Duckworth Lewis Method

Aditya Manjunatha

Btech - 22140

1 Code workflow

1.1 Data preprocessing :-

- Read the dataframe from the csv file
- Only look at the rows with Innings = 1
- Remove rows where Runs.Remaining is < 0
- Keep only the usefull columns such as 'Match', 'Date', 'Innings', 'Over', 'Runs', 'Total.Runs', 'Innings.Total', 'Runs.Remaining', 'Total.Out', 'Innings.Total.Out', 'Outs.Remaining', 'Wickets.in.Hand', 'Innings.Total.Runs', 'Total.Overs'
- Add the column called Overs.Remaining which is needed for the model. It is just 50 - df['Over']

1.2 Creating Dataframes :-

- I have created 11 dataframes df_0, df_1, ..., df_10. Each of these dataframes are those datapoints or matches where 'w' wickets are remaining.
- Again we keep only the important columns :- ['Match', 'Date', 'Innings.Total.Runs','Runs.Remaining', 'Overs.Remaining' ,'Total.Overs']
- Sort the Dataframes based on 'Overs.Remaining' just for visualization sake.

1.3 Adding extra rows to each Dataframe

TO get datapoints for when 50 overs are remaining and 'w' wickets in hand, we had to do this addition of rows. Till now we only had data points for when 49 overs are remaining as the scores were getting reported only after over 1.

1.4 Z_0(w, df):-

This returns the initial guesses for z_0 . Which is based on the average runs that are scored in all the overs when you have 'w' wickets left in hand.

1.5 L_0(w, df) :-

This piece of code is a mechanism to find the initial guesses for L_0. In the slides it was given that L is the change in runs scored when there is one ball left.(Atleast that is what I think it it is) Since there is no data for when one ball is left. I have taken it to be the average run rate for each 'w'.

1.6 Z_0(w)

This just returns the initial guess for Z_0 for a given w using function 1.4 as mentioned above

1.7 L_0(w)

This just returns the initial guess for L_0 for a given w using function 1.5 as mentioned above.

1.8 Z(u, w, L_0, Z_0)

This is the score function as described in the assignment description.

1.9 loss(z, y)

As described in the assignment it has been implemented.

1.10 total_loss(df, w, z_0. l_0)

This function essentially finds the loss for every datapoint in the given dataframe df_w and then sums up all the losses and then normalizes by dividing with len(df_w)

1.11 total_loss_wrapper(params, df, w)

This function is needed while using scipy minimize later

1.12 plot_graph_Q1()

This is the heart of the assignment.
We initialize 2 arrays for storing the optimized Z and L values for each wicket 'w' .
We loop over every data frame df_w to minimize the total_loss(df_w, w, z_0, l_0).
Where z_0, l_0 are got from Functions 1.6 and 1.7 mentioned above.
Scipy minimizes the loss function and then gives me the optimized z_0 and l_0, which is then appended into the arrays.
SO this happens for every dataframe (df_w) or 'w' .
Finally we create a plot using matplotlib.

1.13 plot_graph_Q2()

We do the exact same thing as in Q1. But instead now scipy only optimizes over Z. l_0 is fixed as the weighted average of the L's we got from the Q1.
Here also an array is kept to store the optimized z values for each 'w' or dataframe.
Finally we use matplotlib to plot the graph for the optimized values

2 Partial Results :-

The initial guesses for Z are :-

$$Z = [0, 9.122161315583398, 19.550153531218015, 35.13916061849164, 52.43465909090909, 71.72861621137483, 96.25633868341475, 123.93395577395577, 156.68090293453724, 188.82538369635145, 228.67472296522735]$$

The initial guesses for L are :-

$$L = [0, 1.400625225441866, 2.4472133247918, 3.6286496350364965, 4.426220738253796, 4.71985934031393, 5.181577991760684, 5.236914234455722, 5.314657535505251, 5.2133278525882565, 5.239997285618594]$$

So this encaptures the average runs scored for each wicket left.
And the average run rate for each wicket left.

The optimized Z_0 array for problem 1 is :-

$$[0.0, 6.534584509698644, 19.79130367823702, 40.38111727012784, 67.15301043601984, 92.30544946958776, 122.76577513959798, 156.31059293126035, 194.2125963169502, 218.33782477711665, 272.83597015762933]$$

The optimized L_0 array for problem 1 is :-

[0.0, 6.17436316410188, 7.66195420296417, 10.209584083292379, 10.155256724258942, 10.413609052662231, 11.147028134181776, 10.807427068375134, 10.830776111172819, 11.504750123685795, 10.6844738009]

The optimized Z_0 array for problem 2 without weighted average for l_0 is :-

[0.0030838144209895755, 6.328331506083453, 19.2565947928299, 41.49532947433889, 70.07517945940566, 98.40987746363838, 137.85403304037902, 175.1854553896984, 220.25192779535078, 254.33272941686172, 308.71868168473065]

The fixed L_0 value for problem 2 without the weighted average is :-

[9.053565678690466]

The optimized z_0 array for problem 2 with the weighted average is :-

[0.0030838144209895755, 6.571945033585825, 20.82564152583888, 47.54491580707263, 91.0896374832565, 143.79056834401877, 243.34369105545645, 358.456806376637, 530.5597406160891, 619.0553720696279, 885.5017808917651]

The fixed l_0 value for problem 2 with the weighted average is :-

[5.791905967124149]

Please check next page for the Plots

3 Plots :-

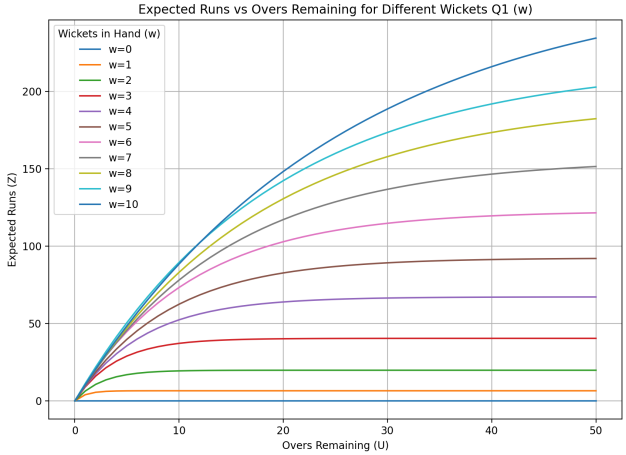

Figure 1: Plot for Q1

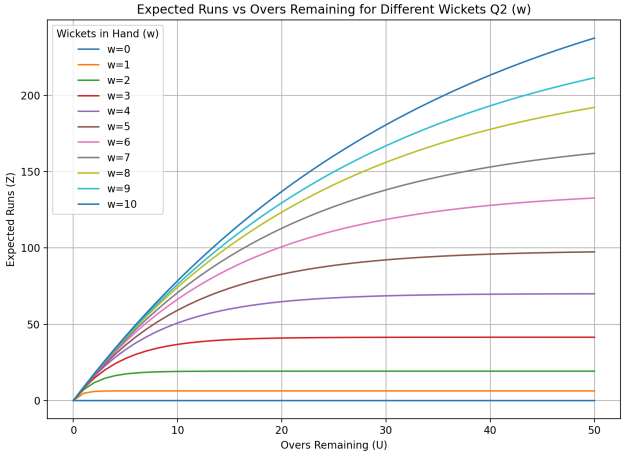

Figure 2: Plot for Q2 with normal averaging

It appears as if both the plots are the same . But upon carefull inspection, we can see that they differ for w = 5. It is easy to spot that because of the grid lines.
The slopes appear to be same even in Q1.
It is same in Q2 by construction.

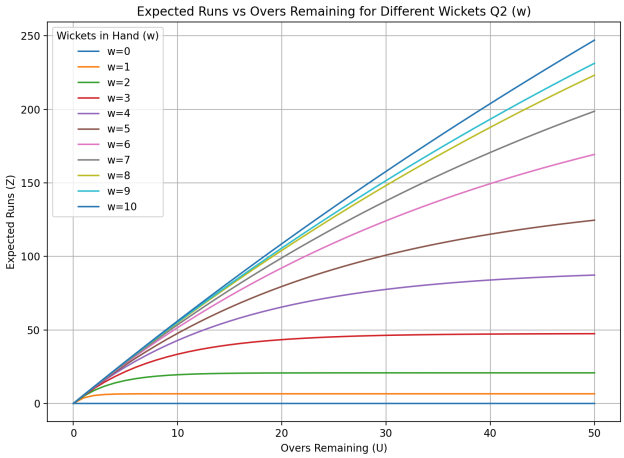

Figure 3: Plot for Q2 with weighted average

This is the plot I got when i did the weighted average for getting the L_0 value for Q2. The formula being.

```
L = np.array(L_opt_arr_2)
Wt = np.array([len(df_0)/len_df, len(df_1)/len_df, len(df_2)/len_df, len(df_3)/len_df, len(df_4)/len_df, len(df_5)/len_df, len(df_6)/len_df,
len(df_7)/len_df, len(df_8)/len_df, len(df_9)/len_df, len(df_10)/len_df])
L_avg = np.dot(L, Wt)  # Fixed l_0 for Q2
```

Figure 4: Weighted average code