Institut Mines-Télécom
École Nationale Supérieure
des Mines de Saint Étienne

2020 - 2021
MMGD
Data Stream Mining

[1] TP: Introduction à MOA (Massive Online Analysis)

Version préliminaire, merci de signaler toute coquille ou erreur à nida.meddouri@emse.fr

1 Introduction

Ce didacticiel est une introduction de base à **MOA (Massive Online Analysis)** . Ceci est un environnement logiciel pour la mise en œuvre des algorithmes et l'exécution des expérimentations d'apprentissage en ligne à partir de flux de données (data streams) en évolution. Nous supposons que **MOA** est installé sur votre système, sinon, vous pouvez télécharger **MOA**[2].

Démarrez une interface utilisateur graphique pour configurer et exécuter des tâches avec la commande:

java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.gui.GUI

ou en utilisant le script

bin/moa.sh

ou

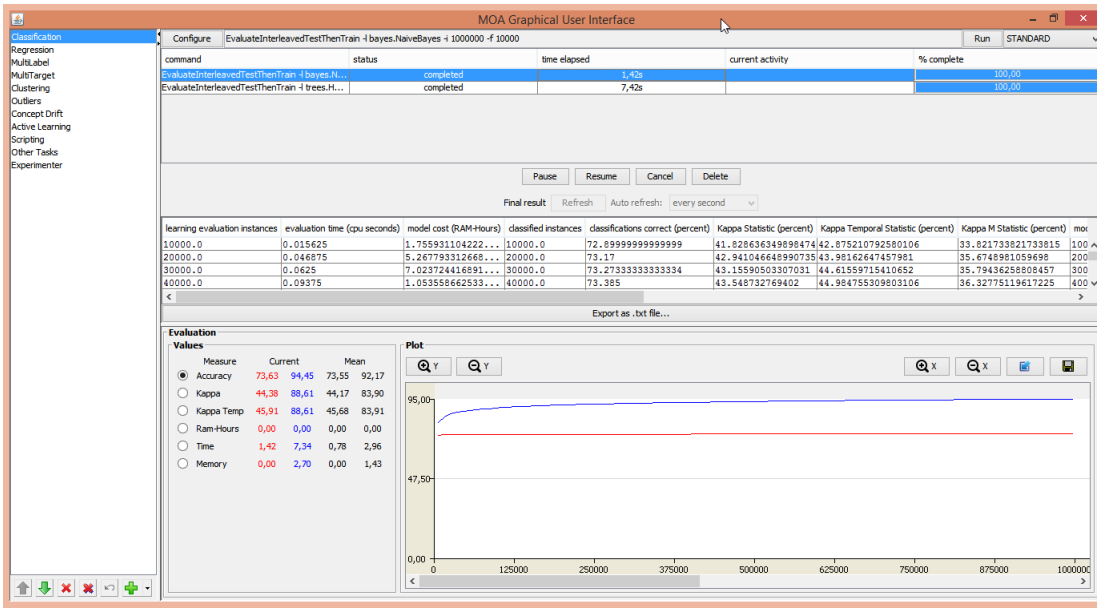
bin/moa.bat

Cliquez sur "Configure" pour configurer une tâche. Lorsque vous êtes prêt, cliquez sur "Run" pour lancer une tâche. Plusieurs tâches peuvent être exécutées simultanément. Cliquez sur différentes tâches dans la liste et contrôlez-les à l'aide des boutons ci-dessous. Si la sortie textuelle d'une tâche est disponible, elle sera affichée au milieu de l'interface graphique et peut être enregistrée sur le disque.

Notez que la zone de texte (text box) de ligne de commande (command line) affichée en haut de la fenêtre, représente des commandes textuelles qui peuvent être utilisées pour exécuter des tâches sur la ligne de commande. Le texte peut être sélectionné puis copié dans le presse-papiers. Au bas de l'interface graphique, il y a un affichage graphique des résultats. Il est possible de comparer les résultats de deux tâches différentes: la tâche en cours est affichée en rouge, et la tâche sélectionnée précédemment est en bleu.

[1] Ce travail demandé est à réaliser en monôme. Il doit être démarré durant la séance de TP, à terminer chez soi pour être remis à votre enseignant avant/à la prochaine séance de TP.

[2] `https://sourceforge.net/projects/moa-datastream/`



2 L'interface utilisateur graphique de classification

Nous commençons par comparer *la précision* (*accuracy*) des deux classificateurs. Tout d'abord, nous expliquons brièvement deux modes différents d'évaluation des flux de données .

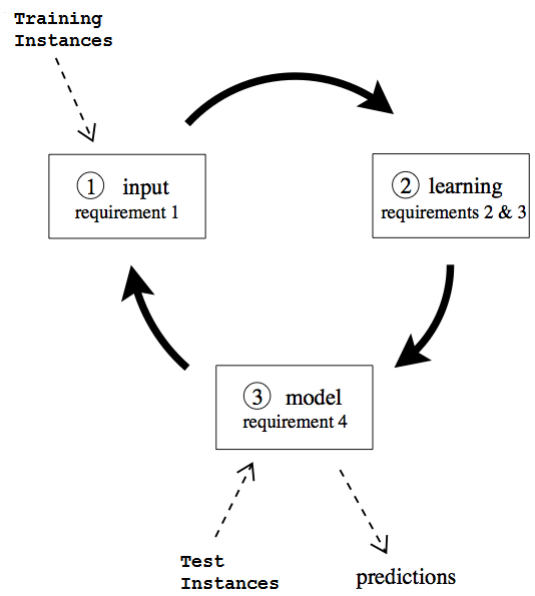
2.1 Évaluation des flux de données

Les conditions requises les plus importantes pour le paramétrage de flux de données, sont les suivantes:

- **Condition Requise 1:** Traiter une instance à la fois et inspecter-la une seule fois (au plus).
- **Condition Requise 2:** Utiliser une quantité limitée de mémoire.
- **Condition Requise 3:** Travailler dans un laps de temps limité.
- **Condition Requise 4:** Être prêt à prédire à tout moment.

Ces conditions requises s'inscrivent dans un cycle répétitif:

1. L'algorithme reçoit la prochaine instance disponible du flux (**Condition Requise 1**).
2. L'algorithme traite l'instance en mettant à jour ses structures de données. Il le fait sans dépasser les limites de mémoire qui lui sont fixées (**Condition Requise 2**), et aussi rapidement que possible (**Condition Requise 3**).



3. L'algorithme est prêt à accepter l'instance suivante. Sur demande, il est capable de prédire la classe d'instances invisibles (**Condition Requise 4**).

Dans le *Batch Learning* traditionnel, le problème des données limitées est surmonté en analysant et en faisant la moyenne de plusieurs modèles, produits avec différents arrangements aléatoires de données d'apprentissage et de test. Dans le cadre du flux des données, le problème des données (effectivement) illimitées pose différents défis. Une solution consiste à prendre des instantanés à différents moments pendant l'induction d'un modèle pour voir dans quelle mesure le modèle s'améliore.

Lors de l'examen de la procédure à utiliser dans le réglage du flux de données, l'une des préoccupations uniques est de savoir comment créer une présentation de la précision (accuracy) au fil du temps. Deux approches principales se présentent:

Holdout : lorsque le *Batch Learning* traditionnel atteint une échelle où la validation croisée prend trop de temps, il est souvent accepté de mesurer les performances sur un seul ensemble de *Holdout*. Ceci est particulièrement utile lorsque la répartition entre l'ensemble de données d'apprentissage et de test a été prédéfinie, de sorte que les résultats de différentes études puissent être directement comparés.

Interleaved Test-Then-Train ou Prequential : Chaque instance individuelle peut être utilisé pour tester le modèle avant qu'il ne soit utilisé pour l'apprentissage, et à partir de là, la précision peut être mise à jour de manière incrémentielle. Lorsqu'il est exécuté intentionnellement dans cet ordre, le modèle

est toujours testé sur des instances qu'il n'a pas vus. Ce schéma présente l'avantage qu'aucun jeu d'*Holdout* n'est nécessaire pour les tests, en utilisant au maximum les données disponibles. Cela garantit également un tracé fluide de la précision au fil du temps, car chaque instance individuelle deviendra de moins en moins significative par rapport à la moyenne globale.

L'évaluation du *Holdout* donne une estimation plus précise de la précision du classifieur sur des données plus récentes. Cependant, il nécessite des données de test récentes qu'il est difficile d'obtenir pour des ensembles de données réels. *Gama et al.* proposent d'utiliser un mécanisme d'oubli pour estimer la précision d'Holdout en utilisant la précision préquentielle: une fenêtre glissante de taille w avec les observations les plus récentes, ou des facteurs d'évanouissement (fading factors) qui pondère les observations en utilisant un facteur de décroissance (decay factor) *alpha* . La sortie des deux mécanismes est très similaire (chaque fenêtre de taille w_0 peut être approximée par un facteur de décroissance $alpha_0$). La classification des flux de données étant un domaine relativement nouveau, ces pratiques d'évaluation ne sont pas aussi bien étudiées et établies qu'elles le sont dans le cadre du *Batch Setting* traditionnel.

Exercice 1. *Comparez la précision de l'arbre Hoeffding avec le classifieur Naive Bayes, pour un flux RandomTreeGenerator de 1 000 000 instances à l'aide de l'évaluation Interleaved Test-Then-Train.*
Utilisez pour toutes les exercices une fréquence d'échantillonnage de 10 000.

Exercice 2. *Comparez et discutez de la précision pour le même flux de l'exercice précédent en utilisant trois évaluations différentes avec un Hoeffding Tree:*

- *Periodic Held Out with 1 000 instances for testing.*
- *Interleaved Test Then Train.*
- *Prequential avec une fenêtre glissante de 1 000 instances.*

3 Générateurs de flux de dérive

Les flux **MOA** sont créés à l'aide de générateurs, en accédant à des fichiers ARFF, en joignant plusieurs flux ou en filtrant les flux. Les générateurs de flux **MOA** permettent de simuler une séquence de données potentiellement infinie. Deux flux évoluant dans le temps sont:

1. Hyperplan rotatif.
2. Random RBF Generator.

Exercice 3. *Comparez la précision de l'arbre de Hoeffding avec le classifieur Naive Bayes, pour un flux RandomRBFGeneratorDrift de 1 000 000 d'instances avec un changement de vitesse de 0,001 à l'aide de l'évaluation Interleaved Test-Then-Train.*

Exercice 4. *Comparez la précision pour le même flux de données de l'exercice précédent en utilisant trois classifieurs différents:*

- *Hoeffding Tree with Majority Class at the leaves.*
- *Hoeffding Adaptive Tree*
- *OzaBagAdwin with 10 HoeffdingTree.*

4 Utilisation de la ligne de commande

Un moyen plus simple pour utiliser la ligne de commande, consiste à copier et coller le texte dans la ligne de configuration de l'interface graphique d'utilisateur . Par exemple, supposons que nous souhaitons traiter la tâche

EvaluatePrequential -l arbres.HoeffdingTree -i 1000000 -w 10000

en utilisant la ligne de commande. Nous écrivons simplement:

java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask "EvaluatePrequential -l trees.HoeffdingTree -i 1000000 -e (WindowClassificationPerformanceEvaluator -w 10000)"

Notez que certains paramètres sont manquants, car ils utilisent des valeurs par défaut.

5 Modèles d'apprentissage et d'évaluation

La classe *moa.DoTask* est la classe principale pour exécuter des tâches sur la ligne de commande. Elle acceptera le nom d'une tâche suivi de tout les paramètres appropriés. La première tâche utilisée est la tâche *LearnModel*. Le paramètre *-l* spécifie l'apprenant, dans ce cas la classe *HoeffdingTree*. Le paramètre *-s* spécifie le flux de données (stream) à partir duquel apprendre, dans ce cas les générateurs. *WaveformGenerator* est spécifié, qui est un générateur de flux de données qui produit un problème d'apprentissage à trois classes d'identification de trois types de *waveforme*. L'option *-m* spécifie le nombre maximum d'instance avec lesquels l'apprenant apprendra, dans ce cas un million d'instances. L'option *-O* spécifie un fichier vers lequel sera rédigé le modèle résultant:

java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask LearnModel -l arbres.HoeffdingTree -s générateurs.WaveformGenerator -m 1000000 -O modèle1.moa

Cela créera un fichier nommé *model1.moa* qui contient le modèle *decision stump* qui a été induit pendant l'apprentissage. L'exemple suivant évaluera le modèle pour voir sa précision sur un ensemble d'instances générés à l'aide d'une graine aléatoire (random seed)vdifférente. La tâche *EvaluateModel* reçoit les paramètres nécessaires pour charger le modèle produit à l'étape précédente, générer un nouveau flux de *waveform* avec une *random seed* de 2 et tester sur un million d'instances:

```
java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask  "EvaluateModel -m
    fichier: model1.moa  -s (générateurs.WaveformGenerator -i 2) -i 1000000 "
```

Ceci est le premier exemple d'imbrication de paramètres à l'aide de crochets (brackets). Des Quaotes ont été ajoutées autour de la description de la tâche, sinon le système d'exploitation peut être confus sur la signification des brackets. Après évaluation, les statistiques suivantes sont générées:

```
classified instances = 1,000,000
classifications correct (percent) = 84.474
Kappa Statistic (percent) = 76.711
```

Notez que les deux étapes ci-dessus peuvent être réalisées en les regroupant en une seule, évitant ainsi de créer un fichier externe, comme suit:

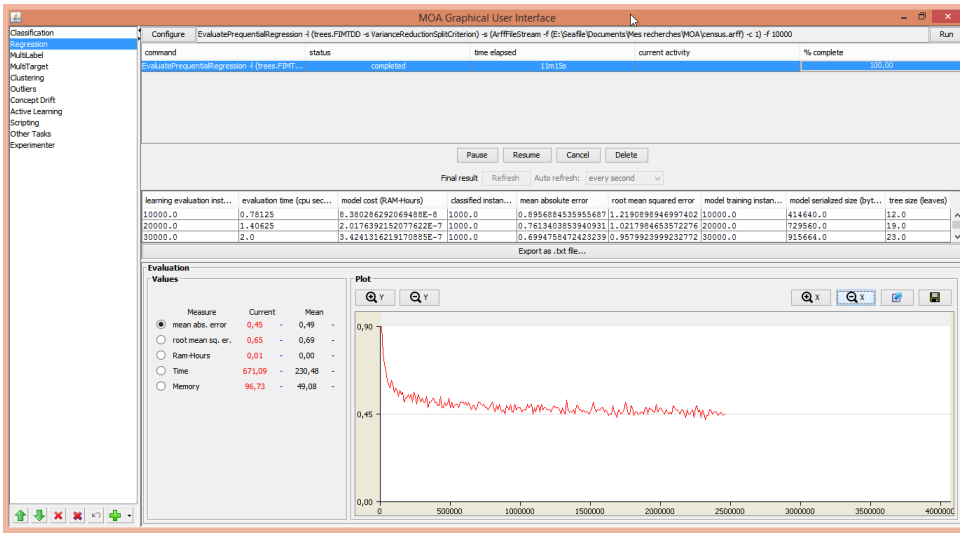
```
java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask  "EvaluateModel -m
    (LearnModel -l arbres.HoeffdingTree  -s générateurs.WaveformGenerator -m
    1000000)  -s (générateurs.WaveformGenerator -i 2) -i 1000000 "
```

La tâche *EvaluatePeriodicHeldOutTest* entraînera un modèle tout en prenant des instantanés des performances à l'aide d'un ensemble de tests suspendu à intervalles réguliers. La commande suivante crée un fichier de valeurs séparées par des virgules, entraînant le classificateur *HoeffdingTree* sur les données *WaveformGenerator*, en utilisant les 100 000 premières instances pour les tests, la formation sur un total de 100 millions d'instances et les tests tous les un million d'instances:

```
java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask
"EvaluatePeriodicHeldOutTest -l arbres.HoeffdingTree  -s
générateurs.WaveformGenerator  -n 100000 -i 10000000 -f 1000000 ">
dsresult.csv
```

Exercice 5. *Répétez les expériences des exercices 1 et 2 en utilisant la ligne de commande.*

Exercice 6. *Comparez la précision et le Ram-Hours nécessaires à l'aide d'une évaluation préquentielle (fenêtre glissante de 1000 instances) de 1 000 000 d'instances pour un flux généré par Random Radius Based Function avec une vitesse de changement de 0,001 en utilisant les méthodes suivantes:*



- *OzaBag with 10 HoeffdingTree.*
- *OzaBagAdwin with 10 HoeffdingTree.*
- *LeveragingBag with 10 HoeffdingTree.*

6 Régression

Pour la régression, revenons à l'interface graphique. Sélectionnez l'onglet *Regression*. Cliquez sur *Configurer* et explorez les options. Vous verrez qu'il y a beaucoup moins de choix que pour la *Classification*. Ce n'est pas nécessairement un problème, car des algorithmes comme la méthode *FIMTDD* basée sur les arbres fonctionnent très bien dans la pratique. Exécutez la configuration par défaut pour une première exploration de l'onglet de régression. Comme nous exploitons ici d'un ensemble de données de classification, tel que généré par *RandomTreeGenerator*, il n'est pas surprenant que les résultats ne soient pas si convaincants. Pour une meilleure expérience, revenez à la configuration, sélectionnez *ArffFileStream* comme source de flux et fournissez un gros fichier *arff* numérique. Vous pouvez utiliser le fichier *census.arff*[3] et spécifier le premier attribut comme cible: *-c 1*. Exécutez à nouveau *FIMTDD*. Si vous voulez un régresseur plus rapide, essayez également le *Perceptron*. Pour des résultats meilleurs, mais une exécution plus lente, expérimentez avec les différents classifieurs de règles (aléatoires).

[3] `http://www.cs.waikato.ac.nz/~bernhard/halifax17/census.arff.gz`