1RT705/1RT003: Project assignment Advanced Probabilistic Machine Learning 2023

Emir Esenov
Master's student in Data Science
Uppsala University

Ola Karrar
Master's student in Data Science
Uppsala University

Marzieh Rahnamatoupkanloo
Master's student in Computer Science
Uppsala University

Abstract

This paper delves into the intricacies of the TrueSkill Bayesian model, a prevalent technique used for player ranking in games. We provide a mathematical formulation of the model and consider its Bayesian network representation. To understand the model's computational aspects, we derive three crucial probability distributions and employ methods such as Gibbs sampling and Assumed Density Filtering (ADF). By applying ADF to Serie A football league data from the 2018 season, we showcase the model's ability to process streams of matches. Our predictive capabilities, tested on the same dataset, yielded a prediction rate of $r = 0.64$, confirming the model's efficacy. Finally, we present the factor graph representation of the TrueSkill model, delineating the message-passing scheme, which offers insights into the computational benefits of factor graphs in Bayesian modeling. This comprehensive study provides a holistic understanding of the TrueSkill model, its applications, and its computational nuances.

Introduction

Understanding player skill levels in competitive scenarios is vital for balanced match-ups and progress tracking. While the Elo rating system has historically been favored in board games like chess, it has limitations. The TrueSkill Bayesian model by Microsoft offers a modern alternative, giving a nuanced view of player abilities by providing a skill distribution. This report delves into TrueSkill's mathematical foundation, its Bayesian Network representation, and its behavior through computational experiments.

1 Modeling

The TrueSkill Bayesian model for one match between two players can be written as follow:

$$p(s_1) = \mathcal{N}(s_1; \mu_1, \sigma_1^2) \qquad p(s_2) = \mathcal{N}(s_2; \mu_2, \sigma_2^2)$$

$$p(t|s_1, s_2) = \mathcal{N}(t; s_1 - s_2, \sigma_3^2) \qquad P(y|t) = \begin{cases} 1, & \text{if } y = \text{sign}(t) \\ 0, & \text{otherwise.} \end{cases}$$

$$p(s_1, s_2, t, y) = p(s_1) \times p(s_2) \times p(t|s_1, s_2) \times p(y|t)$$

this model has five hyperparameters $s_1, s_2, \sigma_1^2, \sigma_2^2, \sigma_3^2$.

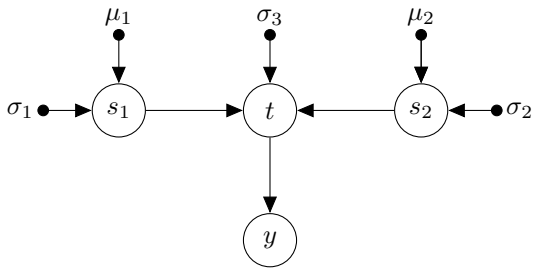

Figure 1: The Bayesian network correspond to TrueSkill model

2 The Bayesian Network Representation

The Bayesian network of the TrueSkill bayesian model can be found in Figure 1. We can prove that s_1 is conditionally independent of y given t as mathematically as follows:

$$p(s_1, y \mid t) = \frac{p(s_1, y, t)}{p(t)} = \frac{p(y \mid t)p(t \mid s_1)p(s_1)}{p(t)} = p(y \mid t)p(s_1 \mid t) \Longrightarrow s_1 \perp y \mid t.$$

Similarly, for s_2:

$$p(s_2, y \mid t) = \frac{p(s_2, y, t)}{p(t)} = \frac{p(y \mid t)p(t \mid s_2)p(s_2)}{p(t)} = p(y \mid t)p(s_2 \mid t) \Longrightarrow s_2 \perp y \mid t.$$

We can further note, as depicted in Figure 1, that there exists a head-to-tail relationship between s_1 and y. Given that t is an observed variable, this path between them becomes blocked, resulting in the independence of s_1 and y. same hold for s_2.

3 Computing with the model

This section elaborates on the derivation of three probability distributions: the full conditional distribution of the skills, denoted as $p(s_1, s_2|t, y)$, the full conditional distribution of the game outcome, represented as $p(t|s_1, s_2, y)$, and the marginal probability of Player 1 winning the game, denoted as $p(y = 1)$.

1. First, To determine $p(s_1, s_2|y, t)$, we apply the definition of conditional independence, which allows us to conclude:

$$p(s_1, s_2|y, t) = \frac{p(s_1, s_2, y|t)}{p(y|t)} \overset{s_i \perp y|t}{=} \frac{p(s_1, s_2|t)p(y|t)}{p(y|t)} = p(s_1, s_2|t),$$

Applying Corollary 1, let:

$$\mathbf{x}_a = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \boldsymbol{\mu}_a = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \boldsymbol{\Sigma}_a = \begin{bmatrix} \sigma_1^2 & 0, \\ 0 & \sigma_2^2 \end{bmatrix},$$
$$\mathbf{x}_b = t, \quad \mathbf{A} = \begin{bmatrix} 1 & -1 \end{bmatrix}, \quad \mathbf{b} = \mathbf{0}, \boldsymbol{\Sigma}_{b|a} = \sigma_3^2$$

Therefore, we can deduce that $p(s_1, s_2|t) = \mathcal{N}(\mathbf{x}a; \boldsymbol{\mu}a|b, \boldsymbol{\Sigma}_{a|b})$ where:

$$\boldsymbol{\Sigma}_{a|b} = \left(\boldsymbol{\Sigma}_a^{-1} + \mathbf{A}^T \boldsymbol{\Sigma}_{b|a}^{-1} \mathbf{A}\right)^{-1} = \left(\begin{bmatrix} \sigma_1^2 & 0, \\ 0 & \sigma_2^2 \end{bmatrix}^{-1} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} \sigma_3^{-2} \begin{bmatrix} 1 & -1 \end{bmatrix} \right)^{-1} \tag{1}$$

$$= \left(\begin{bmatrix} \frac{\sigma_2^2}{\sigma_1^2 \sigma_2^2} & 0, \\ 0 & \frac{\sigma_1^2}{\sigma_1^2 \sigma_2^2} \end{bmatrix} + \begin{bmatrix} \sigma_3^{-2} & -\sigma_3^{-2}, \\ -\sigma_3^{-2} & \sigma_3^{-2} \end{bmatrix} \right)^{-1} = \left(\begin{bmatrix} \sigma_1^{-2} + \sigma_3^{-2} & -\sigma_3^{-2} \\ -\sigma_3^{-2} & \sigma_2^{-2} + \sigma_3^{-2} \end{bmatrix} \right)^{-1} \tag{2}$$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\Sigma}_{a|b}\left(\boldsymbol{\Sigma}_a^{-1}\boldsymbol{\mu}_a + \mathbf{A}^\mathsf{T}\boldsymbol{\Sigma}_{b|a}^{-1}(\mathbf{x}_b - \mathbf{b})\right).$$

2. Second, to calculate $p(t|s_1, s_2, y)$, we can leverage the definition of conditional probability:

$$p(t|s_1, s_2, y) = \frac{p(s_1, s_2, t, y)}{p(s_1, s_2, y)} = \frac{p(s_1)p(s_2)p(t|s_1, s_2)p(y|t)}{p(s_1, s_2, y)}$$

since the probability $p(s_1, s_2, y)$ is equal to:

$$p(s_1, s_2, y) = \int_t p(s_1, s_2, t, y)dt = \int_t p(s_1)p(s_2)p(t|s_1, s_2)p(y|t)dt = p(s_1)p(s_2)\int_t p(t|s_1, s_2)p(y|t)dt$$

substitute the value of $p(s_1, s_2, y)$ in $p(t|s_1, s_2, y)$ yields:

$$p(t|s_1, s_2, y) = \frac{p(s_1)p(s_2)P(t|s_1, s2)P(y|t)}{p(s_1)p(s_2)\int_t p(t|s_1, s2)p(y|t)dt} = \frac{p(t|s_1, s2)p(y|t)}{\int_t p(t|s_1, s2)p(y|t)dt} \propto P(t|s_1, s2)P(y|t)$$

since $p(y|t)$ is 1 if $y = \text{sign}(t)$ and 0 otherwise, we can infer that:

$$p(t|s_1, s_2, y) = \begin{cases}\mathcal{N}(t; s_1 - s_2, \sigma_3^2), & \text{if } y = \text{sign}(t)\\ 0, & \text{otherwise.}\end{cases}$$

3. Finally, to compute $p(y = 1) = p(t > 0)$, we start by applying Corollary 2 to determine $p(t)$. let define:

$$\mathbf{x}_a = \begin{bmatrix}s_1\\ s_2\end{bmatrix}, \boldsymbol{\mu}_a = \begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \boldsymbol{\Sigma}_a = \begin{bmatrix}\sigma_1^2 & 0,\\ 0 & \sigma_2^2\end{bmatrix},$$
$$\mathbf{x}_b = t, \quad \mathbf{A} = \begin{bmatrix}1 & -1\end{bmatrix}, \quad \mathbf{b} = \mathbf{0}, \boldsymbol{\Sigma}_{b|a} = \sigma_3^2$$

Then, applying Corollary 2, we have: $t \sim \mathcal{N}(\mathbf{x}_b, \boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)$, where:

$$\boldsymbol{\mu}_b = \boldsymbol{A}\boldsymbol{\mu}_a + \mathbf{b} = \mu_1 - \mu_2,$$
$$\boldsymbol{\Sigma}_b = \boldsymbol{\Sigma}_{b|a} + \boldsymbol{A}\boldsymbol{\Sigma}_a\boldsymbol{A}^T = \sigma_3^2 + \begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}\sigma_1^2 & 0,\\ 0 & \sigma_2^2\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix} = \sigma_1^2 + \sigma_2^2 + \sigma_3^2$$

Hence, $p(t) = \mathcal{N}(t; \mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2 + \sigma_3^2)$ and as a result:

$$p(y = 1) = p(t > 0) = \int_{t=0}^{\infty} \mathcal{N}(t; \mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2 + \sigma_3^2)dt.$$

4 Gibbs sampler

Using the results from Section 3, we implement a Gibbs sampler to target the distribution $p(s1, s2, t|y = 1)$. With these samples, we recover the Trueskill representation of the skills via Gaussian approximation of $p(s1|y = 1)$, $p(s2|y = 1)$.

The experiment began with analyzing the burn-in to see how long it takes for the sampler to reach its stationary distribution. Figure 2 shows a relatively fast convergence at around 10 samples.

To confirm that selecting a burn-in value of 10 yields better results, we used the Gelman-Rubin (GR) statistic (denoted $\hat{R}$) as a performance measure, which is the square root of the ratio of two estimators for the target variance of two or more MCMC chains. In finite samples, the numerator overestimates the target variance and the denominator underestimates it. Each estimator also converges to the target variance, meaning that $\hat{R}$ converges to 1 as n increases. This means that the closer the value of $\hat{R}$ is to 1 the better, and when $\hat{R}$ is sufficiently close to 1, the GR measure declares convergence (1).

We calculated $\hat{R}$ for two samples taken from the Gibbs sampler, then re-ran the experiment with the new burn value to compare the results. These results can be found in table 1.

After selecting a suitable burn value, we considered the tradeoff between the accuracy of the estimate and the computational time. Although we would have preferred more accurate estimates by selecting

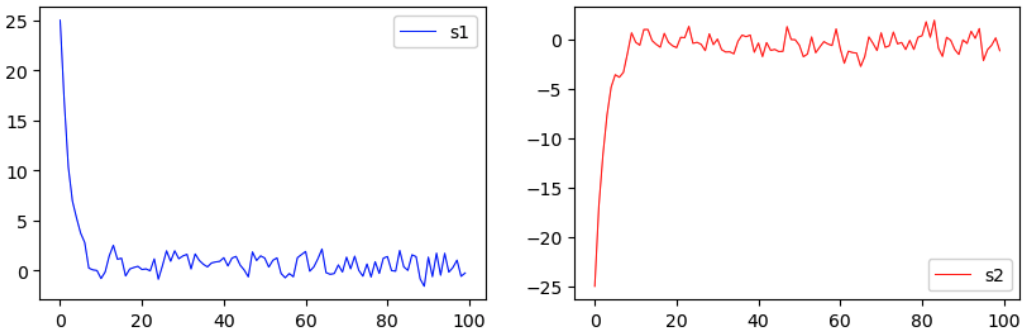

Figure 2: Gibbs sampler results for s_1, s_2 with 100 samples.

	Experiment 1	Experiment 2
n samples	100	100
Burn value	0	10
$\hat{R}$	1.040	1.005

Table 1: Experiment results for the Gibbs sampler.

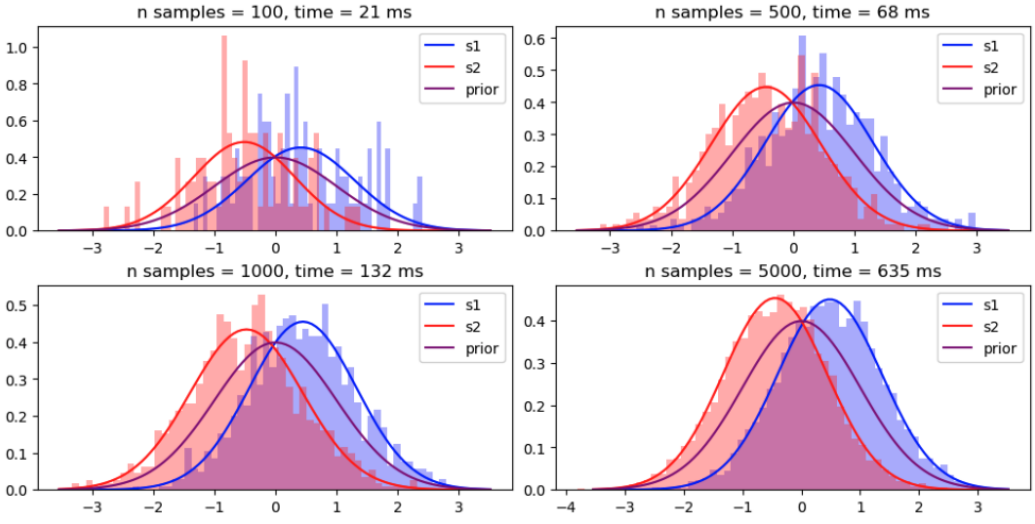

Figure 3: Accuracy and computational time for different amounts of samples.

higher numbers of samples, we also had to take into account computational cost when performing our experiments. Figure 3 shows the tradeoff between accuracy of the estimate from the amount of samples and its effects on the computational time. Based on these results, we selected a sample size of $n = 1000$ for our experiments, as it had a reasonable tradeoff between accuracy and computational time.

Figure 3 also includes the prior distribution for $p(s_1)$ and $p(s_2)$. We can see that the outcome of the game affects the skill of the two players by shifting their respective means and decreasing their variance. Player 1 has increased their skill rating while Player 2 has decreased theirs, both players receiving a lower variance in their skill distributions.

5 Assumed Density Filtering

We can use Assumed Density Filtering (ADF) to process a stream of different matches between teams, where each team is assigned a single skill in the TrueSkill framework. We consider a dataset which consists of all the games from the 2018 season of the Serie A football league, and for each match between two teams, we use the posterior distribution of the previous match as the prior distribution for the current match. We conduct two experiments, one with the regular order of games in the series, and then another one where we shuffle the order of the games in order to analyze the effects that it has on our TrueSkill model. Table 2 shows our results.

Regular order games				Shuffled order games			
Team	Mean	Variance	Rank	Team	Mean	Variance	Rank
Juventus	1.149842	0.120680	1	Juventus	1.290665	0.101507	1
Napoli	0.856466	0.085749	2	Napoli	0.771996	0.064746	2
Milan	0.754339	0.101962	3	Roma	0.669304	0.089386	3
Inter	0.621420	0.082661	4	Torino	0.608599	0.116756	4
Atalanta	0.526668	0.059607	5	Milan	0.608038	0.090422	5
Roma	0.493406	0.087014	6	Inter	0.468824	0.079625	6
Torino	0.473580	0.085092	7	Atalanta	0.430103	0.086941	7
Lazio	0.203698	0.076870	8	Sampdoria	0.138272	0.066391	8
Sampdoria	0.016833	0.063655	9	Lazio	0.106944	0.073622	9
Bologna	-0.088170	0.094308	10	Bologna	-0.047553	0.079493	10
Spal	-0.242983	0.077814	11	Sassuolo	-0.209004	0.087148	11
Genoa	-0.291705	0.082856	12	Fiorentina	-0.297195	0.107691	12
Empoli	-0.306131	0.070067	13	Spal	-0.314325	0.086491	13
Udinese	-0.312908	0.068429	14	Udinese	-0.350802	0.071838	14
Cagliari	-0.396286	0.075673	15	Parma	-0.390394	0.079448	15
Sassuolo	-0.401763	0.104551	16	Cagliari	-0.432870	0.076668	16
Parma	-0.419911	0.085861	17	Genoa	-0.451223	0.071708	17
Fiorentina	-0.544295	0.116863	18	Empoli	-0.456139	0.062237	18
Frosinone	-0.920344	0.091237	19	Frosinone	-0.817829	0.075688	19
Chievo	-1.395620	0.175208	20	Chievo	-1.413659	0.122559	20

Table 2: Assumed Density Filtering experiment results.

From Table 2, we can see that order matters in the TrueSkill model. This makes intuitive sense — for example, if we imagine that the top team was not rank 1 throughout the entire series but only got there by the end, then skill changes resulting from games versus that team would differ depending on when the game was played during the series. However, from Table 2 we can also see that a lot of regularity is maintained even with difference in order.

Additionally, we note that the variances are significantly reduced for all teams, since they were all given an initial prior distribution of $\mathcal{N}(0, 1)$. This means that the TrueSkill model has become more sure of the skills after being trained on the outcomes of all the games.

6 Using the model for predictions

By adding a simple prediction function to our experiment in Section 5 which predicts that the team with the higher skill wins, we can use TrueSkill to predict the outcomes of games between two teams. Specifically, we can compute the *one-step-ahead* predictions of the results based on the model and compare them with the actual results for all matches. From this procedure, we can report the *prediction rate* as

$$r = \frac{\text{number of correct guesses}}{\text{number of total guesses}}.$$

This procedure yields a prediction rate of $r = 0.64$, which shows that our TrueSkill model has some predictive power, and performs better than random guessing at $r \approx 0.5$.

7 Factor graph

Figure 4 visually represents the factor graph for our Bayesian network, which is governed by the joint probability distribution:

$$P(s_1, s_2, t, y) = P(s_1) \cdot P(s_2) \cdot P(t|s_1, s_2) \cdot P(y|t) = f_1(s_1) \cdot f_2(s_2) \cdot f_3(t, s_1, s_2) \cdot f_4(t, y)$$

Here are the explicit message expressions within the factor graph:

$$\mu_1(s_1) = \mathcal{N}(s_1; \mu_1, \sigma_1^2), \quad \mu_2(s_2) = \mathcal{N}(s_2; \mu_2, \sigma_2^2), \quad \mu_3(y) = \delta(y - y_{obs})$$
$$\mu_4(s_1) = \mu_1(s_1) = \mathcal{N}(s_1; \mu_1, \sigma_1^2), \quad \mu_5(s_2) = \mu_2(s_2) = \mathcal{N}(s_2; \mu_2, \sigma_2^2)$$

For $\mu_6(t)$:

$$\mu_6(t) = \int_{s_1, s_2} f_3(s_1, s_2, t)\mu_4(s_1)\mu_5(s_2)ds_1 ds_2 \tag{3}$$
$$= \int_{s_1, s_2} \mathcal{N}(t, s_1 - s_2; \sigma_3^2) \cdot \mathcal{N}(s_1; \mu_1, \sigma_1^2) \cdot \mathcal{N}(s_2; \mu_2, \sigma_2^2)ds_1 ds_2 \quad \text{(Using Corollary 2)} \tag{4}$$
$$= \int_{s_2} \mathcal{N}\left(t, -s_2 + \mu_1; \sigma_1^2 + \sigma_3^2\right) \cdot \mathcal{N}(s_2; \mu_2, \sigma_2^2)ds_2 \quad \text{(Using Corollary 2)} \tag{5}$$
$$= \mathcal{N}\left(t; \mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2 + \sigma_3^2\right) \tag{6}$$

For $\mu_7(y)$:

$$\mu_7(y) = \sum_y f_4(t, y) \cdot \mu_3(y)dy = \sum_y \delta(y = \text{sign}(t)) \cdot \delta(y - y_{obs})dy = \delta(y = \text{sign}(t)) \tag{7}$$
$$= \begin{cases} 1, & \text{if } y_{obs} = \text{sign}(t) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

These derivations clarify the message computations within the factor graph.

8 A message-passing algorithm

This section demonstrates the application of moment-matching to approximate $p(t|y)$ using a Gaussian distribution. Additionally, it outlines the implementation of the message-passing algorithm for computing the posterior distributions $p(s_1|y)$ and $p(s_2|y)$ between two players.

$$P(t|y) = \mu_6(t) \times \mu_7(t) = \mathcal{N}(t, \mu_1 - \mu_2; \sigma_1^2 + \sigma_2^2 + \sigma_3^2) \times \delta(y = \text{sign}(t)) \tag{9}$$
$$\approx \hat{q}(t) = \mathcal{N}(t; \mu_t, \sigma_t^2) \tag{10}$$

Using the above Gaussian distribution, message $\mu_8(t)$ can be approximated as follow:

$$\mu_8(t) = \frac{\hat{q}(t)}{\mu_6(t)} = \frac{\mathcal{N}(t; \mu_t, \sigma_t^2)}{\mathcal{N}(t; \mu_1 - \mu_2; \sigma_1^2 + \sigma_2^2 + \sigma_3^2)} \tag{11}$$
$$= \mathcal{N}\left(t; \frac{\mu_t(\sigma_1^2 + \sigma_2^2 + \sigma_3^2) - (\mu_1 - \mu_2)\sigma_t^2}{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 - \sigma_t^2}, \frac{(\sigma_1^2 + \sigma_2^2 + \sigma_3^2)\sigma_t^2}{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 - \sigma_t^2}\right) = \mathcal{N}\left(t; \mu_8, \sigma_8^2\right) \tag{12}$$

$$\mu_9(s1) = \int_{t,s_2} f_3(s_1, s_2, t)\mu_5(s_2)\mu_8(t)dtds_2 \tag{13}$$
$$= \int_{t,s_2} \mathcal{N}(t; s_1 - s_2, \sigma_3^2) \times \mathcal{N}(s_2; \mu_2, \sigma_2^2) \times \mathcal{N}(t; \mu_8, \sigma_8^2)dtds_2 \tag{14}$$
$$= \int_t \mathcal{N}(t; s_1 - \mu_2, \sigma_2^2 + \sigma_3^2) \times \mathcal{N}(t; \mu_8, \sigma_8^2)dt \quad \text{(corollary 2)} \tag{15}$$
$$= \int_t \mathcal{N}(s_1; \mu_2 + t, \sigma_2^2 + \sigma_3^2) \times \mathcal{N}(t; \mu_8, \sigma_8^2)dt \quad \text{(property 2)} \tag{16}$$
$$= \mathcal{N}(s1; \mu_2 + \mu_8, \sigma_2^2 + \sigma_3^2 + \sigma_8^2) \tag{17}$$
$$\mu_{10}(s2) = \int_{t,s_1} f_3(s_1, s_2, t)\mu_4(s_1)\mu_8(t)dtds_1 \tag{18}$$
$$= \int_{t,s_1} \mathcal{N}(t; s_1 - s_2, \sigma_3^2) \times \mathcal{N}(s_1; \mu_1, \sigma_1^2) \times \mathcal{N}(t; \mu_8, \sigma_8^2)dtds_1 \tag{19}$$
$$= \int_t \mathcal{N}(t; \mu_1 - s_2, \sigma_1^2 + \sigma_3^2) \times \mathcal{N}(t; \mu_8, \sigma_8^2)dt \quad \text{(corollary 2)} \tag{20}$$
$$= \int_t \mathcal{N}(s_2; \mu_1 - t, \sigma_1^2 + \sigma_3^2) \times \mathcal{N}(t; \mu_8, \sigma_8^2)dt \quad \text{(property 2)} \tag{21}$$
$$= \mathcal{N}(s1; \mu_1 - \mu_8, \sigma_1^2 + \sigma_3^2 + \sigma_8^2) \tag{22}$$

Finally, $p(s_1|y)$ and $p(s_2|y)$ can be computed as follows using Gaussian multiplication:

$$p(s_1|y) = \mu_1(s1)\mu_9(s_1) = \mathcal{N}(s_1, \frac{\mu_1(\sigma_2^2 + \sigma_3^2 + \sigma_8^2) + (\mu_2 + \mu_8)\sigma_1^2}{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_8^2}, \frac{\sigma_1^2 \cdot (\sigma_2^2 + \sigma_3^2 + \sigma_8^2)}{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_8^2})$$
$$p(s_2|y) = \mu_2(s2)\mu_{10}(s_2) = \mathcal{N}(s_2, \frac{\mu_2(\sigma_1^2 + \sigma_3^2 + \sigma_8^2) + (\mu_1 - \mu_8)\sigma_2^2}{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_8^2}, \frac{\sigma_2^2 \cdot (\sigma_1^2 + \sigma_3^2 + \sigma_8^2)}{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_8^2})$$

The outcomes of our message passing algorithm can be located in Figure5

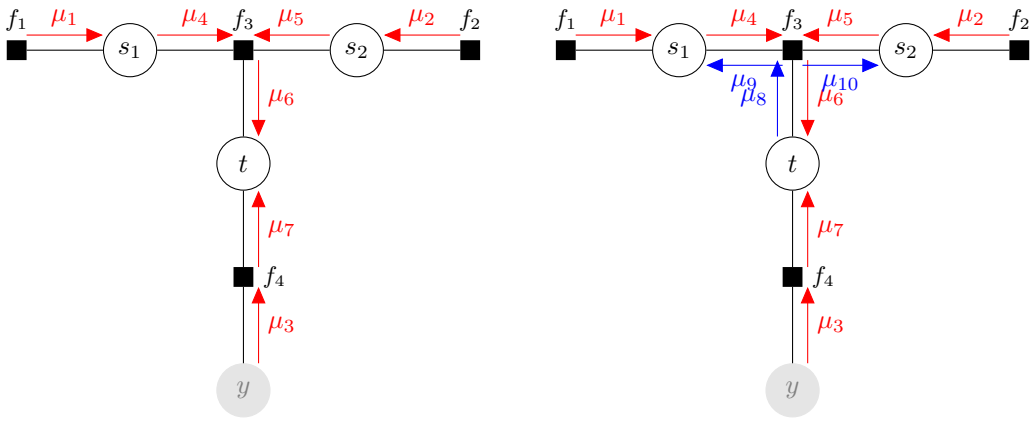

Figure 4: Factor graph representing the Bayesian network for the TrueSkill model. On the left, we demonstrate moment-matching to approximate $p(t|y)$ with a Gaussian distribution. On the right, we illustrate the message-passing algorithm for computing the posterior distribution of each of the two skills based on the outcome of a single game.

9 Our own data

We applied our TrueSkill implementation on the KHL 2018/2019 series [1]. Results from the ADF experiment can be found in table 3, and we obtained a prediction rate of $r = 0.615$ with the *one-step-ahead* prediction method outlined in Section 6.

[1] https://www.kaggle.com/datasets/vadimgladky/ice-hockey-khl-dataset

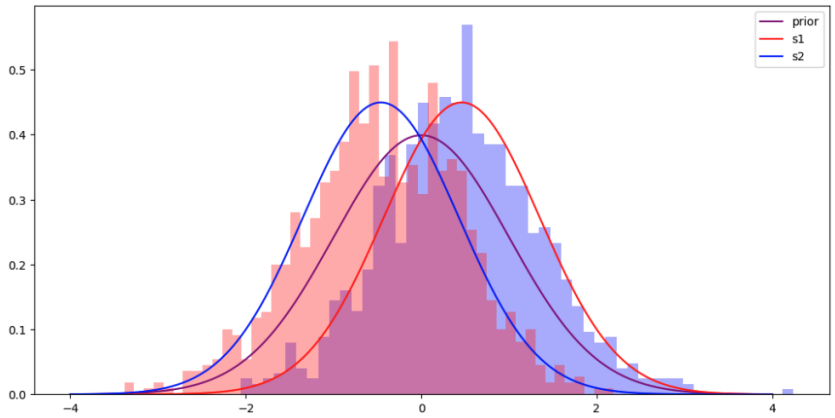

Figure 5: Posteriors $p(s_1|y), p(s_2|y)$ computed using message passing. In the same plot, include the histogram alongside the Gaussian approximation obtained from Gibbs sampling.

Table 3: Assumed Density Filtering experiment results on the KHL 2018/2019 series.

Team	Mean	Variance	Rank	Team	Mean	Variance	Rank
CSKA Moscow	.737	.016	1	B. Nur-Sultan	-.327	.016	10
SKA Petersburg	.496	.013	2	Salavat Ufa	-.335	.013	11
Yekaterinburg	.020	.020	3	Dyn. Moscow	-.368	.018	12
L. Yaroslavl	-.030	.010	4	Sochi	-.456	.017	13
Jokerit	-.087	.013	5	Niznekamsk	-.496	.022	14
M. Magnitogorsk	-.134	.016	6	N. Novgorod	-.503	.013	15
Avangard Omsk	-.159	.013	7	T. Chelyabinsk	-.518	.011	16
Bars Kazan	-.261	.014	8	A. Khabarovsk	-.556	.024	17
Sp. Moscow	-.317	.013	9	S. Novosibirsk	-.586	.025	18

Team	Mean	Variance	Rank
Cherepovets	-.611	.018	19
Din. Minsk	-.673	.014	20
Podolsk	-.684	.013	21
Dinamo Riga	-.760	.017	22
Kunlun	-.832	.014	23
Vladivostok	-.883	.016	24
HC Yugra	-.898	.049	25
Slavan Bratislava	-1.031	.021	26
Lada	-1.039	.029	27

10 Open-ended project extension

For the Serie A 2018 dataset, it is possible to induce prior knowledge before predicting the outcomes. Team performances in sports are often correlated between years, and this information can be leveraged to give the TrueSkill model better prior knowledge instead of giving every team the same initial parameters.

To do this, we considered the final rankings of Serie A in 2017 and gave each team a starting mean that corresponded to their performance that year. The top 17 teams were given initial variances of 5 and means ranging from 3-20, where the top team was given the highest value of 20, and the 17th ranked team was given the lowest value of 3. Serie A replaced the bottom three teams the following year, so we gave these new teams starting means of 0 but higher variances of 10. With this method, we were able to increase the prediction rate from $r = 0.64$ to $r = 0.73$.

References

[1] D. Vats and C. Knudson, "Revisiting the gelman-rubin diagnostic," 2020.