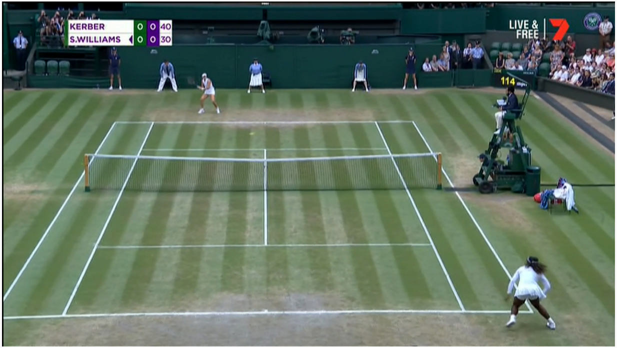
Serve and Return Glicko

An evaluation of a different approach to in-play win prediction in tennis

Martin Ingram, University of Melbourne, Australia

2nd July 2019

Introduction: In-play modelling in tennis



- We would like to update our win expectation given the points played so far.
- Challenge: balance between overreacting and ignoring information.

Setting the stage: The iid model

- Popular model in tennis: assume player win probabilities θ_1 and θ_2 on serve constant throughout match
- Useful for in-play prediction because we can quickly calculate conditional win probabilities $P(\mathrm{p1\ wins}|\mathrm{score}, \theta_1, \theta_2)$
- Example: if $\theta_1 = 0.6$ and $\theta_2 = 0.55$ and best of three sets:
  - Before match: 74.4%
  - If p1 wins first game: 80.8%
  - If p1 wins first set: 89.9%
- Tennis points are *not* iid, but the approximation is convenient and has been shown to be fairly good (Klaassen & Magnus 2001).
- To use the iid model, we need θ_1 and θ_2, and we may want to update them during the match.

Prior work for finding and updating θ_1, θ_2

- Prior win expectations on serve to use for in-play forecasting have been found in two ways:
    1. By using a model that directly predicts the serve-winning probabilities (so-called point based models), e.g. by Barnett & Clarke (2005).
    2. By finding θ_1 and θ_2 that are consistent with a model that predicts only the probability of winning the match. Klaassen & Magnus (2002) do this using player rankings; Kovalchik & Reid (2017) use Elo ratings together with Barnett & Clarke's model.
- In-play updating of θ_1 and θ_2 has been done by Barnett (2011) using an *ad hoc* method, and Kovalchik & Reid using an empirical Bayes approach.

Alternative: Paired comparison models on serve & return

- Another idea: Why not use a dynamic paired comparison model (Elo / Glicko) directly on serve and return?
- Give each player a serve rating S_i and return rating R_i. Then the Bradley-Terry likelihood (rescaled as usual for Elo) is:

$$P(\text{i wins serving to j}|S_i, R_j) = \frac{10^{(S_i - R_j)/400}}{1 + 10^{(S_i - R_j)/400}} \qquad (1)$$

- With two k factors – one for the server and one for the returner – we can calculate Elo as usual, and I present a version of Glicko, too.
- This has been proposed in several blogs online, but not evaluated as far as I could see.
- Key research questions: How well do these work, and what insights can they provide?

Data

- Jeff Sackmann made point-by-point data available on github
- Split data into two parts: training (2011-2014) and test (2015).
- Data is not particularly clean (duplicate names, exhibitions...). Corrected problems but probably still not perfect.

	train	test
Start date	2011-07-28	2015-01-05
End date	2014-11-15	2015-12-20
Matches	8,180	2,482
Points	1,281,672	400,739

Table 1: ATP

	train	test
Start date	2011-07-28	2015-01-04
End date	2014-11-19	2015-12-20
Matches	7,958	2,549
Points	1,118,680	368,198

Table 2: WTA

Serve and return Elo

- Think of each point as a "match" between server and returner.
- After each point, update server rating with a server k-factor, k_{serve}, and returner rating with returner k-factor, k_{return}.
- Three parameters: k_{serve}, k_{return} and return Elo mean (serve fixed at 1500).

	ATP	WTA
k_{serve}	1.725	1.579
k_{return}	1.254	1.207
Return mean	1414.9	1466.5

Figure 1: Optimal values for parameters, obtained by maximising the log predictive likelihood on the training set. k quite similar for ATP and WTA; mean is different ($\approx$ 62% win probability on ATP, 55% on WTA).

Introduction to Glicko

- Roughly speaking, Glicko is a Bayesian version of Elo devised by Mark Glickman (1999).
- Each player's skill is modelled as a normal distribution with a mean and a variance reflecting the uncertainty about the skill.
- Update sizes are determined by how uncertain we are about skill: more uncertainty $\rightarrow$ larger updates.
- Glicko is essentially a Kalman filter, alternating two steps: "measurement" (update based on measurement) and "prediction" (updating between measurements)

Glicko illustration: "Measurement Step"

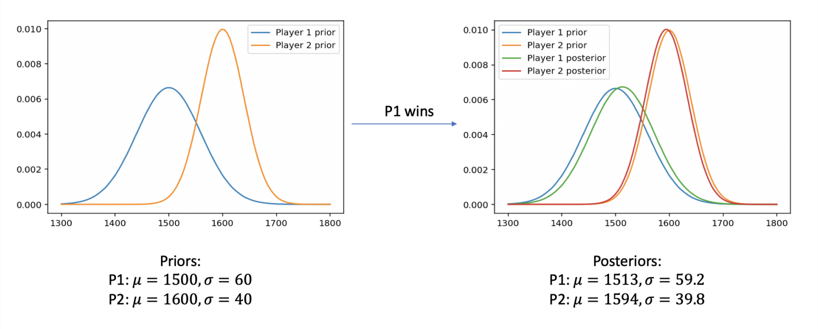


Figure 2: One Glicko "measurement step". Player 1 plays and wins, updating their mean and reducing their uncertainty. Their initial uncertainty was greater, so the update is larger.

Glicko illustration: "Prediction step"

- If we only made updates, the uncertainty would shrink, and updates would get smaller and smaller.
- The "prediction step" consists of adding in some variance to reflect our growing uncertainty. In the original Glicko paper (Glickman 1999), time is broken into periods (e.g. months) and variance is added after each period.
- (Potentially) neat feature of Glicko:
  long absence $\rightarrow$ large variance $\rightarrow$ large updates (while standard Elo has fixed k)

Serve and return Glicko idea

- Like with serve & return Elo, update each point.
- Glicko breaks time into periods (e.g. months), but unclear how to do in-play! Update point-by-point instead.
- Introduce sources of uncertainty (all to be optimised):
  - Within match, add variance after each point
  - Between matches, let variance grow linearly as function of days, with intercept
- Within-match and between-match variance can tell us about how much skills are expected to shift during and between matches.

Serve and return Glicko parameters

	ATP	WTA
Point to point stdev	9.18×10^{-4}	8.00×10^{-4}
Match to match stdev	3.85	3.25
Per day stdev	1.10	0.985
Prior serve stdev	34.81	36.90
Prior return mean	1415.7	1467.5
Prior return stdev	23.65	24.56

Table 3: Optimal values for serve and return Glicko.

- Point-to-point standard deviation is essentially zero, so no apparent benefit in allowing for it. (!)
- Considerable uncertainty between matches for both tours.
- Parameters similar between tours, except return mean (again).

Serve and return Glicko example vs Elo

- 2015 Wimbledon Final: Djokovic def. Federer, 7-6(1) 6-7(10) 6-4 6-3.

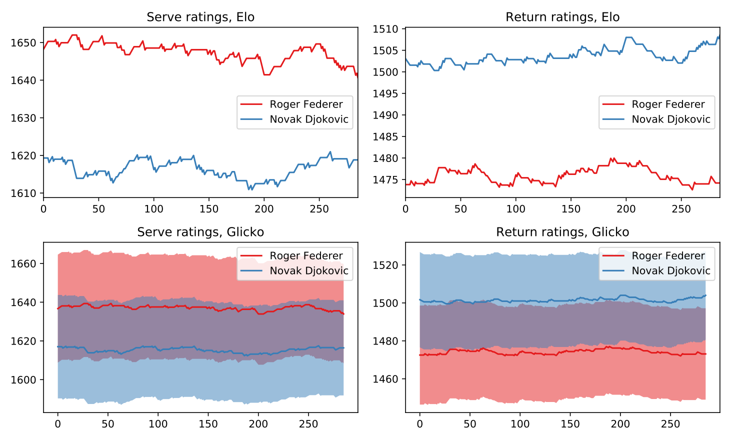


Figure 3: Elo vs. Glicko for Wimbledon final. Note Glicko error bars!

Baseline models for evaluation

- Pick two baseline models (and fit parameters using training data):
  1. Elo with static win probabilities
  2. Elo with beta binomial update.
- Model (2) is close to Kovalchik & Reid's model, but optimised slightly differently (log loss) and using a more naive model to set the sum of θ_1 and θ_2. It might therefore be a bit worse.

Evaluation

- Consider two settings for evaluation:
    1. Predicting the outcome of each point in test set.
    2. Predicting the outcome of the match, pre-match and in-play.
- For the second, consider three scenarios:
    1. Start of match (pre-match)
    2. After fifth game (in-play)
    3. After first set (in-play)
- Evaluate point-level with log loss; evaluate in-play using accuracy and log loss.
- Log loss is negative mean (Bernoulli) log likelihood:

$$\text{log loss} = \frac{1}{N}\sum_{i=1}^{N} -y_i \log p_i - (1 - y_i)\log(1 - p_i) \qquad (2)$$

- where p_i is the predicted probability and y_i is the outcome.

Point-level evaluation

	Log loss
S&R Glicko	**0.64788**
S&R Elo	0.64794
Elo+Beta	0.64819
Static Elo	0.64939

Table 4: ATP

	Log loss
S&R Glicko	**0.68286**
Elo+Beta	0.68287
S&R Elo	0.68298
Static Elo	0.68394

Table 5: WTA

- Results all seem very close, but important to remember we are predicting hundreds of thousands of points, so small differences can be important.
- Serve and return Glicko does best on both tours. Serve and return Elo does well too, but is beaten by Elo+Beta on WTA.

To sample or not to sample

- Both Serve & Return Glicko and the Beta binomial model produce posterior distributions for the serve-winning probabilities, not just point estimates
- Two options for predicting: (1) just use mean, (2) do the "proper" thing and integrate over distributions by Monte Carlo sampling:

$$P(\text{p1 win}|\text{score}) \approx \frac{1}{N}\sum_{n=1}^{N} P(\text{p1 win}|\text{score}, \theta_{1n}, \theta_{2n}) \tag{3}$$

  where θ_{1n} and θ_{2n} are draws from the distributions for the serve-winning probabilities.
- Using samples is more expensive (need to predict N times, e.g. $N = 1000$), but is it worth it?

Sampling vs. not sampling

	S&R Glicko	Elo+Beta
Initial	0.610	**0.580**
Initial MC	**0.596**	0.588
Fifth Game	0.540	0.529
Fifth Game MC	**0.533**	**0.528**
After Set	0.436	0.441
After Set MC	**0.433**	**0.434**

Table 6: Log loss ATP. MC denotes sampling.

	S&R Glicko	Elo+Beta
Initial	0.677	0.634
Initial MC	**0.653**	**0.630**
Fifth Game	0.593	0.579
Fifth Game MC	**0.580**	**0.563**
After Set	0.469	0.470
After Set MC	**0.463**	**0.451**

Table 7: Log loss WTA. MC denotes sampling.

- With the single exception of the initial probability for Elo+Beta on ATP, MC sampling improves log loss everywhere!
- Report with MC sampling from now on.

Evaluation (match level, ATP)

	Start	Game 5	After set
Static Elo	68.0	73.7	81.5
S&R Elo	67.5	73.2	81.7
S&R Glicko	**68.1**	73.3	81.7
Elo+Beta	68.0	**73.8**	81.7

Table 8: Accuracy

	Start	Game 5	After set
Static Elo	**0.580**	**0.524**	0.433
S&R Elo	0.613	0.540	0.435
S&R Glicko	0.596	0.533	0.433
Elo+Beta	0.588	0.528	0.434

Table 9: Log loss

- All models are fairly close on accuracy, with S&R Elo perhaps dropping a little short.
- S&R Elo is poor initially for log loss, and S&R Glicko is worse than Static Elo & Beta binomial.
- Surprisingly, Static Elo has (close to) best metrics throughout!

Evaluation (match level, WTA)

	Start	Game 5	After set
Elo+Beta	63.1	**71.4**	**81.6**
S&R Glicko	**64.1**	69.8	81.1
Static Elo	63.1	69.6	81.4
S&R Elo	63.0	69.5	81.0

Table 10: Accuracy

	Start	Game 5	After set
Elo+Beta	**0.630**	**0.563**	**0.451**
S&R Glicko	0.653	0.580	0.463
Static Elo	0.634	0.569	0.460
S&R Elo	0.686	0.600	0.473

Table 11: Log loss

- Apparently more gains for dynamic models on WTA. S&R Glicko has higher accuracy initially, but Elo + Beta binomial does best later.
- On log loss, Elo + Beta binomial does best, improving slightly on static Elo. S&R Glicko improves on S&R Elo, but lags behind.

One possible problem: Surface changes

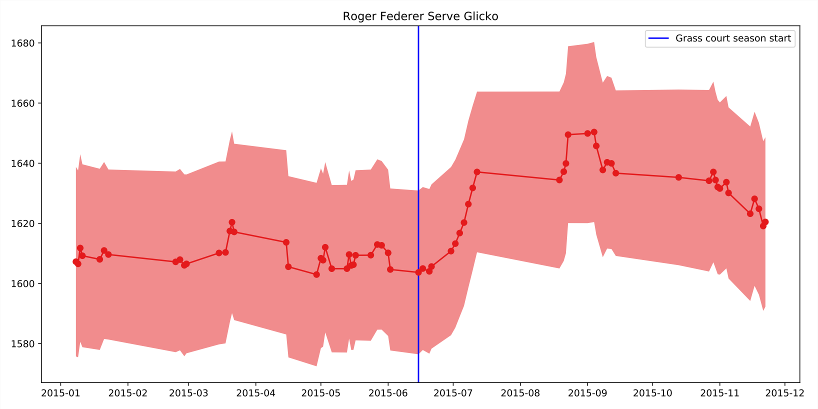


Figure 4: Serve Glicko knows nothing about surfaces. Federer's serve Glicko gets much better on grass, but part of this is likely just due to the surface, not improvement in skill.

Conclusions & Future work

- Serve & Return Glicko improves on Serve & Return Elo in all metrics.
- Serve & Return Elo and Glicko do well at predicting individual points, but worse at predicting match outcomes compared to Static Elo and Elo+Beta binomial. Problem with iid approximation?
- Static Elo (no within-match updating of θ_1 and θ_2) is a surprisingly strong baseline for within-match prediction.
- Using uncertainty when predicting appears to be beneficial and could perhaps also improve Kovalchik & Reid model if practical (need fast code for win prediction).
- Future work:
  - Accounting for surface changes may improve the model.
  - Perhaps optimise match win directly, not points?

Thanks!

Model 1: Elo with static serve probabilities

- You just heard all about Elo! It does well for tennis.
- First (simplest) model: Use Elo win probability to calculate θ_1, θ_2; keep constant throughout match, and update probabilities with iid conditional on score.
- This model has only one parameter: the k-factor.
- Optimise match log loss on training set to find $k = 40.00$ on ATP and 43.66 on WTA.

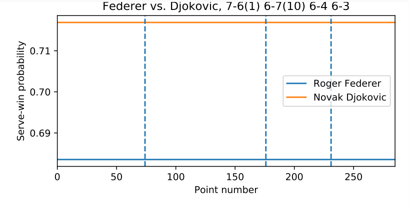


Figure 5: Elo with no updating

Model 2: Elo + Beta binomial

- Use the win probabilities θ_i as means for Beta priors
- Use parameterisation of Beta using mean μ and prior sample size ν.
- Start with $\theta_i \sim \mathrm{Beta}(\mu_i, \nu)$, setting μ_i to Elo probability as before.
- Update with match outcomes (treat player as biased coin).
- Need to optimise ν, the prior sample size. ATP: 71.1. WTA: 66.4. ADD IN PLAY PLOT.

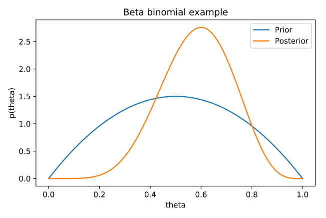
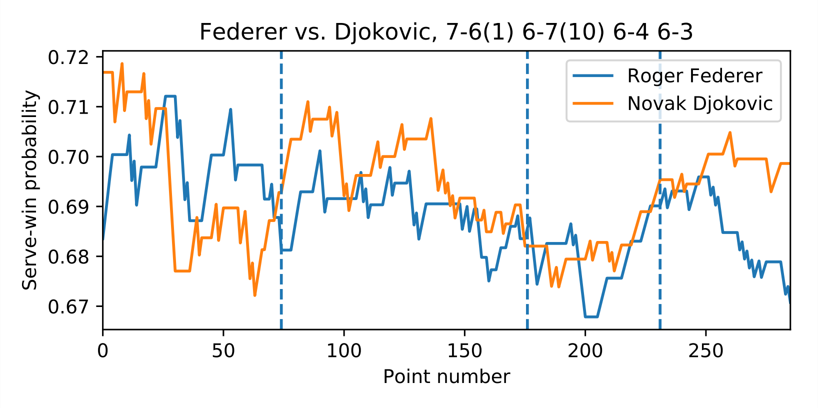


Figure 6: Prior: $\mu = 0.5$, $\nu = 4$; Posterior is result after observing 5 points won, 3 lost.

Beta binomial for Wimbledon final



Prior work: Use win prediction to get serve probabilities

- How to get serve win probabilities θ_1, θ_2 when we only have match win probability?
- Trick: *fix* $\theta_1 + \theta_2$; optimise $\theta_1 - \theta_2$ to match win prediction.
- Example (best of 3): $p(win) = 0.7$, $\theta_1 + \theta_2 = 1.26$ $\rightarrow \theta_1 = 0.651, \theta_2 = 0.609$

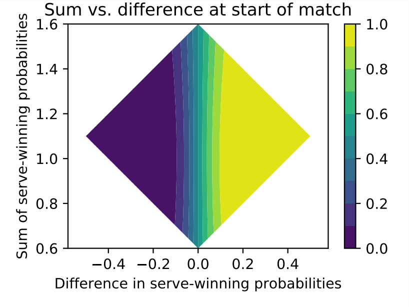


Figure 7: Sum vs. difference for win prediction at start. Lines are almost parallel, indicating difference is much more important.

How to use the iid model to update in-play expectations?

- We'll look at two model classes:
    1. Class 1: Use a win prediction model and find consistent serve-winning probabilities θ_1 and θ_2
    2. Class 2: Use a paired comparison model on points to directly update θ_1 and θ_2.

Running example: Federer vs. Djokovic, 2015 Wimbledon

- Djokovic beat Federer in 4 sets, 7-6(1) 6-7(10) 6-4 6-3.
- Federer won 66% of his points on serve; Djokovic won 69%.
- It's usual for the winner to have a higher percentage of points won on serve, but not always (!) – at the 2015 US Open e.g. Federer had higher percentage but lost in 4.

Serve and return Glicko parameters

TODO: Turn these into (more interpretable) standard deviations.

Also TODO: Illustrate what these mean.

	ATP	WTA
Point to point variance	8.43×10^{-7}	6.41×10^{-7}
Match to match variance	14.845	10.533
Per day variance	1.202	0.971
Prior serve variance	1211.67	1359.68
Prior return mean	1415.7	1467.5
Prior return variance	559.133	603.398

Table 12: Optimal values for serve and return Glicko

Serve and Return Elo example

- Djokovic beat Federer in the 2015 Wimbledon final, 7-6(1) 6-7(10) 6-4 6-3.

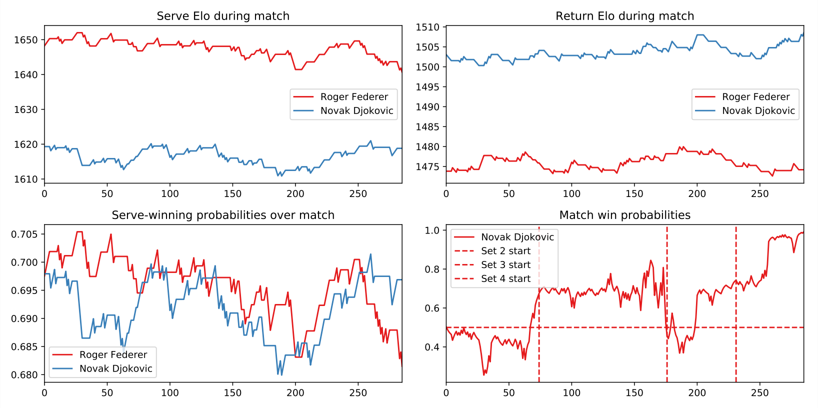


Figure 8: Serve and return Elo evolving over the match. Updates are fairly small, but θ_1 and θ_2 not constant.