STATS 50: Mathematics of Sport **Spring 2019**

Week 9 – Duckworth-Lewis-Stern and prediction in cricket

Lecturer: Damian Pavlyshyn

Warning: *these notes may contain factual errors*

1 The game of (one-day) cricket

Two teams play one innings each, trying to score as many runs as possible. An innings ends after all eleven players are out or 50 overs (sets of six deliveries) have been bowled. The first team to bat sets the target number of runs that the second team to bat must reach is win the game.

Now, even one-day cricket matches take a long time, and it often rains in England, so cricket matches are often interrupted before both teams have completed their innings. This creates a need to predict the number of runs a team would have scored if they were able to complete their innings.

2 The Duckworth-Lewis method

Frank Duckworth and Tony Lewis introduced a system based on "resources", which are the number of wickets and overs the batting team has left in its innings. The idea is that regardless of the total number of total overs in an innings, two teams that have the same number of these two resources are fairly matched, as they would be at the start of the game, before any runs have been scored.

We want to estimate the expected number $Z(u, w)$ of runs that a team with u overs and w wickets remaining will score. In fig. 2, we see why simply taking the average of all remaining runs among games with that particular configuration of resources is not a good idea. In particular, $Z(u, w)$ should be decreasing both u and w — otherwise a team might be incentivised to give away resources to *increase* their expected score!

Duckworth and Lewis's proposal was to model the expected number of runs remaining by the following formula, which obeys our monotonicity requirement.

$$Z(u, w) = a_w(1 - e^{-b_w u}), \tag{1}$$

where u and w are the remaining numbers of overs and wickets respectively, and a_w, b_w are model parameters.

Based on this model, tables of "remaining resource precentage" like the one in fig. 2 are released to the public. The entries of the table denote the percentage of resource remaining, given by the normalisation of Z:

$$p(u, w) = \frac{Z(u, w)}{Z(50, 0)} \cdot 100. \tag{2}$$

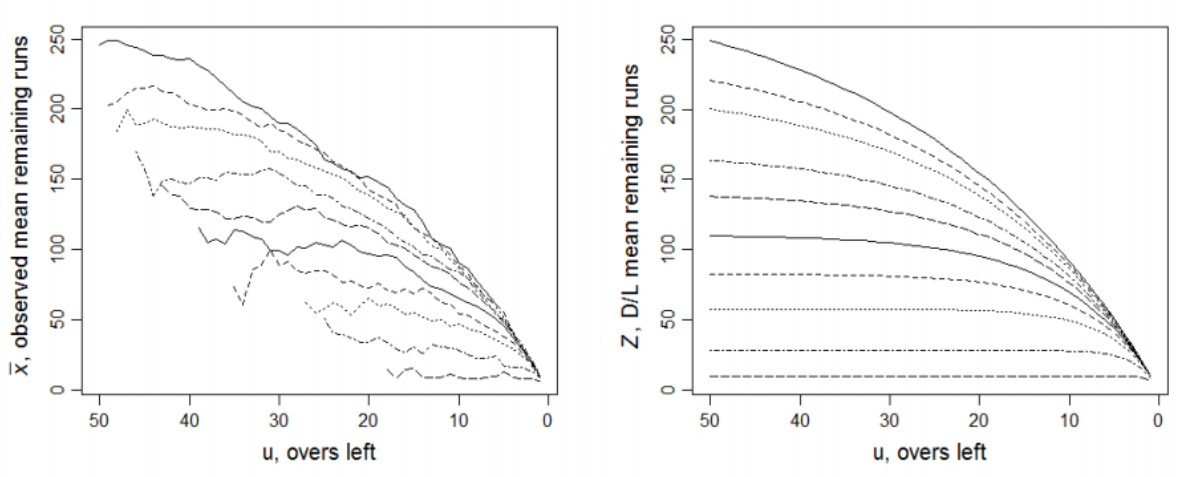

Figure 1: Mean runs scored with given resources and fitted model.

	Wickets Lost									
Overs Left	0	1	2	3	4	5	6	7	8	9
50	100.0	92.4	83.8	73.8	62.4	49.5	37.6	26.5	16.4	7.6
40	90.3	84.5	77.6	69.4	59.8	48.3	37.3	26.4	16.4	7.6
30	77.1	73.1	68.2	62.3	54.9	45.7	36.2	26.2	16.4	7.6
20	58.9	56.7	54.0	50.6	46.1	40.0	33.2	25.2	16.3	7.6
19	56.8	54.8	52.2	49.0	44.8	39.1	32.7	24.9	16.2	7.6
17	52.3	50.6	48.5	45.8	42.2	37.2	31.5	24.4	16.1	7.6
16	49.9	48.4	46.5	44.0	40.7	36.1	30.8	24.1	16.1	7.6
10	34.1	33.4	32.5	31.4	29.8	27.5	24.6	20.6	14.9	7.5
5	18.4	18.2	17.9	17.6	17.1	16.4	15.5	14.0	11.5	7.0
1	3.9	3.9	3.9	3.9	3.9	3.8	3.8	3.7	3.5	3.1

Figure 2: A typical D/L resource table.

2.1 Setting the target based on the resource table

We calculate R_1 and R_2, the available resources of the teams batting first and second respectively. The target for team 2 is then set as

$$T = \begin{cases} S\frac{R_2}{R_1} & \text{if } R_2 \leq R_1, \\ S + G_{50}\frac{R_2 - R_2}{100} & \text{if } R_2 > R_1. \end{cases} \tag{3}$$

Here, G_{50} is the expected number of runs that a team scores over a whole match. This is the most controversial part of the Duckworth-Lewis method.

2.2 Simple estimation of the coefficients

The Duckworth-Lewis method is proprietary, and the method of computing the coefficients is not publicly known, but we can present various reasonable methods of our own.

Figure 3: A scoreboard displaying a D/L par score

We have only specified a mean function, but a possible model might be something like

$$\begin{aligned} y_i &= Z(u_i, w_i) + \varepsilon_i \\ &= a_{w_i}(1 - e^{b_{w_i} u_i}) + \varepsilon_i, \end{aligned}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. This allows us to use a decreasing mean function $Z(u, v)$ while still explaining the erratic non-monotonicity of the observed data. Fitting this model amounts to performing our familiar least-squares optimization:

$$\underset{a_w, b_w : w=0,\ldots,9}{\text{minimize}} \sum_{i=1}^{n} (a_{w_i}(1 - e^{-b_{w_i} u_i}) - y_i)^2, \tag{4}$$

which is similar in principle to regular-old linear regression.

Since there are no terms in the sum that contain two different values of w, we can split the above optimisation into 10 different, easier problems. That is, for each $k = 0, \ldots, 9$, we compute

$$\underset{a_k, b_k}{\text{minimize}} \sum_{i : w_i = k} (a_k(1 - e^{-b_k u_i}) - y_i)^2. \tag{5}$$

2.3 Weighted least-squares

The above model does reasonably well on its own. However, like all models, it is imperfect and making it more complex might make better at prediction. Of course, this is not a given, as more complexity can also lead to overfitting and worse prediction, but as long as we keep this in mind, it's worth trying something.

One flaw with our current model is that the variances of runs scored is constant over all u and v. This is convenient for computation, but not at all realistic — usually, larger observations have more variance, so we would like to update our model:

$$y_i = Z(u_i, w_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2_{u_i,w_i}). \tag{6}$$

That is, the variance of each observation depends on specific number of resources. When this variance is small, we have good reason to trust the observations are close to the mean, whereas when the variance is large, we are much mroe uncertain. Thus, it makes sense (and is principled) to perform a *weighted* minimisation according to the variances:

$$\underset{a_w,b_w:w=0,\ldots 9}{\text{minimize}} \sum_{i=1}^{n} \frac{1}{\sigma^2_{u_i,w_i}} (a_{w_i}(1 - e^{-b_{w_i}u_i}) - y_i)^2. \tag{7}$$

Since we don't know the variances exactly, we must estimate them, so we might instead minimise

$$\underset{a_w,b_w:w=0,\ldots 9}{\text{minimize}} \sum_{i=1}^{n} \frac{1}{s^2_{u_i,w_i}} (a_{w_i}(1 - e^{-b_{w_i}u_i}) - y_i)^2. \tag{8}$$

This is better, but there are still problems

- What if $s^2_{u,w}$ is a poor estimate? This can easily happen when $n_{u,w}$ is small and can introduce a serious bias.
- What if $n_{u,w} = 1$? Then no variance estimate is possible at all.
- What if errors have some non-normal distribution?

There are many things we can do to address these problems, and all have their own advantages and drawbacks. For example:

- Split the data and estimate the variances and coefficients with separate halves. This reduces the bias, but also reduces the effective available data, increasing the model variane.
- Ignore (u, w) combinations with fewer than, say, 10 observations. This might substantially affect constant-w curves with few observations.
- We can estimate the error variance in non-parametric ways, for example the bootstrap...
- We can use a totally different loss function, perhaps penalising outliers less. For example

$$\underset{a_w,b_w:w=0,\ldots 9}{\text{minimize}} \sum_{i=1}^{n} \frac{1}{s^2_{u_i,w_i}} |a_{w_i}(1 - e^{-b_{w_i}u_i}) - y_i|. \tag{9}$$

To compare predictive models, we should split the data into a training and validation set, fitting each model on the training set, and then estimating how well the fitted models would do on new data by evaluating the RSS on the validation set. The model that minimises validation error is then a good candidate for the best model, and a good way of determining how much complexity it is appropriate to include.