

Machine Learning 1 – Fundamentals

Support Vector Machine

Prof. Dr. J. M. Zöllner, M.Sc. Marcus Fechner, M.Sc. Nikolai Polley



Outline

- Recapitulation & Motivation
- Linear Support Vector Machine
- Nonlinear Support Vector Machine
- Version Space and Structural Risk Minimization
- Extensions
- Applications
- Discussion

Principle – Ockham's Razor

- **William of Ockham:** Important medieval philosopher and theologian, born in Ockham in 1287 and died in Munich in 1347.
 - accused of heresy
 - Revocation of teaching license
- **Occam's razor**
 - **Latin:** „*Entia non sunt multiplicanda sine necessitate.*“
 - **German:** „*Löse nie ein Problem komplizierter als nötig, denn die einfachste, richtige Erklärung, ist die Beste.*“
 - **English:** „*Entities should not be multiplied beyond necessity.*“
- Oftentimes interpreted as:
“other things being equal, simpler explanations are generally better than more complex ones”
- Learning theory and successful practice aims to formalize and explain the problem and its solution. Additionally, it tries to explain "simple" and "better"



Formal Description – Supervised Learning

- **Learning:** Find optimal hypothesis $h_{opt}: X \rightarrow Y$ in hypothesis space H
 - Training dataset $\{(x_1, y_1), \dots, (x_N, y_N)\}$
 - Input data $x_i \in X$
 - Output data $y_i \in Y$
 - Taken from (unknown) probability density function $p(x, y)$
 - Learning method: Optimization
- **Types of supervised Learning:** defined by space $X \times Y$
 - $\mathbb{R}^n \times \{y_1, \dots, y_k\}$ - **Classification:** Output data is discrete
 - $\mathbb{R}^n \times \mathbb{R}^m$ - **Regression:** Output data is continuous
 - $\{Atr_1, \dots, Atr_n\}^n \times \{true, false\}$ - **Concept:** Input is discrete, output true/false
 - ... (see e.g. [Goodfellow])

Estimation of Test-Error

- According to Vapnik with probability η :

$$\mathcal{L}(h_\theta) \leq \hat{\mathcal{L}}(h_\theta) + \sqrt{\dots \frac{VC(h_\theta)}{N} \dots}$$

← As small as possible

← As large as possible

- whereas:

- $VC(h_\theta)$ – VC-dimension of learning system
- N – number of learning instances
- $\hat{\mathcal{L}}(h_\theta)$ – empirical error, dependent from $VC(h_\theta)$ und N
- $\mathcal{L}(h_\theta)$ – real error

- Success of learning is dependent on:

- Capacity of learning system (as small as possible)
- Optimization method (as good as possible)
- Learning instances in dataset (representative for real world and as many as possible)

Structural Risk Minimization

- **Goal:** Find a solution for

$$\min_{H^n} \left(\hat{\mathcal{L}}(h_\theta) + \sqrt{\dots \frac{VC(h_\theta)}{N} \dots} \right)$$

→ Find $VC(h_\theta)$ („learning system“), N („number instances“), and θ („minimum of empirical error“)

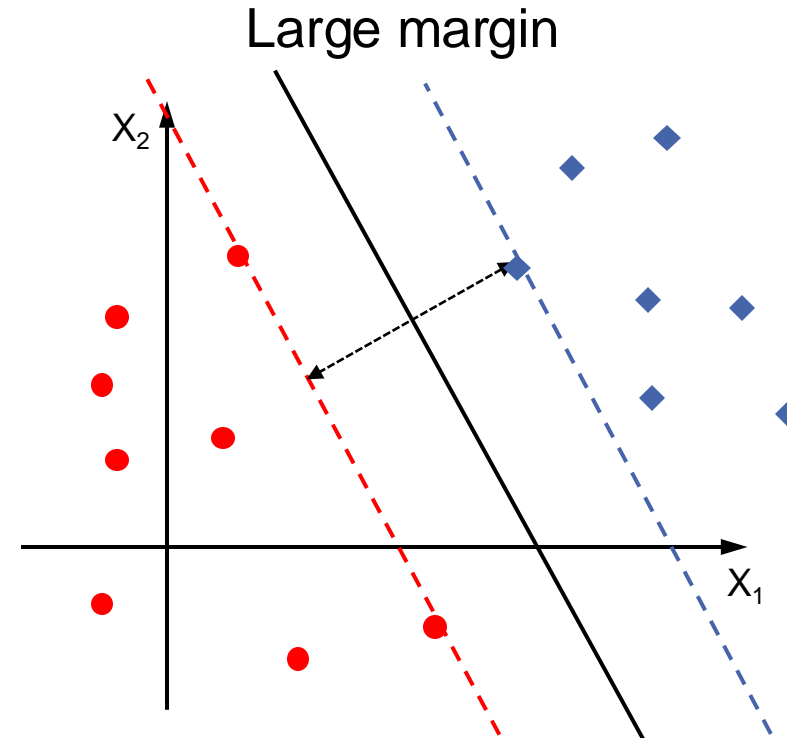
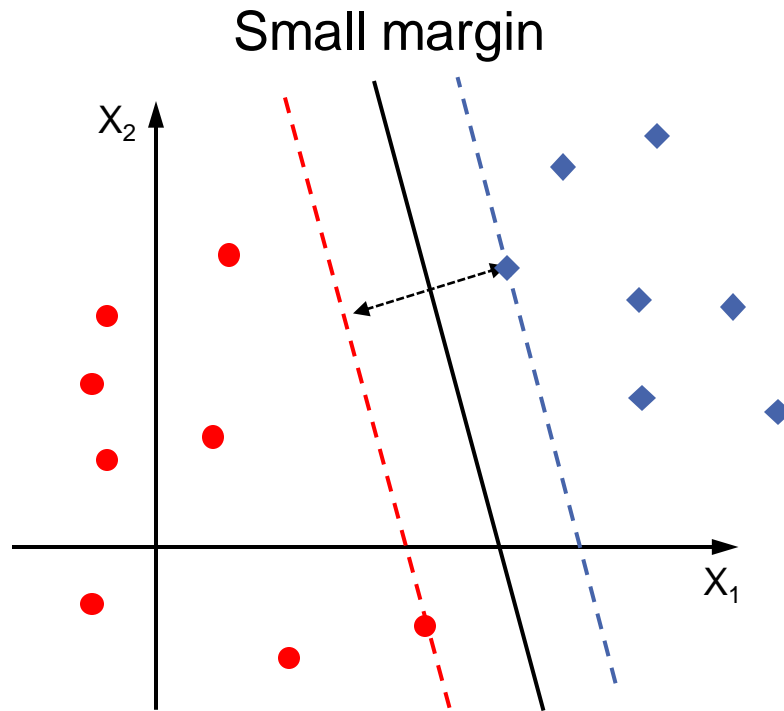
- **Ideal solution (meta-algorithm) :** Minimize whole sum, not individual summands
 - Use learning system that can create multiple hypotheses spaces with changing complexities
 - Structure hypothesis spaces with regards to their VC-dimension

$$H^1 \subset H^2 \subset \dots \subset H^n, VC(h_\theta^i) \leq VC(h_\theta^{i+1})$$
 - Iterate over all hypothesis spaces H^i
 - Train learning system to receive optimal empirical error $\hat{\mathcal{L}}(h_\theta)$ for given hypothesis space
 - If sum is minimized → stop and use this hypothesis.

Outline

- Recapitulation & Motivation
- Linear Support Vector Machine
- Nonlinear Support Vector Machine
- Version Space and Structural Risk Minimization
- Extensions
- Applications
- Discussion

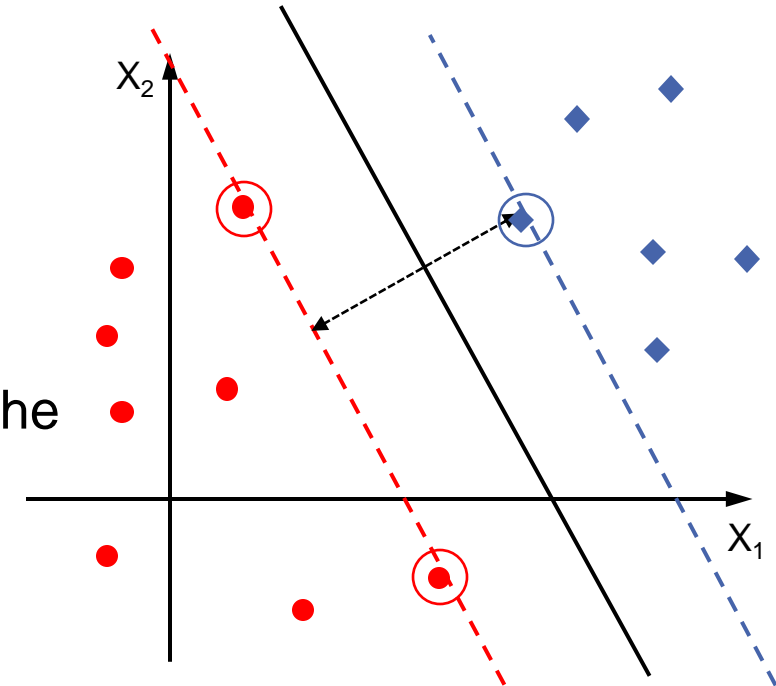
Linear Support Vector Machine



- **Problem:** Classification – There are many possibilities to separate the two sets (red and blue), but what is the optimal solution for the problem?
- **Intuition:** Size of the margin determines the generalization capability
- **Solution:** Find the best separating line/hyperplane with maximum margin to the classes

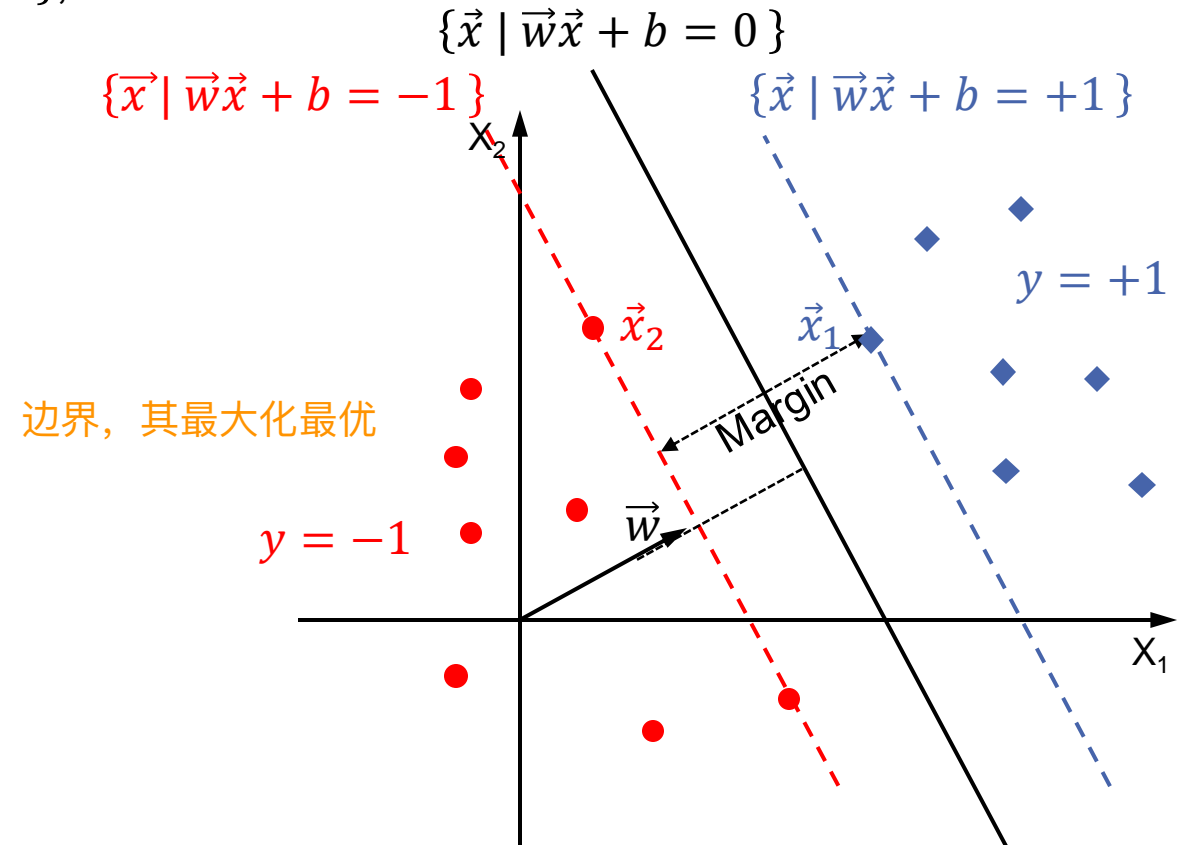
Support Vectors - Intuition

- **Question:** Which data points influence the optimal hyperplane?
 - All data points? → Linear regression, neural networks
 - Or just "difficult" data points close to the decision boundary? → SVM
- **Support Vectors**
 - Data points of the training dataset that have a direct impact on the position of the hyperplane and would change it if removed.
 - Data points closest to the hyperplane
 - Data points that are the "hardest" to classify



SV – Method, Hyperplane

- Find the **Hyperplane** $\{\vec{x} \in X \mid \vec{w}\vec{x} + b = 0, (\vec{w}, b) \in X \times Y\}$, with maximum margin.
- **Parameters:** \vec{w}, b
- **Training data:** $\{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$
 - Input $\vec{x}_i \in X$
 - Output/Target $y_i \in Y$
- **Margin** (= Distance between closest points): $\frac{\vec{w}(\vec{x}_1 - \vec{x}_2)}{|\vec{w}|}$ 边界, 其最大化最优
- **Normalization:** $\vec{w}\vec{x}_1 + b = +1$
 $\vec{w}\vec{x}_2 + b = -1$
- **Then:** $\vec{w}(\vec{x}_1 - \vec{x}_2) = 2$
 $\frac{\vec{w}}{|\vec{w}|}(\vec{x}_1 - \vec{x}_2) = \frac{2}{|\vec{w}|}$



SV – Method, Formalization

■ **Optimal hyperplane condition:** $\min_{i=1\dots n} |\vec{w} \vec{x}_i + b| = 1$

■ Distance to the nearest point: $\frac{1}{|\vec{w}|}$

■ Distance between the 2 classes: $\frac{2}{|\vec{w}|}$ (should be maximized) 优化思路: maxima. this distance(类间距离)

■ With that we can calculate \vec{w} and b

■ **Decision function** for the linear classifier:

$$h_{\vec{w},b}(\vec{x}) = \text{sgn}(\vec{w}\vec{x} + b)$$

SV – Optimization, Problem

- Hyperplane with maximum margin
 - Maximize $\frac{2}{|\vec{w}|}$ → Minimize $|\vec{w}|^2$
 - Under the **conditions**: $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, i = 1 \dots n$
 - That means that the training data is correctly classified
- According to Vapnik, this is the **learning machine with the smallest possible VC dimension** (if the classes are linearly separable)

Constrained Optimization

- Continuous constrained optimization problem: $\min_{\vec{w}} \hat{\mathcal{L}}(\vec{w})$

- With constraints c_i :
 - $i \in \mathbb{N}_g: c_i(\vec{w}) = 0$
 - $i \in \mathbb{N}_u: c_i(\vec{w}) \geq 0$

- Where:

- $\hat{\mathcal{L}}$ is a continuously differentiable cost function
- \mathbb{N}_g Equality constraints (in SVM $\mathbb{N}_g = \emptyset$)
- \mathbb{N}_u Inequality constraints

- Introducing the **Lagrange function**: $L_P(\vec{w}, \vec{\alpha}) = \hat{\mathcal{L}}(\vec{w}) - \sum_{i \in \mathbb{N}_g \cup \mathbb{N}_u} \alpha_i c_i(\vec{w})$ KKT条件

- Transforms the constrained optimization problem into an unconstrained optimization problem
- Extends the original cost function by **constraints** c_i , multiplied with the **Lagrange multipliers** α_i
- Find the solution to the constrained optimization problem $\hat{\mathcal{L}}(\vec{w})$ by optimizing the unconstrained optimization problem $L_P(\vec{w}, \vec{\alpha})$

Original cost function

Constraints

ODS中的约束条件下的最优化求解

Saddle Point of the Lagrange Function

■ Saddle point conditions

对于 \mathbf{w} 和 b , 是最小值 (最小化目标函数)。

对于拉格朗日乘子 α_i , 是最大值 (最大化对偶问题)。

$$\vec{\alpha}^* = \arg \max_{\vec{\alpha}} \left[\min_{\vec{w}} L_p(\vec{w}, \vec{\alpha}) \right]$$

$$\vec{w}^* = \arg \min_{\vec{w}} \left[\max_{\vec{\alpha}} L_p(\vec{w}, \vec{\alpha}) \right]$$

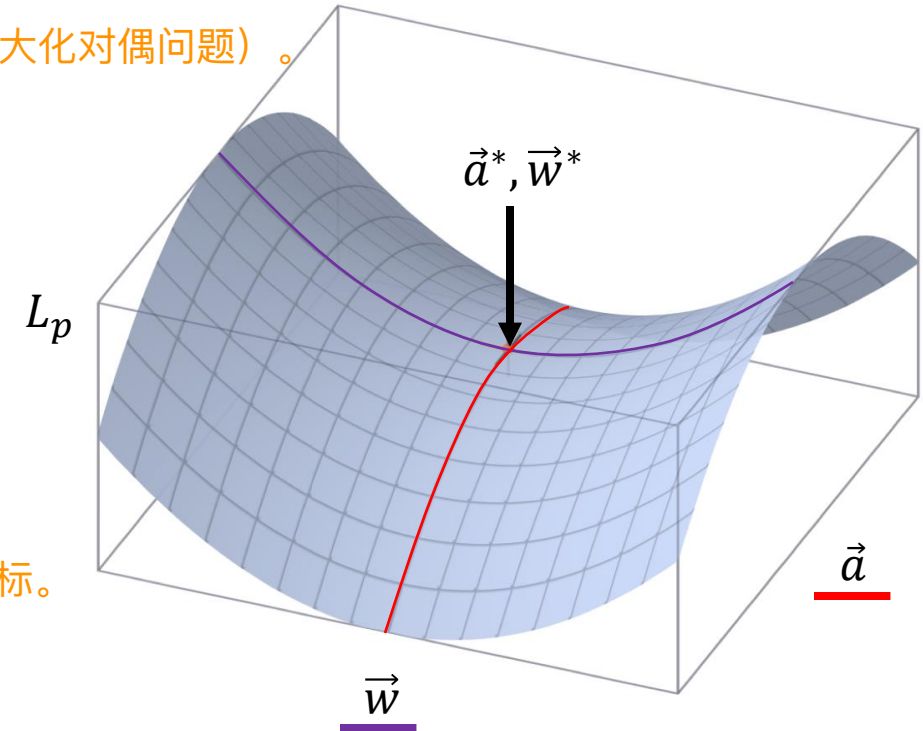
■ Find the (unique) saddle point of the function L_p

■ Minimum of L_p wrt. \vec{w} (and also b)

鞍点是拉格朗日函数的唯一解, 它同时满足分类边界的约束和最大化边界的目标。

$$\nabla_{\vec{w}} L_p(\vec{w}^*, \vec{\alpha}^*) = 0$$

■ Maximum of L_p wrt. $\alpha_1, \dots, \alpha_n$



Lagrange Function SVM

■ Equivalent problem (*Primary optimization problem*):

■ Lagrange function for SVM

$$L_p(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} |\vec{w}|^2 - \sum_{i=1}^n \alpha_i (y_i (\vec{w} \vec{x}_i + b) - 1)$$

$$L_p(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} |\vec{w}|^2 - \sum_{i=1}^n \alpha_i (y_i (\vec{w} \vec{x}_i + b)) + \sum_{i=1}^n \alpha_i$$

■ with the positive Lagrange multipliers

$$\vec{\alpha} = (\alpha_1, \dots, \alpha_n), \quad \alpha_1, \dots, \alpha_n \geq 0$$



Minimization: Lagrange Function SVM

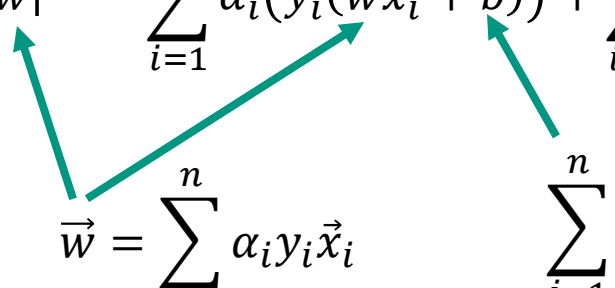
- Find the (unique) saddle point of the function L_P
- Saddle point conditions:
 - Minimum of L_P wrt. \vec{w}, b
 - $\frac{\partial L}{\partial b} = \sum_{i=1}^n y_i \alpha_i = 0$ 该条件确保支持向量的分类边界正确。
 - $\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^n \alpha_i y_i \vec{x}_i = 0 \Rightarrow \vec{w} = \sum_{i=1}^n \alpha_i y_i \vec{x}_i$ 由此得到权重, 该条件确保支持向量的分类边界正确。
 - Maximum of L_P wrt. $\alpha_1, \dots, \alpha_n$ 最大化拉格朗日函数, 确定支持向量对应的拉格朗日乘子 α_i 。
- It follows that: \vec{w} is directly calculable from α_i, \vec{x}_i, y_i
 - $b = \frac{1}{2}(\vec{x}_+ + \vec{x}_-) \vec{w}$ can later be calculated from any 2 support vectors \vec{x}_+, \vec{x}_-

Minimization: Lagrange – Dual Optimization Problem

- **Problem:** Primary optimization problem $L_p(\vec{w}, b, \vec{\alpha})$ is dependent on many variables $\vec{w}, b, \vec{\alpha}$
- **Idea:** Take advantage of the saddle point conditions to substitute \vec{w}, b
- **Solution:** Dual problem formulation $L_d(\vec{\alpha})$ is **only dependent on $\vec{\alpha}$**

■ **Primary Problem:**
$$\max_{\vec{\alpha}} \min_{\vec{w}, b} L_p(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} |\vec{w}|^2 - \sum_{i=1}^n \alpha_i (y_i (\vec{w} \vec{x}_i + b)) + \sum_{i=1}^n \alpha_i$$

$\vec{w} = \sum_{i=1}^n \alpha_i y_i \vec{x}_i$
 $\sum_{i=1}^n y_i \alpha_i = 0$



■ **Dual Problem:**
$$\max_{\vec{\alpha}} L_D(\alpha_i) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)$$

← Dot product

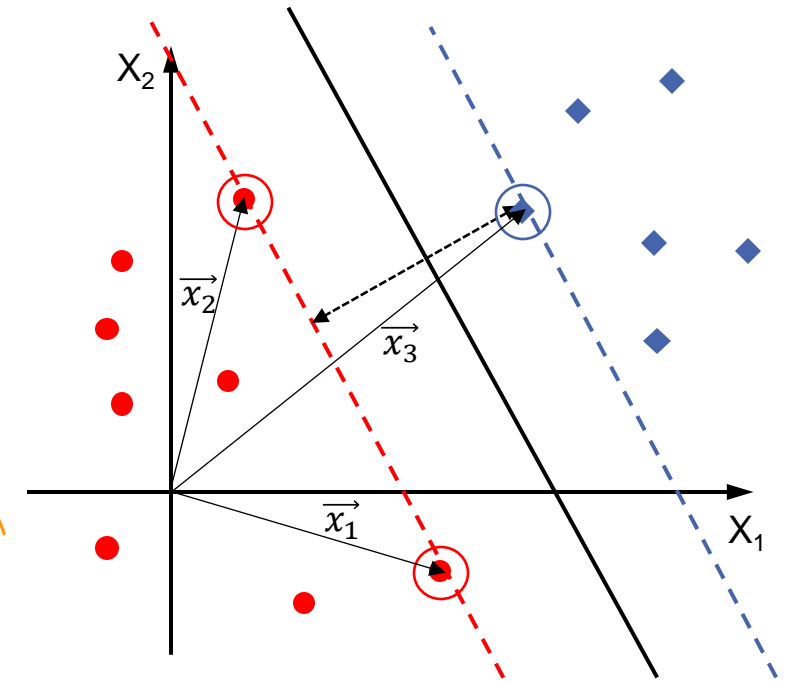
conditioned on $\sum_{i=1}^n y_i \alpha_i = 0$ and $\alpha_i \geq 0$

Support Vectors – Formal

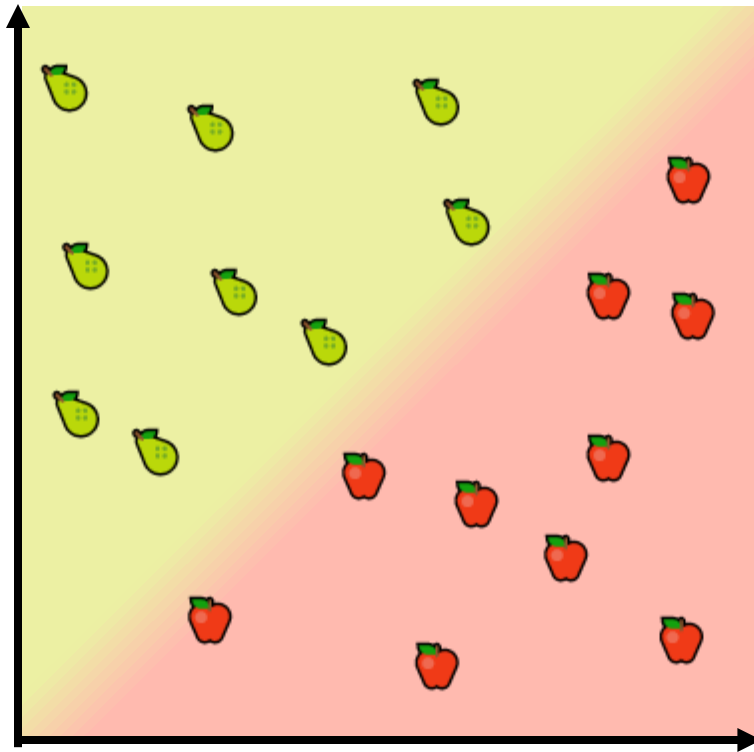
- After solving the optimization problem, the saddle point conditions are met
 - That means most $\alpha_i = 0$
 - Except for the most important data points \vec{x}_i
 - Here for example $\vec{x}_1, \vec{x}_2, \vec{x}_3$

分类边界由支持向量唯一确定。，非支持向量虽然有助于训练，但对最终分类结果的影响很小

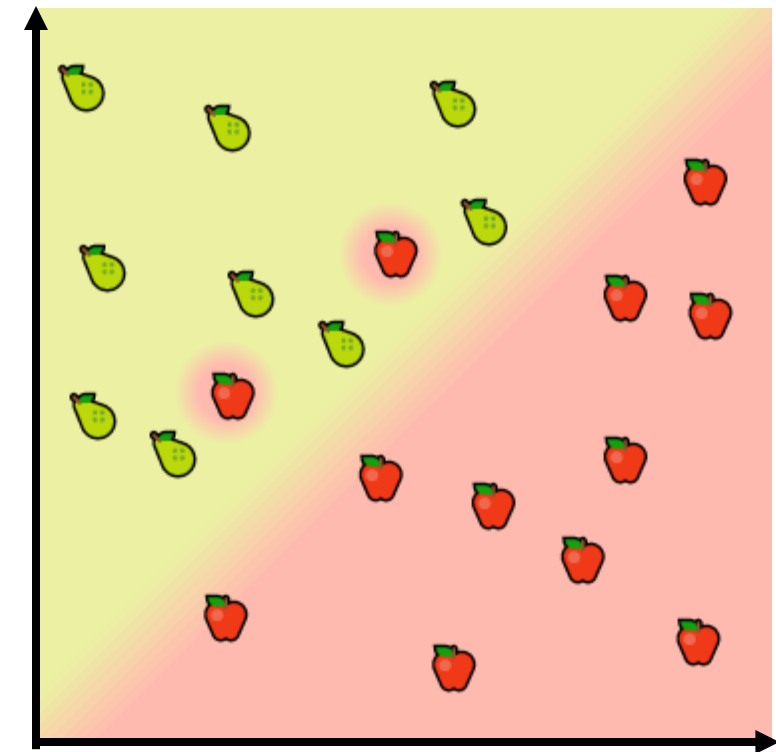
- **Support vectors:** All \vec{x}_i with $\alpha_i > 0$
 - The optimal parameters \vec{w} are a linear combination of the most important data points \vec{x}_i (support vectors)
 - Optimal parameters: $\vec{w} = \sum_{i=1}^n \alpha_i y_i \vec{x}_i$
 - **Support vectors are closest to the hyperplane**



Extension to Nonlinear Problems



Linear separable data



Nonlinear separable data
(error, noise)

Outline

- Recapitulation & Motivation
- Linear Support Vector Machine
- Nonlinear Support Vector Machine
- Version Space and Structural Risk Minimization
- Extensions
- Applications
- Discussion

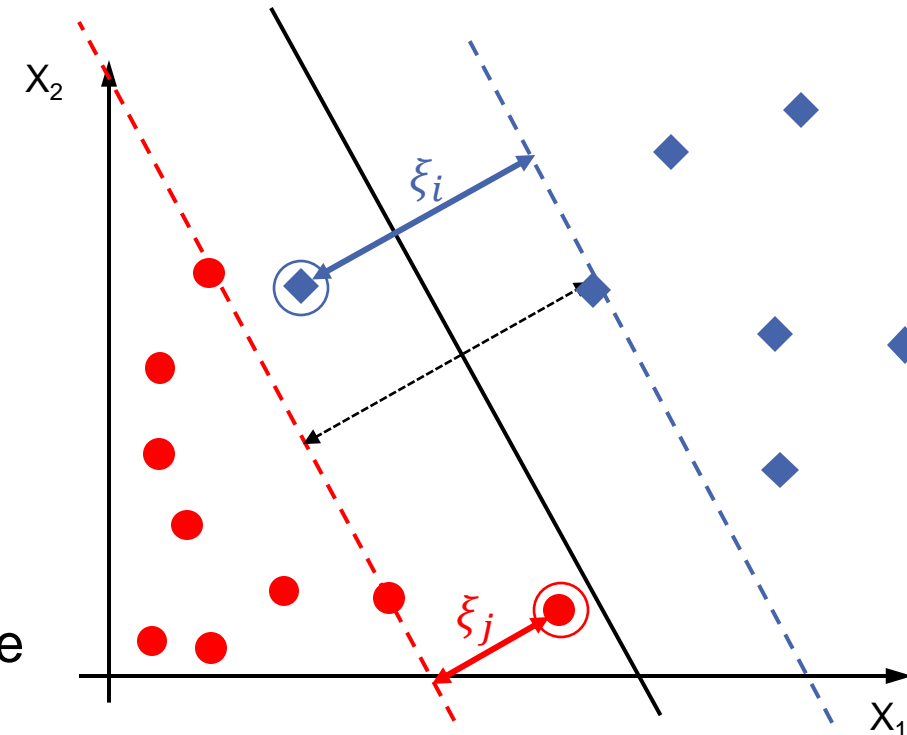
Soft Margin SVM

Idea

- Allow a small number of wrong classifications
- Leads to a larger margin and higher generalization

Modification of boundary conditions

- Introduce slack variable ξ_i for every \vec{x}_i .
- $0 \leq \xi_i < 1$, if \vec{x}_i is **correctly** classified
- $\xi_i > 1$, if \vec{x}_i **not correctly** classified
- ξ_i specifies how far \vec{x}_i lies on the wrong side of the margin.



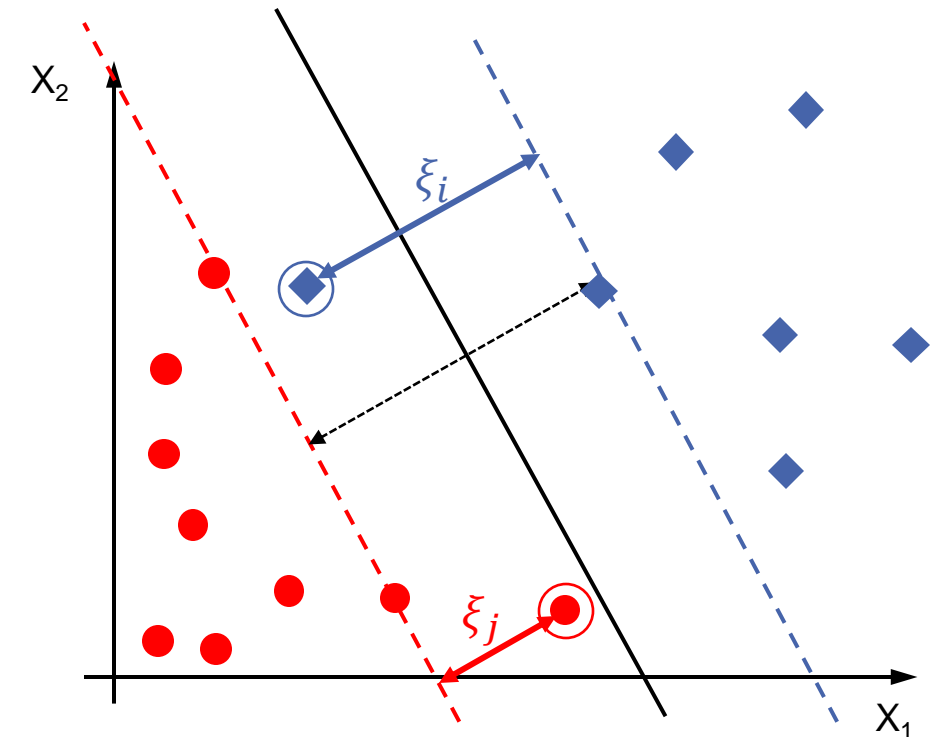
$$y_i(\vec{w}\vec{x}_i + b) \geq 1 - \xi_i \quad i = 1 \dots n, \quad \xi_i \geq 0$$

Generalized Optimal Hyperplane

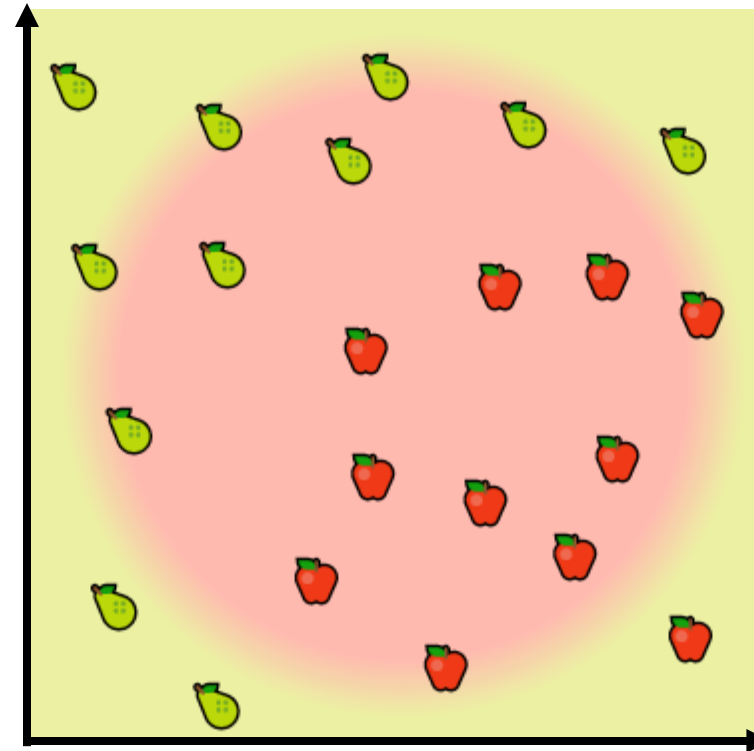
- **Minimize:** $\min_{\vec{w}, b, \xi_i} \frac{1}{2} |\vec{w}|^2 + C(\sum_{i=1}^n \xi_i)^p$
- **Conditions:** $\xi_i \geq 0$ und $y_i(\vec{w}\vec{x}_i + b) \geq 1 - \xi_i$
- **Solution:** Lagrange - Method
- **Idea behind C :**
 - Regularization parameter
 - C – Big \rightarrow Wrong classification are severely penalized
 - C – Small \rightarrow Wrong classification are slightly penalized

c 大: 强调分类错误的惩罚, 倾向于硬间隔 (Hard Margin)。

c 小: 允许更多分类错误, 倾向于软间隔 (Soft Margin)。



Complex Cases



Nonlinear separable data

Nonlinear Kernel Methods (SVM)

Idea

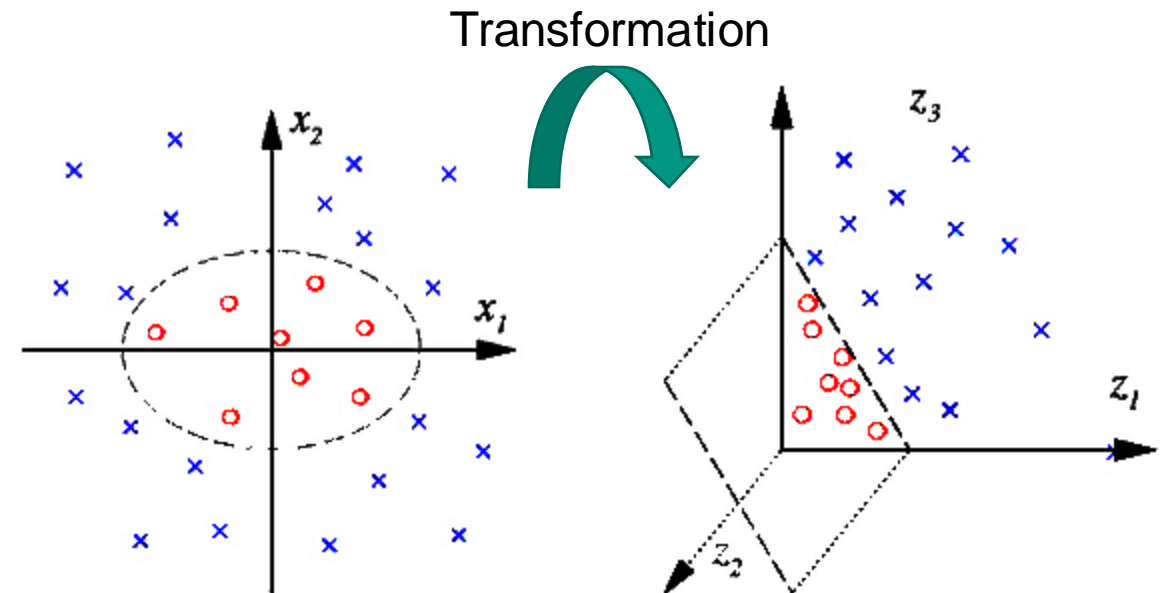
- Transform the data into another (higher-dimensional) space where the data can be separated linearly and solve the problem there
- In case of classification: Linear separation in the transformed space

Example: $\phi: \mathbb{R}^n \rightarrow \mathbb{R}^m$ often $m \gg n$

- Monomial-transformation

$$\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$(x_1, x_2) \rightarrow (z_1, z_2, z_3) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$$



Nonlinear SVM

■ **Before:** $\max_{\vec{\alpha}} L_D(\alpha_i) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)$

■ **Now:** $\max_{\vec{\alpha}} L_D(\alpha_i) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\phi(\vec{x}_i) \cdot \phi(\vec{x}_j))$

■ **Problem:** Transformation and dot product in high-dimensional spaces is computationally challenging and time-consuming.

■ **Solution:** Kernel function $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$

■ We don't need to know or calculate the transformation $\phi(\cdot)$!

Kernel-Trick

- **Kernel function** $K(\vec{x}, \vec{y})$: Defines a dot product / similarity measure in the transformed space without explicitly calculating it.
- **Kernel-trick**: Instead of calculating the complex nonlinear transformation $\phi(\cdot)$ and the scalar product, we use a kernel that does this implicitly, thus saving a lot of computational operations.

隐式

- **Before:** $\max_{\vec{\alpha}} L_D(\alpha_i) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\phi(\vec{x}_i) \cdot \phi(\vec{x}_j))$

- **Now:** $\max_{\vec{\alpha}} L_D(\alpha_i) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j)$

- **Example Monomial-transformation:**

- $K(\vec{x}, \vec{y}) = \phi(\vec{x})\phi(\vec{y}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(y_1^2, \sqrt{2} y_1 y_2, y_2^2)$
 $= (x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + x_2^2 y_2^2)$
 $= (x_1 y_1 + x_2 y_2)^2 = (\vec{x} \vec{y})^2$

Kernel Functions

- **Dot Product:** $K(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y}$
- **Polynomial (Vovk):** $K(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y} + c)^d$
- **Radiale Basis-Function (RBF):** $K(\vec{x}, \vec{y}) = e^{-\frac{\|\vec{x}-\vec{y}\|^2}{2\sigma^2}}$
- **Sigmoid** (see neural networks): $K(\vec{x}, \vec{y}) = \tanh(\kappa(\vec{x} \cdot \vec{y}) + \theta)$
- many more application-specific functions

Outline

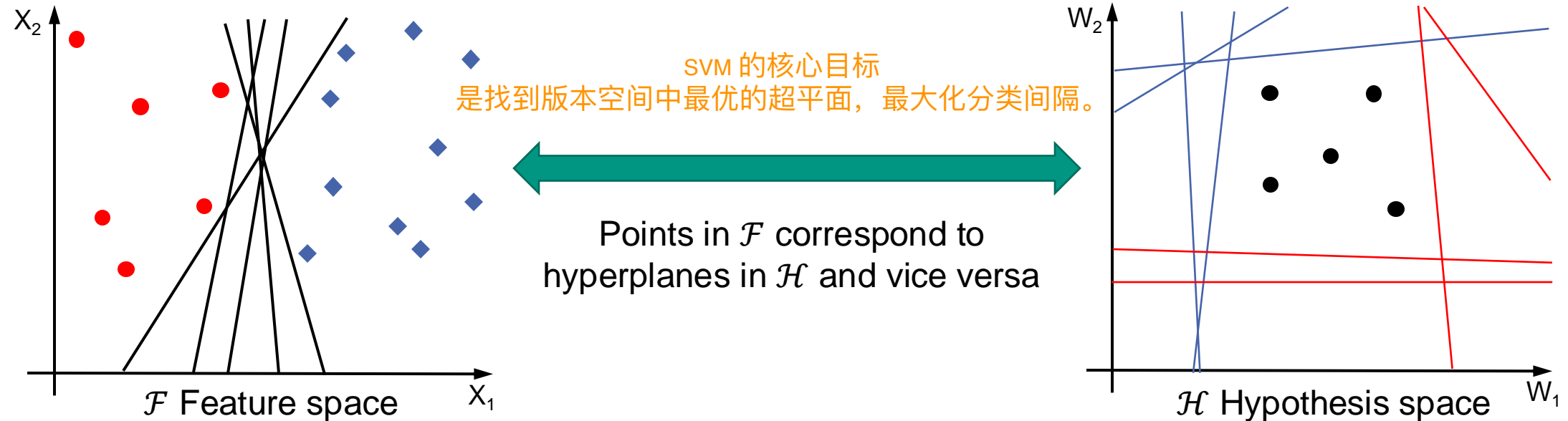
- Recapitulation & Motivation
- Linear Support Vector Machine
- Nonlinear Support Vector Machine
- Version Space and Structural Risk Minimization
- Extensions
- Applications
- Discussion

Version Space for SVM

版本空间是所有满足训练数据分类约束的超平面集合。

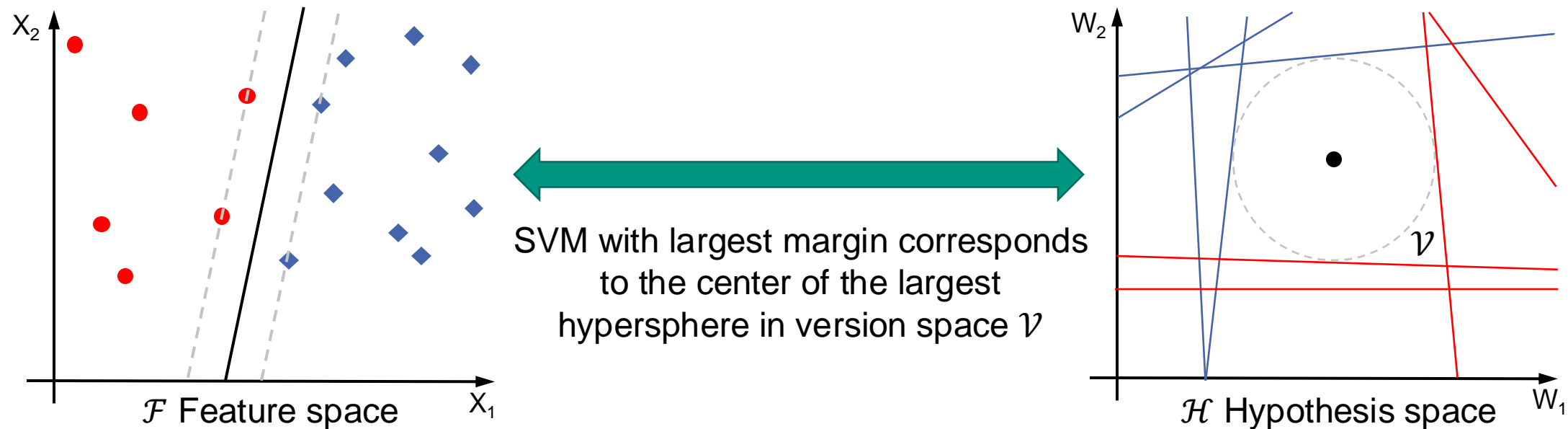
版本空间 是支持向量机优化过程中对超平面选择的限制条件

- **Special Property:** Duality of feature and hypothesis space (Vapnik, 1998)
- Conditions for correct classification: $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, i = 1 \dots n$
- Inequality can be interpreted by \vec{x}_i in **feature space** and by \vec{w} in **hypothesis space**
- Valid parameters \vec{w} lie in a half-space for every \vec{x}_i , thus in a sub-space of the hypothesis space (and for all \vec{x}_i in the average of the valid half-spaces)



Version Space for SVM

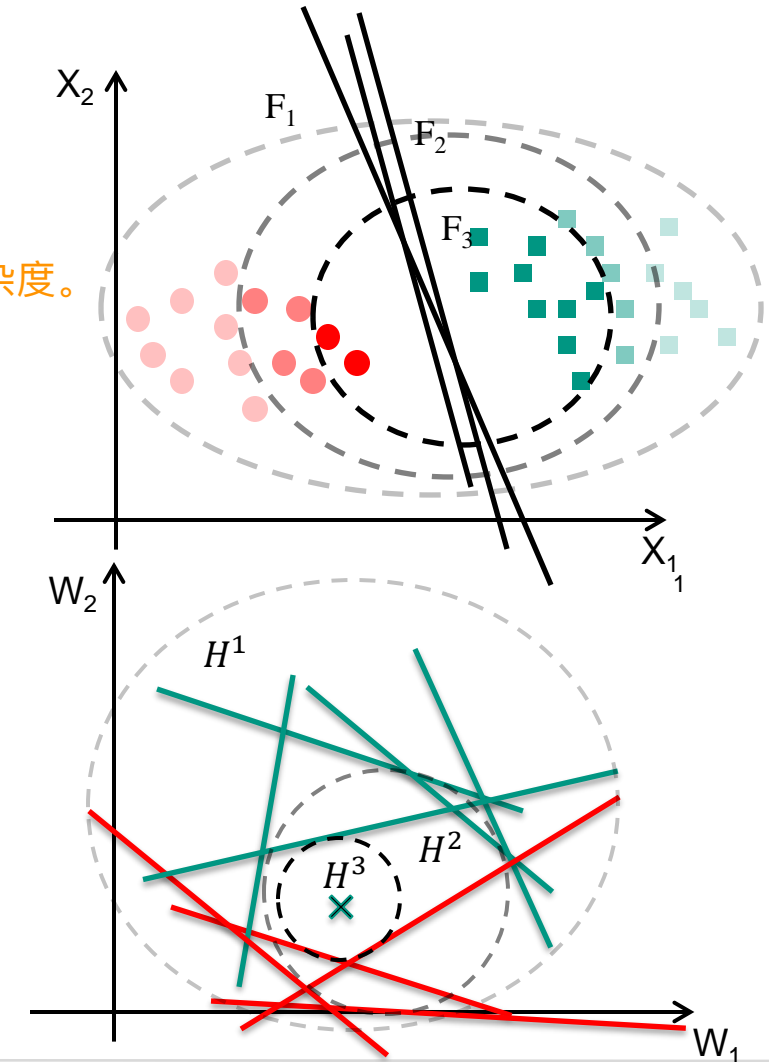
- Largest margin means that we are looking for the \vec{w} that has the maximum distance to all hyperplanes in \mathcal{H} , defined by the data points
- **Center of the remaining Hypersphere in \mathcal{H}**



SRM and SVM

- Search space (H^1, H^2, H^3) for the separation hyperplane is further and further restricted during saddle point search
- Irrelevant data (F^1, F^2, F^3) no longer matters
- **VC- Dimension:** $h \leq \approx D^2 |w|^2$
 - VC 维度 (Vapnik-Chervonenkis Dimension) 衡量假设空间的复杂度。
 - continuously reduced
 - where D (radius of the hypersphere) is given by relevant data/constraints in hypothesis space
- Therefore: $H^3 \subset H^2 \subset H^1$
 - svm 通过优化目标函数和添加约束条件 (如支持向量的定义), 不断减少 vc 维度
- Solution = Center of the remaining hypersphere in hypothesis space

svm 的优化过程实际上是在假设空间中寻找分类间隔最大的超平面。

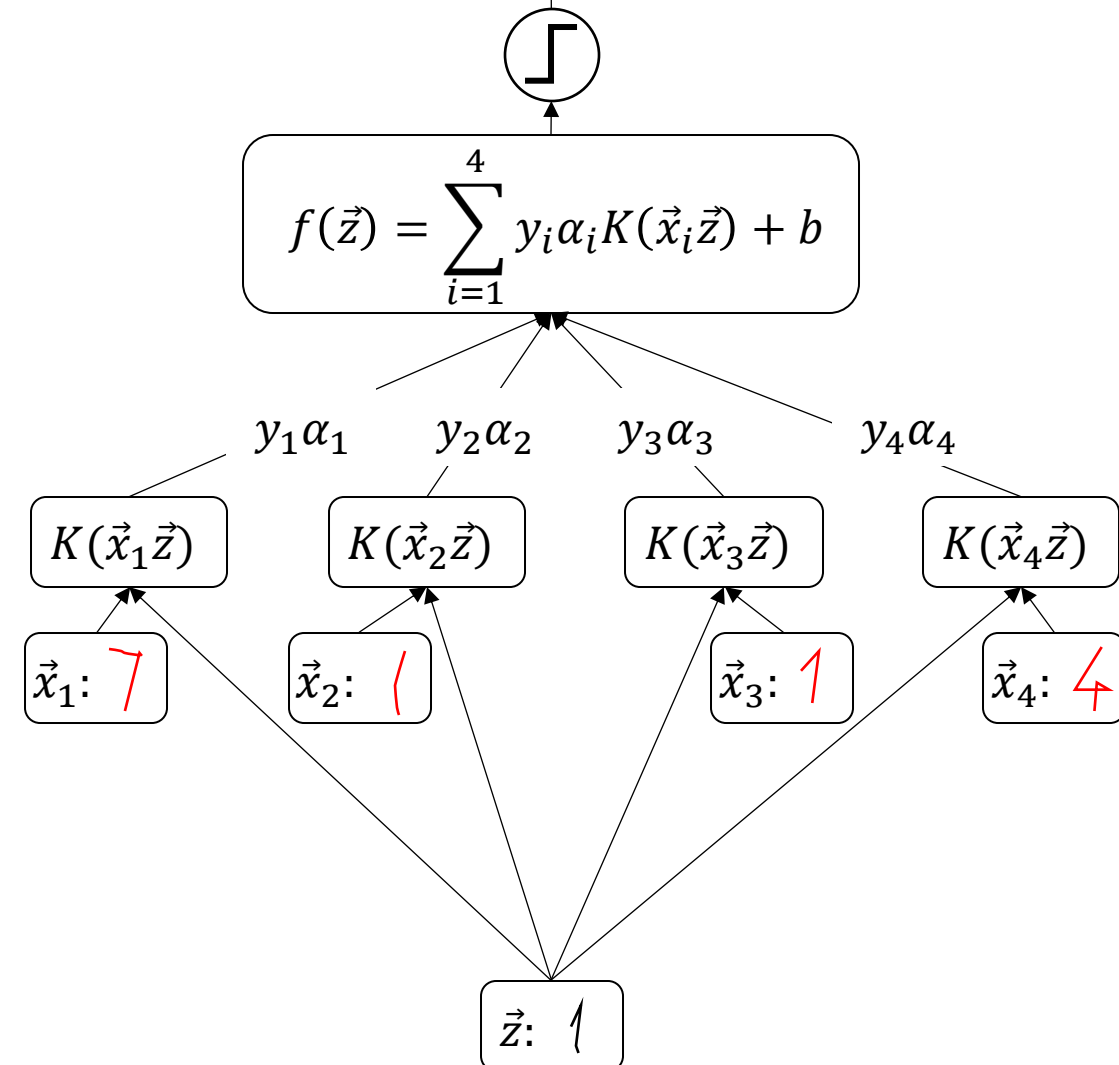


Outline

- Recapitulation & Motivation
- Linear Support Vector Machine
- Nonlinear Support Vector Machine
- Version Space and Structural Risk Minimization
- Extensions
- Applications
- Discussion

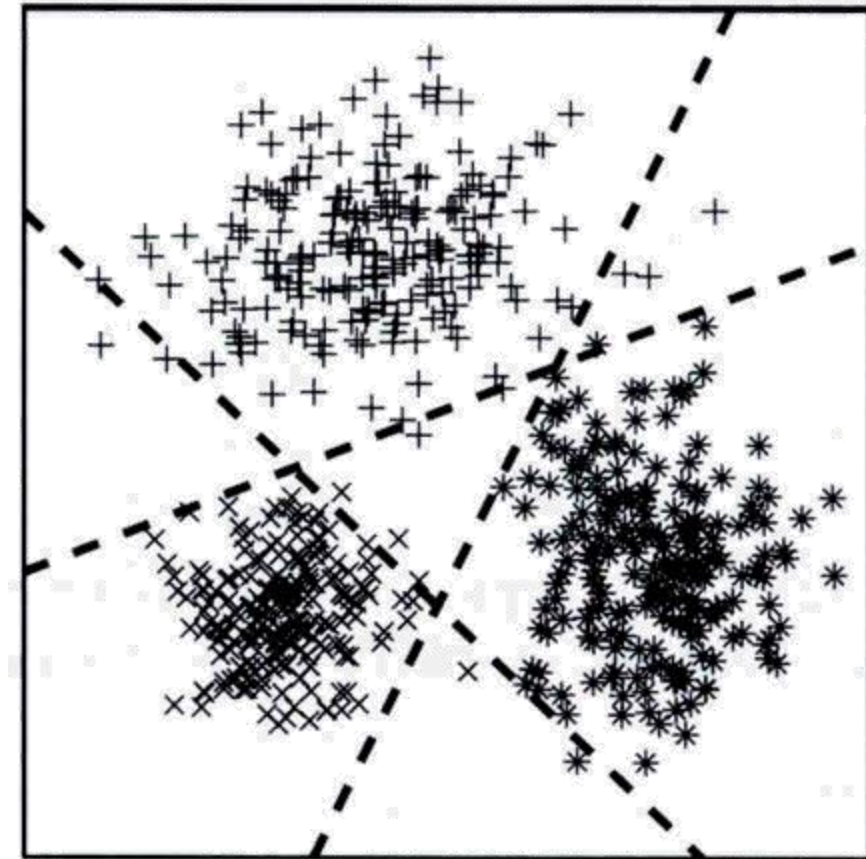
SVM Architecture - Similarity to Neural Networks?

- Example: Classification of input (bottom): „Digit 1, (yes/no)“
- One could assume a certain similarity to NNs:
 - Weights are learned
 - Kernel (with learned support vectors) and e.g.: tanh „Activation functions“ ~ Neuron
 - One „hidden“ layer
- But almost everything else is different
 - Optimization
 - Theoretical background
 - etc.



SVM Extensions - Classification on k Classes

- One – versus – All
 - k SVMs (one for each class)
 - Voting
- One – versus – One
 - $k(k-1)/2$ SVMs
 - Voting
- Multi-class-affiliation
 - k SVMs (one for each class)
 - Voting
- k – class SVM (according to Watkins)
 - One common optimization method
 - No voting

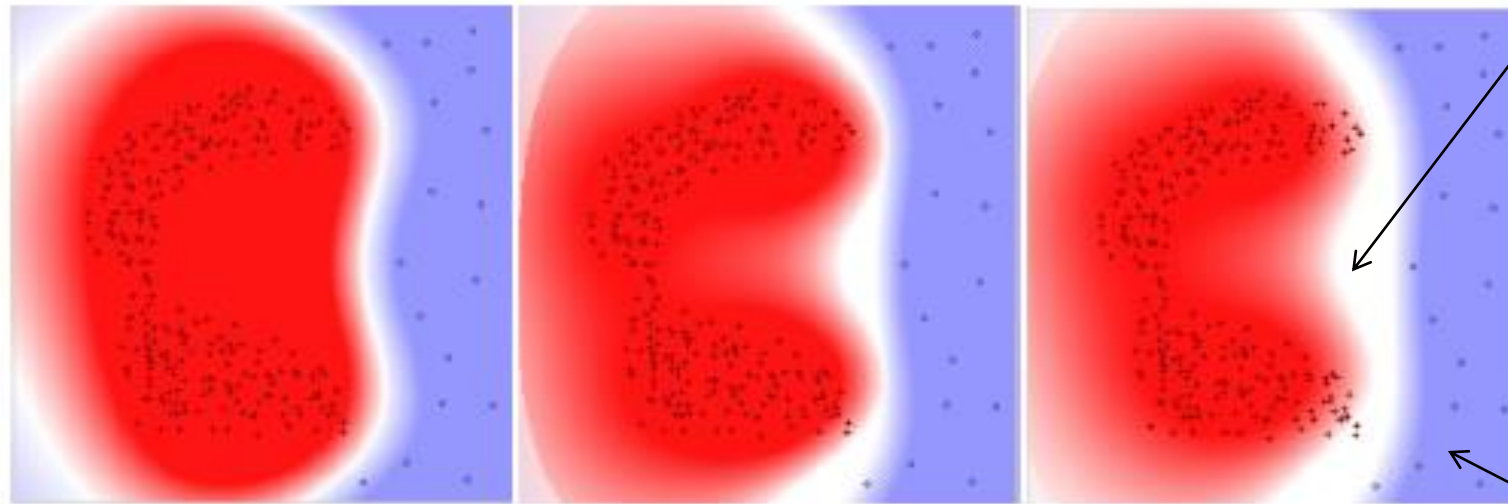


Weighted SVM

■ Minimize

$$\min_{\vec{w}, b, \xi_i} \frac{1}{2} |\vec{w}|^2 + C_+ \left(\sum_{y_i=+1, i=1}^{n_+} \xi_i \right)^p + C_- \left(\sum_{y_i=-1, i=1}^{n_-} \xi_i \right)^p$$

■ One C for each "positive and negative" class



$(C_-, C_+) = (10, 10)$

$(1, 10)$

$(0.1, 10)$

Small error

Probabilistic View of SVM

- Interpretation of the distance from the separation hyperplane as a classification probability

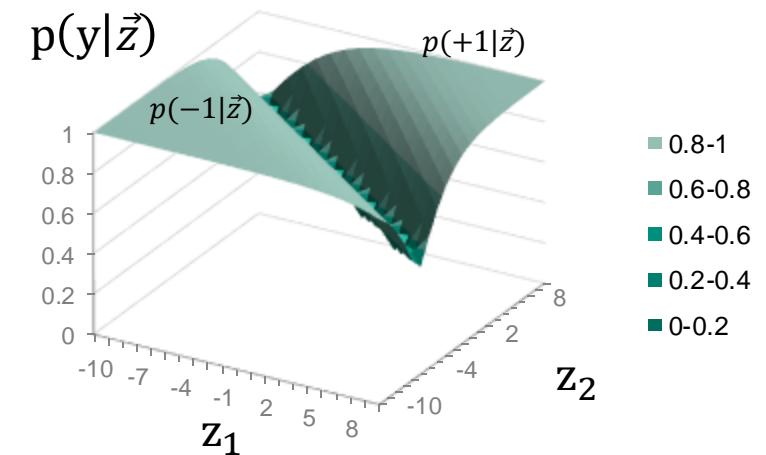
$$p(y|\vec{z}, \vec{w}, b) = \frac{1}{1 + e^{-yf(\vec{z})}} \quad \text{where} \quad f(\vec{z}) = \vec{w}\vec{z} + b$$

- Also allows for alternative training methods:

Maximizing classification probability and margin.

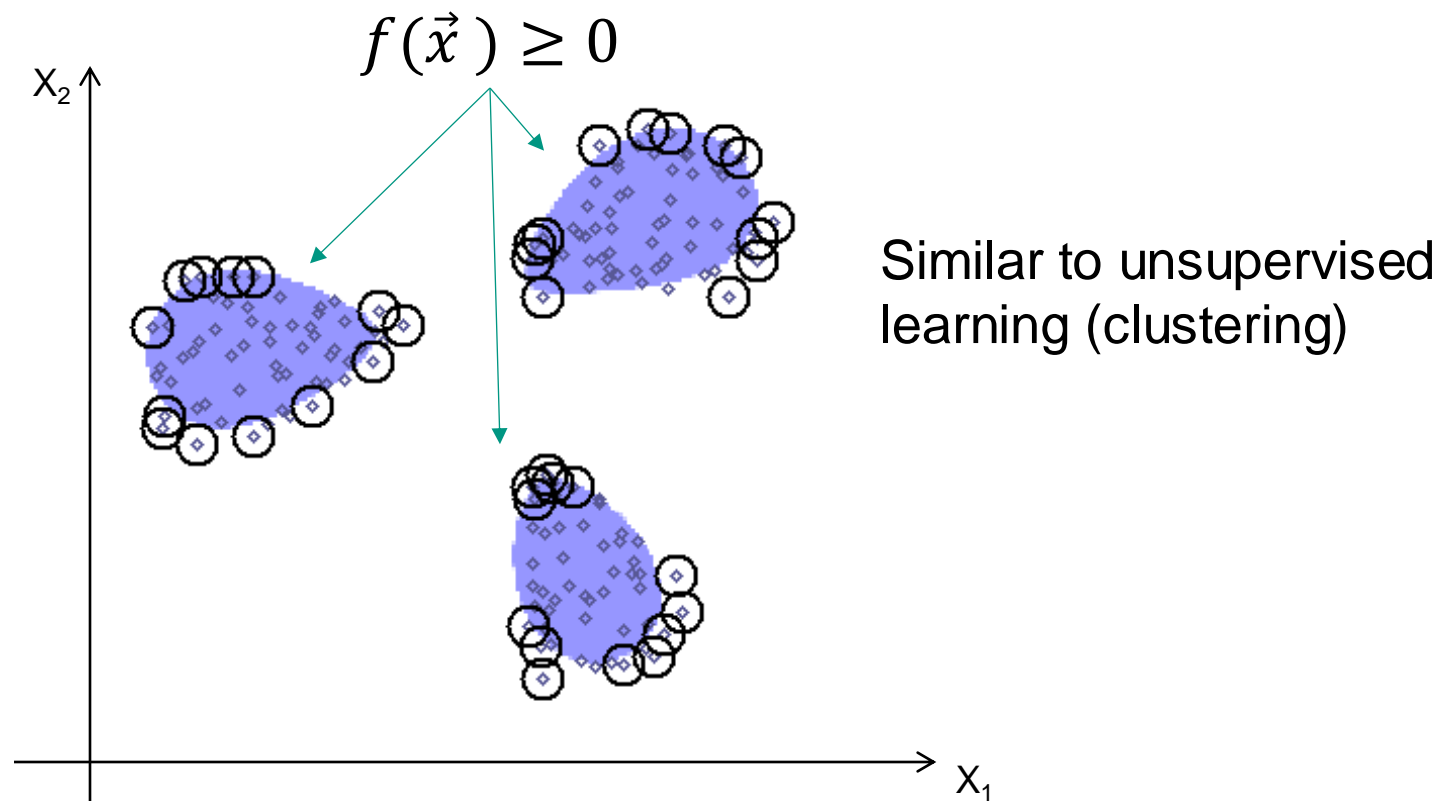
$$\max_{\vec{w}, b} \sum_{i=1}^n \log(p(y_i | \vec{x}_i, \vec{w}, b)) - \lambda |\vec{w}|^2$$

on the training data $\{ (\vec{x}_i, y_i) \mid i = 1..n \}$



Density Estimation

- **Goal:** A function f , that outputs a value > 0 for a small region, which has the most training examples, and outputs a value of 0 or < 0 in all other regions.



SVM Density Estimation

- Separates the learning examples from the origin in the transformed feature space
- Corresponds to a non-linear estimator in input space

■ Approach

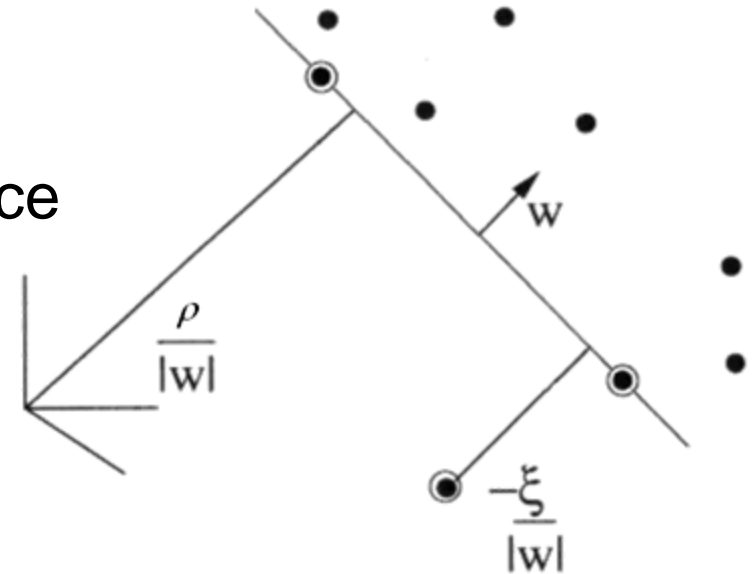
- Primary optimization problem:

$$\min_{\vec{w}, b, \xi_i} \frac{1}{2} |\vec{w}|^2 + \frac{1}{v n} \sum_{i=1}^n \xi_i - \rho$$

- Under the conditions:

$$\vec{w} \cdot \vec{x}_i \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1 \dots n$$

- Further calculation analogous to classification



Other Kernel Methods

- Simple algorithms can often be implemented as a kernel method
- Given vectors \vec{x}, \vec{y} and a transformation $\phi(\vec{x}), \phi(\vec{y})$ into another feature space
 - Then a kernel $K(\vec{x}, \vec{y})$ is defined through the dot product
$$K(\vec{x}, \vec{y}) = \phi(\vec{x})\phi(\vec{y})$$
 - For every kernel there is a transformation ϕ
- If $K(\vec{x}, \vec{y})$ is easily computable (see kernel functions), simple methods in the transformed space can often be used to solve complex problems in the origin space if only scalar products of the transformed data are present

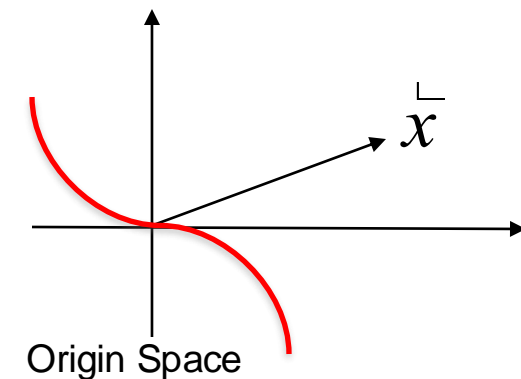
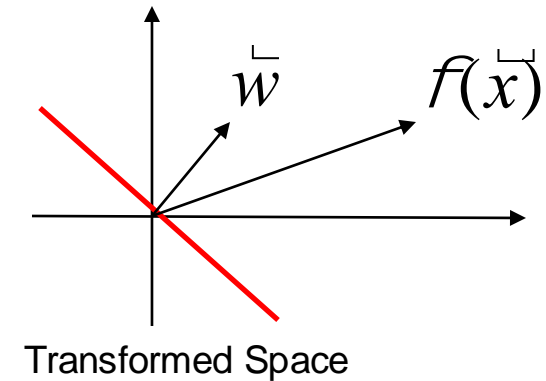
Kernel Perceptron

- **Perceptron algorithm** is a simple algorithm that can be implemented as a kernel method:

- **Classification:** Separation at hyperplane with normal vector \vec{w} in the transformed space

$$h(\vec{z}) = \text{sgn}(\vec{w} \phi(\vec{z}))$$

- Linear separation in the transformed space leads to "complex" separation in origin space
- **Problem:** How is $\vec{w} \phi(\vec{z})$ calculated, given the training examples $\{(\vec{x}_j, y_j) \mid j = 1..n\}$



Kernel Perceptron

- Training a perceptron (in transformed space):

- An update is needed for wrongly classified examples (\vec{x}_j, y_j) :

$$\vec{w} \leftarrow \vec{w} + y_j \phi(\vec{x}_j)$$

- this implicitly calculates the weight/normal vector to:

$$\vec{w} = \sum_{j=1}^n \alpha_j y_j \phi(\vec{x}_j)$$

- where α_j includes the number of updates (\vec{x}_j, y_j)

Kernel Perceptron

- Given a kernel function K , we can derive the following algorithm:
(without explicitly knowing transformation ϕ !)

$$\vec{\alpha} = 0$$

repeat

 randomly choose a learning example (\vec{x}_i, y_i)

 if $\left(\text{sign}(\vec{w} \phi(\vec{x}_i)) = \text{sign}(\sum_{j=1}^n \alpha_j y_j K(\vec{x}_j, \vec{x}_i)) \right) \neq y_i$

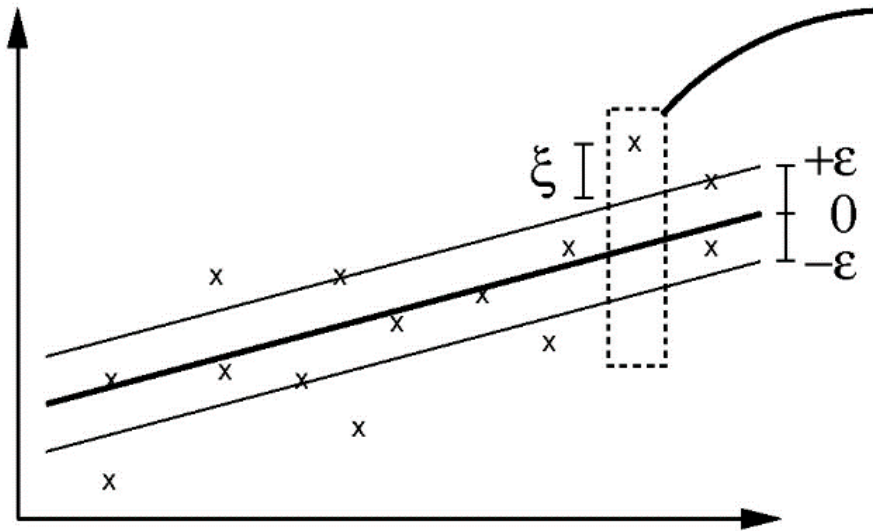
$$\alpha_i = \alpha_i + 1$$

until finished

- Consequently the decision function is $h(\vec{z}) = \sum_{j=1}^n \alpha_j y_j K(\vec{x}_j, \vec{z})$

Other Applications of the SV Method

SV – Regression



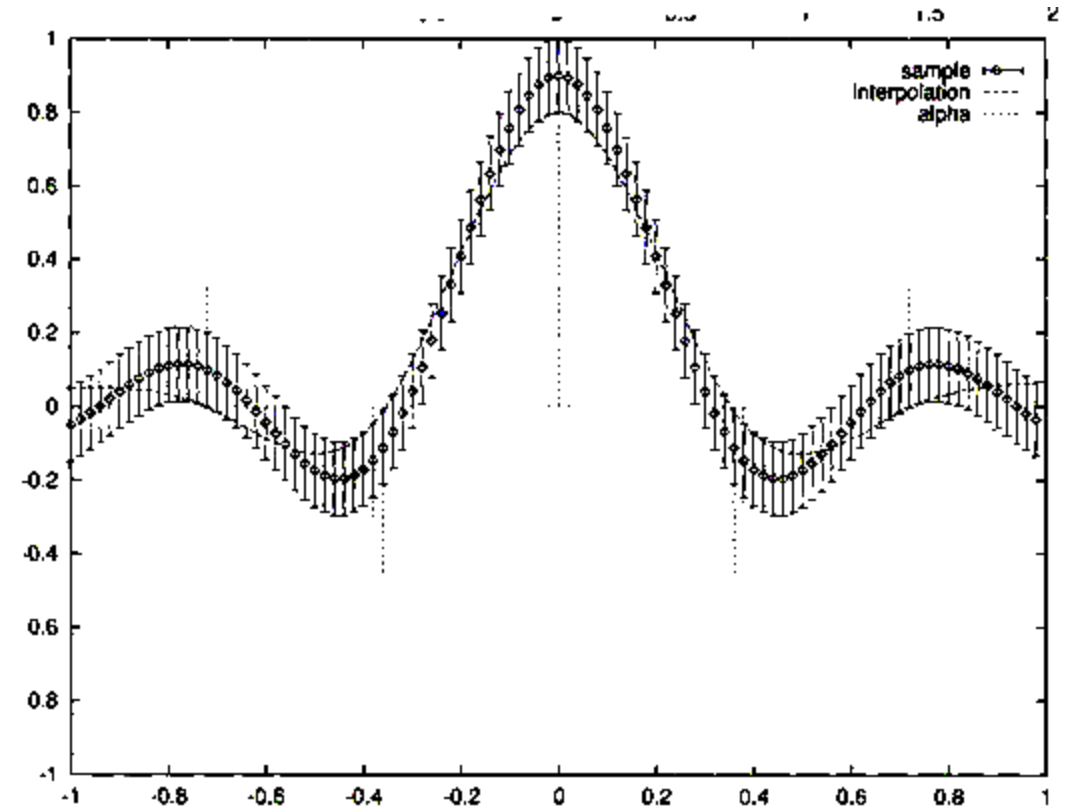
Linear – Regression

+

Constraint = ϵ - environment

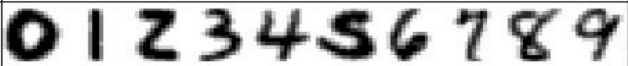
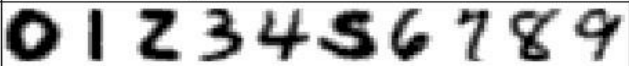










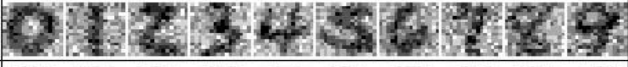










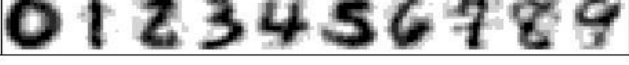
+

Transformation



Other Applications of the SV Method

Nonlinear kernel - PCA

	Gaussian noise	'speckle' noise	
orig.			
noisy			
$n = 1$			linear PCA reconstruction
4			
16			
64			
256			
$n = 1$			kernel PCA reconstruction
4			
16			
64			
256			

Another application: face modelling (Romdhani et al., 1999).

Outline

- Recapitulation & Motivation
- Linear Support Vector Machine
- Nonlinear Support Vector Machine
- Version Space and Structural Risk Minimization
- Extensions
- Applications
- Discussion

SVM – Face Recognition

- Goal: Recognize faces or parts of them in an image

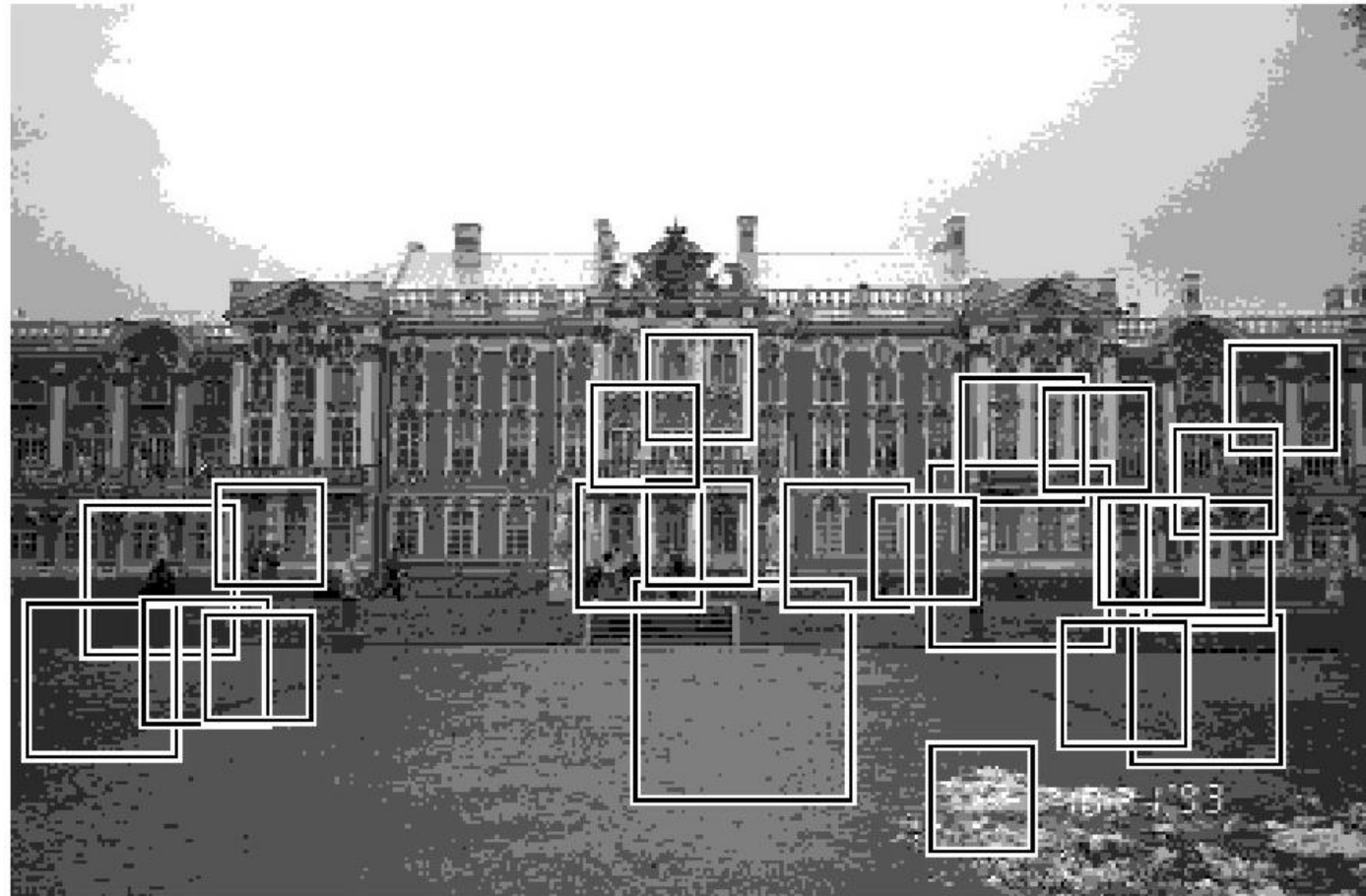
Training data: 19x19 Pixmaps



Noisy training data



SVM – Face Recognition



Training data, no faces

SVM – Face Recognition



Detection of rotated faces

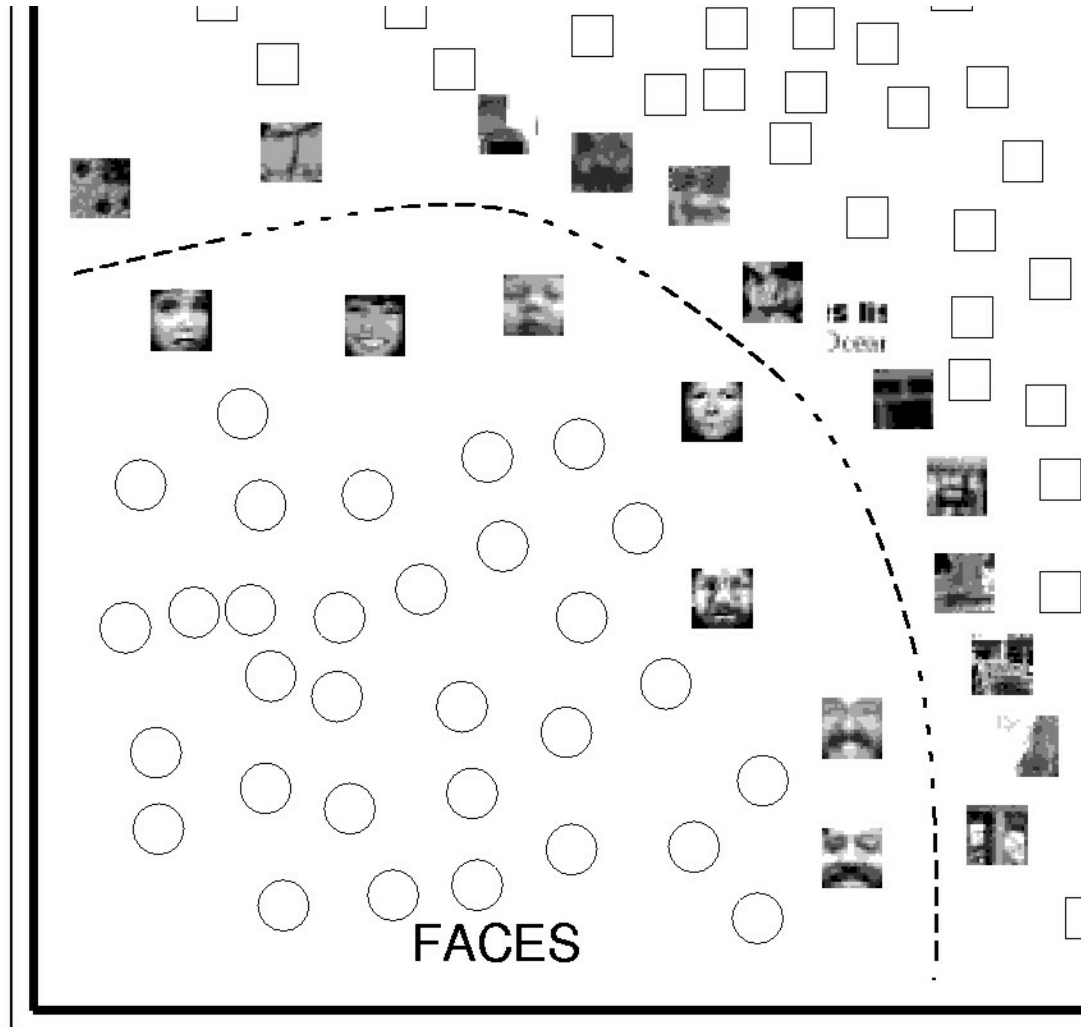
→ Appropriate training data must be generated by augmentation

SVM – Face Recognition



Results

SVM – Face Recognition

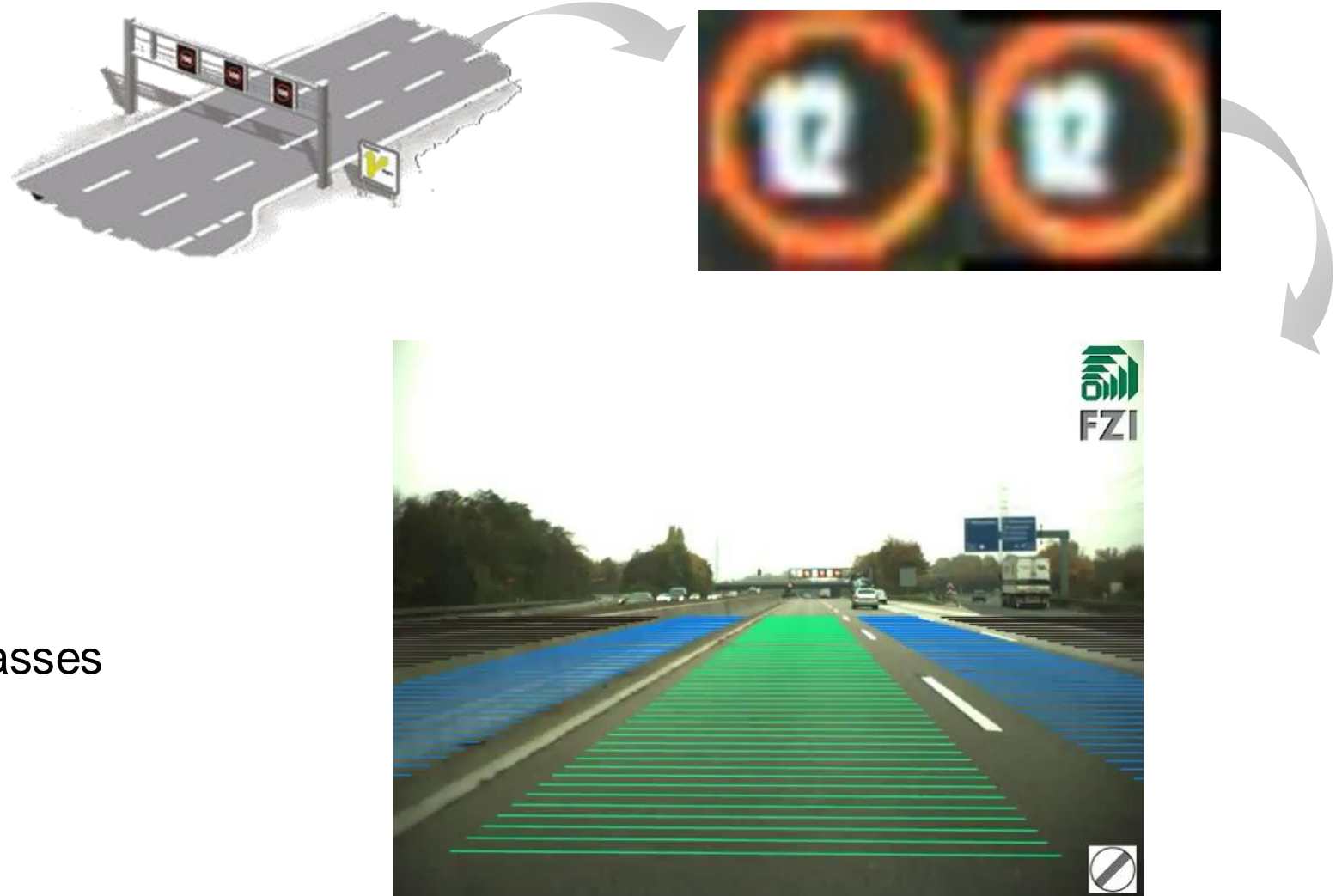


Support Vectors

Plotted according to their
distance from the decision
hyperplane

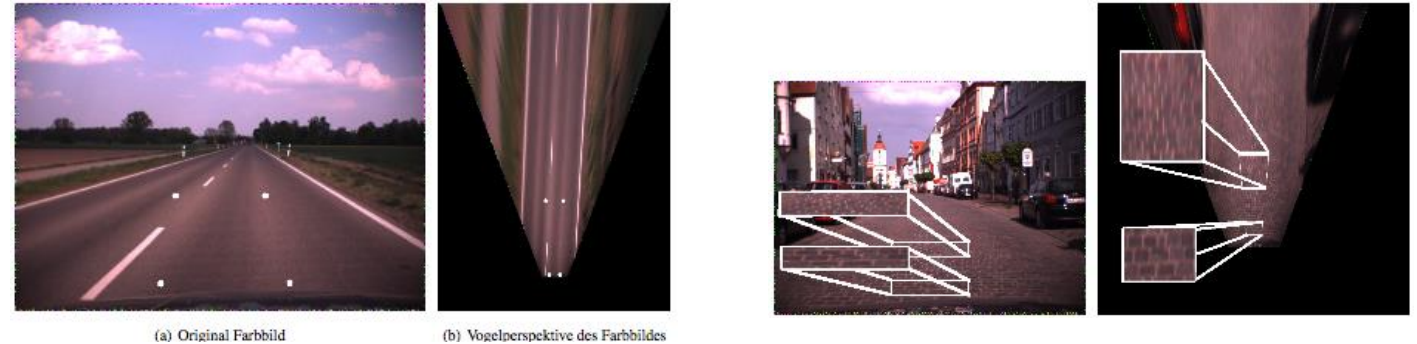
Traffic Sign Recognition

- Preprocessing
 - Color enhancement
- Segmentation
 - Hough-Transformation
 - Artifact removal
- Tracking
 - ...
- Classification
 - Preprocessing
 - SVM with RBF Kernel, 20 classes
- Temporal fusion
- Detection rate $\gg 95\%$



Detection of Traffic Areas

- Preprocessing
 - Transformation
 - Segmentation
- Feature calculation
 - Color features
 - Texture features
- 2-5 Classes
 - Street, Background, Urban, Tar, Nature
- Success rate: 70 – 95%



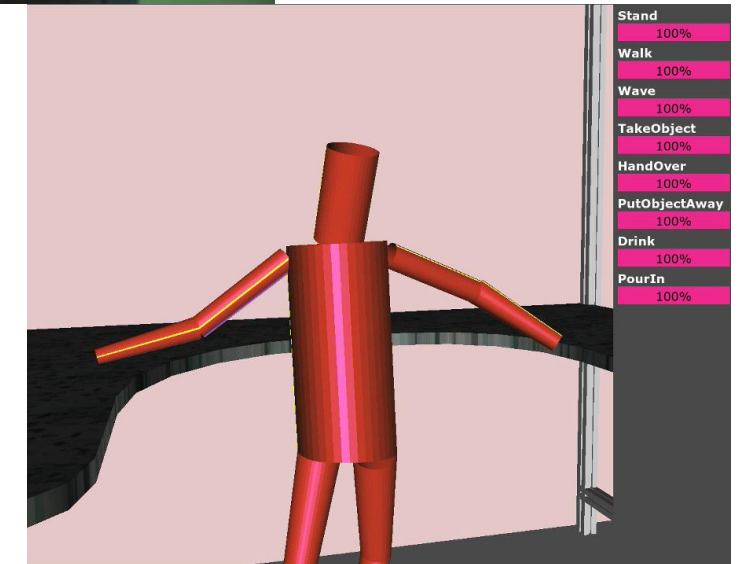
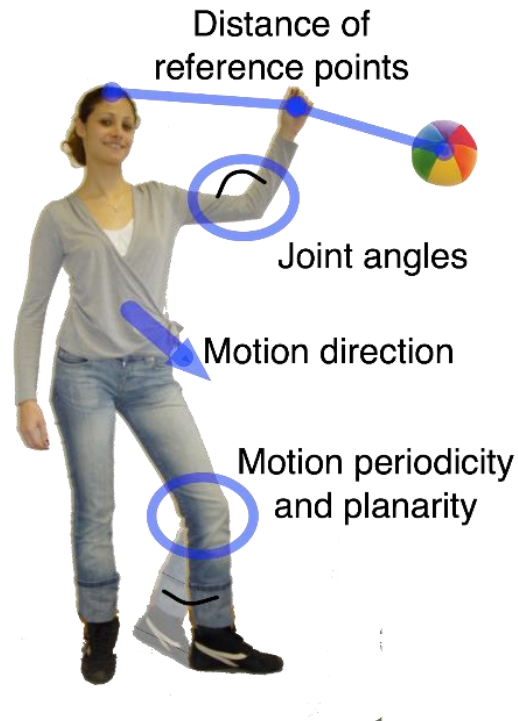
Weather Classification

- Preprocessing
- Feature calculation
 - Intensity / Saturation
 - Contrast, Sharpness
 - ...
- Classification
 - 5 Classes:
Good weather, rain, fog, ...
 - Success rate: ~85%
- Image correction by specific filters



Action Detection

- Modeling
- Tracking
- Feature calculation
 - Time frame
 - Joint angles
 - Velocities
 - ...
- Classification
 - 20 classes
 - Success rate
80 – 90%



Outline

- Recapitulation & Motivation
- Linear Support Vector Machine
- Nonlinear Support Vector Machine
- Version Space and Structural Risk Minimization
- Extensions
- Applications
- Discussion

Pros SVM

■ Pros:

- Optimal hyperplane → Good results
- Finding the optimal VC-dimension → Correct learning
- Processing of high-dimensional data → Rapid evaluation
- Application-specific kernels (with data processing)
- Decision is made on the basis of the margin regions
- Many applications: Classification, Regression, PCA
- Probabilistic view – important for semi-supervised learning → see ML 2
- Description in VS – well suited for active learning → see ML 2

Cons SVM

■ Cons:

- External pre-processing (no "deep" learning)
- Finding an optimal kernel – unsolved/research
- Kernel parameterization – unsolved/research
- Memory and computing power (especially for training with large data sets)
- Number of SVs depends on problem and parameters – but advanced approaches possible

Literature

- Tool - Libsvm: <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- *V.N. Vapnik*: **Statistical Learning Theory**. Wiley, 1998.
- *B. Schölkopf*: **Support Vector Learning**.
- <http://www.kernel-machines.org> (Unfortunately not maintained for a long time)
- And Publications IAR, ATKS, FZI