On-line Independent Support Vector Machines

Francesco Orabona [a], Claudio Castellini [b], Barbara Caputo [a],

Luo Jie [a,c], Giulio Sandini [b,d]

[a] *Idiap Research Institute, Centre du Parc, Rue Marconi 19, P.O. Box 592 — CH-1920 Martigny, Switzerland*

[b] *LIRA-Lab, DIST, University of Genova, viale F. Causa, 13, 16145 Genova, Italy*

[c] *École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland.*

[d] *Italian Institute of Technology, Robotics, Brain and Cognitive Sciences Dept., via Morego 30, 16163 Genova, Italy.*

Abstract

Support Vector Machines (SVMs) are one of the most successful algorithms for classification. However, due to their space and time requirements, they are not suitable for on-line learning, that is, when presented with an endless stream of training observations.

In this paper we propose a new on-line algorithm, called On-line Independent Support Vector Machines (OISVMs), which approximately converges to the standard SVM solution each time new observations are added; the approximation is controlled via a user-defined parameter. The method employs a set of linearly independent observations and tries to project every new observation onto the set obtained so far, dramatically reducing time and space requirements at the price of a negligible loss in accuracy. As opposed to similar algorithms, the size of the solution obtained by OISVMs is always bounded, implying a bounded testing time. These statements are supported by extensive experiments on standard benchmark databases as well as on two real-world applications, namely place recognition

by a mobile robot in an indoor environment and human grasping posture classification.

Key words: Support Vector Machines, on-line learning, bounded testing complexity, linear independence

---

1 Introduction

Support Vector Machines (SVMs) [1] are one of the most popular and promising classification algorithms. As opposed to other learning methods such as, e.g., neural networks, they are strongly theoretically founded, and have been shown to enjoy excellent performance in several applications (see, e.g., [2]). One framework, though, in which their power has not yet been fully developed is *on-line* learning. A system involved in on-line learning must face a potentially endless flow of training data, updating its knowledge after each new sample. This setting is particularly difficult for SVMs as the size of an SVM solution grows linearly with the number of training samples taken into account [3]. Since any real system has access to finite resources (e.g., finite computational power, memory etc.), a strategy to limit the number of data points is required, and a trade-off to accuracy must be accepted. So, it becomes crucial to find a way to save resources while obtaining an acceptable approximation of the ideal (batch) system.

In this paper we describe a new on-line algorithm, On-line Independent SVMs (OISVMs), which approximately converges to the ideal, batch SVM solution. Sim-

Email addresses: forabona@idiap.ch (Francesco Orabona), claudio.castellini@unige.it (Claudio Castellini), bcaputo@idiap.ch (Barbara Caputo), jluo@idiap.ch (Luo Jie), giulio.sandini@iit.it (Giulio Sandini).

ilar to other kernel-based discriminative on-line algorithms, OISVMs construct the hypothesis via a subset of the samples seen so far called *basis*; new samples are put in the basis only if they are linearly independent in the feature space from the current basis. The approximate solution obtained can be controlled via a user-defined parameter. As opposed to similar algorithms, the size of the solution found in our case is *always* bounded, implying a bounded testing time. The reduction in time and space requirement is on average dramatic, at the price of a negligible loss of accuracy. We show both theoretically and empirically that the size of the basis *does not grow* linearly with the training set, but converges to a limit size and then stops growing. OISVMs produce smaller models compared to the standard SVMs, with a training complexity which is asymptotically quadratic in the number of samples, and with bounded testing time. Moreover, they reach near-optimal performance while retaining the good generalization power of standard SVMs.

These statements are supported by an extensive evaluation on standard benchmark databases as well as on two real-world applications, namely (a) place recognition by a mobile robot in an indoor environment, and (b) human grasping posture classification.

The paper is organized as follows. Section 2 sets up the problem and describes our method. Section 3 describes the experimental results, and Section 4 concludes the paper and outlines some future work.

Related work

Attempts to cast the SVMs as an on-line learning algorithm have already appeared (e.g., [4–7]), but in all cases there is no attempt to reduce the growth of the solution.

After-training simplification methods aiming to "shrink" the obtained solution (e.g. [8, 9], chapter 18.3 in [10]) are too computationally expensive in an on-line setting where a new solution must be produced after each new sample. Also, the lack of knowledge of the complete data set typical of on-line learning rules out a number of methods, such as, e.g., kernel matrix low rank approximation methods based on Incomplete Cholesky Factorization [11–13].

Other methods have been proposed to heuristically select a subset of the vectors as basis, e.g., in [14, 15]. In [16], instead, a method to directly build a "vocabulary" of vectors is proposed, but the formulation is not convex. On the other hand, several on-line kernel-based methods have been proposed a priori bounding the complexity of the predictor [17–20]. All of them use a sort of stochastic gradient descent procedure to train a classifier with a fixed size. However the resulting loss of performance compared to standard SVM makes these algorithms unsuitable for most real-world applications. A strategy similar to the one we propose is used in [21], but again the training is achieved through a variant of the stochastic gradient descent procedure. Another possibility is to directly optimize the primal formulation of SVM, but this strategy cannot be used with infinite dimensional kernels.

2 Problem Setting

In this Section we first introduce on-line learning as a general scenario; we then instantiate it for SVMs and then proceed to describe our adaptation of SVMs in the on-line framework. We will always be referring to the problem of *classification*.

In standard learning (*batch*), a set of (sample,label) pairs drawn from an unknown probability distribution is given in advance (*training set*); the task is to find a func-

tion (*hypothesis*) such that its sign best determines the label of any future sample drawn from the same distribution. As opposed to this, in on-line learning samples and labels are made available in time, so that no knowledge of the training set can be assumed a priori. The hypothesis must therefore be built incrementally every time a new sample is available. Let us call this operation of building a new hypothesis, a *round*.

Formally, let $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$, with $l \in \mathbb{R}^+$, $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$, be the *full* training set, and let h_i denote the hypothesis built at round i, when only the (sample,label) pairs up to i are available. At the next round, a new sample $\mathbf{x}_{i+1}$ is available and its label is predicted using h_i. The true label y_{i+1} is then matched against this prediction, and a new hypothesis h_{i+1} is built taking into account the loss incurred in the prediction. In the end, for any given sequence of samples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)$, a sequence of hypotheses $h_1, \cdots, h_l$ is built such that h_i depends only on h_{i-1} and $(\mathbf{x}_i, y_i)$.

Note that any standard machine learning algorithm can be adapted to work in the on-line setting just retraining from scratch each time a new sample is acquired. However this would result in an extremely inefficient algorithm.

In the following we sketch the theory of SVM that gives us the tools to extend it to the on-line setting in an efficient way; the interested reader should refer to, e.g., [2] for a comprehensive treatment of the subject.

2.1 Support Vector Machines

If one assumes the samples are separable via a linear function in $\mathbb{R}^m$ (hyperplane) $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, with $\mathbf{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$, the SVM algorithm finds the one for

which $\|\mathbf{w}\|$ is minimum, enforcing the constraints $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0$. In case the samples are not linearly separable, l slack variables $\xi_i \geq 0$ are introduced and

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} \xi_i^p \tag{1}$$

is sought for, subject to the constraints $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$, where $C \in \mathbb{R}^+$ is an error penalty coefficient and p is usually 1 or 2. The problem is compactly expressed in Lagrangian form by further introducing l pairs of coefficients α_i, μ_i and then minimizing

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i\right) + C\sum_{i=1}^{l} \xi_i^p - \sum_{i=1}^{l} \mu_i \xi_i, \tag{2}$$

subject to $\alpha_i, \mu_i \geq 0$. Using the Karush-Kuhn-Tucker (KKT) optimality conditions [2], we obtain that

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i = 0, \tag{3}$$

that is, $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$. Hence the approximating function $f(\mathbf{x})$ can be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x} \cdot \mathbf{x}_i + b\,. \tag{4}$$

To improve the discriminative power of an SVM, the $\mathbf{x}_i$s are usually mapped to a high, possibly infinite-dimensional space (the *feature space*) via a non-linear mapping $\Phi(\mathbf{x})$; the core of the SVM becomes then the so-called *kernel function* $K(a, b) = \Phi(a) \cdot \Phi(b)$. The *kernel matrix* $\mathbf{K}$ is defined alongside such that $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ (here, and in the following, a boldface letter denotes the vector or the matrix whose components are denoted by the same letter, non-boldface; the term *kernel dimension* will refer, as is customary, to the dimension of the feature space). Widely used kernels are the *polynomial* one (finite-dimensional) and the *Gaussian*

one (infinite-dimensional). In the end, Equation (4) is rewritten as

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \; . \tag{5}$$

After minimization of L_P, some of the α_is (actually most of them in many practical applications) are equal to zero; those $\mathbf{x}_i$s for which this does not hold are called *support vectors*. The solution depends on them only and their number is proportional to the number of training samples [3].

The standard SVM algorithm is meant to be used batch-wise; to extend it to the on-line setting two different approaches have been proposed: (i) the batch algorithm is adapted to examine one sample at the time and produce a new approximate solution, such as in [6] and in [18, 20]; (ii) exact methods that incrementally update the solution as in [4]. In both cases we have that the potentially endless flow of training samples of the on-line setting will bring sooner or later to an explosion of the number of support vectors, and hence of the testing time.

2.2 On-line Independent SVMs

Consider Equation (5) again. The representation of $f(\mathbf{x})$ is *sparse*, meaning that the sum contains fewer members than l, this being a consequence of the α_i which are found to be zero. In practice, $f(\mathbf{x})$ is built by summing up as many factors as support vectors. Really, one step further can be taken: if some of the support vectors are linearly dependent on the others *in the feature space*, some of them can be expressed as a function of the others [8], therefore reducing the expression of $f(\mathbf{x})$. In these cases we can obtain different, possibly sparser, representation of the solution of the optimization problem (1), without changing the solution. A possibility is to simplify the solution after each update, which would be extremely

time consuming; this is why the method outlined in [8], and similar ones, cannot be applied on-line.

The main idea is then to keep a subset of the training vectors (we call them *basis* vectors) to build the classification function (5), independent from the samples used to find out the α_is during minimization. In fact, using the Representer Theorem [22], the solution of the optimization problem (1) can always be written as $\mathbf{w} = \sum_{i=1}^{l} \beta_i \Phi(\mathbf{x}_i)$ for a set of generic coefficients $\beta_i \in \mathbb{R}$. Hence directly plugging-in this expression of $\mathbf{w}$ in (1) will not change the optimum. Remembering the kernel trick and the definition of the matrix $\mathbf{K}$, the expression of $\|\mathbf{w}\|^2$ becomes $\sum_{i,j=1}^{l} \beta_i \beta_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = \sum_{i,j=1}^{l} \beta_i \beta_j K_{ij}$. Hence, we obtain the following optimization problem

$$\begin{aligned} &\arg\min_{\boldsymbol{\beta},b} \frac{1}{2} \sum_{i,j=1}^{l} \beta_i \beta_j K_{ij} + C \sum_{i=1}^{l} \xi_i^p \\ &\text{subject to } y_i \left(\sum_{j=1}^{l} \beta_i K_{ij} + b \right) \geq 1 - \xi_i, \ \forall i = 1, \cdots, l \end{aligned} \tag{6}$$

Due to his derivation, this formulation is completely equivalent to (1), and we can state the following theorem that links the solutions of the two problems.

Theorem 1 *Let $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$, l training samples, $\boldsymbol{\alpha}$ the solution of an SVM optimization problem (2), and $\boldsymbol{\beta}$ the solution of (6). Denote by $\mathbf{v}$ a vector in the null space of $\mathbf{K}$, we have that $\beta_i = \alpha_i y_i + v_i$.*

Proof: We calculate the Lagrangian of (6), introducing l pairs of coefficients α_i, μ_i

$$L'_P = \sum_{i,j}^{l} \left(\frac{1}{2} \beta_i - \alpha_i y_i \right) \beta_j K_{ij} - \sum_{i=1}^{l} \alpha_i (b y_i - 1 + \xi_i) + \sum_{i=1}^{l} C \xi_i^p - \sum_{i=1}^{l} \mu_i \xi_i \ .$$

Now, enforcing the KKT conditions on this, one obtains that

$$\frac{\partial L'_P}{\partial \beta_i} = \sum_{j=1}^{l} (\beta_j - \alpha_j y_j) K_{ij} = 0, \ i = 1, \cdots, l \ . \tag{7}$$

This is the product of the matrix $\mathbf{K}$ for the vector whose components are $\beta_j - \alpha_j y_j$. Clearly, in order for (7) to hold, the vector must be in the null space of $\mathbf{K}$. $\square$

If $\mathbf{K}$ has full rank, the null space only consists of the null vector, and therefore $\beta_i = \alpha_i y_i$, recovering a result already appeared in [15]. However if $\mathbf{K}$ is not full rank, there are infinite equivalent solutions to the SVM problem, and the β_is are not constrained at all: this agrees with Downs et al.'s [8] after-training simplification method and generalizes it.

The above considerations suggest to optimize problem (6), explicitly selecting as basis vectors only independent vectors in the feature space. Hence, instead of training and then simplifying the solution, we propose to *directly* build the solution with a small number of basis vectors. The algorithm can then be summed up as follows:

- check whether the current sample is linearly independent from the basis in the feature space; if it is, add it to basis.
- incrementally optimize (6)

These are the core steps of our algorithm that we call On-line Independent Support Vector Machine (OISVM). In the two following sections we describe in more details these two steps.

2.2.1 Exploiting linear independence on-line

Denote the indices of the vectors in the current basis, after l training examples, by $\mathcal{B}$. When the algorithm receives $\mathbf{x}_{l+1}$ it has to check if it is linearly independent or not from the basis vectors. In general, checking whether a matrix has full rank is

done via some decomposition, or by looking at the eigenvalues of the matrix; but here we want to check whether a *single* vector is linearly independent from a matrix of vectors already known to be full-rank. It is then simpler to exploit the definition of linear independence and check how well the vector can be approximated by a linear combination of the vectors in the matrix [17]. Let $d_j \in \mathbb{R}$; then let

$$\Delta = \min_{\mathbf{d}} \left\| \sum_{j \in \mathcal{B}} d_j \phi(\mathbf{x}_j) - \phi(\mathbf{x}_{l+1}) \right\|^2 . \tag{8}$$

If $\Delta > 0$ then $\mathbf{x}_{l+1}$ is linearly independent with respect to the basis, and $l+1$ is added to $\mathcal{B}$. In practice, one must check whether $\Delta \leq \eta$ where $\eta > 0$ is a tolerance factor, and expect that larger values of η lead to worse accuracy, but also to smaller bases. If η is set to zero the solution found will be *the same* as in the classical SVM formulation; therefore, no approximation whatsoever is involved, unless one gives it up in order to obtain even fewer support vectors (see Section 3 for a deeper discussion on this point). From (8) it is clear that the maximum meaningful value of η is $\max_i \|\phi(\mathbf{x}_i)\|^2 = \max_i K(\mathbf{x}_i, \mathbf{x}_i)$.

An efficient way to evaluate Δ is needed. Expanding (8) and remembering the definition of the kernel matrix $\mathbf{K}$, we get

$$\Delta = \min_{\mathbf{d}} \left(\mathbf{d}^T \mathbf{K}_{\mathcal{B}\mathcal{B}} \mathbf{d} - 2\mathbf{d}^T \mathbf{k} + K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) \right) , \tag{9}$$

where $k_i = K(\mathbf{x}_i, \mathbf{x}_{l+1})$ with $i \in \mathcal{B}$, and $\mathbf{K}_{\mathcal{B}\mathcal{B}}$ is the restriction of the kernel matrix $\mathbf{K}$ to the rows and columns corresponding to the indices in $\mathcal{B}$. Applying the extremum conditions with respect to $\mathbf{d}$ to Equation (9) we obtain that $\tilde{\mathbf{d}} = \mathbf{K}_{\mathcal{B}\mathcal{B}}^{-1} \mathbf{k}$ and, by replacing this in (9) once,

$$\Delta = K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) - \mathbf{k}^T \tilde{\mathbf{d}} . \tag{10}$$

Note that $\mathbf{K}_{\mathcal{B}\mathcal{B}}$ can be safely inverted since, by incremental construction, it is full-

rank. An efficient way to do it, exploiting the incremental nature of the approach, is that of updating it recursively. Using the matrix inversion lemma, after the addition of a new sample the new $\mathbf{K}_{\mathcal{BB}}^{-1}$ becomes

$$\begin{bmatrix} & & 0 \\ & \mathbf{K}_{\mathcal{BB}}^{-1} & \vdots \\ & & 0 \\ 0 & \cdots & 0\ 0 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} \tilde{\mathbf{d}} \\ \\ -1 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{d}}^T & -1 \end{bmatrix}, \tag{11}$$

where $\tilde{\mathbf{d}}$ and Δ are already evaluated during the test. Similar updates have been used in [4, 21]. Thanks to this incremental evaluation, the time complexity of the linear independence check is $O(|\mathcal{B}|^2)$, as one can easily see from the expression of $\tilde{\mathbf{d}}$ above.

2.2.2 Incremental Training

We want to find the solution to the optimization problem (6), without introducing the Lagrangian and its dual formulation. In fact the dual problem would introduce again l coefficients. Instead we want to use just a number of coefficients equal to the number of basis vectors selected. Hence we need a method to optimize the primal formulation of (6). The method that we have chosen is an adaptation of the modified Newton method found in [15, 23]. To apply the method we first have to set $p = 2$ in and transform it to an equivalent unconstrained minimization problem, using the $max(\cdot, \cdot)$ function. Let $\mathcal{D} \subset \{1, \ldots, l\}$; then the unconstrained problem is

$$\arg\min_{\boldsymbol{\beta}} \frac{1}{2}\boldsymbol{\beta}^T \mathbf{K}_{\mathcal{DD}} \boldsymbol{\beta} + \frac{1}{2} C \sum_{i=1}^{l} max\left(0, 1 - y_i \mathbf{K}_{i\mathcal{D}} \boldsymbol{\beta}\right)^2 . \tag{12}$$

We then set $\mathcal{D} = \mathcal{B}$, which assures that the solution to the problem is unique, since $\mathbf{K}_{\mathcal{B}\mathcal{B}}$ is full rank by construction. When a new sample $\mathbf{x}_{l+1}$ is available, the Newton method goes as follows:

(1) use the current value of $\boldsymbol{\beta}$ as starting vector;

(2) let $o_{l+1} = K_{l+1,\mathcal{B}}\boldsymbol{\beta}$; if $1 - y_{l+1}o_{l+1} \geq 0$, then stop: the current solution is already optimal. Otherwise,

(3) let $\mathcal{I} = \{i : 1 - y_i o_i > 0\}$ where $o_i = K_{i,\mathcal{B}}\boldsymbol{\beta}$ is the output of the i-th training sample;

(4) update $\boldsymbol{\beta}$ with a Newton step: $\boldsymbol{\beta} - \gamma \mathbf{P}^{-1}\mathbf{g} \rightarrow \boldsymbol{\beta}$ where $\mathbf{P} = \mathbf{K}_{\mathcal{B}\mathcal{B}} + C\mathbf{K}_{\mathcal{B}\mathcal{I}}\mathbf{K}_{\mathcal{B}\mathcal{I}}^T$ and $\mathbf{g} = \mathbf{K}_{\mathcal{B}\mathcal{B}}\boldsymbol{\beta} - C\mathbf{K}_{\mathcal{B}\mathcal{I}}\left(\mathbf{y}_{\mathcal{I}} - \mathbf{o}_{\mathcal{I}}\right)$;

(5) let $\mathcal{I}^{new} = \{i : 1 - y_i o_i > 0\}$ where o_i are recalculated using new $\boldsymbol{\beta}$. If $\mathcal{I}^{new}$ is equal to $\mathcal{I}$ stop; otherwise $\mathcal{I}^{new} \rightarrow \mathcal{I}$ and go to step 4.

In Step 4 above, we have set γ to one, without experiencing any convergence problem. With this choice the update of $\boldsymbol{\beta}$ is $C\mathbf{P}^{-1}\mathbf{K}_{\mathcal{B}\mathcal{I}}\mathbf{y}_{\mathcal{I}} \rightarrow \boldsymbol{\beta}^{new}$. In order to speed up the algorithm, we maintain an updated Cholesky decomposition of $\mathbf{P}$ and a vector with the product $\mathbf{K}_{\mathcal{B}\mathcal{I}}\mathbf{y}_{\mathcal{I}}$: every time a sample enters or exits from the set $\mathcal{I}$ these two quantities are updated. It turns out that the algorithm converges in very few iterations, usually 1 or 2.

The pseudo-code of the entire OISVM algorithm is summarized in Algorithm 1.

2.2.3 Analysis

The sparsification method used has several properties. We will examine them in this section. First of all, note that if the feature space has finite dimension n, then no more than n linearly independent vectors can be found, and $\mathcal{B}$ will never contain

Algorithm 1 Pseudo-code of OISVM.

Parameters: η

Initialization: $\mathcal{B} = \{\}, \beta = []$

for each time step $t = 1, \ldots, l$ **do**

$\mathbf{k} = K(\mathbf{x}_t, \mathbf{x}_j), j \in \mathcal{B}$

$\tilde{\mathbf{d}} = \mathbf{K}_{\mathcal{B}\mathcal{B}}^{-1}\mathbf{k}$

$\Delta = K(\mathbf{x}_t, \mathbf{x}_t) - \mathbf{k}^T\tilde{\mathbf{d}}$

if $\Delta \geq \eta$ **then** {Linear independence test}

$\mathcal{B} = [\mathcal{B}, t]$

end if

$o_t = \mathbf{k}^T\beta$

if $o_t < 1$ **then**

$\mathcal{I}^{new} = \{i : 1 - y_i o_i > 0\}$

repeat {Incremental update of the solution}

$\mathcal{I} = \mathcal{I}^{new}$

$\mathbf{P} = \mathbf{K}_{\mathcal{B}\mathcal{B}} + C\mathbf{K}_{\mathcal{B}\mathcal{I}}\mathbf{K}_{\mathcal{B}\mathcal{I}}^T$

$\boldsymbol{\beta} = C\mathbf{P}^{-1}\mathbf{K}_{\mathcal{B}\mathcal{I}}\mathbf{y}_{\mathcal{I}}$

Recalculate $o_i, i = 1, \cdots, t$

$\mathcal{I}^{new} = \{i : 1 - y_i o_i > 0\}$

until $\mathcal{I}^{new} = \mathcal{I}$

end if

end for

more than n vectors. However even if the feature space is infinite-dimensional, for any η greater than zero the maximum number of basis vectors will be finite for any training sequence (this argument has already been proved in [17]). Therefore, this method breaks the linear dependency between the number of SVs and the number of training samples mentioned in [3]. Note that any other approach to extend SVM

to the on-line framework, as the ones analyzed in [5], does not have such property. The boundedness of the solution can be easily proved for the specific case of the Gaussian kernel, showing also an interesting property on the distribution of the basis vectors in the input space.

Theorem 2 *Using OISVM with a Gaussian kernel,* $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$*, imposes a minimum distance in the input space among the basis vectors of* $\frac{1}{2\gamma}\log(\frac{1}{1-\eta})$

Proof: Consider plugging $\mathcal{B} = \{i\}$ in (8), that is, $\mathbf{x}_i$ is the only vector in the basis. Then

$$\Delta_i = \min_{d_i} \|d_i\phi(\mathbf{x}_i) - \phi(\mathbf{x}_{l+1})\|^2 \ . \tag{13}$$

Obviously $\Delta_i \geq \Delta, \forall i \in \mathcal{B}$, so if $\Delta_i \leq \eta$ then we have that $\Delta \leq \eta$ and the sample $l+1$ will not be added to the basis set. Remembering (9)-(10), the last equation can be expanded to

$$\Delta_i = K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) - \frac{K(\mathbf{x}_{l+1}, \mathbf{x}_i)^2}{K(\mathbf{x}_i, \mathbf{x}_i)} \ . \tag{14}$$

Remembering the definition of Gaussian kernel and the property that $K(\mathbf{x}, \mathbf{x}) = 1$ for any $\mathbf{x}$ we obtain that

$$\Delta_i \leq \eta \Leftrightarrow \|\mathbf{x}_{l+1} - \mathbf{x}_i\|^2 \leq -\frac{1}{2\gamma}\log(1-\eta) \ .$$

□

As a consequence of the above theorem we have that the number of basis vectors is finite for any compact input domain, when using a Gaussian kernel.

The time complexity of the re-training step is $O(|\mathcal{B}|l)$, as well as its space complexity. So setting η greater than zero, we have that the time complexity for training l training points is $O(l^2)$, since, as said above, after a certain number of samples $\mathcal{B}$

necessarily stops growing. Hence, keeping $\mathcal{B}$ small speeds up the training time as well as the testing time.

We can gather more insight on the role of η observing that OISVM approximates the kernel matrix $\mathbf{K}$ by another matrix $\widehat{\mathbf{K}}$ [13] and the quality of the approximation depends on η. In fact it is easy to show that $trace(\mathbf{K} - \widehat{\mathbf{K}}) \leq \eta|\mathcal{B}| \leq \eta l$, where l is the number of samples acquired [17]. If we consider a normalized kernel, that is a kernel for which $K(x, x)$ is always equal to 1, we can write $trace(\mathbf{K} - \widehat{\mathbf{K}})/trace(\mathbf{K}) \leq \eta$. On the other hand, a bigger η means of course a smaller number of SVs, hence it controls the trade-off between accuracy and speed of OISVM.

As a last remark, OISVMs can be easily extended to multiclass classification. Noting that the sparsification procedure makes no use of the labels y_i, in the one-vs-all multiclass setting the kernel matrix $\mathbf{K}_{\mathcal{BB}}$ is the same for each machine involved. All experiments shown in the next Section on multiclass databases are performed using this methodology.

3 Experimental Results

In this Section we report the experimental evaluation of OISVMs. We first test the method on a set of databases commonly used in the machine learning community (Section 3.1); we then apply it to two more realistic scenarios: the first is about place recognition, where the aim is to update the model to handle variations in an indoor environment (Section 3.2). In the second we show how our method classifies different types of human grasps, incrementally updating the model with the information coming from the observation of different subjects (Section 3.3).

We have implemented OISVM in Matlab, in the DOGMA library [24], and tested it against incremental and standard batch SVM implementations. In particular we have considered the incremental SVM developed in [4,25], that we will denote with IncrSVM [1], and the approximate on-line approach of [7], LASVM [2]. Note that IncrSVM and LASVM use $p = 1$ in (1), that is the norm-1 of the slack variables, that is known to be sparser of the formulation with $p = 2$, norm-2, used in OISVM. Hence, to have a baseline of the exact norm-2 formulation, we have also used the batch implementation of LIBSVM v2.82 [26], modified as suggested by the Authors in order to set $p = 2$ in (1); this modified version is called LIBSVM2 in the following.

3.1 Standard Benchmarks

Fig. 1 shows a comparison on the support vector growth on two standard benchmark databases [3]. For each benchmark, data are obtained by running 10 random $75\%/25\%$ train/test runs. Consider Fig. 1, left panel, Diabetes dataset. When all samples have been loaded, LIBSVM2 has about 427 SVs, and IncrSVM about 290. The kernel used is a homogeneous polynomial with degree 3 and the benchmark has 8 features, therefore the dimension of the feature space is $\binom{10}{3} = 120$ [2]; as expected, OISVM stops acquiring new SVs when there are exactly 120, although it loads a few more than the other approaches before reaching the limit. The accuracy (not displayed) is exactly the same of LIBSVM2. On the other hand, the size of the solutions of IncrSVM and LIBSVM2 will grow beyond the strict necessary 120

[1] Matlab code available at `http://www.cpdiehl.org/`

[2] C code available at `http://leon.bottou.org/`

[3] Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`

vectors, as theoretically proved in [3].

Consider now Fig. 1, right panel, Adult7 dataset. The kernel used is Gaussian and its dimension is infinite. The benchmark is large and complex (16100 samples, 123 features); nevertheless, with $\eta = 0.1$, at the end OISVM has about 2% of the SVs used by LIBSVM2 and less than 4% with respect to IncrSVM. The accuracy is essentially the same as that of LIBSVM2 (namely, an absolute loss of accuracy of $0.069\% \pm 0.068$). In Fig. 2 there is a graph showing the accuracy as a function of the number of basis vectors. It is clear that OISVM reaches better performance with less basis vectors, with respect to IncrSVM and LIBSVM2.

Lastly, consider Table 1, which shows the very same results in compact form for 10 more databases. In each column we show the mean recognition rate on 10 train/test splits taken from [27], and the number of support vectors in parenthesis. We have compared our method to the batch method LIBSVM2 and to IncrSVM. Cross validation was used to find the best parameters for each dataset, separately for the norm-1 and norm-2 formulation, while for OISVM we used the same parameters of the norm-2. OISVM attains a number of SVs which is from about 3 to slightly more than 60 times less than LIBSVM2, whereas the accuracy is not worse than 0.5%. Moreover it is always sparser than the norm-1 formulation.

For a complete comparison we have used also a variation of the Reduced Support Vector Machine (RSVM) method [14], the same used also in [15], in which for each run we have randomly selected a number of basis vectors equal to the one selected by OISVM. In this way we compare the two methods with the same number of basis vectors. We have used a Wilcoxon signed-ranks test to compare the performances of the two methods, as suggested in [28], obtaining a p-value less than 0.02. This confirms that our strategy to select basis vectors is better than the random sampling,

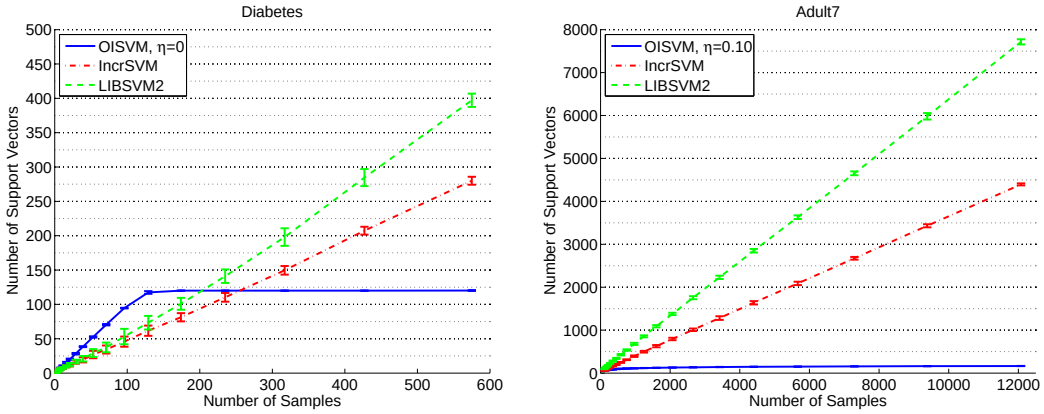

Fig. 1. Comparison of LIBSVM2, IncrSVM and OISVM and on the *Diabetes* (left panel) and *Adult7* (right panel) benchmarks. *Diabetes* is solved using a polynomial kernel with degree 3, while *Adult7* is solved using a Gaussian kernel.

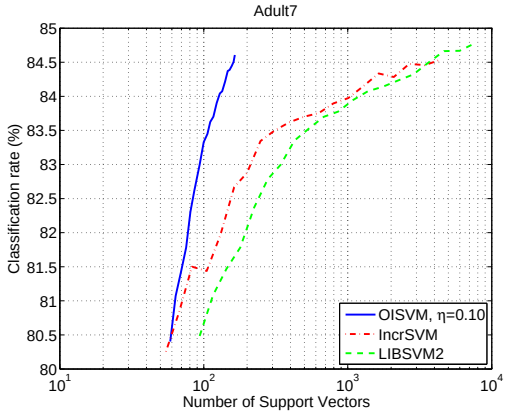

Fig. 2. Comparison of LIBSVM2, IncrSVM and OISVM on the *Adult7* dataset.

suggested by many authors, e.g. [29]. As a plus our strategy assures that, as said before, the growth of basis vectors will always eventually stop.

To gain more understanding on the difference between RSVM and OISVM we have also compared the value of the objective function (1) at the end of each dataset. The results are in Table 2, where the relative difference between the value of the objective function of OISVM and RSVM compared to LIBSVM2 are shown. There is a clear correlation between good classification accuracy and good value of the objective function. Hence our method seems to select the basis vectors in a way that

Table 1

Comparison of OISVM, RSVM, IncrSVM and LIBSVM2 on 10 standard benchmarks, solved using a Gaussian kernel. For each benchmark, we report the classification rate and the number of SVs. The value of η has been chosen in order not to loose more than 0.5% accuracy with respect to LIBSVM2.

Dataset	OISVM	RSVM	LIBSVM2	IncrSVM
Banana	$89.54 \pm 0.41\,(18.90 \pm 1.29)$	88.81 ± 0.63	$89.75 \pm 0.30\,(173.60 \pm 21.20)$	$89.77 \pm 0.28\,(121.10 \pm 9.01)$
Breast	$73.51 \pm 4.21\,(36.00 \pm 2.67)$	71.95 ± 4.79	$74.03 \pm 4.15\,(199.70 \pm 0.67)$	$74.55 \pm 4.16\,(122.60 \pm 4.95)$
Diabetis	$76.83 \pm 2.13\,(8.60 \pm 0.52)$	75.77 ± 2.85	$76.83 \pm 2.21\,(417.00 \pm 4.00)$	$76.83 \pm 1.77\,(283.30 \pm 8.17)$
Flare	$66.35 \pm 1.17\,(10.40 \pm 0.70)$	63.77 ± 4.05	$66.35 \pm 1.23\,(631.10 \pm 4.20)$	$67.15 \pm 1.63\,(555.70 \pm 8.59)$
German	$76.20 \pm 1.93\,(52.60 \pm 2.46)$	75.43 ± 2.20	$76.70 \pm 1.79\,(600.70 \pm 11.89)$	$76.37 \pm 2.60\,(384.70 \pm 7.01)$
Heart	$84.90 \pm 1.91\,(14.30 \pm 0.67)$	84.30 ± 2.95	$85.00 \pm 1.83\,(161.60 \pm 2.63)$	$85.00 \pm 2.87\,(85.00 \pm 3.27)$
Ringnorm	$98.03 \pm 0.27\,(87.60 \pm 6.01)$	98.60 ± 0.10	$98.57 \pm 0.10\,(377.00 \pm 4.06)$	$98.50 \pm 0.10\,(213.00 \pm 4.74)$
Titanic	$77.84 \pm 0.66\,(11.90 \pm 1.37)$	74.81 ± 3.24	$77.60 \pm 1.63\,(146.00 \pm 5.27)$	$77.28 \pm 0.35\,(86.80 \pm 6.91)$
Twonorm	$97.14 \pm 0.34\,(60.60 \pm 5.44)$	97.18 ± 0.32	$97.44 \pm 0.26\,(400.00 \pm 0.00)$	$97.65 \pm 0.09\,(299.90 \pm 5.74)$
Waveform	$89.67 \pm 0.47\,(78.20 \pm 3.12)$	89.66 ± 0.65	$90.10 \pm 0.36\,(324.60 \pm 9.43)$	$89.62 \pm 0.58\,(220.30 \pm 10.49)$

allows a better minimization of (1). This can be explained intuitively noting that adding a linearly dependent vector to the basis set will not help in minimizing the objective function. The negative result on *Ringnorm* is somehow surprising given the corresponding good classification performance, but it should not worry as the value of the objective function is loosely related to the classification performance. Hence the big increment in the objective function does not necessarily correspond to very bad classification performance, as it is shown in Table 1.

We have also conducted other experiments to compare the speed of the algorithms. Given that our implementation is in Matlab, so not optimized for speed, to have a fair comparison we have considered the number of kernel evaluations as a figure of merit. This is a standard measure for an implementation invariant comparison [4], see for example [7, 25]. We have used 3 databases, *Adult7*, *Banana* and *Waveform*,

[4] However note that our Matlab implementation of OISVM is about 1 order of magnitude faster than the Matlab implementation of IncrSVM.

with the same kernel and parameters used in [7]. In Fig. 3 we show the performance on the test set as a function of the number of kernel evaluations. For LIBSVM2 and IncrSVM we have just a point on the graph, while for OISVM we have a curve obtained varying η [5]. In other words, in OISVM we can trade accuracy for training (and testing) speed. From the figure we can see that OISVM achieves better or comparable performance to the other methods with much less kernel computations. Moreover the degree of freedom given by the sparsification method allows to trade just 0.5% of performance to save more than 1 order of magnitude of kernel computations. Note that the accuracy of OISVM is not always monotonically increasing with η going to 0. A possible explanation is that the sparsification procedure can be seen as a form of regularization; a similar view has been proposed in [30]. Hence it is plausible to suppose that there is an optimal value of η that gives better generalization properties.

3.2 Robot Navigation

We performed a second series of experiments, namely place recognition in an indoor environment, to evaluate our algorithm. We considered a realistic scenario where the algorithm had to incrementally update the model, so to adapt to the variations in an indoor environment due to human activities over long time spans. These variations include people appearing in different rooms during working time, objects such as cups moved or taken in/out of the drawers, pieces of furniture pushed around, and so forth.

Experiments were conducted on the IDOL2 database (Image Database for rObot

[5] The values of η for the 3 datasets are respectively $[0.4; 0.3; 0.2; 0.1; 0.05; 0.01]$, $[0.7; 0.6; 0.5; 0.4; 0.3; 0.2; 0.1; 0.01]$, $[0.8; 0.7; 0.6; 0.5; 0.4; 0.3]$.

Table 2

Loss in percentage of the objective function of OISVM and RSVM in relation to the one of LIBSVM2 on 10 standard benchmarks, solved using a Gaussian kernel. The values of η are the same of Table 1.

Dataset	OISVM	RSVM
Banana	13.9 ± 3.2	23.5 ± 10.5
Breast	2.1 ± 0.4	3.3 ± 0.4
Diabetis	1.7 ± 1.2	3.8 ± 4.7
Flare	0.1 ± 0	4.6 ± 6.3
German	9.7 ± 1.4	9.9 ± 0.5
Heart	0.9 ± 0.3	1.5 ± 1.0
Ringnorm	43.0 ± 4.6	3.6 ± 0.4
Titanic	0 ± 0	15.8 ± 10.8
Twonorm	5.6 ± 0.8	8.9 ± 1.0
Waveform	10.4 ± 0.6	10.4 ± 1.0

Localization 2, [31]), which contains 24 image sequences acquired using a perspective camera mounted on two mobile robot platforms. The acquisition was performed within an indoor laboratory environment consisting of five rooms of different functionality. The sequences were acquired under various weather and illumination conditions (sunny, cloudy, and night) and across a time span of six months. Thus, these data capture natural variability that occurs in real-world environments because of both natural changes in the illumination and human activity.

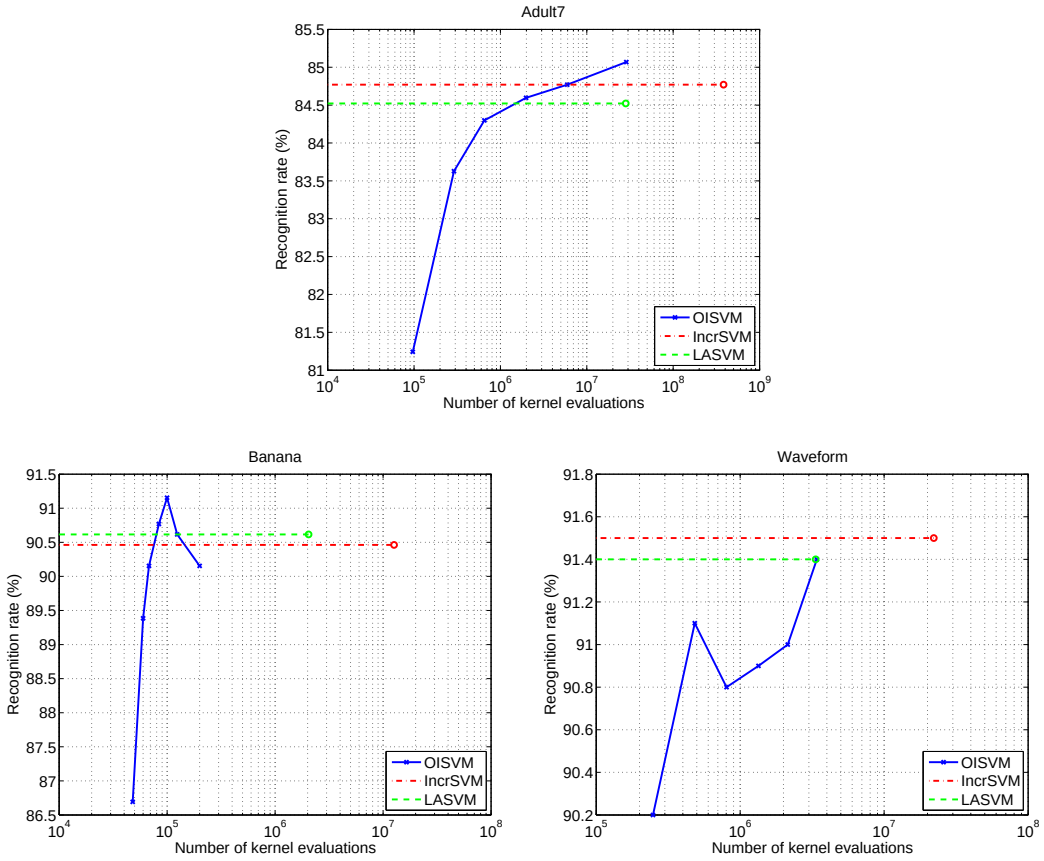

Fig. 3. Trade-off between accuracy and number of kernel evaluations of OISVM, IncrSVM and LASVM on 3 benchmarks. The OISVM curve is obtained plotting the number of kernel evaluations and the recognition rate at the end of the training, for each value of η. The number of kernel evaluations increases as η decreases. In LASVM and IncrSVM there is no parameter to trade accuracy for training speed, so only one point is shown.

Fig. 4 shows some sample images from the database, illustrating the difficulties of the task. The image sequences in the database are divided as follows: for each robot platform and for each type of illumination conditions, there were four sequences recorded. Of these four sequences, the first two were acquired six months before the last two. This means that, for each robot and for every illumination condition, there are always two sequences acquired under similar conditions, and two sequences acquired under very different conditions. This makes the database suitable for different kinds of evaluation on the adaptability of an incremental algo-

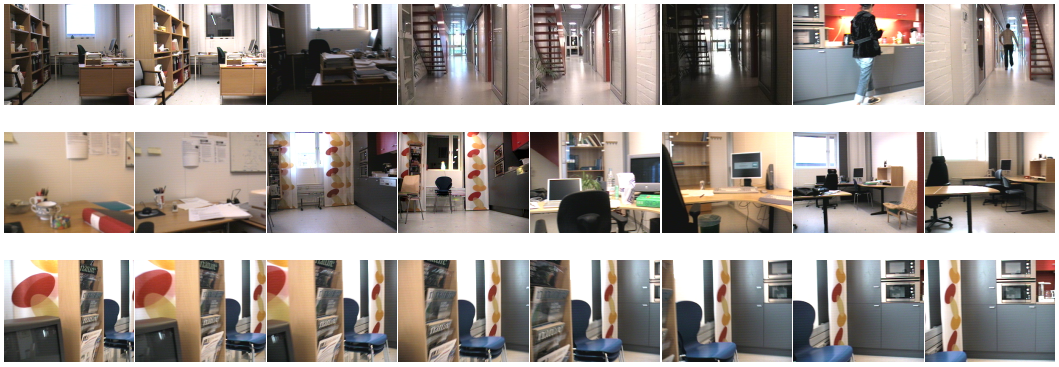

Fig. 4. Sample images illustrating the variations captured in the IDOL2 database. Images in the top row show the variability introduced by changes in illumination for two rooms (first six images) as well as people appearing in the environment. The middle row shows the influence of people's everyday activity (first four images) as well as larger variations which happened over a time span of 6 months. Finally, the bottom row illustrates the changes in viewpoint observed for a series of images acquired one after another in 1.6 seconds.

rithm. For further details about the database, we refer the readers to [31].

The evaluation was performed using composed receptive field histograms (CRFH) [32] as global image features and SIFT [33] for extracting local features. In the experiments, we consider both the *exponential* χ^2 kernel for SVM (when using CRFH), and the *matching kernel* [34] (when using SIFT). Note that the kernel in [34] is not always positive semidefinite [35], so this is also a test on non-Mercer kernels that have proved useful for visual recognition. The kernels used are infinite-dimensional, so for both kernels we run the OISVM using different values of η.

We benchmarked OISVM against the approximate incremental SVM extension of fixed-partition technique [6] and the IncrSVM algorithm. The algorithms were trained incrementally on three sequences from IDOL2, acquired under similar illumination conditions with the same robot platform; the fourth sequence was used for testing. In order to test the various properties of interest of the incremental algorithms, we need a reasonable number of incremental steps. Thus, every sequence

was split into 5 subsequences, so that each subset contained one of the five images acquired by the robot every second (image sequences were acquired at a rate of 5fps). Since during acquisition the camera's viewpoint continuously changes [36], the subsequences could be considered as recorded separately in a static environment but for varying pose. This setup allows us to examine how the algorithms perform on data with less variations. In order to get a feeling of the variations of the frame images in a sequence, the bottom row of Fig. 4 shows some sample images acquired within a time span of 1.6 sec. As a result, training on each sequence was performed in 5 steps, using one subsequence at a time, resulting in 15 steps in total. Overall, we considered 36 different permutations of training and test sequences for the exponential χ^2 kernel and for the matching kernel; here we report the average results, plus/minus one standard deviation. Fig. 5, left, shows the recognition rates of the exponential χ^2 kernel (top) and matching kernel (bottom) experiments obtained at each step using OISVM, IncrSVM and the approximate incremental algorithm. Fig. 5, right, reports the number of support vectors stored in the model at each step of the incremental procedure, for both kernel types.

We see that, performance-wise, all methods achieve statistically comparable results; this is true for both kernel types. As far as the machine size is concerned, the OISVM algorithm shows a considerable advantage with respect to the fixed-partition method and IncrSVM. In the case of the exponential χ^2 kernel this advantage is truly impressive (Fig 5, top right): for $\eta = 0.017$ and 0.025 the size at the final incremental step is $34\%/22\%$ of that of the fixed-partition method and $28\%/18\%$ of that of IncrSVM. Even more important, OISVM, for these two values of η, has found a plateau in memory, while for other methods the trend seems to be of a growth proportional to the number of training data. Note that the choice of the parameter η is crucial for achieving an optimal trade-off between compactness of

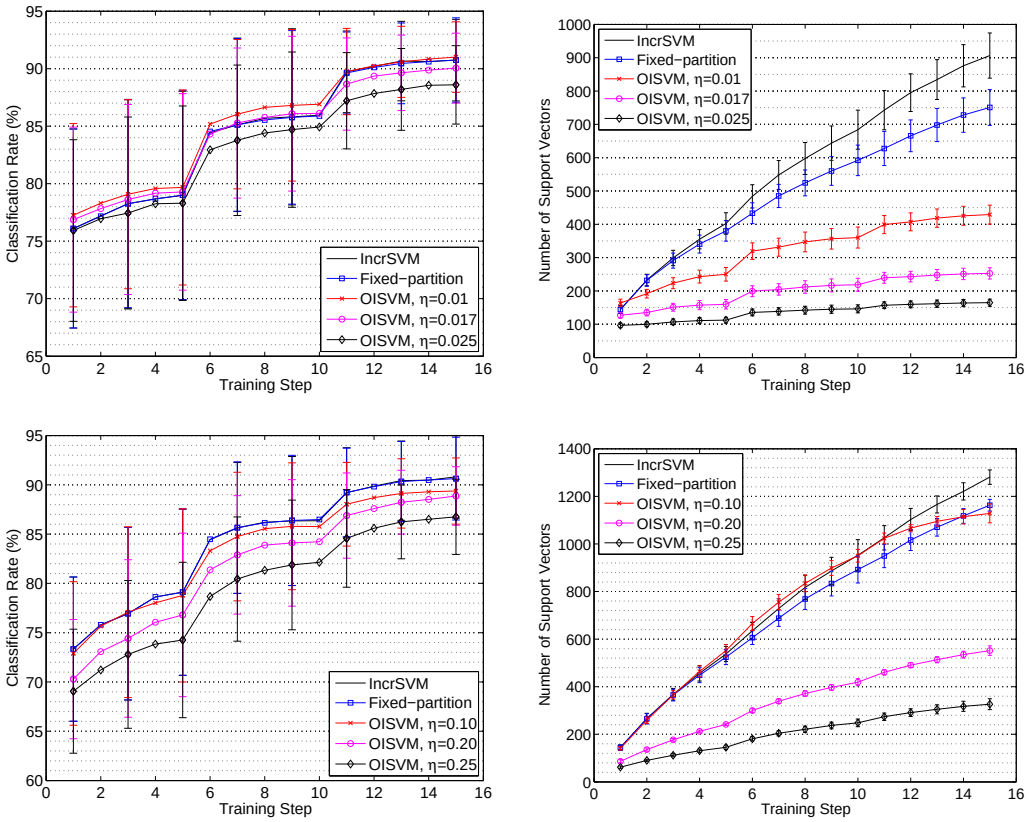

Fig. 5. Experimental results on the IDOL2 database, using OISVM with three different values of η, the fixed-partition and IncrSVM. Top: χ^2 kernel, Bottom: matching kernel.

the solution and optimal performance.

It is very interesting to note that, in the case of the matching kernel, the memory reduction for OISVM is less pronounced, and there is no clear plateau in memory growth by any of the algorithms. This behavior might be due to several factors: to begin with, the matching kernel is not a Mercer kernel [35]; moreover, in the induced space of the matching kernel, there seems to be a high probability that pairs of training points be (almost) orthogonal to each other (note that, as the kernel is not a Mercer one, the geometric interpretation might not be valid). Anyway, given enough training points, the machine will always reach a maximum size and will stop growing [17].

3.3 Grasp Classification

In the third experiment, human subjects are involved in a grasping task, which our system must identify; in particular, we are interested in classifying the grasping posture independently of the subject and the object being grasped. Eight subjects were involved in the experiment, 5 men and 3 women, all able-bodied, aged between 23 and 36. They were given no prior knowledge on the aim and scope of the experiment. Each subject would sit comfortably in front of a workspace large about one squared meter and wear a 22-sensors Immersion CyberGlove [37] on the right hand, and a Force Resistor Sensor (FSR) on the thumb. Figure 6 (upper row) shows the devices, as worn by a subject.

Fig. 6. (above) The devices and some of the objects used for the experiment, left to right: the CyberGlove, the Force Resistor Sensor attached to the subject's thumb, the beer can, the duct tape roll and the mug. (below) The 5 grasp types to be recognized, left to right: power large, power flat, tripodal precision, thumb/index precision and spherical precision.

A number of objects encountered in everyday life were presented to the subjects, who would then repeatedly grasp them in a particular way. The grasp types were selected among those identified by Cutkosky in [38]: power large grasp, power flat grasp, tripodal precision grip, thumb/index precision grip and spherical precision grip. Figure 6 shows three of the objects used in the experiment and five examples of grasp types. Note that each object may afford several grasp types, e.g., a rubber

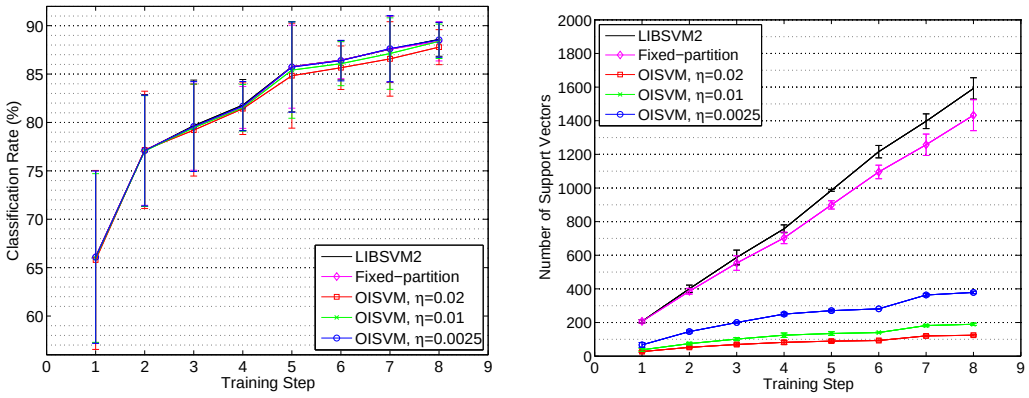

Fig. 7. Classification rate (left) and average number of support vectors (left) for the grasping classification experiment. OISVM uses a Gaussian kernel and three different values of η.

duck can be grasped either via a tripodal precision grip, a thumb/index precision grip and/or a spherical precision grip.

Each grasp is determined by the CyberGlove, therefore the input space is $\mathbb{R}^{22}$; and five categories, each corresponding to a grasp type, were set up. The optimal hyper-parameters C and σ were found via grid search and cross-validation, as is customary. We have compared OISVM with the batch method LIBSVM2 and the fixed-partition technique. As in the experiments on place recognition experiments, we have splitted the training data in several batches, to be able to use the fixed partition method. In particular, in each training step we feed to the systems all the data coming from an user. A Gaussian kernel and three different values of η were used for OISVM. The results are shown in Figure 7.

Consider the Figure 7, left panel: it is apparent that the classification rate is basically the same, uniformly and for all approaches tested. The right panel shows that, in agreement with the previous experiments, LIBSVM2 and the fixed partition method gather a number of SVs which grows proportionally with the training set. On the other hand, OISVM with various values of η uniformly show a dramatically smaller number of SVs, getting to as few as 124 SVs in the case for $\eta = 0.02$, losing only

0.8% of accuracy compared to LIBSVM2.

4 Conclusions

In this paper we have presented, implemented and tested a novel system for on-line learning based upon Support Vector Machines. The method, On-line Independent SVMs (OISVMs), avoids using in the solution those support vectors which are linearly dependent of previous ones in the feature space; linear independence is checked incrementally every time a new sample is made available to the system; the optimization problem is solved via an incremental algorithm which benefits of the small size of the basis set. A parameter called η is employed to trade better/worse accuracy for more/fewer support vectors in the basis.

We tested the method both on a standard set of benchmark databases and on two real-world case studies, namely: (a) place recognition in an indoor environment and (b) human grasping classification. Both real-world experiments have been carefully crafted in order to take into account changing conditions in the environment: robot images were acquired under different weather conditions and across a time span of several months; grasping was done across a time span of several days, with several different objects and by several human subjects with little or no prior knowledge about the experiment.

The experimental results, carried out for various values of η, show that OISVMs enjoy an excellent accuracy/basis size trade off. Moreover, a deeper analysis of the value of the objective function at the end of the training reveals that our method selects the basis vectors in such a way as to better minimize it.

An open issue is how to choose η to achieve an optimal trade-off between accu-

racy and speed for a given task. Both theory and experiments clearly point out the relevance of η for fully exploiting the power of OISVMs, but its choice is today one of the heuristics of the method, along with the choice of C, the kernel function and the kernel parameters. Therefore, the only solution we can point to the reader is the use of a validation set, as it is customary for the other previously mentioned SVM parameters. How to determine η in a more principled way will be the focus of future work.

Another research direction we plan to pursue is to extend the method so to handle an underlying dynamic distribution of the data, as opposed to the static scenario considered in this paper. This problem has been already tackled in generative frameworks (i.e. [39]) but it is still far from being solved in discriminative settings.

Lastly we plan to use OISVM in a life-long learning scenario. So far OISVMs are able to perform continuous learning on data collected on a span of time of up to several months, but it is still not possible to use them with a truly never-ending stream of data, since all training samples must anyway be retained. To achieve this goal we plan to extend the method so to include a forgetting mechanism.

Acknowledgments

Thanks to Joseph Keshet for helping improving the manuscript. This work was supported by EU projects RobotCub (IST-2004-004370), NEURObotics (FP6-IST-001917) and DIRAC (FP6-0027787).

References

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, D. Haussler, Ed. ACM press, 1992, pp. 144–152.

[2] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[3] I. Steinwart, "Sparseness of support vector machines," *Journal of Machine Learning Research*, vol. 4, pp. 1071–1105, 2003.

[4] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems*, 2000, pp. 409–415.

[5] C. Domeniconi and D. Gunopulos, "Incremental support vector machine construction," in *Proceedings of the 2001 IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 589–592.

[6] N. Syed, H. Liu, and K. Sung, "Incremental learning with support vector machines," in *Proceedings of International Joint Conference on Articial Intelligence*, 1999.

[7] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *Journal of Machine Learning Research*, vol. 6, pp. 1579–1619, 2005.

[8] T. Downs, K. E. Gates, and A. Masters, "Exact simplification of support vectors solutions," *Journal of Machine Learning Research*, vol. 2, pp. 293–297, 2001.

[9] D. Nguyen and T. Ho, "An efficient method for simplifying support vector machines," in *Proceedings of the 22nd international conference on Machine learning*. New York, NY, USA: ACM Press, 2005, pp. 617–624.

[10] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA, USA: MIT press, 2002.

[11] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *Journal of Machine Learning Research*, vol. 2, pp. 243–264, 2002.

[12] G. Baudat and F. Anouar, "Feature vector selection and projection using kernels," *Neurocomputing*, vol. 55, no. 1–2, pp. 21–38, 2003.

[13] F. R. Bach and M. I. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[14] Y. J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proceedings of the SIAM International Conference on Data Mining*, 2001.

[15] S. S. Keerthi, O. Chapelle, and D. DeCoste, "Building support vector machines with reduced classifier complexity," *Journal of Machine Learning Research*, vol. 8, pp. 1–22, 2006.

[16] M. Wu, B. Schölkopf, and G. Bakir, "A direct method for building sparse kernel learning algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 603–624, 2006.

[17] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, 2004.

[18] J. Kivinen, A. Smola, and R. Williamson, "Online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[19] J. Weston, A. Bordes, and L. Bottou, "Online (and offline) on an even tighter budget," in *Proceedings of AISTATS 2005*, R. G. Cowell and Z. Ghahramani, Eds. Society for Artificial Intelligence and Statistics, 2005, pp. 413–420.

[20] L. Cheng, S. V. N. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli, "Implicit online learning with kernels," in *Advances in Neural Information Processing Systems*

19, B. Schölkopf, J. Platt, and T. Hoffman, Eds. Cambridge, MA: MIT Press, 2007.

[21] L. Csató and M. Opper, "Sparse representation for gaussian process models," *Advances in Neural Information Processing Systems*, vol. 13, 2001.

[22] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proceedings of the 14th Annual Conference on Computational Learning Theory*. London, UK: Springer-Verlag, 2001, pp. 416–426.

[23] S. S. Keerthi and D. DeCoste, "A modified finite Newton method for fast solution of large scale linear SVMs," *Journal of Machine Learning Research*, vol. 6, pp. 341–361, 2005.

[24] F. Orabona, *DOGMA: a MATLAB toolbox for Online Learning*, 2009, software available at http://dogma.sourceforge.net.

[25] C. P. Diehl and G. Cauwenberghs, "SVM incremental learning, adaptation and optimization," in *In Proceedings of the 2003 International Joint Conference on Neural Networks*, 2003, pp. 2685–2690.

[26] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[27] G. Rätsch, "Benchmark repository," Intelligent Data Analysis Group, Fraunhofer-FIRST, Tech. Rep., 2005, available at http://ida.first.fraunhofer.de/~raetsch.

[28] J. Demar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, January 2006.

[29] R. Rifkin, G. Yeo, and T. Poggio, "Regularized least-squares classification," in *Advances in Learning Theory: Methods, Models and Applications*, ser. NATO Science Series III: Computer and Systems Sciences, J. A. K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, Eds. VIOS Press, 2003, pp. 131–154.

[30] G. Blanchard, P. Massart, R. Vert, and L. Zwald, "Kernel projection machine: a new tool for pattern recognition," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2005, pp. 1649–1656.

[31] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt, "The KTH-IDOL2 database," KTH, CAS/CVAP, Tech. Rep. 304, 2006, available at http://cogvis.nada.kth.se/IDOL2.

[32] O. Linde and T. Lindeberg, "Object recognition using composed receptive field histograms of higher dimensionality," in *Proceedings ICPR'04*, 2004.

[33] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision (ICCV)*, vol. 2. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–1157.

[34] C. Wallraven, B. Caputo, and A. Graf., "Recognition with local features: the kernel recipe," in *Proceedings of ICCV'03*, 2003.

[35] S. Boughorbel, J.-P. Tarel, and F. Fleuret, "Non-mercer kernels for SVM object recognition," in *Proceedings of British Machine Vision Conference*, London, England, 2004, pp. 137–146.

[36] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt, "Incremental learning for place recognition in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS07)*, San Diego, California, October 2007.

[37] *CyberGlove Reference Manual*, Virtual Technologies, Inc., 2175 Park Blvd., Palo Alto (CA), USA, August 1998.

[38] M. R. Cutkosky, "On grasp choice, grasp models and the design of hands for manufacturing tasks," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 3, pp. 269–279, 1989.

[39] J. Yoon, S. Roberts, M.Dyson, and J. Gan, "Sequential bayesian estimation for adaptive classification," in *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008.