Technische Hochschule Rosenheim

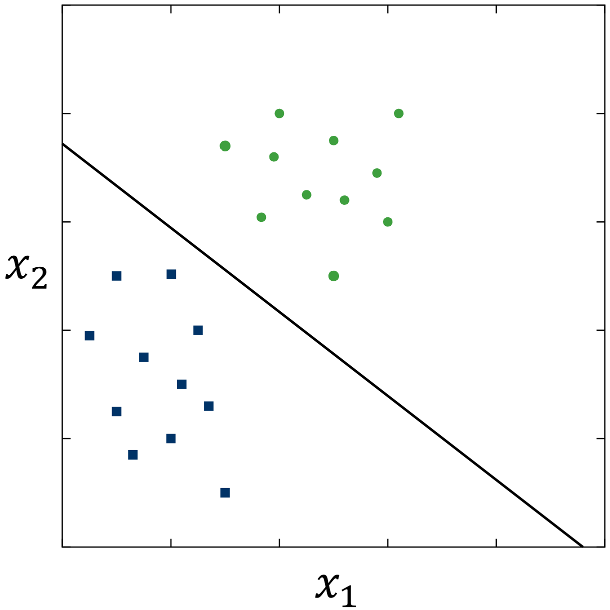
Machine Learning
Support Vector Machines

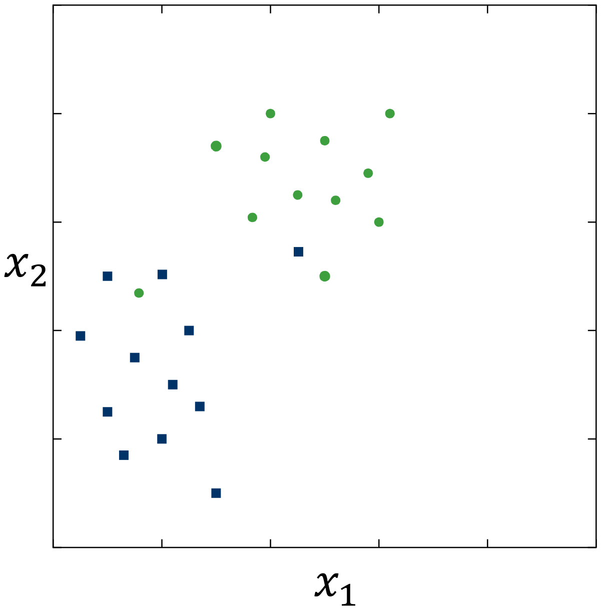
Prof. Dr. Jochen Schmidt

Technische Hochschule Rosenheim

Motivation

- assume two linearly separable classes
- compute linear decision boundary that
  - allows for separation of training data
  - generalizes well

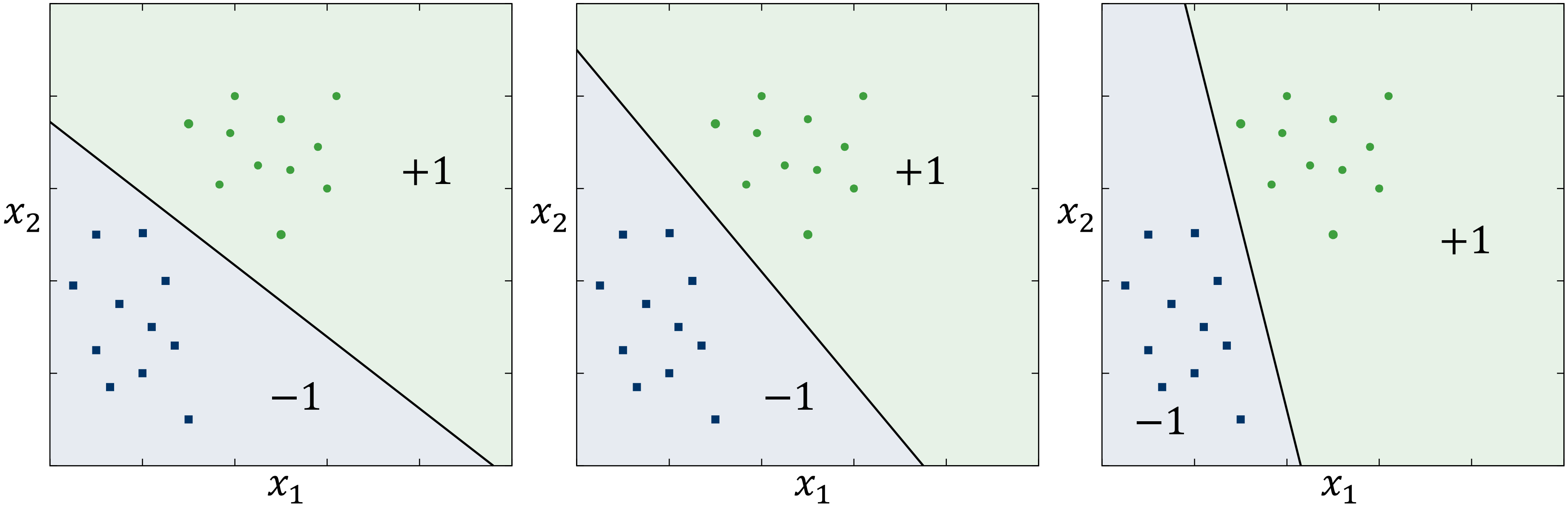




Technische Hochschule Rosenheim

Motivation

Many, many solutions…



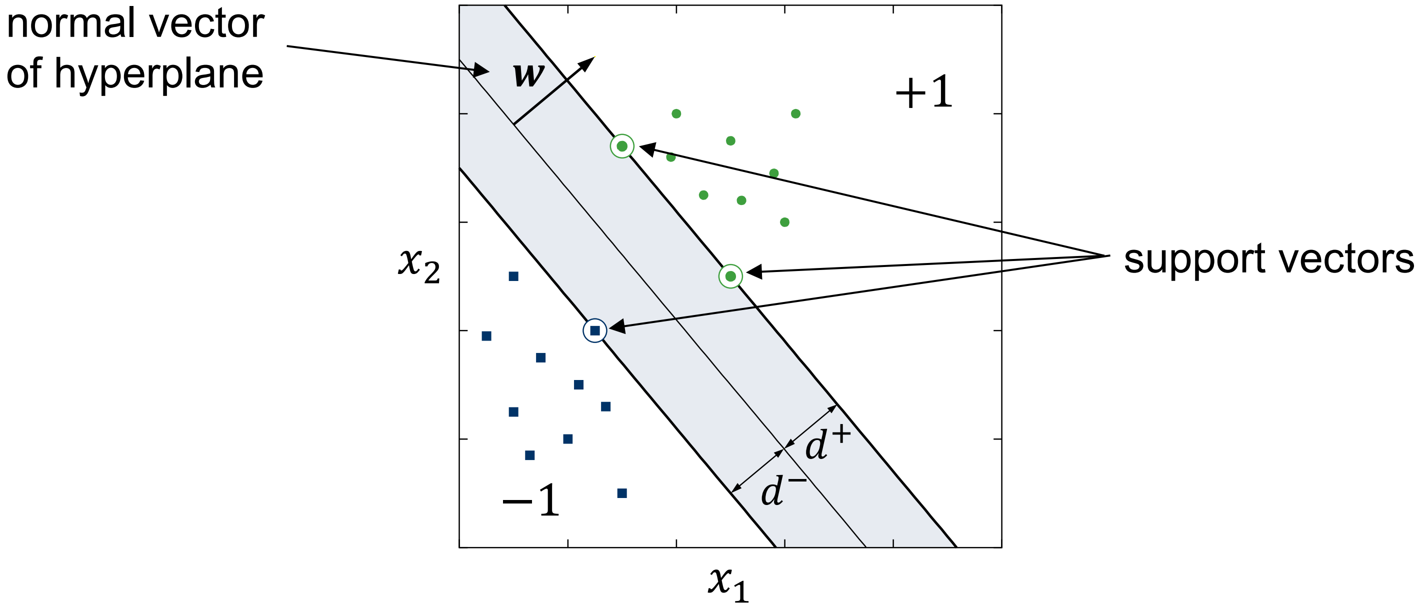
Optimal Separating Hyperplane

Vapnik 1996: Optimal separating hyperplane that

- separates two classes and
- maximizes the distance to the closest point from either class.

This results in

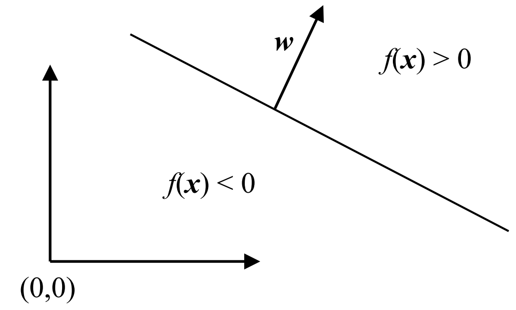
- unique solution for hyperplanes, and
- (in most cases) better generalization.



Functional margin of separating hyperplane: $d^+ + d^-$

Optimal Separating Hyperplane

- Plane equation: $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$
  - normal vector: $\boldsymbol{w}$
  - point on plane: $f(\boldsymbol{x}) = 0$
  - point above plane: $f(\boldsymbol{x}) > 0$
  - point below plane: $f(\boldsymbol{x}) < 0$
  - ”above” = in direction of plane normal

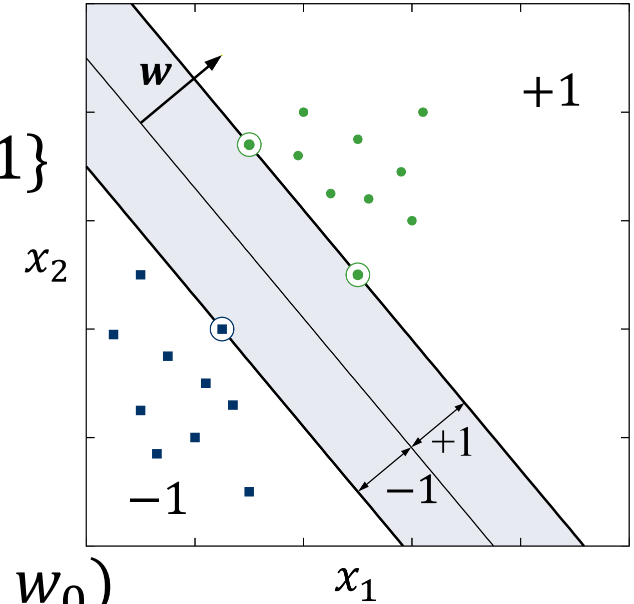


- Signed distance d of a point to hyperplane
  - normalize $\boldsymbol{w}$, such that $|\boldsymbol{w}| = 1$:

$$d = f'(\boldsymbol{x}) = \frac{1}{|\boldsymbol{w}|} f(\boldsymbol{x}) = \frac{1}{|\boldsymbol{w}|}\boldsymbol{w}^T\boldsymbol{x} + \frac{1}{|\boldsymbol{w}|} w_0$$

  - distance of plane from origin: $-\frac{1}{|\boldsymbol{w}|} w_0$

SVM – Classification



- data point: $\boldsymbol{x}_i$
- class of data point $\boldsymbol{x}_i$ is $y_i \in \{-1, +1\}$
- Classifier: $g(\boldsymbol{x}_i) = \operatorname{sgn}(\boldsymbol{w}^T \boldsymbol{x}_i + w_0)$
- Functional margin of $\boldsymbol{x}_i$: $y_i\ (\boldsymbol{w}^T \boldsymbol{x}_i + w_0)$
  - can be increased/decreased by scaling plane equation
  - $\rightarrow$ scale such that support vectors have distance $-1/+1$
  - then $g(\boldsymbol{x}_i)$ is equivalent to: $y_i\ (\boldsymbol{w}^T \boldsymbol{x}_i + w_0) \geq 1$
- Functional margin for data set:
  2 x minimum functional margin of all points: $\frac{2}{|\boldsymbol{w}|}$

SVM – Training

- Training =
  - find hyperplane maximizing the margin $\frac{2}{|\boldsymbol{w}|}$
  - subject to constraint $y_i\,(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) \geq 1$ for all data points

- This is equivalent to
  - minimizing $\frac{1}{2}|\boldsymbol{w}|^2 = \frac{1}{2}\,\boldsymbol{w}^T\boldsymbol{w}$
  - subject to $y_i\,(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1 \geq 0$

- Remarks
  - this is a convex optimization problem
    - local optimum is always a global one – solution is unique
  - there exist efficient algorithms for convex optimization
  - standard libraries can be used

Technische Hochschule Rosenheim

Optimization – Lagrangian

Solving the constrained convex optimization problem requires the Lagrangian, i.e. minimize

Lagrange multipliers

$$L(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_i \lambda_i (y_i\,(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1)$$

Minimization:

- Compute first partial derivatives
- Set derivatives to zero

Here: The problem is reformulated as the so-called Lagrangian Dual, which is then maximized

Technische Hochschule Rosenheim

Optimization – Lagrangian

$$L(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_i \lambda_i(y_i\,(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1)$$

Partial derivative (1):

$$\frac{\partial L(\boldsymbol{w}, w_0, \boldsymbol{\lambda})}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \lambda_i y_i \boldsymbol{x}_i = 0$$

$$\Rightarrow \boldsymbol{w} = \sum_i \lambda_i y_i \boldsymbol{x}_i$$

Partial derivative (2):

$$\frac{\partial L(\boldsymbol{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = -\sum_i \lambda_i y_i = 0$$

the partial derivative for $\boldsymbol{\lambda}$ is not required for the following reformalization

Technische Hochschule Rosenheim

Langrange Dual

$$L_D = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_i \lambda_i(y_i\,(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1)$$

$$= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \underbrace{(\textstyle\sum_i \lambda_i y_i \boldsymbol{x}_i)^T}_{=\boldsymbol{w}^T}\boldsymbol{w} - \underbrace{\textstyle\sum_i \lambda_i y_i w_0}_{=0} + \sum_i \lambda_i$$

partial derivative (1) partial derivative (2)

$$= -\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_i \lambda_i$$

$$= -\frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j + \sum_i \lambda_i$$

This term is now maximized subject to $\lambda_i > 0$

SVM – Optimization Solution

Training: Maximize $L_D = -\frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_i \lambda_i$

Resulting hyperplane:

$$\boldsymbol{w} = \sum_i \lambda_i y_i \boldsymbol{x}_i \qquad w_0 = y_k - \boldsymbol{w}^T \boldsymbol{x}_k \text{ for any } \boldsymbol{x}_k \text{ with non-zero } \lambda_i$$

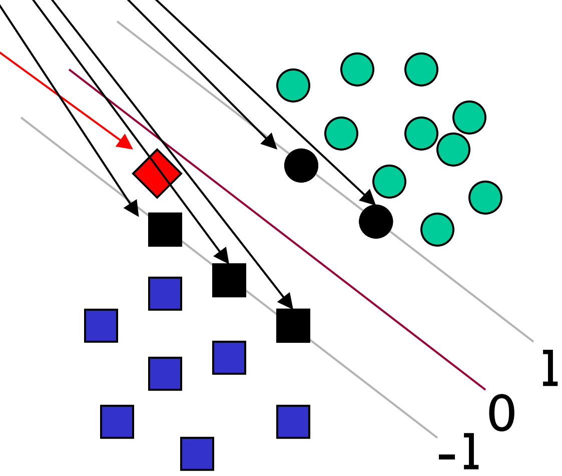
Classification: $g(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{w}^T \boldsymbol{x} + w_0) = \mathrm{sgn}\left(\sum_i \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + w_0\right)$

- the $\boldsymbol{x}_i$ with non-zero λ_i are the support vectors
- the feature vectors $\boldsymbol{x}_i$ only appear in inner products
  - in training as well as classification phase
- $\boldsymbol{w}$ is not required explicitly for classification
  - we need only the support vectors and their Lagrange multipliers

Technische Hochschule Rosenheim

SVM – Classification with threshold

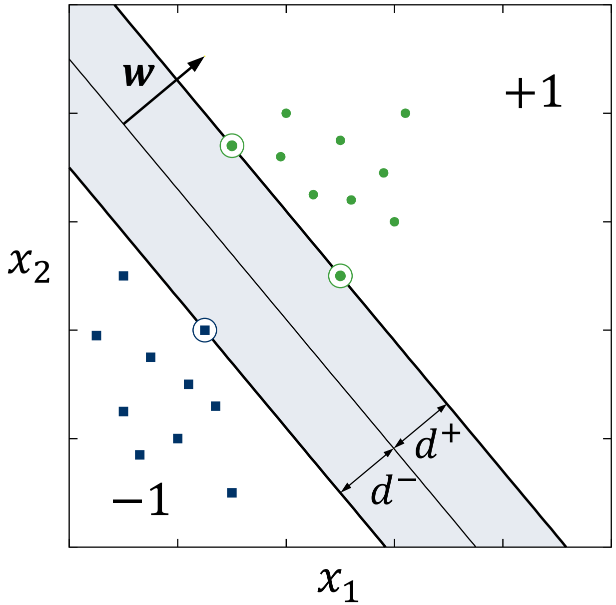
Classification: $g(\boldsymbol{x}) = \sum_i \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + w_0$



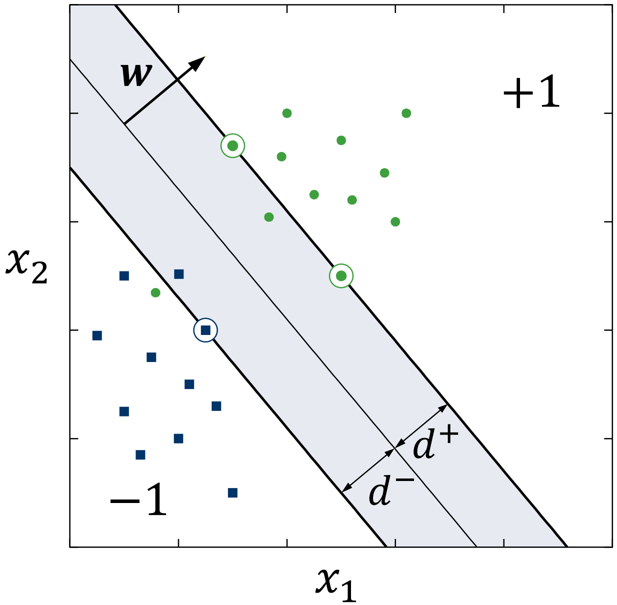
- Classification without threshold
  - decide for class based on $g(\boldsymbol{x}) < 0$ or $g(\boldsymbol{x}) > 0$
- Classification with confidence threshold t
  - $g(\boldsymbol{x}) < -t$: class -1
  - $g(\boldsymbol{x}) > t$: class $+1$
  - $-t < g(\boldsymbol{x}) < t$: reject

Hard and Soft Margin Problem

Hard margin



Soft margin



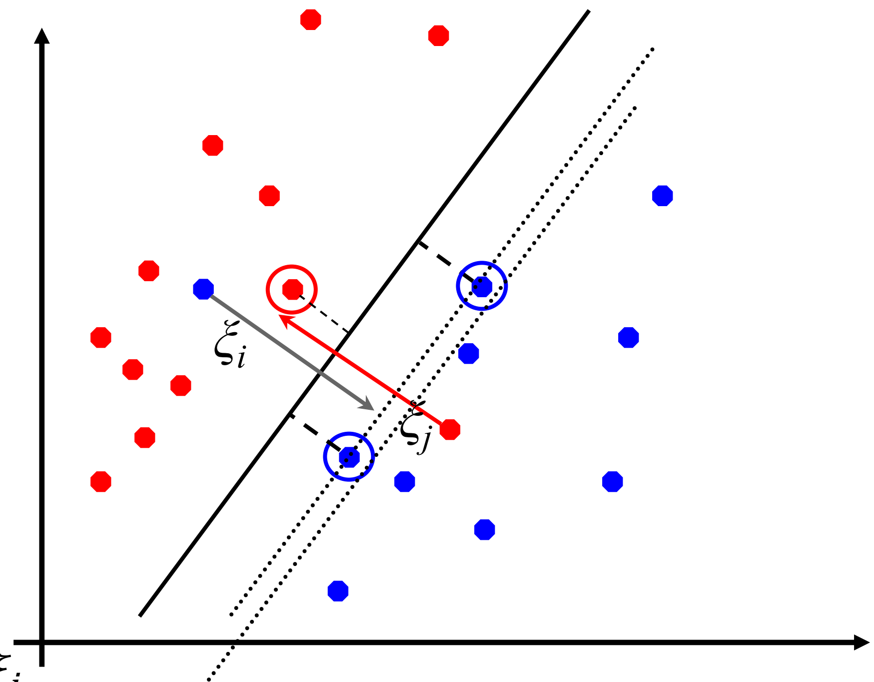
data are not linearly separable in this case

Technische Hochschule Rosenheim

Soft Margin – Slack Variables

for not linearly separable data: allow some errors

- allow missclassification of difficult or noisy samples
- move data slightly, to where they belong
- pay a penalty



Training

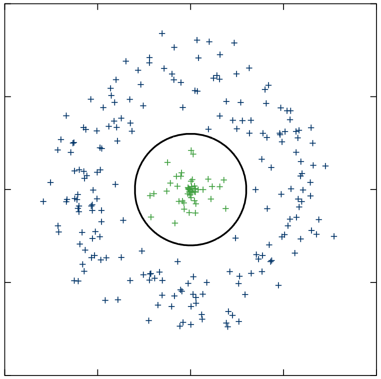
- minimize $\frac{1}{2}|\boldsymbol{w}|^2 + c\sum \xi_i$
- subject to
  $y_i\,(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1 + \xi_i \geq 0$
  $\xi_i \geq 0$
- reformulate as dual problem
  - neither the slack variables ξ_i
  - nor their Lagrange multipliers c appear there

Technische Hochschule Rosenheim

Kernels / Non-linear Boundaries

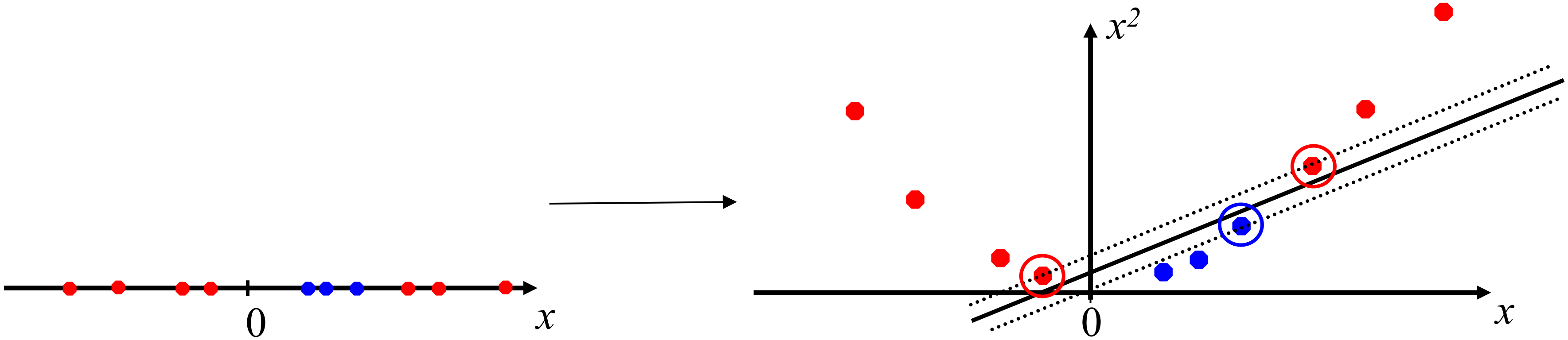
Limitations of linear decision boundaries

- too simple for most practical purposes
- non-linearly separable data cannot be classified
- noisy data cause problems



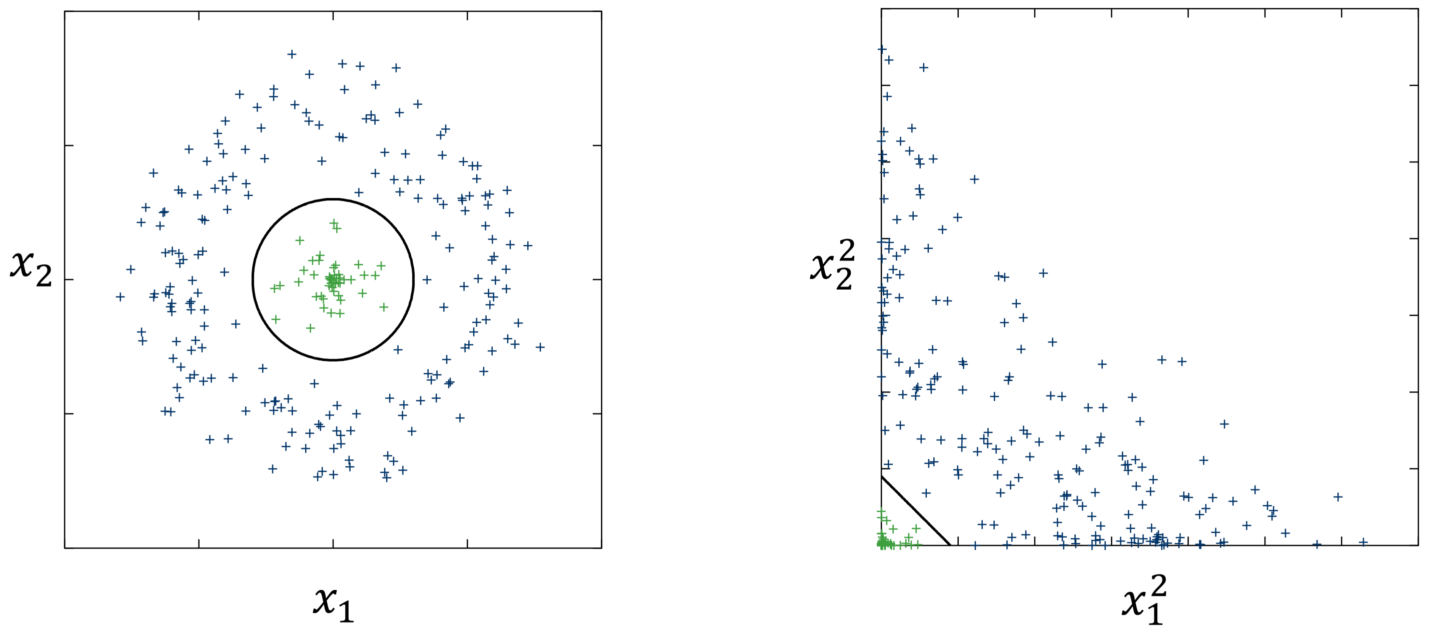
Possible solution

- Map data to higher dimensional feature space using non-linear feature transform,
- then use a linear classifier



Feature Transforms

Select a feature transform ϕ: $\mathbb{R}^d \rightarrow \mathbb{R}^D$ such that the resulting features $\phi(\boldsymbol{x}_i)$ are linearly separable.



Applied feature transform in example: $\phi(\boldsymbol{x}_i) = (x_1^2, x_2^2)^T$

Technische Hochschule **Rosenheim**

Feature Transforms – Example

Assume the decision boundary is defined by the quadratic function

$$f(\boldsymbol{x}) = a_0 + a_1 x_1^2 + a_2 x_2^2 + a_3 x_1 x_2 + a_4 x_1 + a_5 x_2$$

This is obviously non-linear.

Using the following mapping, we get features having a linear decision boundary:

$$\phi(\boldsymbol{x}) = \begin{pmatrix} 1 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \\ x_1 \\ x_2 \end{pmatrix}$$

Feature Transforms – SVM

The feature transforms can be easily incorporated into SVMs:
Replace $\boldsymbol{x}_i^T\boldsymbol{x}$ by $\phi^T(\boldsymbol{x}_i)\phi(\boldsymbol{x}) = \langle\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x})\rangle$ ← $\langle\cdot\rangle$ notation for inner product

Classification/Decision boundary:

$$g(\boldsymbol{x}) = \sum_i \lambda_i y_i \phi^T(\boldsymbol{x}_i)\phi(\boldsymbol{x}) + w_0 = \sum_i \lambda_i y_i \langle\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x})\rangle + w_0$$

Training: Lagrange dual problem

Maximize $$L_D = -\frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j y_i y_j \langle\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle + \sum_i \lambda_i$$

subject to $\lambda_i > 0$

Note:
- the actual transform $\phi(\boldsymbol{x})$ is never required stand-alone
- it only appears as inner products → Kernel Trick

Technische Hochschule Rosenheim

Kernel-Trick

- in SVM training/classification, data appear only in the form of inner products $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$

- a Kernel-function is a function computing this inner product directly: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$
  - i.e., without first transforming the features using $\phi(\boldsymbol{x})$
  - it can be computed in the original low-dimensional space!

- checking whether $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ holds is often cumbersome
  - Mercer's theorem:
    Any positive semi-definite symmetric function is a kernel

Common Kernel Functions

- linear: $$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$
- polynomial: $$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1\right)^d$$
- Laplacian radial basis function (RBF):
$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{\left\|x_i - x_j\right\|_1}{\sigma^2}}$$
- Gaussian radial basis function (RBF):
$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{\left\|x_i - x_j\right\|_2^2}{\sigma^2}}$$
- sigmoid: $$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh\left(\alpha \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \beta\right)$$

Technische Hochschule Rosenheim

Notes

- SVMs are perceptrons
  - training is different
  - compare sigmoid kernel to perceptron computation
  - Multilayer Perceptrons
    - are SVMs maximizing the margin in hidden layer space
    - all hidden units are SVMs

- Multiclass-SVM
  - split into multiple binary classifications
  - one-vs-all
    - one binary SVM per class, separating this class from all others
    - winner-takes all strategy (winner = class with highest value)
  - one-vs-one
    - train binary SVMs for each pair of classes
    - each SVM votes: max-wins strategy

- SVMs in scikit-learn: https://scikit-learn.org/stable/modules/svm.html

Technische Hochschule Rosenheim

References

slides based on

- slides of the lecture *Pattern Recognition* taught at the FAU Erlangen-Nuremberg, courtesy of D. Hahn, J. Hornegger, S. Steidl and E. Nöth.
- Ray Mooney: Support Vector Machines. Slides, University of Texas at Austin.
- Ch. Manning, P. Nayak: Introduction to Information Retrieval, Lecture 14: Support vector machines and machine learning documents. Stanford University.