A Sure Bet: Predicting Outcomes Of Soccer Matches

Sebastien Goddijn, Rohan Challa, Evgeny Moshkovich
Stanford University, CS 230

Background

Competitive sport is a numbers game, and modern athletic teams are truly beginning to embrace this fact. We are seeing a surge of data driven jobs in the back office of myriad different franchises, and people are beginning to wonder just how effective cutting edge AI and ML techniques can be when applied to an athletic context. Sports have always been a purely human endeavor, and there is an inarguable element of randomness and chance that dictates who the victor will be on any given day, but this begs an interesting question: is there an underlying pattern to this randomness?

For our project we focused specifically on matches in the Premier League, as this is the league that we have watched most closely growing up, and has detailed data that is already publicly available due to the quality of the teams involved. We took individual player ratings, team's current record for the season and win/loss streak as our input for both the home team and the away team. The goal was to create an output which will be the outcome of the game in terms of a home win, draw, or away win.

Features And Outputs

Dataset:
We got the data from Kaggle as a SQLite database, we extracted team/player/match data and stored them in separate CSVs, to manage the data better. We split the data 80/10/10 percent for train, dev, test (2432/304/304)

Input features:
The input features varied slightly in the different models which we tried out. Altogether the features included:

•Individual player positions and FIFA ratings for both teams in each game
•Current record (win/loss/draw)
•Current streak

Output:
The output was defined as a one hot vector where (1,0,0) is a home win (0,1,0) is a draw and (0,0,1) is an away win

Logistic Regression

The baseline model which we tried first was a simple logistic regression. The input only contained 22 features – individual player ratings for both teams.

We used the standard logistic regression architecture where the output was defined by a softmax function. The model was run for 50,000 epochs with a learning rate of $1e^{-6}$

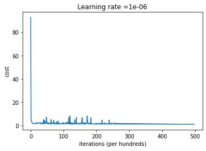
Dev Set Confusion Matrix:

	HOME	DRAW	AWAY
HOME	16	26	36
DRAW	32	22	33
AWAY	30	44	65

Neural Network

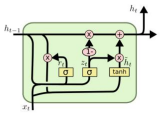
The second model we ran was a 3 layer neural network. For each layer we used the RELU activation function.

For each hyperparameter the optimal value was selected by choosing the one which yielded the highest dev set accuracy:
•Learning rate - $1e^{-6}$
•# Neurons per layer – (27,9,3)
•Number of epochs – 30,000



Sequence Model

The final model that we implemented was an LSTM sequence model. We added 8 more input features – current team record for both teams (win/loss/draw) and the current streak for both teams. There were 160 different combinations different team and season in total. We used multiple regularization techniques including dropout and Adam to prevent our model from overfitting.



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Results

The main parameter we used to appraise our models was accuracy. Below is the breakdown of how the three different model performed on both the training and test sets.

Model	Training Accuracy	Test Accuracy
Logistic Regression	0.357	0.339
3 Layer Neural Network	0.549	0.513
Sequence Model	0.524	0.474

The model that performed the best was the 3 Layer neural network. Running the LSTM model did not seem to improve the accuracy. The state of the art accuracy that we inferred from the literature review is somewhere in the region of 70% so there is definitely improvements that can be made.

Conclusions And Future Work

To conclude, we were unable to achieve high accuracy in our test set, due to a number of factors. Firstly, though the LSTM was able to achieve high accuracies on our training set, this didn't translate to high accuracy at test time, due to the tendency it had to over-fit our data. Though we attempted some regularizing techniques, such as dropout, we were unable to see large improvements in our test accuracy because of this. If we were to continue work on this, we would firstly gather more data to use, as our dataset was relatively small compared to the number of football games we could have been training on. Secondly, we would spend some more time determining how to sequence this data such that our LSTM's accuracy on our training set could translate to the test accuracy. We would also run some more experiments to determine which features were most applicable to our classification task.