CUED - Engineering Tripos Part IIB 2021-2022 **Module Coursework**

Module	4F13	Title of report	Probabilistic Ranking

Date submitted: 11/11/2021	Assessment for this module is [✓ 100%] / ☐ 25% coursework of which this assignment forms 33 %

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:	5585G	Name:		College:	

Feedback to the student

☐ **See also comments in the text**

		Very good	**Good**	Needs improvmt
C O N T E N T	**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:			
P R E S E N T A T I O N	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Indicative grades are not provided for the FINAL piece of coursework in a module

Assessment (circle one or two grades)		A*	A	B	C	D
Indicative grade guideline		>75%	65-75%	55-65%	40-55%	<40%
Penalty for lateness:		*20% of maximum achievable marks per week or part week that the work is late.*				

Marker: Date:

1 Probabilistic Ranking

1.1 Question 1:

```
for p in range(M): # for each player
    # the == operator acts as a delta function for sifting through player identities
    m[p] = np.dot(t.T, (p==G[:, 0]).astype(int) - (p==G[:,1 ]).astype(int))


for j in range(M):
    for k in range(j+1):
    # use expressions in the lecture notes
    if j == k:
        iS[j, k] = np.sum((j==G[:,0]).astype(int)) + np.sum((j==G[:,1]).astype(int))
    else:
        iS[j, k] = -np.sum((j==G[:,0]).astype(int)*(k==G[:,1]).astype(int)) -np.sum((k==G
            [:,0]).astype(int)*(j==G[:,1]).astype(int))
        # exploit symmetry
        iS[k, j] = iS[j, k]
```

Figure 1: Python code for implementing the conditional distribution

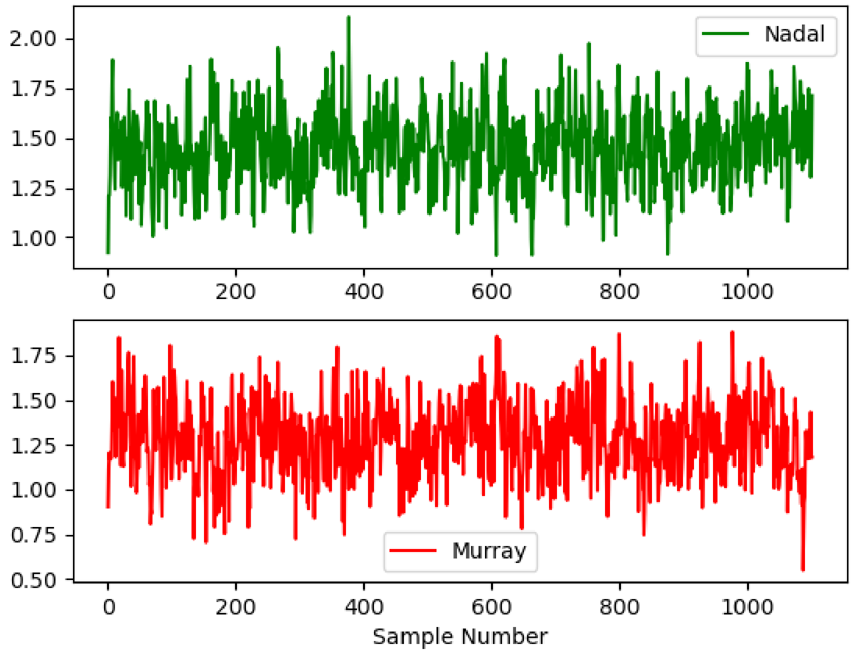


Figure 2: Gibbs samples

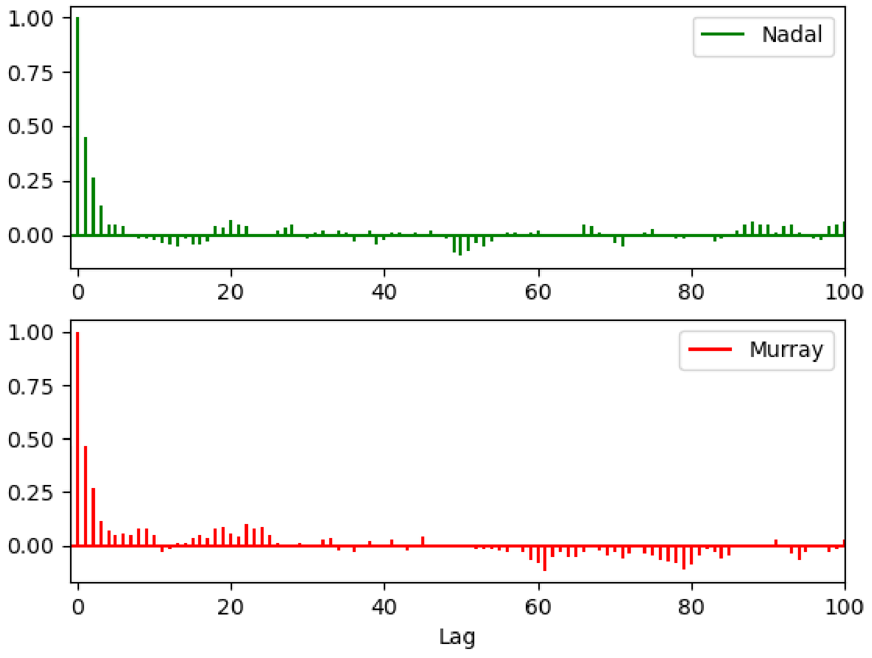


Figure 3: Autocorrelation

If we examine the initial skill samples more closely, the *burn in* time seems quite short (meaning the chain does not take a long time to traverse from its initial point to a region of the density with higher probability). A crude, under-optimistic estimate obtained by looking at skill samples for several players was 10 iterations for burn in.

Further, the autocorrelation plots show that there is significant correlation for short lag times, which wears off quite quickly. An estimate of the autocorrelation time is about 10 samples, again obtained by studying several plots and finding the first time where the autocorrelation drops to 0.

One way to alleviate these issues is to *thin* the samples - we pick out samples at regular intervals, say every 10 samples based on the estimated autocorrelation time and burn in time.

Then, provided that the long-term distribution of samples seems stationary, we can say that these are reliable samples from the posterior, and that the chain is exploring the whole posterior well.

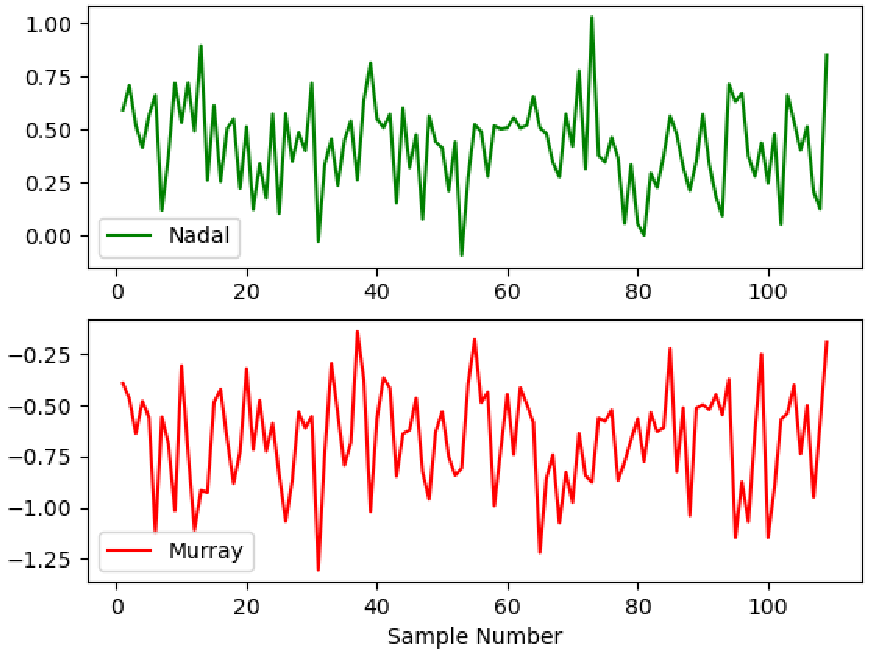


Figure 4: Thinned gibbs samples

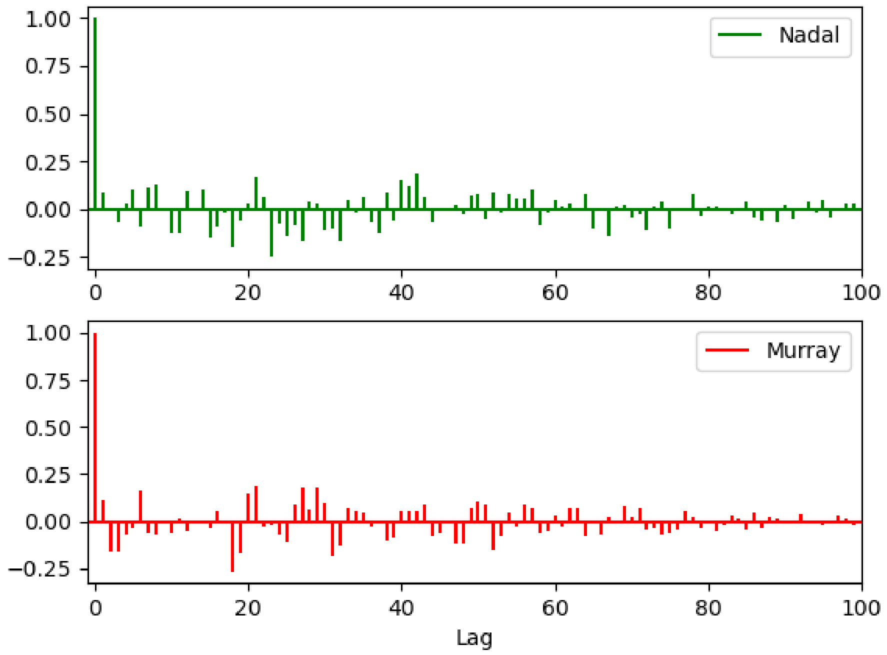


Figure 5: Autocorrelation for thinned gibbs samples

The new *thinned* samples are significantly less correlated.

1.2 Question 2

In the message passing EP algorithm our aim is to infer the parameters of the marginal skill distribution $q(w_i) \equiv (r_i, \lambda_i)$, and the marginal performance distribution $q(t_g) \equiv (\tilde{\mu}_g, \tilde{v}_g)$. Convergence is in the value of the parameters that define the distribution from which we can draw. In this case q is a Gaussian (although it is only an approximation for $q(t_g)$). In each iteration q is updated from q^τ to $q^{\tau+1}$, and so the algorithm has converged once:

$$|q^{\tau+1}(w_i) - q^\tau(w_i)| < \epsilon \quad \text{and} \quad |q^{\tau+1}(t_g) - q^\tau(t_g)| < \epsilon$$

This is very easily measured during running of the algorithm and a convergence parameter, ϵ, could be chosen for termination purposes.

For GS, convergence is achieved once the statistical properties of the sample collection remain fixed. Therefore GS converges to the distribution itself, from which we can calculate parameters. This is slightly harder to measure but could be achieved in similar fashion by calculating online the sample mean and sample variance then checking the conditions:

$$|\hat{\mu}_{(i+1)} - \hat{\mu}_{(i)}| < \epsilon \quad \text{and} \quad |\hat{\sigma}^2_{(i+1)} - \hat{\sigma}^2_{(i)}| < \epsilon$$

Though it would be wise to only start measuring this after a reasonable number of samples have been taken to avoid any kind of statistical anomaly. In practice it is better to just run the chain for a significant number of samples and then thin.

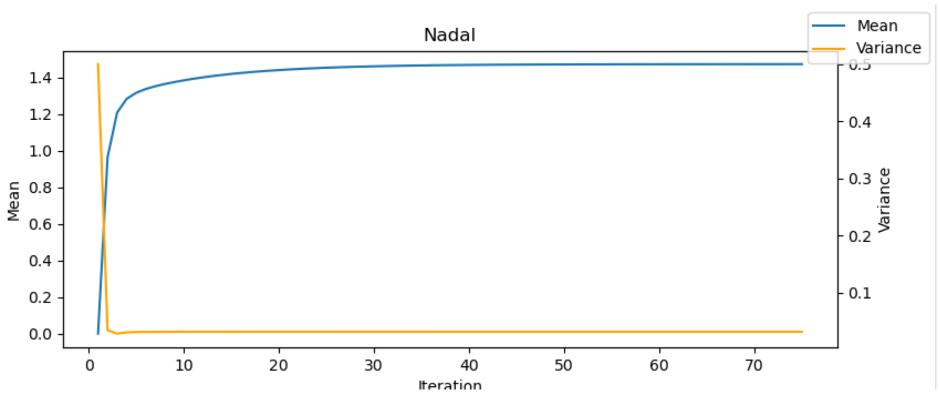


Figure 6: EP convergence for *Nadal*

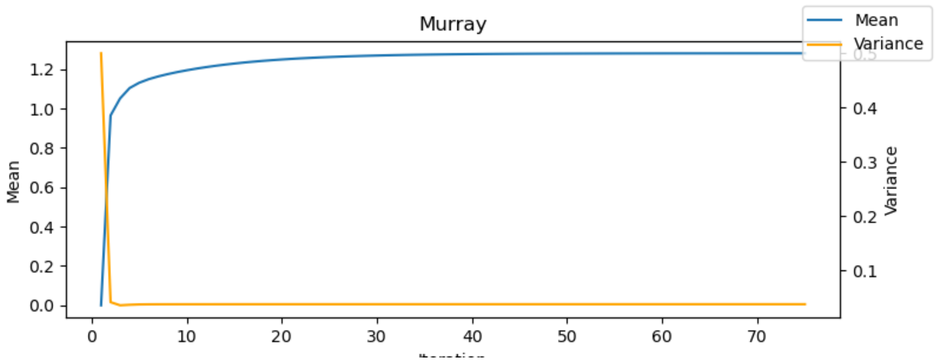


Figure 7: EP convergence for *Murray*

The above figures demonstrate the rapid convergence of the EP algorithm, showing that only 70 iterations or so are required. This is in strong contrast to GS where over 1100 iterations were used, with a significantly higher run-time.

1.3 Question 3

The probability of player one's skill being higher can be expressed as:

$$p(w_1 > w_2) = p(w_2 - w_1 < 0) = \Phi\big(\frac{\mu_2 - \mu_1}{\sqrt{\sigma_1^2 + \sigma_2^2}}\big)$$

Similarly, the probability of player one winning can be expressed as:

$$p(y = 1|w_1, w_2) = p(t > 0|w_1, w_2) = p(w_1 - w_2 - n > 0) \quad \text{where} \quad n \sim \mathcal{N}(0, 1)$$

Which is: $p(y = 1|w_1, w_2) = \Phi\big(\frac{\mu_2 - \mu_1}{\sqrt{1+\sigma_1^2+\sigma_2^2}}\big)$

These expressions were used to generate the following two tables:

	Player 2			
Player 1	Djokovic	Nadal	Federer	Murray
Djokovic		0.9398	0.9089	0.9853
Nadal	0.0602		0.4272	0.7665
Federer	0.0911	0.5728		0.8108
Murray	0.0147	0.2335	0.1892	

Table 1: Probability of player 1 having higher skill than player 2 according to message passing

	Loser			
Winner	Djokovic	Nadal	Federer	Murray
Djokovic		0.6554	0.6380	0.7198
Nadal	0.3446		0.4816	0.5731
Federer	0.3620	0.5184		0.5909
Murray	0.2802	0.4269	0.4091	

Table 2: Winning probabilities according to message passing

These two tables are strongly correlated - a higher probability of having a higher skills implies a higher chance of winning. However the smoothing effect of the noise is clear here. The noise brings the win probability closer to 0.5 for both players as it increases the chance of a large performance difference on the day. In the limit of infinite power noise, skill becomes irrelevant and each player has a 50% chance of winning.

Also, the ATP rankings are not completely in agreement with the TrueSkill model. For example, Federer is predicted on average to beat Nadal, despite Nadal being 1 rank higher than Federer in the ATP.

1.4 Question 4

We consider three approaches for using the gibbs samples to compare two players' skill. Firstly, we model each skill marginal as it's own Gaussian and calculate maximum likelihood estimates of the parameters.

Secondly, we model the two players' skills as a joint Gaussian (therefore we incorporate possible correlation between the skills). Finally we directly compare the skills values and simply measure the proportion of player 1's skills that are greater than player 2.

With Djokovic as player 1 and Nadal as player 2 we obtain the following probabilities:

	Independent	Joint	Direct
	0.9207	0.9204	0.9358

Table 3: Probabilities of $w_{Djok} > w_{Nadal}$

With the estimated distributions:

$$p(w_D) = \mathcal{N}(w_D; 1.8867, 0.0472), \quad p(w_N) = \mathcal{N}(w_N; 1.4803, 0.0359)$$

$$p(\begin{bmatrix} w_D \\ w_N \end{bmatrix}) = \mathcal{N}(\begin{bmatrix} w_D \\ w_N \end{bmatrix}; \begin{bmatrix} 1.8867 \\ 1.4803 \end{bmatrix}, \begin{bmatrix} 0.04763 & 0.0063 \\ 0.0063 & 0.0363 \end{bmatrix})$$

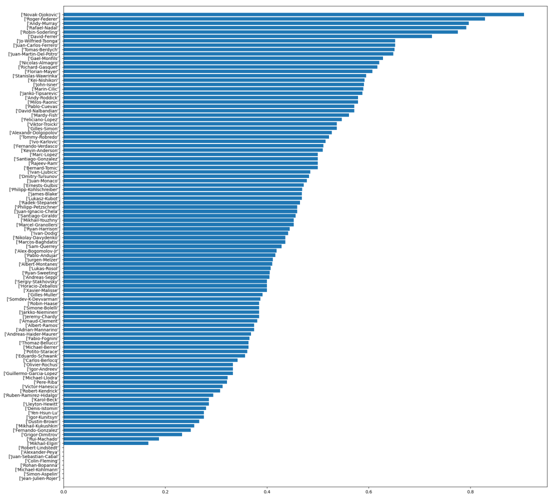
As expected, the samples are effectively uncorrelated - hence it is not worth modelling the skills in a joint way. I checked this for several different players and the correlations were all small. The direct estimation is much more crude, and is likely to be more susceptible to variations in the particular sample. Therefore the first method is used to construct a table for the skills.

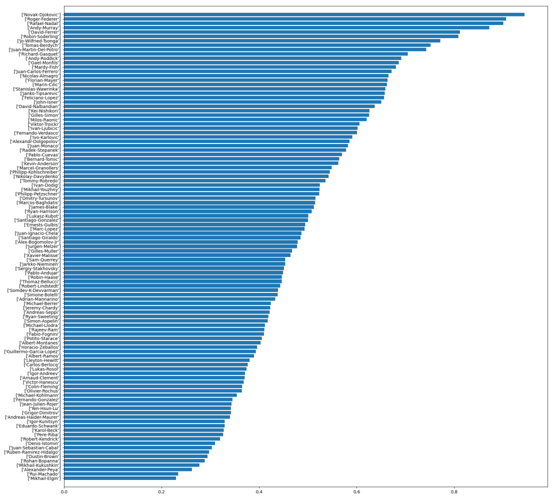
	Player 2			
Player 1	Djokovic	Nadal	Federer	Murray
Djokovic		0.9207	0.8830	0.9731
Nadal	0.0793		0.4048	0.7541
Federer	0.1170	0.5952		0.8172
Murray	0.0269	0.2459	0.1827	

Table 4: Probability of player 1 having higher skill than player 2 according to the gibbs samples

These values are roughly in agreement with the MP values, allowing for some variation due to dependence on the specific samples in the gibbs estimate. If we generated new gibbs samples we'd expect slightly different probabilities, whereas the MP values should stay the same.

1.5 Question 5


(a) Empirical


(b) Gibbs

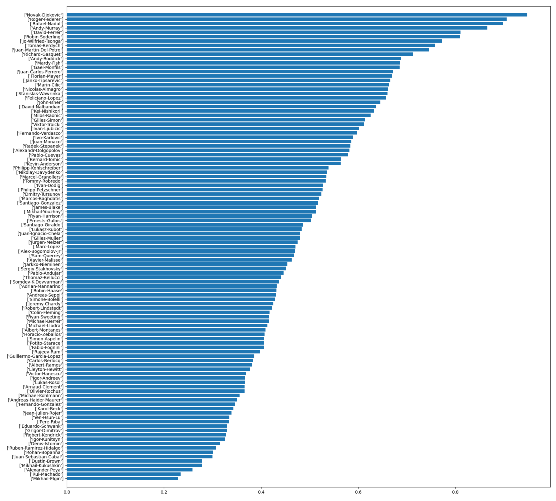

(c) Message Passing

Figure 8: Player Ranking

Above are ordered plots of the average winning probability of each player when playing every other player. Firstly we notice that the GS and MP rankings are very similar - starting with Djokovic at over 0.9, then dropping sharply into a quite closely contested section before dropping off to lower ranked players such as Mikhail Elgin at around 0.2. This gives a strong indication that the MP and GS methods are both converging to equivalent objects as described in question 2. The two methods do disagree slightly on the rankings in the central section - this is most likely down to the slower convergence of the GS, and we may expect different results if re-run, or more similar results to MP if run over a larger number of samples. Therefore of the two methods, MP seems preferable due to it's fast convergence times.

One clear shortcoming of the empirical method is that it has assigned eight players with zero probability of winning (clearly since they did not win a game in the past year). This goes against our intuition since no player would be entering a game with zero chance of winning. Some of these bottom players are ranked quite highly using the other methods (for example Robert Lindstedt has an average winning probability of 0.4). This demonstrates how the TrueSkill model is taking into account the skill of each players opponent, thereby quantifying whether each loss is a *good* loss or a *bad* loss, and similarly for wins.

Word Count: 992 according to Overleaf