Machine Learning - Lecture 8
Linear and Nonlinear SVMs
18.11.2013

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.
  ⇒ Overfitting to the training set!

Recap: Risk
- Empirical risk
  - Measured on the training/validation set
  \[ R_{emp}(α) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; α)) \]
- Actual risk (= Expected risk)
  - Expectation of the error on all data.
  \[ R(α) = \int L(y_i, f(x; α))dP_{X,Y}(x, y) \]
  - \( P_{X,Y}(x,y) \) is the probability distribution of \((x,y)\).
  It is fixed, but typically unknown.
  ⇒ In general, we can’t compute the actual risk directly!

Recap: Statistical Learning Theory
- Idea
  - Compute an upper bound on the actual risk based on the empirical risk
  \[ R(α) · R_{emp}(α) + ε(N, p^*, h) \]
  - where
    - \( N \): number of training examples
    - \( p^* \): probability that the bound is correct
    - \( h \): capacity of the learning machine (“VC-dimension”)
Recap: Upper Bound on the Risk

- Important result (Vapnik 1979, 1995)
  - With probability (1-?), the following bound holds
    \[
    R(\alpha) - R_{\text{emp}}(\alpha) + \sqrt{\frac{\log(2N/h) + 1 - \log(\eta/4)}{N}}
    \]
    “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
    \[
    R(\alpha) - R_{\text{emp}}(\alpha) + \epsilon(N,p^*,h)
    \]

Recap: Structural Risk Minimization

- How can we implement Structural Risk Minimization?
  \[
  R(\alpha) - R_{\text{emp}}(\alpha) + \epsilon(N,p^*,h)
  \]
- Classic approach
  - Keep \( \epsilon(N,p^*,h) \) constant and minimize \( R_{\text{emp}}(\alpha) \).
  - \( \epsilon(N,p^*,h) \) can be kept constant by controlling the model parameters.
- Support Vector Machines (SVMs)
  - Keep \( R_{\text{emp}}(\alpha) \) constant and minimize \( \epsilon(N,p^*,h) \).
  - In fact: \( R_{\text{emp}}(\alpha) = 0 \) for separable data.
  - Control \( \epsilon(N,p^*,h) \) by adapting the VC dimension (controlling the “capacity” of the classifier).

Topics of This Lecture

- Linear Support Vector Machines (Recap)
  - Lagrangian (primal) formulation
  - Dual formulation
  - Discussion
- Linearity non-separable case
  - Soft-margin classification
  - Updated formulation
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels
- 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 \( t_n \in \{-1, 1\} \).

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 \( a_n \) such that
    \[
    \frac{\partial L}{\partial w} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n x_n = 0
    \]
    \[
    \frac{\partial L}{\partial b} = 0 \Rightarrow b = \sum_{n=1}^{N} a_n t_n x_n
    \]

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
    \[
    a_n \geq 0
    \]
    \[
    t_n y(x_n) - 1 \geq 0
    \]
    \[
    f(x) \geq 0
    \]
    \[
    a_n \{ t_n y(x_n) - 1 \} = 0
    \]
    \[
    \lambda f(x) = 0
    \]
In practice, it is more robust to average over all support vectors:

\[ b = \frac{1}{N_S} \sum_{n \in S} \left( t_n - \sum_{m \in S} a_m t_m x_n^T x_m \right) \]

⇒ Only some of the data points actually influence the decision boundary!

**SVM - Dual Formulation**

Improving the scaling behavior: rewrite \( L_p \) in a dual form

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

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

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

**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
    \[ t_n f(x_n) = \sum_{m \in S} a_m t_m x_n^T x_m + b = 1 \quad \text{(KKT: } f(x) \geq 0 \text{)} \]
  - Using \( t_n^2 = 1 \), we can derive:
    \[ b = t_n - \sum_{m \in S} a_m t_m x_n^T x_m \]
  - In practice, it is more robust to average over all support vectors:
    \[ b = \frac{1}{N_S} \sum_{n \in S} \left( t_n - \sum_{m \in S} a_m t_m x_n^T x_m \right) \]

**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 - 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.
  - Formula 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 \( \Rightarrow O(D^3) \)
- Problem: scaling with high-dim. data (“curse of dimensionality")
SVM - Dual Formulation

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

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

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

Inserting this, we get the Wolfe dual

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

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...

SVM - Non-Separable Data

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

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.
  - \( \xi_n = |y_n - \phi(x_n)| \) for all other points (linear penalty).
  - We do not have to set the slack variables ourselves!
  - They are jointly optimized together with \( w \).

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**
- **Separable data**
  - Minimize
  \[
  \frac{1}{2} ||w||^2 + C \sum_{n=1}^{N} \xi_n
  \]
- **Non-separable data**
  - Minimize
  \[
  \frac{1}{2} ||w||^2 + \sum_{n=1}^{N} (y_n(w^T x_n) - 1 + \xi_n)
  \]

**Trade-off parameter**!

**SVM - New Primal Formulation**
- New SVM Primal: Optimize
  \[
  L_p = \frac{1}{2} ||w||^2 + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} a_n (y_n(w^T x_n) - 1 + \xi_n)
  \]
  \[
  \text{Constraint:} \quad t_n \geq 1 - \xi_n
  \]
  \[
  \xi_n \geq 0
  \]
  \[
  \lambda \geq 0
  \]

**KKT conditions**
- \(a_n \geq 0\)
- \(\mu_n \geq 0\)
- \(\xi_n \geq 0\)
- \(f(x) \geq 0\)
- \(\lambda f(x) = 0\)

**SVM - New Dual Formulation**
- New SVM Dual: Maximize
  \[
  L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_n^T x_m)
  \]
  under the conditions
  \[
  0 \cdot a_n \cdot C
  \]
  \[
  \sum_{n=1}^{N} a_n t_n = 0
  \]
  \[
  a_n \geq 0
  \]
  \[
  \mu_n \geq 0
  \]
  \[
  \xi_n \geq 0
  \]
  \[
  f(x) \geq 0
  \]
  \[
  \lambda f(x) = 0
  \]

This is all that changed!

**Interpretation of Support Vectors**
- Those are the hard examples!
- We can visualize them, e.g. for face detection

**SVM - New Solution**
- Solution for the hyperplane
  \[
  w = \sum_{n=1}^{N} a_n t_n x_n
  \]
  \[
  b = \frac{1}{N_M} \sum_{m \in M} \left( t_m - \sum_{m \in M} a_m t_m x_m^T x_m \right)
  \]
- Again sparse solution: \(a_n = 0\) for points outside the margin.
  \[
  \Rightarrow \text{The slack points with } \xi_n > 0 \text{ are now also support vectors!}
  \]
- Compute \(b\) by averaging over all \(N_M\) points with \(0 < a_n < C\):
  \[
  b = \frac{1}{N_M} \sum_{m \in M} \left( t_m - \sum_{m \in M} a_m t_m x_m^T x_m \right)
  \]

**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:

Another Example
- Non-separable by a hyperplane in 2D

Another Example
- Separable by a surface in 3D

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

What Could This Look Like?
- Example: Mapping to polynomial space, \( x, y \in \mathbb{R}^2 \):
  \[ \phi(x) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \]
  - 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
    \[ y(x) = w^T \phi(x) + b \]
  - Using the hyperplane, which is itself defined as
    \[ w = \sum_{n=1}^{N} a_n t_n \phi(x_n) \]

\[ \Rightarrow \text{What happens if we try this for a million-dimensional feature space } \phi(x)? \]

- 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 \( \text{implicitly} \) maps the data to the higher-dimensional space (without having to compute \( \phi(x) \) explicitly!)

Back to Our Previous Example...

- 2nd degree polynomial kernel:
  \[ \phi(x)^T \phi(y) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 \\ x_2^2 & \sqrt{2}y_1y_2 \end{bmatrix} \]
  \[ = x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \]
  \[ = (x^T y)^2 = k(x,y) \]
  - 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...
  - ...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:

\[
K = \begin{bmatrix}
\phi(x_1; x_1) & \phi(x_1; x_2) & \cdots & \phi(x_1; x_N) \\
\phi(x_2; x_1) & \phi(x_2; x_2) & \cdots & \phi(x_2; x_N) \\
\vdots & \vdots & \ddots & \vdots \\
\phi(x_N; x_1) & \phi(x_N; x_2) & \cdots & \phi(x_N; x_N)
\end{bmatrix}
\]

\[ \text{(positive definite \( \Rightarrow \) 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
  \[ k(x, y) = \tanh(\langle x^T y \rangle) \]
  e.g. Sigmoid

(And many, many more...)

Actually, that was wrong in the original SVM paper...
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. χ² kernel

  \[ k_\chi(h, h') = \exp \left( -\frac{1}{\gamma} \frac{(h - h')^2}{h + h'} \right) \]

VC Dimension for Polynomial Kernel

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

VC Dimension for Gaussian RBF Kernel

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

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!
  \[ \exp(x) = 1 + x + \frac{x^2}{2!} + \ldots \]
  - Since only the kernel function is used by the SVM, this is no problem.
  - The hyperplane in \( \mathcal{H} \) then has VC-dimension
    \[ \dim(\mathcal{H}) + 1 = \infty \]

Nonlinear SVM - Dual Formulation

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

Example: RBF Kernels

- Decision boundary on toy problem
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: \( a_1, a_2, \ldots, a_N \) 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
    \[
    h \leq \min \left\{ \frac{D^2}{\rho^2}, m_0 \right\} + 1
    \]
  - where \( \rho \) is the margin, \( D \) is the diameter of the smallest sphere that can enclose all of the training examples, and \( m_0 \) is the dimensionality.
  - Intuitively, this implies that regardless of dimensionality \( m_0 \) we can minimize the VC dimension by maximizing the margin \( \rho \).
  - Thus, complexity of the classifier is kept small regardless of dimensionality.

Summary: SVMs

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

- Limitations
  - How to select the right kernel?
    - 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 \( g(x) \) 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

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

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

Example Application: OCR

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

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

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: Object Detection

- Sliding-window approach

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

Example Application: Pedestrian Detection

Many Other Applications

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

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

You Can Try It At Home...

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

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