Université du Québec
Institut National de la Recherche Scientifique
Énergie Matériaux Télécommunications

SYSTÈMES BAYÉSIENS DE NOTATION: ANALYSE ET AMÉLIORATIONS

Par

Amen Memmi

Mémoire présenté pour l'obtention du grade de
Maître es Sciences, M.Sc.
en télecommunications

Jury d'évaluation

Examinateur externe	Felipe Tobar Professeur à l'Université de Chili
Examinateur interne	Jean-Charles Grégoire Professeur à l'INRS EMT
Directeur de recherche	Long Le Professeur à l'INRS EMT
Codirecteur de recherche	Leszek Szczecinski Professeur à l'INRS EMT

Remerciements

Je voudrais exprimer ma plus profonde gratitude à mes superviseurs, les Professeurs Long Le et Leszek Szczecinski. Leurs conseils et encouragements ont fait du travail une expérience agréable et extrêmement éducative.

Je tiens à remercier particulièrement Pr. Szczecinski pour m'avoir guidé tout au long de mes études de maîtrise. Sa porte m'était toujours ouverte pour m'écouter, discuter et partager ses commentaires et pensées perspicaces. Ce sont ces commentaires et ces pensées qui m'ont gardé sur la bonne voie et qui ont fini par m'aider à trouver les solutions et les nouvelles idées.

J'aimerais également remercier mes collègues du laboratoire pour les moments merveilleux et les discussions intéressantes que nous avons eues au cours des deux années passées. Leur compagnie et leur camaraderie ont considérablement enrichi mon expérience au centre EMT de l'INRS.

Je profite également de l'occasion pour remercier mon meilleur ami Mohamed Amine Arfaoui pour son aide, ses encouragements et son soutien continus.

Plus que quiconque, je voudrais remercier ma famille bien-aimée. C'est leur soutien et leurs encouragements continus qui ont rendu ce travail possible.

Résumé

Les systèmes de notation continuent de faire l'objet d'un intérêt croissant, en particulier après la popularité croissante des jeux multijoueur en ligne. La plupart de ces derniers basent le jumelage des joueurs sur leurs compétences afin de leur donner des matchs équitables et agréables. L'estimation de ces compétences est réalisée à travers un système de notation qui les déduit à partir des données historiques. Le système de notation bayésien TrueSkill™, développé il y a quelques années par Microsoft, fournit un modèle probabiliste pour cette estimation dans un cadre général de compositions et de nombre d'équipes et ce seulement à partir du résultat binaire de chaque match. Dans ce travail, on présente et on analyse le comportement de ce système dans des situations particulières et on se propose ensuite de l'étendre en incorporant les différentes statistiques disponibles relatives au déroulement des jeux, telles que le nombre de kills, les dégâts moyens reçus, le nombre de matchs déjà joués et plein d'autres caractéristiques. On montre que l'extension et les modifications apportées permettent des gains importants capturés par les multiples métriques d'évaluation introduites. Entre autres, la corrélation des mises à jour des compétences par ces statistiques a permis au système des prédictions plus précise, ayant un plus grand gain d'information, plus équitable en relation avec les performances individuelles et mieux représentative des compétences moyennes de certaines sous populations de joueurs définies à travers les statistiques.

Mots-clés Système de notation, compétence, inférence Bayésienne, propagation de messages, statistiques du jeu

Abstract

Rating systems continue to attract increasing interest, especially after the growing popularity of online multiplayer games. Most of the latter base the matching of players on their skills in order to give them fair and enjoyable matches. These skills are estimated using a rating system that derives them from historical data. The TrueSkill™Bayesian rating system, developed a few years ago by Microsoft, provides a probabilistic model for this estimation within a general framework of teams composition and number and inference is done based only on the binary result of each match. In this work, we present and analyze the behavior of this system in particular situations and we then propose to extend it by incorporating the various available in-game statistics, such as the number of kills, the average damage received, the number of matches already played and many other characteristics. We show that the extension and the modifications made allow significant gains captured by the multiple metrics of evaluation introduced. Among other things, the correlation of skills updates by these statistics has allowed the system to make predictions that are more accurate, having a greater gain of information, more equitable in relation to individual performances and with better representative capacity of the average skills of certain statistics-defined sub-populations.

Keywords Rating system, skills, bayesian inference, message passing, in-game statistics

Table des matières

Remerciements iii

Résumé v

Abstract vii

Table des matières ix

I Version française xiii

Liste des figures xv

Liste des tableaux xvii

List of Symbols xix

1 Introduction 1
- 1.1 Contexte et préliminaires . . . 1
  - 1.1.1 Notation et classement : Concepts, objectif et motivation . . . 1
  - 1.1.2 Évolution des systèmes de notation . . . 2
  - 1.1.3 Littérature récente - travaux connexes . . . 4
- 1.2 Le projet de maîtrise . . . 5
  - 1.2.1 Objectifs . . . 5
  - 1.2.2 Outline . . . 6

2 Modèle du système et données utilisées 7
- 2.1 Modèle du système . . . 7
- 2.2 Données . . . 9
  - 2.2.1 Données synthétiques . . . 9
  - 2.2.2 Données réelles du monde eSports . . . 9

3 TrueSkill, présentation et analyse 11
- 3.1 Le modèle de notation TrueSkill^{TM} . . . 11
- 3.2 Étude de cas synthétique . . . 15

4 TrueSkill 1.1, les améliorations proposées 19
- 4.1 Utilisation d'informations supplémentaires relatives au déroulement du jeu . . . 19
- 4.2 Utilisation des scores - Modèle de compétences offensives et défensives . . . 20

4.3 Features et statistiques relatives au déroulement du Jeu . . . 23
4.3.1 Compétences partielles . . . 23
4.3.2 Modèle de statistiques individuelles . . . 25
4.3.3 Aggrégation des compétences partielles . . . 28
4.4 Récompense de la pratique . . . 29
4.5 TrueSkill Through Time et traitement batch . . . 32

5 Résultats et analyse **33**
5.1 Métriques d'évaluation . . . 33
5.2 Résultats de simulation et discussion . . . 35

6 Conclusion et extensions possibles **41**

II English Version **45**

List of figures **47**

List of tables **49**

List of Symbols **51**

1 Introduction **53**
1.1 Background and Preliminaries . . . 53
1.1.1 Rating and Ranking: Concepts, Purpose and Motivation . . . 53
1.1.2 Evolution of Rating Systems . . . 54
1.1.3 Recent Literature - Related works . . . 56
1.2 Master Thesis Project . . . 57
1.2.1 Objectives . . . 57
1.2.2 Outline . . . 58

2 System Model and Used Data **59**
2.1 System Model . . . 59
2.2 Data . . . 61
2.2.1 Synthetic Data . . . 61
2.2.2 Real-life e-Sports Data . . . 61

3 TrueSkill Presentation and Analysis **63**
3.1 The TrueSkill^{TM} Rating Model . . . 63
3.2 Synthetic Study Case . . . 66

4 TrueSkill 1.1, the Proposed Improvements **71**
4.1 Use of in-Game Additional Information . . . 71
4.2 Score Use, Offensive and Defensive Skills Model . . . 72
4.3 In-Game Statistics and Features . . . 75
4.3.1 Partial Skills . . . 75
4.3.2 Individual Stats Model . . . 76
4.3.3 Aggregation of Partial Skills . . . 79
4.4 Practice Reward . . . 81
4.5 TrueSkill Through Time and Batch Treatment . . . 83

5 Results and analysis **85**
5.1 Evaluation metrics 85
5.2 Simulation results and discussion 87

6 Conclusions and Future Work **93**

Références **97**

A Métrique de récompense de pari **99**

B Betting Reward Metric **101**

Première partie

Version française

Liste des figures

2.1 Tâche de notation 8

3.1 Exemple de graphe de facteur de TrueSkill pour deux équipes. Un joueur forme l'équipe bleue et l'équipe rouge comporte deux joueurs. 14
3.3 Le graphe de facteur pour la configuration batch. 16
3.2 Un exemple de graphe de facteur simplifié pour un simple match à deux. 16

4.1 Graphe de facteur de TrueSkill-basé-sur-les-scores. 21
4.2 Graphe de facteur de l'offensive de l'équipe i vs la défense de l'équipe j dans une configuration de 5 joueurs par équipe. 24
4.3 *Gold* collecté vs durée du match pour les joueurs avec un haut niveau de compétence en LoL. 25
4.4 Graphes de facteur d'inférence à partir d'un score individuel 26
4.5 Les poids traduisant l'importance de chaque feature 28
4.6 Configuration multi-couches 30
4.7 Probabilité de victoire en fonction du nombre de matchs joués précédemment. 30
4.8 Effet de la récompense de la pratique - CS:GO 31

5.1 Probabilité de victoire vs features 37

3.1 Simplified 1vs1 match factor graph 67

Liste des tableaux

3.1 Liste des matches dans l'ordre chronologique 15

4.1 Comparaison des compétences offensives entre les deux approches 27

5.1 Comparaison des métriques des deux systèmes 36
5.2 Comparaison des gains de paris des deux systèmes 36
5.3 Probabilités de victoire réelles et espérées pour les différentes sous-populations relatives à chaque feature 38
5.4 Effet de la mise à jour batch hors ligne 39

Liste des symboles

$\mathcal{N}(\mu, \sigma^2)$	La distribution normale avec la moyenne μ et la variance σ^2
$\tilde{\mathcal{N}}(\tau, \pi)$	La distribution normale avec la précision π et la moyenne ajustée τ
$\mathbb{1}(.)$	La fonction indicatrice
x_i	variable aléatoire relative au joueur i
$\mathbf{x}$	vecteur de variables aléatoires
μ_x	Moyenne de la variable aléatoire normale x
σ_x^2	Moyenne de la variable aléatoire normale x
$\Phi(.)$	la fonction de répartition d'une variable aléatoire normale centrée réduite
$\Pr\{A\}$	Probabilité de l'évènement A
$f(x)$	Distribution de la variable aléatoire x

Listes des termes anglais originaux et leurs traductions françaises utilisées dans ce document:

Anglais	**Français**
Ranking	Classement
Rating	Notation
Matchmaking	Jumelage
Message passing	Propagation des messages
Expectation propagation	Propagation d'espérance
Scoring rule	Règle de score
Batch	Par lot
Online update	Mise à jour en ligne

Chapitre 1

Introduction

1.1 Contexte et préliminaires

1.1.1 Notation et classement : Concepts, objectif et motivation

La concurrence est ancrée dans la nature humaine. Qu'il s'agisse de la compétition de frères, d'étudiants, d'hommes d'affaires ou d'athlètes, tous s'efforcent de faire mieux que les autres. Il n'est pas étonnant que nous nous mesurions constamment par rapport à nos pairs. Nous devons être capables de comparer et de contraster les capacités des concurrents pour établir qui est le meilleur, d'où la nécessité d'évaluer et de classer les concurrents. Les compétitions sportives représentent l'un des domaines qui attirent le plus grand besoin de classement. Cela inclut les sports physiques traditionnels et, plus récemment, les sports électroniques ou eSports. En général, les mesures directes des capacités ne sont pas disponibles. La seule information que nous possédons est structurée comme une quantité de résultats et de statistiques qui sont affectés par ces compétences sous-jacentes. Le résultat d'un match est déterminé par un grand nombre de facteurs. Garder la trace de tous les facteurs pertinents tend à être irréaliste. Si l'on ajoute à cela l'incertitude liée à la mesure de ces facteurs et de leur éventuelle variabilité non déterministe, on ne peut s'attendre à une certaine prédiction déterministe d'un résultat de match. Au contraire, une bonne prédiction, en fait, ne prédira pas exactement le résultat, mais anticipera plus précisément les cotes et les probabilités. Cette incertitude comprend également la modélisation des compétences des concurrents. Les statistiques et la théorie des probabilités fournissent des moyens de faire des

inférences dans ce cadre de caractère aléatoire. Par conséquent, la modélisation et la prédiction des résultats des matchs d'équipes compétitifs relève naturellement du domaine des statistiques et de l'apprentissage de machine. Les systèmes de notation et d'évaluation servent deux objectifs principaux : l'estimation des compétences pour le classement et le jumelage (*matchmaking*), ainsi que la prédiction des cotes de futures rencontres. Le classement des joueurs est un objectif en soi, mais il sert aussi à faire jumeler et correspondre des joueurs de niveaux similaires afin de leur assurer des matchs amusants et excitants: Un jeu est le plus amusant lorsque le résultat est le plus incertain, avec des chances a priori de gagner égales à celles de perdre. D'autre part, prédire les chances de gagner est également très important : Les systèmes de notation pourraient être rentables lorsque des prédictions correctes sont récompensées, comme parier sur les gagnants d'un match compétitif. Un système de notation précis qui peut prévoir mieux que d'autres parties peut certainement faire des profits. Par conséquent, le marché des paris est vraiment intéressé par de tels systèmes, tant pour les organisations de paris que pour les joueurs participants.

1.1.2 Évolution des systèmes de notation

Pour de nombreux sports, le système de classement le plus important utilisé de nos jours est probablement le système Elo (ou l'une de ses versions modifiées). Ce système a été développé par Arpad Elo au début des années soixante du siècle précédent, néanmoins la mise à jour séminale d'Elo reste, après plus de 50 ans, une véritable base de référence qui est difficile à améliorer. L'Elo original est basé sur un modèle statistique qui utilise le modèle Thurstone-Mosteller pour estimer la probabilité de résultats individuels, basé sur l'hypothèse que la performance du joueur dans chaque jeu est une variable aléatoire qui est normalement distribuée avec la même variance constante, tout en supposant que la vraie compétence de chaque joueur est la moyenne de ses performances. La version logistique du système remonte aux travaux de Zermelo [1], qui a développé un modèle de comparaison par paires qui est devenu plus tard le modèle Bradley-Terry. Cette dernière version est la plus populaire. A l'origine, Elo a développé son modèle pour le jeu d'échecs, et les fédérations d'échecs du monde entier l'ont rapidement adopté (1970). Il est devenu populaire et commun pour beaucoup d'autres jeux de deux joueurs aussi, y compris Go, Scrabble, le ping-pong, le football américain, le basket-ball, le baseball au Major League ...

Glickman, en 1999, a proposé le système de notation Glicko, qui améliore Elo en incorporant la variabilité dans les estimations des paramètres, ce qui fait de lui le premier système de classement bayésien. Pour commencer, avant une période de classement, la compétence d'un joueur (q) est supposée suivre une distribution gaussienne qui peut être caractérisée par deux nombres : la compétence moyenne du joueur (μ) et le degré d'incertitude dans la compétence du joueur (s). Ensuite, Glicko modélise les résultats du jeu selon le modèle Bradley-Terry et met à jour les compétences des joueurs après une période de classement. Bien que les systèmes de classement Elo et Glicko aient été couronnés de succès, ils sont conçus pour les jeux à deux joueurs. Dans les jeux vidéo en ligne, un jeu implique souvent plus de deux joueurs ou équipes. Pour résoudre ce problème, Microsoft Research a développé TrueSkill[2], un système de classement pour Xbox Live.

Tout comme Glicko, TrueSkill est un système de classement bayésien et il utilise une croyance gaussienne sur les compétences d'un joueur, mais il y présente quelques différences par rapport à Glicko. Il s'agit d'un système d'apprentissage en ligne qui met à jour les compétences après chaque match plutôt que d'attendre la fin d'une période d'évaluation. De plus, Glicko modélise la différence de performance par une distribution logistique (modèle Bradley-Terry), tandis que TrueSkill utilise la distribution gaussienne (modèle Thurstone-Mosteller). De plus, TrueSkill supporte les matchs nuls et offre un moyen de mesurer la qualité d'un jeu. TrueSkill estime les compétences en construisant un modèle graphique et en utilisant le passage approximatif des messages. Dans le cas le plus simple, un jeu à deux équipes, les règles de mise à jour de TrueSkill sont assez simples. Cependant, pour les parties avec plusieurs équipes et plusieurs joueurs, les règles de mise à jour sont plus compliquées et nécessitent une procédure itérative. Le nombre de matchs requis pour que l'algorithme TrueSkill soit suffisamment sûr de la compétence dépend de la composition de l'équipe et son développeur donne des valeurs empiriques approximatives pour les configurations les plus populaires. TrueSkill a été étendu plus tard pour estimer les compétences des joueurs non seulement vers l'avant à travers le temps mais aussi vers l'arrière, permettant aux informations futures d'ajuster le classement passé du joueur en revenant en arrière dans l'estimation[3]. Ils ont appelé cette extension TrueSkill Through Time, TTT. Bien que TrueSkill supporte les configurations d'équipes, la TTT n'a été appliquée et testée que pour deux joueurs. Les auteurs de TTT ont affirmé que leur nouvel algorithme est plus précis, ce qui justifie le temps d'estimation plus long nécessaire pour avancer ou reculer dans le temps. TrueSkill a été ensuite sujet d'une autre extension, notamment celle incorporant les scores des équipes [4].

1.1.3 Littérature récente - travaux connexes

La plupart des systèmes de notation précédents avaient en commun les données d'entrée : ils fondent leurs estimations et leurs prédictions sur les résultats binaires des matchs. Avec l'énorme quantité de données de jeu disponibles, on peut se demander s'il est rentable pour ces systèmes de tenir compte de ces données afin de s'améliorer. Le fait de baser le classement uniquement sur les résultats des matchs pourrait être considéré comme efficace puisqu'il fournit déjà de bons résultats et qu'il s'applique à tous les matchs de la même façon. Cependant, cela pourrait avoir comme conséquence de négliger et se passer de plusieurs aspects pertinents que ces systèmes ne sont pas conçus pour capturer, comme dans les jeux multijoueurs récents. Un exemple est de ces dernier sont les jeux de Multiplayer Online Battle Arena (MOBA). Ce genre de jeux offre une grande variété de paramètres et de statistiques relatives aux joueurs et aux équipes qui pourraient bien être incorporés dans la modélisation de la compétence.

En parcourant la littérature récente, on peut voir que la modélisation des compétences est traitée dans le cadre de la modélisation des joueurs, qui a été longtemps étudié par des chercheurs tels que Yannakakis et al. dans [5] et Bakkes et al. dans [6]. Dans la modélisation des joueurs, les caractéristiques des joueurs humains telles que les stratégies, les préférences et les compétences sont détectées, modélisées, prédites et exprimées. La modélisation des compétences diffère de leur évaluation dans le sens qu'elle vise à englober les compétences des joueurs dans de multiples facettes sans nécessairement classer les joueurs. Stanescu dans son mémoire de maîtrise [7] a modélisé les compétences des joueurs dans plusieurs dimensions (comme les capacités offensives et défensives) mais son travail se limitait à des jeux de 1 contre 1 plutôt qu'à des jeux d'équipe comme MOBA. Parmi ces recherches, seules quelques-unes comme Roy et al. [9] et Rahman et al. [8] ont tendance à étudier les compétitions en équipe. Pour évaluer et vérifier la cohérence des composantes de compétences proposées, la procédure courante consiste à se fier à la précision des prédictions et à la log-vraisemblance du prédicteur. Il y a eu quelques travaux qui ont traité ce sujet en particulier pour les jeux MOBA. Pobiedina et al. [10][11] a montré que quatre facteurs contribuent au succès de l'équipe, à savoir la composition de l'équipe de champions, le nombre d'amis, l'expérience du joueur et la conformité des nationalités des joueurs. Dans [12], les auteurs utilisent divers modèles prédictifs basés sur les compétences pour décomposer les compétences des joueurs en parties interprétatives. En adoptant une approche d'analyse basée sur le modèle, ils constatent que les compétences des joueurs dans les jeux MOBA pourraient inclure les compétences de base du joueur, les compétences

de base du champion et les compétences spécifiques du champion du joueur comme trois composantes importantes. Cela encourage l'idée d'analyser la compétence à partir d'angles multiples et de la traiter en corrélation avec les différentes caractéristiques et statistiques disponibles dans le jeu.

1.2 Le projet de maîtrise

1.2.1 Objectifs

Dans le cadre du projet actuel, on s'intéresse à la réalisation des objectifs que nous présentons dans les cinq points ci-dessous :

Compréhension Assimilation du mécanisme d'inférence bayésienne en utilisant le passage de message gaussien dans les systèmes de notation basés sur des modèles graphiques, à savoir TrueSkill : analyse de la convergence et de l'évolution des paramètres, comparaison au cas parfait et proposition de modifications possibles.

Modélisation Se baser sur le modèle existant pour exprimer les compétences des joueurs et l'étendre pour inclure différentes caractéristiques disponibles au jeu tout en visant une inférence précise.

Collection de Données Acquérir des ensembles de données appropriés pour l'évaluation contenant un nombre suffisant de matchs, présentant la configuration des équipes et offrant diverses statistiques et caractéristiques du jeu.

Systèmes de Notation Mettre en œuvre des algorithmes appropriés pour tenir compte du modèle et des paramètres modifiés dans le but d'obtenir de meilleures performances que les systèmes de notation actuels. Choisir une référence appropriée et représentative pour comparaison.

Améliorations et Évaluation	Suggestion et discussion des critères d'évaluation et des métriques pour analyser les gains de performance éventuels. Identifier les avantages de chaque approche, ainsi que d'éventuels désavantages.

1.2.2 Outline

La structure du projet découle de la liste des objectifs présentés ci-dessus, et le mémoire est structuré en chapitres comme suit :

Chapitre 1	Un bref aperçu de l'évolution des systèmes de notation a été présenté, ainsi que des travaux connexes dans la littérature récente.
Chapitre 2	Le modèle de notation est présenté en plus des ensembles de données que nous utilisons dans ce travail.
Chapitre 3	Le système TrueSkill original est présenté et analysé avant la discussion des améliorations et des extensions.
Chapitre 4	Les critères d'évaluation sont présentés, suivis d'une discussion sur les résultats de la simulation.
Chapitre 5	La conclusion de ce mémoire, où le progrès réalisé au cours du projet est résumé et évalué.

Chapitre 2

Modèle du système et données utilisées

2.1 Modèle du système

Dans cette section, nous formulons la tâche d'évaluation et de classement dans les matchs de compétition et introduisons les notations associées.
Nous modélisons la série de matchs comme suit : Nous considérons une variable aléatoire commune (S, Y) prenant des valeurs en $\mathcal{S} \times \mathcal{Y}$, où $\mathcal{S}$ représente l'ensemble des informations disponibles (y compris la liste des joueurs impliqués, leurs affectations d'équipe et leurs compétences supposées ou antérieures), tandis que $\mathcal{Y}$ est l'ensemble des observations ou des variables de résultats. Dans notre contexte, nous allons considérer le cas $Y \in \{\text{perte,gain}\} := \{0, 1\}$ où l'observation de Y est appelé le résultat ou l'issue du match, ainsi que le cas $Y \in \mathbb{N}^2$, auquel cas nous observons le score final de chaque adversaire, ou enfin le cas des scores de plusieurs caractéristiques disponibles où $Y \in \mathbb{N}^m$, m étant le nombre de caractéristiques.

Un match i pourrait être simplement vu comme une confrontation par paire caractérisée par deux adversaires et un résultat observé. L'intervention du système de notation se résume principalement dans ce qui suit: En se basant sur ses connaissances antérieures relatives aux compétences des équipes 1 et 2, $S_i(1)$ et $S_i(2)$, il prédit le résultat du match en termes de probabilités de gain. Ensuite, il observe le résultat Y_i et met à jour en conséquence sa croyance relatives aux compétences .

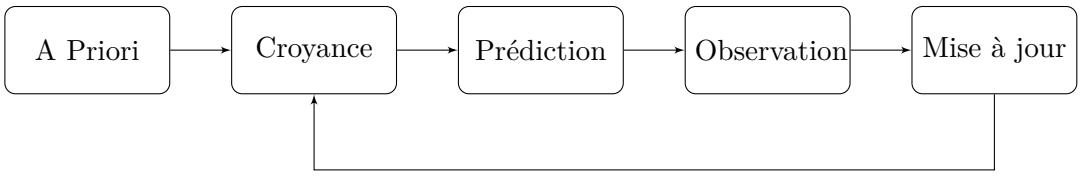


Figure 2.1 – Tâche de notation

Soit Y_{ij} la variable aléatoire binaire qui indique si le match s'est terminé par une victoire ($Y_{ij} = 1$) ou une perte ($Y_{ij} = 0$) pour l'équipe i contre l'équipe j, avec $i, j = 1 \dots n$, $i \neq j$. Les modèles de comparaison de paires décrivent la probabilité de résultat comme suit : $\mathbb{P}\mathrm{r}(Y_{ij} = 1) = \mathbb{F}(a_i - a_j)$; où $\mathbb{F}$ est une fonction de répartition et a_i est un paramètre mesurant la compétence de l'équipe i. Les modèles sont communément classés comme modèles Bradley-Terry (Bradley et Terry, 1952) ou Thurstone-Mosteller (Thurstone, 1927 ; Mosteller, 1951) selon que $\mathbb{F}$ est la fonction de répartition d'une variable logistique ou d'une variable aléatoire normale centrée réduite, respectivement. Une approche fondée sur un modèle pour résoudre ce problème est la suivante : Soit $\beta_i \in \mathcal{R}$ la "force" de l'équipe i, et que le résultat d'un match entre équipes (i, j) soit déterminé par $\beta_i - \beta_j$. Le modèle Bradley-Terry traite ce résultat comme une variable aléatoire Bernoulli indépendante avec distribution Bernoulli $\mathcal{B}(p_{ij})$, où la log-cote correspondant à la probabilité p_{ij} que l'équipe i batte l'équipe j, est modélisée par $\log \frac{p_{ij}}{1-p_{ij}} = \beta_i - \beta_j$. D'une façon équivalente, résoudre pour p_{ij} fournit le suivant:

$$p_{ij}^{\mathrm{BT}} = \frac{e^{\beta_i - \beta_j}}{1 + e^{\beta_i - \beta_j}} = \frac{e^{\beta_i}}{e^{\beta_j} + e^{\beta_j}} = \frac{1}{1 + e^{-(\beta_i - \beta_j)}} = \mathcal{S}(g(\beta_i, \beta_j)), \tag{2.1}$$

où $\mathcal{S}$ est la fonction sigmoïde ou fonction logistique et $g(\beta_i, \beta_j) = \beta_i - \beta_j$
En remplaçant $\mathcal{S}$ par Φ, la fonction de répartition d'une variable aléatoire normale centrée réduite, nous obtenons l'expression pour les modèles Thurstone-Mosteller :

$$p_{ij} = \Phi(g(\beta_i, \beta_j)). \tag{2.2}$$

TrueSkill emploie ce dernier modèle. Puisque nous basons ce travail sur le cadre de TrueSkill, analysant et discutant d'éventuelles améliorations, nous partons d'hypothèses de modélisation similaires. La compétence d'un joueur est modélisée comme un scalaire, mais la connaissance du système par rapport à la compétence tient compte de l'incertitude. Ainsi, la connaissance antérieure est modélisée comme une distribution normale $\mathcal{N}(\mu, \sigma)$, μ étant la "vraie compétence" et σ^2 représentant l'incertitude de l'algorithme quant à son estimation. Une autre incertitude est prise en considération:

la performance d'un joueur dans un match peut être influencée par plusieurs facteurs et, par conséquent, fluctuer autour de la véritable compétence scalaire. Ce phénomène est modélisé, de la même manière, comme une distribution normale avec une autre variance β^2. La performance de l'équipe est simplement la somme des performances individuelles des joueurs qui la composent. Dans la section suivante, nous allons plus en détail sur les différents aspects de modélisation de TrueSkill.

2.2 Données

2.2.1 Données synthétiques

Pour vérifier les constats et démontrer l'évolution des performances, on opte pour la création d'un cadre supervisé avec des paramètres connus à l'avance afin de mener des expériences synthétiques en guise de première étape. Chaque fois qu'une expérience synthétique est réalisée tout au long de ce travail, les données créées sont décrites sur place et les détails nécessaires sont fournis. Mais ce qui est commun, c'est que les systèmes de notation n'ont aucune connaissance préalable des paramètres de génération et n'observent que les résultats des matchs binaires ou les scores des équipes. Leur objectif est d'estimer les compétences des joueurs et de se rapprocher le plus possible des rangs réels, assurant ainsi une grande précision de prédiction. Un système serait considéré comme meilleur s'il est plus performant selon les critères d'évaluation qu'on présentera plus tard dans les sections suivantes.

2.2.2 Données réelles du monde eSports

Au cours de la dernière décennie, un type unique de sport, à savoir le sport électronique, alias eSports, est apparu comme un genre populaire de jeux vidéo, dans lequel la compétition se fait en ligne dans des environnements simulés régis par des règles et des règlements similaires à ceux que l'on trouve dans les formes traditionnelles de sport. eSports est un marché des jeux vidéo en pleine croissance qui attire un nombre considérable de joueurs professionnels, de développeurs, de fanatiques et d'organisateurs de tournois. Un rapport récent (février 2018) publié par Newzoo [18] a montré que l'économie mondiale de l'eSport atteindra 905.6 millions de dollars cette année et passer à 1,4 milliard de dollars d'ici 2020. Avec des millions qui jouent simultanément[19], une telle

popularité devient une base pour de nombreuses sociétés de paris orientées sur les événements eSport [20]. On a choisi d'acquérir nos données de jeux à partir de ce domaine croissant, car il correspond à notre configuration multijoueur et convient à nos besoins expérimentaux. Nous avons recueilli des données de matchs compétitifs pour 2 des 3 jeux multijoueurs en ligne les plus populaires : League of Legends (LoL), le jeu MOBA numéro un et Counter Strike Global Offense (CS:GO), le premier jeu de tir multijoueur en terme de popularité. Pour ce dernier, l'ensemble de données comprend 19415 matchs de compétition de diverses ligues professionnelles et championnats qui ont eu lieu entre 2012 et 2017. Les données ont été recueillies à l'aide d'un «scrapper code» que nous avons construit pour recueillir des informations sur les matchs de HLTV.com, un site Web populaire couvrant les événements CS:GO. D'autre part, le dataset de LoL a été publié par l'utilisateur Kaggle *Chuck Ephron* et il comprend 17620 matchs professionnels compétitifs entre 2015 et début 2018, y compris les ligues NALCS, EULCS, LCK, LMS et CBLoL, ainsi que les championnats du monde et les tournois sur invitation de mi-saison. Les deux ensembles de données fournissent pour chaque match, la composition des équipes en compétition, les résultats des matchs ainsi que les statistiques des équipes et les statistiques individuelles. Cela inclut, à titre d'exemples, le nombre de tuerie, d'aides et de morts, les dégâts moyens reçus, l'or collecté, les tours, les dragons,... et bien d'autres caractéristiques et statistiques du déroulement des jeux.

Chapitre 3

TrueSkill, présentation et analyse

Le système TrueSkill a été conçu par des chercheurs affiliés à Microsoft pour être utilisé dans le classement et le jumelage dans leurs Jeux en ligne sur Xbox Live. Il a été développé dans un cadre probabiliste bayésien pour étendre les principes d'Elo et de Glicko aux compétitions multijoueurs. Il a montré de meilleures capacités de prédiction que ses prédécesseurs en plus de sa capacité à couvrir différentes configurations de matchs allant de la confrontation individuelle à la mise en place de plusieurs joueurs et de plusieurs équipes. Dans ce chapitre, on commence par détailler le fonctionnement d'un tel système avant d'étudier et analyser son comportement dans des situations particulières dans le cadre des simulations synthétiques.

3.1 Le modèle de notation TrueSkill™

TrueSkill modélise le problème de notation comme étant le calcul d'estimations de compétences (en tant que postérieures) par inférence bayésienne. Formellement, il définit une distribution conjointe sur les compétences des joueurs et les résultats des matchs, sous réserve de quelques variables latentes définies par le modèle. Ensuite, selon la règle de Bayes, il déduit que l'estimation des compétences est postérieure après chaque match. Commençons par définir les différents paramètres du système. Pour la compétence initiale d'un joueur i, TrueSkill suppose qu'elle est tirée d'une distribution à priori normale :

$$s_i \sim \mathcal{N}(\mu_0, \sigma_0^2), \tag{3.1}$$

μ_0 et σ_0^2 étant des paramètres système ajustables. Ensuite, après chaque correspondance, le système met à jour ses connaissances relatives à ces deux paramètres et la distribution serait mise à jour à $\mathcal{N}(\mu_i, \sigma_i^2)$ où μ_i peut être vu comme la «vraie compétence» et σ_i^2 comme la variance qui caractérise la façon dont l'algorithme est certain quant à l'exactitude de son estimation. L'algorithme de propagation des messages garantit qu'après chaque correspondance, la variance est censée diminuer (si le modèle ne l'augmente pas artificiellement, ce qui sera discuté dans la section suivante). Après chaque match, le système suppose que les compétences des joueurs évoluent au fil du temps en fonction d'une marche aléatoire, ce qui signifie qu'une augmentation ou une diminution des compétences est tout aussi probable. Ceci est modélisé par l'addition d'une variable Gaussienne supplémentaire de moyenne nulle et de petite variance τ^2, ajoutée entre les matchs. TrueSkill suppose une autre source de variabilité des compétences : Chaque joueur a une performance en valeur réelle, p_i, dans chaque match, distribuée selon :

$$f(p_i) = \mathcal{N}(p_i; s_i, \beta^2). \tag{3.2}$$

La variance β^2 est, par hypothèse, un paramètre global (pour tous les joueurs) ajustable qui reflète la quantité de hasard dans le jeu et ses effets sur les compétences des joueurs. La performance p_i pourrait être exprimée en termes de paramètres initiaux, μ_i et σ_i^2, en intégrant simplement sur toutes les valeurs possibles de s_i comme suit :

$$f(p_i|\mu_i, \sigma_i^2) = \int_{-\infty}^{\infty} \mathcal{N}(p_i; s_i, \beta^2)\ \mathcal{N}(s_i; \mu_i, \sigma_i^2)\ ds_i. \tag{3.3}$$

Pour l'étape suivante, les performances des joueurs sont combinées en performances d'équipe. TrueSkill suppose que la performance de l'équipe est simplement la somme des performances de ses joueurs :

$$t_{équipe_i} = \sum_{j \in \text{équipe}_i} p_j \sim \mathcal{N}(t_i; \sum_{j \in \text{équipe}_i} s_i, \beta^2). \tag{3.4}$$

Les performances de l'équipe sont ensuite comparées, $d = t_{équipei} - t_{équipej}$, et le gagnant anticipé par TrueSkill est l'équipe avec la plus grande performance. Si les matchs nuls sont possibles, le système TrueSkill couvre ce cas en introduisant le concept de «marge de match nul» par le biais d'un nouveau paramètre global, $\epsilon > 0$ et déclare les tirages lorsque la différence de performances de l'équipe est inférieure à ϵ, $|t_1 - t_2| \leq \epsilon$. Cependant, on ne se concentre pas sur ce point car les jeux qu'on étudie ne sont pas susceptibles d'aboutir à des matchs nuls. Les résultats des correspondances

sont sous forme binaire discrète, $y \in \{0, 1\}$, qui doivent être «ajustés» pour s'adapter à l'algorithme gaussien de transmission des messages. L'objectif serait de calculer la distribution postérieure sur les compétences données par rapport au résultat, $f(\mathbf{s}|y)$. Pour ce faire, la règle de Bayes est utilisée et la probabilité jointe de ce modèle peut être écrite comme suit :

$$f(y, \mathbf{d}, \mathbf{t}, \mathbf{p}, \mathbf{s}) = f(y|\mathbf{d})\ f(\mathbf{d}|\mathbf{t})\ f(\mathbf{t}|\mathbf{p}) f(\mathbf{p}|\mathbf{s})\ f(\mathbf{s}), \tag{3.5}$$

où la notation en gras dénote les variables vectorielles qui englobent celles de tous les joueurs. Par exemple, le vecteur performance $\mathbf{p} = [p_1, p_2, \ldots, p_l]$, l étant les nombre de joueurs dans le match. La vraisemblance peut être exprimée comme suit:

$$f(y|\mathbf{s}) = \iiint\ f(y, \mathbf{d}, \mathbf{t}, \mathbf{p}, \mathbf{s})\ d\mathbf{d}\ d\mathbf{t}\ d\mathbf{p}, \tag{3.6}$$

pour avoir finalement, par Bayes:

$$f(\mathbf{s}|y) = \frac{f(y|\mathbf{s})\ f(\mathbf{s})}{\int f(y|\mathbf{s})\ f(\mathbf{s})\ d\mathbf{s}}. \tag{3.7}$$

Par conséquent, le problème se réduit à un problème de marginalisation, qui est l'un des sujets de base de la théorie de l'analyse bayésienne [2, 13]. Ce problème est résolu par l'algorithme de marginalisation bien connu de propagation des messages (un exemple de graphe de facteur peut être trouvé dans la Figure 3.1) avec une astuce supplémentaire qui est la propagation des messages «approximative» dans la partie inférieure du graphique. Presque toutes les distributions de variables, facteurs et messages sont gaussiennes, conduisant à des calculs faciles où la propagation des messages est effectuée de haut en bas et inversement. Cependant, le noeud du bas (i.e, le noeuf $\mathbb{1}(d_1 > \epsilon)$ sur la Figure 3.1), qui est une fonction échelon discrète, introduit une complication et ne se propage pas facilement. Pour résoudre ce problème et s'adapter au cadre gaussien de propagation des messages, cette distribution est approchée avec une gaussienne par correspondance (*matching*) des moments (c'est-à-dire par calcul et correspondance avec les deux premiers moments), et l'algorithme de propagation des messages se poursuit le long de la partie inférieure du graphique jusqu'à la convergence. La méthode qui spécifie cette approximation et les critères de convergence est appelée propagation de l'espérance, ou Expectation Propagation [14]. De cette façon, l'information acquise à partir de la comparaison et du résultat du match est propagée vers le haut pour mettre à jour

les compétences des joueurs après chaque match. Ces compétences postérieures sont alors utilisées comme à priori pour le prochain jeu [15].

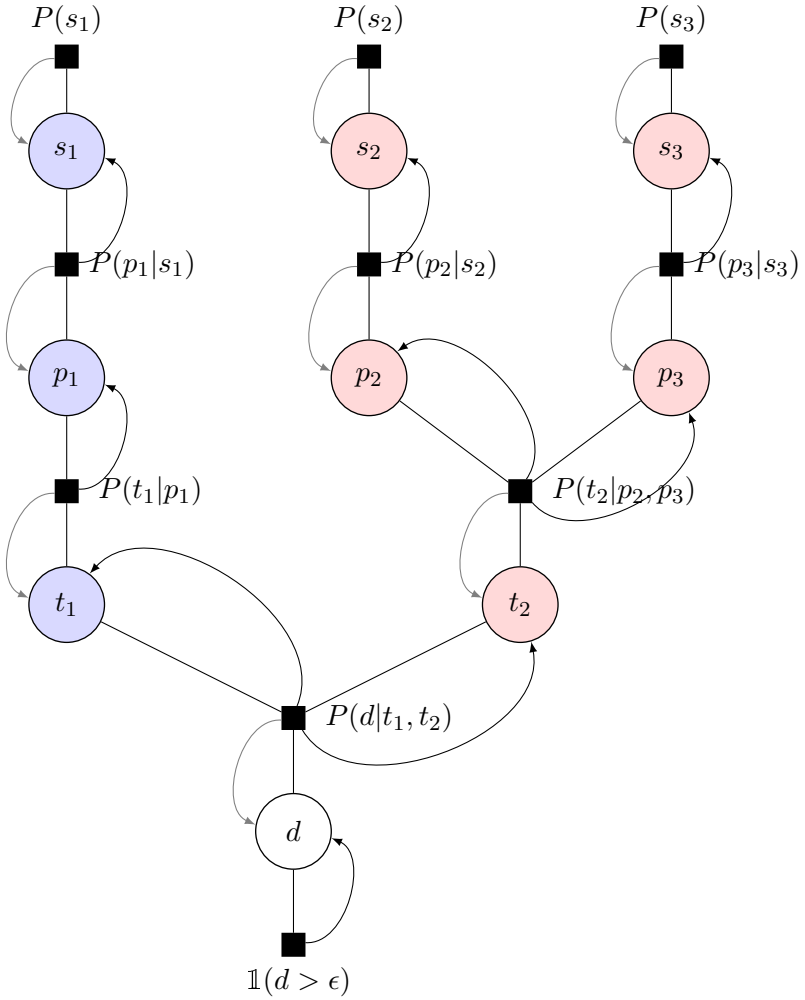


Figure 3.1 – Exemple de graphe de facteur de TrueSkill pour deux équipes. Un joueur forme l'équipe bleue et l'équipe rouge comporte deux joueurs.

Le passage des messages pour les variables continues est caractérisé par les équations suivantes :

$$p(v_k) = \prod_{f \in F_{v_k}} m_{f \to v_k}(v_k), \tag{3.8}$$

$$m_{f \to v_k}(v_k) = \int \dots \int f(\mathbf{v}) \prod_{i \neq j} m_{v_i \to f}(v_i) d\mathbf{v}_{\setminus j}, \tag{3.9}$$

$$m_{v_k \to f}(v_k) = \prod_{\tilde{f} \in F_{v_k} \setminus \{f\}} m_{\tilde{f} \to v_k}(v_k), \tag{3.10}$$

où F_{v_k} dénote l'ensemble des facteurs liés à la variable v_k et $\mathbf{v}_{\setminus j}$ correspond aux composantes du vecteur $\mathbf{v}$ à l'exception de sa composante j^{th}.

3.2 Étude de cas synthétique

TrueSkill est un algorithme de mise à jour en ligne en temps réel, qui base son estimation sur sa précédente à travers le temps. Cet aspect en ligne est utile car il s'agit d'une approximation en temps réel de la distribution postérieure sur les compétences. Cependant, cela peut induire une erreur. Le traitement de toutes les correspondances dans ce que on appelle le calcul batch (par lots) est l'évaluation des compétences la plus précise. Pour voir un exemple de situation où l'aspect séquentiel en ligne mène à un mauvais classement, considérons un tournoi de 7 matchs impliquant 8 joueurs, $p_i, i \in \{1, 2, 3, 3, 4, 5, 6, 7, 8\}$, comme suit :

Match	**Joueur 1**	**Joueur 2**	**Gagnant**
1	p_1	p_2	p_1
2	p_3	p_4	p_3
3	p_5	p_6	p_5
4	p_7	p_8	p_7
5	p_3	p_2	p_2
6	p_5	p_4	p_4
7	p_7	p_6	p_6

Tableau 3.1 – Liste des matches dans l'ordre chronologique

où les matchs se déroulent chronologiquement dans cet ordre spécifique tout en assumant que les joueurs ont initialement la même distribution à priori. Considérons les mises à jour faites par TrueSkill. Les 4 premiers résultats impliquent que les joueurs p_1, p_3, p_5 et p_7, qui ont gagné contre des adversaires avec les mêmes compétences, ont les mêmes compétences après mise à jour. Il en va de même pour p_2, p_4, p_6 et p_8 après avoir perdu contre des adversaires équivalents. Dans les 3 matchs suivants, les joueurs gagnants, p_2, p_4 et p_6, ont commencé avec les mêmes compétences et se retrouvent donc avec des compétences similaires puisqu'ils ont joué contre des adversaires équivalents. Encore une fois, il en va de même du côté des perdants. Les mises à jour en ligne ont alors pour résultat d'avoir $p_2 = p_4 = p_6$ et $p_3 = p_5 = p_7$. Par exemple, si l'on considère un traitement batch, c'est-à-dire tous les matchs se déroulant en même temps, le classement réel serait $p_1 > p_2 > p_3 > p_4 > p_5 > p_5 > p_6 > p_7 > p_8$.

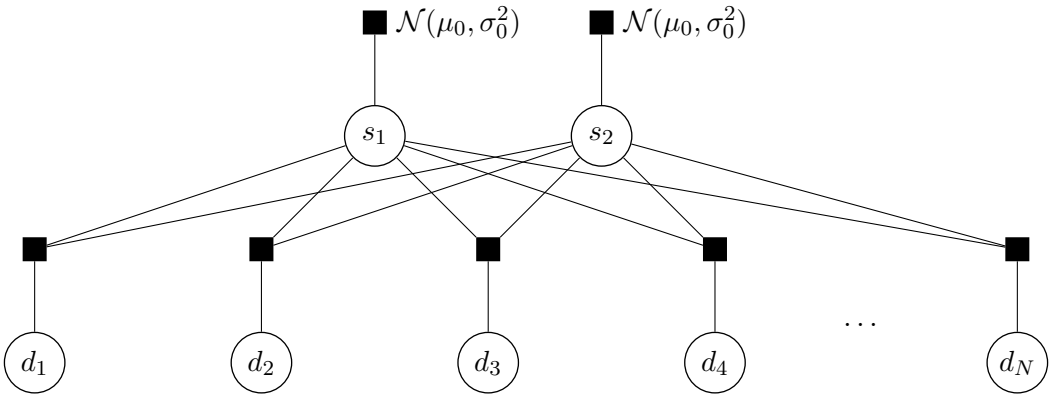


Figure 3.3 – Le graphe de facteur pour la configuration batch.

On peut voir cela en utilisant le raisonnement suivant : p_8 est le seul joueur qui n'a gagné aucun match, donc, il est le plus faible et il réside au bas du tableau. Tous les autres ont gagné exactement un match. p_7 est le seul qui a réalisé sa victoire contre le plus faible p_8, et donc il est le deuxième plus faible. p_6 est le seul à avoir remporté sa victoire contre le deuxième plus faible p_7, ce qui fait de lui le troisième dans le classement des plus faibles. Et ainsi de suite en utilisant le même raisonnement, on aboutit au classement batch que l'on a indiqué auparavant.

Une autre observation relative au comportement de l'estimation des compétences en ligne par le biais de la propagation d'espérance (*Expectation Propagation* [14]) sur les graphes, concerne la création éventuelle d'une situation du genre «bouclé» au niveau du graphe et ce qui en découle concernant la sur-confiance de l'estimation de la variance. Pour voir cet effet, considérons deux joueurs caractérisés par des compétences s_1 et s_2 qui jouent l'un contre l'autre pour N matchs. La distribution à priori est $\mathcal{N}(\mu_0, \sigma_0^2)$. Supposons que $\beta = 0$ et que la variable de différence pour un match k, d_k, est observable avec une variance σ_d^2. Le graphe de facteur pour un seul match se réduit alors à 3.2. Pour les mises à jour en ligne de TrueSkill, le nouveau à priori pour chaque match est la compétence postérieure calculée lors du match précédent. Pour le graphe de traitement batch en revanche, les deux variables de compétences sont attachées en même temps à tous les facteurs les reliant aux variables de différence, $d_k, k = 1...N$, comme on peut le voir dans la Figure 3.3.

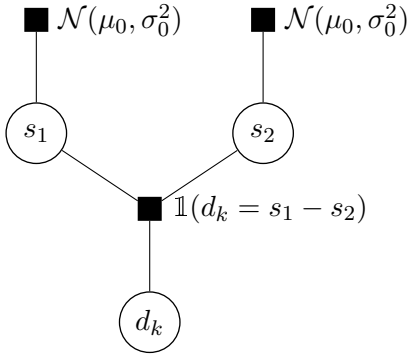


Figure 3.2 – Un exemple de graphe de facteur simplifié pour un simple match à deux.

Nous avons $d_k \propto \mathcal{N}\left(d_k; s_1 - s_2, \sigma_d^2\right)$ pour tous les matchs $k = 1, ..., N$. Par la règle de produit en probabilité (*chain rule*), on obtient la probabilité jointe de tous les nœuds dans le graphe comme étant le produit suivant :

$$P(s_1, s_2, \mathbf{d}) = \prod_{k=1}^{N} P(d_k|s_1, s_2)\ P(s_1)\ P(s_2). \tag{3.11}$$

En divisant par la distribution marginale des différences $\mathbf{d}$ on obtient la distribution jointe des compétences conditionnelle sur $\mathbf{d}$, ou la distribution posteriori. Elle est proportionnelle à :

$$\begin{aligned} P(s_1, s_2|\mathbf{d}) &\propto \prod_{k=1}^{N} P(d_k|s_1, s_2)\ P(s_1)\ P(s_2) \\ &= \prod_{k=1}^{N} \mathcal{N}\left(d_k; s_1 - s_2, \sigma_d^2\right)\ \mathcal{N}\left(s_1; \mu_0, \sigma_0^2\right)\ \mathcal{N}\left(s_2; \mu_0, \sigma_0^2\right), \end{aligned} \tag{3.12}$$

où $\bar{d} = \dfrac{1}{N} \sum_{k=1}^{N} d_k$.
Le produit des vraisemblances pourrait être calculé comme suit :

$$\begin{aligned} \prod_{k=1}^{N} \mathcal{N}\left(d_k; s_1 - s_2, \sigma_d^2\right) &= \prod_{k=1}^{N} \mathcal{N}\left(s_2; s_1 - d_k, \sigma_d^2\right) \\ &= \prod_{k=1}^{N} \tilde{\mathcal{N}}\left(s_2; \frac{s_1 - d_k}{\sigma_d^2}, \sigma_d^{-2}\right) \\ &\propto \tilde{\mathcal{N}}\left(s_2; \sum_{k=1}^{N} \frac{s_1 - d_k}{\sigma_d^2}, N\sigma_d^{-2}\right) \\ &= \mathcal{N}\left(s_2; s_1 - \sum_{k=1}^{N} \frac{d_k}{N}, \frac{\sigma_d^2}{N}\right) \\ &= \mathcal{N}\left(s_2; s_1 - \bar{d}, \frac{\sigma_d^2}{N}\right). \end{aligned} \tag{3.13}$$

De retour à la posteriori, on marginalise par rapport à s_2 pour obtient:

$$\begin{aligned}
P(s_1|\mathbf{d}) &= \int P(s_1, s_2|\mathbf{d})\ ds_2 \\
&\propto \int \mathcal{N}(\left(s_2; s_1 - \bar{d}, \frac{\sigma_d^2}{N}\right)\ \mathcal{N}\left(s_1; \mu_0, \sigma_0^2\right)\ \mathcal{N}\left(s_2; \mu_0, \sigma_0^2\right)\ ds_2 \\
&= \mathcal{N}\left(s_1; \mu_0, \sigma_0^2\right) \int \mathcal{N}\left(s_2; s_1 - \bar{d}, \frac{\sigma_d^2}{N}\right)\ \mathcal{N}\left(s_1; \mu_0 + \bar{d}, \sigma_0^2 + \frac{\sigma_d^2}{N}\right)\ ds_2 \\
&= \mathcal{N}\left(s_1; \mu_0, \sigma_0^2\right)\ \mathcal{N}\left(s_1; \mu_0 + \bar{d}, \sigma_0^2 + \frac{\sigma_d^2}{N}\right) \int \mathcal{N}\left(s_2; s_1 - \bar{d}, \frac{\sigma_d^2}{N}\right)\ ds_2 \\
&= \tilde{\mathcal{N}}\left(s_1; \frac{\mu_0}{\sigma_0^2} + \frac{\mu_0 + \bar{d}}{\sigma_0^2 + \frac{\sigma_d^2}{N}}, \frac{1}{\sigma_0^2 + \frac{\sigma_d^2}{N}} + \frac{1}{\sigma_0^2}\right) \\
&= \mathcal{N}\left(s_1; \mu_0 + \frac{\bar{d}\sigma_0^2}{2\sigma_0^2 + \frac{\sigma_d^2}{N}}, \frac{\sigma_0^2(\sigma_0^2 + \frac{\sigma_d^2}{N})}{2\sigma_0^2 + \frac{\sigma_d^2}{N}}\right).
\end{aligned} \tag{3.14}$$

Nous pouvons voir que lorsque $N \longrightarrow \infty$, la distribution tend à $\mathcal{N}\left(s_1; \mu_0 + \frac{\bar{d}}{2}, \frac{\sigma_0^2}{2}\right)$
Par contre, lorsque on effectue le calcul exact à l'aide des mises à jour en ligne, on observe que l'estimation de la variance est inférieure à la valeur batch et tend vers zéro lorsque N tend vers l'infini :

$$\frac{1}{\sigma_1^2(t+1)} = \frac{1}{\sigma_1^2(t)} + \frac{1}{\sigma_2^2(t) + \sigma_d^2(t)} \Rightarrow \sigma_1^2(N) = \left(\frac{1}{\sigma_1^2(0)} + \sum_t^N \frac{1}{\sigma_2^2(t) + \sigma_d^2(t)}\right)^{-1} \xrightarrow{N\to\infty} 0. \tag{3.15}$$

Ceci est expliqué par l'aspect séquentiel en ligne de la propagation d'espérance sur ce graphe en boucle qui conduit à une estimation trop confiante de la variance, comme prouvé dans [16]. TrueSkill évite ce problème en augmentant artificiellement la variance entre les matchs en ajoutant une variance supplémentaire τ^2 mentionnée dans la section précédente.

Chapitre 4

TrueSkill 1.1, les améliorations proposées

Dans ce chapitre, on se propose de détailler les différentes modifications et extensions apportées à l'algorithme original de TrueSkill dans l'intention d'améliorer ses capacités de classement et de prédiction, compte tenu de la spécificité des jeux et des ensembles de données disponibles. Quelques idées de modifications ont été suggérées récemment, on commence par présenter brièvement deux d'entre eux, Score based TrueSkill [4] et TrueSkill Through Time [3] et on enchaine ensuite avec comment on les adapte dans notre nouvelle configuration pour estimer les compétences en utilisant les nombreuses statistiques et propriétés disponibles dans nos ensembles de données afin d'améliorer les performances en optimisant nos critères d'évaluation.

4.1 Utilisation d'informations supplémentaires relatives au déroulement du jeu

Le système TrueSkill base sa croyance en ce qui concerne les compétences des joueurs sur les résultats des matchs, ce qui a l'avantage d'être général et applicable à tous les genres de jeux dont l'espace de résultats est {victoire, perte, égalité}. Cependant, cette approche pourrait rejeter certaines informations utiles. La plupart des jeux multijoueurs en ligne de nos jours fournissent une multitudes de *features*, soit toute propriété mesurable individuelle ou caractéristique d'un

phénomène observé au cours du jeu. Cela inclut les scores et les statistiques que ce soit individuelles ou collectives collectés tout au long du déroulement du jeu. Ces derniers représentent une source potentielle importante d'informations et de renseignements sur les compétences des joueurs. Dans cette section, on vise à exploiter ces informations supplémentaires pour augmenter le système TrueSkill original afin de capturer un aspect plus large des compétences des joueurs.

4.2 Utilisation des scores - Modèle de compétences offensives et défensives

Les auteurs de [4] ont eu une idée similaire lorsqu'ils ont décidé d'utiliser les scores de jeux comme le football. A la fin de ce genre de jeux, chaque équipe se retrouve avec un score et celui qui a le score le plus élevé est le gagnant. Pour ce faire, ils ont introduit le «modèle de compétences offensives et défensives" pour exploiter les deux scores: Dans un match entre deux équipes i et j produisant des scores respectifs $\alpha_i \in \mathbb{Z}$ et $\alpha_j \in \mathbb{Z}$ pour chaque équipe, l'idée est de considérer α_i comme résultant de la compétence offensive de i $s_i^o \in \mathbb{R}$ et de la compétence défensive de j, $s_j^d \in \mathbb{R}$ et de même pour le score de j j comme résultat de l'attaque de j, $s_j^o \in \mathbb{R}$ et de la défense de i, $s_i^d \in \mathbb{R}$. L'objectif est alors d'inférer les compétences offensives et défensives compte tenu du score observé et des connaissances antérieures. En supposant que le score α_i ne dépend que de s_i^o et s_j^d et α_j seulement sur s_j^o et s_i^d, c'est à dire, $f(\alpha_i, \alpha_j | s_i^o, s_j^o, s_i^d, s_j^d) = f(\alpha_i | s_i^o, s_j^d)\ f(\alpha_j | s_j^o, s_i^d)$, et en supposant que la distribution marginale jointe sur les compétence à priori se factorise indépendamment comme $f(s_i^o, s_j^o, s_i^d, s_j^d) = f(s_i^o)\ f(s_j^o)\ f(s_i^d)\ f(s_j^d)$, la règle de Bayes donne la distribution postérieure suivante :

$$\begin{aligned} f(s_i^o, s_j^o, s_i^d, s_j^d | \alpha_i, \alpha_j) &\propto f(\alpha_i, \alpha_j | s_i^o, s_j^o, s_i^d, s_j^d)\ f(s_i^o, s_j^o, s_i^d, s_j^d) \\ &\propto \underbrace{f(\alpha_i | s_i^o, s_j^d)\ f(s_i^o)\ f(s_i^d)}_{f(s_i^o, s_j^d | \alpha_i)}\ \underbrace{f(\alpha_j | s_j^o, s_i^d)\ f(s_j^o)\ f(s_j^d)}_{f(s_j^o, s_i^d | \alpha_j)}. \end{aligned} \tag{4.1}$$

De cette façon, l'estimation de la distribution postérieure jointe se réduit à deux problèmes d'inférence indépendants, à savoir :

$$f(s_i^o, s_j^d | \alpha_i) \propto f(\alpha_i | s_i^o, s_j^d)\ f(s_i^o)\ f(s_i^d), \text{et} \tag{4.2}$$

$$f(s_j^o, s_i^d | \alpha_j) \propto f(\alpha_j | s_j^o, s_i^d)\ f(s_j^o)\ f(s_j^d). \tag{4.3}$$

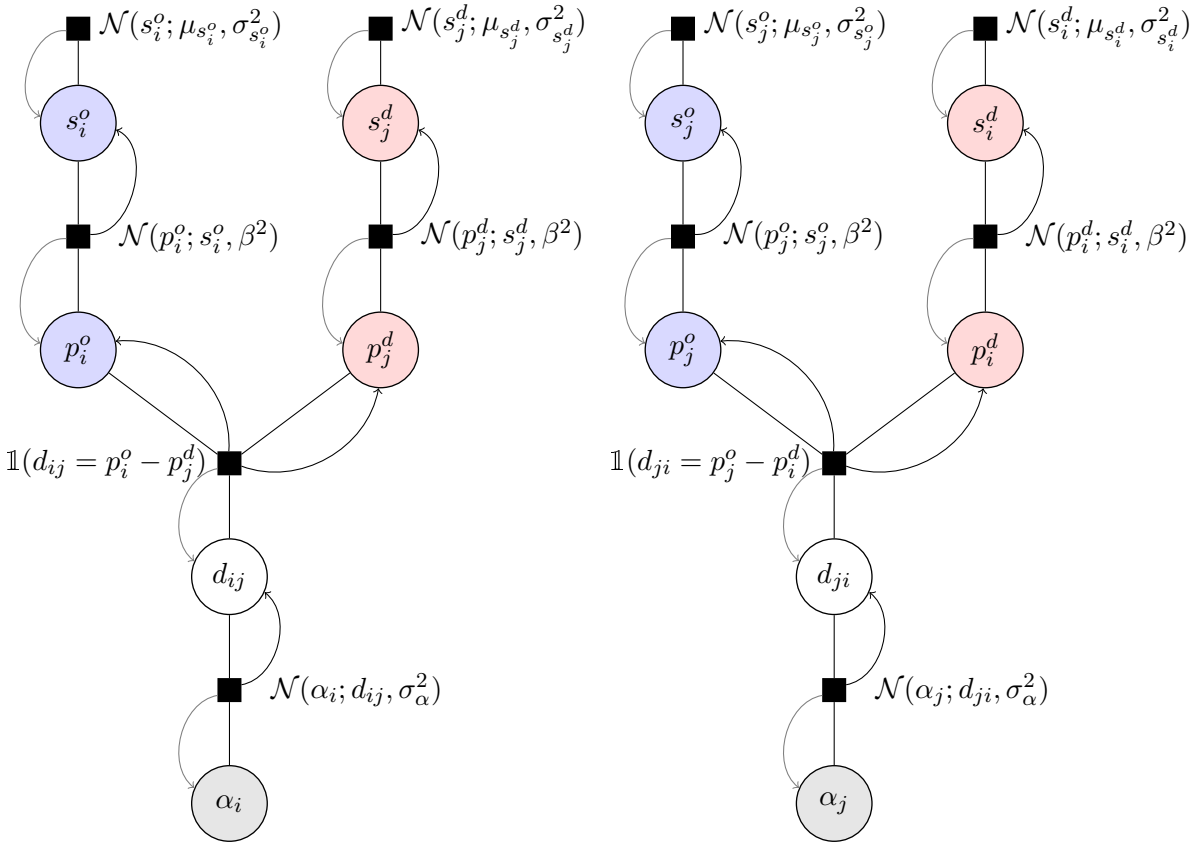


Figure 4.1 – Graphe de facteur de TrueSkill-basé-sur-les-scores.

Les scores α_i et α_j sont modélisés dans le graphe de gauche et le graphe de droite respectivement. Les variables grises sont les variables observées. Pour chaque équipe $k \in \{i, j\}$, il est caractérisé par la compétence offensive s_k^o et la compétence défensive s_k^d. Les distributions postérieures sur $(s_i^o; s_j^d)$ et $(s_j^o; s_i^d)$ sont déduites par la propagation des messages.

Tous les compétences à priori sont supposées être gaussiennes et sont sujettes à des fluctuations de performance à chaque match qui est modélisé par la variance supplémentaire β : Une équipe k manifeste la performance offensive $p_k^o \sim N(s_k^o, \beta^2)$ et la performance défensive $p_k^d \sim N(s_k^d, \beta^2)$. Les performances affectent le score comme suit: l'attaque ou le pouvoir offensif favorise le taux de score tandis que la défense l'inhibe, de telle sorte que la différence $p_k^o - p_h^d$ est le taux de score effectif de l'attaque contre la défense. Les scores sont supposés être tirés de distributions de Poisson ou de Gauss. Le graphe de facteur qui sous-tend ce dernier problème d'inférence est présenté dans la Figure.4.1. Les équations de mise à jour pour les messages sont les mêmes que dans [2] à l'exception de la distribution marginale sur la variable de différence de performance d et du message passé du facteur la reliant à la variable de score. Dans le cas du score distribué en Poisson, des mises à jour approximatives des croyances par inférence bayésienne variationnelle sont établies par les auteurs afin de l'approximer par une gaussienne et ainsi assurer la traçabilité. Lorsque les scores

sont modélisés en tant que des variables gaussiennes, cela garantit que tous les messages sont gaussiens et qu'il est donc possible de calculer les expressions exactes de mise à jour des croyances, comme suit :

$$\sigma^2_{o_i}(t+1) = \left(\frac{1}{\sigma^2_{o_i}(t)} + \frac{1}{2\beta^2 + \gamma^2 + \sigma^2_{d_j}(t)}\right)^{-1}, \tag{4.4}$$

$$\mu_{o_i}(t+1) = \sigma^2_{o_i}(t+1)\left(\frac{\mu_{o_i}(t)}{\sigma^2_{o_i}(t)} + \frac{\mu_{d_j}(t) + \alpha_i}{2\beta^2 + \gamma^2 + \sigma^2_{d_j}(t)}\right)^{-1}, \tag{4.5}$$

$$\sigma^2_{d_j}(t+1) = \left(\frac{1}{\sigma^2_{o_i}(t)} + \frac{1}{2\beta^2 + \gamma^2 + \sigma^2_{o_i}(t)}\right)^{-1}, \tag{4.6}$$

$$\mu_{d_j}(t+1) = \sigma^2_{d_j}(t+1)\left(\frac{\mu_{d_j}(t)}{\sigma^2_{d_j}(t)} + \frac{\mu_{o_i}(t) - \alpha_j}{2\beta^2 + \gamma^2 + \sigma^2_{o_i}(t)}\right)^{-1}, \tag{4.7}$$

où les valeurs des variables à t et $t+1$ sont les valeurs antérieures et postérieures, respectivement. Les équations de mise à jour pour la compétence offensive de l'équipe j, s^o_j et la compétence défensive de l'équipe i, s^d_i, sont dérivées de la même façon. Pour calculer la probabilité de victoire de l'équipe i face à l'équipe j, la propagation des messages est utilisée pour estimer les distributions normales des variables de score α_i et α_j, et alors la probabilité voulue est celle de $\alpha_i - \alpha_j > 0$, c'est-à-dire quand le score de l'équipe i est plus grand que celui de l'équipe j. Étant donné $\alpha_i \propto \mathcal{N}\left(\mu_{\alpha_i}, \sigma^2_{\alpha_i}\right)$ et $\alpha_j \propto \mathcal{N}\left(\mu_{\alpha_j}, \sigma^2_{\alpha_j}\right)$, la probabilité de victoire de l'équipe i est la suivante :

$$\Pr\{\text{équipe}_i \text{ gagne}\} = f(\alpha_i - \alpha_j > 0) = \Phi\left(\frac{\mu_{\alpha_i} - \mu_{\alpha_j}}{\sigma^2_{\alpha_i} + \sigma^2_{\alpha_j}}\right), \tag{4.8}$$

où Φ est la fonction de répartition inverse de la loi normale centrée réduite. Cette dernière équation pourrait être exprimée à l'aide des variables d'entrée, c'est-à-dire les compétences à priori en utilisant les mises à jour de propagation des messagess pour avoir cette nouvelle expression :

$$\Pr\{\text{équipe}_i \text{ gagne}\} = \Phi\left(\frac{\mu_{s^o_i} + \mu_{s^d_i} - \mu_{s^o_j} - \mu_{s^d_j}}{\sigma^2_{s^o_i} + \sigma^2_{s^d_i} + \sigma^2_{s^o_j} + \sigma^2_{s^d_j} + 4\beta^2}\right), \tag{4.9}$$

ce qui nous permet d'avoir l'interprétation suivante : la somme des compétences offensives et défensives pourrait être considérée comme la compétence générale ou globale qui nous permet de récupérer la même probabilité de victoire que dans le TrueSkill original :

$$\Pr\{\text{équipe}_i \text{ gagne}\} = \Phi\left(\frac{\mu_{s_i} - \mu_{s_j}}{\sigma^2_{s_i} + \sigma^2_{s_j} + 4\beta^2}\right), \tag{4.10}$$

où $s_i = s_i^o + s_i^d$ and $s_j = s_j^o + s_j^d$.

4.3 Features et statistiques relatives au déroulement du Jeu

Les résultats des parties ne sont pas toujours déterminés par les scores. *League of Legends*, l'un des deux jeux envisagés pour ce travail, en est un exemple. Une équipe n'est déclarée gagnante que lorsqu'elle détruit le Nexus de l'ennemi, indépendamment des scores. Le jeu lui-même ne fournit pas de score final. D'autre part, nos deux jeux, comme presque tous les récents jeux multijoueurs en ligne, donnent des statistiques de jeu et des caractéristiques variées pour les équipes et les joueurs (individuellement). L'idée est d'utiliser ces derniers, au lieu et de la même manière que les scores, pour augmenter TrueSkill original et capturer de nouveaux aspects de compétences.

4.3.1 Compétences partielles

Les résultats finaux des matchs peuvent être considérés comme des propriétés quantitatives des équipes qui caractérisent leur performance collective pendant la partie. Dans cette perspective, les statistiques et les caractéristiques des équipes assument exactement le même rôle, en fournissant des informations quantitatives sur la manière dont les équipes ont accompli leur mission dans le jeu. Pour cette raison, on traite les statistiques comme la quantité d'or collecté (*gold*), le nombre de morts (*kills*), les dommages moyens reçus (ADR) et autres, on les traite comme s'il s'agissait de scores dans le modèle présenté précédemment, mais il s'agit plutôt de «scores partiels». Ainsi, pour chaque feature, on définie une compétence partielle associée qui est mise à jour avec la caractéristique en tant que score en utilisant les mêmes règles qu'on a définies dans la sous-section précédente. La seule différence est en fait le nombre de joueurs, puisque c'est une configuration de 5 joueurs par équipe qui caractérise nos jeux. Le graphe de facteur pour une caractéristique ϕ_i d'une équipe i impliquant l'attaque de cette dernière et la défense d'une équipe j est présenté dans la Figure 4.2. Les scores des caractéristiques sont modélisés comme des variables normalement distribuées et non comme des variables aléatoires de Poisson comme suggéré dans [4]. On a écarté cette dernière car on a remarqué que la plupart de nos statistiques ne présentent pas la même moyenne et variance quand elles varient dans le temps, comme on peut le voir pour l'or (*gold*) par exemple dans la Figure 4.3, ce qui ne correspond pas à une distribution de Poisson qui se caractérise par une égalité entre sa moyenne et

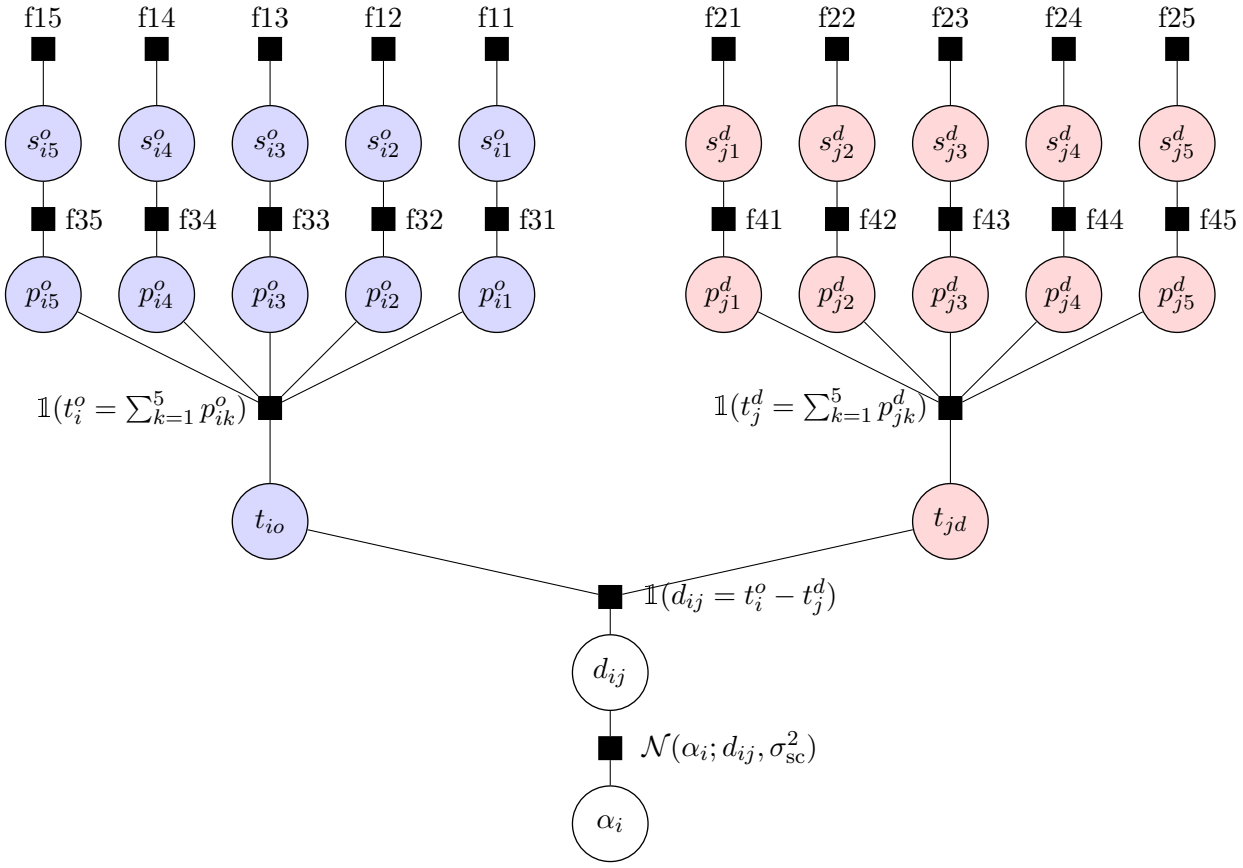


Figure 4.2 – Graphe de facteur de l'offensive de l'équipe i vs la défense de l'équipe j dans une configuration de 5 joueurs par équipe.

sa variance. Il convient également de mentionner que l'on ne se fie pas à la comparaison des scores des caractéristiques pour décider du vainqueur, mais plutôt au résultat du match : L'utilisation du premier viserait à suivre la capacité de marquer un score de cette caractéristique particulière, et non la capacité de gagner qu'on cherche à estimer. Le fait d'avoir le score le plus élevé d'une statistique particulière n'implique pas toujours le gagnant. Prenons l'exemple du nombre de *kills*, dans CS:GO il y a une forte corrélation entre les deux options, puisque le meurtre est le facteur le plus décisif du jeu. Cependant, dans LoL, la corrélation entre la victoire et le nombre de *kills* est beaucoup plus faible, vous pouvez même gagner avec un nombre de *kills* plus faible ou même pas de *kills*. C'est pourquoi on s'appuie sur le résultat réel du match plutôt que sur le signe de la différence de score afin d'avoir une compétence partielle qui caractérise la capacité de gagner corrélée par la caractéristique associée plutôt que la capacité de marquer elle-même.

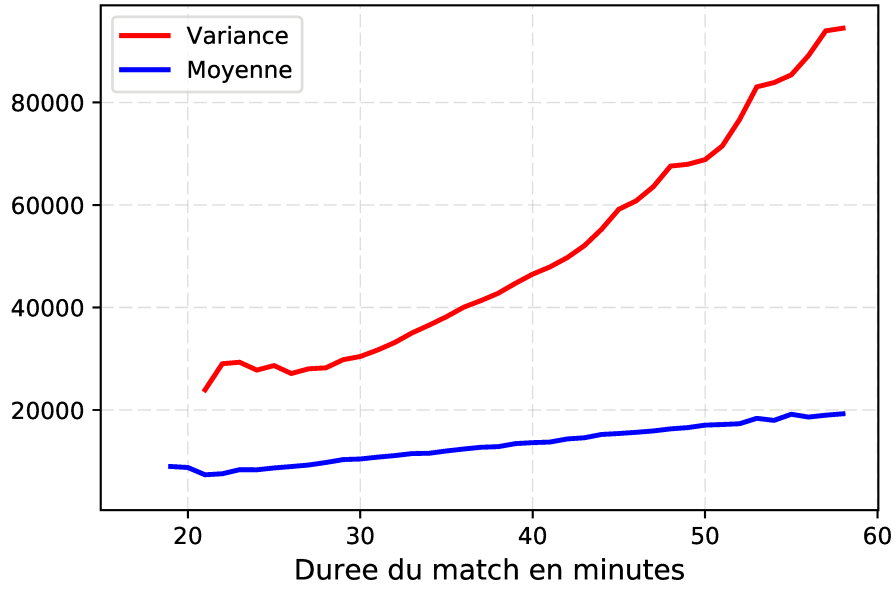
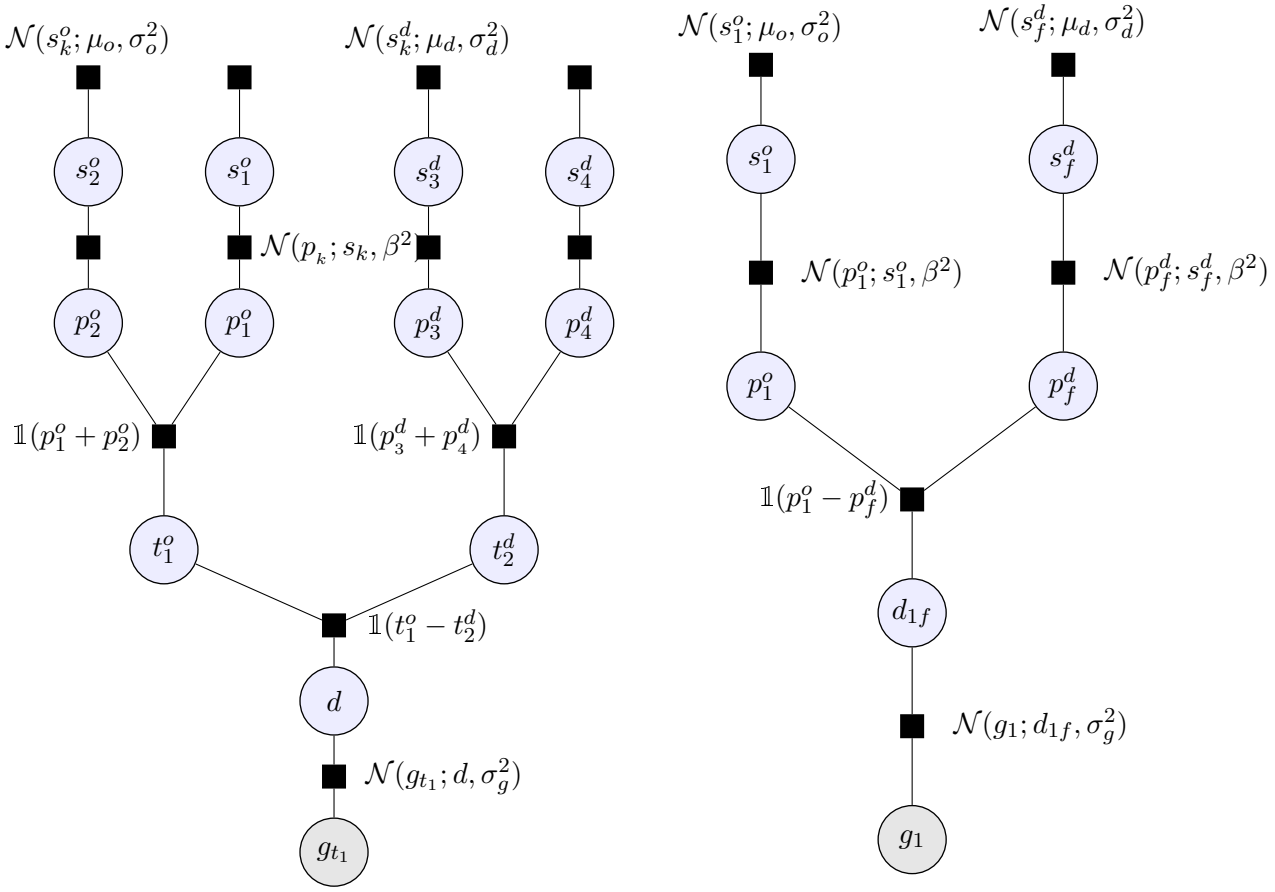


Figure 4.3 – *Gold* collecté vs durée du match pour les joueurs avec un haut niveau de compétence en LoL.

4.3.2 Modèle de statistiques individuelles

En plus des statistiques des équipes, on souhaite également utiliser les statistiques individuelles des joueurs. Dans ce cas-là, ce n'est plus une propriété d'équipe qui pourrait être qualifiée de score d'équipe. La première idée qui vient à l'esprit est de calculer la somme des statistiques de chaque joueur de l'équipe et d'en faire un nouveau score collectif pour revenir au cas présenté précédemment. Cela semble être la solution la plus naturelle, la plus facile pour résoudre le problème, mais, encore une fois, il s'agit d'éliminer des informations importantes et cela pourrait être «injuste» pour certains joueurs. Élaborons ceci: Lorsqu'un joueur CS:GO, par exemple, réalise la plupart des kills de son équipe alors que son coéquipier n'en a pas fait aucun, l'utilisation de la somme des *kills* comme score collectif donne la même mise à jour pour les deux joueurs. Cependant, si le système de classement avait été plus équitable et plus représentatif, le premier joueur aurait dû obtenir plus de crédit, proportionnellement à sa contribution à la victoire de l'équipe. Pour remédier à ce problème, on propose la modification suivante du modèle : chaque score individuel est le résultat de la confrontation entre la compétence offensive du joueur concerné et la compétence défensive d'un joueur fictif défini par la compétence moyenne de l'équipe adverse. Ceci s'applique à chaque joueur des deux équipes. De cette façon, toutes les statistiques individuelles sont utilisées séparément et la



(a) Première approche : additionner les scores individuels $g_{t1} = g_1 + g_2$

(b) Deuxième approche : chaque joueur jouant contre un joueur fictif f en tant que moyenne de l'équipe adverse.

Figure 4.4 – Graphes de facteur d'inférence à partir d'un score individuel

mise à jour de chaque joueur est pondérée par sa propre performance au lieu du score de l'équipe dans sa totalité. Pour bien assimiler et voir ceci de près, considérons une simulation synthétique où on suppose un match opposant deux équipes de deux joueurs : $t_1 = \{j_1, j_2\}$ et $t_2 = \{j_3, j_4\}$. Leurs compétences offensive et défensive sont notées, respectivement, s_k^o et s_k^d pour $k \in \{1, 2, 3, 4\}$. Nous supposons également que t_1 a gagné le jeu et que les compétences antérieures des joueurs sont, $s_k^o \sim \mathcal{N}(\mu_{ko}, \sigma_{ko}^2)$ et $s_k^d \sim \mathcal{N}(\mu_{kd}, \sigma_{kd},^2)$ pour tous les joueurs $k \in \{1, 2, 3, 4\}$. Le *gold* g est la caractéristique d'intérêt et les joueurs j_1 et j_2 ont collecté g_1 et g_2 pièces de *gold*, respectivement. Focalisons sur la compétence offensive corrélée au *gold* pour les joueurs j_1 et j_2 pour comparer les deux approches de l'utilisation des statistiques individuelles : les additionner (Approche 1) ou appliquer l'idée que l'on suggère (Approche 2). Les graphes de facteur connexes sont présentés dans la Figure 4.4. Analytiquement, en appliquant les mises à jour de propagation des messages sur ces

graphes, on a la mise à jour suivante pour les joueurs 1 et 2 selon : Approche 1 :

$$\sigma_{1o}^{2\ \text{new}} = \frac{\sigma_{1o}^2(\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2)}{\sigma_{1o}^2+\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2}, \quad \mu_{1o}^{\text{new}} = \mu_{1o} + \frac{\sigma_{1o}^2(g_{t1}+\mu_{3d}+\mu_{4d}-\mu_{2o}-\mu_{1o})}{\sigma_{1o}^2+\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2} \tag{4.11}$$

$$\sigma_{2o}^{2\ \text{new}} = \frac{\sigma_{2o}^2(\sigma_{1o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2)}{\sigma_{1o}^2+\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2}, \quad \mu_{2o}^{\text{new}} = \mu_{2o} + \frac{\sigma_{2o}^2(g_{t1}+\mu_{3d}+\mu_{4d}-\mu_{2o}-\mu_{1o})}{\sigma_{1o}^2+\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2} \tag{4.12}$$

Approche 2:

$$\sigma_{1o}^{2\ \text{new}} = \frac{\sigma_{1o}^2(\sigma_{3d}^2/2+\sigma_{4d}^2/2+2\beta^2+\sigma_g^2)}{\sigma_{1o}^2+\sigma_{3d}^2/2+\sigma_{4d}^2/2+2\beta^2+\sigma_g^2}, \quad \mu_{1o}^{\text{new}} = \mu_{1o} + \frac{\sigma_{1o}^2(g_1+\mu_{3d}/2+\mu_{4d}/2-\mu_{2o})}{\sigma_{1o}^2+\sigma_{3d}^2/2+\sigma_{4d}^2/2+2\beta^2+\sigma_g^2}. \tag{4.13}$$

$$\sigma_{2o}^{2\ \text{new}} = \frac{\sigma_{2o}^2(\sigma_{3d}^2/2+\sigma_{4d}^2/2+2\beta^2+\sigma_g^2)}{\sigma_{2o}^2+\sigma_{3d}^2/2+\sigma_{4d}^2/2+2\beta^2+\sigma_g^2}, \quad \mu_{2o}^{\text{new}} = \mu_{2o} + \frac{\sigma_{2o}^2(g_2+\mu_{3d}/2+\mu_{4d}/2-\mu_{1o})}{\sigma_{2o}^2+\sigma_{3d}^2/2+\sigma_{4d}^2/2+2\beta^2+\sigma_g^2}. \tag{4.14}$$

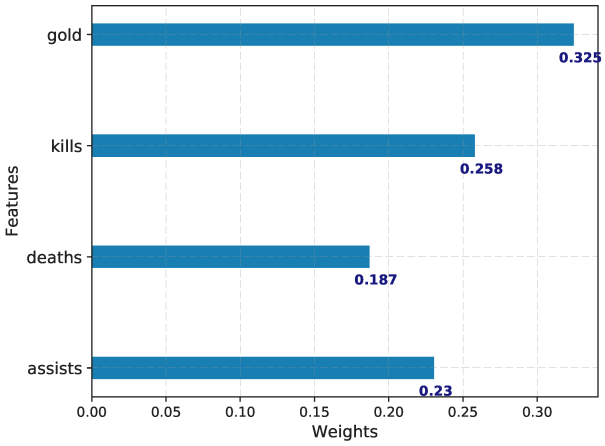
Pour pouvoir percevoir quantitativement la contribution de la deuxième approche, supposons que tous les joueurs ont les mêmes distributions à priori pour les offensives et défensives, $s_k^o \sim \mathcal{N}(25, \frac{25}{3}^2)$ et $s_k^d \sim \mathcal{N}(25, \frac{25}{3}^2)$ pour tous les joueurs $k \in \{1,2,3,4\}$. Soit $g_1 = 7$ et $g_2 = 3$ pour faire en sorte que le joueur 1 ait un impact plus grand et une contribution plus importante. Les compétences offensives mises à jour pour les deux coéquipiers selon les deux approches sont présentées dans le tableau suivant :

		Approche 1		Approche 2	
	Gold	μ	σ^2	μ	σ^2
Joueur 1	7	26.890	7.505^2	27.508	6.676^2
Joueur 2	3	26.890	7.505^2	26.075	6.676^2

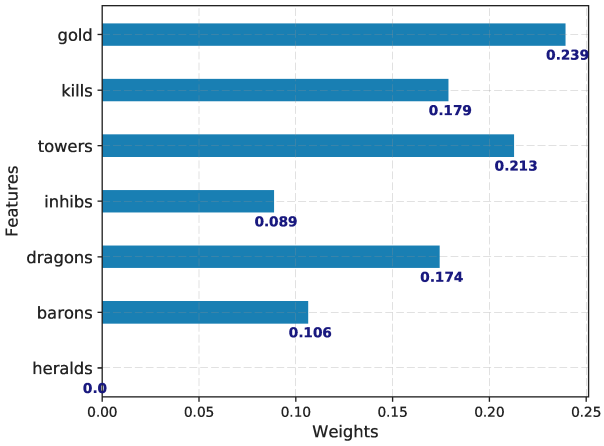
Tableau 4.1 – Comparaison des compétences offensives entre les deux approches

On peut voir que les deux coéquipiers se retrouvent avec la même mise à jour des compétences dans la première approche, la compétence du joueur 2 étant sur-estimée et celle du joueur 1 étant sous-estimée. D'autre part, la deuxième approche assure une mise à jour équitable qui tient compte de la contribution individuelle, plutôt qu'une mise à jour uniforme.

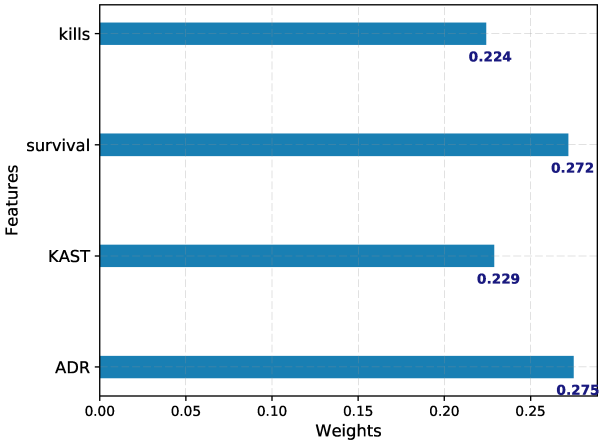
4.3.3 Aggrégation des compétences partielles



(a) LoL: Importance des features individuelles



(b) LoL: Importance des features d'équipe



(c) CS:GO: Importance des features

Figure 4.5 – Les poids traduisant l'importance de chaque feature

Pour nos deux jeux, on a de multiples caractéristiques dont on dérive de multiples compétences partielles. Puisque les caractéristiques n'ont pas le même impact sur un résultat de jeu, certaines étant plus décisives que d'autres, on quantifie l'importance de chacun comme poids visant à agréger la somme pondérée de ces compétences partielles dans une compétence globale. À cette fin, on extrait les statistiques individuelles pour chaque joueur dans chaque match comme données d'entraînement d'entrée et les résultats des matchs comme variable cible à utiliser par un algorithme de classification par arbres basé sur le *gradient boosting* pour qu'on puisse inférer les poids en fonction de l'importance des caractéristiques. L'un des avantages de l'utilisation du gradient boosting est qu'après la construction des arbres boostés, il est relativement simple de récupérer les scores d'importance pour chaque attribut. En général, l'importance fournit un score qui indique l'utilité ou la valeur de chaque

élément dans la construction des arbres de décision boosté à l'intérieur du modèle. Plus un attribut est utilisé pour prendre des décisions clés à l'aide d'arbres décisionnels, plus son importance relative est élevée. Cette importance est calculée explicitement pour chaque attribut de l'ensemble de données, ce qui permet de classer les attributs et de les comparer entre eux. L'importance est calculée pour un arbre de décision unique par le degré selon lequel chaque point de séparation des attributs améliore la mesure de performance, pondéré par le nombre d'observations dont le nœud est responsable. La mesure de performance peut être la pureté (indice de Gini) utilisée pour sélectionner les points de division ou une autre fonction d'erreur plus spécifique. La moyenne des importances des caractéristiques est ensuite calculée sur l'ensemble des arbres de décision du modèle. Dans notre cas, on a utilisé la bibliothèque Python *XGBoost*[1] pour déduire l'importance des caractéristiques et par conséquent leur poids potentiel, comme on peut le voir dans la Figure 4.5.
L'idée est donc de laisser nos compétences partielles sélectionnées, en plus de la compétence originale, être inférées indépendamment en utilisant le système TrueSkill modifié, comme si on a des couches multiples comme on peut le voir est la Figure 4.6, et avant un match, quand il s'agit de calculer les probabilités de victoire, on se fie à la compétence globale qui est la somme de ces compétences partielles pondérées par leur importance :

$$s_{\text{globale}} = \sum_{f \in \text{features}} w_f s_f \tag{4.15}$$

4.4 Récompense de la pratique

Dans le modèle original de TrueSkill, on suppose que les compétences des joueurs changent d'un match à l'autre en fonction d'une marche aléatoire, où les augmentations et les diminutions de compétences sont tout aussi probables. Ceci est modélisé en ajoutant une variance τ^2 à la compétence après chaque match. Cependant, en examinant nos données, on a fait l'observation suivante : Lorsque l'on classe les joueurs en fonction du nombre de matchs qu'ils ont déjà joués, on constate que les probabilités de gagner augmentent avec le nombre de matchs, comme le montre la Figure 4.7. Les joueurs semblent avoir une tendance à augmenter leurs compétences au fur et à mesure qu'ils pratiquent le jeu, les accroissements les plus importants ayant lieu au début. Cela pourrait être

1. https://github.com/dmlc/xgboost

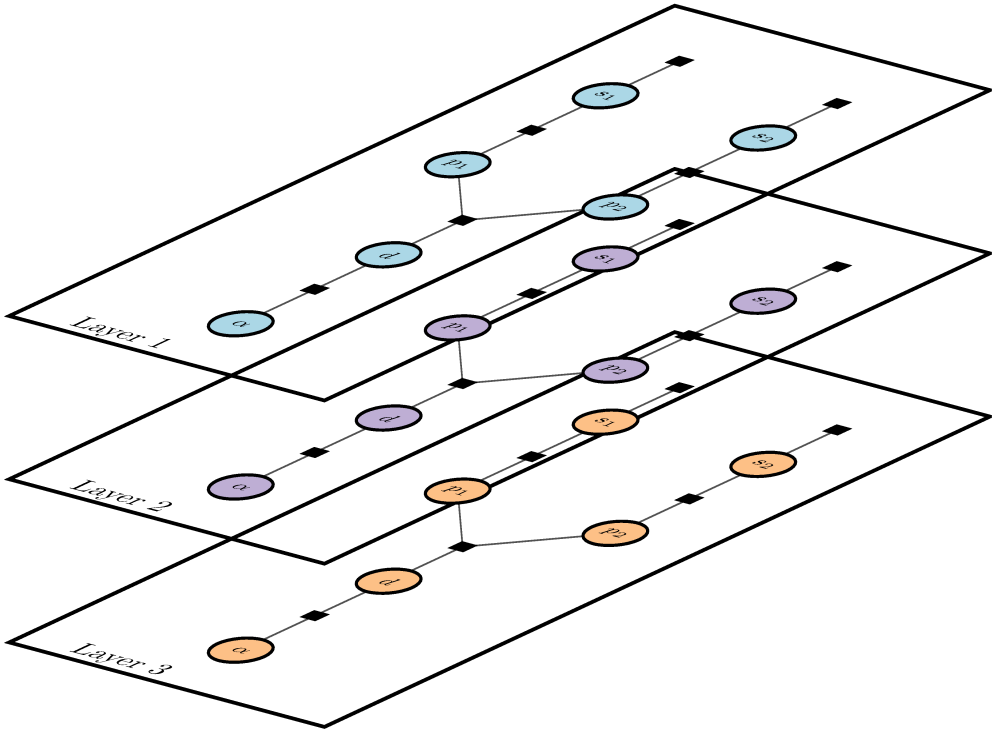
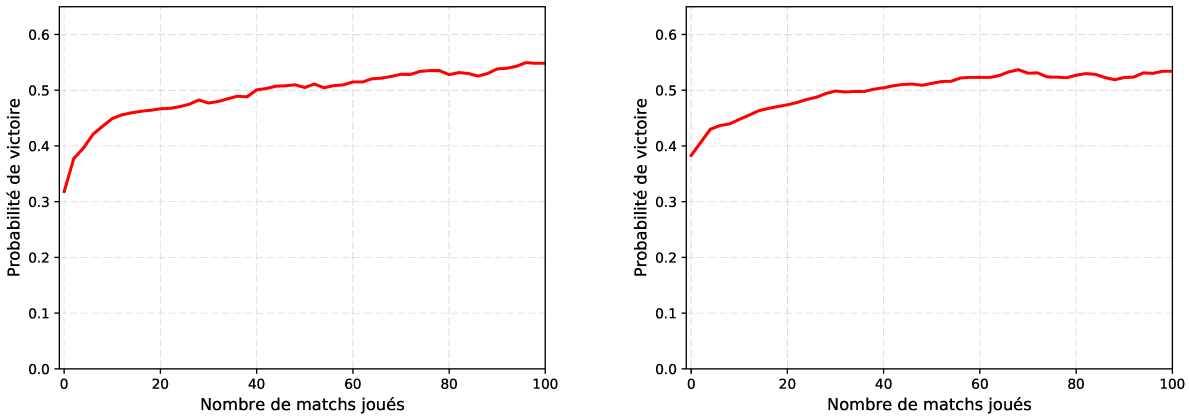


Figure 4.6 – Configuration multi-couches



(a) CS:GO (b) League of Legends

Figure 4.7 – Probabilité de victoire en fonction du nombre de matchs joués précédemment.

expliqué intuitivement si on relie l'évolution des compétences à la pratique et à l'expérience. Pour tenir compte de ce changement, on pousse la compétence plus vers l'augmentation : quel que soit le résultat du match, le joueur gagne une petite augmentation (par rapport à la mise à jour habituelle de gain/perte) dans la compétence comme récompense de pratique ou bonus d'expérience, tout en gardant la variance supplémentaire τ^2 pour tenir compte de la variabilité potentielle des deux côtés.

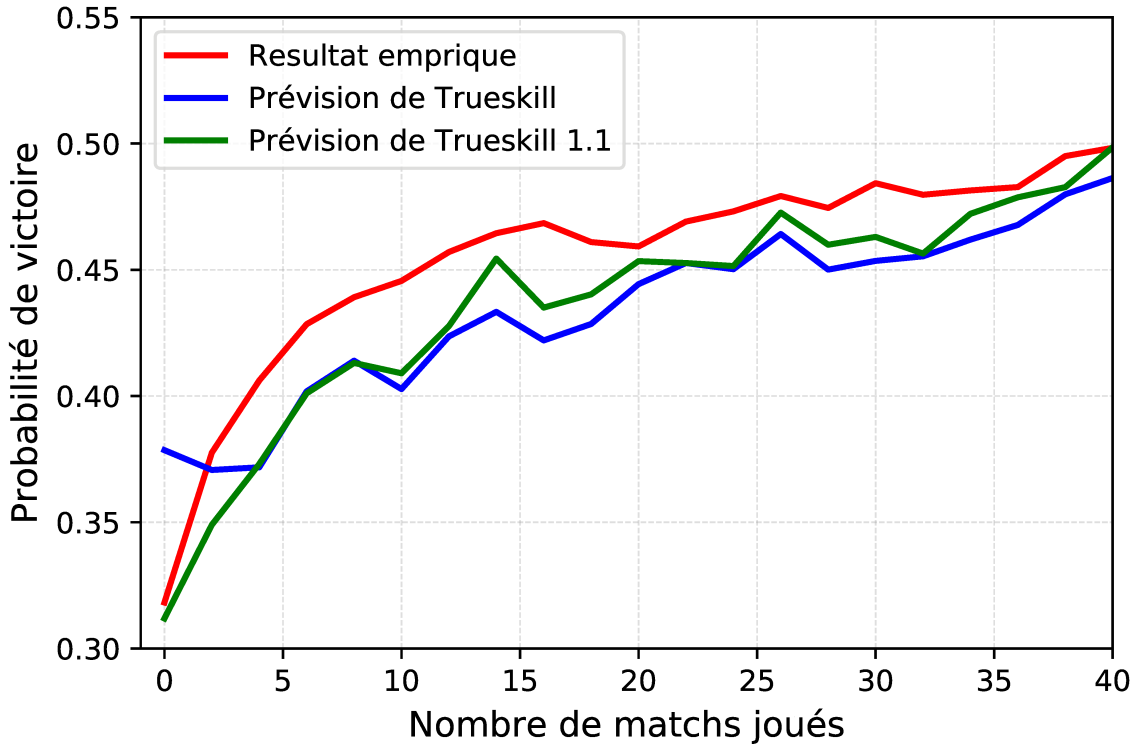


Figure 4.8 – Effet de la récompense de la pratique - CS:GO

De cette façon, la compétence serait mise à jour après chaque match comme suit :

$$s_i(n)^{\text{après}} \sim \mathcal{N}\left(s_i(n) + R_p(n), \tau\right), \tag{4.16}$$

où n est le nombre de matchs déjà joués et $R_p(.)$ est la fonction de récompense de pratique. On la choisit pour être décroissante en $\frac{1}{n}$ pour tenir compte du fait que les augmentations les plus importantes ont lieu au début. Nous laissons l'optimisation de cette fonction pour chacun de nos jeux étudiés à des travaux futurs et on montre le résultat observé dans la Figure 4.8: La récompense d'entraînement supplémentaire permet au système de faire un meilleur suivi des probabilités de gagner en termes de nombre de matchs joués précédemment (bien qu'il continue de le sous-estimer). L'avantage ici est capturé pour le cas où les matchs précédents est à zéro, .i.e., les nouveaux joueurs, pour lesquels le TrueSkill original avait tendance à surestimer les chances de victoire, alors que la modification a assuré une meilleure estimation, en moyenne, pour cette tranche de la population de joueurs, ce qui la rend surtout bénéfique lors du jumelage de nouveaux venus.

4.5 TrueSkill Through Time et traitement batch

Dans une situation où un joueur A bat le joueur B et le joueur B plus tard s'avère être très fort, comme en témoigne le fait qu'il bat à plusieurs reprises un joueur C déjà très fort, le TrueSkill original ne serait pas en mesure de propager cette information pour corriger son estimation. Pour une situation comme celle-ci et celle présentée dans la section 3.2, TrueSkill Through Time (TTT) a été suggéré dans [3]. L'idée pourrait être résumée dans ce qui suit : Les matchs sont divisés en années de classement et chaque joueur a une compétence pour chaque année. Ensuite, l'algorithme d'inférence est exécuté chronologiquement à travers les matchs de façon répétée jusqu'à la convergence. Par la suite, il est exécuté à travers les années de compétences en avant et en arrière dans le temps cette fois, jusqu'à ce qu'il converge. Lors de la propagation, chaque fois qu'un match est revisité, l'effet de visite précédente est supprimé (en divisant le message ascendant sauvegardé décrivant la performance de ce match[3]) avant d'ajouter le nouvel effet, et ce afin d'éviter un double comptage. Pour faire usage de l'idée de «TrueSkill à travers les années», on réalise un simple changement: l'astuce est de réduire la période de notation d'une année à un seul match. De cette façon, on se retrouve avec un algorithme d'inférence qui converge vers les compétences batch discutées dans la section 3.2. Nous suggérons alors ce qui suit : Le nouvelle version améliorée de TrueSkill devrait fonctionner en deux modes, un mode en ligne qui ne propage pas les évaluations de compétences en avant dans le temps, ce qui a l'avantage d'être rapide et immédiat, et un mode batch hors ligne, qui se déroule avant les tournois par exemple, qui fournit de meilleures estimations de compétences grâce à l'inférence dans le temps.

Chapitre 5

Résultats et analyse

5.1 Métriques d'évaluation

En se basant sur les compétences précédemment inférées des joueurs, les systèmes de notation sont capables de prédire les résultats en fournissant des probabilités de victoire, $\mathbf{p}$, pour chaque équipe avant le début de chaque match. L'approche directe intuitive pour évaluer la précision des systèmes est basée sur la décision binaire de victoire/échec : Le pourcentage de prédictions correctes est la précision prédictive. Une façon plus informative d'évaluer les prévisions probabilistes est la définition de la fonction de récompense ou de perte comme règle de score : Une récompense de $R(\mathbf{p}, i)$ est accordée si le $i^{\text{ème}}$ événement se produit, $i \in \{0, 1\}$. Une règle de score est *propre* lorsque la récompense moyenne la plus élevée est obtenue en rapportant la distribution de probabilité réelle. L'utilisation d'une règle de score propre encourage les joueurs à être honnête pour maximiser la récompense moyenne [17]. Une règle de score est strictement propre si elle est optimisée de façon unique par les probabilités réelles. Désignons par p la première probabilité de victoire de l'équipe (et donc $(1 - p)$ comme probabilité de victoire pour l'équipe 2).

- **Précision de prédiction**: Il est toujours intéressant de voir à quel point les modèles étaient précis pour prédire l'issue des matchs en termes de résultats binaires. Pour comparer la performance des classifications de chaque modèle, on rapporte en termes de score de précision prédictive (pourcentage de prédictions correctes) de victoire ainsi que d'aire sous la courbe (AUC). Cette dernière est la deuxième métrique la plus populaire pour la classification

binaire, après la précision. Elle se distingue par le fait qu'elle tient compte de tous les seuils possibles (par rapport au seuil de $0,5$ pour la précision), ce qui se traduit par des taux différents de vrais positifs/faux positifs. Le score AUC pour un classificateur aléatoire est de $0,5$ et serait égal à 1 pour un prévisionniste parfait. Le plus souvent, les prédicteurs donnent quelque chose entre les deux. Veuillez consulter [21] pour plus de détails concernant l'AUC par rapport à la précision.

- **Score quadratique / score de Brier**: La règle de score quadratique est une règle de score strictement propre définie comme $Q(p) = 2p - p^2 - (1-p)^2$, où p est la probabilité assignée au résultat qui s'est effectivement réalisé. Si on dénote par y_i le résultat du match i pour l'équipe 1, c'est-à-dire $y = 1$ ou $y = 0$ s'ils gagnent ou perdent, respectivement, la règle de score quadratique peut être exprimée comme suit :

$$Q(p, y_i) = y_i \ (2p - p^2 - (1-p)^2) + (1 - y_i)(2(1-p) - (1-p)^2 - p^2). \tag{5.1}$$

  Un score équivalent qui peut être obtenu par une simple transformation affine est le score de Brier défini comme $B(p, y_i) = (y_i - p)^2 + (y_i - 1 + p)^2$, avec la différence qu'un joueur devrait s'efforcer de maximiser le score quadratique mais de minimiser le score de Brier.

- **Règle de score logarithmique RSL / Gain d'information GI**: La règle de score logarithmique est une règle strictement propre qui est couramment utilisée comme critère de notation dans l'inférence bayésienne et qui repose sur la théorie de l'information. Elle peut être exprimée en logarithme comme suit : $L(\mathbf{p}) = y \ \ln(p) + (1-y) \ \ln(1-p)$ et puisque la propriété stricte est préservée par transformation linéaire, n'importe quelle base logarithmique pourrait être utilisée. Dans [22], l'auteur a proposé:

$$L(\mathbf{p}) = y \ (1 + \log_2(p)) + (1-y) \ (1 + \log_2(1-p)), \tag{5.2}$$

  qui est en effet une règle de score propre et a la propriété zéro récompense pour une probabilité de prédiction de $0,5$.

- **Métrique de la récompense de pari**: La comparaison entre les performances de deux systèmes d'évaluation pourrait être modélisée comme deux joueurs essayant d'optimiser leurs paris pour maximiser leurs gains dans une situation de «fair-play» où chaque camp commence par fixer les cotes pour l'autre qui choisit ensuite ses paris. La situation est expliquée plus en détail dans l'Annexe A où on trouve que le gain pour un camp utilisant $\tilde{p}$ contre l'autre

utilisant $\hat{p}$, alors que la distribution réelle non observée est y, pourrait être exprimée comme suit :

$$\bar{G} = \frac{y}{\hat{p}} \quad \mathbb{1}_{\{\tilde{p}>\hat{p}\}} + \frac{1-y}{1-\hat{p}} \quad \mathbb{1}_{\{\tilde{p}<\hat{p}\}}. \tag{5.3}$$

- **Métriques liées aux caractéristiques (features)** : Bien que la précision soit utile pour comparer les modèles, elle ne permet généralement pas de dire pourquoi, où et pour qui les erreurs ont été commises. Le modèle peut favoriser ou sous-estimer certains types de joueurs ayant des caractéristiques particulières. Les sous-ensembles ou sous-populations de joueurs pertinents peuvent différer selon le genre de jeu et aussi selon les intentions des gestionnaires de jeux. C'est pourquoi nous envisageons des métriques liées aux caractéristiques pour mesurer la capacité des systèmes de notation à saisir et à réduire l'écart entre le modèle et la réalité lorsqu'il s'agit de certains aspects et caractéristiques. Cela inclut les joueurs ayant un certain nombre de matchs joués précédemment, le nombre de kills par minute, les dégâts moyens reçus, etc. On peut s'intéresser aux nouveaux venus ou à ceux qui obtiennent des taux de kills élevés, qui sont sous ou sur-estimés par exemple. Cette idée pourrait être capturée en comparant le pourcentage de matchs où l'on prédisait qu'un type particulier de joueurs gagnerait, sa probabilité de victoire attendue, au pourcentage de victoire réel obtenu à partir des résultats binaire du jeu. La probabilité de victoire attendue d'une sous-population de joueurs est calculée en enregistrant la prédiction pour tous les jeux, en visitant chaque joueur dans chaque match et en vérifiant si le joueur remplit le critère en question. Si tel est le cas, la prédiction pour l'équipe de ce joueur est ajoutée au décompte et la métrique finale est une moyenne des prédictions pour tous les joueurs qui répondent au critère. Ce type de métriques a été capturé et présenté pour nos deux jeux dans la sous section qui suit dans les Tableaux 5.3 ainsi que la Figure 5.1.

5.2 Résultats de simulation et discussion

Pour les expériences de deux jeux étudiés, la moitié des données a été utilisée comme ensemble entraînement pour régler les hyperparamètres du modèle par recherche sur grille de différentes combinaisons de paramètres pour maximiser le gain d'information : $\mu_0 = 5, \sigma_0^2 = \beta^2 = 2.5^2, \tau = 10^{-3}$ et $\sigma_\alpha^2 = 1.5^2$. Ensuite, on demande au modèle de prédire les résultats dans le reste des données, l'ensemble de tests. Au fur et à mesure que chaque match est traité, on demande au modèle de prédire

l'équipe gagnante et de donner une probabilité pour cet événement. Lors de cette prédiction, seules la composition de l'équipe et les compétences antérieures sont utilisées, toutes les autres informations du jeu ne sont pas exploitées à ce stade, mais plutôt, après le match, où toutes ces informations sont utilisées pour mettre à jour les compétences à l'aide de la mise à jour en ligne. Ceci s'applique à toutes les expériences réalisées dans le cadre de ce travail. Les résultats et les effets de certaines modifications on été déjà présentés dans la section précédente, le reste des résultats est discuté dans ce qui suit. Le Tableau 5.1 dresse un comparaison entre la version originale de TrueSkill, ci-après notée TrueSkill, à notre version améliorée qu'on propose, ci-après dénommée TrueSkill 1.1, et ce en termes de quatre mesures. Pour ce qui est de scores de précision, AUC et de Brier, on constate que TrueSkill 1.1 ne réalise qu'une légère amélioration par rapport à TrueSkill, et ce pour les deux jeux. Il faut noter qu'il y a déjà un système de jumelage en usage sur le serveur de CS:GO et les matches recueilli de LoL sont des compétions professionnelles, et donc les niveaux sont assez proches et les matchs déjà équilibrés et par la suite de petites différences en terme de précision de prédiction sont donc assez significatives. Le gain est plus visible quand il s'agit du gain d'information, vu que le système compte sur de l'information additionnelle. Cela prouve que le système réussit à transmettre cette information qui n'est pas totalement perdue. Le profit apporté par les modifications se manifeste aussi à partir du Tableau 5.2. Il s'agit de la métrique de gains de pari qui oppose les deux systèmes en compétition. TrueSkill 1.1 l'emporte encore une fois dans deux jeux mais cette différence est bien plus nette quand il s'agit de LoL.

Métrique	Précision	AUC	GI	$GI_{normalisé}$	Brier
TrueSkill	0.637	0.689	7855.678	0.334	0.387
TrueSkill 1.1	0.640	0.694	8218.031	0.350	0.374

(a) CS:GO

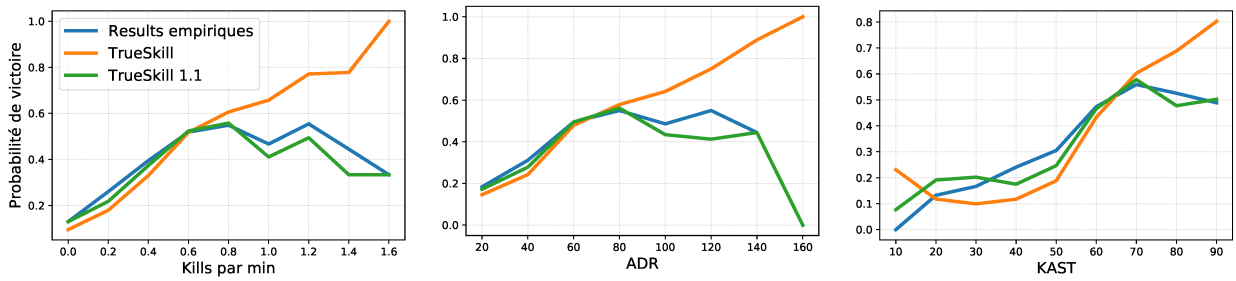
Métrique	Précision	AUC	GI	$GI_{normalisé}$	Brier
TrueSkill	0.638	0.685	2060.731	0.338	0.228
TrueSkill 1.1	0.648	0.702	2184.558	0.360	0.225

(b) LoL

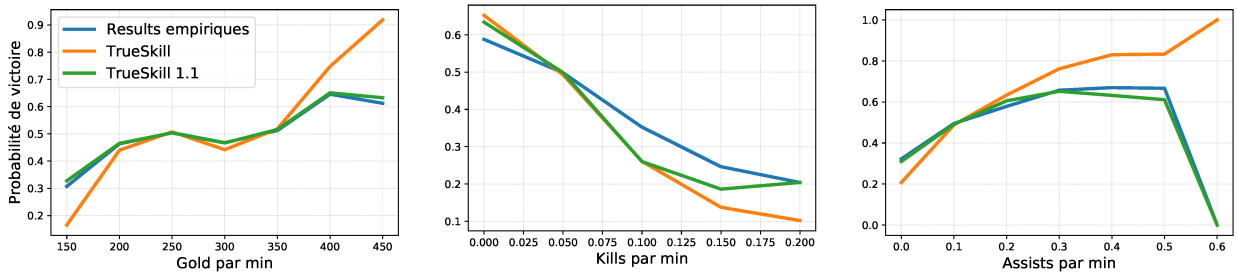
Tableau 5.1 – Comparaison des métriques des deux systèmes

	LoL	CS:GO
TrueSkill	6995.812	35362.502
TrueSkill 1.1	11377.262	36702.719

Tableau 5.2 – Comparaison des gains de paris des deux systèmes



(a) CS:GO



(b) LoL

Figure 5.1 – Probabilité de victoire vs features

On arrive enfin aux métriques qui mettent le plus en valeur les modifications apportées à TrueSkill: celles liées aux features.

La Figure 5.1 et les Tableaux 5.3 montrent l'effet de ces changements. On y répartit les joueurs de l'ensemble de test en fonction de leur taux moyen d'une feature lors de leurs matchs précédents et on évalue les taux ou les probabilités de victoire réelle ainsi que celles attendues par TrueSkill et TrueSkill 1.1 pour chaque sous ensemble des joueurs. Les caractéristiques ou features présentes sont les suivantes: Pour CS:GO, il s'agit du nombre de tués par minute, *Kills*, le dégât moyen reçu, *ADR*, et *KAST* qui modélise pourcentage de tours au cours desquels le joueur a été tué, assisté, a survécu ou a fait l'objet d'un échange. Les statistiques de LoL qu'on représente ici sont les nombres de *Kills*, *Assists* et *Gold* collectés par minute. Les Tableaux 5.3 rapportent aussi la taille de chaque sous-population ainsi que l'erreur absolue moyenne pondéré (EAMP) par les tailles des sous-populations, qui quantifie l'erreur entre les les probabilités prédites et celle réelle.

La première constatation tirée des courbes de la Figure 5.3 est que TrueSkill 1.1 fait un bien meilleur travail en terme de suivi des taux réels de victoire. D'ailleurs on voit bien que la courbe verte a tendance à bien suivre les changements que fait la courbe des résultats empiriques alors que celle orange de TrueSkill a souvent un comportement monotone qui fait qu'il y ait une différence

		Probabilité de victoire		
Kills \min	Nombre	réelle	TS	TS 1.1
0.0 - 0.3	466	0.180	0.146	0.159
0.3 - 0.6	24167	0.390	0.323	0.368
0.6 - 0.9	200608	0.521	0.523	0.527
0.9 - 1.2	2350	0.518	0.644	0.477
1.2 - 1.5	98	0.520	0.745	0.500
1.5 - 1.8	6	0.500	1.000	0.167
	EAMP		0.013	0.007

		Probabilité de victoire		
ADR	Nombre	réelle	TS	TS 1.1
20 - 40	343	0.184	0.146	0.172
40 - 60	5376	0.311	0.242	0.278
60 - 80	154755	0.495	0.479	0.495
80 - 100	66151	0.550	0.579	0.562
100 - 120	965	0.486	0.641	0.434
120 - 140	80	0.550	0.750	0.412
140 - 160	9	0.444	0.889	0.444
	EAMP		0.021	0.004

		Probabilité de victoire		
KAST	Nombre	réelle	TS	TS 1.1
30 - 40	282	0.167	0.099	0.202
40 - 50	1092	0.241	0.117	0.175
50 - 60	6229	0.305	0.188	0.247
60 - 70	115995	0.474	0.433	0.466
70 - 80	101783	0.559	0.603	0.578
80 - 90	1979	0.526	0.688	0.478
90 - 100	223	0.489	0.803	0.502
	EAMP		0.046	0.014

(a) CS:GO

		Probabilité de victoire		
Kills \min	Nombre	réelle	TS	TS 1.1
0.00 - 0.05	9943	0.588	0.652	0.632
0.05 - 0.10	45432	0.498	0.493	0.499
0.10 - 0.15	3999	0.352	0.261	0.262
0.15 - 0.20	349	0.246	0.138	0.189
0.20 - 0.25	49	0.204	0.102	0.204
0.25 - 0.30	4	0.5	0.25	0.5
	EAMP		0.021	0.015

		Probabilité de victoire		
Gold \min	Nombre	réelle	TS	TS 1.1
150 - 200	345	0.307	0.165	0.319
200 - 250	7235	0.463	0.440	0.467
250 - 300	7043	0.504	0.507	0.504
300 - 350	16796	0.468	0.442	0.467
350 - 400	23870	0.512	0.518	0.515
400 - 450	4439	0.646	0.747	0.651
450 - 500	49	0.612	0.918	0.653
	EAMP		0.028	0.016

		Probabilité de victoire		
Assists\min	Nombre	réelle	TS	TS 1.1
0.0 - 0.1	4995	0.322	0.207	0.311
0.1 - 0.2	41043	0.495	0.488	0.491
0.2 - 0.3	12349	0.579	0.633	0.606
0.3 - 0.4	1264	0.657	0.762	0.653
0.4 - 0.5	106	0.670	0.830	0.651
0.5 - 0.6	18	0.667	0.833	0.611
0.6 - 0.7	1	0.000	1.000	0.000
	EAMP		0.021	0.014

(b) LoL

Tableau 5.3 – Probabilités de victoire réelles et espérées pour les différentes sous-populations relatives à chaque feature

bien visible aux extrémums. L'observation que les probabilités fournies par TrueSkill 1.1 suivent mieux la réalité des sous population, est confirmé par l'erreur absolue moyenne pondéré (EAMP) par les tailles des sous-populations, dont les valeurs sont au moins deux fois plus petite que celle de TrueSkill. Il faut préciser aussi que les extrémums correspondent aux populations dont la taille est assez réduite comparée à celles proches du milieux où se concentrent la majorité des joueurs et où la courbe de TrueSkill suit bien celle décrivant les taux réels. En contre partie, ceci souligne une propriété qui distingue TrueSkill 1.1 à la version originale, notamment le pouvoir de bien estimer

les compétences des «minorités» comme ceux qui réalisent un grand nombre d'*assists* ou ceux qui ont tendance à collecter très peu de *gold*. En plus on a vu dans la section 4.4 que l'inclusion de la récompense de pratique permet au modèle de mieux estimer les probabilités de victoire de ceux qui jouent pour la première fois (contrairement au TS qui tendance à surestimer leurs chances). Tout cela peut s'avérer utile surtout lorsque les gestionnaires qui dirigent les jeux veulent biaiser leurs politique de jumelage (*matchmaking*) pour mettre l'accent sur une feature en particulier.

Pour observer l'apport de l'idée suggérée dans la sous section 4.5, on se propose de faire la chose suivante: On suppose que la première tranche de 80% des données constitue l'historique disponible avant le début d'un tournoi ce qui présentera l'ensemble d'entrainement sur lequel la mise à jour batch va être appliquée en mode hors ligne avant de recourir à l'inférence en ligne usuelle pour les mises à jour des matchs du tournoi, i.e, la 20% restante, ou l'ensemble de test. On suit alors l'évolution de certaines métriques après l'application du traitement batch hors ligne. Cet effet sur l'ensemble d'entrainement en soi ainsi que l'ensemble de test avec et sans la mise à jour hors ligne sur les données historiques est résumé dans les Tableaux 5.4:

Métrique	Précision	AUC	$GI_{normalisé}$	Brier
(*) Mise à jour hors ligne sur l'ensemble d'entrainement	0.737	0.807	0.491	0.401
Mise à jour en ligne sur l'ensemble de test sans (*)	0.640	0.694	0.349	0.374
Mise à jour en ligne sur l'ensemble de test avec (*)	0.651	0.707	0.358	0.382

(a) CS:GO

Métrique	Précision	AUC	$GI_{normalisé}$	Brier
(*) Mise à jour hors ligne sur l'ensemble d'entrainement	0.712	0.798	0.468	0.188
Mise à jour en ligne sur l'ensemble de test sans (*)	0.685	0.702	0.360	0.225
Mise à jour en ligne sur l'ensemble de test avec (*)	0.692	0.714	0.368	0.224

(b) LoL

Tableau 5.4 – Effet de la mise à jour batch hors ligne

On voit un gain significatif sur l'ensemble des données historiques grâce au traitement batch assuré par propagation à travers le temps. Ce gain est moins prononcé sur l'ensemble de test, et ceci peut être expliqué principalement par le fait que les matchs du tournoi impliquent de nouveaux joueurs qui faisaient pas partie de l'ensemble d'entrainement. Cependant le gain reste assez important, vu que les joueurs sont déjà jumelés comme expliqué auparavant, ce qui confirme l'intérêt de l'idée de se servir de TTT pour appliquer des mises à jours batch en mode hors ligne dans les périodes inter tournois.

Chapitre 6

Conclusion et extensions possibles

Dans ce mémoire, on s'est proposé d'analyser et d'étendre un système de notation existant, à savoir TrueSkill, pour prendre en compte et incorporer les informations supplémentaires qui sont facilement disponibles dans les jeux multijoueurs en ligne de nos jours. Ceci est disponible grâce à la multitude de statistiques fournies décrivant le déroulement de jeu telles que l'expérience en terme de nombre de matchs d'un joueur, son nombre de *kills*, le *gold* collecté, les dégâts moyens reçus et plein d'autres.

On a alors considéré quelques modifications du système pour incorporer ces features et capturer les nouveaux aspects dans le but d'optimiser les multiples métriques d'évaluation que l'on a suggéré. Tout d'abord, on a commencé par définir le système de notation et le cadre d'évaluation, y compris le modèle de système et les données recueillies. Nous avons ensuite introduit le mécanisme d'inférence bayésienne en utilisant la propagation de messages gaussiens dans TrueSkill avant d'étudier et d'analyser la convergence et l'évolution des paramètres dans le cadre d'études de cas synthétiques: On a montré que la nature en ligne des mises à jour de TrueSkill et les éventuelles boucles qui peuvent se produire dans certaines situations peuvent conduire à des classements imprécis et à une surestimation (trop sûre) de la variance par rapport au cas parfait de traitement batch.

Par la suite, on a enchainé par la discussion des améliorations et des extensions potentielles du TrueSkill original. On a commencé par présenter la façon d'incorporer les scores des features collectives dans la mise à jour des compétences avant de discuter la façon de procéder avec les statistiques individuelles. On a ensuite suggéré un changement de modèle qui permettait de tenir

compte de l'ampleur de la contribution individuelle de chaque joueur dans le match en la corrélant au score d'une feature spécifique. Cela a permis au système d'être plus équitable par rapport au TrueSkill original ou à la mise à jour basée sur le score de l'équipe. Ensuite, on a présenté les compétences partielles corrélées aux différentes caractéristiques et la façon dont on a procédé pour les agréger en une seule compétence et ce en utilisant leur somme pondérée par l'importance de chaque feature.

Enfin, on a présenté les différentes métriques sur lesquelles on s'est basé pour évaluer les gains obtenus grâce aux extensions suggérées. Cela comprenait la précision, le score AUC, le gain d'information, le score Brier, une métrique de récompense de pari ainsi que des métriques liées aux features. Étant donné que les données consistaient en duels déjà jumelés, certaines mesures n'ont montré que de légères améliorations. Le gain était plus prononcé en termes de gain d'information et de récompense de pari. En plus de cela, l'intégration de la récompense de pratique a permis de fixer la surestimation des chances de victoire des nouveaux joueurs, dont a souffert le TrueSkill original. L'utilisation de TTT pour obtenir les mises à jour batch hors ligne dans les périodes inter-tournois s'avère également capable d'améliorer les métriques d'évaluation, surtout pour les données historiques mais aussi pour les nouveaux matchs du tournoi. De plus, l'utilisation de métriques liées aux features pour suivre le comportement du système de notation auprès de certaines sous-populations spécifiques a prouvé sa capacité à mieux suivre les chances réelles de victoire associées et la capacité d'estimer correctement les compétences de certaines minorités. Tout cela pourrait être utile, en particulier lorsque les gestionnaires de jeu veulent biaiser leur politique de matchmaking pour se concentrer sur une feature particulière.

Le futur suivi de ce travail pourrait inclure entre autres ce qui suit :

- Optimiser la fonction de récompense présentée dans la section 4.4 que nous n'avons pas eu l'occasion de mener à bien.
- Étudier d'autres méthodes d'agrégation des compétences partielles, notamment tenir compte de l'inter-corrélation entre les features: Par souci de simplicité, on a supposé qu'elles sont indépendantes.
- Maintenir le suivi des corrélations entre les joueurs, ce qui a déjà été prouvé capable d'augmenter la précision [23], et l'utiliser pour plus étendre le système.
- Pour les jeux MOBA tels que LoL, explorer la modélisation des performances de l'équipe sous forme d'une somme pondérée en fonction du rôle dans l'équipe (i.e., top, jungle, MID, ADC

ou support). De plus, puisque le joueur peut choisir des champions différents, on pourrait penser à des compétences multiples et corrélées au choix du champion et éventuellement comment les agréger.

Il convient de noter qu'au moment de la rédaction de ce rapport, Microsoft a publié sa nouvelle version de son système de notation l'intitulant TrueSkill2 [24]. Les auteurs, eux aussi, ont eu l'idée d'augmenter le système original par des informations recueillies des matchs. Ils ont opté pour l'ajout d'offsets (parametrisés) aux compétences pour tenir compte de certaines caractéristiques telles la taille d'escadrons amis (*squad*), l'expérience et la tendance de quitter au cours du match (*quit penalty*). Pour les statistiques individuels, ils ont introduit la notion de «*counts*» qui est une variable aléatoire, définie à partir du performance du joueur ainsi que celles de ses coéquipiers et celles de l'équipe adverse, et qui sert introduire l'effet des statistiques individuelles sur les mise à jour des compétences. Les auteurs y ont discuté aussi la corrélation entre les différents modes de jeux et comment en profiter pour en inférer les compétences à partir et entre ces modes. Une innovation assez importante à été présentée dans ce travail, celle de l'estimation des tous paramètres, qui définissent le système et les différent modifications proposées, et ce directement et simultanément avec l'inférence des compétences.

Part II

English Version

List of Figures

2.1 Rating task 60

3.1 Example of TrueSkill factor graph for two teams. One player forms the blue team, and the red one has two players 66
3.2 The factor graph for the batch configuration 68

4.1 Score-based-TrueSkill factor graph. 73
4.2 Factor graph of team i offense vs team j defense in 5 players per team configuration 76
4.3 Collected gold vs match length for players with high skill in LoL 77
4.4 Factor graphs of inference from individual score 78
4.5 Features importance weights 80
4.6 Multi-layer configuration 81
4.7 Winning probability as a function of the number of previously played matches 82
4.8 Practice reward effect - CS:GO 83

5.1 Winning probability vs features 89

List of Tables

3.1 Table of matches 67

4.1 Offensive skill of team 1 players after updates 79

5.1 Metrics comparison for both systems 88
5.2 Betting gains comparison for both systems 88
5.3 Real and expected win probabilities for the different sub-populations relative to each of the studied feature 90
5.4 Effect of offline batch update 91

List of Symbols

$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with a mean μ and a variance σ^2
$\tilde{\mathcal{N}}(\tau, \pi)$	Normal distribution with a precision π a precision adjusted mean τ
$\mathbb{1}(.)$	Indicator function
x_i	Random variable relative to player i
$\mathbf{x}$	Vector of random variables
μ_x	Mean of normal random variable x
σ_x^2	Variance of normal random variable x
$\Phi(.)$	Complementary distribution function of standard normal random variable
$\Pr\{A\}$	Probability of event A
$f(x)$	Probability distribution function of the random variable x

Chapter 1

Introduction

1.1 Background and Preliminaries

1.1.1 Rating and Ranking: Concepts, Purpose and Motivation

Competition is embedded in the human nature. From competing little brothers to students, business men and athletes, all of them strive to do better than others. No wonder we constantly measure ourselves against our peers. We need to be able to compare and contrast competitors skill levels to establish who is the best, hence the necessity of rating and ranking contenders emerges. Sports competitions represent one of the fields that attract the most such need for rankings. That includes traditional physical sports and recently, electronic sports or eSports. Usually, direct measurements of the abilities are unavailable. The only information we possess is structured as a set of outcomes and statistics which are affected by these underlying abilities. The outcome of a match is determined by a large number of factors. Keeping track of all relevant factors precisely tends to be unrealistic. In addition to the uncertainty related to measuring these factors and their eventual non-deterministic variability, one cannot expect a certain deterministic prediction of a match outcome. Rather, a good prediction will, hence, not exactly predict the outcome, but will anticipate the odds and probabilities more precisely. This uncertainty also includes the modeling of competitors abilities and skills. Statistics and probability theory provide ways to make inference

under randomness. Therefore, modeling and predicting the results of competitive team matches naturally falls into the area of statistics and machine learning. Ratings systems serve two main purposes: skills estimation for ranking and matchmaking, as well as predicting odds of future encounters. In addition to the fact that rankings players is an objective in itself, it serves also for matching players with similar levels to ensure fun, exciting matches for the players: A game is most fun when the outcome is most uncertain, with the a priori chances of winning equal to those of losing. On the other hand, predicting winning odds is also quite important: Rating systems could be profitable when correct predictions are rewarded, like betting on the winners of competitive matches. An accurate rating system that can foresee better than other parties in terms of winning probabibilities can surely make higher profit. Hence, betting market is really interested and invested in such rating systems for both betting organizations and gamblers, each aiming to maximize their profit.

1.1.2 Evolution of Rating Systems

In sports ranking, possibly the most prominent ranking system in use today is Elo system (or one of its modified versions). This system was developed by Arpad Elo in the early 1960's, nevertheless the seminal Elo update still remains, after more than 50 years, a valuable baseline which is difficult to improve upon. The original Elo is based on a statistical model that uses the Thurstone-Mosteller model to estimate the probability of individual game outcomes, based on the assumption that the player's chess performance in each game is a random variable that is normally distributed with the same constant variance assuming true skill of each player is the mean of that player's performances. The logistic distribution version of the system goes back to Zermelo's work [1], who developed a model for paired comparisons that later became known as the Bradley-Terry model. The latter version is the most popular one. Originally, Elo developed it for the game of chess, and chess federations around the world adopted it quickly (1970). It became popular and common for many other two player games too, including Go, Scrabble, table tennis, American football, basketball, Major League Baseball and other games.

Glickman, in 1999, proposed the Glicko updating system, which improves over Elo by incorporating the variability in parameter estimates, making it the first Bayesian ranking system. To begin, prior

to a rating period, a player's skill (q) is assumed to follow a Gaussian distribution which can be characterized by two numbers: the average skill of the player (μ) and the degree of uncertainty in the player's skill (s). Then, Glicko models the game outcomes by the Bradley-Terry model and updates players' skills after a rating period. Though the Elo and Glicko ranking systems have been successful, they are designed for two-player games. In video games a game often involves more than two players or teams. To address this problem, Microsoft Research developed TrueSkill [[2]], a ranking system for Xbox Live.

Like Glicko, TrueSkill is a Bayesian ranking system as well and it uses a Gaussian belief over a player's skill, but it has few differences compared to Glicko. First and most importantly, its design allows to account for multi-team/ multi-player configurations, and it is an online learning system that updates the skills after each match rather than waiting for the end of a rating period. In addition to that, Glicko models the performance difference by a logistic distribution (Bradley-Terry model), while TrueSkill uses the Gaussian distribution (Thurstone-Mosteller model). Moreover, TrueSkill supports draws and offers a way to measure the quality of a game. The way TrueSkill estimates skills is by constructing a graphical model and using approximate message passing. In the easiest case, a two-team game, the TrueSkill update rules are fairly simple. However, for games with multiple teams and multiple players, the update rules become more complex and less tractable as they require an iterative procedure. The required number of matches played for TrueSkill algorithm to be «sure enough» about skill depends on team compositions and its developer gives a rough empirical values for the most popular set ups. TrueSkill has been extended later to estimate players' skills not only forward through time but also backward allowing future information to adjust player's past ranking by going backward in the estimation [3]. They called this extension TrueSkill Through Time, TTT. Although TrueSkill supports teams setups, the TTT was only applied and tested for two players games. Authors of TTT claimed that their new algorithm is more accurate which justifies the longer estimation time required by additional steps of going forward or backward in time. TrueSkill was later subject to another extension, notably the one incorporating team scores [4].

1.1.3 Recent Literature - Related works

Most of the previous rating systems had in common the input data: they base their estimation and predictions on binary results of matches. With the large amount of available in game data, one can wonder if it is profitable for those systems to exploit this data to enhance their prediction performance. Basing rating only on match outcomes could be considered efficient since it is already providing good results and it is applicable to all games similarly. However, this could result in neglecting and missing relevant games aspects that those systems are not tailored to capture, like in recent multiplayer games. One example is Multiplayer Online Battle Arena (MOBA) games. This genre of games provide a wide variety of in game parameters and player/team statistics that could be incorporated in modeling the skill.

In recent literature, we can see that skill modeling is treated within player modeling, which have been studied for long time by researchers such as Yannakakiset al. in [5] and Bakkes et al. in [6]. In player modeling, characteristics of human players such as strategies, preferences and skills are detected, modeled, predicted and expressed. Skill modeling differs from skill rating in that it aims to encompass player skills in multiple facets and does not necessarily need to rank players. Stanescu in his master thesis [7] modeled player skills in multi-dimensions (such as offensive and defensive abilities) but his work was limited to 1-vs-1 games rather than team based games like MOBA. Among those researches only few like Roy et al. [9] and Rahman et al. [8] tend to study team based competitions. To evaluate and verify the consistency of proposed skill components, the common procedure is to rely on prediction accuracy and predictor log likelihood. There has been some work that treated this particularly for MOBA games. Pobiedina et al. [10][11] showed that four factors contribute to team success namely team composition of champions, the number of friends, the player experience and the conformity of player nationalities. In [12], authors utilize various skill-based predictive models to decompose player skills into interpretative parts. Adopting a model-based analysis approach, they find that player skills in MOBA games could include base skill of player, base skill of champion and player's champion specific skill as three prominent components. This encourages the idea of looking to the skill from multiple angles and correlating it to the various available in game features.

1.2 Master Thesis Project

1.2.1 Objectives

In the framework of the current project, we are interested in fulfilling the following objectives:

Comprehension	Assimilating the mechanism of Bayesian inference using Gaussian message passing in graphical models based rating systems, namely TrueSkill: analyzing convergence and parameters evolution, comparisons to the perfect case and proposing possible modifications.
Modeling	Based on the existing model for expressing the skills of the players, extending it to include different features available from the game aiming more precise inference.
Data collection	Acquiring suitable datasets for evaluation containing large enough number of matches, presenting teams configuration and offering various in game stats and features.
Rating Systems	Implementing appropriate algorithms to account for the modified model and parameters with the aim of obtaining better performance than current rating systems. Choosing a suitable and representative reference to compare against.
Improvements and Evaluation	Suggestion and discussion of evaluation criteria and metrics to analyze the eventual gains in performance. the advantages and disadvantages of each approach.

1.2.2 Outline

The structure of the project derives from the list of objectives presented above, and the thesis is organized into chapters as follows:

Chapter 1	A brief summary of the evolution of rating systems has been presented, along with related work in recent literature.
Chapter 2	The rating model is presented in addition to the datasets we are using in this work.
Chapter 3	The original TrueSkill system is introduced and analyzed before the discussion of improvements and extensions.
Chapter 4	Evaluation criteria are presented followed by the discussion of simulation results.
Chapter 5	The conclusion of this thesis, where the progress made during the project is summarized and assessed.

Chapter 2

System Model and Used Data

2.1 System Model

In this section, we formulate the rating and ranking task in competitive matches and introduce the associated notations.
We model the set of matches as follows: We consider a joint random variable (S, Y) taking values in $\mathcal{S} \times \mathcal{Y}$, where $\mathcal{S}$ represents the input set of available information (lists the players involved, their team assignments and their assumed or prior skills), while $\mathcal{Y}$ is the set of observations or outcome variables. In our setting, we will consider the case $Y \in \{\text{loss,win}\} := \{0, 1\}$ where the observation from Y is a called match outcome, as well as the case $Y \in \mathbb{N}^2$, in which case we observe the final score of each opponent, or finally the case of multiple available features' scores where $Y \in \mathbb{N}^m$, m being the number of features. A match i could be simply seen as a pairwise confrontation characterized by two opponents and an observed result. The intervention of the scoring system is mainly summarized in the following: Based on its prior knowledge relative to Team 1 and Team 2 skills, $S_i(1)$ and $S_i(2)$, it predicts the match outcome in terms of winning probabilities. Then, it observes the outcome Y_i and updates its belief on skills accordingly.

Let Y_{ij} be the binary random variable which denotes whether the match ended in a victory ($Y_{ij} = 1$) or a loss ($Y_{ij} = 0$) for team i against the team j, with $i, j = 1 \dots n$, $i \neq j$. Pair comparison models describe the outcome probability as $\mathbb{Pr}(Y_{ij} = 1) = \mathbb{F}(a_i - a_j)$; where $\mathbb{F}$ is a distribution function and a_i is a parameter measuring the skill of team i. Models are commonly

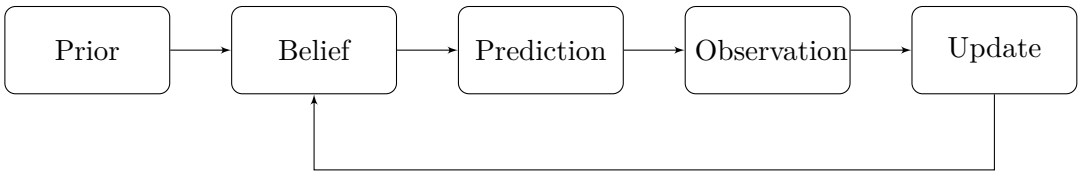


Figure 2.1 – Rating task

categorized as Bradley-Terry models (Bradley and Terry, 1952) or Thurstone-Mosteller models (Thurstone, 1927; Mosteller, 1951) depending on whether $\mathbb{F}$ is the distribution function of a logistic or of a standard normal random variable, respectively. A model-based approach to address this problem is the following: Let $\beta_i \in \mathcal{R}$ represent the "strength" of team i, and let the outcome of a game between teams (i, j) be determined by $\beta_i - \beta_j$. The Bradley-Terry model treats this outcome as an independent Bernoulli random variable with distribution Bernoulli $\mathcal{B}(p_{ij})$, where the log-odds corresponding to the probability p_{ij} that team i beats team j is modeled as: $\log \frac{p_{ij}}{1-p_{ij}} = \beta_i - \beta_j$, or equivalently, solving for p_{ij} yields

$$p_{ij}^{\mathrm{BT}} = \frac{e^{\beta_i - \beta_j}}{1 + e^{\beta_i - \beta_j}} = \frac{e^{\beta_i}}{e^{\beta_j} + e^{\beta_j}} = \frac{1}{1 + e^{-(\beta_i - \beta_j)}} = \mathcal{S}(f(\beta_i, \beta_j)), \tag{2.1}$$

where $\mathcal{S}$ is the sigmoid or logistic function and $f(\beta_i, \beta_j) = \beta_i - \beta_j$
Replacing $\mathcal{S}$ by Φ, the cumulative distribution function of a standard normal random variable, we obtain the expression for Thurstone-Mosteller models:

$$p_{ij} = \Phi(f(\beta_i, \beta_j)). \tag{2.2}$$

TrueSkill employs the latter. Since we base this work on TrueSkill framework, analyzing and discussing eventual improvements, we start from similar modeling assumptions. A player's skill is modeled as a scalar but the system knowledge relative to skill accounts for uncertainty. Thus, the prior knowledge is modeled as a normal distribution $\mathcal{N}(\mu, \sigma)$, μ being the 'true skill' and σ^2 representing the algorithm's uncertainty about its estimation. Another uncertainty is taken into consideration: a player performance in a match may be influenced by several factors and consequently fluctuates around the scalar true skill. This phenomenon is modeled as a normal distribution as well with another variance β^2. The team performance is simply the sum of individual

performances of players who make it up. In the following section we go more in details about the different modeling aspects of TrueSkill.

2.2 Data

2.2.1 Synthetic Data

To verify findings and demonstrate performance evolution, we opt to create a supervised framework with known parameters to conduct synthetic experiments as a first step. Each time a synthetic experiment is performed throughout this work, the related created data is described on the spot and the necessary details are provided. But what is in common, is that rating systems have no prior knowledge relative to generation parameters and observe only the binary match outcomes or the teams' scores. Their objective is to estimate players skills and get as close as possible to the real ranks, ensuring high prediction accuracy. A system would be considered better if it performs better according to the evaluations criteria we will present in the following sections.

2.2.2 Real-life e-Sports Data

In the last decade, a unique type of sports, namely electronic sports a.k.a eSports, emerged as a popular genre of computer games, in which competition is done online in simulated environments governed by rules and regulations similar to those found in traditional forms of sports. eSports is a rapidly growing video game market attracting tremendous numbers of professional players, developers, fanatical audiences and tournament organizers. A recent report (February 2018) released by Newzoo [18] showed that the global eSports economy will reach 905.6 million this year and grow to 1.4 billion by 2020. With millions playing simultaneously[19], such popularity becomes the basis for many event-oriented eSport betting companies[20]. We choose to acquire our datasets from this rising field, since it matches our multiplayer configuration and is suitable for our experiments. We collected data from competitive matches for 2 of the 3 most popular online multiplayer games: League of Legends (LoL), the number one MOBA game and Counter Strike Global Offense (CS:GO), the number one multiplayer first shooter game. For the latter, the dataset consists of 19415 competitive matches from various professional leagues and championships that took place between

2012 and 2017. The data was collected using a scrapper code we built to collect matches informations from HLTV.com, a popular website covering CS:GO events. LoL dataset on the other hand was published by Kaggle user *Chuck Ephron* and it comprises 17620 competitive professional matches between 2015 and early 2018, including the NALCS, EULCS, LCK, LMS, and CBLoL leagues as well as the World Championship and Mid-Season Invitational tournaments. Both datasets provide for each match, the competing teams compositions, match results as well as teams and individual statistics. This includes the number of kills, assists and deaths, the average damage received, gold collected, towers, dragons and many other in games features and statistics.

Chapter 3

TrueSkill Presentation and Analysis

TrueSkill system was designed by Microsoft affiliated researchers and it has been used in ranking and matchmaking for their online Xbox Live games. It was developed in Bayesian probabilistic framework to extend Elo and Glicko principles to multiplayer competitions. It showed better predictive abilities than its predecessors in addition to its ability to cover various match configurations ranging from individual confrontations to multi-player and multi-team setups. In this chapter we start by detailing how such system works before studying and analyzing its behavior in particular situations as part of synthetic simulations.

3.1 The TrueSkill^{TM} Rating Model

TrueSkill models the rating problem as the computation of skills estimates (as posteriors) by Bayesian inference. Formally, it defines a joint distribution over player skills and match results, conditional on few latent variables defined by the model. Then, by Bayes rule, it infers the skill estimate as posterior after each match. Let's start by defining the different system parameters. For the initial skill of a player i, TrueSkill assumes it is drawn from a normal prior distribution:

$$s_i \sim \mathcal{N}(\mu_0, \sigma_0^2), \tag{3.1}$$

μ_0 and σ_0^2 being tunable system parameters. Then after each match, the system updates its knowledge relative to these two parameters and the distribution would be updated to $\mathcal{N}(\mu_i, \sigma_i^2)$

where μ_i can be seen as the «true skill» and σ_i^2 as the variance that characterizes how certain the algorithm is about the accuracy of its estimate. The message passing algorithm ensures that after each match, the variance is meant to decrease (if the model does not artificially increase it, which will be discussed in the next section). After each match, the system assumes that player skills evolve over time according to a random walk, meaning that an increase or decrease in skill is assumed equally likely. This is modeled by an additional zero-mean Gaussian variable with a small variance τ^2, added between matches. TrueSkill assumes another source of skill variability: Each player has a real-valued performance, p_i, in each match, drawn according to:

$$f(p_i) = \mathcal{N}(p_i; s_i, \beta^2). \tag{3.2}$$

The variance β^2 is assumed to be a global (for all players) tunable parameter that reflects the amount of randomness in the game and its effects on players skills. The performance p_i could be expressed in terms of the initial parameters, μ_i and σ_i^2, by simply integrating over the possible S_i as follows:

$$f(p_i|\mu_i, \sigma_i^2) = \int_{-\infty}^{\infty} \mathcal{N}(p_i; s_i, \beta^2)\ \mathcal{N}(s_i; \mu_i, \sigma_i^2)\ ds_i. \tag{3.3}$$

The next step, player performances are combined into team performances. TrueSkill assumes that team performance is simply the sum of performances of its players:

$$t_{team_i} = \sum_{j \in \text{team}_i} p_j \sim \mathcal{N}(t_i; \sum_{j \in \text{team}_i} s_i, \beta^2). \tag{3.4}$$

Team performances are then compared, $d = t_{team_i} - t_{team_j}$, and the winner expected by TrueSkill is the team with the largest performance. If draws are possible, TrueSkill system covers that case by introducing "drawing margin" concept through a new global parameter, $\epsilon > 0$ and declares draws when team performances difference is less than ϵ, $|t_1 - t_2| \leq \epsilon$. However we are not focusing on this point since the games we study are not susceptible to end up in draws. The actual match results are in a binary discrete form, $y \in \{0, 1\}$, which need to be "adjusted" to fit into the Gaussian message passing algorithm. The objective would be to compute the posterior distribution over the skills given the outcome, $f(\mathbf{s}|y)$. For this purpose, Bayes rule is used and the needed joint probability of the model can be written as:

$$f(y, \mathbf{d}, \mathbf{t}, \mathbf{p}, \mathbf{s}) = f(y|\mathbf{d})\ f(\mathbf{d}|\mathbf{t})\ f(\mathbf{t}|\mathbf{p}) f(\mathbf{p}|\mathbf{s})\ f(\mathbf{s}), \tag{3.5}$$

where the bold notation is for vector variables that include all players. For example, the performance vector $\mathbf{p} = [p_1, p_2, \ldots, p_l]$, l being the number of players in the game. The likelihood can be expressed as follows:

$$f(y|\mathbf{s}) = \int\int\int f(y, \mathbf{d}, \mathbf{t}, \mathbf{p}, \mathbf{s})\ d\mathbf{d}\ d\mathbf{t}\ d\mathbf{p}, \tag{3.6}$$

to finally have, by Bayes:

$$f(\mathbf{s}|y) = \frac{f(y|\mathbf{s})\ f(\mathbf{s})}{\int f(y|\mathbf{s})\ f(\mathbf{s})\ d\mathbf{s}}. \tag{3.7}$$

Therefore, the problem reduces to a marginalization problem, which is one of the basic subjects of Bayesian analysis theory [2][13]. This problem is solved by the well-known marginalization message passing algorithm (a sample factor graph could be found in Figure 3.1) with an additional trick which is the "approximate" message passing in the bottom part of the graph. Almost all the distributions are Gaussian, leading to easy computations where the message passing is performed from top to bottom and back. However, the bottom node (e.g., the $\mathbb{1}(d_1 > \epsilon)$ node on Figure 3.1), which is a discrete step function, introduces complication and does not propagate back easily. To solve this issue and adapt to the Gaussian message passing framework, this distribution is approximated with a Gaussian distribution by moment matching (i.e., by computing and matching with the first two moments), and the message passing algorithm goes on along the bottom part of the graph until convergence. The method that specifies this approximation and convergence criteria is Expectation Propagation [14]. That way, the information acquired from the comparison and match result is propagated upward to update the players' skills after each match. Those posterior skills are used then as the prior for the next game [15].

The message passing for continuous variables is characterized by the following equations:

$$p(v_k) = \prod_{f \in F_{v_k}} m_{f \to v_k}(v_k) \tag{3.8}$$

$$m_{f \to v_k}(v_k) = \int \ldots \int f(\mathbf{v}) \prod_{i \neq j} m_{v_i \to f}(v_i) d\mathbf{v}_{\setminus j} \tag{3.9}$$

$$m_{v_k \to f}(v_k) = \prod_{\tilde{f} \in F_{v_k} \setminus \{f\}} m_{\tilde{f} \to v_k}(v_k) \tag{3.10}$$

where F_{v_k} denotes the set of factors connected to the variable v_k and $\mathbf{v}_{\setminus j}$ denotes the components of the vector $\mathbf{v}$ except its j^{th} component.

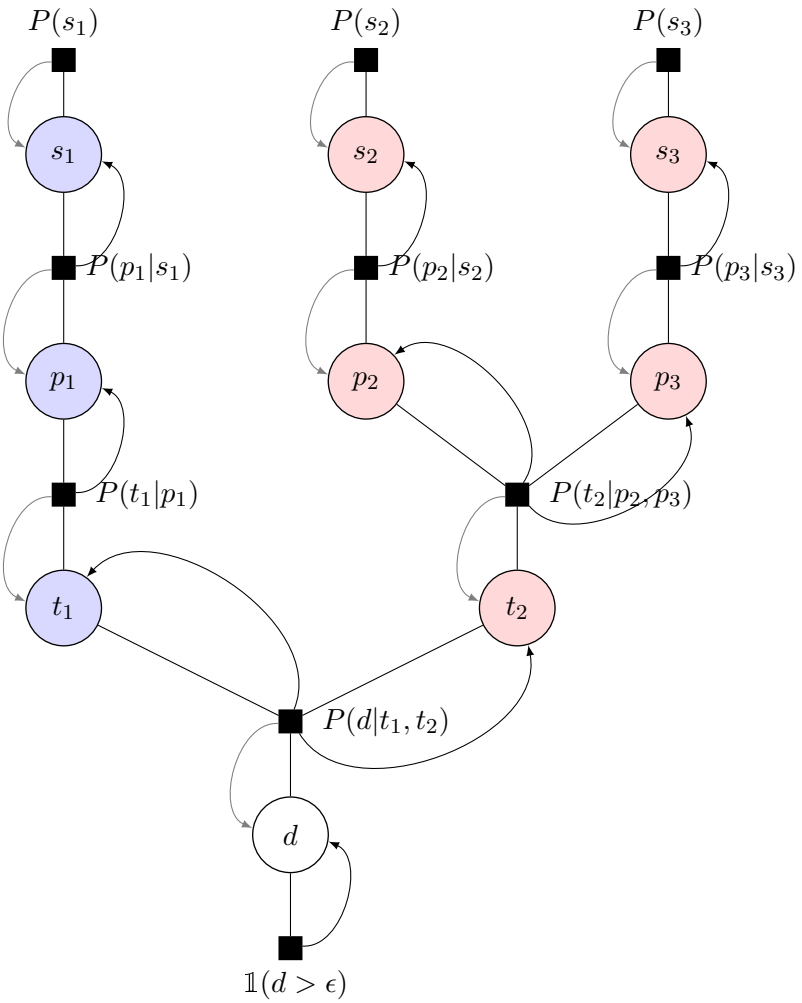


Figure 3.1 – Example of TrueSkill factor graph for two teams. One player forms the blue team, and the red one has two players

3.2 Synthetic Study Case

TrueSkill is an online update algorithm that bases its estimation on the previous one through the time. This online aspect is useful as it is a real-time approximation to the posterior distribution over skills. However, this may induce some mistake. Processing all matches together in what we call batch computation is the most accurate skill ratings. To see an example of a situation where the online sequential aspect leads to an incorrect ranking, let us consider a 7-matche tournament involving 8 players, $p_i, i \in \{1, 2, 3, 4, 5, 6, 7, 8\}$, as follows:

where matches occur in that specific order and players are assumed to initially have the same prior distribution. Let us consider the updates made by TrueSkill. The first 4 results imply that

Match	Player 1	Player 2	Winner
1	p_1	p_2	p_1
2	p_3	p_4	p_3
3	p_5	p_6	p_5
4	p_7	p_8	p_7
5	p_3	p_2	p_2
6	p_5	p_4	p_4
7	p_7	p_6	p_6

Table 3.1 – Table of matches

players p_1, p_3, p_5 and p_7, who won against opponents with same skills, have the same updated skills. Same goes for p_2, p_4, p_6 and p_8 after losing to equivalent opponents. In the following 3 matches, the winning players, p_2, p_4 and p_6, started with the same skills and thus end up with similar skills since they played against equivalent opponents. Again, same goes for the loser side. The online updates result then in having $p_2 = p_4 = p_6$ and $p_3 = p_5 = p_7$. This is certainly not right. For instance, when considering a batch treatment, i.e, all matches at the same time, the actual ranking would be $p_1 > p_2 > p_3 > p_4 > p_5 > p_6 > p_7 > p_8$. We can see that using the following reasoning: p_8 is the only player who did not win any matches, thus, he is the weakest and he resides at the bottom of the table. All others won exactly one match. p_7 is the only one who achieved his win against the weakest p_8, and therefore he is the second weakest. p_6 is the only one who achieved his win against the second weakest p_7 making him the third in the weakest line. And using the same reasoning we end up with the batch ranking we previously stated.

Another observation relative to the comportment of the online skill estimation via Expectation Propagation over graphs, is the eventual possible creation of «loopy» like situation and the over-confident variance estimation. To see this effect, let's consider two players characterized by skills s_1 and s_2 who play against each others for N matches. The prior distribution is $\mathcal{N}(\mu_0, \sigma_0^2)$.

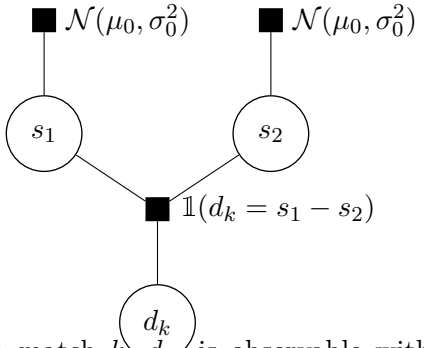


Figure 3.1 – Simplified 1vs1 match factor graph

Let's assume that $\beta = 0$ and that the difference variable for a match k, d_k, is observable with variance σ_d^2. The factor graph for a single match reduces therefore to 3.1. For the online updates of TrueSkill, the new prior for each match is the posterior skill computed in the previous match. For batch graph on the other hand, the two skills variables are attached in the same time to all factors linking them to all the difference variables, $d_k, k = 1..N$, as it can be seen in Figure 3.2.

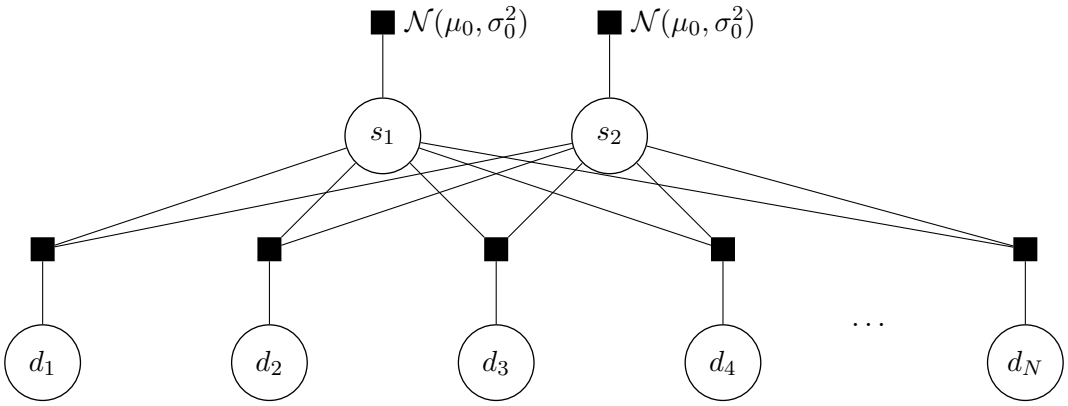


Figure 3.2 – The factor graph for the batch configuration

We have $d_k \propto \mathcal{N}\left(d_k; s_1 - s_2, \sigma_d^2\right)$ for all matches $k = 1, .., N$. By the chain rule of probability, we get the joint probability of all the nodes in the graph as the following product:

$$P(s_1, s_2, \mathbf{d}) = \prod_{k=1}^{N} P(d_k|s_1, s_2)\ P(s_1)\ P(s_2). \tag{3.11}$$

Dividing out the marginal over the differences $\mathbf{d}$ we get the joint skills distribution conditional on $\mathbf{d}$, or the posterior. It is proportional to:

$$\begin{aligned} P(s_1, s_2|\mathbf{d}) &\propto \prod_{k=1}^{N} P(d_k|s_1, s_2)\ P(s_1)\ P(s_2) \\ &= \prod_{k=1}^{N} \mathcal{N}\left(d_k; s_1 - s_2, \sigma_d^2\right)\ \mathcal{N}\left(s_1; \mu_0, \sigma_0^2\right)\ \mathcal{N}\left(s_2; \mu_0, \sigma_0^2\right). \end{aligned} \tag{3.12}$$

The product of likelihoods could be calculated as follows:

$$\begin{aligned} \prod_{k=1}^{N} \mathcal{N}\left(d_k; s_1 - s_2, \sigma_d^2\right) &= \prod_{k=1}^{N} \mathcal{N}\left(s_2; s_1 - d_k, \sigma_d^2\right) \\ &= \prod_{k=1}^{N} \tilde{\mathcal{N}}\left(s_2; \frac{s_1 - d_k}{\sigma_d^2}, \sigma_d^{-2}\right) \\ &\propto \tilde{\mathcal{N}}\left(s_2; \sum_{k=1}^{N} \frac{s_1 - d_k}{\sigma_d^2}, N\sigma_d^{-2}\right) \\ &= \mathcal{N}\left(s_2; s_1 - \sum_{k=1}^{N} \frac{d_k}{N}, \frac{\sigma_d^2}{N}\right) \\ &= \mathcal{N}\left(s_2; s_1 - \bar{d}, \frac{\sigma_d^2}{N}\right), \end{aligned} \tag{3.13}$$

where $\bar{d} = \frac{1}{N}\sum_{k=1}^{N} d_k$. Back to the posterior, we marginalize over s_2 to obtain:

$$\begin{aligned}
P(s_1|\mathbf{d}) &= \int P(s_1, s_2|\mathbf{d})\ ds_2 \\
&\propto \int \mathcal{N}(\left(s_2; s_1 - \bar{d}, \frac{\sigma_d^2}{N}\right)\ \mathcal{N}\left(s_1; \mu_0, \sigma_0^2\right)\ \mathcal{N}\left(s_2; \mu_0, \sigma_0^2\right)\ ds_2 \\
&= \mathcal{N}\left(s_1; \mu_0, \sigma_0^2\right) \int \mathcal{N}\left(s_2; s_1 - \bar{d}, \frac{\sigma_d^2}{N}\right)\ \mathcal{N}\left(s_1; \mu_0 + \bar{d}, \sigma_0^2 + \frac{\sigma_d^2}{N}\right)\ ds_2 \\
&= \mathcal{N}\left(s_1; \mu_0, \sigma_0^2\right)\ \mathcal{N}\left(s_1; \mu_0 + \bar{d}, \sigma_0^2 + \frac{\sigma_d^2}{N}\right) \int \mathcal{N}\left(s_2; s_1 - \bar{d}, \frac{\sigma_d^2}{N}\right)\ ds_2 \\
&= \tilde{\mathcal{N}}\left(s_1; \frac{\mu_0}{\sigma_0^2} + \frac{\mu_0 + \bar{d}}{\sigma_0^2 + \frac{\sigma_d^2}{N}}, \frac{1}{\sigma_0^2 + \frac{\sigma_d^2}{N}} + \frac{1}{\sigma_0^2}\right) \\
&= \mathcal{N}\left(s_1; \mu_0 + \frac{\bar{d}\sigma_0^2}{2\sigma_0^2 + \frac{\sigma_d^2}{N}}, \frac{\sigma_0^2(\sigma_0^2 + \frac{\sigma_d^2}{N})}{2\sigma_0^2 + \frac{\sigma_d^2}{N}}\right).
\end{aligned} \tag{3.14}$$

We can see that when $N \longrightarrow \infty$, the distribution tends to $\mathcal{N}\left(s_1; \mu_0 + \frac{\bar{d}}{2}, \frac{\sigma_0^2}{2}\right)$
On the other hand, when we perform the exact computation using the online updates, we observe that the variance estimate goes below the batch value and tends to zero as N tends to infinity:

$$\frac{1}{\sigma_1^2(t+1)} = \frac{1}{\sigma_1^2(t)} + \frac{1}{\sigma_2^2(t) + \sigma_d^2(t)} \Rightarrow \sigma_1^2(N) = \left(\frac{1}{\sigma_1^2(0)} + \sum_{t}^{N} \frac{1}{\sigma_2^2(t) + \sigma_d^2(t)}\right)^{-1} \xrightarrow{N\to\infty} 0. \tag{3.15}$$

This is explained by the online sequential aspect of the Expectation Propagation on this 'loopy' like graph which leads to an overconfident estimation of the variance, as proved in [16]. TrueSkill avoids this issue by artificially increasing the variance between match by adding an additional variance τ^2 we mentioned in the previous section.

Chapter 4

TrueSkill 1.1, the Proposed Improvements

In this chapter, we propose to detail the various modifications and extensions brought to the original TrueSkill algorithm with the intention of improving its classification and prediction capabilities, taking into account the specificity of the available games and data sets. A few modification ideas have been suggested recently, starting with a brief presentation of two of them, Score based TrueSkill [4] and TrueSkill Through Time [3], we then proceed with how we adapt them in our new configuration to estimate competences using the many statistics and properties available in our data sets in order to improve performance by optimizing our evaluation criteria.

4.1 Use of in-Game Additional Information

TrueSkill system bases its belief regarding players skills on match outcomes solely, which has the advantage of being general and applicable to all game genres whose outcome space is {win, loss, draw}. However, this approach could be discarding some useful information. Most online multiplayer games today provide a multitude of *features*, which could be defined as any measurable individual property or characteristic of a phenomenon observed during the game. This includes scores and statistics, both individual and collective, collected throughout the game. The latter provides an important potential source of information and insights regarding players' skills. In this

section, we aim to leverage this additional information to augment the original TrueSkill system to capture a broader aspect of players' skills.

4.2 Score Use, Offensive and Defensive Skills Model

Authors of [4] had a similar idea when they decided to make use of score outcome of games like football. At the end of such games, each team ends up with a score and the one with higher score is the winner. For this purpose they introduced the "Offensive and Defensive Skills Model" to exploit both scores: In a match between two teams i and j producing respective scores $\alpha_i \in \mathbb{Z}$ and $\alpha_j \in \mathbb{Z}$ for each team, the idea is to think of α_i as resulting from i's offensive skill $s_i^o \in \mathbb{R}$ and j's defensive skill $s_j^d \in \mathbb{R}$ and likewise for j's score as a result of j's offense $s_j^o \in \mathbb{R}$ and i's defense $s_i^d \in \mathbb{R}$. The objective is then to infer the offensive and defensive skills given the observed score and the prior knowledge.Assuming the score α_i depend only on s_i^o and s_j^d and α_j only on s_j^o and s_i^d, i.e., $f(\alpha_i, \alpha_j | s_i^o, s_j^o, s_i^d, s_j^d) = f(\alpha_i | s_i^o, s_j^d)\ f(\alpha_j | s_j^o, s_i^d)$, and assuming that the joint marginal over skill priors independently factorizes as $f(s_i^o, s_j^o, s_i^d, s_j^d) = f(s_i^o)\ f(s_j^o)\ f(s_i^d)\ f(s_j^d)$, Bayes rule gives the following posterior:

$$\begin{aligned} f(s_i^o, s_j^o, s_i^d, s_j^d | \alpha_i, \alpha_j) &\propto f(\alpha_i, \alpha_j | s_i^o, s_j^o, s_i^d, s_j^d)\ f(s_i^o, s_j^o, s_i^d, s_j^d) \\ &\propto \underbrace{f(\alpha_i | s_i^o, s_j^d)\ f(s_i^o)\ f(s_i^d)}_{f(s_i^o, s_j^d | \alpha_i)}\ \underbrace{f(\alpha_j | s_j^o, s_i^d)\ f(s_j^o)\ f(s_j^d)}_{f(s_j^o, s_i^d | \alpha_j)}. \end{aligned} \tag{4.1}$$

This way, estimating the joint posterior reduces to two independent inference problems, namely:

$$f(s_i^o, s_j^d | \alpha_i) \propto f(\alpha_i | s_i^o, s_j^d)\ f(s_i^o)\ f(s_i^d), \text{and} \tag{4.2}$$

$$f(s_j^o, s_i^d | \alpha_j) \propto f(\alpha_j | s_j^o, s_i^d)\ f(s_j^o)\ f(s_j^d). \tag{4.3}$$

All skills priors are assumed to be Gaussian and are subject to performance fluctuations over each match which is modeled by the additional variance β: A team k exhibits offense performance $p_k^o \sim \mathcal{N}(s_k^o, \beta^2)$ and defense performance $p_k^d \sim \mathcal{N}(s_k^d, \beta^2)$. Performances affect the score as follow: Offense promotes the scoring rate while the defense inhibits it, in such a way the difference $p_k^o - p_h^d$ is the effective scoring rate of the offense against the defense. Scores are assumed to be drawn from either Poisson or Gaussian distributions. The factor graph underling the latter inference problem is

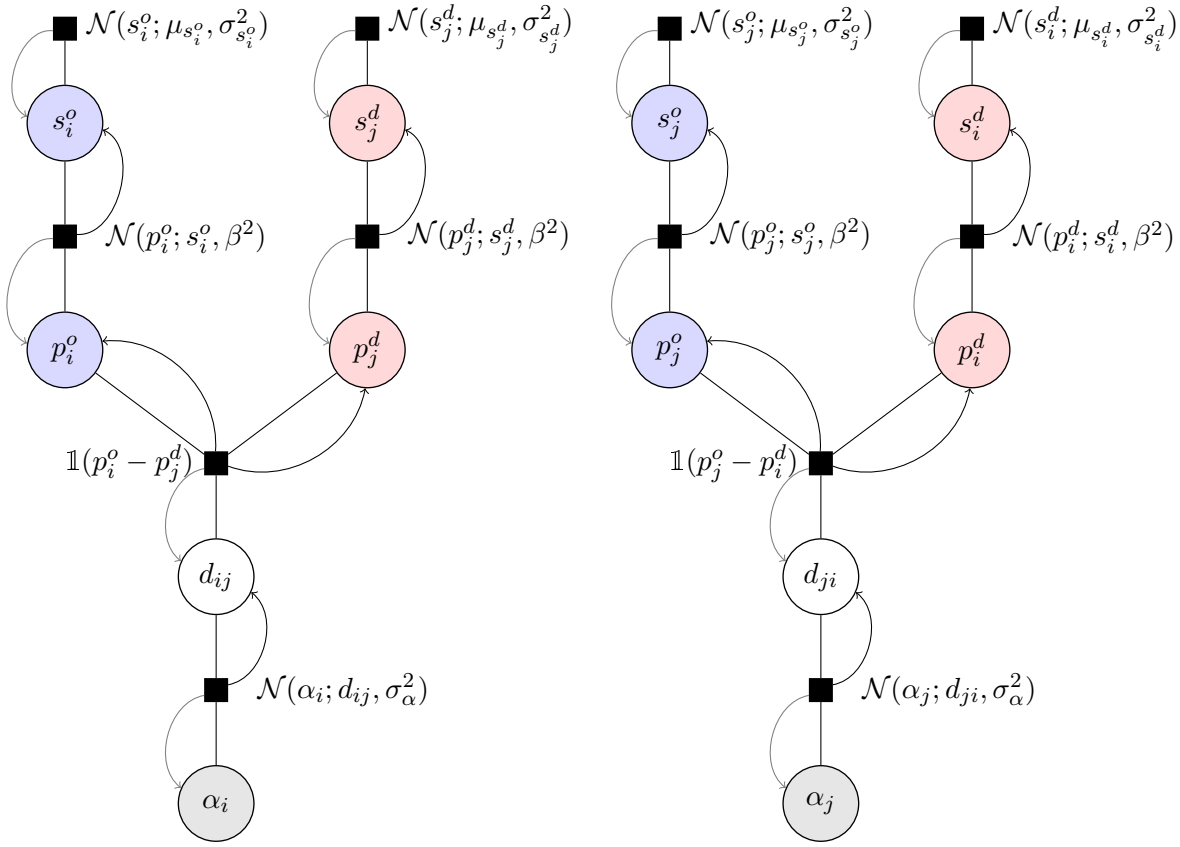


Figure 4.1 – Score-based-TrueSkill factor graph.

Scores α_i and α_j are modeled in the left graph and the right graph respectively. The gray variables are the observed ones. For each team $k \in \{i, j\}$, it is characterized by offensive skill s_k^o and defensive skill s_k^d . The posterior distributions over $(s_i^o; s_j^d)$ and $(s_j^o; s_i^d)$ are inferred via message passing.

presented in Figure 4.1. The update equations for messages are the same as in [2] with the exception of the marginal distribution over the performance difference variable d and the message passed from the factor joining it to the score variable. In the case of Poisson distributed score, approximate belief updates via variational Bayesian inference are established by the authors to approximate it by Gaussian and ensure tractability. When scores are modeled as Gaussian variables, this secures that all messages are Gaussian and therefore it is possible to compute the belief update in closed-form

as follows:

$$\sigma^2_{o_i}(t+1) = \left(\frac{1}{\sigma^2_{o_i}(t)} + \frac{1}{2\beta^2 + \gamma^2 + \sigma^2_{d_j}(t)}\right)^{-1}, \tag{4.4}$$

$$\mu_{o_i}(t+1) = \sigma^2_{o_i}(t+1)\left(\frac{\mu_{o_i}(t)}{\sigma^2_{o_i}(t)} + \frac{\mu_{d_j}(t) + \alpha_i}{2\beta^2 + \gamma^2 + \sigma^2_{d_j}(t)}\right)^{-1}, \tag{4.5}$$

$$\sigma^2_{d_j}(t+1) = \left(\frac{1}{\sigma^2_{o_i}(t)} + \frac{1}{2\beta^2 + \gamma^2 + \sigma^2_{o_i}(t)}\right)^{-1}, \tag{4.6}$$

$$\mu_{d_j}(t+1) = \sigma^2_{d_j}(t+1)\left(\frac{\mu_{d_j}(t)}{\sigma^2_{d_j}(t)} + \frac{\mu_{o_i}(t) - \alpha_j}{2\beta^2 + \gamma^2 + \sigma^2_{o_i}(t)}\right)^{-1}, \tag{4.7}$$

where the values of variables at t and $t+1$ are the prior and posterior values, respectively.The update equations for team j's offensive skill s^o_j and team i's defensive skill s^d_i, are derived similarly. To compute the winning probability of team i facing team j, message passing is used to estimate the normal distributions of score variables α_i and α_j , and then the wanted probability is of that of $\alpha_i - \alpha_j > 0$, i.e., team i's score is larger than team j's. Given $\alpha_i \propto \mathcal{N}\left(\mu_{\alpha_i}, \sigma^2_{\alpha_i}\right)$ and $\alpha_j \propto \mathcal{N}\left(\mu_{\alpha_j}, \sigma^2_{\alpha_j}\right)$, the winning probability of team i is the following:

$$\Pr\{\text{team}_i \text{ wins}\} = f(\alpha_i - \alpha_j > 0) = \Phi\left(\frac{\mu_{\alpha_i} - \mu_{\alpha_j}}{\sigma^2_{\alpha_i} + \sigma^2_{\alpha_j}}\right), \tag{4.8}$$

where Φ is the standard normal complementary distribution function. The latter equation could be expressed using the input variables, i.e., the skills priors using message passing updates to have this new expression:

$$\Pr\{\text{team}_i \text{ wins}\} = \Phi\left(\frac{\mu_{s^o_i} + \mu_{s^d_i} - \mu_{s^o_j} - \mu_{s^d_j}}{\sigma^2_{s^o_i} + \sigma^2_{s^d_i} + \sigma^2_{s^o_j} + \sigma^2_{s^d_j} + 4\beta^2}\right), \tag{4.9}$$

which allow us to have the following interpretation: the sum of offensive and defensive skills could be seen as the general or overall skill that allows us to recover the same winning probability as in the original TrueSkill:

$$\Pr\{\text{team}_i \text{ wins}\} = \Phi\left(\frac{\mu_{s_i} - \mu_{s_j}}{\sigma^2_{s_i} + \sigma^2_{s_j} + 4\beta^2}\right), \tag{4.10}$$

where $s_i = s^o_i + s^d_i$ and $s_j = s^o_j + s^d_j$.

4.3 In-Game Statistics and Features

Game results are not always determined by scores. League of Legends , one of the two games we are considering for this work, is one example. A team is declared as winners only when they destroy their enemy nexus, independently of any scores. The game itself doesn't provide a final match score. On the other hand, both our games, like almost all recent online multiplayer games, yield various in-game statistics and features for teams and players (individually). The idea is to use those, instead of and similarly to scores, to augment the original TrueSkill and capture new skills and rating aspects.

4.3.1 Partial Skills

Matches' final scores could be seen as quantitative teams proprieties that characterize their collective performance during the game. From this prospective, teams statistics and features undertake exactly the same role, providing quantitative insights on how the teams carried out their game. For this reason, we treat statistics like the amount of gold collected, the number of kills scored, the average damage received and others, we treat them as if we are dealing with scores in the model presented previously, but they are rather "partial scores". Thus, for each feature we define an associated partial skill that is updated with the feature as score using the same rules we defined in the previous subsection. The sole difference is actually the number of players, since it is a 5-player teams configuration that characterizes our games. The factor graph for a feature ϕ_i of a team i involving the offense of the latter and defense of a team j is presented in Figure 4.2. Features' scores are modeled as normal distributed and not Poisson random variables as suggested in [4]. We discarded the latter since we noticed that most of our statistics do not present the same mean and variance as they vary in time, as we can see for the gold for example in Figure 4.3, which do not correspond to Poisson distribution, characterized by a variance equal to the mean. It is also worth mentioning that, we do not rely on the comparison of features' scores to decide the winner but rather the actual match result: Using the first would be aiming to track the ability of scoring that particular feature, not the ability to win that we seek to estimate. Having the higher score of a particular statistic does not always imply the winner. Let's take the number of kills as example, in CS:GO there is a high correlation between the two options, since killing is the most deciding factor of the game. However, in LoL, correlation between win and kills is way lower, you even can

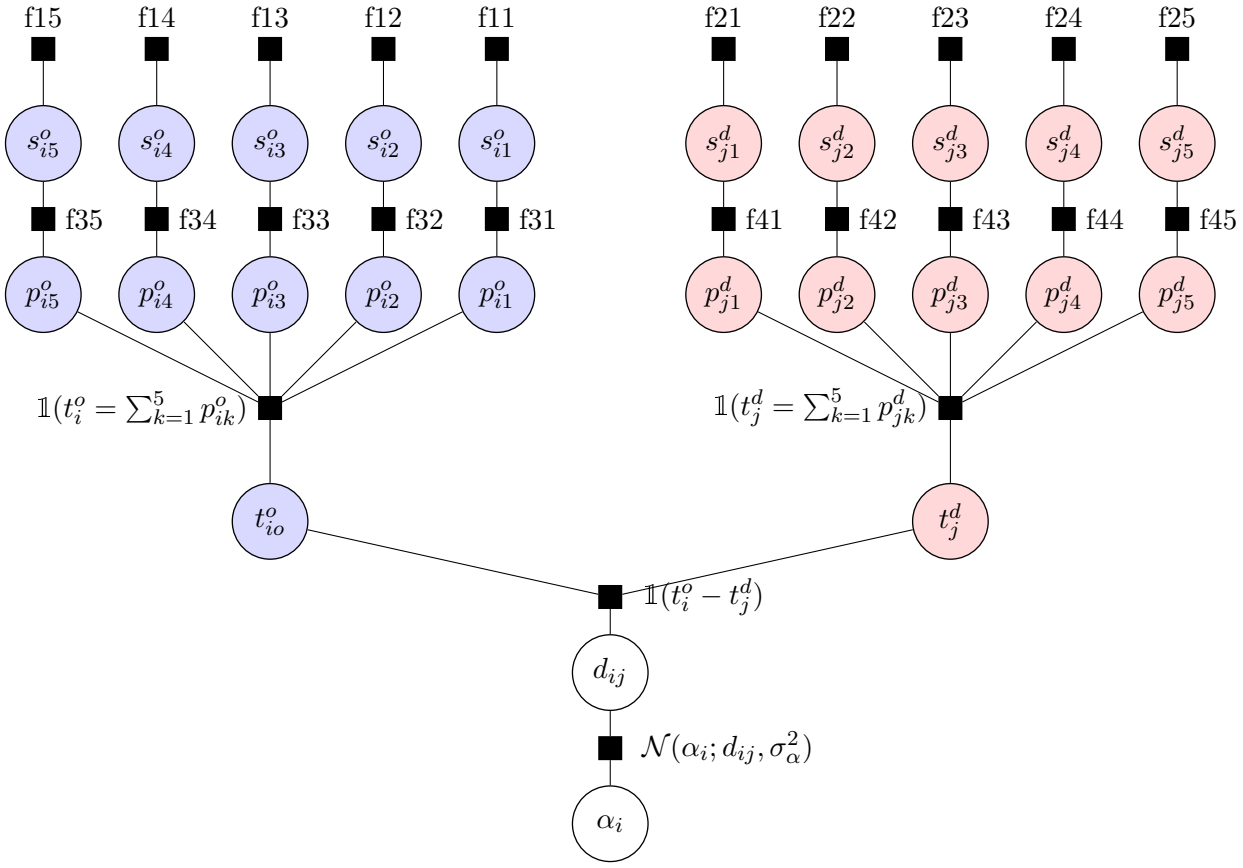


Figure 4.2 – Factor graph of team i offense vs team j defense in 5 players per team configuration

win with lower number of kills or even no kills. That's why we rely on the actual match outcome instead of the sign of score difference in order to have a partial skill that characterizes the winning ability correlated by the associated feature rather than the ability to score itself.

4.3.2 Individual Stats Model

In addition to teams statistics, we want to make use of players' statistics. When it comes to those, things are a bit different. It is no longer a team propriety that could be qualified as team score. The first idea that comes to mind is to sum up the individual statistics of each player on the team and make it into a new collective score and we apply what we already established. This seems to be the easiest solution to solve the issue, however, again, this is discarding important information and could be "unfair" to some players: When a CS:GO player, for example, realizes most of his team kills while his teammate didn't make any, using the sum kills as a collective score gives the same update for both players. However, if the rating system would have been fairer and had been

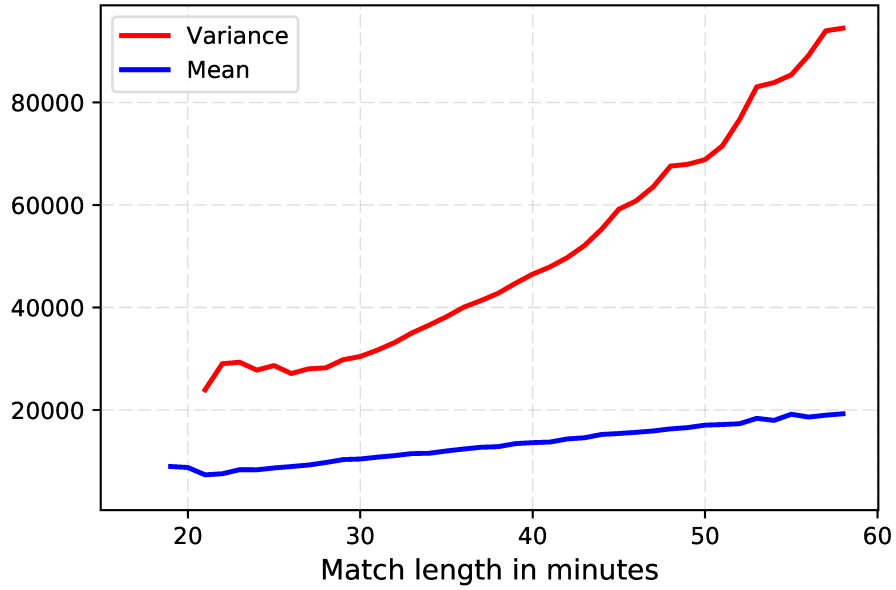
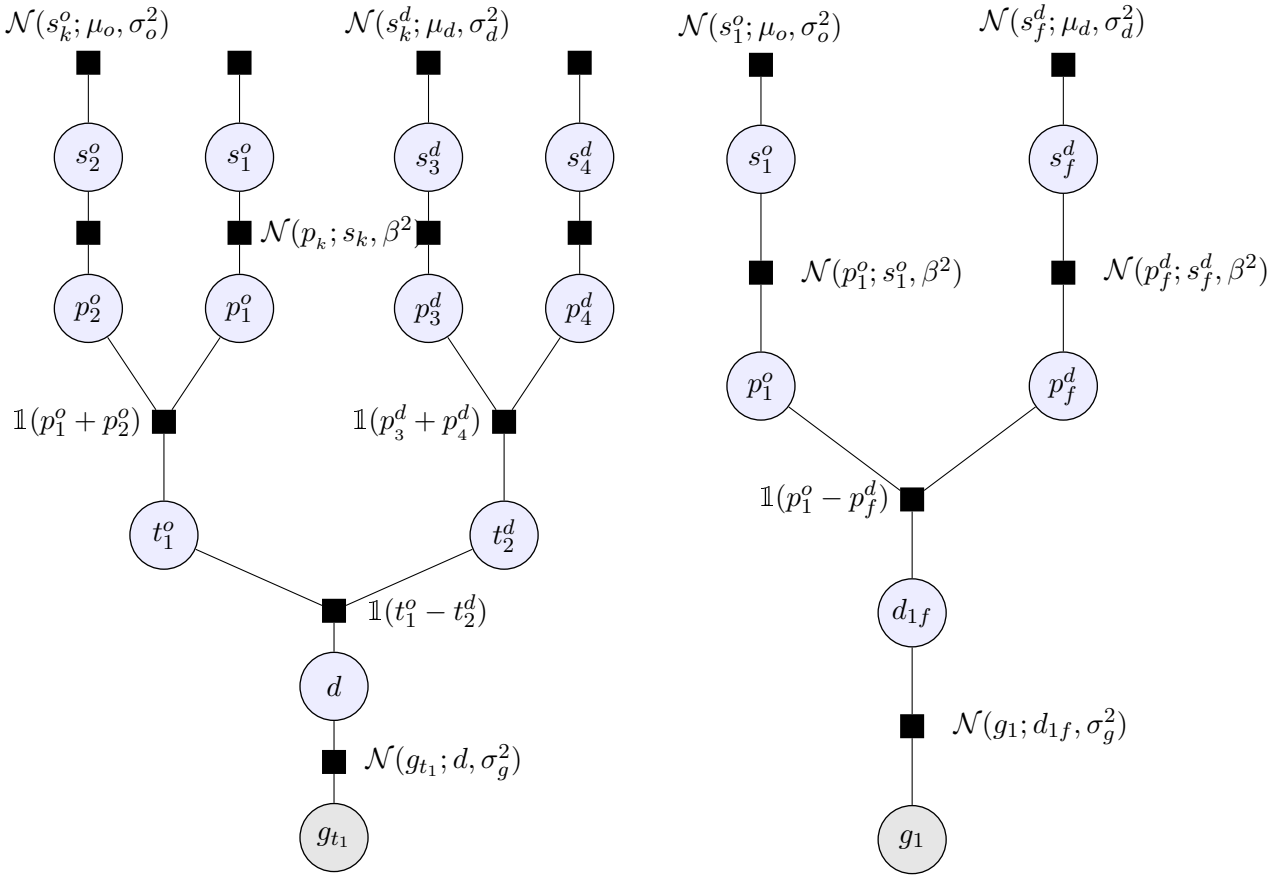


Figure 4.3 – Collected gold vs match length for players with high skill in LoL

more representative, the first player should have got more credit, proportional to his contribution in the team victory. To address this problem, we suggest the following modification to the model: each individual feature score is the result of the confrontation between the offensive skill of the concerned player and the defensive skill of a fictive player defined by the average skill of the opposing team. This is applied for each player of both teams. This way, all individual stats are used separately and each player's update is weighted by his own performance instead of team's score as a whole entity. To see this closely, let's consider a synthetic simulation where we assume a match opposing two teams of two players: $t_1 = \{j_1, j_2\}$ and $t_2 = \{j_3, j_4\}$. Their offensive and defensive skills are noted, respectively, s_k^o and s_k^d for $k \in \{1, 2, 3, 4\}$. We presume also that t_1 won the game and that players' prior skills are, $s_k^o \sim \mathcal{N}(\mu_{ko}, \sigma_{ko}^2)$ and $s_k^d \sim \mathcal{N}(\mu_{kd}, \sigma_{kd}^2)$ for all players $k \in \{1, 2, 3, 4\}$. Gold g is the feature of concern and players j_1 and j_2 had collected g_1 and g_2 golds, respectively. Let's focus on the offensive skill correlated to gold for players j_1 and j_2 to compare the two approaches of using the individual stats: summing them (Approach 1) or applying the idea we suggest (Approach 2). The related factor graphs are presented in Figure 4.4.



(a) First approach: summing individual scores $g_{t1} = g_1 + g_2$

(b) Second approach: each player vs fictive player f as average opposing team

Figure 4.4 – Factor graphs of inference from individual score

Analytically, applying the message passing updates on those graphs we have the following update for players 1 and 2 according to: Approach 1:

$$\sigma_{1o}^{2\ \text{new}} = \frac{\sigma_{1o}^2(\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2)}{\sigma_{1o}^2+\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2}, \quad \mu_{1o}^{\text{new}} = \mu_{1o} + \frac{\sigma_{1o}^2(g_{t1}+\mu_{3d}+\mu_{4d}-\mu_{2o}-\mu_{1o})}{\sigma_{1o}^2+\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2} \tag{4.11}$$

$$\sigma_{2o}^{2\ \text{new}} = \frac{\sigma_{2o}^2(\sigma_{1o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2)}{\sigma_{1o}^2+\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2}, \quad \mu_{2o}^{\text{new}} = \mu_{2o} + \frac{\sigma_{2o}^2(g_{t1}+\mu_{3d}+\mu_{4d}-\mu_{2o}-\mu_{1o})}{\sigma_{1o}^2+\sigma_{2o}^2+\sigma_{3d}^2+\sigma_{4d}^2+4\beta^2+\sigma_g^2}. \tag{4.12}$$

Approach 2:

$$\sigma_{1o}^{2\ \text{new}} = \frac{\sigma_{1o}^2(\sigma_{3d}^2/2 + \sigma_{4d}^2/2 + 2\beta^2 + \sigma_g^2)}{\sigma_{1o}^2 + \sigma_{3d}^2/2 + \sigma_{4d}^2/2 + 2\beta^2 + \sigma_g^2}, \quad \mu_{1o}^{\text{new}} = \mu_{1o} + \frac{\sigma_{1o}^2(g_1 + \mu_{3d}/2 + \mu_{4d}/2 - \mu_{2o})}{\sigma_{1o}^2 + \sigma_{3d}^2/2 + \sigma_{4d}^2/2 + 2\beta^2 + \sigma_g^2} \tag{4.13}$$

$$\sigma_{2o}^{2\ \text{new}} = \frac{\sigma_{2o}^2(\sigma_{3d}^2/2 + \sigma_{4d}^2/2 + 2\beta^2 + \sigma_g^2)}{\sigma_{2o}^2 + \sigma_{3d}^2/2 + \sigma_{4d}^2/2 + 2\beta^2 + \sigma_g^2}, \quad \mu_{2o}^{\text{new}} = \mu_{2o} + \frac{\sigma_{2o}^2(g_2 + \mu_{3d}/2 + \mu_{4d}/2 - \mu_{1o})}{\sigma_{2o}^2 + \sigma_{3d}^2/2 + \sigma_{4d}^2/2 + 2\beta^2 + \sigma_g^2}. \tag{4.14}$$

To be able to quantitatively perceive the contribution of the second approach, let's assume all players have the same priors for offense and defense, $s_k^o \sim \mathcal{N}(25, \frac{25}{3}^2)$ and $s_k^d \sim \mathcal{N}(25, \frac{25}{3}^2)$ for all players $k \in \{1, 2, 3, 4\}$. Let $g_1 = 7$ and $g_2 = 3$ to portray that player 1 had more impact and contribution. The updated offensive skills for both teammates according to the two approaches are presented in the following table:

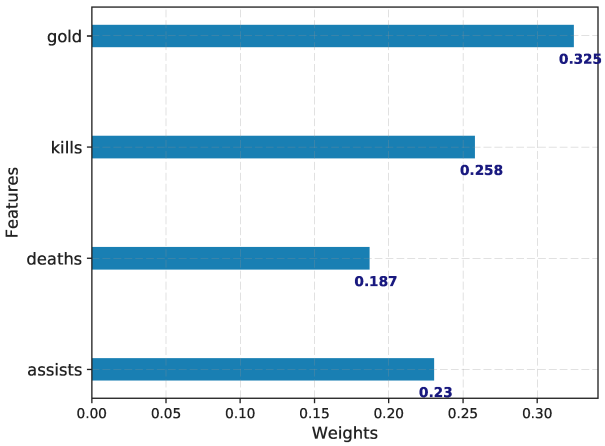
		Approach 1		Approach 2	
	Gold	μ	σ^2	μ	σ^2
Player 1	7	26.890	7.505^2	27.508	6.676^2
Player 2	3	26.890	7.505^2	26.075	6.676^2

Table 4.1 – Offensive skill of team 1 players after updates

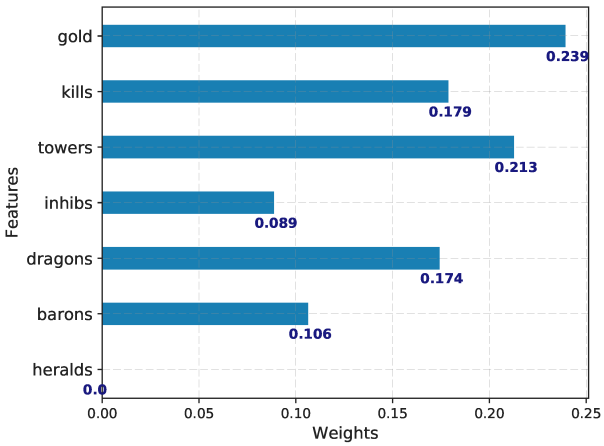
We can see that both teammates end up with same skill update in the first approach, with the skill of player 2 being inflated and player 1's being underestimated. On the other hand, the second approach does ensure a fair update that takes the individual contribution into account, rather than the uniform update.

4.3.3 Aggregation of Partial Skills

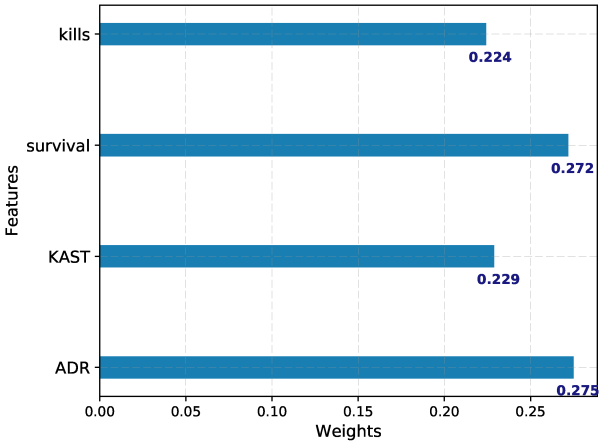
For both our games, we have multiple features from which we derive multiple partial skills. Since features do not have the same impact one game result, some being more decisive then others, we quantify the importance of each one as weights aiming to aggregate the weighted sum of those partial skills in a global skill. For this purpose, we extract individual stats for each player in each match as input training data and match results as target variable to be used by a gradient boosting based tree classification algorithm to allow us to infer the weights through feature importance. A benefit of using gradient boosting is that after the boosted trees are constructed, it is relatively



(a) LoL individual features importance



(b) LoL team features importance



(c) CS:GO individual features importance

Figure 4.5 – Features importance weights

straightforward to retrieve importance scores for each attribute. Generally, importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance. This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other. Importance is calculated for a single decision tree by the amount that each attribute split point improves the performance measure, weighted by the number of observations the node is responsible for. The performance measure may be the purity (Gini index) used to select the split points or another more specific error function. The feature importances are then averaged across all of the the decision trees within the

model. In our case we used the XGBoost [1] Python library to infer the features importance and consequently their potential weights, as it can be seen in Figure 4.5.

So the idea is to let our selected partial skills, along with the original skill, be inferred independently using the modified TrueSkill system, as if we have multiple layers as it can be seen is Figure 4.6, and before a match, when it come to computing winning probabilities, we rely on the global skill which is the sum of those partial skill weighted by their importance:

$$s_{\text{general}} = \sum_{f \in \text{features}} w_f s_f \tag{4.15}$$

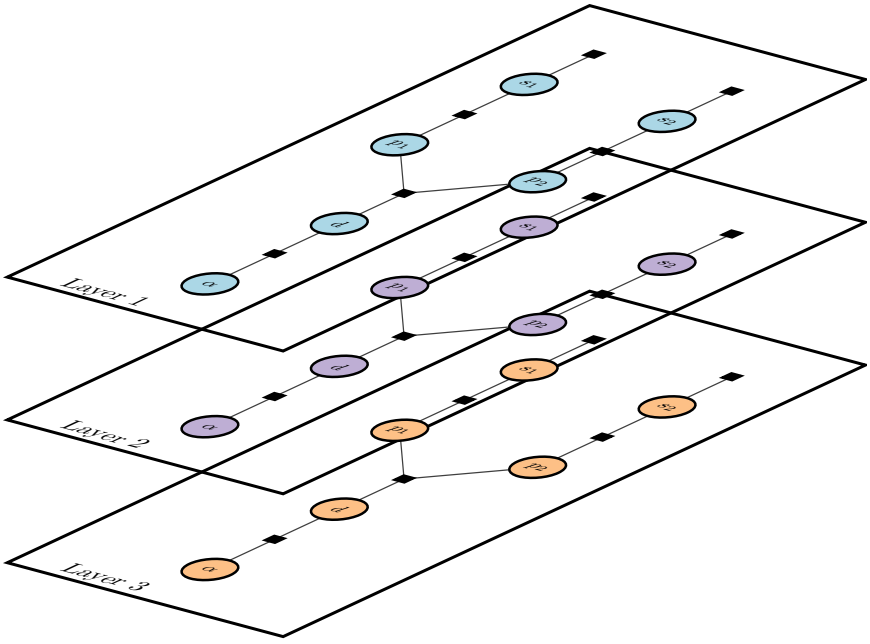
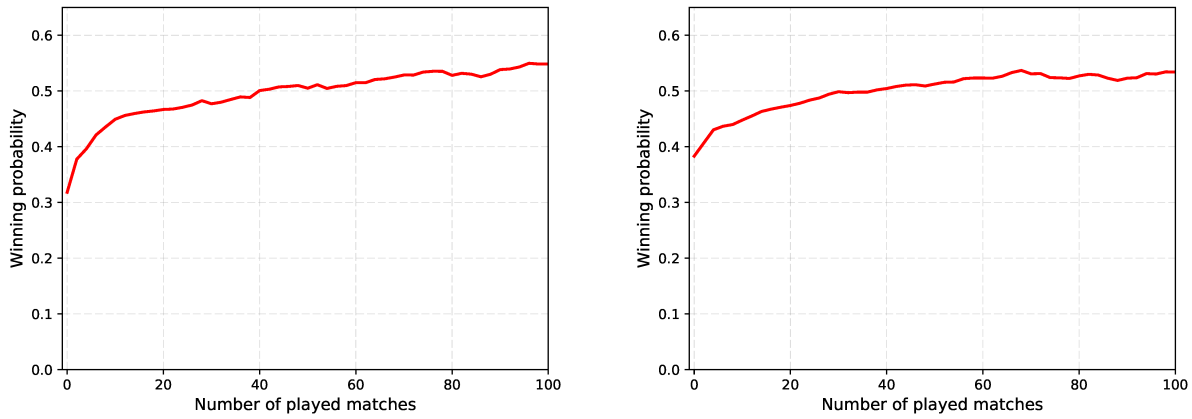


Figure 4.6 – Multi-layer configuration

4.4 Practice Reward

In the original TrueSkill model, it is assumed that players' skills change between matches according to a random walk, where skill increases and decreases are equally likely. This is modeled by adding a variance τ^2 to the skill after each match. However, while investigating our data, we made the following observation: When we classify players according to the number of matches that

1. https://github.com/dmlc/xgboost



(a) CS:GO

(b) League of Legends

Figure 4.7 – Winning probability as a function of the number of previously played matches

they have previously played, we notice that winning probabilities get higher with the number of matches, as shown if Figure 4.7. Players seem to tend to increase in skill the more they practice the game, with the largest increases taking place at the beginning. This could be intuitively explained if we link the skill evolution with practice and experience. To account for this change, we push the skill more for the increase: regardless of match result, the player gains a small (relative to the usual win/ loss update) increase in skill as practice reward or experience bonus, while keeping the additional variance τ^2 to account for the potential both sides variability. That way, the skill would be updated after each match as follows:

$$s_i(n)^{\text{after}} \sim \mathcal{N}\left(s_i(n) + R_p(n), \tau\right), \tag{4.16}$$

where n is the number of previously played matches and $R_p(.)$ is the practice reward function. We choose it to be decreasing in $\frac{1}{n}$ to account for the fact that the largest increases are taking place at the beginning. We leave optimizing this function for each of our studied game to future works and we show the following result obverved in Figure 4.8: The additional practice reward allows the system to do a better job at tracking win probabilities in terms of number of previously played matches (Although it keeps under- estimating it). The improvement here is captured for the case of zero previous matches, i.e., new players, for whom the original TrueSkill had the tendency, in average, to overestimate the winning chances, while the modification ensured a better estimation, in average, for this slice of players population, making it mostly beneficial when matchmaking new comers.

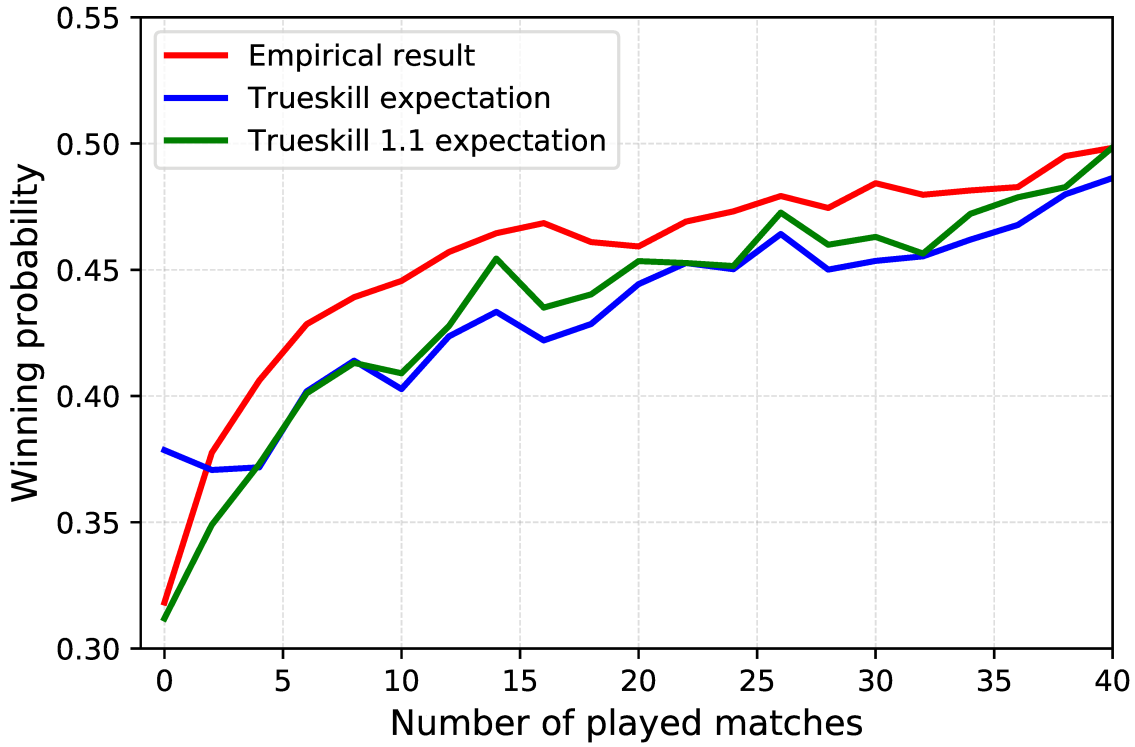


Figure 4.8 – Practice reward effect - CS:GO

4.5 TrueSkill Through Time and Batch Treatment

In a situation where player A beats player B and player B later turns out to be very strong, as evidenced by him beating an already known very strong player C repeatedly, the original TrueSkill would not be able to propagate that information back to correct its estimation. For situation like this and the one presented in section 3.2, TrueSkill Through Time (TTT) has been suggested in [3]. The idea could be summarized in the following: The matches are partitioned into rating years and each player has a skill for each year. Then, the inference algorithm is run chronologically through matches repeatedly until convergence. Next, it is run through the years skills forward and backward this time, until it converges. During propagation, each time a match is revisited, the previous visit effect is removed (by dividing the saved upward message describing that match performance) before adding the new effect, and this to avoid double counting. To make use of this «TrueSkill through the years» idea, we made a simple change: the trick is to reduce the rating period from a year to a single match. This way, we end up with an inference algorithm that converges to the batch skills

discussed in section 3.2. We suggest the following: The new improved TrueSkill version should operates in two modes, an online mode that only propagates skill ratings forward in time, which has the advantage of being fast and immediate, and an off-line batch mode, that takes place before tournament for example, and provides better skills estimates thanks to inference over all time.

Chapter 5

Results and analysis

5.1 Evaluation metrics

Based on the previously inferred players skills, rating systems are able to prediction the outcomes by providing winning probabilities, $\mathbf{p}$, for each team before the beginning of each match. The intuitive direct approach to evaluate the accuracy of the systems is based on win/ loss binary decision: The percentage of correct predictions is the predictive accuracy. A more informative way to evaluate probabilistic forecasting is the definition of reward or loss function as scoring rule: A reward of $R(\mathbf{p}, i)$ is accorded if the i^{th} event occurs, $i \in \{0, 1\}$. A scoring rule is *proper* when it satisfies that the highest expected reward is obtained by reporting the true probability distribution. The use of a proper scoring rule encourages the forecaster to be honest to maximize the expected reward [17]. A scoring rule is strictly proper if it is uniquely optimized by the true probabilities. Let's denote by p the first team winning probability (and thus $(1-p)$ as loss probability or winning probability for team 2).

- **Prediction accuracy**: It is always interesting to see how accurate the models were at predicting match outcomes in terms of binary results. To compare classifications performance of each model, we report in terms of winning predictive accuracy score (percentage of correct predictions) as well as area under the curve (AUC). The latter is the second most popular metric for binary classification, after accuracy. It differs from it by the fact that it considers all possible thresholds (compared to 0.5 threshold for accuracy) which results in different

true positive/false positive rates. AUC score for a random classifier is 0.5 and would be 1 for perfect forecaster. Most often, predictors give something in between. Please check [21] for more about AUC compared to the accuracy.

- **Quadratic score / Brier score**: The quadratic scoring rule is a strictly proper scoring rule defined as $Q(p) = 2p - p^2 - (1-p)^2$, where p is the probability assigned to result that has actually happened. If we denote by y_i the outcome of the match i for team 1, ,.i.e, $y = 1$ or $y = 0$ if they win or lose, respectively, the quadratic scoring rule can be written as:

$$Q(p, y_i) = y_i \ (2p - p^2 - (1-p)^2) + (1 - y_i)(2(1-p) - (1-p)^2 - p^2). \tag{5.1}$$

  An equivalent score that can be obtained by a simple affine transform, is the Brier score defined as:

$$B(p, y_i) = (y_i - p)^2 + (y_i - 1 + p)^2, \tag{5.2}$$

  with the difference that a forecaster should strive to maximize the quadratic score yet minimize the Brier score.

- **Logarithmic scoring rule LSR / Information gain IG**: The logarithmic scoring rule is a strictly proper scoring rule which is commonly used as a scoring criterion in Bayesian Inference and has foundations in information theory. It can be expressed in terms of logarithm as follows: $L(\mathbf{p}) = y \ \ln(p) + (1-y) \ \ln(1-p)$ and since strict property is preserved by linear transformation, any logarithmic base could be used. In [22], author suggested

$$L(\mathbf{p}) = y \ (1 + \log_2(p)) + (1 - y) \ (1 + \log_2(1-p)), \tag{5.3}$$

  which is indeed a proper scoring rule and has the property zero reward for a prediction probability of 0.5.

- **Betting reward metric**: The comparison between the performance of two rating systems could be modeled as two gamblers trying to optimize their bets to maximize their wins in a «fair play» situation where each side starts by fixing the odds for the other who chooses then his bets. The situation is further explained in Annex B where we find that the gain for a side using $\tilde{p}$ against the other using $\hat{p}$, while the unobserved real distribution is y, could be expressed as follows:

$$\bar{G} = \frac{y}{\hat{p}} \ \ \mathbb{1}_{\{\tilde{p} > \hat{p}\}} + \frac{1-y}{1-\hat{p}} \ \ \mathbb{1}_{\{\tilde{p} < \hat{p}\}}. \tag{5.4}$$

- **Features related metrics** : Although accuracy is useful for comparing models, it generally fails tell why, where and for whom errors were made. It may be possible that the model is favoring or underestimating certain kinds of players having particular caracteristiques. The relevant players subsets or subpopupulation may differ between games genre and also according to games' managers intentions. That's why we are considering feature-related metrics to measure the rating systems ability to capture and reduce the gap between the model and reality when it comes to certain aspects and features. This includes players having certain number of matches previously played, kills per minute, average damage received and so on. One may be interested in the new comers or those who scores high kills rates that are being overestimated for example. This idea could be captured by comparing the percentage of matches where a particular kind of players was predicted to win, its expected winning probability, to the actual winning percentage obtaing from the real game results. The expected winning probability of a sub-population of players is calculated by recording the prediction for all games, visiting each player in each game and checking if the player meets the criteria. If this is the case, the prediction for that player's team is added to the count and the final metric is an average of the predictions for all players that met the criteria. This type of metrics was captured and presented for our two games in the following sub-section in the tables 5.3 as well as the Figure 5.1.

5.2 Simulation results and discussion

For both games experiments, half of the data was used as training set to tune the model hyperparameters by grid search over different combinations of settings to maximize the information gain: $\mu_0 = 5, \sigma_0^2 = \beta^2 = 2.5^2, \tau = 10^{-3}$ and $\sigma_\alpha^2 = 1.5^2$. Then the model was asked to predict the outcomes in the rest of the data, the test set. As each match was processed, the model was asked to predict the winning team and give a probability for this event. When making this prediction, only the team composition and prior skills are used, all other in game information is not exploited at that point, but rather, after the match, all that information is used to update the skills using the online updater. This applies to all experiments conducted in the framework of this work. The results and effects of some modifications have already been presented in the previous section, the rest of the results are discussed in the following. The Table 5.1 compares the original version of TrueSkill,

hereinafter TrueSkill, to our proposed enhanced version, hereinafter referred to as TrueSkill 1.1, in terms of four measures. As for the precision, AUC and Brier scores, TrueSkill 1.1 only shows a slight improvement compared to the original for both games. It should be noted that there is already a matchmaking system in use on CS:GO server and LoL matches collected are only from top professional competitions, so the levels are pretty close and the matches are already balanced, making small differences in precision quite significant. The gain is more noticeable when it is the gain of information, since the system relies on additional information. This proves that the system succeeds in transmitting this information which is not totally lost. The benefit of the modifications is also revealed in the Table 5.2. This is the winning bet metric between the two competing systems. TrueSkill 1.1 wins again in both games but this difference is much more pronounced when it comes to LoL.

Metric	Precision	AUC	IG	$IG_{normalized}$	Brier
TrueSkill	0.637	0.689	7855.678	0.334	0.387
TrueSkill 1.1	0.640	0.694	8218.031	0.350	0.374

(a) CS:GO

Metric	Precision	AUC	IG	$IG_{normalized}$	Brier
TrueSkill	0.638	0.685	2060.731	0.338	0.228
TrueSkill 1.1	0.648	0.702	2184.558	0.360	0.225

(b) LoL

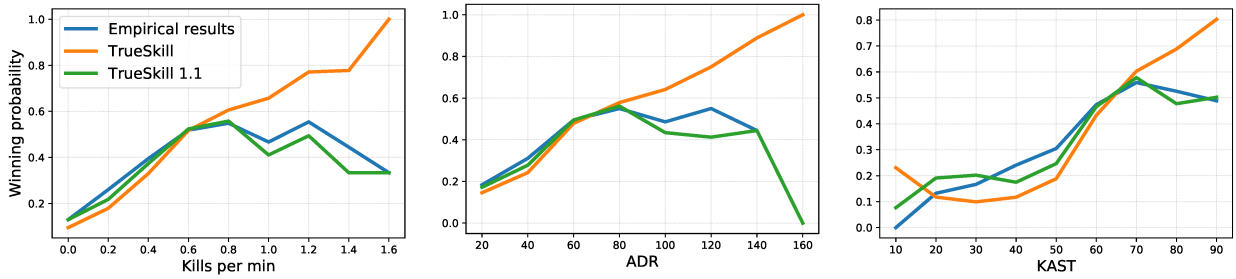
Table 5.1 – Metrics comparison for both systems

	LoL	CS:GO
TrueSkill	6995.812	35362.502
TrueSkill 1.1	11377.262	36702.719

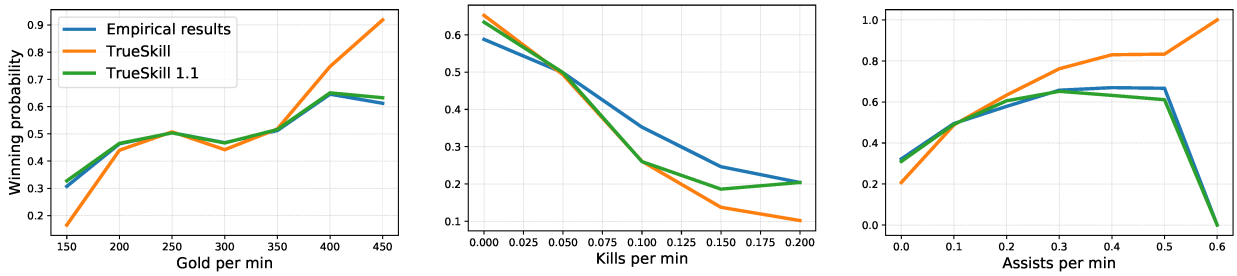
Table 5.2 – Betting gains comparison for both systems

We finally come to the metrics that highlight the most the benefit of the modifications applied to TrueSkill: those related to features.

Figure 5.1 and Tables 5.3 illustrate the effect of these changes. Players in the test set are allocated based on their average rate of a feature in their previous games and the rates or probabilities of actual victory as well as those expected by TrueSkill and TrueSkill 1.1 for each subset of players are evaluated. For CS:GO, this is the number of kills per minute, *Kills*, the average damage received, *ADR*, and *KAST* which models the percentage of turns in which the player was killed, assisted, survived or traded. LoL statistics we represent here are the numbers of *Kills*, *Assists* and *Gold*



(a) CS:GO



(b) LoL

Figure 5.1 – Winning probability vs features

collected per minute.

The Tables 5.3 also report the size of each sub-population and the mean absolute error (MAE) weighted by sub-population size, which quantifies the error between the predicted and actual probabilities.

The first observation drawn from the curves in the Figure 5.1 is that TrueSkill 1.1 does a much better job in terms of tracking the real winning rates. In fact, we can clearly see that the green curve tends to follow well the changes made by the empirical results curve, whereas TrueSkill's orange curve often has a monotonous behavior that causes a visible difference at the extremities. The observation that the probabilities provided by TrueSkill 1.1 have a better tracking ability of sub-population reality is confirmed by the mean absolute error (MAE) weighted by sub-population sizes, whose values are at least half that of TrueSkill. It should also be noted that the extremities correspond to populations whose size is quite small compared to those close to the middle where the majority of players are concentrated and where the TrueSkill curve follows well that describing the real rates. On the other hand, this underlines a property that distinguishes TrueSkill 1.1 apart from its original version, namely the ability to properly estimate the skills of «minorities» such as those who perform a large number of *assists* or those who tend to collect very little *gold*. Moreover, we saw in the section 4.4 that the inclusion of the practice reward allows the model to better estimate,

		Win probability					Win probability		
Kills \min	Counts	Reality	TS	TS 1.1	**Kills** \min	Counts	Reality	TS	TS 1.1
0.0 - 0.3	466	0.180	0.146	0.159	0.00 - 0.05	9943	0.588	0.652	0.632
0.3 - 0.6	24167	0.390	0.323	0.368	0.05 - 0.10	45432	0.498	0.493	0.499
0.6 - 0.9	200608	0.521	0.523	0.527	0.10 - 0.15	3999	0.352	0.261	0.262
0.9 - 1.2	2350	0.518	0.644	0.477	0.15 - 0.20	349	0.246	0.138	0.189
1.2 - 1.5	98	0.520	0.745	0.500	0.20 - 0.25	49	0.204	0.102	0.204
1.5 - 1.8	6	0.500	1.000	0.167	0.25 - 0.30	4	0.5	0.25	0.5
	wMAE		0.013	0.007		wMAE		0.021	0.015

		Win probability					Win probability		
ADR	Counts	Reality	TS	TS 1.1	**Gold** \min	Counts	Reality	TS	TS 1.1
20 - 40	343	0.184	0.146	0.172	150 - 200	345	0.307	0.165	0.319
40 - 60	5376	0.311	0.242	0.278	200 - 250	7235	0.463	0.440	0.467
60 - 80	154755	0.495	0.479	0.495	250 - 300	7043	0.504	0.507	0.504
80 - 100	66151	0.550	0.579	0.562	300 - 350	16796	0.468	0.442	0.467
100 - 120	965	0.486	0.641	0.434	350 - 400	23870	0.512	0.518	0.515
120 - 140	80	0.550	0.750	0.412	400 - 450	4439	0.646	0.747	0.651
140 - 160	9	0.444	0.889	0.444	450 - 500	49	0.612	0.918	0.653
	wMAE		0.021	0.004		wMAE		0.028	0.016

		Win probability					Win probability		
KAST	Counts	Reality	TS	TS 1.1	**Assists**\min	Counts	Reality	TS	TS 1.1
30 - 40	282	0.167	0.099	0.202	0.0 - 0.1	4995	0.322	0.207	0.311
40 - 50	1092	0.241	0.117	0.175	0.1 - 0.2	41043	0.495	0.488	0.491
50 - 60	6229	0.305	0.188	0.247	0.2 - 0.3	12349	0.579	0.633	0.606
60 - 70	115995	0.474	0.433	0.466	0.3 - 0.4	1264	0.657	0.762	0.653
70 - 80	101783	0.559	0.603	0.578	0.4 - 0.5	106	0.670	0.830	0.651
80 - 90	1979	0.526	0.688	0.478	0.5 - 0.6	18	0.667	0.833	0.611
90 - 100	223	0.489	0.803	0.502	0.6 - 0.7	1	0.000	1.000	0.000
	wMAE		0.046	0.014		wMAE		0.021	0.014

(a) CS:GO (b) LoL

Table 5.3 – Real and expected win probabilities for the different sub-populations relative to each of the studied feature

in average, the chances of victory of those who play for the first time (contrary to TS which tends to overestimate their chances). All this can be useful especially when game managers want to bias their matchmaking policy to focus on a particular feature.

To observe the contribution of the proposed idea in the sub-section 4.5, we intend to proceed as follows: We assume that the first 80% of the data makes up the history available before the start of a tournament and it presents the training set on which the batch update will be applied in

offline mode before resorting to the usual online inference for tournament game updates, i.e., the remaining 20%, or the test set. We then follow the evolution of certain metrics after the application of offline batch processing. This effect on the training set itself as well as the test set with and without the offline update on the historical data is summarized in the Tables 5.4:

Metric	Precision	AUC	$GI_{normalized}$	Brier
(*) Offline update of the training set	0.737	0.807	0.491	0.401
Online update of the test set without (*)	0.640	0.694	0.349	0.374
Online update of the test set with (*)	0.651	0.707	0.358	0.382

(a) CS:GO

Metric	Precision	AUC	$GI_{normalized}$	Brier
(*) Offline update of the training set	0.712	0.798	0.468	0.188
Online update of the test set without (*)	0.685	0.702	0.360	0.225
Online update of the test set with (*)	0.692	0.714	0.368	0.224

(b) LoL

Table 5.4 – Effect of offline batch update

We see a significant gain on all historical data thanks to the batch processing thanks to the propagation through time. This gain is less pronounced on the test set, and this can be explained mainly by the fact that the tournament matches involve new players who were not part of the overall training. However the gain remains quite significant, as players are already subjects to matchmaking as explained before, which confirms the additional value of using TTT to apply offline batch updates in inter tournament periods.

Chapter 6

Conclusions and Future Work

In this thesis, we proposed extending an existing rating system, namely TrueSkill, to account for and incorporate additional information that is readily available in nowadays online multiplayer games. This is available through the multitude of the provided in-game statistics such as player past experience, his number of kills, gold collected, average damage received and many others.

We considered then some system modification to incorporate these features and capture those new aspects aiming to optimize multiple evaluation metrics we suggested. First, we started by defining the rating system and the framework of evaluation including the system model and the collected data. We introduced then the mechanism of Bayesian inference using Gaussian message passing in TrueSkill before studying and analyzing convergence and parameters evolution as part of synthetic study cases: We showed that the online nature of TrueSkill updates and the eventual possible loops that may occur in certain situations may lead to non-accurate rankings and overconfident estimation of the variance compared to the perfect batch treatment case.

After that, we followed by the discussion of the ameliorations and extensions to the original TrueSkill. We started by introducing how to incorporate the collective features scores into the skill update before discussing how to proceed with the individual statistics. We then suggested a model change that allowed to account for each player magnitude of contribution in the game result in correlation with a specific feature. This allowed the system to be more fair compared to the original TrueSkill or the update that is based on the team's score. Following this, we introduced the partial

skills correlated to the different features and how we proceeded to aggregate them into a single skill using their sum weighted by each feature's importances.

Finally, we ushered in the different metrics we relied on to evaluate the gains obtained through the suggested extensions. That included the precision, the AUC score, the information gain, the brier score, a betting reward metric as well as features related metrics. Given that data consisted of already matchmaked duels, some metrics showed only some slight improvements. Gain was more pronounce in terms of information gain and betting reward. In addition to that, the integration of practice reward allowed to fix the overestimation of new players winning chances that suffered the original TrueSkill. The use of TTT to obtain the offline batch updates in inter tournaments periods proves also to be able to improve the evaluation metrics not only for the historical data but also for the new tournament matches. Further, the use of feature related metric to track to the rating system comportment for specific sub populations proved its ability to better track the real associated winning chances and the ability to properly estimate skills of some minorities. All this could be useful especially when game managers want to bias their matchmaking policy to focus on a particular feature.

Future follow up to this work could include the following:

- Optimizing the reward function presented in section 4.4 which we have not had the opportunity to carry out.
- Studying other aggregation method of the partial skills including the inter-correlation between features: For simplicity, we have assumed that they are independent.
- Monitoring of the correlations between players, which has already been proved to increase the accuracy [23] , and use it to further extend the system.
- For MOBA games such as LoL, investigating modeling the team performances as a sum weighted according to the role in the team (i.e, top, jungle, MID, ADC or support). Also, since the player can choose different champions, one could think of multiple skills champion-correlated.

It is worth mentioning that as this report was being written, Microsoft has published their new version of their rating system calling it TrueSkill2 [24]. The authors, too, had the idea of augmenting the original system with information gathered from the matches. They opted to add an (parameterized) offsets to the competences to account for certain characteristics such as the size of the «*squad*» of friends, the accumulated *experience* and the tendency to leave during the match (*quit penalty*). For individual statistics, they introduced the notion of «*counts*» which is random

variable, defined from the performance of the player as well as those of his teammates and those of the opposing team, and which serves to introduce the effect of thoses individual statistics on the updating of skills. The authors also discussed the correlation between different game modes and how to take advantage of it to infer from and in between these modes. An important innovation was also presented in this work, namely that of the estimation of all parameters that define the systems and the suggested modifications, directly and simultaneously with the inference of skills.

Références

[1] E. Zermelo, “Die berechnung der Turnier-Ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung” [The calculation of tournament results as a maximum likelihood problem], *Mathematische Zeitschrift*, 29, 436-460, 1929.

[2] R. Herbrich, T. Minka, T. Graepel, . “TrueSkill™: a Bayesian skill rating system”, *Advances in neural information processing systems* (pp. 569-576), 2007

[3] P. Dangauthier, R. Herbrich, T. Minka, T. Graepel, “TrueSkill through time: Revisiting the history of chess”, *Advances in Neural Information Processing Systems*, Vol. 20 (pp. 337-344), 2007.

[4] S. Guo, S. Sanner, T. Graepel, and W. Buntine, 2012, September. “Score-based bayesian skill learning”, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 106-121). Springer, Berlin, Heidelberg.

[5] G. Yannakakis, P. Spronck, D. Loiacono, E. André, “Player Modeling”, *Artificial and computational intelligence in games*, pp. 45-59, 2013.

[6] S. C. J. Bakkes, P. H. M. Spronck, G. van Lankveld, “Player behavioural modelling for video games”, *Entertainment Computing*, pp. 71-79, 2012.

[7] M. Stanescu, “Rating systems with multiple factors”, M.S. thesis, School Inf., Univ. Edinburgh, Edinburgh, U.K., 2011.

[8] H. Rahman, S. Thirumuruganathan, S. B. Roy, S. Amer-Yahia, G. Das, “Worker skill estimation in team-based tasks”, *Proceedings of the VLDB Endowment*, 8(11), 1142-1153, 2015.

[9] S. B. Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, G. Das, “Task assignment optimization in knowledge-intensive crowdsourcing”, *The VLDB Journal*, 24(4), 467-491, 2015.

[10] N. Pobiedina, J. Neidhardt, M. C. Calatrava Moreno, L. Grad-Gyenge, H. Werthner, “On Successful Team Formation: Statistical Analysis of a Multiplayer OnlineGame”, *IEEE Conference on business informatics* (pp. 55-62). IEEE, 2013.

[11] N. Pobiedina, J. Neidhardt, M. C. Calatrava Moreno, L. Grad-Gyenge, H. Werthner, “Ranking factors of team success”, *Conference on world wide web companion*, pp. 1185-1194, 2013.

[12] Z.Chen, Y. Sun, M. S. El-Nasr, T-H. D. Nguyen, “Player Skill Decomposition in Multiplayer Online Battle Arenas”, URL: *https://arxiv.org/abs/1702.06253*

[13] A.V. Tulupyev, S.I. Nikolenko, A.V. Sirotkin: “Bayesian networks: a logical probabilistic approach", St. Petersburg, Nauka (2006)

[14] T. Minka, “Expectation propagation for approximate Bayesian inference”, *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 362-369. Morgan Kaufmann Publishers Inc., 2001.

[15] R. Herbrich. “Ranking and Matchmaking TrueSkill Revealed", 2007. URL http://www.microsoft.com/downloads/en/details.aspx?id=7367

[16] Y.Weiss, and W. T. Freeman, “Correctness of belief propagation in Gaussian graphical models of arbitrary topology", *Advances in Neural Information Processing sSystems*, pp. 673-679. 2000.

[17] T. Gneiting, A. E. Raftery, “Strictly proper scoring rules, prediction, and estimation”, *Journal of the American Statistical Association 102*, no. 477 (2007): 359-378.

[18] J. Pannekeet, “Global Esports Economy Will Reach $v905.6 Million in 2018 as Brand Investment Grows by 48%", 2007. URL `https://newzoo.com/insights/articles/newzoo-global-esports-economy-will-reach-905-6-million-2018-brand-investment-grows-48/`

[19] Powered by Steam: Steamcharts. An ongoing analysis of steam’s concurrent players (2018). http://steamcharts.com/. Accessed May 2018

[20] M. Kaytoue et al., “ Watch me playing, I am a professional: a first study on video game live streaming”, *Proceedings of the 21st International Conference on WWW*, NY, USA, pp. 1181-1188. ACM (2012)

[21] C.X. Ling, J. Huang, H. Zhang, “AUC: a statistically consistent and more discriminating measure than accuracy”, *Proc. Int. Joint Conf. Artif. Intel.* 2003, 519–526 (2003)

[22] D. Dowe, “Probabilistic and Gaussian Football Tipping”, *Vinculum 48*, no. 2 p.10, 2011.

[23] A. Birlutiu, T. Heskes. “Expectation propagation for rating players in sports competitions”, *Knowledge Discovery in Databases: PKDD 2007*, pp. 374-381, 2007.

[24] T. Minka, R. Cleven, Y. Zaykov, “TrueSkill 2: An improved Bayesian skill rating system”. URL: `https://www.microsoft.com/en-us/research/publication/trueskill-2-improved-bayesian-skill-rating-system/`.

Annexe A

Métrique de récompense de pari

Un jeu de pari est basé sur un événement binaire $A \sim \mathcal{B}er(p)$ où $p = \Pr\{A = 0\}$. L'hôte et le joueur estiment p par $\hat{p}$ et $\tilde{p}$, respectivement.
L'hôte fixe les gains $g_A \in \{g_0, g_1\}$: $g_A = g_0$ si $A = 0$ et $g_A = g_1$ si $A = 1$.
Le joueur fait des paris nominaux $\beta_A \in \{\beta, 1-\beta\}$: $\beta_A = \beta$ si $A = 0$ et $\beta_A = 1-\beta$ if $A = 1$. Il gagne alors βg_0 si $A = 0$ et $(1-\beta)g_1$ si $A = 1$. Le gain espéré $\bar{G}$ est alors:

$$\bar{G} = \mathbb{E}_A[g_A\beta_A] - 1 = pg_0\beta + (1-p)(1-\beta)g_1 - 1 \tag{A.1}$$

L'hôte veut choisir des gains d'une manière qu'il s'attend à faire des gains lui-même ($\bar{G} < 0$) ou au moins assure un "fair-play" où aucune des parties ne gagne ou ne perd ($\bar{G} = 0$).

$$\begin{aligned} \bar{G} \leq 0 \Leftrightarrow\ & p\beta g_0 + (1-p)(1-\beta)g_1 \leq 1 \\ \Leftrightarrow\ & \begin{cases} \beta\left(pg_0 - (1-p)g_1\right) \leq 1 - (1-p)g_1 \\ (1-\beta)\left((1-p)g_1 - pg_0\right) \leq 1 - pg_0 \end{cases} \\ \overset{\beta\leq 1}{\Longleftarrow}\ & \begin{cases} pg_0 \leq 1 \Leftrightarrow g_0 \leq \frac{1}{p} \\ (1-p)g_1 \leq 1 \Leftrightarrow g_1 \leq \frac{1}{1-p} \end{cases} \end{aligned} \tag{A.2}$$

Nous considérons le cas où ni l'hôte ni le joueur ne détient la connaissance de la valeur réelle de p. L'hôte fixe les gains pour assurer le fair-play sur la base de son estimation $\hat{p}$, $g_0 = \frac{1}{p}$; $g_1 = \frac{1}{1-p}$ et l'annonce (le joueur connaît $\hat{p}$) :

$$\bar{G} \leq 0 \Leftrightarrow \beta\ \frac{p}{\hat{p}} + (1-\beta)\ \frac{1-p}{1-\hat{p}} \leq 1 \tag{A.3}$$

$\tilde{p}$ est l'estimation du joueur de p.
Si $\tilde{p} > \hat{p}$ (et ainsi $1-\tilde{p} \leq 1-\hat{p}$), il mise tout sur $A = 0$, i.e., $\beta = 1$. Si $\tilde{p} < \hat{p}$, le joueur choisit $\beta = 0$.

$$\Rightarrow \bar{G} = \frac{p}{\hat{p}}\ \ \mathbb{1}_{\{\tilde{p}>\hat{p}\}} + \frac{1-p}{1-\hat{p}}\ \ \mathbb{1}_{\{\tilde{p}<\hat{p}\}}. \tag{A.4}$$

Dans notre problème d'estimation de la notation, nous voulions comparer quel système de notation donne les meilleurs résultats en fonction des probabilités de victoire qu'ils fournissent. Une façon de voir cela est comme un problème de pari où chaque fois que l'un d'entre eux essaie de fixer les

cotes du fair-play, l'autre essaie de le battre. Celui qui réalise le gain le plus élevé serait considéré comme le meilleur système selon cette métrique.

Annexe B

Betting Reward Metric

A betting game is based on a binary event $A \sim \mathcal{B}er(p)$ where $p = \text{Prob}\{A = 0\}$. The host and the gambler estimate p by $\hat{p}$ and $\tilde{p}$, respectively.
The host fixes the gains $g_A \in \{g_0, g_1\}$: $g_A = g_0$ if $A = 0$ and $g_A = g_1$ if $A = 1$.
The gambler makes nominal bets $\beta_A \in \{\beta, 1-\beta\}$: $\beta_A = \beta$ if $A = 0$ and $\beta_A = 1-\beta$ if $A = 1$. He gains then βg_0 if $A = 0$ and $(1-\beta)g_1$ if $A = 1$. The expected gain $\bar{G}$ is then:

$$\bar{G} = \mathbb{E}_A[g_A\beta_A] - 1 = pg_0\beta + (1-p)(1-\beta)g_1 - 1 \tag{B.1}$$

The Host wants to choose gains in a way he expects to make some gains himself ($\bar{G} < 0$) or at least ensures a "fair play" where neither parts gains or looses ($\bar{G} = 0$).

$$\begin{aligned} \bar{G} \leq 0 \Leftrightarrow\ & p\beta g_0 + (1-p)(1-\beta)g_1 \leq 1 \\ \Leftrightarrow\ & \begin{cases} \beta\left(pg_0 - (1-p)g_1\right) \leq 1 - (1-p)g_1 \\ (1-\beta)\left((1-p)g_1 - pg_0\right) \leq 1 - pg_0 \end{cases} \\ \overset{\beta\leq 1}{\Longleftrightarrow}\ & \begin{cases} pg_0 \leq 1 \Leftrightarrow g_0 \leq \frac{1}{p} \\ (1-p)g_1 \leq 1 \Leftrightarrow g_1 \leq \frac{1}{1-p} \end{cases} \end{aligned} \tag{B.2}$$

We consider the case where neither the host or the gambler detains the knowledge of the actual p. The host fixes the gains to ensure fair play based on his estimation $\hat{p}$, $g_0 = \frac{1}{p}; g_1 = \frac{1}{1-p}$ and announces it (the gambler knows $\hat{p}$):

$$\bar{G} \leq 0 \Leftrightarrow \beta\ \frac{p}{\hat{p}} + (1-\beta)\ \frac{1-p}{1-\hat{p}} \leq 1 \tag{B.3}$$

$\tilde{p}$ is gambler's estimation of p.
If $\tilde{p} > \hat{p}$ (and then $1-\tilde{p} \leq 1-\hat{p}$), he bets all on $A = 0$, i.e., $\beta = 1$. If $\tilde{p} < \hat{p}$, the gambler chooses $\beta = 0$.

$$\Rightarrow \bar{G} = \frac{p}{\hat{p}}\ \ \mathbb{1}_{\{\tilde{p}>\hat{p}\}} + \frac{1-p}{1-\hat{p}}\ \ \mathbb{1}_{\{\tilde{p}<\hat{p}\}}. \tag{B.4}$$

In our rating estimation problem, we wanted to compare to which rating system performs better based on the winning probabilities they provide. One way to look at it is as a betting problem where each time one of them tries to fix the fair game odds and the other tries to beat him. The one realizing the higher gain would be considered as the better system according to this metric.