4F13 Machine Learning: Coursework #2: Probabilistic Ranking

Due: 12:00 noon, Friday Nov 21st, 2025 online via moodle

Your answers must contain an explanation of what you do, and 2-4 central commands to achieve it (but complete listings are unnecessary). You must also give an *interpretation* of what the numerical values and graphs you provide *mean* – why are the results the way they are? **Each question should be labelled and answered separately and distinctly.** All questions carry approximately equal weight. Total combined length of answers must not exceed 5 sides of A4 (plus cover page), minimum 11pt font, 1 inch margins. Skeleton code is provided in Python.

In this assignment, you will be using the (binary) results of the 2011 ATP men's tennis singles for 107 players in a total of 1801 games (which these players played against each other in the 2011 season), to compute probabilistic rankings of the skills of these players.

The match data is provided in the file `tennis_data.mat`, which contains two matrices: `W`, whose `i`'th entry is the name of player `i`, and `G` is a 1801 by 2 matrix of the played games, one row per game: the first column is the identity of the player who won the game, and the second column contains the identity of the player who lost. Identical rows may appear multiple times, corresponding to two players having played each other several times with the same outcome.

a) Complete the code in `MHrank`, by adding the lines required to sample from the posterior distribution using the Metropolis-Hastings MCMC algorithm, as discussed in the lectures. Think carefully about the proposal distribution. Run the MCMC sampler for a reasonable length of time, e.g., for 2000 iterations. Plot some of the sampled player skills as a function of the sampler iteration. What are the burn in and auto-correlation times and how long would you run the MCMC sampler to get reliable results? Explain why. It may be helpful to look at the auto covariance coefficient, which can be calculated by the Python lines provided. What is the computational complexity of running the Metropolis-Hastings sampler, and why?

b) Perform the inferences by running message passing and EP using `eprank`. Explain the concept of *convergence* for both the Monte Carlo sampler and the message passing algorithms. What type of object are we converging to in the two cases, and how do you judge convergence, how many iterations are necessary?

c) Using the message passing algorithm, compute two 4 by 4 tables of probabilities for the 4 top players according to the ATP ranking (Djokovic, Nadal, Federer, Murray). First, compute the table for the probabilities that the skill of one player is higher than the other. Second, compute a table for the probability of one player winning a match between the two. Explain the difference.

d) For the Monte Carlo sampler, compare the skills of Nadal and Federer in three different ways: 1) based on approximating their marginal skills by Gaussians, 2) based on approximating their joint skills by a Gaussian or 3) directly from the samples. Which method is best? Using that method, derive a 4 by 4 table for the skills (not the game outcomes) and compare to that of the message passing algorithm (from part c)). What do you conclude about the two methods? Explain why.

e) Compare the rankings of players using predicted outcomes for three different methods of inference: 1) empirical game outcome averages, 2) predictions based on Monte Carlo sampling and 3) predictions based on the message passing algorithm. You may find the bar plot in `cw2.py` useful. Explain the differences and the pros and cons of the three different methods.