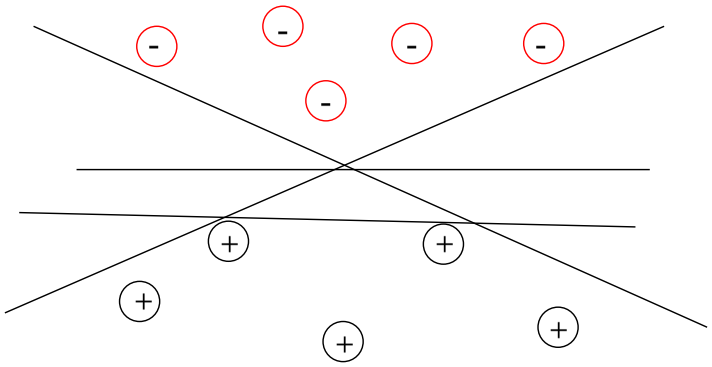
Lecture 18: Support Vector Machines (SVMs) - Part II

- Reminder: Perceptron and linear classification
- Optimal separating hyperplane
- Computing the optimal separating hyperplane
- Performance of SVMs
- SVMs for regression

The generalization problem

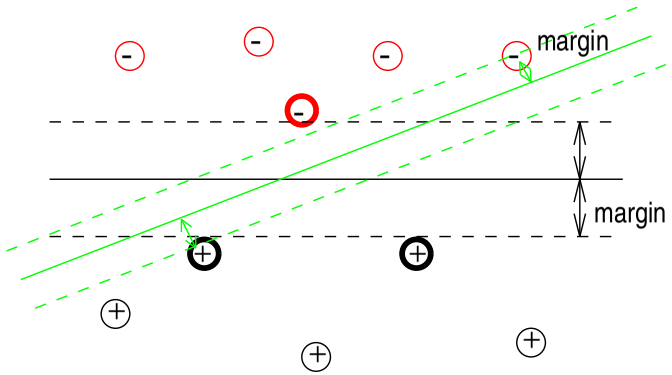
- The curse of dimensionality: it is easy to overfit in high-dimensional spaces
- There exists an infinite number of hyperplanes that separate the data!



- We need a principled way to choose the best possible hyperplane!

Recall: Optimal Separating Hyperplane

Among all the hyperplanes, there is one with the maximum margin to the training examples: *optimal separating hyperplane*



This is uniquely determined by the vectors closest to it, called *support vectors*

Specifying a line and a margin

- We will put in a hyperplane separating the data, and two planes above and below: plus-plane and minus-plane
- Let the hyperplane be given by: $\mathbf{w} \cdot \mathbf{x} + b$.
- Since it separates the data, by assumption we have:

$$(\mathbf{w} \cdot \mathbf{x_i}) + b \geq 1 \text{ if } y_i = 1$$

$$(\mathbf{w} \cdot \mathbf{x_i}) + b \leq -1 \text{ if } y_i = -1$$

- Let plus-plane be: $\mathbf{w} \cdot \mathbf{x_i}) + b = 1$ and minus-plane be: $\mathbf{w} \cdot \mathbf{x_i}) + b = 1$

Computing the width of the margin

- How do we compute M in terms of $\mathbf{w}$ and b?
- Claim: $\mathbf{w}$ is orthogonal to the plus-plane (and the minus-plane). Why?

  Let $\mathbf{u}$ and $\mathbf{v}$ be two vectors on the plus-plane. What is $\mathbf{w} \cdot (\mathbf{u} - \mathbf{v})$?
- Let $\mathbf{x}^-$ be a point on the minus-plane and let $\mathbf{x}^+$ be the point on the plus-plane closest to $\mathbf{x}^-$.

  Claim: $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$. Why?
- We know that $M = |x^+ - x^-|$.
- Can we compute the margin now?

Computing the width of the margin

We have:

$$\mathbf{w} \cdot \mathbf{x}^{+}) + b = 1 \Longrightarrow \mathbf{w} \cdot \mathbf{x}^{-} + \lambda \mathbf{w}) + b = 1$$

$$\Longrightarrow \mathbf{w} \cdot \mathbf{x}^{-} + b + \lambda \mathbf{w} \cdot \mathbf{w} = 1$$

$$\Longrightarrow -1 + \lambda \mathbf{w} \cdot \mathbf{w} = 1 \Longrightarrow \lambda = \frac{1}{\mathbf{w} \cdot \mathbf{w}}$$

We know $M = |\mathbf{x}^{+} - \mathbf{x}^{-}| = |\lambda \mathbf{w}|$, hence $M = \frac{2}{\sqrt{(\mathbf{w} \cdot \mathbf{w})}}$

To maximize M, we have to minimize $||\mathbf{w}||$.

Finding the maximum margin classifier

- Given a guess for $\mathbf{w}$ and b, compute whether the points are all separated, then compute the width of the margin
- Search the space of $\mathbf{w}$ and b to find the largest margin that separates all data points
- We could do gradient descent, simulated annealing, matrix inversion...
- But instead we will use *quadratic programming*
- Quadratic programming is a well-studied class of optimization algorithms to minimize a quadratic function subject to linear constraints.
- Can find extrema much more reliably and efficiently than gradient descent

Finding the optimal hyperplane

We want to minimize: $\frac{||\mathbf{w}||}{2}$ subject to all the points being classified correctly:

$$y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 0$$

This would be easy, if it were not for the constraint.

Lagrange multipliers

- Suppose we want to solve a tiny SVM-like optimization problem: Minimize $\frac{w^2}{2}$ subject to $wx - 1 \geq 0$, where w, x are numbers
- We *incorporate the constraint into the optimization criterion*. We will instead try to minimize:

$$L(w, \alpha) = \frac{w^2}{2} - \alpha(wx - 1)$$

  with $\alpha \geq 0$. We still have a constraint, but it's a simple non-negativity one!
- α is called a *Lagrange multiplier*. We will have one of these for every constraint.
- Now we take the derivative wrt w: $\frac{\partial L}{\partial w} = w - \alpha x = 0$
- Plug the obtained w back into L, take derivative wrt α and set

to 0.

Back to finding the optimal hyperplane

- We introduce the Langrangian:

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i (y_i((w \cdot x_i) + b) - 1)$$

  We have to minimize L wrt $\alpha_i \geq 0$.

- First, we need:

$$\frac{\partial L}{\partial b} = 0 \text{ and } \frac{\partial L}{\partial w} = 0$$

  which yields:

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \text{ and } w = \sum_{i=1}^{l} \alpha_i y_i x_i$$

- The support vectors are the points for which $\alpha_i > 0$.

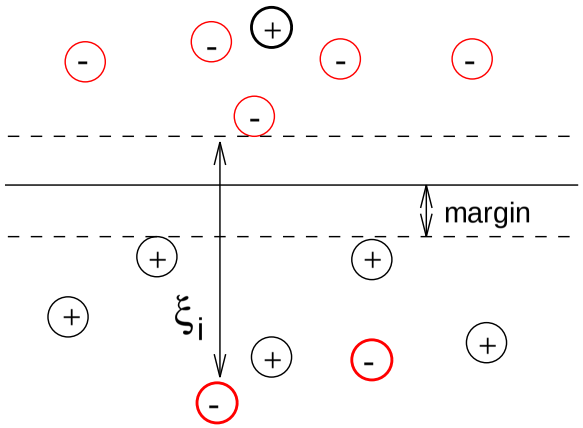
- We get an interpretable decision boundary!

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_{i} \in SV \alpha_i y_i \mathbf{x_i} \cdot \mathbf{x} + b$$

Properties of the Optimal Separating Hyperplane

- It is defined by the support vectors
- Construction of the optimal separating hyperplane does not depend on the dimensionality of the problem
- The description of the hyperplane does not depend on the dimensionality of the problem.

Soft margins



- Suppose the data is not separable. What can we do?
- First idea: minimize $||\mathbf{w}|| + C$, where C is the number of misclassification.

  Problem: this is not Quadratic programming anymore! And it does not account for near vs. far misses.
- Solution: We define **slack variables** wrt a hyperplane and a

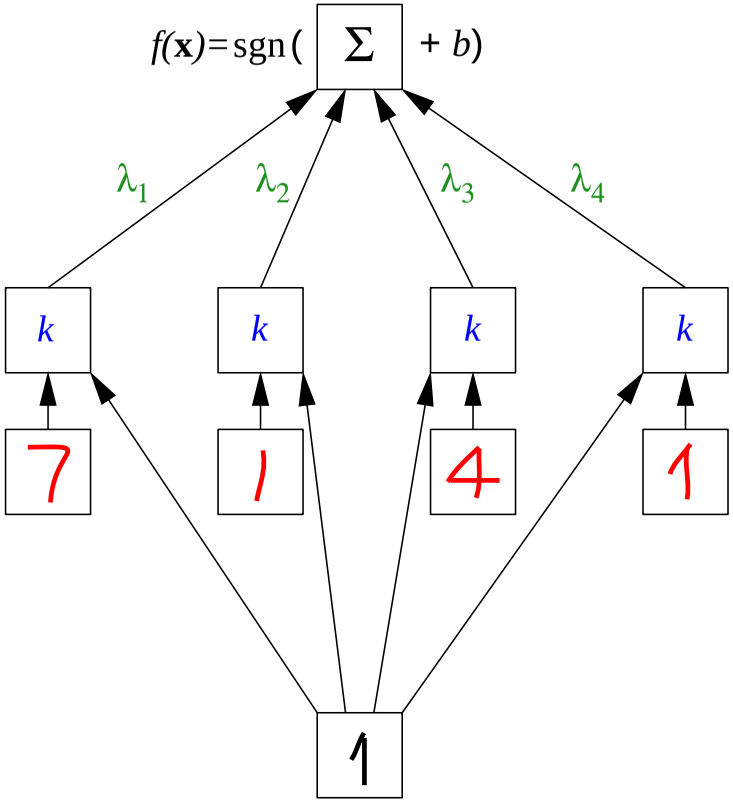
margin γ:

$$\xi_i = \max(0, \gamma - y_i(\mathbf{w} \cdot \mathbf{x}_i + b))$$

Intuitively, the slack allows points to be misclassified, accounting for noise in the data.

- Now we have a quadratic program again, but with $n + 1 + R$ constraints, where R is the size of the data set.

SVM Architecture



classification $f(\mathbf{x}) = \mathrm{sgn}\left(\sum \lambda_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b \right)$

weights

comparison: $k(\mathbf{x},\mathbf{x}_i)$, e.g.

$k(\mathbf{x},\mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)^d$

$k(\mathbf{x},\mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2 / c)$

$k(\mathbf{x},\mathbf{x}_i) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{x}_i) + \theta)$

support vectors
$\mathbf{x}_1 \ldots \mathbf{x}_4$

input vector $\mathbf{x}$

Results for SVMs in Classification Tasks

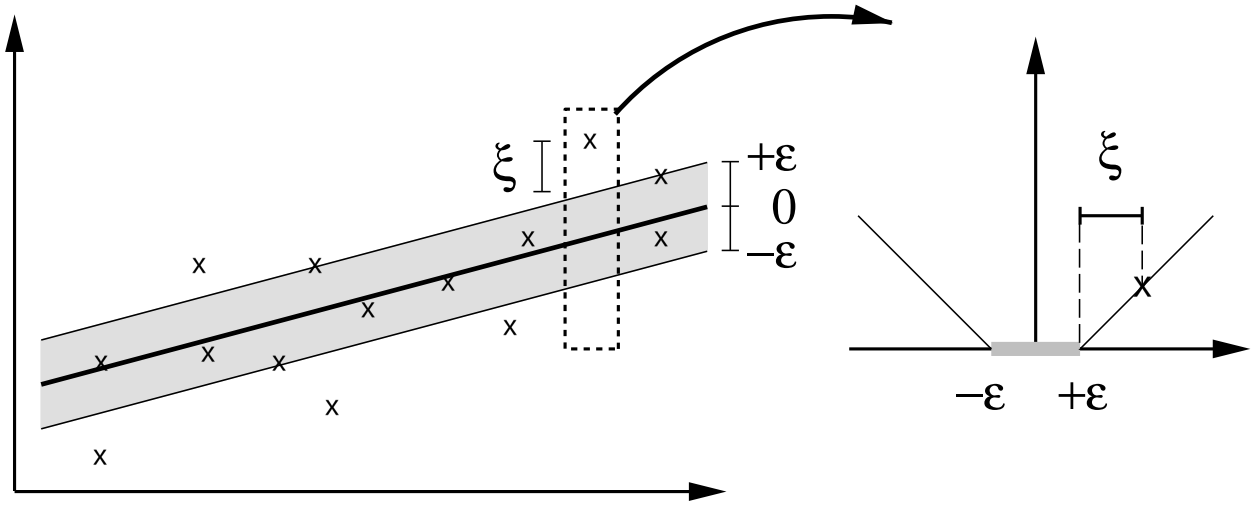
US Postal Service Digit Recognition (AT&T Research):

- 7300 training examples, 2000 test examples, 256 features (digitized image)
- Human error rate: 2.5%
- Decision tree (C4.5): 16.2%
- Best 2-layer neural network: 5.9%
- neural network result known (5 layers): 5.1%
- SVMs: 4-4.2% and 250-300 support vectors (depending on the choice of kernels)

State of art results in face recognition, object recognition etc.

Regression Estimation with Support Vectors

New idea: errors in the approximation smaller than some ϵ are ignored



This is called the ϵ-insensitive zone

Similar to the margin, trades off accuracy vs complexity (number of support vectors)

Generalizing Support Vectors for Regression

We formulate the algorithm for the usual case, then use the kernel trick (just like for classification)

Define ϵ-*insensitive loss*:

$$|y - f(x)|_\epsilon = \max\{0, |y - f(x)| - \epsilon\}$$

Now we want to find a function $f(x) = (w \cdot x) + b$ that minimizes:

$$\frac{1}{2}||w||^2 + \frac{C}{l}\sum_{i=1}^{l}|y_i - f(x_i)|_\epsilon$$

The corresponding optimization problem is:

$$\frac{1}{2}||w||^2 + \frac{C}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

subject to:

$$(w \cdot x_i) + b - y_i \leq \epsilon + \xi_i$$

$$y_i - (w \cdot x_i) + b \leq \epsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0$$

Now we use the Lagrangian and all the previous machinery!

The function we obtain will have the form:

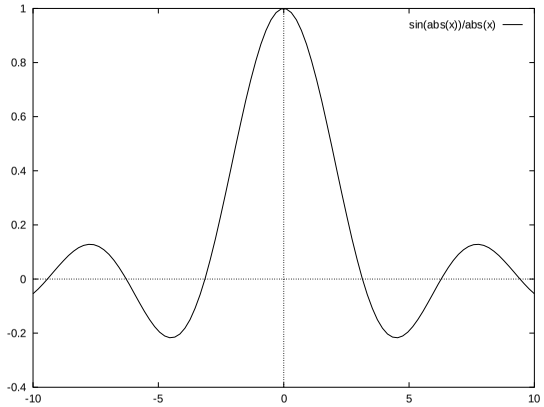
$$f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b$$

What kinds of kernels?

- Fourrier expansions
- Splines
- In general, any series expansions
- Kernels for high-dimensional spaces are often products of 1d kernels

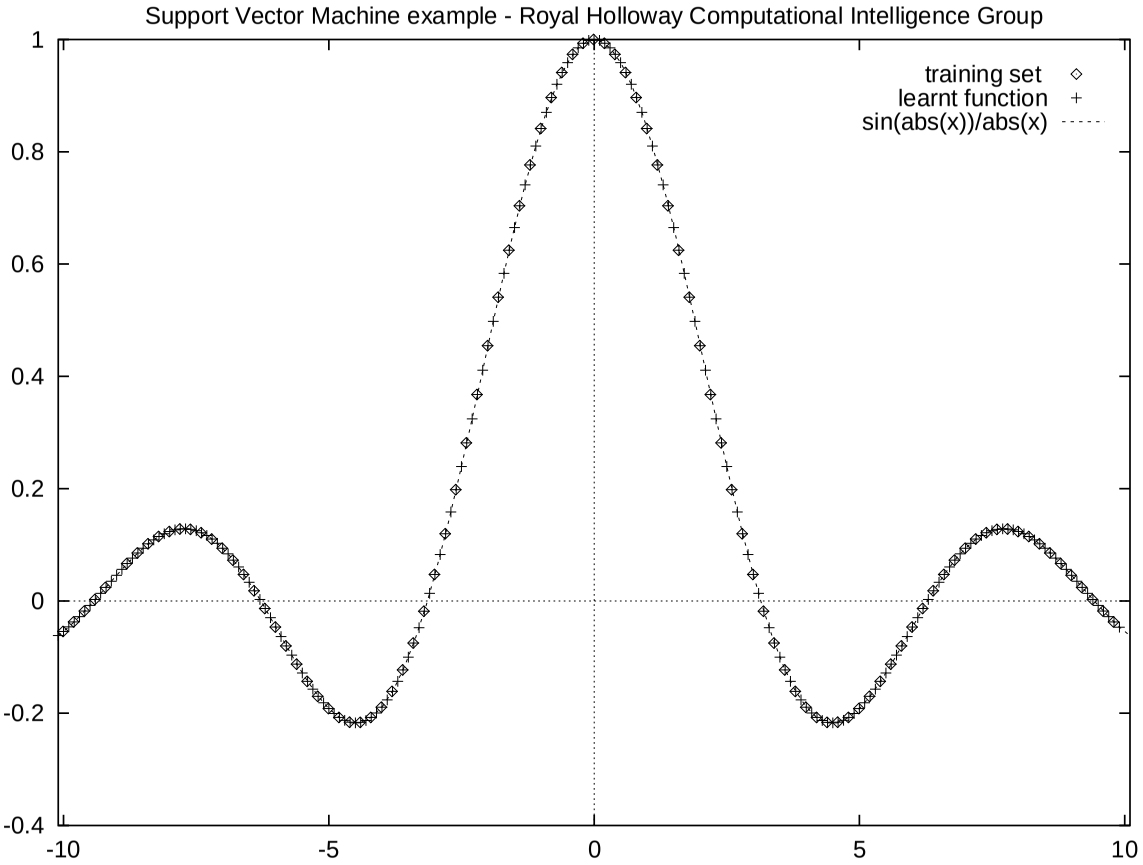
Toy Example: 1D Mexican Hat

$$y = \frac{\sin |x|}{|x|}$$

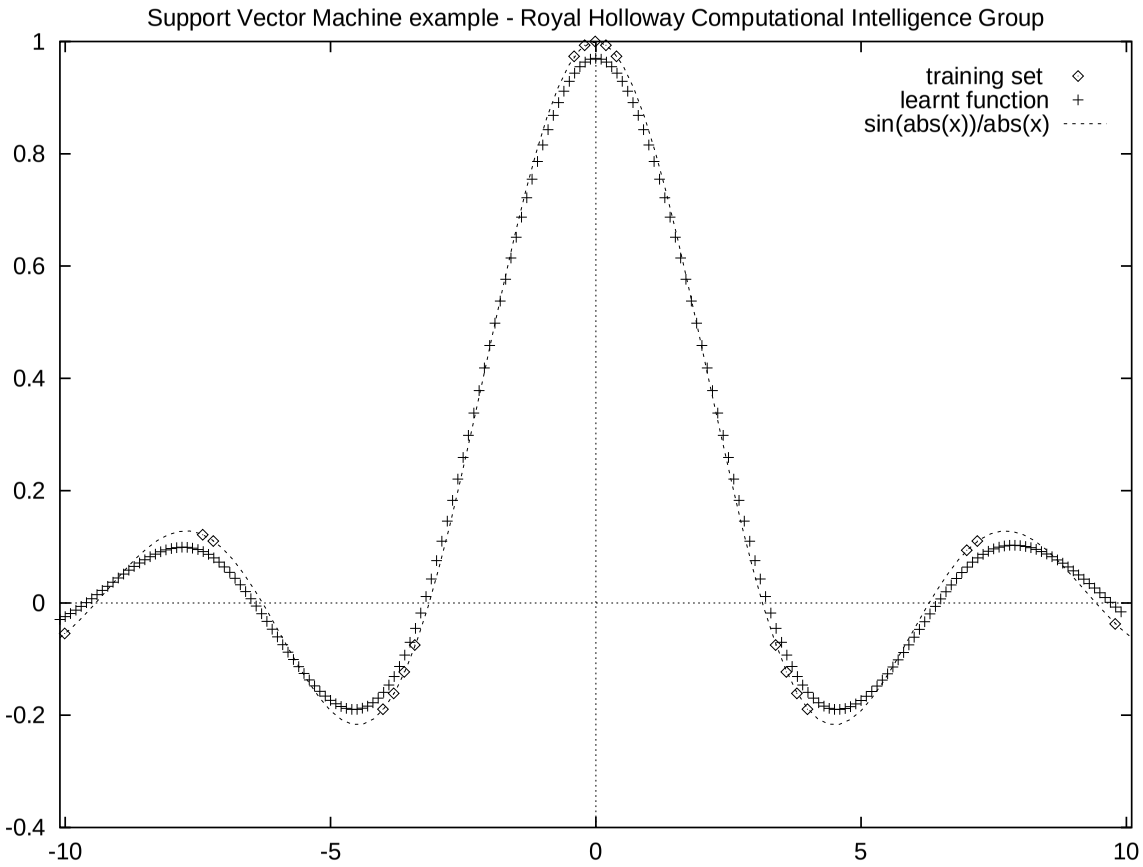


Training on all points on a fixed discretization of $[-1, 1]$ (Vapnik, Golovich and Smola, 1996).

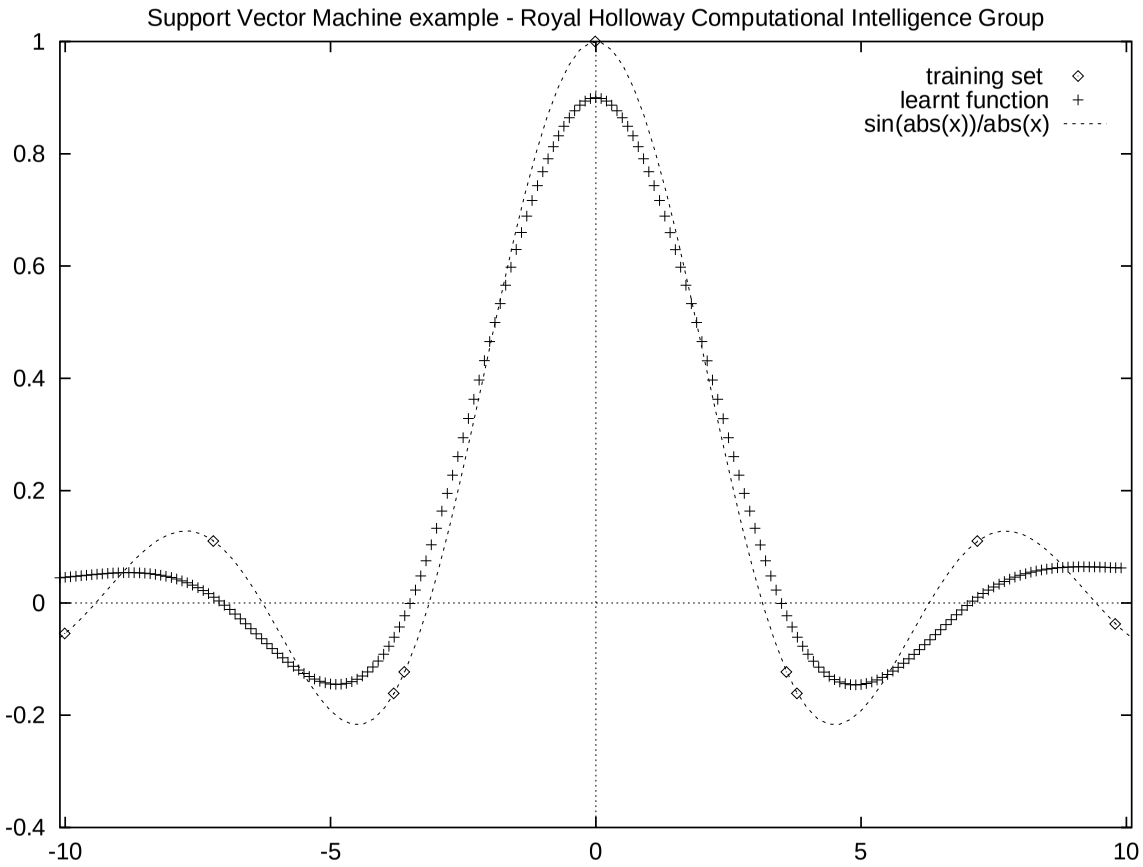
Results for 1D Mexican Hat, $\epsilon = 0$



Results for 1D Mexican Hat, $\epsilon = 0.03$



Results for 1D Mexican Hat, $\epsilon = 0.1$



Mexican Hat: Parameter Influence

Influence of ϵ (training set size = 100):

ϵ	Number of SVs
0	100
0.03	19
0.1	9
0.2	7

Influence of training set size (3D problem, $\epsilon = 0.03$):

Training set size	Number of SVs
400	100
2025	201
7921	267

Regression Results

Boston Housing Problem:

- Regression trees: MSE in the 14-16 range
- Regression trees using bagging: 12
- SVM with splines: 8.8
- Polynomial SVM: 7.2

Summary

- SVMs are an effective general method for representing complex functions in high-dimensional spaces
- Kernels allow a lot of flexibility in the hypothesis space
- Offer polynomial-time *exact optimization* (unlike approximate methods, such as gradient descent and decision trees)
- *Batch method!*
- Offer a lot of choices (kernel form, parameters etc) - which can be good and bad.