Eric Marchessault
111 156 117

Techniques avancées en intelligence artificielle
IFT-4102

Projet final

Travail présenté à
Brahim Chaib-draa

Département d'informatique et de génie logiciel
Université Laval
Hiver 2018

Arbre de décision

Iris
accuracy: [0.8 0.84444444 0.86666667]
confusion matrix :
[[13. 0. 0.]
[5. 8. 0.]
[4. 2. 13.]]
execution time : 0.000949859619141

Congress
accuracy: [0.875 0.875]
confusion matrix :
[[81. 34.]
[4. 185.]]
execution time : 0.00268197059631

Monks-1
accuracy: [0.62037037 0.62037037]
confusion matrix :
[[210. 6.]
[158. 58.]]
execution time : 0.00298094749451

Monks-2
accuracy: [0.66435185 0.66435185]
confusion matrix :
[[280. 10.]
[135. 7.]]
execution time : 0.00276184082031

Monks-3
accuracy: [0.65046296 0.65046296]
confusion matrix :
[[67. 137.]
[14. 214.]]
execution time : 0.00269293785095

Réseau de neurones

Les test de cross-validation on été fait sur 6 dimensions = [5, 10, 20, 30, 40, 50] et sur 5 couche caché = [1, 2, 3, 4, 5]

Iris

Selon le dessins ci-dessus, la meilleure dimensions trouvé est donc 5

Selon le dessin ci-dessus, le meilleur nombre de couche caché est 4

Congress

Selon le dessin ci-dessus, la meilleure dimension est 30

Selon le dessin ci-dessus, le meilleur nombre de couche caché est 3

Monks-1

Selon le dessin ci-dessus, la meilleure dimension est 30

Selon le dessin ci-dessus, le meilleur nombre de couche caché est 4

Monks-2

Selon le dessin ci-dessus, la meilleure dimension est 10

Selon le dessin ci-dessus, le meilleur nombre de couche caché est 1

Monks-3

Selon le dessin ci-dessus, la meilleure dimension est 40

Selon le dessin ci-dessus, le meilleur nombre de couche caché est 2

	Iris	Congress	Monks1	Monks2	Monks3
nombre dimensions	5	30	30	10	40
nombre de couche caché	4	3	4	1	2

Choix du nombre de couches cachées

Les résultats sont plutôt variée. J'ai cependant pu remarquer le dataset Congress qui contenait plus de rangée d'input demandait plus de dimension que le dataset d'iris afin d'avoir un meilleur résultat. La tendance semble être que plus on ajoute de couche mieu le résultat va être positif. Sauf pour monks-2 et 3 dont l'effet inverse est identifié.

Initialisation des poids du Réseau de Neurones

Pour déterminer le poids des réseau de neurones, j'ai utilisé une fonction aléatoire entre -1 et 1. Après lecture de quelques articles sur internet, j'ai pu constaté que cela semblait être un standards de normalisation pour les réseau de neurones. J'ai cependant mis le seed(1) de ma fonction aléatoire afin d'avoir toujours les mêmes valeur et m'aider a mieu débugger. Ne sachant pas comment faire la learning curve. J'ai décidé d'insérer l'accuracy et la confusion matrix à la place

Iris
RN_NON_ZERO
accuracy: [1. 0.7047619 0.7047619]
confusion matrix :
[[37. 0. 0.]
[0. 37. 0.]
[0. 31. 0.]]

RN_ZERO
accuracy: [0.64761905 0.35238095 0.7047619]
confusion matrix :
[[0. 37. 0.]
[0. 37. 0.]
[0. 31. 0.]]
accuracy: [1. 0.7047619 0.7047619]

Congress
RN_NON_ZERO

accuracy: [0.99671053 0.99671053]
confusion matrix :
[[114. 1.]
[0. 189.]]

RN_ZERO
accuracy: [0.62171053 0.62171053]
confusion matrix :
[[0. 115.]
[0. 189.]]

Monks-1
RN_NON_ZERO
accuracy: [1. 1.]
confusion matrix :
[[62. 0.]
[0. 62.]]

RN_ZERO
accuracy: [0.5 0.5]
confusion matrix :
[[0. 62.]
[0. 62.]]

Monks-2
RN_NON_ZERO
accuracy: [0.97633136 0.97633136]
confusion matrix :
[[103. 2.]
[2. 62.]]

RN_ZERO
accuracy: [0.62130178 0.62130178]
confusion matrix :
[[105. 0.]
[64. 0.]]

Monks-3
RN_NON_ZERO
accuracy: [1. 1.]
confusion matrix :
[[62. 0.]
[0. 60.]]

RN_ZERO
accuracy: [0.50819672 0.50819672]
confusion matrix :
[[62.   0.]
 [60.   0.]]

Nous pouvons remarquer qu'il y a un différence entre le résultat d'un réseau de neurones entraîné avec des poids aléatoire et des de 0. Nos réseau de neurones entraîné avec des valeur allant de -1 à 1 nous donne d'excellent résultats. Entre 97 % et 100 %. Contrairement au réseau avec des poids de zéro dont les résultat avoisine les 50 %. Un problème semble cependant survenir. Les RN_ZERO semble prédire toutes les même classes. Est-ce un problème normal ou un que j'ai créer par inadvertance ? Je ne sais pas.

Entrainement et Test

Je n'ai malheureusement pas réussi à utiliser mon réseau de neurones avec les données de test. J'ai réussi a garder la valeur de mes poids générer lors de l'entraînement. Mais lors de la série de test, un problème de multiplication de matrice ce produit. J'ai donc pu mettre le résultat de l'entraînement avec son temps d'exécution.

Iris
confusion matrix :
[[37.   0.   0.]
 [0. 37.   0.]
 [0. 31.   0.]]
execution time : 4.32379412651

Congress
accuracy: [0.99671053 0.99671053]
confusion matrix :
[[114. 1.]
 [0. 189.]]
execution time : 48.1497869492

Monks-1
accuracy: [1. 1.]
confusion matrix :
[[62.   0.]
 [0. 62.]]
execution time : 21.4086170197

Monks-2
accuracy: [0.97633136 0.97633136]
confusion matrix :
[[103. 2.]
 [2.   62.]]

execution time : 7.34144186974

Monks-3
accuracy: [1. 1.]
confusion matrix :
[[62.   0.]
 [0. 60.]]
execution time : 26.1554481983

Problèmes

Malheureusement, je n’ai pas réussi à implémenter correctement la learning curve pour le decision tree. Le code que j’ai tenté affichait la même accuracy pour chaque grandeur de l’ensemble d'entraînement. J’ai essayé plusieurs type de donnée différent pour avoir un résultat, mais en vain. J’ai entrainé mon modèle en utilisant des données d'entraînement puis tenter de prédire les résultat sur ces données, mais mon accuracy avait le résultat inverse c’est à dire une courbe descendante. J’ai utilisé des données d'entraînement différent en faisant de la cross-validation, j’ai tenté d'utiliser les classes de mon ensemble de test. les J’ai laisser le code dans le fichier. Cependant, il ne sera pas exécutée lors du lancement du main.

Par le fait même, je n’ai pas réussi à faire la courbe d’apprentissage pour l’initialisation des poids du réseau de neurones. J’ai donc simplement inséré une comparaison de leur accuracy.

Après avoir entraîné mes réseau de neurones selon le nombre de dimension et le nombre de layers recommandé. Je me suis rendu compte que je n'étais pas capable de tester mes réseau. Lors de la comparaison j’avais une erreur de produit matriciel et je ne savais pas d'où provenait le problème.

Conclusion

La construction d'un arbre de décision est plus simple que celle d'un réseau de neurones. Elle demande aussi beaucoup moins de temps. Les résultats obtenue dans notre arbre était assez satisfaisant. 80% pour Iris et 87% pour Congress.
Le temps d'entraînement d'un réseau de neurone peut être très long si nous devons compté le temps pour trouvé le nombre de dimension et le nombre de layers recommandé en utilisant la validation croisé. Cependant les résultat sont beaucoup plus précis près de 100% (en testant sur les données de test)

J'ai appris qu'il était assez compliqué de faire un réseau de neurone. Je n'ai malheureusement pas réussi à le faire correctement et j'ai donc préféré ne pas ajouter des prédiction erroné dans le projet.

Je suis plutôt déçu du cours IFT-4102, car il était a nous de comprendre pourquoi certains concept ne fonctionnait pas. Il a fallu que je me réfère à d'autre source pour avoir une explication *claire.* Je comprends que le travail consiste à aller fouiller sur internet, mais cela sans indication claire que mon travail est sur la bonne voie, il est possible de travailler des heures dans la mauvaise directions. Peut-être travailler sur un seul dataset simple pourrait faciliter la compréhension ou sur un réseau de neurones au lieu des 2 aurait pu permettre d'avoir un meilleur focus. La charge de travail pour ce TP était assez grande.