AFSCET

Res-Systemica

Revue Française de Systémique

Fondée par Evelyne Andreewsky

Volume 19, automne 2019

Systémique du signe et du sens

Res-Systemica, volume 19, article 01

Compatibilité électromagnétique et réseaux de neurones

Olivier Maurice

12 pages

contribution reçue le 02 mars 2019

Compatibilité électromagnétique et réseaux de neurones

Olivier MAURICE - AFSCET

24 février 2019

olivier.maurice@e-nautia.com

1 Préambule

Les réseaux neuronaux interviennent dans les intelligences artificielles (IA) en permettant l'identification de scènes plus ou moins complexes. Ces identifications agissent comme une mémoire de formes et transmettent à l'IA la capacité de mémoire et tri. Ensuite, l'IA, en se dotant d'organes de décisions, peut constituer une première ébauche d'autonomie au sens de la robotique (et non de la systémique).

Nous pouvons représenter un neurone artificiel par la structure dessinée figure 1.

Des stimuli p^k sont présentés à l'entrée d'un dispositif qui en fait la somme. A cette somme, un terme de biais est retranché et après application d'une fonction f, la sortie est obtenue. Nous écrivons sous une algèbre tensorielle d'opérateur fondamental ω :

$$n_1 = \omega_{1k} p^k, \;\; a_1 = f \cdot n_1 \tag{1}$$

Cette relation est établie pour un seul neurone. Pour un ensemble de neurones nous pouvons écrire :

$$a_q = f \cdot \left(\omega_{qk} p^k - b_q \right) \tag{2}$$

f est couramment appelée "fonction d'activation" .

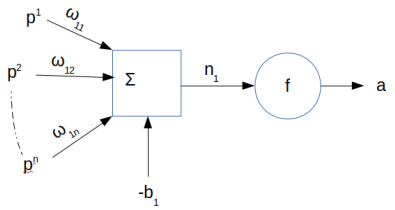

FIGURE 1 – Neurone Artificiel

2 Couches et apprentissage

Les neurones peuvent être organisés en couches. Ainsi, si une première couche est définie par la relation (2), une couche suivante devrait s'écrire, en respectant la nature contravariante des entrées :

$$c_r = g \cdot (\nu_{rq} a^q - d_r) \tag{3}$$

Pour que les deux équations se succèdent, il nous faut créer un lien par l'intermédiaire d'un objet y tel que $a^q = y^{qq} a_q$. C'est en ajustant les poids ω ou ν que le réseau de neurones "apprend". La sortie du réseau passera ainsi à 1 pour un jeu particulier des stimuli en entrée. Comment peut s'effectuer ce réglage des poids ?

Dans le principe nous allons nous intéresser à une dépendance entre la variation des poids et un certain critère de rapprochement entre la solution identifiée et utilisée en apprentissage, et la solution proposée par le réseau. Soit la variation donnée par :

$$\Delta\omega_{k\sigma}(t) = \omega_{k\sigma}(t+1) - \omega_{k\sigma}(t)$$

Nous en déduisons

$$\omega_{k\sigma}(t+1) = \omega_{k\sigma}(t) + \Delta\omega_{k\sigma}(t)$$

Soit a_q le vecteur de sortie d'un réseau neuronal (RN), apprendre c'est déjà connaître l'écart entre une sortie désirée d_q et la sortie effective du RN a_q : $\epsilon_q(t) = d_q(t) - a_q(t)$. L'apprentissage par correction des erreurs (assez inspiré pour le coup des processus réels) consiste à choisir un critère de performance F, par exemple défini par $F = \epsilon_q \epsilon^q$, puis à essayer de le minimiser. Si nous posons

$$\omega_{qr}(t+1) = \omega_{qr}(t) + \eta x_{qr}(t) \tag{4}$$

Par remplacement nous obtenons :

$$a_q = (\omega_{qr}(t+1) p^r - b_q) \cdot f \tag{5}$$

et

$$F = \sum_q \left\{ d_q - f \cdot \left([\omega_{qr}(t) + \eta x_{qr}](t) \right] p^r - b_q \right) \}^2 \tag{6}$$

3 Formulation par stimuli covariants

posons

$$a^\alpha = \tau^{\alpha\sigma} \cdot p_\sigma \tag{7}$$

p_σ sont les stimuli appliqués en entrée du RN et a^α la sortie du RN. Finalement notre objectif est de trouver les opérateurs $\tau^{\alpha\sigma}$. Basiquement si ces opérateurs sont de simples facteurs nous avons :

$$\tau^{\alpha\sigma} = \frac{\partial a^\alpha}{\partial p_\sigma} \tag{8}$$

D'une manière plus générale, nous voulons minimiser :

$$S^\alpha = \int_t \left(d^\alpha(t) - y^{\alpha\sigma} p_\sigma(t) \right)^2 \Rightarrow \frac{\partial S^\alpha}{\partial y^{\alpha\sigma}} = 0 \tag{9}$$

Si les opérateurs y sont des polynômes, cela revient à une régression polinomiale sur chaque α, pour $\{t\}$.

4 Problème à deux couches

Nous avons une première couche de la forme :

$$a^k = y^{kn} p_n$$

et une seconde couche qui suit cette première :

$$b^q = g^{q\sigma} A_{\sigma k} y^{kn} p_n$$

L'opérateur A est l'opérateur de décision.

Considérons des sorties z d'expressions $z_i = ax_i + b$. $ax_i + b$ est la loi (l'opérateur) appliqué aux entrées x et dont nous voulons qu'elle engendre les sorties z. La minimisation de l'écart $z_i - f(x_i)$ devient :

$$S = \sum_i \left(z_i - (ax_i + b)\right)^2 \to 0 \tag{10}$$

soit :

$$S = \sum_i \left\{z_i^2 + a^2 x_i^2 + b^2 + 2ax_i b - 2z_i a x_i - 2z_i b\right\}$$

Alors :

$$\frac{\partial S}{\partial a} = 2a \sum_i x_i^2 + 2 \sum_i x_i b - 2 \sum_i x_i z_i = 0 \tag{11}$$

En réalisant la même opération pour $\partial_b S$ nous obtenons un système de deux équations qui nous permet de résoudre les valeurs de a et b.

Considérons alors un RN définit par

$$a^k = \omega^{k\sigma} \cdot p_\sigma \tag{12}$$

Sur la première sortie S^1 nous avons

$$S^1 = \sum_\sigma \left[d^1 - \omega^{1\sigma}(p_\sigma)\right]^2 \tag{13}$$

Définisson alors

$$\omega^{1\sigma} \cdot_\sigma = \gamma^{1\sigma} p_\sigma + \beta^1$$

et de fait :

$$S^1 = \sum_\sigma \left[d^1 - \left(\gamma^{1\sigma} p_\sigma + \beta^1\right)\right]^2 \tag{14}$$

En calculant :

$$\frac{\partial S^1}{\partial \gamma^{1\sigma}}, \quad \frac{\partial S^1}{\partial \beta^1}$$

Nous obtenons le système :

$$\begin{cases} 2\sum_\sigma p_\sigma \gamma^{1\sigma} + 2\sum_\sigma p_\sigma \beta^1 = 2\sum_\sigma p_\sigma d^1 \\ 2\sum_\sigma p_\sigma \gamma^{1\Sigma} + 2\bar{\sigma}\beta^1 = 2\bar{\sigma} d^1 \end{cases} \tag{15}$$

avec

$$\bar{\sigma} = \sum_\sigma \sigma$$

En résolvant ce système nous trouvons bien sûr ici $\gamma^{1\sigma} = 0$ et $\beta^1 = d^1$.

5 Neurone

Un neurone intègre un certain nombre de percepts p_σ, scalaires et valeurs issus de l'environnement. De cette intégration, il engendre une composante d'un vecteur de réaction, ou réponse a^α, ici pour le premier neurone d'une chaîne a^1. L'opérateur d'intégration qui ne prend pas en charge dans ce modèle la décision : y, est un opérateur quelconque qui s'applique aux percepts. Les stimuli étant indépendants et scalaires, en même temps que la réponse du neurone ne peut être dissociée de ces stimuli ; les deux ensembles sont liés et les stimuli peuvent être vus comme une mesure du comportement du neurone. Nous définissons les p_σ comme des éléments d'un espace dual aux a^α. L'opérateur y est alors le tenseur fondamental qui relie les a^α aux p_σ. Nous écrivons :

$$a^\alpha = y^{\alpha\sigma} p_\sigma \tag{16}$$

soit pour le neurone 1 : $a^1 = y^{11}p_1 + y^{12}p_2 + \ldots$.

Détermination de l'opérateur y Si nous choisissons une loi du type $y^{1\sigma} = \gamma^{1\sigma}(\bullet) + \beta^{1\sigma}_\sigma$. Avec deux stimuli nous trouvons :

$$a^1 = \gamma^{11}p_1 + \beta^{11}_1 + \gamma^{12}p_2 + \beta^{12}_2$$

Le problème est de déterminer les cœfficients γ^{11}, β^{11}_1, γ^{12}, β^{12}_2. Si a^1 doit tendre vers une valeur désirée d^1 nous voulons minimiser :

$$S^{11} = \left|\left|a^1 - d^1\right|\right|^2 \to 0 \tag{17}$$

En développant cette expression nous trouvons

$$S^{11} = d^1d^1 - 2\gamma^{11}d^1p_1 - 2\beta^{11}_1 d^1 - 2\gamma^{12}d^1p_2 - 2\beta^{12}_2 d^1 + \gamma^{11}\gamma^{11}p_1p_1 + 2\gamma^{11}\beta^{11}_1 p_1 + 2\gamma^{11}\gamma^{12}p_1p_2 + \ldots$$

$$\ldots + 2\gamma^{11}\beta^{12}_2 p_1 + \beta^{11}_1\beta^{11}_1 + 2\beta^{11}_1\gamma^{12}p_2 + 2\beta^{11}_1\beta^{12}_2 + \gamma^{12}\gamma^{12}p_2p_2 + 2\gamma^{12}\beta^{12}_2 p_2 + \beta^{12}_2\beta^{12}_2$$

Si nous prenons deux neurones identiques pour simplifier cette expression et être illustratif, soit à poser $\gamma^{11} = \gamma^{12} = \gamma^1$ et $\beta^{11}_1 = \beta^{12}_2 = \beta^1$. L'expression se réduit alors à :

$$S^{11} = d^1d^1 - 2\gamma^1 d^1(p_1+p_2) - 4\beta^1 d^1 + \gamma^1\gamma^1(p_1p_1 + p_2p_2 = 2p_1p_2) + \gamma^1\beta^1 4(p_1+p_2) + 4\beta^1\beta^1$$

Les deux minima sont ici liés :

$$\frac{\partial S^{11}}{\partial \gamma^1} = Q\frac{\partial S^{11}}{\partial \beta^1} = 0$$

avec $Q = p_1 + p_2$.

Finalement les cœfficients sont liés par $Q\gamma^1 + 2\beta^1 = d^1$.

Pour $p_\sigma = 0$ le neurone sort $a^\sigma = n\beta^1$ pour n stimuli. Nous pouvons de fait définir arbitrairement β^1 avec d^1 réponse attendue. β^1 apparaît comme un offset, souvent fixé à une vlaeur négative appelée le **biais**, ou encore **seuil d'activation** du neurone. Alors :

$$\gamma^1 = \frac{d^1 - 2\beta^1}{Q} \tag{18}$$

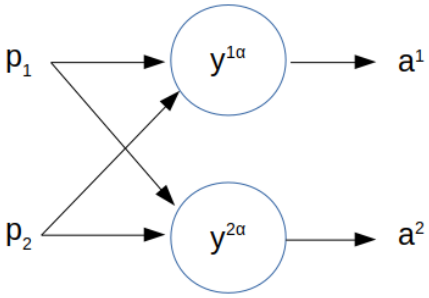

FIGURE 2 – Neurone Artificiel

L'architecture de y est :

$$y^{\alpha\sigma} = \begin{bmatrix} \gamma^1(\bullet) + \beta^1 & 0 \\ 0 & \gamma^1(\bullet) + \beta^1 \end{bmatrix} \tag{19}$$

et $a^1 = y^{1\sigma} p_\sigma$.

6 Couche à deux neurones

Nous considérons le réseau a deux neurones présenté figure 2. Nous choisissons $y^{i\alpha}$ de la forme $\gamma^{i\alpha}(\bullet) + \beta^{i\alpha}_\alpha/2$. L'opérateur y est alors défini par :

$$y^{\alpha\sigma} = \begin{bmatrix} \gamma^{1\sigma}(\bullet) + \frac{\beta^{1\sigma}_\sigma}{2} & \gamma^{1\sigma}(\bullet) + \frac{\beta^{1\sigma}_\sigma}{2} \\ \gamma^{2\sigma}(\bullet) + \frac{\beta^{2\sigma}_\sigma}{2} & \gamma^{2\sigma}(\bullet) + \frac{\beta^{2\sigma}_\sigma}{2} \end{bmatrix} \tag{20}$$

Pour simplifier l'écriture nous notons les fonctions γ^1 et β^1. Nous obtenons le système suivant :

$$\begin{cases} a^1 = \gamma^1(p_1) + \frac{\beta^1}{2} + \gamma^1(p_2) + \frac{\beta^1}{2} \\ a^2 = \gamma^2(p_1) + \frac{\beta^2}{2} + \gamma^2(p_2) + \frac{\beta^2}{2} \end{cases} \tag{21}$$

ou en posant $Q = p_1 + p_2$:

$$\begin{cases} a^1 = \gamma^1 Q + \beta^1 \\ a^2 = \gamma^2 Q + \beta^2 \end{cases} \tag{22}$$

Nous prenons le premier jeu d'équations. Pour deux essais où nous fixons les valeurs de a^1 : $d^1(1)$ et $d^1(2)$ et du stimuli Q, nous trouvons :

$$\begin{cases} d^1(1) = \gamma^1 Q(1) + \beta^1 \\ d^1(2) = \gamma^2 Q(2) + \beta^2 \end{cases} \tag{23}$$

d'où nous déduisons les valeurs des poids. Le processus suivant lequel nous déduisons les valeurs des poids de valeurs tests appliquées au réseau est appelé apprentissage. Suite à cet apprentissage, l'application des poids dans le réseau de neurones engendre des réponse a^σ. Ces réponses s'écartent plus ou moins de la réponse "naturelle". Mais dans le cadre d'une identification, certaines de ces réponses sont en fait acceptables et d'autres non. L'opération d'acceptation, ou de **décision** est assurée par un opérateur A succédant au réseau. Dans le cas précédent nous pourrions accepter toute valeur de a^σ supérieure à une certaine valeur et rejeter toutes celles inférieures à cette même valeur. C'est l'objet de l'opérateur de décision A. La matrice de A est purement orthogonale et ses composantes sont des fonctions sigmoïdes.

$$A_{\nu\sigma} \cdot a^\sigma = [1 + c.exp\,(-u(a^\sigma - d^\sigma))]^{-1} \tag{24}$$

soit que :

$$A_{\nu\sigma} = [1 + c.exp\,(-u(\bullet - d^\sigma))]^{-1} \tag{25}$$

Si $q_\nu = A_{\nu\sigma}a^\sigma$, alors $q_\nu \in [0, 1]$. Le diagnostique peut être rendu plus robuste par l'adjonction d'une seconde couche en série avec la première. Nous pourrions par exemple par cette seconde couche réaliser :

$$\begin{cases} b^1 = \frac{1}{2}(q^1 + q^2) \\ \\ b^2 = 1 - \frac{1}{2}(q^1 + q^2) \end{cases} \tag{26}$$

En posant $R = q^1 + q^2$, $b^1 = R/2$ et $b^2 = 1 - R/2$. Posons $h^1 = 1/2$, $h^2 = -1/2$, $\tau^2 = 1$, $\xi^1 = h^1$, $\xi^2 = h^2 \cdot + \tau^2$, alors :

$$\begin{cases} b^1 = h^1 R \\ \\ b^2 = h^2 R + \tau^2 \end{cases} \tag{27}$$

Pour un stimuli $R(Q)$ nous déterminons les sorties désirées. En faisant correspondre $b^1 \to e^1$, $b^2 \to e^2$, l'apprentissage passe par la définition des stimuli R et des sorties e. Pour deux essais nous pouvons écrire :

$$\begin{cases} e^1(1) = h^1 R(1) \\ \\ e^2(2) = h^1 R(2) \end{cases} \tag{28}$$

de cette essai nous déduisons :

$$h^1 = \frac{1}{2}\left(\frac{e^1(1)}{R(1)} + \frac{e^1(2)}{R(2)}\right)$$

Pour un second essai nous obtenons :

$$\begin{cases} e^2(1) = h^2 R(1) + \tau^2 \\ \\ e^2(2) = h^2 R(2) + \tau^2 \end{cases} \tag{29}$$

Nous obtenons alors :

$$\begin{cases} h^2 = \frac{1}{R(1) - R(2)}\left[e^2(1) - e^2(2)\right] \\ \\ \tau^2 = \frac{1}{R(1) - R(2)}\left[-R(2)e^2(1) + R(1)e^2(2)\right] \end{cases} \tag{30}$$

Nous avons donc élaboré un premier réseau $a^k = y^{k\sigma} p_\sigma$. Ce réseau est suivi d'une décision $q_\alpha = A_{\alpha k} a^k$. Un second réseau analyse les données : $b^\nu = t^{\nu\alpha} q_\alpha$ également suivi d'une décision, $r_\beta = B_{\beta\nu} b^\nu$. Nous avons les définitions des différents opérateurs :

$$y^{k\sigma} = \begin{bmatrix} \gamma^1 \cdot + \frac{\beta^1}{2} & \gamma^1 \cdot + \frac{\beta^1}{2} \\ \gamma^2 \cdot + \frac{\beta^2}{2} & \gamma^2 \cdot + \frac{\beta^2}{2} \end{bmatrix} \tag{31}$$

$$A_{\alpha k} = \begin{bmatrix} \left[1 + c.exp\left(-u(\bullet - d^1)\right)\right]^{-1} & 0 \\ 0 & \left[1 + c.exp\left(-u(\bullet - d^2)\right)\right]^{-1} \end{bmatrix} \tag{32}$$

$$t^{\nu\alpha} = \begin{bmatrix} h^1(\bullet) & h^1(\bullet) \\ h^2(\bullet) + \frac{\tau^2}{2} & h^2(\bullet) + \frac{\tau^2}{2} \end{bmatrix} \tag{33}$$

La relation représentant l'ensemble du RN est : $r_\beta = B_{\beta\nu} t^{\nu\alpha} A_{\alpha k} y^{k\sigma} p_\sigma$.

7 Reconnaissance de deux lettres

Nous appliquons les concepts précédents à la constitution d'un réseau de neurones reconnaissant 2 lettres dans un flux de lettres.

Notre RN simpliste a l'allure :

$$\begin{cases} a^1 = y^{11} x_1 + y^{12} x_2 \\ a^2 = y^{21} x_1 + y^{22} x_2 \end{cases} \tag{34}$$

x_1 et x_2 ont des valeurs particulières pour les deux lettres à reconnaître. Pour ces deux valeurs, par exemple 5 et 20, nous attendons une sortie égale à $1,1$. De fait :

$$\begin{cases} 1 = y^{11} 5 + y^{12} 20 \\ 1 = y^{21} 5 + y^{22} 20 \end{cases} \tag{35}$$

Pour d'autres valeurs, nous attendons 0. Par exemple :

$$\begin{cases} 1 = y^{11} 5 + y^{12} 20 \\ 0 = y^{21} 6 + y^{22} 21 \end{cases} \tag{36}$$

Comprenant le principe, nous généralisons :

$$\begin{cases} 1 = y^{11} 5 + y^{12} 20 \\ 0 = y^{11}(5 + a) + y^{12}(20 + b) \end{cases} \tag{37}$$

$a \neq 0$, $b \neq 0$. Soit finalement :

$$y^{11} = \frac{20+b}{5d-20a} \quad y^{12} = -\frac{5+a}{5b-20a} \tag{38}$$

Le couple $\left(-\frac{21}{5}, -1\right)$ peut convenir et renvoie 1 pour le couple de valeurs recherché et d'autres résultats pour les autres valeurs. Une démarche similaire, bien que "non classique" peut être utilisée pour des données beaucoup plus massives et un groupe de réseaux beaucoup plus important. Nous avons pu programmé le réseau précédent qui détermine avec robustesse les lettres désirées.

Une démarche plus classique consiste à exploiter un algorithme convergent vers un couple de poids adéquat avec l'objectif de reconnaissance.

Nous comparons la sortie d'un RN a^k avec le vecteur des valeurs désirées d^k. L'erreur est définie par :

$$\epsilon^k = a^k - d^k \tag{39}$$

L'apprentissage consiste à minimiser cette erreur, c'est à dire à minimiser un critère défini sur cette erreur $F(\epsilon^k)$. Il est souvent choisi :

$$F(\epsilon^k) = \epsilon_k \epsilon^k$$

La sortie a^k est déterminée, pour une entrée x_m donnée, par les poids ω^{km}. Posons que la variation des poids est une proportion de l'erreur, soit $\Delta\omega^{km} = \alpha^k \epsilon^m$. Si l'attendu est supérieur à la sortie, cela signifie que nous pouvons augmenter la sortie. Donc :

$$\epsilon > 0 \Rightarrow d^k > a^k \Rightarrow \Delta\omega^{km} > 0 \tag{40}$$

Inversement, si l'attendu est inférieur à la sortie, il faut diminuer cette dernière :

$$\epsilon < 0 \Rightarrow d^k < a^k \Rightarrow \Delta\omega^{km} < 0 \tag{41}$$

Partant de $\Delta\omega^{km} = \alpha^k \epsilon^m$ nous pouvons écrire :

$$\epsilon_m \Delta\omega^{km} = \epsilon_m \alpha^k \epsilon^m$$

soit :

$$\epsilon_m = \alpha^k \frac{F}{\Delta\omega^{km}}$$

Nous avons :

$$\frac{F(t+1)}{\Delta\omega^{km}(t+1)} - \frac{F(t)}{\Delta\omega^{km}(t)} = \frac{1}{\alpha^k}\left(\epsilon_m(t+1) - \epsilon_m(t)\right) \tag{42}$$

En notant :

$$\frac{F(t+1)}{\Delta\omega^{km}(t+1)} - \frac{F(t)}{\Delta\omega^{km}(t)} = \nabla F$$

nous obtenons :

$$\nabla F = -\frac{\Delta a_m}{\alpha^k} \tag{43}$$

Nous travaillons à gradient de critère descendant. Nous testons cet algorithme avec le RN suivant :

$$\begin{cases} y^1 = ax_1 + bx_2 + 1 \\ y^2 = dx_1 + fx_2 - 1 \end{cases} \tag{44}$$

A chaque pas de convergence, nous additionnons aux facteurs la moitié de l'écart. Soit par exemple $a(t+1) = a(t) + \epsilon^1/2$. Le listing du programme est le suivant :

```
import numpy as np
import pylab as plt

a = 1.
b = 1.
d = 1.
f = 1.
x1 = 2.
x2 = 1.
y1 = 0.
y2 = 0.
eps1 = 2.
eps2 = 2.

co = 0
y1 = np.zeros(6, dtype=float)
y2 = np.zeros(6, dtype=float)

while eps1 > 0.1 or eps2 > 0.1:
    y1[co] = a * x1 + b * x2 + 1.
    y2[co] = d * x1 + f * x2 - 1.
    eps1 = 3. - y1[co]
    eps2 = 8 - y2[co]
    #
    a = a + eps1 / 2.
    b = b + eps1 / 2.
    d = d + eps2 / 2.
    f = f + eps2 / 2.
    co = co + 1

plt.plot(y1, y2, 'o')
plt.show()
```

La courbe engendré par ce calcul montre les points partant de valeurs décentrées et convergeant vers les deux valeurs solutions au centre. Les valeurs cibles sont 3 et 8 pour les entrées 2 et 1.

8 Perturbation électromagnétique du RN

Si nous écrivons notre réseau comme :

$$\begin{cases} y^\alpha(t) = g^{\alpha\alpha}(t)x_\alpha \\ \\ \epsilon^\alpha(t) = c^\alpha - y^\alpha(t) \\ \\ g^{\alpha\alpha}(t+1) = g^{\alpha\alpha}(t) + \epsilon^\alpha(t) \end{cases} \tag{45}$$

Si au cours de l'apprentissage, qui est la période fragile du RN, une perturbation vient polluer certaines données, par exemple les données captées. La convergence peut être annihilée si ces perturbateurs deviennent trop forts. Notons $\tilde{x}^k$ les perturbateurs en entrées. Nous avons alors :

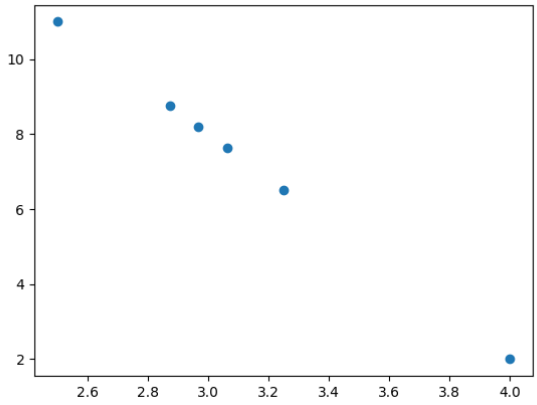

FIGURE 3 – Convergence par gradient décroissant

$$\begin{cases} y^{\alpha}(t) = g^{\alpha\alpha}(t)\,(x_{\alpha} + \tilde{x}_{\alpha}(t)) \\ \epsilon^{\alpha}(t) = c^{\alpha} - y^{\alpha}(t) \\ g^{\alpha\alpha}(t+1) = g^{\alpha\alpha}(t) + \epsilon^{\alpha}(t) \end{cases} \tag{46}$$

Les données polluées $\tilde{x}^k$ peuvent être créées par une interaction électromagnétique avec le capteur ou toute perception qui fournit ces données. Par définition d'un bruit nous les considérons non constantes. Comment peut évoluer une telle structure ? Si au premier instant l'écart est positif, la valeur suivante du poids va décroître. On s'attend à ce que le pas suivant, la valeur calculée soit plus proche de la valeur attendue. Mais si lors de la comparaison suivante, la valeur d'entrée est perturbée, il se peut que la valeur suivante de la sortie s'éloigne encore de l'attendu.

Nous comprenons que si la perturbation est maintenue avec une valeur fluctuante, le processus peut ne jamais converger.

Une autre possibilité est que la perturbation soit fixe pendant le temps d'apprentissage et crée un biais dans le calcul des poids. Nous avons vu que la convergence est très rapide, typiquement 2 à 4 temps horloge. Soit pour un processeur cadencé à 100 MHz (très courant), de l'ordre de 30 ns. Pendant ce temps très court, le perturbateur peut très vraisemblablement être continument présent. La perturbation des données d'entrée va donc être impactée pendant tout le cycle de lecture pour apprentissage. Si les données d'entrée étaient 2 et 1, avec perturbation elles peuvent être modifiées à 2,5 et 1,5 par exemple. Or cette déviation qui peut sembler mineure, conduit le RN à diverger vers des valeurs incohérentes. Le point remarquable est que l'issue de la perturbation n'est pas prévisible linéairement. Si nous avions une fonction de type amplificatrice, la sortie dériverait de façon prévisible avec un bruitage de l'entrée. Ici il n'en est rien. Le bruitage des données pendant l'apprentissage conduit à former un RN qui n'est plus à même de remplir sa mission, mais sans que l'on puisse maîtriser la déviation par rapport à toutes les données recevables en entrée. Le plus

dangereux du point de vue de l'utilisateur étant que l'impact de la déviation n'est pas forcément détectable sur un ensemble de motifs. En effet une autre considération est l'influence de l'opérateur de décision.

Soit un bruit $\tilde{x}$, et une fonction de décision

$$\left[1 + ce^{-a(x-x_0)}\right]^{-1}$$

Cette fonction est perturbée suivant :

$$\left[1 + ce^{-a(x+\tilde{x}-x_0)}\right]^{-1}$$

Si $x + \tilde{x} > x_0$ alors que $x < x_0$, le bruit fait basculer la décision. Du fait de la fonction exponentielle, la valeur de bruit nécessaire pour faire basculer la décision peut être faible, suivant la pente a. La limite est obtenue pour :

$$\left[1 + ce^{-a(x+\tilde{x}_0-x_0)}\right]^{-1} = 0,5 \Rightarrow \tilde{x}_0 = -\frac{1}{a} ln\left(\frac{1}{c}\right) + (x_0 - x) \tag{47}$$

La valeur $\tilde{x}_0$ peut être considérée comme seuil de susceptibilité du RN, x étant la valeur la plus haute du RN qui approche x_0 sans l'atteindre et avant basculement.

9 Réseaux de neurones basés sur des résonateurs

Un ensemble p_q de stimuli comme un ensemble de raies spectrales constitue une information. Cette information est un groupe de mots, chaque raie étant un mot. En reprenant une structure de réseaux de neurones où chaque raie est dirigée vers un ensemble de filtres de bandes étroites, cela revient à présenter à l'entrée de chaque filtre :

$$\sum_n A_n Cos(n\omega_0 t) \tag{48}$$

A ce signal temporel, un filtre de bande étroite sélectionne une composante. Si la fonction de transfert de ce filtre est :

$$R\frac{e}{R + Lp + 1/Cp} = \frac{Re}{R_n + j\left(L_n\omega - \frac{1}{C_n\omega}\right)} \tag{49}$$

Or

$$e = \sum_n A_n \frac{p}{p^2 + n^2\omega_0^2}$$

et donc pour la sortie S_n du filtre n :

$$S_n = \sum_n A_n \frac{p}{p^2 + n^2\omega_0^2}\left(\frac{R_n}{R_n + L_n p + \frac{1}{C_n p}}\right) \tag{50}$$

soit :

$$|S_n| = \frac{A_n R_n \omega}{(n^2\omega_0^2 - \omega^2)\sqrt{R_n^2 + \left(L_n\omega - \frac{1}{C_n\omega}\right)^2}} \tag{51}$$

Par exemple si le premier filtre est accordé à $\omega = \omega_O + \epsilon$ avec $\omega = 1/\sqrt{L_1 C_1}$ alors :

$$|S_1| = \frac{A_1 \omega_0}{\epsilon} \tag{52}$$

alors que

$$|S_i|\,,\ i \neq 1 << |S_1| \tag{53}$$

Le réseau de neurones agit comme un banc de filtres et chaque sortie sélectionne principalement une composante du spectre d'entrée. Si maintenant un mot particulier est représenté par un certain ensemble de raies du spectre lu, nous pouvons avoir une couche de décision qui engendre des 1 sous la condition que les sorties des filtres soient des valeurs fortes ou faibles suivant le contenu spectral recherché. Une seconde couche par le jeu de cœfficients présentera en sortie une combinaison qui sera le code attendu pour le spectre à détecter.

L'avantage du RN basé banc de filtres est qu'il est plus difficile de modifier la fréquence d'un signal par perturbation que de modifier son amplitude. Ici la couche de décision peut être réglée pour que la plage d'amplitude autorisée soit assez large. A contrario, pour qu'un perturbateur externe change la fréquence d'un motif il faut qu'un processus non linéaire intervienne, à moins qu'une composante spectrale soit éliminée, ce qui est peu probable. Autrement dit, si e est bruité nous avons :

$$A_n \rightarrow A_n + \tilde{A}_n$$

et

$$|S_n| = \frac{\left(A_n + \tilde{A}_n\right) R_n \omega}{\left(n^2\omega_0^2 - \omega^2\right)\sqrt{R_n^2 + \left(L_n\omega - \frac{1}{C_n\omega}\right)^2}} \tag{54}$$

De fait :

$$|S_1| = \frac{\left(A_1 + \tilde{A}_1\right)\omega_0}{\epsilon} \tag{55}$$

Si la décision règle son seuil t de telle façon que $exp\left(-a\left[\left(A_n + \tilde{A}_n\right) - t\right]\right) \rightarrow 1$, l'influence du bruit est absorbée.

10 Conclusion

Ces premières ébauches de réflexion tendent à montrer que les réseaux de neurones "classiques" peuvent présenter des fragilités critiques vis à vis de perturbateur qui viendraient à modifier les données d'apprentissage. A contrario, des réseaux de neurones basés sur d'autres mécanismes comme des filtres, pourraient peut-être s'avérer beaucoup plus robustes à ce type d'agressions. Sur la base de ces premiers modèles simplistes il reste à étendre les topologies abordées à des dimensions réalistes. Par ailleurs parmi les agresseurs les plus dangereux vis à vis des effets recherchés, des sources de champs électromagnétiques modulées profitant des effets de détection parasite sont a priori les sources de bruit à étudier en priorité. Ces sources peuvent provenir de diverses électroniques embarquées comme des smartphones, tablettes, mais aussi des émetteurs de puissance comme les radar, etc.