Le génie pour l'industrie

GTI770
Systèmes intelligents et apprentissage machine

TP03 — Machines à vecteur de support et réseaux neuronaux

1. Contexte

Ce troisième laboratoire porte sur l'utilisation de deux nouveaux algorithmes de classification, soit les machines à vecteurs de support (*SVM*) et les réseaux neuronaux. Dans ce laboratoire, vous êtes amenés à utiliser de nouvelles approches à l'aide de ces deux algorithmes afin de résoudre le problème de classification des galaxies introduit dans le cadre du premier laboratoire de ce cours.

En premier lieux, vous serez amenés à travailler avec les réseaux neuronaux. Vous aurez à élaborer un modèle simple appelé *Multi-Layer Perceptron* qui apprendra à classifier les échantillons d'images de l'ensemble de données des galaxies. Dans un deuxième temps, vous étudierez un second modèle d'apprentissage, soit les Machines à vecteur de support (*SVM*), qui propose un modèle d'optimisation convexe. Dans les deux cas, vous devrez utiliser les primitives fournies jumelées aux primitives développées au premier laboratoire lorsque celles-ci ne sont pas redondantes. Vous devrez également étudier les hyperparamètres du modèle.

Tout comme les deux premiers laboratoires de ce cours, vous réaliserez ce troisième travail avec la technologie Python3 conjointement avec la librairie d'apprentissage machine *scikit-learn*. Toutefois, pour résoudre le problème avec les réseaux neuronaux, vous devez utiliser soit la librairie Google *TensorFlow* ou la librairie *Keras* qui propose l'ensemble des méthodes nécessaires afin de bâtir votre réseau de neurones. Vous êtes invités à reprendre le code développé lors des laboratoires 1 et 2 afin de continuer son développement.

2. Durée du laboratoire

Vous disposez d'un total de trois (3) séances de laboratoire afin d'effectuer ce travail. En fonction de votre groupe, vous disposez des séances suivantes :

Tableau 2.1 : Date des séances et date de remise

Date des séances	GTI770-01	GTI770-02
Séance 1	31 octobre 2018	31 octobre 2018
Séance 2	8 novembre 2018	8 novembre 2018

Date des séances	GTI770-01	GTI770-02
Séance 3	15 novembre 2018	15 novembre 2018
Remise	21 novembre 2018	21 novembre 2018

3. Objectifs

Les principaux objectifs de ce laboratoire sont :

1. Produire un code source utilisant les technologies Python et des techniques d'apprentissage machine permettant de classer des échantillons de donnés automatiquement;
2. Se familiariser avec l'algorithme des SVM et les réseaux neuronaux;
3. Analyser l'impact des hyperparamètres des différents algorithmes utilisés et l'impact des structures de modèles de réseaux neuronaux;
4. Se familiariser avec le langage de programmation *Python* et les outils et librairies scientifiques d'apprentissage machine;

4. Matériel fourni

Vous disposez du matériel suivant :

- Gabarit du rapport de laboratoire;
- Ensemble de données d'entrainement comportant 16 908 images de galaxies et un fichier CSV contenant l'étiquette associée à chaque image;
- Code source de base permettant de lire les fichiers images et de lire le fichier contenant le vecteur de primitives des courriels;

5. Manipulations

Réseaux neuronaux

- Avec la librairie TensorFlow / Keras, définissez un réseau de neurones de base. Pour ce faire, vous pouvez vous inspirer de la littérature, de recherches Internet, de la documentation de *TensorFlow* ou d'autres sources. Ce modèle doit être original. Il s'agit ici de construire un *Multi-Layer Perceptron model.* Faites attention aux nombres de perceptrons à l'entrée et à la sortie ; la première valeur doit concorder avec le nombre de caractéristiques [*feature*s] que votre vecteur en entrée comporte [son nombre de dimensions] alors que la deuxième doit concorder avec le nombre de classes que vous avez à la sortie.

- Dans votre code source, vous devrez ajouter le code nécessaire à l'outil *TensorBoard*. Cet outil vous permettra alors de suivre l'évolution des hyperparamètres, via un serveur Web local sur votre machine, et vous facilitera grandement la tâche dans votre étude des réseaux neuronaux. Vous devez ajouter le code permettant de capturer les valeurs de la précision [*accuracy*] et la mesure de l'inconsistance entre la valeur prédite et la vraie valeur [*loss*].

Tableau 5.1: Matrice des hyperparamètres de base

Hyperparamètre	**Valeur de base**
Nombre de couches total	4, y compris la couche d'entrée (*input*) et la couche de sortie (*output*)
Nombre de perceptrons dans la couche cachée (*hidden layer*)	100, 100, 2
Nombre d'itérations (*epochs)*	60
Taux d'apprentissage (*learning rate*)	0.0005
Batch size	100

- Faites l'étude de ce modèle d'apprentissage. Pour ce faire, vous devez faire varier :
    - Le nombre de couches;
    - Le nombre de perceptrons dans ces couches intermédiaires;
    - Le nombre d'itérations (*epochs*);
    - Le taux d'apprentissage (*learning rate*).

  Pour chaque variable ci-dessus, vous devez sélectionner trois valeurs différentes et lancer l'apprentissage avec ces valeurs. Vous aurez alors 12 modèles d'apprentissages différents. **Notez les résultats de la précision [*accuracy*] que vous avez obtenue au final**.

  À la fin de vos manipulations, vous devriez avoir, par le biais de *TensorBoard*, des graphiques comme ci-dessous décrivant la précision [*accuracy*] et le *loss.*

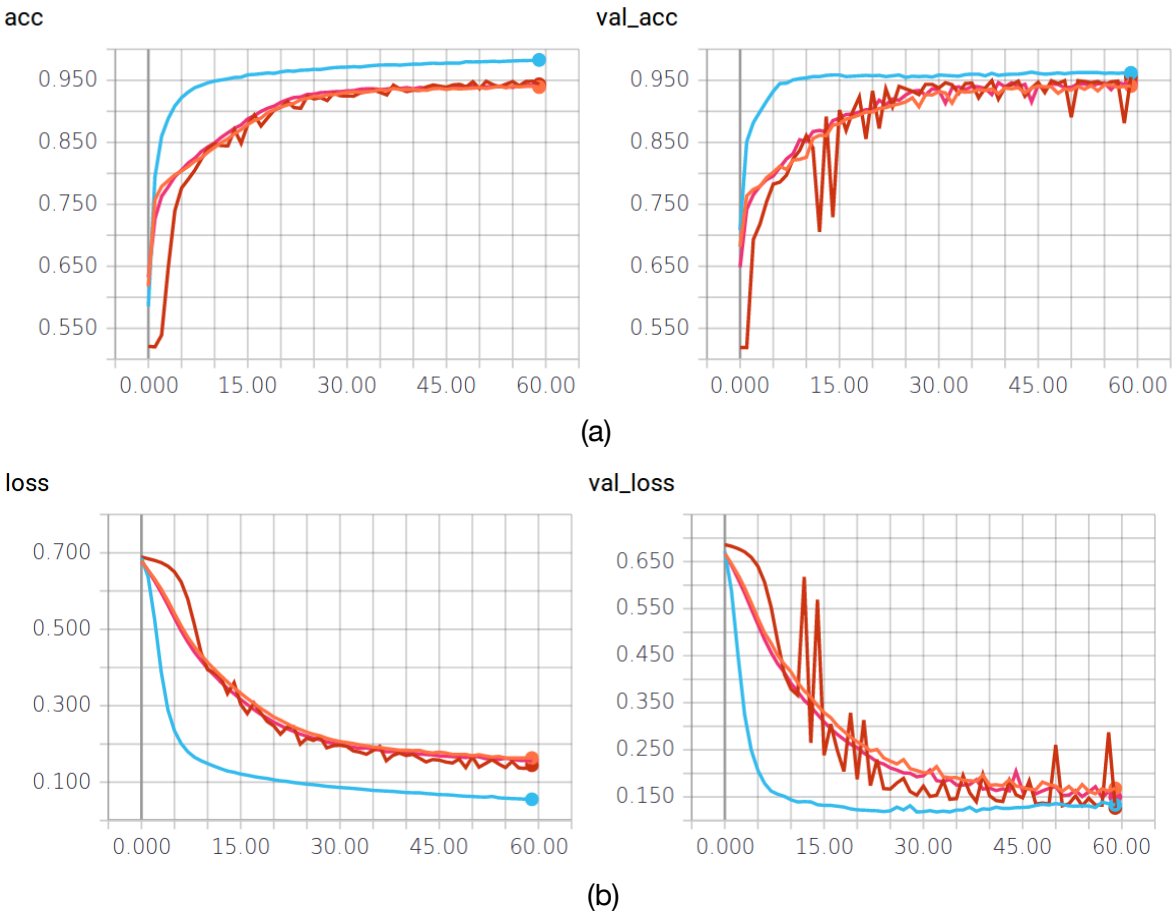

Figure 5.2 : [a] évolution de la précision [*accuracy*] avec le nombre d'itérations (epochs); (b) évolution de la courbe de la mesure de l'inconsistance entre la valeur prédite et la vraie valeur [*loss*].

Machines à vecteur de support

- À l'aide de la librairie *scikit-learn*, produisez un code source permettant de classifier les galaxies avec l'aide de l'algorithme *SVM.*
- Plusieurs modèles calculés à l'aide de *SVM* avec différentes valeurs d'hyperparamètres sont demandés. Fiez-vous aux matrices ci-dessous afin de les réaliser:

Tableau 5.2 : Matrice des hyperparemètres - SVM kernel = 'linear'

Poids des classes / **Variable C**	**class_weight= {'balanced'}**
1E-03	X
1E-01	X
1.0	X
10.0	X

Tableau 5.3 : Matrice des hyperparemètres - SVM kernel = 'rbf'

Paramètre γ (gamma) / **Variable C**	γ=1E-03	γ=1E-01	γ=1.0	γ=10.0
1E-03	X	X	X	X
1E-01	X	X	X	X
1.0	X	X	X	X
10.0	X	X	X	X

Notes sur les SVM :

Vous pouvez utiliser la classe *sklearn.model_selection.GridSearchCV* afin d'effectuer la recherche des meilleurs hyperparamètres. De ce fait, vous n'aurez pas besoin de changer manuellement les valeurs à chaque exécution. De plus, vous pouvez utiliser la classe *sklearn.model_selection. StratifiedShuffleSplit* afin de diviser l'ensemble de données en un *n* d'ensembles afin d'effectuer une validation croisée, le tout de manière automatique et aléatoire. Pour plus d'information, fiez-vous à l'exemple donné à la page *RBF SVM Parameters* disponible en annexe.

De plus, vous pouvez augmenter la vitesse d'exécution en parallélisant la recherche des hyperparamètres. En effet, la fonction *GridSearchCV* possède l'argument `n_jobs=X` où `X` est le nombre d'exécutions parallèle désiré (généralement proportionnel au nombre de coeurs x86 de la machine sur lequel s'exécute le code Python). Aussi, assurez-vous que votre appel de méthode spécifie une taille de cache suffisante pour le noyau SVM. Par l'intermédiaire du paramètre `cache_size`, vous serez en mesure de spécifier la taille de la cache désirée, ce qui peut réduire considérablement le temps de résolution de l'optimisation quadratique du problème. Ainsi, en spécifiant `cache_size=2048` dans la déclaration de votre classificateur, vous spécifiez que l'algorithme peut utiliser jusqu'à 2 Go de mémoire vive [*RAM*] à titre de cache pour le noyau. **Attention : votre ordinateur doit avoir suffisamment de mémoire vive. Assurez-vous de ne pas spécifier une valeur trop haute pour ne pas saturer la mémoire vive de votre ordinateur.**

- Notez bien tous les résultats de la précision [*accuracy*] de chaque modèle calculé. Vous devrez représenter ces résultats dans un tableau dans votre rapport.

5. Rapport

Votre *Jupyter Notebook* devra contenir les réponses aux questions suivantes. Il devra notamment avoir une analyse détaillée des résultats de classification obtenus par les différents modèles et leurs variations d'hyperparamètres.

5.1 Questions du rapport

1. Parmi les méthodes de validation (*Leave-one-out cross-validation*, *Leave-p-out cross-validation, k-fold cross-validation)*, présentez l'approche de validation que vous avez utilisée et pourquoi vous l'avez utilisée. Faites des liens avec les modèles d'apprentissage à l'étude. Rappelez-vous que vous ne devez sélectionner qu'une seule méthode qui servira à la validation des deux modèles à l'études dans ce laboratoire.
2. Décrivez la méthode de normalisation de donnés utilisée.

3. Décrivez la structure et le choix de votre modèle d'apprentissage *MLP*. Faites un parallèle entre votre description et votre implémentation. Quelle est la fonction de coût utilisée ? Pourquoi avez-vous utilisé cette fonction plutôt qu'une autre (quels sont les avantages et inconvénients ?)
4. Avec les graphiques créés par *TensorBoard*, après combien d'itérations/*epochs* êtes-vous en état de surapprentissage [*overfitting*]? À votre avis, quel est le nombre d'*epochs* optimal pour votre modèle?
5. Pour chacun des modèles d'apprentissage élaborés, présentez sous forme de tableau et de graphique les résultats de la précision (*accuracy*) et du score F1. Expliquez l'impact des hyperparamètres sur les performances du modèle *MLP*. Comment se traduisent les divers changements de paramètres ou pourquoi ont-il cet impact sur la performance du modèle ? Expliquer le score F1 que vous avez obtenu en fonction de chaque classe.
6. Présentez brièvement la méthode que vous avez utilisée afin de trouver le meilleur modèle *SVM*. Quels ont été vos résultats? Quels sont les impacts des hyperparamètres et leur utilité respective?
7. Quel est l'impact de la taille de l'ensemble d'apprentissages sur la performance de classification des différents modèles? Inclure dans votre graphique les données des trois algorithmes du laboratoire 2.
8. Quel type de classificateur recommanderiez-vous pour l'ensemble de données des galaxies et dans quelles conditions (par exemple mais non exhaustif, le nombre de données privilégié, les hyperparamètres, le temps de calcul, le matériel nécessaire, les scores de performance)? Discutez des performances que vous avez obtenues entre les modèles d'apprentissage utilisant les réseaux de neurones et les modèles *SVM*.
9. Formulez quelques pistes d'amélioration des classificateurs.

Bonus [+10 points]

Dans ce bonus, vous êtes amené à utiliser directement les pixels de l'image plutôt qu'un vecteur de primitives. Pour ce faire, vous devez créer un modèle appelé «réseau de neurones convolutionnel» ou plus communément appelé *Convolutional Neural Network*.

- À titre de préparation des données, recadrer les images de sorte qu'elles aient toutes :
    - La même taille;
    - Une taille inférieure à la taille originale (par exemple, 212 x 212 pixels);
    - Aucune perte de détails (ne pas trop rapetisser l'image et ainsi perdre de l'information importante de l'image).

  Vous pouvez utiliser la librairie *OpenCV* ou *scikit-image* afin d'effectuer cette opération.
  Ainsi, vous sauverez du temps de calcul puisque le vecteur d'entrées sera composé de moins de pixels.

Si vous décidez d'effectuer ce bonus, vous devez expliquer dans le rapport comment fonctionne, de façon superficielle, ce type de réseau de neurones tout en faisant un parallèle avec l'implémentation réalisée. Vous devrez également comparer les scores de performances avec les autres modèles qui sont à l'étude dans ce présent laboratoire. Vous devrez étudier les différents hyperparamètres qui sont :

- le taux d'apprentissage (*learning rate*) si applicable à votre modèle choisi;
- la taille de la *mini-batch;*
- le nombre de couches de convolution.

Annexe 1 : Sources d'information pertinentes

LIENS DIVERS

Documentation de la librairie *scikit-learn* : http://scikit-learn.org/stable/documentation.html

Documentation de la librairie *Google TensorFlow* : https://www.tensorflow.org/api_docs/python/

TensorBoard: Graph Visualization: https://www.tensorflow.org/get_started/graph_viz

RBF SVM Parameters: http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

LIVRES

Aurélien Géron. 2017. *Hands-On Machine Learning with Scikit-Learn and TensorFlow—Concept, Tools, and Techniques to Build Intelligent Systems.* 566 p. ISBN-13: 978–1491962299

Annexe 3 : Description des primitives des galaxies

#	Nom	Description
1	Couleur moyenne du centre [0]	Sur les pixels différents de zéro de l'image originale, la couleur moyenne du centre est extraite.
2	Couleur moyenne du centre [1]	Sur les pixels différents de zéro de l'image originale, la couleur moyenne du centre est extraite.
3	Couleur moyenne du centre [2]	Sur les pixels différents de zéro de l'image originale, la couleur moyenne du centre est extraite.
4	Couleur moyenne [0]	Sur les pixels différents de zéro de l'image originale, une moyenne de la couleur *bleue* est extraite.
5	Couleur moyenne [1]	Sur les pixels différents de zéro de l'image originale, une moyenne de la couleur *vert* est extraite.
6	Couleur moyenne [2]	Sur les pixels différents de zéro de l'image originale, une moyenne de la couleur *rouge* est extraite.
7	Standard Deviation [0]	Sur les pixels différents de zéro de l'image originale, la déviation standard est extraite.
8	Standard Deviation [1]	Sur les pixels différents de zéro de l'image originale, la déviation standard est extraite.
9	Standard Deviation [2]	Sur les pixels différents de zéro de l'image originale, la déviation standard est extraite.
10	Distribution Kurtosis [0]	Distribution kurtosis (mesure de l'aplatissement) sur les pixels différents de zéro.
11	Distribution Kurtosis [1]	Distribution kurtosis (mesure de l'aplatissement) sur les pixels différents de zéro.
12	Distribution Kurtosis [2]	Distribution kurtosis (mesure de l'aplatissement) sur les pixels différents de zéro.
13	Distribution normale asymétrique [0]	Sur les pixels différents de zéro, la distribution normale asymétrique est appliquée.
14	Distribution normale asymétrique [1]	Sur les pixels différents de zéro, la distribution normale asymétrique est appliquée.

#	Nom	Description
15	Distribution normale asymétrique [2]	Sur les pixels différents de zéro, la distribution normale asymétrique est appliquée.
16	Coefficient Gini [0]	Mesure de la dispersion des pixels différents de zéro.
17	Coefficient Gini [1]	Mesure de la dispersion des pixels différents de zéro.
18	Coefficient Gini [2]	Mesure de la dispersion des pixels différents de zéro.
19	Excentricité	Caractéristique de la courbure de la galaxie.
20	Largeur	Mesure de la largeur de la galaxie.
21	Hauteur	Mesure de la hauteur de la galaxie.
22	Somme	Somme des pixels de l'image *threasholded.*
23	Entropie	Mesure de l'entropie (du hasard) dans l'image.
24	Chiralité	Mesure de l'asymétrie de l'image.
25	Aire de l'ellipse	Mesure de l'aire de l'ellipse.
26	Aire *box-to-image*	Mesure de l'aire de la galaxie adaptée à une boîte.
27	Décalage du centre (*offset*)	Le décalage du centre de la galaxie par rapport à l'image.
28	Rayon de la lumière [0]	La mesure du rayon de la lumière (rayon de la galaxie).
29	Rayon de la lumière [1]	La mesure du rayon de la lumière (rayon de la galaxie).
30	Nombre de *labels*	Le nombre de caractéristiques (f*eatures*) repérées dans l'image.
31	Distribution Kurtosis [0]	Distribution kurtosis (mesure de l'aplatissement) sur les pixels de l'image couleur.
32	Distribution Kurtosis [1]	Distribution kurtosis (mesure de l'aplatissement) sur les pixels de l'image couleur.
33	Distribution Kurtosis [2]	Distribution kurtosis (mesure de l'aplatissement) sur les pixels de l'image couleur.

#	Nom	Description
34	Distribution normale asymétrique [0]	Sur les pixels de l'image couleur, la distribution normale asymétrique est appliquée.
35	Distribution normale asymétrique [1]	Sur les pixels de l'image couleur, la distribution normale asymétrique est appliquée.
36	Distribution normale asymétrique [2]	Sur les pixels de l'image couleur, la distribution normale asymétrique est appliquée.
37	Coefficient Gini [0]	Mesure de la dispersion des pixels de l'image en couleur.
38	Coefficient Gini [1]	Mesure de la dispersion des pixels de l'image en couleur.
39	Coefficient Gini [2]	Mesure de la dispersion des pixels de l'image en couleur.
40	Distribution Kurtosis image noir et blanc	Mesure de la dispersion des pixels de l'image en nuances de gris.
41	Distribution normale asymétrique image noir et blanc	Sur les pixels de l'image en nuances de gris, la distribution normale asymétrique est appliquée.
42	Coefficient Gini image noir et blanc	Mesure de la dispersion des pixels de l'image en nuances de gris.
43	Couleur du centre [0]	Couleur du canal *bleu* du pixel du centre.
44	Couleur du centre [1]	Couleur du canal *vert* du pixel du centre.
45	Couleur du centre [2]	Couleur du canal *rouge* du pixel du centre.
46	Couleur moyenne [0]	Sur les pixels de l'image couleur, une moyenne de la couleur *bleue* est extraite.
47	Couleur moyenne [1]	Sur les pixels de l'image couleur, une moyenne de la couleur *vert* est extraite.
48	Couleur moyenne [2]	Sur les pixels de l'image couleur, une moyenne de la couleur *rouge* est extraite.
49	Couleur moyenne du centre [0]	Sur les pixels de l'image couleur, la couleur moyenne en *bleu* du centre est extraite.
50	Couleur moyenne du centre [1]	Sur les pixels de l'image couleur la couleur moyenne en *vert* du centre est extraite.
51	Couleur moyenne du centre [2]	Sur les pixels de l'image couleur, la couleur moyenne en *rouge* du centre est extraite.

#	Nom	Description
52	Couleur du centre	Couleur du pixel du centre de l'image en nuances de gris.
53	Rapport couleur du centre / moyenne de gris	Rapport.
53 à 74	Moments de l'image	Moyenne pondérée de l'intensité des pixels des images *thresholded*, en nuances de gris et de l'image représentant la magnitude des gradients de l'image originale.