CUED - Engineering Tripos Part IIB 2023-2024 **Module Coursework**

Module	4F13	Title of report	Coursework 2 – Probabilistic Rankings
Date submitted: 21/11/2023		Assessment for this module is ☑ 100% / ☐ 25% coursework of which this assignment forms _33.33__ %	

UNDERGRADUATE and POST GRADUATE STUDENTS		
Candidate number:	5553C	☑ Undergraduate ☐ Post graduate

Feedback to the student ☐ **See also comments in the text**		Very good	**Good**	Needs improvmt
CONTENT	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
PR	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			

1 Exercise A

```
for p in range(M):
     m[p] = sum(t[:, 0] *
            (np.where(G[:, 0] == p -
             np.where(G[:, 1] == p)))
iS = np.zeros((M, M))
for g in range(N):
    win_player = G[g, 0]
    loss_player = G[g, 1]
    iS[win_player, win_player] += 1
    iS[loss_player, loss_player] += 1
    iS[win_player, loss_player] -= 1
    iS[loss_player, win_player ]-= 1
```

Figure 1: Completed code for sampling from the conditional distributions in Gibbs Sampling. The first loop calculates the mean and the second calculates the inverse of the covariance matrix.

Figure 1 is the implementations of the mean and precision equations for the conditional posterior: $\tilde{u}_i = \sum_{g=1}^{G} t_g(\delta(i - I_g) + \delta(i - J_g))$, $[\tilde{\sum}^{-1}]_{ii} = \sum_{g=1}^{G} \delta(i - I_g) + \delta(i - J_g)$ and $[\tilde{\sum}^{-1}]_{i \neq j} = -\sum_{g=1}^{G} \delta(i - I_g)\delta(j - J_g) + \delta(i - J_g)\delta(j - I_g)$.

Using Gibbs sampling we are able to obtain skill samples for each player, as shown in Figure 2. As seen in any Markov Chain Monte Carlo (MCMC) process, we have a number of 'burn-in' iterations before the samples are drawn from the target distribution. This can be seen clearly for Andy Murray in Figure 2; the sampled skill starts low then increases to 1.3/1.4, where it then fluctuates about. The burn-in number of iterations for this case looks to be 20, therefore we shall use double that for safety, We will also add 10, to ensure that the future samples are independent of the burn-in data (see Figure 3).

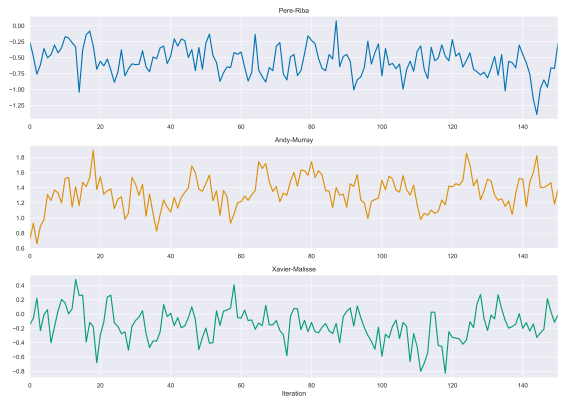

Figure 2: Mean skill for 150 iterations of Gibbs Sampling for 3 sample tennis players.

In order for more accurate sampling it is useful to only use samples which are independent of each other. We can measure the dependence between samples by computing the auto-correlation of the samples with a lagged version of the same samples. The results of this for all players are shown in Figure 3. Here we can see that zero is approximately reached at a time lag of 10 for all players, hence in the future it will be best to thin the data by using every tenth sample for an independent set of samples. This of course significantly increases the number of iterations required to achieve the same number of samples.

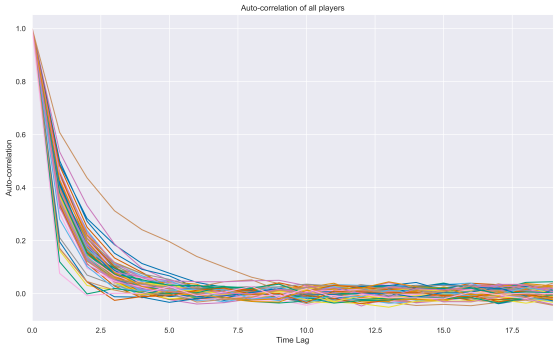

Figure 3: Auto-correlation of all players at different time delays.

from Figure 4 we see that the average mean skill for our samples significantly drops at first as we increase the number of iterations, and then begins to level out. We see that from $15,000$ iterations this has little change, hence we will use this number of iterations moving forward. This is a large amount and requires more computational time, but it is needed as we have $1,500$ samples after thinning.

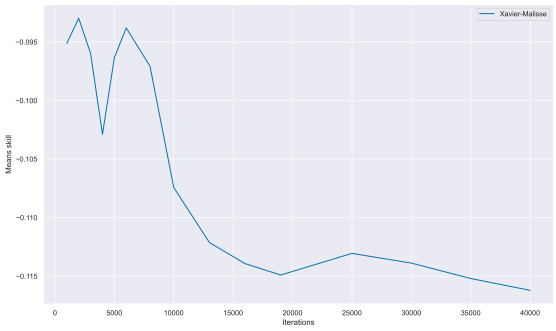

Figure 4: Visualisation of the effect that the number of iterations has on the mean skill in Gibbs Sampling. Burn-in of 50 is used with every tenth sample used.

Figure 5 is an updated version of Figure 2, but with a burn-in of 50 iterations and thinning

implemented.

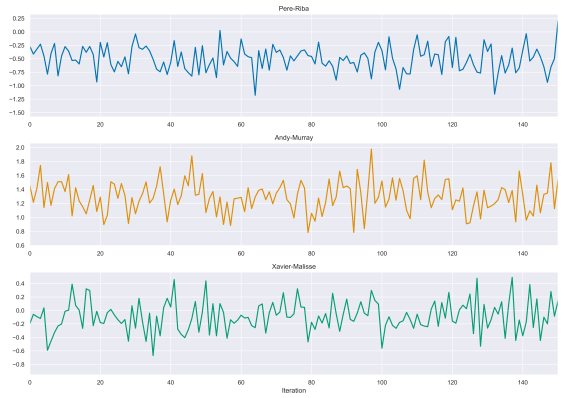

Figure 5: Mean skill for 1,500 iterations of Gibbs Sampling with a burn-in of 50 iterations and thinning (Every tenth sample selected). 3 sample tennis players are shown.

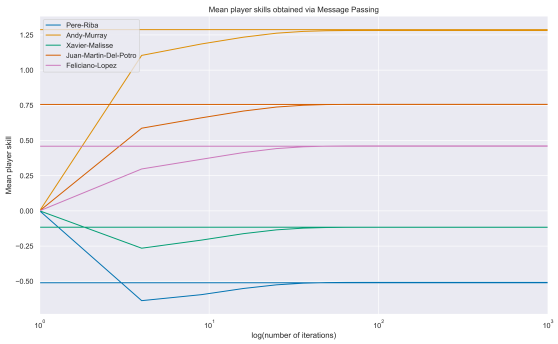

Figure 6: Mean player skill, obtained from Message Passing, against the number of iterations used. The constant lines denote the average value obtained from Gibbs sampling for comparison.

2 Exercise B

```
iterations = (np.indices((num_iter_sequence) + 1)**2
mean_player_skills
precision_player_skills = np.zeros((107, num_iter_sequence))
np.zeros((107, num_iter_sequence))
for index, num_iterations in enumerate(iterations[0]):
    mean_player_skills[:, index] = eprank(G, M, num_iterations)[0]
     precision_player_skills[:, index] = eprank(G, M, num_iterations)[0]
```

Figure 7: Code for obtaining the number of iterations till convergence for Message Passing.

The joint posterior of the rankings, $P(\mathbf{w}|\mathbf{y})$, is intractable due to the presence of truncated Gaussians. Therefore we explore two methods, Gibbs Sampling and Message Passing, to get around this.

Message Passing approximates the truncated Gaussians as Gaussians by matching the first and second moments. Each iteration of this method updates the parameters of these approximations to a greater degree of accuracy. This method is determined to have converged to an optimum when the difference between adjacent estimates has dropped below a small value specified by the user. In our case, we use Figure 6 to determine this; we observe that an acceptable convergence is achieved before 100 iterations. Moving forward we will use 64 iterations for Message Passing.

Gibbs Sampling is an MCMC process that generates samples from the target joint posterior by first sampling the performance differences from their conditional posteriors, followed by jointly sampling the skills from the conditional posterior. The aim of this process is to achieve a steady state that matches the target distribution. Gibbs sampling is converged when the burn-in phase, as mentioned in Section 1, has passed and a steady state has been achieved.

3 Exercise C

Message passing yields the mean skill and precision of each player. This is then used to calculate the probability that player 1 has a higher skill and the probability that player 1 wins, for all possible match-ups amongst the 107 players. The results for the skill and win probabilities for the top 4 players are shown in Table 1 and Table 2.

The results in both of the tables differ due to the inclusion of the performance noise, modelled as $\mathcal{N}(0, 1)$, when calculating the mean skill difference.

$$P(w_1 > w_2) = \Phi\left(\frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}\right) \quad (1)$$

$$P(y = 1) = \Phi\left(\frac{\mu_1 - \mu_2}{\sqrt{1 + \sigma_1^2 + \sigma_2^2}}\right) \quad (2)$$

The probabilities now drift considerably closer to 0.5 as we now model the fact that one player may play worse on a day and therefore have a lower effective skill, whilst the other player may perform better and have a higher effective skill. Without this performance noise, we estimate the players will always compete at the exact same skill level. The drift to 0.5 is intuitive as the increased size of the denominator drives the mean skill difference closer to 0; we know that $\Phi(0) = 0.5$.

```
for row in range(number_players):
    for column in range(number_players):
            variance_sum = precisions[row]**-1 + precisions[column]**-1
            skill_prob[row, column] = scipy.stats.norm.cdf(0, mean_skills[row]
                                      -mean_skills[column], variance_sum**0.5)
            win_prob[row, column] = scipy.stats.norm.cdf(0, mean_skills[row]
                                    -mean_skills[column],(1+variance_sum)**0.5)
```

Figure 8: Code for calculating the probability that player 1 has higher skill and the probability that player 1 wins (implementation of Equations 1 and 2). Results for the top 4 ranked players are shown in Table 1 and Table 2.

Skill	Player 1			
Player 2	Novak Djokovic	Roger Federer	Rafael Nadal	Andy Murray
Novak Djokovic	-	0.091115	0.060178	0.014678
Roger Federer	0.908885	-	0.427170	0.189165
Rafael Nadal	0.939822	0.572830	-	0.233481
Andy Murray	0.985322	0.810835	0.766519	-

Table 1: Probability that Player 1 has higher skill than Player 2, for the top 4 ranked tennis players, with Message Passing

Win	Player 1			
Player 2	Novak Djokovic	Roger Federer	Rafael Nadal	Andy Murray
Novak Djokovic	-	0.361973	0.344633	0.280174
Roger Federer	0.638027	-	0.481648	0.409121
Rafael Nadal	0.655367	0.518352	-	0.426890
Andy Murray	0.719826	0.590879	0.573110	-

Table 2: Probability that Player 1 will win against Player 2, for the top 4 ranked tennis players, with Message Passing

4 Exercise D

```
skill_prob_marginal = scipy.stats.norm.cdf(0, mean_diff, variance_sum**0.5)
joint_gaussian = np.random.multivariate_normal(mean, cov, size=100000)
skill_prob_joint = np.mean(joint_gaussian[:, 0] > joint_gaussian[:, 1])
skill_prob_direct = np.mean(Djok > Nadal)
```

Figure 9: Code for the 3 separate methods (Marginal skills, Joint skills and Direct samples) used after Gibbs Sampling to obtain the probability that Djokovic has a higher skill than Nadal.

Once we have used Gibbs Sampling, there are 3 ways to use the samples to generate the probability that player 1 has higher skill for all possible match-ups amongst the 107 players (as was done using Message Passing in Section 3): The first approximates the marginal skills as Gaussians, the second approximates the joint skills using a multivariate Gaussian and the last directly compares the samples for the players. The results for each method can be seen in Table 3.

	Method		
	Marginal	Joint	Direct
Probability	0.91458	0.94458	0.95385

Table 3: Results for the 3 separate methods, described in Figure 9, used to obtain the probability that Djokovic has a higher skill than Nadal.

Out of the first two methods, the second can better represent the data due to the inclusion of the covariance when approximating the joint skills; when approximating each player separately this is excluded. It is not present in the marginal skills. Although both of these methods are limited as they approximate a non-Gaussian distribution to a Gaussian distribution. Therefore the third method is determined to be the best, as it directly compares the samples, hence loss of accuracy does not occur via Gaussian approximations. The caveat of this method is that it requires a greater number of samples for reliability.

By comparing direct samples we obtain the skill probabilities for the players, as shown in Table 4 for the top 4 players. From Table 1 we see that the probabilities calculated using Message Passing are similar. The comparable performance is extremely impressive as Message Passing took $753ms$ to run whereas the Gibbs Sampling method took 15 minutes to run, which is a huge increase in computational time; even with thinning removed and the same number of samples used, it takes 2 minutes to run.

Skill	Player 1			
Player 2	Novak Djokovic	Roger Federer	Rafael Nadal	Andy Murray
Novak Djokovic	-	0.094983	0.046154	0.012040
Roger Federer	0.905017	-	0.437458	0.186622
Rafael Nadal	0.953846	0.562542	-	0.214047
Andy Murray	0.987960	0.813378	0.785953	-

Table 4: Probability that Player 1 has higher skill than Player 2, for the top 4 ranked tennis players, with direct Gibbs Sampling.

5 Exercise E

```
for i in range(70):
    x.append(i)
    EP_correct.append(np.sum(EP_emp_diff_pos <= i))
    Gibbs_correct.append(np.sum(Gibbs_emp_diff_pos <= i))
```

Figure 10: Code for Generating the posit.

Comparing both Figures 12 and 13 to Figure 11, we observe both methods produce mean skill probabilities that are more gradual than the win-rates in the empirical rankings. The sharper profile in Figure 11 is possible as the absolute number of wins is used to calculate the rankings, this means the players that win a lot will have a large win-rate, whereas those with no wins will have a zero win-rate. This isn't present for MP and Gibbs Sampling as they are probabilistic models, so getting close to 0 or 1 is extremely unlikely.

From Figure 14 we can observe the amount that the positions in MP and Gibbs sampling vary from the empirical rankings; this shows us how the difference between the empirical rankings and the other ranking models is very large. The reason for the difference is simply due to the greater amount of information used when ranking; it's not just the number of wins which is important, but who they're against. For example, if MP and Gibbs observe player 1 beat player 2, who usually wins the vast majority of their games, they will increase the skill of player 1 by a greater amount than they would've if player 2 usually never won a game.

This clearly indicates that both Message Passing and Gibbs Sampling use more contextual

information on wins and therefore produce better rankings.

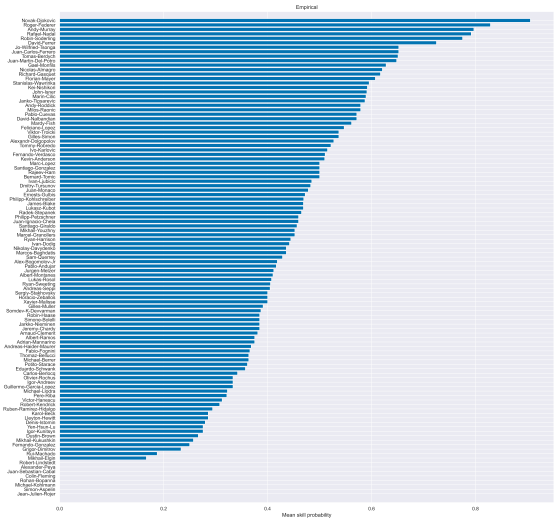

Figure 11: Player rankings ordered by mean empirical skill probability.

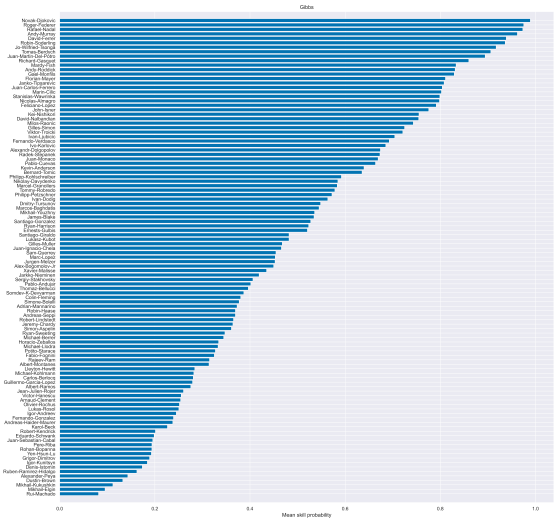

Figure 12: Player rankings ordered by mean Gibbs Sampling skill probability.

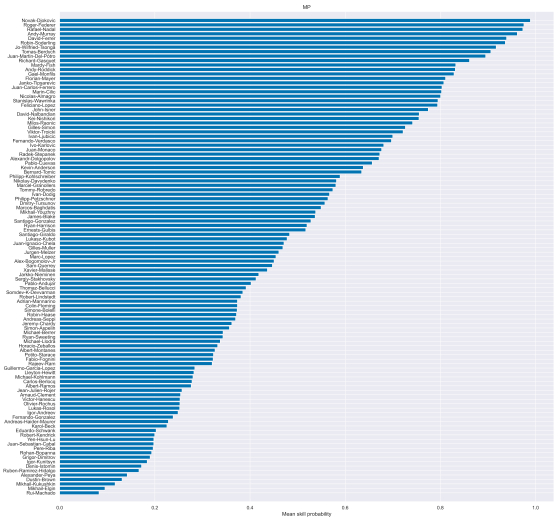

Figure 13: Player rankings ordered by mean Message Passing skill probability.

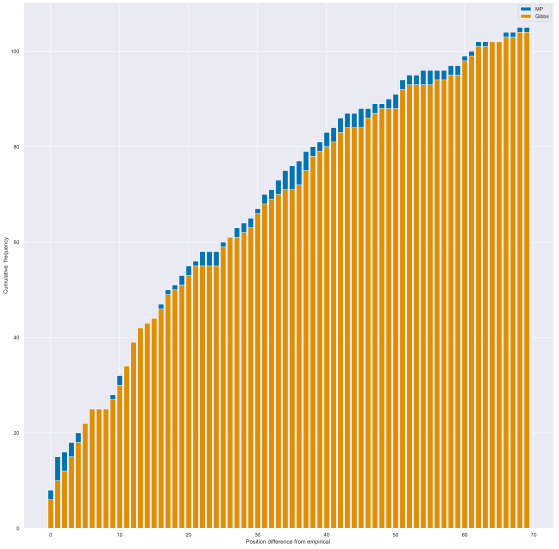

Figure 14: Cumulative number of players with varying position difference from empirical rankings for MP and Gibbs Sampling.