Probabilistic Machine Learning: Probabilistic Ranking

Abhinav Heble

November 19, 2021

1 Abstract

Competition is a central biological trait and the driving force behind most sporting competitions. The challenge of ranking competitors has been discussed for centuries; the birth of machine learning has brought with it novel ways to address this issue. This report explores ranking male tennis players in 2011 based on their match results from that year with three methods for estimating their skill ratings - empirical game averages, inference with Gibbs sampling and message passing with expectation propagation. The latter method forms the basis for the Trueskill algorithm developed by Microsoft.

2 Gibbs Sampling For A Simple Model

The model used is

$$y = sign(w_1 - w_2 + n) \tag{1}$$

Here, y is the game outcome, w_1 and w_2 are the player skill ratings and n is standard Gaussian noise. The likelihood for the game outcome is Gaussian:

$$p(y|w_1, w_2) = \Phi(y(w_1 - w_2)) \tag{2}$$

The game outcomes are known, so the posterior over skills is required to estimate the players' skill ratings. This distribution is intractable, but Gibbs sampling can be used to approximate the joint posterior distribution. It is important to first explore whether Gibbs sampling converges for this data.

```python
for ind,(p1,p2) in enumerate(G): m[p1]+=t[ind]; m[p2]-=t[ind] # filling the mean array
for g in range(N): # filling the inverse covariance matrix
    iS[G[g,0],G[g,0]]+=1; iS[G[g,1],G[g,1]]+=1
    iS[G[g,0],G[g,1]]-=1; iS[G[g,1],G[g,0]]-=1
```

Code Listing 1: Completed gibbsrank code

Figure 1 is obtained using Listing 1 to complete the gibbsrank.py code. The mean matrix was filled by iterating through the games as opposed to the matrix itself for efficiency. The skill ratings of the three players vary more for the initial 25 iterations than for the remainder, so a burn-in period of 50 iterations is sufficient to compensate for the difficulty in determining the exact burn-in time.

Skill Ratings over 1100 iterations of Gibbs sampling for Players 1,50 and 95

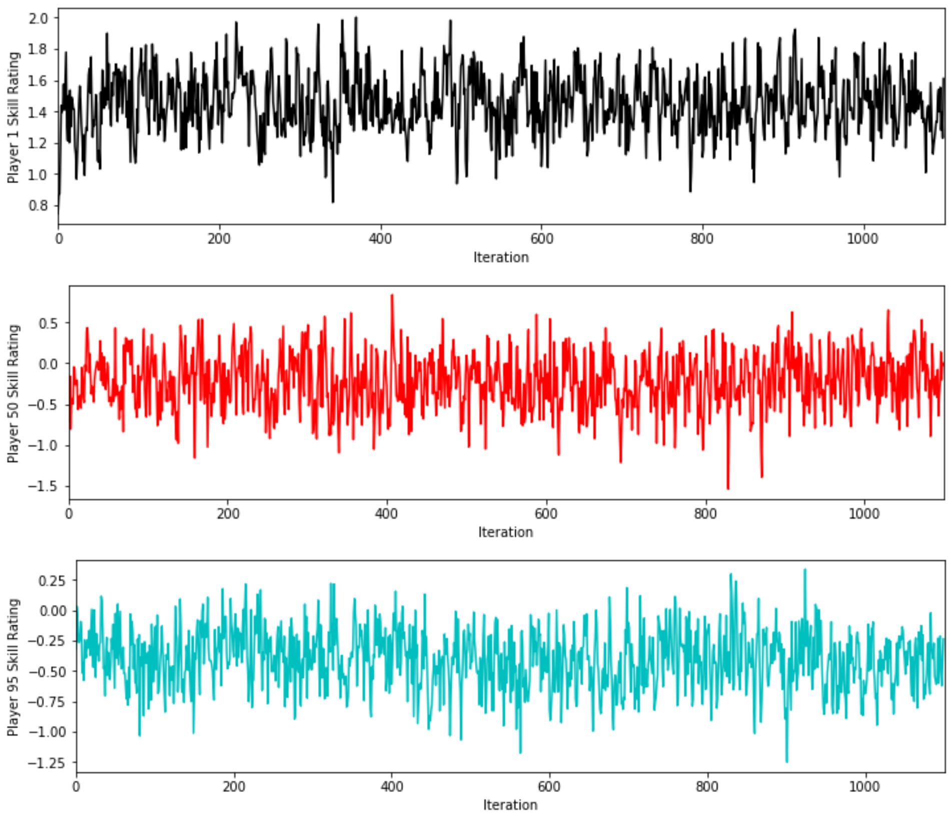

Figure 1: The variations in skill ratings for three players over 1100 iterations of Gibbs sampling.

The autocorrelations of the players' skill ratings with different lag times

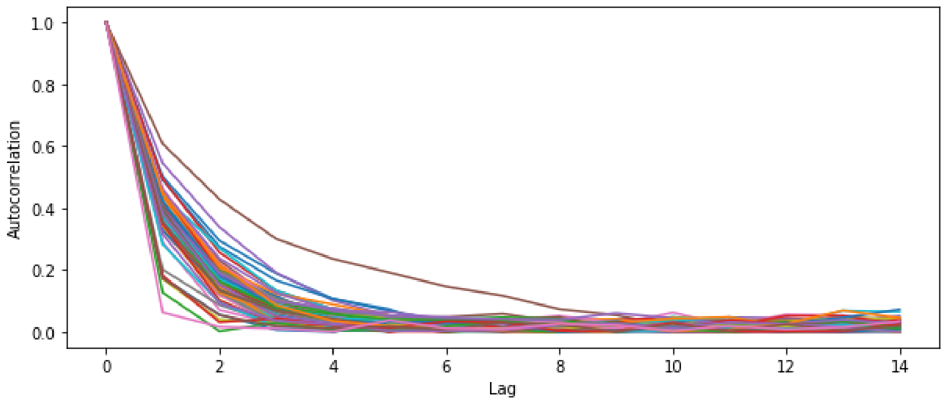

Figure 2: A graph showing the autocorrelation of the players' skill ratings plotted against lag.

The autocorrelation times for the players, seen in Figure 2, appear to approach zero at a lag of 10. Gibbs sampling has a high computational expense (around 27 iterations per second), so I will run the sampler for 3000 iterations, removing the first 50 samples to account for the burn-in time and thinning by taking every tenth sample. The samples for the same three players obtained after this method, seen in Figure 3, appear more uncorrelated than before and the new autocorrelations, seen in Figure 4, approach zero for neighbouring samples, supporting this choice of method.

Skill Ratings over 3000 iterations of Gibbs sampling with burn-in and thinning accounted for

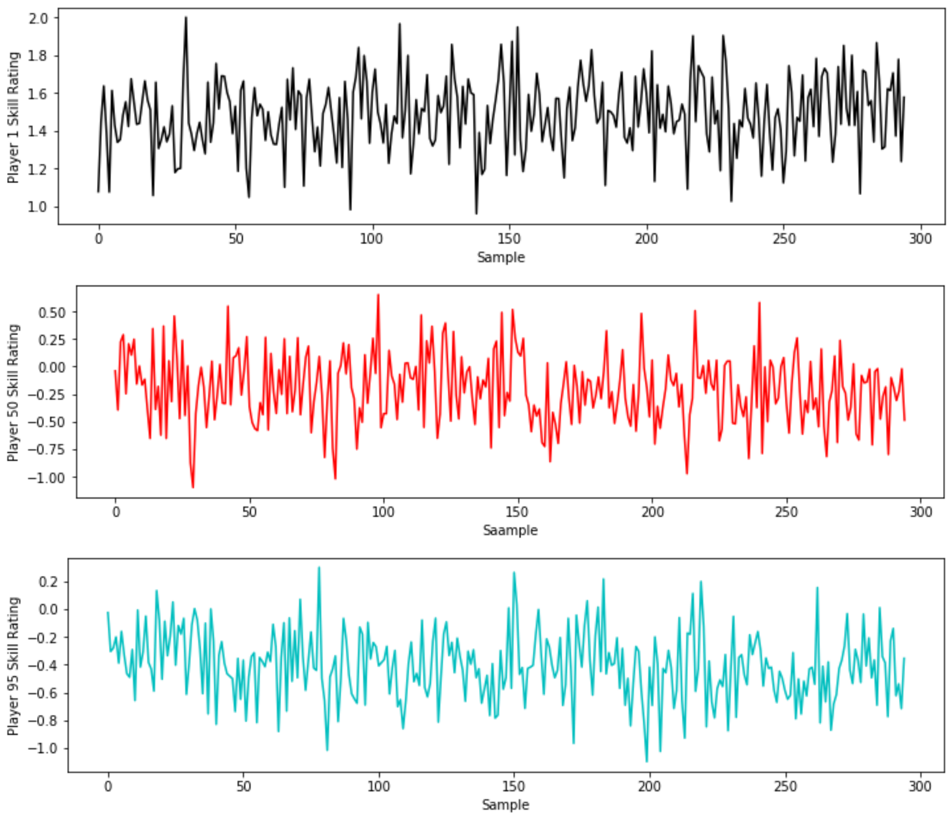

Figure 3: The variations in skill ratings for three players over 3000 iterations of Gibbs sampling after removing the first 50 samples and selecting every 10 samples.

The autocorrelations of the players' skill ratings with different lag times after a burn-in period and thinning

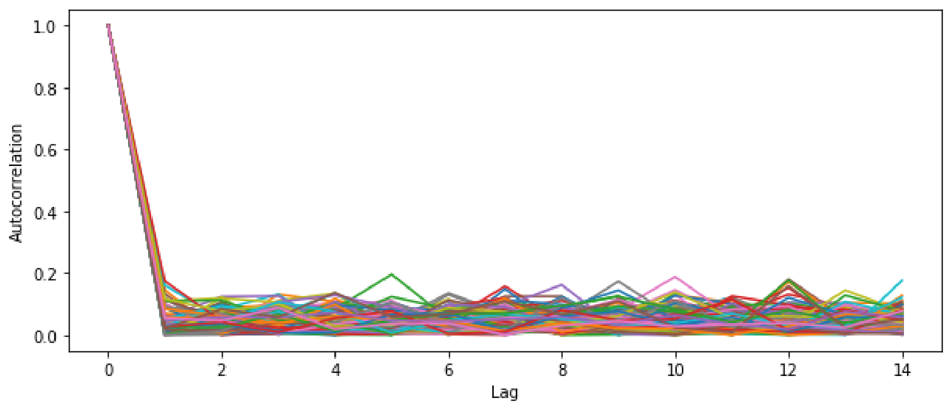

Figure 4: A graph showing the autocorrelation of the players' skill ratings plotted against the lag after burn-in is accounted for and thinning is applied.

3 Inference With Message Passing And Expectation Propagation

In Gibbs sampling, the approximate distribution obtained after each iteration is used as a prior for the next iteration, resulting in the stochastic output seen in Figure 5. Convergence occurs when the sampled distribution approximates the intractable joint Gaussian distribution of the data. Figure 6 shows the sampled skill ratings for player 1 after a burn-in period of 50 iterations and thinning by taking every tenth sample. As the number of samples increases from 10 to 250, the sampled distribution appears to converge to a Gaussian distribution with a mean of 1.46 and a variance of 0.02, so it is reasonable to assume 3000 iterations are sufficient for convergence.

Message passing is a deterministic algorithm that assumes the marginals are Gaussian and sends means and variances as the messages. Convergence occurs when the means and variances at the next iteration are equal to the values at the current. Since samples are generated after convergence, the curves for message passing in Figure 5 are smooth, and convergence can be determined exactly depending on the desired accuracy. For example, player 1's skill rating converges to a mean of 1.4716 at 90 iterations.

Mean and variance values for 100 iterations of Gibbs sampling and EP

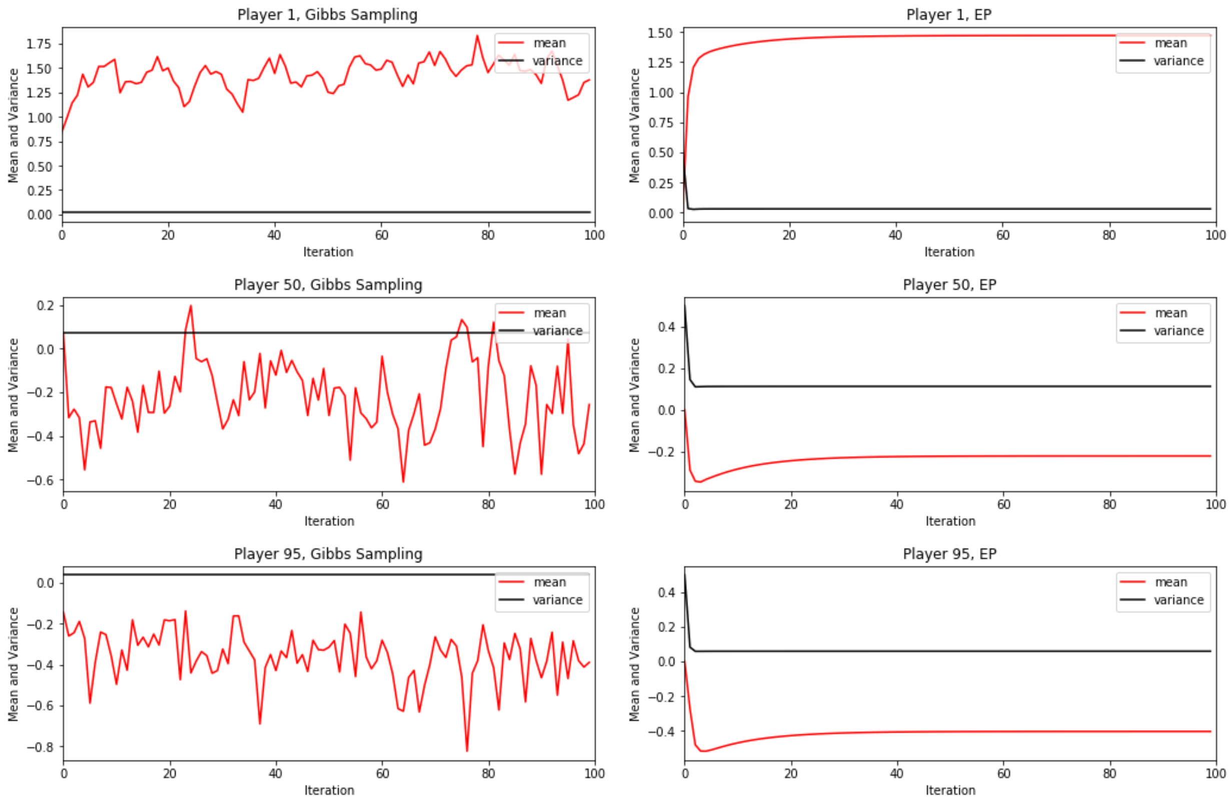

Figure 5: The means and variances used for skill rating calculations for the two methods for players 1, 50 and 95.

Player 1's skill rating samples for different numbers of iterations of Gibbs sampling

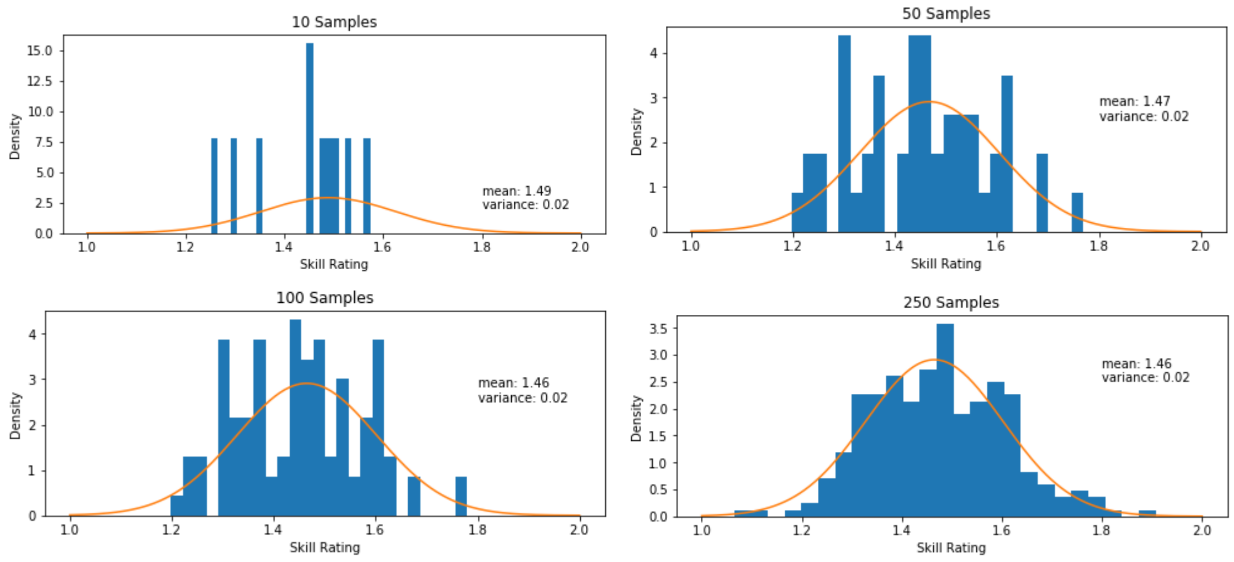

Figure 6: Histograms for the sampled skill ratings of player 1 after a burn-in period of 50 iterations and thinning using every tenth sample, with 100, 500, 1000 and 2500 iterations of Gibbs sampling.

4 Comparing The Top Four Players With Message Passing

The probability that that the skill of one player is greater than another, $p(w_1 > w_2)$, is equivalent to $p(w_1 - w_2 > 0)$. The approximated probability distribution is Gaussian:

$$p(w_1 - w_2 > 0) = \int_0^\infty N(w_1 - w_2|\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2) = 1 - \int_{-\infty}^0 N(w_1 - w_2|\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2) \tag{3}$$

```
prob_skill = lambda i, j: 1 - stats.norm.cdf(0,mu[i] - mu[j],math.sqrt((1/var[i]) + (1/var[j])))
```

Code Listing 2: Code to calculate the probability of player i's skill rating being greater than player j's skill rating.

Using this result, Table 1 is produced from the skill ratings of players 16, 1, 5 and 11 (the top 4 ranked players according to the lecture notes respectively) and the implemented code in Listing 2.

Player 1	Player 2			
	Novak Djokovic	Rafael Nadal	Roger Federer	Andy Murray
Novak Djokovic	-	0.9398	0.9089	0.9853
Rafael Nadal	0.0602	-	0.4272	0.7665
Roger Federer	0.0911	0.5728	-	0.8108
Andy Murray	0.0147	0.2335	0.1892	-

Table 1: Probabilities of the four top ranked players having a higher skill rating than each other.

On the other hand, the probability that one player wins in a game against another also takes into account noise:

$$p(y = 1|w_1 - w_2) = p(t > 0) = \int_0^\infty N(t|\mu_1 - \mu_2, 1 + \sigma_1^2 + \sigma_2^2) = 1 - \int_{-\infty}^0 N(t|\mu_1 - \mu_2, 1 + \sigma_1^2 + \sigma_2^2) \tag{4}$$

Listing 3 uses this result, by adding Gaussian noise with mean 0 and variance 1, to produce the probabilities shown in Table 2.

```
post_skill = lambda i, j: 1 - stats.norm.cdf(0,mu[i] - mu[j],math.sqrt(1 + (1/var[i]) + (1/var[j])))
```

Code Listing 3: Code to calculate the probability of player i winning against player j.

Player 1	Player 2			
	Novak Djokovic	Rafael Nadal	Roger Federer	Andy Murray
Novak Djokovic	-	0.6554	0.6380	0.7198
Rafael Nadal	0.3446	-	0.4816	0.5731
Roger Federer	0.3620	0.5184	-	0.5909
Andy Murray	0.2802	0.4269	0.4091	-

Table 2: Probabilities of the four top ranked players winning matches against each other.

The probabilities in Table 2 are much closer to 0.5 than the corresponding probabilities in Table 1, as they take into account the uncertainty in game outcome via the additive Gaussian noise. In real terms, this noise may compensate for differences in weather and court conditions, health, or other factors contributing to the outcomes of the games aside from the predicted skill ratings.

5 Comparing Nadal And Djokovic With Gibbs Sampling

This task compares the skill ratings of Djokovic and Nadal using three Gibbs sampling based methods. The first obtains the marginal distributions for Djokovic and Nadal as $N(w_d|1.8997, 0.0467)$ and $N(w_n|1.4845, 0.0374)$ respectively, using the code in Listing 4.

```
mu_i, var_i = np.mean(skill_samples[i][50::10]), np.var(skill_samples[i][50::10]) # mean and variance for
    player i
mu_j, var_j = np.mean(skill_samples[j][50::10]), np.var(skill_samples[j][50::10]) # mean and variance for
    player j
```

Code Listing 4: Code to calculate the means and variances for the marginal distributions of player i having a greater skill rating than player j.

Using the code in Listing 2, the probability that Djokovic has a greater skill rating than Nadal is calculated:

$$p(w_d - w_n > 0) = \int_0^\infty N(w_d - w_n|\mu_d - \mu_n, \sigma_d^2 + \sigma_n^2) = 1 - \int_{-\infty}^0 N(w_d - w_n|\mu_d - \mu_n, \sigma_d^2 + \sigma_n^2) = 0.9238 \tag{5}$$

The second method involves approximating the joint skills of both players by a bivariate Gaussian distribution. The covariance matrix can be calculated using the code in Listing 5 to give $p(w_d, w_n) = N(\begin{bmatrix} w_d \\ w_n \end{bmatrix} \mid \begin{bmatrix} 1.8997 \\ 1.4845 \end{bmatrix}, \begin{bmatrix} 0.0469 & 0.0093 \\ 0.0093 & 0.0375 \end{bmatrix})$.

```
joint_skills = [[],[]] # create 2D skill array
for i in range(50,num_iters,10):
    joint_skills[0].append(skill_ratings[15][i]) # fill Djokovic's skill ratings
    joint_skills[1].append(skill_ratings[0][i]) # fill Nadal's skill ratings
cov_mat = np.cov(np.array(joint_skills)) # calculate covariance matrix
```

Code Listing 5: Code to calculate the covariance matrix for the approximate joint Gaussian distribution of the skill ratings for Djokovic and Nadal.

The probability of Djokovic's skill rating being higher than Nadals is then:

$$p(w_d > w_n) = \int_{-\infty}^{\infty} \int_{-\infty}^{w_d} p(w_d, w_n) dw_n dw_d \tag{6}$$

This integral is estimated by taking a large number of random samples from the joint distribution and finding the proportion of those pairs of samples in which Djokovic's skill rating is greater than Nadal's. The implementation of this method, shown in Listing 6, returns $p(w_d > w_n) = 0.9470$.

```
tot_d, tot_n = 0,0 # variables to count number of greater skill ratings
samples = stats.multivariate_normal.rvs([1.8997,1.4845],[[0.0469,0.0093],[0.0093,0.0375]],size) # generate
for d, n in samples:
    if d>n: tot_d+=1 # Djokovic with greater skill rating
    else: tot_n+=1 # Nadal with greater skill rating
prob_d, prob_n = tot_d/size, tot_n/size # Calculate final estimated probabilities
```

Code Listing 6: Code to calculate the probability that Djokovic's skill rating is greater than Nadal's by sampling from the approximate joint Gaussian distribution.

The final method finds the proportion of pairs of generated samples for which one player has a greater skill rating than the other. This method returns $p(w_d > w_n) = 0.9593$.

The first method is least suitable, as it uses the marginal distributions rather than the joint distribution that the Gibbs sampler aims to converge to. The final method continues generating samples after convergence, so using the joint distribution with the converged means and covariances would allow more efficient generation and more samples to use for prediction. Therefore, the second method is best and the probabilities in Table 3 were calculated using this method. These agree with the values in Table 1, suggesting common convergence between the second method and message passing.

Player 1	Player 2			
	Novak Djokovic	Rafael Nadal	Roger Federer	Andy Murray
Novak Djokovic	-	0.9475	0.9140	0.9857
Rafael Nadal	0.0525	-	0.4108	0.7702
Roger Federer	0.0860	0.5892	-	0.8119
Andy Murray	0.0143	0.2298	0.1881	-

Table 3: Probabilities of the four top ranked players having a higher skill rating than each other using samples from a joint Gaussian distribution after convergence from Gibbs sampling.

6 Computing The Final Rankings

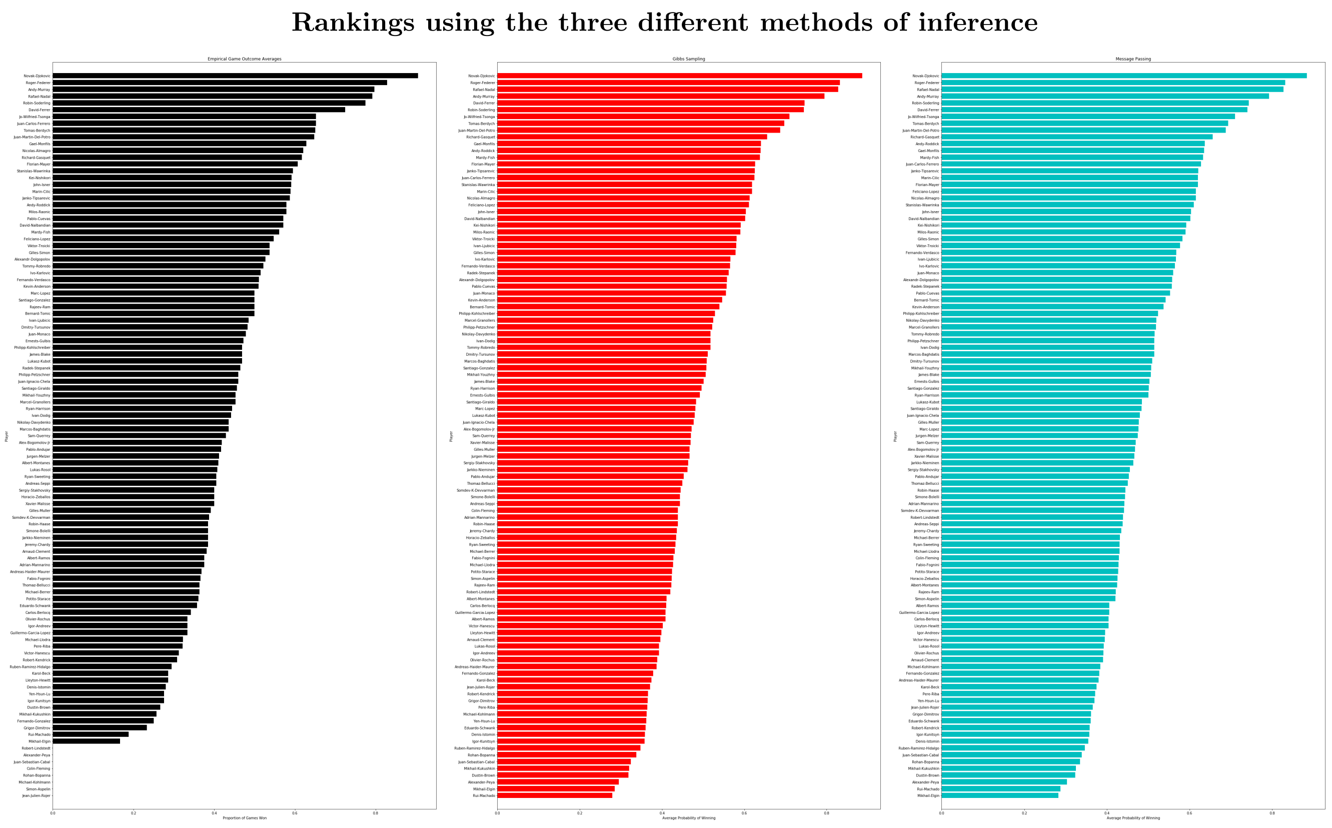

Figure 7: All the players ranked using empirical game outcome averages, Gibbs sampling and message passing.

Figure 7 shows the rankings of the players using the three different methods. The graph on the left was produced by calculating the proportion of wins of each player over the total number of games they played and sorting the resulting values, implemented in Listing 7. This method for ranking is unsuitable, as the players have played different numbers of games against different opponents yet wins are treated the same regardless. There are also 8 players who did not win a game and were given an unreasonable zero skill rating.

The middle graph uses Gibbs sampling to estimate the converged mean and covariance matrices and samples skill ratings from the resulting joint distribution. These samples are then used to predict the outcome of games, implemented in Listing 8. The graph on the right uses message passing to determine the means and variances for the skill ratings of the players, which are then used for game prediction.

The difference between Gibbs sampling and message passing rests with convergence. The thinning required for Gibbs sampling limits its efficiency whereas message passing does not suffer from this issue, while still converging to similar values. For this reason, message passing is the preferred method for ranking this data set.

```python
wins = [[0,0] for a in range(M)] # create wins matrix
for p1,p2 in G: wins[p1][0]+=1; wins[p1][1]+=1; wins[p2][1]+=1 # increment wins and games
for i, val in enumerate(wins): wins[i] = val[0]/val[1] # calculate proportion of wins
```

Code Listing 7: Code to calculate the proportion of wins from all the games each player played.

```python
cov = np.cov(samples) + np.identity(M) # create covariance matrix with noise
for a in range(num_samples):
    samples = stats.multivariate_normal.rvs(means,cov) # create samples using means and covariance
    for b, index in enumerate(sorted(range(len(samples)), key=lambda k: samples[k])): pred[index] += b/M
pred=pred/num_samples # normalise prediction array
```

Code Listing 8: Code to predict game outcomes from the approximated joint distribution.