DE GRUYTER
OPEN

JAISCR, 2018, Vol. 8, No. 3, pp. 159 – 172
10.1515/jaiscr-2018-0011

ONE-MATCH-AHEAD FORECASTING IN TWO-TEAM SPORTS WITH STACKED BAYESIAN REGRESSIONS

Max W. Y. Lam

*Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong,
Shatin, Hong Kong*

Submitted: 10th February 2017; Accepted: 10th April 2017

Abstract

There is a growing interest in applying machine learning algorithms to real-world examples by explicitly deriving models based on probabilistic reasoning. Sports analytics, being favoured mostly by the statistics community and less discussed in the machine learning community, becomes our focus in this paper. Specifically, we model two-team sports for the sake of one-match-ahead forecasting. We present a pioneering modeling approach based on stacked Bayesian regressions, in a way that winning probability can be calculated analytically. Benefiting from regression flexibility and high standard of performance, Sparse Spectrum Gaussian Process Regression (SSGPR) – an improved algorithm for the standard Gaussian Process Regression (GPR), was used to solve Bayesian regression tasks, resulting in a novel predictive model called TLGProb. For evaluation, TLGProb was applied to a popular sports event – National Basketball Association (NBA). Finally, 85.28% of the matches in NBA 2014/2015 regular season were correctly predicted by TLGProb, surpassing the existing predictive models for NBA.

Keywords: Sports analytics, one-match-ahead forecasting, winning probability, Gaussian process regression

1 Introduction

Sports analytics has received much attention in recent years. Its success greatly contributes to the business of professional sports, which nowadays is a multi-billion dollar industry. Well-developed analytical techniques can, to a large extent, benefit many parties including the sports team managers, the coaches, and even the athletes themselves. Examples [1] include opponent analysis, targeted coaching, player acquisition and tailored personal training.

In this paper, we deal with a prominent prediction task – one-match-ahead forecasting, which is defined in a probabilistic manner. Essentially, a predictive sports model featured with winning probability calculation is demanding. This is particularly conspicuous in the business of sports wagering, which is another multi-billion dollar industry on its own. To determine the winning odds of opposing sports teams, nowadays wagering markets mostly adopt the parimutuel schemes [2], in which the winning odds are mainly driven by the public preferences to the sports teams, and occasionally fluctuate on real-time matches. It has been shown [3] that the public preferences tend to result in overestimation of winning probabilities as the general public are likely to have heavy favorites over their beloved teams.

In fact, a number of forecasting tasks in sports analytics appears to be the natural applications of machine learning algorithms, yet only a limited literature dealing with these tasks can be found [4]. For the existing approaches, the formulated prediction tasks consider arguably shallow information as some in-depth factors including the individual performance of each player and the contribution of different player positions were not taken into account. This triggers our initiative that explicit modeling of the individual ability of each player and its contribution to the match outcome will improve the existing approaches in terms of predictive accuracy.

Specifically, a pioneering modeling approach based on stacked Bayesian regressions is proposed. Benefiting from regression flexibility and high standard of performance, Sparse Spectrum Gaussian Process Regression (SSGPR) [5], being an improved algorithm for the standard Gaussian Process Regression (GPR) [6], was used to solve Bayesian regression tasks. Noted that, Bayesian regression infers the predictive distributions from inferences rather than predicting point estimates by optimizing pre-defined rules. This makes it well-suited for analytical calculation of winning probability. Our modeling approach coupled with stacked SSGPR results in a novel predictive model called TLGProb, standing for *Two-Layer Gaussian Process Regression Model for Winning Probability Calculation*.

To our understanding, this paper is the first work that explicitly models sports by associating the players' abilities with teams' strengths using stacked regressions. As the first step, we restrict our analysis on two-team sports in order to make sense of the resultant modeling approach. Noted that two-team sports still covers most of the popular sports nowadays such as football, basketball, and baseball. Generally modeling sports with more than two teams could be more complex, but is still possible and is left to the future work.

National Basketball Association (NBA), being a two-team sports as well as one of the most popular sports events around the world, was used to evaluate our proposed modeling approach. After training TLGProb with a regular reason of NBA, we tested its performance on the subsequent season. Finally, the trained TLGProb correctly predicted the winning teams of 1,049 matches out of 1,230 matches in NBA 2014/2015 season, attaining 85.28% accuracy.

In the following Sections, we first introduce Bayesian regression in Section 2. This Section ends with a discussion on the elegant properties of the GPR framework and several merits of SSGPR. In Section 3, we define our own notations and formulate the task of one-match-ahead forecasting in two-team sports from a probabilistic perspective. Our modeling approach is subsequently described in Section 4, followed by the results of comparative experiments in Section 5. Finally, we conclude our work and identify some potential future works in Section 6.

2 Bayesian Regression

To enable readers understanding the remainder of this paper, we first review Bayesian regression in this Section. Regression is a task of predicting an output $y^\star$ given the associated D-dimensional input $\mathbf{x}^\star$, after training on the data set of N input-output pairs $\mathcal{D}_{\text{train}} \equiv \{(\mathbf{x}_i, y_i)\}_{i=1}^N$. To appreciate Bayesian inference, the output y_i is commonly assumed to be generated by a latent function $f(\mathbf{x}_i)$ corrupted by a Gaussian noise of constant variance

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,\ \varepsilon_i \sim \mathcal{N}\left(0, \sigma^2\right). \tag{1}$$

With this setup, Bayesian regression simply boils down to a Bayesian inference of the latent function $f(\cdot)$, as described in the next Section.

2.1 Bayesian Linear Regression (BLR)

The starting point is given by the standard Bayesian Linear Regression (BLR), where we assume that the latent function $f(\cdot)$ is a simple linear function

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}. \tag{2}$$

Given the vector of outputs $\mathbf{y} = [y_1, ..., y_N]^\top$ and the $N \times D$ input matrix $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^\top$, we get the likelihood distribution:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}\left(\mathbf{w}^\top \mathbf{x}, \sigma^2 \mathbf{I}\right). \tag{3}$$

To derive the posterior, we have to define a prior over the weights $\mathbf{w}$. For mathematical elegance, a zero-mean multivariate Gaussian prior is commonly used

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \Sigma). \tag{4}$$

Then, we can apply Bayes' rule to derive the posterior, which is

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \mathcal{N}\left(\mathbf{H}^{-1}\mathbf{X}^\top \mathbf{y}, \sigma^2 \mathbf{H}^{-1}\right), \tag{5}$$

where

$$\mathbf{H} = \mathbf{X}^\top \mathbf{X} + \sigma^2 \Sigma^{-1}. \tag{6}$$

By integrating over all possible $\mathbf{w}$ with respect to (5),

$$p(\mathbf{y}^\star|\mathbf{X},\mathbf{y},\mathbf{X}^\star) = \int p(\mathbf{y}^\star|\mathbf{w},\mathbf{X},\mathbf{y},\mathbf{X}^\star)\, p(\mathbf{w}|\mathbf{X},\mathbf{y})\, d\mathbf{w}, \tag{7}$$

we obtain the predictive distribution for new inputs $\mathbf{X}^\star = [\mathbf{x}_1^\star, ..., \mathbf{x}_T^\star]^\top$,

$$\begin{aligned} p(\mathbf{y}^\star|\mathbf{X},\mathbf{y},\mathbf{X}^\star) = \mathcal{N}\Big(&\mathbf{X}^{\star\top}\mathbf{H}^{-1}\mathbf{X}^\top\mathbf{y}, \\ &\sigma^2\left(1 + \mathbf{X}^{\star\top}\mathbf{H}^{-1}\mathbf{X}^\star\right)\Big). \end{aligned} \tag{8}$$

To enable non-linear fitting in the Bayesian framework, BLR can be generalized by mapping each input $\mathbf{x}$ to a non-linear feature space with a non-linear function $\phi_\theta(\mathbf{x})$ parametrized by the hyperparameters θ

$$f(\mathbf{x}) = \mathbf{w}^\top \phi_\theta(\mathbf{x}). \tag{9}$$

As in (8), we can obtain a similar predictive distribution by replacing $\mathbf{x}$ with $\phi_\theta(\mathbf{x})$

$$\begin{aligned} p(\mathbf{y}^\star|\mathbf{X},\mathbf{y},\mathbf{X}^\star;\theta) = \mathcal{N}\Big(&\Phi_\theta^{\star\top}\mathbf{A}^{-1}\Phi_\theta^\top\mathbf{y}, \\ &\sigma^2\left(1 + \Phi_\theta^{\star\top}\mathbf{A}^{-1}\Phi_\theta^\star\right)\Big), \end{aligned} \tag{10}$$

where

$$\Phi_\theta = [\phi_\theta(\mathbf{x}_1), ..., \phi_\theta(\mathbf{x}_N)]^\top, \tag{11}$$

$$\Phi_\theta^\star = [\phi_\theta(\mathbf{x}_1^\star), ..., \phi_\theta(\mathbf{x}_T^\star)]^\top, \tag{12}$$

and

$$\mathbf{A} = \Phi_\theta^\top \Phi_\theta + \sigma^2 \Sigma^{-1}. \tag{13}$$

2.2 Gaussian Process Regression (GPR)

An alternative to specifying the feature map $\phi_\theta(\mathbf{x})$ is to use the *kernel tricks*. We define a positive definite function

$$k_\theta(\mathbf{x},\mathbf{x}') = \left(\Sigma^{1/2}\phi_\theta(\mathbf{x})\right)^\top \left(\Sigma^{1/2}\phi_\theta(\mathbf{x}')\right), \tag{14}$$

which is called the *kernel function* or the *covariance function* as it holds a unique statistical meaning in Gaussian process [7]

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k_\theta(\mathbf{x}, \mathbf{x}')\right), \tag{15}$$

which means that

$$k_\theta(\mathbf{x},\mathbf{x}') = \mathbb{E}\left[f(\mathbf{x})f(\mathbf{x}')\right]. \tag{16}$$

To convey with words, the kernel function basically measures the correspondence of every pair of function outputs using the input pairs. Treating a generalized BLR from a Gaussian process perspective is known as Gaussian Process Regression (GPR) [6].

Due to the marginalization property of Gaussian process, while we have a finite set of inputs, the vector of the corresponding function outputs

$$\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_N)]^\top, \tag{17}$$

is simply a multivariate Gaussian

$$p(\mathbf{f};\theta) = \mathcal{N}(\mathbf{m}, \mathbf{K}_\theta), \tag{18}$$

characterized by the mean vector

$$\mathbf{m} = [m(\mathbf{x}_1), m(\mathbf{x}_2), ..., m(\mathbf{x}_N)]^\top, \tag{19}$$

and the kernel matrix

$$\mathbf{K}_\theta = \begin{pmatrix} k_\theta(\mathbf{x}_1,\mathbf{x}_1) & \dots & k_\theta(\mathbf{x}_1,\mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k_\theta(\mathbf{x}_N,\mathbf{x}_1) & \dots & k_\theta(\mathbf{x}_N,\mathbf{x}_N) \end{pmatrix}. \tag{20}$$

To infer the function outputs $f(\mathbf{x}_i^\star)$ of unseen test inputs $\mathbf{x}_1^\star, ..., \mathbf{x}_T^\star$, we can derive the predictive distribution through Bayes' rule by taking (18) as prior and (1) as likelihood. In fact, the resulted predictive distribution is identical to substituting (14) into (10) and setting $\Sigma = \mathbf{I}$, yielding

$$\begin{aligned} &p(\mathbf{y}^\star|\mathbf{X},\mathbf{y},\mathbf{X}^\star;\theta) = \mathcal{N}(\hat{\mathbf{m}}, \hat{\mathbf{K}}_\theta) \\ &\hat{\mathbf{m}} = \mathbf{m}^\star + \mathbf{K}_\theta^\star\left(\mathbf{K}_\theta + \sigma^2\mathbf{I}\right)^{-1}(\mathbf{y} - \mathbf{m}) \\ &\hat{\mathbf{K}}_\theta = \mathbf{K}_\theta^{\star\star} - \mathbf{K}_\theta^\star\left(\mathbf{K}_\theta + \sigma^2\mathbf{I}\right)^{-1}\mathbf{K}_\theta^{\star\top}, \end{aligned} \tag{21}$$

where

$$\mathbf{m}^\star = [m(\mathbf{x}_1^\star), m(\mathbf{x}_2^\star), ..., m(\mathbf{x}_T^\star)]^\top, \tag{22}$$

$$\mathbf{K}_\theta^\star = \begin{pmatrix} k_\theta(\mathbf{x}_1^\star,\mathbf{x}_1) & \dots & k_\theta(\mathbf{x}_1^\star,\mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k_\theta(\mathbf{x}_T^\star,\mathbf{x}_1) & \dots & k_\theta(\mathbf{x}_T^\star,\mathbf{x}_N) \end{pmatrix}, \tag{23}$$

and

$$\mathbf{K}_\theta^{\star\star} = \begin{pmatrix} k_\theta(\mathbf{x}_1^\star,\mathbf{x}_1^\star) & \dots & k_\theta(\mathbf{x}_1^\star,\mathbf{x}_T^\star) \\ \vdots & \ddots & \vdots \\ k_\theta(\mathbf{x}_T^\star,\mathbf{x}_1^\star) & \dots & k_\theta(\mathbf{x}_T^\star,\mathbf{x}_T^\star) \end{pmatrix}. \tag{24}$$

In practice, the mean function $m(\mathbf{x})$ is commonly set to be zero. However, $k_\theta(\mathbf{x},\mathbf{x}')$ should be carefully chosen since the performance of GPR is greatly dependent on it. We use the kernel function to encode curvature assumptions of the latent function $f(\mathbf{x})$, such as noisiness, smoothness and periodicity. Different choices of kernel function result in different regression models [8], including splines, semi-parametric regression, multiple kernel learning, and Bayesian neural networks with infinitely many hidden units [9]. In the light of this flexibility, we may select a proper kernel function with regards to the predictive performances of different kernels in validation dataset. It is advised to refer to [8] for more details of automatic kernel function selection.

The superiority of GPR over other kernel machines lies in its ability to derive a theoretically grounded objective function for hyperparameters θ, i.e., the analytically tractable marginal likelihood

$$\theta^* = \operatorname{argmax}_\theta p(\mathbf{y}|\mathbf{X};\theta) \equiv \operatorname{argmin}_\theta - \log p(\mathbf{y}|\mathbf{X};\theta) \tag{25}$$

where

$$-\log p(\mathbf{y}|\mathbf{X};\theta) \propto (\mathbf{y}-\mathbf{m})^\top \left(\mathbf{K}_\theta + \sigma^2\mathbf{I}\right)^{-1}(\mathbf{y}-\mathbf{m}) + \log\left|\mathbf{K}_\theta + \sigma^2\mathbf{I}\right|. \tag{26}$$

However, a computational bottleneck exists in training GPR as the inversion of $\mathbf{K}_\theta + \sigma^2\mathbf{I}$ in general requires $O(N^3)$ operations and $O(N^2)$ storage. In our regression problem, where N can scale up to 10^4, the standard algorithm for GPR is obviously too slow and not feasible, thus an improved algorithm for GPR is required.

2.3 Sparse Spectrum Gaussian Process Regression (SSGPR)

Towards computational efficiency, an improved algorithm called Sparse Spectrum Gaussian Process Regression (SSGPR) [5] was proposed. Despite speeding up GPR, SSGPR does not sacrifice the predictive performance of standard GPR, on the contrary, SSGPR improves over GPR [5, 10]. What's more, with some tricks in SSGPR, instead of specifying the kernel function, it is possible to analytically learn an optimal kernel function for GPR. Though, in exchange, we need to make one assumption – the kernel function is stationary

$$k_\theta\left(\mathbf{x},\mathbf{x}'\right) = k_\theta\left(\mathbf{x}-\mathbf{x}'\right). \tag{27}$$

Then, according to Wiener-Khintchine theorem [11, 12], the power spectral density and the autocorrelation function of a wide sense stationary stochastic process constitute a Fourier pair. In the case of stationary Gaussian process, the autocorrelation function is simply the kernel function, thus we have

$$k_\theta\left(\mathbf{x}-\mathbf{x}'\right) = \int_{\mathbb{R}^D} e^{2\pi i\omega^\top(\mathbf{x}-\mathbf{x}')} S(\omega)d\omega. \tag{28}$$

On top of that, by Bochner's theorem [13], we know that any stationary kernel function $k_\theta(\mathbf{u})$ can be represented as the Fourier transform of a positive finite Borel measure. In other words, $S(\omega)$ is proportional to certain probability measure $p(\omega)$, and can be written as

$$S(\omega) = \eta p(\omega). \tag{29}$$

By substituting (29) into (28), we now can express any stationary kernel function $k_\theta(\mathbf{u})$ as the expectation of an inner product of two complex exponentials with respect to the probability measure $p(\omega)$

$$k_\theta(\mathbf{x},\mathbf{x}') = \eta\mathbb{E}_p\left[e^{2\pi i\omega^\top(\mathbf{x}-\mathbf{x}')}\right]. \tag{30}$$

Noted that the real part of this inner product has an exploitable structure – a sum of inner product

$$\begin{aligned}\mathrm{Re}\left\{e^{2\pi i\omega^\top(\mathbf{x}-\mathbf{x}')}\right\} &= \cos\left(2\pi\omega^\top\left(\mathbf{x}-\mathbf{x}'\right)\right)\\ &= \cos\left(2\pi\omega^\top\mathbf{x}\right)\cos\left(2\pi\omega^\top\mathbf{x}'\right)\\ &\quad + \sin\left(2\pi\omega^\top\mathbf{x}\right)\sin\left(2\pi\omega^\top\mathbf{x}'\right).\end{aligned} \tag{31}$$

To exploit this structure, the trick is to define a particular feature map $\phi_\theta(\cdot)$

$$\phi_\theta(\mathbf{x}) = \sqrt{\frac{\eta}{M}}\left[\phi_{\sin}(\mathbf{x}) \oplus \phi_{\cos}(\mathbf{x})\right]^\top, \tag{32}$$

where $\oplus$ is the operator of vector concatenation,

$$\phi_{\sin}(\mathbf{x}) = \left[\sin\left(2\pi\tilde{\omega}_1^\top\mathbf{x}\right), ..., \sin\left(2\pi\tilde{\omega}_M^\top\mathbf{x}\right)\right], \tag{33}$$

$$\phi_{\cos}(\mathbf{x}) = \left[\cos\left(2\pi\tilde{\omega}_1^\top\mathbf{x}\right), ..., \cos\left(2\pi\tilde{\omega}_M^\top\mathbf{x}\right)\right], \tag{34}$$

and $\tilde{\omega}_1, ..., \tilde{\omega}_M$ are sampled from $p_\theta(\omega)$. What is tricky is that this feature function is in fact a Monte Carlo approximation of the real part of (30) while setting $\Sigma = \mathbf{I}$ in (14). When viewing this as an approximation of kernel function, we need to derive $p(\omega)$ first in order to sample ω. That is, we still need to choose a kernel function in advance and obtain $p(\omega)$ in closed form using Fourier transform.

To escape from selecting a kernel function in advance, a tricky approach is to make use of the objective function derived in (25). In particular, we include the scaling constant η and the spectral frequencies in the vector of hyperparameters

$$\theta = [\sigma, \eta, \omega_1, ..., \omega_M]. \qquad (35)$$

Then, we jointly optimize the marginal likelihood with respect to the hyperparameters. Though learning the spectral frequencies by optimization departs from Monte Carlo approximation of the stationary kernel function, it in some sense allows us to learn an optimal stationary kernel function that fits our problem at hand. In other words, we learn an optimal kernel function instead of approximating a predefined kernel function.

It has been shown that, optimizing the spectral frequencies rather than sampling the spectral frequencies improves a lot in terms of predictive accuracy due to the additional flexibility [5, 10]. What's more, in SSGPR, the computational cost of training is reduced to $O(NM^2)$, while the memory cost is reduced to $O(NM)$. Here, we manually set M according to our computational resources and the scalability of the problem. Different from other speed-up algorithms for GPR [14] that use a set of sparse inputs to represent the whole set of inputs, we pose sparsity assumption on the spectrum of stationary kernel function in a way that all the data points are used to train the regression. In practice, SSGPR with a sufficiently small $M << N$ is able to achieve predictive performance comparable to a standard GPR with universal kernel (e.g. RBF kernel).

3 Problem Formulation

In this paper, our goal is to analytically derive the winning probability of two teams competing on the next match while only the historical data before the match are used. For easy reference, two competing teams is commonly denoted by the *home team* and the *visiting team*. In other words, we wish to answer how likely is the home team or the visiting team to win the next game. We call this task *one-match-ahead forecasting*. The governing assumption is that, players' abilities on consecutive matches change *smoothly*, even though their performances on consecutive matches vary *occasionally*. With this assumption, the past performances of the players in two competing teams are sufficient to infer the result of next match.

Formally, the result of any match is regarded as an ordered pair, $\left(G_k^H, G_k^V\right)$, which denotes the game points attained by the home team and the visiting team respectively on the kth match. For the next match where the match result is unknown, it is natural for us to use random variables, i.e., $\left(\mathbf{g}_{k+1}^H, \mathbf{g}_{k+1}^V\right)$.

To calculate the winning probabilities of the next match, we only need to concern the signed deviation of the game points attained by two competing teams, that is, our target is only a single random variable, $\mathbf{g}_{k+1} = \mathbf{g}_{k+1}^H - \mathbf{g}_{k+1}^V$, since the winning probabilities of the home team and the visiting team can be directly determined by $Pr(\mathbf{g}_{k+1} > 0)$ and $Pr(\mathbf{g}_{k+1} < 0)$.

By above assumption, the distribution of $\mathbf{g}_{k+1}$ is very much dependent on the past performances of both the home team and the visiting team. In other words, given that the players' performances and the results of first k matches are known, essentially an exploitable estimator for $\mathbf{g}_{k+1}$ should be computed. We define this abstractly as a regression task

$$\mathbf{g}_{k+1} = f(\mathbf{p}_{1:k}) + \varepsilon_{k+1}, \ \varepsilon_{k+1} \sim \mathcal{N}(0, \sigma^2), \qquad (36)$$

where $\mathbf{p}_{1:k}$ is the performances of team's players in first k matches. Now, our goal is to find a "model" $f(\cdot)$ so that it best explain the association between the result of next match and past performances of the players in two competing teams.

4 Our Modeling Approach

4.1 Notation and Terminology

For the ease of explanation, we use NBA as an example of two-team sports throughout this Section. For our concern, historical data of NBA is divided into multiple seasons. Here, we define a season as a sequence of matches

$$\mathcal{M}_y = \left(\mathbf{M}_1^y, \mathbf{M}_2^y, ..., \mathbf{M}_k^y, ...\right),$$

where $\mathcal{M}_y$ denotes the season that starts in y and $\mathbf{M}_k^y$ denotes the kth scheduled match of the season. To refer to a specific team on the match $\mathbf{M}_k^y$, we denote the home team and the visiting team by $\mathbf{T}_k^H$ and $\mathbf{T}_k^V$ respectively. The set of all teams in the season

$\mathcal{M}_y$ is defined as $\mathcal{T}_y$. For simplicity, in the remainder of this Section we use $\mathbf{T}_k$ to denote $\mathbf{T}_k^H$ and $\mathbf{T}_k^V$ which implies that the equation is applicable to both teams.

Table 1. Features representing each player's performance or ability

Feature	Description
FG	Field goals per minute
FGA	Field goals attempts per minute
FG%	Field goals percentage
3P	3-point field goals per minute
3PA	3-point field goals attempts per minute
3P%	3-point field goals percentage
FT	Free throws per minute
FTA	Free throw attempts per minute
FT%	Free throws percentage
ORB	Offensive rebounds per minute
DRB	Defensive rebounds per minute
TRB	Total rebounds per minute
AST	Assists per minute
STL	Steals per minute
BLK	Blocks per minute
TOV	Turnovers per minute
PF	Personal fouls per minute

In our notion, a team $\mathbf{T}_k$ on the kth match is modeled as a set of n players $\{\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_n\}$, in which the order of players does not matter. Usually, for team sports, players are assigned to different positions. For example, in basketball, there are 3 main positions – *Center*, *Forward* and *Guard*. Each position corresponds to a subset of players. It is notable that some players may take 2 positions at the same time, so these subsets are not necessarily disjoint. To encode the information of player positions, we define $\mathcal{G}_k^{(p)}$, a set of players on the kth match that are assigned to the pth position, such that

$$\mathbf{T}_k = \bigcup_{p=1}^{P} \mathcal{G}_k^{(p)}, \tag{37}$$

where P is the number of positions in our concerning sports. To enable a mapping from a player to his position, we define a function that returns the position index

$$p = I_k(\mathbf{P}_i). \tag{38}$$

4.2 Modeling Player's Ability

For any player $\mathbf{P}_i \in \mathbf{T}_k$, we describe his performance on the kth match by a D-dimensional vector $\mathbf{p}_k^i$. We call this the *player's performance*. For example, in NBA, we use 17 attributes, as shown in Table 1, to describe players' performances. In general, we define a match-dependent function ρ

$$\mathbf{p}_k^i = \rho(\mathbf{P}_i, \mathbf{T}_k), \tag{39}$$

which gives the performance vector of the player $\mathbf{P}_i$ in the team $\mathbf{T}_k$.

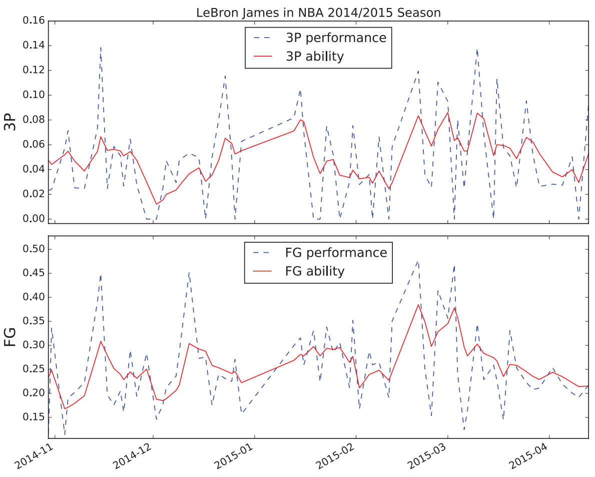

Figure 1. Time series plot of 3-point field goals (3P) and field goals (FG) of LeBron James in NBA 2014/2015 season.

Now, we can express all players' past performances in vector form. This seems fascinating because we may apply techniques in time series forecasting to predict players' performances on the next match. It is essential that the values of players' performances usually fluctuate a lot from matches to matches. One example is illustrated on Figure 1, where LeBron James' performances on 3-point field goals and field goals are investigated. It is suspected that directly using players' performance for regression task defined in (36) is not likely to be meaningful. Therefore, we introduce a concept of *player's ability*. We argue that a player's performance and his true ability are different quantities. Players sometimes make good use of their talents but sometimes not. Essentially, a player's performance is not only affected by the player's ability but also affected by other factors, such as opponents' abilities and team strategies. For practical concern, we consider all the other factors as noise, and propose that a player's performance is an unbiased estimator of his ability. It is reasonable in a sense that a player's performance should be consistent with his ability at that time.

While treating the player's ability as another quantity, we denote it by $\mathbf{a}_k^i$, which is again a D-dimensional vector so that the consistency with player's performance is preserved. Since players are trained from time to time and gaining experiences across the season of matches, players' abilities should change with time. In this regard, we define another match-dependent function α

$$\mathbf{a}_k^i = \alpha(\mathbf{P}_i, \mathbf{T}_k), \tag{40}$$

which gives the ability vector of the player $\mathbf{P}_i$ in the team $\mathbf{T}_k$ before the $(k+1)$th match. While a player's performance is treated as an estimator of his ability, we write

$$\mathbf{a}_k^i = \mathbb{E}\left[\mathbf{p}_k^i\right]. \tag{41}$$

4.3 Inferring Player's Ability

Under this estimation model, it is now possible to infer player's ability by applying time-series techniques like smoothing [15]. In our approach, we employed a simple but effective method – exponential smoothing [16]. We made a small change to the standard exponential smoothing procedure to enhance the adaptiveness of the model for different kinds of sports. To formulate our smoothing method, we define a count-up index set of matches participated by the player $\mathbf{P}_i$ before the kth match

$$Q_k^i = \left\{q \in \mathbb{Z}_+ : \mathbf{P}_i \in \left(\mathbf{T}_k \cap \mathbf{T}_{k-q}\right)\right\}. \tag{42}$$

Then, for any player $\mathbf{P}_i$, the inferred ability is determined by

$$\mathbf{a}_k^i = \begin{cases} \mathbf{a}_0^i & \text{if } Q_k^i = \emptyset; \\ (1-c^\lambda)\mathbf{p}_k^i + c^\lambda \mathbf{a}_{k-r}^i & \text{else,} \end{cases} \tag{43}$$

where $c \in [0,1)$ is a constant that controls the weight of time dependency on the past performances, $r = \min Q_k^i$ is the number of matches that have been passed since last match participated by $\mathbf{P}_i$, λ is a scaling factor that is proportional to the time difference between the $(k-r)$th match and the kth match, and $\mathbf{a}_0^i$ is the initialization function of player's ability while no past performances is recorded in our data

$$\mathbf{a}_0^i = \frac{1}{\left|\mathcal{G}_k^{(p)}\right|} \sum_{\mathbf{P}_j \in \mathcal{G}_k^{(p)}} \rho\left(\mathbf{P}_j, \mathbf{T}_k\right), \tag{44}$$

in which p is the position index as defined in (38). Note that, the only difference between our method and the standard exponential smoothing is the presence of λ. When λ is large, $\alpha(\mathbf{P}_i, \mathbf{T}_k)$ will have less dependence on previous ability $\alpha(\mathbf{P}_i, \mathbf{T}_{k-r})$, but having more dependence on current performance $\rho(\mathbf{P}_i, \mathbf{T}_k)$. This is reasonable because more recent performances are more relevant to current ability.

For NBA games, the time difference of the same player joining any two consecutive matches is usually within 1-6 days. Therefore we simply set $\lambda = d/3$, where d is the time difference in days. After setting λ, we can obtain c by optimization

$$c^* = \operatorname{argmin}_{c\in[0,1)} \|\alpha(\mathbf{P}_i, \mathbf{T}_{k-r}) - \rho(\mathbf{P}_i, \mathbf{T}_k)\|_2^2. \tag{45}$$

Two examples of inferring player's ability from player's performance are shown on Figure 1. In these two plots, it is observed that the red solid line has significant clearer trend than the blue dotted line. For instance, we can witness an improvement of 3-point field goals from early 2014-12 to late 2015-01 from the red solid line, whereas, it is difficult to be witnessed from the blue dotted line due to several sudden drops in that period.

4.4 Inferring Team's Strength

To avoid the abuse of terminology, we refer the *team's strength* to the actual ability of the team as opposed to the team's performance on the court. For any team $\mathbf{T}_k$, the team's strength is denoted by $s(\mathbf{T}_k)$. With the inferred abilities of the players in the team, we now come up with an estimation of team's strength to enable quantitative comparison of two opposing teams. It is sensible that a team's strength is dependent on its players' abilities. In common treatment of feature extraction, this can be achieved by directly concatenating the ability vectors of the players in the same team

$$s(\mathbf{T}_k) = \bigoplus_{\mathbf{P}_i \in \mathbf{T}_k} \alpha(\mathbf{P}_i, \mathbf{T}_k). \tag{46}$$

However, in practical sense, direct concatenation of ability vectors is often problematic for the training of regression model since the vector of team's strength would then be sizable in terms of its dimensionality. Take basketball game as an example, as shown in Table 1, each player's ability is represented by a 17-dimensional vector. Normally having 13 players in a team including the reserves, vector concatenation in (46) will produce a 221-dimensional vector. In this case, our regression model simply needs much more training data to ensure the prediction stability and preventing from over-fitting. Yet, being limited by the number of sports events, the number of training samples is usually not enough to meet this requirement.

A possible solution here is to employ some well-known dimensionality reduction methods, such as principal component analysis (PCA) and linear discriminant analysis (LDA). Nonetheless, these methods do not encode the domain knowledge from the sport itself. Alternatively, we design a dimensionality reduction method that is domain-specific for sports games. The main idea is to train, for each player position, an embedding function that maps player's ability vector to a point estimator that represents the player's contribution to his team. In fact, there are a number of such estimators proposed in the research community of sports analytics [17]. Examples include the plus-minus score, the player efficiency rating, and the individual offensive and defensive ratings. The estimator we used is called Adjusted Plus-Minus (APM) score [18]. One main advantage of using APM score is that each player's APM score is independent of the abilities of that player's opponents and teammates, by contrast, the traditional plus-minus score is highly opponent-dependent. In this sense, APM score is a good measure of a player's individual contribution. Formally, we denote the APM score of player $\mathbf{P}_i$ by $z(\mathbf{P}_i, \mathbf{T}_k)$. Using this point estimate, we can define the embedding function g_p for the pth position

$$z(\mathbf{P}_i, \mathbf{T}_k) = g_p(\alpha(\mathbf{P}_i, \mathbf{T}_k)). \tag{47}$$

Here, it is vital that we have different embedding functions for different player positions because players in different positions usually have distinctly distributed ability vectors even though they contribute equally in terms of APM score. For example, in basketball, Guard is likely to get higher values on 3P, while Forward is likely to get higher values on FG.

To find a suitable embedding function, we treat it as a regression problem: for all $\mathbf{T}_k \in \mathcal{T}_y, \mathbf{P}_i \in \mathcal{G}_k^{(p)}$, our inferred function $\hat{g}_p(\cdot)$ is defined by

$$\begin{aligned} z(\mathbf{P}_i, \mathbf{T}_k) =& \hat{g}_p(\alpha(\mathbf{P}_i, \mathbf{T}_k)) + \\ &+ \varepsilon_{p,k},\ \varepsilon_{p,k} \sim \mathcal{N}(0, \sigma_p^2), \end{aligned} \tag{48}$$

which is regarded as the first Bayesian regression task in our modeling approach. In our proposed model – TLGProb, we adopt the GPR framework

$$\hat{g}_p(\mathbf{a}_k^i) \sim \mathcal{GP}\left(m(\mathbf{a}_k^i), k(\mathbf{a}_k^i, \mathbf{a}_k^j)\right)\ s.t.\ I_k(\mathbf{P}_i) = I_k(\mathbf{P}_j), \tag{49}$$

where the mean function $m(\cdot)$ and the kernel function $k(\cdot,\cdot)$ can be flexibly determined with respect the error measure in SSGPR.

Using the convention of $z(\mathbf{P}_i, \mathbf{T}_k)$, we now determine team's strength by

$$s(\mathbf{T}_k) = \bigoplus_{\mathbf{P}_i \in \mathbf{T}_k} z(\mathbf{P}_i, \mathbf{T}_k). \tag{50}$$

This function returns an n-dimensional vector, where n is the number of players in the team. Now, the size of the vector is acceptable since n is usually small in team sports. In addition, similar to the definition of player's performance, we define a variable to address the actual performance of the team. Naturally, it would be the *game points* scored by the team. Similar to the assumptions that are made for the players, the game point attained by the team is dependent on team's strength.

As mentioned in Section 3, we only concern about the game points differential $\mathbf{g}_{k+1}$, which should be the realization of two teams' strengths. The second regression task is thus formed

$$\mathbf{g}_{k+1} = \hat{f}(\mathbf{s}_k) + \varepsilon_k,\ \varepsilon_k \sim \mathcal{N}\left(0, \sigma^2\right), \tag{51}$$

where

$$\mathbf{s}_k = s(\mathbf{T}_k^H) \oplus s(\mathbf{T}_k^V). \tag{52}$$

In TLGProb, we again handle it with SSGPR:

$$\hat{f}(\mathbf{s}_k) \sim \mathcal{GP}(0, k(\mathbf{s}_k, \mathbf{s}_{k'})), \tag{53}$$

where $k(\cdot,\cdot)$ can be determined empirically. This regression is trained subsequently after the the first regression model are well-trained in terms of mean squared error measure, therefore our approach is interpreted as stacked Bayesian regressions. Figure 2 shows a block diagram of our modeling approach.

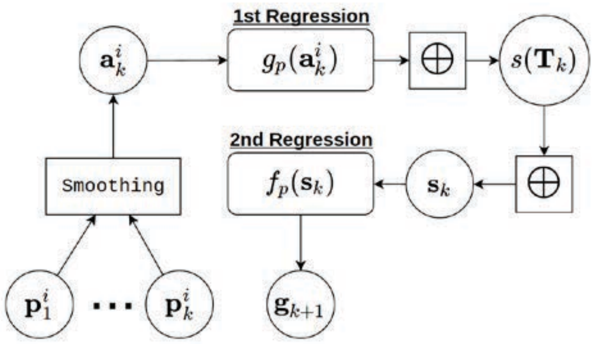

Figure 2. Block diagram of our modeling approach with stacked Bayesian regressions

Table 2. Accuracies attained by baseline predictive models

Baseline Predictive Model	Accuracy
Random Guessing	50%
Counting Victories of Last 5 Games	64.64%
Counting Victories of Last 10 Games	65.67%
Counting Victories of Last 20 Games	64.21%
Counting Victories of Last 50 Games	67.09%

5 Experiments

Our modeling approach coupled with stacked SSGPR results in a novel predictive model called TLGProb. In this Section, we analyze the results of applying TLGProb to NBA 2014/2015 season, and compare its predictive performance to other similar works. The implementation of TLGProb that we present in this Section is available on Github.,[1] which is built upon our own Python implementation of SSGPR available on Github.[2]

5.1 NBA Dataset

NBA historical data was retrieved from Basketball-Reference,[3] which is a site that stores NBA matches data starting from 1948. It provides box score statistics, which can be converted to the player performance statistics that we used in Table 1. For data collection, we implemented our own robot for continuous update of matches data using Selenium,[4] – a Python library that enables browser automation.

5.2 Training Mechanism

Before applying TLGProb to NBA 2014/2015 season, it is first trained with matches data in NBA 2013/2014 regular season. The training process contributes to the two regression task in our model. While using SSGPR to solve for these regression tasks in TLGProb, we can take advantage of its modeling flexibility by random initialization of hyperparameters θ. Though there exists a number of local minimas in optimization, in practice we can still guarantee exploitable performance for each random initialization by using stochastic optimization with appropriate stopping criteria. In particular, we used one of the most popular stochastic optimization algorithm – Adam [19]. Predictive performance of SSGPR were computed using 5-fold cross validation. After training 50 randomly initialized SSGPR instances, the one that gives the least mean squared error was selected for our evaluation.

5.3 Evaluation Metric

Our model was evaluated over 1,230 matches in NBA 2014/2015 season. Since we only have two teams competing on each match, winning team pre-

[1]https://github.com/MaxInGaussian/TLGProb
[2]https://github.com/MaxInGaussian/SSGP
[3]http://www.basketball-reference.com
[4]http://www.seleniumhq.org

Table 3. Performance of different regression models

1st Regression \ 2nd Regression	KRR	DT	AdaDT	GBM	RF	SSGPR
KRR	85.04%	84.03%	84.80%	84.96%	83.82%	84.31%
DT	82.11%	81.68%	84.72%	83.66%	84.55%	84.39%
AdaDT	83.50%	82.00%	84.55%	82.76%	82.28%	82.76%
GBM	82.65%	82.65%	84.39%	84.15%	83.74%	85.04%
RF	85.20%	82.74%	84.39%	84.72%	84.96%	84.15%
SSGPR	**85.28%**	81.51%	84.96%	84.72%	84.55%	**85.28%**

	KRR	DT	AdaDT	GBM	RF	SSGPR
$\mathcal{L}_{\text{NLPD}}$	4.2973	4.3367	4.3125	4.3016	4.3061	**3.2980**

diction is nothing more than a binary classification problem. To evaluate classification performance, we naturally use the accuracy measure, which is calculated given the acceptance threshold of winning probability τ

$$\text{accuracy} = \frac{\sum_{k=1}^{|\mathcal{M}_{2014}|} \mathbb{1}(\mathbf{A}_k)}{|\mathbb{A}|}, \tag{54}$$

where

$$\begin{aligned}\mathbf{A}_k = &\left(\left(G_k^H > G_k^V\right) \wedge \left(Pr\left(\mathbf{g}_k^H > \mathbf{g}_k^V\right) > \tau\right)\right) \\ &\vee \left(\left(G_k^H < G_k^V\right) \wedge \left(Pr\left(\mathbf{g}_k^H < \mathbf{g}_k^V\right) > \tau\right)\right),\end{aligned} \tag{55}$$

$$\begin{aligned}\mathbb{A} = \{k \in \{1,...,|\mathcal{M}_{2014}|\} : &(Pr(\mathbf{g}_k^H > \mathbf{g}_k^V) > \tau) \\ &\vee (Pr(\mathbf{g}_k^H < \mathbf{g}_k^V) > \tau)\},\end{aligned} \tag{56}$$

and $\mathbb{1}(\mathbf{A})$ is a logical indicator function that returns 1 if and only if $\mathbf{A}$ is true otherwise 0 is returned.

5.4 Baseline Predictive Models

Without considering much detail of the game, randomly guessing theoretically could achieve a baseline of 50% accuracy in the long run. Though, we conceived that rational people normally do not guess randomly. Therefore, a new set of baseline models was examined.

To simulate how people choose the game winner, a more reasonable predictor is to look back previous matches and predict the team with more victories to be the winner. In total, we tested 4 predictors corresponding to looking back 5, 10, 20 and 50 games. The result is shown on table 2. Predictors based on the number of victories apparently worked better than random guessing, while the best achieved a modest accuracy up to 67%.

5.5 Comparing to Other Regression Models

In this Section we show that SSGPR is typically superior to existing state-of-the-art regression models, while proposing it to the regression tasks presented in our modeling approach. Towards this argument, a comparative experiment was conducted. In additional to our implemented SSGPR, we tried 5 famous non-Bayesian regression models – Kernal Ridge Regression (KRR) with RBF kernel [20, 21] , Decision Tree (DT) [22], Ada Boosted Decision Tree (AdaDT) [23], Gradient Boosting Machine (GBM) [24] and Random Forests (RF) [25]. We used Scikit-learn [26] to implement these models, in which the model parameters are selected by 5-fold cross validation with grid search algorithm. The accuracies attained by different combination of regression models are shown in Table 3. It reveals that using SSGPR for the first regression task, and then feeding its outputs to either KRR or SSGPR will give slightly better accuracy. Yet, to maintain a Bayesian environment for analytical calculation of winning probability, in TLGProb, we still stick to SSGPR in the second regression task.

To justify this claim, we want to show that Bayesian regression model is more suitable for winning probability calculation. By our definition, winning probability is computed directly from the predictive distribution obtained from the regression model. Here, we measure how well the predictive distribution is predicted with Average Negative Log Predictive Density Loss (NLPD) [27, 28],

$$\mathcal{L}_{\text{NLPD}} = -\frac{1}{2M}\sum_{i=1}^{M}\left(\log \text{Var}\left(y_i^\star\right) + \frac{(y_i^\star - \mu_i^\star)^2}{\text{Var}\left(y_i^\star\right)}\right), \tag{57}$$

Table 4. Accuracies attained by different predictive models for NBA

Predictive Model for NBA	**Accuracy**
One-Layer Model based on Naive Bayes [29]	67%
One-Layer Model based on Simple Logistics Classifier [30]	69.49%
Sport Journalist [31]	71%
One-Layer Model based on Weka [32]	72.8%
NBA Oracle [31]	73%
One-Layer Model based on Neural Network [33]	74.33%
TLGProb	**85.28%**

where $\mu_i^\star = f_{\text{reg}}(\mathbf{X}_i^\star)$ is the predicted testing output from the trained regression model, and $(\mathbf{X}_i^\star, y_i^\star)$ is the ith input-output pair in the testing data. The statistical meaning of NLPD asserts that, the smaller the NLPD is, the better the regression model achieved. Noted that for non-Bayesian models, we get only a point estimate from each prediction. Therefore, we naturally use the Mean Square Error (MSE) of training to approximate the predictive variance

$$\begin{aligned}\text{Var}(\mathbf{y}^\star) \approx \text{Var}(\mathbf{y}) &= \mathbb{E}\left[(\mathbf{y} - \mathbb{E}[\mathbf{y}])^2\right] \\ &= \mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_i)^2,\end{aligned} \tag{58}$$

where $\mu_i = f_{\text{reg}}(\mathbf{X}_i)$ is the predicted training output from the trained regression model, and $(\mathbf{X}_i, y_i)$ is the ith input-output pair in the training data. Finally, the result in Table 3 proves our incentive that SSGPR, being a Bayesian regression model, is the best candidate for winning probability calculation among these regression models.

5.6 Comparing to Other Predictive Models

Several works [29, 30, 31, 32, 33] applying machine learning models to predict NBA results are also investigated. Beckler [31] and Cao [30] proposed to use NBA box score statistics of the team to classify winner of the next match. They tried Linear Classifier, Logistics Classifier, SVM Classifier, and Neural Network Classifier for prediction. Miljković [29] merged win-or-lose records and NBA box score statistics to create a new set of features and classified match outcomes using Naive Bayes. Zdravevski [32] used manual feature selection with classification techniques available in WEKA to predict the winner. Finally, decision tree gave the highest accuracy. Loeffelholz [33] used mid-season average statistics as prediction inputs and used fusion techniques to integrate different neural networks. Noted that all these models use only teams' performances for prediction, whereas individual performances of team players are neglected. We refer this kind of models as the One-Layer Model, as opposed to our stacked regression tasks which we refer as a Two-Layer Model.

The result of comparison in terms of accuracy is described in Table 4, in which TLGProb significantly surpassed the others. When acceptance threshold on the winning probability τ is naturally set to be 0.5, TLGProb correctly predicted the winning teams in 1,049 games out of 1,230 games, giving 85.28% accuracy. Noted that it is hard to improve the accuracy even just a small amount since the result of each game is controlled by many sources of non-deterministic factors, including player injuries, player attitudes, team rivalries and subjective officiating. We conceive that the success of our model is very much due to the high standard performance of SSGPR, explicit consideration of each player's contribution with position dependency, as well as the stacked regressions – associating players' performances with team's strength and associating two teams' performances with the match outcome.

5.7 Properties of Winning Probability

One vital feature of our model is winning probability calculation. In reality, this feature is highly appealing, especially for the determination of winning odds [2]. Simply concerning the task of one-match-ahead prediction, we can see that the winning probability returned from TLGProb is beneficial, as it provides us an option to increase the accuracy by conservatively rejecting the prediction with low confidence.

Here, the acceptance threshold τ acts as our confidence on the prediction result such that matches with winning probability smaller than τ are rejected. It is expected that, if the calculated winning probability is a reasonable measure of uncertainty, then, when we increase the acceptance threshold, the accuracy should be improved. In our results, it is exciting that the outputs obtained from TLGProb truly satisfy our expectation, as plotted on Figure 3. For example, if we set $\tau = 0.6$, the accuracy will be improved to 91.17%, while only 20% of the matches are rejected.

Taking advantage of the Bayesian framework, TLGProb is suited to work with Bayesian decision theory. That is, practitioners can optionally pick the match results where they have a strong belief in, and then ignore the matches with less confidence. For a realistic example, in wagering market, to carefully determine the winning odds, we should give high risks on the matches that high winning probability co-occurs high winning odds.

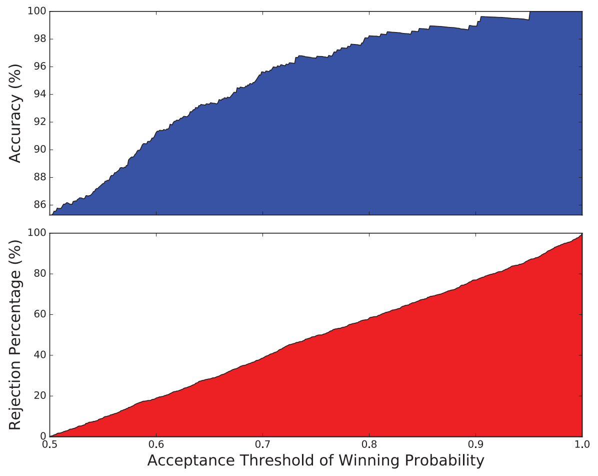

Figure 3. Plot of accuracies and rejection percentages when different acceptance thresholds of winning probability are used

6 Conclusion and Future Work

In this paper, we present a pioneering modeling approach for one-match-ahead forecasting in two-team sports featured with winning probability calculation. Our proposed model – TLGProb achieved 85.28% accuracy in NBA 2014/2015 season. The performance of TLGProb is superior to the existing NBA predictive models. We conceive that its success is mainly due to the high standard performance of SSGPR, explicit consideration of each player's contribution with position dependency, as well as the stacked regressions – associating players' performances with team's strength and associating two teams' performances with the match outcome.

Although superior performance in basketball domain was shown in this paper, we are keen on applying TLGProb to other two-team sports in the future so that comprehensive practicality of TLGProb can be proved. What's more, it is potential to extend our presented modeling approach to multi-teams sports, or any team-based competitions. It will be beneficial if our model can be generally applied as a pre-contest analytical tool.

References

[1] I. Bhandari et al., Advanced Scout: Data Mining and Knowledge Discovery in NBA Data, Data Mining and Knowledge Discovery, 1(1), 121–125, 1997.

[2] D. B. Hausch & W. T. Ziemba, Handbook of Sports and Lottery Markets, Elsevier, 2011.

[3] M. Ottaviani & P. N. Sørensen, Surprised by the Parimutuel Odds?, The American Economic Review, 99(5), 2129–2134, 2009.

[4] M. Haghighat et al., A Review of Data Mining Techniques for Result Prediction in Sports, Advances in Computer Science: an International Journal, 2(5), 7–12, 2013.

[5] M. Lázaro-Gredilla et la., Sparse Spectrum Gaussian Process Regression, Journal of Machine Learning Research, 11(Jun), 1865–1881, 2010.

[6] C. E. Rasmussen & C. K. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.

[7] D. J. MacKay, Introduction to Gaussian Processes. NATO ASI Series F Computer and Systems Sciences, 168, 133–166, 1998.

[8] D. Duvenaud, Automatic Model Construction with Gaussian Processes, Doctoral Dissertation, University of Cambridge, 2014.

[9] R. M. Neal, Bayesian Learning for Neural Networks, Springer Science & Business Media, 118, 2012.

[10] Y. Gal & R. Turner, Improving the Gaussian Process Sparse Spectrum Approximation by Representing Uncertainty in Frequency Inputs, In: 32nd International Conference on Machine Learning, 655–664, 2015.

[11] N. Wiener, Generalized Harmonic Analysis, Acta mathematica, 55(1), 117–258, 1930.

[12] A. Khintchine, Korrelationstheorie der Stationren Stochastischen Prozesse, Mathematische Annalen, 109(1), 604–615, 1934.

[13] S. Bochner, Monotone Funktionen, Stieltjessche Integrale Und Harmonische Analyse, Mathematische Annalen, 108(1), 378–410, 1933.

[14] J. Quiñonero-Candela et la., A Unifying View of Sparse Approximate Gaussian Process Regression, Journal of Machine Learning Research, 6(Dec), 1939–1959, 2005.

[15] J. S. Simonoff, Smoothing Methods in Statistics, Springer Science & Business Media, 2012.

[16] E. S. Gardner, Exponential Smoothing: The State of the Art, Journal of Forecasting, 4(1), 1–28, 1985.

[17] D. Oliver, Basketball on Paper: Rules and Tools for Performance Analysis, Potomac Books, Inc., 2004.

[18] W. L. Winston, Mathletics: How Gamblers, Managers, and Sports Enthusiasts Use Mathematics in Baseball, Basketball, and Football, Princeton University Press, 2012.

[19] D. Kingma & J. Ba, Adam: A Method for Stochastic Optimization, In: International Conference on Learning Representations 2014, 1–13, 2014.

[20] C. Saunders et la., Ridge Regression Learning Algorithm in Dual Variables. In: 15th International Conference on Machine Learning, 515–521, 1998.

[21] S. An et la., Face Recognition Using Kernel Ridge Regression. In: Computer Vision and Pattern Recognition 2007, IEEE, 1–7, 2007.

[22] M. Xu et la., Decision Tree Regression for Soft Classification of Remote Sensing Data, Remote Sensing of Environment, 97(3), 322–336, 2005.

[23] Y. Freund, & R. E. Schapire, A Desicion-Theoretic Generalization of On-line Learning and An Application to Boosting, In: European Conference on Computational Learning Theory 1995, Springer Berlin Heidelberg, 23–37, 1995.

[24] J. H. Friedman, Greedy Function Approximation: a Gradient Boosting Machine, Annals of Statistics, 1189–1232, 2001.

[25] L. Breiman, Random Forests, Machine learning, 45(1), 5–32, 2001.

[26] F. Pedregosa et la., Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12(Oct), 2825–2830, 2011.

[27] J. Quinonero-Candela et la., Evaluating Predictive Uncertainty Challenge. In: Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment, Springer Berlin Heidelberg, 1–27, 2006.

[28] J. Kohonen & J. Suomela, Lessons Learned in the Challenge: Making Predictions and Scoring Them. In: Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment, Springer Berlin Heidelberg, 1–27, 2006.

[29] D. Miljković et la., The Use of Data Mining for Basketball Matches Outcomes Prediction, In: 8th International Symposium on Intelligent Systems and Informatics (SISY), IEEE, 309–312, 2010.

[30] C. Cao, Sports Data Mining Technology Used in Basketball Outcome Prediction, Masters Dissertation, Dublin Institute of Technology, 2012.

[31] M. Beckler et la., NBA Oracle, $https : //www.mbeckler.org/coursework/2008 - 2009/10701_{r}eport.pdf$, 2013.

[32] E. Zdravevski, & A. Kulakov, System for Prediction of the Winner in a Sports Game, In: ICT Innovations, Springer Berlin Heidelberg, 55–63, 2009.

[33] B. Loeffelholz et la., Predicting NBA Games Using Neural Networks, Journal of Quantitative Analysis in Sports, 5(1), 1–15, 2009.

Max W. Y. Lam completed his bachelor's degree in Computer Science from The Chinese University of Hong Kong (CUHK). He then continued his research works as a research assistant in Stanley Ho Big Data Decision Analytics Research Centre, CUHK. He is currently in the process of completing his MPhil degree in Systems Engineering and Engineering Management. His major research interests are in machine learning, notably Gaussian process regression, neural networks, and their applications in probabilistic decision making.