Master 1 SD, SIME
Master 2 SIME
Apprentissage profond

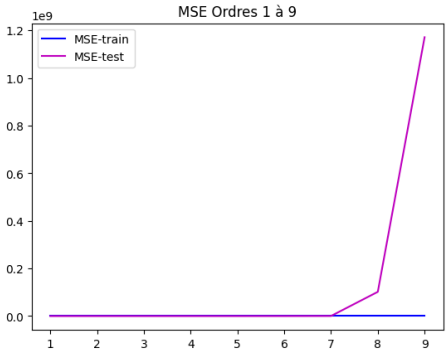
Nom : COELHO FERNANDES
Prénom : Rodrigo César

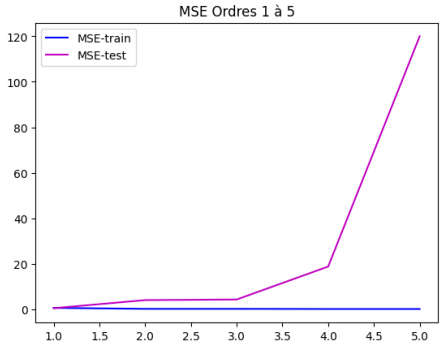
Séance de TP 1
régression, moindres carrés et descente de gradient

1- Modèle polynomiale de régression et moindres carrés

a. Mettre en évidence le compromis biais variance

Nous pouvons constater que la valeur de la MSE pour les ordres 8 et 9 est extrêmement élevée, ce qui impacte l'interprétation du graphique pour les courbes initiales. En limitant l'analyse aux premiers ordres et en se référant aux valeurs dans le tableau, la compréhension devient plus claire.



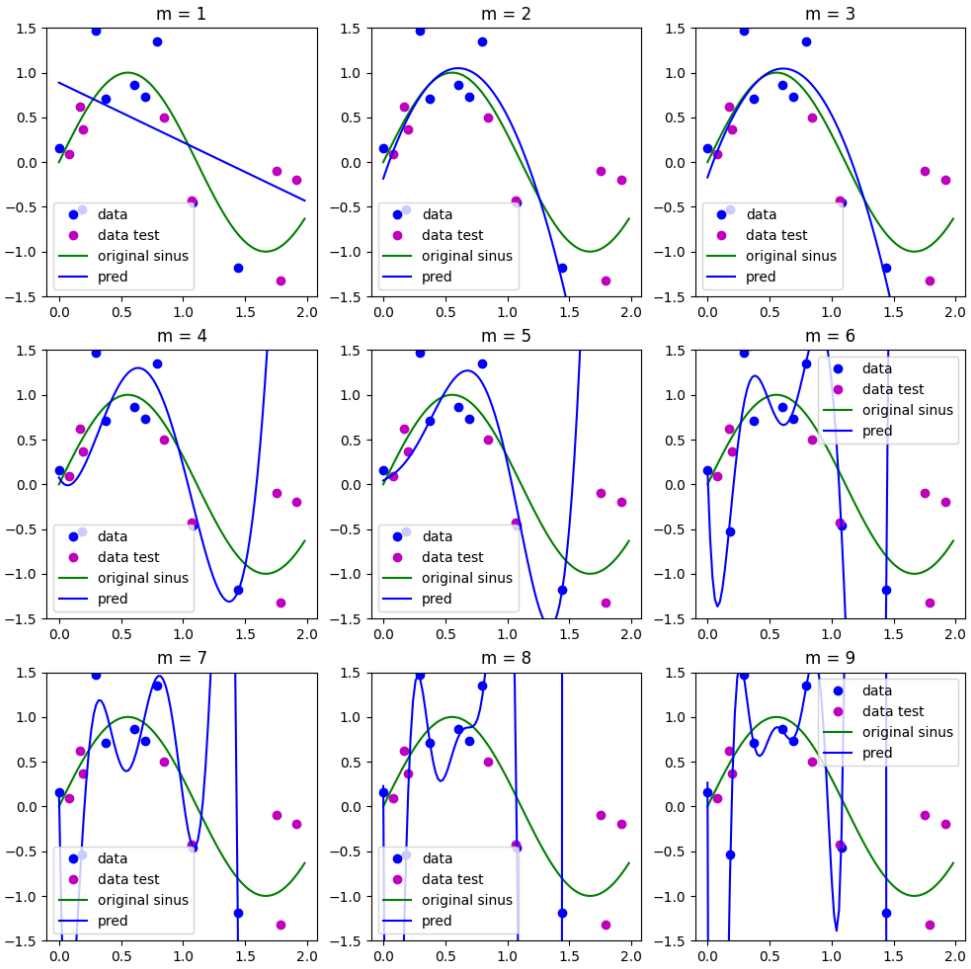


Pour les données de test, la MSE (Variance) augmente fortement à mesure que l'ordre du polynôme croît. En revanche, pour les données d'entraînement, la MSE (Biais) diminue légèrement avec l'augmentation de l'ordre. Le compromis biais-variance consiste à choisir le meilleur modèle, en recherchant de faibles valeurs de MSE, non seulement pour l'entraînement, mais aussi pour le test.

Dans cet exemple, le modèle présentant les plus faibles valeurs pour les deux cas est la droite, c'est-à-dire le polynôme d'ordre 1. Cependant, ce modèle ne représente pas bien la sinusoïde qui a généré les points. Cela indique qu'une quantité plus importante de points est nécessaire pour mieux capturer toutes les nuances de la courbe que nous souhaitons représenter.

b. Mettre en évidence le sur-apprentissage

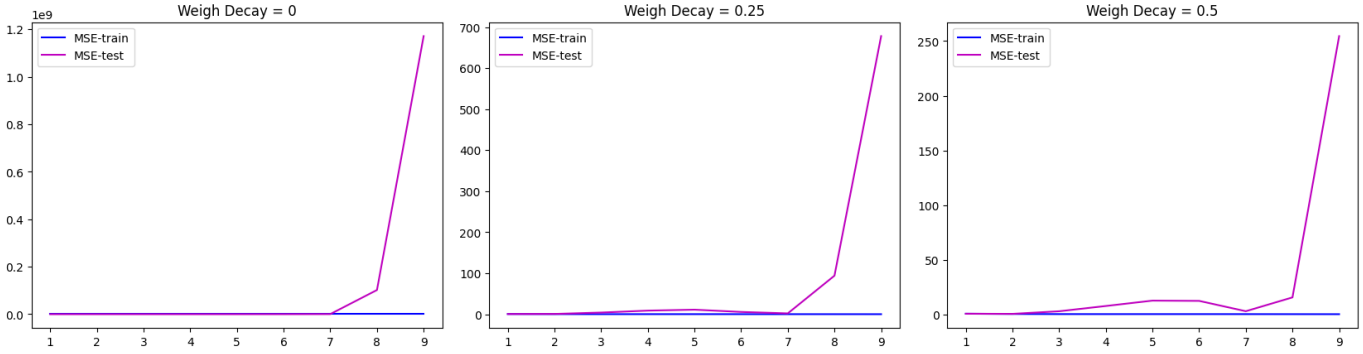
Pour les valeurs d'ordre les plus élevées, nous observons que la courbe s'ajuste de mieux en mieux aux données d'entraînement, au point que, pour le dernier ordre, tous les points sont traversés par la courbe. Cela signifie que le modèle n'a pas appris la fonction souhaitée, mais plutôt les spécificités des données d'entraînement, y compris leur bruit.



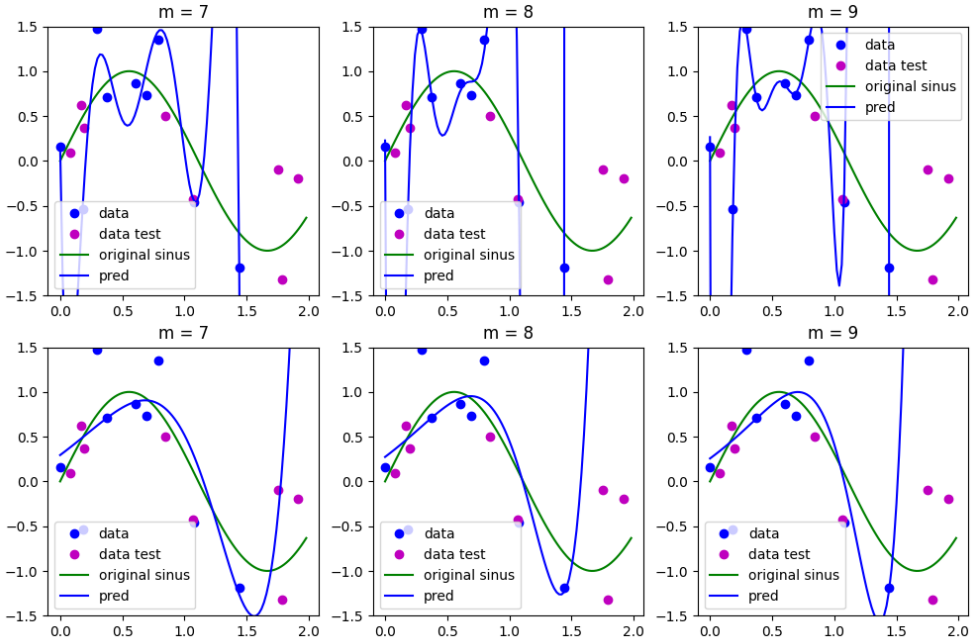
Cela signifie que le modèle ne prédit pas bien d'autres données en dehors de celles utilisées pour l'entraînement. Cela est bien représenté par la valeur élevée de la MSE pour les ordres les plus élevés.

c. Mettre en évidence l'intérêt de la régression ridge en donnant au paramètre weigh_decay différentes valeurs.

La régression Ridge a pour objectif d'éviter le surapprentissage du modèle, généralement en pénalisant les modèles complexes. En activant cette méthode dans la fonction de régression, nous observons que la MSE présente des valeurs considérablement plus faibles, et les modèles varient moins avec l'augmentation de l'ordre polynômial.



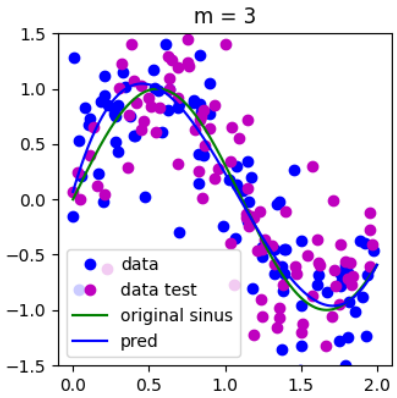
De plus, les signes de surapprentissage pour les ordres élevés sont moins évidents : la première image montre les ordres élevés sans weight_decay, tandis que la seconde montre les mêmes ordres avec un weight_decay de 0,25.

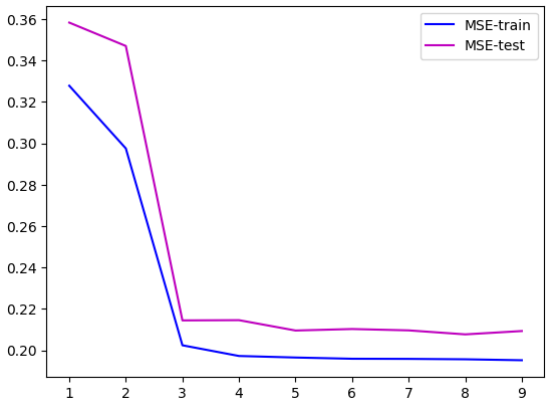


Il est important de souligner qu'il n'existe pas de méthode définie pour trouver la valeur idéale du *weight_decay*.

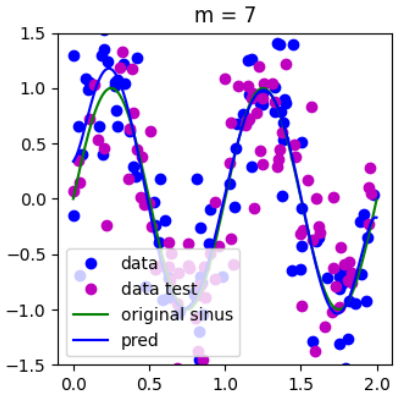
d. Évaluer le comportement de l'algorithme lorsque le nombre de mesures augmente N=100

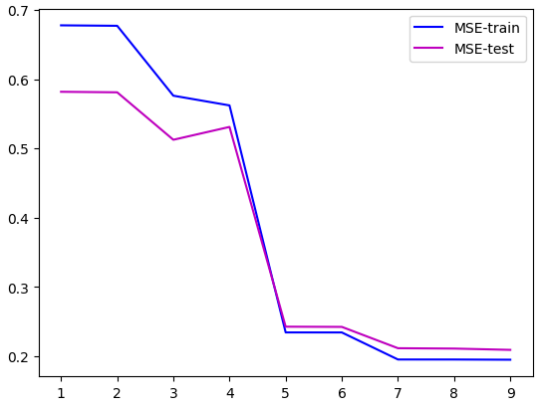
En augmentant le nombre de points dans les ensembles d'entraînement et de test, nous observons que le modèle représente mieux la fonction de manière générale, sans signe de surapprentissage (overfitting) dans les 10 premiers ordres. Nous constatons que la MSE présente un point de coude au polynôme d'ordre 3, ce qui montre que cet ordre représente bien la fonction et que les ordres suivants n'apportent que peu d'améliorations.





Avec un nombre suffisant de points d'entraînement, il est même possible d'augmenter la complexité de la fonction tout en obtenant des résultats satisfaisants. Pour une fréquence de sinus égale à 1, nous voyons une série de points de coude qui se termine à l'ordre 7. Cela signifie que l'augmentation de la complexité de la fonction nécessite une augmentation de l'ordre polynômial.





2- Modèle polynomiale de régression et descente de gradient

a. *Dans la fonction train_loop(), à quel moment le graphe de calcul est-il construit ? Quel est son rôle ? (Référez-vous au cours sur la différenciation automatique.)*

Les paramètres sont initialisés de manière aléatoire, et la fonction de perte (Loss) est calculée à partir de cette initialisation. Ensuite, lors de la rétropropagation, les paramètres de ce modèle sont mis à jour. Cela se fait dans une boucle où chaque itération est considérée comme une époque.

```
for batch, (data, labels) in enumerate(dataloader):
        # Prédiction et calcul de la perte
        predictions = model(data.float())
        loss = loss_fn(predictions.float(), labels.float())
        epoch_loss += loss.item()

        # Rétropropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

La mise à jour des paramètres est donnée par le gradient de la Loss, c'est-à-dire que nous corrigeons les paramètres avec une intensité proportionnelle au learning rate dans la direction opposée au gradient de la perte.

b. *Expliquez le rôle d'un optimizer ainsi que la fonction de chacune des commandes suivantes :*
- optimizer.zero_grad()
- optimizer.step()

L'optimiseur cherche à mettre à jour les paramètres du modèle à partir du gradient de la Loss. La fonction zero_grad() met tous les gradients à zéro afin qu'ils puissent être recalculés à l'étape suivante. Cela est essentiel pour garantir que les gradients d'un batch précédent n'influencent pas le batch actuel.

La méthode optimizer.step() mettra à jour les poids et les biais du modèle avec les gradients calculés. Un pas sera effectué dans la direction qui minimise la Loss.

c. *Expliquez le rôle d'une loss ainsi que la fonction de chacune des commandes suivantes :*

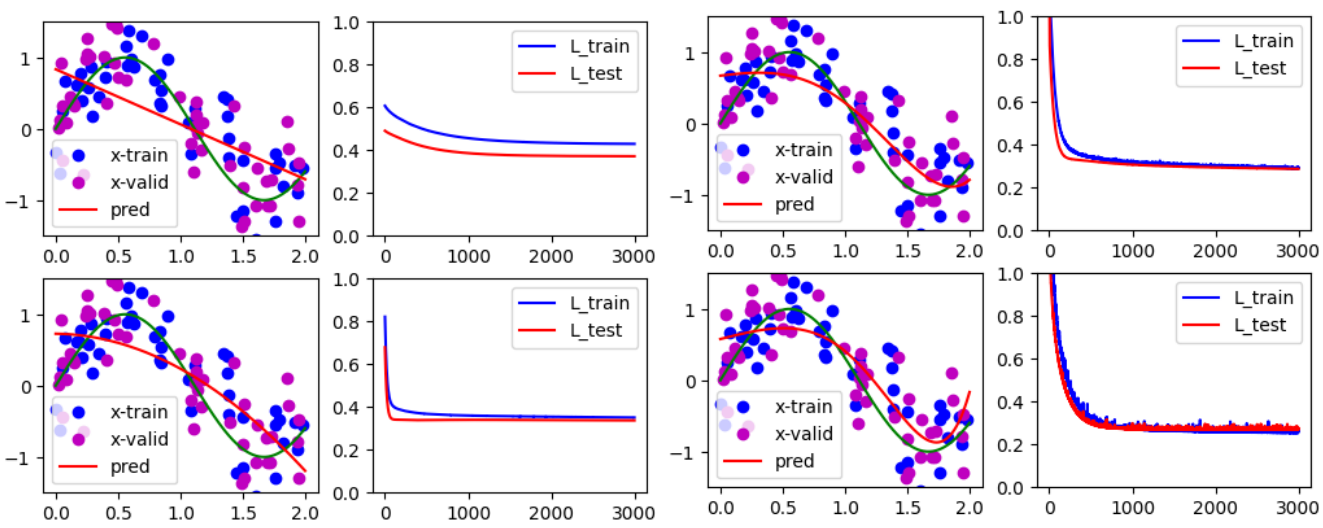
- *loss = loss_fn(pred.float(), y.float())*
- *loss.backward()*

La Loss représente l'erreur de la prédiction, c'est la fonction que nous cherchons à minimiser. La fonction loss_fn() calcule la valeur de l'erreur entre la prédiction du modèle et la valeur réelle. Ensuite, la fonction backward() calcule le gradient de cette erreur pour qu'il soit utilisé lors de la mise à jour des paramètres.

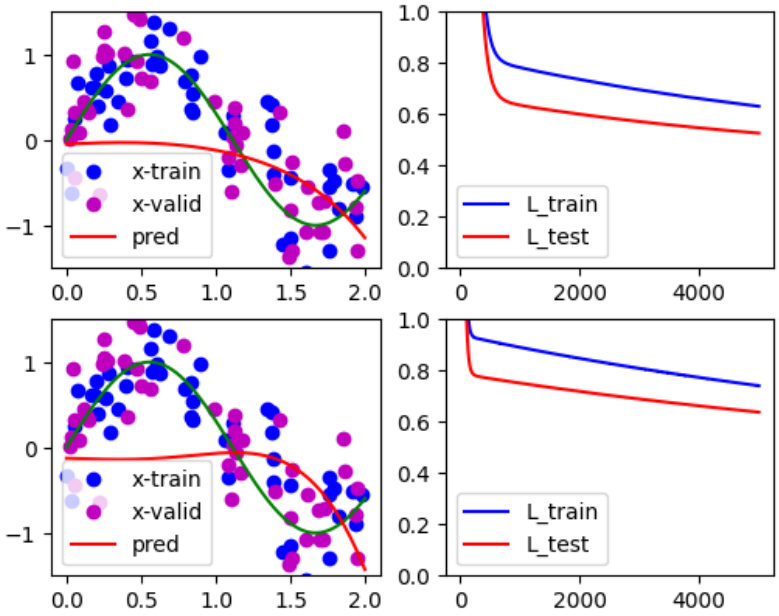
d. *Déterminer les propriétés de convergence de l'algorithme de descente de gradient, pour les différents ordres de la régression envisagée. Déterminer l'influence du learning rate et du nombre d'époques :*
learning_rate = 1e-3, epochs = 5000

Au fil des époques, la Loss pour tous les ordres de régression tend à diminuer et à converger vers une valeur minimale, ce qui est un signe de la validité de la méthode. Cependant, les différents ordres convergent de manière distincte.

On observe que les ordres plus élevés présentent une plus grande instabilité, avec une trajectoire de convergence bruitée. Cela peut indiquer un learning rate trop élevé, dépassant fréquemment le minimum.

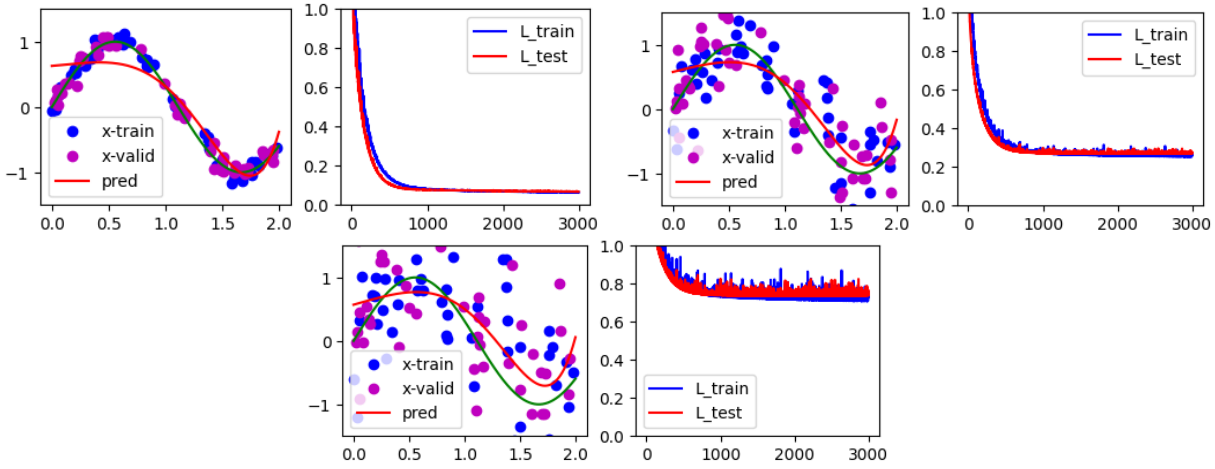


En réduisant le *learning rate* à 1e−5, les ordres plus élevés commencent à diminuer de manière plus cohérente. Cependant, même après 5000 époques, aucun des ordres n'a pu atteindre la convergence ni produire une fonction qui représente correctement les données.



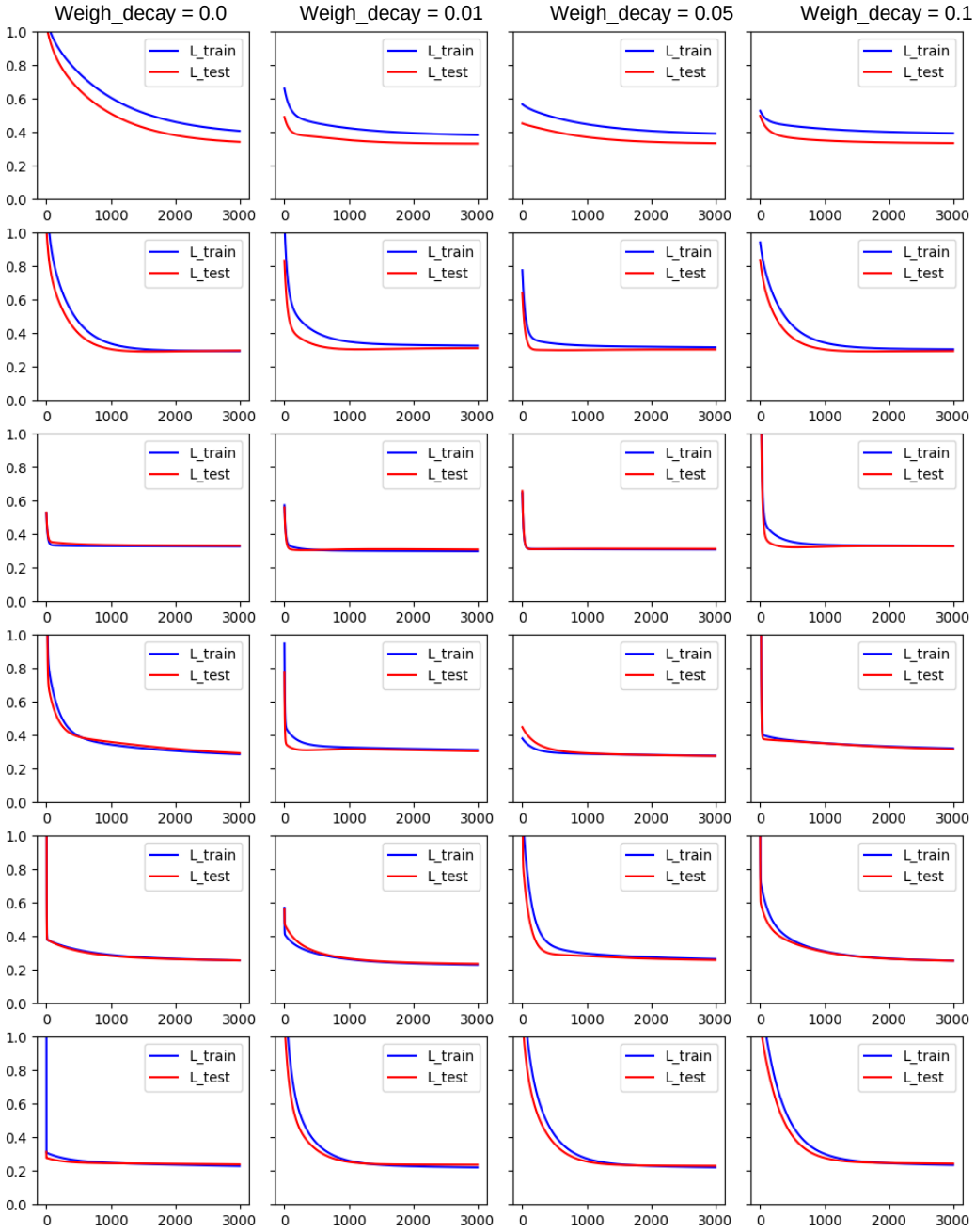
e. Examiner la relation entre le critère MSE et l'amplitude du bruit

L'augmentation de l'amplitude du bruit provoque une augmentation de la Loss, mais n'a pas un grand impact sur la fonction générée pour chaque ordre. Cela s'explique par le fait que la quantité de points est déjà suffisante pour bien représenter la fonction et permettre le bon fonctionnement du modèle. Ainsi, l'augmentation du bruit fait simplement que les points sont plus dispersés, ce qui entraîne une plus grande différence entre les valeurs prévues et réelles.



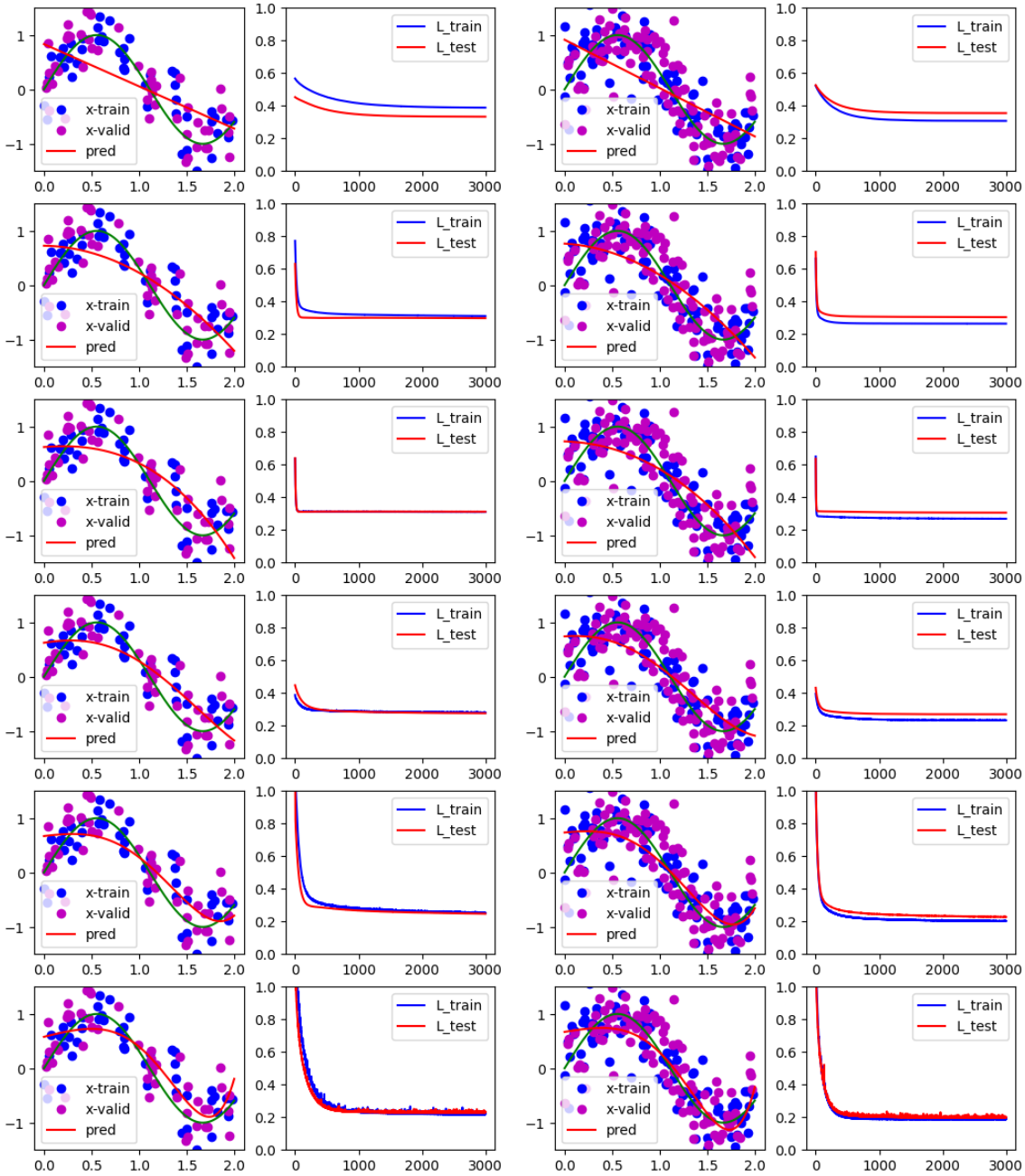
f. *Mettre en évidence l'intérêt du weight_decay en lui donnant différentes valeurs : weight_decay = 0 #, 0.01, 0.05, 0.1*

On peut constater que l'augmentation du weigh_decay ne semble pas avoir d'impact sur la fonction finale de chaque ordre, car tous les tests convergent vers la même valeur de Loss indépendamment du weigh_decay. Cependant, on observe une variation dans la vitesse à laquelle le seuil de l'erreur minimale est atteint. L'augmentation du weigh_decay tend à ralentir la vitesse de diminution de la Loss, notamment pour les ordres plus élevés, ce qui peut être interprété comme une forme de pénalisation pour ces modèles plus complexes.



g. *Evaluer le comportement de l'algorithme lorsque le nombre de mesures augmente N=100*

De manière générale, la *Loss* de test a été légèrement plus élevée avec l'augmentation de N. Cela peut être dû au fait que nous avons ainsi un nombre suffisant de points pour démontrer que le modèle basé sur la descente de gradient ne s'est pas montré adapté pour représenter cette fonction. Dans tous les cas, même après avoir passé toutes les époques et atteint un plateau constant, la fonction de régression ne représente pas bien la fonction originale.



3- Comparer les solutions obtenues avec les moindres carrés et avec la descente de gradient

La méthode des moindres carrés s'est avérée beaucoup plus efficace pour définir le modèle qui représentait la fonction originale, atteignant un MSE faible et donnant de bons résultats pour des modèles peu complexes (de faible ordre).

D'un autre côté, le modèle de descente de gradient a réussi à se rapprocher de la fonction pour un ordre de d=9, mais les autres ordres ont donné des résultats extrêmement insatisfaisants. Cela peut être dû à une sélection non idéale des hyperparamètres ou à la nécessité d'augmenter le nombre de couches cachées pour garantir un réseau neuronal plus robuste.