# 1RT705: Group 7651 (4 members)

**Anonymous**
Uppsala University

## 1 Introduction

Efficient algorithms to rank team or players are important in order to make the game more interesting by matching teams with similar skill against each others. One way to rank players are with the TrueSkill$^{TM}$ [2] algorithm developed by Microsoft. This algorithm utilises Bayesian probabilistic and inference to try and match players given an assumption about their skill. In this project the TrueSkill$^{TM}$ algorithm have been implemented and tested on data from the Italian football league *Serie A*, in order to try and rank the teams after their inferred skill.

## 2 Assignments

### 2.1 Q1 - Modeling

The model consists of four random variables - three Gaussian random variables $s_1$, $s_2$ and $t$ (skills of the players and the outcome of one match between the two) - and one discrete variable $y$. More precisely, the distribution of the continuous variables are

$$\begin{cases} p(s_1) = \mathcal{N}(s_1; \mu_1, \sigma_1^2), \\ p(s_2) = \mathcal{N}(s_2; \mu_2, \sigma_2^2), \\ p(t|s_1, s_2) = \mathcal{N}(t; s_1 - s_2, \sigma_t^2). \end{cases} \tag{1}$$

The discrete variable $y$ is given by $y = \text{sign}(t)$. Thus there are five hyperparameters in the model that need to be specified: $\mu_1$, $\mu_2$, $\sigma_1$, $\sigma_2$ and $\sigma_t$.

### 2.2 Q2 - Computing with the model

Using the results on Gaussian random variables we want to compute the full conditional distributions $p(s_1, s_2|t, y)$ and $p(t|s_1, s_2, y)$. Note that $p(s_1, s_2|t, y) = p(s_1, s_2|t)$, since once $t$ is observed then $y$ is known.

Let $\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$ and $p(\mathbf{s}) = \mathcal{N}(\mathbf{s}; \mu_\mathbf{s}, \Sigma_\mathbf{s})$, where $\mu_\mathbf{s} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$, $\Sigma_\mathbf{s} = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}$. Then we can write $p(t|s_1, s_2) = p(t|\mathbf{s}) = \mathcal{N}(t; M\mathbf{s}, \Sigma_{t|\mathbf{s}})$, where $M = (1, -1)$ and $\Sigma_{t|\mathbf{s}} = \sigma_t^2$. The full conditional distribution $p(\mathbf{s}|t)$ is given by

$$p(\mathbf{s}|t) = \mathcal{N}(\mathbf{s}; \mu_{\mathbf{s}|\mathbf{t}}, \Sigma_{\mathbf{s}|\mathbf{t}}). \tag{2}$$

The expressions of $\mu_{\mathbf{s}|\mathbf{t}}$ and $\Sigma_{\mathbf{s}|\mathbf{t}}$ can be found in the appendix.

To obtain $p(t|s_1, s_2, y)$ we use the fact that if $y$ is observed then the sign of $t$ is known. The result is a truncated Gaussian - whenever the sign of $t$ is not $y$ then $p(t|s_1, s_2, y)$ must be zero. This can be expressed using the Dirac delta function as

$$p(t|s_1, s_2, y) \propto p(t|, s_1, s_2)\delta\left(y - \text{sign}(t)\right). \tag{3}$$

Finally the marginal probability that player 1 wins, i.e. $p(y = 1)$, can be calculated by observing the Bayesian network and marginalizing the joint distribution.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

$$p(y=1) = \int_{s_1}\int_{s_2}\int_t p(s_1, s_2, t, y=1) ds_1 ds_2 dt = \tag{4}$$

$$\int_{s_1} p(s_1) ds_1 \int_{s_2} p(s_2) \int_0^\infty p(t|s_1, s_2) dt ds_2 ds_1 = \tag{5}$$

$$\int_0^\infty N(t; \mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2 + \sigma_t^2) \tag{6}$$

### 2.3 Q3 - Bayesian Network

Observing the Bayesian network in Figure 1 one can find two sets of conditionally independent variables. The first set is $(s_1 \perp\!\!\!\perp y) \mid t$, and by symmetry $(s_2 \perp\!\!\!\perp y) \mid t$, from the head-to-tail rule. A second set is $(s_1 \perp\!\!\!\perp s_2) \mid \varnothing$, from head-to-head rule.

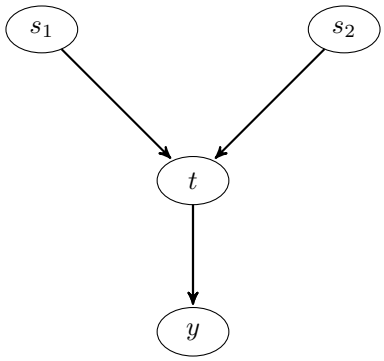



Figure 1: Bayesian network model

### 2.4 Q4 - A first Gibbs Sampler

To estimate the skill given the result from a match a sampler built on the Gibbs sampler algorithm was implemented which targets the posterior distribution $p(s_1, s_2|t)$. This is done by alternating between sampling the result $p(t|s_1, s_2)$ and sampling the skill $p(s_1, s_2|t)$. In the sampling we assume that the initial skill of the teams are modeled by the same Gaussian distributions, $\mu_1 = \mu_2$ and $\sigma_1 = \sigma_2$.

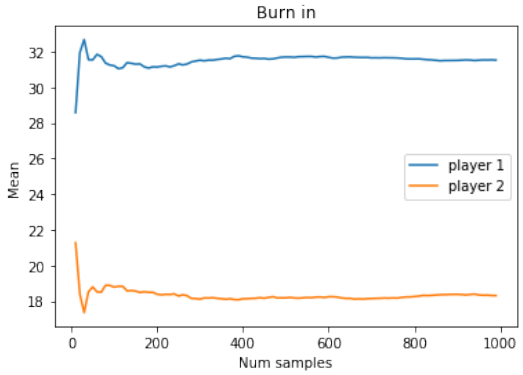



Figure 2: Burn in for the Gibbs sampler

Figure 2 shows the burn-in period, i.e. the amount of samples needed for the mean of skills to converge. The mean is based on the number of samples that are used and it takes roughly 200-300 samples before it converges. Therefore a burn-in of 200 samples is used in future calculations.

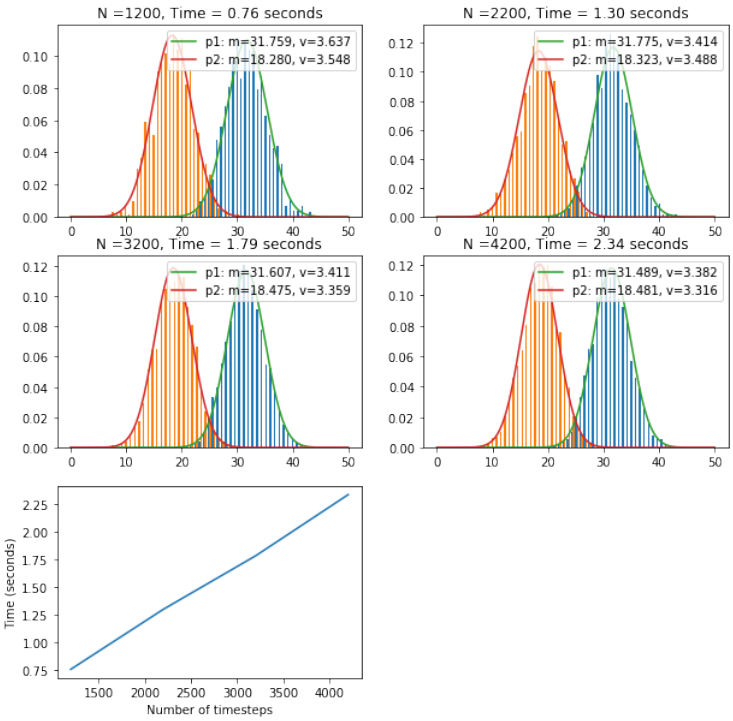



Figure 3: Histogram and fitted Gaussian for different number of samples and computational time as a function of the number of samples used

In Figure 3 the posterior distributions of the skill is shown for four different number of samples used in the Gibbs sampling. The computational time increases linearly which was expected and from these graphs 2200 samples are chosen for future use in the model. Each update of skill takes 1.3 seconds which means that updating the skill 300 times will take about 6 minutes. From the graphs one can see that there is not a huge improvement in accuracy when increasing the number of samples from 2200 to 3200. As the burn-in was set to 200, 2200 samples will result in 2000 usable samples.

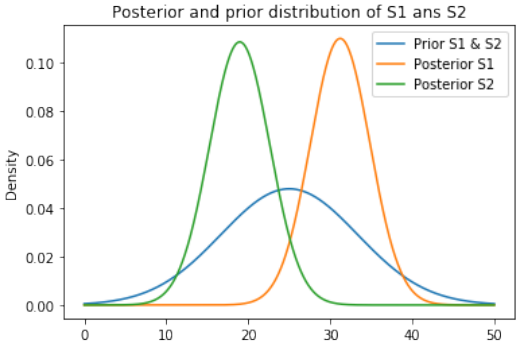
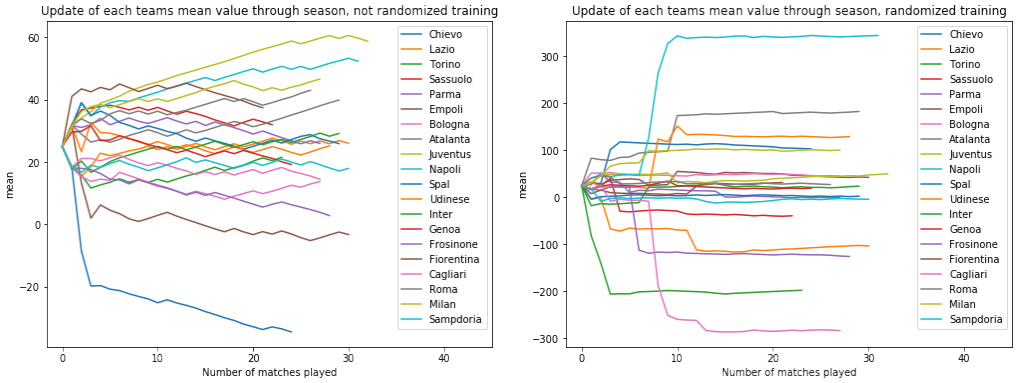



Figure 4: Prior and Posterior distribution of the skill when y=1

In figure 4, the updated posterior of $p(s_1|y = 1)$ is compared with the prior $p(s_1)$. The posterior has a higher mean and a lower variance as player 1 won the match. The same logic follows for player 2 the match was lost and therefore will have a lower mean. The variance has decreased for both posterior distributions as we are more sure about the skill of the players after the match than before.

### 2.5 Q5 - Assumed Density Filtering (ADF)

In order to rank football teams according to our belief in their skill we use assumed density filtering, ADF. Using ADF means that the skill of the teams are estimated in one match using Gibbs sampling, receiving a posterior estimate, this estimate is then used as the prior the next time the team plays.




(a) The mean of the teams during the season. (b) The mean while the match order are randomized.

Figure 5: The mean of all the teams during a whole season. Both the normal season of matches and the randomized order.

Figure 5a shows the progress of each team through the season in terms of its mean value and figure 5b shows the progress of each team when the order the matches are played is randomized.

Comparing the result of the final ranking of the season and when the matches are drawn randomly, differ a lot. The skill for the teams changes more and gives a more uncertain result, specially when teams who hasn't played the same amount of matches play against each other.

### 2.6 Q6 - Using the model for predictions

A basic prediction function to try and predict who will win between two teams is to simply draw X samples from the two normal distributions corresponding to the skill of the two teams $p(s_{1,2}) = \mathcal{N}(s_{1,2;}\mu_{1,2}, \sigma^2_{1,2})$ and calculate which teams skill is highest the majority of the time, 1 if team 1 has highest skill in the majority of samples, else -1. This simple prediction function have a prediction rate of $0.625 = 62.5\%$. If one were to randomly guess without any prior knowledge, i.e. randomly assign 1 or -1 to 380 games, one would expect the prediction rate to be around $0.5$, which is confirmed by drawing the skill for the two teams from the same normal distribution. Using the randomly shuffled data instead the prediction rate becomes $0.606 \approx 61\%$ while randomly guessing has a prediction rate of $0.477 \approx 48\%$, so still around $50\%$

### 2.7 Q7 - Factor graph

The factorization of the joint distribution of the Bayesian network is shown in the factor graph in Figure 6. An additional variable node $w$ is introduced which is simply the difference between $s_1$ and $s_2$, $w = s_1 - s_2$. We are interested in updating $p(s_1)$ and $p(s_2)$ given an observation $y = y_{obs}$. The resulting updates are $p(s_{1,2}) \propto \mu_{f_{1,2} \to s_{1,2}}(s_{1,2}) \hat{\mu}_{f_3 \to s_{1,2}}(s_{1,2})$, where moment matching is used at variable node $t$. The explicit forms of the messages can be viewed in the appendix.

### 2.8 Q8 - A message-passing algorithm

The result of the implemented message-passing algorithm based on the messages in Figure 6 (see appendix) tested on one game in which player 1 wins is shown in Figure 7.

The posterior of the message-passing algorithm after only one cylce is not close to the posterior of the Gibbs sampler, seen in Figure 7a. However, with relatively few additional cycles the results become more similar. Figure 7b shows the posterior distribution using message-passing after 20 cycles compared to the Gibbs sampler. At the same time the variance using message-passing decreases

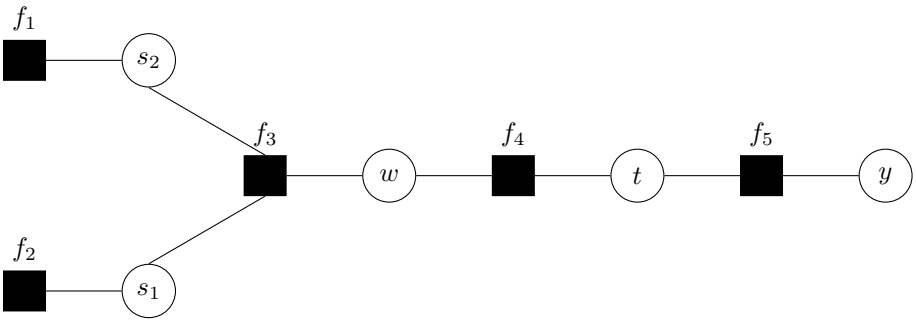
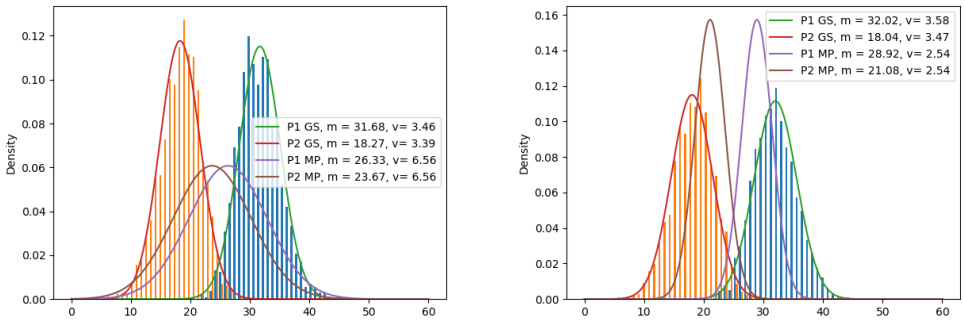



Figure 6: Factor graph of the Bayesian network in Figure 1




(a) The mean and standard deviation using message passing after one cycle compared to using Gibbs sampling.

(b) The mean and standard deviation using message passing after 20 cycles compared to using Gibbs sampling

Figure 7: The mean and standard deviation using Gibbs sampling, GS, compared to using either one or twenty cycles of message passing, MP. When computing several cycles in the message-passing the marginals of $s_1$ and $s_2$ from the previous cycle are used as the new messages from factor $f_1$ and $f_2$

quicker than the mean converges. This difference could be due to the need to approximate the truncated Gaussian, which is not needed using the Gibbs sampler. Still, using message passing for doing the one step ahead prediction resulted in a prediction rate of around $0.62$, which is almost the same as what was achieved with the Gibbs sampler.

### 2.9 Q9 - own data - North America League of Legends Championship

As an extension to this project, the Trueskill model was tested on a dataset from Kaggle [1] with results from the North American League of Legends championship. The dataset contains detailed matchinformation but in this application the result of each match was the only parameter used in the model. In League of Legends draws are not possible. The result is shown in figure 8, where some teams plays more games due to the relegation and promotion system in the end of the season.

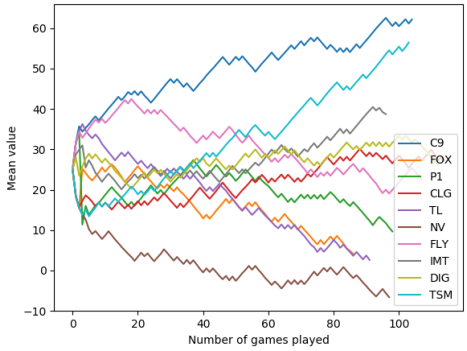



Figure 8: Mean value of teams in North America League of Legends Championship after 2017 season

### 2.10 Q10 - Extension to the project

One way to extend the model is to be able to handle draws. In the original implemented model, section 2.5, we ignored matches resulting in a draw. Instead of ignoring this we can update the standard deviation and mean for each team as the average mean and standard deviation between two games, one where they win and one where they lose. Using this new model resulted in a prediction rate of $0.66$ when trying to predict who would win or loose. The resulting ranking using this new model is shown in Figure 9.

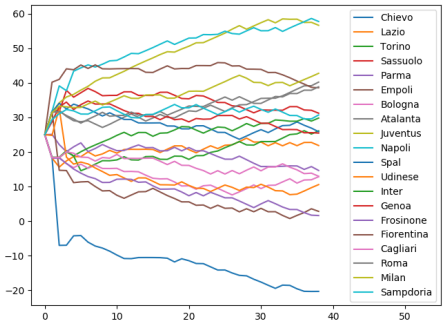



Figure 9: Resulting ranking after extending model to including draws.

Looking at the new result and comparing it with the scoreboard at the end of the season Juventus and Napoli was indeed the two best teams just as our ranking shows. Then there was a gap to the rest of the teams which this new model also shows. However Juventus finished ahead of Napoli while our model shows them having a lower skill. The original model captures the difference between Juventus and Napoli better while the new model seem to be able to better rank the teams closer to the average skill level.

## 3 Conclusion

Using the Gibbs sampler on the Bayesian network to estimate the distribution of the skills is somewhat useful. It allowed us to predict who would win the a match between two teams with a higher prediction rate compared to randomly guessing. Using message propagation led to a similar prediction rate $> 0.6$. This can be compared with random guessing which had a prediction rate of around $0.5$. Changing the model to include draws changed the final ranking a bit but also led to a higher prediction rate, $0.66$, when trying to predict which team would win.

## References


[1] Kaggle. https://www.kaggle.com/chuckephron/leagueoflegends/download. issue date: 2018.

[2] Microsoft. https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/. issue date: November 18, 2005.

# Appendix

### Q.2 Multivariate mean and co-variance matrix

$\mu_{\mathbf{s}|\mathbf{t}}$ and $\Sigma_{\mathbf{s}|\mathbf{t}}$ in equation (2) are given by

$$\mu_{\mathbf{s}|\mathbf{t}} = \frac{1}{\left(\frac{1}{\sigma_1^2}+\frac{1}{\sigma_t^2}\right)\left(\frac{1}{\sigma_2^2}+\frac{1}{\sigma_t^2}\right)-\frac{1}{\sigma_t^4}}\begin{pmatrix}\left(\frac{1}{\sigma_2^2}+\frac{1}{\sigma_t^2}\right)\left(\frac{\mu_1}{\sigma_1^2}+\frac{t}{\sigma_t^2}\right)+\frac{1}{\sigma_t^2}\left(\frac{\mu_2}{\sigma_2^2}-\frac{t}{\sigma_t^2}\right)\\ \frac{1}{\sigma_t^2}\left(\frac{\mu_1}{\sigma_1^2}+\frac{t}{\sigma_t^2}\right)+\left(\frac{1}{\sigma_2^2}+\frac{1}{\sigma_t^2}\right)\left(\frac{\mu_2}{\sigma_2^2}-\frac{t}{\sigma_t^2}\right)\end{pmatrix}, \tag{7}$$

and

$$\Sigma_{\mathbf{s}|\mathbf{t}} = \frac{1}{\left(\frac{1}{\sigma_1^2}+\frac{1}{\sigma_t^2}\right)\left(\frac{1}{\sigma_2^2}+\frac{1}{\sigma_t^2}\right)-\frac{1}{\sigma_t^4}}\begin{pmatrix}\frac{1}{\sigma_2^2}+\frac{1}{\sigma_t^2} & \frac{1}{\sigma_t^2}\\ \frac{a}{\sigma_t^2} & \frac{1}{\sigma_1^2}+\frac{1}{\sigma_t^2}\end{pmatrix}. \tag{8}$$

### Q.7 Message passing

The factors are in Figure 6 are

$$\begin{cases} f_1 = \mathcal{N}(s_1;\mu_1,\sigma_1^2),\\ f_2 = \mathcal{N}(s_2;\mu_2,\sigma_2^2),\\ f_3 = \delta(w-(s_1-s_2)),\\ f_4 = \mathcal{N}(t;s_1-s_2,\sigma_t^2),\\ f_5 = \delta(y-\text{sign}(t)). \end{cases} \tag{9}$$

The explicit form of the messages transmitting from the left to the variable node $t$ are

$$\mu_{f_1\to s_1}(s_1) = \mu_{s_1\to f_3}(s_1) = \mathcal{N}(s_1;\mu_1,\sigma_1^2), \tag{10}$$

$$\mu_{f_2\to s_2}(s_2) = \mu_{s_2\to f_3}(s_2) = \mathcal{N}(s_2;\mu_2,\sigma_2^2), \tag{11}$$

$$\mu_{f_3\to w}(w) = \mu_{w\to f_4}(w) = \mathcal{N}(w;\mu_1-\mu_2,\sigma_1^2+\sigma_2^2), \tag{12}$$

$$\mu_{f_4\to t}(t) = \mathcal{N}(t;\mu_1-\mu_2,\sigma_1^2+\sigma_2^2+\sigma_t^2). \tag{13}$$

The explicit form of the messages transmitting from the right to $t$ are

$$\mu_{y\to f_5}(y) = \mathbb{1}_{y_{obs}}(y), \tag{14}$$

$$\mu_{f_5\to t}(t) = \delta(y_{obs}-\text{sign}(t)). \tag{15}$$

At the variable node $t$ we use moment matching to approximate the truncated Gaussian message with a Gaussian message. The remaining messages transmitting from $f_5$ to $s_1$ and $s_2$ becomes

$$\hat{\mu}_{f_5\to t}(t) = \hat{\mu}_{t\to f_4}(t) = \frac{\hat{p}(t|y=y_{obs})}{\mu_{f_4\to t}(t)} = \mathcal{N}(t;\mu_d,\sigma_d^2), \tag{16}$$

$$\hat{\mu}_{f_4\to w}(w) = \hat{\mu}_{w\to f_3}(w) = \mathcal{N}(t;\mu_d,\sigma_d^2+\sigma_t^2), \tag{17}$$

$$\hat{\mu}_{f_3\to s_1}(s_1) = \mathcal{N}(s_1;\mu_2+\mu_d,\sigma_2^2+\sigma_d^2+\sigma_t^2), \tag{18}$$

$$\hat{\mu}_{f_3\to s_2}(s_2) = \mathcal{N}(s_2;\mu_1-\mu_d,\sigma_1^2+\sigma_d^2+\sigma_t^2). \tag{19}$$