Speed-Up LOO-CV with SVM Classifier

G. Lebrun[1], O. Lezoray[1], C. Charrier[1], and H. Cardot[2]

[1] LUSAC EA 2607, groupe Vision et Analyse d'Image, IUT Dépt. SRC,
120 Rue de l'exode, Saint-Lô, F-50000, France
{gilles.lebrun, c.charrier, o.lezoray}@chbg.unicaen.fr
[2] Laboratoire Informatique (EA 2101), Université François-Rabelais de Tours,
64 Avenue Jean Portalis, Tours, F-37200, France
hubert.cardot@univ-tours.fr

Abstract. Leave-one-out Cross Validation (LOO-CV) gives an almost unbiased estimate of the expected generalization error. But the LOO-CV classical procedure with Support Vector Machines (SVM) is very expensive and cannot be applied when training set has more that few hundred examples. We propose a new LOO-CV method which uses modified initialization of Sequential Minimal Optimization (SMO) algorithm for SVM to speed-up LOO-CV. Moreover, when SMO's stopping criterion is changed with our adaptive method, experimental results show that speed-up of LOO-CV is greatly increased while LOO error estimation is very close to exact LOO error estimation.

1 Introduction

LOO-CV is an useful measure to estimate the generalization of an inducer [1]. Model selection is the main aim of LOO measure [2], especially when dataset size is considered as too small to split it into training and test sets. SVM is an efficient inducer, but training time increases quickly with training set size [3] and it would be a bad candidate for model selection with direct LOO-CV. But others properties of SVM made it a good candidate for smart LOO-CV [4,5]. Decoste et al [4] and others [2,5] have proposed new methods to speed-up exact (or very close) evaluation of LOO error with SVM. All these methods are based either on changing initialization, stopping criterion, or both of SMO algorithm. Next sections present an overview of speed-up LOO-CV methods and explain in what way our method improves them. Many experimental results are also described to highlight the efficiency of our method and to compare it with previous ones.

2 SVM and SMO Overview

Consider a data set S ($S = \{z_1, ..., z_m\} = \{(x_1, y_1), ..., (x_m, y_m)\}$) with m instances (or examples) where each data information z_i is a couple (x_i, y_i) with $y_i \in \{+1, -1\}$ and $x_i \in \mathbb{R}^n$. The main task for training SVM is to solve the following dual quadratic optimization problem [6]:

$$\min_{\alpha} W(\alpha) = \frac{1}{2} \sum \alpha_i \alpha_j Q_{ij} - \sum \alpha_i \tag{1a}$$

$$\text{subject to } 0 \leq \alpha_i \leq C \text{ and } \sum \alpha_i y_i = 0 \tag{1b}$$

E. Corchado et al. (Eds.): IDEAL 2006, LNCS 4224, pp. 108–115, 2006.

with $Q_{ij} = y_i y_j K(x_i, x_j)$ and $K(x_i, x_j)$ is a kernel function. Let us define (see [7] for more details on those formulations):

$$G_i = \sum \alpha_j Q_{ij} - 1 \quad \text{(2a)}$$

$$I_{\text{up}}(\alpha) \equiv \{t | \alpha_t < C, y_t = +1 \text{ or } \alpha_t > 0, y_t = -1\} \quad \text{(2b)}$$

$$I_{\text{low}}(\alpha) \equiv \{t | \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = +1\} \quad \text{(2c)}$$

$$m(\alpha) = -y_{i_1} G_{i_1} | i_1 = \underset{i \in I_{\text{up}}}{\text{argmax}} -y_i G_i, M(\alpha) = -y_{i_2} G_{i_2} | i_2 = \underset{i \in I_{\text{low}}}{\text{argmin}} -y_i G_i \quad \text{(2d)}$$

A solution α is optimal for problem (1) if and only if [7]

$$m(\alpha) < M(\alpha) \quad (3)$$

Let α^* be an optimal solution and $\{I_0, I_{\text{m}}, I_{\text{bound}}\}$ a partition of examples indexes $I = \{1, \ldots, m\}$ in function of α values with $I_0(\alpha) \equiv \{t | \alpha_t = 0\}$, $I_{\text{m}}(\alpha) \equiv \{t | 0 < \alpha_t < C\}$, $I_{\text{bound}}(\alpha) \equiv \{t | \alpha_t = C\}$. A decision function h produced by SVM has the following expression:

$$h(x) = \text{sign}\left(f(x)\right), f(x) = \sum \alpha_i^* y_i K(x_i, x) + b^* \quad (4)$$

with $b^* = -y_i G_i^*, \ \forall i \in I_{\text{m}}(\alpha^*)$ and f the output of SVM [6].

An efficient iterative algorithm named SMO was proposed by Platt [3] to find optimal solution of (1) by using test condition (3). The main idea of this algorithm is that at each iteration only two variables α_{i_1} and α_{i_2} are modified to decrease (1a) value. The synopsis of SMO is given by algorithm 1. $\epsilon = 10^{-3}$ and $\forall i : \alpha_i = 0, G_i = -1$ are classical default initialization values for SMO. As SMO is only asymptotically convergent, the previous ϵ value is an efficient admissible choice for checking optimality [7]. Without any kind of information on optimal solution localization, $\alpha = 0$ ($W(\alpha) = 0$) is a efficient starting solution [3]. Mainly, because for other α values, G_i values must be computed using (2a), which is time expensive. But also because objective initial value could be worst (*i.e.* $W(\alpha) > 0$) and increase the number of iterations for convergence. In order to have the lowest number of iterations for SMO, the procedure BestCouple must select i_1 and i_2 which produce the maximum decrease of (1a) [3]. Variation of (1a) when only two variables are modified is equal to:

$$\begin{aligned} \Delta W(i_1, i_2) = {} & \Delta\alpha_{i_1} G_{i_1} + \Delta\alpha_{i_2} G_{i_2} + \Delta\alpha_{i_1} \Delta\alpha_{i_2} Q_{i_1, i_2} \\ & + \tfrac{1}{2}\left(\Delta\alpha_{i_1}^2 Q_{i_1, i_1} + \Delta\alpha_{i_2}^2 Q_{i_2, i_2}\right) \end{aligned} \quad (5)$$

Algorithm 1. SMO(α,G,ϵ)

while $m(\alpha) - M(\alpha) > \epsilon$ **do**
  (i_1,i_2)=BestCouple()
  ($\Delta\alpha_{i_1}$,$\Delta\alpha_{i_2}$)=OptimalVariation(i_1,i_2)
  (α,G)=Update(α,G,(i_1,$\Delta\alpha_{i_1}$),(i_2,$\Delta\alpha_{i_2}$))
end while

Search of optimal couple with equation (5) is time expensive ($O(m^2)$), and heuristics were proposed (see [7] and references in). The most common one is to select the couple which maximum violates the stopping criterion of SMO (*i.e.* respectively i_1 and i_2 in equations (2d)). A recent alternative is to select the first α_i using previous heuristic (i_1 for example) and to use equation (5) to select the second [7]. After selecting good candidates, the OptimalVariation procedure computes $\Delta\alpha_{i_1}$ and $\Delta\alpha_{i_2}$ values in order to have the maximal decrease of W (see [3,7] for more details). The Update procedure uses (6) to compute variations of G values in function of $\Delta\alpha_{i_1}$ and $\Delta\alpha_{i_2}$.

$$\forall j : \Delta G_j = \Delta\alpha_{i_1} Q_{j,i_1} + \Delta\alpha_{i_2} Q_{j,i_2} \tag{6}$$

3 Speed-Up LOO-CV

LOO-CV definition: Let h_θ^S be the decision function produced by a learning algorithm with training set S. θ is the set of parameters (also named model) used by the training algorithm. The error e_{LOO} measured by LOO-CV procedure is defined by:

$$e_{\mathrm{LOO}}(\theta) = \frac{1}{m} \sum\nolimits_{i=1}^{m} l\left(h_\theta^{S_i}(x_i), y_i\right) \tag{7}$$

with $S_i = S \backslash \{z_i\}$ training sets and $l(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{else} \end{cases}$ the loss function.

LOO error and SVM relation: For many training algorithms, the estimation of e_{LOO} is realized by m trainings on S_i datasets. A better way to do this with SVMs is to first realize a training with all the m examples. This first optimal solution α^* provides several useful informations [5]. Those informations allow to determine values of $l(h_\theta^{S_i}(x_i), y_i)$ without any training with several datasets S_i. For example: $h_\theta^{S_i}(x_i) = y_i$, if $\alpha_i = 0$ in α^* (see [5] for more details). Let I_{LOO} denote set of examples for which SVM trainings with datasets S_i are necessary. Experimental results in [5] illustrate how the size of those set changes in function of θ. Those results show that the rate $|I_{\mathrm{LOO}}|/m$ is not negligible for many models θ. Then, model selection using LOO-CV is always time expensive when m increases, even if only $|I_{\mathrm{LOO}}|$ SVM trainings must be realized. To speed-up those SVM trainings, two possibilities exist: (1) the solution of first training with S could help to determine a better $\tilde{\alpha}$ starting solution for SMO (and associated $\tilde{G}$ values), (2) ϵ stopping value of SMO could be increased. For the next sections, let α_S, G_S denote the final results of SMO training with S.

Alpha Seeding methods: DeCoste and Wagstaff named Alpha Seeding (AS-SMO) a method which determines $\tilde{\alpha}$ next SVM initial solution in function of previous SVM trainings [4]. For LOO-CV with SVM, a starting solution $\tilde{\alpha}$ with $\tilde{\alpha}_i = 0$ is deduced from α_S to realize SVM training with S_i datasets. $\tilde{\alpha}_i$=0 reflects the fact that example z_i is removed from training set S. Moreover, the modification of α_S must respect contraints (1b) to produce a $\tilde{\alpha}$ feasible solution. To have speed-up effect, starting solution $\tilde{\alpha}$ must be near optimal solution $\tilde{\alpha}^*$.

Initially proposed method [4] consists in uniformly adding an equal portion of α_i to each in-bound α_j within the same class (*i.e.* $j \in I_{\mathrm{m}}$, $y_j = y_i$ and $j \neq i$). α_i is then decreased in the same proportion. Due to constraint (1b), $\tilde{\alpha}_i = 0$ could fail, then this action is repeated with remaining in-bound $\tilde{\alpha}_j$ until $\tilde{\alpha}_i = 0$. The main problem with this method is that many α_i variables are modified. The computation cost for updating $\tilde{G}$ values from G_S is then too high. Lee et al noticed this problem and proposed a method which changes only few variables [5]. The main idea is to redistribute an amount δ from α_i to α_k (*i.e.* $\Delta\alpha_i = -\delta$ and $\Delta\alpha_k = y_i y_k \delta$) and to select $k \in I_{\mathrm{m}}(\alpha_S)$ in order to reduce magnitude variation of G. This corresponds to solve: $k = \underset{k \in I_{\mathrm{m}}, k \neq i}{\operatorname{argmin}} (\max_j |\Delta G_j|)$ with $k \equiv i_1$ and $i \equiv i_2$ in (6). This problem has however a too high complexity. Lee proposes as a heuristic to look only at variations of G_k (see [5] for more details). This corresponds to solve the simplified problem : $k = \underset{k \in I_{\mathrm{m}}, k \neq i}{\operatorname{argmin}} (|\Delta G_k|)$ and to make the hypothesis that all other ΔG_i have same or less magnitude variations than $|\Delta G_k|$ when α_k is modified. This procedure is repeated until $\tilde{\alpha}_i = 0$.

New AS-SMO method: Previous studies highlight that an efficient AS-SVM method must modify the lesser possible variables in α_S for a reduce of G update cost. It must also produce a starting solution $\tilde{\alpha}$ for which SMO algorithm has a minimum of iterations to reach the stopping condition. DeCoste et al method [4] focuses only on second key point by making the hypothesis that building a close and valid solution $\tilde{\alpha}$ from α_S produces a solution near the optimal, but neglecks completely the first point. Lee et al method [5] advantages the first point and manages the second point by taking into account heuristic informations. Our proposed method tries to deal with those two key points at the same time. The main idea is to search which α_k variable allows by its modification, to decrease α_i to zero in respect to contraints (1b) and has the lowest $W(\tilde{\alpha})$ (i.e., nearest to optimal solution $\tilde{\alpha}^*$). The synopsis of this new method is resumed in algorithm 2 with $I_1(\alpha, i) = \{k | 0 \leq \alpha_k + y_k y_i \alpha_i \leq C, k \neq i\}$ the set of α_k variables which allow to have $\alpha_i = 0$ under contraint (1b) by modifying only one of them, $I_\delta(\alpha, i) = \{k | \exists \delta > 0 : 0 \leq \alpha_k + y_k y_i \delta \leq C, k \neq i\}$ the set of α_k variables which allow to decrease α_i and $\delta_{max}(k, i)$ the maximal decrease of α_i when only α_k is modified. ($\tilde{\alpha}$,$\tilde{G}$)=AS-SMO(α_S,G_S,i) are initialization values of

Algorithm 2. AS-SMO(α,G,i)

while $\alpha_i > 0$ **do**
  if $I_1(\alpha, i) \neq \emptyset$ **then**
    $\Delta\alpha_i = -\alpha_i$, $\Delta\alpha_k = y_k y_i \alpha_i$ with $k = \underset{k \in I_1(\alpha, i)}{\operatorname{argmax}} -\Delta W(k, i)$
  else
    $k = \underset{k \in I_\delta(\alpha, i)}{\operatorname{argmax}} [\Delta W_{\max} - \Delta W(k, i)] \cdot \delta_{\max}(k, i)$ with $\Delta W_{\max} = \underset{k \in I_\delta(\alpha, i)}{\operatorname{argmax}} \Delta W(k, i)$
    $\Delta\alpha_k = y_k y_i \delta_{\max}(k, i)$, $\Delta\alpha_i = -\delta_{\max}(k, i)$
  end if
  ($\tilde{\alpha}$,$\tilde{G}$)=Update(α,G,(i,$\Delta\alpha_i$),(k,$\Delta\alpha_k$))
end while

SMO when training set S_i is used. In general case, when I_1 is not empty, it is possible to have $\alpha_i = 0$ by modifying only another α_k variable. As regards the first key point, this action has the lowest cost. When I_1 has more than one element, which is generally true, the modified α_k variable must be the one which produces an $\tilde{\alpha}$ starting SMO solution which has the lower $W(\tilde{\alpha})$ value to deal with the second key point. $\Delta\alpha_i = -\alpha_i$ and $\Delta\alpha_k = y_k y_i \alpha_i$ by using contraint (1b). $\Delta W(i,k)$ is determined directly by using (5) and the optimal choice of k has a time complexity of $O(m)$. Paying attention to this method highlights a strong similarity with one step of SMO algorithm, especially by comparing it with *second order* SMO step in [7]. Computing cost for determination of $\tilde{\alpha}$ from α_S is close to one SMO iteration for general case. In the rare case for which $|I_1| = \emptyset$, more than one α variable must be modified. The approach is a greedy one and is guided by a criterion which makes a trade-off between bringing $\tilde{\alpha}$ close to optimal and decreasing greatly α_i in order to have few α modified variables.

Stopping ϵ value change: SMO implementations [8] use low values of ϵ to ensure efficient solution as regards theoretical convergence criterion. This has for effect that the reponse of decision function does not change for many iterations [9]. It is then natural to want to increase ϵ value in order to stop earlier SMO algorithm. But the problem is to select an efficient ϵ. Too high ϵ leads too early stopping and the solution does not correspond to the SVM problem. Too low ϵ does not sufficiently decrease the number of iterations of SMO. Lee [2] compares e_{LOO} variations between $\epsilon = 10^{-1}$ and classical $\epsilon = 10^{-3}$ with several datasets and hyperparameters values. Conclusion is that e_{LOO} estimations are very similar for both ϵ values, but training time is greatly reduced using $\epsilon = 10^{-1}$. In [5] an adaptive ϵ method is proposed. The main idea is to stop SMO algorithm, step by step, for $\epsilon \in [10^{-1}, 10^{-2}, 10^{-3}]$ and to use a heuristic criterion to test if SMO must continue (see [5] for more details). The advantage is that e_{LOO} estimation is more accurate than with the previous method. The disadvantage is that speed-up is reduced mainly because more than one evaluation of the SVM output (4) must be realized [5].

New adaptive method: Taking into account that an efficient stopping ϵ value for SMO training with dataset S must be also efficient for training with datasets S_i, due to the closeness of S and S_i datasets, our new proposed method uses the first training not only to produce an α_S helpful guidline for SMO, but also to deduce an efficient ϵ_{LOO} for LOO-CV with AS-SVM. Let $W_t \equiv W(\alpha_t)$, $Q_t = |(W_t - W_\infty)/W_\infty|$ and $\Delta M_t \equiv m(\alpha_t) - M(\alpha_t)$ respectively denote values of the objective function, a proximity measure of optimal solution and maximum violation of KKT criterion at SMO iteration t. As W_∞ could not be evaluated, SMO classic ending is used: $W_\infty = W_{t_{\max}}$ with $t_{\max}$ the number of SMO iterations when training set is S and $\epsilon = 10^{-3}$. Let t_ϵ be the first SMO iteration for which $\forall t \geq t_\epsilon : \Delta M_t < \epsilon$. An ϵ choice is efficient if Q_{t_ϵ} is close to zero. Let $Q_{t_\epsilon} \leq 10^{-3}$ be a transcription of "Q_{t_ϵ} is close to zero" and $t_S = \max_{1 \leq t \leq t_{\max}} \{t | Q_t > 10^{-3}\}$ be the last SMO iteration with training set S for which this condition is not true. The ϵ_{LOO} corresponding choice is determined

by using W_t and ΔM_t recorded values with this one SMO's training:

$$\epsilon_{\mathrm{LOO}} = \min_{1 \leq t \leq t_S} \Delta M_t \tag{8}$$

4 Experimental Results

For experiments, four datasets are used. Three are from Statlog collection: **Australian (Au)**, **Heart (He)** and **German (Ge)**. Fourth one is from UCI collection: **Adult (Ad)**. General information about used datasets are provided in table 1 where n and $|S|$ are respectively the number of features of examples and the training set sizes.

Table 1. Datasets information

Data sets (S)	n	$\lvert S\rvert$
Australian (Au)	14	390
Heart (He)	13	180
German (Ge)	24	400
Adult (Ad)	123	1605

Let T_M^{LOO} and n_M^{niter} be respectively total training time and total number of iterations to evaluate LOO error by using a SMO initialization method M. For all this section, gain $\mathrm{G}_{\mathrm{LOO}}^{\mathrm{T}} = T_{M_2}^{\mathrm{LOO}}/T_{M_1}^{\mathrm{LOO}}$ ($\mathrm{G}_{\mathrm{LOO}}^{\mathrm{iter}} = n_{M_2}^{\mathrm{niter}}/n_{M_1}^{\mathrm{niter}}$) corresponds to the gain in time (resp. total number of iterations) realized by using a given SMO initialization method M_1 in comparison of classical SMO initialization method M_2 (*i.e* $\alpha = 0$,$\epsilon = 10^{-3}$) for e_{LOO} computation. To illustrate the robustness of our method, experiments are conducted for a great number of θ hyperparameters values. Used procedure corresponds to the well known *grid search* method [8]. Tested values for C SVM hyperparameter are in: $\{2^x | x \in [-2, ..., 12]\}$. For Gaussian kernel: $k(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2)$, tested values for γ are in: $\{2^x | x \in [-10, ..., 2]\}$. For Polynomial kernel: $k(x_i, x_j) = (1 + < x_i, x_j >)^{\gamma}$, tested values for γ are in: $[2, ..., 6]$. Statistical measures within minimal, maximal, average and standard deviation for all tested models (C,γ) are respectively denoted by min, max, STD and AVG acronyms in tables 2. Experimental results with use of Gaussian or Polynomial kernel are mentioned in tables 2 and 3 by using respectively G or P letters between parenthesis after dataset's abreviation name.

Proposed method: First experimentation highlights the speed-up effect of using SMO $\tilde{\alpha}$ initialization produced by our method ($\epsilon_{\mathrm{LOO}} = 10^{-3}$ for M_1 also). Table 2 (up-left) gives statistical measures of gain $\mathrm{G}_{\mathrm{LOO}}^{\mathrm{T}}$ with different data sets. Results here, show that $\tilde{\alpha}$ deduced from α_S is an efficient starting solution for SMO in average and in the worst case it is not as bad as $\alpha = 0$. However, the global speed-up for model selection is not sufficient for LOO error evaluation when training size grows (too expensive with **Adult** dataset for exemple). Second experimentation focuses on the effect of combining $\tilde{\alpha}$ and ϵ_{LOO} SMO initialization of our method. Table 2 (up-right) gives statistical measures of ϵ_{LOO} variation. In a similar way to first experimentation, table 2 (bottom-left) gives statistical information of gain variations in function of θ model. To estimate gain with **Adult** dataset, we made the hypothesis that training time (number of iterations) with $m - 1$ and m exemples are identical with SMO classical

Table 2. Statistical measures of: G^{T}_{LOO} with only our alpha seeding method (up-left) and with our complete method (bottom-left), ϵ_{LOO} with our adaptive ϵ_{LOO} method (up-right) and Δe_{LOO} between an adaptive ϵ_{LOO} and a fixed $\epsilon_{LOO} = 10^{-3}$ for stopping SMO (bottom-right)

G^{T}_{LOO}	**min**	**max**	**AVG**	**STD**
Au(G)	0.95	119.1	**9.67**	18.97
He(G)	1.25	81.38	**8.24**	14.90
Ge(G)	1.34	78.85	**7.13**	8.32

G^{T}_{LOO}	**min**	**max**	**AVG**	**STD**
Au(G)	12.72	364.44	**99.01**	60.79
He(G)	25.08	379.59	**52.10**	63.75
Ge(G)	17.96	224.43	**102.43**	50.70
Ad(G)	82.30	26580	**1587**	4159

ϵ_{LOO}	**min**	**max**	**AVG**	**STD**
Au(G)	0.057	1.665	**0.164**	0.186
He(G)	0.062	1.785	**0.178**	0.216
Ge(G)	0.056	1.986	**0.149**	0.218
Ad(G)	0.058	1.854	**0.287**	0.300

Δe_{LOO}	**min**	**max**	**AVG**	**STD**
Au(G)	-2.0%	+3.67%	**+0.16%**	±0.86%
He(G)	-1.67%	+5.0%	**+0.59%**	±1.05%
Ge(G)	-0.5%	+7.5%	**+1.23%**	±1.85%

initialization[1]. The variation of e_{LOO} with SMO's classical initialization have also been measured. Table 2 (bottom-right) gives statistical information for Statlog collection datasets. Results show that combining efficient alpha seeding and adaptive increase of ϵ stopping criterion permits to speed-up greatly LOO procedure. Speed-up could be spectacular for some θ models, in particular when training set size increases. Even with worse θ cases, the speed-up are not negligible and are in favour of greater training sets again. Moreover, e_{LOO} evaluations have small perturbations when stopping criterion is increased with our method. When results of Δe_{LOO} are examined in more detail, variations close to min or max values in bottom-right table 2 are almost always located in regions with high e_{LOO}. Consequently, model selection is very few impacted by those variations and e_{LOO} value for selected model is very close to values found with $\epsilon = 10^{-3}$. Average values in up-right table 2 highlight the global efficiency of Lee's $\epsilon = 0.1$ heuristic choice, but also the limit of a fixed value.

Comparaison with previous methods: First experimental comparisons have for objective to highlight difference between our adaptive ϵ_{LOO} stopping criterion and fixed $\epsilon_{LOO} = 10^{-1}$. Table 3 resumes results from this comparison. Our alpha seeding method is used for those experiments. Looking at table 3, it is obvious that SMO algorithm stopped earlier in average with our method without important increase of e_{LOO} deviation. Second experimental comparison has for objective to highlight difference between the three alpha seeding methods (ϵ_{LOO} adaptive is used for all of them). In table 3, n_{α_S} corresponds to the number of variables α modified to produce $\tilde{\alpha}$. It is a good indicator of G update cost. G^{iter}_{LOO} (G^{T}_{LOO}) is gain between an alpha seeding method and classical ($\alpha = 0$, $\epsilon = 10^{-3}$) SMO initialization. Results in table 3 show that DeCoste et al method can produce very efficient starting solution (**He**(G) and **Ge**(G)), but update cost of (6) is too high and penalizes G^{T}_{LOO}. It is particularly obvious with **Adult**

[1] Inspect of experimental results with the three Stalog datasets corroborate this hypothesis (*i.e.* variation of training times are negligible).

Table 3. Comparison between: (a) our adaptive or fixed ϵ value for SMO stopping criterion and (b) the three alpha seeding methods: AS_1, AS_2 and AS_3 which are respectively our method, Lee et al [5] and DeCoste et al [4] method

	(a) SMO's ϵ stopping criterion				(b) AS-SMO methods								
	adaptive		fixed		n_{α_S}			G_{LOO}^{iter}			G_{LOO}^{T}		
$S(K)$	G_{LOO}^{iter}	Δe_{LOO}	G_{LOO}^{iter}	Δe_{LOO}	AS_1	AS_2	AS_3	AS_1	AS_2	AS_3	AS_1	AS_2	AS_3
Au(G)	100	1.6 ± 0.9	76	1.8 ± 0.6	1.0	2.9	110	100	89	110	99	82	41
Au(P)	166	1.1 ± 1.6	31	0.5 ± 1.3	1.0	4.9	58	166	152	118	96	86	63
He(G)	49	0.6 ± 1.0	39	0.5 ± 0.9	1.1	2.4	82	49	51	112	52	51	28
He(P)	34	0.9 ± 0.9	31	0.9 ± 0.9	1.0	1.6	50	34	36	28	36	38	26
Ge(G)	109	1.2 ± 1.8	102	0.8 ± 1.3	1.1	2.7	223	109	109	341	102	104	38
Ge(P)	64	1.4 ± 1.3	62	1.1 ± 1.2	1.1	5.2	69	64	65	53	62	59	49
Ad(G)	1787	-	467	-	1.0	5.1	789	1787	529	699	1587	460	115
Ad(P)	1561	-	348	-	1.0	7.2	912	1561	421	731	1419	389	167

dataset. Lee method has a lower update cost, although our method has the lowest, especially when training set size increases.

5 Conclusion and Discussion

We developed an efficient method to speed-up e_{LOO} estimation. Experimental results show that our method outperforms in average previous proposed methods [4,5]. Moreover, speed-up of e_{LOO} evaluation increases with training set size. Our experiments have also highlighted that e_{LOO} deviations, when ϵ stopping criterion is increased efficiently, are mainly due to numerical instabilities when SVM ouptut is not confident. Future works have to extend this method to speed-up bootstrap and k fold cross-validation (with high k value but lower that m).

References

1. Duan, K., Keerthi, S.S., Poo, A.N.: Evaluation of simple performance measures for tuning svm hyperparameters. Neurocomputing **51** (2003) 41–59
2. Lee, J., Lin, C.: Automatic model selection for support vector machines. technical report. http://www.csie.ntu.edu.tw /~cjlin/papers/modelselect.ps.gz (2000)
3. Platt, J.: Fast training of SVMs using sequential minimal optimization, advances in kernel methods-support vector learning. MIT Press (1999) 185–208
4. DeCoste, D., Wagstaff, K.: Alpha seeding for support vector machines. In: Int. Conf. Knowledge Discovery Data Mining. (2000) 345–349
5. Lee, M.M.S., Keerthi, S.S., Ong, C.J., DeCoste, D.: An efficient method for computing leave-one-out error in SVM with gaussian kernels. JAIR **15** (2004) 750–757
6. Vapnik, V.N.: Statistical Learning Theory. Wiley edition (1998)
7. Fan, R.E., Chen, P.H., Lin, C.J.: Working set selection using the second order information for training SVM. JMLR **6** (2005) 1889–1918
8. Chang, C.C., Lin, C.J.: Libsvm: a library for Support Vector Machines. Software Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm (2001)
9. Burbidge, R.: Stopping criteria for SVMs. Available at http://stats.ma.ic.ac.uk/rdb/public/~html/pubs/hermes.pdf (2002)