Machine Learning – Lecture 7

Linear Support Vector Machines

06.11.2017

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de
leibe@vision.rwth-aachen.de
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
Recap: Generalized Linear Models

• Generalized linear model

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

- \( g(\cdot) \) is called an activation function and may be nonlinear.
- The decision surfaces correspond to

\[ y(x) = \text{const.} \iff w^T x + w_0 = \text{const.} \]

- If \( g \) is monotonous (which is typically the case), the resulting decision boundaries are still linear functions of \( x \).

• Advantages of the non-linearity

- Can be used to bound the influence of outliers and “too correct” data points.
- When using a sigmoid for \( g(\cdot) \), we can interpret the \( y(x) \) as posterior probabilities.

\[ g(a) \equiv \frac{1}{1 + \exp(-a)} \]
Recap: Extension to Nonlinear Basis Fcts.

- **Generalization**
  - Transform vector $\mathbf{x}$ with $M$ nonlinear basis functions $\phi_j(\mathbf{x})$:
  $$y_k(\mathbf{x}) = \sum_{j=1}^{M} w_{kj} \phi_j(\mathbf{x}) + w_{k0}$$

- **Advantages**
  - Transformation allows non-linear decision boundaries.
  - By choosing the right $\phi_j$, every continuous function can (in principle) be approximated with arbitrary accuracy.

- **Disadvantage**
  - The error function can in general no longer be minimized in closed form.
  $\Rightarrow$ Minimization with Gradient Descent

B. Leibe
Recap: Basis Functions

• Generally, we consider models of the following form

\[ y_k(x) = \sum_{j=0}^{M} w_{k,j} \phi_j(x) = w^T \phi(x) \]

- where \( \phi_j(x) \) are known as \textit{basis functions}.
- In the simplest case, we use linear basis functions: \( \phi_d(x) = x_d \).

• Other popular basis functions

![Polynomial](image1.png)
![Gaussian](image2.png)
![Sigmoid](image3.png)
Recap: Iterative Methods for Estimation

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

• Newton-Raphson (2\textsuperscript{nd} order)
  \[ w^{(\tau+1)} = w^{(\tau)} - \eta H^{-1} \nabla E(w) \bigg|_{w^{(\tau)}} \]
  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: Gradient Descent

• Iterative minimization
  - Start with an initial guess for the parameter values \( w_{kj}^{(0)} \).
  - Move towards a (local) minimum by following the gradient.

• Basic strategies
  - “Batch learning”
    \[
    w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(w)}{\partial w_{kj}} \right|_{w^{(\tau)}}
    \]
  - “Sequential updating”
    \[
    w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E_n(w)}{\partial w_{kj}} \right|_{w^{(\tau)}}
    \]

where
\[
E(w) = \sum_{n=1}^{N} E_n(w)
\]
Recap: Gradient Descent

• Example: Quadratic error function

\[ E(\mathbf{w}) = \sum_{n=1}^{N} (y(x_n; \mathbf{w}) - t_n)^2 \]

• Sequential updating leads to delta rule (=LMS rule)

\[
\mathbf{w}_{kj}^{(\tau+1)} = \mathbf{w}_{kj}^{(\tau)} - \eta (y_k(x_n; \mathbf{w}) - t_{kn}) \phi_j(x_n)
\]

\[ = \mathbf{w}_{kj}^{(\tau)} - \eta \delta_{kn} \phi_j(x_n) \]

➤ where

\[ \delta_{kn} = y_k(x_n; \mathbf{w}) - t_{kn} \]

⇒ Simply feed back the input data point, weighted by the classification error.

Slide adapted from Bernt Schiele
Recap: Gradient Descent

- Cases with differentiable, non-linear activation function

\[ y_k(x) = g(a_k) = g \left( \sum_{j=0}^{M} w_{ki} \phi_j(x_n) \right) \]

- Gradient descent (again with quadratic error function)

\[
\frac{\partial E_n(w)}{\partial w_{kj}} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn}) \phi_j(x_n)
\]

\[ w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \delta_{kn} \phi_j(x_n) \]

\[ \delta_{kn} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn}) \]

Slide adapted from Bernt Schiele
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: Iteratively Reweighted Least Squares

• Update equations

\[ w^{(\tau+1)} = w^{(\tau)} - (\Phi^T R \Phi)^{-1} \Phi^T (y - t) \]

\[ = (\Phi^T R \Phi)^{-1} \left\{ \Phi^T R \Phi w^{(\tau)} - \Phi^T (y - t) \right\} \]

\[ = (\Phi^T R \Phi)^{-1} \Phi^T R z \]

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.

⇒ Iteratively Reweighted Least-Squares (IRLS)
Topics of This Lecture

- **Softmax Regression**
  - Multi-class generalization
  - Gradient descent solution

- **Note on Error Functions**
  - Ideal error function
  - Quadratic error
  - Cross-entropy error

- **Linear Support Vector Machines**
  - Lagrangian (primal) formulation
  - Dual formulation
  - Discussion
Softmax Regression

- Multi-class generalization of logistic regression
  - In logistic regression, we assumed binary labels \( t_n \in \{0, 1\} \).
  - Softmax generalizes this to \( K \) values in 1-of-\( K \) notation.

\[
y(x; w) = \begin{bmatrix}
P(y = 1|x; w) \\
P(y = 2|x; w) \\
\vdots \\
P(y = K|x; w)
\end{bmatrix}
= \frac{1}{\sum_{j=1}^{K} \exp(w_j^\top x)} \begin{bmatrix}
\exp(w_1^\top x) \\
\exp(w_2^\top x) \\
\vdots \\
\exp(w_K^\top x)
\end{bmatrix}
\]

- This uses the softmax function

\[
\frac{\exp(a_{k})}{\sum_{j} \exp(a_{j})}
\]

- Note: the resulting distribution is normalized.
Softmax Regression Cost Function

- Logistic regression
  - Alternative way of writing the cost function with indicator function $\mathbb{I}(\cdot)$

\[
E(w) = - \sum_{n=1}^{N} \left\{ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right\}
= - \sum_{n=1}^{N} \sum_{k=0}^{1} \left\{ \mathbb{I}(t_n = k) \ln P(y_n = k|x_n; w) \right\}
\]

- Softmax regression
  - Generalization to $K$ classes using indicator functions.

\[
E(w) = - \sum_{n=1}^{N} \sum_{k=1}^{K} \left\{ \mathbb{I}(t_n = k) \ln \frac{\exp(w_k^T x)}{\sum_{j=1}^{K} \exp(w_j^T x)} \right\}
\]
Optimization

• Again, no closed-form solution is available
  ➢ Resort again to Gradient Descent
  ➢ Gradient

\[
\nabla_{\mathbf{w}_k} E(\mathbf{w}) = - \sum_{n=1}^{N} \left[ \mathbb{I}(t_n = k) \ln P(y_n = k|\mathbf{x}_n; \mathbf{w}) \right]
\]

• Note
  ➢ \( \nabla_{\mathbf{w}_k} E(\mathbf{w}) \) is itself a vector of partial derivatives for the different components of \( \mathbf{w}_k \).
  ➢ We can now plug this into a standard optimization package.
Topics of This Lecture

• Softmax Regression
  ➢ Multi-class generalization
  ➢ Gradient descent solution

• Note on Error Functions
  ➢ Ideal error function
  ➢ Quadratic error
  ➢ Cross-entropy error

• Linear Support Vector Machines
  ➢ Lagrangian (primal) formulation
  ➢ Dual formulation
  ➢ Discussion
Note on Error Functions

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

- **Ideal misclassification error function (black)**
  - This is what we want to approximate (error = number of misclassifications)
  - Unfortunately, it is not differentiable.
  - The gradient is zero for misclassified points.
  - \( \Rightarrow \) We cannot minimize it by gradient descent.

Image source: Bishop, 2006
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
  ⇒ Generally does not lead to good classifiers.

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

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

Robust to outliers!

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

\[ z_n = t_n y(x_n) \]

Image source: Bishop, 2006
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

• Softmax Regression
  - Multi-class generalization
  - Gradient descent solution

• Note on Error Functions
  - Ideal error function
  - Quadratic error
  - Cross-entropy error

• Linear Support Vector Machines
  - 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.
  \( \Rightarrow \text{Overfitting to the training set!} \)
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?
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 (capacity for overfitting).

- The SVM takes up this idea
  - It searches for the classifier with maximum margin.
  - Formulation as a convex optimization problem
    \[ \Rightarrow \] 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

\[
f(x; y) = \sum_{i=1}^{N} w_i y_i (x - x_i)
\]

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

Slide credit: Bernt Schiele
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 \( d_- = d_+ = \frac{1}{\|w\|} \)

Image source: C. Burges, 1998

Slide adapted from Bernt Schiele
Support Vector Machine (SVM)

• Since the data is linearly separable, there exists a hyperplane with

\[ w^T x_n + b \geq +1 \quad \text{for} \quad t_n = +1 \]

\[ w^T x_n + b \cdot -1 \quad \text{for} \quad t_n = -1 \]

• 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
  \[
  \arg \min_{w, b} \frac{1}{2} ||w||^2 \\
  \text{under the constraints}
  \]
  \[
  t_n(w^T x_n + b) \geq 1 \quad \forall n
  \]
  - 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?

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

  - 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?

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

$\Rightarrow$ Optimize

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

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

$$\frac{\partial L}{\partial \lambda} = f(x) = 0$$
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 $\|\mathbf{w}\|^2$ under the constraints
  $$t_n (\mathbf{w}^T \mathbf{x}_n + b) - 1 \geq 0 \quad \forall n$$

- Lagrangian formulation
  
  - Introduce positive Lagrange multipliers: $a_n \geq 0 \quad \forall n$
  
  - Minimize Lagrangian ("primal form")
    $$L(w, b, a) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \{t_n (\mathbf{w}^T \mathbf{x}_n + b) - 1\}$$

  - I.e., find $\mathbf{w}$, $b$, and $a$ such that
    $$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n = 0$$
    $$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$
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

  \[ a_n \geq 0 \]

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

  \[ a_n \left( t_n y(x_n) - 1 \right) = 0 \]

  \[ KKT: \]

  \[ \lambda \geq 0 \]

  \[ f(x) \geq 0 \]

  \[ \lambda f(x) = 0 \]

B. Leibe
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
    \[ a_n \left( t_n (w^T x_n + b) - 1 \right) = 0 \]
  - This implies that \( a_n > 0 \) only for training data points for which
    \[ \left( t_n (w^T x_n + b) - 1 \right) = 0 \]
  \[ \Rightarrow \text{Only some of the data points actually influence the decision boundary!} \]
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.

$\Rightarrow$ 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 $x_n$ 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 $t_n^2 = 1$ 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
  - 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 $\mathcal{O}(M^3)$.
  - Here: $D$ variables ⇒ $\mathcal{O}(D^3)$
  - 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

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

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

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

$$\frac{\partial L_p}{\partial b} = 0$$

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

\[
\frac{\partial L_p}{\partial w} = 0
\]
SVM – Dual Formulation

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

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

\[ \frac{1}{2} \mathbf{w}^T \mathbf{w} = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^T \mathbf{x}_n) \]

- Inserting this, we get the Wolfe dual

\[ L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^T \mathbf{x}_n) \]
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:

\[ w = \sum_{n=1}^{N_S} a_n t_n x_n \]

Slide adapted from Bernt Schiele
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 in the 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: