Lecture 10: Kernel Methods (Part I)

Top-Down Introduction to the Kernel Trick and the Support Vector Machine

Machine Learning, Summer Term 2019

Michael Tangermann Frank Hutter Marius Lindauer

University of Freiburg

Lecture Overview

Lecture Overview

(use the stick)

Lecture Overview

Kernel Algorithms

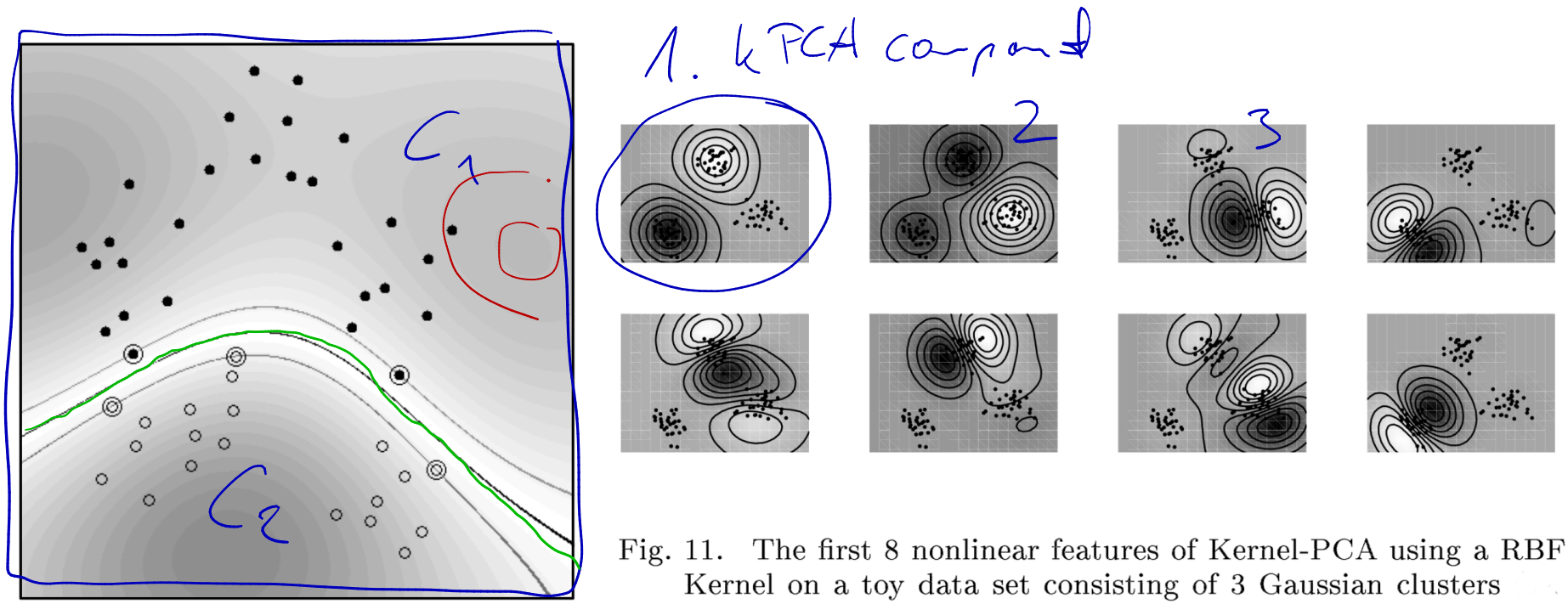


Fig. 11. The first 8 nonlinear features of Kernel-PCA using a RBF Kernel on a toy data set consisting of 3 Gaussian clusters

Kernel algorithms open up new possibilities for non-linear algorithms.
Examples:

- support vector machine (SVM)
- kernel principal component analysis (kPCA)
- Gaussian processes

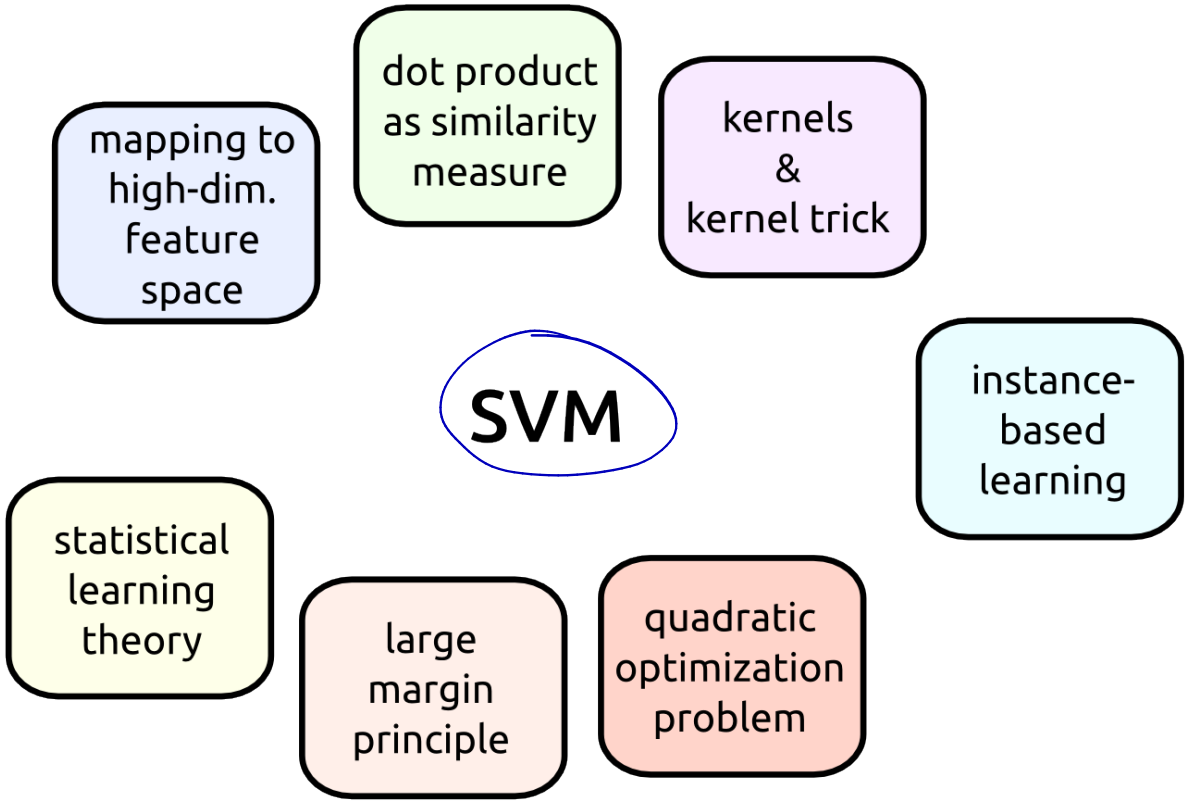
Kernel Algorithms

Kernel methods have rocked the world of machine learning for many application areas, e.g.

- Optical character recognition (OCR), handwritten character recognition
- Image detection
- Bioinformatics (DNA data)
- Biomedical imaging data
- ...

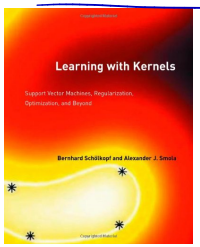
Concepts Underlying Kernel Methods

Today's lecture introduces SVM — however, a number of concepts need to be introduced, in order to get there...



Literature / Materials

- Book by Bernhard Schölkopf and Alexander Smola: Learning with Kernels (MIT Press)

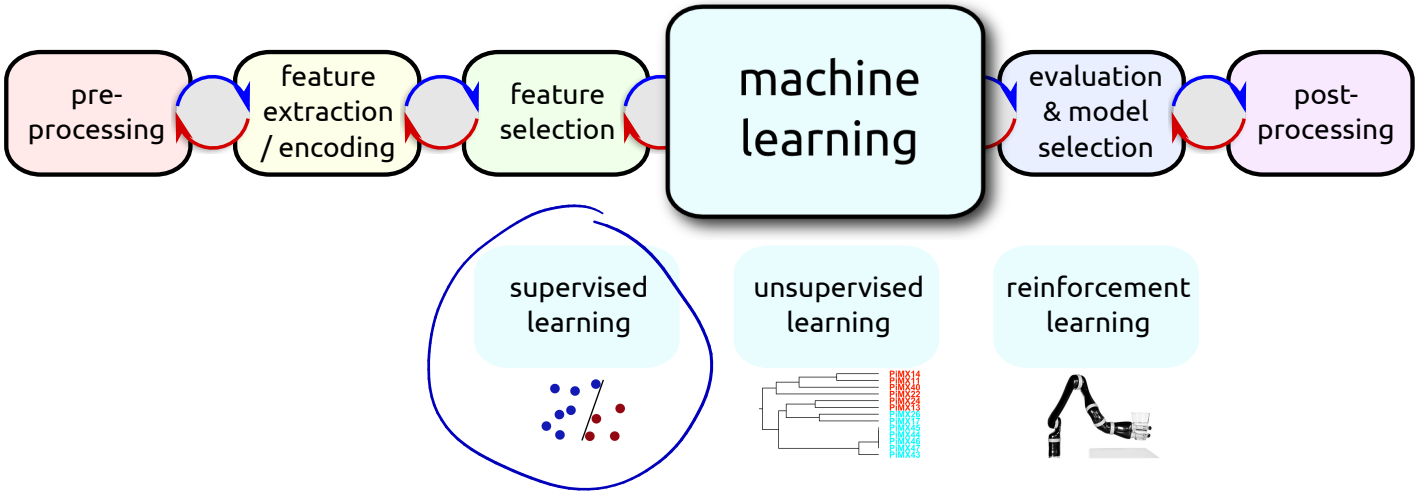


(mostly Sec. 1)

- Tutorial by Hearst et al. "Support vector machines", IEEE Intelligent Systems, 1989.
- Slides by Bernhard Schölkopf: "Introduction to Machine Learning", held at Machine Learning Summer School 2011 in Bordeaux
- Slides by Statnikov et al. "A Gentle Introduction to Support Vector Machines in Biomedicine", held at AMIA 2009.
- Müller et al.: "An Introduction to Kernel-Based Learning Algorithms", IEEE Trans. Neural Networks 2001
- `www.kernel-machines.org`

Lecture Overview

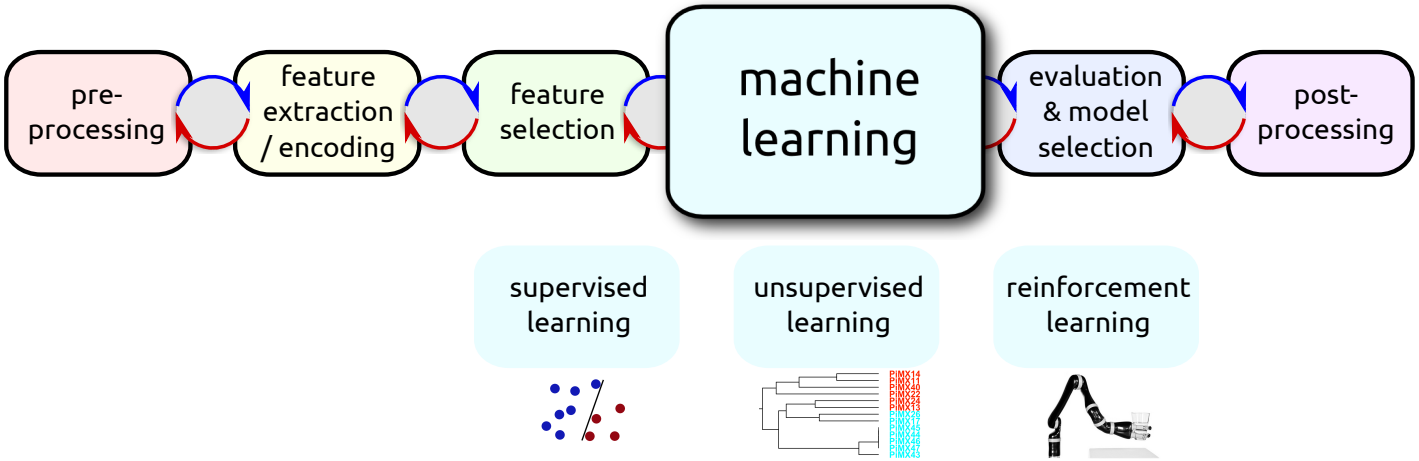
Support Vector Machine within the ML Design Cycle



Context of supervised classification:

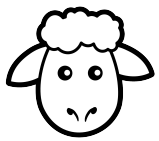
- Use past experience to predict the future
- Typical training data are vectors:
  m-many N-dimensional patterns $\mathbf{x}_i$ and class labels y_i:
  $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in \mathbb{R}^N \times \{\pm 1\}$

Support Vector Machine within the ML Design Cycle



Context of supervised classification:

- Use past experience to predict the future
- Typical training data are vectors:
  m-many N-dimensional patterns $\mathbf{x}_i$ and class labels y_i:
  $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in \mathbb{R}^N \times \{\pm 1\}$
- However, patterns could be from an arbitrary set $\mathcal{X}$:
  $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \{\pm 1\}$



Need for Similarity Measure

Learning task in vector spaces:

- estimate a function (model) f: $\mathbb{R}^N \to \{\pm 1\}$, such that f will correctly classify new examples $(\mathbf{x}, y)$, i.e. $f(\mathbf{x}) = y$

Need for Similarity Measure

Learning task in vector spaces:

- estimate a function (model) $f : \mathbb{R}^N \to \{\pm 1\}$, such that f will correctly classify new examples $(\mathbf{x}, y)$, i.e. $f(\mathbf{x}) = y$

If patterns x_i are from an arbitrary set $\mathcal{X}$, then we need some similarity measure in this space!

Need for Similarity Measure

Learning task in vector spaces:

- estimate a function (model) $f : \mathbb{R}^N \to \{\pm 1\}$, such that f will correctly classify new examples $(\mathbf{x}, y)$, i.e. $f(\mathbf{x}) = y$

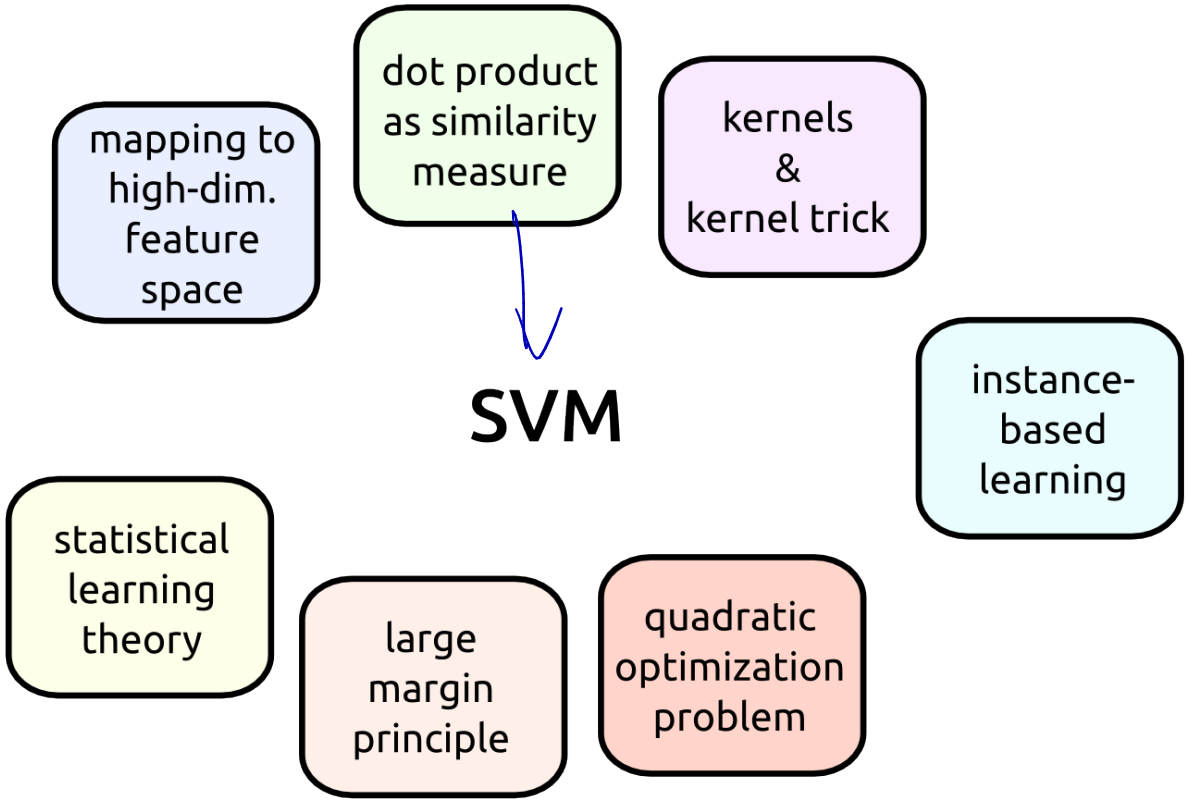
If patterns x_i are from an arbitrary set $\mathcal{X}$, then we need some similarity measure in this space!

Convenient similarity measures should:

- ... compare two patterns x, x'
- ... deliver a real number describing their similarity:
  $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
  $(x, x') \to k(x, x')$
- ... be symmetric, i.e. $k(x, x') = k(x', x)$ for all $x, x' \in \mathcal{X}$
- This similarity function k is called a kernel.

Lecture Overview

Concept: Similarity Measure



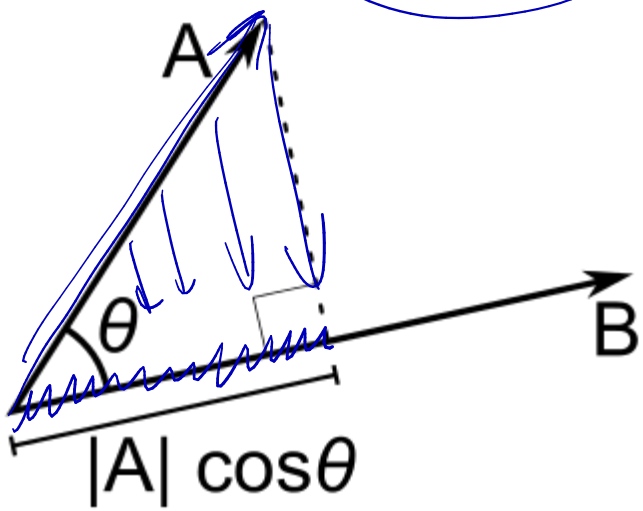
Dot Product as a Convenient Similarity Measure

- Synonyms of dot product: inner product, scalar product.
- Canonical dot product is calculated between two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$
- Definition of dot product: $\langle \mathbf{x}, \mathbf{x}' \rangle := \sum_{i=1}^{N} [\mathbf{x}]_i [\mathbf{x}']_i$
  with $[\mathbf{x}]_i$ being the ith entry / dimension of the vector $\mathbf{x}$.
- Remark: sometimes written as $(x \cdot x')$

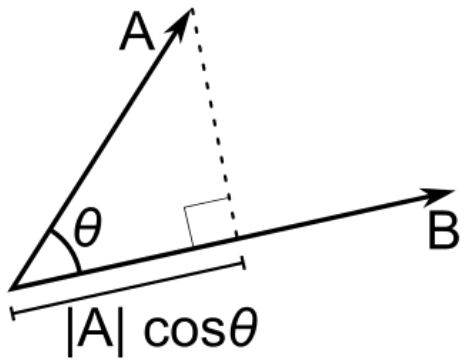
$$x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad x' = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$$

$$\langle x, x' \rangle = 1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6$$

Geometrical interpretation: $\langle \mathbf{A}, \mathbf{B} \rangle = \|A\| \, \|B\| \, \cos(\phi)$

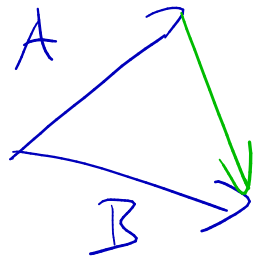


Dot Product as a Convenient Similarity Measure



Dot products are very handy to describe **similarity** in terms of

- angle
- projected length
- length / norm of vector $\mathbf{x}$ as: $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$
- distance between two points (the length of the difference vector)



Thus any geometric construction can be carried out, which can be formulated in terms of angles, lengths and distances!

How Far Do We Get with Angles, Lengths, Distances?

Can we define a simple classification algorithm just based on dot products?

Please vote: **yes** / **no**

How Far Do We Get with Angles, Lengths, Distances?

Can we define a simple classification algorithm just based on dot products?

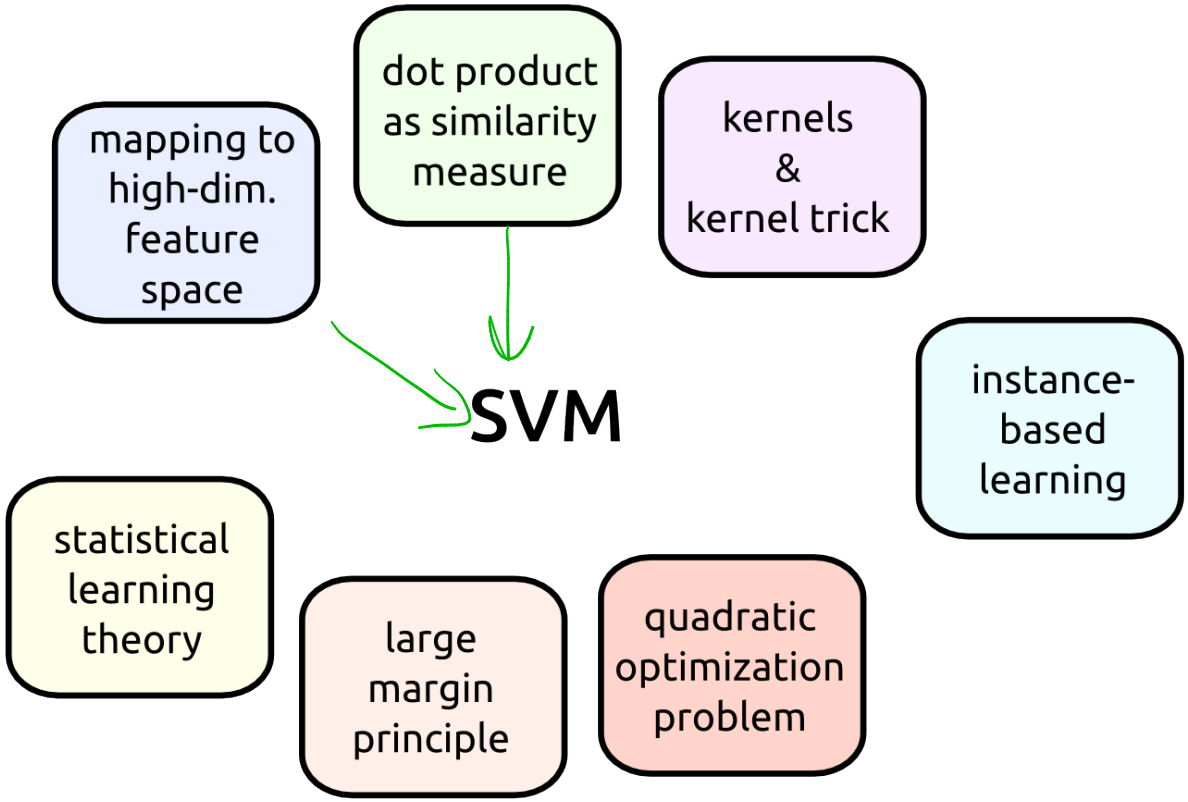
Please vote: **yes** / **no**

(For an example read "Learning with Kernels", Section 1.2)

- Decision function and the offset b can be expressed based on dot products of the training data points
- All computations necessary to obtain the class label for a novel test data point $\mathbf{x}$ involve only dot product operations between $\mathbf{x}$ and some training data points $\mathbf{x}_i$.

Lecture Overview

Concept: Mapping to High-Dimensional Feature Space



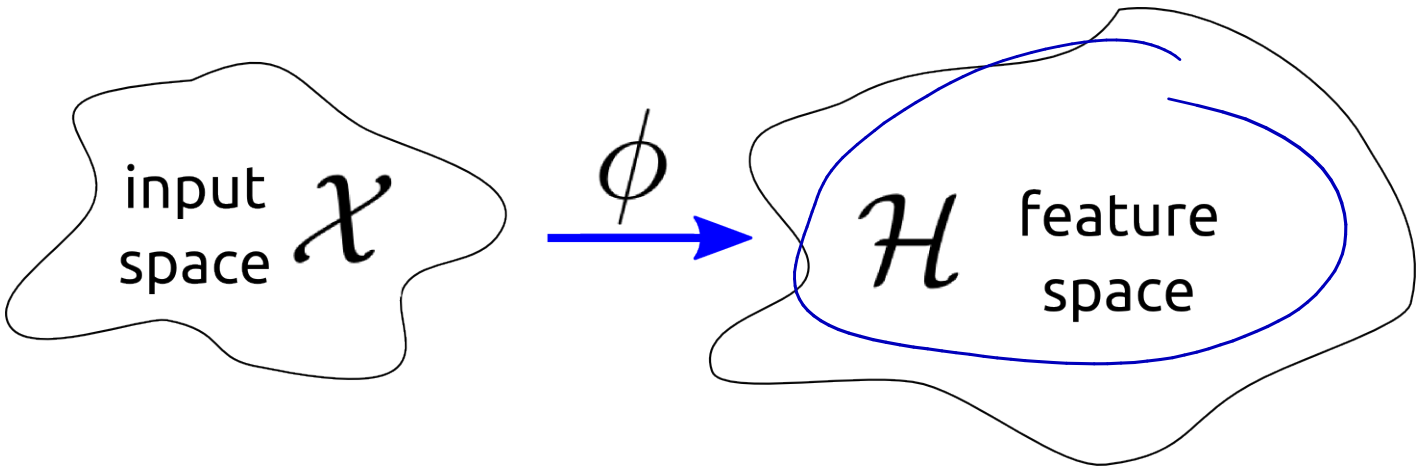
(Non-Linear) Mapping to Feature Space

Please observe:

- The dot product as a similarity measure is OK if patterns $\mathbf{x}_i$ are vectors.
- In general, data points x_i are from the set $\mathcal{X}$ and may need to be mapped to some dot product space first!

Use mapping: dot product space

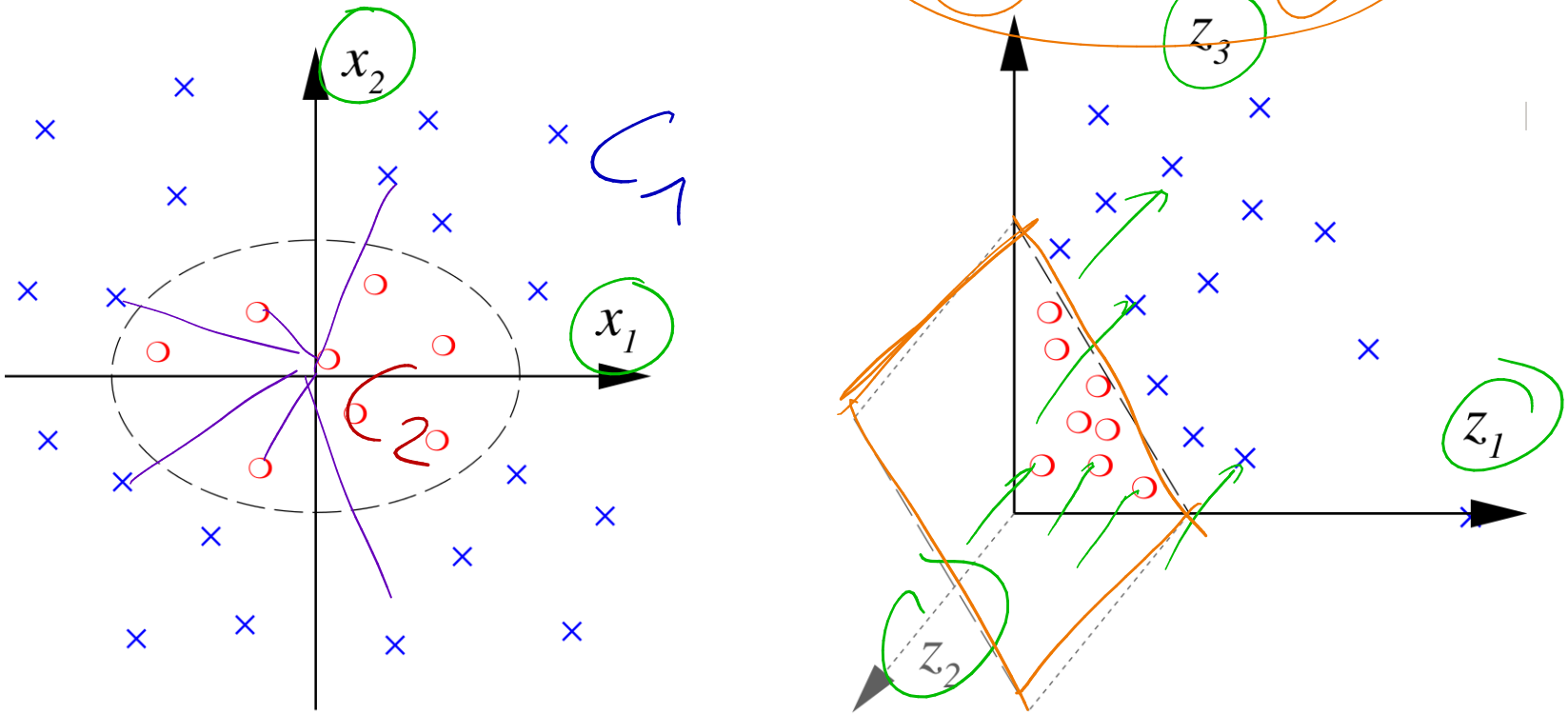
$$\phi : \mathcal{X} \to \mathcal{H}$$
$$x \to \mathbf{x} := \phi(x)$$



(Non-Linear) Mapping to Feature Space

Who remembers this mapping?

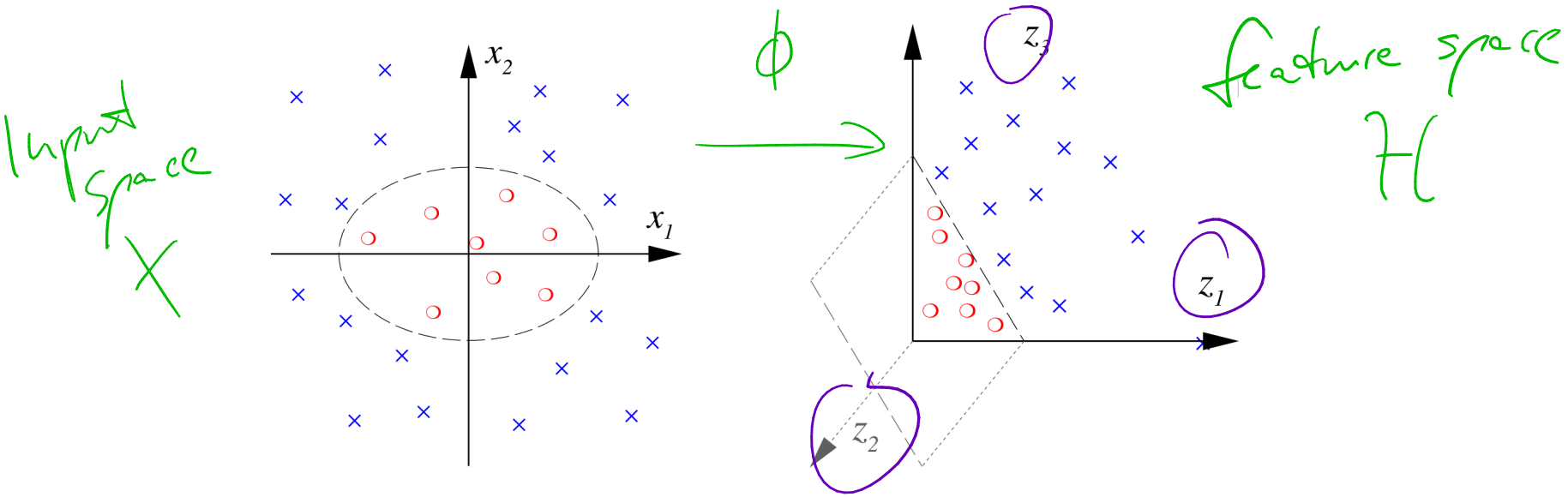
$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$



(Non-Linear) Mapping to Feature Space

Even if the input space is a dot product space already (patterns $\mathbf{x}_i$ are vectors), we may still want to apply a (non-linear) mapping into a feature space.

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

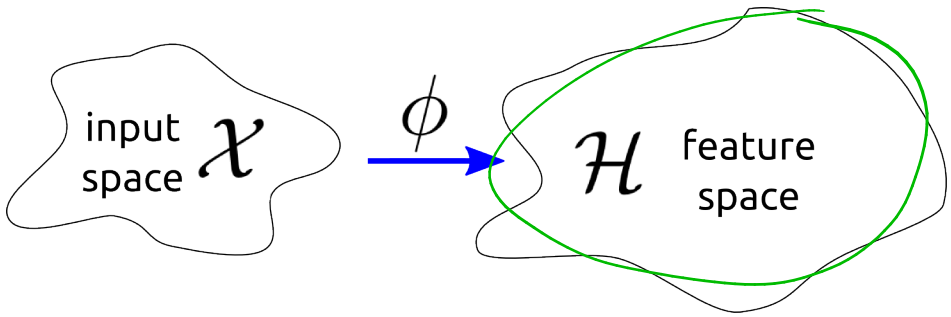


- Feature space $\mathcal{H}$ may have higher dimensionality
- Separability of data points may be easier in $\mathcal{H}$ compared to $\mathcal{X}$

(Non-Linear) Mapping to Feature Space

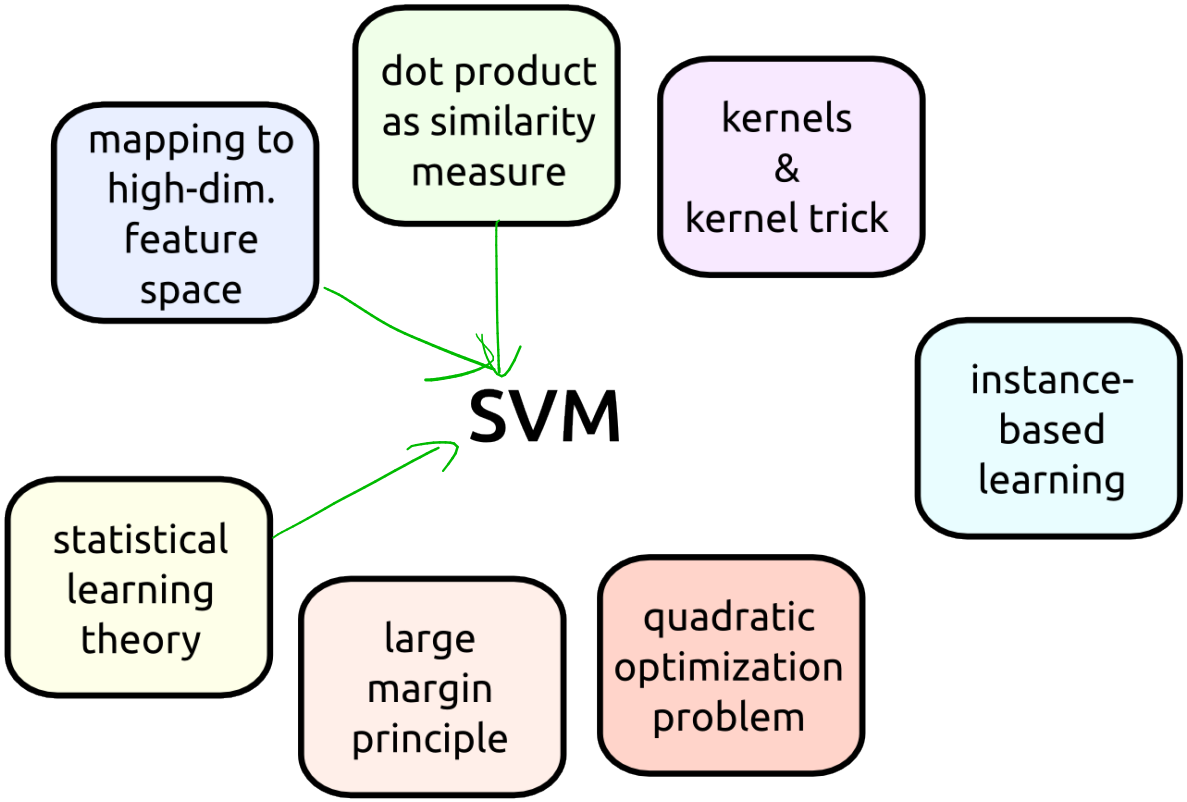
Advantages of applying the mapping:

- Freedom to choose the mapping ϕ opens a wide choice of similarity measures and learning algorithms!
- A suitable mapping ϕ will improve the representation into one that is more suitable to solve the given problem.
- ϕ can define a similarity measure using the dot product in $\mathcal{H}$, $k(x, x') := \langle \mathbf{x}, \mathbf{x}' \rangle = \langle \phi(x), \phi(x') \rangle$
- It allows us to deal with patterns geometrically (we can apply linear algebra + analytic geometry)



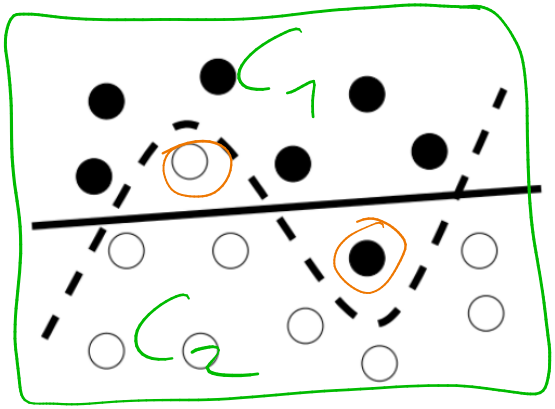
Lecture Overview

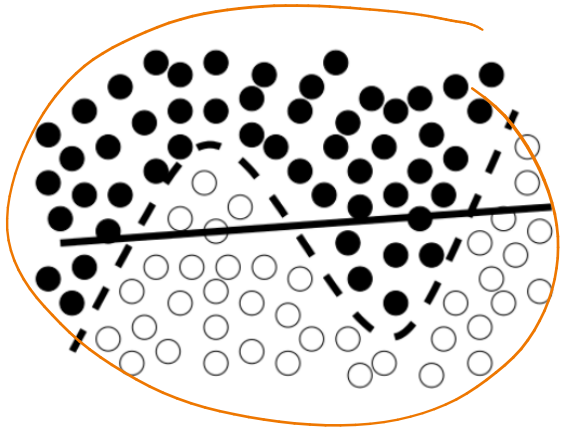
Concept: Statistical Learning Theory

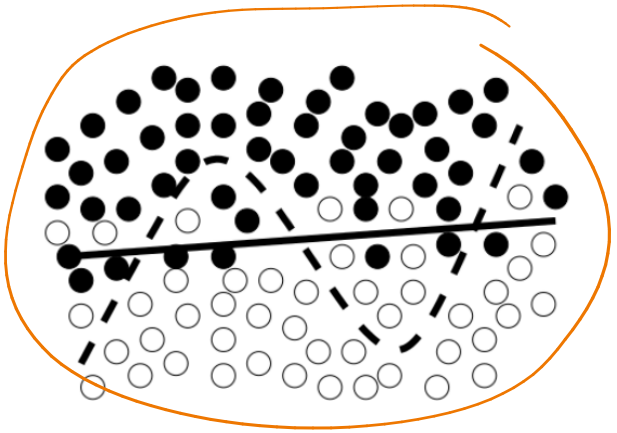


Overfitting Dilemma

Which function (dashed or solid) fits the data on the left better?

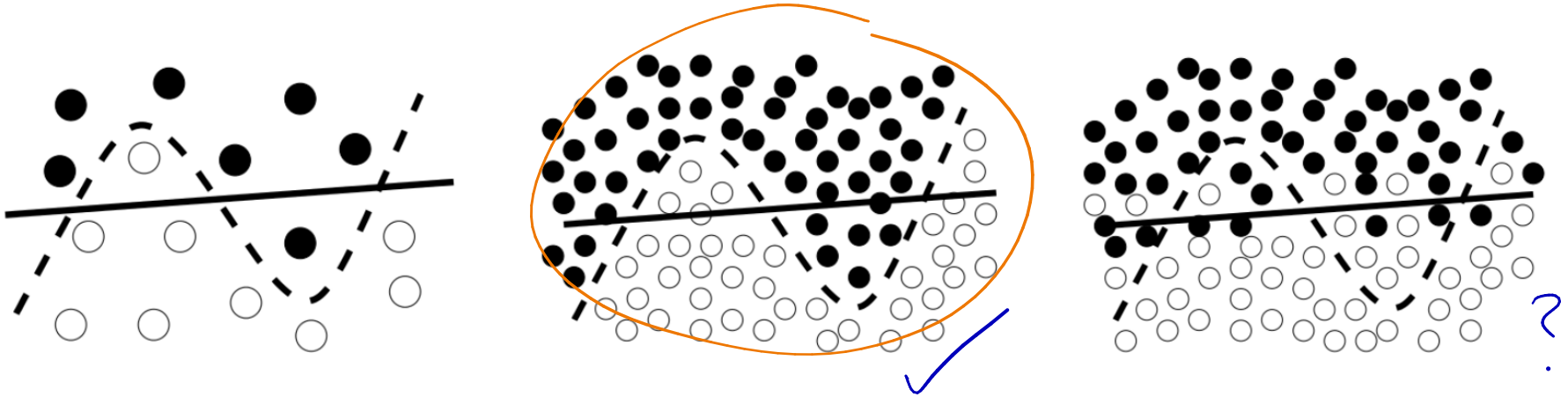






Overfitting Dilemma

Which function (dashed or solid) fits the data on the left better?

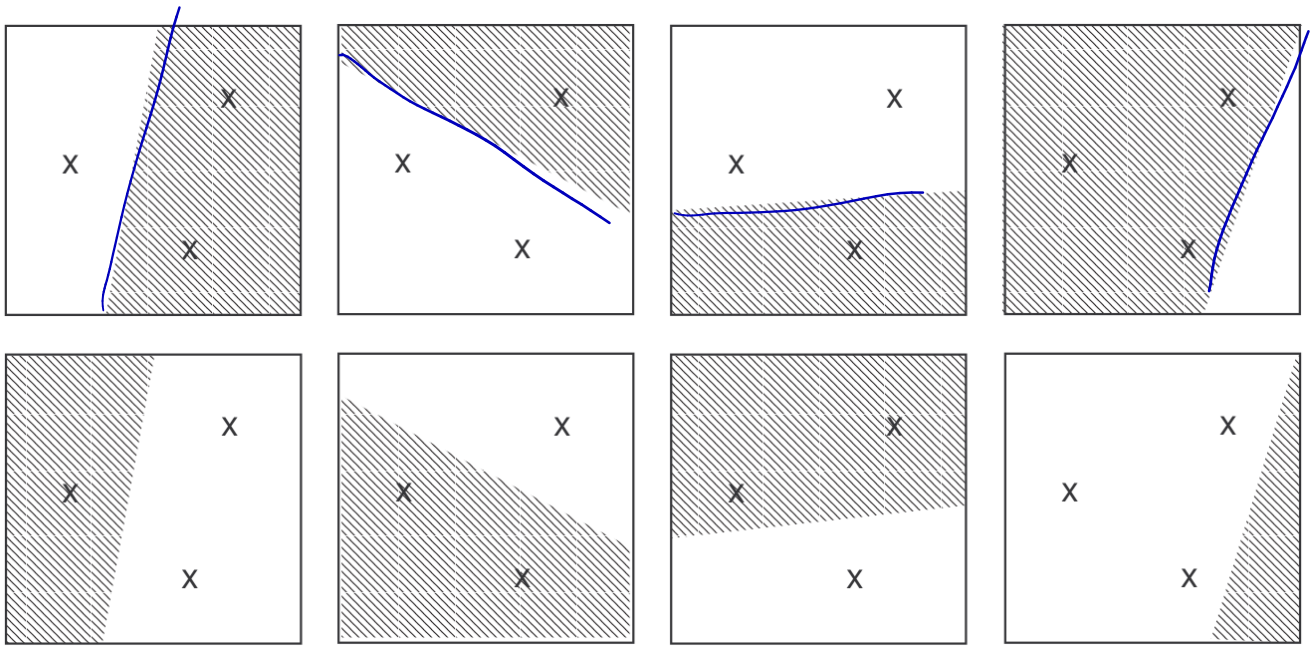


Please observe:

- In supervised learning, the choice of the **function class** / **hypothesis class** is relevant for good generalization of the trained model to novel data.
- Very powerful function classes (high "capacity") tend to overfit the training data.

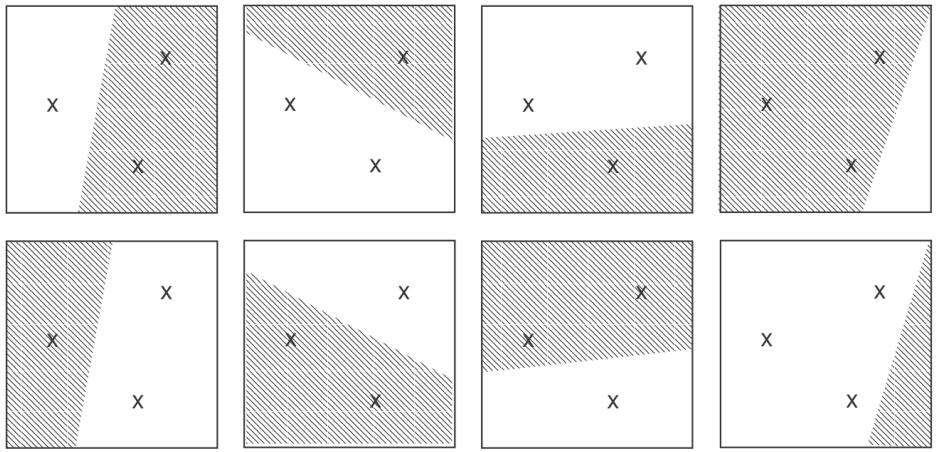
Estimating the Capacity of a Function Class

Scattering



$x \in \mathbb{R}^2$

Estimating the Capacity of a Function Class



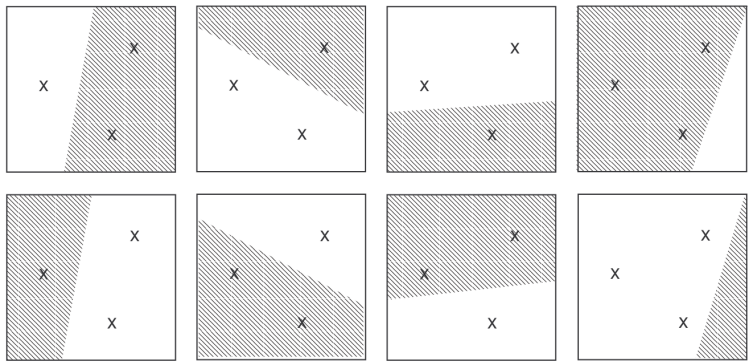
Observe:

- Every function of the class separates patterns in a certain way, thus creating a labelling.
- With labels in $\{\pm 1\}$, there are at most 2^m different labellings for m patterns.
- A rich (high capacity) function class may be able to realize all 2^m separations, thus completely shatter the m points.
- A poorer function class may not be able to shatter all m points.

Definition VC-Dimension

Vapnik & Chervonenkis proposed **VC-dimension** h to estimate the capacity of a function class:

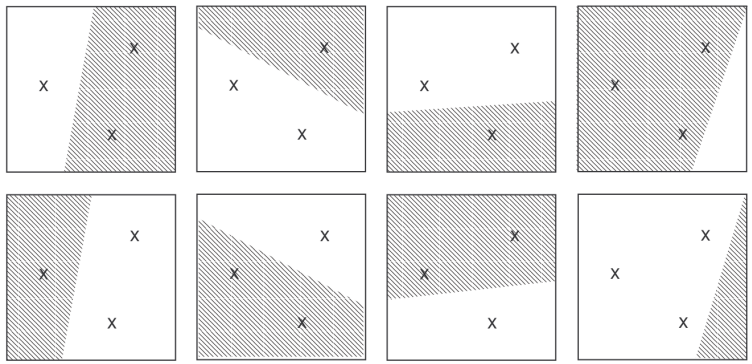
> VC-dimension is defined as the largest number m, such that there exists a set of m data points which the function class can shatter (and ∞ if no such m exists).

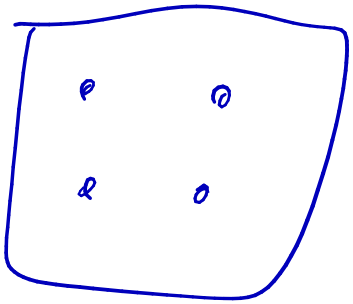


Definition VC-Dimension

Vapnik & Chervonenkis proposed **VC-dimension** h to estimate the capacity of a function class:

> VC-dimension is defined as the largest number m, such that there exists a set of m data points which the function class can shatter (and ∞ if no such m exists).





What is the VC-dimension of the function class of linear functions in $\mathbb{R}^2$? $2, 3, 4, \infty$

Definition VC-Dimension

Vapnik & Chervonenkis proposed **VC-dimension** h to estimate the capacity of a function class:

> VC-dimension is defined as the largest number m, such that there exists a set of m data points which the function class can shatter (and ∞ if no such m exists).

What is the VC-dimension of the function class of linear functions in $\mathbb{R}^2$? $2, 3, 4, \infty$

What is the VC-dimension of the function class of linear functions (2D-hyperplanes) in $\mathbb{R}^3$? $2, 3, 4, \infty$

Bound the Risk!

A low so-called empirical risk / average training error with zero-one loss:

$$R_{emp}[f] = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} |f(x_i) - y_i|$$

does not imply a small test error ("risk") on novel test examples draw from the underlying data distribution $P(x, y)$:

$$R[f] = \int \frac{1}{2} |f(x) - y| \, dP(x, y)$$

Bound the Risk!

A low so-called empirical risk / average training error with zero-one loss:

$$R_{emp}[f] = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} |f(x_i) - y_i|$$

does not imply a small test error ("risk") on novel test examples draw from the underlying data distribution $P(x, y)$:

$$R[f] = \int \frac{1}{2} |f(x) - y| \, dP(x, y)$$

With (*some*) high probability, the VC-bound

$$R[f] \leq R_{emp}[f] + \phi(h, m, \delta)$$

holds, with the capacity term ϕ describing the capacity of the function class (h being the VC-dimension).

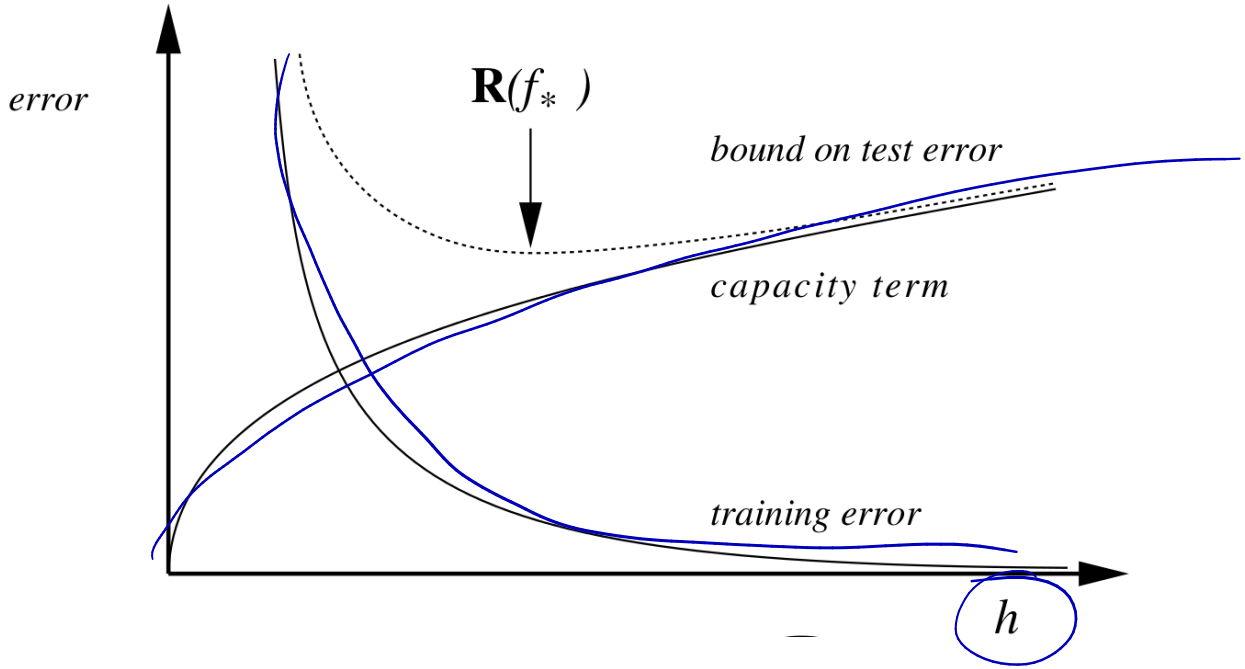
Bound the Risk!

Remark: Large VC-dimension h increases ϕ.

$$R[f] \leq R_{\text{emp}}[f] + \sqrt{\frac{h\left(\log\frac{2m}{h} + 1\right) - \log(\delta/4)}{m}}$$

(For details refer to "Learning with Kernels", Sec. 1.3)

Bound the Risk!



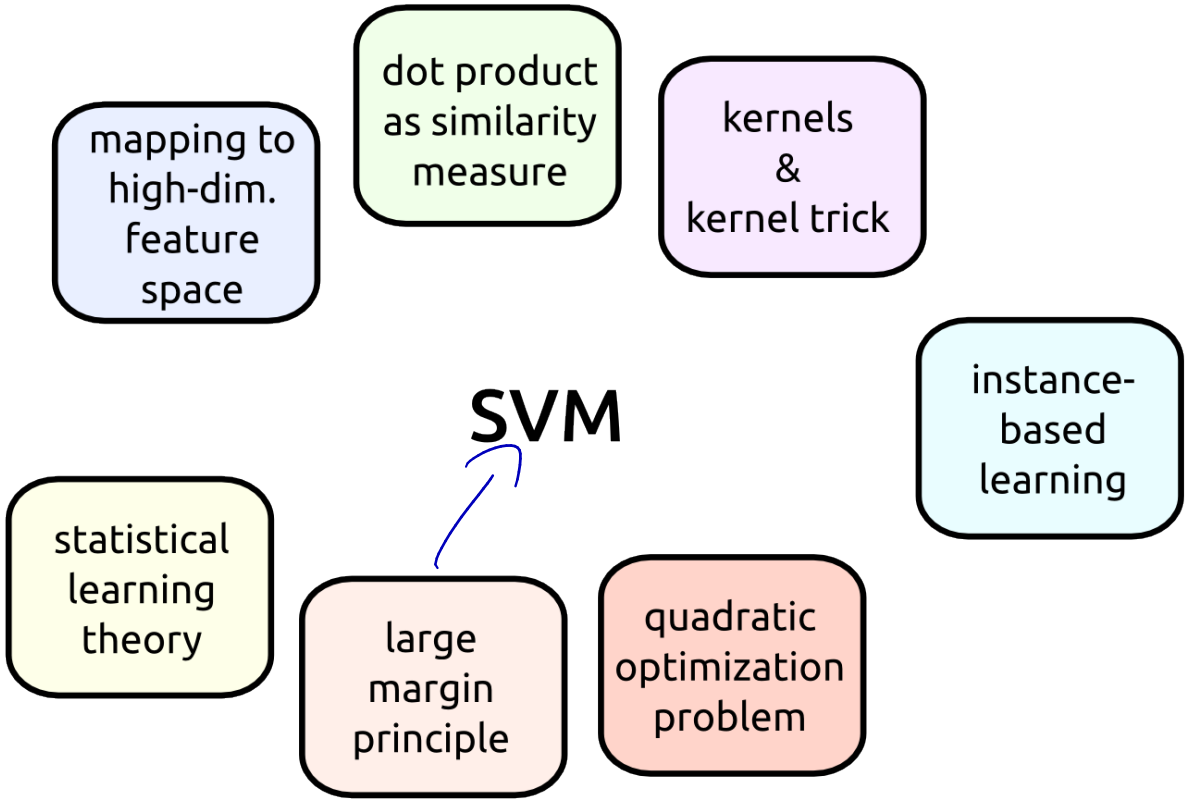
Implications:

- Large VC-dimension h of a function class implies a small empirical risk, but enlarges overall risk!
- A practitioner should prefer function classes of lower capacity, if they perform equally well on the training data, or **restrict** its capacity.

How could such a restriction be carried out? Regularization!

Lecture Overview

Concept: Large Margin Principle



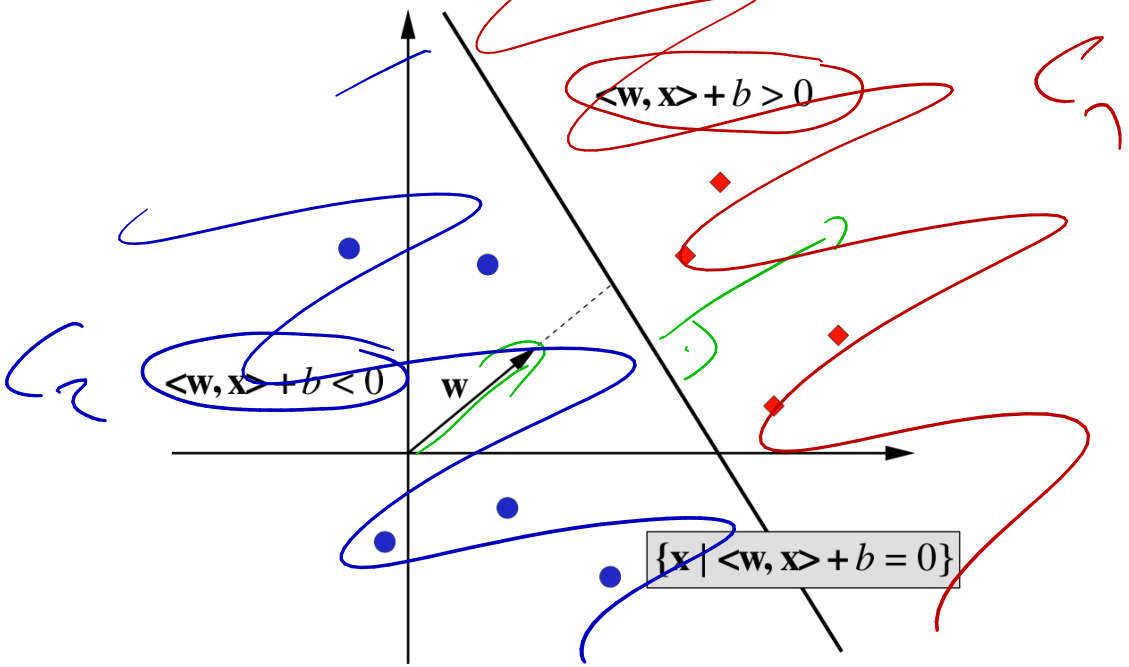
Hyperplane Classifiers

Let's consider the function class of hyperplanes in a dot product space $\mathcal{H}$:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$$

where $\mathbf{w}, \mathbf{x} \in \mathcal{H}, b \in \mathbb{R}$, equipped with the decision function

$$f(\mathbf{x}) = sng\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right)$$



Find the **Optimal** Hyperplane Classifier

For linear separable problems – which criterion could be used to define *"optimal"*?

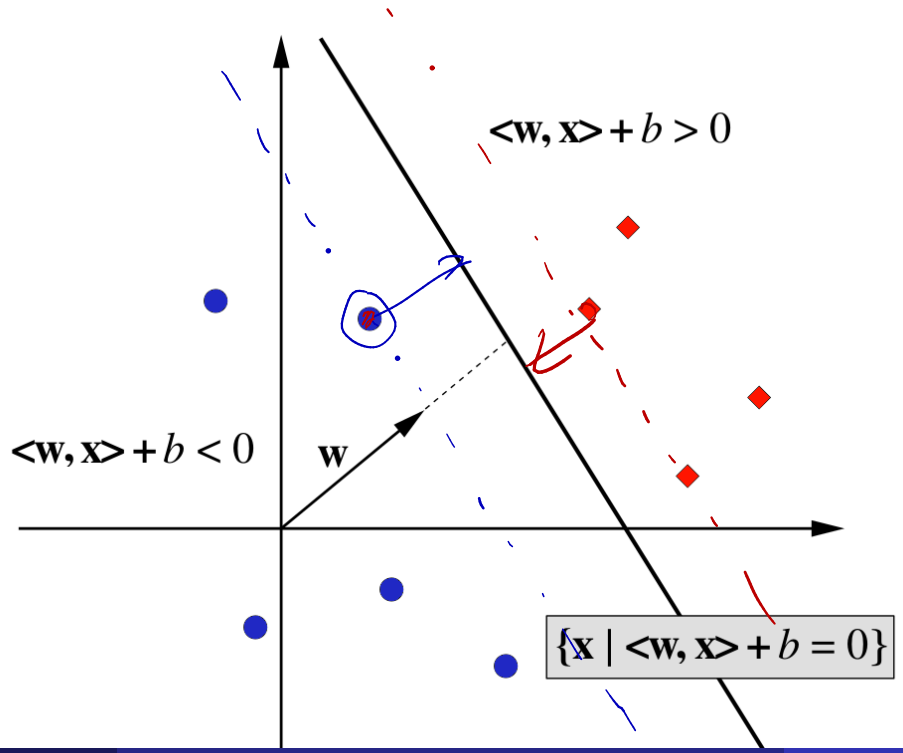
Find the **Optimal** Hyperplane Classifier

For linear separable problems – which criterion could be used to define *"optimal"*?

Vapnik et al. proposed to use the unique hyperplane with the maximum margin of separation between any training point and the hyperplane.

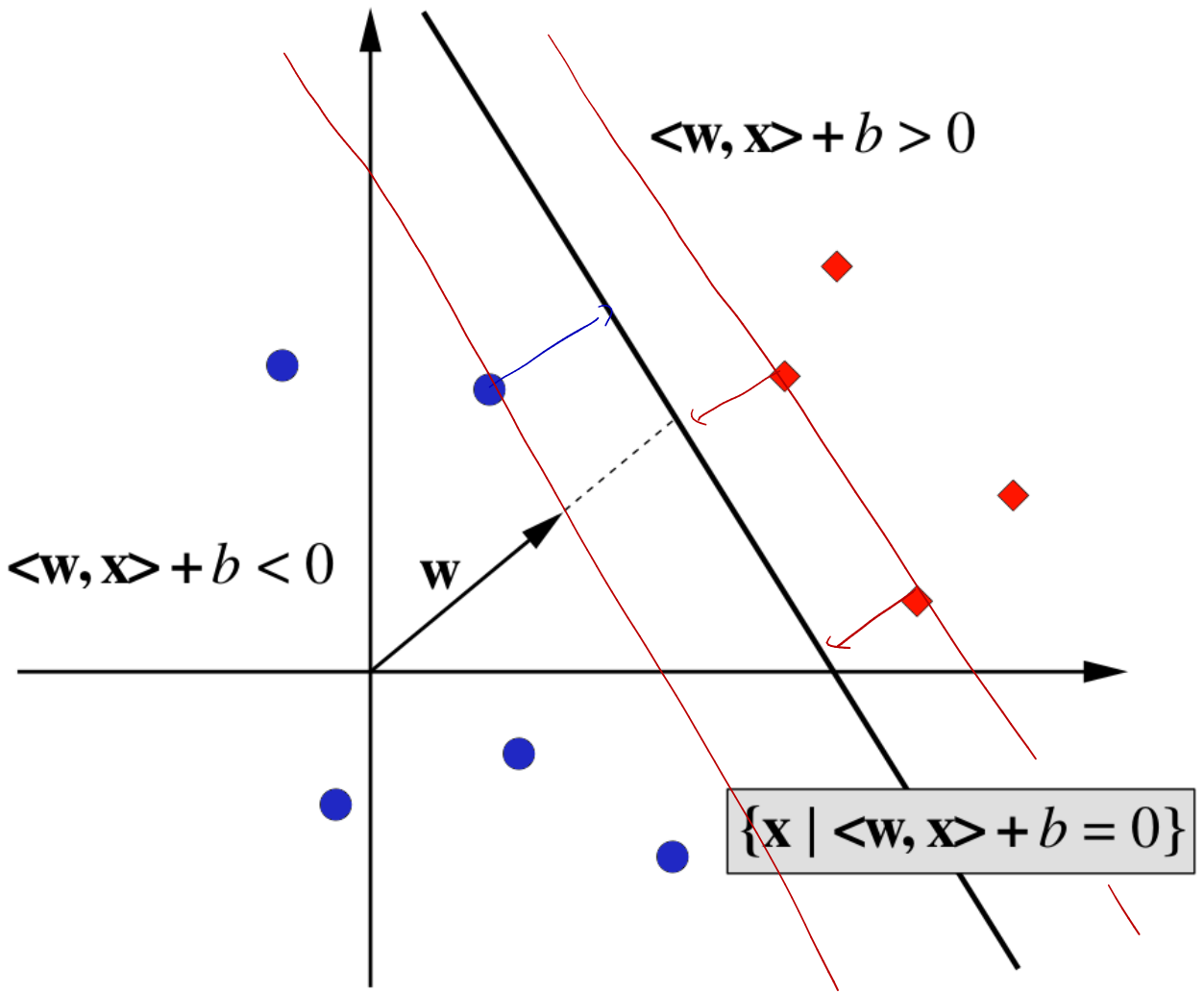
It is the solution of:

$$\underset{\mathbf{w}\in\mathcal{H}, b\in\mathbb{R}}{\text{argmax}}\ min\{\|\mathbf{x}-\mathbf{x}_i\| \mid \mathbf{x}\in\mathcal{H},\ \langle\mathbf{w},\mathbf{x}\rangle + b = 0,\ i = 1,\ldots,m\}$$

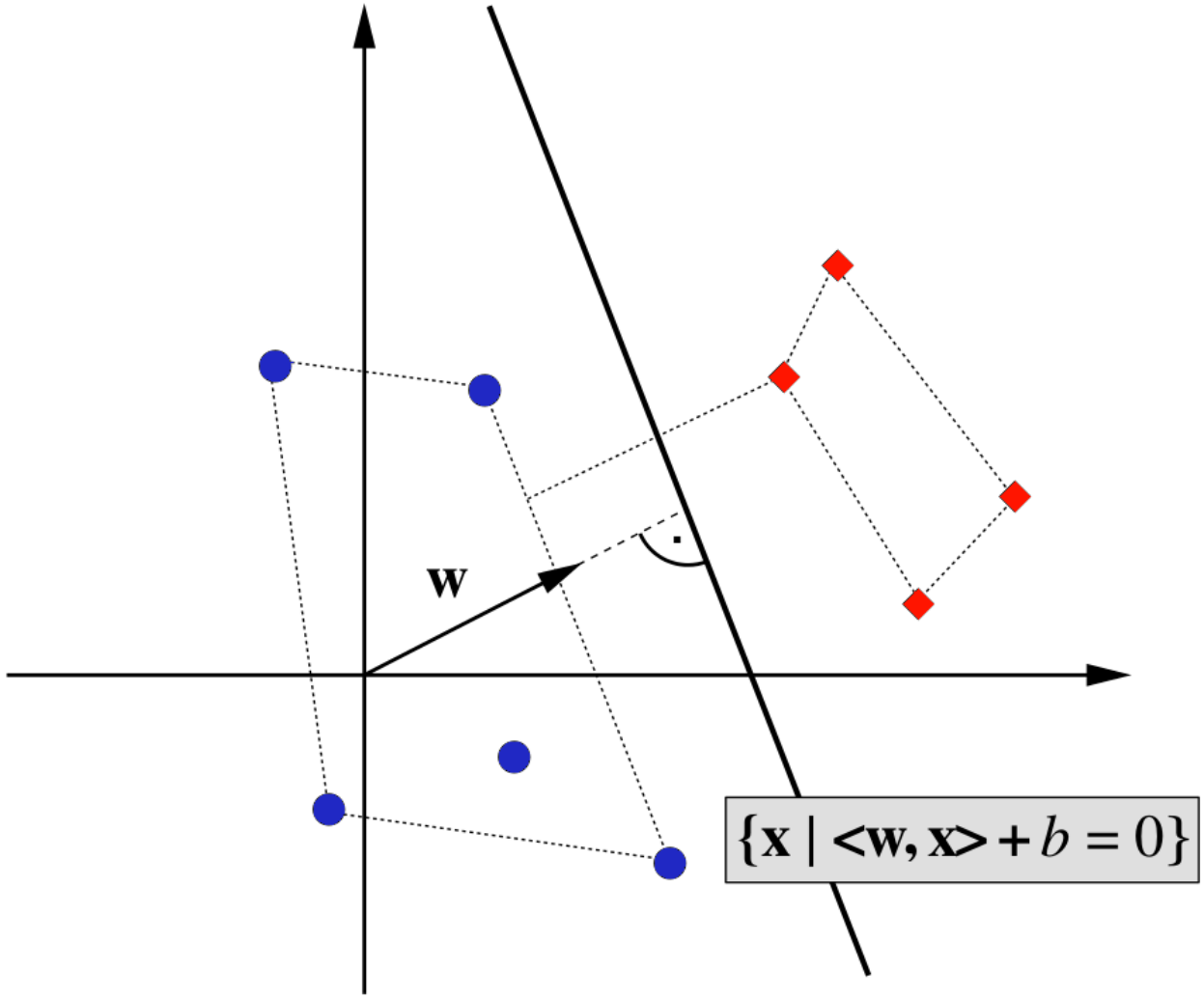


Find the **Optimal** Hyperplane Classifier

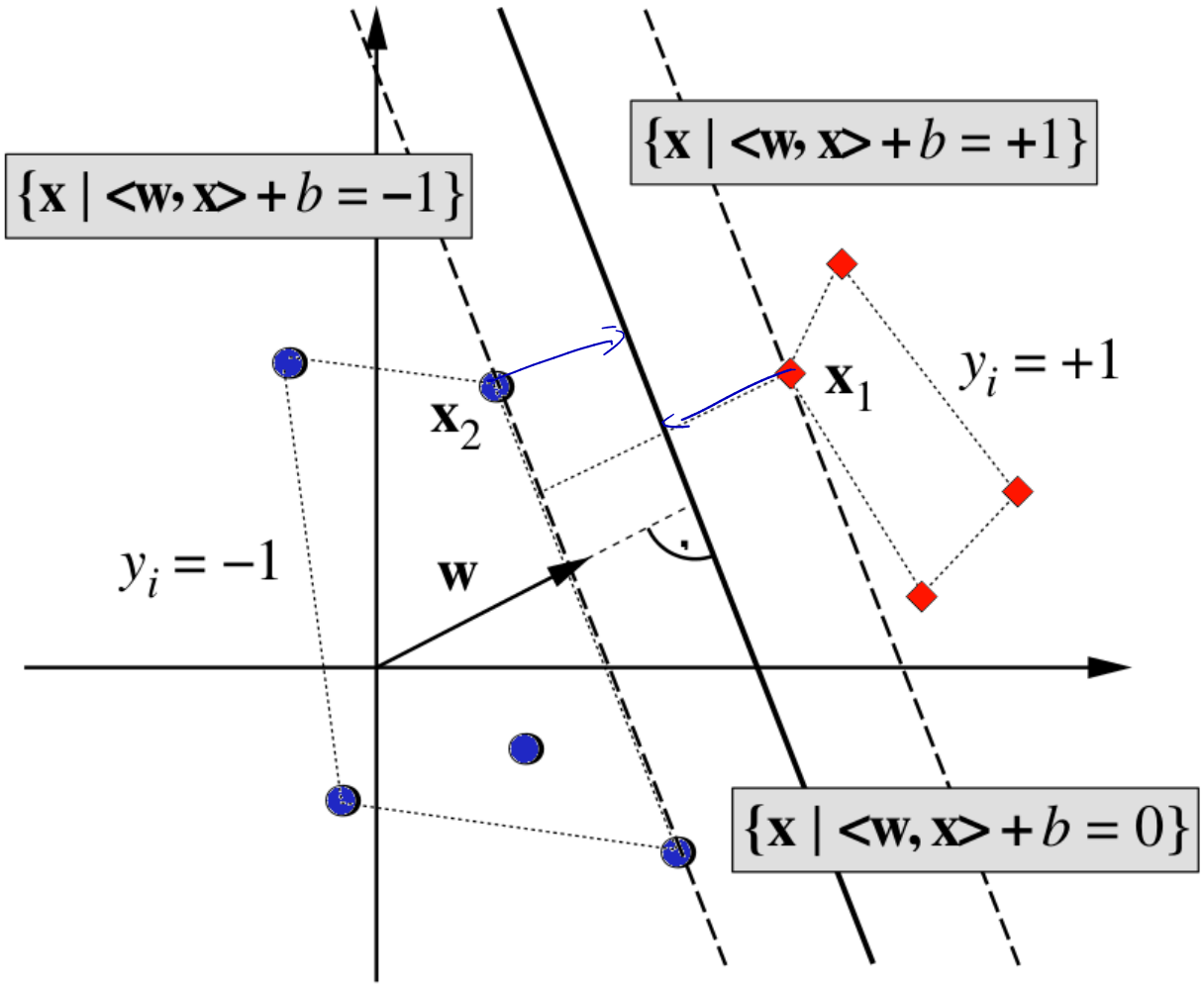
$$\underset{\mathbf{w}\in\mathcal{H},\, b\in\mathbb{R}}{\text{argmax}}\ min\{\|\mathbf{x}-\mathbf{x}_i\| \mid \mathbf{x}\in\mathcal{H},\ \langle\mathbf{w},\mathbf{x}\rangle+b=0,\ i=1,\ldots,m\}$$



Minimal $\mathbf{w}$ Implies Large Margin Implies **Low Capacity**

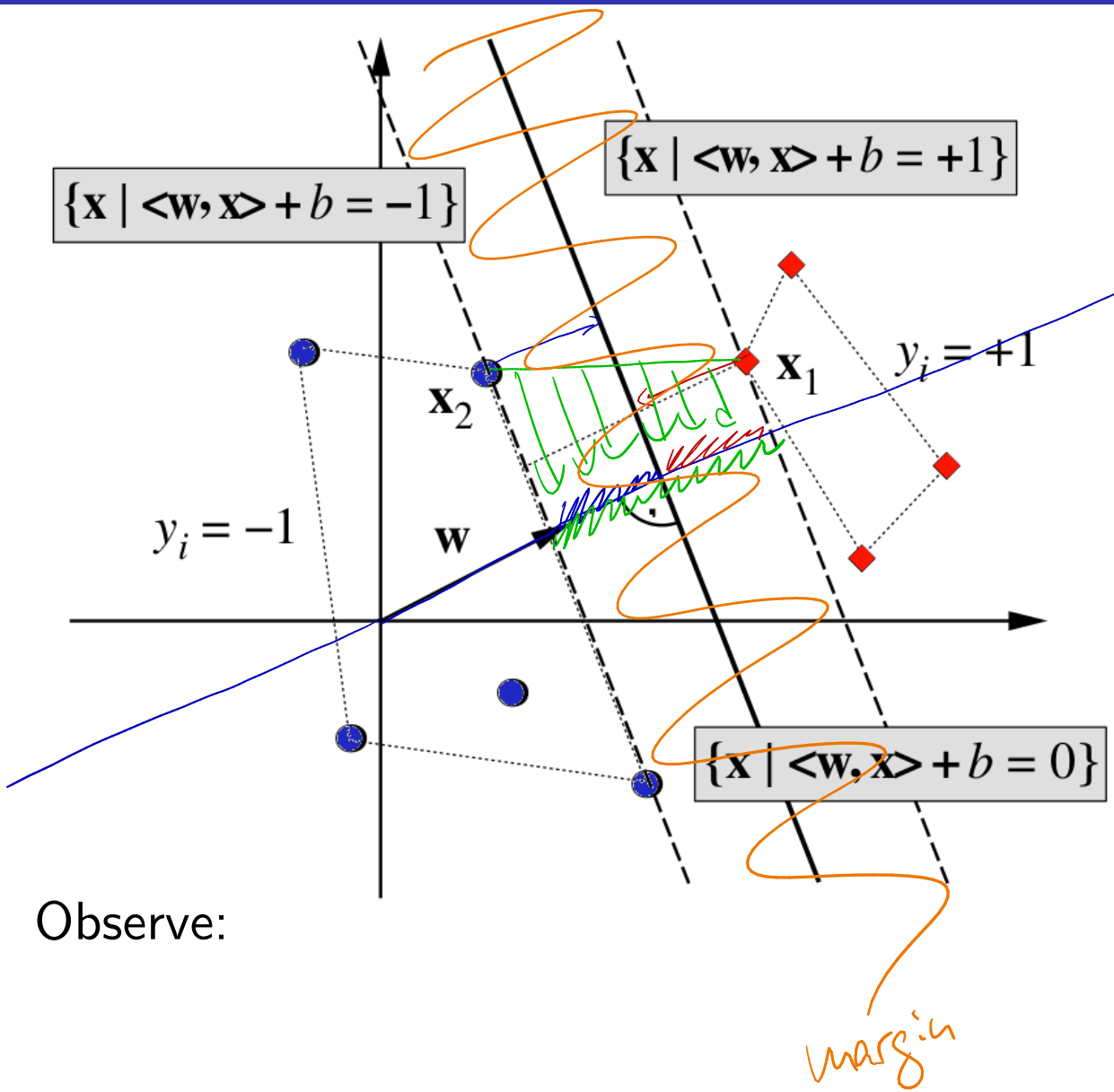


Minimal $\mathbf{w}$ Implies Large Margin Implies **Low Capacity**



Observe:

Minimal $\mathbf{w}$ Implies Large Margin Implies **Low Capacity**



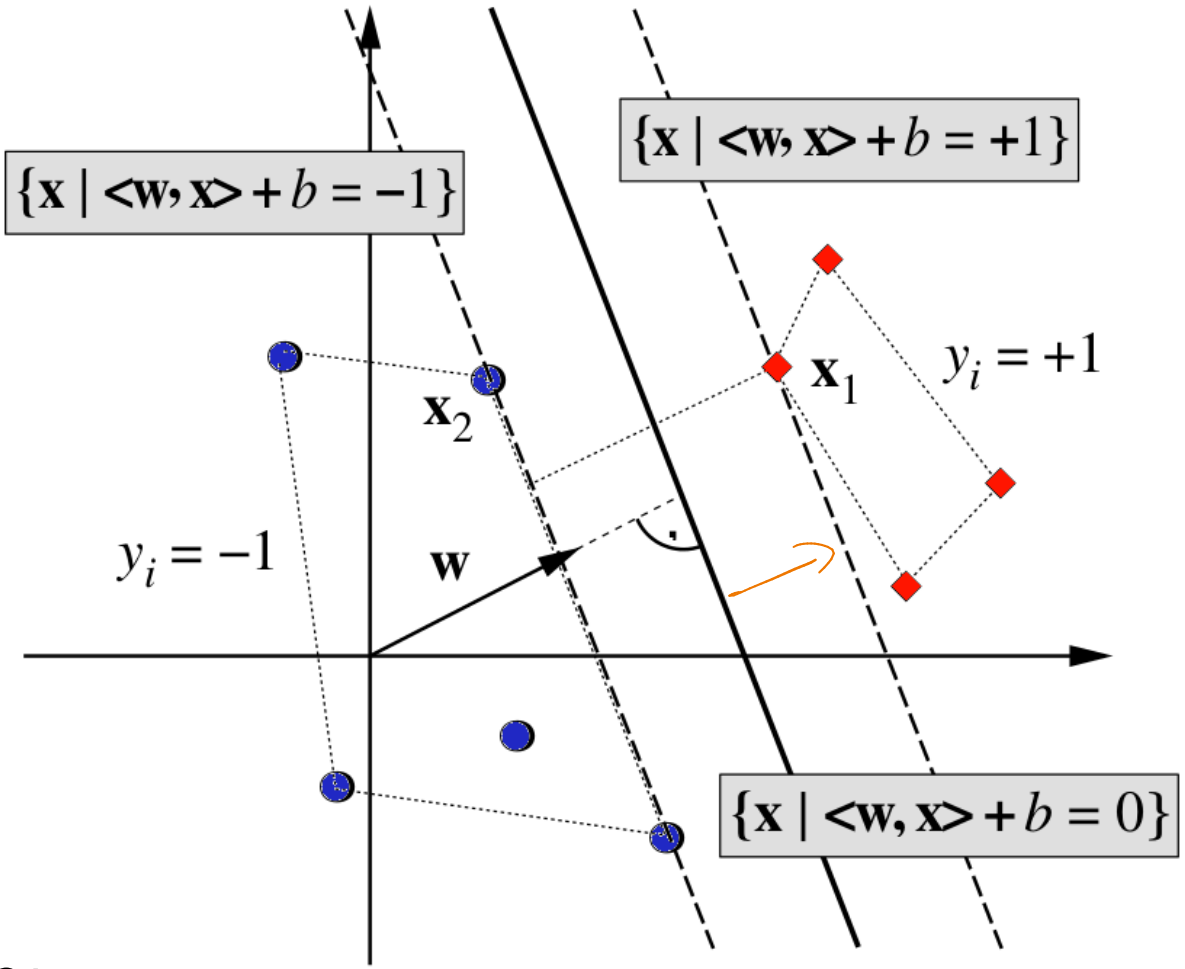
Note:

$$\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = +1$$
$$\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$$
$$\Rightarrow \quad \langle \mathbf{w}, (\mathbf{x}_1 - \mathbf{x}_2) \rangle = 2$$
$$\Rightarrow \quad \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_1 - \mathbf{x}_2) \right\rangle = \frac{2}{\|\mathbf{w}\|}$$

length 1
w is one

Observe:

Minimal $\mathbf{w}$ Implies Large Margin Implies **Low Capacity**



Note:

$$\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = +1$$
$$\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$$
$$\Rightarrow \quad \langle \mathbf{w}, (\mathbf{x}_1 - \mathbf{x}_2) \rangle = 2$$
$$\Rightarrow \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_1 - \mathbf{x}_2) \right\rangle = \frac{2}{\|\mathbf{w}\|}$$

Observe:

1. The optimal hyperplane has the largest margin. (1)
2. The largest margin is defined by the shortest normal vector $\mathbf{w}$ (... which still leads to correct classification). (2)

The Constrained Optimization Problem

The optimal hyperplane classifier leads to the following constrained optimization problem:

$$\underset{\mathbf{w}\in\mathcal{H},\, b\in\mathbb{R}}{\operatorname{argmin}} \ \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \text{for all} \quad i = 1, \ldots, m.$$

- Function τ is called the **objective function**
- Constraints are **inequality constraints**, which ensure, that the labels will be correct
- The "≥ 1" on the right hand side of the constraints effectively fix the scaling of $\mathbf{w}$.

Solving the Constrained Optimization Problem

$$\text{subject to} \quad \underset{\mathbf{w}\in\mathcal{H},\, b\in\mathbb{R}}{\operatorname{argmin}} \; \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \text{for all} \quad i = 1, \ldots, m.$$

Constrained optimization problems can be solved by introducing Lagrange multipliers $\alpha_i \in \mathbb{R}$, $\alpha_i \geq 0$, and by **minimizing** the Lagrangian:

$$\underset{}{\operatorname{argmin}} \quad L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i \left(\langle \mathbf{x}_i, \mathbf{w} \rangle + b\right) - 1)$$

Approach:

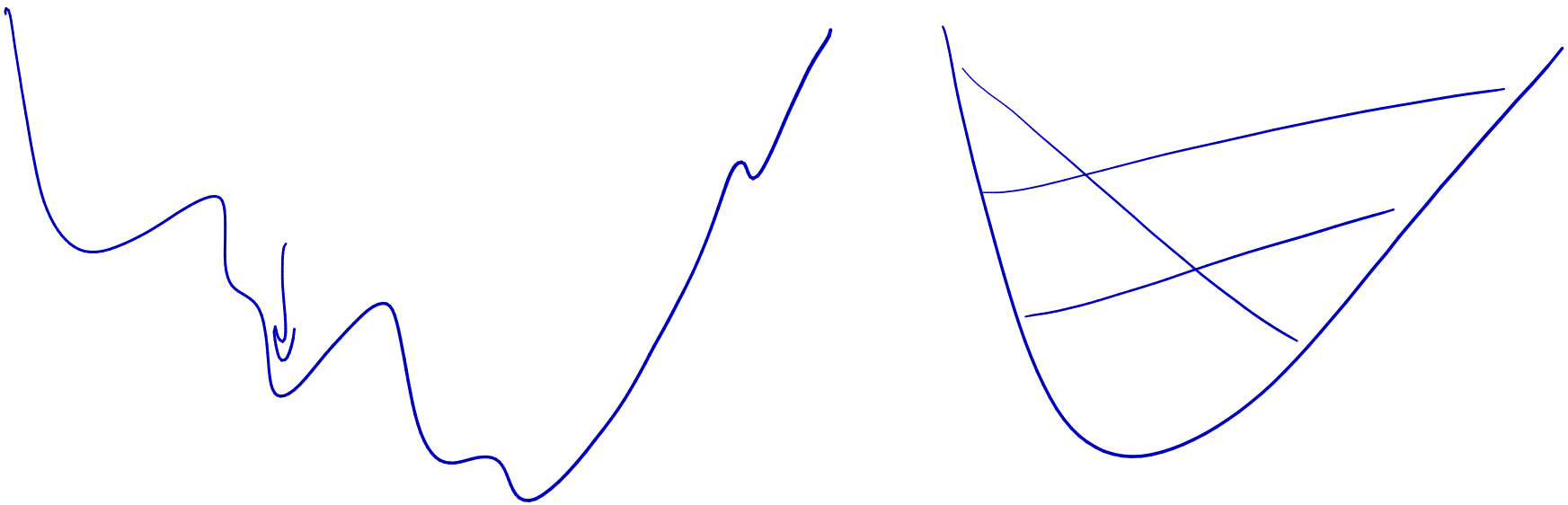
- Using the partial derivatives, minimize the Lagrangian L with respect to the primal variables $\mathbf{w}$ and b and maximize it with respect to the dual variables α_i. Effectively, this finds a solution in a saddle point.
- Desired: Eliminate the primal variables by substituting $\mathbf{w}$ and b.

Solving the Dual Optimization Problem

In practice, the resulting dual optimization problem is solved:

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\operatorname{argmax}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

subject to $\alpha_i \geq 0$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$



Solving the Dual Optimization Problem

In practice, the resulting dual optimization problem is solved:

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{argmax}}\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i\alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

subject to $\alpha_i \geq 0$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$

α_i are known

Once the optimization problem is solved, the hyperplane decision function for a novel data point $\mathbf{x}$ in the feature space $\mathcal{H}$ can be written as:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \left\langle \mathbf{x}, \mathbf{x}_i \right\rangle + b\right)$$

(Interpretation on black board)

Solving the Dual Optimization Problem

Hyperplane decision function:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b\right)$$

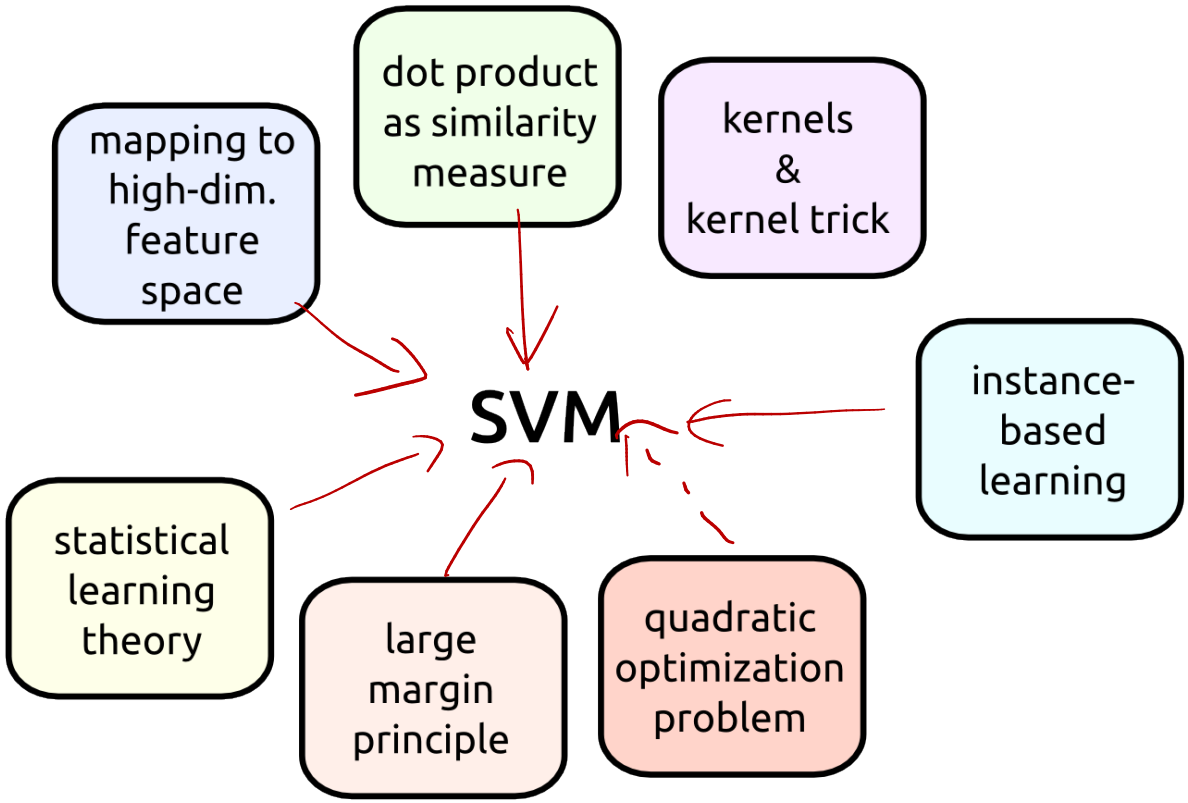
Observe:

- The solution can be expressed completely in terms of dot products of training data points $\mathbf{x}_i$ → "**instance based learning**" (cp. to k-Nearest Neighbour algorithm)
- In practical problems, only a subset of the Lagrange multipliers α_i are active and influence the decision hyperplane; corresponding training data points $\mathbf{x}_i$ are called **support vectors**.
- Other training data points have $\alpha_i = 0$. They do not define the shape of the hyperplane (see mechanical analogy from the beginning of the lecture).

Lecture Overview

Concepts towards SVM

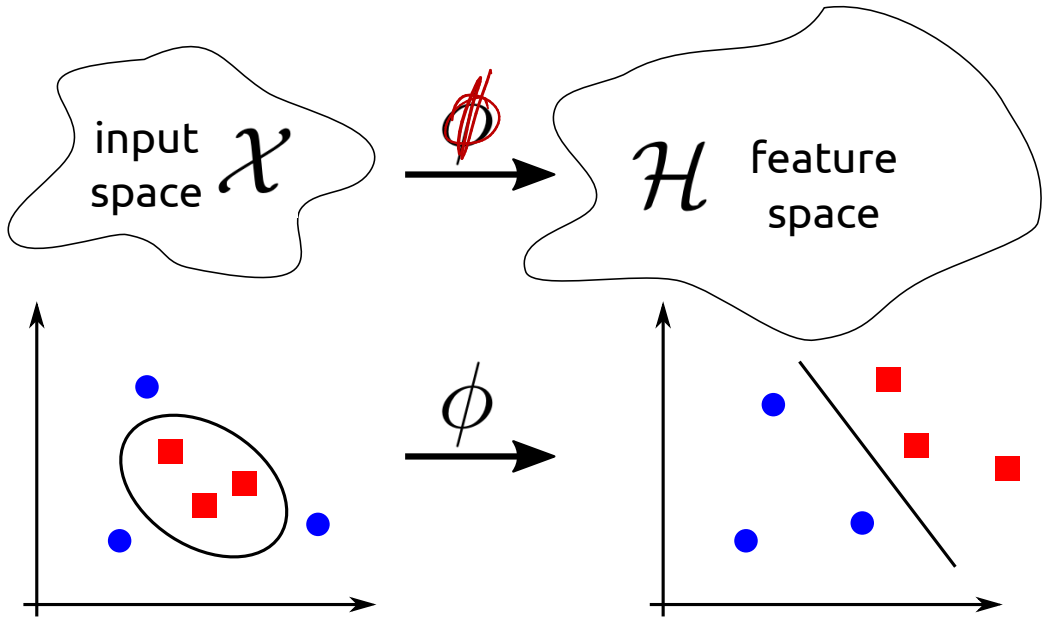
Most tools are ready now to describe support vector machines (SVMs).



To arrive at an SVM, they simply need to be put to work together.

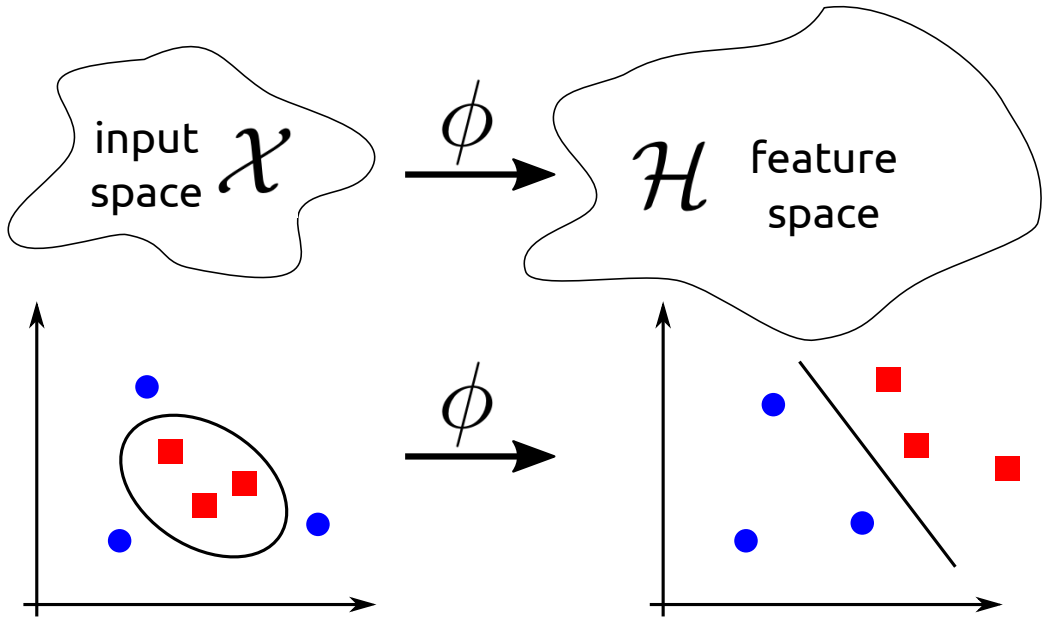
Support Vector Machine

- SVMs implement the large margin principle and thus share the optimization problem of optimal hyperplane classifiers.
- They make use of nonlinear mappings ϕ:



Support Vector Machine

- SVMs implement the large margin principle and thus share the optimization problem of optimal hyperplane classifiers.
- They make use of nonlinear mappings ϕ:



- Can we avoid performing the mapping ϕ explicitly?
- Can we avoid calculating dot products in feature space explicitly? (YES: kernel trick)

Kernel Trick

The kernel trick for SVM and other kernel methods is a substitution. It consists of the following ingredients:

- All computations can be formulated in a dot product space (think of it as the feature space $\mathcal{H}$)
- All computations can be executed as dot product operations in $\mathcal{H}$.
- To express formulas in terms of the input patterns in $\mathcal{X}$, we can make use of a kernel function k.
- This kernel function expresses the dot product of bold face feature vectors $\mathbf{x}, \mathbf{x}'$ in terms of the kernel k evaluated on input patterns x, x'!

$$k(x, x') := \langle \mathbf{x}, \mathbf{x}' \rangle$$

from input space

from feature space

Kernel Trick

The kernel trick for SVM and other kernel methods is a substitution. It consists of the following ingredients:

- All computations can be formulated in a dot product space (think of it as the feature space $\mathcal{H}$)
- All computations can be executed as dot product operations in $\mathcal{H}$.
- To express formulas in terms of the input patterns in $\mathcal{X}$, we can make use of a kernel function k.
- This kernel function expresses the dot product of bold face feature vectors $\mathbf{x}, \mathbf{x}'$ in terms of the kernel k evaluated on input patterns x, x'!

$$k(x, x') := \left\langle \mathbf{x} \, , \mathbf{x}' \right\rangle$$

$\rightarrow$ **The kernel trick replaces the mapping ϕ and following dot product operations by a (simple) calculation in the input space!**

Examples of Kernels

Polynomial kernel (hyperparameter: d):

$$k(x, x') = \langle \mathbf{x}, \mathbf{x}' \rangle^d$$

Gaussian radial basis function kernels (parameter $\sigma > 0$):

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

Sigmoid kernel (parameter $k > 0$ and parameter $\Theta < 0$):

$$k(x, x') = \tanh\left(k \langle \mathbf{x}, \mathbf{x}' \rangle + \Theta\right)$$

Decision Function for SVM

Reminder: hyperplane decision function for linear (separable) case in $\mathcal{H}$:

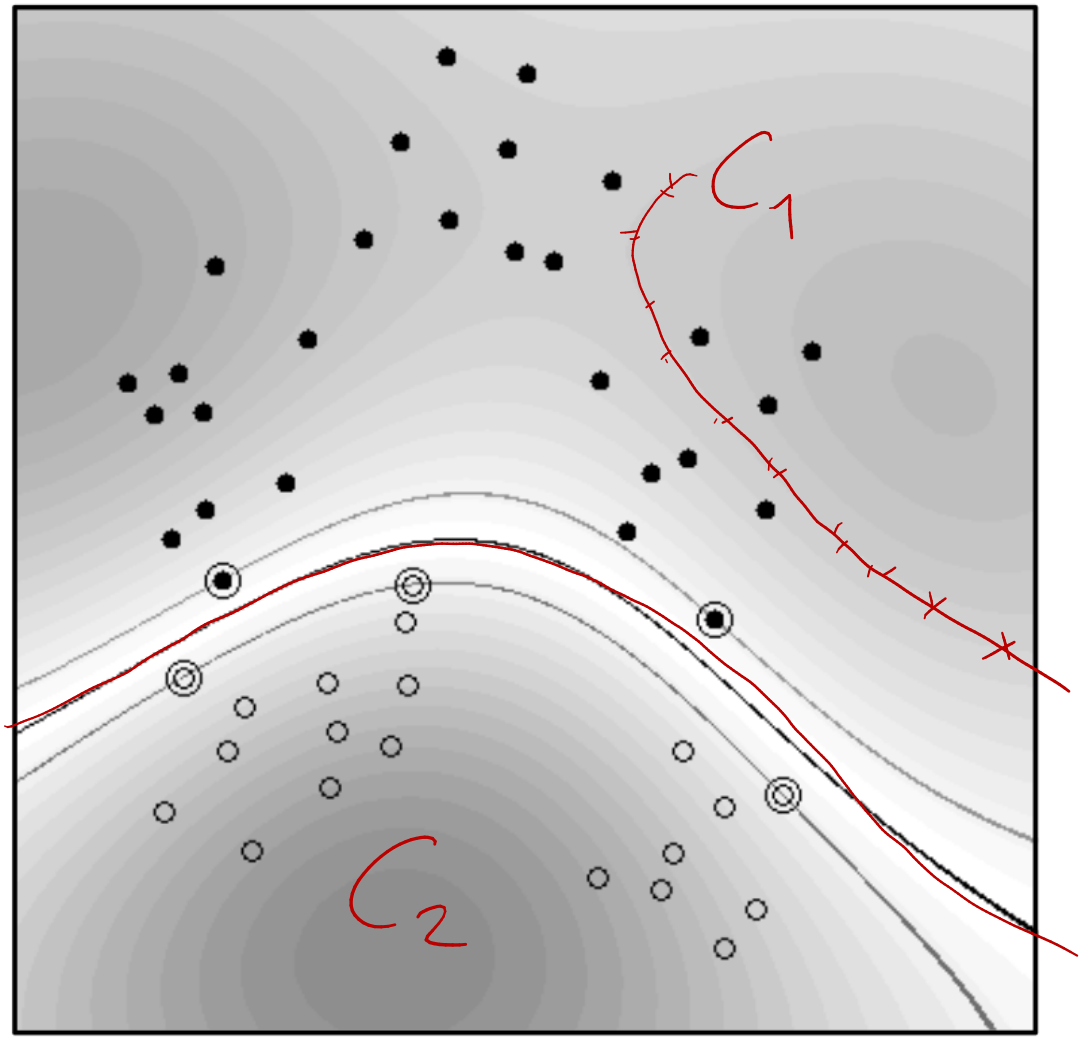
$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b\right)$$

Decision function for SVM including the kernel trick:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle + b\right) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i k(x, x_i) + b\right)$$

$\rightarrow$ Computations are carried out in the input space!

Example of SVM Classifier with Radial Basis Function Kernel



Why is the Kernel Trick Useful?

- Computations in the input space usually are preferred compared to an explicit mapping into a high dimensional (and potentially even infinite-dimensional feature space)!
- Every linear algorithm, which can be expressed by dot product operations, can be "kernelized", thus leading to a non-linear version of the algorithm. Example: kernel-PCA (see Sec. 1.7 and Sec. 14 of "Learning with Kernels")

SVM – Pros and Cons

- Pro: SVM does not make (strong) assumptions about distributions
- Pro: High dimensionality of input data does not hurt much
- Pro: Very robust performance, even when used as black box!
- Pro: Inspecting the selected support vectors may help to understand the problem better
- Pro: SVM formulation can easily be expanded to non-separable cases in the feature space (next lecture):
  - Inclusion of slack variables ξ_i, which allow for a *slight* violation of the margin
  - Penalty for slack variables must be determined at training time by a regularization parameter C or ν (different formulations exist)
- Pro: Very few hyperparameters to tune.
- Pro: Formulation for support vector *regression* and for *one-class SVM* (for outlier detection) both are easy to obtain from the SVM classification formulation.

SVM – Pros and Cons

- Con: Kernel computations get expensive with many training data points! (Runtime for training? Runtime for recall?). Chunking methods like sequential minimal optimization (SMO) avoid calculating the full kernel matrix effectively, though...
- Con: Formerly outstanding performance in benchmarks has been overrun by Deep Neural Networks (at least if training data set is huge).
- Con: compared to truly linear parametric methods, interpretation of a trained SVM is harder.

Wrap-Up: Summary by Learning Goals

Having heard this lecture and done the assignment, you will be able to ...

- describe the concepts underlying kernel methods
- understand, why it is necessary to restrict the capacity of a learning machine
- explain (non-linear) mapping to a feature space and the kernel trick
- formulate the optimization problem for optimal large-margin hyperplane classifiers
- explain the meaning of instant-based learning vs. model-based learning
- explain the role of support vectors and their interpretation with respect to the Lagrange multipliers

Preview of Assignment for Kernel Methods

- Definition of kernels, check definitions on kernel candidates
- Get acquainted with SVM and its hyperparameters
- Get acquainted with k-PCA by using implementations of scikit-learn