Multiclass Reduced-Set Support Vector Machines

Support Vector Machines, Reduced Set Methods, Kernel Pre-Image, Multiclass, Differential Evolution

Benyang Tang BENYANG.TANG@JPL.NASA.GOV
Dominic Mazzoni DOMINIC.MAZZONI@JPL.NASA.GOV
Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Dr, Pasadena, CA 91109

Abstract

There are well-established methods for reducing the number of support vectors in a trained binary support vector machine, often with minimal impact on accuracy. We show how reduced-set methods can be applied to multiclass SVMs made up of several binary SVMs, with significantly better results than reducing each binary SVM independently. Our approach is based on Burges' approach that constructs each reduced-set vector as the pre-image of a vector in kernel space, but we extend this by recomputing the SVM weights and bias optimally using the original SVM objective function. This leads to greater accuracy for a binary reduced-set SVM, and also allows vectors to be "shared" between multiple binary SVMs for greater multiclass accuracy with fewer reduced-set vectors. We also propose computing pre-images using differential evolution, which we have found to be more robust than gradient descent alone. We show experimental results on a variety of problems and find that this new approach is consistently better than previous multiclass reduced-set methods, sometimes with a dramatic difference.

1. Introduction

The time it takes to classify a new example using a trained support vector machine (SVM) is proportional to the number of support vectors, which can often number in the hundreds or thousands. While SVMs are a very robust and powerful technique for supervised classification, the large size and slow query time of a trained SVM is one hindrance to their practical application. Several "reduced-set" methods have been proposed which successfully help to alleviate this problem, either by eliminating less important support vectors or by constructing a new smaller set of vectors, often with minimal impact on accuracy.

While SVMs inherently solve binary classification problems, the theory of multiclass classification using SVMs is now well-studied (e.g., Hsu and Lin (2002), Duan and Keerthi (2005)). Specifically, the most popular and widely successful techniques combine a number of binary SVMs to solve a multiclass problem. Reduced-set methods were developed for binary SVMs and have only been applied to multiclass problems naively, reducing each component SVM in isolation. We propose a new approach that enables the reduced-set vectors to be shared by all of the component SVMs at once, leading to much faster convergence with fewer reduced-set vectors.

We first introduce several innovations to the binary reduced-set method proposed by Burges (1996). First, we formulate the pre-image problem such that it does not require minimizing the vector weight β simultaneous with the reduced-set vector, which avoids a tricky singularity. Second, we use a two-stage optimization procedure to approximate the pre-images, first using differential evolution to avoid shallow local minima, and then using gradient descent to refine the best guess. Finally, after the reduced set vectors are contructed, we optimally compute the reduced-set vector weights and bias term by running the SVM training algorithm with a modified kernel matrix. It is this last innovation that is the key to the success of our multiclass method, as it makes it possible to share vectors between separate binary SVMs.

In the next section, we discuss related work. In section 3, we present the mathematics and our algorithm for creating reduced sets for binary SVMs, including the enhancements necessary to support multiclass

Appearing in Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 2006.

reduced-set SVMs. In section 4, we present our algorithm for multiclass reduced-set SVMs, and we follow this with experimental results and then concluding remarks.

2. Related Work

Burges (1996) developed the first constructive reduced-set algorithm, using a gradient descent method to solve the pre-image problem. This approach is also described in (Burges & Schölkopf, 1996) and further developed in (Schölkopf et al., 1999). As the pre-image problem involves nonlinear optimization, Burges' method sometimes gets stuck in premature local minima, as mentioned by Kwok and Tsang (2004) and Bakir et al. (2004). One has to restart the process with many initial guesses. We will analyze Burges' method in greater detail in Section 3.2.

Schölkopf et al. (1999) introduced a fixed-point iteration method for the pre-image problem for the Gaussian kernel. While this approach has several advantages, it too is unable to avoid the local minimum problem.

Kwok and Tsang (2004) presented a pre-image method that utilizes the relationship between the distances between the input space vectors and the distances between the feature space vectors, the latter being calculated with the kernel function. The method involves only linear matrix calculations and is not iterative. Kwok applied the method to kernel PCA, where it seems to be quite successful, but not to constructing reduced sets, and we did not find it to work as well for this. Bakir et al. (2004) trained an SVM regression model to represent the inverse mapping from the feature space to the input space, thus obtaining a pre-image function. As did Kwok and Tsang (2004), they applied it to the kernel PCA problem successfully, but also found that it did not perform well when applied to the reduced set problem. It is interesting to note that the methods of Kwok and Tsang (2004) and Bakir et al. (2004) can both be applied to pre-image applications with a discrete input space, since they do not require the gradient of the objective function.

While this paper is focused on constructive techinques for reducing support vector machines, other methods have been proposed to speed up SVM query time without reducing the number of support vectors, such as (DeCoste & Mazzoni, 2003).

3. Reducing Binary SVMs

3.1. SVM Classifiers

The details of SVM classification are descibed in (Schölkopf & Smola, 2002; Burges, 1998). Here we outline the basic equations, and we follow the notations of (Schölkopf & Smola, 2002).

Let x_i (for $1 \leq i \leq N_x$) be the input vectors in input space, with corresponding binary labels $y_i \in \{-1, 1\}$. Let $\mathbf{x}_i = \phi(x_i)$ be the corresponding vectors in feature space, where $\phi(x_i)$ is the implicit kernel mapping, and let $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ be the kernel function, implying a dot product in the feature space. Popular kernel functions (with model selection parameters a, p, γ) include:

$$\begin{array}{ll} \text{linear:} & k(u,v) = u \cdot v, \\ \text{polynomial:} & k(u,v) = (u \cdot v + a)^p, \\ \text{gaussian:} & k(u,v) = \exp(-\gamma \cdot ||u - v||^2) \end{array}$$

The optimization problem for a soft-margin SVM is

$$\min_{\mathbf{w},b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \right\}, \tag{1}$$

subject to the constraints $y_i(\mathbf{w} \cdot \mathbf{x} + b) = 1 - \xi_i$ and $\xi_i \geq 0$, where $\mathbf{w}$ is the normal vector of the separating hyperplane in feature space, and $C > 0$ is a regularization parameter controlling the penalty for misclassification.

Equation (1) is referred to as the primal equation. From the Lagrangian form of (1), we derive the dual problem:

$$\max_{\alpha} \left\{ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right\}, \tag{2}$$

subject to $0 \leq \alpha_i \leq C$. This is a quadratic optimization problem that can be solved efficiently using algorithms such as Sequential Minimal Optimization (Platt, 1999). Typically, many α_i go to zero during optimization, and the remaining x_i corresponding to those $\alpha_i > 0$ are called support vectors. To simplify notation, from here on we assume that all non-support-vectors have been removed, so that N_x is now the number of support vectors, and $\alpha_i > 0$ for all i. With this formulation, the normal vector of the separating plane, $\mathbf{w}$, is calculated as:

$$\mathbf{w} = \sum_{i=1}^{N_x} \alpha_i y_i \mathbf{x}_i. \tag{3}$$

Note that because $\mathbf{x}_i = \phi(x_i)$ is defined implicity, $\mathbf{w}$ exists only in feature space and cannot be computed directly. Instead, the classification $f(q)$ of a new query vector q can only be determined by computing the kernel function of q with every support vector:

$$f(q) = \text{sign}\left(\sum_{i=1}^{N_x} \alpha_i y_i \cdot k(q, x_i) + b\right), \tag{4}$$

where the bias term b is the offset of the hyperplane along its normal vector, determined during SVM training. The reduced-set idea is to approximate $\mathbf{w}$ using a smaller number of vectors $\mathbf{z}_i$, $1 \le i \le N_z$, where $N_z < N_x$, with weights β_i in place of $\alpha_i \cdot y_i$, thus speeding up the classification phase:

$$\mathbf{w} \approx \sum_{i=1}^{N_z} \beta_i \phi(z_i) \tag{5}$$

$$f(q) \approx \text{sign}\left[\sum_{i=1}^{N_z} \beta_i \cdot k(q, z_i) + b\right]. \tag{6}$$

The reduced set can be constructed by a series of pre-image solutions. An improved pre-image algorithm is described in the next section.

3.2. Re-Formulating the Pre-Image Problem

The pre-image problem is to find a vector z in the input space whose β-weighted mapping $\phi(z)$ in the feature space best approximates a vector $\Psi = \sum_i^{N_x} \eta_i \phi(x_i)$. To find the pre-image, we solve for β and z by minimizing the distance ρ between the β-weighted mapping of z and the vector Ψ:

$$\begin{aligned}
\min_{\beta,z} \rho &= \min_{\beta,z} \left\| \beta\phi(z) - \sum_{i=1}^{N_x} \eta_i \phi(x_i) \right\|^2 \\
&= \min_{\beta,z} \left\{ \sum_{i,j=1}^{N_x} \eta_i \eta_j k(x_i, x_j) + \beta^2 k(z,z) \right. \\
&\quad \left. -2\beta \sum_{i=1}^{N_x} \eta_i k(x_i, z) \right\}.
\end{aligned} \tag{7}$$

By requiring $\frac{\partial \rho}{\partial \beta} = 0$, we solve for β:

$$\beta = \sum_{i=1}^{N_x} \eta_i k(x_i, z)/k(z,z). \tag{8}$$

Thus the pre-image problem becomes:

$$\begin{aligned}
\min_z \rho &= \min_z \left\{ \|\Psi\|^2 - \frac{\left[\sum_{i=1}^{N_x} \eta_i k(x_i, z)\right]^2}{\|\phi(z)\|^2} \right\} \\
&= \|\Psi\|^2 \min_z \left\{ 1 - \left[\frac{\Psi \cdot \phi(z)}{\|\phi(z)\| \, \|\Psi\|} \right]^2 \right\} \qquad (9) \\
&= \|\Psi\|^2 \min_z \left\{ 1 - \cos^2(\theta) \right\} \\
&= \|\Psi\|^2 \min_z \left\{ \sin^2(\theta) \right\}, \qquad (10)
\end{aligned}$$

where

$$\|\Psi\|^2 = \sum_{i,j=1}^{N_x} \eta_i \eta_j k(x_i, x_j), \tag{11}$$

$$\|\phi(z)\|^2 = k(z,z), \tag{12}$$

and θ is the angle between $\phi(z)$ and Ψ. It is clear that the pre-image problem is to find a θ that is as close to 0 or π as possible, depending on the initial z. In the case of $\theta \to \pi$, this will result in solutions where $\beta < 0$, to flip $\beta\phi(z)$ to the other hyper-hemisphere containing Ψ. Equation 9 was also derived in Schölkopf et al. (1999) and Schölkopf and Smola (2002) using an orthogonal projection argument.

When using the pre-image algorithm for the reduced set, the first reduced set vector z_1 can be found by setting $\Psi = \sum_{j=1}^{N_x} \alpha_j y_j \phi(x_j)$. This approximation is usually not good enough and needs to be augmented with further reduced set vectors. Thus, subsequent z_i $(i > 1)$ can be found by setting Ψ to be the residual:

$$\Psi_i = \sum_{j=1}^{N_x} \alpha_j y_j \phi(x_j) - \sum_{j=1}^{i-1} \beta_j \phi(z_j). \tag{13}$$

Burges (1996) used a gradient descent method to minimize ρ over z and β (7). In his algorithm, there is a singularity when β approaches zero, so he devised a method to avoid this singularity. In our formulation of ρ (Equation 9), β is not a control parameter, so this singularity is avoided.

3.3. Differential Evolution

The pre-image problem given in (9) is a nonlinear optimization problem, and thus optimization algorithms can become trapped in a shallow local minimium. To alleviate the local minimum problem, we propose using differential evolution (DE) (Storn & Price, 1997), an algorithm similar to genetic algorithms, but for optimization on a continuous domain. We found DE combined with gradient descent to be more robust and computationally more efficient.

DE utilizes a population of N_p d-dimensional vectors $z_{i,g}$, $1 \le i \le N_p$ for each generation g. The initial generation, $z_{i,0}$, is filled with uniformly-distributed random numbers. At every iteration of the algorithm,

each vector in turn is modified using crossover data from three other random vectors from that generation, and if the resulting disturbed vector results in a lower objective function ρ, then it is kept, otherwise it is left unmodified.

The algorithm for modifying a member of the population z_i is as follows. First, three mutually different integers $r_1, r_2, r_3 \in [1, N_p]$ are randomly drawn, representing the indices of three other vectors that will be used to modify z_i, and a temporary vector v is computed as a combination of those three:

$$v = z_{r_1,g} + F \cdot (z_{r_2,g} - z_{r_3,g}),$$

where $F \in (0, 2]$ is a parameter to the algorithm.

Second, two integers are chosen to represent the position and length of the crossover segment: the starting position n_0 is a random integer chosen uniformly from $[1, d]$ (d being the dimension of z), while the length L is chosen from a distribution that favors smaller values. The suggestion in (Storn & Price, 1997) is to choose L such that the probability $Pr(L > v) = R^{v-1}$, for some algorithm parameter $0 \leq R \leq 1$.

The ith offspring $u_{i,g+1}$ is copied from $z_{i,g}$, except that the L elements beginning with n_0 are copied from the temporary vector v instead. Note that the crossover segment is allowed to wrap around when $n_0 + L - 1$ goes beyond the dimension of the vector. Formally:

$$u_{i,g+1} = \begin{cases} v[j] & : \quad j \in [n_0, \cdots, n_0 + L - 1] \\ z_{i,g}[j] & : \quad \text{for all other} \quad j \end{cases}$$

The objective function ρ is then evaluated on $u_{i,g+1}$, and the next generation $z_{i,g+1}$ is determined by allowing only those vectors that improved according to this criteria to continue on:

$$z_{i,g+1} = \begin{cases} u_{i,g+1} & \text{if} \quad \rho(u_{i,g+1}) \leq \rho(z_{i,g+1}) \\ z_{i,g} & \text{if} \quad \rho(u_{i,g+1}) > \rho(z_{i,g+1}) \end{cases} \tag{14}$$

There are three parameters for DE: N_p, F, and R. In the experiments presented in Section 5, the parameter values are: $N_p = 5 * d$, $F = 0.8$, and $R = 0.95$, as suggested by Storn and Price (1997). We did a limited sensitivity study, and found that the algorithm is not very sensitive to these parameters.

When applying DE to the pre-image problem, we found that filling the first generation $z_{i,0}$ with random numbers, as suggested by Storn and Price, does not work, as those $z_{i,0}$ are so far from Ψ that numerically $\rho(z_i) = \|\Psi\|^2$ uniformly among the population (Equation 9). When this happens, DE has no way to compare the objective function values among members. To overcome this, we chose $z_{i,0}$ randomly from already calculated reduced set vectors (Equation 13), support vectors, training inputs, and even other unlabeled input vectors.

After N_{DE} generations with DE, we drive the DE solution lower by using a gradient descent method , just as in Burges' method, starting twice from the two best DE solutions. The gradient descent method we use is function $fmin\_bfgs$ in SciPy (Jones et al., 2001), a quasi-Newton method implemented in Python. It is our observation that to a certain extent, the larger N_{DE} is, the better the DE solution becomes for being used as an initial guess of gradient descent. However, large N_{DE} slows down the reduced set construction. In this paper, we typically used $N_{DE} = 100$.

3.4. Re-training on the Reduced Set

After the reduced set is constructed, the equation to determine the binary classification of a query example q is:

$$f(q) = \text{sign}\left[\sum_{i=1}^{N_z} \beta_i \cdot k(q, z_i) + b\right] \tag{15}$$

Burges (1996) proposed using the β_i values computed during the construction of the reduced-set, but computing the optimal value of b to minimize the training error. Instead, we suggest that after computing the reduced-set vectors, we recompute these coefficients so as to minimize the original SVM objective function.

In the derivation from the SVM primal equation (1) to the SVM dual equation (2), if we demand

$$\mathbf{w} = \sum_{i}^{N_z} \beta_i \phi(z_i), \tag{16}$$

we then arrive at a modified dual equation

$$\max_{\alpha} \left\{ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \widetilde{k}_{ij} \right\}, \tag{17}$$

where $\widetilde{k}_{ij}$ are elements of $\widetilde{\mathbf{K}}$. $\widetilde{\mathbf{K}}$ is defined as

$$\widetilde{\mathbf{K}} = \mathbf{K}_{xz} \mathbf{K}_{zz}^{-1} \mathbf{K}_{xz}^T, \tag{18}$$

where $\{\mathbf{K}_{xz}\}_{ij} = k(x_i, z_j)$.

Specifically, we run a SVM training program with $\widetilde{\mathbf{K}}$ as the kernel matrix, giving us a solution vector α. Then the adjusted β_i can be calculated as:

$$\boldsymbol{\beta} = \boldsymbol{\alpha} \boldsymbol{y} \mathbf{K}_{xz} \mathbf{K}_{zz}^{-1}. \tag{19}$$

We found that retraining for β_i usually results in significantly better performance for reduced-set SVMs, as will be seen in section 5. In addition, retraining is the key to our proposed multiclass method.

Note that our proposed retraining method requires taking the inverse of the kernel matrix. In practice, this matrix will sometimes turn out to be singular, in which case it is necessary to take a pseudo-inverse.

4. Reducing Multi-class SVMs

Methods for solving multi-class problems using binary SVMs include one-vs-one, one-vs-all, error-correcting codes, directed acyclic graph, and pairwise coupling (Hsu & Lin, 2002). Our proposed algorithm works with *any* of these methods, although in our experiments we focus on the most popular one-vs-one and one-vs-all.

Suppose that we have some multiclass method for k classes using l binary SVMs. The key innovation of our approach is sharing reduced-set vectors between the binary classifiers. To evaluate a new query q, the output of Equation 15 must be computed for each of the binary SVMs. The most time-consuming part of this computation is computing the kernel function $k(q, z_i)$ for each reduced-set vector z_i from any of the binary SVMs. Except for very low-dimensional problems, the time of multiplying by β_i, summing, and adding b is negligible in comparison. Thus, in effect we say that there is no added cost if the same vector z_i is used in more than one of the l binary SVMs; we can compute $k(q, z_i)$ once and use it in all of the computations.

Note that our proposed technique for retraining β_i and b for the binary reduced-set SVM works regardless of the source of the vectors z_i. Thus, adding additional vectors from other SVMs could not hurt the accuracy, because the retraining is optimal with respect to the original SVM problem, and having more vectors to choose from can only help. Thus, our proposed algorithm to reduce a multiclass SVM is:

1. Begin by creating a single reduced-set vector for each of the l binary SVMs independently.
2. Combine all reduced-set vectors into a single master list and retrain the β_i and b for each binary SVM using all l reduced-set vectors.
3. While more vectors are desired:
   (a) Use a heuristic to determine which binary SVM to improve. Compute a new reduced-set vector for this binary SVM.
   (b) Share all reduced-set vectors and retrain β_i and b for all binary SVMs again.

As the heuristic to determine the binary SVM to receive the next vector, we simply choose the SVM with the lowest binary accuracy. However, one could imagine a slower solution in which a vector is computed for each binary SVM and the one which reduces the overall multiclass error the most is the one chosen.

5. Experiments

We performed three sets of experiments to highlight the performance of our proposed multiclass reduced-set method. In the first experiments, we reproduced the results of Schölkopf et al. (1999) and compared the performance of our method directly against theirs. In the second experiments, we considered five common problems from the UCI Machine Learning Repository (Newman et al., 1998) to show how successful this technique is on a wide variety of problems. Finally, we did an additional experiment to try to quantify the benefit provided by using Differential Evolution, which is an issue of robustness and speed and not accuracy.

5.1. USPS Experiments

In order to compare our new method directly against previously published results, we focused on the USPS database of handwritten digits. The original database consists of 7291 training examples and 2007 test examples, all images of size 16×16. As described in Schölkopf et al. (1999), we used the smoothed USPS data (using a Gaussian kernel of width $\sigma = 0.75$) and followed the procedure in (Schölkopf et al., 1995) to extract a special set of 3000 training examples from the original 7291. Specifically, a multiclass SVM (one-vs-all) was trained on all 7291 training examples, resulting in 1618 of those examples being chosen as support vectors. These 1618 examples were augmented with 1382 other random vectors from the training set, to make a total of 3000. Using the Gaussian kernel $k(x, y) = \exp(-||x-y||^2/(0.5 \cdot 16^2))$ and regularization parameter $C = 10$, we trained a one-vs-all classifier with a 4.4% error rate on the test set and an average of 262 support vectors per class, matching (Schölkopf et al., 1999).

Table 1 compares the Burges method (as implemented by Schölkopf et al.) to our new reduced-set method. While Schölkopf presents results based on the number of reduced-set vectors per class, we present these results relative to the total number of reduced-set vectors in the multi-class classifier, since in our algorithm the same vector can be shared by multiple binary SVMs.

#SVs	Burges	Multiclass Reduced-set
20	–	**8.4%**
50	–	**6.0%**
100	7.1%	**5.3%**
150	6.4%	**5.0%**
200	5.6%	**4.9%**
250	5.1%	**4.7%**
500	5.0%	**4.6%**

Table 1. Percentage of multi-class test errors on the USPS digit recognition task, as a function of the total number of reduced-set vectors. The unreduced SVM had an error rate of 4.4%. Comparison between the Burges method, as reported in (Schölkopf et al., 1999), and our proposed multiclass reduced-set method ("Shared").

The table shows that our algorithm consistently outperforms Burges' method. Notably, it achieves the same level of accuracy (5.0%) after less than a third as many reduced-set vectors (150 compared with 500).

As an illustration of how much the greedy heuristic is affecting the results, the following table shows the number of times that our multiclass reduced-set algorithm added a new vector to each of the 10 classifiers in the USPS digit classification problem after 500 iterations, corresponding to the last line of Table 1:

'0'	**'1'**	**'2'**	**'3'**	**'4'**	**'5'**	**'6'**	**'7'**	**'8'**	**'9'**
9	1	79	75	100	28	15	118	52	23

Note that all 10 classifiers use all 500 vectors, just with different weights.

5.2. UCI Experiments

Table 2 shows the results of our comprehensive experiments on five data sets from the UCI machine learning repository (Newman et al., 1998), and Figure 1 shows the error curves for one problem in particular. We used the data sets and hyperparameters from Duan and Keerthi (2005), a recent paper that compared several multiclass methods and suggested a reproducible way to compute the kernel and C for each experiment. Each experiment was performed 20 times with different splits of the data into training and test sets each time; the training set sizes were the "large" sizes from Duan's paper. The data sets, and number of classes k, dimensionality d, and number of training examples n we used were ABE (the letters A, B, and E only from the "letter" data set; $k = 3$, $d = 16$, $n = 1,120$), DNA ($k = 3$, $d = 180$, $n = 1,000$), WAV ("waveform"; $k = 3$, $d = 21$, $n = 600$), SAT ("satimage"; $k = 6$, $d = 36$, $n = 2,000$), and SEG ("segment"; $k = 7$, $d = 18$, $n = 1,000$). We repeated all experiments with both one-vs-all (winner take all) and one-vs-one (max wins) classifiers.

For each experiment, we show the test error resulting from four different multiclass reduced-set approaches. The first column, labeled, "Orig", shows the original method as described in other papers, with the same number of reduced-set vectors in each binary SVM. "Greedy" adds one reduced-set vector at a time to the binary SVM with the largest error. "Retrain" then retrains for the optimal β and b after reduced-set selection. Finally, "Shared" shares reduced-set vectors between multiple binary SVMs before retraining. In doing these experiments, we occasionally had problems with the retraining algorithm (used in both "Retrain" and "Shared") failing to converge in a reasonable period of time. When this happened, we used the β values from this partially-converged solution but computed the bias b to minimize the training error, which was better than using the partially-converged b.

In almost every case, we found that each of our proposed enhancements provided a measurable increase in the performance of the reduced-set method. Our proposed "Shared" method was dominant in almost every case, though there were a couple of exceptions. The most improvement was seen on the ABE and SAT data sets using one-vs-all. For example after 20 reduced-set vectors, using "Shared" instead of "Orig" on ABE lowers the test error from 10.9% to 3.4%. Both DNA and WAV were surprisingly easy to reduce by all four methods; all four achieved the same performance as the original SVM after fewer than 20 vectors. Finally, note that in some cases, one-vs-all achieved lower error more quickly (e.g., SAT), while in other cases, one-vs-one achieved faster convergence (e.g., SEG). Thus for best results we recommend considering both multiclass methods (if not others) and choosing the one with the best reduced-set performance.

For some data sets (e.g., ABE, SAT), there are degradations of a few percentage points in accuracy from the full multiclass SVMs to various forms of reduced-set SVMs, even when the number of reduced vectors reaches 20 or 40, as shown in Table 2. We comment that with 20 reduced vectors, the degradations for reduced-set binary SVMs are usually small, almost always less than 1%, but those binary degradations sometimes aggregate to more significant ones for multiclass SVMs.

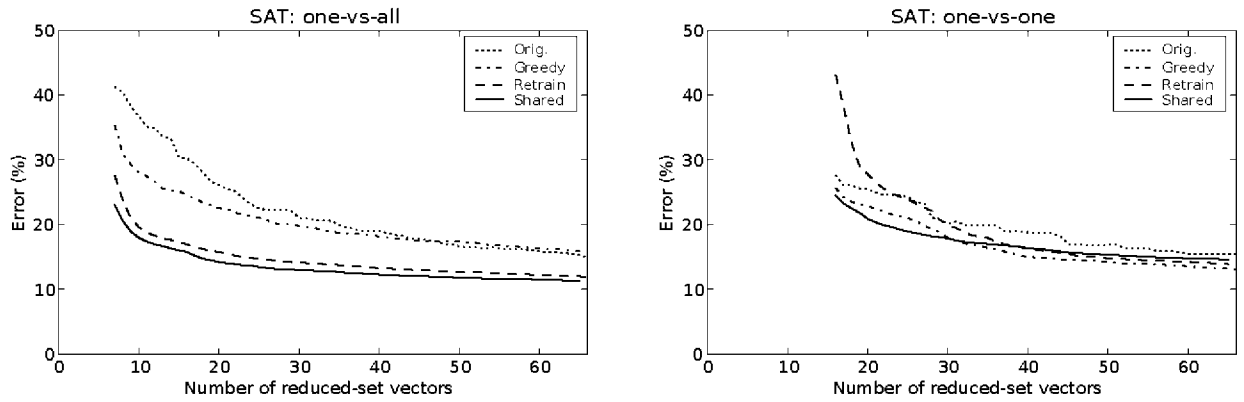

Figure 1. Test error as a function of the number of reduced-set vectors for the SAT (satimage) data set, with a one-vs-all SVM (left) or one-vs-one SVM (right), for four different variations of the reduced-set algorithm. Averages over 20 trials with different training/test splits.

5.3. DE Experiment

In our previous experiments, we ran DE for 100 generations, followed by gradient descent twice from the two best DE solutions. The following table compares this to doing just gradient descent with between 1 and 40 random restarts, using the SAT data set, one-vs-all, and our "Shared" method (numbers shown are the test error percentage):

nSV	1	10	20	40	DE
10	25.5	19.0	22.8	17.5	18.3
20	16.2	14.0	13.7	13.0	13.5
40	14.2	12.4	12.7	12.5	12.4

Note that gradient descent is less accurate than DE until about 40 restarts. While the exact timing depends on the details of the implementation, we found that, due to a better first guess, DE of 100 generation followed by gradient descent takes about the same amount of time as gradient descent (with no restarts) alone. Thus we found DE to speed up the pre-image optimization significantly, however for those who are not interested in implementing it we strongly suggest using at least 40 restarts of gradient descent.

6. Conclusions and Future Work

We have presented several improvements to SVM reduced-set algorithms which, when used together, result in significantly improved accuracy for multiclass reduced-set SVMs. Our core enhancements are (1) greedily choosing the binary SVM with the lowest accuracy to receive the next reduced-set vector, (2) retraining each binary SVM using the original SVM objective function to obtain optimal weights, and (3) sharing reduced-set vectors between multiple component binary SVMs in a multiclass SVM for additional gains. In addition, we have also presented a new derivation of the pre-image problem that avoids the singularity, and proposed differential evolution (DE) as a faster and more robust way to compute the pre-images. Our experimental results have shown conclusively that our enhancements are quite successful on six different benchmark problems.

There are several avenues for future study. We would like to investigate optimizing multiple reduced set vectors simultaneously, which we think will lead to faster convergence. With our current method, the greedy heuristic used to select the next binary SVM could be improved. The problem that the retraining algorithm occasionally fails to converge quickly needs to be further studied. Finally, the idea of finding the optimal weights for a set of vectors which are not the original training vectors could possibly be applied to applications other than reduced sets.

References

Bakir, G. H., Weston, J., & Schlkopf, B. (2004). Learning to find pre-images. *Advances in Neural Information Processing Systems, 16*, 449–456.

Burges, C. J. C. (1996). Simplified support vector decision rules. *International Conference on Machine Learning* (pp. 71–77).

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*, 121–167.

Burges, C. J. C., & Schölkopf, B. (1996). Improving

Data	SVs	Orig	Greedy	Retrain	Shared
	5	26.1	20.1	19.6	**17.2**
ABE	10	15.1	13.2	9.0	**8.7**
$k = 3$	20	10.9	6.6	3.6	**3.4**
	347	unreduced error: 0.6%			
	5	**6.0**	6.5	10.9	8.2
DNA	10	6.3	6.3	**6.0**	6.1
$k = 3$	20	6.0	6.2	**5.9**	6.0
	642	unreduced error: 5.8%			
	5	18.0	15.0	14.5	**14.1**
WAV	10	15.4	14.7	14.3	**14.1**
$k = 3$	20	14.2	14.5	14.1	**14.0**
	455	unreduced error: 14.2%			
	10	36.7	28.0	19.6	**17.9**
SAT	20	26.1	22.5	15.7	**14.2**
$k = 6$	40	19.0	18.1	13.2	**12.3**
	1106	unreduced error: 9.3%			
	10	34.2	31.9	24.8	**22.9**
SEG	20	31.9	22.2	16.8	**13.6**
$k = 7$	40	23.8	14.5	8.5	**8.1**
	293	unreduced error: 5.0%			

Data	SVs	Orig	Greedy	Retrain	Shared
	5	24.5	14.8	18.3	**12.3**
ABE	10	11.2	8.3	7.7	**6.2**
$k = 3$	20	7.4	3.2	3.0	**2.9**
	277	unreduced error: 0.5%			
	5	6.8	6.7	6.4	**6.1**
DNA	10	6.5	6.4	6.1	**6.0**
$k = 3$	20	6.5	6.5	**6.0**	**6.0**
	654	unreduced error: 5.9%			
	5	15.2	15.3	15.5	**14.8**
WAV	10	14.8	14.8	14.7	**14.5**
$k = 3$	20	14.8	14.9	14.7	**14.6**
	370	unreduced error: 14.7%			
	20	25.5	23.0	27.8	**20.9**
SAT	40	18.8	**15.0**	16.3	16.3
$k = 6$	60	15.5	**13.6**	14.2	14.8
	1039	unreduced error: 9.7%			
	25	18.1	12.3	11.7	**5.2**
SEG	50	13.9	5.6	5.0	**4.2**
$k = 7$	75	8.6	5.0	4.7	**4.2**
	201	unreduced error: 4.4%			

Table 2. Results of our experiments on four different reduced-set methods on five benchmark problems. For one-vs-all SVMs (left), our proposed "Shared" algorithm clearly dominates all other approaches on all problems except for DNA. For one-vs-one SVMs (right), "Shared" clearly dominates except for on SAT.

the accuracy and speed of support vector machines. *NIPS* (pp. 375–381).

DeCoste, D., & Mazzoni, D. (2003). Fast query-optimized kernel machine classification via incremental approximate nearest support vectors. *Proc. of the 20th International Conference on Machine Learning*. Washington, DC.

Duan, K., & Keerthi, S. S. (2005). Which is the best multiclass SVM method? An empirical study. *Proc. Multiple Classifier Systems* (pp. 278–285).

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multi-class support vector machines. *IEEE Trans. on Neural Networks*, *13*, 415–425.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python.

Kwok, J. T., & Tsang, I. W. (2004). The pre-image problem in kernel methods. *IEEE Trans. on Neural Networks*, *15*, 1517–1525.

Newman, D., Hettich, S., Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Platt, J. (1999). *Advances in kernel methods: Support vector learning*, chapter Fast training of support vector machines using sequential minimal optimization, 255 – 268. Cambridge, MA: The MIT Press.

Schölkopf, B., Burges, C. J., & Vapnik, V. (1995). Extracting support data for a given task. *Proc. 1st International Conference on Knowledge Discovery and Data Mining* (pp. 252 – 257). Menlo Park, CA.

Schölkopf, B., Mika, S., Burges, C. J., Knirsch, P., Müller, K.-R., Rätsch, G., & Smola, A. J. (1999). Input space versus feature space in kernel-based methods. *IEEE Trans. on Neural Networks*, *10*.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge, MA: The MIT Press.

Storn, R., & Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341–359.