

Algorithmes pour l'IA – TD 4

Perceptrons

Le tableau ci-dessous contient un petit jeu de données¹ de caractéristiques (poids et couleur) d'exemples de fruit (pommes, mandarines et citrons).

	fruit	poids	couleur
0	pomme	164	0.70
1	pomme	152	0.69
2	pomme	156	0.69
3	mandarine	86	0.80
4	mandarine	84	0.79
5	mandarine	80	0.77
6	citron	132	0.73
7	citron	130	0.71
8	citron	116	0.72

T1: Dessinez un perceptron simple qui pourrait être utilisé pour classifier les fruit dans les trois catégories pomme, mandarine et citron d'après leur poids et couleur. N'oubliez pas les biais.

T2: On appelle les variables d'entrée x_p et x_c pour le poids et la couleur, et les variables de sortie z_p , z_m et z_c pour pomme, mandarine et citron et on utilise les mêmes indices pour les poids des connexions (e.g. w_{mc} est le poids de la connexion entre x_c et z_m). wbp , wbm , wbc sont les poids associés au biais pour les trois neurones de sortie (pomme, mandarine et citron). Quels résultats obtient on lorsqu'on initialise ces poids aux valeurs suivantes ? En d'autres termes, quelle est l'erreur absolue moyenne de la classification obtenue pour chaque catégorie de fruit sur le jeu de données ci-dessus.

wpp=	-7.61	wpc=	-13.71	wbp=	-15.45
wmp=	-0.123	wmc=	7.62	wbm=	8.08
wcp=	5.80	wcc=	4.43	wbc=	4.65

T3: Réalisez trois itérations d'apprentissage sur le réseau précédent, initialisés avec les poids précédents, avec un taux d'apprentissage de 0,1.

T4: Essayez de classifier un fruit dont le poids est 192 et la couleur 0,55. Quel est le résultat ?

T5: Essayez de classifier un fruit dont le poids est 154 et la couleur 0,82. Quel est le résultat ?

¹extrait adapté de https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/fruit_data_with_colors.txt

T6: Le notebook à

<https://colab.research.google.com/drive/1ILV7Iu-dlkuFfLGUMzE0IT1ZC2h84K-3> contient du code python pour réaliser les calculs précédents. Exécutez le pour vérifier vos résultats et comprendre comment il fonctionne.

T7: Dans le même notebook, à la suite, vous trouverez une début de code pour réaliser la classification sur le jeu de données complet avec un perceptron multi-couche en utilisant la librairie scikit-learn. Exécutez ce code et observez les résultats. Est-ce que ça marche bien ?

T8: En vous aidant de la documentation de scikit-learn², ajoutez et modifiez les paramètres pour essayer de rendre la classification plus efficace. Concentrez vous en particulier sur le nombre d'itérations (*max_iter*), la taille de la ou des couches cachées (*hidden_layer_sizes*) et le taux d'apprentissage (*learning_rate_init*)³.

T9: Le code suivant réalise une standardisation (normalisation par la moyenne) des variables d'entrée du réseaux. Utilisez le avant la création du réseau et voyez comment les résultats changent. Pourquoi ?

```
for k in X: X[k] = (X[k] - X[k].mean()) / X[k].std()
```

T10: Le code suivant permet de séparer le jeu de données en un jeu d'entraînement (*X_train, y_train*) contenant 80% des données, et un jeu de test (*X_test, y_test*) contenant le reste. Ajoutez ce code au notebook et modifier le code du notebook afin de réaliser l'entraînement sur le jeu d'entraînement, de calculer le score du résultat de la classification sur le jeu d'entraînement et le jeu de test, et de visualiser les résultats (matrice de confusion) sur le jeu de test. Observez et essayer d'expliquer les résultats.

```
X_train,X_test,y_train,y_test = train_test_split(X, y,  
                                                test_size=0.2,random_state=1)
```

Note : L'implementation du perceptron multi-couche dans scikit-learn fonctionne bien, mais ne permet pas de bénéficier de l'utilisation de GPU. Pour cela, on se tournera vers des librairies telles que tensorflow/keras ou PyTorch, qui permettent aussi plus de flexibilité sur le type, la taille et la structure des réseaux de neurones utilisés (e.g. CNN, LSTM, etc.)

²https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

³Le paramètre *random_state* permet de fixer l'initialisation des poids, et donc de toujours obtenir le même résultat si on ne change pas les autres paramètres)