CUED - Engineering Tripos Part IIB 2019-2020 **Module Coursework**

Module		Title of report	
Date submitted:		Assessment for this module is ☐ 100% / ☐ 25% coursework of which this assignment forms _____ %	

UNDERGRADUATE STUDENTS ONLY			**POST GRADUATE STUDENTS ONLY**			
Candidate number:			Name:		College:	

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:			
PRESENTATION	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Overall assessment (circle grade)		A*	**A**	B	C	D
Guideline standard		>75%	**65-75%**	55-65%	40-55%	<40%
Penalty for lateness:		*20% of marks per week or part week that the work is late.*				

Marker: Date:

4F13 Probabilistic Machine Learning - True Skill Ranking

Lawrence Tray
St John's College

November 17, 2020

Abstract

This report outlines the results of the second coursework for 4F13.

Contents

1 Questions

1.a Gibbs Sampling

Gibbs sampling is used to sample from a multi-variate distribution by sequentially sampling from univariate conditionals. In the True-Skill model, we can approximate the conditional distribution with a 1-D Gaussian of a certain mean and variance - which are functions of the other variables. The modified code to compute these samples is given in listing 1.

Listing 1: Gibbs sampling additions

```
m = np.zeros((M, 1))
for p in range(M):
        # fill in m[p] prediction (natural param conditional)
        wins_array = np.array(G[:, 0] == p).astype(int)
        loss_array = np.array(G[:, 1] == p).astype(int)
        m[p] = np.dot(t[:,0], (wins_array - loss_array))

iS = np.zeros((M, M))  # Container for sum of precision matrices (likelihood terms)
for g in range(N):
        # Build the iS matrix
        winner = G[g, 0]
        loser = G[g, 1]

        iS[winner, winner] += 1
        iS[winner, loser] -= 1
        iS[loser, winner] -= 1
        iS[loser, loser] += 1
```

We plot the sampled player skills for a few players (figure 1). These data do appear noisy (in some sense random) but it appears that neighbouring samples are strongly correlated. Furthermore it is hard to tell how soon the Gibbs sampler transitions into a stable probability density region.

To investigate the burn-in time (iterations until the Gibbs is in a stationary state), we can plot the population mean and standard deviation at each Gibbs iteration - figure 2. The population mean gives us little information but we see that the standard deviation converges to a steady value fairly quickly (only after 10 iterations). We can therefore set the burn-in time to $b = 10$ and we would discard all preceding samples.

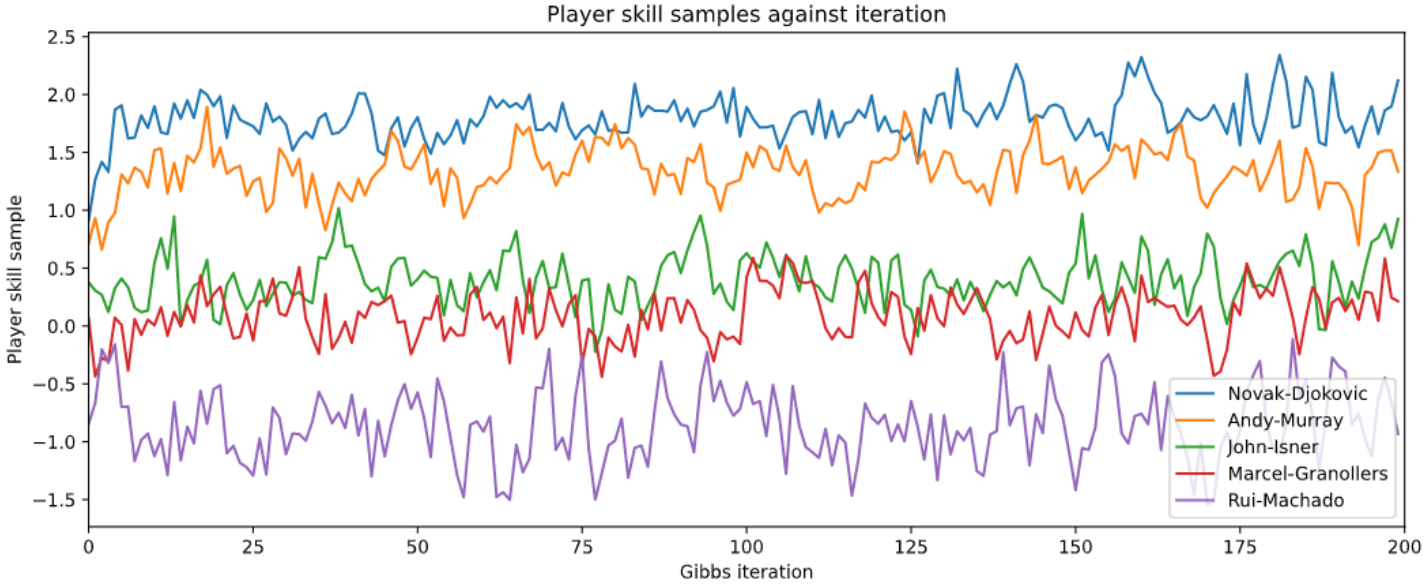


Figure 1: Gibbs skill samples for various players

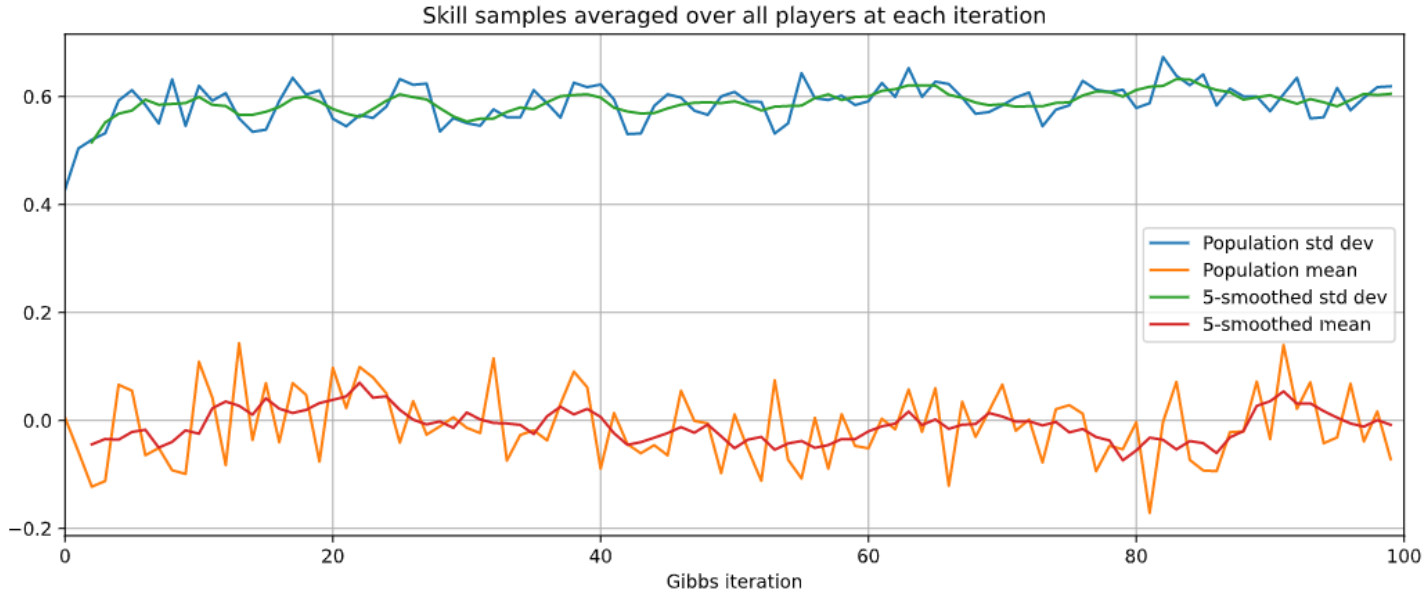


Figure 2: Gibbs skill samples averaged over all players at each iteration

However, there is an additional wrinkle, the plot in figure 1 is smoother than it should be; neighbouring samples are not independent. To test this hypothesis, we plot the auto-correlation of each player's skill samples (figure 3). For samples to be approximately independent, their correlation should be close to 0. This is only true for all players for an offset of at least $k = 10$. Nevertheless, we get good enough for $k = 5$ - excepting one player: Djokovic.

In summary, to run our Gibbs sampler efficiently and reliably, we only use samples after a burn-in period of $b = 10$ iterations. Even then, we thin the samples such that we only keep every t'th ($t = 5$) sample to ensure that they are approximately independent.

1.b EP - Message Passing

1.c EP - Top Four Head to Head

Suppose that a player i goes up against j. In the True-Skill model, we denote their skill by $w_i \sim \mathcal{N}(m_i, \sigma_i^2)$ and $w_j \sim \mathcal{N}(m_j, \sigma_j^2)$ respectively. The mean and variance parameters are estimated through message passing (the variance is just the inverse of the precision).

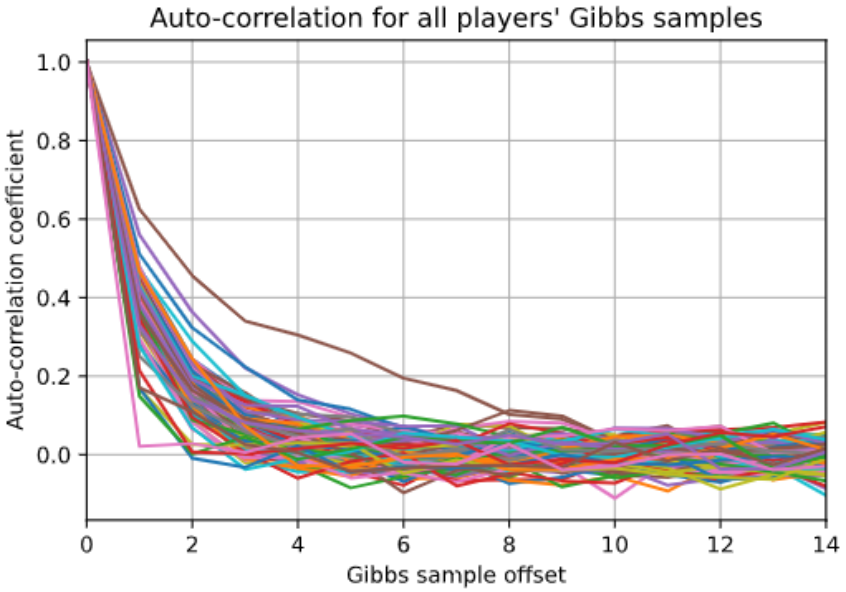
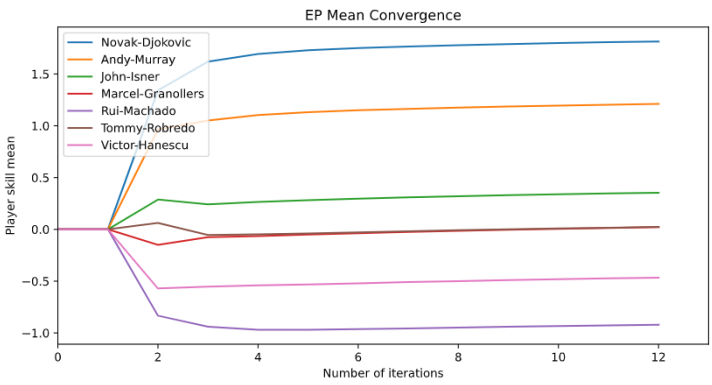
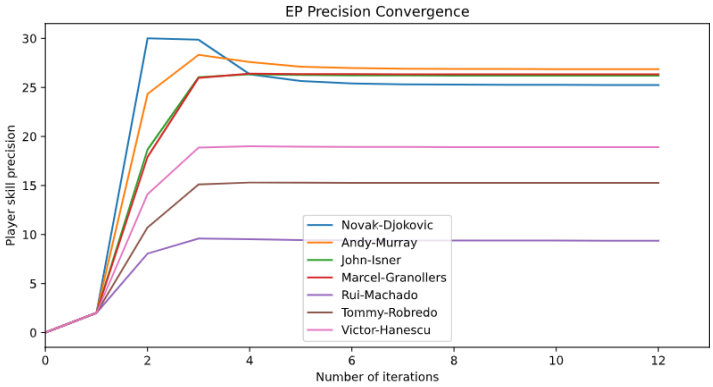


Figure 3: Auto-correlation for Gibbs sampler for all players



(a) EP player skill mean evolution



(b) EP player skill precision evolution

Figure 4: EP player skill parameter evolution against iteration

The performance difference $s_{ij} = w_i - w_j$ is corrupted by Gaussian noise of unit variance $t_{ij} = s_{ij} + n$ to account for performance inconsistency ($n \sim \mathcal{N}(0, 1)$). The match result is then given by the sign of t_{ij} such that $t_{ij} > 0 \Rightarrow i$ wins and j wins otherwise.

Having run message-passing, we wish to compute the probability that i is more skilful than j: $P(s_{ij} > 0)$. This is different from the probability that i beats j in a head-to-head: $P(t_{ij} > 0)$. These probabilities are easy to compute by noting that $w_i \perp\!\!\!\perp w_j \perp\!\!\!\perp n \perp\!\!\!\perp w_i$, so means and variances combine additively:

$$s_{ij} = w_i - w_j \sim \mathcal{N}(\Delta m_{ij} = m_i - m_j, \sigma_{ij}^2 = \sigma_i^2 + \sigma_j^2) \tag{1}$$

$$t_{ij} = s_{ij} + n \sim \mathcal{N}(\Delta m_{ij} = m_i - m_j, \bar{\sigma}_{ij}^2 = \sigma_i^2 + \sigma_j^2 + 1) \tag{2}$$

For any Gaussian r.v. $X \sim \mathcal{N}(\mu, \sigma^2)$, $P(X > 0) = \Phi(\mu/\sigma)$ - where $\Phi(\cdot)$ is the standard Gaussian c.d.f. Applying this result to equations 1 and 2 for the top four ranked players, we obtain table 1.

We clearly see that the probability of player i being more skilful than j is always more extreme (further from 0.5) than the probability of winning a head-to-head. This makes sense mathematically as the variance $V[t_{ij}] > V[s_{ij}]$. It also makes sense intuitively because an individual match is subject to performance variation. If I were to play a single point against Federer, I might win. However, as the number of points increases the chance of winning the majority falls to (in my case) 0. This is the difference between being a better player and just getting lucky.

$P(s_{ij} > 0)$	**Djokovic**	**Federer**	**Nadal**	**Murray**
Djokovic	-	0.92	0.95	0.98
Federer	0.08	-	0.59	0.79
Nadal	0.05	0.41	-	0.73
Murray	0.02	0.21	0.27	-

(a) Prob. row player is more skilful

$P(t_{ij} > 0)$	**Djokovic**	**Federer**	**Nadal**	**Murray**
Djokovic	-	0.64	0.66	0.71
Federer	0.36	-	0.52	0.58
Nadal	0.34	0.48	-	0.56
Murray	0.29	0.42	0.44	-

(b) Prob. row player wins a head-to-head

Table 1: Top four players comparison

If Djokovic played an infinite number of matches against Murray, a priori we can be 98% confident that he will win the majority. However, we would expect Djokovic to only win 71% of these matches - not 98%.

1.d Gibbs - Nadal v Djokovic

1.e Method Comparison: Win ratio, Gibbs and EP

We go about ranking players using different methods: basic win ratio model; Gibbs sampling for True-Skill model; and EP estimates for True-Skill.

The win ratio model assumes that each player has a fixed probability of winning their match irrespective of their opponent. If a player has played n and won k of them we classify their win percentage $p \sim \mathcal{N}(k/n, k(n-k)/n^2)$ - this applies standard results for combining n independent Bernoulli r.v.'s. We plot the mean and standard deviation of p on figure 5.

With Gibbs sampling, we sample the player skill from conditional distributions and update the sample for each player at every iteration. We can restrict the samples to those produced after the burn-in period $b = 20$ and thin further such that every t'th sample is kept $t = 5$. This gives us a set of independent samples for the skill of each player and we can calculate the mean and variance of this set empirically (plotted on figure 6).

For Expectation Propagation, we run the Message Passing algorithm for 3 iterations as that is enough to achieve convergence. This algorithm returns the expected means and precisions (inverse variance) for the skill of each player and so we can plot these directly (figure 7)

Words: XX

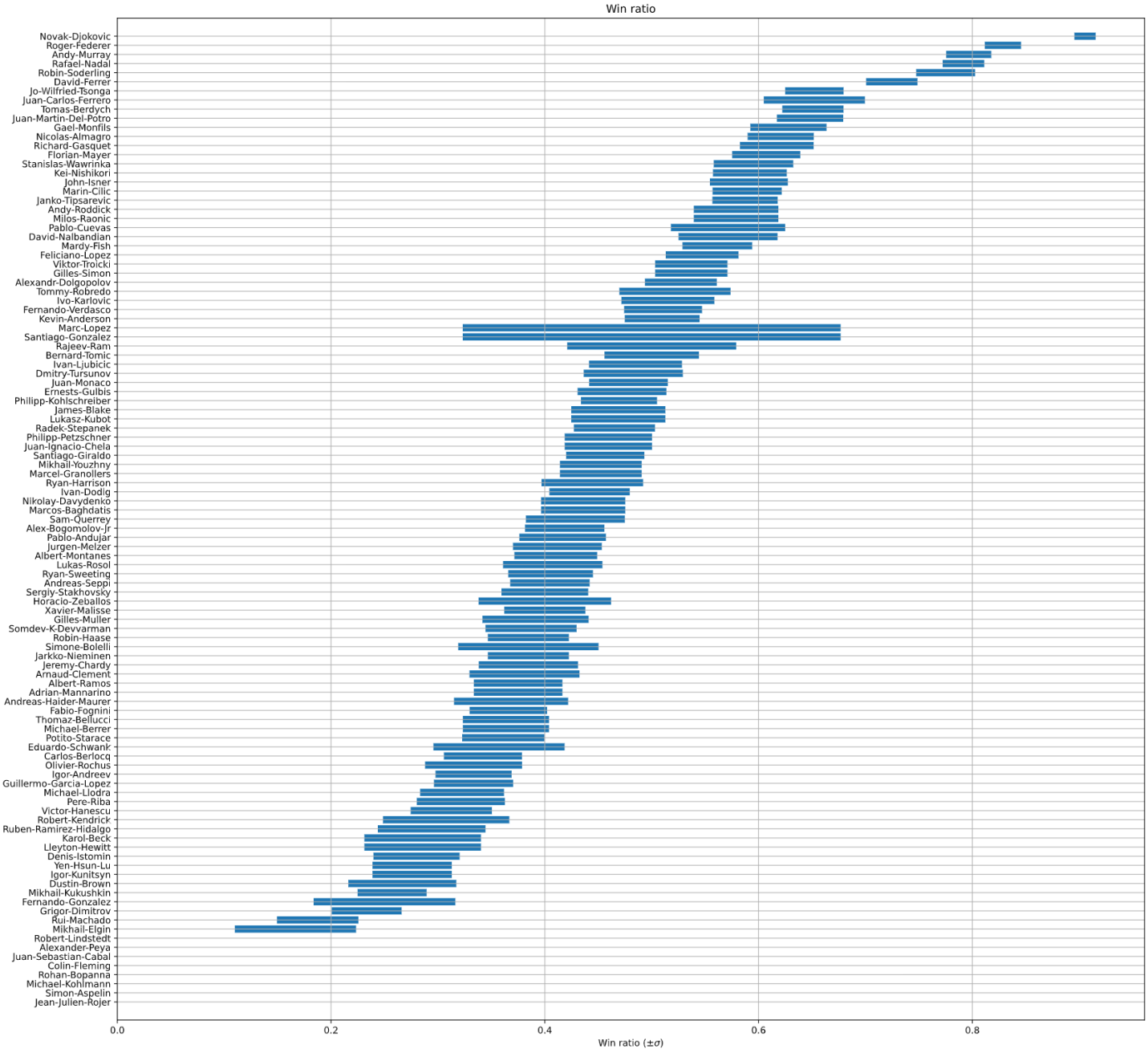


Figure 5: Player rankings by win ratio

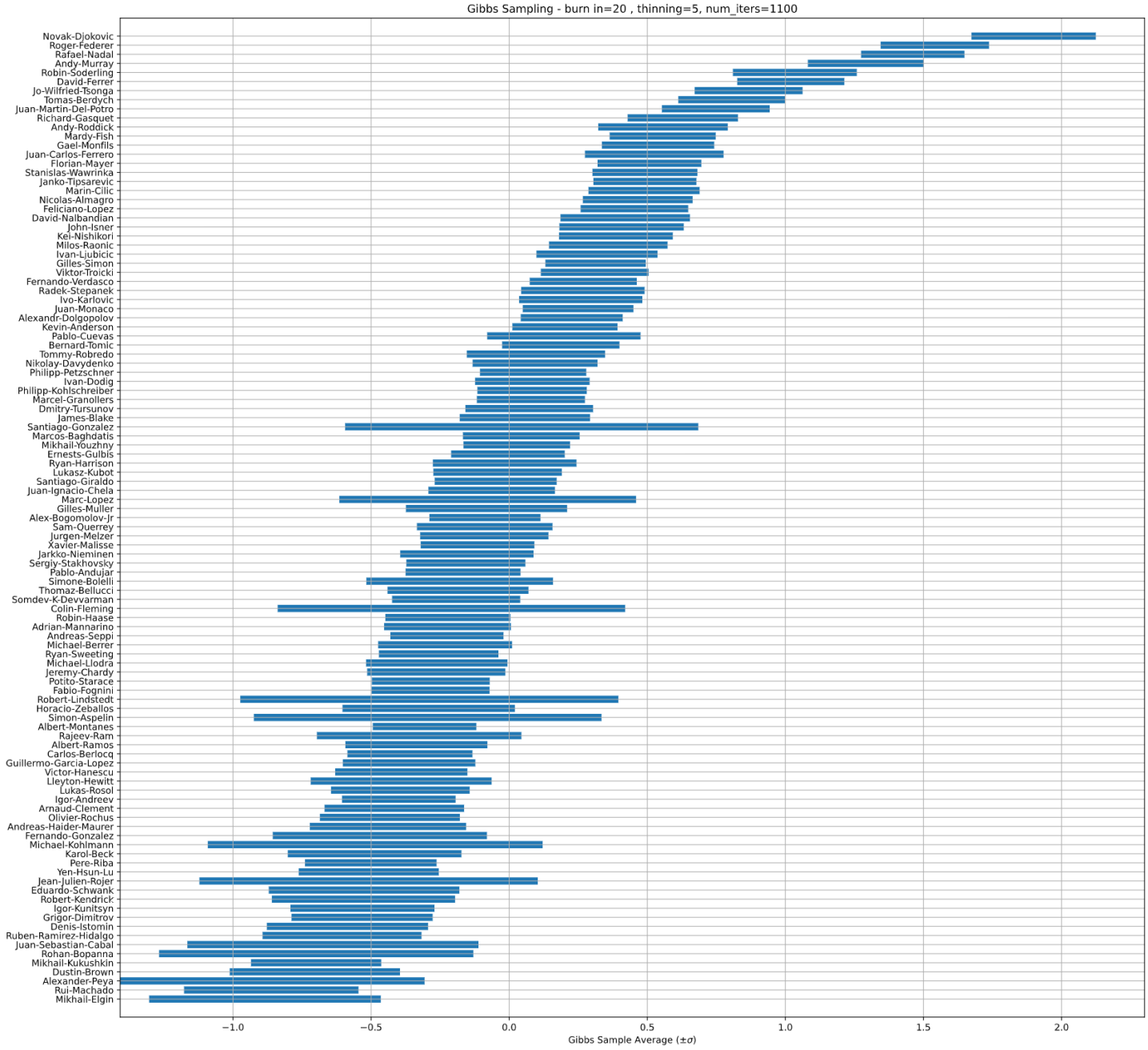


Figure 6: Players rankings by mean of Gibbs skill samples

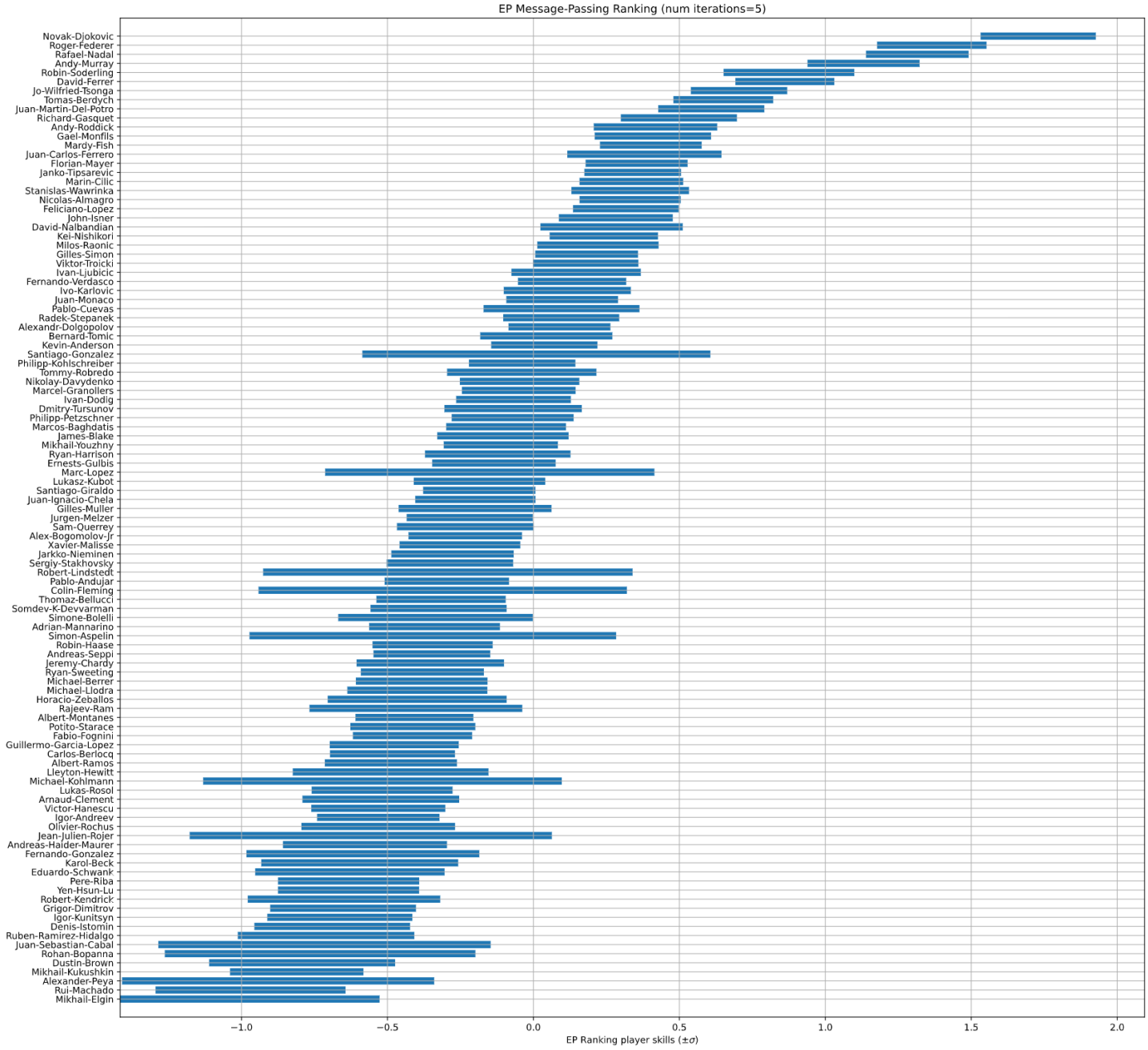


Figure 7: Player rankings by mean of EP-estimated skills