Announcements

• Exam dates
  - 1st date: Monday, 17.02., 13:30h - 17:30h
  - 2nd date: Monday, 17.03., 09:30h - 12:30h
  - We tried to avoid overlaps with other Computer Science Master lectures as much as possible.
  - If you still have conflicts with both exam dates, please tell us.

• Please register for the exam on Campus until this Friday (22.11.)!
Recap: SVM - Solution

- Solution for the hyperplane
  - Computed as a linear combination of the training examples
    \[ w = \sum_{n=1}^{N} a_n t_n x_n \]
  - Sparse solution: \( a_n \neq 0 \) only for some points, the support vectors
    \[ \Rightarrow \text{Only the SVs actually influence the decision boundary!} \]
  - Compute \( b \) by averaging over all support vectors:
  \[
  b = \frac{1}{N_S} \sum_{n \in S} \left( t_n - \sum_{m \in S} a_m t_m (x_m^T x_n) \right)
  \]

Recap: SVM - Dual Formulation

- Maximize
  \[ L_d(a) = \frac{1}{N} \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_n^T x_m) \]
  under the conditions
  \[ a_n \geq 0 \quad \forall n \]
  \[ \sum_{n=1}^{N} a_n t_n = 0 \]

- Comparison
  - \( L_d \) is equivalent to the primal form \( L_p \), but only depends on \( a_n \).
  - \( L_d \) scales with \( O(D) \).
  - \( L_d \) scales with \( O(N^2) \) - in practice between \( O(N) \) and \( O(N^2) \).

Recap: SVM for Non-Separable Data

- Slack variables
  - One slack variable \( \xi_n \geq 0 \) for each training data point.
- Interpretation
  - \( \xi_n = 0 \) for points that are on the correct side of the margin.
  - \( \xi_n = |t_n - y(x_n)| \) for all other points.
  \[ \Rightarrow \text{They are jointly optimized together with } w. \]

Recap: Nonlinear SVMs

- General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:
  \[ \Phi: x \rightarrow \phi(x) \]
Recap: The Kernel Trick

- Important observation
  \( \phi(x) \) only appears in the form of dot products \( \phi(x)^T \phi(y) \):
  \[
  y(x) = w^T \phi(x) + b = \sum_{n=1}^{N} a_n t_n \phi(x_n)^T \phi(x) + b
  \]
  - Define a so-called kernel function \( k(x,y) = \phi(x)^T \phi(y) \).
  - Now, in place of the dot product, use the kernel instead:
    \[
    y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b
    \]
  - The kernel function implicitly maps the data to the higher-dimensional space (without having to compute \( \phi(x) \) explicitly!)

Recap: Nonlinear SVM - Dual Formulation

- SVM Dual: Maximize
  \[
  L_D(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m)
  \]
  under the conditions
  \[
  0 \cdot a_n \cdot C \quad \sum_{n=1}^{N} a_n t_n = 0
  \]
- Classify new data points using
  \[
  y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b
  \]

Topics of This Lecture

- Support Vector Machines (Recap)
  - Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
  - Nonlinear Support Vector Machines
- Analysis
  - VC dimensions
  - Error function
- Applications
- Extensions
  - One-class SVMs

VC Dimension for Polynomial Kernel

- Polynomial kernel of degree \( p \):
  \[
  k(x,y) = (x^T y)^p
  \]
  - Dimensionality of \( H \): \( \frac{D + p - 1}{p} \)
  - Example:
    \[
    D = 16 \times 16 = 256 \\
    p = 4 \\
    \text{dim}(H) = 183.181.376
    \]
  - The hyperplane in \( H \) then has VC-dimension \( \text{dim}(H) + 1 \)

VC Dimension for Gaussian RBF Kernel

- Radial Basis Function:
  \[
  k(x,y) = \exp \left\{ -\frac{(x-y)^2}{2\sigma^2} \right\}
  \]
  - In this case, \( H \) is infinite dimensional!
    \[
    \exp(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots
    \]
  - Since only the kernel function is used by the SVM, this is no problem.
  - The hyperplane in \( H \) then has VC-dimension \( \text{dim}(H) + 1 = \infty \)

VC Dimension for Gaussian RBF Kernel

- Intuitively
  - If we make the radius of the RBF kernel sufficiently small, then each data point can be associated with its own kernel.
  - However, this also means that we can get finite VC-dimension if we set a lower limit to the RBF radius.
Example: RBF Kernels

• Decision boundary on toy problem

RBF Kernel width ($\sigma$)

But... but... but...

• Don’t we risk overfitting with those enormously high-dimensional feature spaces?
  - No matter what the basis functions are, there are really only up to $N$ parameters: $a_1, a_2, ..., a_N$ and most of them are usually set to zero by the maximum margin criterion.
  - The data effectively lives in a low-dimensional subspace of $H$.

• What about the VC dimension? I thought low VC-dim was good (in the sense of the risk bound)?
  - Yes, but the maximum margin classifier “magically” solves this.
  - Reason (Vapnik): by maximizing the margin, we can reduce the VC-dimension.
  - Empirically, SVMs have very good generalization performance.

Theoretical Justification for Maximum Margins

• Vapnik has proven the following:
  - The class of optimal linear separators has VC dimension $h$ bounded from above as
    $$ h \leq \min \left\{ \frac{D^2}{\rho^2}, m_0 \right\} + 1 $$
    where $\rho$ is the margin, $D$ is the diameter of the smallest sphere that can enclose all of the training examples, and $m_0$ is the dimensionality.

• Intuitively, this implies that regardless of dimensionality $m_0$ we can minimize the VC dimension by maximizing the margin $\rho$.

• Thus, complexity of the classifier is kept small regardless of dimensionality.

Idea of the Proof

• Gap Tolerant Classifier
  - Classifier is defined by a ball in $\mathbb{R}^d$ with diameter $D$ enclosing all points and two parallel hyperplanes with distance $M$ (the margin).
  - Points in the ball are assigned class $\{-1, 1\}$ depending on which side of the margin they fall.

• VC dimension of this classifier depends on the margin
  - $M \leq 3/4 D \implies 3$ points can be shattered
  - $3/4 D < M < D \implies 2$ points can be shattered
  - $M \geq D \implies 1$ point can be shattered

• By maximizing the margin, we can minimize the VC dimension

Topics of This Lecture

• Support Vector Machines (Recap)
  - Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
  - Nonlinear Support Vector Machines

• Analysis
  - VC dimensions
  - Error function

• Applications

• Extensions
  - One-class SVMs

SVM Demo

Applet from libsvm
(http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
SVM - Analysis

- **Traditional soft-margin formulation**
  \[
  \min_{w \in \mathbb{R}^p, \xi_n \in \mathbb{R}^+} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n \\
  \text{subject to the constraints} \\
  t_n y(x_n) \geq 1 - \xi_n
  \]
  "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.
  \[
  \min_{w \in \mathbb{R}^p} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} [1 - t_n y(x_n)]_+
  \]
  \[\text{L}_2 \text{ regularizer} \quad \text{"Hinge loss"}\]
  where \([x]_+ := \max(0,x)\).

Recap: Error Functions

- **Ideal misclassification error function (black)**
  - This is what we want to approximate,
  - Unfortunately, it is not differentiable.
  - The gradient is zero for misclassified points.
  \[\Rightarrow \text{We cannot minimize it by gradient descent.}\]

- **Squared error used in Least-Squares Classification**
  - Very popular, leads to closed-form solutions.
  - However, sensitive to outliers due to squared penalty.
  - Penalizes "too correct" data points
  \[\Rightarrow \text{Generally does not lead to good classifiers.}\]

- **"Hinge error" used in SVMs**
  - Zero error for points outside the margin \(z_n > 1\) \(\Rightarrow\) sparsity
  - Linear penalty for misclassified points \(z_n < 1\) \(\Rightarrow\) robustness
  - Not differentiable around \(z_n = 1\) \(\Rightarrow\) Cannot be optimized directly.

SVM - Discussion

- **SVM optimization function**
  \[
  \min_{w \in \mathbb{R}^p} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} [1 - t_n y(x_n)]_+
  \]
  \[\text{L}_2 \text{ regularizer} \quad \text{Hinge loss}\]

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

Recap: Error Functions

- **Error Functions (Loss Functions)**
  - **Squared error**
    - Favors sparse solutions!
  - **Hinge error**
    - Robust to outliers!
    - Not differentiable!

Error Functions (Loss Functions)

- **Ideal misclassification error**
  - This is what we want to approximate,
  - Unfortunately, it is not differentiable.
  - The gradient is zero for misclassified points.
  \[\Rightarrow \text{We cannot minimize it by gradient descent.}\]

Topics of This Lecture

- **Support Vector Machines (Recap)**
  - Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
  - Nonlinear Support Vector Machines

- **Analysis**
  - VC dimensions
  - Error function

- **Applications**
  - One-class SVMs
Example Application: Text Classification

- **Problem:**
  - Classify a document in a number of categories

- **Representation:**
  - "Bag-of-words" approach
  - Histogram of word counts (on learned dictionary)
    - Very high-dimensional feature space (~10,000 dimensions)
    - Few irrelevant features

- **This was one of the first applications of SVMs**
  - T. Joachims (1997)

Example Application: OCR

- **Handwritten digit recognition**
  - US Postal Service Database
  - Standard benchmark task for many learning algorithms

Example Application: OCR

- **Results**

<table>
<thead>
<tr>
<th>degree of polynomial</th>
<th>dimensionality of feature space</th>
<th>support vectors</th>
<th>raw error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>282</td>
<td>8.9</td>
</tr>
<tr>
<td>2</td>
<td>( \approx 3,000 )</td>
<td>227</td>
<td>4.7</td>
</tr>
<tr>
<td>3</td>
<td>( \approx 1 \times 10^6 )</td>
<td>274</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>( \approx 1 \times 10^9 )</td>
<td>321</td>
<td>4.2</td>
</tr>
<tr>
<td>5</td>
<td>( \approx 1 \times 10^{12} )</td>
<td>374</td>
<td>4.3</td>
</tr>
<tr>
<td>6</td>
<td>( \approx 1 \times 10^{14} )</td>
<td>377</td>
<td>4.5</td>
</tr>
<tr>
<td>7</td>
<td>( \approx 1 \times 10^{16} )</td>
<td>422</td>
<td>4.5</td>
</tr>
</tbody>
</table>
Example Application: Object Detection

- Sliding-window approach
  - E.g. histogram representation (HOG)
    - Map each grid cell in the input window to a histogram of gradient orientations.
    - Train a linear SVM using training set of pedestrian vs. non-pedestrian windows.

Example Application: Pedestrian Detection

Many Other Applications

- Lots of other applications in all fields of technology
  - OCR
  - Text classification
  - Computer vision
  - ...
  - High-energy physics
  - Monitoring of household appliances
  - Protein secondary structure prediction
  - Design on decision feedback equalizers (DFE) in telephony

(Detailed references in Schoelkopf & Smola, 2002, pp. 221)

Topics of This Lecture

- Support Vector Machines (Recap)
  - Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
  - Nonlinear Support Vector Machines
- Analysis
  - VC dimensions
  - Error function
- Applications
- Extensions
  - One-class SVMs

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

\[
\begin{align*}
\min_{R \in \mathbb{R}, c \in \mathbb{R}^d, \xi_n \in \mathbb{R}_+} & \quad R + \frac{1}{2}N \sum_{n=1}^{N} \xi_n \\
\text{subject to} & \quad \|x_n - c\|^2 \leq R^2 + \xi_n \quad \text{for } n = 1, \ldots, N
\end{align*}
\]

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

- Kernel formulation
  - Solving the Lagrange primal for \( c \)
    - We obtain
      \[
      \frac{\partial E(c)}{\partial c} = 0 \quad \Rightarrow \quad c = \sum_{n=1}^{N} a_n \phi(x_n)
      \]
  - From this, we can derive
    \[
    \|\phi(x_n) - c\|^2 = \phi(x_n)^T \phi(x_n) - 2 \phi(x_n)^T c + c^T c
    \]
    \[
    = \phi(x_n)^T \phi(x_n) - 2 \sum_{n=1}^{N} a_n \phi(x_n)^T \phi(x_n) + \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m \phi(x_n)^T \phi(x_m)
    \]
    \[
    = k(x_n, x_n) - 2 \sum_{m=1}^{N} a_m k(x_n, x_m) + \sum_{m=1}^{N} \sum_{n=1}^{N} a_n a_m k(x_n, x_m)
    \]

One-Class SVM

- Result
  - We can again use kernels \( k(x_n, x_m) \) to express the solution
    \[
    c = \sum_{n=1}^{N} a_n \phi(x_n)
    \]
  - 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

Example: Intrusion Detection

- Monitor network or system activities
  - Search for anomalies that could correspond to malicious activities or policy violations.
  - Statistical anomaly based approaches learn a model of regular user behavior and flag outliers.
- SVDD has been successfully applied for this in the past
  - Main issue: designing a suitable kernel function for comparing system activity
  - This is where domain knowledge can come in.
Summary: SVMs

Properties
- Empirically, SVMs work very, very well.
- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g., graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been applied to a variety of other tasks (e.g., SV Regression, One-class SVMs, ...)
- The kernel trick has been used for a wide variety of applications. It can be applied wherever dot products are in use (e.g., Kernel PCA, kernel FLD, ...)
- Good overview, software, and tutorials available on http://www.kernel-machines.org/

Limitations
- How to select the right kernel? Requires domain knowledge and experiments...
- How to select the kernel parameters? (Massive) cross-validation. Usually, several parameters are optimized together in a grid search.
- Solving the quadratic programming problem: Standard QP solvers do not perform too well on SVM task. Dedicated methods have been developed for this, e.g., SMO.
- Speed of evaluation: Evaluating \( y(x) \) scales linearly in the number of SVs. Too expensive if we have a large number of support vectors. \( \Rightarrow \) There are techniques to reduce the effective SV set.
- Training for very large datasets (millions of data points): Stochastic gradient descent and other approximations can be used.

You Can Try It At Home...

- Lots of SVM software available, e.g.
  - svmlight (http://svmlight.joachims.org/)
  - Command-line based interface
  - Source code available (in C)
  - Interfaces to Python, MATLAB, Perl, Java, DLL,...
  - libsvm (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
  - Library for inclusion with own code
  - C++ and Java sources
  - Interfaces to Python, R, MATLAB, Perl, Ruby, Weka, C++, .NET,...
- Both include fast training and evaluation algorithms, support for multi-class SVMs, automated training and cross-validation, ...
  \( \Rightarrow \) Easy to apply to your own problems!

References and Further Reading

- More information on SVMs can be found in Chapter 7.1 of Bishop’s book. You can also look at Schölkopf & Smola (some chapters available online).
- A more in-depth introduction to SVMs is available in the following tutorial: