Malembeti Rick José Pansa
Master 1 Informatique Semestre 2
Université Lille 1 – Sciences et Technologies
Année universitaire 2015/2016

FAA Compte-rendu des TPs

Première partie

1. Comparaison graphique matplotlib au modèle y = 2x + 3

L'execution provoque l'affichage dans la console des différentes mesures de performance (`J(theta)`), ainsi que l'apparition d'un graphique sur lequel figure en bleu les données d'entrée et en vert la courbe d'approximation `2x + 3` (theta0 = 2, theta1 = 3)

2. Calcul des différents J(thêta)

J abs (thêta) = 0.73987984094
J l1 (thêta) = 0.0896787983772
J l2 (thêta) = 0.804228687838
J l infini (thêta) = 2.51624302238

Deuxième partie

1. Calcul de thêta (moindres carrés)

Le calcul de thêta montre qu'il vaut pour theta1 1.95293789 et theta0 3.59623499.
Donc theta0 et theta1 se rapprochent respectivement des valeurs 2 et 3.

2. Calcul de l'erreur

jlabs = 0.73987984094
jl1 = 0.0896787983772
jl2 = 0.804228687838
jlinfini = 2.51624302238

3. Comparaison avec l'erreur du TP précédent

On reprend les mêmes données que pour le TP1, mais cette fois ci, l'estimation est calculée par la méthode des moindres carrés. L'execution donne la valeur de theta correspondant à (1.95293789, 3.59623499), et donne egalement l'erreur quadratique moyenne jl2 (0.769619957035)de la methode moinde carré.
Puis, on compare l'erreur quadratique moyenne jl2 du TP1 au jl2 de la methode de moindre carré et on remarque que le jl2 du TP1 est plus grand que celui du TP2 et donc la solution avec la methode des moindres carrés est plus efficace.

4. Subsidiaire : faire pareil avec Scikit Learn (linear regression)

Non traitée.
http://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares

Troisième partie

1. Analyse et comparaison graphique des vitesses de convergence des 2 algorithmes

Pour la descente de gradient globale (Batch) avec les paramètres alpha égale à 0.01 et le nombre d'iterations égale à 60000 et theta initial à [0,0] , on observe que l'erreur quadratique diminue en fonction du nombre d'iterations et finit par converger pour une valeur de j theta égale 0.77, et on a une valeur de théta correspondant à celle dans la méthode de moindre carré (theta :1.95293789, 3.59623499). Et graphiquement, nous obtenons la même droite que celle de la méthode des moindres carrés, ça montre la cohérence de la solution.
Pour des valeurs des paramètres alpha et le nombre d'iterations trop petites ou trop grandes, nous obtenons des résultats mitigés, avec parfois des valeurs de j theta trop grandes et des valeurs de theta trop petites (effet de surapprentissage).

Pour la descente de gradient (stochastique) avec les paramètres alpha égale à 0.0001 et le nombre d'iterations égale à 60000 et theta initial à [0,0], on observe que l'erreur quadratique diminue en fonction du nombre d'iterations et finit par converger beaucoup plus vite en temps d'exécution avec une valeur de j theta égale aussi à 0.77 et on a une valeur de théta correspondant à celle dans la méthode de moindre carré (theta :1.95293789, 3.59623499). Cela se justifie par le fait qu'au lieu d'apprendre sur toutes la base d'apprentissage comme la descente de gradient batch, à chaque iteration, on ne prend qu'une seule donnée de façon aléatoire. Et graphiquement, nous obtenons la même droite que celle de la méthode des moindres carrés, ça montre la cohérence de la solution.
Pour des valeurs des paramètres alpha et le nombre d'iterations trop petites ou trop grandes, nous obtenons des résultats mitigés, avec parfois des valeurs de j theta trop grandes et des valeurs de theta trop petites (effet de surapprentissage).
Ainsi, on a dans les deux algorithmes, une valeur jl2 un peu plus grande que celle de la moindre carré,.

Quatrième partie

1. Estimation du risque réel par validation croisée à K plis

Pour la base de donnée x0, y0, nous avons une fonction polynôme, et le polynôme le plus adapté se trouve à l'ordre de 23, car c'est le polynôme avec le risque réel le plus minimisé.
Cette fonction surapprend, pour un ordre trop grand (24 et plus), car plus l'ordre est grand et plus le modèle devient complexe , et le risque empirique augmente, donc on fait plus d'erreur, et le surapprentissage se justifie.

Pour la base de donnée x1, y1, nous avons une fonction , et la fonction la plus adaptée se trouve à l'ordre de 11, car c'est le polynôme avec le risque réel le plus minimisé.
Cette fonction surapprend, pour un ordre trop grand (14 et plus), car plus l'ordre est grand et plus le modèle devient complexe , et le risque empirique augmente, donc on fait plus d'erreur, et le surapprentissage se justifie.

Pour la base de donnée x2, y2, nous avons une fonction , et la fonction la plus adaptée se trouve à l'ordre de 7, car c'est le polynôme avec le risque réel le plus minimisé.
Cette fonction surapprend, pour un ordre trop grand (15 et plus), car plus l'ordre est grand et plus le modèle devient complexe , et le risque empirique augmente, donc on fait plus d'erreur, et le surapprentissage se justifie.

Cinquième partie

Cette partie a été codé, mais faute d'erreur dans le code, nous n'avons pu avoir des résultats interpretables.

Sixième partie

Non traitée.

Conclusion

En sommes, ces travaux pratiques ont été gratifiant, même si on a rencontré énormément de difficultés dans l'implémentation des algorithmes de gradients, mais également du surapprentissage. Et nous a permis de comprendre les algorithmes de machine learning, et dans quelle situation les utiliser, par exemple, on utilise l'algorithme des moindres carrés, seulement pour des fonctions convexes, car c'est pour ce type de fonction que l'algorithme des moindres carrés est efficace, les algorithmes de descente de gradient batch et stochastique, sont le plus utilisés quand nous avons des fonctions non convexes et la regression logistique permet de prédire l'appartenance d'une donnée à une classe.