Advanced Probabilistic Machine Learning - Miniproject

Isar Silfur **Balasubramani Vishnumpet Sundararaman**

Tsion Tegen **Otema Yirenkyi**

1 Introduction

In this report, we detail the steps taken in implementing different Bayesian methods for estimating the skill of players and ranking them. The target dataset being used is the Serie A division 2018/2019 dataset. But similar datasets were applied to our model for later comparison.

The Bayesian methods used are - ADF with Gibbs Sampling: ADF with Gibbs Sampling is ideal for football leagues where team rankings evolve over time, and unobservable factors like form or injuries influence match outcomes. It provides a fully Bayesian framework for estimating rankings, but the computational demands can be significant.

TrueSkill: TrueSkill is a Bayesian ranking system designed for real-time ranking updates in football, accounting for both team and individual player performances. It includes uncertainty in estimating a team's strength, making it suitable for leagues where results can vary due to unexpected factors like injuries or transfers.

Message-Passing: Message-passing algorithms provide efficient, scalable inference for ranking football teams in large leagues, especially when there are complex interactions between teams, such as home vs. away performances. These methods approximate Bayesian inference quickly, making them practical for large datasets where exact inference would be too slow.

2 Assignments

2.1 Modeling

The model's 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, $y = \text{sign}(t)$, for the game result.

Formulating the TrueSkill Bayesian model for one match between two players:

The 5 hyperparameters in the model are:

$$(\mu_1,\, \mu_2,\, \sigma_1^2, \sigma_2^2, \beta^2)$$

The skills of Player 1 s_1 and Player 2 s_2 are normally distributed.

- Prior: $p(s_1), p(s_2) \sim N(\mu, \sigma^2)$, where μ is the mean and σ^2 is the variance.

Respectively,

$$Prior : s_1 \sim \mathcal{N}(\mu_1,\, \sigma_1^2)\, Prior : s_2 \sim \mathcal{N}(\mu_2,\, \sigma_2^2)$$

After game observation, skills are updated using:

$$p(s_1, s_2 | \mu, \Sigma, y, t) = P(t, s_1, s_2)$$

Outcome t:

$$t \sim \mathcal{N}(s_1 - s_2, \beta^2) = p(t|s_1, s_2)$$

Where β^2 is the variance of the match performance, according to the Trueskill model.

Result y:

$$y = \text{sign}(t)$$

$$p(y|t) = \begin{cases} 1, & t \geq 0 \\ -1, & t < 0 \end{cases}$$

The model is a joint probability distribution of all the game's random variables:

$$P(s_1, s_2, t, y) = P(s_1)P(s_2)P(t|s_1, s_2)P(y|t)$$

2.2 Bayesian Network

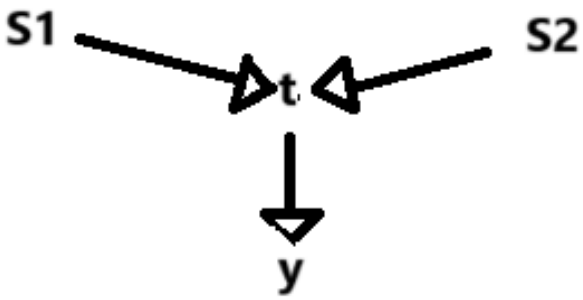

Figure 1: Bayesian Network

Using d-separation we can show conditional Independence between, s_1, s_2 and y since $s_1 \;\; s_2$ only influence y through t. Since y value depends on the value of t, once we know t, knowing s_1 or s_2 doesn't give us additional information about y. If we condition on t, s_1 and s_2 are blocked from y. Additionally, we can look at the only possible path from $s_1 \to y$ is through t. Creating the path $s_1 \to t \to y$ which is blocked due to t being observed and therefore independent. This would be expressed as:

$$p(s_1, y|t) = \frac{p(s_1, y, t)}{p(t)} = \frac{1}{p(t)} p(y|t)p(y|s_1)p(s_1)$$

$$= p(y|t)\frac{p(t|s_1)p(s_1)}{p(t)} = p(y|t)p(s_1|t) \Rightarrow s_1 \perp y|t$$

The same logic would apply for s_2 to show that $s_2 \perp y|t$.

Verifying that $s_1 \perp s_2$ we could see that the only possible path is through t, but since t is a collider node there is no possible path to travel from s_1 to s_2 and visa vie i.e. $s_1 \to t \leftarrow s_2$. This would be expressed as:

$$p(s_1, s_2) = \Sigma_t p(s_1, s_2, t) = \Sigma_t p(t|s_1, s_2)p(s_1)p(s_2)$$

$$\Rightarrow p(s_1)p(s_2) \Rightarrow s_1 \perp s_2|\emptyset$$

Gaussian random variables, conditionally independent of each other.

2.3 Computing with the model

2.3.1 The full conditional distribution of the skills, $p(s_1, s_2|t, y)$.

We can eliminate y in the full conditional distribution because of the findings in Q2 where we showed that y and s_1, s_2 is conditionally independent from one another when t is observed, therefore the expression for the full conditional distribution can be reduced to: $p(s_1, s_2|t, y) = p(s_1, s_2|t)$.

To calculate $p(s_1, s_2|t)$ we have to consider how $p(s_1, s_2)$ and $p(t)$ looks like. We know that both s_1 and s_2 are random normally distributed variables with mean and variance equal to μ_1, σ_1^2 and μ_2, σ_2^2 respectivly. The random variable t is a product of s_1 and s_2 from the equation $t = s_1 - s_2$. Therefore t is a normally distributed variable with mean $\mu_1 - \mu_2$ and variance equal to the match outcome uncertainty β^2.

Using affine transformation (Corollary 2), when assumed that x_b is conditioned on x_a then defined as:

$$p(x_a) = \mathcal{N}(x_a; \mu_a, \Sigma_a),\ p(x_b|x_a) = \mathcal{N}(x_b; Ax_a + b, \Sigma_{b|a})$$

$$p(x_a|x_b) = \mathcal{N}(x_a; \mu_{a|b}, \Sigma_{a|b})$$

$$\Sigma_{a|b} = (\Sigma_a^{-1} + A^T\Sigma_{b|a}^{-1}A)^{-1},\ \mu_{a|b} = \Sigma_{a|b}(\Sigma_a^{-1}\mu_a + A^T\Sigma_{b|a}^{-1}(x_b - b))$$

Since t is conditioned on $\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$ and therefore $A = [1 \quad -1]$ because of $t = s_1 - s_2$. We can use corollary 2, where $\mathbf{s} = x_a$ and $t = x_b$. The variance of t given s, $\Sigma_{t|a}$, is β^2 since we have $t = s_1 - s_2 + \epsilon$, where $\epsilon = \mathcal{N}(0, \beta^2)$, Gaussian noise, that models the uncertainty of the game outcome even if s_1 and s_2 is known.

$$p(\mathbf{s}) = \mathcal{N}(\begin{bmatrix} s_1 \\ s_2 \end{bmatrix}; \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}),\ p(t|\mathbf{s}) = \mathcal{N}(t; [1 \quad -1] \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \beta^2)$$

The conditional distribution $p(s_1, s_2|t)$ is then given by: $p(\mathbf{s}|t) = \mathcal{N}(\mathbf{s}, \mu_{s|t}, \Sigma_{s|t})$, where the variance and mean can be expressed as:

$$\Sigma_{s|t} = (\Sigma_s^{-1} + A^T\Sigma_{t|s}^{-1}A)^{-1} = (\begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} + \frac{1}{\beta^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix})^{-1} = \frac{\sigma_1^2\sigma_2^2}{\beta^2 + \sigma_1^2 + \sigma_2^2}\begin{bmatrix} \frac{\beta^2+\sigma_2^2}{\sigma_2^2} & 1 \\ 1 & \frac{\beta^2+\sigma_1^2}{\sigma_1^2} \end{bmatrix}$$

$$\mu_{s|t} = \Sigma_{s|t}\left(\Sigma_s^{-1}\mu_s + A^T\Sigma_{t|s}^{-1}(t - b)\right) = \frac{1}{\beta^2 + \sigma_1^2 + \sigma_2^2}\begin{bmatrix} \mu_1(\beta^2 + \sigma_2^2) + \sigma_1^2(\mu_2 + t) \\ \mu_2(\beta^2 + \sigma_1^2) + \sigma_2^2(\mu_1 - t) \end{bmatrix} \quad (1)$$

2.3.2 The full conditional distribution of the outcome, $p(t|s_1, s_2, y)$.

Using Bayes theorem we can express the conditional distribution of the outcome as $p(t|y, s_1, s_2) \propto p(y|t)p(t|s_1, s_2)$, because $s_1, s_2 \perp y$ and therefore $p(y|t, s_1, s_2)$ can be written as $p(y|t)$. The prior, the likelihood of outcome t given the players skills s_1, s_2 is:

$$p(t|s_1, s_2) = \mathcal{N}(t|(s_1 - s_2), \beta^2) \quad (2)$$

.

The likelihood $p(y|t)$, can be split into two cases: If $t > 0$ then player 1 wins, so $y = 1$. If $t < 0$ then player 2 wins, so $y = -1$. Therefore the likelihood must encode this relationship and it can be expressed as:

$p(y|t) = \begin{cases} 1, & \text{if} \quad sign(t) = y \\ 0, & \text{otherwise} \end{cases}$ Therefore, the full conditional probabillity is a truncated Guassian with parameters a and b corresponding to either $a = 0, b = +\infty$ for $y = 1$ and for $y = -1$, $a = -\infty, b = 0$. Meaning we are cutting either the positive or negative part of the Guassian $p(t|s_1, s_2)$ depending on the outcome $p(y|t)$.

2.3.3 The marginal probability that Player 1 wins the game, $p(y = 1)$

The marginal probability that Player 1 wins the game is expressed by the Cumulative Distribution Function ie finding the standard for converting a z-score (normalization value) into the probability.

$$\int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

where z is:

$$z = \frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2 + \beta^2}}$$

2.4 A first Gibbs sampler

The solution to the Gibbs sampler requires the following -

Posterior distribution of the Skills given the outcome. This can be derived from the gaussian in 1

Conditional distribution of the outcome given the skills. This can be evaluated by using the gaussian derived in 2

- The initial value for burn-in was set to 100 and it was observed that the distribution produced is not accurate. For the second iteration various burn-in values were tried from 500, 1500, 2500, 3500, 75000, 15000, 50000. From running the various values for burn-ins, for the chosen hyperparameters in TrueSkill with the result of y = 1, the value that would be time efficient is between burn-in = 500 and burn-in = 1000.
- To get a well-described distribution for the sampling from Gibbs we can run the burn-in for 7500 samples. A middle-ground is needed with regard to time taken to generate these samples and therefore it should be 5000 total samples generated with 2500 burned. See Fig 2
- The posterior for S1 and S2 when calculated with Gibbs have shifted from the priors with symmetry. Given the result of S1 > S2 this is expected. The distributions look like this- Sampled Mean of s1: 29.2589, Variance: 50.5593 Sampled Mean of s2: 20.6067, Variance: 48.9899 The execution times with half the samples being burnt are plotted in Fig 3

2.5 Assumed Density Filtering

The Gibbs sampler from Q.4 processes the result of one match to give a posterior distribution of the skills given the match. We can use this posterior distribution as a prior for the next match in what is commonly known as assumed density filtering (ADF). In this way, we can process a stream of different matches between the players, each time using the posterior distribution of the previous match as the prior for the current match.

The solution to the ADF for the data in SeriesA.csv is shown under Fig 4.

- The variance of the skills of each team in the graphs indicate how well the rankings will hold true in the current system. For instance Juventus with its lower variance will rank higher in most runs while Florentina with its higher variance will have more uncertain positions in the ranking after updates.
- After randomization the rankings shift but the variance of the teams remain the same. This is because of the way the skills of each team vary after the processing of each result.

2.6 Using the model for predictions

The defined function in the model predicts the outcome of a match between two teams using the TrueSkill rating system, taking into account both team ratings and home-field advantage. It initializes the ratings for both teams.Predictions are made only if both teams have played at least 3 matches, with a probability calculated using the difference in ratings plus any home advantage. If the win probability for team 1 is greater than 0.5, it predicts team 1 to win; otherwise, it predicts team 2 to win.The function predicts match outcomes using the TrueSkill rating system,adjusting for team ratings and potential home-field advantage. It only makes predictions if both teams have played at

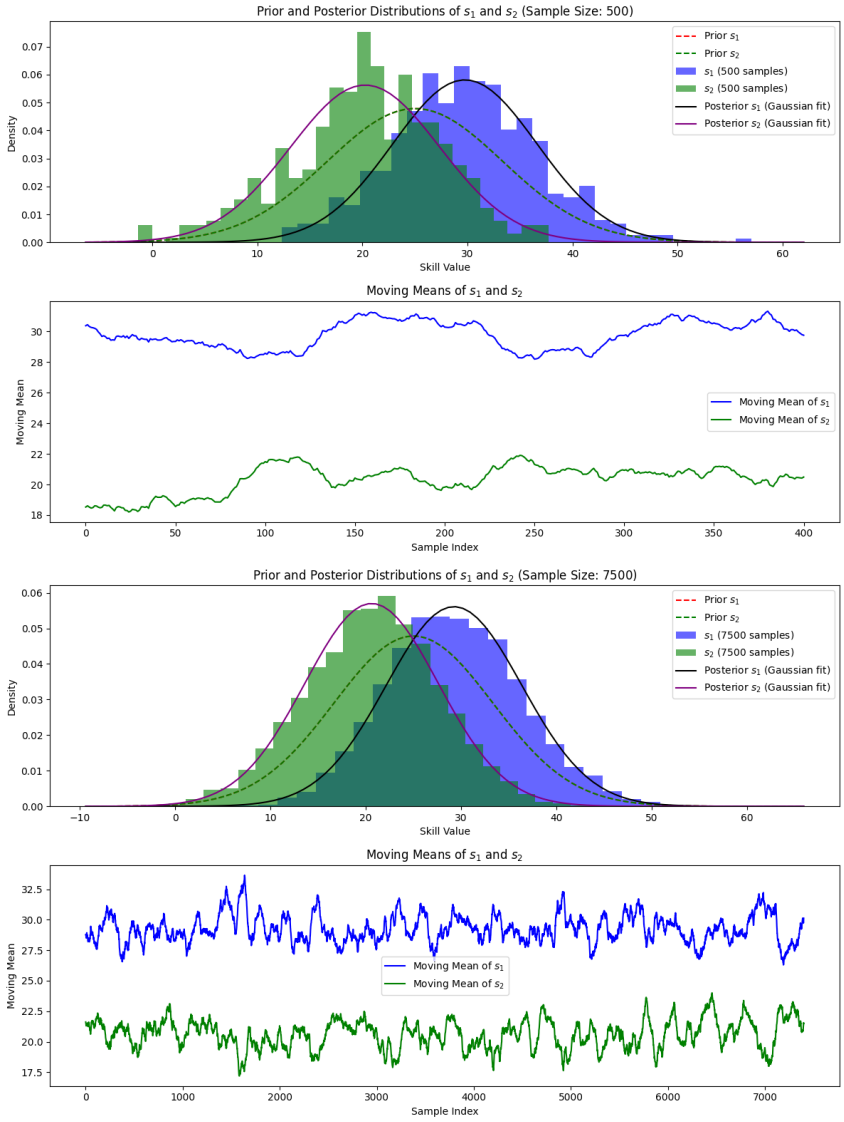

Figure 2: Plotting the samples from Gibbs sampling

least three matches, using a win probability based on the difference in ratings and home advantage. The prediction result for the correct and random guesses can be seen in Fig 5.

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

Therefore the final prediction rate is 0.57, which is better than random guessing. This performance could be improved by for example, adding more contextual data.

2.7 Factor graph

Factoring our model,

$$p(t|s_1, s_2, y) = f_{s1}(s_1) f_{s2}(s_2) f_{ts1s2}(s_1, s_2, t) f_{ty}(t, y)$$

The random variable nodes are denoted as:

Node s_1 = N_{s1}, Node s_2 = N_{s2}, Node t = N_t, Node y = N_y

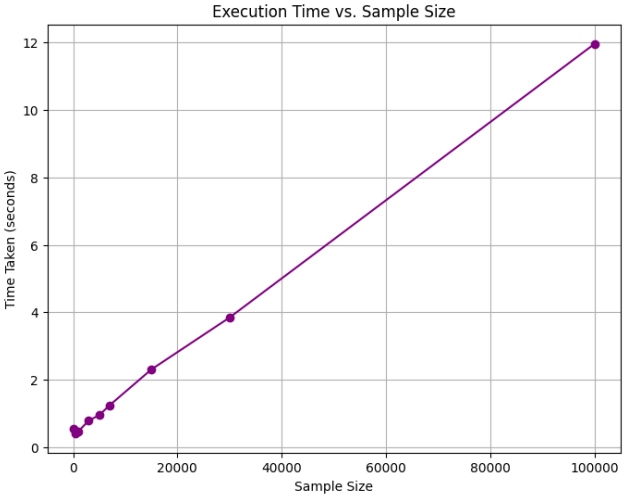

Figure 3: Execution time

The factors are denoted as:

f_{s1} = f_a, f_{s2} = f_b, f_{ts1s2} = f_c, f_{ty} = f_d, f_e = 1.

The messages as represented in the Figure 7 above are:

$\mu_{N_y \to t_y}(y) = 1$

$\mu_{N_{s_2} \to c}(s_2) = \mu_{b \to N_{s_2} = f_b}(s_2)$

$\mu_{N_{s_1} \to c}(s_1) = \mu_{a \to N_{s_1} = f_a}(s_1)$

$\mu_{c \to N_t(t)} = \int f_c\,(s_1 s_2\, t) \cdot \mu_{N_{s_1} \to c}\,(s_1) \cdot \mu_{\,N_{s2} \to\, c}\,(s_2)$

$\mu_{d \to N_t(t)} = \int f_d\,(\,t\; y)$

The explicit messages for computing p(tly) is:

$p\,(t) \;\propto\; \mu_{c \to N_t(t)} \mu_{d \to N_t(t)}$

2.8 Message passing algorithm

The main rationale of the implemented algorithm for this part is that all the incoming messages, as stated in figure 5, are calculated for $p(t)$, i.e. $p\,(t) \;\propto\; \mu_{c \to N_t(t)} \mu_{d \to N_t(t)} = \mathcal{N}(t; \mu_{s_1} - \mu_{s_2}, \sigma^2_{s_1} + \sigma^2_{s_2} + \beta^2) \cdot \mathcal{I}_{t>0} = \mathcal{N}(t; \mu_t, \sigma^2_t) \cdot \mathcal{I}_{t>0}$, a truncated Gaussian for $y = y_{obs}$. Moment matching is used to estimate the marginal probability $p(t|y = 1)$, that is a truncated Gaussian, as a Gaussian $\hat{q}(t)$. The estimated Gaussian $\hat{q}(t) = \mathcal{N}(t; \hat{\mu}, \hat{\sigma}^2)$, where the mean and variance will depend on the truncation, in this case since $y = 1$, a is equal to 0 and b is equal to positive infinity and also the incoming message from f_c. We can now estimate the posterior of $p(s_1|y = y_{obs}) and p(s_2|y = y_{obs})$ We then send back the message to f_c and divide with the incoming previous message $\mu_{c \to} N_t$. This can be done with Gaussian division where $\hat{\mu}_{c \leftarrow} N_t$ is the outgoing message to the factor f_c.

$$\hat{\mu}_{c \leftarrow} N_t = \frac{\hat{q}(t)}{\mu_{c \to} N_t} = \mathcal{N}(t; \frac{\hat{\mu}\sigma^2_t - \mu_t \hat{\sigma}^2}{\sigma^2_t - \hat{\sigma}^2}, \frac{\hat{\sigma}^2 \sigma^2_t}{\sigma^2_t - \hat{\sigma}^2}) = \mathcal{N}(t; \mu_{tc}, \sigma^2_{tc})$$

Then we have to calculate the outgoing message from f_c to s_1 and s_2 variable nodes to estimate $p(s_1|y = y_{obs})$ and $p(s_2|y = y_{obs})$. We call this message $\hat{\mu}_{N_{s_1} \leftarrow c}$ and $\hat{\mu}_{N_{s_2} \leftarrow c}$. This is done by marginalizing over the factor function in f_c wich is $f(s_1, s_2, t)$ with the incoming messages $\hat{\mu}_{c \leftarrow N_t}(t)$ and $\mu_{N_{s_2} \to c}(s_2)$.

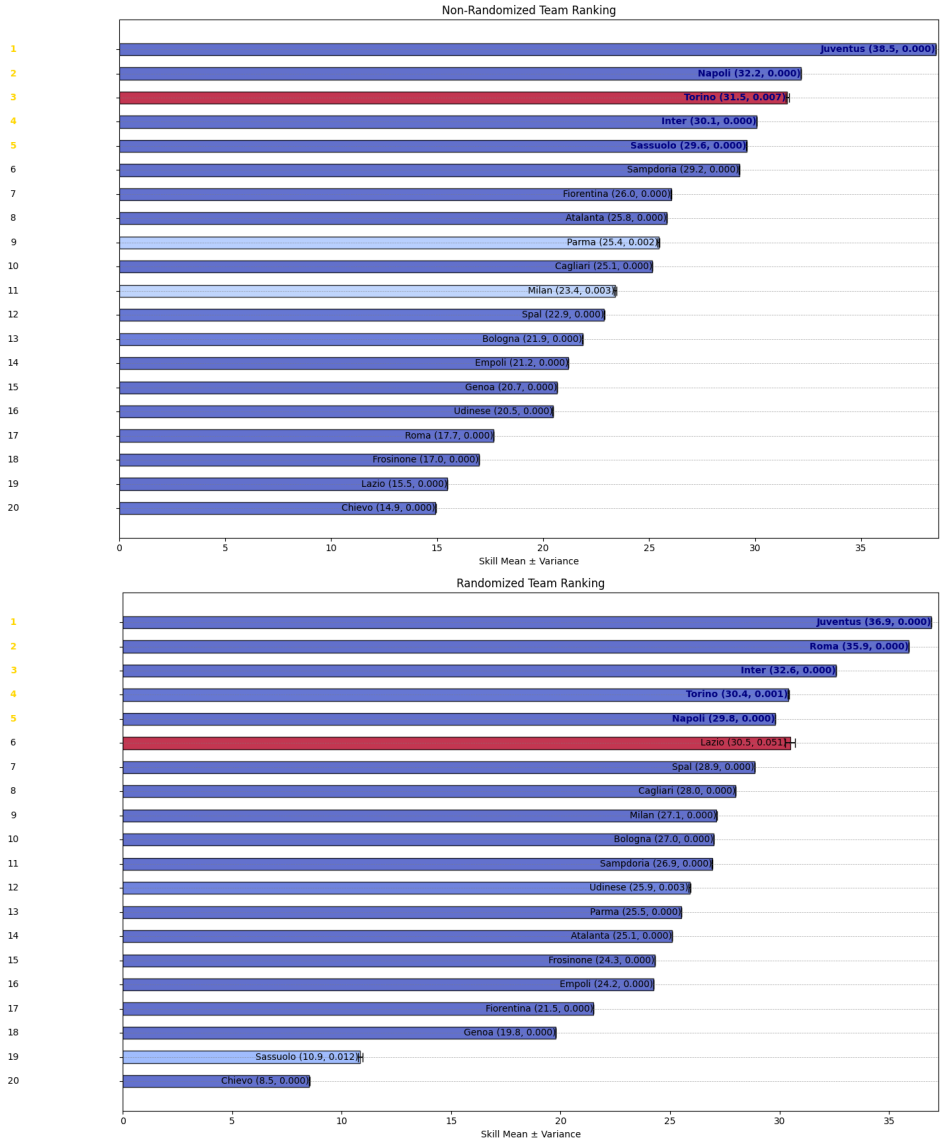

Figure 4: Ranking for skills from ADF processing

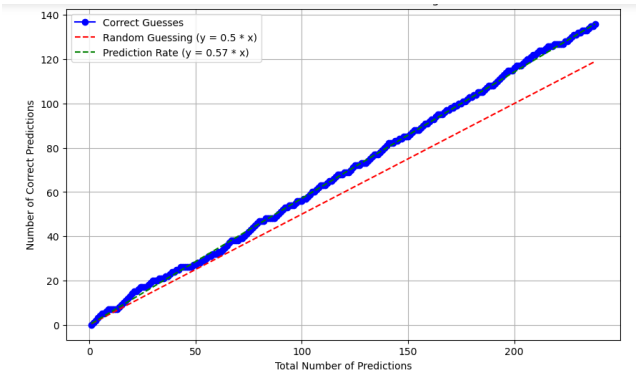

Figure 5: Predicted result

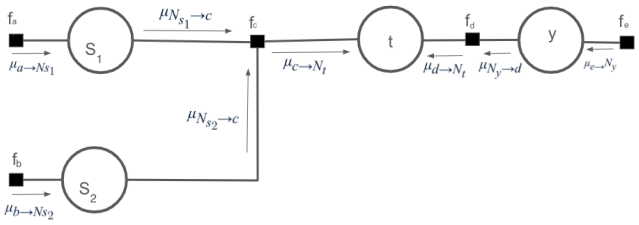

Figure 6: Factor Graph

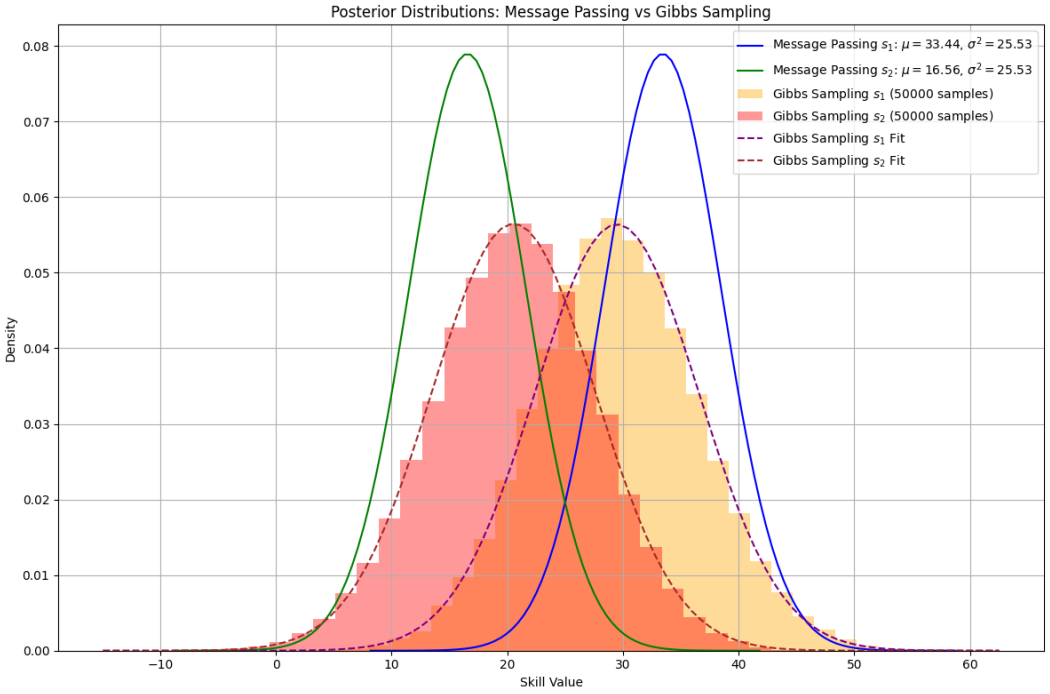

Figure 7: Estimate of S1, S2 after moment matching algorithm and comparison

$$\hat{\mu}_{N_{s_1} \leftarrow c}(s_1) = \int_{s_2} \int_{t} f_c(s_1, s_2, t) \cdot \hat{\mu}_{c \leftarrow N_t}(t) \cdot \mu_{N_{s_2} \rightarrow c}(s_2) ds_2 dt$$

$$\hat{\mu}_{N_{s_1} \leftarrow c}(s_1) = \int_{s_2} \int_{t} \mathcal{N}(s_1 - s_2; t, \beta^2) \mathcal{N}(t; \mu_{tc}, \sigma_{tc}^2) \mathcal{N}(s_2; \mu_{s_2}, \sigma_{s_2}^2) ds_2 dt$$

$$\hat{\mu}_{N_{s_1} \leftarrow c}(s_1) = \mathcal{N}(s_1; \mu_{s_2} + \mu_{tc}, \sigma_{tc}^2 + \sigma_{s_2}^2 + \beta^2) = \mathcal{N}(s_1; \mu_{cs_1}, \sigma_{cs_1}^2)$$

For the message going to s_2 the same method is used execpt we are now instead substituting the message from s_2 with the message from s_1 ($p(s_1)$), and marginalizing over $\int_{s_1,t} ds_1 dt$ instead.

Finally after calculating the message from f_c to s_1, we simply multiply the message with the other incoming message in s_1 which is $\mu_{b \rightarrow} N_{s_1}$ the Gaussian initial distribution for s_1. This will give us a new $p(s_1)$ estimate. We then do the same procedure from f_c to s_2 to get the new value of s_2. We can then estimate a new value for $p(t)$ given the updated values of s_1 and s_2. This should converge the values after a couple of iterations, which it does and expected behavior is observed in figure 6, where since in this case $y_{obs} = 1$ we see the mean of skill for player 1 increasing while the mean for the skill of player 2 decreases.

2.9 Your own data

We tested our Trueskill model by making a prediction for the football matches of the English Premier League Season 0910 Dataset We combined two dataset files, combined them and extracted only needed columns. This model tracks the performance of each team by updating their ratings after each match and only makes predictions once both teams have played at least three matches. The accuracy of the predictions is then calculated based on the actual match results. In this case, the prediction accuracy is 0.64, which is slightly better than that of the previous dataset. This improvement may be due to a better balance in the current dataset, with fewer draws and more consistent team performances, making predictions easier.The prediction result for the correct and random guesses can be seen in Fig 8.

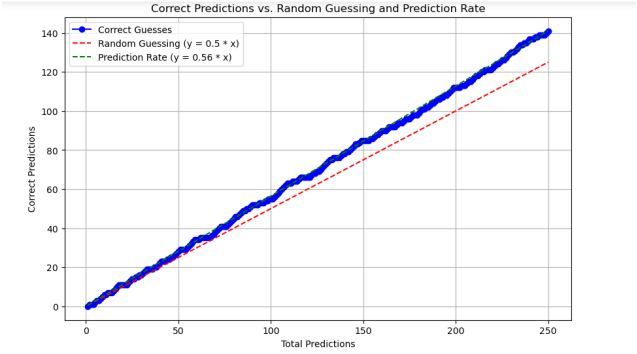

Figure 8: Predicted result

2.10 Open-ended project extension

For this sections, we describe in details the ways in which we made our own extensions to the prediction model.

Initialization of Ratings from Extra Data: We used historical match data (extra dataset) to initialize the ratings of teams before making predictions on the current dataset. This approach allows the model to start with a more informed basis of each team's skill level rather than beginning with neutral ratings.This data helps the model adjust their rating upward, which directly impacts prediction outcomes.

Dynamic Rating Updates: After each match, the ratings for both teams are updated based on the actual match outcome. The model adjusts the ratings using the TrueSkill algorithm, which factors in the uncertainty associated with each team's skill level. This dynamic adjustment allows the model to respond to current performance levels and team form, making it more responsive to changes such as injuries, transfers, or other events affecting team performance. As teams win or lose, their ratings reflect their current abilities, improving prediction accuracy over time.

Minimum Matches Requirement for Predictions: The model only makes predictions if both teams have played at least three matches. This requirement ensures that the ratings are based on a sufficient sample size, reducing the likelihood of random fluctuations affecting predictions. This threshold prevents the model from making predictions based on insufficient data, which can lead to unreliable results. By requiring a minimum number of matches, we enhance the confidence in the predictions made.

Probabilistic Outcome Prediction: The model calculates the probability of one team winning against another using the `quality_1vs1` method from TrueSkill. It returns a prediction only if both teams meet the criteria for a minimum number of matches played. Using probabilistic predictions allows the model to reflect uncertainty in match outcomes. By calculating win probabilities rather than deterministic results, the model can better account for close matchups, where both teams have similar ratings, thereby enhancing its predictive capabilities. An accuracy rate of 0.65 indicates that the model is fairly effective, and monitoring this metric helps identify when to re-evaluate strategies or make additional adjustments.

Goal Difference Predictor This method uses the goal difference (GD) calculated as:

$$GD = \text{Goals Scored} - \text{Goals Conceded} \tag{3}$$

despite applying these method, it yielded similar prediction scores of approximately 0.65.

Conclusion

The TrueSkill model was applied to analyze the dynamics of team skills and match outcomes using the Serie A division 2018/2019 dataset, our results yielded a prediction accuracy of 0.57. This highlights that the model is able to quite confidently capture the complexities of team performance.

Applying ADF with Gibbs sampling on the dataset yields rankings that may fluctuate but over time will better represent the unknown factors captured in the data.

To improve the ADF model, incorporating home vs. away advantage would better account for match location influences on team performance. Using time-dependent skill evolution ensures that skills reflect recent form, either through a time-decay factor or a sliding window. Shifting to a hierarchical Bayesian model allows skills to evolve more naturally over time, making the model more robust.

References

[1] Microsoft Research. (2024). *TrueSkill ranking system*. Microsoft. https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/