Master TRIED,

TPB01: Rapport Partie 1- questions 1- 4

Sujet:

Régression par Perceptron Multicouches

Réalisé par:

FOUTSE YUEHGOH

Année universitaire : 2018/2019

Objectif: Mettre en œuvre une méthodologie constructive pour déterminer une fonction de régression à partir de données brutes simulées pour approximer l'espérance à l'aide de perceptrons multicouches.

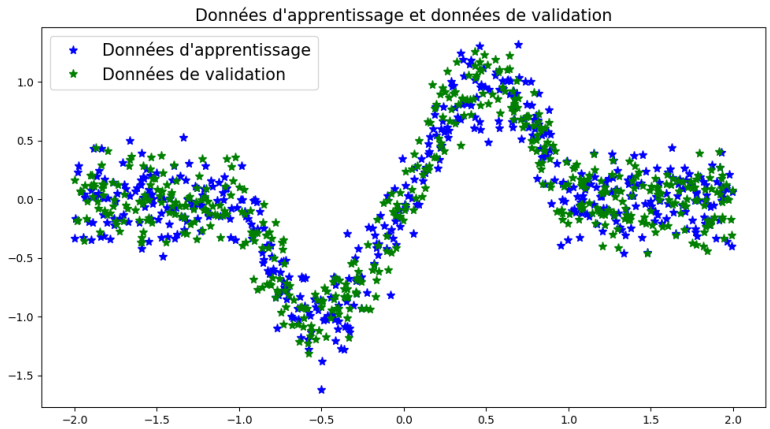
On effectuera ce TP uniquement en utilisant la procédure **schioler** définie de la façon suivante :

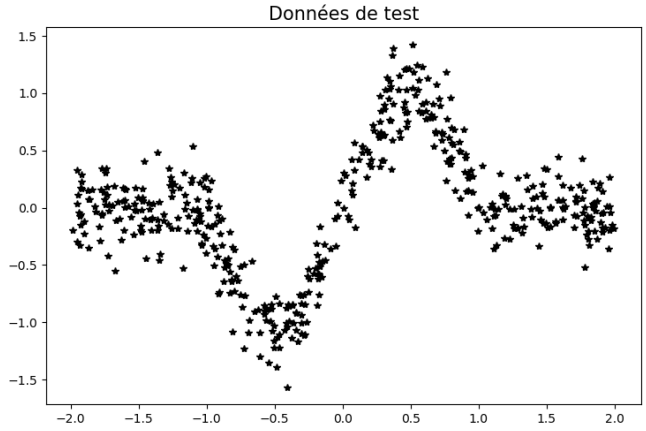
- Sur les abscisses les points suivent une distribution uniforme sur]-*2, 2*[.
- $y = f(x) + delta$ où :

  $f(x) = sin(p\,x)$ *sur*] *-1, 1* [
  $= 0$ *sur*] *-2,-1*] U [*1, 2* [

  et *delta* est un bruit qui suit une distribution normale $N(0, s2)$
  avec l'écart type $s = 0{,}2$.

1) Nous avons généré un ensemble de 1000 points bruitées avec la fonction de **schioler** à partir de laquelle nous avons constitué un ensemble d'apprentissage (500 points) et un ensemble de validation (500 points). Puis nous avons généré par la suite un ensemble test contenant 500 points avec la même fonction. Les données générées sont représentées par les 2 figures suivantes:

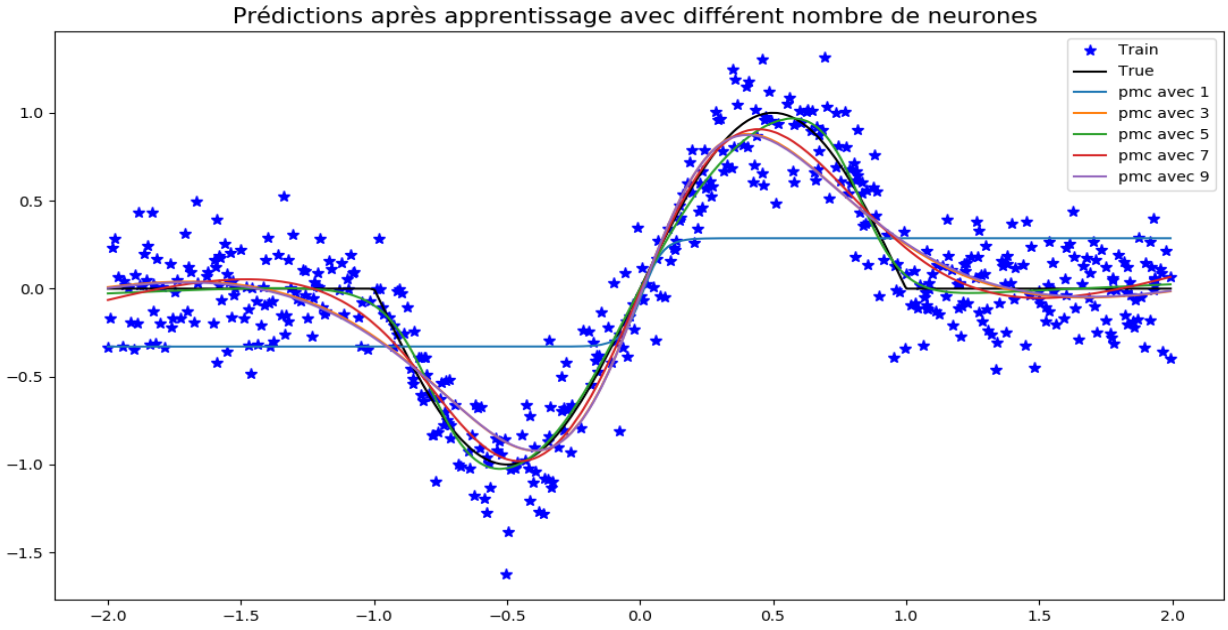




Les données d'apprentissage, validation et test ont été toute générées avec la fonction de **schioler** et ceci de manière uniforme sur l'intervalle [-2,2] ce qui assure une bonne distribution, toutes les zones de représentations sont explorée. Nous estimons que 500 données seront suffisantes pour construire le modèle de prédiction (elle est représentative). L'ensemble de validation nous servira à valider le modèle entraîné sur les 500 données d'apprentissage, c'est-à-dire il servira à modifier l'apprentissage. Les données de test serviront à vérifier l'apprentissage, il va nous permettre de tester la capacité de généralisation du réseau.

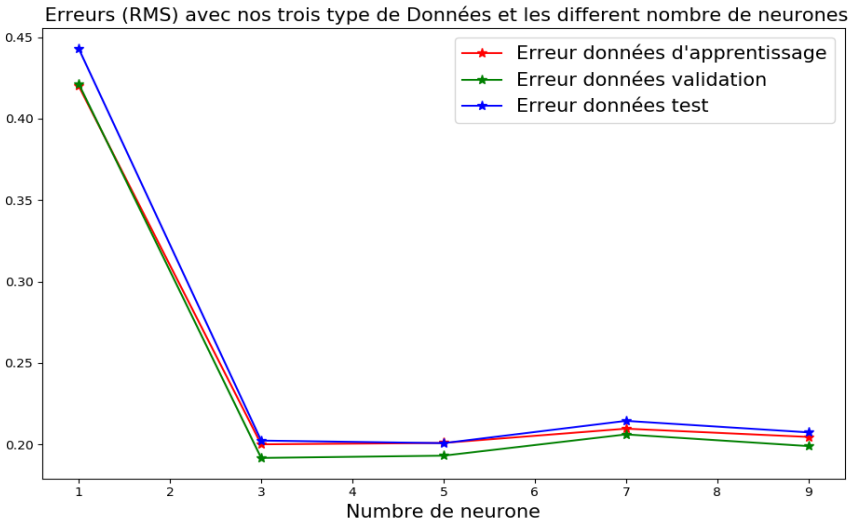
2) Avec les données simulées, nous allons utiliser des données d'apprentissage et les données de validations pour entraîner notre modèle afin de déterminer l'architecture du perceptron multi couche (pmc) optimale en faisant croître le nombre de neurones sur la couche cachée.
Lors de l'apprentissage, les données de validations sont utilisées pour suivre l'évolution de l'erreur quadratique, cela nous permet de stopper l'induction lorsque nous constatons que le

taux d'erreur sur l'ensemble de validation utilisée ne diminue plus lors des itérations ; Nous avons fixé un nombre maximum d'itérations à **1000** pour contrôler l'apprentissage, Un taux d'apprentissage trop rapide peut amener des effets d'instabilités dans le réseau et un taux d'apprentissage trop lent peut amener le réseau à être bloqué dans un minimum local, On conserve les informations relatifs au dernier apprentissage pour en tenir compte dans l'apprentissage courant ainsi on évite les effets d'oscillations ou bien de rester coincé dans un minimum local (on obtient presque les même résultats avec 10000 itération); Aussi, nous avons un paramètres pour l'arrêt sur un seuil minimum de variation des poids d'une valeur par défaut = 10**(-6). Ceci permet de définir les règles d'arrêt du processus d'apprentissage. La figure suivante qui illustre les prédictions avec les données d'apprentissage pour les différent nombre de neurone utilisée.



Nous remarquons que l'architecture à 1 neurone ne parvient pas à prédire correctement la variable à prédire, c'est normal vue que le nuage ressemble beaucoup à une fonction non-linéaire, vue la forme, 1 neurone ne suffit pas pour réaliser cette forme. Les architectures à 3, 5, 7 et 9 neurones se rapprochent mieux de la variable de décision réelle (ils ont la forme du nuage de point). Avec 5 neurones, nous avons une forme qui décrit au mieux ces données, mais comme il faut choisir une architecture optimale, nous opterons pour l'architecture à **3 neurones** car elle fournit de bons résultats et est moins complexe donc moins de calculs à faire.

3) L'erreur root mean square (RMS) est la fonction utilisée par le réseau de neurone pour matérialiser l'apprentissage. Plus le résultat de cette fonction est petit, plus le modèle est performant. L'idée est donc de réajuster les poids du modèle de sorte que l'erreur RMS soit petite.

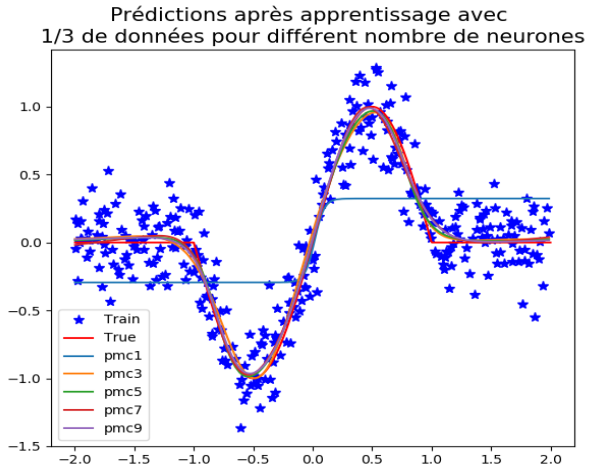


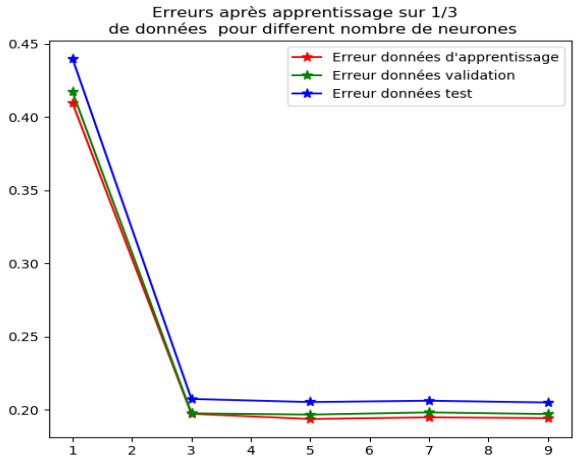
L'objectif de l'apprentissage est de proposer un modèle capable d'avoir de « bonnes performances » sur l'ensemble des exemples possibles. Lorsqu'un apprentissage a été mené, on s'intéresse dès lors au comportement du réseau sur des formes qui n'ont pas servies à l'apprentissage.

Nombre neurones	1	**3**	5	7	9
RMS Apprentissage	0.4199	**0.2001**	0.2009	0.2097	0.2046
RMS Validation	0.4213	**0.1917**	0.1931	0.2061	0.1990
RMS test	0.4429	**0.2024**	0.2008	0.2145	0.2074

On remarque que l'erreur rms est plus petite pour l'architecture 3 pour les données test, apprentissage et validation. Elle est notre architecture optimale.

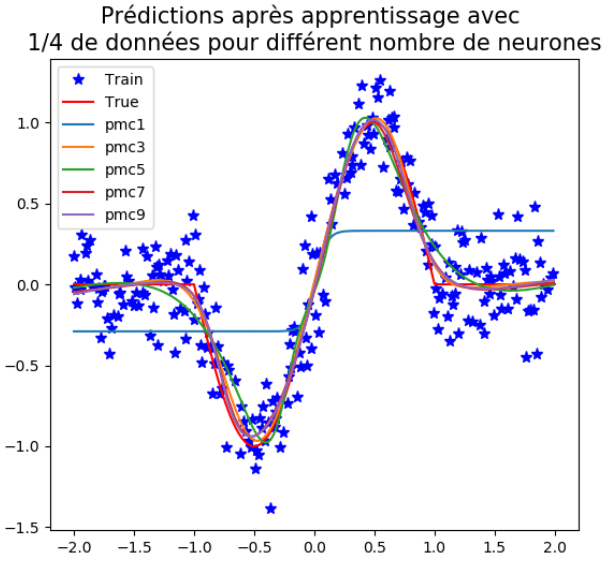
4) Dans cette partie, nous allons au fur à mesure réduire le nombre de données d'apprentissage et de validation tous en maintenant nos données de test. Nous allons refaire l'apprentissage et déterminer le nombre de neurone optimal en les notant en gras dans les tableaux correspondant des erreurs RMS. Les commentaires sera fait après toute les figures.

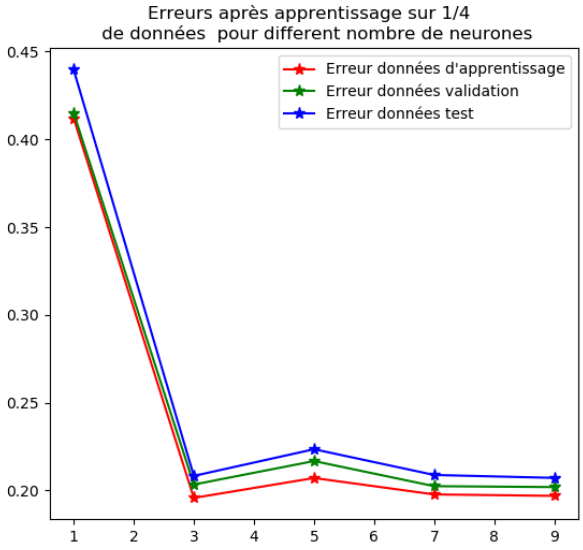




Nombre de neurones	1	3	**5**	7	9
RMS Apprentissage	0.4093	0.1973	**0.1937**	0.1949	0.1943
RMS Validation	0.4170	0.1976	**0.1967**	0.1982	0.1971
RMS test	0.4397	0.2074	**0.2053**	0.2062	0.2050

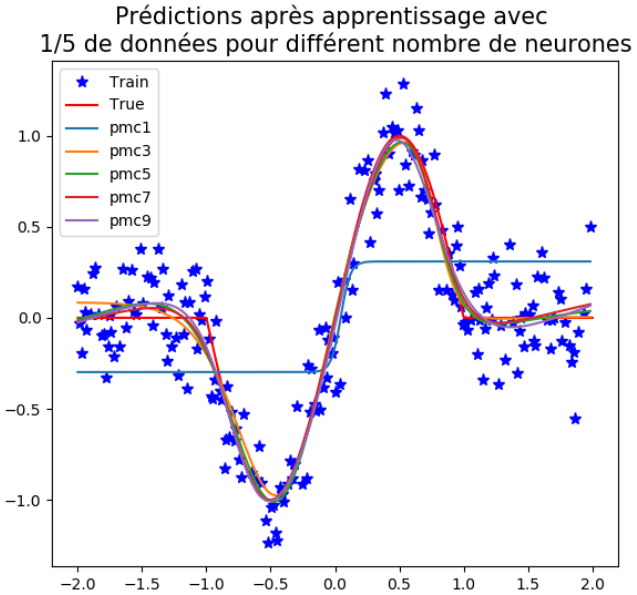
Cette quantité de données semble elle aussi assez représentative, raisonnable pour l'apprentissage.

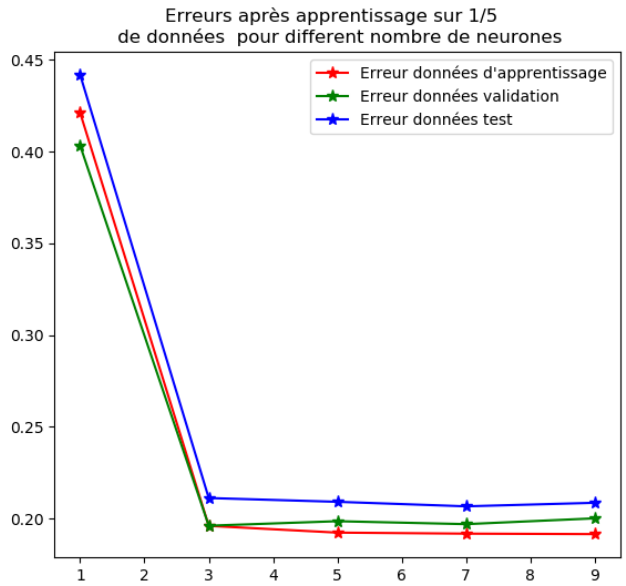




Nombre de neurones	1	**3**	5	7	9
RMS Apprentissage	0.4120	**0.1957**	0.2071	0.1977	0.1969
RMS Validation	0.4150	**0.2033**	0.2167	0.2024	0.2019
RMS test	0.4396	**0.2082**	0.2234	0.2088	0.2071

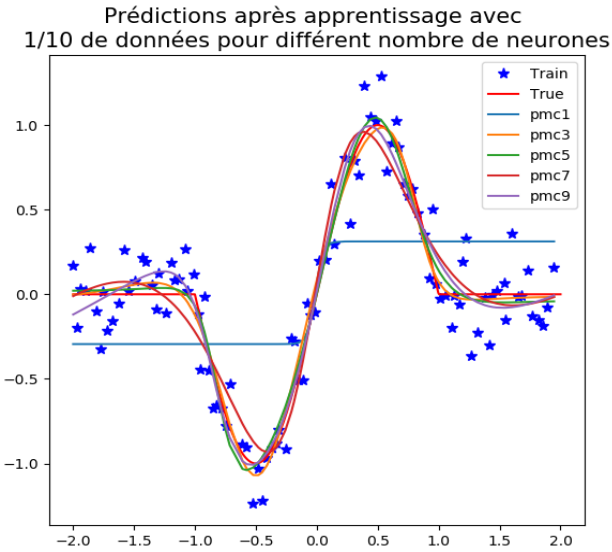
Cette quantité de données semble elle aussi assez représentative, raisonnable pour l'apprentissage.

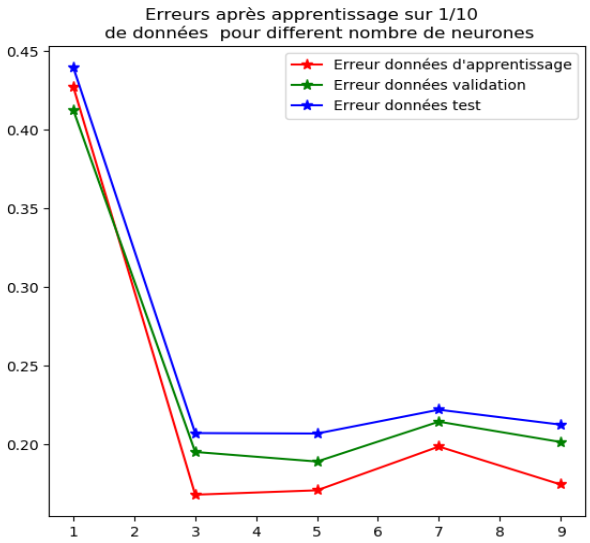




Nombre de neurones	1	**3**	5	7	9
RMS Apprentissage	0.4212	**0.1959**	0.1922	0.1917	0.1914
RMS Validation	0.4033	**0.1960**	0.1984	0.1968	0.2000
RMS test	0.4419	**0.2111**	0.2090	0.2065	0.2085

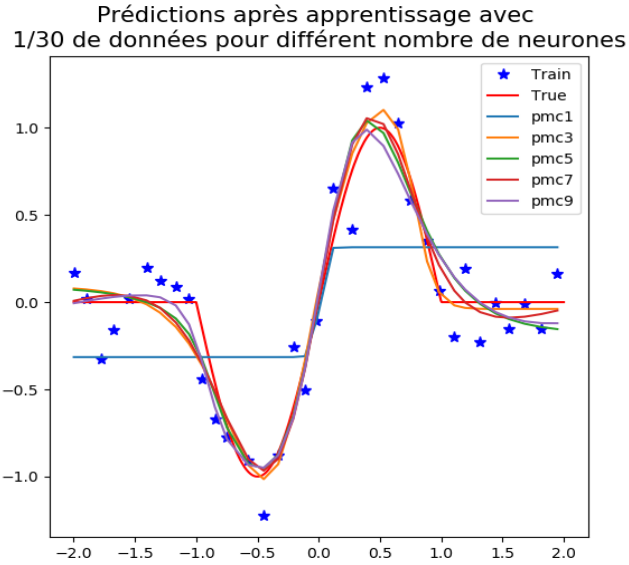
Cette quantité de données semble elle aussi assez représentative, raisonnable pour l'apprentissage.

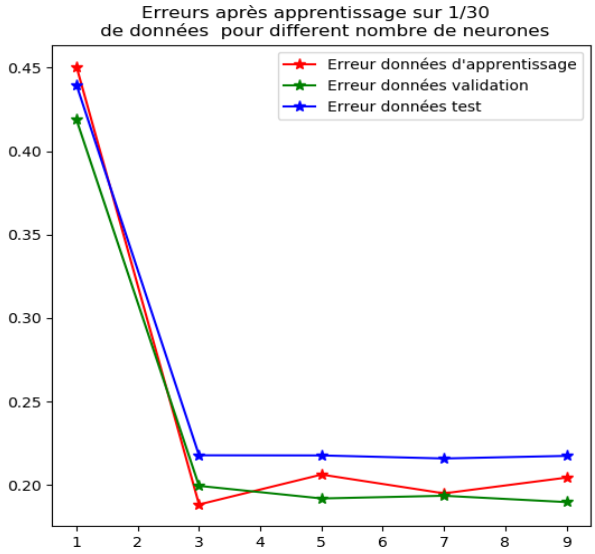




Nombre de neurones	1	3	**5**	7	9
RMS Apprentissage	0.4272	0.1679	**0.1707**	0.1985	0.1744
RMS Validation	0.4125	0.1951	**0.1890**	0.2143	0.2014
RMS test	0.4395	0.2071	**0.2068**	0.2219	0.2125

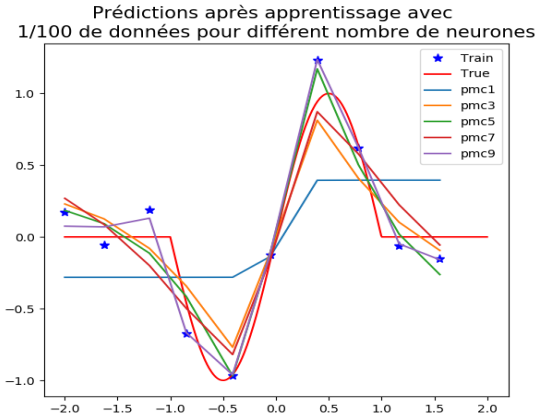
Cette quantité de données semble moins représentative, pas très raisonnable pour l'apprentissage.

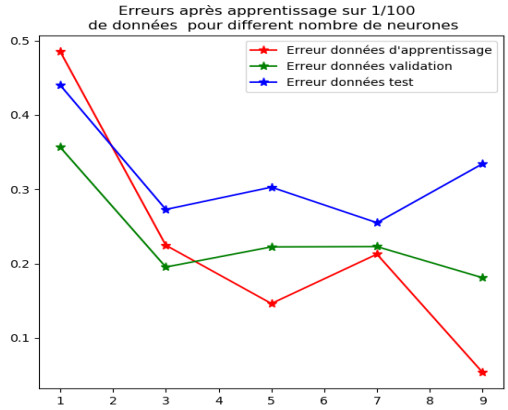




Nombre de neurones	1	3	5	**7**	9
RMS Apprentissage	0.4504	**0.1882**	0.2061	**0.1949**	0.2043
RMS Validation	0.4186	0.1994	0.1918	**0.1934**	0.1897
RMS test	0.4395	0.2177	0.2176	**0.2158**	0.2173

Cette quantité de données semble moins représentative, pas très raisonnable pour l'apprentissage. Les erreurs de test devient plus grand, chose raisonnable du a peu de données d'apprentissage.





Nombre de neurones	1	3	5	7	**9**
RMS Apprentissage	0.4851	0.2246	0.1462	0.2126	**0.0536**
RMS Validation	0.3564	0.1952	0.2224	0.2228	**0.1809**
RMS test	0.4399	0.2727	0.3027	0.2549	**0.3342**

On remarque de gros erreurs pour des entrainements fait avec peut de données. On remarque également que l'architecture avec 9 neurones tend à passer par tous les points et on a de gros erreur pour les prédictions avec les données de test, dans l'apprentissage toute les zones n'ont pas été pris en compte, d'où l'importance du choix de la taille des données. Avec 7 et 5 neurones, la courbe passe par deux points, 3 et 1 passe par un seul point.

CONCLUSION : Il est courant d'atteindre un taux d'erreur égal à zéro en apprentissage lorsque nous mettons beaucoup de neurones dans la couche cachée. On observe se phénomène sur la dernier figure, avec beaucoup de neurone et peut de données. Un des avantages incontestables du perceptron est la possibilité de moduler la puissance du modèle de prédiction en modifiant le nombre de neurones dans la couche cachée. En expérimentent ceci, nous avons remarqué que plus nous ajoutons de neurones, plus le modèle pourra apprendre des formes complexes. Mais en augmentant le nombre de neurones, nous prenons également le risque de trop coller aux données (on remarque avec la dernière figure le cas de 9 neurones pour peu de données). Construire un modèle qui s'attache aux spécificités des données d'apprentissage et non à la « vraie » causalité qui peut exister entre les descripteurs et la variable à prédire n'est pas ce qu'on veut. L'idéal est de construire un modèle qui pourrais être généralisé de tel sort qu'il peut faire de bonne prédictions sur de nouvelles données, il est donc bien de choisir une architecture qui prédit mieux et est simple, c'est-à-dire pas trop de neurone. La taille des données d'apprentissage est aussi un aspect très important dans l'apprentissage, car nous remarquons de grosses erreurs avec peu de données d'apprentissage. Si le nombre de neurones est trop grand, ça peut nous conduire au sur apprentissage car le modèle vas beaucoup se spécialiser. Pour pallier à l'overfitting, on peut diminuer le nombre de neurones du modèle pour que le modèle puis généraliser au mieux. Dans les cas où nous avions faire l'apprentissage avec un nombre raisonnable de données le nombre de neurone idéal est entre 3 ou 5