Probabilistic Machine Learning - TrueSkill ranking system using Bayesian inference and factor graphs

Erik Turesson
Uppsala University

Joel Bjervig
Uppsala University

Nils Gumaelius
Uppsala University

1 Introduction

The Trueskill model is used to estimate different player's skills based on historical results. The original application was to match players with similar skill levels in an online game such that the game is fair and thus enjoyable. However, it's use has later been expanded into other games, such as chess and football. It has proven successful in predicting the outcome of games based on the player's skills. In this report, we apply Bayesian inference, Gibbs sampler, and factor graph theory to calculate the skills in the 17-18 season of men's football in Serie A in order to predict the result of each game before it is played. The model performed as expected with a prediction rate of about 65%. Another data set of the 2016 season of MLB showed a prediction rate of 53%. The model was extended such that the goal difference was taken into account, which improved the model to around 67% and 53% respectively.

1.1 The Trueskill model

In the Trueskill model [1], we consider the estimated skill s of player i as $s_i \sim \mathcal{N}(s_i;\ \mu_i,\ \sigma_i^2)$. In a 1-versus-1 game with player 1 and 2, we denote $\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 \end{bmatrix}^T$, for which the two dimensional probability function is

$$p(s_1,\ s_2) = p(\boldsymbol{s}) = \mathcal{N}(\boldsymbol{s};\ \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix},\ \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}). \tag{1}$$

Between these players, we consider the performance difference t:

$$p(t|s_1,\ s_2) = \mathcal{N}(t;\ s_1 - s_2,\ \beta^2) \text{ or } p(t|\boldsymbol{s}) = \mathcal{N}(t;\ \begin{bmatrix} 1 & -1 \end{bmatrix} \boldsymbol{s},\ \beta^2) \tag{2}$$

where the added variance β^2 accounts for that a player will not perform at the same level each game. Further, in this project we discard draws and assume that the player who performs the best will always win. Thus, we introduce a game result $y = sign(t)$, meaning: $p(y = 1) = p(t > 0)$ and $p(y = -1) = p(t < 0)$

Given y depends on t, and t depends on s, the Trueskill model thus has a joint distribution of

$$p(\boldsymbol{s}, t, y) = p(y|t)p(t|\boldsymbol{s})p(\boldsymbol{s}) \tag{3}$$

The model calculates the posterior skill distributions using Bayesian inference after having observed a match result y:

$$p(\boldsymbol{s}|y, t) = \frac{p(y, t|\boldsymbol{s})p(\boldsymbol{s})}{p(y, t)} \tag{4}$$

2 Inference in the Trueskill model

For later use, we want to prove that s is conditionally independent of y, given t: $\boldsymbol{s} \perp\!\!\!\perp y|t$. Using Bayes' theorem and the probabilistic chain rule, we write:

$$p(\boldsymbol{s}, y|t) = \frac{p(\boldsymbol{s}, y, t)}{p(t)} = \frac{p(y|t)p(t|\boldsymbol{s})p(\boldsymbol{s})}{p(t)} = \frac{p(\boldsymbol{s}|t)p(t)p(y|t)}{p(t)} = p(\boldsymbol{s}|t)p(y|t) \tag{5}$$

which is the definition of conditional independence, hence $s \perp\!\!\!\perp y|t$

From this result it follows that $p(s_1,\ s_2|t,\ y) = p(s_1,\ s_2|t)$. From Bayes' theorem we know $p(s_1,\ s_2|t) \propto p(t|s_1,\ s_2)p(s_1,\ s_2)$. With the distributions defined for the Trueskill model in equation 2, $p(s_1,\ s_2|t)$ can be calculated as [2, p. 202]

$$p(\boldsymbol{s}|t) = \mathcal{N}(\boldsymbol{s};\ \boldsymbol{\mu}_{\boldsymbol{s}|t},\ \Sigma_{\boldsymbol{s}|t})$$

with

$$\boldsymbol{\mu}_{\boldsymbol{s}|t} = \Sigma_{\boldsymbol{s}|t}\left(\begin{bmatrix} \frac{\mu_1}{\sigma_1^2} + \frac{t}{\beta^2} \\ \frac{\mu_2}{\sigma_2^2} - \frac{t}{\beta^2} \end{bmatrix}\right),\ \text{and}\ \Sigma_{\boldsymbol{s}|t} = \begin{bmatrix} \frac{1}{\sigma_1^2} + \frac{1}{\beta^2} & -\frac{1}{\beta^2} \\ -\frac{1}{\beta^2} & \frac{1}{\sigma_2^2} + \frac{1}{\beta^2} \end{bmatrix}^{-1} \tag{6}$$

Given equation 2, result y, and priors s, and the fact that $p(t|y) = \delta(y - sign(t))$, we get:

$$p(t|y,\ \ s_1,\ \ s_2)\ \ \propto\ \ p(y|t)p(t|s_1,\ \ s_2)\ \ =\ \ \begin{cases} \mathcal{N}(t;\ s_1 - s_2,\ \beta^2), \text{when } sign(t) = y \\ 0, \text{otherwise} \end{cases} \tag{7}$$

This is a truncated Gaussian, meaning we need to perform non-exact inference to keep posterior distributions as Gaussians.

2.1 Likelihood of result

We can also calculate the prior probability of player 1 winning the game: $p(y = 1) = p(t > 0) = 1 - p(t < 0) = 1 - \Phi(\frac{-\mu_1 + \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2 + \beta^2}})$ where Φ is the cumulative distribution function of the standard gaussian ($\mathcal{N}(0, 1^2)$). Likewise, $p(y = -1) = \Phi(\frac{-\mu_1 + \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2 + \beta^2}})$. This can be used for predicting the outcome of a game, picking the player with the higest probability to win as the favorite to win the game. Note that this will simply be the player with the highest prior mean.

The Bayesian model can be used for predicting the games. The proposed prediction function follows the argument above and can be seen in equation 8 where μ_1 and μ_2 are the means of the two players before the game. The function returns +1 if player 1 is most likely to win, and -1 for player 2.

$$prediction = 2 * bool(\mu_i^a > \mu_i^b) - 1 \tag{8}$$

2.2 Bayesian Network

To further verify the conditional independence derived in 5 for the Trueskill model, we can utilize Bayesian network theory. Consider the following Bayesian network in 1. Given the head to-tail rule, independence follows if the in-between node of two nodes is observed and "blocks" the path.[3, p.373] In this case an observed t will block the only path between s and y, thus it shows $s \perp\!\!\!\perp y|t$.

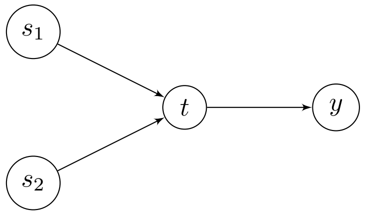


Figure 1: Bayesian network model of the Trueskill model

2.3 Gibbs sampler

Gibbs sampler is a Markov chain Monte Carlo algorithm. It is used when it's computationally difficult to sample from the joint distribution directly. Instead, it samples with conditional distributions, which presents its main weakness, that all conditional probabilities must be known.

The first step in the Gibbs algorithm is to choose an initial guess (S_1^0, S_2^0). Thereafter a random sample of t is drawn, conditioned on the initial guesses, given by $t \sim \mathcal{N}(t;\ S_1^0 - S_2^0,\ \beta^2)$. From that sampled variable, we draw a new $S \sim \mathcal{N}(S;\ \mu_{S|t},\ \Sigma_{S|t})$, of The conditional mean and variance are defined at 6. By drawing one variable conditioned on the other, and alternating this procedure between all random variables, the path the algorithm takes in attempt to approximate the joint distribution cannot be "diagonal" since it only moves in one direction for each step. The initial samples may not be representative of the final distribution since they will be correlated with the initial guess that may be far from the centre of the joint distribution. The samples within this so-called burn in period is usually discarded. [4]. A trick is to use the mean of the random variables as the initial guess $(S_1^0, S_2^0) = (\mu, \mu_2)$.

In Figure 2, Gibbs samples the skills s of each player for a game where $s_1 \sim \mathcal{N}(s_1;\ 12,\ 5)$, $s_2 \sim \mathcal{N}(s_2;\ 10,\ 2.5)$ when team 1 wins are shown. Here, it is seen that the burn in period seems to be quite small, and the samples seem to lose correlation from the initial samples after around sample 50. To be on the safe side, we choose a burn in of 100 samples going forward. This choice produces consistent results across several re-runs. It might be the case that it would be less fit for a different initial guess.

Algorithm 1: Gibbs Sampling in the Trueskill model

```
Assuming $B$ burn in samples, $M$
$s_1^{-B-1} = \mu_1$
$s_2^{-B-1} = \mu_2$
$\Sigma_{\mathbf{s}|t} = \begin{bmatrix} \frac{1}{\sigma_1^2} + \frac{1}{\beta^2} & -\frac{1}{\beta^2} \\ -\frac{1}{\beta^2} & \frac{1}{\sigma_2^2} + \frac{1}{\beta^2} \end{bmatrix}^{-1}$
for $i = -B, \dots,\ -1,\ 0,\ 1,\ \dots,\ M - 1$ do
    $t^i$ = sample from $\mathcal{N}(t^i;\ s_1^{i-1} - s_2^{i-1},\ \beta)$
    $\mu_{\mathbf{s}|t}^i = \Sigma_{\mathbf{s}|t} \begin{bmatrix} \frac{\mu_1}{\sigma_1^2} + \frac{t}{\beta^2} \\ \frac{\mu_2}{\sigma_2^2} - \frac{t}{\beta^2} \end{bmatrix}$
    $s_1^i, s_2^i$ = sample from $\mathcal{N}(\mathbf{s^i};\ {}^{-\mathbf{i}}_{\mathbf{s}|\mathbf{t}}{}_{\mathbf{s}|t},\ \Sigma_{\mathbf{s}|t})$
    if $i \geq 0$ then
        save $s_1^i, s_2^i$
    end
end
```

Because of the short burn in, one might wonder what effect removing this period has. Figure 3 shows two histograms from running the Gibbs algorithm first with a burn in of a 100 samples and one run with no burn in at all. When the first 100 samples are discarded there is indeed an improvement as the histogram seems to follow a gaussian curve more precisely than if we were to include the first 100 samples (no burn in).

A reasonable number of samples also need to be drawn for each game after the burn in. To decide this, we look at histograms for different sample counts in Figure 4, and say that 450 samples or more seem to indicate a well-sampled Gaussian distribution, which is used going forward.

2.4 Factor Graph with Message Passing

To determine $P(t|y_{obs})$ we construct the factor graph in Figure 5.

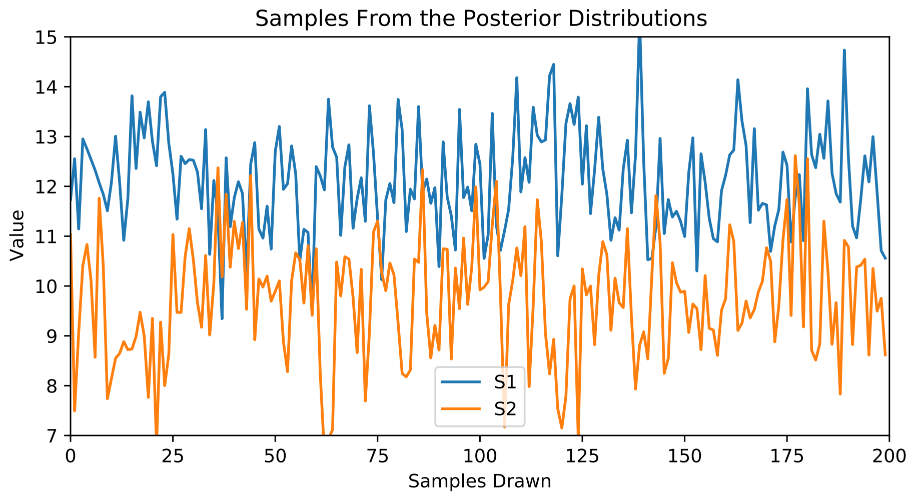


Figure 2: The posterior Gibbs samples for $s_1 \sim \mathcal{N}(s_1;\ 12,\ 5)$, $s_2 \sim \mathcal{N}(s_2;\ 10,\ 2.5)$ when team 1 wins

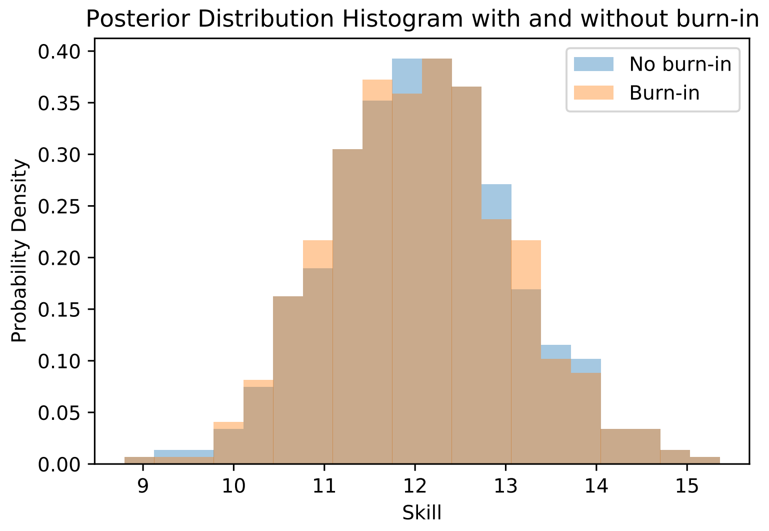


Figure 3: The sample histograms for the skills in the simulation in Figure 2, with and without burn in and for 450 samples

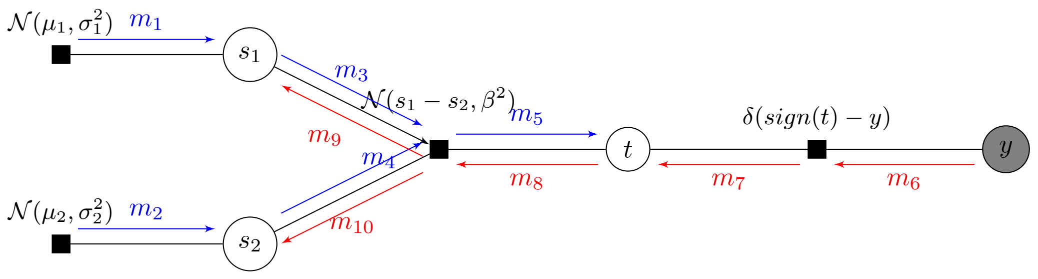

$\mathcal{N}(\mu_1, \sigma_1^2)$
m_1
s_1
m_3
$\mathcal{N}(s_1 - s_2, \beta^2)$
m_5
$\delta(sign(t) - y)$
m_9
m_4
t
y
m_8
m_7
m_6
$\mathcal{N}(\mu_2, \sigma_2^2)$
m_2
m_{10}
s_2

Figure 5: Factor graph with message passing for a 2 player match with observed outcome y

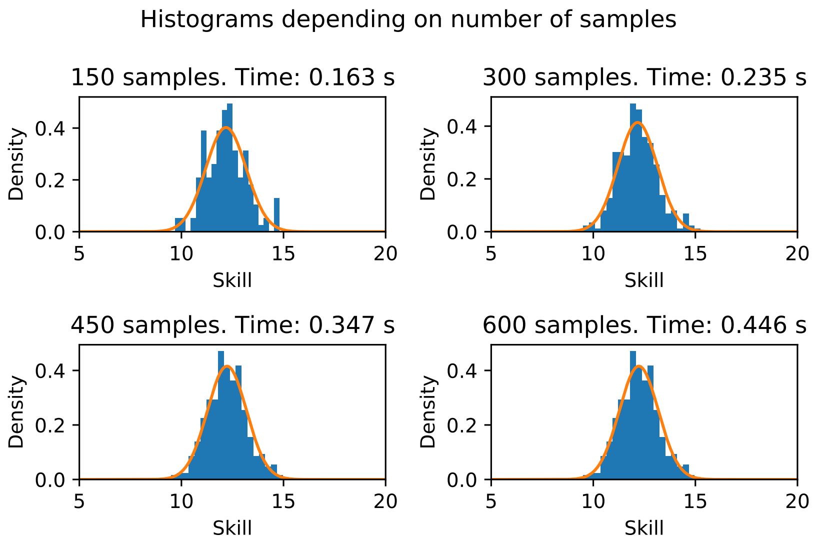


Figure 4: Histograms (blue) for a player's posterior skill rating after 150, 300, 450, and 600 samples have been drawn using Gibbs sampler, and a fitted gaussian probability density function (orange) for each histogram

2.5 Message passing with moment matching

$P(t|y_{obs})$ can be calculated in node t as a product of all incoming messages defined below, resulting in a truncated Gaussian $\delta(t > 0)\mathcal{N}(t;\ m_1 - m_2,\ \sigma_1^2 + \sigma_2^2 + \beta^2)$ as previously seen in section 2. If a complicated distribution arises in the factor graph, it will make the integrals cumbersome. For this reason, one may apply moment matching which approximates the distribution function with a regular Gaussian. In Figure 5, $P(t|y)$ is a truncated Gaussian, and therefore we approximate it as $P(t|y) \approx \hat{q}(t) = \mathcal{N}(\mu_t,\ \sigma_t)$.

$$m_1 = \mathcal{N}(\mu_1,\ \sigma_1^2) \qquad m_2 = \mathcal{N}(\mu_2,\ \sigma_2^2) \tag{9}$$

$$m_3 = m_1 \qquad m_4 = m_2 \tag{10}$$

$$m_5 = \mathcal{N}(\mu_1 - \mu_2,\ \sigma_1^2 + \sigma_2^2 + \beta^2) \qquad m_6 = \delta(y - y_{obs}) \tag{11}$$

$$m_7 = \delta(sign(t) - y_{obs}) \qquad m_8 = \frac{\hat{q}(t)}{m_5} \tag{12}$$

$$m_9 = \int \mathcal{N}(s_1 - s_2,\ \beta^2) m_8 m_4\ dt\ ds_2 = \mathcal{N}(\mu_8 + \mu_4,\ \sigma_8^2 + \sigma_4^2 + \beta^2) \tag{13}$$

$$m_{10} = \int \mathcal{N}(s_1 - s_2,\ \beta^2) m_8 m_3\ dt\ ds_1 = \mathcal{N}(\mu_3 - \mu_8,\ \sigma_8^2 + \sigma_3^2 + \beta^2) \tag{14}$$

μ_t and σ_t^2 are found using `scipy.stats.truncnorm.stats` which calculates the mean and variance of a given truncated gaussian. The remaining messages m_i with respective means μ_i and variances σ_i^2 are calculated using affine transformations [2, p. 202] and Gaussian multiplications and divisions. Message 9 and 10 are calculated as the product of all incoming messages and the connected factor $\mathcal{N}(s_1 - s_2,\ \beta^2)$, marginalized over t and $s_{1,2}$. After each game, the posterior distributions will be $s_1^{post} = m_1 m_9$, $s_2^{post} = m_2 m_{10}$.

In Figure 6 the posterior distributions of a simulated game between two players using Gibbs and message passing is seen to have very similar resulting distributions.

2.6 Assumed Density Filtering

To apply the model over a series of games, the posterior skill distribution after a game is used as the prior distribution to the upcoming game with the same player. This approach is called Assumed Density Filtering (ADF) and can be used to update the posterior distribution over many iterations.

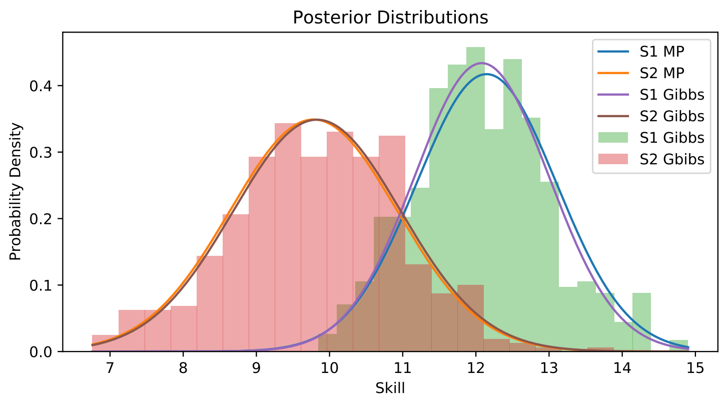


Figure 6: Posterior distributions or two players using Gibbs sampling and message passing

From the method, we get a distribution for each player with the mean representing the players' skill levels and the variance can be interpreted as how certain the model is on the players' skill levels.

However, the order the games are played in will affect the final result as the variance of estimates decreases. To showcase this effect, we later perform ADF with both the chornological and a shuffled game order.

2.7 Testing a new dataset - MLB Baseball

Even though the Trueskill ELO system is first made for football it can be used as a ranking system for many different sports. Different sports with different scoring systems and different playing formats challenges the model in different ways. We chose to implement a dataset from the 2016 season of MLB, the American baseball league. As a sport, baseball is more random in its results and the final table is much more even compared to football. Thus making it interesting to test the performance of our model.

3 Score Difference Extension

The no-draw vanilla Trueskill model treats results as binary, but often the result can tell us more than that. A win with 4 points is indicative of a larger performance difference than a win with a single point. Thus, the resulting update should be larger as well. Our solution to this is to define some performance margins $\epsilon_i,\ i \in [0, n],\ \epsilon_0 = 0$, where n is the limit at which we stop to consider the increasing score difference. These margins define new boundaries to the performance difference such that a score $y = i > 0$ means $\epsilon_{i-1} < t < \epsilon_i$, and $y = i < 0$ means $-\epsilon_{-i} < t < -\epsilon_{-i-1}$. The values of margins are calculated from the empirical probability of a given score difference, such that with Φ being the cumulative distribution fucntion of a standard gaussian we have:

$$p(|y| \leq i) = 2\Phi\left(\frac{\epsilon_i}{\beta}\right) - 1$$

This formula is based on the calculation of the draw margin in the original model [1], and our extension could easily model draws by just not defining $\epsilon_0 = 0$. The only difference this makes to the Trueskill model is to the limits of the truncated gaussian, thus we need only to change the sampling and messages from this distribution.

In this project, we condition the values of ϵ_i on the same data we test it on. In reality, one should use data from a larger data set, e.g. from multiple seasons and leagues (within the same sport).

4 Results

4.1 Serie A dataset

To test the effectiveness of the ADF implementations over a set of games, both the Gibbs sampling and the message passing models were applied on the Serie A (football) 17/18 and the MLB (baseball)

2016 seasons, with and without the score difference extensions. Each team is modelled to be a player s_i. Before any games are played, all teams share a default rating of $s \sim \mathcal{N}(s;\ \mu = 10,\ \sigma^2 = 5)$. Further, we use $\beta^2 = 2$.

The final team rankings of the Serie A being processed chronologically can be seen in table 1.a). Also, in table 1.b) the result for a shuffled game order can be seen. Comparing the two tables, we can see that the shuffling of the matches changed the final order quite a lot. This is due to the earlier games having a bigger impact than the later games. Since the variance gets smaller for each update, the distribution will change less, and thus playing well in the games processed earlier will have a bigger impact than playing well in the later ones.

Iterated over the full data set, predictions were made before each game. The prediction rate $r = \frac{\text{number of correct guesses}}{\text{number of total guesses}}$ was calculated to $r = 0.654$. This shows that the model performs better than random guessing (50% since we disregard draws).

Table 1: Final Serie A 2017/2018 standings with Gibbs ADF

(a) Chronological game order

Team	Mean	Variance
Juventus	12.313	0.179
Napoli	11.094	0.251
Roma	11.031	0.266
Milan	10.984	0.216
Inter	10.967	0.143
Atalanta	10.840	0.127
Torino	10.733	0.227
Lazio	10.239	0.152
Sampdoria	9.987	0.109
Parma	9.724	0.129
Bologna	9.671	0.157
Spal	9.577	0.185
Empoli	9.540	0.179
Udinese	9.526	0.107
Genoa	9.435	0.165
Cagliari	9.423	0.164
Fiorentina	9.332	0.194
Sassuolo	9.224	0.238
Frosinone	8.497	0.244
Chievo	7.373	0.338

(b) Random game order

Team	Mean	Variance
Juventus	12.310	0.184
Napoli	11.813	0.186
Milan	11.423	0.224
Roma	11.401	0.173
Torino	11.190	0.230
Inter	11.138	0.188
Lazio	10.909	0.180
Atalanta	10.647	0.134
Sampdoria	10.444	0.148
Bologna	10.287	0.162
Parma	10.111	0.175
Udinese	9.996	0.177
Sassuolo	9.730	0.225
Spal	9.724	0.207
Fiorentina	9.685	0.185
Cagliari	9.552	0.191
Empoli	9.475	0.197
Genoa	9.070	0.260
Frosinone	8.358	0.262
Chievo	7.638	0.479

4.2 MLB dataset

The suggested model was tested on a new dataset for the MLB Baseball league season 2016. The model was tested with Gibbs sampling and predictions were made using the prediction function 8. The calculated prediction rate was calculated to $r = 0.53$

Even though it's a much lower score compared to the football simulation, that is to be expected. Looking at the final standing in the baseball league we can see that there was only one team having a better win ratio than 60%, compared to football where the top team had roughly 90%. Even though 53% is close to the random guess of 50% the big sample size of 2500 games indicates that the model has a stable slightly better performance than random guessing.

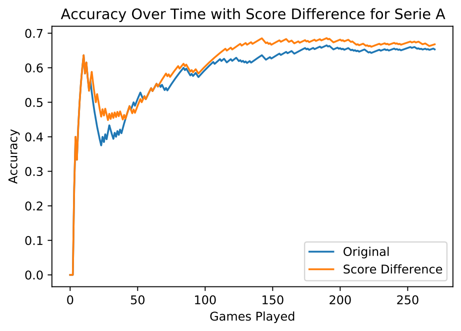
4.3 Extension

Using this model with ADF on the Serie A data, we improve the accuracy from 65.44% to 66.91% as can be seen in table 2. In Figure 7 we show how the prediction rate develops for each game played with and without the extension.

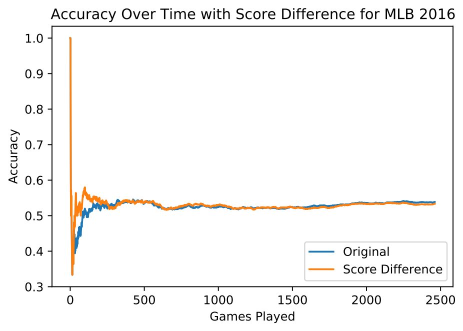
In Figure 7, the effect of the extension can be seen in that the model seems to train a little faster, but that the prediction rates ultimately seem to converge towards similar values.

	Original		Score Difference	
League	Gibbs	MP	Gibbs	MP
Serie A	65.44%	65.44%	66.91%	66.91%
MLB	53.51%	53.88%	53.31%	53.31%

Table 2: Percentage of correctly predicted games depending on the model and league



(a) Serie A



(b) MLB

Figure 7: The prediction rates with and without the score difference extension after each game

5 Conclusion

A Bayesian ranking system for player skills in different sports was implemented and analysed. Two different approaches to calculating the posterior used in the probabilistic models were implemented, Gibbs sampling and Message passing.

The models were implemented and tuned on a dataset containing games from the Italian Football league Serie A. The resulting team ranking seemed to fit well to the final standings in the league, thus the ranking system seemed to capture the team skill well. Predictions given the team skill were made before every game and resulted in 64% prediction rate.

The model was also tested on a dataset over MLB, the American Baseball league. Due to a smaller difference between the skills of the better and worse team in baseball the prediction ratio was worse with a prediction rate of 54%.

An extension was made by implementing a model for score difference, making winning by a bigger score updating the ranking more. This resulted in slightly better predictions. Possible further extensions include side advantage, modelling of draws, adding variance over time to keep it from approaching 0, or even modelling a team as a group of individual players.

References

[1] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: A bayesian skill rating system. In *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press, January 2007.

[2] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. 2021.

[3] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, January 2006.

[4] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.