APML Mini-Project

Oliver Cronsten
M.Sc. Industrial Analytics

Joel Korpi
M.Sc. Industrial Analytics

Danyal Mirza
M.Sc. Computer Science

Niclas Svensson
M.Sc. Computational Science

Abstract

This project estimates player skills in a competitive setting using Microsoft's *TrueSkill* Bayesian machine learning model, which represents player skills as Gaussian random variables and computes their posterior distribution via Bayesian inference. A *one-step ahead* predictor, based on moment matching, was applied to sequentially process match data. The model was initially tested on 2018/2019 Serie A football data, then adapted to UFC match-ups from 1993-2021. No prior skill level was assumed for Serie A, while the UFC model assumed heavier fighters had an advantage. Both stochastic and deterministic predictors were implemented, yielding mixed results. The stochastic predictor achieved the highest accuracy of $r = 0.721$ for Serie A, but also the lowest of $r = 0.510$ for UFC, possibly due to differences in sports such as the role of luck, or the time ranges of the datasets.

1 Introduction

Probabilistic machine learning models are effective for solving estimation and classification problems, including skill estimation in competitive settings. This project uses a Bayesian inference model with Gibbs sampling to estimate the skill levels of teams in the Italian football division Serie A from 2018/2019. The model is based on Microsoft's *TrueSkill* system, which extends the Elo rating system by providing a probabilistic framework that quantifies both skill and uncertainty.

In this project, the *TrueSkill* Model will be implemented, and methods such as Bayesian inference, Gibbs sampling, factor graphs, and message passing will be utilized throughout the project to iteratively update team skills estimates and later predict match outcomes. Once the model has been tested on the Serie A dataset, it will be further evaluated on a dataset from the Ultimate Fighting Championship (UFC). This is done to investigate if the model translates to different competitive domains efficiently.

2 Assignments

This chapter introduces the *TrueSkill* model, progressing from basic probability concepts to a fully developed predictive machine-learning model.

Q1 Modelling

We begin by declaring the random variables of each player's skill, s_1 and s_2, according to the *TrueSkill* model. This gives us two Gaussian distributions with the mean μ and variance σ^2.

$$\begin{aligned} p(s_1) &= \mathcal{N}(s_1; \mu_{s1}, \sigma_{s1}^2) \\ p(s_2) &= \mathcal{N}(s_2; \mu_{s2}, \sigma_{s2}^2) \end{aligned} \tag{1}$$

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

A random variable t, representing the game's outcome, follows a Gaussian distribution with mean equal to the skill difference $s_1 - s_2$, and variance β^2 (*to be specific, β controls the rating gap needed for a higher-rated player to have a given probability of winning*). If $t > 0$, player 1 wins; if $t < 0$, player 2 wins. The conditional probability of t is:

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

The discretization of t is y, since $y = sign(t)$. Thus, the random variable for the result where a certain player wins is:

$$p(y|t) = \mathbf{1}_{\{y-sign(t)\}} = \begin{cases} 1 & \text{if } y = \text{sign}(t) \\ 0 & \text{otherwise } \textit{(since no draws can occur)} \end{cases} \tag{3}$$

For these distributions, there are 5 hyperparameters in total: $\mu_{s1}, \mu_{s2}, \sigma_{s1}, \sigma_{s2}, \beta$. The resulting joint distribution is the product of all the equations above, representing the *TrueSkill* Bayesian model for match outcomes between players 1 and 2:

$$p(s_1, s_2, y, t) = p(y|t)p(t|s_1, s_2)p(s_1)p(s_2) \tag{4}$$

Q2 Bayesian Network

Given the joint distribution in equation 4, a Bayesian network is formed (see Figure 1). Initially, s_1 and s_2 are independent. Then, t depends on s_1 and s_2 via $p(t|s_1, s_2)$, and y depends on t through $p(y|t)$. To verify that s_1 and s_2 are conditionally independent of y given t, we condition on t as observed. Since t is a head-to-tail node for s_1 and s_2 individually, they become conditionally independent of y given t.

Moreover, s_1 and s_2 remain independent because their unobserved descendant t forms a head-to-head node, and further descendants are also unobserved.

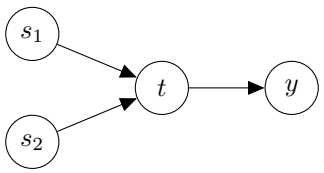

Figure 1: A Bayesian network illustrating the dependencies between skills s_1, s_2, task performance t, and the outcome y.

Q3 Computing with the model

In this chapter, the full conditional distribution of the players' skills is derived, as well as incorporating posterior updates based on the observed match outcomes.

Q3.1 The full conditional distribution of the skills

To predict players' skill levels using the model, the posterior needs to be described as a full conditional distribution of t given both players' skill levels $\mathbf{s}$, i.e. $p(t|\mathbf{s})$. As described in Q2, s_1 and s_2 are independent, which makes the calculation of their joint posterior distribution efficient:

$$p(s_1, s_2, |t, y) = \frac{p(s_1, s_2, y, t)}{p(y, t)} = \frac{\cancel{p(y|t)}p(t|s_1, s_2)p(s_1)p(s_2)}{\cancel{p(y|t)}p(t)} \tag{5}$$

Next, the chain rule is applied to expand $p(t|s_1, s_2)$ and find the final posterior:

$$\longrightarrow \frac{p(t, s_1, s_2)\cancel{p(s1)}\cancel{p(s2)}}{p(t)\cancel{p(s1)}\cancel{p(s2)}} = \frac{p(s_1, s_2, t)}{p(t)} \tag{6}$$

Next, the conditional probability $p(t|\mathbf{s})$ is expressed as a Gaussian distribution, given the skill ratings in the vector form $\mathbf{s} = [s_1, s_2]^\top$. For $p(t|\mathbf{s})$, the mean is represented as $\mathbf{As}$, where $\mathbf{A}$ is a transformation matrix that adjusts the mean of s_1 and s_2. In this case, $\mathbf{A} = [1, -1]$, since $p(t|s_1, s_2) = \mathcal{N}(t; s_1 - s_2, \beta^2)$. Thus, the new mean of the distribution becomes $[\mu_{s1}, \mu_{s2}]^\top$, where μ_{s1} and μ_{s2} are the means of the prior distributions of s_1 and s_2, respectively. As the skill levels are independent, the covariance matrix is $\Sigma = \begin{pmatrix} \sigma_{s1}^2 & 0 \\ 0 & \sigma_{s2}^2 \end{pmatrix}$. Putting this together gives:

$$p(s) = \mathcal{N}(t; [\mu_{s1}, \mu_{s2}]^\top, \Sigma) \tag{7}$$

Next, *Corollary 1 (Affine transformation - Conditioning)* from Lecture 2 (1) to switch $p(t|\mathbf{s})$ to $p(\mathbf{s}|t)$.

$$p(\mathbf{s}|t) = \mathcal{N}(\mathbf{s}; \mu_{\mathbf{s}|t}, \Sigma_{\mathbf{s}|t}) \tag{8}$$

Where:

$$\Sigma_{\mathbf{s}|t} = (\Sigma^{-1} + A^\top \frac{1}{\beta^2} A)^{-1} = \left(\begin{bmatrix} \sigma_{s1}^2 & 0 \\ 0 & \sigma_{s2}^2 \end{bmatrix}^{-1} + \frac{1}{\beta^2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right)^{-1} \tag{9}$$

$$\mu_{\mathbf{s}|t} = \Sigma_{\mathbf{s}|t} \left(\begin{bmatrix} \sigma_{s1}^2 & 0 \\ 0 & \sigma_{s2}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mu_{s1} \\ \mu_{s2} \end{bmatrix} + \frac{t}{\beta^2} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) \tag{10}$$

Q3.2 Conditional distribution of the outcome

The conditional distribution of the outcome is described by a two-sided truncated Gaussian. This is computed using the unnormalized form of Bayes' theorem, where the conditional probability of t serves as prior, and the result y conditions the random variable t as likelihood:

$$p(t|s_1, s_2, y) \propto p(t|s_1, s_2) p(y|t) \tag{11}$$

$$\hookrightarrow p(t|s_1, s_2, y) = \begin{cases} \mathcal{N}(t; s_1 - s_2, \beta^2) & \text{if } a > t > b \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

With a = 0, b = ∞ if player 1 wins ($y = 1$) and a = $-\infty$, b = 0 if player 2 wins ($y = -1$).

Q3.3 Marginal probability that player 1 wins

We have that $p(y = 1) = p(t > 0)$ and with *Corollary 2* (1) marginalize out s_1 and s_2 from $p(t, s_1, s_2)$ to get:

$$p(t) = \mathcal{N}(t; \mu_t, \Sigma_t) \tag{13}$$

where:

$$\mu_t = A\mu = \mu_{s1} - \mu_{s2} \tag{14}$$

$$\Sigma_t = \Sigma_{t|s} + A\Sigma A^T = \beta^2 + \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} \sigma_{s1}^2 & 0 \\ 0 & \sigma_{s2}^2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \beta^2 + \sigma_{s1}^2 + \sigma_{s2}^2 \tag{15}$$

To calculate $p(t > 0)$ we get:

$$p(y = 1) = p(t > 0) = \int_0^\infty p(t) dt = \int_0^\infty \mathcal{N}(t; \mu_{s1} - \mu_{s2}, \beta^2 + \sigma_{s1}^2 + \sigma_{s2}^2) dt \tag{16}$$

Q4 A first Gibbs sampler

An extreme initial value was used to visualize the burn-in period of the Gibbs sampler. At $t = 1000$ the samples in Figure 2 show the stationary distribution starting after about 30 iterations. Although a more extreme guess doesn't alter the convergence rate, 50 iterations were chosen for safety.

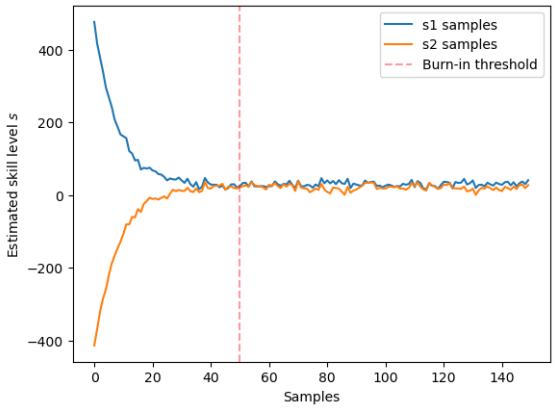

Figure 2: Burn-in period of Gibbs sampler for $y = 1$, initial guess: $t = 1000$.

The samples approximated by the Gibbs sampler should represent the *TrueSkill* skills as Gaussian random variables. Their corresponding sample mean and standard deviations are used to plot the approximated posterior distribution of the skills in Figure 3. As seen in the figure, s_1 had a higher mean than s_2 which is due to approximating with $y = 1$ (i.e. that player 1 (s_1) wins).

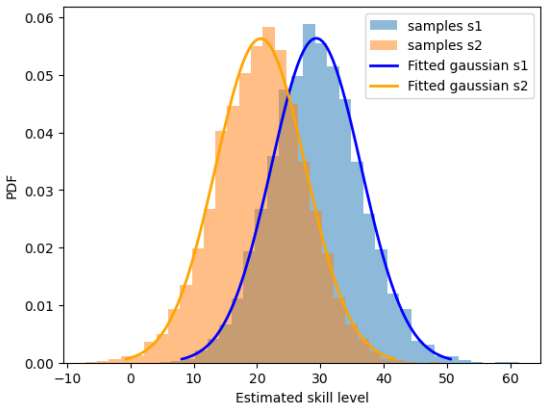

Figure 3: The approximated samples s_1 and s_2 with their corresponding Gaussian distributions.

To approximate the posterior $p(s_1, s_2|y = 1)$, 10,000 samples were generated, as shown in Figure 4, which appears sufficient to accurately approximate the Gaussian posterior within a reasonable time frame. The shared prior of s_1 and s_2 is included in the figure. The posterior for s_1 shifts right of the prior, and s_2 shifts left, indicating player one has a higher probability of winning. The posterior's variance for both players is reduced compared to the prior and will be used for future matches. Appendix A shows the tradeoff between accuracy and computational time, where generating 100,000 samples takes over 25 seconds (up to 70 seconds on some of the authors' computers), while 100 samples fail to accurately estimate the posterior.

Q5 Assumed Density Filtering

To apply the Bayesian model using the Gibbs sampler, the `SerieA` dataset was utilized. As preprocessing all games that ended in a draw were removed, and a new column "y" was created, denoting if team 1 or team 2 won. For all the Italian football teams the *TrueSkill* default parameter values for skill level and uncertainty were initialized at $\mu = 25$ and $\sigma = 25/3 = 8.33$ (2). By feeding the Gibbs Sampler with a match and its winner as input, two new discrete Gaussians are created, which are then used to estimate new values of μ and σ and update each team's skill and uncertainty rating.

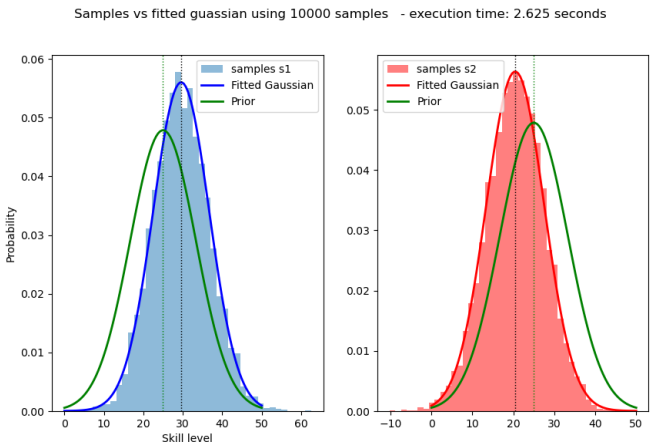

Figure 4: Histogram of s_1 and s_2 together with their shared prior and approximated posterior (A gaussian distribution). Vertical lines indicate the mean for prior and posterior.

Q5.1 Sequential Dataset

Iterating through the original ordering of the `SerieA` dataset, the new skills and uncertainty levels were computed using the Gibbs sampler with 10000 iterations and a burn-in period of 50 samples. The final rankings are shown in Figure 5, where Juventus is placed highest with $\mu \approx 27.9$ and $\sigma \approx 1$, while Chievo is placed lowest with $\mu \approx 21.8$ and $\sigma \approx 1$. The final skill variance reflects the rating's certainty, with a lower variance indicating higher certainty and thus creating a narrower Gaussian.

Q5.2 Shuffled Dataset

Shuffling the dataset leads to changes in the results, as each team's updated mean and variance depend on both teams' skill levels before the match. Since uncertainty is highest at the start of the league, early wins have a greater impact on the final rating than later ones. Therefore, the order of matchups significantly influences the final ranking, with teams benefiting from important early wins being ranked higher. As shown in Figure 5, Napoli becomes the highest rated, while Juventus places fourth.

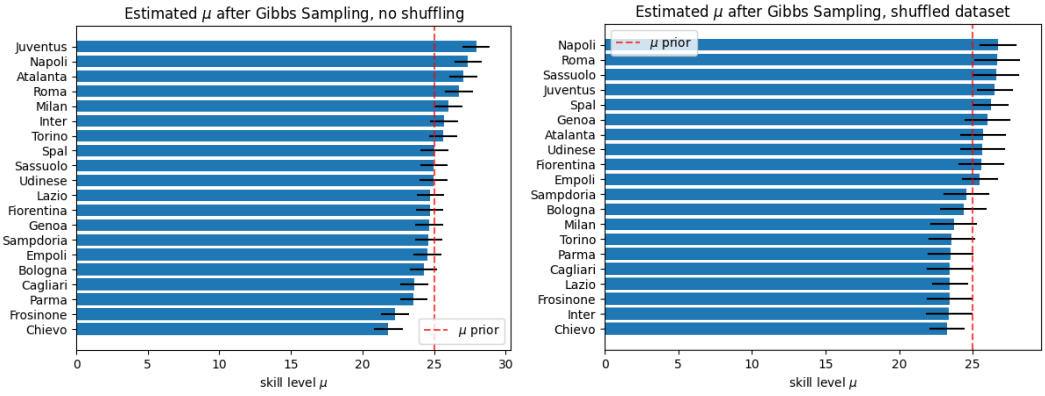

Figure 5: Final ranking of teach teams skill level μ after processing the matches sequentially (left) and in shuffled order (right). The error bars represent each team's respective uncertainty σ.

Q6 Using the model for predictions

With the *TrueSkill* model implementation built on the Assumed Density Filtering (ADF), a predictive *one-step-ahead* model was created using a simple binary predictor. This predictive model assumes that the team with the highest skill level μ will always win the match. This model is integrated

with the ADF, such that the prediction is made first, and secondly the team ratings are updated. As all teams share the same skill at the start, it's assumed that the home team has an advantage and wins these matches. A naive predictive model, which assumes that the home team always wins, was implemented as a control, to highlight potential bias in the dataset. Both model's accuracies are presented in Table 1.

Table 1: Accuracy of the TrueSkill model and Naive model, Serie A dataset

Number of games (no draws)	**Correct predictions**	**TrueSkill model (*r*)**	**Naive model**
272	196	0.721	0.61

Q7 Factor Graph

The factor graph of the model is represented by Figure 6, which is also accompanied by the messages to compute $p(t|y)$. From the factor graph, the joint distribution can be described as: $p(s_1, s_2, t, y) = f_{s_1}(s_1) f_{s_2}(s_2) f_{s_1,s_2,t}(s_1, s_2, t) f_{t,y}(t, y)$, where, $f_{s_2}(s_2) = p(s_2)$, $f_{s_1,s_2,t}(s_1, s_2, t) = p(t|s_1, s_2) = \mathcal{N}(t; s_1 - s_2, \beta^2)$, $f_{t,y}(t, y) = \mathbf{1}_{\{y=sign(t)\}}$ and $f_y(y) = \mathbf{1}_{\{y=y_{obs}\}}$ where y_{obs} is the observed value of y, either 1 or -1.

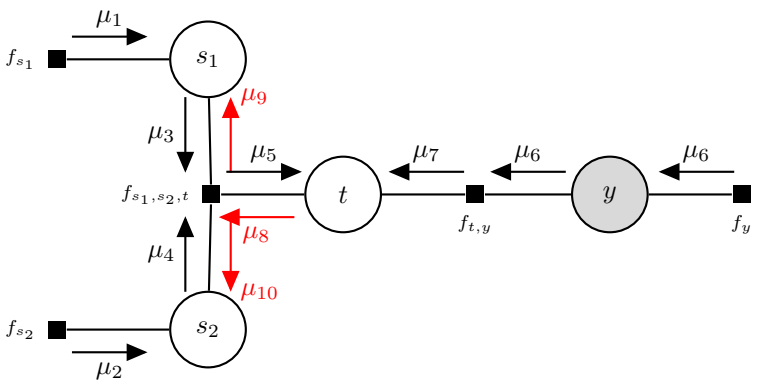

Figure 6: The factor graph of the model accompanied by the messages (in black) to compute $p(t|y)$

The given factor graph with messages colored in black in Figure 6 indicates that $p(t|y) = \mu_5(t)\mu_7(t)$. Later, $p(s|y)$ is calculated via the additional red messages. The explicit messages are stated in the next chapter.

Q8 A message-passing algorithm

This chapter explicitly states and explains the messages given in Figure 6. Beginning from the top left, these messages are given by:

$$\mu_1(s_1) = f_{s_1} = \mu_3(s_1) = p(s_1), \quad \mu_2(s_2) = f_{s_2} = \mu_4(s_2) = p(s_2),$$

$$\mu_5(s_1, s_2, t) = \iint f_{s_1,s_2,t}(s_1, s_2, t)\mu_3(s_1)\mu_4(s_2)\, ds_1\, ds_2 =$$

$$\int p(t|\mathbf{s})p(\mathbf{s})\, d\mathbf{s} = \mathcal{N}(t; \mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2 + \beta^2),$$

$$\mu_6(t) = \mathbf{1}_{\{y=y_{obs}\}},$$

$$\mu_7(y) = \sum_y f_{t,y}\mu_6 = \sum_y \mathbf{1}_{\{y=sign(t)\}}\mathbf{1}_{\{y_{obs}=y\}} = \mathbf{1}_{\{t>0 \text{ if } y_{obs}=1 \text{ else } t<0\}}$$

Where μ_5 is from Chapter 2.3.3 and μ_7 involves a truncated (non)-Gaussian, we approximate $p(t|y)$ with a Gaussian using moment matching: $p(t|y) \approx \hat{q}(t)$, where $\hat{q}(t)$ has the same mean and variance as $p(t|y)$. The distribution of $\mathbf{s}$ becomes: $p(\mathbf{s}|y) = p(\mathbf{s}|t)p(t|y) \approx p(\mathbf{s}|t)\hat{q}(t)$.

To compute the posterior, we need the explicit forms of messages μ_8, μ_9, and μ_{10}, along with those in black in Figure 6, to back-propagate to s_1 and s_2. μ_8 is obtained by performing Gaussian division on $\hat{q}(t)$ (outgoing message) over μ_5 (incoming message).

$$\mu_8(t) \propto \frac{\hat{q}(t)}{\mu_5(t)} = \mathcal{N}(t; \frac{\hat{\mu}_q\sigma_5^2 - \mu_5\hat{\sigma}_q^2}{\sigma_5^2 - \hat{\sigma}_q^2}, \frac{\hat{\sigma}_q^2\sigma_5^2}{\sigma_5^2 - \hat{\sigma}_q^2}) \tag{17}$$

Where $\mu_8(t)$ will be expressed as $\mu_8(t) = \mathcal{N}(t; \hat{\mu}_t, \hat{\sigma}_t)$. The two remaining messages μ_9 and μ_9 are obtained by computing the product of the incoming messages and the factor itself, except the message coming from the target variable, which in this case are s_1 and s_2 respectively, thus:

$$\mu_9(s_1) = \iint \mathcal{N}(t; s_1 - s_2, \beta^2)\mu_4(s_2)\mu_8(t)\, dt\, ds_2 \tag{18}$$

$$\propto \iint \mathcal{N}(t + s_2; s_1, \beta^2)\mu_4(s_2)\mu_8(t)\, dt\, ds_2 \tag{19}$$

$$\propto \iint \mathcal{N}(s_1; t + s_2, \beta^2)\mu_4(s_2)\mu_8(t)\, dt\, ds_2 \tag{20}$$

Given $t \sim \mathcal{N}(t; \hat{\mu}_t, \hat{\sigma}_t^2)$, and $s_2 \sim \mathcal{N}(s_2; \mu_{s_2}, \sigma_{s_2}^2)$, the resulting distribution can be expressed as $(t + s_2) \sim \mathcal{N}(t + s_2; \hat{\mu}_t + \mu_{s_2}, \hat{\sigma}_t^2 + \sigma_{s_2}^2)$. Using this expression to continue from equation (20):

$$= \iint \mathcal{N}(s_1; \hat{\mu}_t + \mu_{s_2}, \hat{\sigma}_t^2 + \sigma_{s_2}^2 + \beta^2)\mu_4(s_2)\mu_8(t)\, dt\, ds_2 \tag{21}$$

$$= \mathcal{N}(s_1; \hat{\mu}_t + \mu_{s_2}, \hat{\sigma}_t^2 + \sigma_{s_2}^2 + \beta^2) \tag{22}$$

By utilizing Gaussian shifting and Gaussian scaling in combination with the distributions for t and s_2 the explicit form for $\mu_9(t)$ has been derived. $\mu_{10}(s_2)$ was derived through the same procedure, and resulted in $\mu_{10}(s_2) = \mathcal{N}(s_2; \mu_{s_1} - \hat{\mu}_t, \hat{\sigma}_t^2 + \sigma_{s1}^2 + \beta^2)$.

Lastly, the approximate posterior of s_1 and s_2 from the message-passing algorithm is given via Gaussian multiplication: $p(s_1|y = y_{obs}) \propto \mu_3(s_1)\mu_9(s_1|y)$ and $p(s_2|y = y_{obs}) \propto \mu_4(s_2)\mu_{10}(s_2|y)$.

The result can be seen in Figure 7, which compares approximating the posterior for s_1 and s_2 given that player one has won the match. The Gibbs sampler and the message-passing algorithm with the generated samples converge well.

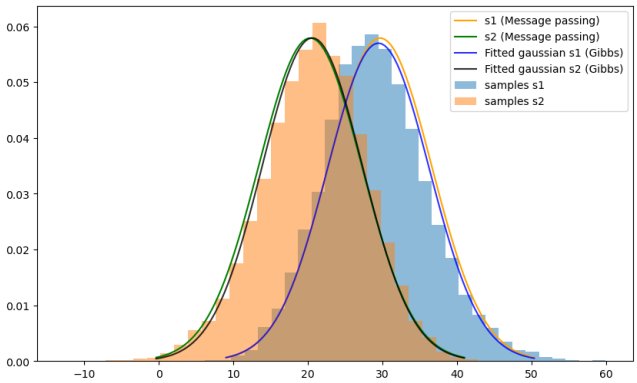

Figure 7: A plot for comparing the posterior formed from the message-passing algorithm and the Gibbs sampler with a histogram of the generated samples.

Q9 Our own data

To test the *TrueSkill* model in a different context, it was applied to rate the skills of fighters in the mixed martial arts tournament *Ultimate Fighting Championship* (UFC). Two relevant datasets were sourced from the machine learning community website *Kaggle*: one containing historic match-ups of fighters from 1993 to 2021 (3), and the other detailing fighter records and characteristics (4). The

first dataset was used for sequentially processing match-ups and predicting with the model, while the second was used for computing a prior skill for each fighter.

Fighters in martial arts are split into weight classes, as weight correlates with characteristics like height, reach, and force output, giving heavier fighters a statistical advantage. Based on this, each fighter's prior skill was computed as $\mu_{\text{prior},i} = 25 \times \frac{w_i}{\bar{w}}$, where w_i is the fighter's weight, and $\bar{w}$ is the average weight. This prior favors heavier fighters and assumes that a heavier fighter would generally beat a lighter, "more skilled" one (if this assumption seems unreasonable, watch *this*). UFC match-ups are strictly within weight classes, so the model only favors heavier fighters within the same class. Prior uncertainty and variance was set to the *TrueSkill* default of $\sigma_{\text{prior}} = \frac{25}{3}$, and $\beta = \frac{25}{6}$.

The necessary pre-processing of the historic match-ups dataset consisted of removing all draws and creating a new column y specifying which fighter won the match. The *one-step ahead* predictive model from Chapter Q6 with ADF was then utilized to predict the winner of each fight. The model obtained a fairly poor predictive accuracy of r = 0.510, being on the verge of a random guesser. This could be explained by the unpredictable nature of martial arts, where factors such as varying styles, physical conditions, and referees' judgments can impact fight outcomes. For examples of top- and bottom-ranked fighter's priors and posteriors, see Appendix B.

Q10 Open-ended project extension

To expand the *TrueSkill* model, the deterministic predictive function was replaced with a stochastic predictor. Instead of simply selecting the winner as the player with the highest prior skill mean, the predictor function randomly samples from the Gaussian skill distributions of the two players and declares the player with the highest sample as winner. This takes the variance into account and introduces some stochasticity early in the model, while the uncertainty is still high for most players' ratings. This could lead to an improvement and make a "smarter" prediction algorithm as it is possible, and not unlikely, that e.g. player 1 who has much higher variance but slightly lower mean compared to player 2, could beat player 2 in a match. The previous prediction algorithm only considered the mean and thus, in the previously mentioned scenario, we would always predict that player 2 beats player 1. However, this stochasticity also makes the model's performance differ slightly between runs. With the stochastic predictor, the accuracy for the UFC dataset was improved, but for the Serie A dataset, performance became worse. The different accuracies for both datasets with both predictor types are presented in Table 2.

Table 2: Comparing accuracies of TrueSkill model with a deterministic or stochastic predictor

Dataset	**Deterministic, accuracy (*r*)**	**Stochastic, accuracy (*r*)**
Serie A	0.721	0.618
UFC	0.510	0.532

3 Discussion and Conclusions

The model's accuracy with the UFC dataset is a bit disappointing, with both the predictor versions being close to 50%. As previously discussed in Chapter Q9, UFC is a sport with many random factors where it is not certain that the higher-rated player wins. To account for this randomness in the sport, the hyperparameter β could be adjusted to a higher value, which might have given better results. With a higher β, it's likelier that an underdog wins due to luck, which reflects a sport such as mixed martial arts better (compared to e.g. chess). Due to the code's extensive running time, this was not tried.

The size and the time span of the Serie A and UFC datasets are drastically different. The UFC dataset captures every fight from 1993-2021 whilst the Serie A dataset only captures 1 year of matches. This drastic difference makes the datasets incomparable and could be the cause for the low accuracy using the UFC dataset as it captures the variance across entire careers of fighters, with valleys and peaks.

In conclusion, the results using the Serie A dataset are satisfactory with an accuracy of 0.721 in the deterministic version. However, the results for the UFC dataset are more disappointing with a likely cause being the dataset's quality and format. The results of Q10 led to some uncertainty regarding the model, and the predictor may give better results but also worse results depending on the dataset.

References

[1] N. Wahlström, *Advanced Probabilistic Machine Learning, Lecture 2 - Conjugate prior, Binomial-Beta pair, Multivariate Gaussian*. 2024.

[2] R. Herbrich, T. Minka, and T. Graepel, "Trueskill™: A bayesian skill rating system," in *Advances in Neural Information Processing Systems 20*, pp. 569–576, MIT Press, January 2007.

[3] R. Warrier, "Ufc-fight historical data from 1993 to 2021." `https://www.kaggle.com/datasets/rajeevw/ufcdata`. Accessed: 2024-10-01.

[4] Asaniczka, "Ufc fighters' statistics dataset." `https://www.kaggle.com/datasets/asaniczka/ufc-fighters-statistics`. Accessed: 2024-10-01.

Appendices

A Q4 Burn-in runtime

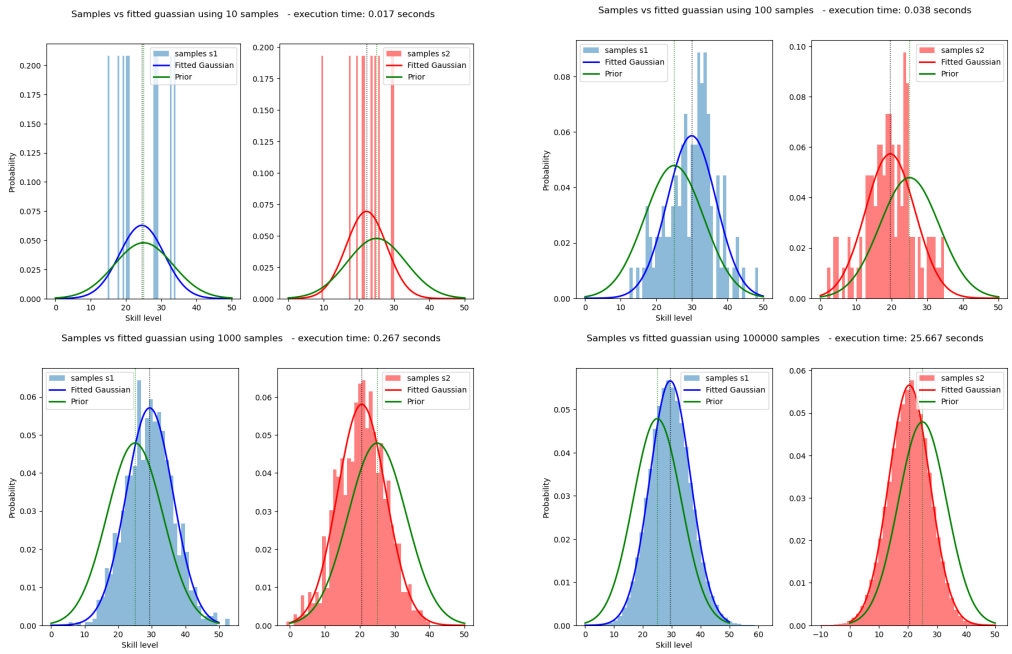

Figure 8: Histogram of samples with their prior and approximated posterior. Vertical lines indicate the mean for prior and posterior. Execution time in the headings.

B Q9 UFC Dataset

Table 3: Skill ratings and uncertainty for the top 5 highest rated (i.e. heaviest) fighters at initialization and after processing the dataset. These fighters all belong to the weight class super heavyweight (which no longer exists in the UFC).

(a) Priors

Fighter name	**prior** μ	**prior** σ
Emmanuel Yarborough	111.96	8.33
Thomas Ramirez	59.61	8.33
John Matua	58.16	8.33
Koji Kitao	50.89	8.33
Andre Roberts	50.16	8.33

(b) Posteriors

Fighter name	**posterior** μ	**posterior** σ
Emmanuel Yarborough	85.80	2.38
Thomas Ramirez	51.57	2.40
John Matua	50.34	2.48
Andre Roberts	48.75	1.64
Dan Bobish	46.01	1.55

Table 4: Skill ratings and uncertainty for the top 5 lowest rated (i.e. lightest) fighters at initialization and after processing the dataset. These fighters all belong to the weight class strawweight.

(a) Priors

Fighter name	**prior** μ	**prior** σ
Polyana Viana	16.72	8.33
Amanda Cooper	16.72	8.33
Cristina Stanciu	16.72	8.33
Mackenzie Dern	16.72	8.33
Livinha Souza	16.72	8.33

(b) Posteriors

Fighter name	**posterior** μ	**posterior** σ
Jodie Esquibel	14.38	1.05
Alex Chambers	14.72	1.00
Emily Kagan	14.75	1.58
Chan-Mi Jeon	14.77	1.57
Hannah Cifers	14.87	0.99