Projet : Introduction au Machine Learning

2.1 Implémentation LDA et régression logistique

1. DESCENTE DE GRADIANT

Ce code implémente l'algorithme de descente de gradient pour la régression linéaire simple. Il utilise une fonction normalize pour normaliser les données, la fonction mean\_squared\_error pour calculer la moyenne des erreurs carrées, la fonction update\_t\_b pour mettre à jour les paramètres theta et b en utilisant la dérivée, la fonction gradient\_descent pour effectuer la descente de gradient en utilisant les fonctions précédentes, et la fonction main pour exécuter le code. Le graphe final montre la convergence du coût (moyenne des erreurs carrées) au fil des itérations.

Le paramètre rate détermine la vitesse d'apprentissage du modèle, c'est-à-dire la vitesse à laquelle les paramètres theta et b sont mis à jour à chaque itération. Si rate est trop élevé, le modèle peut ne pas converger ou osciller autour de la valeur optimale. Si rate est trop faible, le modèle peut prendre beaucoup de temps pour converger.

Ainsi, la modification du paramètre rate aura un impact sur la forme du graphe final. Si rate est augmenté, le coût peut augmenter et fluctuer au fil des itérations. Si rate est diminué, le coût peut converger plus lentement mais de manière plus stable. Il est donc important de choisir un taux approprié pour obtenir des résultats optimaux.

Le paramètre threshold détermine la précision avec laquelle nous voulons que le modèle converge vers la solution optimale. Il représente la différence minimale entre les coûts précédents et actuels à laquelle le processus d'apprentissage doit s'arrêter. Si threshold est trop faible, le processus d'apprentissage peut s'arrêter trop tôt, alors que si threshold est trop élevé, le processus d'apprentissage peut continuer pendant un nombre important d'itérations inutiles.

Ainsi, la modification du paramètre threshold aura un impact sur le nombre d'itérations nécessaires pour converger et donc sur le nombre de points sur le graphe final. Si threshold est réduit, le nombre d'itérations nécessaires pour converger peut être réduit, mais le modèle peut ne pas être optimalement formé. Si threshold est augmenté, le nombre d'itérations nécessaires pour converger peut augmenter, mais le modèle peut être mieux formé. Il est donc important de choisir un seuil approprié pour obtenir des résultats optimaux.

Différents ensembles de données pour remplacer X et Y et observer les différences dans les graphiques produits. Par exemple :

X = np.array([1,2,3,4,5])

Y = np.array([2,4,6,8,10])

X = np.array([-1,-0.5,0,0.5,1])

Y = np.array([-2,-1,0,1,2])

X = np.array([1,2,3,4,5,6,7,8,9,10])

Y = np.array([2,3,5,7,11,13,17,19,23,29])

En utilisant des jeux de données différents, vous pouvez observer des différences dans les tendances et les formes des graphiques produits. Il est important de noter que le choix des données peut également affecter la performance de l'algorithme.

L'algorithme du gradient descent est un algorithme populaire pour la régression linéaire car il permet de trouver les valeurs optimales des coefficients (intercept et slope) pour un modèle de régression linéaire en minimisant la fonction de coût (mean squared error). Il s'agit d'un algorithme de descente stochastique qui itère sur les coefficients en les mettant à jour en utilisant la dérivée de la fonction de coût pour chaque itération, en utilisant la taux de mise à jour spécifié pour minimiser le coût. Cette méthode peut généralement trouver une solution optimale pour les jeux de données simples à complexe, ce qui la rend très utile pour la régression linéaire.

L'algorithme de descente de gradient est un algorithme populaire pour la régression linéaire car il permet de trouver les coefficients de la régression linéaire de manière efficace en minimisant l'erreur entre les valeurs prédites et les valeurs réelles. Cela est réalisé en utilisant des méthodes itératives pour ajuster les coefficients, ce qui est plus efficace que d'utiliser des méthodes analytiques telles que la méthode LDA qui nécessitent des calculs complexes pour trouver la solution.

D'un autre côté, la méthode LDA nécessite une distribution normale des données et des hypothèses supplémentaires sur les relations linéaires entre les variables, ce qui peut rendre cette méthode moins robuste et applicable uniquement à certaines situations. La descente de gradient, quant à elle, peut être appliquée à de nombreuses situations différentes sans ces hypothèses et restrictions supplémentaires.