SYSTEMS FOR RATING RATING SYSTEMS WITH RATING SYSTEMS

Clayton W. Thorrez
claytonthorrez@gmail.com

ABSTRACT

The field of rating systems is devoted to assigning numerical ratings to things based on data representing comparisons between those things, and using those ratings to create ordered rankings or predict future comparisons. For instance, Elo ratings are assigned to Chess players based the results of their games against other players, with higher ratings representing higher skill levels. However, when it comes to comparing rating systems to each other, we ironically defer to metrics such as "predictive accuracy" or "log-likelihood". In this work we aim to clear up this oversight and propose a novel framework for the comparison and evaluation of the rating systems, by the rating systems, and for the rating systems.

1 RATING SYSTEMS BACKGROUND

A rating system is a method used to assign numerical ratings to items representing in some sense their strength when compared to other items. A rating system is described as "online" or "dynamic" if it is meant to track the skills of competitors as they change over time, this is the setting considered in the present work. Much of the early development of this field was motivated by the desire to rate chess players in order to form world rankings and properly match players to other players of equivalent skill at tournaments. The Elo rating system (Elo & Sloan, 1978) was developed in the 1960's and remains in use in many applications to this day. Later, Glicko was developed as a Bayesian extension of Elo and applied to other competitions such as football games in the NFL. (Glickman, 1999) Subsequent work on online rating systems applied them to video games, extended to team competitions, and made various improvements to the update rules based on different statistical distributions, methods of approximation and optimization. (Herbrich et al., 2006; Weng & Lin, 2011)

1.1 RATING SYSTEMS EVALUATION

A variety of different methods have been used in order to compare rating systems with each other. The methods typically follow from the idea of making accurate predictions. This is commonly measured in terms of accuracy, log-likelihood or brier score on unseen data. Let n be the number of match-ups in the evaluation set, $\hat{p} \in (0, 1)$ be the predicted probability of the first competitor winning, $y \in [0, 1]$ be the true outcome with 1 representing a win for the first competitor and 0 representing a win for the second competitor, and $\hat{y} = 1[\hat{p_i} > 0.5]$ be the predicted outcome derived from the predicted probability.

$$\text{accuracy} = \frac{1}{n}\sum_{i=1}^{n} 1[y_i = \hat{y_i}] \tag{1}$$

$$\text{log-likelihood} = \frac{1}{n}\sum_{i=1}^{n} y_i * log(\hat{p_i}) + (1 - y_i) * log(1 - \hat{p_i}) \tag{2}$$

$$\text{brier score} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{p_i})^2 \tag{3}$$

(sorry but you have to include some equations if you want people to take you seriously)

This is sometimes computed over the same data used to fit the ratings in a setup where predictions are first made for a comparison or batch of comparisons, then those comparisons are "trained on" by updating the ratings for the competitors based on the results. This method is valid as the the predictions are not based on any data the model has trained on yet.

Like most machine learning methods, rating systems have plenty of hyperparameters which control things like how sensitive ratings are to new results, initial estimates of the rating means and variances, and numerical constants to calibrate predicted probabilities. If one does hyperparameter tuning (as one should if one wants to have good performance), then one should tune on only a subset of the data, and then evaluate on additional data that was not seen during the tuning and fitting phase. In online rating systems, the data is ordered and thus standard k-fold cross validation and random train-test splitting is not appropriate, instead we can only split based on time.

These evaluations, though useful, fall short of the full potential of rating systems research. When working in a field devoted to comparing things, it seems like a missed opportunity to not use your own methods when making your own comparisons. Glickman & Jones (1999) came tantalizingly close to with the title "Rating the Chess Rating System", unfortunately the paper did not actually use a rating system to perform the ratings. In this work we devise novel methods of evaluation for rating systems, using the rating systems themselves.

2 Understanding Rating Systems as Systems to be Rated

The main difficulty with using rating system to rate rating systems is casting the evaluation as a time dynamic problem. In the aforementioned evaluation methods, given a dataset and a rating system, you get some number as the metric. If you have two rating systems, you can get two numbers and compare them and see which one is better but that's kind of as far as you can go, you can't effectively fit a rating system on a dataset with only one comparison. So the problem becomes how to get multiple comparisons, and not just any comparisons, they need to represent the change in strength over time so just adding some random noise and calculating the numbers a lot of times doesn't really do anything. (we tried this) Understanding these constraints, we propose two systems for rating rating systems with rating systems. Both of these measure real properties of rating systems which to our knowledge have not been discussed at length in the prior literature and which are useful to understand in practical applications.

2.1 system 1

This idea was inspired by the common situation in machine learning where the experimenter tries many versions of their model with different hyperparameters and can sometimes achieve large gains by finding a particularly well-suited configuration. With large numbers of hyperparamters, the search space can be large and therefore it is expensive to do a fine-grained grid-search over it. In this case it is a desirable quality of a model to be relatively invariant to the hyperparameters, and to give strong performance "out of the box" or "off the shelf".

With this in mind, we formulate a setting where the time dimension is the number of iterations we spend sweeping over the hyperparameter space. We can compare the best metrics achieved so far for different models doing the sweeps. For an experimental set up with m candidate models and sweeping h hyperparameter configurations, we would have m choose 2 comparisons at each time step for a dataset of $h * \binom{m}{2}$ match-ups. In order to measure overall aptitude for hyperparameter tuning, this can be done over d different datasets and the results from all sweeps merged into a single sweep dataset. In practice we find it is useful to repeat the experiment k times with different random seeds used to sample from the hyperparameter space to reduce noise and stabilize the resulting ratings. (they can still be rather unstable though.)

2.2 System 2

The scenario uses the same time dimension as the original rating system, but makes explicit comparisons across models of the metrics computed for each time step individually. In this setting, $h * \binom{m}{2}$ match-ups are created for each time step in the original setup. This setup measures several important properties of online rating systems: warm up and saturation.

2.2.1 WARM UP

The warm up problem is the measure of how quickly the model incorporates information in order to start making good predictions. In an uninitialized state, all competitors will have equal ratings and thus $\forall i \ \hat{p_i} = 0.5$. A model is said to have poor warm up if it takes a long time and a lot of data before the ratings and predicted probabilities start to accurately model the data.

2.2.2 SATURATION

Saturation is the opposite problem, some rating systems have mechanisms by which the magnitude of updates for players shrike over time as they compete in more games. This has some benefits as it corresponds to our uncertainty about the player decreasing and can be seen as convergence. However for very large datasets over a long period of time, this can be a detriment as it is possible that the rating updates get too small for the model to be able to accurately change the true underlying changes in skill that occur.

3 EXPERIMENTS

We perform experiments using both of our proposed systems to rate four commonly used rating systems on real world datasets.

3.1 DATASETS

We use 3 real datasets spanning different time ranges, time scales, types of competitions, number of competitors and number of match-ups.

3.1.1 CHESS

We use a subset of the Chess data from the FIDE/Deloitte Chess Ratings Challenge. (Sonas, 2011) This is a dataset of millions of rated Chess games from "a recent 11 year period". Due to computational limitations (we ran these experiments on a laptop), we limit our experiments to 200,000 games involving 15,992 players over 35 one month long rating periods. The data is available on kaggle.

3.1.2 NBA

We also accessed a dataset of historical basketball games from the National Basketball Association. (Paine, 2015). This dataset has 126,314 games involving 104 team from 1946 to 2015 which we split up into 1 year rating periods. This data was also accessed from kaggle.

3.1.3 LEAGUE OF LEGENDS (LOL)

This dataset contains matches from professional League of Legends games played both at both in person and online events from all over the world including international tournaments. It contains data covering 121,160 matches involving 12,675 teams from 2011 until 2024 and split into 681 1 week rating periods. The data was collected from the Leaguepedia API.

3.2 RATING SYSTEMS

We compare 4 commonly used rating systems both to rate the players and teams in our datasets, as well as to rate each other.

3.2.1 ELO

This is the orignal one by the OG Arpad Elo himself. It's pretty simple but holds up considering how old it is and that it was only invented for Chess. Elo & Sloan, 1978

3.2.2 GLICKO

Glicko is Bayesian extension of Elo where each competitor has both a rating mean and a rating deviation to measure how uncertain we are about a competitor's rating. (Glickman, 1999)

3.2.3 TRUESKILL

This was developed by Microsoft in order to facilitate fair and balanced matchmaking for online Halo games. The main advantage is that it supports team games (which we are not evaluating in this work) and that it is based on a Gaussian skill distribution rather than the logistic distribution which is used in Elo and Glicko.

3.2.4 WENG-LIN THURSTONE-MOSTELLER

This one is super similar to TrueSkill (supports team play and is Gaussian based) but has some minor differences in how the player variances are updated. If you care that much go read the paper. (Weng & Lin, 2011) Honestly I'm including it because I have an implementation of it and 2 by 2 plots look nicer than 3 by 1.

3.3 SOFTWARE

The code and data for these experiments is available at https://github.com/cthorrez/rs4rs. (but good luck reading it lol) For the rating system implementations we use the open source python package **riix** (which I also wrote and maintain go check it out this is shameless self-promotion)

4 OK THE ACTUAL EXPERIMENTS NOW

4.1 SYSTEM 1

The methodology we used for this was to split each dataset 80/20 based on date for train and test. For each rating system, we defined hyperparameter spaces by low and high values for each parameter. We used the default values or values reported as the best in the experiments in their respective papers as a starting point and defined the ranges as one order of magnitude larger and smaller to simulate the situation where we truly have minimal knowledge of what reasonable values are. Since different methods have different numbers of hyperparameters, we employ uniform sampling over the entire legal space and use the same number of samples for each method. We used 100 samples, repeated over 5 random seed, for each of the 3 datasets, and each of the 4 rating systems creating a dataset of $100 * 5 * 3 * \binom{4}{2} = 9000$ match-up over 100 rating periods. The outcome for each match-up is determined by whether the accuracy on the test set for that specific dataset/hyperparameter/rating system is higher than the rating system it is compared against.

Once that dataset was created we do another grid search hyperparameter sweep over it and plot the evolution of the ratings over time for each rating system. The grid search range is the same as over the original datasets and are listed in the appendix.

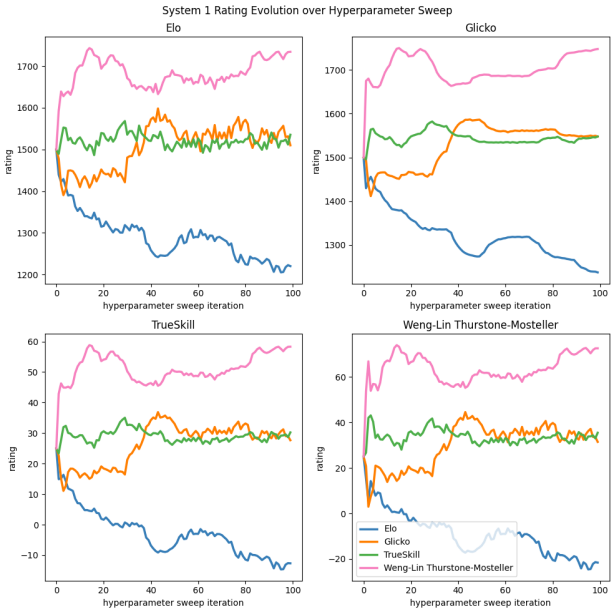
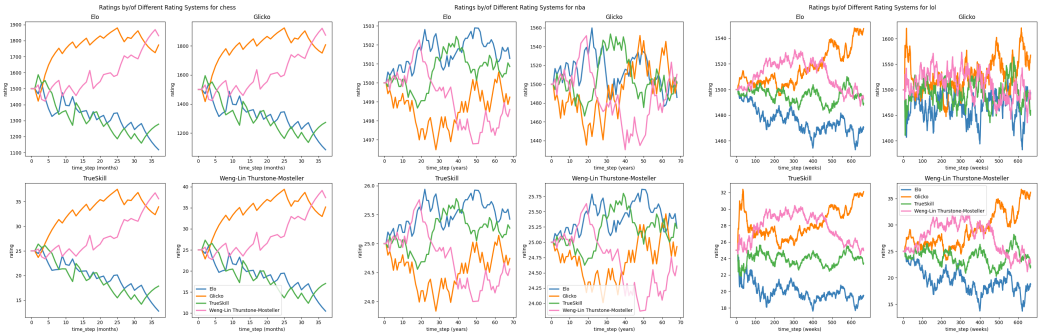


Figure 1: The ratings produced by each of the rating systems, on the rating data representing the evolution of relative performance during the hyperparameter tuning process.

We learn several things from 1. One is that relative to other models Weng-Lin Thurstone-Mosteller performs the strongest throughout the entire hyperparameter tuning process, and that Elo performs the worst. Additionally Elo loses performance through the process which means that it benefits the least from parameter tuning. While the ratings from all methods broadly agree, it is also interesting to note that the Glicko ratings are noticeably smoother than the others.

4.2 System 2

For this system, we do not pool the datasets since the time dimension of the constructed dataset is the same as for the original, and the different datasets do now have the same time scales. Instead for each dataset we initialize rating systems with the best hyperparameters found in the sweeps during the System 1 experiments and fit them on the datasets logging the pre-match predictions and real outcomes. Then we do another pass and compute the predictive accuracy for matches only within a single rating period. Finally, for each combination of rating systems, we define the outcome of this synthetic match based on which has the higher accuracy for that time step.



(a) Chess (b) NBA (c) LoL

Figure 2: The ratings produced by each of the rating systems, on the rating data representing the evolution of relative predictive strength as they are trained on more data.

We learn some interesting things from the system 2 graphs as well. To begin, all of the ratings systems produce nearly identical (in shape, though different in scale) results for chess, this may be due to chess having the most matches and the fewest rating periods potentially leading to more convergence during each rating period. It is interesting to that Glicko and Weng-Lin beat Elo and TrueSkill so thoroughly, and that the margin increases as more data is trained on. This could indicate that TrueSkill has saturated. (It's actually mechanistically impossible for Elo to saturate so idk what is going on there it is possible that Elo is just weaker overall). Also it's possible we are seeing Glicko finally saturate at the end as it dips slightly and Weng-Lin overtakes.

The NBA data is just super noisy. This is due to many factors, the most obvious being that it is data over a much longer period of time, it is a team game so the total variance is larger due to the variance contributed by each player, and there are substantial changes to the competitors (teams) themselves, both within and across seasons as players change teams. We are fairly hesitant to conclude anything there, though if we were to cautiously suggest something to check more later it could be the idea that different eras of basketball had different features which made them more well suited as prediction tasks for different rating systems but that is kind of a stretch.

For League of Legends we see some interesting patterns:

1. Weng-Lin is the strongest roughly in the first half before Glicko overtakes
2. Weng-Lin strongly saturates and performance degrades to even lower than TrueSkill
3. Unlike in 1, Glicko produces the most noisy ratings here

The overall findings are that different rating systems are optimal for different types of competitions, and even for different stages of a competition.

5 Future Work (Stuff I ran out of time to do)

One factor we neglected in this analysis is runtime. For hyperparameter sweeps we allowed each method the same number of runs even though some methods are more expensive than others. Another interesting dimension could be to use wall-clock time as the time dimension to scale the results taking into account total compute required by each method.

The other obvious thing is to run this stuff on more datasets and more rating systems but yeah I'm just out of time.

6 Closing Remarks

We identified a hole in the evaluation method currently used in rating systems research and found a way to fill that hole with even more rating systems.

Author Contributions

It's all me baby

REFERENCES

Arpad E Elo and Sam Sloan. The rating of chessplayers: Past and present. 1978.

Mark E Glickman. Parameter estimation in large dynamic paired comparison experiments. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 48(3):377–394, 1999.

Mark E Glickman and Albyn C Jones. Rating the chess rating system. *CHANCE-BERLIN THEN NEW YORK-*, 12:21–28, 1999.

Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*, 19, 2006.

Neil Paine. How we calculate nba elo ratings, May 2015. URL `https://fivethirtyeight.com/features/how-we-calculate-nba-elo-ratings/`.

Jeff Sonas. The deloitte/fide chess rating challenge, Jun 2011. URL `https://en.chessbase.com/post/sonas-the-deloitte-fide-che-rating-challenge`.

Ruby C Weng and Chih-Jen Lin. A bayesian approximation method for online ranking. *Journal of Machine Learning Research*, 12(1), 2011.

A APPENDIX

A.1 HYPERPARAMETER SWEEP RANGES

For each hyperparameter the two numbers are the low and high value for sampling.

```
ELO_PARAM_RANGES = {
    'k' : (0.32, 320)
}

GLICKO_PARAM_RANGES = {
    'initial_rating_dev': (35, 3500),
    'c': (6.3, 630.0),
}

TRUESKILL_PARAM_RANGES = {
    'initial_sigma': (0.833, 83.33),
    'beta':  (0.4166, 41.66),
    'tau': (0.00833, 0.833)
}

WL_TM_PARAM_RANGES = {
    'initial_sigma': (0.833, 83.33),
    'beta':  (0.4166, 41.66),
    'kappa': (0.00001, 0.001),
    'tau': (0.00833, 0.833),
}
```