Projet-1ère Partie

IFT-4102 et IFT-7025 : Techniques avancées en Intelligence Artificielle

À rendre avant le 23 mars 2020 à 23h55

Introduction

Dans cette première partie du projet, vous devez implémenter deux techniques d'apprentissage machine : les K plus proches voisins et la classification naïve bayésienne. La meilleure façon d'apprendre c'est de programmer sans avoir recours aux bibliothèques déjà prêtes à faire tout le travail d'apprentissage (telle que `scikit-learn`), il est donc interdit d'utiliser toute bibliothèque ou tout programme déjà fait. Dès lors, il vous faut programmer de A à Z et si jamais vous voulez utiliser une partie déjà faite il vous faut demander l'autorisation à Alexandre Lemire-Paquin. Afin d'évaluer l'efficacité de ces techniques, vous devez entrainer et tester votre code sur des bases de données (datasets) réelles.

Pour toute aide veuillez vous adresser à Alexandre Lemire-Paquin ou Amar Ali-Bey.

1 Bases de données (Datasets)

Les datasets (bases de données) fournis ici proviennent tous du site UCI [1] (un dépôt de datasets d'apprentissage machine). Sous la catégorie classification, nous avons sélectionné trois datasets qui, selon nous, donneront une diversité de résultats avec les méthodes d'apprentissage automatique sélectionnées.

Note : tous ces datasets sont fournis dans le dossier `Code/datasets` en attaché.

1.1 Classification des Iris (Iris Dataset)

Ce dataset [2] classe les fleurs en trois différentes classes d'iris en fonction de la taille des différentes caractéristiques (longueur du sépale, largeur du sépale, longueur du pétale et largeur du pétale). Il s'agit d'un problème de classification simple et classique. C'est un mélange de classes linéairement séparables et inséparables.

Quelques caractéristiques sur le dataset

Le fichier contenant le dataset s'appelle `bezdekIris.data` et se trouve dans ce répertoire [3].

— Nombre d'instances (entrainement et test) : 150 (50 instance dans chaque classe)

1. http://archive.ics.uci.edu/ml/datasets.html
2. https://archive.ics.uci.edu/ml/datasets/Iris
3. http://archive.ics.uci.edu/ml/machine-learning-databases/iris/

— Nombre d'attributs pour chaque instance : 4 numériques, et un définissant la classe à laquelle appartient l'instance (étiquette).
— **Détails sur les attributs :** chaque ligne dans le fichier du dataset représente une instance sous la forme : `<attr1>,<attr2>,<attr3>,<attr4>,<class>`
  — `attr1` : longueur du sépale (en cm)
  — `attr2` : largeur du sépale (en cm)
  — `attr3` : longueur du pétale (en cm)
  — `attr4` : largeur du pétale (en cm)
  — `class` : on en distingue 3 types
    — `Iris-setosa`
    — `Iris-versicolour`
    — `Iris-virginica`

1.2 Base pour des problèmes MONKS (MONKS Problems Dataset)

Ce dataset [4] contient des attributs arbitraires et utilise un classification binaire (0 et 1 sont utilisés comme étiquettes pour chaque exemple). Il reflète 3 problèmes MONKS difficiles à résoudre, et a été utilisé pour une compétition d'algorithmes d'apprentissage. Assurez vous d'avoir fait vos tests sur les trois problèmes. Ce dataset est un peu plus difficile, ne vous attendez pas à de très bonnes performances.

Quelques caractéristiques sur le dataset

Les fichiers d'entraînement et de test sont respectivement `monks-i.train` et `monks-i.test (i = 1,2,3)`, et se trouvent dans ce répertoire [5].

— Nombre d'instances (entraînement et test) : 432
— Nombre d'attributs dans chaque instance : 8 y compris le label (l'étiquette de la classe 0 ou 1)
— Détails sur les attributs : Chaque ligne dans le dataset représente une instance sous la forme : `<class> <attr1> <attr2> <attr3> <attr4><attr5> <attr6> <Id>`
  — `class` : $\{0, 1\}$
  — `attr1` : $\{1, 2, 3\}$
  — `attr2` : $\{1, 2, 3\}$
  — `attr3` : $\{1, 2\}$
  — `attr4` : $\{1, 2, 3\}$
  — `attr5` : $\{1, 2, 3, 4\}$
  — `attr6` : $\{1, 2\}$
  — `Id` : un nom unique pour chaque instance

Notez que certains attributs ont des domaines de valeurs différents. Pour plus de détails sur le dataset, visitez ce lien [6]

4. https://archive.ics.uci.edu/ml/datasets/MONK's+Problems
5. http://archive.ics.uci.edu/ml/machine-learning-databases/monks-problems/
6. http://archive.ics.uci.edu/ml/machine-learning-databases/monks-problems/monks.names

1.3 Base d'enregistrements de votes au Congès (Congressional Voting Records Dataset)

Ce dataset [7] classe les membres du Congrès en Démocrates et Républicains en fonction de leur dossier de vote. C'est un problème un peu plus facile à résoudre, et vous devriez atteindre des performances assez élevées. Il s'agit d'un bon dataset à tester pour le débogage. Il vous revient de regarder de plus près et d'analyser son contenu.

Remarque : il est important de noter que le symbole « **?**» dans ce dataset ne signifie pas que la valeur de l'attribut est inconnue ou manquante, il signifie simplement que la valeur n'est pas «oui» ou «non», voir plus de détails ici [8].

2 Techniques d'apprentissage automatique à développer

Cette section décrit les techniques d'apprentissage automatique que vous aurez à implémenter pour mettre en œuvre et tester les datasets fournis.
En Apprentissage Automatique, plusieurs termes sont utilisés pour évaluer les performances des différentes techniques. Nous trouvons par exemple l'*Accuracy* (exactitude) qui est tout simplement le pourcentage des exemples bien classifiés par rapport au nombre total des exemples présentés à l'algorithme.

1. Faites une recherche et définissez les termes suivants :
   — La matrice de confusion (Confusion matrix) pour la classification
   — La précision (Precision)
   — Le rappel (Recall)

Remarques :
— Vous pouvez voir sur Wikipedia. C'est à vous de faire un petit résumé et d'inclure dans votre rapport une définition de chaque terme.
— En général, les termes Précision et Rappel sont définis pour les problèmes de classification binaire. Pour le dataset Iris, la classification n'est pas binaire (les instances sont de 3 différentes classes). Dans ce cas il convient de calculer la Précision et le Rappel séparément pour chaque classe (une approche de un-contre-tous).
— Pensez à utilisez la matrice de confusion pour vous aider à déduire la Précision et le Rappel.

2.1 K plus proches voisins (K-nearest neighbors)

Pour cette méthode vous devez implémenter la technique des K *plus proches voisins* introduite au *chapitre 18.8.1* et très bien expliquée sur Wikipédia [9].

1. Mentionnez la métrique de distance que vous avez choisie, et justifiez votre choix ;
2. Donnez le pseudo-code (et l'initialisation des paramètres s'il y a lieux) de l'algorithme que vous avez implémenté.
3. Donnez alors :

7. https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records
8. https://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/house-votes-84.names
9. https://fr.wikipedia.org/wiki/Recherche_des_plus_proches_voisins

— L'exactitude (Accuracy)
— La matrice de confusion (Confusion matrix)
— La précision (Precision)
— Le rappel (Recall)

4. Pour le choix de K :
   — **IFT-4102 seulement :** utilisez $K = 5$ (vous avez aussi la possibilité de tester avec d'autres valeurs de K).
   — **IFT-7025 seulement :** utilisez la validation croisée (cross-validation) pour déterminer la *meilleure valeur* de K. Expliquez votre démarche pour la validation croisée et détaillez, dans la mesure du possible, votre démarche.
   Pour la validation croisée et pour chacun des K variant entre $K_{Min}, \ldots, K_{Max}$, on divise l'échantillon original en L échantillons, puis on sélectionne un des L échantillons comme ensemble de validation et les $(L-1)$ autres échantillons constitueront l'ensemble d'apprentissage. On calcule alors l'erreur. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les $(L-1)$ échantillons qui n'ont pas encore été utilisés pour la validation du modèle. L'opération se répète ainsi L fois (généralement 10 fois) pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation. La moyenne des L erreurs est enfin calculée pour estimer l'erreur de prédiction. On choisit alors le K entre $K_{Min}, \ldots, K_{Max}$ qui donne la plus faible erreur.

2.2 Classification Naïve Bayésienne

Ce type de classification est décrit dans le *chapitre 20.2* du livre, mais aussi sur Wikipédia [10].

1. Donnez le pseudo-code (avec l'initialisation des paramètres) de l'algorithme implémenté.
2. Implémentez ce modèle et faites les tests sur les datasets.
3. Donnez alors :
   — L'exactitude (Accuracy)
   — La matrice de confusion (Confusion matrix)
   — La précision (Precision)
   — Le rappel (Recall)

3 Directives

Commencez par voir de plus près le code (voir dossier `Code`) qui vous est donné, vous devez compléter les fonctions de lecture (ou chargement) des datasets, ensuite, implémenter les deux techniques en suivant le squelette des classes tel qu'imposé dans les fichier python en attaché.

— Complétez le fichier `load_datasets.py`, cela vous permettra de charger vos datasets.
— Lisez bien le fichier `classifieur.py` pour vous aider à implémenter les deux techniques d'apprentissage machine, nommez le fichier selon la technique : `BayesNaif.py` pour le modèle *Bayésien Naïf* et `Knn.py` pour la technique des *K plus proches voisis*.

10. https://fr.wikipedia.org/wiki/Classification_na%C3%AFve_bay%C3%A9sienne

— Compléter le fichier `entrainer_tester.py` pour lancer l'entraînement et le test de vos techniques, c'est le fichier principal pour l'exécution.

4 Livrables

Le travail doit être rendu dans un dossier compressé (.zip), contenant :

— README : un fichier texte contenant une brève description des classes, la répartition des tâches de travail entre les membres d'équipe, et une explication des difficultés rencontrées dans ce travail.
S'il y a des erreurs ou des bogues dans votre code, mentionnez les, et expliquez les démarches que vous avez prises pour déboguer le code (cela vous permettra d'avoir des points même si votre code n'a pas fonctionné)

— Tout le code que vous avez écrit doit être remis dans le dossier `Code`, vous avez le droit de modifier le code que nous vous avons fournis, mais les noms des méthodes (tels que : `train(...)`, `predict(...)`, `test(...)`, ...etc) doivent rester inchangés

— Un document PDF contenant :
  — Les réponses aux questions
  — Les discussions des résultats obtenus
  — Une comparaison entre les deux techniques d'apprentissage en terme de performances : Temps d'exécution, Accuracy, Precision et Recall. (Faites un tableau récapitulatif).
  — Une conclusion (mentionnez aussi les difficultés rencontrées)