1RT705/1RT003: Project assignment Advanced Probabilistic Machine Learning 2024

Chen Gu, André Ramos Ekengren, Seyedehmoniba Ravan, Nora Derner
Group 11

Abstract

The *TrueSkill* algorithm, a Bayesian ranking system, is applied to estimate the skills of players in a sports competition using both the Gibbs sampler and the Message Passing algorithm. Conditional distributions are derived, and the Gibbs sampler is implemented to calculate the rankings of teams in the Serie A soccer league. The effects of the sample size in Gibbs sampling and changing the order of matches on the rankings are analyzed. Additionally, a prediction function is introduced to forecast match outcomes. The results are compared with those from the Message Passing algorithm using moment matching. Furthermore, the model is validated on NBA data and extended by taking draws into account and expanding the data set, further refining the ranking process.

1 Introduction

In this report, we implement a refined version of the *TrueSkill* algorithm, focusing on modeling the skills of individual teams. Developed by Microsoft, *TrueSkill* leverages Bayesian inference to provide dynamic skill estimates by continuously updating team skill levels after each match. This approach offers more accurate rankings compared to traditional methods like *Elo*, as it accounts for uncertainty and variability in performance, providing a more precise reflection of each team's true ability over time (4; 7).

2 Methodology and Results

Q1 - Modeling

In the *Trueskill* Bayesian model for a match between two players, s_1 and s_2 represent the skill levels of the players, t denotes the latent outcome of the game (i.e., the difference in skill levels), and y is the observed result of the game. The model can be formulated as

$$\begin{cases} s_1 & \sim \mathcal{N}(\mu_1, \sigma_1^2) \\ s_2 & \sim \mathcal{N}(\mu_2, \sigma_2^2) \\ t \,|\, s_1, s_2 & \sim \mathcal{N}(s_1 - s_2, \sigma_t^2) \\ y & = \text{sign}(t) \end{cases} \tag{1}$$

where the five hyperparameters $\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \sigma_t^2$ need to be selected. The joint distribution representing the full factorization of the model based on the conditional dependencies is expressed as:

$$p(s_1, s_2, t, y) = p(y \mid t) \cdot p(t \mid s_1, s_2) \cdot p(s_1) \cdot p(s_2) \tag{2}$$

Q2 - Conditional Independence

According to the joint distribution (2), the observed outcome y depends directly on t through $p(y \mid t)$. The skill levels s_1 and s_2 only influence y indirectly via t, as captured by the term $p(t \mid s_1, s_2)$.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Assuming that the variable t is observed, s_1 (resp. s_2) and y are conditionally independent. This is illustrated in the Bayesian network in Figure 1 by the head-tail relationship between s_1 and y. The variable t "blocks" the path between s_1 and y. This means that as soon as t is known, the information from s_1 no longer has any additional relevance for the prediction of y (2). This can be expressed mathematically for s_1 by equation (3) and applies similarly to s_2. Hence, $s_1 \perp y \mid t$ and likewise $s_2 \perp y \mid t$.

$$p(s_1, y \mid t) = \frac{p(s_1, y, t)}{p(t)} = \frac{p(y \mid t) \cdot p(t \mid s_1) \cdot p(s_1)}{p(t)} = p(y \mid t) \cdot p(s_1 \mid t) \quad \Rightarrow \quad s_1 \perp y \mid t \tag{3}$$

Independence of Skills Assuming that the variable t is not observed, the skills of the players s_1 and s_2 are modeled as independent random variables (1). The joint distribution of s_1 and s_2 can be written as the product of the individual marginal distributions:

$$p(s_1, s_2) = p(s_1) \cdot p(s_2) \quad \Leftrightarrow \quad s_1 \perp s_2 \tag{4}$$

There is no connecting arrow between the two variables in the Bayesian network in Figure 1, which verifies that the independence of the skills is incorporated into the model. The skill of one player s_1 therefore provides no information about the skill of the other player s_2, prior to a match.

Q3 - Computing with the model

Assume that t conditioned on S with $S = [s_1 \; s_2]^T$, are Gaussian distributed according to:

$$p(S) = \mathcal{N}\left(S; \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}\right) \tag{5}$$

$$p(t \mid S) = \mathcal{N}\left(t \mid S; [1 \; -1] \cdot \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \sigma_t^2\right) \tag{6}$$

Then the full conditional distribution, using that s_1, s_2 are conditionally independent of y given t,

$$p(s_1, s_2 \mid t, y) = p(s_1, s_2 \mid t) = \frac{p(s_1, s_2, t)}{p(t)} = \frac{p(t \mid s_1, s_2) \cdot p(s_1) \cdot p(s_2)}{p(t)} \tag{7}$$

is a multivariate normal distribution, as given by Given Corollary 1 (Affine transformation - Conditioning) from (3), since the mean of t is a linear function of s_1 and s_2, with $S = [s_1 \; s_2]^T$ defined as:

$$\begin{aligned} p(S \mid t) &= \mathcal{N}(S; \mu_{S|t}, \Sigma_{S|t}) \\ \mu_{S|t} &= \Sigma_{S|t}\left(\Sigma_S^{-1}\mu_S + [1 \; -1]^T \Sigma_{t|S}^{-1} t\right) \\ &= \Sigma_{S|t}\left(\begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix}\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix}\frac{1}{\sigma_t^2} t\right) \\ \Sigma_{S|t} &= \left(\Sigma_S^{-1} + [1 \; -1]^T \Sigma_{t|S}^{-1} [1 \; -1]\right)^{-1} \\ &= \left(\begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix}\frac{1}{\sigma_t^2}[1 \; -1]\right)^{-1} \end{aligned} \tag{8}$$

The posterior probability $p(t \mid y, s_1, s_2)$ can be derived using Bayes' theorem (1), combining the likelihood of the match outcome $p(y \mid t)$ and the prior $p(t \mid s_1, s_2)$. Using the Gaussian formula sheet (3), this results in a two-sided truncated Gaussian distribution. Since the condition $y = \text{sign}(t)$ restricts t to either positive values when $y = 1$ or negative values when $y = -1$, the normal distribution is truncated accordingly. Using Gaussian integrals the distribution is normalized for each case. Applying Bayes' theorem results in:

$$\begin{aligned} p(t \mid y, s_1, s_2) &\propto p(y \mid t) p(t \mid s_1, s_2) \\ &= \mathbf{1}_{y=\text{sign}(t)} \cdot \mathcal{N}(t; s_1 - s_2, \sigma_t^2) \end{aligned} \tag{9}$$

And normalizing the right hand side gives us the final distribution for $p(t \mid y, \, s_1, \, s_2)$:

$$\mathcal{N}(t; s_1 - s_2, \sigma_t^2) \cdot \begin{cases} \frac{1}{F(\infty; s_1-s_2, \sigma_t^2) - F(0; s_1-s_2, \sigma_t^2)} = \frac{1}{\int_0^\infty \mathcal{N}(t; s_1-s_2, \sigma_t^2) dt}, & \text{for } y = +1, \quad t > 0 \\ \frac{1}{F(0; s_1-s_2, \sigma_t^2) - F(-\infty; s_1-s_2, \sigma_t^2)} = \frac{1}{\int_{-\infty}^0 \mathcal{N}(t; s_1-s_2, \sigma_t^2) dt}, & \text{for } y = -1, \quad t < 0 \\ 0, & \text{otherwise} \end{cases}$$

Assume that S as well as $t \mid S$ are Gaussian distributed according to (5) and (6). Then, according to Corollary 2 (Affine transformation - Marginalization), the marginal distribution p(t) can be written as

$$\begin{aligned} p(t) &= \mathcal{N}(t; \mu_t, \Sigma_t), \\ \mu_t &= [1 \ -1] \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \mu_1 - \mu_2, \\ \Sigma_t &= \sigma_t^2 + [1 \ -1] \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \sigma_t^2 + \sigma_1^2 + \sigma_2^2, \\ &\Rightarrow p(t) = \mathcal{N}(t; \mu_1 - \mu_2, \sigma_t^2 + \sigma_1^2 + \sigma_2^2), \\ P(y=1) &= P(t > 0) = 1 - P(t \le 0) = 1 - \Phi\left(\frac{\mu_2 - \mu_1}{\sqrt{\sigma_t^2 + \sigma_1^2 + \sigma_2^2}} \right). \end{aligned} \tag{10}$$

Q4 - Bayesian Network

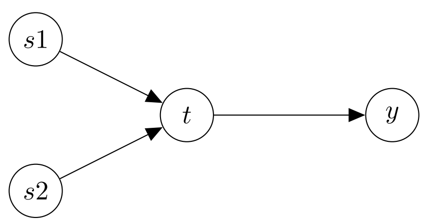

Figure 1: Bayesian network

Q5 - A first Gibbs sampler

The calculated conditional distributions from equations (8) and (9) can now be utilized to implement a Gibbs sampler. The Gibbs sampler is a Markov Chain Monte Carlo (MCMC) method used when we cannot directly sample from the joint distribution of variables, but the conditional distributions can be analytically calculated (1). In our case, we alternate between two steps: First, we draw new values for s_1 and s_2 from the conditional distribution $p(s_1, s_2 \mid t, y)$ (8), based on the current value of t. Then, we draw a new t from the distribution $p(t \mid s_1, s_2, y)$ (9), depending on the updated values of s_1 and s_2. This iterative process eventually converges to the target distribution $p(s_1, s_2 \mid y)$ when t is disregarded, allowing us to generate samples from the desired distribution.

Stationary distribution After enough iterations, the Gibbs sampler converges to a stationary distribution, regardless of the starting conditions. In the trace plots of s_1 and s_2 in Figure 2, we see influence from the starting values at the beginning, which shows that the sampler has not yet reached the desired target distribution. Since we initially chose reasonable starting values with $s_1 = 25$, $s_2 = 25$, and t was randomly drawn, the burn-in period is less noticeable in the plot compared to what it would have been if we had used random values for s_1 and s_2.

Different sample sizes In order to determine the optimal number of samples for the Gibbs sampler, a balanced compromise between accuracy and calculation time needs to be established. In Figure 3, we see that at 200 samples, the calculation is very fast (0.25 seconds), but the estimates for mean and covariance are less stable. With 800 and 1200 samples, the accuracy improves noticeably, and the calculation times of 1.12 and 1.72 seconds respectively are still efficient. Here the estimates stabilize

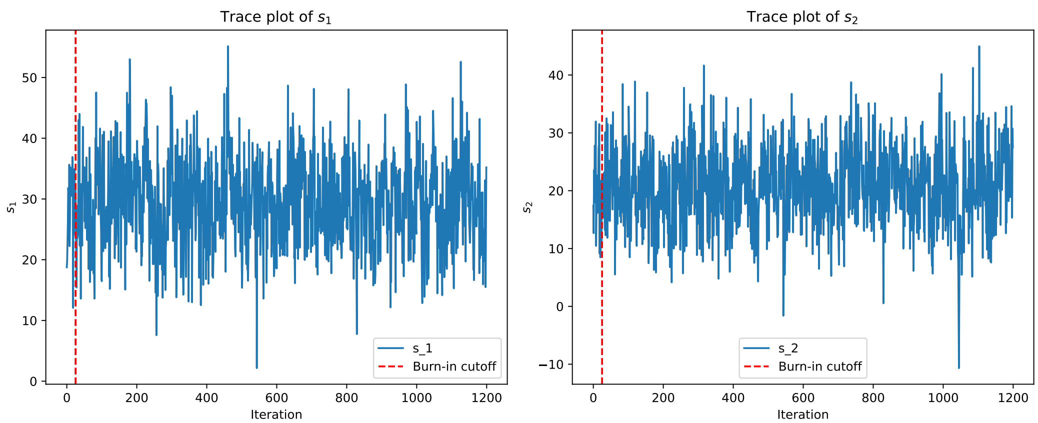

Figure 2: Trace plots of s_1 and s_2

and the covariance matrices show good consistency. At 3000 samples, the estimation improves slightly, but the computation time increases significantly to 3.04 seconds.

Given this trade-off, a sample size between 800 and 1200 seems to be optimal, as the estimates are sufficiently stable and the computation time remains in the acceptable range.

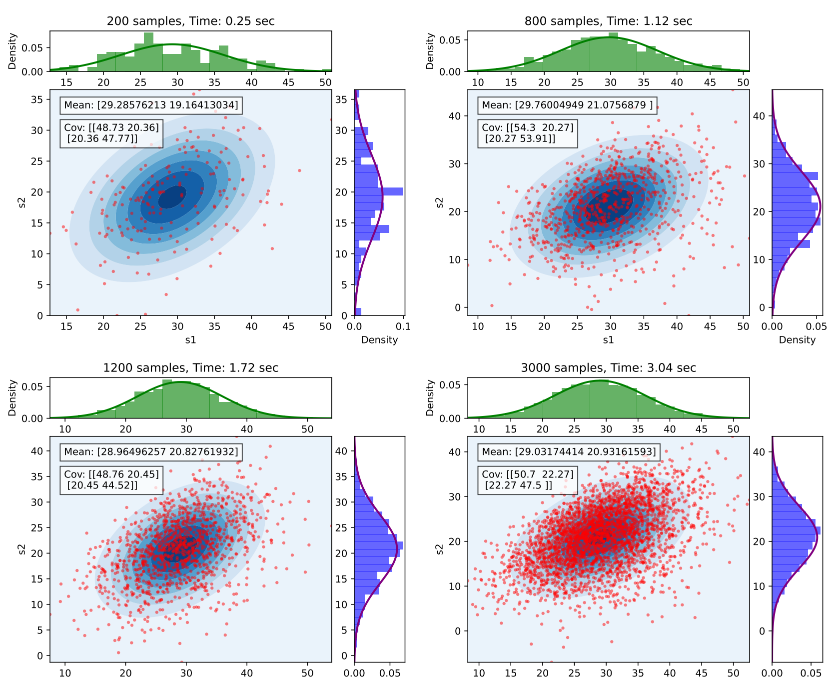

Figure 3: Gibbs Sampling: Impact of Sample Size on Distribution Approximation

Comparison: Prior - Posterior When comparing the prior and Gibbs posterior distributions for s_1 and s_2 in Figure 5, we observe that the curve for s_1 has shifted to the right. This indicates that player 1's skill has improved in the model, as the outcome $y = 1$ implies player 1 won. Additionally, the posterior distribution for s_1 is narrower than the prior, suggesting that we are more confident about player 1's skill after observing the win. Conversely, the curve for s_2 shifts to the left, indicating a decrease in player 2's skill. The posterior for s_2 is also narrower, showing that we have gained more certainty about player 2's skill based on the observed match outcome.

Q6 - Assumed Density Filtering

Application to Serie A Football League Using the implemented Gibbs sampling, we rank football teams in Serie A. The two ranking results, before and after changing the order of the matches, are shown in Table 1. Teams are ranked using Microsoft's conservative skill estimate, defined as $(\text{mean\_team} - 3 \times \text{std\_team})$. Since we use this metric for ranking, the teams are not ordered by their mean skill alone, but also take into account the uncertainty in their skill estimates. The variance reflects the uncertainty in each team's skill assessment. Lower-ranked teams and higher-ranked teams exhibit higher variance, suggesting that their actual skill levels might be even lower and higher than estimated. In contrast, middle-ranked teams show lower variance, reflecting more consistent performance and greater confidence in their skill estimates. (4)

Different ranking outcomes were observed before and after changing the match order. This happens because the process of updating the mean and variance of each team's skill occurs in a different sequence, which means that the skill estimates used for Gibbs sampling vary throughout the process. As a result, the final rankings depend on the order in which the matches are processed, leading to variations in the ranking outcomes.

Table 1: Serie A 18/19: Final Ranking before and after changing the order of the matches

Before Shuffling				**After Shuffling**			
Rank	Team Name	Skill	Var	Rank	Team Name	Skill	Var
1	Napoli	29.748	2.104	1	Napoli	31.375	3.196
2	Atalanta	28.386	1.644	2	Juventus	30.273	2.608
3	Inter	28.302	1.736	3	Milan	28.990	1.909
4	Juventus	29.215	3.045	4	Torino	28.958	2.353
5	Milan	28.878	2.781	5	Atalanta	27.929	1.600
6	Torino	28.016	2.372	6	Roma	28.169	1.927
7	Roma	26.977	1.893	7	Inter	27.845	1.965
8	Lazio	25.587	1.795	8	Lazio	26.330	1.618
9	Sampdoria	24.669	1.610	9	Sampdoria	25.700	1.472
10	Bologna	24.157	1.644	10	Bologna	25.322	1.688
11	Empoli	23.948	1.610	11	Empoli	23.749	1.532
12	Spal	23.862	1.740	12	Genoa	24.330	2.114
13	Parma	23.454	1.454	13	Udinese	23.864	1.740
14	Udinese	23.947	2.430	14	Cagliari	23.843	2.109
15	Cagliari	23.181	1.936	15	Sassuolo	24.154	2.463
16	Genoa	23.550	2.402	16	Parma	23.432	1.921
17	Fiorentina	22.491	2.137	17	Spal	23.517	2.064
18	Sassuolo	22.858	2.638	18	Fiorentina	23.512	2.220
19	Frosinone	20.214	2.529	19	Frosinone	21.129	2.834
20	Chievo	17.811	4.850	20	Chievo	17.575	5.520

Q7 - Using the model for predictions

To predict the winning team, we implement a `predict_winner` function based on the marginal distribution derived in (10), which represents the probability of team 1 winning. In each iteration, we calculate this probability using the most recently updated estimates of each team's skill mean and variance. After each match, we update the skill levels of both teams by treating the updated posterior as the prior for predicting the next match. As more match results are processed, the skill estimates improve, and the model becomes more effective at predicting outcomes. This is reflected in our final prediction rate of 65 %, which outperforms random guessing of 50 %.

Q8 - Factor graph

The factor graph of the model is presented in Figure 4, and we identify some of the relevant messages needed to calculate $p(t|y = y_{obs})$:

$$\mu_1 = \mu_2 = \mathbf{1}_{\{y=y_{\text{obs}}\}}, \mu_3 = \sum_y \mathbf{1}_{\{y=\text{sign}(t)\}}\mathbf{1}_{\{y=y_{\text{obs}}\}} = \mathbf{1}_{\{y_{\text{obs}}=\text{sign}(t)\}} \tag{11}$$

$$\mu_7 = \mu_8 = \mathcal{N}(s_1; m_{s_1}, \sigma_{s_1}^2) \tag{12}$$

$$\mu_6 = \mu_5 = \mathcal{N}(s_2; m_{s_2}, \sigma_{s_2}^2) \tag{13}$$

$$\mu_4 = \iint \mathcal{N}(t; s_1 - s_2, \sigma_t^2)\mu_5\mu_8\, ds_1\, ds_2 = \mathcal{N}(t; m_{s_1} - m_{s_2}, \sigma_{s_1}^2 + \sigma_{s_2}^2 + \sigma_t^2) \tag{14}$$

We have that $p(t|y = y_{obs}) \propto \mu_4\mu_3$, resulting in a truncated gaussian as the distribution for $t|y = y_{obs}$ when the product of the messages are normalized with respect to t.

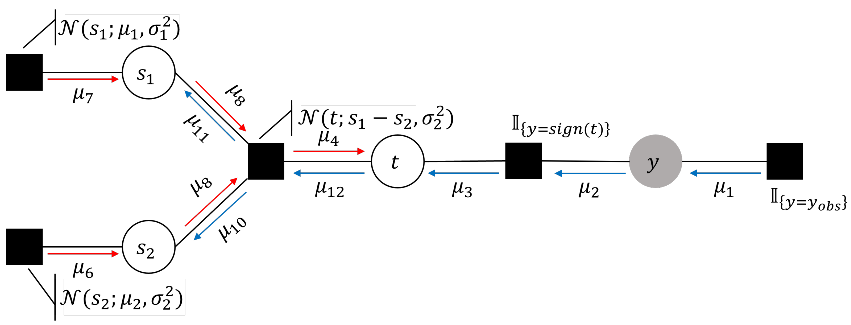

Figure 4: Factor graph

Q9 - A Message-Passing algorithm

A message-passing algorithm Instead of using the Gibbs Sampler we now use the Message Passing Algorithm to calculate the posterior distributions of s_1 and s_2 more efficiently by leveraging the fact that the involved distributions are approximately Gaussian. This makes Moment Matching particularly suitable, as we can update the distributions of s_1 and s_2 through simple updates of the means and variances (1). At the moment, $p(t|y = y_{obs})$ is a truncated gaussian, but using moment matching we approximate it with the corresponding gaussian $\hat{p}(t|y = y_{obs})$ that has the same mean and variance. The outgoing message from t will then be $\mu_{12} = \frac{\hat{p}(t|y=y_{obs})}{\mu_4}$, a division between two gaussian distributions. All the following messages to s_1, s_2 that are used in the calculation of their posterior will simply be combinations of multiplications and marginalizations of gaussian distributions as follows:

$$\mu_{12} = \frac{\hat{p}(t \mid y = y_{\text{obs}})}{\mu_4} = \frac{\mathcal{N}(t; m_{MM}, \sigma_{MM}^2)}{\mathcal{N}(t; m_4, \sigma_4^2)} \tag{15}$$

$$\mu_{10} = \iint \mu_8\mu_{12}p(t|s)\, ds_1\, dt = \iint \mathcal{N}(s_1; m_8, \sigma_8^2)\mathcal{N}(t; m_{12}, \sigma_{12}^2)\mathcal{N}(t; s_1 - s_2, \sigma_t^2)\, ds_1\, dt \tag{16}$$

$$= \mathcal{N}(s_1; m_8 - m_{12}, \sigma_8^2 + \sigma_{12}^2 + \sigma_t^2) \tag{17}$$

$$\mu_{11} = \iint \mu_5\mu_{12}p(t|s)\, ds_2\, dt = \iint \mathcal{N}(s_2; m_5, \sigma_5^2)\mathcal{N}(t; m_{12}, \sigma_{12}^2)\mathcal{N}(t; s_1 - s_2, \sigma_t^2)\, ds_2\, dt \tag{18}$$

$$= \mathcal{N}(s_2; m_5 + m_{12}, \sigma_5^2 + \sigma_{12}^2 + \sigma_t^2) \tag{19}$$

Where m_i, σ_i^2 are the means and variances of message i. Finally, the posterior of the skills are calculated as

$$p(s_1|y) = \mu_{11} \cdot \mu_7 = \mathcal{N}(s_1, m_5 + m_{12}, \sigma_5^2 + \sigma_{12}^2 + \sigma_t^2) \cdot \mathcal{N}(s_1; m_7, \sigma_7^2), \tag{20}$$

$$p(s_2|y) = \mu_{10} \cdot \mu_6 = \mathcal{N}(s_1, m_8 - m_{12}, \sigma_8^2 + \sigma_{12}^2 + \sigma_t^2) \cdot \mathcal{N}(s_2; m_6, \sigma_6^2). \tag{21}$$

The updated mean for Player 1 is 29.4327 with a variance of 49.7957, while for Player 2, the mean is 20.5673 with a variance of 49.7957.

Comparison with Gibbs Sampler When comparing the two different methods for calculating the posterior distributions—Gibbs sampling and Message Passing—we observe that the resulting posteriors are nearly identical. The posteriors for both s_1 and s_2 from the two methods lie almost perfectly on top of each other, as shown in Figure 5. This indicates that both methods provide highly consistent results, even though the computational approaches differ. The Gibbs sampler iteratively samples from the conditional distributions, while the Message Passing Algorithm uses efficient updates via Moment Matching. Despite these differences, the fact that the posteriors are so close demonstrates that both approaches accurately capture the underlying distribution of s_1 and s_2.

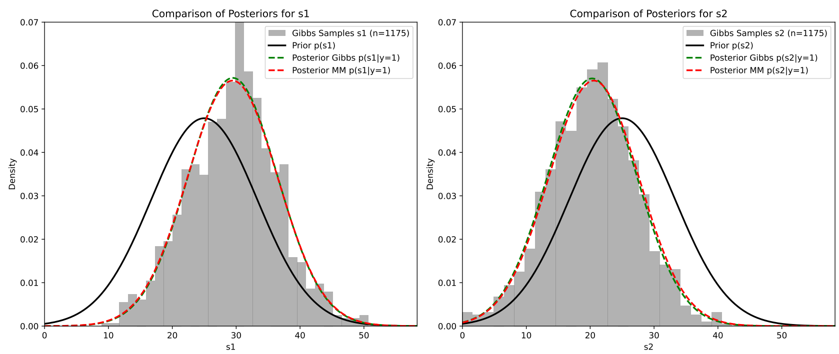

Figure 5: Posterior Distributions for s1 and s2: Gibbs Sampling vs Moment Matching

Q10 - Your own data

Table 2: NBA: Final Ranking before and after changing the order of the matches

Before Shuffling				**After Shuffling**			
Rank	Team Name	Skill	Var	Rank	Team Name	Skill	Var
1	Lakers	29.416	0.245	1	Lakers	27.894	0.294
2	Royals	28.827	0.390	2	Royals	27.182	0.221
3	Nationals	28.147	0.395	3	Nationals	26.878	0.361
4	Knicks	27.391	0.190	4	Knicks	25.982	0.333
5	Pistons	27.299	0.509	5	Pistons	25.699	0.331
6	Olympians	26.377	0.267	6	Stags	25.946	0.508
7	Packers	26.135	0.306	7	Olympians	25.489	0.352
8	Stags	26.900	0.658	8	Warriors	24.545	0.229
9	Blackhawks	26.110	0.420	9	Packers	24.833	0.536
10	Warriors	25.963	0.397	10	Celtics	24.198	0.296
11	Celtics	25.337	0.256	11	Blackhawks	23.954	0.243
12	Capitols	25.145	0.250	12	Bullets	23.691	0.318
13	Bullets	25.176	0.452	13	Capitols	23.840	0.393
14	Bombers	25.037	0.601	14	Bombers	23.963	0.452
15	Redskins	23.520	0.458	15	Redskins	22.323	0.597
16	Hawks	23.491	0.898	16	Hawks	22.131	0.806
17	Nuggets	21.599	0.858	17	Nuggets	20.378	1.018

To verify whether our implemented *TrueSkill* model can be applied to other datasets, we tested it on the first 800 matches from the "Complete NBA Match Results (1949-2024)" dataset on Kaggle (5). The dataset includes 17 basketball teams, and similar to our previous analysis, we focus solely on wins and losses, ignoring score differences. In preprocessing, we loaded the NBA dataset, mapped team names to numeric indices, added these indices as columns, and computed the score differences.

A new DataFrame with team indices and score differences was created and then converted into a NumPy array for further analysis. We employed the Gibbs sampler for the calculations and further examined how the team rankings change when the order of the matches is shuffled. The results of this analysis are presented in Table 2. In both versions, we see the greatest variance in the last two teams. After shuffling, the ranking of the teams has changed minimally. Both Gibbs sampling processes took 10 minutes each to compute.

Q11 - Open-ended project extension

Large dataset The first extension focuses on improving our prediction accuracy with increasing the size of the dataset. Instead of using only the provided dataset of the years 2018/2019, we extend the dataset by a factor of 10 with additional data from Serie A for the years 2014 to 2022 (6). By enlarging the data set, we observed the final ranking on the left site in Table 3. An overall lower variance compared to previous rankings with less data indicates increased confidence in the accuracy of the skill assessments. Our prediction accuracy is 70%. However, the Gibbs sampling along with predicting the winner after each match took a total of 28.54 minutes (The hardware specifications include a Microsoft Windows 10 Pro operating system, an Intel Core i7-7700HQ CPU @ 2.80GHz with 4 cores and 8 logical processors, and 16 GB of installed RAM).

Including draws In the second extension, we expanded the model consider the large dataset by including draws. We changed the architecture of the Gibbs sampler by adding the third case when calculating t. This was done by considering a draw margin ε_G for t, such that if the outcome of a match is a draw, t will be sampled from a truncated gaussian with bounds based on ε_G. We also altered the prediction function such that any time the predicted probability of winning or losing is within $[0.5-\varepsilon_P, 0.5+\varepsilon_P]$ the game is considered a draw, where ε_P is a hyperparameter representing the needed deviation from 50% to be certain that some team won. In the final results presented in Table 3 on the right side, we observe a shift in the rankings, especially in terms of who is leading the table, which differs significantly. Moreover, the variances have decreased as more data was incorporated by including the draws into the model. The prediction rate is 51%, which is less than the 70% achieved in the model where we ignored the draws, indicating that the model's performance is almost as weak as random guessing.

3 Discussion

The Gibbs Sampler provides a straightforward approach to estimating posterior distributions but can be computationally intensive, as we observed in our experiments. Additionally, we observe the order in which matches are processed affects the final rankings, emphasizing the dependency of match sequence. As skill estimates are updated from game to game, the skill updates become smaller as the model's certainty increases.

Another key observation is that the Gibbs Sampler is only feasible when conditional distributions can be computed, which becomes increasingly cumbersome as model complexity grows. Additionally, handling imbalanced classes is crucial. We found that ignoring draws led to a significant loss of valuable match outcome data, which negatively impacted the accuracy of the skill estimates. Conversely, incorporating draws ($y = 0$) increased the complexity of the model but reduced the uncertainty in skill estimates, although this added complexity also resulted in higher computational demands. Therefore, it is crucial to balance model complexity, accuracy, and computational efficiency. While Message Passing offers a more efficient alternative, especially for larger datasets, Gibbs Sampling provides more model flexibility but at a higher computational cost.

Conclusion In this report, we demonstrated the effectiveness of a refined *TrueSkill* model for ranking teams using Bayesian inference. Both Gibbs Sampling and the Moment Matching approach yielded accurate rankings. Notably, the Moment Matching approach is deterministic, ensuring consistent results, whereas Gibbs Sampling is stochastic, introducing randomness in the estimates across different runs. Future work could explore incorporating additional features, such as score differences or home team advantage, to further improve estimate accuracy. Moreover, the model could be applied to other domains beyond sports where ranking entities based on performance is essential.

References

[1] Bishop, Christopher M. Pattern Recognition and Machine Learning. Information Science and Statistics. New York: Springer, 2006.

[2] Hamis, Sara. Lecture 4 notes in 1RT003/1RT705: Advanced probabilistic machine learning, October 2024.

[3] Wahlström, Niklas. Formula sheet for the Gaussian distribution in 1RT003/1RT705: Advanced probabilistic machine learning, October 2024.

[4] Microsoft Research. TrueSkill ranking system, `https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/`, accessed October 07, 2024.

[5] Joy Biswas. NBA Matches Results (1949-2024), `https://www.kaggle.com/datasets/joybiswas389/nba-matches-results-1949-2024`, accessed October 06, 2024.

[6] Ferrariboy4k. Top 5 European Football Leagues (1993-2022) Results Dataset, `https://www.kaggle.com/datasets/ferrariboy4k/top5leagues`, accessed October 06, 2024.

[7] R. Herbrich, T. Minka, and T. Graepel. "TrueSkill(TM): A Bayesian Skill Rating System." In *Advances in Neural Information Processing Systems*, January 2007, MIT Press.

Appendix

Table 3: Large Dataset: Serie A - 2014-2022

Without Draws				Including Draws			
Rank	Team Name	Skill	Var	Rank	Team Name	Skill	Var
1	Juventus	29.351	0.113	1	Lazio	28.310	0.102
2	Napoli	28.248	0.067	2	Sampdoria	27.280	0.056
3	Inter	26.875	0.076	3	Inter	26.803	0.041
4	AC Milan	27.205	0.225	4	Carpi	26.755	0.093
5	Lazio	26.650	0.149	5	Napoli	26.324	0.063
6	Roma	26.947	0.309	6	Chievo	26.285	0.062
7	Atalanta	26.440	0.218	7	Roma	25.657	0.066
8	Fiorentina	25.091	0.130	8	Juventus	25.443	0.110
9	Sassuolo	24.523	0.177	9	Brescia	24.884	0.048
10	Torino	24.439	0.178	10	Benevento	24.686	0.068
11	Genoa	23.638	0.061	11	SPAL	24.861	0.134
12	Sampdoria	24.030	0.198	12	Cesena	23.863	0.043
13	Bologna	23.432	0.071	13	Spezia	23.924	0.103
14	Udinese	23.178	0.085	14	Empoli	23.834	0.100
15	Cagliari	23.408	0.140	15	AC Milan	23.589	0.057
16	Chievo	23.136	0.213	16	Torino	23.796	0.117
17	Verona	22.927	0.179	17	Venezia	23.447	0.061
18	Empoli	23.132	0.338	18	Fiorentina	23.575	0.147
19	Parma	22.649	0.203	19	Pescara	22.877	0.131
20	Palermo	22.802	0.357	20	Lecce	23.211	0.226
21	Spezia	22.809	0.364	21	Frosinone	23.071	0.233
22	SPAL	22.347	0.371	22	Verona	23.068	0.332
23	Crotone	21.526	0.430	23	Udinese	22.775	0.234
24	Carpi	23.202	1.581	24	Palermo	23.788	0.760
25	Benevento	21.836	0.664	25	Parma	23.385	0.720
26	Frosinone	21.269	0.614	26	Atalanta	22.549	0.343
27	Lecce	22.310	1.280	27	Cagliari	22.411	0.384
28	Salernitana	21.824	1.861	28	Crotone	22.958	0.795
29	Venezia	20.935	1.316	29	Sassuolo	21.888	0.819
30	Brescia	20.584	1.293	30	Salernitana	22.222	0.583
31	Pescara	18.481	2.412	31	Bologna	21.137	0.768
32	Cesena	18.494	2.487	32	Genoa	21.723	0.578