Télécom ParisTech

Projet de filière SR2I

Détection d'anomalies de classification dans l'IoT via Machine Learning

Antoine Urban, Yohan Chalier

encadré par
Jean-Philippe Monteuuis
Houda Labiod

16 juin 2018

Résumé

Dans le contexte de l'internet des objets, les messages peuvent contenir de fausses informations générées par un utilisateur authentifié. Par conséquence, les mécanismes de sécurité reposant sur le chiffrement et la signature sont inutiles face à ces attaques. Les algorithmes reposant sur l'apprentissage machine (Machine Learning) permet de classer si un objet émettant des données est malicieux ou non

I Introduction

L'exercice de la conduite est synonyme de changements constant. Ainsi, lorsque nous conduisons, notre attention est tout entière consacré à notre environnement, car notre sécurité est celle des personnes qui nous entourent sont en jeu. Nous prêtons ainsi naturellement une attention toute particulière aux obstacles qui peuvent surgir, qu'il s'agisse tout simplement d'autres voitures partageant la route, des piétons ou encore des motocyclistes. Longtemps objets fantasmés, les véhicules autonomes deviennent aujourd'hui réalités grâce aux incroyables progrès réalisés dans les domaines de l'intelligence artificielle et du développement de capteurs. Si ces processus visent à remplacer le conducteur humain, il semble naturel qu'ils prêtent la même attention aux obstacles. Les méthodes utilisées se doivent d'être fiables et performante, car en matière de sécurité, le droit à l'erreur n'existe pas !

Dans ce travail, nous nous concentrerons sur les 3 types d'obstacles les plus communs sur une route :

— Une voiture ;
— Une moto ;
— Un piéton.

Tout travail sur l'analyse d'obstacles nécessite la détection préalable de régions d'intérêt en utilisant par exemple la méthode du "point-in-polygon"[1], ou encore à l'aide de capteurs à ultrasons et d'analyse de signal[**?**]. Ce travail n'a pas comme objectif de revenir sur l'utilisation de ces techniques, mais se concentrera sur l'analyse à posteriori des dimensions ainsi collectées avec pour but de proposer une classification efficace.

Objectifs

A travers un cas simple, nous cherchons à vérifier l'appartenance d'un objet à une classe (e.g. un véhicule) en utilisant un algorithme de machine learning et les dimensions de l'objet. Il s'agit donc de proposer un modèle de classification multi-classes en réalisant un classeur à partir d'un algorithme d'apprentissage supervisé. Au préalable, nous nous assurons de trouver des bases de données avec les dimensions pour chacune des classes pour entraîner et valider notre modèle. Ensuite, nous proposerons une méthodologie de sélection d'un algorithme d'apprentissage supervisé en évaluant leur précision et leur efficacité respectives. Dans ce travail, nous avons considérés les algorithmes (c) suivant :

— Réseau de Neurones (c = 1) ;
— Adaboost (c = 2) ;
— SVM(c=3) ;
— Random Forest (c=4).

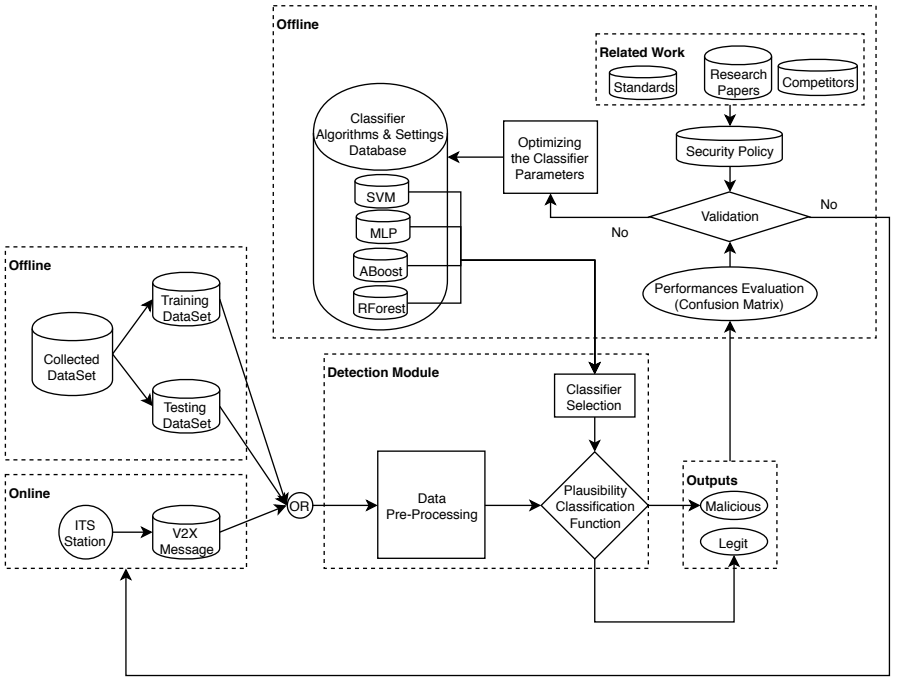
La fonction suivante permet de résumer le déroulé du processus de prédiction :

Algorithm 1: Fonction de production

Donnée : Un message i envoyé par un objet communicant contenant :
- Classe de l'objet(Classe$_i$ = ($Voiture, moto, piéton$))
- Dimension de l'objet (Longueur : L$_i$; $Largeur : l_i$).

Result: Détection des objets malicieux

if *function (Class, Longueur, Largeur)* **then**
  Return "malicieux";
else
  Return "non malicieux";
end



II Méthode d'évaluation

II.1 Matrice de confusion

Une technique pour évaluer la performance d'un algorithme de classification est d'utiliser une matrice de confusion. Il s'agit d'un résumé des résultats de prédiction sur un problème de classification, qui donne un aperçu du degré de confusion de notre modèle.

Terminologie

Pour définir de manière formelle une matrice de confusion, il convient d'introduire les termes suivant :

— TP (Vrai Positif) : item correctement détecté positif
— FP (Faux Positif) : item déclaré positif, là où il est en réalité négatif.
— FN (Faux négatif) : item déclaré négatif alors qu'il était en réalité positif.
— TN (Vrai négatif) : item correctement détecté comme négatif

Ces derniers permettent de définir la classification, correct ou non, des objets données en entrée de notre fonction. Nous pouvons dès lors construire la matrice suivante :

		Actual class: Positive	Actual class: Negative		
Predicted classes	Positive	TP	FP	PPV	FDR
	Negative	FN	TN	FOR	NPV
		TPR	FPR		
		FNR	TNR		

Pour la suite, et pour comprendre cette matrice, il convient d'introduire les résulats suivants :

— Ratio de Vrai positif : TPR= $\frac{TP}{TP+FN}$

— Ratio de Vrai négatif : TNR= $\frac{TN}{TN+FP}$

— Ratio de Vrai positif : TPR= $\frac{TP}{TP+FN}$

— Ratio de Vrai négatif : TNR= $\frac{TN}{TN+FP}$

— Ratio de Faux positif : FPR= $\frac{FP}{TP+TN}$

— Ratio de Faux négatif : FNR= $\frac{FN}{TP+FN}$

— Valeur prédictive positive : PPV= $\frac{TP}{TP+FP}$

— Valeur prédictive négative : NPV= $\frac{TN}{TN+FN}$

— Taux de fausses découvertes : TPR= $\frac{FP}{TP+FP}$

— Taux de fausses omissions : FOR= $\frac{FN}{TN+FN}$

Ce mode de représentation permet non seulement de mettre en évidence les erreurs qui sont faites par votre classificateur, mais surtout des types d'erreurs

qui sont commises. En effet, l'utilisation de la seule précision peut être trompeuse dans la mesure où le nombre d'observation dans chaque classe est inégale. Pour illustrer ce phénomène, prenons un exemple précis. Considérons l'algorithme SVM (support vector Machine) avec les paramètres suivants :

C=0.1, cache–size=200, class–weight=None, coef0=0.0, decision–function–shape='ovr', degree=3, gamma='auto', kernel='linear', max–iter=-1, probability=False, randomState=None, shrinking=True, tol=1e-05, verbose=False

Le taux d'exactitude de cet algorithme avec ces paramètres est égale à 0.96660 et renvoie la matrice de confusion suivante :

$$\begin{bmatrix} 2771 & 74 \\ 239 & 239 \end{bmatrix}$$

Nous remarquons bien que la classification est loin d'être satisfaisante alors que le score d'exactitude est, lui, important. Il y a en effet dans ce dernier aucune pondération. Si une classe contenant un nombre de paramètres important est bien classifiée, elle peut "masquer" les mauvais résultats obtenus pour d'autres classes plus petites.

Score F1

Comme nous l'avons vu, le taux d'exactitude reste trop général pour correctement caractériser la performance d'une classification. Nous allons donc utiliser une autre métrique : le score F1. Pour ce faire, introduisons deux termes :

La précision : la précision est le ratio d'observations positives correctement prédites sur le total des observations positives prédites. On a donc :

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Le taux de rappel : le taux de rappel est le ratio d'observations positives correctement prédites sur le total des observation de cette classe. On a donc :

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Il est souvent intéressant de comparer 2 versions d'un classeur pour déterminer lequel est le plus performant. Une idée intuitive serait de faire la moyenne de la précision et du taux de rappel. Pourtant, dans le cas d'une précision très mauvaise et d'un taux de rappel très élevé, nous pouvons rapidement à une mauvaise estimation de la performance de notre algorithme. C'est pour cette raison que nous introduisons une nouvelle métrique : **le score F1**. elle est défini comme étant la moyenne harmonique de la précision et du taux de rappel. On a donc :

$$\text{F1 Score} = \frac{2 * (Recall * Precision)}{(Recall + Precision)} = 2 * \frac{ppv * tpr}{ppv + tpr} (3)$$

Dans notre cas, nous cherchons à minimiser le nombre de faux positifs et de faux négatifs. L'utilisation du score F1 nous permet de les prendre tous deux en compte.

II.2 SVM

L'algorithme du SVM est initialement définis pour la discrimination d'une variable qualitative binaire. Il est à ce titre particulièrement indiqué dans le cas de problèmes linéairement séparables. Or, nous sommes en présences d'un problème à 3 classes. Nous allons donc devoir adapter la méthode générale. Plusieurs adaptations existent. Parmi ces dernières, nous avons considéré les deux suivantes :

— La méthode **"One-Versus-the-Rest"** est la plus simple et la plus ancienne des méthodes d'otpimisation indirectes. Il s'agit d'une extension au cas multiclasses proposée par Vapnik[3]. Elle consiste à construire p classifieurs binaires (à vecteurs de supports) pour classifier p classes. S'il s'agit d'une extension facile à mettre en place, cette méthode risque d'introduire artificiellement un déséquilibre des classes dans la construction des modèles individuels.

— Méthode directe de **Crammer et Singer**[4] : Contrairement aux schémas de décomposition indirect (comme la méthode 'OVR", cette approche consiste à séparer les classes en résolvant un unique problème d'optimisation.Ainsi, cela revient à résoudre un problème d'optimisation à (p-1).n contraintes (où p est le nombre de classe et n le nombre d'observations).

Comme pour les méthodes précédentes, la table1 liste les paramètres testés

paramètre	ensemble des valeurs testées
`multi_class`	'ovr' , 'crammer_singer'
`C`	$\{10^k \mid k \in [\![-2, 3]\!]\}$
`tol`	$\{10^- k \mid k \in [\![3, 6]\!]\}$

TABLE 1 – Liste des paramètres testés pour SVM

Bibliographie

[1] Deepika, N AND Sajith Variyar, V. V., Obstacle classification and detection for vision based navigation for autonomous driving, Addison Wesley, Massachusetts, IEEEl, 2017.

[2] Gerard Gibbs and Huamin Jia and Irfan Madani, Obstacle Detection with Ultrasonic Sensors and Signal Analysis Metrics, Transportation Research Procedia, 2017.

[3] Vapnik, V. : Statistical Learning Theory. Wiley, New York (1998).

[4] Crammer, K., Singer, Y. : On the algorithmic implementation of multiclass kernel-based vector machines. J. Mach. Learn. Res. 2, 265–292 (2001)