SVM, traitement des données en ligne et invariances, quelles solutions?

Gaëlle Loosli,
Stéphane Canu
gaelle.loosli@insa-rouen.fr

Laboratoire d'Informatique, Traitement de l'Information, Systèmes.

La Reconnaissance des formes. Quelles méthodes pour quelles applications?
Journées d'Etudes Co-organisées par le Club 29 de la SEE et le GDR-ISIS

23 et 24 mars 2006, Paris

Introduction

Les algorithmes à noyaux et à vecteurs supports (SVM et autres algorithmes de la même famille)

- sont souvent très efficaces en terme de résultats numériques,
- mais fonctionnent hors ligne,
- et sont limités par des problèmes de temps ou de mémoire.

Contenu de cet exposé

- principes de résolution efficace des SVM,
- application de ces principes sur des problèmes peu traités par méthodes à noyaux.

Plan

Algorithmes itératifs pour résoudre les SVM

Principe des SVM :

- déterminer l'hyperplan qui sépare au mieux les données,
- déterminer l'influence α de chaque point d'apprentissage sur la construction de l'hyperplan,
- séparer les points dont l'influence est nulle ($\alpha = 0$) des points utiles ($\alpha > 0$),
- limiter l'influence de chaque point ($\alpha \leq C$).

Notations

- I_a est l'ensemble des points avec influence ($\alpha > 0$),
- I_0 est l'ensemble des points sans influence (contraints à 0, $\alpha = 0$),
- I_C est l'ensemble des points bornés (contraints à C, $\alpha = C$),
- I_w est l'ensemble des points non contraints ($0 < \alpha < C$).

Algorithmes itératifs pour résoudre les SVM

Principe des SVM :

- déterminer l'hyperplan qui sépare au mieux les données,
- déterminer l'influence α de chaque point d'apprentissage sur la construction de l'hyperplan,
- séparer les points dont l'influence est nulle ($\alpha = 0$) des points utiles ($\alpha > 0$),
- limiter l'influence de chaque point ($\alpha \leq C$).

Notations

- I_a est l'ensemble des points avec influence ($\alpha > 0$),
- I_0 est l'ensemble des points sans influence (contraints à 0, $\alpha = 0$),
- I_C est l'ensemble des points bornés (contraints à C, $\alpha = C$),
- I_w est l'ensemble des points non contraints ($0 < \alpha < C$).

Algorithmes itératifs pour résoudre les SVM

Principe des SVM :

- déterminer l'hyperplan qui sépare au mieux les données,
- déterminer l'influence α de chaque point d'apprentissage sur la construction de l'hyperplan,
- séparer les points dont l'influence est nulle ($\alpha = 0$) des points utiles ($\alpha > 0$),
- limiter l'influence de chaque point ($\alpha \leq C$).

Notations

- I_a est l'ensemble des points avec influence ($\alpha > 0$),
- I_0 est l'ensemble des points sans influence (contraints à 0, $\alpha = 0$),
- I_C est l'ensemble des points bornés (contraints à C, $\alpha = C$),
- I_w est l'ensemble des points non contraints ($0 < \alpha < C$).

Algorithmes itératifs pour résoudre les SVM

Principe des SVM :

- déterminer l'hyperplan qui sépare au mieux les données,
- déterminer l'influence α de chaque point d'apprentissage sur la construction de l'hyperplan,
- séparer les points dont l'influence est nulle ($\alpha = 0$) des points utiles ($\alpha > 0$),
- limiter l'influence de chaque point ($\alpha \leq C$).

Notations

- I_a est l'ensemble des points avec influence ($\alpha > 0$),
- I_0 est l'ensemble des points sans influence (contraints à 0, $\alpha = 0$),
- I_C est l'ensemble des points bornés (contraints à C, $\alpha = C$),
- I_w est l'ensemble des points non contraints ($0 < \alpha < C$).

Algorithmes itératifs pour résoudre les SVM

Principe des SVM :

- déterminer l'hyperplan qui sépare au mieux les données,
- déterminer l'influence α de chaque point d'apprentissage sur la construction de l'hyperplan,
- séparer les points dont l'influence est nulle ($\alpha = 0$) des points utiles ($\alpha > 0$),
- limiter l'influence de chaque point ($\alpha \leq C$).

Notations

- I_a est l'ensemble des points avec influence ($\alpha > 0$),
- I_0 est l'ensemble des points sans influence (contraints à 0, $\alpha = 0$),
- I_C est l'ensemble des points bornés (contraints à C, $\alpha = C$),
- I_w est l'ensemble des points non contraints ($0 < \alpha < C$).

Algorithmes itératifs pour résoudre les SVM - 2

Quelques impératifs pratiques :

- obtenir la solution la plus parcimonieuse possible,
- obtenir la solution en utilisant le moins de mémoire possible,

Particularités des bons algorithmes

- partir d'une solution *vide* ($\alpha_i = 0, \forall i \in 1, n$),
- ne faire de calculs coûteux que pour les points de I_w.

Algorithmes itératifs pour résoudre les SVM - 2

Quelques impératifs pratiques :

- obtenir la solution la plus parcimonieuse possible,
- obtenir la solution en utilisant le moins de mémoire possible,

Particularités des bons algorithmes

- partir d'une solution *vide* ($\alpha_i = 0, \forall i \in 1, n$),
- ne faire de calculs coûteux que pour les points de I_w.

Algorithmes itératifs pour résoudre les SVM - 3

Structure des algorithmes itératifs :

- choisir une répartition initiale des points dans I_w, I_C et I_0,
- calculer les valeurs initiales des α_w,
- tant que tous les points ne sont pas bien répartis
  - changer la répartition,
  - mettre à jours les α_w,
- vérifier l'optimalité de la solution

Méthodes de résolution répondant à ce schéma

- SMO (libSVM, SVM^{light}, ...) [6],
- SimpleSVM [7, 4],
- LASVM, CVM (méthodes approximatives)[1].

Algorithmes itératifs pour résoudre les SVM - 3

Structure des algorithmes itératifs :

- choisir une répartition initiale des points dans I_w, I_C et I_0,
- calculer les valeurs initiales des α_w,
- tant que tous les points ne sont pas bien répartis
  - changer la répartition,
  - mettre à jours les α_w,
- vérifier l'optimalité de la solution

Méthodes de résolution répondant à ce schéma

- SMO (libSVM, SVM^{light}, ...) [6],
- SimpleSVM [7, 4],
- LASVM, CVM (méthodes approximatives)[1].

Algorithmes itératifs pour résoudre les SVM - 3

Structure des algorithmes itératifs :

- choisir une répartition initiale des points dans I_w, I_C et I_0,
- calculer les valeurs initiales des α_w,
- tant que tous les points ne sont pas bien répartis
  - changer la répartition,
  - mettre à jours les α_w,
- vérifier l'optimalité de la solution

Méthodes de résolution répondant à ce schéma

- SMO (libSVM, SVM^{light}, ...) [6],
- SimpleSVM [7, 4],
- LASVM, CVM (méthodes approximatives)[1].

Algorithmes itératifs pour résoudre les SVM - 3

Structure des algorithmes itératifs :

- choisir une répartition initiale des points dans I_w, I_C et I_0,
- calculer les valeurs initiales des α_w,
- tant que tous les points ne sont pas bien répartis
  - changer la répartition,
  - mettre à jours les α_w,
- vérifier l'optimalité de la solution

Méthodes de résolution répondant à ce schéma

- SMO (libSVM, SVM^{light}, ...) [6],
- SimpleSVM [7, 4],
- LASVM, CVM (méthodes approximatives)[1].

Algorithmes itératifs pour résoudre les SVM - 4

	SMO	SimpleSVM	LASVM
Sélection pour ajout	Un point de I_0 et un point de I_a	Un point de I_0 ou de I_C le plus mal classé par la solution courante	Le prochain point de la base et un point de I_a
Mise à jour des α	Recherche de point admissible sur la ligne définie par les deux points	Mise à jour d'un système linéaire sur I_w	Recherche de point admissible sur la ligne définie par les deux points
Sélection pour retrait	Un des deux points précédents dont le α viole les contraintes 0 ou C	Un point de I_w dont le α viole les contraintes 0 ou C	Deux points de I_a définissant la meilleure direction d'optimisation
Arrêt	Aucune direction donnée par un couple de points n'améliore la solution	Tous les points de I_0 et I_C sont bien classés par la solution courante	Il n'y a plus de points non vus

Algorithmes itératifs pour résoudre les SVM - 5 (fin)

Points forts

- SMO : résolution analytique (par 2 points), gestion efficace de la mémoire
- SimpleSVM : facilité de la reprise à chaud, converge en moins d'étapes que SMO
- LASVM : utilisation en ligne (donc reprise à chaud immédiate), résolution analytique

Points faibles

- SMO : converge, mais en un nombre d'étapes indeterminé, reprise à chaud moins aisée que SimpleSVM
- SimpleSVM : limitation mémoire par la taille de I_w
- LASVM : solution sous optimale, plusieurs passes nécéssaires pour les petites bases de données

Algorithmes itératifs pour résoudre les SVM - 5 (fin)

Points forts

- SMO : résolution analytique (par 2 points), gestion efficace de la mémoire
- SimpleSVM : facilité de la reprise à chaud, converge en moins d'étapes que SMO
- LASVM : utilisation en ligne (donc reprise à chaud immédiate), résolution analytique

Points faibles

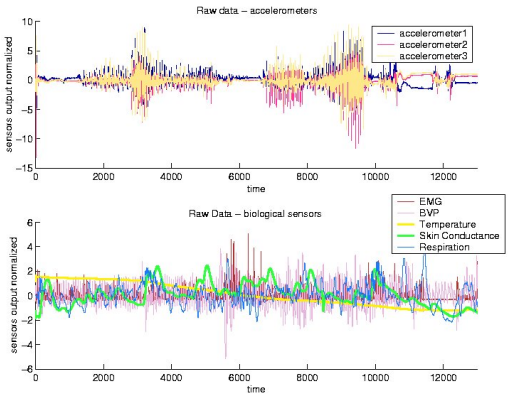
- SMO : converge, mais en un nombre d'étapes indeterminé, reprise à chaud moins aisée que SimpleSVM
- SimpleSVM : limitation mémoire par la taille de I_w
- LASVM : solution sous optimale, plusieurs passes nécéssaires pour les petites bases de données

Plan

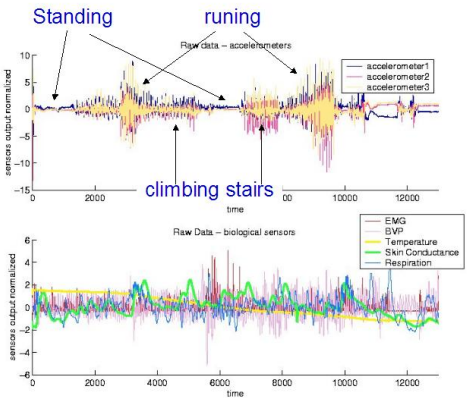
Détection de changement [5]

- l'utilisateur est équippé de capteurs,
- on souhaite déterminer s'il y a un changement dans son activité

⇒



⇒



SVM à une Classe

Principes de l'apprentissage à une classe

- apprend avec des exemples issus d'une seule classe,
- la frontière de décision *entoure* les données,
- détection d'outliers (points loin des autres)

Idée pour la détection de rupture

- estimer le contour de la situation courante,
- classer les points futurs,
- s'ils sont à l'intérieur du contour, pas de changement,
- s'ils sont à l'extérieur du contour, changement.

SVM à une Classe

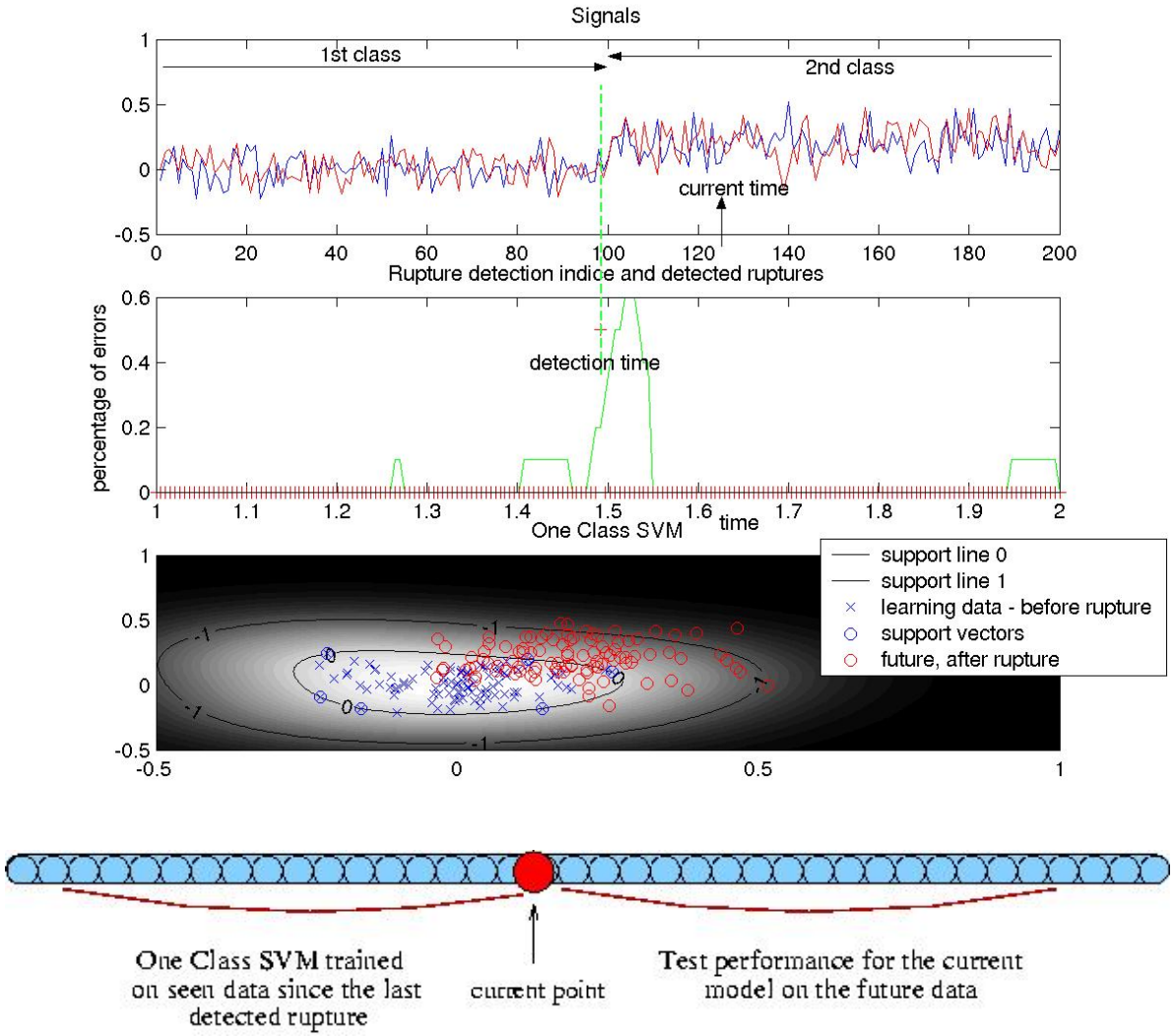
Principes de l'apprentissage à une classe

- apprend avec des exemples issus d'une seule classe,
- la frontière de décision *entoure* les données,
- détection d'outliers (points loin des autres)

Idée pour la détection de rupture

- estimer le contour de la situation courante,
- classer les points futurs,
- s'ils sont à l'intérieur du contour, pas de changement,
- s'ils sont à l'extérieur du contour, changement.

Méthodes


Signals
1st class
2nd class
current time
Rupture detection indice and detected ruptures
percentage of errors
detection time
time
One Class SVM
support line 0
support line 1
learning data - before rupture
support vectors
future, after rupture
One Class SVM trained on seen data since the last detected rupture
current point
Test performance for the current model on the future data

Méthodes - 2

Utilisation de SimpleSVM pour cette tâche :

- si le point le plus *vieux* est vecteur support : l'enlever et faire un mise à jour de la solution
- si le nouveau point ne tombe pas à l'intérieur de la classe courante : l'ajouter aux vecteurs support et faire une mise à jour de la solution

Pourquoi SimpleSVM?

- mise à jour aisée
- travaille sur peu de points

Méthodes - 2

Utilisation de SimpleSVM pour cette tâche :

- si le point le plus *vieux* est vecteur support : l'enlever et faire un mise à jour de la solution
- si le nouveau point ne tombe pas à l'intérieur de la classe courante : l'ajouter aux vecteurs support et faire une mise à jour de la solution

Pourquoi SimpleSVM?

- mise à jour aisée
- travaille sur peu de points

Méthodes - 2

Utilisation de SimpleSVM pour cette tâche :

- si le point le plus *vieux* est vecteur support : l'enlever et faire un mise à jour de la solution
- si le nouveau point ne tombe pas à l'intérieur de la classe courante : l'ajouter aux vecteurs support et faire une mise à jour de la solution

Pourquoi SimpleSVM?

- mise à jour aisée
- travaille sur peu de points

Résultats

	Jeu		Neutre	
Ruptures (a posteriori)	31		24	
Manuel	26/31	83.9%	17/24	70.1%
Auto	21/31	67.7%	22/24	91.7%
Commun	16/31	51.6 %	15/24	62.5 %
Manuel seul	10/31	32.3%	2/24	8.3%
Auto seul	5/31	16.1%	7/24	29.1%
fausse alarme	6/33	18.8%	12/36	33.3%
Non detection	10/31	32.2%	2/24	8.3%

- Jeu : les données sont issues de capteurs biologiques seulement, pas de déplacement de l'utilisateur,
- Neutre : les données sont issues de capteurs biologiques et accéléromètres, déplacement de l'utilisateur.

Clef du succés - Problème en ligne avec peu de points

La mise à jour rapide de la solution.

Résultats

	Jeu		Neutre	
Ruptures (a posteriori)	31		24	
Manuel	26/31	83.9%	17/24	70.1%
Auto	21/31	67.7%	22/24	91.7%
Commun	16/31	51.6 %	15/24	62.5 %
Manuel seul	10/31	32.3%	2/24	8.3%
Auto seul	5/31	16.1%	7/24	29.1%
fausse alarme	6/33	18.8%	12/36	33.3%
Non detection	10/31	32.2%	2/24	8.3%

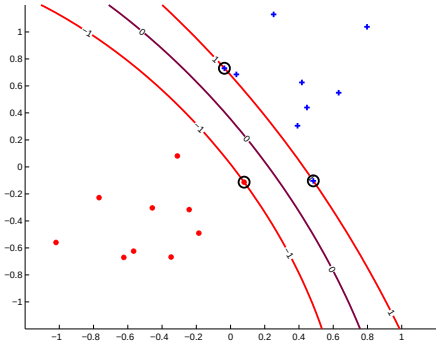
- Jeu : les données sont issues de capteurs biologiques seulement, pas de déplacement de l'utilisateur,
- Neutre : les données sont issues de capteurs biologiques et accéléromètres, déplacement de l'utilisateur.

Clef du succés - Problème en ligne avec peu de points

La mise à jour rapide de la solution.

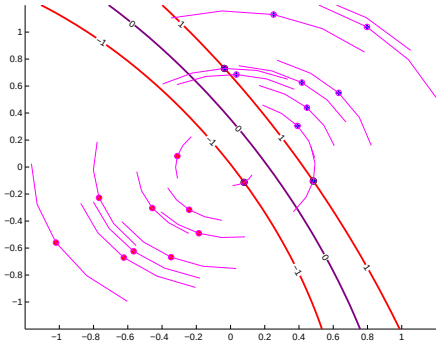
Plan

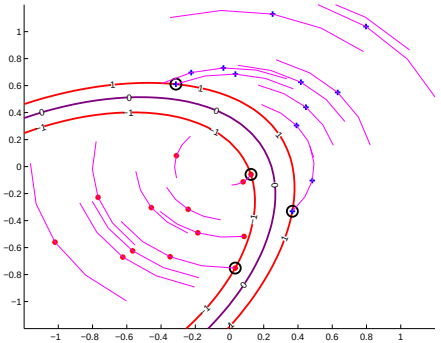
Invariances

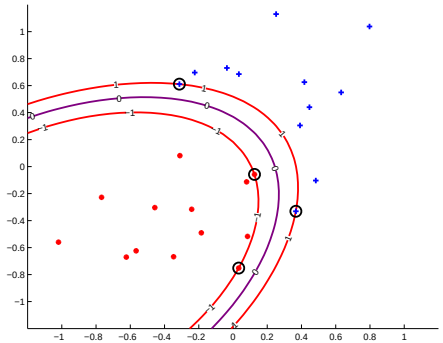


Problèmes posés par les invariances traités par des méthodes à noyaux[3] :

- solution 1 : générer les variations des points et utiliser la base de données élargie $\Rightarrow$ problèmes de taille,
- solution 2 : utiliser un noyau invariant $\Rightarrow$ limitation des types d'invariances,
- solution 3 : générer à la demande des variations pendant l'apprentissage $\Rightarrow$ demande un algorithme en ligne.







LASVM Invariant

Utilisation de LASVM pour les invariances [2] :

- apprend sur la base d'origine (n'utilise qu'une fois chaque exemple),
- à chaque itération suivante, génère une déformation d'un exemple original et s'en sert comme nouveau point,
- continue tant que le temps imparti ou la mémoire le permet...

Pourquoi LASVM?

- fonctionne en ligne,
- propose des heuristiques de type *active learning* pour accélerer l'apprentissage.

LASVM Invariant

Utilisation de LASVM pour les invariances [2] :

- apprend sur la base d'origine (n'utilise qu'une fois chaque exemple),
- à chaque itération suivante, génère une déformation d'un exemple original et s'en sert comme nouveau point,
- continue tant que le temps imparti ou la mémoire le permet...

Pourquoi LASVM?

- fonctionne en ligne,
- propose des heuristiques de type *active learning* pour accélerer l'apprentissage.

LASVM Invariant

Utilisation de LASVM pour les invariances [2] :

- apprend sur la base d'origine (n'utilise qu'une fois chaque exemple),
- à chaque itération suivante, génère une déformation d'un exemple original et s'en sert comme nouveau point,
- continue tant que le temps imparti ou la mémoire le permet...

Pourquoi LASVM?

- fonctionne en ligne,
- propose des heuristiques de type *active learning* pour accélerer l'apprentissage.

LASVM Invariant

Utilisation de LASVM pour les invariances [2] :

- apprend sur la base d'origine (n'utilise qu'une fois chaque exemple),
- à chaque itération suivante, génère une déformation d'un exemple original et s'en sert comme nouveau point,
- continue tant que le temps imparti ou la mémoire le permet...

Pourquoi LASVM?

- fonctionne en ligne,
- propose des heuristiques de type *active learning* pour accélerer l'apprentissage.

Résultats

Sur la base de données MNIST :

- invariances : affines (translation, rotation,...) , élastiques (champs de déformation aléatoire)
- nombre de variations : arrêt à 8 millions d'exemples, soit 130 déformations par points
- hyper paramètres : réglés par validation croisée sur 500 000 points

Résultats sur MNIST étendue

- 8 jours d'apprentissage pour les 10 classifieurs 1 contre tous ($10 \times 8 millions$) et le test
- de 8 000 à 25 000 vecteurs supports par classifieurs
- 0.67% d'erreur en test

Clef du succés - Problème en ligne avec beaucoup de points

La parcimonie des solutions

Résultats

Sur la base de données MNIST :

- invariances : affines (translation, rotation,...) , élastiques (champs de déformation aléatoire)
- nombre de variations : arrêt à 8 millions d'exemples, soit 130 déformations par points
- hyper paramètres : réglés par validation croisée sur 500 000 points

Résultats sur MNIST étendue

- 8 jours d'apprentissage pour les 10 classifieurs 1 contre tous ($10 \times 8 millions$) et le test
- de 8 000 à 25 000 vecteurs supports par classifieurs
- 0.67% d'erreur en test

Clef du succés - Problème en ligne avec beaucoup de points

La parcimonie des solutions

Plan

Conclusion

L'apprentissage et l'optimisation

- la parcimonie est essentielle,
- l'optimalité complète l'est moins,
- la souplesse permet de traiter des problèmes réalistes (mise à jour facile ou méthode en ligne),
- le choix de l'algorithme de résolution est important.

Bibliographie

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou.
Fast kernel classifiers with online and active learning.
Journal of Machine Learning Research, 6:1579–1619, September 2005.

Gaëlle Loosli, Léon Bottou, and Stéphane Canu.
Training invariant support vector machines using selective sampling.
Technical report, LITIS - NEC Laboratories of America, 2006.

Gaëlle Loosli, Stéphane Canu, SVN Vishwanathan, and Alexander J.Smola.
Invariances in classification : an efficient svm implementation.
Applied Stochastic Models and Data Analysis, 2005.

Gaëlle Loosli, Stéphane Canu, S.V.N. Vishwanathan, Alexander J. Smola, and Monojit Chattopadhyay.
Boîte à outils svm simple et rapide.
Revue d'intelligence artificielle, 19, 2005.

Gaëlle Loosli, Sang-Goog Lee, and Stéphane Canu.
Context changes detection by one-class svms.
UM 2005: Workshop on Machine Learning for User Modeling: Challenges, 2005.

John Platt.
Fast training of support vector machines using sequential minimal optimization.
In B. Scholkopf, C. Burges, and A. Smola, editors, *Advanced in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1999.

S. V. N Vishwanathan, Alexander J. Smola, and M. Narasimha Murty.
SimpleSVM.
Proceedings of the Twentieth International Conference on Machine Learning, 2003.