SVM Incremental Learning, Adaptation and Optimization

Christopher P. Diehl
Applied Physics Laboratory
Johns Hopkins University
Laurel, MD 20723
Chris.Diehl@jhuapl.edu

Gert Cauwenberghs
ECE Department
Johns Hopkins University
Baltimore, MD 21218
gert@jhu.edu

***Abstract*— The objective of machine learning is to identify a model that yields good generalization performance. This involves repeatedly selecting a hypothesis class, searching the hypothesis class by minimizing a given objective function over the model's parameter space, and evaluating the generalization performance of the resulting model. This search can be computationally intensive as training data continuously arrives, or as one needs to tune hyperparameters in the hypothesis class and the objective function. In this paper, we present a framework for exact incremental learning and adaptation of support vector machine (SVM) classifiers. The approach is general and allows one to learn and unlearn individual or multiple examples, adapt the current SVM to changes in regularization and kernel parameters, and evaluate generalization performance through exact leave-one-out error estimation.**

I. Introduction

SVM techniques for classification and regression provide powerful tools for learning models that generalize well even in sparse, high dimensional settings. Their success can be attributed to Vapnik's seminal work in statistical learning theory [15] which provided key insights into the factors affecting generalization performance. SVM learning can be viewed as a practical implementation of Vapnik's *structural risk minimization* induction principle which involves searching over hypothesis classes of varying capacity to find the model with the best generalization performance.

SVM classifiers of the form $f(\mathbf{x}) = \mathbf{w}\cdot\Phi(\mathbf{x})+b$ are learned from the data $\{(\mathbf{x}_i, y_i) \in \mathbb{R}^m \times \{-1,1\}\ \forall\ i \in \{1,\ldots,N\}\}$ by minimizing

$$\min_{\mathbf{w},b,\boldsymbol{\xi}}\ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i^p \tag{1}$$

for $p \in \{1,2\}$ subject to the constraints

$$y_i\left(\mathbf{w}\cdot\Phi(\mathbf{x}_i)+b\right) \geq 1-\xi_i,\ \xi_i \geq 0\ \forall\ i \in \{1,\ldots,N\}. \tag{2}$$

We will focus on the $p=1$ case which is generally preferred due to the improved robustness to outliers offered by the hinge loss ($p=1$) over the quadratic loss ($p=2$). To simplify matters when learning nonlinear SVMs, this quadratic program is typically expressed in its dual form

$$\min_{0\leq\alpha_i\leq C} W = \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i Q_{ij}\alpha_j - \sum_{i=1}^{N}\alpha_i + b\sum_{i=1}^{N} y_i\alpha_i \tag{3}$$

with Lagrange multiplier (and offset) b and $Q_{ij} = y_i y_j \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$. The resulting dual form of the SVM is then $f(\mathbf{x}) = \sum_{i=1}^{N} y_i\alpha_i\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x})+b$. Given the nonlinearly transformed (training and test) examples $\Phi(\mathbf{x})$ appear only in dot product terms $\Phi(\mathbf{x})\cdot\Phi(\mathbf{y})$, we can employ a positive definite kernel function $K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x})\cdot\Phi(\mathbf{y})$ to implicitly map into a higher (possibly infinite) dimensional feature space and compute the dot product.

The search for a SVM that generalizes well involves repeatedly selecting a kernel function and regularization parameter C, solving the quadratic program, and evaluating generalization performance. Typically a parameterized kernel function is utilized that varies smoothly with respect to one or more continuous parameters $\boldsymbol{\sigma}$. Therefore the search task becomes one of finding a set of parameters $(C,\boldsymbol{\sigma})$ that maximize generalization performance.

As we incrementally vary the regularization and kernel parameters, we expect that the resulting SVM will not change dramatically. To minimize the computational burden of this search, we require a process that utilizes the current SVM solution to simplify the solution of the next quadratic program in the search. As new data becomes available, we would also like to integrate these examples into the quadratic program and modify $(C,\boldsymbol{\sigma})$ as necessary. This is a critical capability for online, active and incremental batch learning scenarios.

Incremental techniques have been developed to facilitate batch SVM learning over very large data sets, and have found widespread use in the SVM community, *e.g.* [12], [13], [8], [9]. Incremental SVM learning is particularly attractive in an on-line setting, and for active learning [3]. Most of these techniques are approximate and require several passes through the data to reach convergence. A procedure for exact "adiabatic" incremental learning of SVM classifiers, in a single pass through the data, was introduced in [4] and extended to SVM regression [11] and larger-set increments [7]. Investigations in incremental learning have largely focused on on-line learning as opposed to the larger model selection problem. Dynamic kernel adaptation is presented in [6], and approximate procedures for model selection, using leave-one-out (LOO) approximations and bounds, are given in [10] and [5]. The incremental SVM learning procedure can be adiabatically reverted to perform decremental unlearning, for exact LOO-based model selection [4].

In this paper, we extend the incremental SVM learning paradigm of [4] to a general framework for incremental learning, adaptation and optimization that allows one to learn and unlearn individual or multiple examples, adapt the current

SVM to changes in regularization and kernel parameters, and evaluate generalization performance through exact leave-one-out error estimation. In the following section, we begin by addressing the problem of learning the SVM solution for $N+M$ training examples given the SVM solution for N training examples and a new batch of M training examples where $M \geq 1$. Then we illustrate how the basic approach to this problem can be utilized to achieve our remaining objectives. In the final section, we examine some initial experimental results demonstrating the potential of this approach.

II. Incremental/Decremental Learning

A. Karush-Kuhn-Tucker Conditions

The *Karush-Kuhn-Tucker* (KKT) conditions uniquely define the solution of dual parameters $\{\boldsymbol{\alpha}, b\}$ minimizing the form (3):

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_{j=1}^{N} Q_{ij}\alpha_j + y_i b - 1 \begin{cases} > 0 & \alpha_i = 0 \\ = 0 & 0 \leq \alpha_i \leq C \\ < 0 & \alpha_i = C \end{cases} \quad (4)$$

$$h = \frac{\partial W}{\partial b} = \sum_{j=1}^{N} y_j \alpha_j \equiv 0 \quad (5)$$

Based on the partial derivatives g_i, the training examples can be partitioned into three different categories: the set $\mathcal{S}$ of *margin support vectors* on the margin ($g_i = 0$), the set $\mathcal{E}$ of *error support vectors* violating the margin ($g_i < 0$), and the remaining set $\mathcal{R}$ of *reserve vectors* exceeding the margin ($g_i > 0$). During incremental learning, new training examples with $g_i > 0$ are assigned directly to $\mathcal{R}$, as they by construction do not enter the solution. All other new examples are initially elements of a specially designated set $\mathcal{U}$ of *unlearned vectors*, and eventually become margin or error support vectors.

B. Adiabatic Increments

As we increment the unlearned example(s) into the solution, the goal will be to simultaneously preserve the KKT conditions for all previously seen training data. The KKT conditions are maintained by varying the margin vector coefficients in response to the perturbation imparted by the incremented new coefficient(s). In the process, elements of the different categories may change state, so that the incremental learning proceeds through a sequence of "adiabatic" steps, of amplitude determined by "bookkeeping" of category membership as given by conditions (4).

Prior to a given perturbation of the SVM solution, the partial derivatives with respect to $\{\alpha_i, b : \ \forall\ i \in \mathcal{S}\}$ equal

$$g_i = \sum_j Q_{ij}\alpha_j + y_i b - 1 = 0 \ \ \forall\ i \in \mathcal{S} \quad (6)$$

$$h = \sum_j y_j \alpha_j = 0. \quad (7)$$

Following the perturbation, the partial derivatives become

$$g_i = \sum_j Q_{ij}\alpha_j + \sum_{k\in\mathcal{S}} Q_{ik}\Delta\alpha_k + \sum_{l\in\mathcal{U}} Q_{il}\Delta\alpha_l + y_i(b+\Delta b) - 1 = 0 \ \ \forall\ i \in \mathcal{S} \quad (8)$$

$$h = \sum_j y_j \alpha_j + \sum_{k\in\mathcal{S}} y_k \Delta\alpha_k + \sum_{l\in\mathcal{U}} y_l \Delta\alpha_l = 0. \quad (9)$$

Expressing these conditions differentially, we obtain

$$\Delta g_i = \sum_{k\in\mathcal{S}} Q_{ik}\Delta\alpha_k + \sum_{l\in\mathcal{U}} Q_{il}\Delta\alpha_l + y_i \Delta b = 0 \ \ \forall\ i \in \mathcal{S} \quad (10)$$

$$\Delta h = \sum_{k\in\mathcal{S}} y_k \Delta\alpha_k + \sum_{l\in\mathcal{U}} y_l \Delta\alpha_l = 0. \quad (11)$$

For a given perturbation of the unlearned vector coefficients $\{\Delta\alpha_l : \forall\ l \in \mathcal{U}\}$, our objective is to determine the necessary changes in the margin vector coefficients $\{\Delta\alpha_k : \forall\ k \in \mathcal{S}\}$ and the bias Δb that preserve the KKT conditions on all learned data. The overall perturbation process is controlled by a perturbation parameter p which varies from 0 to 1 as the SVM solution is perturbed from its initial "unlearned" to final "learned" result. At $p = 0$, the solution initializes to the previous solution, prior to presentation of the new examples. During each perturbation, the perturbation parameter p is incremented by the smallest value $\Delta p_{\min}$ which leads to a category change for at least one example. By $p = 1$, all unlearned vectors have reached either of the three categories $\mathcal{S}$, $\mathcal{E}$ or $\mathcal{R}$, and both new and old data satisfy the KKT conditions (4) and (5).

The adiabatic increments $\Delta\alpha_i$'s are expressed as the product of Δp and the corresponding *coefficient sensitivities*. Let $\Delta\alpha_k = \beta_k \Delta p \ (k \in \mathcal{S}), \Delta\alpha_l = \lambda_l \Delta p \ (l \in \mathcal{U})$, and $\Delta b = \beta \Delta p$. Substituting these expressions into (10) and (11) and dividing through by Δp yields the differential KKT conditions expressed in terms of the coefficient sensitivities

$$\gamma_i = \frac{\Delta g_i}{\Delta p} = \sum_{k\in\mathcal{S}} Q_{ik}\beta_k + \sum_{l\in\mathcal{U}} Q_{il}\lambda_l + y_i\beta = 0 \ \ \forall\ i \in \mathcal{S} \quad (12)$$

$$\frac{\Delta h}{\Delta p} = \sum_{k\in\mathcal{S}} y_k \beta_k + \sum_{l\in\mathcal{U}} y_l \lambda_l = 0\ . \quad (13)$$

In principle, the *perturbation coefficients* λ_l can be freely chosen. Given the unlearned vector coefficients will change at most by C before the unlearned examples change categories, a natural choice is $\{\lambda_l = C : \forall\ l \in \mathcal{U}\}$. The corresponding coefficient sensitivities $\{\beta_k, \beta : \ \forall\ k \in \mathcal{S}\}$ are obtained by solving this system of equations. Once the coefficient sensitivities are known, we can compute the *margin sensitivities* γ_i for the error, reserve and unlearned vectors.

Both $\{\beta_k, \beta : \forall\ k \in \mathcal{S}\}$ and $\{\gamma_i : \forall\ i \in \mathcal{E}, \mathcal{R}, \mathcal{U}\}$ are necessary for computing $\Delta p_{\min}$. Table I lists the possible category changes that can occur during incremental/decremental learning. The smallest Δp among applicable conditions determines the category change and perturbation step $\Delta p_{\min}$. To determine $\Delta p_{\min}$, we first compute the minimum $\Delta p^c_{\min}$ for each set of examples that can undergo a given category change

TABLE I

SUMMARY OF BOOKKEEPING CONDITIONS

Common			
Initial Category	*New Category*	Δp	*Condition*
Margin	Reserve	$\frac{-\alpha_i}{\beta_i}$	$\beta_i < 0$
Error	Margin	$\frac{-g_i}{\gamma_i}$	$\gamma_i > 0$
Reserve	Margin	$\frac{-g_i}{\gamma_i}$	$\gamma_i < 0$
Incremental/Decremental Learning			
Margin	Error	$\frac{C-\alpha_i}{\beta_i}$	$\beta_i > 0$
Unlearned ($g_i < 0$)	Margin	$\frac{-g_i}{\gamma_i}$	$\gamma_i > 0$
Unlearned ($g_i < 0$)	Error	$\frac{C-\alpha_i}{\lambda_i}$	$\lambda_i > 0$
Regularization Parameter Perturbation			
Margin	Error	$\frac{C_t-\alpha_i}{\beta_i-\Delta C}$	$\beta_i > \Delta C$
Kernel Parameter Perturbation			
Margin	Error	$\frac{C-\alpha_i}{\beta_i}$	$\beta_i > 0$
Unlearned ($g_i \neq 0$)	Margin	$\frac{-g_i}{\gamma_i}$	$\gamma_i g_i < 0$
Unlearned ($g_i > 0$)	Reserve	$\frac{-\alpha_i}{\lambda_i}$	$\lambda_i < 0$
Unlearned ($g_i < 0$)	Error	$\frac{C-\alpha_i}{\lambda_i}$	$\lambda_i > 0$

c. Then $\Delta p_{\min} = \min_{c \in \mathcal{C}} \Delta p^c_{\min}$ where $\mathcal{C}$ is the set of possible category changes.

Once $\Delta p_{\min}$ is known, we update the coefficients for the margin vectors ($\alpha_k \rightarrow \alpha_k + \beta_k \Delta p : \forall\ k \in \mathcal{S}$) and unlearned vectors ($\alpha_l \rightarrow \alpha_l + \lambda_l \Delta p : \forall\ l \in \mathcal{U}$). After noting the category change, we recompute the coefficient and margin sensitivities and determine the next perturbation. This process repeats until $p = 1$.[1]

C. The Initial Perturbation

Special consideration must be given to the case where no SVM solution initially exists. In this scenario, all of the training examples are initially unlearned and $\{\alpha_l = 0, b = 0 : \forall\ l \in \mathcal{U}\}$. This implies the equality condition (5) is immediately satisfied. If we begin incrementing the unlearned vector coefficients, we will immediately violate this condition unless $\sum_{k \in \mathcal{U}} y_k = 0$ so that

$$h = \sum_j y_j \alpha_j + C \Delta p \sum_{k \in \mathcal{U}} y_k = 0. \tag{14}$$

Most often, there will not be an equal number of examples from each class, and the equality condition cannot generally be preserved.

The margin vector coefficients allow the preservation of the equality condition when an initial SVM solution is given. The margin vectors provide the degree of freedom to counterbalance the changes in the unlearned vector coefficients. One way to preserve condition (5) is to bootstrap the process by selecting one example from each class and learning an initial SVM. This is accomplished by first selecting one example and modifying only the bias so that this example is a margin vector. Then the second example from the opposite class can be successfully incremented into the solution.[2]

Another option is to simply proceed with the initial perturbation and disregard condition (5) until the margin vector set is no longer empty. Beyond that point, we can incrementally correct the violation in a manner that guarantees the condition will be satisfied once the perturbation process is complete. We will explore this approach in detail below when addressing the problem of adapting the SVM solution to changes in kernel parameters.

D. Solving for the Coefficient Sensitivities $\{\beta_k, \beta\}$

To obtain the coefficient sensitivities $\{\beta_k, \beta : \forall\ k \in \mathcal{S}\}$, we must solve the system of equations represented by (12) and (13). Expressed in matrix-vector form, we have

$$\mathbf{Q}\boldsymbol{\beta} = -\sum_{l \in \mathcal{U}} \lambda_l \mathbf{v}_l \tag{15}$$

where

$$\boldsymbol{\beta} = \begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_n} \end{bmatrix},\ \mathbf{v}_l = \begin{bmatrix} y_l \\ Q_{s_1 l} \\ \vdots \\ Q_{s_n l} \end{bmatrix}, \tag{16}$$

$n = |\mathcal{S}|$ and

$$\mathbf{Q} = \begin{bmatrix} 0 & y_{s_1} & \dots & y_{s_n} \\ y_{s_1} & Q_{s_1 s_1} & \dots & Q_{s_1 s_n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{s_n} & Q_{s_n s_1} & \dots & Q_{s_n s_n} \end{bmatrix}. \tag{17}$$

We require the inverse $\mathbf{R} = \mathbf{Q}^{-1}$ to compute the sensitivities

$$\boldsymbol{\beta} = -\sum_{l \in \mathcal{U}} \lambda_l \mathbf{R} \mathbf{v}_l. \tag{18}$$

As the perturbation process proceeds, we can easily adapt $\mathbf{R}$ as examples are added and removed from the margin vector set $\mathcal{S}$ [4]. When adding an example to $\mathcal{S}$, $\mathbf{R}$ expands as

$$\mathbf{R} \longleftarrow \begin{bmatrix} & & 0 \\ & \mathbf{R} & \vdots \\ & & 0 \\ 0 & \dots\ 0 & 0 \end{bmatrix} + \frac{1}{\gamma^{s_{n+1}}_{s_{n+1}}} \begin{bmatrix} \boldsymbol{\beta}_{s_{n+1}} \\ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_{s_{n+1}} \\ 1 \end{bmatrix}^{\mathrm{T}} \tag{19}$$

where $\boldsymbol{\beta}_{s_{n+1}} = -\mathbf{R}\mathbf{v}_{s_{n+1}}$ and $\gamma^{s_{n+1}}_{s_{n+1}} = Q_{s_{n+1} s_{n+1}} + \mathbf{v}^{\mathrm{T}}_{s_{n+1}} \boldsymbol{\beta}_{s_{n+1}}$. When removing a margin vector k from $\mathcal{S}$, $\mathbf{R}$ contracts as

$$R_{ij} \longleftarrow R_{ij} - \frac{R_{ik} R_{kj}}{R_{kk}} \quad \forall\ i, j \in \mathcal{S} \cup \{0\}; i, j \neq k \tag{20}$$

where index 0 refers to the b term.

[1] Note that it is possible for the unlearned vector set to become empty, and the incremental perturbation to terminate, when $p < 1$.

[2] There is the potential for a degenerate solution to arise if the regularization parameter C is set too low such that both examples become error vectors [2]. If this does occur, the problem can be easily corrected by increasing C.

E. Decremental Learning and Cross-Validation

The adiabatic increment process is fully reversible [4]. Decremental "unlearning" provides the flexibility to perform leave-one-out error estimation (or more generally k-fold cross-validation). This is accomplished simply by "unlearning" the relevant example(s) and computing the corresponding class label(s) using the resulting SVM.

When unlearning a subset $\mathcal{U}$ of the training examples, our objective is to decrement the coefficients $\{\alpha_l : \forall\ l \in \mathcal{U}\}$ to 0, while retaining the KKT conditions on all other data. Before we begin the perturbation process, if any of the examples in $\mathcal{U}$ are currently margin vectors, we must first contract the inverse $\mathbf{R}$ so that the corresponding KKT conditions of the unlearned vectors are no longer enforced. Once this is complete, we compute the coefficient sensitivities $\{\beta_k, \beta : \forall\ k \in \mathcal{S}\}$ and margin sensitivities $\{\gamma_i : \forall i \in \mathcal{E}, \mathcal{R}\}$ using equations (18) and (12) as before. In the decremental scenario, the coefficient sensitivities for the examples in $\mathcal{U}$ equal $\{\lambda_l = -\alpha_l : \forall\ l \in \mathcal{U}\}$. Note that when determining $\Delta p_{\min}$ for each perturbation, we only need to track category changes for the remaining margin, error and reserve vectors.

III. Parameter Optimization

The key condition that enables incremental learning when using the hinge loss is the fact that the partial derivatives g_i and h are linear functions of the parameters $\{\alpha_i, b\}$. This allows us to adiabatically perturb the SVM solution in the manner introduced in the previous section. In general, the SVM solution can be perturbed adiabatically with respect to any parameters that are linearly related to the partial derivatives g_i and h. This implies for example that the SVM can be perturbed adiabatically with respect to the regularization parameter C [14]. This also suggests that in general the SVM *cannot* be perturbed adiabatically with respect to nonlinear parameters in the kernel, for instance $\boldsymbol{\sigma}$ in a radial basis Gaussian kernel. Although this would seem to limit the utility of the algorithm for general parameter optimization, we will propose a slight modification to the overall strategy that allows us to address the nonlinear case as well.

A. Regularization Parameter Perturbation

Perturbing the SVM with respect to the regularization parameter C amounts to simply incrementing or decrementing the error vector coefficients $\{\alpha_l : \forall\ l \in \mathcal{E}\}$. By replacing the unlearned vector set $\mathcal{U}$ with the error vector set $\mathcal{E}$ in equations (8) through (13), we obtain the differential KKT conditions

$$\gamma_i = \frac{\Delta g_i}{\Delta p} = \sum_{k \in \mathcal{S}} Q_{ik}\beta_k + \sum_{l \in \mathcal{E}} Q_{il}\lambda_l + y_i\beta = 0 \quad \forall\ i \in \mathcal{S} \tag{21}$$

$$\frac{\Delta h}{\Delta p} = \sum_{k \in \mathcal{S}} y_k\beta_k + \sum_{l \in \mathcal{E}} y_l\lambda_l = 0. \tag{22}$$

This implies the coefficient sensitivities $\{\beta_k, \beta : \forall\ k \in \mathcal{S}\}$ equal

$$\boldsymbol{\beta} = -\sum_{l \in \mathcal{E}} \lambda_l \mathbf{R}\mathbf{v}_l. \tag{23}$$

TABLE II
Category reassignment following kernel parameter change

Previous Example Category	$g_i > 0$	$g_i < 0$
Reserve Vector ($\alpha_i = 0$)	Reserve	Unlearned
Margin Vector ($0 \le \alpha_i \le C$)	Unlearned	Unlearned
Error Vector ($\alpha_i = C$)	Unlearned	Error

Given the error vector coefficients will change at most by $\Delta C = C_{\text{new}} - C$, a natural choice for perturbation coefficients is now $\{\lambda_l = \Delta C : \forall\ l \in \mathcal{E}\}$.

The only subtle difference in the perturbation process arises when computing the Δp required for a margin vector to become an error vector. Unlike in incremental learning where C is fixed, C varies during the perturbations:

$$C_{t+1} = C_t + \Delta C \Delta p,\ C_0 = C. \tag{24}$$

If a margin vector becomes an error vector for some positive Δp, the following must hold:

$$\alpha_{s_k} + \beta_{s_k}\Delta p = C_t + \Delta C \Delta p. \tag{25}$$

This implies

$$\Delta p = \frac{C_t - \alpha_{s_k}}{\beta_{s_k} - \Delta C} \tag{26}$$

as indicated in table I.

B. Kernel Parameter Perturbation

To perturb the SVM solution as we vary the kernel parameters requires a slightly different strategy. Unlike incremental learning and regularization parameter perturbation, we cannot in general define a method for adiabatically perturbing the SVM solution while transforming the kernel. Therefore our approach will involve first modifying the kernel parameters and then incrementally correcting the previous solution until the KKT conditions are satisfied for the new kernel parameter settings.

Once the kernel parameters are changed, we begin by recomputing the partial derivatives g_i for the margin, error and reserve vectors. Then we reclassify the category of each example as necessary. Table II presents the mapping from old to new category as a function of g_i. For the reserve and error vectors with partial derivatives that do not change sign, no category change occurs. The α's for all of the other examples will need to be modified; therefore these examples are reclassified as unlearned.

During each perturbation when there are no margin vectors, only the unlearned vector coefficients change ($\alpha_i \rightarrow \alpha_i + \lambda_i \Delta p\ :\ \forall\ i \in \mathcal{U}$). Once there is at least one margin vector, we modify the margin vector coefficients $\{\alpha_i :\ \forall\ i \in \mathcal{S}\}$ and the bias b in a manner that preserves the KKT conditions for the current margin vector set $\mathcal{S}$. The perturbation coefficients λ_i for the unlearned vectors $\mathcal{U}$ are now defined as

$$\lambda_i = \begin{cases} -\alpha_i & g_i > 0 \\ C - \alpha_i & g_i < 0 \end{cases}. \tag{27}$$

For all examples in $\mathcal{U}$ with $g_i > 0$, α_i must be decremented until the example either becomes a margin vector ($g_i = 0$) or

reserve vector ($\alpha_i = 0$). For examples with $g_i < 0$, α_i must be incremented until the example either becomes a margin vector or error vector ($\alpha_i = C$).

Before the perturbation process begins, the KKT condition

$$h = \sum_j y_j \alpha_j = 0 \tag{28}$$

still holds. In order to preserve this condition, the constraint

$$\sum_{j \in \mathcal{U}} y_j \lambda_j = 0 \tag{29}$$

must be satisfied. In general, this constraint cannot always be satisfied, even by rescaling the perturbation coefficients in such a way that the signs of the coefficients are preserved. Therefore during the first perturbation we will intentionally violate the KKT condition and correct the violation incrementally once the margin vector set is no longer empty. We shall see next that this is easily resolved.

Prior to a given perturbation of the SVM solution when $|\mathcal{S}| > 0$, the partial derivatives with respect to $\{\alpha_i, b : \ \forall \ i \in \mathcal{S}\}$ equal

$$g_i = \sum_j Q_{ij} \alpha_j + y_i b - 1 = 0 \ \ \forall \ i \in \mathcal{S} \tag{30}$$

$$h = \sum_j y_j \alpha_j = \epsilon \tag{31}$$

Following the perturbation, the partial derivatives equal

$$g_i = \sum_j Q_{ij} \alpha_j + \sum_{k \in \mathcal{S}} Q_{ik} \beta_k \Delta p + \sum_{l \in \mathcal{U}} Q_{il} \lambda_l \Delta p + y_i (b + \beta \Delta p) - 1 = 0 \quad \forall \ i \in \mathcal{S} \tag{32}$$

$$h = \sum_j y_j \alpha_j + \sum_{k \in \mathcal{S}} y_k \beta_k \Delta p + \sum_{l \in \mathcal{U}} y_l \lambda_l \Delta p = \epsilon + \rho \Delta p \tag{33}$$

Before we can solve for the coefficient sensitivities, we must first define the constraint sensitivity $\rho = \frac{\Delta h}{\Delta p}$. Given we want $h = 0$ when the perturbation process is complete (when $p = 1$), $\Delta h = -\epsilon$ when $\Delta p = 1 - p$. Therefore $\rho = \frac{-\epsilon}{1-p}$.

Differencing the equations for g_i and h and dividing through by Δp leads to

$$\gamma_i = \frac{\Delta g_i}{\Delta p} = \sum_{k \in \mathcal{S}} Q_{ik} \beta_k + \sum_{l \in \mathcal{U}} Q_{il} \lambda_l + y_i \beta = 0 \ \ \forall \ i \in \mathcal{S} \tag{34}$$

$$\frac{\Delta h}{\Delta p} = \sum_{k \in \mathcal{S}} y_k \beta_k + \sum_{l \in \mathcal{U}} y_l \lambda_l = \rho. \tag{35}$$

Expressing this system of equations in matrix-vector form, we obtain

$$\mathbf{Q}\boldsymbol{\beta} = \rho \mathbf{e}_1 - \sum_{l \in \mathcal{U}} \lambda_l \mathbf{v}_l \tag{36}$$

where $\mathbf{e}_1 = [1 \ 0 \ \ldots \ 0]^{\mathrm{T}} \in \mathbb{R}^{|\mathcal{S}|+1}$. This implies the coefficient sensitivities $\{\beta_k, \beta : \forall \ k \in \mathcal{S}\}$ equal

$$\boldsymbol{\beta} = \rho \mathbf{R} \mathbf{e}_1 - \sum_{l \in \mathcal{U}} \lambda_l \mathbf{R} \mathbf{v}_l. \tag{37}$$

Comparing equation (18) with (37), we see that one additional term has been introduced to force $h \rightarrow 0$.

With the equations for the coefficient and margin sensitivities in hand, the perturbation process proceeds as before. Table I lists the possible category changes that must be tracked during the perturbation process. Note that the constraint sensitivity ρ in equation (37) must be updated prior to each perturbation to ensure that $h = 0$ when the final perturbation is complete.

IV. Experimental Results

In order to assess the benefits offered by the incremental framework, we conducted two experiments using the Pima Indians dataset from the UCI machine learning repository [1]. Using an RBF kernel $K(x, y) = \exp\left(-\frac{\|x-y\|^2}{\sigma^2}\right)$, we first fixed the kernel width and varied (increased or decreased) the regularization parameter over the range indicated in table III. Then we fixed the regularization parameter and varied (decreased or increased) the kernel width over the range indicated in table IV. Both tables list the number of floating point operations[3], kernel evaluations and perturbations required for both full retraining and the incremental approach. Note that the center row in each table does not list statistics for the incremental approach because the corresponding fully (re)trained SVM serves as the initial SVM for the series of perturbations. Full retraining corresponds to an incremental learning session over the entire training data, starting from the empty set.

Beginning with table III, we find that as the regularization parameter is increased, the computational savings offered by the incremental approach increases. There are two components to the overall cost in terms of floating point operations for the incremental approach that are worth noting. First, as C increases, the number of error vectors decreases. This leads to a steady decline in the number of kernel evaluations required. At the same time, the number of margin vectors is increasing which adds to the expense of the repeated computation of the margin sensitivities γ_i and leads to increases in the overall cost.

Examining table IV, we see that as the kernel width is decreased, the computational savings offered by the incremental approach decreases. This is due to the increasing number of margin vectors that result. Unlike in regularization parameter perturbation, cached rows of the kernel matrix do not provide any benefit when modifying the kernel width. We must instead recompute the needed rows of the kernel matrix as is the case in full retraining. Therefore as the relative fraction of unlearned vectors that result in margin vectors increases, the computational cost of the incremental approach becomes comparable to that of full retraining.

V. Conclusions

In this paper, we have presented a framework for incremental learning and adaptation of support vector machine

[3]The number of floating point operations was determined using the *flops* command in MATLAB.

TABLE III

COMPARISON OF THE COSTS OF FULL RETRAINING WITH THE INCREMENTAL APPROACH FOR FIXED σ^2 AND VARYING C

Pima Indians, $\sigma^2 = 4$									
				Floating Point Operations (x10^8)		*Kernel Evaluations* (x10^6)		*Perturbations*	
C	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	*Full Retraining*	*Incremental*	*Full Retraining*	*Incremental*	*Full Retraining*	*Incremental*
0.354	72	432	264	1.23	0.200	1.190	0.320	1061	58
0.5	97	401	270	1.46	0.250	1.200	0.296	1086	67
0.707	121	364	283	1.71	0.255	1.200	0.263	1119	60
1.0	150	328	290	1.12	-	0.965	-	825	-
1.41	188	284	296	2.37	0.403	1.210	0.294	1164	73
2.0	223	244	301	1.82	0.430	1.040	0.256	944	64
2.83	247	212	309	3.05	0.417	1.220	0.217	1198	53

TABLE IV

COMPARISON OF THE COSTS OF FULL RETRAINING WITH THE INCREMENTAL APPROACH FOR FIXED C AND VARYING σ^2

Pima Indians, $C = 1$											
				Initial	*Floating Point Operations* (x10^8)		*Kernel Evaluations* (x10^6)		*Perturbations*	
σ^2	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{U}\rvert$	*Full Retraining*	*Incremental*	*Full Retraining*	*Incremental*	*Full Retraining*	*Incremental*
1.41	347	260	161	500	2.690	2.94	1.004	1.000	730	721
2.0	269	286	213	483	1.980	2.07	0.991	0.975	774	740
2.83	204	299	265	442	1.520	1.49	0.981	0.944	813	760
4.0	150	328	290	-	1.120	-	0.965	-	825	-
5.66	109	341	318	205	0.852	0.488	0.952	0.533	832	286
8.0	81	356	331	154	0.700	0.347	0.943	0.479	836	224
11.3	51	370	347	120	0.602	0.267	0.924	0.446	818	197

classifiers that aims to simplify the model selection task by perturbing the SVM solution as the regularization and kernel parameters are adjusted. Empirical results on UCI benchmark data suggest regularization parameter perturbation can offer significant computational savings, whereas the computational benefits of kernel parameter perturbation may be more limited. In general, the benefits of incremental adaptation are most substantial for small perturbation of kernel or regularization parameters, as one may expect in each optimization step for model selection.

Acknowledgments

C. Diehl was supported by the FY03 JHU/APL IR&D program for Applied Mathematics. G. Cauwenberghs was supported by grant IIS-0209289 from the National Science Foundation and grant N00014-99-1-0612 from the Office of Naval Research.

REFERENCES

[1] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. University of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[2] Chris J. C. Burges and David J. Crisp. Uniqueness of the SVM solution. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*. Morgan Kaufmann, 2000.

[3] Colin Campbell, Nello Cristianini, and Alex Smola. Query learning with large margin classifiers. In *Proceedings, 17th International Conference on Machine Learning*, pages 111–118. Morgan Kaufmann, San Francisco, CA, 2000.

[4] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[5] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. In *Machine Learning*. Kluwer Academic, 2002.

[6] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. In *NeuroCOLT Technical Report NC-TR-98-017*. Royal Holloway College, University of London, UK, 1998.

[7] Shai Fine and Katya Scheinberg. Incremental learning and selective sampling via parametric optimization framework for SVM. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[8] T.-T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: A fast and simple learning procedure for support vector machines. In *Proceedings, 15th International Conference on Machine Learning*. Morgan Kaufmann, 1998.

[9] Thorsten Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[10] J.-H. Lee and C.-J. Lin. Automatic model selection for support vector machines. http://www.csie.ntu.edu.tw/˜cjlin/papers/ modelselect.ps.gz, 2000.

[11] M. Martin. On-line support vector machines for function approximation. http://www.lsi.upc.es/dept/techreps/html/R02-11.html, 2002.

[12] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings, 1997 IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, 1997.

[13] J.C. Platt. Fast training of support vector machines using sequential minimum optimization. In *Advances in Kernel Methods– Support Vector Learning*, pages 185–208. Cambridge MA: MIT Press, 1998.

[14] M. Pontil and A. Verri. Properties of support vector machines. In *Neural Computation*, volume 10, pages 955–974. MIT Press, 1997.

[15] Vladimir V. Vapnik. *Statistical Learning Theory*. Springer-Verlag New York, 1998.