Razvan Kusztos
Girton College
`rek43@cam.ac.uk`

Computer Science Tripos, Part III

LE49: Probabilistic Machine Learning

Dota2 Player Ranking Using Trueskill$^{\text{TM}}$

Report

23 February 2018

Word count: nana

1 Introduction

Player ranking has many important uses in competitive sports or games. Firstly, they provide a basis of matchmaking systems, improving the experinces of both player and watchers. They can lead to balanced games for the participants, while keeping them dynamic for audiences. Secondly, ranking players can be used to determine tournament qualification (such as the access to ranked matches in Dota2). Lastly, ranking triggers the human ingrained response to competition, attracting players to stay for longer, thus benefiting the industry.

The essence of ranking algorithms is determining a real skill score for each player, such that its magnitude is representative of the player's future performances. In a probabilistic setting, these skills must also be depictive of the probability of games' outcome.

One of the most influential statistical ranking systems is due to Arpad Elo (1959) [1], and has been used extensively in chess ranking [2]. In the online game scene, it has been used in World of Warcraft's PvP[1] matchmaking [2].

The core of the *Elo* algorithm is that each player's game performance is normally distributed around their skill, $p \sim \mathcal{N}(skill, \beta^2)$. The probability that one player wins is then proportional to the probability that its performance is higher than its adversary's.

$$P(p_1 > p_2 | skill_1, skill_2) = \Phi\left(\frac{s_1 - s_2}{\sqrt{2}\beta}\right),$$

where Φ is the cumulative density of a normalised Gaussian distribution.

Seen as improvements to *Elo* are the *Glicko* rating systems [3], [4]. They replace the constant variance parameter used by *Elo* system by shifting to a bayesian framework. In the setting, the player skill itself is seen as coming from a prior distribution and, with each game, the uncertainty (variance) of their skill decreases.

Unfortunately *Glicko* is limited to single player games. Microsoft TrueSkill$^{\text{TM}}$was developed to address this limitation, by allowing individuals's skills to be updated given the team's performance, as well as considering games with more than 2 competing entities.

An important question that needs yet to be addressed is how to enhance these existing systems by incorporating other information about the players, rather than just win vs. loss information. For instance, in chess, incoroporate knowledge of whether the player controls the black or the white. A way in which this problem has been addressed is due to *Zhang et al.* [10]. They suggested that an entity skill, rather than being a single real value, could be understood as a vector of skill components: *a virtual team of [the player's] various skills.* Of particular interest to the current work is that they validate their algorithm on *DotA*, a precursor of the game I will discuss in this work.

In the rest of this report, I will introduce factor graphs and the message passing algorithms, necessary for understanding the *TrueSkill^{TM}* original implementation. I will then describe the assumptions made in the *TrueSkill^{TM}* model. Next, I will introduce the particularities of the *Dota2* game, a multiplayer online battle arena (MOBA) that has been part of the advent of e-sports. Lastly, I incorporate these considerations into an implementation that I evaluate on both real and synthesized datasets.

[1]player vs. player

[2]http://wowwiki.wikia.com/wiki/Arena_personal_rating

2 Implementation

2.1 Factor Graphs

The *Factor Graph* is a graph-based graphical model. Like *Markov Random Fields* or *Bayesian Networks*, it is used to model multivariate probability distribution while making explicit the conditional independence assumptions of the random variables. The *Factor Graph* is based on a bipartite undirected graph with two types of nodes: The **variables** (depicted as circles), and the **factors** (depicted as squares). The factors represent functorial combinations of the neighbouring variables. The bipartite structure is achieved by enforcing the constraint that edges can only connect nodes of different types.

Another way to see it is that the factor graph expresses a factorisation of a joint distribution.

$$g(x_1, x_2, x_3, ..., x_n) = \sum_{j \in J} f_j(\mathbf{x}_j)$$

Where J is the set of factors and $\mathbf{x}_j$ is the list of variables that are adjoint to the node representing the factor f_j.

We are generally interested in performing marginalisation. This is usually required in practise, for example when computing likelyhood functions.

$$g_i(x_i) = \sum_{x_j, j \neq i} g(x_1, x_2, ...x_n)$$

2.2 Message Passing

Marginalisation approximations are performed using the so called *sum-product algorithm*. Also called *Belief propagation* [8], this algorithm was designed to be a general, easily paralellisable, exact inference of graphical model based on trees. However, it is known to provide approximations on general graphs [9], including cyclic graphs [7].

It has been shown that this algorithm is an instance of the *Expectation propagation* [6] on factor graphs. The algorithm approximates marginals by propagating *messages* along the edges of the factor graphs. In the context of *Expectation propagation*, messages are nothing but factor approximations obtained in the main loop of the algorithm [6].

Concretely, message passing can be defined through the following equations:

$$\begin{aligned}
p(v_i) &= \prod_{f \in F_{v_i}} m_{f \to v_i}(v_i) \\
m_{f \to v_i}(v_i) &= \int ... \int f(\mathbf{v}) \prod_{i \neq j} m_{v_j \to f}(v_i) dv_j \\
m_{v_i \to f}(v_i) &= \frac{p(v_i)}{m_{f \to v_i}}
\end{aligned}$$

In the previous, f refers to a factor and v to a variable node. I use the standard notation $m_{f \to v}$ to represent the message from node f to node v.

2.3 TrueSkill Implementation

For simplification, I will introduce this problem with no-draw games consisting of two teams, each with two players.

We model all players skills as being drawn from a prior distribution. This encodes the knowledge we have about the players before begining to analyse their game outcomes. Then, we calculate the team performance as being the sum of the members' skills with an added *Gaussian noise*, representing an *luck* element. Lastly, the probability of the team winning, is given by the probability the first team's performance is higher than the second team's performance

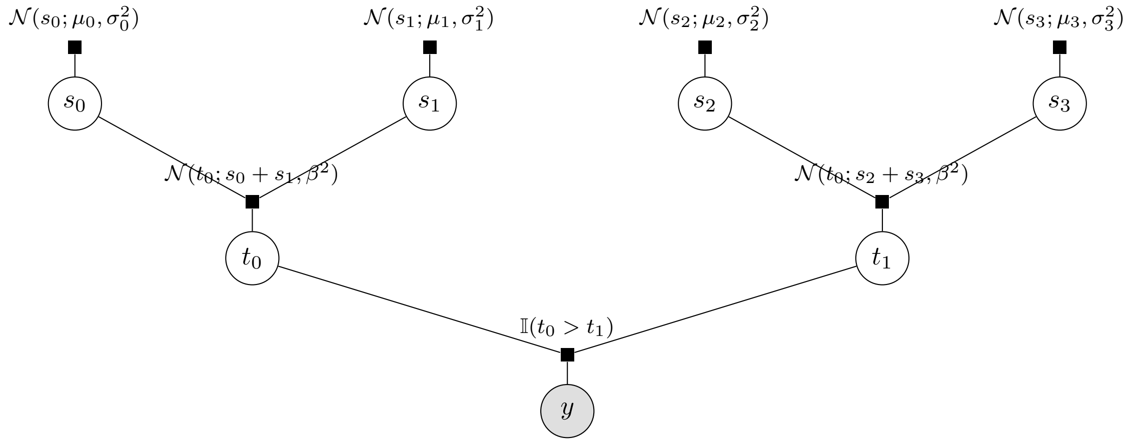
$\mathcal{N}(s_0; \mu_0, \sigma_0^2)$
$\mathcal{N}(s_1; \mu_1, \sigma_1^2)$
$\mathcal{N}(s_2; \mu_2, \sigma_2^2)$
$\mathcal{N}(s_3; \mu_3, \sigma_3^2)$
s_0
s_1
s_2
s_3
$\mathcal{N}(t_0; s_0 + s_1, \beta^2)$
$\mathcal{N}(t_0; s_2 + s_3, \beta^2)$
t_0
t_1
$\mathbb{I}(t_0 > t_1)$
y

Figure 1: Original TrueSkill$^{\text{TM}}$model for a game with two teams of two members each. Note that, in the above I only show a small part of the full graph, related to a single game. In reality, we have N players and G games.

We are interested in computing the marginal distribution, $\mathbb{P}(s_i|\mathbf{s}, \mathbf{t}, \mathbf{y})$ for each player i. We can do these by applying the sum product algorithm. The derivation of the message passing equation is an exercise in *Gaussian* distribution manipulations [5]. Notice however that we need to make two further approximations.

Firstly, because of the presence of cyclic structures (since the same player can take part in multiple games), we will have to make use of the *loopy belief propagation* ideas and iterate a few times through the steps of the algorithm.

Secondly, the efficiency of *expectantion propagation* is based on having to propagate few moments through the network. In the case of the *normal distribution*, two moments suffice (μ and σ). However, for the factor computing $\mathbb{I}(t_0 > t_1)$, we can apply *moment matching* to approximate its result to a *Gaussian*. This reduces to applying *moment matching* on a truncated *Gaussian*, say: $t \sim \frac{1}{Z_t}\delta(t > 0)\mathcal{N}(t; \mu, \sigma^2)$. By using the notation:

$$\Psi(z) = \frac{\mathcal{N}(z; 0, 1)}{\Phi(z)}$$
$$\Lambda(z) = \Psi(z)(\Psi(z) + z),$$

We can approximate the distribution of t by a Gaussian whose moments are [3]:

$$\mathbb{E}(t) = \mu + \sigma\Psi(\frac{\mu}{\sigma})$$
$$\text{Var}(t) = \sigma^2(1 - \Lambda(\frac{\mu}{\sigma}))$$

[3] http://mlg.eng.cam.ac.uk/teaching/4f13/1718

Notice that under this model, the only factors that are open to modification are the prior distribution of the skills and the outcome of the game. There is no other way to incorporate new assumptions that are valid in particular games we apply this model to. In the following, I will present a few simple modification that could better fit the game of DotA2, as well as validate them on syntesized and real datasets.

2.4 DotA2

Dota2 (Defence of the Ancients 2) is a Multiplayer Online Battle Arena(MOBA). The *Arena* is a square map containing two adversary bases on opposite corners, united by three lanes heavily guarded by turrets. There are two teams, refered to as the *Radiance* and *Dire*, each of which is composed of five human players. The goal of a team is to infiltrate the enemy base and destroy their *Ancient*, a static structure. Upon achieving this feat, the game terminates.

Each human player controls a *hero* which they can chose before the game commences. The *heros* are all difference, requiring particular strategies. During clashes between teams, *heros* might get killed, which confers their slayers *gold* and *experience*. Upon death, *heros respawn* after a penalty time during which the player can only observe the game.

Although this is an incomplete description of the game[4] , it suffices for understanding the metrics provided by specialised datasets, such as: duration of the game, hero name, kills, deaths, gold per minute, xp per minute etc.

[4]https://dota2.gamepedia.com/Dota_2_Wiki

References

[1] Arpad E Elo. *The rating of chessplayers, past and present.* Arco Pub., 1978.

[2] Mark E Glickman. A comprehensive guide to chess ratings. *American Chess Journal*, 3:59–102, 1995.

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

[4] Mark E Glickman. Example of the glicko-2 system. *Boston University*, 2012.

[5] Ralf Herbrich, Tom Minka, and Thore Graepel. TrueskillâĎć: a bayesian skill rating system. In *Advances in neural information processing systems*, pages 569–576, 2007.

[6] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[7] Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

[8] Judea Pearl. *Reverend Bayes on inference engines: A distributed hierarchical approach.* Cognitive Systems Laboratory, School of Engineering and Applied Science, University of California, Los Angeles, 1982.

[9] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Elsevier, 2014.

[10] Lei Zhang, Jun Wu, Zhong-Cun Wang, and Chong-Jun Wang. A factor-based model for context-sensitive skill rating systems. In *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, volume 2, pages 249–255. IEEE, 2010.