Intelligence Artificielle
Réseaux de neurones
Perceptron et perceptron multi-couches avec Neuroph

MEYER Cyril

S6 Printemps 2017

Table des matières

Chapitre 1

Apprentissage d'une fonction

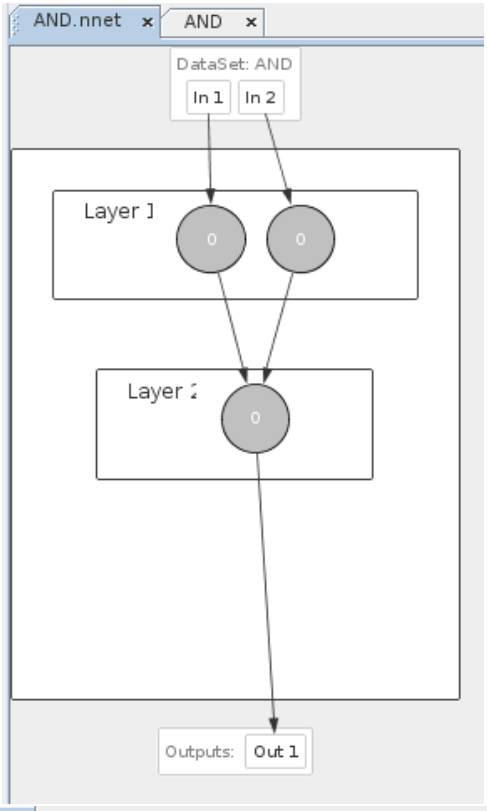
1.1 Fonctions booléennes

1.1.1 "Et" logique

La table de vérité du "Et" logique est la suivante :

a	b	a $\wedge$ b
0	0	0
0	1	0
1	0	0
1	1	1

Nous créons un percetron mono-couche avec 2 neurones d'entrée et 1 neurone de sortie. Le dataset utilisé est la table logique de la fonction et (cf dataset_AND.csv).



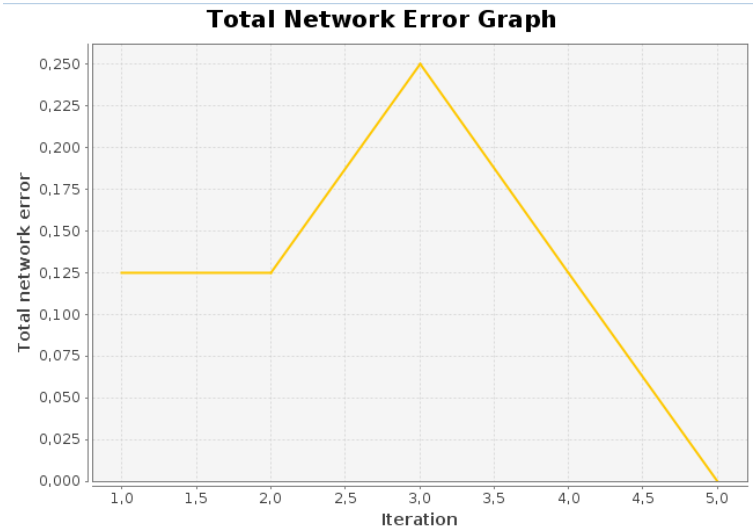
AND.nnet | AND

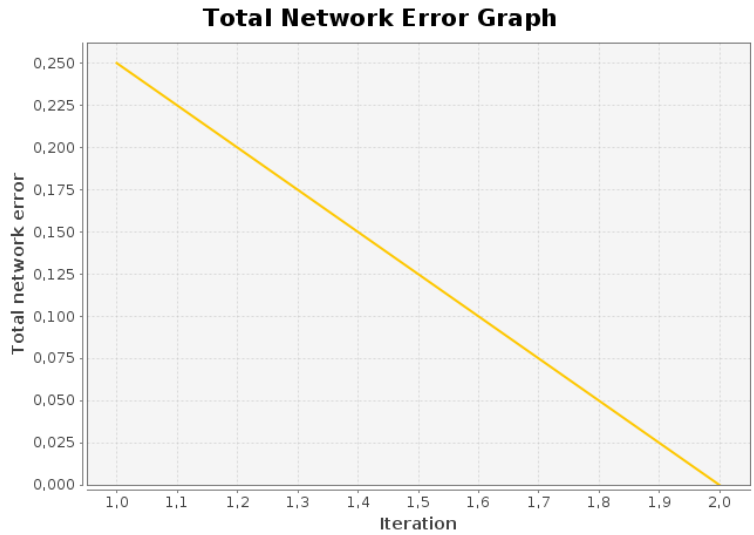
Input1	Input2	Output1
0.0	0.0	0.0
0.0	1.0	0.0
1.0	0.0	0.0
1.0	1.0	1.0

Configuration d'entraînement :

Max Error	0.01
Learning Rate	0.2
Momentum	0.7

Résultats (2 test) :



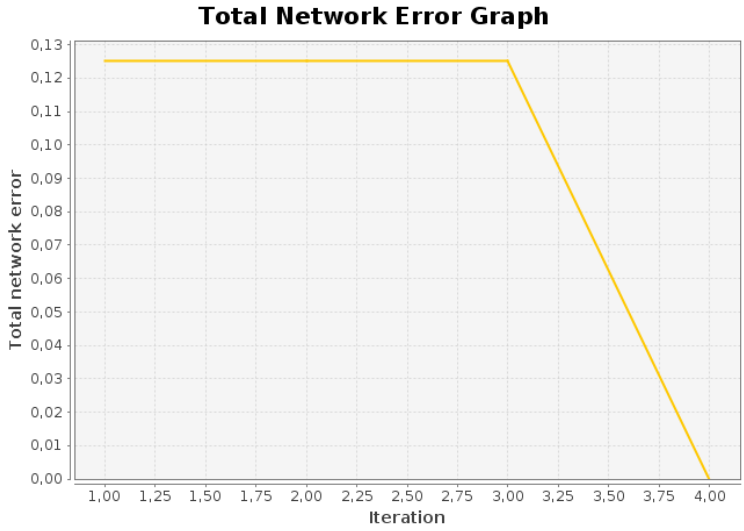


La courbe tend très rapidement vers la solution parfaite (taux d'erreur de 0.0). On peut tout de même remarqué que dans lors du premier essai, le réseaux s'est dégradé lors de la troisième itération.

Configuration d'entraînement :

Max Error	0.01
Learning Rate	0.1
Momentum	0.7

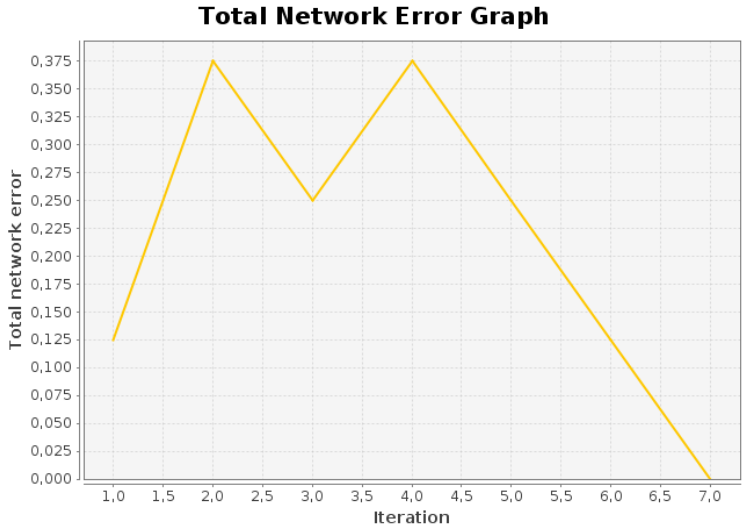
Résultats :



Configuration d'entraînement :

Max Error	0.01
Learning Rate	1
Momentum	0.7

Résultats :



On ce rend compte que le pas d'apprentissage importe beaucoup sur les résultats du réseau.

1.1.2 "Équivalence" logique

La table de vérité de l'equivalence logique est la suivante :

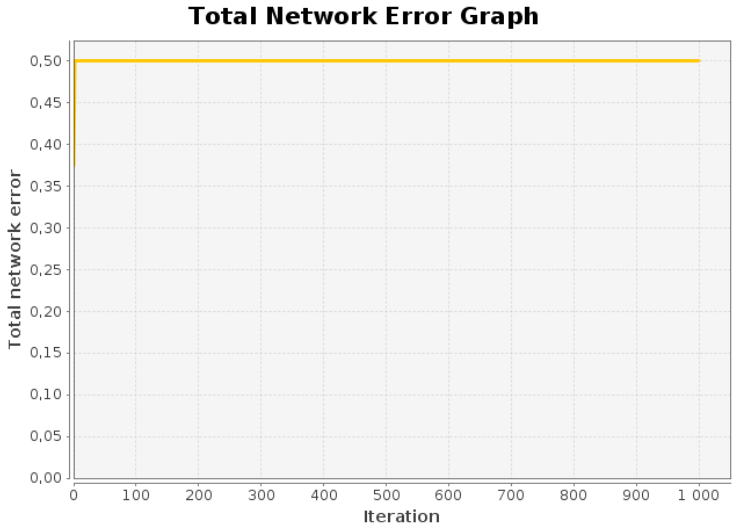
a	b	a $\iff$ b
0	0	1
0	1	0
1	0	0
1	1	1

Nous créons un percetron mono-couche avec 2 neurones d'entrée et 1 neurone de sortie. Le dataset utilisé est la table logique de la fonction d'équivalence (cf dataset_EQ.csv).

Configuration d'entraînement :

Max Error	0.01
Limite Max Iterations	1000
Learning Rate	0.2
Momentum	0.7

Résultats :



L'apprentissage ne permet pas de converger. Les pas d'apprentissage différents ne changerais rien, un perceptron mono-couche ne peut pas résoudre ce genre de problème car ce n'est pas linéairement séparable.

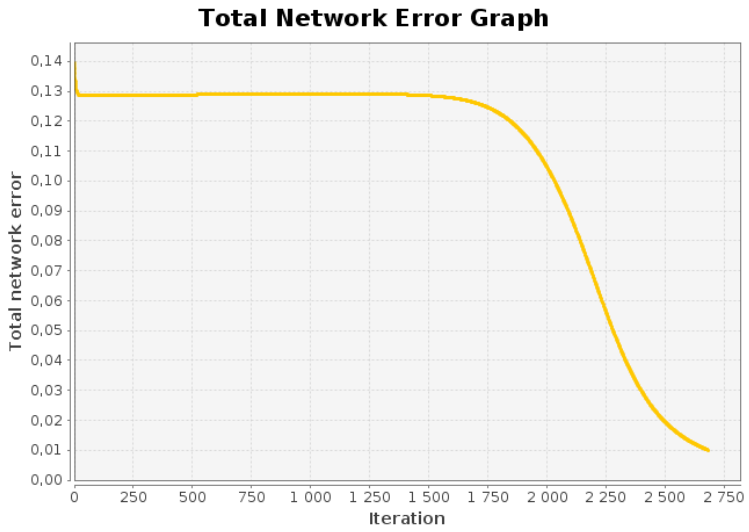
Nous créons un percetron multi-couche (1 couche cachée de 3 noeuds) avec 2 neurones d'entrée et 1 neurone de sortie. Le dataset utilisé est la table logique

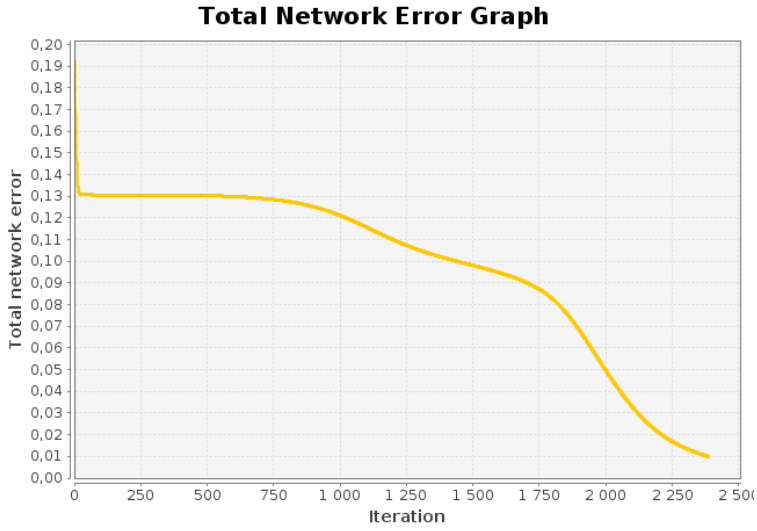
de la fonction d'équivalence (cf dataset_EQ.csv).

Configuration d'entraînement :

Max Error	0.01
Learning Rate	0.2
Momentum	0.7

Résultats (2 test) :





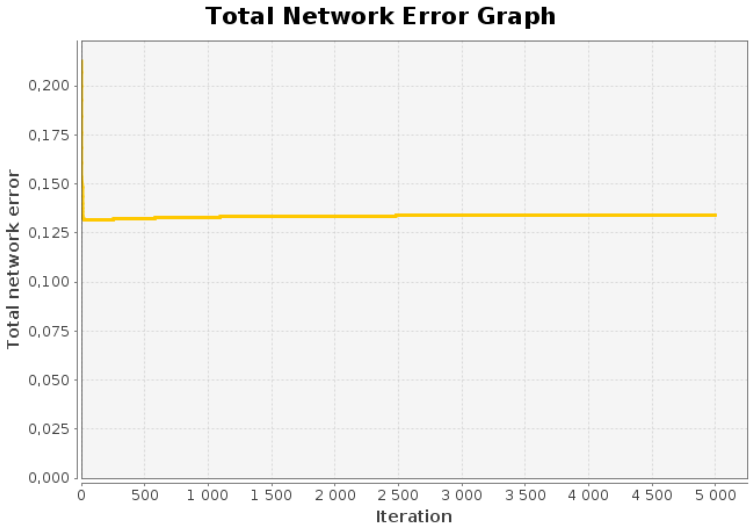
La courbe d'apprentissage stagne un bon moment avant de tendre plus rapidement vers la solution.

Influence des paramètres

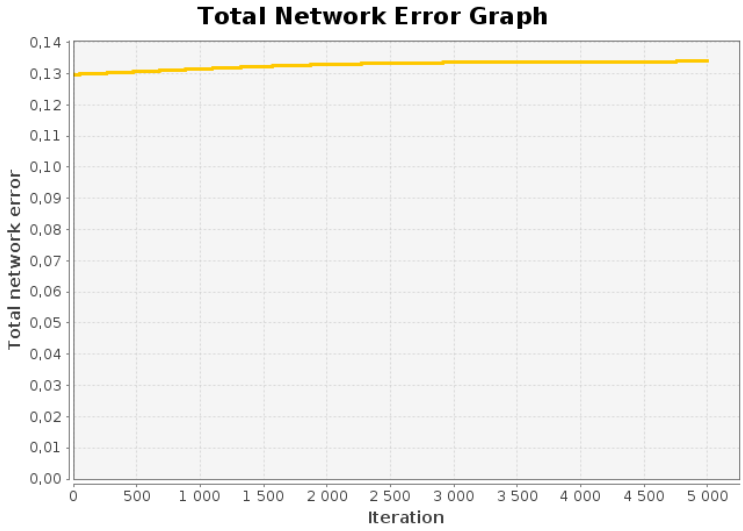
Lorsque ce ne sera pas précisé, la configuration d'entraînement sera la suivante :

Max Error	0.01
Limite Max Iterations	5000
Learning Rate	0.2
Momentum	0.7

1. Nombre de couche Résultats (2 couches de 3 noeuds) :

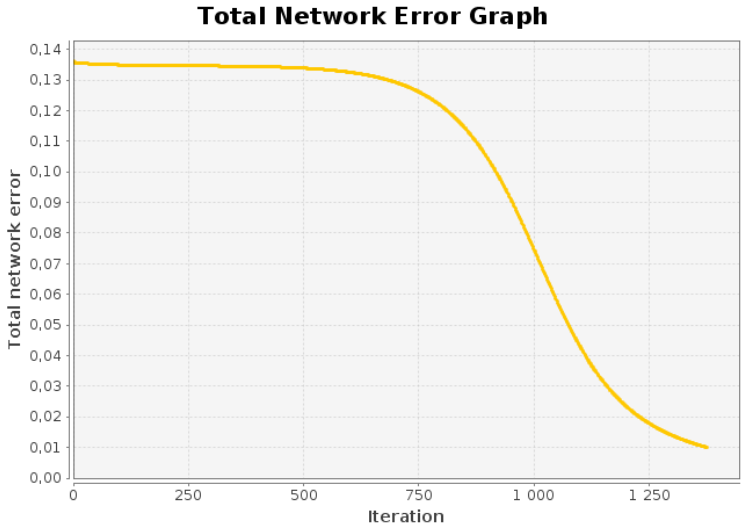


Résultats (3 couches de 3 noeuds) :

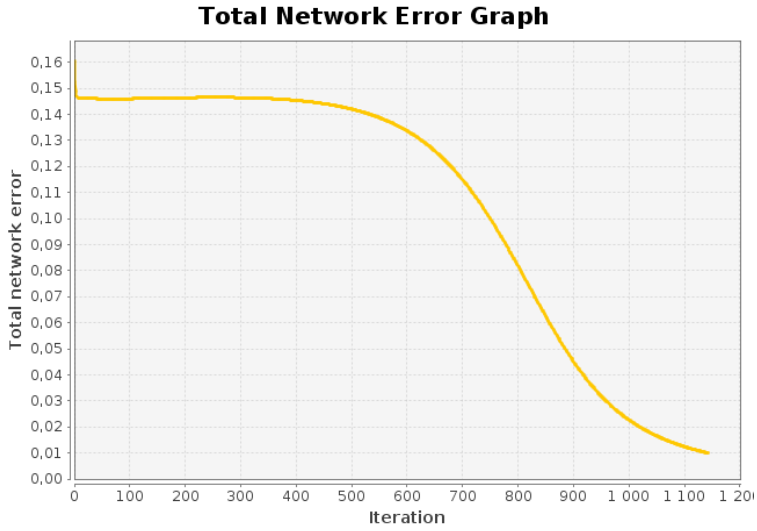


Conclusion : Ajouté des couches n'améliore pas les résultats.

2. Nombre de noeuds Résultats (1 couches de 12 noeuds) :

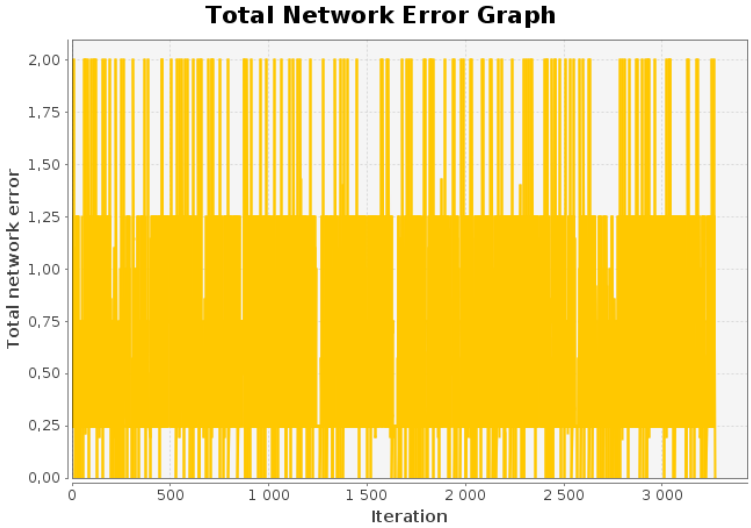


Résultats (1 couches de 24 noeuds) :

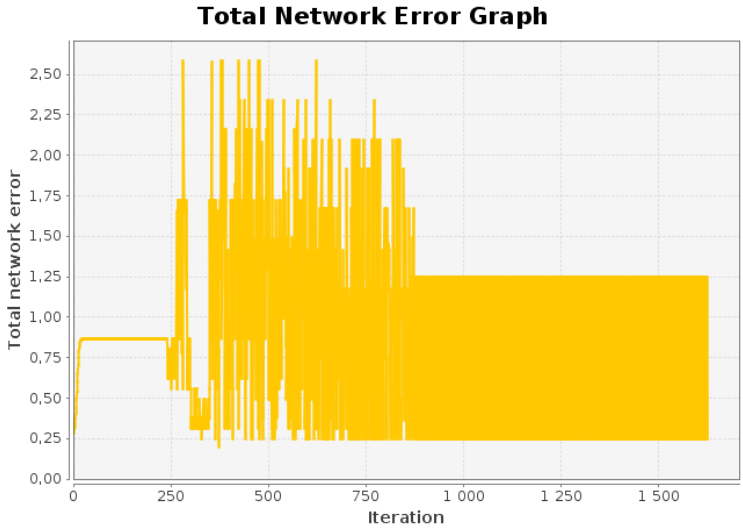


Conclusion : Ajouté des noeuds permet de converger plus rapidement vers le résultat.

3. Fonction d'activation Tanh Résultats :

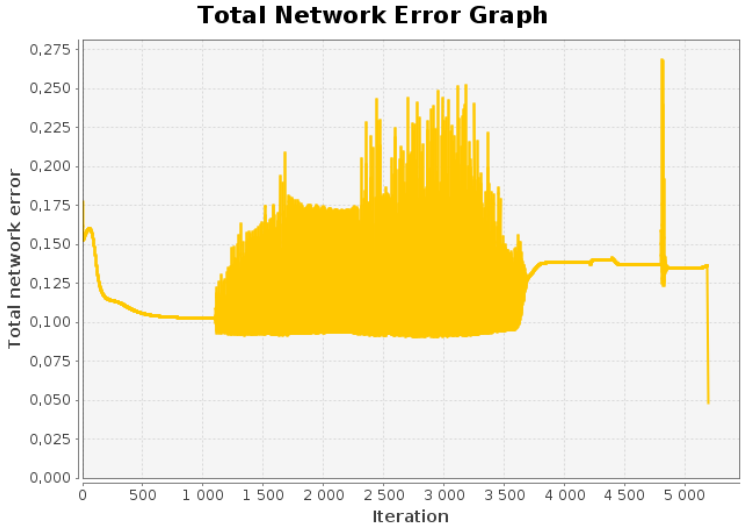


Résultats (Pas d'apprentissage de 0.01) :

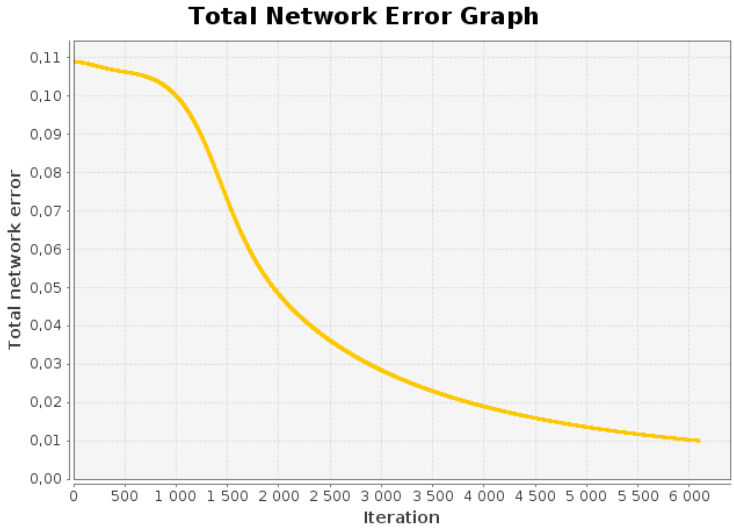


Conclusion : La fonction Tanh crée des variation de l'erreur beaucoup plus grande, cela n'améliore pas la convergence.

4. Pas d'apprentissage Résultats (Pas d'apprentissage de 1) :



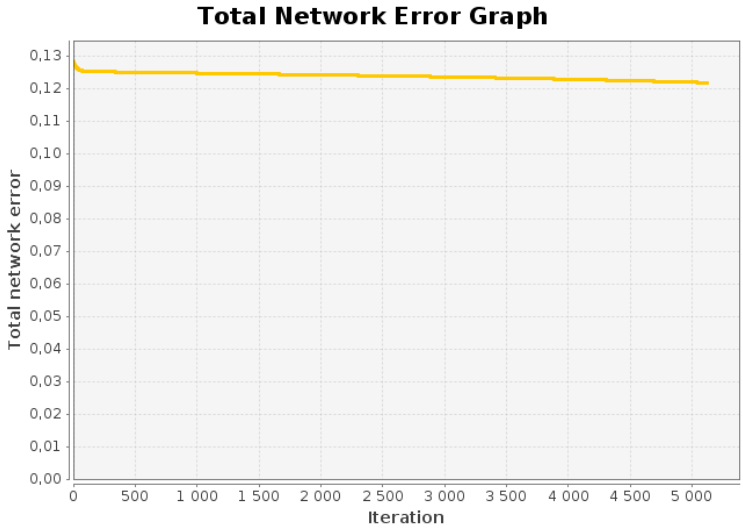
Résultats (Pas d'apprentissage de 0.01) :



Conclusion : Un pas d'apprentissage plus petit permet de converger plus précisément vers le résultat.

Conclusion En utilisant les conclusions précédentes, on peut se demander si un pas d'apprentissage petit ainsi qu'un nombre de noeuds supplémentaire améliorerais les résultats.
Résultat d'un test avec les paramètres suivants :

Max Error	0.01
Learning Rate	0.01
Momentum	0.7
Couches	1
Noeuds	20



Conclusion : Les résultats ne sont pas positifs.

1.2 Fonctions simples

1.2.1 Fonction $f(x) = x^2$

Je commence tout d'abord en crée plusieurs réseaux de neurones multi-couches, le problème $f(x) = x^2$ est un problème qui n'est pas linéairement séparable, donc impossible a résoudre avec un perceptron mono-couche.

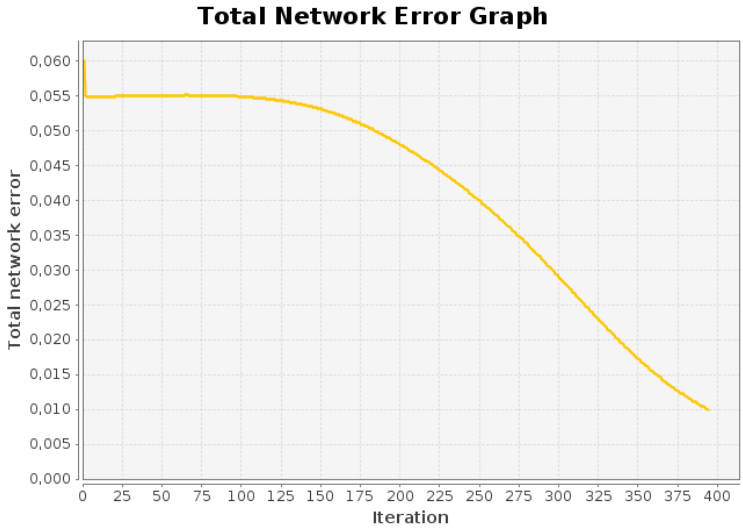
J'ai choisis de tester différent dataset. Respectivement, 21, 101, 302 et 704 exemples. Les dataset sont disponible sous les noms : "dataset_SQUARE_xxx" Ces dataset sont crée avec le programme "squareDataset" disponible avec le dossier.

Ces premiers tests me permettent de comprendre quels réseaux et quels dataset choisir pour les tests suivants qui seront plus précis.

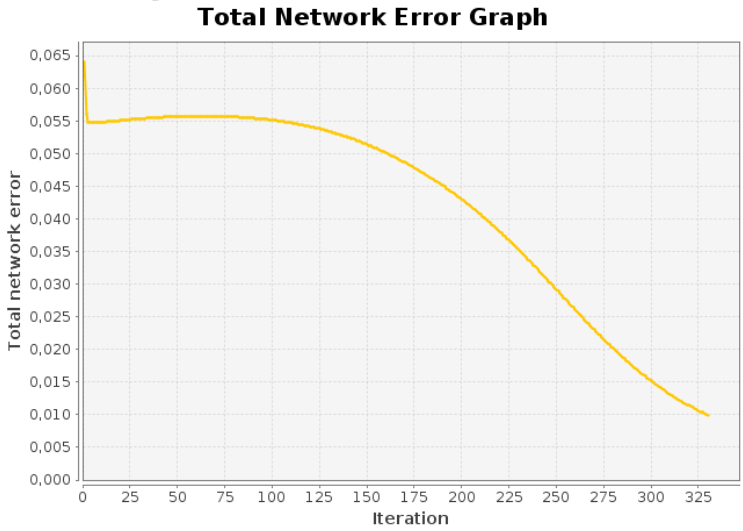
Le premier dataset (21) est utilisé pour tester les différence entre les différentes architectures de réseau.

Max Error	0.01
Learning Rate	0.2
Momentum	0.7

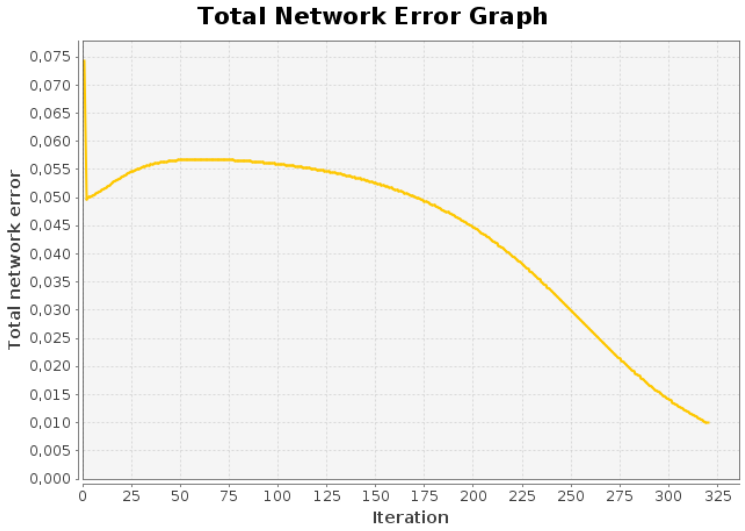
Perceptron multicouche : 1 couche de 8 noeuds.



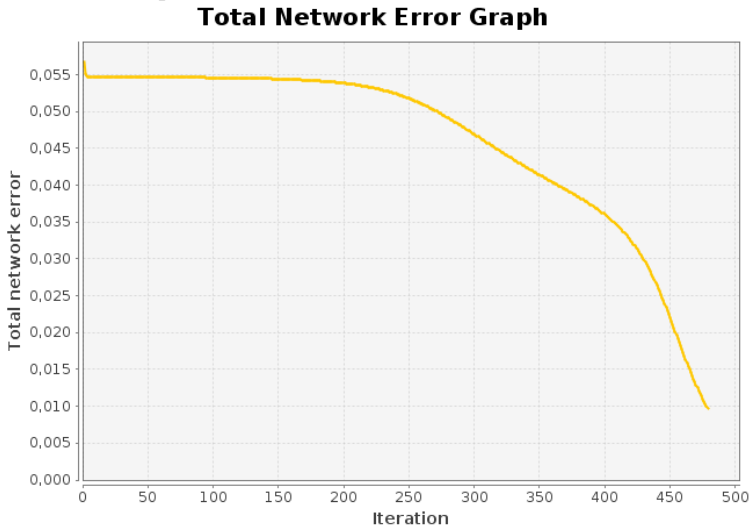
Perceptron multicouche : 1 couche de 16 noeuds.



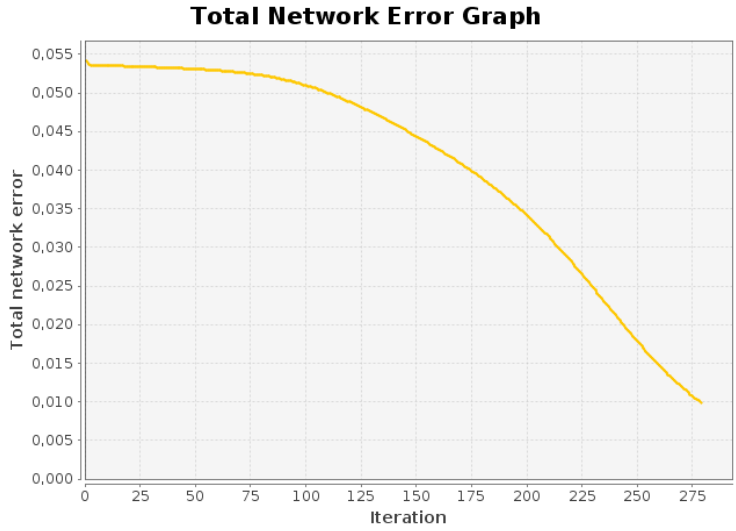
Perceptron multicouche : 1 couche de 32 noeuds.



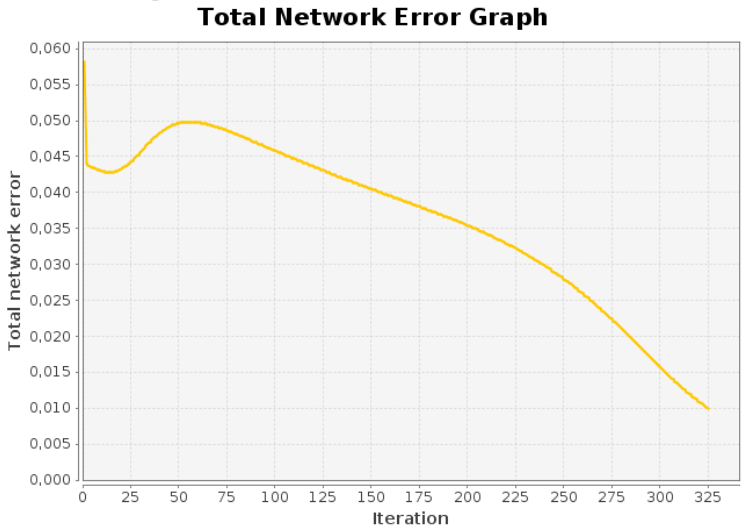
Perceptron multicouche : 2 couche de 8 noeuds.



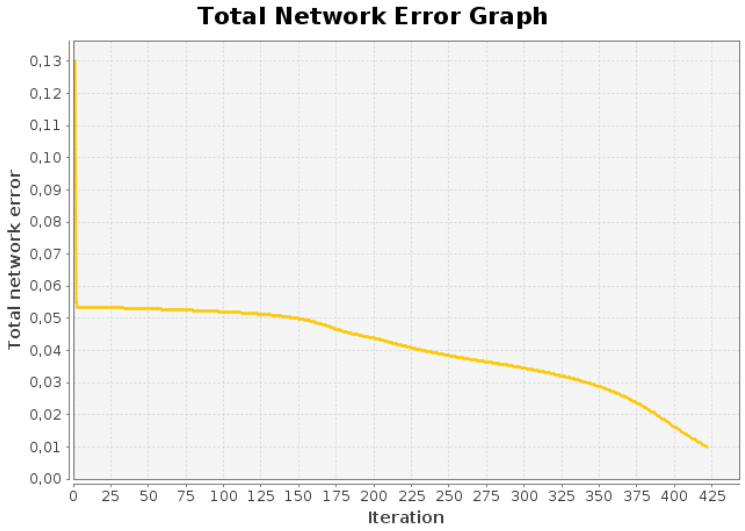
Perceptron multicouche : 2 couche de 16 noeuds.



Perceptron multicouche : 2 couche de 32 noeuds.



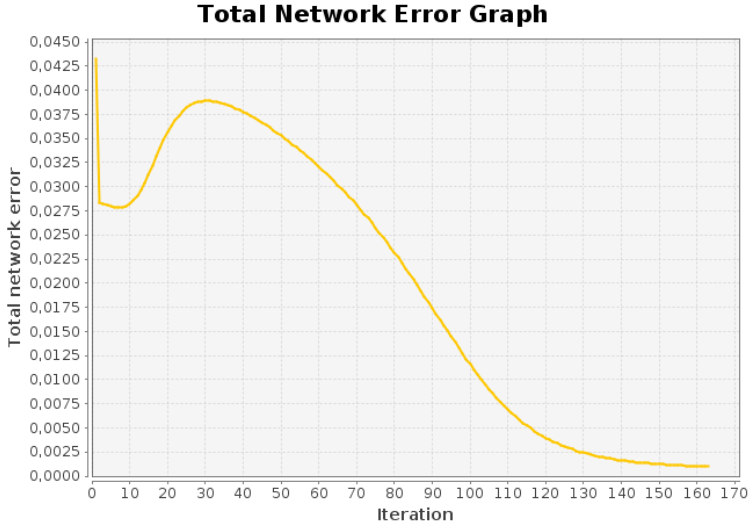
Perceptron multicouche : 3 couche de 16 noeuds.



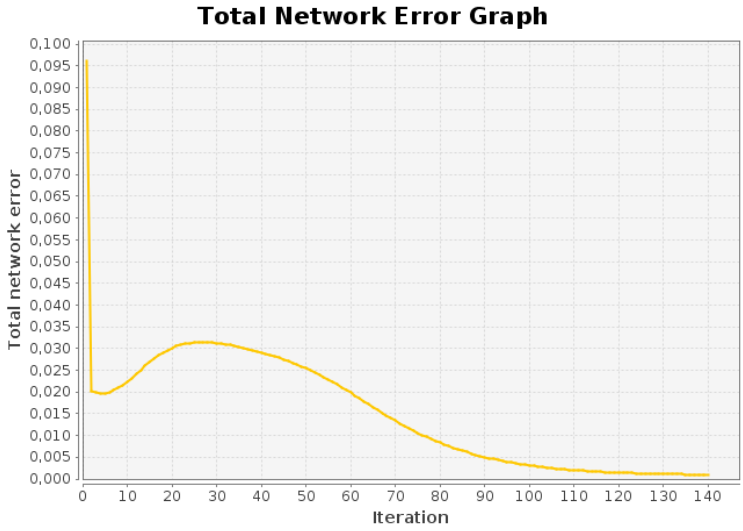
Le second dataset (101) est utilisé pour choisir parmi les meilleurs résultats précédents.

Max Error	0.001
Learning Rate	0.2
Momentum	0.7

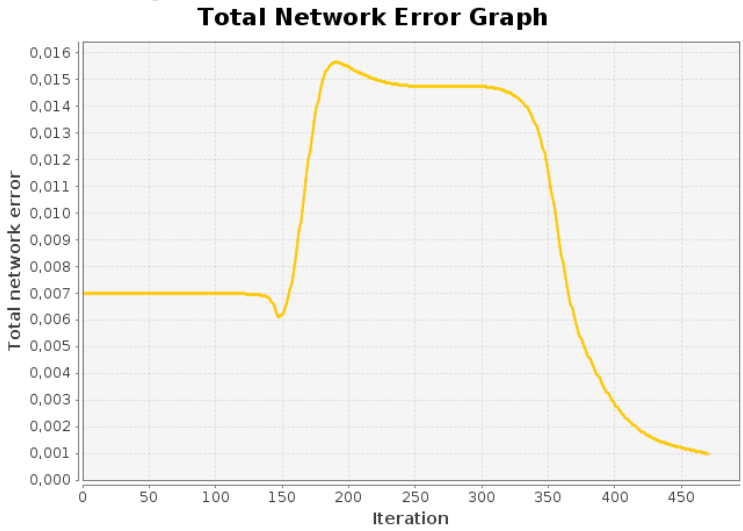
Perceptron multicouche : 1 couche de 8 noeuds.



Perceptron multicouche : 1 couche de 16 noeuds.



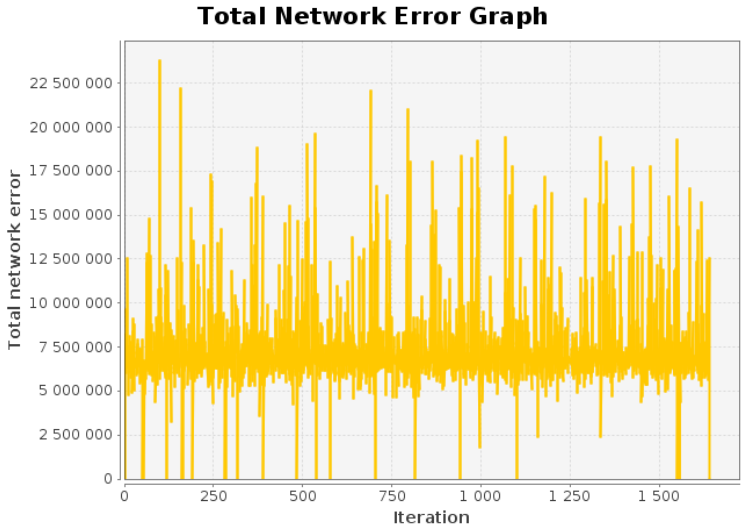
Perceptron multicouche : 2 couche de 16 noeuds.



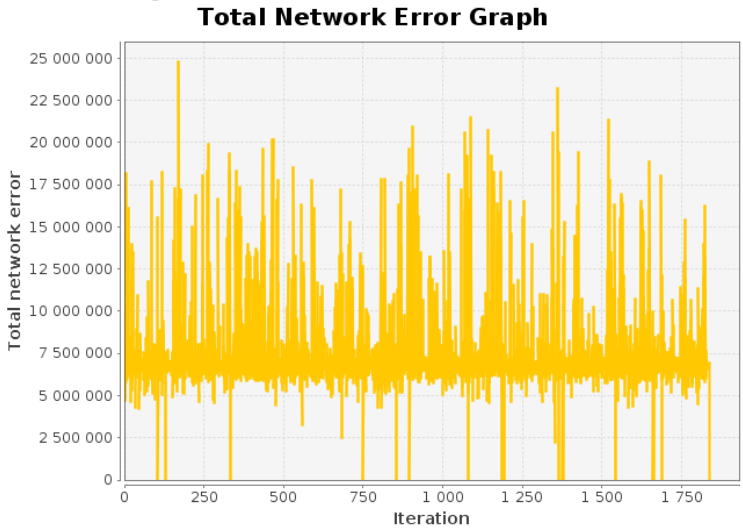
Le troisième dataset (302) est utilisé pour choisir parmi les meilleurs résultats précédents.

Max Error	0.1
Learning Rate	0.2
Momentum	0.7

Perceptron multicouche : 1 couche de 8 noeuds.



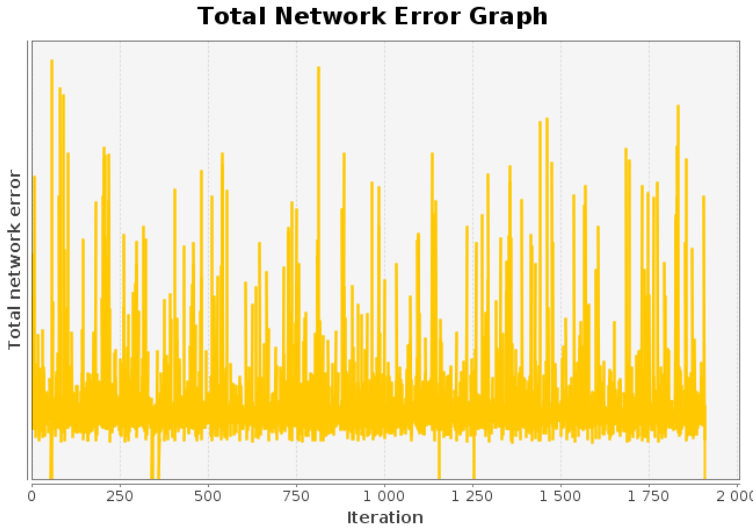
Perceptron multicouche : 1 couche de 16 noeuds.



Aucun des réseau ne converge.
Un dernier essai avec le dernier dataset (704) :

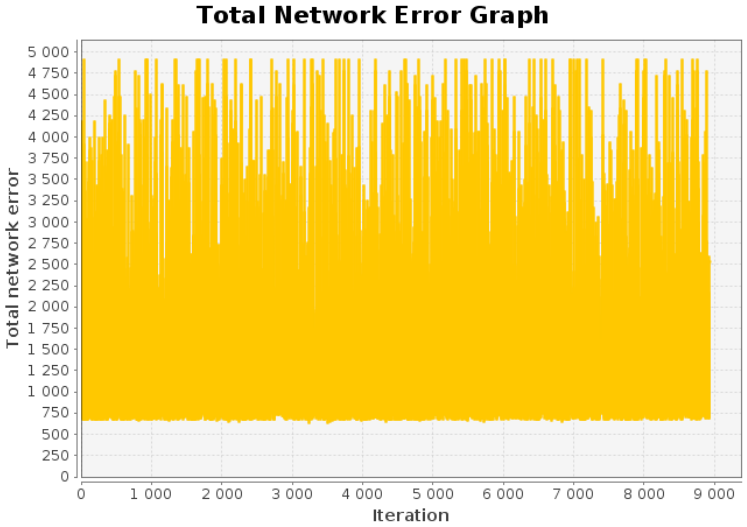
Max Error	1000
Learning Rate	0.2
Momentum	0.7

Perceptron multicouche : 1 couche de 16 noeuds.

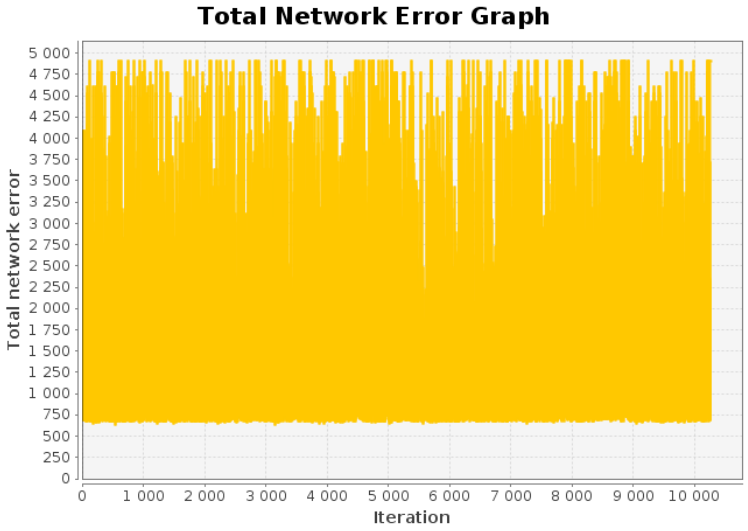


A ce moment, je décide de créer un nouveau dataset calqué sur le second dataset (101) mais contenant tout de même plus d'exemples. Le dataset 401 : de -2 à 2 avec un pas de 0.01. Les résultats n'étant pas concluant (pas de convergence avec les réseaux), je décide de créer un dataset, le dataset 101_2 : de -10 à 10 avec un pas de 0.2.
Une fois de plus, le dataset est trop complexe et aucun des perceptron multi-couche n'arrive a converger (testé avec plus de 10000 itération maximum)

Perceptron multicouche : 1 couche de 16 noeuds.



Perceptron multicouche : 5 couche de 16 noeuds.

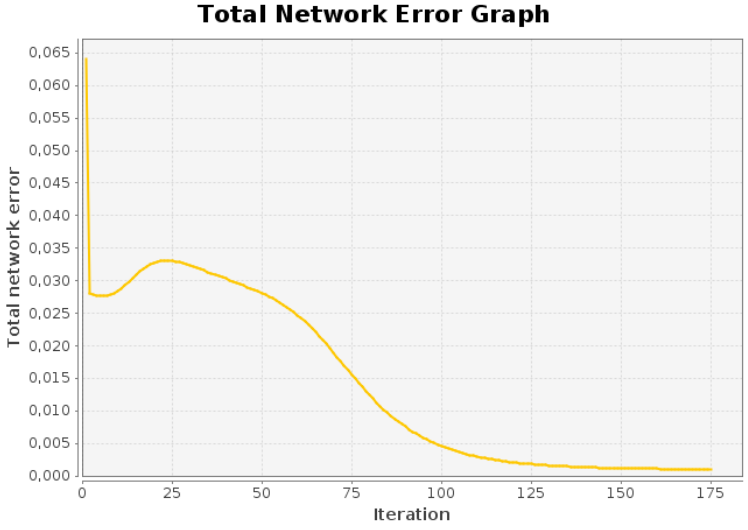


Une dernière tentative de création de dataset, le dataset 101_3 : les -100 à 100 avec un pas de 2. Les résultats ne convergent toujours pas, malgré les changements de paramètres d'apprentissages.

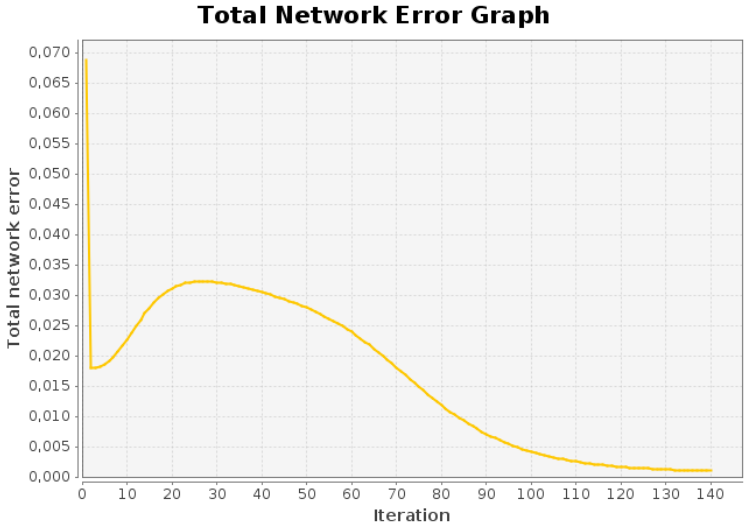
Pour améliorer mes résultats précédents, je décide de travailler avec le dataset 101 et en changeant les paramètre d'apprentissages afin de trouver de bons paramètres d'apprentissage pour mes 3 réseau retenus.

Max Error	0.001
Learning Rate	0.2
Momentum	0.7

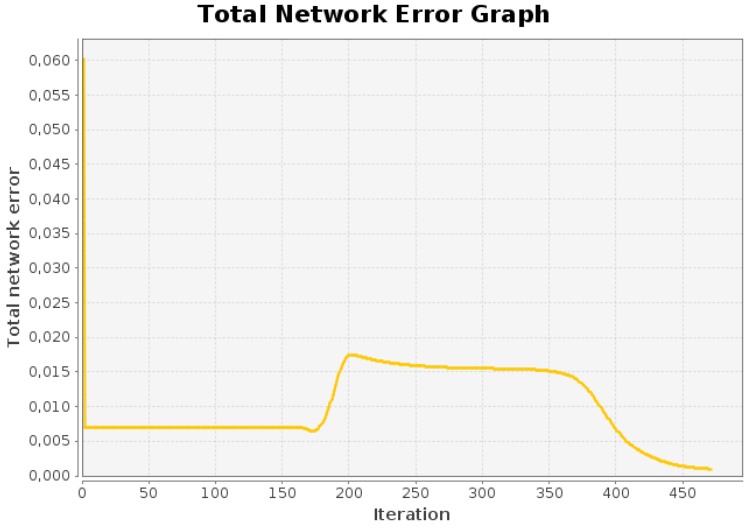
Perceptron multicouche : 1 couche de 8 noeuds.



Perceptron multicouche : 1 couche de 16 noeuds.

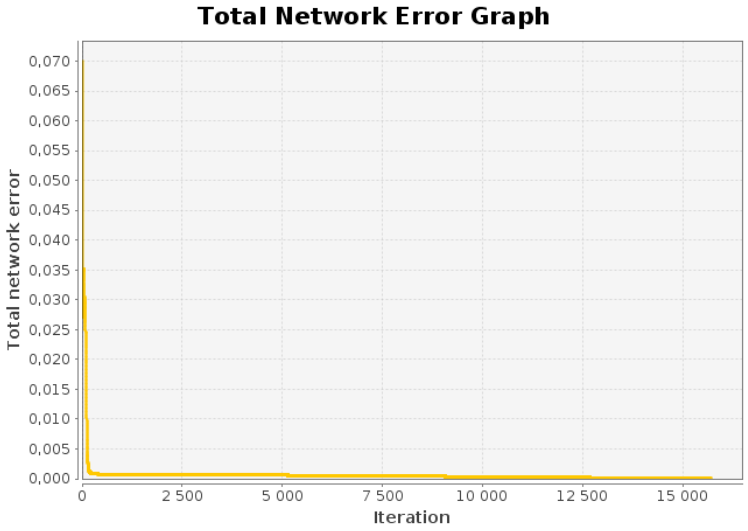


Perceptron multicouche : 2 couche de 16 noeuds.

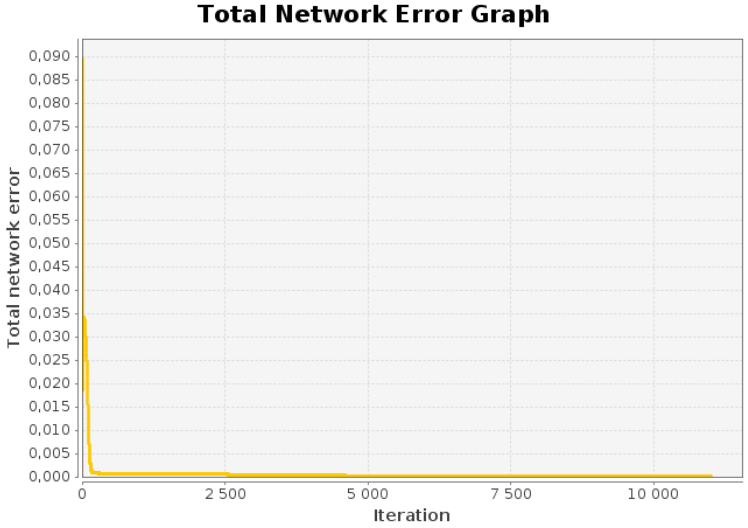


Max Error	0.0001
Learning Rate	0.2
Momentum	0.7

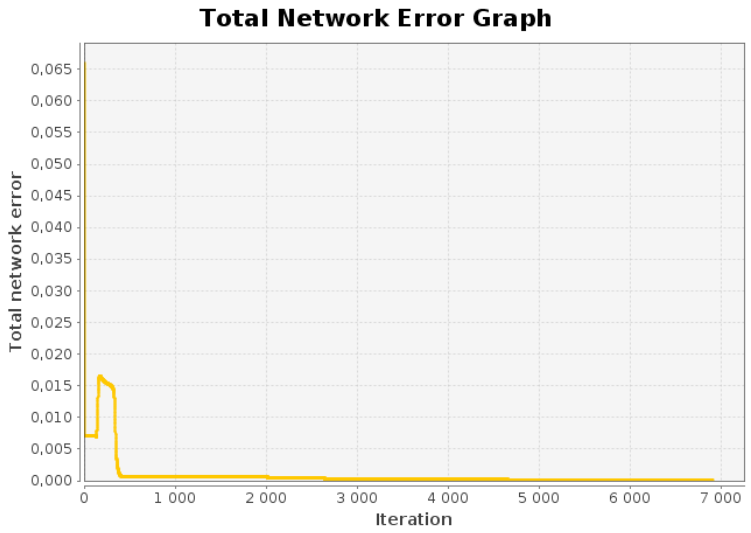
Perceptron multicouche : 1 couche de 8 noeuds.



Perceptron multicouche : 1 couche de 16 noeuds.

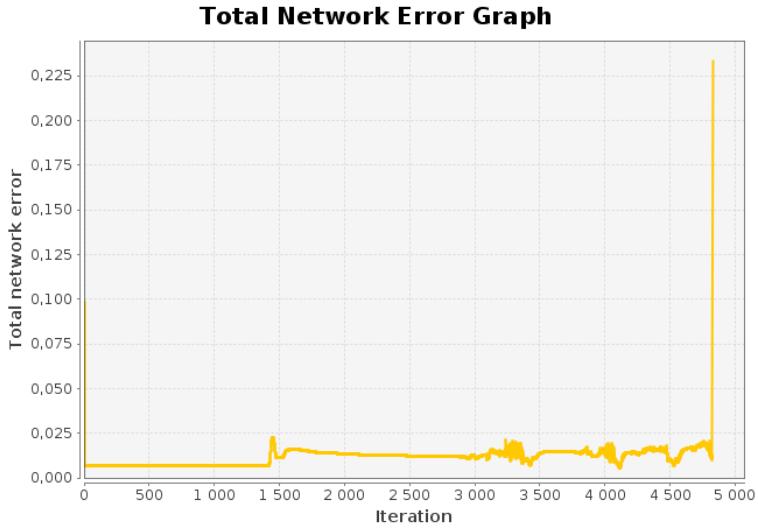


Perceptron multicouche : 2 couche de 16 noeuds.



A ce moment, on se rend compte que pour une précision supplémentaire, le réseau a 2 couche devient intéressant.

Petite note intéressantes, voici le graph d'erreur d'un perceptron multi-couche : 3 couche de 16 noeuds (qui ne converge pas).

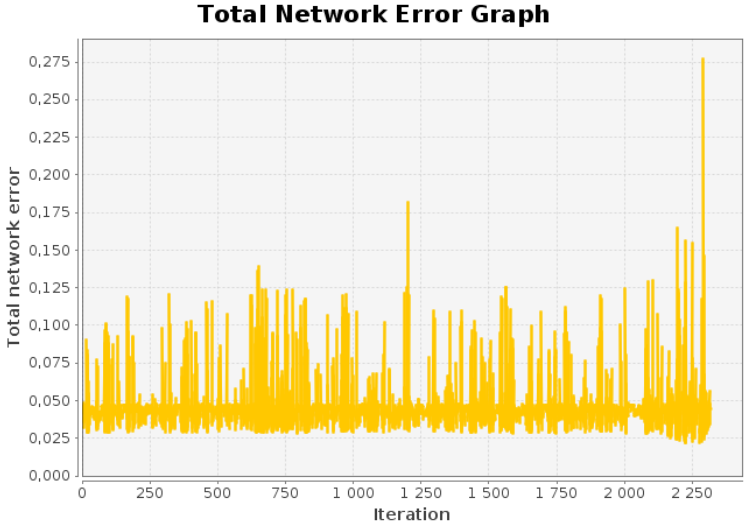


Pour continuer, nous allons donc nous baser sur ces deux dernier perceptrons multicouche : 1 : 1 couches de 16 noeuds et 2 : 2 couches de 16 noeuds

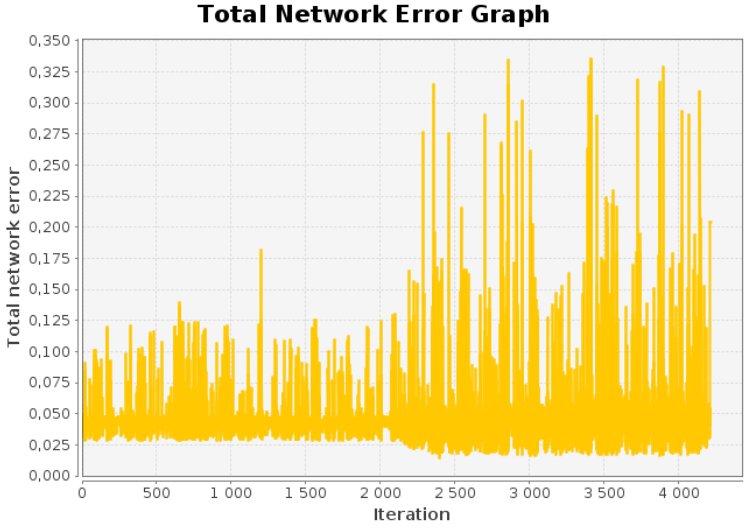
Variation du Learning Rate

Max Error	0.0001
Learning Rate	0.02
Momentum	0.7

Perceptron multicouche : 1 couche de 16 noeuds.

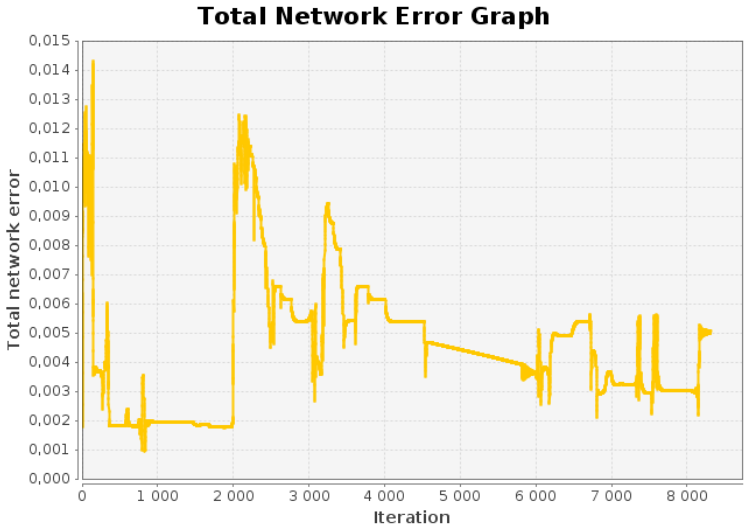


Perceptron multicouche : 2 couche de 16 noeuds.



Max Error	0.0001
Learning Rate	2
Momentum	0.7

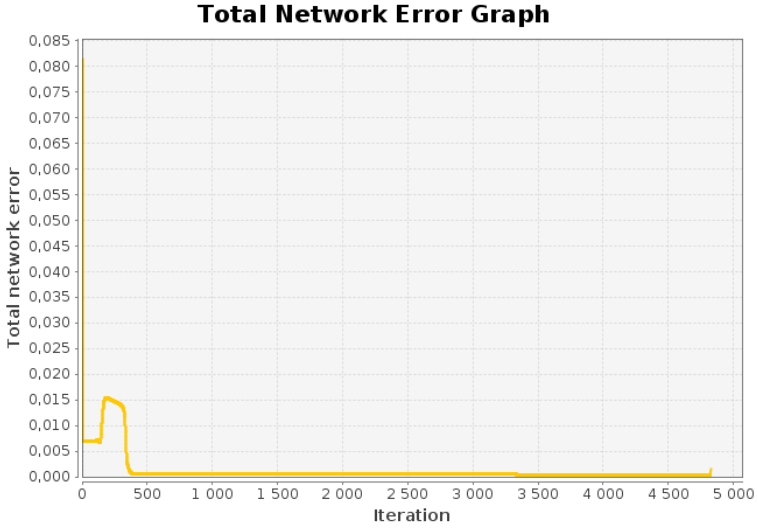
Perceptron multicouche : 2 couche de 16 noeuds.



Variation du Momentum

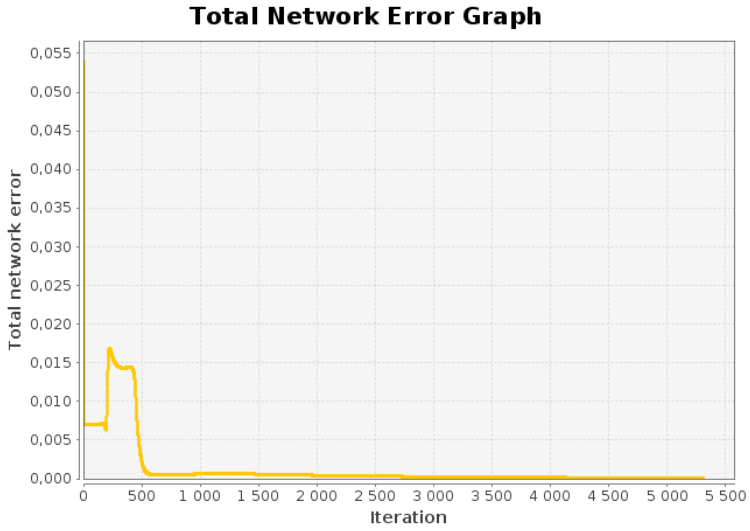
Max Error	0.0001
Learning Rate	0.2
Momentum	0.07

Perceptron multicouche : 2 couche de 16 noeuds.



Max Error	0.0001
Learning Rate	0.2
Momentum	7

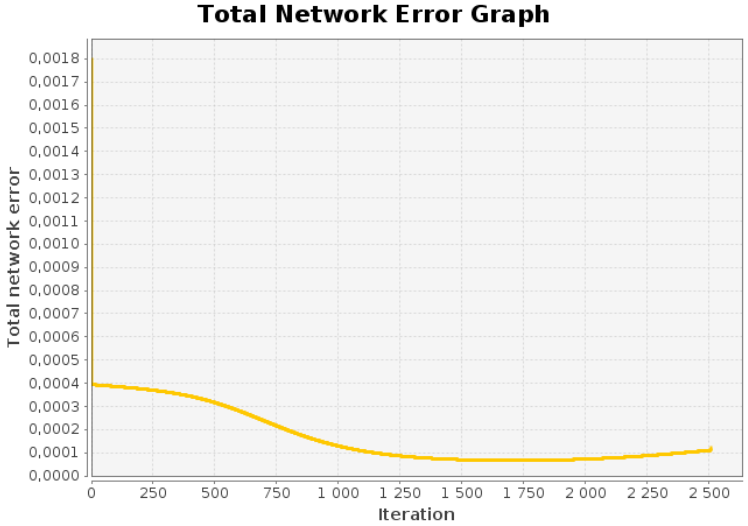
Perceptron multicouche : 2 couche de 16 noeuds.



Après plus de recherche dans la documentation de Neuroph, il se trouve que les données de sortie ne peuvent pas être supérieur a 1. Je choisie donc de créer un nouveau dataset de 2001 exemples, de -100 à 100 avec un pas de 0.1 mais les données sont divisé par 10000 (dataset_SQUARE_2001.csv)

Je test uniquement ce dataset avec un réseau d'une couche caché et 16 noeuds.

Max Error	0.00001
Learning Rate	0.2
Momentum	7



Conclusion

Le dataset utilisé est très important. Lorsqu'il contient trop peu d'exemple, le réseau peut renvoyer des erreurs basse et pourtant ne pas réellement reproduire les effets demandé. Lorsqu'il contient trop d'exemple, le réseau devient lent, et a tendance a ne pas réussir a converger aussi facilement. Le juste milieu est difficile a trouvé, et dans mon cas, c'est uniquement des tests empirique qui m'ont permit de décider quel dataset utiliser.

Finalement, les résultats de mon perceptron multicouche de 2 couche et 16 noeuds sont satisfaisant.

Mon objectif initiale était d'arriver a faire descendre l'erreur en dessous de 0.0001, et c'est donc une mission accomplie.

La variation de tous les paramètres est importante, malheureusement il n'existe pas de règles ou de guide a suivre pour créer un bon réseau de neurones.

De nombreux tests effectués n'apparaissent pas dans ce rapport (faute de bon résultats) mais les 19 réseaux de neurones et les 10 ensemble de test sont disponibles dans le projet neuroph si nécessaire.

Ce que je retiens, c'est que les réseau de neurone, sont encore très difficile a utiliser, pas a cause de la mise en oeuvre, mais parce qu'une grosse partie du problème reviens a faire des test empirique pour trouver une architecture viable pour le réseau.

Chapitre 2

Annexes

2.1 NeurophStudio

De nombreux problème lié a l'utilisation du logiciel NeurophStudio on retardé l'avancement de ce rapport. Voici donc de petit conseil pour utiliser convenablement le logiciel.

2.1.1 Reset / Randomize

Pour reset ou randomize un réseau de neurone, la meilleur façon reste de le fermé, de le ré-ouvrir, de lui donné un training set, de lancé un première entraînement, de l'annule et de randomize ou reset le réseau. De cette façon, les poids devrait apparaître randomize ou reset a l'écran.

2.1.2 Format de données

Les données de sortie du perceptron multi-couche ne peuvent pas prendre n'importe quel valeur, il faut se renseigner préalablement sur la documentation pour les données.