TP: Détection d'anomalies avec des réseaux de neurones

Alexandre Dey

11 février 2020

1 Consignes

1.1 Introduction

L'objectif du TD est d'implémenter un réseau de neurone auto-encodeur et de trouver une configuration optimisée afin de répondre à un problème de détection d'anomalie.

Le jeu de données fourni est extrait du dataset UNSW-NB15 [1]. Ce dataset orienté cybersécurité regroupe plusieurs attributs relatives à des connexions réseau au sein d'un système d'information. Certaines de ces connexions sont attribuables à des attaques, d'autres à du trafic normal.

Le nombre d'attaques labellisées étant très faible, nous sommes dans un contexte d'apprentissage non-supervisé, et plus spécifiquement de détection d'anomalies (i.e on apprend sur les connexions normales et on considérera les anomalies comme étant des attaques).

Note : Le code doit être inséré au niveau des TODO dans le base code.

1.2 Exercice 1 : Identifier les types des variables

Le dataset contient plusieurs variables de différents types (catégorielle, binaire et numériques) et chacun de ces types doit être géré différemment des autres. Identifier le type de chacune des variables.

1.3 Exercice 2 : Appliquer une méthode de transformation des variables

Chacune des variables doit être transformée en nombres avant de pouvoir être utilisée. Cette méthode de transformation est différente en fonction du type de variables.

1. Source : `https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/`

Pour les variables catégorielles, vous pouvez utiliser *CategoricalEncoder* fourni dans le code.

Note : Certaines des transformations ont des paramètres ajustés en fonction des données. Pensez à trouver ces paramètres sur les données d'entraînement, les sauvegarder et réutiliser les mêmes pour les données de test. Vous pouvez utiliser le module *pickle* pour sauvegarder et recharger ces paramètres.

1.4 Exercice 3 : Implémenter l'auto-encodeur

1. Pour chaque type de variables, définir les premières couches (entrée et embedding)
2. Définir les couches cachées de l'auto-encodeur
3. Définir les couches de sortie et les *loss* spécifiques à chaque type de variable

1.5 Exercice 4 : Normaliser les loss autour d'une échelle cohérente

Pour chaque variable d'entrée, l'amplitude et les valeurs de la fonction coût seront différentes. Trouver un moyen de rendre ces valeurs cohérentes, à la même échelle.

Hint : Il est possible d'utiliser les données d'entraînement pour avoir une idée des valeurs prises par chaque loss

1.6 Exercice 5 : Définir un seuil de décision

Utiliser les méthodes d'évaluation appropriées pour définir le seuil de décision au delà duquel une connexion sera considérée comme anormale. Inclure les visualisations dans le rapport (vous pouvez utiliser *matplotlib*[2] pour créer les visualisations).

Note : Pour cette étape, vous disposez du dataset *evaluation.csv* pour lequel les attaques sont identifiables par leur label (1 pour les attaques, 0 sinon).

1.7 Exercice 6 : Labelliser les données inconnues

Après avoir construit un réseau dont les performances sont satisfaisantes, donnez une prédiction pour le label des données du dataset *unknown.csv*. Joindre au rapport un CSV avec trois colonnes :

1. score : le score d'anomalie en sortie de l'autoencodeur
2. pred_label : le label prédit (1 anomalie, 0 normal)
3. hidden_label : fournit dans le jeux d'entrée

2. Voir : `https://matplotlib.org/`