

CHAPITRE 7

ESTIMATION DES COÛTS DE PRODUCTION PAR APPRENTISSAGE

Sommaire

7.1	Introduction	207
7.2	Schémas d'apprentissage	209
7.2.1	Construction du réseau SIMPLE pour la prédiction du coût de production	210
	Description des entrées du réseau de neurones	210
	Description de l'architecture du réseau de neurones	210
	Description des sorties du réseau de neurones	212
7.2.2	Construction du réseau MIXTE pour la prédiction du coût de production et du numéro de période de la dernière recharge	212
	Description des entrées du réseau de neurones	212
	Description de l'architecture du réseau de neurones	213
	Description des sorties du réseau de neurones	215
7.3	Schémas d'apprentissage dont les entrées sont des indicateurs	215
7.3.1	Indicateurs	216
7.3.2	Construction du réseau INDIC_TEMPS pour la prédiction du numéro de période de la dernière recharge	218
	Description des entrées du réseau de neurones	218
	Description de l'architecture du réseau de neurones	219
	Description des sorties du réseau de neurones	220
7.3.3	Construction du réseau INDIC_COUT pour la prédiction du coût de production	220
	Description des entrées du réseau de neurones	220
	Description de l'architecture du réseau de neurones	221
	Description des sorties du réseau de neurones	221
7.4	Expérimentations numériques	222
7.4.1	Objectifs et contexte technique	222
7.4.2	Instances	223
	Analyse statistique des instances	224
7.4.3	Moyenne de la valeur absolue des gaps	227
7.4.4	Résultats des réseaux SIMPLE_TYPE et SIMPLE_PERIODE	229
	Résultats du réseau SIMPLE_TYPE	230

Résultats du réseau SIMPLE_PERIODE	231
7.4.5 Résultats du réseau MIXTE_COUT pour la prédiction du coût de production . .	233
7.4.6 Résultats du réseau INDIC_COUT pour la prédiction du coût de production . .	233
7.4.7 Résultats du réseau MIXTE_TEMPS pour la prédiction du numéro de période de la dernière recharge	234
7.4.8 Résultats du réseau INDIC_TEMPS pour la prédiction du numéro de période de la dernière recharge	235
7.5 Conclusion	237

Noms	Significations
C^{Tank}	Capacité de la citerne d'hydrogène
C^{Veh}	Capacité du réservoir d'hydrogène du véhicule
N	Nombre de périodes de production
H_0	Charge initiale de la citerne d'hydrogène
$Cost^F$	Coût d'activation de la micro-usine
Pour $i = 0, \dots, N - 1, R_i$	Rendement de production lié à la période i
Pour $i = 0, \dots, N - 1, Cost_i^V$	Coût de production lié à la période i
Q	Nombre d'opérations de recharge en carburant effectuées par le véhicule
Pour $q = 1, \dots, Q, \mu_q$	Quantité d'hydrogène qui est rechargée lors de la $q^{ième}$ opération de recharge
Pour $q = 1, \dots, Q, [m_q, M_q]$	Fenêtre de temps de la $q^{ième}$ opération de recharge

TABLE 7.1 – Entrées utilisées pour construire les schémas d'apprentissage.

7.1 Introduction

Dans ce chapitre, on présente un outil d'aide à la décision construit avec les schémas d'apprentissage qui pourra estimer rapidement le coût de production associé à une stratégie de recharge réduite décrite à la section (6.5.5) du chapitre précédent. Autrement dit, on construit donc ici un estimateur rapide des coûts de production. Cet outil pourrait servir à aider un décideur à prendre des décisions sur des stratégies de recharge réduite afin de limiter les risques de perte (de temps ou d'argent). Par exemple le décideur pourrait avoir plusieurs stratégies de recharge réduite et voudrait décider rapidement quelle stratégie réaliser.

Cet estimateur pourrait aussi par exemple servir à optimiser la tournée du véhicule. Dans les chapitres précédents, on a présenté plusieurs méthodes de résolution du problème SMEPC. On a notamment présenté au chapitre 6 un schéma collaboratif nommé Pipe-line VD_PM. Car il a été constaté que le problème SMEPC peut être divisé en deux sous-problèmes à savoir le problème *Vehicle-Driver* et le problème de production *Production-Manager*. De nos expérimentations numériques il ressort que parmi les deux algorithmes de résolution des deux sous-problèmes, celui qui est lourd en temps CPU est l'algorithme de résolution du problème *Production-Manager*. Dans les chapitres précédents, on a fixé la tournée du véhicule, autrement dit on a supposé qu'on connaît dans quel ordre les stations seront visitées car notre objectif était de se focaliser uniquement sur les aspects synchronisation du problème SMEPC. L'outil d'aide à la décision construit dans ce chapitre pourrait être utilisé pour optimiser la tournée du véhicule. Pour évaluer le coût de production, on pourrait bien évidemment utiliser le module PM de Pipe-line VD_PM mais ce module étant lourd en temps CPU, on veut construire un réseau plus rapide qui peut remplacer ce module et évaluer le coût de la partie production PM. Pour cela, on peut utiliser cet outil qui nous permettrait d'obtenir des approximations de la qualité d'une tournée très rapidement.

Les entrées du modèle SMEPC que nous utiliserons pour construire les schémas d'apprentissage pour le problème de production sont présentées au tableau (7.1).

Les schémas d'apprentissage que nous avons choisi d'explorer ici sont les réseaux de neurones.

se différenciant

Il existe plusieurs types de réseaux de neurones. Ces réseaux de neurones ~~sont différentiables~~ par la manière dont les informations sont propagées entre les différentes couches de neurones. La variante la plus simple est celle du réseau de neurones dit *feed-forward*, ici, les informations passent directement de la couche d'entrée aux couches cachées puis à la couche de sortie. Dans ce chapitre, on s'intéresse uniquement à ce type de réseau de neurones. Mais il existe d'autres types de réseaux de neurones à savoir les réseaux de neurones récurrents et les réseaux de neurones convolutifs. Les réseaux de neurones récurrents sauvegardent les résultats produits par les neurones et nourrissent le réseau à l'aide de ces résultats. Ce mode d'apprentissage est un peu plus complexe. Les réseaux de neurones convolutifs sont quant à eux de plus en plus utilisés dans différents domaines : reconnaissance d'images, traitement naturel du langage. En reconnaissance d'images par exemple, un réseau de neurones convolutifs est constitué de deux parties : une première partie dite convolutive qui est chargée de faire la convolution (réduction de la dimension) des images et une deuxième partie classification du réseau qui correspond à un réseau Perceptron Multicouche dont l'acronyme est MLP (*Multi Layers Perceptron*). Un MLP est un réseau à propagation *feed-forward* constitué de plusieurs couches.

Le nombre de données d'entrée et les données de sortie du réseau de neurones sont généralement fixés par la nature du problème. Le nombre de couches cachées et le nombre de neurones par couche n'est pas facile à déterminer, cela dépend ~~principale~~ de la quantité et de la complexité des données. En général, quand on augmente le nombre de neurones cachés, on gagne de la précision sur les données utilisées en apprentissage mais le réseau perd sur son pouvoir de généralisation pour d'autres données, d'où l'expression sur-apprentissage. Ainsi, plus le nombre de couches et de neurones est élevé, plus le réseau aura besoin d'une grande quantité de données pour être entraîné efficacement. Une architecture qui donne de bons résultats pour une application donnée ne peut être déterminée que d'une façon expérimentale. Pour construire un réseau de neurones, on a besoin de fixer un certain nombre de paramètres. Parmi eux, on peut citer la fonction d'activation, le nombre d'itérations et la taille du batch. La fonction d'activation permet de normaliser les sorties de neurones dans un intervalle prédéfini. Par exemple, les sorties de neurones peuvent être échelonnées sur un intervalle $[0,1]$ par la fonction sigmoïde : $f(x) = \frac{1}{1 + e^{-x}}$. Le nombre d'itérations représente le nombre de fois que l'ensemble de données sera passé au réseau de neurones. Plus le nombre d'itérations est grand, plus votre réseau sera entraîné longtemps, ce qui vous donnera également de meilleurs résultats. La taille du batch représente le nombre d'échantillons qui passeront par le réseau de neurones à chaque cycle d'apprentissage.

Au début de la phase d'apprentissage, les données d'apprentissage et de validation sont présentées au réseau avec la valeur de sortie (le cout de production) correspondantes. Les valeurs des poids sont ajustées et affinées continuellement tout au long de la phase d'apprentissage. La correction des poids au cours de l'entraînement ne tient compte que des données d'apprentissage. Au cours de cette phase, les poids du réseau sont corrigés de manière à minimiser l'erreur au carré entre la réponse calculée par le réseau et la réponse attendu. Généralement, l'erreur calculée sur les données d'apprentissage diminue continuellement au cours de l'entraînement. Toutefois, une longue phase d'entraînement diminue la capacité de généralisation du réseau en l'adaptant uniquement aux données d'apprentissage. À cet effet, les données de validation déterminent à quel moment l'apprentissage doit être arrêté. Les données de validation représente généralement 1/3 des données d'apprentissage. Les données de validation servent uniquement à vérifier le comportement du réseau au cours de l'entraînement face à des données qui lui sont étrangères. Généralement, contrairement à l'erreur calculée sur les données d'ap-

apprentissage qui diminue continuellement au cours de l'entraînement, celle calculée sur les données de validation diminue dans la première phase d'entraînement en suivant une allure semblable à celle des données d'apprentissage avant de commencer une lente ascension. Ceci est expliqué par le fait que le réseau commence à perdre son pouvoir de généralisation en adaptant ses neurones uniquement aux données d'apprentissage. L'entraînement du réseau sera donc arrêté dès que cette erreur commence son ascension. Toutefois, afin d'éviter un arrêt prématuré d'apprentissage causé par une augmentation ponctuelle de l'erreur des données de validation, on introduit souvent un seuil de décision qui tolère des légères ascensions successives de l'erreur. Si l'erreur validation continue son ascension au delà de ce seuil, on arrête l'apprentissage du réseau et on conserve les valeurs des poids qui correspondent à l'itération qui précède cette ascension.

Après l'arrêt de la phase d'apprentissage, on vérifie la performance du réseau avec les données de test. Ces données n'ont pas servi à l'apprentissage et n'ont joué aucun rôle dans la prise de décision dans l'arrêt de l'apprentissage. Les données de test sont utilisées uniquement pour mesurer la performance du réseau après l'arrêt de l'apprentissage. Si le réseau arrive à faire des prédictions correctes avec un gap acceptable, on peut dire que le réseau est opérationnel. Dans le cas contraire, il faut réviser le réseau et recommencer l'apprentissage.

Dans la section 7.2 On présente deux réseaux de neurones. l'un est nommé **SIMPLE** et l'autre **MIXTE**. Le réseau **MIXTE** contrairement au réseau **SIMPLE**, essaye d'épouser les caractéristiques du problème de production. Puisque les données d'entrées des ces réseaux de neurones sont nombreuses cela a pour effet d'augmenter considérablement le nombre de poids de nos réseaux neurones. Pour pallier à ce problème, on va définir un ensemble d'indicateurs qui vont représenter nos données de façon plus compacte. Dans la section 7.3, on commence par présenter les indicateurs qui serviront d'entrée aux réseaux de neurones afin de faire une approximation du coût économique. On finit cette section en présentant l'architecture de nos réseaux de neurones. Dans la section 7.4, on montre les résultats des expérimentations numériques en présentant au préalable les instances qui ont été utilisées pour entrainer nos réseaux de neurones.


7.2 Schémas d'apprentissage



Cette section présente la méthodologie suivie pour la conception et l'apprentissage des schémas d'apprentissage pour la prédiction du coût de production correspondant à une stratégie de recharge réduite. Plus précisément, on décrit un réseau qu'on nomme **SIMPLE** et un réseau qu'on nomme **MIXTE** en définissant les éléments suivants : le format des données d'entrées, l'architecture des réseaux de neurones, la fonction d'activation et l'algorithme d'apprentissage. Les réseaux de neurones de cette section ont été implémenté en python à l'aide de Keras qui dispose de plusieurs modules de création et d'entrainement de réseaux de neurones.

Tout au long de ce chapitre, nous utiliserons les expressions suivantes :

- *Layer Dense* correspond à des couches dont les neurones reçoivent en entrée une valeur égale à la somme pondérée de l'ensemble des neurones de la couche précédente ;
- *Multiply Dense* permet de multiplier la valeur de sortie de 2 couches de neurones terme à terme.
- *Layer Concatenate* recopie les neurones de leur couche initiale vers la couche de concaténation sans modification de leur valeur de sortie

7.2.1 Construction du réseau SIMPLE pour la prédiction du coût de production

Dans cette section, notre objectif est de construire un réseau de neurones **simple** qui servira de base pour la comparaison avec les modèles plus sophistiqués. Concernant les entrées du réseau, nous allons tester deux façons de les agencer qu'on nomme ici agencement par période et par type.  remment dit, on a classé les données par type et par période. Par exemple **si les données sont les valeurs 10 10 10 pour le rendement et 1 1 1 pour le coût de production**, alors les données classées par type sont agencées de la façon suivante 10 10 10 1 1 1 et les données classées par période sont agencées de la façon suivante 10 1 10 1 10 1. Concernant les fonctions d'activation, on va tester plusieurs combinaisons pour voir celles qui fonctionnent le mieux pour notre problème.

Pour se familiariser avec les réseaux de neurones, on construit un réseau qu'on  me **SIMPLE**. On suppose ici que la sortie qu'on veut calculer **est une combinaison linéaire de nos entrées**. Ceci étant dit, il est fort probable que le réseau **SIMPLE** ne fournisse pas de bons résultats, c'est-à-dire ne fasse pas de bonnes approximations. 

Dans ce **perceptron multicouche**, les entrées de chaque couche sont les sorties de la couche précédente. Et chaque neurone de chaque couche est connecté à tous les neurones de la couche précédente **au cas où c'est un réseau complètement connecté**. Dans cette section, on va décrire les données d'entrée, l'architecture et les données de sortie de notre réseau.

Description des entrées du réseau de neurones **SIMPLE**

Les données utilisées lors de la phase d'entraînement des réseaux de neurones doivent être pertinentes pour la prédiction du coût de production. Dans notre cas, les données d'entrée seront les suivantes es.

- ☐ $Cost^F$: Coût d'activation de la micro-usine. On construira un vecteur A_i de taille N contenant le coût fixe;
- ☐ Pour $i = 0, \dots, N-1$, $Cost_i^V$: Coût de production lié à la période i ;
- ☐ Le numéro de période du début de la dernière recharge
- ☐ Pour $i = 0, \dots, N-1$, R_i : Rendement de production lié à la période i ;
- ☐ Un vecteur *Recharge* rempli de la façon suivante : Pour $q = 1, \dots, Q$, μ_q et $i \in [m_q, M_q]$, $Recharge[i] = \mu_q$ sinon $Recharge[i] = 0$.

~~On considère que la taille de notre réseau est la plus grande valeur de N .~~

Description de l'architecture du réseau de neurones **SIMPLE**

Ce réseau de neurones est complètement connecté c'est-à-dire que tous les neurones d'une couche sont connectés à tous les neurones de la couche précédente. Dans notre modèle, le nombre de neurones des couches a été choisi arbitrairement. La fonction d'agrégation utilisée est la somme pondérée. L'architecture du réseau de neurones (Voir figures (7.1) et (7.2)) que nous avons conçu est la suivante :

- ☐ Elle a une couche d'entrées constituée de huit neurones
- ☐ Elle a une couche cachée constituée de quatre neurones



- Elle a une couche de sortie constituée d'un neurone

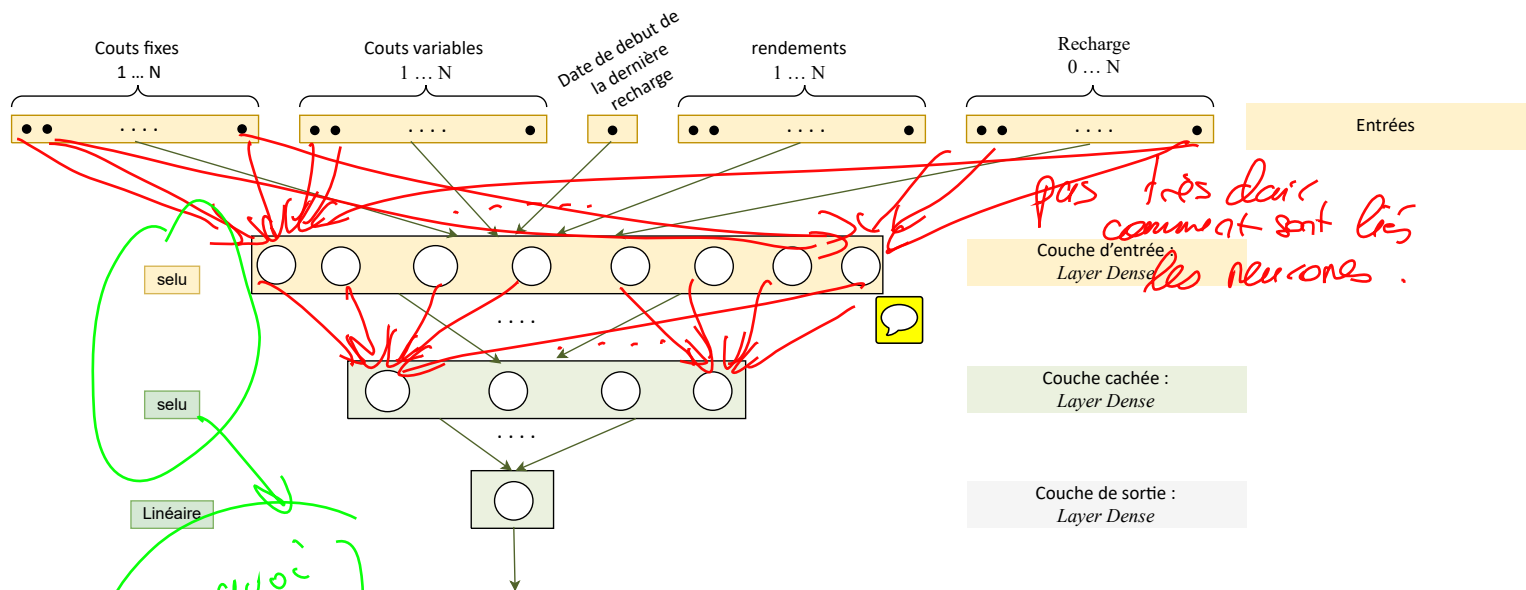


FIGURE 7.1 – Réseau **SIMPLE_TYPE** prédisant le coût de production.

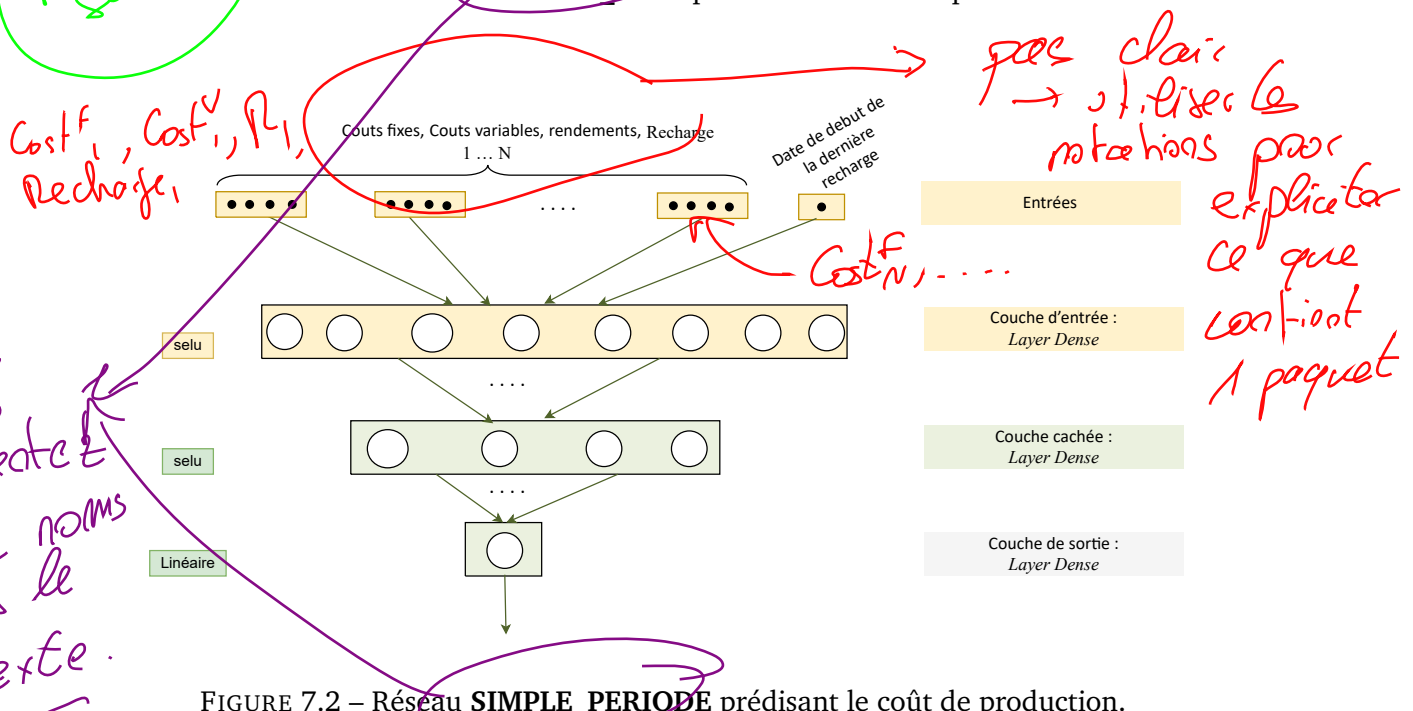


FIGURE 7.2 – Réseau **SIMPLE_PERIODE** prédisant le coût de production.

Le nombre de poids et de biais du réseau de neurones **SIMPLE** dépend de la taille des instances, on a : $8 \times (4N + 2)$ poids à la couche d'entrée et 8 biais. on a $4 \times 8 = 32$ poids et 4 biais à la couche cachée. On a 4 poids à la couche de sortie et 1 biais.

Un taux d'apprentissage indique la vitesse à laquelle les coefficients évoluent. Plus α sera proche de 1, mieux on corrigera les poids. autrement dit, les poids varieront avec une assez grande amplitude, puisqu'un exemple pourra augmenter nettement les poids alors qu'un autre pourrait les diminuer nettement. L'avantage d'un α proche de 1 est que, s'il n'est pas trop proche de 1, il fera converger assez rapidement vers des poids corrects. Le désavantage, c'est que justement s'il est trop proche de 1, on pourrait dans de nombreux cas faire osciller les poids sans jamais converger, ou alors au bout d'un milliard d'itérations sur la base d'exemples. Si α s'éloigne de 1, il corrigera petit à petit les

pourquoi 1 milliard ?

→ où intervient ce coefficient exactement ?

poids sur chaque exemple, sans une grosse variation d'amplitude. L'avantage d'un α éloigné de 1 est que, s'il n'est pas trop éloigné, il fera converger petit à petit mais assez rapidement les poids vers de bonnes valeurs. Le désavantage c'est que s'il est trop éloigné de 1, les poids varieront tellement doucement qu'on risque de ne jamais les voir converger vers de bonnes valeurs, ou alors encore une fois au bout de milliards d'itérations.

pas d'air non plus L'algorithme d'optimisation utilisé est Adam. Adam est différent de la descente de gradient stochastique classique. La descente de gradient stochastique maintient un taux d'apprentissage ($\alpha \in [0, 1]$) unique pour toutes les mises à jour de poids et le taux d'apprentissage ne change pas pendant l'entraînement. Il est maintenu pour chaque poids de réseau (paramètre) et adapté séparément à mesure que l'apprentissage se déroule. La méthode calcule des taux d'apprentissage adaptatifs individuels pour différents paramètres à partir d'estimations des premiers et second moments des gradients.

Pour finir, la fonction de perte utilisée est le carré des erreurs. Les données de validation sont 1/3 des données d'apprentissage. Le nombre d'itérations fixé est 200. La taille des batchs fixés ici est 32.

Description des sorties du réseau de neurones *SIMPLE*

La sortie du réseau de neurones **SIMPLE** est le coût de production.

Il y a 2 réseaux simple.

Le type d'architecture de réseau de neurones construit dans cette section est limité en terme de topologie modèle. Il existe un autre type d'architecture qui offre plus de flexibilité dans la conception de la topologie du modèle. Avec cette architecture, on peut construire des réseaux de neurones plus complexes tels que des réseaux *multi-input/multi-output*, des graphiques acycliques orientés (~~ils ne possèdent pas de circuits~~), et des réseaux à couches partagées. Dans la section suivante, on va construire un réseau **MIXTE** qui épouse la logique de notre problème.

7.2.2 Construction du réseau MIXTE pour la prédiction du coût de production et du numéro de période de la dernière recharge

? Le réseau **MIXTE** offre plus de flexibilité pour structurer le réseau de neurones à notre convenance. Avec ce type d'architecture, on peut avoir plusieurs couches d'entrées et plusieurs couches de sorties. Notre objectif ici est de construire un réseau de neurones qui épouse les mécanismes de calcul et les caractéristiques du problème de production. Dans cette section, on va décrire les données d'entrée, l'architecture et les données de sortie de notre réseau.

Description des entrées du réseau de neurones *MIXTE*

Les entrées du réseau de neurones **MIXTE** sont :

- ☐ $Cost^F$: Coût d'activation de la micro-usine. On construira un vecteur de taille N contenant le coût fixe ;
- ☐ Pour $i = 0, \dots, N - 1$, $Cost_i^V$: Coût de production lié à la période i ;
- ☐ Pour $i = 0, \dots, N - 1$, R_i : Rendement de production lié à la période i ;

- N périodes $1, \dots, N$ et 2 périodes fictives $0, N+1$; Q recharges + 2 recharges fictives $(0, Q+1)$ qui modélisent les niveaux de départ et d'arrivée de la citerne; μ_1, \dots, μ_Q : quantité à recharger par le véhicule; F_1, \dots, F_Q fenêtres de temps pour les recharges ($F_q = [m_q, M_q]$); L_1, \dots, L_Q longueurs des fenêtres (NB. dans les instances actuelles on suppose connaître exactement les périodes de recharge donc $L_i = 1 \forall i = 1, \dots, N$); $\forall q = 1, \dots, Q, \mu_q^* = \mu_q / L_q$ (NB. actuellement on aura $\mu_q^* = \mu_q$)

$\forall i \in 1, \dots, N, \lambda_i = \sum_{q=1, \dots, Q, \forall i \in F_q} \mu_q^*$ (permet de se ramener à un vecteur indicé sur les périodes)

- $\lambda_0 = -H_0$ et $\lambda_{N+1} = H_0$

Exemple 10 Calcul du λ : $N = 6, Q = 2$ (5 périodes et 2 recharges)

$$F_1 = [2, 3], F_2 = [3, 6], \mu_1 = 8, \mu_2 = 20, L_1 = 2, L_2 = 4, H_0 = 3$$

$$\mu_1^* = 4, \mu_2^* = 5$$

$$\lambda_0 = -3, \lambda_7 = 3 \text{ (quantités liées aux périodes fictives)} \quad \lambda_1 = 0, \lambda_2 = \mu_1^* = 4, \lambda_3 = \mu_1^* + \mu_2^* = 9, \lambda_4 = \mu_2^* = 5, \lambda_5 = \mu_2^* = 5, \lambda_6 = \mu_2^* = 5$$

La matrice des entrées étant une matrice creuse car les entrées sont de taille différente, on la remplira de façon cyclique (en répétant les valeurs du début à la fin autant de fois qu'il faut pour atteindre la taille maximal) afin que les données forment une matrice rectangulaire. C'est cette opération qui permettra de connaître exactement le nombre de valeurs en entrées. ~~On considère que la taille de notre réseau est la valeur de N maximale.~~

Description de l'architecture du réseau de neurones

MIXTE

Notre objectif est de représenter avec un réseau de neurone la formule de calcul du coût de production. ~~Pour commencer, on normalise nos entrées.~~ On les divise d'abord en deux catégories à savoir les prix ($Cost_i^V, Cost^F$) et les valeurs qui désignent les quantités d'hydrogène (R_i, λ_i). Puis, on normalise chaque catégorie avec la fonction sigmoïde. On sait que le coût de production est constitué de deux composantes à savoir le coût d'activation de la micro-usine (coût fixe) et le coût de production variable. On sait que les vecteurs désignant les périodes de production et les périodes de démarrage de la micro-usine sont des vecteurs de booléens. Pour chaque période i , on veut connaître la probabilité γ_i qu'elle soit une période de production et la probabilité γ_i^* qu'elle soit une période de démarrage de la micro-usine. Une fois qu'on connaîtra les probabilités γ_i et γ_i^* , il suffira de les multiplier (à l'aide d'une couche *Multiply Dense*) respectivement par le coût variable $Cost_i^V$ et par le coût fixe $Cost^F$ puis de sommer les valeurs de toutes les périodes pour obtenir le coût de production car la formule de calcul du coût de production est $COST = \sum_{i=0, \dots, N-1} \gamma_i \times Cost_i^V + \gamma_i^* \times Cost^F$. La probabilité γ_i est calculée avec la fonction sigmoïde car on peut avoir plus d'une période de production. La probabilité γ_i^* se déduit du vecteur des γ_i car si on connaît les périodes de production, on peut aisément déduire quelles sont les périodes d'activation de la micro-usine. Pour le calcul de la probabilité τ_i que i soit la dernière période de recharge on utilise la fonction softmax car on a qu'une unique dernière période de recharge donc il faut que $\sum_{i=0, \dots, N-1} \tau_i = 1$. On veut éviter que le calcul du numéro de période de la dernière recharge $T = \sum_{i=1, \dots, N} i \times \tau_i$ soit faussé. L'architecture du réseau de neurones (Voir figure (7.3)) que nous avons conçu est la suivante :

- ☐ Elle a deux couches d'entrées constituées chacune de N neurones ;
- ☐ Elle a quatre couches cachées constituées chacune de N neurones ;
- ☐ Elle a deux couches de sortie constituée l'une de N neurones et l'autre d'un neurone.

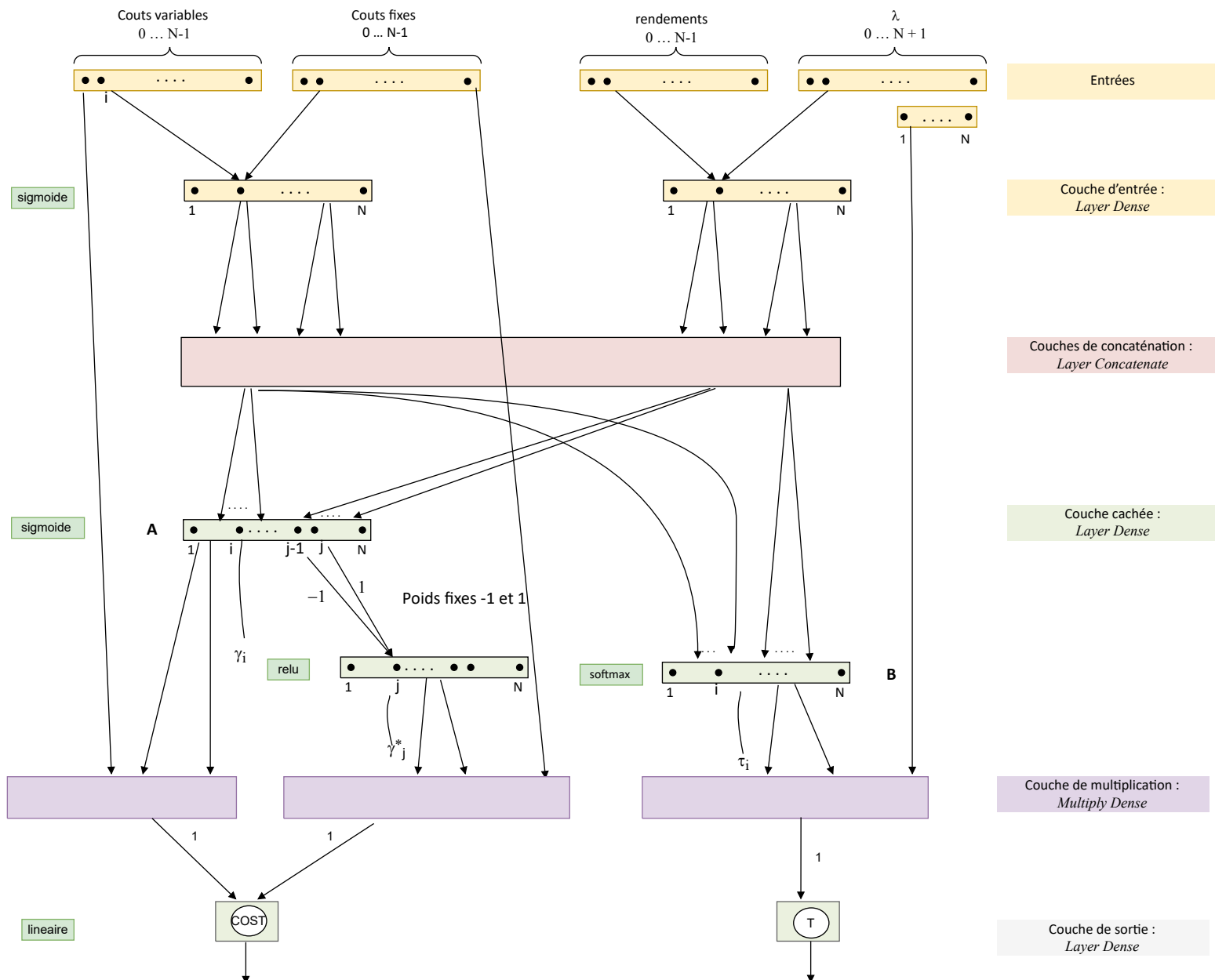


FIGURE 7.3 – Réseau **MIXTE** prédisant le coût de production et le numéro de période de la dernière recharge.

Les neurones du réseau ici ne sont pas complètement connectés c'est-à-dire que les neurones d'une couche sont connectés à certains neurones de la couche précédente. Le nombre de neurones des couches a été choisi en fonction du nombre de période. La fonction d'agrégation utilisée est la somme pondérée. Les fonctions d'activation utilisées sont la fonction sigmoïde, la fonction softmax, la fonction RELU (Elle vaut x si x est supérieur à 0, et 0 sinon. Autrement dit, c'est le maximum entre x et 0) et la fonction linéaire.

Le nombre de poids et de biais du réseau de neurones **MIXTE** est **quadratique** en fonction de N

et de la taille des instances. Le nombre de poids non fixés est : $2 \times N \times N + 2 \times N \times N + N + N + 2 \times N \times N + 2 \times N \times N = 8N^2 + 2N$. L'algorithme d'optimisation utilisé est Ada. Les données de validation sont 1/3 des données d'apprentissage. Le nombre d'itérations fixé ici est 2000. La taille des batch fixé ici est 32. La fonction de perte utilisé est le carré des erreurs. La Fonction d'erreur est :

$Err = (V - V_{th})^2$ avec V_{th} qui représente le coût théorique (prédit) de la solution

$$V = \alpha(\sum_{i=1,\dots,N} i \times \tau_i) + \beta(\sum_{i=1,\dots,N} Cost_i^V \times \gamma_i + \sum_{i=1,\dots,N} A_i \times \gamma_i^*)$$

Le numéro de période de la dernière recharge est $\sum_{i=1,\dots,N} i \times \tau_i$ et le coût de production est $\sum_{i=1,\dots,N} Cost_i^V \times \gamma_i + \sum_{i=1,\dots,N} A_i \times \gamma_i^*$.

Lorsqu'on ne veut traiter que la partie production avec le réseau de neurones de la figure (7.3) on supprime de la couche B à la couche de sortie du modèle. On nomme le réseau ainsi obtenu **MIXTE_COUT**. Lorsqu'on ne veut traiter que la partie recharge avec le réseau de neurones de la figure (7.3) on supprime de la couche A à la couche de sortie du modèle. On nomme le réseau ainsi obtenu **MIXTE_TEMPS**.

Description des sorties du réseau de neurones

La sortie de ce réseau de neurones est un vecteur de taille $3N$, leur signification est :

- ☐ $\gamma_i \in [0, 1], i = 1, \dots, N$: la probabilité que i soit une période de production
 - ☐ $\gamma_i^* \in [0, 1], i = 1, \dots, N$: la probabilité que i soit une période de démarrage de la micro-usine
 - ☐ $\tau_i^* \in [0, 1], i = 1, \dots, N, \sum_i \tau_i = 1$ probabilité période i d'être la période de dernière recharge
- γ_i et γ_i^* sont agrégés pour calculer le coût de production.

Dans la section suivante, on va construire un autre schéma d'apprentissage orienté réseau de neurones.

7.3 Schémas d'apprentissage dont les entrées sont des indicateurs

Dans les schémas d'apprentissage précédent, le nombre de poids et de biais du réseau dépend fortement de la taille des instances. Donc si on a des entrées de grande taille on aura un grand nombre de poids et de biais. Pour éviter que la taille de nos réseaux de neurones dépende de la taille des instances on définit dans cette section un ensemble d'indicateurs. Dans cette section, on compacte les entrées de nos réseaux de neurones en un ensemble d'indicateurs de telle sorte que le réseau ait une taille fixe. Autrement dit, on essaye d'agréger les entrées. On présente les réseaux de neurones dont les entrées sont des indicateurs. Ces réseaux seront utilisés pour apprendre la valeur du coût de production. Les entrées du modèle SMEPC que nous utiliserons pour construire nos schémas d'apprentissage pour le problème de production sont présentées au tableau (7.1). En plus de ces données, nous aurons aussi besoin des données suivantes :

- ☐ Le coût d'activation $A = A_i$, avec $i \in \{0, \dots, N - 1\}$ puisque dans notre problème le coût d'activation est $Cost^F$, on pose ici $\forall i, A_i = Cost^F$;
- ☐ Q est le nombre d'opérations de recharge en carburant effectuées par le véhicule ;

il faut factoriser pour tous les réseaux et faire 1 tableau avec tout -

- ☐ μ_q est la quantité d'hydrogène qui est rechargée lors de la $q^{i\text{eme}}$ opération de recharge ;
- ☐ Des bornes inférieures m_1, \dots, m_Q respectivement pour les numéros de période $i_1, \dots, i_Q \in \{0, \dots, N-1\}$ lorsque les opérations de recharge en hydrogène ont lieu ;
- ☐ Des bornes supérieures M_1, \dots, M_Q respectivement pour les numéros de période $i_1, \dots, i_Q \in \{0, \dots, N-1\}$ lorsque les opérations de recharge en hydrogène ont lieu ;
- ☐ Le coût unitaire de production est $P_i = \text{Cost}_i^V / R_i$
- ☐ L'écart du coût unitaire de production est $G_i = |P_{i+1} - P_i|$.

7.3.1 Indicateurs

Dans cette section, on définit des indicateurs qui permettent de faire une approximation du coût de production et du temps de parcours du véhicule. On va définir des indicateurs basés sur les moyennes de nos données. On veut calculer des quantités agrégées qui auront un impact prévisible sur le coût de production. Ces quantités agrégées seront utilisées comme entrées de nos réseaux de neurones afin que la taille de ceux-ci ne dépende plus de la taille des instances. On veut donc avoir des réseaux de taille fixe. Les indicateurs encadrés sont ceux qui sont utilisés comme entrées sans aucune modification. Nous introduisons plusieurs indicateurs comme suit :

☐ Indicateurs énergétiques :

- $Mu = \sum_q \mu_q$: Mu est la quantité totale d'hydrogène rechargée. Si la valeur de Mu augmente alors la valeur du coût de production augmente aussi ;
- $K = \lceil Mu / C^{\text{Tank}} \rceil$: K représente le nombre minimum de fois qu'il faudra faire le plein de la citerne de la micro-usine pour pouvoir satisfaire toutes les recharges. Si la valeur de K augmente alors la valeur du coût de production augmente aussi ;
- $R = (\sum_i R_i) / N$: R est la quantité moyenne de carburant qu'on peut produire par période. Si la valeur de R augmente alors la valeur du coût de production diminue car on aura besoin de beaucoup moins de période pour produire le carburant dont on a besoin ;
- $R^0 = (\sum_i |R_i - R|) / N$: R^0 est l'écart moyen des rendements, plus cette valeur est proche de zéro, plus les R_i sont proches de R . Donc si la valeur de R augmente et que R^0 diminue alors la valeur du coût de production diminue car on aura besoin de beaucoup moins de période pour produire le carburant dont on a besoin ;

☐ Indicateurs libres :

- $C = (\sum_i R_i) / Mu$: C est la difficulté qu'on a à produire le carburant dont le véhicule aura besoin lors de ces recharges. Plus la valeur C est élevée, plus on pourra facilement produire du carburant ;
- $C(q) = (\sum_{i \leq M_q} R_i) / (\mu_1 + \dots + \mu_q - H_0)$: $C(q)$ est la difficulté qu'on a à produire le carburant dont le véhicule aura besoin à la recharge q . Plus les valeurs $C(q)$ sont élevées, plus on pourra facilement produire du carburant pour satisfaire les recharges 1 à q ;
- $C^* = \text{Inf}_q C(q)$: plus la valeur de C^* augmente, plus on pourra facilement produire du carburant dont aura besoin le véhicule. Donc plus cette valeur est grande, plus le numéro de période de la dernière recharge diminue ;

☐ Indicateurs de temps :

idem
pourquoi ?

- I^0 est la plus petite valeur de l'indice i_0 tel que $(\sum_{i \leq i_0} R_i) \geq Mu$; si on décidait de produire tout le carburant à recharger Mu durant des périodes consécutives en commençant à la première période, I^0 est le nombre de périodes qui seraient nécessaire pour produire tout ce que le véhicule va recharger. $[I^0/N] \in]0, 1]$: plus cette valeur se rapproche de 1, plus on aura besoin de périodes consécutives pour satisfaire les recharges, ce qui repoussera le numéro de période de la dernière recharge.
- $R^* = (\sum_i i \times R_i) / (\sum_i R_i)$: R^* est la moyenne des rendements pondérés par les numéros de périodes. Plus cette valeur est grande, plus les meilleures rendements de production (les plus élevés) se trouvent vers la fin de l'espace temps (proche de la période N);
- $P^* = (\sum_i i \times P_i) / (\sum_i P_i)$: P^* est la moyenne des coûts unitaires pondérés par les numéros de périodes. Plus cette valeur est grande, plus les plus mauvais coûts unitaires de production (les plus élevés) se trouvent vers la fin de l'espace temps (proche de la période N);
- $V^* = (\sum_i i \times Cost_i^V) / (\sum_i Cost_i^V)$: V^* est la moyenne des coûts variables pondérés par les numéros de périodes. Plus cette valeur est grande, plus les plus mauvais coûts variables de production (les plus élevés) se trouvent vers la fin de l'espace temps (proche de la période N);
- $A^* = (\sum_i i \times A_i) / (\sum_i A_i)$: A^* est la moyenne des coûts fixe pondérés par les numéros de périodes. Plus la valeur $[A^*/N]$ est grande, plus les plus mauvais coûts fixes de production (les plus élevés) se trouvent vers la fin de l'espace temps (proche de la période N);
- $G^* = (\sum_i i \times G_i) / (\sum_i G_i)$: G^* est la moyenne des écarts des coûts unitaires de production pondérés par les numéros de périodes.
- $I(q) = \text{le plus petit } i \text{ tel que } R_1 + \dots + R_i \geq \mu_1 + \dots + \mu_q - H_0$;
- $\Delta(q) = \text{Sup}(I(q), m_q) - m_q$;
- $De = \text{Sup}_q \Delta(q)$;

□ Indicateurs de coût unitaire :

- $P = (\sum_i P_i) / N$: P est la moyenne des coûts unitaires de production; plus P augmente, plus ça coûte cher de produire du carburant donc le coût de production va augmenter;
- $P^0 = (\sum_i |P_i - P|) / N$: P^0 est l'écart moyen du coût unitaire de production; plus cette valeur est proche de zéro, plus les P_i sont proches de P . Plus cette valeur diminue et si P augmente, alors ça coûtera chère de produire du carburant donc le coût de production va augmenter;
- $G = (\sum_i G_i) / N$; $G^0 = (\sum_i |G_i - G|) / N$: G est la moyenne des écarts des coûts unitaires de production; G^0 est l'écart moyen des écarts des coûts unitaires de production;

□ Indicateurs de coût absolu :

- $A = (\sum_i A_i) / N$: A est le coût fixe moyen de production de carburant par période; plus A augmente, plus ça coûte cher de démarrer la micro-usine pour produire du carburant donc le coût de production va augmenter;
- $A^0 = (\sum_i |A_i - A|) / N$: A^0 est l'écart moyen des coûts fixes de production; plus cette valeur est proche de zéro, plus les A_i sont proches de A . Plus cette valeur diminue et si A augmente, alors ça coûtera cher de démarrer la micro-usine pour produire du carburant donc le coût de production va augmenter;

Pas
clair !

- $V = (\sum_i Cost_i^V)/N$: V est le coût moyen de production variable de carburant par période ; plus V augmente, plus ça coûte cher à la micro-usine de produire du carburant donc le coût de production va augmenter ;
- $V^0 = (\sum_i |Cost_i^V - V|)/N$: V^0 est l'écart moyen, des coûts de production variables ; plus cette valeur est proche de zéro, plus les V_i sont proches de V . Plus cette valeur diminue et si V augmente, alors ça coûtera cher de produire du carburant donc le coût de production va augmenter.

Exemple 11 Reprenons l'exemple (1) avec les entrées $H_0 = 4$, $C^{Tank} = 15$, les valeurs des indicateurs sont $Mu = 25$, $K = 2$, $R = 2,666$, $R^0 = 1,2$, $C = 1,6$, $C(1) = 1,8$, $C(2) = 1$, $C^* = 1$, $I^0 = 8$, $R^* = 7,075$, $P^* = 9,396$, $V^* = 8,385$, $A^* = 8$, $G^* = 4,467$, $I(1) = 1$, $I(2) = 7$, $\Delta(1) = 0$, $\Delta(2) = 0$, $De = 0$, $P = 0,851$, $P^0 = 0,472$, $G = 0,033$, $G^0 = 0,387$, $A = 3$, $A^0 = 0$, $V = 1,733$, $V^0 = 0,684$.

Exemple 12 Reprenons l'exemple (3) avec les entrées $H_0 = 1$, $C^{Tank} = 105$, les valeurs des indicateurs sont $Mu = 96$, $K = 1$, $R = 15,367$, $R^0 = 6,682$, $C = 4,802$, $C(1) = 4,72$, $C(2) = 6,545$, $C(3) = 3,955$, $C(4) = 3,926$, $C^* = 3,926$, $I^0 = 6$, $R^* = 14,575$, $P^* = 13,169$, $V^* = 10,719$, $A^* = 15,5$, $G^* = -0,319$, $I(1) = 2$, $I(2) = 2$, $I(3) = 4$, $I(4) = 6$, $\Delta(1) = 0$, $\Delta(2) = 0$, $\Delta(3) = 0$, $\Delta(4) = 0$, $De = 1$, $P = 0,156$, $P^0 = 0,104$, $G = -0,032$, $G^0 = 0,133$, $A = 13$, $A^0 = 0$, $V = 2,133$, $V^0 = 1,457$.

→ oh. On en déduit quoi ?

7.3.2 Construction du réseau INDIC_TEMPS pour la prédiction du numéro de période de la dernière recharge

Dans cette section, on va décrire les données d'entrée, l'architecture et les données de sortie de notre réseau pour la prédiction de la date de la dernière recharge.

Description des entrées du réseau de neurones

INDIC_TEMPS

Les entrées du réseau de neurones INDIC_TEMPS sont :

- ☐ H_0/Mu : plus cette valeur est grande, plus la date de la dernière recharge diminue car on aura tendance à ne pas produire car la quantité d'hydrogène initiale de la citerne est suffisante pour assurer les recharges du véhicule ;
- ☐ P^*/N : plus cette valeur est grande, plus la date de la dernière recharge diminue car on aura tendance à vouloir produire plutôt car les meilleurs coûts de production variable s'y trouvent ;
- ☐ A^*/N : plus cette valeur est grande, plus la date de la dernière recharge diminue car les meilleurs coûts d'activation s'y trouvent ;
- ☐ C : plus cette valeur est grande, plus la date de la dernière recharge diminue ;
- ☐ C^* : plus cette valeur est grande, plus la date de la dernière recharge diminue ;
- ☐ K : si la valeur de K augmente alors la valeur de la date de la dernière recharge augmente ;
- ☐ $I^0/N \in]0, 1]$: plus cette valeur se rapproche de 1, plus on aura besoin de périodes consécutives (la date de la dernière recharge augmente) pour satisfaire les recharges, ce qui augmente la date de la dernière recharge ;
- ☐ $[m_Q, M_Q]$: c'est l'intervalle de temps durant laquelle la dernière recharge doit être réalisée.

Description de l'architecture du réseau de neurones **INDICTEMPS**

Nous voulons obtenir une approximation du numéro de période de la dernière opération de recharge en carburant T . Le calcul de T dépend de l'intervalle $[m_Q, M_Q]$. Donc nous définissons :

- $T = Ga \times (m_Q + De) + (1 - Ga) \times M_Q$ avec $Ga \in [0, 1]$;
- $Ga = \Phi^x(X)$, où Φ^x est une fonction sigmoïde $Exp(x.X)/(Exp(x.X) + 1)$, $x \geq 0$;
- $X = w_0.(H_0/Mu) + w_1.(P^*/N) + w_2.(A^*/N) + w_3.C + w_4.C^* - w_5.K - w_6.I^0/N - w_7$, où $w_0, w_1, \dots, w_6 \geq 0$ et w_7 est non signé.

A partir de ces équations, nous construisons l'architecture du réseau de neurones (Voir figure (7.4)). L'architecture est la suivante :

- Elle a huit couches d'entrées constituées chacune d'un neurone ;
- Elle a six couches cachées ;
- Elle a une couche de sortie d'un neurone.

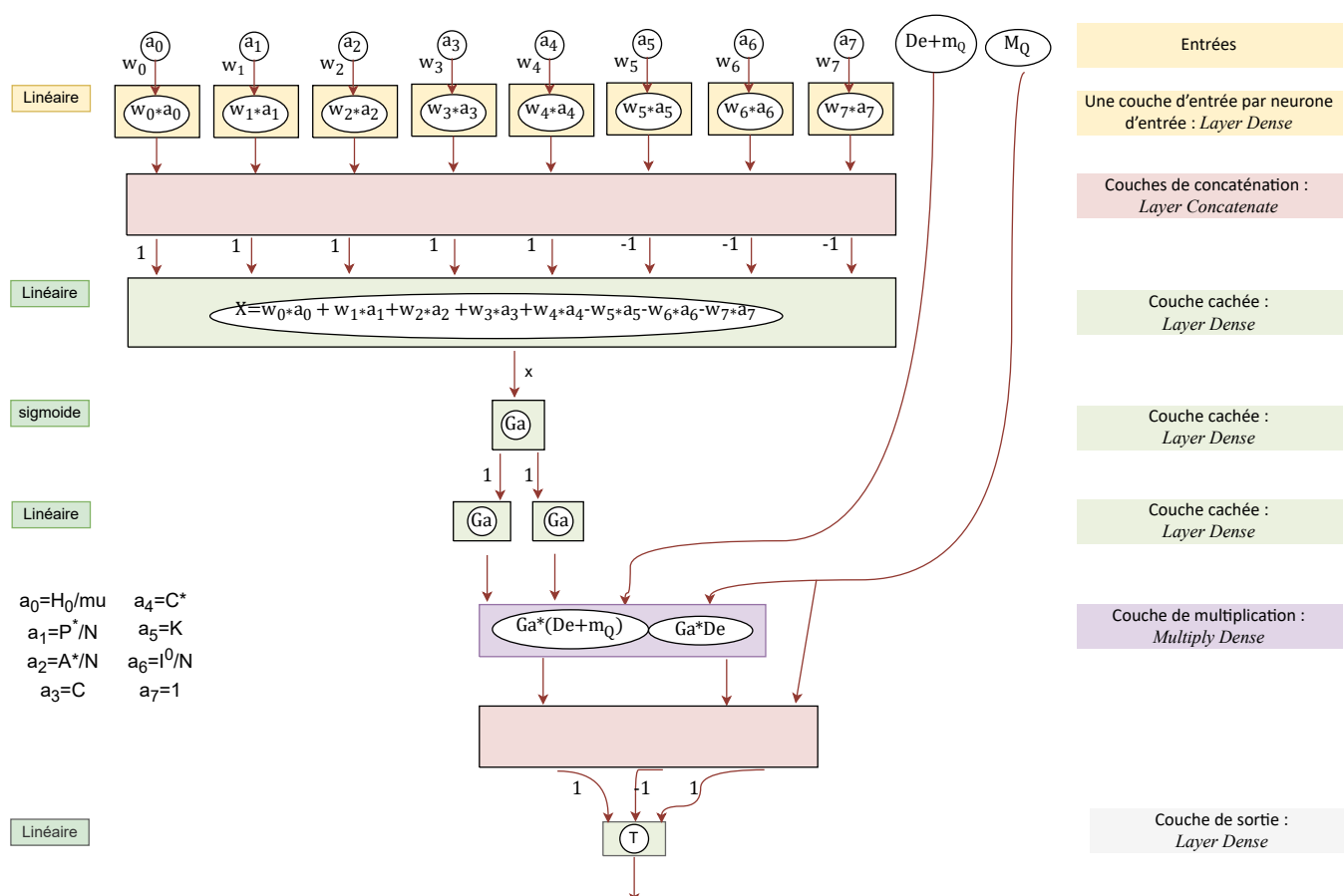


FIGURE 7.4 – Réseau **INDICTEMPS** prédisant le numéro de période de la dernière opération de recharge. Les w_i sont les coefficients synaptiques ou poids.

La fonction d'agrégation utilisée est la somme pondérée. Les fonctions d'activation utilisées sont la fonction sigmoïde et la fonction Linéaire. Le nombre de poids et de biais du réseau de neurones **INDICTEMPS** ne dépend pas de la taille des instances. L'algorithme d'optimisation utilisé est Adam.

Les données de validation sont 1/3 des données d'apprentissage. Le nombre d'itérations fixé ici est 2000. La taille des batch fixé ici est 32. La fonction de perte utilisé est le carré des erreurs.

Description des sorties du réseau de neurones *INDIC TEMPS*

La sortie du réseau de neurones **INDIC_TEMPS** est T le numéro de période de la dernière recharge.

7.3.3 Construction du réseau **INDIC_COUT** pour la prédiction du coût de production

Dans cette section, on va décrire les données d'entrée, l'architecture et les données de sortie de notre réseau pour la prédiction du coût de production.

Description des entrées du réseau de neurones *INDIC COUT*

Les entrées du réseau de neurones **INDIC_COUT** sont :

- ☐ PR/A : on multiplie la moyenne des coûts unitaires par le rendement moyen de production. Si cette valeur augmente alors le coût de production augmente aussi ;
- ☐ RP^0/A : on multiplie l'écart moyen des coûts unitaires par le rendement moyen de production. Si cette valeur augmente alors le coût de production augmente aussi ;
- ☐ RG/A : on multiplie la moyenne des écarts des coûts unitaires par le rendement moyen de production. Si cette valeur augmente alors le coût de production augmente aussi ;
- ☐ A/PR : Si cette valeur augmente alors le coût de production augmente aussi ;
- ☐ A^*/N : Plus la valeur est grande, plus les plus mauvais coûts fixes de production (les plus élevés) se trouvent vers la fin de l'espace temps (proche de la période N) ; Si cette valeur augmente alors le coût de production augmente aussi ;
- ☐ Si les valeurs $A - A^0$ et $A + A^0$ augmentent alors le coût d'activation de la micro-usine augmente aussi ;
- ☐ P^*/N : Si cette valeur augmente alors le coût de production diminue ;
- ☐ G/P : c'est la moyenne des écarts des coûts unitaires divisé par la moyenne des coûts unitaires ; si cette valeur augmente alors le coût de production augmente aussi ;
- ☐ V^0/V : c'est l'écart moyen des coûts de production variable divisé par le coût de production variable moyen ; si cette valeur augmente alors le coût de production augmente aussi ;
- ☐ C : si cette valeur augmente alors le coût de production augmente aussi ;
- ☐ Si les valeurs $P - P^0$ et $P + P^0$ augmentent alors le coût de production variable augmente aussi ;
- ☐ K : c'est le nombre minimum de fois qu'il faut faire de le plein de la citerne pour satisfaire toutes les recharges. si cette valeur augmente alors le coût de production augmente aussi ;
- ☐ Mu : c'est la quantité totale d'hydrogène à recharger. si cette valeur augmente alors le coût de production augmente aussi.

Description de l'architecture du réseau de neurones

On veut faire une approximation du coût $Cost_A$ de chaque activation de la micro-usine, le coût total d'activation est environ $Cost_A \times K$ K est le nombre minimum de fois qu'il faut faire de le plein de la citerne pour satisfaire toutes les recharges. On veut faire une approximation du coût de production $Cost_P$ par unité de production, le coût total de production sera environ $Cost_P \times Mu$ Mu est la quantité totale d'hydrogène à recharger. Nous voulons obtenir une approximation du coût de production $COST$. Donc nous définissons :

$$COST = K.Cost_A + Mu.Cost_P;$$

$$\square Cost_A = (1 + Ta) (Be(A - A^0) + (1 - Be).(A + A^0)) \text{ avec } Ta \geq 0 \text{ et } Be \in [0, 1];$$

$$— Ta = (R/A).(Ta_1.P + Ta_2.P^0 + Ta_3.G), \text{ avec } Ta_1, Ta_2, Ta_3 \geq 0;$$

$$— Be = \Phi^y(Y);$$

$$— Y = Be_1.(A/P.R) + Be_2.(A^*/N) - Be_0, \text{ avec } Be_1, Be_2 \geq 0;$$

$$\square Cost_P = Al.(p - p^0) + (1 - Al).(p + p^0) \text{ avec } Al \in [0, 1];$$

$$— Al = \Phi^z(Z);$$

$$— Z = Al_1.(P^*/N) - Al_2.(A/(P.R)) - Al_3.(G/P) - Al_4.(V^0/V) - Al_5.C - Al_0, \text{ où } Al_1, \dots, Al_5 \geq 0 \text{ et } Al_0 \text{ est non signé.}$$

pourquoi ces équations ?

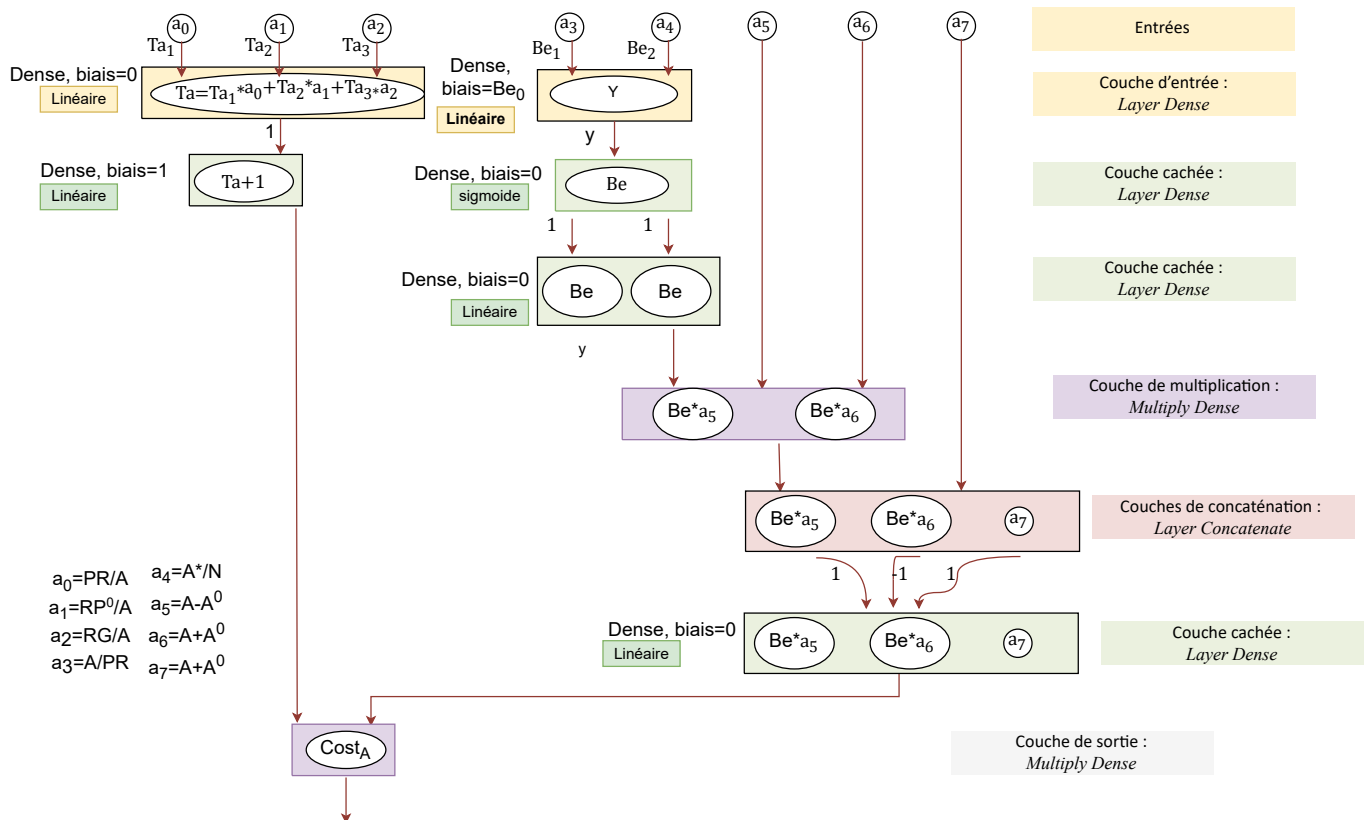
A partir de ces équations, nous construisons les réseaux de neurones des figures (7.5) et (7.6). Ces réseaux de neurones seront fusionnés (Voir figure 7.7) afin d'obtenir le réseau de neurones qui va prédire la valeur $COST$. Les sorties des réseaux de neurones des figures (7.5) et (7.6) seront les entrées d'une couche de multiplication avec les valeurs K et Mu . Les sorties de cette précédente couche seront les entrées d'un layer Dense et la sortie de ce layer Dense sera la valeur $COST$. Les poids de ce layer Dense seront tous fixés à 1. L'architecture du réseau de neurones de cette partie est la suivante :

- ☐ Elle a huit couches d'entrées constituées chacune d'un neurone;
- ☐ Elle a sept couches cachées;
- ☐ Elle a une couche de sortie d'un neurone.

La fonction d'agrégation utilisée est la somme pondérée. Les fonctions d'activation utilisées sont la fonction sigmoïde et la fonction Linéaire. Le nombre de poids et de biais du réseau de neurones **INDIC_COUT** ne dépend pas de la taille des instances. L'algorithme d'optimisation utilisé est Adam. Les données de validation sont 1/3 des données d'apprentissage. Le nombre d'itérations fixé ici est 2000. La taille des batch fixé ici est 32. La fonction de perte utilisé est le carré des erreurs.

Description des sorties du réseau de neurones

La sortie du réseau de neurones **INDIC_COUT** est la valeur du coût de production $COST$.

FIGURE 7.5 – Réseau de neurones prédisant le coût $Cost_A$.

7.4 Expérimentations numériques

Dans cette section dédiée aux expérimentations numériques, on va d'abord présenter les objectifs et le contexte technique de nos expériences. Ensuite, on présente les instances utilisées pour entraîner et tester nos réseaux. On finit cette section en présentant les résultats des tests effectués sur les réseaux présentés ci-dessus.

7.4.1 Objectifs et contexte technique

Ce que nous voulons ici, c'est obtenir une évaluation des performances des réseaux de neurones des figures (7.1), (7.2), (7.3), (7.4) et (7.7). Cela consiste à comparer les valeurs prédites par les réseaux aux valeurs optimales.

Les réseaux de neurones ont été implémentés en Python, sur un ordinateur fonctionnant sous le système d'exploitation Windows 10 avec un processeur IntelCore i5-6500@3.20 GHz, 16 Go de RAM et l'IDE utilisé est Visual Studio 2019. On a utilisé Tensorflow/Keras pour implémenter et manipuler nos réseaux de neurones.

Les graphiques de gaps montrent la variation de l'erreur des données d'apprentissage et des données de test lorsqu'on fait varier le nombre d'itérations lors de la phase d'entraînement du réseau. La courbe rouge représente l'évolution de l'erreur calculée pour le groupe de test et la courbe bleue représente l'erreur calculée pour le groupe apprentissage.

→ avant vous parlez de validation -

en fonction du carré dans lequel les coordonnées des instances sont sélectionnées aléatoirement : on a les instances dont les coordonnées de stations sont sélectionnées dans un carré 10×10 , celles sélectionnées dans un carré 20×20 et celles sélectionnées dans un carré 30×30 .

3. Concernant les coûts fixes et les coûts variables, on a décidé de générer ces coûts aléatoirement dans certains intervalles. Les instances ici peuvent être subdivisées en 5 catégories :

- On a les instances dont le coût fixe est 5 et les coûts variables appartiennent à l'intervalle $[1, 10]$. Pour cette catégorie d'instances même si le nombre de stations augmente, le coût de production aura du mal à augmenter car les coûts de production sont très petits ;
- On a les instances dont le coût fixe est 50 et les coûts variables appartiennent à l'intervalle $[20, 40]$;
- On a les instances dont le coût fixe est 100 et les coûts variables appartiennent à l'intervalle $[20, 40]$;
- On a les instances dont le coût fixe est 100 et les coûts variables appartiennent à l'intervalle $[50, 100]$;
- On a les instances dont le coût fixe est 150 et les coûts variables appartiennent à l'intervalle $[50, 100]$.

En croisant les critères 1, 2 et 3 ci-dessus, on obtient $4 \times 3 \times 5 = 60$ possibilités et pour chacune on génère 100 instances. Ce qui nous permet d'obtenir nos 6000 instances.

A partir de chaque instance, on calcule les valeurs des indicateurs qui serviront à avoir les valeurs utilisées comme entrées des réseaux de neurones. On récupère après exécution de l'algorithme Pipeline VD_PM du chapitre précédent pour chaque instance le numéro de période de la dernière opération de recharge et le coût de production. Ces données sont utilisées pour entraîner et tester nos réseaux de neurones. Les 6000 instances seront séparées en deux catégories d'instance à savoir les instances d'apprentissage qui nous serviront à entraîner les réseaux et les instances de test qui nous serviront à analyser le comportement de nos réseaux sur de nouvelles données. On décide d'avoir 90 instances de test et 5910 instances d'apprentissage. Pour avoir une idée des caractéristiques des instances utilisées, on fait une analyse statistique de nos instances. La section suivante est consacrée à présenter cette analyse.

Analyse statistique des instances

Les figures (7.10), (7.11), (7.12), (7.9) et (7.13) illustrent l'analyse statistique des instances. Par souci de lisibilité de nos figures, on affiche pas les instances une par une mais on affiche la moyenne de chaque paquet de 100 instances. On ordonne préalablement nos instances par coût de production croissant. Les instances générées avec le coût fixe 5 et les coûts variables appartenant à l'intervalle $[1, 10]$ ont des coûts tellement petits que le nombre de stations n'impacte pas trop le coût d'où les pics observés sur les figures (7.10) (b), (7.11) (b), (7.12) (b)

La figure (7.8) représente le coût moyen de production regroupé par groupes de 100 instances et préalablement ordonné par ordre croissant. On constate que les coûts de production de nos instances sont compris entre 0 et 2000.

La figure 7.9(a) représente le coût fixe moyen de production regroupé par paquet de 100 instances. La figure 7.9(a) représente le coût variable moyen de production regroupé par paquet de 100 instances

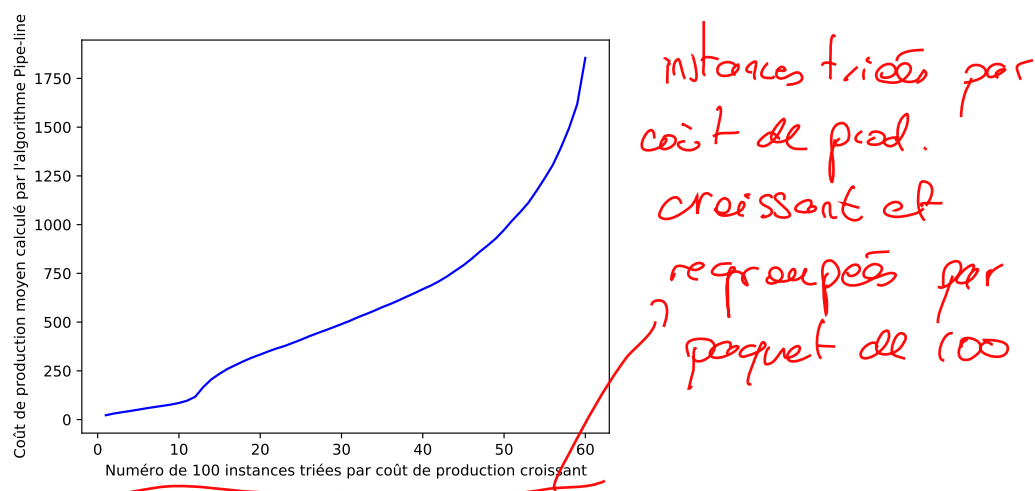


FIGURE 7.8 – Coût moyen de production croissant regroupé par paquet de 100 instances. Ce coût a été obtenu en exécutant l'algorithme Pipe-line VD_PM du chapitre précédent.

préalablement classées par coût de production croissant. De ces figures, on peut dire que le coût de production croît lorsque le coût fixe et le coût variable croient aussi. Ce qui montre que ces coûts peuvent être utilisés comme entrées des réseaux qui prédissent le coût de production.

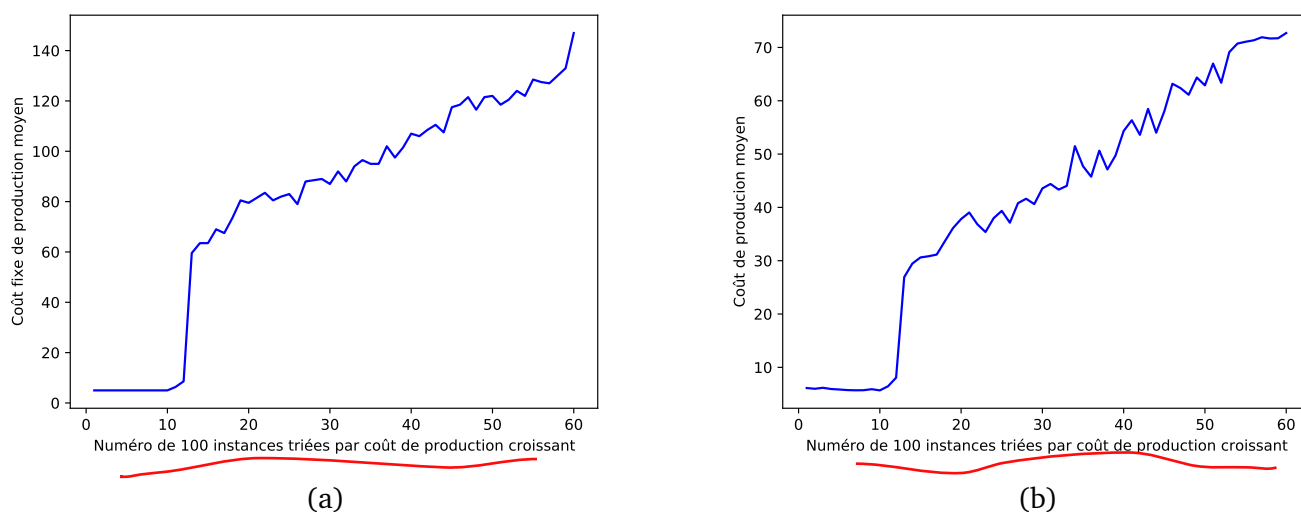


FIGURE 7.9 – La figure (a) représente le coût fixe moyen de production regroupé par paquet de 100 instances. La figure (b) représente le coût variable moyen de production regroupé par paquet de 100 instances. Sachant que les instances ont été classées par coût de production croissant.

La figure (7.10)(a) représente le nombre d'instance 9 par nombre de stations. On constate qu'on a 1500 instances de 8 stations, 1500 instances de 10 stations, 1500 instances de 15 stations et 1500 instances de 20 stations. La figure (7.10)(b) représente la moyenne du nombre de stations regroupé par paquet de 100 instances. On constate que du paquet d'instances 1 au paquet d'instances 12, le nombre de stations croît au fur et à mesure que le coût de production croît. Mais à partir du paquet d'instances 13 on a une brusque chute du nombre de stations moyen malgré le fait que le coût augmente. Ceci est dû au fait qu'à la génération des instances, on a créé des instances avec un coût fixe très petit 5, alors que les autres coûts sont au moins quatre fois plus élevés. On peut conclure que

le nombre de stations n'est pas une valeur pertinente à utiliser pour prédire le coût de production.

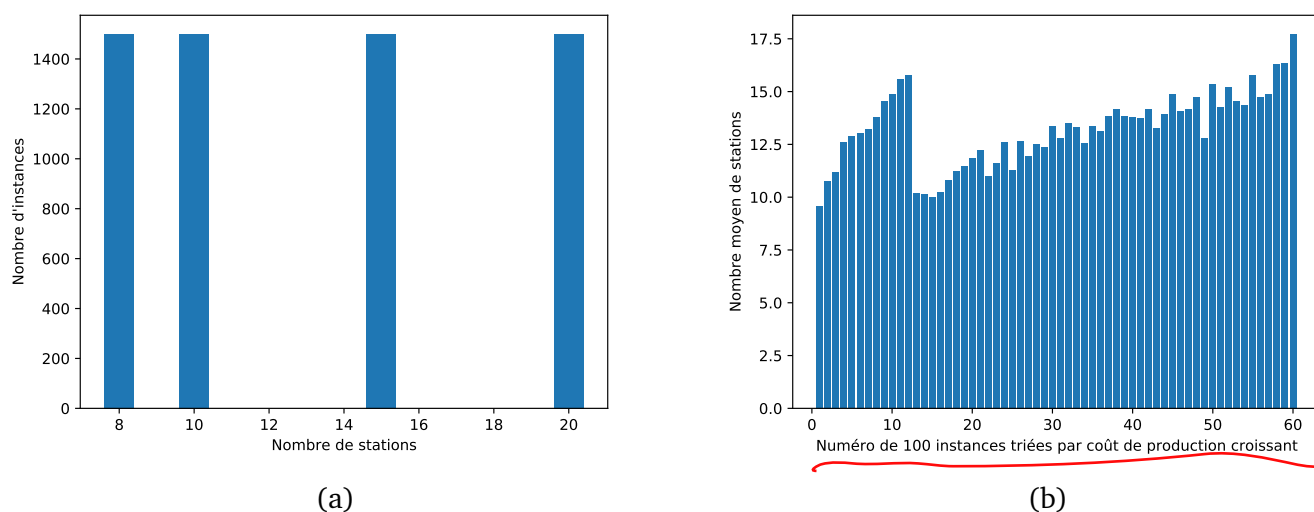


FIGURE 7.10 – La figure (a) représente le nombre d'instance par nombre de station. La figure (b) représente la moyenne du nombre de stations regroupé par paquet de 100 instances. Sachant que les instances ont été classées par coût de production croissant.

On a exécuté l'algorithme DPS_VD du chapitre précédent pour obtenir la stratégie de recharge réduite. La figure (7.11)(a) représente le nombre d'instance par nombre de recharge. On constate que la majorité des instances ont entre 4 et 6 recharge. La figure (7.11)(b) représente la moyenne du nombre de recharge regroupé par paquet de 100 instances préalablement classées par coût de production croissant. On constate que du paquet d'instances 1 au paquet d'instances 12, le nombre moyen de recharge croît au fur et à mesure que le coût de production croît. Mais à partir du paquet d'instances 13 on a une brusque chute du nombre moyen de recharge malgré le fait que le coût augmente. Ceci est dû au fait qu'à la génération des instances, on a créé des instances avec un coût fixe très petit 5, alors que les autres coûts sont au moins quatre fois plus élevés. On peut conclure que le nombre moyen de recharge n'est pas une valeur pertinente à utiliser pour prédire le coût de production.

La figure (7.12) (a) représente la durée moyenne d'une période de production regroupé par paquet de 100 instances ordonnées par coût de production croissant. La figure (7.12) (b) représente l'évolution du temps maximal moyen (TM_{ax}) regroupé par paquet de 100 instances préalablement classées par coût de production croissant. On peut conclure que la durée moyenne d'une période et le temps maximal moyen ne sont pas des valeurs pertinentes à utiliser pour prédire le coût de production.

La figure (7.13)(a) représente la capacité moyenne de la citerne regroupé par paquet de 100 instances. La figure (7.13)(a) représente la capacité moyenne du réservoir regroupé par paquet de 100 instances. Sachant que les instances ont été classées par coût de production croissant. On peut conclure que la capacité moyenne de la citerne et la capacité moyenne du réservoir ne sont pas des valeurs pertinentes à utiliser pour prédire le coût de production.

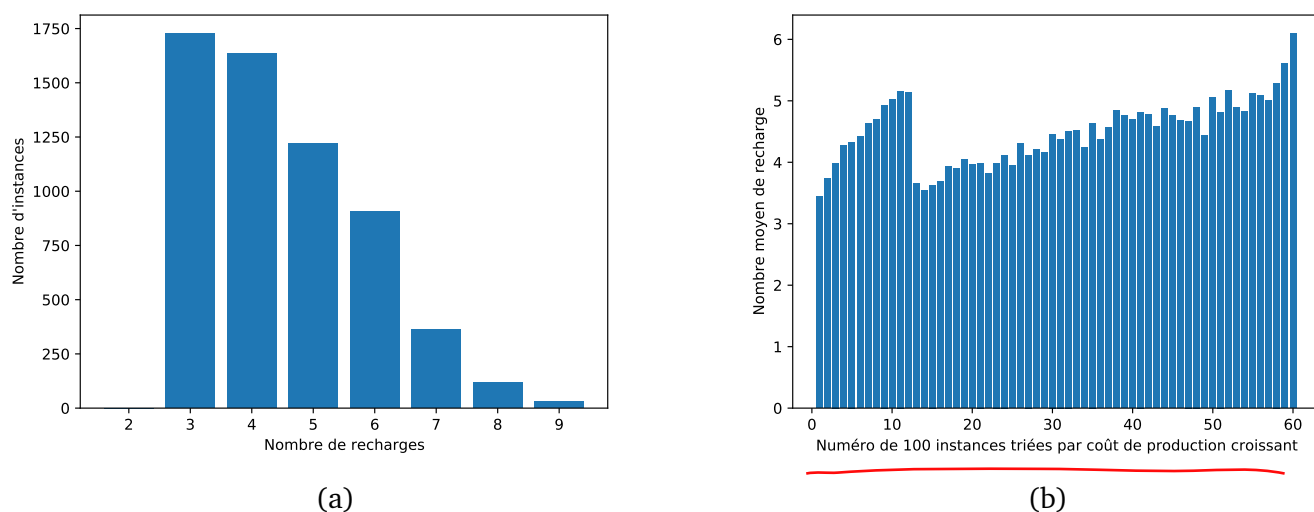


FIGURE 7.11 – La figure (a) représente le nombre d'instance par nombre de recharge. La figure (b) représente la moyenne du nombre de recharge regroupé par paquet de 100 instances. Sachant que les instances ont été classées par coût de production croissant.

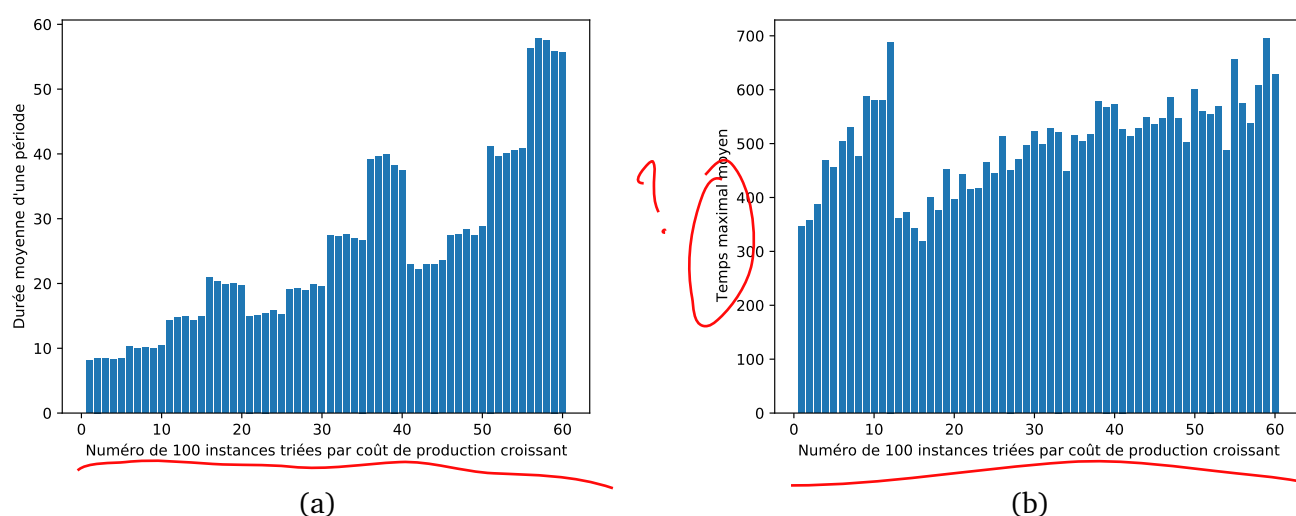


FIGURE 7.12 – La figure (a) représente la durée moyenne d'une période de production regroupé par paquet de 100 instances. La figure (b) représente l'évolution du temps maximal moyen regroupé par paquet de 100 instances. Sachant que les instances ont été classées par coût de production croissant.

7.4.3 Moyenne de la valeur absolue des gaps

Le but de cette section est de présenter les gaps obtenus après entraînement de nos réseaux de neurones. La métrique d'évaluation de nos réseaux est le gap entre la valeur V_{Opt} utilisée pour entraîner nos réseaux et la valeur V_{Predit} prédit par nos réseaux : $(|V_{Opt} - V_{Predit}|/V_{Opt})$. On présente ici la moyenne des gaps des données d'apprentissage et des données de test. ~~Vous ne trouverez~~ au tableau (7.2) les résultats de nos réseaux qui prédisent le coût de production. ~~Vous trouverez~~ au tableau (7.3) les résultats de nos réseaux qui prédisent le numéro de période de la dernière recharge. La signification de chaque colonne est :

- la colonne nommée « Nom du réseau de neurones » est le nom du réseau de neurones ;

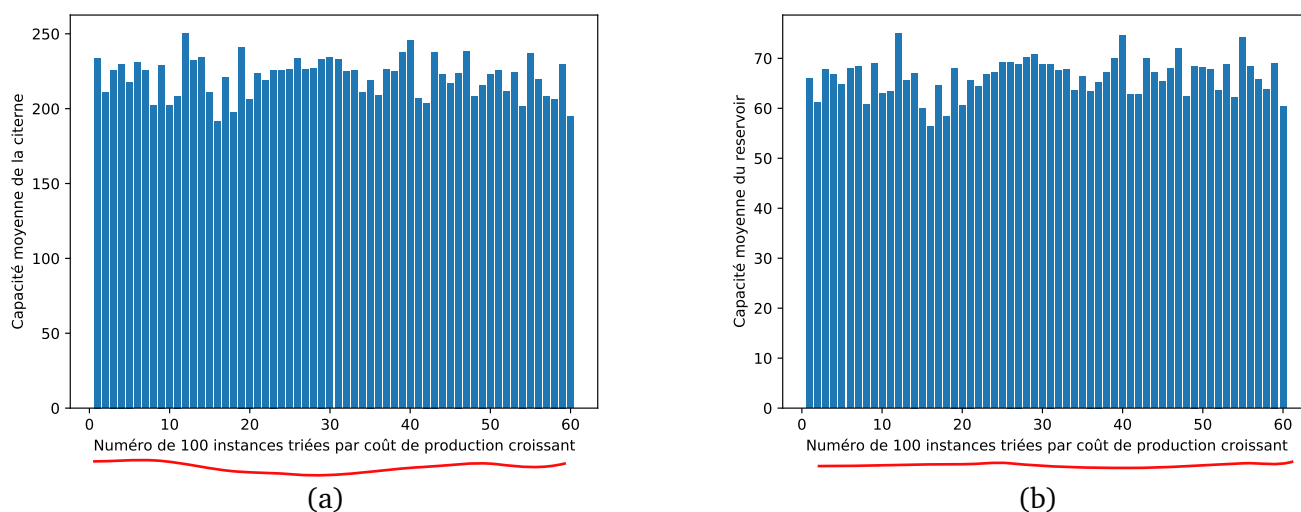


FIGURE 7.13 – La figure (a) représente la capacité moyenne de la citerne regroupé par paquet de 100 instances. La figure (a) représente la capacité moyenne du réservoir regroupé par paquet de 100 instances. Sachant que les instances ont été classées par coût de production croissant.

- ☐ la colonne nommée « fig. réseau » ^{donne} est les numéros des figures qui illustrent l'architecture du réseau de neurones ;
- ☐ la colonne nommée « fig. résultats » ^{donne} est le numéro de figures qui illustrent la courbe des valeurs prédites par le réseau de neurones, la courbe des valeurs utilisées pour entraîner le réseau de neurones et les courbes de l'évolution du gap au fil des itérations ;
- ☐ la colonne nommée « # itérations » est le nombre d'itérations ;
- ☐ la colonne nommée « Gap test » est la moyenne en pourcentage des gaps des données de test et d'apprentissage ;
- ☐ la colonne nommée « Gap entraînement » est la moyenne en pourcentage des gaps des données d'entraînement ;
- ☐ la colonne nommée « # poids » est le nombre de poids synaptiques des réseaux de neurones.

Nom du réseau de neurones	fig. réseau	fig. résultats	# itérations	Gap test(%)	Gap entraînement(%)	# poids
Réseau SIMPLE_TYPE	(7.1)	(7.14) (7.15)	200	31,61	22,32	697
Réseau SIMPLE_PERIODE	(7.2)	(7.16) (7.17)	200	30,57	22,22	729
Réseau MIXTE_COUT	(7.3)	(7.18) (7.19)	200	16,06	12,12	2500
Réseau INDIC_COUT	(7.5)(7.6) (7.7)	(7.20) (7.21)	200	14,87	16,87	14

TABLE 7.2 – Apprentissage du coût de production : Moyenne de la valeur absolue des gaps des données de test et d'apprentissage.

Notre objectif en terme de gap lorsqu'on construisait nos réseaux de neurones était d'être en dessous de 5%, ce qui est la plupart du temps le cas d'une bonne heuristique en recherche opérationnelle. Autrement dit, on espérait ne pas être très loin des résultats d'une heuristique mais ce n'est malheureusement pas le cas. Les gaps des réseaux **SIMPLE_TYPE** et **SIMPLE_PERIODE** sont les plus élevés,

Nom du réseau de neurones	fig. réseau	fig. résultats	# itérations	Gap test (%)	Gap entraînement(%)	# poids
Réseau MIXTE_TEMPS	(7.3)	(7.22) (7.23)	200	28,54	7,69	2500
Réseau INDIC_TEMPS	(7.4)	(7.24) (7.25)	200	14,23	8,31	9

TABLE 7.3 – Apprentissage du numéro de période de la dernière recharge : Moyenne de la valeur absolue des gaps des données de test et d'apprentissage.

peut-être es-ce à cause de la simplicité de ces réseaux. On est assez déçu des résultats du réseau **MIXTE_COUT** et **MIXTE_TEMPS** car il est le réseau qui a le plus grand nombre de poids, on s'attendait à un meilleur gap. Les gaps du réseau **INDIC_COUT** ne sont pas très décevants au vue du nombre de poids de ce réseau. Cela est peut-être dû au fait qu'on a peu d'indicateurs. On constate ici que le gap du réseau **INDIC_COUT** des données d'entraînement est supérieur à celui des données de test peut-être es-ce dû au fait qu'on a juste 90 instances de test contre 5910 instances d'apprentissage. On constate que le réseau **MIXTE_TEMPS** donne un grand écart entre les données d'apprentissage et les données de test.

La capacité de généralisation d'un réseau est sa capacité à faire de bonnes prédictions sur de nouvelles données. On constate ici que les réseaux qui ont la plus bonne capacité de généralisation sont les réseaux **INDIC_COUT** et **INDIC_TEMPS**. Tandis que les réseaux qui ont un gap entraînement le plus faible sont les réseaux **MIXTE_COUT** et **MIXTE_TEMPS**.

7.4.4 Résultats des réseaux SIMPLE_TYPE et SIMPLE_PERIODE

La fonction d'activation permet de changer la façon dont une valeur est représentée. Il existe plusieurs types de fonction d'activation parmi lesquelles on peut citer :

- ☐ La fonction linéaire : $f(x) = x$, elle laisse passer toutes les valeurs dans les couches suivantes.
- ☐ La fonction relu (Rectified Linear Unit) :

$$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ x & \text{si } x \geq 0. \end{cases}$$

Elle ne laisse passer dans les couches suivantes que les valeurs positives.

- ☐ La fonction elu (Exponential Linear Unit) :

$$f(x) = \begin{cases} \alpha \times (e^x - 1) & \text{si } x < 0 \\ x & \text{si } x \geq 0. \end{cases}$$

Elle est une amélioration de la fonction RELU car elle lisse aussi les valeurs négatives

- ☐ La fonction selu (Scaled Exponential Linear Unit) est une amélioration de la fonction elu.

$$f(x) = \begin{cases} scale \times \alpha \times (e^x - 1) & \text{si } x < 0 \\ scale \times x & \text{si } x \geq 0. \end{cases}$$

Les valeurs sont fixes $\alpha = 1,67326324$, $scale = 1,05070098$ et on ne peut pas les changer.

- La fonction softplus : $f(x) = \log(e^x + 1)$, est une approximation de la fonction relu.
- La fonction sigmoïde : $f(x) = \frac{1}{1 + e^{-x}}$ elle donne une valeur dans l'intervalle $[0,1]$.
- La fonction tangente hyperbolique $f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$ elle donne une valeur dans l'intervalle $[-1,1]$.

Le tableau (7.4) présentent les gaps obtenus avec le réseau SIMPLE lorsqu'on modifie les fonctions d'activation de la couche d'entrées et de la couche cachée. On constate que c'est la fonction selu qui nous donne les meilleures gaps.

Activation couche 1	Activation couche 2	Gap test (%)	Gap entraînement(%)
linéaire	linéaire	206,6	91,32
tangente h.	tangente h.	76,09	69,57
sigmoïde	sigmoïde	54,21	68,14
relu	relu	30,31	34,81
relu	selu	36,24	34,67
selu	relu	31,51	22,96
softplus	softplus	38,66	21,79
elu	elu	23,80	20,79
selu	selu	22,58	20,66

TABLE 7.4 – Comparaison des fonctions d'activation du réseau **SIMPLE**.

Résultats du réseau SIMPLE_TYPE

La figure (7.14) représente l'évolution du gap moyen des données d'apprentissage (courbe bleue) et des données de test (courbe rouge) lorsqu'on augmente le nombre d'itérations. Globalement, ce gap décroît. On constate que le gap ici se stabilise vite. Au bout de 25 itérations, on a déjà un un gap assez stable.

Les résultats du réseau **SIMPLE_TYPE** sont illustrés par les figures (7.15). On veut voir la corrélation entre les valeurs prédites par le réseau et les valeurs optimales. L'axe des abscisses est respectivement pour la figure (7.15) (a) et (7.15) (b) le numéro des groupes de 100 instances et le numéro d'une instance. L'axe des ordonnées est respectivement pour la figure (7.15) (a) et (7.15) (b) le coût de production moyen et le coût de production. La courbe verte est la courbe des valeurs optimales et la courbe rouge est la courbe des valeurs prédites par le réseau de la figure (7.1). **Le fait de faire des moyennes a lissé les courbes de la figure (7.15) (a) sinon ces dernières se présentent aussi en dents de scie.** On observe que la courbe des valeurs prédites a l'allure de la courbe des valeurs optimales avec une tendance à être légère en dessous de celle-ci.

On constate que le réseau **SIMPLE_TYPE** est de mauvaise qualité, ceci peut s'expliquer par le fait que ce réseau est ^ssimple, ^{!"}c'est-à-dire qu'il n'épouse pas la logique de notre problème.

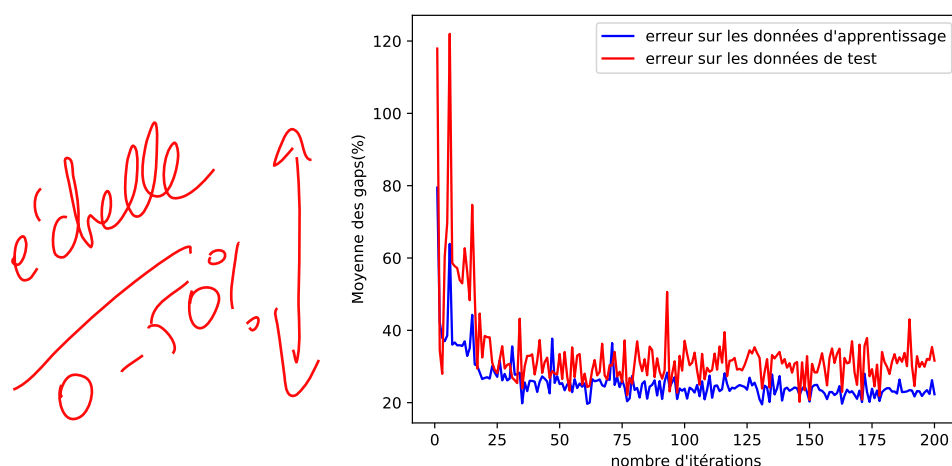
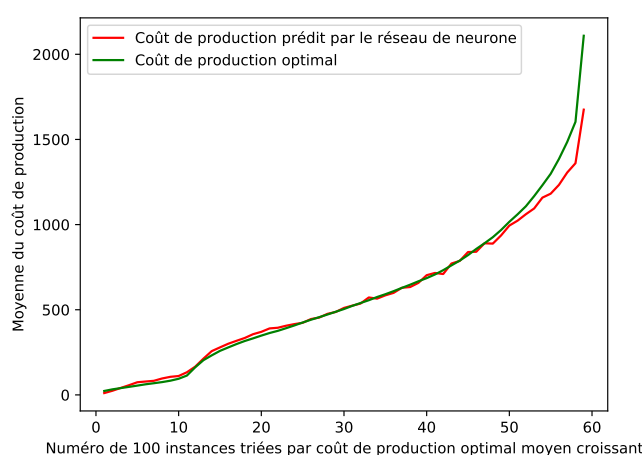
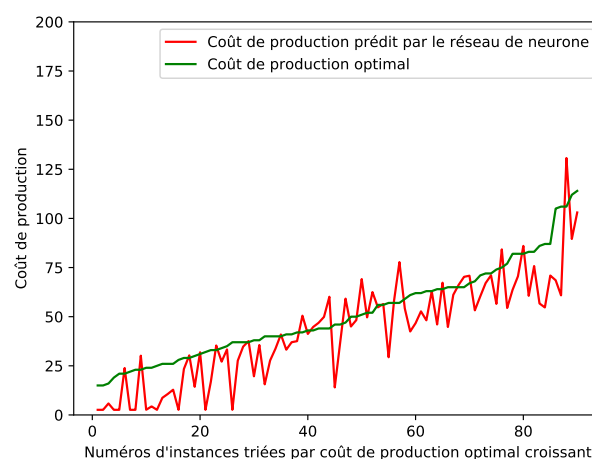


FIGURE 7.14 – Cette figure représente l'évolution du gap lorsqu'on fait varier le nombre d'itérations lors de l'apprentissage du réseau **SIMPLE_TYPE**. Ici, les données d'entrées sont classées par type. La courbe bleue est l'évolution de l'erreur sur les données d'apprentissage et la courbe rouge est l'évolution de l'erreur sur les données de test.



(a)



(b)

FIGURE 7.15 – Résultats du schéma d'apprentissage **SIMPLE_TYPE** pour la prédiction du coût de production avec des données classées type par type. (a) représente les données d'apprentissage et (b) représente les données de test. La courbe rouge est la courbe des valeurs prédites et la courbe verte est la courbe des valeurs optimales.

Résultats du réseau **SIMPLE_PERIODE**

La figure (7.16) représente l'évolution du gap moyen des données d'apprentissage (courbe bleue) et des données de test (courbe rouge) lorsqu'on augmente le nombre d'itérations. Globalement, ce gap décroît. On constate que le gap ici se stabilise vite. Au bout de 25 itérations, on a déjà un un gap assez stable.

Les résultats du réseau **SIMPLE_PERIODE** sont illustrés par les figures (7.17). On veut voir la corrélation entre les valeurs prédites par le réseau et les valeurs optimales. L'axe des abscisses est respectivement pour la figure (7.17) (a) et (7.17) (b) le numéro des groupes de 100 instances et le

numéro d'une instance. L'axe des ordonnées est respectivement pour la figure (7.17) (a) et (7.17) (b) le coût de production moyen et le coût de production. La courbe verte est la courbe des valeurs optimales et la courbe rouge est la courbe des valeurs prédites par le réseau de la figure (7.2). Le fait de faire des moyennes a lissé les courbes de la figure (7.17) (a) sinon ces dernières se présentent aussi en dents de scie. On observe que la courbe des valeurs prédites a l'allure de la courbe des valeurs optimales avec une tendance à être légère au dessus de celle-ci.

On constate que le réseau **SIMPLE_PERIODE** est de mauvaise qualité, ceci s'explique par le fait que ce réseau est simple, c'est-à-dire qu'il n'épouse pas la logique de notre problème.

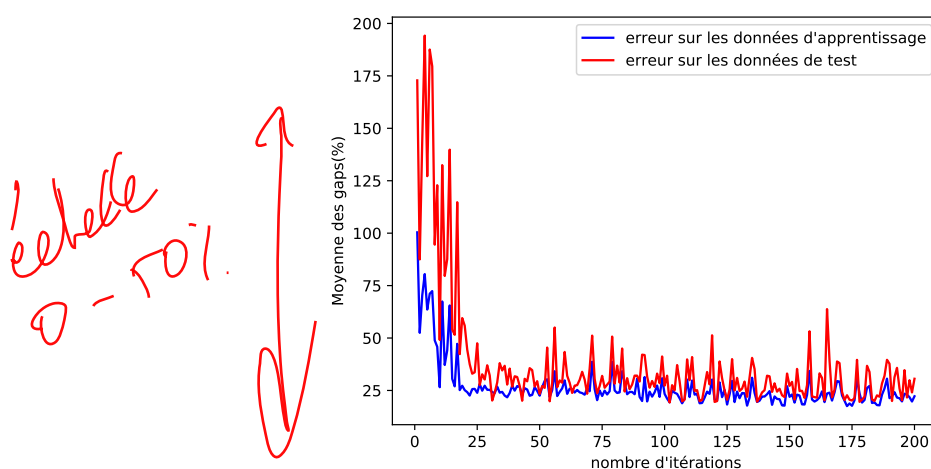
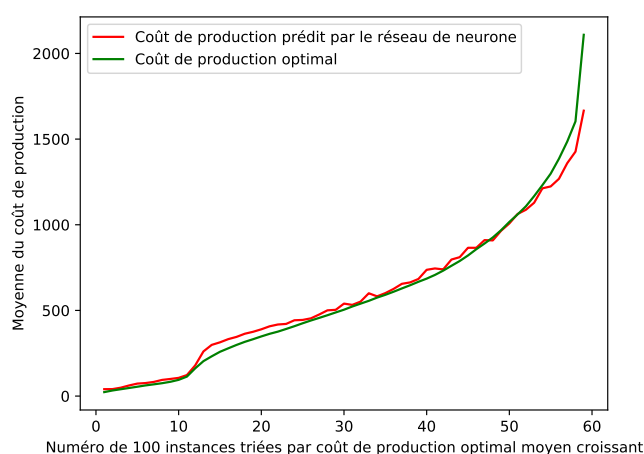
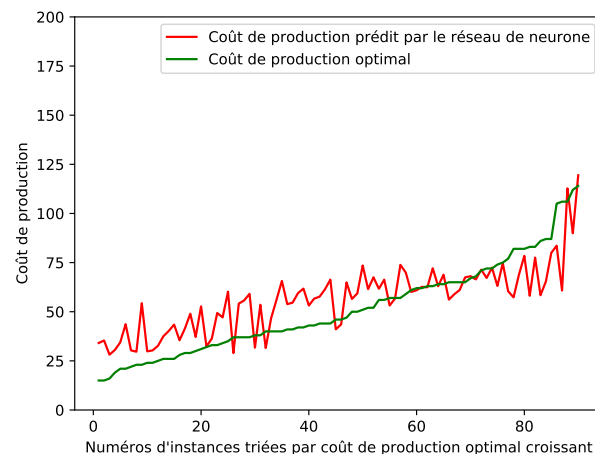


FIGURE 7.16 – Cette figure représente l'évolution du gap lorsqu'on fait varier le nombre d'itérations lors de l'apprentissage du réseau **SIMPLE_PERIODE**. Ici, les données d'entrées sont classées par période. La courbe bleue est l'évolution de l'erreur sur les données d'apprentissage et la courbe rouge est l'évolution de l'erreur sur les données de test.



(a)



(b)

FIGURE 7.17 – Résultats du réseau **SIMPLE_PERIODE** pour la prédiction du coût de production avec des données classées par période. (a) représente les données d'apprentissage et (b) représente les données de test. La courbe rouge est la courbe des valeurs prédites et la courbe verte est la courbe des valeurs optimales.

7.4.5 Résultats du réseau MIXTE_COUT pour la prédiction du coût de production

La figure (7.18) représente l'évolution du gap moyen des données d'apprentissage (courbe bleue) et des données de test (courbe rouge) lorsqu'on augmente le nombre d'itérations. Par souci de lisibilité, on a limité l'axe des abscisses à l'intervalle [0,40]. Globalement, ce gap décroît. On constate que le gap ici se stabilise vite. Au bout d'environ 10 itérations, on a déjà un gap assez stable.

Les résultats du réseau MIXTE_COUT pour la prédiction du coût de production sont illustrés par les figures (7.19). On veut voir la corrélation entre les valeurs prédites par le réseau et les valeurs optimales. L'axe des abscisses est respectivement pour la figure (7.19) (a) et (7.19) (b) le numéro des groupes de 100 instances et le numéro d'une instance. L'axe des ordonnées est respectivement pour la figure (7.19) (a) et (7.19) (b) le coût de production moyen et le coût de production. La courbe verte est la courbe des valeurs optimales et la courbe rouge est la courbe des valeurs prédites par le réseau de la figure (7.3). Le fait de faire des moyennes a lissé les courbes de la figure (7.19) (a) sinon ces dernières se présentent aussi en dents de scie. On observe que la courbe des valeurs prédites a l'allure de la courbe des valeurs optimales.

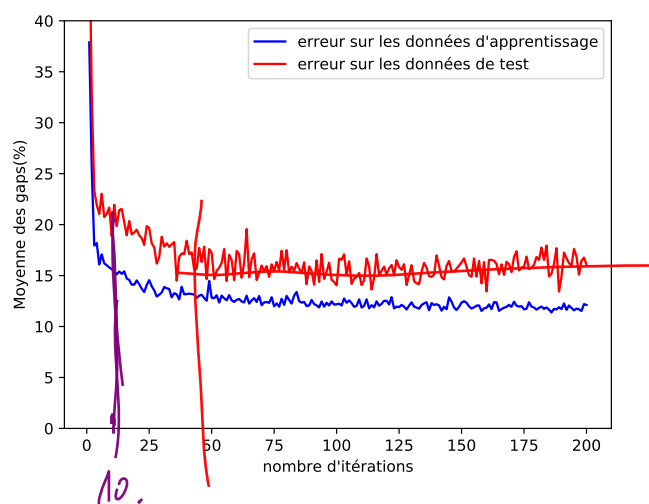


FIGURE 7.18 – Cette figure représente l'évolution du gap lorsqu'on fait varier le nombre d'itérations lors de l'apprentissage du réseau MIXTE_COUT pour la prédiction du coût de production. La courbe bleue est l'évolution de l'erreur sur les données d'apprentissage et la courbe rouge est l'évolution de l'erreur sur les données de test.

7.4.6 Résultats du réseau INDIC_COUT pour la prédiction du coût de production

La figure (7.20) représente l'évolution du gap moyen des données d'apprentissage (courbe bleue) et des données de test (courbe rouge) lorsqu'on augmente le nombre d'itérations. Par souci de lisibilité, on a limité l'axe des abscisses à l'intervalle [0,40]. Globalement, ce gap décroît. On constate que le gap ici se stabilise au bout de 85 itérations.

Les résultats du réseau INDIC_COUT pour la prédiction du coût de production sont illustrés par les figures (7.21). On veut voir la corrélation entre les valeurs prédites par le réseau et les valeurs optimales. L'axe des abscisses est respectivement pour la figure (7.21) (a) et (7.21) (b) le numéro des groupes de 100 instances et le numéro d'une instance. L'axe des ordonnées est respectivement pour la

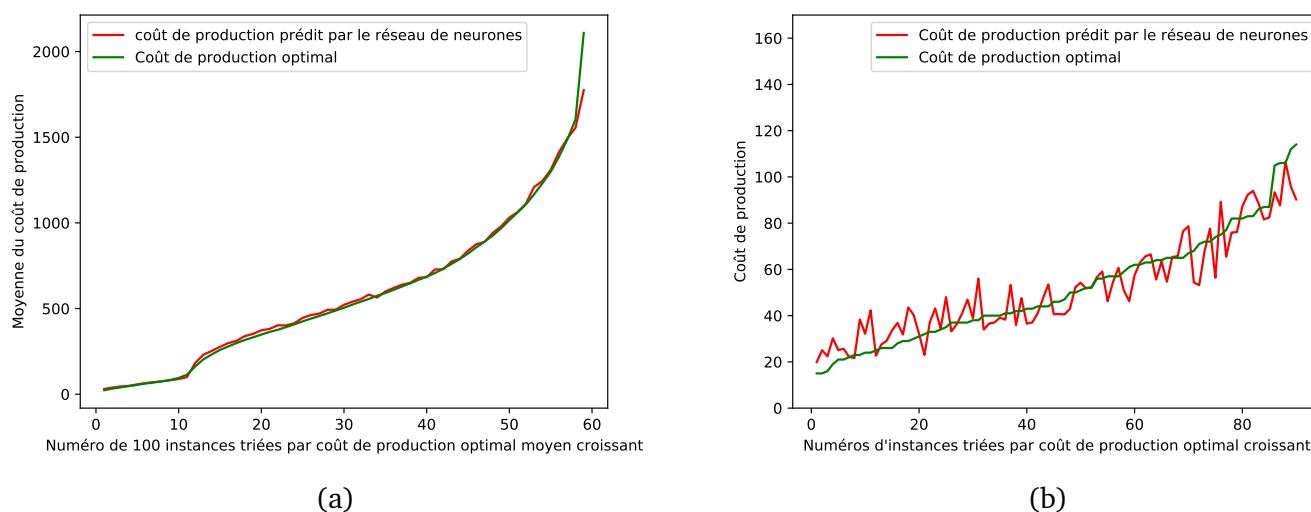


FIGURE 7.19 – Résultats du réseau **MIXTE_COUT** pour la prédiction du coût de production. (a) représente les données d'apprentissage et (b) représente les données de test. La courbe rouge est la courbe des valeurs prédites et la courbe verte est la courbe des valeurs optimales.

figure (7.21) (a) et (7.21) (b) le coût de production moyen et le coût de production. La courbe verte est la courbe des valeurs optimales et la courbe rouge est la courbe des valeurs prédites par le réseau de la figure (7.7). Le fait de faire des moyennes a lissé les courbes de la figure (7.21) (a) sinon ces dernières se présentent aussi en dents de scie. On observe que la courbe des valeurs prédites a l'allure de la courbe des valeurs optimales.

7.4.7 Résultats du réseau **MIXTE_TEMPS** pour la prédiction du numéro de période de la dernière recharge

La figure (7.22) représente l'évolution du gap moyen des données d'apprentissage (courbe bleue) et des données de test (courbe rouge) lorsqu'on augmente le nombre d'itérations. Par souci de lisibilité, on a limité l'axe des abscisses à l'intervalle [0,40]. Globalement, ce gap décroît. On constate que le gap ici se stabilise vite. Au bout d'environ 10 itérations, on a déjà un gap assez stable. Mais à l'itération 200 le gap remonte.

Les résultats du réseau **MIXTE_TEMPS** pour la prédiction du numéro de période de la dernière recharge sont illustrés par les figures (7.23). On veut voir la corrélation entre les valeurs prédites par le réseau et les valeurs optimales. L'axe des abscisses est respectivement pour la figure (7.23) (a) et (7.23) (b) le numéro des groupes de 100 instances et le numéro d'une instance. L'axe des ordonnées est respectivement pour la figure (7.23) (a) et (7.23) (b) le numéro de période moyen de la dernière opération de recharge et le numéro de période de la dernière opération de recharge. Les points verts sont les valeurs optimales et les points rouges sont les valeurs prédites par le réseau de la figure (7.3). On observe que les valeurs prédites ont l'allure des valeurs optimales.

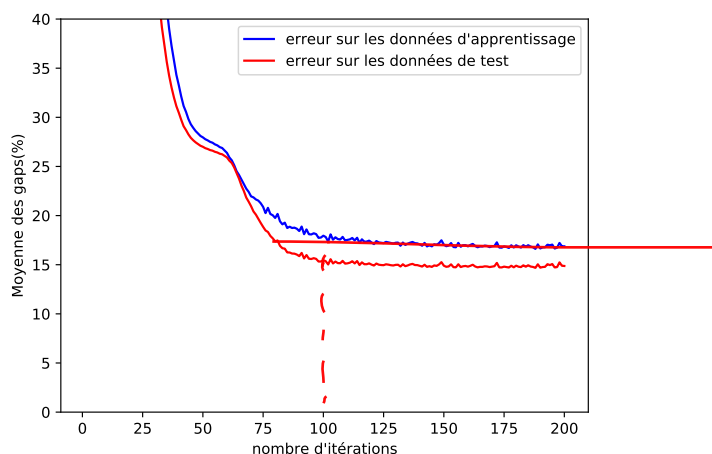
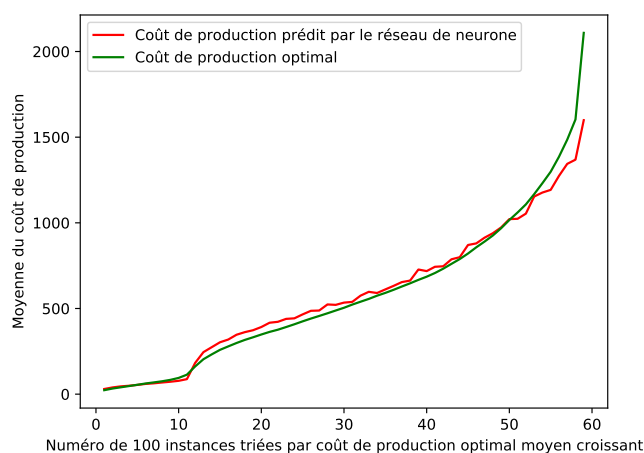
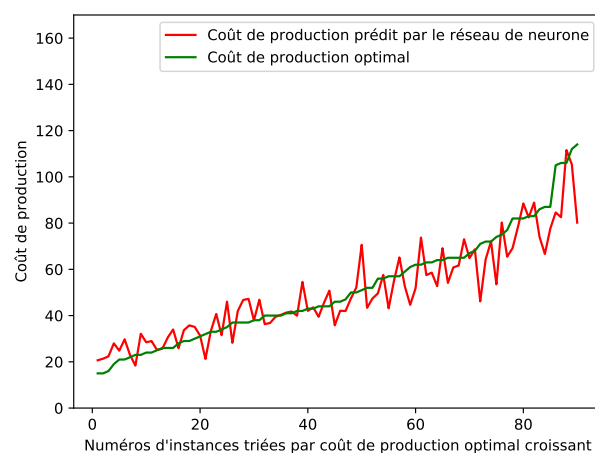


FIGURE 7.20 – Cette figure représente l'évolution du gap lorsqu'on fait varier le nombre d'itérations lors de l'apprentissage du schéma d'apprentissage **INDIC_COUT** pour la prédiction du coût de production. La courbe bleue est l'évolution de l'erreur sur les données d'apprentissage et la courbe rouge est l'évolution de l'erreur sur les données de test.



(a)



(b)

FIGURE 7.21 – Résultat du schéma d'apprentissage **INDIC_COUT** pour la prédiction du coût de production. (a) représente les données d'apprentissage et (b) représente les données de test. La courbe rouge est la courbe des valeurs prédites et la courbe verte est la courbe des valeurs optimales.

7.4.8 Résultats du réseau **INDIC_TEMPS** pour la prédiction du numéro de période de la dernière recharge

La figure (7.24) représente l'évolution du gap moyen des données d'apprentissage (courbe bleue) et des données de test (courbe rouge) lorsqu'on augmente le nombre d'itérations. Par souci de lisibilité, on a limité l'axe des abscisses à l'intervalle [0,40]. Globalement, ce gap décroît. On constate que le gap ici se stabilise vite. Au bout d'environ 10 itérations, on a déjà un un gap assez stable.

Les résultats du réseau **INDIC_TEMPS** pour la prédiction du numéro de période de la dernière recharge sont illustrés par les figures (7.25). On veut voir la corrélation entre les valeurs prédites par le réseau et les valeurs optimales. L'axe des abscisses est respectivement pour la figure (7.25) (a) et

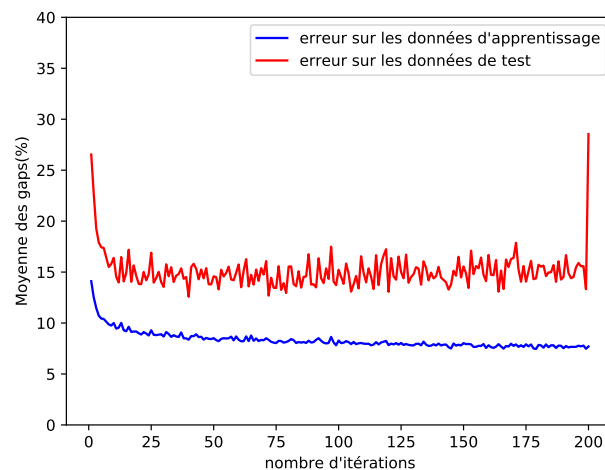


FIGURE 7.22 – Cette figure représente l'évolution du gap lorsqu'on fait varier le nombre d'itérations lors de l'apprentissage du réseau **MIXTE_TEMPS** pour la prédiction du numéro de période de la dernière recharge. La courbe bleue est l'évolution de l'erreur sur les données d'apprentissage et la courbe rouge est l'évolution de l'erreur sur les données de test.

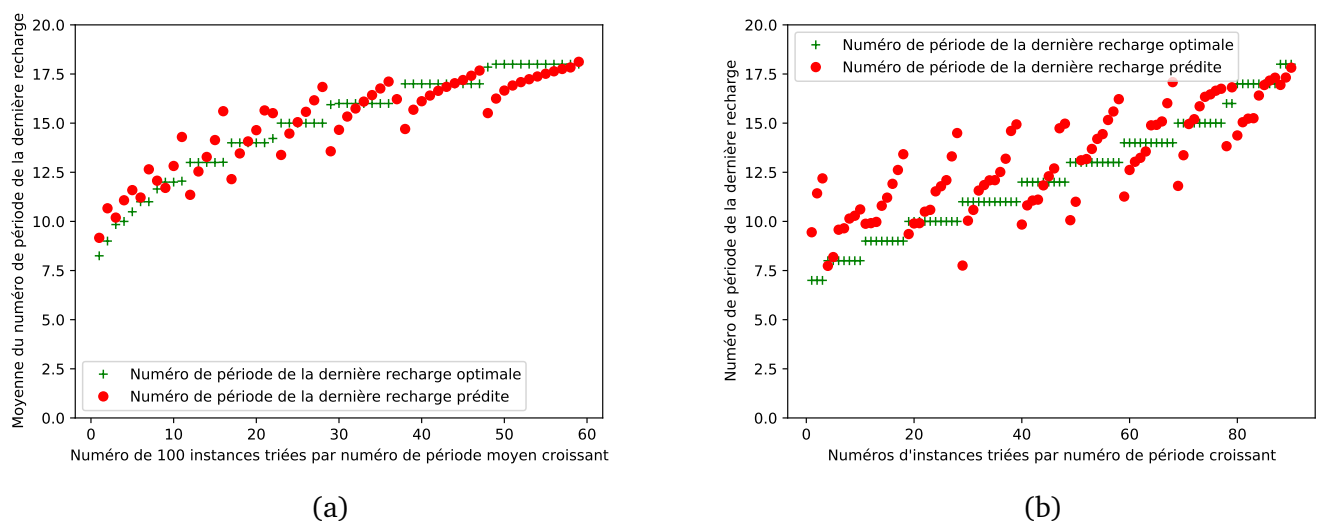


FIGURE 7.23 – Résultats du réseau **MIXTE_TEMPS** pour la prédiction du numéro de période de la dernière recharge. (a) représente les données d'apprentissage et (a) représente les données de test. Les points rouges sont les valeurs prédites et les points verts sont les valeurs optimales.

(7.25) (b) le numéro des groupes de 100 instances et le numéro d'une instance. L'axe des ordonnées est respectivement pour la figure (7.25) (a) et (7.25) (b) le numéro de période moyen de la dernière opération de recharge et le numéro de période de la dernière opération de recharge. Les points verts sont les valeurs optimales et les points rouges sont les valeurs prédites par le réseau de la figure (7.4). On observe que les valeurs prédites ont l'allure des valeurs optimales avec une tendance à être au dessus.

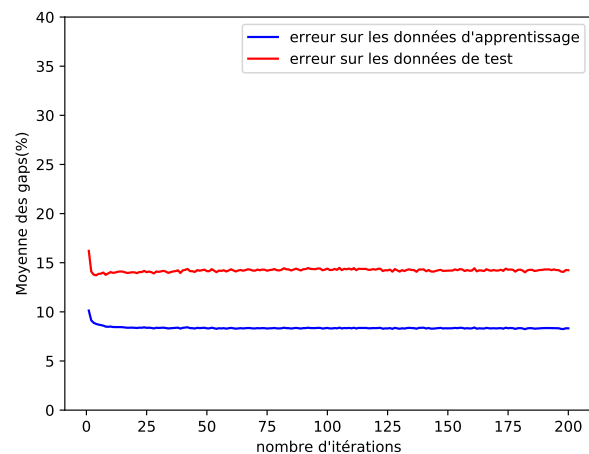


FIGURE 7.24 – Cette figure représente l'évolution du gap lorsqu'on fait varier le nombre d'itérations lors de l'apprentissage du schéma d'apprentissage **INDIC_TEMPS** pour la prédiction du numéro de période de la dernière opération de recharge. La courbe bleue est l'évolution de l'erreur sur les données d'apprentissage et la courbe rouge est l'évolution de l'erreur sur les données de test.

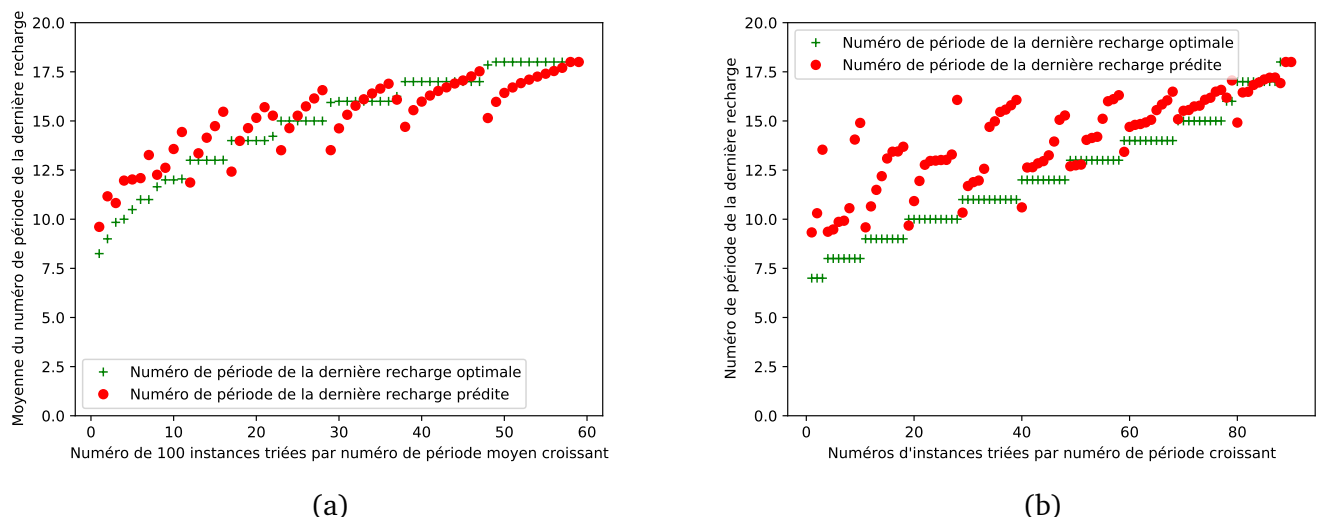


FIGURE 7.25 – Résultats du schéma d'apprentissage **INDIC_TEMPS** pour la prédiction du numéro de période de la dernière opération de recharge. (a) représente les données d'apprentissage et (b) représente les données de test. Les points rouges sont les valeurs prédites et les points verts sont les valeurs optimales.

7.5 Conclusion

Dans ce chapitre, on a présenté plusieurs réseaux de neurones et les résultats expérimentaux de ceux-ci. On a utilisé pour la phase expérimentale, ~~les~~ ^{des} instances dont le nombre de périodes est 20. Au cas où on veut adapter nos réseaux à des instances dont le nombre de période est supérieur à 20, on peut transformer ces instances. Si on veut limiter à 20 le nombre des périodes des instances de notre problème de production, en procédant comme suit : ^{grammaire -}

- Si N se situe entre $20.K$ et $20.(K + 1)$ avec $K \geq 1$, alors, pour tout u allant de 1 à $[N/K]$, on fusionne les coefficients des différents vecteurs indexés de 1 à N de la façon suivante :

- il faudrait expliquer pourquoi
- S'il s'agit de R_i qui donne les rendements, on pose $R_u^* = \lceil (\sum_{k.u \leq i < (k+1).u} R_i) / K \rceil$;
 - S'il s'agit de $Cost_i^V$ qui donne les coûts de production, on pose $Cost_u^{V*} = \lceil (\sum_{k.u \leq i < (k+1).u} Cost_i^V) / K \rceil$;
 - S'il s'agit du coût d'activation $Cost^F$, on pose $Cost_u^F = \lceil Cost^F / K \rceil$;
 - Pour ce qui est des fenêtres de temps $[m_q, M_q]$ et de μ_q , $q = 1, \dots, Q$, sur les périodes et quantités de recharge, on divise chaque quantité m_q , M_q et μ_q par K (division entière).

□ Quand on obtient le résultat $Cost$ et T , où $Cost$ est le coût et T le numéro de période de la dernière recharge, alors on multiplie $Cost$ et T par K pour reconstituer le résultat souhaité.

pas clair

La matrice des entrées étant une matrice creuse car les entrées sont de taille différente, on peut la remplir avec des zéros ou de façon cyclique afin que les données forment une matrice rectangulaire. C'est cette opération qui permettra de connaître exactement le nombre de valeurs en entrées des réseaux **SIMPLE** et **MIXTE**.

De nombreuses questions restent à résoudre notamment étendre notre approche à plusieurs véhicules; adapter les algorithmes à l'optimisation de la tournée. Le chapitre suivant est consacré à la présentation brève de deux perspectives notamment l'inclusion de la tournée du véhicule dans la décision et à la question de la robustesse à travers la gestion des incertitudes.

→ Ceci n'est pas une conclusion →
expliquer 1 peu plus et mettre dans
1 § à part.

→ faire 1 vraie conclusion en reprenant
les idées clés du chapitre.