Group 15: Mini Project Report

September 29, 2020

1 Introduction

In this mini-project we develop a Gaussian approach for keeping track of the skill-rating of players or teams, based on their wins or losses, known as Microsoft's *Truekskill*. We define a Bayesian model for skill-ratings and outcome of the match and apply this in a Gibbs sampler and message-passing algorithm to update the skills. We extend the model by taking score difference into account. The model is applied on Italian football league Serie A and tennis Grand Slam tournaments.

2 Bayesian Network and Modeling (Q.1 and Q.2)

The Bayesian model for one match between two players is the joint distribution of all the concerned random variables. The model consists of the following four random variables: the skill of player 1, $s_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$; the skill of player 1, $s_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$; the outcome of the game $t \sim \mathcal{N}(s_1 - s_2, \sigma_t^2)$; and the result of the game $y = sign(t)$. Using the *Trueskill* model, the skill of players can be updated according to a bayesian framework. In short, the skills contribute to and are updated by the match outcome (win or lose), which can in a bayesian network be modelled as shown in Figure 1.

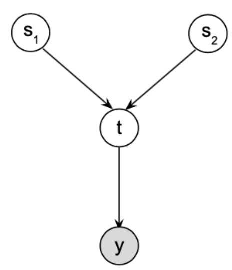

Figure 1: Bayesian network of the skills of player 1 and 2 and outcome of the game.

In accordance with the bayesian network in Figure 1, the joint distribution of all the random variables is found to be as presented in equation (1) below.

$$p(s_1, s_2, t, y) = p(s_1)p(s_2)p(t|s_1, s_2)p(y|t) \tag{1}$$

From the bayesian network of the model $p(s_1, s_2, t, y)$, two conditionally independent sets of variables were identified. From Figure 1 and shown in (2), s_1 and s_2 are *head-to-head* nodes which are independent if the node between them and any of its descendent are not observed (according to slide 21 in lecture 3). Furthermore, presented in (3), y and s_1, s_2 are *head-to-tail* nodes which are independent if the node between them is observed (according to slide 22 in lecture 3).

$$\begin{aligned} p(s_1, s_2) = \int p(s_1, s_2, t, y)dtdy = \int p(y|t)p(t|s_1, s_2)p(s_1)p(s_2)dtdy \\ = p(s_1)p(s_2) \implies s_1 \perp\!\!\!\perp s_2 \mid \varnothing \end{aligned} \tag{2}$$

$$
\begin{aligned}
p(y, s_1, s_2|t) &= \frac{p(s_1, s_2, t, y)}{p(t)} = \frac{p(y|t)p(t|s_1, s_2)p(s_1)p(s_2)}{p(t)} \\
&= p(y|t)p(s_1, s_2|t) \implies y \perp\!\!\!\perp s_1, s_2 \mid t
\end{aligned} \tag{3}
$$

3 Computing with the model (Q.3)

Given the model presented above, the full conditional distributions of the skills $p(s_1, s_2|t, y)$ and game outcome $p(t|s_1, s_2, y)$ can be computed. First, when computing the full conditional distribution of the skills, y can is redundant as it is conditionally independent from s_1 and s_2, given t.

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

According to the assignment formulation, the conditional probability $p(t|s_1, s_2)$ and the multivariate distribution $p(s_1, s_2)$ are given by the following Gaussian distribution.

$$
p(t|s_1, s_2) \sim \mathcal{N}(t; [1 \quad -1]\, x_{s_1,s_2}, \Sigma_{t|s_1,s_2}) \tag{5}
$$

$$
p(s_1) \cdot p(s_2) = p(s_1, s_2) \sim \mathcal{N}(x_{s_1,s_2}; \mu_{s_1,s_2}, \Sigma_{s_1,s_2}) \tag{6}
$$

Where $x_{s_1,s_2} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}; \mu_{s_1,s_2} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}; \Sigma_{s_1,s_2} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$.

Using *Corollary 1*, presented in slide 28 of the lecture 2, the conditional distribution $p(s_1, s_2|t)$ can be computed as follows:

$$
p(s_1, s_2|t) \sim \mathcal{N}(\begin{bmatrix} s_1 \\ s_2 \end{bmatrix}; \mu_{s_1,s_2|t}, \Sigma_{s_1,s_2|t}) \tag{7}
$$

Where

$$
\Sigma_{s_1,s_2|t} = \frac{1}{\sigma^2_{t|s_1,s_2} + \sigma_1^2 + \sigma_2^2} \begin{bmatrix} \sigma_1^2 \cdot (\sigma^2_{t|s_1,s_2} + \sigma_2^2) & \sigma_1^2 \cdot \sigma_2^2 \\ \sigma_1^2 \cdot \sigma_2^2 & \sigma_2^2 \cdot (\sigma^2_{t|s_1,s_2} + \sigma_1^2) \end{bmatrix}
$$

and

$$
\mu_{s_1,s_2|t} = \Sigma_{s_1,s_2|t} \cdot \begin{bmatrix} \frac{\mu_1}{\sigma_1^2} + \frac{t}{\sigma^2_{t|s_1,s^2}} \\ \frac{\mu_2}{\sigma_2^2} + \frac{t}{\sigma^2_{t|s_1,s^2}} \end{bmatrix}
$$

.

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

$$
p(t|s_1, s_2, y) = \frac{p(s_1)p(s_2)p(t|s_1, s_2)p(y|t)}{p(s_1, s_2, y)} \propto p(t|s_1, s_2)p(y|t) \tag{8}
$$

The distribution $p(t|s_1, s_2, y)$ is a truncated Gaussian because it is the product of a Gaussian distribution $p(t|s_1, s_2)$, and a dirac-function which is non-zero only when $sign(t) = sign(y)$. For the case of $y = 1$, when s_1 wins, the bounds are between lower bound $a = 0$ and upper bound $b = \inf$, as t is then within the bound of being positive. When s_2 wins, $y = -1$, the bounds are between $a = -\inf$ and $b = 0$ accordingly.

A part of the problem formulation is to find the marginal probability that Player 1 wins the game, $p(y = 1)$. This is obtaineed from finding $p(y = 1) = p(t > 0)$ where $p(t)$ is the Gaussian distribution obtained from marginalizing s_1, s_2 from $p(t, s_1, s_2)$ in accordance with *Corollary 2*, presented in slide 34 of the lecture nodes of this years course. We know the distribution of t_{s_1,s_2} from the previous calculations, namely $p(t|s_1, s_2) \sim \mathcal{N}(t; \mu_{t|s_1,s_2}, \Sigma_{t|s_1,s_2})$. With this we can calculate the probability that player 1 wins as follows.

$$
p(y = 1) = p(t > 0) = 1 - p(t < 0) = 1 - \Phi(\frac{0 - \mu_t}{\sigma_t}) \tag{9}
$$

4 A first Gibbs sampler (Q.4)

In Gibbs sampling, the target function is $\pi(s_1, s_2) = p(s_1, s_2|y)$. This is acquired through sampling the multivariate and truncated Gaussian distributions presented below given in (7) and (8) as $\pi(s_1, s_2|t) \sim \mathcal{N}(s_1, s_2; \mu_{s_1,s_2|t}, \Sigma_{s_1,s_2|t})$ and $\pi(t|s_1, s_2) \sim \mathcal{TN}(t; s_1 - s_2, \Sigma_{t|s_1,s_2})$ respectively. The posterior distribution of the skills is the stationary distribution of the Gibbs sampler, after the burn-in period. In the figures below we found that stationary distribution, when $y = 1$, by sampling 1000 samples and plotted the first 100. We used the following hyperparameters for the prior:

$$\mu_{s_1}=\mu_{s_2}=25,\sigma^2_{s_1}=\sigma^2_{s_2}=(\tfrac{25}{3})^2,\sigma^2_{t|s_1,s_2}=(\tfrac{25}{6})^2 \tag{10}$$

The parameters in (10) produced the plot in Figure 2, from which we found the burn-in to be 25. For the re-run of the experiment, we discarded the first 25 samples (the burn-in).

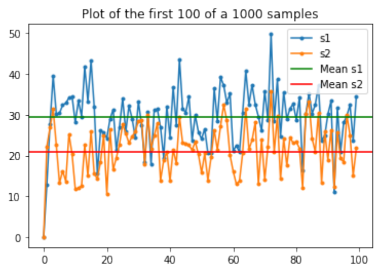

Figure 2: Plot of the first 100 of 1000 samples (with burn-in)

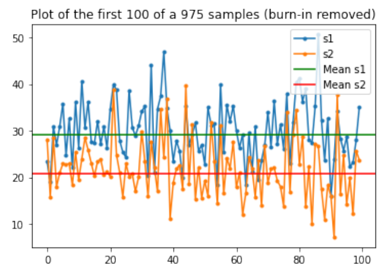

Figure 3: Plot of the first 100 of 975 samples (burn-in removed)

The choice of burn-in seems to reasonable, since all the extreme outliers have been removed. Deciding how many samples to use is a matter of trade-off between estimation accuracy and time of computation. We sampled with sizes of 400, 800, 1600 and 3200. When using 400 samples, we could see from Figure 4 that the approximated Gaussian did not fit a Gaussian distribution perfectly in the histogram. When using a sample size of 3200 the histogram will fit within the Gaussian distribution nicely.

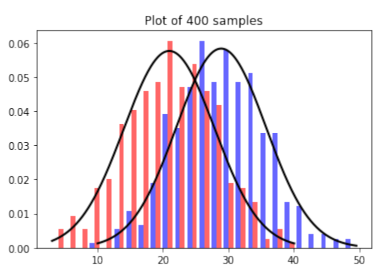

Figure 4: Plot of 400 samples sampled with the Gibbs sampler

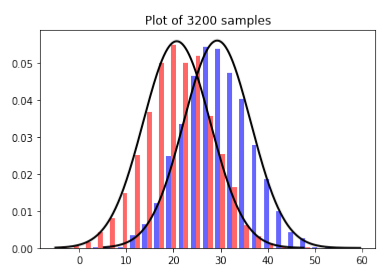

Figure 5: Plot of 3200 samples sampled with the Gibbs sampler

However, when using 3200 samples, the computational time was a lot longer. We found that 1000 samples was a good trade-off between time and estimation, as the sample's approximation fitted a Gaussian distribution good enough without being too computationally heavy. When executing a Gibbs sample test-run, we see that the mean value of the posterior distribution, given that s_1 won, is higher than it's prior distribution. This is intuitive, as if s_1 wins, it should probably get a higher skill rating. A similar pattern is observed in the comparison of the prior and posterior of s_2. The posterior has a lower mean value, as s_2 lost and will get a lower skill value.

5 Assumed Density Filtering (Q.5)

The posterior distribution of the skills given a specific match from Q.4 will now be used as a prior for the next match according to an assumed density filtering implementation. A stream of matches will be generated, always using the posterior distribution from the previous match as the prior for the next match.

Starting with initial skill value $\mu_{s_i} = 25$ and variances $\sigma^2_{s_i|t} = (\frac{25}{3})^2$, $\sigma^2_{t|s_i} = (\frac{25}{12})^2$, a trained model is obtained with skills ranging from 21.24 to 27.14 and a variance from 0.42 to 1.44. When compared to the real results of the Serie A 2018/2019 season, it is not a perfect predictor of final standings. However, the model successfully predicts what part of the table the teams would end up in. For example, the top five and bottom five teams of our skill-table are also the top five and bottom five of the Serie A result table for that season. All of the teams final rankings are within being obtainable as the skill levels are not far away within the first standard deviation. The variance after all matches in the data set has significantly decreased to a value around 1. This is as expected, as the skills become more solidified for each game played, and analogously model updated.

The result are presented in order in the following regime. {Team: skill (variance)}.
Milan: 27.14 (0.77) | Inter: 27.14 (0.62) | Juventus: 27.07 (0.97) | Napoli: 27.04 (0.80) |
Torino: 26.65 (0.94) | Atalanta: 26.63 (0.55) | Roma: 25.86 (0.62) | Lazio: 25.50 (0.59) |
Sampdoria: 24.53 (0.51) | Spal: 24.41 (0.57) | Bologna: 24.30 (0.51) | Genoa: 24.26 (0.70) |
Udinese: 24.18 (0.70) | Fiorentina: 24.12 (0.65) | Empoli: 24.01 (0.57) | Cagliari: 24.00 (0.42) |
Sassuolo: 23.65 (0.93) | Parma: 23.94 (0.59) | Frosinone: 21.97 (0.75) | Chievo 21.24 (1.44)

When random-shuffling the match order, the individual skills will change. Overall, there is no significant difference in the mean values of the team skills between random shuffling the data and keeping the true match order. However, we see that the variance for all team skills decreases. We don't surely know the reason why the variance is overall small when random shuffling. One theory is that over the season, teams skills evolve or decrease, which means that even if a team has skill 23 it starts to perform as skill 27, and in turn this will imply a larger variance in the growth. When randomly shuffled, the teams skills will probably be more over the whole data set.

6 Using the model for predictions (Q.6)

To evaluate a single-step prediction model, we start by training the model, on 66% of the data, splitting the data set to a training set of size 182 randomly sampled matches and a test set of the remaining 90 matches. The implementation of the *one-step-ahead* function is as follows: before the match is sampled and the model updated, a prediction is made given the current model. The prediction is decided using the cumulative distribution function of $p(t|s_1, s_2, y)$, $\Phi(\frac{\mu_{t|s_1,s_2}}{\sigma^2_{t|s_1,s_2}})$, where $\mu_{t|s_1,s_2} = s1 - s2$ and $\sigma^2_{t|s_1,s_2}$ is are design-values described in Q5. According to (9), the prediction that team 1 wins is $\Phi(\frac{\mu_{t|s_1,s_2}}{\sigma_{t|s_1,s_2}}) > 0.5$. If the equation doesn't hold, the model predicts that team 2 wins. This prediction is made for using an updated model every test match. The performance, or prediction rate, is calculated as $r = \frac{\text{number of correct guesses}}{\text{number of total guesses}}$. The prediction rate is compared to a random guess predictor, taken from the uniform distribution. We obtained a result of $r_{model} \approx 0.71$ and $r_{guess} \approx 0.46$. We see through multiple runs that the model predictions are consistently better than simply randomly guessing.

7 Factor graph (Q.7)

Figure 6 represents the model's *Factor graph* and its messages. In (11) the factor node expressions are found.

$$\begin{gathered} f_a(s_1) = p(s_1);\;\; f_b(s_2) = p(s_2) \\ f_c(s_1, s_2, t) = p(t|s_1, s_2) \sim \mathcal{N};\; f_d(t, y) = p(y|t) = \delta(y = sign(t)) \end{gathered} \tag{11}$$

In (12) the first two messages are found. The explicit form of $\mu_2(t)$ is a non-Gaussian marginal, let's call it $p(t) = \mu_2(t)\mu_7(t)$. This is a truncated Gaussian message which will be approximated with a Gaussian message using *moment-matching* in the next task.

$$\begin{gathered} \mu_1(y) = \delta(y = 1) \\ \mu_2(t) = \sum_y f_d(t, y)\mu_1(y) = \sum_y \delta(y = \text{sign}(t))\delta(y = 1) = \delta(t > 0) \end{gathered} \tag{12}$$

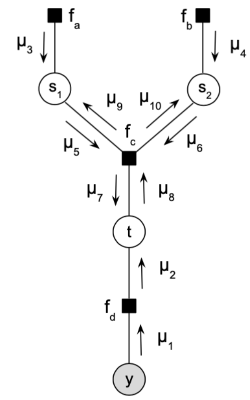

Figure 6: The model's factor graph and its messages

Furthermore, in (13) the messages for $\mu_3 - \mu_6$ are found.

$$\begin{aligned} \mu_3(s_1) = f_a(s_1) = p(s_1) \sim \mathcal{N}(s_1; m_1, \sigma_1^2) \\ \mu_4(s_2) = f_b(s_2) = p(s_2) \sim \mathcal{N}(s_2; m_2, \sigma_2^2) \\ \mu_5(s_1) = \mu_3(s_1) \\ \mu_6(s_2) = \mu_4(s_2) \end{aligned} \tag{13}$$

In (14) the message μ_7 is found by integrating over $f_c(s_1, s_2, t)$ and the incoming nodes.

$$\begin{aligned} \mu_7(t) = \iint f_c(s_1, s_2, t)\mu_5(s_1)\mu_6(s_2) ds_1 ds_2 = \\ = \iint p(t|s_1, s_2)p(s_1)p(s_2)ds_1 ds_2 = \iint \mathcal{N}(t; A \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \sigma_t^2)\mathcal{N}(\begin{bmatrix} s_1 \\ s_2 \end{bmatrix}; \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix})ds_1 ds_2 = \\ = [\text{Corollary 2}] = \mathcal{N}(t; \mu_1 - \mu_2, \sigma_t^2 + \sigma_1^2 + \sigma_2^2) \end{aligned} \tag{14}$$

$\hat{p}(t|y = 1)$ is the approximated Gaussian message of the truncated Gaussian message

$$p(t|y = 1) \propto \mu_2(t)\mu_7(t) = \delta(t > 0)\mathcal{N}(t; \mu_1 - \mu_2, \sigma_t^2 + \sigma_1^2 + \sigma_2^2)$$

$$\mu_8(t) = \frac{\hat{p}(t|y = 1)}{\mu_7(t)} \sim \frac{\mathcal{N}(t; \mu_{t|y}, \sigma_{t|y}^2)}{\mathcal{N}(t; \mu_7, \sigma_7^2)} = \mathcal{N}(t; \frac{\mu_{t|y}\sigma_7^2 - \mu_7\sigma_{t|y}^2}{\sigma_7^2 - \sigma_{t|y}^2}, \frac{\sigma_7^2\sigma_{t|y}^2}{\sigma_7^2 - \sigma_{t|y}^2}) \tag{15}$$

$$\begin{aligned} \mu_9(s_1) = \iint \mu_6(s_2)\mu_8(t)f_c(s_1, s_2, t)dtds_2 = \iint \mathcal{N}(s_2; \mu_6, \sigma_6^2)\mathcal{N}(t; \mu_8, \sigma_8^2)\mathcal{N}(t; A \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \sigma_t^2)dtds_2 = \\ = [\text{Property 1 and 2}] = \iint \mathcal{N}(s_2; \mu_6, \sigma_6^2)\mathcal{N}(t; \mu_8, \sigma_8^2)\mathcal{N}(s_1; [1 \quad -1] \begin{bmatrix} t \\ s_2 \end{bmatrix}, \sigma_t^2)dtds_2 = \\ [\text{Corollary 2}] = \mathcal{N}(s_1; \mu_6 - \mu_8, \sigma_t^2 + \sigma_8^2 + \sigma_6^2) \end{aligned} \tag{16}$$

$\mu_{10}(s_2)$ is calculated in the same manner as $\mu_9(s_1)$.

$$\mu_{10}(s_2) = \iint \mu_5(s_1)\mu_8(t)f_c(s_1, s_2, t)dtds_1 = \mathcal{N}(s_2; \mu_5 - \mu_8, \sigma_t^2 + \sigma_8^2 + \sigma_5^2) \tag{17}$$

8 A message-passing algorithm (Q.8)

By using *message-passing* the truncated Gaussian message μ_2 is approximated with a Gaussian message. The message-passing algorithm generates the posterior distribution of the skills, shown in Figure 7 (blue and coral line). The histogram and the Gaussian approximations (black lines) from Gibbs sampling are also shown in Figure 7. The posteriors look similar, which was the goal for this task.

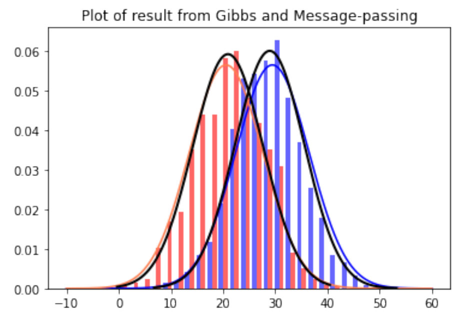

Figure 7: Posteriors from Message-passing and Gibbs sampling

9 Model application on tennis matches (Q.9)

In this section we have tested and evaluated our *Trueskill* models on data from tennis tournaments. The data set (1) that we used contains matches from all the competitions on the ATP tour[1]; however, we decided to only use match data from the four Grand Slam tournaments because in those only the top players compete. Therefore, there are fewer players that get to play more matches against each other. Hence, the skills of players are updated more frequently in the model and the match predictions become more fair. When using the ADF method with Gibbs Sampling the model had a prediction accuracy of $r_{model} \approx 60\%$, which was the same accuracy as the message passing algorithm achieved. However, the computational time was noticeably lower with the message passing algorithm. The models prediction accuracy on tennis matches is higher than a random predictor but lower than what we achieved on Serie A football games. This could be explained by the characteristics of tennis as a sport and the variables that dictates the outcome of a match.

10 Score Difference Extension (Q.10)

Both tennis and soccer matches have a score difference, which can separate close wins (similar skill) from convincing wins. By using data of the scoring difference in matches we could extend our *Trueskill* models to possibly get a better skill indicator and prediction. By calculating the difference of games won by each player in a tennis match we can get a sense of the performance difference in a match. In tennis, the score difference in a match between two players can be denoted as $g_w - g_l$, where g_w is the number of games won by the winning player and g_l is the number of games won by the losing player. The percentage of games won by the winning players is given by delta $\Delta = \frac{g_w}{g_w + g_l}$. A high Δ would indicate a big performance difference between the players. When Δ exceeds 66% we classify that as a convincing win. Similarly, when the goal difference in soccer match exceeds 3 it is a convincing win. On a convincing win, an additional skill update is given by: $s \longleftarrow K\cdot$, where K is the constant that determines the weight of a convincing win. Through iteration, we found the value $K = 1.4$ for tennis and $K = 1.3$ for soccer to be the best prediction rates.
The score difference extension combined with ADF and Gibbs sampling achieved an prediction accuracy of $r_{model,tennis} \approx 65\%$ for tennis and $r_{model,soccer} \approx 66\%$ for the Series A data set. The extension combined with the message passing model did not yield any better results. The extension performs better than the simpler implementation for tennis, but not for soccer. This is likely due to the fact that our extension is a design value for convincing wins given by a static threshold. Dynamic threshold and skill adjustments could be implemented, possibly implementing functions similar to more advanced *Trueskill* or ELO systems. This would probably improve both prediction rates even more. Also, other variables such as court surface or weather forecast which are values affecting teams or players skill.

[1]The ATP tour is a tennis series for men with about 60-70 tournaments in 31 different countries. There are about 1800 players ranked on the ATP-tour.(2)

References

[1] Dataset, “Atp matches 2019,” accessed 28 September 2020. [Online]. Available: https://github.com/JeffSackmann/tennis_atp/blob/master/atp_matches_2019.csv

[2] Atp tour, “2019 atp world tour calendar,” accessed 28 September 2020. [Online]. Available: https://www.atptour.com/en/news/atp-announces-2019-atp-world-tour-calendar

11 Appendix

The python code implementations for this miniproject is attached as a .zip file.