Un système collectif d'utilisation d'un grand ensemble de classifieurs sur le Cloud pour la classification de Big Data

Rabah Mazouzi*, Cyril de Runz**, Herman Akdag*

*LIASD, Université Paris 8, 2 rue de la Liberté - 93526 Saint-Denis cedex
rabah@ai.univ-paris8.fr, akdag@ai.univ-paris8.fr
http://www.ai.univ-paris8.fr/
**CReSTIC, IUT de Reims, Chemin des Rouliers CS30012 51687 REIMS CEDEX 2
cyril.de-runz@univ-reims.fr
http://crestic.univ-reims.fr/

Résumé. Au vu de l'évolution des volumes de données (Big Data) et des problématiques associées (vélocité, variété et véracité), nous proposons dans cet article la conception d'un nouveau système collectif d'utilisation massive d'ensemble de classifieurs pour les Big Data sur le Cloud. Nous combinons les avantages de la labellisation par consensus entre plusieurs décisions de classifieurs distribués sur le Cloud avec l'utilisation du paradigme Map/Reduce pour l'apprentissage des modèles par chacun des classifieurs. Pour cela, nous considérons un réseau de classifieurs déployé sur le Cloud. Par l'intermédiaire des Mappers, nous répartissons les données d'apprentissage sur les différents nœuds (classifieurs) tandis que les Reducers lancent la phase d'apprentissage et retourne le modèle du classifieur ainsi qu'un indicateur de performance à optimiser. Ensuite, pour chaque donnée qui arrive, quel que soit le nœud du réseau sur lequel elle arrive, le nœud labellise la donnée et demande à ces voisins d'en faire tout autant. Ils forment ainsi un ensemble de classifieurs. Enfin, à l'aide d'un vote majoritaire pondéré, le nœud questionné renvoie la décision finale. Ainsi, plus le voisinage est étendu, plus la performance cherchée s'améliore. Cependant, il faut limiter cette extension car sinon nous n'obtenons plus des temps de traitements compatibles avec les Big Data.

1 Introduction

Généralement, les algorithmes de classification utilisent, pour la phase d'apprentissage, des ensembles de données limités en taille et en nature. La problématique de la classification prend une autre dimension avec des données très volumineuses (Big Data), notamment à cause du volume et de la variété des données, ainsi que de la vitesse de réponse du système. Pour pallier aux problèmes liés à la classification des Big Data, le partitionnement des données sur un nombre élevé de classifieurs de nature diverse, constitue, selon nous, une solution idéale.

De nos jours, de nombreuses ressources sont disponibles et mises à disposition dans l'objectif de mettre en place des solutions autrefois très coûteuses et peu accessibles. Ainsi, le

développement du Cloud Computing a grandement facilité la construction de systèmes répartis, supportant des solutions distribuées et collaboratives. Ces systèmes ouvrent de nouvelles perspectives pour ce qui est de l'apprentissage automatique et notamment de la classification.

La classification distribuée, ou plus généralement la fouille de données distribuée ou DDM (*Distributed Data Mining*), ne se limite pas seulement aux faits de réaliser des gains en temps d'exécution, mais ouvre aussi des horizons en matière d'amélioration de la précision de calcul (défi de la véracité), de la scalabilité et de la capacité à traiter des données très volumineuses (Big Data).

De nombreuses recherches montrent que l'approche collective d'un système de classification améliore la qualité des résultats (Dietterich, 2000; Zouari, 2004). Cette approche trouve son implémentation idéale dans l'architecture totalement distribuée, sans entité centrale et sans hiérarchisation (à la manière des réseaux P2P ou un nœud joue à la fois le rôle du client et celui du serveur). Dans cet article, nous essayons de mettre en exergue certains bénéfices de l'utilisation d'une telle approche, notamment pour améliorer la classification de données en précision et en scalabilité.

Plusieurs approches basées Cloud Computing et/ou multi-agents ont récemment été utilisées dans divers domaines, où la mise en place d'une décision collective au sein du système conduit à l'amélioration de la pertinence des résultats globaux. On les trouve notamment dans le cas où le système est naturellement réparti, tel qu'en sécurité des réseaux, où des systèmes de détection d'intrusion distribués sont proposés (Zhou et al., 2010). C'est dans ce contexte que nous plaçons notre démarche.

Nous proposons d'utiliser la distribution des données et des traitements afin de réaliser un gain considérable en temps de calcul et de ressources utilisées. Nous souhaitons ainsi tendre vers le traitement de très grands volumes de données (Big Data). Pour ce faire, nous combinons l'utilisation du Cloud Computing et le paradigme Map/Reduce (Gillick et al., 2006), dont l'objectif est de montrer l'impact de l'utilisation des classifieurs massifs sur la qualité des résultats produits par un système Multi-Classifieurs. Map/Reduce est un patron de conception, ayant connu un grand succès, largement utilisé comme support de mise en œuvre pour la distribution de traitement et de données (Gillick et al., 2006).

L'objectif de cet article, qui est une extension de Mazouzi et al. (2014), est de proposer une spécification fonctionnelle et technique d'un système collectif de classification, qui prend en charge le Big Data et essaie d'en tirer profit afin d'améliorer la performance de la classification. Notre système utilise de manière massive différents classifieurs adaptés aux données à traiter, dans notre cadre des données multivariées et hétérogènes. Ainsi, nous prenons comme premier principe que la variété des Big Data est gérée par les méthodes de classifications exploitées dans notre système.

Nous partons de l'idée que dans le cas de l'apprentissage avec des données distribuées sur plusieurs classifieurs, le modèle sous-jacent de données est réparti sur l'ensemble de ces classifieurs, et de ce fait, le résultat obtenu par la combinaison des prédictions des différents classifieurs est meilleur que ceux de tous les classifieurs pris séparément. Ce travail porte sur la question de la véracité dans les traitements associés aux Big Data et sur la manière d'optimiser un indicateur de celle-ci.

Cependant, l'utilisation massive de classifieurs pose certaines questions : quelle méthode doit-on utiliser pour combiner les résultats ? Quelles techniques peut-on exploiter afin de pallier aux problèmes de temps et de ressource ? Et, surtout, quel est l'impact d'une telle approche sur

la précision des résultats ? Dans le but de répondre à ces questions, nous proposons dans cet article une approche exploitant à la fois la dynamique Map/Reduce et le Cloud Computing dans le contexte des Big Data. Les problématiques de la vélocité et de la volumétrie sont traitées lors de la phase d'apprentissage par l'intermédiaire du support Map/Reduce et lors de la phase de décision par celui du Cloud. Nous mettrons en lumière l'intérêt de notre approche à l'aide d'une simulation sur un jeu de données de référence (KDD Cup 1999).

La suite de cet article est organisée comme suit. La section 2 présente des travaux connexes en classification distribuée, et collective de données massives. Ensuite, dans la section 3, nous décrivons notre système de classification distribuée et consensuelle. Puis, nous exposons, dans la section 4, une spécification technique possible de mise en œuvre de notre système et une implémentation de test. Enfin nous proposerons nos conclusions et perspectives.

2 Exemples de travaux connexes

2.1 Classification et Big Data

Pour classifier les Big Data, Suthaharan (2014) a eu recours aux outils mathématiques et statistiques pour effectuer une analyse préliminaire afin de déterminer les caractéristiques (volume, variété et vélocité) des données et les représenter dans un espace 3D défini sur la base de trois nouveaux paramètres : cardinalité, continuité et complexité. En se basant sur cette représentation, les auteurs ont utilisé des modèles d'apprentissage continu (Machine Lifelong Learning) pour s'adapter aux différentes caractéristiques de données en entrée du système. Cependant, leur méthode ne précise pas de topologie claire du réseau et ni ne démontre sa capacité à passer à l'échelle.

Dans une autre approche, Angiulli et Folino (2007) utilisent une version distribuée de l'algorithme du plus proche voisin (PFCNN) pour extraire des sous-ensembles condensés et représentatifs des Big Data pour construire des classifieurs performants, leur travail se focalise plus sur les gains en mémoire et en CPU, mais il ne traite que peu la précision. Or cette problématique est cruciale, car la qualité de l'analyse et de la prise de décision dépend grandement de la qualité de l'information exploitée. La précision est un indicateur important de cette qualité. Augmenter la précision permet d'avoir une information plus fiable et minimise le risque d'erreur. En cela, la précision est un indicateur de véracité dans les Big Data. C'est le principal objectif qui a guidé notre travail.

2.2 Classification distribuée

Nous présentons ici quelques travaux ayant traité le problème de l'apprentissage distribué, que ce soit pour la classification automatique, supervisée ou non supervisée, de données. On s'intéresse principalement à ceux qui visent à améliorer la précision de la classification globale obtenue à partir de multiples classifieurs locaux, entraînés individuellement.

Ping Luo et al. ont proposé, dans Luo et al. (2007), une approche collective pour la classification distribuée de données, dans un système P2P (paire à paire ou *peer to peer*). Selon leur approche, chaque paire construit ses propres classifieurs, en utilisant des données locales, et en exécutant l'algorithme d'apprentissage *Pasting bites*. Ensuite, tous les résultats sont combinés, en utilisant la technique du vote majoritaire. Il s'agit d'un protocole de vote distribué, basé sur

l'échange de message entre les paires du réseau. Le modèle de distribution proposé dans ce travail, ne peut être envisagé dans le cas d'un réseau large échelle, étant donné que dans ce genre de réseau, le vote majoritaire de toutes les paires ne peut pas être envisagé.

Une version distribuée de l'algorithme de clustering k-moyennes, dans un environnement P2P a été proposé dans Datta et al. (2009). L'algorithme ne nécessite que l'échange d'information locale. Selon les auteurs, il s'agit du premier algorithme des K-moyennes qui pourrait être appliqué dans le cas d'un réseau large-échelle. Chaque nœud du réseau calcule les centroïdes des clusters, et les échange avec ses voisins. Chaque voisin recalcule ses centroïdes, en utilisant ses données locales, et les centroïdes obtenus de ses voisins. L'algorithme étant asynchrone et les nœuds ne communiquant qu'avec leurs voisins directs, la dynamique globale de décision qui permet l'émergence du clustering final est difficile à appréhender. D'ailleurs, les auteurs exploitent une horloge globale dans le cadre de leur expérimentation et considère le résultat majoritaire après un certain nombre de mises à jour du réseau.

En terme de distribution de données volumineuses d'apprentissage sur un réseau de nœuds, plusieurs approches ont été proposées dans la littérature (Moretti et al., 2008). On distingue quatre méthodes possibles de mise en œuvre de la distribution des données d'apprentissage, et ce en considérant l'emplacement de ces données sur les nœuds du système :

— La méthode "Streaming" : s'applique au cas de sources de données réparties, où la fonction de partitionnement relie simplement chaque source à un classifieur dans le système via un flux, telle qu'une connexion TCP.
— La méthode "Pull" : la fonction de partitionnement lit les données d'apprentissage à partir d'un nœud et écrit les partitions sur ce même nœud. Chacun des classifieurs des autres nœuds importe une partition.
— La méthode "Push" : la fonction de partitionnement lit les données d'un nœud et écrit les partitions directement sur les nœuds distants, où les classifieurs lisent leurs copies en local.
— La méthode "Hybride" : la fonction de partitionnement choisit un ensemble réduit de nœuds intermédiaires rapides, fiables, et d'une capacité suffisante pour écrire les données partitionnées. Lors de l'exécution, chaque nœud lit sa partition à partir de ces nœuds.

Mais dans le cadre du Big Data, une méthode est particulièrement exploitée du fait de sa forte scalabilité : l'approche Map/Reduce. Cette approche utilise une fonction de mapping qui répartit les données et une valeur sur les différents nœuds reducers. L'algorithme de mapping correspond à une opération de type "push". La nouvelle API Map/Reduce autorise le "pull". Nous nous plaçons dans le second cas.

3 Description de notre plateforme d'apprentissage

Contrairement aux systèmes multi-classifieurs à base d'une entité centrale, comme l'approche parallèle de la combinaison de classifieurs (Zouari, 2004), qui nécessite l'existence d'une entité centrale de combinaison pour effectuer le traitement de fusion ou sélection, le système décrit dans cet article utilise des agents autonomes répartis sur un réseau de nœuds. Chaque nœud contient un agent qui représente un classifieur élémentaire. Ce dernier reçoit une portion des données d'apprentissage, et l'utilise pour son propre entraînement. Ces portions proviennent des ensembles volumineux de données, ou bien des sources distribuées et

hétérogènes. Contrairement, à certains travaux ayant utilisé des méthodes à base d'agents (c.f. Saidane et al. (2005)), où tous les agents sont mis en interaction, dans notre cas, seuls les nœuds, qui sont voisins, interagissent entre eux. Ceci permet la construction de systèmes de classification large échelle à l'aide d'ensemble de classifieurs définis dynamiquement.

Nous partons de l'hypothèse que c'est l'interaction et la collaboration au sein du système qui permet l'obtention d'un résultat consensuel meilleur que tous les résultats locaux (individuels) au niveau des entités distribuées.

Les données à classifier seront mises sur un nœud du réseau, où ce dernier les diffuse à ses nœuds voisins. Au fur et à mesure que ces données sont classifiées par les nœuds auxquelles elles aboutissent, elles se propagent pour atteindre tous les nœuds d'un voisinage défini. Ce voisinage peut s'élargir selon un paramètre défini par l'utilisateur (niveau de voisinage). Dans l'absolu, l'utilisateur à la possibilité de faire en sorte que l'ensemble des nœuds du réseau soit activé.

Dans la suite de la section, nous montrons les différents aspects qui relèvent de notre système, notamment l'apprentissage des classifieurs locaux, et comment la classification distribuée et consensuelle est opérée.

3.1 Partitionnement, Apprentissage et statistiques

La capacité de généralisation des classifieurs dans notre système dépend, entre autres, de la méthode utilisée lors du partitionnement des données d'apprentissage. Le choix d'une méthode doit prendre en compte les 4V du Big Data, Ce choix dépend donc de la nature de données, et du domaine d'application dont elles sont issues. Ici, nous avons appliqué une loi de Poisson pour partitionner les données. Cette loi permet un échantillonnage en une seule passe avec un tirage sans remise et une perte minimale dans le jeu de données initiales [1].

3.1.1 Map/Reduce pour l'apprentissage

Pour construire un ensemble $C_1 \dots C_M$ de M classifieurs, nous avons choisi le modèle Map/Reduce. Nous avons, en entrée de notre système, un ensemble de données d'apprentissage réparties. Chaque mapper répartit les données qui lui sont affectés (sous ensemble des données) en une passe selon une loi de Poisson pour définir des bases d'apprentissage différentes mais représentant les structures des données initiales. Pour cela, nous considérons une représentation équiprobable de la taille des sous-ensembles d'apprentissage. Celle-ci peut cependant être définie de manière expérimentale et en suivant d'autres procédés considérant par exemple les informations sur le nombre d'individus et de classes. Ce dernier choix pourrait être intéressant sauf qu'il ne permet pas facilement le traitement de flux qui est l'une de nos perspectives d'exploitation de notre système. *In fine*, nous réduisons les informations aux modèles appris par les méthodes de classification et à leur niveau de performance comme illustré dans la figure 1.

1. Uri Laserson présente une démarche de rééchantillonage basée sur la loi de Poisson en 2013 `http://blog.cloudera.com/blog/2013/02/how-to-resample-from-a-large-data-set-in-parallel-with-r-on-hadoop/` (consulté le 27 aout 2015)

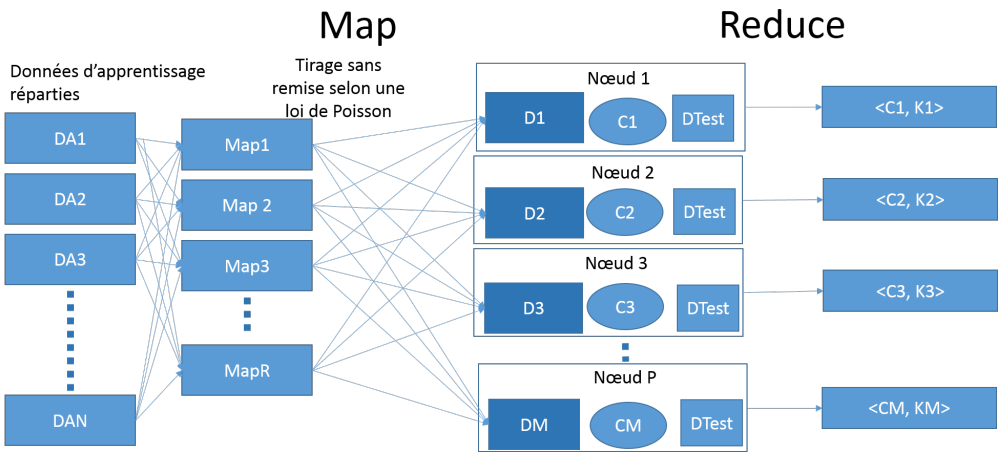

FIG. 1 – *Pipeline de la phase d'apprentissage selon le paradigme Map/Reduce*

3.1.2 Combinaison sur un seul niveau (voisinage proche)

Le pseudo code ci-après présente les fonctions Map et Reduce. Les Mappers (c.f. algorithme 1) sont responsables du partitionnement des données en sous-ensembles $D_1 \dots D_M$, et les Reducers (c.f. algorithme 2) permettent l'entraînement de chaque classifieur C_i à l'aide de la partie D_i ainsi que le calcul de l'indice de performance K_i de chaque C_i. La fonction Poisson renvoie à chaque appel un ensemble d'indices sur les données pour leur sélection afin de définir des sous-ensembles de données en une seule passe sur l'ensemble globale des données d'apprentissage.

Algorithme 1 *Map(.,D)*

D : base d'apprentissage
M : nombre classifieurs
$P \leftarrow (Card(D)/M)$: proportion de données pour chaque classifieur
for each i in M **do**
  $D_i \leftarrow$ Poisson(D,P)
  Envoyer(i, D_i)
end for

Algorithme 2 *Reduce(i,D_i)*

$DTest$: base de test
Apprentissage de C_i avec D_i
Calcul de l'indice de performance K_i de C_i avec $DTest$

Le facteur de performance K_i calculé dans chaque reducer représente le dégrée de confiance (facteur de pondération) lors du vote majoritaire pondéré.

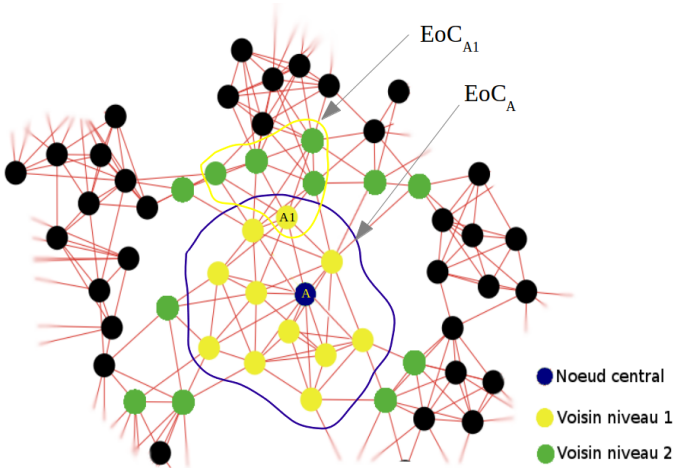

FIG. 2 – *Réseau et niveaux de voisinage*

3.2 La distribution de la labélisation sur le Cloud

Notre système étant composé d'un grand nombre de classifieurs que nous voulons faire coopérer afin d'obtenir une meilleure classification, nous avons décidé de distribuer nos classifieurs sur le Cloud. Le Cloud est ici assimilé à un réseau physique ou virtuel d'entités de calcul. Ce réseau porte sa propre topologie, où celle-ci n'est pas facilement détectable, on peut créer une topologie virtuelle reliant les nœuds. Sur ce réseau, chaque nœud du réseau abrite un agent. On affecte à ce dernier un classifieur avec son degré de confiance. Cette affectation doit aussi préserver une certaine diversité dans chaque voisinage afin d'augmenter les performances de classification (Kuncheva et Whitaker, 2003). Chaque agent doit pouvoir :

— Étiqueter les données avec son classifieur.
— Former un EoC (Ensemble of Classifiers) avec ses voisins et combiner son résultat avec.

3.3 Décision consensuelle

Notre modèle de classification repose sur une logique de combinaison utilisant plusieurs niveaux de voisinage (cf. figure 2).

3.3.1 Combinaison sur un seul niveau (voisinage proche)

Lorsqu'un agent reçoit une donnée, il effectue une première prédiction en utilisant son classifieur, et en même temps il transmet ladite donnée à ses voisins, provoquant ainsi l'activation des classifieurs au niveau de ces derniers. L'ensemble constitue un EoC local où le résultat est combiné par un vote majoritaire pondéré, calculé par la formule suivante Stefano et al. (2002) : pour une donnée d, la classe w_k est la plus probable si :

$$\sum_{i=1}^{M} K_i \cdot d_{i,k} = Max_{j=1}^{N} \sum_{i=1}^{M} K_i \cdot d_{i,j}$$

$d_{i,j} = 1$ si le classifieur C_i étiquette l'attribut avec w_j, 0 sinon,
K_i : facteur de pondération du classifieurs C_i ; M : nombre de classifieurs ; N : nombre de classes.

L'utilisation du vote majoritaire pondéré est ici justifié par le fait que les résultats de nos différents classifieurs sont indépendants, que nos classifieurs sont diversifiés, forment différents modèles avec différents indicateurs de performance et que le système doit retourner un label unique pour chaque donnée à classifier. Dans ce cadre, ce vote conduit à une amélioration de la performance. Bien que des choix plus complexes auraient pu être fait (Kuncheva et al., 2001), de nombreuses études empiriques ont montré que les règles les plus simples à l'instar de celle que nous avons choisie fonctionnent le plus souvent remarquablement bien (Kittler et al., 1998). Nous sommes avec ce choix dans la lignée de la démarche AdaBoost (Freund, 2001). Par ailleurs, avec des classifieurs donnant un ensemble de labels (supports), notre choix aurait naturellement été porté vers un vote de Borda.

3.3.2 Combinaison sur plusieurs niveaux (voisinage lointain)

Nous pouvons élargir notre voisinage (voisins des voisins) en fonction de la capacité des infrastructures sous-jacentes et des contraintes de temps de calcul. Dans ce cas, notre système permet que l'agent voisin (niveau 1) puisse à son tour interroger et combiner son résultat avec ceux de ses voisins (niveau 2), ces derniers pouvant eux-même constituer d'autres EoC (niveau 3) et ainsi de suite. Dans la définition des différents voisinages, les nœuds déjà activés ne sont pas pris en compte dans les EoC suivants : si A questionne B qui questionne son voisinage, B ne questionnera pas A, A ne fera donc pas partie de l'EoC de B. Ce processus permet d'éviter les boucles et limite la congestion du réseau. Le pseudo code 3 illustre le mécanisme récursif de combinaison sur plusieurs niveaux.

Algorithme 3 *Combine(x : attribut, A : agent, $niveau$: entier)*

```
EoC_A = Voisins de A ;
if niveau = 0 then
   x.label = VoteMajoritairePondéré(EoC_A,x) ;
else
   for each agent V_a in EoC_A do
      Combine(x,V_a,niveau − 1)
      EoCV_a = Voisins de A ;
      x.label = VoteMajoritairePondéré(EoCV_a,x) ;
   end for
end if
```

Afin d'expliciter le principe de la classification distribuée et consensuelle, proposée dans cet article, nous montrons un scénario d'interaction entre les classifieurs d'un voisinage pour la détermination collective d'une classe (Figure 3). Soit un nœud A au centre, entouré d'un ensemble V de nœuds voisins : $V = \{B, C, D, E, F\}$. Le classifieur au niveau du nœud A

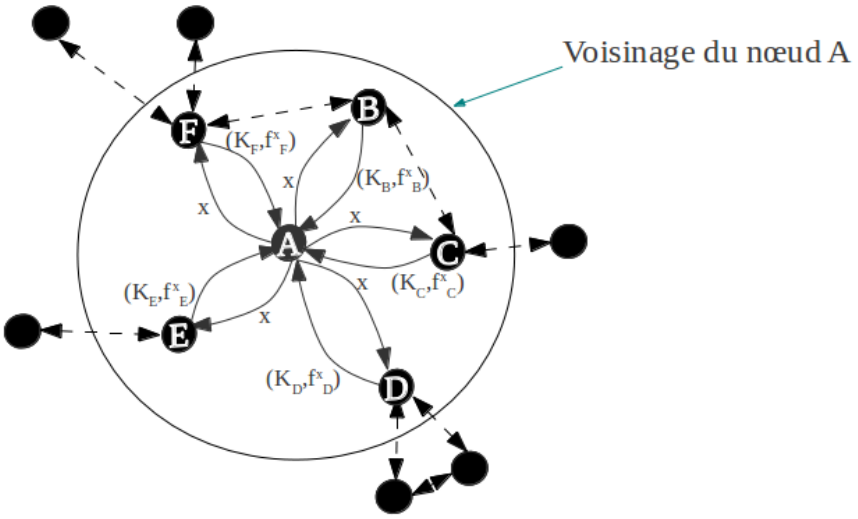

FIG. 3 – *Exemple d'interaction locale au sein du système*

calcule la classe de la donnée en question (x), et il reçoit les choix ($f_B^x, f_C^x, f_D^x, f_E^x, f_F^x$) de tous ses voisins avec leurs facteurs de performances(K_i) ; $i < |V|$. Ensuite, à l'aide d'un vote pondéré, il obtient un label pour la donnée x issu du consensus entre la classe issue de sa propre classification et les classes choisies par ses voisins (f_i^x) dans une logique qui prend en compte les indicateurs de performance de chaque classifieur (K_i).

L'affection de la donnée à classifier à un nœud initial, devenant central dans notre scénario, se fait par l'intermédiaire des services de répartition de charge sur le Cloud.

Il est à noter que notre approche est centrée sur le nœud, et aucune entité centrale n'est utilisée, sauf pour le cas de la distribution de données d'apprentissage qui se fait avant que le processus de labélisation ne commence. Aussi, cette entité se contente de l'assignation des sources de données aux nœuds contenant les classifieurs. Ce caractère de distribution totale, permet ainsi la construction de classifieurs massifs large échelle, qui ne peuvent pas être envisagés avec toute approche nécessitant une entité centrale.

En complément, si l'utilisateur souhaite obtenir la meilleure décision possible, il peut être tenté de faire en sorte que pour chaque donnée, l'ensemble des nœuds du Cloud soit activé. Dans ce cadre, l'utilisateur positionne le système dans une volonté de faire émerger le résultat par la collaboration de l'ensemble des classifieurs. Dans ce cadre, le système mis en œuvre peut alors s'apparenter à une vision système multi-agents.

3.4 Vision système multi-agents large échelle et émergence du résultat

La transposition dans un système multi-agents large échelle est possible. Dans ce cas, on souhaite que pour chaque donnée l'ensemble des agents converge sur la même solution. Pour cela, on peut utiliser la propagation incrémentale par voisinage. De cette manière, dans un

voisinage donné, tous les classifieurs participant à la classification se mettent d'accord (un consensus) sur un choix unique. Ce scénario se répète pour chaque nœud, et cela dans une dynamique permanente. Ainsi, par propagation et échange des résultats, d'un voisinage à un autre, l'émergence d'un résultat final est obtenue.

Un nœud influence le calcul au sein de ses voisins par son propre calcul. Cependant, il est influencé par le calcul des autres. La classe la plus probable est créée dans certains nœuds du réseau. Pour cette classe, les facteurs de performance, sont naturellement les plus hauts. Ces nœuds vont être plus influents que les autres, et la classe qui y est calculée va se propager, et devient de plus en plus dominante dans le système, pour représenter à la fin la classe émergente.

Après un certain temps, la dynamique du système se stabilise et le même résultat est présent dans tous les nœuds du système. L'utilisateur récupère le résultat sur le nœud auquel il a envoyé ses données à classifier. Le problème étant le temps de convergence qui est indéfini, et surtout la possible non convergence du système. Dans ce cadre, nous sommes en train d'étudier des procédés permettant d'assurer cette convergence.

4 Expérimentation

4.1 Dispositif expérimental

Nous avons utilisé un cluster *Hadoop* de 24 Go de ram (16 cœurs à 3.4 ghz) sur lequel on déploie 100 classifieurs engendrés à partir de 5 algorithmes d'apprentissage (J48, oneR, NaiveBayes, JRip, randomForest) implémentés dans la bibliothèque RWeka de l'environnement R Pour leur répartition sur le cluster, nous avons exploité le framework RHadoop. De plus, chaque classifieur est relié à un agent sur le système netLogo. Cette liaison se fait à travers une connexion TCP par le biais de l'extension Rserve. La figure 2 est une représentation graphique de notre déploiement d'agent sur netLogo.

4.2 Jeu de données extrait de KDD CUp 1999

Le jeu de données KDD CUP 1999 Stolfo et al. (2000) a été utilisé pour le concours de la compétition internationale "Third International Knowledge Discovery and Data Mining Tools Competition", qui s'est tenue conjointement à la conférence internationale KDD-99 ("The Fifth International Conference on Knowledge Discovery and Data Mining").

L'objectif de la compétition était de construire un modèle capable de détecter les intrusions sur un ensemble de connexions à un réseau. Le modèle devait être capable de distinguer les mauvaises ("bad") connexions, appelés intrusions ou attaques, des connexions normales ("good").

Chaque connexion du jeu d'apprentissage est labélisée soit comme normale, soit comme attaque (avec le type spécifique d'attaque associée). Chaque connexion enregistrée fait aux alentours de 100 octets. Les connexions sont décrites à l'aide de 41 variables qui sont soit des valeurs entières soit des valeurs catégorielles. Dans Tavallaee et al. (2009), les auteurs proposent une analyse plus détaillée des données de KDD Cup 1999.

L'ensemble de données qu'on a utilisé est tiré de la base KDD Cup 1999 (cf. 4.2). Le rôle des classifieurs est de déterminer si cette connexion est « bonne » ou l'une des quatre sortes « d'attaque ». La base comporte :

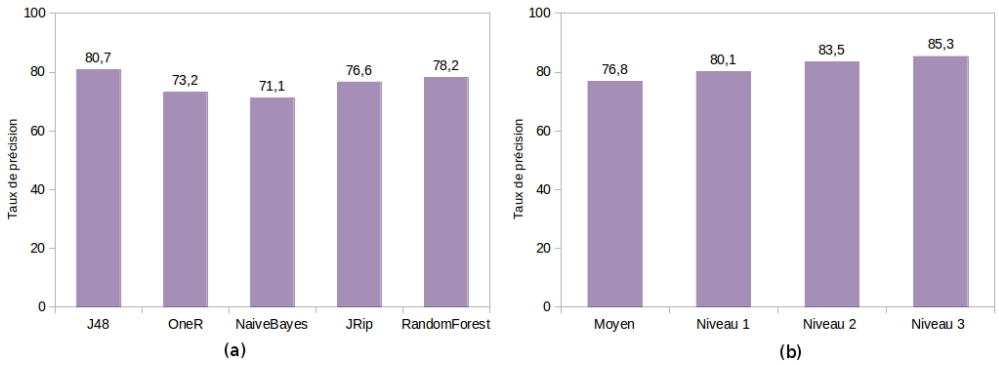

FIG. 4 – *Taux moyens de précision (a) pour chaque type de classifieurs (b) en utilisant 3 niveaux de voisinage et le consensus sur tous les classifieurs*

— KDDTrain : 4 940 000 enregistrements pour l'apprentissage.
— KDDTest : 3 110 290 pour les tests (pour certains modèles nous l'avons réduit à 10000 enregistrement pour garder le calcul réalisable avec la configuration de notre système).

4.3 Résultats avec différents niveaux de voisinages

Outre que le passage à l'échelle qui caractérise notre système, les résultats en termes de précision de classification ont montré une nette amélioration (cf. 4) par rapport aux classifieurs initiaux déployés sur le réseau.

Il est cependant à noter que certaines approches d'apprentissage notamment par ensemble de classifieurs donnent de meilleurs résultats que ceux présentés ici Aburomman et Reaz (2015)). Cependant, ces approches combinent généralement des classifieurs de même type choisis pour leur performance sur le jeu de données. Dans notre approche, nous proposons un système adaptable pouvant ou non être contextualisé, via le choix des méthodes de classification distribuées sur le Cloud, qui permet l'amélioration (ou au pire la conservation) de la précision des résultats.

On a constaté que le temps de calcul pour l'affectation des labels augmente en fonction du nombre de niveau de voisin recherché. En effet, l'activitation des différents niveaux de voisinage provoque, selon la connectivité du réseau, pour chaque nouveau niveau l'activation d'un nombre important et croissant de nœuds et donc de classifieurs.

Par exemple, dans notre expérimentation, le voisinage de niveau 1 permet l'activation d'en moyenne 7 classifieurs, pour environ 35 pour le niveau 2 et 115 pour le niveau 3. Malgré cela, pour les 10000 données tests, par rapport à une activation du voisinage de niveau 1 (voisin direct du nœud), le temps de labellisation avec un voisinage de niveau 2 est de 40% supérieur et celle de niveau 3 est de 70%. En moyenne, la labélisation d'une donnée au niveau 1 prend 0.1 secondes, pour 0,17 pour le niveau 3. Cela est dû au support parallèle où les classifieurs d'un même EoC s'exécutent au même temps.

4.4 Résultats du système orienté émergence globale

Dans l'expérimentation du système orientée émergence globale, nous avons mis 500 agents en interaction pour classifier notre base de test constituée de 10.000 instances. La comparaison des cinq meilleurs taux obtenus par des classifieurs, de type de Bayes Naïves, chacun pris séparément, avec les cinq meilleurs ensembles de classifieurs (5 agents avec leurs voisins directs) et le taux réalisé par le système complexe global donne :

- 84.11% de taux de bonne classification en moyenne pour les 5 meilleurs classifieurs avec un taux maximum de 84,55%,
- 84.88% pour les ensembles de classifieurs avec un maximum de 85,3%,
- 86,85% réalisé par le système global.

Ce résultat vient confirmer notre hypothèse que dans le cadre d'un système complexe avec des interactions « pertinentes » au niveau local, les labels qui émergent au niveau global au sein de ce système sont meilleurs que tous les résultats obtenus par des agents individuels ou des ensembles locaux d'agents. Par contre, le point faible de la version système complexe réside principalement dans le temps de calcul. En effet, le temps de labélisation de l'ensemble de la base de test est, sur notre architecture d'expérimentation, de 7920 secondes, soit près de 4,5 fois plus long qu'une labélisation avec un voisinage de niveau 3, et près de 8 fois plus long que celle à l'aide d'un niveau 1. Le portage de l'expérience sur une réelle architecture Cloud pourraient réduire ce coût, mais cela ne semble pas totalement compatible avec la labélisation de données massives car :

- le temps d'exécution plus long,
- la mobilisation de l'ensemble de l'architecture pour labéliser chaque donnée.

Par ailleurs, comme indiqué précédemment, on trouve des taux nettement meilleurs dans d'autres travaux qui traitent KDDCup99. Cela peut principalement être la conséquence du choix du type de classifieur (bayes naïve) qui a des performances individuelles relativement faibles sur ce jeu de données.

Cependant, le but de nos expérimentations était de confirmer l'amélioration apportée par notre approche à base d'ensemble de classifeurs vis-à-vis des classifieurs pris séparément et de celle cherchant l'émergence du label sur le système globale sur celle à base d'ensemble de classifieurs. Enfin, pour classifier des données massives, il semble plus raisonnable de porter notre choix sur notre architecture à base d'ensemble de classifieurs défini dynamiquement selon le niveau de voisinage et selon le nœud d'entrée des données.

Conclusion

Dans cet article, nous avons proposé une nouvelle approche de classification distribuée et consensuelle de données. L'objectif de notre approche est de tendre à améliorer la véracité dans le traitement des big data. Contrairement à la majorité des approches qui existent dans la littérature, elle se caractérise par son aspect totalement réparti, étant donné qu'aucune entité centrale n'est envisagée dans le système. La classification, se fait d'une manière collective et consensuelle, au sein de chaque voisinage des nœuds du système. La dynamique du voisinage local, possiblement sur plusieurs niveaux, dans le système permet la labellisation de chaque donnée selon la classe ayant le plus fort poids, en considérant les résultats locaux des classifieurs et leurs facteurs de performance.

Ainsi, nous avons, dans cet article, proposé une approche permettant de lever un certain nombre de verrous pour l'utilisation massive de classifieurs sur le Cloud afin de traiter un volume de données très importants. Bien que le dispositif expérimental soit limité, les résultats obtenus sont très encourageants (gain en précision, impact temporel réduit). De plus, la flexibilité de notre approche permet d'envisager son déploiement sur le Cloud, et la distribution des calculs avec l'utilisation du Cloud permet de considérer l'approche scalable pour les Big Data.

Nous envisageons dans un futur proche l'intégration de cette approche dans une plateforme de combinaison de classifieurs orientée Big Data et de continuer à travailler sur la définition d'approches qui utilisent des systèmes modélisés en SMA et d'envisager d'autres formes d'interaction entre agents afin améliorer la classification des données volumineuses, cela en exploitant d'une manière pertinente le support du Cloud Computing. Nos pistes de recherche portent premièrement sur l'adapation de notre système pour la gestion des données évolutives et notamment du concept drift.

Par ailleurs, au sein de tels systèmes, l'étude des différentes interactions entre les classifieurs permet l'émergence de nouveaux algorithmes issus de la combinaison automatique de classifieurs collaborants. Aussi nous souhaitons étudier de manière plus approfondie ces phénomènes. Par exemple, cette approche ne permet pas de faire un ressortir des décisions issues des consensus locaux ou globaux un modèle intelligible ni d'explorer facilement l'espace des résultats. L'étude des différentes interactions pourraient permettre d'obtenir une meilleure compréhension du modèle global de classification obtenu.

Références

Aburomman, A. A. et M. B. I. Reaz (2015). A novel svm-knn-pso ensemble method for intrusion detection system. *Applied Soft Computing*, –.

Angiulli, F. et G. Folino (2007). Distributed nearest neighbor-based condensation of very large data sets. *Knowledge and Data Engineering, IEEE Transactions on 19*(12), 1593–1606.

Datta, S., C. Giannella, et H. Kargupta (2009). Approximate distributed k-means clustering over a peer-to-peer network. *IEEE Trans. Knowl. Data Eng. 21*(10), 1372–1388.

Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, London, UK, UK, pp. 1–15. Springer-Verlag.

Freund, Y. (2001). An adaptive version of the boost by majority algorithm. *Machine Learning 43*(3), 293–318.

Gillick, D., A. Faria, et J. Denero (2006). Mapreduce : Distributed computing for machine learning.

Kittler, J., M. Hatef, R. P. W. Duin, et J. Matas (1998). On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell. 20*(3), 226–239.

Kuncheva, L., J. C. Bezdek, et R. P. W. Duin (2001). Decision templates for multiple classifier fusion : an experimental comparison. *Pattern Recognition 34*(2), 299–314.

Kuncheva, L. I. et C. J. Whitaker (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach. Learn. 51*(2), 181–207.

Luo, P., H. Xiong, K. Lü, et Z. Shi (2007). Distributed classification in peer-to-peer networks. In *KDD*, pp. 968–976.

Mazouzi, R., C. de Runz, et H. Akdag (2014). Système collectif d'utilisation massive de classifieurs pour les big data. In *Société Francophone de Classification*, Rabat, Maroc.

Moretti, C., K. Steinhaeuser, D. Thain, et N. V. Chawla (2008). Scaling up classifiers to cloud computers. In *ICDM*, pp. 472–481. IEEE Computer Society.

Saidane, A., H. Akdag, et I. Truck (2005). Une approche SMA de l'Agrégation et de la Coopération des Classifieurs. In *Conférence Internationale SETIT'2005 Sciences Electroniques, Technologiques de l'Information et des Télécommunication*, pp. 126–126.

Stefano, C. D., A. D. Cioppa, et A. Marcelli (2002). An adaptive weighted majority vote rule for combining multiple classifiers. In *ICPR (2)*, pp. 192–195.

Stolfo, S., W. Fan, W. Lee, A. Prodromidis, et P. Chan (2000). Cost-based modeling for fraud and intrusion detection : results from the jam project. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, Volume 2, pp. 130–144.

Suthaharan, S. (2014). Big data classification : Problems and challenges in network intrusion prediction with machine learning. *SIGMETRICS Perform. Eval. Rev. 41*(4), 70–73.

Tavallaee, M., E. Bagheri, W. Lu, et A. Ghorbani (2009). A detailed analysis of the kdd cup 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pp. 1–6.

Zhou, C., C. Leckie, et S. Karunasekera (2010). A survey of coordinated attacks and collaborative intrusion detection. *Computers - Security 29*(1), 124–140.

Zouari, H. (2004). *Contribution a l'évaluation des méthodes de combinaison parallèle de classifieurs par simulation*. Thèse de doctorat, Université de Rouen.

Summary

Considering the growing volumes of data (Big Data) and the associated issues (velocity, variety and veracity), we propose, in this paper, the design of a new collective system of massive use of set of classifiers for Big Data through the Cloud. We combine the advantages of labeling by consensus between multiple result decisions distributed on the Cloud with the use of the Map/Reduce paradigm for the learning of the models by each of the classifiers. For this, we consider a classifier network deployed through the Cloud. Using mappers, we divide the training data on different nodes (classifiers) while Reducers launch the learning phase and returns the performance index and the model of the classifier. Then, for each datum in input, whatever the network node on which it arrives, the node labels the datum and asks neighbors to do the same. Thus, they form an ensemble of classifiers. Finally, using a weighted majority vote, the questioned node returns the final decision. Larger the neighborhood is, better the quality of results is. However, this extension must be limited because otherwise the time of treatment is not consistent with Big Data.