A Sequential Minimal Optimization Algorithm for the All-Distances Support Vector Machine*

Diego Candel[1], Ricardo Ñanculef[1], Carlos Concha[1], and Héctor Allende[1,2]

[1] Universidad Técnica Federico Santa María,
Departamento de Informática, CP 110-V Valparaíso, Chile
{dcontard,jnancu,cconcha,hallende}@inf.utfsm.cl
[2] Universidad Adolfo Ibáñez, Facultad de Ingeniería y Ciencia, Santiago, Chile
hallende@uai.cl

Abstract. The All-Distances SVM is a single-objective light extension of the binary μ-SVM for multi-category classification that is competitive against multi-objective SVMs, such as *One-against-the-Rest* SVMs and *One-against-One* SVMs. Although the model takes into account considerably less constraints than previous formulations, it lacks of an efficient training algorithm, making its use with medium and large problems impracticable. In this paper, a *Sequential Minimal Optimization*-like algorithm is proposed to train the All-Distances SVM, making large problems abordable. Experimental results with public benchmark data are presented to show the performance of the AD-SVM trained with this algorithm against other single-objective multi-category SVMs.

Keywords: Kernel Machines, Multi-category Classification, Support Vector Machines, Sequential Minimal Optimization.

1 Introduction

Support Vector Machines [20] (SVMs) are currently well known methods for pattern recognition and other data analysis, with strong theoretical properties and practical results when applied to real-world problems. Originally formulated to deal with linearly separable binary classification problems, they can also deal with noisy data and non-linearly separable cases using a regularization and a kernel method extension respectively.

Although the training of these machines can be assumed as finding the solution to a quadratic optimization problem with linear restrictions, traditional approaches are impractical due to the dense nature of the Hessian Matrix involved in the problem definition. To deal with this, chunking and decomposition algorithms have been proposed through time, being the Sequential Minimal Optimization (SMO) [18,14,10] one of the most popular methods employed for this purpose.

* This work was supported in part by Research Grant FB0821 "Centro Científico Tecnológico de Valparaíso" UTFSM and by DGIP-UTFSM Grant.

I. Bloch and R.M. Cesar, Jr. (Eds.): CIARP 2010, LNCS 6419, pp. 484–491, 2010.

In a multi-category context, the use of these training algorithms is straight forward when several binary SVMs are used in combination (a multi-objective approach), as with the *One-against-the-Rest* scheme [20] and the *One-versus-One* scheme [15]. The same is not true when the classifier is a single machine extending a binary SVM to classify more than two classes (a mono-objective approach), since the objective function of the machine has changed and the underlying components in which the solvers relay on are not the same (like the Karush-Kuhn-Tucker conditions used by the SMO algorithm). In these cases, a suitable training algorithm needs to be designed to address the single-objective formulation of this new kind of machines. This is the case of the method of Weston and Watkins [21], the framework of Crammer and Singer [8,7] and the All-Distances SVM (AD-SVM) [17], among others. Here we focus on AD-SVMs, a method recently proposed to formulate the multi-category problem using a reduced number of constraints.

An efficient solver for the AD-SVMs does not currently exist, and only general-purpose solvers like the one proposed in [6] have been employed until now, making possible the use of this machine only for small problems (no more than 500 training examples). The use of general-purpose solvers gets impractical as the problem size grows, since training time and memory requirements scale above a quadratic rate, due to the dense Hessian Matrix issue mentioned above. Therefore a solver specifically designed for the AD-SVM is needed.

In this paper, a specific algorithm to train the AD-SVM is proposed. Its design, derivation and components are based in the SMO algorithm for binary SVMs. The SMO was chosen as base for this new solver as it is a fast and well-known algorithm commonly used in SVM training. The performance of the new solver is compared against other multi-category mono-objective machine (described in [7]) both in terms of accuracy and training time efficiency.

The rest of the paper is organized as follow: An overview of binary SVMs and the AD-SVM is given in section 2; The components and the general structure of the new training algorithm for the AD-SVM are described in section 3; Finally, experiments and conclusions are provided in section 4.

2 Background

Given a set of examples $S = \{\mathbf{x}_i : i \in I\} \subset \mathcal{X} \subset \mathbb{R}^n$ of two classess, $\mathcal{C}_-$ and $\mathcal{C}_+$, the binary classification problem asks to learn a decision function $f(\mathbf{x}) : \mathcal{X} \to \{-1, +1\}$ to distinguish patterns of one class from the other class. SVMs accomplish this by modeling the boundary between $\mathcal{C}_-$ and $\mathcal{C}_+$ as the hyperplane $H = \{x : \mathbf{w}^T\mathbf{x}+b = 0\}$ whose parameters $\mathbf{w}$ and b are determined by minimizing a risk functional [20]. To deal with non linearly separable data, SVMs use non-linear kernel functions $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$ instead of the linear inner products $\mathbf{x}i^T\mathbf{x}_j$ and have different ways to treat noisy data through the use of slag variables, being the C-SVM and the ν-SVM [19] two of the most popular approaches.

The All-Distances SVM (AD-SVM) [17] can be considered the natural extension of the μ-SVM [9] to multiple-classes. The extension consists in minimizing

the sum of all pairwise distances among the different K convex hulls (each generated for one class). Its dual form can be stated as follows:

$$D(\mathbf{u}) : \mathbf{minimize}_{\{\mathbf{u}\}} \frac{1}{4} \sum_{i \in I} \sum_{j \in I} u_i \overline{\mathbf{k}}_{ij} u_j \tag{1}$$

$$\text{subject to } \sum_{i \in I_r} u_i = 1,\ r \in \{1, \cdots, K\} \ \wedge\ 0 \leq\ u_i \leq \mu,\ \forall i \in I, \tag{2}$$

where $\overline{\mathbf{k}}_{ij} = \alpha_{ij} \mathbf{k}_{ij}$, $\mathbf{k}_{ij} = \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$, I is the set of all indexes and I_r is the subset of indexes belonging only to class $\mathcal{C}_r$. Values α_{ij} involved in the definition of $\overline{\mathbf{k}}_{ij}$ are defined as:

$$\alpha_{ij} = \mathbf{y}_i \cdot \mathbf{y}'_j = \begin{cases} K-1, \text{ if } i \in I_r \wedge j \in I_r \\ -1, \text{ in any other case,} \end{cases} \tag{3}$$

$$\text{where } \mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iK}], \quad y_{is} = \begin{cases} K-1, \text{ if } s = r, \text{ where } i \in I_r \\ -1, \text{ in any other case.} \end{cases} \tag{4}$$

Note that the labels are not scalar values but K-dimensional vectors, equivalent to those proposed in the formulation of other multi-category classifier [16]. Note also that, when $K = 2$, the possible values of α_{ij} are the same that those calculated with the scalar y_i labels used within the binary SVM [9]. The formulation lead to one hyperplane $\mathbf{w}_r$, one offset b_r and one ρ_r for every class $r \in \{1, \cdots, K\}$:

$$\mathbf{w}_r = \frac{1}{K} \sum_{i \in I} \alpha_{ir} u_i \mathbf{x}_i,\ b_r = \frac{-1}{K^2} \sum_{i \in I} \sum_{j \in I} u_i \alpha_{ir} \mathbf{k}_{ij} u_j, \tag{5}$$

$$\rho_r = \frac{1}{K^2} \sum_{i \in I} \sum_{j \in I} u_i \alpha_{ir} \mathbf{k}_{ij} \alpha_{jr} u_j, \text{ where } \alpha_{ir} = \begin{cases} K-1, \text{ if } i \in I_r \\ -1, \text{ in any other case.} \end{cases} \tag{6}$$

As with the α_{ij} values, for $K = 2$, $\mathbf{w}_1 = \mathbf{w}_2 = \mathbf{w}$, $\rho_1 = \rho_2 = \rho$ and $b_2 = -b_1 = b$, that is the hyperplanes model the half-spaces induced by the binary SVM hyperplane. With this elements, the decision funtion $f(\cdot)$ used to classify new examples is given by

$$f(\mathbf{x}) = \arg\max_r \left(\frac{1}{K} \sum_{i \in I} \alpha_{ir} u_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) + b_r - \rho_r \right). \tag{7}$$

which again coincides with the binary SVM decision function for $K = 2$ [9].

3 SMO Algorithm for the AD-SVM

The SMO scheme to train binary SVMs works iterating through a sequence of steps until convergence is reached. At every step, only two variables are selected for optimization and the others are temporary frozen. An algorithm of this kind

requires of the following principal components: an (usually) analytic **optimization step** to calculate new values for two Lagrange multipliers; a **selection strategy** (heuristic or not) to choose these two Lagrange multipliers, so that the convergence to the optimum is as fast as possible in every step; a **stopping criteria** to efficiently determine when the optimal (or a near optimal) solution has been achieved; an **update system** that efficiently updates the values involved in the selection and optimization of two Lagrange multipliers, every time that the vector of Lagrange multipliers is changed; and an algorithm that utilizes all of the later components to achieve the optimal or near optimal solution of the SVM problem. Here, extensions of each of these components are given to define a functional SMO solver to train the AD-SVM. We start by defining the subsets I_r^0, I_r^1 and I_r^2

$$\mathrm{I}_r^0 = \{i : i \in \mathrm{I}_r,\ 0 < u_i < \mu\}\,,\ \mathrm{I}_r^1 = \{i : i \in \mathrm{I}_r,\ u_i = 0\}\,,\ \mathrm{I}_r^2 = \{i : i \in \mathrm{I}_r,\ u_i = \mu\}\,,$$

and the quantities β_r^{up} and β_r^{low}

$$\beta_r^{\mathrm{up}} = \min\left\{\mathrm{F}_i\ ,\ i \in \mathrm{I}_r^{\mathrm{up}} := \mathrm{I}_r^0 \cup \mathrm{I}_r^1\right\},\ \beta_r^{\mathrm{low}} = \max\left\{\mathrm{F}_i\ ,\ i \in \mathrm{I}_r^{\mathrm{low}} := \mathrm{I}_r^0 \cup \mathrm{I}_r^2\right\}$$
$$\text{where } \mathrm{F}_i = \sum_{j \in I} u_j \overline{\mathbf{k}}_{ij}.$$

These elements will be useful for the definition of the SMO components.

3.1 Stopping Criteria

At any given moment of the training, it is useful and necesary to know if optimality has been reached. As demonstrated in [13] for the binary case and futher extended in [3] for multi-category instances, when $\beta_r^{\mathrm{low}} - \beta_r^{\mathrm{up}} \leq 0$ for all $r \in \{1, \ldots, K\}$ classes, the algorithm has reached its optimum. Since it is not always possible to achieve optimality due to the limits of computer arithmetics and other numerical issues, a tolerance $\tau > 0$ is conveniently defined by the user. If well defined, the use of this tolerance also allows a faster convergency of the algorithm at expenses of a low precision loss. With this in mind, a τ-tolerance optimum is achieved when $\beta_r^{\mathrm{low}} - \beta_r^{\mathrm{up}} \leq 2\tau$.

3.2 Selection Strategy

If the algorithm has not achieved optimality, it means that at least one pair of indexes $\{i,\ j\}$ in a class r is violating optimality, *i.e.* $\mathrm{F}_j - \mathrm{F}_i > 2\tau$, with $i \in \mathrm{I}_r^{\mathrm{up}}$ and $j \in \mathrm{I}_r^{\mathrm{low}}$. Most of the time, there will be several of these violating pairs, and choosing the most violating one at each step will lead to a faster convergence.

Here, we implement an extension for the AD-SVM of the heuristic proposed in [10] that uses second order information: For the index $i = \arg\min_t \{\mathrm{F}_t\ ,\ t \in \mathrm{I}_r^{\mathrm{up}}\}$ of every class r, find index j such that

$$j = \arg\max_t \left\{ \frac{b_{it}^2}{a_{it}}\ ,\ t \in \mathrm{I}_r^{\mathrm{low}} \wedge \mathrm{F}_t > \mathrm{F}_i \right\} \tag{8}$$

$$\text{where } a_{it} = \overline{\mathbf{k}}_{ii} - 2\overline{\mathbf{k}}_{it} + \overline{\mathbf{k}}_{tt},\ b_{it} = \mathrm{F}_t - \mathrm{F}_i \tag{9}$$

Select the pair of indexes $\{i, j\}$ among all K classes such that the factor b_{it}^2/a_{it} is maximal. Note that a similar strategy can be followed with $i = \arg\max_t \left\{ \mathrm{F}_t \ , \ t \in \mathrm{I}_r^{\mathrm{low}} \right\}$.

If the kernel function is not positive definite, there will be cases in which a_{it} will adopt problematic values ($a_{it} \leq 0$). It has been shown in [5] for binary SVMs that in these cases the value of a_{it} can be set to a very small positive value $0 < \varepsilon \ll 1$, redefining the problem as convex and thus it can be solved in the same way as when $a_{it} > 0$. In the experiments that have been carried for the AD-SVM, this strategy has worked in the same way as expected for the binary case.

3.3 Optimization Step

As it is shown in [3], when two Lagrange multipliers u_i, u_j exist whose indexes are a violating pair, new values can be analytically calculated in order to achieve optimality for the problem when all other variables are left constant. Here, we start by calculating the new Lagrange multiplier for j, as $u_j^{\mathrm{new}} = u_j - \frac{b_{ij}}{a_{ij}}$. Note that, since a_{ij} and b_{ij} were already calculated for selecting the pair $\{i, j\}$ in (8), they do not need to be recomputed here. Also note that u_j^{new} needs to be clipped to satisfy its boundary constraints, that is

$$u_j^{\mathrm{new,clipped}} = \begin{cases} \mathrm{L}, & \text{if } u_j^{\mathrm{new}} \leq \mathrm{L} \\ u_j^{\mathrm{new}}, & \text{if } \mathrm{L} < u_j^{\mathrm{new}} < \mathrm{H} \\ \mathrm{H}, & \text{if } u_j^{\mathrm{new}} \geq \mathrm{H} \end{cases} \tag{10}$$

$$\text{where } L = \max\left\{0, (\gamma - \mu)\right\}, \ H = \min\left\{\gamma, \mu\right\}, \ \gamma = u_i + u_j \tag{11}$$

Now u_i^{new} can be computed as $u_i^{\mathrm{new}} = \gamma - u_j^{\mathrm{new}}$. Since u_i^{new} also needs to fulfill boundary constraints, it must be clipped as u_j^{new} was. After this step, the new Lagrange multipliers $u_i^{\mathrm{new,clipped}}$ and $u_j^{\mathrm{new,clipped}}$ are returned to the main algorithm.

3.4 F_t's Update

As with traditional SMO algorithms, F_t's values can be updated efficiently after new values of a pair of Lagrange multipliers are calculated:

$$\mathrm{F}_t^{\mathrm{new}} = \mathrm{F}_t + \left(u_i^{\mathrm{new,clipped}} - u_i\right) \cdot \overline{\mathbf{k}}_{it} + \left(u_j^{\mathrm{new,clipped}} - u_j\right) \cdot \overline{\mathbf{k}}_{jt} \tag{12}$$

3.5 Algorithm Structure

The components of the SMO procedure just defined are organized in algorithm 1.1.

Algorithm 1.1 is a very general implementation of the SMO algorithm. In practice, the SMO implemented for the experiments of this contribution [2] follows a scheme similar to those proposed in [18] or [14], where each iteration in the training process works first by using only the I_r^0 set of each class, and then, in a second stage, optimality is checked with all remaining indexes. Also, a LRR cache strategy is used to store $\mathbf{k}_{ij}$ products.

Algorithm 1.1. SMO Algorithm for the AD-SVM

1: Initialize u satisfying constraints stated in 2.
2: Calculate $\mathrm{F}_t,\ \forall t \in I$.
3: Find β_r^{low} and β_r^{up} for each class.
4: **while** $\beta_r^{\mathrm{low}} - \beta_r^{\mathrm{up}} > 2\tau$ for at least one $r \in \{1, \ldots, K\}$, **do**
5: For every class r, select $i = \arg\min_t \{\mathrm{F}_t\ ,\ t \in \mathrm{I}_r^{\mathrm{up}}\}$.
6: For the selected $i \in I_r$, select $j = \arg\max_t \left\{ \frac{b_{it}^2}{a_{it}}\ ,\ t \in \mathrm{I}_r^{\mathrm{low}} \wedge \mathrm{F}_t > \mathrm{F}_i \right\}$.
7: Select the pair of Lagrange multipliers $\{u_i, u_j\}$ with maximal $\frac{b_{ij}^2}{a_{ij}}$ among all the classes.
8: Calculate $\left\{ u_i^{\mathrm{new,clipped}}, u_j^{\mathrm{new,clipped}} \right\}$.
9: Update $\mathrm{F}_t,\ \forall t$.
10: Find new β_r^{low} and new β_r^{up} for each class.
11: **end while**

4 Experiments and Conclusions

Experiments were conducted to measure and compare the classification accuracy and training runtime of the AD-SVM trained with the proposed SMO algorithm against the Multi-Category SVM proposed in [7] (MC-SVM). In order to obtain a platform and implementation independent comparison, the *number of kernel calls*[1] was used instead of the execution time to measure runtime complexity. Also, no *cooling* of the tolerance[2] was used in any of the algorithms.

A RBF Kernel $\mathbf{K}(\mathbf{x}_j, \mathbf{x}_j) = e^{\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2\right)}$ was used in all the experiments, with parameter σ. To find optimal values for μ and σ, a grid search was performed using k-fold cross-validation, with $k = 10$ folds for relatively small datasets (Glass, Vowel, Satimage, Shuttle small, Letter small, MNIST small) and $k = 5$ folds for relatively large datasets (USPS, Letter, Shuttle).

The values tested for hyper-parameters correspond as usual to a regular logarithmic grid in base 2: for σ it was $\left\{2^{-4/2}, 2^{-3/2}, \ldots, 2^{9/2}, 2^{10/2}\right\}$ and for μ, $\left\{1, 2^{-(1\cdot\log_2(m_s)/14)}, \ldots, 2^{-(13\cdot\log_2(m_s)/14)}, 2^{-(14\cdot\log_2(m_s)/14)}\right\}$, where $m_s = \min\left(\overline{\overline{I_1}}, \overline{\overline{I_2}}, \ldots, \overline{\overline{I_K}}\right)$ is the size of the smaller class in the training sets of the cross-validation folds. The values for μ obey to the observation that values lower than $2^{-(14\cdot\log_2(m_s)/14)} = 1/m_s$ lead to an infeasible optimization problem, while values greater than 1 do not change the feasible space. In the case of the MC-SVM, a parameter B must be set instead of μ. The same values for σ were tested, with $B \in \left\{2^{-2}, 2^{-1}, \ldots, 2^{11}, 2^{12}\right\}$.

[1] The *number of kernel calls* counts every time a $\mathbf{k}_{ij}$ product is used in the algorithm, either being calculated in the moment or retrieved from the cache.

[2] The *cooling* of the tolerance is the iterative refinement of the numerical tolerance until a desired precision is obtained.

The usps [11], Glass, Vowel, Satimage, Shuttle and Letter datasets [1] were used in their normalized and publicly available versions [4] (this reference also provides datasets descriptions). The training datasets for Shuttle small, Letter small and MNIST small are all subsets of 5000 examples randomly selected from the original datasets. Unlike the others, the Glass dataset does not have a separated test set. In this case, 5-fold cross-validation with the whole dataset was used to evaluate test performance. Results obtained with the values of hyper-parameters selected in the cross-validation procedure are listed in table 1.

Table 1. Experiment Results

Datasets	Machine	σ	B & μ	Test Acc. %	Kernel Calls
Glass	MC-SVM	0.707	0.5	72.89	2.96×10^6
$m_s = 8$	AD-SVM	2.828	0.125	**80.61**	$\mathbf{1.97 \times 10^5}$
Vowel	MC-SVM	0.25	0.25	**50.87**	$\mathbf{7.43 \times 10^5}$
$m_s = 43$	AD-SVM	2	0.068	44.81	2.97×10^6
Satimage	MC-SVM	2	0.5	**91.40**	2.60×10^9
$m_s = 374$	AD-SVM	2.828	0.034	88.30	$\mathbf{4.49 \times 10^7}$
Shuttle small	MC-SVM	2	0.25	**99.79**	3.46×10^8
$m_s = 5$	AD-SVM	4	0.224	99.71	$\mathbf{2.85 \times 10^7}$
Letter small	MC-SVM	2.828	0.25	**63.73**	7.02×10^{10}
$m_s = 1$	AD-SVM	2.828	1	60.82	$\mathbf{1.19 \times 10^8}$
MNIST small	MC-SVM	16	1	**99.04**	5.05×10^8
$m_s = 407$	AD-SVM	32	0.032	93.55	$\mathbf{8.14 \times 10^7}$
USPS	MC-SVM	5.657	0.5	**95.37**	5.48×10^9
$m_s = 433$	AD-SVM	32	0.272	93.21	$\mathbf{1.22 \times 10^8}$
Letter	MC-SVM	2.828	0.25	**71.31**	1.24×10^{12}
$m_s = 1$	AD-SVM	2	1	65.66	$\mathbf{8.43 \times 10^8}$
Shuttle	MC-SVM	0.707	0.25	**99.90**	2.26×10^9
$m_s = 5$	AD-SVM	4	0.447	99.89	$\mathbf{5.15 \times 10^8}$

As it can be noted, in most cases the number of kernel calls is bigger for the MC-SVM than the AD-SVM, with a difference in order of magnitude of at least 1. In the classification performance, the situation changes, exhibiting the MC-SVM a better classification accuracy most of the time. This is expected, since the AD-SVM is a light extension of the μ-SVM with a number of constraints significatively lower than MC-SVM. Nevertheless, note that the difference is not larger than 6%.

Further work can be done to improve the time performance of the algorithm applying for example *tolerance cooling* or *dinamic shrinking* techniques [12]. The theoretical analysis of the algorithm can be also expanded concerning the algorithm: We believe that the convergence proof for the binary SMO presented in [13] can be extended to this multi-category SMO. The same can be said about the redefinition of the term a_{ij} explained at the end of subsection 3.1 to handle semi-definite or indefinite kernels.

References

1. Blake, C., Merz, C.: UCI repository of machine learning databases (1998), http://mlr.cs.umass.edu/ml/index.html
2. Candel, D.: Source code of the smo algorithm for the ad-svm, http://git.inf.utfsm.cl/?p=dcontard.git;a=summary
3. Candel, D.: Algoritmo tipo SMO para la AD-SVM aplicado a Clasificación Multicategoría. Master's thesis, Universidad Técnica Federico Santa María, Valparaíso, Chile (2010), http://www.alumnos.inf.utfsm.cl/~dcontard/tesis.pdf
4. Chang, C.C., Lin, C.J.: Libsvm data: Classification, multi-class (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html
5. Chen, P.H., Fan, R.E., Lin, C.J.: A study on SMO-type decomposition methods for support vector machines. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council 17(4), 893–908 (2006)
6. Coleman, T.F., Li, Y.: A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. SIAM J. on Optimization 6(4), 1040–1058 (1996)
7. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research (2), 265–292 (2001)
8. Crammer, K., Singer, Y.: On the learnability and design of output codes for multiclass problems. Machine Learning 47(2-3), 201–233 (2002)
9. Crisp, D., Burges, C.: A Geometric Interpretation of ν-SVM Classifiers. In: Advances in Neural Information, vol. (12), pp. 244–250. MIT, Cambridge (2000)
10. Fan, R.e., Chen, P.h., Lin, C.j.: Working Set Selection Using Second Order Information for Training Support Vector Machines. Journal of Machine Learning Research 6, 1889–1918 (2005)
11. Hull, J.J.: A database for handwritten text recognition research. IEEE Trans. Pattern Anal. Mach. Intell. 16(5), 550–554 (1994)
12. Joachims, T.: Making large-scale support vector machine learning practical (1998)
13. Keerthi, S., Gilbert, E.: Convergence of a Generalized SMO Algorithm for SVM Classifier Design. Machine Learning 46(1), 351–360 (2002)
14. Keerthi, S., Shevade, S., Murthy, K., Bhattacharyya, C.: Improvements to Platt's SMO Algorithm for SVM Classifier Design. Neural Computation 13(3), 637–649 (2001)
15. Kressel, U.: Pairwise classification and support vector machines. In: Advances in kernel methods: support vector learning, pp. 255–268. MIT Press, Cambridge (1999)
16. Lee, Y., Li, Y., Wahba, G.: Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. Journal of the American Statistical Association 99(465), 67–81 (2004)
17. Ñanculef, R., Concha, C., Allende, H., Candel, D., Moraga, C.: Ad-svms: A light extension of svms for multicategory classification. International Journal of Hybrid Intelligent Systems 6(2), 69–79 (2009)
18. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization, pp. 185–208 (1999)
19. Scholkopf, B., Smola, A., Williamson, R., Bartlett, P.: New support vector algorithms. Neural computation 12(5), 1207–1245 (2000)
20. Vapnik, V.N.: The nature of statistical learning theory. Springer, Heidelberg (1995)
21. Weston, J., Watkins, C.: Support vector machines for multiclass pattern recognition. In: Proceedings of the Seventh ESSAN, pp. 219–224 (1999)