempt at predicting matches without averaged features fared much worse, achieving only **53.2%** prediction accuracy on the test set after 100 epochs of training. Even after training many variations on this architecture (networks with different numbers of layers and different numbers of hidden units per layer), and searching a large range of hyperparameters, we could not achieve better accuracy. Although a four layer fully connected neural network is theoretically able to approximate any function, we believe we did not have enough data to train a network with this large of an input space. This network architecture did not implicitly model the relationship between different features, meaning that in order to make sense out of the many statistics we passed as input it would have to learn that some subset of features represented aces for player 1 from past matches while another subset represented aces for player 2 from past matches. Our dataset was not big enough to allow the network to infer these relations so it performed poorly. Additionally, this network had no built-in way to understand the time ordering of past matches, making the task of prediction even more difficult.

Our LSTM model addressed the many shortcomings of our first attempt at making predictions from raw statistics. The LSTM layer automatically models the fact that the statistics vector from each past match is of the same format and is designed to comprehend time series data like ours. Additionally, the fact that the past matches from both players are fed through an identical LSTM layer forces the network to consider each player's statistics on equal ground. These structural advantages allowed our LSTM network to achieve an accuracy of **69.6%** on the training set after 100 epochs of training. Figure 3 shows the test set accuracy of the LSTM network versus epoch over this training period. While training, we found that no matter the hyperparameters used, our network almost always seemed to converge around this value of 70%, which is quite similar to the maximum values for accuracy we found in related literature. This suggests that Bayes error is close to 30% for this prediction problem, at least with the data given by our dataset.

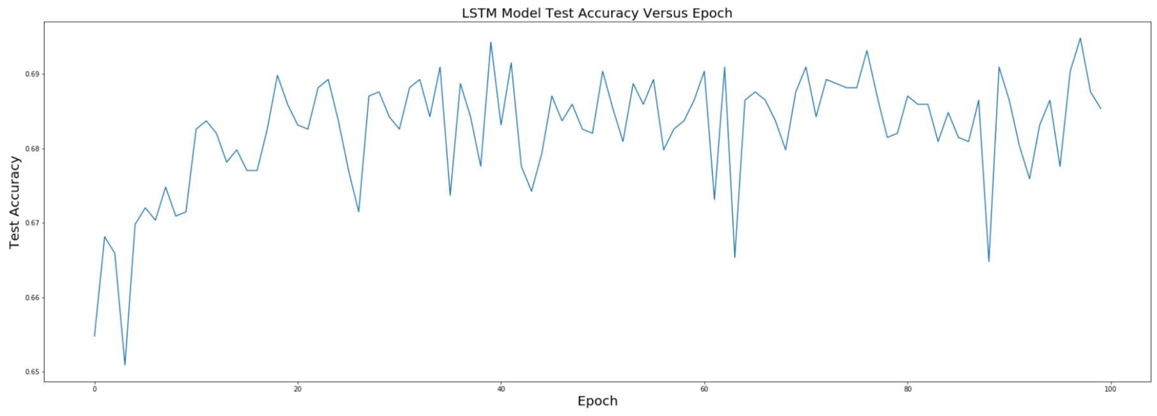

Figure 3: *The accuracy of the LSTM network over 100 epochs of training.*

6 Conclusion

The authors designed a variety of neural network architectures to predict the result of a tennis match from the statistics of past matches played by each player in the match as well as information about each player at the time of the match such as age and rank. Although moderate accuracy was achievable with a baseline model that used hand engineered features (the averaged statistics over the past 50 matches played by each player), the best performing model used raw statistics from each past match to make predictions. Using an LSTM to create an encoding for the performance of each player over the past 50 matches they had played allowed us to model more complex relationships in the time series data than simple averages. This model achieved an 4% greater test accuracy than a naïve rank-based prediction scheme, putting it in line with the highest prediction accuracy seen by many of the models in literature on the subject. Although a 4% improvement may not seem massive, this could make a large difference in one's performance against the betting markets in which the margins are often quite narrow.

Going forward, we believe that our LSTM model is a sensible way to approach the problem of tennis match prediction and that a dataset containing more features would be required to improve predictions significantly. The performance of tennis players is notoriously inconsistent and many external factors can have a major effect on a player's mental state and the ultimate outcome of the match. A dataset containing features including, for example, weather at the time of each match, the size of the crowd, and the time of day at which the match was played could allow our encoding to better represent how a player responds to a given match environment and ultimately make a more accurate prediction.

7 Contributions

Both team members worked on all parts of the project. We pair coded all of the files in the project, and made all decisions about the network architecture, data pipeline, and project strategy together.

8 References

[1] W. J. Knottenbelt, D. Spanias, and A. M. Madurska. *A common-opponent stochastic model for predicting the outcome of professional tennis matches*. Computers and Mathematics with Applications, 64:3820–3827, 2012.

[2] Cornman, A., Spellman, G. and Wright, D. (2017). *Machine Learning for Professional Tennis Match Prediction and Betting*. [pdf] Available at: http://cs229.stanford.edu/proj2017/final-reports/5242116.pdf [Accessed 10 Jun. 2018].

[3] A. Somboonphokkaphan, S. Phimoltares, and C. Lursinsap. *Tennis Winner Prediction based on Time-Series History with Neural Modeling*. IMECS 2009: International Multi-Conference of Engineers and Computer Scientists, Vols I and II, I:127–132, 2009.

[4] Michal Sipko. " textitMachine Learning for the Prediction of Professional Tennis Matches. MA thesis. Imperial College London, 2015.

[5] https://github.com/JeffSackmann/tennis_atp