Online Independent Support Vector Machines

Anonymous CVPR submission

Paper ID 631

Abstract

Support Vector Machines (SVMs) are a machine learning method widely employed in, e.g., visual recognition, medical diagnosis and robotic control. One of their most interesting characteristics is that the solution achieved is sparse: a few samples (support vectors) usually account for the complexity of the classification task. Both the training and testing time crucially depend on the number of support vectors, and it is then very important to keep it small, trying at the same time to retain the accuracy of the solution. This is especially evident in online settings such as topological mapping for a robotic platform, where a large amount of unknown data is expected to be available as the robot adapts to the changing environment.

In this paper we introduce a novel approach to the problem, called Online Independent SVMs, in which the solution is built online using only those support vectors which are linearly independent in the feature space. We then show that OISVMs achieve an excellent accuracy vs. compactness trade-off in general, while retaining the full accuracy of ordinary SVMs in case a finite-dimensional kernel is used. This statement is supported by experiments on standard benchmark databases as well as on a real-world application, namely place recognition in an indoor environment, from sequences acquired by robot platforms under different weather conditions and across a time span of several months.

1. Introduction

Introduced in the early 90s by Boser, Guyon and Vapnik [4], *Support Vector Machines* (SVMs) are a class of machine learning algorithms deeply rooted in Statistical Learning Theory [27], able to classify data taken from an unknown probability distribution, given a set of training examples. As opposed to analogous methods such as, e.g., artificial neural networks, they have the main advantages that (a) training is guaranteed to end up in a global minimum, (b) their generalization power is theoretically well founded, (c) they can easily work with highly dimensional, non-linear data, and (d) the solution achieved is sparse. Due to these good properties, they have been now extensively used in, e.g., speech recognition, object classification and function approximation [10]. On the other hand, one of their main drawbacks is their alleged inability to cope with large datasets [14].

Yet, in most real-life applications, datasets *are* large, for example when online learning must be performed. Online learning is a scenario in which training data is provided one example at a time, as opposed to the batch mode in which all examples are available at once (see [16] and citations therein). In the case of, e.g., non-stationary data, online algorithms will generally perform better since ambiguous information (i.e., whose distribution varies over time) is present, and couldn't possibly be taken into account by the batch algorithm. Online algorithms allow to incorporate additional training data, when it is available, without re-training from scratch.

In an online setting there is no guarantee that the flow of data will *ever* cease; therefore, applying SVMs here looks appealing but we need a way to cope with large datasets. One of the keys to the problem seems to lie in the sparseness of their solution. That an SVM solution is *sparse* means that usually just a few samples account for its complexity; in fact, SVMs can be seen as a way of compressing data by selecting "the most important" samples (*support vectors*, SV) among those in the training set. Keeping the number of SVs small without losing accuracy is therefore a major challenge. This is even more relevant since a recent result [25] shows that this number grows indefinitely with the number of training samples, and the testing time — a central issue in online learning, since one might want to test in real time — crucially depends on it.

In this paper we propose a method of incrementally selecting SVs based upon *linear independence in the feature space*: SVs which are linearly dependent on already stored ones are rejected, and a smart, incremental minimization algorithm is employed to find the new minimum of the cost function. The size of the kernel matrix (the core of an SVM and its major bottleneck) is therefore kept small. Our experiments indicate that SVMs employing this idea, that we call

Online Independent Support Vector Machines (OISVMs), do not grow linearly with the training set, as it was the case in [25], but reach a limit size and then stop growing [13]. In the case of finite-dimensional feature spaces they also *keep the full accuracy of standard SVMs*; whereas in the infinite-dimensional case, at the price of a negligible loss in accuracy, one can tune the growing rate of the machine.

To support this claim, we show an extensive set of experimental results obtained by comparing SVMs and OISVMs on standard benchmark databases as well as on a real-world, online application coming from robotic vision: place recognition in an indoor environment, from sequences acquired by robot platforms under different weather conditions and across a time span of several months. Our results show that, on standard benchmarks, the accuracy of OISVMs can be up to only 0.063% worse than SVMs, with less than 5% of the support vectors; whereas, on the real-world application, we get as few as one third of the SVs required by an approximated incremental method, while retaining essentially the same accuracy.

The paper is structured as follows: after a review of the relevant literature, Section 2 gives an overview of background mathematics proper to SVMs; in Section 3 OISVMs are described. Section 4 shows the experimental results and lastly, in Section 5, conclusions are drawn and future work is outlined.

1.1. Previous Work

A simplification of the decision function is proposed in [11], based upon linear independence of the SVs in the feature space, performed *after* the training is done. This is a simple consequence of the fact that, if the feature space has dimension n, at most $n+1$ independent SVs are required to build the solution [23]. The idea is useful in reducing the testing time, but it is unfeasible in an online setting, since the simplification must be performed every time a new sample is acquired. The same consideration applies, e.g., to the after-training simplification proposed in [22]. On the other hand, discarding from the sample set the linearly dependent SVs will result in an approximation; other methods to heuristically select a subset of the support vectors have been proposed, e.g., in [17, 29, 14]. Besides this, these methods require the knowledge of the full training set, and therefore are not suited for online learning.

In order to keep the solution compact without losing accuracy, the key is to keep the size of the kernel matrix small, i.e., reducing its rank. Unsupervised rank reduction methods have been proposed [3] as well as supervised ones [2], but no application of these ideas appears so far, to the best of our knowledge, in online settings. This is particularly important since it has been shown [25] that the SV set grows linearly with the sample set; therefore, in an online setting, the size of a SVM would grow indefinitely, and so would the testing time.

The exact solution to online SVM learning was given by Cauwenberghs and Poggio in 2000 [8], but their idea has received little attention in the community so far [16], probably due to the lack of a detailed analysis of its complexity.

As far as our main testbed is concerned, visual place recognition for robot platforms is a widely researched topic in which online learning is critical. Incremental learning approaches had been so far mostly used for constructing the geometrical map, or the environment representation, online. Brunskill et al. [6] proposed a model using incremental PCA for simultaneous localization and mapping (SLAM). A similar approach was used in the only work we are aware of that uses an incremental method in the context of place recognition. In [1] incremental PCA was used to update low-dimensional representations of images taken by a mobile robot as it moved around in an environment. They also tested repetitive learning of their model with the same training images several times. While they obtained impressive results in terms of reconstruction, their method was not tested for classification.

2. Background Mathematics

Due to space limitations, this is a very quick account of SVMs — the interested reader is referred to [7] for a tutorial, and to [10] for a comprehensive introduction to the subject. Assume $\{\mathbf{x}_i, y_i\}_{i=1}^l$, with $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$, is a set of samples and labels drawn from an unknown probability distribution; we want to find a function $f(\mathbf{x})$ such that $sign(f(\mathbf{x}))$ best determines the category of any future sample $\mathbf{x}$. In the most general setting,

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

where $b \in \mathbb{R}$ and $K(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1) \cdot \Phi(\mathbf{x}_2)$, the *kernel function*, evaluates inner products between samples through a non-linear mapping Φ. The α_is are Lagrangian coefficients obtained by solving (the dual Lagrangian form of) the problem

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C \textstyle\sum_{i=1}^{l} \xi_i^p \\ \text{subject to} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{2}$$

where $\mathbf{w}$ defines a *separating hyperplane* in the feature space, i.e., the space where Φ lives, whereas $\xi_i \in \mathbb{R}$ are slack variables, $C \in \mathbb{R}^+$ is an error penalty coefficient and p is usually 1 or 2.

Thanks to the introduction of $K(\cdot, \cdot)$ and the associated *kernel matrix* K, with $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, SVMs are able to find $f(\mathbf{x})$, complex as it may be, by solving a linear separation problem in a highly-dimensional space. This idea is often called *kernel trick*. In practice, most of the α_i are found to be zero after training and therefore can be neglected in evaluating the separating hyperplane in formula (1) (those $\mathbf{x}_i$s which cannot be neglected are called *support vectors*, SV).

In the following, the term *kernel dimension* will refer, as is customary, to the dimension of the feature space. The kernel dimension is related to the generalization power of the machine, and it depends on the choice of the kernel itself. Widely used kernels include the *polynomial* one (finite-dimensional) and the *Gaussian* one (infinite-dimensional).

3. Online Independent Support Vector Classification

The possibility to obtain a more compact representation of $f(\mathbf{x})$ follows from the fact that the solution to a SVM problem (that is, the α_is) is not unique if K does not have full rank [7], which is equivalent to some of the SVs being linearly dependent on some others *in the feature space* (this is the core of Downs et al.'s [11] original idea).

To avoid simplify the solution each time a new sample is acquired like in [11], we need a way to use independent SVs only. Hence, the main idea is to decouple the concept of "basis" vectors, used to build the classification function (1), from the samples used to find out the α_is. If the selected basis vectors span the same subspace as the whole sample set, the solution found will be equivalent — that is, we will not lose any precision.

We hereby propose, after having received a new training sample, to incrementally add it to the basis if it is linearly independent in the feature space from those already present in the basis itself. The solution found is *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 4 for a deeper discussion on this point).

Denoting the indexes of the vectors in the current basis, after l training samples, by $\mathcal{B}$, and the new sample under judgment by $\mathbf{x}_{l+1}$, the algorithm can then be summed up as follows:

1. check whether $\mathbf{x}_{l+1}$ is linearly independent from the basis in the feature space; if it is, add it to $\mathcal{B}$; otherwise, leave $\mathcal{B}$ unchanged.

2. incrementally re-train the machine.

In the following, the notations A_{IJ} and $\mathbf{v}_I$, where A is a matrix, $\mathbf{v}$ is a vector and $I, J \subset \mathbb{N}$ denote in turn the sub-matrix and the sub-vector obtained from A and $\mathbf{v}$ by taking the indexes in I and J.

3.1. Linear independence

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 which is already known to be full-rank. Inspired by the definition of linear independence, we check how well the vector can be approximated by a linear combination of the vectors in the set [12]. Let $d_j \in \mathbb{R}$; then let

$$\Delta = \min_{\mathbf{d}} \left\| \sum_{j \in \mathcal{B}} d_j \phi(\mathbf{x}_j) - \phi(\mathbf{x}_{l+1}) \right\|^2 \qquad (3)$$

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, we 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. As a matter of fact, if η is set at machine precision, OISVMs retain the exact accuracy of SVMs. Notice also 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 more than n vectors.

Expanding equation (3) we get

$$\begin{aligned}\Delta = \min_{\mathbf{d}} (& \sum_{i,j \in \mathcal{B}} d_j d_i \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_i) \\ & -2 \sum_{j \in \mathcal{B}} d_j \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_{l+1}) \\ & +\phi(\mathbf{x}_{l+1}) \cdot \phi(\mathbf{x}_{l+1}) \ )\end{aligned} \qquad (4)$$

that is, applying the kernel trick,

$$\Delta = \min_{\mathbf{d}} \left(\mathbf{d}^T K_{\mathcal{B}\mathcal{B}} \mathbf{d} - 2\mathbf{d}^T \mathbf{k} + K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) \right) \qquad (5)$$

where $k_i = K(\mathbf{x}_i, \mathbf{x}_{l+1})$ with $i \in \mathcal{B}$. Solving (5), that is, applying the extremum conditions with respect to $\mathbf{d}$, we obtain

$$\tilde{\mathbf{d}} = K_{\mathcal{B}\mathcal{B}}^{-1} \mathbf{k} \qquad (6)$$

and, by replacing (6) in (5) once,

$$\Delta = K(\mathbf{x}_{l+1}, \mathbf{x}_{l+1}) - \mathbf{k}^T \tilde{\mathbf{d}} \qquad (7)$$

Note that $K_{\mathcal{B}\mathcal{B}}$ 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: after the addition of a new sample, the new $K_{\mathcal{B}\mathcal{B}}^{-1}$ then becomes

$$\begin{bmatrix} & & & 0 \\ & K_{\mathcal{BB}}^{-1} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} \tilde{\mathbf{d}} \\ -1 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{d}}^T & -1 \end{bmatrix} \quad (8)$$

where $\tilde{\mathbf{d}}$ and Δ are already evaluated during the test (this method matches the one used in Cauwenberghs and Poggio's incremental algorithm [8]). Thanks to this incremental evaluation, the time complexity of the linear independence check is $O(|\mathcal{B}|^2)$, as one can easily see from Equation (6).

3.2. Training the machine

The training method largely follows Keerthi et al. [15, 14], that we have adapted for online training. The algorithm directly minimizes problem (2) as opposed to the standard way of minimizing its dual Lagrangian form, allowing to select explicitly the basis vectors to use. We set $p = 2$ in (2) and transform it to an unconstrained problem. Let $\mathcal{D} \subset \{1, \ldots, l\}$; then the unconstrained problem is

$$\min_{\boldsymbol{\beta}} \left(\frac{1}{2} \boldsymbol{\beta}^T K_{\mathcal{DD}} \boldsymbol{\beta} + \frac{1}{2} C \sum_{i=1}^{l} max\left(0, 1 - y_i K_{i\mathcal{D}} \boldsymbol{\beta}\right)^2 \right) \quad (9)$$

where $\boldsymbol{\beta}$ is the vector of the Lagrangian coefficients involved in $f(\mathbf{x})$, analogously to the α_is in the original formulation. If we set $\mathcal{D} = \mathcal{B}$, then the solution to the problem is unique since $K_{\mathcal{BB}}$ is full rank by construction. Newton's method as modified by Keerthi et al. [15, 14] can then be used to solve (9) after each new sample. When the new sample $\mathbf{x}_{l+1}$ is received the method goes as follows:

1. let $\mathcal{I} = \{i : 1 - y_i o_i < 0\}$ where $o_i = K_{i\mathcal{B}} \boldsymbol{\beta}$ and $\boldsymbol{\beta}$ is the vector of optimal coefficients with l training samples; if $\mathcal{I}$ hasn't changed, stop.

2. otherwise, let the new $\boldsymbol{\beta}$ be $\boldsymbol{\beta} - \gamma \mathbf{P}^{-1} \mathbf{g}$, where $\mathbf{P} = K_{\mathcal{BB}} + C K_{\mathcal{BI}} K_{\mathcal{BI}}^T$ and $\mathbf{g} = K_{\mathcal{BB}} \boldsymbol{\beta} - C K_{\mathcal{BI}} (\mathbf{y}_{\mathcal{I}} - \mathbf{o}_{\mathcal{I}})$.

3. go back to Step 1.

In Step 2 above, γ is set to one. In order to speed up the algorithm, we maintain an updated Cholesky decomposition of $\mathbf{P}$. It turns out that the algorithm converges in very few iterations, usually 0 to 2; the time complexity of the re-training step is $O(|\mathcal{B}|l)$, as well as its space complexity; hence, keeping $\mathcal{B}$ small will speed up the training time as well as the testing time.

4. 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 4.1); we then apply it to the more realistic scenario of place recognition, where the aim is to update the model to handle variations in an indoor environment (section 4.2).

OISVMs have been implemented in Matlab and tested against LIBSVM v2.82 [9]. For the sake of comparison, LIBSVM has been modified as suggested by the Authors in order to set $p = 2$ in (2); this modified version is called LIBSVM-2 in the following. The software library was also extended to various families of kernels, and to two approximate incremental extensions of SVM, the fixed-partition incremental SVM [26], and the memory-controlled incremental SVM [24]. In the case of finite-dimensional kernels, we only show the performance of LIBSVM-2 against OISVMs with η at machine precision, since the solution found by OISVM is exactly equivalent; in the case of infinite-dimensional kernels, we show curves for various values of η.

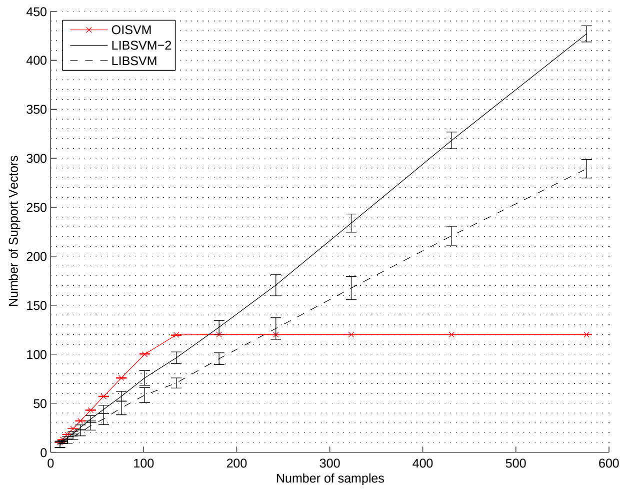
4.1. Experiments with Standard Benchmark Databases

Figure 1 and Table 4.1 show the above mentioned comparison on some standard benchmark databases, available at http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets. For each benchmark, data are obtained by running 10 random 75%/25% train/test runs.

Consider Figure 1, left pane: when all samples have been loaded, LIBSVM-2 has about 427 SVs, and LIBSVM about 290. The kernel used is polynomial with degree 3 and the benchmark has 8 features, therefore the dimension of the feature space is $\binom{10}{3} = 120$ (see, e.g., [7]); and, as expected, OISVM stops acquiring new SVs when there are exactly 120, although it loads a few more before reaching the limit, with respect to the other approaches. The accuracy (not displayed) is exactly the same. Again, notice that, after having acquired 120 SVs, OISVM will never acquire any more ever, while keeping the same accuracy, whereas the LIBSVMs do, as theoretically proved in [25].

Consider now Figure 1, right pane: the kernel used is Gaussian and its dimension is infinite. The benchmark is relevantly large (16100 samples) and complex (123 features). Nevertheless, with an η as small as 0.1, at the end OISVM has less than 5% of the SVs used by LIBSVM-2 and less than 8% with respect to LIBSVM. The accuracy is $0.063\% \pm 0.14$ worse than that of LIBSVM-2.

Lastly, consider Table 4.1, which shows the very same data in compact form for 4 more standard databases. OISVM attains a number of SVs which is about 6% to



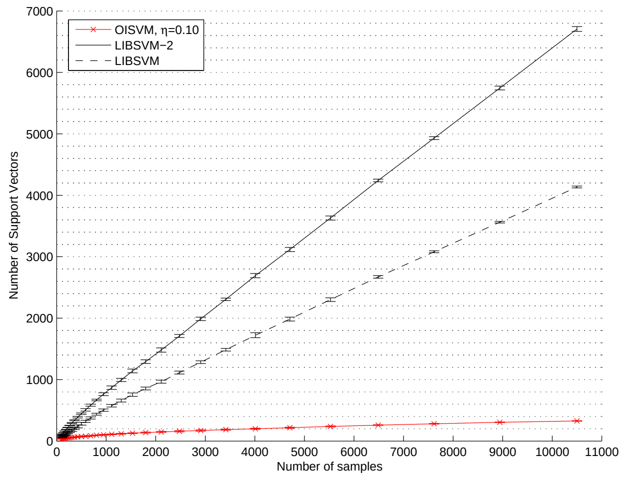


Figure 1. Comparison of OISVM and LIBSVM on the *Diabetes* (left pane) and *Adult7* (right pane) benchmarks. Diabetes is solved using a polynomial kernel with degree 3, while Adult7 is solved using a Gaussian kernel.

Benchmark name	Class. rate loss	% SVs vs. LIBSVM-2	% SVs vs. LIBSVM
Breast	0.47 ± 0.82	10.2 ± 0.87	22.1 ± 1.77
Diabetes	-0.52 ± 2.1	40.2 ± 2.1	55.2 ± 2.73
German	0.40 ± 1.15	6.1 ± 0.23	9.2 ± 0.35
Heart	-0.45 ± 1.01	10.3 ± 0.56	15.5 ± 0.94

Table 1. Comparison of OISVM and LIBSVM on more standard benchmarks, solved using a Gaussian kernel. For each benchmark, we report the difference in classification rate with respect to LIBSVM-2 and the percentage of the number of SVs with respect to LIBSVM and LIBSVM-2. The value of η has been chosen in order not to loose more than 0.5% accuracy.

slightly more than 55% of LIBSVM, whereas the accuracy is basically the same, being slightly better than LIBSVM in two cases (*Diabetes*, this time solved via a Gaussian kernel, and *Heart*).

4.2. Experiments with Real-world Application

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 a newly introduced database called IDOL2 (Image Database for rObot Localization 2, [21]), 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, this data capture natural variability that occurs in real-world environments because of both natural changes in the illumination and human activity. Fig. 2 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 algorithm. For further details about the database, we refer the readers to [21].

The evaluation was performed using composed receptive field histograms (CRFH) [18] as global image features and SIFT [19] for extracting local features. In the experiments, we consider both exponential χ^2 kernel for SVM (when use CRFH), and local kernels [28] (SIFT). Similarly to the previous set of experiments, we run the OISVM using different values of η, for both kernels.

We benchmarked OISVM against two approximate in-

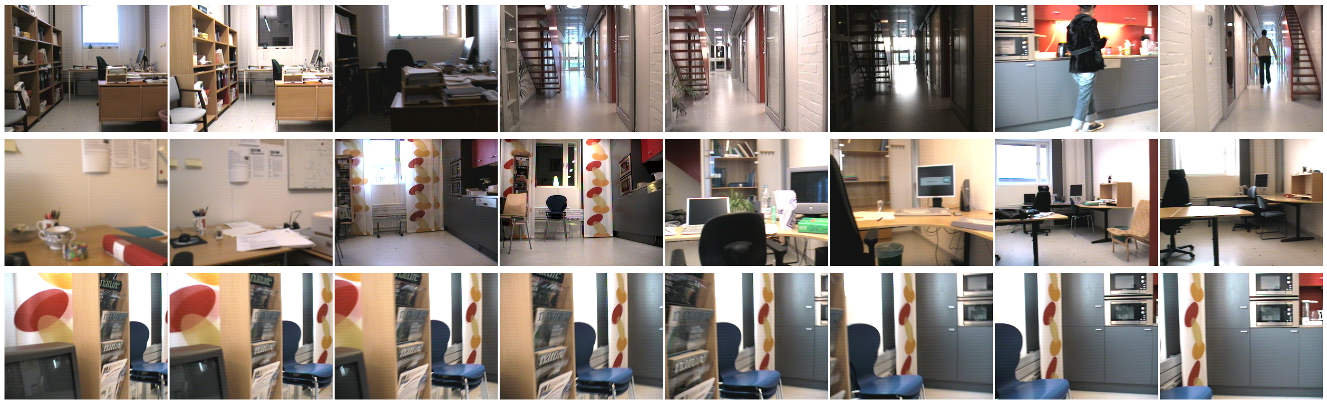


Figure 2. Sample images illustrating the variations captured in the IDOL2 database. Images in the top row show the variability introduced by changes in illumination for two rooms (first six images) as well as people appearing in the environment. The middle row shows the influence of people's everyday 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.

cremental SVM extensions: the standard fixed-partition technique [26] and the memory-controlled incremental SVM [24], a recently introduced version of incremental SVM, inspired by the work of Downs et al. [11]; this replicates the experimental setup of [20], thus allowing for a straightforward comparison between exact and approximate methods on this task [1]. The algorithm was 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 splitted 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 [20], 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, bottom row of Fig. 2 shows some sample images acquired within a time span of 1.6 sec. This setup allows us to examine how the algorithms perform on data with less variations. 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 12 permutations for the matching kernel [2]; here we report average results with standard deviation. Fig. 3, left, shows the recognition rates of the exponential χ^2 kernel (top) and matching kernel (bottom) experiments obtained at each step using OISVM and the other approximate incremental algorithms. Fig. 3, right, reports the number of support vectors stored in the model at each step of the incremental procedure, for both kernel types.

[1]We gratefully thank Jie Luo who gave us the software and assistance necessary to replicate the experiments.

[2]Upon acceptance of the paper, we will report results on 36 permutations for the matching kernel as well (experiments currently running).

We see that, performance-wise, all methods achieves statistically comparable results; this is true for both kernel types. Regarding the growth of the SV solution as the incremental steps grow, the OISVM algorithm shows a considerable advantage with respect to the fixed partition and the memory controlled incremental methods. In the case of the exponential χ^2 kernel this advantage is truly impressive (Fig 3, top right): for $\eta = 0.017$ and 0.025 the reduction in the number of stored SV is of $34\%/21\%$ (final incremental step) and compared to the fixed-partition method, and of $55\%/36\%$ (final incremental step) with respect to the memory-controlled algorithm. Even more important, OISVM, for these two values of η, has found a plateau in memory, while for the other two methods the trend seems to be of a growth (less pronounced for the memory controlled algorithm, which plot might be interpreted as an asymptotic plateau) 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 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 with respect to the memory-controlled method, and there isn't a 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 [5], which might affect the algorithm. Also, the algorithm might not reach a plateau in the SVs growth because, in the induced space of the matching kernel, there seems to be a high probability that couples training points are orthogonal or almost orthogonal (note that as the kernel is not a Mercer one, the geometric interpretation might not be valid).

We can conclude that these results are in agreement with those obtained on the machine learning benchmark databases, and show the effectiveness of OISVM in controlling the growth of the SV solution for online learning scenarios.

5. Conclusions

We presented a new method to reduce the number of SVs needed by a Support Vector Machine in an online setting, called OISVMs (Online Independent Support Vector Machines). OISVMs avoid using in the solution those support vectors which are linearly dependent of previous ones in the feature space — in other words, the kernel matrix is always kept at full rank. The optimization problem is solved via an incremental algorithm which benefits of the small size of the kernel matrix.

We tested the method both on a standard set of benchmark databases and a real-world case study, namely place recognition in an indoor environment, from sequences acquired by robot platforms under different weather conditions and across a time span of several months.

The experimental results show that (i) in the case of finite-dimensional kernels, OISVMs attain the theoretical limit of linearly independent support vectors allowed by the feature space, without losing any accuracy with respect to ordinary SVMs; (ii) in the case of infinite-dimensional kernels, they dramatically reduce the number of support vectors at the price of a negligible degradation in the accuracy. In fact, OISVMs obtain an accuracy up to 0.063% worse than SVMs with less than 5% of the support vectors of a standard SVM on a set of standard benchmark databases; whereas, on the real-world application, we get as few as one third of the SVs required by the fixed-partition method, while retaining essentially the same accuracy.

A current limitation of the algorithm is that it requires to store in memory all training data. This is unfeasible for applications like place recognition by robot platforms, as it might lead to a memory explosion. A possible solution might be to combine the OISVM with the KNN-SVM algorithm [30], so to keep separated the data storage from the discriminative classification. Another possibility is to discard training data in a principled way, so to minimize the effect on the SV solution. Future work will focus on this issue.

References

[1] M. Artač, M. Jogan, and A. Leonardis. Mobile robot localization using an incremental eigenspace model. In *Proc. ICRA'02*, 2002. 2

[2] F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005. 2

[3] G. Baudat and F. Anouar. Feature vector selection and projection using kernels. *Neurocomputing*, 55(1-2):21–38, 2003. 2

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

[5] S. Boughorbel, J.-P. Tarel, and F. Fleuret. Non-mercer kernels for svm object recognition. In *Proceedings of British Machine Vision Conference (BMVC'04)*, pages 137 – 146, London, England, 2004. 6

[6] E. Brunskill and N. Roy. Slam using incremental probabilistic pca and dimensionality reduction. In *Proc. ICRA'05*, 2005. 2

[7] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998. 2, 3, 4

[8] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *NIPS*, pages 409–415, 2000. 2, 4

[9] 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`. 4

[10] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods)*. CUP, 2000. 1, 2

[11] T. Downs, K. E. Gates, and A. Masters. Exact simplification of support vectors solutions. *Journal of Machine Learning Research*, 2:293–297, 2001. 2, 3, 6

[12] Y. Engel, S. Mannor, and R. Meir. Sparse online greedy support vector regression. In *Proc. 13th European Conference on Machine Learning*, 2002. 3

[13] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least squares algorithm. *IEEE Transactions on Signal Processing*, 52(8), 2004. 2

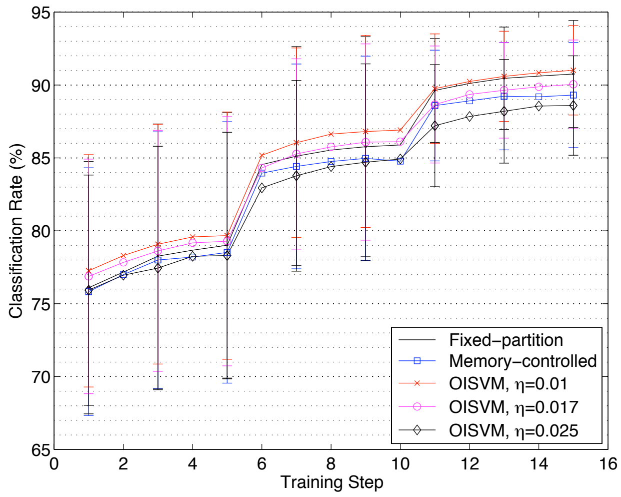
[14] S. S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 8:1–22, 2006. 1, 2, 4

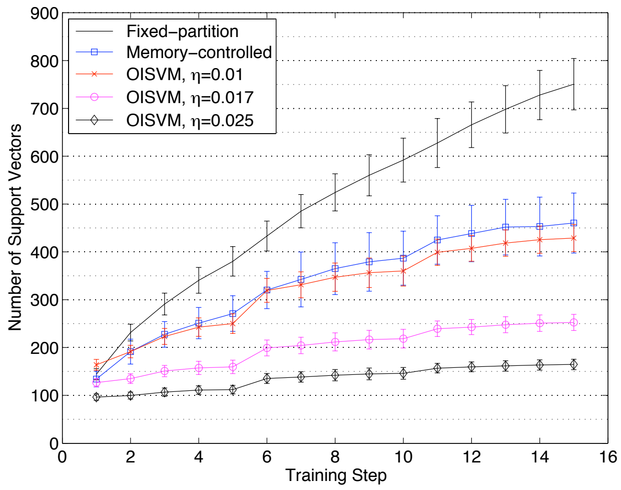
[15] S. S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005. 4

[16] P. Laskov, C. Gehl, S. Krger, and K.-R. Müller. Incremental support vector learning: analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909—1936, 2006. 1, 2

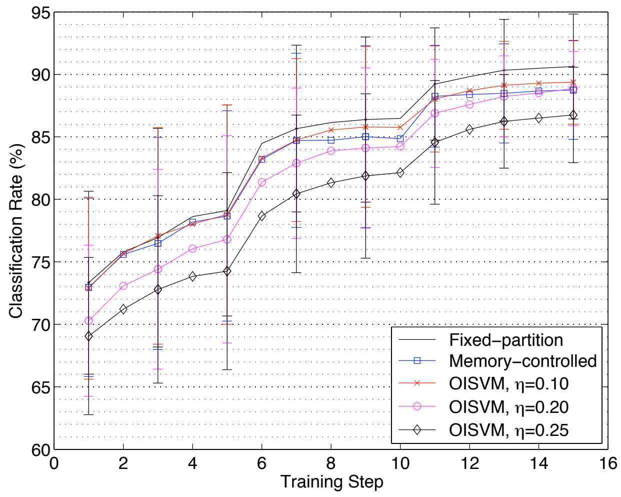
[17] Y. J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceedings of the SIAM International Conference on Data Mining*, 2001. 2

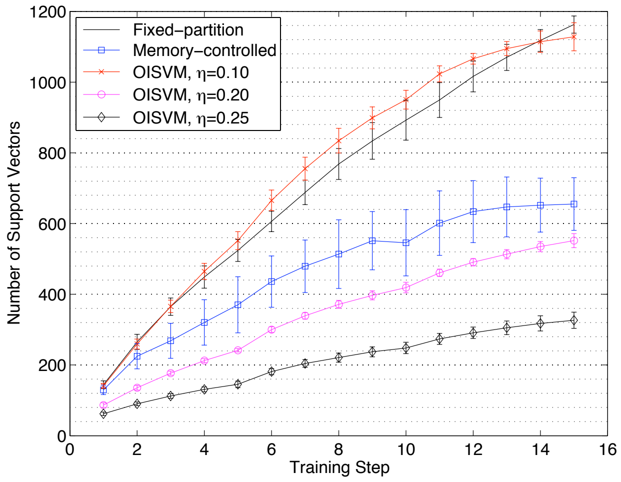
[18] O. Linde and T. Lindeberg. Object recognition using composed receptive field histograms of higher dimensionality. In *Proc. ICPR'04*. 5





(a) Number of support vectors and classification rate obtained at each incremental step using χ^2 kernel.





(b) Number of support vectors and classification rate obtained at each incremental step using local kernel.

Figure 3. Average results obtained for experiment performed on the IDOL2 database, using OISVM with three different values of η, the fixed-partition and memory-controlled methods.

[19] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of ICCV'99*. 5

[20] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. Incremental learning for place recognition in dynamic environments". Number IDIAP-RR-06-52, 2006. Available at `http://www.idiap.ch/publications.php`. 6

[21] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. The KTH-IDOL2 database. Technical Report 304, KTH, CAS/CVAP, 2006. Available at `http://cogvis.nada.kth.se/IDOL2/`. 5

[22] D. Nguyen and T. Ho. An efficient method for simplifying support vector machines. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 617–624, New York, NY, USA, 2005. ACM Press. 2

[23] M. Pontil and A. Verri. Properties of support vector machines. *Neural Computation*, 10:955–974, 1998. 2

[24] A. Pronobis and B. Caputo. The more you learn, the less you store: memory-controlled incremental support vector machines. In *International Cognitive Vision Workshop*, Gratz, Austria, 2006. 4, 6

[25] I. Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4:1071–1105, 2003. 1, 2, 4

[26] N. Syed, H. Liu, and K. Sung. Incremental learning with support vector machines. In *Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence (IJCAI-99)*, 1999. 4, 6

[27] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998. 1

[28] C. Wallraven, B. Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *Proc. of ICCV'03*. 5

[29] M. Wu, B. Schölkopf, and G. Bakir. A direct method for building sparse kernel learning algorithms. *Journal of Machine Learning Research*, 7:603–624, 04 2006. 2

[30] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: discriminative nearest neighbor classification for visual category recognition. In *Proc. CVPR'06*, 2006. 7