Optimisation temps réel des tables horaires

« *Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution. It is, strictly speaking, a real factor in scientific research.* »

Albert Einstein

Sommaire

4.1 Introduction

4.1.1 Limites de l'approche hors-ligne

Les tables horaires sont conçues pour des conditions d'exploitation optimales où aucune perturbation de trafic ne se produit.

Cependant, dans un cas réel d'exploitation les aléas sont inévitables du fait de la présence de facteurs humains qui influent sur le fonctionnement de la ligne de métro automatique.

L'optimisation hors-ligne des paramètres d'exploitation permet de définir un ou plusieurs points de fonctionnement de la ligne jugés comme optimaux d'un point de vue énergétique, mais s'avère inefficace dès lors que le système s'écarte de ces points de fonctionnement.

En pratique, des marges de régulation sont prévues pour assurer la stabilité de l'horaire de passage des trains vis à vis des perturbations mineures qui peuvent être rencontrées. Néanmoins, la régulation n'a pas pour objectif d'assurer un optimum de consommation énergétique et il s'avère alors nécessaire d'insérer de nouvelles règles de fonctionnement pour assurer la réalisation de cet objectif.

4.1.2 Enjeux de l'approche temps réel

L'enjeu de ce chapitre est de définir une méthode pour rendre les travaux présentés précédemment applicables en temps réel, en considérant des conditions réelles d'exploitation intégrant des perturbations de trafic.

Les défis scientifiques et techniques à relever peuvent être synthétisés par la problématique générale suivante : *Comment réaliser une aide à la décision capable de s'adapter aux perturbations d'un système dynamique et de fournir une réponse optimale en temps réel ?* ou en d'autre termes plus spécifiques au sujet de thèse : *Comment effectuer une replanification en temps réel des temps de stationnement des trains pour minimiser la consommation énergétique du carrousel ?*[1]

Nous décrirons donc dans ce chapitre une méthodologie capable de déduire une politique décisionnelle optimale du fonctionnement nominal d'un système, puis de modifier celle-ci pour qu'elle s'adapte aux perturbations rencontrées par ce système.

Concrètement, cela implique de prendre en compte les modifications des conditions de trafic dans la boucle d'optimisation, mais également d'atteindre un temps de calcul pour la boucle d'optimisation qui soit suffisamment faible pour que celle-ci puisse être mise en œuvre en temps réel sur la ligne en exploitation.

Dans la première partie du chapitre, une introduction à l'intelligence artificielle (IA) et au principe de fonctionnement d'un réseau de neurones artificiels (RNA) est réalisée.

Cette introduction présente d'une part nos attentes vis à vis de l'implémentation d'une IA, d'autre part des exemples concrets d'applications qui justifient l'adéquation des RNA pour concrétiser les enjeux visés. Ensuite les différentes méthodes d'apprentissage sont passées en revue pour déterminer celle qui est la plus adaptée pour résoudre

1. Dans ces travaux, toutes les actions de replanification sont assimilées à des modifications de temps d'arrêt en station par rapport aux temps de stationnement nominaux.

la problématique.

La deuxième partie du chapitre est dédiée à la méthodologie de conception d'un estimateur neuronal. L'objectif est de synthétiser la méthode de résolution itérative des flux de puissance dans un RNA. Le RNA serait alors capable d'estimer les flux de puissance qui se produisent sur le réseau électrique entre les trains et les sous-stations en fonction du déplacement des trains et de fournir une approximation de ces flux en un temps très court.

La troisième partie s'intéresse quant à elle à la résolution de la problématique d'optimisation dynamique.
En effet, au chapitre précédent, il a été montré que l'exploration d'une solution du problème d'optimisation des temps de stationnement nécessite environ une dizaine de secondes de temps de calcul, ce qui est incompatible avec un objectif d'optimisation en temps réel. Pour ce faire, le principe d'*apprentissage par renforcement* (AR) est introduit.
Cette méthode permet de déduire une politique décisionnelle d'une suite d'essais et d'erreurs issus d'interactions successives d'un agent apprenant avec son environnement.

Enfin la dernière partie du chapitre est consacrée à l'implémentation de la méthode d'apprentissage par renforcement, et de sa mise en œuvre pour fournir une aide à la décision optimale sur la valeur du temps de stationnement que doit effectuer chaque train pour respecter les contraintes d'exploitation, tout en minimisant la consommation énergétique de la ligne. Une étude des performances de cette méthode est alors effectuée pour en évaluer la capacité à effectuer une optimisation temps réel d'une ligne de métro.

4.1.3 Cahier des charges

Objectifs	• Optimisation temps réel de la consommation énergétique d'une ligne de métro
Contraintes	• Marge de variation des temps d'arrêt en station • Temps de battement • Prise de décision optimale en temps réel • Aléas d'exploitation
Moyens d'action	• Apprentissage des solutions issues d'optimisations
Indicateurs	• Taux de réutilisation du freinage électrique • Déviation par rapport à la table horaire initiale • Temps de calcul

Tableau 4.1 – Cahier des charges de l'optimisation énergétique temps réel d'une ligne de métro automatique

Le cahier des charges de l'optimisation énergétique temps réel est résumé dans le tableau 4.1 et présente les objectifs, les contraintes de l'étude, les moyens d'actions pour réaliser l'optimisation ainsi que les indicateurs utilisés pour évaluer le niveau d'atteinte de l'objectif.

Dans ce chapitre, l'objectif se limite à effectuer une optimisation temps réel des temps de stationnement, car il est considéré que l'intervalle d'exploitation est une contrainte imposée par l'exploitant et que les profils de vitesse ne sont pas des variables d'ajustement puisqu'imposés par la régulation du trafic.

4.1.4 Etat de l'art sur l'optimisation temps réel ferroviaire

Dans le domaine ferroviaire, la notion de replanification en temps réel adopte de nombreuses interprétations. Il convient ici de différencier les Systèmes Légers sur Rails (SLR, dérivé du terme anglais *light rail*), des autres systèmes ferroviaires (comme le fret, les réseaux intercités, ...). La gestion du trafic dans les SLR présente généralement beaucoup moins de contraintes d'exploitation du fait de sa faible longueur et de sa relative simplicité par rapport à un grand réseau ferroviaire présentant des interconnections. Ainsi dans un SLR, des contraintes/actions comme la priorisation des trains aux nœuds d'un réseau, les rotations courtes, le saut de station ou le surstationnement pour réduire le coût opérationnel de transport des passagers ne sont pas mises en œuvre.

Dans cette section, tous les travaux portant sur la replanification temps réel dans le domaine ferroviaire sont traités sans distinction. Cependant il est à noter que certaines actions de replanification mentionnées n'ont pas de raison d'être ou ne peuvent pas être appliquées dans des réseaux ferrés de type métro automatique.

Néanmoins, il reste intéressant d'étudier les procédés explorés dans ces travaux pour traiter le problème de replanification.

Gestion de conflits. Dans [109–112], la replanification temps réel consiste à gérer les conflits d'itinéraires dûs aux aléas d'exploitation pour suivre une table horaire de référence. Des algorithmes d'optimisation sont implémentés pour re-concevoir des tables horaires optimales et robustes après détection d'une perturbation de trafic, afin de continuer à assurer le maximum de connections aux nœuds du réseau ferroviaire tout en réduisant le temps d'attente des usagers.

Respect de la qualité de service et de l'intervalle. [73] tente d'utiliser un système expert à base de logique floue pour effectuer une replanification en temps réel visant à respecter un certain taux de service après aléa [2].

[113] et [114] réalisent une régulation de trafic afin de garantir un intervalle d'exploitation constant entre les trains. Les mesures de régulation ont essentiellement pour objectif d'optimiser les services offerts aux usagers, notamment le temps moyen d'attente.

[115] propose un modèle de contrôle temps réel de l'intervalle d'exploitation par couples de trains consécutifs, visant à modifier les horaires de départs de station pour minimiser la variance de l'espacement temporel entre les trains à chaque station. Néanmoins, avec cette méthode, l'erreur de prédiction peut être propagée et amplifiée au fil de l'optimisation lorsque le nombre de stations augmente. Une approche similaire est également explorée par [116] pour effectuer un contrôle de l'intervalle moyen entre les trains et une minimisation du temps moyen d'attente des passagers, tout en intégrant la notion de réduction de la consommation énergétique de la ligne dans la fonction objectif du problème d'optimisation.

2. Cependant cette technique présente le grand désavantage de nécessiter l'expertise d'un humain pour réaliser des règles floues et des fonctions d'appartenance qui décrivent les contraintes opérationnelles inhérentes à la ligne étudiée.

Dans [117] et [118], Lin propose également de définir des régulations automatiques de trafic pour augmenter la robustesse des tables horaires face aux aléas et assurer la stabilité de l'intervalle d'exploitation par programmation dynamique. Le principe retenu concerne la modification du temps de parcours interstation et la modification des temps de stationnement. Cependant comme l'indique l'auteur dans [119], la résolution par programmation dynamique nécessite de réaliser des recherches vers l'avant qui peuvent engendrer une explosion du temps de calcul. Ainsi, dans [119] et [120], il suggère d'utiliser un processus d'apprentissage par renforcement avec une architecture acteur-critique afin de réaliser les mêmes objectifs que dans ses travaux précédents.

Maximisation de la récupération du freinage électrique. Dans [121], Qu présente un algorithme pour recalculer en temps réel les profils de vitesse optimaux que doivent suivre les trains en interstation pour minimiser la dissipation du freinage électrique, en mettant particulièrement l'accent sur l'analyse de la topographie de la voie pour définir des consignes d'éco-conduites.

[122] et [123] prennent aussi le parti de réduire la consommation énergétique de transports urbains en optimisant les profils de vitesses des trains par alternance de phases de traction, de freinage, de maintien de vitesse et de marche sur l'erre.

Dans [124] et [125], Yin se propose de résoudre le problème de minimisation de la consommation par un processus d'apprentissage par renforcement, en modifiant dynamiquement les profils de vitesse et les temps de stationnement.

Les travaux présentés dans [126] [127] utilisent une approche sensiblement différente : la notion de replanification en temps réel implique qu'aucune table horaire ni aucun intervalle d'exploitation ne sont définis au préalable. L'optimisation a alors pour but de minimiser les coûts opérationnels, le temps de trajet total des passagers ainsi que la consommation énergétique de la ligne. Le principe retenu est la modification des horaires de départs, des temps d'arrêt en station des trains et la modification des profils de vitesse interstation. Dans ces articles, l'auteur fait le choix de proposer des algorithmes qui exploitent de nombreux degrés de liberté pour réaliser un double objectif : satisfaire le client ainsi que l'exploitant.

Parmi les travaux évoqués ci-dessus, ceux traitant d'une replanification temps réel avec un objectif de minimisation de la consommation énergétique souffrent de l'inconvénient majeur de ne pas intégrer de modèle de consommation des trains qui soit fiable ou même représentatif du comportement hautement non linéaire de la caractéristique de renvoi de puissance lors des phases de freinage [3].

3. Lors de cette thèse, en première approche, une modélisation simpliste de la consommation énergétique des trains, avait été employé mais il s'est révélé que l'erreur de simulation commise était comprise entre 20 et 30%. En comparaison, la modélisation énergétique présentée au chapitre 2 entraîne une erreur moyenne de simulation d'environ 6%.

4.1.5 Concept d'intelligence artificielle

Parmi l'ensemble des méthodes d'optimisation existantes, nous avons fait le choix dès l'élaboration du sujet de thèse, de faire appel à des méthodes d'intelligence artificielle (IA).

Ce terme fait référence à des programmes informatiques qui tentent d'imiter le raisonnement humain pour réaliser des tâches complexes. Les comportements adoptés par ces programmes apparaissent alors comme intelligents aux yeux d'un observateur humain.

Au sens strict, le terme *intelligence* implique qu'une machine ou un programme est capable de résoudre des problèmes par une construction originale, sans qu'aucune résolution n'ait été déterminée a priori [128].

Les premières évocations du terme intelligence artificielle remontent aux travaux d'Alan Turing en 1950, dans lequel l'auteur définit un test permettant de déterminer le degré de *conscience* d'une machine : un examinateur juge du type d'interlocuteur à qui il a affaire (humain ou machine) en analysant les réponses fournies à une série de questions.

En 1943, Mc Culloch et Pitts proposent un modèle mathématique pour modéliser le fonctionnement du cerveau : le *neurone formel*. Ils posent ainsi les bases du *réseau neuronal*.

En 1957, Rosenblatt implémente le premier *perceptron*, qui consiste en un réseau de neurones formels composé d'une couche d'entrée et d'une couche de sortie. Ce perceptron a été utilisé pour effectuer de la reconnaissance de forme.

Par la suite, de nombreux projets portés sur le développement d'IA ont vu le jour [4].

Le paradigme qui en résulte est que le problème de conception d'une IA s'est détourné de l'enjeu de l'intelligence vers celui de la *connaissance*.

En effet, dans la pratique, l'implémentation d'une IA se fait par un processus d'apprentissage sur les données disponibles du problème à modéliser. De fait, le degré d'intelligence du système est alors déterminé par la quantité et la qualité des connaissances accumulées lors de l'apprentissage.

En outre, à notre connaissance, seuls [119], [120], [124] et [125] choisissent d'employer des outils issus de l'intelligence artificielle pour réaliser une replanification en temps réel des trains dans un réseau ferroviaire.

A ce titre, [124] et [125] se distinguent singulièrement en offrant une vision intéressante de la replanification adaptée au cas des lignes de métro automatique. La formulation proposée dans [125] est ainsi assez proche de celle proposée dans le chapitre 3 pour réaliser une optimisation hors-ligne des paramètres d'exploitation.

De fait, cette méthode est explorée dans ce chapitre pour rendre le processus d'optimisation hors-ligne réalisable en temps réel.

4. La section 4.2.1 dresse une liste non exhaustive des différents projets qui ont été passés en revue dans le cadre de cette thèse afin de cerner les enjeux concrets de la conception d'une IA efficace.

4.1.6 Nécessité de synthétiser le processus de résolution itératif des flux de puissance

Avant de mettre en œuvre la méthode d'apprentissage pour déduire une politique décisionnelle optimale pour le problème de modification dynamique des temps d'arrêt, il apparait nécessaire de synthétiser la méthode de résolution itérative des flux de puissance sur le réseau.

En effet, dans la section 4.1.4, l'étude des différents travaux portant sur la replanification temps réel dans le domaine ferroviaire a mis en lumière que l'utilisation d'un modèle de consommation énergétique simpliste était privilégié. Cette constatation est conforme à ce qui a été établi au chapitre précédent où il a été montré qu'environ 90% du temps d'exploration d'une solution était consacré à la résolution itérative des flux de puissance entre trains et sous-stations.

Dans le cadre de cette thèse, l'utilisation d'un modèle énergétique simpliste n'aurait pas de sens puisque cela reviendrait à sacrifier la précision du processus d'optimisation au profit d'un gain en temps de calcul. En revanche, la synthèse du processus de résolution itératif par un réseau neuronal permettrait d'annuler en grande partie le temps de calcul nécessaire à l'évaluation de la consommation énergétique liée à une solution.

4.2 Réseaux de neurones artificiels

Le principe des réseaux de neurones artificiels (RNA) est issu de l'analogie entre la biologie et les mathématiques opérée par Mc Culloch et Pitts. Les RNA visent à mimétiser le fonctionnement des neurones à l'intérieur du cerveau humain en propageant les signaux synaptiques et en stockant les informations pertinentes à l'apprentissage d'une tâche donnée.

Un RNA est caractérisé par deux attributs : d'une part, l'organisation des neurones au sein du réseau que l'on appelle architecture et d'autre part l'algorithme d'apprentissage.

L'architecture d'un réseau définit entre autres choses le nombre de neurones composant le RNA et les connexions entre les neurones des différentes couches, à savoir la manière dont le signal est propagé au sein du RNA.

L'algorithme d'apprentissage a pour rôle de stocker le savoir accumulé au sein des coefficients synaptiques du réseau de manière à obtenir le comportement souhaité.

4.2.1 Applications

Les applications du principe de RNA sont multiples dans la littérature :

- La robotique pour le contrôle et le guidage de robots ou de véhicules autonomes [129], [130]
- Les statistiques pour la prévision, la classification et l'analyse de données [131]
- Le traitement du signal pour la reconnaissance de formes et de sons [132], [133]
- La finance pour le calcul de la volatilité d'un marché [134] et la prévision économique de séries temporelles [135]
- Le diagnostique médical [136],[137]

— La création d'intelligence artificielle dans les jeux vidéos [138], [139]
— La gestion de systèmes hydrauliques [140] et des applications en aérospatiale [141]
— Le contrôle de machines électriques [142], la sureté de systèmes électriques [143] et la synthèse de résolution *load flow* [144], [145], [146]
— L'approximation de paramètres de systèmes fortement non-linéaires [147]

Cette liste est loin d'être exhaustive tant les travaux mettant en œuvre des réseaux neuronaux pour des applications spécifiques sont abondants dans la littérature.

La lecture de ces différents travaux fournit un vaste aperçu de ce qu'il est possible de réaliser via la mise en œuvre d'un réseau neuronal, mais permet surtout d'avoir connaissance des obstacles qui ont dû être surmontés pour rendre ce concept mathématique applicable à des cas concrets. A ce titre, ces travaux se révèlent bien plus utiles que ceux portant sur le domaine ferroviaire puisqu'ils permettent d'envisager une approche différente que ce soit sur la formulation, la modélisation ou même la résolution du problème étudié.

4.2.2 Principe des Réseaux de Neurones Artificiels

4.2.2.1 Modèle biologique

En biologie, un neurone est une cellule spécialisée dans le traitement et la transmission d'information dans le cerveau qui constitue l'unité élémentaire du système nerveux. Il se compose généralement d'un corps cellulaire (*péricaryon* ou *soma*) et de prolongements : un axone et des dendrites. Les dendrites sont les ramifications du neurone qui lui permettent de recevoir les signaux électriques et chimiques issus des autres neurones. L'axone est un prolongement de la cellule qui conduit le signal électrique sortant jusqu'aux dendrites auxquelles il est interconnecté. L'échange d'information entre un axone et une dendrite s'effectue par la synapse. L'axone génère un *potentiel d'action* et la synapse assure la conversion et la transmission du signal à la dendrite.

La transmission de l'information neuronale est illustrée par la figure 4.1 [148].

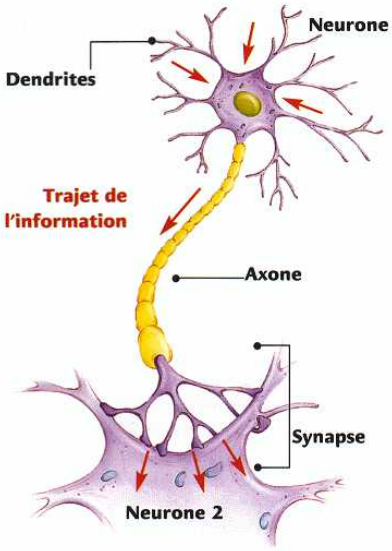

FIGURE 4.1 – Structure typique d'un neurone.

La propagation des signaux électriques ne se fait pas de manière linéaire, mais par un effet de seuil : l'information n'est transmise que lorsqu'un potentiel d'action adéquat est reçu par le neurone.

4.2.2.2 Le neurone formel

Un neurone formel est une fonction algébrique paramétrée de variables réelles dont les valeurs de sorties sont bornées. Un neurone formel est composé de quatre éléments fondamentaux : les entrées, les poids synaptiques associés, une fonction d'agrégation [5] et une fonction d'activation.

Les données d'entrées correspondant aux variables du problème sont pondérées par les poids synaptiques puis sommées et enfin évaluées par une fonction d'activation pour obtenir une sortie. La fonction d'activation permet de recréer l'effet de seuil qui se produit lors de la propagation de l'information dans les synapses. La sortie est la réponse du neurone formel au stimulus reçu en entrée.

Sur la figure 4.2 est représenté le schéma d'un neurone formel possédant n entrées, un biais unitaire et produit une sortie unique notée y.

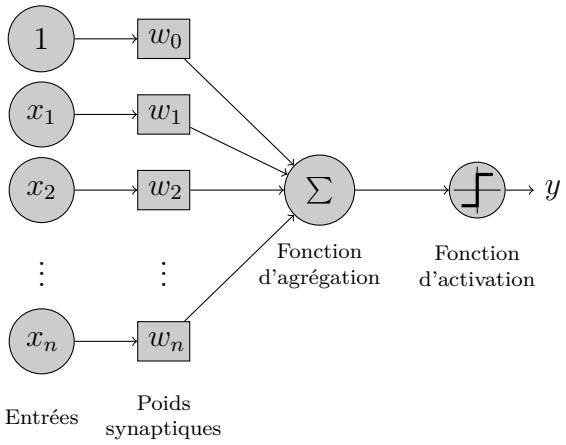

FIGURE 4.2 – Représentation d'un neurone formel à n entrées et 1 sortie.

De fait, le fonctionnement d'un neurone formel est régi par le système d'équations (4.1), où x_i et ω_i représentent respectivement les entrées et les poids synaptiques du neurone. Le biais b intègre une entrée supplémentaire, dont la valeur est fixée à 1, et permet de créer un *offset* pour discriminer les données d'entrées (son rôle est similaire à celui de l'ordonnée à l'origine pour une fonction affine ; ici, $b = 1 \cdot \omega_0$). Le terme Ψ est appelé potentiel d'activation du neurone tandis que ϕ est la fonction d'activation du neurone qui permet de produire la valeur de sortie y.

$$\begin{cases} \Psi = b + \sum_{i=1}^{n} \omega_i x_i = \sum_{i=0}^{n} \omega_i x_i \\ y = \phi(\Psi) \end{cases} \tag{4.1}$$

Les poids synaptiques pondèrent les signaux transmis et régissent le fonctionnement du RNA en fournissant une application de l'espace des entrées vers l'espace des sorties.

5. Dans cette exemple, la fonction d'agrégation est une combinaison linéaire

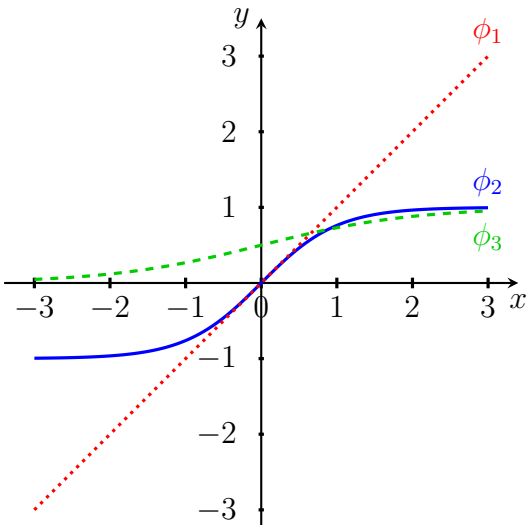

FIGURE 4.3 – Représentation des fonctions d'activation.

Trois types de fonctions d'activation sont classiquement employées.

La fonction identité : $\phi_1(\Psi) = \Psi$

La fonction tangente hyperbolique : $\phi_2(\Psi) = tanh(\Psi)$

La fonction sigmoïde : $\phi_3(\Psi) = \dfrac{1}{1 + e^{-\Psi}}$

La figure 4.3 présente le tracé des fonctions ϕ_1,ϕ_2 et ϕ_3 définies sur l'intervalle [-3,3].

4.2.2.3 Le perceptron multicouche

Un perceptron multicouche (PMC) est une structure composée de plusieurs couches de neurones. Dans un PMC, l'information est propagée de la couche d'entrée vers la couche de sortie. Cette architecture est la plus courante dans la littérature sur les RNA, ainsi, la quasi totalité des travaux cités en section 4.2.1 emploient cette architecture neuronale. L'idée sous-jacente est de combiner plusieurs fonctions élémentaires pour former des fonctions plus complexes.

Un PMC contient une couche d'entrée, une couche de sortie et une (ou plusieurs) couche(s) cachée(s). La couche d'entrée est une couche virtuelle étant donné qu'elle n'a pour rôle que de recevoir les signaux entrants et de les propager aux couches suivantes.

Un perceptron multicouche est alors paramétré par le nombre de neurones composant chacune des couches du réseau, la topologie des connexions entre les neurones (ici, le choix a été fait que tous les neurones d'une couche soient reliés à tous les autres neurones de la couche adjacente), la fonction d'agrégation, l'algorithme d'apprentissage et les fonctions d'activation utilisées par les différentes couches du réseau. Chaque couche du réseau peut ainsi utiliser une fonction d'activation différente afin d'atteindre les objectifs visés.

En outre, il est à noter que hormis les perceptrons multicouches, les réseaux de fonctions à base radiale (RFBR) sont également un type d'approximateur universel populaire. Les RBFR sont en beaucoup de points similaires aux PMC et différent principalement par la manière dont les signaux émanant des couches précédentes sont

combinées. Ainsi, les RFBR utilisent une distance euclidienne en guise de fonction d'agrégation.

Dans cette étude, nous avons fait le choix d'étudier l'approximation par PMC, d'une part pour sa facilité d'implémentation, d'autre part pour sa très bonne capacité à fournir un approximateur robuste dans un grand nombre d'applications solutionnant des problèmes de grandes dimensions, et enfin les algorithmes d'apprentissage sont efficaces pour converger en un nombre raisonnable d'itérations.

La figure 4.4 représente un PMC constitué de quatre entrées, une couche cachée composée de cinq neurones et une couche de sortie à deux dimensions. Dans ce formalisme, chaque cercle correspond à un neurone et les flèches désignent les poids synaptiques. De plus, la couche cachée et la couche d'entrée contiennent également un biais pour permettre au réseau de développer une meilleure capacité de généralisation.

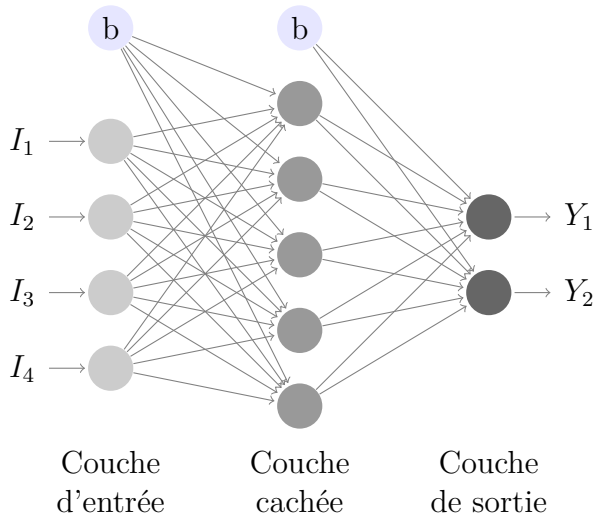

FIGURE 4.4 – Exemple structurel d'un perceptron multicouche.

Pour plus de clarté, les vecteurs contenant les p signaux entrants et les q signaux de sorties sont respectivement notés $I = (I_1, ..., I_p)$ et $Y = (Y_1, ..., Y_q)$.

4.2.3 Notion d'apprentissage

Pour un réseau de neurones artificiels, l'apprentissage d'une tâche ou d'un processus est réalisé par la mise à jour des poids synaptiques. Une phase d'apprentissage consiste donc à modifier ces poids synaptiques jusqu'à ce que le RNA effectue les actions souhaitées.

Les réponses attendues du RNA dépendent alors du problème considéré : dans le cas d'un problème de classification, il peut s'agir de déterminer le centre des classes ou une surface de séparation pour discriminer les cas d'apprentissage, en revanche, dans le cas d'un problème de régression ou d'approximation de fonction le RNA a pour objectif d'approcher une fonction continue sur l'intégralité de son domaine de définition.

Il existe principalement trois types d'apprentissage : l'apprentissage supervisé, l'apprentissage non-supervisé et l'apprentissage par renforcement. Ces apprentissages peuvent être réalisés selon deux modes : en *offline* ou en *online*.

4.2.3.1 Apprentissage supervisé

Lorsque le comportement de la fonction à modéliser est connu, il est possible de constituer une base d'apprentissage composée de couples *entrées-sorties*. Ces couples sont alors utilisés pour effectuer un apprentissage supervisé du réseau neuronal pour l'entraîner à prédire les sorties correspondantes aux entrées. Le réseau s'adapte en modifiant ses poids synaptiques pour converger vers la sortie désirée [149] (figure 4.5).

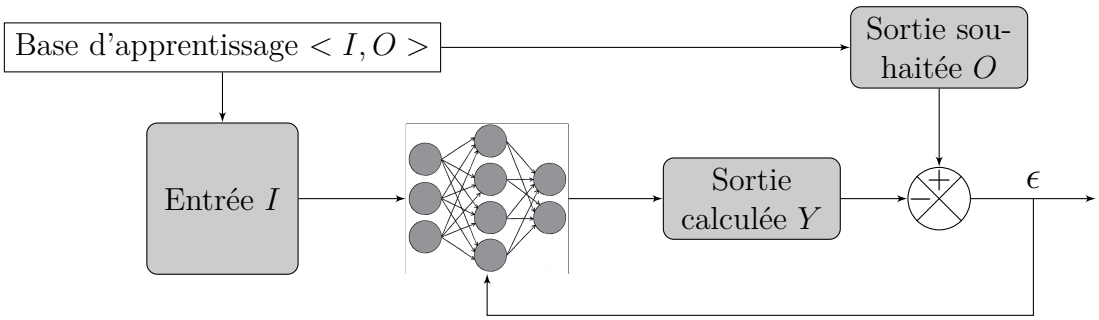

FIGURE 4.5 – Structure de l'apprentissage supervisé.

Le vecteur contenant les sorties souhaitées est formalisé par $O = (O_1, ..., O_q)$, tandis que ϵ représente l'erreur d'estimation commise par le réseau neuronal.

Le défi à relever pour mener à bien un apprentissage supervisé est de calculer l'influence de chaque coefficient synaptique du réseau sur l'erreur d'estimation commise en sortie, puis d'appliquer une règle de modification de ces coefficients pour améliorer le comportement global de l'estimateur neuronal.

4.2.3.2 Apprentissage non-supervisé

Lors d'un apprentissage non-supervisé, le réseau est laissé libre d'évoluer et de converger vers un état final. Les données d'entrées sont présentées et les poids synaptiques sont mis à jour selon une distribution probabiliste : les cas sont classifiés selon leur degré d'appartenance à un sous-ensemble. Le réseau doit déterminer par lui même quelle est la meilleure réponse possible à renvoyer (figure 4.6).

Ce type d'apprentissage est aussi appelé *auto-organisationnel* et est communément utilisé dans des applications de partitionnement, de détections d'anomalies, d'extraction de caractéristiques ou de réduction de dimensions.

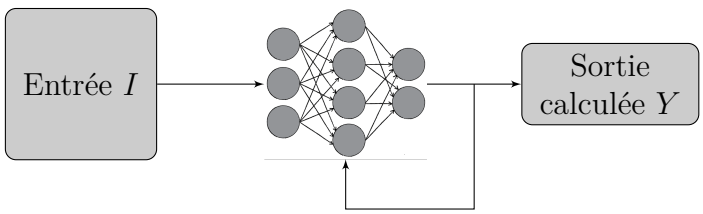

FIGURE 4.6 – Structure de l'apprentissage non-supervisé.

Ce type d'apprentissage est particulièrement employé lorsqu'aucun modèle du système à synthétiser n'est disponible ou que celui-ci est difficilement manipulable. Des exemples typiques d'utilisation seraient : *"A partir de données caractérisant l'évolution*

du courant circulant dans des résistances en fonction de la tension à leurs bornes, comment découvrir la loi d'Ohm ?" ou *"Considérant un ensemble de pages Web, comment classer efficacement ces sites ?"*

Il apparait donc que cette méthode d'apprentissage est plus hasardeuse que l'apprentissage supervisé puisqu'aucune référence n'est utilisée pour orienter la manière dont est réalisé l'apprentissage.

4.2.3.3 Apprentissage par renforcement

L'apprentissage par renforcement (AR) consiste à déduire une stratégie comportementale optimale à partir d'observations de l'état du système. A chaque itération n de l'apprentissage, l'agent apprenant effectue une action a_n depuis un état courant s_n. Cela conduit l'agent à un nouvel état s'_n et à recevoir une récompense r_n pour l'action effectuée.

L'agent apprenant va alors tenter de maximiser les récompenses reçues au cours du temps. La politique de décision est ainsi améliorée itérativement pour atteindre les objectifs de l'approximation (figure 4.7).

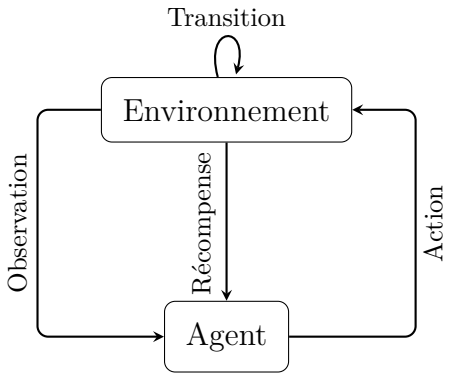

FIGURE 4.7 – Structure de l'apprentissage par renforcement.

En pratique, l'amélioration de la politique se fait par le biais d'une *fonction de valeur*, qui permet de quantifier l'intérêt qu'a l'agent d'effectuer une action a dans l'état s.

L'AR s'avère particulièrement utile pour certains types de problème où l'évolution de l'environnement est incertaine et lorsque les stratégies comportementales efficaces ne sont pas connues.

L'apprentissage par renforcement se distingue de l'apprentissage supervisé par le fait que, la récompense reçue n'indique pas à l'agent si l'action effectuée est optimale.

L'agent doit ainsi effectuer plusieurs actions dans un même état afin de déterminer quelle est la meilleure action possible. Il s'agit donc d'une méthode d'apprentissage par essais et erreurs.

4.2.3.4 Apprentissage *online* ou *offline*

Un apprentissage est qualifié d'incrémental ou online lorsque les données d'apprentissage sont reçues par le système apprenant au fur et à mesure au cours du temps.

A l'inverse, un apprentissage est qualifié de offline lorsque toutes les données nécessaires à l'apprentissage sont connues en amont de l'étape d'apprentissage.

Classiquement, un apprentissage offline ou *batch learning* est composé de deux étapes : une phase d'apprentissage sur les données disponibles puis une phase de test pour évaluer les performances de l'apprentissage sur une nouvelle série de données.

En revanche un apprentissage online adopte une structure itérative : à chaque itération un exemple est reçu par le réseau, puis une estimation est donnée et enfin la performance de prédiction est évaluée pour améliorer les réponses futures.

Le choix d'un apprentissage online ou offline dépend alors surtout de l'application, de la fréquence de disponibilité des données et de la durée d'apprentissage allouée.

4.2.3.5 Choix de la méthode d'apprentissage

Dans la suite des travaux de thèse, nous privilégions l'utilisation de l'apprentissage supervisé pour synthétiser le fonctionnement d'un système. D'une part pour sa simplicité d'implémentation et d'autre part pour la rapidité de convergence, lorsqu'un modèle du système est connu.

Puis lorsque le fonctionnement du système ne peut plus être prédit par un modèle ou est soumis à de trop fortes perturbations, l'utilisation de l'apprentissage par renforcement est envisagé puisque cette méthode permet de réaliser l'apprentissage d'un comportement optimal à partir d'un système de récompense.

L'apprentissage non-supervisé n'est donc pas considéré ici puisque ce type d'apprentissage n'est pas aussi approprié que l'apprentissage supervisé ou que l'apprentissage par renforcement pour synthétiser le fonctionnement d'un réseau ferroviaire en un temps de calcul raisonnable et avec une erreur d'estimation réduite.

4.3 Apprentissage d'un estimateur neuronal des flux de puissance sur un réseau DC

La synthèse d'un processus de résolution par un réseau neuronal a été de multiples fois effectuée dans la littérature. Ainsi, dans [144] , [146], [150], [151] [152] ou [153] un RNA est utilisé pour synthétiser une résolution de type *load flow* dans un système électrique. [140] choisit quant à lui d'effectuer un apprentissage pour contrôler les flux hydrauliques dans une pompe.

Il est intéressant de noter que le dénominateur commun de ces travaux est l'utilisation de l'apprentissage supervisé pour effectuer l'apprentissage du réseau neuronal.

4.3.1 Caractéristiques du problème à estimer

Comme il a été rappelé précédemment, l'apprentissage supervisé est la forme d'apprentissage la plus efficace quand on dispose d'un modèle du système à estimer.

La conception d'un estimateur neuronal se décompose en trois étapes : la création de la base de données, le paramétrage du réseau neuronal et l'implémentation de l'algorithme d'apprentissage.

Une architecture simplifiée de la problématique est résumée par la figure (4.8). Le terme $\Delta\omega_{ij}$ représente la matrice de modification des poids synaptiques.

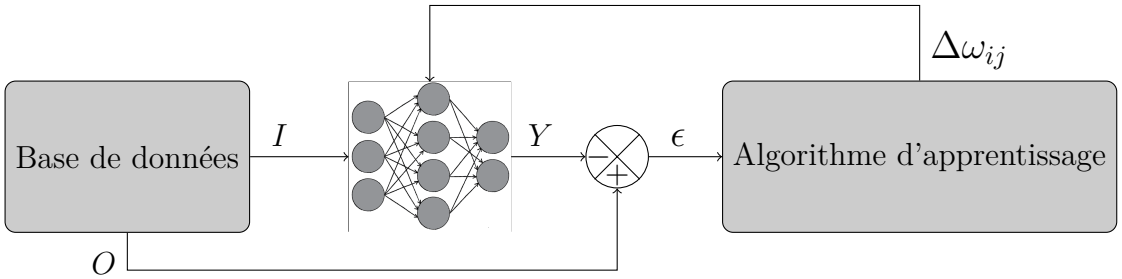

FIGURE 4.8 – Architecture simplifiée de la méthode d'apprentissage supervisé.

4.3.2 Constitution de la base de données

4.3.2.1 Modélisation et simulation des cas d'apprentissage

La constitution de la base de données d'apprentissage se fait par simulation des points de fonctionnement du système que l'on souhaite faire apprendre au réseau neuronal.

Pour cela, il est nécessaire de modéliser puis de simuler le fonctionnement énergétique de la ligne de métro dans différentes conditions d'exploitation afin de caractériser les flux de puissance électrique générés par le déplacement des trains.

Les entrées I du système à synthétiser sont définies comme les positions X_i et les puissances électriques théoriques ${P^{th}}_i$ des i trains composant le carrousel tandis que les sorties O sont les puissances électriques réelles ${P^r}_i$ consommées/renvoyées par les trains : $I = [X_i, {P^{th}}_i]$ et $O = [{P^r}_i]$, $\forall i \in [\![1, N_{trains}]\!]$

4.3.2.2 Segmentation de la base d'apprentissage

Dans le cadre d'un apprentissage supervisé, une base de données est classiquement composée de trois sous-ensembles : une base d'apprentissage, une base de test et une base de validation.

La base d'apprentissage concentre les cas qui seront appris par le réseau neuronal, la base de validation a pour but d'une part de vérifier l'évolution de l'erreur d'apprentissage et d'autre part de s'assurer qu'il n'y ait pas de sur-apprentissage durant le processus et enfin, la base de test est utilisée pour évaluer l'erreur d'estimation commise par le réseau obtenu après apprentissage.

La base de test peut alors avoir comme intérêt de permettre une comparaison objective entre différentes architectures de réseau neuronal ayant appris la même base d'apprentissage.

L'appartenance des données à l'un ou l'autre de ces sous-ensembles est définie aléatoirement pour ne pas biaiser l'estimation et permettre une répartition des données selon une loi normale.

4.3.3 Paramétrage du réseau neuronal

4.3.3.1 Paramétrage de l'apprentissage

La mise en œuvre d'un algorithme de rétropropagation pour l'apprentissage d'un réseau de neurones nécessite de régler un certain nombre de paramètres.

Le nombre de couches : ce paramètre doit être choisi par essais et erreurs puisque l'architecture d'un perceptron est largement dépendante du problème à approcher et des données disponibles dans la base d'apprentissage. [154] a ainsi montré qu'un perceptron possédant deux couches cachées est capable d'approximer n'importe quelle fonction non-linéaire et de constituer des sous-ensembles pour n'importe quel problème de classification.

Le nombre de neurones par couche influence directement la capacité de généralisation du RNA et le temps de calcul nécessaire à la rétropropagation du gradient de l'erreur d'estimation. En effet, plus un réseau possède de neurones plus il est capable de décrire précisément un phénomène, mais parallèlement le temps de calcul augmente puisque chaque neurone d'une couche i impose de devoir calculer N_j coefficients synaptiques et donc N_j opérations de dérivations à chaque itération (où N_j est le nombre de neurones composant la couche suivante).

Les fonctions d'activation : le choix des fonctions d'activation utilisées par chaque couche du RNA est dépendant de l'espace de définition des sorties souhaitées. Ainsi, comme l'explique [155], l'utilisation d'une fonction sigmoïde est très populaire car elle produit des sorties dont la moyenne est nulle. De la même manière, la fonction tangente hyperbolique produit des sorties strictement positives, ce qui peut être une caractéristique importante dans certains cas. Le choix des fonctions d'activation doit donc faire l'objet soit d'une analyse poussée sur les sorties souhaitées au niveau de chaque couche, soit d'un grand nombre d'essais et d'erreurs pour mettre en évidence l'architecture la plus adaptée au problème.

Initialisation de poids synaptiques : généralement, les poids synaptiques sont tirés aléatoirement sur l'intervalle $]-1;1[$, mais il est possible de réduire ce domaine pour ajuster l'apprentissage aux fonctions d'activation choisies. De la même manière que pour le choix des fonctions d'activation, la plage d'initialisation des poids synaptiques est largement dépendante de la nature du problème à synthétiser.

Dans la littérature, malgré les études complètes sur le sujet comme celle de [155], il n'existe aucune règle absolue pour déterminer l'architecture optimale d'un perceptron.

4.3.3.2 Construction et élagage

Une méthode éprouvée pour déterminer la structure optimale d'un PMC consiste à comparer les performances de différentes architectures par essais et erreurs ; cependant, cette méthode présente des limites lorsque la taille de la base de données augmente puisque la durée d'apprentissage croît également.

Dans la littérature, deux méthodes ont été proposées pour déterminer l'architecture optimale d'un réseau de neurones pour réaliser un objectif donné.

Les méthodes de construction. A partir d'une architecture simple, des neurones ou des couches cachées sont ajoutés au cours de l'apprentissage pour accélérer la diminution de l'erreur d'apprentissage [156]. L'architecture du réseau est alors