Nouveau Modèle de Sélection de Caractéristiques basé sur la Théorie des Ensembles Approximatifs pour les Données Massives

Zaineb Chelly Dagdia[*,**] Christine Zarges[*]
Gaël Beck[***], Mustapha Lebbah[***]

[*]Department of Computer Science, Aberystywth University, United Kingdom
[**]LARODEC, Institut Supérieur de Gestion de Tunis, Tunisia
{zaineb.chelly, c.zarges}@aber.ac.uk,
[***]Computer Science Laboratory (LIPN), University Paris-North-13, Villetaneuse, France
{beck, mustapha.lebbah}@lipn.univ-paris13.fr

1 Modèle de Sélection de Caractéristiques

Notre modèle, Sp-RST, défini le problème d'apprentissage des données massives comme un système d'information T_{RDD} où l'univers $U = \{x_1, \ldots, x_N\}$ est l'ensemble des objets, $C = \{c_1, \ldots, c_V\}$ est l'ensemble des caractéristiques conditionnelles parmi lesquelles Sp-RST sélectionne les caractéristiques les plus pertinentes et la caractéristique de décision $D = \{d_1, \ldots, d_W\}$ correspond à la classe. Afin d'assurer la scalabilité de notre algorithme, Sp-RST partage la T_{RDD} en m blocs de données basés sur des partitions de C. Par conséquent, $T_{RDD} = \bigcup_{i=1}^{m}(C_r)T_{RDD_{(i)}}$; où $r \in \{1, \ldots, V\}$. Chaque $T_{RDD_{(i)}}$ est construit en fonction de r caractéristiques aléatoirement sélectionnées à partir de C ; où $\forall T_{RDD_{(i)}} : \nexists\{c_r\} = \bigcap_{i=1}^{m} T_{RDD_{(i)}}$. De ce fait, au lieu d'appliquer Sp-RST (voir Algorithme 1) sur la T_{RDD} comprenant l'ensemble des attributs C, l'algorithme distribué sera appliqué sur chaque $T_{RDD_{(i)}}$.

Nous appliquons l'algorithme N fois sur les m blocs de la T_{RDD}. Plus précisément, à travers toutes les itérations, l'algorithme générera d'abord les m $T_{RDD_{(i)}}$, ensuite pour chaque partition, les instructions distribuées de Sp-RST (Algorithme 1, lignes 5 à 7) seront exécutées à part la ligne 1 de l'algorithme. Cette tâche est indépendante des m partitions générées vu qu'elle calcule la relation d'indiscernabilité de la caractéristique de la décision $IND(D)$ et elle est non liée aux caractéristiques conditionnelles. En dehors de la boucle, ligne 9, le résultat de chaque partition est soit un seul reduct $RED_{i_{(D)}}(C_r)$ ou un ensemble de reducts $RED^{F}_{i_{(D)}}(C_r)$. Si Sp-RST génère un seul reduct, pour une partition $T_{RDD_{(i)}}$ alors la sortie de cette phase de sélection de caractéristiques est l'ensemble des caractéristiques de $RED_{i_{(D)}}(C_r)$. Ces caractéristiques sont les plus pertinentes parmi l'ensemble C_r et résultant a un nouveau système d'information $T_{RDD_{(i)}}$, $T_{RDD_{(i)}}(RED)$, qui préserve quasiment la même qualité des données que $T_{RDD_{(i)}}(C_r)$ qui est basé sur tout l'ensemble des caractéristiques C_r. Si Sp-RST génère une famille de reducts alors l'algorithme choisit aléatoirement un reduct pour représenter la $T_{RDD_{(i)}}$. À ce stade, à chaque bloc de données i correspond un ensemble de caractéristiques sélectionnées $RED_{i_{(D)}}(C_r)$. Cependant, puisque chaque $T_{RDD_{(i)}}$

Algorithm 1 Sp-RST

Inputs : T_{RDD} Système d'information ; m nombre de partitions ; N nombre d'itérations
Output : $Reduct$

1: Calculer $IND(D)$
2: **for** each iteration $n \in [1, \ldots, N]$ **do**
3: Générer $T_{RDD_{(i)}}$ en se basant sur les m partitions
4: **for** each $T_{RDD_{(i)}}$ partition, $i \in [1, \ldots, m]$ **do**
5: Générer $AllComb_{(C_r)}$; Calculer $IND(AllComb_{(C_r)})$
6: Calculer $DEP(AllComb_{(C_r)})$; Séléctionner $DEP_{max}(AllComb_{(C_r)})$
7: Filtrer $DEP_{max}(AllComb_{(C_r)})$; Filtrer $NbF_{min}(DEP_{max}(AllComb_{(C_r)}))$
8: **end for**
9: **for** each $T_{RDD_{(i)}}$ output **do**
10: $Reduct_m = \bigcup_{i=1}^{m} RED_{i_{(D)}}(C_r)$
11: **end for**
12: **end for**
13: **return** ($Reduct = \bigcap_{n=1}^{N} Reduct_m$)

est basé sur des caractéristiques distinctes, une union des caractéristiques sélectionnées est nécessaire pour représenter la T_{RDD} initiale (Algorithme 1, lignes 9 à 11). L'algorithme est itéré N fois générant N $Reduct_m$. Ainsi, à la fin, une intersection de tous les $Reduct_m$ obtenus est nécessaire (Algorithme 1, ligne 13).

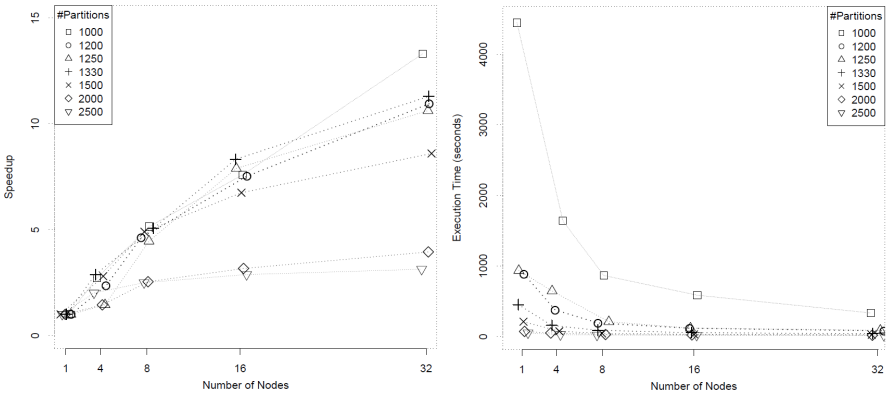

FIG. 1 – *Speedup et le temps d'exécution.*

Pour l'évaluation, nous observons qu'il existe un compromis entre le nombre de partitions et le nombre de noeuds utilisés. Si quelques noeuds sont disponibles, il est conseillé d'utiliser un plus grand nombre de partitions pour réduire le temps d'exécution alors que le nombre de partitions devient moins important si l'on peut accorder un degré élevé de parallélisation. Ce travail s'inscrit dans le cadre d'un projet du programme H2020 de la bourse Marie Sklodowska-Curie, accord Numéro 702527.