Rank-r Updating Techniques for Fast Exact Cross-Validation

Yi-Cheng Tseng[⋆] and Yuh-Jye Lee

Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, 106 Taiwan
{D9915006,yuh-jye}@mail.ntust.edu.tw

Abstract. In recent years, Support Vector Machines (SVMs) have been successfully developed and have become powerful tools for pattern recognition and machine learning. Although SVMs have shown excellent classification and prediction performance in many real applications, the parameters setting is very crucial to the SVMs' performance. The k-fold cross-validation (k-fold CV) and the leave-one-out cross-validation (LOOCV) are two popular methods to obtain the best parameters setting. However, the computational costs of them are prohibitively expensive, especially for large-scale problems. Based on the observation of Smooth Support Vector Machine (SSVM) updating from the computation point of view, we proposed two efficient updating strategies by the Sherman-Morrison-Woodbury formula to reduce the cost of finding the Hessian inverse in this work. We introduced our two updating strategies in the k-fold CV and the LOOCV. It will dramatically reduce the computational cost and still can have the exact answers. In the experiments, we demonstrated the effectiveness of SSVM with we proposed strategies on several datasets. Two different types of datasets are chosen to demonstrate the advantage of two different strategies. Our updating strategies can be applied to any learning algorithm which it solved iteratively and involved the Hessian inverse in each iteration, such as Smooth Support Vector Machine for ϵ-Insensitive Regression (ϵ-SSVR), Least-Square Support Vector Machine (LSSVM), Training SVM in the Primal, Second-Order Online Perceptron Algorithm and so forth.

Keywords: Support Vector Machine, Sherman-Morrison-Woodbury Formula, Cross-Validation, Hessian Inverse, Newton's Method.

In recent years, Support Vector Machines (SVMs) [4] have been successfully developed and have become powerful tools for pattern recognition and machine learning such as classification problems and so forth. In the classification problems, SVMs determine an optimal separating hyperplane that classify the data instances into the different classes. The "optimal" means that the separating hyperplane has the best generalization ability for the unseen data instances based on the statistical learning theory [24]. Traditionally, most of the optimization

[⋆] #43, Sec.4, Keelung Rd., Taipei, 106, Taiwan, R.O.C problems of SVMs are solved in the dual space. At the same time, there are some variant SVMs in the primal space have been developed, such as Smooth Support Vector Machine (SSVM) [17], Training SVM in the Primal [8] and so forth. In SSVM, smoothing methods are applied to generate and solve an unconstrained smooth reformulation of the SVM in the primal space using a completely arbitrary kernel. A fast Newton-Armijo algorithm for solving the SSVM converges globally and quadratically. However, finding the Hessian inverse in each SSVM iterations is very costly and becomes the bottleneck of SSVM.

Although SVMs have shown excellent classification and prediction performance in many real world applications, the parameters setting is very crucial to the SVMs' performance [2, 15]. The most common method to obtain the best parameters is the k-fold cross-validation (k-fold CV). An extreme form of the k-fold CV is the leave-one-out cross-validation (LOOCV), where only one instance is treated as test data and the rest part play the role as training data at each time. The LOOCV also is a widely used measuring the generalization ability of the model, because it has been shown to give an almost unbiased estimator of the generalization ability of the model [2, 13]. However, the computational costs of them are prohibitively expensive, especially for large-scale problems.

In this work, we proposed two efficient updating strategies by the observation of SSVM updating from the computation point of view, called "*Strategy I*" and "*Strategy II*", to reduce the cost of finding the Hessian inverse. Both our strategies belong to the incremental/decremental updating schemes. We introduced our two updating strategies in the k-fold CV and the LOOCV. In the experiments, we demonstrated the effectiveness of SSVM with we proposed strategies on several datasets such as Face [22], Ionosphere [3], Pima [3], Colon [1] and Leukemia [12]. Our updating strategies can be applied to any learning algorithm which it solved iteratively and involved the Hessian inverse in each iteration, such as Smooth Support Vector Machine for ϵ-Insensitive Regression (ϵ-SSVR) [18], Least-Square Support Vector Machine (LSSVM) [23], Training SVM in the Primal [8], Second-Order Online Perceptron Algorithm [6] and so forth.

This paper is organized as follows. We briefly introduce the SSVM in Section 1. In Section 2, we describe the observation of SSVM updating from computation point of view. Section 3 shows how to apply our Hessian inverse updating strategies to obtain the exact results of the k-fold CV and the LOOCV efficiently. The numerical experiments are shown in Section 4. In Section 5, we conclude this work and future research direction.

1 Smooth Support Vector Machine (SSVM)

In SSVM, the binary problem of classification on ℓ instances in the n-dimensional real space R^n. The standard linear SSVM is given by the following:

$$\min_{b,w} f_\beta(b,w) = \tfrac{1}{2}(\|w\|_2^2 + b^2) + \tfrac{c}{2}\|p(\mathbf{1} - Y(Xw + \mathbf{1}b), \beta)\|_2^2 \ , \qquad (1)$$

where $p(\cdot)$ is the p-function [17], $X \in R^{\ell \times n}$ is the data matrix, $\mathbf{1}$ is a column vector of ones of ℓ dimension, w is the normal vector, b determines their location relative to the origin, Y is the diagonal matrix with ones or minus ones along its diagonal to specify the membership of each instance, in other words, $Y_{jj} = \pm 1$ depending on whether the label of j^{th} data instance is $+1$ or -1, and c is a positive value for balancing the training error and the regularization term in the objective function. Too large c might cause the overfitting problem.

The objective function in Problem(1) is twice differentiable, then Newton's method which is quadratically convergent algorithm can be applied. However, Newton's method might lead to the oscillation phenomenon. In order to avoid this phenomenon, the Armijo step-size rule is employed to make the solution convergent globally. In our implementation, we use following formulations to calculate the gradient and the Hessian matrix.

$$g = \lim_{\beta \to \infty} \nabla f_\beta (b, w) = \begin{bmatrix} b - \mathbf{1}^\top Y(v)_+ \\ w - X^\top Y(v)_+ \end{bmatrix} = \begin{bmatrix} b \\ w \end{bmatrix} - A^\top Y(v)_+ \ , \tag{2}$$

$$H = \lim_{\beta \to \infty} \nabla^2 f_\beta (b, w) = \mathbf{I} + c \begin{bmatrix} \mathbf{1}^\top S \mathbf{1} & \mathbf{1}^\top S X \\ X^\top S \mathbf{1} & X^\top S X \end{bmatrix} = \mathbf{I} + c A^\top S A \ , \tag{3}$$

where

$$v = \mathbf{1} - Y(Xw + \mathbf{1}b) \ , \ (s_\infty)_j = \begin{cases} 1 & \text{if} \quad v_j > 0 \\ \frac{1}{2} & \text{if} \quad v_j = 0 \\ 0 & \text{if} \quad v_j < 0 \end{cases} \ , \ A = \begin{bmatrix} \mathbf{1} \mid X \end{bmatrix} \text{ and } S = diag(s_\infty) \ .$$

We use $g_{(i)}$ and $H_{(i)}$ to denote the gradient and the Hessian matrix at i^{th} iteration. The conventional SSVM algorithm is described in [16, 17].

By applying the *kernel trick* [4, 24], which uses a kernel function to represent the inner product of a pair of training instance images in the feature space, linear SSVM can be extended to the nonlinear SSVM easily. By the results of Generalized Support Vector Machines (GSVM) [20], we only need to do simply change the input data from the data matrix X to the nonlinear kernel matrix $Ker(X, X^\top)$ for the nonlinear case. The nonlinear decision function is

$$(Yu)^\top Ker(X, x) + b = 0 \ .$$

More details about SSVM can be found in [16, 17].

2 Observation of SSVM Updating

In SSVM training procedure, we observed that the values of $(s_\infty)_j$ is 0 for those X_j with $1 - Y_{jj}(X_j w + b) < 0$, where X_j is the j^{th} row of X which is a row vector in R^n and Y_{jj} is the label corresponding to X_j. This kind of training instances will not have any contribution in constructing the gradient and the Hessian matrix in Eq(2) and Eq(3). For each iteration, we partition the indices of entire training set T into the two sets, $SV(b_{(i)}, w_{(i)}, T)$ and $NSV(b_{(i)}, w_{(i)}, T)$.

$$SV(b_{(i)}, w_{(i)}, T) = \{ j \mid 1 - Y_{jj}(X_j w_{(i)} + b_{(i)}) \geq 0, (X_j, Y_{jj}) \in T, j = 1, 2, \dots, \ell \}$$

$NSV(b_{(i)}, w_{(i)}, T) = \{j \mid 1 - Y_{jj}(X_j w_{(i)} + b_{(i)}) < 0, (X_j, Y_{jj}) \in T, j = 1, 2, \ldots, \ell\}$

In the i^{th} iteration, some indices in $SV(b_{(i)}, w_{(i)}, T)$ may come from $SV(b_{(i-1)}, w_{(i-1)}, T)$ or $NSV(b_{(i-1)}, w_{(i-1)}, T)$. For example, if the indices in $NSV(b_{(i-1)}, w_{(i-1)}, T)$ are moved to $SV(b_{(i)}, w_{(i)}, T)$, its means that the instance corresponding its index change to tentative support vector from tentative non-support vector at i^{th} iteration.

We denote by $r_{(i)}$ the number of the instances which need to be removed and to be added at i^{th} iteration. In other words, r is the number of the indices migrate across the SV set and NSV set in each iteration during training SSVM. We use the set operators, intersection $\cap$ and difference $\setminus$ to describe the change between two iterations and define

$$K_{(i)} = SV(b_{(i-1)}, w_{(i-1)}, T) \cap SV(b_{(i)}, w_{(i)}, T) \ ,$$

$$D_{(i)} = SV(b_{(i-1)}, w_{(i-1)}, T) \setminus SV(b_{(i)}, w_{(i)}, T) \ ,$$

$$I_{(i)} = SV(b_{(i)}, w_{(i)}, T) \setminus SV(b_{(i-1)}, w_{(i-1)}, T) \ .$$

Then, we can have a simple representation

$$A_{SV_{(i-1)}} = \begin{bmatrix} A_{K_{(i)}} \\ A_{D_{(i)}} \end{bmatrix} \ , \quad A_{SV_{(i)}} = \begin{bmatrix} A_{K_{(i)}} \\ A_{I_{(i)}} \end{bmatrix} \ ,$$

where $A_{SV_{(i)}}$, $A_{K_{(i)}}$, $A_{D_{(i)}}$ and $A_{I_{(i)}}$ are the sub-matrix of A formed by selecting specific rows of A by $SV(b_{(i)}, w_{(i)}, T)$, $K_{(i)}$, $D_{(i)}$ and $I_{(i)}$, respectively. We note that $r_{(i)} = \mid D_{(i)} \mid + \mid I_{(i)} \mid$.

We compare the number of tentative support vectors ℓ_{sv} and r in each iteration on a certain datasets. According to Figure 1, we observe that r is decreasing rapidly iteration by iteration. This phenomenon is a key for reducing the computational cost for finding the Newton direction z.

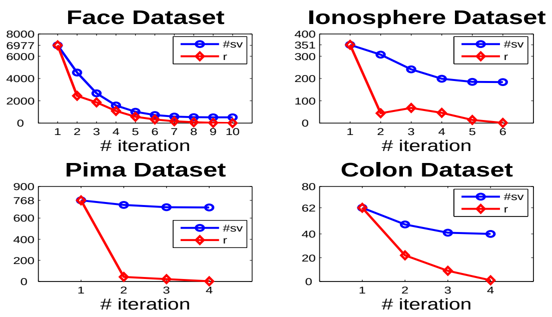

Fig. 1. The number of the support vectors ℓ_{sv} and r at each iteration

3 Hessian Inverse Updating Strategies

Although the main cost of finding the Newton direction z is solving the linear system $Hz = -g$ or finding the Hessian inverse, there are some useful tricks to compute the inverse of a specific matrix such as Sherman-Morrison-Woodbury formula [14]. In this section, we derived two efficient updating strategies for finding the Hessian inverse which is involved in training a SSVM by Sherman-Morrison-Woodbury formula, called *strategy I* and *strategy II*.

During the updating process of the Hessian inverse, we introduce the decremental and incremental operations. The decremental/incremental operation means that to remove/add the information of the instances that will be removed/added from/to current Hessian inverse.In our implementation, we perform the decremental operation first, then incremental operation.

Note that we denote n the dimensionality of the gradient and the Hessian matrix and use the notation A to represent $\begin{bmatrix}\mathbf{1} \mid X\end{bmatrix}$ in the rest of this paper for simplicity.

3.1 Hessian Inverse Updating Strategy I

Sherman-Morrison-Woodbury formula is

$$(T + UCV)^{-1} = T^{-1} - T^{-1}U(C^{-1} + VT^{-1}U)^{-1}VT^{-1} \ , \tag{4}$$

where

$$T \in R^{n\times n} \ , U \in R^{n\times r} \ , V \in R^{r\times n} \ , C \in R^{r\times r} \ .$$

It is the well known *rank-r updating scheme* that enables us to obtain the inverse of $(T + UCV)$ by inverting a low rank matrix $(C^{-1} + VT^{-1}U)$. We use Sherman-Morrison-Woodbury formula only when T^{-1} is available and the size of C is much smaller than T (i.e. $r \ll n$). Therefore, we can take the advantage of solving a small size inverse matrix to gain a much large size inverse matrix.

The Hessian matrix at i^{th} iteration is

$$H_{(i)} = I + cA^{\top}_{SV_{(i)}} S_{SV_{(i)}} A_{SV_{(i)}} \ , \tag{5}$$

where $A_{SV_{(i)}}$ is a sub-matrix of A formed by selecting specific rows of A by $SV(b_{(i)}, w_{(i)}, T)$, and $S_{SV_{(i)}}$ is a sub-squared matrix of S with corresponding indices in $SV(b_{(i)}, w_{(i)}, T)$. In the updating process from $H^{-1}_{(i-1)}$ to $H^{-1}_{(i)}$, we need to compute $H^{-1}_{K_{(i)}}$ by the decremental operation, and use $H^{-1}_{K_{(i)}}$ to compute $H^{-1}_{(i)}$ by the incremental operation. We can rewrite $H_{(i-1)}$ and $H_{(i)}$ as follows:

$$H_{(i-1)} = H_{K_{(i)}} + cA^{\top}_{D_{(i)}} S_{D_{(i)}} A_{D_{(i)}} \ ,$$

$$H_{(i)} = H_{K_{(i)}} + cA^{\top}_{I_{(i)}} S_{I_{(i)}} A_{I_{(i)}} \ .$$

Given $H^{-1}_{(i-1)}$ and $H_{K_{(i)}} = H_{(i-1)} - cA^{\top}_{D_{(i)}} S_{D_{(i)}} A_{D_{(i)}}$, we can apply Eq(4) to get $H^{-1}_{K_{(i)}}$. Applying Eq(4) again, we can obtain $H^{-1}_{(i)}$ easily. Based on the observation in Section 2, this updating scheme will becomes more efficiency than computing $H_{(i)}^{-1}$ directly.

We analyze the time and space complexities of one Newton step for *strategy I*, the time complexity is $O(n^2 r + nr^2 + r^3)$. If r is small enough, then the time complexity is reduced to $O(n^2)$. The space complexity is $O(n^2 + r^2)$ which is not included the data matrix itself. In fact, Sherman-Morrison-Woodbury formula has been applied to SVM training, e.g., [9, 19]. In most of the cases, Sherman-Morrison-Woodbury formula is applied to incremental learning, while decremental learning is less mentioned. However, decremental learning can be applied to some situations, for example, the k-fold cross-validation [2, 13, 15] and its special case: leave-one-out cross-validation [2, 5].

3.2 Hessian Inverse Updating Strategy II

In the process of the deriving of Sherman-Morrison-Woodbury formula, we can know that

$$\begin{bmatrix} T & U \\ V & C \end{bmatrix}^{-1} = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} \text{ , where } \begin{cases} M_1 = T^{-1} + T^{-1}U(C - VT^{-1}U)^{-1}VT^{-1} \\ M_2 = -T^{-1}U(C - VT^{-1}U)^{-1} \\ M_3 = -(C - VT^{-1}U)^{-1}VT^{-1} \\ M_4 = (C - VT^{-1}U)^{-1} \\ T \in R^{n\times n} \text{ , } U \in R^{n\times r} \text{ , } V \in R^{r\times n} \text{ , } C \in R^{r\times r} \end{cases} . \tag{6}$$

Eq(6) enables us to obtain the inverse of $\begin{bmatrix} T & U \\ V & C \end{bmatrix}$ by inverting T and $C^{-1} + VT^{-1}U$. If T^{-1} is already known and the size of C is much smaller than T, then Eq(6) can be computed efficiently. We want to compute the Hessian inverse by an $\ell_{sv} \times \ell_{sv}$ matrix instead of an $n \times n$ matrix. As above, we know the Hessian matrix of SSVM is Eq(5), then we substitute the Hessian matrix to Sherman-Morrison-Woodbury formula directly, we can obtain that

$$H_{(i)}^{-1} = I - cA_{SV_{(i)}}^{\top}(S_{SV_{(i)}}^{-1} + cA_{SV_{(i)}}A_{SV_{(i)}}^{\top})^{-1}A_{SV_{(i)}} \text{ ,}$$

where $H_{(i)}^{-1} \in R^{n\times n}$, $(S_{SV_{(i)}}^{-1} + cA_{SV_{(i)}}A_{SV_{(i)}}^{\top})^{-1} \in R^{\ell_{sv}\times\ell_{sv}}$.

The bottleneck of inverting the Hessian matrix is changed to $(S_{SV}^{-1} + cA_{SV}A_{SV}^{\top})^{-1}$ from $(I + cA_{SV}^{\top}S_{SV}A_{SV})^{-1}$. Therefore, we focus on the incremental and decremental operations for $(S_{SV}^{-1} + cA_{SV}A_{SV}^{\top})^{-1}$. Note that the size of $(S_{SV}^{-1} + cA_{SV}A_{SV}^{\top})^{-1}$ will be increased when some tentative support vectors are incremented and the size will be decreased when some tentative support vectors are decremented.

In incremental case, we assume T^{-1}, U, V, C are already known, and $\begin{bmatrix} T & U \\ V & C \end{bmatrix}^{-1}$ is unknown. Then

$$\begin{aligned}
\left(S_{SV_{(i)}}^{-1} + cA_{SV_{(i)}}A_{SV_{(i)}}^{\top}\right)^{-1} &= \left(\begin{bmatrix} S_{K_{(i)}} & O \\ O & S_{I_{(i)}} \end{bmatrix}^{-1} + c\begin{bmatrix} A_{K_{(i)}} \\ A_{I_{(i)}} \end{bmatrix}\begin{bmatrix} A_{K_{(i)}} \\ A_{I_{(i)}} \end{bmatrix}^{\top}\right)^{-1} \\
&= \begin{bmatrix} S_{K_{(i)}}^{-1} + cA_{K_{(i)}}A_{K_{(i)}}^{\top} & cA_{K_{(i)}}A_{I_{(i)}}^{\top} \\ cA_{I_{(i)}}A_{K_{(i)}}^{\top} & S_{I_{(i)}}^{-1} + cA_{I_{(i)}}A_{I_{(i)}}^{\top} \end{bmatrix}^{-1} \\
&= \begin{bmatrix} T & U \\ V & C \end{bmatrix}^{-1} ,
\end{aligned}$$

where $S_{SV_{(i)}}^{-1} + cA_{SV_{(i)}}A_{SV_{(i)}}^{\top}$ has the same form as Eq(6), therefore we can use the block matrix operation to obtain its inverse efficiently.

In decremental case, we assume $\begin{bmatrix} T & U \\ V & C \end{bmatrix}^{-1}$, U, V, C are already known (i.e. M_1, M_2, M_3, M_4, U, V and C are already known), and T^{-1} is unknown. Consider that

$$T^{-1}U(C - VT^{-1}U)^{-1}VT^{-1} = M_2M_4^{-1}M_3 ,$$

where M_2, M_3 and M_4 are defined in Eq(6), it can be represented by M_2, M_3 and M_4, then substitute to M_1 part,

$$M_1 = T^{-1} + M_2M_4^{-1}M_3 ,$$

therefore,

$$T^{-1} = M_1 - M_2M_4^{-1}M_3 . \tag{7}$$

We obtain T^{-1} by Eq(7). Then we can apply Eq(7) to decremental operation of the Hessian inverse as follows,

$$\begin{aligned}
(S_{SV_{(i-1)}}^{-1} + cA_{SV_{(i-1)}}A_{SV_{(i-1)}}^{\top})^{-1} &= \left(\begin{bmatrix} S_{K_{(i)}} & O \\ O & S_{D_{(i)}} \end{bmatrix}^{-1} + c\begin{bmatrix} A_{K_{(i)}} \\ A_{D_{(i)}} \end{bmatrix}\begin{bmatrix} A_{K_{(i)}} \\ A_{D_{(i)}} \end{bmatrix}^{\top}\right)^{-1} \\
&= \begin{bmatrix} S_{K_{(i)}}^{-1} + cA_{K_{(i)}}A_{K_{(i)}}^{\top} & cA_{K_{(i)}}A_{D_{(i)}}^{\top} \\ cA_{D_{(i)}}A_{K_{(i)}}^{\top} & S_{D_{(i)}}^{-1} + cA_{D_{(i)}}A_{D_{(i)}}^{\top} \end{bmatrix}^{-1} \\
&= \begin{bmatrix} T & U \\ V & C \end{bmatrix}^{-1} = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} ,
\end{aligned}$$

then

$$(S_{K_{(i)}}^{-1} + cA_{K_{(i)}}A_{K_{(i)}}^{\top})^{-1} = T^{-1} = M_1 - M_2M_4^{-1}M_3 .$$

Algorithm 1: The Algorithm for SSVM with the Updating Strategies

```
Input: Training set $T$, the initial model $(b_{(0)}, w_{(0)})$ and the initial Hessian
       inverse $H^{-1}_{(0)}$ (or $(S^{-1}_{SV} + cA_{SV}A^{\top}_{SV})^{-1}$) .

Repeat
  – Update the new Hessian inverse $H^{-1}_{(i)}$ by applying strategy I or strategy II to
    $H^{-1}_{(i-1)}$ .
  – Obtain the Newton direction $z_{(i)}$ by $z_{(i)} = -H^{-1}_{(i)} g_{(i)}$ .
  – Choose a step size by Armijo's rule, i.e., $\lambda_{(i)} = \max\{1, \frac{1}{2}, \frac{1}{4}, \ldots,\}$ such that
        $f_\beta(b_{(i)}, w_{(i)}) - f_\beta((b_{(i)}, w_{(i)}) + \lambda_{(i)} z_{(i)}) \geq -\delta\lambda_{(i)} g_{(i)} z_{(i)}$ ,
    where $\delta \in (0, \frac{1}{2})$ .
  – Update the current model by $(b_{(i+1)}, w_{(i+1)}) = (b_{(i)}, w_{(i)}) + \lambda_{(i)} z_{(i)}$ .
Until stopping criteria is satisfied.

Output: The final model $(b_{(i+1)}, w_{(i+1)})$ .
```

We note that the matrix $(S^{-1}_{SV_{(i-1)}} + cA_{SV_{(i-1)}} A^{\top}_{SV_{(i-1)}})^{-1}$ has been permuted according to the indices set $K_{(i)}$ and $D_{(i)}$. Thus, given $(S^{-1}_{SV_{(i-1)}} + cA_{SV_{(i-1)}} A^{\top}_{SV_{(i-1)}})^{-1}$, we can apply Eq(7) to get $(S^{-1}_{K_{(i)}} + cA_{K_{(i)}} A^{\top}_{K_{(i)}})^{-1}$. Then applying Eq(6), we can obtain $(S^{-1}_{SV_{(i)}} + cA_{SV_{(i)}} A^{\top}_{SV_{(i)}})^{-1}$ efficiently. Based on the observation in Section 2, this updating scheme will become more efficiency than computing $(S^{-1}_{SV_{(i)}} + cA_{SV_{(i)}} A^{\top}_{SV_{(i)}})^{-1}$ directly when $r \ll \ell_{sv}$.

We analyze the time and space complexities of one Newton iteration for *strategy II*. The time and space complexity are $O(\ell^2_{sv} r + \ell_{sv} r^2 + r^3 + n\ell_{sv} r + nr^2)$ and $O(\ell^2_{sv} + r^2)$, respectively. Note that the space complexity is not included the data matrix itself. In face, the decremental operation Eq(7) can be seen in [10]. However, it only discussed in the dual space (LSVM [21] and LSSVM [23]). We show that the decremental operation Eq(7) not only can apply to solve the dual problem, but also the primal problem.

We applied our two strategies to the SSVM training and rewrite the conventional SSVM algorithm, see Algorithm 1. Instead of solving a system of linear equations in conventional SSVM algorithm, we update the Hessian inverse directly by strategy I or II in Algorithm 1. It is the key difference between the conventional SSVM algorithm and Algorithm 1. We use the strategy I or II to update the new Hessian inverse $H^{-1}_{(i)}$ from $H^{-1}_{(i-1)}$ depending on whether the number of the support vectors ℓ_{sv} is much smaller than the dimensionality of dataset n. When $\ell_{sv} \ll n$, *strategy II* will be used. Otherwise, we use the *strategy I*.

Algorithm 2: Efficient version of the k-fold CV

```
Input: Training set $T$, the model $(b, w)$ trained by the entire training set $T$,
       $H^{-1}$ for strategy I (or $(S_{SV}^{-1} + cA_{SV}A_{SV}^\top)^{-1}$ for strategy II)
       corresponding to $(b, w)$.

$counter = 0$
For each fold $T_k \subset T$ ,
  – $Diff_{SV} = SV(b, w, T) \setminus SV(b, w, T \setminus T_k)$, where $SV(b, w, T)$ is defined in Section 2.
  – If $Diff_{SV} \neq \emptyset$,
      • Apply the Algorithm 1 with the input $(b, w)$, $T \setminus T_k$ and $H^{-1}$ (or
        $(S_{SV}^{-1} + cA_{SV}A_{SV}^\top)^{-1}$) .
      • Return the model $(b', w')$ without the instances in $T_k$ from Algorithm 1.
      • Compute the number of the misclassified in $T_k$ by the model $(b', w')$, $\#error$.
      • $counter = counter + \#error$.

Output: $k$-fold CV Correctness $(\ell - counter)/\ell$
```

3.3 Apply Our Updating Strategies to Cross-Validation

Although the computational cost of the k-fold CV is prohibitively expensive, especially for large-scale problems, there are some exist approaches to improve the LOOCV [2, 5, 11, 13, 19, 25]. In [5, 19], there is a basic assumption that if a data point is not an outlier, then with or without this data point will not affect the model too much. In other words, if the difference between two training datasets are only very few data points, the final Hessian inverses of these two learning models should be very close. Taking this advantage, we can generate the model for the entire dataset and use the decremental operation to have the exact LOOCV correctness. In Algorithm 2, we describe the efficient k-fold CV procedure for SSVM with the Hessian inverse updating strategies.

For the nonlinear case, we need to modify the Algorithm 2 as follows. If $Diff_{SV} \neq \emptyset$, for *strategy I*, we obtain the temporary Hessian inverse $\hat{H}^{-1}$ by applying Eq(7) to remove the attributes corresponding to the instances in T_k from H^{-1}, and reduce the size of H^{-1}; for *strategy II*, obtain the temporary $\hat{B}^{-1}$ by applying Eq(4) to remove the attributes corresponding to the instances in T_k from B^{-1}, where B stands for $(S_{SV}^{-1} + cA_{SV}A_{SV}^\top)$ for simplicity. At the same time, we obtain the temporary datasets $\hat{T}_k$, $\hat{T}$ and the temporary model $\hat{w}$ by removing the attributes corresponding to the instances in T_k from T_k, T and w, respectively. Apply the Algorithm 1 with the input $(b, \hat{w})$, $\hat{T} \setminus \hat{T}_k$ and $\hat{H}^{-1}$ (or $\hat{B}^{-1}$). Then return the model (b', w') without any information of the instances in $\hat{T}_k$ from Algorithm 1. Thus we can compute the number of the misclassified in $\hat{T}_k$ by the model (b', w').

4 Experiments

The k-fold cross-validation (k-fold CV) and the leave-one-out cross-validation (LOOCV) are two popular methods to estimate the generalization ability of the model, they provide the criterion for choosing a good parameter setting for a better generalization performance in a learning task. Both of them are computational intensive.

In this section, we demonstrated the effectiveness of SSVM with the Hessian inverse updating strategies on several datasets (see Table 1). Two different types of datasets are chosen to demonstrate the advantage of two different Hessian inverse updating strategies. The first one is $\ell \gg n$ type and the second one is $\ell \ll n$ type. In the nonlinear case, we generate an $\ell \times \ell$ kernel matrix by nonlinear mapping as our training data to show the effectiveness of two Hessian inverse updating strategies.

All our experiments were performed using the MATLAB(R2007b). The personal computer consists of 3.00GHz CPU, 3.24GB RAM. In our experiments, we focus on the training speed and will not too worry about the accuracy. Thus we avoid the tuning procedure and set the penalty c equal to 1. For the nonlinear case, we use the radial basis function kernel and set $\gamma = 0.1$. We compared our updating strategies with original SSVM and LIBSVM [7]. We reserve 2GB RAM of cache for LIBSVM in all experiments. The cache size will affect the LIBSVM training speed. All tables in this section, "SSVM" means original SSVM, "SSVM$_{\mathbf{I}}$" means that the *strategy I* is used in SSVM and "SSVM$_{\mathbf{II}}$" means that the *strategy II* is used. We indicate the best results in bold face.

In *http://www.work.caltech.edu/~htlin/program/libsvm/*, one part of this web page demonstrates how to do decremental learning in LIBSVM efficiently. We use this strategy to compute the LOOCV, and its implementation is done with LIBSVM 3.0. Since LIBSVM doesn't have special treatment for the k-fold CV, we obtain the results of the k-fold CV by the default setting and set 2GB cache for LIBSVM.

In Table 2, we can see that the computational cost of the LOOCV can be significantly reduced if a good initial solution and the Hessian inverse (or$(S_{SV}^{-1} + cA_{SV}A_{SV}^{\top})^{-1}$) are available. These experiments indicate that the incremental/decremental schemes are the efficient approach for performing the LOOCV [19]. In Table 2, it also shows that the accuracies of SSVM, SSVM$_{\mathbf{I}}$ and SSVM$_{\mathbf{II}}$

Table 1. Datasets description

	#instances	#features
Face	6977	361
Ionosphere	351	34
Pima	768	8
Colon	62	2000
Leukemia	72	7129

Table 2. The consuming time and accuracies of the LOOCV of SSVM, SSVM$_{\mathbf{I}}$, SSVM$_{\mathbf{II}}$ and LIBSVM on several datasets

LOOCV	**- Time (sec.) -**				**- Accuracy (%) -**			
`(Linear)`	`SSVM`	`SSVM`$_{\mathbf{I}}$	`SSVM`$_{\mathbf{II}}$	`LIBSVM`	`SSVM`	`SSVM`$_{\mathbf{I}}$	`SSVM`$_{\mathbf{II}}$	`LIBSVM`
`Face`	22.41	**14.32**	25.48	1041.11	97.73	97.73	97.73	98.08
`Ionosphere`	0.16	**0.09**	0.23	0.53	88.89	88.89	88.89	87.18
`Pima`	0.37	**0.23**	8.86	5.53	76.56	76.56	76.56	76.56
`Colon`	38.81	2.97	**0.59**	0.64	83.87	83.87	83.87	80.65
`Leukemia`	89.37	62.72	2.61	**2.42**	98.61	98.61	98.61	97.22
`(Nonlinear)`	`SSVM`	`SSVM`$_{\mathbf{I}}$	`SSVM`$_{\mathbf{II}}$	`LIBSVM`	`SSVM`	`SSVM`$_{\mathbf{I}}$	`SSVM`$_{\mathbf{II}}$	`LIBSVM`
`Face`	4508.58	1652.27	**1002.83**	35786.39	99.76	99.76	99.76	99.83
`Ionosphere`	1.57	0.87	**0.41**	0.94	95.73	95.73	95.73	93.73
`Pima`	95.22	69.77	63.72	**19.89**	77.21	77.21	77.21	75.52

are the exactly same. Similarly, both strategies also can applied to perform the k-fold CV efficiently, and we set k equal to 10 in our experiments of the k-fold CV. In Table 3, it can be seen that the experimental results of the k-fold CV are very similar to the results of the LOOCV experiments.

5 Conclusions

In this paper, we proposed two Hessian inverse updating strategies to improve the computational cost of finding the Hessian inverse. In the cross-validation procedure, we introduced the strategies for the leave-one-out cross-validation and the k-fold cross-validation. If the number of the indices migrate across the SV set and NSV set at each fold is much smaller than dimensionality of the input space, then the cost of updating the Hessian inverse and finding the Newton direction will be reduced dramatically. This benefit will become more significantly when the data is in a highly dimensional space.

Our updating strategies can be applied to any learning algorithm which it solved iteratively and involved the Hessian inverse in each iteration, such as ϵ-SSVR, LSSVM, Training SVM in the Primal, Second-Order Online Perceptron Algorithm and so forth.

Acknowledgments. We would like to thank the anonymous referees for providing constructive comments.

Table 3. The consuming time and accuracies of the 10-fold CV of SSVM, SSVM$_{\mathbf{I}}$, SSVM$_{\mathbf{II}}$ and LIBSVM on several datasets

10-fold CV	**- Time (sec.) -**				**- Accuracy (%) -**			
(Linear)	SSVM	SSVM$_{\mathrm{I}}$	SSVM$_{\mathrm{II}}$	LIBSVM	SSVM	SSVM$_{\mathrm{I}}$	SSVM$_{\mathrm{II}}$	LIBSVM
Face	4.57	**3.94**	5.00	43.41	97.76	97.76	97.76	98.01
Ionosphere	0.07	**0.04**	0.09	0.11	87.68	87.68	87.68	87.46
Pima	0.04	**0.03**	0.66	0.19	76.41	76.41	76.41	76.56
Colon	21.73	4.77	0.37	**0.33**	83.75	83.75	83.75	82.26
Leukemia	196.98	40.30	1.03	**0.97**	97.80	97.80	97.80	97.22
(Nonlinear)	SSVM	SSVM$_{\mathrm{I}}$	SSVM$_{\mathrm{II}}$	LIBSVM	SSVM	SSVM$_{\mathrm{I}}$	SSVM$_{\mathrm{II}}$	LIBSVM
Face	314.51	75.11	**36.37**	186.53	99.72	99.72	99.72	99.81
Ionosphere	0.38	0.23	0.15	**0.12**	95.54	95.54	95.54	93.73
Pima	3.48	3.15	3.83	**0.42**	77.11	77.11	77.11	75.39

References

1. Alon, U.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. In Proceedings of the National Academy of Science, 96, pp. 6745–6750. USA (1999)
2. Arlot, S., Celisse, A.: A survey of cross-validation procedures for model selection. Statistics Surveys, 4, 40-79 (2010)
3. UCI repository of machine learning databases, `http://archive.ics.uci.edu/ml/index.html`
4. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2), 121–167 (1998)
5. Cawley, G.C., Talbot, N.L.C.: Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. Neural Networks, 17(10), 1467–1475 (2004)
6. Cesa-Bianchi, N., Conconi, A., Gentile, C.: A second-order perceptron algorithm. SIAM Journal on Computing, 34, 129–140 (2005)
7. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001) Software availabe at `http://www.csie.ntu.edu.tw/$\sim$cjlin/libsvm`
8. Chapelle, O.: Training a support vector machine in the primal. Neural Computation, 19(5), 1155–1178 (2007)
9. Chi, H.M., Ersoy, O.K.: Recursive update algorithm for least squares support vector machines. Neural Processing Letters, 17(2), 165–173 (2003)
10. Duan, H., Li, H., He, G., Zeng, Q.: Decremental learning algorithms for nonlinear langrangian and least squares support vector machines. In Proceedings of the First International Symposium on Optimization and Systems Biology (OSB'07), pp. 358–366, Beijing, China (2007)
11. Franc, V., Laskov, P., Műller, K.R.: Stopping conditions for exact computation of leave-one-out error in support vector machines. In Proceedings of the 25th International Conference on Machine Learning, pp. 328–335. ACM, New York (2008)
12. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A.: Molecular classification of

cancer: class discovery and class prediction by gene expression monitoring. Science 286, 531–537 (1999)

13. Golub, G.H., Heath, M., Wahba, G.: Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics, 21, 215-224 (1979)
14. Hager, W.W.: Updating the inverse of a matrix. In SIAM Review, 31(2), 221–239 (1989)
15. Hunag, C.-H., Lee, Y.-J., Lin, D.K.J., Huang, S.-Y.: Model selection for support vector machines via uniform design. The special issue on Machine Learning and Robust Data Mining of Computational Statistics and Data Analysis (2006)
16. Lee, Y.-J.: Support vector machines in data mining. PhD Thesis, The University of Wisconsin (2001)
17. Lee, Y.-J., Mangasarian, O.L.: SSVM: A smooth support vector machine for classification. Computational Optimization and Applications, 20(1), 5–22 (2001)
18. Lee, Y.-J., Hsieh, W.-F., Huang, C.-M.: ε–SSVR: A smooth support vector machine for ε–insensitive regression. IEEE Transactions on Knowledge and Data Engineering, 678–685 (2005)
19. Liang, Z., Li, Y.-F.: Incremental support vector machine learning in the primal and applications. Neurocomputing, 72, 2249–2258 (2009)
20. Mangasarian, O.L.: Generalized support vector machines. In Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D., editors, Advances in Large Margin Classifiers, pp. 135–146. Cambridge, MA, MIT Press (2000)
21. Mangasarian, O.L., Musicant, D.R.: Lagrangian support vector machines. Journal of Machine Learning Research, 1, 161–177 (2001)
22. MIT-CBCL Face Recognition Database, `http://cbcl.mit.edu/software-datasets/FaceData2.html`
23. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classilers. Neural Processing Letters, 9(3), 293–300 (1999)
24. Vapnik, V.N.: The nature of statistical learning theory. Springer (2000)
25. Zhang, T.: A leave-one-out cross validation bound for kernel methods with applications in learning. In Proceedings of the 14th Annual Conference on Computational Learning Theory, Lecture Notes in Artificial Intelligence 2111, pp. 427–443. (2001)