Algorithme de rétropropagation du gradient

Parmi tous les algorithmes de modification des poids synaptiques d'un RNA qui ont été proposés dans la littérature au fil des années, celui de la rétropropagation du gradient de l'erreur est celui qui a été le plus couramment utilisé et qui bénéficie de la plus vaste littérature dédiée [154], [155]. Pour cette raison, son utilisation dans cette thèse a été privilégiée au profit d'autres algorithmes d'apprentissage. En outre, les règles mathématiques sur lesquelles cet algorithme est basé sont intuitives, ce qui permet une prise en main rapide de la méthode.

4.3.4.1 Calcul de l'erreur de propagation

L'algorithme de rétropropagation du gradient de l'erreur consiste à mesurer l'erreur d'estimation commise par le perceptron entre le vecteur des sorties souhaitées O et le vecteur des sorties observées Y, puis à rétropropager le gradient de l'erreur pour ajuster la valeur des poids synaptiques.

Les équations développées dans cette section concernent le cas général d'un perceptron multicouche possédant plusieurs couches cachées.

L'erreur ϵ_q observée par le k-ième neurone de la couche de sortie est (4.2).

$$\epsilon_k = o_k - y_k \tag{4.2}$$

L'erreur globale E vue par la couche de sortie est définie comme la somme des erreurs quadratiques observées (4.3) [6], où le terme n_{dim} correspond à la dimension du vecteur de sortie. Le terme E peut également être perçu comme la fonction coût du problème que l'on cherche à minimiser.

$$E = \frac{1}{2} \sum_{k=1}^{n_{dim-out}} (o_k - y_k)^2 = \frac{1}{2} \sum_{k=1}^{n_{dim}} (\epsilon_k)^2 \tag{4.3}$$

La réponse donnée par un neurone j de la couche courante suite à un signal reçu de la couche précédente i est exprimée par le système (4.4a & 4.4b). Le terme ω_{ij} est le poids synaptique entre le neurone de la couche précédente et celui de la couche courante, n_{dim-i} représente le nombre de neurones composant la couche i et ϕ est la fonction d'activation utilisé par le neurone j.

$$\begin{cases} \Psi_j = \sum_{i=0}^{n_{dim-i}} (\omega_{ij} y_i) & \text{(4.4a)} \\ y_j = \phi(\Psi_j) & \text{(4.4b)} \end{cases}$$

A chaque itération n, le gradient de l'erreur $\frac{\partial E}{\partial \omega_{ij}}$ est calculé pour être propagé de la couche de sortie vers la couche d'entrée afin de modifier les poids synaptiques w_{ij}. La mise à jour des poids synaptiques Δw_{ij} s'effectue alors selon l'équation (4.5), où μ est le taux d'apprentissage du perceptron. Dans cette équation, il est intéressant de noter que la mise à jour se fait dans la direction opposée du gradient afin de se rapprocher du minimum global de la fonction erreur.

$$\omega_{ij}(n) = \omega_{ij}(n) + \Delta\omega_{ij}(n) = \omega_{ij}(n) - \mu \cdot \frac{\partial E}{\partial \omega_{ij}(n)} \tag{4.5}$$

Le développement de l'expression du gradient de l'erreur permet de mettre en évidence les différents termes utiles pour effectuer la mise à jour des poids synaptiques (4.6).

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \Psi_j} \cdot \frac{\partial \Psi_j}{\partial \omega_{ij}} \tag{4.6}$$

Les termes $\frac{\partial y_j}{\partial \Psi_j}$ et $\frac{\partial \Psi_j}{\partial w_{ij}}$ conservent une même expression quelque soit la couche pour laquelle l'erreur est rétropropagée (4.7 & 4.8).

$$\frac{\partial y_j}{\partial \Psi_j} = \frac{\partial \phi(\Psi_j)}{\partial \Psi_j} = \phi'(\Psi_j) \tag{4.7}$$

$$\frac{\partial \Psi_j}{\partial \omega_{ij}} = \frac{\partial (\sum_{i=0}^{n_{dim_i}} (\omega_{ij} y_i))}{\partial \omega_{ij}} = y_i \tag{4.8}$$

Cependant, le terme $\frac{\partial E}{\partial y_j}$ voit son expression être modifiée pour les différentes couches du perceptron, en effet, la valeur de l'erreur E est dépendante de y_j, selon la couche j considérée.

6. Le coefficient 1/2 permet d'effectuer des simplifications dans la suite du développement

4.3.4.2 Cas de la couche de sortie

Dans le cas de la couche de sortie, la rétropropagation de l'erreur a pour objectif de modifier les poids synaptiques entre la dernière couche cachée et la couche de sortie.

$$\frac{\partial E}{\partial y_j} = \frac{\partial E}{\partial \epsilon_j} \cdot \frac{\partial \epsilon_j}{\partial y_j} = \epsilon_j \cdot -1 = -\epsilon_j \tag{4.9}$$

L'expression générale du gradient de l'erreur pour la couche de sortie est donnée par (4.10).

$$\frac{\partial E}{\partial \omega_{ij}} = -\epsilon_j \cdot \phi'(\Psi_j) \cdot y_i \tag{4.10}$$

D'après (4.6), le terme de mise à jour $\Delta\omega_{ij}$ est alors définie par (4.11). Cette équation est plus connue sous le nom de règle de Widrow-Hoff ou règle du delta, où δ_j est le *gradient local* (4.12).

$$\Delta\omega_{ij} = \mu \cdot \epsilon_j \cdot \phi'(\Psi_j) \cdot y_i = \mu \cdot \delta_j \cdot y_i \tag{4.11}$$

$$\delta_j = \epsilon_j \cdot \phi'(\Psi_j) \tag{4.12}$$

4.3.4.3 Cas d'une couche cachée

Dans le cas d'une couche cachée, le terme $\frac{\delta E}{\delta y_j}$ désigne la variation de l'erreur observée sur la couche de sortie par rapport à la variation de la sortie de la couche courante (4.13). Les indices j et i désignent toujours respectivement la couche courante et la couche précédente tandis que l'indice k désigne la couche suivante.

$$\begin{aligned}
\frac{\partial E}{\partial y_j} &= \frac{\partial(\frac{1}{2}\sum_{k=1}^{n_{dim-out}}(\epsilon_k)^2)}{\partial y_j} = \sum_{k=1}^{n_{dim-out}}(\epsilon_k \cdot \frac{\partial \epsilon_k}{\partial y_j}) = \sum_{k=1}^{n_{dim-out}}(\epsilon_k \cdot \frac{\partial \epsilon_k}{\partial \Psi_k} \cdot \frac{\partial \Psi_k}{\partial y_j}) \\
&= \sum_{k=1}^{n_{dim-out}}(\epsilon_k \cdot \frac{\partial(o_k - \phi(\Psi_k))}{\partial \Psi_k} \cdot \frac{\partial(\sum_{j=0}^{n_{dim-j}}(\omega_{jk} y_j))}{\partial y_j}) \\
&= \sum_{k=1}^{n_{dim-out}}(\epsilon_k \cdot (-\phi'(\Psi_k)) \cdot \omega_{jk}) \\
&= -\sum_{k=1}^{n_{dim-out}}(\delta_k \cdot \omega_{jk})
\end{aligned} \tag{4.13}$$

En insérant (4.13) dans (4.6), le terme $\frac{\partial E}{\partial \omega_{ij}}$ devient (4.14).

$$\frac{\partial E}{\partial \omega_{ij}} = -[\sum_{k=1}^{n_{dim-out}}(\delta_k \cdot \omega_{jk})] \cdot \phi'(\Psi_j) \cdot y_i \tag{4.14}$$

L'expression finale de la mise à jour $\Delta\omega_{ij}$ dans le cas d'une couche cachée est similaire à l'expression finale de l'équation (4.11), à la différence que le gradient local δ_j est défini comme (4.15).

$$\delta_j = [\sum_{k=1}^{n_{dim-out}} (\delta_k \cdot \omega_{jk})] \cdot \phi'(\Psi_j) \tag{4.15}$$

4.3.4.4 Taux d'apprentissage et coefficient d'inertie

En plus du taux d'apprentissage μ, qui a pour but de contrôler l'amplitude de la mise à jour des poids synaptiques, un autre paramètre est couramment ajouté pour aider l'algorithme à s'extirper des extrema locaux : le coefficient d'inertie (ou *momentum*) noté α. Ce terme a pour objectif d'empêcher la fonction erreur E de se stabiliser dans un minimum local en ajoutant au terme de mise à jour $\Delta\omega_{ij}(n)$ une fraction de la mise à jour précédente $\Delta\omega_{ij}(n-1)$.

Le terme $\Delta\omega_{ij}$ à l'itération n se définit alors par l'équation (4.16).

$$\Delta\omega_{ij}(n) = -\mu \cdot \frac{\delta E}{\delta\omega_{ij}(n)} + \alpha \cdot \Delta\omega_{ij}(n-1) \tag{4.16}$$

Le taux d'apprentissage et le facteur d'inertie doivent faire l'objet d'une étude attentive pour assurer une diminution de l'erreur de prédiction au fil des itérations. D'après LeCun [155] et [165], une méthode efficace pour dimensionner ces termes est de les rendre dépendants de l'erreur globale du réseau. Ainsi, lorsque l'erreur de prédiction devient faible, le réseau semble converger vers une réponse stable. Le taux d'apprentissage et le terme d'inertie doivent alors avoir une valeur relativement faible pour ne pas perturber la convergence.

4.3.4.5 Normalisation des données

La normalisation des données d'apprentissage n'est pas une nécessité, mais il a été remarqué qu'une convergence plus rapide était obtenue dans les applications où les données d'entrées et de sorties sont normalisées [155], [166]. Ici, la normalisation désigne l'action de réduire la taille de l'espace des données en la ramenant sur un intervalle $]0;1[$ ou $]-1;1[$. Certains auteurs font aussi le choix de resserrer encore l'espace de normalisation en se ramenant à l'intervalle $]0,1;0,9[$. Cependant chaque problème étant unique, il convient de choisir l'intervalle de normalisation le plus approprié afin de réduire le temps de convergence de l'apprentissage.

Plusieurs raisons expliquent l'intêret d'effectuer une normalisation des données :

— La normalisation des données d'entrées permet de donner une importance égale à chacune des composantes du vecteur d'entrée sans tenir compte des valeurs réelles de ces composantes, ni de leurs unités respectives.

— La normalisation permet également de diminuer la difficulté du calcul numérique (ici, la rétropropagation de l'erreur) en permettant un conditionnement faible du problème d'apprentissage. Cette notion de conditionnement fait appel à des théories mathématiques qui ne seront pas explicitées ici, mais le lecteur intéressé peut se référer aux travaux de [167] qui étudie précisément l'impact du conditionnement sur l'apprentissage des réseaux de neurones, mais également sur l'influence d'autres facteurs comme le taux d'apprentisage ou le momentum.

— La normalisation des données permet également d'éviter les effets de saturation : lorsque les signaux reçus par les neurones sont très supérieurs à 1, les fonctions d'activations sont saturées et transmettent en sortie une valeur proche de leurs limites de variations. Ainsi, pour des fonctions d'activation de type sigmoide, ou hyperbolique, dès que la combinaison linéaire des entrées est supérieure en valeur absolue à 3, la sortie renvoyée est proche de 1 en valeur absolue. Ainsi, pour toute valeur supérieure en entrée, la réponse ne variera que très peu.

— Dans le cas des problèmes de régression, [155] précise qu'il est généralement utile de centrer les données d'apprentissage autour de 0 et d'effectuer une normalisation permettant d'obtenir une covariance [7] de 1 pour toutes les données. La mise à l'échelle permet alors d'équilibrer la manière dont l'algorithme de rétropropagation va modifier les coefficients synaptiques connectés à la couche d'entrée.

Le choix de l'intervalle de normalisation dépend également du problème traité. Seul un paramétrage par essais et erreurs peut permettre de déterminer la meilleure plage de variation de la normalisation. Cependant, une méthode assez simple pour avoir une idée de l'intervalle de normalisation est de déterminer quel intervalle est le plus susceptible de faire converger l'apprentissage en fonction du nombre de données d'entrée, de la valeur des sorties souhaitée, du nombre de couches cachées et des fonctions d'activation sur chaque couche. En d'autres termes, il est nécessaire d'évaluer le processus de propagation de l'information dans le réseau avant de choisir la méthode de normalisation.

4.3.4.6 Définition de l'erreur d'apprentissage

L'évaluation des performances d'estimation du RNA sur la base d'évaluation requiert de mettre en place plusieurs types d'erreurs d'estimation afin de suivre l'évolution de l'apprentissage et de s'assurer que celui-ci converge correctement vers l'erreur minimale admise E_{min}. [168]

La racine de l'erreur quadratique moyenne notée $RMSE$ (4.17) est la plus courante des erreurs utilisées, où N_{cas} est le nombre de cas composant la base d'évaluation.

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{N_{cas}}(O_k - Y_k)^2} \tag{4.17}$$

L'erreur quadratique relative notée RSE (4.18) compare l'erreur quadratique par rapport à la variance de l'estimation, il est possible de trouver cette erreur sous le nom d'erreur quadratique moyenne normalisée puisque le dénominateur est égal au produit de la variance estimée de la série σ^2 par le nombre de cas d'apprentissage dans la base de données. Ce type d'erreur présente l'avantage que sa valeur n'augmente pas proportionnellement à la taille de la base de données. En outre, la normalisation par la variance permet de comparer des séries d'amplitudes différentes [169]. Le terme $\overline{Y_k}$ représente la moyenne des sorties du réseau neuronal pour le k-ième cas d'apprentissage.

$$RSE = \frac{\sum_{k=1}^{N_{cas}}(O_k - Y_k)^2}{\sum_{k=1}^{N_{cas}}(\overline{Y_k} - Y_k)^2} = \frac{\sum_{k=1}^{N_{cas}}(O_k - Y_k)^2}{N_{cas}\,\sigma^2} \tag{4.18}$$

7. L'utilisation d'une covariance permet de spécifier la dispersion des données : c'est une mesure qui caractérise l'écart entre les données par rapport à leurs espérances mathématiques respectives. La covariance étudie donc la variation des variables entre elles.

L'erreur absolue moyenne notée MAE (4.19) offre un indicateur sur la déviation moyenne de l'erreur d'estimation.

$$MAE = \frac{1}{n} \sum_{k=1}^{N_{cas}} |O_k - Y_k| \tag{4.19}$$

Le taux d'erreur moyen noté MPE (4.20) intégre un facteur de mise à l'échelle afin de pouvoir comparer différentes prédictions quelque soit le point de fonctionnement du système.

$$MPE = \frac{1}{n} \sum_{k=1}^{N_{cas}} \frac{Y_k - O_k}{Y_k} \tag{4.20}$$

Généralement, il est supposé que le bruit entachant les données est supposé suivre une distribution gaussienne. Dans le cadre de cette thèse, nous ne disposons pas des outils mathématiques pour effectuer la démonstration d'une telle hypothèse. De fait, un autre type d'erreur est introduit : l'erreur Laplacienne normalisée notée NLE (4.21). Comme son nom l'indique, la NLE présume que le bruit des données suit une distribution laplacienne et de la même manière que l'erreur RSE, la NLE subit une mise à l'échelle afin que sa valeur ne subisse pas l'influence de l'augmentation de la base de données.

$$NLE = \sum_{k=1}^{N_{cas}} \frac{|Y_k - O_k|}{|\overline{Y_k} - Y_k|} \tag{4.21}$$

Dans [168], l'auteur implémente les erreurs MSE, RSE, MAE et MPE afin de comparer les performances de cinq types de RNA ayant appris la même base de données.

Le suivi de l'évolution de ces erreurs permet alors entre autres choses de savoir s'il est nécessaire de modifier le paramétrage du RNA ou d'arrêter le processus en cas de stagnation des erreurs d'estimation.

4.3.4.7 Implémentation de l'algorithme

L'algorithme de rétropropagation, dans la version basique d'un apprentissage supervisé, est composé des étapes présentées dans l'algorithme 8.

Algorithme 8 Algorithme de rétropropagation du gradient de l'erreur

Entrée(s) Base d'apprentissage composée de N_{cas}-couples (I_n,O_n)
Initialiser la structure du PMC, les coefficients synatiques ω_{ij} de toutes les couches et définir les fonctions d'activation de chaque couche.
Tant que E est supérieur au critère d'arrêt **Faire**
Pour n = 1 : N_{cas} **Faire**
Propager le signal d'entrée I_n sur les couches du PMC (4.4a & 4.4b)
Calculer la sotie du réseau (4.4b pour la couche de sortie)
Calculer l'erreur d'estimation par l'équation (4.3)
Mettre à jour les poids synaptiques par l'équation (4.16)
Fin du Pour
Fin du Tant que
Sortie(s) Coefficients synatiques entrainés ω_{ij}

La mise en œuvre de l'algorithme 8 nécessite de définir certains paramètres :

Le critère minimal d'estimation E_{min}. L'algorithme de rétropropagation doit itérer la modification des poids synaptiques jusqu'à ce que le coût de la fonction E converge vers E_{min}. Ce critère est choisi de manière à arrêter l'apprentissage lorsqu'une précision suffisante est atteinte par le perceptron. En outre, il est également nécessaire de vérifier qu'au cours de l'apprentissage, les différentes erreurs d'estimation décroissent constamment. Une stagnation de ces erreurs sur un trop grand nombre d'itérations amène le RNA à faire du sur-apprentissage (*overfitting* en anglais) ce qui a pour conséquence principale de lui faire perdre ses capacités de généralisation. En outre, un sur-apprentissage peut aussi survenir lorsque la taille du réseau de neurones est trop conséquente par rapport à la taille de la base de données à apprendre : le RNA se comporte alors comme une table stockant les cas d'apprentissage.

La taille de la base d'apprentissage. Une base d'apprentissage de taille importante impose une phase d'apprentissage longue afin de développer les capacités de généralisation du RNA. Mais plus la base d'apprentissage est grande et plus l'estimation du système est précise.

La méthode d'apprentissage. Lors de l'apprentissage deux modes d'apprentissage peuvent être envisagés : l'apprentisssage offline ou l'apprentissage online. Le premier consiste à présenter successivement tous les cas de la base d'apprentissage au réseau de neurone afin de calculer une erreur moyenne à rétropropager alors que dans le second, un cas d'apprentissage est sélectionné aléatoirement à chaque itération, et l'erreur commise sur l'estimation de ce cas d'apprentissage est utilisée pour mettre à jour les coefficients synaptiques. L'apprentissage stochastique en ligne est très souvent privilégié pour ses propriétés de convergence plus élevées et sa meilleure capacité à suivre l'évolution de l'erreur de prédiction [170], [155].

Taux d'apprentissage μ et momentum α. Ces deux paramètres sont à choisir par essais et erreurs pour définir le couple (μ, α) qui permet d'effectuer un apprentissage efficace le plus rapidement possible.

4.3.4.8 Performances de l'estimation

La base d'apprentissage permet au RNA de développer des capacités d'estimation et de généralisation pour fournir la sortie la plus proche de la réponse attendue grâce à l'algorithme de rétropropagation du gradient, puis la base de validation est utilisée pour suivre l'évolution de l'erreur d'apprentissage et ainsi prendre les décisions adéquates sur la modification des paramètres de l'apprentissage ou la poursuite/arrêt du processus d'apprentissage.

Lorsque les erreurs définies en section 4.3.4.6 ont atteint les seuils souhaités, il est nécessaire d'effectuer une dernière validation des performances d'estimation du RNA en évaluant son aptitude à fournir une estimation sur la base de test. Cette dernière étape est indispensable pour certifier que le réseau est capable de fournir une estimation fiable pour des points de fonctionnement non appris.

Dans ce contexte, deux critères sont classiquement utilisés : le premier est le coefficient de corrélation linéaire (CCL) noté R, appelé aussi coefficient de Bravais-Pearson, qui permet de mesurer la qualité de l'estimation. Le CCL entre les sorties souhaitées et les sorties produites par le réseau neuronal est défini par le rapport entre la covariance

des données et le produit de leurs écart-types (4.22).

$$R = \frac{\sum_{k=1}^{n}(o_k - \overline{o})(t_k - \overline{t})}{\sqrt{\sum_{k=1}^{n}(o_k - \overline{o})^2 \sum_{k=1}^{n}(t_k - \overline{t})^2}} \tag{4.22}$$

Ce coefficient est donc une mesure de l'intensité de la liaison linéaire unissant les variables estimées et calculées. Le CCL est un nombre adimensionel défini sur l'intervalle $[-1; 1]$. Un CCL de valeur nulle indique que les séries O et Y sont linéairement indépendantes, tandis qu'un coefficient proche de l'unité indique une forte corrélation. Un CCL négatif révèle que Y varie en sens inverse de O et inversement en cas de CCL positif.

Le deuxième critère R^2 est le coefficient de détermination (CD), il est défini par (4.23), où le numérateur représente l'erreur quadratique de l'estimation tandis que le dénominateur représente la variance des données. La formulation du CD est ainsi fonction de l'erreur quadratique relative (RSE).

$$R^2 = 1 - \frac{\sum_{k=1}^{N_{cas}}(O_k - Y_k)^2}{\sum_{k=1}^{N_{cas}}(\overline{Y_k} - Y_k)^2} = 1 - RSE \tag{4.23}$$

Ce critère a pour objectif de donner une indication sur l'erreur d'estimation commise, mais également sur la dispersion des valeurs estimées [171]. Pour un problème de régression, une valeur de R^2 très proche de 1 est souhaitée à l'issue de l'apprentissage, autrement dit le terme d'erreur doit tendre vers 0 ($RSE \approx 0$).

Dans la pratique, ce critère est une sécurité supplémentaire qui n'a pas réellement d'intérêt sous l'hypothèse d'un suivi adéquat de la décroissance des erreurs d'estimation commises sur la base de validation. Cependant, il s'agit d'un critère universel de qualité de la régression qui permet de comparer les performances de convergence de divers algorithmes d'apprentissage.

4.3.5 Description des cas d'étude

Les performances de la méthodologie de conception d'un estimateur neuronal sont évaluées selon deux aspects : la précision de l'estimation et la rapidité de l'estimation.

Pour cela, l'étude des performances de l'estimateur est effectuée sur deux cas d'étude. Le premier cas d'étude consiste à estimer les puissances électriques vues par un carrousel de 16 trains sur un tour de boucle à un intervalle donné. Le second cas d'étude est un peu plus complexe et consiste à estimer les puissances électriques vues par un carrousel de 16 trains sur un tour de boucle pour l'ensemble des intervalles d'exploitation possibles : un carrousel établi de 16 trains est d'abord modélisé puis le modèle énergétique lié au déplacement des trains est résolu itérativement pour connaitre à chaque pas de temps les flux de puissance aux bornes des trains. Le processus est répété pour simuler le fonctionnement du carrousel pour les différents intervalles possibles. Les données issues de ces simulations sont ensuite présentées au réseau neuronal via un apprentissage supervisé. Un pas d'échantillonnage de 1s est utilisé, ce qui donne

une base de données constituée d'environ 3000 cas d'apprentissage pour chaque intervalle possible.

Le second cas d'étude est composé d'environ 12 fois plus de données que le premier cas d'étude, ce qui permet d'effectuer une comparaison de la méthodologie proposée pour différentes tailles de base de données.

Dans ces deux cas d'études, la base d'apprentissage est composée de 70% des données, tandis que les bases de validation et de test sont composées respectivement de 20 et 10% des données. Ainsi, les performances de l'estimation sont évaluées en ne connaissant que 70% des points de fonctionnement du système, ce qui permet de vérifier la capacité de généralisation du RNA.

Il est à noter que la rapidité de l'estimateur concerne le temps de calcul nécessaire à l'estimation d'un certain nombre de points de fonctionnement et non le temps d'apprentissage. En effet, plus le problème à approximer est complexe, plus il sera nécessaire d'allouer un temps d'apprentissage conséquent afin que l'estimateur puisse converger vers le taux d'erreur E_{min} souhaité.

En plus du critère d'arrêt E_{min}, un critère supplémentaire basé sur la valeur du coefficient de corrélation est défini afin que les bases de validation et de test jouent un rôle dans la décision de stopper l'apprentissage.

4.3.6 Performances de l'estimateur neuronal

4.3.6.1 Précision de l'estimation

4.3.6.1.1 Évolution des erreurs d'apprentissage sur la base de validation

Au fil des expérimentations, trois erreurs se sont avérées utiles pour suivre l'évolution de l'erreur d'apprentissage : RMSE, RSE et NLE (figures 4.9 et 4.10).

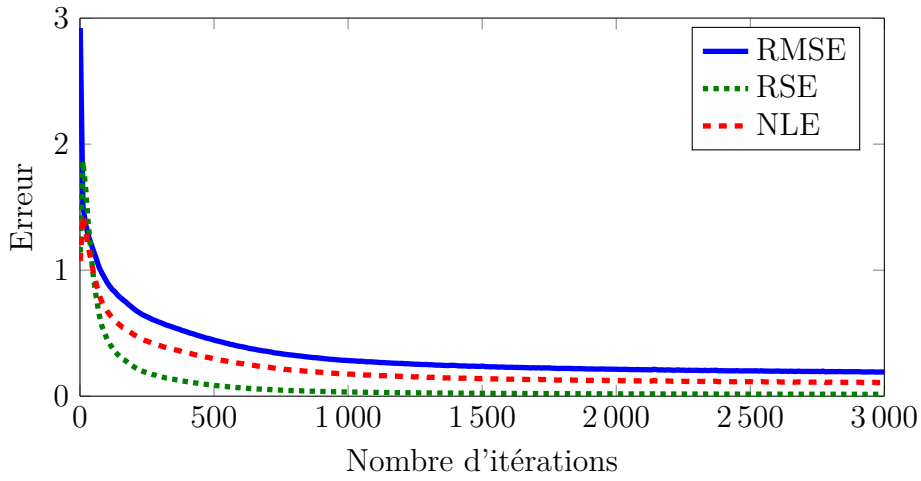

FIGURE 4.9 – Évolution des erreurs d'estimation sur la base de validation (cas 1).

L'erreur MAE a un comportement plus erratique au fil des itérations puisqu'elle représente une moyenne absolue des écarts d'estimation, cependant sa valeur finale à

l'issue de l'apprentissage est une bonne indication de la précision globale de l'estimation sur l'ensemble des points de fonctionnement décrits dans la base de validation.

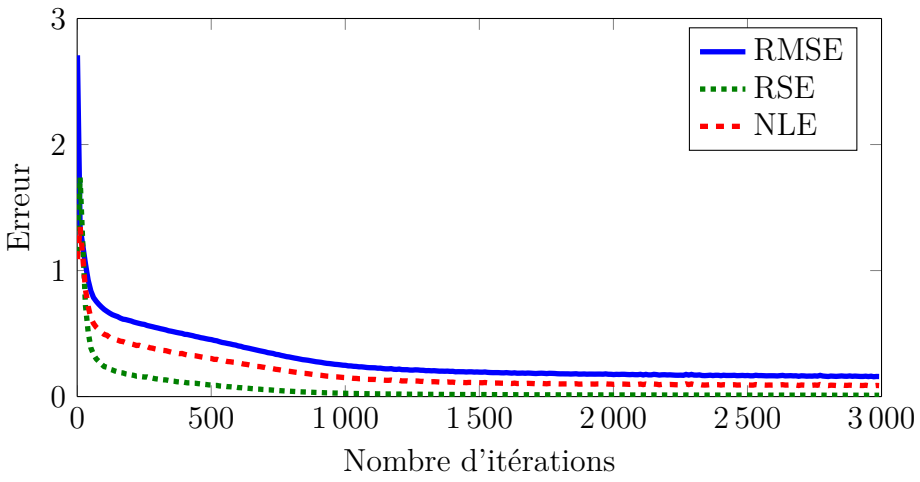

FIGURE 4.10 – Évolution des erreurs d'estimation sur la base de validation (cas 2).

La décroissance continue des erreurs sur les figures 4.9 et 4.10 est une première indication de la validité de l'architecture neuronale et du paramétrage de l'algorithme de rétropropagation.

De ce point de vue, seules les premières itérations présentent un intérêt puisque à partir d'un certain moment l'évolution des erreurs d'estimation n'est plus perceptible.

Ainsi, la première phase de décroissance rapide peut être assimilée à la phase de généralisation tandis que la seconde phase peut être vue comme une phase de spécialisation.

En revanche, l'erreur MPE est inadaptée pour suivre l'évolution de l'erreur d'approximation. En effet, pour plusieurs cas d'apprentissage, la sortie attendue est proche d'une valeur nulle ce qui engendre une MPE très élevée. De fait, cette erreur n'est pas considérée dans le reste de l'étude.

4.3.6.1.2 Évolution des coefficients de corrélation et de détermination sur la base de test

L'évolution du coefficient de corrélation et du coefficient de détermination pour les deux cas d'études sont exprimés par les figures 4.11 et 4.12.

La convergence des coefficients semble plus rapide dans le deuxième cas d'étude. L'une des raisons est que la base d'apprentissage du deuxième cas d'étude est composée de 12 fois plus de données que le premier cas d'étude.

A chaque itération le réseau neuronal ajuste ses poids synaptiques pour adapter son estimation à un plus grand nombre de points de fonctionnement, ce qui entraîne une vitesse initiale de convergence plus élevée.

En revanche, la convergence finale vers une valeur proche de 1 est beaucoup plus longue à obtenir pour les mêmes raisons.

L'allure de ces courbes renseigne également sur l'adéquation de l'architecture du réseau neuronal et des paramètres de l'algorithme de rétropropagation avec le problème à estimer.

— Une évolution plus irrégulière de ces courbes aurait ainsi mis en évidence que le taux d'apprentissage et le momentum ont des valeurs trop élevées ce qui ne permet pas à l'algorithme de rétropropagation de converger dans de bonnes conditions.

— Une convergence des coefficients vers une valeur très inférieure à 1 aurait signifié que l'architecture du RNA n'est pas suffisante pour développer des propriétés de généralisation. De même, une lente convergence peut également signifier que l'architecture du RNA est surdimensionnée.

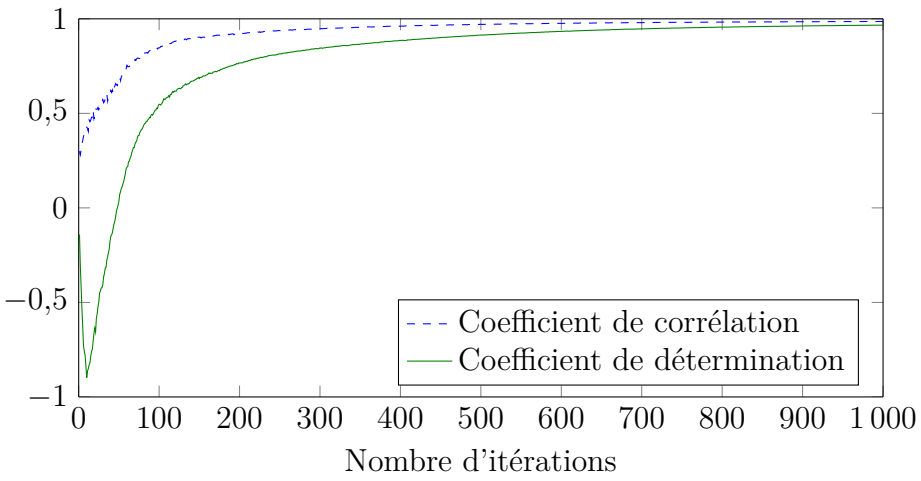

FIGURE 4.11 – Évolution de la corrélation entre données estimées et calculées (cas 1).

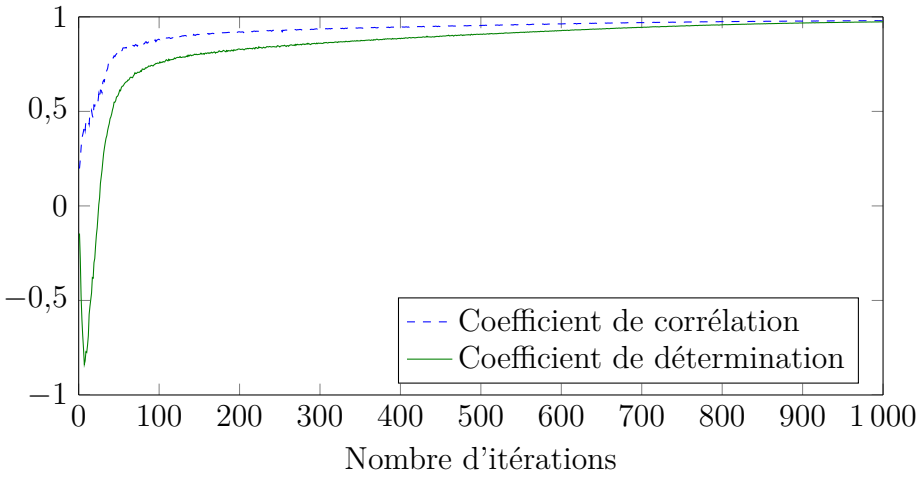

FIGURE 4.12 – Évolution de la corrélation entre données estimées et calculées (cas 2).

4.3.6.1.3 Représentativité de la base de test

A l'issue de l'apprentissage, le coefficient de corrélation et le coefficient de détermination issus de l'évaluation de la base de test ont une valeur très proche de 1, ce qui implique que le RNA a développé des capacités de généralisation permettant de donner une bonne estimation sur de nouveaux points de fonctionnement.

Les figures 4.13 et 4.14 représentent une analyse graphique de la corrélation sur l'ensemble de la base de données initiale.

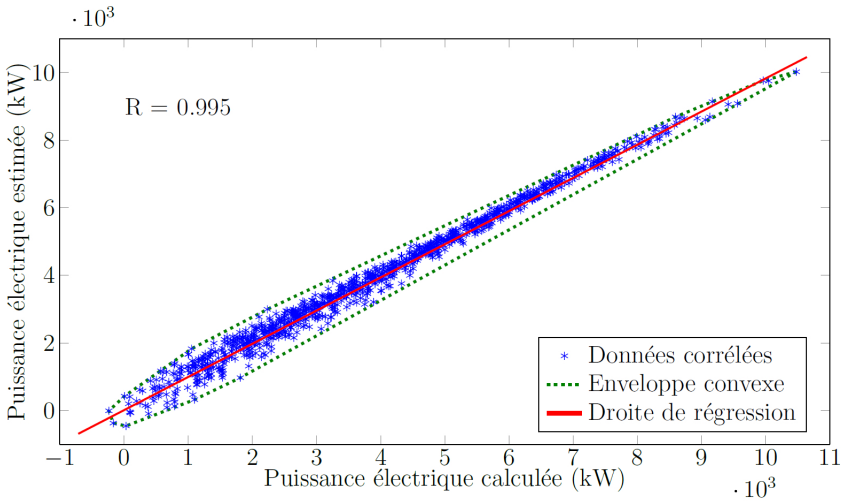

FIGURE 4.13 – Corrélation entre données estimées et calculées (cas 1).

L'abscisse correspond aux données calculées O et l'ordonnée aux données estimées Y. L'intérêt de ces graphiques est de visualiser les performances d'estimation sur l'ensemble des points de fonctionnement du système.

Dans les deux cas d'étude, l'apprentissage a été effectué jusqu'à ce que le coefficient de corrélation atteigne une valeur de 0,995.

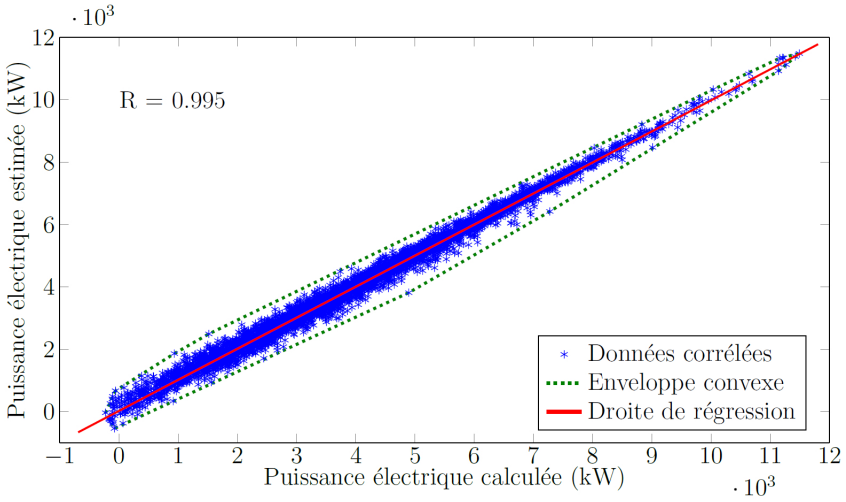

FIGURE 4.14 – Corrélation entre données estimées et calculées (cas 2).

Sur ces figures, l'enveloppe convexe représente la dispersion entre les données calculées et estimées. En théorie, un estimateur parfait verrait l'enveloppe convexe être confondue avec la droite de régression de pente unitaire.

Ainsi, la surface décrite par l'enveloppe convexe est définie comme un critère de dimensionnement supplémentaire pour choisir l'architecture du réseau neuronal la plus adaptée au problème.

Un sous-objectif de la méthodologie pourrait alors être de déterminer l'architecture neuronale et les paramètres de l'algorithme d'apprentissage permettant de minimiser l'aire de l'enveloppe convexe.

4.3.6.1.4 Visualisation de l'erreur d'apprentissage

Les figures 4.15 et 4.16 proposent une comparaison entre données calculées et estimées pour les 500 premières secondes de fonctionnement de la base de données pour les deux cas d'études. Elles constituent une illustration objective de la précision des estimateurs.

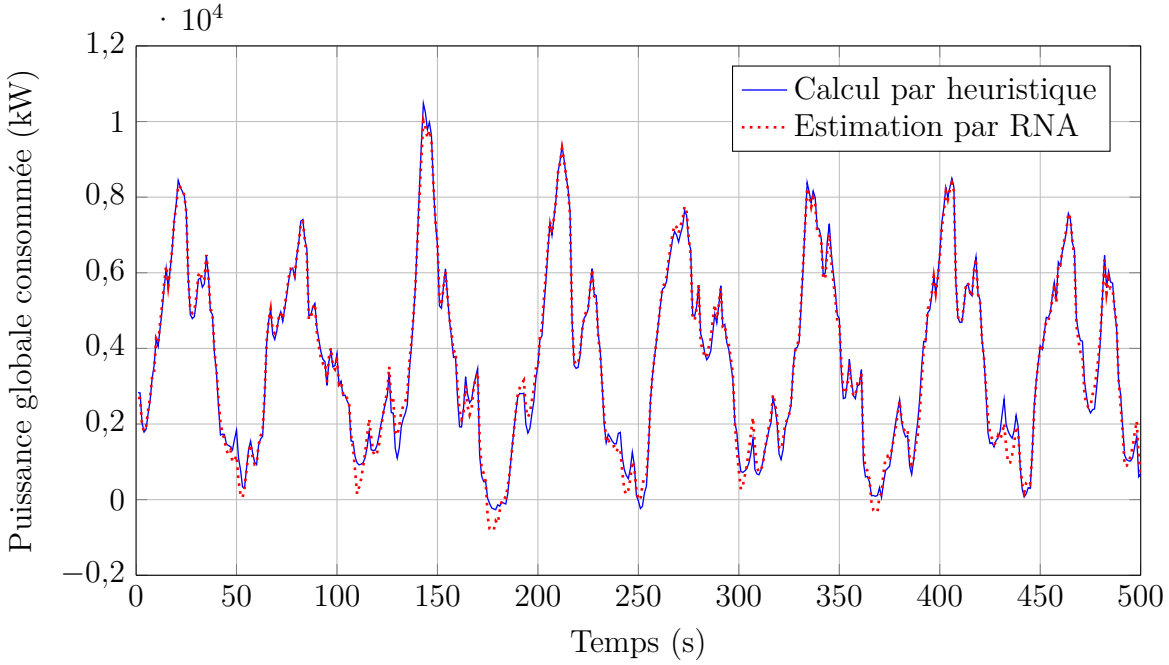

FIGURE 4.15 – Comparaison entre les données estimées et calculées (cas 1).

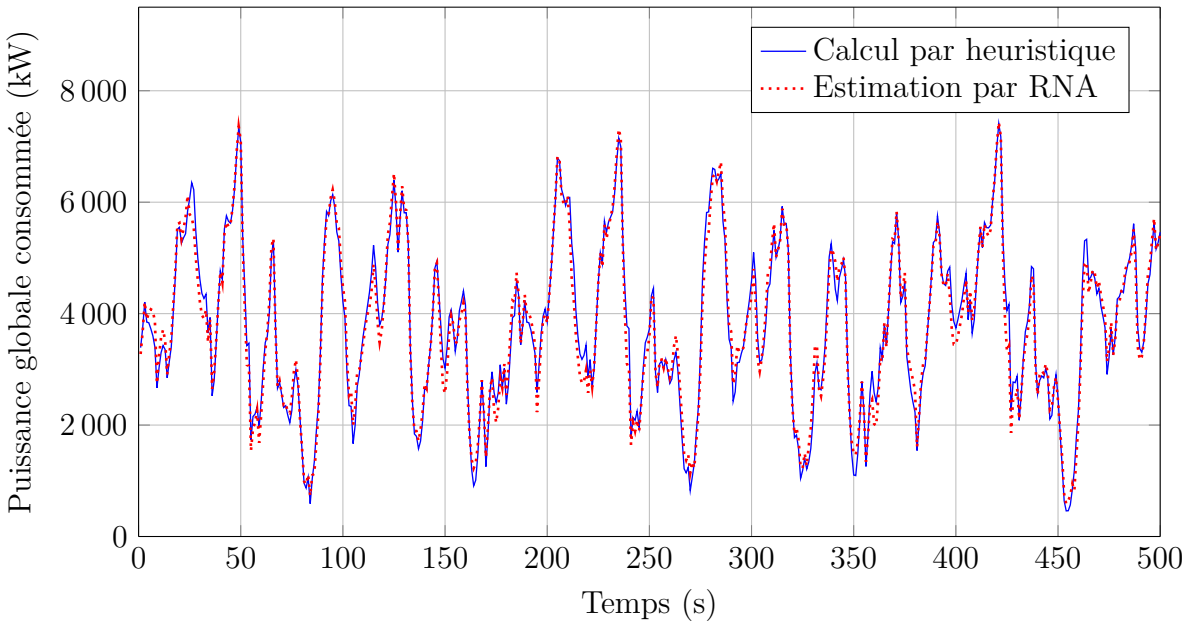

FIGURE 4.16 – Comparaison entre les données estimées et calculées (cas 2).

Ces figures sont obtenues en sommant les puissances électriques aux bornes des trains. Logiquement, la puissance globale résultante devrait être strictement positive quelles que soient les conditions d'exploitation des carrousels, cependant la résolution

itérative permet de calculer les flux de puissance vus par les trains avec une certaine erreur, ainsi la sommation des données calculées et estimées peut être légèrement négative du fait de cette marge d'erreur.

En théorie, le critère d'arrêt de la résolution itérative devrait donc être affiné pour réduire l'erreur commise, cependant le gain en précision pour la méthodologie globale serait nul puisque comme il est montré par la suite, une consommation énergétique du carrousel faible voire négative transmet la même information : la synchronisation des trains est optimale.

En effet, l'objectif final de la méthodologie n'est pas de déterminer précisément la consommation énergétique de la ligne, mais de définir dans quelles conditions d'exploitation cette consommation est minimisée. Ainsi, la plage de fonctionnement où l'estimateur neuronal produit la plus grande erreur est paradoxalement celle où cette erreur a le moins d'impact sur la prise de décision.

4.3.6.1.5 Remarques sur la précision de l'estimation

Précédemment, il a été fait mention de l'occurrence de deux phases lors de l'apprentissage : une phase de généralisation et une phase de spécialisation. Dans le cas d'une base de donnée de petite taille, la phase de spécialisation peut se transformer en phase de sur-apprentissage lorsque la phase d'apprentissage est prolongée excessivement pour améliorer l'estimation du réseau neuronal.

Ce problème de sur-apprentissage est surmonté dès lors que la taille de la base de données augmente. En effet, un sur-apprentissage sur une base de données exhaustive de tous les points de fonctionnement du système n'est rien de plus que l'objectif primaire visé lors de la conception de l'estimateur.

4.3.6.2 Rapidité de l'estimation

Le tableau 4.2 dresse un bilan des performances de l'estimateur en terme de précision et de rapidité. Le speedup lié à l'utilisation de l'estimateur est plus important lorsque le nombre de points de fonctionnement à estimer est faible.

Par la suite, nous privilégierons l'emploi de l'estimateur sur de faibles horizons temporels d'une part pour maximiser le speedup et d'autre part car il n'est pas pertinent d'effectuer des prédictions sur un horizon trop lointain du fait de l'occurrence de perturbations de trafic.

	Durée de résolution itérative	Durée d'estimation par approximateur neuronal	Speedup	MAE finale
Cas 1 ($N_{data} \approx 3.10^3$)	18,23 s	0,14 s	130,2	0,0178
Cas 2 ($N_{data} \approx 4.10^4$)	205,80 s	2,76 s	74,6	0,0057

Tableau 4.2 – Récapitulatif des performances de l'estimateur.

Dans le deuxième cas d'étude, la MAE est plus faible que dans le premier cas d'étude, ce qui suggère qu'en moyenne l'estimation est plus précise lorsque la taille de