Regression and Classification using Kernel Methods

Barnabás Póczos
University of Alberta

Oct 15, 2009
Roadmap I

Classification

• Primal Perceptron $\Rightarrow$ Dual Perceptron $\Rightarrow$ Kernels
  
  *(It doesn’t want large margin...)*

• Primal hard SVM $\Rightarrow$ Dual hard SVM $\Rightarrow$ Kernels
  
  *(It wants large margin, but assumes the data is linearly separable in the feature space)*

• Primal soft SVM $\Rightarrow$ Dual soft SVM $\Rightarrow$ Kernels
  
  *(It wants large margin, and doesn’t assume that the data is linearly separable in the feature space)*
Roadmap II

Regression

• Ridge Regression ⇒ Dual form ⇒ Kernels

• Primal SVM for Regression ⇒ Dual SVM ⇒ Kernels

• Logistic Regression ⇒ Kernels

• Bayesian Ridge Regression in feature space (Gaussian Processes) ⇒ Dual form ⇒ Kernels
The Perceptron
The primal algorithm in the feature space

Algorithm 1 Perceptron learning algorithm (in primal variables).

Require: A feature mapping $\phi : \mathcal{X} \rightarrow \mathcal{K} \subseteq \ell_2^n$
Ensure: A linearly separable training sample $z = ((x_1, y_1), \ldots, (x_m, y_m))$

$w_0 = 0; t = 0$

repeat
  for $j = 1, \ldots, m$ do
    if $y_j \langle \phi(x_j), w \rangle \leq 0$ then
      $w_{t+1} = w_t + y_j \phi(x_j)$
      $t \leftarrow t + 1$
    end if
  end for

until no mistakes have been made within the for loop

return the final weight vector $w_t$

Testing: $f_{w_t}(x) = \langle w_t, \phi(x) \rangle$

If $x_j$ is misclassified

explicit features!
The Dual Perceptron

Algorithm 2 Perceptron learning algorithm (in dual variables).

Require: A feature mapping $\phi: \mathcal{X} \rightarrow \mathcal{K} \subseteq \ell^2$

Ensure: A linearly separable training sample $z = ((x_1, y_1), \ldots, (x_m, y_m))$

$\alpha = 0$

repeat

for $j = 1, \ldots, m$ do

if $y_j \sum_{i=1}^{m} \alpha_i \langle \phi(x_i), \phi(x_j) \rangle \leq 0$ then

$\alpha_j \leftarrow \alpha_j + y_j$

end if

end for

until no mistakes have been made within the for loop

return the vector $\alpha$ of expansion coefficients

$\hat{w} = \sum_{i=1}^{m} \alpha_i \phi(x_i)$

Testing: $f_{\hat{w}}(x) = \langle \hat{w}, \phi(x) \rangle = \sum_{i=1}^{m} \alpha_i \langle \phi(x_i), \phi(x) \rangle \frac{k(x_i, x)}{k(x_i, x)}$

Picture is taken from R. Herbrich
The SVM
Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x + b) \]

- denotes +1
- denotes −1

How to classify this data?
Each of these seems fine...
... which is best?

taken from Andrew W. Moore
Classifier Margin

- denotes +1
- denotes −1

The margin of a linear classifier = the width that the boundary could be increased by, before hitting a datapoint

\[ f(x, w, b) = \text{sign}(w \cdot x + b) \]

taken from Andrew W. Moore
Maximum Margin

- denotes +1
- denotes −1

Support Vectors are datapoints that “touch” the margin

\[ f(x, w, b) = \text{sign}(w \cdot x + b) \]

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

- the simplest kind of SVM – an LSVM

taken from Andrew W. Moore
Specifying a Line and a Margin

• **Plus-plane**  =  \{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}
• **Minus-plane**  =  \{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}

Classify as:

- **+1**  if  \( \mathbf{w} \cdot \mathbf{x} + b \geq 1 \)
- **-1**  if  \( \mathbf{w} \cdot \mathbf{x} + b \leq -1 \)

- **Universe explodes**  if  \(-1 < \mathbf{w} \cdot \mathbf{x} + b < 1\)

taken from Andrew W. Moore
Computing the margin width

Given ...

- \( \mathbf{w} \cdot \mathbf{x}^+ + b = +1 \)
- \( \mathbf{w} \cdot \mathbf{x}^- + b = -1 \)
- \( \mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w} \)
- \( |\mathbf{x}^+ - \mathbf{x}^-| = M \)
- \( \lambda = \frac{2}{\mathbf{w} \cdot \mathbf{w}} \)

\[ M = \text{Margin Width} = \frac{2}{\sqrt{\mathbf{w} \cdot \mathbf{w}}} \]

\[ M = |\mathbf{x}^+ - \mathbf{x}^-| = |\lambda \mathbf{w}| = \lambda |\mathbf{w}| = \lambda \sqrt{\mathbf{w} \cdot \mathbf{w}} \]

\[ = \frac{2 \sqrt{\mathbf{w} \cdot \mathbf{w}}}{\mathbf{w} \cdot \mathbf{w}} = \frac{2}{\sqrt{\mathbf{w} \cdot \mathbf{w}}} \]

Yay! Just maximize \( \frac{2}{\sqrt{\mathbf{w} \cdot \mathbf{w}}} \)

\( ...\equiv \text{minimize } \mathbf{w} \cdot \mathbf{w} \)

Wait...OMG, I forgot the data!

taken from Andrew W. Moore
The Primal Hard SVM

- Given $D = \{(x_i, y_i), i = 1, \ldots, m\}$ training data set.
- $x_i = \phi(x_i) \in \mathcal{K} \subset \mathbb{R}^n$ explicit feature map.
- Assume that $D$ is **linearly separable** in the feature space $\mathcal{K}$.

\[ \hat{w} = \arg \min_{w \in \mathcal{K}} \frac{1}{2} \|w\|^2 \]

subject to \( y_i \langle \phi(x_i), w \rangle \geq 1, \forall i = 1, \ldots, m \)

The computational effort is $O(n^2)$

**This is a QP problem** $\Rightarrow$ solution is expressible in dual form

**Testing:** $f_{\hat{w}}(x) = \langle \hat{w}, \phi(x) \rangle$
Quadratic Programming – in general

Find \( \arg \min_w \ c + d^T w + \frac{w^T K w}{2} \)

Subject to
\[
\begin{align*}
    a_{11} w_1 + a_{12} w_2 + \ldots + a_{1m} w_m & \leq b_1 \\
    a_{21} w_1 + a_{22} w_2 + \ldots + a_{2m} w_m & \leq b_2 \\
    \vdots & \\
    a_{n1} w_1 + a_{n2} w_2 + \ldots + a_{nm} w_m & \leq b_n
\end{align*}
\]

and to
\[
\begin{align*}
    a_{(n+1)1} w_1 + a_{(n+1)2} w_2 + \ldots + a_{(n+1)m} w_m & = b_{(n+1)} \\
    a_{(n+2)1} w_1 + a_{(n+2)2} w_2 + \ldots + a_{(n+2)m} w_m & = b_{(n+2)} \\
    \vdots & \\
    a_{(n+e)1} w_1 + a_{(n+e)2} w_2 + \ldots + a_{(n+e)m} w_m & = b_{(n+e)}
\end{align*}
\]

Note \( w \cdot x = w^T x \)

\( n \) additional linear inequality constraints

\( e \) additional linear equality constraints

taken from Andrew W. Moore
The Dual Hard SVM

\[ Y \doteq \text{diag}(y_1, \ldots, y_m), \quad y_i \in \{-1, 1\}^m \]

\[ G \in \mathbb{R}^{m \times m} \doteq \{G_{ij}\}_{i,j}^{m,m}, \quad \text{where } G_{ij} \doteq \left< \frac{\phi(x_i)}{k(x_i, x_j)}, \frac{\phi(x_j)}{k(x_j, x_i)} \right> \mathcal{K}, \quad \text{Gram matrix.} \]

\[ \hat{\alpha} = \arg \max_{\alpha \in \mathbb{R}^m} \alpha^T 1_m - \frac{1}{2} \alpha^T Y G Y \alpha \]

subject to \( \alpha_i \geq 0, \quad \forall i = 1, \ldots, m \)

Quadratic Programming, the computational effort is \( \mathcal{O}(m^2) \)

Lemma \( \hat{w} = \sum_{i=1}^{m} \hat{\alpha}_i y_i \phi(x_i) \)

Testing: \( f_{\hat{w}}(x) = \langle \hat{w}, \phi(x) \rangle = \sum_{i=1}^{m} \hat{\alpha}_i y_i \langle \phi(x_i), \phi(x) \rangle \frac{k(x_i, x)}{k(x_j, x_j)} \)
Proof

Primal Lagrangian:

\[ \alpha = (\alpha_1, \ldots, \alpha_m)^T \geq 0 \] Lagrange multipliers

\[ L(w, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left( y_i \langle \phi(x_i), w \rangle - 1 \right) \]

The solution: \( (\hat{w}, \hat{\alpha}) = \arg \min_{w \in \mathcal{K}} \max_{0 \leq \alpha} L(w, \alpha) \)

\[
0 = \left. \frac{\partial L(w, \alpha)}{\partial w} \right|_{w = \hat{w}} = \hat{w} - \sum_{i=1}^{m} \alpha_i y_i \phi(x_i)
\]

\[ \Rightarrow \hat{w} = \sum_{i=1}^{m} \alpha_i y_i \phi(x_i) \]

\[ \Rightarrow L(\hat{w}, \alpha) = \frac{1}{2}\|\hat{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left( y_i \langle \phi(x_i), \hat{w} \rangle - 1 \right) \]
Proof cont.

\[ L(\hat{w}, \alpha) = \frac{1}{2} \|\hat{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left( y_i \langle \phi(x_i), \hat{w} \rangle - 1 \right) \]

\[ = \frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i \phi(x_i) \right\|^2 + \alpha^T 1_m - \sum_{i=1}^{m} \alpha_i y_i \langle \phi(x_i), \sum_{j=1}^{m} \alpha_j y_j \phi(x_j) \rangle \]

\[ = \alpha^T 1_m - \frac{1}{2} \alpha^T Y G Y \alpha \]
The Problem with Hard SVM

It assumes samples are linearly separable...

Solutions:

• Use kernel with **large expressive power**
  (e.g. RBF with small $\sigma$)
  $\Rightarrow$ each training samples are linearly separable
  in the feature space
  $\Rightarrow$ Hard SVM can be applied 😊
  $\Rightarrow$ overfitting... 😞

• **Soft margin** SVM instead of Hard SVM
  • Slack variables...
The Hard SVM problem can be rewritten:

\[
\hat{w}_{hard} = \arg\min_{w \in \mathcal{K}} \frac{1}{2} \|w\|^2 \\
\text{subject to } y_i \langle \phi(x_i), w \rangle \geq 1, \ \forall i = 1, \ldots, m
\]

where

\[
l_{0-\infty}(a, b) \doteq \begin{cases} 
\infty : ab < 0 & \text{Misclassification} \\
0 : ab > 0 & \text{Correct classification}
\end{cases}
\]
From Hard to Soft constraints

Instead of using hard constraints (points are linearly separable)

\[
\hat{w}_{hard} = \arg \min_{w \in \mathcal{K}} \sum_{i=1}^{m} l_{0-\infty}(\langle \phi(x_i), w \rangle, y_i) + \frac{\lambda}{2} \|w\|^2
\]

We can try solve the soft version of it

\[
\hat{w}_{soft} = \arg \min_{w \in \mathcal{K}} \sum_{i=1}^{m} l_{0-1}(\langle \phi(x_i), w \rangle, y_i) + \frac{\lambda}{2} \|w\|^2
\]

where

\[
l_{0-1}(y, f(x)) = \begin{cases} 
1 : yf(x) < 0 & \text{Misclassification} \\
0 : yf(x) > 0 & \text{Correct classification}
\end{cases}
\]
Problems with $l_{0-1}$ loss

It is not convex in $yf(x) \Rightarrow$ It is not convex in $w$, either...
... and we like only convex functions...

Let us approximate it with convex functions!

- Piecewise linear approximations (hinge loss, $l_{\text{lin}}$)

$$l_{\text{lin}}(f(x), y) = \max\{1 - yf(x), 0\}$$

[We want $yf(x) > 1$]

- Quadratic approximation ($l_{\text{quad}}$)

$$l_{\text{quad}}(f(x), y) = \max\{1 - yf(x), 0\}^2$$
Approximation of the Heaviside step function

\[ l_{0-1}(y, \langle w, \phi(x) \rangle) = \begin{cases} 
1 : y \langle w, \phi(x) \rangle < 0 \\
0 : y \langle w, \phi(x) \rangle \geq 0 
\end{cases} \]

\[ l_{\text{quad}}(y, f(x)) \]
\[ l_{\text{lin}}(y, f(x)) \]

Picture is taken from R. Herbrich
The hinge loss approximation of $l_{0-1}$

\[ \hat{w} = \arg \min_{w \in \mathcal{K}} \sum_{i=1}^{m} l_{lin}(\langle \phi(x_i), w \rangle, y_i) + \frac{\lambda}{2} \|w\|^2 \]

Where,

\[ \xi_i = l_{lin}(f(x_i), y_i) = \max\{1 - y_i f(x_i), 0\} \geq 1 - y_i \langle w, \phi(x_i) \rangle \geq l_{0-1}(y_i, f(x_i)) \]

$\xi_i$: Slack variables
The Slack Variables

$wx + b = 0$

$x_1$

$x_2$

$x_7$

$M = \sqrt{\frac{2}{W \cdot W}}$

taken from Andrew W. Moore
The Primal Soft SVM problem

Using lin (hinge) loss approximation:

\[
\widehat{w}_{soft} = \arg \min_{w \in \mathcal{K}, \xi \in \mathbb{R}^m} \sum_{i=1}^{m} \xi_i + \frac{\lambda}{2} \|w\|^2 \\
\text{subject to } y_i \langle \phi(x_i), w \rangle \geq 1 - \xi_i, \forall i = 1, \ldots, m \\\n\xi_i \geq 0, \forall i = 1, \ldots, m
\]

Or we can use this form, too...

where \( C = \frac{1}{\lambda} \)

\[
\widehat{w}_{soft} = \arg \min_{w \in \mathcal{K}, \xi \in \mathbb{R}^m} C \sum_{i=1}^{m} \xi_i + \frac{1}{2} \|w\|^2 \\
\xi^T 1_m
\]
The Dual Soft SVM (using hinge loss)

\[ Y = \text{diag}(y_1, \ldots, y_m) \in \{-1, 1\}^m \]

\[ G \in \mathbb{R}^{m \times m} = \{G_{ij}\}_{i,j}^{m \times m}, \text{ where } G_{ij} = \frac{k(x_i, x_j)}{\phi(x_i) \phi(x_j)} \text{, Gram matrix.} \]

\[ \hat{\alpha} = \arg \max_{\alpha \in \mathbb{R}^m} \alpha^T 1_m - \frac{1}{2} \alpha^T Y G Y \alpha \]

subject to \( 0 \leq \alpha_i \leq C \)

where \( C = \frac{1}{\lambda} \)

If \( \lambda \to 0 \Rightarrow \text{soft-SVM} \to \text{hard-SVM} \)

This is the same as the dual hard-SVM problem, but now we have the additional \( 0 \leq \alpha_i \leq C \) constraints.
Proofs

\[ \alpha = (\alpha_1, \ldots, \alpha_m)^T \geq 0 \text{ Lagrange multipliers} \]

\[ \beta = (\beta_1, \ldots, \beta_m)^T \geq 0 \text{ Lagrange multipliers} \]

\[ \xi = (\xi_1, \ldots, \xi_m)^T \geq 0 \text{ Slack variables} \]

\[
\begin{aligned}
(\hat{w}, \hat{\xi}, \hat{\alpha}, \hat{\beta}) &= \arg \min_{\mathbf{w} \in \mathcal{K}} \max_{0 \leq \alpha} \max_{0 \leq \xi \in \mathbb{R}^m} \max_{0 \leq \beta} L(\mathbf{w}, \xi, \alpha, \beta) \\
&= \arg \min_{\mathbf{w} \in \mathcal{K}} \max_{0 \leq \alpha} \max_{0 \leq \xi \in \mathbb{R}^m} \max_{0 \leq \beta} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i \hat{\xi}^T \mathbf{1}_m
\end{aligned}
\]

where

\[
L(\mathbf{w}, \xi, \alpha, \beta) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i \xi^T \mathbf{1}_m
\]

\[
- \sum_{i=1}^{m} \alpha_i (y_i \langle \phi(x_i), \mathbf{w} \rangle - 1 + \xi_i)^T - \sum_{i=1}^{m} \beta_i \xi_i
\]
Proofs

\[ L(w, \xi, \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \xi^T 1_m - \sum_{i=1}^{m} \alpha_i y_i \langle \phi(x_i), w \rangle + \alpha^T 1_m - \xi^T (\alpha + \beta) \]

\[ 0 = \left. \frac{\partial L(w, \xi, \alpha, \beta)}{\partial w} \right|_{w = \hat{w}} = \hat{w} - \sum_{i=1}^{m} \alpha_i y_i \phi(x_i) \Rightarrow \hat{w} = \sum_{i=1}^{m} \alpha_i y_i \phi(x_i) \]

\[ 0 = \left. \frac{\partial L(w, \xi, \alpha, \beta)}{\partial \xi} \right|_{\xi = \hat{\xi}} = C 1_m - \alpha - \beta \Rightarrow \beta = C 1_m - \alpha \geq 0 \]
\[ \Rightarrow 0 \leq \alpha \leq C \]

\[ \Rightarrow (\hat{\alpha}, \hat{\beta}) = \text{arg} \max_{0 \leq \alpha} L(\hat{w}, \hat{\xi}, \alpha, \beta) \]
\[ \Rightarrow \hat{\alpha} = \text{arg} \max_{0 \leq \alpha \leq C} \alpha^T 1_m - \frac{1}{2} \alpha^T YGY \alpha \]
Classifying using SVMs with Kernels

- Choose a kernel function
- Solve the dual problem

\[ \hat{\alpha} = \arg \max_{\alpha \in \mathbb{R}^m} \alpha^T 1_m - \frac{1}{2} \alpha^T \mathbf{Y} \mathbf{G} \mathbf{Y} \alpha \]

subject to \( 0 \leq \alpha_i \leq C \)

where \( C = \frac{1}{\lambda} \). Let \( \hat{\mathbf{w}} = \sum_{i=1}^{m} \hat{\alpha}_i y_i \phi(x_i) \).

On test data \( x \): \( f_{\hat{\mathbf{w}}}(x) = \langle \hat{\mathbf{w}}, \phi(x) \rangle = \sum_{i=1}^{m} \hat{\alpha}_i y_i \frac{\langle \phi(x_i), \phi(x) \rangle}{k(x_i, x)} \)
A few results
Kernels and Linear Classifiers

Let $\vec{x} = [x_1, x_2] \in \mathbb{R}^2$ be a vectorial representation of object $x \in \mathcal{X}$.

Let $\phi : \mathcal{X} \rightarrow \mathcal{K} \subset \mathbb{R}^3$ feature map be given by

$$\phi(\vec{x}) = [x_1, x_1^2, x_1 x_2]^T \in \mathcal{K} \subset \mathbb{R}^3$$

**Def.** Feature space: $\mathcal{K}$

**We will use linear classifiers in this feature space.**

In the original space $\mathbb{R}^2$ for a given $w \in \mathbb{R}^3$ the decision surface is:

$$\mathcal{X}_0(w) = \{ \vec{x} \in \mathbb{R}^2 \mid w_1 x_1 + w_2 x_2^2 + w_3 x_1 x_2 = 0 \}$$

- This is nonlinear in $\vec{x} \in \mathbb{R}^2$
- This is linear in the feature space $\phi(\vec{x}) \in \mathcal{K} \subset \mathbb{R}^3$
The Decision Surface

\[ \{ \vec{x} \in \mathbb{R}^2 \mid w_1 \vec{x}_1 + w_2 \vec{x}_2^2 + w_3 \vec{x}_1 \vec{x}_2 = 0 \} \]

decision surface for fixed \( w \) vectors.

\[ \{ \vec{x} \in \mathbb{R}^2 \mid w_1 \vec{x}_1 + w_2 \vec{x}_2^2 + w_3 \vec{x}_1 \vec{x}_2 = \pm 1 \} \]

margin for fixed \( w \) vectors.

For general feature maps it can be very difficult to solve these...

Picture is taken from R. Herbrich
Steve Gunn’s svm toolbox
Results, Iris 2vs13, Linear kernel
Results, Iris 1vs23, 2\textsuperscript{nd} order kernel

No. of Support Vectors: 12 (10.0\%)
Results, Iris 1vs23, 2nd order kernel
Results, Iris 1vs23, 13th order kernel

No. of Support Vectors: 12 (10.0%)
Results, Iris 1vs23, RBF kernel
Results, Iris 1vs23, RBF kernel
Results, Chessboard, Poly kernel

Polynomial
Degree 3
Separable
Bound 1

No. of Support Vectors: 263 (87.7%)
Results, Chessboard, Poly kernel

Polynomial

Degree

Separable

Bound

1

No. of Support Vectors: 183 (61.0%)
Results, Chessboard, Poly kernel

Polynomial
Degree 9
Separable
Bound 1

No. of Support Vectors: 164 (54.7%)
Results, Chessboard, Poly kernel

No. of Support Vectors: 147 (49.0%)
Results, Chessboard, poly kernel

Polynomial

Degree 13

Separable

Bound 1

No. of Support Vectors: 102 (34.0%)
Results, Chessboard, RBF kernel

No. of Support Vectors: 174 (58.0%)
Regression
Ridge Regression

Linear regression: \( f(x) = \langle w, \phi(x) \rangle \)

Primal:

\[
\hat{w} = \arg \min_{w \in \mathcal{K}} \sum_{i=1}^{m} \xi_i^2 \\
\text{subject to } y_i - \langle \phi(x_i), w \rangle = \xi_i, \forall i = 1, \ldots, m \\
\text{and } \|w\| \leq B
\]

\[
L(w, \xi, \alpha, \beta) = \sum_{i=1}^{m} \xi_i^2 + \sum_{i=1}^{m} \alpha_i (y_i - \langle \phi(x_i), w \rangle - \xi_i) + \lambda (\|w\|^2 - B^2) \\
\lambda > 0
\]

Dual for a given \( \lambda \): ...after some calculations...

\[
\hat{\alpha} = \arg \max_{\alpha \in \mathbb{R}^m} \lambda \sum_{i=1}^{m} \alpha_i^2 - 2 \sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j k(x_i, x_j)
\]

This can be solved in closed form:
Kernel Ridge Regression Algorithm

Given \( D = \{(x_i, y_i), i = 1, \ldots, m\} \) training data set, \( k(\cdot, \cdot) \) kernel, \( \lambda > 0 \) parameter. \( y = (y_1, \ldots, y_m)^T \in \{-1, 1\}^m \)

- \( G \in \mathbb{R}^{m \times m} \equiv \{G_{ij}\}_{i,j}^{m,m} \), \( k(x_i, x_j) \)
  where \( G_{ij} \equiv \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{K}} \), Gram matrix.

- \( \hat{\alpha} = (G + \lambda I_m)^{-1} y \)

- \( \hat{\mathbf{w}} = \sum_{i=1}^{m} \hat{\alpha}_i \phi(x_i) \).

- \( f(x) = \langle \hat{\mathbf{w}}, \phi(x) \rangle = \sum_{i=1}^{m} \hat{\alpha}_i k(x_i, x) \)

hmmm... We haven't optimized in \( \lambda \)...and where is \( B \)???

Different values of \( \lambda \) correspond to different choice of \( B \)
SVM Regression

• Typical loss function: $C \sum_n (y_n - t_n)^2 + \frac{1}{2} \|w\|^2$
  (ridge regression)
  ...
  quadratic penalty whenever $y_n \neq t_n$

• To be sparse... don’t worry if “close enough”

  • $E_{\epsilon}(y, t) = \begin{cases} 0 & \text{if } |y - t| < \epsilon \\ |y - t| - \epsilon & \text{otherwise} \end{cases}$

• ... $\epsilon$ insensitive loss function

  • $C \sum_n E_{\epsilon}(y_n, t_n) + \frac{1}{2} \|w\|^2$

• No penalty if in $\epsilon$-tube

taken from Andrew W. Moore
SVM Regression

\[ C \sum_n E_\varepsilon (y_n, t_n) + \frac{1}{2} \|w\|^2 \]

\[ \text{E}_\varepsilon(y, t) = \begin{cases} 
0 & \text{if } |y - t| < \varepsilon \\
|y - t| - \varepsilon & \text{otherwise}
\end{cases} \]

- No penalty if \( y_n - \varepsilon \leq t_n \leq y_n + \varepsilon \)
- Slack variables: \( \{\xi_{n+}, \xi_{n-}\} \)
  - \( t_n \leq y_n + \varepsilon + \xi_{n+} \)
  - \( t_n \geq y_n - \varepsilon - \xi_{n-} \)
- Error function:
  \[ C \sum_n (\xi_{n+} + \xi_{n-}) + \frac{1}{2} \|w\|^2 \]

- ... use Lagrange Multipliers \( \{a_{n+}, a_{n-}, \mu_{n+}, \mu_{n-}\} \geq 0 \)
  \[ \min L(...) = C \sum_n (\xi_{n+} + \xi_{n-}) + \frac{1}{2} \|w\|^2 - \sum_n (\mu_{n+} \xi_{n+} + \mu_{n-} \xi_{n-}) \]
  \[ - \sum_n a_{n+} (\varepsilon + \xi_{n+} + y_n - t_n) - \sum_n a_{n-} (\varepsilon + \xi_{n-} - y_n + t_n) \]

taken from Andrew W. Moore
SVM Regression, con’t

\[ L(...) = C \sum_n (\xi_{n+} + \xi_{n-}) + \frac{1}{2} \|w\|^2 - \sum_n (\mu_{n+} \xi_{n+} + \mu_{n-} \xi_{n-}) + \sum_n a_{n+} (\varepsilon + \xi_{n+} + y_n - t_n) - \sum_n a_{n-} (\varepsilon + \xi_{n-} - y_n + t_n) \]

- Set derivatives to 0, solve for \( \{\xi_{n+}, \xi_{n-}, \mu_{n+}, \mu_{n-}\} \) ...

\[ \min_{\tilde{a}_+, \tilde{a}_-} \tilde{L}(\tilde{a}_+, \tilde{a}_-) = \frac{1}{2} \sum_n \sum_m (a_{n+} - a_{n-})(a_{m+} - a_{m-})k(x_n, x_m) - \varepsilon \sum_n (a_{n+} - a_{n-}) + \sum_n (a_{n+} - a_{n-})t_n \]

s.t. \( 0 \leq a_{n+} \leq C \quad 0 \leq a_{n-} \leq C \)

- Prediction for new \( \mathbf{x} \):

\[ y(x) = \sum_n (a_{n+} - a_{n-}) k(x_n, x) \]

taken from Andrew W. Moore
SVM Regression, con’t

\[ y(x) = \sum_{n} (a_{n+} - a_{n-}) k(x_n, x) \]

- Can ignore \( x_n \) unless either \( a_{n+} > 0 \) or \( a_{n-} > 0 \)
  - \( a_{n+} > 0 \) only if \( t_n = y_n + \varepsilon + \xi_{n+} \)
    - ie, if on upper boundary of \( \varepsilon \)-tube (\( \xi_{n+} = 0 \))
    - or above (\( \xi_{n+} > 0 \))
  - \( a_{n-} > 0 \) only if \( t_n = y_n - \varepsilon - \xi_{n-} \)
    - ie, if on lower boundary of \( \varepsilon \)-tube (\( \xi_{n-} = 0 \))
    - or below (\( \xi_{n-} > 0 \))

Taken from Andrew W. Moore
Kernels in Logistic Regression

\[ P(Y = 1 \mid x, w) = \frac{1}{1 + e^{-(w \cdot \Phi(x) + b)}} \]

• Define weights in terms of support vectors:

\[ w = \sum_i \alpha_i \Phi(x_i) \]

\[ P(Y = 1 \mid x, w) = \frac{1}{1 + e^{-(\sum_i \alpha_i \Phi(x_i) \cdot \Phi(x) + b)}} = \frac{1}{1 + e^{-(\sum_i \alpha_i K(x, x_i) + b)}} \]

• Derive simple gradient descent rule on \( \alpha_i \)
taken from Andrew W. Moore
A few results
Sinc = \frac{\sin(\pi x)}{\pi x}, RBF kernel
$\text{Sinc} = \frac{\sin(\pi x)}{(\pi x)}$, RBF kernel

No. of Support Vectors: 31 (60.8%)
Sinc = \frac{\sin(\pi x)}{\pi x}, RBF kernel
Sinc = \frac{\sin(\pi x)}{(\pi x)}, RBF kernel

No. of Support Vectors: 51 (100.0\%)
Sinc = \frac{\sin(\pi x)}{\pi x}, RBF kernel

Gaussian RBF

Sigma: 0.01

Bound

Inf

\epsilon\text{ insensitivity: 0}

No. of Support Vectors: 51 (100.0%)
Sinc = \frac{\sin(\pi x)}{\pi x}, RBF kernel
Sinc = sin(\(\pi\ x\)) / (\(\pi\ x\)), poly kernel
Sinc = \frac{\sin(\pi x)}{\pi x}, poly kernel
Sinc = \frac{\sin(\pi x)}{\pi x}, poly kernel
Sinc = \sin(\pi x) / (\pi x), poly kernel

Polynomial

Degree: 33

Bound: 10

E insensitivity: 0.1

No. of Support Vectors: 15 (29.4%)
Sinc = \frac{\sin(\pi x)}{\pi x}, \text{ poly kernel}
Sinc = sin(\(\pi x\)) / (\(\pi x\)), poly kernel
Summary
Key SVM Ideas

- Maximize the **margin** between + and − examples
  - connects to PAC theory
- Sparse: Only the **support vectors** contribute to solution
- Penalize errors in non-separable case
- **Kernels** map examples into a new, usually nonlinear space
  - Implicitly do dot products in this new space (in the “dual” form of the SVM program)

- Kernels are separate from SVMs
  ... but they combine very nicely with SVMs
Summary I

Advantages

• Systematic implementation through quadratic programming
  \forall \exists very efficient implementations
• Excellent data-dependent generalization bounds
• Regularization built into cost function
• Statistical performance is independent of dim. of feature space

• Theoretically related to widely studied fields of regularization theory and sparse approximation
• Fully adaptive procedures available for determining hyper-parameters

taken from Andrew W. Moore
Summary II

Drawbacks

• Treatment of non-separable case somewhat heuristic
• Number of support vectors may depend strongly on the kernel type and the hyper-parameters
• Systematic choice of kernels is difficult (prior information)
  • ... some ideas exist
• Optimization may require clever heuristics for large problems
What You Should Know

• Definition of a maximum margin classifier
• Sparse version: (Linear) SVMs
• What QP can do for you (but you don’t need to know how it does it)
• How Maximum Margin = a QP problem
• How to deal with noisy (non-separable) data
  • Slack variable
• How to permit “non-linear boundaries”
  • Kernel trick
• How SVM Kernel functions permit us to pretend we’re working with a zillion features

taken from Andrew W. Moore
Thanks for the Attention! 😊
Attic
## Difference between SVMs and Logistic Regression

<table>
<thead>
<tr>
<th></th>
<th>SVMs</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loss function</strong></td>
<td>Hinge loss</td>
<td>Log-loss</td>
</tr>
<tr>
<td><strong>High dimensional features with kernels</strong></td>
<td>Yes!</td>
<td>No</td>
</tr>
<tr>
<td><strong>Solution sparse</strong></td>
<td>Often yes!</td>
<td>Almost always no!</td>
</tr>
<tr>
<td><strong>Semantics of output</strong></td>
<td>“Margin”</td>
<td>Real probabilities</td>
</tr>
</tbody>
</table>