Predicting Game Results in Football and League of Legends using Probabilisitic Machine Learning Tools

Gabriel Y. Arteaga
Uppsala University
gabrielyanci.arteaga.6822@student.uu.se

Inga Wohlert
Uppsala University
inga.wohlert.8507@student.uu.se

Alexander Sabelström
Uppsala University
alexander.sabelstrom.1040@student.uu.se

Introduction

In this project we are using the TrueSkill rating system, to represent player skill levels dynamically using data sourced from the Italian Serie A in Football and then later applying the model to data from the NA LCS, the North American League of Legends series. League of Legends is an online multiplayer game, with a ever growing fan base and is highly established in the Esport Scene. Similarly to football, it features two competitive teams. As tools, we are building a Bayesian model and are using Gibbs Sampling with Assumed Density Filtering to draw inferences. Furthermore, we are building a Factor Graph and are implementing the Message Passing algorithm to both verify and repeat our predictions.

Q.1

A model in Bayesian parlance is a joint distribution of all included random variables. The model is therefore a joint distribution as such:

$$p(s_1, s_2, t, y)$$

The skills of the players, s_1 and s_2, are assumed to be independent of each other. *Trueskill* characterises its belief on a player's skill using a bell-curve belief distribution [1]. Each player's skill is therefore described by a normal distribution:

$$\mathcal{N}(\mu_{s_1}, \sigma^2_{s_1}) \text{ and } \mathcal{N}(\mu_{s_2}, \sigma^2_{s_2})$$

represented with the parameters μ and σ. The outcome of the game t is a normal distribution with the mean $s_1 - s_2$ and therefore dependent on both s_1 and s_2. Lastly, the result of the game y, is dependent on the outcome of the game t, as y is defined as $sign(t)$.

The joint distribution therefore becomes:

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

Q.2

Given the joint distribution in Q.1, we are getting the following Bayesian Network, in which t is given. Using the head-to-tail rule, we obtain the following independencies.

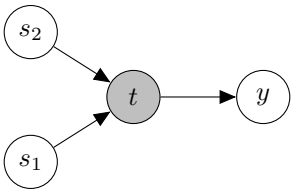

Figure 1: Bayesian network with nodes s_1, s_2, t, and y.

$s_1 \perp y \mid t$ Yes, because $s_1 \rightarrow t \nrightarrow y$ (observed head-to-tail)
$s_2 \perp y \mid t$ Yes, because $s_2 \rightarrow t \nrightarrow y$ (observed head-to-tail)

Q.3

Full conditional distribution of skills

We want to find the full conditional distribution of the skills $p(s_1, s_2|t, y)$. We start by expanding the expression and attempt to remove y from the expression.

$$\begin{aligned} p(s_1, s_2|t, y) &\underset{(7)}{=} \frac{p(t, y|s_1, s_2)p(s_1, s_2)}{p(t, y)} \underset{(6)}{=} \frac{p(s_1, s_2, t, y)}{p(t|y)p(y)} \\ &\underset{(1)}{=} \frac{p(s_1)p(s_2)p(t|s_1, s_2)p(y|t)}{p(t|y)p(y)} \end{aligned}$$

Using the fact that s_1, s_2 are independent we can rewrite their probabilities as a joint probability.

$$p(s_1)p(s_2) = p(s_1, s_2)$$

Further, using the independence of t and y we can remove their dependencies in the equation.

$$\frac{p(s_1, s_2)p(t|s_1, s_2)}{p(t)} \underset{(7)}{=} p(s_1, s_2|t) \tag{2}$$

Now, given the previous result we can use the Affine Transformation to find the probability function of $p(s_1, s_2|t)$. Let us define our corresponding variables for the transformation, let $x_a = [s_1 \quad s_2]$ and $x_b = t$. Then we can express our probability function as follows:

$$p(s_1, s_2|t) = \mathcal{N}\left(s_1, s_2; \mu_{s_1,s_2|t}, \Sigma_{s_1,s_2|t}\right)$$

where

$$\begin{aligned} \Sigma_{s_1,s_2|t} &\underset{(9)}{=} \left(\Sigma^{-1}_{s_1,s_2} + A^T \Sigma_{t|s_1,s_2} A\right)^{-1} \\ &= \left(\begin{bmatrix} \sigma^2_{s_1} & 0 \\ 0 & \sigma^2_{s_2} \end{bmatrix}^{-1} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} \frac{1}{\sigma^2_t} \begin{bmatrix} 1 & -1 \end{bmatrix} \right)^{-1} \\ &= \frac{1}{(\frac{1}{\sigma^2_{s_1}} + \frac{1}{\sigma^2_t})(\frac{1}{\sigma^2_{s_2}} + \frac{1}{\sigma^2_t}) - \frac{1}{\sigma^4_t}} \begin{bmatrix} \frac{1}{\sigma^2_{s_2}} + \frac{1}{\sigma^2_t} & \frac{1}{\sigma^2_t} \\ \frac{1}{\sigma^2_t} & \frac{1}{\sigma^2_{s_1}} + \frac{1}{\sigma^2_t} \end{bmatrix} \end{aligned}$$

and

$$\begin{aligned} \mu_{s_1,s_2|t} &\underset{(8)}{=} \Sigma_{s_1,s_2|t} \left(\Sigma^{-1}_{s_1,s_2} \mu_{s_1,s_2} + A^T \Sigma^{-1}_{t|s_1,s_2} (t - b)\right) \\ &= \Sigma_{s_1,s_2|t} \left(\begin{bmatrix} \frac{1}{\sigma^2_{s_1}} & 0 \\ 0 & \frac{1}{\sigma^2_{s_2}} \end{bmatrix} \begin{bmatrix} \mu_{s_1} \\ \mu s_2 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} \frac{t}{\sigma^2_t} \right) \\ &= \Sigma_{s_1,s_2|t} \begin{bmatrix} \frac{\mu_{s_1}}{\sigma^2_{s_1}} + \frac{t}{\sigma^2_t} \\ \frac{\mu_{s_2}}{\sigma^2_{s_2}} - \frac{t}{\sigma^2_t} \end{bmatrix} \end{aligned}$$

Full conditional distribution of outcome

We want to find the full conditional distribution of the outcome $p(t|y, s_1, s_2)$. We start by applying bayes' theorem.

$$\begin{aligned} p(t|y, s_1, s_2) &\underset{(7)}{=} \frac{p(y, s_1, s_2|t)p(t)}{p(y, s_1, s_2)} \\ &\underset{(6)}{=} \frac{p(s_1, s_2, y, t)}{p(y, s_1, s_2)} \\ &\underset{(1)}{=} \frac{p(s_1)p(s_2)p(t|s_1, s_2)p(y|t)}{p(y|s_1, s_2)p(s_1, s_2)} \end{aligned}$$

Now, using the same fact that s_1, s_2 are independent we get the following expression.

$$\begin{aligned} &\frac{p(t|s_1, s_2)p(y|t)}{p(y|s_1, s_2)} \\ &\propto p(t|s_1, s_2)p(y|t) \end{aligned}$$

Since the probability is not allowed to be less than 0 and we have been given that a draw is not a possible outcome, this indicates that t must be strictly greater that 0. Hence, we end up with the following truncated Gaussian.

$$p(t|y, s_1, s_2) \propto p(t|s_1, s_2)p(y|t) = \begin{cases} \mathcal{N}(t; \mu_t, \sigma_t^2) & \text{if } a < t < \infty \\ 0 & \text{otherwise} \end{cases}$$

Marginal probability that Player 1 wins the game

We will use the property of the sign function resulting in $p(y = 1) = p(t > 0)$. Therefore, we need to find the marginal probability $p(t > 0)$.

$$p(t > 0) = \int_0^\infty \mathcal{N}(t; \mu_{s_1} - \mu_{s_2}, \sigma_{s_1}^2 + \sigma_{s_2}^2 + \sigma_t) = \Phi\left(\frac{(\mu_{s_1} - \mu_{s_2})}{\sqrt{\sigma_{s_1}^2 + \sigma_{s_2}^2 + \sigma_t^2}}\right)$$

where $\Phi(\cdot)$ is the standard normal distribution.

Q.4

Our objective is to draw samples from the target distribution $p(s_1, s_2|y)$. However, it is a difficult task to sample from it directly since we can not represent this distribution explicitly. Thus, we use Gibbs sampling with the following proposal distributions:

$$\begin{aligned} p(s_1, s_2|t) &= \mathcal{N}\left(t_{s_1,s_2}; \mu_{s_1,s_2|t}, \Sigma_{s_1,s_2|t}\right) \\ p(t|y, s_1, s_2) &\propto \begin{cases} \mathcal{N}(t; \mu_t, \sigma_t^2) & \text{if } a < t < \infty \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Our hyperparameters for our proposal distributions are the following, $\mu_{s_1} = 0$, $\sigma_{s_1}^2 = 1$, $\mu_{s_2} = 0$ and $\sigma_{s_2}^2 = 1$, further, let $\sigma_t^2 = \sigma_{s_1}^2 + \sigma_{s_2}^2 = 2$.

To figure out the burn-in period, we plot the samples. There is no significant burn-in period for this example and to accommodate outlier priors, we decided for 10 samples to be a reasonable burn-in period.

Further, we try out different sample sizes. You can see a clear trade of in accuracy and time. Detailed plots and corresponding execution times are available in the appendix in Figure 12. Gibbs Sampling becomes computationally heavy rather quickly. For instance, when working with approximately 10,000 samples, the process already consumes over 4 seconds. This consider this time is crucial since we will need to resample regularly from the Gibbs sampler in subsequent steps. Consequently, we opted for a sample size of 1,000. Although this choice sacrifices some accuracy compared to a size of

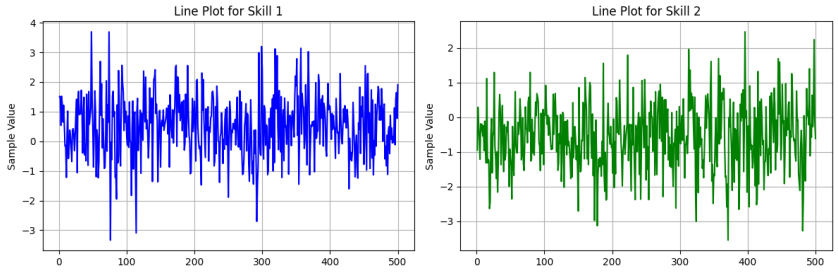

Figure 2: Gibbs Sampling over 500 samples with 0 burn in

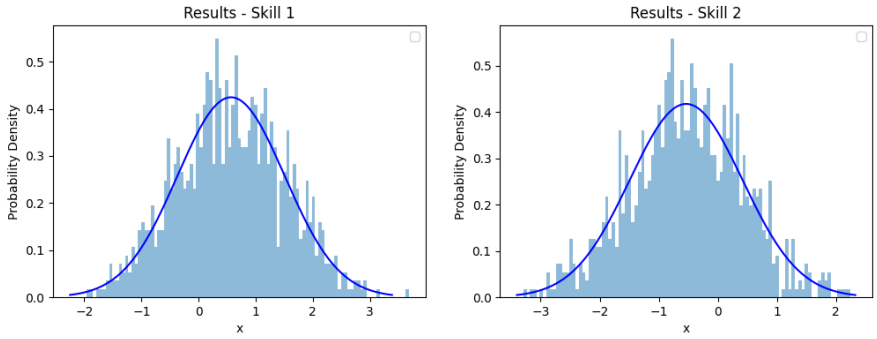

Figure 3: Gibbs Sampling 1000 samples, ≈ 0.51 sec

5,000, the substantial reduction in computation time makes it a pragmatic decision for our purposes, especially as the approximation of the mean is already quite close. We initialized the distributions for s_1 and s_2 with 0 mean and 1 standard deviation. After our Gibbs sampling, we obtain a mean from ≈ 0.54 and standard deviation from ≈ 0.97 for s_1 and a mean from ≈ -0.55 and a standard deviation from ≈ 0.92.

Q.5

Using assumed density filtering with Gibbs sampling produces skill estimates for each team in Serie A (season 2018-2019), read C in the appendix for the overview represented in a table. For these estimates, 1000 samples were produced with a burn-in of 10 for each game. The variance in the result is small even with 1000 samples, with slightly greater values for teams with lowest skill estimates.

With a shuffled dataset, the results have a minimal difference. The greatest and worst teams have a similar ranking as expected. The difference is in the middle positioned teams, since their rankings are similar, and they are inconsistent. Draws weren't consider either, which will impact teams with a lot of draws, ending up in less accurate skill estimates. There is a slight difference since assumed density filtering uses sequential sampling, which will affect how the posterior for each game was calculated.

Q.6

Bayesian linear regression (BLR) was chosen as a prediction function. Using the skill estimates from Q.5, we can create an input vector x as such:

$$x = \begin{bmatrix} game_1(\mu_{s_1} - \mu_{s_2}) \\ game_2(\mu_{s_1} - \mu_{s_2}) \\ \vdots \\ game_n(\mu_{s_1} - \mu_{s_2}) \end{bmatrix} \tag{3}$$

where $game_{n+1}$ is the game we're supposed to predict. $game_n(\mu_{s_1} - \mu_{s_2})$ represents the difference in mean of the skills between the two teams for the game . The output vector y, can be created as

such:

$$y = \begin{bmatrix} game_1(if(score_1 - score_2 > 0) : 1 \; else: \; -1) \\ game_2(if(score_1 - score_2 > 0) : 1 \; else: \; -1) \\ \vdots \\ game_n(if(score_1 - score_2 > 0) : 1 \; else: \; -1) \end{bmatrix} \tag{4}$$

where $game_n(if(score_1 - score_2 > 0) : 1 \; else: \; -1)$, assigns the value 1 to y[n] if team 1 won, otherwise y[n] will be -1.
The mean vector and covariance matrix will then be updated after every match, which also updates the weight vector accordingly. With one weight vector and sample seed 100, the prediction rate was ≈ 0.64, which is better than guessing. More information can be found in our code.

Q.7

Following all the information given in the project description, we obtain the following factor graph.

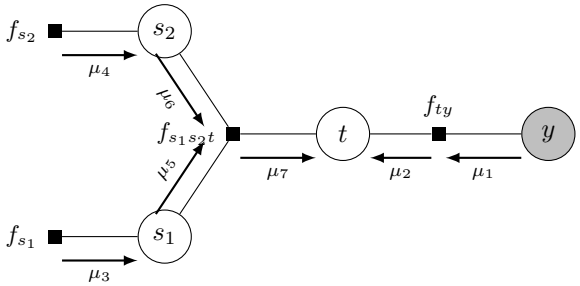

Figure 4: Factor Graph with messages for $p(t|y)$.

The factors are as follows:

$$\begin{aligned} f_{s_1}(s_1) &= \mathcal{N}(s_1, \mu_1, \sigma_1^2) \\ f_{s_2}(s_2) &= \mathcal{N}(s_2, \mu_2, \sigma_2^2) \\ f_{s_1 s_2 t}(t) &= \mathcal{N}(t, s_1 - s_2, \sigma_t^2) \\ f_{ty}(t, y) &= \delta(y = \text{sign}(t)) \end{aligned}$$

The messages are as follows:

$$\begin{aligned} \mu_1(y) &= \delta(y = 1) \\ \mu_2(y) &= \delta(y = \text{sign}(t)) \\ \mu_3(s_1) &= \mathcal{N}(s_1, \mu_1, \sigma_1^2) \\ \mu_4(s_2) &= \mathcal{N}(s_2, \mu_2, \sigma_2^2) \\ \mu_5(s_1) &= \mu_3(s_1) \\ \mu_6(s_2) &= \mu_4(s_2) \\ \mu_7(t) &= \int\int f_5(s_1, s_2, t)\mu_5(x)\mu_6(x) ds_1 ds_2 \\ &= \int\int \mathcal{N}(s_1, \mu_1, \sigma_1^2)\mathcal{N}(s_2, \mu_2, \sigma_2^2)\mathcal{N}(t, s_1 - s_2, \sigma_t^2) ds_1 ds_2 \\ &= \int\int p(t|s_1, s_2)p(s_1, s_2) ds_1 ds_2 \\ &= \mathcal{N}(t, \mu_{s_1} - \mu_{s_2}, \Sigma_t) \end{aligned}$$

Finally to obtain $p(t|y)$, we get the following truncated normal, as already in question 3:

$$\begin{aligned} p(t|y) &= \mu_2 \times \mu_7 \\ &= \mathcal{N}(t, \mu_{s_1} - \mu_{s_2}, \Sigma_t)\delta(y = sign(t)) \\ &= \begin{cases} \mathcal{N}(t; \mu_t, \Sigma_t) & \text{if } a < t < \infty \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} \mathcal{N}(t; \mu_{s_1} - \mu_{s_2}, \sigma_t^2 + \sigma_1^2 + \sigma_2^2) & \text{if } a < t < \infty \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Q.8

Now, we will use message passing to get the $p(s_1|y)$ and $p(s_2|y)$. For this, we get three additional messages into our factor graph.

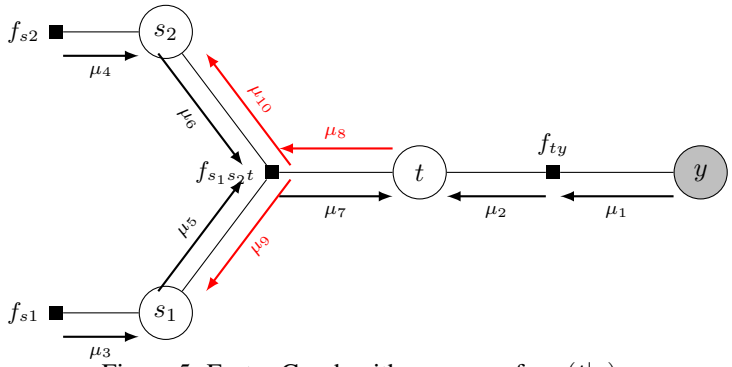

Figure 5: Factor Graph with messages for $p(t|y)$.

Now we will do the message passing algorithm:

$$\mu_{s_1}(y) = \delta(y = 1)$$

$$\mu_{s_2}(t) = \sum_y f_{ty}(t, y)\mu_{s_1}(y) = \sum y\delta(y = sign(t))\delta(y = 1) = \delta(t > 0).$$

From question 7, we know that $p(t|y)$ is a trunacted gaussian. We want to approximate this truncated gaussian with a gaussian message to facilitate our integrals in our message passages. Hence, we can define $q(t)$ as follows:

$$\begin{gathered} q(t) \propto p(t|y) \\ q(t) = \mathcal{N}(t; m_t, \sigma_t^2) \\ \textit{where} \\ m_t = \mathbb{E}_{p(t|y)}[t] \textit{ and } \sigma_t^2 = Var_{p(t|y)}[t] \end{gathered}$$

Therefore, we can define μ_8 as follows:

$$\begin{aligned} \mu_8 &= \frac{q(t)}{\mu_7} \\ &= \frac{\mathcal{N}(t; m_t, \sigma_t^2)}{\mathcal{N}(t, \mu_{s_1} - \mu_{s_2}, \Sigma_t)} \\ &\propto \mathcal{N}\left(t; \frac{m_t\Sigma_t - (\mu_{s_1} - \mu_{s_2})\sigma_t^2}{\Sigma_t - \sigma_t^2}, \frac{\sigma_t^2\Sigma_t}{\sigma_t - \sigma_t^2}\right) \\ &= \mathcal{N}(t; \hat{\mu}, \hat{\sigma}^2) \end{aligned}$$

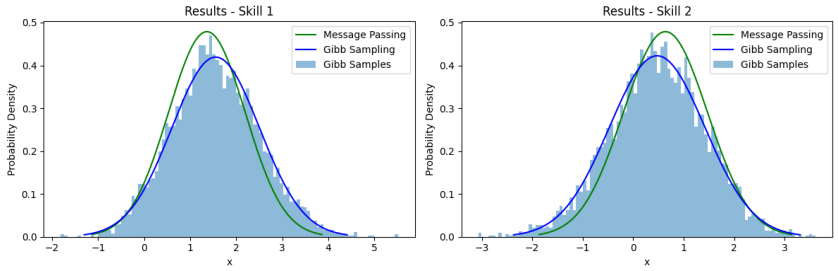

Figure 6: Comparison Gibbs and Message Passing

Using Property 2 and Corollary 2, we get:

$$\begin{aligned}
&\mu_9(s_1) \\
&\quad = \int\int \mathcal{N}(t; s_1 - s_2, \sigma_t^2)\mu_8(t)\mu_5(s_2)dtds_2 \\
&\quad \propto \int\int \mathcal{N}(s_1, t + s_2, \sigma_t^2)\mu_8(t)\mu_5(s_2)dtds_2 \\
&\quad = \mathcal{N}(s_1, \hat{\mu} + \mu_{s_2}, \sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_2}^2)
\end{aligned}$$

$$\begin{aligned}
&\mu_{10}(s_2) \\
&\quad = \int\int \mathcal{N}(t; s_1 - s_2, \sigma_t^2)\mu_8(t)\mu_5(s_1)dtds_1 \\
&\quad \propto \int\int \mathcal{N}(s_2, s_1 - t, \sigma_t^2)\mu_8(t)\mu_5(s_1)dtds_1 \\
&\quad = \mathcal{N}(s_2, \mu_{s_1} - \hat{\mu}, \sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_1}^2)
\end{aligned}$$

Finally, the probabilities for the skills we use Gaussian multiplication at both their nodes:

$$\begin{aligned}
p(s_1|y) &= \mathcal{N}(s_1; \mu_{s_1}, \sigma_{s_1}^2)\mathcal{N}(s_1, \hat{\mu} + \mu_{s_2}, \sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_2}^2) \\
&= \mathcal{N}\left(s_1; \frac{\mu_{s_1}(\sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_2}^2) + (\mu_t + \mu_{s_2})\sigma_{s_1}^2}{\sigma_{s_1}^2 + \sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_2}^2}, \frac{\sigma_{s_1}^2(\sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_2}^2)}{\sigma_{s_1}^2 + \sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_2}^2}\right) \\
p(s_2|y) &= \mathcal{N}(s_2, \mu_{s_2}, \sigma_{s_2}^2)\mathcal{N}(s_2, \mu_{s_1} - \mu_t, \sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_1}^2) \\
&= \mathcal{N}\left(\frac{\mu_{s_2}(\sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_1}^2) + (\mu_{s_1} - \mu_t)\sigma_{s_2}^2}{\sigma_{s_2}^2 + \sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_1}^2}, \frac{\sigma_{s_1}^2(\sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_1}^2)}{\sigma_{s_2}^2 + \sigma_t^2 + \hat{\sigma}^2 + \sigma_{s_1}^2}\right)
\end{aligned}$$

Finally, we compare our results from message passing to our results form Gibbs sampling in figure 6. There is a small difference between both distributions, but overall they match very well.

Q.9

We decided work with data consisting of games from League of Legends (Esports) from the years 2015-2018 [2]. League of Legends (LoL) is a popular multiplayer online battle arena video game developed and published by Riot Games. It is a team-based strategy game that has gained a massive player base since its release. Two teams of five compete against each other, with the goal to destroy the other team's base. Each player is allocated a different role within the team, depending on which lane they are playing on, the roles are called: top, jungle, mid lane, bot lane and support, but team coordination is highly important [3]. There is a highly competetive Esport scene for LoL, including multiple leagues, that are included in our data set.

We chose to work with data from NA LCS, the North American League of Legends series and focused on the year 2015, as for this year we had data on spring and summer season (185), promotion (62), playoffs (46) and regional games (14) making up a total of 307 games we could use for the analysis.

The LoL data is a perfect match for our model, as there is no possibilities for draws. Furthermore, in our data set the two teams (red or blue), where given a 1 for a win and and a 0 for a loss. Hence, we could run our created model, step by step on this data. Finally, using our prediction model from Question 5 and 6. The results from using our assumed density filtering with Gibbs sampling can be found in the appendix. As our training data included promotion events, with teams outside of the normal league, the outcome includes more teams, than the actual league, however the general tendencies have been captured well. To verify the results more, we could have exluded the promotion games, in order to keep the same teams as in the actual league. Furthermore, we used Bayesian Linear Regression (BLR) with Gibbs Sampling to predict the games using all NA LCS games from 2015, resulting in an accuracy from ≈ 0.56.

Q.10

In order to further improve the model, several advancements were done.

For the Gibbs sampling BLR we attempted to hyperparameter tune our different posterior distributions. Our number of weight vector was increased from 1 in Q.6, to multiple higher numbers. The different predictions were then returned as mean of the different predictions instead, which made the model slightly more accurate. With reaching an accuracy from ≈ 0.61 for setting our weight vector to 5. A further increase to 10 did result in the exact same outcome.

In the game of LoL there is a small advantage of starting at the blue side, hence we wanted to incorporate this prior knowledge into our model. In our particular data which we used there was approximately 55% win rate for whichever team was on the blue side.

For the Gibbs Sampling BLR we tried to incorporate this knowledge by changing the prior mean vector to greater values than 0, indicating that the blue side team has a higher probability of winning. This didn't affect the results, most probably because the difference is too small. Furthermore, the Season is divided into Spring and Summer Season, giving us a good opportunity to train the data on the Spring Season and then use this as prior knowledge for the Summer Season. This resulted in a ≈ 0.6 prediction rate.

We also tried to incorporate the prior knowledge in the belief propagation BLR (message passing). It was extended by adding a small bias to the mean of whichever team was on blue side. We added a constant of 0.03 to the mean. It improved our prediction rate slightly, from 56.9% to 61.3%.

Discussion and conclusions

Having run both message passing and the prediction approach based on sampling and simulating on the Football and the League of Legends data sets, we experienced that the time to run the message passing is much faster. However, setting up the message passing can be very tricky and time consuming. Therefore, we came to the conclusion, that the message passing is superior with a lot of data, however, if a model is getting very complex, the simulation approach might be favourable, as setting up the message passing is getting too complex.

In addition, there are many ways to extend the model, that all immediately make the model more complex. For the League of Legends data, we also discussed to take the Game Length or the Skill Sets of the individual players into account, e.g. changing the prior belief of the game result, if they have a champion on their position. Many of those ideas, massively change the model structure and would mean to revise many things from beginning on. Furthermore, we opted for a rather simple way to add our prior knowledge to the model. There would also be other ways to do this, as e.g. changing the mean of t, this however would effect the message passing a lot more and would require us to revise this part. Furthermore, we could extend on the idea of using the results from previous seasons as priors for our models, however in LoL there is a much higher changing rate of teams compared to other sports, given some limitation to this.

References

[1] "TrueSkill™ Ranking System," https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/, accessed: 2023-10-03.

[2] "League of Legends - Data ," https://www.kaggle.com/datasets/chuckephron/leagueoflegends/, accessed: 2023-10-04.

[3] "League of Legends," https://www.leagueoflegends.com, accessed: 2023-10-04.

A Theorems, Definitions and Corollaries

Definition: Product Rule

The **conditional probability** is defined as

$$p(x|y) = \frac{p(x,y)}{p(y)} \quad \text{where } p(y) \neq 0 \tag{5}$$

$$\Rightarrow p(x,y) = p(x|y)p(y) \tag{6}$$

also called the **product rule**.

Theorem: Bayes' Theorem

The joint distribution factorizes into a **likelihood** and a **prior**.

$$p(D,\theta) = p(D|\theta)p(\theta)$$

We find $p(\theta|\mathcal{D})$ using **Bayes' Theorem**

$$p(\theta|D) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \tag{7}$$

Corollary: Affine transformation – Conditioning

Assume that x_a, as well as x_b conditioned on x_a, are Gaussian distributed according to

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

Then the conditional distribution

$$p(x_a|x_b) = \frac{p(x_a,x_b)}{p(x_b)} = \frac{p(x_b|x_a)p(x_a)}{p(x_b)}$$

is given by

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

with

$$\mu_{a|b} = \Sigma_{a|b}(\Sigma_a^{-1}\mu_a + A^T\Sigma_{b|a}^{-1}(x_b - b)) \tag{8}$$

$$\Sigma_{a|b} = (\Sigma_a^{-1} + A^T\Sigma_{b|a}^{-1}A)^{-1}. \tag{9}$$

B Plots and Times for Gibbs sampling

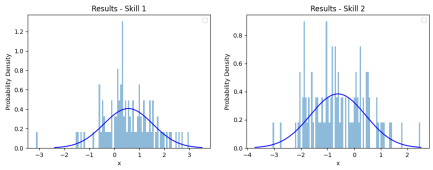

Figure 7: 100 samples, ≈ 0.10 sec

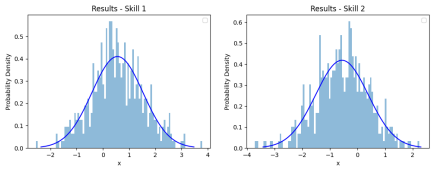

Figure 8: 500 samples, ≈ 0.27 sec

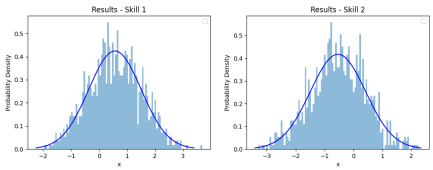

Figure 9: 1000 samples, ≈ 0.51 sec

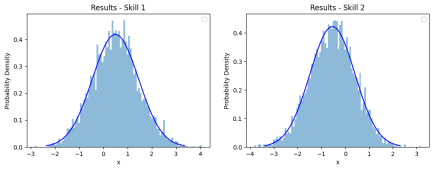

Figure 10: 5000 samples, ≈ 2.33 sec

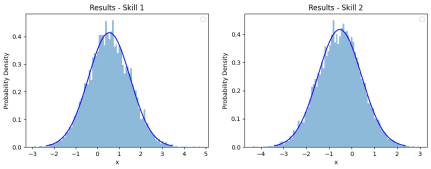

Figure 11: 10000 samples, ≈ 4.71 sec

Figure 12: Gibbs Sampling Results

C Tables for Q.5

Team	μ	σ
Juventus	0.506	0.092
Napoli	0.409	0.078
Atalanta	0.352	0.097
Inter	0.339	0.095
Milan	0.315	0.095
Roma	0.301	0.098
Torino	0.159	0.098
Sampdoria	0.078	0.077
Spal	0.071	0.08
Genoa	0.067	0.111
Bologna	0.059	0.078
Lazio	-0.002	0.082
Empoli	-0.044	0.077
Udinese	-0.105	0.077
Cagliari	-0.157	0.079
Parma	-0.2	0.096
Fiorentina	-0.243	0.105
Sassuolo	-0.277	0.129
Frosinone	-0.46	0.094
Chievo	-0.978	0.131

Table 1: Skill estimates for each team in serie A(2018-2019) ordered by μ

Team	μ	σ
Juventus	0.472	0.078
Napoli	0.298	0.06
Inter	0.219	0.074
Roma	0.212	0.077
Milan	0.205	0.086
Atalanta	0.137	0.068
Torino	0.13	0.093
Sampdoria	0.129	0.052
Lazio	0.07	0.061
Empoli	-0.048	0.054
Genoa	-0.065	0.078
Bologna	-0.113	0.087
Fiorentina	-0.14	0.104
Sassuolo	-0.142	0.093
Parma	-0.146	0.081
Cagliari	-0.157	0.093
Spal	-0.175	0.086
Udinese	-0.25	0.07
Frosinone	-0.574	0.089
Chievo	-1.267	0.091

Table 2: Skill estimates for each team in serie A(2018-2019) ordered by μ, with a shuffled dataset

D Results for League of Legends Dataset

Team	μ	σ	Actual Results (Summer Standings)
CLG	0.64	0.04	Team Liquid (TL)
CA	0.6	0.12	Counter Logic Gaming (CLG)
TSM	0.57	0.04	Team Impulse (TIP)
EG	0.43	0.64	Gravity (GV)
TL	0.43	0.03	TSM
TIP	0.33	0.03	Dignitas (DIG)
C9	0.29	0.03	Cloud9 (C9)
GV	0.22	0.04	Team 8 (T8)
FSN	0.02	0.08	Enemy (NME)
DIG	-0.09	0.04	Team Dragon Knights (TDG)
F5	-0.12	0.16	
MKZ	-0.25	0.5	
COW	-0.25	0.35	
CST	-0.26	0.07	
T8	-0.27	0.05	
COL	-0.4	0.27	
CLB	-0.52	0.69	
TDK	-0.54	0.1	
WFX	-0.56	0.09	
NME	-0.68	0.1	
ZEN	-0.69	0.67	
FIS	-0.69	0.67	
PRO	-0.7	0.36	
NTR	-1.16	0.5	
CG	-1.5	0.48	