LE MESLE Maxime
ROBLIN Clément

2021

Rapport de projet Big Data

Réseau de neurones : reconnaissance des fonctions alcools

I/ Présentation du sujet

Le but de ce projet était de créer un réseau de neurones linéaires dans le but de reconnaître les fonctions alcool dans une molécule écrite en formule condensée.

Concernant l'organisation et la répartition des tâches, nous avons passé une première partie assez importante concernant la compréhension du réseau de neurones étant donné qu'il n'avait pas été traité en cours. *Nous vous remercions par ailleurs sur les éclaircissements donnés en TP à ce sujet*.

Ensuite nous avons commencé par implémenter la version de détection de la lettre a, afin de valider l'algorithme et faire nos premières analyses sur les résultats.

Enfin, nous avons cherché sur quelles entrées ce réseau pouvait s'appliquer, et les molécules d'alcool étaient effectivement parmi les plus simples à mettre en place.

Pour la répartition des tâches, tout a été fait en binôme lors des TP et de conférences pour mettre en commun l'avancement que ce soit sur l'implémentation de l'algo et de son fonctionnement ainsi que les différentes batteries de test et leur analyse.

II/ Implémentation

L'implémentation s'est faite sur le modèle suivant :

- les mots en entrée sont convertis en binaire pour être traités par les neurones
- 3 matrices W correspondant aux couches du réseau, contenant les poids générés aléatoirement ([-10, 10]) ainsi que le biais dans la dernière colonne (+-1)
- Un comparateur de Heaviside pour décider de la sortie de chaque neurone

Le mot converti en binaire est transposé dans un vecteur b contenant une ligne supplémentaire avec 1 pour appliquer le biais.
La sortie de la première couche est ye=We.'*b
De la même manière yc=Wc.'*ye
En sortie ys=Ws.'*yc , indiquant la nature du mot.

L'entraînement permet d'ajuster le poids et le bias de chaque neurone avec :

- faux positif : augmentation du biais , diminution des poids utilisés (là où xi=1)
- faux négatif : diminution du biais, augmentation des poids où xi=1

En sortie on affiche le nombre de mots reconnus ainsi que les faux positifs et les faux négatifs

III/ Analyse des résultats

1) Détection de 'a'

Après avoir implémenté le réseau, nous l'avons testé sur la recherche de mots contenant le caractère 'a'. Les résultats ci-dessous montrent l'évolution du taux de succès d'identification en fonction de la taille du mot en lettres et de la taille du réseau de neurone. Les données de test contiennent environ 15% de mots avec une lettre 'a', en effet nous avons généré des mots de 4 lettres dont chaque lettre est tirée aléatoirement parmi les 26. Avoir un ratio de mots contenant la lettre 'a' de 50% dans les données d'entraînement nous aurait certainement permis d'obtenir de meilleurs résultats, selon la taille de celles-ci. On obtient en moyenne les résultats ci-dessous :

nbr neurones = neurones entrée, neurones cachés, neurones sortie

taille du mot -> nbr neurones :	4	8	20
3,2,1	84,1%	74,5%	45,8%
5,5,1	78,0%	73,8%	46,8%
20,20,1	81,3%	68,3%	46,5%

On peut remarquer que pour les mots très petits le taux de détection avoisine les 80%, et qui descend assez rapidement si le mot est plus grand.

Le nombre de neurones ne semble pas améliorer la détection, cela pourrait s'expliquer par la simplicité de la couche cachée implémentée (1 seule couche, fonction très basique). En revanche, nous avons remarqué que plus la taille du réseau augmente, plus le nombre de faux positifs/négatifs s'équilibre, le nombre de faux négatifs étant toujours le plus élevé et celui de faux négatifs presque nul pour des réseaux à peu de neurones,

De plus, il arrive fréquemment d'obtenir de temps à autre des résultats très inférieurs ou très supérieurs à ceux obtenus dans la plupart des cas, cela étant lié à la génération aléatoire des matrices W.

Le taux d'entraînement n'influe pas ou peu sur le ratio de détection, comme le montre les résultats du réseau 5,5,1 avec des mots de 4 lettres :

taux d'entraînement	100	1000	10000
ratio	80%	86%	85,7%

2) Détection des fonctions alcool

Pour parvenir à la détection des fonctions alcool, il nous fallait dans un premier temps une base de données contenant diverses molécules en écriture condensée, et renseignant si ces molécules possèdent ou non un groupement alcool, c'est-à-dire un atome de carbone relié à un atome d'oxygène lui même relié à un atome d'hydrogène.

La génération de la base de données ne faisant pas partie du sujet, nous ne l'avons pas codée en matlab mais en python par souci de rapidité. Le code nous

ayant permis de générer ces molécules, appelé genmol.py, est associé à ce rapport. Son fonctionnement est assez basique. Il génère d'abord aléatoirement un nombre d'atomes de carbone entre 2 et 5, puis les dispose aléatoirement et ajoute tous les atomes d'hydrogène nécessaires pour équilibrer la molécule.

Ensuite, le code a une certaine probabilité de remplacer deux hydrogènes d'un carbone par un oxygène faisant une double liaison.

Puis enfin le code a une probabilité de remplacer un atome d'hydrogène par soit un groupement -OOH, auquel cas nous n'avons toujours pas de fonction alcool, soit de le remplacer par un groupement -OH, qui correspond forcément à un groupement alcool ici puisque tous les hydrogènes sont encore liés à un carbone.

Cette génération a ses limites, car on essaye jamais d'ajouter plusieurs =O, -OOH ou -OH à une molécule. De plus, lorsqu'on remplace un hydrogène dans une parenthèse, on change en réalité plusieurs hydrogènes à la fois, ex : CH(CH3)3 devient CH(CH2OH)3 et non CH(CH3)2(CH2OH). Enfin, nos molécules ne contiennent que trois atomes différents.

Nous avons ainsi généré une centaine de molécules très différentes de taille variable dans un fichier .csv que nous avons importé sur matlab. Nous avons standardisé la taille des molécules à 18, la taille de la plus longue molécule du csv, en ajoutant des caractères 'S' à la fin des molécules trop courtes. Par souci d'optimisation, nous n'avons pas utilisé de2bi pour coder nos molécules en binaire, mais avons créé notre propre codage binaire sur trois bits, puisque nous n'utilisons que 8 caractères (C, O, H, (,), 2, 3 et S). Enfin nous avons pu tester notre réseau de neurones presque de la même manière que nous l'avions fait pour la détection de la lettre 'a'.

En terme de résultats, la moyenne avoisine les 60 % de taux de réussite ce qui est plutôt élevé étant donné la simplicité du réseau et la longueur des molécules (18*3 bits), avec un réseau de neurones de taille 5,5,1.

IV/ Conclusion

Le réseau de neurones est effectivement un algorithme très simple et sensible aux matrices W initiales ainsi qu'à la taille des données d'entrée. Toutefois nous avons observé des performances surprenamment satisfaisantes sur l'exemple des fonctions alcool.