IMPROVING DUCKWORTH-LEWIS: STATISTICAL METHODS FOR REVISING SCORE TARGETS IN LIMITED OVERS CRICKET

MATTHEW KNOWLES

1. Introduction and Aims

In recent years, the use of data science in sport has become far more common place. In cricket in particular, questions are asked about metrics such as run-rates or bowling economy (average of how many runs conceeded for every ball bowled) in order to select a starting XI with the best lineup. Data can give a great insight into a team's performance, and if used correctly can aid a team greatly in their performance in competitions. The vast amount of data available is also helpful in finding patterns in matches. This is the area of focus for this project.

The specific aim of this project is to use more sophisticated methods for revising score targets in limited overs cricket. For the reader unfarmiliar with the game of cricket. Two teams have 50 'overs' each to score as many 'runs' as possible. However, sometimes, a rain delay, pitch invasion or other interruption leaves the team batting second with less time to complete their overs. For this reason, Duckworth and Lewis, two British statisticians wrote a paper outlining a method for how score targets are reset given the state of play when the interruption happened. (**?**)

We begin by looking at the maths behind the Duckworth-Lewis method, and the extension made by Stern (REF) in the early 2000s. We look at problems with this method, and how it is vulnerable to certain biases. Further, an investigation is undertaken into a few statistical methods aimed to reduce the exposure of the method to certain biases, the main method we look at here uses techniques from Bayesian Statistcs (REF).

Following on from this, we undertake some exploratory data analysis to look for immediate patterns in the match data we have, and to influence decisions that need to be made in building the models that are the central aspect of the project.

2. Data

2.1. **Data Origin.** The primary source of data for carrying out this project was downloaded from "cricsheet"[1] and stored locally on a private server. In total there are 2167 individual matches of data, each in a JSON format. These cover matches ranging from the 3^{rd} of January, 2004. Up to the 20^{th} of July, 2021.

2.2. **Attributes.** Each JSON file contains a considerable amount of metadata surrounding the match in question. Along with ball-by-ball data for the entire match. We have access to attributes such as the date, where the match was played, the entire teamsheet for both teams, who the officials were, who won- and by what margin, who won the toss; and many others.

We also have the ball-by-ball data. So for every ball bowled, it gives who were the striking and non-striking batsmen, how many runs were scored and how. It also details if a wicket was taken that ball, and how.

Date: Autumn Term 2021.

[1]https://cricsheet.org/

2.3. **Pre-Processing.** In order to get data in to a usable form, Python scripts were written to read the JSON files and extract features necessary for this project. These Python scripts output CSV files that can be fed into R for easier statistical analysis. For example, when training the neural network model, a Python script was built to extract the runs scored in each over for over 1400 games. The labelled CSV data was then used by an R script to output a matrix of runrates and corresponding final runs, which was used to train the Neural Network.

2.4. **Problems with the Data.** When it comes to machine learning, the more data the better is a general rule. Now this can sometimes lead to sub-problems, such as overfitting. But on the whole. it is much better to have as much data as possible. We will be training a neural network on 1435 data points. Stictly speaking, this isn't a lot of data, but there isn't much that can be done about this due to the cricketing calender only having a certain number of limited-overs matches each year.

3. Duckworth-Lewis

We begin by looking at the main equation that is the backbone of the Duckworth-Lewis method.

$$Z(u,w) = Z_0(w)[1 - exp(-b(w)u)] \tag{1}$$

In **??**, u represents the number of overs the runs are scored in, w is the number of wickets lost, and b(w) is an exponential decay constant. Due to commercial reasons, the authors were unble to publish some of the mathematical constants that they used. The fact this model follows an exponential decay pattern is the key thing. Infact, this is consistent with how runs per wicket evolves.

4. Improving DLS - Statistical Pattern Recognition

The main aim of this project, as previously mentioned, is to investigate ways of producing more realistic score targets for interruption. The main way we look to achieve this is through statistical pattern recognition methods. Below we discuss each of the methods that will be investigated in the main dissertation.

4.1. **Neural Networks.** At the time of writing, the Neural Network is still under construction.The idea behind using a neural network is that by using a backpropogation algorithm, the network can be trained to identify patterns in how the run-rate affects the final score. In order to achieve this, the network has been built from scratch to allow for as much control over it as possible, and fed runrate data from 1436 games of cricket.

We use a feed-forward multi-layer neural network. What this means is that we have a network with 50 "perceptrons" in the input layer, and then 3 hidden layers consiting of 25,10 and 3 perceptrons respectively. Finally, the output layer consists of a single perceptron giving a target score. Each node in one layer connects to each node in the proceeding layer. Each connection has a weight value and a bias value. Although randomly initialised, these wights and biases are updated using the backpropogation algorithm, to minimise an error function.

The only issue at this stage, given that we currently have no results to present in this report, is that 1436 datapoints isn't many in the grand scheme of things, so there is a possibility that this model will fall victim to overfitting- meaning that it will be really good at predicting datapoints from the set we train and test it on, but may not be as accurate when given future data.

In the dissertation itself, we use this as motivation for taking a look at some of the maths behind neural netwroks, rather than just mindlessley applying them using an R function. The "backpropogation" algorithm is derived step by step, supplememnted by Python code to show how the network is being built and trained from the ground up.

4.2. **Clustering.** The next method we plan to look at is that of clustering. We can cluster games based on certain metrics in order to prpedict how as a score will evolve.