# LE PERCEPTRON

NILS HOLZENBERGER

Les neurones artificiels sont fondés sur une analogie avec les neurones biologiques, dans l'espoir qu'ils possèdent des propriétés analogues : par exemple, la faculté de reproduire l'intelligence humaine. Le perceptron est un algorithme permettant au neurone artificiel d'apprendre. Cet algorithme est décrit pour la première fois par Frank Rosenblatt avec la publication de *Principles of Neurodynamics* en 1962. Sept ans plus tard, Minsky et Papert démontrent dans *Perceptrons* un théorème fondamental dans l'apprentissage par les réseaux neuronaux : le perceptron peut apprendre à faire tout ce pour quoi il peut être programmé. Nous commencerons par définir mathématiquement le neurone artificiel, puis nous introduirons les notations pertinentes avant de présenter le perceptron. Nous chercherons ensuite à améliorer cet algorithme d'apprentissage afin, entre autres, de permettre au neurone artificiel de résoudre de nouveaux problèmes, et aboutirons sur une comparaison de deux algorithmes.

Les énoncés sont numérotés dans leur ordre d'apparition. La preuve correspondante se trouve au même numéro en annexe. Les ouvrages cités se trouvent par ordre alphabétique des auteurs en annexe.

## 1. Position du problème

On cherche une machine qui peut répondre oui ou non à une question à partir de plusieurs données numériques. Par analogie avec les synapses du neurone biologique, la machine dispose de $N \in \mathbb{N}^*$ entrées. Chaque signal (numérique) est multiplié par le poids de l'entrée correspondante, puis ces réels sont sommés. Si la somme est supérieure à un certain seuil, la réponse est oui (ce qui correspond à la sortie $s = 1$), sinon, non (sortie $s = -1$).

Avec les notations de la figure

$$s(x_1, ..., x_N) = \begin{cases} 1 & \text{si} \quad \sum_{i=1}^{N} w_i x_i > \theta \\ -1 & \text{sinon} \end{cases}$$

$x_i$ est le signal que reçoit l'entrée $i$, $w_i$ est le poids de l'entrée $i$, $\theta$ est le seuil

Considérons par exemple le ou logique, pour lequel les poids $w_1 = w_2 = 1$ et le seuil $\theta = -1$ conviennent :

| $A$ | $B$ | $A \vee B$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −1 | 1 |
| −1 | 1 | 1 |
| −1 | −1 | −1 |

| $x_1$ | $x_2$ | $w_1x_1 + w_2x_2$ | $s$ |
|---|---|---|---|
| 1 | 1 | 2 | 1 |
| 1 | −1 | 0 | 1 |
| −1 | 1 | 0 | 1 |
| −1 | −1 | −2 | −1 |

Tout le problème est de trouver, pour des motifs donnés, des poids et un seuil qui permettent au neurone de renvoyer la réponse attendue. Avant de passer à l'algorithme de Rosenblatt, qui permet de trouver de tels poids, il est utile de simplifier le problème en introduisant des notations.

## 2. Problème matriciel

On pose $w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}$ le vecteur des poids, et pour un motif $(x_1, x_2, \cdots, x_N)$, $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$. Alors $s(x_1, ..., x_N) =$

$\begin{cases} 1 & \text{si} \quad w \cdot x > \theta \\ -1 & \text{sinon} \end{cases}$ où $\cdot$ note le produit scalaire canonique sur $\mathbb{R}^N$

En ajoutant à chaque motif $(x_1, ..., x_N)$ une $N+1^{\text{e}}$ entrée valant 1, et en posant $w_{N+1} = -\theta$ :

$s(x_1, ..., x_N) = \begin{cases} 1 & \text{si} \quad w \cdot x > 0 \\ -1 & \text{sinon} \end{cases}$

On supposera donc sans restriction que le seuil vaut 0, en ajoutant à chaque motif une entrée supplémentaire.

On dispose à présent de $p \in \mathbb{N}^*$ motifs, et pour chaque motif d'une réponse attendue. Le but est de trouver des poids tels que le perceptron renverra toujours la réponse attendue. On identifie les $p$ motifs à des éléments de $\mathbb{R}^N$ ou encore de $\mathcal{M}_{N,1}(\mathbb{R})$ (ce qui permet de noter le produit scalaire $a \cdot b = {}^t ab$). On note les motifs $\{(x_1, y_1), (x_2, y_2), \cdots, (x_p, y_p)\}$ où $x_i \in \mathbb{R}^N$ et $y_i = \pm 1$ est la sortie attendue pour $x_i$

Alors le vecteur $w$ convient si et seulement si $\forall i \in [\![1, p]\!]$, $\begin{cases} w \cdot x_i > 0 & \text{si} \quad y_i = 1 \\ w \cdot x_i \leq 0 & \text{si} \quad y_i = -1 \end{cases}$

On peut réecrire cette condition en $\forall i \in [\![1, p]\!]$, $\begin{cases} w \cdot x_i > 0 & \text{si} \quad y_i = 1 \\ w \cdot x_i < 0 & \text{si} \quad y_i = -1 \end{cases}$ (1)

En effet, on obtient les inégalités strictes à partir des inégalités larges en diminuant légèrement $w_N$.

Donc le vecteur $w$ convient si et seulement si $\forall i \in [\![1, p]\!], y_i w \cdot x_i > 0$

Posons $T = \begin{pmatrix} y_1 x_1 & y_2 x_2 & ... & y_p x_p \end{pmatrix} \in \mathcal{M}_{N,p}$ ie $T = (y_j x_{j,i})_{i \in [\![1,N]\!], j \in [\![1,p]\!]}$

On a ${}^t Tw = \begin{pmatrix} y_1 x_{1,1} & y_1 x_{1,2} & \cdots & y_1 x_{1,N} \\ y_2 x_{2,1} & y_2 x_{2,2} & \cdots & y_2 x_{2,N} \\ \vdots & & & \\ y_p x_{p,1} & y_p x_{p,2} & \cdots & y_p x_{p,N} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} = \begin{pmatrix} y_1 x_1 \cdot w \\ y_2 x_2 \cdot w \\ \vdots \\ y_p x_p \cdot w \end{pmatrix}$

En conséquence (1) $\Leftrightarrow \forall i \in [\![1, p]\!], [\,{}^t Tw]_{i,1} > 0$, ce que l'on résumera en ${}^t Tw > 0$ ou encore en $\min\limits_{i \in [\![1,p]\!]} {}^t c_i w > 0$ où $c_i$ désigne la $i^{\text{e}}$ colonne de $T$

Par exemple pour le ou logique, $T = \begin{pmatrix} 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$

Par la suite on appelera solution un élément $z \in \mathcal{M}_{N,1}$ tel que ${}^t Tz > 0$. Pour simplifier, on supposera $\forall i, ||c_i|| = 1$ où $||.||$ désigne la norme euclidienne sur $\mathbb{R}^N$, le problème en étant inchangé.

## 3. Algorithme perceptron

Le perceptron sera abusivement identifié à la suite $w \in (\mathcal{M}_{N,1}(\mathbb{R}))^{\mathbb{N}}$ :

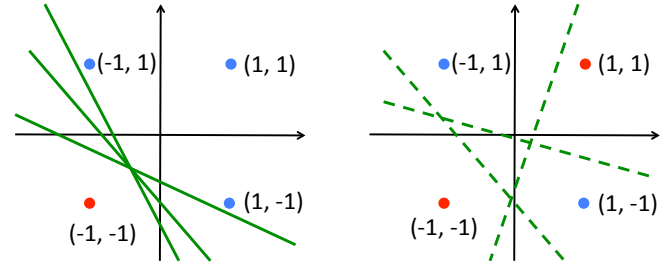
- $w_0 = 0$
- Si ${}^t Tw_k > 0$ l'algorithme a terminé. $w_k$ est solution.

Sinon, soit $i \in [\![1, p]\!]$ tel que ${}^t w_k c_i \leq 0$ ; alors $w_{k+1} = w_k + c_i$

Les poids ne sont donc modifiés que si le perceptron s'est trompé.

*Théorème 1 (Minsky et Papert).* Le perceptron converge en un nombre fini d'étapes vers une solution s'il existe $u \in \mathbb{R}^N$ tel que ${}^t Tu > 0$.

En somme, le perceptron converge s'il existe des poids convenables. Ou encore, le perceptron converge s'il existe un hyperplan séparant les motifs (vus comme des points d'un espace vectoriel de dimension $N$) tels que $y_i = -1$ des motifs tels que $y_i = 1$. Il existe en effet des cas pour lesquels il n'existe pas de poids convenables car pas d'hyperplan séparateur, comme par exemple le ou exclusif (XOR).




OU : existence d'hyperplans séparateurs

XOR : pas d'hyperplan séparateur

Figure 2. Séparabilité par un hyperplan. En bleu (resp. rouge) les motifs de la classe 1 (resp. -1). En vert les hyperplans.

Il serait donc utile de pouvoir estimer a priori si le perceptron converge, ou le nombre maximum d'itérations nécessaires avant de pouvoir conclure quant à l'existence d'une solution.

## 4. Théorème de Gordan

Le théorème de Gordan permet de caractériser l'existence d'une solution.

*Théorème 2 (de Gordan).* Soit $A \in \mathcal{M}_{n,p}$, alors
un et un seul des deux systèmes a une solution :
(i) ${}^tAx > 0$ ; $x \in \mathcal{M}_{n,1}$
(ii) $x \geq 0$ et $x \neq 0$ et $x \in \mathrm{Ker}A$ ; $x \in \mathcal{M}_{p,1}$

Il suffit pour savoir s'il existe une solution pour $T$ de résoudre (ii).

## 5. Algorithme de von Neumann

*Algorithme de von Neumann :*
$(e_i)_i$ désigne la base canonique de $\mathbb{R}^p$
On pose $\forall x, y \in \mathbb{R}^p, B(x,y) = {}^tx\,{}^tTTy$ ; $B$ est bilinéaire symétrique. Soit $q$ la forme quadratique associée.

- $x_0 = e_1$
- Soit $j$ tel que $\forall i, B(x_k, e_j) \leq B(x_k, e_i)$

$\lambda_k := \arg\min_{\lambda \in [0,1]} q((1-\lambda)x_k + \lambda e_j)$
$x_{k+1} := (1-\lambda_k)x_k + \lambda_k e_j$

*Théorème 3.* Supposons que (ii) admette une solution. Soit $\epsilon > 0$. Alors l'algorithme de von Neumann converge en un nombre fini d'étapes vers une $\epsilon$-solution, i.e. vers $x \in \mathbb{R}^p$ tel que $||Tx|| \leq \epsilon$

Comme pour le perceptron, cet algorithme ne fournit pas de réponse explicite, puisqu'il n'est a priori pas possible de savoir en un nombre fini d'étapes si (ii) admet une solution.

## 6. Solution optimale

Jusqu'ici il s'agissait de trouver une solution au problème ${}^tTx > 0$ ; $x \in \mathcal{M}_{N,1}$. Dans le but d'utiliser le perceptron pour classer un grand nombre de motifs, il est utile de maximiser $\min\limits_{i\in[\![1,p]\!]} {}^twc_i$, tout en conservant $||w|| = 1$. On cherche ainsi le vecteur ou plutôt l'hyperplan qui pourra le mieux séparer les deux classes de motifs. On espère ainsi que la solution obtenue permettra de classer correctement de nouveaux motifs.

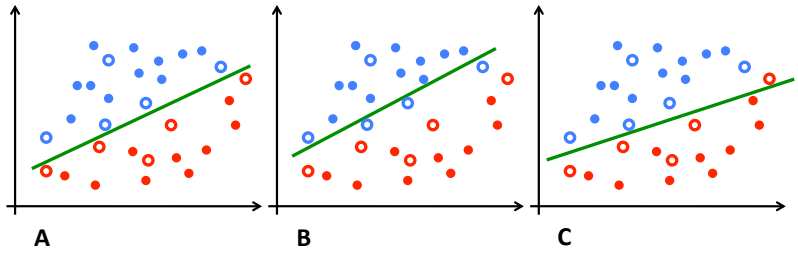



Figure 3. Différents hyperplans séparateurs.
Chaque plan sépare correctement les points pleins, mais seul le A peut être généralisé correctement aux points vides. De même le plan C se généralise mieux aux points vides que le plan B car il fait moins d'erreurs.

Le paramètre pertinent est $\rho(T) := \max\limits_{||w||=1} \min\limits_{i\in[\![1,p]\!]} {}^twc_i$. $\rho(T)$ est bien un maximum car $\forall\, i \in [\![1,p]\!]$, $\mathcal{M}_{N,1} \to \mathbb{R}, x \mapsto {}^txc_i$ est continue, donc $\mathcal{M}_{N,1} \to \mathbb{R}, x \mapsto \min\limits_{i\in[\![1,p]\!]} {}^txc_i$ est continue comme minimum de fonctions continues. Donc cette dernière est bornée et atteint ses bornes sur la sphère unité car celle-ci est compacte. On appelera, pour $y$ tel que $||y|| = 1$, le réel $\min\limits_i {}^tyc_i$ la marge de $y$.

*Proposition 4.* Si $\rho(T) > 0$, il existe un unique $w \in \mathcal{M}_{N,1}$ tel que $||w|| = 1$ et $\min\limits_{i\in[\![1,p]\!]} {}^twc_i = \rho(T)$

Comment trouver ce vecteur $w$ tel que $||w|| = 1$ et $\min\limits_i {}^twc_i = \rho(T)$ ? On l'appellera la solution optimale, qui est unique d'après ce qui précède. En modifiant légèrement l'algorithme du perceptron, et en conservant en mémoire la meilleure solution rencontrée, il est possible de converger vers la solution optimale.

*Perceptron amélioré :*

- $w_0 = 0$
- $w_{k+1} = w_k + c_j$ où $j = \arg\min\limits_i {}^t(\frac{w_k}{||w_k||})c_i$ (si $||w_k|| = 0$, arrêter l'algorithme, (i) n'a pas de solution)

*Théorème 5.* Supposons $\rho(T) > 0$. Soit $\epsilon > 0$. L'algorithme du perceptron amélioré converge en un nombre fini d'étapes vers $w \in \mathbb{R}^N$ tel que ${}^tT(\frac{w}{||w||}) > (1-\epsilon)\rho(T)$

Pour $F \subset \mathbb{R}^n$ posons $\text{Cone}\, F := \{\sum\limits_{i\in I} \lambda_i x_i \ ;\ I \text{ fini}, (\lambda_i)_i \in (\mathbb{R}_+)^I, (x_i)_i \in F^I\}$ l'enveloppe conique des éléments de $F$

*Corollaire 6.* Supposons $\rho(T) > 0$. Soit $u$ la solution optimale pour $\{c_i\}_{i\in[\![1,p]\!]}$. Alors $u \in \text{Cone}\,\{c_i\}_{i\in[\![1,p]\!]}$
Il est possible d'affiner le résultat précédent.

*Proposition 7.* Supposons $\rho(T) > 0$. Soit $w$ la solution optimale. Alors $w \in \text{Cone}\{c_i\,;\ {}^t w c_i = \rho(T)\}$

À l'aide des résultats sur la solution optimale, on va essayer d'établir un algorithme permettant de déterminer en un nombre fini d'étapes si $T$ est classifiable, et si oui, de déterminer la solution optimale.

## 7. Optimisation de l'algorithme

L'algorithme suivant est adapté de celui présenté dans *A Fast Method for Calculating the Perceptron with Maximal Stability* de Pál Ruján.

*Algorithme Pál Ruján.* On construit les suites $A \in (\mathcal{P}([\![1,p]\!]))^{\mathbb{N}}$, et $x \in (\mathbb{R}^n)^{\mathbb{N}}$ telles que pour tout $m \in \mathbb{N}$
(a) $\exists \{\lambda_i\}_{i\in A_m} \subset \mathbb{R}$ ; $x_m = \sum_{i\in A_m} \lambda_i c_i$ ; $||x_m|| = 1$ ou $x_m = 0$
(b) $\forall i \in A_m$, ${}^t c_i x_m = \alpha_m$
Si $x_m = 0$ (ou de façon équivalente $\alpha_m = 0$) et $\exists \{\lambda_i\}_{i\in A_m} \subset \mathbb{R}_+^*$ ; $x_m = \sum_{i\in A_m} \lambda_i c_i$ l'algorithme a terminé.
Si $\forall i \in [\![1,p]\!]$, ${}^t c_i x_m \geq \alpha_m$ et $\exists \{\lambda_i\}_{i\in A_m} \subset \mathbb{R}_+^*$ ; $x_m = \sum_{i\in A_m} \lambda_i c_i$ l'algorithme a terminé.

*Construction 8.*

- Soit $A_0 = \{1\}$ ; $x_0 = c_1$
- Supposons les suites construites jusqu'au rang $m$, et que l'algorithme n'a pas convergé au rang $m$

1) Soit $c_j \in \{c_i\}_{1\leq i\leq p}$ tel que ${}^t x_m c_j < \alpha_m$
2) On pose $S_m = (c_i)_{i\in A_m \cup \{j\}} \in \mathcal{M}_{N,|A_m|+1}$

Soit $\begin{pmatrix} z \\ a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix}$ une solution en $Z \in \mathcal{M}_{|A_m|+2,1}$ de l'équation $\left(\begin{array}{c|ccc} 0 & 1 & \cdots & 1 \\ \hline -1 & & & \\ \vdots & & {}^t S_m S_m & \\ -1 & & & \end{array}\right) Z = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

Si cette équation n'admet pas de solution, considérer $A'_m$ obtenu à partir de $A_m$ en lui retirant un élément, et $S'_m = (c_i)_{i\in A'_m \cup \{j\}}$, et recommencer.

3) On pose $x'_{m+1} = S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} \in \mathcal{M}_{N,1}$
Si $x'_{m+1} = 0$ alors $x_{m+1} = 0$
Si $x'_{m+1} \neq 0$ alors $x_{m+1} = \frac{x'_{m+1}}{||x'_{m+1}||}$
Alors $\alpha_{m+1} = \frac{z}{||x'_{m+1}||}$
On peut donc écrire $x_{m+1} = \sum_{i\in A_m\cup\{j\}} \mu_i c_i$ où $\mu_i = \frac{a_i}{||x'_{m+1}||}$
Soit $k \in A_m$ tel que $\mu_k = \min_i \mu_i$
Si $\mu_k \leq 0$ alors on pose $A_{m+1} = (A_m \setminus \{k\}) \cup \{j\}$ ; sinon $A_{m+1} = A_m \cup \{j\}$
Répéter les étapes 2 et 3 tant que $\min_i \mu_i \leq 0$

D'après les résultats qui précèdent, il existe $B \subset [\![1,p]\!]$ tel que si $y$ est la solution optimale, $y \in \text{Cone}\{c_i\}_{i\in B}$ et $\forall i \in B$, ${}^t y c_i = \rho(T)$. Il suffit donc pour déterminer la solution optimale de résoudre l'équation matricielle mentionnée plus haut dans la construction pour un certain $A_m = C \subset B$. On se contentera, pour justifier la pertinence de l'algorithme, de le comparer au perceptron sur différents ensembles de données.

## 8. Le perceptron en action

Les données utilisées proviennent du UCI Machine Learning Repository, et peuvent être trouvées ainsi que leur description sous http://archive.ics.uci.edu/ml/datasets/Iris. Il s'agit de données mesurées sur trois espèces d'iris (Setosa, Versicolor, Virginica) : en cm, 1) longueur des sépales, 2) largeur des sépales, 3) longueur des pétales, 4) largeur des pétales (les pétales entourent les organes reproducteurs de la fleur et forment la corolle. Les sépales forment le calice et supportent la corolle). En ajoutant le 1 nécessaire pour le seuil, $N = 5$. On regroupe dans une même matrice les données concernant deux espèces d'iris (ce qui fait $p = 100$), et on cherche à l'aide des différents algorithmes s'il existe des poids séparant les deux espèces d'iris. Les tableaux contenant les résultats se trouvent en annexe.
L'algorithme Pál Ruján donne des résultats satisfaisants sur ces données, la marge obtenue étant toujours supérieure à celle du perceptron (tableaux I et II). À partir de 10000 itérations, l'écart entre la marge du perceptron et celle de l'algorithme Pál Ruján est de l'ordre de $10^{-6}$. Le perceptron est donc efficace également. L'algorithme Pál Ruján termine également dans le cas non classifiable (tableau III). Plus la marge de la solution utilisée est grande, plus le nombre de nouveaux motifs correctement classés est élevé, donc mieux la solution se généralise (tableaux V et VI). De plus, une solution non optimale fait aussi bien qu'une solution optimale à partir d'un certain nombre d'itérations.

## Conclusion

L'étude du perceptron a permis de dégager et d'expliquer un algorithme permettant de converger en un nombre fini d'étapes vers la solution optimale si elle existe, et sinon, de résoudre le système d'équations alternatif du théorème de Gordan. L'algorithme perceptron et le principe du neurone formel a donné lieu à de nombreux développements comme le support vector machine et le perceptron multicouche, capables de classer des motifs qui ne sont pas séparables par un hyperplan. Ceux-ci sont souvent utilisés pour traiter des données physiques avant que celles-ci ne soient analysées par des chercheurs, comme les résultats des expériences au CERN (Hakl et al.), ou dans la reconnaissance de caractères (Jarousse et Viard-Gaudin). Le perceptron reste pour les chercheurs en sciences cognitives un modèle relativement réaliste du cerveau (Hung et al.), au contraire des réseaux de neurones non linéaires.

## Remerciements

Je tiens à remercier Dr. Srdjan Ostojic du Laboratoire de Neurosciences Cognitives de l'ENS pour son aide généreuse.

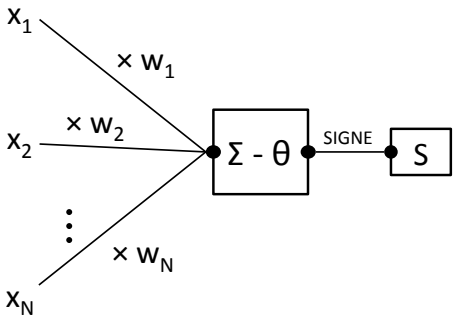



Figure 1. Schéma de fonctionnement du neurone formel. $x_i$ est le signal que reçoit l'entrée i, $w_i$ le poids de l'entrée i, θ le seuil.

## ANNEXES

### A. Preuves et démonstrations

*Démonstration 1.* Posons, en supposant $||u|| = 1$, $\alpha := \min_i {}^t u c_i > 0$

Soit $n \in \mathbb{N}$ tel que l'algorithme n'ait pas convergé au rang $n$

Donc $\forall k \leq n, \exists j \in [\![1,p]\!],\ {}^t c_j w_k \leq 0$

Soit $k \leq n-1$, et $j \in [\![1,p]\!]$, tel que ${}^t c_j w_k \leq 0$

$w_{k+1} = w_k + c_j \Rightarrow {}^t u(w_{k+1} - w_k) = {}^t u c_j \geq \alpha$

En sommant pour $0 \leq k \leq n-1$ : ${}^t u w_n \geq n\alpha$ et avec l'inégalité de Cauchy Schwarz $||w_n|| \geq n\alpha$ (1)

Soit $0 \leq k \leq n-1$, $||w_{k+1}||^2 = ||w_k||^2 + ||c_j||^2 + 2\,{}^t w_k c_j \leq ||w_k||^2 + 1 \Rightarrow ||w_{k+1}||^2 - ||w_k||^2 \leq 1$

En sommant pour $0 \leq k \leq n-1$, $||w_n||^2 \leq n$ (2)

(1) et (2) $\Rightarrow \alpha^2 n^2 \leq n \Rightarrow n \leq \frac{1}{\alpha^2}$

Par contraposition, si $n > \frac{1}{\alpha^2}$ alors l'algorithme a convergé.

*Démonstration 2.* On pose $|| \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} ||_1 = \sum_{i=1}^{p} |x_i|$

- Supposons par l'absurde que (i) et (ii) admettent une solution.

Il existe $x \geq 0, ||x||_1 = 1$ tel que $Ax = 0$ et il existe $y$ tel que ${}^t T y > 0$

Alors $0 = {}^t(Ax)y = {}^t x({}^t T y) > 0$, contradiction.

- Montrons que si (ii) n'a pas de solution alors (i) en a une.

Posons $K := \{Ax; x \geq 0, ||x||_1 = 1\}$ ($K$ est l'enveloppe convexe des colonnes de $A$).

Remarquons que $K$ est compact car fermé et borné, et convexe. Il existe donc $y \in K$ tel que $||y|| = \min_{x \in K} ||x||$

Par hypothèse $0 \notin K$ donc $y \neq 0$.

On a pour tout $x \in K$, pour tout $\lambda \in [0,1]$, $\lambda x + (1-\lambda)y \in K$ et

$||y|| \leq ||(\lambda x + (1-\lambda)y)|| = ||y + \lambda(x-y)||$

D'où $||y||^2 \leq ||y||^2 + \lambda^2 ||x-y||^2 + 2\lambda\,{}^t y(x-y) \Rightarrow 0 \leq \lambda ||x-y||^2 + 2\,{}^t y(x-y) \Rightarrow 0 \leq {}^t y(x-y)$ sinon on obtient une contradiction pour $\lambda > 0$ suffisamment proche de 0

On réecrit cette inégalité en $\forall x \in K, ||y||^2 \leq {}^t y x$

En particulier, si $(e_i)_{1 \leq i \leq p}$ est la base canonique de $\mathbb{R}^p$, $\forall i \in [\![1,p]\!], Te_i \in K$

$\Rightarrow \forall i \in [\![1,p]\!]\ {}^t y T e_i \geq ||y||^2$

$\Rightarrow \forall i \in [\![1,p]\!]\ {}^t({}^t T y) e_i \geq ||y||^2$

$\Rightarrow {}^t T y \geq ||y||^2 > 0$ car $y \neq 0$

Donc (ii) admet $y$ comme solution.

*Démonstration 3.* Supposons que l'algorithme n'a pas convergé après $n \in \mathbb{N}$ itérations. Soit $k \leq n-1$, on a $q(x_k) \geq \epsilon$

Comme (ii) admet une solution, (i) n'en admet pas, donc $\min_{i \in [\![1,p]\!]} B(x_k, e_i) \leq 0$

Soit $j$ tel que $\forall i, B(x_k, e_j) \leq B(x_k, e_i)$, d'après ce qui précède $B(x_k, e_j) \leq 0$

Posons $f : [0,1] \to \mathbb{R}, \lambda \mapsto q((1-\lambda)x_k + \lambda e_j)$

$f(\lambda) = (1-\lambda)^2 q(x_k) + \lambda^2 q(e_j) + 2(1-\lambda)\lambda B(x_k, e_j)$

$f(\lambda) = \lambda^2(q(x_k) + q(e_j) - 2B(x_k, e_j)) - 2\lambda(q(x_k) - B(x_k, e_j)) + q(x_k)$

$f'(\lambda) = 2[\lambda(q(x_k) + q(e_j) - 2B(x_k, e_j)) - (q(x_k) - B(x_k, e_j))]$

Avec Cauchy-Schwarz, et comme $|||T||| = 1$,

$f'(0) = -2(q(x_k) - B(x_k, e_j)) \leq 0$ et $f'(1) = q(e_j) - B(x_k, e_j) \geq 0$

Si les deux inégalités sont strictes, $f$ atteint un minimum strictement entre les deux.

Le minimum étant atteint en $\lambda_k$,

$f'(\lambda_k) = 0 \Rightarrow \lambda_k = \frac{q(x_k) - B(x_k, e_j)}{q(x_k) + q(e_j) - 2B(x_k, e_j)} = \frac{q(x_k) - B(x_k, e_j)}{q(x_k - e_j)}$

Cette expression de $\lambda_k$ est toujours valable lorsque $f'(0) = 0$ (minimum atteint en 0) ou $f'(1) = 0$ (minimum atteint en 1) puisqu'alors $\lambda_k = 0$ ou $\lambda_k = 1$. Il suffit de prendre en compte le fait que $f$ est une fonction polynômiale de degré 2.

On en déduit $x_{k+1} = x_k + \lambda_k(e_j - x_k) = x_k + \frac{q(x_k) - B(x_k, e_j)}{q(x_k - e_j)}(e_j - x_k)$

Alors $q(x_{k+1}) = q(x_k) + (\frac{q(x_k) - B(x_k, e_j)}{q(x_k - e_j)})^2 q(e_j - x_k) + 2\frac{q(x_k) - B(x_k, e_j)}{q(x_k - e_j)} B(x_k, e_j - x_k)$

$= q(x_k) + \frac{(q(x_k) - B(x_k, e_j))^2}{q(x_k - e_j)} - 2\frac{(q(x_k) - B(x_k, e_j))^2}{q(x_k - e_j)}$

$\Rightarrow q(x_{k+1}) - q(x_k) = -\frac{(q(x_k) - B(x_k, e_j))^2}{q(x_k - e_j)}$

Or $(q(x_k) - B(x_k, e_j))^2 \geq (q(x_k))^2 \geq \epsilon^2$

et $q(x_k - e_j) = {}^t x_k({}^t T T) e_j = {}^t(Tx_k)(Te_j) \leq ||Tx_k|| ||c_j|| \leq 1$

car $x_k = \sum_{i=1}^{p} \lambda_i e_i \Rightarrow Tx_k = \sum_{i=1}^{p} \lambda_i c_i \Rightarrow ||Tx_k|| \leq \sum_{i=1}^{p} |\lambda_i| ||c_i|| = \sum_{i=1}^{p} \lambda_i = 1$

D'où $\frac{(q(x_k) - B(x_k, e_j))^2}{q(x_k - e_j)} \geq \epsilon^2$

et $q(x_{k+1}) - q(x_k) \leq -\epsilon^2$

En sommant pour $k = 0, ..., n-1$ : $q(x_n) - q(x_0) \leq -n\epsilon^2$

donc $0 < q(x_n) \leq q(x_0) - n\epsilon^2 \Rightarrow n \leq \frac{1}{\epsilon^2}$

Par contraposition, si $n > \frac{1}{\epsilon^2}$, alors l'algorithme a convergé.

*Preuve 4.* Soit $u, v$ tels que $||u|| = ||v|| = 1$ et $\min_{i \in [\![1,p]\!]} {}^t u c_i = \min_{i \in [\![1,p]\!]} {}^t v c_i = \rho(T)$

Si $u = -v$, alors $\rho(T) \leq {}^t v c_1 = -{}^t u c_1 \leq -\min_i {}^t u c_i = -\rho(T)$

$\Rightarrow \rho(T) \leq 0$ ce qui est absurde.

Considérons $\frac{u+v}{||u+v||}$

$\min_i {}^t(\frac{u+v}{||u+v||}) c_i = \frac{1}{||u+v||} \min_i({}^t u c_i + {}^t v c_i)$

Or $||u+v|| \leq ||u|| + ||v|| = 2$ et

$\forall j,\ {}^t u c_j + {}^t v c_j \geq \min_i {}^t u c_i + \min_i {}^t v c_i \Rightarrow \min_i({}^t u c_i + {}^t v c_i) \geq \min_i {}^t u c_i + \min_i {}^t v c_i$

et $\rho(T)$ étant maximum, $\rho(T) \geq \min_i {}^t(\frac{u+v}{||u+v||}) c_i$

D'où $\rho(T) \geq \min_i {}^t(\frac{u+v}{||u+v||}) c_i \geq \frac{1}{2}(\min_i {}^t u c_i + \min_i {}^t v c_i) = \rho(T)$

Donc les inégalités sont des égalités.

On a $||u+v|| = ||u|| + ||v||$ donc $u = \lambda v$ où $\lambda \geq 0$

En passant à la norme, $\lambda = 1$ d'où $u = v$

On remarque qu'on a alors également $\min_i({}^t u c_i + {}^t v c_i) = \min_i {}^t u c_i + \min_i {}^t v c_i$

*Démonstration 5.* Pour simplifier les notations on pose $\alpha := \rho(T)$

Supposons que l'algorithme n'ait pas convergé au rang $n \in \mathbb{N}$, i.e.

$\forall k \in [\![0, n]\!], \min_i(c_i | \frac{w_k}{||w_k||}) \leq (1-\epsilon)\alpha$ (en effet à aucun moment $w_k = 0$ d'après le théorème de Gordan).

Soit $k \leq n-1$

Pour $t \leq k-1$, $w_{t+1} - w_t = c_j \Rightarrow {}^t u(w_{t+1} - w_t) = {}^t u c_j \geq \alpha$

En sommant ${}^t u w_k \geq k\alpha \Rightarrow ||w_k|| \geq k\alpha$ (1)

Donc à partir d'un certain rang, $||w_k|| \geq 2$ ce qu'on va supposer par la suite quitte à réindexer la suite $(w_k)_k$

Soit $k \leq n-1$, $w_{k+1} = w_k + \Delta_k$ où ${}^t \frac{w_k}{||w_k||}\Delta_k \leq (1-\epsilon)\alpha$ et $\Delta_k \in \{c_i\}_{1\leq i\leq p}$

On écrit $\Delta_k = \lambda_k w_k + y_k$ où $y_k \in (\text{Vect}\{w_k\})^\perp$

Donc $1 = ||\Delta_k||^2 = \lambda_k^2||w_k||^2 + ||y_k||^2 \Rightarrow ||y_k|| \leq 1$

- $1^{\text{er}}$ cas $\lambda_k \leq 0$

$\lambda_k w_k = \Delta_k - y_k \Rightarrow |\lambda_k|||w_k|| \leq 2$ car $||\Delta_k|| = 1$

Donc $|\lambda_k| \leq 1$

Alors $||w_{k+1}||^2 = (1+\lambda_k)^2||w_k||^2 + ||y_k||^2 \leq ||w_k||^2 + 1$ car $-1 \leq \lambda_k \leq 0$

Donc $||w_{k+1}||^2 \leq (||w_k|| + \frac{1}{2||w_k||})^2$

- $2^{\text{e}}$ cas $\lambda_k > 0$

Alors ${}^t \frac{w_k}{||w_k||}\Delta_k \leq (1-\epsilon)\alpha \Rightarrow {}^t \frac{w_k}{||w_k||}(\lambda_k w_k + y_k) \leq (1-\epsilon)\alpha \Rightarrow \lambda_k||w_k|| \leq (1-\epsilon)\alpha$

$||w_{k+1}|| \leq ||w_k + y_k|| + \lambda_k||w_k||$

On a $||w_k + y_k||^2 = ||w_k||^2 + ||y_k||^2 \leq ||w_k||^2 + 1 \leq (||w_k|| + \frac{1}{2||w_k||})^2$

et $\lambda_k||w_k|| \leq (1-\epsilon)\alpha$

Donc $||w_{k+1}|| \leq ||w_k|| + \frac{1}{2||w_k||} + (1-\epsilon)\alpha$

Dans les deux cas ci-dessus, $||w_{k+1}|| - ||w_k|| \leq \frac{1}{2||w_k||} + (1-\epsilon)\alpha$

et avec (1) $||w_{k+1}|| - ||w_k|| \leq \frac{1}{2\alpha}\frac{1}{k} + (1-\epsilon)\alpha$

En sommant pour $k = 1, ..., n-1$ : $||w_n|| \leq ||w_1|| + \frac{1}{2\alpha}H_{n-1} + (n-1)(1-\epsilon)\alpha$ (2)

où $H_n = \sum_{k=1}^{n} \frac{1}{k}$

(1) et (2) $\Rightarrow n\alpha \leq ||w_1|| + \frac{1}{2\alpha}H_{n-1} + (n-1)(1-\epsilon)\alpha$ i.e. $n\epsilon\alpha \leq ||w_1|| + (\epsilon-1)\alpha + \frac{1}{2\alpha}H_{n-1}$ ce qui n'est valable que pour $n$ suffisamment petit puisque $H_n \sim \ln n$

Donc à partir d'un certain rang, l'algorithme a convergé.

*Preuve 6.* Posons pour $x \in \mathbb{R}^N$, $f(x) = \min_i {}^t x c_i$ ; $f$ est continue. Considérons la suite $(w_k)$ définie dans l'algorithme du perceptron amélioré. D'après le théorème précédent, $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}, f(\frac{w_k}{||w_k||}) \geq (1-2^{-n})\rho(T)$

On peut donc construire $\varphi : \mathbb{N} \to \mathbb{N}$ telle que $\forall n \in \mathbb{N}, f(\frac{w_{\varphi(n)}}{||w_{\varphi(n)}||}) \geq \rho(T)(1-2^{-n})$

Or $\forall n \in \mathbb{N}, \frac{w_\varphi(n)}{||w_\varphi(n)||} \in S(0,1)$ compacte en dimension finie, donc il existe $\psi$ strictement croissante telle que $\frac{w_{\varphi\circ\psi(n)}}{||w_{\varphi\circ\psi(n)}||} \to w$

On a toujours $\forall n \in \mathbb{N}, f(\frac{w_{\varphi\circ\psi(n)}}{||w_{\varphi\circ\psi(n)}||}) \geq \rho(T)(1-2^{-\psi(n)})$ et par continuité de $f$, $f(w) \geq \rho(T)$

Par unicité de la solution optimale, $w = u$

Or $\forall n, \frac{w_n}{||w_n||} \in \text{Cone}\{c_i\}_i$ qui est un fermé, donc $w \in \text{Cone}\{c_i\}_i$ ie $u \in \text{Cone}\{c_i\}_i$

*Preuve 7.* Pour simplifier soit $\alpha := \rho(T)$. Soit $A = \{i\,;\ {}^t w c_i = \alpha\}$

Soit $y$ la solution optimale pour $\{c_i\,;\, i \in A\}$ et $\beta = \min_{i\in A} {}^t y c_i$. Nécessairement $\beta \geq \alpha$.

Posons pour $\lambda \in [0,1]$, $z(\lambda) = \frac{\lambda w + (1-\lambda)y}{||\lambda w + (1-\lambda)y||}$

$z(1) = w$ donc le produit scalaire étant continu et $[\![1,p]\!] \setminus A$ étant fini, il existe $1 > \eta > 0$ tel que pour tout $i \notin A$, $1-\eta \leq \lambda \leq 1 \Rightarrow {}^t z(\lambda)c_i > \alpha$

Or $||\lambda w + (1-\lambda)y|| \leq |\lambda| + |1-\lambda| = 1$ et donc

$\forall i \in A$, ${}^t z(\lambda)c_i = \frac{\lambda\, {}^t w c_i + (1-\lambda)\, {}^t y c_i}{||\lambda w + (1-\lambda)y||} \geq \lambda\alpha + (1-\lambda)\beta$

Si $\beta > \alpha$, fixons $\lambda$ tel que $\lambda < 1$ et $\lambda > 1-\eta$

Avec ce qui précède on a $\forall i \in A$, ${}^t z(\lambda)c_i > \lambda\alpha + (1-\lambda)\alpha = \alpha$ et $\forall i \notin A$, ${}^t z(\lambda)c_i > \alpha$, ce qui contredit l'optimalité de $w$

Donc nécessairement $\alpha = \beta$. Alors $w = y$ par unicité de la solution optimale.

Or avec le corollaire 6, $y \in \text{Cone}\,\{c_i\}_{i\in A}$

Donc $w \in \text{Cone}\,\{c_i\,;\ {}^t w c_i = \rho(T)\}$

*Justification de la construction 8.*

On reprend les notations de la construction en supposant les suites construites jusqu'au rang $m$.

• Montrons que $x_{m+1}$ existe. Si l'équation ci-dessus admet une solution, alors $x_{m+1}$ existe. Sinon, si l'on est amené à retirer suffisamment d'éléments de $A_m$, alors $A_m = \varnothing$ et $S_m \in \mathcal{M}_{N,1}$

Alors l'équation s'écrit $\begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix} Z = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et admet clairement $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ comme solution.

Donc $x_{m+1}$ existe.

• Supposons $x_m = \sum\limits_{i\in A_m} \mu_i c_i$ construit.

Si $x'_{m-1} = 0$ alors $\sum\limits_{i\in A_m} \mu_i = \sum\limits_{i\in A_m} a_i = 1$

Si $x'_{m-1} \neq 0$ alors $\sum\limits_{i\in A_m} \mu_i = \sum\limits_{i\in A_m} \frac{a_i}{||x'_{m-1}||} = \frac{1}{||x'_{m-1}||} > 0$

Donc $\sum\limits_{i\in A_m} \mu_i \neq 0$

• Soit $S_m \in \mathcal{M}_{N,|A_m|+1}$ et $j$ comme définis dans la construction.

Considérons $Q := \left(\begin{array}{c|ccc} 0 & 1 & \cdots & 1 \\ \hline -1 & & & \\ \vdots & & {}^t S_m S_m & \\ -1 & & & \end{array}\right) \in \mathcal{M}_{N,|A_m|+2}$

Soit $\begin{pmatrix} y \\ a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} \in \text{Ker}Q$

Alors

$$||S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix}||^2 = {}^t(S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix})(S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix}) = {}^t \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} ( {}^t S_m S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix})$$

$$||S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix}||^2 = {}^t \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} y \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = y(\sum_{i=1}^{|A_m|+1} a_i)$$

Or $\sum\limits_{i=1}^{|A_m|+1} a_i = 0$ d'où $S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} = 0$

D'où $\begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} \in \text{Ker}S_m$. Donc $y \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = {}^t S_m S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} = 0 \Rightarrow y = 0$

Donc $\text{Ker}Q \subset \{ \begin{pmatrix} 0 \\ a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} ; \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} \in \text{Ker}S_m \}$

$x_{m+1}$ est de la forme $x_{m+1} = S_m \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_{|A_m|+1} \end{pmatrix}$ avec $\sum_i \lambda_i \neq 0$

On a $Q \frac{1}{\sum_i \lambda_i} \begin{pmatrix} \alpha_{m+1} \\ \lambda_1 \\ \vdots \\ \lambda_{|A_m|+1} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

Si $\begin{pmatrix} z \\ a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix}$ est solution de l'équation en $Z$ : $QZ = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ alors

$\frac{1}{\sum_i \lambda_i} \begin{pmatrix} \alpha_{m+1} \\ \lambda_1 \\ \vdots \\ \lambda_{|A_m|+1} \end{pmatrix} - \begin{pmatrix} z \\ a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} \in \text{Ker}Q$

$\Rightarrow S_m(\frac{1}{\sum_i \lambda_i} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_{|A_m|+1} \end{pmatrix} - \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix}) = 0$

$\Rightarrow \frac{1}{\sum_i \lambda_i} x_{m+1} = S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix}$

Si $x_{m+1} = 0$ alors $S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix} = 0$ et on a le résultat attendu.

Sinon posons $w = \dfrac{S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix}}{||S_m \begin{pmatrix} a_1 \\ \vdots \\ a_{|A_m|+1} \end{pmatrix}||}$

Alors $\forall i \in A_m \cup \{j\}$, ${}^t w c_i = {}^t(\frac{\frac{1}{\sum_i \lambda_i} x_{m+1}}{||\frac{1}{\sum_i \lambda_i} x_{m+1}||}) c_i = {}^t x_{m+1} c_i = \alpha_{m+1}$

Donc $w = x_{m+1}$

Donc toute solution de l'équation convient.

## C. Résultats numériques

Tableau I. Résultats des algorithmes perceptron et Pál Ruján pour Setosa et Versicolor.

Pour ces données, l'algorithme von Neumann renvoie, après une dizaine d'itérations, l'exception "classifiable".

| Algorithme utilisé | Perceptron | | | | Pál Ruján |
|---|---|---|---|---|---|
| Nombre d'itérations | 100 | 1000 | 10000 | 100000 | 5 |
| Poids 1 | 0.1400766649040 | 0.1546403153510 | 0.1541962758770 | 0.1541628699560 | 0.1541505358360 |
| Poids 2 | 0.4205896963710 | 0.4353372224960 | 0.4351936729490 | 0.4351869907260 | 0.4351910464120 |
| Poids 3 | -0.7638051339760 | -0.7566023468160 | -0.7567049221750 | -0.7566976971950 | -0.7566978439400 |
| Poids 4 | -0.4648301866680 | -0.4557571263640 | -0.4554152555730 | -0.4554176531520 | -0.4554097344570 |
| Poids 5 | 0.0633833584521 | 0.0800387789719 | 0.0826110239566 | 0.0827613973201 | 0.0828052665177 |
| Marge | 0.0990964805488 | 0.1234363669740 | 0.1234723676810 | 0.1234750469340 | 0.1234751417700 |

Tableau II. Résultats des algorithmes perceptron et Pál Ruján pour Setosa et Virginica.

Pour ces données, l'algorithme von Neumann renvoie, après une dizaine d'itérations, l'exception "classifiable".

| Algorithme utilisé | Perceptron | | | | Pál Ruján |
|---|---|---|---|---|---|
| Nombre d'itérations | 100 | 1000 | 10000 | 100000 | 5 |
| Poids 1 | 0.188762894919 | 0.1956346036740 | 0.194172185122 | 0.194153811209 | 0.194150868708 |
| Poids 2 | 0.457251697248 | 0.467882621586 | 0.467183077930 | 0.467178247813 | 0.467179522171 |
| Poids 3 | -0.736795341695 | -0.730717580296 | -0.731365519268 | -0.731366109426 | -0.731366797432 |
| Poids 4 | -0.444172469400 | -0.437523588989 | -0.437628195522 | -0.437611407389 | -0.437607161134 |
| Poids 5 | 0.123015839424 | 0.132052569793 | 0.132752302925 | 0.132848230958 | 0.132858249210 |
| Marge | 0.174779090355 | 0.186127813083 | 0.187720698182 | 0.187721891670 | 0.187721918269 |

Tableau III. Résultats des algorithmes perceptron et Pál Ruján pour Versicolor et Virginica.

| Algorithme utilisé | Perceptron | | | | Pál Ruján |
|---|---|---|---|---|---|
| Nombre d'itérations | 100 | 1000 | 10000 | 100000 | 8 |
| Poids 1 | 0.367919023088 | 0.367919023088 | 0.367919023088 | 0.367919023088 | - |
| Poids 2 | -0.445265880893 | -0.445265880893 | -0.445265880893 | -0.445265880893 | - |
| Poids 3 | -0.732534191146 | -0.732534191146 | -0.732534191146 | -0.732534191146 | - |
| Poids 4 | -0.330358556792 | -0.330358556792 | -0.330358556792 | -0.330358556792 | - |
| Poids 5 | -0.143634155127 | -0.143634155127 | -0.143634155127 | -0.143634155127 | - |
| Marge | -0.418294498009 | -0.418294498009 | -0.418294498009 | -0.418294498009 | 0 |

Tableau IV. Résultats de l'algorithme von Neumann pour Versicolor et Virginica. Les coefficients renvoyés par l'algorithme ont été volontairement omis.

| Nombre d'itérations | 100 | 1000 | 10000 | 25000 |
|---|---|---|---|---|
| Norme | 0.0243107666328 | 0.00785110452992 | 0.00216187359078 | 0.00140032759161 |

Dans les tableaux I à IV, pour chaque ensemble de données, $p = 100$. Les motifs ont été normalisés ($\forall i \in [\![1, 100]\!], ||c_i|| = 1$).

Tableau V. Résultats obtenus en entrainant l'algorithme perceptron et Pál Ruján sur $\{c_i; i \mod 10 = 0\}$ $(p = 10)$, puis en testant la solution obtenue sur $\{c_i; i \mod 10 \neq 0\}$ $(p = 90)$, à partir de Virginica et Versicolor.

| Algorithme utilisé | Perceptron | | | | Pál Ruján |
|---|---|---|---|---|---|
| Nombre d'itérations | 100 | 1000 | 10000 | 100000 | 4 |
| Poids 1 | -0.216424457817 | -0.19354798980700 | -0.2142199141150 | -0.2146130817680 | -0.2146428709890 |
| Poids 2 | 0.542206589399 | 0.56044856152000 | 0.5418628872420 | 0.5413673965790 | 0.5413381552030 |
| Poids 3 | -0.331720337755 | 0.04246400290110 | 0.0531195258783 | 0.0544422422697 | 0.0545454591253 |
| Poids 4 | -0.647426690655 | -0.62173463247000 | -0.6257356489500 | -0.6266112826290 | -0.6266881058040 |
| Poids 5 | 0.360517359108 | 0.50997984350000 | 0.5158756181770 | 0.5150309794510 | 0.5149449005350 |
| Marge | -0.227985758001 | -0.00201912812196 | 0.0134735920685 | 0.0134762964608 | 0.0134765024015 |
| Nombre de bonnes réponses | 45 | 73 | 79 | 79 | 79 |
| Taux de réussite | 50% | 81,1% | 87,8% | 87,8% | 87,8 % |

Tableau VI. Résultats obtenus en entrainant l'algorithme perceptron et Pál Ruján sur $\{c_i; i \mod 20 = 0\}$ ($p = 10$), puis en testant la solution obtenue sur $\{c_i; i \mod 10 \neq 0\}$ ($p = 90$), à partir de Virginica et Versicolor.

| Algorithme utilisé | Perceptron | | | | Pál Ruján |
|---|---|---|---|---|---|
| Nombre d'itérations | 100 | 1000 | 10000 | 100000 | 3 |
| Poids 1 | -0.352872300091 | -0.18694816554100 | -0.2048001538160 | -0.2049010608690 | -0.2049100471040 |
| Poids 2 | 0.654325864301 | 0.76305919735800 | 0.7552130391820 | 0.7552626547920 | 0.7552671733430 |
| Poids 3 | -0.206173174060 | -0.05029481053510 | -0.0607196302002 | -0.0603313845352 | -0.0602959144453 |
| Poids 4 | -0.573242495370 | -0.56578490675200 | -0.5725894811770 | -0.5728959805970 | -0.5729239597530 |
| Poids 5 | 0.276087788459 | 0.24525275952200 | 0.2369906643750 | 0.2361020901810 | 0.2360209916700 |
| Marge | -0.203870755616 | -0.00917129518279 | 0.0144867694591 | 0.0144887943349 | 0.0144888220863 |
| Nombre de bonnes réponses | 45 | 73 | 81 | 81 | 81 |
| Taux de réussite | 50% | 81,1% | 90% | 90% | 90% |

On remarque à partir des tableaux IV et V que le taux de réussite peut être supérieur si l'on entraîne le perceptron sur un nombre plus faible de motifs. Cela s'explique par le fait que les motifs ne sont pas séparables par un hyperplan, et qu'ainsi un hyperplan qui convient pour un petit nombre de motifs n'a aucune raison de convenir pour un grand nombre.

## D. Bibliographie

Références citées :


Frantisek Hakl, Marek Hlavcek et Roman Kalous, *Application of Neural Networks Optimized by Genetic Algorithms to Higgs Boson Search* in *Lecture Notes in Computer Science*, Volume 2331, éd. Springer, 2002, p. 554-563.

Chou P. Hung, Gabriel Kreiman, Tomaso Poggio et James J. DiCarlo, *Fast Readout of Object Identity from Macaque Inferior Temporal Cortex* in Science, Volume 310, 2005, p. 863-866

C. Jarousse et C. Viard-Gaudin, *Localisation du code postal par rseau de neurones sur bloc adresse manuscrit non contraint* in CIFED'98, Mai 1998, p. 72-81.

Marvin L. Minsky et Seymour A. Papert, *Perceptrons : an introduction to computational geometry*, éd. The MIT Press, Cambridge MA, 1962

Frank Rosenblatt, *Principles of Neurodynamics: Perceptrons and the theory of brain mechanisms*, éd. Spartan Books, 1962

Pál Ruján, *A Fast Method for Calculating the Perceptron with Maximal Stability*, Journal de Physique I, Volume 3, 1993, p. 277-290


Références supplémentaires :


Dan Li et Tams Terlaky, *The Duality between the Perceptron Algorithm and the von Neumann Algorithm*, Lehigh University, 2012

J.K. Anlauf et M. Biehl, *The AdaTron : an Adaptive Perceptron Algorithm*, Europhysics Letters, Volume 10, 1989, p. 687-692

Negar Soheili et Javier Pea, *A smooth perceptron algorithm*, Carnegie Mellon University, 2011

Omri Barak et Mattia Rigotti, *A Simple Derivation of a Bound on the Perceptron Margin Using Singular Value Decomposition*, in *Neural Computation*, Volume 23, 2011, 19351943

E. Gardner, *The space of interactions in neural network models*, in *Journal of Physics A: Mathematical and General*, Volume 21, 1988, p. 257-270

Avrim Blum, Alan Frieze, Ravi Kannan et Santosh Vempala, *A Polynomial-time Algorithm for Learning Noisy Linear Threshold Functions*, in *Proceedings of 37th Conference on Foundations of Computer Science*, 1996