PERFORMANCE RATING ALGORITHMS

Alex Bullard

Adviser: Daniel Scharstein

A Thesis

Presented to the Faculty of the Computer Science Department

of Middlebury College

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Arts

May 2011

ABSTRACT

Competition surrounds us, and if we wish to understand the progress of winners and losers while predicting the results of competition it is necessary to understand performance rating. This thesis presents an overview of the goals and challenges of rating systems and the approaches used to address them. It identifies and describes three distinct classes of rating algorithm and provides explanations of example algorithms of each class. The thesis ends with overview of the various methods of comparing performance rating algorithms.

ACKNOWLEDGEMENTS

I would like to thank Daniel Scharstein for advising me on this project, the computer science department, my friends for letting me bore them about rating algorithms and my family for being my family

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION TO PERFORMANCE RATING

Humans surround themselves with competition. These competitions are diverse as one-on-one games like chess, go, and table-tennis, team sports like soccer and baseball, single-player games like bowling and golf, computer games, horse races, and Guinness World Records. But wherever we find these competitions the questions people ask are always the same :

- Who is the best in the world?
- Where am I on that list?
- Who will win the championship match?

Performance ranking and rating answers these questions through use of various rating and ranking methods and algorithms.

Ranking is the process of describing the skill of a player in relation to that player's competitors. A player is assigned a number along an ordinal scale (see Section 2.2.2) that informs that player where they stand. With a ranking we can say Xiaoxia Li is the best Woman's Table Tennis player in the world because she is ranked number 1 and that Rashidat Ogundele is the worst of the professional players because she is rated 1274 out of 1274 players [16]. This means that we can answer the first two questions posed above. However we do not have a method of estimating the skill difference between the two players and although we might assume that Li would dominate Ogundele in a match we don't have any way of actually evaluating Ogundele's chances. Its possible that the field of Woman's Table Tennis is so extraordinarily close in skill that the last place player regularly beats the first place player.

To solve this problem and to aide our ability to predict the results of games competitive systems often institute performance rating. Performance rating arranges players

along an interval scale (Section 2.2.3) representative of that player's skill. With a rating we can now declare that Xiaoxia Li has 2847 "rating points" (see Table 1.1) to Ogundele's 237 and that difference of 2610 points means that it is extremely unlikely that Li would lose a match between them.

Table 1.1: Woman's table tennis world rankings/ratings [16]

Name	Ranking	Rating
Xiaoxia Li	1	2837
Yan Gou	2	2805
...	...	...
Tanya Guttersberger	1273	349
Rashidat Ogundele	1274	237

An important point to note is that given a rating we can develop a ranking that accurately reflects the current state of the game. However we cannot construct a rating from a ranking, which makes performance rating systems much more powerful than ranking systems. A rating also allows us to answer a more specific version of the second question posed above. Instead of wondering what position a player holds in a competitive system that player can now ask, "What is the skill difference between me and first place?."

The methods used to generate ratings/rankings range from the simple to the extremely complex. In this thesis we explore three distinct categories of methods for developing ratings both in general and with specific examples. Chapter 2 provides an overview of the concepts and considerations central to rating systems. Chapter 3 concerns traditional methods used to rate performance while Chapters 4 and 5 explore algorithmic approaches to the problem.

1.1 Rating Applications

Rating systems have applications in every competitive activity. They provide the means for comparison between competitors and create a basis with which to predict the out-

comes of matches. Ratings systems extend the possibilities of winning to allow for players to win beyond the confines of the game itself. Instead of simply beating individual opponents, players gain the ability to become the champion of their game. Performance ratings are used in a wide variety of sports organizations, news agencies and games to organize their players into leagues, ladders and top-ten lists. Whenever a paired comparison between two competitors takes place some form of rating system is in effect. Ratings appear in Major League Baseball, where teams are compared by their records during the season and performance in the playoffs, in professional golf where tournaments determine rankings, and in the computer game StarCraft 2 in which a players position in their league is affected by every game they play. There are as many applications for rating systems as there are competitive systems to apply them to.

The importance of exploring rating as a scientific topic is most evidenced by the large number of organizations that exist solely to satisfy the public's desire for rating information. The NBA, NFL, FIDE and FIFA are all tasked with the organization of competitions in their respective sports, but their true purpose could simply be viewed as the official decider of who the best teams/individuals in their organizations are. After all they pick the league participants, seed the tournaments, publish rankings, and ultimately award the trophies. Without that responsibility such organizations would have little influence in their sports. This suggests that rating systems are one of the more important aspects of a competitive activity, and it is easy to see why. Ratings create a dynamic system with champions, underdogs and exciting matches. The entire story of a sport can be told through the rise and fall of player's ratings. For example, Figure 1.1 is a graph of the ratings of chess grandmasters from 1950 to 1990. The fluctuations of players ratings lead to interesting events like Bobby Fischer's dominating performance in 1972 and his subsequent decline. Such information would not be available without some form of performance rating.

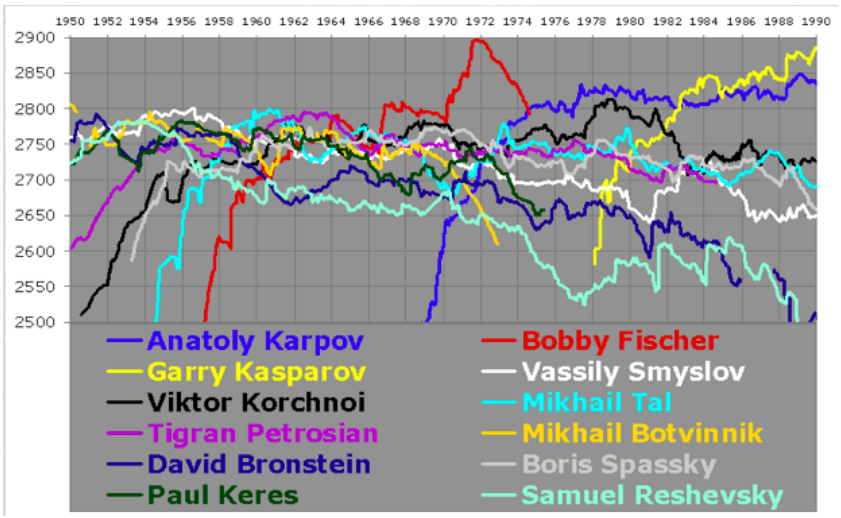

Figure 1.1: Chessmetrics rating of chess grandmasters from 1950 to 1990 [18]

But rating systems are not only used for inflating the egos of players and fans. They are also essential for predicting the outcomes of games. This outcome is commonly used in matchmaking systems, where two players likely to draw against each other are selected as ideal opponents. The comparison between the rankings of two competitors can provide a wealth of information including which player is higher rated, their historical performance, potential gap in skill, etc. Predicting who will win a game depends on all of these indicators and as ratings become more accurate so do the predictions.

Nor must we restrict the use of rating algorithms to sports and games. Though they provide a wealth of statistics and real-world applications, the utility of rating algorithms extends far beyond the confines of such structured competition. Rating algorithms have been implemented in areas as diverse as psychology [5], biology [22] and electrical engineering [24].

1.2 Deloitte/FIDE Chess Rating Challenge

Chess could very well be the perfect game to use for understanding and developing rating algorithms. With a huge following around the world and multiple rating organi-

zations with thousands of members, chess has the unique ability to provide hundreds of thousands of real-world game results. All chess players that play in organization-sponsored events are rated on the same scale and in the same system as professional players. The chess rating of a player is that player's defining characteristic, and sufficiently high ratings are always required in order to achieve the status of chess master or grandmaster [12]. For this reason chess organizations are continually interested in learning and developing rating methods that behave fairly towards all participants.

In such spirit, the Deliotte/FIDE Rating Challenge (CRC) that began in February 2011 provides access to 2,418,212 chess games played over 135 months by 54,205 players [21]. Hosted by the statistics competition website Kaggle and organized by Jeff Sonas (the author of the rating algorithm Chessmetrics described in Section 5.6.1) this competition challenges participants to predict the outcomes of 100,000 games played during the three months following the last game from the dataset.189 teams entered the contest, demonstrating there exists a strong interest in rating algorithms in the statistics community.

As there are two sponsors, there are two prizes, each with different requirements for winning that prize. The Deliotte prize, $10,000, is rewarded to the most accurate algorithm submitted during the 12 week time-line. The FIDE prize, a trip to Athens and a presentation to a FIDE board, is rewarded to the most accurate algorithm that also meets the criteria of a "practical chess rating system." The restrictions for such a system are extensive and specific and a generalization of this definition is presented below [21]. The idea of a "practical" system is a powerful and useful one that indicates that a system is much more likely to be used in real-world applications.

A completed algorithm is entered into the competition through the submission of a CSV file containing the algorithms predictions for the outcomes of the 100,000 test games. While the competition is in progress algorithm accuracy is judged with the

binomial deviance between predicted game outcomes and actual outcomes of the first 25% of the 100,000 test games. (See Section 7.1.1 for details on binomial deviance calculation.) This allows participants to get a sense of where they belong in the rankings while protecting the system against overfitting. At the conclusion of the contest the previously submitted files are compared to the results from the remaining 75,000 test games and the algorithm with the lowest binomial deviance is judged the most accurate [21].

This competition was the inspiration for this thesis and so over the course of the contest I entered a number of algorithms. These algorithms are detailed in Chapter 6.

1.3 A Practical Chess Rating System

At the most basic level FIDE's definition of a practical chess rating system indicates a desire for a system in which ratings can be calculated quickly without significant overhead. According to the definition posted in the chess competition rules, a practical system must:

- Track no more than 10 numbers for each player in the system
- Perform no more than two iterations over the dataset when updating ratings, i.e. algorithms may not employ methods that attempt to converge to optimal ratings
- Initialize the system with ratings provided
- Utilize only the ratings of the players, system constants, month the game was played and the knowledge of which player is white when generating the expected result of a game

These items are only a small subset of the restrictive rules imposed by FIDE. Although they might be criticized as overly restrictive, (Sonas claims that FIDE's current implementation of the Elo rating system does not meet their own specifications [21]) this

definition will prove useful to us as we explore various methods of rating. A practical chess rating system that performs well is sure to be of interest in areas beyond chess, and challenging computer scientists to develop a better one is definitely beneficial to the state of the art.

CHAPTER 2

THE BASICS OF RATING

This chapter summarizes some of the aspects, challenges and features of the typical rating system. Though every rating system approaches the problem differently many share common methods and histories, the knowledge of which helps to better understand the benefits and costs of specific approaches.

A basic rating system must incorporate a rating scale with which to rate players, a method for generating initial ratings for players, a method for comparing those ratings and predicting match results and a method for updating ratings after a match has been played.

2.1 Goals in Performance Rating

The goals of a rating systems are threefold. A system should be:

- Accurate
- Fair
- Practical

Accuracy is of paramount concern because it is through accuracy that a rating system finds itself useful. A system that was both fair and practical would still be worth little if the ratings produced did not reflect the actual skills of the players involved. Accuracy has shades, however, and in some cases organizations may consciously sacrifice on accuracy to increase the fairness or practicality of the system. Methods for measuring the accuracy of a system are discussed in Section 7.1.

The fairness of a rating system refers to the notion that a system should treat each player equally. For systems in which this is not the case, participation may wane as

players determine that they are not ranked correctly or equally. The fairness of a rating system is by far its most subjective trait. What seems fair to one participant may appear biased to another. Therefore clarity of an algorithm is closely related to its perceived fairness. Algorithms that are easy to understand are more likely to be accepted by competitors so they are more likely to be used in the real-world. This is one of the goals of FIDE's rules for a practical chess rating system because acceptance of a new algorithm must be strong before they can risk changing the way they rate players.

Practicality is the the final requirement for a system and the most dynamic. Depending on what application the system is intended for, different constraints might be placed on the system's computational complexity, space required, time for execution, number of games played, etc. However in general the faster an algorithm can complete the better the chances the algorithm will find real-world uses. The space and time requirements for an algorithm can be found using standard techniques for determining complexity.

2.1.1 Challenges in Performance Rating

In striving towards these goals rating systems may experience a number of challenges including human bias, game bias, and scale. In some cases generating a rating that is accurate and fair for all players and all circumstances may not even be possible.

Human bias is present in some form in probably every rating system, leading to a decrease in performance [17]. We can attempt to reduce human influence by relying on mathematics as much as possible, but the fact remains that if a human designed the system some assumptions were made in creation. This bias is most easily recognizable in traditional rating systems because one of the defining aspects of such systems is the human influence involved.

Bias in the game itself can also present a challenge to address. In many competitive games and sports opponents do not always face equal odds when competing against each

other. Factors like home-field advantage mean that in a game with two participants of equal skill level the participant holding the advantage is more likely to win. In chess an advantage exists that is so significant that chess is not considered a ”fair” game [25]. When players of significant skill level play a match, the first player to move obtains an ”advantage that persists into the endgame.” This imbalance is typically referred to as the first-move or white advantage. Its effects are so noticeable that some chess experts have expressed the opinion that while White should play to win, Black must play to draw [25]. Consequently rating systems that attempt to predict the outcomes of games must be modified to model this asymmetry. Although tournaments are usually organized so that players play both sides equally, we can imagine a scenario in which a player plays only black. In an unadjusted system this player finds that her rating rises too little when she wins and falls too much when she loses. Since the system is not taking the difficulty of winning as black into account, it does not reward those wins more than it rewards wins as white. Implementing a first-mover advantage is also extremely important if the goal is to predict the outcome of games. Without information about which player has the advantage, the system is likely to produce incorrect results.

Another barrier to developing an accurate rating method for games like chess and go is the extremely large number of games played around the world. FIDE alone records the results of hundreds of thousands of chess games annually, each one of which must influence player ratings. Tournaments and polls, with their high overhead and poor results, could not suffice for ranking anyone but top players. A successful system must be minimally reliant on human interaction or the workload will approach the impossible for both players and organizers. At the same time, even some approaches taken by computers can be quickly overrun if attempted on too large of a dataset. For example, my own experimentation revealed that using neural networks to predict games results like in [38] does not scale to systems as large as the Deloitte/FIDE database. My implementation

ran for hours without converging to suitable results.

In addition to the difficulties inherent to the systems above, a number of other complications and questions must be considered in the development of a rating system. How many games must be played in order to produce accurate ratings? How much weight should be given to previous performance vs. current performance? Should performance in adverse playing conditions be weighted against performance in normal conditions? How individual rating systems choose to address these sorts questions is what separates them from each other, and various methods for doing so are discussed in the next few sections.

2.2 Scale Types

The first step in developing a rating/ranking method is to determine the scale type to be used. The four types of scale, nominal, ordinal, interval, and ratio, provide the means to arrange a set of data with increasing complexity and information. A description of measurement scales was first published in *Science* in 1946 by S. Stevens and a similar description featured prominently in Arpad Elo's book on rating "The Rating of Chess Players, Past and Present." An algorithm's choice of scale determines what information it provides and how that information is portrayed. As has already been mentioned, a ranking method will implement an ordinal scale while a rating method uses an interval scale. All four scales are discussed here, however, as they build off each other, and understanding the abilities and limitations of each is useful when thinking about the rating methods that follow in the next chapters.

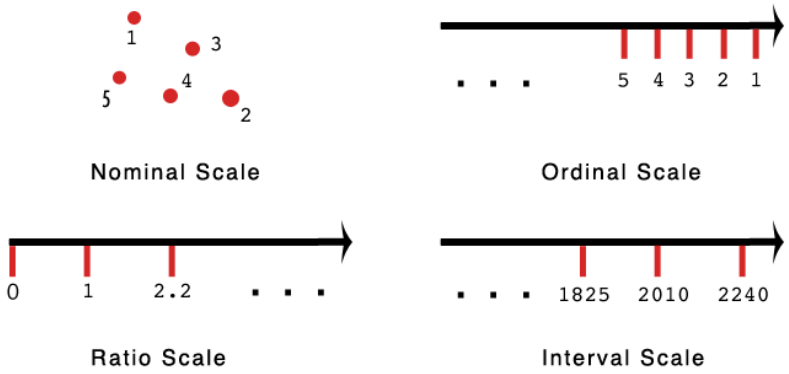

Figure 2.1: An illustration of the four types of scale

2.2.1 The Nominal Scale

The most basic scale type, the nominal scale, allows data to be partitioned into different sets but provides no hierarchical structure between those sets. Therefore while we can make a ”determination of equality” when comparing two elements by comparing the sets they belong to, it is impossible to compare elements across the boundaries of the sets [40]. An excellent example of the nominal scale is the numbers on the jerseys of a sports team. Though they differentiate the players the numbers provide no information about the value of a specific player to a team. The absence of ordering ability makes nominal scales essentially useless for rating or ranking the abilities of competitors, but nominal scales still appear in some competitions in the form of constructs like leagues or divisions. Since competitors often do not compete across such boundaries, and divisions, being nominal, are not ranked, such systems can present unique challenges for algorithms that attempt to rate every member of a system.

2.2.2 The Ordinal Scale

In an ordinal scale, elements are partitioned into sets that can be compared using less than and greater than operations [40]. Therefore ordinal scales provide the scale of

measurement for performance ranking. Participants in a competition can be ranked in order of performance by putting them in a set with a value greater than the sets of their weaker opponents but less than their stronger opponents. Since only relative position calculations can be performed on these sets, we cannot provide any information about the skill difference between players. In other words the ordinal results from a tournament cannot reveal whether the first place player performed only slightly better than her opponent or if she completely dominated the field. Some methods of ranking using ordinal scales are discussed in Chapter 3.

2.2.3 The Interval Scale

Interval scales allow for "determination of equality of intervals or differences" by partitioning elements in sets whose values lie along an abstract scale with no bounds [40]. Such a system allows for nominal ordering while providing additional information about the intervals between elements. This makes it extremely useful for rating systems because it is able to express the difference in skill level between two opponents instead of simply providing an ordering [3]. Such distinction is necessary to accurately predict the outcomes of events, fairly compare people that have never competed, and correctly organize tournaments. However it is important to note that interval scales have no physical basis. The values of the sets on an interval scale can be shifted along that scale without affecting the relationship between the sets or any meaning assigned to their values. This means that the values used in rating systems that employ interval scales are ultimately meaningless. Only the intervals between player ratings are significant.

2.2.4 The Ratio Scale

Elements in sets arranged according to a ratio scale are ordered based on a comparison to some basic measurement [40]. The value of a set in a ratio scale represents the number of that basic measurement that equals the value of the set. Many real-world definitions use the ratio scale including measurements of length (the meter, the lightyear) and mass (the kilogram) so as to provide a physical basis for that measurement. Such a system allows for the ”determination of equality of ratios” which in turn provides all of the comparison benefits of an interval scale. However, unlike the interval scale, ratio scales provide a firm definition of absolute zero. This is important because it means that ratio scales are concrete in the world in a way that interval scales are not. The implementation of a ratio scale in a performance rating context would prove extremely useful as it might allow comparison of skill across the boundaries of individual systems. However determining the basic level of skill on which to base a rating system may be completely impossible. There is no physical manifestation of skill to compare competitors to and attempting to create an abstract measure leads to a number of questions. What does it mean for a player to have zero skill? Is skill an additive property? Are two Michael Jordan's of skill twice as good as one Michael Jordan? For this reason performance rating systems can not use ratio scales for actual ratings.

2.3 Rating Algorithm Considerations

Every rating algorithm must perform a few basic tasks. It must rate every player above a certain experience threshold, have some method for generating updated ratings and provide some scale or reference with which its ratings can be understood. This section presents the general considerations that rating algorithms must address and some of the typical approaches used to do so. Designing a rating system is a process full of trade-offs

that affect aspects of the system as diverse as accuracy, functionality and practicality. Algorithms and systems that can clearly, consistently and accurately organize players while remaining open about their faults are extremely valuable, and those perform these functions are widely used. The goal of this section is to introduce various practices and considerations so that they can be recognized in the specific rating implementations explored in following chapters.

2.3.1 Generating Initial Ratings

When applying a rating system to a new activity some method of generating initial ratings for use by the rating algorithm must be employed. Choosing an acceptably accurate method for generating these initial ratings is essentially a form of meta-rating that requires rating players that do not yet have ratings. Methods for doing so include propagating provided ratings, assigning default ratings and random ratings.

Propagating Ratings

If the organization to be rated has a historical data set that includes initial ratings that covers a large enough subset of the players (as in the Deloitte/FIDE Chess Rating Challenge) the algorithm can use those ratings as a seed for rating the unrated players. As long every player is connected through some sequence of games to player seeded with an initial rating, a rating for every player can be produced [29]. An unrated player is assumed to have the same rating as their opponent and the player's score is updated after the results of the match are revealed. Implementations of Elo rating systems (Section 4.2.2) often use this approach for initial rating generation by running the algorithm over game data multiple times, retaining only rating information after every pass. This system is useful for obtaining an accurate rating system at the introduction of the system. However, the accuracy of the initial ratings produced depends completely on the size

and accuracy of the initial rating set. The larger and more accurate the system, the better the results produced.

Default Ratings

Alternatively, systems may choose to use provided rating information and simply assign unrated players a rating value based on perceived strength, tournament outcome or any other chosen metric. This method is useful in systems in which many games are played since ratings will quickly adjust to realistic values and little to no computation is required in generating them. However if a large number of players are seeded with such values there exists a period of time in which ratings and result prediction are very unreliable. The Harkness system discussed in Section 4.2.1 is an example of a system in which ratings are typically seeded with static initial values [23].

Random Ratings

The key difference between using random ratings and the default rating approach is that with default ratings an effort is made to make the assigned initial ratings as close to accurate as possible. Random ratings can be used in systems (like Chessmetrics in Section 5.6.1) in which the initial ratings have no influence on the final ratings produced by the program. Such systems are perhaps the most versatile since they do not need any amount of rating information for seeding but typically require that the algorithm iterate over the dataset multiple times while generating ratings, which can lead to slowdowns in performance [18].

2.3.2 Static vs. Incremental Rating

The distinction between a static and incremental rating system is a very important one. In a static system, once a rating or ranking has been computed, that value cannot be

updated independently. To generate a new rating the entire algorithm must be re-run because static systems do not model change in player ability over time [33]. Tournaments are the most common type of static rating system. In a system that uses tournament rankings the only time rankings are updated is after a new tournament has taken place.

Incremental rating systems allow results gathered after the initial formation of the rating system to modify the original system without the need to entirely recalculate. One of the requirements of a practical chess rating system is that it be incremental. This makes computing player ratings more efficient and possible to do after every game. Therefore incremental systems are commonly used in applications like computer games where players expect their ratings to change immediately upon completion of a match. Static ratings, which encompass the majority of both traditional rating systems and machine learning systems are better suited for use in historical rating in which computation time is not a factor and new results are not constantly added.

2.3.3 Modeling Player Performance Over Time

If a performance rating system uses an incremental approach for rating, the system must take into account the gradual improvement or decline of its members when generating ratings that use historical data or reflect a large period of time [29]. It is not enough to generate a rating for each player once, but those ratings must be updated to reflect a players current skill. For this reason, incremental rating systems must include some equation with which to update a player's rating based on their performance during that rating period. To do so a number of methods may be employed. Ratings may be updated after any interval of time or games. In systems that implement a small ratings period, ratings respond quickly to changes in player skill but may but are not as stable and require additional computation that may make the system infeasible. FIDE uses a rating period of two months, which makes sense for a physical activity with organized rating

events [12]. StarCraft 2 on the other hand updates ratings after every game played, a system made necessary by the fact that games are played extremely frequently and can be as short as 10 minutes.

A unique approach taken by the Edo system treats each player each year as a completely different person with a completely new rating. Since major improvement tends to take place over a longer time period, the Elo system is able to employ static rating techniques while modeling player change [34]. The disadvantage of such a system is the long time periods between generating new ratings. For this reason Edo is generally used to rate historical systems instead of ongoing ones.

2.3.4 Updating Ratings

Rating algorithms face a number of choices and challenges specific to updating the ratings of competitors. These include inflation, correctly weighting historical games and negative incentives for players to play. Solutions to these problems are more open-ended than some others in rating and therefore this is one of the areas in which the differences between rating algorithms start to show.

The Timeline Problem

One of the most important decisions an algorithm must make is how it will treat historical games. Rating algorithms must strike a constant balance between assigning inaccurate low ratings because of a player's past and promoting players too soon based on fluke events [19]. They also must acknowledge that without up-to-date information the rating they attribute to a player may no longer be accurate. This is such a problem with systems like Elo (Section 4.2.2) that do not adjust the influence of a match based on when it was played, that when attempting to get the most accurate ratings possible implementors will sometimes throw out huge chucks of historical data [41]. Some other

solutions typically used with more success are to weight games based on when they were played or to allow for more dramatic rating changes for those returning to the game after an absence from the system. This weight can be as simple as a linear equation like the one used by Chessmetrics [18] or as complex as the Bayesian inference method used by TrueSkill to determine rating certainty [32]. In any case methods that consider the consequences of historical games preform more accurately (see Section 7.4).

Rating Inflation

Rating inflation occurs in systems that use interval scales to rate their participants. Since there is no upper bound to the score achievable, it is possible for the ratings of the highest-ranked players to continue to rise, which, in some cases, can cause the ratings of those below them to begin to rise as well. There is no immediate disadvantage to rating inflation as it affects everyone, but it is undesirable because it makes ratings comparisons across inflation periods difficult. A common example comes from chess ratings which have shown increased inflation over the years [35]. To adjust for inflation, systems can take steps like shifting the ratings of the entire body of players down by a constant amount. Doing so does not affect effective score but does keep ratings from getting out of control. Other systems like Chessmetrics may build immunity to inflation into their system by constantly recalculating ratings from scratch in every rating period [18].

Negative Incentives for Competing

In some systems players may be discouraged to play since doing so is almost sure to decrease their rating [29]. This happens typically with new players when a few lucky wins leads the system to believe that the player is more skilled than she actually is. Fixing such a system involves a difficult trade-off. More caution regarding increasing the scores of new players prevents this problem but may annoy new players that find they

cannot quickly rise to their actual skill level. One solution used by the computer game StarCraft 2 is to require players participate in a rating period, (in StarCraft, 5 games) during which the player receives no rating. Only after the period is completed is the player assigned a rating. This allows the system to get a better handle on the actual skill of the player and potentially provide a better situation for all involved.

2.3.5 Rating Input

Every rating system uses competition results as an indicator of player skill and modifies a players ranking accordingly. However any number of other variables could also be used in such calculations. Like adjusting rankings over time, systems can attempt to use other information that might be used to modify result prediction, for example home-field advantage, presence of a winning/losing streak, preference for playing a specific position, perceived value of the event, etc. Tracking a new variable may increase the accuracy of the system, or, if no correlation exists or modeling it is difficult, negatively effect results. Adding new variables is non-trivial in the algorithms discussed in Chapter 4 because the algorithm must be completely rewritten to use it. Conversely, machine learning approaches like those in Chapter 5 can be provided with such information and chose independently whether it is valuable in prediction. In such cases it is still important that the algorithm designer carefully evaluate results, however, because adding too much extraneous data can lead to overfitting. An algorithm striving to meet FIDE's practicality definition must also be careful about what input it uses in calculations. FIDE allows a maximum of only ten values to be tracked per player and has specific requirements for data that can be used when updating ratings.

2.3.6 Representing Game Outcomes

A rating algorithm must choose some form with which to represent a player's performance in an event. In a game with three possible outcomes, like those covered in this thesis, systems almost unilaterally employ a simple method in which game outcome is represented as a number distributed between 0.0 and 1.0, with 0.0 representing a loss, 0.5 representing a tie, and 1.0 a win [3]. This allows a system to make accurate predictions about the expected outcome of a game on any range between 0.0 and 1.0. Such values no longer correspond directly to the outcome of the game. They instead represent the expected score of the game :

$$E_A = \frac{\sum_{i=0}^{n} S_{A_i}}{n} \tag{2.1}$$

Where n is a large number of games and S_{A_i} is the score of game i, or equivalently :

$$E_A = P(W_A) + .5P(T_A) \tag{2.2}$$

in which $P(W_A)$ is the probability that Player A will win and $P(T_A)$ is the probability she will tie. For example a player with a 25% chance of winning and a 50% chance of drawing will be assigned the score 0.5. Someone with an expected score of 0.75 might have a 75% chance of winning with 0% chance of drawing, a 50% chance of winning with a 50% chance of drawing, etc. Equation 2.2 can be calculated in advance by the system so it is typically used to generated the expected value.

CHAPTER 3

TRADITIONAL RATING SYSTEMS

In the context of this thesis, traditional rating algorithms refer to systems that don't attempt to model or predict the results of a match between two players but instead use various organizational structures to determine player rankings. This category includes tournaments and polls, discussed here, as well as any other structure in which humans have a defining role in producing ratings.

3.1 Ranking Polls

The rankings poll, a method commonly used by sports reporters and some sports organizations is particularly noted for its inaccuracy. In this system a group of people that may consist of 'experts', players, or the public, is asked for their predictions on an upcoming event or series of events. These opinions are then amalgamated into a single list arranged by the number of votes/points each competitor received. An example of such a system is shown in Figure 3.1 in which the polls of two separate news organizations are compared. Polls like these hold a special place in college football and a large amount of faith is placed in the rankings, but a cursory examination of the top 15 teams as chosen by the Associated Press reveals that six of the top fifteen picks are ranked differently than they are in a poll on the same event by USA Today. Clearly this system does not lead to reliable rankings and yet in some organizations, like the Bowl Championship Series in NCAA Division 1 football, the teams that get to participate in bowl games are partially determined by these opinions [6]. In separate papers Rick Wilson of Oklahoma State University and James Keener of the University of Utah both reach the conclusion that "intuition is not a good guide to determining a rating system" and that therefore polls provide a poor method of ranking a system [17] [38]. Polls are an excellent example of human bias, one of the most prevalent ranking system problems. However in

AP Top 25

RK	TEAM	RECORD	PTS
1	Auburn (56)	14-0	1472
2	TCU (3)	13-0	1392
3	Oregon	12-1	1379
4	Stanford	12-1	1300
5	Ohio State	12-1	1220
6	Oklahoma	12-2	1108
7	Wisconsin	11-2	1055
8	LSU	11-2	1051
9	Boise State	12-1	1031
10	Alabama	10-3	961
11	Nevada	13-1	866
12	Arkansas	10-3	863
13	Oklahoma State	11-2	833
14	Michigan State	11-2	696
15	Mississippi State	9-4	578

USA Today Poll

RK	TEAM	RECORD	PTS
1	Auburn (56)	14-0	1424
2	TCU (1)	13-0	1336
3	Oregon	12-1	1333
4	Stanford	12-1	1254
5	Ohio State	12-1	1197
6	Oklahoma	12-2	1096
7	Boise State	12-1	1012
8	LSU	11-2	1007
8	Wisconsin	11-2	1007
10	Oklahoma State	11-2	883
11	Alabama	10-3	860
12	Arkansas	10-3	818
13	Nevada	13-1	734
14	Michigan State	11-2	676
15	Virginia Tech	11-3	636

Figure 3.1: Comparison of two football college rankings based on polls by the Associated Press and USA Today [11]

some situations, like college football, they may be unavoidable. Since regional separations dictate that many teams will never play each other outside of bowl games, it is impossible to use a different rating system based only on game results since the pool of teams is not connected.

3.2 Tournaments

A technique commonly used to rank competitors is the use of tournament results. Tournaments are ubiquitous in professional and amateur sports as the final decider of the best team or individual. Used in everything from baseball to soccer to the Olympics, tournaments are popular because they force competitors to directly face each other resulting in ranking information that derives directly from the match-up. There are in general two types of tournaments, elimination and round robin, both of which exhibit promising qualities but fail to provide fair and practical methods for rating.

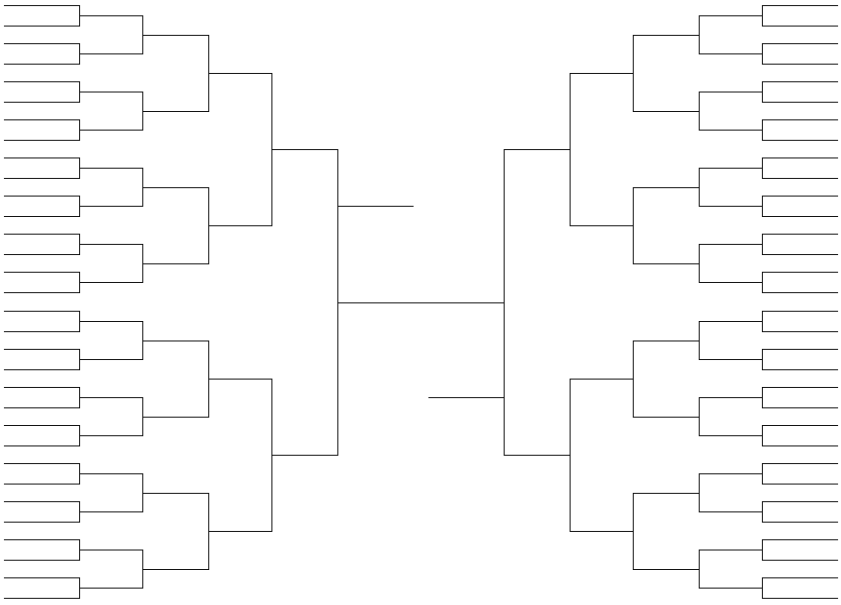

Figure 3.2: Visualization of elimination bracket

3.2.1 Elimination Tournaments

In elimination tournaments (Figure 3.2) competitors are arranged in brackets in which each matched pair of players plays one or more games in order to determine who advances to the next round. To determine a winner only $n-1$ games must be played, a very small number. Organizers pay the price for such efficiency however as unfortunately the structure of an elimination tournament leads to inconclusive results regarding the rankings of players eliminated in the lower brackets. Two players that compete in the second round and are eliminated by different opponents cannot make definitive statements about their rankings with regards to one another. While we obtain a ranking for the first and second players, the best we can typically say about those lower is given by statements like ”top 16”, ”top 32” etc. The structure of an elimination tournament also leads to inherent unfairness. Imagine the second best player and the best player in a tournament meet in the first round. One of them will be eliminated and therefore ranked in the bottom of the tournament, but if they had not faced each other right away both might have found themselves in the finals. This sort of problem is often addressed with concepts

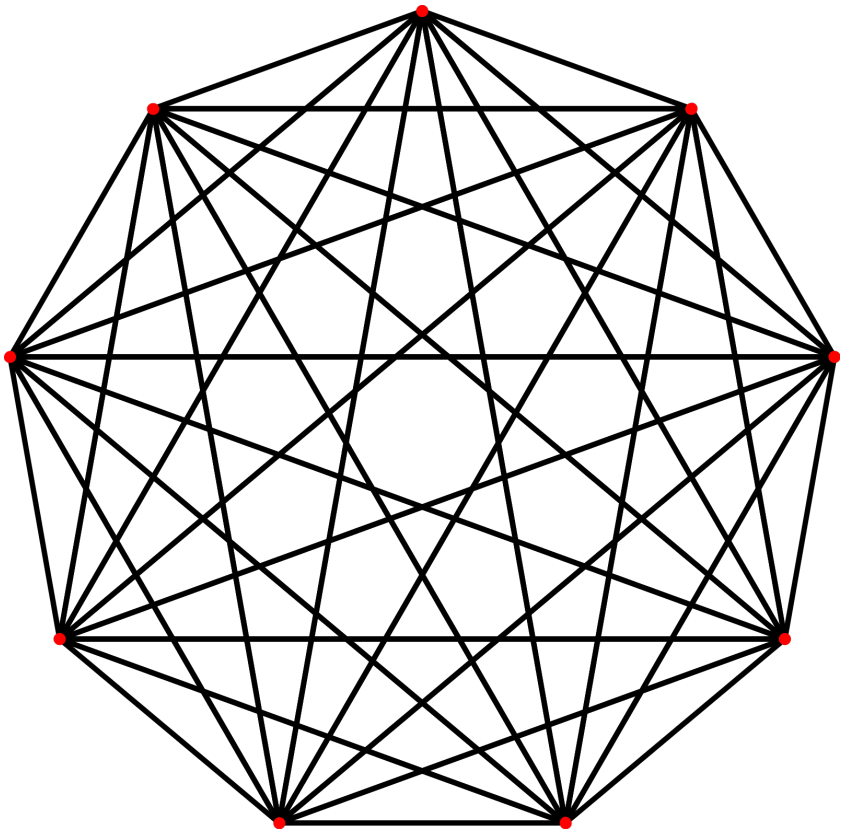

Figure 3.3: Visualization of a round robin tournament in which nodes are players and edges are games played

like ”seeding” players into higher positions and providing double or triple elimination structure, but the inherent problems remain although they might be obscured.

3.2.2 Round Robin Tournaments

To address some of the problems of an elimination tournament a round robin tournament (Figure 3.3) is organized so that every player plays every other player. This allows for a better ranking of players because we can now evaluate the performance of a player relative to every other player, but it still retains problems. If player A defeats player B but loses to player C while player B beats player C which player is better, A or C? The answer is not clear and it's not even clear within the confines of the tournament

Official World Golf Ranking

Week 17 2011

Week Ending 24 April 2011

World Ranking

This Week	Last Week	End 2010	Name	Country	Average Points	Total Points	Events Played (Divisor) Min-40 Max-56	Points Lost in 2011	Points Gained in 2011	Events Played (Actual)
1	(2)	<1>	Lee Westwood	England	**7.653**	359.704	47	-115.329	50.080	47
2	(1)	<3>	Martin Kaymer	Germany	**7.524**	376.209	50	-98.390	118.797	50
3	(3)	<9>	Luke Donald	England	**7.375**	383.494	52	-78.551	151.477	52
4	(4)	<4>	Phil Mickelson	United States	**6.520**	293.398	45	-100.831	106.053	45
5	(5)	<6>	Graeme McDowell	N Ireland	**5.840**	321.193	55	-72.463	60.164	55
6	(6)	<2>	Tiger Woods	United States	**5.715**	228.613	40	-129.442	42.801	36
7	(7)	<10>	Rory McIlroy	N Ireland	**5.641**	298.982	53	-86.991	78.201	53
8	(8)	<8>	Paul Casey	England	**5.590**	245.972	44	-82.119	62.768	44
9	(9)	<7>	Steve Stricker	United States	**5.478**	224.585	41	-95.151	56.942	41
10	(10)	<13>	Matt Kuchar	United States	**5.212**	276.223	53	-64.985	94.137	53

Figure 3.4: Current official world golf ranking generated using a tournament point system showing the top 10 golfers

how we might address such problems. We could require that players play series longer than one game, but even if just one game is played large player bases make round robin tournaments completely impractical. They require a minimum of $n(n-1)/2$ games played, a number that rises quadratically with n [14]. Additionally, tournaments games traditionally represent only a small portion of games played in a sport or game. Using tournaments as the only ranking system in an organization risks alienation of more casual players, rankings that do not update as quickly as might be desired and exclusion of those that do not participate in tournaments.

3.2.3 Multi-Tournament Point Systems

In professional golf, the Official World Golf Ranking (Figure 3.4) is determined by points earned through performance in tournaments. Since golf is in essence an individual sport its tournaments do not suffer as much from the problems outlined above. Player scores can be directly compared to each other to determine an overall ranking for every participant like a round robin tournaments without the performance problems. Unfortunately the desire to rank tournament participants outside the boundaries of individual tournaments leads to an unfair rating system. The official ranking points table of

the International Federation of PGA Tours [31] uses human determined minimum point values for tournaments instead of basing the points solely on the abilities of the players that compete. That means that the points gained by players do not accurately reflect the skill required to gain those points. Additionally in such a system it can be difficult to determine how points should be lost, which means that player's ratings can stagnate at higher levels than they should.

3.3 Summary

The methods discussed in this chapter are not very good systems for generating accurate, fair and practical ratings. However they see wide implementation as the deciding ranking factor for most if not all major sports. This probably occurs because they are publicly perceived as fair, which makes sense since all results are based on real-world performance. However if the goal is to produce a player rating that actually reflects the state of the system, it's necessary to approach the rating problem a different way. This is especially true if the implementation requires a non-static rating method in which ratings can be quickly updated. Traditional rating methods required a large amount of overhead and organization to update player ratings, something that may not always be available and is frequently undesirable.

CHAPTER 4

PRACTICAL RATING ALGORITHMS

This chapter presents the influences and methods of a number of rating algorithms that would be considered practical by FIDE and therefore do not contain the organizational problems of traditional rating methods. In general such algorithms employ methods developed by Arpad Elo to rate players based solely on their performance in games against other rated opponents. Theses ratings are placed on an interval scale without need for machine learning techniques or multiple iterations over a dataset of game results. Eschewing custom models for each player they instead use constructs like distribution curves to model the variability of player abilities. The variables used in their calculations are therefore essentially restricted to player ranks, the date of the contest and system constants exactly as required by the definition of a practical rating algorithm. For this reason these algorithms are generally easy to understand and compute, but can suffer somewhat in accuracy. The most famous and widely used of these algorithms is the Elo rating system detailed in Section 4.2.2.

4.1 Distribution Curves

As mentioned in Section 2.3.3, competitors in games, as they are human, do not perform at a consistent skill level. They experience winning streaks and losing streaks, beat players better than them and suffer fluke defeats to players well below their skill level. This uncertainty is what makes competition exciting, but it is also a large factor in the difficulty of modeling an accurate rating system. Such uncertainty cannot be modeled by directly comparing the ratings of two players because such a system will always predict victory for the higher ranked player. Some sort of probability model is required to produce results that more accurately represent the inherent randomness.

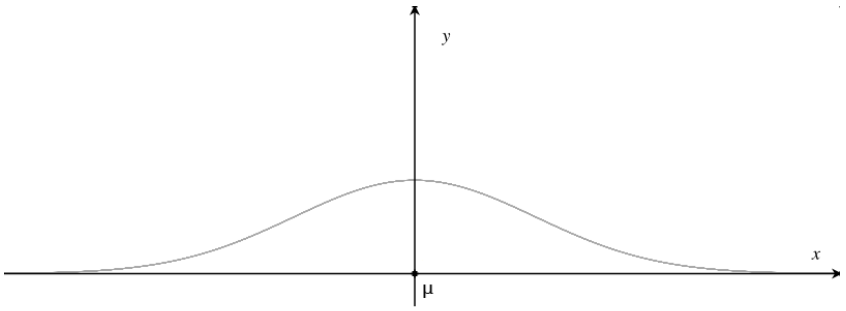

Figure 4.1: A graph of the normal distribution, one model used to represent variable player skill level

A widely used solution to this problem was first proposed by Arpad Elo in his development of the Elo rating system **??** discussed in Section 4.2.2. Elo suggested that a players performance in a specific game be represented using a random variable indicating the level of skill presented by that player. By using a distribution curve to model the likelihood of a player performing at various skill levels, the system is able to produce the probability that one player beats another. This method increases the amount of information that can be expressed by a game prediction. Instead of restricting the prediction of Player A's performance to a win, loss, or draw (1.0, 0.0, 0.5) the system can produce an expected value for the outcome of a match, any number between 0.0 and 1.0.

In such a system a player's skill level is no longer represented by just a number but by the entire distribution that defines their range of skill. The rating of a player is typically stored as simply the mean of their probability distribution. Therefore the choice of probability distribution with which to model player performance is extremely important. Once a function has been selected, however, the values of the constants that adjust features like the global distribution's variance are not as important. Since such systems use an interval scale by necessity, the meaning of the scale can be adjusted to allow any version of a distribution function to work equally well.

The original distribution used by Elo was the normal distribution defined by the

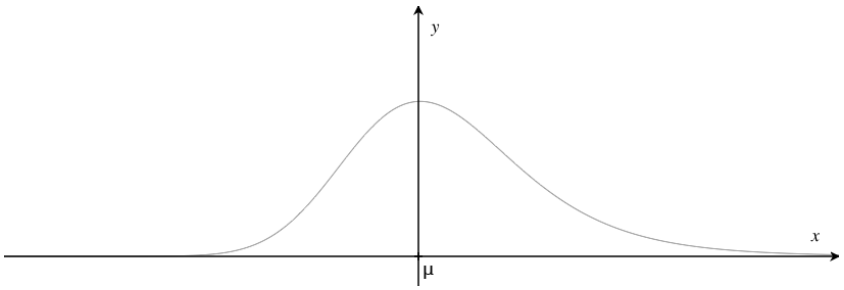

Figure 4.2: A graph of the logistic distribution a common model used to represent variable player skill level in chess

probability density function

$$\phi(x, \mu) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2} x^2} \tag{4.1}$$

This distribution, pictured in Figure 4.1, is characterized by its symmetry with respect to the line $x = \mu$ [13]. Since μ is defined as the mean of the distribution and is therefore also the player's rating score, this system predicts that a player will play above and below their rated level with equal probability. This distribution is perhaps the most intuitive model for a player's performance. In statistics the normal distribution is used to describe the probability of random variables clustering around a single value, and since a player rating is a single value it seems like it should work well. However experimentation has shown that at least in chess, the normal distribution tends to predict worse performance for lower ranked players than actually occurs [42]. For that reason many applications use the logistic distribution in Figure 4.2 defined with the cumulative distribution function :

$$F(x, \mu, s) = \frac{1}{1 + e^{-(x-\mu)/s}} \tag{4.2}$$

In which x is a random variable, μ is the mean and s is a constant proportional to the standard deviation [13]. As evidenced by the figure, the normal distribution is very similar to the logistic one. However, the logistic distribution has a heavier tail, meaning the distribution is more biased towards higher values.

It is possible, even probable, that different probability distributions work better for different activities. However since chess is a leader in the world of ratings, and the

logistic distribution works best for it, systems that use it are more common than those that don't.

4.1.1 The Bradley-Terry Model for Paired Comparisons

The Bradley-Terry model is method of determining the expected outcome of a competition between two competitors [14]. It provides the basic theory behind the implementation of many rating algorithms and is necessary to provide the basis for predicting the outcome of games [37]. In its most basic form it states :

$$P(\textit{Player i beats Player j}) = \frac{R_i}{R_i + R_j} \tag{4.3}$$

Where R_i and R_j are the respective skill of players i and j. This equation does not consider draws or white advantage [37] but can be modified to do so. Though a requirement for every rating system that directly compares the ratings of its players, the Bradley-Terry model is more recognizable in some methods than in others. While it all but disappears in Elo, it forms the kernel of ratings comparisons in the Whole History Rating System (Section 5.6.2).

4.2 Implementations

The algorithms in this section all rate players along an interval scale with ratings generated in a manner that conforms to FIDE's definition of a practical chess rating system. To that end they do not iterate over the dataset multiple times, use any sort of machine learning or track a large amount of information about each player. Both Elo and Glinko use distribution curves to model expected player performance, while the Harkness System, a precursor to Elo, relies on a more static interpretation of player skill.

4.2.1 The Harkness System

An ancestor of the Elo system, the Harkness System was developed in 1950s by Kenneth Harkness and implemented in the United States Chess Federation (USCF) from 1950 to 1960 as their rating algorithm until it was replaced by Elo [23]. The algorithm is extremely simplistic, based solely on the winning percentage of players. Player A's rating after an event is given by:

$$R_A = R_{avg} - 1000 * (0.5 - \frac{\sum_{i=0}^{G_{num}} S_{A_i}}{G_{num}}) \tag{4.4}$$

where S_{A_i} is Player A's score in game i and G_{num} is the number of games in the event [23]. The important things to notice with this algorithm are that a player's prior ranking does not factor into the ranking assigned after the event and that player variability is not modeled. These oversights result in a ranking that performs quite poorly in real-world testing (see Chapter 7). However its simplicity of design and ease of implementation makes Harkness a potentially good choice for small rating applications in which accuracy is not extremely important.

4.2.2 The Elo Rating System

Arpad E. Elo developed the Elo rating system in 1959 to address the need of USCF for an algorithm that could more accurately rate its chess players in terms of skill. The proposed system was highly successful and was adopted by the USCF in 1960 and the World Chess Federation (FIDE) in 1970 [3] . Although modern implementations often modify the original equations, the Elo algorithm remains largely intact and in use today, a testament to its accuracy and ease of use. The algorithm itself has found success far beyond the boundaries of chess; as Elo wrote in the book he later published on his rating system, "Although the system ... described was originally developed for rating chess players, its application is by no means limited to chess" [3]. Scrabble, multiple sports

organizations and many video games all use variations of the Elo system to develop their ratings.

Theory

The Elo rating system uses the logistic distribution when evaluating player skill. As a player improves, the mean of her distribution increases, representing the fact that she is now more likely to win against better players and less likely to lose to worse ones [9]. The average player rating used by the system is completely arbitrary: one of the more famous ranking organizations, the USCF, originally chose 1500 as the average player rank, with a difference of 400 ranking points indicating a 90% advantage for the higher ranked player. This was chosen to superficially resemble the ratings of the Harkness System. Since then it has become the norm for Elo implementations which in turn inspired the rating scales of subsequent algorithms. Some influence of Elo's approach to rating can be found in every modern rating system, even those that use machine learning to model the system.

One of the Elo system's greatest advantages is the simplicity of its implementation. For every game played the system executes three steps. They are:

- Generate the expected score of the game
- Compare the expected score to the actual outcome
- Adjust the ratings of the participants based on that comparison

The expected score for each player, E, follows the definition detailed in Section 2.3.6. E is calculated using both player's current skill ratings and the cumulative distribution function of the chosen distribution. The expected score equation for Player A using a logistic distribution and the FIDE characteristics described above is

$$E_a = \frac{1}{1 + 10^{(R_b - R_a)/400}} \tag{4.5}$$

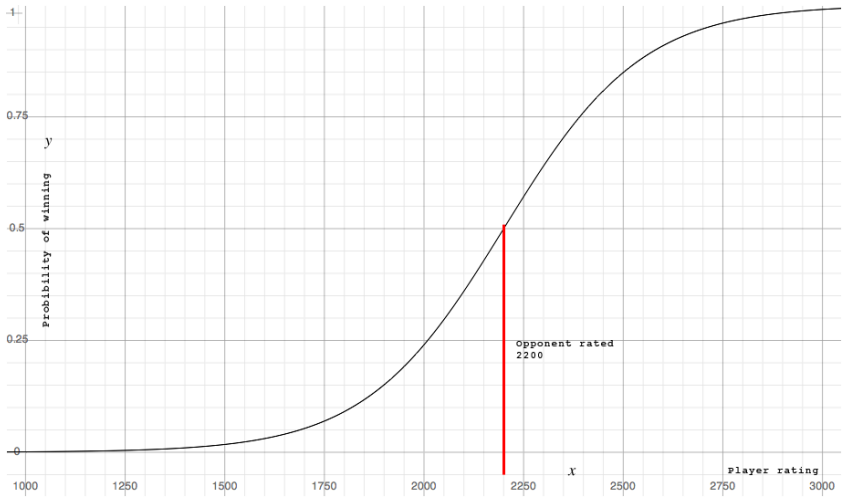

Figure 4.3: A graph of the expected score of a game in which $R_b = 2200$ and R_A is given by the x-axis

where R_a and R_b are the players 'true' skill levels. Since true skill is impossible to quantitatively determine (and is always changing) each player's estimated skill level is used instead. Notice that this equation is essentially the same as the logistic cumulative distribution function in Equation 4.2 with the constants changed. The 400 point difference rule is applied by changing the base to 10 and s to 400. In the case that the players differ in rating by 400 points this yields $1/(1 + 10^{-1}) = .9090...$, which is ten times the expected score of the other player, $0.09090....$ Also observe that:

$$E_a + E_b = 1 \tag{4.6}$$

Therefore E_B can be found simply with:

$$E_B = 1 - E_A \tag{4.7}$$

which improves the speed with which expected scores for each player can be calculated, an important feature in a system attempting to rate over millions of games. Figure 4.3 shows a graph of such expected scores.

Once a match has been played, the expected scores of the players are compared to the actual score of the match (denoted by S_A and S_B) and each player's rating is adjusted

to reflect the new information gathered. A player that performed better than expected will have their rating raised. One that performed worse will obtain a lower rating as a result. The equation for doing so is:

$$R'_a = R_a + k(S_a - E_a) \tag{4.8}$$

where k is the maximum possible point adjustment per game, often referred to as the k-factor. Picking a good k-factor is therefore extremely important to the accuracy of the rating system. If k is too high and too much emphasis is placed on recent games and each win or loss bounces the player up and down through the rankings. if k is too low and not enough emphasis is placed on recent games; player rankings stagnate because no one can gain enough points to advance. In chess, a k-factor of 24 has been advocated as the most accurate [19], but most major chess organizations have chosen to adopt a tiered approach in which players that pass certain ratings thresholds are assigned new k-factors. For example, FIDE uses a k-factor of 25 for new players with under 30 games, 15 for players rated under 2400 and 10 after a player breaks 2400.[12] This allows the FIDE players to quickly rise or fall to an appropriate skill level when they begin playing in the Federation, but prevents too much tumultuous movement at the top. Unfortunately this leads to a negative incentive for competing as discussed in Section 2.3.4. Since a player's initial k-factor is so high, if they win the majority of their initial games they may find themselves rated too highly and face the options of playing more games leading to a lower rating or keeping their artificially high rating by not playing.

The theory behind Elo is the basis for all modern rating methods. The simple idea of using a probability distribution to model a player's skill is repeated in the Glinko Rating System [28] (Section 4.2.3), Elo++ [41] (Section 5.6.3), TrueSkill [36] and Chessmetrics [18] (Section 5.6.1). Although these algorithms all modify Elo's ideas in a different way, his equations are usually still visible in some form.

4.2.3 The Glicko Rating System

The Glicko Rating System [28] is the most complex of the rating systems covered in this chapter. Developed in 1995 by Mark Glickman, the system uses an Elo approach modified to include a ”ratings deviation” (RD) that measures the system’s confidence in a player’s rating. This attempts to resolve Glickman’s primary concern with Elo, namely that by ignoring the number of recent games a player has participated in, Elo allows too much variation for players that compete often and too little for those that have not recently played [28]. In his introduction to the Glicko Rating System, Glickman gives the following example as a justification:

> Suppose two players, both rated 1700, played a tournament game with the first player defeating the second. Under the US Chess Federation’s version of the Elo system, the first player would gain 16 rating points and the second player would lose 16 points. But suppose that the first player had just returned to tournament play after many years, while the second player plays every weekend. In this situation, the first player’s rating of 1700 is not a very reliable measure of his strength, while the second player’s rating of 1700 is much more trustworthy. My intuition tells me that (1) the first player’s rating should increase by a large amount (more than 16 points) because his rating of 1700 is not believable in the first place, and that defeating a player with a fairly precise rating of 1700 is reasonable evidence that his strength is probably much higher than 1700, and (2) the second player’s rating should decrease by a small amount (less than 16 points) because his rating is already precisely measured to be near 1700, and that he loses to a player whose rating cannot be trusted, so that very little information about his own playing strength has been learned. [28]

The effect incorporating a ratings deviation into the Elo system is an indirect method of modeling the challenge from section 2.3.3 concerning player ability over time. Although Glicko does not assign weights to games based on when they were played, his method is able to differentiate between players whose ratings are likely to be accurate and those whose ratings may lie within a much larger range. In doing so Glickman makes the assumption that in-game improvement is made gradually and linearly over time and can therefore be modeled by Elo. However if a player removes themselves from a rating system while continuing to improve in skill a simple Elo system is unable to quickly and accurately generate a new rating for that player when they re-enter the system. Glicko uses the ratings deviation assigned to each player as a means of modifying the standard deviation of that player's ratings distribution independently.

Theory

At the beginning of each rating period, a new player is assigned the rating of 1500 and a ratings deviation (RD) of 350, which indicates the system is at the maximum level of uncertainty regarding the player's rating. If the player was rated in previous events, then their RD is calculated with the equation:

$$RD'_A = min(\sqrt{RD_A^2 + c^2 t}, 350) \tag{4.9}$$

where c is the uncertainty constant that determines the rate the system becomes uncertain in a player's rating and t is the number of rating periods since the player last participated. A higher RD indicates greater uncertainty, and as the periods between the current rating period and the last one in which the player competed increases that uncertainty grows. A player that participated in the most recent rating period uses the value $t = 1$, which means that the value of RD'_A will be only slightly higher than RD_A. The existence of this equation is very important as it is one of the requirements for a practical rating system that the system can be seeded with ratings from another system. After the initial

RD for each player has been determined, Player A's rating is calculated.

$$R'_A = R_A + \frac{q}{1/RD^2 + 1/d^2} \sum_{i=1}^{n} g(RD_i)(S_{A_i} - E(R_A, R_i, RD_i)) \tag{4.10}$$

where q is n is the number of opponents, RD_i is the ratings deviation of player i and:

$$q = \frac{ln10}{400} \tag{4.11}$$

$$g(RD) = \frac{1}{\sqrt{1 + 3q^2RD^2/\pi^2}} \tag{4.12}$$

$$E(R_A, R_i, RD_i) = \frac{1}{1 + 10^{-g(RD_i)(R_A - R_i)/400}} \tag{4.13}$$

$$d^2 = (q^2 \sum_{i=1}^{n} g(RD_i)^2 E(R_A, R_i, RD_i)(1 - E(R_A, R_i, RD_i))^{-1} \tag{4.14}$$

[26]

Elo's influence is obvious in the similarities between equations 4.5 and 4.13. Both approaches use the logistic curve to determine a player's expected score but the addition of the ratings deviation allows the system to slow the rating changes of the most prolific players while increasing the change in new player ratings. This is made possible by the independent nature of equation 4.13. Unlike equation 4.6, the expected scores of Players A and B do not add to 1 which allows for the scenario described by Glickman above, where one player gained a large number of points and the other lost little, to be possible. In a sense it is a refinement of the technique used by FIDA described and in Section 4.2.2 that modifies the k-factor as players move up in rank [28].

As will be shown in Chapter 7, the Glicko System preforms well in practice, beating the Elo system in predicting the outcomes of future games. However it does not reach the level of accuracy attained by Chessmetrics, and because of its increased implementation complexity is not as widely used. However as Chessmetrics does not qualify as a practical chess rating system, modified versions of Glinko are a popular choice for entries into the Deloitte/FIDE Chess Rating Challenge. A more complicated and accurate system, Glicko-2 was developed in 2001 as a replacement for Glicko [27]. It was used as

p	d_p	p	d_p	p	d_p	p	d_p	p	d_p	p	d_p
1.0	800	.83	273	.66	117	.49	-7	.32	-133	.15	-296
.99	677	.82	262	.65	110	.48	-14	.31	-141	.14	-309
.98	589	.81	251	.64	102	.47	-21	.30	-149	.13	-322
.97	538	.80	240	.63	95	.46	-29	.29	-158	.12	-336
.96	501	.79	230	.62	87	.45	-36	.28	-166	.11	-351
.95	470	.78	220	.61	80	.44	-43	.27	-175	.10	-366
.94	444	.77	211	.60	72	.43	-50	.26	-184	.09	-383
.93	422	.76	202	.59	65	.42	-57	.25	-193	.08	-401
.92	401	.75	193	.58	57	.41	-65	.24	-202	.07	-422
.91	383	.74	184	.57	50	.40	-72	.23	-211	.06	-444
.90	366	.73	175	.56	43	.39	-80	.22	-220	.05	-470
.89	351	.72	166	.55	36	.38	-87	.21	-230	.04	-501
.88	336	.71	158	.54	29	.37	-95	.20	-240	.03	-538
.87	322	.70	149	.53	21	.36	-102	.19	-251	.02	-589
.86	309	.69	141	.52	14	.35	-110	.18	-262	.01	-677
.85	296	.68	133	.51	7	.34	-117	.17	-273	.00	-800
.84	284	.67	125	.50	0	.33	-125	.16	-284		

Figure 4.4: The logistic Elo table used by FIDE players to determine their ratings. p is the score of the game, while d_p is the ratings difference [12]

the basis of Microsoft's TrueSkill rating system which was designed to infer individual rankings from team games [36].

4.3 Effectiveness of Practical Algorithms

4.3.1 Advantages

Ease of calculation is one of the largest contributing factors to the success of practical rating algorithms. In real-world applications it is often not practical to find a model that fits all the data accurately. According to FIDE, a practical chess rating system "cannot involve any iterative computation that is carried out to convergence in order to solve an optimality criterion" [21]. While the algorithms presented in this chapter meet that criterion, the machine learning approaches discussed in Chapter 5 will not. Therefore it seems likely that although these algorithms may be less accurate than other models they

will continue to experience more real-world usage, at least within the world of chess.

The impression of fairness generated by the open and understandable assumptions and computations made by most practical algorithms is considered very valuable. Large competitive organizations rely heavily on the trust of their members and using an algorithm that allows members and observers to do ratings calculations by hand themselves reinforces that trust. Concepts that are easy to grasp and seem grounded in reality, like Elo's logistic distribution, are more readily accepted by players then abstract concepts like neural networks or Bayesian inference. Therefore organizations work hard to keep things simple, going so far as to publish charts (see Figure 4.4) that break down ratings calculations into simple arithmetic. This allows competitive chess players to compute their own ratings independent of the governing chess body and therefore verify for themselves that rating calculations have been executed correctly.

4.3.2 Drawbacks

Although the advantages of practical algorithms are clear and have lead to widespread adoption of Elo and other similar systems, for applications in which accuracy is the most important metric practical rating algorithms will not perform optimally. Systems that evaluate every unique participant with an assumption of identical consistency, knowledge progression and motivation are doomed to perform less accurately then algorithms that take those factors into account. This can be seen in the difference in accuracy between Whole History Rating and Chessmetrics, which model skill level changes over time, and Elo which does not. Similarly Elo, which tries to represent the inconsistency of players performs much better than The Harkness System.

CHAPTER 5

RATING WITH MACHINE LEARNING

Machine learning is a subset of artificial intelligence research that focuses on algorithms capable of developing statistical models of data systems from limited amounts of data [39]. As rating data is likely to contain a wealth of information that may or may not be relevant to a player's skill level, machine learning techniques are excellent candidates for providing the framework for an extremely accurate rating algorithm. With machine learning we can independently model each player, explore new indicators for performance and continue to update the rating method as more games are played. In this chapter we present a basic overview of some machine learning concepts and techniques while exploring the benefits and challenges of using machine learning for rating. Also included is an overview of four algorithms that take advantage of machine learning techniques.

5.1 Supervised Learning

All of the algorithms presented in this chapter use supervised learning to develop models for rating. This means that all input to the system is given as a pairs of input and desired output. With this method the system can iterate through its learning method until the results are below some predetermined error rate. In other words, the system "learns by example" to predict an outcome based on input previously supplied [15]. Since functions are defined to match this provided output, when given input not previously observed by the system an output value is still defined. As the system tunes its methods to match the desired output the hope is that it also improves its accuracy with unobserved input-output pairs. Alternative learning methods like unsupervised learning are not as efficient in rating systems because they provide no means to nudge the algorithm in the right direction after an incorrect answer has been reached.

5.2 Optimization

Optimization is a core component of machine learning. Its purpose is to develop an optimal equation or value to represent known data which can then be applied to unknown data with good results. Since rating applications involve uncertainty by definition the best we can do is optimize to some solution that fits all of data available and is in some way "optimal" [4]. However this does not mean that the resulting optimization will be accurate for all data. Some perceived indicators may turn out to be random while underlying information might be lost in a datasetf too small in size. There are many techniques available for use in optimization, including stochastic gradient descent used in Elo++, and Newton's method employed by Whole History Rating.

Optimization algorithms typically take an iterative approach to solving an optimization problem. In each iteration an adjustment is made to the proposed solution bounded by the learning rate or step size of the function. This step size is adjusted to become smaller and smaller as the algorithm continues and get closer to minimum or maximum. When it reaches convergence the algorithm halts. Some algorithms utilize random movements or reordering of the dataset to avoid converging on local minima or maxima.

5.3 Overfitting

Overfitting is a serious problem for rating algorithms that attempt to use optimization techniques. Overfitting occurs when a statistical model ties itself too closely to the data used to train it and ends up modeling the random noise in the data instead of the underlying model. When the algorithm is used on real data it then makes assumptions it shouldn't, resulting in incorrect results. Overfitting is typically caused by attempting to infer too much information from not enough data. A system that violates parsimony by

including more sources or complicated approaches than are necessary is by definition overfitting [30], but it can be difficult to tell because the model appears accurate through the lens of the test data. The easiest and most common method for detecting overfitting is through cross-validation, performed through use of at least two test data sets, the first for training the model and the second for evaluating that model. If the system does not perform as well on the second system as it does on the first then overfitting is likely occurring.

During competitions like the Deloitte/FIDE Chess Rating Competition, overfitting frequently occurs because participants attempting to raise their place on the leaderboard inadvertently overfit to the test data. Yannis Sismanis, the winner of of the first Kaggle chess rating competition experienced this first hand when his winning algorithm, submitted just four weeks into the contest did not appear to be in contention for first place according to his own calculations [41]. However when the final results of the competition were announced it turned out that those with better preliminary scores had overfit their algorithms and therefore did not provide accurate predictions for the evaluated games.

5.4 Advantages and Disadvantages in Rating Systems

Machine learning approaches to rating systems provide a lot of potential in rating systems by allowing for extremely accurate and easy to generalize solutions. By generating a model that is unique for each game and each player involved, such a system can potentially more accurately describe the eccentricities of individual players and competitions. Since the algorithm controls the influence of input data, it can also be easier with such a system to discover input variables that affect the output and ignore the ones that don't.

The disadvantages of machine learning techniques mainly concern efficiency. Since ML algorithms must learn the characteristics of the system instead of merely applying

rules, such algorithms may take much longer to complete. They typically must iterate over the dataset multiple times compared to the one or two iterations allowed in a practical rating system. Machine learning systems are also at a disadvantage when new games are added to the system after it has been trained. While a practical system is able to update the rating of a player with minimal effort, a machine learning approach is likely to need to retrain the entire system if utmost accuracy is required. This makes machine learning algorithms difficult to apply in applications like computer games where instant rating updates are a requirement. However when rating accuracy is the most important factor it is usually advantageous to use one of the techniques described in this chapter.

5.5 Bayesian Learning

Bayesian learning methods use Bayesian inference to determine the statistical validity of hypotheses made about a data set and create a statistical model based on those hypotheses. This model is then used to predict the likelihood of specific events and therefore the results of actions performed on the system.

5.5.1 Bayesian Inference

Bayesian inference is a statistical method applied to a collection of variables initialized with a standard probability. Through observation and application of Bayes' rule these probabilities are modified to reflect the rules of the system. As additional information about the system is collected, the confidence in a hypothesis made about the system approaches 100% or is reduced to almost 0%. This reflects the same method of learning and hypothesis generation that is employed by the scientific method [7]. Scientific theories grow and die by the evidence presented for and against them; by modeling this process of evidence collection with mathematics Bayesian inference constructs a system

in which not only are the results reliable and accurate, but the steps taken to generating the resulting model are intuitive and understandable.

Bayes Rule states that :

$$P(H|E) = \frac{P(E|H)P(E)}{P(E)} \tag{5.1}$$

where H is a hypothesis and E is the evidence currently available to the system. $P(H|E)$ is the conditional probability that H occurs given evidence E. The result is an indicator of the likelihood that H is true. [2] The equation itself is quite intuitive. If it is more likely that when H is observed E is observed than E occurring on its own the factor $P(E|H)/P(E)$ is greater than 1, representing a positive correlation. In the opposite case, where E has been observed alone more often than in conjunction with H, the factor is less than 1, which negatively modifies the stand alone probability of H resulting in a smaller outcome [7]. $P(H)$ and $P(E|H)$ are defined from the current probability model, while $P(E)$ is calculated from the probability of E given all hypotheses H_i that are mutually exclusive with H, i.e:

$$P(E) = \sum P(E|H_i)P(H_i) \tag{5.2}$$

Bayesian inference uses an iterative model that updates its stored probabilities each time new evidence is provided. The result of each iteration modifies the probabilities in the system in preparation for the next iteration. As the system's hypotheses become more accurate changes from iteration to iteration become smaller and smaller, reflecting the increased knowledge of the system.

The challenge in applying Bayesian inference directly as a machine learning technique is the shear amount of computation required in order to model a system of any complexity [8]. Instead methods like Monte Carlo and approximate Bayesian computation (ABC) are used to reduce complexity.

5.5.2 Use in Rating

Bayesian learning has recently found success in rating applications. The ability to model the uncertainty and value of a player's rating in a more formal way is highly useful. The outcome of Bayes' rule is a single probability that is a function of any number of additional variables that might include past performance, a time-line of when games were played, teammate ratings etc. This allows the rating system to easily add new metrics for rating without redesigning the entire algorithm for the system. Metrics only influence an outcome if they have been determined to effect that outcome. The flexibility allowed by such a system is why a large number of contestants in the Deloitte/FIDE Chess Rating Challenge used such systems along with customized sets of tracked data [21]. However, overfitting becomes a larger and larger potential problem as data is added so the system must remain careful about the information it adds to its model.

5.6 Implementations

This section describes the architecture and implementation of three ranking systems that use machine learning techniques to calculate performance ratings. While all of the systems are efficient enough to run on large datasets and use no precomputed models, they do not qualify as practical rating systems for a number of reasons.

5.6.1 Chessmetrics

Creator Jeff Sonas describes Chessmetrics as a "weighted and padded simultaneous performance rating" [18]. Developed in 2002, Chessmetrics is one of the most accurate public rating algorithms. It was designed primarily with the purpose of constructing historical chess ratings but since then has been used in a variety of applications. The system is centered around the idea of a player's "performance rating," a score assigned

to a particular event, be it a tournament, single match or casual game, that rates the difficulty of that performance. The better the performance the higher the score awarded. The compilation of all of a player's performance scores is then used in the computation of her ranking. The equation for the performance rating (P_A) of player A is:

$$P_A = R_{avg} + [(P_s - 0.5) * 850] \tag{5.3}$$

Where R_{avg} is the average ranking of the opponents faced and P_s is the player's percentage score for the event. A player's rating is then calculated with the formula:

$$R_A = \frac{(P_A * G_n) + (R_{avg} * 4) + (2300 * 3)}{G_n + 7} + 43 \tag{5.4}$$

where G_n is the number of games played. R_A is then linearly weighted depending on the number of months separating the game from the present day. This system addresses the time-line problem head on by giving much more emphasis to recent games and completely ignoring those played longer than 48 months before the rating period. This feature combined with the addition of seven fake draws added to a players rating also address the problem of negative incentives for competing. In a system like Elo players sometimes refrain from playing due to fears of lowering their ranking. In Chessmetrics since every player's rating is recalculated during each rating period whether or not they play, a player who avoids playing will find that her rating slowly but steadily decreases.

Sonas considers Chessmetrics a "simultaneous" performance rating system because unlike iterative rating systems like Elo and Glicko, Chessmetrics calculates player ratings from scratch at the beginning of every rating period. It does not matter what ratings players are seeded with, Chessmetrics uses the propagation method of defining new ratings iteratively until every player has been given a rating and the system converge. To ensure convergence only one restriction must be applied to the rating system. These restrictions can be as simple as "the tenth ranked player should have a rating of 2600" or "the top 10% of players should have ratings above 2300" [18].

Although equation 5.4 may seem almost arbitrary at first glance (43 is tacked on the end ”because the padding tends to pull down the rating”) the Chessmetrics algorithm has performed surprisingly well in practice [19]. As a benchmark in the 2011 Chess Rating competition, Chessmetrics performed well above Elo, Glicko and FIDE in predicting the results of 7,809 games more accurately [21]. This accuracy improvement is the result of the ”simultaneous” rating of players which allows the system to produce ratings consistent with the entire body of competitors.

5.6.2 Whole History Rating

The Whole History Rating system developed in 2008 by Remi Coulom uses Bayesian learning to construct a rating system in which time and the dynamic ratings of player are both factors [33]. It is based on the concepts from the Edo system which uses a static rating method [34] to develop historical chess ratings.

To motivate his algorithm, Coulom gives the example of two players new to a game that play only each other. After a number of games the players rankings will be accurate relative to their specific match-up, but completely removed from the rest of the system. Coulom makes the argument that if Player A then begins to play games against other more established players Player B should also find her rating updated, reflecting the new knowledge the system has gained. In an incremental rating system like Elo or Glicko this sort of change is not possible. A method that rates only a limited scope of games based on when they were played and generates a new rating for each time-frame, like a modified version of Chessmetrics, could address this problem, but Coulom believes that this does not lead to a good user experience. Players returning to the game find that their ratings jump quickly out of control and those that play often experience ratings that seem to stagnate.

Coulom’s solution as implemented in Whole History Rating uses Bayesian inference

and the Bradley-Terry model (denoted $P(S_{i,j}, R_i)$) to form an equation dependent on player ranking R_i and game score $S_{i,j}$:

$$p(R_i|S_{i,j}) = \frac{P(S_{i,j}, R_i)p(R_i)}{P(S_{i,j})} \tag{5.5}$$

in which $p(R_i)$ is the probability distribution of Player i's skill and $P(S_{i,j})$ is a normalization constant. The goal of the algorithm is to find an equation for a player's rating over time, $R_i(t)$, that maximizes the value of $p(R_i|S_{i,j})$. This maximum is optimized using Newton's method detailed in [33] applied to a vector that hold a player's ranking during every game played. This allows the system to address the problem outlined by Coulson above – future ratings of a player have the ability to effect the past ratings of their opponents. According to Coulson even just one iteration of Newton's method is sufficient to generate reasonably accurate ratings. However to achieve optimal results the algorithm must iterate until convergence.

Whole History Rating suffers from several problems common to machine learning approaches. The optimization that takes place makes it difficult to add games into the system while maintaining high accuracy without re-running the entire algorithm. Coulson suggests recording the outcomes of games played until a large enough batch has been collected to re-run Newtons method for each player. In his experiments he used a batch size of 1000 games that worked reasonably well. This inefficiency restricts the potential use of the system to games that update ratings over a rating period instead of those that update instantly.

Forgetting these faults however, Whole History Rating appears to perform extremely well at rating. In an experiment running on 10.8 million games the system took 7 minutes to complete 200 iterations. Although this is much longer than other algorithms, it outperformed Elo, Glicko and TrueSkill by 0.1%, a large margin in rating algorithms [33].

5.6.3 Elo++

The Deloitte/FIDE Chess Rating challenge that inspired this thesis was not the first chess rating challenge to appear on Kaggle. In August of 2010, the "Elo vs. the Rest of the World" challenge began. It ended on November 17th with submissions by 258 teams, the most accurate of which was dubbed by its author, Yannis Sismanis, Elo++ [20]. As it's name suggests, Elo++ borrows much of its theory from Arpad Elo's system discussed in Section 4.2.2. Sismanis' improvement mainly concerns avoiding overfitting while training the system to assign more accurate ratings. Ratings are calculated with the same logistic curve used in Elo (Equation 4.5) with the simple addition of a white advantage constant [41]. However instead of weighting every game played equally or linearly, games are weighted according to the following equation:

$$w_{i,j} = (\frac{1 + t_{i,j} - t_{min}}{1 + t_{max} - t_{min}})^2 \tag{5.6}$$

In which $w_{i,j}$ is the weight applied to the game between players i and j at time $t_{i,j}$. This equation addresses the timeframe problem without the need to throw out older games as many of Sismanis's competitors did. Since t_{min} and t_{max} are the minimum and maximum timeframes, games are weighted along a global instead of individual scale, a distiction from the much more complex systems of Glicko and TrueSkill that assign different weights to each player of the game.

The key assumption made by Elo++ is that since the majority of recorded chess games are played in tournaments in which players tend to play players of their own skill level, there is a strong correlation between the ratings of players that meet in these games. Used as a central component of the training algorithm, this assumption appears to have paid off well. Unfortunately other rating scenarios may not exhibit a similar structure, in which case Elo++ may not provide the most accurate rating method.

Sismanis used his hypothesis to avoid overfitting ratings while keeping them in the

range suggested by the opponents. Specifically, Elo++ defines a player's "neighborhood" as the multiset of all opponents that player has played against, regardless of which player played white. Since it is a multiset, an opponent will appear in a set multiple times if the opponents faced each other multiple times. A weighted average for player strengths is calculated for each player as follows, with the neighborhood for player i denoted N_i.

$$a_i = \frac{\sum_{k \in N_i} w_{i,k} t_k}{\sum_{k \in N_i} w_{i,k}} \tag{5.7}$$

a_i is used when updating ratings as it is most heavily influenced by a player's most recent opponents and opponent faced most often. It is also used as a neutral prior for regularization. The exact equation used to calculate total loss l is:

$$l = \sum_{i,j \in T} w_{i,j}(E_{A_i} - S_{A_i})^2 + \lambda \sum_{i \in D} (r_i - a_i)^2 \tag{5.8}$$

in which T is the training set that consists of the tuple:

$$T_i = \langle i, j, t_{i,j}, S_{A_i} \rangle \tag{5.9}$$

As in previous equations, S_{A_i} represents the actual score of player i in the game while E_{A_i} is the predicted score. λ is the white advantage constant. To optimize the ratings assigned by Elo++ the system is optimized in order to minimize the total loss. This is accomplished through use of iterative learning in the form of a stochastic gradient descent algorithm that trains with an estimate of the total loss gradient modified with noisy data. As the system iterates over the game, the introduced noise cancels itself out and the system begins to converge to a local minimum. Since rating systems rely on an abstract scale in which ratings are only meaningful in relation to other ratings, a system can have an infinite number of local minima, only some of which are optimal. The addition of noise ensures that non-optimal local minima can be avoided while the system continues to progress towards an optimal minima [1].

As Elo++ iterates through the system, player ratings are updated according to the learning rate ω and rating update equation defined in equations 5.10 and 5.11 respectively. ω is a function of the current iteration number i and the total number of iterations I. This rate allows initial iterations to greatly modify player ratings while later iterations can only make small adjustments. This specific equation is based on the learning rate equation from [1] but can be modified to produce different learning curves for different applications that converge at different rates.

$$\omega = (\frac{1 + 0.1I}{i + 0.1I})^{0.602} \tag{5.10}$$

$$R_{A_i} = R_{A_i} - \omega(w_{i,j}(E_{A_i} - S_{A_i})E_{A_i}(1 - E_{A_i}) + \frac{\lambda}{N_{i_{size}}}(R_{A_i} - a_i)) \tag{5.11}$$

The Elo++ rating update formula is based on the Elo update formula defined in equation 4.8. Change in rating is determined by the difference between expected score E_A and actual score S_A. However, unlike the static k-factors of Elo, the amount of variation in score is determined by a number of factors including the regularization constant λ, the size of the player's neighborhood, the average rating of their opponents, and the weight $w_{i,j}$. Since the update algorithm is expected to run multiple times, Elo++ does not have Elo's burden of correctly updating the player's rating on the first try. Inaccuracies produced by the first few iterations of the algorithm are corrected as the algorithm continues and ultimately results in a better scoring. According to Sismanis, $I = 5$ is high enough to produce ratings in the "vicinity of the minimum" and 50 iterations was sufficient to converge to the minimum in his tests [41].

Although Elo++ proved a very competent system at predicting the outcomes of chess games, it is not immune from from complications. The reliance on global constants λ and γ means that the system must be tweaked for each use, and assumptions about the the skill of players that play each other potentially reduces the usefulness of this algorithm in the general case. Additionally Elo++'s reliance on large numbers of iterations

over the data set disqualifies it as a ”practical chess rating system” and may mean that it is not practical for larger data sets.

CHAPTER 6

IMPLEMENTATIONS

This chapter presents the algorithms I implemented over the course of this thesis. The original goal of the thesis was to design an algorithm that could compete with the best of the competition, and while that goal was eventually abandoned, the process of designing, programming and testing various documented and undocumented algorithms was a valuable part of the learning process. Through this I was able to gain a greater understanding of the challenges and limitations faced by performance rating algorithms and also try my hand at developing my own.

The algorithms I implemented fell into two general categories: documented rating algorithms currently in use, and algorithms I made up myself. Most of the algorithms were written in Java to take advantage of a common class I developed to make use of the datasets provided by Kaggle and avoid too much repetitive code. This reliance on Java was definitely a factor in the runtimes of my algorithms but as rapid development was more important to me than overall efficiency, Java was the better choice.

In the documented algorithm category I implemented Harkness, Elo, Glicko and Chessmetrics. Of these Harkness was of course the easiest to implement as the algorithm is exceedingly simple. Elo was also relatively easy as the methods involved are straightforward. I wrote my version of Elo in both Java and C++ to gain a greater understanding of the performance difference. The C++ version performed much faster but took longer to debug, validating my choice to use Java. My implementation of Glicko was aided by a library developed for Glicko-2 implementations in web games available at [10]. By examining and modifying its code I was better able to understand the Glicko algorithm. Of these algorithms my implementation of Chessmetrics proved the most accurate though my version did not perform as well as the one written by Jeff Sonas.

I also implemented a number of rating methods for which I developed the algorithm. These included a modified version of the Elo algorithm, two approaches using neural networks as the basic learning method and a method using scalar vector machines. My modified Elo approach worked the best of the algorithms I designed. Instead of making only one pass through the set of games, my version iterated through the entire dataset until player's rating changed only slightly from beginning to end. This acted as a simple implementation of the optimization problems outlined above, and earned a score placing it solidly in the middle of the pack. The machine learning approaches using neural networks and SVMs proved to be my downfall. Attempting to get these programs to work on the huge Kaggle dataset taught me a lot about the practicality of machine learning algorithms. Though the approaches of the previous chapter also used machine learning, they implemented methods that are much better for large systems. In the end these algorithms were only able to complete on small sets of data making them useless for actual rating applications and useless for the Kaggle competition.

When my implemented algorithms met with limited success, I turned my focus towards learning more about the algorithms that do perform well in the real-world. If given more time I believe I could develop a least a few more algorithms that would perform reasonably well without nasty performance problems. A good starting point is the Glicko algorithm which has proven remarkably accurate for the limited amount of resources it needs to complete. Glicko only tracks a small amount of information about each player. By increasing this, perhaps by storing a player's preference to playing black or white or some information about performance consistency, it is very possible that the accuracy of Glicko could be improved.

I do not compare my versions of algorithms in the next section because my code potentially contained bugs and inefficiencies that would modify the results. Luckily enough the actual authors of every algorithm compared (except, of course Arpad Elo)

implemented versions of their own algorithms which I consider much more trustworthy.

CHAPTER 7

METHODS OF COMPARISON

Rating algorithms can be compared across a variety of metrics. We can examine aspects as diverse as scalability, simplicity and the structure of assigned ratings. Perhaps the most important however are accuracy, performance and fairness. In the next section we explore the ways to measure and compare these metrics and evaluate some of the algorithms presented in this thesis. When comparing techniques it is important to realize that dominance in one area does not mean an algorithm will perform as well in another and that algorithms may perform differently on different datasets. Therefore any results gained through these methods should only be compared to other algorithms run in the same conditions.

7.1 Methods of Determining Accuracy

This section contains two algorithms for directly determining the accuracy of a rating algorithm and one technique for comparing the output of two algorithms to determine how their ratings differ. In general we evaluate the accuracy of algorithms through their performance on real-world data. First the rating system is trained on some dataset of games that contain the results of the matches. Using the generated model the system then predicts the results of some number of games for which the outcome is unknown. By measuring the difference between predicted and actual outcome we can get a sense of the accuracy of the method used.

7.1.1 Binomial Deviance

Binomial deviance calculates the distance of a set of predictions from the actual desired result. This is the method used in the Deloitte/Fide Chess Rating Challenge to rank the

accuracy of entries. In the contest participants were required to predict the outcome of 100,000 chess games based on a model generated from the training set. The rating system with the lowest deviance was then declared the winner. The equation for calculating the binomial deviance is :

$$B = \frac{\sum_{i=1}^{n} S_{a_i} \log_{10} E_{a_i} + (1 - S_{a_i}) \log_{10} (1 - E_{a_i})}{n} \tag{7.1}$$

where n is the number of games predicted, S_{a_i} is Player A's actual score in game i, and E_{a_i} is the predicted score of Player A in game i [21]. The lower the binomial deviance the more accurate the system, as that indicates that the predicted game outcomes were closer to the desired result. A typical rating algorithm has a deviance of around 0.26 while good ones hover in the 0.25 range. The best entries in the Kaggle competition were able to secure binomial deviances in the 0.24 range [21]. However these algorithms used methods not available to rating algorithms in the real-world and therefore such results are of questionable usefulness. Selected results of the competition are presented in Table 7.1.

7.1.2 Root Mean Square Deviation

The root mean squared deviation of a set of predictions is calculated by first finding the squared error of each prediction with the equation

$$SE = (S_{a_i} - E_{a_i})^2 \tag{7.2}$$

The lowest possible value is the most desirable, indicating that a prediction is 100% accurate. To estimate the squared error across an entire algorithm we take the average of the errors. Finding the root of that average gives the root mean square deviation for the entire set of predictions. This measurement is useful because it describes the error with the same units used in the prediction [13]. It is a commonly used method

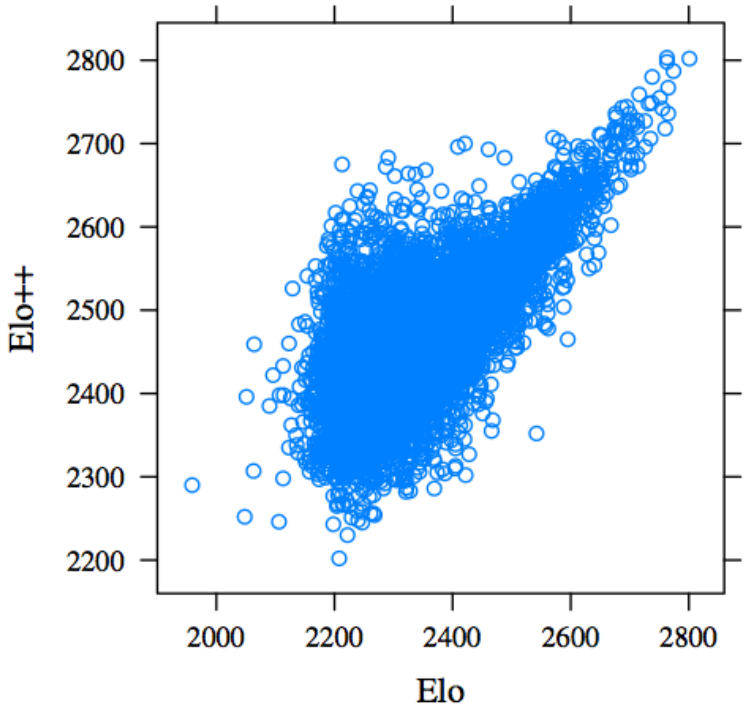

Figure 7.1: A scatter plot depicting the ratings of players as calculated by Elo++ versus the ratings produced by Elo [41]

in statistics to describe the difference in a prediction and the actual result and has been used in a previous performance rating competition [20].

7.1.3 Comparing Rankings

A visually interesting way to investigate the product of a ranking algorithm is to directly compare the ratings of players as calculated by two different systems. Although the ratings scale is almost certain not to match between the two systems, as long as both systems use interval scales this does not matter. A linear relationship between plotted points indicates that the systems agree on a rating while points above and below that line indicate players on with which a system has assigned a relatively higher or lower rating. In Figure 7.1 such a plot is shown in which it is clear that while Elo and Elo++ agree on

the ratings of very strong players, Elo++ tends to rate some weaker players somewhat higher than Elo does. While plots like this do not directly give a measure of accuracy, if two algorithms exhibit disparities in their accuracies, one way to get a high level sense of why that is occurring is by using one of these graphs. This specific example is very interesting because of the large difference in prediction ability between Elo and Elo++. Since Elo++ performs much better, based on this plot we can make the educated guess that mid-range players in Elo rated systems tend to be under-rated.

7.2 Determining Practicality

The notion of the practicality of a performance rating method can take a variety of meanings. It can concern the number of games a player must play, the time-frame required for rating accuracy or the organizational costs incurred in adopting it. Typically however when evaluating performance rating algorithms we discuss practicality through the lens of the computational power required to completely rate the system. Such a definition of performance can be measured using standard mathematical techniques for estimating complexity. Simply counting the number of times a program must iterate through a dataset before completing is reasonably accurate method of estimating the amount of time the algorithm will take to complete. Similarly, counting the amount of data stored for each player allows us to quickly get a sense of how much storage space the algorithm will use. These measurements are extremely important because they define the applications for which a rating algorithm can be used. The faster and more efficiently an algorithm can produce a usable rating the better.

7.3 Determining Fairness

There are two sides to the question of fairness in performance rating algorithms. The first, the actual definition, concerns whether an algorithm treats each player equally, giving each player a fair chance to rise and fall in the rankings. This question can easily be answered through an investigation of the methods used during rating. The second side is much more difficult and subjective. The perception of fairness is just as important, if not more important, than actual fairness. If players perceive a method to be unfair they will refuse to participate in systems that use it. Unfortunately, determining whether a system seems fair to competitors is not an easy task to undertake. Making an algorithm as simple as possible and including as little optimization and machine learning as possible is definitely desirable, but ultimately fairness can only be decided by those that participate in a system.

7.4 Algorithm Comparison

The Deloitte/FIDE Chess Rating Challenge provides an excellent data set with which to compare the results of a number of the algorithms discussed in this thesis. Since all algorithms are run on the same data, the results are directly comparable. However it is important to remember that these algorithms evaluated here can be "temperamental" and if run under different settings or on different data could produce different results. The specific implementations shown here are provided by the authors of the algorithms and can therefore be considered to be representative of the abilities of the algorithms.

Rows for the first place entry and the all draws benchmark are provided to locate the data for the algorithms discussed in this thesis. Interestingly but not unexpectedly, the approaches that use machine learning techniques like Chessmetrics, Elo++, and the current first place entry perform more accurately than practical systems like Glicko and

Table 7.1: Deloitte/FIDE Chess Rating Results for selected algorithms

Algorithm	Binomial Deviance
First Place Entry	0.246679
Improved Elo++	0.251337
Chessmetrics	0.256770
Glicko	0.257676
Optimized Elo	0.259514
FIDE Rating	0.259564
All Draws	0.301030

Elo. However if the leaderboards focused on runtime instead it is likely that those positions would be reversed as they are in data from Remi Coulom's paper on Whole History Rating [33].

Table 7.2: Comparison of algorithm runtimes showing the difference in performance between machine learning and "practical" systems

Algorithm	Runtime in Seconds
Elo	0.41
Glicko	0.73
TrueSkill	0.40
Whole History Rating	252.00

The runtime data, presented in Table 7.2 was produced with a dataset of 726,648 games of Go taken from the KGS Go Server. From this table, the performance gain from using practical rating methods as opposed to machine learning approaches becomes obvious. Elo took less than $1/600$th of the time required to run Whole History Rating. Additionally runtime data only takes into account the time required to seed the initial system ratings. This emphasizes the main reason why systems like Whole History Rating are not practical for real time systems like computer games. To retain accurate information about a growing system Whole History Rating must recalculate periodically. With a runtime of over four minutes it is not possible to develop this into a real time approach. However the gain such systems exhibit in accuracy are certainly desirable for applications like retroactive rating in which we are interested in comparing

the historical performance of competitors across era boundaries.

Of the algorithms discussed in this thesis, the Elo Rating System would probably be considered the most ”fair” by the general population. Though the ratings generated by it are not the most accurate, it’s methodology is simple, understandable, and it treats each player equally. Chessmetrics does an incredibly good job at predicting match results, but its bizarre equations and tendency to lower the rankings of infrequent players might make many suspicious. Similarly the Glicko system, though it performs well and makes fair assumptions about the skill of players, contains so many complicated equations that Jeff Sonas predicted that it was right on the edge of what would be considered ”acceptable” by FIDE [21]. Whether or not this matters depends on the application of the algorithm. Historical chess players tend not to complain about the methods used to rate them, so again more complicated and tuned methods can be used in such calculations. Modern competitions however tend to stick to Elo and Elo-like systems, sacrificing accuracy for piece of mind.

CHAPTER 8

CONCLUSION

It might be noted that the algorithms described in this thesis follow a pattern that flows from the simple to the complex. Certainly it is the case that as access to computational power has increased, the desire for models that can more accurately and quickly predict the outcomes of pairwise comparisons has also increased. The overwhelming participation in the Deloitte/FiDE Chess Rating Challenge is evidence of that interest. While increased accuracy is an area of much current research, the algorithms that are most commonly implemented for active systems are the Elo Rating System and the Glicko Rating System. These are algorithms with good accuracy but more importantly the ability to perform incremental rating, efficient computation, and do so with clarity. This large and important trade-off between practicality and accuracy is at the center of developing new performance rating methods. Computer games are sure to lead the way in developing more and more accurate systems for practical rating. As online games gain participants and legitimacy as a form of competition, players will demand more accurate and up-to-date rating information. Machine learning approaches and other ”impractical” systems will also continue to improve as passionate hobbyists and researchers strive to develop the next best rating system. Though the demand for historical ratings is not backed by the excitable world of online gaming, the passion and interest of people, like Jeff Sonas and Rod Edwards that drive it forward is unmatched.

Performance rating is such a dynamic field, full of different requirements and costs, that there truly is no best rating solution. Every algorithm has its place and its use and it’s own interesting perspective on the nature of competition.

BIBLIOGRAPHY

[1] J. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley-Interscience series in discrete mathematics and optimization. Wiley-Interscience, 2003.

[2] A. Dempster. A Generalization of Bayesian Inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, 30(2):pp. 205–247, 1968.

[3] A. Edo and S. Sloan. *The Rating of Chess Players, Past and Present*. Ishi Press, 2008.

[4] A. Shapiro, D. Dentcheva, A. Ruszczyński and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2009.

[5] W. Batchelder and N. Bershad. The statistical analysis of a thurstonian model for rating chess players. *Journal of Mathematical Psychology*, 19(1):39 – 60, 1979.

[6] Bowl Championship Series. BCS selection procedures. http://www.bcsfootball.org/news/story?id=4819597, 2011.

[7] C. Howson and P. Urbach. *Scientific reasoning: The Bayesian approach*. Open Court Publishing Co, 1989.

[8] G. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393 – 405, 1990.

[9] D. Ross. Arpad Elo and the Elo Rating System. http://www.chessbase.com/newsdetail.asp?newsid=4326, Fall 2007.

[10] dhilder. Java Rating Service. `http://java.net/projects/jrs/`.

[11] ESPN. 2010 NCAA Football Rankings - Postseason. http://espn.go.com/college-football/rankings/_/seasontype/3.

[12] FIDE. The FIDE Handbook. http://www.fide.com/fide/handbook.html.

[13] G. McPherson. *Statistics in Scientific Investigation: Its Basis, Application, and Interpretation*. Springer texts in statistics. Springer-Verlag, 1990.

[14] H. David. *The Method of Paired Comparisons*. Griffin's Statistical Monographs Courses. C. Griffin, 1988.

[15] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2001.

[16] ITTF. ITTF World Ranking - Womens. `http://www.ittfranking.com/gen/world/worldW_en.htm`.

[17] J. Keener. The Perron-Frobenius Theorem and the Ranking of Football Teams. *SIAM Review*, 35(1):pp. 80–93, 1993.

[18] J. Sonas. Chessmetrics. http://db.chessmetrics.com/.

[19] J. Sonas. The Sonas Rating Formula Better than Elo? http://www.chessbase.com/newsdetail.asp?newsid=562, October 2002.

[20] J. Sonas. Chess ratings - Elo versus the Rest of the World. http://www.kaggle.com/c/chess, 2010.

[21] J. Sonas. Deloitte/FIDE Chess Rating Challenge. http://www.kaggle.com/ChessRatings2, 2011.

[22] D. Jennings, C. Carlin, T. Hayden, and M. Gammell. Third-party intervention behaviour during fallow deer fights: the role of dominance, age, fighting and body size. *Animal Behaviour*, In Press, Corrected Proof:–, 2011.

[23] K. Harkness and United States Chess Federation. *Official chess handbook*. D. McKay Co., 1956.

[24] M. Khan and A. Iqbal. Speed Estimation in Five-Phase Syn-Rel Machines using ELO-Algorithm. In *16th National Power Systems Conference*, December 2010.

[25] L. Kaufman. *The Chess Advantage in Black and White: Opening Moves of the Grandmasters*. McKay chess library. Random House Puzzles & Games, 2004.

[26] M. Glickman. Parameter Estimation in Large Dynamic Paired Comparison Experiments. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 48(3):377–394, 1999.

[27] M. Glickman. The Glicko-2 System. 2001.

[28] M. Glickman. The Glicko System. April 2001.

[29] M. Glickman and A. Jones. Rating the Chess Rating System. *Chance*, 12, 1999.

[30] M. Hawkins. The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004.

[31] Official World Golf Ranking. Structure of Ranking Points and Rating Values from January 1 2007. http://dps.endavadigital.net/owgr/doc/content/doc_9_94.pdf.

[32] P. Dangauthier, R. Herbrich, T. Minka, T. Graepel. TrueSkill through time: Revisiting the history of chess. *Advances in Neural Information Processing Systems*, 20, 2007.

[33] R. Coulom and I. Sequel and C. Grappa. Whole-History Rating: A Bayesian Rating System for Players of Time-Varying Strength.

[34] R. Edwards. Edo Historical Chess Ratings. 2006.

[35] R. Edwards. FIDE Chess Rating Inflation. May 2006.

[36] T. Graepel R. Herbrich, T. Minka. Trueskill tm: A bayesian skill rating system. In *Advances in Neural Information Processing Systems 20*, pages 569–576, January 2007.

[37] R. Hunter. MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, 32:2004, 2004.

[38] R. Wilson. Ranking College Football Teams: A Neural Network Approach. *Interfaces*, 25(4):pp. 44–59, 1995.

[39] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2009.

[40] S. Stevens. On the Theory of Scales of Measurement. *Science*, 103(2684):pp. 677–680, June 1946.

[41] Y. Sismanis. How I won the "Chess Ratings - Elo vs the Rest of the World" Competition. December 2010.

[42] T. Fenner, M. Levene and G. Loizou. A Discrete Evolutionary Model for Chess Players' Ratings. *ArXiv e-prints*, March 2011.