Tennis Match Predictions Using Neural Nets

Trevor Howarth, Mitchell Dumovic

Motivation

We 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 gambling markets in the world. Simple statistical models used for tennis match prediction in the past were limited by their creator's beliefs about what decides a tennis match. A neural network is able to take advantage of the unexpected correlations neglected by traditional methods.

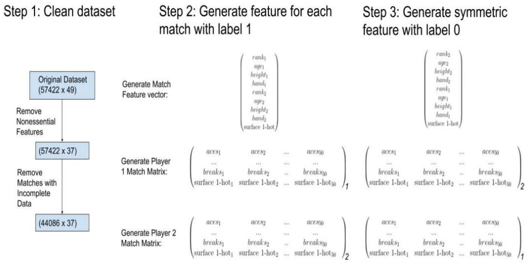
Data

All of our data comes from a public GitHub repo which accumulates statistics from every professional men's ATP tennis match since 2000. Some of the most important features are listed below. After culling some matches with insufficient data, we ended up with data on approximately **50,000** matches.

Rank
Handedness
1st serves won
Height
Aces
2nd serves won
Age
Double Faults
Service games won
Surface
1st serves in
Break points saved

Features

Features are generated for each match by gathering data for each player over the last 50 matches that they played. Additionally, we include per-match features such as rank, height, age, and handedness to predict match results.



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:

$$\begin{pmatrix} rank_1 \\ age_1 \\ height_1 \\ hand_1 \\ rank_2 \\ age_2 \\ height_2 \\ hand_2 \\ \text{surface 1-hot} \end{pmatrix} \qquad \begin{pmatrix} rank_2 \\ age_2 \\ height_2 \\ hand_2 \\ rank_1 \\ age_1 \\ height_1 \\ hand_1 \\ \text{surface 1-hot} \end{pmatrix}$$

Generate Player 1 Match Matrix:

$$\begin{pmatrix} aces_1 & aces_2 & \dots & aces_{50} \\ \dots & \dots & \dots & \dots \\ breaks_1 & breaks_2 & .. & breaks_{50} \\ \text{surface 1-hot}_1 & \text{surface 1-hot}_2 & \dots & \text{surface 1-hot}_{50} \end{pmatrix}_1 \qquad \begin{pmatrix} aces_1 & aces_2 & \dots & aces_{50} \\ \dots & \dots & \dots & \dots \\ breaks_1 & breaks_2 & .. & breaks_{50} \\ \text{surface 1-hot}_1 & \text{surface 1-hot}_2 & \dots & \text{surface 1-hot}_{50} \end{pmatrix}_2$$

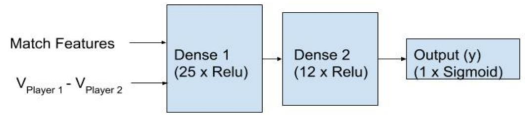
Generate Player 2 Match Matrix:

$$\begin{pmatrix} aces_1 & aces_2 & \dots & aces_{50} \\ \dots & \dots & \dots & \dots \\ breaks_1 & breaks_2 & .. & breaks_{50} \\ \text{surface 1-hot}_1 & \text{surface 1-hot}_2 & \dots & \text{surface 1-hot}_{50} \end{pmatrix}_2 \qquad \begin{pmatrix} aces_1 & aces_2 & \dots & aces_{50} \\ \dots & \dots & \dots & \dots \\ breaks_1 & breaks_2 & .. & breaks_{50} \\ \text{surface 1-hot}_1 & \text{surface 1-hot}_2 & \dots & \text{surface 1-hot}_{50} \end{pmatrix}_1$$

Models

Baseline Model:

Our baseline model was a **two hidden layer feed-forward network** that took match features (player rank, age, etc.) as well as a single feature vector representing the past match performance of both participants as inputs. To generate a vector representing past match statistics, we took the average of the available statistics over the past 50 matches for each player (V) and subtracted $V_{\text{Player 2}}$ from $V_{\text{Player 1}}$. This model used an **Adam optimizer** with **Xavier initialization** and a learning rate of **.0001**.



Match Features

$V_{\text{Player 1}} - V_{\text{Player 2}}$

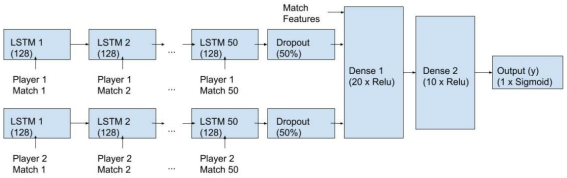
Dense 1 (25 x Relu)

Dense 2 (12 x Relu)

Output (y) (1 x Sigmoid)

LSTM Model:

The second architecture we implemented used an **LSTM** layer to create an **encoding** vector for each player from the last 50 matches they played. The statistics vectors for each of the previous 50 matches for each player were fed to the same LSTM layer. We fed the outputs of each LSTM layer into a **dropout** layer, which gave a regularizing effect and greatly increased test accuracy. The encoding from the LSTM layer from player 1 was concatenated with the encoding for player 2, as well as with the match specific features. This vector was then fed to a two hidden fully connected layer feed-forward network which made the final prediction. This model used an **Adam optimizer** with **Xavier initialization** and a learning rate of **.0001**.



Match Features

LSTM 1 (128) · LSTM 2 (128) ··· LSTM 50 (128) · Dropout (50%)

Player 1 Match 1 · Player 1 Match 2 ··· Player 1 Match 50

LSTM 1 (128) · LSTM 2 (128) ··· LSTM 50 (128) · Dropout (50%)

Player 2 Match 1 · Player 2 Match 2 ··· Player 2 Match 50

Dense 1 (20 x Relu)

Dense 2 (10 x Relu)

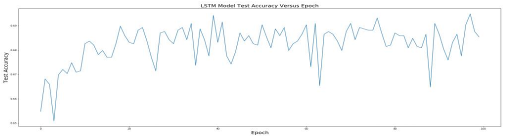
Output (y) (1 x Sigmoid)

Results

Our baseline model achieved **64.4%** accuracy on the test set and our LSTM model achieved **69.6%** accuracy on the test set. This accuracy was achieved in both after 15 epochs of training and then performance plateaued. A naïve model that predicted entirely on who has more ATP rank points got **65.6% accuracy.**

Discussion

- With our given data, it was not trivial to predict match results with more accuracy than a simple naïve strategy based on rank. Even the best models we tried and found in literature do not perform much better.
- The accuracy of 70% achieved with our LSTM model is quite close to the maximum accuracy we found in many of the other projects on tennis match prediction we reviewed. While tuning hyperparameters, we found that it was very difficult to get above this threshold, and our test accuracy would almost always converge to about this value. This suggests that with the data available, 30% is close to the Bayes' error for this prediction problem and we will need significantly better data to improve predictions.
- The poor performance of the baseline model likely comes from the fact that the sequential relationship between statistics from each of the past matches is not factored into the model architecture. This also suggests that averaging statistics is not a reasonable way to encode past match performance for the purpose of predicting a winner.
- In all networks trained, test set accuracy plateaued after 20 epochs of training. Even in models with larger numbers of hidden units or more layers, additional epochs of training did little to help. This suggests that to achieve greater accuracy, a larger dataset with a greater number of features would be required.
- Going forward, we believe our LSTM model is a reasonable way to attack the problem of tennis match prediction. We would like to find a dataset that has a greater number of features per match and more matches with complete data. We believe our model could leverage a better dataset to predict with significantly improved accuracy.



References

- 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.
- Michal Sipko. Machine learning for the prediction of professional tennis matches. Meng computing – final year project, Imperial College London, June 2015.
- 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.