

# Learning with Kernels

---

*Bernhard Schölkopf*

*Max-Planck-Institut für biologische Kybernetik*

*72076 Tübingen, Germany*

*bs@tuebingen.mpg.de*

# Roadmap

---

- Elements of Statistical Learning Theory
- Kernels and feature spaces
- Support vector algorithms and other kernel methods
- Applications

# Roadmap of Today

---

- Informal introduction to ideas of machine learning
- Learning theory: Uniform convergence

## Learning and Similarity: some Informal Thoughts

---

- input/output sets  $\mathcal{X}, \mathcal{Y}$
- training set  $(x_1, y_1), \dots, (x_m, y_m) \in \mathcal{X} \times \mathcal{Y}$
- “generalization”: given a previously unseen  $x \in \mathcal{X}$ , find a suitable  $y \in \mathcal{Y}$
- $(x, y)$  should be “similar” to  $(x_1, y_1), \dots, (x_m, y_m)$
- how to measure similarity?
  - for outputs: *loss function* (e.g., for  $\mathcal{Y} = \{\pm 1\}$ , zero-one loss)
  - for inputs: *kernel*

## Similarity of Inputs

---

- symmetric function

$$\begin{aligned}k &: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \\(x, x') &\mapsto k(x, x')\end{aligned}$$

- for example, if  $\mathcal{X} = \mathbb{R}^N$ : canonical dot product

$$k(x, x') = \sum_{i=1}^N [x]_i [x']_i$$

- if  $\mathcal{X}$  is not a dot product space: assume that  $k$  has a **representation** as a dot product in a linear space  $\mathcal{H}$ , i.e., there exists a map  $\Phi : \mathcal{X} \rightarrow \mathcal{H}$  such that

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle .$$

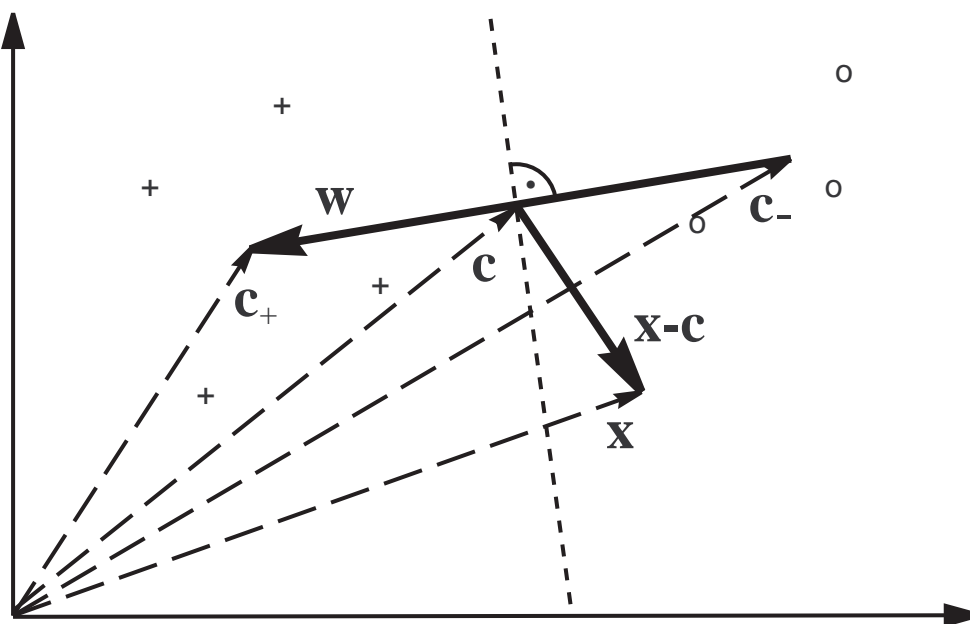
- in that case, we can think of the patterns as  $\Phi(x), \Phi(x')$ , and carry out geometric algorithms in the dot product space (“**feature space**”)  $\mathcal{H}$ .

## An Example of a Kernel Algorithm

---

Idea: classify points  $\mathbf{x} := \Phi(x)$  in feature space according to which of the two **class means** is closer.

$$\mathbf{c}_+ := \frac{1}{m_+} \sum_{y_i=1} \Phi(x_i), \quad \mathbf{c}_- := \frac{1}{m_-} \sum_{y_i=-1} \Phi(x_i)$$



Compute the sign of the dot product between  $\mathbf{w} := \mathbf{c}_+ - \mathbf{c}_-$  and  $\mathbf{x} - \mathbf{c}$ .

## An Example of a Kernel Algorithm, ctd. [56]

---

$$\begin{aligned} f(x) &= \operatorname{sgn} \left( \frac{1}{m_+} \sum_{\{i:y_i=+1\}} \langle \Phi(x), \Phi(x_i) \rangle - \frac{1}{m_-} \sum_{\{i:y_i=-1\}} \langle \Phi(x), \Phi(x_i) \rangle + b \right) \\ &= \operatorname{sgn} \left( \frac{1}{m_+} \sum_{\{i:y_i=+1\}} k(x, x_i) - \frac{1}{m_-} \sum_{\{i:y_i=-1\}} k(x, x_i) + b \right) \end{aligned}$$

where

$$b = \frac{1}{2} \left( \frac{1}{m_-^2} \sum_{\{(i,j):y_i=y_j=-1\}} k(x_i, x_j) - \frac{1}{m_+^2} \sum_{\{(i,j):y_i=y_j=+1\}} k(x_i, x_j) \right).$$

- provides a geometric interpretation of Parzen windows
- the decision function is a hyperplane. Will it generalize well?

## An Example of a Kernel Algorithm, ctd.

---

- Demo
- Exercise: derive the Parzen windows classifier by computing the distance criterion directly



# Statistical Learning Theory

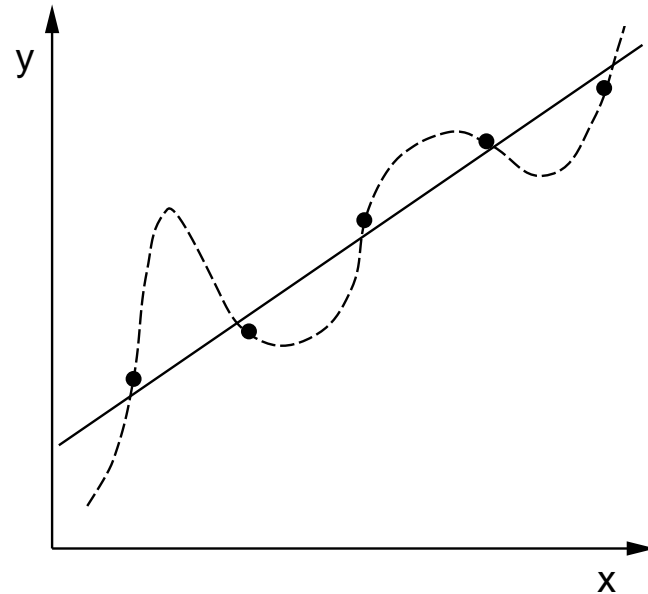
---

1. started by Vapnik and Chervonenkis in the Sixties
2. model: we observe data generated by an unknown stochastic regularity
3. *learning* = extraction of the regularity from the data
4. the analysis of the learning problem leads to notions of *capacity* of the function classes that a learning machine can implement.
5. *support vector machines* use a particular type of function class: classifiers with large “margins” in a feature space induced by a *kernel*.

[72, 73]

## Example: Regression Estimation

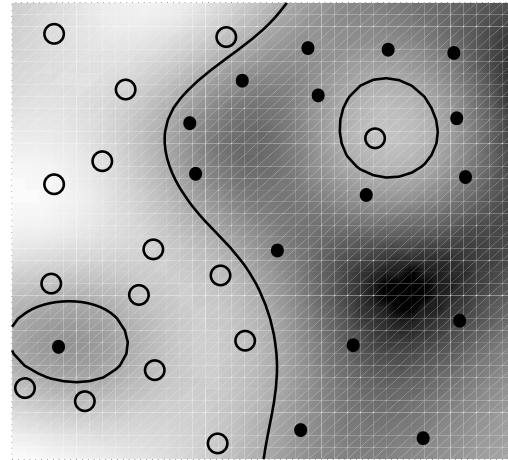
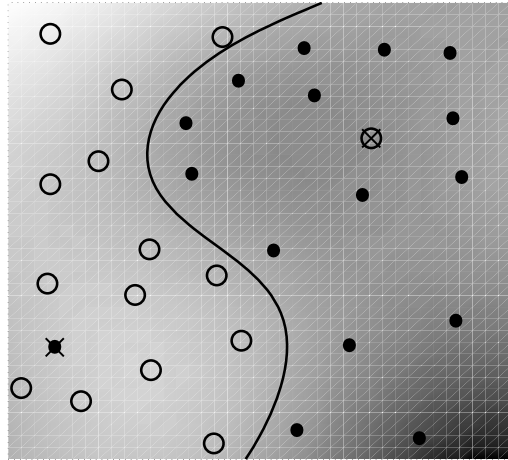
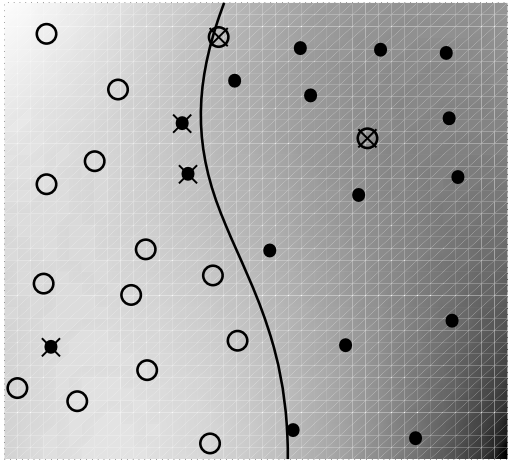
---



- *Data*: input-output pairs  $(x_i, y_i) \in \mathbb{R} \times \mathbb{R}$
- *Regularity*:  $(x_1, y_1), \dots, (x_m, y_m)$  drawn from  $P(x, y)$
- *Learning*: choose a function  $f : \mathbb{R} \rightarrow \mathbb{R}$  such that the error, averaged over  $P$ , is minimized.
- *Problem*:  $P$  is unknown, so the average cannot be computed — need an “*induction* principle”

# Example: Pattern Recognition

---



# Pattern Recognition

---

Learn  $f : \mathcal{X} \rightarrow \{\pm 1\}$  from examples

$(x_1, y_1), \dots, (x_m, y_m) \in \mathcal{X} \times \{\pm 1\}$ , generated i.i.d. from  $P(x, y)$ ,

such that the expected misclassification error on a test set, also drawn from  $P(x, y)$ ,

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

is minimal (*Risk Minimization (RM)*).

**Problem:**  $P$  is unknown.  $\longrightarrow$  need an *induction principle*.

*Empirical risk minimization (ERM)*: replace the average over  $P(x, y)$  by an average over the training sample, i.e. **minimize the training error**

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

- **Regression estimation.** RM: minimize

$$R[f] = \int (f(x) - y)^2 dP(x, y)$$

— leads to the *regression*  $y(x) = \int y dP(y|x)$ .

ERM gives **least mean squares**: minimize

$$\sum_i (f(x_i) - y_i)^2$$

- **Density estimation.** RM: minimize

$$R[f] = \int (-\log p(x)) dP(x)$$

ERM gives **maximum likelihood estimation**: maximize

$$\sum_i \log p(x_i) = \log\left(\prod_i p(x_i)\right)$$

# Convergence of Means to Expectations

---

Law of large numbers:

$$R_{\text{emp}}[f] \rightarrow R[f]$$

as  $m \rightarrow \infty$ .

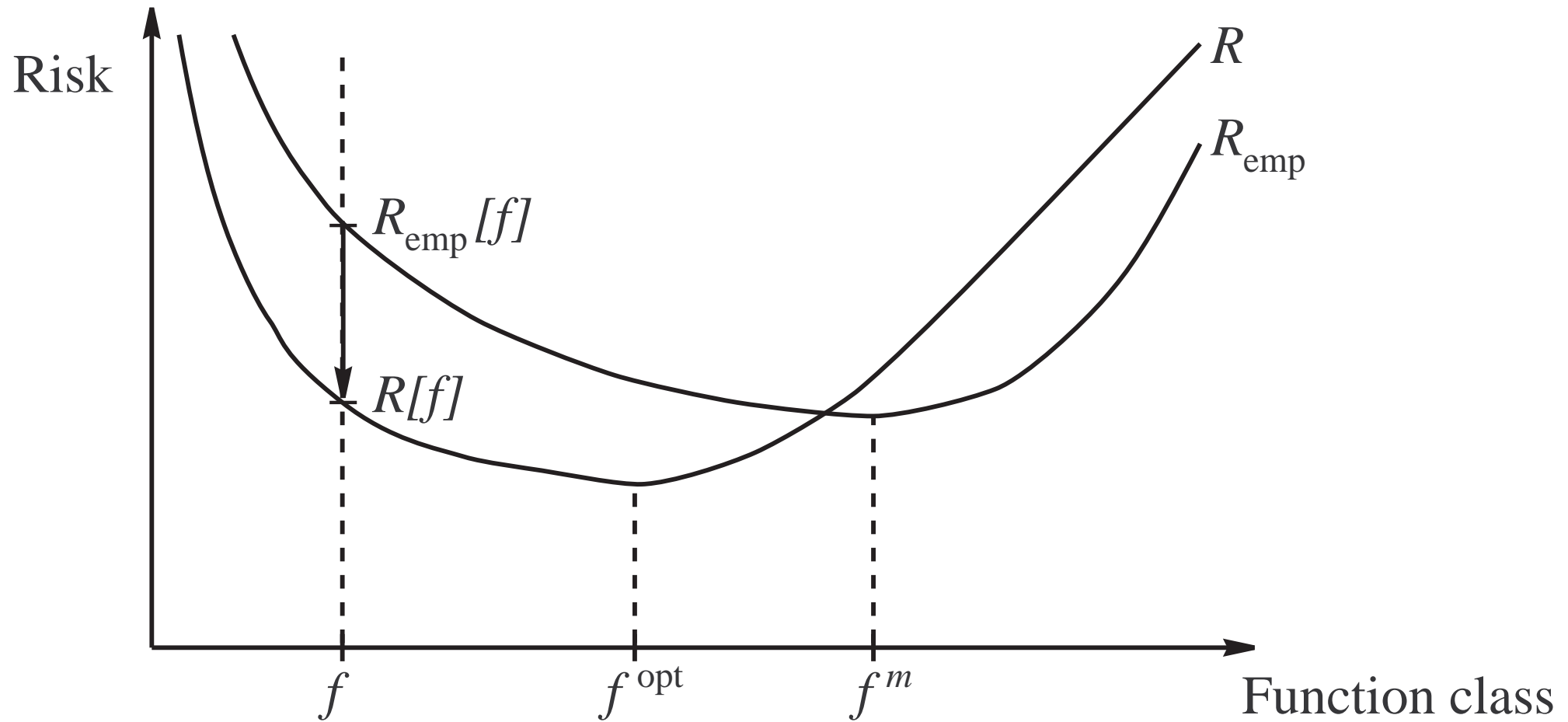
Does this imply that empirical risk minimization will give us the optimal result in the limit of infinite sample size (“*consistency*” of empirical risk minimization)?

**No.**

Need a *uniform* version of the law of large numbers. Uniform over *all functions that the learning machine can implement*.

# Consistency and Uniform Convergence

---



## The Importance of the Set of Functions

---

What about allowing *all* functions from  $\mathcal{X}$  to  $\{\pm 1\}$ ?

Training set  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m) \in \mathcal{X} \times \{\pm 1\}$

Test patterns  $\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_{\bar{m}} \in \mathcal{X}$ ,

such that  $\{\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_{\bar{m}}\} \cap \{\mathbf{x}_1, \dots, \mathbf{x}_m\} = \{\}$ .

For any  $f$  there exists  $f^*$  s.t.:

1.  $f^*(\mathbf{x}_i) = f(\mathbf{x}_i)$  for all  $i$
2.  $f^*(\bar{\mathbf{x}}_j) \neq f(\bar{\mathbf{x}}_j)$  for all  $j$ .

Based on the training set alone, there is *no* means of choosing which one is better. On the test set, however, they give *opposite* results. There is 'no free lunch' [32, 82].

→ a restriction must be placed on the *functions* that we allow



# Restricting the Class of Functions

---

Two views:

1. Statistical Learning (VC) Theory: take into account the *capacity* of the class of functions that the learning machine can implement
2. The Bayesian Way: place *Prior distributions*  $P(f)$  over the class of functions

## Detailed Analysis

---

- loss  $\xi_i := \frac{1}{2}|f(x_i) - y_i|$  in  $\{0, 1\}$
- the  $\xi_i$  are independent Bernoulli trials
- empirical mean  $\frac{1}{m} \sum_{i=1}^m \xi_i$  (by def: equals  $R_{\text{emp}}[f]$ )
- expected value  $\mathbf{E}[\xi]$  (equals  $R[f]$ )

## Chernoff's Bound

---

$$\mathbb{P} \left\{ \left| \frac{1}{m} \sum_{i=1}^m \xi_i - \mathbf{E}[\xi] \right| \geq \epsilon \right\} \leq 2 \exp(-2m\epsilon^2)$$

- here,  $\mathbb{P}$  refers to the probability of getting a sample  $\xi_1, \dots, \xi_m$  with the property  $\left| \frac{1}{m} \sum_{i=1}^m \xi_i - \mathbf{E}[\xi] \right| \geq \epsilon$  (is a product measure)

Useful corollary: Given a  $2m$ -sample of Bernoulli trials, we have

$$\mathbb{P} \left\{ \left| \frac{1}{m} \sum_{i=1}^m \xi_i - \frac{1}{m} \sum_{i=m+1}^{2m} \xi_i \right| \geq \epsilon \right\} \leq 4 \exp \left( -\frac{m\epsilon^2}{2} \right).$$

## Chernoff's Bound, II

---

Translate this back into machine learning terminology: the probability of obtaining an  $m$ -sample where the training error and test error differ by more than  $\epsilon > 0$  is bounded by

$$P \{ |R_{\text{emp}}[f] - R[f]| \geq \epsilon \} \leq 2 \exp(-2m\epsilon^2).$$

- refers to one fixed  $f$
- not allowed to look at the data before choosing  $f$ , hence not suitable as a bound on the test error of a learning algorithm using empirical risk minimization

## Two Observations

---

- denote the minimizer of  $R$  by  $f^{\text{opt}}$ ,  
and the minimizer of  $R_{\text{emp}}$  by  $f^m$ .

Then we have in particular

$$R[f^m] - R[f^{\text{opt}}] \geq 0$$

and

$$R_{\text{emp}}[f^{\text{opt}}] - R_{\text{emp}}[f^m] \geq 0.$$

- For consistency, would like the LHS of both to converge to 0 in probability.
- If the sum of the two converges to 0, we are done.

---

The sum of these two inequalities satisfies

$$\begin{aligned} 0 &\leq R[f^m] - R[f^{\text{opt}}] + R_{\text{emp}}[f^{\text{opt}}] - R_{\text{emp}}[f^m] \\ &= R[f^m] - R_{\text{emp}}[f^m] + R_{\text{emp}}[f^{\text{opt}}] - R[f^{\text{opt}}] \\ &\leq \sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) + (R_{\text{emp}}[f^{\text{opt}}] - R[f^{\text{opt}}]). \end{aligned}$$

- second half of RHS:  $f^{\text{opt}}$  is fixed (independent of training sample), hence by Chernoff: for all  $\epsilon > 0$ ,

$$\lim_{m \rightarrow \infty} \mathbb{P}\{|R_{\text{emp}}[f^{\text{opt}}] - R[f^{\text{opt}}]| > \epsilon\} = 0$$

(“convergence in probability”)

- 
- If the first half of RHS also converges to zero (in probability), i.e.,

$$\lim_{m \rightarrow \infty} \mathbb{P} \left\{ \sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon \right\} = 0,$$

for all  $\epsilon > 0$ , then

$$\begin{aligned} R[f^m] - R[f^{\text{opt}}] &\rightarrow 0 \\ R_{\text{emp}}[f^{\text{opt}}] - R_{\text{emp}}[f^m] &\rightarrow 0 \end{aligned}$$

in probability — in this case, empirical risk minimization can be seen to be *consistent*.

## Uniform Convergence (Vapnik & Chervonenkis)

---

*Necessary and sufficient* conditions for nontrivial consistency of empirical risk minimization (ERM):

One-sided convergence, uniformly over all functions that can be implemented by the learning machine.

$$\lim_{m \rightarrow \infty} P\left\{ \sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon \right\} = 0$$

for all  $\epsilon > 0$ .

- note that this takes into account the whole set of functions that can be implemented by the learning machine
- this is hard to check for a learning machine

Are there properties of learning machines ( $\equiv$  sets of functions) which ensure uniform convergence of risk?



## How to Prove a VC Bound

---

Take a closer look at  $P\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\}$ .

Plan:

- if the function class  $\mathcal{F}$  contains only one function, then Chernoff's bound suffices:

$$P\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\} \leq 2 \exp(-2m\epsilon^2).$$

- if there are finitely many functions, we use the 'union bound'
- even if there are infinitely many, then *on any finite sample* there are effectively only finitely many (use *symmetrization* and *capacity concepts*)

## The Case of Two Functions

---

Suppose  $\mathcal{F} = \{f_1, f_2\}$ . Rewrite

$$\mathbb{P}\left\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\right\} = \mathbb{P}(C_\epsilon^1 \cup C_\epsilon^2),$$

where

$$C_\epsilon^i := \{(x_1, y_1), \dots, (x_m, y_m) \mid (R[f_i] - R_{\text{emp}}[f_i]) > \epsilon\}$$

denotes the event that the risks of  $f_i$  differ by more than  $\epsilon$ .

The RHS equals

$$\begin{aligned} \mathbb{P}(C_\epsilon^1 \cup C_\epsilon^2) &= \mathbb{P}(C_\epsilon^1) + \mathbb{P}(C_\epsilon^2) - \mathbb{P}(C_\epsilon^1 \cap C_\epsilon^2) \\ &\leq \mathbb{P}(C_\epsilon^1) + \mathbb{P}(C_\epsilon^2). \end{aligned}$$

Hence by Chernoff's bound

$$\begin{aligned} \mathbb{P}\left\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\right\} &\leq \mathbb{P}(C_\epsilon^1) + \mathbb{P}(C_\epsilon^2) \\ &\leq 2 \cdot 2 \exp(-2m\epsilon^2). \end{aligned}$$

## The Union Bound

---

Similarly, if  $\mathcal{F} = \{f_1, \dots, f_n\}$ , we have

$$\mathbb{P}\left\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\right\} = \mathbb{P}(C_\epsilon^1 \cup \dots \cup C_\epsilon^n),$$

and

$$\mathbb{P}(C_\epsilon^1 \cup \dots \cup C_\epsilon^n) \leq \sum_{i=1}^n \mathbb{P}(C_\epsilon^i).$$

Use Chernoff for each summand, to get an extra factor  $n$  in the bound.

Note: this becomes an equality if and only if all the events  $C_\epsilon^i$  involved are *disjoint*.

## Infinite Function Classes

---

- Note: empirical risk only refers to  $m$  points. On these points, the functions of  $\mathcal{F}$  can take at most  $2^m$  values
- for  $R_{\text{emp}}$ , the function class thus “looks” finite
- how about  $R$ ?
- need to use a trick

## Symmetrization

---

**Lemma 1 (Vapnik & Chervonenkis (e.g., [69, 20]))**

*For  $m\epsilon^2 \geq 2$  we have*

$$P\left\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\right\} \leq 2P\left\{\sup_{f \in \mathcal{F}} (R_{\text{emp}}[f] - R'_{\text{emp}}[f]) > \epsilon/2\right\}$$

*Here, the first  $P$  refers to the distribution of iid samples of size  $m$ , while the second one refers to iid samples of size  $2m$ . In the latter case,  $R_{\text{emp}}$  measures the loss on the first half of the sample, and  $R'_{\text{emp}}$  on the second half.*

## Shattering Coefficient

---

- Hence, we only need to consider the maximum size of  $\mathcal{F}$  on  $2m$  points. Call it  $\mathcal{N}(\mathcal{F}, 2m)$ .
- $\mathcal{N}(\mathcal{F}, 2m) = \max.$  number of different outputs  $(y_1, \dots, y_{2m})$  that the function class can generate on  $2m$  points — in other words, the max. number of different ways the function class can separate  $2m$  points into two classes.
- $\mathcal{N}(\mathcal{F}, 2m) \leq 2^{2m}$
- if  $\mathcal{N}(\mathcal{F}, 2m) = 2^{2m}$ , then the function class is said to *shatter*  $2m$  points.

## Putting Everything Together

---

We now use (1) symmetrization, (2) the shattering coefficient, and (3) the union bound, to get

$$\begin{aligned} & \mathbb{P}\left\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\right\} \\ & \leq 2\mathbb{P}\left\{\sup_{f \in \mathcal{F}} (R_{\text{emp}}[f] - R'_{\text{emp}}[f]) > \epsilon/2\right\} \\ & = 2\mathbb{P}\left\{(R_{\text{emp}}[f_1] - R'_{\text{emp}}[f_1]) > \epsilon/2 \vee \dots \vee (R_{\text{emp}}[f_{\mathcal{N}(\mathcal{F}, 2m)}] - R'_{\text{emp}}[f_{\mathcal{N}(\mathcal{F}, 2m)}]) > \epsilon/2\right\} \\ & \leq \sum_{n=1}^{\mathcal{N}(\mathcal{F}, 2m)} 2\mathbb{P}\left\{(R_{\text{emp}}[f_n] - R'_{\text{emp}}[f_n]) > \epsilon/2\right\}. \end{aligned}$$

ctd.

---

Use Chernoff's bound for each term:\*

$$\mathbb{P} \left\{ \frac{1}{m} \sum_{i=1}^m \xi_i - \frac{1}{m} \sum_{i=m+1}^{2m} \xi_i \geq \epsilon \right\} \leq 2 \exp \left( -\frac{m\epsilon^2}{2} \right).$$

This yields

$$\mathbb{P} \left\{ \sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon \right\} \leq 4\mathcal{N}(\mathcal{F}, 2m) \exp \left( -\frac{m\epsilon^2}{8} \right).$$

- provided that  $\mathcal{N}(\mathcal{F}, 2m)$  does not grow exponentially in  $m$ , this is nontrivial
- such bounds are called *VC type inequalities*
- two types of randomness: (1) the  $\mathbb{P}$  refers to the drawing of the training examples, and (2)  $R[f]$  is an expectation over the drawing of test examples.

\* A rigorous treatment would need to use a second randomization over permutations of the  $2m$ -sample, see [56].



## Confidence Intervals

---

Rewrite the bound: specify the probability with which we want  $R$  to be close to  $R_{\text{emp}}$ , and solve for  $\epsilon$ :

With a probability of at least  $1 - \delta$ ,

$$R[f] \leq R_{\text{emp}}[f] + \sqrt{\frac{8}{m} \left( \ln(\mathcal{N}(\mathcal{F}, 2m)) + \ln \frac{4}{\delta} \right)}.$$

This bound holds independent of  $f$ ; in particular, it holds for the function  $f^m$  minimizing the empirical risk.

## Discussion

---

- tighter bounds are available (better constants etc.)
- cannot minimize the bound over  $f$
- other capacity concepts can be used

## VC Entropy

---

On an example  $(\mathbf{x}, y)$ ,  $f$  causes a loss

$$\xi(x, y, f(x)) = \frac{1}{2}|f(x) - y| \in \{0, 1\}.$$

For a larger sample  $(x_1, y_1), \dots, (x_m, y_m)$ , the different functions  $f \in \mathcal{F}$  lead to a *set* of loss vectors

$$\boldsymbol{\xi}_f = (\xi(x_1, y_1, f(x_1)), \dots, \xi(x_m, y_m, f(x_m))),$$

whose cardinality we denote by

$$\mathcal{N}(\mathcal{F}, (x_1, y_1) \dots, (x_m, y_m)).$$

The *VC entropy* is defined as

$$H_{\mathcal{F}}(m) = \mathbf{E} [\ln \mathcal{N}(\mathcal{F}, (x_1, y_1) \dots, (x_m, y_m))],$$

where the expectation is taken over the random generation of the  $m$ -sample  $(x_1, y_1) \dots, (x_m, y_m)$  from  $P$ .

$H_{\mathcal{F}}(m)/m \rightarrow 0 \iff$  uniform convergence of risks (hence consistency)

## Further PR Capacity Concepts

---

- exchange 'E' and 'ln': *annealed entropy*.

$H_{\mathcal{F}}^{\text{ann}}(m)/m \rightarrow 0 \iff$  exponentially fast uniform convergence

- take 'max' instead of 'E': *growth function*.

Note that  $G_{\mathcal{F}}(m) = \ln \mathcal{N}(\mathcal{F}, m)$ .

$G_{\mathcal{F}}(m)/m \rightarrow 0 \iff$  exponential convergence for all underlying distributions P.

$G_{\mathcal{F}}(m) = m \cdot \ln(2)$  for all  $m \iff$  for any  $m$ , all loss vectors can be generated, i.e., the  $m$  points can be chosen such that by using functions of the learning machine, they can be separated in all  $2^m$  possible ways (*shattered*).

## Structure of the Growth Function

---

**Either**  $G_{\mathcal{F}}(m) = m \cdot \ln(2)$  for all  $m \in \mathbb{N}$

**Or** there exists some *maximal*  $m$  for which the above is possible. Call this number the *VC-dimension*, and denote it by  $h$ . For  $m > h$ ,

$$G_{\mathcal{F}}(m) \leq h \left( \ln \frac{m}{h} + 1 \right).$$

Nothing “in between” linear growth and logarithmic growth is possible.

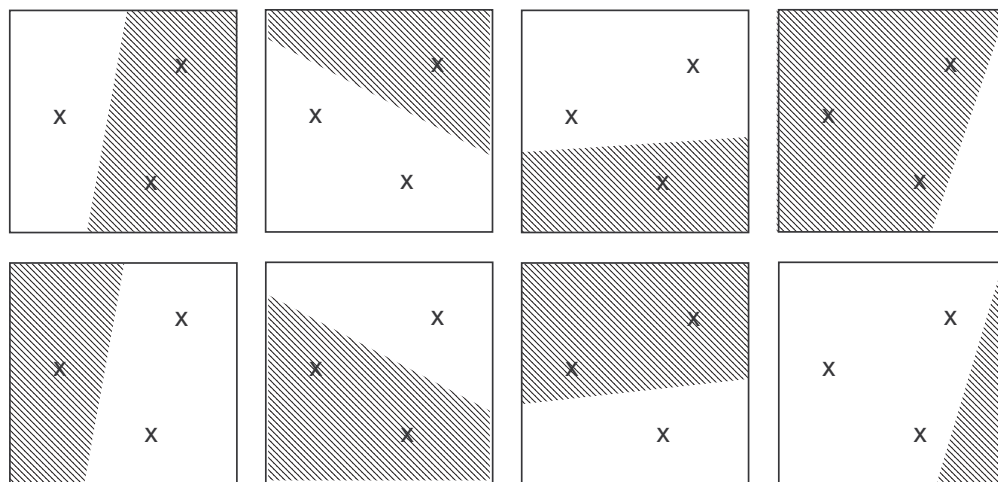
## VC-Dimension: Example

---

Half-spaces in  $\mathbb{R}^2$ :

$$f(x, y) = \text{sgn}(a + bx + cy), \quad \text{with parameters } a, b, c \in \mathbb{R}$$

- Clearly, we can shatter three non-collinear points.
- But we can never shatter four points.
- Hence the VC dimension is  $h = 3$  (in this case, equal to the number of parameters)



## A Typical Bound for Pattern Recognition

---

For any  $f \in \mathcal{F}$  and  $m > h$ , with a probability of at least  $1 - \delta$ ,

$$R[f] \leq R_{\text{emp}}[f] + \phi \left( \frac{h}{m}, \frac{\log(\delta)}{m} \right)$$

holds, where the *confidence term*  $\phi$  is defined as

$$\phi \left( \frac{h}{m}, \frac{\log(\delta)}{m} \right) = \sqrt{\frac{h \left( \log \frac{2m}{h} + 1 \right) - \log(\delta/4)}{m}}.$$

- does this mean, that we can learn *anything*?
- The study of the consistency of ERM has thus led to concepts and results which lets us formulate a better induction principle: we can use this bound to get a low risk!
- in practice: use as a guideline for designing algorithms

## Examples of Induction Principles

---

- *Empirical risk minimization*: minimize

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

- *Minimum description length*: minimize some measure of the description length of the sequence  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$  by a function  $f$ .
- *Structural risk minimization (SRM) (Vapnik, 1979)*: minimize the RHS of

$$R[f] \leq R_{\text{emp}}[f] + \phi \left( \frac{h}{m} \right).$$

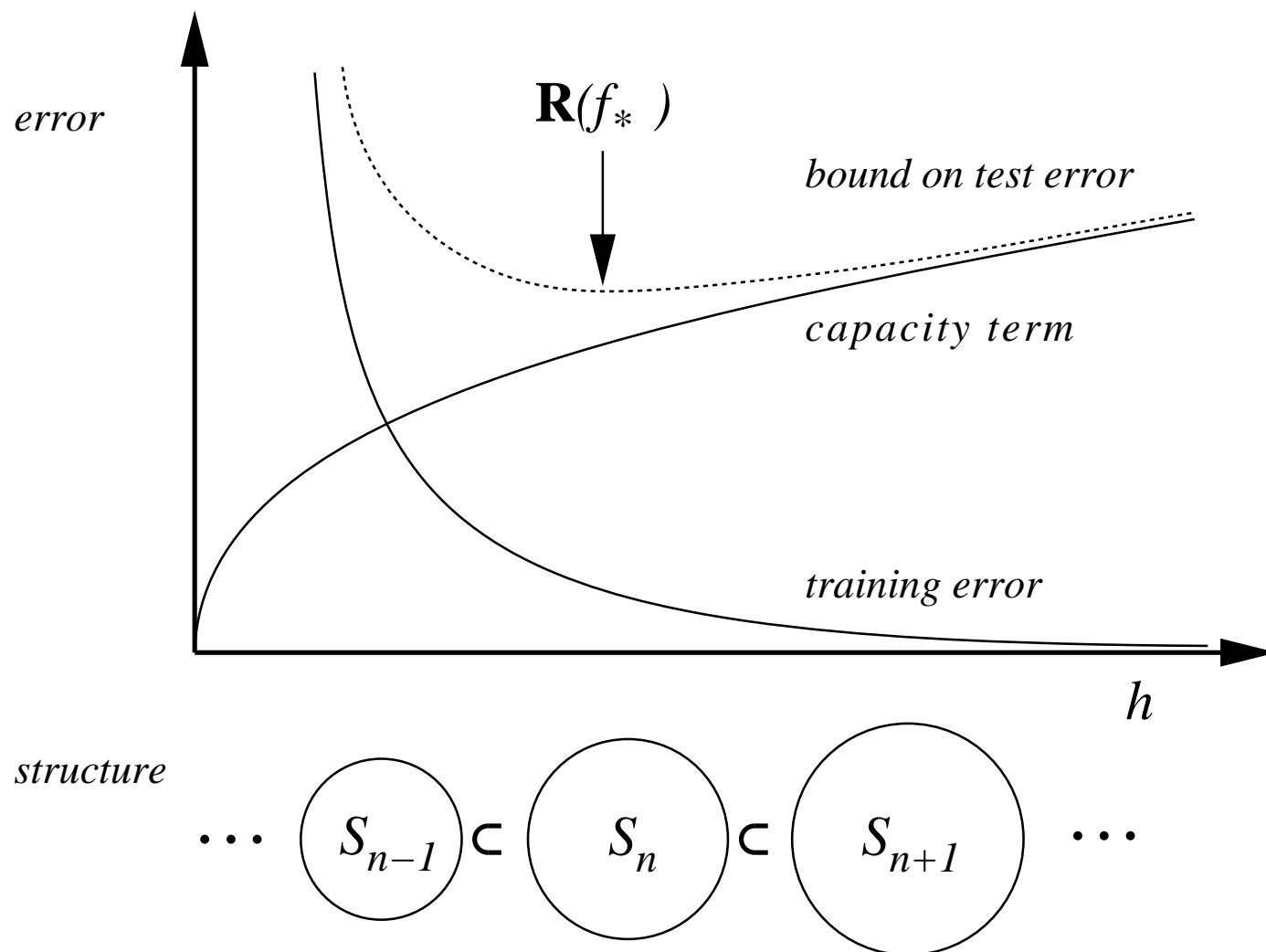
To this end, introduce a structure on  $\mathcal{F}$ .

Learning machine  $\equiv$  a set of functions and an induction principle



# SRM: The Picture

---



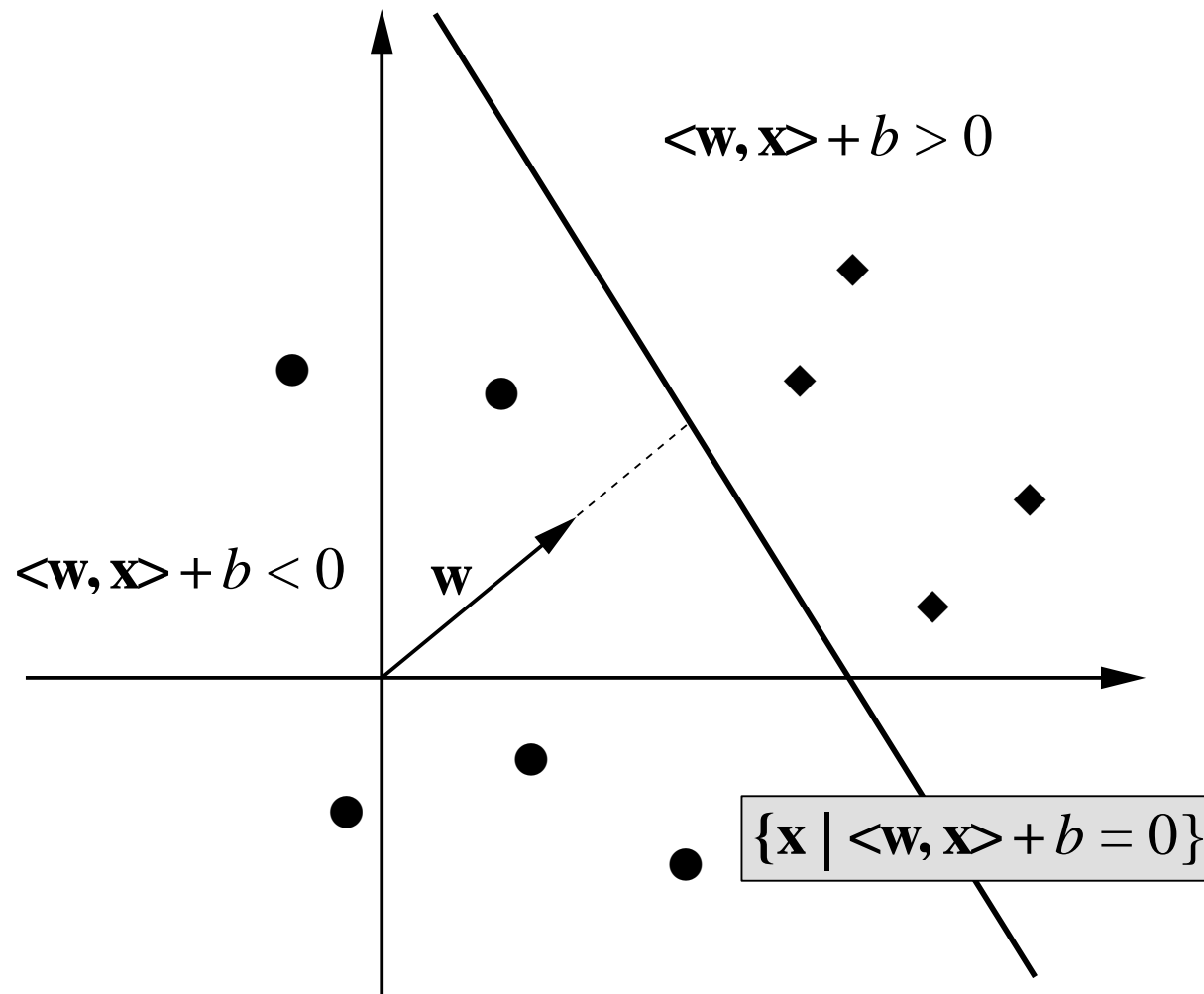
## Finding a Good Function Class

---

- recall: separating hyperplanes in  $\mathbb{R}^2$  have a VC dimension of 3.
- more generally: separating hyperplanes in  $\mathbb{R}^N$  have a VC dimension of  $N + 1$ .
- hence: separating hyperplanes in high-dimensional feature spaces have extremely large VC dimension, and may not generalize well
- however, *margin* hyperplanes can still have a small VC dimension

# Separating Hyperplane

---



# Canonical Hyperplanes

---

[72]

Note: if  $c \neq 0$ , then

$$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\} = \{\mathbf{x} \mid \langle c\mathbf{w}, \mathbf{x} \rangle + cb = 0\}.$$

Hence  $(c\mathbf{w}, cb)$  describes the same hyperplane as  $(\mathbf{w}, b)$ .

**Definition:** The hyperplane is in *canonical* form w.r.t.  $X^* = \{\mathbf{x}_1, \dots, \mathbf{x}_r\}$  if  $\min_{\mathbf{x}_i \in X} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1$ .

Note that for canonical hyperplanes, the distance of the closest point to the hyperplane (“margin”) is  $1/\|\mathbf{w}\|$ :

$$\min_{\mathbf{x}_i \in X} \left| \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{x}_i \right\rangle + \frac{b}{\|\mathbf{w}\|} \right| = \frac{1}{\|\mathbf{w}\|}.$$

---

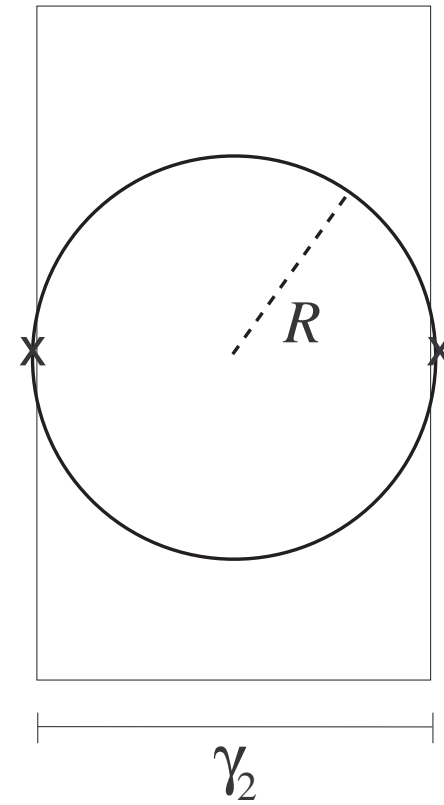
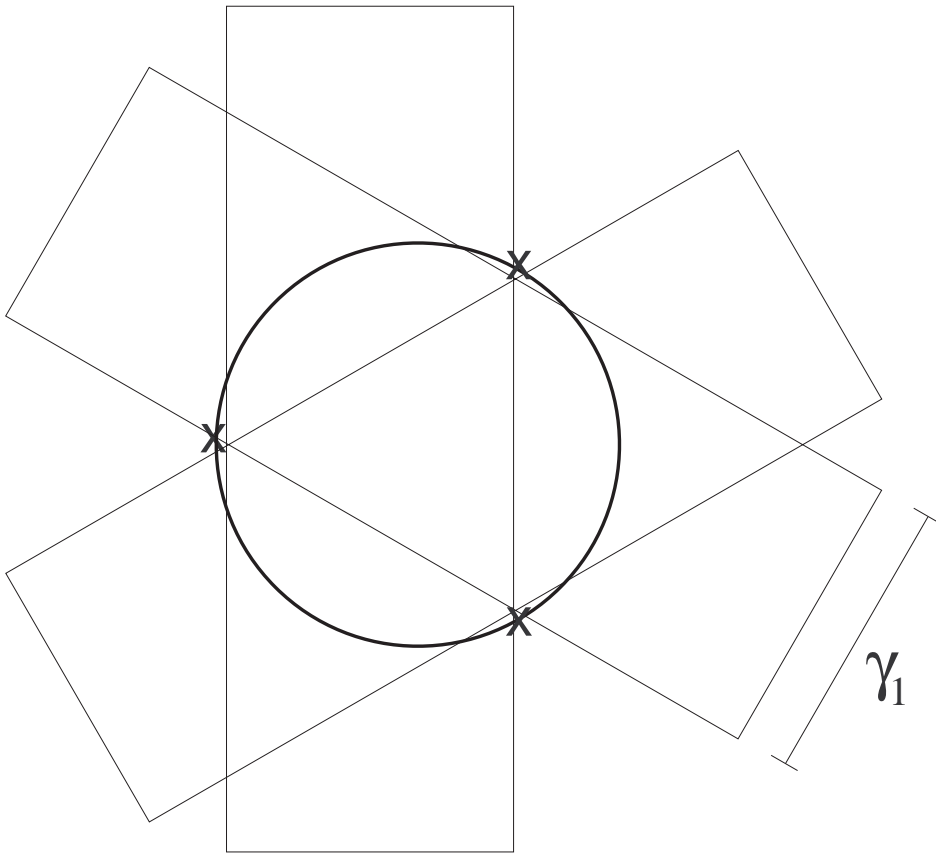
**Theorem 2 (Vapnik [69])** Consider hyperplanes  $\langle \mathbf{w}, \mathbf{x} \rangle = 0$  where  $\mathbf{w}$  is normalized such that they are in canonical form w.r.t. a set of points  $X^* = \{\mathbf{x}_1, \dots, \mathbf{x}_r\}$ , i.e.,

$$\min_{i=1, \dots, r} |\langle \mathbf{w}, \mathbf{x}_i \rangle| = 1.$$

The set of decision functions  $f_{\mathbf{w}}(\mathbf{x}) = \text{sgn} \langle \mathbf{x}, \mathbf{w} \rangle$  defined on  $X^*$  and satisfying the constraint  $\|\mathbf{w}\| \leq \Lambda$  has a VC dimension satisfying

$$h \leq R^2 \Lambda^2.$$

Here,  $R$  is the radius of the smallest sphere around the origin containing  $X^*$ .



## Proof Strategy (Gurvits, 1997)

---

Assume that  $\mathbf{x}_1, \dots, \mathbf{x}_r$  are shattered by canonical hyperplanes with  $\|\mathbf{w}\| \leq \Lambda$ , i.e., for arbitrary  $y_1, \dots, y_r \in \{\pm 1\}$ , there exists a  $\mathbf{w}$  such that

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 \quad \text{for all } i = 1, \dots, r. \quad (1)$$

Two steps:

- prove that the more points we want to shatter (1), the larger  $\|\sum_{i=1}^r y_i \mathbf{x}_i\|$  must be
- upper bound the size of  $\|\sum_{i=1}^r y_i \mathbf{x}_i\|$  in terms of  $R$

Combining the two tells us how many points we can at most shatter.

## Part I

---

Summing (1) over  $i = 1, \dots, r$  yields

$$\left\langle \mathbf{w}, \left( \sum_{i=1}^r y_i \mathbf{x}_i \right) \right\rangle \geq r.$$

By the Cauchy-Schwarz inequality, on the other hand, we have

$$\left\langle \mathbf{w}, \left( \sum_{i=1}^r y_i \mathbf{x}_i \right) \right\rangle \leq \|\mathbf{w}\| \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\| \leq \Lambda \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|.$$

Combine both:

$$\frac{r}{\Lambda} \leq \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|. \quad (2)$$



## Part II

---

Consider independent random labels  $y_i \in \{\pm 1\}$ , uniformly distributed (*Rademacher variables*).

$$\begin{aligned} \mathbf{E} \left[ \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|^2 \right] &= \sum_{i=1}^r \mathbf{E} \left[ \left\langle y_i \mathbf{x}_i, \sum_{j=1}^r y_j \mathbf{x}_j \right\rangle \right] \\ &= \sum_{i=1}^r \mathbf{E} \left[ \left\langle y_i \mathbf{x}_i, \left( \left( \sum_{j \neq i} y_j \mathbf{x}_j \right) + y_i \mathbf{x}_i \right) \right\rangle \right] \\ &= \sum_{i=1}^r \left( \left( \sum_{j \neq i} \mathbf{E} [\langle y_i \mathbf{x}_i, y_j \mathbf{x}_j \rangle] \right) + \mathbf{E} [\langle y_i \mathbf{x}_i, y_i \mathbf{x}_i \rangle] \right) \\ &= \sum_{i=1}^r \mathbf{E} [\|y_i \mathbf{x}_i\|^2] = \sum_{i=1}^r \|\mathbf{x}_i\|^2 \end{aligned}$$

## Part II, ctd.

---

Since  $\|\mathbf{x}_i\| \leq R$ , we get

$$\mathbf{E} \left[ \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|^2 \right] \leq rR^2.$$

- This holds for the *expectation* over the random choices of the labels, hence there must be at least one set of labels for which it also holds true. Use this set.

Hence

$$\left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|^2 \leq rR^2.$$

## Part I and II Combined

---

$$\text{Part I: } \left(\frac{r}{\Lambda}\right)^2 \leq \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|^2$$

$$\text{Part II: } \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|^2 \leq rR^2$$

Hence

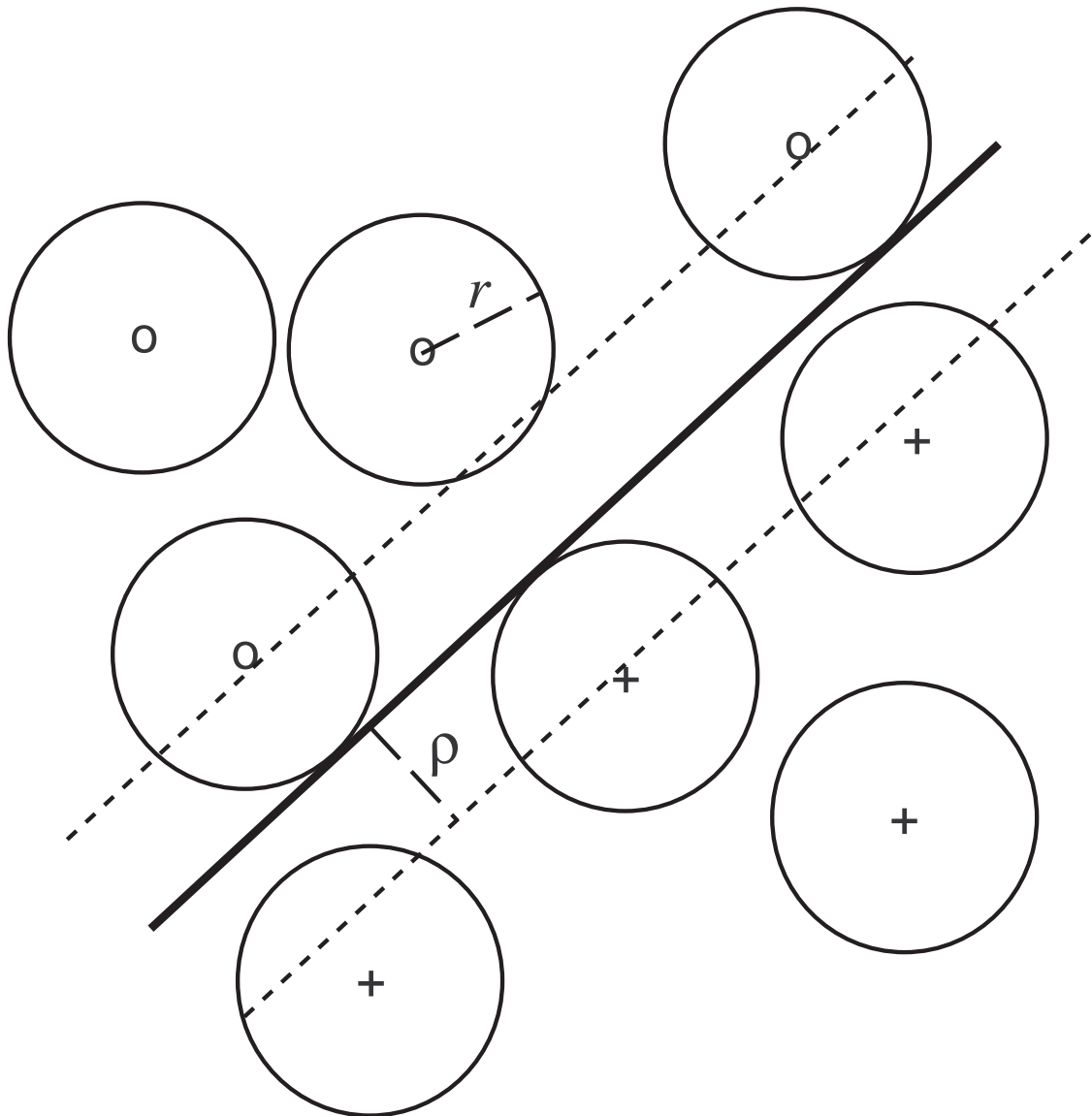
$$\frac{r^2}{\Lambda^2} \leq rR^2,$$

i.e.,

$$r \leq R^2 \Lambda^2.$$

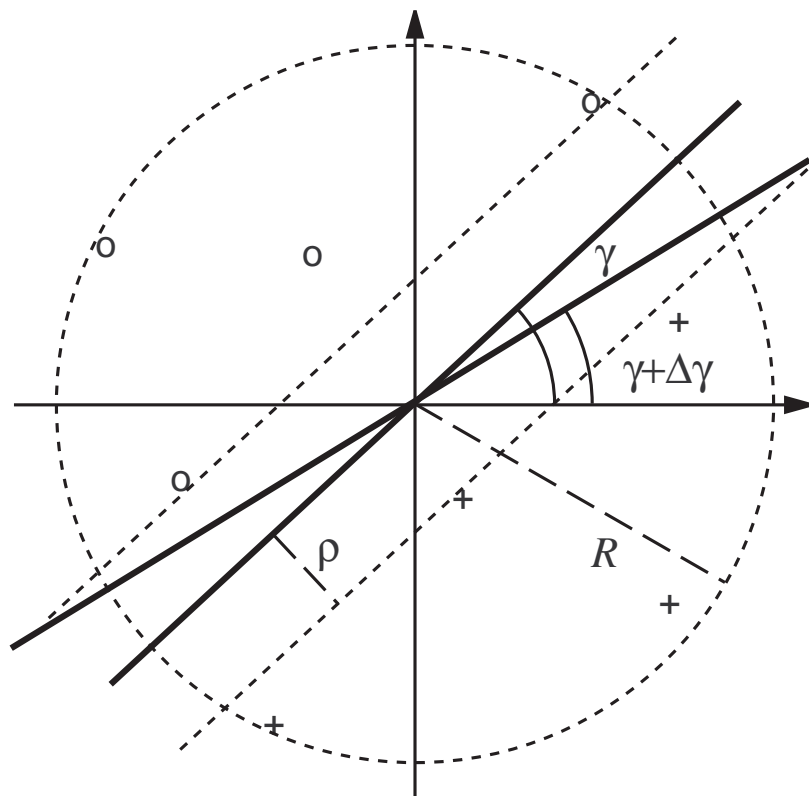
# Pattern Noise as Maximum Margin Regularization

---



## Maximum Margin vs. MDL — 2D Case

---



Can perturb  $\gamma$  by  $\Delta\gamma$  with  $|\Delta\gamma| < \arcsin \frac{\rho}{R}$  and still correctly separate the data.

Hence only need to store  $\gamma$  with accuracy  $\Delta\gamma$  [56, 75].

# Kernels and Feature Spaces

---

Preprocess the data with

$$\begin{aligned}\Phi : \mathcal{X} &\rightarrow \mathcal{H} \\ x &\mapsto \Phi(x),\end{aligned}$$

where  $\mathcal{H}$  is a dot product space, and learn the mapping from  $\Phi(x)$  to  $y$ .

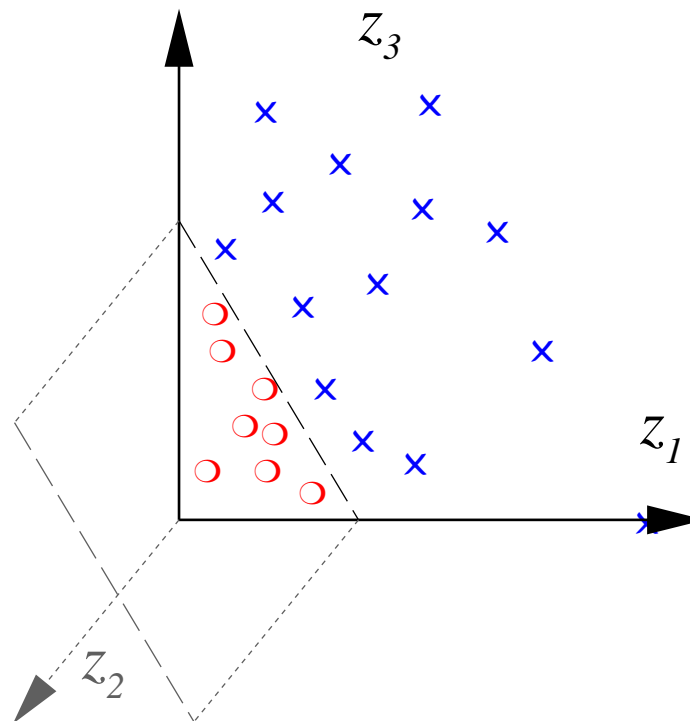
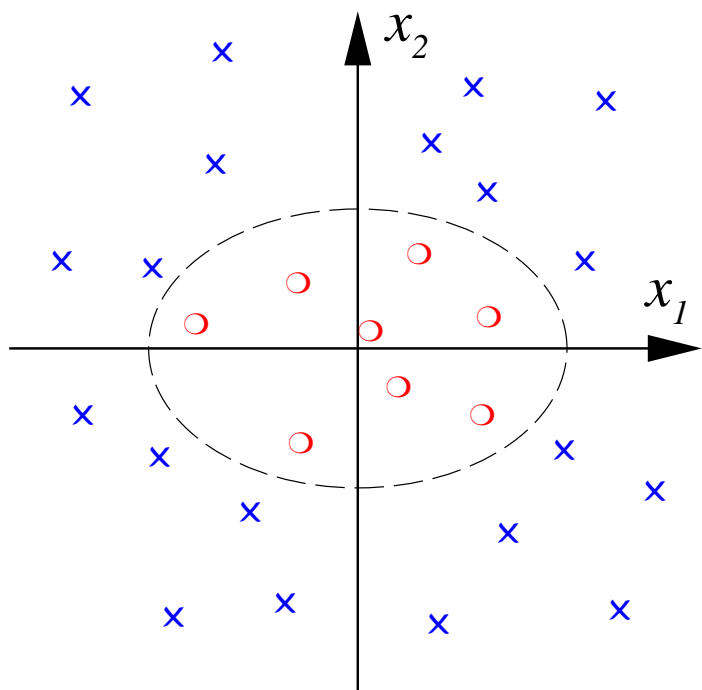
- usually,  $\dim(\mathcal{X}) \ll \dim(\mathcal{H})$
- “Curse of Dimensionality”?
- crucial issue: *capacity*, not *dimensionality*

## Example: All Degree 2 Monomials

---

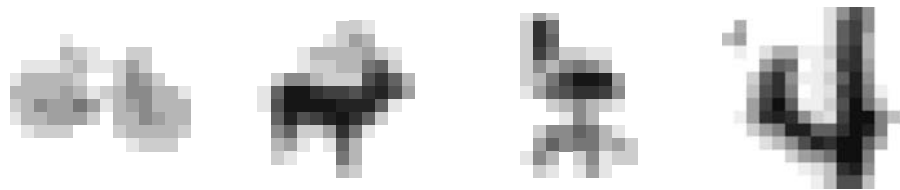
$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

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



## General Product Feature Space

---



How about patterns  $x \in \mathbb{R}^N$  and product features of order  $d$ ?

Here,  $\dim(\mathcal{H})$  grows like  $N^d$ .

E.g.  $N = 16 \times 16$ , and  $d = 5 \longrightarrow$  dimension  $10^{10}$



## The Kernel Trick, $N = d = 2$

---

$$\begin{aligned}\langle \Phi(x), \Phi(x') \rangle &= (x_1^2, \sqrt{2} x_1 x_2, x_2^2) (x_1'^2, \sqrt{2} x_1' x_2', x_2'^2)^\top \\ &= \langle x, x' \rangle^2 \\ &=: k(x, x')\end{aligned}$$

→ the dot product in  $\mathcal{H}$  can be computed in  $\mathbb{R}^2$

## The Kernel Trick, II

---

More generally:  $x, x' \in \mathbb{R}^N$ ,  $d \in \mathbb{N}$ :

$$\begin{aligned}\langle x, x' \rangle^d &= \left( \sum_{j=1}^N x_j \cdot x'_j \right)^d \\ &= \sum_{j_1, \dots, j_d=1}^N x_{j_1} \cdots x_{j_d} \cdot x'_{j_1} \cdots x'_{j_d} = \langle \Phi(x), \Phi(x') \rangle,\end{aligned}$$

where  $\Phi$  maps into the space spanned by all ordered products of  $d$  input directions

## Mercer's Theorem

---

If  $k$  is a continuous kernel of a positive definite integral operator on  $L_2(\mathcal{X})$  (where  $\mathcal{X}$  is some compact space),

$$\int_{\mathcal{X}} k(x, x') f(x) f(x') dx dx' \geq 0,$$

it can be expanded as

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x')$$

using eigenfunctions  $\psi_i$  and eigenvalues  $\lambda_i \geq 0$  [42].

# The Mercer Feature Map

---

In that case

$$\Phi(x) := \begin{pmatrix} \sqrt{\lambda_1}\psi_1(x) \\ \sqrt{\lambda_2}\psi_2(x) \\ \vdots \end{pmatrix}$$

satisfies  $\langle \Phi(x), \Phi(x') \rangle = k(x, x')$ .

Proof:

$$\begin{aligned} \langle \Phi(x), \Phi(x') \rangle &= \left\langle \begin{pmatrix} \sqrt{\lambda_1}\psi_1(x) \\ \sqrt{\lambda_2}\psi_2(x) \\ \vdots \end{pmatrix}, \begin{pmatrix} \sqrt{\lambda_1}\psi_1(x') \\ \sqrt{\lambda_2}\psi_2(x') \\ \vdots \end{pmatrix} \right\rangle \\ &= \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x') = k(x, x') \end{aligned}$$

## The Kernel Trick — Summary

---

- *any* algorithm that only depends on dot products can benefit from the kernel trick
- this way, we can apply linear methods to vectorial as well as *non-vectorial data*
- think of the kernel as a nonlinear *similarity measure*
- examples of common kernels:

Polynomial  $k(x, x') = (\langle x, x' \rangle + c)^d$

Sigmoid  $k(x, x') = \tanh(\kappa \langle x, x' \rangle + \Theta)$

Gaussian  $k(x, x') = \exp(-\|x - x'\|^2 / (2\sigma^2))$

- Kernels are studied also in the Gaussian Process prediction community (covariance functions) [79, 76, 81, 41]

## Positive Definite Kernels

---

It can be shown that (modulo some details) the admissible class of kernels coincides with the one of **positive definite (pd) kernels**: kernels which are symmetric (i.e.,  $k(x, x') = k(x', x)$ ), and for

- any set of training points  $x_1, \dots, x_m \in \mathcal{X}$  and
- any  $a_1, \dots, a_m \in \mathbb{R}$

satisfy

$$\sum_{i,j} a_i a_j K_{ij} \geq 0, \quad \text{where } K_{ij} := k(x_i, x_j).$$

$K$  is called the *Gram matrix* or *kernel matrix*.

## Elementary Properties of PD Kernels

---

*Kernels from Feature Maps.*

If  $\Phi$  maps  $\mathcal{X}$  into a dot product space  $\mathcal{H}$ , then  $\langle \Phi(x), \Phi(x') \rangle$  is a pd kernel on  $\mathcal{X} \times \mathcal{X}$ .

*Positivity on the Diagonal.*

$k(x, x) \geq 0$  for all  $x \in \mathcal{X}$

*Cauchy-Schwarz Inequality.*

$k(x, x')^2 \leq k(x, x)k(x', x')$  (Hint: compute the determinant of the Gram matrix)

*Vanishing Diagonals.*

$k(x, x) = 0$  for all  $x \in \mathcal{X} \implies k(x, x') = 0$  for all  $x, x' \in \mathcal{X}$

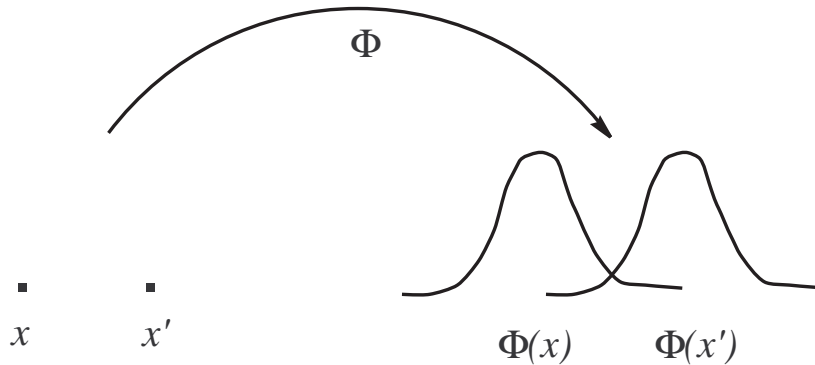
# The Feature Space for PD Kernels

[4, 1, 50]

- define a feature map

$$\begin{aligned}\Phi : \mathcal{X} &\rightarrow \mathbb{R}^{\mathcal{X}} \\ x &\mapsto k(\cdot, x).\end{aligned}$$

E.g., for the Gaussian kernel:



Next steps:

- turn  $\Phi(\mathcal{X})$  into a linear space
- endow it with a dot product satisfying  $\langle \Phi(x), \Phi(x') \rangle = k(x, x')$ , i.e.,  $\langle k(\cdot, x), k(\cdot, x') \rangle = k(x, x')$
- complete the space to get a *reproducing kernel Hilbert space*



## Turn it Into a Linear Space

---

Form linear combinations

$$f(\cdot) = \sum_{i=1}^m \alpha_i k(\cdot, x_i),$$

$$g(\cdot) = \sum_{j=1}^{m'} \beta_j k(\cdot, x'_j)$$

$(m, m' \in \mathbb{N}, \alpha_i, \beta_j \in \mathbb{R}, x_i, x'_j \in \mathcal{X})$ .

## Endow it With a Dot Product

---

$$\begin{aligned}\langle f, g \rangle &:= \sum_{i=1}^m \sum_{j=1}^{m'} \alpha_i \beta_j k(x_i, x'_j) \\ &= \sum_{i=1}^m \alpha_i g(x_i) = \sum_{j=1}^{m'} \beta_j f(x'_j)\end{aligned}$$

- This is well-defined, symmetric, and bilinear (more later).

# The Reproducing Kernel Property

---

Two special cases:

- Assume

$$f(\cdot) = k(\cdot, x).$$

In this case, we have

$$\langle k(\cdot, x), g \rangle = g(x).$$

- If moreover

$$g(\cdot) = k(\cdot, x'),$$

we have

$$\langle k(\cdot, x), k(\cdot, x') \rangle = k(x, x').$$

$k$  is called a *reproducing kernel*

## Endow it With a Dot Product, II

---

- It can be shown that  $\langle \cdot, \cdot \rangle$  is a p.d. kernel on the set of functions  $\{f(\cdot) = \sum_{i=1}^m \alpha_i k(\cdot, x_i) \mid \alpha_i \in \mathbb{R}, x_i \in \mathcal{X}\}$  :

$$\begin{aligned} \sum_{ij} \gamma_i \gamma_j \langle f_i, f_j \rangle &= \left\langle \sum_i \gamma_i f_i, \sum_j \gamma_j f_j \right\rangle =: \langle f, f \rangle \\ &= \left\langle \sum_i \alpha_i k(\cdot, x_i), \sum_i \alpha_i k(\cdot, x_i) \right\rangle = \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) \geq 0 \end{aligned}$$

- furthermore, it is *strictly* positive definite:

$$f(x)^2 = \langle f, k(\cdot, x) \rangle^2 \leq \langle f, f \rangle \langle k(\cdot, x), k(\cdot, x) \rangle = \langle f, f \rangle k(x, x)$$

hence  $\langle f, f \rangle = 0$  implies  $f = 0$ .

- Complete the space in the corresponding norm to get a Hilbert space  $\mathcal{H}_k$ .

# Explicit Construction of the RKHS Map for Mercer Kernels

---

Recall that the dot product has to satisfy

$$\langle k(x, \cdot), k(x', \cdot) \rangle = k(x, x').$$

For a Mercer kernel

$$k(x, x') = \sum_{j=1}^{N_F} \lambda_j \psi_j(x) \psi_j(x')$$

(with  $\lambda_i > 0$  for all  $i$ ,  $N_F \in \mathbb{N} \cup \{\infty\}$ , and  $\langle \psi_i, \psi_j \rangle_{L_2(\mathcal{X})} = \delta_{ij}$ ), this can be achieved by choosing  $\langle \cdot, \cdot \rangle$  such that

$$\langle \psi_i, \psi_j \rangle = \delta_{ij} / \lambda_i.$$

ctd.

---

To see this, compute

$$\begin{aligned}\langle k(x, \cdot), k(x', \cdot) \rangle &= \left\langle \sum_i \lambda_i \psi_i(x) \psi_i, \sum_j \lambda_j \psi_j(x') \psi_j \right\rangle \\ &= \sum_{i,j} \lambda_i \lambda_j \psi_i(x) \psi_j(x') \langle \psi_i, \psi_j \rangle \\ &= \sum_{i,j} \lambda_i \lambda_j \psi_i(x) \psi_j(x') \delta_{ij} / \lambda_i \\ &= \sum_i \lambda_i \psi_i(x) \psi_i(x') \\ &= k(x, x').\end{aligned}$$

## Deriving the Kernel from the RKHS

---

An RKHS is a Hilbert space  $\mathcal{H}$  of functions  $f$  where all *point evaluation functionals*

$$\begin{aligned} p_x: \mathcal{H} &\rightarrow \mathbb{R} \\ f &\mapsto p_x(f) = f(x) \end{aligned}$$

exist and are continuous.

*Continuity* means that whenever  $f$  and  $f'$  are close in  $\mathcal{H}$ , then  $f(x)$  and  $f'(x)$  are close in  $\mathbb{R}$ . This can be thought of as a topological prerequisite for generalization ability.

By Riesz' representation theorem, there exists an element of  $\mathcal{H}$ , call it  $r_x$ , such that

$$\langle r_x, f \rangle = f(x),$$

in particular,

$$\langle r_x, r_{x'} \rangle = r_{x'}(x).$$

Define  $k(x, x') := r_x(x') = r_{x'}(x)$ .

(cf. Canu & Mary, 2002)

# The Empirical Kernel Map

---

Recall the feature map

$$\begin{aligned}\Phi : \mathcal{X} &\rightarrow \mathbb{R}^{\mathcal{X}} \\ x &\mapsto k(\cdot, x).\end{aligned}$$

- each point is represented by its similarity to *all* other points
- how about representing it by its similarity to a *sample* of points?

Consider

$$\begin{aligned}\Phi_m : \mathcal{X} &\rightarrow \mathbb{R}^m \\ x &\mapsto k(\cdot, x)|_{(x_1, \dots, x_m)} = (k(x_1, x), \dots, k(x_m, x))^\top\end{aligned}$$



ctd.

---

- $\Phi_m(x_1), \dots, \Phi_m(x_m)$  contain *all* necessary information about  $\Phi(x_1), \dots, \Phi(x_m)$
- the Gram matrix  $G_{ij} := \langle \Phi_m(x_i), \Phi_m(x_j) \rangle$  satisfies  $G = K^2$  where  $K_{ij} = k(x_i, x_j)$
- modify  $\Phi_m$  to

$$\begin{aligned} \Phi_m^w : \mathcal{X} &\rightarrow \mathbb{R}^m \\ x &\mapsto K^{-\frac{1}{2}}(k(x_1, x), \dots, k(x_m, x))^\top \end{aligned}$$

- this “whitened” map (“kernel PCA map”) satisfies

$$\langle \Phi_m^w(x_i), \Phi_m^w(x_j) \rangle = k(x_i, x_j)$$

for all  $i, j = 1, \dots, m$ .

## Some Properties of Kernels [56]

---

If  $k_1, k_2, \dots$  are pd kernels, then so are

- $\alpha k_1$ , provided  $\alpha \geq 0$
- $k_1 + k_2$
- $k_1 \cdot k_2$
- $k(x, x') := \lim_{n \rightarrow \infty} k_n(x, x')$ , provided it exists
- $k(A, B) := \sum_{x \in A, x' \in B} k_1(x, x')$ , where  $A, B$  are finite subsets of  $\mathcal{X}$   
(using the feature map  $\tilde{\Phi}(A) := \sum_{x \in A} \Phi(x)$ )

Further operations to construct kernels from kernels: tensor products, direct sums, convolutions [30].

## Properties of Kernel Matrices, I [51]

---

Suppose we are given distinct training patterns  $x_1, \dots, x_m$ , and a positive definite  $m \times m$  matrix  $K$ .

$K$  can be diagonalized as  $K = SDS^\top$ , with an orthogonal matrix  $S$  and a diagonal matrix  $D$  with nonnegative entries. Then

$$K_{ij} = (SDS^\top)_{ij} = \langle S_i, DS_j \rangle = \langle \sqrt{D}S_i, \sqrt{D}S_j \rangle,$$

where the  $S_i$  are the rows of  $S$ .

We have thus constructed a map  $\Phi$  into an  $m$ -dimensional feature space  $\mathcal{H}$  such that

$$K_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle.$$

## Properties, II: Functional Calculus [60]

---

- $K$  symmetric  $m \times m$  matrix with spectrum  $\sigma(K)$
- $f$  a continuous function on  $\sigma(K)$
- Then there is a symmetric matrix  $f(K)$  with eigenvalues in  $f(\sigma(K))$ .
- compute  $f(K)$  via Taylor series, or eigenvalue decomposition of  $K$ : If  $K = S^\top D S$  ( $D$  diagonal and  $S$  unitary), then  $f(K) = S^\top f(D) S$ , where  $f(D)$  is defined elementwise on the diagonal
- can treat functions of symmetric matrices like functions on  $\mathbb{R}$

$$(\alpha f + g)(K) = \alpha f(K) + g(K)$$

$$(fg)(K) = f(K)g(K) = g(K)f(K)$$

$$\|f\|_{\infty, \sigma(K)} = \|f(K)\|$$

$$\sigma(f(K)) = f(\sigma(K))$$

(the  $C^*$ -algebra generated by  $K$  is isomorphic to the set of continuous functions on  $\sigma(K)$ )

## Computing Distances in Feature Spaces

---

Clearly, if  $k$  is positive definite, then there exists a map  $\Phi$  such that

$$\|\Phi(x) - \Phi(x')\|^2 = k(x, x) + k(x', x') - 2k(x, x')$$

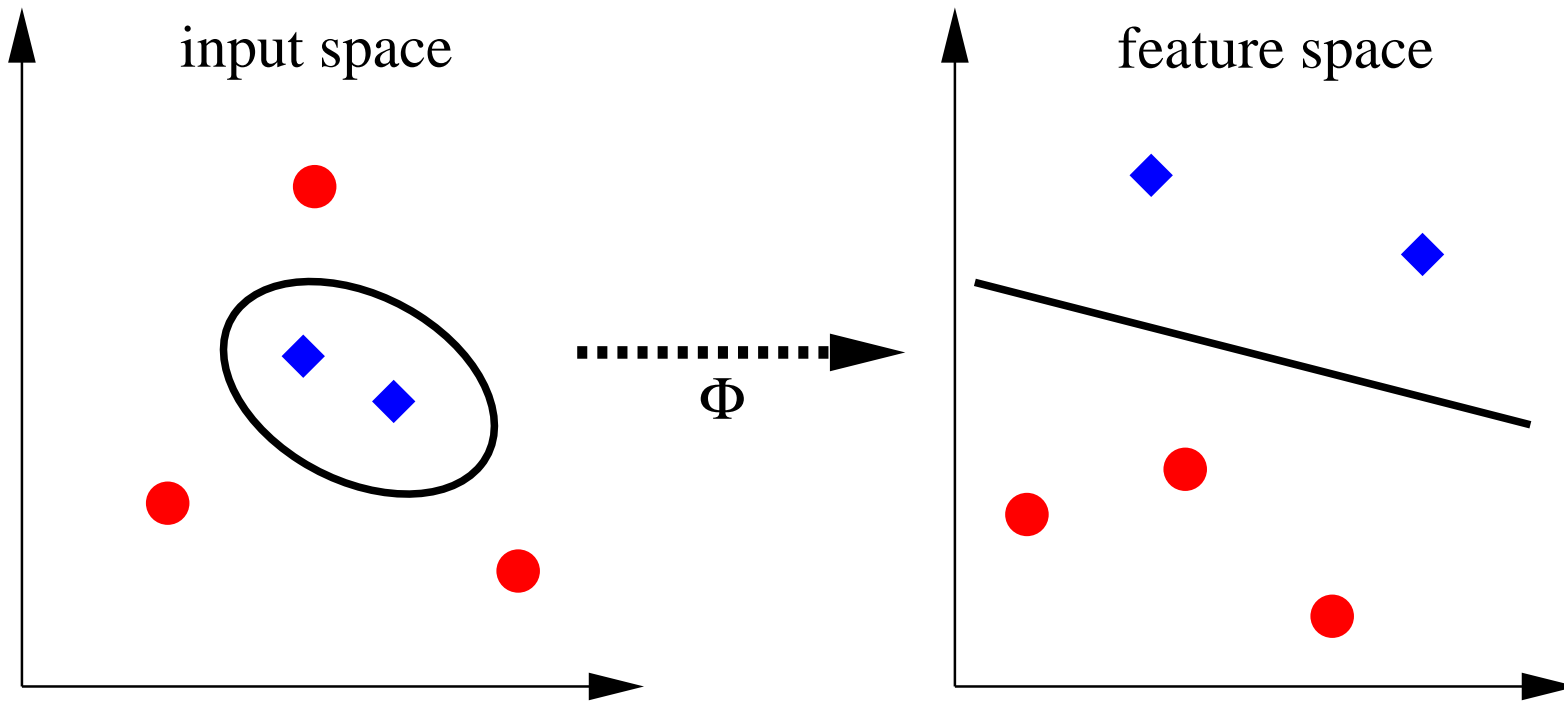
(it is the usual feature map).

This embedding is referred to as a *Hilbert space representation* as a distance. It turns out that this works for a larger class of kernels, called *conditionally positive definite*.

In fact, all algorithms that are translationally invariant (i.e. independent of the choice of the origin) in  $\mathcal{H}$  work with cpd kernels [56].

# Support Vector Classifiers

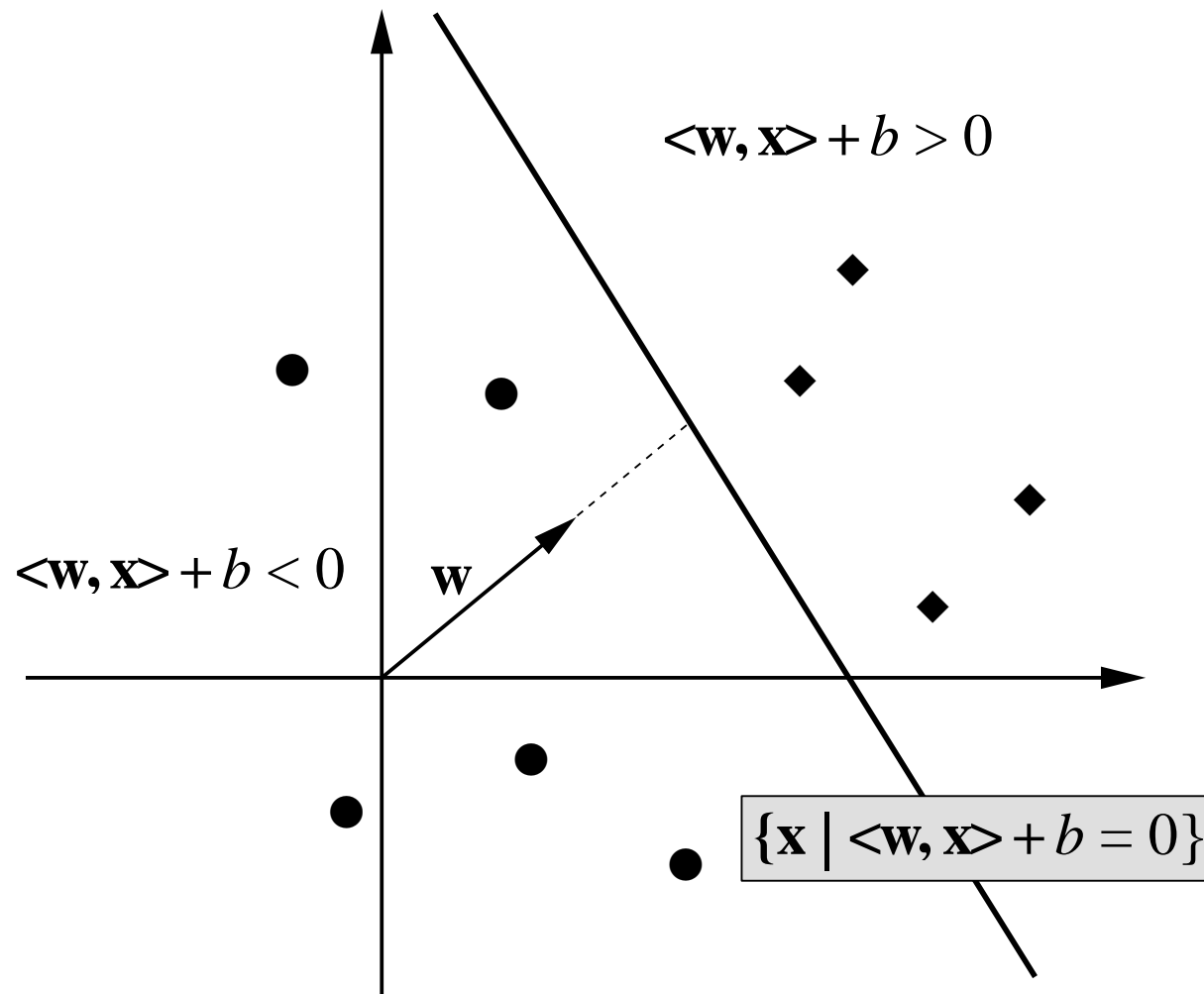
---



[6]

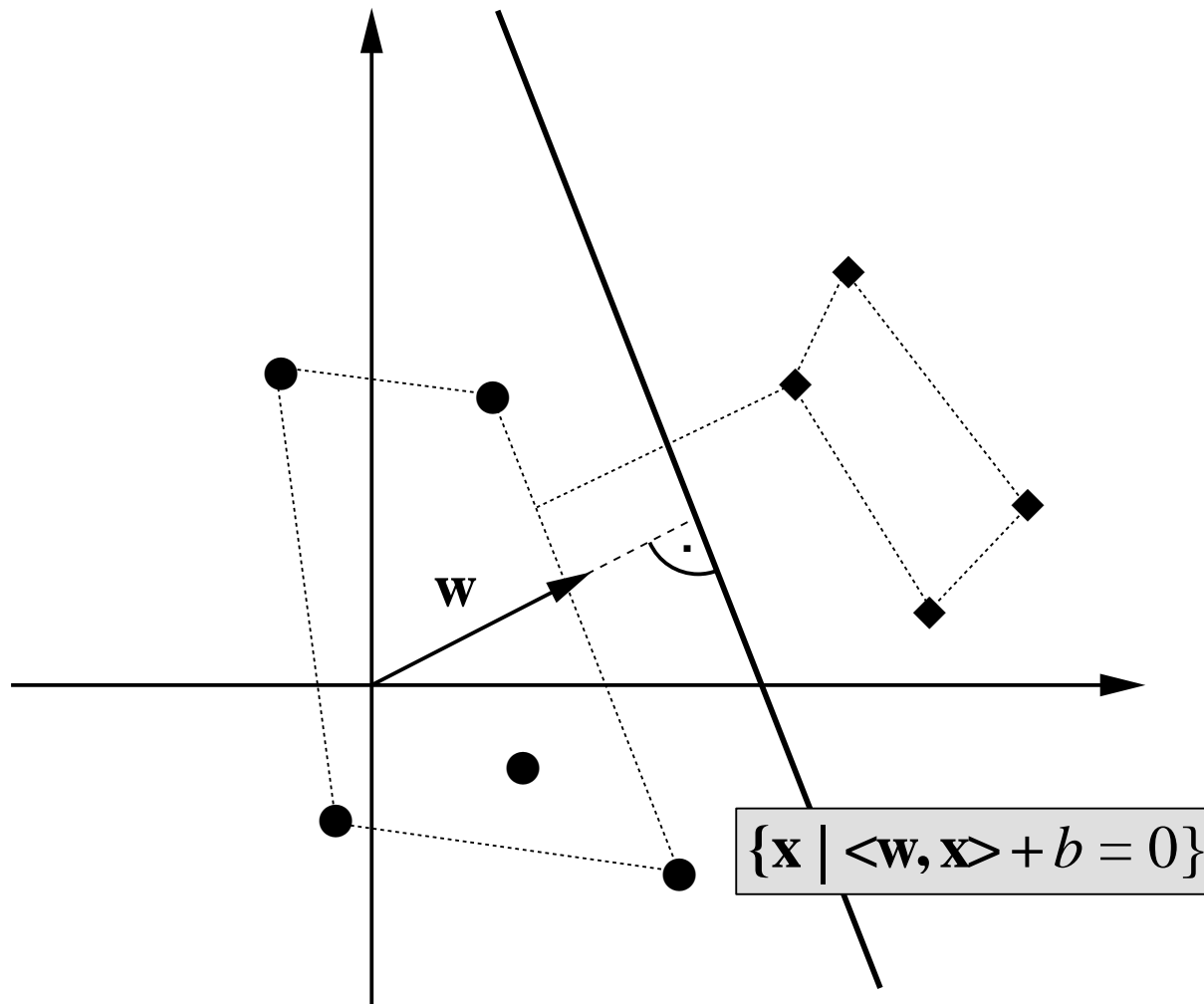
# Separating Hyperplane

---



# Optimal Separating Hyperplane

[71]





## Eliminating the Scaling Freedom

---

[72]

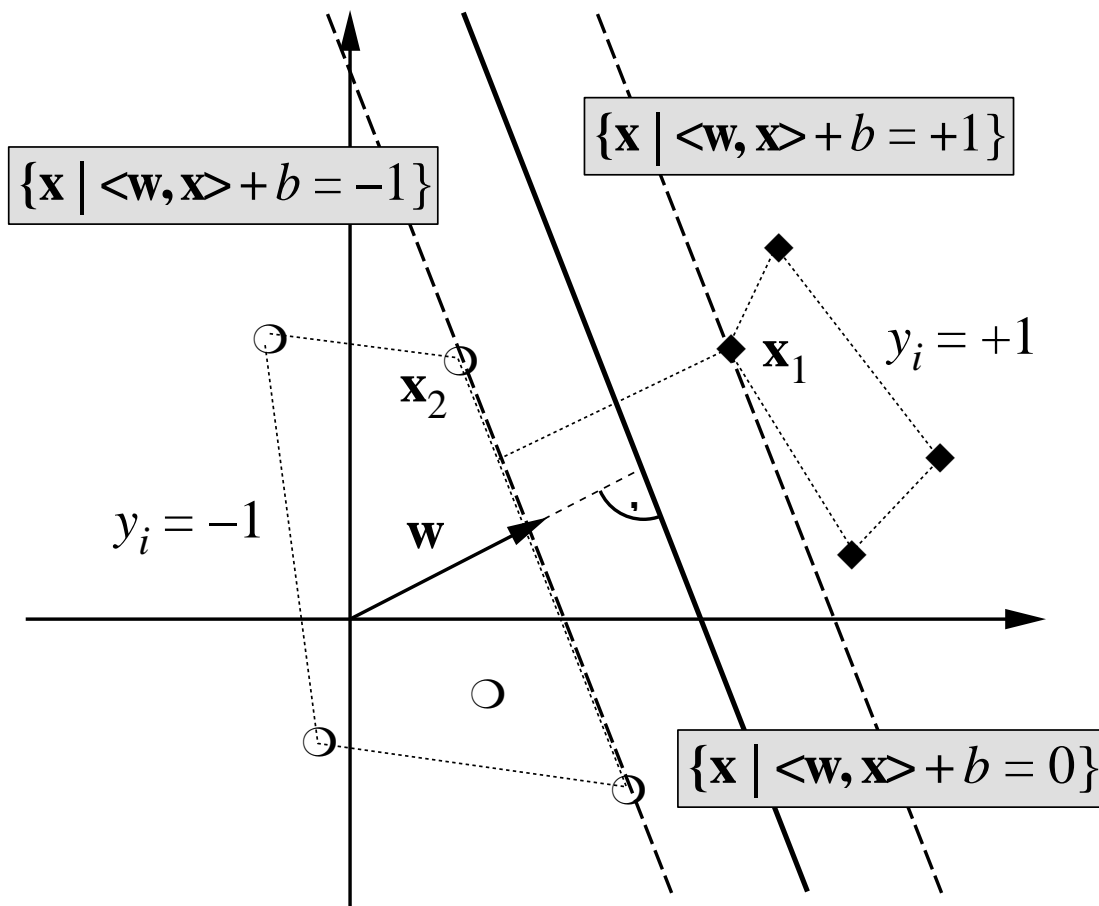
Note: if  $c \neq 0$ , then

$$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\} = \{\mathbf{x} \mid \langle c\mathbf{w}, \mathbf{x} \rangle + cb = 0\}.$$

Hence  $(c\mathbf{w}, cb)$  describes the same hyperplane as  $(\mathbf{w}, b)$ .

**Definition:** The hyperplane is in *canonical* form w.r.t.  $X^* = \{\mathbf{x}_1, \dots, \mathbf{x}_r\}$  if  $\min_{\mathbf{x}_i \in X} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1$ .

# Canonical Optimal Hyperplane



Note:

$$\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = +1$$

$$\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$$

$$\Rightarrow \langle \mathbf{w}, (\mathbf{x}_1 - \mathbf{x}_2) \rangle = 2$$

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

## Formulation as an Optimization Problem

---

Hyperplane with **maximum margin**: minimize

$$\|\mathbf{w}\|^2$$

(recall: margin  $\sim 1/\|\mathbf{w}\|$ ) subject to

$$y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1 \quad \text{for } i = 1 \dots m$$

(i.e. the training data are separated correctly).

# Lagrange Function

(e.g., [5])

Introduce Lagrange multipliers  $\alpha_i \geq 0$  and a Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^m \alpha_i (y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1).$$

$L$  has to be minimized w.r.t. the *primal variables*  $\mathbf{w}$  and  $b$  and maximized with respect to the *dual variables*  $\alpha_i$

- if a constraint is violated, then  $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 < 0 \longrightarrow$ 
  - $\alpha_i$  will grow to increase  $L$  — how far?
  - $\mathbf{w}$ ,  $b$  want to decrease  $L$ ; i.e. they have to change such that the constraint is satisfied. If the problem is separable, this ensures that  $\alpha_i < \infty$ .
- similarly: if  $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 > 0$ , then  $\alpha_i = 0$ : otherwise,  $L$  could be increased by decreasing  $\alpha_i$  (*KKT conditions*)

## Derivation of the Dual Problem

---

At the extremum, we have

$$\frac{\partial}{\partial b}L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}}L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0,$$

i.e.

$$\sum_{i=1}^m \alpha_i y_i = 0$$

and

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i.$$

Substitute both into  $L$  to get the *dual problem*

# The Support Vector Expansion

---

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

where for all  $i = 1, \dots, m$  either

$$y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] > 1 \quad \implies \alpha_i = 0 \longrightarrow \mathbf{x}_i \text{ irrelevant}$$

or

$$y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] = 1 \text{ (on the margin)} \longrightarrow \mathbf{x}_i \text{ “Support Vector”}$$

The solution is determined by the examples on the margin.

Thus

$$\begin{aligned} f(\mathbf{x}) &= \text{sgn}(\langle \mathbf{x}, \mathbf{w} \rangle + b) \\ &= \text{sgn}\left(\sum_{i=1}^m \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b\right). \end{aligned}$$

## Why it is Good to Have Few SVs

---

Leave out an example that does not become SV  $\longrightarrow$  same solution.

**Theorem [70]:** Denote  $\#SV(m)$  the number of SVs obtained by training on  $m$  examples randomly drawn from  $P(\mathbf{x}, y)$ , and  $\mathbf{E}$  the expectation. Then

$$\mathbf{E} [\text{Prob}(\text{test error})] \leq \frac{E [\#SV(m)]}{m}$$

Here,  $\text{Prob}(\text{test error})$  refers to the expected value of the risk, where the expectation is taken over training the SVM on samples of size  $m - 1$ .

## A Mechanical Interpretation

[11]

Assume that each SV  $\mathbf{x}_i$  exerts a perpendicular force of size  $\alpha_i$  and sign  $y_i$  on a solid plane sheet lying along the hyperplane.

Then the solution is mechanically stable:

$$\sum_{i=1}^m \alpha_i y_i = 0 \quad \text{implies that the forces sum to zero}$$

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i \quad \text{implies that the torques sum to zero,}$$

via

$$\sum_i \mathbf{x}_i \times y_i \alpha_i \cdot \mathbf{w} / \|\mathbf{w}\| = \mathbf{w} \times \mathbf{w} / \|\mathbf{w}\| = 0.$$



## Dual Problem

---

Dual: maximize

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

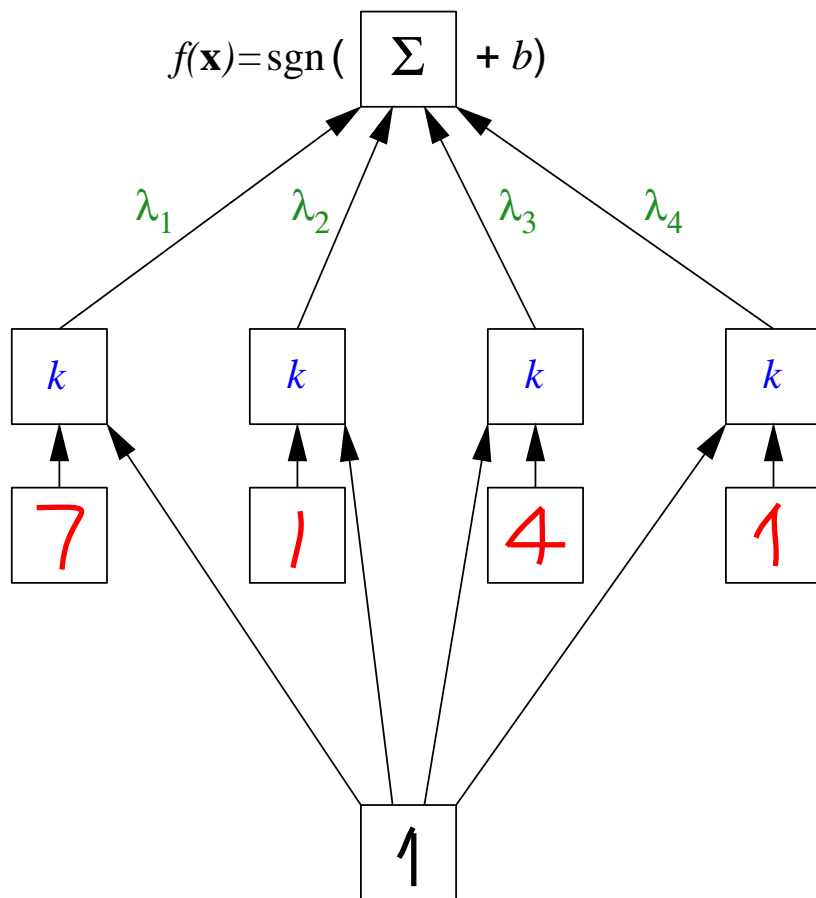
subject to

$$\alpha_i \geq 0, \quad i = 1, \dots, m, \quad \text{and} \quad \sum_{i=1}^m \alpha_i y_i = 0.$$

Both the final decision function and the function to be maximized are expressed in dot products  $\longrightarrow$  can use a **kernel** to compute

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j).$$

# The SVM Architecture



classification

weights

comparison:  $k(\mathbf{x}, \mathbf{x}_i)$ , e.g.  $k(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)^d$

support vectors  
 $\mathbf{x}_1 \dots \mathbf{x}_4$

input vector  $\mathbf{x}$

$$f(\mathbf{x}) = \text{sgn}(\sum \lambda_i k(\mathbf{x}, \mathbf{x}_i) + b)$$

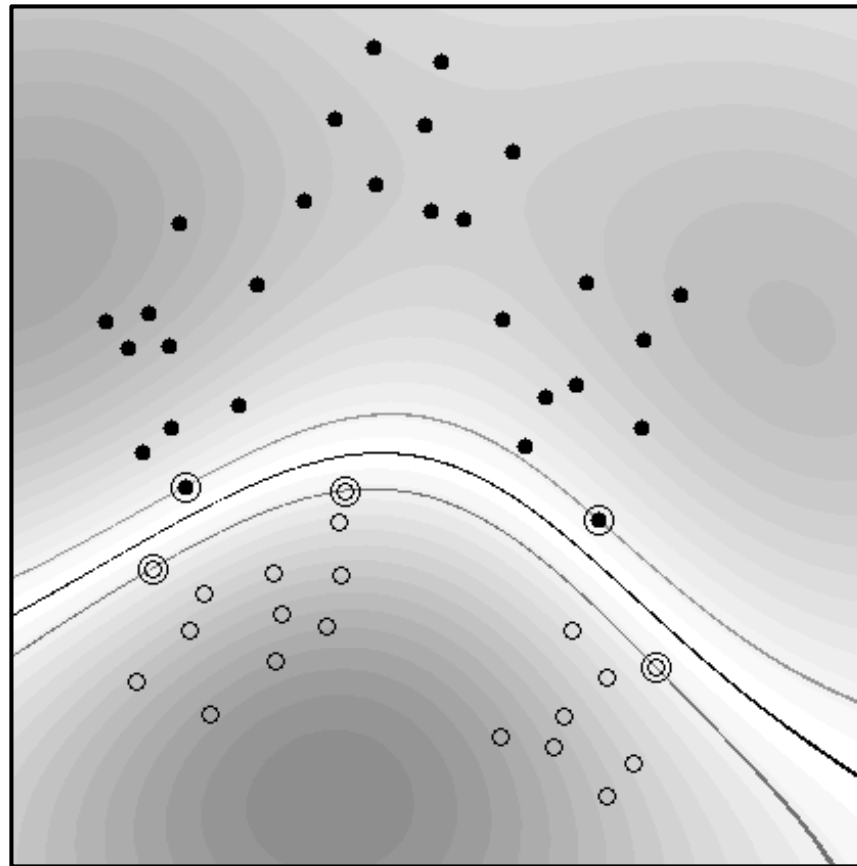
$$k(\mathbf{x}, \mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2 / c)$$

$$k(\mathbf{x}, \mathbf{x}_i) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{x}_i) + \theta)$$

# Toy Example with Gaussian Kernel

---

$$k(x, x') = \exp\left(-\|x - x'\|^2\right)$$



## Nonseparable Problems

---

[3, 15]

If  $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$  cannot be satisfied, then  $\alpha_i \rightarrow \infty$ .

Modify the constraint to

$$y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$$

with

$$\xi_i \geq 0$$

(“*soft margin*”) and add

$$C \cdot \sum_{i=1}^m \xi_i$$

in the objective function.

## Soft Margin SVMs

---

*C-SVM* [15]: for  $C > 0$ , minimize

$$\tau(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i$$

subject to  $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$ ,  $\xi_i \geq 0$  (margin  $2/\|\mathbf{w}\|$ )

*$\nu$ -SVM* [58]: for  $0 \leq \nu < 1$ , minimize

$$\tau(\mathbf{w}, \boldsymbol{\xi}, \rho) = \frac{1}{2} \|\mathbf{w}\|^2 - \nu \rho + \frac{1}{m} \sum_i \xi_i$$

subject to  $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \rho - \xi_i$ ,  $\xi_i \geq 0$  (margin  $2\rho/\|\mathbf{w}\|$ )

## The $\nu$ -Property

---

SVs:  $\alpha_i > 0$

“margin errors:”  $\xi_i > 0$

KKT-Conditions  $\implies$

- All margin errors are SVs.
- Not all SVs need to be margin errors.

Those which are *not* lie exactly on the edge of the margin.

### Proposition:

1. *fraction of Margin Errors*  $\leq \nu \leq$  *fraction of SVs*.
2. *asymptotically*: ... =  $\nu$  = ...

## Duals, Using Kernels

---

$C$ -SVM dual: maximize

$$W(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to  $0 \leq \alpha_i \leq C$ ,  $\sum_i \alpha_i y_i = 0$ .

$\nu$ -SVM dual: maximize

$$W(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to  $0 \leq \alpha_i \leq \frac{1}{m}$ ,  $\sum_i \alpha_i y_i = 0$ ,  $\sum_i \alpha_i \geq \nu$

In both cases: *decision function*:

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

## Connection between $\nu$ -SVC and $C$ -SVC

---

**Proposition.** If  $\nu$ -SV classification leads to  $\rho > 0$ , then  $C$ -SV classification, with  $C$  set a priori to  $1/\rho$ , leads to the same decision function.

**Proof.** Minimize the primal target, then fix  $\rho$ , and minimize only over the remaining variables: nothing will change. Hence the obtained solution  $\mathbf{w}_0, b_0, \boldsymbol{\xi}_0$  minimizes the primal problem of  $C$ -SVC, for  $C = 1$ , subject to

$$y_i \cdot (\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq \rho - \xi_i.$$

To recover the constraint

$$y_i \cdot (\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1 - \xi_i,$$

rescale to the set of variables  $\mathbf{w}' = \mathbf{w}/\rho, b' = b/\rho, \boldsymbol{\xi}' = \boldsymbol{\xi}/\rho$ . This leaves us, up to a constant scaling factor  $\rho^2$ , with the  $C$ -SV target with  $C = 1/\rho$ .



## SVM Training

---

- naive approach: the complexity of maximizing

$$W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

scales with the third power of the training set size  $m$

- only SVs are relevant  $\longrightarrow$  only compute  $(k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$  for SVs. Extract them iteratively by cycling through the training set in chunks [69].
- in fact, one can use chunks which do not even contain all SVs [43]. Maximize over these sub-problems, using your favorite optimizer.
- the extreme case: by making the sub-problems very small (just two points), one can solve them analytically [46].
- <http://www.kernel-machines.org/software.html>

# MNIST Benchmark

---

handwritten character benchmark (60000 training & 10000 test examples,  $28 \times 28$ )



## MNIST Error Rates

---

Classifier	test error	reference
linear classifier	8.4%	[7]
3-nearest-neighbour	2.4%	[7]
SVM	1.4%	[11]
Tangent distance	1.1%	[62]
LeNet4	1.1%	[39]
Boosted LeNet4	0.7%	[39]
Translation invariant SVM	0.56%	[19]

Note: the SVM used a polynomial kernel of degree 9, corresponding to a feature space of dimension  $\approx 3.2 \cdot 10^{20}$ .

Other successful applications: e.g., [35, 33, 31, 12, 67, 9, 84, 26, 24, 14, 22, 45, 77, 83]

## Speeding up the decision rule

---

Approximate

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \Phi(x_i)$$

by

$$\mathbf{w}' = \sum_{i=1}^{N_z} \gamma_i \Phi(z_i),$$

with  $N_z \ll m$ : Minimize

$$\rho = \|\mathbf{w} - \mathbf{w}'\|^2$$

Note that  $\rho$  can be expressed in terms of  $k$  by using  
 $\langle \Phi(x), \Phi(x') \rangle = k(x, x')$

Construct the new expansion sequentially.

“reduced set methods”, [e.g. 10, 11, 44, 53]

# Face Detection

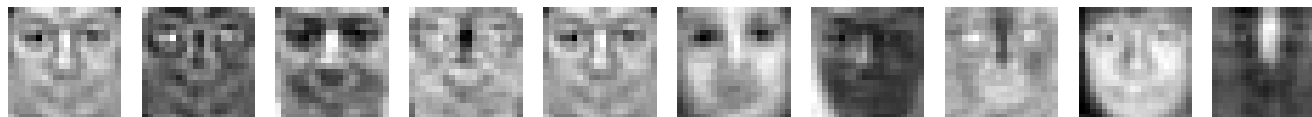
---

- scan test images in several resolutions
- critical issue: runtime speed. Compute sequential approximation via reduced set expansion.
- need to evaluate on average 2 – 3 kernels per image location [49]



after 0, 1 (*13.3% patches remaining*), 10 (*2.6%*), 20 (*0.01%*) and 30 (*0.002%*) kernels

templates:



Consider decision functions  $f(x) = \text{sgn}(g(x))$ , where

$$g(x) := \sum_{i=1}^m v_i \langle Bx, Bx_i \rangle + b.$$

To get local invariance under transformations of the Lie group  $\{\mathcal{L}_t\}$ , minimize the regularizer

$$\frac{1}{m} \sum_{j=1}^m \left( \left. \frac{\partial}{\partial t} \right|_{t=0} g(\mathcal{L}_t x_j) \right)^2.$$

This corresponds to an SV optimization after preprocessing with

$$B = C^{-\frac{1}{2}},$$

where

$$C = \frac{1}{m} \sum_{j=1}^m \left( \left. \frac{\partial}{\partial t} \right|_{t=0} \mathcal{L}_t x_j \right) \left( \left. \frac{\partial}{\partial t} \right|_{t=0} \mathcal{L}_t x_j \right)^\top.$$

## The Tangent Covariance Matrix

---

$C$  = covariance matrix of  $\pm \frac{\partial}{\partial t} \Big|_{t=0} \mathcal{L}_t x$

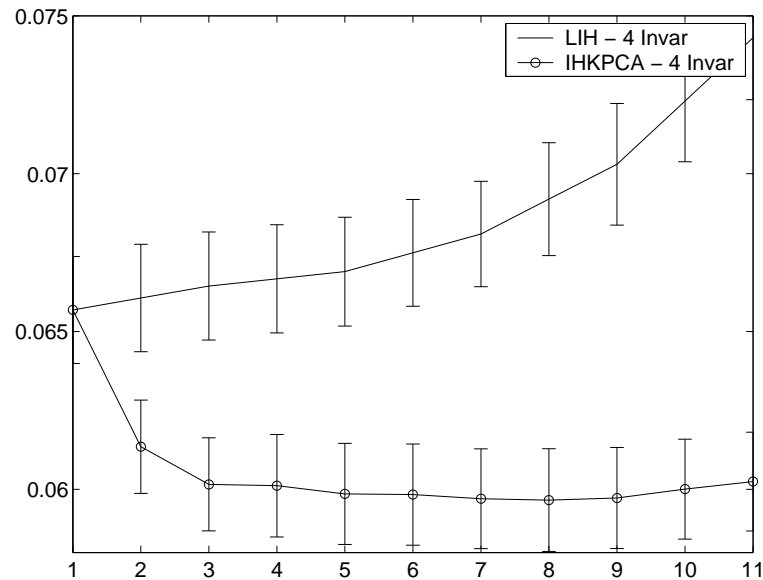
Preprocessing of  $x$ :

$$Bx = C^{-\frac{1}{2}}x = SD^{-\frac{1}{2}}S^\top x$$

1. project  $x$  onto the Eigenvectors of  $C$
  2. divide by the square roots of the Eigenvalues, i.e.: the directions of main variance of  $\pm \frac{\partial}{\partial t} \Big|_{t=0} \mathcal{L}_t x$  are scaled back
- in practice, use  $C_\lambda := (1 - \lambda)C + \lambda I$
  - for the nonlinear case, use the kernel PCA map [13]

# USPS Digit Recognition Application [13]

---



Results for 4 invariance transformations (translations) and different trade-offs between margin maximization and invariance enforcement (left: standard SVM).



## SV Regression: $\varepsilon$ -Insensitive Loss (*Vapnik, 1995*)

---

Goal: generalize SV pattern recognition to regression, preserving the following properties:

- formulate the algorithm for the linear case, and then use kernel trick
- sparse representation of the solution in terms of SVs

$\varepsilon$ -Insensitive Loss:

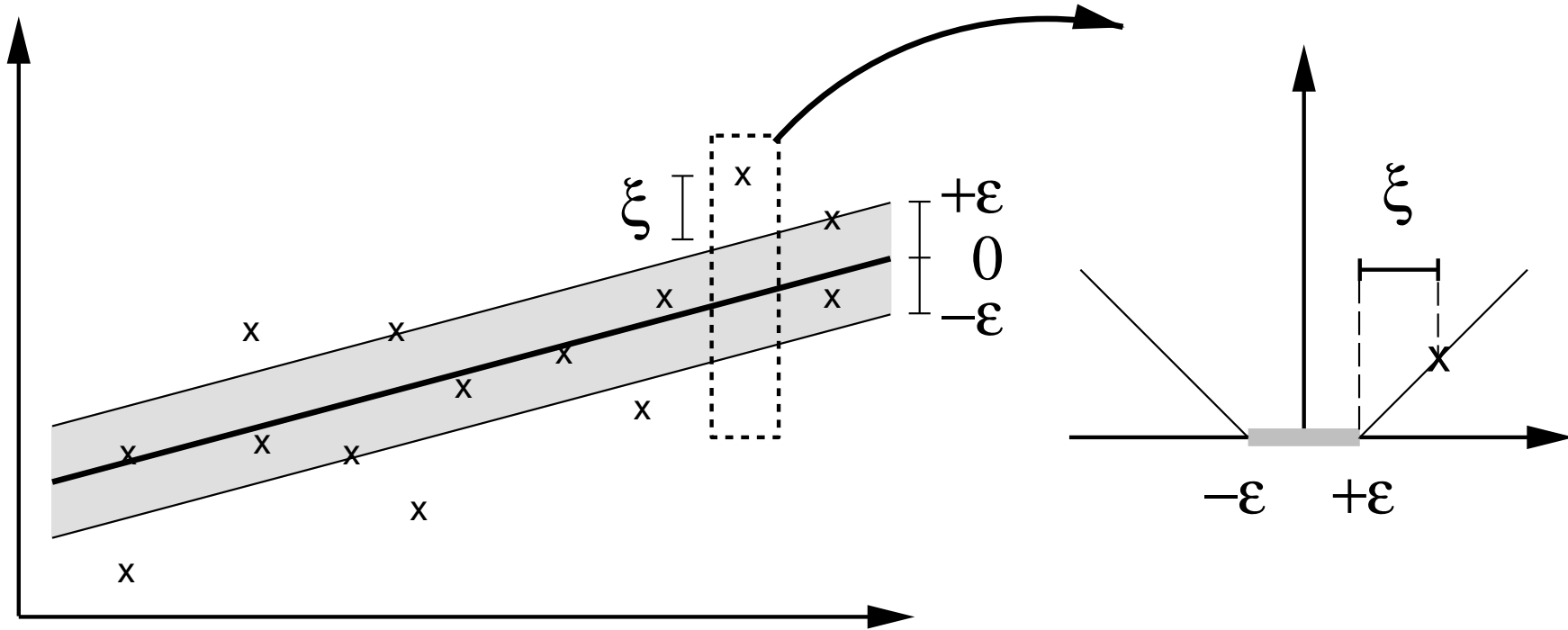
$$|y - f(\mathbf{x})|_{\varepsilon} := \max\{0, |y - f(\mathbf{x})| - \varepsilon\}$$

Estimate a linear regression  $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$  by minimizing

$$\frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{m} \sum_{i=1}^m |y_i - f(\mathbf{x}_i)|_{\varepsilon}.$$

# $\varepsilon$ -SV Regression Estimation

[72]



## Formulation as an Optimization Problem

---

Estimate a linear regression

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

with precision  $\varepsilon$  by minimizing

$$\text{minimize} \quad \tau(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m (\xi_i + \xi_i^*)$$

$$\begin{aligned} \text{subject to} \quad & (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \varepsilon + \xi_i \\ & y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \end{aligned}$$

for all  $i = 1, \dots, m$ .

## Dual Problem, In Terms of Kernels

---

For  $C > 0, \varepsilon \geq 0$  chosen a priori,

$$\begin{aligned} \text{maximize} \quad W(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) &= -\varepsilon \sum_{i=1}^m (\alpha_i^* + \alpha_i) + \sum_{i=1}^m (\alpha_i^* - \alpha_i) y_i \\ &\quad - \frac{1}{2} \sum_{i,j=1}^m (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) k(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

$$\text{subject to} \quad 0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1, \dots, m, \quad \text{and} \quad \sum_{i=1}^m (\alpha_i - \alpha_i^*) = 0.$$

The regression estimate takes the form

$$f(\mathbf{x}) = \sum_{i=1}^m (\alpha_i^* - \alpha_i) k(\mathbf{x}_i, \mathbf{x}) + b,$$

## $\nu$ -SV Regression

---

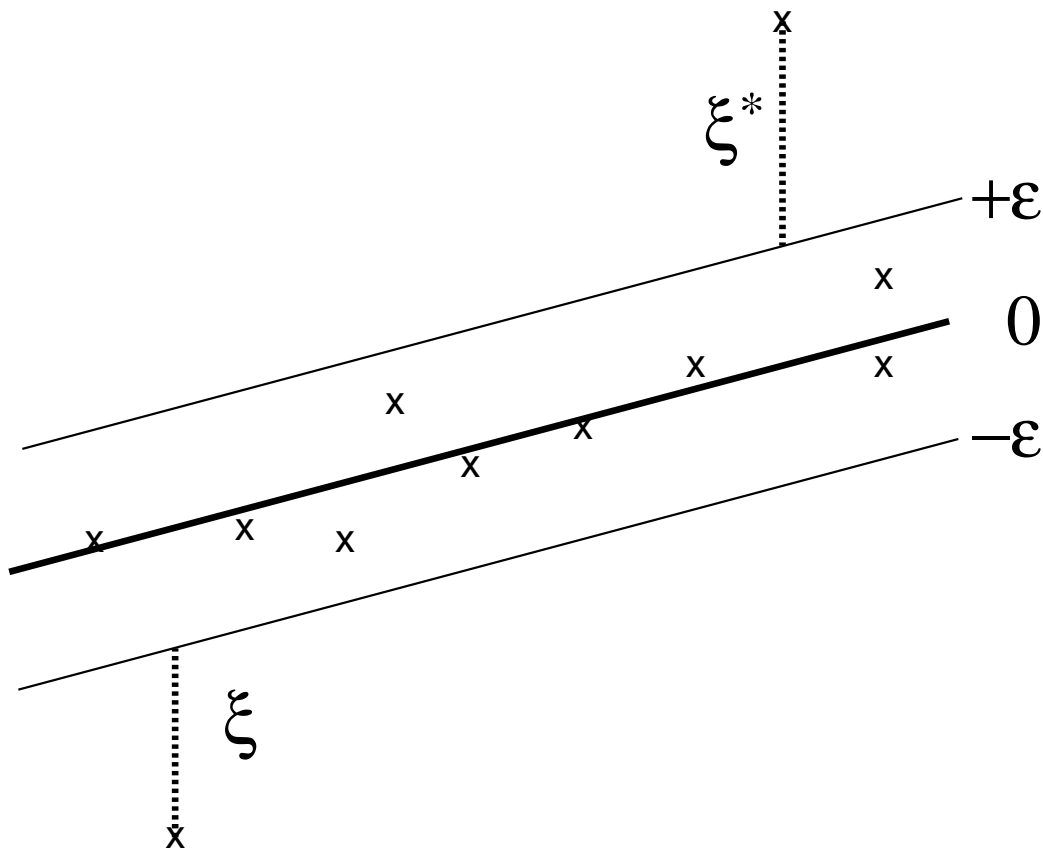
Again, use  $\nu$  to eliminate another parameter:  
Estimate  $\varepsilon$  from the data s.t. the  $\nu$ -property holds.

Primal problem: for  $0 \leq \nu \leq 1$ , minimize

$$\tau(\mathbf{w}, \varepsilon) = \frac{1}{2} \|\mathbf{w}\|^2 + C \left( \nu \varepsilon + \frac{1}{m} \sum_{i=1}^m |y_i - f(\mathbf{x}_i)|_\varepsilon \right)$$

# A Graphical Proof of the $\nu$ -Property

---



$$\text{Cost function: } \frac{1}{2C} \|\mathbf{w}\|^2 + \nu\epsilon + \frac{1}{m} \sum_{i=1}^m (\xi_i + \xi_i^*)$$

## The $\nu$ -Property

---

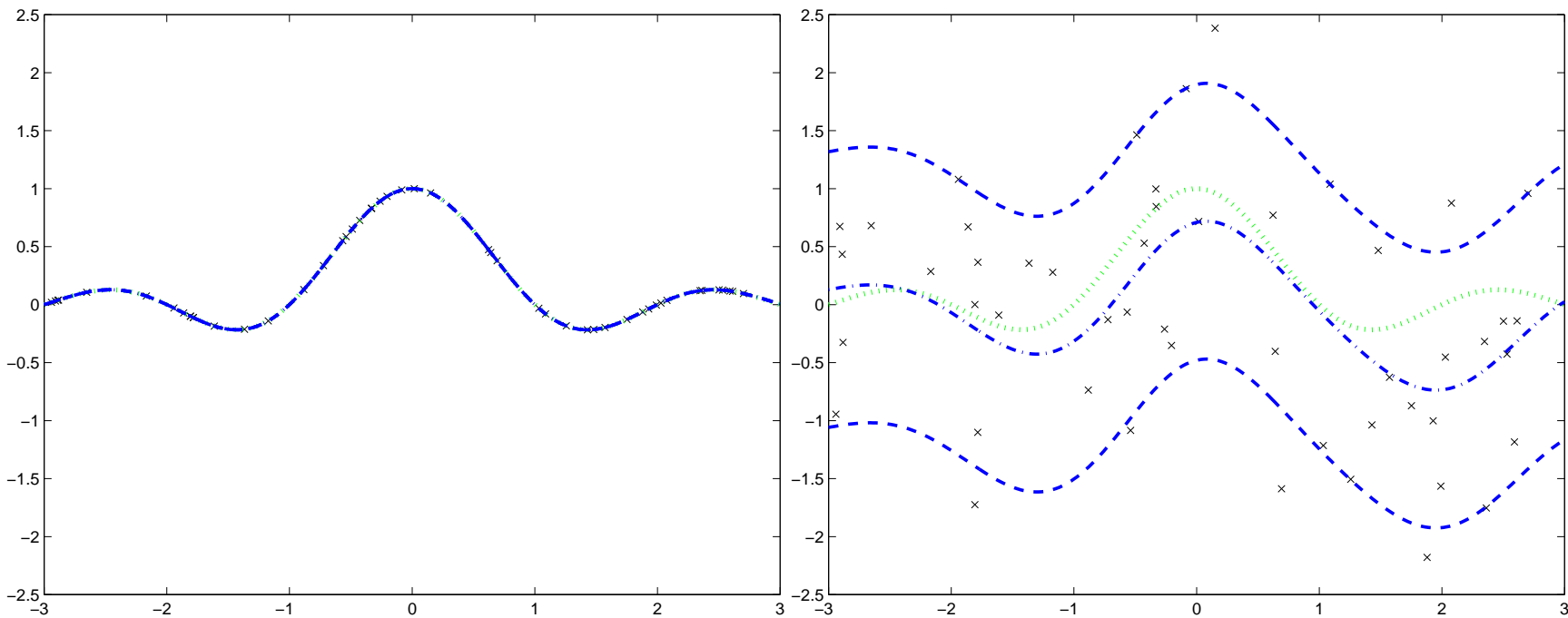
**Proposition 3** *Assume  $\varepsilon > 0$ . The following statements hold:*

- (i)  $\nu$  is an upper bound on the fraction of **errors**.*
- (ii)  $\nu$  is a lower bound on the fraction of **SVs**.*
- (iii) Suppose the data were generated iid from a 'well-behaved'\* distribution  $P(\mathbf{x}, y)$ . With probability 1, asymptotically,  $\nu$  equals both the fraction of **SVs** and the fraction of **errors**.*

\* Essentially,  $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$  with  $P(y|\mathbf{x})$  continuous (some details omitted).

## $\nu$ -SV-Regression: Automatic Tube Tuning

---

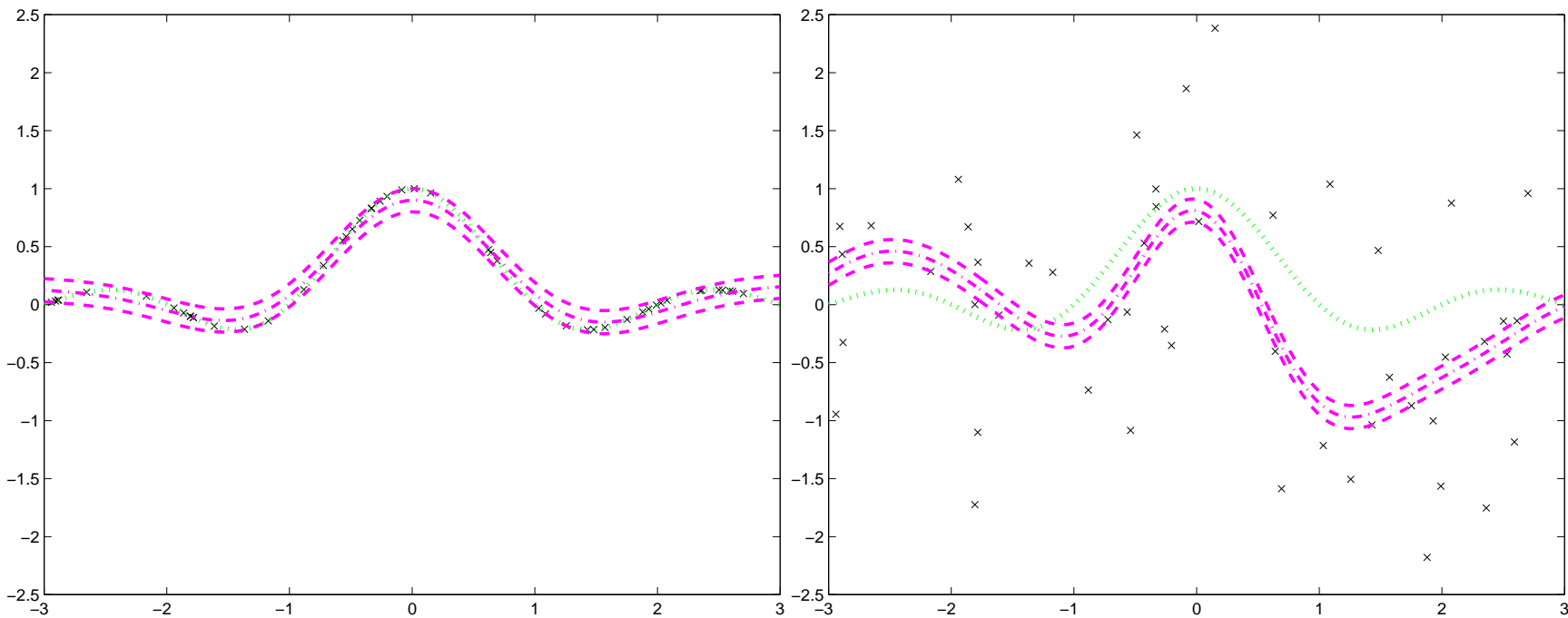


*Identical* machine parameters ( $\nu = 0.2$ ), but different amounts of noise in the data.



## $\varepsilon$ -SV-Regression, Run on the Same Data

---



*Identical* machine parameters ( $\varepsilon = 0.2$ ), but different amounts of noise in the data.

## Toy Examples: Estimating a Noisy Sinc Function

---

$$\nu = 0.2$$

$m$	10	50	100	200	500	1000	1500	2000
$\varepsilon$	0.27	0.22	0.23	0.25	0.26	0.26	0.26	0.26
fraction of errors	0.00	0.10	0.14	0.18	0.19	0.20	0.20	0.20
fraction of SVs	0.40	0.28	0.24	0.23	0.21	0.21	0.20	0.20

- automatically computed  $\varepsilon$  largely independent of  $m$
- asymptotics consistent with theorem

## Boston Housing Benchmark

---

- 506 examples, 13-dimensional.

Results (MSE):

- Bagging regression trees: 11.7 [8]
- $\varepsilon$ -SV regression: 7.6 [64]
  
- 100 runs, with 25 randomly selected test points.
- training set is split into actual training set and validation set (80 points) for selecting  $\varepsilon$ ,  $C$ , and kernel parameters
- <ftp://ftp.ics.uci.com/pub/machine-learning-databases/housing>

## Comparison: $\nu$ vs. $\varepsilon$

---

$\nu$ -SVR	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	
automatic $\varepsilon$	2.6	1.7	1.2	0.8	0.6	0.3	0.0	0.0	0.0	0.0	
<b>MSE</b>	9.4	8.7	9.3	9.5	10.0	10.6	11.3	11.3	11.3	11.3	
Errors	0.0	0.1	0.2	0.2	0.3	0.4	0.5	0.5	0.5	0.5	
SVs	0.3	0.4	0.6	0.7	0.8	0.9	1.0	1.0	1.0	1.0	
$\varepsilon$ -SVR	0	1	2	3	4	5	6	7	8	9	10
<b>MSE</b>	11.3	9.5	8.8	9.7	11.2	13.1	15.6	18.2	22.1	27.0	34.3
Errors	0.5	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SVs	1.0	0.6	0.4	0.3	0.2	0.1	0.1	0.1	0.1	0.1	0.1

- RBF kernel,  $C$  and  $\sigma$  chosen as in [59]

## Parametric Error Models

---

Use a tube of varying radius  $\zeta(\mathbf{x}) \geq 0$ :

minimize

$$\tau(\mathbf{w}, \boldsymbol{\xi}^{(*)}, \varepsilon) = \|\mathbf{w}\|^2/2 + C \cdot \left( \nu m \varepsilon + \sum_{i=1}^m (\xi_i + \xi_i^*) \right)$$

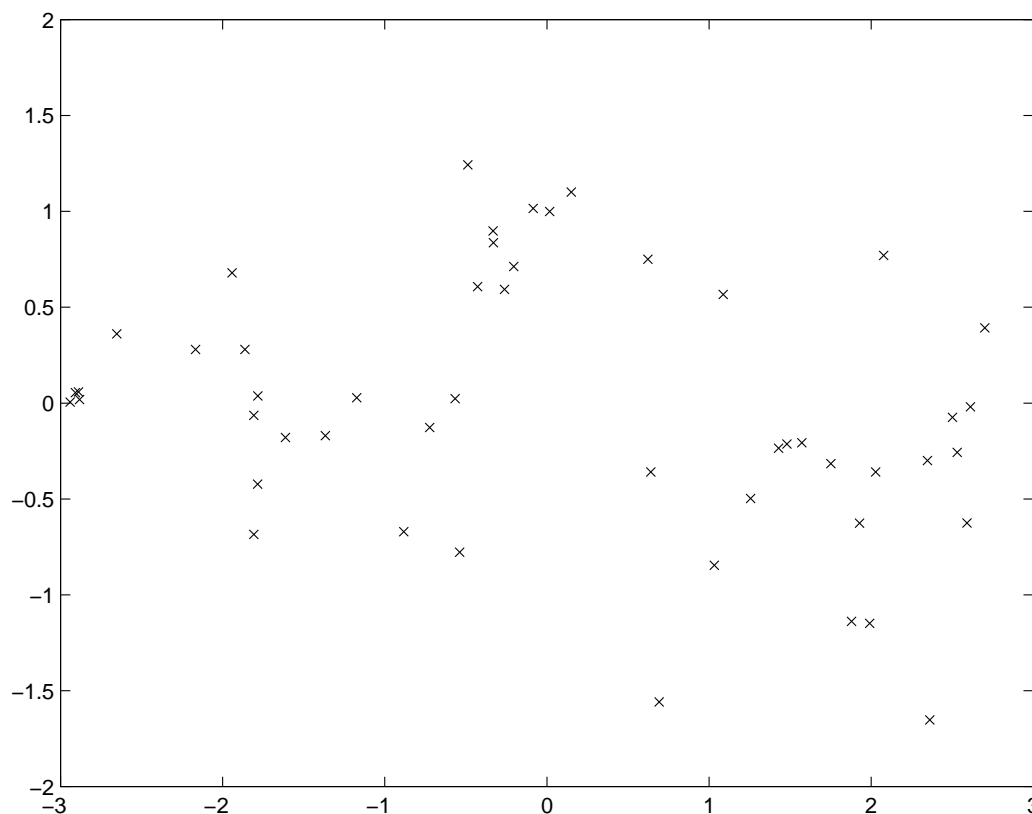
$$\begin{aligned} \text{subject to} \quad & (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \varepsilon \zeta(\mathbf{x}_i) + \xi_i \\ & y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \varepsilon \zeta(\mathbf{x}_i) + \xi_i^* \\ & \xi_i^{(*)} \geq 0, \quad \varepsilon \geq 0. \end{aligned}$$

This leads to the “usual” dual, with the modified last constraint

$$\sum_{i=1}^m (\alpha_i + \alpha_i^*) \zeta(\mathbf{x}_i) \leq C m \nu.$$

## Toy Example: Some Noisy Data

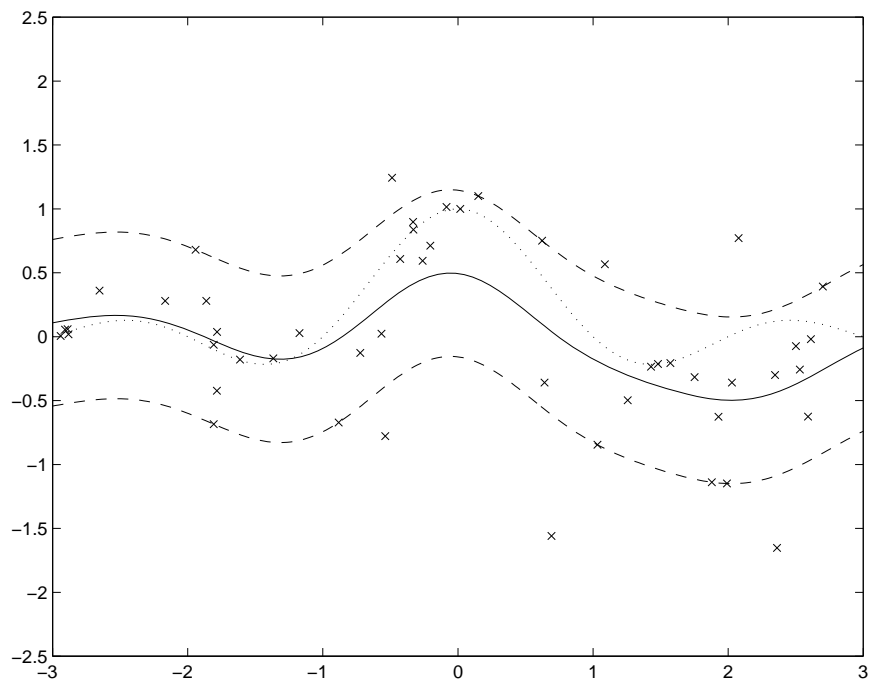
---



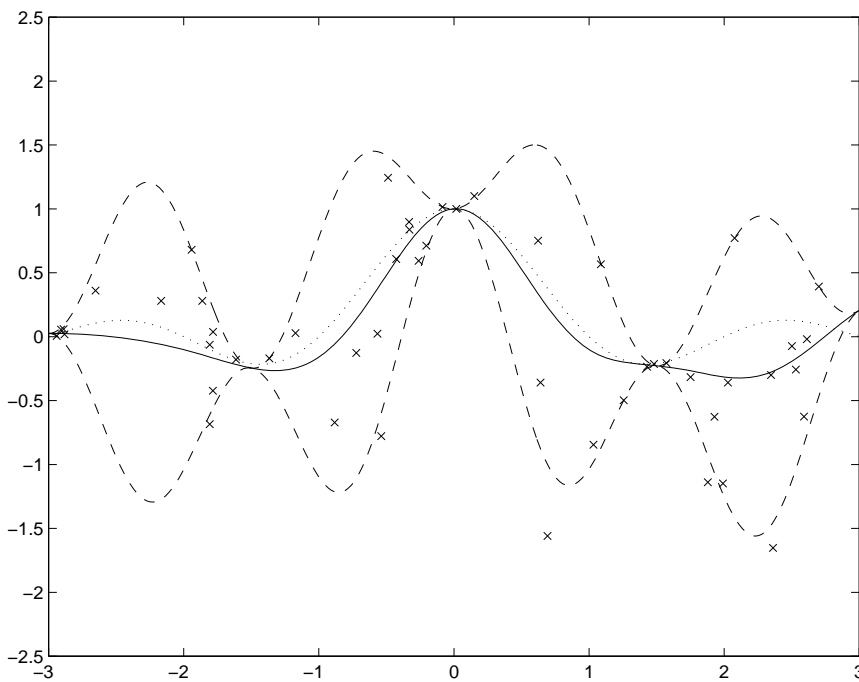
Assumption: we have prior knowledge indicating that the noise is modulated by  $\zeta(x) = \sin^2((2\pi/3)x)$ .

## Toy Example, II

---



constant-radius tube



parametric model using  $\zeta(x)$

## Robustness of SV Regression

---

**Proposition.** Using SVR with  $|\cdot|_\varepsilon$ , local movements of target values of points outside the tube do not change the estimated regression.

### Proof.

1. Shift  $y_i$  locally  $\longrightarrow (\mathbf{x}_i, y_i)$  still outside the tube  $\longrightarrow$  original dual solution  $\boldsymbol{\alpha}^{(*)}$  still feasible ( $\alpha_i^{(*)} = C$ , since *all* points outside the tube are at the upper bound).
2. The primal solution, with  $\xi_i$  transformed according to the movement, is also feasible.
3. The KKT conditions are still satisfied, as still  $\alpha_i^{(*)} = C$ . Thus [5, e.g.],  $\boldsymbol{\alpha}^{(*)}$  is still the optimal solution.



# The Representer Theorem

---

**Theorem 4** *Given: a p.d. kernel  $k$  on  $\mathcal{X} \times \mathcal{X}$ , a training set  $(x_1, y_1), \dots, (x_m, y_m) \in \mathcal{X} \times \mathbb{R}$ , a strictly monotonic increasing real-valued function  $\Omega$  on  $[0, \infty[$ , and an arbitrary cost function  $c : (\mathcal{X} \times \mathbb{R}^2)^m \rightarrow \mathbb{R} \cup \{\infty\}$*

*Any  $f \in \mathcal{H}$  minimizing the regularized risk functional*

$$c((x_1, y_1, f(x_1)), \dots, (x_m, y_m, f(x_m))) + \Omega(\|f\|) \quad (3)$$

*admits a representation of the form*

$$f(\cdot) = \sum_{i=1}^m \alpha_i k(x_i, \cdot).$$

## Remarks

---

- significance: many learning algorithms have solutions that can be expressed as expansions in terms of the training examples
- original form, with mean squared loss

$$c((x_1, y_1, f(x_1)), \dots, (x_m, y_m, f(x_m))) = \frac{1}{m} \sum_{i=1}^m (y_i - f(x_i))^2,$$

and  $\Omega(\|f\|) = \lambda \|f\|^2$  ( $\lambda > 0$ ): [37]

- generalization to non-quadratic cost functions: [16]
- present form: [56]

## Proof

---

Decompose  $f \in \mathcal{H}$  into a part in the span of the  $k(x_i, \cdot)$  and an orthogonal one:

$$f = \sum_i \alpha_i k(x_i, \cdot) + f_{\perp},$$

where for all  $j$

$$\langle f_{\perp}, k(x_j, \cdot) \rangle = 0.$$

Application of  $f$  to an arbitrary training point  $x_j$  yields

$$\begin{aligned} f(x_j) &= \langle f, k(x_j, \cdot) \rangle \\ &= \left\langle \sum_i \alpha_i k(x_i, \cdot) + f_{\perp}, k(x_j, \cdot) \right\rangle \\ &= \sum_i \alpha_i \langle k(x_i, \cdot), k(x_j, \cdot) \rangle, \end{aligned}$$

independent of  $f_{\perp}$ .

## Proof: second part of (3)

---

Since  $f_{\perp}$  is orthogonal to  $\sum_i \alpha_i k(x_i, \cdot)$ , and  $\Omega$  is strictly monotonic, we get

$$\begin{aligned}\Omega(\|f\|) &= \Omega\left(\left\|\sum_i \alpha_i k(x_i, \cdot) + f_{\perp}\right\|\right) \\ &= \Omega\left(\sqrt{\left\|\sum_i \alpha_i k(x_i, \cdot)\right\|^2 + \|f_{\perp}\|^2}\right) \\ &\geq \Omega\left(\left\|\sum_i \alpha_i k(x_i, \cdot)\right\|\right),\end{aligned}\tag{4}$$

with equality occurring if and only if  $f_{\perp} = 0$ .

Hence, any minimizer must have  $f_{\perp} = 0$ . Consequently, any solution takes the form

$$f = \sum_i \alpha_i k(x_i, \cdot).$$

## Application: Support Vector Classification

---

Here,  $y_i \in \{\pm 1\}$ . Use

$$c((x_i, y_i, f(x_i)))_i = \frac{1}{\lambda} \sum_i \max(0, 1 - y_i f(x_i)),$$

and the regularizer  $\Omega(\|f\|) = \|f\|^2$ .

$\lambda \rightarrow 0$  leads to the hard margin SVM

## Further Applications

---

*Bayesian MAP Estimates.* Identify (3) with the negative log posterior (cf. Kimeldorf & Wahba, 1970, Poggio & Girosi, 1990), i.e.

- $\exp(-c((x_i, y_i, f(x_i))_i))$  — likelihood of the data
- $\exp(-\Omega(\|f\|))$  — prior over the set of functions; e.g.,  $\Omega(\|f\|) = \lambda\|f\|^2$  — Gaussian process prior [81] with covariance function  $k$
- minimizer of (3) = MAP estimate

*Kernel PCA* (see below) can be shown to correspond to the case of

$$c((x_i, y_i, f(x_i))_{i=1, \dots, m}) = \begin{cases} 0 & \text{if } \frac{1}{m} \sum_i \left( f(x_i) - \frac{1}{m} \sum_j f(x_j) \right)^2 = 1 \\ \infty & \text{otherwise} \end{cases}$$

with  $g$  an arbitrary strictly monotonically increasing function.

## Regularization Interpretation of Kernel Machines

---

The norm in  $\mathcal{H}$  can be interpreted as a regularization term (Girosi 1998, Smola et al., 1998, Evgeniou et al., 2000): if  $P$  is a regularization operator (mapping into a dot product space  $\mathcal{D}$ ) such that  $k$  is Green's function of  $P^*P$ , then

$$\|\mathbf{w}\| = \|Pf\|,$$

where

$$\mathbf{w} = \sum_{i=1}^m \alpha_i \Phi(x_i)$$

and

$$f(x) = \sum_i \alpha_i k(x_i, x).$$

Example: for the Gaussian kernel,  $P$  is a linear combination of differential operators.

---


$$\begin{aligned}
\|\mathbf{w}\|^2 &= \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \\
&= \sum_{i,j} \alpha_i \alpha_j \langle k(x_i, \cdot), \delta_{x_j}(\cdot) \rangle \\
&= \sum_{i,j} \alpha_i \alpha_j \langle k(x_i, \cdot), (P^* P k)(x_j, \cdot) \rangle \\
&= \sum_{i,j} \alpha_i \alpha_j \langle (P k)(x_i, \cdot), (P k)(x_j, \cdot) \rangle_{\mathcal{D}} \\
&= \left\langle \left( P \sum_i \alpha_i k \right)(x_i, \cdot), \left( P \sum_j \alpha_j k \right)(x_j, \cdot) \right\rangle_{\mathcal{D}} \\
&= \|P f\|^2,
\end{aligned}$$

using  $f(x) = \sum_i \alpha_i k(x_i, x)$ .



# Further Kernel Algorithms — Design Principles

---

## 1. “Kernel module”

- similarity measure  $k(x, x')$ , where  $x, x' \in \mathcal{X}$
- data representation  
(in associated feature space where  $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ )  
— thus can construct geometric algorithms
- function class (representer theorem,  $f(x) = \sum_i \alpha_i k(x, x_i)$ )

## 2. “Learning module”

- classification
- quantile estimation / novelty detection
- feature extraction
- ...

# SV Morphing

---

...powerpoint

# Unsupervised SVM Learning

---

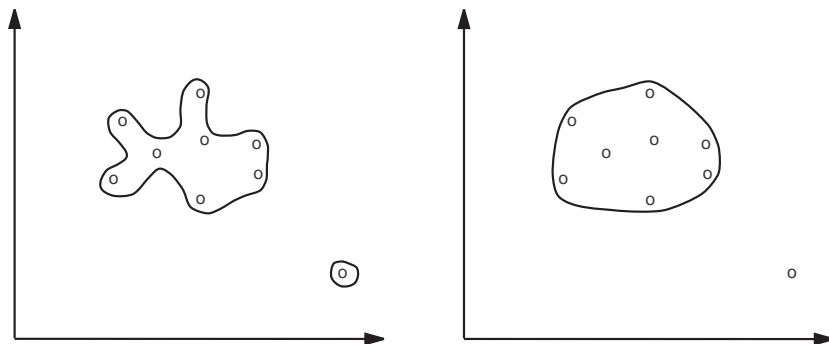
$x_1, \dots, x_m \in \mathcal{X}$  i.i.d. sample from  $P$

- extreme view: unsupervised learning = density estimation
- easier problem: for  $\alpha \in (0, 1]$ , compute a region  $R$  such that

$$P(R) \approx \alpha,$$

i.e., estimate *quantiles* of a distribution, not its density.

- becomes well-posed using a regularizer: find “smoothest” region that contains a certain fraction of the probability mass
- given only the training data, we will get a trade-off: try to enclose many training points (more than  $\alpha$ ) in a smooth region



## Multi-Dimensional Quantiles

---

- $\mathcal{C}$  a class of measurable subsets of  $\mathcal{X}$
- $\lambda$  a real-valued function on  $\mathcal{C}$
- *quantile function* with respect to  $(P, \lambda, \mathcal{C})$ :

$$U(\alpha) = \inf\{\lambda(C) \mid P(C) \geq \alpha, C \in \mathcal{C}\} \quad 0 < \alpha \leq 1.$$

- present case [54]:  $\lambda(C) \propto \frac{1}{\text{margin}^2}$ , where

$$\mathcal{C} := \{\text{half-spaces in } \mathcal{H}, \text{ not containing the origin}\}$$

## Separating Unlabelled Data from the Origin

---

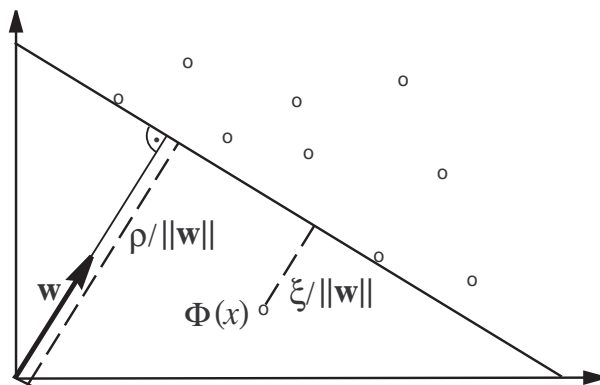
One can show: if  $\Phi(x_1), \dots, \Phi(x_m)$  are separable from the origin in  $\mathcal{H}$ , then the solution of

$$\min_{\mathbf{w} \in \mathcal{H}} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad \langle \mathbf{w}, \Phi(x_i) \rangle \geq 1$$

is the normal vector of the hyperplane separating the data from the origin with **maximum margin**.

## $\nu$ -Soft Margin Separation

---



For  $\nu \in (0, 1]$ , compute

$$\min_{\mathbf{w} \in \mathcal{H}, \boldsymbol{\xi} \in \mathbb{R}^m, \rho \in \mathbb{R}} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{m} \sum_i \xi_i - \nu \rho$$

subject to  $\langle \mathbf{w}, \Phi(x_i) \rangle \geq \rho - \xi_i, \quad \xi_i \geq 0 \quad \text{for all } i.$

Result:

- the decision function  $f(x) = \text{sgn}(\langle \mathbf{w}, \Phi(x) \rangle - \rho)$  will be positive for “most” examples  $x_i$  contained in the training set
- $\|\mathbf{w}\|$  will be small, hence the separation from the origin large

Related approaches: enclose data in a sphere [52, 65]

## Deriving the Dual Problem

---

Using multipliers  $\alpha_i, \beta_i \geq 0$ , we introduce a Lagrangian

$$L = \frac{\|\mathbf{w}\|^2}{2} + \frac{1}{\nu m} \sum_i \xi_i - \rho - \sum_i \alpha_i (\langle \mathbf{w}, \Phi(x_i) \rangle - \rho + \xi_i) - \sum_i \beta_i \xi_i,$$

and set the derivatives w.r.t. the primal variables  $\mathbf{w}, \boldsymbol{\xi}, \rho$  equal to zero, yielding

$$\mathbf{w} = \sum_i \alpha_i \Phi(x_i), \quad (5)$$

$$\alpha_i = \frac{1}{\nu m} - \beta_i \leq \frac{1}{\nu m}, \quad (6)$$

$$\sum_i \alpha_i = 1. \quad (7)$$

Patterns with  $\alpha_i > 0$  are **Support Vectors**.

## Dual Problem

---

$$\begin{aligned} & \min_{\boldsymbol{\alpha} \in \mathbb{R}^m} \quad \frac{1}{2} \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) \\ & \text{subject to } 0 \leq \alpha_i \leq \frac{1}{\nu m}, \quad \sum_i \alpha_i = 1. \end{aligned}$$

The decision function is

$$f(x) = \text{sgn} \left( \sum_i \alpha_i k(x_i, x) - \rho \right).$$

— a thresholded sparsified Parzen windows estimator



# Support Vectors and Outliers

---

$$SV := \{i \mid \alpha_i > 0\}; \quad OL := \{i \mid \xi_i > 0\}$$

The KKT-Conditions imply:

- $\xi_i > 0 \implies \alpha_i = 1/(\nu m)$ , hence  $OL \subset SV$
- $SV \setminus OL \subset \{i \mid \sum_j \alpha_j k(x_j, x_i) - \rho = 0\}$

## The Meaning of $\nu$

---

### Proposition.

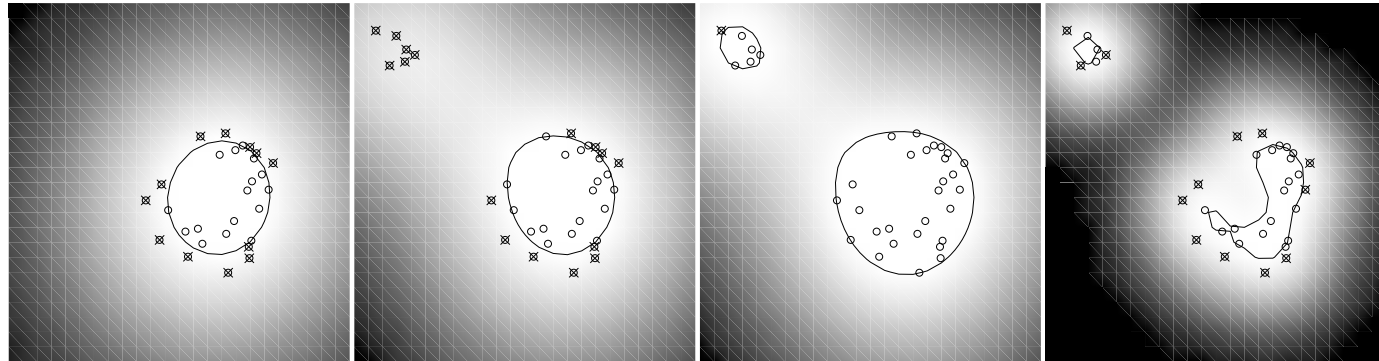
(i)

$$\frac{|OL|}{m} \leq \nu \leq \frac{|SV|}{m}$$

(ii) Suppose  $P$  does not contain discrete components, and the kernel is analytic and non-constant. With probability 1, asymptotically,

$$\frac{|OL|}{m} = \nu = \frac{|SV|}{m}.$$

# Toy Examples using $k(x, y) = \exp(-\frac{\|x-y\|^2}{c})$



$\nu$ , width $c$	0.5, 0.5	0.5, 0.5	0.1, 0.5	0.5, 0.1
SVs/OLs	0.54, 0.43	0.59, 0.47	0.24, 0.03	0.65, 0.38

## Error Bound for Single-Class Classification

---

For  $x \in \mathcal{X}$ ,  $\theta \in \mathbb{R}$ , let  $d(x, f, \theta) := \max\{0, \theta - f(x)\}$ . Similarly for  $X := (x_1, \dots, x_m)$ ,  $\mathcal{D}(X, f, \theta) := \sum_{x \in X} d(x, f, \theta)$ .

**Theorem 5** Denote

- $X \in \mathcal{X}^m$  a sample generated from an unknown distribution  $P$ , without discrete components
- $f_{\mathbf{w}}$  the solution of the optimization problem,
- $R_{\mathbf{w}, \rho} := \{x | f_{\mathbf{w}}(x) \geq \rho\}$  the induced decision region.

With probability  $1 - \delta$ , for any  $\gamma > 0$ ,

where 
$$P \{x' | x' \notin R_{\mathbf{w}, \rho - \gamma}\} \leq \frac{2}{m} (k + \log m^2 / (2\delta)),$$

$$k = \frac{c_1 \log(c_2 \hat{\gamma}^2 m)}{\hat{\gamma}^2} + \frac{2\mathcal{D}}{\hat{\gamma}} \log \left( e \left( \frac{(2m-1)\hat{\gamma}}{2\mathcal{D}} + 1 \right) \right) + 2,$$

$c_1 = 16c^2$ ,  $c_2 = \ln(2)/(4c^2)$ ,  $c = 103$ ,  $\hat{\gamma} = \gamma / \|\mathbf{w}\|$ ,  $\mathcal{D} = \mathcal{D}(X, f_{\mathbf{w}, 0}, \rho) = \mathcal{D}(X, f_{\mathbf{w}, \rho}, 0)$ .

## Discussion

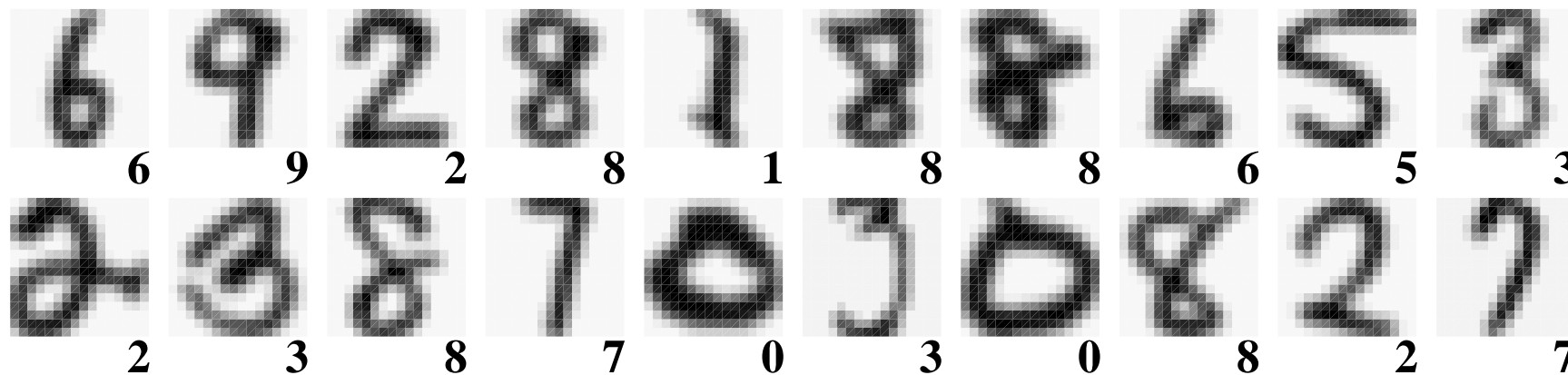
---

- algorithm tries to enclose training sample in  $R_{\mathbf{w},\rho}$
- theorem bounds the probability that *test* points will be in the larger region  $R_{\mathbf{w},\rho-\gamma}$
- a **small**  $\gamma$  leads to a **small** region but a **large** complexity term
- a **small**  $\|\mathbf{w}\|$  leads to a **small** complexity term (recall  $\hat{\gamma} = \gamma/\|\mathbf{w}\|$ )

# USPS Handwritten Digit Outlier Detection

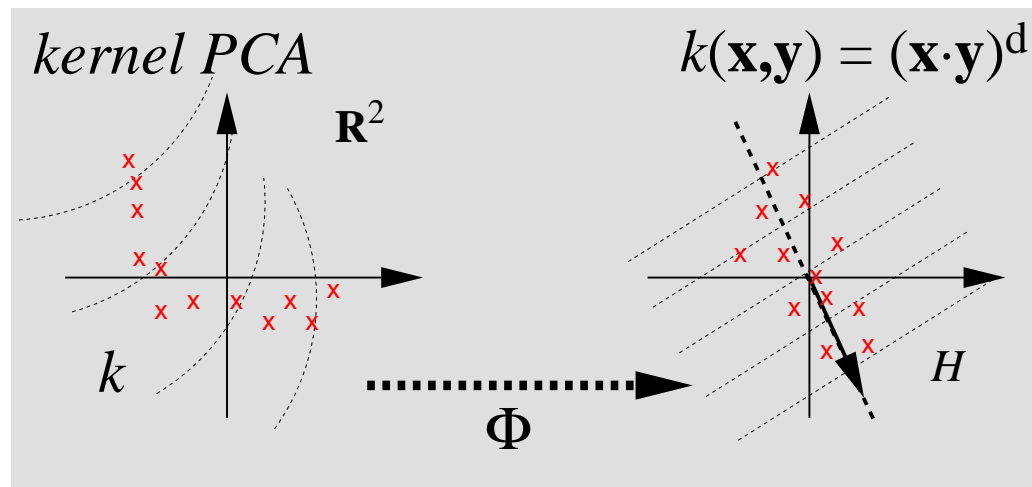
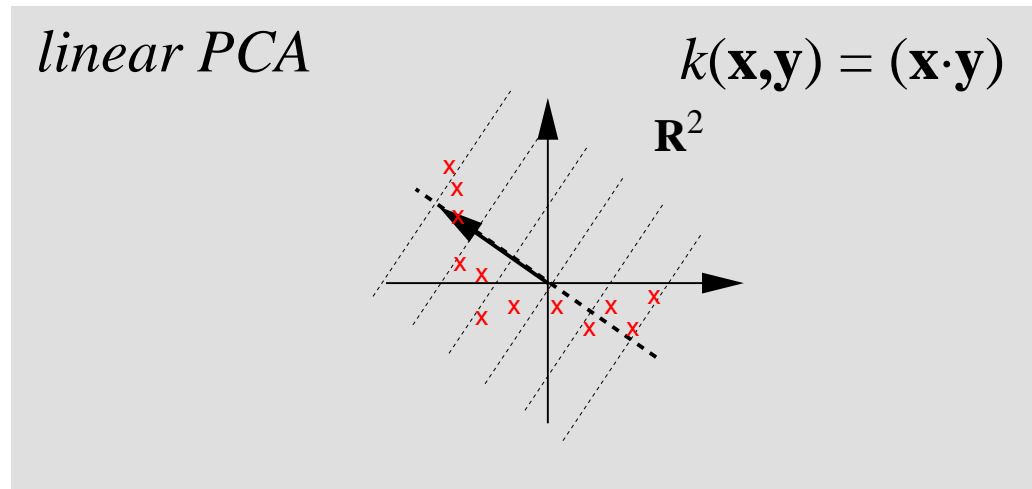
---

Typical examples (random selection):



Experiment: perform outlier detection on the 2007-element USPS test set (using  $\nu = 5\%$ )

Next slides: the outliers, ranked by their “badness”



## Kernel PCA, II

---

$$x_1, \dots, x_m \in \mathcal{X}, \quad \Phi : \mathcal{X} \rightarrow \mathcal{H}, \quad \mathbf{C} = \frac{1}{m} \sum_{j=1}^m \Phi(x_j) \Phi(x_j)^\top$$

Eigenvalue problem

$$\lambda \mathbf{V} = \mathbf{C} \mathbf{V} = \frac{1}{m} \sum_{j=1}^m \langle \Phi(x_j), \mathbf{V} \rangle \Phi(x_j).$$

For  $\lambda \neq 0$ ,  $\mathbf{V} \in \text{span}\{\Phi(x_1), \dots, \Phi(x_m)\}$ , thus

$$\mathbf{V} = \sum_{i=1}^m \alpha_i \Phi(x_i),$$

and the eigenvalue problem can be written as

$$\lambda \langle \Phi(x_n), \mathbf{V} \rangle = \langle \Phi(x_n), \mathbf{C} \mathbf{V} \rangle \quad \text{for all } n = 1, \dots, m$$



## Kernel PCA in Dual Variables

---

In term of the  $m \times m$  Gram matrix

$$K_{ij} := \langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j),$$

this leads to

$$m\lambda K\boldsymbol{\alpha} = K^2\boldsymbol{\alpha}$$

where  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)^\top$ .

Solve

$$m\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha}$$

→  $(\lambda_n, \boldsymbol{\alpha}^n)$

$$\langle \mathbf{V}^n, \mathbf{V}^n \rangle = 1 \iff \lambda_n \langle \boldsymbol{\alpha}^n, \boldsymbol{\alpha}^n \rangle = 1$$

thus divide  $\boldsymbol{\alpha}^n$  by  $\sqrt{\lambda_n}$

## Feature extraction

---

Compute projections on the Eigenvectors

$$\mathbf{V}^n = \sum_{i=1}^m \alpha_i^n \Phi(x_i)$$

in  $\mathcal{H}$ :

for a test point  $x$  with image  $\Phi(x)$  in  $\mathcal{H}$  we get the features

$$\begin{aligned} \langle \mathbf{V}^n, \Phi(x) \rangle &= \sum_{i=1}^m \alpha_i^n \langle \Phi(x_i), \Phi(x) \rangle \\ &= \sum_{i=1}^m \alpha_i^n k(x_i, x) \end{aligned}$$

## The Kernel PCA Map

---

Recall

$$\begin{aligned}\Phi_m^w : \mathcal{X} &\rightarrow \mathbb{R}^m \\ x &\mapsto K^{-\frac{1}{2}}(k(x_1, x), \dots, k(x_m, x))^\top\end{aligned}$$

If  $K = UDU^\top$  is  $K$ 's diagonalization, then  $K^{-1/2} = UD^{-1/2}U^\top$ . Thus we have

$$\Phi_m^w(x) = UD^{-1/2}U^\top(k(x_1, x), \dots, k(x_m, x))^\top.$$

We can drop the leading  $U$  (since it leaves the dot product invariant) to get a map

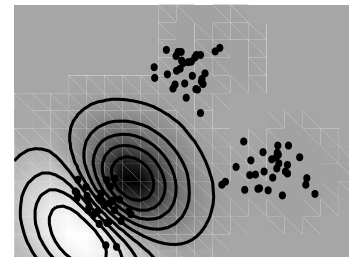
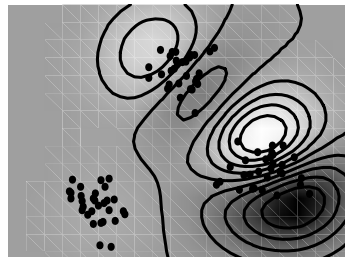
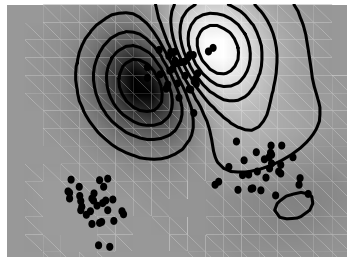
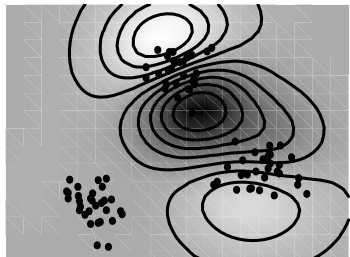
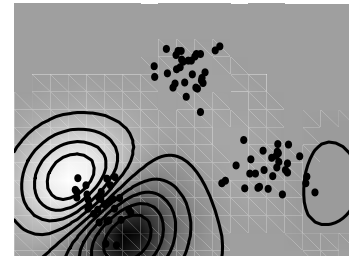
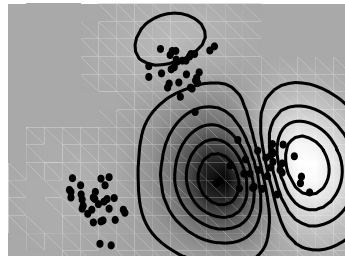
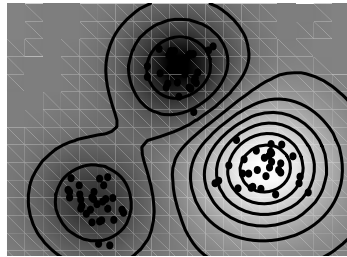
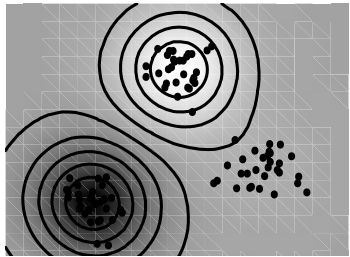
$$\Phi_{KPCA}^w(x) = D^{-1/2}U^\top(k(x_1, x), \dots, k(x_m, x))^\top.$$

The rows of  $U^\top$  are the eigenvectors  $\alpha^n$  of  $K$ , and the entries of the diagonal matrix  $D^{-1/2}$  equal  $\lambda_i^{-1/2}$ .

# Toy Example with Gaussian Kernel

---

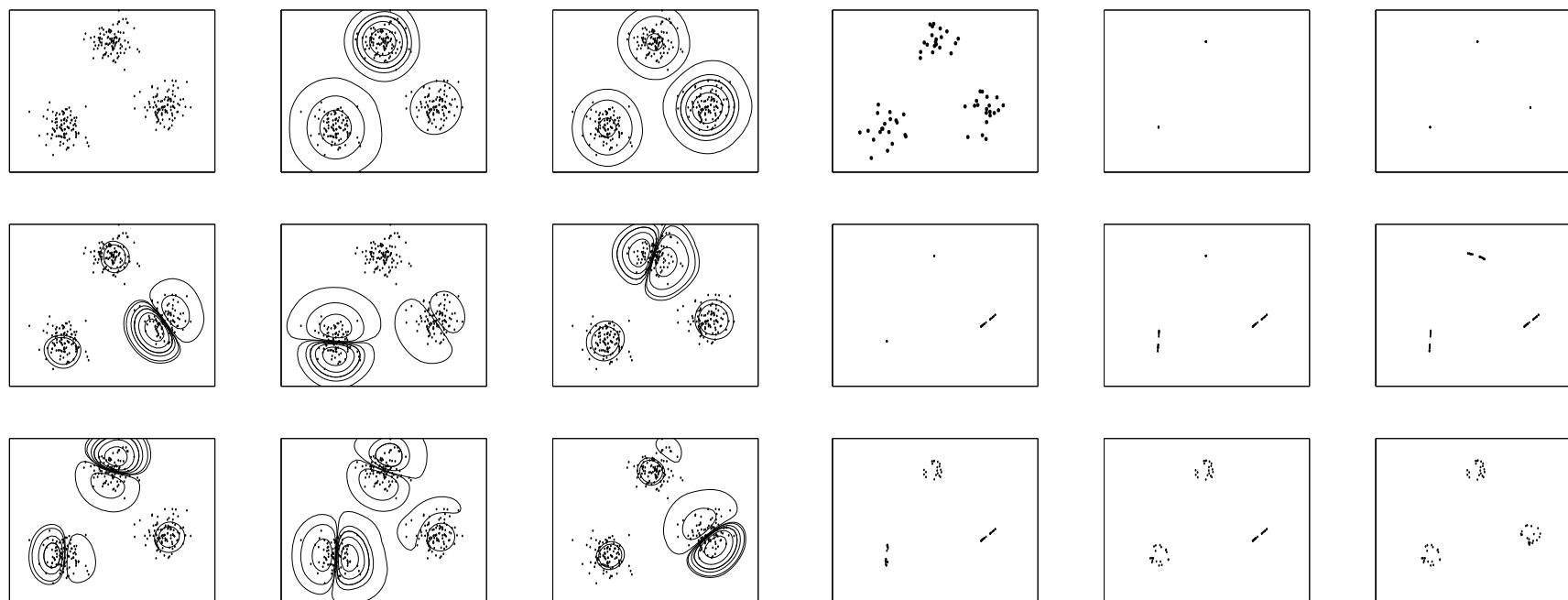
$$k(x, x') = \exp(-\|x - x'\|^2)$$



## Kernel PCA Denoising

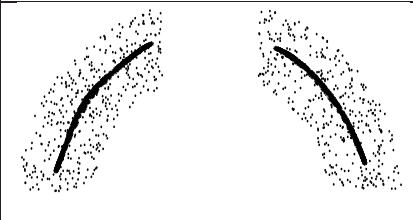
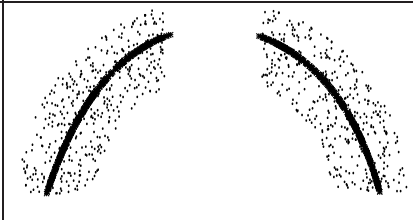
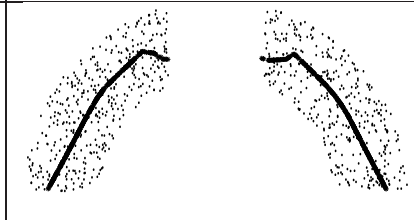
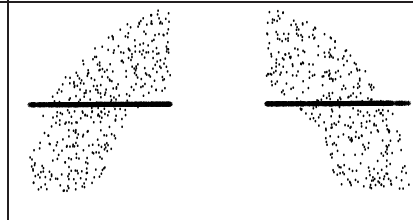
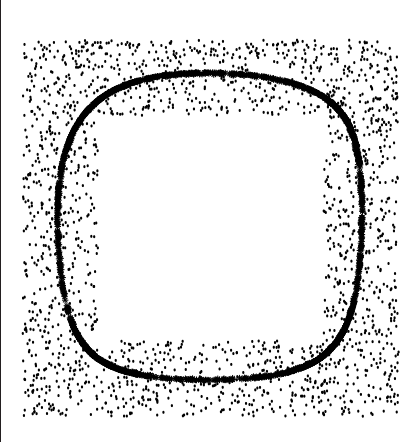
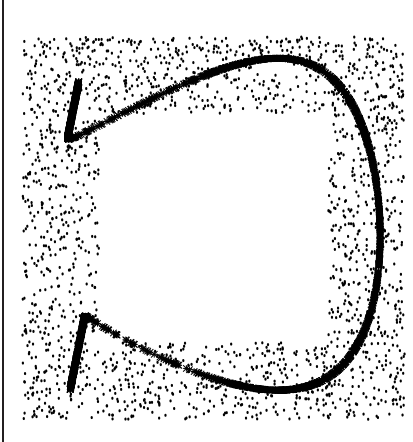
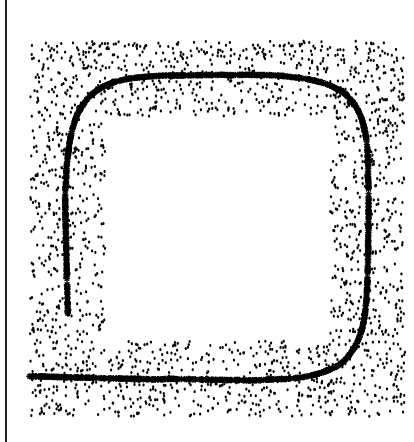
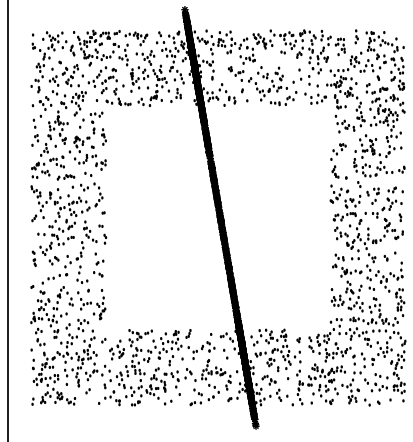
---

Idea: in feature space, discard higher-order principal components, and compute approximate pre-images [53].



Original data, first 8 feature extractors (*left*), pre-images computed by retaining 1...8 components in feature space (*right*).

# Comparison of Different Algorithms

kernel PCA (4 PCs)	nonlinear autoencoder	Principal Curves	linear PCA (1 PC)
			
			

[53, 29, 21]

# Denoising of USPS Digits

---

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

Other applications: face modeling [48], image superresolution (see below).

# Natural Image KPCA Model

---



Training images of size  $396 \times 528$ . The  $12 \times 12$  training patterns are obtained by sampling 2,500 patches at random from each image.





a



b



c



d

Example of natural image super-resolution: a. original image of resolution  $528 \times 396$ , b. low resolution image ( $264 \times 198$ ) stretched to the original scale, c. reconstruction of the high-frequency com-

# Super-Resolution

(Kim, Franz, & Schölkopf, 2004)



a. original image of resolution  $528 \times 396$



b. low resolution image ( $264 \times 198$ ) stretched to the original scale



c. bicubic interpolation



d. supervised example-based learning based on nearest neighbor classifier



f. unsupervised KHA reconstruction



g. enlarged portions of a-d, and f (from left to right)

Comparison between different super-resolution methods.

## Kernel Dependency Estimation

---

[80]

Given two sets  $\mathcal{X}$  and  $\mathcal{Y}$  with kernels  $k$  and  $k'$ , and training data  $(x_i, y_i)$ .

Estimate a dependency  $\mathbf{w} : \mathcal{H} \rightarrow \mathcal{H}'$

$$\mathbf{w}(\cdot) = \sum_{ij} \alpha_{ij} \Phi'(y_j) \langle \Phi(x_i), \cdot \rangle.$$

This can be evaluated in various ways, e.g., given an  $x$ , we can compute the pre-image

$$y = \operatorname{argmin}_{\mathcal{Y}} \|\mathbf{w}(\Phi(x)) - \Phi'(y)\|.$$

A convenient way of learning the  $\alpha_{ij}$  is to work in the kernel PCA basis.

## Application to Image Completion

---



Shown are all digits where at least one of the two algorithms makes a mistake (73 mistakes for  $k$ -NN, 23 for KDE).

*(from [80])*

# Vector Quantization

---

- given a set of  $m$  data vectors  $X = x_1, \dots, x_m$
- wish to represent them by a reduced number of  $M$  ‘codebook’ vectors  $V = v_1, \dots, v_M$
- Codebook  $V$  is chosen such that some overall measure of distortion is (approximately) minimized when each  $x$  is represented by its ‘nearest’  $v$ :

$$E_{VQ} = \sum_{n=1}^m D[v(x_n), x_n]$$

where  $v(x_n) = \operatorname{argmin}_{v \in V} D[v, x_n]$

- A common distortion is squared Euclidean distance:  $D[v, x_n] = \|v - x_n\|^2$

## Kernel VQ

---

- Conventionally: specify codebook size  $M$  and minimize  $E_{VQ}$  over  $V$ 
  - e.g., *Linde-Buzo-Gray* (LBG) algorithm

- kernel approach [66]:

- specify a *maximum distortion guarantee*:

$$D[v(x_n), x_n] \leq R \quad (*)$$

- constrain the codebook to be a *subset* of the data set:

$$\{v_1, \dots, v_M\} \subseteq \{x_1, \dots, x_m\}$$

- try to find  $v_1, \dots, v_M$  with minimal  $M$  such that  $(*)$  holds

(Tipping & Schölkopf, 2001 [66])

- 
- define a kernel:

$$k(x_i, x_n) = \begin{cases} 1 & \text{if } D[x_i, x_n] \leq R \\ 0 & \text{otherwise} \end{cases}$$

- seek a *sparse* vector  $\mathbf{w} = (w_1, \dots, w_m)$  such that for all  $x_n$

$$\sum_{i=1}^m w_i k(x_i, x_n) > 0$$

- Every  $x_n$  lies within ‘distance’  $R$  of at least one  $x_i$  for which  $w_i > 0$
- recall the empirical kernel map

$$\Phi_m(x) = (k(x_1, x), \dots, k(x_m, x))$$

- 
- seek solutions with few positive  $w_m$  by solving the optimization problem:

$$\begin{aligned} & \min_{\mathbf{w}} \quad \|\mathbf{w}\|_q \\ & \text{subject to} \quad \mathbf{w}^\top \Phi_m(x_n) \geq 1 \quad \text{for all } x_n \in X \end{aligned}$$

- Ideally, we would choose  $q = 0$ , since  $\|\mathbf{w}\|_0$  counts the non-zero coefficients
- But  $q = 1$  leads to a tractable *linear programming* problem
- Penalizers of the form  $\|\mathbf{w}\|_1$  generally lead to sparse solutions



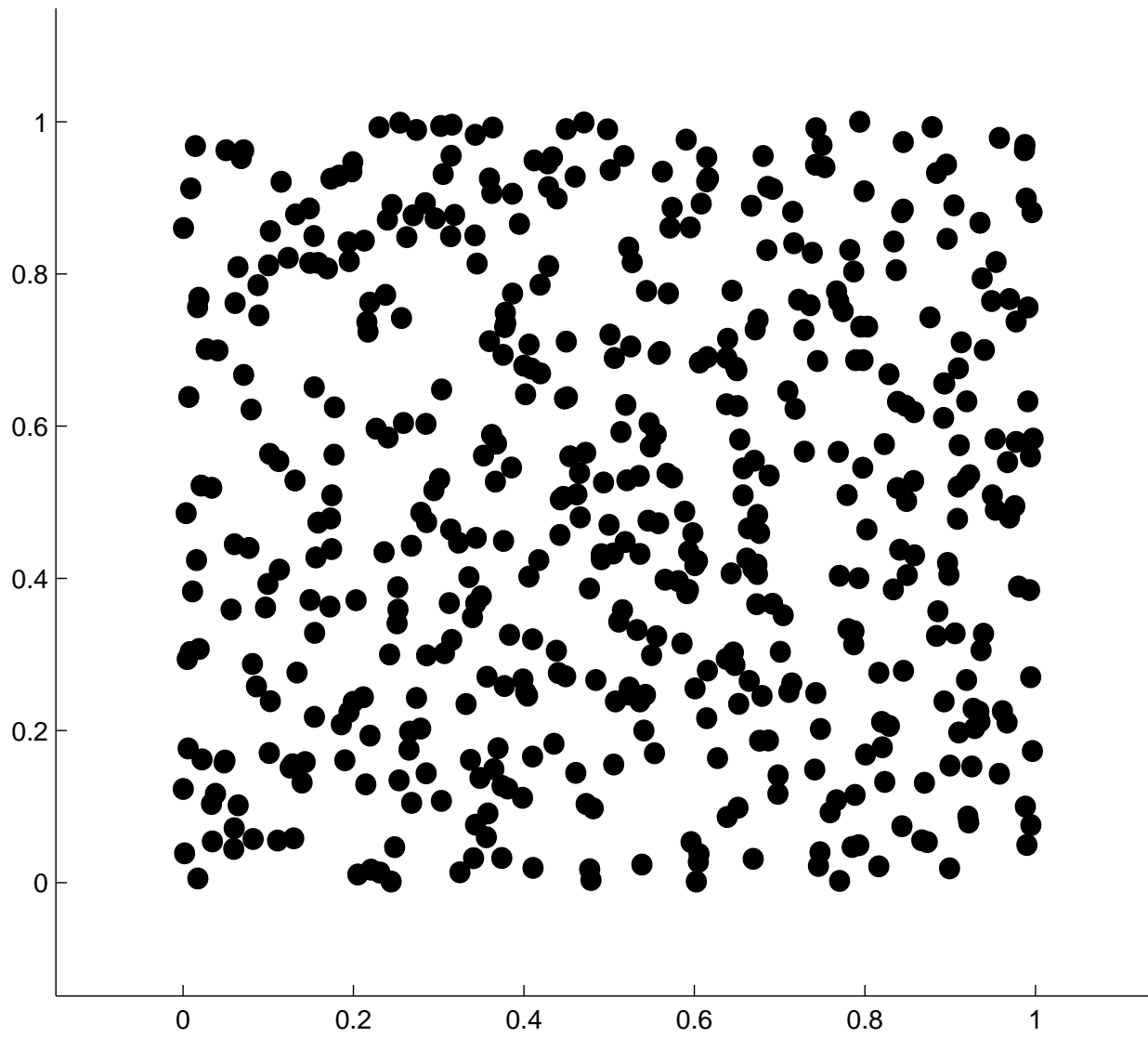
## Practicalities

---

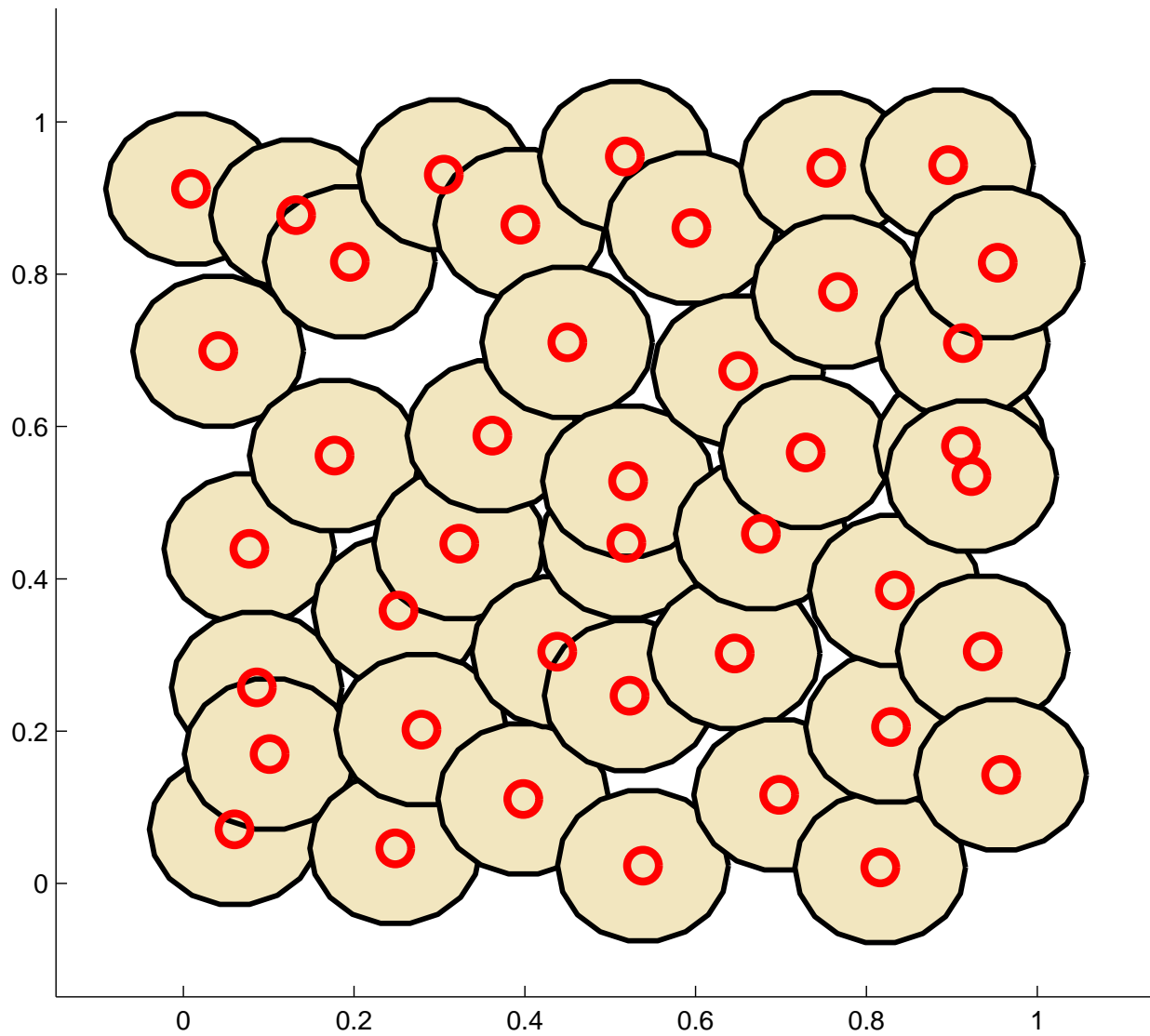
- Actual penalty used:

$$\sum_{m=1}^m \frac{|w_m|}{c_m}$$

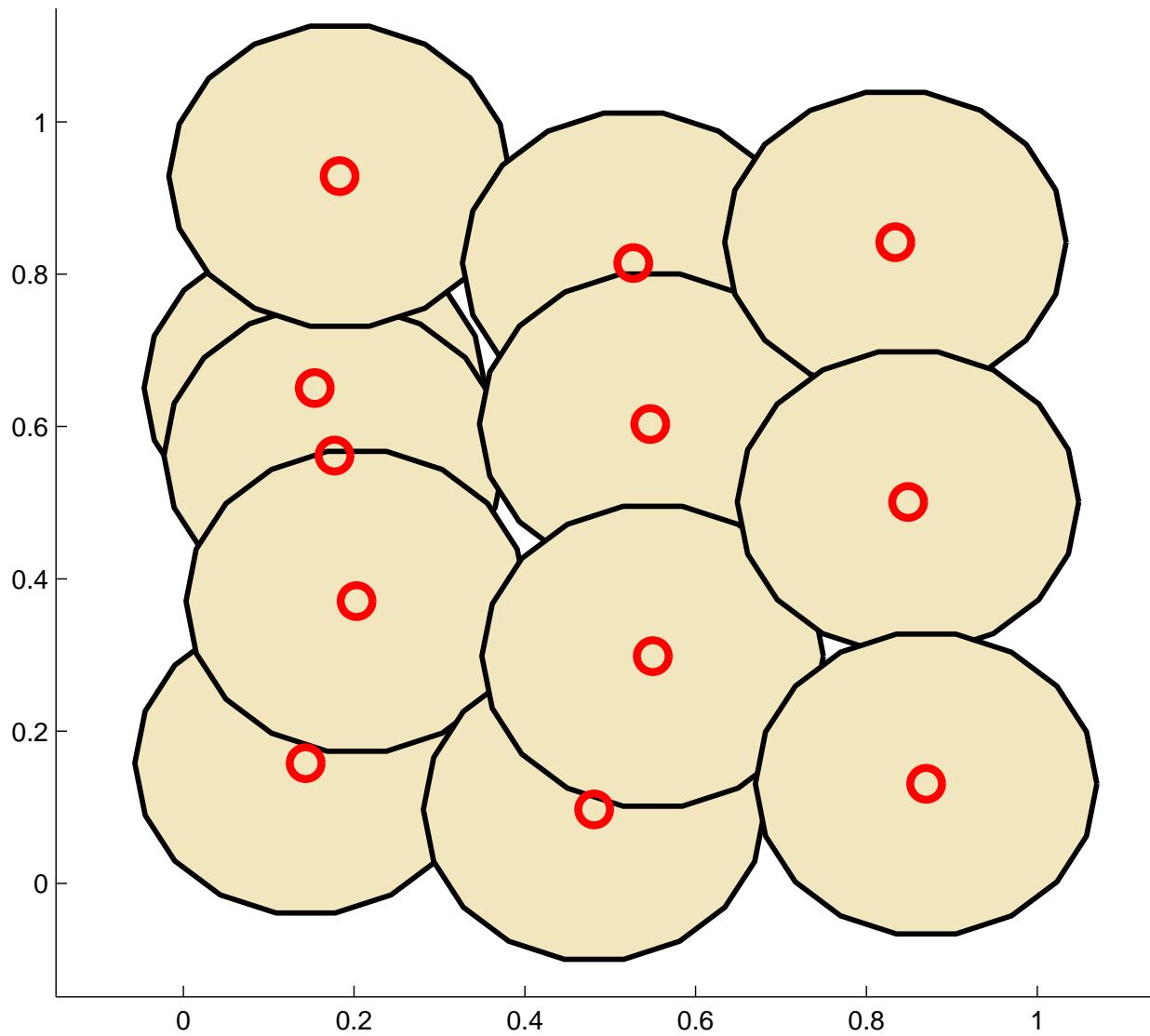
- $c_m = \sum_n k(\mathbf{x}_m, \mathbf{x}_n)$  the number of examples in the support of  $k(\mathbf{x}_m, \mathbf{x})$
- this improves sparsity without affecting the constraints
- perform a final ‘pruning’ step since symmetries in many tasks still give a number of superfluous vectors
  - a consequence of using the  $q = 1$  rather than  $q = 0$  penalty
  - typically, this step removes a further 1%–5% of vectors



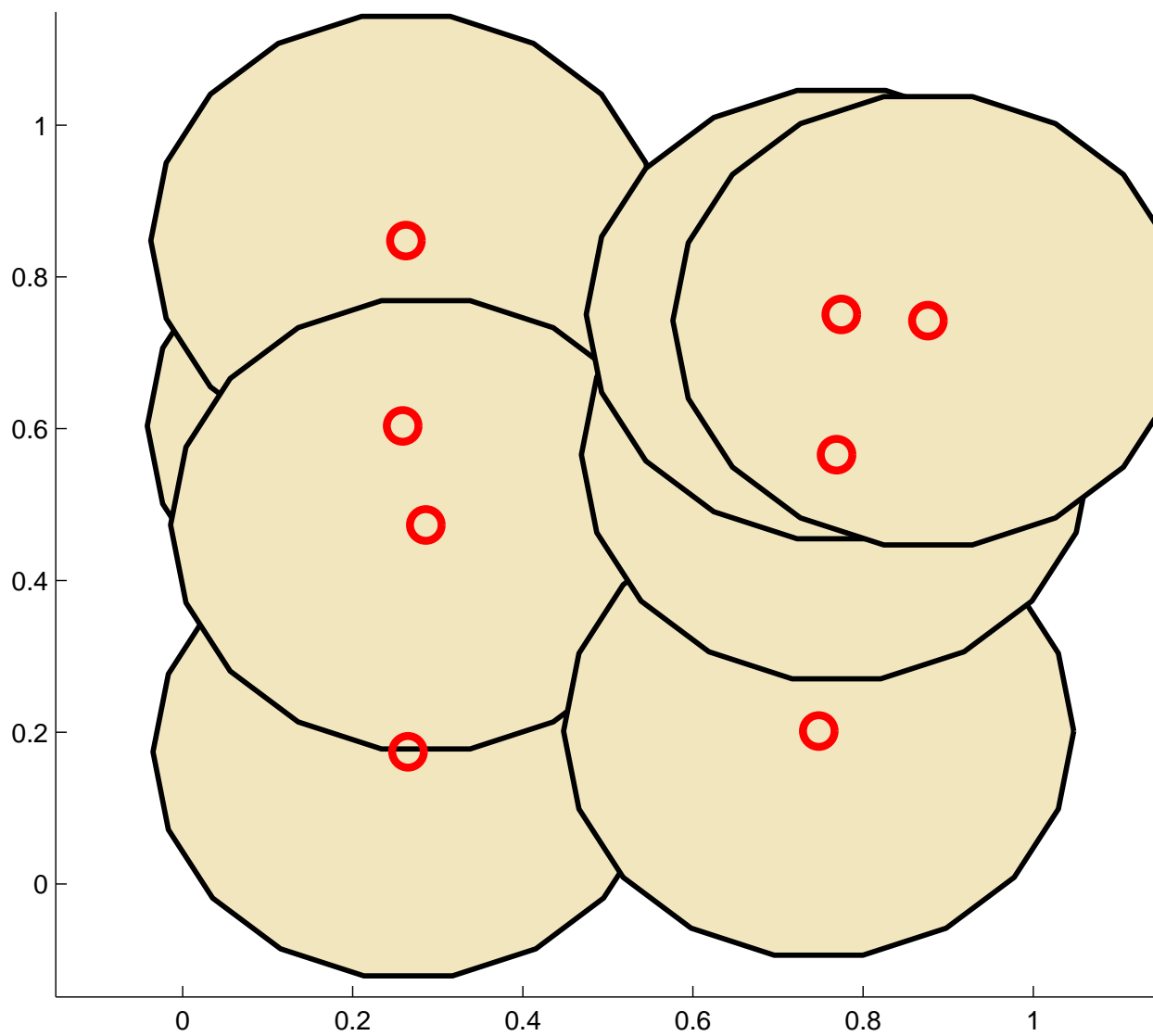
$R=0.1$   $M=42$



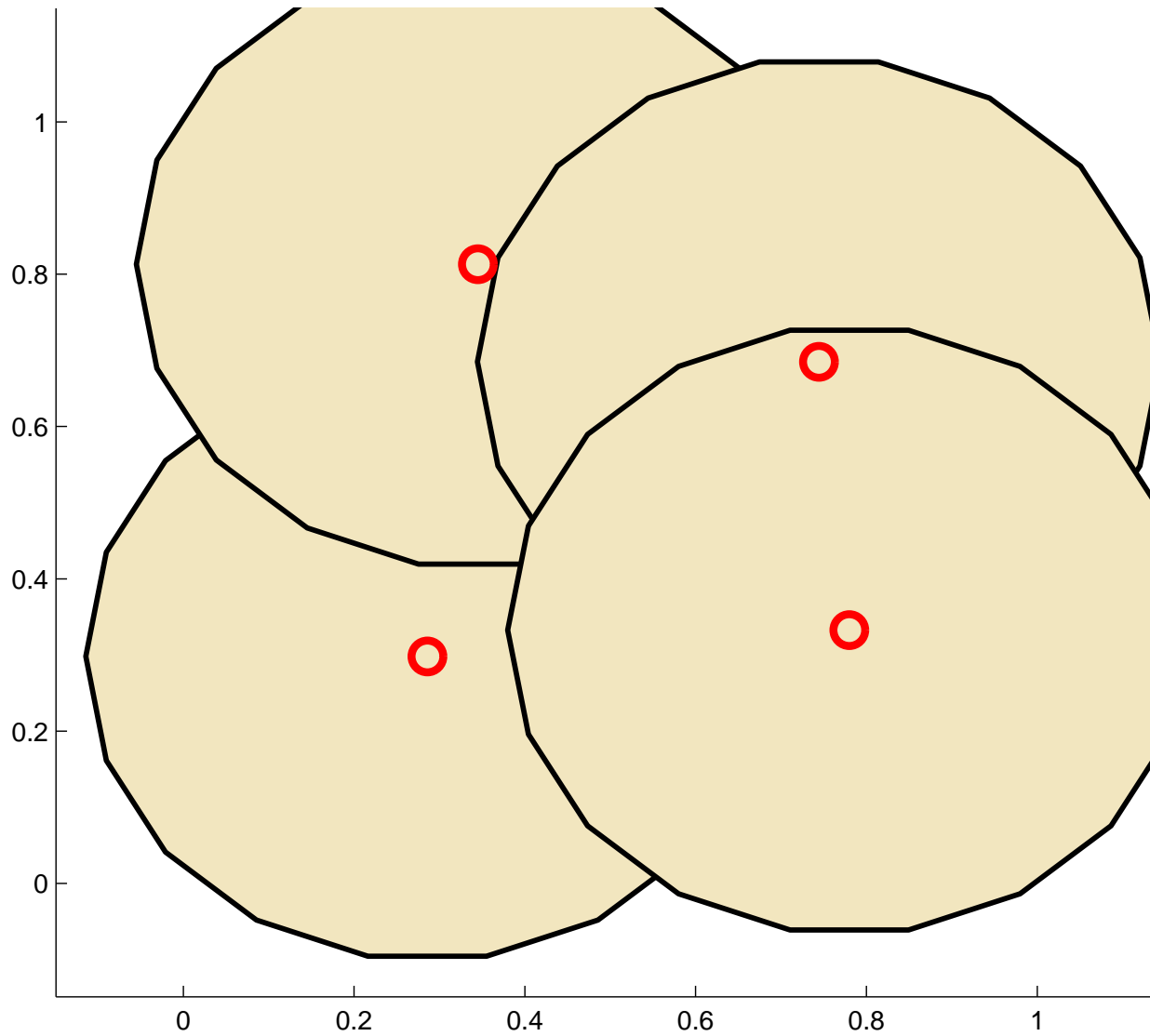
$R=0.2$   $M=12$



$R=0.3$   $M=8$



$R=0.4$   $M=4$



# Application to Block Coding of Images

---

- Popular use of conventional VQ

- Example  $384 \times 256$  image:



- Split into  $8 \times 8$  blocks
- $X$  comprises  $m = 1536$  examples of 192-dimensional vectors ( $64 \times 3$  colours)

Original Image (288KB)



LP-VQ reconstruction with  $R=200$ , 144KB (50%)



LP-VQ reconstruction with  $R=500$ , 33KB (12%)



LBG reconstruction, 33KB (12%)





## Image Statistics

---

Image	Size	Ratio	$R$	$M$	$E_{max}$	$E_{rms}$
Original	288KB	100%	0	1536	0	0
LP-VQ Reconstruction	144KB	50%	200	757	199.9	88.7
LP-VQ Reconstruction	33KB	12%	500	170	499.5	283.8
LBG Reconstruction	33KB	12%	-	170	816.4	229.8

## Discussion

---

- Complementary approach to standard VQ
- Useful where:
  - a ‘genuine’  $R$  exists
  - ‘outliers’ must be accurately coded
  - prototypes must be representative of data
  - as an initialiser for standard VQ
- Need not be a vector space as long as  $D[v, x_n]$  defined

# Kernel Machines Research

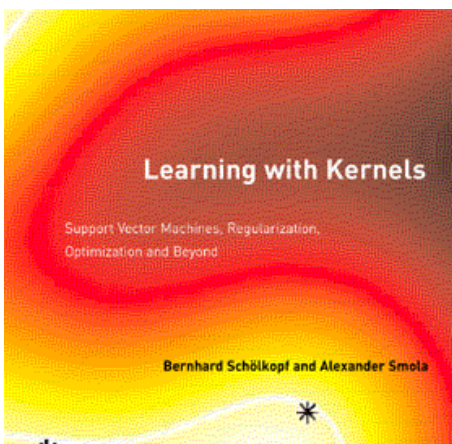
---

- algorithms/tasks: KDE, feature selection (*Weston et al., 2002*), multi-label-problems (*Elisseeff & Weston, 2001*), unlabelled data (*Szummer & Jaakkola, 2002, Zhou et al., 2004*), ICA [28], canonical correlations (*Bach & Jordan, 2002; Kuss, 2002*)
- optimization and implementation: QP, SDP (*Lanckriet et al., 2002*), online versions, ...
- theory of empirical inference: sharper capacity measures and bounds (*Bartlett, Bousquet, & Mendelson, 2002*), generalized evaluation spaces (*Mary & Canu, 2002*), ...
- kernel design
  - transformation invariances [13]
  - kernels for discrete objects [30, 78, 40, 18, 74]
  - kernels based on generative models [34, 61, 68]
  - local kernels [*e.g.*, 84]
  - complex kernels from simple ones [30, 2], global kernels from local ones [38]
  - functional calculus for kernel matrices [60]
  - model selection, e.g., via alignment [17]
  - kernels for dimensionality reduction [27]

## Conclusion

---

- crucial ingredients of SV algorithms: **kernels** that can be represented as dot products, and **large margin** regularizers
- kernels allow the formulation of a multitude of geometrical algorithms (Parzen windows, SVMs, kernel PCA,...)
- the choice of a kernel corresponds to
  - choosing a similarity measure for the data, or
  - choosing a (linear) representation of the data, or
  - choosing a hypothesis space for learning,and should reflect prior knowledge about the problem at hand.



For further information, cf.

<http://www.kernel-machines.org>,

<http://www.learning-with-kernels.org>.

---

## References

- [1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.
- [2] P. L. Bartlett and B. Schölkopf. Some kernels for structured data. Technical report, Biowulf Technologies, 2001.
- [3] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.
- [4] C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, New York, 1984.
- [5] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.
- [6] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.
- [7] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Müller, E. Säckinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th International Conference on Pattern Recognition and Neural Networks, Jerusalem*, pages 77–87. IEEE Computer Society Press, 1994.
- [8] L. Breiman. Bagging predictors. Technical Report 421, Department of Statistics, UC Berkeley, 1994. <ftp://ftp.stat.berkeley.edu/pub/tech-reports/421.ps.Z>.
- [9] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267, 2000.
- [10] C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, San Mateo, CA, 1996. Morgan Kaufmann.

- [11] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.
- [12] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5), 1999.
- [13] O. Chapelle and B. Schölkopf. Incorporating invariances in nonlinear SVMs. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.
- [14] S. Chen and C. J. Harris. Design of the optimal separating hyperplane for the decision feedback equalizer using support vector machines. In *IEEE International Conference on Acoustic, Speech, and Signal Processing*, Istanbul, Turkey, 2000.
- [15] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
- [16] D. Cox and F. O’Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Annals of Statistics*, 18:1676–1695, 1990.
- [17] N. Cristianini, A. Elisseeff, and J. Shawe-Taylor. On optimizing kernel alignment. Technical Report 2001-087, NeuroCOLT, 2001.
- [18] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK, 2000.
- [19] D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46:161–190, 2002. Also: Technical Report JPL-MLTR-00-1, Jet Propulsion Laboratory, Pasadena, CA, 2000.
- [20] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Applications of mathematics*. Springer, New York, 1996.
- [21] K. I. Diamantaras and S. Y. Kung. *Principal Component Neural Networks*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons, New York, 1996.
- [22] H. Drucker, B. Shahraray, and D. C. Gibbon. Relevance feedback using support vector machines. In *Proceedings of the 18th International Conference on Machine Learning*. Morgan Kaufmann, 2001.
- [23] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 171–203, Cambridge, MA, 2000. MIT Press.

- [24] T. S. Furey, N. Duffy, N. Cristianini, D. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.
- [25] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.
- [26] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.
- [27] J. Ham, D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of ICML*. 2004.
- [28] S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel feature spaces and nonlinear blind source separation. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. To appear.
- [29] T. J. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.
- [30] D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science Department, University of California at Santa Cruz, 1999.
- [31] M. A. Hearst, B. Schölkopf, S. Dumais, E. Osuna, and J. Platt. Trends and controversies — support vector machines. *IEEE Intelligent Systems*, 13:18–28, 1998.
- [32] I. A. Ibragimov and R. Z. Has’minskii. *Statistical Estimation — Asymptotic Theory*. Springer-Verlag, New York, 1981.
- [33] T. S. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7:95–114, 2000.
- [34] T. S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.
- [35] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of the European Conference on Machine Learning*, pages 137–142, Berlin, 1998. Springer.
- [36] G. S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41:495–502, 1970.

- [37] G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.
- [38] I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of ICML'2002*, 2002.
- [39] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.
- [40] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. Technical Report 2000-79, NeuroCOLT, 2000. Published in: T. K. Leen, T. G. Dietterich and V. Tresp (eds.), *Advances in Neural Information Processing Systems 13*, MIT Press, 2001.
- [41] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 133–165. Springer-Verlag, Berlin, 1998.
- [42] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A 209:415–446, 1909.
- [43] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, MIT A.I. Lab., 1996.
- [44] E. Osuna and F. Girosi. Reducing run-time complexity in SVMs. In *Proceedings of the 14th Int'l Conf. on Pattern Recognition, Brisbane, Australia*, 1998.
- [45] P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy. Gene functional classification from heterogeneous data. In *Proceedings of the Fifth International Conference on Computational Molecular Biology*, pages 242–248, 2001.
- [46] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.
- [47] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), September 1990.
- [48] S. Romdhani, S. Gong, and A. Psarrou. A multiview nonlinear active shape model using kernel PCA. In *Proceedings of BMVC*, pages 483–492, Nottingham, UK, 1999.
- [49] S. Romdhani, B. Schölkopf, P. Torr, and A. Blake. Fast face detection, using a sequential reduced support vector evaluation. TR 73, Microsoft Research, Redmond, WA, 2000. Published as: Computationally efficient face detection, *Proceedings of the International Conference on Computer Vision 2001*, pp. 695–700.



- [50] S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Longman Scientific & Technical, Harlow, England, 1988.
- [51] B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, München, 1997. Doktorarbeit, Technische Universität Berlin. Available from <http://www.kyb.tuebingen.mpg.de/~bs>.
- [52] B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings, First International Conference on Knowledge Discovery & Data Mining*, Menlo Park, 1995. AAAI Press.
- [53] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.
- [54] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.
- [55] B. Schölkopf, P. Simard, A. J. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 640–646, Cambridge, MA, 1998. MIT Press.
- [56] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- [57] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.
- [58] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.
- [59] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45:2758–2765, 1997.
- [60] B. Schölkopf, J. Weston, E. Eskin, C. Leslie, and W. S. Noble. A kernel approach for learning from almost orthogonal patterns. In *Proceedings of the 13th European Conference on Machine Learning (ECML'2002) and Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2002), Helsinki*, volume 2430/2431 of *Lecture Notes in Computer Science*, Berlin, 2002. Springer.
- [61] M. Seeger. Bayesian methods for support vector machines and Gaussian processes. Master's thesis, University of Edinburgh, Division of Informatics, 1999.

- [62] P. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5. Proceedings of the 1992 Conference*, pages 50–58, San Mateo, CA, 1993. Morgan Kaufmann.
- [63] A. J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.
- [64] M. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with ANOVA decomposition kernels. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 285–292, Cambridge, MA, 1999. MIT Press.
- [65] D. M. J. Tax and R. P. W. Duin. Data domain description by support vectors. In M. Verleysen, editor, *Proceedings ESANN*, pages 251–256, Brussels, 1999. D Facto.
- [66] M. Tipping and B. Schölkopf. A kernel approach for vector quantization with guaranteed distortion bounds. In T. Jaakkola and T. Richardson, editors, *Artificial Intelligence and Statistics*, pages 129–134, San Francisco, CA, 2001. Morgan Kaufmann.
- [67] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning*, San Francisco, California, 2000. Morgan Kaufmann.
- [68] K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.R. Müller. A new discriminative kernel from probabilistic models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.
- [69] V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).
- [70] V. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition [in Russian]*. Nauka, Moscow, 1974. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie-Verlag, Berlin, 1979).
- [71] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.
- [72] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.
- [73] V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

- [74] J.-P. Vert. A tree kernel to analyze phylogenetic profiles. In *Proceedings of ISMB'02*, 2002.
- [75] U. von Luxburg, O. Bousquet, and B. Schölkopf. A compression approach to support vector model selection. Technical report, Max Planck Institute for Biological Cybernetics, 2002. To appear in JMLR, 2004.
- [76] G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1990.
- [77] M. K. Warmuth, G. Rätsch, M. Mathieson, J. Liao, and C. Lemmen. Active learning in the drug discovery process. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. To appear.
- [78] C. Watkins. Dynamic alignment kernels. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press.
- [79] H. L. Weinert, editor. *Reproducing Kernel Hilbert Spaces — Applications in Statistical Signal Processing*. Hutchinson Ross, Stroudsburg, PA, 1982.
- [80] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. Technical Report 98, Max Planck Institute for Biological Cybernetics, 2002.
- [81] C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*. Kluwer, 1998.
- [82] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.
- [83] C.-H. Yeang, S. Ramaswamy, P. Tamayo, S. Mukherjee, R. M. Rifkin, M. Angelo, M. Reich, E. Lander, J. Mesirov, and T. Golub. Molecular classification of multiple tumor types. *Bioinformatics*, 17:S316–S322, 2001. ISMB'01 Supplement.
- [84] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.