Stopping on the ν-SVM regularization path using LOO

Gaëlle Loosli, Stéphane Canu

June 30, 2006

Abstract

This paper presents the ν-SVM full regularization path along with a leave-one-out inspired stopping criterion and an efficient implementation. This stopping criterion aims at estimating efficiently when the solution offers a good generalization. To reinforce efficiency, the regularization path is followed from the most parsimonious error-free solution towards the solution with all training points as support vectors (contrarily to what is usually done). The initial solution is also computed efficiently (with the Simple-ν-SVM briefly introduced here) because it does not require to compute the full Gram matrix. Stopping on the path prevents from learning non sparse solutions hence the full Gram matrix is also avoided. Large databases could hence be treated with the proposed method.

1 Introduction

In the context of classification tasks, Support Vector Machines are now very popular. However, their utilization by neophyte users is still hampered by the need to supply values for control parameters in order to get the best attainable results. Mainly, given clean data, SVM's users must make three choices: the type of kernel, its bandwidth and the regularization parameter. It would be convenient to provide users with a *push-button* SVM, that would be able to *auto-set* to the best possible values. This paper presents a new method that approaches this goal. Given the importance of this problem for reaping all the potential benefits of the use of SVM, many research work have been dedicated to ways of helping the setting of the parameters. Most rely on either *outer* measures, such as cross-validation, to guide the selection, or to measures embedded in the learning method itself (for kernels, see [1, 8]). In place of empirical approaches to the setting of the control parameters, regularization paths have been proposed [5, 4, 2] and widely studied these past years since they provide a smart and fast way to access all the optimal solutions of a problem according to all compromises between bias and variance for regression or compromises between bias and regularity in classification. For instance, in the case of classification tasks, as studied in this paper, Soft margins SVM deal with non separable

problem thanks to slack variables that are parametrized by a slack trade-off (usually noted C, it is the regularization parameter). Within the usual formulation of the Soft margins SVM, this trade-off takes its value between 0 (random) and ∞ (hard-margins). The ν-SVM [3] technique reformulates the SVM problem so that C is replaced by a ν parameters taking values in $[0, 1]$. This renormalized parameter has a more intuitive meaning: it represents the minimal proportion of points in the solution and the maximal proportion of misclassified points. The advantages of such an approach are detailed in [3].

However, having the whole regularization path is not enough. Indeed, the end user still needs to retrieve from it the best values for the regularization parameters. Instead of selecting these values by k-fold cross-validation or leave-one-out, or other approximations [10], we propose to include the leave-one-out estimator inside the regularization path in order to have an idea of the generalization error at each step. We explain why it is less expansive than selecting the best parameter *a posteriori* and give a method to stop learning before attaining the end of the path to save useless efforts. This paper presents a low-cost (in term of computational time and memory) method for the auto-setting of the regularization parameter. Contrarily to what is usually done for regularization path, our method does not start with all points as support vectors. Doing so we avoid the computation of the whole Gram matrix at the first step. Then, since the proposed method stops on the path, this extreme non-sparse solution is never attained and thus the whole Gram matrix never required. One of the main advantage of this is that it is possible to use this setting for large databases.

The paper is organized as follows. In the remaining of the introduction section, we first briefly recall the ν-SVM formulation of the problem. Then we introduce the ν-SVM regularization path. Section 3 presents criteria to stop learning on the path when the attained solution is the most parsimonious error-free solution . Finally we present practical results that illustrate our method insights, and show its value.

1.1 The ν-SVM setting

We consider a binary classification problem with training patterns $x_1 \dots x_m \in \mathcal{X}$ and associated classes $y_1 \dots y_m \in \{+1, -1\}$. A ν-SVM classifies a pattern x according to the sign of the decision function $f(.) = \frac{1}{m} \sum_{i=1}^{m} \alpha_i y_i k(x_i, .) + b$. A pattern x_i is called "support vector" when the corresponding coefficient α_i is non zero.

The primal ν-SVM problem is written as follows:

$$\left\{\begin{array}{lcll} & \min\limits_{f,b,\rho,\xi_i} \frac{1}{2}\|f\|^2 - \nu\rho + \frac{1}{m}\sum\limits_{i=1}^{m}\xi_i & & \\ \text{s.t.} & y_i\big(f(x_i)+b\big) \geq \rho - \xi_i & & \forall i \in [1,...,m] \\ \text{and} & \rho \geq 0 & & \\ \text{and} & \xi_i \geq 0 & & \forall i \in [1,...,m] \end{array}\right.$$

1.2 The Simple-ν-SVM solver

The Simple-ν-SVM solver, is derived from the SimpleSVM [9]. It is an active set method, based on the repartition of the training points into three groups that are denoted here as $\mathcal{R}$ for unused points, $\mathcal{E}$ for support vectors placed on the margins and $\mathcal{L}$ for support vectors placed further. The iterative scheme consists in changing the groups such that the costs decreases (by taking the most misclassified points to put it in $\mathcal{E}$ or by taking a point from $\mathcal{E}$ for which α_i is not taking its value between 0 and 1 anymore) and then update $\alpha_{\mathcal{E}}$, b andρ until convergence.

Let reformulate the problem in term of a parameter $\lambda \in [1, m]$ so that the constraints on α do not depend on the number of training points.

$$\left\{\begin{array}{lcll} & \min\limits_{f,b,\rho,\xi_i} \frac{m}{2}\|f\|^2 - \lambda\rho + \sum\limits_{i=1}^{m}\xi_i & & \\ \text{s.t.} & y_i\big(f(x_i)+b\big) \geq \rho - \xi_i & & \forall i \in [1,...,m] \\ \text{and} & \rho \geq 0 & & \\ \text{and} & \xi_i \geq 0 & & \forall i \in [1,...,m] \end{array}\right.$$

With $|\mathcal{L}|$ denoting the cardinal of $\mathcal{L}$, we have $\lambda \geq |\mathcal{L}|$ and $\lambda \leq |\mathcal{L}+\mathcal{E}|$. This is easily seen by writing down the dual formulation of 1.2:

$$\left\{\begin{array}{lcll} & \max\limits_{\boldsymbol{\alpha}} -\frac{1}{2}\boldsymbol{\alpha}^\top G\boldsymbol{\alpha} & & \\ \text{s.t.} & \boldsymbol{\alpha}^\top\mathbf{1} \geq \lambda & & \\ \text{and} & \boldsymbol{\alpha}^\top\mathbf{y} = 0 & & \\ \text{and} & 0 \leq \alpha_i \leq 1 & & \forall i \in [1,...,m] \end{array}\right. \tag{1}$$

where $G(i,j) = \frac{1}{m} y_i y_j k(x_i, x_j)$.

If only the points from $\mathcal{E}$ are considered, we get:

$$\left\{\begin{array}{lcl} & \max\limits_{\boldsymbol{\alpha}_{\mathcal{E}}} -\frac{1}{2}\boldsymbol{\alpha}_{\mathcal{E}}^\top G_{\mathcal{E}}\boldsymbol{\alpha}_{\mathcal{E}} - \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{L}}^\top G_{\mathcal{L}\mathcal{E}}\boldsymbol{\alpha}_{\mathcal{E}} & \\ \text{s.t.} & \boldsymbol{\alpha}_{\mathcal{E}}^\top\mathbf{1}_{\mathcal{E}} \geq \lambda - \boldsymbol{\alpha}_{\mathcal{L}}^\top\mathbf{1}_{\mathcal{L}} & \\ \text{and} & \boldsymbol{\alpha}_{\mathcal{E}}^\top\mathbf{y}_{\mathcal{E}} = -\boldsymbol{\alpha}_{\mathcal{L}}^\top\mathbf{y}_{\mathcal{L}} & \end{array}\right. \tag{2}$$

Note that $\boldsymbol{\alpha}_{\mathcal{L}} = \mathbf{1}_{\mathcal{L}}$. For the sake of simplicity, the dual variables which are known to be equivalent to the primal variables are noted according to their meaning in the primal. The Lagrangian is used to retrieve the following equations:

$$\boldsymbol{\alpha} = G_{\mathcal{E}}^{-1}\big(\rho\mathbf{1}_{\mathcal{E}} + b\mathbf{y}_{\mathcal{E}} - \frac{1}{2}G_{\mathcal{E}\mathcal{L}}\mathbf{1}_{\mathcal{L}}\big)$$

$$\rho = \frac{\lambda - |\mathcal{L}| + \frac{1}{2}\mathbf{1}_{\mathcal{L}}^\top G_{\mathcal{L}\mathcal{E}} G_{\mathcal{E}}^{-1} \mathbf{1}_{\mathcal{E}}}{\mathbf{1}_{\mathcal{E}}^\top G_{\mathcal{E}}^{-1} \mathbf{1}_{\mathcal{E}}}$$

$$b = \frac{-\mathbf{1}_{\mathcal{L}}^\top \mathbf{y}_{\mathcal{L}} - \rho \mathbf{1}_{\mathcal{E}}^\top G_{\mathcal{E}}^{-1} \mathbf{y}_{\mathcal{E}} + \frac{1}{2}\mathbf{1}_{\mathcal{L}}^\top G_{\mathcal{L}\mathcal{E}} G_{\mathcal{E}}^{-1} \mathbf{y}_{\mathcal{E}}}{\mathbf{y}_{\mathcal{E}}^\top G_{\mathcal{E}}^{-1} \mathbf{y}_{\mathcal{E}}}$$

Remark here that ρ is computed ignoring the variable b. This trick does not affect numerical results, but is required for efficient resolution. Indeed, if both constraints from 2 must be satisfied at any moment, the iterative method starting from no support vector cannot start while, at the same time, the first solution should contain at least λ points.

One reason to use this solver instead of the SMO is that SMO is known to be slower when C (in the standard version) increases, which corresponds to small values of λ (in the ν version) while SimpleSVM is faster when C grows. Since our method starts from small λ and aims at stoping before reaching very large values, using the active set method is a natural choice.

2 Regularization path

Our aim is to compute the ν-SVM solution for all values of ν. As shown in [5], the path is piecewise linear. It means that the support vectors set does not change between two values of ν. Hence we only need to identify when a change occurs in the sets. Similarly to what is done in active set methods solving the SVM [9, 6], the first steps consists in identifying the best direction to follow and the second step is to determine how far to follow it.

Let note $g(x_i) = y_i\big(f(x_i) + b\big) - \rho$. Then we have:

$$\begin{cases} \mathcal{L}: & g(x_i) < 0 & \forall i \in \mathcal{L} & \alpha_i = 1 & \text{bounded points} \\ \mathcal{E}: & g(x_i) = 0 & \forall i \in \mathcal{E} & 0 < \alpha_i < 1 & \text{margins points} \\ \mathcal{R}: & g(x_i) > 0 & \forall i \in \mathcal{R} & \alpha_i = 0 & \text{useless points} \end{cases}$$

The idea of the path is to provide an iterative method that follows the path by pieces, stopping at each change among the groups. Each point of the path reflects the optimal solution of the ν-SVM according to a particular value of λ. We begin with the smallest value of λ and let it grow up to its larger value. Since the provided solutions are equivalent as long as the groups do not change, we only need to identify for which values of λ a point is going to move from one group to another. We identify four possible movements: from $\mathcal{L}$ to $\mathcal{E}$ (from the wrong side of the margin to the margin), from $\mathcal{R}$ to $\mathcal{E}$ (the point becomes support vector), from $\mathcal{E}$ to $\mathcal{L}$ (a support vector becomes bounded) and from $\mathcal{E}$ to $\mathcal{R}$ (a support vector is no longer one). We will denote those steps respectively as $in(\ell)$, $in(r)$, $out(\ell)$ and $out(r)$.

Movements $in(\ell)$ and $in(r)$ happen when $\exists i \in \mathcal{L}$ or $i \in \mathcal{R}$ such that $g(x_i) = 0$. Movements $out(\ell)$ and $out(r)$ occur respectively when $\exists i \in \mathcal{E}$ such that $\alpha_i = 1$ or $\alpha_i = 0$.

Hence we need to express $g(x)$ and $\boldsymbol{\alpha}$ depending on steps t and $t+1$. For each point we get:

$$\begin{aligned} g^{t+1}(x_i) = & \quad g^{t+1}(x_i) - g^t(x_i) + g^t(x_i) \\ = & \quad G(i,:)\boldsymbol{\alpha}^{t+1} + b^{t+1}y_i - \rho^{t+1} - G(x_i,:)\boldsymbol{\alpha}^t - b^t y_i + \rho^t + g^t(x_i) \\ = & \quad G(i,:)\boldsymbol{\delta}_{\boldsymbol{\alpha}} + \delta_b y_i - \delta_\rho + g^t(x_i) \end{aligned} \tag{3}$$

with $\boldsymbol{\delta}_{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{t+1} - \boldsymbol{\alpha}^t$, $\delta_b = b^{t+1} - b^t$ and $\delta_\rho = \rho^{t+1} - \rho^t$. $G(i,:)$ designates the i^{th} row the the matrix. From equation 3 and depending on the concerned event, it is possible to find which value of λ to choose next. We summarize in table 1 the events and the algorithm mechanic.

Step	$in(r)$	$out(r)$	$out(\ell)$	$in(\ell)$
t	$i \in \mathcal{R}$ $g^t(x_i) > 0$ $\alpha_i = 0$	$i \in \mathcal{E}$ $\star\mathbf{g^t(x_i) = 0}$ $0 < \alpha_i < 1$	$i \in \mathcal{E}$ $\star\mathbf{g^t(x_i) = 0}$ $0 < \alpha_i < 1$	$i \in \mathcal{L}$ $g^t(x_i) < 0$ $\alpha_i = 1$
$t+1$	$i \in \mathcal{E}$ $\star\mathbf{g^{t+1}(x_i) = 0}$ $0 < \alpha_i < 1$	$i \in \mathcal{R}$ $\star\mathbf{g^{t+1}(x_i) \geq 0}$ $\star\mathbf{\alpha_i = 0}$	$i \in \mathcal{L}$ $\star\mathbf{g^{t+1}(x_i) \leq 0}$ $\star\mathbf{\alpha_i = 1}$	$i \in \mathcal{E}$ $\star\mathbf{g^{t+1}(x_i) = 0}$ $0 < \alpha_i < 1$

Table 1: Summary of the events. each column stands for a particular event. In blue starred are noted the properties that are used to compute the corresponding λ^{t+1}

2.1 Points in $\mathcal{E}$ and detection of $out(\ell)$ and $out(r)$

Events $out(\ell)$ and $out(r)$ are detected using their values of α. Indeed, one condition to remain in $\mathcal{E}$ is to keep $0 < \alpha < 1$. Retrieving λ^{t+1} for which one of these conditions is violated for each point requires to write α_i depending on λ^{t+1}.

Remark that in $\mathcal{E}$, $g^t(x) = 0$ and $g^{t+1}(x) = 0$. Equation 3 together with the constraints from 1 ($\boldsymbol{\alpha}^\top \mathbf{1} \geq \lambda$, hence $\boldsymbol{\delta}_{\boldsymbol{\alpha}}^\top \mathbf{1} \geq \lambda^{t+1} - \lambda^t$ and $\boldsymbol{\alpha}^\top \mathbf{y} = 0$, hence $\boldsymbol{\delta}_{\boldsymbol{\alpha}}^\top \mathbf{y} = 0$) leads to a system of linear equations $A\boldsymbol{\delta} = (\lambda^{t+1} - \lambda)\mathbf{c}$, where

$$A = \begin{bmatrix} G & \mathbf{y} & -\mathbf{1} \\ \mathbf{y}^\top & 0 & 0 \\ \mathbf{1}^\top & 0 & 0 \end{bmatrix} \quad \delta = \begin{bmatrix} \boldsymbol{\delta}_\alpha \\ \delta_b \\ \delta_\rho \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} \mathbf{0} \\ 0 \\ 1 \end{bmatrix}$$

This leads to $\boldsymbol{\delta} = (\lambda^{t+1} - \lambda^t)A^{-1}\mathbf{c}$. So for points in the set $\mathcal{E}$ there is:

$$\begin{cases} \boldsymbol{\alpha}^{t+1} = \boldsymbol{\alpha}^t + (\lambda^t - \lambda^{t+1})\boldsymbol{\eta}_{\boldsymbol{\alpha}} \\ b^{t+1} = b^t + (\lambda^t - \lambda^{t+1})\eta_b \\ \rho^{t+1} = \rho^t + (\lambda^t - \lambda^{t+1})\eta_\rho \end{cases}$$

where $\boldsymbol{\eta}$ denote the vector $A^{-1}\mathbf{c}$.

As mentioned earlier, a change in the groups will occur as soon as one of the α_i will meet on of the boundaries, *i.e.* when $\alpha_i = 0$ or $\alpha_i = 1$ for $i \in \mathcal{E}$. This gives two change conditions:

$$\begin{cases} \lambda^{t+1}_{out(r)} = \dfrac{-\alpha^t_i}{\eta_i} + \lambda^t \\ \lambda^{t+1}_{out(\ell)} = \dfrac{1-\alpha^t_i}{\eta_i} + \lambda^t \end{cases}$$

2.2 Points in $\mathcal{L}$ and $\mathcal{R}$ and detection of $in(\ell)$ and $in(r)$

Events $in(\ell)$ and $in(r)$ are detected using their values of $g(x_i)$. Indeed, one condition to remain in $\mathcal{R}$ is to keep $g(x_i) > 0$ and $g(x_i) < 0$ to stay in $\mathcal{L}$. Retrieving λ^{t+1} for which one of these conditions is violated for each point requires to write $g^{t+1}(x_i)$ depending on λ^{t+1}. For a point moving inside $\mathcal{E}$, the particularity is that $g^{t+1}(x_i)$ becomes nul.

Defining $h(x) = G(i,:)\boldsymbol{\eta_\alpha} + \eta_b - \eta_\rho$ leads to

$$\begin{aligned} g^{t+1}(x_i) = &\quad G(i,:)\boldsymbol{\delta_\alpha} + \delta_b y_i - \delta_\rho + g^t(x_i) \\ = &\quad (\lambda^{t+1} - \lambda^t)\big(G(i,:)\boldsymbol{\eta_\alpha} + \eta_b - \eta_\rho\big) + g^t(x_i) \\ = &\quad (\lambda^{t+1} - \lambda^t)h^t(x_i) + g^t(x_i) = 0 \end{aligned}$$

and thus

$$\begin{aligned} \lambda^{t+1}_{in(\ell)} = &\quad \frac{-g^t(x_i)}{h^t(x_i)} + \lambda^t \quad i \in \mathcal{L} \\ \lambda^{t+1}_{in(r)} = &\quad \frac{-g^t(x_i)}{h^t(x_i)} + \lambda^t \quad i \in \mathcal{R} \end{aligned}$$

Note that it may happen that several points reach $\mathcal{E}$ at the same time. Even though this does not change equations, it is a relevant remark for implementation. Indeed, if a point is missed, the path is left and the missed point will not be selected afterwards.

2.3 Regularization path algorithm

At each step we look for the smallest $\lambda^{t+1} > \lambda^t$ among $\big\{\lambda^{t+1}_{in(\ell)}, \lambda^{t+1}_{in(r)}, \lambda^{t+1}_{out(\ell)}, \lambda^{t+1}_{out(r)}\big\}$. We update the $\boldsymbol{\alpha}^{t+1}$ according to the chosen λ^{t+1} and then the groups. The process is stopped when $\lambda = m$.

Initialisation The initialisation is done for $\lambda = 1$, which means that no training error is tolerated, via the Simple-ν-SVM. The first solution is thus the most parsimonious error-free solution one on the path (and potentially one of the most over-fitted). If the problem is truly separable, it will also be the best one. This initialization can fail, even using Gaussian kernel, as is it underlined in [5], when the true rank of the kernel matrix is less than m. When this happens, it is easily detected (the first solution does not have a zero training error) and can be fixed by narrowing the kernel bandwidth or initializing with a larger λ.

3 Stopping on the path using Leave One Out

Now we have the entire path and we know that we begin with the most over-learned solution (zero-error) and end with all the points as support vectors. At each step we get a more regular solution and we would like to know when this solution is optimal in term of generalization. The idea here is to compute an estimate of the generalization error at each step and to study if we can find a stopping criteria along the path. To do so the idea is to compute together with the regularization path a sequence of estimates of the generalization error to stop when this sequence reaches a minimum.

Among all possible estimations of the generalization error, the leave-one-out seems to be the one advocated by practitioners. The major drawback of the leave-one-out estimate is the time required to compute it. Solutions have been proposed to overcome this deficiency. [10] propose to use an approximation easily available called the GCV (Generalized Cross Validation). Others [7] propose to take advantage of efficient implementation of the SVM with warm-start (starting from the current solution as an *a priori* on the next solution) to derive acceptable procedures. Following this idea, we show next how to integrate those estimators in the algorithm and we point out that this method is much cheaper than an external LOO method.

The leave-one-out error is defined as the mean error done for the removed points. We also compute a second leave-one-out estimation to have an idea of the variance of the solution:

$$LOO1_{error} = \frac{1}{n}\sum_i 1 - sign(\hat{y}_i y_i) \qquad LOO2_{error} = \frac{1}{n}\sum_i max(0, \rho - \hat{y}_i y_i)$$

This second formula is very helpful to detect over-fitting. Indeed, outliers will be very penalized.

The leave-one-out error rates are estimated at each step. Since no point from $\mathcal{R}$ participate to the solution, they would necessaraly have a zero error if once removed from the training set. Hence we only need to compute the LOO errors of each point ℓ of $\mathcal{E}$ and $\mathcal{L}$. The solution $S_{-\ell}$ is close to the current solution S given along the path. Thus we obtain $S_{-\ell}$ from S with Simple-ν-SVM warm start. The cost of the computation of the LOO at each step is $\big(|\mathcal{E}|+|\mathcal{L}|\big)$ update steps.

Since Simple-ν-SVM is also an active set method, if it is given an almost true repartition, then it converges quickly to the solution.

If the generalization error if significantly better for some range of parameters λ, we expect to see it through the LOO error rates. Hence we monitor this value to detect when it starts to increase significantly. Then we can stop learning a go back to S_λ which has given the lowest LOO errors. Doing so, we avoid to compute the part of the path for which most of the points are bounded support vectors. Indeed those solution contradict the SVM goal: sparseness. Algorithm 3 describes the whole procedure.

Algorithm for ν-SVM regularization path with stopping criteria

1) initialize with Simple-ν-SVM with $\lambda = 1$
2) while $\lambda < m$ or smoothed $LOO2_{error}$ decreases
   2.1) find the smallest $\lambda^{t+1} > \lambda^t$ among $\{\lambda^{t+1}_{in(\ell)}, \lambda^{t+1}_{in(r)}, \lambda^{t+1}_{out(\ell)}, \lambda^{t+1}_{out(r)}\}$
   2.2) update the $\boldsymbol{\alpha}$, b, ρ and the groups $\mathcal{R}$, $\mathcal{E}$, $\mathcal{L}$
   2.3) for each $i \in \mathcal{E}$ and $\mathcal{L}$
      2.3.1) adapt the current solution without point i
      2.3.2) $LOO1_i \leftarrow$ test on point i
      2.3.3) $LOO2_i \leftarrow$ test on point i
   end for each
   2.4) $LOO1_{error}(\lambda) = mean(LOO1_i)$
   2.5) $LOO2_{error}(\lambda) = mean(LOO2_i)$
end while
3) return $\lambda \leftarrow \lambda(min(LOO1_{error}))$ and corresponding solution

4 Experimental results

For the evaluation of our method, we have used artificial datasets. The first one is the *apple and banana* dataset (figure 1 on the left). The second one is the *mixture* dataset[1] which is a noisy dataset (with a Bayes error of 21%), used in [5] to illustrate the behaviour of the SVM regularization path.

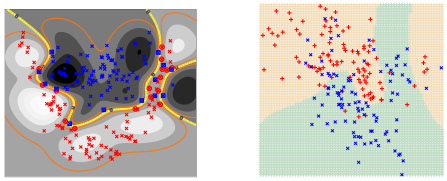

Figure 1: Datasets. Left: the apple and banana. Right: mixture

4.1 Illustration of the paths

The regularization paths can be represented via the values taken by the α coefficients during learning. Those coefficients follow piecewise linear paths. Figure 2 illustrates this behavior for the ν-SVM regularization path. In order to clirify the figure, $\alpha_i y_i$ is plot, such that all negative points appear bellow zeros.

4.2 Efficiency of the ν-SVM regularization path

We are able to reproduce the C-SVM regularization path for the mixture dataset, in terms of accuracy, number of change points and speed. The main difference

[1] available at `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`

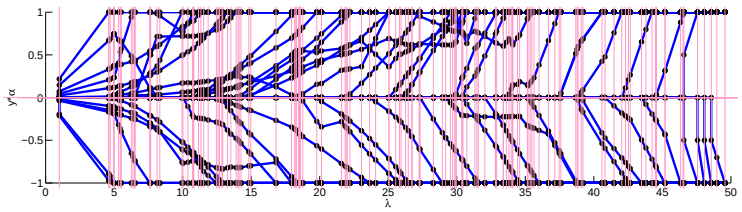

Figure 2: Evoluation of the $y_i\alpha_i$ along the path for the apple and banana problem

is that we follow the path in the other direction. Following the path in the other direction mainly presents practical advantages. Indeed, if the whole path is computed, there is no major difference, except that the first solution for our method requires to solve a QP. However, when we introduce the stopping criteria and the LOO estimation at each step, there is a huge difference. If we start from all points in $\mathcal{L}$, it means that all points are support vectors and we need to update the current solution for every point to have the LOO estimation. On the contrary, if we start from the other side, we begin with a sparse solution. Moreover, if we stop before the end, we never attain very large solutions.

4.3 On the stopping criteria

We have conducted experiments on artificial datasets in order to illustrate how the LOO estimation can be a criteria to stop learning on the path before attaining non sparse solutions. Figures 3and 4 respectively give examples of results on the apple and banana dataset and on the mixture dataset, both with an *rbf* kernel. Each LOO estimate provide usefull information. The first one (based on the counting of the errors) gives a good approximation of the generalisation error. The second one, based on the output value of the SVM represents the variance of the solution and we look for a low variance solution.

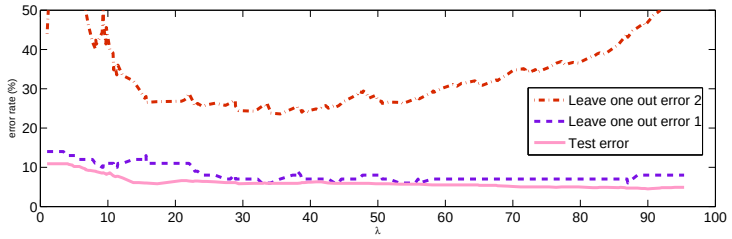

Figure 3: Illustration on the apple en banana dataset of the LOO error rate evolution according to λ, reported with the test error

From a practical point of view, determining the correct moment to stop requires some heuristic and using a smoothed curved of the LOO error is usefull.

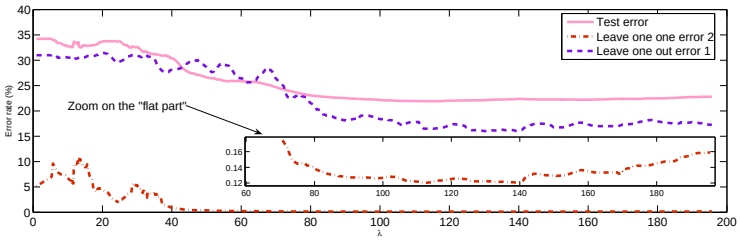

Figure 4: Illustration on the mixture dataset of the LOO error rate evolution according to λ, reported with the test error

Our heuristic consists in choosing to stop when the smoothed LOO2 error has not been decreasing for a period lasting as long as it take for λ to grow of 10. Then we choose backward the λ corresponding to the minimum achieved by smoothed LOO1.

4.4 Implementation details

Stability The regularization path method is very sensitive to the initialisation quality and to the data. It may happen that the solution leaves the path because of numerical approximation. Ones the path is left, the algorithm shows a very instable behaviour. To address this problem we have implemented a monitoring system of the solution. Each time $\exists i \in \mathcal{E} \text{s.t.} \alpha_i > 1$ or $\alpha_i < 0$, we run an external QP to correct the problem. This is done cheaply using the Simple-ν-SVM warm start. This trick is also applied when set $\mathcal{E}$ becomes empty (see [5] for details on this problem) or of size one (then matrix A is singular) or when all point in $\mathcal{E}$ belong to the same class (A is singular as well). This last situation can be avoided in practice if the bias b is incorporated in the ξ_i (which are then allowed to be negative). This trick removes the constraint $\boldsymbol{\alpha}^\top \mathbf{y} = 0$ in equation 1 and the matrix A will remain invertible.

Initialization For the initialisation, if the first point is not feasible (which is easily seen if the training error is not 0), we increase the initial value of λ (by steps of 1) until we find a first feasible solution (λ is an upper bound on the number of training errors).

5 Conclusion

We have presented in this paper the Simple-ν-SVM solver as well as a low cost leave-one-out stopping criteria in the ν-SVM regularization path. The path is followed in the *good* direction in terms of memory requirements and computational time. Obviously this statement is true only if we can stop before reaching the other side of the path but after attaining a good generalization error. By evaluating two LOO estimators at each step of the path, we save

the time needed to compute a part of the path since we can stop on the way. We also propose a good value for the regularization parameter λ. The LOO step is done cheaply since only a part of the points require an update of the solution. Moreover we can approximate the LOO estimation in order to reduce the computational time. We are now very close to the *push-button* SVM and our future research will include the bandwidth selection as well.

Acknowledgments

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

[1] Andreas Argyriou, Raphael Hauser, Charles A. Micchelli, and Massimiliano Ponti. A dc-programming algorithm for kernel selection. *ICML*, 2006.

[2] Francis Bach, David Heckerman, and Eric Horvitz. On the path to an ideal ROC curve: Considering cost asymmetry in learning classifiers. In Robert G. Cowell and Zoubin Ghahramani, editors, *AISTATS*, pages 9–16. Society for Artificial Intelligence and Statistics, 2005.

[3] P. H. Chen, C. J. Lin, and B. Schölkopf. A tutorial on v-support vector machines. *Applied Stochastic Models in Business and Industry*, 21(2):111–136, 2005.

[4] Lacey Gunter and Ji Zhu. Computing the solution path for the regularized support vector regression. In *NIPS*, 2005.

[5] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

[6] Linda Kaufman. Solving the quadratic programming problem arising in support vector classification. *Advances in kernel methods: support vector learning*, pages 147–167, 1999.

[7] J.-H. Lee and C.-J. Lin. Automatic model selection for support vector machines. Technical report, Dept. of Computer Science and Information Engineering, National Taiwan University, 2000.

[8] Charles A. Micchelli and Massimiliano Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

[9] S. V. N Vishwanathan, Alexander J. Smola, and M. Narasimha Murty. SimpleSVM. *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[10] Grace Wahba. *Support Vector Machines, Reproducing Kernel Hilbert spaces and the randomized GACV*, pages 69–88. MIT Press, B. Scholkopf and C. Burges and A. Smola edition, 1999.