Méthode d'apprentissage pour la classification à partir d'exemples positifs

A single-class learning method for classification

B. Luvison[12] T. Chateau[1] Q.C. Pham[2] P. Sayd[2] J.T. Lapresté[1]

[1] LASMEA, Université Blaise Pascal, Aubière, France
[2] CEA, LIST, LSVE, Boîte Courrier 94, F-91191 Gif-sur-Yvette, France

bertrand.luvison@lasmea.univ-bpclermont.fr

Résumé

Cet article présente une méthode d'apprentissage générique et non supervisée à partir d'une seule base d'exemple positif, pour la classification. Le système est formalisé dans un cadre probabiliste. Nous proposons une méthode originale pour approximer la densité de probabilité de la fonction de vraisemblance correspondant aux évènements dit normaux, en utilisant un modèle parcimonieux basé sur des fonctions noyaux. Ce modèle présente l'avantage des méthodes non-paramétriques tout en limitant le coût algorithmique souvent important qui leur est lié. La classification est ensuite effectuée à partir de cette approximation grâce à une notion de confiance. La méthode sera comparée à celle des One Class SVM et testée dans le cas de la détection d'événements rares liés au trafic routier.

Mots Clef

Machine d'apprentissage, machine à noyau, One Class SVM.

Abstract

This paper presents a generic unsupervised learning method from a single class positive database. The system can be represented into a probabilistic framework. We propose an original method to approximate the likelihood function using a sparse vector machine based model. This model keeps the assets of non parametric models while reducing the computation time of these methods. Classification will be done thanks to a confidence notion. The method will be compared to One Class SVM and tested on a traffic unexpected event detection application.

Keywords

Machine learning, kernel machine, One Class SVM

1 Introduction

La classification est un des problèmes récurrents en vision par ordinateur. Les champs d'application des algorithmes de classification sont on ne peut plus variés, que ce soit pour de l'indexation d'image, que de la reconnaissance d'objets, etc. On peut cependant distinguer deux familles de problèmes. La première concerne la classification de primitives parmi N classes. L'autre famille, qui est une restriction de la première, consiste à n'effectuer la classification qu'entre deux classes. Ce genre de problème vise essentiellement à classifier une primitive expliquée par le modèle appris ou non, comme étant normale ou non. Sous cette hypothèse supplémentaire, d'autres types de machines d'apprentissage peuvent être utilisée, comme par exemple, les machines de type SVM (machine à vecteurs support)[3], RVM (machine à vecteurs de pertinence)[15] ou encore le Boosting [5]. C'est cette dernière famille de problème binaire qui nous intéresse ici.

Pour illustrer l'utilité de ces machines d'apprentissage, on peut donner des exemples pour l'analyse de foule (au sens large du terme, à savoir multitude d'objet en un même lieu). La classification pourra, par exemple :

- Servir à définir si la trajectoire d'un individu ressemble à celle préalablement apprise ou si au contraire elle correspond à une trajectoire inconnue et donc potentiellement dangereuse, comme c'est le cas dans [8].
- Ou encore dans [1], où ce sont les composantes principales du champs de déplacement (extraite d'une décomposition SVD) qui sont apprises localement en vue de détecter des variations anormales sur la vraisemblance apprise (e.g perturbations du mouvement de foule due à une chute ou des mouvements de panique).

Pour la surveillance routière, les mouvements à contre-sens sur autoroute peuvent aussi être détecté en apprenant la direction principale de circulation par une mixture de gaussienne (MMG) [10]. D'autres applications pour véhicules embarqués, par exemple, utilisent des machines d'apprentissage de type SVM pour la détection de piétons [12].

Habituellement les machines d'apprentissage binaires utilisent une base d'exemples positifs et d'exemples négatifs comme c'est le cas pour les machines à vecteurs supports (SVM)[3], les machines à vecteurs de pertinence (RVM) [15] et le Boosting [5]. Néanmoins dans certain cas, la

construction de la base d'exemples négatifs est délicate voire impossible, pour deux raisons : (1) ce sont des événements rares et (2) la variabilité intra-classe est importante. On peut citer l'exemple, de la détection de panne de moteur [9]. Les pannes sont des évènements généralement rares et surtout imprévisibles, ce qui rend difficile leur exploitation pour l'apprentissage.

Pour répondre à ce problème, des approches spécifiques sont habituellement utilisées. La première consiste à utiliser des densités de probabilité de type mixture de gaussienne (MMG) pour modéliser les distributions des primitives étudiées. Par exemple, Stauffer et Grimson dans [14] utilisent un nombre fixe de gaussiennes pour modéliser une fonction de vraisemblance dans une application de soustraction de fond. Les récents algorithmes EM (Expectation Maximization) sont aussi utilisés pour estimer les paramètres des MMG sans hypothèse sur le nombre de gaussiennes [11]. Récemment, Han dans [6] propose une approche pour approximer une distribution par un MMG en s'appuyant sur l'algorithme mean-shift. La seconde approche consiste à utiliser des algorithmes de type SVM spécialement adaptés pour ces problèmes, à savoir les algorithmes One Class SVM [13].

C'est pour traiter ces cas de figure que nous proposons une méthode qui n'utilise que des exemples positifs durant l'apprentissage, comme le font les algorithmes One Class SVM. La partie suivante décrit le mécanisme d'apprentissage mis au point ainsi que la méthode de classification. Les performances du système seront évaluées sur la qualité du modèle appris dans la partie 3 et sur les résultats obtenus en conditions réelles dans la partie 4.

2 Méthode Proposée

2.1 Fonction de vraisemblance

Soit l'ensemble des primitives constituant la base d'apprentissasge $\boldsymbol{Z} \doteq (z_1, z_2, ..., z_K)^T$.

Nous définissons une variable aléatoire de Bernouilli $\omega_k \in \{\omega_1; \omega_2\}$ pouvant prendre les deux états : $\omega_k = \omega_1$ si la primitive définit un état possible du modèle et $\omega_k = \omega_2$ si elle ne le définit pas. La fonction de vraisemblance $P(z_k|\omega_2)$ étant complètement inconnue, en raison de la rareté des observations, toute primitive mal expliquée par la fonction de vraisemblance $P(z_k|\omega_1)$ est considérée comme appartenant à la classe ω_2. En pratique, les primitives pour lequelles la fonction de vraisemblance répond en dessous d'un certain seuil C, sont considérées comme inconnue du modèle.

$$\widehat{\omega} = \omega_2 \; si \; P(z_k|\omega_1) < C \tag{1}$$

La fonction de vraisemblance $P(z_k|\omega_1)$ peut être représentée soit par modèle paramétrique ou non.

Dans le premier cas, les MMG sont souvent utilisés pour approximer la vraisemblance. Les modèles non paramétiques, quant à eux, approximent une distribution à partir d'exemples en utilisant une estimation basée sur des fonctions à noyaux (modèle à fênetre de Parzen [4]) :

$$\widehat{P}(z_k|\omega_1) = K^{-1} \sum_{k'=1}^{K} \phi_{k'}(z_k) \tag{2}$$

où les $\phi_{k'}$ sont des fonctions noyaux. Néanmoins, cet estimateur de densité de probabilité a un coût algorithmique proportionnel au nombre de points de l'ensemble d'apprentissage (taille de $\boldsymbol{Z}$). Nous proposons d'approximer cet estimateur à l'aide d'un modèle épars obtenu à partir de l'équation (2) par une somme pondérée de fonctions noyaux.

2.2 Approximation de l'estimateur de vraisemblance non paramétrique

L'équation 2 peut aussi être exprimée de manière réduite sous la forme :

$$\widehat{P}(z_k|\omega_1) \approx \boldsymbol{w}^T \left(\boldsymbol{\phi}(z_k)\right) \tag{3}$$

où $\boldsymbol{w}^T = (1, ..., 1)^T/K$ est un vecteur de taille K et $\boldsymbol{\phi}(z_k)$ une fonction vectorielle définie par $\boldsymbol{\phi}(z_k) = (\phi_1(z_k), \phi_2(z_k), ..., \phi_K(z_k))$. Ce modèle a la même forme que les modèles utilisés pour les SVM [3] ou les RVM [15]. Obtenir un modèle épars à partir de l'équation 3 est réalisé en fixant la plupart des paramètres de $\boldsymbol{w}$ à zéro. Notons Φ, la matrice de design de taille $K \times K$ associée au bloc n et construisons la, telle que l'élément de la ligne i et de la colonne j soit $\Phi_{i,j} = \phi_i(z_j)$. Φ est une matrice carrée, symétrique, pour laquelle un estimateur de la vraisemblance associé à l'élément z_k de l'ensemble d'apprentissage est donné par :

$$\widehat{P}(z_k|\omega_1) = K^{-1} \sum_{k'=1}^{K} \Phi_{k,k'} \tag{4}$$

Un vecteur de vraisemblance $\boldsymbol{\varphi}$ en relation avec l'ensemble d'apprentissage est construit à partir des estimateurs :

$$\boldsymbol{\varphi} = \left(\widehat{P}(z_1|\omega_1), ..., \widehat{P}(z_K|\omega_1)\right)^T \tag{5}$$

Le modèle épars est construit via une méthode dans laquelle on sélectionne de manière récursive des vecteurs pertinents de l'ensemble d'apprentissage $\boldsymbol{Z}$. Cette dernière se décompose comme illustrée dans l'algorithme 1. La notation $\mathbf{1}_K$ représente un vecteur de longueur K contenant uniquement des valeurs 1. La sortie de l'algorithme est un ensemble de vecteur pertinents et leurs poids associés. Le paramètre t_v représente la précision de l'approximation de la vraisemblance. Pour une approximation grossière, t_v doit être grand. Dans ce cas le nombre de vecteurs utilisés diminue. Des illustrations de l'effet de ce paramètre sont données en section 3.1.

Finalement, l'estimateur peut s'écrire de la façon suivante :

$$\widehat{P}(z_k|\omega_1) \approx \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{\phi}}(z_k) \tag{6}$$

avec

$$\tilde{\boldsymbol{\phi}}(z_k) = \left(\phi_{v(1)}(z_k), \phi_{v(2)}(z_k), ..., \phi_{v(M)}(z_k)\right)^T \tag{7}$$

Algorithme 1 Algorithme d'approximation d'un estimateur non paramétrique

ENTRÉES: matrice de design Φ, critère d'arrêt t_v

Calcul du vecteur de vraisemblance initial :

$$\boldsymbol{\varphi}_1 = K^{-1}{}_{\Phi}\, \mathbf{1}_K$$

Initialisation : $m = 0$

répéter

$m = m + 1$

Extraction du point de vraisemblance maximale :

$$v(m) = \arg\max_i \varphi_{m,i}$$

Calcul du poids associé : $\tilde{w}_m = \frac{\varphi_{m,v(m)}}{\Phi_{v(m),v(m)}}$

Mise à jour du vecteur de vraisemblance :

$$\boldsymbol{\varphi}_{m+1} = \boldsymbol{\varphi}_m - \tilde{w}_m\, \boldsymbol{\phi}\left(z_{v(m)}\right)$$

jusqu'à $\varphi_{m,v(m)}/\varphi_{1,v(1)} < t_v$

$M = m$

retourne Le vecteur des poids : $\tilde{\boldsymbol{w}}^T = (\tilde{w}_1, ..., \tilde{w}_M)$ et les points sélectionnés : $\widetilde{\boldsymbol{Z}} = (z_{v(1)}, z_{v(2)}, ..., z_{v(M)})$

Cette approche permet d'approximer la vraisemblance à partir de quelques vecteurs seulement. Initialement, l'ensemble résultant de l'apprentissage de la méthode non paramétrique $\boldsymbol{Z}$ contient K éléments alors que l'ensemble de la méthode éparse $\widetilde{\boldsymbol{Z}} = (z_{v(1)}, z_{v(2)}, ..., z_{v(M)})$, ne contient plus que M éléments avec $M \ll K$.

2.3 Fonction discriminante

La classification à la volée est basée sur l'équation (1) qui peut être reécrite à l'aide du modèle épars :

$$\widehat{\omega} = \omega_2 \; si \; \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{\phi}}(z) < C \tag{8}$$

Pour un bloc donné, un ensemble d'orientations du flot optique est calculé. L'orientation dominante (z) est obtenue par un maximum de vraisemblance. La vraisemblance de chaque point candidat est calculée par un estimateur non paramétrique dont la forme est identique à celle présentée dans l'équation 3, en utilisant l'ensemble d'orientations du flot optique du bloc.

Le seuil C est calculé en fonction de la distribution de vraisemblance selon une notion de confiance [11]. Intuitivement, le seuil C devra être d'autant plus élevé (la classification d'autant plus restrictive) que la distribution de vraisemblance possède un support restreint. Inversement, une distribution de vraisemblance uniforme est supposée tout modéliser, en conséquence le classifeur devra être laxiste, et le seuil C devra être bas.

Par définition, pour une densité de probabilité p, la confiance est la valeur $k \in [0, 1]$ tel que :

$$\int_{\Omega \setminus R} p(\boldsymbol{x}) d\boldsymbol{x} = k \tag{9}$$

avec $R \subseteq \Omega$ l'intervalle minimale. Cette intervalle R délimite ainsi la région qui concentre la plus forte densité, la région de densité maximale.

Si la résolution d'un tel problème est simple pour des densités de probabilité simple, tel que des densités gausiennes. Le problème ne possède plus de solution exacte utilisable en pratique dans des cas plus complexes, comme celui de l'utilisation de mélanges de gaussiennes. Il existe cependant des méthodes approchées pour résoudre ce problème. Soit $F(\tau)$ le quantile de densité associé à la valeur τ pour une densité de probabilité donnée :

$$F(\tau) = \int_{p(\boldsymbol{x}) \geq \tau} p(\boldsymbol{x}) d\boldsymbol{x} \tag{10}$$

Ce quantile de densité correspond à une région de densité maximale pour l'intervalle des valeurs de la densité de probabilité supérieur à τ. Inversement, la fonction réciproque tel que $h(F(\tau)) = \tau$ peut être définie de sorte que $F \in [0, 1]$ soit le quantile désiré (0.9 pour 90% de la densité de probabilité par exemple).

La méthode approchée pour résoudre ce problème se base sur une approche de Monte Carlo. Soit $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N)$ N points tirés aléatoirement selon une distribution p et $p_i = p(\boldsymbol{x}_i)\ \forall i \in [1, N]$. Les p_i sont ensuite triés par ordre croissant pour former l'ensemble $Y = (y_1, y_2, ..., y_N)$. Cette ensemble Y va servir à estimer $F(\tau)$ et $h(F)$ par l'interpolation linéaire suivante :

$$i = \underset{i}{\operatorname{argmax}} \{y_i | y_i \leq \tau\} \tag{11}$$

$$F(\tau) \approx \begin{cases} 1 - \frac{l(0,\tau)}{N} & si\ \tau < y_1 \\ 0 & si\ \tau \geq y_N \\ 1 - \frac{i + l(i,\tau)}{N} & sinon \end{cases} \tag{12}$$

avec

$$l(i, \tau) = \begin{cases} \frac{\tau}{y_1} & si\ i = 0 \\ 0.5 & si\ y_{i+1} - y_i = 0 \\ \frac{\tau - y_i}{y_{i+1} - y_i} & sinon \end{cases} \tag{13}$$

On peut en déduire la transformation inverse $h(F)$ par :

$$\tau = h(F) \approx \begin{cases} y_N & si\ i = N \\ (N(1-F))y_1 & si\ i = 0 \\ y_i + (N(1-F) - i)(y_{i+1} - y_i) & sinon \end{cases} \tag{14}$$

avec $i = \lfloor N \times (1 - F) \rfloor$.

Cette dernière formule permet d'obtenir le seuil τ qui n'est ni plus ni moins que le seuil C de l'equation 8. En pratique, l'ensemble des Y n'est pas obtenu à partir d'un tirage aléatoire de x_i mais en triant le vecteur de vraisemblance $\boldsymbol{\varphi}$. L'intérêt essentiel de la confiance est de donner un sens

plus intuitif au seuil que l'utilisateur doit fixer pour délimiter ce qui est considéré comme faisant partie du modèle. En effet, en choisissant un quantile de 95% (F = 0.95), ce calcul permet d'obtenir le seuil C associé quelque soit la distribution.

3 Résultats

Cette partie présente les évaluations réalisées afin de valider la méthode proposée. La première partie montre la faculté du modèle épars $\widetilde{\boldsymbol{Z}}$ à approximer la distribution non paramétrique $\boldsymbol{Z}$ (équation 2). Cette distribution non paramétrique $\boldsymbol{Z}$ sera notée distribution KDE (Kernel Density Estimation) par la suite et fera office de distribution de référence. La seconde partie compare les performances de l'algorithme avec celle du One Class SVM.

3.1 Approximation de la représentation non paramérique

Afin de donner une représentation graphique de la qualité d'approximation éparse du modèle présenté plus haut, celle-ci a été comparée au modèle KDE sur des primitives monodimensionelles, à savoir un ensemble d'angles représentant l'orientation de vecteurs de déplacement (cf. partie 4).

Nous avons fait le choix classique d'utiliser des fonctions à base radiale, $\phi_{k'}(z_k) = \frac{1}{\sigma_l\sqrt{2\pi}} \exp\{-\frac{d_\theta^2(z_k,z_{k'})}{2\sigma_l^2}\}$. De plus, $d_\theta(z_k, z_{k'})$ est la distance entre deux angles définie par :

$$d_\theta(\theta_1,\theta_2) = \frac{min(\ (\theta_2-\theta_1), |\theta_2-(\theta_1+2\pi)|\ )}{\pi}$$

$$avec\ \theta_1 < \theta_2$$

La distribution à deux modes représente des orientations réelles issues de l'application, alors que la distribution quasi uniforme a été obtenue à partir de donner de synthèse. La taille de la base d'apprentissage et donc, du modèle non paramétrique, est dans les deux cas de 2000 primitives. Pour faire cette comparaison, la fonction de vraisemblance calculée à partir du KDE est représentée en jaune délimitée par la courbe en pointillé noir sur la figure 1. La fonction de vraisemblance approximée est représentée pour différentes valeurs de t_v par les courbes bleu, rouge et verte sur la figure 1.

Les courbes du modèle épars sur la figure 1 montrent que la représentation réduite $\widetilde{\boldsymbol{Z}}$ correspond au modèle complet. La largeur de bande σ_l a été fixée à $\sigma_l = 0.1$. Le paramètre t_v détermine la précision de l'approximation de la vraisemblance. La distribution à deux modes de la figure 1 montre que pour des grandes valeurs de t_v (courbes bleue et rouge), les largeurs de bande des modes de l'approximation sont tronquées comparées à la densité non paramétrique. Des modes plus petits peuvent même être oubliés dans certains cas (courbe bleue). Les nombres de vecteurs conservés, c'est-à-dire le nombre de vecteurs pertinents nécessaires à l'approximation sont respectivement 1, 4 et 8 pour la courbe bleue, rouge et verte. Pour une distribution

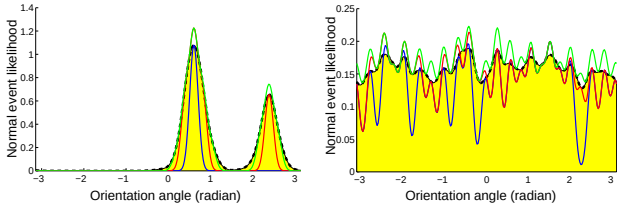


FIG. 1 – Approximation de la densité non paramétrique basée sur les fonctions noyaux par un modèle épars de fonctions noyaux sur deux types différents de distribution : La zone jaune délimitée par la courbe en pointillé noir montre la représentation non paramétrique. La courbe bleue ($t_v = 0.7$), la rouge ($t_v = 0.4$) et la verte ($t_v = 0.08$) montrent l'estimation du modèle épars pour différentes valeurs de t_v.

plus uniforme, de nombreux vecteurs sont nécessaires pour une approximation correcte. En conséquence, le paramètre t_v doit être suffisamment petit pour approximer correctement toutes les orientations possibles, ce qui n'est pas le cas de la courbe bleue qui sous-estime certaines orientations. Ici, 21, 26 et 39 gaussiennes ont été nécessaires respectivement pour les courbes bleues, rouges et vertes.

Une autre validation a été réalisée sur une base de données pour la classification de pixels[1] que nous nommerons par la suite B_{pixel}. Cette base comporte une base d'apprentissage constituée de 30 primitives par classe de pixel (7 classes au total), chaque primitive appartenant à un espace de dimension 19.

σ	1	10	25	40	100
EQM	7.8e-7	3.1e-5	3.2e-4	7e-4	1.8e-3
$\sharp\widetilde{\boldsymbol{Z}}$	30	26.3	15.4	6.7	2.1

TAB. 1 – Erreur quadratique moyenne et nombre moyen de vecteurs pertinents conservés pour l'estimation des vraisemblances des 7 classes d'apprentissage. Le seuil t_v a été choisi égal à 0.1.

Le tableau 1 montre l'erreur quadratique moyenne entre les estimations issues du KDE et celle issues de notre méthode sur les primitives de l'apprentissage. La deuxième ligne du tableau montre aussi le nombre moyen de vecteurs support conservés sur les 7 apprentissages. On constate que pour de très faibles valeurs de sigma l'ensemble des vecteurs de la base d'apprentissage sont conservés, la largeur de noyau est trop faible pour pouvoir regrouper plusieurs gaussiennes ensemble. Dans ce cas, notre modèle épars est quasiment identique au modèle KDE, ce qui se constate d'ailleurs sur l'erreur quadratique moyenne. La représentation KDE tout comme la nôtre pourrait être apparentées à un peigne de Dirac dans l'espace des primitives, chaque pseudo Dirac étant la gaussienne associé à une primitive. Pour des valeurs de sigma plus grandes, on s'aperçoit que

[1] http ://archive.ics.uci.edu/ml/datasets/Image+Segmentation

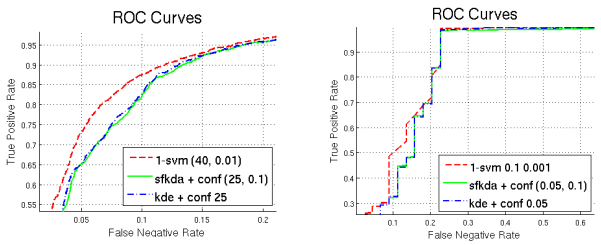


FIG. 2 – Courbes ROC sur les bases d'apprentissage B_{pixel} et B_{blocs}. Le nombre moyen de vecteurs conservés pour les One Class SVM est de 15.85 et 8.6 pour les base B_{pixel} et B_{blocs} respectivement. Pour notre algorithme le nombre de vecteurs conservés est de 15.42 et 4.2 respectivement. Le nombre de primitives pour chacune des bases d'apprentissage de B_{pixel} est de 30 vecteurs et de 2000 pour B_{blocs}.

le nombre de vecteurs sélectionnés diminue alors que l'erreur quadratique moyenne augmente. Pour de grandes valeurs de sigma, l'approximation reste toujours assez fine puisque l'erreur quadratique moyenne est de l'ordre de 10^{-3}. Cependant, une trop grande valeur de sigma fausse le modèle KDE vis-à-vis des données réelles. En effet, une trop grande largeur de bande peut forcer la modélisation KDE, et donc à fortiori la nôtre, à fusionner plusieurs modes de la distribution idéale, pourtant bien distincts, en un seul.

3.2 Comparaison des résultats de classification

Notre algorithme vise à concurrencer celui des One Class SVM [13]. En conséquence, une comparaison avec ce dernier a été réalisée sous MATLAB avec l'implémentation de Rakotomamonjy[2]. Les bases exploitées pour l'évaluation sont, la base B_{pixel} présentée dans la partie précédente avec une base de test de 300 primitives positives pour chaque classe et une base B_{blocs} comportant 2000 primitives d'orientation pour 5 classes différentes et 180 primitives de test pour chacune. On peut remarquer ici l'utilité de la notion de confiance qui permet d'adapter le seuil de la fonction discriminante selon l'apprentissage de chacune des classes à partir du même quantile. C'est d'ailleurs ce quantile qui sera le paramètre des courbes roc.
La figure 2 représente les courbes optimales pour chacune des méthodes. On constate, tout d'abord, que notre modèle approché obtient des performances équivalentes au modèle théorique avec environ deux fois moins de vecteurs support pour la base B_{pixel}. On peut constater que les performances du One Class SVM sont légèrement supérieures aux nôtres pour un nombre moyen de vecteurs support conservés équivalent pour la base B_{pixel} et inférieur pour la base B_{blocs}. En revanche, le temps de calcul pour l'apprentissage du One Class SVM est bien supérieur au temps d'apprentissage de notre méthode. En effet, le tableau 2 montre que notre algorithme met en moyenne 7 fois moins

[2]http ://asi.insa-rouen.fr/enseignants/∼arakotom/toolbox/index.html

	ν	nb cycle	t_v	nb cycle
B_{pixel} ($\sigma_l = 40$)	0.001	0.0374	0.1	0.00476
	0.01	0.0374	0.01	0.0068
	0.1	0.10098	0.001	0.00714
B_{Blocs} ($\sigma_l = 0.1$)	0.001	6.2	0.1	1.8
	0.01	27.2	0.01	6.46
	0.1	68.34	0.001	6.46

TAB. 2 – Nombre de cycle machine pour l'apprentissage en fonction du critère de convergence de chaque méthode (ν pour les One Class SVM et t_v pour notre méthode). Les temps sont exprimés en milliards de cycles.

de temps que le One Class SVM. Sur de plus grosses bases d'apprentissage avec 2000 primitives, le temps d'apprentissage de notre algorithme est quasiment constant contrairement à celui des One Class SVM. En effet, l'essentiel du temps de calcul est celui de la matrice de noyau qui est une étape également réalisée par le One Class SVM.
La qualité des courbes ROC pour la base B_{blocs}, sur la figure 2 s'explique par le fait que la base B_{blocs} est une base de primitives monodimensionnelles d'une part et que les orientations accumulées sont issue du déplacement de véhicule. En conséquence, de nombreuses orientations sont sensiblement identiques, leur classification s'effectuant ainsi par paquet, d'où l'aspect en escalier de la courbe.

$\sharp Z$	1-SVM		Nôtre méthode	
	nb cycle	$\sharp \tilde{Z}$	nb cycle	$\sharp \tilde{Z}$
100	0.0680	5.4	0.0129	4.2
200	0.1593	5.6	0.0535	4.2
400	0.6929	6.8	0.2392	4.2
800	2.6662	8	0.9824	4.2
1600	12.0744	8.4	4.0284	4.2
2000	18.8931	8.6	5.9825	4.2

TAB. 3 – Evolution du nombre de cycle machine et du nombre de vecteurs support moyen conservés en fonction de la taille de l'ensemble d'apprentissage. Les nombres de cycle sont exprimés en milliards.

Les résultats du tableau 3 montrent l'évolution du temps de calcul (exprimé par le nombre de cycle) de l'apprentissage en fonction de la taille de la base d'apprentissage. Les apprentissages ont été réalisé avec les paramètres permettant d'obtenir le plus haut taux de bonne détection à savoir 93% avec un taux de fausse alarme de 22% (cf. courbe de la figure 2 sur la base B_{blocs}). On observe que les deux algorithmes ont un temps de calcul qui quadruple lorsque l'on double la taille de la base d'apprentissage. De plus, comme constaté au tableau 2, nôtre algorithme est plus rapide que celui des One Class SVM, d'un facteur 3 avec le jeux de paramètres choisi.

4 Application

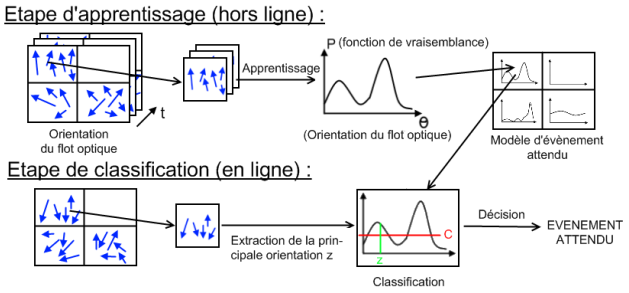


FIG. 3 – Schema du système.

Cette application, inspirée des travaux de [10], vise à l'assistance pour la vidéosurveillance de véhicules mais peut aussi être appliquée à la vidéosurveillance de foule. La figure 3 illustre le fonctionnement de l'application. Étant donné un flux vidéo issu d'une caméra fixe, le système est capable d'apprendre, à partir des orientations du flot optique cumulées au sein d'un bloc de l'image, une représentation non-paramétrique de la vraisemblance. La complexité de ce modèle dépendant directement de la taille de la base d'apprentissage, qui dans le cas présent explose, une approximation de la vraisemblance est réalisée grâce à notre méthode. Enfin, une fonction discriminante est déduite de cette vraisemblance afin de classifier l'orientation principale de chaque bloc pendant une phase d'analyse en temps réel. A noter ici, l'intérêt de la notion de confiance qui permet d'avoir un seuil différent pour chaque bloc en accord avec la distribution au sein de ce bloc à l'aide d'un seul paramètre, le quantile pris généralement à 99%.
Les performances du système ont été évaluées sur deux scènes différentes : un carrefour routier et un péage. Les séquences pour chacune des scènes sont issues de webcams réelles dans des conditions climatiques et d'illumination très différentes (e.g. fig.4). Chaque vidéo de la base de données est au format de 320x240 en 12 images par seconde. Les vidéos ont été prises toutes les 20 minutes et durent 30 secondes chacunes. L'apprentissage a été réalisé sur cent vidéos. Les blocs ont été fixés à une taille de 10x10 pour un total de 768 blocs par image.
Pour l'estimation du mouvement, différentes techniques de flot optique ont été évaluées et c'est celle de Black et Anandan [2] qui a été conservée pour sa robustesse et la qualité de ses résultats comparée aux autres méthodes, mais aussi pour son temps de calcul relativement rapide.
Les résultats de la classification du système ont été évalués en conditions réelles. L'analyse a été réalisée sur 162 vidéos pour le carrefour routier et 256 vidéos pour le péage. La classification a été réalisée en C++ sur un Pentium 4, 3.4GHz sur lequel, le calcul du flot optique prend 0.11s par image. Le temps de calcul pour l'analyse d'un bloc dépend du nombre de vecteurs de déplacement présents dans ce bloc. L'analyse sur les blocs sans mouvement n'étant pas effectuée, celle-ci prend environ 5ms, soit 0.2s en moyenne

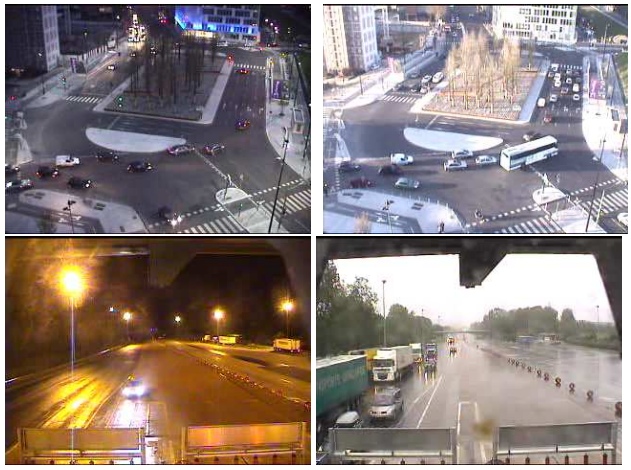
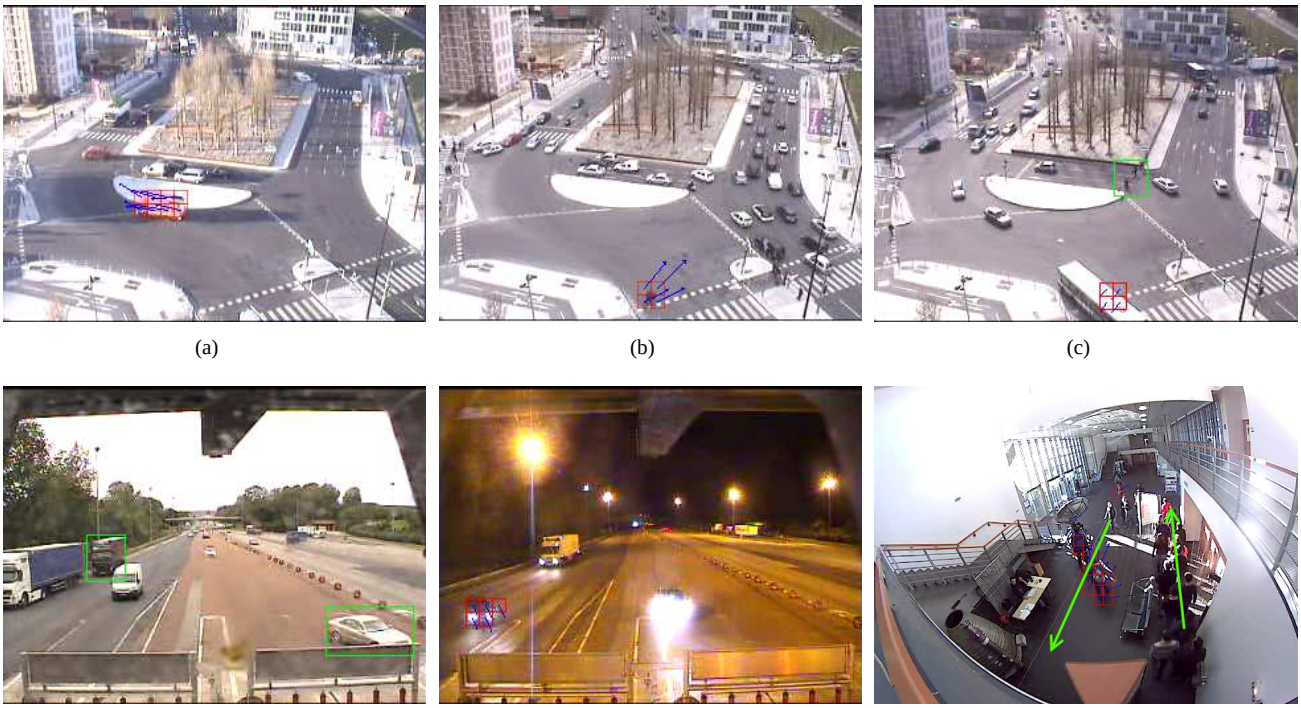


FIG. 4 – Exemples de conditions climatiques (soleil, pluie) et d'illumination (reflet, nuit) des séquences.

par image. Pour les séquences étudiées jusqu'à présent, ces performances sont proches du temps réel bien que le code ne soit pas optimisé. De plus, la classification réalisée s'adapte bien à la parallélisation sur carte graphique ou sur architecture multi-coeur, en vue d'augmenter encore la fréquence de traitement.
Un évènement inattendu même restreint au sens du mouvement, est une notion assez subjective. Différents operateurs humains ont effectué le travail d'analyse avec des résultats différents (sensibilité plus ou moins prononcée à certain écart de trajectoire, ...). Aussi la vérité terrain que nous considérons, est constituée de l'union de ces résultats. Une anomalie est considérée comme détectée si le système met en exergue au moins un bloc au niveau de l'objet coupable durant le temps de la manoeuvre (e.g. fig.5(a)). Le système permet de mettre en évidence différents types d'évènements allant du véhicule à contre-sens (e.g. fig.5(a)), au scooter suivant une trajectoire inhabituelle (e.g. fig.5(b)), en passant par les changments de file au niveau du péage d'autoroute (e.g. fig.5(e)). Cependant, le système alerte aussi l'utilisateur à tort, sur des bloc ou des images isolés. Pour éviter ce phénomène, des filtrages ont été appliqués à la sortie du classifieur. En supposant qu'une anomalie apparait sur plusieurs images et se propage d'un bloc à un bloc voisin, un filtre temporel recherchant l'orientation la plus représentantive parmis les orientations des k dernières images. Expérimentalement, utiliser $k = 2$ supprime des fausses détections due aux erreurs dans le calcul du champs de déplacement. De plus, un filtre temporel du premier ordre avec un filtrage spatial qui équilibre les réponses en fonction des voisinages a été appliqué. En conséquence de l'utilisation de ces filtres, certaines bonnes détections ont été suprimées après application du filtre (e.g figures 5(c) et 5(d) où les évènements à détecter ont été encadrés en vert). Mais de manière générale, ces filtres sont obligatoires pour que l'application reste utile et ne devienne pas une gêne pour l'utilisateur.
Les différentes fausses alarmes sont essentiellement dues



(a) (b) (c)

(d) (e) (f)

FIG. 5 – Exemples d'évènements inattendus. Les carrés rouges représentent les blocs inattendus détectés en accord avec la direction principale de ce bloc (flèche bleue) à l'instant t. Les carrés verts mettent en évidence les évènements inattendus ratés. (a) Camion de pompier allant à contre sens pour éviter des voies de circulation emcombrées. (b) Scooter quittant le trottoire en suivant une trajectoire inhabituelle. (c) Piétons traversant en dehors des passages piétons et exemple de fausse alarme sur le bus en bas de l'image. (d) Doubles détections ratées sur une voiture à droite et un camion à gauche qui déboitent soudainement tous les deux en même temps. (e) Voiture changeant de direction à la dernière minute sur la gauche de l'image. (f) Exemple de surveillance de foule. Le sens de circulation attendu est représenté par les flèches vertes.

à deux raisons : la première est un éventuel mouvement possible qui n'a pas été appris (ce mouvement n'a pas été suffisamment vu durant l'apprentissage). Une illustration de ce phénomène est montré sur la figure 6(a) où les orientations apprises ont été représentées en vert. On peut voir que l'apprentissage de ce bloc n'a pas retenu l'orientation vers la gauche (vue trop rarement pendant l'apprentissage), ce qui entraine des fausses alarmes lorsqu'un véhicule provient de la droite. Un autre cas de figure où ce problème intervient, est illustré sur la figure 6(c). Dans le cas de scène ensoleillée, le mouvement de l'ombre projettée cohérente à celui du véhicule, est classifié dans une zone où l'apprentissage ne correspond pas. La seconde raison intervient lorsque le bas niveau (le flot optique) fournit des primitives bruitées. Ici, le problème d'ouverture du flot optique est la principale cause des primitives bruitées. Ce problème, intrinsèque à la manière dont le flot optique est calculé, apparaît souvent sur les ombres ou les faisceaux lumineux des phares retournant des directions de déplacement complètement érronées (e.g. fig.6(b)). Pour atténuer ces effets non désirés, un calcul du flot optique est effectué uniquement sur des points possédants un score de Harris [7] important. Afin d'illustrer la généricité du système, l'algorithme peut aussi servir à la surveillance de foule dans des couloirs par exemple, comme illustré sur la figure 5(f), où deux personnes sont détectées à contre sens dans une zone où le déplacement normal s'effectue dans le sens trigonométrique.

5 Perspectives et Conclusion

Nous avons proposé une méthode d'apprentissage concurrente à celle des One Class SVM, l'objectif étant la classification binaire à partir uniquement de base d'exemples positifs. Pour cela, notre méthode s'appuie sur une représentation éparse de la distribution des primitives par une combinaison linéaire de fonctions noyaux. L'algorithme a été comparé à celui des One Class SVM avec lequel nous obtenons des résultats quasi équivalent pour un temps d'apprentissage inférieur. En condition réelle, la méthode a aussi été évaluée sur des séquences réelles et nous avons montré qu'elle est capable de classifier des évènements inattendus tel que des mouvements à contre sens, des vehicules déboitant subitement, des marches arrière, etc.

Les travaux futurs visent à affiner l'approximation par rapport au modèle KDE, et augmenter par la même occasion la qualité de la classification mais aussi à étudier l'influence du choix de la fonction noyau pour les modélisations de distribution. Notre algorithme présente l'avantage

(a) (b) (c)

FIG. 6 – Exemples de fausses alarmes. (a) Oubli lors de l'apprentissage. Les trois vecteurs du modèle épars ont été représentés en vert. Les trois vont vers la bas. (b) Problème d'ouverture. Le flot optique est orthogonal à la bordure du faisceau de lumière, donnant des résultats complètement faux. (c) Problème de projection. L'ombre s'étend sur des zones où aucun mouvement vers le haut gauche de l'image n'a été appris.

d'être plus simple que celui du One Class SVM, ce qui s'en ressent sur la compléxité. Cette simplicité nous permet d'envisager des mécanismes de mise-à-jour ou d'apprentissage séquentielle qui pourraient être utilisés lors de phase online. Dans le cas de l'utilisation de noyaux gaussiens dans notre méthode, une comparaison de notre approximation avec les approches EM [11] pour calculer les paramètres des mixtures de gaussiennes est aussi envisagée. Tout comme une comparaison avec des méthodes récentes, tel que la méthode SKDA [6].

Références

[1] E. L. Andrade, S. Blunsden, and R. B. Fisher. Hidden markov models for optical flow analysis in crowds. In *International Conference on Pattern Recognition (ICPR)*, volume 1, pages 460–463, 2006.

[2] Michael J. Black and P. Anandan. The robust estimation of multiple motions : parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding (CVIU)*, 63(1) :75–104, 1996.

[3] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2 :121–167, 1998.

[4] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons Inc., 2001.

[5] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning (ICML)*, pages 148–156, 1996.

[6] Bohyung Han, Dorin Comaniciu, Ying Zhu, and Larry S. Davis. Sequential kernel density approximation and its application to real-time visual tracking. *IEEE Pattern Analysis and Machine Intelligence*, 30 :1186–1197, 07 2008.

[7] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[8] Weiming Hu, Xuejuan Xiao, Zhouyu Fu, Dan Xie, Tieniu Tan, and Steve Maybank. A system for learning statistical motion patterns. *IEEE Pattern Analysis And Machine Intelligence*, 28 :1450–1464, 09 2006.

[9] T. Lindh, J. Ahola, J.K. Kamarainen, V. Kyrki, and J. Partanen. Bearing damage detection based on statistical discrimination of stator current. *International Symposium on Diagnostics for Electric Machines, Power Electronics and Drives (SDEMPED)*, pages 177– 181, 2003.

[10] G. Monteiro, M. Ribeiro, J. Marcos, and J.P. Batista. A framework for wrong way driver detection using optical flow. In *International Conference on Image Analysis and Recognition (ICIAR)*, pages 1117–1127, 2007.

[11] Pekka Paalanen, Joni-Kristian Kamarainen, Jarmo Ilonen, and Heikki Kälviäinen. Feature representation and discrimination based on gaussian mixture model probability densities-practices and algorithms. *Pattern Recognition*, 39 :1346–1358, 2006.

[12] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1) :15–33, 2000.

[13] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*, 2000.

[14] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Pattern Analysis And Machine Intelligence*, 22 :747–757, 08 2000.

[15] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Reasearch 1*, 2001.