# Incremental and Decremental Training for Linear Classification

Cheng-Hao Tsai
Dept. of Computer Science
National Taiwan Univ., Taiwan
r01922025@csie.ntu.edu.tw

Chieh-Yen Lin
Dept. of Computer Science
National Taiwan Univ., Taiwan
r01944006@csie.ntu.edu.tw

Chih-Jen Lin
Dept. of Computer Science
National Taiwan Univ., Taiwan
cjlin@csie.ntu.edu.tw

## ABSTRACT

In classification, if a small number of instances is added or removed, incremental and decremental techniques can be applied to quickly update the model. However, the design of incremental and decremental algorithms involves many considerations. In this paper, we focus on linear classifiers including logistic regression and linear SVM because of their simplicity over kernel or other methods. By applying a warm start strategy, we investigate issues such as using primal or dual formulation, choosing optimization methods, and creating practical implementations. Through theoretical analysis and practical experiments, we conclude that a warm start setting on a high-order optimization method for primal formulations is more suitable than others for incremental and decremental learning of linear classification.

## Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

## Keywords

warm start; incremental learning; decremental learning; linear classification

## 1. INTRODUCTION

In supervised learning, when a small amount of data is added or removed, the classification model may not change much. Therefore, incremental and decremental algorithms are useful to update the model without re-training the problem from scratch. However, the design of good incremental and decremental learning algorithms is never easy. First, they are often extended from a standard learning method, but the resulting procedure may become very complicated. Second, applying incremental and decremental algorithms is not guaranteed to be faster than re-training the data from scratch. Third, the scenarios to use incremental and decremental learning may significantly vary according to applications, so designing software for general use is difficult.

In this paper, we study incremental and decremental algorithms for logistic regression (LR) and linear support vector machine (SVM). The decision to work on linear rather than kernel classifiers comes from a long journey of attempting to support incremental and decremental learning in our SVM software LIBSVM [4]. Although many users have requested this function and some studies have been conducted, we were hampered by two difficulties. First, extending the commonly used decomposition methods to flexibly enlarge or shrink the cached kernel elements during training is complicated. Note that adding or removing data causes a "two-dimensional" change of rows and columns of the kernel matrix. Second, existing extensions of decomposition methods for incremental or decremental learning may not effectively reduce the running time. Recently, in contrast to using kernels, linear classification has been shown to give comparable accuracy on some applications (see, e.g., a survey in [19]). The popularity of linear classification has motivated us to study incremental and decremental learning again. However, the goal becomes to support this functionality in another software LIBLINEAR [8] for large-scale linear classification.

We point out that the difficulties occurred in incremental and decremental learning for kernel classifiers may be alleviated for linear classifiers because more optimization algorithms are applicable. For kernel classifiers, it is rather restrictive to design optimization algorithms because the model is represented by a linear combination of training instances. Therefore, the optimization problem must be designed to find coefficients for this linear combination. In contrast, as we will explain in Section 2, there are more options in solving optimization problems for linear classifiers. We show that this difference between linear and kernel classifiers strongly affects their extensions to incremental and decremental training.

In Section 3, we apply a warm start strategy for incremental and decremental training of linear classification. The optimal solution of training past data is modified as an initial solution for solving the new problem after data addition or removal. We consider in Section 4 three representative optimization algorithms for linear classification. They differ in several aspects such as solving primal or dual formulation, and using first-order (i.e., gradient) or second-order information of the optimization problem. Our main findings are that the warm start setting is in general more effective to improve the primal initial solution than the dual and that the warm start setting more effectively speeds up


*KDD'14,* August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623661.

a high-order optimization method than a low-order one. After addressing some implementation issues, we have successfully finished an extension of the software LIBLINEAR for incremental and decremental learning. It is available at `http://www.csie.ntu.edu.tw/~cjlin/papers/ws`.

This paper is organized as follows. In Section 2, we show the formulations of SVM and LR, and discuss existing methods for incremental and decremental learning. In Section 3, we propose a warm start setting, followed by analyzing the difference between using primal and dual forms. The comparison between high-order and low-order optimization methods is described in Section 4. In Section 5, we discuss implementation issues for building a solid tool. Experiments are presented in Section 6. Section 7 concludes our work. Supplementary materials with additional results and programs for experiments are also available at the above-mentioned web site.

# 2. SVM, LR, AND THEIR INCREMENTAL AND DECREMENTAL TRAINING

Assume we are given (label, feature-vector) pairs of training data $(y_i, \boldsymbol{x}_i) \in \{-1, 1\} \times R^n, i = 1, \ldots, l$. Linear classification involves the following optimization problem.

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) \text{ where } f(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum\nolimits_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i). \quad (1)$$

where $\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$ is the loss function and $C$ is a user-specified parameter. Common loss functions include

$$\xi_{\text{L1}}(\boldsymbol{w}; \boldsymbol{x}_i, y_i) \equiv \max(0, 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i), \quad (2)$$

$$\xi_{\text{L2}}(\boldsymbol{w}; \boldsymbol{x}_i, y_i) \equiv \max(0, 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i)^2, \quad \text{and} \quad (3)$$

$$\xi_{\text{LR}}(\boldsymbol{w}; \boldsymbol{x}_i, y_i) \equiv \log(1 + e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i}). \quad (4)$$

If (2) or (3) is used, then problem (1) is SVM [1, 7]. If (4) is used, then we have logistic regression. The three loss functions in (2)-(4) have different differentiability.

L1 loss: not differentiable
L2 loss: differentiable but not twice differentiable
LR loss: twice differentiable

Many convex optimization methods have been considered to find the optimal $\boldsymbol{w}$. Alternatively, from optimization theory or the representer theorem [12], the optimal $\boldsymbol{w}$ is a linear combination of training instances with coefficients $\boldsymbol{\alpha}$.

$$\boldsymbol{w} = \sum\nolimits_{i=1}^{l} y_i\alpha_i\boldsymbol{x}_i. \quad (5)$$

Then we can instead solve an optimization problem over $\boldsymbol{\alpha}$. A common way is to derive the dual problem of (1).

$$\max_{\boldsymbol{\alpha}} f^D(\boldsymbol{\alpha}) \equiv \sum_{i=1}^{l} h(\alpha_i, C) - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j K(i,j) - \sum_{i=1}^{l}\frac{\alpha_i^2}{2}d$$
$$\text{subject to} \quad 0 \le \alpha_i \le U, \forall i = 1, \ldots, l, \quad (6)$$

where $K(i,j) = y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j$ and

$$U = \begin{cases} C \\ \infty \end{cases} \quad d = \begin{cases} 0 & \text{for L1-loss SVM and LR,} \\ \frac{1}{2C} & \text{for L2-loss SVM.} \end{cases}$$

Further, $h(\alpha_i, C) =$

$$\begin{cases} \alpha_i & \text{for L1-loss and L2-loss SVM,} \\ C\log C - \alpha_i\log\alpha_i - (C-\alpha_i)\log(C-\alpha_i) & \text{for LR.} \end{cases}$$

We define $0\log 0 = 0$, so

$$h(0, C) = 0 \quad (7)$$

for all three losses. Besides (6), the dual problem can also be represented by the following matrix form.

$$\max_{\boldsymbol{\alpha}} \quad \sum\nolimits_{i=1}^{l} h(\alpha_i, C) - \frac{1}{2}\boldsymbol{\alpha}^T\bar{Q}\boldsymbol{\alpha}$$
$$\text{subject to} \quad 0 \le \alpha_i \le U, \forall i = 1, \ldots, l, \quad (8)$$

where $\bar{Q} = Q + D \in R^{l\times l}$, $Q_{ij} = K(i,j)$, and $D$ is diagonal with $D_{ii} = d, \forall i$. We refer to (1) as the primal problem.

We briefly discuss the difference between linear and kernel classifiers. A kernel classifier maps data to a high (maybe infinite) dimensional space, so problem (1) cannot be directly solved. Instead, by kernel tricks we solve an optimization problem of $\boldsymbol{\alpha}$ like (6).[1] In contrast, for linear classifiers, it is more flexible to design optimization algorithms because we can solve problems of $\boldsymbol{w}$ or $\boldsymbol{\alpha}$. We will show in Section 3 that this difference between linear and kernel classifiers has an impact on their incremental and decremental training.

## 2.1 Existing Methods

Most existing studies on incremental or decremental training focus on kernel rather than linear classifiers. By considering the dual problem of kernel SVM, [3] and subsequent works [9, 11, 13, 16] investigate how to maintain the optimality condition when data are slightly changed. Note that at an optimum, all $\alpha_i \in (0, U)$ correspond to the solution of a linear system. They propose methods to quickly identify these $\alpha_i$'s for the new data and efficiently solve the linear system. A kernel sub-matrix is required to be enlarged or shrunk. Careful implementations are essential as we can see that [13] wrote "incremental SVM may not be so easy to implement." The only work known to us that solves the primal problem is [14]. They propose a Newton method for both kernel and linear SVM. However, their method is not scalable because matrix inversions are needed.

# 3. INCREMENTAL AND DECREMENTAL LEARNING WITH WARM START

We consider a warm start setting by adjusting the optimal solution before data change as the initial solution of the new optimization problem. Because optimization algorithms are iterative procedures, we hope that a good initial solution can reduce the number of iterations to save the training time. Let

$$(y_i, \boldsymbol{x}_i), i = 1, \ldots, l$$

be the original training set, and $\boldsymbol{w}^*$ and $\boldsymbol{\alpha}^*$ be optimal solutions of the primal and the dual problems, respectively. For the primal problem with the variable $\boldsymbol{w}$, the number of variables is independent of data change because $\boldsymbol{w}$'s size is the same as the number of features. Therefore, $\boldsymbol{w}^*$ can be directly used such that

$$\bar{\boldsymbol{w}} \equiv \boldsymbol{w}^*$$

is the new initial solution. In contrast, if a dual problem is used, $\boldsymbol{\alpha}$'s size is changed after data addition or removal. For incremental learning, we assume

$$(y_i, \boldsymbol{x}_i), i = l+1, \ldots, l+k$$

[1] Besides (6), other optimization problems over $\boldsymbol{\alpha}$ may be considered; see, for example, [6].

are instances being added, and the following $\bar{\boldsymbol{\alpha}}$ is considered as the initial dual solution.

$$\bar{\boldsymbol{\alpha}} = [\alpha_1^*, \dots, \alpha_l^*, 0, \dots, 0]^T \in R^{(l+k)\times 1}.$$

For $\bar{\alpha}_{l+1}, \dots, \bar{\alpha}_{l+k}$, any value in $[0, U]$ can be chosen to ensure the feasibility. Because it is unclear which value is the best, we simply consider the zero value. For decremental learning, we assume instances

$$(y_i, \boldsymbol{x}_i), i = 1, \dots, k$$

are removed, and consider the following feasible $\bar{\boldsymbol{\alpha}}$ as the initial dual solution.

$$\bar{\boldsymbol{\alpha}} = [\alpha_{k+1}^*, \dots, \alpha_l^*]^T. \tag{9}$$

We hope that an initial solution closer to the optimum helps to reduce the running time. Subsequently, we check the initial objective values of solving the primal and the dual problems. A finding is that the primal initial point is closer to the optimal point if the change of data is not significant.

## 3.1 Initial Values for Incremental Learning

The optimization problem after considering new data is

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum\nolimits_{i=1}^{l+k}\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i). \tag{10}$$

The primal and dual initial values are, respectively,

$$\frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + C\sum\nolimits_{i=1}^{l}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) + C\sum\nolimits_{i=l+1}^{l+k}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) \tag{11}$$

and

$$\begin{aligned}&\sum_{i=1}^{l} h(\alpha_i^*, C) + \sum_{i=l+1}^{l+k} h(0, C) - \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}^{*T} & \mathbf{0}^T\end{bmatrix}\begin{bmatrix}\bar{Q} & \vdots \\ \cdots & \end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}^* \\ \mathbf{0}\end{bmatrix}\\ =&\frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + C\sum\nolimits_{i=1}^{l}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i),\end{aligned} \tag{12}$$

where (12) comes from (7), and the property of same primal and dual optimal values in training the original set. Then

$$\begin{aligned}\text{dual initial value} &\le \text{new optimal objective value}\\ &\le \text{primal initial value}\end{aligned}$$

because $\bar{\boldsymbol{w}}$ and $\bar{\boldsymbol{\alpha}}$ are feasible solutions of the new primal and dual problems, respectively.

We argue that the primal initial value is usually closer to the optimal objective value of the new problem than the dual. A scaled form of problem (1) is

$$\min_{\boldsymbol{w}} \quad \frac{1}{2Cl}\boldsymbol{w}^T\boldsymbol{w} + \frac{1}{l}\sum\nolimits_{i=1}^{l}\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i), \tag{13}$$

which minimizes the average loss with regularization. If the original and new data points follow a similar distribution, and the original optimal solution $\boldsymbol{w}^*$ describes the average loss well, then the optimal $\boldsymbol{w}$ of (10) should be similar to $\boldsymbol{w}^*$. Therefore, (11) may be a good approximation of the new optimal objective value.

For the dual initial objective value, from (10)-(12), we suspect that it may be far away from the new optimal value because of lacking the following term

$$C\sum\nolimits_{i=l+1}^{l+k}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i).$$

One may question the above conclusion on the superiority of the primal initial value by claiming that the regularization parameter $C$ must be adjusted for a larger training set. That is, to keep the same amount of total training losses, the following optimization problem can be considered.

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \Delta_I C\sum\nolimits_{i=1}^{l+k}\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i),$$

where $\Delta_I = l/(l+k)$. The corresponding dual problem is

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad &\sum_{i=1}^{l+k} h(\alpha_i, \Delta_I C) - \frac{1}{2}\sum_{i=1}^{l+k}\sum_{j=1}^{l+k}\alpha_i\alpha_j K(i,j) - \sum_{i=1}^{l+k}\frac{\alpha_i^2}{2}\frac{d}{\Delta_I}\\ \text{subject to} \quad &0 \le \alpha_i \le U, \forall i = 1, \dots, l+k.\end{aligned} \tag{14}$$

For LR and L1-loss SVM, the upper bound $U$ becomes

$$U = \Delta_I C. \tag{15}$$

We then consider the following initial solutions

$$\text{primal: } \bar{\boldsymbol{w}} \qquad \text{dual: } \Delta_I\bar{\boldsymbol{\alpha}}. \tag{16}$$

We must scale $\bar{\boldsymbol{\alpha}}$ because of the new upper bound in (15), but $\bar{\boldsymbol{w}}$ can still be used without modification. We have

$$\begin{aligned}&\text{new optimal objective value}\\ \le\ &\text{primal initial value}\\ =\ &\frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + \Delta_I C\sum\nolimits_{i=1}^{l}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) + \Delta_I C\sum\nolimits_{i=l+1}^{l+k}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i).\end{aligned} \tag{17}$$

Further,

$$\begin{aligned}&\text{dual initial value} = \sum\nolimits_{i=1}^{l+k} h(\Delta_I\bar{\alpha}_i, \Delta_I C)\\ &- \frac{1}{2}\Big(\sum\nolimits_{i=1}^{l+k}\sum\nolimits_{j=1}^{l+k}\Delta_I\bar{\alpha}_i\Delta_I\bar{\alpha}_j K(i,j) + \sum\nolimits_{i=1}^{l+k}(\Delta_I\bar{\alpha}_i)^2\frac{d}{\Delta_I}\Big)\\ =&\Delta_I C\sum\nolimits_{i=1}^{l}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) + \frac{l(l+2k)}{2(l+k)^2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}},\end{aligned} \tag{18}$$

where details of deriving (18) are in Appendix. Because

$$\frac{l(l+2k)}{2(l+k)^2} \le \frac{1}{2} \text{ and } \Delta_I C\sum_{i=l+1}^{l+k}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) \text{ is lacked,} \tag{19}$$

(18) tends to be too small. Thus, even if the parameter $C$ has been adjusted, the dual initial solution may still be farther away from the optimum than the primal. In Section 6, we will conduct experiments to confirm our findings.

## 3.2 Initial Values for Decremental Learning

The optimization problem after data removal is

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum\nolimits_{i=k+1}^{l}\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i). \tag{20}$$

The corresponding dual problem is

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad &\sum_{i=k+1}^{l} h(\alpha_i, C) - \frac{1}{2}\sum_{i=k+1}^{l}\sum_{j=k+1}^{l}\alpha_i\alpha_j K(i,j) - \sum_{i=k+1}^{l}\frac{\alpha_i^2}{2}d\\ \text{subject to} \quad &0 \le \alpha_i \le U, \forall i = k+1, \dots, l.\end{aligned} \tag{21}$$

Using the initial points defined in (9), the primal and dual initial objective values are, respectively,

$$\frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + C\sum\nolimits_{i=k+1}^{l}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) \tag{22}$$

and

$$\sum_{i=k+1}^{l} h(\alpha_i^*, C) - \frac{1}{2}\Big(\sum_{i=k+1}^{l}\sum_{j=k+1}^{l} \alpha_i^*\alpha_j^* K(i,j) + \sum_{i=k+1}^{l} (\alpha_i^*)^2 d\Big)$$
$$= \frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + C\sum\nolimits_{i=k+1}^{l} \xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i)$$
$$- \frac{1}{2}\sum\nolimits_{i=1}^{k}\sum\nolimits_{j=1}^{k} \alpha_i^*\alpha_j^* K(i,j). \quad (23)$$

Details are in Appendix. Similar to (10), we have

$$\text{dual initial value} \le \text{new optimal objective value} \le \text{primal initial value}$$

because $\bar{\boldsymbol{w}}$ and $\bar{\boldsymbol{\alpha}}$ are feasible solutions of the new primal and dual problems, respectively.

For decremental learning, we can also argue that the primal initial objective value is usually closer than the dual to the optimal objective value of the new problem. By (1), (20), and the same explanation for incremental learning, if not many data are removed, the optimal $\boldsymbol{w}$ should not change significantly. In contrast, because of the

$$-\frac{1}{2}\sum\nolimits_{i=1}^{k}\sum\nolimits_{j=1}^{k} \boldsymbol{\alpha}_i^*\boldsymbol{\alpha}_j^* K(i,j).$$

term in (23), the dual initial objective value should be less close to the new optimal value.

Like incremental learning, we discuss the situation of adjusting the parameter $C$ to maintain a similar total training loss. The following optimization problem is considered.

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \Delta_D C\sum\nolimits_{i=k+1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i),$$

where $\Delta_D = l/(l-k)$. The dual problem is the same as (21) except that $C$ becomes $\Delta_D C$, $d$ becomes $d/\Delta_D$, and $U = \Delta_D C$ for L1-loss SVM and LR. Like (16), the following initial solutions can be used.

$$\text{primal: } \bar{\boldsymbol{w}} \qquad \text{dual: } \Delta_D\bar{\boldsymbol{\alpha}}.$$

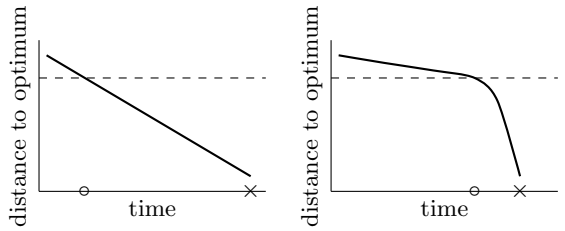
For initial objective values, we have

$$\text{new optimal objective value} \quad (24)$$
$$\le \text{primal initial value} = \frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + \Delta_D C\sum\nolimits_{i=k+1}^{l} \xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i)$$

and

$$\text{dual initial value} = \sum\nolimits_{i=k+1}^{l} h(\Delta_D\alpha_i^*, \Delta_D C) -$$
$$\frac{1}{2}\sum\nolimits_{i=k+1}^{l}\sum\nolimits_{j=k+1}^{l} \Delta_D^2\alpha_i^*\alpha_j^* K(i,j) - \sum\nolimits_{i=k+1}^{l} (\Delta_D\alpha_i^*)^2\frac{d}{2\Delta_D}$$
$$= \frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + \Delta_D C\sum_{i=k+1}^{l} \xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) - \frac{\Delta_D}{2}\sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_i^*\alpha_j^* K(i,j)$$
$$- \frac{\Delta_D^2 - \Delta_D}{2}\sum_{i=k+1}^{l}\sum_{j=k+1}^{l} \alpha_i^*\alpha_j^* K(i,j) + \frac{\Delta_D - 1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}}, \quad (25)$$

where the details of (25) are in supplementary materials because of space limitation. If not many data are removed, $\Delta_D \approx 1$ and the last two terms in (25) are close to zero. Then the difference between (24) and (25) is similar to that between (22) and (23) without changing $C$. Therefore, even if the regularization parameter has been adjusted, the dual initial solution tends to be farther away from the optimum than the primal.




(a) Linear convergence (b) Superlinear or quadratic convergence

**Figure 1: An illustration on how optimization methods may affect the effectiveness of a warm start setting. The $y$-axis is log-scaled. ○: initial solution ×: optimum. The dashed horizontal line indicates the initial distance to the optimum.**

# 4. OPTIMIZATION METHODS AND INCREMENTAL/DECREMENTAL LEARNING

The effectiveness of a warm start strategy may be strongly related to the optimization method. We use Figure 1 to illustrate this point. If an initial solution is close to the optimum, then a high-order optimization method may be advantageous because of the fast final convergence; see Figure 1(b). Therefore, higher-order methods such as quasi Newton or Newton may be preferred for incremental and decremental learning.

Interestingly, for linear classification, lower-order methods such as stochastic gradient (SG) or coordinate descent are more commonly used than high-order methods. The reason is that low-order methods can quickly return a useful model. On the contrary, a high-order method like Newton methods may take considerable time to finish the first few iterations for obtaining an approximate solution.

To investigate if high-order methods become useful for incremental and decremental learning, in the rest of this section, we briefly describe three methods that will be detailedly compared in Section 6.

- Newton method to solve the primal problem.
- Coordinate descent method to solve the primal problem.
- Coordinate descent method to solve the dual problem.

The Newton method uses second-order information, while the coordinate descent method considers only gradient (i.e., first order) information.

## 4.1 Solving Primal Problem by a Trust Region Newton Method

We consider the Newton method in LIBLINEAR to solve the primal problem (1). It is a trust region Newton (TRON) method developed in [15]. Because differentiability is required, this method is not applicable to L1-loss SVM.

At current iterate $\boldsymbol{w}$, TRON obtains an approximate Newton step $\boldsymbol{d}$ within the trust region by solving the following sub-problem.

$$\min_{\boldsymbol{d}} \quad q(\boldsymbol{d}) \quad \text{subject to} \quad \|\boldsymbol{d}\|_2 \le \Delta, \quad (26)$$

where $$q(\boldsymbol{d}) \equiv \nabla f(\boldsymbol{w})^T\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T\nabla^2 f(\boldsymbol{w})\boldsymbol{d}$$

is an approximation of $f(\boldsymbol{w}+\boldsymbol{d}) - f(\boldsymbol{w})$ and $\Delta$ is the size of the trust region. Afterward by checking the ratio between

real function value reduction $f(\boldsymbol{w}+\boldsymbol{d})-f(\boldsymbol{w})$ and the estimated reduction $q(\boldsymbol{d})$, TRON decides if $\boldsymbol{w}$ should be updated and then adjusts the current trust region $\Delta$. The sub-problem (26) is solved by the conjugate gradient method, so *matrix inversion is not needed.* For LR, whose objective function is twice differentiable, [15] shows that TRON has quadratic convergence.

## 4.2 Solving Primal Problem by a Coordinate Descent Method

We consider the coordinate descent method in [5] to solve the primal problem (1). This method updates one component $w_i$ of $\boldsymbol{w}$ at a time.

$$w_i \leftarrow w_i + \arg\min_d f(\boldsymbol{w}+d\boldsymbol{e}_i), \tag{27}$$

where $\boldsymbol{e}_i \equiv [0,\dots,0,1,0,\dots,0]^T$ is zero except the $i$th element. The work [5] applies a Newton method to approximately solve the one-variable sub-problem in (27), which does not have a closed-form solution. Their approach, applicable to L2-loss SVM and LR, requires only $\nabla_i f(w)$ and $\nabla^2_{ii} f(w)$ at each step. Therefore, this method is a low-order one in compared with TRON that needs the whole Hessian matrix. The linear convergence is established in [5]. We refer to this method as PCD.

## 4.3 Solving Dual Problem by a Coordinate Descent Method

For solving dual problem (6), we consider the coordinate descent methods in [10, 18] that update $\alpha_i$ at a time by

$$\alpha_i \leftarrow \min\left(\max\left(\alpha_i + \arg\min_d f^D(\boldsymbol{\alpha}+d\boldsymbol{e}_i), 0\right), U\right), \tag{28}$$

where $f^D(\cdot)$ is the dual objective function. A difference between (28) and (27) is that in (28) we must ensure the new $\alpha_i$ is in $[0, U]$. The one-variable problem in (28) has a closed-form solution for L1-loss and L2-loss SVM, but for LR, we need an optimization procedure to obtain an approximate solution. It is shown in [10, 17] that this dual coordinate descent method linearly converges.

# 5. IMPLEMENTATION ISSUES

Although warm start is a simple strategy, its implementation may be complicated. Recall in Section 1 we mentioned that the need to maintain the kernel cache is the main obstacle for us to support incremental and decremental learning in the kernel SVM software LIBSVM. Now for linear classification, although the implementation is more straightforward, many issues still need to be addressed in this section.

We begin with checking if additional information must be maintained in the model after training. If the primal problem is considered, then $\bar{\boldsymbol{w}}$ is readily available because the previously obtained $\boldsymbol{w}$ must be stored for prediction. In contrast, the dual solution $\boldsymbol{\alpha}$ is not maintained in a linear classifier even if a dual-based solver is used. The reason is because we can generate and store $\boldsymbol{w}$ by (5) for prediction. Therefore, to employ a dual solver for incremental and decremental learning, $\boldsymbol{\alpha}$ must be additionally stored. Unfortunately, maintaining $\boldsymbol{\alpha}$ is a complicated task because of the following concerns on the correspondence between $\boldsymbol{\alpha}$ and data instances.

- If new instances are randomly inserted into the existing set, it is difficult to maintain the mapping between $\boldsymbol{\alpha}$ and instances. Similarly, for decremental learning, the indices of removed instances must be known, though in practice this information may not be available.
- The task of multi-class classification is often decomposed to several binary classification problems, so a set of $\boldsymbol{\alpha}$ vectors must be maintained. The storage cost can be high if both numbers of instances and classes are large. Further, if each binary problem involves a subset of data (e.g., the one-against-one approach for multi-class classification), the above-mentioned problem of mapping $\boldsymbol{\alpha}$ and data occurs.

Therefore, the implementation of dual solvers with warm start is more complicated than that of primal solvers.

Stopping conditions of optimization algorithms are another implementation issues because some relative conditions depend on the initial point. For example, the TRON implementation in LIBLINEAR employs the following condition.

$$\frac{\|\nabla f(\boldsymbol{w}^k)\|_2}{\|\nabla f(\boldsymbol{w}^0)\|_2} \le \epsilon \cdot \frac{\min(l^+, l^-)}{l}, \tag{29}$$

where $\boldsymbol{w}^0$ is the initial point, $\boldsymbol{w}^k$ is the current iterate, $\epsilon$ is the user-specified stopping tolerance, and $l^+$ and $l^-$ are numbers of positive and negative data, respectively. For standard linear classification, in general $\boldsymbol{w}^0 = \mathbf{0}$ is used. However, with a warm start setting, the initial $\boldsymbol{w}^0$ is better than $\mathbf{0}$, so a smaller $\|\nabla f(\boldsymbol{w}^0)\|_2$ appears in the denominator in (29). Then the stopping condition becomes too strict under the same $\epsilon$. To address this problem, in our implementation, $\|\nabla f(\boldsymbol{w}^0)\|_2$ in (29) is fixed to be $\|\nabla f(\mathbf{0})\|_2$.

**Table 1: Data statistics: Density is the average ratio of non-zero features per instance.**

| Data set | $l$: #instances | $n$: #features | density |
|---|---|---|---|
| ijcnn | 49,990 | 22 | 41.37% |
| webspam | 350,000 | 254 | 33.51% |
| news20 | 19,996 | 1,355,191 | 0.03% |
| rcv1 | 20,242 | 47,236 | 0.16% |
| real-sim | 72,309 | 20,958 | 0.24% |
| yahoo-japan | 176,203 | 832,026 | 0.02% |

# 6. EXPERIMENTS

In this section, we experimentally compare primal and dual solvers with the warm start strategy. We consider data sets ijcnn, webspam, news20, rcv1, real-sim and yahoo-japan, where details are shown in Table 1. All sets except yahoo-japan are available at LIBSVM data set.[2] Note that we choose the unigram version of webspam. All the experiments are conducted on a 2.50 GHz computer with 16 GB of RAM.

To evaluate methods for incremental learning, we randomly divide each data set to $r$ parts so that

$r-1$ parts as original data, and
1 part as new data.

Therefore, the solution of training the $r-1$ parts is used to construct the initial point for training the whole set of $r$ parts. We consider $r = 5, 50$ and 500 to investigate the effectiveness of the warm start method under different levels of data changes. Note that a larger $r$ means a smaller increase of the data. Our setting ensures that the enlarged set remains the same regardless of different $r$ values. We need this property so in figures for comparison only one curve is

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

drawn for the approach without a warm start setting. For decremental learning, we use the following setting to ensure that the reduced set is the same regardless of $r$. We split 1/5 data to $r$ parts and have

4/5 data + 1 part of 1/5 data as the original data and
4/5 data as the reduced set.

Our evaluation is by the relative difference to the optimal function value:

$$|\frac{f(\boldsymbol{w}) - f(\boldsymbol{w}^*)}{f(\boldsymbol{w}^*)}| \text{ and } |\frac{f^D(\boldsymbol{\alpha}) - f^D(\boldsymbol{\alpha}^*)}{f^D(\boldsymbol{\alpha}^*)}| \quad (30)$$

for primal- and dual-based algorithms, respectively. Note that optimal $\boldsymbol{w}^*$ and $\boldsymbol{\alpha}^*$ are not available, so we obtain their approximations by running a huge number of iterations of optimization methods.

## 6.1 Analysis on Initial Values

In Sections 3.1 and 3.2, we discuss properties of primal and dual initial solutions. To confirm our analysis, Table 2 shows the relative difference between initial and optimal function values for incremental and decremental learning. We check two values of the regularization parameter: $C = 1$ and $C = 128$. Because of space limitation, we consider only logistic regression, although results for l2-loss SVM (in Tables **??** and **??** of supplementary materials) are similar.[3] As a comparison, we include results without the warm start setting by using $\boldsymbol{w} = \mathbf{0}$ and $\boldsymbol{\alpha} = \mathbf{0}$ as initial points for primal- and dual-based methods, respectively. Note that $f_D(\mathbf{0}) = 0$, so

$$|\frac{f^D(\mathbf{0}) - f^D(\boldsymbol{\alpha}^*)}{f^D(\boldsymbol{\alpha}^*)}| = 1.$$

In contrast, primal $f(\mathbf{0})$ is larger. For example, for logistic loss, $f(\mathbf{0}) = Cl\log(2)$. Therefore, without a warm start setting, the primal initial point is far away from the optimal solution. This situation can be clearly seen in Table 2.

We further make the following observations. First, from results of each row of Table 2, the warm start setting significantly reduces the distance from the primal initial solution to the optimum. In contrast, the improvement on the dual initial solution is only modest.

Second, with the warm start setting, if $C = 1$, the distance of the primal initial solution to the optimum is smaller than that of the dual. In particular, for data with a small number of features (e.g., ijcnn), the primal initial solution is very close to the optimum. This result confirms the calculation in Sections 3.1 and 3.2. However, the difference becomes smaller for some sparse data (e.g., news20). An explanation is that because of more features, more instances are needed to have a stable optimal $\boldsymbol{w}$.

Third, if a larger $C = 128$ is used, the superiority of the primal initial solution becomes less clear. For sparse data, it is sometimes worse than the dual initial solution under the incremental learning scenario. This result is reasonable because a large $C$ causes a better fitting of the original data. Then the solution $\boldsymbol{w}$ is sensitive to the increase of data. However, we notice that for these data sets, a large $C$ is often not needed. Interestingly, for decremental learning with a large $C$, the primal initial point is competitive for sparse data. The reason might be that the remained data have been used in training the original set. In contrast, for incremental learning, some unseen instances are added to form the new training set.

Finally, we observe that under a larger $r$ both primal and dual initial solutions are closer to the optimum. This result follows from our setting that a larger $r$ implies a smaller change of the data. If the change of data is minor, then the initial solution $\bar{\boldsymbol{w}}$ or $\bar{\boldsymbol{\alpha}}$ is almost good enough to be a model for the modified problem. For example, even under $r = 5$, if $C = 1$ and the primal-based algorithm TRON is used for incremental learning, the default stopping condition of LIBLINEAR is reached within two Newton iterations for all data sets except webspam.

[3]We did not consider L1-loss SVM here because our primal-based optimization methods require the differentiability of the objective function.

## 6.2 Comparison of Optimization Methods for Incremental and Decremental Learning

In Figures 2 and 3, we compare the three optimization methods discussed in Section 4. Each sub-figure shows the relationship between the training time and the relative difference to the optimal function value; see (30). Because of space limitation, we present results of only logistic regression and leave the results of L2-loss SVM in supplementary materials. We can make the following observations.

First, the warm start strategy is useful regardless of optimization methods. If data are not significantly changed (i.e., larger $r$), the improvement is dramatic.

If without applying warm start, DCD is the fastest among the three optimization methods. This situation has been know from earlier works such as [10]. However, with warm start settings TRON becomes competitive. Generally TRON benefits more from warm start than DCD. For example, in Figure 2(b), if $r = 500$, to reach $10^{-8}$ relative difference, the warm start strategy reduces the DCD's training time from 4.5 to 3 seconds, while TRON's training time from 40 to around 10 seconds. This result confirms our conjecture in Figure 1 that warm start strategies are more helpful for high-order optimization methods.

Although the warm start setting significantly improves the training speed of TRON for solving the primal problem, from Figures 2 and 3, DCD is still faster in general. Past works (e.g., [10]) have shown that DCD may become inferior to TRON if $C$ is larger or feature values are in a large numerical range. In Figures **??** and **??** of supplementary materials, we present results of using $C = 128$. For data such as ijcnn, webspam, rcv1 and real-sim, we can clearly see that TRON is generally faster than DCD.

The two primal-based methods (PCD and TRON) share the same initial point. PCD quickly decreases the function value, but becomes slow in the end. In contrast, TRON is overall superior because of fast final convergence.

# 7. CONCLUSIONS

This research has lead to an extension of LIBLINEAR for incremental and decremental learning. Currently we choose TRON as the underlying solver because of the following results obtained in this research work.

- The warm start setting generally gives a better primal initial solution than the dual (Section 3).
- The warm start setting more effectively speeds up a high-order optimization method such as TRON (Section 4).
- For implementation, a primal-based method is more straightforward than a dual-based method (Section 5).

**Table 2: Relative objective value difference between the initial point and the optimal solution. Logistic regression is considered. wo-ws: without warm start. The better value between primal and dual is boldfaced.**

| Data set | Formulation | $C = 1$ | | | | $C = 128$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | wo-ws | $r = 5$ | $r = 50$ | $r = 500$ | wo-ws | $r = 5$ | $r = 50$ | $r = 500$ |
| ijcnn | Primal | 2.4e+00 | **3.4e−04** | **2.2e−05** | **1.3e−06** | 2.5e+00 | **4.0e−04** | **2.5e−05** | **1.4e−06** |
| | Dual | **1.0e+00** | 1.9e−01 | 1.8e−02 | 1.2e−03 | **1.0e+00** | 1.9e−01 | 1.9e−02 | 1.2e−03 |
| webspam | Primal | 2.1e+00 | **1.2e−04** | **8.5e−06** | **1.4e−06** | 2.4e+00 | **2.7e−04** | **4.5e−05** | **2.9e−06** |
| | Dual | **1.0e+00** | 1.9e−01 | 1.9e−02 | 2.1e−03 | **1.0e+00** | 2.0e−01 | 2.0e−02 | 2.3e−03 |
| news20 | Primal | 1.3e+00 | **1.7e−02** | **1.3e−03** | **1.3e−04** | 2.5e+01 | 5.6e−01 | 4.6e−02 | 2.2e−03 |
| | Dual | **1.0e+00** | 1.6e−01 | 1.4e−02 | 1.4e−03 | **1.0e+00** | **1.6e−01** | **1.3e−02** | **9.5e−04** |
| rcv1 | Primal | 2.4e+00 | **1.4e−02** | **9.9e−04** | **1.5e−04** | 3.4e+01 | 8.2e−01 | 5.8e−02 | 1.7e−02 |
| | Dual | **1.0e+00** | 1.5e−01 | 1.3e−02 | 1.7e−03 | **1.0e+00** | **2.0e−01** | **1.5e−02** | **2.1e−03** |
| real-sim | Primal | 3.8e+00 | **1.2e−02** | **1.1e−03** | **1.4e−04** | 3.4e+01 | 5.5e−01 | 5.9e−02 | 5.2e−03 |
| | Dual | **1.0e+00** | 1.5e−01 | 1.4e−02 | 1.7e−03 | **1.0e+00** | **2.2e−01** | **2.2e−02** | **1.8e−03** |
| yahoo-japan | Primal | 2.4e+00 | **1.6e−02** | **1.3e−03** | **1.2e−04** | 1.1e+01 | 6.7e−01 | 6.2e−02 | 6.6e−03 |
| | Dual | **1.0e+00** | 1.9e−01 | 1.9e−02 | 2.0e−03 | **1.0e+00** | **2.5e−01** | **2.5e−02** | **2.7e−03** |

(a) Incremental learning

| Data set | Formulation | $C = 1$ | | | | $C = 128$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | wo-ws | $r = 5$ | $r = 50$ | $r = 500$ | wo-ws | $r = 5$ | $r = 50$ | $r = 500$ |
| ijcnn | Primal | 2.3e+00 | **3.3e−04** | **3.2e−05** | **1.9e−06** | 2.5e+00 | **4.1e−04** | **3.8e−05** | **2.1e−06** |
| | Dual | **1.0e+00** | 4.6e−02 | 5.0e−03 | 2.2e−04 | **1.0e+00** | 4.2e+00 | 6.5e−01 | 2.5e−02 |
| webspam | Primal | 2.0e+00 | **1.1e−04** | **1.1e−05** | **1.9e−06** | 2.4e+00 | **2.6e−04** | **5.7e−05** | **4.5e−06** |
| | Dual | **1.0e+00** | 1.0e−02 | 9.0e−04 | 9.1e−05 | **1.0e+00** | 3.5e−01 | 9.6e−02 | 9.3e−03 |
| news20 | Primal | 1.2e+00 | 1.3e−02 | 1.5e−03 | 1.4e−04 | 2.3e+01 | **7.2e−02** | **8.2e−03** | 5.6e−04 |
| | Dual | **1.0e+00** | **8.5e−03** | **8.2e−04** | **4.0e−05** | **1.0e+00** | 2.8e−01 | 3.9e−02 | **2.2e−04** |
| rcv1 | Primal | 2.2e+00 | 1.1e−02 | 1.1e−03 | 1.7e−04 | 3.3e+01 | **1.1e−01** | **8.8e−03** | 1.7e−03 |
| | Dual | **1.0e+00** | **3.6e−03** | **3.4e−04** | **3.6e−05** | **1.0e+00** | 4.6e−01 | 4.5e−02 | **7.8e−04** |
| real-sim | Primal | 3.5e+00 | 8.9e−03 | 1.1e−03 | 1.6e−04 | 3.4e+01 | **1.1e−01** | **1.2e−02** | **1.7e−03** |
| | Dual | **1.0e+00** | **5.2e−03** | **6.3e−04** | **9.4e−05** | **1.0e+00** | 9.9e−01 | 8.5e−02 | 5.8e−03 |
| yahoo-japan | Primal | 2.4e+00 | 1.2e−02 | 1.5e−03 | **1.4e−04** | 1.2e+01 | **1.4e−01** | **1.5e−02** | **1.9e−03** |
| | Dual | **1.0e+00** | **9.5e−03** | **1.5e−03** | 1.5e−04 | **1.0e+00** | 1.2e+00 | 1.6e−01 | 1.5e−02 |

(b) Decremental learning

With the release of the software, we hope feedbacks from real applications can lead us to refine the methods for incremental and decremental learning.

## 8. ACKNOWLEDGMENTS

This work was supported in part by the National Science Council of Taiwan via the grant 101-2221-E-002-199-MY3. The authors thank Chia-Hua Ho for fruitful discussion.

## 9. REFERENCES

[1] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, 1992.

[2] L. Bottou and C.-J. Lin. Support vector machine solvers. In *Large Scale Kernel Machines*. 2007.

[3] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *NIPS*. 2001.

[4] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2(3):27:1–27:27, 2011.

[5] K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. Coordinate descent method for large-scale L2-loss linear SVM. *JMLR*, 9:1369–1398, 2008.

[6] O. Chapelle. Training a support vector machine in the primal. *Neural Comput.*, 19:1155–1178, 2007.

[7] C. Cortes and V. Vapnik. Support-vector network. *MLJ*, 20:273–297, 1995.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: a library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[9] S. Fine and K. Scheinberg. Incremental learning and selective sampling via parametric optimization framework for SVM. In *NIPS*. 2001.

[10] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *ICML*, 2008.

[11] M. Karasuyama and I. Takeuchi. Multiple incremental decremental learning of support vector machines. *IEEE TNN*, 21:1048–1059, 2010.

[12] G. S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Ann. Math. Stat.*, 41:495–502, 1970.

[13] P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller. Incremental support vector learning: Analysis, implementation and applications. *JMLR*, 7:1909–1936, 2006.

[14] Z. Liang and Y. Li. Incremental support vector machine learning in the primal and applications. *Neurocomputing*, 72(10):2249–2258, 2009.

[15] C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. *JMLR*, 9:627–650, 2008.

[16] A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi. Incremental training of support vector machines. *IEEE TNN*, 16:114–131, 2005.

[17] P.-W. Wang and C.-J. Lin. Iteration complexity of feasible descent methods for convex optimization. *JMLR*, 2014.

[18] H.-F. Yu, F.-L. Huang, and C.-J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *MLJ*, 85:41–75, 2011.

[19] G.-X. Yuan, C.-H. Ho, and C.-J. Lin. Recent advances of large-scale linear classification. *PIEEE*, 100:2584–2603, 2012.

# APPENDIX

## Details of Deriving (18)

To begin, we show that for any constant $\Delta > 0$,

$$h(\Delta\alpha, \Delta C) = \Delta h(\alpha, C). \tag{31}$$

For L1 and L2 losses, $h(\alpha, C) = \alpha$, so (31) obviously holds. For LR loss,

$$\begin{aligned}
&h(\Delta\alpha, \Delta C)\\
=&\Delta C \log \Delta C - \Delta\alpha \log \Delta\alpha - (\Delta C - \Delta\alpha)\log(\Delta C - \Delta\alpha)\\
=&\Delta C(\log\Delta + \log C) - \Delta\alpha(\log\Delta + \log\alpha)\\
&- \Delta(C-\alpha)(\log\Delta + \log(C-\alpha))\\
=&\Delta(C\log C - \alpha\log\alpha - (C-\alpha)\log(C-\alpha)) = \Delta h(\alpha, C).
\end{aligned}$$

With $\Delta_I = l/(l+k)$ defined in Section 3.1, we have

$$\begin{aligned}
&\frac{1}{2}\boldsymbol{\alpha}^{*T}\bar{Q}\boldsymbol{\alpha}^* - \frac{\Delta_I}{2}\boldsymbol{\alpha}^{*T}(Q + \frac{D}{\Delta_I})\boldsymbol{\alpha}^* \qquad (32)\\
=&\frac{1}{2}\boldsymbol{\alpha}^{*T}(Q+D)\boldsymbol{\alpha}^* - \frac{\Delta_I}{2}\boldsymbol{\alpha}^{*T}(Q + \frac{D}{\Delta_I})\boldsymbol{\alpha}^* = \frac{k}{2(l+k)}\boldsymbol{\alpha}^{*T}Q\boldsymbol{\alpha}^*.
\end{aligned}$$

Finally, we derive (18) in detail.

$$\begin{aligned}
&\text{dual initial value} = \textstyle\sum_{i=1}^{l+k} h(\Delta_I\bar{\alpha}_i, \Delta_I C)\\
&- \frac{1}{2}\left(\textstyle\sum_{i=1}^{l+k}\sum_{j=1}^{l+k}\Delta_I\bar{\alpha}_i\Delta_I\bar{\alpha}_j K(i,j) + \sum_{i=1}^{l+k}(\Delta_I\bar{\alpha}_i)^2\frac{d}{\Delta_I}\right)\\
=&\Delta_I\left(\textstyle\sum_{i=1}^{l+k} h(\bar{\alpha}_i, C) - \frac{\Delta_I}{2}\boldsymbol{\alpha}^{*T}(Q + \frac{D}{\Delta_I})\boldsymbol{\alpha}^*\right)\\
=&\Delta_I\left(\frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + C\textstyle\sum_{i=1}^{l}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i)\right.\\
&\left.+\frac{1}{2}\boldsymbol{\alpha}^{*T}\bar{Q}\boldsymbol{\alpha}^* - \frac{\Delta_I}{2}\boldsymbol{\alpha}^{*T}(Q + \frac{D}{\Delta_I})\boldsymbol{\alpha}^*\right)\\
=&\Delta_I C\textstyle\sum_{i=1}^{l}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) + \frac{\Delta_I}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + \frac{\Delta_I k}{2(l+k)}\boldsymbol{\alpha}^{*T}Q\boldsymbol{\alpha}^*\\
=&\Delta_I C\textstyle\sum_{i=1}^{l}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) + \frac{l(l+2k)}{2(l+k)^2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}}.
\end{aligned}$$

The second equality is from (31) and $\bar{\alpha}_{l+1} = \cdots = \bar{\alpha}_{l+k} = 0$. The third equality is from (7) and (12). The fourth equality is from (32). The last equality uses the property $\boldsymbol{w}^{*T}\boldsymbol{w}^* = \boldsymbol{\alpha}^{*T}Q\boldsymbol{\alpha}^*$ of optimal solutions.

## Details of Deriving (23)

To begin, we show that

$$h(\alpha_i^*, C) - \frac{1}{2}d\alpha_i^{*2} - y_i\alpha_i^*\boldsymbol{w}^{*T}\boldsymbol{x}_i = C\xi(\boldsymbol{w}^*; \boldsymbol{x}_i, y_i) \tag{33}$$

holds for L1, L2 and LR losses. For L1-loss SVM,

$$\alpha_i^* - y_i\alpha_i^*\boldsymbol{w}^{*T}\boldsymbol{x}_i = \alpha_i^*(1 - y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i) = C\max(1 - y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i, 0)$$

by the following optimality condition (e.g., Eq. (17) in [2]).

$$\alpha_i^* = \begin{cases} C & \text{if } 1 - y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i > 0,\\ 0 & \text{if } 1 - y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i < 0.\end{cases}$$

For L2-loss SVM,

$$\begin{aligned}
\alpha_i^* - \frac{1}{4C}\alpha_i^{*2} - y_i\alpha_i^*\boldsymbol{w}^{*T}\boldsymbol{x}_i &= \alpha_i^*(1 - y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i) - \frac{1}{4C}\alpha_i^{*2}\\
&= C\max(1 - y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i, 0)^2.
\end{aligned}$$

The last equality is from the optimality condition

$$\alpha_i^* = 2C\max(1 - y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i, 0).$$

For LR, consider the following optimality condition

$$\alpha_i^* = C\frac{e^{-y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i}}{1 + e^{-y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i}}$$

in Section 3.4 of [18]. Then we have (33) by

$$\begin{aligned}
&C\log C - \alpha_i^*\log\alpha_i^* - (C-\alpha_i^*)\log(C-\alpha_i^*) - y_i\alpha_i^*\boldsymbol{w}^{*T}\boldsymbol{x}_i\\
=&C\log\frac{C}{C-\alpha_i^*} + \alpha_i^*\log\frac{C-\alpha_i^*}{\alpha_i^*} - y_i\alpha_i^*\boldsymbol{w}^{*T}\boldsymbol{x}_i\\
=&C\log(1 + e^{-y_i\boldsymbol{w}^{*T}\boldsymbol{x}_i}).
\end{aligned}$$

Denote the optimal value of the original problem as $V^*$.

$$V^* = \sum_{i=1}^{l} h(\alpha_i^*, C) - \frac{1}{2}\left(\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i^*\alpha_j^*K(i,j) + \sum_{i=1}^{l}d\alpha_i^{*2}\right).$$

Next, we denote the initial value of the decremented problem as $V_{\text{init}}$ and extract the term $V_{\text{init}}$ from $V^*$ to have

$$\begin{aligned}
V^* &= V_{\text{init}} - \sum_{i=1}^{k}\sum_{j=1}^{l}\alpha_i^*\alpha_j^*K(i,j) + \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i^*\alpha_j^*K(i,j)\\
&- \frac{1}{2}\textstyle\sum_{i=1}^{k}d\alpha_i^{*2} + \sum_{i=1}^{k}h(\alpha_i^*, C)\\
&= V_{\text{init}} - \textstyle\sum_{i=1}^{k}y_i\alpha_i^*\boldsymbol{w}^{*T}\boldsymbol{x}_i + \sum_{i=1}^{k}h(\alpha_i^*, C) - \frac{1}{2}\sum_{i=1}^{k}d\alpha_i^{*2}\\
&+ \frac{1}{2}\textstyle\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i^*\alpha_j^*K(i,j)\\
&= V_{\text{init}} + C\textstyle\sum_{i=1}^{k}\xi(\boldsymbol{w}^*; \boldsymbol{x}_i, y_i) + \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i^*\alpha_j^*K(i,j).
\end{aligned}$$

The second equality uses (5) and the last equality is from (33). By the definitions of $\bar{\boldsymbol{w}}$ and $V^*$, and (12),

$$\begin{aligned}
V_{\text{init}} &= V^* - C\sum_{i=1}^{k}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) - \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i^*\alpha_j^*K(i,j)\\
&= \frac{1}{2}\bar{\boldsymbol{w}}^T\bar{\boldsymbol{w}} + C\sum_{i=k+1}^{l}\xi(\bar{\boldsymbol{w}}; \boldsymbol{x}_i, y_i) - \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i^*\alpha_j^*K(i,j).
\end{aligned}$$

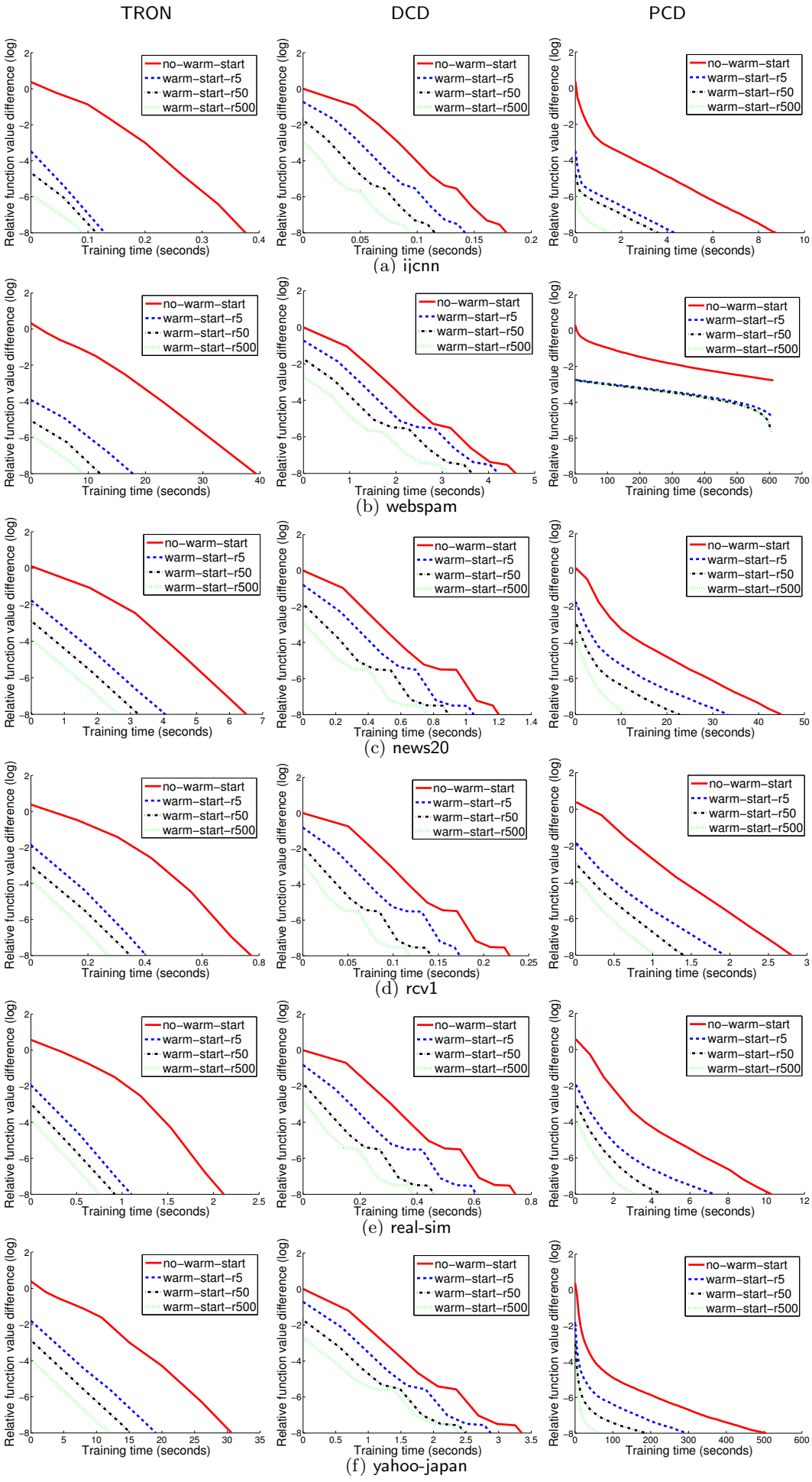



**Figure 2: Incremental learning: running time (in seconds) versus the relative objective value difference. Logistic regression with $C = 1$ is used.**

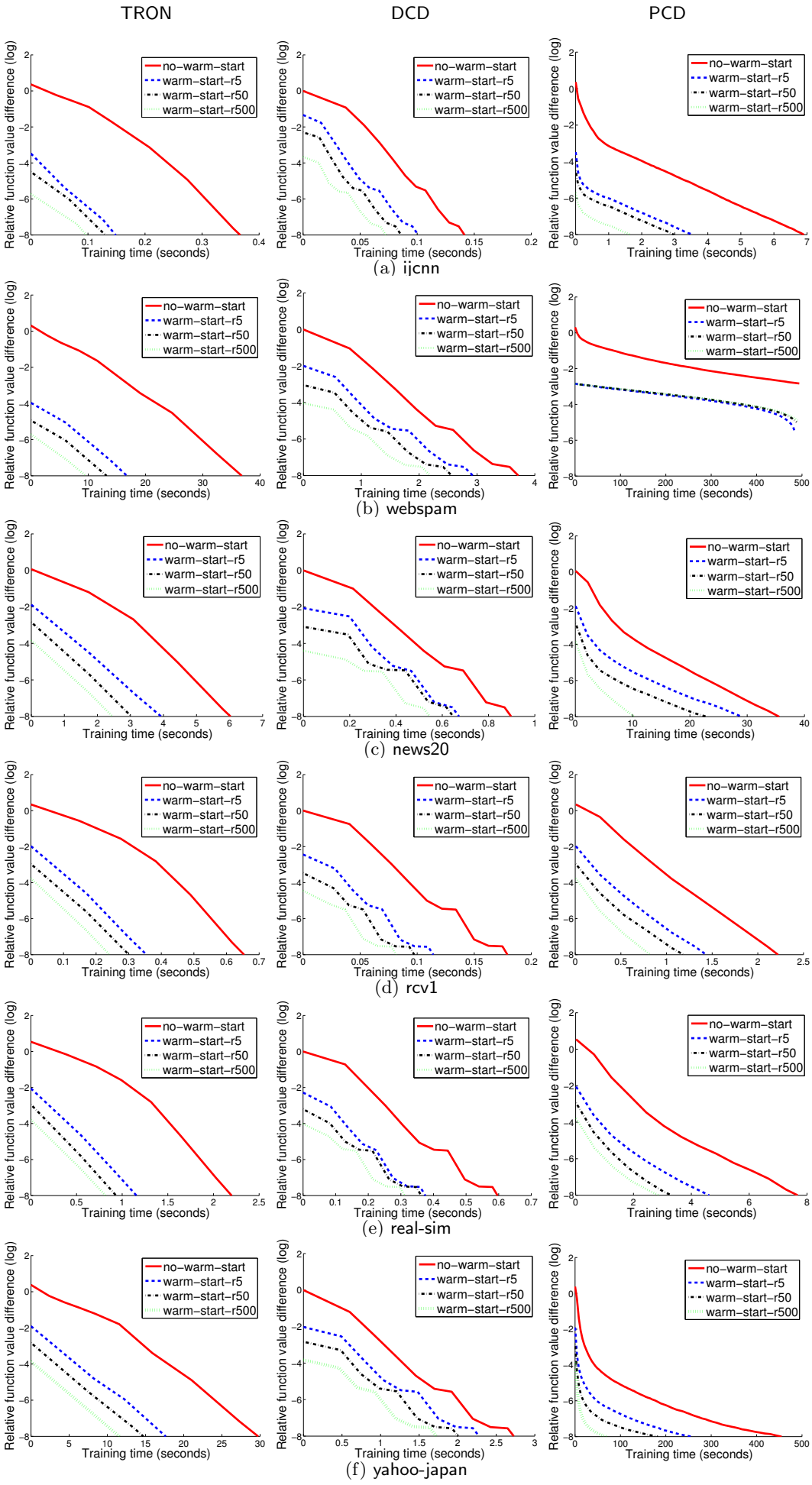



**Figure 3: Decremental learning: running time (in seconds) versus the relative objective value difference. Logistic regression with $C = 1$ is used.**