Machine Learning - Lecture 7

Statistical Learning Theory & SVMs

13.11.2013

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

leibe@vision.rwth-aachen.de

Many slides adapted from B. Schiele
Course Outline

• Fundamentals (2 weeks)
  - Bayes Decision Theory
  - Probability Density Estimation

• Discriminative Approaches (5 weeks)
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns

• Generative Models (4 weeks)
  - Bayesian Networks
  - Markov Random Fields
Recap: Classification as Dim. Reduction

- **Classification as dimensionality reduction**
  - Interpret linear classification as a projection onto a lower-dim. space.
    
    \[ y = w^T x \]

  \[ \Rightarrow \] Learning problem: Try to find the projection vector \( w \) that maximizes class separation.

*Image source: C.M. Bishop, 2006*
Recap: Fisher’s Linear Discriminant Analysis

- Maximize distance between classes
- Minimize distance within a class

**Criterion:**

\[ J(w) = \frac{w^T S_B w}{w^T S_W w} \]

- \( S_B \) ... between-class scatter matrix
- \( S_W \) ... within-class scatter matrix

The optimal solution for \( w \) can be obtained as:

\[ w \propto S_W^{-1}(m_2 - m_1) \]

**Classification function:**

\[ y(x) = w^T x + w_0 \]

where \( w_0 = -w^T m \)
Recap: Probabilistic Discriminative Models

- Consider models of the form

\[ p(C_1 | \phi) = y(\phi) = \sigma(w^T \phi) \]

with

\[ p(C_2 | \phi) = 1 - p(C_1 | \phi) \]

- This model is called logistic regression.

- Properties
  - Probabilistic interpretation
  - But discriminative method: only focus on decision hyperplane
  - Advantageous for high-dimensional spaces, requires less parameters than explicitly modeling \( p(\phi | C_k) \) and \( p(C_k) \).
Recap: Logistic Regression

- Let’s consider a data set \{\phi_n,t_n\} with \(n = 1,\ldots,N\), where \(\phi_n = \phi(x_n)\) and \(t_n \in \{0, 1\}\), \(t = (t_1, \ldots, t_N)^T\).

- With \(y_n = p(C_1|\phi_n)\), we can write the likelihood as
  \[
p(t|w) = \prod_{n=1}^{N} y_n^{t_n} \{1 - y_n\}^{1-t_n}
  \]

- Define the error function as the negative log-likelihood
  \[
  E(w) = -\ln p(t|w)
  \]
  \[
  = - \sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}
  \]
  ➢ This is the so-called cross-entropy error function.
Recap: Iterative Methods for Estimation

- **Gradient Descent (1\textsuperscript{st} order)**
  \[ w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E(w)\big|_{w^{(\tau)}} \]
  - Simple and general
  - Relatively slow to converge, has problems with some functions

- **Newton-Raphson (2\textsuperscript{nd} order)**
  \[ w^{(\tau+1)} = w^{(\tau)} - \eta H^{-1} \nabla E(w)\big|_{w^{(\tau)}} \]
  where \( H = \nabla \nabla E(w) \) is the Hessian matrix, i.e. the matrix of second derivatives.
  - Local quadratic approximation to the target function
  - Faster convergence
Recap: Iteratively Reweighted Least Squares

- Update equations

\[
\begin{align*}
    w^{(\tau+1)} &= w^{(\tau)} - (\Phi^T R \Phi)^{-1} \Phi^T (y - t) \\
    &= (\Phi^T R \Phi)^{-1} \left\{ \Phi^T R \Phi w^{(\tau)} - \Phi^T (y - t) \right\} \\
    &= (\Phi^T R \Phi)^{-1} \Phi^T R z
\end{align*}
\]

with \( z = \Phi w^{(\tau)} - R^{-1} (y - t) \)

- Very similar form to pseudo-inverse (normal equations)
  - But now with non-constant weighing matrix \( R \) (depends on \( w \)).
  - Need to apply normal equations iteratively.
  \[ \Rightarrow \text{Iteratively Reweighted Least-Squares (IRLS) } \]
A Note on 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.
  - \( \Rightarrow \) We cannot minimize it by gradient descent.

\[ z_n = t_n y(x_n) \]

\[ E(z_n) \]

Image source: Bishop, 2006
A Note on Error Functions

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

Sensitive to outliers!

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
  $\Rightarrow$ Generally does not lead to good classifiers.

Ideal misclassification error
Squashed error

Penalizes “too correct” data points!

Image source: Bishop, 2006
A Note on Error Functions

Cross-Entropy Error

- Minimizer of this error is given by posterior class probabilities.
- Concave error function, unique minimum exists.
- Robust to outliers, error increases only roughly linearly
- But no closed-form solution, requires iterative estimation.

Ideal misclassification error
Squared error
Cross-entropy error

$z_n = t_n y(x_n)$

$E(z_n)$

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

Robust to outliers!
Overview: Error Functions

• **Ideal Misclassification Error**
  - This is what we would like to optimize.
  - But cannot compute gradients here.

• **Quadratic Error**
  - Easy to optimize, closed-form solutions exist.
  - But not robust to outliers.

• **Cross-Entropy Error**
  - Minimizer of this error is given by posterior class probabilities.
  - Concave error function, unique minimum exists.
  - But no closed-form solution, requires iterative estimation.

⇒ *Analysis tool to compare classification approaches*
Topics of This Lecture

• Statistical Learning Theory
  - Generalization and overfitting
  - Empirical and actual risk
  - VC dimension
  - Empirical Risk Minimization
  - Structural Risk Minimization

• Linear Support Vector Machines (SVMs)
  - Linearly separable case
  - Lagrange multipliers
  - Lagrangian (primal) formulation
  - Dual formulation
  - Discussion
Generalization and Overfitting

- Goal: predict class labels of new observations
  - Train classification model on limited training set.
  - The further we optimize the model parameters, the more the training error will decrease.
  - However, at some point the test error will go up again.
  ⇒ Overfitting to the training set!

Image source: B. Schiele
Example: Linearly Separable Data

- Overfitting is often a problem with linearly separable data
  - Which of the many possible decision boundaries is correct?
  - All of them have zero error on the training set...
  - However, they will most likely result in different predictions on novel test data.
    ⇒ Different generalization performance

- How to select the classifier with the best generalization performance?
A Broader View on Statistical Learning

- **Formal treatment:** Statistical Learning Theory

- **Supervised learning**
  - **Environment:** assumed stationary.
  - I.e. the data $x$ have an unknown but fixed probability density $p_X(x)$
  - **Teacher:** specifies for each data point $x$ the desired classification $y$ (where $y$ may be subject to noise).
    $$ y = g(x, \nu) \quad \text{with noise } \nu $$
  - **Learning machine:** represented by class of functions, which produce for each $x$ an output $y$:
    $$ y = f(x; \alpha) \quad \text{with parameters } \alpha $$

Slide credit: Bernt Schiele
Statistical Learning Theory

- Supervised learning (from the learning machine’s view)
  - Selection of a specific function \( f(x; \alpha) \)
  - Given: training examples \( \{(x_i, y_i)\}_{i=1}^{N} \)
  - Goal: the desired response \( y \) shall be approximated optimally.

- Measuring the optimality
  - Loss function
    \[
    L(y, f(x; \alpha))
    \]
  - Example: quadratic loss
    \[
    L(y, f(x; \alpha)) = (y - f(x; \alpha))^2
    \]
Risk

- Measuring the “optimality”
  - Measure the optimality by the **risk** (= expected loss).
  - Difficulty: how should the risk be estimated?

- Practical way
  - **Empirical risk** (measured on the training/validation set)
    \[
    R_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; \alpha))
    \]
  - Example: quadratic loss function
    \[
    R_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i; \alpha))^2
    \]
Risk

- However, what we’re really interested in is
  - Actual risk = Expected risk
    \[ R(\alpha) = \int L(y, f(x; \alpha))dP_{X,Y}(x, y) \]
  - \( P_{X,Y}(x, y) \) is the probability distribution of \((x,y)\).
  - \( P_{X,Y}(x, y) \) is fixed, but typically unknown.
    \( \Rightarrow \) In general, we can’t compute the actual risk directly!
  - The expected risk is the expectation of the error on all data.
  - I.e., it is the expected value of the generalization error.

Slide credit: Bernt Schiele
Summary: Risk

- **Actual risk**
  - Advantage: measure for the generalization ability
  - Disadvantage: in general, we don’t know $P_{X,Y}(x, y)$

- **Empirical risk**
  - Disadvantage: no direct measure of the generalization ability
  - Advantage: does not depend on $P_{X,Y}(x, y)$
  - We typically know learning algorithms which minimize the empirical risk.

⇒ **Strong interest in connection between both types of risk**

Slide credit: Bernt Schiele
Statistical Learning Theory

• Idea
  - Compute an upper bound on the actual risk based on the empirical risk
    \[ R(\alpha) \cdot R_{emp}(\alpha) + \epsilon(N, p^*, h) \]
  - where
    - \( N \): number of training examples
    - \( p^* \): probability that the bound is correct
    - \( h \): capacity of the learning machine (“VC-dimension”)

• Side note:
  - (This idea of specifying a bound that only holds with a certain probability is explored in a branch of learning theory called “Probably Approximately Correct” or PAC Learning).
VC Dimension

- Vapnik-Chervonenkis dimension
  - Measure for the capacity of a learning machine.

- Formal definition:
  - If a given set of \( \ell \) points can be labeled in all possible \( 2^\ell \) ways, and for each labeling, a member of the set \( \{f(\alpha)\} \) can be found which correctly assigns those labels, we say that the set of points is **shattered** by the set of functions.
  
  The **VC dimension** for the set of functions \( \{f(\alpha)\} \) is defined as the maximum number of training points that can be shattered by \( \{f(\alpha)\} \).
VC Dimension

• Interpretation as a two-player game
  ➢ Opponent’s turn: He says a number $N$.
  ➢ Our turn: We specify a set of $N$ points $\{x_1, \ldots, x_N\}$.
  ➢ Opponent’s turn: He gives us a labeling $\{x_1, \ldots, x_N\} \in \{0, 1\}^N$
  ➢ Our turn: We specify a function $f(\alpha)$ which correctly classifies all $N$ points.

⇒ If we can do that for all $2^N$ possible labelings, then the VC dimension is at least $N$. 
VC Dimension

• Example
  - The VC dimension of all oriented lines in $\mathbb{R}^2$ is 3.
    1. Shattering 3 points with an oriented line:
       ![Diagram of 3 points shatterable by oriented lines]

      2. More difficult to show: it is not possible to shatter 4 points (XOR)...

  - More general: the VC dimension of all hyperplanes in $\mathbb{R}^n$ is $n+1$.  

VC Dimension

- Intuitive feeling (unfortunately wrong)
  - The VC dimension has a direct connection with the number of parameters.

- Counterexample

  \[ f(x; \alpha) = g(\sin(\alpha x)) \]
  \[
  g(x) = \begin{cases} 
  1, & x > 0 \\
  -1, & x \cdot 0 
  \end{cases}
  \]
  
  - Just a single parameter \( \alpha \).
  - Infinite VC dimension
    - Proof: Choose \( x_i = 10^{-i}, i = 1, \ldots, \ell \)
    \[
    \alpha = \pi \left( 1 + \sum_{i=1}^{\ell} \frac{(1 - y_i)10^i}{2} \right)
    \]

Slide adapted from Bernt Schiele
Upper Bound on the Risk

- Important result (Vapnik 1979, 1995)
  - With probability $(1-\eta)$, the following bound holds
    \[ R(\alpha) \cdot R_{emp}(\alpha) + \sqrt{\frac{h(\log(2N/h) + 1) - \log(\eta/4)}{N}} \]
    “VC confidence”
  - This bound is independent of $P_{X,Y}(x,y)$!
  - Typically, we cannot compute the left-hand side (the actual risk)
  - If we know $h$ (the VC dimension), we can however easily compute the risk bound
    \[ R(\alpha) \cdot R_{emp}(\alpha) + \epsilon(N, p^*, h) \]
Upper Bound on the Risk

\[ \varepsilon(N, p^*, h) \]

\[ R_{emp}(\alpha) \]
Structural Risk Minimization

- How can we implement this?

\[ R(\alpha) \cdot R_{emp}(\alpha) + \epsilon(N, p^*, h) \]

- Classic approach
  - Keep \( \epsilon(N, p^*, h) \) constant and minimize \( R_{emp}(\alpha) \).
  - \( \epsilon(N, p^*, h) \) can be kept constant by controlling the model parameters.

- Support Vector Machines (SVMs)
  - Keep \( R_{emp}(\alpha) \) constant and minimize \( \epsilon(N, p^*, h) \).
  - In fact: \( R_{emp}(\alpha) = 0 \) for separable data.
  - Control \( \epsilon(N, p^*, h) \) by adapting the VC dimension (controlling the “capacity” of the classifier).
Topics of This Lecture

• Statistical Learning Theory
  - Generalization and overfitting
  - Empirical and actual risk
  - VC dimension
  - Empirical Risk Minimization
  - Structural Risk Minimization

• Linear Support Vector Machines (SVMs)
  - Linearly separable case
  - Lagrange multipliers
  - Lagrangian (primal) formulation
  - Dual formulation
  - Discussion
Revisiting Our Previous Example...

- **How to select the classifier with the best generalization performance?**
  - Intuitively, we would like to select the classifier which leaves maximal “safety room” for future data points.
  - This can be obtained by maximizing the **margin** between positive and negative data points.
  - It can be shown that the larger the margin, the lower the corresponding classifier’s VC dimension.

- **The SVM takes up this idea**
  - It searches for the classifier with maximum margin.
  - Formulation as a convex optimization problem
  - Possible to find the globally optimal solution!
Support Vector Machine (SVM)

- Let’s first consider linearly separable data
  - \( N \) training data points \( \{(x_i, y_i)\}_{i=1}^N \) \( x_i \in \mathbb{R}^d \)
  - Target values \( t_i \in \{-1, 1\} \)
  - Hyperplane separating the data

\[ w^T x + b = 0 \]
Support Vector Machine (SVM)

- **Margin of the hyperplane:** \( d_- + d_+ \)
  - \( d_+ \): distance to nearest pos. training example
  - \( d_- \): distance to nearest neg. training example

- We can always choose \( w, b \) such that \( d_- = d_+ = \frac{1}{\|w\|} \).

Slide adapted from Bernt Schiele

Image source: C. Burges, 1998
Support Vector Machine (SVM)

- Since the data is linearly separable, there exists a hyperplane with
  \[ w^T x_n + b \geq +1 \quad \text{for} \quad t_n = +1 \]
  \[ w^T x_n + b \leq -1 \quad \text{for} \quad t_n = -1 \]

- Combined in one equation, this can be written as
  \[ t_n(w^T x_n + b) \geq 1 \quad \forall n \]

⇒ Canonical representation of the decision hyperplane.

- The equation will hold exactly for the points on the margin
  \[ t_n(w^T x_n + b) = 1 \]

- By definition, there will always be at least one such point.
Support Vector Machine (SVM)

• We can choose $w$ such that
  \[ w^T x_n + b = +1 \quad \text{for one} \quad t_n = +1 \]
  \[ w^T x_n + b = -1 \quad \text{for one} \quad t_n = -1 \]

• The distance between those two hyperplanes is then the margin
  \[ d_- = d_+ = \frac{1}{\|w\|} \]
  \[ d_- + d_+ = \frac{2}{\|w\|} \]

$\Rightarrow$ We can find the hyperplane with maximal margin by minimizing $\|w\|^2$. 
Support Vector Machine (SVM)

- **Optimization problem**
  - Find the hyperplane satisfying
    \[
    \arg\min_{w,b} \frac{1}{2} \|w\|^2
    \]
    under the constraints
    \[
    t_n (w^T x_n + b) \geq 1 \quad \forall n
    \]
  - 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 $K(x)$ subject to constraints $f(x) = 0$.
- Example: we want to get as close as possible, but there is a fence.
- How should we move?
  
  \[
  f(x) = \begin{cases} 
  0 & f(x) > 0 \\
  < 0 & f(x) < 0 
  \end{cases}
  \]

- We want to maximize $\nabla K$.
- But we can only move parallel to the fence, i.e. along
  
  \[
  \nabla_{\parallel} K = \nabla K + \lambda \nabla f
  \]

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?

$$f(x) = 0 \quad f(x) > 0$$

⇒ Optimize

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

$$\frac{\partial L}{\partial x} = \nabla K = 0$$

$$\frac{\partial L}{\partial \lambda} = f(x) = 0$$
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 $\max_{x, \lambda} L(x, \lambda) = K(x) + \lambda f(x)$

- **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
    $\Rightarrow \lambda f(x) = 0$
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")

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

  - I.e., find $w$, $b$, and $a$ such that

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n = 0$$
$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N} a_n t_n x_n$$
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
    
    \[ a_n \geq 0 \]
    
    \[ t_n y(x_n) - 1 \geq 0 \]
    
    \[ a_n \left\{ t_n y(x_n) - 1 \right\} = 0 \]

\[
\text{KKT:} \quad \begin{align*}
\lambda & \geq 0 \\
 f(x) & \geq 0 \\
 \lambda f(x) & = 0
\end{align*}
\]
**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
    \[
    a_n (t_n (w^T x_n + b) - 1) = 0
    \]
  - This implies that \(a_n > 0\) only for training data points for which
    \[
    (t_n (w^T x_n + b) - 1) = 0
    \]
  - \(\Rightarrow\) *Only some of the data points actually influence the decision boundary!*

---

KKT:
\[
\begin{align*}
\lambda f(x) &= 0 \\
(1) t_n (w^T x_n + b) - 1 &= 0
\end{align*}
\]
**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.

$\Rightarrow$ Robustness to “too correct” points!

Image source: C. Burges, 1998
SVM - Solution (Part 2)

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

KKT:
\[
f(x) \geq 0
\]
SVM - Discussion (Part 1)

• Linear SVM
  - Linear classifier
  - Approximative implementation of the SRM principle.
  - In case of separable data, the SVM produces an empirical risk of zero with minimal value of the VC confidence (i.e. a classifier minimizing the upper bound on the actual risk).
  - SVMs thus 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”)

Slide adapted from Bernt Schiele
SVM - Dual Formulation

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

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

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

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

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

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

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

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

\[ = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_m^T x_n) + \sum_{n=1}^{N} a_n \]
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)
        - runtime between $O(N)$ and $O(N^2)$.
  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 $\phi(x)$.
     ⇒ Next lecture...
Historical Importance

• 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)

*(We will see those in the next 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.

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

⇒ Next Lecture...
References and Further Reading

• More information on SVMs can be found in Chapter 7.1 of Bishop’s book.

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

• Additional information about Statistical Learning Theory and a more in-depth introduction to SVMs are available in the following tutorial: