MINESnancy

30/01/2022

Projet départemental IM

Échecomètre : Estimation du niveau de jeu d'un joueur d'échecs

Rapport final de projet

Ingénierie Mathématiques

Imad AYAD

Encadrant :

Rémi PEYRE

Sommaire :

I- Introduction

II- Quelques définitions et vocabulaire

III- Constitution d'une base de données

IV- Modèle mathématique

V- Amélioration du modèle

VI- Résultats finaux

VII- Conclusion

VIII- Bibliographie

I- Introduction

Le **niveau** est une notion que nous abordons généralement intuitivement, voire instinctivement au quotidien. Par exemple, une formulation telle que « *le niveau au tennis d'un joueur professionnel est plus élevé que celui de n'importe quel amateur* » est difficilement contestable. Pour autant, la formalisation rigoureuse de ce concept est loin d'être immédiate et est sujette à diverses interprétations ; il serait alors bien ambitieux, et peut-être même hors propos, de prétendre définir **le** concept de niveau dans ce rapport.

Néanmoins, dans le fond, on sent que l'on parvient à se saisir de ce que serait censée représenter une telle quantité si elle était parfaitement définie : une mesure objective des performances dudit joueur dans une discipline.

Ainsi, afin d'y parvenir, la **Fédération internationale des échecs** (FIDE) choisit de se servir d'un **classement Elo**, qui attribue à chaque joueur un nombre de points appelé Elo. Le classement d'un joueur dépend alors de ses performances passées. Plus précisément, lorsque ce dernier vainc un adversaire supposé meilleur que lui, il se voit récompenser par une hausse de points de son classement Elo. Au contraire, si ce dernier cède face à un adversaire normalement moins fort, il est pénalisé et voit son classement Elo baisser. Le classement Elo vers lequel converge un joueur après un nombre important de parties est la quantité que nous décidons **arbitrairement** d'appeler **le niveau du joueur.**

Par conséquent, l'objectif de ce projet est d'estimer le niveau du joueur (au sens où on l'a défini ci-dessus) à partir de l'analyse d'une ou plusieurs parties jouées par ce dernier.

Notre démarche est, à notre connaissance, **inédite** car elle consiste à mettre en œuvre un moteur d'échecs dont les capacités de calculs sont volontairement réduites, dans l'espoir que ce dernier puisse émuler fidèlement le raisonnement humain du joueur d'échecs. Le pari est audacieux, car rien ne garantit que de tels algorithmes puissent approcher suffisamment bien la réflexion humaine. Par ailleurs, le **cœur de ce projet** est de parvenir à estimer le niveau d'un joueur d'échecs, en évaluant seulement les **coups joués dans l'absolu**, là où le classement Elo nécessite de faire confronter les joueurs pour les évaluer.

Nous proposons de mettre en œuvre des méthodes probabilistes pour parvenir à nos fins. Ainsi, tout l'enjeu de notre démarche réside dans la construction d'un modèle de probabilité sur les coups jouables dans une partie en fonction du niveau ; autrement dit, **il s'agit de pouvoir émuler fidèlement le raisonnement d'un joueur de niveau variable.**

II- Quelques définitions et vocabulaire

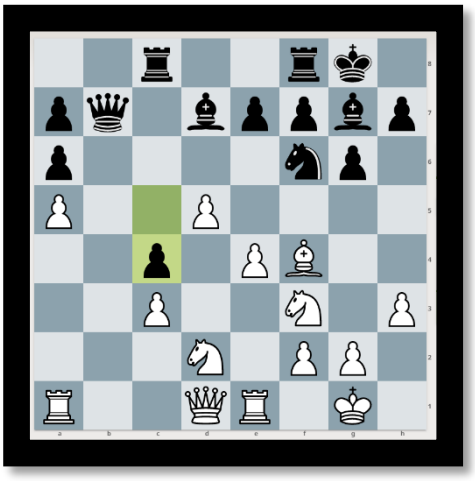
Dans cette partie, on se propose d'introduire quelques définitions qui éclaireront la compréhension du lecteur.

Quelques définitions et vocabulaires

Position

Aux échecs, la partie commence systématiquement avec un positionnement initial des pièces bien défini, conformément aux règles du jeu. Ensuite, chacun des coups joués va venir modifier la position des pièces précédente pour atteindre à terme une position victorieuse en faveur de l'un des deux joueurs, ou nulle.

En règle générale, la position des pièces à elle seule suffit à décrire complètement la situation de la partie à un certain instant. Par conséquent, si vous souhaitez discuter avec votre collègue amateur d'échecs d'une partie que vous avez récemment joué, il suffirait de lui partager une image comme celle ci-dessous pour qu'il puisse immédiatement se saisir des enjeux de la partie !



Exemple de positionnement des pièces d'échecs

Néanmoins, en toute rigueur, il manquerait les informations suivantes :

- Roques possibles (par exemple, il n'est pas possible pour le Roi de roquer avec une Tour qui a changé de position au cours des coups précédents, même si elle revient à sa position initiale ensuite)
- Case d'une prise en passant possible, nombre de coups depuis le début de la partie, et depuis la dernière prise ou mouvement d'un pion[1]

On adjoint alors ces informations à la donnée de la position des pièces, pour former un objet qu'on appelle **position.** La position d'une partie est un objet qui concentre toutes les informations nécessaires qui permettent de procéder à son **évaluation**.

Évaluation

Soit $\boldsymbol{X}$ une **position,** au sens où on l'a défini précédemment. On appelle $\boldsymbol{Evaluation\ X}$ la donnée d'un réel quantifiant qui des blancs ou des noirs sont gagnants (et à quel point) dans la position $\boldsymbol{X}$.

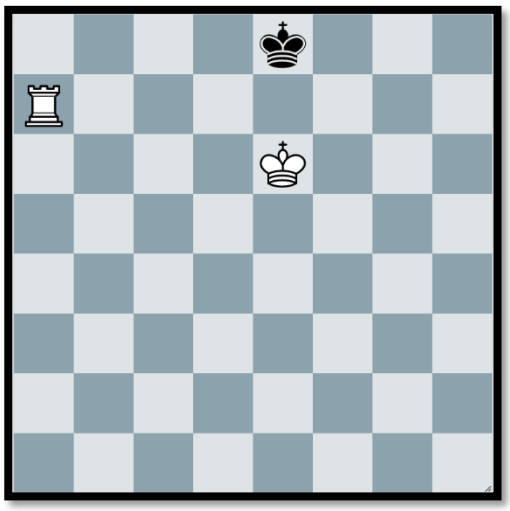
On prendra toujours la convention du référentiel blanc pour le signe. Ainsi, si $\boldsymbol{Evaluation\ X}$ renvoie $+77$ par exemple, alors les blancs dominent. Au contraire, si $\boldsymbol{Evaluation\ X}$ renvoie -77, ce sont les noirs qui sont en train de dominer.

De plus, $\boldsymbol{Evaluation\ X}$ peut également prendre la forme d'une chaine de texte sous la forme « $\#x$ » où x est un entier relatif. Si $\boldsymbol{x}$ est positif, alors l'évaluation prédit un mat **imparable** dans **au plus** x coups en faveur des blancs.

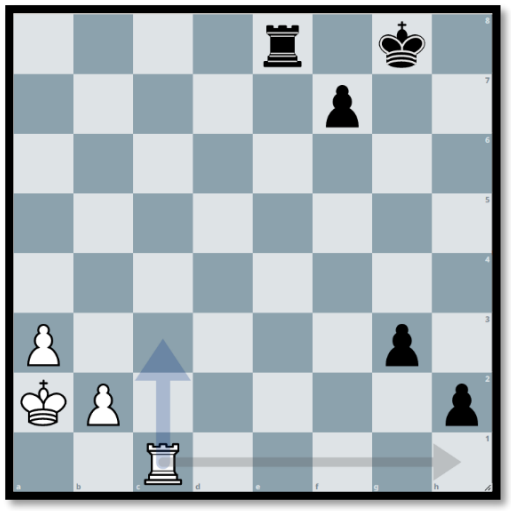
Par exemple, supposons $Evaluation\ X = \# + 4$. Alors, si les blancs jouent parfaitement, ils gagneront à coup sûr dans au plus 4 coups.

[1] Il ne nuit pas à la compréhension du sujet de ne pas saisir tous les termes employés. Il s'agit simplement de retenir que dans la majorité des cas, le positionnement des pièces suffit à décrire tous les enjeux, néanmoins il se peut que *parfois* il faille préciser davantage.

Exemples :



Soit **X** une telle position (trait aux blancs). $Evaluation(X) = \# + 1$ car les blancs peuvent gagner au prochain coup.



Soit **X** une telle position (trait aux blancs). $Evaluation(X) = -5300$ selon ***Stockfish***.
➔ Les blancs sont très mal en point, à cause des pions F et G sur le point d'être promus.

Moteur d'échecs Stockfish

Une partie importante des méthodes que nous mettons en œuvre repose sur le bon fonctionnement du moteur d'échecs **Stockfish**. Ce dernier a été publié sous licence libre pour la première fois en 2008, et est depuis maintenu très régulièrement.

Stockfish est un outil qui peut essentiellement se résumer à la donnée d'une fonction **d'évaluation** s suivante :

$$s : Position\ X \rightarrow \mathbb{R} \cup \# \pm \mathbb{N}$$

Ainsi, **Stockfish** prend en entrée une position X et renvoie un **réel,** ou bien une **chaine de caractères sous la forme** "$\# \pm \boldsymbol{k}$" avec $k \in \mathbb{N}$.

Stockfish effectue ses calculs à travers un arbre parcourant grossièrement toutes les possibilités après qu'un coup ait été joué, **jusqu'à une certaine profondeur** $\boldsymbol{P}$. En effet, il n'est pas envisageable d'explorer indéfiniment toutes les possibilités, tant un tel arbre serait large. Ainsi, **Stockfish** ne renvoie une telle chaine de caractère que quand la profondeur de calcul qui lui est allouée lui permet de prévoir un mat dans au plus $\boldsymbol{k}$ coups.

À l'unanimité, Stockfish fait partie des meilleurs moteurs d'échecs qui aient été développés à ce jour. En particulier, ce dernier a l'avantage d'être fourni sous licence libre, c'est-à-dire que son usage n'est pas restreint et que le code dont il est constitué est public.

À la lumière de ces éléments, Stockfish est l'outil que l'on choisit pour mener à bien tout le projet. En particulier, toutes les évaluations des positions échiquéennes seront calculées à partir de ce dernier.

Application softmax

Soit $T > 0$ et $n \in \mathbb{N}^*$.

Dans ce paragraphe, on introduit l'application $softmax$ comme suit :

$$\boldsymbol{softmax_T} : \mathbb{R}^n \to \mathbb{R}^n$$

$$(x_1, \dots, x_n) \to (p_1, \dots, p_n), \text{ où } p_i = \frac{\exp\frac{x_i}{T}}{\sum \exp\frac{x_j}{T}}$$

Le lecteur l'aura sans doute remarqué : cette application transforme un vecteur en un vecteur de même taille, dont les coordonnées sont positives et dont la somme est égale à 1. Autrement dit, on peut interpréter ces dernières comme des probabilités : ce point de vue nous sera particulièrement utile pour établir notre modèle probabiliste dans la suite.

Par ailleurs, on peut ajuster le « déterminisme » de la sortie du $softmax_T$ À l'aide du paramètre de température noté T. En particulier, lorsque ce dernier est grand, le vecteur de probabilité se rapproche d'une distribution uniforme, donc plus « aléatoire ». Au contraire, lorsque T devient petit, le vecteur probabilité devient très déterministe autour de quelques valeurs.

Exemple :

Considérons $\boldsymbol{x = (-4, 3, 5)}$. On a alors :

- $softmax_{T=10}(x) \approx (0.18; 0{,}37; 0{,}45)$
- $softmax_{T=0.01}(x) \approx (0.00; 0.00; 1{,}00)$ → Quasi déterministe
- $softmax_{T=100}(x) \approx (0{,}32; 0{,}34; 0{,}35)$ → Quasi uniforme

Fonction couleur

On définit la fonction $couleur : Position\ X \rightarrow \{-1,\ 1\}$ qui renvoie 1 si c'est aux blancs de jouer dans la position X, -1 sinon.

Cette fonction, à l'apparence modeste, n'a cessé d'être sollicitée durant notre étude tant l'information qu'elle recèle est importante : elle vient préciser le point de vue (noir ou blanc) depuis lequel nous nous plaçons lorsque nous considérons une position.

Métrique d'erreur partiellement supervisée

Nous verrons dans la suite que nous ne connaitrons jamais les vraies valeurs de θ. En revanche, nous aurons accès aux classements Elo des joueurs sur lesquels nous inférerons les valeurs de θ. **Ainsi, comment se prononcer sur la qualité des classifications de notre modèle, lorsque nous ne connaissons pas la valeur θ attendue pour chaque joueur ?**

Nous ne disposons certes pas de relation directe $Elo \leftrightarrow \theta$, mais en toute logique, nous devrions avoir :

$$Elo(Joueur\ 1) > Elo(Joueur\ 2) \implies \theta_{Joueur\ 1} > \theta_{Joueur\ 2}$$

Ainsi, quitte à ne pas pouvoir mesurer directement la différence $\boldsymbol{\theta_{obtenus} - \theta_{souhaités}}$, on peut mesurer le bon « ordonnement[2] » de la liste de valeurs $(\boldsymbol{\theta_{Joueur\ 1}, \dots, \theta_{Joueur\ n}})$ en définissant une certaine métrique.

Ainsi, supposons que l'on dispose du classement Elo de n joueurs. On ordonne ces derniers dans l'ordre croissant, donc :

$$Elo_1 < Elo_2 < \cdots < Elo_n$$

On note alors θ_i le maximum de vraisemblance renvoyé pour le joueur i.

Dans la pratique, pour mesurer l'erreur d'ordonnement d'une liste, on compte le nombre **d'inversions** et l'on normalise en divisant par $\frac{n \times (n-1)}{2}$ de la manière suivante :

[2] Ce mot n'existe pas mais exprime bien l'idée.

$$Erreur = \frac{2}{n \times (n-1)} \times \sum_{1 \leq i < j \leq n} 1_{\{\theta_i \geq \theta_j\}}$$

Or, nous tenons ici à ce qu'il soit plus dommageable d'ordonner incorrectement deux joueurs dont les classements Elo sont éloignés (où l'on devrait donc en théorie avoir moins de chance de se tromper !). Par conséquent, on **pondère** l'indicatrice pour finalement obtenir :

$$Erreur = \frac{2}{n \times (n-1)} \times \sum_{1 \leq i < j \leq n} 1_{\{\theta_i \geq \theta_j\}} \times (Elo_j - Elo_i)$$

Le calcul de $Erreur$ ne nécessitant pas les vraies valeurs de θ, mais requérant tout de même l'information sur les classements Elo des joueurs, on la qualifie **d'erreur partiellement supervisée**.

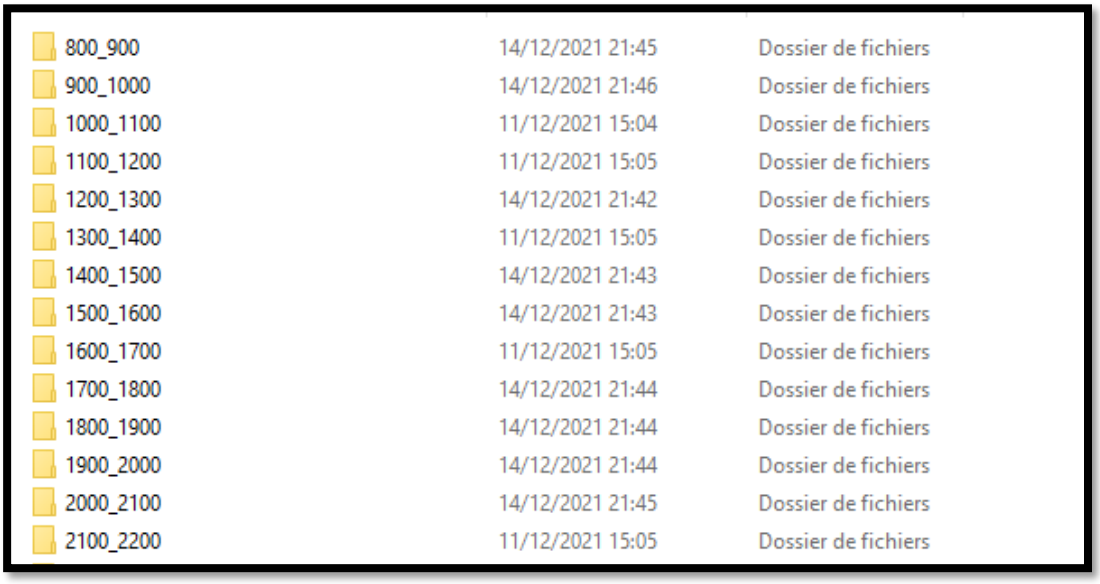
III- Constitution d'une base de données

Dans ce projet, le temps de calcul s'avère être une contrainte majeure, du fait du temps qu'il est nécessaire d'accorder à **Stockfish** afin que ce dernier puisse procéder à l'évaluation de différentes positions. Notamment, les calculs en profondeur élevés sont particulièrement exigeants.

On décide alors de construire une base de données, qui regroupera un nombre important de calculs déjà faits. Plus précisément, on se sert de l'API de la célèbre plateforme **chesscom**, qui nous permet de récupérer automatiquement un certain nombre de parties d'échecs.

Traitement de ces parties d'échecs

On réorganise alors ces parties selon le classement *ELO* des deux joueurs en les répartissant dans les catégories suivantes :



Par exemple, le dossier **1700_1800** ne fera uniquement intervenir des parties entre des joueurs dont les classements Elo se positionnent dans l'intervalle **[1700 ; 1800]**.

Or, étant donné que l'on souhaiterait à terme que notre approche soit statistique, on aspire à disposer d'un nombre important de données afin que nos estimateurs soient les plus proches des valeurs réelles qu'ils représentent.

Au total, nous avons fait analyser par Stockfish **5975** parties d'échecs, dans un délai de **725** heures (environ **1** mois de calculs).

Calculs sur le « BabyCluster »

L'analyse de ces parties représente une quantité importante de calculs.

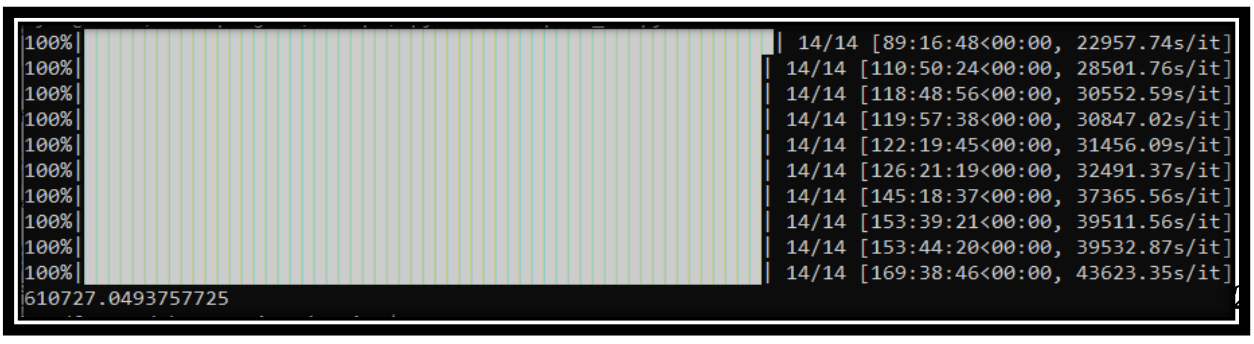
Ainsi, on se sert du BabyCluster, centre de calcul mis à disposition par l'IECL. En particulier, on se sert du module ***multiprocessing*** afin de pouvoir mener les calculs en parallèle, et gagner en efficacité.

Il se pose alors la question du nombre de processus N que l'on souhaite exécuter en parallèle, sachant que le serveur dont on se sert dispose de 4 cœurs. Afin de répondre à cette question, on compare les performances pour plusieurs valeurs de N sur une petite portion de cette base de données.

N	*Temps en secondes*
1	**150**
4	**57**
8	**46**
10	**40**
20	**40**
32	**40**

Naturellement, on atteint une valeur palier pour le temps de calcul lorsque N devient suffisamment grand : cela traduit simplement le fait que l'on dépasse la capacité du processeur à gérer en parallèle les N tâches auxquelles il est soumis.

Il semble alors naturel de choisir la valeur **N minimale** pour laquelle on atteint la valeur palier en temps : $N = 10$ est alors notre choix.



Interface Python après la fin de l'analyse
d'une partie des joueurs

Stockage de cette base de données

On fait le choix de conserver l'analyse relative à une partie d'un joueur sous la forme d'une table de valeurs M où chaque $M_{i,j}$ est un vecteur construit de la forme suivante :

$$M_{i,j} = [Evaluation\ initiale, x_1, x_2, \dots, x_m, index\ du\ coup\ joué, couleur\ du\ joueur]$$

avec $\boldsymbol{m}$ le nombre de coups légaux

$\boldsymbol{x_k}$ l'évaluation de la position *après* que le coup $\boldsymbol{k}$ ait été joué parmi les $\boldsymbol{m}$ coups légaux

$\boldsymbol{i}$ correspond à une indexation arbitraire de tous les coups dont on dispose sur ce joueur

$\boldsymbol{j}$ correspond à la **profondeur** utilisée pour fournir les évaluations ci-dessus.

$M_{i,j}$ est donc de taille $1 + m + 1 + 1 = m + 3$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	['+38', '-66', '+19', '0', '-51', '0', '+10', ...	['+82', '-66', '+19', '0', '-51', '0', '+10', ...	['+55', '-77', '+28', '+14', '-37', '+29', '+5...	['+22', '-62', '+22', '-3', '-196', '-162', '-...	['+54', '-62', '-54', '-3', '-62', '-16', '+16...	['+54', '-78', '+25', '+10', '-41', '-13', '-2...	['+50', '-85', '+26', '-20', '-53', '-2', '-17...	['+50', '-95', '+22', '-8', '-78', '-22', '+19...	['+54', '-89', '+23', '+21', '-44', '0', '+33'...	['+35', '-70', '+29', '-8', '-64', '+1', '0', ...	['+50', '-83', '+26', '+12', '-90', '-18', '+2...	['+37', '-83', '+32', '+7', '-79', '-4', '-8',...	['+33', '-77', '+26', '+13', '-86', '-4', '+20...	['+38', '-82', '+39', '0', '-100', '-20', '+21...	['+42', '-80', '+25', '-15', '-100', '-18', '+...
1	['+84', '-78', '+65', '+13', '-955', '+19', '-...	['+123', '-78', '+65', '+13', '-955', '+19', '...	['+156', '-13', '+94', '+45', '-955', '+52', '...	['+44', '-246', '+26', '-14', '-979', '-24', '...	['+75', '-246', '+26', '-14', '-927', '-24', '...	['+59', '-216', '+26', '-4', '-922', '-67', '-...	['+89', '-89', '+24', '-25', '-805', '-7', '-1...	['+92', '-72', '+62', '+23', '-893', '0', '-18...	['+62', '-95', '+54', '-1', '-751', '+34', '+2...	['+83', '-58', '+73', '+42', '-743', '+23', '-...	['+93', '-72', '+34', '+9', '-701', '+32', '-7...	['+78', '-75', '+62', '+15', '-646', '+10', '-...	['+57', '-75', '+75', '-27', '-674', '+28', '-...	['+62', '-109', '+39', '-59', '-646', '+30', '...	['+70', '-94', '+65', '-25', '-725', '+16', '-...
2	['+101', '-246', '+26', '+26', '-781', '-219',...	['+728', '-246', '+26', '+26', '-781', '-219',...	['+728', '-338', '+26', '+26', '-781', '-219',...	['+836', '-338', '+26', '-25', '-781', '-476',...	['+105', '-107', '+2', '+2', '-781', '-308', '...	['+115', '-210', '+48', '+22', '-677', '-213',...	['+98', '-129', '+75', '+75', '-781', '-30', '...	['+90', '-51', '+65', '+40', '-609', '-189', '...	['+99', '-176', '+62', '+77', '-667', '-135', ...	['+93', '-132', '+34', '+35', '-658', '-144', ...	['+86', '-190', '+31', '+15', '-679', '-199', ...	['+81', '-169', '+40', '+12', '-697', '-180', ...	['+78', '-116', '+8', '+12', '-694', '-194', '...	['+78', '-176', '+29', '+29', '-672', '-173', ...	['+88', '-160', '+27', '+22', '-669', '-194', ...
3	['+154', '+40', '+154', '+69', '-758', '+56', ...	['+300', '+40', '+300', '+69', '-758', '+56', ...	['+241', '+61', '+241', '+81', '-758', '+38', ...	['+125', '+54', '+55', '+51', '-684', '-27', '...	['+94', '+18', '+103', '+28', '-684', '-388', ...	['+93', '-3', '+56', '+37', '-742', '-109', '+...	['+87', '-20', '+87', '+75', '-755', '-7', '+2...	['+92', '+4', '+42', '+69', '-710', '-9', '+21...	['+95', '+5', '+50', '+92', '-661', '+6', '+44...	['+99', '+52', '+72', '+87', '-661', '0', '+35...	['+92', '+77', '+65', '+90', '-666', '+11', '+...	['+117', '+74', '+82', '+84', '-653', '-1', '+...	['+114', '+72', '+99', '+83', '-684', '+3', '+...	['+105', '+55', '+78', '+84', '-700', '+12', '...	['+105', '+61', '+76', '+72', '-661', '+7', '+...
4	['+75', '-865', '-32', '+56', '+31', '+47', '+...	['+156', '-865', '-32', '+56', '+31', '+47', '...	['+81', '-865', '-32', '+57', '+40', '+50', '+...	['+114', '-828', '-743', '+61', '+37', '+40', ...	['+75', '-800', '-48', '+72', '+46', '+40', '+...	['+61', '-806', '-83', '+43', '+32', '+8', '-1...	['+70', '-806', '-21', '+43', '+32', '+26', '+...	['+95', '-730', '-3', '+49', '+57', '+26', '0'...	['+73', '-760', '+13', '+31', '+45', '+19', '-...	['+82', '-735', '+16', '+35', '+46', '+25', '+...	['+84', '-720', '+24', '+50', '+59', '+49', '+...	['+84', '-724', '+3', '+47', '+56', '+26', '+5...	['+76', '-768', '+9', '+39', '+29', '+12', '+8...	['+78', '-748', '+12', '+43', '+47', '+31', '+...	['+96', '-707', '+30', '+39', '+63', '+38', '-...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
432	['+976', '+500', '-561', '-553', '-415', '+953...	['+902', '+473', '-561', '-553', '-482', '+469...	['+902', '+473', '-561', '-553', '-482', '+469...	['+845', '+473', '-530', '-517', '-482', '+472...	['+789', '+557', '-538', '-530', '-366', '+492...	['+792', '+551', '-500', '-507', '-436', '+540...	['+769', '+562', '-540', '-538', '-460', '+569...	['+648', '+475', '-584', '-566', '-405', '+557...	['+795', '+464', '-584', '-563', '-457', '+561...	['+723', '+482', '-581', '-589', '-420', '+593...	['+682', '+464', '-607', '-592', '-422', '+671...	['+730', '+470', '-609', '-593', '-400', '+666...	['+765', '+520', '-607', '-592', '-429', '+667...	['+746', '+493', '-607', '-592', '-415', '+700...	['+832', '+527', '-612', '-615', '-415', '+745...
433	['+946', '-515', '-507', '-515', '-387', '+926...	['+607', '-515', '-507', '-515', '-387', '+469...	['+609', '-515', '-507', '-515', '-387', '+449...	['+685', '-479', '-479', '-484', '-445', '+487...	['+738', '-500', '-492', '-500', '-332', '+500...	['+924', '-492', '-523', '-512', '-401', '+956...	['+730', '-532', '-555', '-584', '-457', '+753...	['+800', '-527', '-542', '-576', '-462', '+678...	['+746', '-571', '-569', '-592', '-455', '+650...	['+715', '-557', '-569', '-589', '-451', '+725...	['+730', '-565', '-576', '-624', '-439', '+600...	['+738', '-569', '-566', '-616', '-438', '+723...	['+730', '-571', '-578', '-638', '-407', '+746...	['+757', '-584', '-567', '-586', '-426', '+759...	['+791', '-600', '-601', '-646', '-392', '+784...

Exemple de table de valeurs M d'un joueur

	0	1	2	3	4	5	6	7	8	9	...	22	23	24	25	26	27	28	29	30	31
0	+946	-515	-507	-515	-387	+926	+917	+924	-515	+894	...	+438	+461	+446	+438	+446	+446	+438	+461	11	True

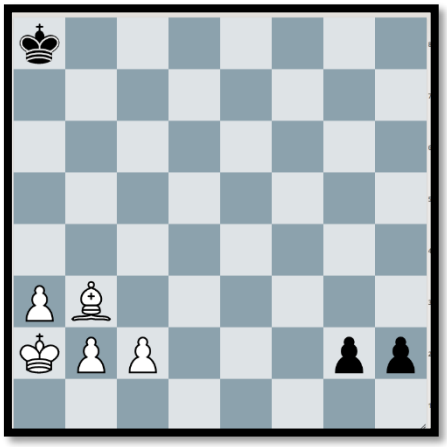
Zoom sur un coefficient de M

IV- Modèle mathématique

Transformation de la fonction d'évaluation

Nous l'avons vu plus haut, Stockfish renvoie une évaluation dans une échelle non bornée lorsque sa profondeur de calcul ne lui permet pas d'anticiper un mat forcé, et renvoie une chaine de caractère sous la forme $\# \pm k$ le cas échéant. Or, un tel fonctionnement peut poser au moins deux problèmes d'ordre pratique :

- L'interprétation de l'évaluation numérique, lorsqu'elle devient grande en valeur absolue, n'est pas tout à fait intuitive. Par exemple, en fin de partie, il est tout à fait courant que l'évaluation renvoyée par Stockfish soit de l'ordre de 5000 centipions[3]. Ainsi, tout se passe comme si le joueur gagnant disposait d'un avantage matériel supplémentaire de 50 pions, ou bien d'environ 6 dames. Or, on a alors du mal à saisir quelle est la différence avec une position dont l'évaluation est cette fois égale à 2500 centipions. En effet, cette évaluation est deux fois inférieure à la première, pourtant, le constat est le même dans les deux cas : le joueur en présence d'un tel avantage a une position écrasante sur son adversaire et gagnera probablement la partie.



Trait au blanc
+6410 CP
D'après Stockfish profondeur 23

[3] 100 centipions représentent un pion. De plus, il est courant de considérer que 9 pions représentent l'équivalent d'une dame en termes d'avantage matériel, 5 pions représentent une tour etc. On abrégera l'unité des centipions par CP.

- De plus, dans le cas où **Stockfish** anticipe un mat forcé en k coups, l'évaluation n'est alors plus numérique ni quantifiable. Concrètement, on est gêné par le fait qu'on ne puisse plus calculer la différence entre les évaluations des mats forcés, et des évaluations numériques. Par exemple, une expression telle que $\mathbf{60\ CP - (\# + 4)}$ n'a alors plus de sens. Or, nous aurons intérêt à pouvoir calculer une telle différence dans la suite, afin d'attribuer un score à chacun des coups qui sont joués.

Ainsi, on propose d'introduire la **fonction de transformation d'évaluation** suivante, qui permet de s'affranchir des deux problèmes ci-dessus. On note $\boldsymbol{x}$ et $\text{"\#} \pm n\text{"}$ deux évaluations renvoyées par **Stockfish** et on définit alors f_a comme suit :

$$\epsilon = 0.01,\ \boldsymbol{a}\ \textit{paramètre de distorsion à déterminer}$$

$$f_a(x) = (1-\epsilon) \times \frac{x}{|x| + a}\ \textit{si Stockfish n'indique pas de mat}$$

$$f_a(\text{"\#} \pm\ n\text{"}) = \pm\left(1 - \frac{\epsilon \times n}{1+n}\right)$$

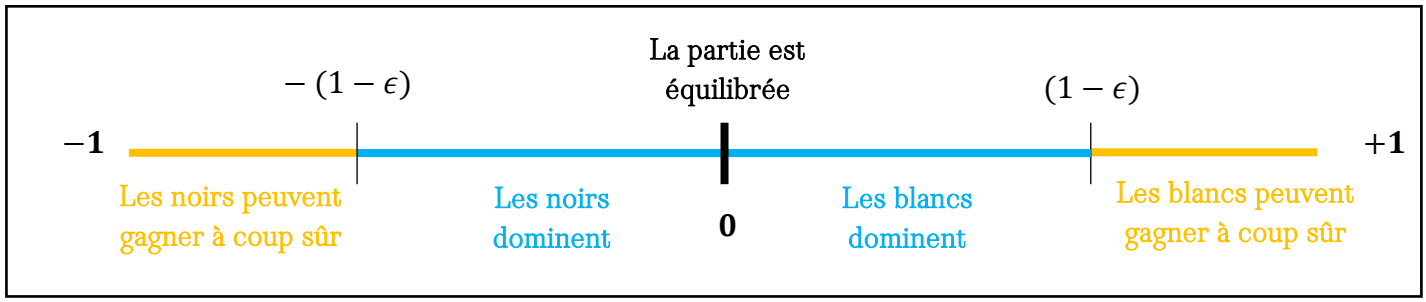
Désormais, lorsque nous mentionnerons le terme « **évaluation** », il sera toujours entendu implicitement « **évaluation transformée par la fonction f_a** »

Ainsi, l'évaluation prend toujours ses valeurs dans l'intervalle $[-1,\ 1]$, et peut alors être interprétée comme une **espérance de gain**. Par exemple, si l'évaluation est égale à 0, les deux joueurs ont 50% de chance chacun de gagner. Différemment, si l'évaluation est égale à $+0{,}3$, on considère que le joueur blanc a 65% de gagner la partie[4].

Il faut noter la présence du paramètre ϵ, qui permet de hiérarchiser les évaluations en absence et présence de mat forcés. Ce paramètre permet d'intégrer le fait rationnel suivant : **une position aussi avantageuse soit-elle, sera toujours d'évaluation inférieure à une position où il est possible de mater en un nombre fini de coups.**

Ainsi, si aucun mat n'est en vue, la probabilité que le joueur blanc ou noir gagne est majorée par $1 - \frac{\epsilon}{2}$. Au contraire, dès lors qu'il est possible de mater à coup sûr, la probabilité que l'un des deux joueurs gagne est minorée par cette quantité.

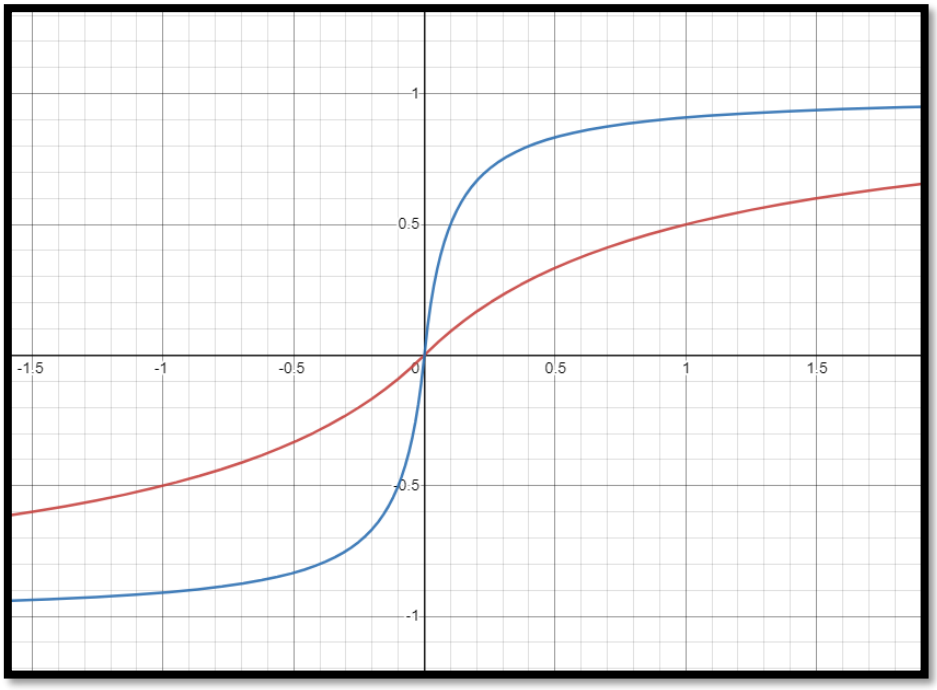
[4] On applique la transformation affine $x \to \frac{1}{2}(x+1)$ pour envoyer $[-1,\ 1]$ dans $[0\,,1]$ et estimer les probabilités de gains



Interprétation de l'évaluation transformée par f_a

Choix du paramètre de distorsion a

Le lecteur attentif aura également remarqué la présence du paramètre a qu'on appelle « paramètre de distorsion » intervenant dans la fonction de transformation de l'évaluation. Ce dernier paramètre va nous permettre d'ajuster la vitesse d'aplanissement de la courbe de la f_a. En particulier, nous nous servirons de ce paramètre a afin d'émuler un joueur plus ou moins bon dans une section suivante.



Tracé de $f_{a=1}$ en rouge et $f_{a=0.1}$ en bleu

Score :

Dans ce paragraphe, on définit la notion de Score, qui va jouer un rôle majeur dans notre modèle mathématique.

Soit $\boldsymbol{X}$ une position. Soit $\boldsymbol{i}$ un coup légal dans cette position. En notant $\boldsymbol{X_i}$ la position obtenue après que le coup $\boldsymbol{i}$ ait été joué, on définit alors :

$$Score_i(X) = couleur(\boldsymbol{X_i}) \times (\text{Evaluation } X - \text{Evaluation } X_i)$$

Intuitivement, $Score_i\ (X)$ représente alors la qualité du coup i qui a été joué dans la position X. Cette quantité est théoriquement[5] toujours positive par construction de la quantité $Evaluation\ X$ qui représente l'évaluation de la partie après que le joueur ait joué le meilleur coup. On peut interpréter cette quantité comme l'évaluation « dans le meilleur des cas ». Alors, le mieux que puisse faire un joueur, c'est ne pas dégrader cette dernière.

Ainsi, les meilleurs coups indiqueront un $Score$ très faible (voire nul) tandis que les coups les plus mauvais engendreront un $Score$ élevé.

Exemple :

Imaginons une position où c'est au tour des noirs de jouer. Avant que les noirs ne jouent leur prochain coup, l'évaluation est de $-0{,}50$. Les noirs jouent un coup qui amène l'évaluation à $-0{,}49$.

➔ On a alors par définition : $Score\ = -1 \times \big(-0{,}50 - (-0{,}49)\big) = 0{,}01$. Le $Score$ de ce coup étant très proche de 0, c'c'est essentiellement le meilleur coup qu'on puisse jouer dans cette position.

[5] Des effets de bruits aléatoires liés à la découverte de nouveaux horizons sont susceptibles de rendre le score parfois « légèrement » négatif. Néanmoins, cela ne pose pas de problèmes dans la pratique.

Construction du premier modèle

Hypothèses

L'objectif ultime de ce projet est de parvenir à estimer le niveau d'un joueur d'échecs, à partir d'une ou plusieurs de ses parties. On suppose donc qu'il **existe** pour tous les joueurs, une quantité θ qui caractérisent leur **niveau**. Tout l'enjeu de notre démarche est alors d'estimer un tel nombre pour un joueur donné.

De plus, on suppose pour des raisons pratiques que θ prend ses valeurs dans l'ensemble discret $\Omega = \{1, 2, \dots, 15\}$. Ce choix est arbitraire, et n'importe en réalité que peu. En effet, si l'on parvient à établir un modèle qui permet de discriminer les joueurs selon leur niveau, il ne s'agira alors plus que de procéder à un étalonnage pour se ramener à une autre échelle[6].

Enfin, on suppose que les différents coups qui sont joués dans une partie sont **indépendants.** Cette hypothèse est très raisonnable, car le joueur d'échecs va naturellement réévaluer la position à chaque coup et repartir depuis zéro « mentalement » : il y est en effet contraint, car entre temps, son adversaire aura joué un nouveau coup, altérant la position.

Notion de vraisemblance

Notre approche va alors consister à trouver quelle valeur de θ est la plus vraisemblable au vu des coups joués par le joueur dont on cherche à estimer le niveau. Mathématiquement, l'outil le plus adéquat pour parvenir à nos fins est naturellement la **vraisemblance.**

On note alors Y la **variable aléatoire** des coups joués par le joueur, et y^* la réalisation d'une telle variable. Il est important de remarquer que Y contient également l'information des positions successives de la partie, car il ne nous suffit pas de savoir que le joueur a joué « pion C4 » dans l'absolu. En effet, un tel coup pourrait très bien être excellent ou mauvais selon la position. On souhaite alors être informé par exemple que le joueur a joué « pion C4 dans la **position X** ». Enfin, il convient de préciser que nous raisonnons comme si les coups joués par les adversaires auxquels notre joueur s'est confronté étaient des

[6] A terme, il serait idéal de pouvoir attribuer un niveau θ dans une échelle qui coïncide avec l'Elo officiel, ou bien avec l'Elo d'une plateforme d'échecs en ligne.

paramètres fixes, c'est-à-dire qu'on ne considère l'aléa que provenant du joueur dont on étudie les coups.

On dispose alors de tous les outils pour définir la **vraisemblance**, dont l'expression est :

$$\mathcal{L}(\theta) = P(Y = y^* \mid \boldsymbol{\theta})$$

Or, Y peut s'écrire $Y = (Y_1, \dots, Y_n)$ avec n le nombre de coups connus du joueur, et où Y_i représente l'information sur le coup i. Ainsi, par hypothèse d'indépendance entre les coups, on a alors :

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} P(Y_i = y_i^* \mid \boldsymbol{\theta})$$

À mesure que n devient grand, la vraisemblance $\mathcal{L}$ peut devenir très petite et provoquer des erreurs numériques. On préfère alors manipuler la **log-vraisemblance** $\log \mathcal{L}$ qui est plus commode :

$$\log \mathcal{L}(\theta) = \sum_{i=1}^{n} \log P(Y_i = y_i^* \mid \boldsymbol{\theta})$$

Ainsi, le joueur dont on cherche à estimer le niveau se verra attribuer le niveau θ_{max} qui maximise la vraisemblance. De ce fait, on a :

$$\boldsymbol{\theta_{max} = argmax \{ \mathcal{L}(\theta) / \theta \in \{1, \dots, 15\} \}}$$

Par conséquent, on observe que pour pouvoir calculer une telle quantité, il faut d'abord pouvoir calculer les vraisemblances $\boldsymbol{\mathcal{L}(\theta)}$; autrement dit, il s'agit à ce stade de définir la façon dont nous calculerons nos probabilités, ce que nous développons dans la section suivante.

Calcul de la vraisemblance

Dans la section précédente, nous avons rendu compte du fait qu'afin d'estimer valeur de θ, il impliquait de pouvoir calculer la vraisemblance $\mathcal{L}(\theta) = \prod_{i=1}^{n} P(Y_i = y_i^* \mid \boldsymbol{\theta})$. On détaille ci-dessous la démarche qui permet d'aboutir au calcul d'une telle quantité.

Concentrons-nous sur le $i^{ème}$ coup joué par notre joueur. Ce dernier a joué ce coup dans une certaine position. Dans cette position, le joueur avait le choix entre m coups légaux $C_i = (coup_1, .., coup_m)$ et a décidé de jouer le $j^{ème}$ coup légal, c'est-à-dire $coup_j$. Quelle était la probabilité qu'il joue un tel coup ?

Naturellement, le joueur rationnel aura joué le coup qu'il estimait être le meilleur parmi la liste des coups qu'il était autorisé à jouer, **depuis son point de vue.** Assurément, cette notion de « meilleur coup » varie d'un joueur à l'autre, autrement, tous les joueurs auraient le même niveau ! Par conséquent, en se plaçant depuis le point de vue d'un joueur d'un certain niveau θ, nous procédons au calcul des scores de chacun des coups :

$$Score_\theta(C_i) = -1 \times (Score_{coup_1}, .., Score_{coup_m})$$

Où $Score_{coup_k}$ est calculé avec une **profondeur** de calcul de P_θ et un paramètre de **distorsion de la fonction d'évaluation a_θ.** Par exemple, plus le joueur est fort (i.e θ élevé), plus il devra être approché par une profondeur de calcul P élevée, et un paramètre a faible. On déploie alors dans une section suivante la façon dont seront paramétrés P et a en fonction de θ.

A ce stade, il ne s'agit alors plus que d'appliquer la fonction $softmax_T$ au vecteur $Score_\theta(C)$, avec T un paramètre pris constant pour le moment. Cela nous permet de nous ramener à un vecteur où chacune des coordonnées désigne une probabilité ; en appelant Λ un tel vecteur, on définit alors :

$$\boldsymbol{P(Y_i = y_i^* \mid \theta) \coloneqq \Lambda_j = \frac{\exp\left(-Score_{coup_j} \times T^{-1}\right)}{\sum_k \exp\left(-Score_{coup_k} \times T^{-1}\right)}}$$

Paramétrisation

Calibrage du paramètre $\boldsymbol{a}$

Soit β une valeur représentant la probabilité de gagner d'un joueur ayant obtenu un avantage décisif. Nous prendrons en l'occurrence $\beta = 90\%$ dans un premier temps, même si nous verrons dans la suite que ce paramètre sera amené à être optimisé numériquement.

On note alors $\Delta_\beta(\theta)$ l'avantage dont doit disposer un joueur pour qu'il ait la probabilité β de gagner. Naturellement, cette quantité dépend du niveau du joueur. Un très bon joueur pourra conclure avec un avantage minime, tandis qu'un très mauvais joueur exigera un avantage plus conséquent pour gagner.

On considère qu'un très bon joueur (de niveau professionnel par exemple) a une probabilité β de gagner lorsque son avantage est de $\Delta_\beta = 100$ CP.

Entre autres, nous sommes en train de dire que le joueur excellent aura quasiment gagné la partie s'il parvient à obtenir un avantage de $1\ PION$ net face à son adversaire. Ce choix peut paraître brutal et exagéré pour lecteur. Néanmoins, en réalité, les parties de très haut niveau sont très serrées, de sorte qu'un pion net en faveur d'un joueur dans une position relativement équilibrée ou symétrique suffise à garantir la victoire de ce dernier.

Pour nuancer, il est vrai qu'un joueur peut être dans une situation perdante malgré un avantage matériel, car il faut également tenir compte de la position de la partie. On laisse naturellement **Stockfish** tenir compte de ces subtilités ; si **Stockfish** attribue **$+100$** à un joueur A en désavantage matériel, c'est qu'il considère que **tout se passe *comme* si le joueur A avait un pion en plus, car son avantage positionnel compense son déficit matériel**.

De la même façon, on considère qu'un très mauvais joueur doit en gros avoir un avantage net $\Delta_\beta = 900$ CP (valeur d'une **dame**) pour pouvoir conclure la partie.

Connaissant les valeurs aux extrémités de la fonction Δ_β, il faut désormais définir le comportement de cette fonction pour des valeurs intermédiaires de θ. On observe alors deux choses :

- On souhaiterait que $\Delta_\beta \to 0$ lorsque $\theta \to +\infty$, ce qui traduirait que le joueur $parfait$ n'a besoin que d'un avantage infinitésimal pour gagner une partie.

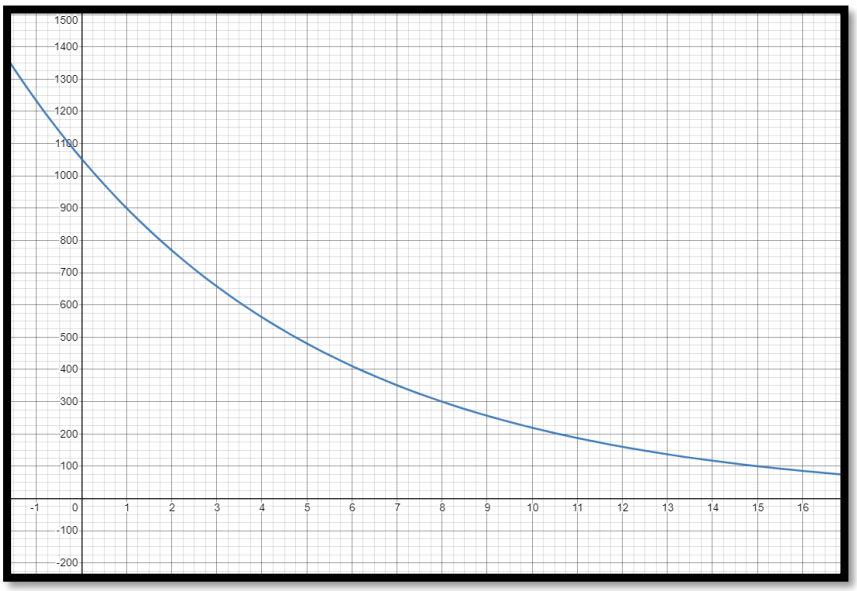
- La progression d'un joueur n'est pas linéaire, au sens où plus l'on devient fort, plus il est dur de devenir fort. On souhaiterait donc intuitivement que le saut entre

$\Delta_\beta(\theta = 1)$ et $\Delta_\beta(\theta = 2)$ soit plus grand que celui entre $\Delta_\beta(\theta = 14)$ et $\Delta_\beta(\theta = 15)$. Autrement dit, on cherche une fonction au comportement **convexe.**

À la lumière de ces éléments, on propose la fonction suivante :

$$\Delta_\beta(\theta) = \exp(u \times \theta + v)$$
$$\Delta_\beta(\theta = 1) = 900$$
$$\Delta_\beta(\theta = 15) = 100$$

Avec les contraintes aux bords, on trouve les constantes u, v



Tracé de la fonction $\theta \rightarrow \Delta_\beta(\theta)$

Enfin, par construction de Δ_β, on a l'équation suivante, d'inconnue a, qui est satisfaite :

$$\forall\, \theta \in \{1, \dots, 15\}, f_a\left(\Delta_\beta(\theta)\right) = \beta$$

Cette équation peut facilement être résolue analytiquement, et on dispose alors d'une valeur a_θ calibrée pour tous les niveaux θ.

θ	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0
a	37.9	32.4	27.7	23.7	20.2	17.3	14.8	12.6	10.8	9.2	7.9	6.7	5.8	4.9	4.2

Tableau récapitulatif des valeurs du paramètre a

Choix de la température T

Le choix de la température a des conséquences lourdes sur les résultats. Si l'on prend une température trop faible ou trop élevée, on arrive à des extrêmes dans les deux cas.

- En effet, si la température est trop faible, il ne tient alors seulement à ce qu'un seul coup légèrement plus improbable que les autres soit joué pour arriver à des conclusions abusives telles que « Si Monsieur X avait réellement un niveau $\theta = 10$, il n'aurait jamais joué ce coup ! Peu importe ce qu'il joue dans la suite de la partie, désormais c'est certain, il n'est pas de niveau $\theta = 10$ »

- Au contraire, si la température est trop élevée, on permet à peu près tout et n'importe quoi : « Un tel coup aurait très bien pu être joué par un joueur de niveau $\theta = 1$ particulièrement en forme, ou par un joueur de niveau $\theta = 15$ légèrement fatigué... Il est impossible de se prononcer. »

On comprend naturellement pourquoi il est crucial de choisir de manière pertinente le paramètre température. Néanmoins, quitte à choisir, on préfère plutôt « surestimer » la température. En effet, il est d'usage courant en statistiques de privilégier les priores à queues lourdes. L'idée est de se laisser un peu de *marge* en accordant toujours une faible probabilité de réalisation plutôt qu'une probabilité presque nulle aux évènements les moins probables. Ainsi, on adopte le même état d'esprit pour le choix de la température : on va préférer une température un peu plus grande plutôt qu'un peu plus petite.

Ces considérations nous amènent à prendre une température T égale à $0,5$ dans un premier temps.

Choix de la profondeur P

Désormais, il s'agit de procéder au paramétrage de la profondeur de calcul de notre moteur d'échecs Stockfish P_θ en fonction de θ. Nous le rappelons, P_θ désigne la profondeur jusqu'à laquelle Stockfish parcourt « l'arbre des coups » lorsqu'il procède à l'évaluation d'une position. Intuitivement, la profondeur P_θ va quantifier le nombre de coups qui seront anticipés par Stockfish.

De manière équivalente, bien que le joueur humain ne raisonne pas exactement sous la forme d'arbres en son for intérieur, les joueurs vont se distinguer les uns des autres par leurs aptitudes à pouvoir se représenter mentalement une position après qu'un certain nombre de coups ait été joué. Ainsi, tandis que le très bon joueur n'aura aucune peine à anticiper jusqu'à une dizaine de coups, le joueur amateur trouvera déjà bien compliqué de se représenter en amont les conséquences de trois ou quatre coups d'affilée. De ce fait, on sent que chaque joueur dispose une profondeur de calcul intrinsèque qui le caractérise, et qui constitue une partie du niveau de ce dernier.

Par ailleurs, il est raisonnable de considérer que le gain en termes de niveau de jeu conféré par un incrément d'une unité de la profondeur est le même, quelle que soit la valeur initiale de la profondeur. Par exemple, que le joueur voie sa profondeur intrinsèque évoluer de 2 à 3 ou de 10 à 11, toutes choses égales par ailleurs, on considérera que le gain en niveau de jeu est semblable. En effet, on motive cela par le fait que la capacité à pouvoir anticiper un coup supplémentaire octroie en moyenne toujours le même avantage.

Ces considérations nous amènent donc à imposer une **relation linéaire** entre P_θ et θ. Or, il se trouve que le cardinal de l'espace Ω dans lequel nous cherchons θ coïncide avec la profondeur de calcul maximale à laquelle nous souhaitons nous restreindre[7], à savoir 15. Adeptes du principe du rasoir d'Occam, **nous fixons alors $P = \theta$**

[7] À cause de l'explosion combinatoire, les temps de calcul deviennent excessivement lents au-delà.

Matrice des duels de Stockfish : suppression des faibles profondeurs

On souhaite s'assurer du bon paramétrage de **Stockfish** ainsi que notre compréhension de la notion de meilleur coup, on peut faire s'affronter le *vrai* **Stockfish** contre la version modifiée que nous utilisons.

Contexte :

- Joueur ***A*** : Stockfish original de paramètre $X = (1 \leq Profondeur \leq 15, Skill\ level^{8} = Profondeur)$
- Joueur ***B*** : Stockfish à température nulle de paramètre $Y = (1 \leq Profondeur \leq 15)$
- ***A*** et ***B*** s'affrontent **20** fois pour chaque (X, Y). Chacun joue **10** fois les noirs et **10** fois les blancs.
- On dénombre donc $15 \times 15 \times 20 = 4500$ parties à simuler.
- Dans la matrice, le score qui est indiqué correspond **au score depuis le point de vue de B (donc depuis le point de vue de *notre* Stockfish)**
  Exemple : S'il est indiqué 20. 0, cela signifie que le joueur B a gagné les 20 parties.
- Une victoire octroie **1** point, une défaite ne fait gagner aucun point, un match nul fait gagner ½ aux deux joueurs.
- Les colonnes correspondent à Y et les lignes correspondent à X

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	18.00	13.00	18.50	15.50	18.50	20.00	19.50	19.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
2	17.00	11.50	13.00	11.50	16.00	17.50	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
3	17.50	8.00	7.00	10.00	14.50	19.00	17.50	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
4	12.00	6.50	3.00	7.00	10.50	18.50	19.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
5	7.00	0.50	1.00	1.00	4.00	9.50	13.50	19.00	19.00	20.00	20.00	20.00	20.00	20.00	20.00
6	4.50	0.00	0.00	0.50	1.00	5.50	13.00	19.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
7	1.50	0.00	0.00	0.50	0.00	2.00	8.00	15.50	19.00	20.00	19.50	20.00	20.00	20.00	20.00
8	1.50	0.00	0.00	0.00	0.00	1.00	2.00	10.00	13.50	18.00	20.00	20.00	19.50	20.00	20.00
9	0.00	0.00	0.00	0.00	0.00	0.00	1.00	3.50	10.00	14.50	19.50	19.50	19.00	20.00	20.00
10	1.00	0.00	0.00	0.00	0.00	0.50	1.00	2.00	4.00	13.00	18.50	19.00	20.00	19.50	20.00
11	0.50	0.00	0.00	0.00	0.50	0.00	0.50	2.50	7.50	12.00	14.50	19.00	19.50	20.00	20.00
12	0.00	0.00	0.00	0.00	0.00	0.00	0.50	1.50	5.50	10.00	12.00	13.00	18.50	19.00	20.00
13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	3.00	6.50	11.00	16.50	16.50	18.50	19.00
14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	5.50	5.00	11.50	11.50	17.50	18.50
15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	1.00	1.50	5.00	6.50	8.00	12.00	18.00

[8] Le « Skill level » est un paramètre de Stockfish, permettant de moduler le niveau de jeu de ce dernier. Lorsque ce paramètre est élevé, Stockfish joue presque les meilleurs coups. Au contraire, si le *Skill level* est bas, le niveau de jeu de Stockfish sera fortement dégradé.

Comme attendu, le joueur ***B*** domine très fortement dans la partie triangulaire supérieure de la matrice. En effet, dans cette région de la matrice, le joueur ***B*** est avantagé car il dispose d'une profondeur de calcul supérieur, et joue systématiquement les meilleurs coups, contrairement à son adversaire. En revanche, dans la partie triangulaire inférieure de la matrice, c'est bien le joueur ***A*** qui domine, dont la profondeur de calcul compense largement l'affaiblissement stratégique causé par un ***Skill level*** moindre.

En revanche, on remarque une singularité inattendue dans la première colonne de la matrice. Le joueur ***B*** de profondeur seulement égale à **1** parvient à vaincre le joueur ***A*** jusqu'à la profondeur **4** ! Or, un tel écart de profondeur ne devrait théoriquement pas permettre au joueur ***B*** une telle performance. De plus, la colonne **1** présente de meilleures performances que la colonne **2**, ce qu'on ne parvient à expliquer de manière rationnelle. Comment une version plus affaiblie de Stockfish peut-elle mieux performer ?

Le fin mot de cette histoire nous demeure inconnu, et suggère peut-être que les développeurs de l'outil Stockfish aient choisi d'implémenter un comportement singulier pour les profondeurs **1** et **2**.

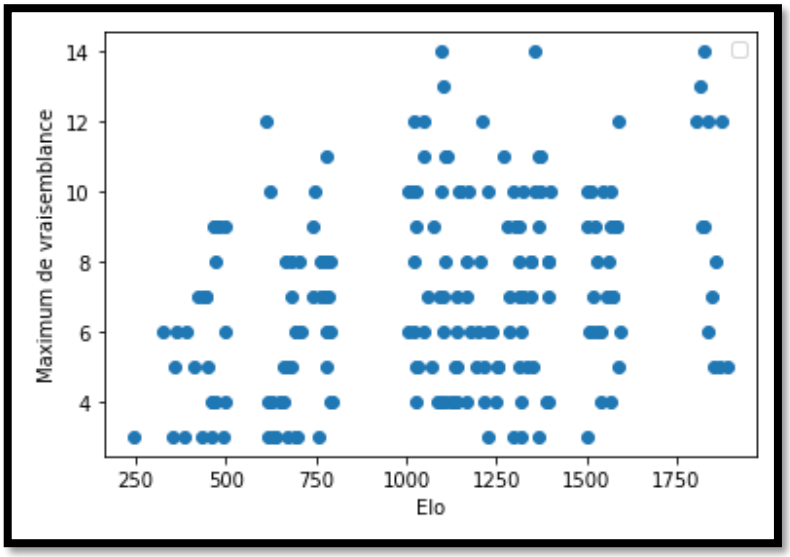
Quoi qu'il en soit, ces considérations nous amènent donc à **supprimer de notre modèle toute influence des versions de profondeurs 1 et 2 de Stockfish**, car nous jugeons ces dernières nuisibles à notre étude.

Premiers résultats

Précédemment, nous avons construit un modèle et avons avancé une première façon de le paramétrer. Nous tirons alors 200 joueurs aléatoirement dans la base de données et observons les résultats de nos prédictions sur ces derniers.

	Max Vraisemblance Moyen	Elo	Nombre de joueurs
0	3.000000	(200, 300)	1
1	4.833333	(300, 400)	6
2	5.947368	(400, 500)	19
3	6.000000	(500, 600)	1
4	5.333333	(600, 700)	21
5	6.875000	(700, 800)	24
6	7.909091	(1000, 1100)	22
7	7.291667	(1100, 1200)	24
8	6.666667	(1200, 1300)	18
9	7.482759	(1300, 1400)	29
10	7.423077	(1500, 1600)	26
11	9.000000	(1800, 1900)	13

Tableau opposant une tranche d'Elo et le maximum de vraisemblance moyen dans celle-ci



Nuage de points du maximum de vraisemblance confronté au classement Elo

Erreur partiellement supervisée = **187**

Les résultats obtenus sont alors mitigés. D'une part, le modèle ne semble pas totalement « à côté de la plaque », étant donné qu'il est possible d'observer une légère tendance croissante dans la colonne du maximum de vraisemblance moyen. Cela viendrait traduire le fait que le niveau prédit est d'autant plus grand que le classement Elo du joueur est bon. De plus, en appliquant une méthode de Monte Carlo[9], **on estime qu'en répondant complètement au hasard,** la probabilité d'obtenir une erreur inférieure à 200 est au plus[10]

[9] Méthode de Monte-Carlo classique avec $N = 10^5$ simulations, appliquée à l'estimation de $E(1_{g(X)\leq 200})$ où $g(X)$ renvoie l'erreur induite par les prédictions X et $X \sim \mathcal{U}(1;15)^{\otimes n}$ avec n le nombre de joueurs.

[10] Borne supérieure de l'intervalle de confiance à 95% obtenu par la méthode de Monte-Carlo.

égale à 3%. Cela vient appuyer le constat que notre modèle fait mieux que « répondre au hasard » car il se situe clairement au-delà du seuil de significativité.

Toutefois, les résultats demeurent insatisfaisants tant l'on peine à distinguer la corrélation positive entre le classement Elo et le niveau prédit dans le nuage de points, sans mentionner combien les prédictions semblent bruitées.

Dans la section suivante, nous passons en revue différentes méthodes mises en œuvre permettant d'améliorer le modèle.

V- Amélioration du modèle

Optimisation des paramètres

Nous disposons désormais d'une métrique permettant d'attester objectivement de la qualité des prédictions de notre modèle. Par conséquent, on peut chercher à optimiser les divers paramètres ayant été fixés arbitrairement jusqu'ici. Autrement dit, on essaie de sélectionner nos paramètres de sorte à **minimiser l'erreur partiellement supervisée.**

Après tâtonnement, il est apparu clair que les paramètres qui influencent substantiellement les prédictions sont les suivants : $\beta, \Delta_\beta(\theta = 1), \Delta_\beta(\theta = 15), T$. Or, il est possible de montrer qu'une hausse ou une baisse du paramètre β revient à multiplier par un facteur constant la famille $(\Delta_\beta(\theta = i))_{1 \leq i \leq 15}$.

Ainsi, pour s'affranchir des redondances, β est arbitrairement fixé à 0,75 et l'on définit alors le vecteur $Z = (\Delta_\beta(\theta = 1), \Delta_\beta(\theta = 15),\ T)$. Tout l'enjeu est alors de trouver un tel Z optimal, noté Z^*.

En se renseignant dans la littérature sur les méthodes les plus judicieuses pour optimiser les paramètres d'un modèle, il a été évoqué maintes fois **l'optimisation bayésienne**, particulièrement adaptée pour les problèmes de minimisation de fonctions coûteuses en calculs. Or, le calcul de notre fonction d'erreur est particulièrement lourd, car il nécessite de reproduire toutes les prédictions à chaque changement de paramètre, aussi léger soit-il. De plus, le gradient nous est étant évidemment inconnu, cette méthode semble d'autant plus adaptée à notre problème.

Brièvement, l'optimisation bayésienne consiste à considérer que les observations des différentes valeurs de l'erreur sont a priori les réalisations d'un **processus gaussien**. Par conséquent, cette hypothèse nous permet d'approximer en tout point de l'espace la fonction qu'on cherche à minimiser, en considérant l'espérance d'un tel processus.

Par exemple, ci-dessus, nous avons observé 8 points de la fonction dont on cherche le minimum. On peut alors tracer en pointillé une approximation de la fonction, à partir de l'espérance. Le « halo » qui entoure la courbe noire désigne l'incertitude (i.e la variance), qui est d'autant plus grande que deux points sont éloignés.

Il est alors défini une fonction **d'acquisition,** qui viendra proposer un nouveau point à explorer.

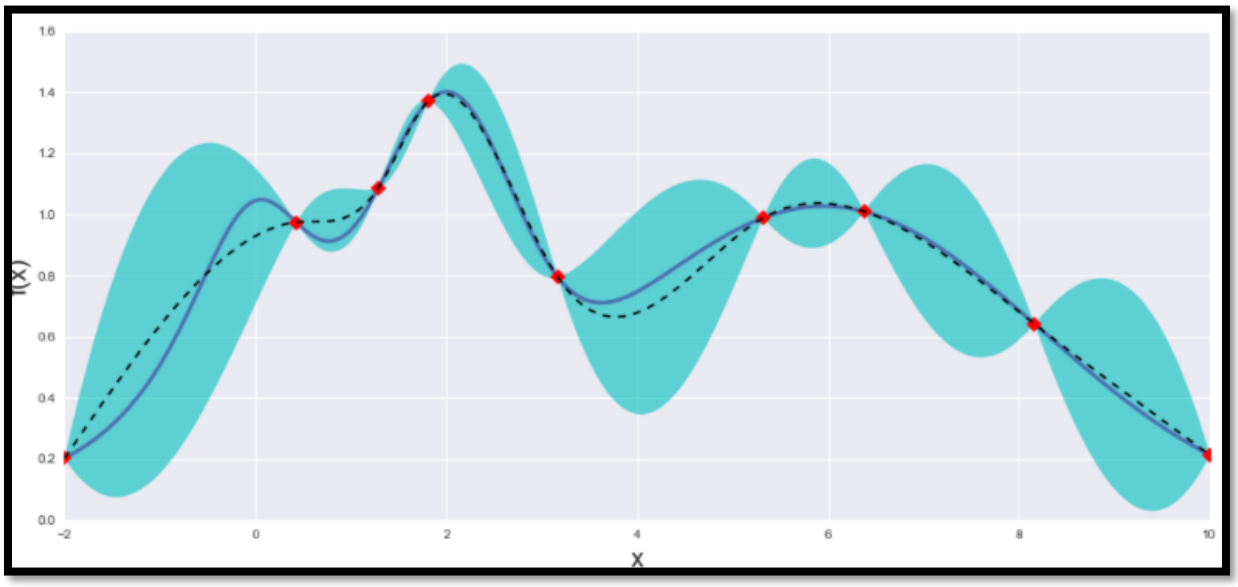


Illustration de l'approximation d'une fonction f (en bleu) par
un processus gaussien grâce aux observations (points rouges)
Source : https://github.com/fmfn/BayesianOptimization

Par exemple, une fonction d'acquisition populaire est la suivante :

$$u(x) = \max\big(0, f_0 - f(x)\big)$$

avec f la fonction à minimiser, et f_0 le meilleur minimum observé. Le nouveau point qui sera exploré sera celui qui maximise l'espérance $E(u(x)\,|observations\ précédentes)$.

Ainsi, après une centaine d'itérations d'un tel algorithme, la solution Z^* optimale rencontrée est la suivante :

$$\boldsymbol{Z^* = (\Delta_\beta(\theta = 1)^*, \Delta_\beta(\theta = 15)^*, T^*) = (3306; 79; 0,23)}$$
$$\textbf{Erreur partiellement supervisée} = \mathbf{155}$$

L'optimisation est ainsi satisfaisante, car elle permit de diminuer d'environ 20% l'erreur.

Par conséquent, dans toute la suite seront utilisés les paramètres suivants :

$$\boldsymbol{(\beta, \Delta_\beta(\theta = 1)^*, \Delta_\beta(\theta = 15)^*, T^*) = (0,75; 3306; 79; 0,23)}$$

Variation de la température en fonction de $\boldsymbol{\theta}$

Jusqu'ici, nous avions décidé de garder constante la température T en fonction du niveau de jeu θ : désormais, nous rendons variable ce paramètre. Ce choix est motivé par l'hypothèse que le joueur a un jeu qui s'approche de plus en plus de celui de Stockfish à mesure qu'il progresse.

Par conséquent, les évaluations de Stockfish ont moins intérêt à être bruitées lorsqu'on souhaite savoir combien il est vraisemblable que le joueur ait un bon niveau. Le paramètre T étant le principal régisseur d'un tel bruit, on cherche à construire une fonction décroissante de ce dernier.

On prend $T = T_\infty(1 + \exp(-a \times \theta + b))$

$T_\infty = 0{,}1$
$a = 0{,}1$
$b = 0{,}5$

Avec de tels paramètres, la température T prend alors les valeurs suivantes :

```
Theta 1 --> Temperature : 0.249
Theta 2 --> Temperature : 0.235
Theta 3 --> Temperature : 0.222
Theta 4 --> Temperature : 0.211
Theta 5 --> Temperature : 0.2
Theta 6 --> Temperature : 0.19
Theta 7 --> Temperature : 0.182
Theta 8 --> Temperature : 0.174
Theta 9 --> Temperature : 0.167
Theta 10 --> Temperature : 0.161
Theta 11 --> Temperature : 0.155
Theta 12 --> Temperature : 0.15
Theta 13 --> Temperature : 0.145
Theta 14 --> Temperature : 0.141
Theta 15 --> Temperature : 0.137
```

Modèle à profondeur aléatoire géométrique

Lorsque le joueur calcule différentes combinaisons aux échecs, il ne va pas **rigoureusement calculer à profondeur constante** : il existerait comme un « bruit », lui permettant tantôt de calculer à profondeur supérieure, et tantôt le contraignant à une profondeur inférieure.

On pourrait alors conjecturer qu'au moment de calculer, partant de la profondeur la plus basse, le joueur franchit mentalement avec succès chaque palier de profondeur avec une certaine probabilité, jusqu'à échouer à un certain point. Le dernier palier de profondeur franchi constituerait alors la profondeur avec laquelle il calculerait son prochain coup, cette expérience devant être réitérée à chaque nouveau coup. On modélise alors ce phénomène par la loi de probabilité adéquate, à savoir la **loi géométrique.**

Admettons que nous souhaitons émuler le comportement d'un joueur, dont la profondeur intrinsèque de calcul est P^*. Appelons alors **profondeur effective** cette quantité. Ainsi, considérons $X \sim Géometrique(q)$ (dans sa version à support sur $\mathbb{N}^*$) avec $q = \frac{1}{P-2}$ et $Y = \min(X+2, P_{max})$ avec P_{max} correspondant à la profondeur maximale permise, égale dans la pratique à $P_{max} = 15$.

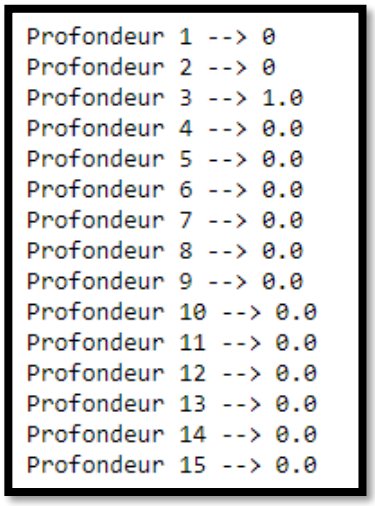
Nous souhaitons éliminer l'influence des profondeurs 1 et 2 qui ont révélé des comportements bien trop singuliers (cf page 25), et perturbateurs, c'est pourquoi la variable aléatoire X est translatée de deux unités (son espérance étant ajustée en conséquence).

Ainsi, on a $\forall\, k \in \mathbb{N}, P(X=k) = (1-q)^{k-1}q$.

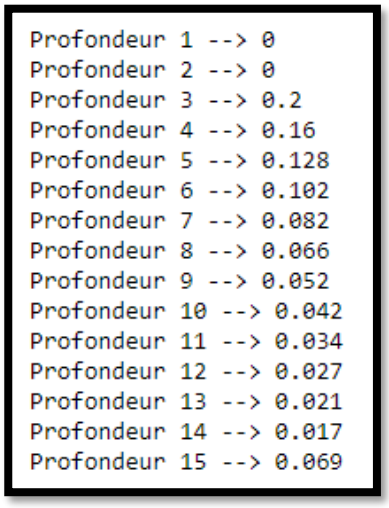
Après calculs, nous obtenons la loi de Y :

$$\forall\, \boldsymbol{k \in [3,15]}, \boldsymbol{P(Y=k) = P(X=k-2) + 1_{\{k=15\}} \sum_{j>15-2} P(X=j)}$$

$$\boldsymbol{P(Y=k) = q(1-q)^{k-3} + 1_{\{k=15\}} \times (1-q)^{15-2}}$$



Poids des profondeurs pour une profondeur effective $P^* = 3$



Poids des profondeurs pour une profondeur effective $P^* = 7$

On interprète alors ces probabilités comme des fréquences de réalisations (loi des grands nombres). Par exemple, nous pouvons observer ci-dessus $\boldsymbol{P}(Y = 3|P^* = 7) = 0{,}20$, cela signifie que, environ 1 fois sur 5, un joueur de profondeur effective égale à 7 jouera son prochain coup en ayant anticipé 3 coups à l'avance. Il est alors naturel de considérer que les évaluations Stockfish de profondeur 3 doivent avoir une influence à hauteur de 20% dans les résultats finaux.

Plus généralement, à P^* fixée, **toutes** les évaluations de Stockfish de profondeur P se voient être pondérées par le poids $\boldsymbol{P}(Y = P|P^*)$.

VI- Résultats finaux

A ce stade, nous avons fini de paramétrer notre modèle. En particulier, on dispose désormais d'une large base de données de joueurs, sur lesquels les analyses de Stockfish ont déjà été calculées. Il s'agit alors de tester notre modèle mathématique sur des données réelles.

Classification binaire

Dans un premier temps, on se propose de réaliser une classification binaire sur la variable Y où $Y = 1$ si le joueur est *bon* $(1300 \leq ELO \leq 2000)$ et $Y = 0$ si le joueur est *faible* $(0 \leq ELO \leq 1000)$ afin de s'assurer que l'on puisse déjà discriminer deux populations dont le niveau de jeu est drastiquement différent.

Ainsi, il est clair que nous devons d'abord être en mesure de réaliser ce type de classifications élémentaires si l'on souhaite être en mesure d'estimer efficacement le niveau d'un joueur sur une échelle plus fine.

Par conséquent, on applique notre modèle de vraisemblance global au cas particulier binaire. Pour ce faire, on attribue une valeur θ_{faible} à la population des joueurs faibles, et une valeur θ_{fort} pour la population des joueurs forts.

Ainsi, la prédiction binaire de notre modèle correspondra à attribuer la valeur $\theta \in \{\theta_{faible}, \theta_{fort}\}$ au joueur selon la population à laquelle il est le plus vraisemblable qu'il appartient. Autrement dit, on calcule $\mathcal{L}(\boldsymbol{\theta_{faible}})$ et $\mathcal{L}(\boldsymbol{\theta_{fort}})$ et $\theta = \theta_{faible}$ si $\mathcal{L}(\boldsymbol{\theta_{faible}}) > \mathcal{L}(\boldsymbol{\theta_{fort}})$, ou bien $\theta = \theta_{fort}$ si $\mathcal{L}(\boldsymbol{\theta_{faible}}) \leq \mathcal{L}(\boldsymbol{\theta_{fort}})$,

Enfin, pour tous les couples $(\boldsymbol{\theta_{faible}}, \boldsymbol{\theta_{fort}}) \in \{\mathbf{1}, \dots, \mathbf{15}\}^{\mathbf{2}}$, on calcule le score[11] de nos prédictions, et représentons les résultats sous forme d'une matrice $\boldsymbol{S}$ **où** $\boldsymbol{S_{i,j}}$ **représente le score des prédictions pour** $(\boldsymbol{\theta_{faible} = i, \theta_{fort} = j})$.

On peut montrer analytiquement que le score est significatif[12] à partir de $\mathbf{57.97\%}$, ainsi le code couleur de la matrice est paramétré en conséquence : une cellule ne se distingue par sa couleur que lorsque le score qu'elle indique est supérieur à cette valeur seuil.

[11] Par score, on entend que l'on compte le nombre de prédictions justes, et l'on divise par le nombre total de prédictions. Le terme anglophone équivalent est *accuracy.* A noter que l'on peut calculer une telle quantité car nous connaissons le niveau réel des joueurs sur lesquels nous effectuons les prédictions.

[12] Test statistique avec un risque de première espèce égal à 5%.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.00	0.51	0.51	0.50	0.50	0.50	0.50	0.50	0.52	0.53	0.57	0.62	0.64	0.72	0.76
2	0.51	0.00	0.51	0.50	0.50	0.50	0.50	0.50	0.52	0.53	0.57	0.62	0.64	0.72	0.76
3	0.49	0.49	0.00	0.49	0.49	0.51	0.54	0.59	0.71	0.78	0.86	0.91	0.91	0.84	0.75
4	0.50	0.50	0.51	0.00	0.55	0.58	0.70	0.85	0.88	0.92	0.88	0.80	0.72	0.64	0.58
5	0.50	0.50	0.51	0.45	0.00	0.72	0.87	0.89	0.90	0.86	0.76	0.68	0.60	0.56	0.53
6	0.50	0.50	0.49	0.42	0.28	0.00	0.89	0.91	0.86	0.75	0.67	0.59	0.54	0.52	0.52
7	0.50	0.50	0.46	0.30	0.13	0.11	0.00	0.84	0.73	0.64	0.59	0.54	0.53	0.52	0.52
8	0.50	0.50	0.41	0.15	0.11	0.09	0.16	0.00	0.64	0.59	0.54	0.53	0.52	0.51	0.51
9	0.48	0.48	0.29	0.12	0.10	0.14	0.27	0.36	0.00	0.54	0.53	0.52	0.51	0.51	0.51
10	0.47	0.47	0.22	0.08	0.14	0.25	0.36	0.41	0.46	0.00	0.53	0.51	0.51	0.51	0.50
11	0.43	0.43	0.14	0.12	0.24	0.33	0.41	0.46	0.47	0.47	0.00	0.51	0.51	0.50	0.50
12	0.38	0.38	0.09	0.20	0.32	0.41	0.46	0.47	0.48	0.49	0.49	0.00	0.50	0.50	0.50
13	0.36	0.36	0.09	0.28	0.40	0.46	0.47	0.48	0.49	0.49	0.49	0.50	0.00	0.50	0.51
14	0.28	0.28	0.16	0.36	0.44	0.48	0.48	0.49	0.49	0.49	0.50	0.50	0.50	0.00	0.51
15	0.24	0.24	0.25	0.42	0.47	0.48	0.48	0.49	0.49	0.50	0.50	0.50	0.49	0.49	0.00

Matrice des scores S selon les couples de valeurs $(\theta_{faible}, \theta_{fort})$

La matrice S concentre les scores les plus élevés dans la partie **triangulaire supérieure.** C'est un constat très rassurant, car cela signifie que la classification performe davantage lorsqu'elle attribue un niveau à la population **forte** supérieur à celui de la population **faible**.

Le score le plus élevé est environ égal à $\mathbf{0,92}$, atteint pour le couple de valeurs :

$$(\theta_{faible}, \theta_{fort}) = (4{,}10)$$

Cela signifie qu'on parvient à discriminer un joueur de la population faible ou forte dans environ 92% des cas pour un tel couple de valeurs. Ainsi, c'est un **excellent** score, atteint avec un jeu de données équilibré (68 joueurs forts contre 70 joueurs faibles). En effet, dans le cas où l'une des deux populations serait très majoritaire (par ex. les joueurs faibles), le modèle pourrait se contenter de prédire systématiquement la valeur 0 et pourrait atteindre un tel score.

En outre, il est rassurant d'observer que localement, autour du couple de valeur optimal $(\theta_{faible}, \theta_{fort}) = (4{,}10)$, les scores demeurent très bons. Alors, dans la pratique, si nous n'avions pas précisément choisi un tel couple (dans la situation où nous ignorions lequel

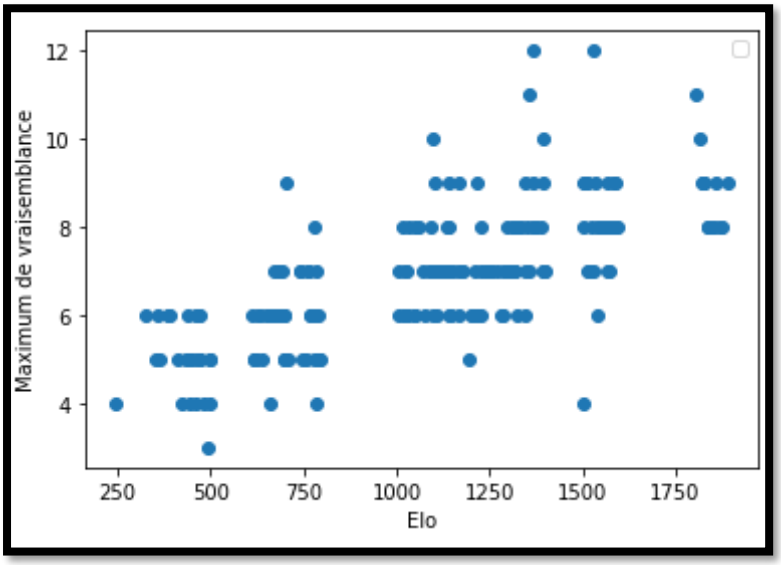
est optimal), nous aurions tout de même eu de bons résultats : le modèle est **robuste** de ce point de vue.

Classification globale

Dans cette section, nous exposons les résultats finaux de notre étude dans le cas général de la classification, c'est-à-dire dans le cas où chaque joueur se voit attribuer une valeur (non binaire) $\theta \in \{1,..,15\}$.

	Max Vraisemblance Moyen	Elo	Nombre de joueurs
0	4.00	(200, 300)	1
1	5.67	(300, 400)	6
2	4.84	(400, 500)	19
3	5.00	(500, 600)	1
4	5.76	(600, 700)	21
5	6.00	(700, 800)	24
6	6.91	(1000, 1100)	22
7	7.17	(1100, 1200)	24
8	6.89	(1200, 1300)	18
9	7.93	(1300, 1400)	29
10	8.00	(1500, 1600)	26
11	8.77	(1800, 1900)	13

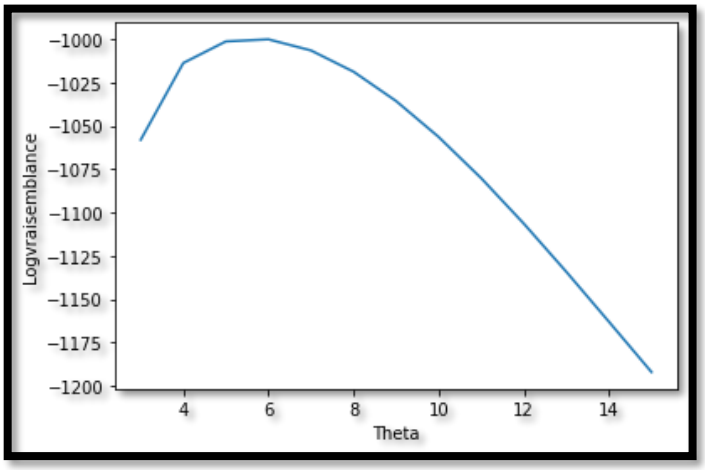
Tableau opposant une tranche d'Elo et le maximum de vraisemblance moyen dans celle-ci



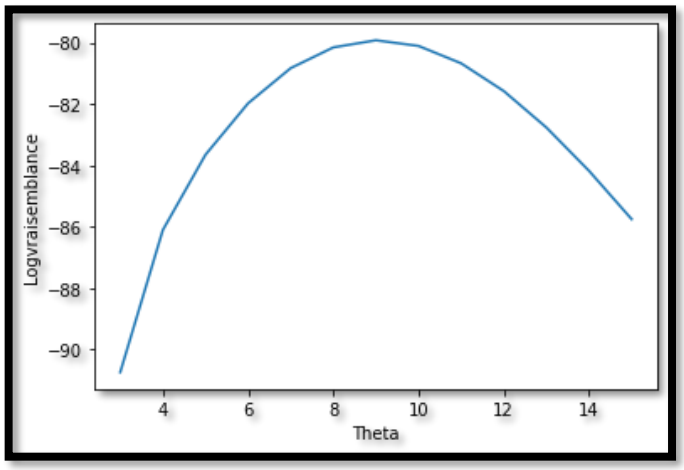
Nuage de points du maximum de vraisemblance confronté au classement Elo

Erreur partiellement supervisée = **88**

Désormais, nous nous réjouissons de pouvoir observer une **nette** tendance croissante dans le nuage de points. Cela signifie que le modèle parvient efficacement à discriminer les joueurs selon le niveau auquel ils jouent. De plus, **l'erreur** est relativement faible, ce qui traduit le fait que les joueurs ont globalement été bien ordonnés.



Log-vraisemblance en fonction de θ pour un joueur dont l'Elo est **626**



Log-vraisemblance en fonction de θ pour un joueur dont l'Elo est **1889**

De plus, l'allure « croissante puis décroissante » des courbes de log-vraisemblance exprime avec succès l'idée qu'il soit de moins en moins vraisemblable qu'un joueur ait un niveau θ à mesure que θ s'éloigne du niveau **réel** du joueur. A noter que l'apparence lisse et régulière des courbes est en fait la conséquence de l'effet « régularisateur » de la pondération des différentes variables dans notre modèle à profondeur aléatoire géométrique.

Validation de notre démarche

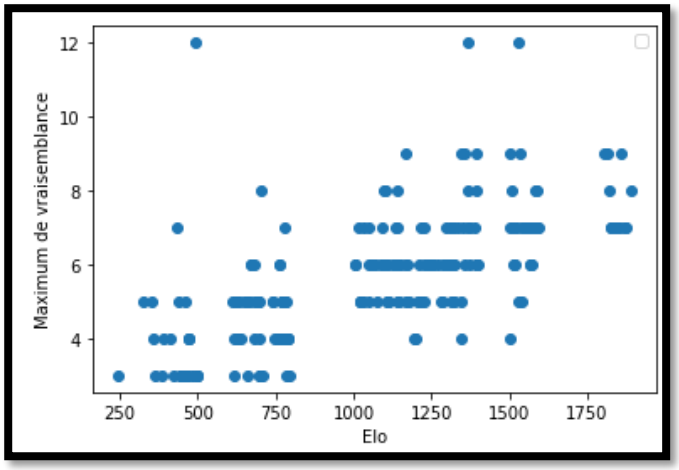
Notre démarche se distingue particulièrement par la mise en œuvre du moteur d'échecs **Stockfish** à des profondeurs de calcul **variables**.

En effet, nous avons émis l'hypothèse que le meilleur moyen pour simuler la façon de jouer d'un joueur d'un certain niveau, n'était pas de considérer une version déjà très forte de Stockfish, que l'on affaiblirait par un bruit. Plutôt, nous considérons qu'il faille mettre en œuvre des versions de Stockfish intrinsèquement plus **faibles** (i.e dont les profondeurs de calcul sont limitées).

Par analogie, admettons que vos recherches vous amènent à vouloir reproduire la course d'un sprinteur amateur, afin d'y étudier scientifiquement la posture, la gestuelle, les foulées, les appuis etc. Ils se présentent alors à vous deux personnes volontaires. Le premier est champion du monde en titre du sprint et vous propose de courir en étant abondamment alourdi par des poids de sorte à générer des performances similaires à celles des amateurs. Enfin, le second est un sprinteur amateur avéré. **Notre postulat est alors celui qui consiste à rejeter le champion du monde et dire que la meilleure façon d'approcher le sprint d'un amateur, est simplement de faire courir un amateur.**

Ainsi, on se propose d'observer les résultats que nous obtiendrions si nous ne tenions compte que de l'avis de **Stockfish** à profondeur maximale[13] afin de valider ou invalider notre postulat.

	Max Vraisemblance Moyen	Elo
0	3.00	(200, 300)
1	4.00	(300, 400)
2	4.11	(400, 500)
3	3.00	(500, 600)
4	4.57	(600, 700)
5	4.58	(700, 800)
6	5.86	(1000, 1100)
7	6.00	(1100, 1200)
8	5.78	(1200, 1300)
9	6.83	(1300, 1400)
10	7.00	(1500, 1600)
11	7.62	(1800, 1900)



[13] Nous remplaçons le vecteur de poids géométrique par un Dirac sur la profondeur 15 : $(0,..,0,1)$

Erreur partiellement supervisée : **95**

Ainsi, l'erreur est légèrement supérieure à celle de notre modèle (95 contre 88), ce qui pourrait suggérer que le modèle à profondeur variable est plus judicieux. Néanmoins, en toute rigueur, il n'est pas certain que cet écart soit significatif. Pour s'en assurer, il faudrait re-optimiser les paramètres pour ce modèle à profondeur maximale, et essayer d'observer que l'erreur demeure toujours supérieure à 88.

Finalement, ce petit « test » ne nous permet pas de conclure quelle est l'approche qui est réellement la plus judicieuse.

VII- Conclusion

L'essence de ce projet **Échecomètre** était de parvenir à évaluer le niveau d'un joueur d'échecs, à partir des **coups** qu'il a joués dans une ou plusieurs parties, sans ne tenir compte d'aucune autre information, là où le classement Elo évalue les joueurs en les faisant se confronter.

Pour mener à bien cette démarche, il nécessitait de se munir d'un moteur d'échecs : **Stockfish** nous est alors apparu comme le choix le plus naturel. Nous manions alors ce moteur d'échecs à des profondeurs de calculs variables, dans l'espoir de mieux approcher le jeu humain, notamment à plus bas niveau. Ainsi, tout l'enjeu de la démarche a été d'adjoindre à Stockfish un modèle probabiliste pertinent.

Les premiers résultats obtenus (cf page 29) pouvaient présager d'un certain potentiel de la méthode, mais demeuraient insuffisants. Par conséquent, un temps conséquent a été alloué à l'amélioration du modèle. Les résultats finaux (cf page 36 et suite) devinrent alors **satisfaisants.**

Naturellement, il subsiste des imperfections dans les résultats (comme peut en témoigner le nuage de points page 39). Ces dernières étaient inéluctables et peuvent trouver leurs origines à travers trois principales raisons :

- Le modèle pourrait encore gagner à être amélioré et paramétré différemment[14], de sorte à accroitre son pouvoir discriminateur. Il existe encore un très grand nombre de pistes qui peuvent qui peuvent être creusées pour arriver à ces fins. A défaut d'explorer toutes les possibilités, nous nous contentons de mettre en lumière le **potentiel** qu'il réside au sein de notre démarche.
- Nous attestons de la qualité de nos prédictions à travers notre connaissance du classement Elo des joueurs. Néanmoins, rien n'empêche dans la pratique que les joueurs performent sur une ou plusieurs parties à un niveau bien inférieur ou supérieur à celui que viendrait décrire leur classement. Autrement dit, il existerait là comme un **bruit inhérent aux données**, dont il faut simplement avoir conscience.
- Enfin, le cœur de notre démarche a visé à « expliquer » le niveau des joueurs **à travers** la vision du moteur d'échecs Stockfish. Or, il se pourrait qu'il soit loin d'être

[14] Comme tous les modèles, car *tous les modèles sont faux, mais certains sont utiles* !

optimal de procéder ainsi, tant Stockfish aurait un jeu trop différent du joueur humain **moyen**[15].

En définitive, nous nous réjouissons d'avoir tout de même abouti à un modèle **satisfaisant**, en moyenne capable de discriminer les joueurs selon leur niveau, à partir des **coups joués uniquement** et des évaluations de Stockfish de profondeurs **1** à **15**.

[15] Les joueurs professionnels ont au contraire un jeu très semblable à celui de Stockfish, dans la mesure où ces derniers se préparent et s'entrainent avec Stockfish et d'autres outils similaires ! Néanmoins, pour le joueur moyen, c'est une autre histoire...

VIII - Bibliographie

[1] **Optimisation bayésienne à la rescousse** par **Vincent Gatien,** vu le 29/01/2022 sur https://medium.com/videns-analytics/optimisation-bay%C3%A9sienne-%C3%A0-la-rescousse-b9bd2db32ff6

2] **Documentation de l'API de la plateforme chess.com,** vu le 29/01/2022 sur https://www.chess.com/news/view/published-data-api