

Fast Support Vector Machines for Continuous Data

Kurt A. Kramer,
Lawrence O. Hall,
Dmitry B. Goldgof

Computer Science and Engineering

ENB118, 4202 E. Fowler Ave.

University of South Florida

Tampa, FL 33620

email:(hall,goldgof,kkramer)@cse.usf.edu

Andrew Remsen

College of Marine Science

University of South Florida

St. Petersburg, Fl.

email: aremsen@marine.usf.edu

Tong Luo

Searchspark.

email: luotong@gmail.com

December 16, 2008

Abstract

Support vector machines can be trained to be very accurate classifiers and have been used in many applications. However, the training and to a lesser extent prediction time of support vector machines on very large data sets can be very long. This paper presents a fast compression method to scale up support vector machines to large data sets. A simple bit reduction method is applied to reduce the cardinality of the data by weighting representative examples. We then develop support vector machines trained on the weighted data. Experiments indicate that the bit reduction support vector machine produces a significant reduction in the time required for both training and prediction with minimum loss in accuracy. It is also shown to, typically, be more accurate than random sampling when the data are not over-compressed.

1 Introduction

Support vector machines (SVMs) achieve high accuracy in many application domains, such as character [35] and face recognition [40], as well as our work [23][22] in recognizing

underwater zooplankton. However, scaling up SVMs to very large data sets is still an open problem. Training an SVM requires solving a constrained quadratic programming problem, which usually takes $O(m^3)$ computations where m is the number of examples. Predicting a new example involves $O(sv)$ computations where sv is the number of support vectors and is usually proportional to m . As a consequence, SVMs' training time and prediction time to a lesser extent on a very large data set can be quite long, thus making it impractical for some real-world applications. In plankton recognition, retraining is often required as new plankton images are labeled by marine scientists and added to the training library on the ship. As we acquire a large number of plankton images, training an SVM with all labeled images becomes extremely slow.

In this paper, we propose a simple strategy to speed up the training and prediction procedures for a SVM: bit reduction. Bit reduction reduces the resolution of the input data and groups similar data into one bin. A weight is assigned to each bin according to the number of examples from a particular class in it and a weighted example is created. This data reduction and aggregation step is very fast and scales linearly with respect to the number of examples. Then an SVM is built on a set of weighted examples which are the exemplars of their respective bins. Optimal compression parameters need only be computed once and can be re-used if data arrives incrementally as in plankton recognition or an online learning application or image labeling or any domain in which data will be incrementally labeled.

This approach allows us to investigate the use of SVM's on very large data sets where it has previously been impractical. We know for smaller data sets a support vector machine is often significantly more accurate than a decision tree or naive Bayes classifier, which scale much better.

This approach can be applied to any learning algorithm which takes weighted examples, though without a reliance on support points its utility is unclear. We are only concerned with support vector machines because they are much more time intensive with much smaller data sets than other learning algorithms such as decision trees. In fact, decision trees train fast enough that they would not need to use data compression for most of the data sets in our experiments. Our experiments indicate that bit reduction SVM (BRSVM) significantly reduces the training time and prediction time with a minimal loss in accuracy. It results in more accurate classifiers than random sampling on most data sets when the data are not over-compressed.

2 Previous work

There are two main approaches to speed up the training of SVMs. One approach is to find a fast algorithm to solve the quadratic programming (QP) problem for a SVM. “Chunking”, introduced in [42], solves a QP problem on a subset of data. Chunking only keeps the support vectors chosen on the subset and replaces others with data that violate the Karush-Kuhn-Tucker (KKT) conditions. Using an idea similar to chunking, decomposition [18] puts a subset of data into a “working set”, and solves the QP problem by optimizing the coefficients of the data in the working set while keeping the other coefficients unchanged. In this way, a large QP problem is decomposed into a series of small QP problems, thus making it possible to train an SVM on large scale problems. Sequential minimum optimization (SMO) [33] and its enhanced versions [19] [12] take decomposition to the extreme: Each working set only has two examples and their optimal coefficients can be solved for analytically. SMO is easy to implement and does not need any third-party QP solvers. SMO is widely used to train SVMs. Another way of solving large scale QP problems [16][44] is to use a low-rank matrix to approximate the Gram matrix of an SVM. As a consequence, the QP optimization on the small matrix requires significantly less time than on the whole Gram matrix. In [41] a Core Vector Machine that implements an approximation algorithm using minimum enclosing balls was proposed. In [2], a multi-class core vector machine was introduced which scales better with data set size and many classes. In [25] a Lagrangian Support vector machine was introduced which produced speed-ups in training with comparable test set accuracy.

In [39], a priori knowledge is incorporated in the training of a least-squares support vector machine. Our approach doesn’t use knowledge about the training set, but a more general data compression approach.

The other main approach to speeding up SVM training comes from the idea of “data squashing”, which was proposed in [13] as a general method to scale up data mining algorithms. Data squashing divides massive data into a limited number of bins. The statistics of the examples from each bin are computed. A model is fit by only using the statistics instead of all examples within a bin. The reduced training set results in significantly less training time. Several clustering algorithms [46][38][3] were used to partition data and build an SVM based on the statistics from each cluster. In [3], the SVM model built on the reduced set was used to predict on all of the training data. Examples falling in the margin or being misclassified were taken from their original clusters and added back into the training data for retraining. However, both [46] and [38] assumed a linear kernel which

might not generalize well to other kernels. In [3], two experiments were done with a linear kernel and only one experiment used a third-order polynomial kernel. Moreover, it is not unusual for many examples to fall into the margin of an SVM model especially for an RBF kernel. In such cases, retraining with all examples within the margin is computationally expensive.

In [20] a method of compressing data for SVMs was introduced. It involves a time-consuming clustering approach. On one small data set they found higher accuracy at the cost of longer training times, but shorter testing times. Given that it increases the training time, it does not appear to be scalable for large data sets. In [1] the authors used clustering to reduce the size of the training set. On just two data sets they showed comparable accuracy to using all the data with speedups in training of 38.7 times and 15.6 times. It is not clear that the pre-processing clustering times were included in determining speedups.

Most work [4][5][28][36] on enabling fast prediction with SVMs has focused on the problem of reducing the number of support vectors (SVs) obtained. Since the prediction time of an SVM depends on the number of support vectors, a search was done for a reduced set of vectors which can approximate the decision boundary. The prediction using the reduced set was faster than using all support vectors. However, reduced set methods involve searching for a set of pre-images [36][37], which is a set of constructed examples used to approximate the solution of a SVM. It should be noted that the searching procedure is computationally expensive.

In [17] two separate decision boundaries were created one for each class rather than a single one. For each decision boundary only the examples for its respective class were used thus reducing by about half the number of examples used for finding the decision boundary. The solvers are polynomial complexity and thus the training time for each solver individually is reduced by more than 50%. The experiments performed in the paper used relatively small data sets, the largest being 1473 examples.

Data squashing approaches seem promising and can be combined with fast QP approaches like SMO etc. for fast training and prediction. However, most work [38][3][46] in data squashing+SVM requires clustering the data and/or linear kernels [38][46][30]. Clustering usually needs $O(m^2)$ computations and high-order kernels, like the RBF kernel, are widely used and essential to many successful applications. Therefore, a fast squashing method and experiments on high-order kernels are necessary to apply data squashing+SVMs to real-world applications. In this paper, we propose a simple and fast data compression method: bit-reduction SVM (BRSVM). It does not require any computa-

tionally expensive clustering algorithms and works well with RBF kernels as shown in our experiments. This work is an expansion of that in [21] with more data sets used, significantly more detailed experimental analysis and a guide to parameter setting.

3 Bit reduction SVM

Bit reduction SVM (BRSVM) works by reducing the resolution of examples and representing similar examples as a single weighted example. In this way, the data size is reduced and training time is saved. It is simple and much faster than clustering, for instance. Another even simpler data reduction method is random sampling. Random sampling subsamples data without replacement. Compared to weighted examples, random sampling suffers from high variance of estimation in theory [8]. In spite of its high variance, random sampling has been shown to work very well in experiments [29][38]: It was as accurate as or slightly less accurate than more complicated methods.

3.1 Bit reduction

Bit reduction is a technique to reduce the data resolution. It was used to build a bit reduction fuzzy c-means (BRFCM) method [15], which applied bit reduction to speed up the fuzzy c-means (FCM) clustering algorithm. However, classification problems differ in that only examples from the same class should be aggregated together.

There are three steps involved in bit reduction for an SVM: normalization, bit reduction and aggregation.

1. Normalization is used to ensure equal resolution for each feature. To avoid losing too much information during quantization, an integer is used to represent each normalized feature value. The integer $I(v)$ for a floating point value v is constructed as follows:

$$I(v) = \text{int}(Z * v)$$

where Z is a domain dependent number used to scale v and the function $\text{int}(k)$ returns the integer part of k . In this way, the true value of v is kept and only $I(v)$ is used in bit reduction. In our experiments, we used $Z = 1000$. It is important to note that all feature data prior to the bit reduction process described here goes through a z-normalization such that each feature will have a mean centered on zero and unit variance. Unless you wish different features to have different levels of significance it is important that they are all treated the same.

2. Bit reduction is performed on the integer $I(v)$. Given b , the number of bits to be reduced, $I(v)$ is right-shifted and its precision is reduced. We slightly abuse notation here by letting the $I(v)$ in the right hand side of Eq. (1) be the $I(v)$ before bit reduction and $I(v)$ in the left hand side be the $I(v)$ after bit reduction.

$$I(v) \leftarrow I(v) \gg b \tag{1}$$

where $k \gg b$ shifts the integer k to the right by b bits. Given an r -dimensional example $x_i=(x_{i1},x_{i2},\dots,x_{ir})$, its integer expression after bit reduction is $(I(x_{i1}),I(x_{i2}),\dots,I(x_{ir}))$.

3. The aggregation step groups the examples from the same class whose integer expressions fall into the same bin. For each class, its mean within the bin is calculated separately from only the examples with that class label. The weight of the representative mean equals the number of examples from that class. During the mean computation, the real values $(x_{i1}, x_{i2}, \dots, x_{ir})$ are used.

Note the bit reduction procedure reduces data precision. A very large b results in too many examples falling in the same bin. The mean statistic is not enough to capture the location information of many examples. A small b does not provide enough data reduction, thus still leaving training slow. The best number of bits reduced (b) varies for different data sets. It can be found by search on a small subset of data as described later. The optimal number b for bit reduction will be used for retraining on the same type of data.

During bit reduction, it is very likely that some bins have examples from many different classes. Therefore, in the aggregation step, the mean statistic of examples in a bin was computed individually for each class based only on the examples in that class. This alleviates the side effect of grouping examples from different classes into the same bin. As a result, one bin may contain weighted examples for multiple classes.

Table 1 describes the bit reduction procedure for four 1-d examples with class label y_i .

The four examples from two classes are first scaled to integer values by using $Z = 1000$. Then 2-bit reduction is performed by right shifting its integer expression by 2 bits. All four examples end up having the same value, which means all four examples fall into one bin after a 2-bit reduction. Table 2 shows the weighted examples, one for each class, after the aggregation step.

Since all four examples are in the same bin, we aggregate them by class and compute their mean for each class using the original values x_i . The weight is computed by simply

Table 1: A 1-d example of bit reduction in BRSVM

i	Example (x_i, y_i)	$I(x_i)$ and its bit expression $Z = 1000$	$I(x_i)$ after 2-bit reduction
1	(0.008, 1)	8 (1000)	2 (10)
2	(0.009, 1)	9 (1001)	2 (10)
3	(0.010, 2)	10 (1010)	2 (10)
4	(0.011, 2)	11 (1011)	2 (10)

Table 2: Weighted examples after the aggregation step.

i	New examples (x_i, y_i)	Weight
1	(0.0085, 1)	2
2	(0.0105, 2)	2

counting the number of examples from the same class.

Although bit reduction is fast, a sloppy implementation of aggregation may easily cost $O(m^2)$ computations where m is the number of examples. We implemented a hash table for the aggregation step as done in [15]. Universal hashing [10] was used as the hash function. Collisions were resolved by chaining. When inserting the bit-reduced integer values into the hash table, we used a list to record the places that were filled in the hash table. The mean statistics were computed by re-visiting all the filled places in the hash table. The average computational complexity for our implementation is $O(2m)$. Please see [10] for more information about the universal hashing function.

3.1.1 Unbalanced bit reduction

The simplest way to apply bit reduction is equally to all attributes. However, at some point in the process, say after a bits, there may occur a large compression (say from 1000 examples to 10). At that point, the data is typically over compressed and accuracy will be poor. We would like to have the option of compressing the training data *a little* more from $a - 1$ bits. To do this we will apply bit reduction to only a subset of the attributes.

The attributes will be ordered from lowest to highest variance and will be reduced in order by 1 more bit. We will denote the number of attributes processed in the unbalanced way by \mathbf{s} which can vary between zero and one less than the number of attributes in the data set.

3.2 Weighted SVM

It is important to weight aggregated examples by the number of source examples. Bins that have a greater number of examples should have a greater impact on the decision boundary. This is what would happen if bit reduction does not take place. Consider if there are 10 examples that fall into the same bucket and one noisy example (misclassified) in its own bin. Without weighting the noisy example would have as much impact on the decision boundary as the 10 examples that fall into the same bucket.

Pavlov et al. [30] proposed a method to train a weighted SVM, although its description in [30] is concise and lacks significant details. Following their work, we describe how to train a weighted SVM in more detail in this subsection.

Given examples x_1, x_2, \dots, x_m with class label $y_i \in \{-1, 1\}$, an SVM solves the following problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \langle w, w \rangle + \frac{C}{m} \sum_{i=1}^m \xi_i & (2) \\ \text{subject to:} \quad & y_i (\langle w, \phi(x_i) \rangle + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

where w is normal to the decision boundary (a hyperplane), C is the regularization constant that controls the trade-off between the empirical loss and the margin width, the slack variable ξ_i represents the empirical loss associated with x_i . In the case of weighted examples, the empirical loss of x_i with a weight β_i is simply $\beta_i \xi_i$. Intuitively, it could be interpreted as β_i identical examples x_i . Accumulating the loss of the β_i examples results in a loss of $\beta_i \xi_i$. Substitute ξ_i with $\beta_i \xi_i$ in Eq. (2), and we derive the primal problem of a weighted SVM:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \langle w, w \rangle + \frac{C}{m} \sum_{i=1}^m \beta_i \xi_i & (3) \\ \text{subject to:} \quad & y_i (\langle w, \phi(x_i) \rangle + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, \dots, m \end{aligned}$$

The constraint in Eq. (3) remains unchanged because the constraint for each of the β_i examples x_i is identical. The β_i identical constraint formulas can be reduced to one constraint as shown in Eq. (3).

The dual form of a weighted SVM is as follows.

$$\begin{aligned}
 & \text{maximize} && \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j) && (4) \\
 & \text{subject to} && 0 \leq \alpha_i \leq \frac{C\beta_i}{m}, i = 1, \dots, m \\
 & && \sum_{i=1}^m \alpha_i y_i = 0
 \end{aligned}$$

The dual form of a weighted SVM is almost identical to a normal SVM except for the boundary condition of $\alpha_i \leq \frac{C\beta_i}{m}$ while in a normal SVM $\alpha_i \leq \frac{C}{m}$. Therefore, efficient solvers for an SVM such as SMO [33] can be used to solve a weighted SVM by modifying the boundary condition slightly. We note that this formulation is equivalent to having multiple examples that are exactly the same which won't affect the convergence properties.

4 Experiments

We discuss in detail experiments with BRSVM on twelve data sets including banana [34], phoneme [14], shuttle [27], page [26], pendigit [26], letter [26], SIPPER II plankton images [22], waveform [26] and satimage [26]. They come from several sources ranging in size from 5000 to 58,000 examples and from 2 to 36 attributes. We also ran experiments on the Adult, Forest, and Web data sets to compare with previous work as detailed in subsection 4.3. The details of all the data sets are summarized in Table 3.

Table 3: Description of the data sets.

data set	# of data	# of attributes	# of classes
Adult	45222	14	2
banana	5300	2	2
Forest	495141	54	2
phoneme	5404	5	2
shuttle	58000	9	7
page	5473	10	5
pendigit	10992	16	10
letter	20000	16	26
plankton	8440	17	5
waveform	5000	40	3
satimage	6435	36	6
Web	36818	293	2

The Libsvm tool [7] for training support vector machines was modified and used in all experiments. For four of the data sets the kernel, the γ and regularization constant (C)

parameters were the same as used in the original source papers. For the Adult, Forest, and Web datasets the linear kernel was used with C set to 1 as in [30]. The plankton dataset used the RBF kernel ($k(x, y) = \exp(-\gamma\|x - y\|^2)$) with $C = 16$ and $\gamma = 0.04096$ as in [22]. For the remaining 8 datasets the RBF kernel was used with the kernel parameters γ and C being tuned by a two pass grid search, coarse followed by fine. In the first pass γ started at 0.00001 and was incremented by multiples of 2 and in the second by multiples of 1.1. In both passes C goes from 1 to 18 by increments of 1. A 5-fold cross validation on the training data was used to evaluate each γ and C parameter pair. The starting and ending γ for the second pass was derived from the smallest and largest γ chosen from the 2% of parameter pairs (C and γ) that resulted in the most accurate classifiers in the first pass. The starting γ will then be reduced to the smallest nearest power of 10 (e.g. 0.16384 would reduce to 0.10000).

We used the same training and test data separation as given by the original users of the data sets. For those data sets which do not have a separate test set, we randomly selected 80% of the examples as the training set and 20% of the examples as the test set. Since the nine data sets without given train/test splits have more than 5000 examples, 20% of the total data will have more than 1000 examples. We believe this provided a relatively stable estimation. We built SVMs on the training set with the optimal parameters and report the accuracy on the test set. Z-Normalization with mean centered on zero and unit variance was applied to all data sets. Table 4 gives the specific parameter settings used for each data set. All our experiments were done on a SUN-Fire-880 using a single SUN Ultra-sparc III processor at 750mHz with 32 GB memory under the SunOS 5.8 operating system.

4.1 Experiments with pure bit reduction

The experimental description starts with pure or balanced bit reduction in which all attributes are compressed by the same number of bits.

For brevity, detailed experimental results using BRSVM are examined on three representative data sets, banana, plankton and phoneme, as shown in Tables 5–7 and Figures 1–3. Results from the other data sets will be summarized. The last row of each table records the result of an SVM trained on the uncompressed data set. The other rows present the results from BRSVM. The first column is the number of bits reduced. The second column is the compression ratio, which is defined as $\frac{\# \text{ of examples after bit reduction}}{\# \text{ of examples}}$. We start off with 0-bit reduction which may not correspond to a 1.0 compression ratio.

Table 4: SVM Parameters Used.

Data Set	Kernel	C	(γ) for RBF Kernel
Adult	Linear	1	–
Banana	RBF	1	3.4004
Forest	Linear	1	–
Letter	RBF	6	0.079578
Page	RBF	1	0.08955
Pendigit	RBF	5	0.08955
Phoneme	RBF	13	1.48005
Plankton	RBF	16	0.04096
SatImage	RBF	9	0.14202
Shuttle	RBF	16	0.274727
Waveform	RBF	1	0.008138
Web	Linear	1	–

The reason is that repeated examples are grouped together even when no bit is reduced. This results in compression ratios less than 1.0 at 0-bit reduction in some cases. The third column is the accuracy of BRSVM on the test set. McNemar’s test [11] is used to check whether BRSVM accuracy is statistically significantly different from the accuracy of an SVM built on the uncompressed data set. An accuracy percentage in bold indicates the difference is not statistically significant at the $p = 0.05$ level. The fourth column represents the number of support vectors found during the training process. The fifth column is the time for bit reduction which includes the time required to do example aggregation. The sixth column is the BRSVM training time. The seventh column is the prediction time on the test set. All of the timing results were recorded in seconds. The precision of the timing measurement was 0.01 seconds. The training and prediction speedup ratio are defined as $\frac{\text{SVM training time}}{\text{BRSVM training time}}$ and $\frac{\text{SVM prediction time}}{\text{BRSVM prediction time}}$, respectively. For random sampling the sub-sampling ratio was set to equal the compression ratio of BRSVM. Since random sampling has a random factor, we did 50 experiments for each sub-sampling ratio and report the average accuracy and average number of support vectors (SVs). These two values are listed in the last two columns of Tables 5–7 titled random sampling and random SVs.

Figures 1–3 show the effect on classification accuracy and the number of support vectors as bit reduction goes from none to 12. The x-axis indicates the bit reduction level, the left y axis indicates classification accuracy while the right y-axis indicates the number of support vectors.

The experimental results on the banana data set are shown in Table 5 and Figure 1. As more bits were reduced, fewer examples were used in training. Thus training time was

Table 5: BRSVM on the banana data set. The accuracy in bold indicates not statistically significantly different from the accuracy of SVM.

bit reduction	compression ratio	BRSVM accuracy	support vectors	BRSVM reduction time	BRSVM training time	BRSVM prediction time	random sampling	random SVs
0	1.000	90.57%	1055	0.03s	4.47s	0.49s	90.57%	1055.0
1	0.999	90.57%	1055	0.03s	4.45s	0.50s	90.56%	1055.2
2	0.997	90.57%	1055	0.03s	4.48s	0.49s	90.58%	1052.8
3	0.988	90.57%	1052	0.03s	4.48s	0.49s	90.57%	1045.5
4	0.960	90.57%	1034	0.02s	5.21s	0.48s	90.65%	1015.5
5	0.846	90.57%	976	0.03s	3.87s	0.45s	90.70%	905.8
6	0.575	90.57%	775	0.02s	1.77s	0.36s	90.67%	641.7
7	0.244	90.66%	433	0.02s	0.38s	0.20s	90.50%	315.1
8	0.077	90.28%	174	0.01s	0.07s	0.08s	89.71%	132.5
9	0.024	86.23%	72	0.01s	0.01s	0.04s	87.07%	61.3
10	0.007	76.32%	30	0.02s	0.00s	0.03s	77.10%	25.6
SVM	1.000	90.57%	1055	0.00s	4.61s	0.50s		

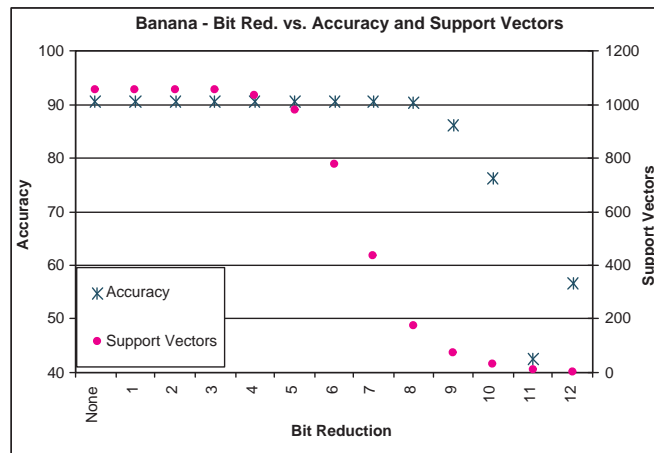


Figure 1: Reduction vs. Accuracy and Support Vectors for Banana dataset.

reduced. Also, less training data results in a classifier with fewer support vectors. The prediction time is proportional to the total number of support vectors. Therefore, the prediction time of BRSVM was reduced accordingly. When 8 bits were reduced, BRSVM was 57 times faster during training and 6 times faster during prediction than a normal SVM. Its accuracy was not statistically significantly different from an SVM built on all the data at the $p = 0.05$ level. Figure 1 provides a good visual picture of what happens as the data is compressed. Starting at a bit reduction level of 5 the number of support vectors starts to decline while the classification accuracy held until a bit reduction level of 9 was reached.

Phoneme is another relatively low-dimensional data set with five attributes. Table 6 and Figure 2 present the experimental results of BRSVM on this data set. When 8 bits

Table 6: BRSVM on the phoneme data set. The accuracy in bold indicates not statistically significantly different from the accuracy of SVM.

bit reduction	compression ratio	BRSVM accuracy	Support vectors	BRSVM reduction time	BRSVM training time	BRSVM prediction time	random sampling	rand SVs
0	0.993	90.93%	1353	0.03s	13.69s	0.83s	90.73%	1345.7
1	0.988	90.93%	1353	0.03s	13.63s	0.83s	90.66%	1337.0
2	0.979	90.93%	1343	0.03s	13.21s	0.82s	90.57%	1325.1
3	0.973	90.93%	1336	0.03s	13.10s	0.82s	90.52%	1317.4
4	0.969	90.93%	1336	0.03s	13.17s	0.82s	90.50%	1313.4
5	0.965	90.93%	1332	0.03s	12.94s	0.81s	90.46%	1307.8
6	0.955	90.93%	1327	0.03s	13.22s	0.82s	90.39%	1296.6
7	0.899	90.84%	1292	0.03s	11.71s	0.82s	90.11%	1233.4
8	0.686	90.93%	1164	0.03s	7.47s	0.74s	89.26%	995.2
9	0.306	86.68%	745	0.02s	1.98s	0.46s	86.25%	534.1
10	0.060	78.35%	234	0.02s	0.07s	0.15s	80.07%	166.5
SVM	1.000	90.93%	1356	0.00s	13.66s	0.850		

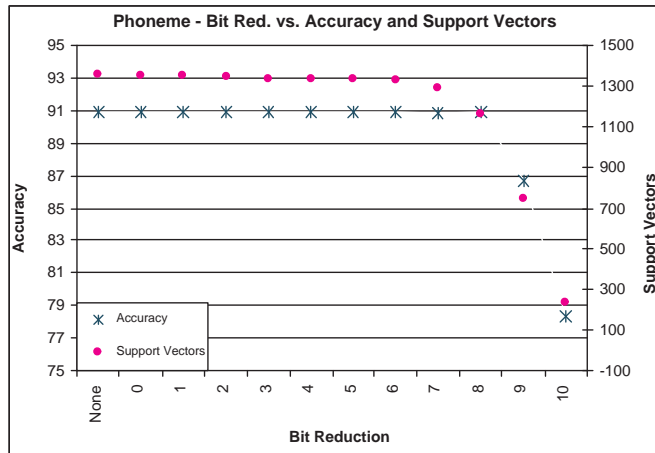


Figure 2: Reduction vs. Accuracy and Support Vectors for the Phoneme dataset.

were reduced, BRSVM was 1.8 times faster during training and 1.1 times faster during prediction than a normal SVM. Its accuracy was not statistically significantly different from an SVM built on all the data at the $p = 0.05$ level. BRSVM was as or more accurate than random sampling when the compression ratio was larger than 0.060. The number of support vectors reduces in a way similar to that seen with the banana data set. In Figure 2 the classification accuracy holds right up to bit reduction level 8 while the number of support vectors starts to drop after a bit reduction level of 6.

Table 7 and Figure 3 show the experimental results on a higher dimensional data set—plankton. BRSVM was slightly more accurate than random sampling when the number of reduced bits is up to 9. At the 10-bit reduction level, the compression ratio of BRSVM drops sharply from 0.961 to 0.353, resulting in a significant loss in accuracy. At the 10

Table 7: BRSVM on the plankton data set. The accuracy in bold indicates not statistically significantly different from accuracy of SVM.

bit reduction	compression ratio	BRSVM accuracy	Support vectors	BRSVM reduction time	BRSVM training time	BRSVM prediction time	random sampling	random SVs
0	0.995	89.60%	2615	0.08s	24.32s	2.94s	89.42%	2623.3
1	0.995	89.60%	2615	0.09s	24.51s	2.96s	89.42%	2623.3
2	0.995	89.60%	2615	0.08s	24.56s	2.96s	89.42%	2623.3
3	0.995	89.60%	2615	0.09s	24.60s	2.91s	89.42%	2623.3
4	0.995	89.60%	2615	0.09s	24.70s	2.95s	89.42%	2623.3
5	0.995	89.60%	2615	0.08s	24.67s	2.85s	89.42%	2623.3
6	0.995	89.60%	2615	0.08s	24.58s	2.97s	89.42%	2623.3
7	0.995	89.60%	2615	0.09s	24.71s	2.89s	89.42%	2623.3
8	0.995	89.60%	2610	0.08s	24.32s	2.84s	89.40%	2622.3
9	0.961	89.30%	2547	0.09s	23.17s	2.88s	89.19%	2547.8
10	0.353	84.50%	928	0.06s	4.00s	1.05s	86.37%	1092.9
11	0.065	68.40%	236	0.05s	0.20s	0.27s	80.35%	278.1
SVM	1.000	89.60%	2628		24.70s	2.96s		

Table 8: Summary of BRSVM on the shuttle, page, letter, and waveform data sets.

Data set	Optimal b	Comp Ratio	SVM Accuracy	BRSVM Accuracy	Train Speedup	Pred Speedup	Random Sampling Accuracy
shuttle	$b = 8$	0.031	99.86%	99.84%	34.7	1.2	99.41%
page	$b = 9$	0.180	95.44%	95.35%	8.6	1.8	93.93%
letter	$b = 9$	0.887	97.53%	97.30%	1.2	1.1	97.22%
waveform	$b = 11$	0.450	86.70%	85.80%	4.2	1.8	86.47%

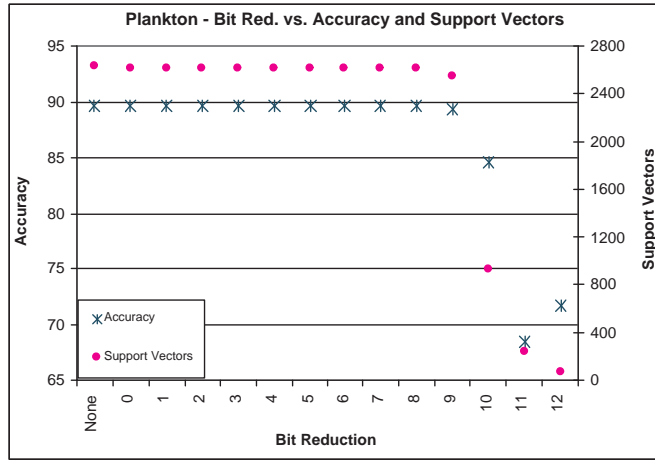


Figure 3: Reduction vs. Accuracy and Support Vectors for Plankton dataset.

and 11 bit reduction levels where the compression ratios were less than or equal to 0.353, the accuracies of BRSVM were much lower than random sampling. This phenomenon was observed on several other data sets in our experiments. The reason is that when the compression ratio is small, it is very likely that many examples from different classes fall into the same bin and the number of examples distribute far from uniformly among different bins. For instance, suppose bit reduction compresses the data into several bins and one bin has 80% of the examples from different classes. BRSVM uses the mean statistic as the representative for each class, which may not be able to capture the information about the decision boundary in its bin. Random sampling, on the other hand, selects the examples more uniformly. If 80% of the examples fall into one bin, random sampling will effectively sample four times more examples that reside in this bin than all others together, and preserve the local information of the decision boundary much better than BRSVM. As a result, random sampling is likely to be as or more accurate than BRSVM when the compression ratio is very low. This tends to happen on high dimensional data sets. On the other hand, at a higher compression ratio, where examples from the same class fall into the same bin and distributions of the number of examples in bins are not very skewed, BRSVM preserves the statistics of all examples while random sampling suffers from high sampling variance. Therefore, BRSVM is more accurate than random sampling when the compression ratio is relatively high.

Results ranging from training speedups of 34.7 times to 1.2 times were observed on shuttle, page, letter, and waveform as summarized in Table 8. The least improvement in

training time came on the 16 feature letter data set.

4.2 Adding unbalanced bit reduction

As shown in Section 4.1 using balanced bit reduction there is a point where there is a large reduction in the number of examples after the reduction from $a - 1$ to a bits (e.g. Tables 5–7). In this section, we explore how to get a smaller reduction in the number of examples and hence higher accuracy, by using unbalanced bit reduction after compression by $a - 1$ bits.

In the experiments shown in Tables 9 and 10 the number of attributes for unbalanced bit reduction, s , was varied from 0 to one less than the number of attributes in the data set. The entry with the largest b and largest number of unbalanced bits (s) that is still not statistically different from the accuracy of the SVM entry as per a McNemar’s test [11] with $p = 0.05$ would be considered the best values for b and s .

We leave to future work the idea of investigating multiple resolution unbalanced bit-reduction. That is different features can be reduced by different levels of bit reduction. Ex: feature 1 by $a + 1$, feature 2 by a , feature 3 by $a + 4$ and so forth. An optimization algorithm would need to be developed to determine the various levels of bit reduction for the different features.

The results in Tables 9 and 10 consist of 9 columns. The first column indicates the number of unbalanced bits(s). The second column indicates the compression achieved. The third column is the classification accuracy. The fourth column is the number of support vectors. The fifth, sixth, and seventh columns are the time in seconds to perform reduction, training and testing respectively. The eighth column shows the average random sampling accuracy on the test set and the ninth is the average number of SVs for random sampling. The sub-sampling ratio is set equal to the compression ratio of BRSVM. The random sampling accuracy is the mean of 50 experiments.

We discuss experiments with UBR on the phoneme, pendigit, plankton, waveform and satimage data sets, for which pure bit reduction did not result in incremental changes in compression ratios. In this paper we define a good compression ratio as the minimum compression ratio where accuracy was not statistically significantly different from that obtained from an SVM trained on the uncompressed data set.

The pure bit reduction experiments on the Plankton data set were recorded in Table 7. After 9 bit reduction, BRSVM results in a 0.961 compression ratio and 1.06 times speedup in the training phase with no loss in accuracy. While after 10 bit reduction, the

compression ratio drops to 0.353 and the corresponding 4.80% accuracy loss could not be tolerated. Table 9 shows the results of applying UBR to search for a compression ratio between 0.961 and 0.353. We expected that this would result in more speedup than the 1.06 times from 9-bit reduction without sacrificing accuracy. Nine bit reduction is first applied to the data and then additional attributes are reduced further by 1 bit, starting with 1 attribute ($s = 1$) to 16 attributes ($s = 16$). At $s = 8$ an approximately 2.0 times speed-up in training time was achieved with a loss in accuracy of 1.4%.

Table 9: Unbalanced bit reduction on plankton data set with $b = 9$. The accuracies in bold are not statistically different from the accuracy produced from a SVM.

UnBal Bits(s)	compression Ratio	Accuracy	SVs	Red Time	Train Time	Test Time	Random Sampling	Random SVs
SVM	1.0	89.60%	2628	-	24.70s	2.96s	—	
0	0.961	89.30%	2547	0.09s	23.17s	2.88s	89.19%	2547.8
1	0.951	89.60%	2517	0.11s	22.28s	2.84s	89.15%	2525.9
2	0.934	89.20%	2467	0.12s	21.87s	2.80s	89.06%	2490.6
3	0.912	89.30%	2411	0.10s	21.20s	2.76s	88.96%	2437.0
4	0.874	89.30%	2299	0.10s	19.54s	2.65s	88.89%	2350.2
5	0.843	89.20%	2220	0.10s	18.09s	2.51s	88.76%	2280.7
6	0.799	88.80%	2076	0.10s	15.97s	2.36s	88.58%	2174.9
7	0.747	88.20%	1924	0.10s	14.57s	2.15s	88.36%	2055.3
8	0.706	88.20%	1815	0.10s	12.51s	2.00s	88.18%	1959.3
9	0.644	87.10%	1639	0.10s	10.84s	1.81s	87.97%	1814.8
10	0.603	86.40%	1545	0.10s	9.57s	1.69s	87.88%	1715.1
11	0.557	86.70%	1434	0.09s	8.47s	1.59s	87.69%	1602.1
12	0.531	86.10%	1387	0.10s	8.01s	1.53s	87.54%	1537.3
13	0.487	85.60%	1254	0.09s	6.75s	1.39s	87.37%	1433.6
14	0.463	86.00%	1190	0.09s	6.11s	1.32s	87.14%	1371.8
15	0.413	85.10%	1044	0.09s	5.12s	1.18s	86.70%	1244.4
16	0.359	84.80%	939	0.09s	4.03s	1.06s	86.48%	1106.1

With pure bit reduction on the letter data set, bit reduction ($b = 9$) resulted in 0.887 compression ratio and 0.03% loss in accuracy with only a 1.1 speedup in training time. At bit reduction ($b = 10$) accuracy dropped by 0.82%. By using UBR (see Table 10) a speedup of 1.7 was achieved with a loss in accuracy of 0.23% by using bit reduction ($b = 9$) plus an additional 13 attributes ($s = 13$) reduced by one more bit.

In Figure 4 we can see a typical example of how accuracy changes on the phoneme data set with the compression ratio. In this case BRSVM stayed higher in accuracy than random sampling until the compression ratio was low. In Figure 5, we can see how accuracy

Table 10: Unbalanced bit reduction on letter data set with $b = 9$. The accuracies in bold are not statistically different from the accuracy produced from a SVM.

UnBal Bits(s)	compression Ratio	Accuracy	SVs	Red Time	Train Time	Test Time	Random Sampling	Random SVs
SVM	1.000	97.53%	7051	-	75.60s	49.39s		
0	0.887	97.50%	6778	0.17s	66.99s	46.99s	97.22%	6491.2
1	0.869	97.50%	6743	0.22s	65.39s	46.60s	97.17%	6402.6
2	0.859	97.50%	6710	0.22s	64.53s	46.83s	97.14%	6356.4
3	0.850	97.43%	6679	0.23s	64.36s	49.26s	97.13%	6309.9
4	0.843	97.45%	6633	0.22s	63.66s	47.33s	97.12%	6278.9
5	0.836	97.43%	6602	0.22s	62.89s	46.99s	97.09%	6245.5
6	0.819	97.40%	6559	0.21s	61.10s	46.52s	97.07%	6158.7
7	0.801	97.43%	6501	0.22s	59.46s	46.76s	97.01%	6066.9
8	0.785	97.38%	6437	0.21s	58.87s	45.77s	96.98%	5990.6
9	0.748	97.40%	6292	0.22s	54.75s	44.94s	96.87%	5798.0
10	0.738	97.45%	6234	0.21s	53.73s	43.77s	96.84%	5746.4
11	0.700	97.38%	6050	0.22s	49.95s	42.59s	96.74%	5552.0
12	0.670	97.35%	5879	0.20s	46.98s	41.90s	96.65%	5395.7
13	0.654	97.30%	5758	0.21s	45.32s	41.32s	96.61%	5306.8
14	0.583	97.00%	5410	0.20s	38.57s	38.55s	96.33%	4909.0
15	0.538	96.98%	5093	0.20s	34.66s	36.39s	96.14%	4646.7

changes on the page data set with compression. In this case, BRSVM was significantly better until nine bits were compressed with five unbalanced bits compressed. After that point, it's selection of examples results in lower accuracy.

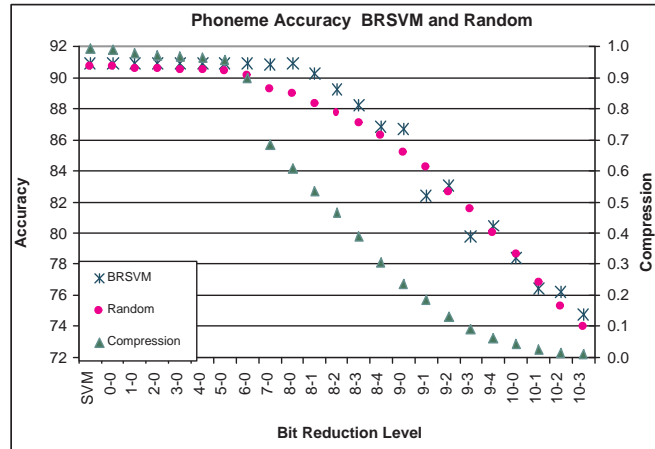


Figure 4: Phoneme accuracy with BRSVM and Random selection of examples at different compression ratios.

4.3 Comparison

Table 11: Results for the Adult data set.

Method	Accuracy	Training Time	Reduction Time	Speed Up	Compression Ratio
Majority Class	75.53%				
9-Bits $s = 0$	84.56%	694.25s	0.73s	1.424	0.788
9-Bits $s = 4$	84.64%	526.21s	0.69s	1.878	0.634
No Reduction	84.65%	989.69s			1.000
srs-SMO	79.7%	0.2s	1.5s	677.1	0.01
squash-SMO	83.0%	0.2s	132.4s	8.7	0.01
boost-SMO	83.3%	7.5s	-	153.47	0.01
full-SMO	84.2%	1151			

A comparison of bit reduction to the squashing methods of Pavlov and Chudova [30] for the Adult, Forest, and Web data sets was done. Their results are the best we have found.

Following the idea of likelihood-based squashing [24] [29], a squashing method was developed for an SVM by Pavlov and Chudova [30]. The likelihood squashing method

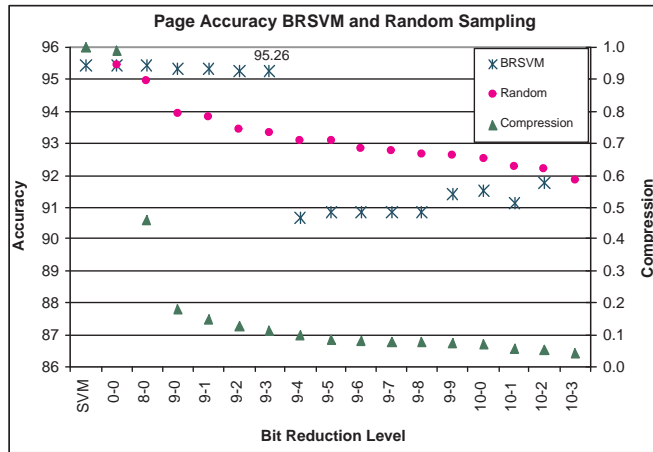


Figure 5: Page accuracy with BRSVM and Random selection of examples at different compression ratios.

assumes a probability model as the classifier. Examples with similar probability $p(x_i, y_i|\theta)$ are grouped together and taken as a weighted exemplar. Pavlov and Chudova used a probabilistic interpretation of SVMs to perform the likelihood squashing with a linear kernel. They call the approach squash-SMO because it was paired with sequential minimal optimization. Simply taking a random 1-5% of the data was called srs-SMO.

The boost-SMO algorithm [32] trains a sequence of classifiers on a small subset of the data (say 1%) while concentrating on examples misclassified (in all the data). While generally suboptimal, it is fast to train and provides accuracy comparable to using all the data.

For comparison the C parameter of the SVM was set to 1.0 for all three data sets, and a **Linear Kernel** was used. All 3 data sets contain nominal features for which no compression, just strict matching, was done. For the adult data set the 8 nominal attributes took on multiple values. We encoded these attributes as groups of inputs. For example, an attribute that took on three values would consist of three inputs only one of which could be non-zero. This resulted in a total of 99 nominal features and 6 continuous features for the SVM.

For the Adult data set, a compression ratio of 0.634 was achieved using 9-bit reduction ($b = 9$) and 4 unbalanced attributes ($s = 4$). The reduction itself took 0.69 seconds, and the time required to train using the reduced set was 526.21 seconds. This resulted in a speedup of 1.88 times with only a 0.01% accuracy loss. In comparison, the squash-SMO method had an 8.7 times speedup with a 1.2% accuracy loss, boost-SMO had a 153.5 times speedup

with only a 0.9% accuracy loss, and the srs-SMO method had a 677.1 times speedup with a 4.5% reduction in accuracy. The results are shown in Table 11. The training times are significantly lower using the 1% selection methods of [30], but the accuracy suffers as a result. However, a 9-bit reduction plus 4 unbalanced attributes produced no statistically significant difference in accuracy.

The forest cover data set has more than 500,000 examples. To compare with the work in [30, 32], rather than other work with more training data but no comparable compression experiments [9], we used the two most populated classes which had the highest misclassification rates: Spruce-fur and Lodge-pole pine. The training data set consisted of 98,884 examples and the test set consisted of 396,257 examples with details in [31]. With the Forest cover data set, bit reduction using 10-bits ($b = 10$) and 4 unbalanced attributes ($s = 4$) resulted in a speedup of 54.0. This result is roughly 30 times faster than what was obtained for the Adult data set. This occurs because more than half of the features in Adult data set are nominal in comparison to 17% in the Forest data set. The number of nominal features is significant for bit-reduction based squashing since the reduction calculation is not applied to nominal features; they are used as-is. In contrast, the squash-SMO method only resulted in a 1.79 times speed-up and resulted in a 0.6% accuracy loss. Boost-SMO had a 14.36 times speedup with only a 1.1% accuracy loss. Using bit reduction, a 0.1% gain in accuracy was achieved as shown in Table 12.

Table 12: Results for the Forest data set.

Method	Accuracy	Training Time	Reduction Time	Speed Up	Compression Ratio
Majority Class	57.26%				
10-Bits $s = 0$	76.10%	271.01s	2.09s	27.50	0.181
10-Bits $s = 4$	75.89%	136.64s	2.34s	54.04	0.076
10-Bits $s = 9$	74.18%	4.62s	2.45s	1061.18	0.017
No Reduction	75.85%	7511.02s		–	1.000
srs-SMO	74.2%	2.0s	13.0s	81.67	0.01
squash-SMO	77.1%	2.0s	682.9s	1.79	0.01
boost-SMO	76.6%	85.3s	-	14.36	0.01
full-SMO	77.7%	1225s	-		1.00

For the Web data set, bit reduction is marginalized in its effectiveness since all of the attributes are boolean and will not have the reduction calculation applied. Thus, a 0-bit reduction is the only level of reduction that will produce meaningful results. In other words, the best that can be done is to remove all of the duplicate examples from the

training set. Even so, we managed to achieve a compression ratio of 0.352, which indicates there are a significant number of duplicate examples in the Web data set. The speed-up was 2.81 times with minimal loss (0.02%) in classification accuracy. From Table 13, we can see squash-smo results in a speedup of 11 times at a cost of a 0.8% drop in accuracy and boost-SMO results in a 402 times speed-up with just a 0.6% loss in accuracy.

Table 13: Results for the Web data set.

Method	Accuracy	Training Time	Reduction Time	Speed Up	Compression Ratio
Majority Class	68.05%				
0-Bits $s = 0$	74.56%	162.87s	5.13s	2.81	0.352
No Reduction	74.58%	471.47s			1.000
srs-SMO	71.2%	0.2	1.5	2363.53	0.01
squash-SMO	74.4%	0.2	361.7	11.10	0.01
boost-SMO	74.6%	10	-	401.8	
full-SMO	75.2%	4018			

The results of our comparison to [30] show that bit reduction has limited utility for data sets that contain more nominal than continuous attributes. In general, the presence of nominal attributes will mean lower compression ratios. Thus, squash-SMO or boost-SMO may be the more appropriate technique for such data sets. However, since bit-reduction uses more of the original data than the squash-SMO method, the classification accuracy will suffer less. It should also be pointed out that the time needed to reduce the training set is significantly lower when using bit reduction.

The Core Vector Machine (CVM) paper [41] gave results for the Forest dataset with a different train/test data split. At 100,000 examples training time was not even twice as fast as the SVM while BRSVM at 98,000 examples was 54 times faster.

4.4 Summary and discussion

Table 14 summarizes the performance of BRSVM on all data sets. The first column gives the name of the data set with the number of attributes in parentheses. The second column has the optimal b and s results for a “good” compression ratio, at which BRSVM achieves speedup with an accuracy that is not statistically different from that obtained by SVM using uncompressed data. A McNemar’s test with $p = 0.05$ was used to determine statistical significance. The third column indicates the compression ratio achieved. The fourth column shows the accuracy achieved with SVM. The fifth column shows the accuracy when using BRSVM with the settings indicated in the second column. The sixth and

seventh columns indicate the speedup in training and test times respectively. The training time speedup reflects the time needed for bit reduction and SVM training. The last three columns show the accuracy of Random Sampling with the same number of examples as the optimal BRSVM, BRSVM accuracy with $s=0$ and random sampling accuracy with the same compression as for BRSVM with $s=0$.

Table 14: Summary of BRSVM on all twelve data sets.

Data set	Optimal b and s	Comp Ratio	SVM Accuracy	BRSVM Accuracy	Train SpeedUp	Pred SpeedUp	Random Sampling	BRSVM $s = 0$	Random $s = 0$
adult(105)	$b = 9, s = 4$	0.634	84.65%	84.64%	1.9	1.8	84.62%	84.56%	84.67%
banana(2)	$b = 8, s = 0$	0.077	90.57%	90.28%	57.6	6.3	89.71%	90.28%	89.70%
forest(54)	$b = 10, s = 4$	0.076	75.85%	75.89%	54.0	14.6	75.79%	76.10%	75.79%
letter(16)	$b = 9, s = 13$	0.654	97.53%	97.30%	1.7	1.2	96.61%	97.22%	95.15%
page(10)	$b = 9, s = 3$	0.112	95.44%	95.26%	15.7	2.4	93.31%	95.35%	93.93%
pendigit(16)	$b = 9, s = 11$	0.621	98.14%	98.03%	1.6	1.0	98.01%	98.17%	98.14%
phoneme(5)	$b = 8, s = 1$	0.609	90.93%	90.29%	2.1	1.2	88.96%	90.93%	89.26%
plankton(17)	$b = 9, s = 8$	0.706	89.60%	88.20%	2.0	1.5	88.18%	89.30%	89.18%
satimage(36)	$b = 9, s = 28$	0.833	91.65%	90.95%	1.2	1.1	91.41%	91.65%	91.65%
shuttle(9)	$b = 8, s = 5$	0.019	99.86%	99.83%	48.9	1.3	99.11%	99.84%	99.41%
waveform(40)	$b = 11, s = 22$	0.185	86.70%	85.60%	19.4	4.1	85.17%	85.80%	86.47%
web(293)	$b = 0, s = 0$	0.352	74.58%	74.56%	2.8	2.6	74.88%	74.56%	74.89%

BRSVM results in speed up ratios from 1.2 to 57.6 in training and 1.0 to 14.6 for prediction. Its application holds the promise of applying SVM to larger problems, as even in the worst case it provided a 20% speedup. It can be very effective (e.g. for forest which has a lot of examples the speedup is 54 times during training and 14.6 times during testing). There is a small accuracy loss ($\leq 0.76\%$ for all but two).

Results are typically improved by using unbalanced bit reduction. The banana data set is the only one for which unbalanced bit reduction does not improve the speedup, but it began with a 57.6 times speed up on the training data. BRSVM with pure bit reduction is more accurate than random sampling on 8 data sets (and with unbalanced bit reduction more accurate on 10). On the letter, satimage and plankton data sets which are relatively high dimensional, pure bit reduction fails to provide a very good compression ratio. It is more difficult to get smooth changes to the compression ratio with the larger search space induced by higher dimensionality. The best bit reduction and compression ratio vary across data sets. For example, pure bit reduction fails to provide a compression ratio between 0.362 and 0.962 on the plankton data set. When unbalanced bit reduction was introduced an intermediate compression ratio of 0.706 was obtained resulting in a speedup in training of 1.96.

Although random sampling has higher variances in theory, it works fairly well in our experiments except for page, phoneme, and shuttle where random sampling was more than

1.2% less accurate than BRSVM. It performs only slightly worse than BRSVM on seven out of twelve data sets and provides slightly higher accuracy on web and satimage. This phenomenon was also observed in [29][38], where complicated data squashing strategies brought small or no advantages over random sampling.

On smaller dimensional datasets such as page, phoneme and shuttle the accuracy of BRSVM was respectively more than 2.0%, 1.5%, and 1.2% accurate than random sampling. BRSVM also gives us finer control in the search for an appropriate level of compression than random sampling. By adjusting compression by one bit at a time a level of compression that does not significantly sacrifice classification accuracy can be quickly found. Unbalanced bit reduction then gives a fine tuning mechanism to squeeze out more compression. Section 4.4.1 discusses how and when to tune these two new parameters b and s . We note that for random sampling to achieve the results given here, one would need to develop an algorithm to determine the size of the random sample which has currently been chosen to match that obtained by bit reduction.

BRSVM should also work well for high dimensional data sets that have relatively tightly packed groupings within classes. If the classes are well spread out in feature space, it may not be as effective and will require effective unbalanced bit reduction, which with lots of attributes (say > 100000) will take more time.

One advantage of our approach when compared with other squashing approaches [3, 30] is that the time to do the squashing is minimal. For nine data sets it was less than 1 second, one required 1.09 seconds, and two data sets, forest and web, had the longest times with 2.3 and 5.1 seconds respectively. The much longer time required for the web data set was because there were 31932 examples with 293 attributes each. Forest with 98,884 examples and 54 features had the second longest reduction time at 2.3 seconds. The compression time is typically orders of magnitude less than the training time whereas in [30] it was sometimes two orders of magnitude greater than the training time. We specifically compare on the Adult data set from the UCI repository using a linear kernel. The same training/test sets were used. Our accuracy went from 84.65% to 84.64% after 9 bit reduction and 4 unbalanced attributes ($b = 9, s = 4$). With Squash-SMO accuracy went from 84.2% down to 83.0% which represents a 1.2% drop in accuracy. BRSVM had a speedup of 1.88 compared to 8.7 for Squash-SMO. If you are willing to build an ensemble of classifiers, boost-SMO generally provides only a small loss of accuracy and was fast for the three data sets for which the authors reported results. However, [32] there is a random selection feature in choosing a small subset of data for boosting that needs to be fully explored to ensure boost-SMO is stable. Generally, BRSVM was more accurate (no loss

in accuracy) on the comparison data sets and provided competitive speed-ups.

In [3] compression by clustering was used. The compression time was $\frac{1}{6}^{th}$ the training time. They looked at just 2 datasets and found speedups of less than two times.

The BRSVM approach does not change the computational complexity of SVM training. It simply reduces the number of examples for training which is crucial as the most straightforward training algorithm is $O(m^3)$ for m examples and SMO which is used here scales approximately quadratically in the number of examples. Hence, it can be used in conjunction with faster SVM training algorithms [43, 45] which do approximations within the learning algorithm. It can be used in conjunction with a parallel learning algorithm such as in [6]. In both cases, this will enable SVMs to be applied to large data sets where their use was not feasible, previously.

4.4.1 Bit Reduction Parameter Tuning

The two bit reduction parameters b and s need to be treated as additional parameters that are to be tuned for a given dataset; in the same way the $\text{cost}(C)$ and γ parameters are tuned for the SVM. This tuning would only need to be done when there is a drastic change in the training data. Otherwise the same values initially determined can be reused.

For very large datasets, such as the Adult and Forest datasets, it is simple to select a small dataset to tune the bit reduction parameters before training on the full training dataset and still achieve a significant speedup. For smaller datasets you would incur a potentially noticeable time cost in tuning the parameters. However, if you collected more labeled examples for the smaller data set, you would not need to choose new compression parameters.

We have implemented a simple down-hill search strategy to automatically select b and s on a random subset of data. The search was performed on a random sample of $n\%$ of the data which was randomly divided into a train set consisting of 50% of the data and a test set of the remainder the data. The strategy is to find the best compression while keeping accuracy loss below a user selected threshold. First the level of bit reduction b that produces compression more than 0.99 was determined. Using that b , we incremented by one until the accuracy loss on the tuning test set was greater than a threshold of 0.5% and chose the value b just before the threshold was reached. Next the number of unbalanced bits s was determined. Initially, $s = 1$ and it was incremented by one until the accuracy loss exceeded the 0.5% threshold. The selected s was the one before the accuracy loss exceeded the threshold.

Experimental results are reported on three large data sets, Forest, Shuttle and Adult. The timing results include the tuning time and are an average over 30 trials with different random seeds. The values for b and s are median values. We have kept the subsampling size small, but enough to have a representative sample. The subsample sizes were 8% for the forest cover data and 20% for the other two data sets.

Table 15 shows the results. The average accuracy loss was just 0.08% using the Forest data and yielded an average 45 times speedup. The average speedup with the shuttle data was 6.6 times with a average accuracy loss of 0.56%. The average accuracy loss with adult was 2.31% with a 2.25 average training speedup which is not so good. The median compression is 1 bit more than the best that can be found by hand resulting in some accuracy loss. Hence, the proposed tuning strategy is suboptimal in some cases and improvements are left for future work.

Table 15: Bit reduction tuning example on Adult, Forest and Shuttle datasets.

Data set	Parameter Search time	Bit Red Parameters	Train Time Using Parameters	Pure SVM Train Time	Overall Speedup	Accuracy Loss
forest	217.12s	$b = 10, s = 2$	981.17s	54152.90s	45.19	0.09%
shuttle	13.87s	$b = 9, s = 0$	0.95s	98.30s	6.63	0.56%
adult	84.28s	$b = 10, s = 6$	349.69s	976.16s	2.25	2.31%

5 Conclusion

In this paper, a bit reduction SVM was proposed to speed up SVMs' during training and prediction. BRSVM groups similar examples together by reducing their resolution. Such a simple method significantly reduces the training time and the prediction time of an SVM in our experiments when bit reduction can compress the data well. It is more accurate than random sampling when the data set is not over-compressed. BRSVM tends to work better with relatively lower dimensional data sets, on which it is more accurate than random sampling and also results in larger speedups. Therefore, feature selection methods might be used to reduce the data dimensionality and potentially help BRSVM to obtain further speedups.

We note that the bit reduced data sets can be given to other types of learning algorithms. Certainly, for SVM variant classifiers [41] and different learning paradigms within SVM's this approach can be applied. Neural networks or import vector machines [47], for instance, take a significant amount of training time for very large data to varying degrees. However, it is not clear that the same amount of compression would necessarily be accept-

able. The algorithms must be modified to accept weighted examples. The characteristics of the algorithms may or may not lend themselves to this type of compression.

We can also conclude if a very high speedup is desired in which a lot of compression is required, random sampling may be a competitive choice. This tends to happen with high dimensional data. For those data sets, BRSVM and random sampling have the potential to be used together. Instead of using one weighted exemplar for each bin, one might randomly sample several examples at a ratio proportional to the number of examples in this bin. Then several weighted exemplars would be used to represent the examples in this bin. This combination method can help when the example distribution is skewed across the bins, and has the potential to improve BRSVM on high dimensional data sets.

6 Acknowledgments

The research was partially supported by the United States Navy, Office of Naval Research, under grant number N00014-02-1-0266 and the NSF under grant EIA-0130768. The authors thank Kevin Shallow, Scott Samson, and Thomas Hopkins for their cooperation in producing and classifying the plankton data set.

References

- [1] S. Asharaf and M. N. Murty. Scalable non-linear support vector machine using hierarchical clustering. *Pattern Recognition, International Conference on*, 1:908–911, 2006.
- [2] S. Asharaf, M. N. Murty, and S. K. Shevade. Multiclass core vector machine. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 41–48, New York, NY, USA, 2007. ACM.
- [3] D. Boley and D. Cao. Training support vector machines using adaptive clustering. In *SIAM International Conference on Data Mining*, 2004.
- [4] C. J. C. Burges. Simplified support vector decision rules. In *International Conference on Machine Learning*, pages 71–77, 1996.
- [5] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems*, volume 9, pages 375–381, 1997.

- [6] L. Cao, S. Keerthi, C.-J. Ong, J. Zhang, U. Periyathamby, X.-J. Fu, and H. Lee. Parallel sequential minimal optimization for the training of support vector machines. *IEEE Transactions on Neural Networks*, 17(4):1039–1049, 2006.
- [7] C. Chang and C. Lin. LIBSVM: a library for support vector machines (version 2.3), 2001.
- [8] W. G. Cochran. *Sampling Techniques*. John Wiley and Sons, Inc., 3 edition, 1977.
- [9] R. Collobert, Y. Bengio, and S. Bengio. Scaling large learning problems with hard parallel mixtures. *Int. J. Pattern Recogn. Artific. Intell.*, 17(3):349–365, 2003.
- [10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2 edition, 2001.
- [11] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.
- [12] J. X. Dong and A. Krzyzak. A fast svm training algorithm. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):367–384, 2003.
- [13] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon. Squashing flat files flatter. *Data Mining and Knowledge Discovery*, pages 6–15, 1999.
- [14] ELENA. <ftp://ftp.dice.ucl.ac.be/pub/neural-nets/elena/database>.
- [15] S. Eschrich, J. Ke, L. Hall, and D. Goldgof. Fast accurate fuzzy clustering through data reduction. *IEEE Transactions on Fuzzy Systems*, 11(2):262–270, 2003.
- [16] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.
- [17] Jayadeva, R. Khemchandani, and S. Chandra. Twin support vector machines for pattern classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(5):905–910, 2007.
- [18] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*, pages 169–184, 1999.
- [19] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt’s smo algorithm for svm design. *Neural Computation*, 13:637–649, 2001.

- [20] Y. Liu, Z. You, , and L. Cao. A novel and quick svm-based multiclassifier. *Pattern Recognition*, 39:22582264, 2006.
- [21] T. Luo, L. O. Hall, D. B. Goldgof, and A. Rensen. Bit reduction support vector machine. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, 2005.
- [22] T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Rensen, and T. Hopkins. Active learning to recognize multiple types of plankton. In *17th conference of the International Association for Pattern Recognition*, volume 3, pages 478–481, 2004.
- [23] T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Rensen, and T. Hopkins. Recognizing plankton images from the shadow image particle profiling evaluation recorder. *IEEE Transactions on System, Man, and Cybernetics–Part B: Cybernetics*, 34(4):1753–1762, August 2004.
- [24] D. Madigan, N. Raghavan, W. Dumouchel, M. Nason, C. Posse, and G. Ridgeway. Likelihood-based data squashing: a modeling approach to instance construction. *Data Mining and Knowledge Discovery*, 6(2):173–190, 2002.
- [25] O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *J. Mach. Learn. Res.*, 1:161–177, 2001.
- [26] C. J. Merz and P. M. Murphy. UCI repository of machine learning database. <http://www.ics.uci.edu/mllearn/MLRepository.html>, 1999.
- [27] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. Machine learning, neural and statistical classification, 1994.
- [28] E. Osuna and F. Girosi. Reducing the run-time complexity of support vector machines. In *Advances in Kernel Methods: support vector machines*, 1999.
- [29] A. Owen. Data squashing by empirical likelihood. *Data Mining and Knowledge Discovery*, pages 101–113, 2003.
- [30] D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 295–299, 2000.
- [31] D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, page 295299, 2000.

- [32] D. Pavlov, J. Mao, and B. Dom. Scaling-up support vector machines using boosting algorithm. In *Proceedings. 15th International Conference on Pattern Recognition*, volume 2, pages 219–222, 2000.
- [33] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. The MIT Press, 1999.
- [34] G. Ratsch, T. Onoda, and K. Muller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001.
- [35] B. Scholkopf, S. Kah-Kay, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radialbasis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765, 1997.
- [36] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K. R. Muller, G. Rätsch, and A. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.
- [37] B. Schölkopf and A. J. Smola. *Learning with kernels*. The MIT Press, 2002.
- [38] Y. C. L. Shih, J. D. M. Rennie, and D. R. Karger. Text bundling: Statistics based data-reduction. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 696–703, 2003.
- [39] B.-Y. Sun, D.-S. Huang, and H.-T. Fang. Lidar signal denoising using least-squares support vector machine. *IEEE Signal Processing Letters*, 12(2):101–104, 2005.
- [40] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1), 1998.
- [41] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. *The Journal of Machine Learning Research*, 6:363–392, 2005.
- [42] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, 2001.
- [43] L. Wang, S. Sun, and K. Zhang. A fast approximate algorithm for training L1-SVMs in primal pspace. *Neurocomputing*, 70:1554–1560, 2007.

- [44] C. K. I. Williams and M. Seeger. Using the nystrom method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688. the MIT Press, 2001.
- [45] G. Wu, E. Chang, Y.-K. Chen, and C. Hughes. Incremental approximate matrix factorization for speeding up support vector machines. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 760–766, 2006.
- [46] H. Yu, J. Yang, and J. Han. Classifying large data sets using svm with hierarchical clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315, 2003.
- [47] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine, 2001.