BBM406
Fundamentals of Machine Learning
Lecture 17: Kernel Trick for SVMs
Risk and Loss
Support Vector Regression
Administrative

• Project progress reports are due soon!

Due: December 22, 2019 (11:59pm)
Each group should submit a project progress report by December 22, 2019. The report should be **3-4 pages** and should describe the following points as clearly as possible:

- **Problem to be addressed.** Give a short description of the problem that you will explore. Explain why you find it interesting.
- **Related work.** Briefly review the major works related to your research topic.
- **Methodology to be employed.** Describe the neural architecture that is expected to form the basis of the project. State whether you will extend an existing method or you are going to devise your own approach.
- **Experimental evaluation.** Briefly explain how you will evaluate your results. State which dataset(s) you will employ in your evaluation. Provide your preliminary results (if any).
Theorem (Minsky & Papert)
Finding the minimum error separating hyperplane is NP hard

Last time... **Soft-margin Classifier**

$$\langle w, x \rangle + b \leq -1$$

$$\langle w, x \rangle + b \geq 1$$

minimum error separator is impossible
Last time... Adding Slack Variables

\[ \xi_i \geq 0 \]

\[ \langle w, x \rangle + b \leq -1 + \xi \]

\[ \langle w, x \rangle + b \geq 1 - \xi \]

Convex optimization problem

minimize amount of slack
Last time... Adding Slack Variables

- for $0 < \xi \leq 1$ point is between the margin and **correctly classified**
- for $\xi_i \geq 0$ point is **misclassified**

Convex optimization problem

$$\langle w, x \rangle + b \leq -1 + \xi$$

$$\langle w, x \rangle + b \geq 1 - \xi$$

minimize amount of slack

adopted from Andrew Zisserman
Last time... Adding Slack Variables

- Hard margin problem

\[
\min_{w,b} \frac{1}{2} \|w\|^2 \text{ subject to } y_i [\langle w, x_i \rangle + b] \geq 1
\]

- With slack variables

\[
\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum \xi_i
\]

subject to \( y_i [\langle w, x_i \rangle + b] \geq 1 - \xi_i \) and \( \xi_i \geq 0 \)

Problem is always feasible. Proof:

\( w = 0 \) and \( b = 0 \) and \( \xi_i = 1 \) (also yields upper bound)
Soft-margin classifier

- Optimization problem:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\
\text{subject to} & \quad y_i (\langle w, x_i \rangle + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0
\end{align*}
\]

\(C\) is a **regularization** parameter:

- small \(C\) allows constraints to be easily ignored → large margin
- large \(C\) makes constraints hard to ignore → narrow margin
- \(C = \infty\) enforces all constraints: hard margin

adopted from Andrew Zisserman
Last time... Multi-class SVM

- Simultaneously learn 3 sets of weights:
- How do we guarantee the correct labels?
- Need new constraints!

The “score” of the correct class must be better than the “score” of wrong classes:

\[ w(y_j) \cdot x_j + b(y_j) > w(y) \cdot x_j + b(y) \quad \forall j, y \neq y_j \]
Last time... Multi-class SVM

- As for the SVM, we introduce slack variables and maximize margin:

\[
\text{minimize}_{w,b} \sum_{y} w(y) \cdot w(y) + C \sum_{j} \xi_j \\
\quad w(y_j) \cdot x_j + b(y_j) \geq w(y') \cdot x_j + b(y') + 1 - \xi_j, \quad \forall y' \neq y_j, \quad \forall j \\
\quad \xi_j \geq 0, \quad \forall j
\]

To predict, we use:

\[
\hat{y} \leftarrow \text{arg max}_k w_k \cdot x + b_k
\]

Now can we learn it?
Last time... Kernels

- Original data
- Data in feature space (implicit)
- Solve in feature space using kernels
Last time… Quadratic Features

Quadratic Features in $\mathbb{R}^2$

$$\Phi(x) := \left( x_1^2, \sqrt{2} x_1 x_2, x_2^2 \right)$$

Dot Product

$$\langle \Phi(x), \Phi(x') \rangle = \left\langle \left( x_1^2, \sqrt{2} x_1 x_2, x_2^2 \right), \left( x_1'^2, \sqrt{2} x_1' x_2', x_2'^2 \right) \right\rangle = \langle x, x' \rangle^2.$$

Insight

Trick works for any polynomials of order via $\langle x, x' \rangle^d$. 

slide by Alex Smola
Today

• Kernels (cont’d.)
• The Kernel Trick for SVMs
• Risk and Loss
• Support Vector Regression
Computational Efficiency

Problem

- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions. This leads to $5 \cdot 10^5$ numbers. For higher order polynomial features much worse.

Solution

Don’t compute the features, try to compute dot products implicitly. For some features this works . . .

Definition

A kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a symmetric function in its arguments for which the following property holds

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

for some feature map $\Phi$. If $k(x, x')$ is much cheaper to compute than $\Phi(x)$ . . .
Recap: The Perceptron

initialize $w = 0$ and $b = 0$
repeat
  if $y_i [\langle w, x_i \rangle + b] \leq 0$ then
    $w \leftarrow w + y_i x_i$ and $b \leftarrow b + y_i$
  end if
until all classified correctly

- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i x_i$
- Classifier is linear combination of inner products $f(x) = \sum_{i \in I} y_i \langle x_i, x \rangle + b$
Recap: The Perceptron on features

initialize \( w, b = 0 \)

repeat

Pick \((x_i, y_i)\) from data

if \( y_i(w \cdot \Phi(x_i) + b) \leq 0 \) then

\[
\begin{align*}
    w' &= w + y_i \Phi(x_i) \\
    b' &= b + y_i
\end{align*}
\]

until \( y_i(w \cdot \Phi(x_i) + b) > 0 \) for all \( i \)

• Nothing happens if classified correctly
• Weight vector is linear combination \( w = \sum_{i \in I} y_i \phi(x_i) \)
• Classifier is linear combination of inner products \( f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b \)
The Kernel Perceptron

initialize \( f = 0 \)

repeat

\( \text{Pick } (x_i, y_i) \text{ from data} \)

\( \text{if } y_i f(x_i) \leq 0 \text{ then} \)

\( f(\cdot) \leftarrow f(\cdot) + y_i k(x_i, \cdot) + y_i \)

until \( y_i f(x_i) > 0 \) for all \( i \)

• Nothing happens if classified correctly
• Weight vector is linear combination \( w = \sum_{i \in I} y_i \phi(x_i) \)
• Classifier is linear combination of inner products

\[
f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b = \sum_{i \in I} y_i k(x_i, x) + b
\]
Processing Pipeline

- Original data
- Data in feature space (implicit)
- Solve in feature space using kernels
Polynomial Kernels

Idea

We want to extend $k(x, x') = \langle x, x' \rangle^2$ to

$$k(x, x') = \left( \langle x, x' \rangle + c \right)^d$$

where $c > 0$ and $d \in \mathbb{N}$.

Prove that such a kernel corresponds to a dot product.

Proof strategy

Simple and straightforward: compute the explicit sum given by the kernel, i.e.

$$k(x, x') = (\langle x, x' \rangle + c)^d = \sum_{i=0}^{m} \binom{d}{i} (\langle x, x' \rangle)^i c^{d-i}$$

Individual terms $(\langle x, x' \rangle)^i$ are dot products for some $\Phi_i(x)$. 
Kernel Conditions

Computability
We have to be able to compute $k(x, x')$ efficiently (much cheaper than dot products themselves).

“Nice and Useful” Functions
The features themselves have to be useful for the learning problem at hand. Quite often this means smooth functions.

Symmetry
Obviously $k(x, x') = k(x', x)$ due to the symmetry of the dot product $\langle \Phi(x), \Phi(x') \rangle = \langle \Phi(x'), \Phi(x) \rangle$.

Dot Product in Feature Space
Is there always a $\Phi$ such that $k$ really is a dot product?
Mercer’s Theorem

The Theorem
For any symmetric function \( k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) which is square integrable in \( \mathcal{X} \times \mathcal{X} \) and which satisfies
\[
\int_{\mathcal{X} \times \mathcal{X}} k(x, x') f(x) f(x') dx dx' \geq 0 \text{ for all } f \in L_2(\mathcal{X})
\]
there exist \( \phi_i : \mathcal{X} \rightarrow \mathbb{R} \) and numbers \( \lambda_i \geq 0 \) where
\[
k(x, x') = \sum_i \lambda_i \phi_i(x) \phi_i(x') \text{ for all } x, x' \in \mathcal{X}.
\]

Interpretation
Double integral is the continuous version of a vector-matrix-vector multiplication. For positive semidefinite matrices we have
\[
\sum_i \sum_j k(x_i, x_j) \alpha_i \alpha_j \geq 0
\]
Properties

Distance in Feature Space
Distance between points in feature space via

\[ d(x, x')^2 := \| \Phi(x) - \Phi(x') \|^2 \]
\[ = \langle \Phi(x), \Phi(x) \rangle - 2 \langle \Phi(x), \Phi(x') \rangle + \langle \Phi(x'), \Phi(x') \rangle \]
\[ = k(x, x) + k(x', x') - 2k(x, x) \]

Kernel Matrix
To compare observations we compute dot products, so we study the matrix \( K \) given by

\[ K_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j) \]

where \( x_i \) are the training patterns.

Similarity Measure
The entries \( K_{ij} \) tell us the overlap between \( \Phi(x_i) \) and \( \Phi(x_j) \), so \( k(x_i, x_j) \) is a similarity measure.
Properties

\( K \) is Positive Semidefinite

Claim: \( \alpha^\top K \alpha \geq 0 \) for all \( \alpha \in \mathbb{R}^m \) and all kernel matrices \( K \in \mathbb{R}^{m \times m} \). Proof:

\[
\sum_{i,j} \alpha_i \alpha_j K_{ij} = \sum_{i,j} \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle
\]

\[
= \left\langle \sum_{i=1}^m \alpha_i \Phi(x_i), \sum_{j=1}^m \alpha_j \Phi(x_j) \right\rangle = \left\| \sum_{i=1}^m \alpha_i \Phi(x_i) \right\|^2
\]

Kernel Expansion

If \( w \) is given by a linear combination of \( \Phi(x_i) \) we get

\[
\langle w, \Phi(x) \rangle = \left\langle \sum_{i=1}^m \alpha_i \Phi(x_i), \Phi(x) \right\rangle = \sum_{i=1}^m \alpha_i k(x_i, x).
\]
A Counterexample

A Candidate for a Kernel

\[ k(x, x') = \begin{cases} 
1 & \text{if } \|x - x'\| \leq 1 \\
0 & \text{otherwise}
\end{cases} \]

This is symmetric and gives us some information about the proximity of points, yet it is not a proper kernel . . .

Kernel Matrix

We use three points, \( x_1 = 1, x_2 = 2, x_3 = 3 \) and compute the resulting “kernel matrix” \( K \). This yields

\[
K = \begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 1 \\
0 & 1 & 1 \\
\end{bmatrix}
\]

and eigenvalues \((\sqrt{2}-1)^{-1}, 1 \) and \((1-\sqrt{2})\).

as eigensystem. Hence \( k \) is not a kernel.
Examples

Examples of kernels $k(x, x')$

- **Linear**: $\langle x, x' \rangle$
- **Laplacian RBF**: $\exp (-\lambda \|x - x'\|)$
- **Gaussian RBF**: $\exp (-\lambda \|x - x'\|^2)$
- **Polynomial**: $(\langle x, x' \rangle + c)^d$, $c \geq 0$, $d \in \mathbb{N}$
- **B-Spline**: $B_{2n+1}(x - x')$
- **Cond. Expectation**: $E_c[p(x|c)p(x'|c)]$

**Simple trick for checking Mercer’s condition**

Compute the Fourier transform of the kernel and check that it is nonnegative.
Laplacian Kernel

\[ k(x, y) \text{ for } y = 1 \]
Gaussian Kernel

\[ k(x,y) \text{ for } y=1 \]
Polynomial of order 3
$B_3$ Spline Kernel

$k(x,y)$ for $y=1$
Kernels in Computer Vision

- Features $x = \text{histogram (of color, texture, etc)}$

- Common Kernels
  - Intersection Kernel
  - Chi-square Kernel

\[
K_{\text{intersect}}(u, v) = \sum_i \min(u_i, v_i)
\]

\[
K_{\chi^2}(u, v) = \sum_i \frac{2u_i v_i}{u_i + v_i}
\]
The Kernel Trick for SVMs
The Kernel Trick for SVMs

• Linear soft margin problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\
\text{subject to} & \quad y_i \left[\langle w, x_i \rangle + b \right] \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0
\end{align*}
\]

• Dual problem

\[
\begin{align*}
\text{maximize} & \quad -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i \\
\text{subject to} & \quad \sum_i \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \in [0, C]
\end{align*}
\]

• Support vector expansion

\[
f(x) = \sum_i \alpha_i y_i \langle x_i, x \rangle + b
\]
The Kernel Trick for SVMs

• Linear soft margin problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\
\text{subject to} & \quad y_i \left[ \langle w, \phi(x_i) \rangle + b \right] \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0
\end{align*}
\]

• Dual problem

\[
\begin{align*}
\text{maximize} & \quad -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_i \alpha_i \\
\text{subject to} & \quad \sum_i \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \in [0, C]
\end{align*}
\]

• Support vector expansion

\[
f(x) = \sum_i \alpha_i y_i k(x_i, x) + b
\]
C = 2
C = 10
C = 20
C = 50
C = 5
C = 10
C=50
C = 5
$C=100$
C = 1
And now with a narrower kernel
And now with a very wide kernel
Nonlinear Separation

- Increasing $C$ allows for more nonlinearity
- Decreases number of errors
- SV boundary need not be contiguous
- Kernel width adjusts function class
Overfitting?

- Huge feature space with kernels: should we worry about overfitting?
- SVM objective seeks a solution with large margin
  - Theory says that large margin leads to good generalization (we will see this in a couple of lectures)
- But everything overfits sometimes!!
- Can control by:
  - Setting C
  - Choosing a better Kernel
  - Varying parameters of the Kernel (width of Gaussian, etc.)
Risk and Loss
Loss function point of view

- Constrained quadratic program
  
  \[
  \min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\
  \text{subject to } y_i [\langle w, x_i \rangle + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0
  \]

- Risk minimization setting
  
  \[
  \min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_i \max[0, 1 - y_i [\langle w, x_i \rangle + b]]
  \]

  empirical risk

Follows from finding minimal slack variable for given \((w,b)\) pair.
Soft margin as proxy for binary

- **Soft margin loss** \( \max(0, 1 - yf(x)) \)
- **Binary loss** \( \{yf(x) < 0\} \)
More loss functions

- Logistic: \( \log \left[ 1 + e^{-f(x)} \right] \)
- Huberized loss:
  \[
  \begin{cases}
    0 & \text{if } f(x) > 1 \\
    \frac{1}{2}(1 - f(x))^2 & \text{if } f(x) \in [0, 1] \\
    \frac{1}{2} - f(x) & \text{if } f(x) < 0
  \end{cases}
  \]
- Soft margin: \( \max(0, 1 - f(x)) \)
Risk minimization view

• Find function $f$ minimizing classification error
  
  $R[f] := \mathbb{E}_{x,y \sim p(x,y)} \{y f(x) > 0\}$

• Compute empirical average
  
  $R_{\text{emp}}[f] := \frac{1}{m} \sum_{i=1}^{m} \{y_i f(x_i) > 0\}$

  – Minimization is nonconvex
  – Overfitting as we minimize empirical error

• Compute convex upper bound on the loss

• Add regularization for capacity control
  
  $R_{\text{reg}}[f] := \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i f(x_i)) + \lambda \Omega[f]$
Support Vector Regression
Regression Estimation

• Find function $f$ minimizing regression error

$$R[f] := \mathbb{E}_{x,y \sim p(x,y)} [l(y, f(x))]$$

• Compute empirical average

$$R_{\text{emp}}[f] := \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(x_i))$$

Overfitting as we minimize empirical error

• Add regularization for capacity control

$$R_{\text{reg}}[f] := \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(x_i)) + \lambda \Omega[f]$$
Squared loss

\[ l(y, f(x)) = \frac{1}{2} (y - f(x))^2 \]
$l(y, f(x)) = |y - f(x)|$
\( \varepsilon \)-insensitive Loss

\[
l(y, f(x)) = \max(0, |y - f(x)| - \varepsilon)
\]

allow some deviation without a penalty
Penalized least mean squares

- Optimization problem

\[
\min_w \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle x_i, w \rangle)^2 + \frac{\lambda}{2} \|w\|^2
\]

- Solution

\[
\partial_w \ldots = \frac{1}{m} \sum_{i=1}^{m} \left[ x_i x_i^\top w - x_i y_i \right] + \lambda w \\
= \left[ \frac{1}{m} XX^\top + \lambda 1 \right] w - \frac{1}{m} X y = 0
\]

hence \( w = \left[ XX^\top + \lambda m 1 \right]^{-1} X y \)

**Conjugate Gradient**

**Sherman Morrison Woodbury**

**Outer product matrix in X**
Penalized least mean squares ... now with kernels

- Optimization problem

\[
\text{minimize } \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle \phi(x_i), w \rangle)^2 + \frac{\lambda}{2} \|w\|^2
\]

- Representer Theorem (Kimeldorf & Wahba, 1971)

\[
\|w\|^2 = \|w\|^2 + \|w_\perp\|^2
\]

empirical risk dependent
Penalized least mean squares
... now with kernels

- Optimization problem

\[
\min_w \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle \phi(x_i), w \rangle)^2 + \frac{\lambda}{2} ||w||^2
\]

- Representer Theorem (Kimeldorf & Wahba, 1971)
  - Optimal solution is in span of data \( w = \sum \alpha_i \phi(x_i) \)
  - Proof - risk term only depends on data via \( \phi(x_i) \)
  - Regularization ensures that orthogonal part is 0

- Optimization problem in terms of \( w \)

\[
\min_{\alpha} \frac{1}{2m} \sum_{i=1}^{m} \left( y_i - \sum_j K_{ij} \alpha_j \right)^2 + \frac{\lambda}{2} \sum_{i,j} \alpha_i \alpha_j K_{ij}
\]

solve for \( \alpha = (K + m\lambda 1)^{-1} y \) as linear system
Penalized least mean squares ... now with kernels

- Optimization problem
  \[
  \min_w \frac{1}{2m} \sum_{i=1}^{m} (y_i - \langle \phi(x_i), w \rangle)^2 + \frac{\lambda}{2} \|w\|^2
  \]
- Representer Theorem (Kimeldorf & Wahba, 1971)
  \[
  w = \sum_i \alpha_i \phi(x_i)
  \]
- Optimal solution is in span of data
- Proof - risk term only depends on data via \( \phi(x_i) \)
- Regularization ensures that orthogonal part is 0
- Optimization problem in terms of \( \alpha \)
  \[
  \min_{\alpha} \frac{1}{2m} \sum_{i=1}^{m} \left( y_i - \sum_j K_{ij} \alpha_j \right)^2 + \frac{\lambda}{2} \sum_{i,j} \alpha_i \alpha_j K_{ij}
  \]
solve for \( \alpha = (K + m\lambda 1)^{-1} y \) as linear system
SVM Regression
(\(\epsilon\)-insensitive loss)

don’t care about deviations within the tube
SVM Regression (ε-insensitive loss)

- Optimization Problem (as constrained QP)

$$\begin{aligned} &\text{minimize} \quad w, b \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} [\xi_i + \xi_i^*] \\
&\text{subject to} \quad \langle w, x_i \rangle + b \leq y_i + \epsilon + \xi_i \quad \text{and} \quad \xi_i \geq 0 \\
&\quad \langle w, x_i \rangle + b \geq y_i - \epsilon - \xi_i^* \quad \text{and} \quad \xi_i^* \geq 0
\end{aligned}$$

- Lagrange Function

$$L = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} [\xi_i + \xi_i^*] - \sum_{i=1}^{m} [\eta_i \xi_i + \eta_i^* \xi_i^*] + \sum_{i=1}^{m} \alpha_i \left[\langle w, x_i \rangle + b - y_i - \epsilon - \xi_i\right] + \sum_{i=1}^{m} \alpha_i^* \left[y_i - \epsilon - \xi_i^* - \langle w, x_i \rangle - b\right]$$
SVM Regression
(ε-insensitive loss)

• First order conditions

$$\partial_w L = 0 = w + \sum_i [\alpha_i - \alpha_i^*] x_i$$

$$\partial_b L = 0 = \sum_i [\alpha_i - \alpha_i^*]$$

$$\partial_{\xi_i} L = 0 = C - \eta_i - \alpha_i$$

$$\partial_{\xi_i^*} L = 0 = C - \eta_i^* - \alpha_i^*$$

• Dual problem

$$\text{minimize} \quad \frac{1}{2} (\alpha - \alpha^*)^T K (\alpha - \alpha^*) + \epsilon 1^T (\alpha + \alpha^*) + y^T (\alpha - \alpha^*)$$

subject to $$1^T (\alpha - \alpha^*) = 0$$ and $$\alpha_i, \alpha_i^* \in [0, C]$$
Properties

- Ignores ‘typical’ instances with small error
- Only upper or lower bound active at any time
- QP in 2n variables as cheap as SVM problem
- Robustness with respect to outliers
  - $l_1$ loss yields same problem without epsilon
  - Huber’s robust loss yields similar problem but with added quadratic penalty on coefficients
Regression example

From top to bottom: approximation of the function $sinc(x)$ with precisions $\epsilon = 0.1$, $\epsilon = 0.2$, and $\epsilon = 0.5$. The solid top and dashed bottom lines indicate the size of the $\epsilon$-tube, here drawn around the target function $sinc(x)$. The dotted line between them is the regression function.
Regression example

Figure 9.3

From top to bottom: approximation of the function sinc $x$ with precisions $\varepsilon = 0.1$, $0.2$, and $0.5$. The solid top and dashed bottom lines indicate the size of the $\varepsilon$-tube, here drawn around the target function sinc $x$. The dotted line between them is the regression function.
Regression example

Figure 9.3 From top to bottom: approximation of the function \( \text{sinc} \ x \) with precisions \( \epsilon = 0 \), \( \epsilon = 0.1 \), and \( \epsilon = 0.5 \). The solid top and dashed bottom lines indicate the size of the \( \epsilon \)-tube, here drawn around the target function \( \text{sinc} \ x \). The dotted line between them is the regression function.
Regression example

Figure 9.4
Left to right: regression (solid line), datapoints (small dots) and SVs (big dots) for an approximation of \( \text{sinc} x \) (dotted line) with \( \epsilon = 0.1, 0.2, \) and \( 0.5 \). Note the decrease in the number of SVs.

redundant — even without these patterns in the training set, the SVM would have constructed exactly the same function \( f \).

We might use this property as an efficient means of data compression, namely by storing only the support patterns, from which the estimate can be reconstructed completely. Unfortunately, this approach turns out not to work well in the case of noisy high-dimensional data, since for moderate approximation quality, the number of SVs can be rather high.

The parameter \( \epsilon \) of the \( \epsilon \)-insensitive loss is useful if the desired accuracy of the approximation can be specified beforehand. In some cases, however, we just want the estimate to be as accurate as possible, without having to commit ourselves to a specific level of accuracy a priori. When we describe a modification of the \( \epsilon \)-SVR algorithm, called \( \nu \)-SVR, which automatically computes \( \epsilon \).
Huber’s robust loss

\[ l(y, f(x)) = \begin{cases} 
\frac{1}{2} (y - f(x))^2 & \text{if } |y - f(x)| < 1 \\
|y - f(x)| - \frac{1}{2} & \text{otherwise}
\end{cases} \]
Summary

• Advantages:
  - Kernels allow very flexible hypotheses
  - Poly-time exact optimization methods rather than approximate methods
  - Soft-margin extension permits mis-classified examples
  - Variable-sized hypothesis space
  - Excellent results (1.1% error rate on handwritten digits vs. LeNet’s 0.9%)

• Disadvantages:
  - Must choose kernel parameters
  - Very large problems computationally intractable
  - Batch algorithm
Software

- **SVM*light**: one of the most widely used SVM packages. Fast optimization, can handle very large datasets, C++ code.
- **LIBSVM**
- Both of these handle multi-class, weighted SVM for unbalanced data, etc.
- There are several new approaches to solving the SVM objective that can be much faster:
  - Stochastic subgradient method (discussed in a few lectures)
  - Distributed computation (also to be discussed)
Next Lecture: Decision Trees