1RT705/1RT003 APML - Group Project(Team 26)

Yue WANG, Yuehua QIN, Jiayan YANG
Department of Information Technology
Uppsala University

1 Modeling

The Trueskill Bayesian model for one match between two players consists of four random variables, two Gaussian random variables s_1 and s_2 for the skills of the players, one Gaussian random variable t, with mean $s_1 - s_2$, for the outcome of the game, and one discrete random variable y for the game. There is no possibility of a draw between the players.

$$s_1 \sim \mathcal{N}(s_1; \mu_1, \sigma_1^2) \tag{1}$$
$$s_2 \sim \mathcal{N}(s_2; \mu_2, \sigma_2^2) \tag{2}$$
$$p(t|s_1, s_2) \sim \mathcal{N}(t; s_1 - s_2, \sigma_t^2) \tag{3}$$
$$p(y) = sign(t) \tag{4}$$

Here explain (4) as follows:

1. when t>0, the player 1 wins the game and it is represent by y=1;
2. when t<0, the player 1 loses the game and it is represented by y=-1.

There are 5 hyperparameters $\mu_1, \mu_2, \sigma_1^2, \sigma_2^2$ and σ_t^2 in the model to specify. Because the default Trueskill of a new player is assigned a mean skill of $\mu = 25$ and a skill uncertainty of $\sigma = \frac{25}{3}$, the 5 hyperparameters are defined as follows[1],

$$\mu_1 = \mu_2 = 25, \sigma_1^2 = \sigma_2^2 = \frac{25}{3}, \sigma_t^2 = \frac{25}{3}$$

2 Conditional independence

The conditional dependencies among the four variables are clear, based on the description of variables and model. The variables s_1 and s_2 are independent and can be denoted as s. The variable t is dependent on variables s, and variable y is dependent on variable t. The dependencies between s and y can be found out as below,

$$p(s, y \mid t) = \frac{p(s, t, y)}{p(t)} = \frac{p(s)p(t \mid s)p(y \mid t)}{p(t)} = p(s \mid t)p(y \mid t)$$

In this model, s and y are conditionally independent given t, written symbolically as: $s \perp y \mid t$.

3 Computing with the model

3.1 The full conditional distribution of the skills

Because s and y are conditionally independent given t with $s = \{s_1, s_2\}$, the full conditional distribution of the skill $p(s_1, s_2 \mid t, y) = p(s \mid t, y)$.

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

$$p(s \mid t) = \mathcal{N}(s; \mu_{s|t}, \Sigma_{s|t}) \tag{6}$$

where

$$\mu_{s|t} = \Sigma_{s|t}(\Sigma_s^{-1}\mu_s + M^T\Sigma_{t|s}^{-1}t)$$
$$\Sigma_{s|t} = (\Sigma_s^{-1} + M^T\Sigma_{t|s}^{-1}M)^{-1}$$

and

$$\mu_s = \begin{pmatrix}\mu_1 \\ \mu_2\end{pmatrix}, \Sigma_s = \begin{pmatrix}\sigma_1^2 & 0 \\ 0 & \sigma_2^2\end{pmatrix}, \Sigma_{t|s} = \sigma_t^2, M = \begin{pmatrix}1 & -1\end{pmatrix}$$

3.2 The full conditional distribution of the outcome

The full conditional distribution of the outcome can be written as below,

$$p(t \mid s_1, s_2, y) \propto p(y \mid t)p(t \mid s_1, s_2) \tag{7}$$

and it is a Truncated Normal Distribution. The Truncated Normal Distribution is a general normal probability distribution function by specifying parameters $\mu = s_1 - s_2$ and $\sigma^2 = \sigma_t^2$, and a truncation range (a, b). There are two different range depending on the different value of y:

$$if\ y = 1,\ a = 0 \text{ and } b = \infty$$
$$if\ y = -1,\ a = -\infty \text{ and } b = 0$$

3.3 The marginal probability that Player 1 wins the game

As defined of the discrete random variable y in section 1, the marginal probability that Player 1 win the game $p(y = 1)$ is the equivalent of $p(t > 0)$. First, calculate the marginal probability of variable t. The distributions of $p(ts_1, s_2)$ and $p(s_1, s_2)$ are known, so $p(t)$ can be calculated by using Corollary 2 (Affine transformation - Marginalization).

$$p(t) = \mathcal{N}(t, \mu_t, \Sigma_t) \tag{8}$$

where

$$\mu_t = \mu_1 - \mu_2, \Sigma_t = \sigma_t^2 + \sigma_1^2 + \sigma_2^2$$

Then $p(t > 0)$ is a cumulative distribution function of $p(t)$ and can be found.

4 Bayesian Network

Figure 1 is a Bayesian Network that represents the 4 variables s_1, s_2, t, y in the model and their conditional dependencies.

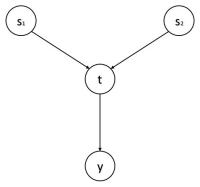

Figure 1: The Bayesian Network

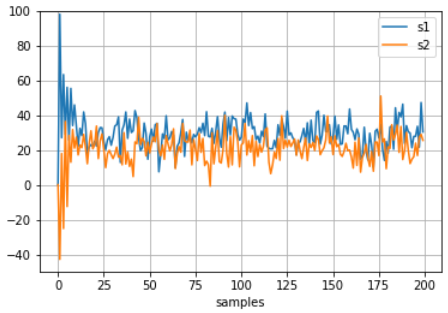

Figure 2: Estimate based on Gibbs Sampling

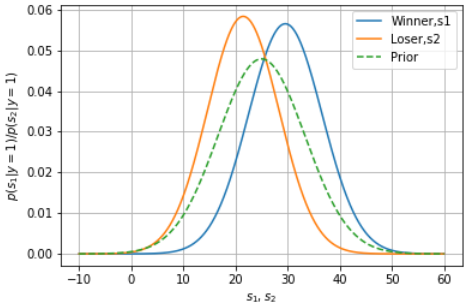

Figure 3: Gaussian approx. of posterior distribution of skills under the assumption $y = 1$

5 A first Gibbs sampler

- Figure 2 displays 200 samples of the posterior distribution $p(s_1, s_2|y = 1)$ generated by Gibbs Sampler based on the parameters in Q1 such as $\mu_1 = \mu_2 = 25$, $\sigma_1 = \sigma_2 = \sigma_t = \frac{25}{3}$ and $y = 1$. We also set the outcome for initial samples, i.e, $t_0 = 200$, indicating the player 1 is far ahead of player 2 and wins the first match. The distribution has a sharp decrease at the beginning and converges after $\sim$20 samples, although fluctuating during the observation, so we decide a reasonable burn-in can be 20. The figure has no apparent change in the second experiment and we can say it is a good choice for this case.

- Figure 3 demonstrates the function that uses the mean and covariance of the samples drawn by Gibbs sampler to fit a Gaussian distribution for posterior distribution. It is observed that μ_1 is larger than μ_2, which is in line with our assumption that s_1 won the game, i.e., $y = 1$. No obvious differences are observed in terms of standard deviation.

- We try 100, 200, 500 and 1000 samples and set burn-in as 20. The resulting plots with histogram using post-burn-in samples and corresponding fitted Gaussian posterior are shown in Figure 4. As expected, more samples consume more time but run time is not proportional to the number of samples, e.g., 1000 samples take more than twice as long as 500 samples but they have similar patterns in terms of both histograms and fitted Gaussian distribution, The cases of 100 and 200 samples have more discrete patterns and they have smaller mean. All can be regarded as a good choice as the resulting patterns and run-time are quite similar but we consider 500 as the best if we have to make a choice, because we only have 380 matches in the following case and it requires relatively less time comparing with 1000 samples.

- The comparison can be found in Figure 3. As observed, the posteriors of both players have smaller variance, comparing with prior, but player 1 has a larger mean while player 2 has a smaller one, which is based on the observation or assumption that $y = 1$, so player 1 has more possibility to win in the posterior distribution.

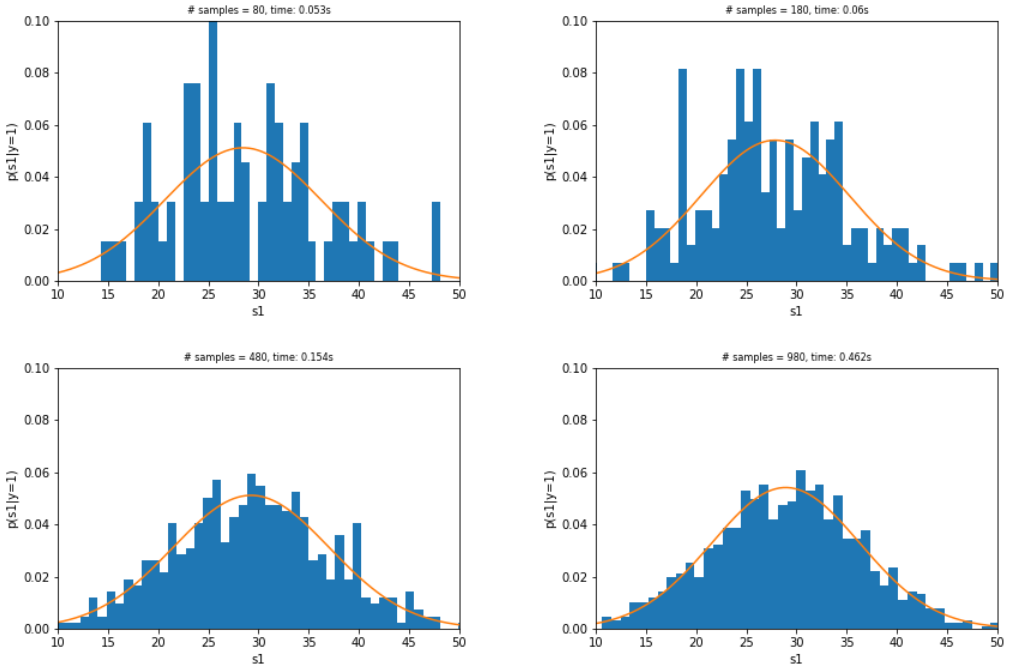

Figure 4: Histogram of post-burn-in samples and fitted Gaussian posterior for different # samples

6 Assumed Density Filtering

- We set the same initial condition as the previous section and use ADF with Gibbs sampling to process the matches in the given dataset. The matches with draw as outcome is not considered. The final ranking is displayed as Table 1a, but it changes each execution. The mean of skill, skill uncertainty(variance) and conservative estimate[1] $\mu - 3 * \sigma$ are computed for each team. In particular, the variance quantities a team's skill away from its mean, and a larger variance indicates the high possibility of a less-skilled team wins. The conservative estimate for skill considered both average and uncertainty of a team's skill, and a more-skilled team always enjoys a larger conservative estimate in this case.

- We shuffle the order and re-run the experiment and the rank becomes different as Table 1b, which means ADF is sensitive to the order of data as the posterior distribution relies on current observation, e.g., winning at the start will affect a estimate for team's skill more than winning at the middle or end.

7 Using the model for predictions

The prediction relies on ADF model, where a team's skill is recomputed and updated every time it participates in a match and is used to predict a new match using Gibbs sampling. The prediction is then compared with the actual result and counted for the prediction rate. Disregarding the match with the outcome of draw, we have 272 matches, and the model gives 64.34% and 65.07% as prediction rate using averaged skill and conservative estimated skill for computation, respectively, which is better than random guessing if we assume that win and lose for each team happen in an equal possibility of 50%. If match with the outcome of draw is counted, the model gives 46.05% and 46.58% as correct prediction for averaged skill and conservative skill, and it is also better than a random guess for the assumption that win, lose and draw are independent events and each takes $\frac{1}{3}$ possibility to happen.

Table 1: Mean, standard deviation and conservative estimate for each team using ADF

Team	mean	std	skillEstimate	rank
Juventus	34.75	2.62	26.9	1.0
Napoli	32.08	2.66	24.1	2.0
Milan	31.87	2.07	25.67	3.0
Torino	30.11	2.69	22.04	4.0
Inter	30.07	2.23	23.37	5.0
Atalanta	29.57	2.22	22.92	6.0
Roma	28.16	2.17	21.66	7.0
Lazio	26.67	2.06	20.49	8.0
Sampdoria	24.89	2.01	18.85	9.0
Bologna	23.64	2.28	16.79	10.0
Spal	23.58	2.4	16.38	11.0
Genoa	23.5	2.24	16.78	12.0
Fiorentina	23.23	2.2	16.63	13.0
Sassuolo	22.82	2.47	15.42	14.0
Empoli	22.51	2.04	16.4	15.0
Cagliari	22.19	2.3	15.28	16.0
Udinese	22.17	2.33	15.18	17.0
Parma	21.87	2.04	15.76	18.0
Frosinone	16.9	2.8	8.49	19.0
Chievo	12.66	2.77	4.35	20.0

(a) SerieA order

Team	mean	std	skillEstimate	rank
Juventus	34.44	2.12	28.07	1.0
Napoli	31.21	2.06	25.04	2.0
Milan	29.69	2.01	23.65	3.0
Roma	29.63	2.18	23.08	4.0
Inter	29.41	2.43	22.11	5.0
Atalanta	29.17	2.28	22.34	6.0
Torino	29.12	2.77	20.82	7.0
Lazio	26.21	2.07	19.98	8.0
Sampdoria	25.31	2.13	18.91	9.0
Parma	23.39	2.36	16.31	10.0
Sassuolo	23.36	2.52	15.79	11.0
Fiorentina	23.23	2.74	15.02	12.0
Bologna	23.05	2.29	16.18	13.0
Udinese	22.1	2.56	14.42	14.0
Spal	21.48	1.92	15.73	15.0
Empoli	21.28	1.93	15.48	16.0
Cagliari	20.94	1.99	14.97	17.0
Genoa	19.93	2.49	12.46	18.0
Frosinone	16.11	2.65	8.15	19.0
Chievo	12.72	3.21	3.09	20.0

(b) Random order

8 Factor graph

Figure 5 is a factor graph for the model.

$$f_1(s_1) = \mathcal{N}(s_1; \mu_1, \sigma_1^2) \tag{9}$$
$$f_2(s_2) = \mathcal{N}(s_2; \mu_2, \sigma_2^2) \tag{10}$$
$$f_{sw}(s, w) = \delta(w - (s_1 - s_2)) \tag{11}$$
$$f_{wt}(w, t) = \mathcal{N}(t; w, \sigma_t^2) \tag{12}$$
$$f_{ty}(t, y) = \delta(y - sign(t)) \tag{13}$$

$$\mu_{y \to f_{ty}}(y) = 1, \mu_{f_{ty} \to t}(t) = \sum_y f_{ty} \mu_{y \to f_{ty}}(y) \tag{14}$$
$$\mu_{f_1 \to s_1}(s_1) = f_1, \mu_{f_2 \to s_2}(s_2) = f_2 \tag{15}$$
$$\mu_{s_1 \to f_s w}(s_1) = \mu_{f_1 \to s_1}(s_1), \mu_{s_2 \to f_s w}(s_2) = \mu_{f_2 \to s_2}(s_2) \tag{16}$$
$$\mu_{f_{sw} \to w}(w) = \int\int f_{sw} \mu_{f_1} \mu_{f_2} ds_1 ds_2 \tag{17}$$
$$\mu_{w \to f_{wt}}(w) = \mu_{f_{sw}}, \mu_{f_{wt} \to t}(t) = \int f_{wt} \mu_w dw \tag{18}$$

9 A message-passing algorithm

Figure 6 compares the posterior distribution generated by message-passing algorithm with that of Gibbs sampling that shares the same setting as previous, i.e., $m_1 = m_2 = 25, s_1 = s_2 = s_t = \frac{25}{3}, y_0 = 1$. The burn-in and samples are set as 20 and 500, respectively, and only those post-burn-in samples are used to plot. It can be observed that two distributions almost overlap, indicating that these two algorithms give a similar posterior distribution for both players.

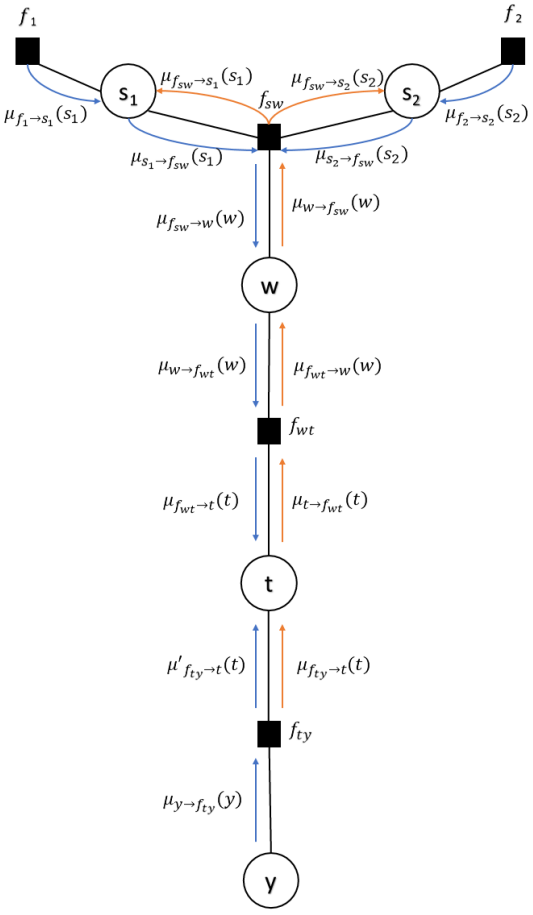

Figure 5: The factor graph with messages

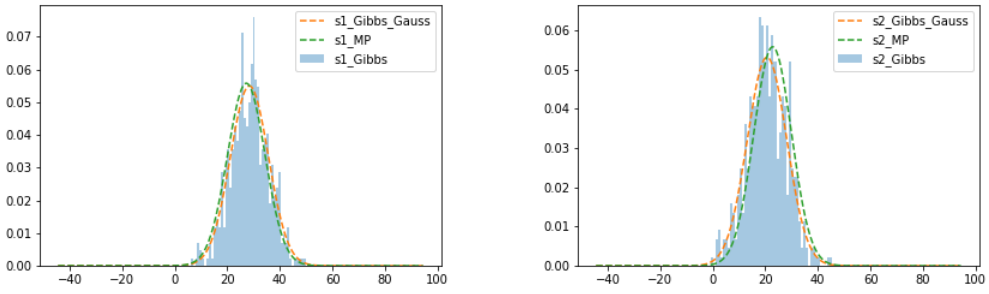

Figure 6: Posterior distribution produced by Gibbs sampling and message-passing algorithm

10 You own data

Based on the previous Trueskill method, it was applied to test the dataset of an inaugural season of Overwatch Esports League, which is comprised of 12 city-based teams. All data were obtained from the Overwatch league official website overwatchleague.com. The pre-processing is to clean up the data in order to fit the same structure with the previous dataset serieA, which would be compatible with the TrueSkill functions. Every match has two teams 'team1' and 'team2' with their final results 'score1' and 'score2'. No more significant processing was necessary. The difference from serieA is that there are no match results in a draw.

By testing it on the Trueskill methods we have developed, the ranking shows a great similarity with the official ranking but not all the same. The ranking accuracy rate of the twelve teams has reached 83%. For instance, the top3 teams in results are 'New York Excelsior', 'Los Angeles Valiant' and 'Boston Uprising', which are also the top3 teams at the end of the season. We applied mean and estimate for prediction and the correct guess rates are 68.00% and 65.60%, respectively. For the higher correction in this dataset, there are two reasonable guesses. Firstly there is no draw in the results. It helps to better understand the skills of the two teams. Secondly, the general standard deviations are smaller which means the scores are closer to the teams' true skills.

Table 2: Mean, standard deviation, conservative estimate and true rank of own dataset

Team	mean	std	skillEstimate	rank	True Rank
New York Excelsior	34.56	1.91	28.82	1.0	New York Excelsior
Los Angeles Valiant	30.33	1.47	25.91	2.0	Los Angeles Valiant
Boston Uprising	29.51	1.85	23.96	3.0	Boston Uprising
Los Angeles Gladiators	28.19	1.78	22.85	4.0	Los Angeles Gladiators
Seoul Dynasty	27.9	1.63	23.02	5.0	London Spitfire
London Spitfire	27.86	1.78	22.5	6.0	Philadelphia Fusion
Philadelphia Fusion	27.41	2.0	21.42	7.0	Houston Outlaws
Houston Outlaws	25.83	1.6	21.02	8.0	Seoul Dynast
San Francisco Shock	24.48	2.06	18.3	9.0	San Francisco Shock
Dallas Fuel	21.66	1.94	15.83	10.0	Dallas Fuel
Florida Mayhem	17.68	2.04	11.57	11.0	Florida Mayhem
Shanghai Dragons	7.7	2.32	0.73	12.0	Shanghai Dragons

11 Open-ended project extension

In this section, the algorithm was improved by tuning hyperparameters to get better model performance. The grid-search was applied to find the best σ_t value. The linear interval values were set with $N = 20000$ in the range of $\sigma_t = [0.1, 10]$. As the Figure 7 shows that the highest correct prediction rate appears when σ_t is 0.3.

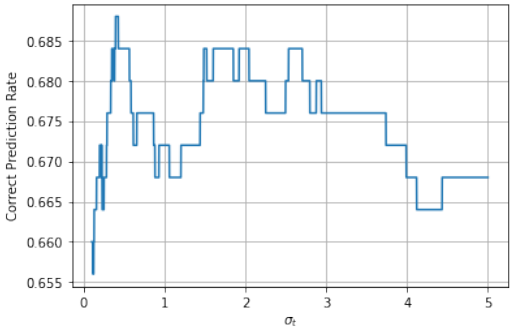

Figure 7: Grid search results of finding a σ_t to get a higher prediction rate

References

[1] Minka, T. Zaykov, Y Tseran, H.(2005) *"TrueSkill Ranking System"*. Available through https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/!overview , accessed Sep 2021.

[2] C.Bishop. *Pattern Recognition And Machine Learning.*, Springer, 2006.