# The More you Learn, the Less you Store: Memory-Controlled Incremental SVM

Andrzej Pronobis and Barbara Caputo
Computational Vision and Active Perception Laboratory
School of Computer Science and Communication
Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden
{pronobis,caputo}@nada.kth.se

## Abstract

*The capability to learn from experience and update incrementally its internal representation is a key property for a visual recognition algorithm aiming to work in real-world scenarios. This paper presents a novel SVM-based algorithm for visual object recognition, capable of learning model representations incrementally. We combine an incremental extension of SVMs [14] with a method which reduces the number of support vectors needed to build the decision function without any loss in performance [5]. The resulting algorithm is guaranteed to achieve the same recognition performance as the original incremental method while reducing the memory requirements. We then introduce a parameter which permits a user-set trade-off between performance and memory reduction. This property is potentially useful in applications where the memory size of the visual models must be kept under control. Results show that it is possible to achieve a consistent reduction of the memory requirements with only a moderate loss in performance. For example, experiments show that when the user accepts a reduction in recognition rate of 5%, this yields a memory reduction of up to 50%.*

## 1. Introduction

Basic visual operations such as categorization and complex tasks such as scene interpretation have long been major challenges in computer vision. A system able to perform these tasks should include facilities for understanding and learning, where understanding here means both recognition and categorization of objects and scenes. A system with a realistic complexity cannot be engineered: this calls for methods for automatic acquisition of models and representations allowing the system to work in an open-ended fashion, i.e. beyond initial specification. A highly desirable property for a visual recognition algorithm working in realistic settings is the capability to learn from experience and update incrementally its internal representation. The possibility to learn continuously is particularly important for all those applications where, in spite of a vast batch training set and unlimited training time, it is impossible to provide a database which will remain representative of the modeled visual class during time. For instance, some visual categories like phones or computers have grown dramatically over the last 20 years. New category members like cell phones and laptops have appeared, changing our visual models of these categories. Another example is indoor place recognition, where the variability of a room's appearance is so high (people using the room, furniture relocated or changed, objects being taken out of drawers, illumination changes and so forth) that is virtually impossible to collect a training database covering all these possibilities.

Discriminative methods have become widely popular for visual recognition, achieving impressive results on several applications [16, 19, 8, 10]. Within discriminative classifiers, SVM techniques provide powerful tools for learning models with good generalization capabilities; in some domains like object and material categorization, SVM-based algorithms are state of the art [1, 6]. This makes it worth to investigate whether it is possible to perform continuous learning with this type of methods. Several incremental extensions of SVMs have been proposed in the machine learning community [4, 2, 14, 9, 15]. Between these methods, the approximate techniques [4, 9, 14] seem better suited for visual recognition because, at each incremental step, they discard non-informative training vectors, thus reducing the memory requirements. Other methods, such as [2], instead require to store in memory all the training data, eventually leading to a memory explosion; this makes them unfit for continuous learning of visual patterns.

This paper presents a novel SVM-based incremental method which performs like the batch algorithm while reducing the memory requirements. We combine an approximate technique for incremental SVM [14] with an exact method that reduces the number of support vectors needed to build the decision function without any loss in perfor-

mance [5]. This results in a novel algorithm performing as the original incremental method with a reduction in the memory requirements. We then present an extension of the method for the exact simplification of the support vector solution [5]. We introduce a parameter that links SVM's performance to the amount of vectors that is possible to discard. This allows a user-set trade-off between performance and memory reduction. The algorithms were tested on a material categorization problem [1], showing a remarkable reduction of the memory requirements compared to standard SVM. In summary, the contributions of this paper are:

- We implemented and tested the fixed-partition incremental SVM [14] and benchmarked it against the batch algorithm. Results show that their performance is statistically equivalent, but the incremental method does not consistently achieve a memory reduction compared to the batch model. To our knowledge, these experiments constitutes the first application of an incremental SVM method to visual recognition.

- We implemented a method for the exact simplification of the support vector solution [5]. The algorithm was further extended by introducing a parameter, to be set by the user, which allows a trade-off between performance and memory. Although our interest here was in combining these algorithms with incremental SVM, these methods can be used for any SVM-based classification algorithm. To our knowledge, this is the first time that the method presented in [5] is tested on visual data.

- We combined these algorithms obtaining a new incremental SVM with a mechanism for memory control. At each incremental step the number of support vectors to be stored can be reduced, depending on the application, *(a)* without any loss in performance, or *(b)* with a controlled decrease in recognition rate which yields a consistent memory reduction. Experiments show that in the case *(a)* the algorithm achieves a reduction on the number of stored support vectors of 32%. In the case *(b)*, results show that when it is acceptable a reduction in recognition rate of 5%, this yields a reduction of 50% in the number of stored support vectors.

The paper is organized as follows: section 2 reviews approximate techniques for incremental SVM and presents an experimental evaluation of one of these methods. Section 3 describes the memory reduction algorithms and evaluates their performance for different kernel types. Section 4 presents our memory-controlled incremental SVM and shows its effectiveness with a set of experiments. The paper concludes with a summary discussion and possible directions for future work.

[1] A similar experimental evaluation, using both local and global descriptors and several kernel types (including non-Mercer kernels [20] was conducted in the domain of place recognition. The experimental findings are similar to those reported here, and thus are omitted for space's reasons. The interested reader can find more details in [12].

## 2 Incremental SVM

This section presents the incremental SVM technique that will be one of the building blocks of our new memory--controlled algorithm. After a brief review of the theory behind this type of algorithms (section 2.1), the remaining of the section describes the approximate technique used in this paper (section 2.2) and it presents an experimental evaluation of its performance against the batch method (section 2.3).

### 2.1 SVM: the batch algorithm

Support Vector Machines (SVMs, [3]) belong to the class of large margin classifiers. Consider the problem of separating the set of training data $(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_m, y_m)$, where $\boldsymbol{x}_i \in \Re^N$ is a feature vector and $y_i \in \{-1, +1\}$ its class label. If we assume that the two classes can be separated by an hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$, the optimal hyperplane is the one which has maximum distance to the closest points in the training set. The optimal values for $\boldsymbol{w}$ and $b$ can be found by solving a constrained minimization problem, which results in a classification function

$$f(x) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b\right), \qquad (1)$$

where $\alpha_i$ and $b$ are found by using an SVC learning algorithm [3]. Most of the $\alpha_i$s' take the value of zero; those $\boldsymbol{x}_i$ with nonzero $\alpha_i$ are the "support vectors". The extension to multiclass can be done following several strategies [3]; here we used the pairwise approach [3]. To obtain a nonlinear classifier, one maps the data from the input space $\Re^N$ to a high dimensional feature space $\mathcal{H}$ by $\boldsymbol{x} \rightarrow \Phi(\boldsymbol{x}) \in \mathcal{H}$, such that the mapped data points of the two classes are linearly separable in the feature space. Assuming there exists a kernel function $K$ such that $K(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{y})$, then a nonlinear SVM can be construct by replacing the inner product $\boldsymbol{x} \cdot \boldsymbol{y}$ in the linear SVM by the kernel function $K(\boldsymbol{x}, \boldsymbol{y})$

$$f(\boldsymbol{x}) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right) \qquad (2)$$

This corresponds to constructing an optimal separating hyperplane in the feature space. In this paper we will use an RBF kernel $K(\boldsymbol{x}, \boldsymbol{y}) = \exp\{-\gamma||\boldsymbol{x} - \boldsymbol{y}||^2\}$.

**Figure 1. The variations within each category of the TIPS2 database. Each row shows one example image from each of four samples of a category. In addition, each sample was imaged under varying pose, illumination and scale conditions.**

### 2.2 SVM: an Incremental Extension

Among all incremental SVM extensions proposed in the machine learning literature so far [14, 4, 11, 2], approximate methods seem to be the most suitable for visual recognition: firstly - as opposed to exact methods like [2]- they discard a significant amount of the training data at each incremental step. Secondly, they are expected to achieve performances not too far from those obtained by an SVM trained on the complete data set (batch algorithm), because at each incremental step the algorithm remembers the essential class boundary information regarding the data seen so far (in form of support vectors). This information contributes properly to generate the classifier at the next iteration.

Once a new batch of data is loaded into memory, there are different possibilities for the updating of the current model, which might discard a part of the new data according to some fixed criteria [4, 14]. In this paper we use the fixed--partition technique, which was introduced first in [14]. In this method the training data set is partitioned in batches of fixed size $k$:

$$\boldsymbol{T} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots \boldsymbol{T}_n\},$$

with

$$\boldsymbol{T}_i = \{(\boldsymbol{x}_j^i, y_j^i)\}_{j=1}^k.$$

At the first step, the model is trained on the first batch of data $\boldsymbol{T}_1$, obtaining a classification function

$$f_1(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m_1} \alpha_i^1 y_i^1 \boldsymbol{x}_i^1 \cdot \boldsymbol{x} + b^1\right).$$

At the second step, a new batch of data is loaded into memory; then, the *new* training set becomes

$$\boldsymbol{T}_2^{inc} = \{\boldsymbol{T}_2 \cup \boldsymbol{S}V_1\}, \qquad \boldsymbol{S}V_1 = \{\boldsymbol{x}_i^1\}_{i=1}^{m_1},$$

where $\boldsymbol{S}V_1$ are the support vectors learned at the first step. The new classification function will be:

$$f_2(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m_2} \alpha_i^2 y_i^2 \boldsymbol{x}_i^2 \cdot \boldsymbol{x} + b^2\right).$$

Thus, as new batches of data points are loaded into memory, the existing support vector model is updated, so to generate the classifier at that incremental step. Note that this incremental method can be seen as an approximation of the chunking technique used for training SVM [3]. Indeed, the chunking algorithm is an exact decomposition which iterates through the training set to select the support vectors. The fixed-partition incremental method instead scan through the training data just once, and once discarded, does not consider them anymore. The fixed-partition incremental algorithm has been tested on several benchmark databases commonly used in the machine learning community [4] and on a simple optical character recognition problem [11], obtaining good performances compared to the batch algorithm and other approximate methods.

### 2.3 Experimental Evaluation

In order to evaluate the fixed-partition incremental SVM, we performed a set of experiments on the TIPS2 database [1]. This database contains 11 material categories (4 planar samples for material, Fig 1). Many of the materials have 3D structure, implying that their appearance can change

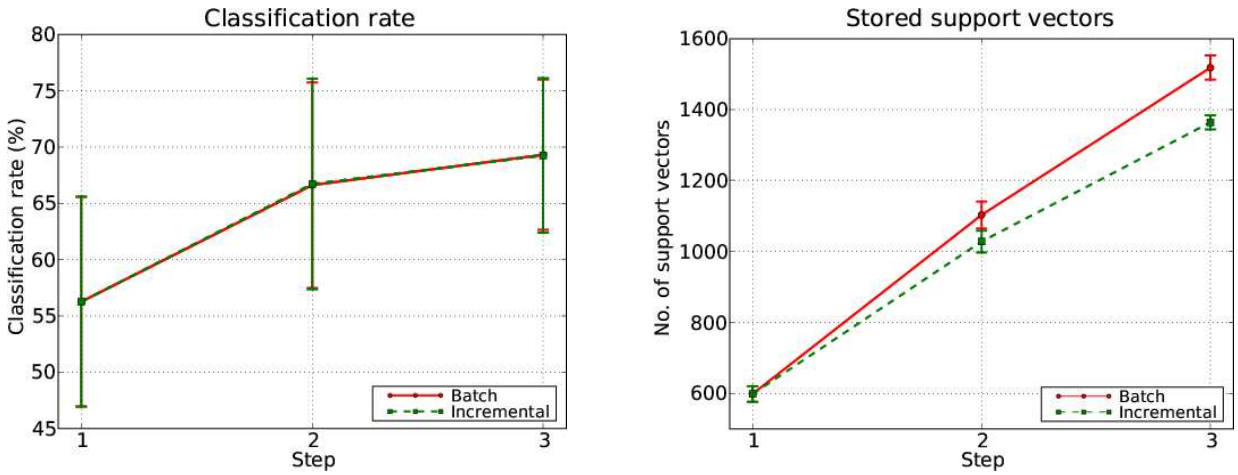


**Figure 2. Results on the TIPS2 database using incremental SVM and the batch algorithm. The incremental method achieves a memory reduction while obtaining the same performance as batch SVM**

considerably as pose and lighting are changed. TIPS2 contains images at 9 scales equally spaced logarithmically over two octaves. At each scale, materials were imaged under 3 poses (frontal, rotated $22.5°$ left and $22.5°$ right) and 4 illumination conditions (frontal, $45°$ from the top and $45°$ from the side, all taken with a desk-lamp with a Tungsten light bulb; the fourth illumination condition consisted of the fluorescent lights in the laboratory). In total there are $9 \times 3 \times 4 = 108$ images per sample. As descriptors we used the rotationally invariant MR8 [18] which has shown good performances on this database [1]. For these experiments we used our extended version of the *libsvm* software, and we set $C = 100$. Kernel parameters were determined via cross-validation. We splitted the database in a training and test set, with the training set consisting of three samples per material, and the test set consisting of the remaining fourth material; as in [1], we considered 4 possible splits. The training set was further divided in three subsets, each consisting of all the views for one sample per material. Each subset was added to the model during each incremental step. For each partition, we performed experiments on four different orderings of the incremental steps. For instance, for the partition with sample 1, 2, 3 into the training set we ran 4 different experiments considering the incremental sequences (a) 1, 2, 3; (b) 2, 1, 3; (c) 2, 3, 1 (d) 3, 2, 1. Thus, in total we ran 16 different experiments; here we report the averaged results with standard deviations. Fig 2, left, shows the recognition rates obtained, at each step of the incremental update, using the batch SVM on the whole training data and the fixed-partition incremental algorithm. Fig 2, right, shows the number of support vectors stored by both algorithms at each step of the incremental procedure. We see that there is no loss in performance of the incremental algorithm compared to the batch one, while there is a statistically significant reduction of the number of support vectors in the incremental algorithm. These results are in agreement with those reported in [4], where only two-class problems were considered.

From these results, and from those reported in [4, 14, 11], we can conclude that the fixed-partition incremental SVM performs as the batch algorithm. A significant drawback of the method is that it does not guarantee a reduction of the number of stored support vectors; this is a serious issue, as it may lead to a memory explosion. In the next section we will present a method for the reduction of the memory requirements of the support vector solution. As it will be shown in section 4, this method, combined with the fixed-partition approach, can be used to design a memory-controlled incremental SVM.

## 3 Exact Simplification of SVM Solution

Experiments presented in section 2.3 showed that the fixed-partition incremental SVM can achieve the classification performance of the batch algorithm while using fewer support vectors. While this suggests that the solution found using the standard SVC learning algorithm is not always minimal, experiments presented in [14] showed that rejecting even a small amount of support vectors may cause a strong decrease in performance. This raises the question of whether the complexity of the support vector solution can be reduced while preserving its optimal performance. A possible solution has been proposed by Downs *et al* [5]. Their method reduces the number of support vectors of a

trained classifier, eliminating those which can be expressed as a linear combination of the others in the feature space. The weights are updated accordingly, which ensures that the decision function is exactly the same as the original. This results in a reduction of the complexity of the classifier, without any loss in performance.

The rest of this section is organized as follows: section 3.1 reviews the method presented in [5], gives some details on its implementation and presents an extension of the algorithm that allows the user to trade performance for memory requirements, when necessary. Section 3.2 describes a series of experiments evaluating both methods. Although we developed the algorithm having in mind its integration with incremental SVM techniques, it can be used for reducing the memory requirements (as well as speed during recognition) of any SVM-based classification method.

### 3.1 The Algorithm

The idea behind the algorithm by Downs *et al* [5] is that the set of support vectors $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^m$ is not guaranteed to be linearly independent. Let us suppose that the first $r$ support vectors are linearly independent, and the remaining $m-r$ depend linearly on those in the feature space: $\forall j = r+1, \dots m$, $\boldsymbol{x}_j \in span\{\boldsymbol{x}_i\}_{i=1}^r$. Then it holds

$$K(\boldsymbol{x}, \boldsymbol{x}_j) = \sum_{i=1}^{r} c_{ij} K(\boldsymbol{x}, \boldsymbol{x}_i), \tag{3}$$

and the classification function (2) can be rewritten as

$$f(\boldsymbol{x}) = \mathrm{sgn}\,(\sum_{i=1}^{r} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + \sum_{j=r+1}^{m} \alpha_j y_j \sum_{i=1}^{r} c_{ij} K(\boldsymbol{x}, \boldsymbol{x}_i) + b). \tag{4}$$

If we define the coefficients $\gamma_{ij}$ such that $\alpha_j y_j c_{ij} = \alpha_i y_i \gamma_{ij}$ and $\gamma_i = \sum_{j=r+1}^{m} \gamma_{ij}$, then eq. (4) can be written as

$$\begin{aligned} f(\boldsymbol{x}) &= \mathrm{sgn}\,(\sum_{i=1}^{r} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + \sum_{i=1}^{r} \alpha_i y_i \sum_{j=r+1}^{m} \gamma_{ij} K(\boldsymbol{x}, \boldsymbol{x}_i) + b) \\ &= \mathrm{sgn}\left(\sum_{i=1}^{r} \widehat{\alpha}_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b\right) \end{aligned} \tag{5}$$

where

$$\widehat{\alpha}_i = \alpha_i(1+\gamma_i) = \alpha_i \left(1 + \sum_{j=r+1}^{m} \frac{\alpha_j y_j c_{ij}}{\alpha_i y_i}\right). \tag{6}$$

Thus, the resulting classification function (eq. (5)) requires now $m-r$ less kernel evaluations than the original one (eq. (2)). In order to find the linearly independent subset of the support vectors and the values of the $c_{ij}$ coefficients, we applied the QR factorization algorithm with column pivoting [7] to the support vector matrix.

Two interesting points can be made on the QR factorization and the column pivoting strategy: first, it allows to reveal the numerical rank of the matrix with respect to a parameter $\tau$, which acts as a threshold in defining the condition of linear dependence. Second, the algorithm performs a permutation of the columns of the matrix such that, if for a given value of $\tau$ the rank of the matrix is $r$, then the linearly independent columns will occupy the first $r$ positions. Also, these $r$ columns will be ordered according to the degree of their relative linear independence. On the basis of these observations, we propose to consider the threshold $\tau$ as a parameter of the algorithm that allows the user to control the number of support vectors to be kept in memory. Clearly, as the value of $\tau$ grows, eq. (5) will become more and more an approximation of the exact solution. Anyway, we want to underline that the informative content of a discarded support vector $\boldsymbol{x}_j$ is not completely lost, as its weight $\alpha_j$ is used to compute the updated value of the weights $\widehat{\alpha}_i$ for the remaining support vectors. This should result in a graceful decrease of classification performance compared to the optimal solution. Thus, the parameter $\tau$ can be used as an effective way to trade performance for memory requirements and speed during classification, depending on the task at hand.

### 3.2 Experimental Evaluation

We tested the algorithm with an extensive set of experiments on the TIPS2 database. For each experiment, we first trained the classifier using the standard SMO algorithm. Then, starting from the obtained decision function, we applied the reduction algorithm increasing the value of the $\tau$ parameter, which led to a progressive reduction of the classification rates and of the number of support vectors. Table 1 shows the results obtained for a series of experiments where the training set consisted of 3 samples per material, and the test set consisted of the remaining sample. As in section 3.2, experiments were performed on 4 different partitions and we report here the averaged results (the symbol $\Theta$ was used to denote the percentage of the original classification rate that is guaranteed to be preserved after the reduction [2]). We see that the algorithm achieves a reduction in the number of stored vectors of $\sim$ 47%, while keeping

[2] In the experiments presented here, we considered the classification rate of the resulting solution as a constraint on the amount of reduced support vectors. However, depending on the application, the problem may be reformulated to provide the amount of desired memory reduction.

| | Θ (%) | Class. rate (%) | No. of SVs | Red. rate (%) |
|---|---|---|---|---|
| ORIG. | | 69.32±6.67 | 1518±34 | — |
| REDUCED | 100 | 69.32±6.67 | 805±111 | 46.95±7.03 |
| | 98 | 67.93±6.54 | 744±56 | 50.98±3.31 |
| | 95 | 65.85±6.34 | 616±77 | 59.43±4.65 |
| | 90 | 62.39±6.01 | 539±82 | 64.46±5.13 |
| | 80 | 55.45±5.34 | 428±83 | 71.82±5.24 |

**Table 1. Average results of the evaluation of the reduction algorithm on the TIPS2 database. The Θ parameter denotes a percentage of the original classification rate, that is guaranteed to be preserved after the reduction. The uncertainties are given as one standard deviation.**

the classification rate intact. When the application allows a small loss in performance, memory requirements may be further reduced by exploiting the ability of the algorithm to approximate the solution. Note that the classification rate decreases monotonically with the number of support vectors.

Results presented in Table 1, left, were obtained using a Gaussian kernel. The experiments were repeated for other kernel types (polynomial and $\chi^2$ kernels), and different number of training samples per material (1 or 2), yielding results consistent with those reported here. It is interesting to observe that the reduction rate in support vectors grows with the dimension of the training set. For instance, the average reduction rate obtained training on one sample per material is $\sim$44% when it is acceptable a decrease in recognition rate of 5%. This value is considerably lower of the $\sim$59% in reduction rate obtained under similar conditions, using 3 samples per material during training (Table 1).

## 4 Memory-controlled Incremental SVM

The fixed-partition incremental learning algorithm described in section 2 was shown to perform well on visual data. However, experiments revealed that at each incremental step the memory requirements can grow considerably. This is a serious limitation for an incremental method aiming to work in a cognitive visual system. In section 3 we presented a technique for controlling the amount of stored support vectors in a principled way, and we extended it so to obtain an even greater reduction rate when the user accepts a fixed decrease in performance. This is a reasonable assumption, particularly for multi-sensory systems. Our idea is to combine these two algorithms together, obtaining a new incremental SVM method with a mechanism for a controlled growth of the memory requirements. We propose to apply the reduction algorithm at *each incremental step*. The new representation of the data is then built from the remaining support vectors. We will show through experiments that this approach can successfully control the amount of vectors to be kept in memory.

We benchmarked our new algorithm against the fixed-partition incremental SVM by repeating the experiments described in section 2.3. Fig 3 reports the obtained results: on left, it shows the recognition rates achieved using different values of the parameter Θ, and on right it shows the reduction rates obtained at each incremental step. Note that the parameter Θ decides the amount of discarded support vectors at each incremental step.

We first observe that our new method controls the memory growth much more successfully than the original incremental technique (Fig 3, top right). This is especially true when it is accepted a reduction Θ in classification rate (Fig 3, middle and bottom right). It is interesting to note that, for $\Theta = 95\%$ and $\Theta = 90\%$, the gain in memory compression is always greater than the overall reduction in performance. A last word should be said regarding the training time at each incremental step. On one side, our method uses two algorithms in cascade while the original incremental technique uses just one. On the other side, the training time for an SVM depends on the dimension of the training set, and we have shown experimentally that our method yields far more consistent reductions. Thus, as the incremental learning proceeds, the training time of our algorithm actually becomes comparable, and eventually lower, than the original methods.

## 5 Conclusions

In this paper we explored the possibility to extend SVMs to incremental learning for visual recognition. Starting from an existing incremental SVM algorithm, we proposed a new method which is capable of learning model representations incrementally while controlling the number of support vectors to be stored. This is obtained combining the fixed-partition incremental technique with a method which reduces the number of support vectors needed to build the decision function without any loss in performance. A further extension of the algorithm permits a user-set trade-off between performance and memory reduction. An experimental evaluation of the method shows its potential for computer vision applications.

This work can be extended in many ways. First, here we chose the fixed partition technique for incremental SVM, but other approximate methods might be more suitable and/or perform better. Thus, we plan to develop memory-controlled version of those algorithms and to benchmark them with our method. These methods should also be com-

pared with other incremental methods presented in the vision literature, like for instance [13]. Second, we would like to study the algorithm's performance as the dimension of the batch set changes, with respect to different multi-class extension, and to test it on the domain of human action recognition. Finally, we plan to speed-up our algorithm by incorporating fast training techniques like [17], so to move towards on-line, continuous learning using SVMs.

**Acknowledgments**

This work has been supported by the EU project FP6-IST-0042500 *CoSy*.

## References

[1] B. Caputo, E. Hayman, P. Mallikarjuna. Class-specific material categorisation. *Proc ICCV05*.

[2] G. Cauwenberghs, T. Poggio. Incremental and decremental support vector machine learning. *Proc NIPS00*.

[3] N. Cristianini, J. S. Taylor, *"An introduction to support vector machines and other kernel-based learning methods"*, Cambridge university press, 2000.

[4] C. Domeniconi, D. Gunopulos. Incremental support vector machine construction. *Proc ICDM01*.

[5] T. Downs, K. E. Gates, A. Masters. Exact simplification of support vector solutions. *JMLR*, 2: 293-297, 2001.

[6] M. Fritz, B. Leibe, B. Caputo, B. Schiele. Integrating representative and discriminant models for object category detection. *Proc ICCV05*.

[7] G. H. Golub, C. F. Van Loan. Matrix computations (3rd ed.). Johns Hopkins University Press, 1996.

[8] K. Grauman, T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. *Proc ICCV05*.

[9] P. Mitra, C. A. Murthy, S. K. Pal. Data condensation in large databases by incremental learning with support vector machines. *Proc ICPR00*.

[10] A. Opelt, A. Pinz, M. Fussenegger, P. Auer. Generic object recognition with boosting. *IEEE Tran on PAMI*, Vol 28, N 3, March 2006.

[11] B. Peng, J. Xiaugyn, Z. Sun, L. Wenyin. Study of SVM--based incremental learning for user adaptation in multi-class classification environment. *Proc ICONIP02*.

[12] A. Pronobis. Indoor Place Recognition Using Support Vector Machines. M. Sc Thesis, NADA/CVAP, KTH, Dec 2005. Available at http://www.nada.kth.se/ pronobis/

[13] D. Skocaj, A. Leonardis. Weighted and robust incremental method for subspace learning. *Proc ICCV03*

[14] N. A. Syed, H. Liu, K. K. Sung. Incremental learning with suppoert vector machines. *Proc IJCAI99*.

[15] S. Tong, D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. *JMLR*, 2: 45-66, 2001.

[16] A. Torralba, K. Murphy, W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. *Proc CVPR04*.

[17] I. W. Tsang, J. T. Kwok, P.-M. Cheung. Core vector machines: fast SVM training on very large data sets. *JMLR*, 6: 363-392, 2005.

[18] M. Varma, A. Zisserman. Classifying images of meterial: achieving viewpoint and illumination independence. *Proc ECCV02*.

[19] P. Viola, M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc CVPR01*.

[20] C. Wallraven, B. Caputo, A. Graf, "Recognition with local features: the kernel recipe", *Proc ICCV03*.

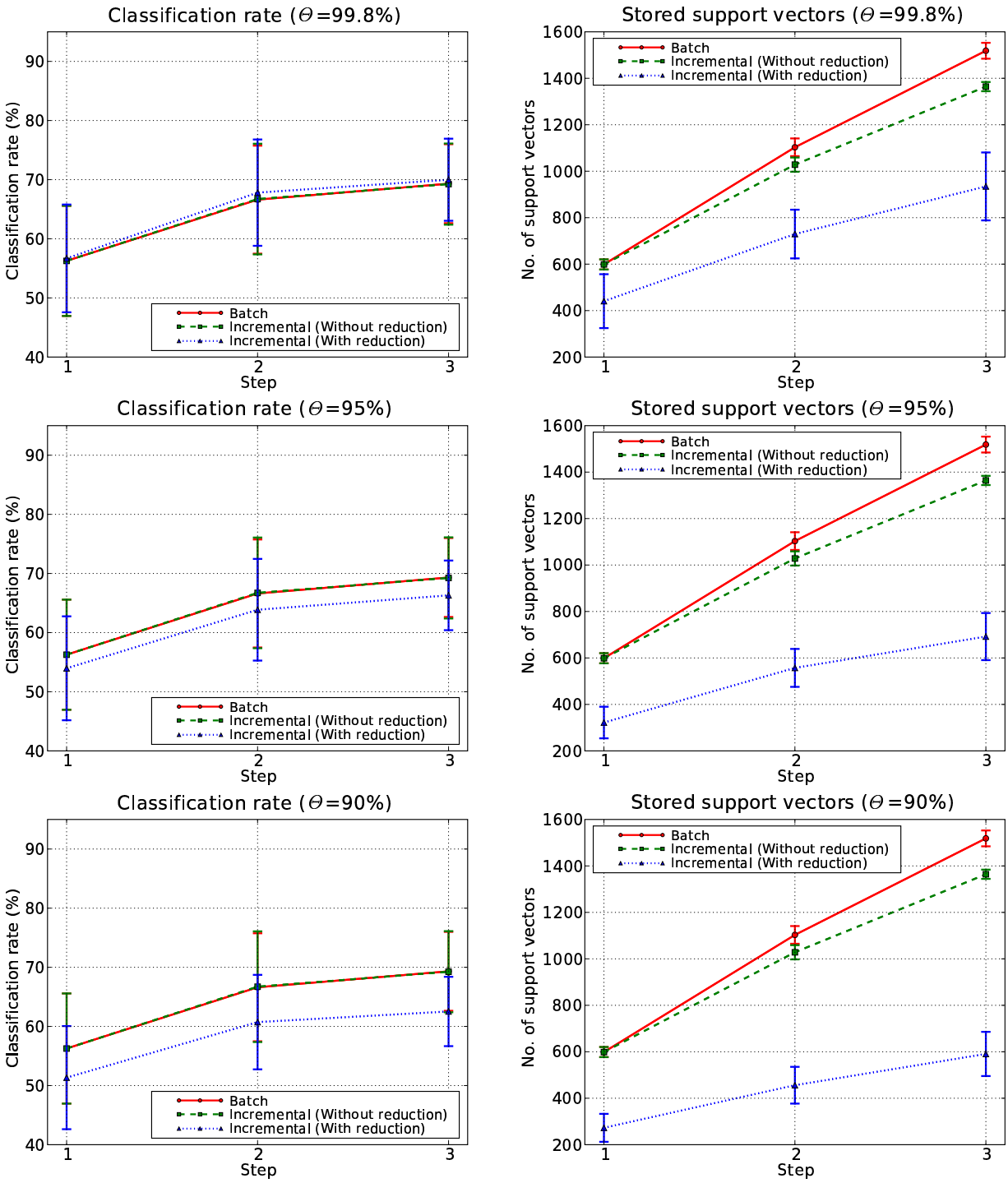


**Figure 3. Results on the TIPS2 database obtained using batch SVM, the fixed-partition incremental method, and the memory controlled incremental algorithm.**