# Machine Learning – Lecture 8

## Support Vector Machines

06.11.2019

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

leibe@vision.rwth-aachen.de

# Announcements

- Exam dates
  - 1st date: Saturday, 29.02., 13:30h – 15:30h
  - 2nd date: Thursday, 19.03., 11:00h – 13:00h

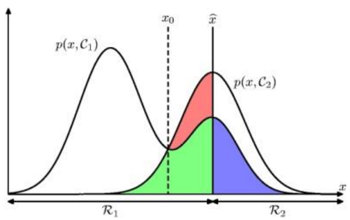
  - The exam dates have been optimized to avoid overlaps with other Computer Science Master lectures as much as possible.
  - If you still have conflicts with *both* exam dates, please tell us.

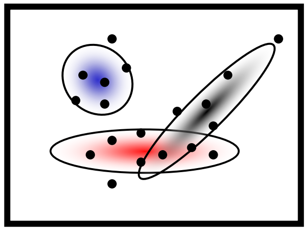
  - If you're *an exchange student and need to leave RWTH before the first exam date*, we will offer some special oral exam slots
    - Please do NOT contact us about those yet.
    - We will let you sign up for those special exam slots in early January

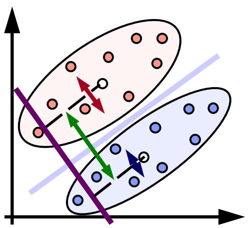
- *Please do not forget to register for the exam in RWTH online!*

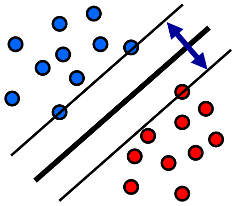
# Course Outline

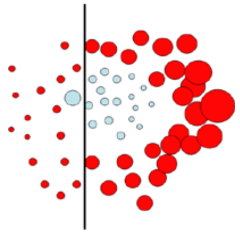
- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
- Classification Approaches
  - Linear Discriminants
  - Support Vector Machines
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns
- Deep Learning
  - Foundations
  - Convolutional Neural Networks
  - Recurrent Neural Networks

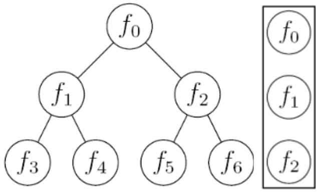


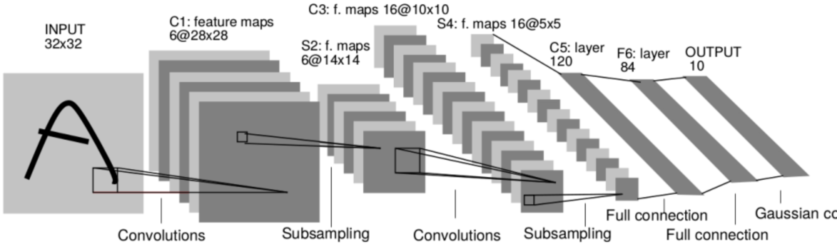










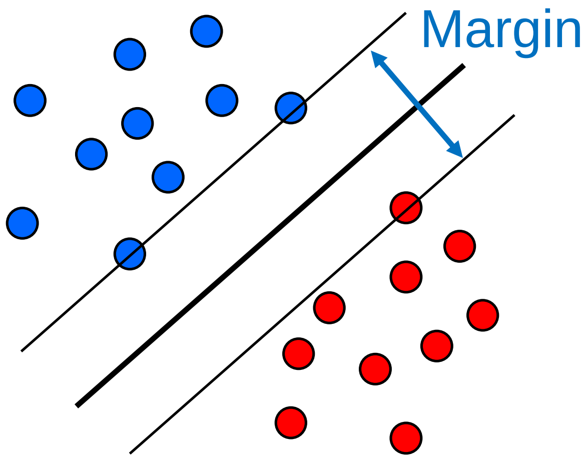
# Topics of This Lecture

- Support Vector Machines
  - Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer's condition
  - Popular kernels
- Analysis
  - Error function
- Applications

# Recap: Support Vector Machine (SVM)

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

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$$




- Formulation as a convex optimization problem
  - Find the hyperplane satisfying

$$\underset{\mathbf{w},b}{\arg\min} \frac{1}{2}\|\mathbf{w}\|^2$$

under the constraints

$$t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

based on training data points $\mathbf{x}_n$ and target values $t_n \in \{-1, 1\}$

# Support Vector Machine (SVM)

- Optimization problem
  - Find the hyperplane satisfying

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

    under the constraints

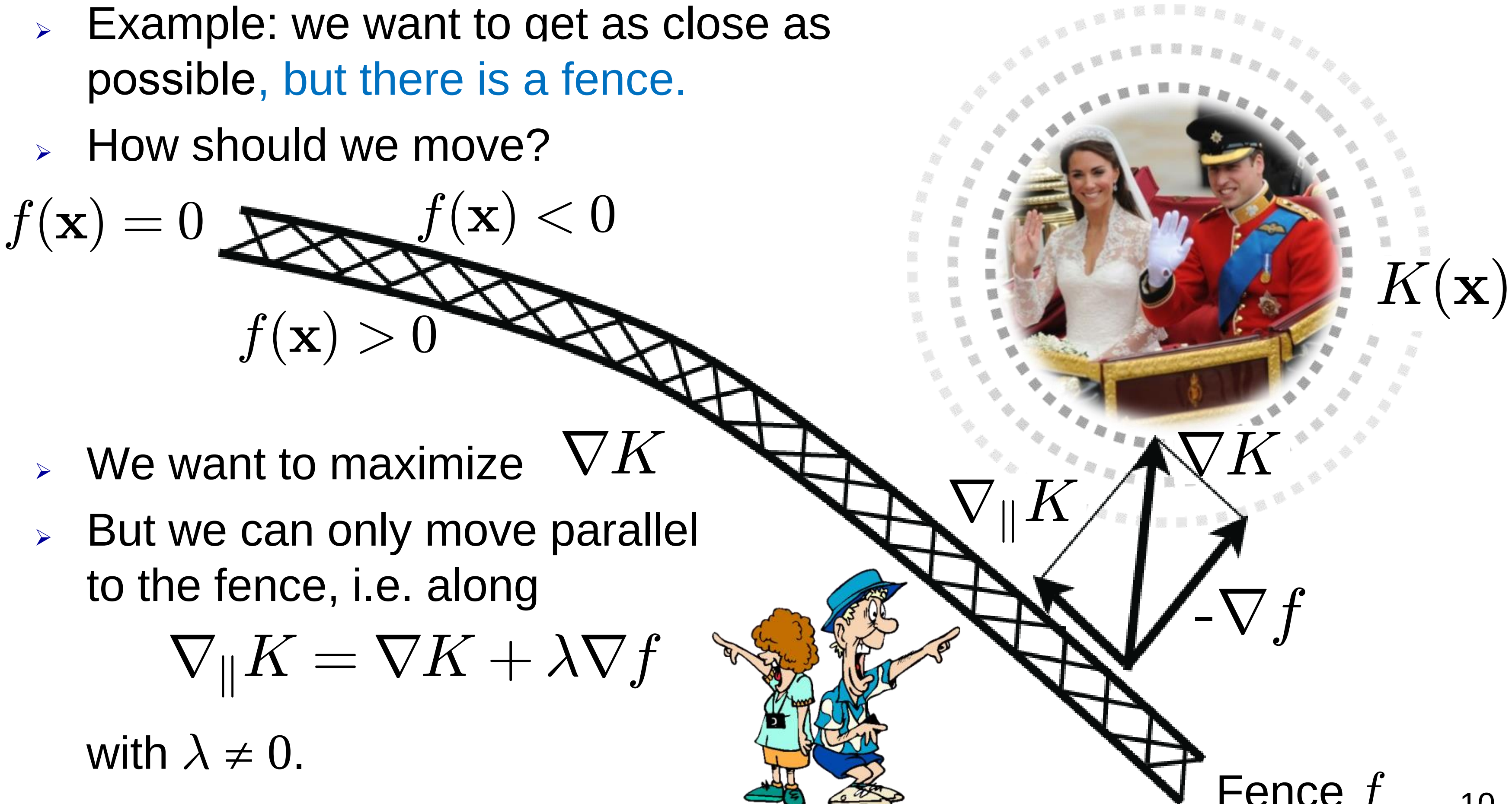
$$t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

  - Quadratic programming problem with linear constraints.
  - Can be formulated using Lagrange multipliers.

- *Who is already familiar with Lagrange multipliers?*
  - Let's look at a real-life example…

# Recap: Lagrange Multipliers

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



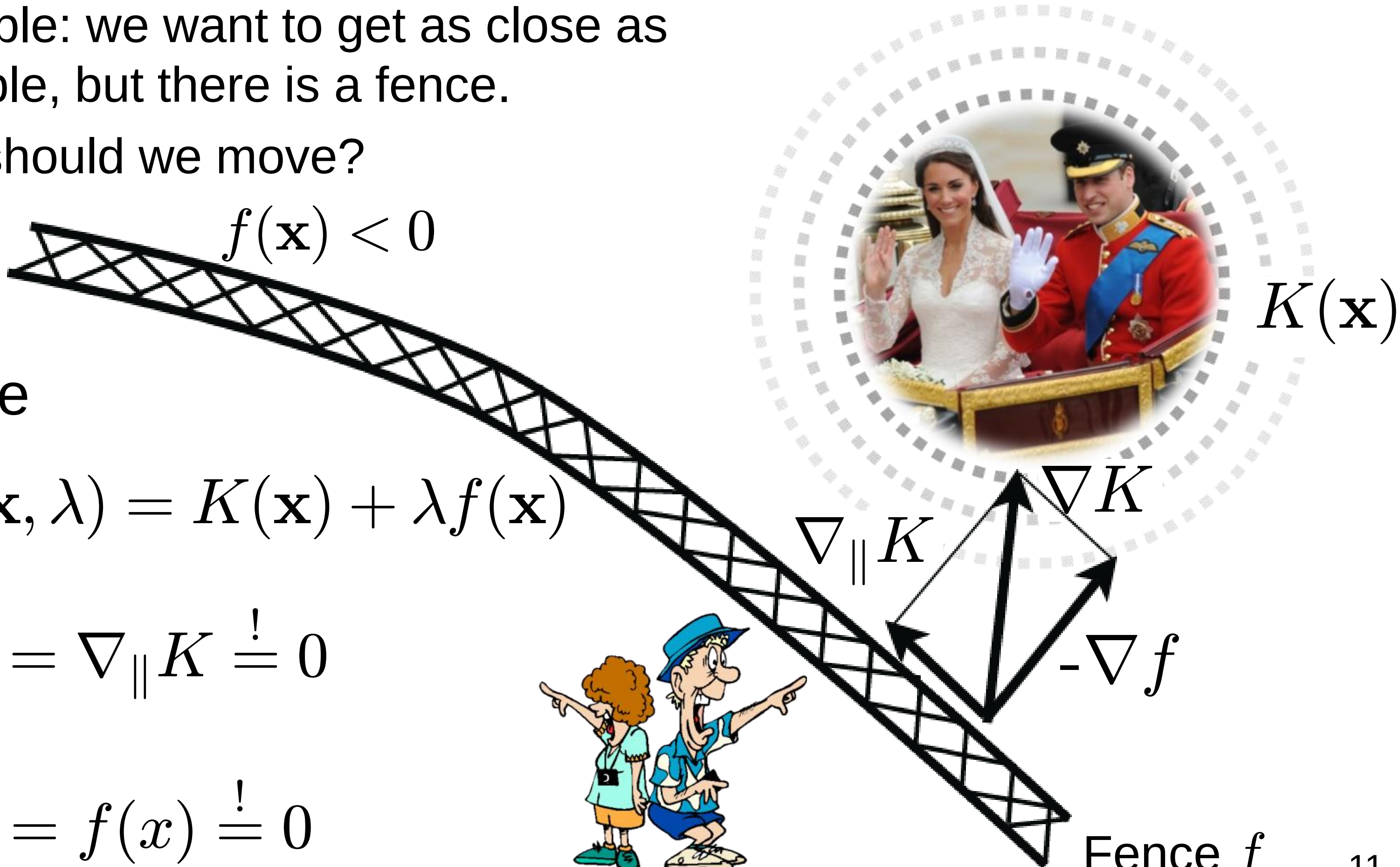

  - We want to maximize $\nabla K$
  - But we can only move parallel to the fence, i.e. along

$$\nabla_{\|} K = \nabla K + \lambda \nabla f$$

with $\lambda \neq 0$.

# Recap: Lagrange Multipliers

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




$\Rightarrow$ Optimize

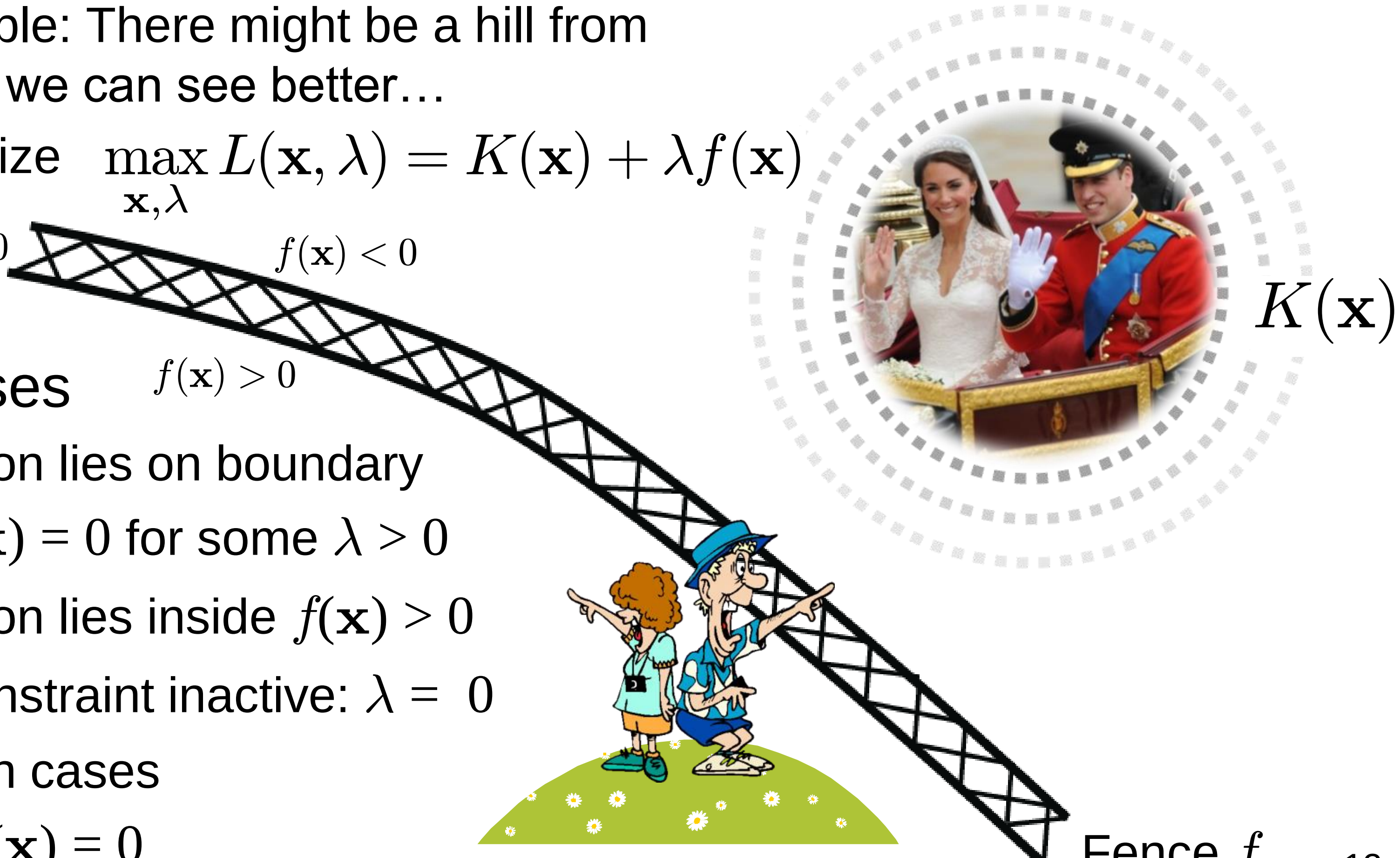
$$\max_{\mathbf{x},\lambda} L(\mathbf{x}, \lambda) = K(\mathbf{x}) + \lambda f(\mathbf{x})$$

$$\frac{\partial L}{\partial \mathbf{x}} = \nabla_{\|} K \stackrel{!}{=} 0$$

$$\frac{\partial L}{\partial \lambda} = f(x) \stackrel{!}{=} 0$$

# Recap: Lagrange Multipliers

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




- Two cases
  - Solution lies on boundary

    $\Rightarrow f(\mathbf{x}) = 0$ for some $\lambda > 0$
  - Solution lies inside $f(\mathbf{x}) > 0$

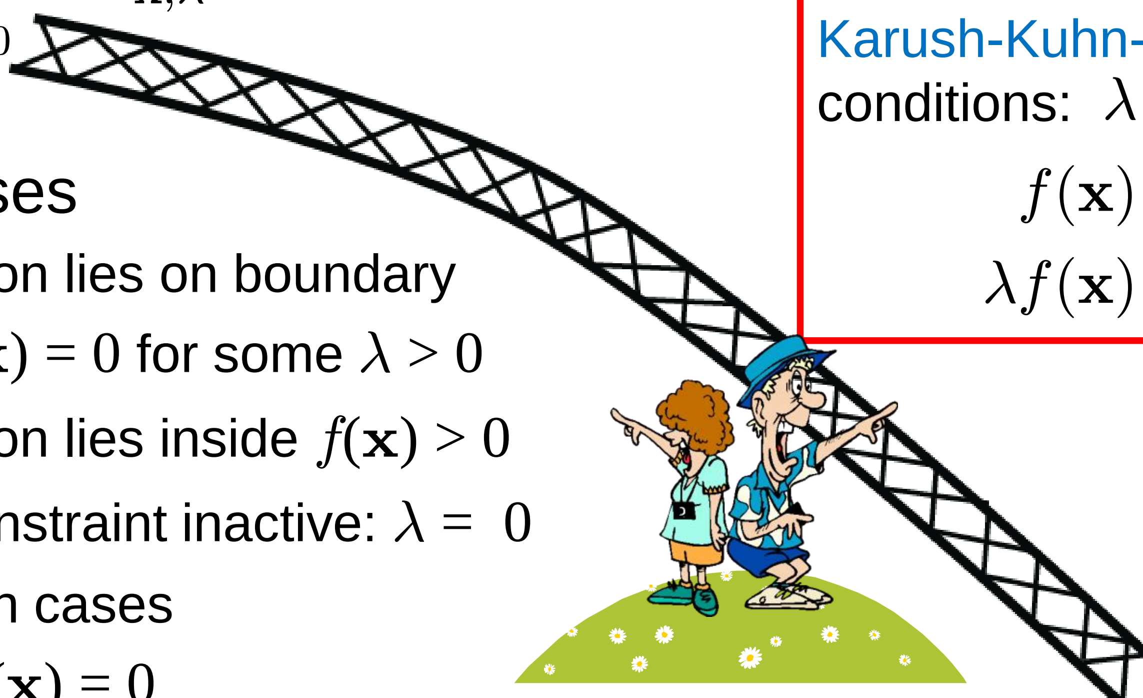
    $\Rightarrow$ Constraint inactive: $\lambda = 0$
  - In both cases

    $\Rightarrow \lambda f(\mathbf{x}) = 0$

# Recap: Lagrange Multipliers

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

$f(\mathbf{x}) = 0$



Fence $f$

**Karush-Kuhn-Tucker (KKT) conditions:**

$$\begin{aligned} \lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0 \end{aligned}$$

- Two cases
  - Solution lies on boundary

    $\Rightarrow f(\mathbf{x}) = 0$ for some $\lambda > 0$
  - Solution lies inside $f(\mathbf{x}) > 0$

    $\Rightarrow$ Constraint inactive: $\lambda = 0$
  - In both cases

    $\Rightarrow \lambda f(\mathbf{x}) = 0$

# SVM – Lagrangian Formulation

- Find hyperplane minimizing $\|\mathbf{w}\|^2$ under the constraints

$$t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) - 1 \geq 0 \quad \forall n$$

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

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) - 1\right\}$$

  - I.e., find $\mathbf{w}$, $b$, and $\mathbf{a}$ such that

$$\frac{\partial L}{\partial b} = 0 \;\Rightarrow\; \boxed{\sum_{n=1}^{N} a_n t_n = 0} \qquad \frac{\partial L}{\partial \mathbf{w}} = 0 \;\Rightarrow\; \boxed{\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n}$$

# SVM – Lagrangian Formulation

- Lagrangian primal form

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

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{t_n y(\mathbf{x}_n) - 1\right\}$$

- The solution of $L_p$ needs to fulfill the KKT conditions
  - Necessary and sufficient conditions

$$\begin{aligned} a_n &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 &\geq 0 \\ a_n \left\{t_n y(\mathbf{x}_n) - 1\right\} &= 0 \end{aligned}$$

KKT:

$$\begin{aligned} \lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0 \end{aligned}$$

# SVM – Solution (Part 1)

- Solution for the hyperplane
  - Computed as a linear combination of the training examples

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$

  - Because of the KKT conditions, the following must also hold

$$a_n \left( t_n (\mathbf{w}^{\mathrm{T}} \mathbf{x}_n + b) - 1 \right) = 0$$

KKT:
$\lambda f(\mathbf{x}) = 0$

  - This implies that $a_n > 0$ only for training data points for which

$$\left( t_n (\mathbf{w}^{\mathrm{T}} \mathbf{x}_n + b) - 1 \right) = 0$$

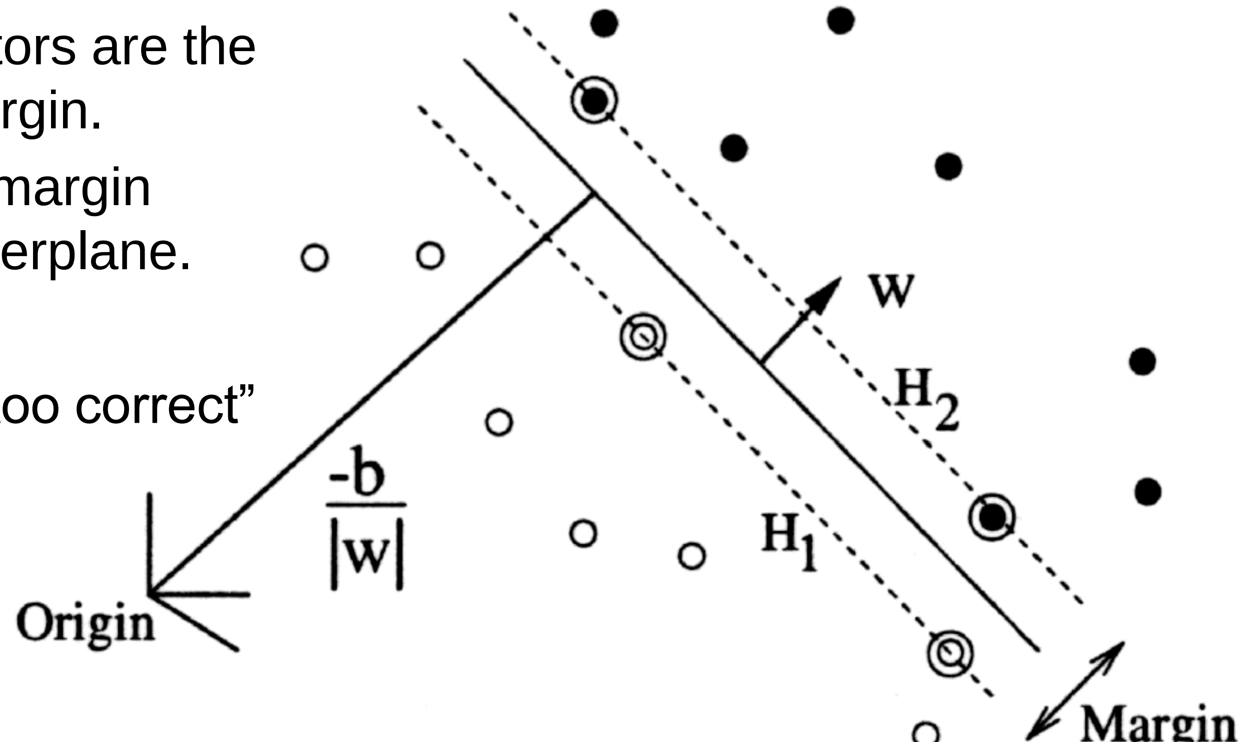
$\Rightarrow$ *Only some of the data points actually influence the decision boundary!*

# 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 $\mathbf{x}_n$ satisfies

$$t_n y(\mathbf{x}_n) = t_n \left( \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n + b \right) = 1$$

KKT:
$f(\mathbf{x}) \geq 0$

  - Using $t_n^2 = 1$ we can derive: $b = t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n$
  - In practice, it is more robust to average over all support vectors:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left( t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n \right)$$

# SVM – Discussion (Part 1)

- Linear SVM
  - Linear classifier
  - SVMs have a "guaranteed" generalization capability.
  - Formulation as convex optimization problem.

  $\Rightarrow$ Globally optimal solution!

- Primal form formulation
  - Solution to quadratic prog. problem in $M$ variables is in $\mathcal{O}(M^3)$.
  - Here: $D$ variables $\Rightarrow \mathcal{O}(D^3)$
  - Problem: scaling with high-dim. data ("curse of dimensionality")

# SVM – Dual Formulation

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

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

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n t_n \mathbf{w}^{\mathrm{T}}\mathbf{x}_n - \overset{=0}{\cancel{b\sum_{n=1}^{N} a_n t_n}} + \sum_{n=1}^{N} a_n$$

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

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

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

# SVM – Dual Formulation

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

- Using the constraint $\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$ we obtain

$$\boxed{\frac{\partial L_p}{\partial \mathbf{w}} = 0}$$

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n t_n \sum_{m=1}^{N} a_m t_m \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n + \sum_{n=1}^{N} a_n$$

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

# SVM – Dual Formulation

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

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

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

- Inserting this, we get the Wolfe dual

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

# SVM – Dual Formulation

- Maximize

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

under the conditions

$$\begin{aligned} a_n &\geq 0 \quad \forall n \\ \sum_{n=1}^{N} a_n t_n &= 0 \end{aligned}$$

  - The hyperplane is given by the $N_S$ support vectors:

$$\mathbf{w} = \sum_{n=1}^{N_{\mathcal{S}}} a_n t_n \mathbf{x}_n$$

# 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!

  $\Rightarrow$ This makes it possible to construct efficient algorithms
    - e.g. Sequential Minimal Optimization (SMO)
    - Effective runtime between $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$.

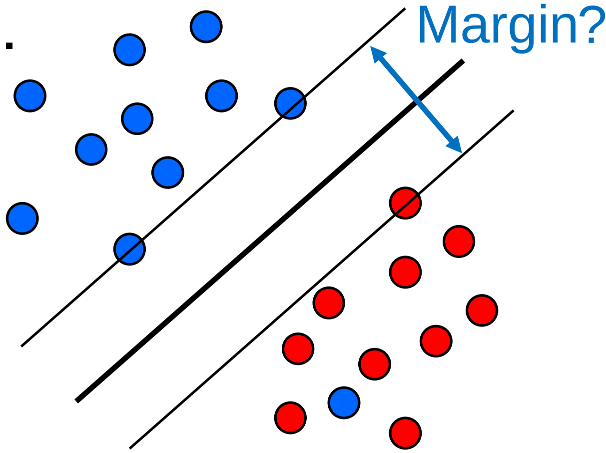
  2. We have avoided the dependency on the dimensionality.

  $\Rightarrow$ This makes it possible to work with infinite-dimensional feature spaces by using suitable basis functions $\phi(\mathbf{x})$.

  $\Rightarrow$ We'll see that later in today's lecture…

# So Far…

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



# SVM – Non-Separable Data

- Non-separable data
  - I.e. the following inequalities cannot be satisfied for all data points

$$
\begin{aligned}
\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b &\geq +1 && \text{for} \quad t_n = +1 \\
\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b &\leq -1 && \text{for} \quad t_n = -1
\end{aligned}
$$

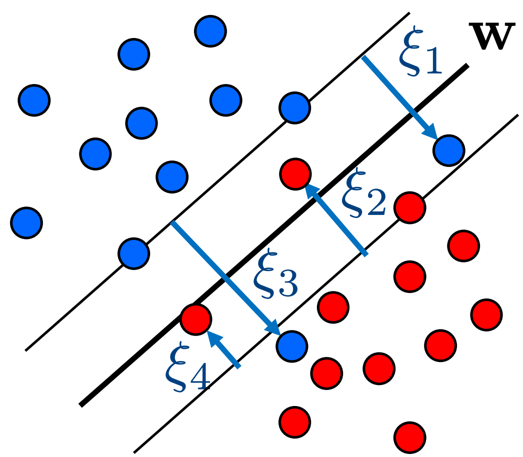
  - Instead use

$$
\begin{aligned}
\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b &\geq +1 - \xi_n && \text{for} \quad t_n = +1 \\
\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b &\leq -1 + \xi_n && \text{for} \quad t_n = -1
\end{aligned}
$$

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.
  - $\xi_n = |t_n - y(\mathbf{x}_n)|$ for all other points (linear penalty).




Point on decision boundary: $\xi_n = 1$

Misclassified point: $\xi_n > 1$

- We do not have to set the slack variables ourselves!

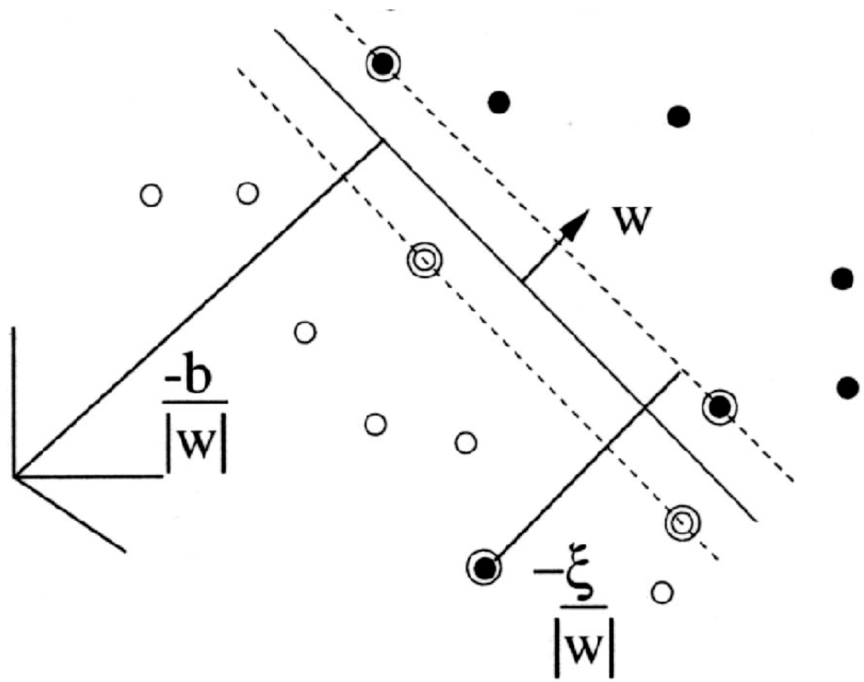
$\Rightarrow$ They are jointly optimized together with $\mathbf{w}$.

How that?

# SVM – Non-Separable Data

- Separable data
  - Minimize $$\frac{1}{2}\|\mathbf{w}\|^2$$
- Non-separable data
  - Minimize $$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

Trade-off parameter!



# SVM – New Primal Formulation

- New SVM Primal: Optimize

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \underbrace{\sum_{n=1}^{N} a_n\left(t_n y(\mathbf{x}_n) - 1 + \xi_n\right)}_{\substack{\text{Constraint} \\ t_n y(\mathbf{x}_n) \geq 1 - \xi_n}} - \underbrace{\sum_{n=1}^{N}\mu_n \xi_n}_{\substack{\text{Constraint} \\ \xi_n \geq 0}}$$

- KKT conditions

$$\begin{aligned} a_n &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n &\geq 0 \\ a_n\left(t_n y(\mathbf{x}_n) - 1 + \xi_n\right) &= 0 \end{aligned} \qquad \begin{aligned} \mu_n &\geq 0 \\ \xi_n &\geq 0 \\ \mu_n \xi_n &= 0 \end{aligned}$$

KKT:

$$\boxed{\begin{aligned} \lambda &\geq 0 \\ f(\mathbf{x}) &\geq 0 \\ \lambda f(\mathbf{x}) &= 0 \end{aligned}}$$

# SVM – New Dual Formulation

- New SVM Dual: Maximize

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

under the conditions

$$0 \cdot \quad a_n \cdot \quad C$$

$$\sum_{n=1}^{N} a_n t_n \;=\; 0$$

This is all that changed!

- This is again a quadratic programming problem

  $\Rightarrow$ Solve as before… (more on that later)

# SVM – New Solution

- Solution for the hyperplane
  - Computed as a linear combination of the training examples

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$

  - Again sparse solution: $a_n = 0$ for points outside the margin.

$\Rightarrow$ The slack points with $\xi_n > 0$ are now also support vectors!

  - Compute $b$ by averaging over all $N_{\mathcal{M}}$ points with $0 < a_n < C$:

$$b = \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \left( t_n - \sum_{m \in \mathcal{M}} a_m t_m \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n \right)$$

# Interpretation of Support Vectors

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

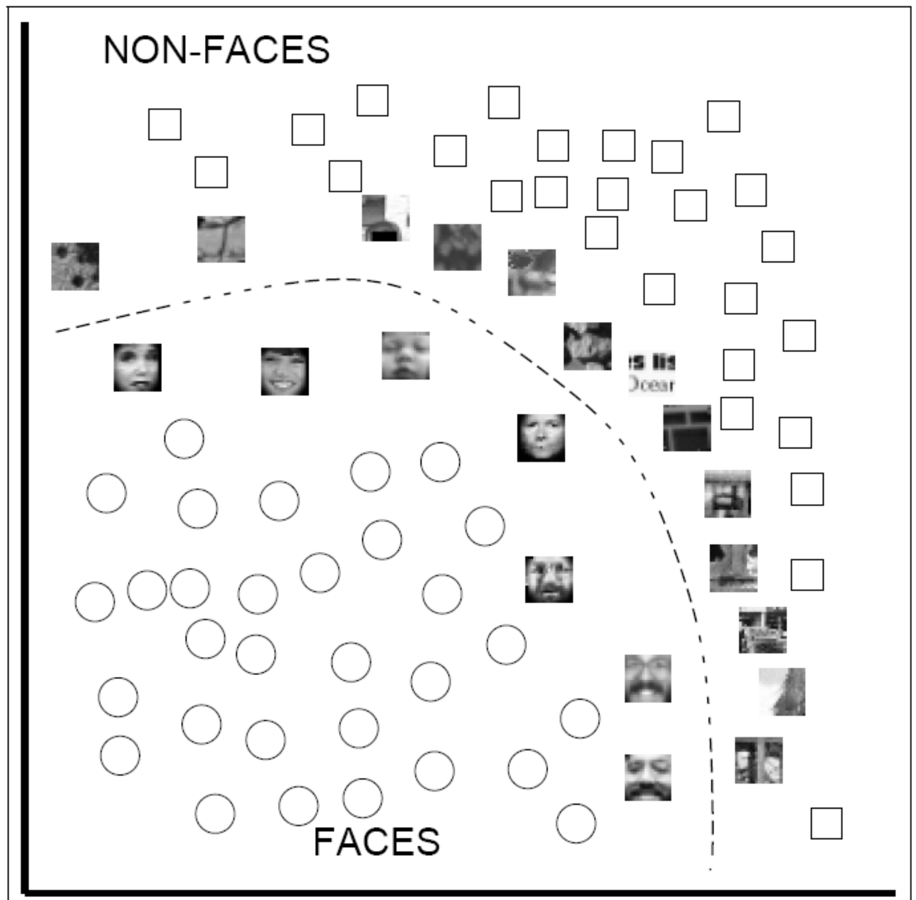




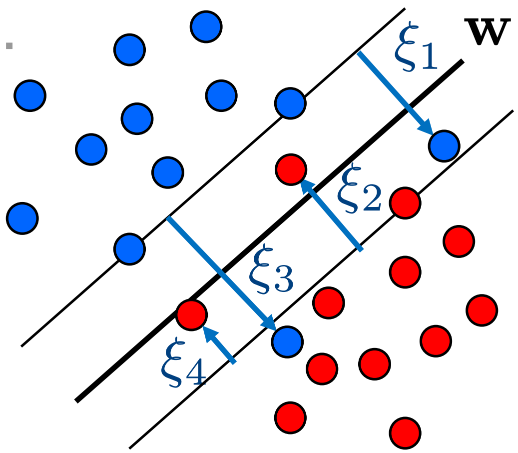

Image source: E. Osuna, F. Girosi, 1997

# Topics of This Lecture

- Support Vector Machines
  - Recap: Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels
- Analysis
  - Error function
- Applications

# So Far…

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




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

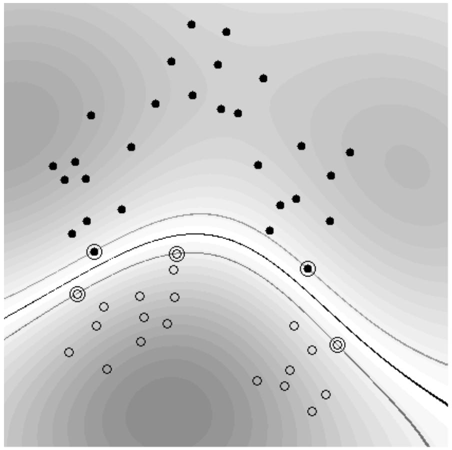
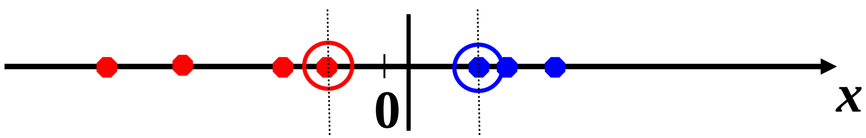


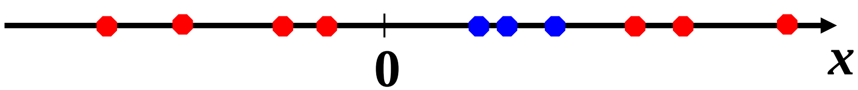
Image source: B. Schoelkopf, A. Smola, 2002

# Nonlinear SVM

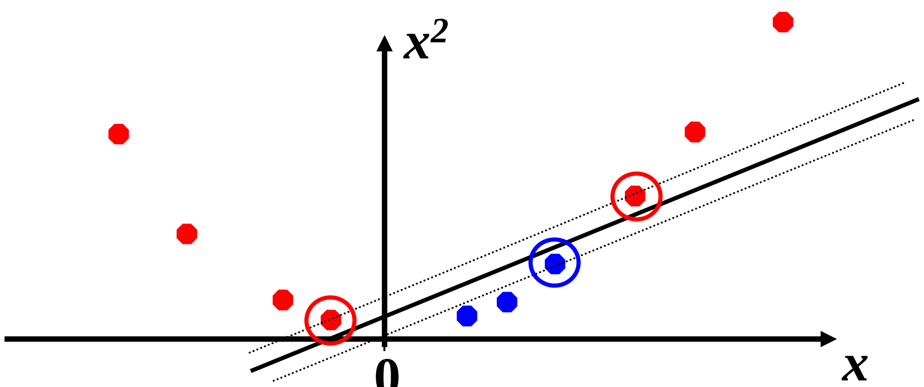
- Linear SVMs
  - Datasets that are linearly separable with some noise work well:




  - But what are we going to do if the dataset is just too hard?

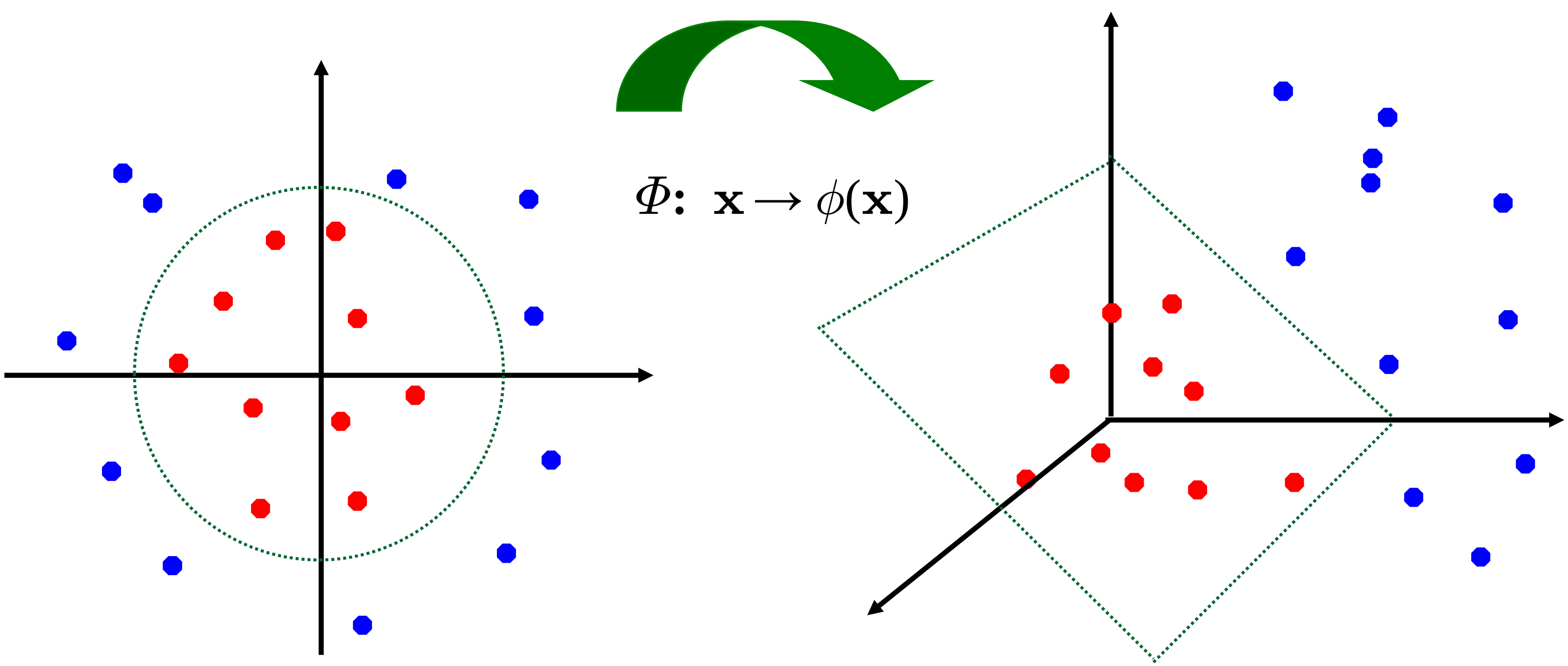



  - How about… mapping data to a higher-dimensional space:



# 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:








Slide credit: Raymond Mooney

# Nonlinear SVM

- General idea
  - Nonlinear transformation $\phi$ of the data points $\mathbf{x}_n$:

$$\mathbf{x} \in \mathbb{R}^D \quad \phi : \mathbb{R}^D \rightarrow \mathcal{H}$$

  - Hyperplane in higher-dim. space $\mathcal{H}$ (linear classifier in $\mathcal{H}$)

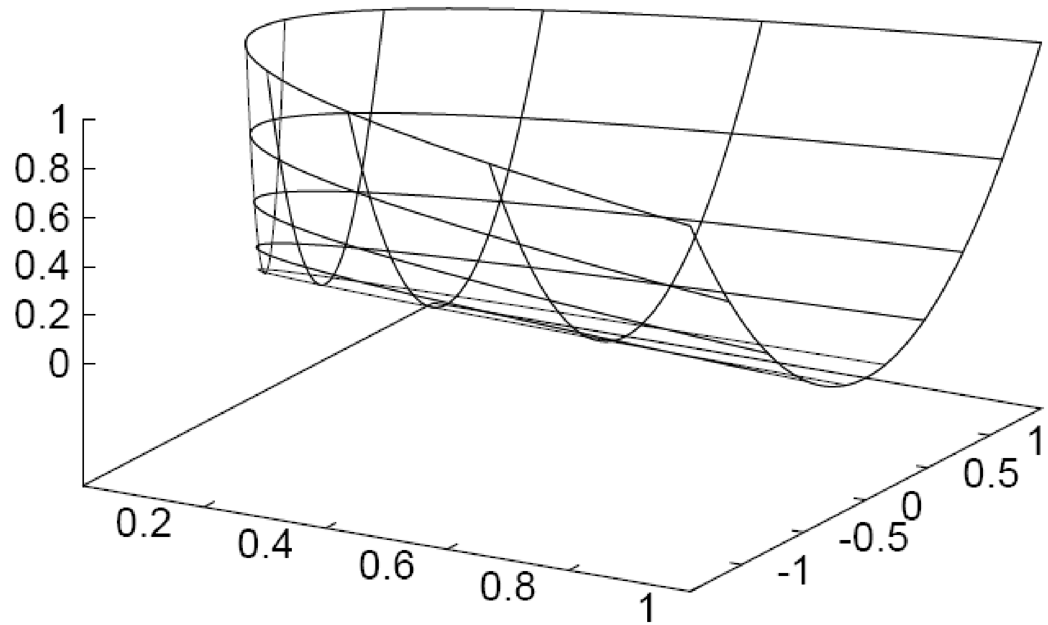
$$\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b = 0$$

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

# What Could This Look Like?

- Example:
  - Mapping to polynomial space, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$:

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$




  - Motivation: Easier to separate data in higher-dimensional space.
  - But wait – isn't there a big problem?
    - How should we evaluate the decision function?



Image source: C. Burges, 1998

# Problem with High-dim. Basis Functions

- Problem
  - In order to apply the SVM, we need to evaluate the function

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$$

  - Using the hyperplane, which is itself defined as

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

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

  - Oh-oh…

# Solution: The Kernel Trick

- Important observation
  - $\phi(\mathbf{x})$ only appears in the form of dot products $\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{y})$:

$$\begin{aligned} y(\mathbf{x}) &= \mathbf{w}^\mathrm{T}\phi(\mathbf{x}) + b \\ &= \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)^\mathrm{T}\phi(\mathbf{x}) + b \end{aligned}$$

  - Trick: Define a so-called kernel function $k(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{y})$.
  - Now, in place of the dot product, use the kernel instead:

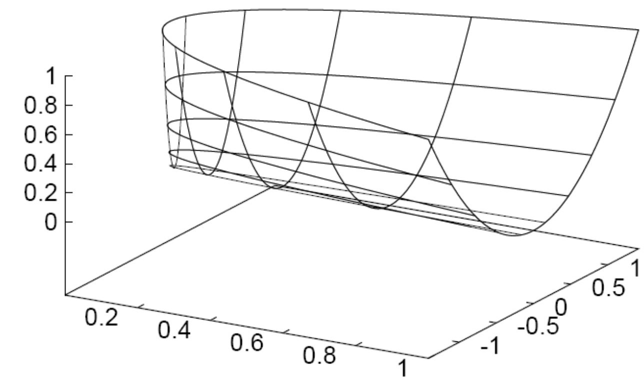
$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

  - The kernel function *implicitly* maps the data to the higher-dimensional space (without having to compute $\phi(\mathbf{x})$ explicitly)!

# Back to Our Previous Example…

- 2nd degree polynomial kernel:

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




$$= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2$$

$$= (\mathbf{x}^{\mathrm{T}}\mathbf{y})^2 =: k(\mathbf{x},\mathbf{y})$$

  - Whenever we evaluate the kernel function $k(\mathbf{x},\mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y})^2$, we implicitly compute the dot product in the higher-dimensional feature space.


Image source: C. Burges, 1998

# SVMs with Kernels

- Using kernels
  - Applying the kernel trick is easy. Just replace every dot product by a kernel function…

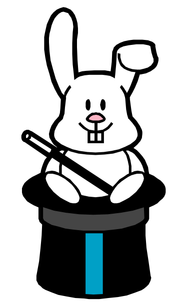
$$\mathbf{x}^{\mathrm{T}}\mathbf{y} \quad \rightarrow \quad k(\mathbf{x}, \mathbf{y})$$

  - …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 $\phi(\mathbf{x})$.
  - 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:

$$K = \begin{array}{|c|c|c|c|c|}
\hline
k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & k(\mathbf{x}_1,\mathbf{x}_3) & \dots & k(\mathbf{x}_1,\mathbf{x}_n) \\
\hline
k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & k(\mathbf{x}_2,\mathbf{x}_3) & & k(\mathbf{x}_2,\mathbf{x}_n) \\
\hline
 & & & & \\
\hline
\dots & \dots & \dots & \dots & \dots \\
\hline
k(\mathbf{x}_n,\mathbf{x}_1) & k(\mathbf{x}_n,\mathbf{x}_2) & k(\mathbf{x}_n,\mathbf{x}_3) & \dots & k(\mathbf{x}_n,\mathbf{x}_n) \\
\hline
\end{array}$$

(positive definite = all eigenvalues are > 0)

# Kernels Fulfilling Mercer's Condition

- Polynomial kernel

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y} + 1)^p$$

- Radial Basis Function kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp\left\{-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}\right\}$$

e.g. Gaussian

- Hyperbolic tangent kernel

$$k(\mathbf{x}, \mathbf{y}) \cancel{= \tanh(\kappa\mathbf{x}^{\mathrm{T}}\mathbf{y} + \delta)}$$

e.g. Sigmoid

Actually, this was wrong in the original SVM paper...

(and many, many more…)

# 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

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_m, \mathbf{x}_n)$$

under the conditions

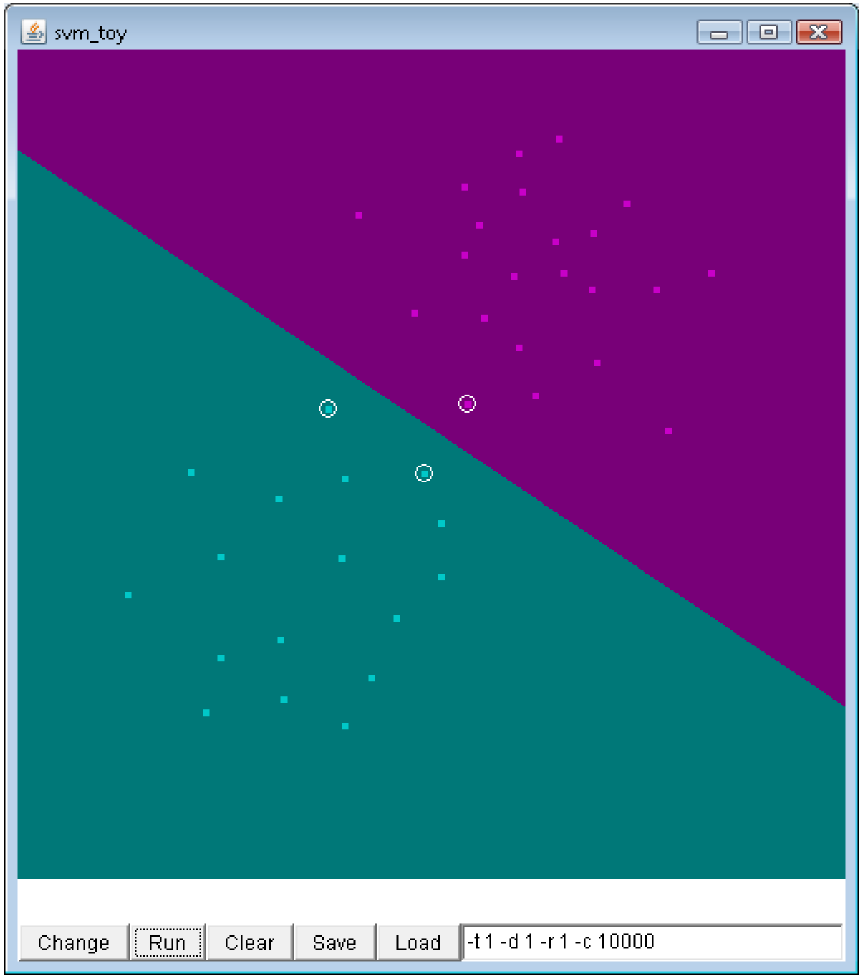
$$0 \cdot \quad a_n \cdot \quad C$$

$$\sum_{n=1}^{N} a_n t_n \quad = \quad 0$$

- Classify new data points using

$$y(\mathbf{x}) \quad = \quad \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

# SVM Demo




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

# 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/

# Summary: SVMs

- Limitations
  - How to select the right kernel?
    - Best practice guidelines are available for many applications
  - How to select the kernel parameters?
    - (Massive) cross-validation.
    - Usually, several parameters are optimized together in a grid search.
  - Solving the quadratic programming problem
    - Standard QP solvers do not perform too well on SVM task.
    - Dedicated methods have been developed for this, e.g. SMO.
  - Speed of evaluation
    - Evaluating $y(\mathbf{x})$ scales linearly in the number of SVs.
    - Too expensive if we have a large number of support vectors.

    $\Rightarrow$ There are techniques to reduce the effective SV set.
  - Training for very large datasets (millions of data points)
    - Stochastic gradient descent and other approximations can be used

# Topics of This Lecture

- Support Vector Machines
  - Recap: Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer's condition
  - Popular kernels
- Analysis
  - Error function
- Applications

# SVM – Analysis

- Traditional soft-margin formulation

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

“Maximize the margin”

subject to the constraints

$$t_n y(\mathbf{x}_n) \;\geq\; 1-\xi_n$$

“Most points should be on the correct side of the margin”

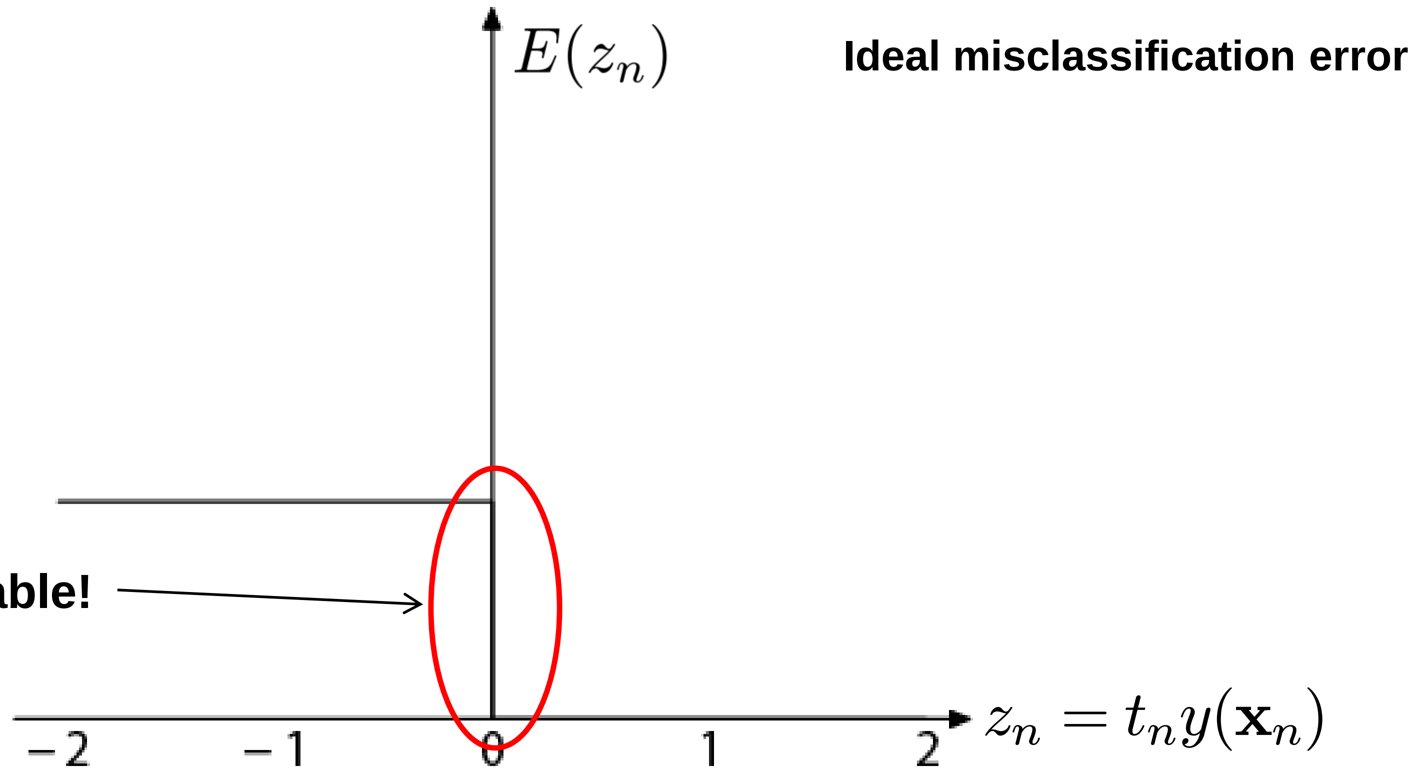
- Different way of looking at it
  - We can reformulate the constraints into the objective function.

$$\min_{\mathbf{w}\in\mathbb{R}^D} \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{L_2 \text{ regularizer}} + \underbrace{C\sum_{n=1}^{N}[1-t_n y(\mathbf{x}_n)]_+}_{\text{“Hinge loss”}}$$

where $[x]_+ := \max\{0,x\}$.

# Recap: Error Functions

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




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

Image source: Bishop, 2006

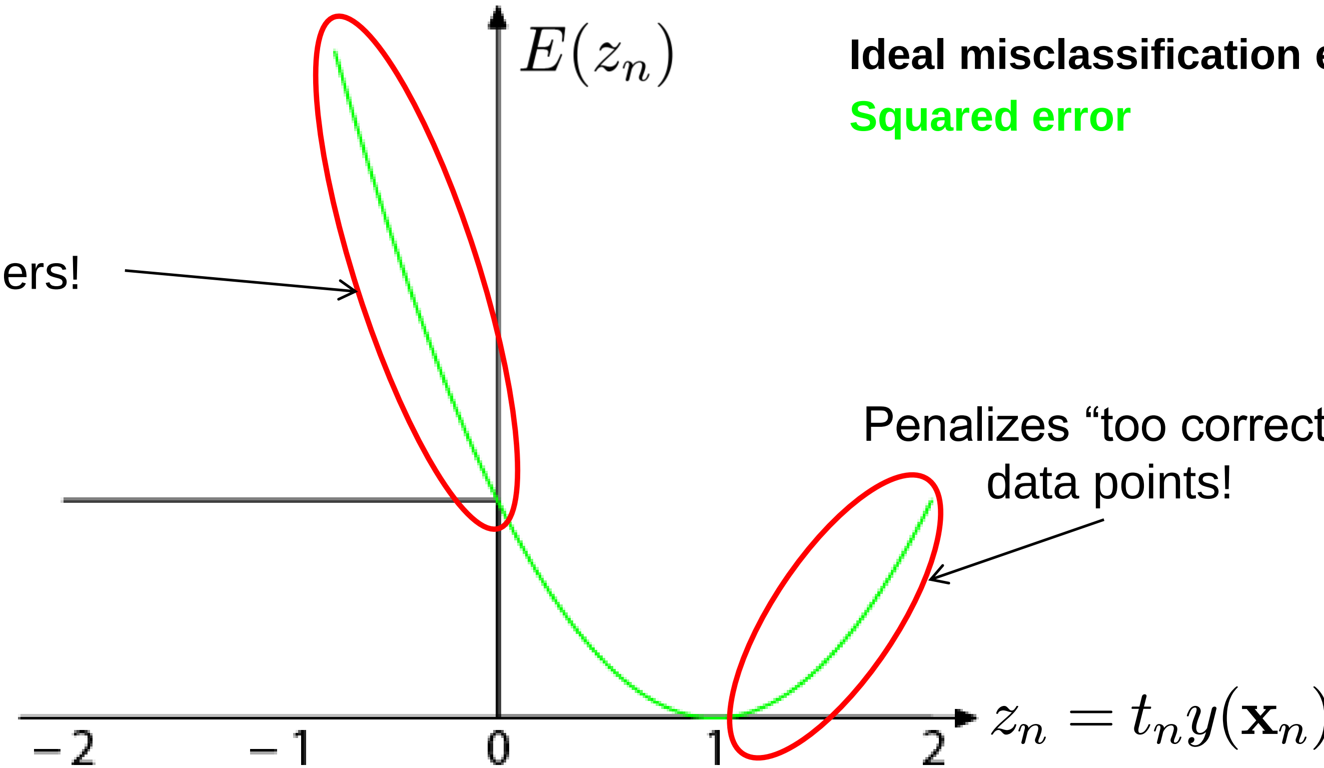
# Recap: Error Functions

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

Ideal misclassification error

Squared error

Sensitive to outliers!




Penalizes "too correct" data points!

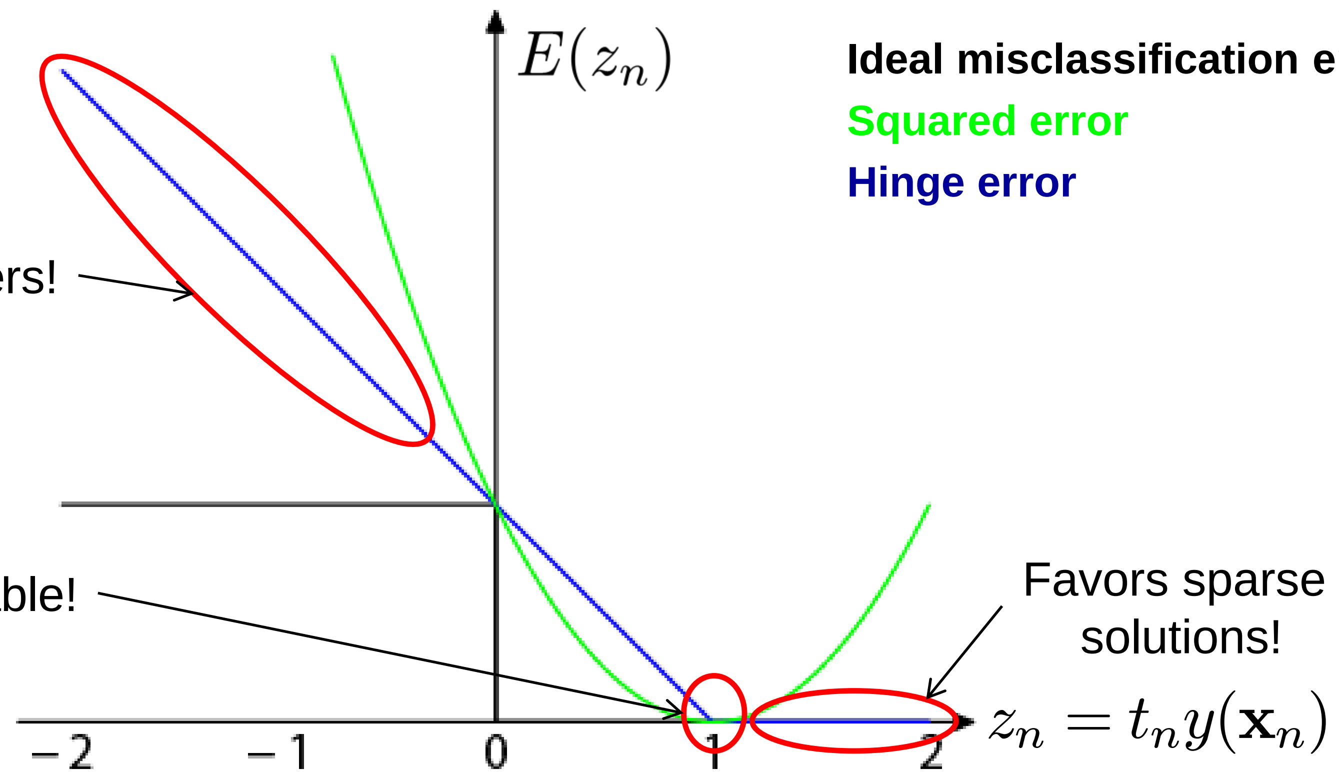
- Squared error used in Least-Squares Classification
  - Very popular, leads to closed-form solutions.
  - However, sensitive to outliers due to squared penalty.
  - Penalizes "too correct" data points

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



Image source: Bishop, 2006

# Error Functions (Loss Functions)




- "Hinge error" used in SVMs
  - Zero error for points outside the margin ($z_n > 1$) $\Rightarrow$ sparsity
  - Linear penalty for misclassified points ($z_n < 1$) $\Rightarrow$ robustness
  - Not differentiable around $z_n = 1$ $\Rightarrow$ Cannot be optimized directly.


Image source: Bishop, 2006

# SVM – Discussion

- SVM optimization function

$$\min_{\mathbf{w}\in\mathbb{R}^D} \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{L}_2\text{ regularizer}} + \underbrace{C\sum_{n=1}^{N}[1-t_n y(\mathbf{x}_n)]_+}_{\text{Hinge loss}}$$

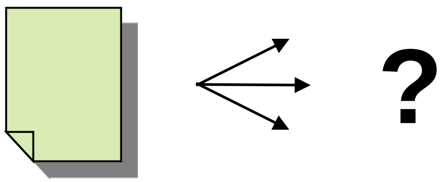
- Hinge loss enforces sparsity
  - Only a subset of training data points actually influences the decision boundary.
  - This is different from sparsity obtained through the regularizer! There, only a subset of input dimensions are used.
  - Unconstrained optimization, but non-differentiable function.
  - Solve, e.g. by *subgradient descent*
  - Currently most efficient: *stochastic gradient descent*

# Topics of This Lecture

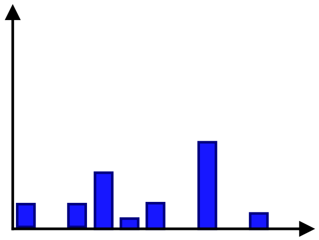
- Support Vector Machines
  - Recap: Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer's condition
  - Popular kernels
- Analysis
  - Error function
- Applications

# Example Application: Text Classification

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




- Representation:
  - "Bag-of-words" approach
  - Histogram of word counts (on learned dictionary)
    - Very high-dimensional feature space (~10.000 dimensions)
    - Few irrelevant features



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

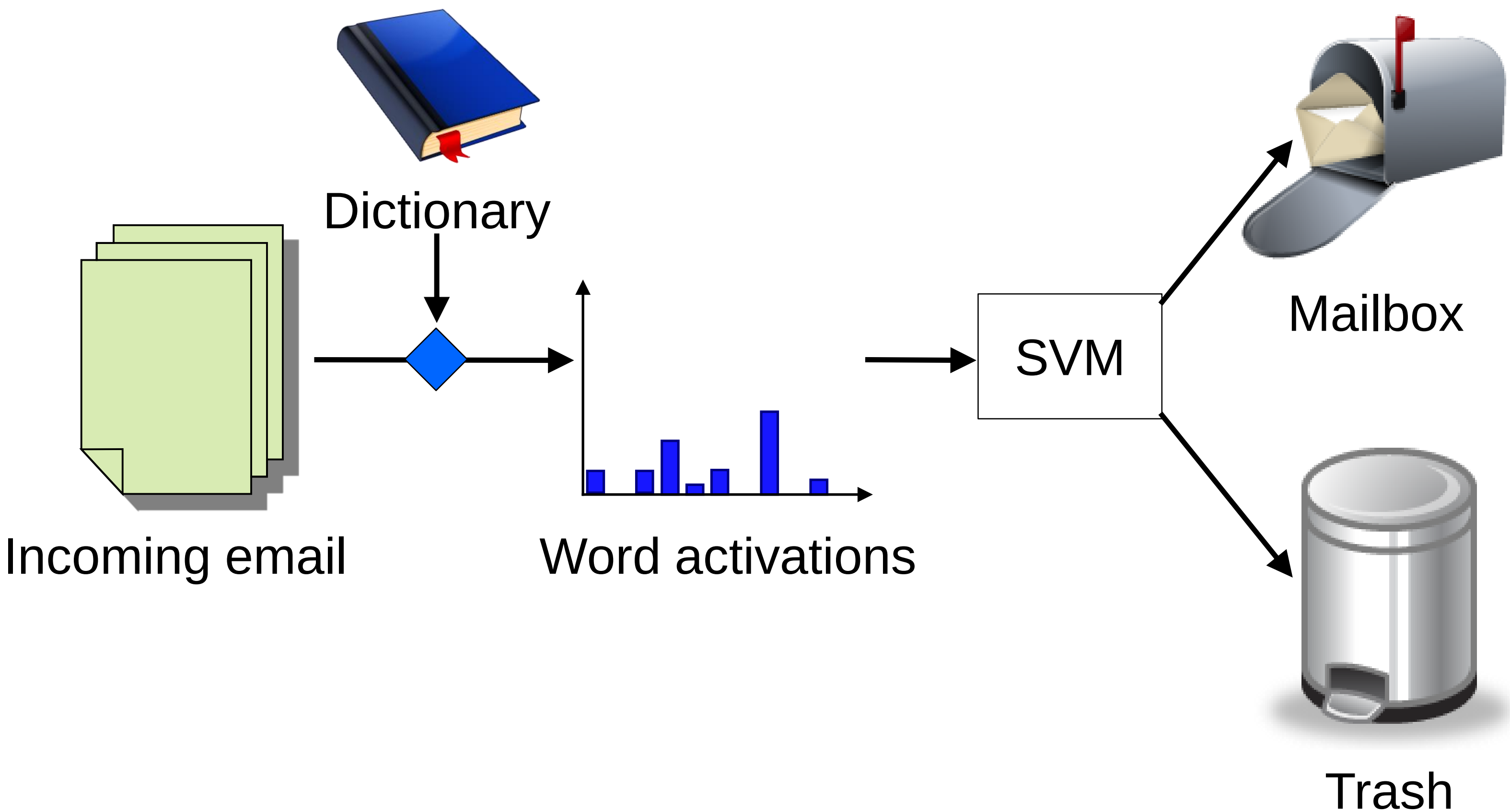
# Example Application: Text Classification

- Results:

| | Bayes | Rocchio | C4.5 | k-NN | SVM (poly) degree $d =$ | | | | | SVM (rbf) width $\gamma =$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1 | 2 | 3 | 4 | 5 | 0.6 | 0.8 | 1.0 | 1.2 |
| earn | 95.9 | 96.1 | 96.1 | 97.3 | 98.2 | 98.4 | **98.5** | 98.4 | 98.3 | **98.5** | 98.5 | 98.4 | 98.3 |
| acq | 91.5 | 92.1 | 85.3 | 92.0 | 92.6 | 94.6 | **95.2** | 95.2 | 95.3 | 95.0 | 95.3 | 95.3 | **95.4** |
| money-fx | 62.9 | 67.6 | 69.4 | 78.2 | 66.9 | 72.5 | 75.4 | 74.9 | **76.2** | 74.0 | 75.4 | **76.3** | 75.9 |
| grain | 72.5 | 79.5 | 89.1 | 82.2 | 91.3 | 93.1 | **92.4** | 91.3 | 89.9 | **93.1** | 91.9 | 91.9 | 90.6 |
| crude | 81.0 | 81.5 | 75.5 | 85.7 | 86.0 | 87.3 | 88.6 | **88.9** | 87.8 | **88.9** | 89.0 | 88.9 | 88.2 |
| trade | 50.0 | 77.4 | 59.2 | 77.4 | 69.2 | 75.5 | 76.6 | 77.3 | **77.1** | 76.9 | 78.0 | **77.8** | 76.8 |
| interest | 58.0 | 72.5 | 49.1 | 74.0 | 69.8 | 63.3 | 67.9 | 73.1 | **76.2** | 74.4 | 75.0 | **76.2** | 76.1 |
| ship | 78.7 | 83.1 | 80.9 | 79.2 | 82.0 | 85.4 | 86.0 | **86.5** | 86.0 | **85.4** | 86.5 | 87.6 | 87.1 |
| wheat | 60.6 | 79.4 | 85.5 | 76.6 | 83.1 | 84.5 | 85.2 | **85.9** | 83.8 | **85.2** | 85.9 | 85.9 | 85.9 |
| corn | 47.3 | 62.2 | 87.7 | 77.9 | 86.0 | 86.5 | 85.3 | **85.7** | 83.9 | **85.1** | 85.7 | 85.7 | 84.5 |
| microavg. | **72.0** | **79.9** | **79.4** | **82.3** | 84.2 | 85.1 | 85.9 | 86.2 | 85.9 | 86.4 | 86.5 | 86.3 | 86.2 |
| | | | | | combined: **86.0** | | | | | combined: **86.4** | | | |

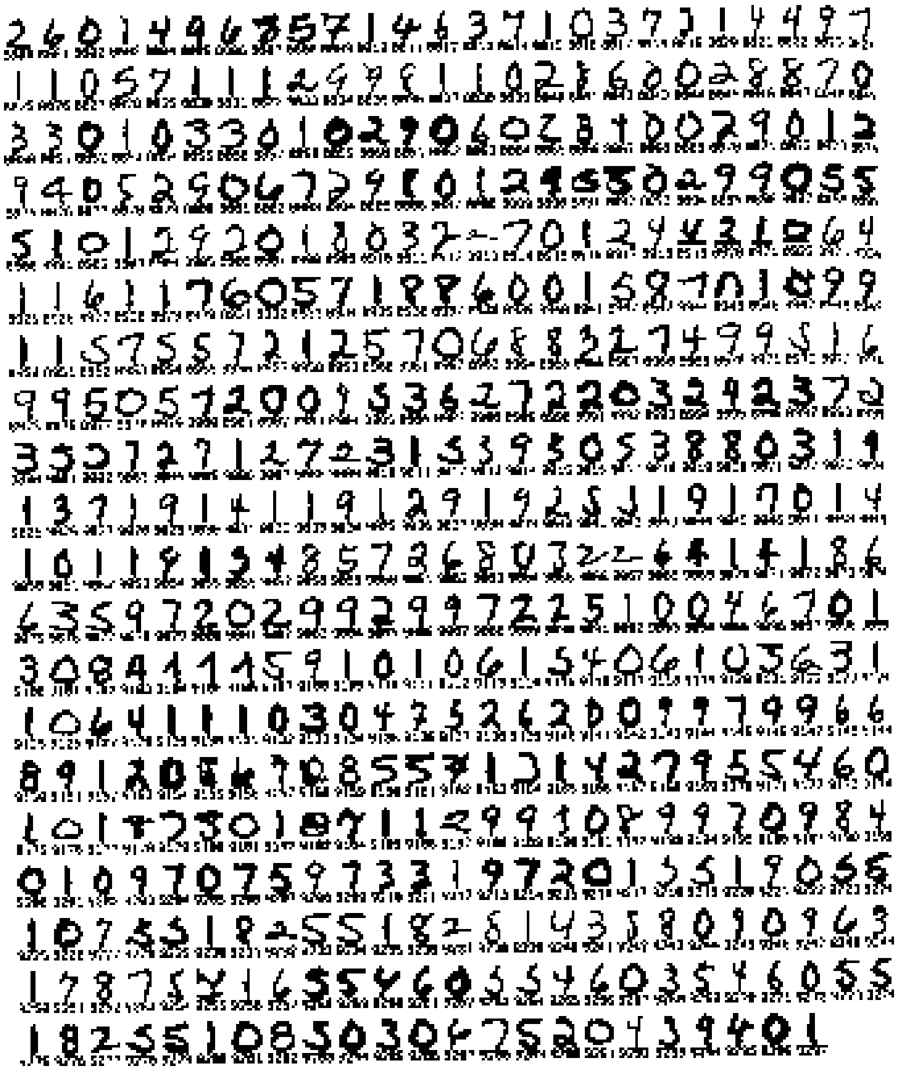
# Example Application: Text Classification

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



# 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 ($\sigma$=0.3, 291 support vectors)

# Example Application: OCR

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

| degree of polynomial | dimensionality of feature space | support vectors | raw error |
|---|---|---|---|
| 1 | 256 | 282 | 8.9 |
| 2 | $\approx 33000$ | 227 | 4.7 |
| 3 | $\approx 1 \times 10^6$ | 274 | 4.0 |
| 4 | $\approx 1 \times 10^9$ | 321 | 4.2 |
| 5 | $\approx 1 \times 10^{12}$ | 374 | 4.3 |
| 6 | $\approx 1 \times 10^{14}$ | 377 | 4.5 |
| 7 | $\approx 1 \times 10^{16}$ | 422 | 4.5 |

# Example Application: Object Detection

Real-time capable!

- Sliding-window approach


→  →

Obj./non-obj.
Classifier

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

[Dalal & Triggs, CVPR 2005]

# Example Application: Pedestrian Detection


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

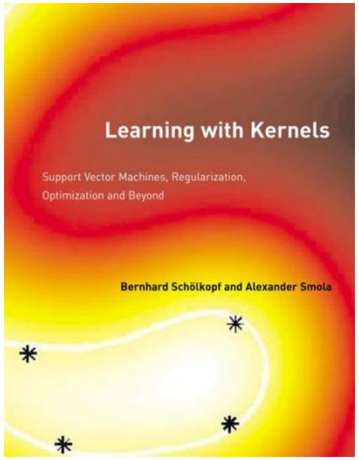
# Many Other Applications

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

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

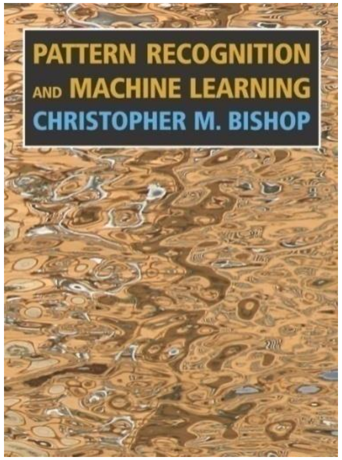
# References and Further Reading

- More information on SVMs can be found in Chapter 7.1 of Bishop's book. You can also look at Schölkopf & Smola (some chapters available online).




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

B. Schölkopf, A. Smola
Learning with Kernels
MIT Press, 2002
http://www.learning-with-kernels.org/




- A more in-depth introduction to SVMs is available in the following tutorial:
  - C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, Vol. 2(2), pp. 121-167 1998.