

Insertion de proximal SVM dans des arbres aléatoires, mode d'emploi

Cédric Simon¹, Jérôme Meessen², Christophe De Vleeschouwer¹

¹ Laboratoire de Télécommunication et Télédétection, UCL
2, Place du Levant, 1348 Louvain-la-Neuve, Belgique
cedric.simon@uclouvain.be
christophe.devleeschouwer@uclouvain.be

² Multitel
Mons, Belgique
jerome.meessen@multitel.be

Résumé :

En insérant plusieurs classifieurs SVM dans une architecture d'arbre binaire, il est possible de changer un problème multi-classes arbitraire en une hiérarchie de classifications binaires. Un des problèmes essentiels consiste à déterminer à chaque nœud de l'arbre la façon de regrouper les classes multiples en une paire de classes *superposées* à discriminer. Comme contribution principale, cet article propose d'utiliser un ensemble d'arbres aléatoires au lieu d'un seul arbre de décision optimisé, et ce de façon à réduire l'impact du choix de la paire de classes superposées sur les performances du classifieur. Les résultats empiriques obtenus sur différents ensembles de données UCI démontrent une amélioration des performances de classification, en comparaison aux solutions SVM et aux ensembles d'arbres aléatoires conventionnels.

Mots-clés : SVM, proximal SVM, ensemble d'arbres de décision, classification multi-classes.

1 Introduction

Les algorithmes de Séparateurs à Vaste Marge (SVM) initialement proposés par Vapnik (Vapnik, 1995) sont devenus des méthodes incontournables pour l'exploration de données, appliqués avec succès à la classification, la sélection de caractéristiques, le partitionnement de données ou encore l'analyse de séries temporelles. Dans le contexte de la classification, les SVM s'adressent implicitement aux problèmes à classes binaires. Plusieurs méthodes ont toutefois été proposées afin d'étendre son utilisation aux données multi-catégories (Crammer & Singer, 2001; Platt *et al.*, 2000; Hsu & Lin, 2002), mais sans amélioration significative par rapport à la stratégie "*un-contre-un*", si ce n'est un avantage de rapidité pour la méthode DAGSVM. Dans (Hsu & Lin, 2002), Hsu et Lin distinguent deux approches permettant de réaliser de la classification SVM

multi-classes. La première est de construire et puis de combiner plusieurs classificateurs binaires, alors que la deuxième considère directement toutes les données dans une seule formulation. La méthode que nous proposons ici se situe à l'intersection de ces deux approches.

Récemment, il a été suggéré que les SVM peuvent bénéficier de l'architecture des arbres de décision pour réaliser des classifications à catégories multiples de façon plus naturelle (Cheong *et al.*, 2004; Cheng *et al.*, 2008). L'idée principale consiste à partitionner à chaque nœud d'un arbre de décision l'ensemble de classes fournies en entrée en une paire de classes *superposées*, qui sont ensuite analysées avec un classifieur SVM binaire. Dans (Cheong *et al.*, 2004), Cheong propose une architecture d'arbre binaire à base de SVM afin d'obtenir un bon compromis entre la rapidité computationnelle des arbres de décision, utilisés pour une séparation hiérarchique des classes, et les bonnes performances de classification des SVM. Néanmoins, ses principales conclusions mettent en avant le fait que convertir un problème multi-classes en un arbre binaire optimal soulève une question délicate. En effet, les définitions de paires de classes superposées qui sont mises en œuvre dans les premiers nœuds de l'arbre affectent les options d'agrégation ultérieures des classes. Cela engendre une certaine difficulté à créer une structure d'arbre de décision optimale, du fait que cela nécessite une investigation exhaustive de toutes les options d'agrégation possible à chaque nœud de l'arbre. Pour réduire ce problème, Cheng *et al.* ont récemment proposé une approche ascendante pour déterminer les partitions hiérarchiques de données multi-classes en une séquence de SVM binaires (Cheng *et al.*, 2008). Leur approche fait l'hypothèse que les premières classes à isoler dans une structure binaire sont celles qui ont la distance relative moyenne par rapport aux autres classes la plus grande. Ainsi, ils suivent une approche ascendante dans laquelle ils inspectent en premier lieu comment séparer les classes les plus similaires, tel que mesurée par une métrique heuristique. Bien que possiblement efficace, il est utile de mentionner qu'un tel procédé heuristique pour guider la conception d'un arbre optimal ne diminue en rien les inconvénients d'un arbre de décision unique. En particulier, ces arbres sont connus pour offrir des propriétés de généralisations relativement pauvres.

De façon à éviter ces désavantages, nous proposons de construire un ensemble d'arbres aléatoires, au lieu d'un seul arbre de décision optimisé. L'efficacité d'un ensemble d'arbre de décision a été démontrée récemment dans plusieurs articles (Breiman, 2001; Geurts *et al.*, 2006), ainsi que leur consistance statistique (Biau *et al.*, 2008). Dans notre cas, cette approche a l'avantage (1) d'assouplir et de contourner virtuellement la question associée à la définition d'une paire de classes superposées optimale à chaque nœud des arbres de décision, et (2) d'augmenter la robustesse et la capacité de généralisation de l'ensemble de classifieurs résultant.

Officiellement, chaque arbre individuel examine à chacun de ses nœuds une partition pseudo-aléatoire des classes d'entrée en deux 'super-classes', appelées classes superposées. En rendant aléatoire le processus de construction de l'arbre et en combinant le résultat de chacun des arbres construits, aucune décision stricte n'est prise regardant les séquences de partitions de classes rencontrées le long de la structure binaire. La définition des paires de classes superposées dans chaque arbre fait donc partie du processus aléatoire, et aide ainsi à réduire la variance des résultats. La pénalité de temps de calcul

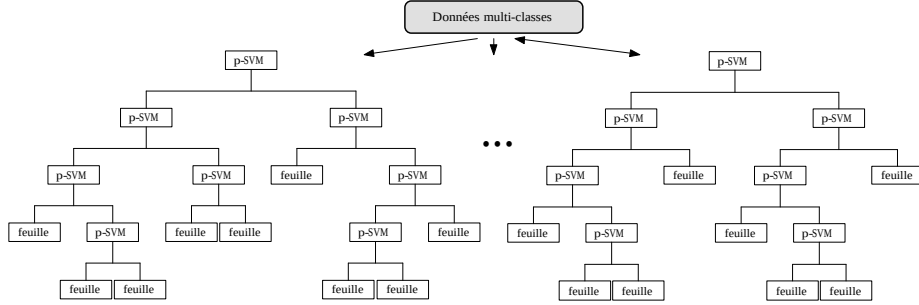


FIG. 1 – Structure générale du classifieur à base d'arbres à PSVM

provoquée par l'utilisation des SVM dans plusieurs arbres peut être réduite en utilisant des classifieurs plus simples, tel que les LS-SVM (Suykens & Vandewalle, 1999) ("least square support vector machine") ou encore les PSVM (Fung & Mangasarian, 2005) ("proximal support vector machine"). Tous deux peuvent être interprétés comme une solution aux moindres carrés régularisée d'un système d'équation linéaire (Tikhonov & Arsenin, 1977), et sont des méthodes extrêmement rapides et efficaces. Ce travail propose donc l'utilisation des PSVM, ce qui permet d'obtenir un bon compromis entre performance de classification et rapidité de calcul.

Du point de vue de la méthode d'arbre de décision, l'utilisation de PSVM à chaque nœud permet de trouver une frontière quasi-optimale en utilisant des attributs multiples à chaque nœud. Or, des travaux récents (Lee & Olafsson, 2006) ont montré que des arbres de décision utilisant plusieurs attributs à chaque nœud pouvaient améliorer la performance des arbres de décision classiques. Les résultats fournis dans la section expérimentale viennent renforcer cette affirmation.

Le reste de cet article est organisé de la façon suivante. La section 2 présente notre méthode. Dans la section 3, les résultats expérimentaux sont reportés pour cinq ensembles de données multi-classes provenant de la UCI Machine Learning repository ¹. Finalement, la section 4 tire les conclusions et ouvre des perspectives à ce travail.

2 Arbres à PSVM

La figure 1 montre la structure générale du classifieur proposé. Nous expliquons ci-dessous la théorie et les spécificités du PSVM ainsi que celles des arbres aléatoires, et la façon dont ils peuvent être combinés ensemble.

2.1 PSVM linéaire

Les éléments clés de la théorie des PSVM, tel que détaillé dans (Fung & Mangasarian, 2001, 2005) sont les suivants. Nous considérons le problème de classification de m points dans un espace réel $\mathbb{R}^n$ à n dimensions. Les m points sont représentés par la

¹<http://archive.ics.uci.edu/ml/>

matrice A de dimension $m \times n$. Chaque point A_i appartient à la classe positive ou négative, tel que spécifié par une matrice D diagonale $m \times m$ avec $+1$ ou -1 le long de sa diagonale. Pour ce problème, un SVM traditionnel avec un noyau linéaire est donné par le programme quadratique suivant, avec le paramètre $\nu > 0$ (Vapnik, 1995) :

$$\begin{aligned} \min_{(w, \gamma, y) \in \mathbb{R}^{n+1+m}} \quad & \nu e' y + \frac{1}{2} w' w \\ \text{s.t.} \quad & D(Aw - e\gamma) + y \geq e \\ & y \geq 0 \end{aligned} \quad (1)$$

Géométriquement, cette formulation calcule deux hyperplans $x'w - \gamma = \pm 1$ qui séparent la plupart des échantillons positifs et négatifs. En minimisant la norme 2 du vecteur d'erreur y (au lieu de la norme 1), et en remplaçant la contrainte d'inégalité par une égalité, Fung et Mangasarian (Fung & Mangasarian, 2001) changent les SVM traditionnels en formulant le problème d'optimisation *libre* suivant :

$$\min_{(w, \gamma) \in \mathbb{R}^{n+1+m}} \frac{\nu}{2} \| D(Aw - e\gamma) - e \|^2 + \frac{1}{2} (w'w + \gamma^2). \quad (2)$$

Cette nouvelle formulation peut être vue comme une solution aux moindres carrés régularisée du système d'équations linéaires $D(Aw - e\gamma) = e$. Ainsi, du point de vue mathématique, cette formule remplace la résolution d'un programme linéaire ou quadratique par la résolution d'un système non singulier d'équations linéaires, ce qui fut une idée initialement développée par Suykens et al. (Suykens & Vandewalle, 1999). Géométriquement, dans cette formulation alternative, les hyperplans $x'w - \gamma = \pm 1$ peuvent être considérés comme des hyperplans 'proximaux', poussés aussi loin que possible l'un de l'autre, et autour desquels les points de chaque classe sont rassemblés.

Les "proximal SVM" (PSVM) ont des performances comparables aux classifieurs SVM, mais avec une vitesse de calcul souvent bien plus rapide. Or, cette rapidité est indispensable dans notre algorithme basé sur un ensemble d'arbre, où dans certains cas quelques milliers de PSVM sont calculés. Notons que l'extension de PSVM linéaires à des PSVM non linéaires est testée dans (Fung & Mangasarian, 2001), mais n'est pas prise en considération dans ce travail. Fung et Mangasarian propose aussi dans (Fung & Mangasarian, 2005) un PSVM multi-catégories, qui peut être vu comme une stratégie "*un contre tous*". Leurs résultats sont comparés aux nôtres dans la section expérimentale.

2.2 Insertion des PSVM dans une structure d'arbre

De façon à pouvoir utiliser les PSVM à chaque nœud des arbres, la classe de chaque échantillon de données doit être convertie en une valeur binaire, correspondant à une des deux classes superposées choisies par le nœud. Ainsi, si les labels des classes à l'entrée d'un nœud sont 1, 2 et 3, la classe 1 peut par exemple être considérée comme la première catégorie de classes superposées, alors que la combinaison des classes 2 et 3

définit alors la deuxième catégorie de classes superposées. Le choix de ce partitionnement en deux catégories est réalisé de façon aléatoire. La seule contrainte imposée au partitionnement est d'éviter, dans la mesure du possible, d'avoir des catégories de tailles fortement inégales, ce qui peut entraîner une réduction significative des performances du PSVM (Fung & Mangasarian, 2005). En pratique, les classes d'entrée sont sélectionnées aléatoirement et progressivement pour former la première catégorie de classes superposées jusqu'à ce qu'au moins la moitié des échantillons ait été assignée à cette catégorie. Le label de la seconde catégorie est ensuite donné à toutes les autres classes. Evidemment, d'autres critères peuvent être envisagés pour construire ces deux catégories, mais nos résultats ont montré que cette approche générique et peu contraignante donne de bonnes performances au sein d'arbres aléatoires.

Pour le reste, notre système utilise une procédure classique descendante pour construire des arbres de décision non élagués à partir des données d'entraînement. Chaque nœud peut être vu comme un classifieur faible qui organise les échantillons d'entrée en deux branches, de manière à fournir un gain d'information significatif, c'est à dire réduisant l'impureté de la variable de sortie (Breiman *et al.*, 1984). Dans notre cas, cette partition s'effectue sur base d'un classifieur PSVM. Une fois que l'arbre entier a été construit, un échantillon à classifier tombe simplement de la racine à une des feuilles de l'arbre, et reçoit le label de la classe dominante parmi les échantillons d'entraînement qui ont atteint cette même feuille durant la phase d'apprentissage.

2.3 Ensemble d'arbres et processus aléatoires

De manière similaire à de nombreux travaux précédents (Breiman, 2001; Geurts *et al.*, 2006; Ho, 1998), une méthode utilisant de l'aléatoire est examinée afin d'améliorer les performances d'un arbre de décision. Ces méthodes proposent de rendre aléatoire ou semi-aléatoire la construction des arbres, et de créer ainsi un ensemble d'arbres diversifiés, dont les prédictions sont alors combinées par un simple vote majoritaire. Le but principal d'un ensemble aléatoire d'arbres de décision est de résoudre l'erreur d'approximation et l'erreur d'estimation en même temps. Dans notre structure, l'utilisation d'aléatoire permet non seulement de construire des arbres diversifiés, mais a aussi l'avantage de diminuer l'impact d'une décision non optimale à chaque nœud des arbres, en répartissant les classes d'entrée en deux catégories de classes superposées.

En pratique, la méthode proposée dans ce travail est proche de celle des sous-espaces aléatoires proposée par Ho dans (Ho, 1998). Celle-ci propose de choisir aléatoirement un sous-ensemble de N_a attributs parmi les n attributs possibles pour calculer un test à chaque nœud de l'arbre. Cette méthode recherche alors le meilleur test possible pour chacun de ces attributs pris individuellement (c.à.d. qu'il compare la valeur d'un attribut par rapport à un seuil), le caractère aléatoire étant alors inversement proportionnel au paramètre N_a . Similairement, durant notre processus d'apprentissage, N_a attributs sont sélectionnés au hasard à chaque nœud, parmi les n possibles. Ces attributs sont néanmoins utilisés tous ensemble par un PSVM afin de calculer les hyperplans. Ici aussi le degré d'aléatoire est proportionnel au paramètre N_a .

Le processus d'entraînement de chaque nœud d'un arbre est résumée dans l'algorithme 1. Dans celui-ci, m^{min} définit le nombre minimum d'échantillons d'entraîne-

ment que doit avoir chacune des classes superposées pour continuer la séparation d'un nœud, la taille de l'ensemble d'entraînement dans un nœud étant de m échantillons. Pour nos expériences, m^{min} est fixé à 5. N_a^{min} et N_a^{max} représentent les nombres d'attributs minimum et maximum à utiliser dans chaque *PSVM*, et N_a est alors une variable aléatoire pouvant varier entre N_a^{min} et N_a^{max} . m^{min} , N_a^{min} et N_a^{max} sont tous trois des paramètres influençant la complexité et la rapidité du classifieur.

Algorithme 1 (Entraînement d'un nœud N)

Split_a_node(S)

Entrée : un ensemble d'entraînement de m échantillons $S = (s_1, \dots, s_m)$ avec n attributs, N_c classes

Sortie : un nœud labélisé par les paramètres *PSVM* w et γ , et la branche de sortie de chaque échantillon d'entraînement

```

1:  $[S_1, S_2] = \text{Split\_in\_2\_categories}(S)$ 
2:  $m_1$  &  $m_2$  : nombre d'échantillons d'entraînement dans chaque catégorie ;
3:  $[a_1, \dots, a_{N_a}] = \text{Select\_attributes}(n)$ ;
4: if  $m_1$  &  $m_2 > m^{min}$  then
5:    $[w, \gamma] = \text{PSVM}(S_1, S_2, (a_1, \dots, a_{N_a}))$ ;
6:    $i = 1$ ;
7:   while  $i < m$  do
8:     if  $w * s_i - \gamma \leq 0$  then
9:        $s_i$  va dans le nœud de droite ;
10:    else
11:       $s_i$  va dans le nœud de gauche ;
12:    end if
13:     $i \leftarrow i + 1$ ;
14:  end while
15: else
16:    $N$  est une feuille ;
17: end if
```

Split_in_2_categories(S)

Entrée : un ensemble d'échantillons S avec N_c classes

Sortie : deux ensembles d'échantillons S_1 et S_2 avec 2 classes

```

1:  $[x_1, \dots, x_{N_c}] = \text{Nombre d'échantillons d'entraînement dans chaque classe}$ 
2:  $S1 = []$ ;  $S2 = []$ ;
3: while  $\text{taille}(S1) < m/2$  do
4:   Choisir aléatoirement une classe parmi les  $N_c$  à ajouter à  $S_1$ , sans remplacement
5: end while
```

Select_attributes(n)

Entrée : le nombre d'attributs n

Sortie : un ensemble d'indices liés aux attributs $a_1, \dots, a_{N_a}$

```

1: Choisir aléatoirement un nombre  $N_a$  entre  $N_a^{min}$  &  $N_a^{max}$ 
2: Choisir  $N_a$  indices d'attributs aléatoirement
```

$\text{PSVM}(S_1, S_2, (a_1, \dots, a_{N_a}))$

Entrée : les échantillons d'entraînement départagés en deux catégories avec les attributs choisis

Sortie : les paramètres *PSVM* w & γ

3 Résultats préliminaires

Afin d'analyser les performances des arbres à *PSVM*, nous avons utilisé des données provenant de la UCI Machine Learning Repository ². Ces données contiennent 4 à 617

²<http://archive.ics.uci.edu/ml/>

TAB. 1 – Performance à l’entraînement et au test (avec une 10-fold cross validation) pour le MPSVM linéaire, le SVM linéaire “un-contre-un”, les Extra-Trees et nos arbres à PSVM linéaire.

Dataset $m \times n$ nb classes	MPSVM linéaire Entraînement Test	UCU SVM linéaire Entraînement Test	ET ^d Entraînement Test	arbres à PSVM Entraînement Test
<i>Vowel</i> 528 × 10, 11 cl.	65.7 % 59.3 %	87.9 % 81.12 %	100 % 98.26 %	99.88 % 97.5 %
<i>Vehicle</i> 846 × 18, 4 cl.	79.3 % 77.2 %	83.8 % 79.7 %	100 % 74 %	100 % 84.02 %
<i>Segment</i> 2310 × 19, 7 cl.	91.2 % 91.0 %	96.9 % 93.37 %	100 % 98.23 %	99.95 % 98.23 %
<i>Isolet</i> 7797 × 617, 26 cl.	95.1 % 93.8 %	97.87 % 95.9 %	100 % 92.39 %	100 % 96.64 %
<i>Spambase</i> 4601 × 57, 2 cl.	94.65 92.2	95.45 % 92.89 %	100 % 95.83 %	98.93 % 95.96 %

attributs, et de 2 à 26 classes, couvrant ainsi de larges conditions.

Nous comparons à la table 3 les pourcentages de classification correcte de la méthode proposée avec les résultats obtenus en utilisant un ensemble d’arbres aléatoires et une méthode basée sur des PSVM. Les performances des PSVM multi-classes linéaires (MPSVM linéaires) sont tirées d’un article de Fung and Mangasarian (Fung & Mangasarian, 2005), alors que les performances d’arbres extrêmement aléatoires (c.à.d. les Extra-trees ET^d) sont reprises d’un article de Geurts (Geurts *et al.*, 2006). Nous montrons également les résultats obtenus par un SVM linéaire “un-contre-un” (UCU SVM), disponible dans la OSU SVM Classifier Matlab Toolbox ³.

Le nombre d’arbres est fixé à 100 pour les arbres à PSVM ainsi que pour les Extra-trees. La table 3 résume les résultats obtenus, m étant le nombre d’échantillons de données, et n le nombre d’attributs. La comparaison des arbres à PSVM avec les autres méthodes montre une amélioration non négligeable des performances sur les ensembles de données où les méthodes reposant sur des SVM surpassent les méthodes à base d’arbres (c.a.d. les données Vehicle et Isolet). Pour les autres ensembles de données, nos performances sont comparables aux performances des Extra-trees. Grâce à la rapidité de PSVM, le temps d’entraînement de notre méthode est la plupart du temps similaire aux ensembles d’arbres aléatoire, dont la rapidité est démontrée dans (Geurts *et al.*, 2006). Nous n’avons délibérément pas ajouté dans les résultats obtenus les temps de calcul mis par les différentes méthodes, car celles-ci ont été implémentées dans différents langages et sur différentes machines. Néanmoins, sur les données identiques à celles utilisées dans ce travail, Fung et Mangasarian annonce une différence de vitesse supérieur à un ordre de magnitude entre un SVM linéaire et un PSVM linéaire, en utilisant dans les deux cas une stratégie “un-contre-tous”.

³<http://www.kernel-machines.org/>

4 Conclusion et travaux futurs

Nous avons proposé dans cet article d'insérer des PSVM à chaque nœud d'un ensemble d'arbres aléatoires, de façon à résoudre des problèmes de classification de données de grande dimension et multi-classes. Les résultats préliminaires obtenus montrent que cette méthode permet d'obtenir sur les données analysées, de meilleurs résultats par rapport aux méthodes basées sur des SVM linéaires ou celles utilisant des ensembles d'arbres aléatoires. Ce bénéfice vient de (1) la capacité à prendre plusieurs attributs en compte à chaque nœud des arbres ; et (2) la capacité à convertir un problème multi-classe en une hiérarchie de classifications binaires, sans avoir à prendre de décisions strictes sur la façon de partitionner les classes multiples en un ensemble binaire. Des travaux futurs pourraient effectuer une comparaison plus exhaustive, avec plus de données (de plus grandes dimensions) et différents types de classifieurs. Ils pourraient aussi examiner le bénéfice obtenu en utilisant des noyaux non linéaires dans les PSVM, ou encore analyser comment choisir un meilleur compromis entre le degré d'aléatoire et une sélection optimale d'attributs et/ou de classes superposées à chaque nœud.

Références

- BIAU G., DEVROYE L. & LUGOSI G. (2008). Consistency of random forests and other averaging classifiers. *J. Mach. Learn. Res.*, **9**, 2015–2033.
- BREIMAN L. (2001). Random forests. *Machine Learning*, **45**(1), 5–32.
- BREIMAN L., FRIEDMAN J., OLSEN R. & STONE C. (1984). *Classification and Regression Trees*. CRC Press. ISBN 0412048418.
- CHENG L., ZHANG J., YANG J. & MA J. (2008). An improved hierarchical multi-class support vector machine with binary tree architecture. In *International Conference on Internet Computing in Science and Engineering (ICICSE'08)*, p. 106–109, Harbin, China.
- CHEONG S., OH S. & LEE S.-Y. (2004). Support vector machines with binary tree architecture for multi-class classification. *Neural Information Processing Letters and Reviews*, **2**(3), 47–51.
- CRAMMER K. & SINGER Y. (2001). On the algorithmic implementation of multi-class kernel-based vector machines. *Journal of Machine Learning Research*, **2**, 265 – 292.
- FUNG G. & MANGASARIAN O. (2001). Proximal support vector machine classifiers. In *International Conference on Knowledge Discovery and Data Mining*, p. 64–70, San Francisco, CA, USA.
- FUNG G. & MANGASARIAN O. (2005). Multicategory proximal support vector machine classifiers. *Machine Learning*, **59**(1-2), pp. 77–97(21).
- GEURTS P., ERNST D. & WEHENKEL L. (2006). Extremely randomized trees. *Machine Learning*, **63**(1), 3–42.
- HO T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(8), 832–844.
- HSU C.-W. & LIN C.-J. (2002). A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, **13**(2), 415–425.

- LEE J.-Y. & OLAFSSON S. (2006). Multi-attribute decision trees and decision rules. *in Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques*, p. 327 – 358.
- PLATT J., CRISTIANINI N. & SHAWE-TAYLOR J. (2000). Large margin dags for multiclass classification. *Advances in Neural Information Processing Systems*, **12**, 547 – 553.
- SUYKENS J. A. K. & VANDEWALLE J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, **9**(3), 293–300.
- TIKHONOV A. & ARSENIN V. (1977). *Solutions of Ill-Posed Problems*. New York : John Wiley & Sons.
- VAPNIK V. N. (1995). The nature of statistical learning theory. In *Springer*, New York, USA.