Exploring Methods to Forecast T20 Cricket Game Scores

Anish Deshpande

2025-05-09

Abstract

This project focuses on various methods to forecast final scores in T20 games during the course of the game.

Introduction

T20 cricket is a game format that has risen in popularity in recent years. In limited overs cricket, there are two innings, each consisting of 20 overs (or 120 balls, since there are 6 balls per over). The game starts with a coin flip, and the winning team decides whether they would like to bat or bowl first. Team A is sent in to bat in the first inning, and they have the full 20 overs to score as many runs as possible, unless they lose 10 wickets before that period. Then, in the second inning, Team B has to surpass the score set by Team A in the first innings in order to win. The game ends once Team B surpasses the target score, plays all 20 of their overs, or loses all 10 of their wickets.

A well-known fact about T20 cricket is that the short duration of the game allows teams to bat quite aggressively and take more risks. However, it is usually seen that when teams lose quick wickets, they tend to play more cautiously in order to keep wickets in hand, and this usually lowers their scoring rate. These subtle patterns and tendencies between runs, wickets, and overs are what we hope to understand in order to predict and forecast scores in an ongoing inning.

Problem Statement

Assume we have ball-by-ball data for N number of innings across T balls and K metrics. The metrics of our interest are cumulative runs and cumulative wickets up to that point in the inning. We arrange this data in a 3-dimensional matrix (tensor) of dimensions $[N, T, K]$ for the rest of our analysis.

Let T_0 be the time of intervention. Then from time $T = 0$ until T_0 is considered pre-intervention, and time T_0 to T is considered post-intervention. We would like to predict the post-intervention data for a given inning n given its pre-intervention data, as well as the complete ball-by-ball data of all other $N - 1$ innings.

In the context of a cricket game, an example would be the following: Team A comes in to bat first and is currently 60/2 after 8 overs. We want to predict their score trajectory for the rest of the innings based on their pre-intervention trajectory, while also drawing and learning from the data of all other innings' trajectories. We call all these other innings the **donor units**, and the inning we are interested in the **treatment unit**.

Data Collection

All ball-by-ball data was collected through downloading JSON files for 18,500+ cricket matches over the past 20 years from cricsheet.org. Once the data was downloaded, it was parsed to collect relevant game and ball-by-ball information, and the data was then populated into a MySQL AWS relational database.

SQL was used to extract the appropriate $[N, T, K]$ matrix for all T20 games over the last 20 years. The dimensions of this tensor were $(22312, 120, 2)$. The data was stored efficiently in a `.npz` file for further analysis, which is essentially a compressed way to store numpy arrays.

Methods

Linear Regression

We start off by implementing linear regression as a baseline model, where we regress the runs scored in overs 2, 4, 6, 8, and 10 to predict the score at over 12. We then use the same predictors to predict the score at over 15. We evaluate the R^2 returned by these models and use them as a baseline to see how much better our other approaches perform.

The model is:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_5 x_5 + \epsilon$$

where x_i represents cumulative runs at a specific earlier over.

mRSC (Multi-dimensional Robust Synthetic Control)

Based on the model proposed by Vishal Misra et al., we attempt to replicate a robust synthetic control method using low-rank matrix completion for donor-based forecasting. We estimate a low-rank matrix from pre-intervention trajectories and extrapolate the missing post-intervention trajectory.

(Explain full SVD + reconstruction method)

ARIMA

We fit ARIMA(2,1,2) models to individual cumulative run time series up to the intervention point and forecast forward. The model captures autocorrelation and local trends in the run sequence.

K-Nearest Neighbors Simulation

For this simulation, we implemented a weighted distance metric prioritizing wickets ($\lambda = 20$) to match treatment innings with similar donor units. At each ball after T_0, we identify the top 50 nearest neighbors, sample their next-ball delta, and simulate forward until the 120th ball or the 10th wicket.

The distance function is:

$$D(i, j) = \sum_{t=1}^{T_0} (r_{it} - r_{jt})^2 + \lambda (w_{it} - w_{jt})^2$$

Mactivation (Bayesian Activation Model)

Using Professor Zes' `mactivate` library in R, we modeled the binary run activation probability per ball using splines and posterior inference. This allowed us to generate a posterior predictive distribution for scores after intervention by simulating from estimated activation curves.

Results

- Linear regression predicting 12th over score from overs 2, 4, 6, 8, 10: $R^2 = 0.93$
- Linear regression predicting 15th over score from same predictors: $R^2 = 0.80$
- mRSC produced reasonable counterfactuals in majority of innings
- ARIMA model performed well in smooth innings, but struggled with irregularity (high RMSE outliers)

(Include plots of forecast vs actual, error distributions, etc.)

Conclusion

(Write later)

References

- Misra, Vishal. "Multi-dimensional Robust Synthetic Control."
- Kalman Filters and State Space Models.
- Bayesian DLS methods using resource tables.
- Cricsheet.org: Open-source ball-by-ball cricket data.
- Zes, B. "mactivate" R package documentation.