BATTING STRATEGY BUILDING IN LIMITED OVER CRICKET MATCHES

ABSTRACT

Cricket has been amongst the most popular sports globally, though not as popular as football, has its enigma. It has various aspects where statistical models can improve its understanding and thus improve the game altogether. Data generated through Cricket is enormous and can be used for the betterment of a player or a team. Even if a Cricket team squad consists of a limited number of players, their combination makes out a complicated problem with many possible line-ups. The problem of selecting the best eleven players, with readily available data, has attracted many sports analysts to work. Our objective is to provide a model that can suggest a line-up to the team's head coach based on the actual line-ups that had played in the past. Different batting orders affect the overall run scored in first innings, and a good batting order will ultimately increasing the chance of winning as runs scored will be higher. This research proposes a new multi-headed Convolutional Neural Network (CNN) machine learning model with six heads, each involving deep learning to analyze the player's batting capabilities. This model is tested against various baseline models. A player can be represented through a feature vector, reflecting a player's abilities, skills, impact on the team's winning, etc. This research can have extended applications in various leagues worldwide, like IPL, Big Bash, and various domestic tournaments. Dataset is collected from espncricinfo.com.

***K*eywords** Multi-headed · Machine Learning (ML) · Convolutional Neural Network (CNN) · Masking · Line-up · Cricket

1 Introduction

1.1 About the game

Cricket is a bat and ball game played between two teams, 11 players each, on a field which has a rectangular 22-yard-long pitch in the centre. The game is played by 120 million players worldwide, making it the second most popular sport in the world after football, in which a player can be a batsman who has good experience of batting strategies, can be a bowler having good experience of bowling strategies, and all-rounder with experience of both batting as well as bowling strategies. The aim of the game is to outscore your opponents. The decision to bat first or not is given to the team winning a coin toss. A team batting first tries to score as much as possible by analyzing the conditions of the game, and then the team batting in the second innings attempts to outscore the opponent's target.

Generally, a total of 15 Players are selected for a series or league forms the squad, and from these, the playing eleven is selected, which forms the team for a particular match. A team, in general, comprises six specialist batsmen[1], one all-rounder, and four bowlers. The order (commonly referred to as batting line-up) in which these batsmen are sent in for batting plays a significant role in the final team score. For that purpose, it is very important to have a batting line-up that has all kinds of weapons in its armory like hitting power, innings building, strike sustainability, etc. Since there are so many young talented individuals competing for each batting position, it becomes very hard for the selection committee to reach on a common decision and get the right combination [2]. Researchers over the years have tried various statistical methods to predict the batting order. Most of these researches had been done in times when the use of deep neural networks was not as prevalent as today. We are taking a step forward in the direction of using modern-day deep net approaches on the batting line-up prediction.

1.2 Brief idea about Neural Networks

Neural nets are a means of doing machine learning, in which a computer learns to perform some tasks by analyzing training examples. Usually, the examples have been hand-labeled in advance. An object recognition system, for instance, might be fed thousands of labeled images of cars, houses, coffee cups, and so on, and it would find visual patterns in the images that consistently correlate with particular labels.[3]

Neural Network is made up of artificial neurons or nodes; it contains various layers which has different functions. All neuron of one layer is interconnected to all its previous and foremost layer neurons. Inputs are processed through successive layers of neurons. There are multiple variants of the neural network, of which vanilla and convolution have been used in this paper.

Convolutional Neural Network(CNN) has been proved to have shift-variant property, making it quite intuitive to use it in our task. CNN are neural networks that use at least one convolution layer in addition to Dense layers, which extracts only spatial features that are crucial for the image classification task. There is a kernel that is traversed on whole data to capture important features. There can be a lot of important features in a dataset, so to capture the maximum number of these features, there can be many convolution filters on a single convolutional layer. Multi-headed models are models that have a backbone and multiple heads. As all positions share some similarities between each, we have tried a multi-headed convolutional neural network model with CNN as the backbone and Dense layers on each head.

2 Related Works

There is a general tradition in Cricket to have a player having higher batting skills play more balls than the player who has comparatively less batting skills. The number of balls a player will play depends highly on the position he bats on, i.e., a player batting at position six will generally have lower numbers of balls to face, so we would want such a player to have an excellent striking rate. The number of criteria that can be taken to predict a batting line-up can be huge, and deciding which criteria will serve the purpose is hard. Earlier people have relied on approaches like *Integer Programming*[4], *dynamic programming*[5], and the *Markov model with simulated annealing*[6] for lineup prediction. There can be 11! line-ups for a single match, and as the number of matches grows, it becomes very hard to apply the proposed solution to each line-up permutation. *Ovens and Bukiet*[7] considered an approach using a Markov chain for a subset of 163,324 orders, where only three or four batsmen in the line-up change positions. *Swartz et al.*[6] also studied simulated annealing to reduce the number of batting strategies choices; however, strong assumptions limit this approach's applicability[8]. The advent of Neural nets has enabled us to try and fit complex curves on large amounts of features. *Condon, Golden and Wasil*[9] and *Flitman*[10] has shown that neural networks can be used with good effect in sports analytics. We have tried to use deep learning to reduce the number of sufficient permutations that we need, to simulate results for all 11!

3 Dataset

The raw data has been extracted from ESPNCricinfo. We have scrapped the batting scorecard of the top 6 batsmen for each match in ODI format from 2010-2019. The raw data contains runs per match, balls faced per match and the playing position of the player in that match, and the total balls played by every player playing from 2010 to 2020. All the data, along with the date of the match, were extracted.

Dataset	Train	Test
Original	199	50
Augmented	3984	996

Table 1: Dataset Statistics

The raw data needs a little bit of cleaning as it might happen that for a match, we may not have any data for the lower order positions *i.e.* some player DNB (Player did not bat) or TDNB (Team did not bat). We handled missing/anomaly data by inserting details of player's latest performance. Due to the small size of the dataset, we augmented the dataset by permuting the rows of the original dataset. Each match can be represented by 6! ways, we have added 20 random permutations for each match to scale up the dataset. Each example in final data after preprocessing has six rows and 13

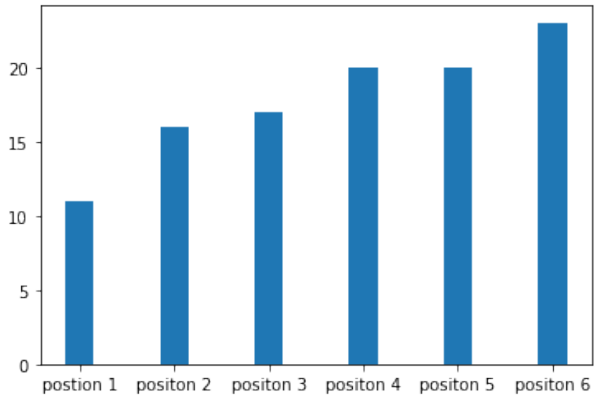

Figure 1: Number of players played at each postion

columns, specifying six players having 13 features each. Overall players from 2010 to 2019 are 37, resulting in 37 classes for each position.

4 Features

Each batsman is represented uniquely by a feature vector. The neural net is trained for each batting position in a separate branch of the same model, so it is crucial to discuss what these features are. Since the most important criteria for selecting a player at a position are his batting capabilities and performance at that position, irrespective of the team he played against. Since a player's performance will keep varying as he plays more and more matches, we will have to maintain his performance vector accordingly, so to account for that, we have taken all the features dynamic on the time axis.

4.1 Dynamic Average

In Cricket, whether a batsman is good or bad relies heavily upon the average he has throughout his career. To take account of the crest and trough throughout his career, we have taken average runs of the batsman till i^{th} match.

$$Avg_i = \frac{R_i}{i}$$

where,
Avg_i is the average of a player till i^{th} match,
R_i is the total runs scored by the player till i^{th} match and
i denotes the number of matches in which he was dismissed.

4.2 Dynamic Strike-Rate

Apart from average, the rate at which a batsman scores runs has a direct impact on the final score of the team. Batsmen which/who bat lower down the order have higher strike-rate as compared to the top-order batsmen. A dependable and reliable batsman generally has a good average and decent strike-rate.

$$SR_i = \frac{R_i}{B_i} \times 100$$

where,
SR_i is the strike rate of a player till i^{th} match,
R_i is the total runs scored by the player till i^{th} match and
B_i is the total number of balls, the player has faced *till* i^{th} *match.*

4.3 Dynamic Reliability

We have taken the reliability of a batsman as an average number of balls he faces per match. More reliability means that the batsman holds the crease for a longer duration, which directly reflects his contribution to the team's final score. The higher the reliability, the higher will be the contribution to the total score.

$$Rel_i = \frac{B_i}{i}$$

where,
Rel_i is the current reliability of a player at i^{th} match,
B_i is the total number of balls, player has faced *till* i^{th} *match.*
i denotes the match number $i.e.$ i^{th} match, where $i \epsilon \{1, 2, 3...\}$.

4.4 Dynamic Experience

Although reliability is enough to decide the effectiveness of a player, there may be a case where Rel_i of a less experienced player may become greater than Rel_i of a more experienced player. To handle this problem, we have taken account of the total number of balls faced till the i^{th} match. A new player playing can have a higher Rel_i, but then his dynamic experience will have a very low value compared to experienced players.

4.5 Dynamic probability for each position

The position at which a batsman generally bats plays a huge factor in his selection. A batsman prefers a batting position according to his temperament, game-play, and skills.

4.6 Features denoting Time

As the nature of the data has a resemblance to the time-series data, the matches cannot be treated as independent to each other; the feature vectors of players in the current match are dependent on the previous matches and also adds a dependency to the future matches. The feature requirements of a player at a certain position can vary with time, e.g., batsmen at position sixth nowadays are expected to have a good strike rate (above 120) as compared to matches played in 2012, when strike rates such as 90 or 100 were considered to be very good. To remove this time dependency from our dataset, we have added day, month, and year as features to introduce a time perspective to the dataset so that the model can be trained to handle test data that contains matches from the various instances of time.

5 Model

We are providing the feature vector of six batsmen, and our task is to predict the coach-fitted batting order of six players selected as batsmen for the playing 11 for a given match. We are using a mask layer to truncate the probability of remaining players from the set of 37 players, hence selecting only six of them.

Masking vector for each example is defined as:

$$M_{\text{A}} : X \to \{0, 1\}$$
$$M_A[x] := \begin{cases} 1, & if\ x \in A \\ 0, & if\ x \notin A \end{cases}$$

where,
A is player set,which consist id of players, for a given match, and
x is player id of players where $x \in$ {0,1,2,3,..,36}

5.1 Logistic Regression:

The task directly relates to a classification problem of predicting a player from a pool of players for each position for each match, where the classes correspond to the players. Our training algorithm uses the one-vs-rest (OvR) scheme for classification into 37 players. This model takes a 1-dimensional feature vector for each match, combining all six-player feature vectors as an input and outputs each players' probability score for each position.

5.2 Single-headed:

Deep learning models are known to perform well when the task is to fit on complex data. This model comprises of six different models for each of the top-six batting positions. Each position has its own configured deep learning model. We have also tried the convolutional neural network(CNN), which is known for its shift-invariant property. As our players can be in any order in input data, it would be intuitional to use it. We have tried Single-headed Neural Network(NN) model,which is 6 independent Neural Network(NN) models for each position. Similarly, Single-headed Convolutional Neural Network(CNN) model is 6 independent Convolutional Neural Network(CNN) models for each position.

CNN are neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Then, the Softmax layer is used for the probability distribution, followed by the final masking layer. It is a smooth approximation of one-hot argmax, mathematically defined by:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{x} = (x_1, \ldots, x_K) \in \mathbb{R}^K$$

$where,$
K is the number of total players, and
x_i is the input value of each classes(players) to be distributed, where $i \in$ {0,1,2...36}

5.3 Multi-headed:

Inter-dependency of positions directs us towards using a joint feature extractor for all heads(positions). We have tried the Convolutional layer for feature extraction as a backbone and Dense layers for each head. The depiction of multi-headed convolutional neural network(CNN) is presented at Figure 2.

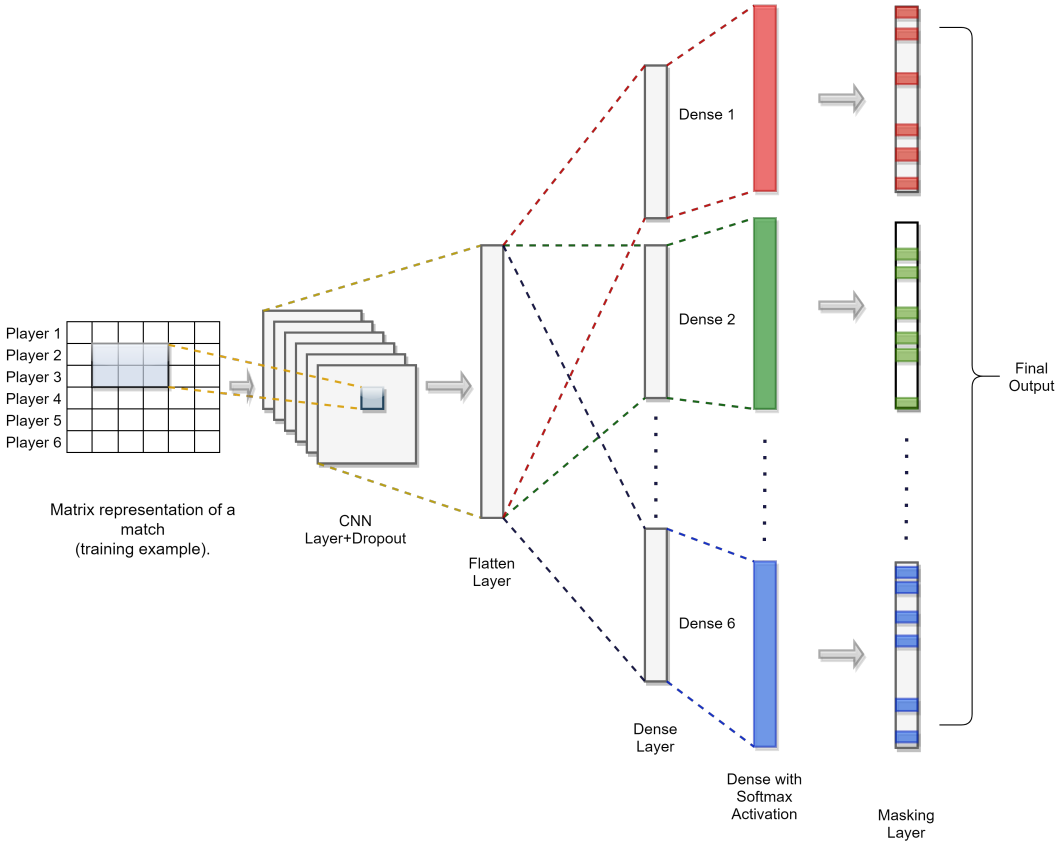

Figure 2: Multi-headed CNN with Masking

5.4 Prediction

At the end of each model, we have used a custom function for predicting unique line-ups. Algorithms is described below:

1. Output probabilities of each position are concatenated in a 1-D vector.
2. For each value in this vector, position number and player id are appended.
3. This vector is then sorted in descending order based on probabilities.
4. When a player with the highest probability and is selected, the position id and player id are marked as done. Players and positions which are marked as done are then skipped.
5. We repeat the process for the next highest probability until all positions are marked.

6 Reproducibility and Implementation

The whole dataset has been split into 80% train and 20% test. Again train is split into 8:2, 20% for validation. We have used three strategies, namely:

OO: Original train and Original test.
OA: Original train and Augmented test.
AA: Augmented train and Augmented test.
AO: Augmented train and Original test.

Before feeding the data into models, we have applied Standard Scaling to whole data using Sci-kit learn.

6.1 Logistic Regression:

Scikit-learn Library is used to train the logistic regression model. We have trained the model on 200, 200, 1000 and 1000 iterations corresponding to OO, OA, AA and AO respectively.

6.2 Keras-based:

We have used TensorFlow as backend here. We have used the functional API of Keras for the complex topology of our model. As the model uses masking and multi-headed, Keras is a known platform for designing complex architectures at ease. The input example is a 2-dimensional matrix, comprising of features vectors of 6 players, where the total features used in our case are 13. ReLU activation is used for non-linearity. Kernel size of (6,1) is kept the same as wherever CNN is used. Using a different set of hyperparameters, we achieved the individual best for each model. Optimization is done using Stochastic Gradient Descent. Below are configurations:

Dataset / Models	OO			AA		
	Epochs	Nodes	Filters	Epochs	Nodes	Filters
Single-Headed NN	80	8	–	100	16	–
Single-Headed CNN	80	8	6	50	16	16
Multi-Headed CNN	80	8	6	30	16	16

Table 2: Configurations of each model for reproducibility

We have trained all the models with a learning rate of 0.01 with a batch size of 32. A dropout of 0.25 is used for regularisation after each CNN layer.

6.3 Metric

Now that we have a unique line-up, we will test it against the actual line-up proposed by the coach for that particular match. So for each position (head), we have chosen the weighted average of f1-score of all labels as our metric.

Mathematically,

$$\textit{weighted average} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

$where$,
n is the total number of players who have played on that position,
w_i is support (number of times in ground data) of labels (player IDs),
x_i is F1-score of that label (player ID).

7 Results and Discussion

As mentioned earlier in the Model Section, our task is a multiclass classification task; hence we have used one-vs-rest (OvR) logistic regressor. Results on various combination of the dataset are given in Table 3

Logistic Regression												
OO			OA			AA			AO			
Position	Train	Test	Position	Train	Test	Position	Train	Test	Position	Train	Test	
1	99	99	1	99	56	1	89	88	1	89	96	
2	99	97	2	99	58	2	91	89	2	91	97	
3	99	95	3	99	61	3	92	90	3	92	88	
4	99	85	4	99	36	4	76	74	4	76	62	
5	99	80	5	99	40	5	72	65	5	72	63	
6	99	91	6	99	46	6	77	72	6	77	74	

Table 3: Results for Logistic Regressor

It can be seen that logistic regression works exceptionally well on almost every position giving high train and test accuracy on used metric for the original dataset(OO), but when we tested the same model on the augmented dataset (OA), the accuracies dropped to 40-50 %. The model is then trained on the augmented dataset; we found that the training accuracy was very low, and the test was even lower (on both AA and AO). We concluded that the logistic regressor is not complex enough to fit the augmented dataset. Also, it can be deduced that AO and AA's results are almost similar and AO $\subseteq$ AA, hence calculating results on AA alone suffices.

To introduce the desired complexity, we have used deep neural network models as they are known to fit on complex datasets. Two structures of Models are used, one with a CNN layer and one without it. When both the models were trained and tested on the original dataset(OO), both CNN and NN models overfitted. It can be concluded that the data is significantly less for NN and CNN models. Both models were then trained and tested on the augmented dataset(AA). It can be seen in Table 4 that both models produced the highest accuracy on the used metric for all the positions on the AA dataset. Although both models' results were similar, the CNN layered model is expected to perform better on a larger dataset containing more permutations.

Although the single-headed model obtained desired results, it does not deal with the batting positions' inter-dependency. Moreover, the requirement of using a separate CNN layer for each position is replaced by just one parent CNN layer, which serves as a backbone layer for the Multi-headed CNN. It can be observed from Table 5 that with a loss of very little accuracy (1-2)%, the model's complexity can be reduced significantly, and at the same time, the model can incorporate the inter-dependency between positions.

	OO			AA		
NN	Position	Train	Test	Position	Train	Test
	1	97	98	1	99	97
	2	97	95	2	99	98
	3	98	91	3	99	98
	4	96	70	4	99	93
	5	97	67	5	98	93
	6	94	75	6	98	94
CNN	Position	Train	Test	Position	Train	Test
	1	96	95	1	98	97
	2	97	96	2	98	97
	3	97	95	3	98	98
	4	95	78	4	95	92
	5	95	70	5	95	93
	6	94	85	6	95	93

Table 4: Results for Single-Headed models

Multi-Headed CNN					
OO			AA		
Position	Train	Test	Position	Train	Test
1	97	99	1	98	97
2	96	98	2	99	99
3	98	92	3	97	97
4	97	75	4	92	91
5	96	69	5	94	93
6	93	82	6	92	91

Table 5: Results for Multi-Headed CNN Model

8 Conclusion and Future Work

We have observed that if the size of the dataset is small, the logistic regression can give high accuracy results. When the complexity of the dataset is increased as required in the AA dataset case, a deep learning based architecture can be used with great effect. Among various models that we have tried, the Multi-Headed CNN model gives the best results alongside the Single-Headed CNN model but with relatively less complexity. Although we expect the Multi-Headed CNN model to outperform Single-Headed CNN, it cannot be said with certainty.

This research can be effectively used for other international teams as well as major leagues around the world like IPL, BigBash, etc. by changing the dataset used. As the model is made using Keras, it serves as a placeholder and can be reused again and again for different datasets. It should be noted that our dataset is a time-series dataset, and

memory-based models like GRU, LSTM can be applied. CRF can also be applied as it performs better as there is interdependence on all positions.

References

[1] Md Jakir Hossain, Md Abul Kashem, Md Saiful Islam, E Marium, et al. Bangladesh cricket squad prediction using statistical data and genetic algorithm. In *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*, pages 178–181. IEEE, 2018.

[2] Subramanian Rama Iyer and Ramesh Sharda. Prediction of athletes performance using neural networks: An application in cricket team selection. *Expert Systems with Applications*, 36(3):5510–5522, 2009.

[3] Explained: Neural networks. `https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414`. Accessed: 2020-09-30.

[4] Hannah Gerber and Gary D Sharp. Selecting a limited overs cricket squad using an integer programming model. *South African Journal for Research in Sport, Physical Education and Recreation*, 28(2):81–90, 2006.

[5] Stephen R Clarke. Dynamic programming in one-day cricket-optimal scoring rates. *Journal of the Operational Research Society*, 39(4):331–337, 1988.

[6] Tim B Swartz, Paramjit S Gill, David Beaudoin, and Basil M desilva. Optimal batting orders in one-day cricket. *Computers & operations research*, 33(7):1939–1950, 2006.

[7] Bruce Bukiet and Matthews Ovens. A mathematical modelling approach to one-day cricket batting orders. *Journal of sports science & medicine*, 5(4):495, 2006.

[8] Masoumeh Izadi and Simranjeet Narula. Batting order setup in one day international cricket. In *AAAI*, 2018.

[9] Edward M Condon, Bruce L Golden, and Edward A Wasil. Predicting the success of nations at the summer olympics using neural networks. *Computers & Operations Research*, 26(13):1243–1265, 1999.

[10] Andrew M Flitman. Towards probabilistic footy tipping: A hybrid approach utilising genetically defined neural networks and linear programming. *Computers & operations research*, 33(7):2003–2022, 2006.