

# Auto-setting of the regularization hyper-parameter in SVM

Gaëlle Loosli and Stéphane Canu  
gaelle.loosli@insa-rouen.fr

Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes.

October 9, 2006





# Introduction

## Goal: auto-setting of the regularization parameter for SVM

- Simple- $\nu$ -SVM solver
- Regularization path for SVM
- Leave-one-out validation method

### What is $\nu$ -SVM?

- $\nu$ -SVM is a soft-margin SVM
- the regularization hyper-parameter is  $\nu$
- it is an upper bound on the proportion of saturated support vectors
- it is a lower bound on the proportion of total support vectors

For clarity in equations, we use  $\lambda = n\nu$  :  $\lambda$  directly represents the number of support vectors instead of the proportion.

## ν-SVM

We consider a binary classification problem with training patterns  $x_1 \dots x_m \in \mathcal{X}$  and classes  $y_1 \dots y_m \in \{+1, -1\}$ .

## Primal

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

## Dual

$$\left\{ \begin{array}{ll} \max_{\alpha} & -\frac{1}{2} \alpha^\top G \alpha \\ \text{s.t.} & \alpha^\top \mathbf{1} \geq \lambda (\Leftrightarrow \alpha^\top \mathbf{1} = \lambda) \\ \text{and} & \alpha^\top \mathbf{y} = 0 \\ \text{and} & 0 \leq \alpha_i \leq 1 \quad \forall i \in [1, \dots, m] \end{array} \right.$$

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

$\nu$ -SVM

We consider a binary classification problem with training patterns  $x_1 \dots x_m \in \mathcal{X}$  and classes  $y_1 \dots y_m \in \{+1, -1\}$ .

## Primal

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

## Dual

$$\left\{ \begin{array}{ll} \max_{\alpha} & -\frac{1}{2} \alpha^\top G \alpha \\ \text{s.t.} & \alpha^\top \mathbf{1} \geq \lambda (\Leftrightarrow \alpha^\top \mathbf{1} = \lambda) \\ \text{and} & \alpha^\top \mathbf{y} = 0 \\ \text{and} & 0 \leq \alpha_i \leq 1 \quad \forall i \in [1, \dots, m] \end{array} \right.$$

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

# Simple- $\nu$ -SVM solver

The SimpleSVM algorithm is an iterative method that starts from an empty solution. It works on three groups of points.  $\mathcal{E}$  for the margin support vectors,  $\mathcal{L}$  for the bounded support vectors and  $\mathcal{R}$  for the non support vectors.

## active set method. iterates over:

1. solve the dual without the box constraint on  $\mathcal{E}$
2. check the box constraints for  $\mathcal{E}$  : update the groups if constraints are violated by a point of  $\mathcal{E}$ . Go to 1.
3. check the quality of the solution : update the groups if a point from  $\mathcal{L}$  or  $\mathcal{R}$  violates constraints

until all constraints satisfied

## problems to apply this to $\nu$ -SVM

- there's an extra constraint on the number of support vectors
- the only way is to start with a random selection of support vectors to meet this constraint for the first loop
- a heuristic consists in starting with  $\lambda = 1$  and let it grow with the number of support vectors until it reaches its given value

# What is it?

## Definition

It is the ensemble of all possible solutions for a given dataset when an hyper-parameter varies continuously inside its range.

## More practically for SVMs

- all solutions sharing the same groups  $\mathcal{R}, \mathcal{E}$  and  $\mathcal{L}$  are equivalent for classification task
- interesting values of the hyper-parameter are the ones for which groups change
- hence the path is composed of a finite number of interesting solutions
- the path is piecewise linear between group changes

Regularization path for  $\nu$ -SVM

We only need to identify when a change occurs in the groups.

## Groups properties

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

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

## Defining steps

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

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

with  $\delta \alpha = \alpha^{t+1} - \alpha^t$ ,  $\delta_b = b^{t+1} - b^t$  and  $\delta_\rho = \rho^{t+1} - \rho^t$ .

$G(i, :)$  designates the  $i^{th}$  row the the matrix.

Depending on the concerned event, it is possible to find which value of  $\lambda$  to choose next.

### Linear system to solve

$\delta$  is the solution of a linear system.

Points in  $\mathcal{E}$ 

Remark that in  $\mathcal{E}$ ,  $g^t(x) = 0$  and  $g^{t+1}(x) = 0$ .  $A\delta = (\lambda^{t+1} - \lambda)\mathbf{c}$ , where

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

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

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

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

As mentioned earlier, a change in the groups will occur as soon as one of the  $\alpha_i$  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:

$$\begin{cases} \lambda_{out(r)}^{t+1} = \frac{-\alpha_i^t}{\eta_i} + \lambda^t \\ \lambda_{out(\ell)}^{t+1} = \frac{1 - \alpha_i^t}{\eta_i} + \lambda^t \end{cases}$$

Points in  $\mathcal{L}$  and  $\mathcal{R}$ 

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

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

and thus

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

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.

# Summary

We identified four possible movements:

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

## Algorithm

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

## $\nu$ -SVM solver

### Algorithm

```

1: initialisation,  $\lambda = 1$ 
2: while  $\lambda < m$  do
3:   solve linear system ▷  $\alpha_w$ 
4:   if non admissible solution then
5:     project solution in the admissible set ▷ Simple- $\nu$ -SVM step
6:   else if non optimal solution then
7:     select next point ▷ Simple- $\nu$ -SVM step
8:   else
9:     compute the next  $\lambda$  ▷ regularization path step
10:  end if
11: end while

```

### Side effect : a new efficient $\nu$ -SVM solver

By changing the WHILE condition by  $\lambda \leq \lambda^*$ , we can reach efficiently a specific solution.

# Stopping on the path

## Why should we stop?

- The last solution takes all points as support vectors : can't be a good SVM solution
- The last solution requires to compute the complete Gram matrix : can't be used with large datasets
- For  $\nu$  auto-setting, we want the "sparsest good" solution (realistic application can't afford non sparse solution but may tolerate compromise on the quality of the solution - see early stopping strategies, non optimal solvers...)

## How can we stop?

- evaluate the evolution of the generalization error estimation at each step
- evaluate the over-fitting at each step (the first proposed solution on the path is hard-margin)
- stop when the solution is regular and the generalization error begins to increase

# Leave-one-out evaluation

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:

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

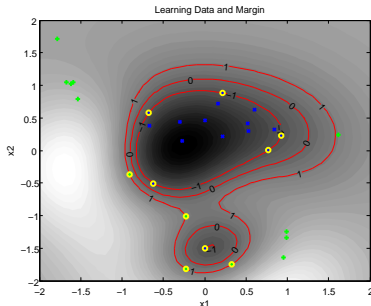
This second formula is very helpful to detect over-fitting. Indeed, outliers will be very penalized.

# Leave-one-out evaluation

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:

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

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

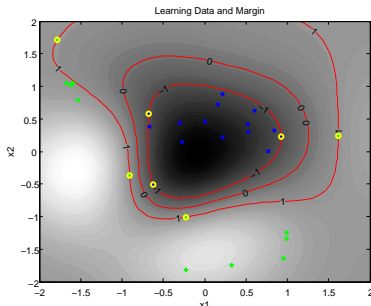


# Leave-one-out evaluation

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:

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

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



# Leave-one-out evaluation

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:

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

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

## How to do it practically

- leave-one-out error rates are estimated at each step,
- no point from  $\mathcal{R}$  participate to the solution  $\rightarrow$  zero error, no computational effort,
- need to compute the LOO errors of each point of  $\mathcal{E}$  and  $\mathcal{L}$  only,
- use of Simple- $\nu$ -SVM warm start (we already know a close solution of the groups)

For sparse solution, it is very efficient. In any case, it is more efficient than external LOO evaluation.

# Stopping criteria

## When to stop

Based on LOO2, we can estimate if the current solution is still over-fitting (stop when LOO2 doesn't vary anymore).

## Estimate the generalization error

LOO1 directly gives the leave-one-out estimation of the generalization error.

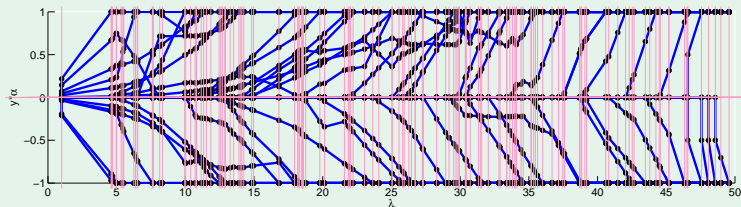
## Why not external LOO?

- Overall, we compute less solutions of SVM (we make the economy of every useless point estimation)
- Thanks to LOO2, we know when it is no longer useful to follow the path (the solution is stable)

# Experiments

For the evaluation of our method, we have used the artificial *apple and banana* dataset.

## Path



Evaluation of the  $y_i \alpha_i$  along the path for the apple and banana problem. The regularization paths can be represented via the values taken by the  $\alpha$  coefficients during learning. Those coefficients follow piecewise linear paths.  $\alpha_i y_i$  is plot, such that all negative points appear below zeros.

## Experiments

## LOO

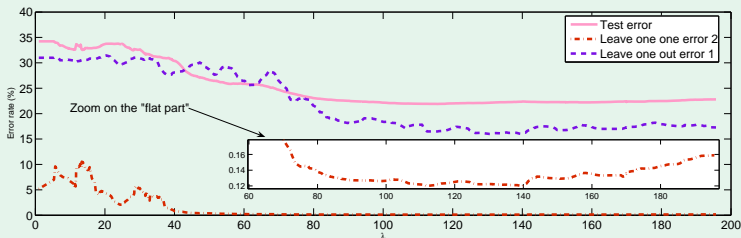


Illustration on the apple and banana dataset of the LOO error rate evolution according to  $\lambda$ , reported with the test error.

## Results

- we can compute efficiently the  $\nu$ -SVM path (just a derivation of existing method)
- we follow the path from sparse to plain solution, contrarily to what is usually done
- we obtained as a side effect a very efficient  $\nu$ -SVM solver
- we can stop on the path automatically and obtain an auto-setting of the regularization term

## Perspectives

- we think it is possible to derive an online or at least an incremental method for regularization paths
- we are in the process of testing the method on large databases
- we plan to extend this to other hyper-parameters (and obtain a push-button SVM?)

# Thank you

Questions?