République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOHAMED KHIDER, BISKRA

FACULTÉ des SCIENCES EXACTES et des SCIENCES de la NATURE et de la VIE

DÉPARTEMENT DE MATHÉMATIQUES

Mémoire présenté en vue de l'obtention du Diplôme **:**

MASTER en Mathématiques

Option : **Analyse**

Par

Selmi Bouthaina

Titre :

Calcul de la valeur propre dominante et le vecteur propre correspondant d'une matrice carrée symétrique à l'aide de réseaux de neurones artificiels

Membres du Comité d'Examen :

Dr.	Amrane Houas	UMKB	Président
Pr.	Naceur Khelil	UMKB	Encadreur
Dr.	Asia Senoussi	UMKB	Examinateur

Juin 2025

Dédicace

À la source de mon existence, à celui dont la sagesse éclaire mes pas et dont la force m'a appris à ne jamais baisser les bras, à mon cher père Selmi Amar, et à celle dont l'amour inconditionnel est mon refuge, à ma douce et précieuse mère Khoudrane Ghania, je vous dois tout.

Ce travail est le fruit de vos sacrifices, de votre patience et de votre foi en moi. À mon fils adoré, Abou Bakr Esseddik, à celui que j'ai tant espéré tenir entre mes bras, à celui dont l'absence pèse mais dont le souvenir illumine chacun de mes pas. J'aurais tant souhaité partager avec toi cette joie, mon enfant... Ce travail, je te le dédie avec tout l'amour d'une mère, pour toujours.

À mes frères bien-aimés : Abdallah, Moutia, et Ayoub, et à mes sœurs chéries : Dhikra et Hadil, merci d'avoir été ma force silencieuse, ma source de rires et de réconfort.

À mes grands-parents, qui m'ont portée dans leurs prières et dont l'affection m'a enveloppée comme un doux manteau.

À toute ma famille, véritable pilier de mon parcours, je vous offre ce modeste accomplissement en signe de reconnaissance et d'amour.

À mes amies de cœur, celles qui ont su faire briller la lumière même dans mes jours sombres, et tout particulièrement à Moussi Radhia,

confidente fidèle, âme précieuse, qui a su être présente au bon moment, avec les mots justes.

REMERCIEMENTS

Je tiens tout d'abord à remercier Pr. Khelil Naceur à l'Université de Biskra, qui m'a donné ce sujet et m'a encadré pendant mon Master. Il a toujours été à mon écoute et son point de vue complémentaire est souvent été trés utile.

Je suis trés reconnaissante envers Dr. Amrane Haous et Dr.Senoussi Assia d'avoir manisfesté de l'interêt pour mon travail en acceptant de participer à ce jury. Leurs remarques et commentaires constructifs m'ont permis d'en améliorer le manuscrit.

Enfin, que toutes les personnes qui ont partcipé de prés ou de loin à la réalisation de ce travail trouvent ici l'expression de mes sincères remerciements.

Mes derniers remerciements vont à mes parents, mes frères et soeurs et tous les amies.

ملخص

في هذه الرسالة، تُقدّم شبكة عصبية وتُستخدَم لحساب القيمة الذاتية السائدة والمتجه الذاتي المرتبط بها لمصفوفة مربعة متماثلة. صُمِّمت هذه الشبكات العصبية لقدرتها على التعامل مع دوال الخسارة المُخصّصة وحساب التقديرات للمسائل واسعة النطاق.

RESUME

Dans ce mémoire de master, un réseau neuronal est introduit et utilisé pour calculer la valeur propre dominante et le vecteur propre associé d'une matrice carrée symétrique. Ces réseaux neuronaux sont conçus pour leur capacité à traiter des fonctions de perte personnalisées et à calculer des estimations pour des problèmes à grande échelle.

ABSTRACT

In this master thesis, a neural network is introduced and utilized to calculate the dominant eigenvalue and related eigenvector of a symmetric square matrix. These neural networks are built because of their capacity to deal with custom loss functions and compute estimations for large-scale problems, these neural networks are constructed.

Table des matières

Table des figures

Introduction

Le calcul des valeurs propres et des vecteurs propres associés d'une matrice réelle donnée est nécessaire dans de nombreuses disciplines scientifiques. Ce calcul est important pour les problèmes scientifiques et techniques tels que le traitement des signaux, la théorie du contrôle et la géophysique [4]. Les solutions générales des systèmes d'équations différentielles nécessitent souvent la connaissance des quantités spectrales, c'est-à-dire les vecteurs propres et les valeurs propres. En outre, la signification de la matrice de covariance en statistique est plus claire lorsque les paires propres sont connues.

Outre les méthodes standard de calcul des valeurs propres et de leurs vecteurs propres, le calcul des paires propres à l'aide de techniques neuronales suscite un grand intérêt
Une valeur propre est l'une des valeurs spéciales d'un paramètre dans une équation particulière pour laquelle l'équation a une solution. Plus précisément, la valeur propre non triviale.
Les solutions de l'équation $Ax = \lambda x$ ont été introduites par Lagrance en 1762 pour résoudre des systèmes d'équations différentielles à coefficients constants. Les solutions non nulles sont valeurs propres, et le terme a été introduit par Hilbert en 1904 pour désigner une propriété des équations intégrales. Plus tard, les valeurs propres ont été rattachées aux matrices [5]. Dans le cas d'une équation différentielle, une solution monovaleur, finie et continue n'est trouvée que pour des valeurs particulières d'un paramètre et ce sont les valeurs propres de l'équation différentielle. Des définitions mathématiques détaillées sont données à la section 1.

Plusieurs méthodes peuvent être utilisées pour calculer les solutions de l'équation $Ax = \lambda x$. Une des options consiste à utiliser la méthode de la puissance ou les solveurs de Matlab préexistants. Dans ce mémoire, nous proposons une deuxième option, qui construit des réseaux de neurones. Ces réseaux de neurones sont créés en raison de la capacité à travailler avec des fonctions de perte personnalisées et calculer des estimations de la valeur propre dominante λ pour des problèmes à grande échelle.

Ainsi, le but de ce mémoire est de calculer la valeur propre dominante et le vecteur propre associé avec un réseau de neurones et discuter des avantages de l'utilisation des réseaux de neurones par rapport aux solveurs habituels.

Ce mémoire commence par une brève introduction .

Le reste de ce mémoire est structuré comme suit. Dans le chapitre 1 une brève introduction aux valeurs propres et vecteurs propres associés. Et nous présenterons aussi le quotient de Rayleigh.

Dans le deuxième chapitre 2 nous metons en evidence l'interet de rechercher et d'étudier comment les structures, comme les réseaux neuronaux, peuvent résoudre des problèmes tels que le calcul des valeurs propres et des vecteurs propres correspondants. Selon de nombreux chercheurs, l'informatique neuronale définie par des systèmes dynamiques est très prometteuse pour résoudre les problèmes de calcul en temps réel. Et on examine les travaux anterieurs .

Le chapitre 3, s'ocupe de la description et l'implémentation neuronale, propose une représentation intéressante des solutions du réseau. La convergence du réseau est mis en évidence par un exemple et une discussion.

Enfin, nous concluons par une conclusion dans la section.

Chapitre 1

Valeurs et Vecteurs propres

1.1 Valeurs et vecteurs propres

Les valeurs propres et les vecteurs propres d'une matrice A interviennent dans des nombreux problèmes (équations differentielles, diagonalistion des matrices,...). Il est donc intéressant ici d'indiquer quelques algorithmes permettant de les obtenir numériquement. Auparavant faisons quelques rappels [2] :

Définition 1.1.1 *Les valeurs propres λ et les vecteurs propres $X \neq 0$, d'une matrice A vérifient* $\boxed{AX = \lambda X}$

Nous pouvons savoir si le problème soulevé par cette définition a une solution, si nous le réécrivons $(A - \lambda I)X = 0.$, ainsi le problème devient un système linéaire homogène $BX = 0$, que nous savons qu'il possède une solution unique $X = 0$ lorsque $det(B) \neq 0$. Justement c'est le cas qui ne nous intéresse pas.

λ est appelée valeur propre de A (matrice carrée) si et seulement si $\det(A - \lambda I)X = 0$, c'est l'équation caractéristique de la matrice A et sera notée $\phi(\lambda) = 0$.

$\phi(\lambda)$ est appelé polynôme caractéristique il est de degré n,

Exemple 1.1.1 *Calculer les vecteurs et les valeurs propres de* $A = \begin{bmatrix} 4 & -5 \\ 2 & -3 \end{bmatrix}$

$\det(A-\lambda I) = \det\begin{bmatrix} 4-\lambda & -5 \\ 2 & -3-\lambda \end{bmatrix} = (4-\lambda)(-3-\lambda)+10 = \lambda^2-\lambda-2 = 0 \Rightarrow \lambda_1 = -1$ et $\lambda_2 = 2$.

Les vecteurs propres sont :

- pour λ_1, se calculent de $(A-\lambda_1 I)X = 0 \Leftrightarrow \begin{bmatrix} 5 & -5 \\ 2 & -2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.Le système obtenu a une infinité de solutions de la forme $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, par exemple $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ est vecteur propre correspondant à $\lambda_1 = -1$.
- pour λ_2, se calcule de $(A-\lambda_1 I)X = 0 \Leftrightarrow \begin{bmatrix} 2 & -5 \\ 2 & -5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.De nouveau le système obtenu a une infinité de solutions de la forme $\begin{bmatrix} x_1 \\ 0.4x_1 \end{bmatrix}$, par exemple $\begin{bmatrix} 5 \\ 2 \end{bmatrix}$ est vecteur propre correspondant à $\lambda_2 = 2$.

Remarque 1.1.1 *De la défintion on a : $AV = VD$, où $D = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ et $V = (X_1, X_2, ..., X_n)$ les vecteurs propres correspondants.*

Il facile de démontrer les propriétés suivantes

- $\lambda_1 + \lambda_2 + ... + \lambda_n = tr(A)$ (car $tr(AB) = tr(BA)$)
- $\lambda_1\lambda_2...\lambda_n = \det(A)$ (car $det(A) = det(VDV^{-1})$)
- Les valeurs propres d'une matrice triangulaire supérieure ou inférieure sont les eléments de la diagonale.
- Les vecteurs propres sont linéairement indépendants (par récurrence)

1.1.1 Quotient de Rayleigh

Définition 1.1.2 *Soit A une matrice symétrique réelle de taille $n \times n$ (mais la définition fonctionne aussi pour les matrices complexes hermitiennes), et $x \in R^n - \{0\}$ un vecteur non nul. Le quotient de Rayleigh de x par rapport à A est défini par :* $\boxed{R_A(x) = \frac{x^T Ax}{x^T x},\ x \neq 0}$

Remarque 1.1.2 *Remarquons que Si x est un vecteur propre de A, alors $R_A(x)$ donne exactement la valeur propre associée. en particulier, pour v_1 le vecteur propre correspondant à λ_1 la valeur propre dominante* $\max_{x\neq 0} R_A(x) = \max_{x\neq 0} \frac{x_1^T A x_1}{x_1^T x_1} = \frac{v_1^T \lambda_1 v_1}{v_1^T v_1} = \lambda_1$, *de plus tout multiple par un scalaire non nul du vecteur v_1 est également un maximiseur. Par souci de simplicité, nous nous concentrons sur les vecteurs propres de norme unitaire comme maximiseurs* $\max_{x_1^T x_1=1} R_A(x) = \max_{x^T x=1} x^T A x$, *car* $R_A(kx) = \frac{(kx)^T A(kx)}{(kx)^T (kx)} = \frac{x^T A x}{x^T x} = R_A(x)$.

Il suffit donc de se concentrer sur la sphère unitaire dans $R^n . x \in R^n = \{x \in R^n, \|x\| = 1\}$ sur laquelle le quotient de Rayleigh se réduit à $R/_{S_n} = x^T A x, x \in R^n - \{0\}$

Pour une matrice symétrique A de valeurs propres $\lambda_{min} \leq .... \leq \lambda_{max}$, de la définition de $R_A(x)$, on a :

$\lambda_{min} \leq R_A(x) \leq \lambda_{max}$, c'est-à-dire, les valeurs propres de A peuvent être caractérisées comme les extrémums du quotient de Rayleigh.

Soit :

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}, x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Calcul du quotient de Rayleigh :

$$R_A(x) = \frac{x^T A x}{x^T x} = \frac{[1\ 1]\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}}{[1\ 1]\begin{bmatrix} 1 \\ 1 \end{bmatrix}} = \frac{[1\ 1]\begin{bmatrix} 3 \\ 4 \end{bmatrix}}{2} = \frac{7}{2} = 3.5$$

Donc, $R_A(x) = 3.5$, qui est une approximation entre les deux valeurs propres réelles de A.

Chapitre 2

Utilisation de réseaux neuronaux

2.1 Utilisation de réseaux neuronaux pour calculer les paires propres

Les réseaux neuronaux artificiels (RNA) constituent une part croissante de l'étude de l'intelligence artificielle et visent à un lien avec les véritables machines biologiques [6]. Pour construire des machines intelligentes, le modèle naturel est le cerveau humain. À cette fin, l'une des premières choses qui vient à l'esprit est de simuler le fonctionnement du cerveau directement sur un ordinateur. Les ordinateurs d'aujourd'hui ont des capacités remarquables, notamment celle de stocker de grandes quantités d'informations et d'effectuer des calculs arithmétiques poussés sans erreur.

Leurs circuits fonctionnent "très vite" et les humains ne peuvent pas approcher de telles capacités [6].

La nécessité d'un processeur ayant la fonctionnalité du cerveau humain et la vitesse d'un ordinateur a attiré et attire encore de nombreux chercheurs vers les réseaux neuronaux artificiels (ANN). Un réseau neuronal artificiel est une machine ou un algorithme modelé sur la conception et le fonctionnement cerveau. Dans l'ensemble, les architectures de réseaux neuronaux ne sont pas destinées à reproduire fonctionnement du cerveau humain,

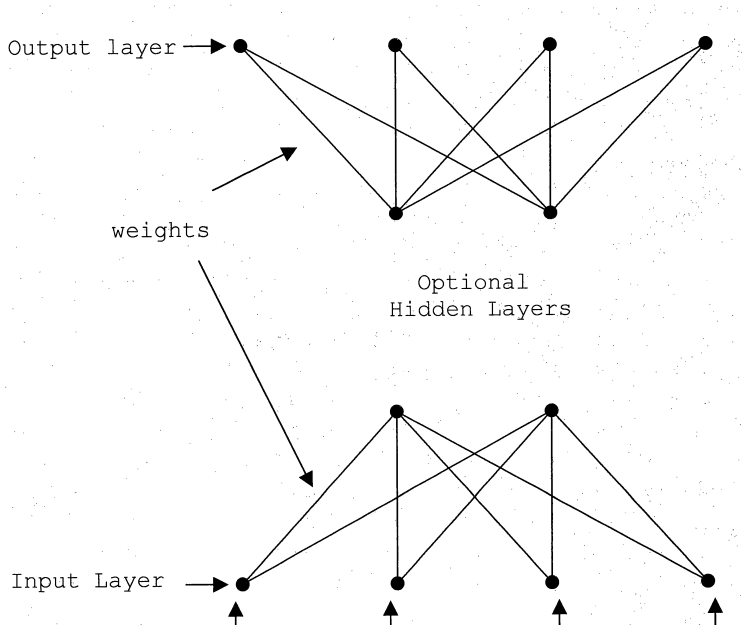

FIG. 2.1 – Réseau neuronal feedforward simple

mais à s'inspirer de faits connus sur le fonctionnement du cerveau [6].

En général, un réseau se compose de nombreux processeurs simples, également appelés nœuds ou neurones, qui sont reliés entre eux par des couches. Il existe des couches d'entrée et de sortie, chacune contenant un nombre quelconque de nœuds. Comme l'illustre la figure 2.1, il peut y avoir un certain nombre de couches cachées séparant l'entrée de la sortie, contenant également un nombre arbitraire de nœuds. Chaque nœud contient une petite quantité de données et chaque lien entre les nœuds est associé à une valeur (poids), comme le montre la figure 2.1. Le concept de machine biologique découle de l'idée que les nœuds d'entrée sont équivalents aux neurones et que les liens sont équivalents aux synapses et aux axones.

Le réseau est formé de manière à ce que les poids soient modifiés jusqu'à ce que l'ANN, pour une entrée donnée, produise la sortie correcte ou la plus correcte possible. Cette formation peut se faire par apprentissage **supervisé** ou **non supervisé**. Un ANN fait l'objet d'un apprentissage supervisé lorsque les vecteurs d'entrée et les vecteurs de sortie correspondants sont utilisés. D'une certaine manière, il y a un enseignant qui guide le

réseau vers la sortie correcte.

L'apprentissage dans les réseaux supervisés est réalisé par une méthode appelée rétro-propagation. La différence entre les sorties souhaitées et les sorties réelles du réseau est observée, puis le réseau est modifié et le résultat de la rétro-propagation est obtenu.

La procédure se répète jusqu'à l'obtention de résultats corrects. Ainsi, le réseau neuronal minimise une fonction d'erreur de la sortie. Malheureusement, la rétropropagation présente des problèmes. Premièrement, elle est lente, deuxièmement, il est difficile d'analyser les actions des couches cachées et, enfin, les résultats ne sont pas toujours obtenus en raison des faiblesses de la méthode de descente du gradient (c'est-à-dire que les minima locaux peuvent perturber la descente du gradient) [7].

Dans l'apprentissage non supervisé, il n'y a pas d'enseignant, mais le réseau neuronal incorpore des informations locales et des règles internes pour associer différentes entrées aux différentes sorties. Cela le rend plus proche du fonctionnement du cerveau, qui n'a pas d'enseignant interne. L'apprentissage non supervisé convient mieux "aux situations où y a beaucoup de redondance dans les données d'entrée". Par la répétition, le réseau s'organise pour distinguer des modèles ou des caractéristiques dans les données [8].

Il est intéressant de rechercher et d'étudier comment les structures, comme les réseaux neuronaux, peuvent résoudre des problèmes tels que le calcul des valeurs propres et des vecteurs propres correspondants. Selon de nombreux chercheurs, l'informatique neuronale définie par des systèmes dynamiques est très prometteuse pour résoudre les problèmes de calcul en temps réel [9]-[4].

2.2 Examen des travaux antérieurs

Dans la recherche non supervisée, un réseau neuronal doit découvrir par lui-même des modèles, des régularités, des caractéristiques, des corrélations ou des catégories de données d'entrée et les coder dans la sortie [7]. En découvrant ces éléments, le réseau modifie ses

paramètres un processus appelé auto-organisation [7].

Supposons que nous ayons un vecteur d'entrée avec des composantes" et que chaque composante soit associée d'un poids. Si nous considérons le cas le plus simple d'une seule unité linéaire, sa sortie scalaire V est

$$V = \sum w_i \xi_i = w^T \xi = \xi w^T \tag{2.1}$$

où w est le vecteur de poids. L'architecture du réseau est présentée à la figure 2.2 . L'apprentissage hebbien, un mécanisme d'apprentissage fondamental [7], est représenté par cette règle d'apprentissage :

$$\Delta w_i = \eta V \xi_i \tag{2.2}$$

où η est le taux d'apprentissage, une petite constante positive. Le produit $V\xi_i$ est la règle de Hebb standard et est présent sous une forme ou une autre dans de nombreuses règles d'apprentissage (section 2.1).

Le problème est que les poids continuent à "croître sans limite et que l'apprentissage ne s'arrête jamais" [7]. Pour éviter cela, [10] a ajouté une décroissance des poids proportionnelle au carré de la sortie V^2 à la règle de Hebbian classique

$$\Delta w_i = \eta V \left(\xi_i - V w_i\right) \tag{2.3}$$

La règle d'Oja ci-dessus fait converger vecteur de poids vers le vecteur propre qui correspond à la plus grande valeur propre max de la matrice C, la matrice de covariance de l'ensemble de données [10].

Plusieurs chercheurs ont étendu la règle d'Oja à des réseaux de neurones multiples qui extraient tous les vecteurs propres de la matrice de covariance C d'un ensemble de vecteurs

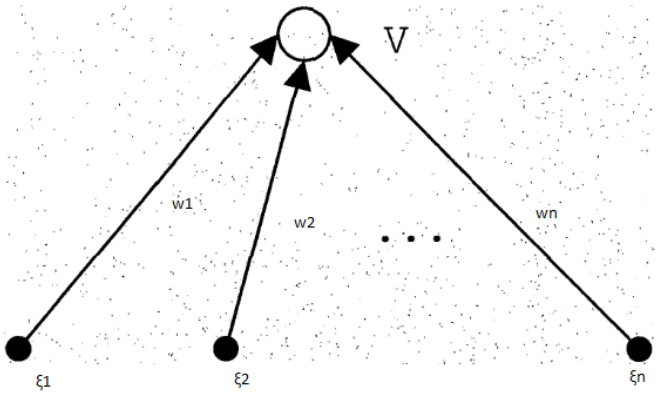

FIG. 2.2 – Architecture pour l'apprentissage hebbien simple

d'entrée donné.[7]- [11]-[12]. Règle de Sanger [12] ; par exemple, les projets les sorties d'un vecteur ζ d'entrée sur l'espace des M premières composantes principales. La règle mise à jour est

$$\Delta w_{ij} = \eta V_i \left(\xi_i - \sum_k V_k w_{ki} \right) \tag{2.4}$$

Cette règle est le plus souvent utilisée dans les applications car elle est robuste et permet d'extraire les composantes principales individuellement dans l'ordre [7].
Georgiou et Tsai ont abordé le problème de la recherche des vecteurs propres d'une matrice symétrique positive définie (avec des réseaux neuronaux) d'une nouvelle manière [9]. Les données ayant approximativement une matrice de covariance spécifique (la matrice donnée) sont générées aléatoirement, puis l'architecture neuronale et l'algorithme APEX sont utilisés pour extraire vecteurs propres [8].

Dans les études susmentionnées appliquent des règles d'apprentissage à la matrice de covariance des vecteurs de données (d'entrée) et les valeurs propres et vecteurs propres

extraits sont ceux de la de covariance de la matrice de covariance. Dans ce mémoire, le problème direct est étudié : **étant donné une matrice A, trouver toutes les valeurs propres et les vecteurs propres associés de A.**

Dans [13], une méthode dynamique produisant des estimations vecteurs et valeurs propres réels a été présentée. La technique proposée est appliquée pour estimer les spectres propres de formes k réelles à n dimensions. Leur approche "est basée sur une propriété d'épissage spectral des manifolds linéaires souvent trouvés dans les solutions d'équations différentielles polynomiales".

Dans [4] un système dynamique pour calculer les vecteurs propres associés d'une matrice A définie positive est décrit par la règle :

$$\frac{dx}{dt} = Ax - f\left(x\right)x \tag{2.5}$$

où $x = (x_1, x_2, ...x_n,)^T$ et la fonction $f(x)$ satisfait à certaines hypothèses [4]. Comme il est mentionné dans le même document, le premier terme du côté droit de l'équation 2.5 peut être considéré comme le terme standard de la règle de Hebb (équation 2.1)et le second terme a pour effet de limiter la longueur du vecteur x [4].

Chapitre 3

Solution neuronale

3.1 Formulation du problème

Soit A une matrice symétrique réelle de taille $n \times n$. Considérons le problème de maximisation sous contrainte [1]-[3] .

$$\begin{cases} \max \left(x^T A x\right)^2 \\ x^T x = 1 \end{cases} \tag{3.1}$$

Nous pouvons résoudre ces problèmes d'optimisation en utilisant la méthode du multiplicateur de Lagrange. Soit λ un multiplicateur de Lagrange. Le problème revient alors à maximiser $E(x)$:

$$E(x) = \frac{1}{2}\left(x^T A x\right)^2 - \lambda(x^T x - 1) \tag{3.2}$$

On peut utiliser la descente de gradient pour minimiser $-E(x)$. Le gradient de $E(x)$ par rapport à x est :

$$\nabla_x E = -2\left(x^T A x\right) A x + 2\lambda x$$

À l'équilibre, $\nabla_x E = 0$, donc

$$-2\left(x^T Ax\right) Ax + 2\lambda x = 0$$

Multiplions à droite par x^T,

$$-\left(x^T Ax\right)\left(x^T Ax\right) + \lambda x^T x = 0$$

or

$$\lambda = \left(x^T Ax\right)^2$$

par conséquent, nous écrivons

$$\nabla_x E = -2\left(x^T Ax\right)(Ax - \lambda x)$$

Le gradient ci-dessus peut être écrit sous forme de système dynamique :

ou comme règle d'apprentissage :

$$\Delta_t x = \eta\left(x^T Ax\right)\left(Ax - \left(x^T Ax\right)x\right) \tag{3.3}$$

3.2 Simulation par ordinateur

3.2.1 Description de l'implémentation neuronale

La recherche donnée se concentre principalement sur l'équation différentielle classique du réseau neuronal, comme indiqué dans l'équation 3.3, où $A = (a_{ij}), i, j = 1, 2, \ldots, n$ est une matrice symétrique qui nécessite le calcul de sa valeur propre dominante et de leur vecteur propre associé, $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]$ est un vecteur colonne représentant les états des neurones du système dynamique du réseau neuronal, ainsi que les éléments d'une matrice carrée symétrique A représentant les poids des connexions entre ces neurones.
En pratique, nous avons besoin d'un vecteur colonne non nul,

$$x(0) = [x_1(0), x_2(0), \ldots, x_n(0)]$$

pour démarrer le système de réseau neuronal en utilisant la règle de mise à jour suivante :

$$x\left(t+1\right) = \eta\left(x\left(t\right)^T Ax\left(t\right)\right)\left(Ax\left(t\right) - \left(x\left(t\right)^T Ax\left(t\right)\right)x\left(t\right)\right)$$

où t désigne l'itération à t et η est un petit pas de temps.

L'itération se termine lorsque

$$\left\|x\left(t+1\right) - x\left(t\right)\right\| \prec \epsilon$$

où ϵ est une petite erreur de contrainte qui peut être prédéterminée. on pourrait considérer que $\left\|x\left(t+1\right) - x\left(t\right)\right\| = 0$, c'est-à-dire, $Ax\left(t\right) = \left(x\left(t\right)^T Ax\left(t\right)\right)x\left(t\right)$, Selon la théorie de la section 3, $x(t)$ est le vecteur propre correspondant à la valeur propre dominante, qui peut s'écrire $x\left(t\right)^T Ax\left(t\right)$.

3.2.2 Exemple et discussion

Cette section présente un exemple de résultats de simulation informatique pour illustrer la théorie précédente. La simulation démontrera que le réseau proposé permet de calculer le vecteur propre correspondant à la valeur propre dominante de toute matrice symétrique. De manière simple, une matrice symétrique A pourrait être générée aléatoirement. Soit B une matrice réelle générée aléatoirement, et définissons $A = B.B^T$ est évidemment une matrice symétrique.

$$A = \begin{bmatrix} 2.8966 & 2.1881 & 1.1965 & 2.1551 \\ 2.1881 & 1.9966 & 0.6827 & 1.8861 \\ 1.1965 & 0.6827 & 0.7590 & 0.5348 \\ 2.1551 & 1.8861 & 0.5348 & 2.0955 \end{bmatrix},$$

Pour laquelle nous calculons la plus grande valeur propre en se servant du code MATLAB suivant.

```
function [c, v] = evann3()
% Génère une matrice symétrique positive définie
B = rand(4);
A = B * B'; % Matrice symétrique positive définie
% Initialisation du vecteur
x = rand(4,1);
x = x / norm(x); % Normalisation
eta = 0.01; % Pas d'apprentissage
iterations = 50;
valeurs_propres = zeros(iterations, 1); % Stocker les valeurs approchées
% Itérations
for i = 1 :iterations
lambda = x' * A * x;
grad = A*x - lambda*x;
x = x + eta * lambda * grad;
x = x / norm(x); % Normalisation
valeurs_propres(i) = lambda;
end
% Résultats finaux
c = x' * A * x; % Valeur propre approchée
v = x; % Vecteur propre associé
% Tracé de la convergence
figure;
plot(1 :iterations, valeurs_propres, 'b-', 'LineWidth', 2);
hold on;
vrai_lambda_max = max(eig(A));
yline(vrai_lambda_max, 'r–', 'Ligne de référence');
```

```
xlabel('Itérations');
ylabel('Valeur propre approchée');
title('Convergence vers la plus grande valeur propre');
legend('Valeur propre approchée', 'Valeur propre réelle max');
grid on;
end
```

Lors de l'utilisation du modèle de réseau, la valeur propre dominante estimée est $\lambda_{\max} = 6.8648$

et le vecteur propre correspondant est : $v_{\max} = [0.6380\ 0.5202\ 0.2287\ 0.5197\quad]^T$.

Comme le montre la figure 3.11, la convergence du calcul de $\lambda(t)$ vers $\lambda_{\max}$, synthétisée par la formule de mise à jour des pondérations 3.3, est évidente. Après 6 itérations, $\lambda(t)$ dans le code MATLAB ci-dessus, pourrait converger vers 6.8648, la valeur propre dominante de A.

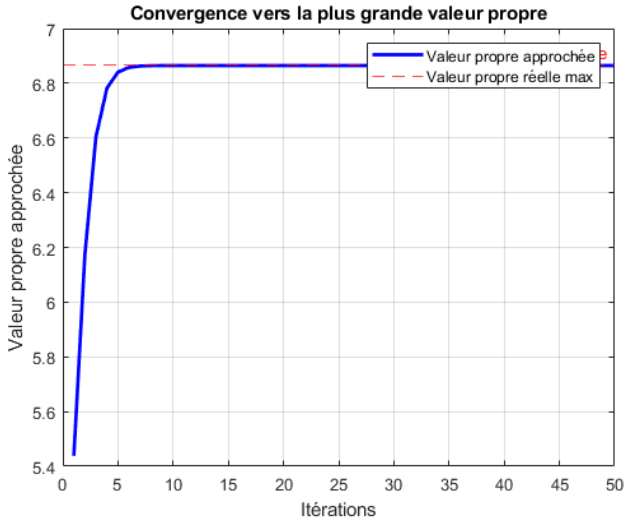

FIG. 3.1 – Estimation de la valeur propre dominante de la matrice A.

Conclusion

Ce mémoire examine la question de la recherche des vecteurs propres d'une matrice carrée symétrique à l'aide d'une approche par réseau de neurones. Il a été suggéré qu'une classe de réseaux de neurones artificiels récurrents puisse être utilisée pour calculer le vecteur propre correspondant à la valeur propre dominante de toute matrice carrée symétrique réelle. Ce concept de réseau offre d'excellentes performances de calcul car il prend en charge le traitement parallèle asynchrone. La connaissance mathématique des comportements dynamiques du modèle de réseau est rigoureuse et sans ambiguïté. D'après les résultats de simulation, le modèle de réseau est efficace.

Bibliographie

[1] William W. Hager, “Minimizing a quadratic over a sphere”, SIAM Journal on Optimization, vol. 12, no.1, (2001) , pp. 188–208.

[2] N. Khelil, H. Khelil, L. Djerou, Analyse numérique 2 , ISBN 978-3-8416-2306-5, Editions Universitaires Europeennes. Site Web : https ://www.morebooks.de/

[3] H. Khelil, N. Khelil and L. Djerou, Towards the dominant eigenvalue and the corresponding eigenvector of a symmetric square matrix using neural networks . YMER , pp.435-439, doi : 10.37896/YMER22.09/37.

[4] Q. Zhang and Z. Bao, "Dynamical System for computing, the eigenvectors associated with the largest eigenvalue of a positive definite matrix", IEEE Transactions on Neural Networks, vol. 6, pp. 790,-791, 1995.

[5] D. Hilbert, "Grundzuge einer-allgemeinen Theorie,der linearen Intergralgleicungen (Foundations of a General Theory of Linear Integral Equations)", B. G. Teubner, Berlin, I9I2.

[6] E. Rich and K. Knight, Artificial, Intelligence, 2^{nd} edition, McGraw-Hill, New York, 1991.

[7] J. Hertz, A. Krogh and R. Palmer, "Introduction to the Theory of Neural Computation", Addison-Wesley, ,1991.

[8] J. Tsai, "Neural Computation of the Eigenvectors of a symmetric positive definite Matrix"„ M.S. Thesis, Department of Computer Science, CSUSB, May 1996.

[9] G. M. Georgiou and J. Tsai, "Stochastic/neural computation of the eigenvectors of a symmetric positive definite matrix,"-In Proceedings of Joint Conference on Information Sciences, vol. 2, pp. 219-222, 1997.

[10] E. Oja, "A simplified neuron model as a principal components analyzer". Journal of Mathematical Biology, vol. 15, pp. 267-273, 1982.

[11] T. D. Sanger, "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network", Neural Networks, Vol. 2, pp. 459-473, 1989.

[12] T. D. Sanger, "An Optimality Principle for Unsupervised Learning", Advances in Neural Information Processing Systems I, Denver, 1989.

[13] N. Samardzija and R. L. Waterland, "A neural network for computing eigenvalues and eigenvectors". Biological,.Cybernetics, vol. 68, pp. 155-164, 1992.