Devoir 1

Réalisé par Tristant Lamblot et Yoann Switala

Présenté à Abdenour Bouzouane
Dans le cadre du cours
8INF954 : Forage de données

UQAC
Université du Québec
à Chicoutimi

15 octobre 2016

1 Synthèse

This article introduces the different ways to improve network security towards intrusions of hackers. In order to protect the network from an intrusion, a function has been invented : IDS (Intrusion Detection System). This system is using data mining and probes to find unusual activity or user on the network. It also highlight the issue of FP (False Positives) that detects threats which are not.

It describes the different algorithms used to detect the threats : Decision Tree (DT), ID3 and ID3 with Havrda and Charvat Entropy.

In order to determine which one has the most efficient detection rate and the less FP rate, it has been tested with the KDD-99 dataset (containing single connection vectors, each with a different type of attack). After different situation tests, the article proves that the ID3 algorithm using Havrda and Charvat Entropy instead of Shannon's is more accurate in detection and makes less mistakes creating FPs for 5 types of attacks (Normal, Denial of Service, Remote to User, User to Root and Probing).

2 Modification Id3

L'algorithme Id3 modifié se situe dans l'arborescence de dossier qui constitue weka. Il se situe dans le package "tree". Son code est lisible dans le fichier "Id3Modified.java" qui se situe dans weka/src/-main/java/weka/classifiers/trees/. Les sources de l'algorithme provienne du package Simple Learning Machine.

Pour compiler weka : il suffit de ce positionner dans le dossier weka et d'executer la commande "ant exejar" puis de lancer weka.jar qui se situe dans le dossier weka/dist/.

De plus dans l'interface pour les options pour les classifiers, la gestion du paramètre alpha a été rajouté pour faciliter celle-ci comme indiqué comme bonus de ce devoir.

3 Expérimentation

Dans le cadre de l'expérimentation un échantillonneur a été écris. Il permet de faire une échantillonnage avec remise. Il fonctionne avec tous fichier de type arff (pas seulement pour celui de ce devoir). Ce programme se trouve dans le dossier Echantillonneur. Pour le lancer voir le fichier "commande.txt".

La mémoire RAM de nos ordinateurs étants trop faible, nous avons diminué l'échantillonnage pour pouvoir efféctuer les tests. Les échantillons se trouve dans le dossier "Data". Suite à cette échantillonnage plusieurs échantillons de taille variable ont été faits. Pour nos machines l'échantillon de taille maximum supporté est de 17500 entrées. On retrouve cette échantillon dans le dossier "Data".

L'algorithme a été testé avec différente valeur de alpha. Les resultats pour ces différentes valeurs sont dans le dossier "Result". La plage de test pour alpha était [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. On remarque que les valeurs de test ne bouge pas lorsque alpha bouge. Le taux de succées reste identique et vaut 96,7059. Le reste des taux de succées sont inférieurs à 80. Cependant lorsqu'on éxécute l'algorithme

Id3 avec l'entropie de Shanon on trouve un taux de succées de 96.6387. On remarque donc une différence, certes faible, entre les deux algorithmes.

On en déduit que l'échantillon n'est pas assez grand. En effet dans le papier l'échantillon comportait 49500 entrées. Ainsi les différences entre nos résultats et les résultats théoriques proviennent de la différence entre le nombre d'entrée de chaque échantillon. Ce fait est soutenue par le résultat que l'on obtient en passant un échantillon de 49500 individus dans l'algorithme J48 de Weka qui donne un résultat de 99.4831. un taux de succées supérieur à ceux annoncé par le papier.

4 Conclusion

On remarquera que la valeur optimale de alpha ne peut être calculé dans notre cas. Les résultats obtenus sont un taux de succées de 96.7059 pour tout alpha, ce qui constitue une différence avec l'algorithme Id3.