Kernel-based online machine learning and support vector reduction

Sumeet Agarwal, V. Vijaya Saradhi and Harish Karnick [1,2]

Abstract

We apply kernel-based machine learning methods to online learning situations, and look at the related requirement of reducing the complexity of the learnt classifier. Online methods are particularly useful in situations which involve streaming data, such as medical or financial applications. We show that the concept of *span* of support vectors can be used to build a classifier that performs reasonably well while satisfying given space and time constraints, thus making it potentially suitable for such online situations.

1 Introduction

Kernel-based learning methods [16] have been successfully applied to a wide range of problems. The key idea behind these methods is to implicitly map the input data to a new feature space, and then find a suitable hypothesis in this space. The mapping to the new space is defined by a function called the kernel function. Due to promising generalization performance, kernel methods have been widely used in many applications. However, they may not always be the most efficient technique; training and classification times are the two main issues which concern researchers and practitioners. Many techniques have been proposed to speed up training time of (offline) SVMs. Sequential minimal optimization (SMO) [15], modified SMO [11], decomposition method [9] and low rank kernel matrix construction method [8] are some of the methods proposed towards speeding up the training time.

Email address: lawraga.teemus@gmail.com, {saradhi, hk}@cse.iitk.ac.in (Sumeet Agarwal, V. Vijaya Saradhi and Harish Karnick).
[1] IBM India Research Lab, New Delhi, India
[2] Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur, India

Classification time on the other hand is primarily determined by the number of support vectors (SVs). Vapnik has given a lower bound on the number of SVs in terms of the expected probability of error, $E[p]$, and the number of training data points $(N-1)E[p]$ [19]. Steinwart [17] established a relationship between the number of SVs and the number of training data points. This relation gives a lower bound on the number of SVs; thus it has a significant impact on the classifier's time complexity. Several techniques have been proposed to address the problem of reducing the time complexity of the SVM classifier as well. These include: the reduced set method [2, 16, 13], exact simplification of support vectors [7], classifier complexity reduction using basis vectors [10], sparse large margin classifiers [21], variant of reduced set method [14], relevance vector machine [18], classification on a budget [6] and online SVM with a budget [5].

1.1 Offline Learning

The reduced set method aims to minimize the distance between the original separating hyperplane (which uses all of the SVs) and a new hyperplane (which uses fewer vectors) for obtaining a reduced set of vectors and their associated coefficients. The new hyperplane is described in terms of vectors which are not SVs (and not necessarily from the training set) and do not lie on the margin. Let $\mathbf{w} = \sum_{i=1}^{N_{SV}} \alpha_i y_i \varphi(\mathbf{X}_i)$ be the optimal hyperplane in the mapped space using all the SVs and let $\mathbf{w}' = \sum_{i=1}^{N_{SV\,z}} \gamma_i \varphi(\mathbf{Z}_i)$ be a new hyperplane with a specified size of the set of vectors. The reduced set $\{(\gamma_i, \mathbf{Z}_i)\}_{i=1}^{N_{SV\,z}}$ is computed by minimizing the objective function: $||\mathbf{w} - \mathbf{w}'||$.

In [16], reduced subset *selection* methods (proposed by Burges [2]) and *extraction* methods were discussed in detail. In subset selection methods a set of vectors is obtained from the training dataset. On the other hand, in subset extraction methods, the set of vectors obtained need not be from the training dataset. Recently, a variant of the reduced set method has been proposed by DucDung Nguyen *et al.*, in [14]. In their method, two SVs of the *same* class are replaced by a vector that is a convex combination of the two. The weights are expressed in terms of the Lagrangian multipliers of the two SVs. Different expressions are used to obtain a new vector for Gaussian and polynomial kernels. Mingrui Wu *et al.,* [21] introduced an explicit constraint, $\mathbf{w} = \sum_{i=1}^{N_{SV\,z}} \beta_i \varphi(\mathbf{Z}_i)$, in the primal formulation of the SVM for controlling its sparseness. The aim has been to minimize β and $\mathbf{Z}$ in addition to the primal variables in the SVM formulation. In Keerthi's work [10] basis vectors are obtained by solving the

primal formulation of SVMs using Newton's gradient descent technique. This is a promising approach giving an order of magnitude reduction in the number of SVs. It uses heuristics in the primal space to build the classifier. Downs *et al.,* [7] have proposed a method for retaining linearly *independent* SVs and thereby reducing the complexity of the classifier. Some of the above methods reduce SVs after the training process of the SVM is completed. These include the reduced set method and its variants [2, 16, 14]. On the other hand, the basis vector method proposed by Keerthi *et al.,* [10], Mingrui *et al.,* [21] aims at pruning SVs during the training process itself.

In practical situations memory can be an important constraint and designing SVMs to address memory requirements is recognized as an important research problem. Reducing the number of SVs is a potential solution for addressing memory constraints. Some of the techniques mentioned above can be thought of as a "budgetary constraint" on the number of SVs. The linear growth of the number of SVs in proportion to the number of training data points may be unaffordable in practical situations. A few applications where this budgetary constraint comes into play, as pointed out by Ofer Dekel *et al.,* [6], include: (a) Speech recognition systems. In these, the classifier needs to keep up with the rate at which the speech signal is acquired. (b) A classifier that is designed to run on mobile phones needs to consume very little space and should be very fast with respect to classification time.

The SVM formulation does not take into account memory limits as a parameter at training time. Some of the classifier complexity reduction methods discussed above do take this sort of constraint into account to obtain a decision function that incurs the minimum increase in the generalization error compared to original decision function, whilst meeting the specified limits. A recent effort by Ofer Dekel *et al.* [6] addresses this issue of incorporating a budget parameter into SVMs at training time and has given an algorithm for the same.

1.2 Online Learning

In online kernel-based learning algorithms, typically one data point at a time is considered for updating the parameters involved in specifying the optimal hyperplane. Budget requirements become one of the important parameters in online learning. When there is a memory limit on the number of SVs that can be stored, one has to choose from amongst the available set of SVs, the 'good' SVs that 'best' describe the decision surface. The aim is to find an approximate decision surface which is as close as possible to the decision surface obtained when all of the SVs are used. Discarding SVs involves mainly two steps: (1) deciding which SV should be discarded; and (2) updating the Lagrangian

multipliers for the retained points. The first step is crucial as discarding SVs directly affects the decision function and hence the generalization error.

Koby *et al.* [5] have proposed a method for online classification on a budget for the perceptron problem. The budgetary constraint is taken into account via a two step process. If the t^{th} test data point is misclassified then in the first step the misclassified data point is added to the set of SVs and the weight vector is updated as $\mathbf{w}_t = \mathbf{w}_{t-1} + y_t \alpha_t \mathbf{X}_t$; where $\mathbf{w}_t$ is the new hyperplane and $\mathbf{w}_{t-1}$ is the hyperplane using $(t-1)$ data points. In second step, an SV is found from the set of SVs such that removing it yields the maximum margin, i.e., compute $i = \arg\max_{j \in C_t} \{y_j (\mathbf{w}_{t-1} - \alpha_j y_j \mathbf{X}_j)\}$. The i^{th} SV is discarded from the set thus maintaining the adherence to the memory limit. This approach is implemented using an online algorithm: the margin infused relaxed algorithm (MIRA) [4].

In [12], a *truncation* method was proposed: once the memory has filled up, the oldest SVs are knocked out to make way for new ones. A bound on the increase in the generalization error (as compared to the infinite memory case) was also given in the context of truncation. The intuition behind discarding old SVs is that as we see more data points, the quality of the decision function improves and old SVs contribute little to the quality this decision function; they can thus potentially be discarded [5]. However, this idea simply amounts to maintaining a queue of support vectors.

In this paper, we look to address the constrained memory problem based on a concept known as support vector span. The rationale for using the span is that it directly influences the generalization error bound. Our experiments show that often it is possible to reduce classifier complexity substantially without significant loss in performance.

The rest of this paper is organized as follows: Section 2 looks at the concept of support vector span, which allows us to bound the generalization error. In Section 3, we describe the online learning framework, and present our proposed span-based algorithm, comparing its performance with a standard method for this setting. We also evaluate the performance of the span-based heuristic by comparing it with other "budgeted classification" methods, and present our experimental results in Section 4. Section 5 summarizes our work and concludes the paper.

2 Span of Support Vectors

In the present work, we address the memory constraint problem in online SVMs using the concept of *span* of support vectors introduced by Vapnik [20].

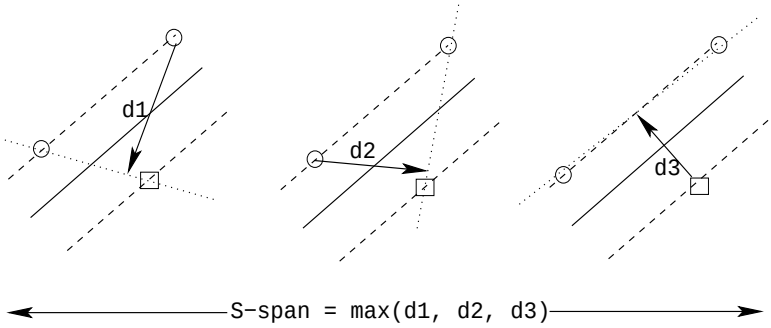


Fig. 1. Geometric interpretation of $S-$span

The span of support vectors is used to bound the estimated generalization error for SVMs. It gives an estimate of the leave-one-out error and has been empirically shown to be tight.

Let $\{(\mathbf{X}_i, y_i)\}_{i=1}^{N_{SV}}$ be the set of SVs. Let the corresponding Lagrange multipliers for the data points be $(\alpha_1^0, \alpha_2^0, \ldots, \alpha_{N_{SV}}^0)$. Define a set Λ_j for a fixed SV $\mathbf{X}_j$ as:

$$\Lambda_j = \left\{ \sum_{i=1,\, i\neq j}^{N_{SV}} \lambda_i \, \mathbf{X}_i : \sum_{i=1,\, i\neq j}^{N_{SV}} \lambda_i = 1, \; \forall \, i \neq j, \;\; \alpha_i^0 + y_i \, y_j \, \alpha_j^0 \, \lambda_i \geq 0 \right\}$$

where $\lambda_i \in \mathbb{R}$. The span of support vector $\mathbf{X}_j$ is denoted by S_j and is given by:

$$S_j^2 = d^2(\mathbf{X}_j, \Lambda_j) = \min_{\mathbf{X}\in\Lambda_j} (\mathbf{X}_j - \mathbf{X})^2$$

The maximal value of S_j over all j is known as S-span and is given by:

$$S = \max \left\{ d\left(\mathbf{X}_1, \Lambda_1\right), d\left(\mathbf{X}_2, \Lambda_2\right), \ldots, d\left(\mathbf{X}_{N_{SV}}, \Lambda_{N_{SV}}\right) \right\} = \max \, S_j$$

The above definition is geometrically illustrated in Figure 1. In this, one SV at a time is removed and a set Λ is constructed using the rest of SVs. Then the minimum distance from the left-out SV to the set Λ is found. This process is repeated for all SVs and maximum of all such distances is said to be the $S-$span.

Using the above definition of $S-$span, the following theorem has been proved:

Theorem 2.1 *Suppose that an SVM separates training data of size N without error. Then the expectation of the probability of error p_{error}^{N-1} for the SVM trained on training data of size $(N-1)$ has the bound:*

$$E p_{error}^{N-1} \leq E\left(\frac{SD}{N\rho^2}\right) \tag{1}$$

where the values of the span of support vectors S, the diameter of the smallest sphere containing the training data points D, and the margin ρ are considered for training sets of size N.

3 Streaming Data and Online Learning Applications

For streaming and online applications the amount of data and consequently the size of the training set keep increasing, and as per [17], the number of SVs will also increase at least proportionately. Clearly, a method is needed to update the classifier continuously. If there is a size limit of N_{SVm} on the SV set of the SVM classifier, then for every subsequent misclassified data point we must decide which one of the existing SVs (if any) will be replaced by the new point.

To formalize, assume we have a memory limit of N_{SVm} SVs. Initially, the memory is empty and training data points keep streaming in. We cannot retrain the SVM from scratch for every new data point since it would be prohibitively expensive to do so. One approach is to use some kind of gradient-descent to adjust the old coefficient (α_i) values based on the extent to which the new point was misclassified (note that points classified correctly can be ignored). Simultaneously, we must also compute the multiplier for the new point. This procedure continues as long as the number of SVs is less than N. Once we have N_{SVm} SVs, we cannot add a point to the set without first discarding an existing SV. We propose to use the S-span to decide which point to remove.

Let X_{old} be the set (of size N_{SVm}) of SVs, S_{old} the corresponding S-span, $\mathbf{X}$ the new misclassified data point and $\mathbf{X}_p \in X_{old}$ the point that leads to the *least* S-span, say S_{new}, when $\mathbf{X}$ replaces $\mathbf{X}_p$ in X_{old} (see algorithm below). We replace $\mathbf{X}_p$ by $\mathbf{X}$ if $S_{new} < S_{old}$. The basic objective is to not increase the bound on the predicted generalization error at each step. From (1), it is clear that the expected value of the generalization error depends on the value of S, the span of the set of SVs. We cannot really control the variation in D and ρ as they will follow the same trend irrespective of which vectors we throw out. However, the variation in span is not bound to follow any such trend and therefore becomes the key determining factor. So, our idea is to try and replace points in the set of SVs so as to maximally reduce the S-span of the resulting set. The simplest replacement strategy would be to leave out the oldest point in the memory each time a new one was to be added; this is essentially what was done in the algorithm proposed by Kivinen et al. [12]. Although this has the advantage of implicitly accounting for time-varying distributions, it also runs the risk of discarding a more informative point in favor of a less useful one. The span-based idea is unlikely to encounter this difficulty. We say unlikely since we do not yet have a proof that the method will always work optimally.

However, our experiments with synthetic and real-life datasets suggest that the approach seems to do well. The algorithm is summarized below:

(1) Let the current set of SVs in the memory be $X_{old} = ((\mathbf{X}_1, y_1), ..., (\mathbf{X}_{N_{SVm}}, y_{N_{SVm}}))$, and let their multipliers be $(\alpha_1, ..., \alpha_{N_{SVm}})$ (we assume the memory is full; otherwise, one can keep adding the incoming SVs until it is full). Let $(\mathbf{x}, y)$ be an incoming data point. We first compute: $f(\mathbf{x}) = \sum_{i=1}^{N_{SVm}} \alpha_i y_i k(\mathbf{X}_i, \mathbf{x})$ Here k is the usual kernel function.
(2) If $f(\mathbf{x}) = y$, we ignore it, return to step 1 and wait for the next point.
(3) Compute S_{old}, the $S-$span of X_{old}. Remove support vector $\mathbf{X}_i$ from X_{old} and include the new data point $\mathbf{x}$. Let the new set of SVs be: $X^i_{new} = \{(\mathbf{X}_1, y_1), \cdots, (\mathbf{X}_{i-1}, y_{i-1}), (\mathbf{X}_{i+1}, y_{i+1}), \cdots, (\mathbf{X}_{N_{SVm}}, y_{N_{SVm}}), (\mathbf{x}, y)\}$. Compute the span corresponding to this set of SVs; call it S_i.
(4) Repeat the above step for all N_{SVm} data points. Find the least $S-$span among the $S_i \; \forall \; i = 1, 2, \cdots, N$. Let this be S_p; the corresponding support vector removed is $\mathbf{X}_p$.
(5) If $S_p < S_{old}$ then we remove $\mathbf{X}_p$ from the set by making $\alpha_p = 0$ while simultaneously adjusting the other coefficients in order to maintain the optimality conditions for the SVM. This is done via the Poggio-Cauwenberghs decremental unlearning procedure [3]. If $S_p \geq S_{old}$ do not include the new point in the SV set.
(6) If $\mathbf{X}_p$ was removed in step 5, add the new point to the SV set, i.e. compute the appropriate multiplier for it. Once again the other α_i values must be adjusted so as to maintain optimality. Use the Poggio-Cauwenberghs incremental learning procedure [3] to carry out this adjustment.
(7) We now have an updated set of N_{SVm} SVs, with the new point possibly replacing one of the earlier ones. Run the current classifier on an independent test set, to gauge its generalization performance. Return to step 1 and await the next point.

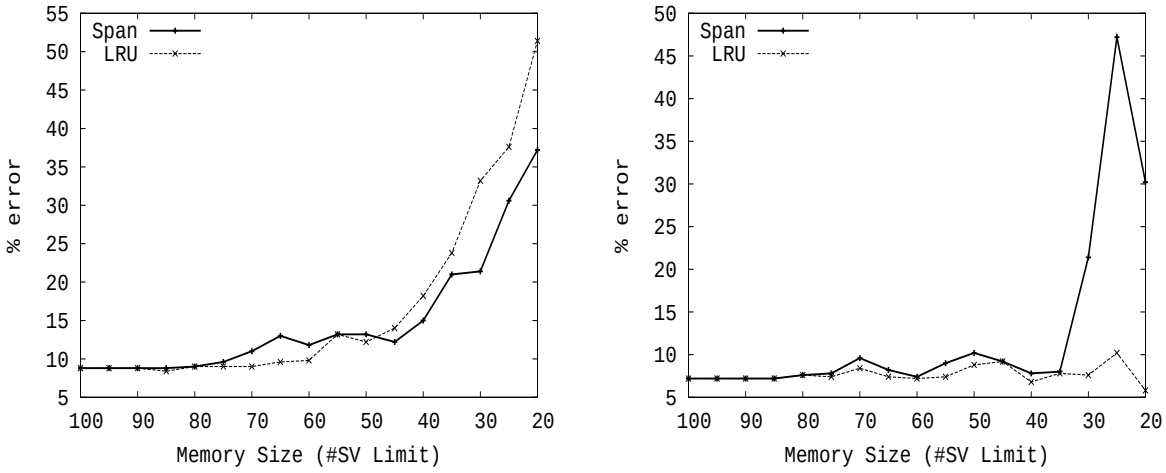
3.1 Baseline: Margin Maximization

Our proposed span-based method is one possible heuristic among several others for achieving the budgetary requirements in online learning situations. Others include: leaving out the oldest SV (which we refer to as the least recently used (LRU) method); or retaining that set of SVs which results in the largest margin, as the key idea in kernel-based methods is to maximize the margin of separation. Margin maximization is a *natural* heuristic for the budget problem and we call this the baseline heuristic. We compare the proposed span-based heuristic with (1) LRU, (2) the baseline heuristic, and (3) the budget classification algorithm proposed by Koby *et al.* [5]. We note that the online learning algorithm employed in [5] is different from the one used in the present work (the incremental-decremental learning algorithm).

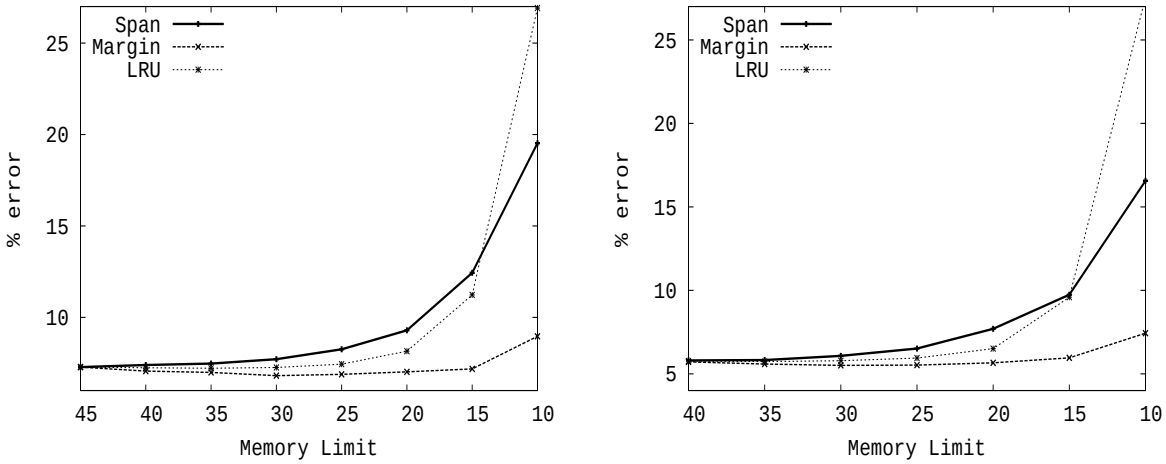
4 Experimental Results

For evaluating the performance of the proposed span-based heuristic for the online budget problem, we consider comparisons with three other methods as described in Section 3.1. We experimented with both static and time-varying distributions of data in the case of artificial datasets. The artificial datasets we used were generated according to multi-variate Gaussian distributions. A total of 500 points were generated for the training dataset and another 500 points for the test dataset. To these datasets, 5% Gaussian noise was added (we call the resulting set g5). For simulating a dataset with a time-varying distribution, the generation procedure was similar, but after every 100 data points the means of the two classes being considered were changed by introducing uniform perturbations (the set thus generated is referred to as chng-g5). For the static distribution case, a memory limit was fixed (smaller than the training set size), and all the training data was streamed in. The final classifier obtained was then tested on an independent test set drawn from the same distribution. This procedure was repeated for varying values of the memory limit. For the time-varying distribution case, the procedure used was similar, except that the training data's distribution varied gradually (changing after every 100 data points) as it was streamed in. The test set was drawn from the current distribution, i.e., the distribution prevailing at the end of the training data stream. In this way, the ability of the algorithm to adapt to the variation could be gauged. We report both preliminary experimental results on `g5, chng-g5` and extensive results averaged over 100 realizations of each dataset. In the case of real-world data, we consider 3 datasets, namely `breast-cancer, thyroid` and `waveform`, each having 100 realizations.

Results for the artificial datasets are shown in Figures 2 (a) to (d). In (a) and (b), we consider one realization each of `g5` and `chng-g5` data sets [1]. It is evident that for all the heuristic approaches, the generalization error increases as the memory limit decreases; in the case of artificial datasets (for both static and time-varying distributions), the margin-based heuristic outperforms the rest of the methods; while span and LRU compete closely. As the memory limit is reduced further, span performs better than LRU. Using the span-based heuristic, a reduction of 44.44% of the original SVs is observed with an increase in generalization error of less than 1.0%. For LRU the comparable reduction in SVs is observed to be 55.55%, and it is 66.66% when using the margin-based heuristic. For the time-varying distribution, the span method is again seem to compete closely with LRU. However, when the amount of deviation from the original distribution increases, then the span method's performance breaks down (this was observed in our experiments for higher amounts of variation in the distribution; however, those results are not included here). This is because the span-based bounds are derived based on the assumption that the data points are drawn from *independent and identical distributions*.



(a) g5: Fixed Distribution (b) chng-g5: Varying Distribution

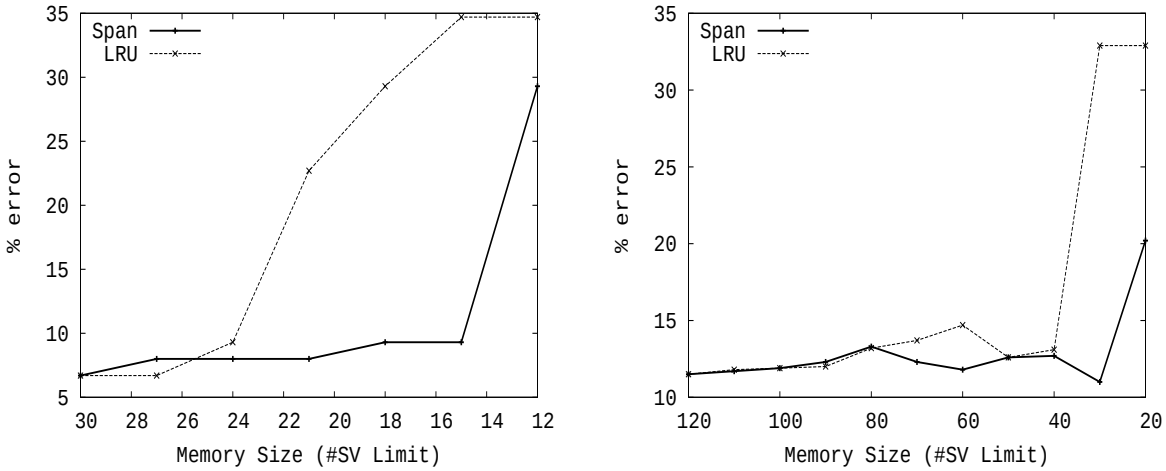


(c) g5: Fixed Distribution (d) chng-g5: Varying Distribution

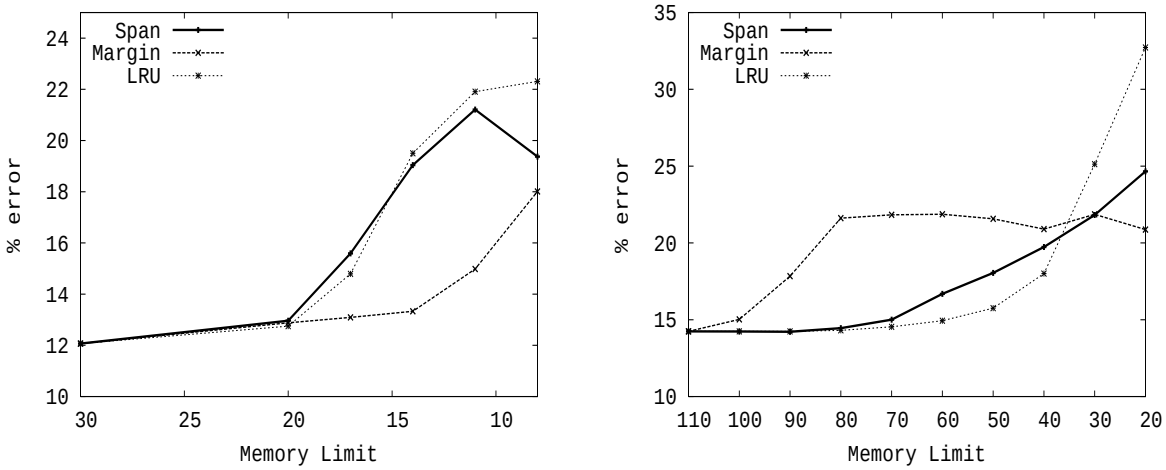
Fig. 2. Comparison of the Span-based method with the LRU and Maximum Margin heuristics. We have generated synthetic two-class data consisting of two overlapping 3-D Gaussian 'bells', with 5% Gaussian noise added in. Preliminary experimental results are reported in (a) and (b) and extensive results averaged over 100 realizations are given in (c) and (d).

In the case of the real-world datasets, the performance of the span-based and LRU methods is better than that of the baseline method in the majority of cases. For the breast-cancer and waveform datasets, the margin-based heuristic performs particularly poorly compared to the others. Results are given in Figures 3 (a) to (d) and 4 (a), (b) for real world data sets. Here too the span-based algorithm is seen to compete closely with LRU.

We compare the proposed span-based method's performance with that of the "classification on a budget" (budget perceptron) method proposed by [5] on



(a) Thyroid (b) Waveform

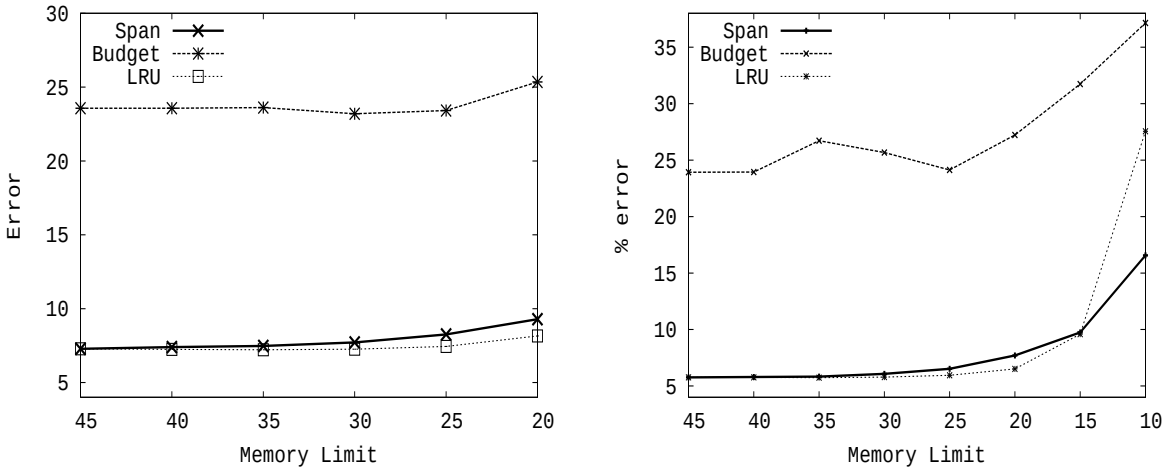


(c) Thyroid (d) Waveform

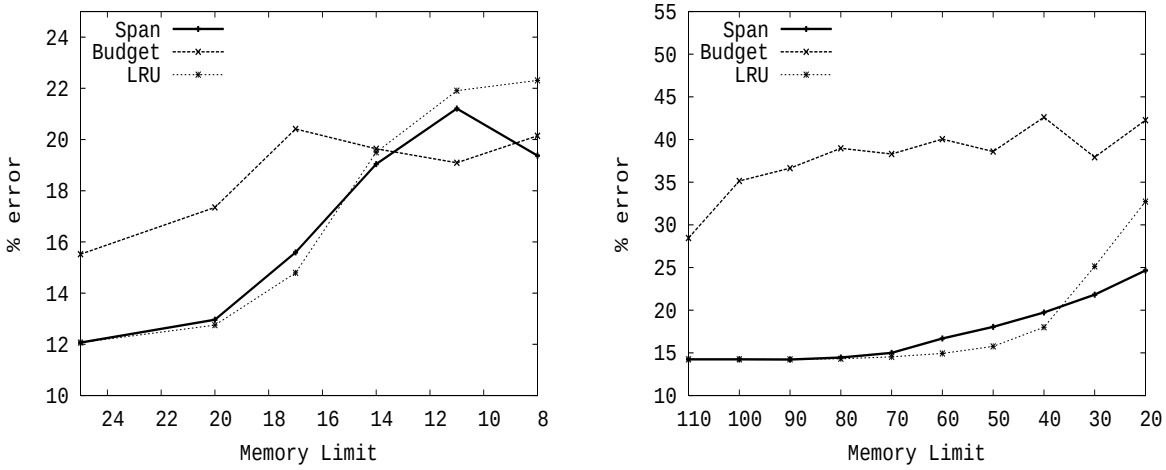
Fig. 3. Comparison between the Span, LRU and Margin heuristics. Results on benchmark datasets: waveform (training size: 400, test size: 4600) and thyroid (training size: 140, test size: 75). (a) and (b) give preliminary results using only one realization; (c) and (d) show extensive experimental results averaged over 100 realizations.

both artificial and real-world datasets [3]. The comparison is not a fair one because we use the incremental-decremental SVM (with minor variations so as to take into account the online setting) whereas Koby *et al.* [5] use the MIRA algorithm for online learning. Hence a comparison based on absolute percentages is not fair; nevertheless, a comparison of the trends of the graphs

[3] incremental-decremental learning algorithm is obtained from author's web page `http://bach.ece.jhu.edu/pub.gert/svm/incremental`; budget perceptron algorithm is obtained from `The Spider` general purpose machine learning toolbox `http://www.kyb.tuebingen.mpg.de/bs/people/spider/`



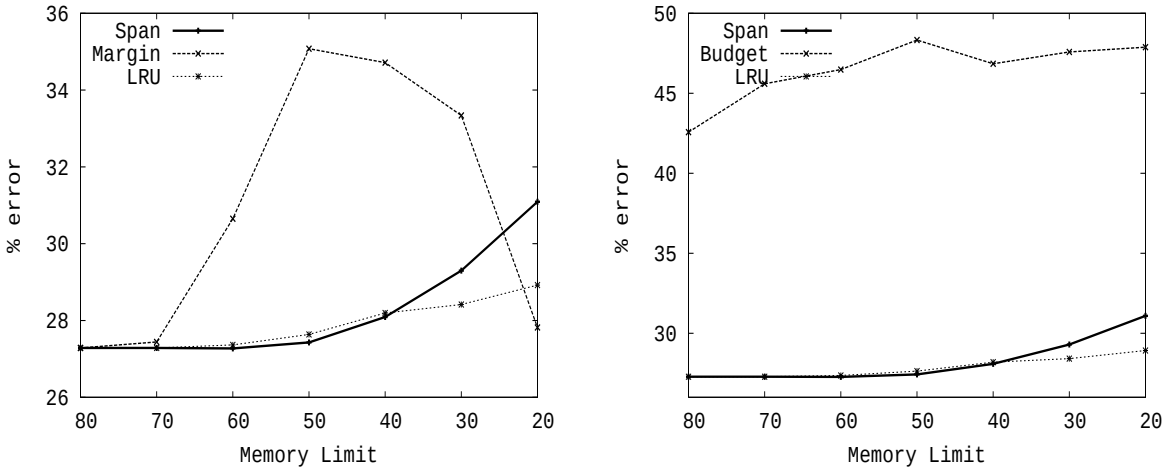
(a) g5: Fixed Distribution (b) chng-g5: Varying Distribution



(c) Thyroid (d) Waveform

Fig. 4. Comparison between Span, LRU and Budget perceptron. Results on artificial datasets: g5 and chng-g5; and benchmark datasets: thyroid and waveform.

gives us an intuitive idea of how well both methods are performing under varying memory constraints. Results are presented in Figures 4 (a) to (d) and 5 (a) and (b). On the whole, our proposed method gives encouraging results in an online setting. It is not as fast as the simple LRU approach; but for small memory sizes, it competes very well in terms of actual time taken. Also, we see that when the memory limit is very low, the span-based heretic's performance is considerably better than that of the LRU method (Figures 2, 3 and 4).



(a) Margin heuristic (b) Budget perceptron

Fig. 5. Comparisons of Span and LRU with: (a) Margin heuristic and (b) Budget perceptron, using the benchmark dataset breast-cancer (training size: 200, test size: 77).

5 Conclusions and Future Work

This work proposes a new learning procedure for a kernel-based classifier for streaming, online data where the maximum size of the support vector set is fixed. We use the concept of the span of the set of SVs to decide which support vector to replace from the current set when a new SV is to be added. While we do not have a formal theoretical justification for the procedure, the experimental results are quite encouraging. Establishing performance guarantees in the form of error bounds remains the major outstanding task.

References

[1] Sumeet Agarwal, V. Vijaya Saradhi, and Harish Karnick. Kernel-based online machine learning and support vector reduction. In *The* 15^{th} *European Symposium on Artificial Neural Networks*, 2007.

[2] C. J. C. Burges. Simplified support vector decision rules. In 13^{th} *International Conference on Machine Learning*, pages 71 – 77, 1996.

[3] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Neural Information Processing Systems*, 2000.

[4] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951 – 991, 2003.

[5] Koby Crammer, Jaz Kandola, and Yoram Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems*, 2003.

[6] Ofer Dekel and Yoram Singer. Support vector machines on a budget. In *Advances in Neural Information Processing Systems*, 2006.

[7] Tom Downs, Kevin E. Gates, and Annette Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

[8] Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243 – 264, 2001.

[9] Chih-Wei Hsu and Chih-Jen Lin. A simple decomposition method for support vector machines. *Machine Learning*, 46:291 – 314, 2002.

[10] S. Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 8:1 – 22, 2006.

[11] S. Sathiya Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124 – 136, Jan 2000.

[12] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, August, 2004.

[13] Y. J. Lee and O. L. Mangasarian. Rsvm: reduced support vector machines. In *CD Proceedings of the first SIAM International Conference on Data Mining, Chicago*, 2001.

[14] DucDung Nguyen and TuBao Ho. An efficient method for simplifying support vector machines. In 22^{nd} *International Conference on Machine Learning*, pages 617 – 624, Bonn, Germany, 2005.

[15] J. C. Platt. *Advances in Kernel Methods - Support Vector Learning*, chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization. Cambridge, MA, MIT Press, 1998.

[16] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, USA, 2002.

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

[18] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211 – 214, 2001.

[19] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[20] V. Vapnik and O Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013 – 2036, 2000.

[21] Mingrui Wu, Bernhard Scholkopf, and Gokhan Bakir. Building sparse large margin classifiers. In 22^{nd} *International Conference on Machine Learning*, Bonn, Germany, 2005.