Régularisation

Bassam Kurdy Ph.D
<bassam.kurdy@apinum.fr>

Veille individuelle **30 min**

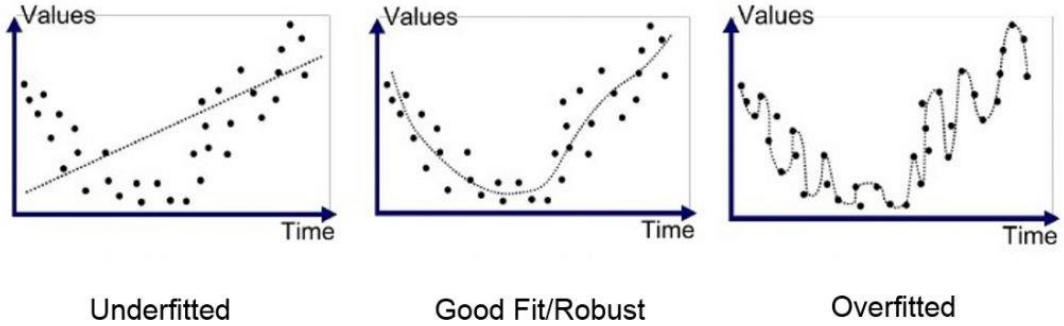
Sur-apprentissage

- Qu’est ce que le sur-apprentissage ?
- Comment détecter le sur-apprentissage ?

Sur-apprentissage (overfitting)

Le **surapprentissage** (ou overfitting) est un phénomène qui se produit lorsque

- la **performance** d'un modèle est très élevée sur les données d'entraînement, mais très faible sur les données de test.
- le modèle est trop **complexe** par rapport aux données d'entraînement.
- le modèle a trop de **variables** ou de degrés de liberté par rapport au nombre de données d'entraînement,
- les données d'entraînement sont **bruyantes** ou présentent des anomalies.
- les données d'entraînement et de test ne sont pas **représentatives** de la population sous-jacente.



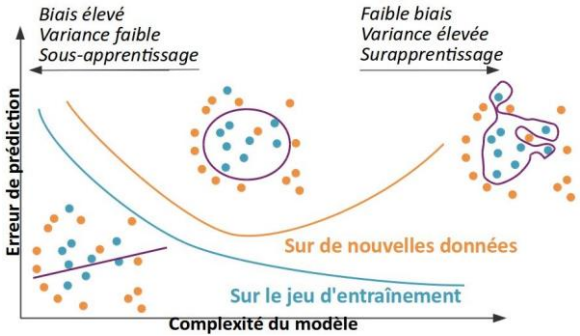
Sur-apprentissage (overfitting)

Biais : erreur systèmatique qui se produit lorsque le modèle est incapable de capturer les relations sous-jacentes entre les variables.

- Lorsque le modèle est trop simple par rapport aux données d'entraînement, il présente un **biais élevé.**.
- le modèle est en **sous-apprentissage**.

Variance : erreur aléatoire qui se produit en raison de la sensibilité du modèle aux variations aléatoires dans les données d'entraînement.

- lorsque le modèle est trop **complexe** par rapport aux données d'entraînement, il présente une **variance élevée** et est très sensible aux variations aléatoires dans les données d'entraînement.
- le modèle est **sur-apprentissage**.



	Underfitting	Just right	Overfitting
Symptômes	• Erreur d'entraînement élevé • Erreur d'entrainement proche de l'erreur de test • Biais élevé	• Erreur d'entrainement légèrement inférieure à l'erreur de test	• Erreur d'entrainement très faible • Erreur d'entrainement beaucoup plus faible que l'erreur de test • Variance élevée
Illustration dans le cas de la régression			
Illustration dans le cas de la classification			
Illustration dans le cas de l'apprentissage profond			
Remèdes possibles	• Complexifier le modèle • Ajouter plus de variables • Laisser l'entraînement pendant plus de temps		• Effectuer une régularisation • Avoir plus de données

Veille individuelle **30 min**

La régularisation

- ❏ Qu’est ce que la régularisation ?
- ❏ Quel est l’intérêt de la régularisation ?
- ❏ Quels sont les types de régularisation ?
- ❏ Comment implémenter la régularisation avec scikit-learn ?

La régularisation

- En machine learning, la **régularisation** est une technique qui vise à réduire l'overfitting (surapprentissage) d'un modèle en ajoutant une **pénalité** à la fonction de coût qui mesure la qualité de la prédiction.
- Elle permet de trouver un compromis entre la complexité du modèle et sa capacité de généralisation en limitant la magnitude des poids des paramètres du modèle.
- La régularisation est particulièrement utile lorsque les données d'entraînement sont limitées et que le modèle est complexe.
- Il existe différentes méthodes de régularisation :
  - La régularisation **L1** ajoute une pénalité égale à la somme de la valeur absolue des poids
  - la régularisation **L2** ajoute une pénalité égale à la somme des carrés des poids.
- Ces pénalités sont ajoutées à la fonction de coût lors de l'entraînement du modèle.

La régularisation

La formule mathématique pour la régularisation LASSO (L1) est : **J(w) = MSE + alpha * ||w||1**

- J(w) est la fonction de coût régularisée avec la pénalité LASSO
- w est le vecteur des coefficients de régression à estimer
- ||w||1 est la norme L1 du vecteur w, qui est la somme des valeurs absolues de tous les coefficients de régression
- alpha est le paramètre de régularisation, qui contrôle la force de la pénalité L1

```
from sklearn.linear_model import Lasso
# Créer un objet Lasso avec un paramètre alpha de 0,1
lasso = Lasso(alpha=0.1)
# Entraîner le modèle sur les données d'entraînement
# X_train et y_train
lasso.fit(X_train, y_train)
# Prédire les valeurs pour les données de test X_test
y_pred = lasso.predict(X_test)
```

La régularisation

La formule mathématique pour la régularisation Ridge est : **J(w) = MSE + alpha * ||w||2^2**

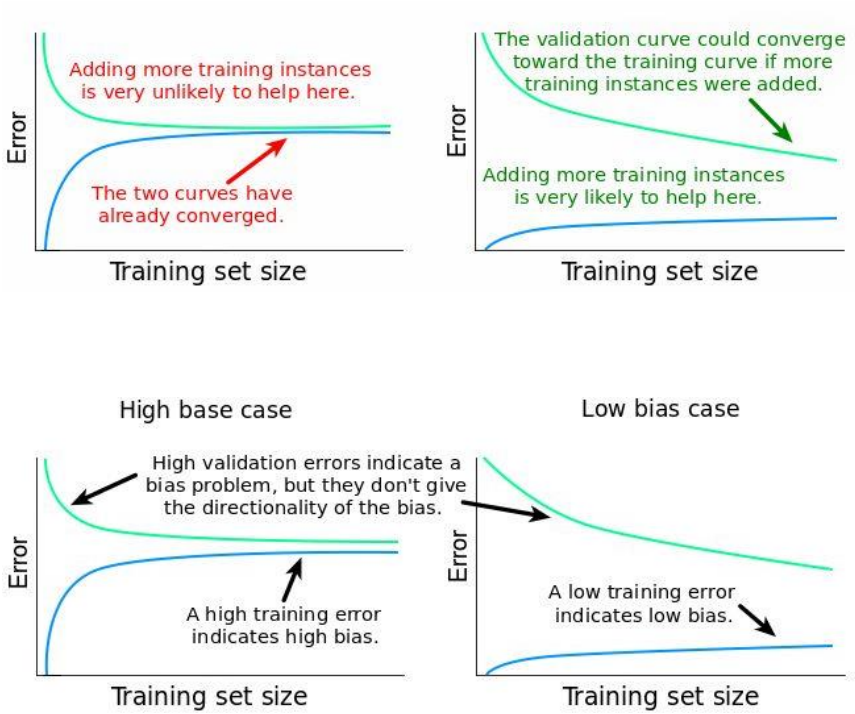
- J(w) est la fonction de coût régularisée avec la pénalité Ridge
- w est le vecteur des coefficients de régression à estimer
- ||w||2 est la norme L2 du vecteur w, qui est la racine carrée de la somme des carrés de tous les coefficients de régression
- alpha est le paramètre de régularisation, qui contrôle la force de la pénalité L2

```
from sklearn.linear_model import Ridge
# Créer un objet Ridge avec un paramètre alpha de 0,1
ridge = Ridge(alpha=0.1)
# Entraîner le modèle sur les données d'entraînement
# X_train et y_train
ridge.fit(X_train, y_train)
# Prédire les valeurs pour les données de test X_test
y_pred = ridge.predict(X_test)
```

Régularisation L1 (Lasso)	Régularisation L2 (Ridge)	
Type de pénalité	Somme de la valeur absolue des poids	Somme des carrés des poids
Impact sur les poids	Met à zéro les poids des variables moins importantes	Réduit la magnitude de tous les poids
Nombre de variables	Utile pour un grand nombre de variables, certaines étant moins importantes	Utile pour un grand nombre de variables ayant des effets similaires sur la variable à prédire
Sélection de variables	Sélectionne automatiquement les variables les plus importantes	Ne sélectionne pas automatiquement les variables
Performance	Peut être plus performante lorsque peu de variables sont importantes	Peut être plus performante lorsque de nombreuses variables sont importantes
Problèmes numériques	Peut rencontrer des problèmes numériques en présence de variables fortement corrélées	Moins sensible aux problèmes numériques en présence de variables fortement corrélées

Courbe d’apprentissage

- La **courbe d'apprentissage** est un outil couramment utilisé en machine learning pour
  - **évaluer la performance** d'un modèle en fonction de la quantité de données d'entraînement utilisée.
  - **identifier les problèmes de sous-apprentissage ou de surapprentissage**
  - **déterminer la taille optimale de l'ensemble d'entraînement** pour obtenir une bonne performance de prédiction.
- La courbe d'apprentissage affiche l'évolution de l'erreur de prédiction du modèle en fonction de la taille de l'ensemble d'entraînement.



Workshop

Vous avez été chargé d'évaluer deux méthodes de régularisation (Ridge et Lasso) pour prédire le prix des maisons dans le dataset California Housing de scikit-learn. Le but de cet exercice est d'entraîner des modèles Ridge et Lasso avec différents paramètres alpha, et de comparer leur performance en utilisant la courbe d'apprentissage.

1. Chargez le dataset California Housing à partir de scikit-learn.
2. Divisez le dataset en ensembles d'entraînement et de validation, en utilisant une proportion de 80% pour l'ensemble d'entraînement.
3. Créez des modèles Ridge et Lasso avec les paramètres alpha suivants : 0.1, 1, 10.
4. Pour chaque modèle, entraînez-le sur l'ensemble d'entraînement, puis évaluez son erreur de prédiction sur l'ensemble de validation. Affichez le score MSE (Mean Squared Error) pour chaque modèle.
5. Tracez la courbe d'apprentissage pour le modèle Ridge avec alpha = 1 utilisez la fonction learning_curve.
6. Commentez les résultats obtenus.
    a. Quel est le modèle qui donne les meilleurs résultats ?
    b. La courbe d'apprentissage montre-t-elle un problème de surapprentissage ou de sous-apprentissage ?
    c. À partir de quelle taille de l'ensemble d'entraînement peut-on considérer que le modèle a atteint sa limite de performance ?