Saphire

211 - Identification paramétrique et optimisation

-

Optimisation paramétrique appliquée à la prédiction de l'issue de matchs de rugby

PIERRE-ALEXANDRE PEYRONNET

1ER MAI 2021

école normale supérieure paris–saclay

université PARIS-SACLAY

Table des matières

Introduction

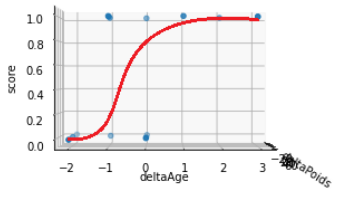
Peut-on prédire le résultat d'un match de rugby avant même qu'il n'ait commencé, seulement par la connaissance de statistiques disponibles avant la rencontre ? Voila une question dont la réponse pourrait faire le bonheur des parieurs, ou bien accentuer le malheur des supporters des équipes les moins en forme. Ainsi, j'ai voulu déterminer au cours de mon projet s'il est possible de faire ce genre de prédictions, à l'aide de relativement peu d'échantillons et de variables d'entrée, et en en déduisant un modèle de comportement. Pour ce faire, j'ai emprunté des données au site internet *allrugby.com*, et j'ai fais mes essais à l'aide du langage de programmation python.

1 Modélisation

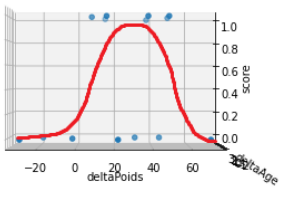
1.1 Définition du système, choix des données

Avant de s'enfoncer dans la modélisation du système, il convient de le définir. Le système étudié est un match de rugby qui, dans le cadre du projet, prend deux entrées et une sortie. Les entrées sont la différence de poids Δm et la différence de l'âge moyen Δa des packs [1] des deux équipes, ces grandeurs sont réelles. La sortie est le score s (plus précisément, l'issue du match), cette grandeur est donc binaire en supposant que le match nul n'existe pas au rugby. Dans la suite, une équipe de référence parmi les deux est choisie de sorte que ce soit celle à qui on soustrait le poids et l'âge de l'autre équipe. Ainsi, si cette équipe de référence gagne, le score du match sera égal à 1, sinon il sera nul.

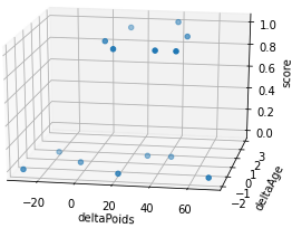
Comme évoqué en introduction, je me suis donc rendu sur le site internet *all-rugby.com* afin de récupérer des échantillons de match. Afin que ces échantillons, une fois placés sur un graphique, suivent une certaine tendance, j'ai décidé de prendre les statistiques des quinze premiers matchs qu'avait disputé le RCT [2] au 15 février 2021, pour la saison 2020/2021 du Top 14. Cela peut sembler très insuffisant et peu varié étant donné qu'on a peu matchs et de la même équipe, mais l'avantage de ce choix est qu'il rend l'imagination du modèle plus simple, la tendance est plus visible qu'avec un graphique noircit par les échantillons. Une fois les données récupérées, je les ai donc affiché sur un graphique via la librairie *matplotlib.pyplot* du langage python, et j'en ai déduit une tendance :



(a) Projection sur l'axe de la différence d'âge



(b) Projection sur l'axe de la différence de poids



(c) Affichage des échantillons

FIGURE 1.1

1. Le pack désigne les avants (joueurs 1 à 8 inclus).
2. Rugby Club Toulonnais

1.2 Exploitation des données, création du modèle

Bien que les valeurs prisent par la sortie du système soient totalement binaire, on peut imaginer qu'elles suivent une certaine tendance, comme présenté en [1.1a] et [1.1b]. Ainsi, en supposant que chacun des matchs sont indépendants entre eux, on peut dire que le score suit une loi de Bernoulli dont la probabilité dépend des entrées, une modélisation probabiliste est donc à privilégier. Sur la projection sur l'axe Δa, les échantillons semblent suivre la tendance d'une arctangente, décalé de sorte à être comprise entre 0 et 1, et centré pour sur une différence d'âge environ égale à -0.5. Sur la projection sur l'axe Δm, on voit plutôt la forme d'une gaussienne centré à 30 kg. J'ai donc choisi de modéliser la probabilité de gagner un match par la structure suivante :

$$q(\Delta a, \Delta m, \mathbf{p}) = \left(\frac{\arctan(p_0(\Delta a - p_1))}{\pi} + \frac{1}{2}\right) e^{-(p_2(\Delta m - p_3))^2}$$

Cette structure fait intervenir quatre paramètres, aillant chacun leur utilité dans la forme de la surface de probabilité obtenu :

- p_0 : Paramètre de dilatation de l'arctangente ;
- p_1 : Paramètre de décalage de l'arctangente ;
- p_2 : Paramètre de dilatation de la gaussienne ;
- p_3 : Paramètre de décalage de la gaussienne (différence de poids optimale) ;

Concernant l'identifiabilité structurelle du modèle, bien cette étape ne sois pas nécessaire compte tenu du fait que seule la sortie m'intéresse, on peut aisément dire que ce modèle est S.L.I. [3]. En effet, pour un second jeu de paramètres $\widehat{\mathbf{p}}$, on a des modèles égaux si :

$$\begin{cases} p_0 &= \widehat{p}_0 \\ p_1 &= \widehat{p}_1 \\ p_2^2 &= \widehat{p}_2^2 \\ p_3 &= \widehat{p}_3 \end{cases}$$

Or, ce système possède deux solutions. Si l'on ne voulait pas que le modèle puisse être le même malgré des paramètres différents, il aurait fallut borner p_2 à $\mathbb{R}_+$ ou $\mathbb{R}_-$.

L'intervention de l'arctangente et de l'exponentielle dans la structure fait qu'elle est non-linéaire en les entrées et en les paramètres, ainsi, les méthodes d'estimations ne sont pas envisageables. J'ai donc choisi, pour déterminer les meilleurs paramètres, de faire une optimisation par la méthode de la descente de gradient.

Ceci fait, pour simuler le score d'un match, il suffit de réaliser une épreuve de Bernoulli avec la probabilité obtenue. On peut donc schématiser le modèle de la manière suivante :

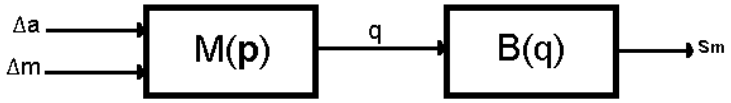


FIGURE 1.2 – Schéma du modèle

3. Structurellement localement identifiable

2 | Optimisation

2.1 Définition du critère, recherche des paramètres initiaux

Étant donné que le modèle est probabiliste et qu'il suit une loi de Bernoulli, le critère à optimiser est la vraisemblance de la loi de Bernoulli, avec une légère modification. En effet, la vraisemblance de la loi de Bernoulli est la fonction suivante :

$$\pi(\mathbf{y}|\mathbf{p}) = q^{\sum_{i=0}^{N-1} y_i} \cdot (1-q)^{N-\sum_{i=0}^{N-1} y_i}$$

Cette fonction n'est pas adapté à notre problème où la probabilité n'est pas constante. Je l'ai donc modifié afin d'y inclure la dépendance aux entrées :

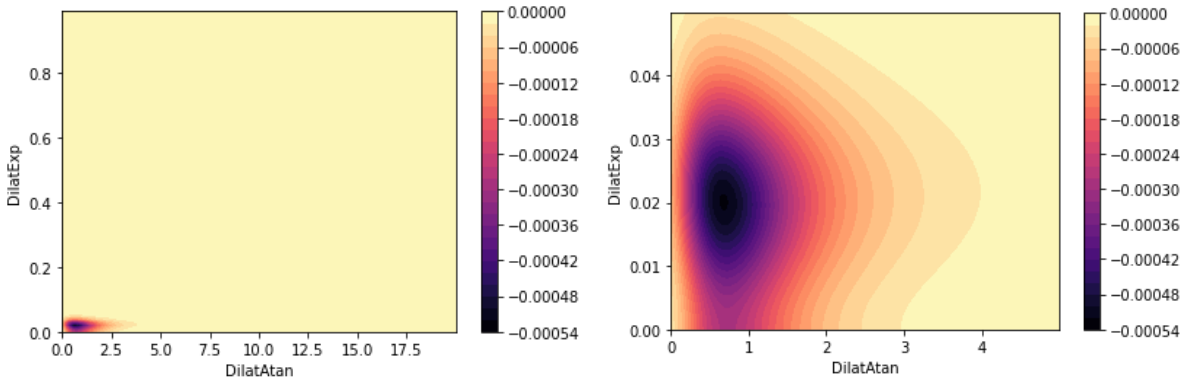
$$\pi(\mathbf{s}|\mathbf{p}) = \prod_{i=0}^{N-1} q(\mathbf{u}_i, \mathbf{p})^{s_i} \cdot \prod_{i=0}^{N-1} (1 - q(\mathbf{u}_i, \mathbf{p}))^{1-s_i}$$

Le but étant de maximiser le critère avec un algorithme itératif, je n'ai pas utilisé le logarithme de la vraisemblance. De plus, la vraisemblance peut atteindre des valeurs très proche de zéro, ce qui n'est pas compatible avec le logarithme. Enfin, afin de maximiser ce critère par une méthode de descente, j'ai choisi de minimiser l'opposé de $\pi(\mathbf{s}|\mathbf{p})$, afin de ne pas créer de conflits avec les algorithmes d'optimisation du pas.

Ceci fait, j'ai voulu obtenir des paramètres initiaux pertinents, afin que l'algorithme ne s'arrête pas dès la première itération sur un point où la vraisemblance est nulle. Néanmoins, la dimension du vecteur des paramètres est égale à 4, on ne peut donc pas afficher un graphique du critère en fonction de tout les paramètres. Ainsi, j'ai fait des hypothèses sur les paramètres idéaux :

- p_0 est forcement positif afin que l'arctangente soit dans le bon sens ;
- $p_1 \approx -0.5$ (cf. [1.1a]) ;
- Le critère est pair en p_2 et la gaussienne est large, on observe donc p_2 sur $\mathbb{R}_+$ et on le suppose proche de 0 ;
- $p_3 \approx 30$ (cf. [1.1b]) ;

J'ai donc commencé par observé le critère en fixant p_1 et p_3, et pour $p_0 \in [0,20]$ et $p_2 \in [0,1]$, j'ai ensuite affiné l'échelle de manière à mieux observer les variations du critère (cf. [2.1b]). À partir de ces graphique, j'ai alors choisi de façon arbitraire que le vecteur de paramètres initiaux serait $\mathbf{p}_0 = (1, -0.5, 0.02, 30)^T$.

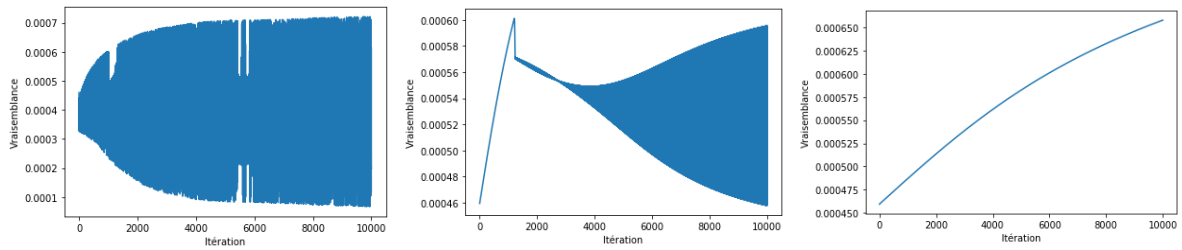


(a) $(p_0, p_2) \in [0,20] \times [0,1]$ (b) $(p_0, p_2) \in [0,5] \times [0,0.05]$

FIGURE 2.1 – Affichage du critère pour $(p_1, p_3) = (-0.5, 30)$

2.2 Optimisation du critère par descente de gradient à pas constant

Après avoir défini mon critère et mes paramètres initiaux, j'ai enfin pu passer à l'optimisation par descente de gradient. Lors de l'écriture de l'algorithme, j'ai décidé de retenir trois critères d'arrêts : un nombre maximale d'itérations, une norme minimale du gradient et une variation relative minimale du critère. Tout au long de mes tests, ces critères avaient respectivement les valeurs 10^4, 10^{-6} et 10^{-8}. J'ai d'abord testé cet algorithme pour des pas constants, afin de vérifier sa convergence, et son effet sur le critère. Ainsi, j'ai retenu trois pas : 1, 0.5 et 0.1. En règle général, l'algorithme s'arrêtait dès la première itération pour des pas supérieurs ou inférieurs à ces valeurs. J'ai alors pu observer les variations du critère. Sur ces tracés, on remarque deux choses : le critère



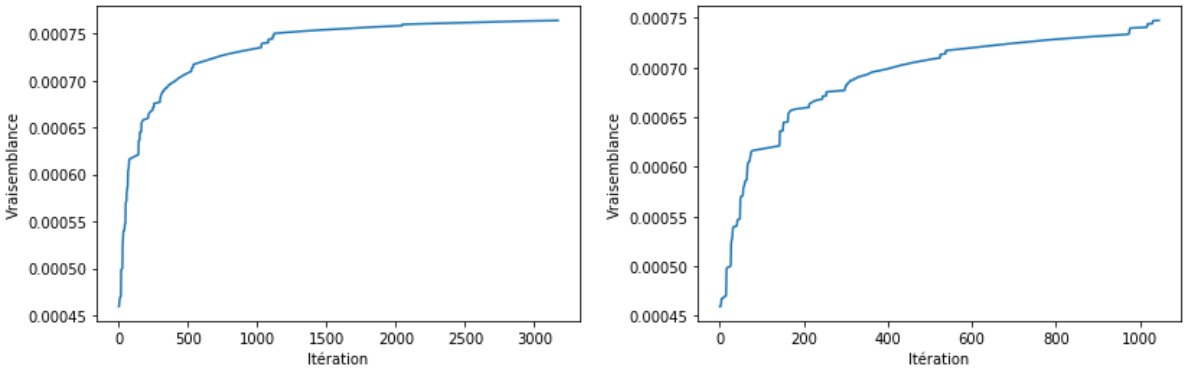
(a) Pas : 1 (b) Pas : 0.5 (c) Pas : 0.1

FIGURE 2.2 – Tracés du critère au cours des itérations

se met à osciller pour un pas trop élevé et le critère d'arrêt est le nombre d'itérations. Le premier point empêche de manière quasi-certaine d'obtenir des paramètres optimaux, mais le deuxième est plus sournois dans le sens où le critère peut suivre une variation constante, mais on ne sais pas si les paramètres obtenus sont réellement les meilleurs.

2.3 Optimisation du pas par l'algorithme de Backtracking-Armijo

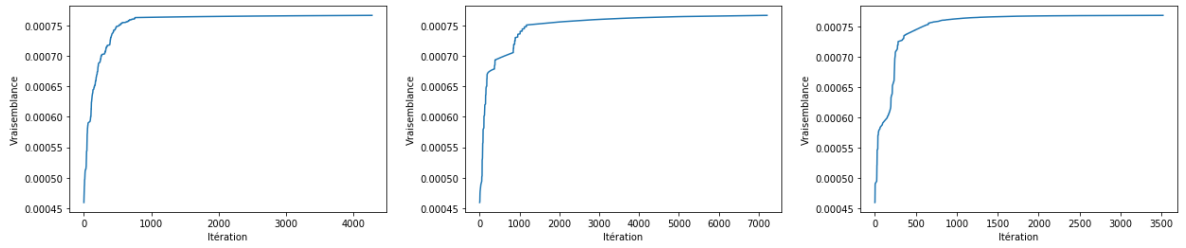
Afin d'accélérer la convergence du critère et surtout pour supprimer les oscillations, j'ai ajouté une optimisation du pas à mon algorithme de descente. Pour cela, j'ai utilisé l'algorithme de *Backtracking-Armijo* qui se sert de la condition d'Armijo pour obtenir un pas suffisamment petit afin d'avoir une variation de critère monotone. Cet algorithme fait intervenir deux paramètres : un coefficient τ associé à la réduction du pas et un coefficient β qui impose la décroissance du critère. Afin d'avoir la possibilité d'avoir une forte décroissance, j'ai choisie un pas initial de 100, puis j'ai testé différentes valeurs de τ et β afin de voir leur influence. D'abord, j'ai fixé τ à 0.5, puis j'ai testé quelque valeurs de β. Ce qu'on remarque en manipulant β, c'est que sa décroissance accélère la satisfaction



(a) β = 0.001 (b) β = 0.00001

FIGURE 2.3 – Tracés du critère au cours des itérations, $\tau = 0.5$

des critères d'arrêts. On a donc un algorithme qui s'arrête plus rapidement, mais au prix d'une vraisemblance légèrement diminué. J'ai ensuite essayé de fixer $\beta = 0.001$ pour faire varier τ. Le comportement de l'algorithme pour des variations de τ est un peu plus



(a) $\tau = 0.875$ (b) $\tau = 0.25$ (c) $\tau = 0.125$

FIGURE 2.4 – Tracés du critère au cours des itérations, $\beta = 0.001$

aléatoire. Néanmoins, on peut dire que l'algorithme sera, ici, plus rapide pour $\tau = 0.5$. Il est aussi important de noter que, étant donné que le pas de départ est élevé, un τ élevé implique que l'exécution de l'algorithme d'optimisation du pas sera beaucoup plus longue. Enfin, j'ai observé les variation du pas (cf. [2.5]). Ceci nous permet de voir qu'un grand pas était utile pour avoir une forte augmentation de la vraisemblance dans les premières itérations. En effet, en comparant cette figure avec [2.3a], on observe bien une marche à chaque fois que les pas est égal à 100.

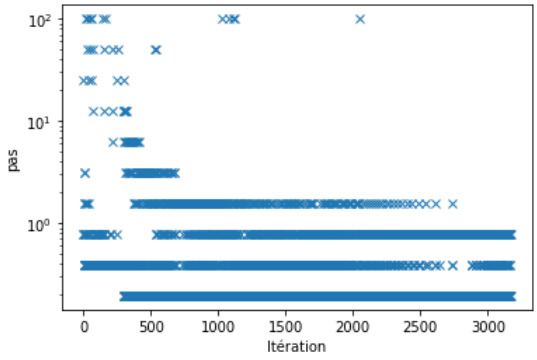


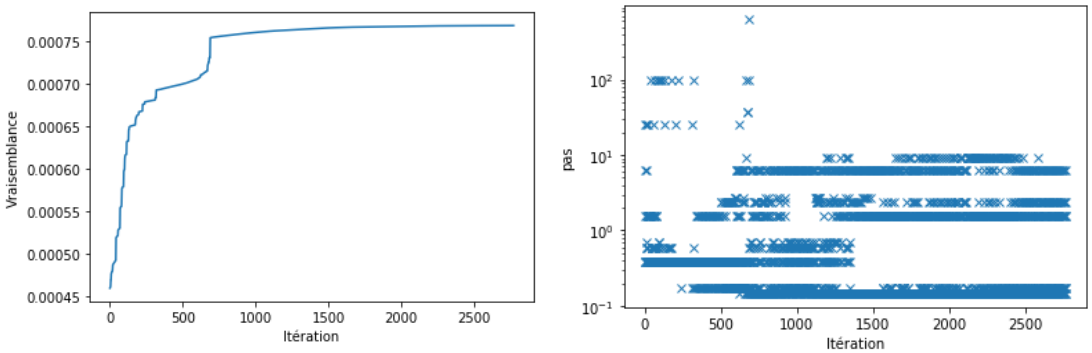
FIGURE 2.5 – Affichage du pas au cours des itérations, $\beta = 0.001$, $\tau = 0.5$

2.4 Optimisation du pas par la condition de Goldstein-Armijo

Pour finir avec l'optimisation, j'ai voulu essayé d'améliorer l'optimisation du pas en utilisant la condition de Goldstein-Armijo :

$$\begin{cases} f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) & \leq & f(\mathbf{x}_k) + \alpha_k \beta_1 \mathbf{d}_k^T \mathbf{g}_k \\ f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) & \geq & f(\mathbf{x}_k) + \alpha_k \beta_2 \mathbf{d}_k^T \mathbf{g}_k \\ 0 < \beta_1 & < & \beta_2 < 1 \end{cases}$$

Ici, la deuxième ligne est la condition de Goldstein. Pour trouver un pas avec ces conditions, il faut d'abord chercher un pas minimale qui respecte la condition d'Armijo (en utilisant la méthode de la section précédente), un pas maximale en doublant le pas minimale jusqu'à ce qu'il respecte la condition de Goldstein, puis on peut rechercher par dichotomie un pas qui respecte les deux conditions. Cette nouvelle condition, couplé à la condition d'Armijo, permet en théorie d'avoir un pas suffisamment petit, mais aussi suffisamment grand. On pourrait donc encore accélérer la convergence de l'algorithme. Néanmoins, dans mon cas, j'ai rencontré beaucoup de difficultés à trouver des coefficients qui améliore la convergence du critère. Finalement, on peut observer une accélération de l'algorithme pour $\beta_1 = 0.001$, $\beta_2 = 0.6$ et $\tau = 0.25$. On observe donc ici un gain au ni-



(a) Vraisemblance

(b) Pas

FIGURE 2.6 – Évolution de l'algorithme au cours des itérations

veau du nombre d'itérations et une vraisemblance finale un peu plus forte. Pourtant, la difficulté de la recherche des coefficients ne me semble pas justifié un gain aussi faible.

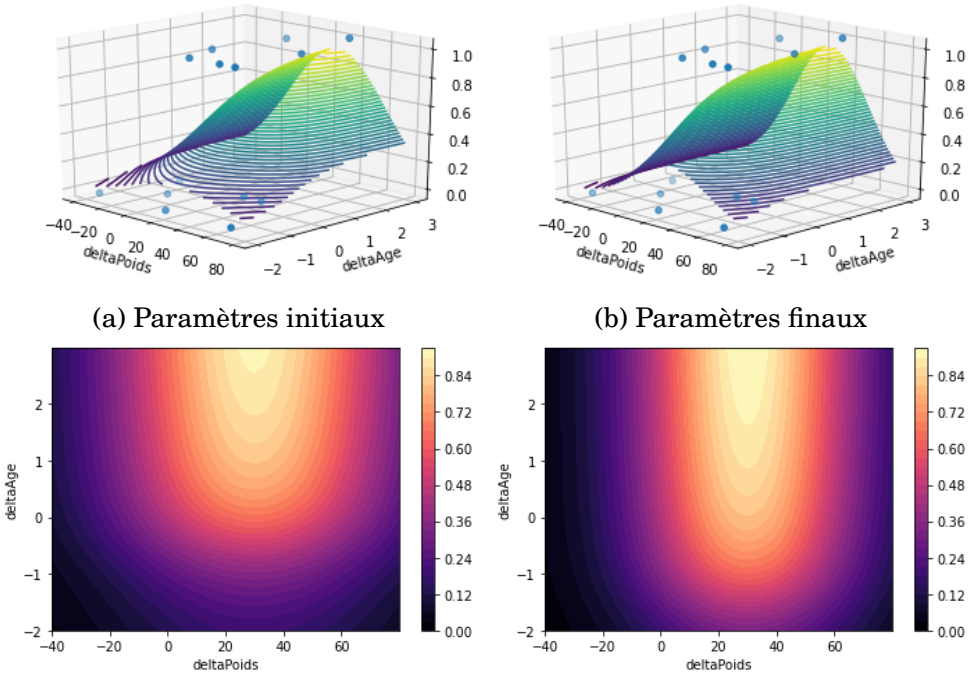
3 Simulation

3.1 Résultat de l'optimisation

Pour la suite, je n'utiliserai que le résultat que j'ai obtenu avec la méthode de Goldstein, avec les paramètres évoqués précédemment. Une fois la descente de gradient terminé, j'ai obtenu les paramètres suivant :

$$\mathbf{p} = (1.00548519, -1.41589987, -2.56206981 \cdot 10^{-2}, 30.0175696)^T$$

Ainsi, on peut voir que parmi les hypothèses que j'avais fait sur les paramètres, c'est celle sur le décalage de l'arctangente qui était la "moins vraie". Ensuite, j'ai affiché la surface de probabilité obtenu, en fonction des paramètres initiaux et finaux, afin de les comparer aux échantillons. On peut voir, en comparant ces deux images, que le modèle final prends



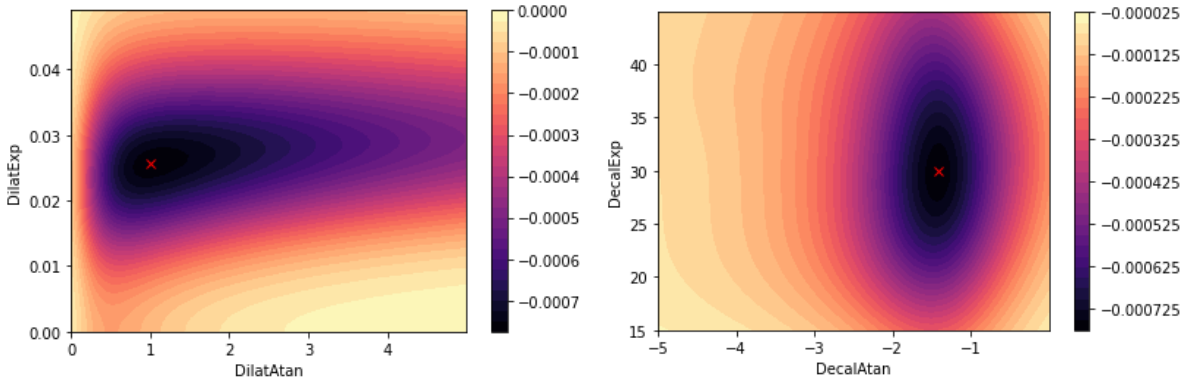
(a) Paramètres initiaux (b) Paramètres finaux

(c) Paramètres initiaux (d) Paramètres finaux

FIGURE 3.1 – Surface de probabilité

mieux en compte les victoire à $\Delta a = -1$, mais aussi que la gaussienne est moins large, ce qui conforte l'idée qu'il existe une différence de poids idéale.

Enfin, on peut de nouveau observer le critère afin de voir ce qu'a apporté l'optimisation (cf. [3.2a] et [3.2b]). On peut voir que la forme du critère, pour des décalages



(a) Décalage fixe, $(p_1, p_3) = \text{cte}$

(b) Dilatation fixe, $(p_0, p_2) = \text{cte}$

FIGURE 3.2 – Affichage du critère pour les paramètres finaux

constants, a été légèrement modifiée, mais pas assez pour que les hypothèses de bases soient invalidées. Pour des dilatations constantes, on peut observer que les paramètres obtenus se situent pile sur le minimum du critère, on peut donc en conclure que les paramètres obtenus sont les meilleurs, pour cette structure de modèle.

3.2 Application

Pour finir, j'ai voulu vérifier si ce modèle pouvait effectivement prédire l'issue d'un match, ou au minimum donner une probabilité cohérente avec l'issue. Pour cela, j'ai décidé de procéder de deux façons. Premièrement, j'ai utilisé le modèle pour prédire les issues de neuf matchs de trois équipes différentes. Ensuite, j'ai fais une deuxième optimisation, en enlevant la moitié des échantillons, afin d'essayer de retrouver leur issue. Pour obtenir l'issue d'un match, j'ai simplement utilisé la fonction « *random.binomial* » de la bibliothèque NumPy, à laquelle j'ai appliqué la probabilité obtenu suivant les statistiques de chaque match. Pour la première partie, j'ai rangé les valeurs dans le tableau [3.1], dont les équipes en gras sont celles prises comme référence. On peut voir que le

Match	**ST** - ASM	**ST** - SF	**ST** - UBB	**CO** - SR	**CO** - CAB
Probabilité	**0.74**	0.16	**0.23**	0.18	0.27
Issue modélisé	0	0	0	0	0
Issue réelle	0	0	1	0	0

CO - R92	**SUA** - AB	**SUA** - MHR	**SUA** - SP
0.74	0.56	0.11	**0.87**
1	0	0	1
1	0	0	0

TABLEAU 3.1 – Probabilités et issues associées à différent matchs du Top 14 2020/2021

modèle nous donne quelque probabilités peu cohérentes avec les issues, bien que le modèle semble pouvoir prédire deux tiers des matchs testés. Finalement, si on enlève les

7 derniers échantillons avant l'optimisation, le modèle obtenu est modifié au point qu'il n'utilise plus l'effet de la gaussienne. J'ai alors pu relever les résultats de la prédiction dans le tableau 3.2. On peut voir dans ce tableau que les probabilités ne sont pas tota-

Journée 9	J10	J11	J12	J13	J15	J16
0.43	0.59	0.59	0.71	0.30	0.43	0.30
0	1	1	0	0	0	0
1	1	0	1	0	0	0

TABLEAU 3.2 – Probabilités et issues associées aux matchs de Toulon du Top 14 2020/2021

lement incohérentes avec les issues réelles, néanmoins les issues simulées sont quant à elle un peu moins en phase avec la réalité.

Conclusion

Au travers de ce projet, j'ai pu voir qu'il était possible de modéliser de façon probabiliste un phénomène qui n'est a priori pas lié au hasard. Néanmoins, cette modélisation a ses limites. En effet, mon modèle ne renvoie pas assez souvent des valeurs correspondante à la réalité et cela est dû à plusieurs facteur. Premièrement, le nombre d'échantillons choisie est extrêmement faible et peu varié, il ne prend donc en compte que peu de résultats des efforts et d'une façon de jouer d'une seule équipe. Ensuite, le modèle manque cruellement de paramètres d'entrée, le rugby ne se joue pas que sur des paramètres aussi abstrait que la différence d'âge et de poids. Il faudrait prendre en compte la stratégie adopté, l'état d'esprit des joueurs et pourquoi pas la météo ou l'heure de la rencontre. Enfin, cette modélisation est incohérente dans le sens où elle dépend du référentiel. Ainsi, si on cherche la probabilité q qu'une équipe de référence A gagne face à une équipe B, la probabilité que l'équipe de référence B perde sera très probablement différente de $1-q$. Finalement, le rugby est un jeu où le hasard n'a que peu de place, et le meilleur moyen de connaître l'issue d'un match est de disposer d'une DeLorean et d'un almanach des sports.

Bibliographie

- *Allrugby* [en ligne], 2021 [consulté le 30 avril 2021]. Disponible sur https://www.allrugby.com/
- Frouvelle, Amic. *Méthodes numériques : optimisation*. 2015. p. 21 et 22.
- *Statify – Statistical methods for dealing with complex ... - Inria* [en ligne], Inria Alpes [consulté le 30 avril 2021]. Pratique du maximum de vraisemblance. Dispobible sur https://mistis.inrialpes.fr/software/SMEL/cours/ep/node12.html