COGS 185 Final Project Report
Raghav Kansal
A92121242

Predicting soccer results using sequential models

Abstract

This project tested the effectiveness of using the sequential Hidden Markov Model (HMM) and Recurrent Neural Network (RNN) machine learning algorithms in predicting the results of soccer matches given a previous sequence of results for each team. This sequential approach in principle allows the models to consider temporal information about the team's results, i.e. the team's 'form', when predicting new results. This gives an opportunity to study how advantageous a sequential approach is as opposed to a non-sequential model for predicting soccer results, and how significant a team's form is when predicting their future performances.

A neural network non-sequential baseline gave a prediction training accuracy of 37% and testing accuracy of 39%. The HMM got similar training and testing accuracies of 40% and 38% respectively. As opposed to this, the RNN was able to get training and testing accuracies of 65% and 54% respectively. This illustrates the RNN's superiority over the HMM, that indeed a sequential model is a better predictor of match outcomes, and also that indeed a team's form is significant in determining its future results.

Intro

The significance of a team’s form in future performances is a subject of great debate in soccer. Intuitively it may seem that a team recently performing well would have a high likelihood of continuing to do so, however statistically there is not a lot to back this notion up [1]. ‘Form’ is inherently a temporal and sequential characteristic, so to test this notion, and simultaneously the effectiveness of the models, this study used sequential models to predict team results based on previous performances.

Sequential models are those which can incorporate temporal information about the inputs when performing e.g. classification or regression. For sequential inputs, where later observations are dependent on those that come before, these models are superior to traditional non-sequential models such as neural networks and support vector machines. In this project two popular sequential models, the HMM and RNN, were applied on this problem and compared to each other and to the performance of a non-sequential baseline neural network.

HMMs are designed using the Markov assumption, that the likelihood of a system being in a particular state at a discrete time t is dependent solely on the system’s state at time t-1. Using this assumption the HMM (in problems such as what this project is studying) takes as input a sequence of observations and calculates the transition and emission probabilities of “hidden states” at every discrete timestep, corresponding to each observation. These can then be used to predict future observations.

An RNN is a essentially a neural network with ‘memory’, in that it can retain information about previous inputs when calculating the output for later inputs, which makes it ideal for sequential data. The RNN in this project was implemented with long short-term memory (LSTM) cells, each one of which is composed of an input, forget and output gate, which can increase the RNN’s effectiveness.

Methodology

Dataset

Soccer data is notoriously difficult to work with and parse because of the ever-changing format of competitions throughout Europe. This means it was important to define a precise problem statement and preprocess the data accordingly. The problem each model is solving is: given sequences of 20 dimensional vectors, where each entry in the vector corresponds to a single team's game result (the encoding of the result will be discussed shortly), and where the order of the vectors in the sequence correspond to the order in which the games were played, predict the next 20 dimensional vector, corresponding to the next game result for each of the 20 teams.

Football.csv [3] was chosen as a suitable dataset for this purpose; it contains the results of every single match for the past 20 or so seasons in various European leagues. One issue of concern was that not every season involved 20 teams, as countries constantly change their league formats and vary the number of teams participating. Truncating or padding the number of teams by removing or randomly generating respectively the games involving specific teams was considered but ultimately decided against because the intrinsic information within a single observation, in which each of the 20 teams has played against another, would be lost or at minimum distorted. It was therefore decided to just include the seasons which had 20 teams (20 was chosen because the current format of most leagues is a 20-team one, and because there were more 20 team seasons in the dataset than any other number). This left 64 seasons to work with, each with 38 matches (each team plays every other twice in a season) so a total of 2,400 observations.

Parsing the observations was also non-trivial as, while in most cases all 20 teams played within a single week, and ideally the games within the week would constitute a single observation, many times a team's game would be delayed or rescheduled such that in some weeks fewer than or more than 20 teams played (some teams playing no or two games in a week). To deal with this all the games of each individual team in a single season were stored in order in a matrix where the j^{th} row of the matrix corresponded to the j^{th} team's results, and the i^{th} column corresponded to the i^{th}, out of 38, game the teams had played. Each column therefore was a single observation and each matrix, corresponding to a single season, was a sequence.

Two different encodings of the results were tested: the first was a simple ternary encoding with 1 for a win, 0 for a draw and -1 for a loss. The second was the the goal difference of a game, i.e. if team A beat team B 5-3, the entry for team A would be 2 and team B would be -2. The latter encoding shall be referred to henceforth as the 'non-ternary' encoding.

Sequences

To perform a comprehensive analysis of the effectiveness of sequential input, all possible sequence lengths were tested. This means all models were tested on how well they could predict the next results given i previous results, where i ranged from 1 to 37.

Training

The 64 seasons were split into 45 for training and 19 for testing (roughly 70/30 split).

Two different types of neural networks were implemented, one with the 20D input observations, and the second with 1D input observations with each team's results as individual observations, to see which was more effective. For each input type, 37 different neural networks were trained, one for every possible input sequence length. 37 copies of the training data were made, split into sequences of 1, 2, 3, … 37 observations respectively, each sequence with a single output observation (the next game after the input sequence). Since there are 38 games in a season, the longest sequence that can trained is 37, otherwise there would be no output to train on. The observations within a sequence were then concatenated into a single vector for the input, and each neural network was trained and tested individually. The best results out of all these was chosen as the baseline to compare the sequential models' results with. The sklearn library's MLP regression neural network was employed.

The HMM was trained simply by inputting all 45 training seasons/sequences, and assuming Gaussian emission probabilities, using the hmmlearn python library.

The RNN was trained using the same training set as for the neural network, with 37 copies of the training data split into sequences of differing lengths. However, instead of concatenating the observations in a single sequence, they were input one at a time into the RNN to preserve the temporal information. The RNNs were implemented using Keras and its inbuilt LSTM cells.

Hyperparameters

The 20D vs 1D input observations for the baseline, the input sequence length and the type of encoding are already hyperparameters in a sense, and are being varied as explained above. In addition, for the neural network, layer sizes of 30, 50, 100 and 200 neurons were tested. For the HMM, models with 30, 50, and 100 and 200 hidden states were tried. Finally, for the RNN, layers of 30, 50, 100 and 200 LSTM cells were used.

Accuracy

Accuracy was measured in two ways. The output for the ternary encoding is always a continuous value between -1 and 1. The standard accuracy check involved rounding the output value to -1, 0, or 1 and if the rounded value did not match the ground truth it was considered an error. The second metric was the Euclidean distance between the predicted result (no rounding) and the ground truth which in some sense is a better indication of the accuracy since predicting a win instead of draw is worse than predicting a win instead of a loss; a win is "closer" to a draw than a loss.

For the non-ternary encoding, the Euclidean measure is the same idea (and indeed measuring Euclidean distance was also motivated by the use of the non-ternary encoding). The Euclidean measure cannot be used to compare models with ternary and non-ternary encoding since non-ternary encoding inherently has larger values and so will naturally have a higher Euclidean error. The standard accuracy check still only considers wins and losses, since predicting the score of a game was not the intention of the project (which is why the results and conclusion emphasize the accuracy more than the Euclidean measure which is more an interesting byproduct). When measuring the accuracy, an output of greater than 0.5 is considered a win (i.e. a '1'), an output of less than -0.5 is considered a loss, and anywhere in between is considered a draw. If the outputs and the ground truths differ then it is considered an error.

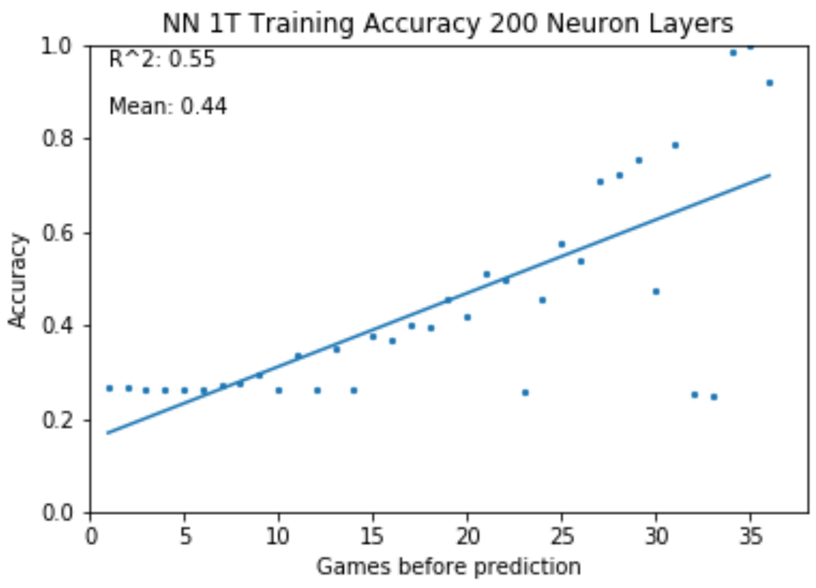
Experiment

All the spreadsheets and plots are presented in the project .zip.

Baseline

The different non-sequential tests had wildly varying results. Firstly the standard neural network with a ternary encoding and 20D input achieved the highest accuracy with a 30 neuron layer, which had an average training and testing accuracy of 30% and 29% respectively. The 1D inputs all had testing accuracies of around 30% but training accuracy rising to ~100% at large sequence lengths. The neural network was clearly overfitting here because of the small sample size of large sequences, and so these results were not at all statistically significant. Because of these poor and unusual results in the 1D case the 20D input vectors were used thenceforth.

Fig 1. Neural network overfitting



The most successful test was using the non-ternary encoding and 20D input which, produced a training and testing accuracy of around 38% and 39% respectively and was invariant to the input sequence length and the size of the layers. This is the baseline that shall be used for comparing with the sequential models.

A useful sanity check on top of the baseline neural network is to see how well we can predict the scores with the absolutely most basic statistics. In the whole training dataset, wins and losses occur 36.7% of the time each, while draws occur 26.5% of the time. This means if you were to simply predict the outcome with the highest likelihood, a win (or a loss), every time, you would have a training accuracy of 36.7% and a testing accuracy 36.2%. Our baseline model does just slightly better than this, indicating its effectiveness, or rather the lack thereof.

HMM

Unlike the neural network, the encoding had little effect on the accuracy of the HMM. With the ternary encoding, increasing the number of hidden states improved the accuracy, while for the non-ternary, increasing the states had no effect. As one would expect, because of the Markov assumption employed in a HMM, the input sequence length had next to no correlation to the accuracy.

Fig. 2 Typical HMM Accuracy

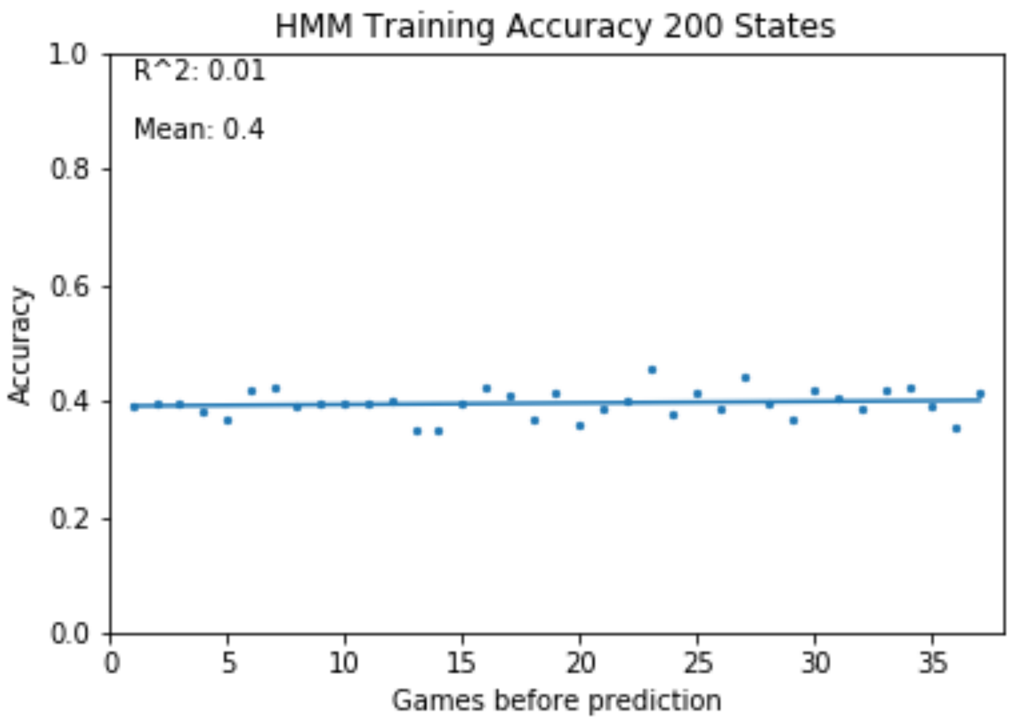


Fig 2. shows the typical (lack of) correlation between the HMM accuracy and the number of games inputted before a prediction. The R^2 value of 0.01 indicates no correlation whatsoever.

The highest accuracy achieved was an average training and testing accuracy of 40% and 38% respectively, with the ternary encoding and 200 hidden states. This is almost exactly the same accuracy as with the baseline and indicates that the HMM is not very effective for this problem.

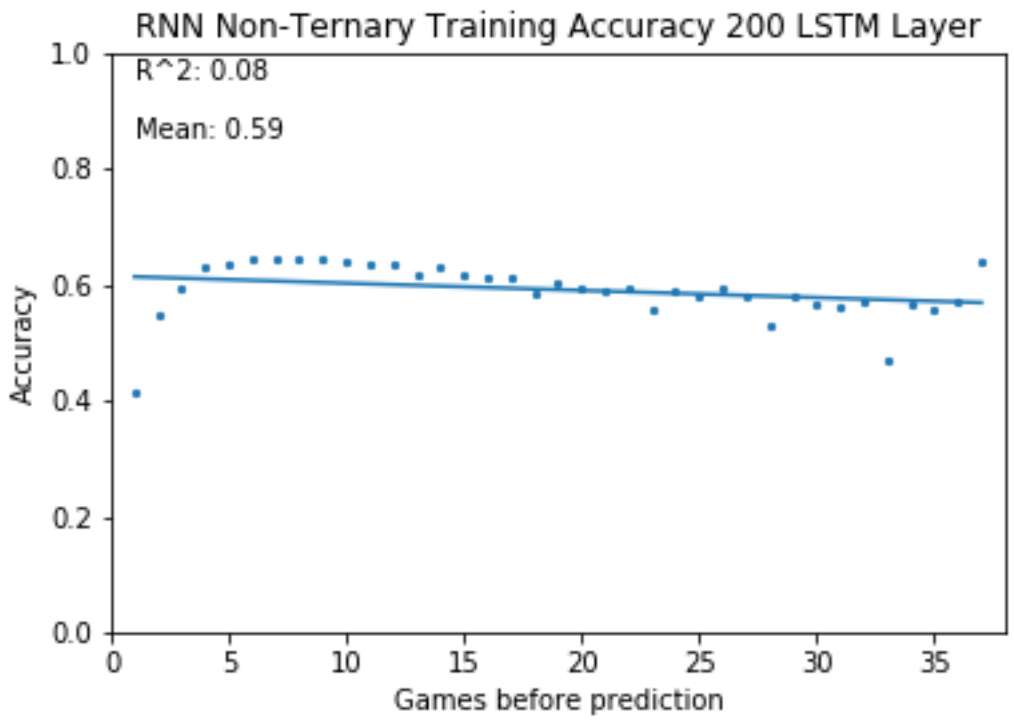
RNN

Like the neural network the RNN had a significantly higher accuracy when using the non-ternary encoding. The accuracy also increased as the number of LSTM cells increased. With 200 LSTM cells, the ternary encoding had an average training and testing accuracy of 44% and 39% respectively, whereas with the non-ternary encoding the average training and testing accuracy was 59% and 50%. The highest total accuracy occurred for an input sequence length of 8 with a 65% training accuracy and a 54% testing accuracy. It also had the lowest average Euclidean error of 1.10 for the training set and 1.17 for the testing set, which means on average it was only off on the goal difference of the game by 1 goal.

Since the RNN is not limited by the Markov assumption, one would expect an increase in accuracy as the sequence length increases. This was the case in the experiment, to an extent. Fig. 3 shows the training accuracy increasing rapidly initially, until settling after a sequence length of around 5. The accuracy appears to decrease at longer sequence lengths, however this is likely due to the sample size for longer sequence lengths being smaller.

The RNN was significantly more effective in predicting match outcomes than both the baseline and the HMM.

Fig 3. Correlation of RNN Accuracy and Sequence Length



Conclusion

This project demonstrated the effectiveness of using the sequential RNN model in predicting soccer results, illustrating as well the significance of a team’s form in determining their future results. With an extremely limited and small dataset, fairly high testing accuracy was achieved. This indicates that this model is a good platform to build on for predicting match results with high accuracies, wherein the accuracies can be improved by using a larger dataset and involving more features in the input, such as player injuries, managerial records etc.

The large difference in accuracies for an RNN and HMM also gives insight into the significance of a team’s form (as well as showcasing the superiority of an RNN over an HMM). Clearly considering only the previous match when predicting the outcome of the next match is not very useful, as indicated by the low accuracy of the HMM and also of the RNN at very short sequence lengths; both were similar to the accuracy of the baseline. Only after using sequences of 5 games or longer was there significant improvement in the RNN, meaning while one match is not useful, the sequence of the previous five or so matches does in fact help predict the outcome of the next match for a team.

References

[1] Tan, Brandon. “Form in Soccer: Not Always a Winning Formula.” *Princeton Sports Analytics*, Princeton Sports Analytics, 20 Sept. 2017, princetonsportsanalytics.com/2016/06/12/form-in-soccer-not-always-a-winning-formula/.

[2] Dietterich, T. G. (2002). Machine Learning for Sequential Data: A Review**.** In T. Caelli (Ed.) Structural, Syntactic, and Statistical Pattern Recognition; Lecture Notes in Computer Science, Vol. 2396. (pp. 15-30). Springer-Verlag

[3] https://github.com/footballcsv