Advanced Machine Learning
Lecture 18

Support Vector Regression & Co.

16.01.2013

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/
leibe@vision.rwth-aachen.de
This Lecture: *Advanced Machine Learning*

- **Regression Approaches**
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes

- **Bayesian Estimation & Bayesian Non-Parametrics**
  - Prob. Distributions, Approx. Inference
  - Mixture Models & EM
  - Dirichlet Processes
  - Latent Factor Models
  - Beta Processes

- **SVMs and Structured Output Learning**
  - SVMs, SVDD, SV Regression
  - Large-margin Learning
Topics of This Lecture

- Recap: Support Vector Machines
  - Discussion & Analysis

- Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering

- Support Vector Data Description (1-class SVMs)
  - Motivation
  - Definition
  - Applications

- Support Vector Regression
  - Error function
  - Primal form
  - Dual form
Recap: SVM - Analysis

- Traditional soft-margin formulation
  \[
  \min_{\mathbf{w} \in \mathbb{R}^D, \xi_n \in \mathbb{R}^+} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^{N} \xi_n
  \]

  subject to the constraints
  \[
  t_n y(x_n) \geq 1 - \xi_n
  \]

  “Maximize the margin”

  “Most points should be on the correct side of the margin”

- Different way of looking at it
  > We can reformulate the constraints into the objective function.

  \[
  \min_{\mathbf{w} \in \mathbb{R}^D} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^{N} [1 - t_n y(x_n)]_+
  \]

  \(L_2\) regularizer

  “Hinge loss”

  where \([x]_+ := \max\{0,x\}\).
Recap: SVM - Discussion

• SVM optimization function

\[
\min_{\mathbf{w} \in \mathbb{R}^D} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^{N} [1 - t_n y(\mathbf{x}_n)]_+
\]

L₂ regularizer \hspace{2cm} \text{Hinge loss}

• Hinge loss enforces sparsity
  - Only a \textbf{subset of training data points} actually influences the decision boundary.
  - This is different from sparsity obtained through a regularizer! There, only a \textbf{subset of input dimensions} are used.
  - Unconstrained optimization, but non-differentiable function.
  - Solve, e.g. by \textit{subgradient descent}
  - Currently most efficient: \textit{stochastic gradient descent}

Slide adapted from Christoph Lampert
Outline of the Remaining Lectures

- *We will generalize the SVM idea in several directions...*

- Other Kernel methods
  - Kernel PCA
  - Kernel k-Means

- Other Large-Margin Learning formulations
  - Support Vector Data Description (one-class SVMs)
  - Support Vector Regression

- Structured Output Learning
  - General loss functions
  - General structured outputs
  - Structured Output SVM
  - Example: Multiclass SVM
Topics of This Lecture

- Recap: Support Vector Machines
  - Discussion & Analysis

- Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering

- Support Vector Data Description (1-class SVMs)
  - Motivation
  - Definition
  - Applications

- Support Vector Regression
  - Error function
  - Primal form
  - Dual form
Recap: PCA

- **PCA procedure**
  - Given samples $x_n \in \mathbb{R}^d$, PCA finds the directions of maximal covariance. Without loss of generality assume that $\sum_n x_n = 0$.
  - The PCA directions $e_1, \ldots, e_d$ are the eigenvectors of the covariance matrix
    $$C = \frac{1}{N} \sum_{n=1}^{N} x_n x_n^T$$
    sorted by their eigenvalue.
  - We can express $x_n$ in PCA space by
    $$F(x_n) = \sum_{k=1}^{K} \langle x_n, e_k \rangle e_k$$
  - **Lower-dim. coordinate mapping:**
    $$x_n \mapsto \begin{pmatrix} 
\langle x_n, e_1 \rangle \\
\langle x_n, e_2 \rangle \\
\ldots \\
\langle x_n, e_K \rangle 
\end{pmatrix} \in \mathbb{R}^K$$

Slide credit: Christoph Lampert
**Kernel-PCA**

- **Kernel-PCA procedure**
  - Given samples $\mathbf{x}_n \in \mathcal{X}$, kernel $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with an implicit feature map $\phi: \mathcal{X} \to \mathcal{H}$. Perform PCA in the Hilbert space $\mathcal{H}$.
  - The kernel-PCA directions $e_1, \ldots, e_d$ are the eigenvectors of the covariance operator
  
  $\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T$

  sorted by their eigenvalue.

- Lower-dim. coordinate mapping: $\mathbf{x}_n \mapsto \begin{pmatrix} \langle \phi(\mathbf{x}_n), e_1 \rangle \\
\langle \phi(\mathbf{x}_n), e_2 \rangle \\
\vdots \\
\langle \phi(\mathbf{x}_n), e_K \rangle \end{pmatrix} \in \mathbb{R}^K$
Kernel-PCA

- **Kernel-PCA procedure**
  - Given samples $\mathbf{x}_n \in \mathcal{X}$, kernel $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ with an implicit feature map $\phi: \mathcal{X} \rightarrow \mathcal{H}$. Perform PCA in the Hilbert space $\mathcal{H}$.
  - Equivalently, we can use the eigenvectors $e'_k$ and eigenvalues $\lambda_k$ of the kernel matrix
    \[
    K = \left( \langle \phi(\mathbf{x}_m), \phi(\mathbf{x}_n) \rangle \right)_{m,n=1,...,N} = \left( k(\mathbf{x}_m, \mathbf{x}_n) \right)_{m,n=1,...,N}
    \]
  - Coordinate mapping:
    \[
    \mathbf{x}_n \mapsto (\sqrt{\lambda_1}e'_1, \ldots, \sqrt{\lambda_K}e'_K)
    \]
Example: Image Superresolution

- **Training procedure**
  - Collect high-res face images
  - Use KPCA with RBF-kernel to learn non-linear subspaces

- **For new low-res image:**
  - Scale to target high resolution
  - Project to closest point in face subspace

---


Reconstruction in $r$ dimensions
Kernel k-Means Clustering

- Kernel PCA is more than just non-linear versions of PCA
  - PCA maps $\mathbb{R}^d$ to $\mathbb{R}^{d'}$, e.g., to remove noise dimensions.
  - Kernel-PCA maps $\mathcal{X} \rightarrow \mathbb{R}^{d'}$, so it provides a vectorial representation of non-vectorial data.

  $\Rightarrow$ We can apply algorithms that only work in vector spaces to data that is not in a vector representation.

- Example: k-Means clustering
  - Given $x_1, \ldots, x_n \in \mathcal{X}$.
  - Choose a kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.
  - Apply kernel-PCA to obtain vectorial $v_1, \ldots, v_n \in \mathbb{R}^{d'}$.
  - Cluster $v_1, \ldots, v_n \in \mathbb{R}^{d'}$ using K-Means.

  $\Rightarrow x_1, \ldots, x_n$ are clustered based on the similarity defined by $k$. 
Example: Unsupervised Object Categorization

- Automatically group images that show similar objects
  - Represent images by bag-of-word histograms
  - Perform Kernel k-Means Clustering
  => Observation: Clusters get better if we use a good image kernel (e.g., $\chi^2$) instead of plain k-Means (linear kernel).

Topics of This Lecture

• Recap: Support Vector Machines
  ➢ Discussion & Analysis

• Other Kernel Methods
  ➢ Kernel PCA
  ➢ Kernel k-Means Clustering

• Support Vector Data Description (1-class SVMs)
  ➢ Motivation
  ➢ Definition
  ➢ Applications

• Support Vector Regression
  ➢ Error function
  ➢ Primal form
  ➢ Dual form
One-Class SVMs

• Motivation
  ➢ For unlabeled data, we are interested in detecting outliers, i.e. samples that lie far away from most of the other samples.

• Problem statement
  ➢ For samples $x_1, \ldots, x_N$, find the smallest ball (center $c$, radius $R$) that contains “most” of the samples.
  ➢ “Most” again means that we allow some points to have slack.
One-Class SVMs

• Formalization
  
  Solve

  \[
  \min_{R \in \mathbb{R}, \mathbf{c} \in \mathbb{R}^D, \xi_n \in \mathbb{R}^+} \quad R + \frac{1}{\nu N} \sum_{n=1}^{N} \xi_n
  \]

  subject to

  \[
  \| \mathbf{x}_n - \mathbf{c} \|^2 \leq R^2 + \xi_n \quad \text{for } n = 1, \ldots, N
  \]

  where \( \nu \in (0,1) \) upper bounds the number of outliers.
One-Class SVMs

- Again apply the kernel trick
  - Use a kernel \( k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) with an implicit feature map \( \phi: \mathcal{X} \rightarrow \mathcal{H} \).
  - Do outlier detection for \( \phi(x_1), \ldots, \phi(x_N) \):
  - Find the smallest ball (center \( c \in \mathcal{H} \), radius \( R \)) that contains “most” of the samples.

- Solve

\[
\min_{R \in \mathbb{R}, \ c \in \mathcal{H}, \ \xi_n \in \mathbb{R}^+} \quad R + \frac{1}{\nu N} \sum_{n=1}^{N} \xi_n \\
\text{subject to} \quad \| \phi(x_n) - c \|^2 \leq R^2 + \xi_n \quad \text{for } n = 1, \ldots, N
\]
One-Class SVM

- **Solution**
  - The **representer theorem** states that we can write the solution only in terms of the kernel $k(x_n, x_m)$ as

  $$
c = \sum_{n=1}^{N} a_n \phi(x_n)
  $$

  - where again we know from the KKT conditions that for each point $x_n$, either the constraint is active (i.e., the point is on the circle $R$) or the Lagrange multiplier $a_n = 0$.

  ⇒ Sparse solution, depends only on few data points, the support vectors.
  - Because of this, the formulation is called **Support Vector Data Description (SVDD)** or one-class SVM.

  ⇒ Often used for outlier/anomaly detection.
Example: Steganalysis

- Steganography
  - Hide data in other data (e.g. in images)
  - E.g., flip some least significant bits

- Steganalysis
  - Given any data, find out if some data is hidden

Original

With 23’300 hidden bits
Example: Steganalysis

- Possible procedure
  - Compute image statistics (color wavelet coefficients)
  - Train SVDD with RBF-kernel
  - Identified outlier images are suspicious candidates

S. Lyu, H. Farid. Steganalysis using color wavelet statistics and one-class support vector machines, SPIE EI, 2004

Slide adapted from Christoph Lampert
Topics of This Lecture

- Recap: Support Vector Machines
  - Discussion & Analysis

- Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering

- Support Vector Data Description (1-class SVMs)
  - Motivation
  - Definition
  - Applications

- Support Vector Regression
  - Error function
  - Primal form
  - Dual form
SVMs for Regression

- **Linear regression**
  - Minimize a regularized quadratic error function
    \[ \frac{1}{2} \sum_{n=1}^{N} \{y_n - t_n\}^2 + \frac{\lambda}{2} \|w\|^2 \]

- **Problem**
  - Sensitive to outliers, because the quadratic error function penalizes large residues.
  - This is the case even for (Kernel) Ridge Regression, although regularization helps.
SVMs for Regression

- Obtaining sparse solutions
  - Define an $\epsilon$-insensitive error function
    \[
    E_\epsilon(y(x) - t) = \begin{cases} 
    0, & \text{if } |y(x) - t| < \epsilon \\
    |y(x) - t| - \epsilon, & \text{otherwise}
    \end{cases}
    \]
  - and minimize the following regularized function
    \[
    C \sum_{n=1}^{N} E_\epsilon(y_n - t_n) + \frac{1}{2} \|w\|^2
    \]
Dealing with Noise and Outliers

- Introduce slack variables
  - We now need two slack variables \( \xi_n \geq 0 \) and \( \hat{\xi}_n \geq 0 \).
  - A target point lies in the \( \epsilon \)-tube if \( y_n - \epsilon \leq t_n \leq y_n + \epsilon \).
  - The corresponding conditions are

\[
\begin{align*}
t_n & \leq y(x_n) + \epsilon + \xi_n \\
t_n & \geq y(x_n) - \epsilon - \hat{\xi}_n
\end{align*}
\]
Dealing with Noise and Outliers

- Optimization with slack variables
  - The error function can then be rewritten as
    \[ C \sum_{n=1}^{N} [|y(x_n) - t_n| - \epsilon]_+ + \frac{1}{2}||w||^2 \]
  - Using the conditions for the slack variables, we obtain
    \[ t_n \leq y(x_n) + \epsilon + \xi_n \quad \Rightarrow \quad \xi_n \geq -(y(x_n) - t_n) - \epsilon \]
    \[ t_n \geq y(x_n) - \epsilon - \hat{\xi}_n \quad \Rightarrow \quad \hat{\xi}_n \geq (y(x_n) - t_n) - \epsilon \]
  - And thus
    \[ C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2}||w||^2 \]
    \[ \xi_n \geq 0 \]
    \[ \hat{\xi}_n \geq 0 \]
Support Vector Regression - Primal Form

- **Lagrangian primal form**

\[
L_p = C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} (\mu_n \xi_n + \hat{\mu}_n \hat{\xi}_n) \\
- \sum_{n=1}^{N} a_n (\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^{N} \hat{a}_n (\epsilon + \hat{\xi}_n - y_n + t_n)
\]

- **Solving for the variables**

\[
\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N} (a_n - \hat{a}_n) \phi(x_n) \\
\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n + \mu_n = C
\]

\[
\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} (a_n - \hat{a}_n) = 0 \\
\frac{\partial L}{\partial \hat{\xi}_n} = 0 \Rightarrow \hat{a}_n + \hat{\mu}_n = C
\]
Support Vector Regression - Dual Form

• From this, we can derive the dual form
  
  $$L_d(a, \hat{a}) = -\frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} (a_n - \hat{a}_n)(a_m - \hat{a}_m)k(x_n, x_m)$$

  $$-\epsilon \sum_{n=1}^{N} (a_n + \hat{a}_n) + \sum_{n=1}^{N} (a_n - \hat{a}_n)t_n$$

  under the conditions

  $$0 \leq a_n \leq C$$
  $$0 \leq \hat{a}_n \leq C$$

  Predictions for new inputs are then made using

  $$y(x) = \sum_{n=1}^{N} (a_n - \hat{a}_n)k(x, x_n) + b$$
KKT Conditions

• KKT conditions

\[ a_n(\epsilon + \xi_n + y(x_n) - t_n) = 0 \]
\[ \hat{a}_n(\epsilon + \hat{\xi}_n - y(x_n) + t_n) = 0 \]
\[ (C - a_n)\xi_n = 0 \]
\[ (C - \hat{a}_n)\hat{\xi}_n = 0 \]

• Observations

- A coefficient \( a_n \) can only be non-zero if the first constraint is active, i.e., if a point lies either on or above the \( \epsilon \)-tube.
- Similarly, a non-zero coefficient \( \hat{a}_n \) must be on/below the \( \epsilon \)-tube.
- The first two constraints cannot both be active at the same time.
  \[ \Rightarrow \] Either \( a_n \) or \( \hat{a}_n \) or both must be zero.
- The support vectors are those points for which \( a_n \neq 0 \) or \( \hat{a}_n \neq 0 \), i.e., the points on the boundary of or outside the \( \epsilon \)-tube.
Discussion

- Slightly different interpretation
  - For SVMs, classification function depends only on SVs.
  - For SVR, support vectors mark outlier points. SVR tries to limit the effect of those outliers on the regression function.
  - Nevertheless, the prediction $y(x)$ only depends on the support vectors.

Image source: Christoph Lampert
Example: Head Pose Estimation

- Procedure
  - Detect faces in image
  - Compute gradient representation of face region
  - Train support vector regression for yaw, tilt (separately)

References and Further Reading

- More information on Kernel PCA can be found in Chapter 12.3 of Bishop’s book. Support Vector Regression is described in Chapter 7.1. You can also look at Schölkopf & Smola (some chapters available online).

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006

B. Schölkopf, A. Smola
Learning with Kernels
MIT Press, 2002
http://www.learning-with-kernels.org/