Piecewise Linearity and Solution Path

Jie Tang
Knowledge Engineering Group
Department of Computer Science and Technology
Tsinghua University
2012

Support Vector Machines

$$\min_{w,b,\xi} \frac{\|w\|}{2} + C\sum_{i=1}^{N}\xi_i$$

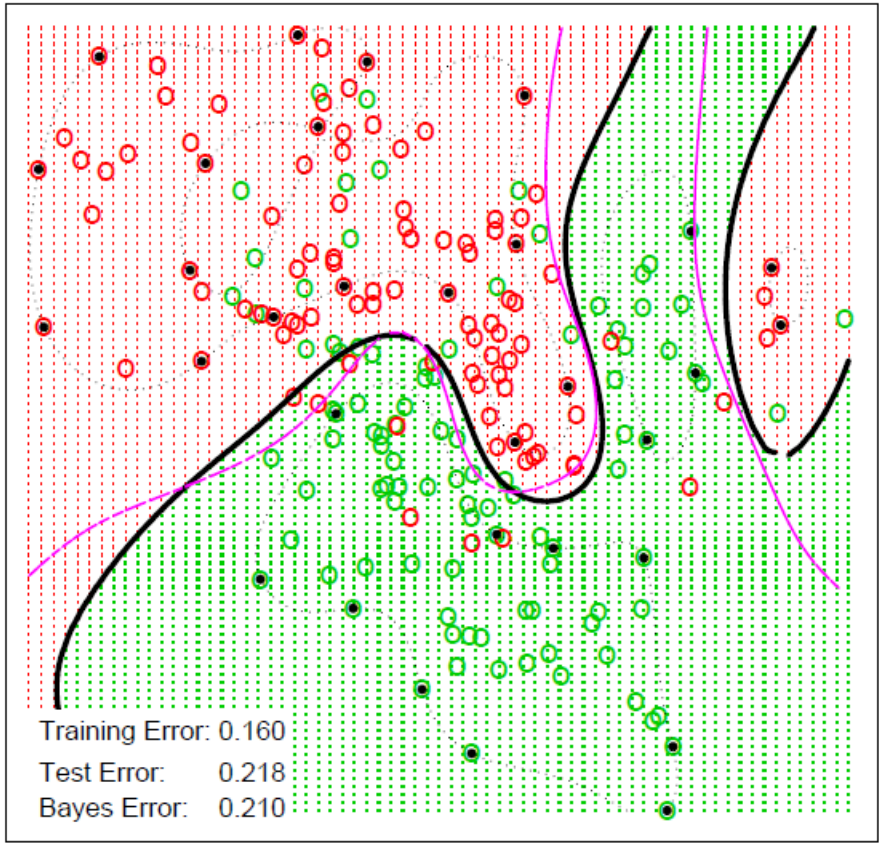
$$s.t. \qquad y(w^T x^{(i)} + b) \geq 1 - \xi_i, 1 \leq i \leq N$$

$$\xi_i \geq 0, 1 \leq i \leq N$$

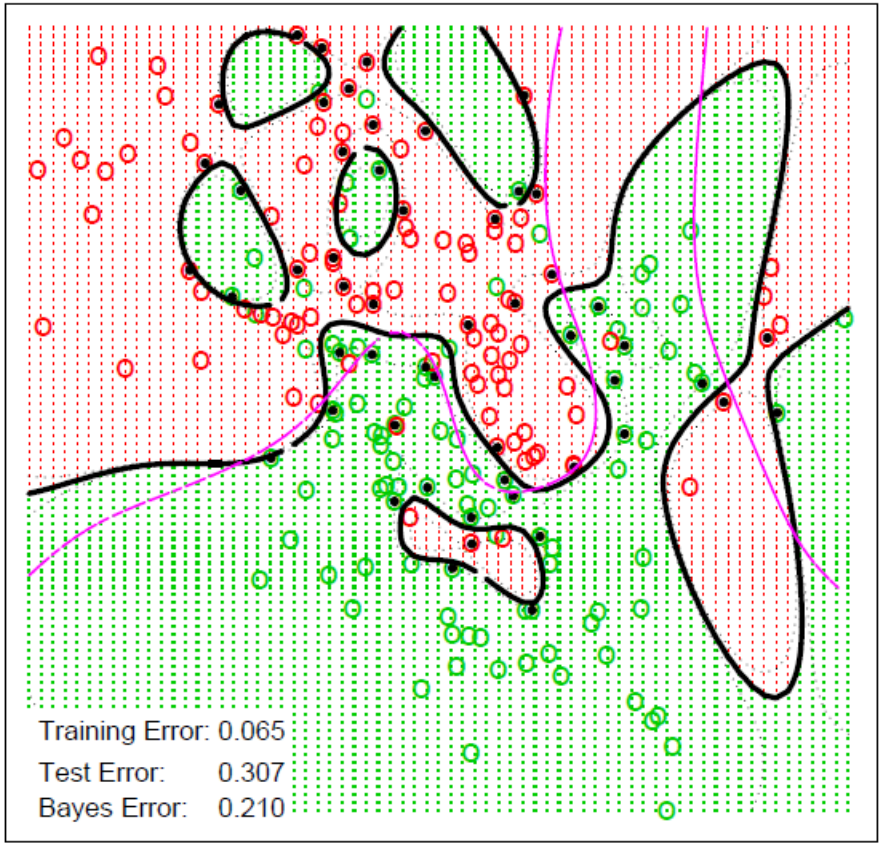
Usually, we find parameters *w* by optimization algorithm and set the parameter *C* manually.

Effect of the cost parameter

Radial Kernel: $C = 2, \gamma = 1$



Radial Kernel: $C = 10,000, \gamma = 1$



Recall: an alternative formulation

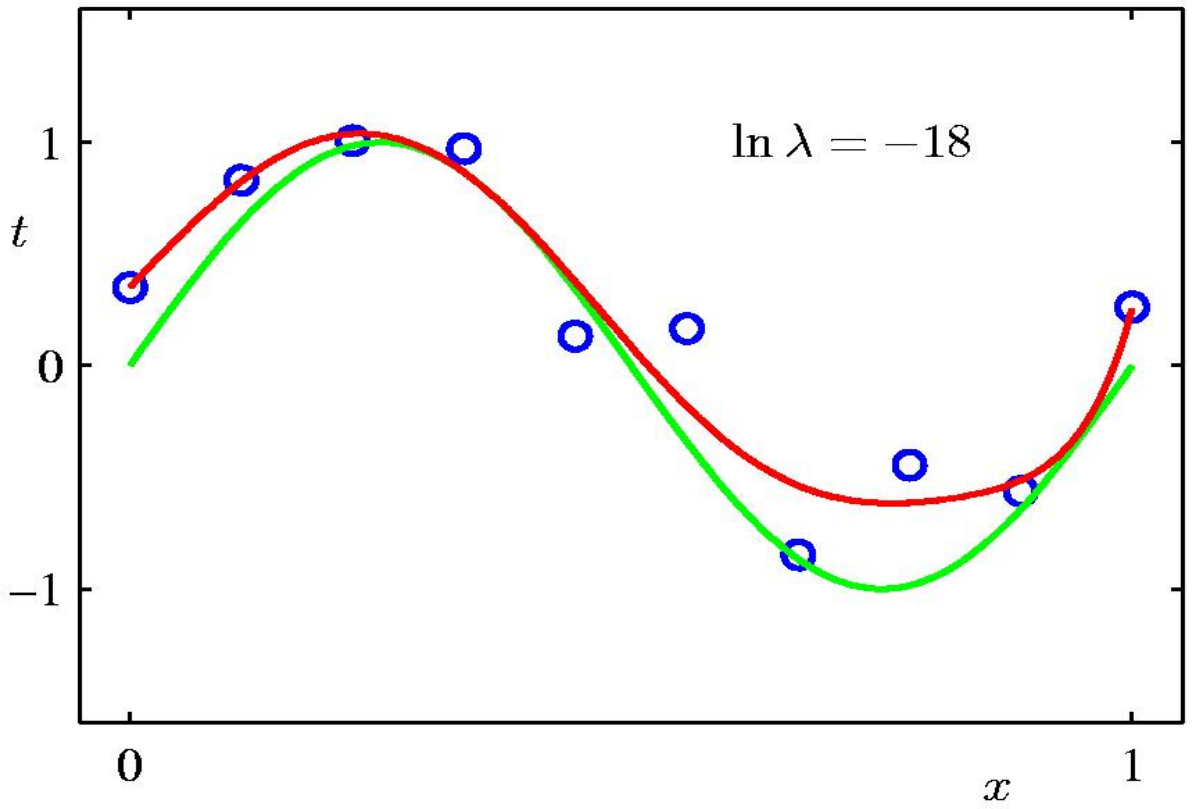
—Effect of the cost parameter

Penalize large coefficient values

$$L(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(f(x_i,\mathbf{w}) - y_i)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

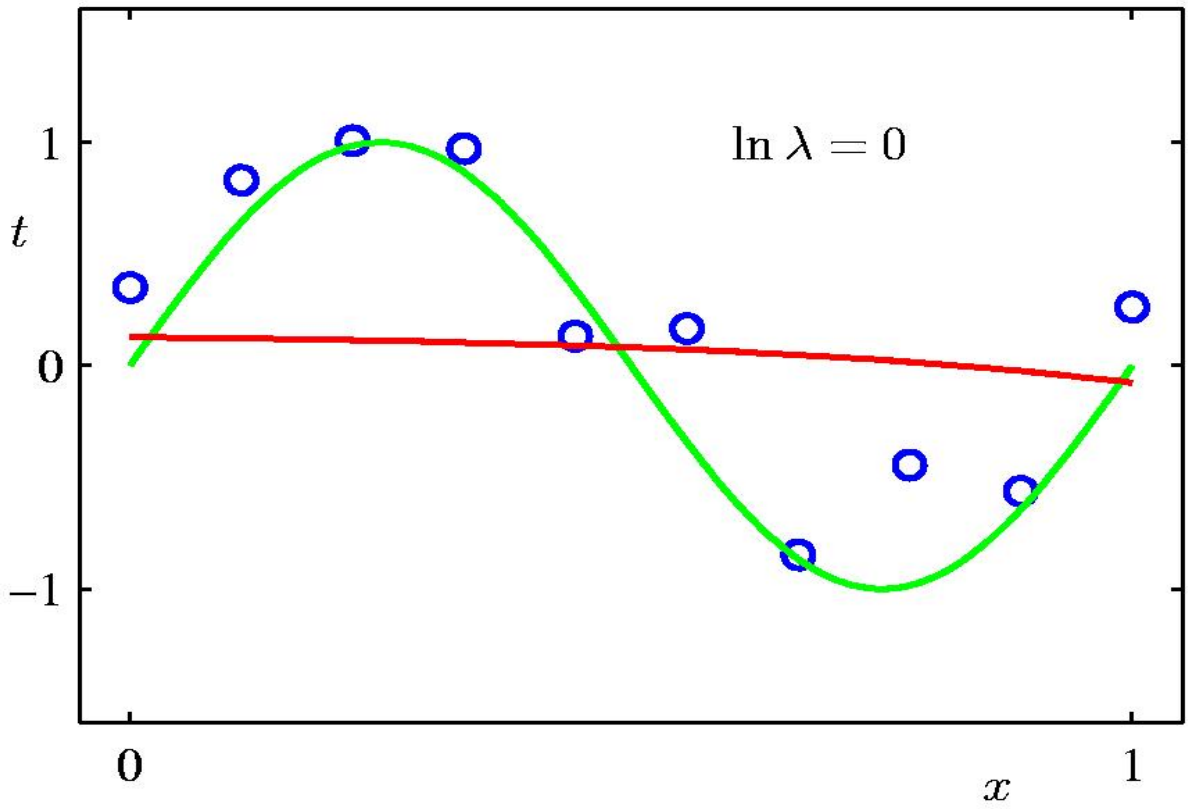
Regularization

$$\ln \lambda = -18$$



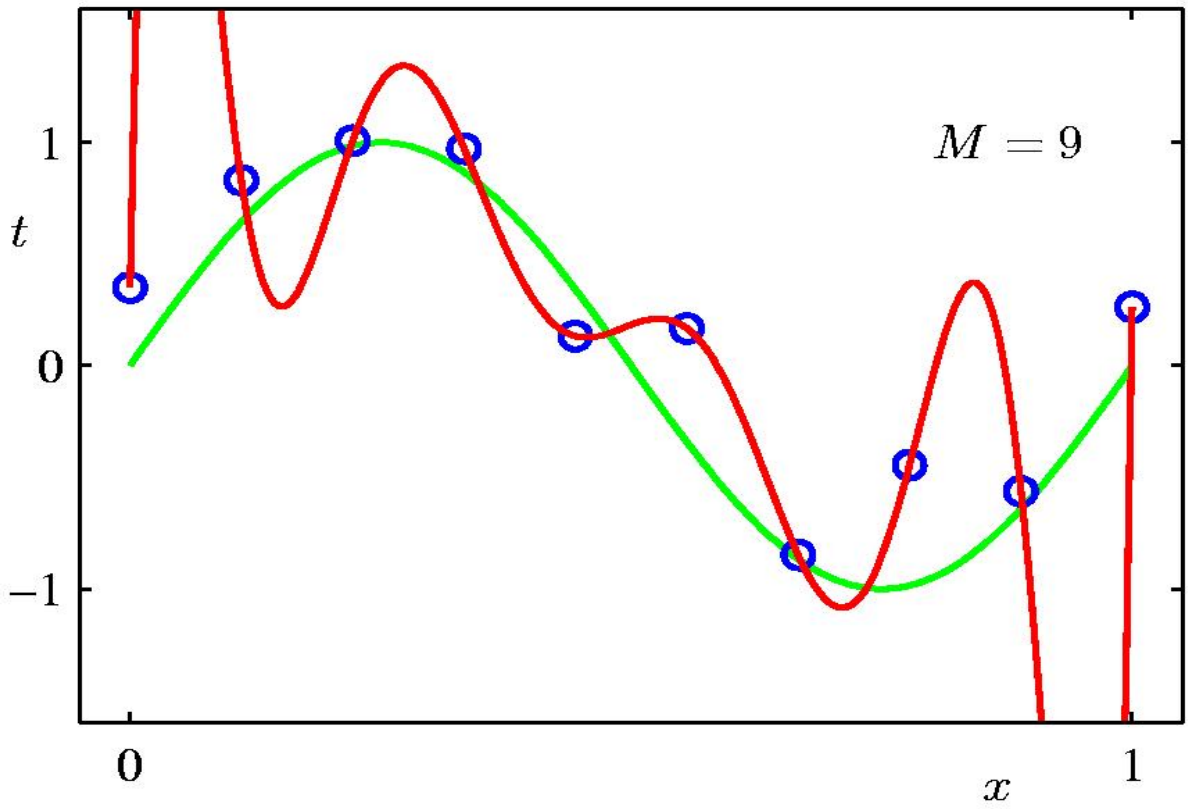
Regularization

$$\ln \lambda = 0$$



9th Order Polynomial

$\lambda = 0$

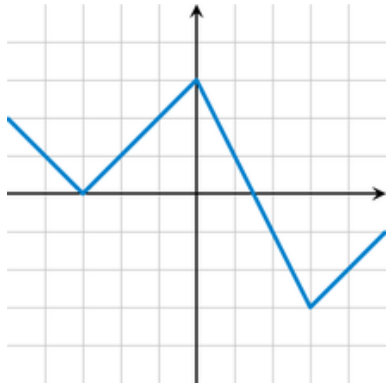


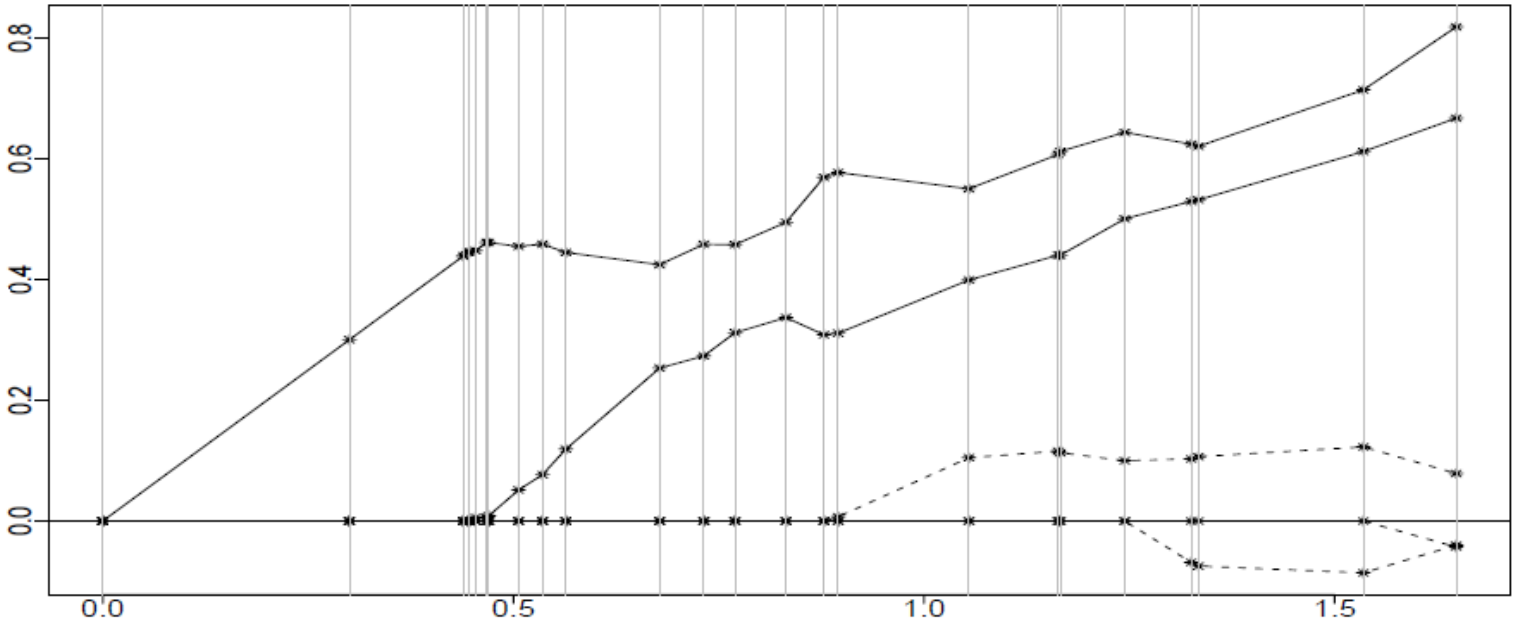
Finding Solution Path

$$\min_{w,b,\xi} \frac{\|w\|}{2} + C\sum_{i=1}^{m} \xi_i$$

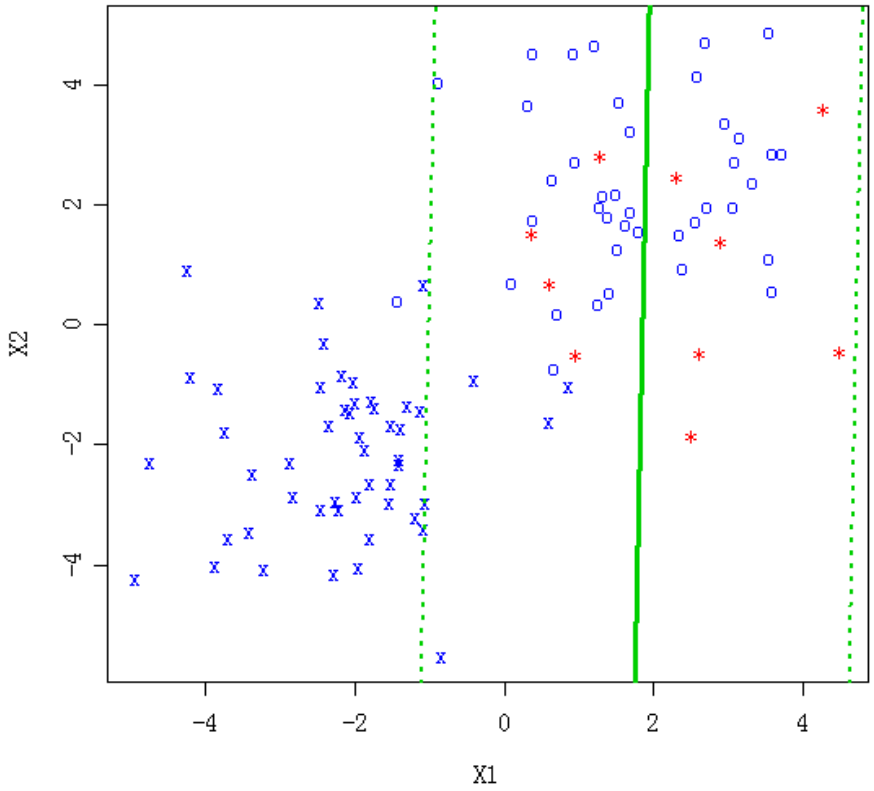
- Can we find *w* and *C* simultaneously (called Solution Path)?
- What is the relationship between w and C?
- **α and 1/C is piecewise-linear!**
- We especially focus on PU (Positive and Unlabeled) classification.

Piecewise Linearity

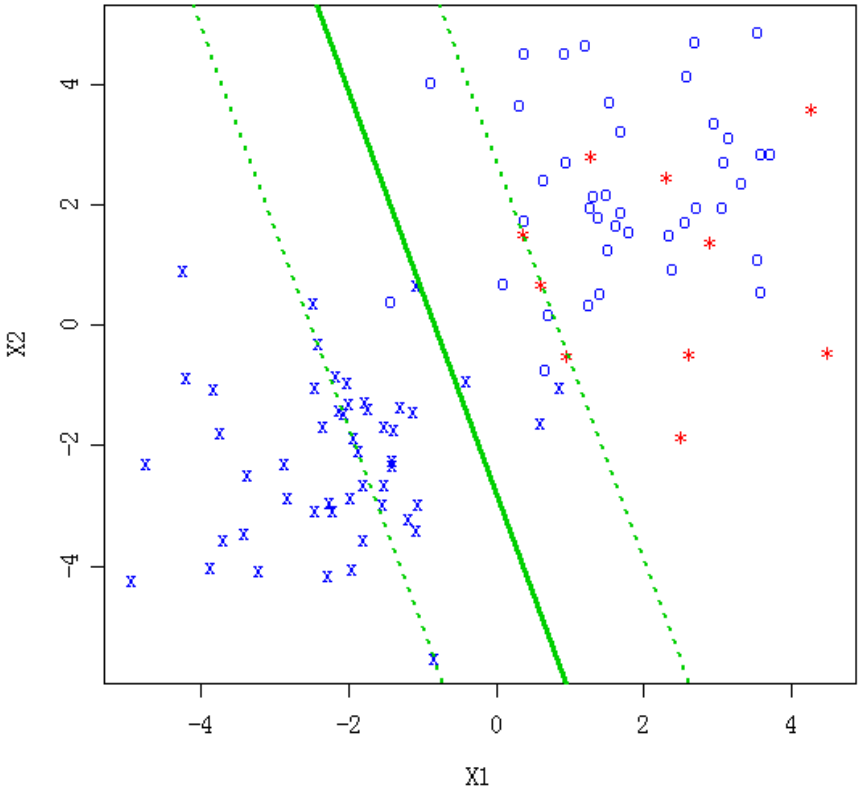




PU Classification Problem



Hyperplane obtained by a traditional SVM



Ideal hyperplane

o: unlabeled positive points
x: unlabeled negative points
*: labeled positive points

Problem Setting

- Data: $S=\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$, and $x_i \in X=R^n$, $y_i \in Y=\{1,-1\}$, the first k-1 positive, the remaining unlabeled, which we consider as negative.
- Notations:
  - $I_+=\{1,2,\ldots,k-1\}$, $n_+=|I_+|$
  - $I_-=\{k,k+1,\ldots,n\}$, $n_-=|I_-|$

$$I = I_+ \cup I_-$$

PU-SVM

- We only allow errors in unlabeled examples, so we have:

$$\min_{\beta} \frac{1}{2}\|\beta\|^2 + C\sum_{i=k}^{n}\xi_i$$

$$s.t. \quad y_i(\beta_0 + \beta^T x_i) \geq 1 \qquad i \in I_+$$

$$y_i(\beta_0 + \beta^T x_i) \geq 1 - \xi_i \qquad i \in I_-$$

$$\xi_i \geq 0 \qquad i \in I_-$$

Eliminating C, we have:

$$\min_{\beta} \sum_{i=k}^{n}\xi_i + \frac{\lambda}{2}\beta^T\beta \qquad \lambda = \frac{1}{C}$$

Note: To distinguish from the conventional formula, we use β here

PU-SVM (cont.)

- Lagrangian primal formula:

$$\sum_{i=k}^{n}\xi_i + \frac{\lambda}{2}\beta^T\beta + \sum_{i=1}^{n}\alpha_i(1-y_i f(x_i)) - \sum_{i=k}^{n}\alpha_i\xi_i - \sum_{i=k}^{n}\gamma_i\xi_i$$

- Set the derivative to zero:

$$\frac{\partial}{\partial\beta}: \quad \beta = \frac{1}{\lambda}\sum_{i=1}^{n}\alpha_i y_i x_i, \qquad i\in I \qquad \alpha_i(1-y_i f(x_i)) = 0, \qquad i\in I_+$$

$$\frac{\partial}{\partial\beta_0}: \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \qquad i\in I \qquad \alpha_i(1-y_i f(x_i)-\xi_i) = 0, \qquad i\in I_-$$

$$\frac{\partial}{\partial\xi_i}: \quad 1-\alpha_i-\gamma_i = 0, \qquad i\in I_- \qquad \gamma_i\xi_i = 0, \qquad i\in I_-$$

Analysis

- $\alpha_i \geq 0$ ($i \in I_+$)
- $\alpha_i > 0$ ($i \in I_+$) , positive support vector
- $1 > \alpha_i > 0$ ($i \in I_-$), negative support vector
- $\alpha_i = 0$ ($i \in I$), outside the margin
- $\alpha_i = 1$ ($i \in I_-$), negative inside the margin

The Entire Generalization Path for PU-SVM

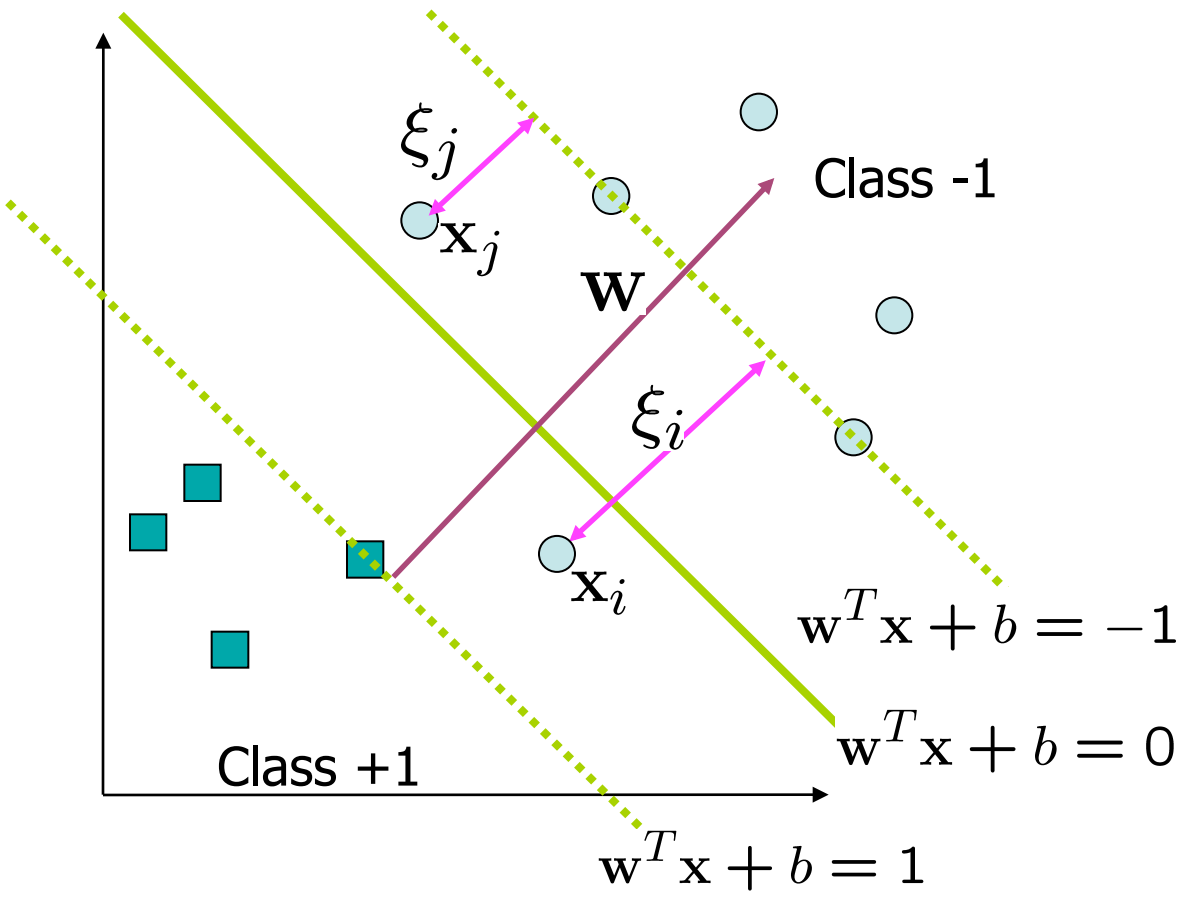
- Initial hyper-plane, find a large λ while keeping all positive points correct, all negative points inside the margin
- λ decreases toward zero, β increases, and the margin $2/\|\beta\|$ decreases

$$\frac{\partial}{\partial \beta}: \qquad \beta = \frac{1}{\lambda}\sum_{i=1}^{n}\alpha_i y_i x_i, \qquad i \in I$$

- Compute α based on the fact that α_i changes linearly with λ
- Termination

The Approach

- Each point x_i belongs to one of the following three sets:
  - M={i: $y_i f(x_i) = 1$, $\alpha_i^p > 0$, $1 > \alpha_i^n > 0$}. M for on the margin ($f(x) = \pm 1$).
  - L={i: $y_i f(x_i) < 1$, $\alpha_i^n = 1$}. L for inside the margin, only unlabeled points.
  - R={i: $y_i f(x_i) > 1$, $\alpha_i = 0$}, R for outside the margin.



The Approach (cont.)

- Points in various sets may have various events
- By tracking all the events in iterations, we can solve the entire generalization path and select the best solution.
- Events:
  - a. The initial event, which means two or more points enter the margin.
  - b. A point leaves from R to enter M, with its value of α_i initially 0;
  - c. A point $i \in I_-$ leaves from L to enter M, with its value of α_i initially 1;
  - d. One or more points in M leave to join R.
  - e. One or more points $i \in I_-$ in M enter L.

The PU-SvmPath Algorithm

- The algorithm consists of three steps:
- Step 1: Initialization
  - It determines the initial values for β, α and λ.
- Step 2: Finding entire generalization path
  - The algorithm runs iteratively.
  - In each iteration, we try to find a new λ_{l+1}, based on the current value λ_l (l denotes the l-th iteration).
- Step 3: Termination
  - The algorithm runs until some terminal conditions are satisfied.

Step 1: Initialization

- Finding the initial value of β_0, α, and λ_0 , as well as a hyper-plane

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j x_i x_j y_i y_j$$

$$s.t. \quad \alpha_i \geq 0, i \in I_+$$

$$1 \geq \alpha_i \geq 0, i \in I_-$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\beta^*(\alpha) = \sum_{i=1}^{n} \alpha_i y_i x_i$$

Step 1: Initialization (cont.)

- Objective function:

$$\min_{\alpha} \left\| \beta^*(\alpha) \right\|^2$$

$$s.t. \quad \alpha_i \in [0,1], i \in I_+$$

$$\alpha_i = 1, i \in I_-$$

$$\sum_{i \in I_+} \alpha_i = n_-$$

  - Solution: some $\alpha_i^p > 0$, other $\alpha_i^p = 0$
    - i_+: the index corresponding to $\alpha_i^p > 0$
- $i_- = \text{argmin}_{i \in I_-} \beta^T x_i$
- $\beta^T x_{i_+} + \beta_0 = 1$, $\beta^T x_{i_-} + \beta_0 = -1$

$$\lambda_0 = \frac{\beta^{*T} x_{i_+} - \beta^{*T} x_{i_-}}{2} \qquad \beta_0 = -\frac{\beta^{*T} x_{i_+} + \beta^{*T} x_{i_-}}{\beta^{*T} x_{i_+} - \beta^{*T} x_{i_-}}$$

For how to solve the function, please refer to:

Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. 2004. The Entire Regularization Path for the Support Vector Machine. J. Mach. Learn. Res. 5 (December 2004), 1391-1415.

Step 2: Finding entire generalization path

- For each point, there are **several** possible events
  - For example, a positive point on the margin has two events: leave *M* into *L* and leave *M* into *R.*
- **Compute** the new λ for every possible event
- **Select** the maximal value as the next λ
- Now, the problem is how to compute the λ for every event

Finding λ_{l+1}

$$f(x) = \frac{1}{\lambda}(\sum_{i=1}^{n} y_j \alpha_j (x \bullet x_j) + \alpha_0)$$

$$f(x) = [f(x) - \frac{\lambda_l}{\lambda} f^l(x)] + \frac{\lambda_l}{\lambda} f^l(x)$$

$$= \frac{1}{\lambda}[\sum_{j \in M_l} (\alpha_j - \alpha_j^l) y_j (x \bullet x_j) + (\alpha_0 - \alpha_0^l) + \lambda_l f^l(x)]$$

$y_i f(x_i) = 1$

$$y_i f(x_i)$$

$$= \frac{1}{\lambda}[\sum_{j \in M_l} (\alpha_j - \alpha_j^l) y_j y_i (x_i \bullet x_j) + y_i (\alpha_0 - \alpha_0^l) + \lambda_l] = 1$$

- Notations:
  - l represents the l-th event
  - $|M_l|=m$: m points on the margin
  - α^i_l, β^0_l, and λ_l , f_l
  - $\alpha^0=\lambda\beta^0$, and hence $\alpha^0_l=\lambda\beta^0_l$

Finding λ_{l+1} (cont.)

$\delta_j = \alpha_j^l - \alpha_j$

$$\sum_{j \in M_l} \delta_j y_i y_j (x_i \bullet x_j) + y_i \delta_0 = \lambda_l - \lambda, \forall i \in M_l$$

$$\sum_{j \in M_l} \delta_j y_j = 0$$

The two equations constitute m+1 linear equations in m+1 unknown δ_j, and can be solved.

Solve the equations, get $\delta_j = (\lambda_l - \lambda) b_j$

$$\alpha_j = \alpha_j^l - (\lambda_l - \lambda) b_j, j \in 0 \cup M_l$$

α_j changes piecewise-linearly with λ

Finding λ_{l+1} (cont.)

$$f(x) = [f(x) - \frac{\lambda_l}{\lambda} f^l(x)] + \frac{\lambda_l}{\lambda} f^l(x)$$

$$= \frac{1}{\lambda}[\sum_{j \in M_l} (\alpha_j - \alpha_j^l) y_j (x \bullet x_j) + (\alpha_0 - \alpha_0^l) + \lambda_l f^l(x)]$$

the function changes in an inverse manner in λ for $\lambda_l > \lambda > \lambda_{l+1}$.

can be rewritten as

$$f(x) = \frac{\lambda_l}{\lambda}[f^l(x) - h^l(x)] + h^l(x)$$

with $h^l(x) = \sum_{j \in M_l} y_j b_j (x \bullet x_j) + b_0$

Computing λ_{l+1}

- For event d and e:
  - when one of the point x_i leave *M* to enter *L* or *R*, we have its α_i=1 or α_i=0. We can compute its λ by:

$$\alpha_j = \alpha_j^l - (\lambda_l - \lambda)b_j, j \in 0 \cup M_l \Rightarrow \lambda = \frac{(1-\alpha_j^l)}{b_j} + \lambda_l \qquad \lambda = \frac{(0-\alpha_j^l)}{b_j} + \lambda_l$$

- For event b and c:
  - when one of points from *L* or *R* to enter *M*, we have $y_i f(x_i)$=1. We can obtain its λ by:

$$f(x) = \frac{\lambda_l}{\lambda}[f^l(x) - h^l(x)] + h^l(x) \Rightarrow \lambda = \frac{\lambda_l}{y_i - h^l(x)}[f^l(x) - h^l(x)]$$

Selecting λ_{l+1}

- Select the largest $\lambda<\lambda_l$ as the λ_{l+1}
- Update α_j, according to α_j change piecewise-linearly in λ

$$\alpha_j = \alpha_j^l - (\lambda_l - \lambda)b_j$$

Step 3: Terminal Condition

$$f(x) = \frac{\lambda_l}{\lambda}[f^l(x) - h^l(x)] + h^l(x)$$

$\lambda \downarrow 0$, without *f(x) blowing up,* we must have $f_l - h_l = 0$

Therefore, our terminal conditions are:

- $\lambda = 0$
- $f_l - h_l = 0$

Kernels

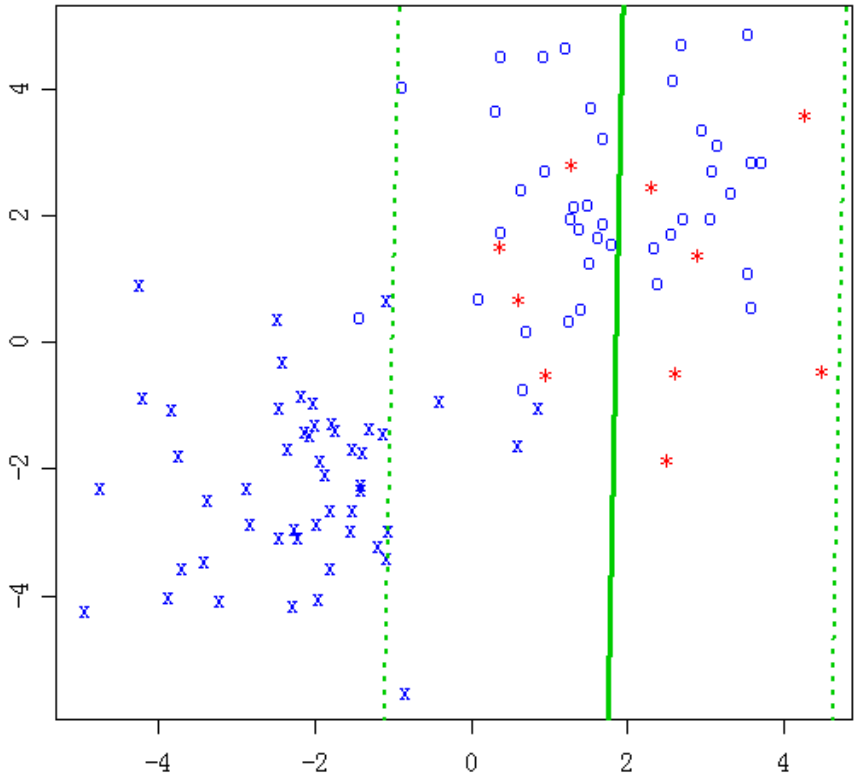
The proposed algorithm can be easily extended to the kernel form:

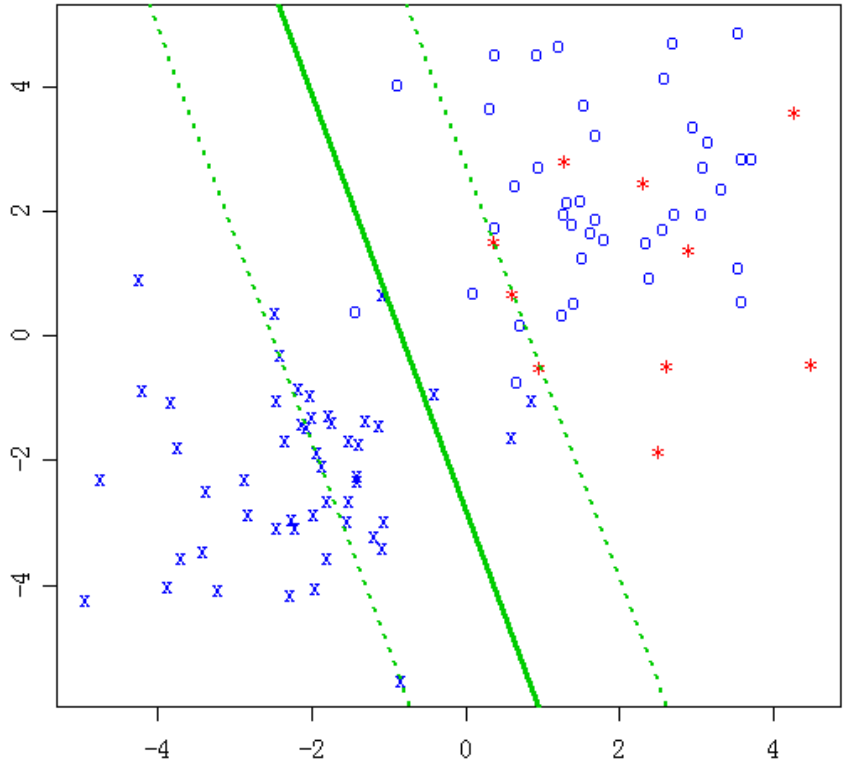
$$f(x) = \frac{1}{\lambda}\left(\sum_{i=1}^{n} y_j \alpha_j (x \bullet x_j) + \alpha_0\right)$$

Replace by

$$f(x) = \frac{1}{\lambda}\left(\sum_{i=1}^{n} y_j \alpha_j K(x \bullet x_j)\right) + \beta_0$$

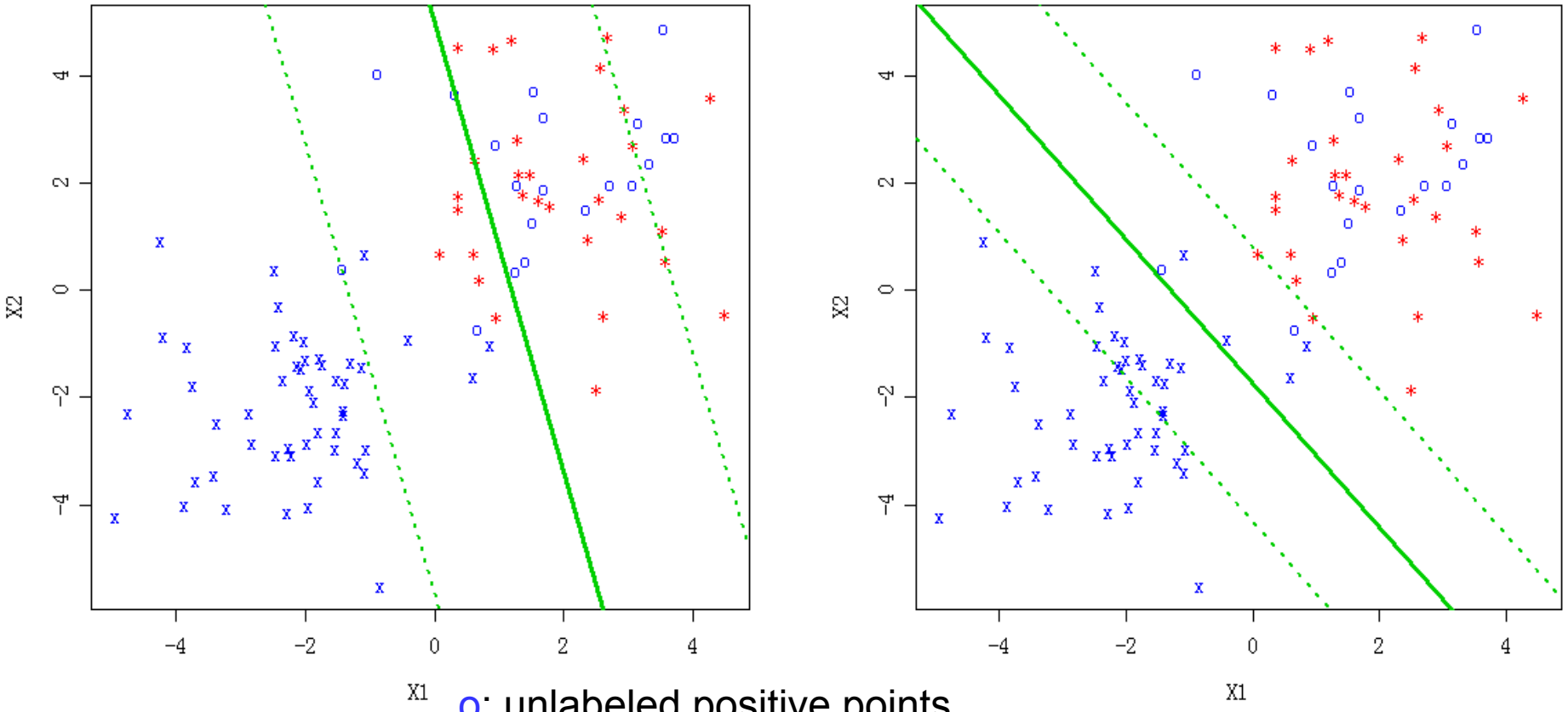
Experimental Results on Synthesis Data





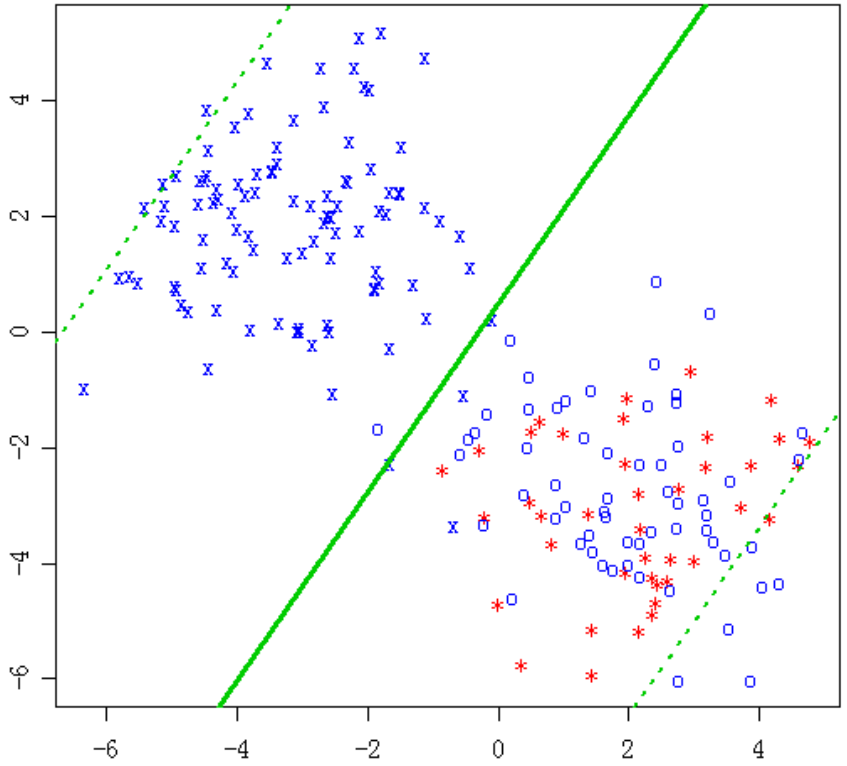
o: unlabeled positive points
x: unlabeled negative points
*: labeled positive points
left: SvmPath, right: PU-SvmPath

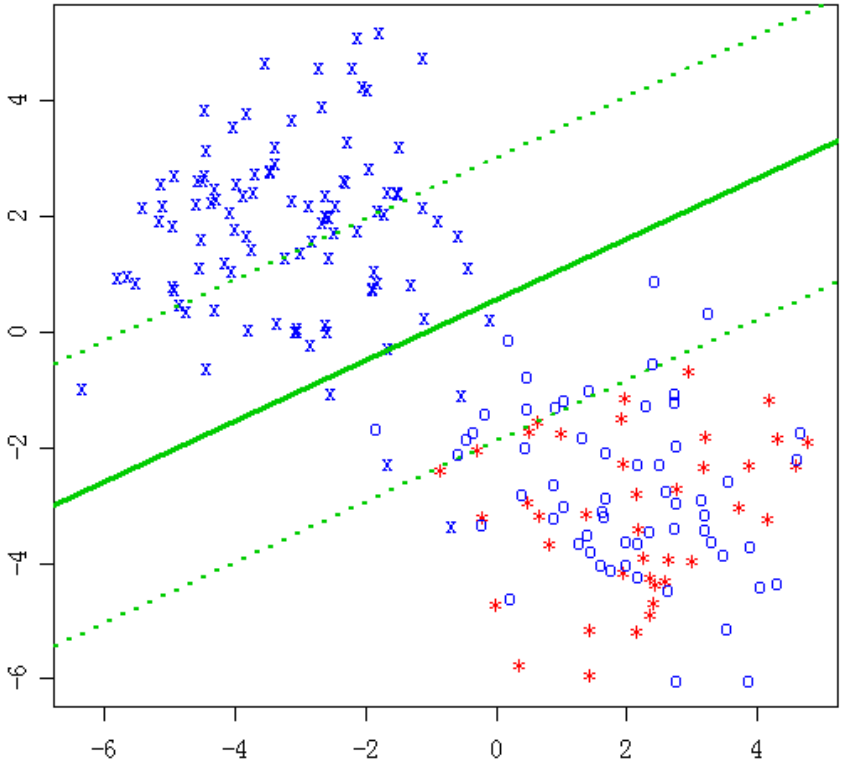
Experimental Results (cont.)



o: unlabeled positive points
x: unlabeled negative points
*: labeled positive points
left: SvmPath, right: PU-SvmPath

Experimental Results (cont.)





o: unlabeled positive points
x: unlabeled negative points
*: labeled positive points
left: SvmPath, right: PU-SvmPath

Performances on Bio-medical data

γ	SvmPath	PU-SvmPath	SVM-Light	Bias-SVM	One-class SVM
0.50	43.48	**47.06**	0.00	15.38	0.00
0.75	41.67	**73.68**	28.57	58.82	0.00

Performances on 20newsgroup

γ	SvmPath	PU-SvmPath	SVM-Light	Bias-SVM	One-class SVM
0.1	13.72	11.10	3.92	11.09	14.27
0.2	29.88	21.58	4.46	21.85	21.95
0.3	40.33	35.83	3.46	36.60	24.06
0.4	39.55	45.34	7.60	50.43	24.62
0.5	56.15	55.22	13.02	60.42	24.45
0.6	57.87	60.17	24.63	71.00	24.81
0.7	67.88	69.72	43.35	80.51	24.66
0.8	68.47	78.20	65.06	87.79	25.05
0.9	73.54	85.45	76.81	93.22	24.63

Summary

- SVM-light only work well with enough labeled examples and bad with few labeled examples
- Bias-SVM achieves better results than SVM-light
  - However, its performances are sensitive to the data sets
- One-class SVM can learn a classifier with only positive examples
  - However, it cannot take advantage of the unlabeled data
- SvmPath can fit the entire regularization path
  - However, it is proposed for classical classification, not for PU classification

1-norm Support Vector Machines

Ji Zhu, Saharon Rosset, Trevor Hastie, Rob Tibshirani. 2003. 1-norm Support Vector Machines. NIPS 2003.

1-norm Support Vector Machines

- 1-norm SVM may have some advantage over the standard 2-norm SVM, especially when there are redundant noise features.

$$\min_{\beta_0,\beta} \quad \sum_{i=1}^{n}\left[1-y_i\left(\beta_0+\sum_{j=1}^{q}\beta_j h_j(x_i)\right)\right]_+$$

$$\text{s.t.} \quad \|\beta\|_1 = |\beta_1|+\cdots+|\beta_q| \le s,$$

1-norm constraint

Where $D=\{h_j(x_i)\}_i$ is a dictionary of basis functions and *s* is a tuning parameter to control the tradeoff.

Piece-wise linearity

- Another form 1-norm SVM:

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} \left[1 - y_i \left(\beta_0 + \sum_{j=1}^{q} \beta_j h_j(x_i)\right)\right]_+ + \lambda\|\beta\|_1,$$

Lasso penalty

Hinge Loss

*note: ||.||^2 is called *ridge* penalty

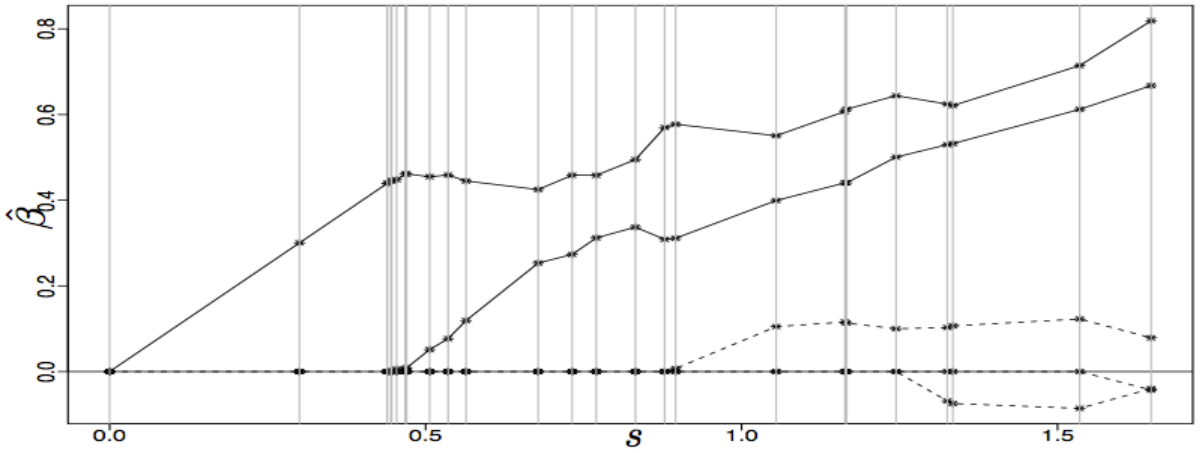
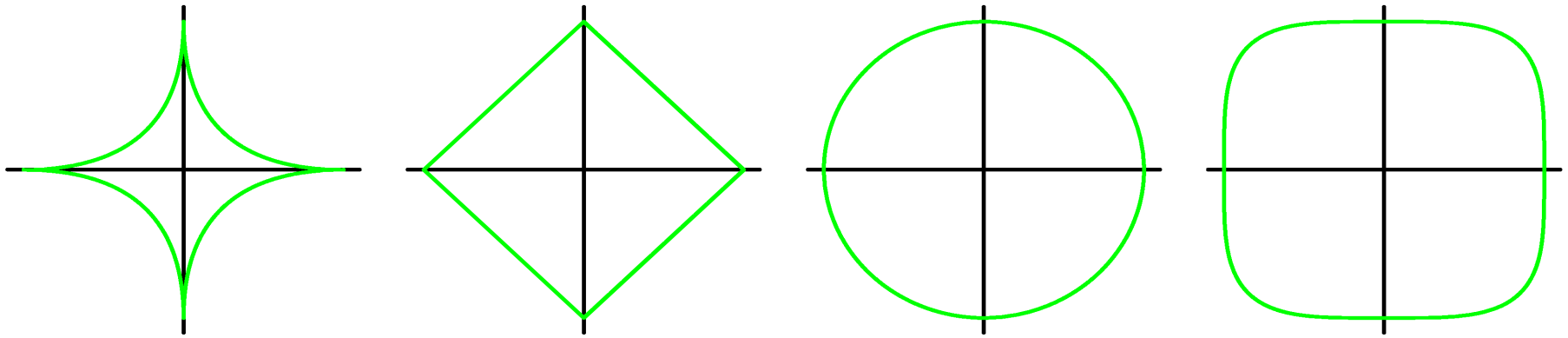


Figure 1: The solution path $\hat{\beta}(s)$ as a function of s.

Regularized Least Squares (2)

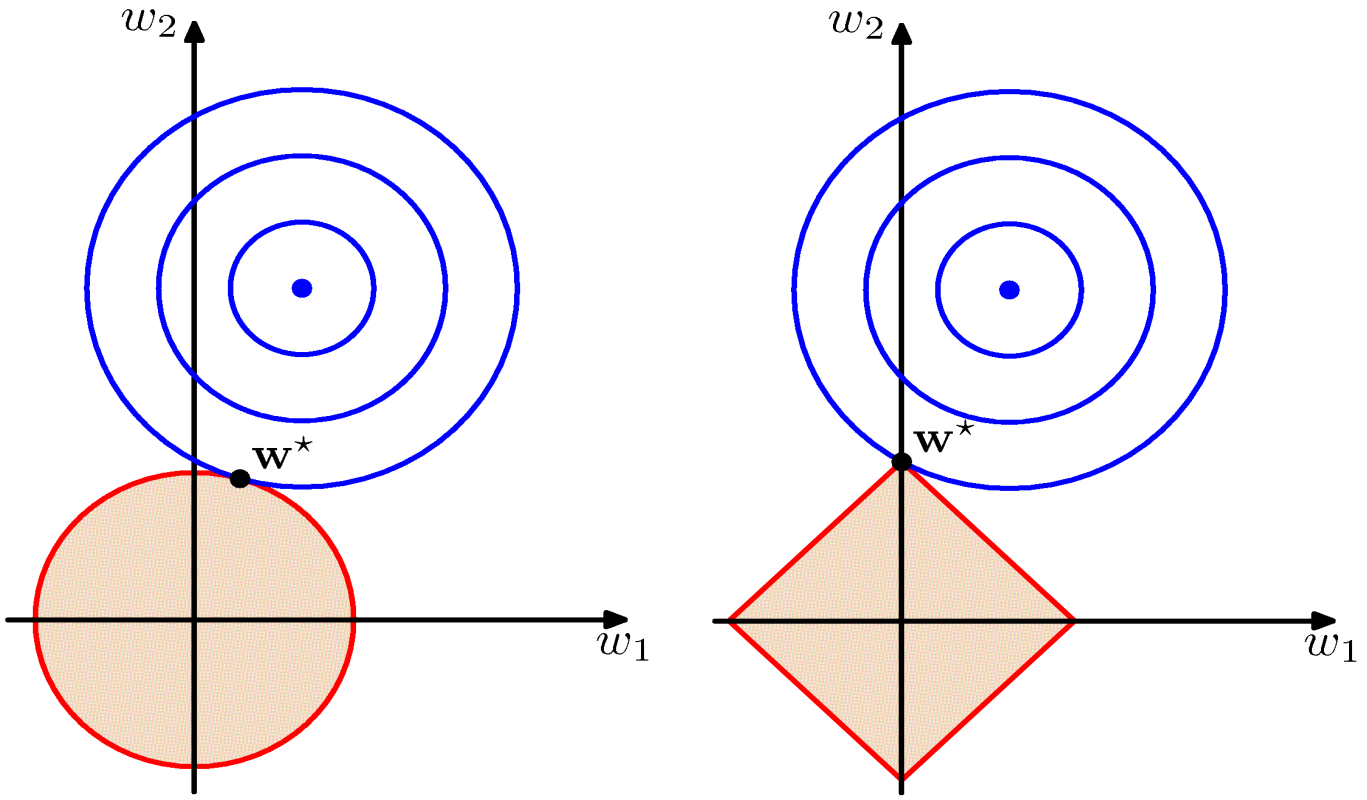
- With a more general regularizer, we have

$$\frac{1}{2}\sum_{i=1}^{N}(1-y_i\beta^T\mathbf{x}_i)^2+\frac{\lambda}{2}\|\beta\|^q$$



Regularized Least Squares (3)

- Lasso tends to generate sparser solutions than a quadratic regularizer.



Solving 1-norm SVM

$$\min_{\beta_0,\beta} \quad \sum_{i=1}^{n}\left[1-y_i\left(\beta_0+\sum_{j=1}^{q}\beta_j h_j(x_i)\right)\right]_+$$

$$\text{s.t.} \quad \|\beta\|_1 = |\beta_1|+\cdots+|\beta_q| \le s,$$

Initial solution

- We consider a modified problem:

$$\min_{\beta_0,\beta_j} \quad \sum_{y_i=1}(1-y_i f_i)_+ + \sum_{y_i=-1}(1-y_i f_i)$$

$$\text{s.t.} \quad \|\beta\|_1 \le \Delta s, \quad f_i = \beta_0 + \sum_{j=1}^{q}\beta_j h_j(x_i).$$

if $y_i = 1$, the loss is still $(1-y_i f_i)_+$

but if $y_i = -1$, the loss becomes $(1-y_i f_i)$

- Transform into a simple linear programming problem:

$$\begin{pmatrix} \hat{\beta}_0(\Delta s) \\ \hat{\beta}_{\mathcal{V}}(\Delta s) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \Delta s \cdot \begin{pmatrix} u_0 \\ u \end{pmatrix}.$$

Δs starts at 0 and increases.

Main algorithm

1. Increase Δs until one of the following two events happens:
   - A training point hits $\mathcal{E}$, i.e. $1 - y_i f_i \neq 0$ becomes $1 - y_i f_i = 0$ for some i.
   - A basis function in $\mathcal{V}$ leaves $\mathcal{V}$, i.e. $\hat{\beta}_j \neq 0$ becomes $\hat{\beta}_j = 0$ for some j.

   Let the current $\hat{\beta}_0$, $\hat{\beta}$ and s be denoted by $\hat{\beta}_0^{old}$, $\hat{\beta}^{old}$ and s^{old}.

2. For each $j^* \notin \mathcal{V}$, we solve:

$$\begin{cases} u_0 + \sum_{\mathcal{V}} u_j h_j(x_i) + u_{j^*} h_{j^*}(x_i) &= 0 \quad \text{for } i \in \mathcal{E} \\ \sum_{\mathcal{V}} sign(\hat{\beta}_j^{old}) u_j + |u_{j^*}| &= 1 \end{cases}$$

   where u_0, u_j and u_{j^*} are the unknowns. We then compute:

$$\frac{\Delta loss_{j^*}}{\Delta s} = \sum_{\mathcal{L}} y_i \left(u_0 + \sum_{\mathcal{V}} u_j h_j(x_i) + u_{j^*} h_{j^*}(x_i) \right).$$

3. For each $i' \in \mathcal{E}$, we solve:

$$\begin{cases} u_0 + \sum_{\mathcal{V}} u_j h_j(x_i) &= 0 \quad \text{for } i \in \mathcal{E} \backslash \{i'\} \\ \sum_{\mathcal{V}} sign(\hat{\beta}_j^{old}) u_j &= 1 \end{cases}$$

   where u_0 and u_j are the unknowns. We then compute:

$$\frac{\Delta loss_{i'}}{\Delta s} = \sum_{\mathcal{L}} y_i \left(u_0 + \sum_{\mathcal{V}} u_j h_j(x_i) \right).$$

4. Compare the computed values of $\frac{\Delta loss}{\Delta s}$ from step 2 and step 3. There are $q - |\mathcal{V}| + |\mathcal{E}| = q + 1$ such values. Choose the smallest negative $\frac{\Delta loss}{\Delta s}$. Hence,
   - If the smallest $\frac{\Delta loss}{\Delta s}$ is non-negative, the algorithm terminates; else
   - If the smallest negative $\frac{\Delta loss}{\Delta s}$ corresponds to a j^* in step 2, we update

$$\mathcal{V} \leftarrow \mathcal{V} \cup \{j^*\}, \quad u \leftarrow \begin{pmatrix} u \\ u_{j^*} \end{pmatrix}.$$

   - If the smallest negative $\frac{\Delta loss}{\Delta s}$ corresponds to a i' in step 3, we update u and $\mathcal{E} \leftarrow \mathcal{E} \backslash \{i'\}$, $\mathcal{L} \leftarrow \mathcal{L} \cup \{i'\}$ if necessary.

   In either of the last two cases, $\hat{\beta}(s)$ changes as:

$$\begin{pmatrix} \hat{\beta}_0(s^{old} + \Delta s) \\ \hat{\beta}_{\mathcal{V}}(s^{old} + \Delta s) \end{pmatrix} = \begin{pmatrix} \hat{\beta}_0^{old} \\ \hat{\beta}_{\mathcal{V}}^{old} \end{pmatrix} + \Delta s \cdot \begin{pmatrix} u_0 \\ u \end{pmatrix},$$

   and we go back to step 1.

$O(nq \min(n, q)^2)$ in the worst case and $O(nq)$ in the best case.

Numerical results

Table 1: Simulation results of 1-norm and 2-norm SVM

	Simulation	Test Error (SE) 1-norm	2-norm	No Penalty	$\lvert\mathcal{D}\rvert$	# Joints
1	No noise input	0.073 (0.010)	0.08 (0.02)	0.08 (0.01)	5	94 (13)
2	2 noise inputs	0.074 (0.014)	0.10 (0.02)	0.12 (0.03)	14	149 (20)
3	4 noise inputs	0.074 (0.009)	0.13 (0.03)	0.20 (0.05)	27	225 (30)
4	6 noise inputs	0.082 (0.009)	0.15 (0.03)	0.22 (0.06)	44	374 (52)
5	8 noise inputs	0.084 (0.011)	0.18 (0.03)	0.22 (0.06)	65	499 (67)

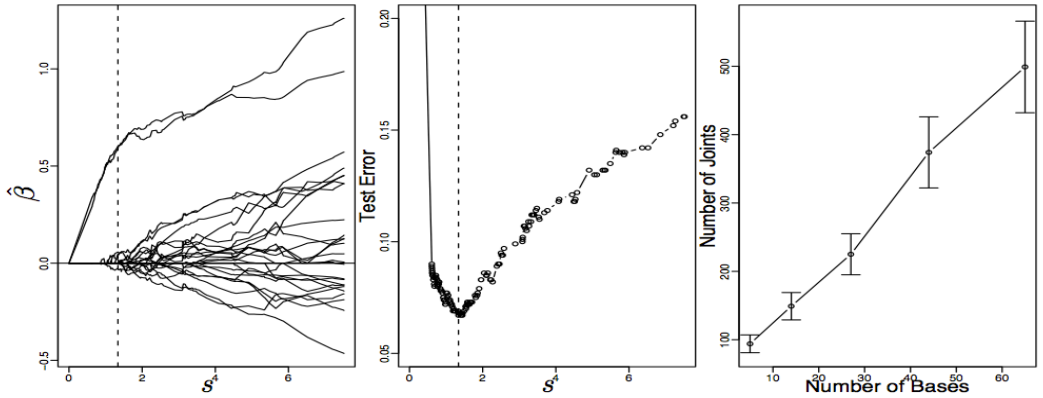


Figure 2: Left and middle panels: 1-norm SVM when there are 4 noise inputs. The left panel is the piece-wise linear solution path $\hat{\beta}(s)$. The two upper paths correspond to x_1^2 and x_2^2, which are the relevant features. The middle panel is the test error along the solution path. The dash lines correspond to the minimum of the test error. The right panel illustrates the linear relationship between the number of basis functions and the number of joints on the solution path when $q < n$.

Table 2: Results on Microarray Classi£cation

Method	CV Error	Test Error	# of Genes
2-norm SVM UR	2/38	3/34	22
2-norm SVM RFE	2/38	1/34	31
1-norm SVM	2/38	2/34	17

A Walk from 2-Norm SVM to 1-Norm SVM

- A simple iterative scheme which approaches the 1-norm SVM by solving a series of standard 2 -norm SVM problems.
- Each 2-norm SVM solves a problem where the features are weighted depending on the solution of the previous 2-norm SVM problem. Hence, we will refer to this algorithm as the ***re -weighting algorithm (RW).***
- More generally, we can apply it to minimize any convex error function regularized with 1-norm.

$$\|w\|_1 + \underbrace{C\sum_{i=1}^{m}(y_i - w\cdot x_i)^2}_{E_{\text{squared}}(w)}.$$

A simple linear regression problem, where we wish to minimize over w the following expression for continuous outputs yi

$$w^{(t+1)} = \arg\min_w \frac{1}{2}\sum_{i=1}^{d}\frac{w_i^2}{|w_i^{(t)}|} + E_{\text{squared}}(w),$$

EM algorithm for computing the MAP estimate of the weight vector w for the Laplacian prior, and arrived at the following iterative updates

The above expression reduces a 1-norm regularized least-squares problem to 2-norm regularized least-squares problem, where the feature i is weighted with $\sqrt{|w_i^{(t)}|}$.

Algorithm

Algorithm 1 1-norm SVM via 2-norm SVM

Input: a training set $\{(x^i, y^i)\}_{i=1}^m$ and number of iterations N.
Output: a weight vector w.
Initialize vector $v^{(1)}$ to all ones.
for $t = 1$ **to** N **do**
  for each example x^i **do**
    for each coordinate j **do**
      Set $x'^i_j := x^i_j v^{(t)}_j$.
    end for
  end for
  $w^{(t)} :=$
  solution to SVM with examples $\{(x'^i, y^i)\}_{i=1}^m$.
  for each coordinate i **do**
    Set $v_i^{(t+1)} := \sqrt{|w_i^{(t)} v_i^{(t)}|}$.
  end for
end for
for each coordinate i **do**
  Set $w_i := w_i^{(N)} v_i^{(N)}$.
end for
Return w

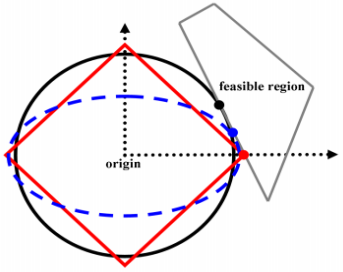


Fig. 1. 2-norm contour as the black circle, 1-norm as the red tilted square, and scaled 2-norm with the blue dashed ellipse.

The benefits of L1-regularization over L2-regularization.

1) *Hard-margin:*

Theorem 1. *Let u be any vector and define v as the weight vector induced by u, i.e., $u = v \otimes |v|$. Assume that w satisfies all constraints $y_i(w \cdot (x_i \otimes |v|)) \geq 1$. The weighted problem implies a new solution $u^{new} = w \otimes |v|$ for the original unweighted problem. Now, if $\|w\|^2 < \|v\|^2$, then $\|u^{new}\|_1 < \|u\|_1$.*

2) *Soft-margin and Generalization to Any Error Function:*

Theorem 2. *If $u = v \otimes |v|$ and $u^{new} = w \otimes |v|$ and*

$$\frac{1}{2}\|w\|^2 + E_v(w) < \frac{1}{2}\|v\|^2 + E_v(v), \tag{4}$$

then u^{new} is a better solution to the L1-regularized optimization than u:

$$\|u^{new}\|_1 + E(u^{new}) < \|u\|_1 + E(u).$$

3) *Convex Error Gives Convergence:*

Theorem 3. *Let h be a vector which is normalized so that $\|h\|_u = 1$. Let vector $c^\star h$, where $c^\star$ is a scalar, denote any direction in which the L1-regularized objective value decreases when starting from u. Hence,*

$$\|u + c^\star h\|_1 + E(u + c^\star h) < \|u\|_1 + E(u).$$

Then the weighted objective function decreases to that same direction, i.e., there is a scalar $c > 0$ such that

$$\frac{1}{2}\|u + c\,h\|_u + E(u + c\,h) + \frac{1}{2}c^2 \leq \frac{1}{2}\|u\|_u + E(u).$$

More precisely, the step size c is at least the minimum of $c^\star$ and

$$-\left(\sum_{i=1}^{d} sign\,(u_i)\,h_i\right) + \frac{1}{c^\star}\left(E\,(u) - E\,(u + c^\star h)\right).$$

Analysis

TABLE II
CLASSIFICATION ACCURACIES OF THE RW ALGORITHM AND 1-NORM SVM RUNS WITH DIFFERENT NUMBERS OF ITERATIONS. THE EMPIRICAL STANDARD DEVIATION OF THE ACCURACY IS GIVEN. THE i-TH ITERATION OF THE RW ALGORITHM IS DENOTED BY RW(i). FOR GISETTE AND REUTERS OUR 1-NORM SVM SOLVER RUNS OUT OF MEMORY.

DATA SET	2-NORM SVM	RW(2)	RW(3)	RW(5)	RW(10)	1-NORM SVM
REUTERS	93.4	93.5	93.5	93.4	93.3	NA
REUTERS-SAMPLED	85.9 ± 0.2	84.9 ± 0.3	83.7 ± 0.3	82.2 ± 0.3	80.6 ± 0.3	72.8 ± 0.8
GISETTE	97.8	98.3	98.5	98.2	98.1	NA
DLBCL	68.6 ± 1.0	91.4 ± 0.7	93.2 ± 0.7	93.5 ± 0.8	91.7 ± 1.1	85.3 ± 1.3
LEUKEMIA	82.7 ± 1.0	95.2 ± 0.6	96.0 ± 0.5	95.8 ± 0.6	94.7 ± 0.8	90.1 ± 1.5
LUNG	92.9 ± 0.4	95.1 ± 0.3	95.0 ± 0.4	94.5 ± 0.5	93.8 ± 0.6	92.6 ± 0.5
PROSTATA	90.8 ± 0.6	91.4 ± 0.5	92.0 ± 0.5	92.5 ± 0.5	91.6 ± 0.6	88.5 ± 0.8
SRBCT	88.8 ± 1.1	98.1 ± 0.4	98.7 ± 0.3	98.0 ± 0.5	96.8 ± 0.6	95.7 ± 0.5
SONAR	73.3 ± 0.7	73.0 ± 0.6	73.1 ± 0.7	72.8 ± 0.8	72.6 ± 0.7	72.2 ± 0.8

RW algorithm is competitive with the 1-norm SVM already when run with a low number of iterations

How fast the RW algorithm converges to the 1-norm SVM solution

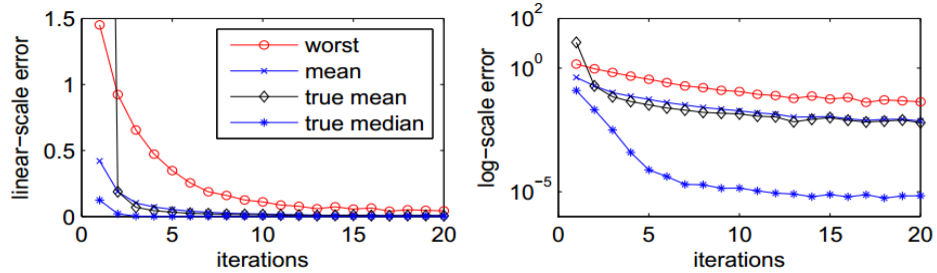
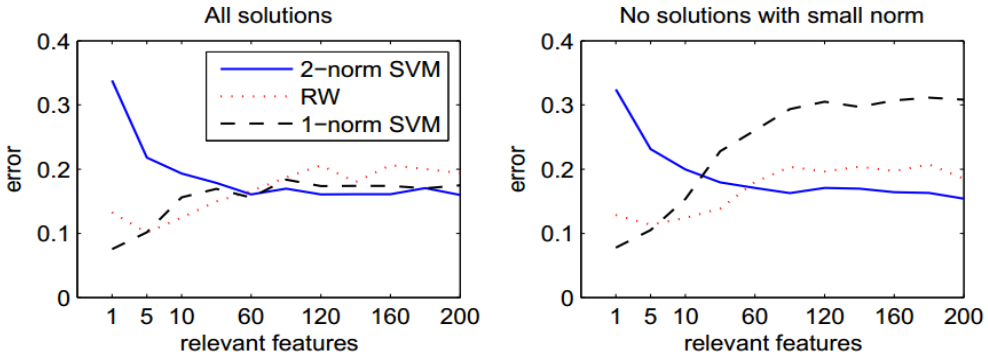


Fig. 2. The behavior of the objective with different number of iterations. The first two curves are derived from worst-case performances over C on different data sets at the 20th iteration. The true mean and median are derived over all data sets and all values of the tradeoff parameter C.



In the left graph the 1-norm SVM solver performs well when the number of relevant features is high.

In the right graph the 1-norm SVM SVM is better only when the data set has a single relevant feature

Fig. 3. How the error behaves as a function of relevant features. The number of features is 200. The solvers are the 2-norm SVM, the RW algorithm with one iteration, and the 1-norm SVM.

Initial solution

The following notation is used. Let $\mathcal{V} = \{j : \hat{\beta}_j(s) \neq 0\}$, $\mathcal{E} = \{i : 1 - y_i \hat{f}_i = 0\}$, $\mathcal{L} = \{i : 1 - y_i \hat{f}_i > 0\}$ and u for the right derivative of $\hat{\beta}_{\mathcal{V}}(s)$: $\|u\|_1 = 1$ and $\hat{\beta}_{\mathcal{V}}(s)$ denotes the components of $\hat{\beta}(s)$ with indices in $\mathcal{V}$. Without loss of generality, we assume $\#\{y_i = 1\} \geq \#\{y_i = -1\}$; then $\hat{\beta}_0(0) = 1, \hat{\beta}_j(0) = 0$. To compute the path that $\hat{\beta}(s)$ follows, we need to compute the derivative of $\hat{\beta}(s)$ at 0. We consider a modi£ed problem:

$$\min_{\beta_0, \beta_j} \quad \sum_{y_i=1} (1 - y_i f_i)_+ + \sum_{y_i=-1} (1 - y_i f_i) \tag{6}$$

$$\text{s.t.} \quad \|\beta\|_1 \leq \Delta s, \quad f_i = \beta_0 + \sum_{j=1}^{q} \beta_j h_j(x_i). \tag{7}$$

Notice that if $y_i = 1$, the loss is still $(1 - y_i f_i)_+$; but if $y_i = -1$, the loss becomes $(1 - y_i f_i)$. In this setup, the derivative of $\hat{\beta}(\Delta s)$ with respect to Δs is the same no matter what value Δs is, and one can show that it coincides with the right derivative of $\hat{\beta}(s)$ when s is suf£ciently small. Hence this setup helps us £nd the initial derivative u of $\hat{\beta}(s)$. Solving (6)-(7), which can be transformed into a simple linear programming problem, we get initial $\mathcal{V}$, $\mathcal{E}$ and $\mathcal{L}$. $|\mathcal{V}|$ should be equal to $|\mathcal{E}|$. We also have:

$$\begin{pmatrix} \hat{\beta}_0(\Delta s) \\ \hat{\beta}_{\mathcal{V}}(\Delta s) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \Delta s \cdot \begin{pmatrix} u_0 \\ u \end{pmatrix}. \tag{8}$$

Δs starts at 0 and increases.

Reference

- C. Cortes and V. Vapnik. 1995. Support-vector networks. Machine Learning, Vol. 20, 1995, pp273-297
- T. Hastie, S. Rosset, R. Tibshirani and J. Zhu. 2004. The entire regularization path for the support vector machine. Journal of Machine Learning Research, (5):1391-1415, 2004

Thanks!

HP: http://keg.cs.tsinghua.edu.cn/jietang/

Email: jietang@tsinghua.edu.cn