UPPSALA
UNIVERSITET

TVE-F 19019
Examensarbete 15 hp
Juni 2019

Application and Further Development of TrueSkill™ Ranking in Sports

Julia Ibstedt
Elsa Rådahl
Erik Turesson
Magdalena vande Voorde

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Application and Further Development of TrueSkill™ Ranking in Sports

Julia Ibstedt, Elsa Rådahl, Erik Turesson, Magdalena vande Voorde

The aim of this study was to explore the ranking model TrueSkill™ developed by Microsoft, applying it on various sports and constructing extensions to the model. Two different inference methods for TrueSkill was constructed using Gibbs sampling and message passing. Additionally, the sequential method using Gibbs sampling was successfully extended into a batch method, in order to eliminate game order dependency and creating a fairer, although computationally heavier, ranking system. All methods were further implemented with extensions for taking home team advantage, score difference and finally a combination of the two into consideration. The methods were applied on football (Premier League), ice hockey (NHL), and tennis (ATP Tour) and evaluated on the accuracy of their predictions before each game.

On football, the extensions improved the prediction accuracy from 55.79% to 58.95% for the sequential methods, while the vanilla Gibbs batch method reached the accuracy of 57.37%. Altogether, the extensions improved the performance of the vanilla methods when applied on all data sets. The home team advantage performed better than the score difference on both football and ice hockey, while the combination of the two reached the highest accuracy. The Gibbs batch method had the highest prediction accuracy on the vanilla model for all sports.

The results of this study imply that TrueSkill could be considered a useful ranking model for other sports as well, especially if tuned and implemented with extensions suitable for the particular sport.

Handledare: Riccardo Sven Risuleo
Ämnesgranskare: Ken Welch
Examinator: Martin Sjödin
ISSN: 1401-5757, TVE-F 19019

Populärvetenskaplig sammanfattning

Att kunna uppskatta någons färdigheter är av stor vikt inom många olika tillämpningsområden. Det kanske mest uppenbara området där behovet är stort är inom sport och andra tävlingssammanhang där en skattning på en individs färdighet kan användas för att jämföra och hitta jämna motståndare. Med en fungerande rankningsmetod kan spelare som aldrig mött varandra ändå jämföras. I och med framkomsten av större onlinespel där spelare önskar möta andra spelare med en liknande färdighet växte också behovet av ett uppdaterat system för uppskattningar av färdigheter. I samband med detta utvecklades bland annat TrueSkill av Microsoft, vars första version publicerades 2006.

TrueSkill-modellen skattar en spelares färdighet enligt en normalfördelning, med medelvärde och varians. Dess matematiska grund bygger på Bayesiansk inferens och statistik, där vi uppdaterar vår uppskattning med hjälp av empiriska observationer och utfall. Kort sagt kan det sägas att en vinst höjer värdet på ens av TrueSkill uppskattade färdighet och att variansen sjunker med ett ökat antal spelade matcher, och vice versa. Storleken av fördelningens uppdatering beror på hur oväntat resultatet tros vara, om utfallet är väntat sker inga stora förändringar, men vid ett oväntat utfall sker stora uppdateringar.

Huvudsyftet med vårt projekt var att undersöka hur TrueSkill-modellen kan implementeras och vidareutvecklas för att kunna uppskatta färdigheter inom olika sporter. Vi implementerade två grundläggande numeriska metoder som applicerar TrueSkill enligt olika tillvägagångssätt: samplingmetoden Gibbs sampling samt message passing. Vi testade metoderna för tre olika sporter: fotboll (Premier League), hockey (NHL) och tennis (ATP Tour).

För att vidareutveckla våra metoder och göra dem bättre anpassade till just sport, med särskilt fokus på fotboll, utvecklade vi den grundläggande modellen till att även ta hänsyn till hemmaplansfördel och målskillnader istället för att enbart ta hänsyn till vinnaren. Dessa vidareutvecklingar förbättrade metodernas förmåga att förutsäga resultat för hockey och fotboll. De bästa resultaten gavs av att kombinera våra vidareutvecklingar. För fotbollsdatan gav de grundläggande metoderna en prediktiv förmåga på 55.79% och den kombinerade metoden för hemmalagsfördel och målskillnader höjde förmågan till 58.95%.

De slutsatser vi kan dra gällande TrueSkill som rankningsmetod är att den är väl fungerande för sportresultatsdata, och att metoderna Gibbs sampling och message passing ger snarlika resultat i förmåga att förutspå spelresultat. Metoderna visade sig kunna förbättras genom utvecklingar anpassade för de sporter vi testade, och det finns stor potential för vidare utveckling inom sportrelaterad rankning.

List of Symbols

Symbol	Description
$\mathcal{N}(x;\mu,\sigma^2)$	Meaning x is distributed according to the Gaussian probability density function with mean μ and variance σ^2, following the equation $\mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$
$\Phi(x)$	Standard normal cumulative function: the area under the density function $\mathcal{N}(x;0,1)$ from $-\infty$ to x
μ	Mean of a distribution
σ^2	Variance of a distribution
$\pi = \sigma^{-2}$	Precision, the inverse of the variance
$\lambda = \pi\mu$	Precision adjusted mean
s_i	Skill distribution of player i
p_i	Performance distribution of player i
β^2	Performance variance
p_{diff}	Performance difference
ε	Draw margin
ε_i	Score margin
y	Game outcome
$\delta(x)$	Dirac delta function: value ∞ if expression equals 0, and value 0 otherwise, and with the property $\int_a^b \delta(x) = 1$ if $a < 0 < b$
$\mathbb{1}_{x>k}$	Indicator function equal to 1 if $x > k$, and 0 otherwise
x_i	Variable node
f_j	Factor node
$m_{f_j \to x_i}$	Message from a factor node to a variable node
$m_{x_i \to f_j}$	Message from a variable node to a factor node

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Purpose . . . 2
- 1.3 Contributions . . . 2

2 Theory **3**
- 2.1 Bayesian Inference . . . 3
- 2.2 Bayesian Networks . . . 4
- 2.3 Gibbs Sampling . . . 4
- 2.4 Factor Graphs . . . 5
- 2.5 Message Passing . . . 6
- 2.6 Expectation Propagation . . . 8

3 Vanilla TrueSkill Model **9**

4 Inference Algorithms **13**
- 4.1 TrueSkill with Gibbs Sampling . . . 13
  - 4.1.1 Sequential Method . . . 13
  - 4.1.2 Batch Method . . . 14
- 4.2 TrueSkill with Message Passing . . . 15

5 Extensions **18**
- 5.1 Home Team Advantage . . . 18
- 5.2 Score Difference . . . 19
- 5.3 Home Team Advantage and Score Difference . . . 21

6 Evaluation **22**
- 6.1 Data Sets . . . 23

7 Results and Discussion **23**
- 7.1 Gibbs Sampler Design . . . 24
- 7.2 Applications on Different Data Sets . . . 25
  - 7.2.1 Football . . . 25
  - 7.2.2 Ice Hockey . . . 27
  - 7.2.3 Tennis . . . 28
- 7.3 Home Team Advantage Extension . . . 28
- 7.4 Score Difference Extension . . . 29
- 7.5 Discussion of Inference Methods . . . 30
- 7.6 Further Improvements and Applications . . . 31

8 Conclusions **31**

References **33**

1 Introduction

1.1 Background

Being able to statistically rank participants in sports and games is of great value in game matchmaking to create enjoyable and even game experiences, as well as for determining qualifications for various tournaments. It allows us to examine the individual skill developments which can be used for comparison between teams or players. An early but well-known skill ranking model is the Elo ranking system developed by Arpad Elo in 1959 [1]. It was designed as a chess rating system, hence aiming to predict the outcome of a one versus one player game with win and loss as possible outcomes while handling a draw as half a win and half a loss. However, it does not predict the probability of a draw occurring. The core idea of Elo is to model the probability of a game outcome from estimations of the players' skill represented by a single number.

TrueSkill™, a rating system designed for online gaming, was developed by Microsoft and released in 2006 [2]. It evaluates the skills as Gaussian distributions with a mean and a variance, compared to a single number in Elo. This adds an uncertainty to a player's rating and results in finding a correct rating in fewer games, whereas Elo would be as certain about a new player's skill as the skill of a player with many games played. Another great difference from the Elo system is that it can handle multiplayer games. It also handles draws in a different way than Elo by introducing a draw margin ε related to a draw probability, defining the difference in performance needed to not draw.

As described in the TrueSkill original paper [2], TrueSkill uses a Bayesian approach to find the skill of a player given a game outcome. This is done by constructing a model with a latent skill variable s_i for each player i and assuming that s_i is a Gaussian distribution. The prior probability distribution becomes $p(s_i) = \mathcal{N}(s_i; \mu_i, \sigma_i^2)$ with mean μ_i and variance σ_i^2. A large variance σ_i^2 signals that the model is unsure about the correct latent skill μ_i, whereas a smaller variance tells us that there is a higher certainty in the rating. Each player is said to achieve a performance p_i in a given game, which is also assumed to be Gaussian distributed centered around the player's skill s_i with additional fixed variance β^2, and thus the performance distribution of player i is $p(p_i) = \mathcal{N}(p_i; s_i, \beta^2) = \mathcal{N}(p_i; \mu_i, \sigma_i^2 + \beta^2)$.

Since TrueSkill is built for multiplayer games it needs to handle teams playing against each other, yet still calculate the independent skills for each player. To achieve this, teams are constructed by dividing a set of n players into K teams. Each team k is then defined by its players in the subset $A_k \subset \{1, ..., n\}$. A team performance t_k for team k is introduced and modeled as the sum of all performances of players i belonging to that team such that

$$t_k = \sum_{i \in A_k} p_i.$$

The outcome of a game is introduced as $\mathbf{r} = (r_1, ..., r_K) \in \{1, ..., K\}$ where r_k is the resulting rank of team k in the played game. While team l is expected to win over team m if $t_l > t_m$, for the possibility of the teams drawing due to a small performance difference

needs to be considered. On that account a draw margin ε is introduced, defined by the assumption that if $|t_l - t_m| < \varepsilon$ we expect a draw.

Finally the calculated probabilities can be put into Bayes' theorem. The posterior probability distribution of $\mathbf{s}$ given a game outcome $\mathbf{r}$ is then,

$$p(\mathbf{s}|\mathbf{r}, A) = \frac{p(\mathbf{r}|\mathbf{s}, A)p(\mathbf{s})}{p(\mathbf{r}|A)},$$

where $p(\mathbf{s}) = \prod_{i=1}^{n} p(s_i)$ is the prior probability distribution of all players' skills, $p(\mathbf{r}|\mathbf{s}, A)$ is the likelihood of the game outcome given the prior skills and the team assignments and $p(\mathbf{r}|A)$ is the marginal distribution of the outcome. Computing this posterior distribution is intractable for this case, instead it can be approximated with either a factor graph with message passing and expectation propagation, as is done and described in the TrueSkill paper [2], or with Gibbs sampling described in Rasmussen's lecture slides [3].

An updated version focusing more on online multiplayer games known as TrueSkill 2 was published in 2018 [4]. It includes improvements such as:

- A player's skill is updated not only by respect to a win/loss, but can also be score-based. In e-games this can be translated to e.g. kill- and death counts.
- Skills are no longer assumed to be game mode-independent
- For players who are in a squad, their assumed performance is increased

This updated version of the basic TrueSkill was tested on the video game Halo 5. For historical match outcomes, it resulted in an improvement for prediction accuracy from 52% (TrueSkill) to 68% (TrueSkill 2), suggesting that the base model has ability to be further developed and refined according to different game aspects.

1.2 Purpose

This project aims to explore TrueSkill as a ranking method for various sport applications. Implementing a TrueSkill model in Python allows us to examine how it applies and performs on different sport-result data sets. By expanding the basic model, we aim to evaluate how the prediction performance can be improved by taking aspects such as score result and home team advantage into account. The model could ultimately be used to obtain qualified predictions of prospective sport results.

There is also an idea of using this project for educational purpose. The Department of Information Technology at Uppsala University is currently developing a new course called Advanced Probabilistic Machine Learning. Our aim is that this project will be of use when constructing a mini project to be used within that course.

1.3 Contributions

The main part of this project has consisted of implementation in Python. We have written code where we applied a TrueSkill model to sport-result data in two different

ways. Firstly, with message passing based on the method stated by Microsoft and used in basic TrueSkill [2]. Secondly, with Gibbs sampling, where the methods we used are partly based on lecture slides from professor Carl Edward Rasmussen [3].

We wanted to develop the Gibbs sampling method so that it could handle multiple games at once, making the order of games being played irrelevant and making it possible to rank team performances in a certain time window, such as a season in a sport league. Therefore, we constructed a batch method that processes all the result-data simultaneously instead of sequentially.

Moreover, we constructed two extensions to the vanilla TrueSkill model. They allow us to take home team advantage and score-based outcomes into account to improve the prediction accuracy. These have been implemented in the message passing method as well as the Gibbs sampling method and tested on three different sports: football, ice hockey and tennis.

2 Theory

Explained in this section is the theoretical foundation to the TrueSkill model and two different methods which can be used to implement it, Gibbs sampling and message passing. Both methods are based on Bayesian inference. While both methods are approximate, Gibbs sampling is a stochastic method while message passing is deterministic.

2.1 Bayesian Inference

Bayesian inference is a way to update a belief or hypothesis as new evidence is provided. It is mainly built upon Bayes' theorem which relates the probability of an event based on factors related to the event. Starting from the intuitive notion that the probability of two events A and B having the joint probability $P(A, B)$ can be written as the product of the marginal probability $P(B)$ and the conditional distribution of A given that B has already happened, $P(A|B)$, and vice-versa for $P(A)$ and $P(B|A)$, it can be written that

$$P(B|A)P(A) = P(A, B) = P(A|B)P(B),$$

which can be rewritten into Bayes' theorem,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{1}$$

In Bayesian inference the distribution of a hypothesis A, $P(A)$ in the case above, is updated according to Bayes' theorem. $P(A)$ is called the prior probability, the hypothesis before some evidence B is observed. The posterior probability $P(A|B)$ is then the hypothesis after the evidence has been introduced, and $P(B|A)$ is the probability of observing the evidence B given the hypothesis A, also called the likelihood.

2.2 Bayesian Networks

A way to represent probabilistic models visually is with graphical models. Three advantages for using graphical models are that they are powerful for visualizing the structure of a probabilistic model, they give an understanding of properties like conditional independence and they express complex computations according to Bishop [5], chapter 8. A graphical model consists of nodes and edges. A node represents a random variable and an edge between two nodes means that there is a probabilistic relationship between the two variables.

A Bayesian network, which is a probabilistic directed acyclic graphical model, is shown in Figure 1. Here each node represents a random variable and the edges represent conditional distributions. The direction of the edge expresses the dependency, thus if a directed edge points from one node, say node x_1, to node x_2 the conditional distribution is $p(x_2|x_1)$. The node x_1 is also known as the parent to node x_2, which is the child. Looking at all nodes and edges in a graph, it can be seen that a Bayesian network describes a joint distribution by the product of conditional distributions. A general definition for a Bayesian network is of the form,

$$p(x_1, ..., x_N) = \prod_{i=1}^{N} p(x_i|\text{pa}(x_i)), \tag{2}$$

where $\text{pa}(x_i)$ is the set of parents for x_i. The i^{th} node in the graph corresponds to the factor $p(x_i|\text{pa}(x_i))$ [6].

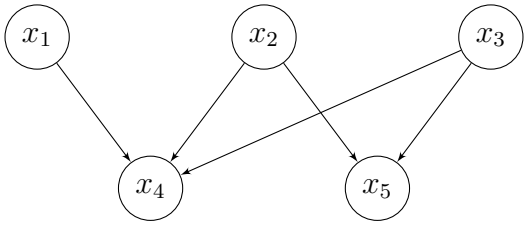

Figure 1: A Bayesian network representing a joint distribution with a variable node for each variable, x_i, and directed edges

A Bayesian network graphically shows the conditional independence by not drawing an edge between two nodes. As an example the joint distribution $p(x_1, x_2, x_3, x_4, x_5)$ is described in a Bayesian network and shown in Figure 1. Each variable x_1, x_2, x_3, x_4 and x_5 is represented by a node. If we have the Bayesian network shown in Figure 1, the joint distribution is given by $p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_2, x_3)$.

2.3 Gibbs Sampling

Gibbs sampling is often used in Bayesian inference and is based on the Metropolis-Hastings algorithm, which is a Markov chain Monte Carlo (MCMC) method [5]. MCMC is a class of algorithms used when sampling from an arbitrary probability distribution of two or more dimensions. Assume x_i is to be sampled, where $i \in \{1, N\}$, from the joint

probability distribution $p(\mathbf{x}) = p(x_1, x_2, ..., x_N)$. Firstly, the ground state $(x_1^0, x_2^0, ..., x_N^0)$ is initialized and then according to Gibbs sampling, the variables are respectively replaced repeatedly by independently sampling from the distribution of that variable being conditioned on the remaining variables. Thus, when a variable is replaced the previous value of that variable in the previous iteration is used to define a conditional probability distribution to sample from for the next value. This is done for all variables in every iteration. The first iteration can therefore be written as

$$\begin{aligned} x_1^1 &\sim p(x_1^1 | x_2^0, ..., x_N^0) \\ x_2^1 &\sim p(x_2^1 | x_1^1, ..., x_N^0) \\ &\vdots \\ x_N^1 &\sim p(x_N^1 | x_1^1, x_2^1, ..., x_{N-1}^0). \end{aligned}$$

The values of the variables are continued to be updated for the given τ number of iterations. The last iteration can then be written as

$$\begin{aligned} x_1^\tau &\sim p(x_1^\tau | x_2^{\tau-1}, ..., x_N^{\tau-1}) \\ x_2^\tau &\sim p(x_2^\tau | x_1^\tau, ..., x_N^{\tau-1}) \\ &\vdots \\ x_N^\tau &\sim p(x_N^\tau | x_1^\tau, x_2^\tau, ..., x_{N-1}^\tau). \end{aligned}$$

As the number of iterations increases, the sampled distribution approaches the true distribution, according to the central limit theorem. In the beginning however, the samples may deviate noticeably from the true distribution. This period before reaching stabilization is called the burn-in phase and is the number of samples required before reaching an accurate approximation of the true distribution. Thus, the samples within the burn-in are discarded when using the Gibbs sampler.

Representing $p(\mathbf{x})$ as a Bayesian network with the nodes as the variables $x_{1,...,N}$, the conditional distributions of node x_i is only dependent on its parents, children and children's other parents in the graph. This set of variables is also called the Markov blanket of x_i. That is, in order to sample x_i only the values of the nodes in its Markov blanket has to be known [7]. The final joint distribution can therefore be written as Equation (2).

2.4 Factor Graphs

Factor graphs is a way to express computations in graphical models and are used to visually describe factorization of a distribution function. Factor graphs are said to be appropriate to use instead of directed graphs like a Bayesian Network for the motive to solve inference problems, according to Bishop [5], chapter 8. In TrueSkill this is how it is used by being a part of the Bayesian inference algorithm. A factor graph is bipartite, thus it consists of two types of nodes where factor nodes, described as functions f_i, are given by a square and variable nodes, described as x_i, are given as circles for each variable.

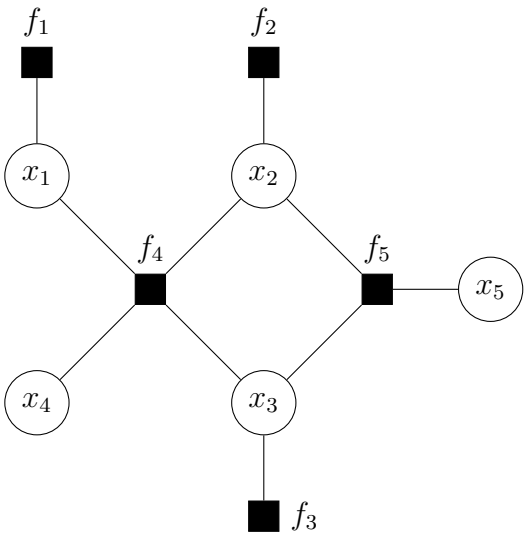

Figure 2: Factor graph of the Bayesian network in Figure 1. Each variable, x_i, has a corresponding variable node and each factor, f_k, has a corresponding factor node.

All edges in a factor graph are between nodes of different sorts, i.e. between a factor node and a variable node. An example of a factor graph is presented in Figure 2. It is equivalent to the Bayesian network in Figure 1, hence the joint distribution can be described as $f_1(x_1)f_2(x_2)f_3(x_3)f_4(x_1, x_2, x_3, x_4)f_5(x_2, x_3, x_5)$. The factors are given by $f_1(x_1) = p(x_1)$, $f_2(x_2) = p(x_2)$, $f_3(x_3) = p(x_3)$, $f_4(x_1, x_2, x_3, x_4) = p(x_4|x_1, x_2, x_3)$ and $f_5(x_2, x_3, x_5) = p(x_5|x_2, x_3)$.

2.5 Message Passing

As graphical models are used to visually see the structure of probabilistic models, message passing is used in efficient algorithms for inference problems on these graphical models. In message passing the main concept for it being efficient is how all the information from the graph is expressed and kept locally [6]. This is particularly efficient for calculations of marginals of variables in the graph. To illustrate this with an example inspired from Carl Edward Rasmussen's lecture note "Factor graphs" [3], a factor graph is seen in Figure 3 and represents a probabilistic model; by applying the idea of message passing it is shown how the computation is cheaper and hence more efficient: to compute the marginal distribution of x_4 the only messages needed are $m_{f_3 \to x_4}$ and $m_{f_4 \to x_4}$, and nothing else in the graph.

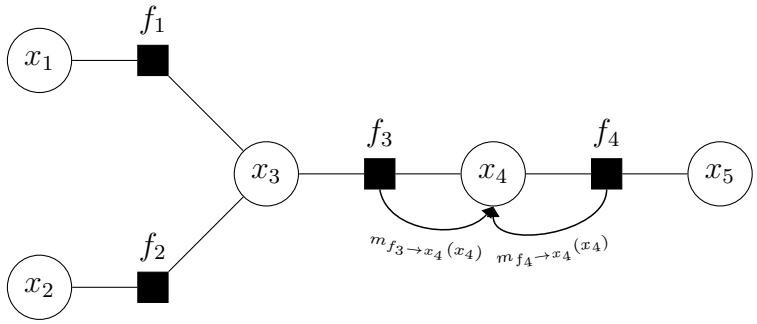

Figure 3: Illustrative example of message passing. Each variable, x_i, has a corresponding variable node and each factor, f_k, has a corresponding factor node. Two messages from f_3 and f_4 are shown as $m_{f_3 \to x_4}$ and $m_{f_4 \to x_4}$.

By definition [5], the marginal distribution for variable x_4 is,

$$p(x_4) = \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_5} f_1(x_1, x_3) f_2(x_2, x_3) f_3(x_3, x_4) f_4(x_4, x_5)$$

If in the set of variables each variable can take K states the computational cost for $p(x_4)$ will be K^5 as the sums are K^4 for each state K in x_4. Considering that multiplication is distributive over addition, $xy + xz = x(y + z)$, and in this example seeing that the right hand side requires less operations than the left hand side, the same property can be used in the computation of the marginal to lower the cost. Looking at the factor graph at variable x_4 the graph can be split in two parts, one on each side of the variable, this can then be expressed in computing the marginal as follows,

$$\begin{aligned} p(x_4) &= \left(\sum_{x_1} \sum_{x_2} \sum_{x_3} f_1(x_1, x_3) f_2(x_2, x_3) f_3(x_3, x_4) \right) \cdot \left(\sum_{x_5} f_4(x_4, x_5) \right) \\ &= m_{f_3 \to x_4}(x_4) \cdot m_{f_4 \to x_4}(x_4), \end{aligned}$$

which lower the computational cost to K^4 as in the simple example above. This leads to the idea of message passing, by expressing the two groups of sums that are multiplied as incoming messages along the edges to the variable from the left- and right-hand side.

One efficient message passing algorithm formed specifically for tree-structured factor graphs is the sum-product algorithm. In the sum-product algorithm there are three updating equations, two types of messages, one for a message from a variable node, x, to a factor node, f, and vice versa, and a general form for computing marginals. The marginal at a variable node, x_k, is given by the product of all incoming messages from the set of neighboring factor nodes, $F(x_k)$, to the variable,

$$p(x_k) = \prod_{f \in F(x_k)} m_{f \to x_k}(x_k).$$

The message from a factor node to a variable node takes the product of all incoming messages to the factor node except the one from the variable which the message is sent

to, x_j, multiplies by the factor given at that factor node, f_s, and then marginalizes over all the variables related to the incoming message. The set of all variables associated with the factor node f_s is $x_j, x_1, x_2, ..., x_m$ and can be denoted as $\mathbf{x_s}$. The message is expressed as

$$m_{f_s \to x_j}(x_j) = \sum_{x_1} \sum_{x_2} ... \sum_{x_m} f_s(\mathbf{x_s}) \prod_{i \neq j} m_{x_i \to f_s}(x_i).$$

At last the message from a variable node to a factor node is the product of all the incoming messages to the variable node, x_k, except the one from the factor node, f_s, which the message is sent to. That message is given as

$$m_{x_k \to f_s}(x_k) = \prod_{i \in F(x_k) \setminus f_s} m_{f_i \to x_k}(x_k) = \frac{p(x_k)}{m_{f \to x_k}(x_k)}.$$

Implementing the sum-product algorithm on the example mentioned before gives

$$p(x_4) = \left(\sum_{x_3} f_3(x_3, x_4) \cdot \left(\left(\sum_{x_2} f_2(x_2, x_3) \right) \cdot \left(\sum_{x_1} f_1(x_1, x_2) \right) \right) \right) \cdot \left(\sum_{x_5} f_4(x_4, x_5) \right),$$

which lowers the computational cost to K^2. In TrueSkill, the sum-product algorithm is used for message passing in the factor graph.

2.6 Expectation Propagation

There are situations where inference can not be performed exactly with message passing and instead needs approximation. In TrueSkill the game outcome leads to an intractable truncated Gaussian that gets approximated to a tractable Gaussian, achieved by using the method expectation propagation. Expectation propagation was introduced by Tom Minka [8]. In short, the way expectation propagation approximates is by minimizing the Kullback-Leibler divergence, a measurement of how good an approximated distribution, $q(x)$, is compared to the true distribution, $p(x)$. If $q(x)$ is a Gaussian, the Kullback-Leibler divergence is minimized by choosing the mean and variance of $q(x)$ equal to the mean and variance of $p(x)$. This case of expectation propagation is sometimes called moment matching [5]. An algorithm for expectation propagation used in TrueSkill can be seen in Algorithm 1.

Algorithm 1: Expectation Propagation

1. Consider expectation propagation as outlined in Bishop, chapter 10.7 [5] for the specific case of pairwise messages. Let $\mathbf{x}$ be the set of variables and $m_i(x)$ the messages sent in a factor graph, here represented as factors which constitute the marginal likelihood $p(x)$. Declare the distribution as follows:

$$p(\mathbf{x}) = m_1(x)m_2(x)$$

2. Approximate the distribution, where $\hat{m}_i(x)$ is the approximated messages, as the posterior:

$$q(x) \propto \hat{m}_1(x)\hat{m}_2(x)$$

3. Repeat until convergence is reached:
   - Remove $\hat{m}_1(x)$ from the posterior distribution by division, resulting in the new posterior:

$$q^{\setminus 1}(x) \propto \frac{q(x)}{\hat{m}_1(x)} = \hat{m}_2(x)$$

   - Find a new $\hat{m}_1^{'}(x)$ that reduces the distance between the true and approximated posterior by minimizing the Kullback-Leibler divergence:

$$q^{'}(x) = \underset{\hat{m}_1(x)}{\arg\min}\, D_{\mathrm{KL}}(f_1(x)\hat{m}_2(x)||\hat{m}_1(x)\hat{m}_2(x))$$

   - Replace the removed factor $\hat{m}_1(x)$ with $\hat{m}_1^{'}(x)$, resulting in a new updated posterior:

$$q^{'}(x) = \hat{m}_1^{'}(x)\hat{m}_2(x)$$

   - Repeat the substeps of step 3, but now removing $\hat{m}_2(x)$ from the new $q^{'}(x)$, resulting in yet a new posterior:

$$q^{''}(x) = \hat{m}_1^{'}(x)\hat{m}_2^{'}(x)$$

4. Finally, approximate the marginal likelihood $p(D)$ as:

$$p(D) \approx \int \prod_i \hat{m}_j^{'}(x)dx$$

3 Vanilla TrueSkill Model

TrueSkill uses the intuitive idea that a game outcome reflects the players' performances which in turn is dependent on the players' skills, i.e. that a player with a higher skill than another player does a better performance and wins. In the TrueSkill model the approximated skill of a player is seen as a Gaussian distribution, hence the mean gives the skill value and the variance the uncertainty of the skill value. The TrueSkill model then performs Bayesian inference to find and update the skill of a player given a game outcome. To illustrate this concept, two players' approximated skills from TrueSkill before and after a game are shown in Figure 4. To the left in the figure are the skills, s_1 and s_2, before a game and it is shown that player two, seen in red, seems to be estimated

as the more skillful player with mean $\mu_2 = 30$, compared to player one, seen in blue, with mean $\mu_1 = 20$. By also studying the variance of both players, $\sigma_1^2 = 6.0^2$ and $\sigma_2^2 = 2.0^2$, TrueSkill seems to be more certain of the approximated skill of player two.

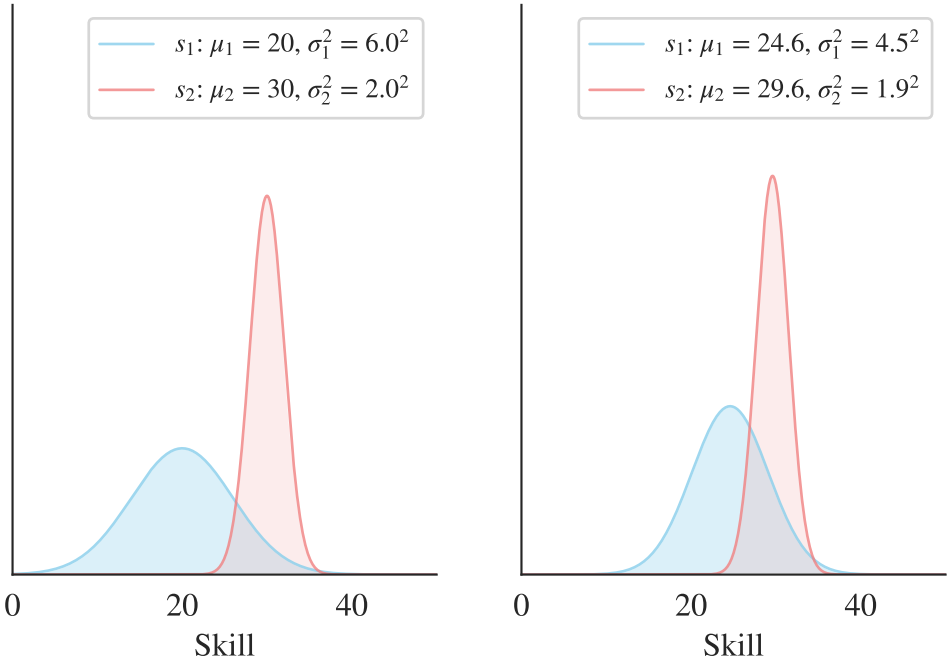

Figure 4: The skill distributions of two players before a game, to the left, and after a game, to the right. Player one, with skill s_1 having mean and variance μ_1 respectively σ_1^2, is seen in blue and player two, with skill s_2 having mean and variance μ_2 respectively σ_2^2, in red.

Then there is a game between the two players where the first player wins, TrueSkill now updates the players' skills given the game outcome and the new approximated skills are shown in Figure 4 to the right. The mean and variance for both players are adjusted. The losing player, in red, now has a mean of μ_2=29.6 which is lower than before and the winning player, in blue, has a mean of $\mu_1 = 24.6$ which is higher than before. The variances of both players have decreased to $\sigma_1^2 = 4.5^2$ and $\sigma_2^2 = 1.9^2$. Player one has a more significant adjustment than player two because of the differences in prior variances.

To implement a simple TrueSkill model a game between two players is considered, where the game outcome, y, is a random variable and each player's skill, s_i, is a latent variable given as s_1 for player one, and s_2 for player two. The skill is Gaussian distributed as $s_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ for $i = 1, 2$. Taking the difference between the two players' skills, s_{diff}, and adding a noise term, $\eta \sim \mathcal{N}(0, 2\beta^2)$, for performance inconsistency, the performance difference, $p_{\text{diff}} = s_{\text{diff}} + \eta$, is received. Here β is the performance variance, a model parameter determining how much a performance varies around a set skill for each individual player. The game outcome can then be computed as

$$y = f(p_{\text{diff}}) = \begin{cases} 1 & \text{if} \quad p_{\text{diff}} > \varepsilon \\ 0 & \text{if} \quad |p_{\text{diff}}| \leq \varepsilon \\ -1 & \text{if} \quad p_{\text{diff}} < -\varepsilon, \end{cases}$$

where ε is the draw margin, which is implemented to handle a draw between the players. In the original TrueSkill paper [2], the authors define ε through the relation

$$p(\text{draw}) = 2\Phi\left(\frac{\varepsilon}{\beta\sqrt{n}}\right) - 1, \tag{3}$$

where draw probability, $p(\text{draw})$, is the empirical probability of a draw from all played games in a sample size, $\Phi(x)$ is the cumulative Gaussian probability distribution and n is the total number of players [2]. In this case, $n = 2$ and Equation (3) is solved for ε resulting in

$$\varepsilon = \sqrt{2}\beta\Phi^{-1}\left(\frac{1 + p(\text{draw})}{2}\right). \tag{4}$$

The whole model for a two player game can then be constructed as

$$\begin{cases} s_1 \sim \mathcal{N}(\mu_1, \sigma_1^2) \\ s_2 \sim \mathcal{N}(\mu_2, \sigma_2^2) \\ s_{\text{diff}} = s_1 - s_2 \\ p_{\text{diff}} = s_{\text{diff}} + \eta \\ \eta \sim \mathcal{N}(0, 2\beta^2) \\ y = f(p_{\text{diff}}), \end{cases} \tag{5}$$

To illustrate this model in Equation (5), a Bayesian network is shown in Figure 5, where each variable is represented by a node and the relationship between the variables is shown with directed edges.

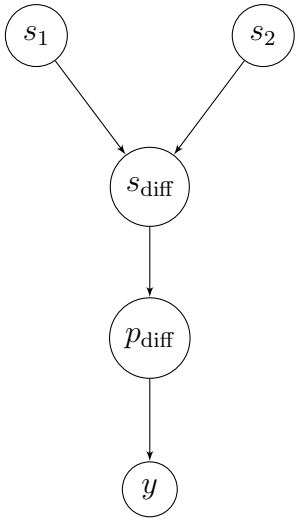

Figure 5: A Bayesian Network representing the TrueSkill model, with variable nodes for each variable in the model seen in Equation (5)

The goal is to find the skills given a game outcome, i.e. $p(s_1, s_2|y)$. According to Bayes' theorem in Equation (1), this can be described as

$$p(s_1, s_2|y) = \frac{p(y|s_1, s_2)p(s_1, s_2)}{p(y)}. \tag{6}$$

The skill of one player is said to be independent of another player's skill, hence the prior is given as $p(s_1, s_2) = p(s_1)p(s_2)$ where $p(s_i) \sim \mathcal{N}(s_i; \mu_i, \sigma_i^2)$. The likelihood, $p(y|s_1, s_2)$, can be seen in Figure 5 and is given by

$$p(y|s_1, s_2) = \iint p(y|p_{\text{diff}})p(p_{\text{diff}}|s_{\text{diff}})p(s_{\text{diff}}|s_1, s_2)ds_{\text{diff}}dp_{\text{diff}}.$$

The marginal likelihood for given outcome is defined as

$$p(y) = \begin{cases} P[y=1] = P[p_{\text{diff}} > \varepsilon] = 1 - P[p_{\text{diff}} < \varepsilon] = 1 - \phi(\frac{\varepsilon - \mu_1 + \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2 + 2\beta^2}}), & \text{if} \quad y = 1 \\ P[y=-1] = P[p_{\text{diff}} < -\varepsilon] = \phi(\frac{-\varepsilon - \mu_1 + \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2 + 2\beta^2}}), & \text{if} \quad y = -1 \\ P[y=0] = -\phi(\frac{\varepsilon - \mu_1 + \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2 + 2\beta^2}}) - \phi(\frac{-\varepsilon - \mu_1 + \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2 + 2\beta^2}}), & \text{if} \quad y = 0. \end{cases} \tag{7}$$

Computing the posterior skill distribution $p(s_1, s_2|y)$ from Equation (6) is intractable and can instead be approximated by different methods such as Gibbs sampling and message passing with expectation propagation.

As more observations of results are made, the certainty of ratings will increase. Thus the model of Equation (5) would ultimately result in the rating variance σ^2 of a player to decrease for each game played, making the rating harder to change with time as more games are played. To prevent this and allow the skill rating to change over time, a

dynamic factor γ can be added to the uncertainty of a player's rating variance between each game, such that $s_i^{t+1}|s_i^t \sim \mathcal{N}(s_i^t; \gamma^2)$, so that the prior skill distribution to game $t+1$, s_i^{t+1}, has the same mean as the posterior skill distribution to game t, s_i^t, and variance equal to the variance of s_i^t plus an additional γ^2.

4 Inference Algorithms

Two different inference algorithms, message passing and Gibbs sampling, are implemented on the vanilla TrueSkill model. The message passing method is based on the theory used by Microsoft's TrueSkill model. The other approach is Gibbs sampling, a way to sample a probability distribution. The Gibbs sampling sequential method was also extended into a batch method, where all the results are handled in a batch, making sure that the game order does not affect the predictions.

The programming language Python was used to implement the model with inference algorithms and obtain interpretable results. Python was considered appropriate for this project due to its wide range of suitable libraries adapted for statistics.

4.1 TrueSkill with Gibbs Sampling

Gibbs sampling is a method where a sought after joint distribution is approximated by extracting samples. As mentioned in Section 2.3, it involves a sampling procedure where, for each step, one of the variable's value is replaced with a new one. The new value is drawn from a distribution of that specific variable, conditioned on the values of the rest of the variables [5]. This method is well suited for sampling the posterior joint distribution for TrueSkill, since the conditional distributions are known. Following sections describe the basic sequential method as well as our extension to a batch method.

4.1.1 Sequential Method

As described in Section 3, computing $p(s_1, s_2|y)$ from Equation (6) is intractable. Instead, for this method this is approximated by the sampled joint distribution $p(s_1, s_2, p_{\text{diff}}|y)$, which is computed by sampling from $p(p_{\text{diff}}|s_1, s_2, y)$ and $p(s_1, s_2|p_{\text{diff}}, y)$, since these are tractable. In the sampling process, $p(p_{\text{diff}}|s_1, s_2, y)$ is sampled given fixed s_1 and s_2 and $p(s_1, s_2|p_{\text{diff}}, y)$ is sampled given a fixed p_{diff}, alternating between the two. Hence in the Gibbs sampling method, initial values for s_1 and s_2 are taken as means μ_1 and μ_2 from $p(s_1)$ and $p(s_2)$. These values are then used in

$$p(p_{\text{diff}}|s_1, s_2, y) = k \cdot p(y|p_{\text{diff}})p(p_{\text{diff}}|s_1, s_2) \propto \delta(y - f(p_{\text{diff}})) \cdot \mathcal{N}(p_{\text{diff}}; , s_1 - s_2, 1),$$

which can be calculated by using rejection sampling from $\mathcal{N}(p_{\text{diff}}; s_1 - s_2, 1)$, discarding all values for $f(p_{\text{diff}})$ not equal to y. The k is introduced as the normalization constant to collect both $p(p_{\text{diff}})$, since it is independent of s_i, and the constant terms from the distributions. The sampled value p_{diff} is then used in

$$p(s_1, s_2|p_{\text{diff}}, y) = p(s_1, s_2|p_{\text{diff}}).$$

This equality can be seen in the Bayesian network, see Figure 5. This distribution is given by Bayes' theorem (1),

$$p(s_1, s_2|p_{\text{diff}}) = \frac{p(p_{\text{diff}}|s_1, s_2)p(s_1, s_2)}{p(p_{\text{diff}})}, \tag{8}$$

where $p(p_{\text{diff}}|s_1, s_2) = \mathcal{N}(p_{\text{diff}}; s_1 - s_2, 2\beta^2)$ and $p(s_1, s_2) = \mathcal{N}(s_1; \mu_1, \sigma_1^2) \cdot \mathcal{N}(s_2; \mu_2, \sigma_2^2)$. Then if the Gaussian distributions are expanded, the expression of the right-hand side of Equation (8) is

$$k \exp\left\{-\frac{1}{2}[(p_{\text{diff}} - (s_1 - s_2))^2 + \frac{1}{\sigma_1^2}(s_1 - \mu_1)^2 + \frac{1}{\sigma_2^2}(s_2 - \mu_2)^2]\right\}, \tag{9}$$

where k is a normalization constant. Declaring the following matrices,

$$P = \begin{bmatrix} 1 - \frac{1}{\sigma_1^2} & -1 \\ -1 & 1 - \frac{1}{\sigma_2^2} \end{bmatrix}^{-1}, \quad m = P \begin{bmatrix} \frac{\mu_1}{\sigma_1^2} - p_{\text{diff}} \\ \frac{\mu_2}{\sigma_2^2} - p_{\text{diff}} \end{bmatrix},$$

Equation (9) results in

$$p(s_1, s_2|p_{\text{diff}}) = \mathcal{N}(s_1, s_2; m, P).$$

Thus, Equation (8) is the multivariate Gaussian distribution $\mathcal{N}(\mathbf{s}; m, P)$, and the sampled values s_1 and s_2 are found and then used to sample p_{diff} and start the process all over again. In Gibbs sampling this process is repeated multiple times for a game, discarding the first set of sampled values of s_1 and s_2, the burn-in. This results in the sampled distribution of skills given a game outcome.

In the case of several played games as input data, the model runs sequentially, updating the skills between each game. It begins with the first game in the set, where the posterior joint distribution is approximated as a Gaussian with the same mean and variance as the sampled distribution. For each new game, a player's previously approximated posterior skill distribution is set as the prior skill distribution. This sort of algorithm is called Assumed Density Filtering (ADF).

4.1.2 Batch Method

In the sequential method above, using ADF, there is a dependency on which order the games are played, assuming results are listed in chronological order. This is because the skill is inclined to increase or decrease more at the first games as the system's certainty increases with each game played, meaning the updates of player's skills become smaller

as time goes on. Consider the following example: two players face each other, both with the same skill value. The winner's skill increases and the loser's decreases. At the next game the loser faces a new player with the same skill as the two teams had at the previous game. If the new player wins then its skill would increase, but not as much as the winner in the previous game since this time the loser already had a lower skill, even though they have displayed identical results. To avoid this order dependency, we implemented a batch method. In this method a set of games between different teams is considered at the same time, taking away the order dependency.

A matrix A is defined where each team is represented by a column and each game by a row. For one game, the home team is assigned the value 1 and the away team -1. The teams that are not participating in this specific game is assigned 0. The vector $\mathbf{s}$ of all players' ratings is distributed as a multivariate Gaussian with prior mean vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots]$ and a diagonal precision matrix Λ where $\Lambda_{ii} = \frac{1}{\sigma_i^2}$ is the precision of the rating of player i. Consider $\mathbf{p}_{\text{diff}}$, a vector including each game's performance difference, conditioned on $\boldsymbol{s}$. Its mean is then given as $A\mathbf{s}$ with performance variance $2\beta^2$ represented by the precision matrix B with $\frac{1}{2\beta^2}$ on the diagonal. Thus

$$\begin{cases} p(\mathbf{s}) \sim \mathcal{N}(\boldsymbol{\mu},\ \Lambda^{-1}) \\ p(\mathbf{p}_{\text{diff}}|\mathbf{s}) \sim \mathcal{N}(A\mathbf{s},\ B^{-1}). \end{cases}$$

By introducing a covariance matrix Σ as

$$\Sigma = (\Lambda + A^T B A)^{-1}$$

and applying Bayes' theorem, $p(\mathbf{s}|\mathbf{p}_{\text{diff}})$ can be calculated as

$$p(\mathbf{s}|\mathbf{p}_{\text{diff}}) \sim \mathcal{N}\left(\Sigma(A^T B \mathbf{p}_{\text{diff}} + \Lambda\boldsymbol{\mu}),\ \Sigma\right),$$

which is used in the TrueSkill with Gibbs sampling method to update the teams' individual skill by sampling the performance difference for each game at the same time.

4.2 TrueSkill with Message Passing

Using message passing, as mentioned in Section 2.5, for the TrueSkill model the set of variables in the model are arranged in a factor graph, see Figure 6. In this case the former variable s_{diff} from the model, see Equation (5), is not included. Instead it is replaced with a performance variable, p_i, for each player i=1,2, where p_i is distributed as a Gaussian centered at the skill, s_i, such that $p_i \sim \mathcal{N}(s_i, \beta^2)$. The performance difference variable is now declared as $p_{\text{diff}} = p_{winner} - p_{loser}$.

Performing Bayesian inference with message passing on the factor graph seen in Figure 6 starts by passing down messages with the sum-product algorithm following the black arrows to get a prior for p_{diff}. Then including the game outcome gives a posterior for p_{diff}, this posterior is analytically intractable and thus approximated with expectation

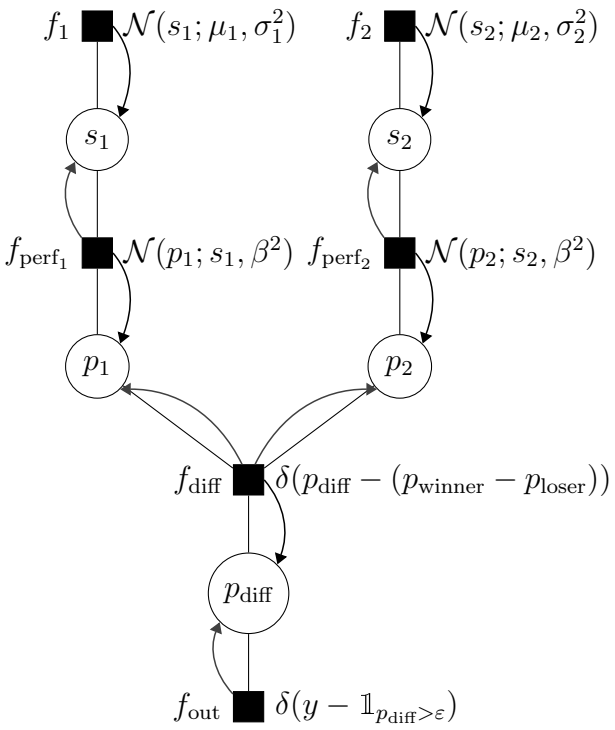

Figure 6: The TrueSkill factor graph for the case of two players. There are three variables with corresponding variable nodes: the skill of player i as s_i, performance as p_i and the performance difference as p_{diff}. The first row of factor nodes encodes the prior and the product of the remaining factor nodes is the likelihood. The arrows show the updating scheme.

propagation, see Algorithm 1, to a new approximated Gaussian. By iterating between the factor nodes f_{diff} and f_{out}, this approximated Gaussian is updated until convergence. Lastly, messages are passed back up the graph with the sum-product algorithm, following the dark grey arrows, using posterior values to update the skill nodes.

In more detail, the factor graph and messages could be explained as follows: at the top of the factor graph are the prior factors $f_i = \mathcal{N}(\mu_i, \sigma_i^2)$. Starting from the prior factors the skill marginals are updated by the downward message $m_{f_i \to s_i}(s_i) = \mathcal{N}(\mu_i, \sigma_i^2)$, to the variables s_i. Thus, by the sum-product algorithm, the marginal probability distribution of the skill of player i is $p(s_i) = \mathcal{N}(s_i; \mu_i, \sigma_i^2)$ before further messages are sent. Then the next messages are passed on to the corresponding performance factors $f_{\text{perf}_i} = \mathcal{N}(s_i, \beta^2)$. The performance marginals p_i are updated with the messages $m_{f_{\text{perf}_i} \to p_i}(p_i) = \mathcal{N}(s_i, \beta^2)$, adding the performance variance β^2 to the already existing skill marginal. The message sent down from f_{diff} to update the performance difference marginal p_{diff} is the difference between the incoming messages from p_i such that

$$m_{f_{\text{diff}} \to p_{\text{diff}}}(p_{\text{diff}}) = m_{p_{\text{winner}} \to f_{\text{diff}}}(p_{\text{winner}}) - m_{p_{\text{loser}} \to f_{\text{diff}}}(p_{\text{loser}}),$$

which before the outcome is observed as $\mathcal{N}(p_{\text{diff}}; \mu_{\text{winner}} - \mu_{\text{loser}}, \sigma^2_{\text{winner}} + \sigma^2_{\text{winner}} + 2\beta^2)$. The two bottom factors are Dirac delta-functions, $\delta(\text{expression})$, which means that their value is ∞ if the expression within the brackets equals 0, otherwise it is 0. The outcome

factor is given as $f_{\text{out}} = \delta(y - \mathbb{1}_{p_{\text{diff}}>\varepsilon})$, where

$$\mathbb{1}_{p_{\text{diff}}>\varepsilon} = \begin{cases} 1 & \text{if} \quad p_{\text{diff}} > \varepsilon \\ 0 & \text{otherwise,} \end{cases}$$

relays the game outcome to p_{diff} where $y = 1$ implies a win, and $y = 0$ that a draw occurred.

In an exact inference solution the upwards message $m_{f_{\text{out}} \to p_{\text{diff}}}(p_{\text{diff}})$ would be a three-dimensional categorical distribution. However, by the sum-product algorithm of message passing this would mean that the marginal distribution of the performance difference $p(p_{\text{diff}})$ would no longer be Gaussian, but a product of a cumulative Gaussian distribution and a Gaussian distribution. In order to obtain a closed-form expression that can be used, expectation propagation is used to make a Gaussian approximation of the marginal such that it has the same mean and variance of the exact distribution.

In order to simplify the multiplications of Gaussians, a Gaussian precision is introduced as the inverse of the variance, $\pi = \frac{1}{\sigma^2}$, and a precision adjusted mean as $\lambda = \pi\mu$. This means that the product of two Gaussians $\mathcal{N}(\pi_1, \lambda_1) \cdot \mathcal{N}(\pi_2, \lambda_2)$ is simply $\mathcal{N}(\pi_1 + \pi_2, \lambda_1 + \lambda_2)$. If $m_{f_{\text{diff}} \to p_{\text{diff}}}(p_{\text{diff}}) = \mathcal{N}(\lambda, \pi)$, performing expectation propagation on the marginal of p_{diff} results in the message from the outcome factor being

$$m_{f_{\text{out}} \to p_{\text{diff}}}(p_{\text{diff}}) = \mathcal{N}\left(\frac{\lambda + \sqrt{\pi} \cdot V(\lambda/\sqrt{\pi}, \varepsilon\sqrt{\pi})}{1 - W(\lambda/\sqrt{\pi}, \varepsilon\sqrt{\pi})}, \frac{\pi}{1 - W(\lambda/\sqrt{\pi}, \varepsilon\sqrt{\pi})}\right), \tag{10}$$

where ε is the draw margin and V and W are the following, based on if the game outcome is a win or a draw:

$$\begin{gathered} V_{\text{Win}}(t,\ \alpha) = \frac{\mathcal{N}(t-\alpha)}{\Phi(t-\alpha)}, \quad W_{\text{Win}}(t,\ \alpha) = V_{\text{Win}}(t,\ \alpha)\left(V_{\text{Win}}(t,\ \alpha) + t - \alpha\right), \\ V_{\text{Draw}}(t,\ \alpha) = \frac{\mathcal{N}(\alpha - t) - \mathcal{N}(\alpha - t)}{\Phi(\alpha - t) - \Phi(\alpha - t)}, \\ W_{\text{Draw}}(t,\ \alpha) = V_{\text{Draw}}^2 \frac{(\alpha - t) \cdot \mathcal{N}(\alpha - t) + (-\alpha + t) \cdot \mathcal{N}(-\alpha + t)}{\Phi(\alpha - t) - \Phi(\alpha - t)}. \end{gathered} \tag{11}$$

The functions V and W are given from the original TrueSkill paper [2] and is further explained there. The messages $m_{f_{\text{diff}} \to p_{\text{diff}}}(p_{\text{diff}})$ and $m_{f_{\text{out}} \to p_{\text{diff}}}(p_{\text{diff}})$ are then iterated over until the approximation of the marginal $p(p_{\text{diff}})$ converges. Now f_{diff} updates each player's performance variables with the messages

$$\begin{aligned} m_{f_{\text{diff}} \to p_{\text{winner}}}(p_{\text{winner}}) &= m_{\ p_{\text{loser}}\ \to f_{\text{diff}}}(p_{\text{loser}}) + m_{f_{\text{diff}} \to p_{\text{diff}}}(p_{\text{diff}}), \\ m_{f_{\text{diff}} \to p_{\text{loser}}\ }(p_{\text{loser}}) &= m_{p_{\text{winner}} \to f_{\text{diff}}}(p_{\text{winner}}) - m_{f_{\text{diff}} \to p_{\text{diff}}}(p_{\text{diff}}), \end{aligned}$$

after which the final messages from the performance factors for player i are sent to the skill variables as $m_{f_{p_i} \to s_i}(s_i) = m_{f_{\text{diff}} \to p_i}(p_i) + \mathcal{N}(0, \beta^2)$. Then the posterior skill is calculated as

$$p(s_i) = m_{f_{p_i} \to s_i}(s_i) \cdot m_{f_i \to s_i}(s_i),$$

and in the case of several played games as input data the posterior from the first game is used as a prior skill in the next game and so on, just as in the Gibbs sequential method.

5 Extensions

In an attempt at improving the vanilla TrueSkill model to make it better adapted to traditional sports, we made two extensions to the model: one considering playing with a side advantage and one taking score difference into consideration. Finally, these were combined into one single extension.

5.1 Home Team Advantage

Playing games at home ground has been proven advantageous in a variety of sports, including football; hence home team advantage (HTA) was considered a suitable aspect to adjust the TrueSkill model for. The approach described here could be used for any sort of side imbalance, such as white/black player in chess, where the white player generally has the advantage of starting the game, resulting in the white player winning around 1.35 times as many games as the black player over a data set of 910661 games [9]. To implement this occurrence for football we introduced two separate draw margins, ε_H for the home team and ε_A for the away team. They were constructed to replace the previously used single symmetrical $\pm\varepsilon$. By examining result data for a sport, the empirical probability for home win, draw, and away win can be calculated. Similar to the calculation of draw margins in basic TrueSkill, see (3) and (4), the probability of an outcome, although now shifted by respect to the home team advantage, is used to calculate the draw margins. In the general case the probability of a certain game outcome between two teams with the same skill, $s_1 = s_2$, with performance difference p_{diff} is given by

$$\begin{aligned} p(\text{HomeWin}) &= \int_{\varepsilon_H}^{\infty} \mathcal{N}(p_{\text{diff}}; 0, 2\beta^2)\ dp_{\text{diff}} = 1 - \Phi(\frac{\varepsilon_H}{\sqrt{2}\beta}), \\ p(\text{AwayWin}) &= \int_{-\infty}^{\varepsilon_A} \mathcal{N}(p_{\text{diff}}; 0, 2\beta^2)\ dp_{\text{diff}} = \Phi(\frac{\varepsilon_A}{\sqrt{2}\beta}), \\ p(\text{Draw}) &= \int_{\varepsilon_A}^{\varepsilon_H} \mathcal{N}(p_{\text{diff}}; 0, 2\beta^2)\ dp_{\text{diff}} = \Phi(\frac{\varepsilon_H}{\sqrt{2}\beta}) - \Phi(\frac{\varepsilon_A}{\sqrt{2}\beta}), \end{aligned} \tag{12}$$

from which the now separated draw margins are solved as

$$\varepsilon_H = \sqrt{2}\beta\Phi^{-1}\Big(1 - p(\text{HomeWin})\Big), \quad \varepsilon_A = \sqrt{2}\beta\Phi^{-1}\Big(p(\text{AwayWin})\Big).$$

Using these asymmetrical draw margins allows the extension to adjust the rating update from a more realistic prediction. We implemented the extension in the basic methods of TrueSkill with Gibbs sampling as well as message passing. The Gibbs method was simply extended by sampling the performance difference p_{diff} from these new margins, disregarding all sampled p_{diff}:s from outcomes other than the result. In message passing the message $m_{f_{\text{out}} \to p_{\text{diff}}}(p_{\text{diff}})$, from Equation (10), becomes as follows:

$$m_{f_{\text{out}} \to p_{\text{diff}}}(p_{\text{diff}}) = \mathcal{N}\left(\frac{\lambda + \sqrt{\pi} \cdot V(\lambda/\sqrt{\pi}, \varepsilon_H\sqrt{\pi}, \varepsilon_A\sqrt{\pi})}{1 - W(\lambda/\sqrt{\pi}, \varepsilon_H\sqrt{\pi}, \varepsilon_A\sqrt{\pi})}, \frac{\pi}{1 - W(\lambda/\sqrt{\pi}, \varepsilon_H\sqrt{\pi}, \varepsilon_A\sqrt{\pi})}\right),$$

where update equations V and W from Equations (11) are modified into:

$$V_{\text{HomeWin}}(t,\ \alpha_H,\ \alpha_A) = \frac{\mathcal{N}(t - \alpha_H)}{\Phi(t - \alpha_H)}, \quad W_{\text{HomeWin}}(t,\ \alpha_H,\ \alpha_A) = V_{\text{HomeWin}}(V_{\text{HomeWin}} + t - \alpha_H)$$

$$V_{\text{AwayWin}}(t,\ \alpha_H,\ \alpha_A) = \frac{\mathcal{N}(t + \alpha_A)}{\Phi(t + \alpha_A)}, \quad W_{\text{AwayWin}}(t,\ \alpha_H,\ \alpha_A) = V_{\text{AwayWin}}(V_{\text{AwayWin}} + t + \alpha_A)$$

$$V_{\text{Draw}}(t,\ \alpha_H,\ \alpha_A) = \frac{\mathcal{N}(\alpha_A - t) - \mathcal{N}(\alpha_H - t)}{\Phi(\alpha_H - t) - \Phi(\alpha_A - t)}$$

$$W_{\text{Draw}}(t,\ \alpha_H,\ \alpha_A) = V_{\text{Draw}}^2 \frac{(\alpha_H - t) \cdot \mathcal{N}(-\alpha_H - t) + (-\alpha_A + t) \cdot \mathcal{N}(\alpha_A + t)}{\Phi(\alpha_H - t) - \Phi(\alpha_A - t)}.$$

5.2 Score Difference

The score difference (SD) of a game outcome may indicate the skill ratio between two teams. Therefore, it is interesting to investigate the effect the score difference has on the facing teams' skill updates and if this makes the prediction model more accurate than when only considering win, loss or draw as possible outcomes when ranking teams. In order to do so, we once again extend the concept of a draw margin and implement it as score margins $\boldsymbol{\varepsilon} = [\varepsilon_1,\ \varepsilon_2,\ \ldots]$ instead. For a performance difference $p_{\text{diff}} \in [\varepsilon_2,\ \varepsilon_3]$, a two point win is expected, and for $p_{\text{diff}} \in [-\varepsilon_1,\ \varepsilon_1]$ a draw is expected and so on. These margins are visualized in Figure 7 for a p_{diff} with mean 0.

In this extension the possible outcomes are represented as the score difference of a game. For each score difference outcome $sd = |\textit{home team goals} - \textit{away team goals}|$ we determine the probability $p(sd)$ based on historical events by analyzing empirical data. For example, $p(sd = 2)$ is given by the empirical proportion of all games that ended up with a score difference of two.

Since training data for football might become sparse and unrepresentative for very large score differences we capped the score margins at ε_{5+}, thus all $p_{\text{diff}} \in [\varepsilon_{5+}, \infty]$ results in a 5 or more point win and were treated the same. The score margins take part in determining the skill updates of each team and were calculated through the probability in a likewise manner as the home team advantage extension by setting up the expression similar to Equation (12) but for the different score outcomes:

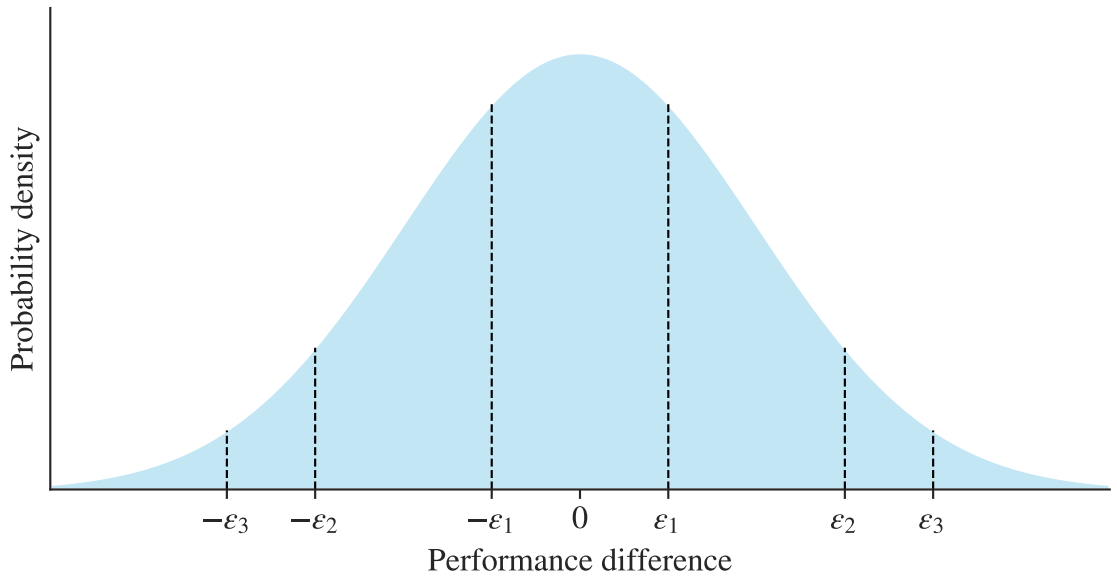

Figure 7: Arbitrary Gaussian distribution of a performance difference p_{diff} centered at 0 with score margins ε_i. The area under the curve between two score margins is the estimated probability of that outcome.

$$p(sd = i) = 2 \int_{\varepsilon_i}^{\varepsilon_{i+1}} \mathcal{N}(p_{\text{diff}}; 0, 2\beta^2)\ dp_{\text{diff}} = \int_{-\varepsilon_{i+1}}^{\varepsilon_{i+1}} \mathcal{N}(p_{\text{diff}}; 0, 2\beta^2)\ dp_{\text{diff}} - \sum_{\text{n}=0}^{i-1} p(sd = n)$$

where $i \in [1, 2, ...]$ and $\varepsilon_0 = 0$. Solving for ε_i gives the expression

$$\varepsilon_i = \sqrt{n}\beta\Phi^{-1}\left(\frac{1}{2} + \frac{1}{2}\sum_{sd=0}^{i-1} p(sd)\right).$$

For this extension, we redefined the message

$$m_{f_{\text{out}} \to p_{\text{diff}}}(p_{\text{diff}}) = \mathcal{N}\left(\frac{\lambda + \sqrt{\pi} \cdot V(\lambda/\sqrt{\pi}, \boldsymbol{\varepsilon}\sqrt{\pi}, sd)}{1 - W(\lambda/\sqrt{\pi}, \boldsymbol{\varepsilon}\sqrt{\pi}, sd)}, \frac{\pi}{1 - W(\lambda/\sqrt{\pi}, \boldsymbol{\varepsilon}\sqrt{\pi}, sd)}\right),$$

with V and W for a score difference sd as

$$V_{\text{Win}}(t,\ \boldsymbol{\alpha},\ sd) = \frac{\mathcal{N}(\alpha_{sd} - t) - \mathcal{N}(\alpha_{sd+1} - t)}{\Phi(\alpha_{sd+1} - t) - \Phi(\alpha_{sd} - t)},$$

$$V_{\text{Draw}}(t,\ \boldsymbol{\alpha}) = \frac{\mathcal{N}(\alpha_1 - t) - \mathcal{N}(-\alpha_1 - t)}{\Phi(\alpha_1 - t) - \Phi(-\alpha_1 - t)},$$

$$W_{\text{Win}}(t,\ \boldsymbol{\alpha},\ sd) = V_{\text{Win}}^2 \frac{(\alpha_{sd+1} - t) \cdot \mathcal{N}(\alpha_{sd+1} - t) + (\alpha_{sd} + t) \cdot \mathcal{N}(\alpha_{sd} + t)}{\Phi(\alpha_{sd+1} - t) - \Phi(\alpha_{sd} - t)},$$

$$W_{\text{Draw}}(t,\ \boldsymbol{\alpha}) = V_{\text{Draw}}^2 \frac{(\alpha_1 - t) \cdot \mathcal{N}(\alpha_1 - t) + (\alpha_1 + t) \cdot \mathcal{N}(\alpha_1 + t)}{\Phi(\alpha_1 - t) - \Phi(-\alpha_1 - t)}.$$

The Gibbs method was once again extended by sampling p_{diff} from the truncated Gaussians between the margins for the resulting score difference. It should be noted that while using this extension, a player's rating can decrease even if they win the game. That can happen if they were heavily favored to win the game, and were expected to win the game by a higher score than the resulting score, meaning they performed worse than expected and thereby lowered their rating.

5.3 Home Team Advantage and Score Difference

Lastly the home team advantage and score difference extensions were combined into a single extension (HTA+SD) to incorporate both of the individual extensions. For this $\boldsymbol{\varepsilon}$ needs to be expanded into $\boldsymbol{\varepsilon} = [\ldots, \varepsilon_{-2}, \varepsilon_{-1}, \varepsilon_1, \varepsilon_2, \ldots]$, creating separate margins for a home win and away win with the same absolute score difference as illustrated in Figure 8. Here a performance difference $p_{\text{diff}} \in [\varepsilon_{-3}, \varepsilon_{-2}]$ would correspond to a 2 point win for the away team. On the whole the method remains very similar to the score-based extension in implementation, except now keeping track of the home team and using the correct margins. Once again the margins were computed through the empirical probability for each scenario, but now treating home wins and away wins separately.

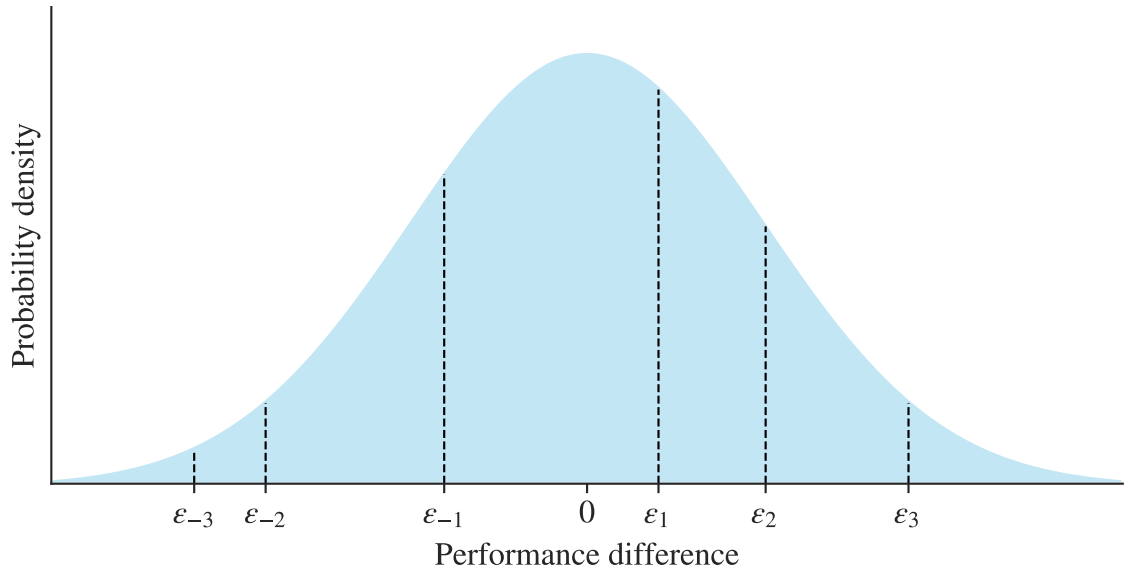

Figure 8: Arbitrary Gaussian distribution of a performance difference p_{diff} centered at 0, with score margins ε_i with respect to home team advantage. The area under the curve between two score margins is the estimated probability of that outcome.

In the message passing scheme, the message

$$m_{f_{\text{out}} \to p_{\text{diff}}}(p_{\text{diff}}) = \mathcal{N}\left(\frac{\lambda + \sqrt{\pi} \cdot V(\lambda/\sqrt{\pi}, \boldsymbol{\varepsilon}\sqrt{\pi}, sd)}{1 - W(\lambda/\sqrt{\pi}, \boldsymbol{\varepsilon}\sqrt{\pi}, sd)}, \frac{\pi}{1 - W(\lambda/\sqrt{\pi}, \boldsymbol{\varepsilon}\sqrt{\pi}, sd)}\right),$$

uses V and W for a non-absolute score difference $sd = \textit{home team goals} - \textit{away team goals}$

according to

$$V_{\text{HomeWin}}(t,\ \boldsymbol{\alpha},\ sd) = \frac{\mathcal{N}(\alpha_{sd} - t) - \mathcal{N}(\alpha_{sd+1} - t)}{\Phi(\alpha_{sd+1} - t) - \Phi(\alpha_{sd} - t)},$$

$$V_{\text{Draw}}(t,\ \boldsymbol{\alpha}) = \frac{\mathcal{N}(\alpha_{-1} - t) - \mathcal{N}(\alpha_{1} - t)}{\Phi(\alpha_{1} - t) - \Phi(\alpha_{-1} - t)},$$

$$V_{\text{AwayWin}}(t,\ \boldsymbol{\alpha},\ sd) = \frac{\mathcal{N}(\alpha_{sd-1} - t) - \mathcal{N}(\alpha_{sd} - t)}{\Phi(\alpha_{sd} - t) - \Phi(\alpha_{sd-1} - t)},$$

$$W_{\text{HomeWin}}(t,\ \boldsymbol{\alpha},\ sd) = V_{\text{Win}}^2 \frac{(\alpha_{sd+1} - t) \cdot \mathcal{N}(\alpha_{sd+1} - t) + (\alpha_{sd} + t) \cdot \mathcal{N}(\alpha_{sd} + t)}{\Phi(\alpha_{sd+1} - t) - \Phi(\alpha_{sd} - t)},$$

$$W_{\text{Draw}}(t,\ \boldsymbol{\alpha}) = V_{\text{Draw}}^2 \frac{(\alpha_{1} - t) \cdot \mathcal{N}(\alpha_{1} - t) + (-\alpha_{-1} + t) \cdot \mathcal{N}(-\alpha_{-1} + t)}{\Phi(\alpha_{1} - t) - \Phi(\alpha_{-1} - t)},$$

$$W_{\text{AwayWin}}(t,\ \boldsymbol{\alpha},\ sd) = V_{\text{Win}}^2 \frac{(\alpha_{sd} - t) \cdot \mathcal{N}(\alpha_{sd} - t) + (\alpha_{sd-1} + t) \cdot \mathcal{N}(\alpha_{sd-1} + t)}{\Phi(\alpha_{sd} - t) - \Phi(\alpha_{sd-1} - t)}.$$

Once again we extended the Gibbs method by sampling p_{diff} from the truncated Gaussians between the margins for the resulting score difference.

6 Evaluation

In our models a team is handled the same way as an individual player. This is due to the assumption that the team consists of the same players during the season, i.e. the composition of individual skills within the team are assumed to be consistent. While this seem to work for sports like football and ice hockey, this assumption would not be well suited for e.g. online games where players often and easily move between teams. In those cases it would be necessary to treat a team as an assemblage of individual skills.

For the Gibbs sampling method to perform well an appropriate number of samples from the beginning of the iteration process, the burn-in, needs to be discarded. A too long burn-in results in unnecessary use of computer performance. On the other hand, a too short choice might give inaccurate samples. To find a suitable burn-in, each sample's mean and standard deviation was examined with a trace-plot. Another parameter to be evaluated for the Gibbs method is the number of samples needed. To reach a good value, the results difference for a range of sample sizes was observed with a trace-plot, now with the burn-in included. The basic model is based on the assumption of a Gaussian distributed skill rating. To assess the propriety of this assumption for the Gibbs sampling method, the sampled rating is plotted together with the Gaussian curve fitting.

After having set these parameters and tested the underlying theoretical assumptions, the performance and accuracy of the extensions were tested by calculating both a prediction accuracy rate and the area under curve (AUC) of how the prediction accuracy rate changes with the number of games played for the applied data sets. The relationship between the predictive rate and the AUC tells us how quickly the model is trained on the data set. An AUC close to the predictive rate would suggest the model learns quickly, while a much lower AUC than the predictive rate would suggest it learns slowly and therefore needs to train on many games before giving somewhat good predictions. The predictions

are given by the largest likelihood of each outcome as seen in Equation (7). Only the outcomes are predicted, not the score differences even though it could be applied. The sequential methods predictions are done with the involved players' prior ratings before running inference on that game, while the batch method makes predictions by running a full batch inference on all games played prior the the game to predict.

6.1 Data Sets

To evaluate the vanilla TrueSkill model and extensions with different inference methods, they were applied on sport-result data sets. Data sets for football (Premiere League) [10], tennis (Association of Tennis Professionals (ATP) Tour) [11] and ice hockey (National Hockey League) [12] were used.

Football was chosen due to its amount of well sorted data, and because in a league all teams play the same amount of matches and face each other exactly two times, which is also why mainly football results were used, for easy comparison. Additionally, football is a familiar sport to many. The reason for also choosing to evaluate the ranking methods on a tennis tournament is because it includes individual players, not all players play the same amount of games and two players do not face each other an equal number of times. Also, there are more players in the ATP tour than in Premier League, which could affect the rating accuracy of the methods as well. As opposed to in football leagues, in tennis tournaments all matches do not have a player who is having any sort of home- or side advantage. Therefore two of the extensions, HTA and HTA+SD, were not considered applicable. Hence in order to examine all extensions on a sport besides football, ice hockey was chosen as well. Moreover, another interesting aspect of ice hockey is that there are no draws, unlike for football.

To calculate the required draw- or score margins in Premier League, seasons 2009/2010-2017/2018 of the same league were used as training data, calculating the empirical probabilities of all outcomes through that data. For ice hockey the regular seasons of NHL from 2012/2013 to 2016/2017 were used to calculate score and draw margins. Testing football was performed on the 2018/2019 season of Premier League, while testing ice hockey was performed on the regular season of NHL 2017/2018 and the tennis data set included ATP Tour games from 2018 only. By setting an initial skill rating to $s^0 = \mathcal{N}(\mu = 20, \sigma^2 = 25)$, the remaining model parameters were manually set to maximize the prediction accuracy for the training data set. For Premier League this resulted in a performance variance $\beta^2 = 25$ and a dynamic factor $\gamma^2 = 2/3$. For ice hockey the model parameters were set the same, not optimizing the parameters for NHL.

7 Results and Discussion

Although a wide range of results and observable data can be obtained from our methods, we have focused on the following aspects: graphs that motivate our Gibbs sampler design choices, performances for the different methods and extensions, and how the methods perform and adapt differently for varying sport-result data sets.

7.1 Gibbs Sampler Design

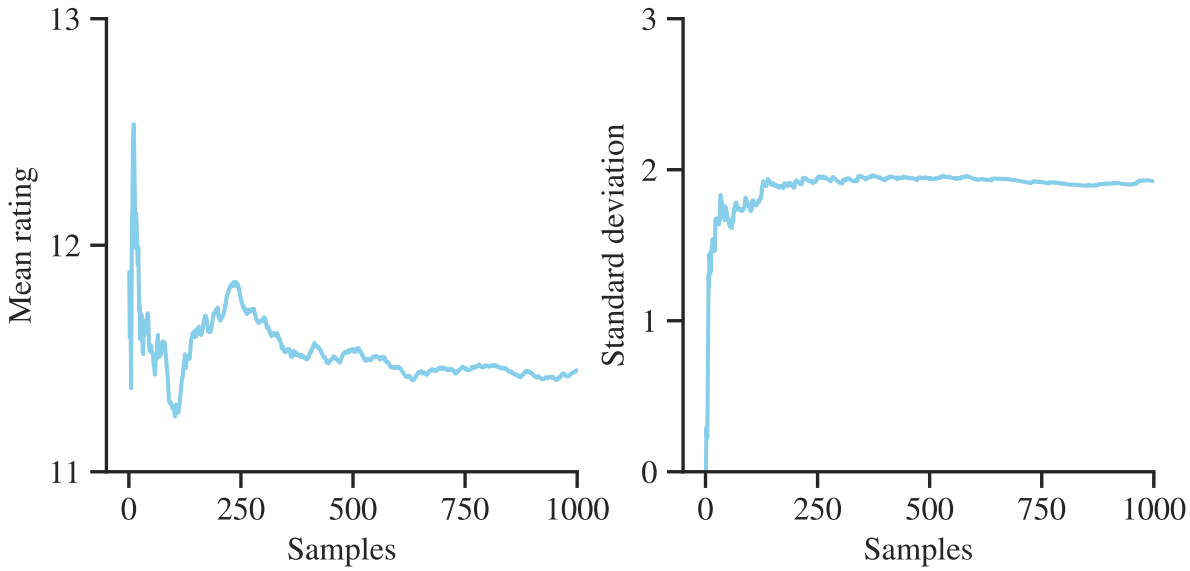

Figure 9: Trace plots of a rating sampled with Gibbs, illustrating the length of the burn-in phase before mean and standard deviation stabilizes

As shown in Figure 9, the trace plots of a rating sampled with Gibbs start to reach a steady state at around 500 samples, which is why it is our chosen burn-in for future results. By observing the shape of the rating distribution for a very high number of samples and an appropriate burn-in, see the fourth graph in Figure 10, it is clear that a Gaussian approximation is well suited since the graph converges to a Gaussian shape.

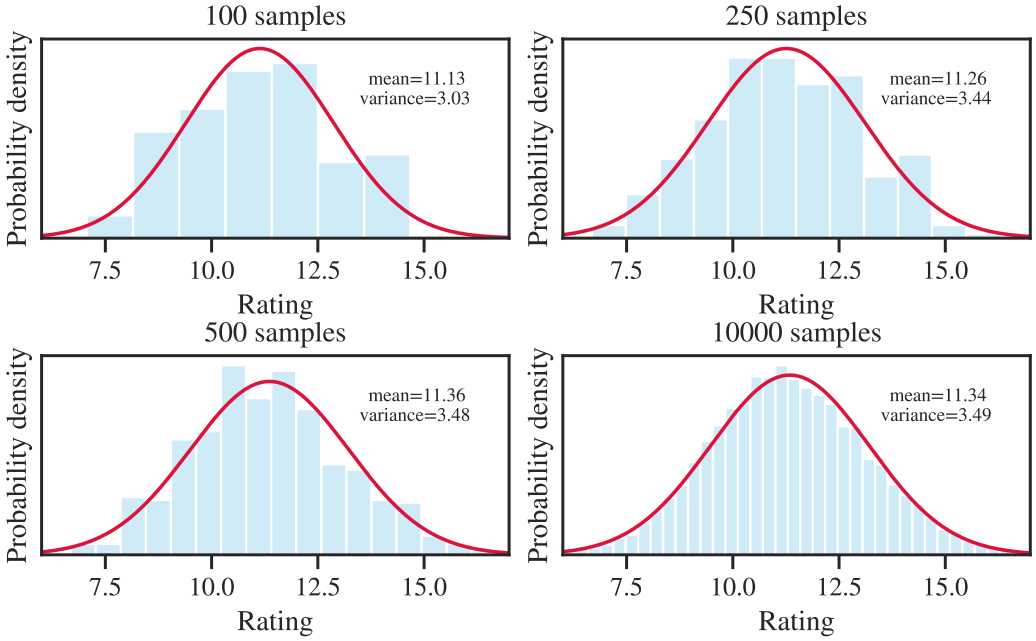

Figure 10: Histograms (blue) after using Gibbs sampling with 500 burn-in samples and 100, 250, 500 and 10000 samples after burn-in with an approximated distribution (red)

Then we have to decide how many samples to use after burn-in. From Figure 10 we see that after 250 samples the distribution has started to follow the approximate shape of a Gaussian distribution, which is what we desire. Therefore, a sample size of 250 or larger is to opt for.

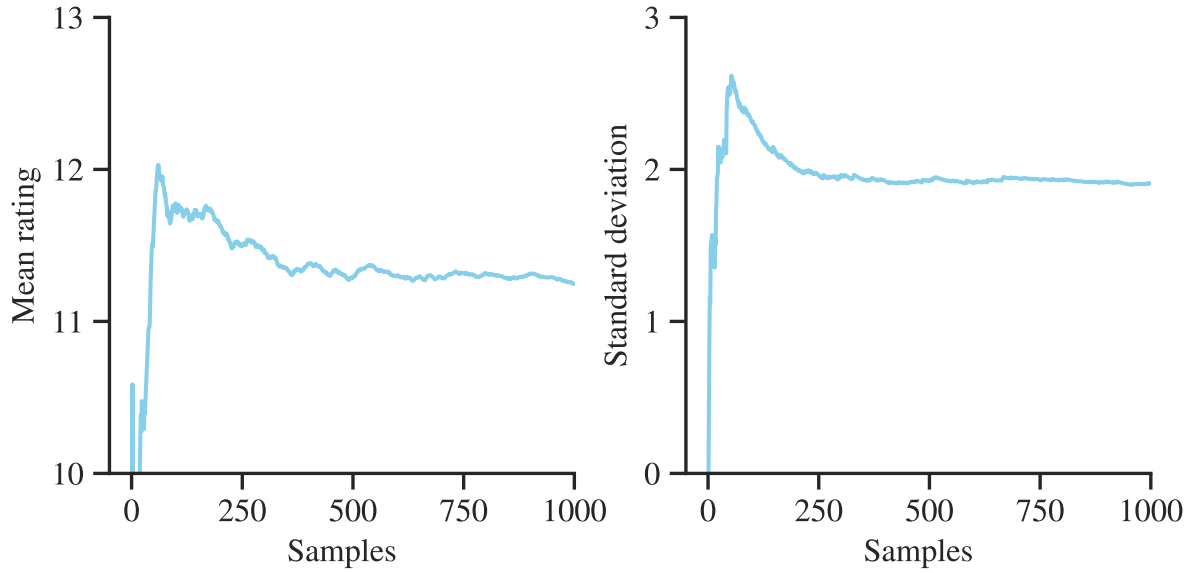

Figure 11: Trace plots of a rating sampled with Gibbs after a burn-in phase of 500 samples

Another trace plot, this time with the chosen burn-in 500, seen in Figure 11, we also see that after around 300 samples both the mean and the standard deviation has settled in at around its final values. Thus, we do not choose 250 samples but 300 samples instead as a minimum after burn-in to be an appropriate sample size to use when studying the Gibbs sampling methods' performance accuracy.

7.2 Applications on Different Data Sets

We constructed and tuned our methods mainly for football result data, thus it is optimized for that usage. When comparing performance results for different sorts of sport data sets, it is important to consider the sport-specific aspects affecting the predictions, such as number of possible outcomes. Tennis and ice hockey games result in a win or loss, whereas football has a third option in drawing. For example, a base case of always predicting home win in Premier League seasons 09/10 through 17/18 gives a correct prediction rate of 46.11%, whereas in tennis the base case would give 50% prediction rate.

7.2.1 Football

To measure the rating accuracy for each method we chose to calculate both the prediction rate and the AUC of the prediction accuracy as more games are played. We started by

testing all of our extension by applying them on the football data set for Premier League 18/19. The results are displayed in Table 1.

If we first compare just the extensions, the results show that applying HTA raises the AUC for all methods and the prediction rate for all but Gibbs batch, compared to basic vanilla. The same applies for the SD extension, with the message passing AUC as only exception. Even though the improvements when these extensions are applied are relatively small, it shows that the extensions can contribute to more accurate predictions. The best predictions were obtained by the combined extension HTA+SD, which results succeed the other methods in every aspect. With this extension we reached the highest prediction rate of 58.95%.

Looking at the methods, Gibbs sequential and message passing showed similar results. This was rather expected, since they are both sequential methods based on the same TrueSkill model. Worth mentioning is that since Gibbs sampling is a stochastic method, we can obtain slightly varying values for different runs, whereas the message passing methods gives deterministic results. If results were to vary a lot, it might suggest that too few samples are drawn in the Gibbs sampler each game, or that the burn-in is too short.

Using the batch method clearly affects the results. For vanilla, HTA and SD it is the superior method regarding both prediction rate and AUC. However, used for HTA+SD, it actually impairs the prediction accuracy as well as the AUC. Further, if we look in Table 1 we see that the Gibbs batch prediction rate is similar for vanilla, HTA and SD, they only differ from each other with 0.03 percentage points. This implies that the extensions are not informative enough to be applied for the batch method, and that they are of more importance for the sequential methods.

Table 1: All methods applied on Premier League 18/19 with corresponding prediction rate and area under curve of its prediction rate curve

Extensions	Method	Prediction rate	Area under curve
Vanilla	Gibbs sequential	55.79%	0.551
	Gibbs batch	57.37%	0.554
	Message passing	55.79%	0.543
HTA	Gibbs sequential	56.58%	0.557
	Gibbs batch	57.34%	0.575
	Message passing	56.84%	0.559
SD	Gibbs sequential	56.05%	0.548
	Gibbs batch	57.37%	0.569
	Message passing	56.05%	0.541
HTA + SD	Gibbs sequential	58.95%	0.585
	Gibbs batch	58.17%	0.576
	Message passing	58.95%	0.589

7.2.2 Ice Hockey

When testing our methods on ice hockey data, we set our draw margin to $\varepsilon = 0$, since an ice hockey game never ends with a draw. Intuitively, this should improve the prediction rate because there are only two possible outcomes for predictions. Nevertheless, the prediction rates, see Table 2, is lower than the similar results for Premier League, see Table 1, for most of the extensions. The only exception is HTA+SD, although such marginal differences is inexpedient to draw any general conclusions from.

Table 2: Methods applied on NHL (ice hockey) with corresponding prediction rate and area under curve of its prediction rate curve

Extensions	Method	Prediction rate	Area under curve
Vanilla	Gibbs sequential	56.18%	0.540
	Gibbs batch	57.83%	0.545
	Message passing	55.07%	0.527
HTA	Gibbs sequential	56.81%	0.557
	Gibbs batch	58.30%	0.567
	Message passing	56.96%	0.570
SD	Gibbs sequential	55.07%	0.529
	Gibbs batch	57.44%	0.538
	Message passing	56.73%	0.539
HTA + SD	Gibbs sequential	57.20%	0.542
	Gibbs batch	58.54%	0.550
	Message passing	57.51%	0.542

There are a couple of factors that make ice hockey harder to predict with TrueSkill compared to football. The games are shorter and players are replaced more often, hence the approximation of handling the whole team as one entity is not as suitable as for football. There is also the concept of powerplay, where one team is numerically superior temporarily, which could affect the outcome. This feature could hypothetically be considered and implemented into the model, which could potentially improve the accuracy.

Like in the Premier League results, the prediction rate is generally higher for Gibbs batch than for Gibbs sequential. Since NHL seasons also are over a relatively short period of time where the teams' performance is assumed to be quite consistent, it seems fair that neglecting the order of games improves the prediction rate. The accuracy in ice hockey is also higher for the extension HTA as well as HTA+SD, suggesting that home team advantage and impact of score difference is of importance.

7.2.3 Tennis

Tennis deviates significantly compared to the other sports we have applied our models to. Similar to ice hockey, we set our draw margin to $\varepsilon = 0$ since draw is not a possible outcome. Tennis is a single player sport, so we no longer have to make the approximation of a team as a player. Since there is not always a player with home turf in a tennis tournament, the HTA model was not considered applicable for this set.

We noticed that an issue when predicting tennis tournament results is that the number of games that each player play varies a lot. For the players that only get to play a couple of games, their skill ratings are not updated enough to be predicted fairly. To avoid this, each player who played 5 number of games or less was removed from the set of predictions. This refining, hereon called Drop 5, improved the prediction rate significantly for Gibbs sequential as well as message passing, see Table 3. For Gibbs batch the improvement was less significant but still noticeable. Something we believe lowers the prediction accuracy, even with Drop 5, is the fact that tournament participants play a different number of games compared to each other. This creates an imbalance when the players are compared to each other, since the models' uncertainty differs between players. A way to develop the TrueSkill method for tennis could be to adapt and apply the SD extension for tennis.

Table 3: Methods applied on ATP (tennis) with corresponding prediction rate and area under curve of its prediction rate curve

	All games		Drop 5	
Method	Prediction rate	AUC	Prediction rate	AUC
Gibbs sequential	56.82%	0.527	58.76%	0.533
Gibbs batch	59.62%	0.555	60.14%	0.557
Message passing	56.05%	0.544	59.57%	0.548

7.3 Home Team Advantage Extension

As a confirmation of the HTA extension working as intended we studied the updated skill ratings of Southampton and Burnley in the Premier League who drew their game in the first round of play, meaning they had the same initial rating s^0 before the game. Without the extension the game results in both teams getting identical posterior ratings. However, in Figure 12 we see that using the HTA extension with message passing inference, the skill s_2 of Burnley (red) is slightly higher than the s_1 of Southampton (blue) after the game. This is due to Southampton having the home team advantage, meaning they are somewhat more expected to win than Burnley although having the same prior. Consequently, the method favors Burnley who is rewarded with a higher rating without having won the game.

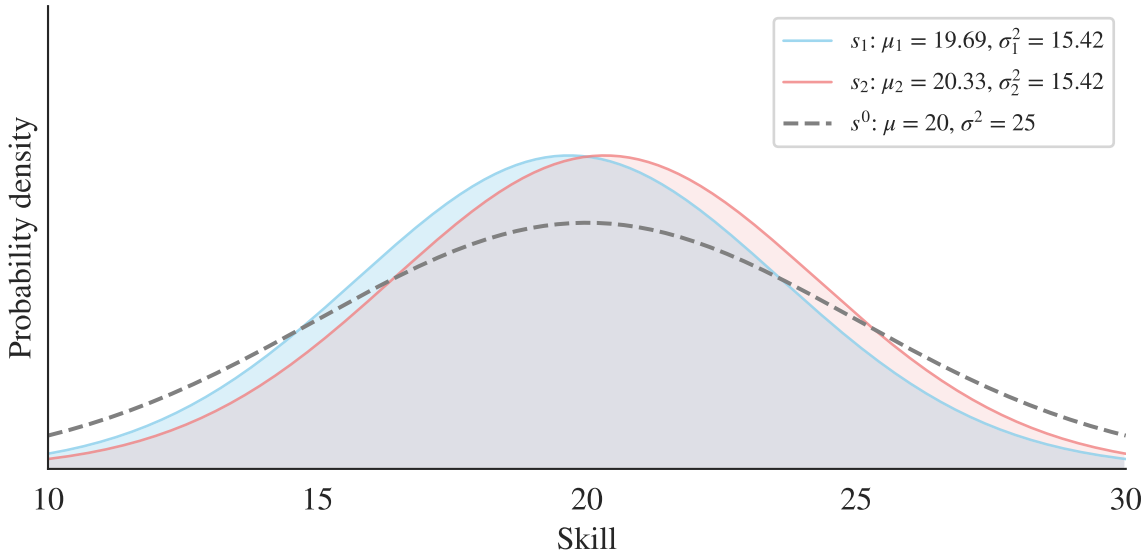

Figure 12: The home advantage model skill update after the home team Southampton (blue) with skill s_1 draws with the away team Burnley (red) with skill s_2, both with same prior s^0 (black)

From the results in Table 1 we see that using the HTA extension results in a somewhat higher accuracy in predicting the correct game outcome of games in the Premier League 18/19 than the original model, although by a quite small margin, suggesting that the extension works. Its improvement over the vanilla model might be somewhat limited since in the Premier League each team faces each other team twice, once at home and once away, possibly making it less important to differentiate between the two sides. It might also be faulty to assume that each team are affected the same when playing at home or away, and it could be interesting to assign individual margins for different teams.

7.4 Score Difference Extension

To confirm the impact of the SD extension we look at two other games from the first round of play: Liverpool versus West Ham where Liverpool won with 4-0, and Manchester United versus Leicester where Manchester United won with 2-1. With all teams having the same prior skill rating s^0, the updated skill ratings after inference with message passing can be seen in Figure 13. We note that the update in skill is larger when the score difference is larger, suggesting the extension works as intended.

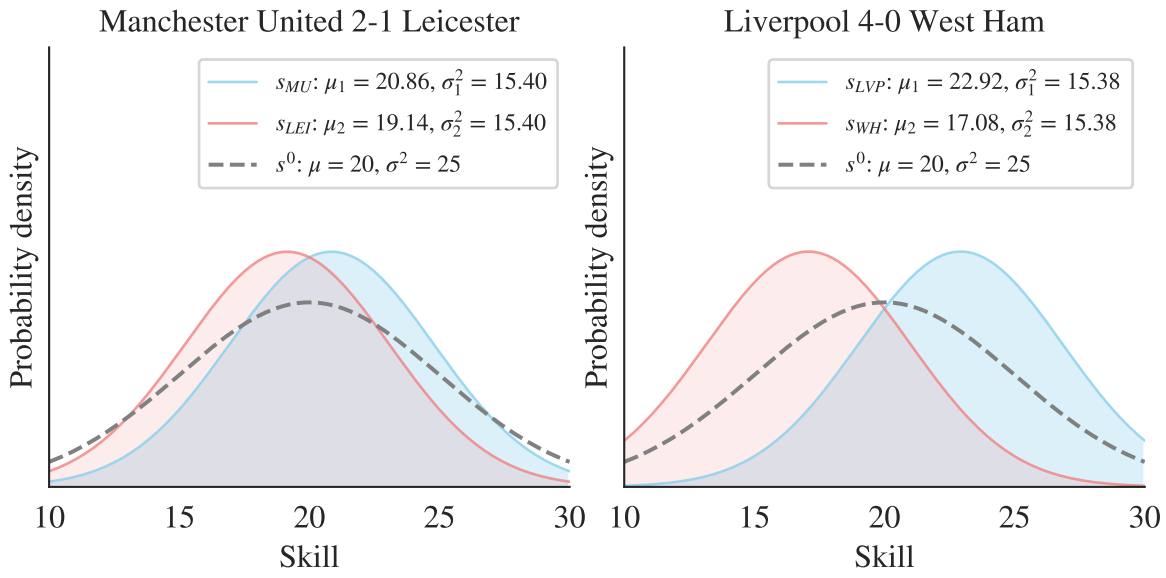

Figure 13: The score-based skill updates from the same prior skill s^0 (black) after Manchester United (blue) with skill s_{MU} wins with 2-1 against Leicester (red) with rating s_{LEI} (left), and Liverpool (blue) with skill s_{LVP} wins against West Ham (red) with skill s_{WH} with 4-0 (right)

Once again looking at Table 1, we see that the SD extension performs better than vanilla TrueSkill using all inference methods on Premier League, although it performs worse than the HTA extension.

7.5 Discussion of Inference Methods

If we first consider just Gibbs sequential compared to message passing, a noticeable difference between the methods is the run time, where the deterministic message passing algorithm is faster than the stochastic Gibbs sampling. According to Tables 1, 2, and 3, the accuracy and AUC metrics are generally very similar for Gibbs sequential and message passing, which is expected since they follow the same model. Because of this we consider message passing to be more efficient in the sequential approach and thence the one we would recommend in the general case.

Taking the Gibbs batch method into consideration, we notice that the batch method has a higher performance than Gibbs sequential in most extensions for football, tennis and ice hockey, suggesting that neglecting the match order has a positive impact on the results. Regardless of the resulting prediction accuracy, one aspect where the batch method definitely succeeds is in being fair to players/teams. It does not give unearned advantage to chronologically early wins. Thus, we consider the batch model fair to the game and the players, something that is very important in many sports. It is also very fitting for the seasonal approach in Premier League, NHL and most other leagues, where a batch inference can be run on one season at the time, giving each team the same initial rating at the start of the season. In leagues of this sort, where a closed set of teams faces one another a set amount of times per season, teams are traditionally ranked through

awarding a fixed number of points for a win or draw. In those cases, a batch inference model could replace the traditional ranking table without changing the structure of the league while still giving a fair ranking. A drawback though is that it needs to be fully run between each new added game. This makes it unsuitable for large scale applications such as online games or other data sets where new results are being added rapidly, compared to a sequential inference where a new result is easily added without running inference on earlier games.

7.6 Further Improvements and Applications

Our results have shown that rating using the TrueSkill model has room for further sport-related extensions, such as considering injuries, play time etc. The score-based method could be altered to handle other result differences than number of goals, e.g. time differences in sports where you win by reaching a finish line, where we could use a continuous value as outcome. Another possible improvement is to implement a weighted batch method, such that more recent matches have a bigger impact on the skill rating, while still reducing most of the order dependency of matches.

In the HTA extension, an individual draw margin could possibly be implemented, since not all teams perform equally well at home field. Another future suggestion is to test on non-closed football data sets, such as international leagues where teams from different leagues play against each other, as well. Furthermore, it could be considered how skills carry over between different modes. For example, how much a tennis player's individual skill in one versus one matches to a team tennis game of two versus two, or the correlation of a cross-country skier's skill in sprint to its skill in a long distance run, could be studied and implemented. If more features are added resulting in an overall higher accuracy, it might be interesting to compare the model predictions to sport betting odds, analyzing whether the model might do better predictions than a human bettor.

In our methods we searched for one set of model parameters, β^2 and γ^2, that gave the highest prediction accuracy overall for all our extensions for football. When using one specific extension for a known data set, these parameters could be individually tuned, which potentially could improve the accuracy. As the same model parameters were used in all sports for easier comparison, tuning the model parameters individually for ice hockey and tennis could also improve the accuracy.

8 Conclusions

The ranking methods were constructed to be suited for football. It worked on tennis and ice hockey as well, but performed best for football, taking the different conditions of the sports into account, e.g. the possibility of a draw in football. This performance superiority could be due to a few reasons, e.g. that the parameters have been optimized while testing on Premier League data, that home team advantage is a more prevalent phenomenon and that the score difference has a higher correlation to the skill within football.

The run time is significantly different between the three methods, factor graph being the fastest. For Gibbs sampling, the running time of a full batch and the sequential method is similar. Since the former has to run a full batch inference on every game played prior to the one which is to be predicted, while the latter only has to consider the two concerned teams' rating prior to that game, a sequential method would be to prefer in large scale applications. However, the batch method surpasses the sequential in prediction accuracy for almost all extensions. The batch method probably performs better because the match order dependency has been eliminated.

The extensions generally improved the performance of the vanilla methods on all data sets applied to. The home team advantage method performs better on both football and ice hockey than the score difference one does, and the combination of the two performs even better.

TrueSkill is already known to work well for online games, as shown in the original TrueSkill paper [2] and the improved TrueSkill 2 [4]. Based on the results of this study, it could be considered a useful ranking model for other sports as well, especially if the vanilla model is tuned and adjusted with extensions for the sport it is to be applied on.

References

[1] Elo Arpad. The rating of chessplayers, past and present. *Arco Pub*, 1978.

[2] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill$^{\text{TM}}$: a bayesian skill rating system. In *Advances in neural information processing systems*, pages 569–576, 2007.

[3] Professor Carl Edward Rasmussen. Lecture notes in 4f13: Probabilistic machine learning, October 2018.

[4] Tom Minka, Ryan Cleven, and Yordan Zaykov. Trueskill 2: An improved bayesian skill rating system. *Tech. Rep.*, 2018.

[5] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, New York, 2006.

[6] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, Cambridge, England, 2012.

[7] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, Cambridge, Massachusetts, 2012.

[8] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[9] Chessgames.com. Chess Statistics. Available at: http://www.chessgames.com/chessstats.html, 2019. [Accessed 24 May 2019].

[10] Football-Data.co.uk. English Premier League (football). Available at: https://datahub.io/sports-data/english-premier-league, 2019. [Accessed 20 May 2019].

[11] Jeff Sackmann. ATP Tennis Rankings, Results, and Stats. Available at: https://github.com/JeffSackmann/tennis_atp. [Accessed 20 May 2019].

[12] Hockey Reference. 2017-18 NHL Schedule and Results. Available at: https://www.hockey-reference.com/leagues/NHL_2018_games.html, 2019. [Accessed 20 May 2019].