Chapter 3

TrueSkill™ Rating System

TrueSkill [3] is a Gaussian rating system developed by Microsoft Research in 2007 to rank players on Xbox Live for the purpose of creating competitive matches amongst players with similar skill levels. While the Elo rating system is applicable to only two-player matches, the TrueSkill ranking system extends the use cases to multi-player matches. Section 1.1 describes Approximate Bayesian Inference — the mathematical backbone of the TrueSkill rating system and Section 1.2 discusses the prediction performances of several models built upon an open-source implementation of the TrueSkill rating system.

3.1 Approximate Bayesian Inference

The approach described in Section 2.2 can be modelled as a Bayesian Inference process. Define a vector $\boldsymbol{s} = [s_1, s_2, \ldots, s_n]^\intercal$ representing the individual *skill* of n teams in a match (in the case of NCAA basketball tournament, $n = 2$). Further assume that each skill follows a Gaussian distribution $s_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and collectively, the *prior* joint distribution of skills is a n-variate Gaussian distribution $p(\boldsymbol{s}) = \mathcal{N}_n(\boldsymbol{s}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

In a match, each team is expected to exhibit a certain *performance* $p_i \sim \mathcal{N}(s_i, \beta_i^2)$ that varies around its skill. The match result can be represented as a set of rankings in ascending order for each team, namely, $\boldsymbol{r} = [r_1, r_2, \ldots, r_n]^\intercal$ where $r_1 \leq r_2, \ldots, r_{n-1} \leq r_n$. In reality, there are many factors that may affect the outcome of a match, but for the purpose of Bayesian Inference, the assumption is that it is the differences in team performances that cause the differences in team rankings. According to the Bayes' Theorem, the posterior distribution $p(\boldsymbol{s}|\boldsymbol{r})$ is given by

$$p(\boldsymbol{s}|\boldsymbol{r}) = \frac{P(\boldsymbol{r}|\boldsymbol{s})p(\boldsymbol{s})}{P(\boldsymbol{r})} = \frac{P(\boldsymbol{r}|\boldsymbol{s})p(\boldsymbol{s})}{\int P(\boldsymbol{r}|\boldsymbol{s})p(\boldsymbol{s})d\boldsymbol{s}} \tag{3.1}$$

The above formula demonstrates the use of Bayes' Theorem to calculate the updated skill distribution (posterior) given the match outcome (evidence) and the expectations (likelihood and prior) on the match outcome before the match. However, the posterior distribution is no longer a Gaussian distribution and oftentimes, the integral is intractable. A common solution is to calculate a Gaussian distribution $\mathcal{N}_n(\boldsymbol{s}; \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$ as an approximation to the posterior distribution $p(\boldsymbol{s}|\boldsymbol{r})$ so that the Kullback-Leibler divergence between the two distributions is minimised.

A number of approximation techniques are available. For the TrueSkill rating system, it approximates the posterior distribution by setting up a *factor graph* which is a bipartite graph containing two kinds of vertices: variables and factors. A variable stores some value of interests and a factor represents some operations on one or more variables. The Bayesian Inference process is modelled as passing some *message*s, which are some real-valued functions of a variable or a factor, throughout the factor graph. Figure 1-1 shows an example factor graph of a simple two-player match. Based on skill s_i and performance p_i, it calculates the expected performance difference $p_1 - p_2$ and compares that with an externally supplied outcome d. The $\mathbb{I}$ denotes an indicator function to assert whether d is consistent with the expectation, depending on which a different update is performed on s_1 and s_2.

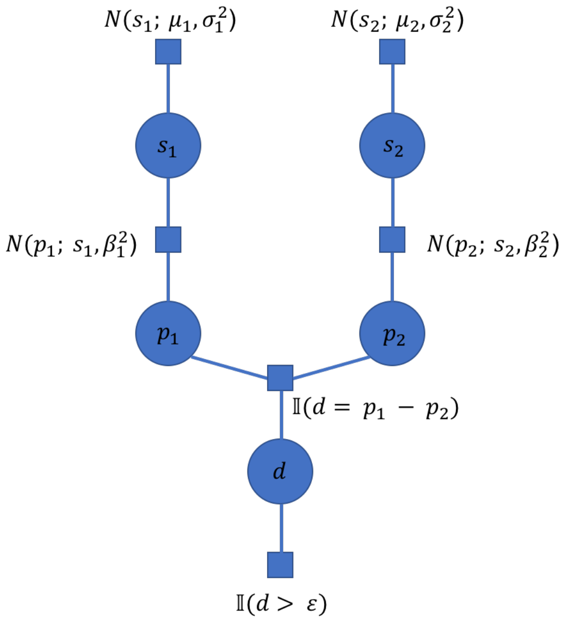
$N(s_1;\, \mu_1, \sigma_1^2)$
$N(s_2;\, \mu_2, \sigma_2^2)$
s_1
s_2
$N(p_1;\, s_1, \beta_1^2)$
$N(p_2;\, s_2, \beta_2^2)$
p_1
p_2
$\mathbb{I}(d = p_1 - p_2)$
d
$\mathbb{I}(d > \varepsilon)$

Figure 3-1: An example of factor graph

The ultimate goal of a factor graph is to obtain a set of update equations that indicate how each μ_i and σ_i should be updated. However, passing messages throughout larger factor graphs can involve very intensive computations. As suggested by [1], there is an alternative but easier way of getting the same set of update equations based the following theorem.

Theorem 1. *Let $\boldsymbol{z}$ be a k-dimensional random vector $[z_1, z_2, \ldots, z_k]^\intercal$ with a probability distribution function of the form*

$$\frac{\phi_k(\boldsymbol{z})f(\boldsymbol{z})}{\int \phi_k(\boldsymbol{z})f(\boldsymbol{z})d\boldsymbol{z}} \tag{3.2}$$

where ϕ_k is a k-variate standard Gaussian distribution function. Then, the expectation of $\boldsymbol{z}$ is given by

$$E[\boldsymbol{z}] = E\Big[\frac{\nabla f(\boldsymbol{z})}{f(\boldsymbol{z})}\Big] \tag{3.3}$$

and also

$$E[z_i z_j] = \delta_{ij} + E\Big[\frac{\nabla^2 f(\boldsymbol{z})}{f(\boldsymbol{z})}\Big] \tag{3.4}$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise.

To link Equation 1.2 with the Bayes' Theorem in Equation 1.1, notice that the match outcome is dependent on performances and performances are related to skills; therefore, the likelihood probability $P(\boldsymbol{r}|\boldsymbol{s})$ can be expressed as some function of skill $f : \boldsymbol{s} \to [0, 1]$. Furthermore, since each $s_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, define the z-score vector $\boldsymbol{z} = [z_1, z_2, \ldots, z_n]^\intercal$ where

$$z_i = \frac{s_i - \mu_i}{\sigma_i} \tag{3.5}$$

Therefore, the probability distribution $p(\boldsymbol{z}) = \phi_n(\boldsymbol{z})$ and the likelihood probability $P(\boldsymbol{r}|\boldsymbol{s})$ is also a function of $\boldsymbol{z}$. Then according to Equation 1.1,

$$p(\boldsymbol{z}|\boldsymbol{r}) = \frac{P(\boldsymbol{r}|\boldsymbol{z})p(\boldsymbol{z})}{P(\boldsymbol{r})} = \frac{\phi_n(\boldsymbol{z})f(\boldsymbol{z})}{\int \phi_n(\boldsymbol{z})f(\boldsymbol{z})d\boldsymbol{z}} \tag{3.6}$$

and therefore by Theorem 1, the values of interest μ_i^{new} and σ_i^{new} are related to $E[z_i]$ through

$$\begin{aligned} \mu_i^{new} &= E[s_i] = E[\mu_i + \sigma_i z_i] = \mu_i + \sigma_i E[z_i] \\ &= \mu_i + \sigma_i E\Big[\frac{\partial f(\boldsymbol{z})/\partial z_i}{f(\boldsymbol{z})}\Big] \end{aligned} \tag{3.7}$$

and

$$\begin{aligned}\sigma_i^{new} &= Var[s_i] = \sigma_i^2 Var[z_i] = \sigma_i^2(E[z_i^2] - E[z_i]^2) \\ &= \sigma_i^2\Big(1 + E\Big[\frac{\nabla^2 f(\boldsymbol{z})}{f(\boldsymbol{z})}\Big]_{ii} - E\Big[\frac{\partial f(\boldsymbol{z})/\partial z_i}{f(\boldsymbol{z})}\Big]^2\Big)\end{aligned} \tag{3.8}$$

The above update equations are applicable to the general cases where n teams are involved. However, for NCAA tournament matches that are between only two teams, a simpler version of the update equation can be derived. Now consider two teams with skill $\boldsymbol{s} = [s_i, s_j]^\intercal$. In general, there is no draw in basketball matches, but for completeness, assume the *draw margin* is ε such that team i beats team j when $s_i - s_j > \varepsilon$ and the outcome is considered as draw when $|s_i - s_j| \leq \varepsilon$. Furthermore, define $c^2 = 2\beta^2 + \sigma_i^2 + \sigma_j^2$. Then

$$f(\boldsymbol{z}) = \Phi\Big(\frac{\mu_i - \mu_j - \varepsilon}{c}\Big) \tag{3.9}$$

when team i beats team j. Alternatively,

$$f(\boldsymbol{z}) = \Phi\Big(\frac{\varepsilon - (\mu_i - \mu_j)}{c}\Big) - \Phi\Big(\frac{-\varepsilon - (\mu_i - \mu_j)}{c}\Big) \tag{3.10}$$

when team i draws with team$j$. In both cases, Φ represents the standard Gaussian cumulative distribution function.

Another approximation applied above is to assume $\boldsymbol{z} = \boldsymbol{0}$, namely, the two teams are assumed to have skills that are equal to their mean skills ($\boldsymbol{s} = \boldsymbol{\mu}$). The rationale for the above equations is to construct a Gaussian difference distribution [5] $\mathcal{N}_{i-j}(s_{i-j}; \mu_i - \mu_j, c^2)$ considering the performance variance β^2. Then for the case that team i beats team j, it is to calculate the probability that $\mu_i - \mu_j - \varepsilon > 0$ using a standard Gaussian Φ function. Similar arguments apply to the draw case.

Now that $f(\boldsymbol{z})$ is known, for the case team i beats team j

$$\begin{aligned}\frac{\partial f(\boldsymbol{z})}{\partial z_i} &= \frac{\partial f(\boldsymbol{z})}{\partial \mu_i} \cdot \frac{\partial \mu_i}{\partial z_i} = \sigma_i \frac{\partial}{\partial \mu_i}\Phi\Big(\frac{\mu_i - \mu_j - \varepsilon}{c}\Big) \\ &= \frac{\sigma_i}{c}\phi\Big(\frac{\mu_i - \mu_j - \varepsilon}{c}\Big)\end{aligned} \tag{3.11}$$

where ϕ is the standard Gaussian probability density function. Following Equation 1.7

and 1.8 , the skill of team i can be updated accordingly:

$$\mu_i^{new} = \mu_i + \frac{\sigma_i^2}{c} \cdot \frac{\phi\left(\frac{\mu_i - \mu_j - \varepsilon}{c}\right)}{\Phi\left(\frac{\mu_i - \mu_j - \varepsilon}{c}\right)} \tag{3.12}$$

$$\sigma_i^{new} = \sigma_i^2 \Bigg\{ 1 + \frac{\sigma_i^2}{c} \Bigg[- \frac{\mu_i - \mu_j - \varepsilon}{c^2} \cdot \frac{\phi\left(\frac{\mu_i - \mu_j - \varepsilon}{c}\right)}{\Phi\left(\frac{\mu_i - \mu_j - \varepsilon}{c}\right)} - \frac{1}{c} \cdot \frac{\phi\left(\frac{\mu_i - \mu_j - \varepsilon}{c}\right)^2}{\Phi\left(\frac{\mu_i - \mu_j - \varepsilon}{c}\right)^2} \Bigg] \Bigg\} \tag{3.13}$$

Detailed derivations are omitted and can be found in [1]. Update equations for team j and for the case of draw can be derived in a similar way. By means of Approximate Bayesian Inference, the TrueSkill rating system is able to update the skill of each team upon the end of a match and to estimate a team's probability of winning an upcoming match.

3.2 Kaggle Results & Discussions

Section 2.1.1 lists many potential features that can be used as inputs to a predictive model. However, the required inputs to a TrueSkill rating system are very simple — the winning and losing team for a particular match. It does not even take into consideration by how many points the winning team leads the losing team, which is also known as margin of victory (MOV).

But some other features may provide valuable insights into the "true skill" of a team. For example, the feature 'Wloc', whether the winning team was at its home court, a neutral site or the losing team's court, indicates whether the winning team was granted the "Home Court Advantage (HCA)". HCA can sometimes improve a team's *performance* significantly and lead to a victory even if the "true skill" of that team is lower than its opponent's.

Since TrueSkill does not accept these features as inputs directly, to incorporate them into the predictive models, their magnitudes are converted into some effects on the MOV of a match. For instance, HCA may be worth five points in a basketball game and therefore, five points are deducted from the winning team's final score, which may possibly result in a *reverse* of the match outcome. Adjustments of this kind are carried out before the match records are fed to TrueSkill. The final features used in this project are summarised in Table 1.1.

A larger MOV may be a good indicator of a larger skill gap between the winning team and the losing team. In this project, the effect of MOV is added iteratively: as long as the MOV is larger than some threshold value m, the match is considered a

Feature	**Possible Effects**
Wteam	Identifies the winning team
Wscore	Identifies the winning team's score
Lteam	Identifies the losing team
Lscore	Identifies the losing team's score
MOV	The larger the MOV, the larger the skill gap
Wloc	Identifies whether HCA exists
Numot	The more overtimes, the smaller the skill gap
Wto	The less turnovers, the stronger the skill

Table 3.1: A Summary of the Selected Features and Effectives

victory for the winning team and fed to TrueSkill, after which the MOV is deducted by m and the process continues until MOV $< m$. Therefore, it is possible that an actual match is treated as *multiple* victories because the match has a very large MOV. Although TrueSkill does not consider MOV, the *number* of victories does matter in TrueSkill.

Occasionally, there is overtime in a basketball match if two teams tie at the end of the normal hours. According to the rules of basketball, additional overtime must be given if two teams still tie after the last overtime. In the case of multiple overtime, it is true that there will be a winner eventually, but the fact that the first overtime happens already provides some important information about the two teams' skill levels — they should have close skill levels that cause overtime. In this project, whenever overtime occurs in a match, regardless of how many times it occurs, the scores of the two teams are made equal to reflect this intuition.

Turnovers can be thought as mistakes a team made in a match. There are many possible factors from both the team and its opponents that can cause turnovers, such as a double dribble or the ball being stolen. Therefore, it is a very comprehensive feature that may contain information from many other invisible factors. In this project, if a team commits more turnovers than its opponents', then its final score will be deducted by some variable amounts.

Table 1.2 summaries all the models built in this project and their respective ranks on Kaggle's leaderboard.

Model No.	**Feature Used**	**LogLoss**	**Rank**	**Top**
1	Naïve TrueSkill	0.531040	13th	2.17%
2	MOV	0.538214	34th	5.68%
3	HCA	0.513445	5th	0.83%
4	Overtime	0.514140	5th	0.83%
5	Turnover	0.503654	3rd	0.50%

Table 3.2: A summary of the final results

Model No.	**Feature Used**	**LogLoss**	**Rank**	**Top**
1	Naïve TrueSkill	1.542861	585th	97.66%
2	MOV	2.046105	593rd	99.00%
3	HCA	0.505484	3rd	0.50%
4	Overtime	0.504135	3rd	0.50%
5	Turnover	0.493128	2nd	0.33%

Table 3.3: A summary of the final results