Machine Learning – Lecture 8

Support Vector Machines

06.11.2019

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

leibe@vision.rwth-aachen.de
Announcements

• Exam dates
  - 1st date: Saturday, 29.02., 13:30h – 15:30h
  - 2nd date: Thursday, 19.03., 11:00h – 13:00h
  - The exam dates have been optimized 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.
  - If you’re an exchange student and need to leave RWTH before the first exam date, we will offer some special oral exam slots
    - Please do NOT contact us about those yet.
    - We will let you sign up for those special exam slots in early January

• Please do not forget to register for the exam in RWTH online!
Course Outline

• Fundamentals
  ➢ Bayes Decision Theory
  ➢ Probability Density Estimation

• Classification Approaches
  ➢ Linear Discriminants
  ➢ Support Vector Machines
  ➢ Ensemble Methods & Boosting
  ➢ Randomized Trees, Forests & Ferns

• Deep Learning
  ➢ Foundations
  ➢ Convolutional Neural Networks
  ➢ Recurrent Neural Networks
Topics of This Lecture

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

• **Nonlinear Support Vector Machines**
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels

• **Analysis**
  - Error function

• **Applications**
Recap: Support Vector Machine (SVM)

• Basic idea
  ➢ The SVM tries to find a classifier which maximizes the margin between pos. and neg. data points.
  ➢ Up to now: consider linear classifiers

\[ w^T x + b = 0 \]

• Formulation as a convex optimization problem
  ➢ Find the hyperplane satisfying

\[ \arg \min_{w, b} \frac{1}{2} \|w\|^2 \]

under the constraints

\[ t_n (w^T x_n + b) \geq 1 \quad \forall n \]

based on training data points \( x_n \) and target values \( t_n \in \{-1, 1\} \)
Support Vector Machine (SVM)

- Optimization problem
  - Find the hyperplane satisfying
    \[
    \arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2
    \]
    under the constraints
    \[
    t_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \quad \forall n
    \]
  - Quadratic programming problem with linear constraints.
  - Can be formulated using Lagrange multipliers.

- **Who is already familiar with Lagrange multipliers?**
  - Let’s look at a real-life example…
Recap: Lagrange Multipliers

- Problem
  - We want to maximize $K(\mathbf{x})$ subject to constraints $f(\mathbf{x}) = 0$.
  - Example: we want to get as close as possible, but there is a fence.
  - How should we move?

\[ f(\mathbf{x}) = 0 \quad f(\mathbf{x}) < 0 \quad f(\mathbf{x}) > 0 \]

- We want to maximize $\nabla K$
- But we can only move parallel to the fence, i.e. along

\[ \nabla \| K = \nabla K + \lambda \nabla f \]

with $\lambda \neq 0$. 

Slide adapted from Mario Fritz
Recap: Lagrange Multipliers

• Problem
  - We want to maximize $K(x)$ subject to constraints $f(x) = 0$.
  - Example: we want to get as close as possible, but there is a fence.
  - How should we move?

\[
f(x) = 0 \quad f(x) < 0
\]

$\Rightarrow$ Optimize

\[
\max_{x, \lambda} L(x, \lambda) = K(x) + \lambda f(x)
\]

\[
\frac{\partial L}{\partial x} = \nabla || K = 0
\]

\[
\frac{\partial L}{\partial \lambda} = f(x) \neq 0
\]
Recap: Lagrange Multipliers

- **Problem**
  - Now let’s look at constraints of the form $f(x) \geq 0$.
  - Example: There might be a hill from which we can see better…
  - Optimize $\max_{x,\lambda} L(x, \lambda) = K(x) + \lambda f(x)$
    - $f(x) = 0$
    - $f(x) < 0$

- **Two cases**
  - $f(x) > 0$
    - Solution lies on boundary
      $\Rightarrow f(x) = 0$ for some $\lambda > 0$
    - Solution lies inside $f(x) > 0$
      $\Rightarrow$ Constraint inactive: $\lambda = 0$
    - In both cases
      $\Rightarrow \lambda f(x) = 0$
Recap: Lagrange Multipliers

- Problem
  - Now let’s look at constraints of the form \( f(x) \geq 0 \).
  - Example: There might be a hill from which we can see better…
  - Optimize \( \max_{x, \lambda} L(x, \lambda) = K(x) + \lambda f(x) \) \( f(x) = 0 \)

Two cases
- Solution lies on boundary
  \( \Rightarrow f(x) = 0 \) for some \( \lambda > 0 \)
- Solution lies inside \( f(x) > 0 \)
  \( \Rightarrow \) Constraint inactive: \( \lambda = 0 \)
- In both cases
  \( \Rightarrow \lambda f(x) = 0 \)

Karush-Kuhn-Tucker (KKT) conditions:
- \( \lambda \geq 0 \)
- \( f(x) \geq 0 \)
- \( \lambda f(x) = 0 \)
SVM – Lagrangian Formulation

• Find hyperplane minimizing \( \|w\|^2 \) under the constraints

\[
t_n (w^T x_n + b) - 1 \geq 0 \quad \forall n
\]

• Lagrangian formulation
  
  - Introduce positive Lagrange multipliers: \( a_n \geq 0 \quad \forall n \)
  
  - Minimize Lagrangian (“primal form”)

\[
L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n (w^T x_n + b) - 1 \right\}
\]

  - I.e., find \( w, b, \) and \( a \) such that

\[
\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} a_n t_n = 0
\]

\[
\frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{n=1}^{N} a_n t_n x_n
\]
SVM – Lagrangian Formulation

• Lagrangian primal form

\[
L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n (w^T x_n + b) - 1 \right\}
\]

\[
= \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n y(x_n) - 1 \right\}
\]

• The solution of \( L_p \) needs to fulfill the KKT conditions
  
  - Necessary and sufficient conditions
    \[
    a_n \geq 0 \\
    t_n y(x_n) - 1 \geq 0 \\
    a_n \left\{ t_n y(x_n) - 1 \right\} = 0
    \]

  
  \[
  \text{KKT:} \\
  \lambda \geq 0 \\
  f(x) \geq 0 \\
  \lambda f(x) = 0
  \]
SVM – Solution (Part 1)

• Solution for the hyperplane
  - Computed as a linear combination of the training examples
    \[ w = \sum_{n=1}^{N} a_n t_n x_n \]
  - Because of the KKT conditions, the following must also hold
    \[ a_n \left( t_n \left( w^T x_n + b \right) - 1 \right) = 0 \]
  - This implies that \( a_n > 0 \) only for training data points for which
    \( \left( t_n \left( w^T x_n + b \right) - 1 \right) = 0 \)

⇒ Only some of the data points actually influence the decision boundary!
SVM – Support Vectors

• The training points for which \( a_n > 0 \) are called "support vectors".

• Graphical interpretation:
  - The support vectors are the points on the margin.
  - They define the margin and thus the hyperplane.

⇒ Robustness to "too correct" points!
SVM – Solution (Part 2)

• Solution for the hyperplane
  - To define the decision boundary, we still need to know $b$.
  - Observation: any support vector $\mathbf{x}_n$ satisfies
    \[
    t_n y(\mathbf{x}_n) = t_n \left( \sum_{m \in S} a_m t_m \mathbf{x}_m^T \mathbf{x}_n + b \right) = 1
    \]
  - Using $t_n^2 = 1$ we can derive:
    \[b = t_n - \sum_{m \in S} a_m t_m \mathbf{x}_m^T \mathbf{x}_n\]
  - In practice, it is more robust to average over all support vectors:
    \[b = \frac{1}{N_S} \sum_{n \in S} \left( t_n - \sum_{m \in S} a_m t_m \mathbf{x}_m^T \mathbf{x}_n \right)\]
SVM – Discussion (Part 1)

• Linear SVM
  ➢ Linear classifier
  ➢ SVMs have a “guaranteed” generalization capability.
  ➢ Formulation as convex optimization problem.
    ⇒ Globally optimal solution!

• Primal form formulation
  ➢ Solution to quadratic prog. problem in \( M \) variables is in \( O(M^3) \).
  ➢ Here: \( D \) variables ⇒ \( O(D^3) \)
  ➢ Problem: scaling with high-dim. data ("curse of dimensionality")
SVM – Dual Formulation

- Improving the scaling behavior: rewrite $L_p$ in a dual form

\[
L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n (w^T x_n + b) - 1 \right\}
\]

\[
= \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n t_n w^T x_n - b \sum_{n=1}^{N} a_n t_n + \sum_{n=1}^{N} a_n
\]

- Using the constraint $\sum_{n=1}^{N} a_n t_n = 0$ we obtain

\[
\frac{\partial L_p}{\partial b} = 0
\]

\[
L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n t_n w^T x_n + \sum_{n=1}^{N} a_n
\]

Slide adapted from Bernt Schiele
SVM – Dual Formulation

\[ L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n t_n w^T x_n + \sum_{n=1}^{N} a_n \]

- Using the constraint \( w = \sum_{n=1}^{N} a_n t_n x_n \) we obtain

\[ \nabla L_p = 0 \]

\[ L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n t_n \sum_{m=1}^{N} a_m t_m x_m^T x_n + \sum_{n=1}^{N} a_n \]

\[ = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) + \sum_{n=1}^{N} a_n \]
SVM – Dual Formulation

\[ L = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) + \sum_{n=1}^{N} a_n \]

- Applying \( \frac{1}{2} \|w\|^2 = \frac{1}{2} w^T w \) and again using \( w = \sum_{n=1}^{N} a_n t_n x_n \)

\[ \frac{1}{2} w^T w = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) \]

- Inserting this, we get the Wolfe dual

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) \]
SVM – Dual Formulation

• Maximize

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) \]

under the conditions

\[ a_n \geq 0 \quad \forall n \]

\[ \sum_{n=1}^{N} a_n t_n = 0 \]

➢ The hyperplane is given by the \( N_S \) support vectors:

\[ w = \sum_{n=1}^{N_S} a_n t_n x_n \]
SVM – Discussion (Part 2)

• Dual form formulation
  - In going to the dual, we now have a problem in $N$ variables ($a_n$).
  - Isn’t this worse??? We penalize large training sets!

• However…
  1. SVMs have sparse solutions: $a_n \neq 0$ only for support vectors!
     ⇒ This makes it possible to construct efficient algorithms
        - e.g. Sequential Minimal Optimization (SMO)
        - Effective runtime between $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$.
  2. We have avoided the dependency on the dimensionality.
     ⇒ This makes it possible to work with infinite-dimensional feature spaces by using suitable basis functions $\phi(x)$.
     ⇒ We’ll see that later in today’s lecture…
So Far…

• Only looked at linearly separable case…
  - Current problem formulation has no solution if the data are not linearly separable!
  - Need to introduce some tolerance to outlier data points.
SVM – Non-Separable Data

• Non-separable data
  - I.e. the following inequalities cannot be satisfied for all data points
    \[ \mathbf{w}^T \mathbf{x}_n + b \geq 1 \quad \text{for} \quad t_n = +1 \]
    \[ \mathbf{w}^T \mathbf{x}_n + b \leq -1 \quad \text{for} \quad t_n = -1 \]
  - Instead use
    \[ \mathbf{w}^T \mathbf{x}_n + b \geq 1 - \xi_n \quad \text{for} \quad t_n = +1 \]
    \[ \mathbf{w}^T \mathbf{x}_n + b \leq -1 + \xi_n \quad \text{for} \quad t_n = -1 \]
  - with “slack variables” \( \xi_n \geq 0 \quad \forall n \)
SVM – Soft-Margin Classification

- 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 (linear penalty).
  - We do not have to set the slack variables ourselves!
    $\Rightarrow$ They are jointly optimized together with $w$.

Point on decision boundary: $\xi_n = 1$

Misclassified point: $\xi_n > 1$
SVM – Non-Separable Data

- Separable data
  - Minimize
    \[ \frac{1}{2} \|w\|^2 \]

- Non-separable data
  - Minimize
    \[ \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n \]
SVM – New Primal Formulation

• New SVM Primal: Optimize

\[ L_p = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} a_n (t_n y(x_n) - 1 + \xi_n) - \sum_{n=1}^{N} \mu_n \xi_n \]

Constraint

\[ t_n y(x_n) \geq 1 - \xi_n \]

Constraint

\[ \xi_n \geq 0 \]

• KKT conditions

\[ a_n \geq 0 \quad \mu_n \geq 0 \]

\[ t_n y(x_n) - 1 + \xi_n \geq 0 \quad \xi_n \geq 0 \]

\[ a_n (t_n y(x_n) - 1 + \xi_n) = 0 \quad \mu_n \xi_n = 0 \]

**KKT:**

\[ \lambda \geq 0 \]

\[ f(x) \geq 0 \]

\[ \lambda f(x) = 0 \]
SVM – New Dual Formulation

- New SVM Dual: Maximize

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) \]

under the conditions

\[ 0 \cdot a_n \cdot C \]

\[ \sum_{n=1}^{N} a_n t_n = 0 \]

- This is again a quadratic programming problem

⇒ Solve as before… (more on that later)

This is all that changed!
SVM – New Solution

• Solution for the hyperplane
  - Computed as a linear combination of the training examples
    \[ \mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n \]
  - Again sparse solution: \( a_n = 0 \) for points outside the margin.
  \( \Rightarrow \) The slack points with \( \xi_n > 0 \) are now also support vectors!
  - Compute \( b \) by averaging over all \( N_M \) points with \( 0 < a_n < C \):
    \[
    b = \frac{1}{N_M} \sum_{n \in \mathcal{M}} \left( t_n - \sum_{m \in \mathcal{M}} a_m t_m \mathbf{x}_m^T \mathbf{x}_n \right)
    \]
Interpretation of Support Vectors

- Those are the hard examples!
  - We can visualize them, e.g. for face detection

Image source: E. Osuna, F. Girosi, 1997
Topics of This Lecture

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

• Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels

• Analysis
  - Error function

• Applications
So Far…

- Only looked at linearly separable case…
  - Current problem formulation has no solution if the data are not linearly separable!
  - Need to introduce some tolerance to outlier data points.
    ⇒ Slack variables.

- Only looked at linear decision boundaries…
  - This is not sufficient for many applications.
  - Want to generalize the ideas to non-linear boundaries.
Nonlinear SVM

• Linear SVMs
  - Datasets that are linearly separable with some noise work well:
    - But what are we going to do if the dataset is just too hard?
  - How about… mapping data to a higher-dimensional space:

Slide credit: Raymond Mooney
Nonlinear SVM – Feature Spaces

• General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: \ x \rightarrow \phi(x) \]
Nonlinear SVM

• General idea
  
  ➢ Nonlinear transformation $\phi$ of the data points $x_n$:
  
  $$ x \in \mathbb{R}^D \quad \phi : \mathbb{R}^D \rightarrow \mathcal{H} $$

  ➢ Hyperplane in higher-dim. space $\mathcal{H}$ (linear classifier in $\mathcal{H}$)
  
  $$ \mathbf{w}^T \phi(x) + b = 0 $$

  $\Rightarrow$ Nonlinear classifier in $\mathbb{R}^D$. 

Slide credit: Bernt Schiele
What Could This Look Like?

- Example:
  - Mapping to polynomial space, $x, y \in \mathbb{R}^2$:
    \[
    \phi(x) = \left[ \begin{array}{c}
    x_1^2 \\
    \sqrt{2}x_1x_2 \\
    x_2^2
    \end{array} \right]
    \]
  
  - Motivation: Easier to separate data in higher-dimensional space.
  - But wait – isn’t there a big problem?
    - How should we evaluate the decision function?
  
  Image source: C. Burges, 1998
Problem with High-dim. Basis Functions

- Problem
  - In order to apply the SVM, we need to evaluate the function
    \[ y(x) = w^T \phi(x) + b \]
  - Using the hyperplane, which is itself defined as
    \[ w = \sum_{n=1}^{N} a_n t_n \phi(x_n) \]

⇒ What happens if we try this for a million-dimensional feature space \( \phi(x) \)?
  - Oh-oh…
Solution: The Kernel Trick

- Important observation
  - $\phi(x)$ only appears in the form of dot products $\phi(x)^T\phi(y)$:
    \[
y(x) = w^T\phi(x) + b
    = \sum_{n=1}^{N} a_n t_n \phi(x_n)^T \phi(x) + b
    \]
  - Trick: 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)!
Back to Our Previous Example…

- 2\textsuperscript{nd} degree polynomial kernel:

\[
\phi(x)^T \phi(y) = \begin{bmatrix}
x_1^2 \\
\sqrt{2}x_1x_2 \\
x_2^2
\end{bmatrix} \cdot \begin{bmatrix}
y_1^2 \\
\sqrt{2}y_1y_2 \\
y_2^2
\end{bmatrix}
\]

\[
= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2
\]

\[
= (x^Ty)^2 =: k(x, y)
\]

- Whenever we evaluate the kernel function \(k(x,y) = (x^Ty)^2\), we implicitly compute the dot product in the higher-dimensional feature space.
SVMs with Kernels

• Using kernels
  - Applying the kernel trick is easy. Just replace every dot product by a kernel function…
    \[ x^T y \rightarrow k(x, y) \]
  - …and we’re done.
  - Instead of the raw input space, we’re now working in a higher-dimensional (potentially infinite dimensional!) space, where the data is more easily separable.

“Sounds like magic…”

• Wait – does this always work?
  - The kernel needs to define an implicit mapping to a higher-dimensional feature space \( \phi(x) \).
  - When is this the case?
Which Functions are Valid Kernels?

- Mercer’s theorem (modernized version):
  - Every positive definite symmetric function is a kernel.

- Positive definite symmetric functions correspond to a positive definite symmetric Gram matrix:

\[
K = \begin{bmatrix}
  k(x_1,x_1) & k(x_1,x_2) & k(x_1,x_3) & \cdots & k(x_1,x_n) \\
  k(x_2,x_1) & k(x_2,x_2) & k(x_2,x_3) & & k(x_2,x_n) \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  k(x_n,x_1) & k(x_n,x_2) & k(x_n,x_3) & \cdots & k(x_n,x_n)
\end{bmatrix}
\]

(positive definite = all eigenvalues are > 0)
Kernels Fulfilling Mercer’s Condition

- Polynomial kernel
  \[ k(x, y) = (x^T y + 1)^p \]

- Radial Basis Function kernel
  \[ k(x, y) = \exp \left\{ -\frac{(x - y)^2}{2\sigma^2} \right\} \]  
  e.g. Gaussian

- Hyperbolic tangent kernel
  \[ k(x, y) = \tanh(\kappa x^T y + \delta) \]  
  Actually, this was wrong in the original SVM paper...
  e.g. Sigmoid

(and many, many more…)

Slide credit: Bernt Schiele
Example: Bag of Visual Words Representation

- General framework in visual recognition
  - Create a codebook (vocabulary) of prototypical image features
  - Represent images as histograms over codebook activations
  - Compare two images by any histogram kernel, e.g. $\chi^2$ kernel

$$k_{\chi^2}(h, h') = \exp \left( -\frac{1}{\gamma} \sum_{j} \frac{(h_j - h'_j)^2}{h_j + h'_j} \right)$$

Slide adapted from Christoph Lampert
Nonlinear SVM – Dual Formulation

- SVM Dual: Maximize

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_m, x_n) \]

under the conditions

\[ 0 \cdot a_n \cdot C \]

\[ \sum_{n=1}^{N} a_n t_n = 0 \]

- Classify new data points using

\[ y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b \]
SVM Demo

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

B. Leibe
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/](http://www.kernel-machines.org/)
Summary: SVMs

• Limitations
  - How to select the right kernel?
    - Best practice guidelines are available for many applications
  - 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.
      ⇒ 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
Topics of This Lecture

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

• Nonlinear Support Vector Machines
  ➢ Nonlinear basis functions
  ➢ The Kernel trick
  ➢ Mercer’s condition
  ➢ Popular kernels

• Analysis
  ➢ Error function

• Applications
SVM – Analysis

• Traditional soft-margin formulation

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

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.

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

L₂ regularizer

“Hinge loss”

where \([x]_+ := \max\{0,x\}\).
Recap: Error Functions

\[ t_n \in \{-1, 1\} \]

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

$t_n \in \{-1, 1\}$

Sensitive to outliers!

\[ E(z_n) \]

- 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
  - Generally does not lead to good classifiers.

Image source: Bishop, 2006
Error Functions (Loss Functions)

- “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

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

- L₂ regularizer
- 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
Topics of This Lecture

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

• Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels

• Analysis
  - Error function

• Applications
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: Text Classification

• Results:

<table>
<thead>
<tr>
<th></th>
<th>Bayes</th>
<th>Rocchio</th>
<th>C4.5</th>
<th>k-NN</th>
<th>SVM (poly) degree $d = $</th>
<th>SVM (rbf) width $\gamma = $</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>earn</td>
<td>95.9</td>
<td>96.1</td>
<td>96.1</td>
<td>97.3</td>
<td>98.2</td>
<td>98.4</td>
</tr>
<tr>
<td>acq</td>
<td>91.5</td>
<td>92.1</td>
<td>85.3</td>
<td>92.0</td>
<td>92.6</td>
<td>94.6</td>
</tr>
<tr>
<td>money-fx</td>
<td>62.9</td>
<td>67.6</td>
<td>69.4</td>
<td>78.2</td>
<td>66.9</td>
<td>72.5</td>
</tr>
<tr>
<td>grain</td>
<td>72.5</td>
<td>79.5</td>
<td>89.1</td>
<td>82.2</td>
<td>91.3</td>
<td>93.1</td>
</tr>
<tr>
<td>crude</td>
<td>81.0</td>
<td>81.5</td>
<td>75.5</td>
<td>85.7</td>
<td>86.0</td>
<td>87.3</td>
</tr>
<tr>
<td>trade</td>
<td>50.0</td>
<td>77.4</td>
<td>59.2</td>
<td>77.4</td>
<td>69.2</td>
<td>75.5</td>
</tr>
<tr>
<td>interest</td>
<td>58.0</td>
<td>72.5</td>
<td>49.1</td>
<td>74.0</td>
<td>69.8</td>
<td>63.3</td>
</tr>
<tr>
<td>ship</td>
<td>78.7</td>
<td>83.1</td>
<td>80.9</td>
<td>79.2</td>
<td>82.0</td>
<td>85.4</td>
</tr>
<tr>
<td>wheat</td>
<td>60.6</td>
<td>79.4</td>
<td>85.5</td>
<td>76.6</td>
<td>83.1</td>
<td>84.5</td>
</tr>
<tr>
<td>corn</td>
<td>47.3</td>
<td>62.2</td>
<td>87.7</td>
<td>77.9</td>
<td>86.0</td>
<td>86.5</td>
</tr>
<tr>
<td>microavg.</td>
<td><strong>72.0</strong></td>
<td><strong>79.9</strong></td>
<td><strong>79.4</strong></td>
<td><strong>82.3</strong></td>
<td>84.2</td>
<td>85.1</td>
</tr>
</tbody>
</table>

combined: **86.4**
Example Application: Text Classification

- This is also how you could implement a simple spam filter…

Incoming email → Dictionary → Word activations → SVM → Mailbox, Trash
Example Application: OCR

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

• USPS benchmark
  - 2.5% error: human performance

• Different learning algorithms
  - 16.2% error: Decision tree (C4.5)
  - 5.9% error: (best) 2-layer Neural Network
  - 5.1% error: LeNet 1 – (massively hand-tuned) 5-layer network

• Different SVMs
  - 4.0% error: Polynomial kernel (p=3, 274 support vectors)
  - 4.1% error: Gaussian kernel  (σ=0.3, 291 support vectors)
Example Application: OCR

- Results
  - Almost no overfitting with higher-degree kernels.

<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>$\approx 33000$</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.

[Dalal & Triggs, CVPR 2005]
Example Application: Pedestrian Detection

N. Dalal, B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005
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)
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).

  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/

• A more in-depth introduction to SVMs is available in the following tutorial: