Machine Learning - Lecture 8

Linear Support Vector Machines

12.05.2015

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/
leibe@vision.rwth-aachen.de
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: 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.
  \[ \Rightarrow \textit{Overfitting to the training set!} \]
Recap: Risk

- **Empirical risk**
  - Measured on the training/validation set
  \[
  R_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; \alpha))
  \]

- **Actual risk (= Expected risk)**
  - Expectation of the error on all data.
  \[
  R(\alpha) = \int L(y_i, f(x; \alpha)) dP_{X,Y}(x, y)
  \]

  - \(P_{X,Y}(x, y)\) is the probability distribution of \((x,y)\).
  It is fixed, but typically unknown.

  \(\Rightarrow\) In general, we can’t compute the actual risk directly!

Slide adapted from Bernt Schiele
Recap: 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")

Slide adapted from Bernt Schiele
Recap: 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 \( \{ f(\alpha) \} \).
Recap: Upper Bound on the Risk

• Important result (Vapnik 1979, 1995)
  ➢ With probability $(1-\eta)$, the following bound holds

\[
R(\alpha) \cdot R_{emp}(\alpha) + \sqrt{\frac{h(\log(2N/h) + 1)}{N}} - \log(\eta/4)
\]

  “VC confidence”

➢ This bound is independent of $P_{X,Y}(x,y)$!
➢ If we know $h$ (the VC dimension), we can easily compute the risk bound

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

• How can we implement Structural Risk Minimization?

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

• 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).

Slide credit: Bernt Schiele
Topics of This Lecture

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

- **Linearly non-separable case**
  - Soft-margin classification
  - Updated formulation

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

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

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

\[ \Rightarrow \] 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.

Slide adapted from Bernt Schiele
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
  \[ d_- = d_+ = \frac{1}{||w||} \]
  \[ d_- + d_+ = \frac{2}{||w||} \]

⇒ We can find the hyperplane with maximal margin by minimizing $||w||^2$. 

Slide credit: Bernt Schiele
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 $K(x)$ subject to constraints $f(x) = 0$.
  - Example: we want to get as close as possible, but there is a fence.
  - How should we move?
    
    $f(x) = 0$  
    $f(x) > 0$  
    $f(x) < 0$

  - 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?

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

⇒ Optimize

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

\[
\frac{\partial L}{\partial x} = \nabla_{\parallel} K = 0
\]

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

- **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

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} a_n t_n = 0 \quad \frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{n=1}^{N} a_n t_n x_n$$
SVM - Lagrangian Formulation

- Lagrangian primal form

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

- The solution of \( L_p \) needs to fulfill the KKT conditions
  
  - Necessary and sufficient conditions

\[
\begin{align*}
  a_n &\geq 0 \\
  t_n y(x_n) - 1 &\geq 0 \\
  a_n \{ t_n y(x_n) - 1 \} &= 0
\end{align*}
\]

KKT:

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

\[ \Rightarrow \text{Robustness to "too correct" points!} \]
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
  - 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

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

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

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

\[
\frac{\partial L_p}{\partial b} = 0
\]
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)
        - Effective 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 $\phi(x)$.
     ⇒ We’ll see that in a few minutes...
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:
    
    \[ \mathbf{w}^T \mathbf{x}_n + b \geq +1 \quad \text{for} \quad t_n = +1 \]
    
    \[ \mathbf{w}^T \mathbf{x}_n + b \cdot -1 \leq -1 \quad \text{for} \quad t_n = -1 \]

  - Instead use:
    
    \[ \mathbf{w}^T \mathbf{x}_n + b \geq +1 - \xi_n \quad \text{for} \quad t_n = +1 \]
    
    \[ \mathbf{w}^T \mathbf{x}_n + b \cdot -1 + \xi_n \leq -1 \quad \text{for} \quad t_n = -1 \]

  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

- **Non-separable data**
  - Minimize

\[
\begin{align*}
\frac{1}{2} & \|w\|^2 \\
\frac{1}{2} & \|w\|^2 + C \sum_{n=1}^{N} \xi_n
\end{align*}
\]

Trade-off parameter!
**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**

\[
\begin{align*}
  a_n &\geq 0 & \mu_n &\geq 0 \\
  t_n y(x_n) - 1 + \xi_n &\geq 0 & \xi_n &\geq 0 \\
a_n (t_n y(x_n) - 1 + \xi_n) &= 0 & \mu_n \xi_n &= 0
\end{align*}
\]

KKT:
\[
\begin{align*}
  \lambda &\geq 0 \\
  f(x) &\geq 0 \\
  \lambda f(x) &= 0
\end{align*}
\]
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
  \[ \Rightarrow \text{Solve as before... (more on that later)} \]

This is all that changed!
SVM - New Solution

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

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

![Graph showing support vectors for face detection](Image source: E. Osuna, F. Girosi, 1997)
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.
    \( \Rightarrow \) 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
Another Example

- Non-separable by a hyperplane in 2D

Slide credit: Bill Freeman
Another Example

- Separable by a surface in 3D

Slide credit: Bill Freeman
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:

$$\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$$
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, \( x, y \in \mathbb{R}^2 \):

\[
\phi(x) = \begin{bmatrix}
    x_1^2 \\
    \sqrt{2} x_1 x_2 \\
    x_2^2
\end{bmatrix}
\]

- **Motivation:** Easier to separate data in higher-dimensional space.
- **But wait - isn’t there a big problem?**
  - How should we evaluate the decision function?
Problem with High-dim. Basis Functions

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

⇒ 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...

- **2nd degree polynomial kernel:**

\[
\phi(x)^T \phi(y) = \begin{bmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ \sqrt{2} y_1 y_2 \\ y_2^2 \end{bmatrix} = x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2 = (x^T y)^2 =: k(x, y)
\]

Whenever we evaluate the kernel function \( k(x, y) = (x^T y)^2 \), 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?

B. Leibe
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:

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

(positive definite = all eigenvalues are > 0)
Recap: 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**
  \( \boxed{\text{Actually, this was wrong in the original SVM paper...}} \)
  \[ k(x, y) = \tanh(\kappa x^T y + \delta) \]
  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

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

\[ y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b \]
VC Dimension for Polynomial Kernel

- Polynomial kernel of degree $p$:
  \[ k(x, y) = (x^T y)^p \]

  - Dimensionality of $\mathcal{H}$:
    \[ \binom{D + p - 1}{p} \]

  - Example:
    \[
    \begin{align*}
    D &= 16 \times 16 = 256 \\
    p &= 4 \\
    \dim(\mathcal{H}) &= 183.181.376
    \end{align*}
    \]

  - The hyperplane in $\mathcal{H}$ then has VC-dimension
    \[ \dim(\mathcal{H}) + 1 \]
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, $\mathcal{H}$ 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 $\mathcal{H}$ then has VC-dimension

$$\dim(\mathcal{H}) + 1 = \infty$$
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 $\mathcal{H}$.

- 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
    \[
    h \leq \min \left\{ \left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0 \right\} + 1
    \]
    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.

Slide credit: Raymond Mooney
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?
    - Still something of a black art...
  - 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

• Linear Support Vector Machines (Recap)
  ➢ Lagrangian (primal) formulation
  ➢ Dual formulation
  ➢ Discussion

• Linearly non-separable case
  ➢ Soft-margin classification
  ➢ Updated formulation

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

• Applications
Example Application: Text Classification

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

  ![Diagram](image)

• 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 $d = 1$</th>
<th>SVM (poly) degree $d = 2$</th>
<th>SVM (poly) degree $d = 3$</th>
<th>SVM (poly) degree $d = 4$</th>
<th>SVM (poly) degree $d = 5$</th>
<th>SVM (poly) degree $d = 0.6$</th>
<th>SVM (poly) degree $d = 0.8$</th>
<th>SVM (poly) degree $d = 1.0$</th>
<th>SVM (poly) degree $d = 1.2$</th>
</tr>
</thead>
<tbody>
<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>
<td>98.5</td>
<td>98.4</td>
<td>98.3</td>
<td>98.5</td>
<td>98.5</td>
<td>98.4</td>
<td>98.3</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>
<td>95.2</td>
<td>95.2</td>
<td>95.3</td>
<td>95.0</td>
<td>95.3</td>
<td>95.3</td>
<td>95.4</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>
<td>75.4</td>
<td>74.9</td>
<td>76.2</td>
<td>74.0</td>
<td>75.4</td>
<td>76.3</td>
<td>75.9</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>
<td>92.4</td>
<td>91.3</td>
<td>89.9</td>
<td>93.1</td>
<td>91.9</td>
<td>91.9</td>
<td>90.6</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>
<td>88.6</td>
<td>88.9</td>
<td>87.8</td>
<td>88.9</td>
<td>89.0</td>
<td>88.9</td>
<td>88.2</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>
<td>76.6</td>
<td>77.3</td>
<td>77.1</td>
<td>76.9</td>
<td>78.0</td>
<td>77.8</td>
<td>76.8</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>
<td>67.9</td>
<td>73.1</td>
<td>76.2</td>
<td>74.4</td>
<td>75.0</td>
<td>76.2</td>
<td>76.1</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>
<td>86.0</td>
<td>86.5</td>
<td>86.0</td>
<td>85.4</td>
<td>86.5</td>
<td>87.6</td>
<td>87.1</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>
<td>85.2</td>
<td>85.9</td>
<td>83.8</td>
<td>85.2</td>
<td>85.9</td>
<td>85.9</td>
<td>85.9</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>
<td>85.3</td>
<td>85.7</td>
<td>83.9</td>
<td>85.1</td>
<td>85.7</td>
<td>85.7</td>
<td>84.5</td>
</tr>
<tr>
<td>microavg.</td>
<td>72.0</td>
<td>79.9</td>
<td>79.4</td>
<td>82.3</td>
<td>84.2</td>
<td>85.1</td>
<td>85.9</td>
<td>86.2</td>
<td>85.9</td>
<td>86.4</td>
<td>86.5</td>
<td>86.3</td>
<td>86.2</td>
</tr>
</tbody>
</table>

B. Leibe
Example Application: Text Classification

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

- Diagram showing:
  - 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>≈ 33000</td>
<td>227</td>
<td>4.7</td>
</tr>
<tr>
<td>3</td>
<td>≈ 1 × 10^6</td>
<td>274</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>≈ 1 × 10^9</td>
<td>321</td>
<td>4.2</td>
</tr>
<tr>
<td>5</td>
<td>≈ 1 × 10^{12}</td>
<td>374</td>
<td>4.3</td>
</tr>
<tr>
<td>6</td>
<td>≈ 1 × 10^{14}</td>
<td>377</td>
<td>4.5</td>
</tr>
<tr>
<td>7</td>
<td>≈ 1 × 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)
You Can Try It At Home...

- Lots of SVM software available, e.g.
    - Command-line based interface
    - Source code available (in C)
    - Interfaces to Python, MATLAB, Perl, Java, DLL,...
  
    - 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).

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

  B. Schölkopf, A. Smola
  Learning with Kernels
  MIT Press, 2002

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