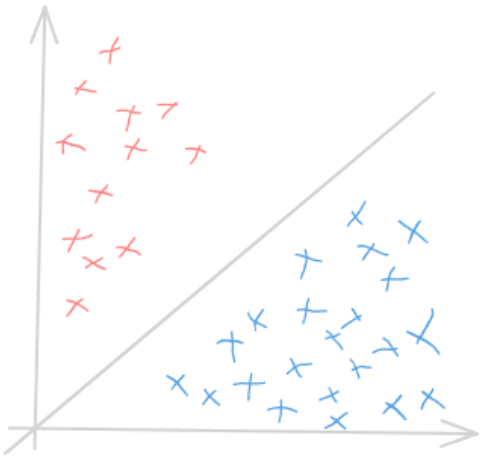
Modèle Linéaire

#cours #linear

Introduction

Un **modèle linéaire** en machine learning est une **méthode** simple et efficace pour **prédire** des valeurs numériques ou **classifier** des données. Il repose sur l'hypothèse que la relation entre les entrées et la sortie peut être représentée par une ligne droite (en deux dimensions) ou un hyperplan (en dimensions supérieures).



Concept De Base

Un modèle linéaire est défini par une formule qui prend plusieurs paramètres en entrée et les transforme en une sortie prédictive. La formule générale d'un modèle linéaire est

$$y = w0 + w1x1 + w2x2 + \cdots + wnxn$$

avec pour paramètres :

- **Les entrées (x_i) :** Les entrées, ou caractéristiques (features), sont les variables indépendantes utilisées pour faire des prédictions. Chaque entrée x_i représente une caractéristique particulière d'une observation.
  - **Exemple :** Dans un modèle prédisant le prix des maisons, les entrées pourraient inclure la surface de la maison, le nombre de chambres, l'année de construction, etc.
- **La sortie(y) :** La sortie est la variable dépendante que le modèle essaie de prédire.

- **Les poids (w_i) :** Les poids sont les coefficients qui multiplient chaque entrée dans l'équation du modèle linéaire. Ils déterminent l'importance de chaque caractéristique pour la prédiction finale.
    - **Exemple :** Si une maison a une surface de 70 m² et le poids pour la surface est 2, alors la contribution de la surface à la prédiction du prix serait $70 \times 2 = 140$
- **Le biais (w_0) :** Le biais est une constante ajoutée au modèle pour ajuster la prédiction indépendamment des entrées. Il permet de décaler la ligne de régression vers le haut ou vers le bas.
    - **Exemple :** Si le biais est 30, cela signifie que même si toutes les entrées étaient égales à zéro, la prédiction de base commencerait à 30.

Les Hyperparamètres

Les hyperparamètres sont des paramètres définis avant le processus d'entraînement d'un modèle. Contrairement aux paramètres du modèle (comme les poids et les biais) qui sont appris à partir des données d'entraînement, les hyperparamètres sont configurés pour contrôler le comportement de l'algorithme d'apprentissage.

Taux D'Apprentissage (Learning Rate)

Le taux d'apprentissage détermine la taille des pas effectués pour mettre à jour les poids du modèle lors de l'optimisation.

- **Valeur élevée :** Converge plus rapidement mais peut dépasser le minimum de la fonction de coût.
- **Valeur faible :** Converge plus lentement mais est plus précise et stable.

Nombre D'Itérations (Epoch)

Le nombre d'itérations est le nombre de fois que l'algorithme parcourt l'ensemble de données d'entraînement.

- **Plus d'itérations :** Peut améliorer l'entraînement mais peut entraîner un surapprentissage si trop nombreuses.
- **Moins d'itérations:** Peut entraîner un sous-apprentissage si insuffisantes.

Taille Du Batch (Batch)

La taille du batch est le nombre d'exemples de données utilisés pour calculer l'erreur et mettre à jour les poids à chaque itération.

- **Batch size = 1 :** Apprentissage en ligne (ou stochastique), plus de bruit mais peut trouver de meilleurs minima.
- **Batch size = totalité des données :** Apprentissage par batch complet, moins de bruit mais peut être moins efficace.
- **Batch size intermédiaire:**** Compromis entre les deux, souvent utilisé dans la pratique.

Nombre De Neurones

Le nombre de neurones par couche et le nombre de couches dans un réseau de neurones sont des hyperparamètres critiques.

- **Plus de neurones/couches :** Permet de modéliser des relations plus complexes mais peut entraîner un surapprentissage et augmenter le temps de calcul.
- **Moins de neurones/couches :** Plus simple et moins susceptible de surapprentissage, mais peut manquer de capacité pour modéliser des relations complexes.

Fonction D'Activation

Les fonctions d'activation sont essentielles dans les réseaux de neurones car elles introduisent de la non linéarité dans le modèle, permettant ainsi de capturer des relations complexes entre les entrées et les sorties. Cela ce traduit par la transformation des sommes pondérées des entrées pour produire la sortie. Dans les modèles linéaires simples, il n'y a pas toujours une fonction d'activation explicite.

Fonction D'Activation Linéaire (Identité)

La fonction d'activation linéaire est simplement l'identité, ce qui signifie que la sortie est égale à l'entrée. Elle est utilisée dans les modèles linéaires classiques.

$$f(x) = x$$

Caractéristiques :

- Simplicité, adaptée aux problèmes de régression.
- Ne capture pas la non-linéarité.

Fonction Sigmoïde (Sigmoid)

La fonction sigmoïde transforme les valeurs en une sortie comprise entre 0 et 1. Elle est souvent utilisée pour les modèles de classification binaire.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Caractéristiques :

- Sortie comprise entre 0 et 1.
- Utilisée pour la classification binaire.
- Problème de saturation : les gradients deviennent très faibles pour les valeurs extrêmes de xx, ce qui ralentit l'apprentissage.

Fonction Tangente Hyperbolique (Tanh)

La fonction tanh est similaire à la sigmoïde mais elle transforme les valeurs en une sortie comprise entre -1 et 1.

$$f(x) = tanh(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}}$$

Caractéristiques :

- Sortie comprise entre -1 et 1.
- Centrée autour de 0, ce qui peut aider à la convergence plus rapide que la sigmoïde.
- Problème de saturation similaire à la sigmoïde.

Exemple d'application : Modèles de classification où les sorties peuvent être négatives ou positives.

Unité Linéaire Rectifiée (ReLU)

La fonction ReLU est largement utilisée dans les réseaux de neurones profonds. Elle renvoie 0 pour les entrées négatives et la valeur d'entrée elle-même pour les entrées positives.

$$f(x) = max(0, x)$$

Caractéristiques :

- Sortie non bornée pour les valeurs positives et 0 pour les valeurs négatives.
- Introduction de la non-linéarité sans problème de saturation pour les valeurs positives.
- Peut entraîner des neurones "morts" (sortie toujours 0) si beaucoup de valeurs négatives.

Exemple d'application : Réseaux de neurones convolutifs pour la reconnaissance d'images.

Fonctions De Coût (Loss)

Les fonctions de coût, ou fonctions de perte, mesurent la performance d'un modèle de machine learning en quantifiant l'erreur entre les prédictions du modèle et les valeurs réelles des données. Dans les modèles linéaires, ces fonctions jouent un rôle crucial pour guider l'optimisation des paramètres du modèle.

Erreur Quadratique Moyenne (MSE - Mean Squared Error)

L'Erreur Quadratique Moyenne est la fonction de coût la plus couramment utilisée pour la régression. Elle mesure la moyenne des carrés des différences entre les valeurs prédites et les valeurs réelles.

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (y'_i - y_i)^2$$

- y'_i est la valeur prédite par le modèle.
- y_i est la valeur réelle.
- m est le nombre total d'observations.

Avantages :

- Punir davantage les grands écarts grâce au carré de la différence, ce qui peut aider à obtenir un modèle plus précis.

Inconvénients :

- Peut être excessivement affectée par les valeurs aberrantes (outliers) en raison du carré des différences.

Erreur Absolue Moyenne (MAE - Mean Absolute Error)

L'Erreur Absolue Moyenne mesure la moyenne des valeurs absolues des différences entre les prédictions et les valeurs réelles. C'est une alternative au MSE.

$$MAE = 1m \sum i = 1m \mid y^i - yi \mid$$

- y'_i est la valeur prédite par le modèle.
- y_i est la valeur réelle.
- m est le nombre total d'observations.

Avantages :

- Moins sensible aux valeurs aberrantes par rapport au MSE.

Inconvénients :

- Moins lisse que le MSE, ce qui peut rendre l'optimisation des paramètres plus difficile car la dérivée de la valeur absolue n'est pas définie à zéro.

Conclusion>

Le modèle linéaire est une méthode fondamentale en machine learning pour faire des prédictions. En comprenant les concepts de biais, poids, fonction de coût et gradient descent, nous pouvons ajuster notre modèle pour minimiser les erreurs et améliorer les prédictions.