Smooth Support Vector Machines for Classification and Regression

Yuh-Jye Lee
National Taiwan University of Science and Technology

International Summer Workshop on the Economics, Financial and Managerial Applications of Computational Intelligence
August 16~20, 2004
Fundamental Problems in Data Mining

- Supervised learning:
  - Classification problems
  - Regression problems

- Unsupervised learning:
  - Clustering algorithms

- Feature selection
  - Too many features could degrade generalization performance, *curse of dimensionality*

- Association Rules
Binary Classification Problem

(A Fundamental Problem in Data Mining)

- Find a decision function (rule) to discriminate two categories data sets.

- Supervised learning in Machine Learning
  - Decision Tree, Neural Network, k-NN and Support Vector Machines, etc.

- Discrimination Analysis in Statistics
  - Fisher Linear Discriminator

- Successful applications:
  - Marketing, Bioinformatics, Fraud detection
Bankruptcy Prediction

Binary classification of firms: solvent vs. bankrupt

The data are financial indicators from middle-market capitalization firms in Benelux. From a total of 422 firms, 74 went bankrupt and 348 were solvent. The variables to be used in the model as explanatory inputs are 40 financial indicators such as: liquidity, profitability and solvency measurements.

Binary Classification Problem

Linearly Separable Case

\[ x'w + b = 0 \]
\[ x'w + b = +1 \]

A-
Bankrupt

A+
Solvent
Why Use Support Vector Machines?

Powerful tools for Data Mining

- SVM classifier is an **optimally** defined surface
- SVMs have a good geometric interpretation
- SVMs can be generated very efficiently
- Can be extended from linear to **nonlinear** case
  - Typically nonlinear in the input space
  - Linear in a higher dimensional “feature” space
  - Implicitly defined by a *kernel* function
- Have a sound theoretical foundation
  - Based on Statistical Learning Theory
Support Vector Machines

Maximizing the Margin between Bounding Planes

\[ x'w + b = +1 \]

\[ x'w + b = -1 \]

\[ \frac{2}{||w||_2} = \text{Margin} \]
Algebra of the Classification Problem

Linearly Separable Case

Given $l$ points in the $n$ dimensional real space $\mathbb{R}^n$

- Represented by an $\ell \times n$ matrix $A$
- Membership of each point $A_i$ in the classes $A_-$ or $A_+$ is specified by an $\ell \times \ell$ diagonal matrix $D$:

\[ D_{ii} = -1 \text{ if } A_i \in A_- \text{ and } D_{ii} = 1 \text{ if } A_i \in A_+ \]

- Separate $A_-$ and $A_+$ by two bounding planes such that:

\[
A_iw + b \geq +1, \quad \text{for} \quad D_{ii} = +1,
A_iw + b \leq -1, \quad \text{for} \quad D_{ii} = -1
\]

- More succinctly: $D(Aw + eb) \geq e$, where $e = [1, 1, \ldots, 1]' \in \mathbb{R}^\ell$. 
Support Vector Classification

(Linearily Separable Case)

Let \( S = \{(x^1, y_1), (x^2, y_2), \ldots (x^l, y_l)\} \) be a linearly separable training sample and represented by matrices

\[
A = \begin{bmatrix}
(x^1)'
(x^2)'
\vdots
(x^l)'
\end{bmatrix} \in \mathbb{R}^{l \times n}, \quad D = \begin{bmatrix}
y_1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & y_l
\end{bmatrix} \in \mathbb{R}^{l \times l}
\]
Robust Linear Programming

Preliminary Approach to SVM

\[ \min_{w, b, \xi} e'\xi \]

\[ \text{s.t. } D(Aw + eb) + \xi \geq e \]  
\[ \xi \geq 0 \]  

where \( \xi \) : nonnegative slack (error) vector.

\[ \text{The term } e'\xi, \text{ 1-norm measure of error vector, is called the training error.} \]

\[ \text{For the linearly separable case, at solution of (LP):} \]

\[ \xi = 0 \]
Two Different Measures of Training Error

2-Norm Soft Margin:
\[
\min_{(w,b,\xi) \in \mathbb{R}^{n+1+l}} \frac{1}{2} \| w \|^2 + \frac{C}{2} \| \xi \|^2 + D(Aw + eb) + \xi \geq e
\]

1-Norm Soft Margin:
\[
\min_{(w,b,\xi) \in \mathbb{R}^{n+1+l}} \frac{1}{2} \| w \|^2 + Ce'\xi + D(Aw + eb) + \xi \geq e, \quad \xi \geq 0
\]

◆ Margin is maximized by minimizing reciprocal of margin.
1-Norm Soft Margin
Dual Formulation

The Lagrangian for 1-norm soft margin:

\[ \mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2}w'w + Ce'\xi + \alpha'[e - D(Aw + eb) - \xi] - r'\xi \]

where \( \alpha \geq 0 \) & \( r \geq 0 \)

The partial derivatives with respect to primal variables equal equal zeros

\[ \frac{\partial \mathcal{L}(w,b,\xi,\alpha)}{\partial w} = w - A'D\alpha = 0 \]
\[ \frac{\partial \mathcal{L}(w,b,\xi,\alpha)}{\partial b} = e'D\alpha = 0 \]
\[ \frac{\partial \mathcal{L}(w,b,\xi,\alpha)}{\partial \xi} = C - \alpha - r = 0 \]
Dual Maximization Problem
for 1-Norm Soft Margin

Dual:

\[
\begin{align*}
\max_{\alpha \in \mathbb{R}^l} & \quad e'\alpha - \frac{1}{2}\alpha'DAA'D\alpha \\
\text{s.t.} & \quad e'D\alpha = 0 \\
& \quad 0 \leq \alpha \leq Ce
\end{align*}
\]

◆ The corresponding KKT complementarity:

\[
\begin{align*}
0 \leq \alpha & \perp D(Aw + eb) + \xi - e \geq 0 \\
0 \leq \xi & \perp \alpha - Ce \leq 0
\end{align*}
\]
Slack Variables for 1-Norm Soft Margin \( (w^* = A'D\alpha^*) \)

- Non-zero slack can only occur when \( \alpha_i^* = C \)
  - The contribution of outlier in the decision rule will be at most \( C \)
  - The trade-off between accuracy and regularization directly controls by \( C \)

- The points for which \( 0 < \alpha_i^* < C \) lie at the bounding planes
  - This will help us to find \( b^* \)
Tuning Procedure

The final value of parameter is one with the maximum testing set correctness!
SVM as an Unconstrained Minimization Problem

\[
\begin{align*}
\min_{\xi \geq 0, w, b^2} & \frac{C}{2} \| \xi \|^2 + \frac{1}{2} (\| w \|^2 + b^2) \\
\text{s. t.} & \quad D(Aw + eb) + \xi \geq e
\end{align*}
\]

(QP)

At the solution of (QP) : \( \xi = (e - D(Aw + eb))_+ \)

where \((\cdot)_+ = \max\{\cdot, 0\}\).

Hence (QP) is equivalent to the nonsmooth SVM:

\[
\begin{align*}
\min_{w, b} & \quad \frac{C}{2} \|(e - D(Aw + eb))_+\|^2 + \frac{1}{2} (\| w \|^2 + b^2)
\end{align*}
\]

◆ Change (QP) into an unconstrained MP

◆ Reduce \((n+1+m)\) variables to \((n+1)\) variables
Smooth the Plus Function: Integrate

\[ p(x, \beta) := x + \frac{1}{\beta} \log \left( 1 + e^{-\beta x} \right) \]

Step function: \( x_* \)

Sigmoid function: \( \frac{1}{1 + e^{-5x}} \)

Plus function: \( x_+ \)

\( p \)-function: \( p(x, 5) \)
SSVM:

Smooth Support Vector Machine

Replacing the plus function $(\cdot)_+$ in the nonsmooth SVM by the smooth $p(\cdot, \beta)$, gives our SSVM:

$$\min_{(w, b) \in \mathbb{R}^{n+1}} \frac{C}{2} \left\| p((e - D(Aw + eb)), \beta) \right\|_2^2 + \frac{1}{2}(\|w\|_2^2 + b^2)$$

Here, $p(\cdot, \beta)$ is an accurate smooth approximation of $(\cdot)_+$, obtained by integrating the sigmoid function of neural networks. (sigmoid = smoothed step)

The solution of SSVM converges to the solution of nonsmooth SVM as $\beta$ goes to infinity.

(Typically, $\beta = 5$)
Newton-Armijo Method:
Quadratic Approximation of SSVM

The sequence $\{(w^i, b_i)\}$ generated by solving a quadratic approximation of SSVM, converges to the unique solution $(w^*, b^*)$ of SSVM at a quadratic rate.

- Converges in 6 to 8 iterations

- At each iteration we solve a linear system of:
  - n+1 equations in n+1 variables
  - Complexity depends on dimension of input space

- It might be needed to select a stepsize
Newton-Armijo Algorithm

Start with any \((w^0, b_0) \in R^{n+1}\). Having \((w^i, b_i)\), stop if \(\nabla \Phi_\beta(w^i, b_i) = 0\), else:

(i) Newton Direction:

\[
\nabla^2 \Phi_\beta(w^i, b_i) d^i = - \nabla \Phi_\beta(w^i, b_i)'
\]

(ii) Armijo Stepsize:

\[
(w^{i+1}, b_{i+1}) = (w^i, b_i) + \lambda_i d^i
\]

\[
\lambda_i = \{ 1, \frac{1}{2}, \frac{1}{4}, \ldots \}
\]

globally and quadratically converge to unique solution in a finite number of steps
It can not converge to optimum solution!

\[ f(x) = -\frac{1}{6}x^6 + \frac{1}{4}x^4 + 2x^2 \]

\[ g(x) = f(x^i) + f'(x^i)(x - x^i) + \frac{1}{2} f''(x^i)(x - x^i) \]

It can not converge to optimum solution!
Comparisons of SSVM with other SVMs

Tenfold test set correctness % (best in Red)
CPU time in *seconds*

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size $m \times n$</th>
<th>SSVM Linear Eqns.</th>
<th>SVM $| \cdot |_1$ LP</th>
<th>SVM $| \cdot |_2$ QP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleveland Heart</td>
<td>297 x 13</td>
<td>86.13</td>
<td>84.55</td>
<td>72.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.63</td>
<td>18.71</td>
<td>67.55</td>
</tr>
<tr>
<td>BUPA Liver</td>
<td>345 x 6</td>
<td>70.33</td>
<td>64.03</td>
<td>69.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.05</td>
<td>19.94</td>
<td>124.23</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351 x 34</td>
<td>89.63</td>
<td>86.10</td>
<td>89.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.69</td>
<td>42.41</td>
<td>128.15</td>
</tr>
<tr>
<td>Pima Indians</td>
<td>768 x 8</td>
<td>78.12</td>
<td>74.47</td>
<td>77.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.54</td>
<td>286.59</td>
<td>1138.0</td>
</tr>
<tr>
<td>WPBC(24 months)</td>
<td>155 x 32</td>
<td>83.47</td>
<td>71.08</td>
<td>82.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.32</td>
<td>6.25</td>
<td>12.50</td>
</tr>
<tr>
<td>WPBC(60 months)</td>
<td>110 x 22</td>
<td>68.18</td>
<td>66.23</td>
<td>61.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.03</td>
<td>3.72</td>
<td>4.91</td>
</tr>
</tbody>
</table>
Two-spiral Dataset
(94 White Dots & 94 Red Dots)
Dual Representation of SVM

(Key of Kernel Methods: \( w = A'D\alpha^* = \sum_{i=1}^{l} y_i\alpha_i^* A_i' \))

The hypothesis is determined by \((\alpha^*, b^*)\)

\[
h(x) = \text{sgn}(\langle x, A'D\alpha^* \rangle + b^*) \\
= \text{sgn}(\sum_{i=1}^{l} y_i\alpha_i^* \langle x^i, x \rangle + b^*) \\
= \text{sgn}(\sum_{\alpha_i^* > 0} y_i\alpha_i^* \langle x^i, x \rangle + b^*)
\]

Remember: \( A_i' = x^i \)
Linear Machine in Feature Space

Let $\phi : X \rightarrow F$ be a nonlinear map from the input space to some feature space.

The classifier will be in the form (Primal):

$$f(x) = \left( \sum_{i=1}^{?} w_i \phi_i(x) \right) + b$$

Make it in the dual form:

$$f(x) = \left( \sum_{i=1}^{l} \alpha_i y_i \langle \phi(x^i) \cdot \phi(x) \rangle \right) + b$$
Kernel: Represent Inner Product in Feature Space

Definition: A kernel is a function $K : X \times X \to R$ such that for all $x, z \in X$

$$K(x, z) = \langle \phi(x) \cdot \phi(z) \rangle$$

where $\phi : X \to F$

The classifier will become:

$$f(x) = \left( \sum_{i=1}^{l} \alpha_i y_i K(x^i, x) \right) + b$$
A Simple Example of Kernel

Polynomial Kernel of Degree 2: \( K(x, z) = \langle x, z \rangle^2 \)

Let \( x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \), \( z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \in \mathbb{R}^2 \) and the nonlinear map

\( \phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \) defined by

\( \phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{bmatrix} \).

Then \( \langle \phi(x), \phi(z) \rangle = \langle x, z \rangle^2 = K(x, z) \).

\( \triangleright \) There are many other nonlinear maps, \( \psi(x) \), that satisfy the relation: \( \langle \psi(x), \psi(z) \rangle = \langle x, z \rangle^2 = K(x, z) \)
Consider a nonlinear map $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^p$ that consists of distinct features of all the monomials of degree $d$.

Then $p = \binom{n + d - 1}{d}$.

For example: $n = 11, \ d = 10, \ p = 92378$

Is it necessary? We only need to know $\langle \phi(x), \phi(z) \rangle$!

This can be achieved $K(x, z) = \langle x, z \rangle^d$
More Examples of Kernel

\[ K(A, B) : R^{m \times n} \times R^{n \times l} \rightarrow R^{m \times l} \]

\( A \in R^{m \times n}, a \in R^m, \mu \in R, \ d \) is an integer:

- **Polynomial Kernel**: \((AA' + \mu aa')^d\)
  (Linear Kernel \(AA'\): \(\mu = 0, d = 1\))

- **Gaussian (Radial Basis) Kernel**: 
  \[ K(A, A')_{ij} = \varepsilon^{-\mu \|A_i - A_j\|^2_2}, \ i, j = 1, \ldots, m \]

- The \(ij\)-entry of \(K(A, A')\) represents the “similarity” of data points \(A_i\) and \(A_j\)
Kernel Technique
Based on Mercer’s Condition (1909)

- The value of kernel function represents the inner product of two training points in feature space.

- Kernel functions merge two steps
  1. map input data from input space to feature space (might be infinite dim.)
  2. do inner product in the feature space
Nonlinear SVM Motivation

- **Linear SVM:** (Linear separator: \( x'w + b = 0 \))

\[
\begin{align*}
\min_{\xi \geq 0, w, b} & \frac{C}{2} \| \xi \|_2^2 + \frac{1}{2} (\| w \|_2^2 + b^2) \\
\text{s. t.} & \quad D(Aw + eb) + \xi \geq e
\end{align*}
\]

By QP “duality”, \( w = A'D\alpha \). Maximizing the margin in the “dual space” gives:

\[
\begin{align*}
\min_{\xi \geq 0, \alpha, b} & \frac{C}{2} \| \xi \|_2^2 + \frac{1}{2} (\| \alpha \|_2^2 + b^2) \\
\text{s. t.} & \quad D(AA'D\alpha + eb) + \xi \geq e
\end{align*}
\]

- **Dual SSVM with separator:** \( x'A'D\alpha + b = 0 \)

\[
\begin{align*}
\min_{\alpha, b} & \frac{C}{2} \| p(e - D(AA'D\alpha + eb), \beta) \|_2^2 + \frac{1}{2} (\| \alpha \|_2^2 + b^2)
\end{align*}
\]
Nonlinear Smooth SVM

Nonlinear Classifier: \( K(x', A') D\alpha + b = 0 \)

- Replace \( AA' \) by a nonlinear kernel \( K(A, A') \):

\[
\min_{\alpha, b} \frac{C}{2}\|p(e - D(K(A, A')D\alpha + eb, \beta))\|^2_2 + \frac{1}{2}(\|\alpha\|^2_2 + b^2)
\]

- Use Newton-Armijo algorithm to solve the problem
  - Each iteration solves \( m+1 \) linear equations in \( m+1 \) variables

- Nonlinear classifier depends on entire dataset:

\[
K(x', A') D\alpha + b = 0
\]
Difficulties with Nonlinear SVM for Large Problems

- The nonlinear kernel $K(A, A') \in \mathbb{R}^{l \times l}$ is fully dense
  - Long CPU time to compute the dense kernel matrix
  - Runs out of memory while storing the kernel matrix
  - Need to generate and store $O(l^2)$ entries
- Computational complexity depends on # of example
  - Complexity of nonlinear SSVM $\sim O((l + 1)^3)$
- Separating surface depends on almost entire dataset
  - Need to store the entire dataset even after solving the problem
Solving the SVM with Massive Dataset

- Standard optimization techniques require that the data are held in memory
  - Limit the SVM to dataset of a few thousand points

- Solution I: SMO (Sequential Minimal Optimization)
  - Solve the sub-optimization problem defined by the working set (size = 2)
  - Increase the objective function iteratively

- Solution II: RSVM (Reduced Support Vector Machine)
Support Vector Regression

(Linear Case: \( f(x) = x'w + b \))

- Given the training set:
  \[ S = \{(x^i, y^i) | x^i \in \mathbb{R}^n, y^i \in \mathbb{R}, i = 1, \ldots, l\} \]

- Find a linear function, \( f(x) = x'w + b \) where \((w, b)\) is determined by solving a minimization problem that guarantees the smallest overall experiment error made by \( f(x) = x'w + b \)

- Motivated by SVM:
  - \( \|w\|_2 \) should be as small as possible
  - Some tiny error should be discard
**$\varepsilon$-Insensitive Loss Function**

- **$\varepsilon$-insensitive loss function:**
  
  $$|\xi|_\varepsilon = \max\{0, |\xi| - \varepsilon\} = \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases}$$

- If $\xi \in \mathbb{R}^n$ then $|\xi|_\varepsilon \in \mathbb{R}^n$ is defined as:
  
  $$(|\xi|_\varepsilon)_i = |\xi_i|_\varepsilon, \quad i = 1 \ldots n$$

- The loss made by the estimation function, $f$, at the data point $(x^i, y_i)$ is
  
  $$|y_i - f(x^i)|_\varepsilon = \max\{0, |y_i - f(x^i)| - \varepsilon\}$$
$\varepsilon$-Insensitive Linear Regression

Find $(w, b)$ with the smallest overall error

$$f(x) = x'w + b$$
\( \epsilon \text{- insensitive Support Vector Regression Model} \)

- **Motivated by SVM:**
  - \( \|w\|_2 \) should be as small as possible
  - Some tiny error should be discarded

\[
\min_{(w,b,\xi) \in \mathbb{R}^{n+1+m}} \frac{1}{2}\|w\|_2^2 + Ce' |\xi|_\epsilon
\]

where \( |\xi|_\epsilon \in \mathbb{R}^m, \quad (|\xi|_\epsilon)_i = \max(0, |A_iw + b - y_i| - \epsilon) \)
Reformulated $\epsilon$-SVR as a Constrained Minimization Problem

$$\min_{(w,b,\xi,\xi^*) \in \mathbb{R}^{n+1+2m}} \frac{1}{2}w'w + Ce'(\xi + \xi^*)$$

subject to

$$y - Aw - eb \leq ee + \xi$$
$$Aw + eb - y \leq ee + \xi^*$$

$$\xi, \xi^* \geq 0$$

$n+1+2m$ variables and $2m$ constrains minimization problem

Enlarge the problem size and computational complexity for solving the problem
Five Wild Used Loss Functions

(a) Quadratic

(b) Laplace

(c) Huber

(d) $\varepsilon$-insensitive

(e) Quadratic $\varepsilon$-insensitive
SV Regression by Minimizing Quadratic $\varepsilon$-Insensitive Loss

- We have the following (nonsmooth) problem:

$$\min_{(w,b,\xi) \in \mathbb{R}^{n+1+l}} \frac{1}{2}(\|w\|^2_2 + b^2) + \frac{C}{2} \|(|\xi|_\varepsilon)\|^2_2$$

where $\|\xi|_\varepsilon\}_i = |y_i - (w'x^i + b)|_\varepsilon$

- We minimize $\|(w, b)\|^2_2$ at the same time

  ➢ *Occam’s razor*: the simplest is the best

  ➢ Have the strong convexity of the problem
$\epsilon$ - insensitive Loss Function

\[ (-x - \epsilon)_+ \quad \left| x \right|_\epsilon \quad (x - \epsilon)_+ \]
Quadratic $\epsilon$-insensitive Loss Function

$$|x|^2_\epsilon = ((x - \epsilon)_+ + (-x - \epsilon)_+)^2$$

$$= (x - \epsilon)_+^2 + (-x - \epsilon)_+^2$$

$$(x - \epsilon)_+ \cdot (-x - \epsilon)_+ = 0$$
Use $p^2_\epsilon$-function replace

**Quadratic $\epsilon$-insensitive Function**

$$p^2_\epsilon(x, \beta) = (p(x - \epsilon, \beta))^2 + (p(-x - \epsilon, \beta))^2$$

which $p(x, \beta)$ is defined by

$$p(x, \beta) = x + \frac{1}{\beta} \log(1 + \exp^{-\beta x})$$

$p$-function with

$\alpha=10$, $p(x, 10)$, $x \in [-3, 3]$
$|x|^2 \frac{2}{\epsilon}$

$p^2_{\epsilon}(x, \beta), \quad \epsilon = 1, \quad \beta = 5$
ε - insensitive Smooth Support Vector Regression

\[
\min_{(w,b) \in \mathbb{R}^{n+1}} \Phi_{\epsilon,\alpha}(w, b) :=
\]

\[
\min_{(w,b) \in \mathbb{R}^{n+1}} \frac{1}{2}(w'w + b^2) + \frac{C}{2} \sum_{i=1}^{m} p_c^2(A_i w + b - y_{ii} |^2)
\]

This problem is a strongly convex minimization problem without any constrains.

The object function is twice differentiable thus we can use a fast Newton-Armijo method to solve this problem.
Nonlinear \( \epsilon \)-SVR

Based on duality theorem and KKT–optimality conditions

\[
\begin{align*}
    w &= A'\alpha, \quad \alpha \in \mathbb{R}^m \\
    y &\approx Aw + eb \\
    y &\approx AA'\alpha + eb
\end{align*}
\]

In nonlinear case:

\[
y \approx K(A, A')\alpha + eb
\]
Nonlinear $\epsilon - SVR$

$$\min_{(\alpha,b) \in \mathbb{R}^{m+1}} \frac{1}{2} ||\alpha||^2_2 + C \sum_{i=1}^{m} |K(A_i, A')\alpha + b - y_i|_\epsilon$$

Let $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times l}$

and $K(A, B) : \mathbb{R}^{m \times n} \times \mathbb{R}^{n \times l} \rightarrow \mathbb{R}^{m \times l}$

and $K(A_i, A') \in \mathbb{R}^{1 \times m}$

Nonlinear regression function : $f(x) = K(x, A')\alpha + b$
Nonlinear Smooth Support Vector
\( \epsilon \)-insensitive Regression

\[
\min_{(\alpha, b) \in \mathbb{R}^{m+1}} \frac{1}{2} (\alpha' \alpha + b^2)
\]

\[
+ \frac{C}{2} \sum_{i=1}^{m} p_{\epsilon}^2 K(K(A_i, A_{i}')) \alpha + bb - y_i y_i \alpha_i^2
\]
Numerical Results

- Training set and testing set (Slice method)
- **Gaussian kernel** is used to generate nonlinear $\varepsilon$-SVR in all experiments
- **Reduced kernel technique** is utilized when training dataset is bigger than 1000
- Error measure: 2-norm relative error
  \[
  \frac{\|y - \hat{y}\|_2}{\|y\|_2}
  \]
  $y$ : observations
  $\hat{y}$ : predicted values
101 Data Points in $R \times R$

Nonlinear SSVR with Kernel: $\exp^{-\mu\|x^i - x^j\|_2^2}$

$f(x) = 0.5 \ast sinc\left(\frac{10}{\pi}x\right) + \text{noise}$

$x \in [-1, 1], 101$ points

Noise: mean=0

$\sigma = 0.04$

Parameter:

$\nu = 50, \mu = 5, \varepsilon = 0.02$

Training time : 0.3 sec.
First Artificial Dataset

\[ f(x) = 0.5 \ast \frac{sinc\left(\frac{30}{\pi}x\right)}{\frac{30}{\pi}x} + \rho \]

\( \rho \) random noise with mean=0, standard deviation 0.04

\( \epsilon \) - SSVR
Training Time : 0.016 sec.
Error : 0.059

LIBSVM
Training Time : 0.015 sec.
Error : 0.068
Noise : mean=0 , \sigma = 0.4

Parameter : \nu = 50, \mu = 1, \varepsilon = 0.5

Mean Absolute Error (MAE) of 49x49 mesh points : 0.1761
Training time : 9.61 sec.
Using Reduced Kernel: $K(A, \overline{A}') \in R^{28900 \times 300}$

Noise: mean=0, $\sigma = 0.4$

Parameter: $C = 10000$, $\mu = 1$, $\epsilon = 0.2$

MAE of 49x49 mesh points: 0.0513

Training time: 22.58 sec.
## Real Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training/Testing size</th>
<th>Algorithm</th>
<th>Train Error</th>
<th>Test Error</th>
<th>CPU Time in second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Housing</td>
<td>$456 \times 13/ 50 \times 13$</td>
<td>$\epsilon$-SSVR, Conv. SVR</td>
<td>0.0907, 0.0777</td>
<td>0.1251, 0.1307</td>
<td>1.5, 3508.5</td>
</tr>
<tr>
<td>Comp-Activ</td>
<td>$300 \times 21/ 30 \times 21$</td>
<td>$\epsilon$-SSVR, Conv. SVR</td>
<td>0.0244, 0.2362</td>
<td>0.05, 0.059</td>
<td>0.13, 791</td>
</tr>
<tr>
<td></td>
<td>$1000 \times 21/ 100 \times 21$</td>
<td>$\epsilon$-SSVR, Conv. SVR</td>
<td>0.0278*, 0.037*</td>
<td>0.13, 791</td>
<td>6.1, &gt;22000</td>
</tr>
<tr>
<td></td>
<td>$7373 \times 21/ 819 \times 21$</td>
<td>$\epsilon$-SSVR, 5% Reduced Kernel</td>
<td>0.031</td>
<td>0.037</td>
<td>6.05</td>
</tr>
<tr>
<td>Kin-fh</td>
<td>$7373 \times 21/ 819 \times 21$</td>
<td>$\epsilon$-SSVR, 5% Reduced Kernel</td>
<td>0.1289</td>
<td>0.1343</td>
<td>5.2</td>
</tr>
</tbody>
</table>
### Linear $\epsilon$-SSVR

#### Tenfold Numerical Result

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Methods</th>
<th>$(C, \epsilon)$</th>
<th>#SVs</th>
<th>Train Error</th>
<th>Test Error</th>
<th>CPU Sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Housing</td>
<td>$\epsilon$-SSVR</td>
<td>$(16, 0.1)$</td>
<td>444</td>
<td>0.1934</td>
<td>0.1851</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>$(16, 0.1)$</td>
<td>447</td>
<td>0.2049</td>
<td>0.1855</td>
<td>0.047</td>
</tr>
<tr>
<td></td>
<td>$\text{SVM}^{light}$</td>
<td>$(16, 0.1)$</td>
<td>447</td>
<td>0.2049</td>
<td>0.1855</td>
<td>0.530</td>
</tr>
<tr>
<td>Comp-Activ</td>
<td>$\epsilon$-SSVR</td>
<td>$(1, 0.1)$</td>
<td>7114</td>
<td>0.0338</td>
<td>0.0347</td>
<td>0.110</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>$(1, 0.1)$</td>
<td>7048</td>
<td>0.0350</td>
<td>0.0360</td>
<td>16.656</td>
</tr>
<tr>
<td></td>
<td>$\text{SVM}^{light}$</td>
<td>$(1, 0.1)$</td>
<td>7047</td>
<td>0.0350</td>
<td>0.0360</td>
<td>17.020</td>
</tr>
<tr>
<td>Kin-fh</td>
<td>$\epsilon$-SSVR</td>
<td>$(1, 0.1)$</td>
<td>5316</td>
<td>0.1343</td>
<td>0.1365</td>
<td>0.188</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>$(1, 0.1)$</td>
<td>5323</td>
<td>0.1345</td>
<td>0.1371</td>
<td>39.547</td>
</tr>
<tr>
<td></td>
<td>$\text{SVM}^{light}$</td>
<td>$(1, 0.1)$</td>
<td>5321</td>
<td>0.1345</td>
<td>0.1370</td>
<td>39.740</td>
</tr>
</tbody>
</table>
## Nonlinear $\epsilon$-SSVR

Tenfold Numerical Result 1/2

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Methods</th>
<th>$(\mu, C, \epsilon)$</th>
<th>#SVs</th>
<th>Train Error</th>
<th>Test Error</th>
<th>CPU Sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Housing</td>
<td>$\epsilon$-SSVR</td>
<td>(1.1, 2750, 0.1)</td>
<td>442</td>
<td>0.0775</td>
<td>0.1171</td>
<td>0.895</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>(0.98, 90, 0.1)</td>
<td>438</td>
<td>0.0942</td>
<td>0.1168</td>
<td>0.400</td>
</tr>
<tr>
<td></td>
<td>$SVM_{light}$</td>
<td>(0.98, 90, 0.1)</td>
<td>438</td>
<td>0.0942</td>
<td>0.1168</td>
<td>4.908</td>
</tr>
<tr>
<td>Comp-Activ</td>
<td>$\epsilon$-SSVR</td>
<td>(0.08, 7250, 0.1)</td>
<td>868</td>
<td>0.0287</td>
<td>0.0300</td>
<td>3.461</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>(0.14, 100, 0.1)</td>
<td>859</td>
<td>0.0293</td>
<td>0.0307</td>
<td>0.502</td>
</tr>
<tr>
<td></td>
<td>$SVM_{light}$</td>
<td>(0.14, 100, 0.1)</td>
<td>858</td>
<td>0.0293</td>
<td>0.0307</td>
<td>2.881</td>
</tr>
<tr>
<td>Comp-Activ</td>
<td>$\epsilon$-SSVR</td>
<td>(0.3, 3800, 0.1)</td>
<td>7121</td>
<td>0.0296</td>
<td>0.0299</td>
<td>8.772</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>(0.3, 1000, 0.1)</td>
<td>6943</td>
<td>0.0271</td>
<td>0.0280</td>
<td>237.631</td>
</tr>
<tr>
<td></td>
<td>$SVM_{light}$</td>
<td>(0.3, 1000, 0.1)</td>
<td>6930</td>
<td>0.0271</td>
<td>0.0280</td>
<td>8624.250</td>
</tr>
</tbody>
</table>
Nonlinear $\varepsilon$-SSVR
Tenfold Numerical Result 2/2

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Methods</th>
<th>$(\mu, C, \varepsilon)$</th>
<th>#SVs</th>
<th>Train Error</th>
<th>Test Error</th>
<th>CPU Sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kin-fh</td>
<td>$\varepsilon$-SSVR</td>
<td>(0.05, 1, 0.1)</td>
<td>646</td>
<td>0.1308</td>
<td>0.1385</td>
<td>2.523</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>(0.05, 1, 0.1)</td>
<td>645</td>
<td>0.1090</td>
<td>0.1403</td>
<td>0.371</td>
</tr>
<tr>
<td></td>
<td>SVM$^{light}$</td>
<td>(0.05, 1, 0.1)</td>
<td>646</td>
<td>0.1090</td>
<td>0.1403</td>
<td>0.763</td>
</tr>
<tr>
<td>Kin-fh</td>
<td>$\varepsilon$-SSVR($363)$</td>
<td>(0.04, 101, 0.1)</td>
<td>5822</td>
<td>0.1284</td>
<td>0.1319</td>
<td>8.999</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>(0.02, 1, 0.1)</td>
<td>5280</td>
<td>0.1276</td>
<td>0.1322</td>
<td>24.692</td>
</tr>
<tr>
<td></td>
<td>SVM$^{light}$</td>
<td>(0.02, 1, 0.1)</td>
<td>5281</td>
<td>0.1276</td>
<td>0.1322</td>
<td>62.002</td>
</tr>
</tbody>
</table>
We introduced SVMs for classification and regression. SVMs can be extended from linear to nonlinear by using kernel trick. We applied smooth technique to SVMs to propose smooth SVMs for classification and regression. We also described the Newton-Armijo algorithm to solve smooth SVMs which has been shown convergent globally and quadratically in finite steps to the solution. The numerical results show the effectiveness and correctness of linear and nonlinear smooth SVMs. Our smooth SVMs formulation is only need to solve a system of linear equations iteratively instead of solving a convex quadratic problem.