

Bit Reduction Support Vector Machine

Tong Luo, 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,tluo2)@csee.usf.edu

Andrew Remsen
College of Marine Science
University of South Florida
St. Petersburg, Fl.
email: aremsen@marine.usf.edu

Abstract—Support vector machines are very accurate classifiers and have been widely 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 of the time required for both training and prediction with minimum loss in accuracy. It is also shown to be more accurate than random sampling when the data is not over-compressed.

I. INTRODUCTION

Support vector machines (SVMs) achieve high accuracy in many application domains including our work [16][15] in recognizing underwater zooplankton. However, scaling up SVMs to a very large data set is still an open problem. Training a 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, fast 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 a SVM with all labeled images becomes extremely slow.

In this paper, we propose a simple strategy to speedup 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 in it. This data reduction and aggregation step is very fast and scales linearly with respect to the number of examples. Then a SVM is built on a set of weighted examples which are the exemplars of their respective bins. Our experiments indicate that bit reduction SVM (BRSVM) significantly reduces the training time and prediction time with a minimal loss in accuracy. It outperforms random sampling on most data sets when the data are not over-compressed. We also find that on one high

dimensional data set, that bit reduction does not perform as well as random sampling, thus providing a limit on the performance of BRSVM for high dimensional data sets.

II. PREVIOUS WORK

There are two main approaches to speed up training of SVMs. One approach is to find a fast algorithm to solve the quadratic programming (QP) problem for a SVM. "Chunking", introduced in [28], solves a QP problem on a subset of data. Chunking only keeps the support vectors on the subset and replaces others with data that violate the Karush-Kuhn-Tucker (KKT) conditions. Using an idea similar to chunking, decomposition [13] 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 a SVM on large scale problems. Sequential minimum optimization (SMO) [23] and its enhanced versions [14] [8] take decomposition to the extreme: Each working set only has two examples and their optimal coefficients can be solved 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 [12][29] is to use a low-rank matrix to approximate the Gram matrix of a SVM. As a consequence, the QP optimization on the small matrix requires significantly less time than on the whole Gram matrix.

The other main approach of speeding up SVM training comes from the idea of "data squashing", which was proposed in [9] 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. Researchers have applied data squashing to SVMs. Several clustering algorithms [30][27][1] were used to partition data and build a SVM based on the statistics from each cluster. In [1], the SVM model built on the reduced set was used to predict on the whole training data. Examples falling in the margin or being misclassified were taken out from their original clusters and added back into the training data for retraining. However, both [30] and [27] assumed a linear kernel and it might not generalize well

to other kernels. In [1], two experiments were done with a linear kernel and only one experiment used a third-order polynomial kernel. Moreover, it is not unusual that many examples fall into the margin of a SVM model especially for a RBF kernel. In such cases, retraining with all examples within the margin is computationally expensive. Following the idea of the likelihood-based squashing [17] [21], a likelihood squashing method was developed for a SVM by Pavlov and Chudova [22]. The likelihood squashing method 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. Still, only a linear kernel was used in their experiments.

Most work [2][3][20][25] on enabling fast prediction with SVMs focused on the problem of reducing the number of SVs obtained. Since the prediction time of a SVM depends on the number of support vectors, they searched 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 [25][26], 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.

Data squashing approaches seem promising and can be combined with fast QP like SMO etc. for fast training and prediction. However, most work [27][1][30] in data squashing+SVM requires clustering the data and/or linear kernels [27][30][22]. 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 is necessary to apply data squashing+SVMs to real-world applications. In this paper, we propose a simple and fast method data compression method: bit-reduction SVM (BRSVM). It does not require any computationally expensive clustering algorithms and works well with RBF kernels as shown in our experiments.

III. 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. 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 [5]. In spite of its high variance, random sampling has been shown to work very well in experiments [21][27]: It was as accurate as or slightly less accurate than more complicated methods.

A. 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 [11], which applied bit reduction to speed up the fuzzy c-means (FCM) clustering algorithm. However, in classification only examples from the same class should be aggregated together.

There are three steps involved in bit reduction for a 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 an arbitrary number used to scale v and 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 $Z = 1000$.

- 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.

$$I(v) \leftarrow I(v) \gg b \quad (1)$$

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, the mean of examples within the same bin is computed as their representative. The weight of the representative 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 leaving training still slow. The best number of bits reduced (b) varies for different data sets. It can be found by trial-and-error by searching for an appropriate reduction in training data set size. 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 a bin has examples from many different classes. Therefore, in the aggregation step, the mean statistic of examples in the same bin was computed individually for each class. This can at least alleviate 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 I describes the bit reduction procedure for four 1-d examples with class label y_i .

TABLE I
AN 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 II
WEIGHTED EXAMPLES AFTER THE AGGREGATION STEP.

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

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 II shows the weighted examples 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 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 [11]. Universal hashing [6] 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 $2m$. Please see [6] for more detail about universal hashing function.

B. Weighted SVM

Pavlov et al. [22] proposed a method to train a weighted SVM, although its description in [22] 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 $y_i \in \{-1, 1\}$, a SVM solves the following problem

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

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:

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

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).

Introducing the Lagrangian multiplier α_i , Eq. (3) leads to

$$\begin{aligned} L(\alpha, w, b) = & \frac{1}{2} \langle w, w \rangle + \frac{C}{m} \sum_{i=1}^m \beta_i \xi_i \\ & - \sum_{i=1}^m \alpha_i (y_i (\langle w, \phi(x_i) \rangle + b) - 1 + \xi_i) \\ \alpha_i > & 0, i = 1, \dots, m \end{aligned} \quad (4)$$

where α is the vector $(\alpha_1, \alpha_2, \dots, \alpha_m)$. Its saddle point solution can be computed by taking the partial derivatives of $L(\alpha, w, b)$.

$$\frac{\partial L(\alpha, w, b)}{\partial w} = 0 \text{ and } \frac{\partial L(\alpha, w, b)}{\partial b} = 0 \quad (5)$$

We get

$$w = \sum_{i=1}^m \alpha_i y_i \phi(x_i) \quad (6)$$

$$\sum_{i=1}^m \alpha_i y_i = 0 \quad (7)$$

Substitute them into Eq. (4) and the dual form of a weighted SVM is as follows.

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \frac{C \beta_i}{m}, i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \quad (8)$$

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 a normal SVM such as the SMO [23] can be used to solve a weighted SVM by modifying the boundary condition slightly.

IV. EXPERIMENTS

We experimented with BRSVM on nine data sets: banana [24], phoneme [10], shuttle [19], page, pendigit, letter [18], SIPPER II plankton images, waveform and satimage. They come from several sources ranging in size from 5000 to 58,000

examples and from 2 to 36 attributes. They are summarized in Table III.

The plankton data set was originally used in [15]. Its objective is to classify the five most abundant types of plankton from 17 selected image features from 3-bit plankton images.

TABLE III
DESCRIPTION OF THE NINE DATA SETS.

Dataset	# of data	# of attributes	# of classes
banana	5300	2	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	21	3
satimage	6435	36	6

The Libsvm tool [4] for training support vector machines was modified and used in all experiments. The RBF kernel ($k(x, y) = \exp(-g\|x - y\|^2)$) was employed. The kernel parameter g and the regularization constant C were tuned by a 5-fold cross validation on the training data. g and C were searched for from all combinations of the values in $(2^{-10}, 2^{-9}, \dots, 2^4)$ and $(2^{-5}, 2^{-4}, \dots, 2^9)$ respectively. We used the same training and test separation as given by original uses 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 all nine data sets have more than 5000 examples, 20% of the total data will have more than 1000 examples. We believe it provided a relatively stable estimation. We built SVMs on the training set with the optimal parameters and reported the accuracy on the test set. All our experiments were run on a Pentium 4 PC at 2.6 GHZ with 1 GB memory under the Redhat 9.0 operation system.

A. Experiments with pure bit reduction

Due to space limits, we only describe in detail the experimental results using BRSVM on three data sets as shown in Tables IV–VI. Other data sets results are summarized in a later section. The last row of each table records the result of a 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. 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 [7] is used to check whether BRSVM accuracy is statistically significantly different from the accuracy of a SVM built on the uncompressed data set. The number in bold indicates the difference is not statistically significant at the $p = 0.05$ level. The fourth column is the time for bit reduction plus BRSVM training time. The time

required to do example aggregation is included in this training time. The fifth 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. In the last column, the average accuracy of random sampling on the test set is listed for comparison. The subsampling ratio is set to equal the compression ratio of BRSVM. Since the random sampling has a random factor, we did 50 experiments for each subsampling ratio and recorded the average statistics. This accuracy is listed in the last column of Tables IV–VI titled random subsampling accuracy.

The experimental results on the banana data set are shown in Table IV. As more bits are reduced, fewer examples are used in training. Thus training time is 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 9 bits are reduced, BRSVM runs 129 times faster during training and 33 times faster during prediction than a normal SVM. Its accuracy is not statistically significantly different from a SVM built on all the data at the $p = 0.05$ level. BRSVM is as or more accurate than a SVM with random sampling up to 10-bit reduction.

Phoneme is another relatively low-dimensional data set with five attributes. Table V presents the experimental results of BRSVM on this data set. When 8 bits are reduced, BRSVM runs 1.9 times faster during training and 1.2 times faster during prediction than a normal SVM. Its accuracy is not statistically significantly different from a SVM built on all the data at the $p = 0.05$ level. BRSVM is as or more accurate than random sampling when the compression ratio is larger than 0.059.

Similar positive results were observed on shuttle, page, letter, and waveform and are summarized in a later section.

Table VI shows the experimental results on a higher dimensional data set—plankton. BRSVM is 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 0.962 to 0.362, resulting in a significant loss in accuracy.

At the 10 and 11 bit reduction levels where the compression ratios are less than or equal to 0.362, the accuracies of BRSVM are 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 this bin. Random sampling, on the other hand, selects the examples more uniformly. If 80% of the

TABLE IV

BRSVM ON THE BANANA DATA SET. THE ACCURACY IN BOLD MEANS IT IS NOT STATISTICALLY SIGNIFICANTLY DIFFERENT FROM THE ACCURACY OF A SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	random subsampling accuracy
0-1	1.000	0.902	2.59s	0.33s	0.902
2	0.996	0.902	2.59s	0.33s	0.902
3	0.987	0.902	2.59s	0.33s	0.902
4	0.957	0.902	2.45s	0.31s	0.902
5	0.842	0.902	1.99s	0.29s	0.902
6	0.572	0.902	0.98s	0.23s	0.901
7	0.245	0.903	0.21s	0.12s	0.895
8	0.077	0.900	0.03s	0.05s	0.890
9	0.024	0.890	0.02s	0.01s	0.865
10	0.007	0.740	0.01s	0.01s	0.687
SVM	1.000	0.902	2.58s	0.33s	

TABLE V

BRSVM ON THE PHONEME DATA SET. THE ACCURACY IN BOLD MEANS IT IS NOT STATISTICALLY SIGNIFICANTLY DIFFERENT FROM THE ACCURACY OF A SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	random subsampling accuracy
0	0.992	0.895	18.61s	1.03s	0.895
1	0.984	0.895	18.59s	1.03s	0.895
7	0.891	0.895	14.21s	0.97s	0.890
8	0.679	0.893	9.28s	0.83s	0.873
9	0.303	0.846	2.01s	0.41s	0.824
10	0.059	0.752	0.09s	0.09s	0.730
SVM	1.000	0.895	17.51s	1.03s	

TABLE VI

BRSVM ON THE PLANKTON DATA SET. THE ACCURACY IN BOLD MEANS IT IS NOT STATISTICALLY SIGNIFICANTLY DIFFERENT FROM THE ACCURACY OF A SVM.

bit reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	random subsampling accuracy
0-8	0.995	0.889	24.02s	2.42s	0.886
9	0.962	0.887	23.14s	2.31s	0.884
10	0.362	0.829	2.79s	0.74s	0.854
11	0.070	0.695	0.09s	0.12s	0.771
SVM	1.000	0.887	24.23s	2.42s	

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.

It should be noted that the compression ratios on some high-dimensional data sets drop much faster than those on the low-dimensional data sets. This phenomenon is caused by the ‘‘Curse of Dimensionality’’. The corresponding interpretation in our case is that the data in a high-dimensional space are sparse and far from each other. Bit reduction will either group very few data together or put too many data in the same bin. As

a result, BRSVM on high dimensional data does not perform as well as on relatively lower dimensional data sets.

B. Experiments with unbalanced bit reduction

We used a simple solution to get a better compression ratio: unbalanced bit reduction (UBR). UBR works by reducing a different number of bits for different attributes. For instance, if reduction of a bits results in very little compression while reduction of $a + 1$ bits compresses the data too much, UBR randomly selects several attributes to reduce $a + 1$ bits while it applies a -bit reduction to the rest of the attributes. In this way, an intermediate compression ratio can be obtained. Since trying all of attributes to get a desired compression ratio is time consuming especially for high dimensional data sets, we use Algorithm 1 to choose the optimal number of attributes as follows:

In Algorithm 1, a bits are reduced on all the attributes initially. The desired compression ratio would be a range given by the user. Since one more bit reduction on all the attributes would compress the data too much, steps 2–12 determine the number of attributes s to be reduced by one more bit,

Algorithm 1 Unbalanced Bit Reduction

- 1: I^a and C^a are the data set and the compression ratio after reduction of a bits respectively. C^a is too large while C^{a+1} is too small. $A = \{a_1, a_2, \dots, a_r\}$ is the set of r attributes.
 - 2: $s = v = \lfloor r/2 \rfloor$.
 - 3: **if** $v=0$ **then**
 - 4: Stop.
 - 5: **end if**
 - 6: Randomly select s attributes from A , apply 1 more bit reduction on the s attributes, I^a is further compressed to $I^{a,s}$ with compression ratio $C^{a,s}$.
 - 7: **if** $C^{a,s} >$ desired compression ratio range **then**
 - 8: $v = \lfloor v/2 \rfloor$, $s = s + v$, go to 3.
 - 9: **end if**
 - 10: **if** $C^{a,s} <$ desired compression ratio range **then**
 - 11: $v = \lfloor v/2 \rfloor$, $s = s - v$, go to 3.
 - 12: **end if**
 - 13: Apply BRSVM on the reduced data set $I^{a,s}$ with randomly selected s 50 times and record the mean and the standard deviation of the compression ratio and the test accuracy over the 50 runs.
-

which enables a compression ratio falling into a desired range. Algorithm 1 can be also run in an interactive mode by asking the user to judge whether the $C^{a,s}$ is good enough at step 7 and 10. Considering the random factor in selecting the s attributes, we run the UBR 50 times and record the statistics in step 13. This provides more stable results because it experiments with BRSVM on compression ratios resulting from 1 more bit reduction on different combinations of s attributes.

We experimented with UBR on phoneme, pendigit, plankton, waveform and satimage, on which pure bit reduction did not result in ideal incremental compression ratios. In this paper we define a good compression ratio as the minimum compression ratio with an accuracy within 1.2% of that obtained from a SVM trained on the uncompressed data set. In our UBR experiments, Algorithm 1 was applied in interactive mode. Basically, the program asked one to decide whether $C^{a,s}$ fell into the desired compression ratio range at step 7 and step 10. If the ratio was acceptable, the program proceeded to build SVMs on the reduced data set at step 13. We ran the random subsampling 50 times at the same compression ratio as UBR for comparison.

Due to space limits, we only present the experimental results of UBR on phoneme and plankton as shown in Tables VII–VIII. Algorithm 1 was applied to find a s which gave a good compression ratio. In the tables, the first column records the s , the second column is the the mean and the standard deviation (in parentheses) of compression ratios from the 50 runs. The third column and the last column record the mean and the standard deviation of the accuracies over the 50 runs on the test set from BRSVM using UBR and random sampling respectively. Assuming the accuracies of 50 runs follow a normal distribution, we applied the t test to check whether

the accuracy is statistically significantly different from the accuracy of a SVM built on the uncompressed data set. The number in bold indicates the difference is not statistically significant at $p = 0.05$ level. The fourth and the fifth column are the average training time and prediction time respectively.

The pure bit reduction experiments on phoneme were recorded in Table V. After 8 bit reduction, BRSVM gives a 0.679 compression ratio and 1.9 times speedup in the training phase with a loss of 0.3% in accuracy. While after 9 bit reduction, the compression ratio drops to 0.303 and the corresponding 4.9% accuracy loss could not be tolerated. Since we will accept up to 1.2% accuracy loss, we applied UBR to search for a compression ratio between 0.679 and 0.303. We hoped this could give more speedup than 1.9 times from a 8-bit reduction. We first applied 8 bit reduction to the data and then used Algorithm 1 to find an s which gives a good compression ratio. See Table VII for the improved UBR results on the phoneme data set with over a 2 times speed-up with small accuracy loss.

From Table VIII, we see UBR provides a compression ratio of 0.739 on the plankton data set at $b = 9, s = 10$. The corresponding training and prediction phase were 1.6 and 1.4 times faster respectively with a 0.11 accuracy loss. BRSVM is just slightly more accurate than random sampling.

C. Summary and discussion

Table IX summarizes the performance of BRSVM on all nine data sets. The second column is the optimal b and s resulting in a “good” compression ratio, at which BRSVM achieves significant speedup with an accuracy loss less than 1.2%. The accuracy loss in the third column is defined as (accuracy of SVM – accuracy of BRSVM). The number in bold means the loss is not statistically significant. The speedups in the fourth and fifth columns are calculated as the speedup ratio in the previous experiments.

BRSVM works well on the nine data sets. At a small accuracy loss (less than 1.5%), the training and prediction speedup ratios range from 1.3 and 1.1 on the data set with the highest dimension to 245.2 and 33.0 on the lower dimensional data sets. Although accuracy loss exists (e.g. statistically significant) on seven out of nine data sets, it is small (less than 1.2%) and potentially acceptable to save time on large data sets.

Pure bit reduction ($s = 0$) performs very well on the four data sets with up to 10 attributes: banana, phoneme, shuttle and page. It achieves up to 245.2 times speedup in training and up to 33.0 times speedup in prediction without much loss in accuracy on the four data sets. On one relatively high-dimensional data set–letter, BRSVM with pure bit reduction is 2.6 times faster in training and 1.5 times faster in prediction with 0.9% loss in accuracy. BRSVM with pure bit reduction is more accurate than random sampling on five data sets. On the pendigit, plankton and waveform data sets with relatively high dimensional data, pure bit reduction fails to provide a very good compression ratio, hence making BRSVM not as effective as random sampling. The justification is as follows:

TABLE VII

BRSVM OF UBR AFTER 8-BIT REDUCTION ON THE PHONEME DATA SET. THE ACCURACY IN BOLD MEANS IT IS NOT STATISTICALLY SIGNIFICANTLY DIFFERENT FROM THE ACCURACY OF A SVM. THE NUMBER IN PARENTHESES IS THE STANDARD DEVIATION.

# of attributes reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	random subsampling accuracy
2	0.550 (0.003)	0.888 (0.0027)	6.40s	0.67s	0.863 (0.0059)
3	0.467 (0.008)	0.880 (0.0049)	4.74s	0.59s	0.856 (0.0067)
SVM	1.000	0.895	17.51s	1.03s	

TABLE VIII

BRSVM OF UBR AFTER 9-BIT REDUCTION ON THE PLANKTON DATA SET. THE ACCURACY IN BOLD MEANS IT IS NOT STATISTICALLY SIGNIFICANTLY DIFFERENT FROM THE ACCURACY OF A SVM. THE NUMBER IN PARENTHESES IS THE STANDARD DEVIATION.

# of attributes reduction	compression ratio	BRSVM accuracy	BRSVM training time	BRSVM prediction time	random subsampling accuracy
8	0.814 (0.034)	0.881 (0.0040)	18.03s	1.94s	0.880 (0.0030)
10	0.739 (0.039)	0.876 (0.0055)	15.03s	1.73s	0.875 (0.0039)
12	0.638 (0.036)	0.866 (0.0059)	11.22s	1.50s	0.872 (0.0045)
SVM	1.000	0.887	24.23s	2.42s	

TABLE IX

SUMMARY OF BRSVM ON ALL NINE DATA SETS. THE ACCURACY IN BOLD MEANS IT IS NOT STATISTICALLY SIGNIFICANTLY DIFFERENT FROM THE ACCURACY OF A SVM.

Data set	Optimal b and s	Accuracy loss (BRSVM)	Speedup in training	Speedup in prediction
banana	$b=9, s=0$	1.2%	129.0	33.0
phoneme	$b=8, s=2$	0.7%	2.7	1.7
shuttle	$b=10, s=0$	1.2%	245.2	2.4
page	$b=9, s=0$	-0.5%	7.9	1.8
pendigit	$b=10, s=8$	1.1%	12.1	3.0
letter	$b=10, s=0$	0.9%	2.6	1.5
plankton	$b=9, s=10$	1.1%	1.6	1.4
waveform	$b=10, s=18$	0.9%	13.0	4.0
satimage	$b=9, s=31$	1.0%	1.3	1.1

a high compression ratio results in minimal speedup while a too low compression ratio makes BRSVM less accurate. The best bit reduction and compression ratio vary across data sets. In our experiments, a high compression ratio is good for low-dimensional data sets while an intermediate compression ratio is desired for high-dimensional data sets. For instance, a 49% compression ratio is very good for BRSVM on the letter data set. As pure bit reduction fails to provide a compression ratio between 0.362 and 0.962 on the plankton data set, BRSVM is not as effective as random sampling. When unbalanced bit reduction was introduced for the data sets, BRSVM obtained intermediate compression ratios, which result in better accuracies than random sampling and significant speedups. On the highest dimensional data satimage, BRSVM is not as accurate as random sampling: At the optimal $b = 9$ and $s = 31$, the compression ratio of BRSVM is 0.885 and its corresponding accuracy is 90.7%, which is 0.6% less than that of random sampling.

Although random sampling has higher variances in theory, it works fairly well in our experiments except for banana and phoneme where random sampling is more than 2% less accurate than BRSVM. It performs only slightly worse than

BRSVM on six out of nine data sets. This phenomenon was also observed in [21][27], where complicated data squashing strategies brought small or no advantages over random sampling. On satimage—the highest dimensional data set, random subsampling is slightly more accurate than BRSVM. Moreover, when a large compression ratio is needed for very fast training, random sampling outperforms BRSVM especially on high dimensional data sets.

One advantage of our approach when compared with other squashing approaches [1], [22] is that the time to do the squashing is minimal. The longest time required to squash data was a 10 bit reduction on the shuttle data set and that was .07 seconds. The time to squash the data for pendigits took the second-most time at 0.03 seconds. The compression time is typically orders of magnitude less than the training time whereas in [22] 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 was 83.98% and 83.924% after 9 bit reduction, which is a bit better than their accuracy of 82.95%. In training time, our data reduction to training time ratio was 0.0038 compared to

662. The time required for data reduction in our approach was significantly less. Our speed-up was 5.5 times vs. 8.7 times for them. Since the processors used are different it is hard to compare times, but we believe they have a much faster classifier based on the listed times but the overall process is faster in our approach.

In [1] the training time was comparable to six times more than the compression/clustering time. On the other hand, they found speedups of less than two times for a couple of data sets.

V. CONCLUSION

In this paper, a bit reduction SVM is proposed to speed up SVMs' training and prediction. BRSVM groups similar examples together by reducing their resolution. Such a simple method reduces the training time and the prediction time of a SVM significantly 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 shows more significant speedups. Therefore, feature selection methods might be used to reduce the data dimensionality and potentially help BRSVM to obtain further speedups. It should be noted that no feature reduction has been done on most of the data sets used in our experiments.

We can also conclude if a very high speedup is desired in which a high compression ratio is required, random sampling may be a better 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 can 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 examples distribution is skewed across the bins, and has the potential to improve BRSVM on high dimensional data sets.

VI. 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, Kurt Kramer, Scott Samson, and Thomas Hopkins for their cooperation in producing and classifying the plankton data set.

REFERENCES

- [1] D. Boley and D. Cao. Training support vector machines using adaptive clustering. In *SIAM International Conference on Data Mining*, 2004.
- [2] C. J. C. Burges. Simplified support vector decision rules. In *International Conference on Machine Learning*, pages 71–77, 1996.
- [3] 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.
- [4] C. Chang and C. Lin. LIBSVM: a library for support vector machines (version 2.3), 2001.
- [5] W. G. Cochran. *Sampling Techniques*. John Wiley and Sons, Inc., 3 edition, 1977.

- [6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2 edition, 2001.
- [7] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.
- [8] J. X. Dong and A. Krzyzak. A fast svm training algorithm. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):367–384, 2003.
- [9] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon. Squashing flat files flatter. *Data Mining and Knowledge Discovery*, pages 6–15, 1999.
- [10] ELENA. <ftp://ftp.dice.ucl.ac.be/pub/neural-nets/elena/database>.
- [11] 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.
- [12] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.
- [13] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*, pages 169–184, 1999.
- [14] 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.
- [15] T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, 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.
- [16] T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, 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.
- [17] 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.
- [18] C. J. Merz and P. M. Murphy. UCI repository of machine learning database. <http://www.ics.uci.edu/mllearn/MLRepository.html>, 1999.
- [19] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine learning, neural and statistical classification*, 1994.
- [20] E. Osuna and F. Girosi. Reducing the run-time complexity of support vector machines. In *Advances in Kernel Methods: support vector machines*, 1999.
- [21] A. Owen. Data squashing by empirical likelihood. *Data Mining and Knowledge Discovery*, pages 101–113, 2003.
- [22] 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.
- [23] 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.
- [24] G. Ratsch, T. Onoda, and K. Muller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001.
- [25] 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.
- [26] B. Schölkopf and A. J. Smola. *Learning with kernels*. The MIT Press, 2002.
- [27] 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.
- [28] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, 2001.
- [29] 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.
- [30] 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.