Advanced Statistical Machine Learning
Project Report: The "True skill" of the serie A

Jacobson, Oscar
Oscar.Jacobson.9201@student.uu.se

Englund, Markus
Markus.Englund.6963@student.uu.se

December 29, 2022

Introduction

In the research paper [Herbrich et al.(2007)Herbrich, Minka, and Graepel] by Microsoft, the $TrueSkill^{TM}$ model is presented as a skill rating system for matches between players in any game. *TrueSkill* is built as an extension of the zero sum, ELO rating system, popularized by the International Chess Federation FIDE in 1970 to rank the skill of all competitive chess players. For Microsoft, players being matched against others of equal skills in an online setting is of crucial importance and a big part of making online gaming fun.

The *TrueSkill* model uses Gaussian random variables which can be updated after any performance from a player. By using machine learning this report aims to explain and implement the building blocks required to emulate the *TrueSkill* rating system. This will then be evaluated on data from the Italian top division of football, the Serie A 2018/2019 and from the candidates Chess tournament from 2022.

Formulating the trueskill bayesian model

The TrueSkill model uses the final standings in any match to determine the skill of players involved in that match. The predicted skill of a player is assumed to be a Gaussian distribution with their expected skill as the mean value and the variance reflecting the uncertainty of performance from that player. This way a more skilled player is always assumed to win but the chance of an upset is clearly defined and can be modeled. After every match the skill of involved players is updated by Bayesian inference given the outcome of the game. In this project a two player model is implemented to reflect the two teams in a game of football.

The model consists of the outcome y, two Gaussian random variables s_1 and s_2, representing the skills of the respective teams and a Gaussian random variable t, with mean $s_1 - s_2$ for the outcome of the game and one discrete random variable y = *sign(t)* for the result of the game. As of this part of the project the possibility of a draw is not considered.

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

The model is the joint distribution of s_1, s_2, t and y. Where t is dependent of s_1 and s_2 and y being dependent of t.

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

where the distributions of $\boldsymbol{s}$ is given by:

$$p(\mathbf{s}) \sim \mathcal{N}(\mathbf{s}; \mu_s, \Sigma_s) = \mathcal{N}\left(\begin{bmatrix} s_1 \\ s_2 \end{bmatrix}; \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}\right) \tag{3}$$

$$\begin{aligned} p(t|\mathbf{s}) \sim \mathcal{N}(t; A\mathbf{s}, \Sigma_{t|s}) = \mathcal{N}(t; s_1 - s_2, \sigma_{t|\mathbf{s}}^2) \\ A = [1, -1] \end{aligned} \tag{4}$$

$$p(y|t) \sim p(y = sign(t)) \tag{5}$$

Which gives the model five unknown hyperparameters: the variables $\mu_1, \mu_2, \sigma_1, \sigma_2$ and $\sigma_{t|s}$, to be set for the model.

The prediction and update of skills in this project makes use of Bayesian inference where we are searching for the posterior distribution of the skills. This posterior distribution is the product of our prior and likelihood distribution and is visualized in [Moser(2010)] P.17. Our prior is here given by the multivariate Gaussian 3 which is the product of $p(s_1)$ and $p(s_2)$. Our likelihood is also given similarly by the product of 4 and 5 which will turn out to be a truncated Gaussian and represent the likelihood for a particular outcome of a game given our set hypothesis.

To use these probabilities for computations we must firstly make sure that the skills $\boldsymbol{s}$ and the outcome y given t is conditionally independent from each other. Equation 6 is the basic definition for conditional independence and using Bayes rule equation 7 shows this to be true in our case.

$$p(A, B|C) = p(A|C)p(B|C) \tag{6}$$

$$p(\mathbf{s}, y|t) = \frac{p(\mathbf{s}, y, t)}{p(t)} = \frac{p(\mathbf{s}|t)p(t|y)p(y)}{p(t)} = p(\mathbf{s}|t)\frac{p(t|y)p(y)}{p(t)} = p(\mathbf{s}|t)p(y|t) \rightarrow \mathbf{s} \perp y|t \tag{7}$$

Another way to prove this conditional independence is explained in [M.Bishop(2006)]. The so called d-separation can be found when displaying the acyclic directed graph of the Bayesian network for the model. d-separation has a set of rules that have to be met for conditional independence. Firstly, conditional independence of variables is the result of a blocked path. A path is considered blocked if any vectors on the path point to the same node if the node is **not in** C which in our case is t. If all vectors in a path is pointing in the same direction (no vectors are facing each other head to head or tail to tail) the path is considered blocked if any node on the path is **in** C. As seen by the red arrows in the right side of figure 1 the path is blocked when t is observed (**in** C) but not the other way around as demonstrated on the left.

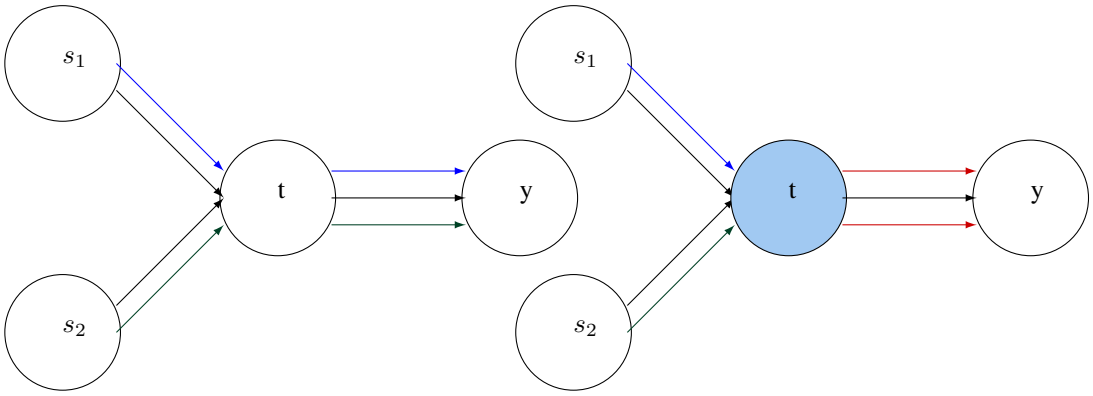

Figure 1: Acyclic directed graph of the Bayesian model, left is p(**s**,y), right is p(**s**,ylt)

Using all of our previous information we can compute the full conditional distribution $p(\mathbf{s}|t)$ using corollary 1 from lecture 2 [Wahlström(2021)].

$$\begin{aligned} p(\mathbf{s}, y|t) = \frac{p(\mathbf{s}, y|t)}{p(y|t)} = \frac{p(\mathbf{s}|t)p(y|t)}{p(y|t)} = p(\mathbf{s}|t) \\ p(\mathbf{s}|t) = \frac{p(t|\mathbf{s})p(\mathbf{s})}{p(t)} \rightarrow p(\mathbf{s}|t) \propto p(t|\mathbf{s})p(\mathbf{s}) \\ p(t|\mathbf{s}) = \mathcal{N}(t; A\mathbf{s} + b, \sigma_t) \end{aligned} \tag{8}$$

$$
\begin{aligned}
p(\mathbf{s}|t) &= \mathcal{N}(\mathbf{s}; \mu_{\mathbf{s}|t}, \Sigma_{\mathbf{s}|t}) \\
\Sigma_{\mathbf{s}|t} &= (\Sigma_{\mathbf{s}}^{-1} + A^T \Sigma_{t|\mathbf{s}}^{-1} A)^{-1} \\
\mu_{\mathbf{s}|t} &= \Sigma_{\mathbf{s}|t}(\Sigma_{\mathbf{s}}^{-1}\mu_{\mathbf{s}} + A^T \Sigma_{t|\mathbf{s}}^{-1} t)
\end{aligned} \tag{9}
$$

Where the full conditional distribution of the **outcome** is given by Bayes theorem and equation 7 to be:

$$
p(t|s_1, s_2, y) = \frac{p(y|t, s_1, s_2)p(t|s_1, s_2)}{p(y|s_1, s_2)} \propto p(y|t)p(t|s_1, s_2) = \delta(y = sign(t))\mathcal{N}(t; A\mathbf{s}, \sigma_{t|\mathbf{s}}^2) \tag{10}
$$

The above equation can be used to represent the outcome of a game, *t*, as a truncated Gaussian

$$
p(t|s_1, s_2, y) \propto \begin{cases} \mathcal{N}(t; s_1 - s_2, \sigma_{t|\mathbf{s}}^2), & y = sign(t) \\ 0, & otherwise \end{cases} \tag{11}
$$

Furthermore we can compute the probability of the outcome being $y = 1$ and player 1 winning. This is called the marginal probability and is found by marginalizing ***s*** from the probability of *t* given ***s*** [Wahlström(2021)].

$$
p(t) = \mathcal{N}(t; \mu_t, \Sigma_t) \tag{12}
$$

Where:

$$
\begin{aligned}
\mu_t &= A\mu_{\mathbf{s}} \\
\Sigma_t &= \Sigma_{t|\mathbf{s}} + A\Sigma_{\mathbf{s}}A^T
\end{aligned} \tag{13}
$$

Also if $t > 0$ then $y = 1$ which gives $p(y = 1) = p(t > 0)$. To get the marginal probability we use the cumulative distribution from [M.Bishop(2006)].

$$
\int_0^\infty p(t)dt = \int_0^\infty \mathcal{N}(t; \mu_t, \Sigma_t)dt \tag{14}
$$

Figure 4 in appendix A shows an example game with the initial skills s_1 and s_2 with the resulting probabilities that either team wins with its marginal probabilities being the area of the truncated Gaussian above or below the origin respectively.

A Gibbs sampler

When using Bayesian inference the posterior is needed to make updates to the predicted skills of players. In the *TrueSkill* model this posterior cannot be calculated explicitly, we therefore implement a Gibbs sampler to sample the posterior $p(s_1, s_2|y)$ using the conditional probabilities $p(t|s_1, s_2, y)$ and $p(s_1, s_2|t, y)$. As previously mentioned *t* is sampled from the truncated Gaussian in equation 11 by using $y = 1 \rightarrow t \geq 0$ and the initial values of s_1 and s_2. This sampling creates three Markov chains, one for s_1, s_2 and *t* respectively.

Setting the initial hyper parameters as the example values given in [Herbrich et al.(2007)Herbrich, Minka, and Graepel] our sampling model uses the following parameters:

$$
\begin{aligned}
\mu_s &= \begin{bmatrix} \mu_{s_1} = 25, \mu_{s_2} = 25 \end{bmatrix} \\
\Sigma_s &= \begin{bmatrix} \sigma_{s_1}^2 = 25/3 & 0 \\ 0 & \sigma_{s_2}^2 = 25/3 \end{bmatrix} \\
\sigma_{t|\mathbf{s}} &= 1
\end{aligned} \tag{15}
$$

With the initial values of the Markov chains being set equal to the above values $s_1 = \mu_{s_1}$, $s_2 = \mu_{s_2}$, $t_0 = 1$. Figure **??** shows the initial values for the skill distributions s_1 and s_2 as well as the burn-in of the s_1 sampling. The bundown shows that the final value takes some time to reach when sampling, the first 50 values are therefore not considered correct as the graph is not in balance. After 50 values though the graph seems to stabilize, in the implementation burn-in is set to be 100 iterations to be sure we have reached a balanced point in the sampling.

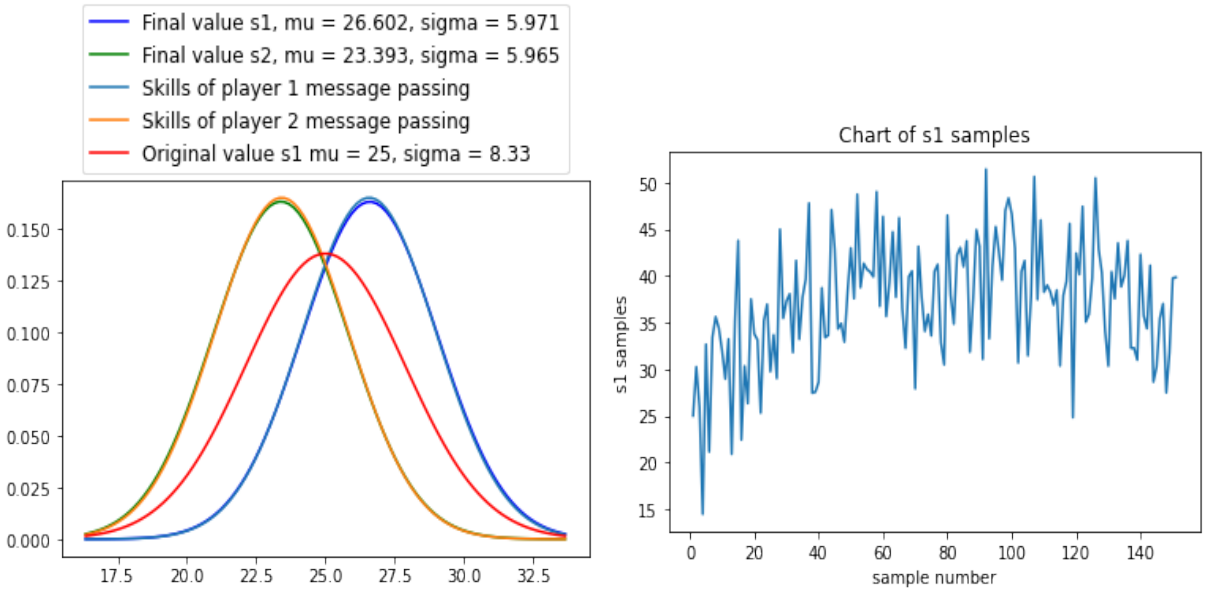

Figure 2: Calculated posterior for s_1 and s_2 using 5200 samples and a 200 sample burn-in period. As well as example of burn-in period for $i < 50$ when sampling s_1 using the Gibbs sampler.

Assumed density filtering

When using the Gibbs sampler to sample from a series of matches the posterior can be updated after every instance of a game. Previously we assumed $y = 1$ and player 1 won every game. Now the outcome is imported from data directly taken from the italian serieA. Depending on the outcome of the actual game *y* takes on the value *0* or *1*. *y = 1* still means player 1 won.

Every team gets their own index T_i. Every team has to get a starting value for their skills which is sampled from the Gaussian $\mathcal{N}(\mu_0, \sigma_0)$. In chronological order, the teams are then matched up as T_1 versus T_2. Using the outcome of the actual game the posterior can be calculated using the above equations to sample the posteriors s_1 and s_2 and further updating the priors.

$$\begin{aligned} \mu_{\mathbf{s}} &= \begin{bmatrix} \mu_{T_1}, \mu_{T_2} \end{bmatrix} \\ \Sigma_{\mathbf{s}} &= \begin{bmatrix} \sigma^2_{T_1} & 0 \\ 0 & \sigma^2_{T_2} \end{bmatrix} \end{aligned} \tag{16}$$

Model predictions

The *TrueSkill* model was implemented without the inclusion of draws and used to predict the following skills for both the Italian SerieA in table 1 and the Candidates Chess tournament, 2022, in table 2. For the SerieA a lot of the predictions seem to be in the general area of their actual placement in the league but the table also does have some major misspredictions. Most notably the model predicted the skill of Fiorentina to be the 7th highest among all teams while the team actually only placed as the 16th best team out of 20 in the league that year. For the chess data all positions are at most one spot away from their actual placement in the tournament.

Running the gibbs sampler in reverse chronological order gives completely different results. As all the teams are assigned equal skill levels in the initial step the first match might play a significant role in the final skill values. When running the sampler in chronological order Juventus wins their first game and proceeds to top the skill predictions. In reverse order Juventus looses their first game and is predicted to be twelve placements lower even though they won the league that year.

Prediction scores for the tables were calculated using a simple algorithm using the final predicted skills of every team and comparing the predicted skill difference to the actual outcome of the game. If the team with higher

Table 1: Table of skills in the Seire A 2018/2019 calculated by the Gibbs sampler compared to actual placement

Place	Team	Average Skill	Standard deviation	Place	Team	Average Skill	Standard deviation
1	Juventus	31.17	0.164	6	Roma	25.04	0.229
2	Napoli	29.0	0.18	13	Spal	24.52	0.176
7	Torino	27.76	0.161	17	Genoa	24.47	0.137
5	Milan	27.22	0.188	12	Udinese	24.32	0.22
11	Sassuolo	26.95	0.226	18	Empoli	23.92	0.21
3	Atalanta	26.84	0.146	10	Bologna	22.77	0.21
16	Fiorentina	26.6	0.186	15	Cagliari	22.27	0.183
8	Lazio	25.39	0.18	14	Parma	22.04	0.232
9	Sampdoria	25.37	0.299	20	Chievo	20.14	0.167
4	Inter	25.28	0.195	19	Frosinone	17.33	0.185

Table 2: Table of skills in the Chess Candidates tournament 2022 calculated by the Gibbs sampler compared to actual points acquired

Points	Player	Average Skill	Standard deviation
9.5	Nepomniachtchi	26.75	0.884
8	Ding	26.64	1.002
7.5	Nakamura	26.14	0.768
5.5	Rapport	25.71	0.867
6	Firouzja	25.45	1.004
6.5	Caruana	24.84	0.83
7.5	Radjabov	23.75	0.828
5.5	Duda	21.47	0.903

predicted skill wins the game a point is added to the score which is the divided by the total amount of games. No draws were considered. Prediction score of table 1 was 72.79% and prediction score of table 2: 69.57%

A deterministic prediction algorithm was also implemented for the football data. Using the average amount of goals scored by each team to predict the outcome of the game. The algorithm uses all the goals scored by the respective teams during the season prior to that game and divides them by the amount of games played by that team. The player with the highest amount of average goals scored is predicted to win the game. For the serieA this deterministic algorithm resulted in a prediction score of 65.9% which is close to the number given by the original Gibbs sampler.

Extension: Including draws

When considering the outcome of a match, the possibility of a draw is often very high depending on the sport or game played. Chess and football often see a lot of draws happening from evenly skilled opponents and will therefore produce a lot of data which have not been assessed in the model until now. Ideally for skill prediction all data available should probably be utilized in some form. By this logic the existing *TrueSkill* model will be extended to include the possibility of a draw.

By [Herbrich et al.(2007)Herbrich, Minka, and Graepel] the probability of a draw in the Trueskill model is given by equation 17

$$Draw\ probability = \Phi\left(\frac{\epsilon}{\sqrt{n_1+n_2}\beta}\right) - \Phi\left(\frac{-\epsilon}{\sqrt{n_1+n_2}\beta}\right) = 2\Phi\left(\frac{\epsilon}{\sqrt{n_1+n_2}\beta}\right) - 1 \tag{17}$$

Which gives:

$$\begin{aligned} \frac{p(Draw)+1}{2} &= \Phi\left(\frac{\epsilon}{\sqrt{n_1+n_2}\beta}\right) \\ \epsilon &= \Phi^{-1}\left(\frac{p(Draw)+1}{2}\right)\sqrt{n_1+n_2}\beta \end{aligned} \tag{18}$$

And is the inverse of a Cumulative Distribution function (CDF) for a zero-mean univariate Gaussian. In this project we have assumed that one team in a game of football is one player in the skill prediction model, therefore n_1 and n_2 are both equal to one. β and $p(Draw)$ are both new hyperparameters to be chosen for the model. β is defined by [Herbrich et al.(2007)Herbrich, Minka, and Graepel] as the performance variance indicating the uncertainty of the game itself. In the case of chess which is very skillful in nature this performance difference would be very small compared to a game involving more luck. With that said a very skillful chess player is often able to make a draw out of a unwinnable situation which would increase the $p(Draw)$ parameter. For now the value for β is the same as previous $\sigma_{t|\mathbf{s}}$ in equation 15 and $p(Draw)$ is set to be 30%.

The above equation defines what [Moser(2010)] and [Herbrich et al.(2007)Herbrich, Minka, and Graepel] defines as the draw margin ϵ. The draw margin can be used to estimate if the outcome of a draw is expected by the model or if there is significant reason for a update in skills given the drawn outcome. If the estimated skill difference of two teams fall outside of the draw margin and a draw is the outcome of a game between the two, the weaker player is rewarded with an increase in skill while the stronger player gets a decrease in skill. If the estimated skill difference fall within the draw margin a draw is expected by the model and therefore there is no need for a skill update.

This rule was implemented to complement the already existing Gibbs sampler for when draws occur. When a draw is detected the new sampler calculates the draw margin by equation 18 and determines if the skill difference $|\mu_{s_1} - \mu_{s_2}|$ is within the draw margin. If it is the sampler samples *t* as equation 19 below.

$$p(t|s_1, s_2, y) \propto \begin{cases} \mathcal{N}(t; s_1 - s_2, \sigma^2_{t|\mathbf{s}}), & sign(\mu_1 - \mu_2) = -sign(t) \;\; and \;\; |t| < \epsilon \\ 0, & otherwise \end{cases} \tag{19}$$

Notice the similarity to equation 11. The only difference is the truncated space sampled, which now is opposite the origin from the stronger team but limited to the width of the draw margin. This should make sure that the stronger team is predicted weaker after the skill update and the weaker team getting a stronger prediction. Because of the fact that the sampling of *t* is limited to the absolute value of the draw margin the effect of a skill update for a draw will most probably be less impactful than the skill update for a win or loss. Also, a larger difference in team rating give a more impactful skill update for the upset. These are both, in theory, great features for modeling a drawn game.

The prediction results for this sampler was slightly better for the football data but slightly worse for chess. The Gibbs sampler using draws had prediction scores of 74.26% for the football and 65.22% for the chess tournament, compared to 72.79% and 69.57% respectively.

A Message Passing algorithm

The joint distribution can also be illustrated by a factor graph, which can be seen in figure 3. The circles represents the different random variables and the black squares represents factors of the joint distribution. With the help of a message passing algorithm the distribution of $p(t|y = 1)$ will once more be approximated. The message passing function can be described as the following:

$$\begin{aligned} \mu_1 = \mu_4 = f_{s_2}(s_2) = p(s_2) \\ \mu_2 = \mu_3 = f_{s_1}(s_1) = p(s_1) \\ \mu_7 = 1 \end{aligned} \tag{20}$$

μ_5 has its message as the marginalized factor f_{st} on s and itself.

$$\mu_5 = \int \mu_3(s_1)\mu_4(s_2)f_{st}(s_1, s_2, t)ds_1 ds_2 \tag{21}$$

Since s_1 and s_2 are independent the distribution **s** for is obtained:

$$\begin{aligned} \mu_3(s_1)\mu_4(s_2) = f_{s1}(s_1)f_{s_2}(s_2) = p(s_1)p(s_2) = p(s_1, s_2) = \\ = p(\mathbf{s}) = \mathcal{N}\left(\begin{bmatrix} s_1 \\ s_2 \end{bmatrix}; \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \right) \end{aligned} \tag{22}$$

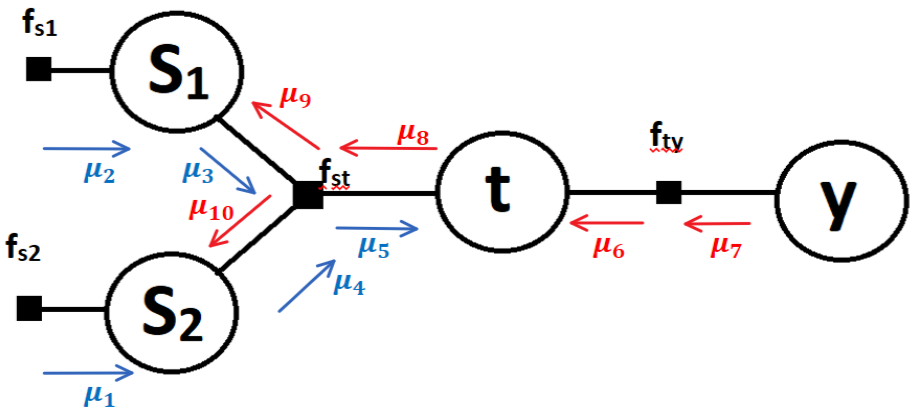

Figure 3: Factor graph with messages.

μ_5 can now be solved by using corollary 2 from lectures [Wahlström(2021)].

$$\mu_5 = \int p(s)p(t|s)ds = \mathcal{N}(t; m_1 - m_2, \sigma_1^2 + \sigma_2^2 + \sigma_{t|s}^2) \tag{23}$$

$$\mu_6 = f_{ty}(t, y) = \delta(y = sign(t)) = \delta(t > 0) \tag{24}$$

By multiplying all the messages at t we get the final distribution:

$$p(t|y = 1) \propto \mu_6\mu_5 = \mathcal{N}(t; m_1 - m_2, \sigma_1^2 + \sigma_2^2 + \sigma_{t|s}^2)\delta(t > 0) \tag{25}$$

$p(t|y = 1)$ is once more a truncated Gaussian. By figure 3 the messages are then sent back through μ_8, μ_9 and μ_{10} where the posterior can be marginalized. But to be able to compute the outgoing message from the t variable node we first need to approximate the truncated Gaussian as a Gaussian. To find $\hat{q}(t)$ the mean and variance of the Gaussian corresponding to the truncated Gaussian is determined. μ_8 is then determined by dividing the Gaussian distribution $\hat{q}(t)$ by μ_5

$$\hat{q}(t) = \mathcal{N}(t; m_t, \sigma_t^2) \tag{26}$$

$$\mu_8(t) = \frac{\hat{q}(t)}{\mu_5} \propto \mathcal{N}(t; m_8, \sigma_8^2) \tag{27}$$

To compute the outgoing messages μ_9 and μ_{10} from f_{st} it's factor is multiplied with the incoming messages from the t and respective s nodes. The resulting double integral is computed with Gaussian shifting and corollary 2 from lectures [Wahlström(2021)].

$$\mu_9 = \int\int \mathcal{N}(t, m_8, \sigma_8^2)\mathcal{N}(t, s_1 - s_2, \sigma_{t|s}^2)\mathcal{N}(s_2, m_2, \sigma_2^2)dtds = \mathcal{N}(s_1, m_2 + m_8, \sigma_8^2 + \sigma_2^2 + \sigma_{t|s}^2) \tag{28}$$

$$\mu_{10} = \int\int \mathcal{N}(t, m_8, \sigma_8^2)\mathcal{N}(t, s_1 - s_2, \sigma_{t|s}^2)\mathcal{N}(s_1, m_1, \sigma_1^2)dtds = \mathcal{N}(s_1, m_1 - m_8, \sigma_8^1 + \sigma_1^2 + \sigma_{t|s}^2) \tag{29}$$

To compute the marginal at the s variable nodes the incoming messages from from the factor node f_{st} are multiplied with the respective s priors. The resulting distributions for $p(t|y = 1)$ are determined by using the formulas for Gaussian multiplication from the lectures [Wahlström(2021)]. The resulting gaussians can be seen in figure 2 and fit very well with the gaussians from the Gibbs sampler.

Discussion

When it comes to the *TrueSkill* model implemented there are a lot of things that might be improved upon. Firstly, there are a lot of hyperparameters to set in the model created, none of the parameters have been adequately tuned. This could quite easily be done using some form of optimization but would have to be tailor made for every game and, or, sport as rules and conditions vary. The game of chess and the game of football for example, have large differences in how they are played and scored. High level chess between evenly skilled opponents are very often drawn even though it is a game of extremely high skill expression. Both of these facts have to be considered when tuning the model.

For the case of football the model in this state only considers wins, losses and draws. A win is 1 point, a draw is 0 and a loss is -1. Wins are considered as 1 regardless of score which neglects a lot of the information given every game. An even game should probably indicate closely matched skill updates whereas a more one-sided game should also be reflected in the skill updates. *TrueSkill* is developed for large multiplayer games where an almost infinite level of complexity can be introduced to the model given in-game data. Examples of these kinds of parameters in football could be: the score, possession, corners, free kicks, passes, shots taken and more. The model could even be extended to measure the skill of every individual player in a team and have the entirety of a team be represented as some sort of aggregate of the individuals.

In this simple implementation the score of the matches are of great importance. 59% of the chess games ended up in a draw. The chess data overall only featured 56 matches which might not be enough for a skill assessment, especially for the first model who only made skill updates for 26 separate non-drawn games. 29% of football games were draws and the total set of games were 380. More data could be gathered for earlier and later seasons but then player transfers and new teams would have to be addressed. For the chess data there is easy access to a lot of it in online databases but the playerbase is enormous and players play new opponents all the time.

Conclusions

We have managed to implement a model that has proven to be useful when predicting the outcome of two player games as well as predicting the skill of the players involved in said game. This model is malleable and can use a wide variety of extensions making it well fitted to use in many settings. This includes, but is not limited to, chess, football and other team sports with similar rules for scoring. The extension of including draws was successful in proving the concept of slight skill adjustments when draws are not expected, but was not successful in significantly improving the performance of the model itself. An increase in performance was probably not to expect though since there are are even more hyperparameters described and used in the model, none of which have been tuned. It is our belief that a well tuned TrueSkill model can be more effective than demonstrated in this report.

The model performed predictions with up to 74% accuracy which is better than guessing 50/50 as well as better than the deterministic algorithm for goals scored in football. The first model also showed promising results for ranking the skills of the players with almost all chess players being within 1 or 2 places of their final rankings as well ass most of the football teams being within 4 or 5 ranking spots of their actual placement in the league with some exceptions. The model also showed great sensitivity to the ordering of the games which need to be addressed. Overall the model performed well when considering how unpolished it is and would be well suited to develop further.

References

[Herbrich et al.(2007)Herbrich, Minka, and Graepel] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill^{TM}: A bayesian skill rating system, 2007. URL https://www.microsoft.com/en-us/research/publication/trueskilltm-a-bayesian-skill-rating-system/.

[M.Bishop(2006)] Cristopher M.Bishop. *Pattern recognition and Machine Learning, Springer*. 2006. ISBN 10: 0-387-31073-8.

[Moser(2010)] Jeff Moser. The math behind trueskill^{TM}, 2010. URL https://www.moserware.com/assets/computing-your-skill/The%20Math%20Behind%20TrueSkill.pdf.

[Wahlström(2021)] Niclas Wahlström. Lectures (mostly lecture 2 and 7), 2021. Advanced Statistical Machine Learning course, Uppsala Universitet.

A Figure Appendix

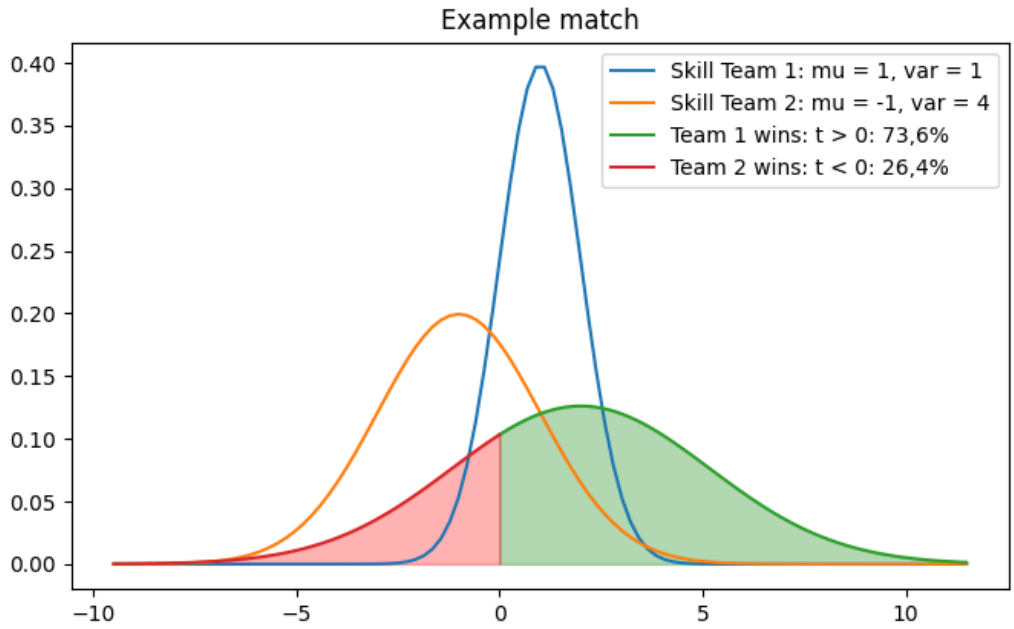

Figure 4: Calculations of marginal outcome probabilities for a example match.