PREDICTING GOALS SCORED FOR FOOTBALL MATCHES

Arish Dutt – ad428642@stanford.edu

1. Introduction

Soccer is the most played and watched sport in the world. Over 200 countries have a national team with the industry being worth billions of dollars. This leads to a huge betting market, not just in regards to which team will win a match but also what the score will be.

The purpose of this project is to predict the number of goals a team will score in a match. A soccer match is always played between a home team and an away team. The aim is to predict how many goals each team will score in the match. In the models these are separate predictions (one prediction for the number of goals the home team scores and another for the number of goals the away team scores). Taken together these 2 predictions give you the predicted score line for the match.

The input to generate this prediction is the performance of the team in previous matches (e.g. goals scored, shots taken). I also take into account the defensive prowess of the team they are playing against (e.g. goals conceded, shots conceded).

The predictions are team agnostic, i.e. they do not take into account the specific teams playing, rather just looks objectively at their past results

2. Related work

Karlis et. al (Karlis & Ntzoufras, 2003)[2] and Rue and Salvesen (Rue & Salvesen, 2000)[3] took into account dependence in areas where I assumed independence, the former in regards to the correlation between the number of goals the home team scores in a match with the number of goals the away team scores, the latter created a "skill" metric that was dependent on all other teams in the league. Karlis et. Al was less focused on the predictions and more on how well a bivariate Poisson could be fit to the data. They also put strong emphasis on the benefits of playing at home, which I also considered. Rue and Salvesen considered both the attacking skill of a team and the defensive skill of the opposition, which I also incorporated into my work.

Egidi, Leonardo & Torelli, Nicola & Pauli, Francesco. (2018)[5] looks into combining both historical performance and bookmakers odds to provide a prediction for footballing scores. My aim was to use solely historical performance, it is interesting to note though that incorporating bookmaker's odds does add predictive power. This is expected since the bookmaker's odds take into account not only the knowledge of experts making the odds but those confident enough to make a bet on the score of a match.

Another interesting related paper is Mchale, I.G. & Scarf, Phil. (2011)[6]. In my work I assumed that the number of goals a team scores in a match is independent of the number of goals the opposing team scores. This paper shows that this is largely true except for uncompetitive matches, e.g. one team is far better than the other. There are no predictions made here, rather this paper is interesting in that it shows how the attacking performance of one team can be impacted by the attacking performance of the other.

There are also many papers that look to predict the outcome of a match. For example Goddard and Asimakopoulos (2004)[4] use an ordered probit regression model. They not only take into account historical results but also the time of the season and geographical distance between the 2 teams' towns. Instead of predicting goals scored, they looked to test the efficiency of bookmaker's prices.

3. Dataset and features

For my predictions the essential data that is needed is on a team level, specifically a team's past results. This was available at football-data.co.uk[1]. It has data not only on results but also on goals, shots, shots on target, corners and many other useful statistics for every match from about 2004. All these attributes are useful as they paint a fuller picture of the performance of both

teams in match, relative to just looking at the final score.

I also looked into incorporating player specific data into the models. Unfortunately features based on such data was not incorporated into the final models because:

- Reliable data for this does not exist for many years. The most accurate source I could find only went back 3 years
- After incorporating team level data from the above source, the features I created did not add significant predictive power (discussed later)

Because of this, the final set of features were all at a team level

Many different team level features were tested. Initially features were removed using a combination of the GLM and GBM (discussed in detail later in the paper)

After that features were removed using a version of backward search that used RMSE as the error metric

The final list of features used in the model:

- Goals scored in the last x matches (x =1, 5) by the attacking team
- Shots taken in the last x matches (x =1, 5) by the attacking team
- Shots on target in the last x matches (x =1, 5) by the attacking team
- Corners earned in the last x matches (x =1, 5) by the attacking team
- Goals conceded in the last x matches (x =1, 5) by the defending team
- Shots conceded in the last x matches (x =1, 5) by the defending team
- Shots on target conceded in the last x matches (x =1, 5) by the defending team
- Corners conceded in the last x matches (x =1, 5) by the defending team
- Indicator variable if the attacking team is playing at home (1 if home, 0 if away)
- Goals scored (attacking team) / conceded (defending team) in the last 3 home / away matches
- Shots taken (attacking team) / conceded (defending team) in the last 3 home / away matches
- Shots on target (attacking team) / conceded (defending team) in the last 3 home / away matches
- Corners earned (attacking team) / conceded (defending team) in the last 3 home / away matches

I also tested other lags (besides the 1, 3 and 5 shown above). These were removed as they were overall less predictive (discussed later)

Attacking team refers to the team for whom we are predicting goals scored for, defending team is the team they are playing against.

All these features were created by me (feature creation too the bulk of the time for this project). Many other team related features and player related features were tested but were removed since they did not add significant predictive power. Some examples of removed features:

- Total goals scored in last x matches by the attacking team's starting lineup (x =1, 5)
- Goals scored in last x matches by the attacking team's leading scorer in the starting lineup (x =1, 5)
- Average goals scored per match in that season so far

5 years (seasons) of data from the English Premier League was used. There are 20 teams playing 38 games a season each, leading to 380 games. Given we are predicting goals scored for both teams, each season gives us 720 observations. Thus overall we have 3,600 observations.

I split the data into 3 sets: training, validation and test (60/20/20 split). Training was used to train parameters, validation to decide on hyperparameters and test to test final accuracy.

Data was scaled before being modelled. For feature x, observation i:

$$x^i = \frac{x^i - \min(x)}{\max(x) - \min(x)}$$

This was also done for the label

4. Methods

Since the label is a count we need to use algorithms that allow us to model counts. The 3 below algorithms can (once tailored) be used to do so

Gradient Boosted Machine (GBM)

A type of machine learning algorithm[8] that uses decision trees that sequentially learn from one another. A decision tree is first fit to the data and the next tree is fit to the residuals (they become the labels) from the prediction of the previous tree, and so on. The number of trees built is a hyperparameter, along with the depth of each tree. Subsequent trees predict observations that were predicted poorly by the previous trees. Predictions from the final ensemble model are therefore the weighted sum of the predictions made by the previous trees.

You also need to specify an objective function which determines the loss function that the algorithm will try and minimize. In my case the objective function was "count: Poisson". The loss function used is Root Mean Square Error:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}}$$

A learning rate also has to be specified. Every time a new tree is built, the predictions are updated using this learning rate. In class we generally have dealt with algorithms with parameters, where the parameter is updated using gradient descent. In GBMs, to perform the gradient descent procedure, an additional decision tree is added that aims to reduce the loss function, i.e. follow the gradient. The xgboost package on R was used[12]

Neural network

The error function used is the sum of squared errors

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

The activation function used is the softplus[9]:

$$f(x) = \log(1 + e^x)$$

This is a smooth approximation of the rectified linear unit (ReLU) function (which could not be used itself with the R package neuralnet since it is not cannot differentiate it). The neuralnet package on R was used[11]

Generalized linear model (GLM) with regularization

Here a poisson GLM is fit, in the same way as discussed in lectures. A regularization term is added, specifically the penalty is:

$$(1 - \alpha)/2||\beta||_2^2 + \alpha||\beta||_1,$$

Where the alpha used is 0.5. The alpha is chosen by the user, specifically an alpha of 1 is a lasso regularization while an alpha of 0 is a ridge regularization. This term is also multiplied by λ, which is picked via cross validation

The regularization term is to help minimize the chance of overfitting to our training set and thus giving us the best possible prediction on our test set. The Poisson GLM is used since the label is a count and thus can be said to come from a Poisson. The glmnet package on R was used[10]

5. Experiments/Results/Discussion

To ensure I could appropriately compare each model type, the features used across them had to be consistent.

I started with a very large list of potential features that could feed into each of the model (discussed in the Dataset and Features section). To whittle them down in an effective manner, I used a combination of the GBM and the Poisson GLM (without regularization).
A key diagnostic that is available with the xgboost package is "feature importance", this will tell you how often each feature was used and how predictive it was. Features that added virtually no predictive power were removed.
GLMs not only give you an estimate of the coefficients of each feature but also give a standard deviation which can be used to compute a p-value. After removing the clearly unnecessary features using xgboost, I also used a p-value cut-off of 0.02 to remove features using the GLM.

After that I ran a standard backward search using RMSE to remove any more features (using the GBM) and arrived at a list of the most predictive features
I wanted consistency in terms of the number of matches I looked back into the past for features. It was clear that the most predictive were generally

looking back either 1 match or 5 matches (e.g. goals in the last 5 matches, shots taken in the last 1 match) so I kept the features that did this and removed all others. For features that focused on home or away matches the most predictive were usually those that looked back 3 matches so I kept those and removed all others.

After arriving at my final list of features I ran them through my 3 models. I compared the models using RMSE

Gradient Boosted Machine
Key hyperparameters:

- Learning rate: 0.05
- Tree depth = 3
- Number of trees = 130

I arrived at these hyperparameters through a grid search. First I fixed the learning rate and tree depth (0.05 and 4 respectively) and found the number of trees that led to the lowest RMSE. Once that was fixed I ran a similar grid search for tree depth. RMSE was measured on the validation set after training on the training set

Neural net
Key hyperparameters:

- Hidden layers: 1, 1
- Activation function: softplus

I tested both the sigmoid activation function and the softplus and found that in general the softplus gave a lower error measure on the validation set. From there I tested many different permutations of hidden layer values and found a single hidden layer with 2 neurons gave the lowest error metric

GLM
The alpha value needed for regularization was set to 0.5 since I wanted to build a model that was in between a lasso and ridge regression.
A λ value also has to be chosen. This was done by k-fold cross validation (k = 10)

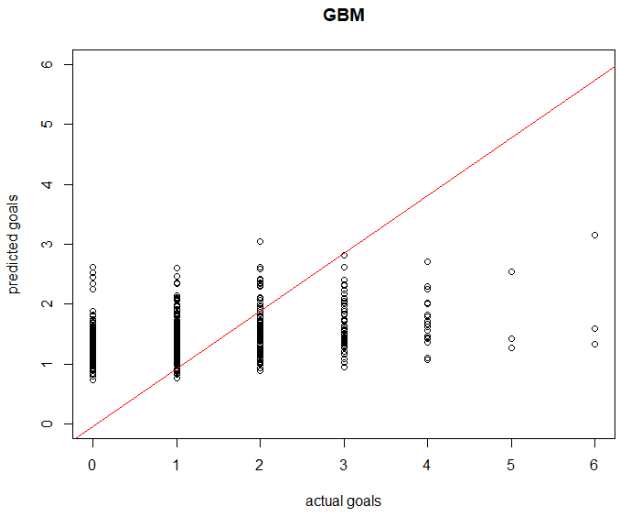
Results:
The results in terms of RMSE

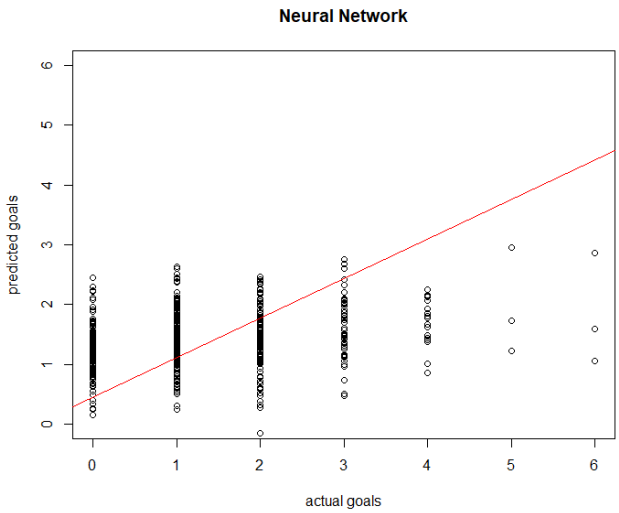
Model	Training error	Test error
GBM	0.157	0.159
Neural Network	0.162	0.164
Poisson GLM	0.158	0.162

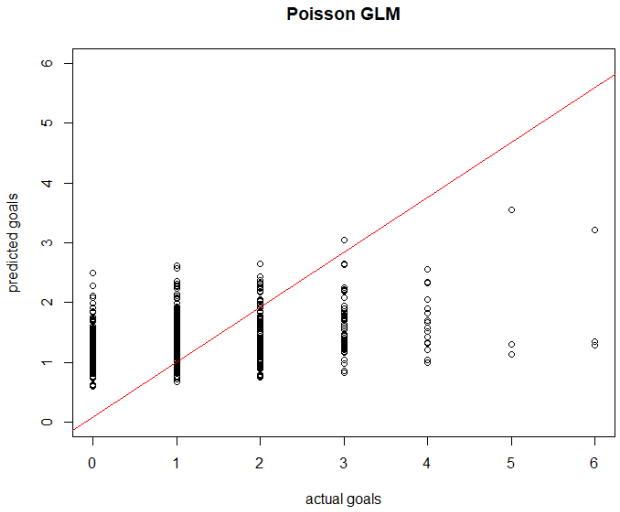
Overall the GBM performed the best, followed by the GLM and then the Neural Network.

Neural Networks anecdotally generally outperform other methods when there are many different types of features; it can find hidden relationships between features and the label better than most other algorithms to deliver the best possible prediction. Given that for this project the features were whittled down early on in the process, this may be why it performed the worst out of the 3 methods.

Plots of actual vs observed with a line of best fit:







The results show that across all models that there is over prediction when actual goals = 0 and under prediction when actual goals >= 3. This would suggest that the features that were built into the model were not sufficient to accurately predict the tail ends of the distribution of goals

What is also seen in the results is that the models perform significantly worse when predicting games at the beginning of the season (defined as within the first 10 games for a team within a 38 game season).
As a soccer fan I would hypothesize that this is because at the beginning of the season teams are still working out their tactics / approach; during the break between seasons teams hire new players / staff and so performance week to week is quite volatile. Also statistically some of the features used look at the last 5 games so they are not as useful until more than 5 game have occurred.

Some interesting notes about the features:

- In both the GLM and GBM the indicator variable came out as one of the most important features suggesting that home advantage is a real phenomenon
- Shots taken and shots conceded in previous matches were far more predictive than goals scored / conceded. The former is less volatile and a better representation of the strength of the team

Overfitting was avoided across the GBM and Neural Network via the use of a validation set to set hyperparameters, the test set was only used right at the end of the process to make the final predictions. For the GLM, regularization was used to avoid overfitting, with cross validation being used to come up with the lambda used in this process. Again the test set was left alone until it came time to make the final predictions

6. Conclusion / Future work

Out of all the models, the GBM performed the best.
All of the models unfortunately predicted the lower and higher ends of label distribution poorly. I believe this is due to features used / my approach to feature reduction. Given that soccer is a low scoring game (teams score < 3 goals in over 80% of games), being able to predict when a team will score over >= 3 goals would likely require additional features / feature interactions.

Future work would involve creating player specific features. I did build some basic player related features that did not add much predictive power but I believe there are other features that I did not have the time to test that could have. Incorporating the specific players and their attributes into a model may give a better insight into when one team will score several goals. Examples of such features:

- Total price of starting line up according to Fantasy Soccer[7]
- Incorporating subjective 3rd party ratings regarding players (e.g. from Fantasy Soccer) into the model
- What proportion of players that played last match will play this current match (would give an information regarding injuries)

An additional avenue I would have liked to have explored is how well the Neural Network would have performed before I had whittled down all the features. I was hoping to build my final models parsimoniously, using only the features that were the most predictive. Unfortunately this might have meant I eliminated some features that would have added some value via their interactions with other features, and it might have been possible to see this by building a neural network.
In general my approach to this project was to focus on only the features that were the most predictive, in the future it would be interesting to see how much accuracy improves (especially at the tail ends of the label distribution) if all the features had been kept / additional player specific features had been added.

GITHUB CODE:
https://github.com/arishd/CS_229

REFFERENCES

1. Historical football results and betting odds data. *http: //www.football-data.co.uk/data.php.* Accessed: 2019-05-10.
2. Karlis, Dimitris and Ntzoufras, Ioannis. Analysis of sports data by using bivariate poisson models. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 52 (3):381–393, 2003.
3. Rue, Havard and Salvesen, Oyvind. Prediction and retrospective analysis of soccer matches in a league. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 49(3):399–418, 2000.
4. Goddard, John, and Ioannis Asimakopoulos. "Forecasting Football Results and the Efficiency of Fixed-Odds Betting." *Journal of Forecasting, vol. 23, no. 1*, 2004
5. Egidi, Leonardo, et al. "Combining Historical Data and Bookmakers' Odds in Modelling Football Scores." *Statistical Modelling, vol. 18, no. 5-6*, 2018
6. Mchale, Ian, and Phil Scarf. "Modelling the Dependence of Goals Scored by Opposing Teams in International Soccer Matches*." Statistical Modelling: An International Journal, vol*. 11, no. 3, 2011
7. Vaastav Anand, Fantasy-Premier-League, (2019), GitHub repository, https://github.com/vaastav/Fantasy-Premier-League/tree/master/data
8. Brownlee, J. (2019). *A Gentle Introduction to XGBoost for Applied Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
9. 4. Sefik Ilkin Serengil. (2019). *Softplus as a Neural Networks Activation Function - Sefik Ilkin Serengil*. [online] Available at: https://sefiks.com/2017/08/11/softplus-as-a-neural-networks-activation-function/ [Accessed 11 Jun. 2019].
10. Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.
11. Stefan Fritsch, Frauke Guenther and Marvin N. Wright (2019). neuralnet: Training of Neural Networks. R package version 1.44.2. https://CRAN.R-project.org/package=neuralnet
12. Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng and Yutian Li (2019). xgboost: Extreme Gradient Boosting. R package version 0.82.1. https://CRAN.R-project.org/package=xgboost