CUED - Engineering Tripos Part IIB 2024-2025 Module Coursework

Module	4F13	Title of report	Coursework #2: Probabilistic Ranking

Date submitted: 21/11/25	Assessment for this module is ☐ 100% / ☑ 25% coursework of which this assignment forms _____ %

UNDERGRADUATE and POST GRADUATE STUDENTS

Candidate number:	rn436	☑ Undergraduate ☐ Post graduate

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:			

Marker: Date:

4F13: Probabilistic Ranking
Coursework 2

Raghavendra Narayan Rao

November 21, 2025

Abstract

This report investigated Metrapolis-Hastings Monte Carlo Markov Chain (MCMC) and Expectation propogation (EP) to rank tennis players based on match data. EP converges to a similar distribution as MCMC but requires only 20 iterations (as opposed to 2000).

1 Task a - Metropolis-Hastings MCMC

In this task, we ran a MCMC algorithm. The proposal distribution used is symmetric: $g(x'|x) = \mathcal{N}(x, 1.4^2) = \frac{1}{\sqrt{2\pi(1.4)^2}} \exp\left(-\frac{(x'-x)^2}{2(1.4)^2}\right) = \mathcal{N}(x', 1.4^2) = g(x|x')$. This allows for the acceptance ratio to be only dependant on the stationary distribution. The corresponding Python code is provided in Listing 1.

```python
num_its = 2000 # number of iterations

# perform Metropolis MCMC sampling
skill_samples, acceptance_ratios = MH_sample(games
    , num_players, num_its, std=1.4)

# burn-in period
skill_samples_no_burnin = skill_samples[:,num_its
    //2:]
```

Listing 1: MCMC Code

1.1 Choice of Proposal Distribution: variance = 1.4^2

This choice of variance allows for the acceptance ratio to have a $\mu = 0.234, \sigma = 0.035$ over 1000 iterations (after burn-in). 0.234 is an "optimal" acceptance rate calculated by [2] and suitable for a high dimensional MCMC (200 dimensions = number of players) as used here.

1.2 Computational Complexity of Metropolis-Hastings sampler

In each MCMC iteration, the algorithm iterates over all users and each of their matches. Therefore, $2 \times \text{num\_games} = \mathcal{O}(\text{num\_games})$ is the complexity of one iteration. The total complexity is then $\mathcal{O}(\text{num\_its} \times \text{num\_games})$

1.3 Autocorrelation and Burn-in Times

One definition of Autocorrelation Time is τ in $Cov(X(s), X(s+t)) \sim e^{-t/\tau}$. Figure 2 uses an exponential fit to obtain $\tau = 8.9$ and $\tau = 4.5$ for the best and worst player respectively. By 30 iterations, most players' samples tend to become uncorrelated. We can estimate the effective sample size $= \frac{1000}{30} \approx 33.333$. This is sufficient to approximate a gaussian (by CLT).

By calculating the Gelman-Rubin statistic on the best and worst players, ($1.015 > \hat{R} > 1.0019$), we have statistically shown that the posterior has converged with a burn-in of 1000. Visually, figure 1 shows that the skill level stabalises by 500 iterations.

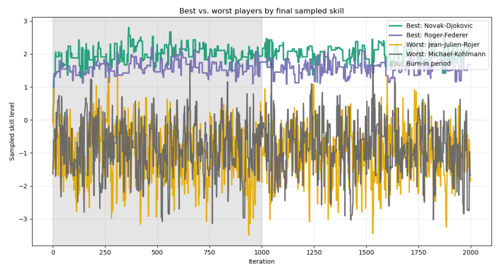
Figure 1: Skill level of Best and Worst players

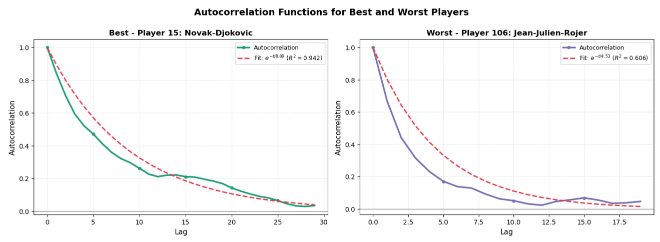
Figure 2: Autocorrelation Times

2 Task b - Convergence

Both algorithms converge to the posterior distribution of the players' skills. However, the EP algorithm has assumed that the posterior will be unimodal and hence well approximated by a Gaussian. Therefore, the algorithm only tracks the mean and variance of each player's skill. This however assumes that the covariance matrix is purely diagonal but we know this is incorrect as players' skills are correlated (same coach etc.).

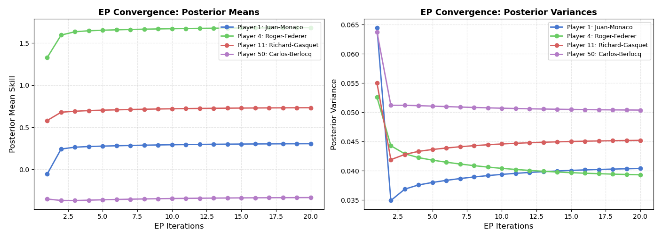
Figure 3: Expectation Propogation

The Gelman-Rubin statistic is used to test MCMC's convergence (check task a where the convergence of MCMC is discussed under burn-in-times). The EP algorithm converges extremely quickly with very little variation (visually check Figure 3). The means converged within 5 iterations. The variances however take about 25 iterations to stabalise for certain players. However, EP does not converge to the true posterior as there are cycles in this factor graph.

The MCMC algorithm converges to the true posterior distribution but it is much more flexible in the posteriors it can represent.

```
ep_post = exprop(games, num_players, num_its=20)
```

Listing 2: Expectation Propogation

3 Task c - Probability Table

Figure 4 highlights the difference in the two probabilities. The probability that the skill of one player is higher than the other reflects an epistemic uncertainty. It is devoid of any randomness from the specific outcomes of the match.

$$P(X_i - X_j > 0) = \Phi\left(\frac{\mu_i - \mu_j}{\sqrt{\sigma_i^2 + \sigma_j^2}}\right)$$

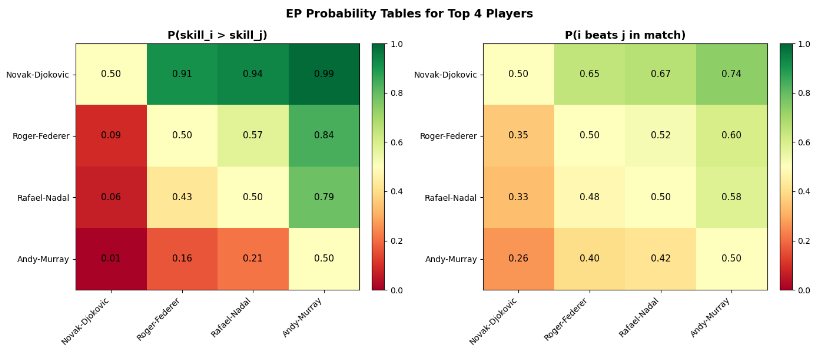

Figure 4: probabilities that the skill of one player is higher than the other (left) vs probabilities of one player winning a match between the two (right)

The probability of one player winning a match between the two reflects aleatoric uncertainty. This includes the randomness of a match. Therefore, using the probit model (cdf),

$$P(i \text{ beats } j) = \iint \Phi(X_i - X_j)\mathcal{N}(X_i;\mu_i,\sigma_i^2)\mathcal{N}(X_j;\mu_j,\sigma_j^2)\, dX_i\, dX_j = \Phi\left(\frac{\mu_i - \mu_j}{\sqrt{1+\sigma_i^2+\sigma_j^2}}\right)$$

The second probability has higher variance term in the denominator, which is due to some randomness of a win in a match. Therefore, the cdf expression will result in a lower probability of winning (if above $\frac{1}{2}$) and a higher probability of winning (if below $\frac{1}{2}$).

4 Task d - Marginal vs Joint vs Samples

Figure 5 computes the probabilities of player i's skill being higher than player j, which standerdises the comparison with EP's 4x4 table in part c.

$$P(X_i > X_j) = \begin{cases} \Phi\left(\frac{\mu_i-\mu_j}{\sqrt{\sigma_i^2+\sigma_j^2}}\right) & \text{(1) Marginal} \\ \Phi\left(\frac{\mu_i-\mu_j}{\sqrt{\Sigma_{ii}+\Sigma_{jj}-2\Sigma_{ij}}}\right) & \text{(2) Joint} \\ \frac{1}{N}\sum_{k=1}^{N}\mathbf{1}\big(X_i^{(k)} > X_j^{(k)}\big) & \text{(3) Samples} \end{cases}$$

Using a joint gaussian is better than the marginal as players are clearly correlated with each other (discussed in part b). However, these are still just gaussian approximations. The best method is method 3 as the MCMC samples capture (asymptotically) the true posterior anyway. However, all 3 methods produce very similar 4x4 table, which is also very similar to EP. However, method 3 is more confident that everyone will lose to Djokovic (the first column probabilities are lower in method 3).

5 Task e - Comparison of all methods

5.1 Comparison of Ranking Methods

1. Empirical Win Rates

Pros: Computationally cheap and interpretable.

Cons: Does not differentiate between wins against strong and weak opponents. Severely biased for players with few games (Figure 6 shows weakest players with score of -1 but 4 of them only played a single game and lost). The distribution is also skewed due to this.

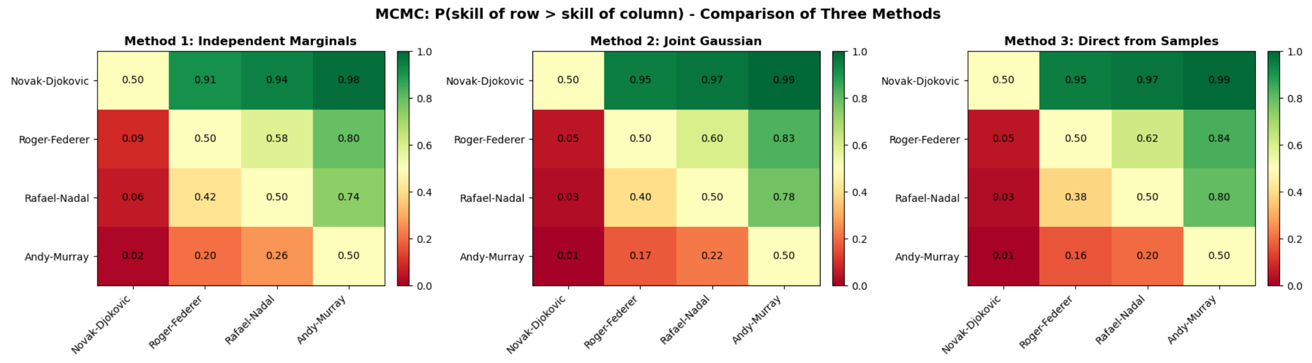

Figure 5: Marginal (left), Joint (middle), Samples (right)

2. **MCMC**

   Pros: Wins against stronger opponents is worth more. Regularizes players with few games toward prior (skill ≈ 0). Asymptotically exact inference.

   Cons: Computationally expensive (~ 2000 iterations). Requires tuning proposal distribution variance to obtain suitable acceptance ratio. Stochastic so not repeatable (unless seed used in code).

3. **EP**

   Pros: Much faster convergence compared to MCMC. Deterministic so easily reproducible.

   Cons: Assumes Gaussian posteriors which may not be suitable in certain situations. Due to cycles in this factor graph (player A plays player B. Player B plays player C. Player C plays player A), the posterior obtained will not be exact.

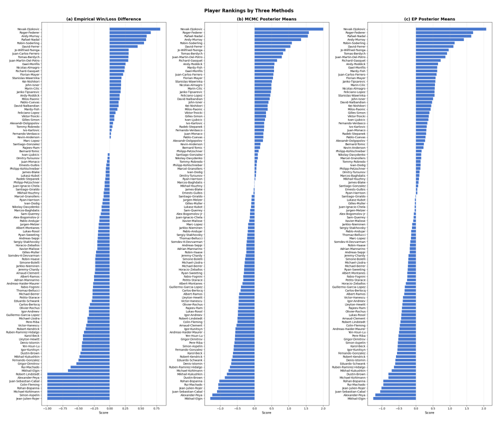

Figure 6: Empirical (left), MCMC (middle), EP (right)

References

[1] *GPy.kern.src Package* SheffieldML, 2020 (Accessed 03 Oct 2025). Available at https://gpy.readthedocs.io.

[2] A. Gelman, G. O. Roberts, and W. R. Gilks. Efficient Metropolis jumping rules. In J. O. Bernardo, J. M. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics*, volume 5, pages 599–607. Oxford University Press, 1996.