Support Vector Learning

Bernhard Schölkopf

GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany

bs@first.gmd.de
http://www.first.gmd.de/∼bs

Acknowledgements

Alexander Smola, Klaus-Robert Müller, Sebastian Mika
GMD First, Berlin

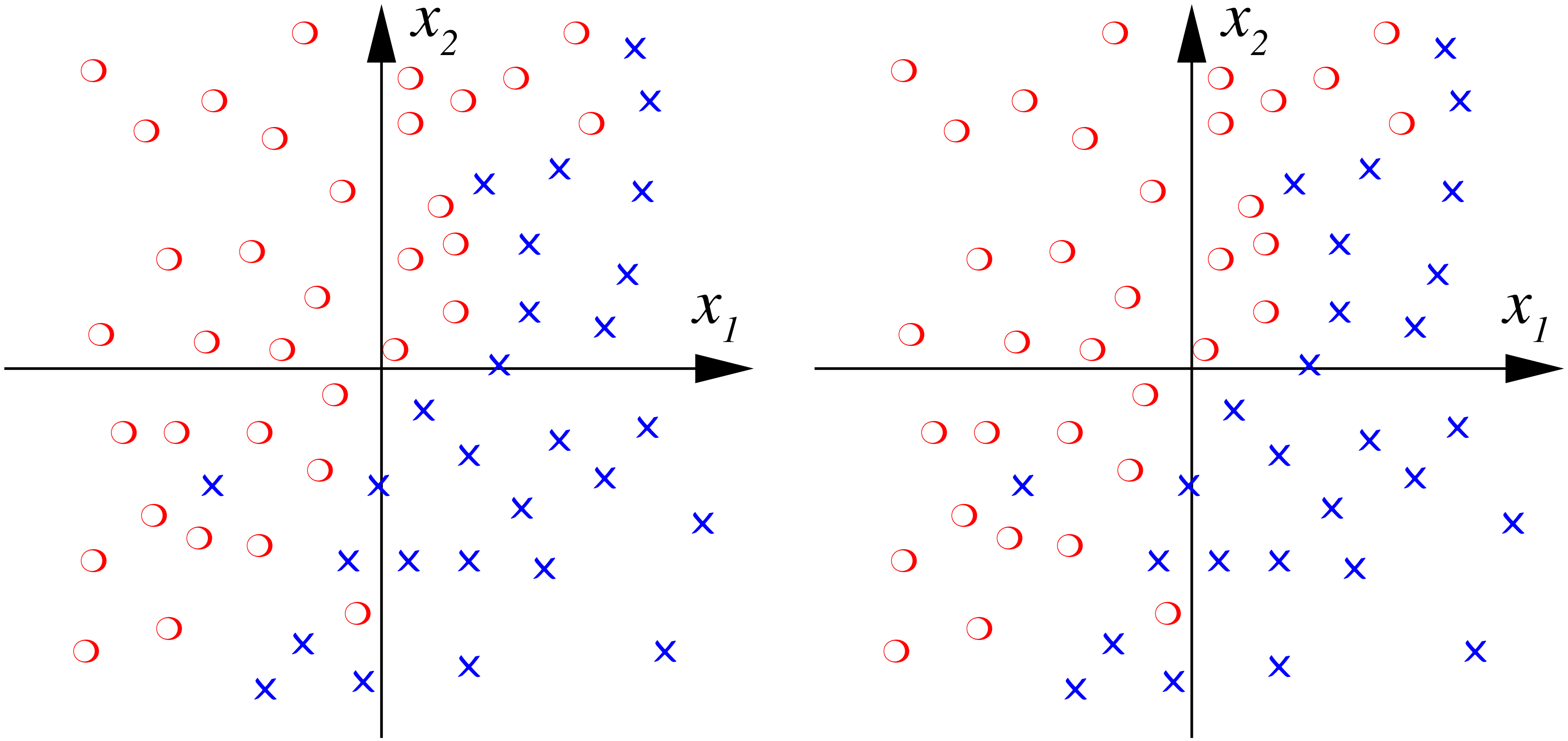
Vladimir Vapnik, Patrice Simard
AT&T, Red Bank, USA

Chris Burges
Bell Labs, Holmdel, USA

Roadmap

- Basics
  - learning from examples, risk minimization
  - induction principles, capacity concepts
  - nonlinear feature spaces, reproducing kernels
- Support Vector Pattern Recognition
  - optimal hyperplane classifiers
  - nonseparable and nonlinear classification problems
  - applications, methods for improvement
- Other Kernel Algorithms
  - support vector regression
  - regression applications
  - kernel PCA

Pattern Recognition



B. Schölkopf, 7/1998

Pattern Recognition, II

Statistical Learning Theory:

- minimal training error with minimal model *capacity*
- capacity measure: e.g. VC(Vapnik-Chervonenkis)-dimension

Pattern Recognition, III

Learn f from examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell) \in \mathbf{R}^N \times \{\pm 1\}, \quad \text{generated from } P(\mathbf{x}, y),$$

such that the expected number of errors on a test set, also drawn from $P(\mathbf{x}, y)$,

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

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

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

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

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

Risk minimization *(Vapnik, 1979)*

- Regression estimation. RM: minimize

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

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

ERM gives least mean squares: minimize

$$\sum_i (f(\mathbf{x}_i) - y_i)^2$$

- Density estimation. RM: minimize

$$R[f] = \int (-\log p(\mathbf{x})) \, dP(\mathbf{x})$$

ERM gives maximum likelihood estimation: maximize

$$\sum_i \log p(\mathbf{x}_i) = \log \left(\prod_i p(\mathbf{x}_i) \right)$$

Convergence of Means to Expectations

Law of large numbers:

$$R_{emp}[f] \to R[f]$$

as $\ell \to \infty$.

Does this imply that for the function f^* minimizing R_{emp}, and the function f^{**} minimizing R, we have

$$R_{emp}[f^*] \to R[f^{**}], \quad R[f^*] \to R[f^{**}]$$

as $\ell \to \infty$ (classical definition of *"consistency"* of ERM: for $\ell \to \infty$, ERM should lead to the same result as RM)?

No! Need a *uniform* version of the law of large numbers.

The Importance of the Set of Functions

Recall: need uniform convergence over all functions that the learning machine can implement.
What about taking *all* functions from $\mathbf{R}^N$ to $\{\pm 1\}$?

Training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell) \in \mathbf{R}^N \times \{\pm 1\}$
Test patterns $\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_{\bar{\ell}} \in \mathbf{R}^N$,
such that $\{\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_{\bar{\ell}}\} \cap \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\} = \{\}$.

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. $\longrightarrow$ the set of functions must be restricted

Vapnik-Chervonenkis(VC)-Theory

Necessary and sufficient conditions for consistency of empirical risk minimization (ERM):
One-sided convergence, uniformly over all functions that can be implemented by the learning machine.

$$\lim_{\ell\to\infty} P\{\sup_f (R[f] - R_{emp}[f]) > \epsilon\} = 0$$

for all $\epsilon > 0$.

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

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

Pattern Recognition Capacity Concepts

- *VC entropy*: on an example $(\mathbf{x}, y)$, f_α causes a loss $Q(\mathbf{x}, y, \alpha)$. On a training set, different values of $\alpha \in \Lambda$ lead to N^Λ different loss vectors $q_\alpha = (Q(\mathbf{x}_1, y_1, \alpha), \ldots, Q(\mathbf{x}_\ell, y_\ell, \alpha))$. Define
$$H^\Lambda(\ell) = E \log N^\Lambda.$$
$H^\Lambda(\ell)/\ell \to 0 \iff$ uniform convergence
- exchange 'E' and 'log': *annealed entropy*.
$H^\Lambda_{ann}(\ell)/\ell \to 0 \Longrightarrow$ exponential convergence
$$P\{\sup_\alpha(R(\alpha) - R_{emp}(\alpha)) > \epsilon\} \leq 4 \cdot \exp(((H^\Lambda_{ann}(2\ell)/\ell) - \epsilon^2) \cdot \ell).$$
- take 'max' instead of 'E': *growth function*.
$G^\Lambda(\ell)/\ell \to 0 \iff$ exponential convergence for all underlying distributions

Proof Ingredients

One function: Hoeffding's inequality states that for the probability of error $R(\alpha)$ of some function f_α,

$$P\{|R(\alpha) - R_{emp}(\alpha)| > \epsilon\} < 2\exp(-2\epsilon^2\ell).$$

Finitely many functions: for all $(f_\alpha)_{\alpha\in\Lambda}$, simultaneously fulfil $|R(\alpha) - R_{emp}(\alpha)| \leq \epsilon$. This yields

$$P\{\sup_\alpha |R(\alpha) - R_{emp}(\alpha)| > \epsilon\} < 2|\Lambda|\exp(-2\epsilon^2\ell).$$

Infinitely many functions: if the capacity of the set of functions is small ($H(\ell)/\ell \to 0$), then asymptotically, the "effective" number of functions *on a set of points of given size* ℓ is finite.

Structure of the Growth Function

Either $G^{\Lambda}(\ell) = \ell \cdot \ln(2)$ — this means that for any sample size ℓ the points can be chosen such that by using functions of the learning machine, they can be separated in all 2^{ℓ} possible ways (*"shattered"*).

Or there exists some *maximal* ℓ for which the above is possible. Call this number the *VC-dimension*, and denote it by h. Then one can prove that for $\ell > h$,

$$G^{\Lambda}(\ell) \leq h \left(\log \frac{\ell}{h} + 1 \right).$$

Nothing "in between" linear growth and logarithmic growth is possible.

VC-Dimension: Example

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

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

- Clearly, we can shatter three non-collinear points.
- But we can never shatter four points.
- Hence the VC dimension is $h = 3$

A Typical Bound for Pattern Recognition

For any $\alpha \in \Lambda$ and $\ell > h$, with a probability of at least $1 - \eta$,

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

holds, where the *confidence term* ϕ is defined as

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

(Derivation: in uniform convergence bounds, set RHS $= \eta$, and solve for ϵ to get the confidence term.)

- does this mean, that we can learn *any*thing?
- 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!

Examples of Induction Principles

- *Empirical risk minimization:* minimize

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

- *Minimum description length:* minimize some measure of the description length of the sequence $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)$ by a function f.

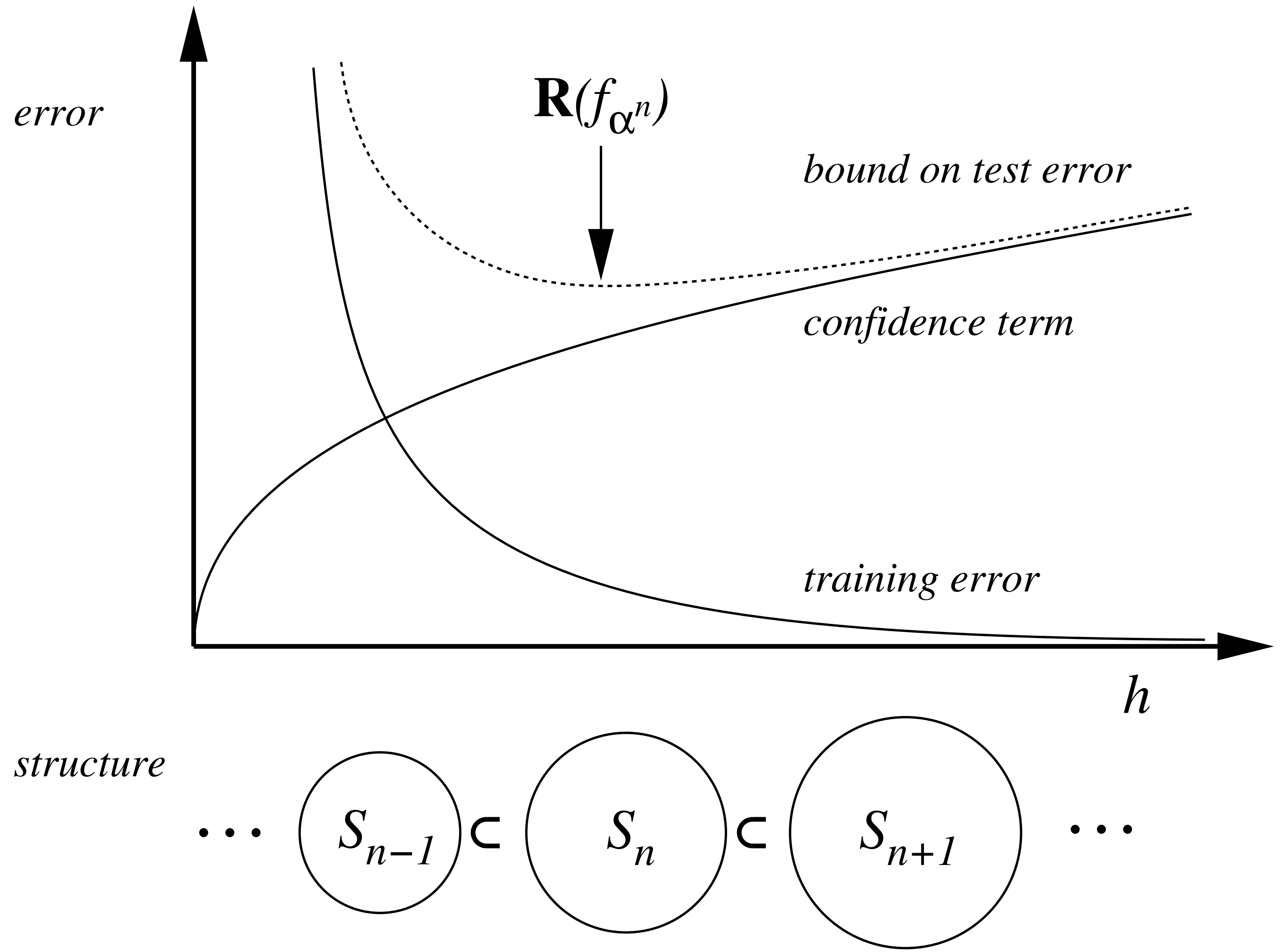
- *Structural risk minimization (SRM) (Vapnik, 1979):* minimize the RHS of

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

  To this end, introduce a structure on $(f_\alpha)_{\alpha \in \Lambda}$.

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

SRM: The Picture



Feature Spaces

Learning f requires small training error *and* small capacity of the set $\{f_\alpha\}$.

The Support Vector (SV) approach: *preprocess* the data with

$$\begin{aligned} \Phi : \mathbf{R}^N &\to F \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}) \end{aligned}$$

$$\text{where } N \ll \dim(F).$$

to get data

$$(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_\ell), y_\ell) \in F \times \{\pm 1\}.$$

Learn $\tilde{f}$ to construct

$$f = \tilde{f} \circ \Phi$$

The "Curse of Dimensionality"

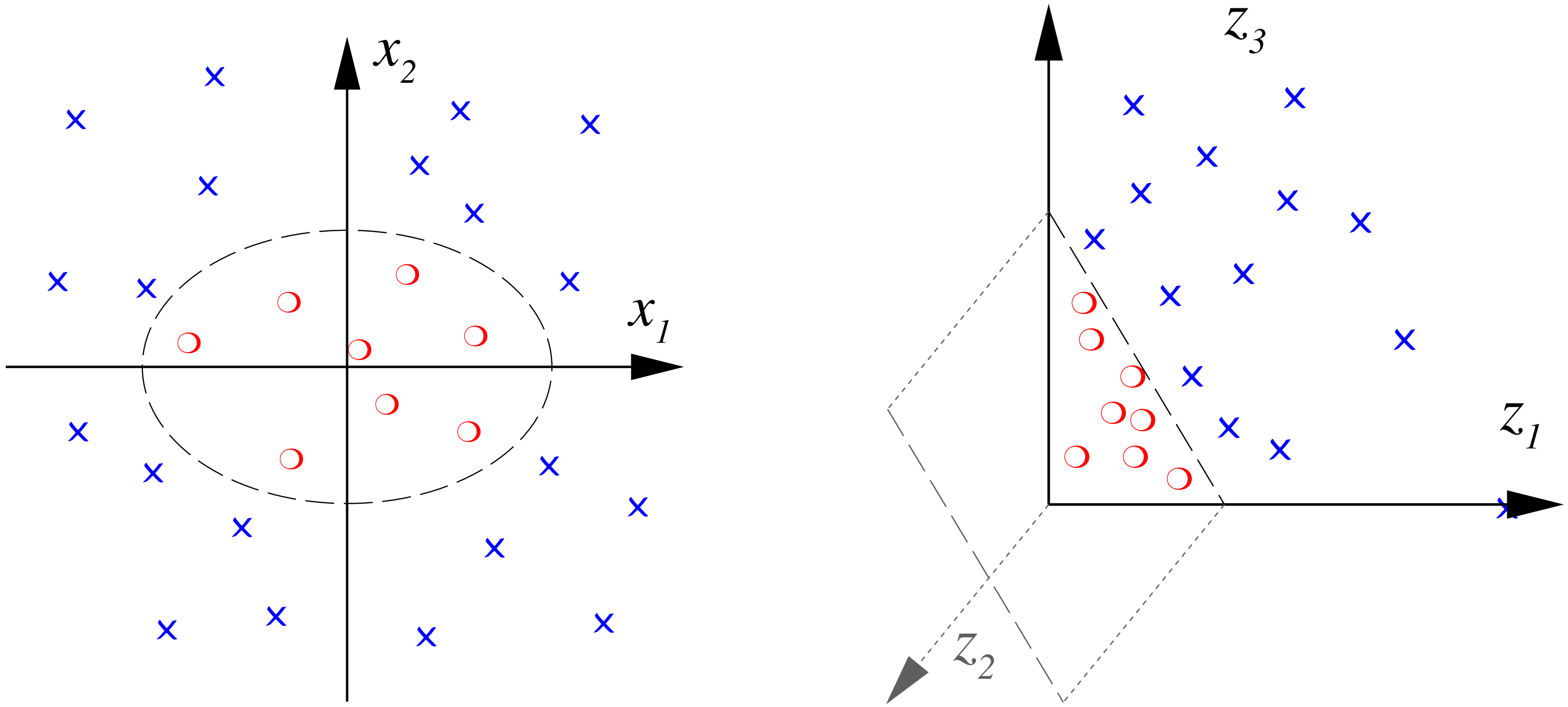
- classical statistics: **harder**, as the data are high-dimensional
- SV-Learning: (in some cases) **simpler**:

If Φ is chosen such that $\{\tilde{f}\}$ allows small training error *and* has low capacity, then we can guarantee good generalization.

The *capacity* matters, not the *dimensionality* of the space.

Example: All Degree 2 Monomials

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



B. Schölkopf, 7/1998

Example, Part II

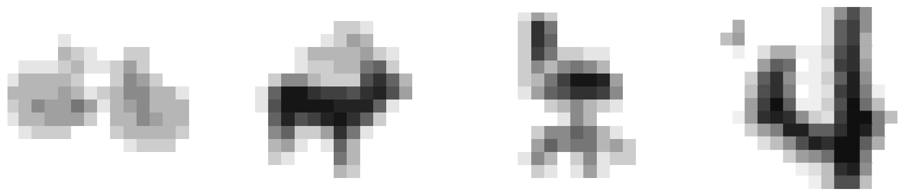
Use

$$f(\mathbf{z}) = \mathrm{sgn}((\mathbf{w} \cdot \mathbf{z}) - 1).$$

(In the example: $\mathbf{w} = ((0.6)^{-2}, 0, (0.4)^{-2})^\top)$

"Simple" (low-capacity) decision function (hyperplane), which allows computation of the VC-dimension

Product Feature Space



Patterns $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbf{R}^N$ where most information is contained in products between entries x_i, e.g. in $x_1 \cdot x_2$.
$\longrightarrow$ work in the space F of all products of d entries (e.g. pixels).

Number of products of d pixels in N-dimensional images:

$$\frac{(N+d-1)!}{d!(N-1)!}$$

— grows like N^d.

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

Problem: Dimensionality of F

More extreme: there might be cases where $\dim(F)=\infty$.

- Explicitly carrying out Φ is not feasible.

Possible Solutions:

- incorporate Φ somehow into the algorithm — an algorithm in an infinite-dimensional space?
- preprocessing: select only certain directions from the image of Φ — how can one select directions from infinitely many?

The Kernel Trick *(Boser, Guyon, and Vapnik, 1992)*

$$
\begin{aligned}
(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) &= (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)(y_1^2, \sqrt{2}\, y_1 y_2, y_2^2)^\top \\
&= (\mathbf{x} \cdot \mathbf{y})^2 \\
&= : k(\mathbf{x}, \mathbf{y})
\end{aligned}
$$

$\longrightarrow$ the dot product in F can be computed in $\mathbf{R}^2$

The Kernel Trick, II

More general: $\mathbf{x}, \mathbf{y} \in \mathbf{R}^N$, $d \in \mathbf{N}$:

$$
\begin{aligned}
(\mathbf{x} \cdot \mathbf{y})^d &= \left(\sum_{j=1}^{N} x_j \cdot y_j \right)^d \\
&= \sum_{j_1, \ldots, j_d = 1}^{N} x_{j_1} \cdot \ldots \cdot x_{j_d} \cdot y_{j_1} \cdot \ldots \cdot y_{j_d} = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})),
\end{aligned}
$$

where Φ maps into the space spanned by all ordered products of d input directions

The Kernel Trick, III (Mercer's Theorem)

Still more general: If k is a continuous kernel of a positive integral operator on $L_2(\mathcal{C})$ (where $\mathcal{C}$ is some compact space),

$$\int f(\mathbf{x})k(\mathbf{x},\mathbf{y})f(\mathbf{y})\,d\mathbf{x}\,d\mathbf{y} \geq 0,$$

it can be expanded as

$$k(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N_F} \lambda_i \psi_i(\mathbf{x})\psi_i(\mathbf{y})$$

with $\lambda_i > 0$, and $N_F \in \mathbf{N}$ or $N_F = \infty$.
In that case

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

satisfies $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) = k(\mathbf{x},\mathbf{y})$.

The Kernel Trick, IV

Examples of common kernels:

$$\begin{aligned} \text{Polynomial} \quad k(\mathbf{x}, \mathbf{y}) &= (\mathbf{x} \cdot \mathbf{y} + c)^d \\ \text{Sigmoid} \quad k(\mathbf{x}, \mathbf{y}) &= \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \Theta) \\ \text{RBF} \quad k(\mathbf{x}, \mathbf{y}) &= \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\,\sigma^2)\right) \end{aligned}$$

Note: if the scaling of the data is such that $\mathbf{x}, \mathbf{y} \in [-1, 1]^N$, then it usually is a good idea to divide $(\mathbf{x} \cdot \mathbf{y})$ and $\|\mathbf{x} - \mathbf{y}\|^2$ by N.

The Kernel Trick, V

Possible uses:

in any algorithm that only depends on dot products!

- SV machines
- Kernel PCA

Another Representation of F

$$\tilde{\Phi} : \mathbf{R}^N \longrightarrow \mathcal{H}, \qquad \mathbf{x} \mapsto k(\mathbf{x}, .)$$

Need a dot product $\langle ., . \rangle$ for $\mathcal{H}$ such that

$$\langle \tilde{\Phi}(\mathbf{x}), \tilde{\Phi}(\mathbf{y}) \rangle = k(\mathbf{x}, \mathbf{y}),$$

i.e. require

$$\langle k(\mathbf{x}, .), k(\mathbf{y}, .) \rangle = k(\mathbf{x}, \mathbf{y}).$$

For a Mercer kernel $k(\mathbf{x}, \mathbf{y}) = \Sigma_j \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{y})$, with $\lambda_i > 0$ for all i, and $(\psi_i \cdot \psi_j)_{L_2(\mathcal{C})} = \delta_{ij}$, this can be achieved by choosing $\langle ., . \rangle$ such that

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

F as a Reproducing Kernel Hilbert Space

To see this, compute

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

$\mathcal{H}$, the closure of the space of all functions

$$f(\mathbf{x}) = \sum_i a_i k(\mathbf{x}, \mathbf{x}_i),$$

with the dot product $\langle ., .\rangle$, is called a reproducing kernel Hilbert space.

The Connection Between F and $\mathcal{H}$

Expand f, g in the ONB $(\sqrt{\lambda_i}\psi_i)_i$, using coordinate vectors $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$:

$$f(\mathbf{x}) = (\boldsymbol{\alpha} \cdot \Phi(\mathbf{x})) \qquad g(\mathbf{x}) = (\boldsymbol{\beta} \cdot \Phi(\mathbf{x}))$$

Then the dot product in F equals that of the RKHS:

$$(\boldsymbol{\alpha} \cdot \boldsymbol{\beta}) = \langle f, g \rangle$$

In particular, if $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are images of some points $\mathbf{s}, \mathbf{t}$ under Φ, there is a simple way to compute the dot product...

$$(\boldsymbol{\alpha} \cdot \boldsymbol{\beta}) = (\Phi(\mathbf{s}) \cdot \Phi(\mathbf{t})) = k(\mathbf{s}, \mathbf{t}).$$

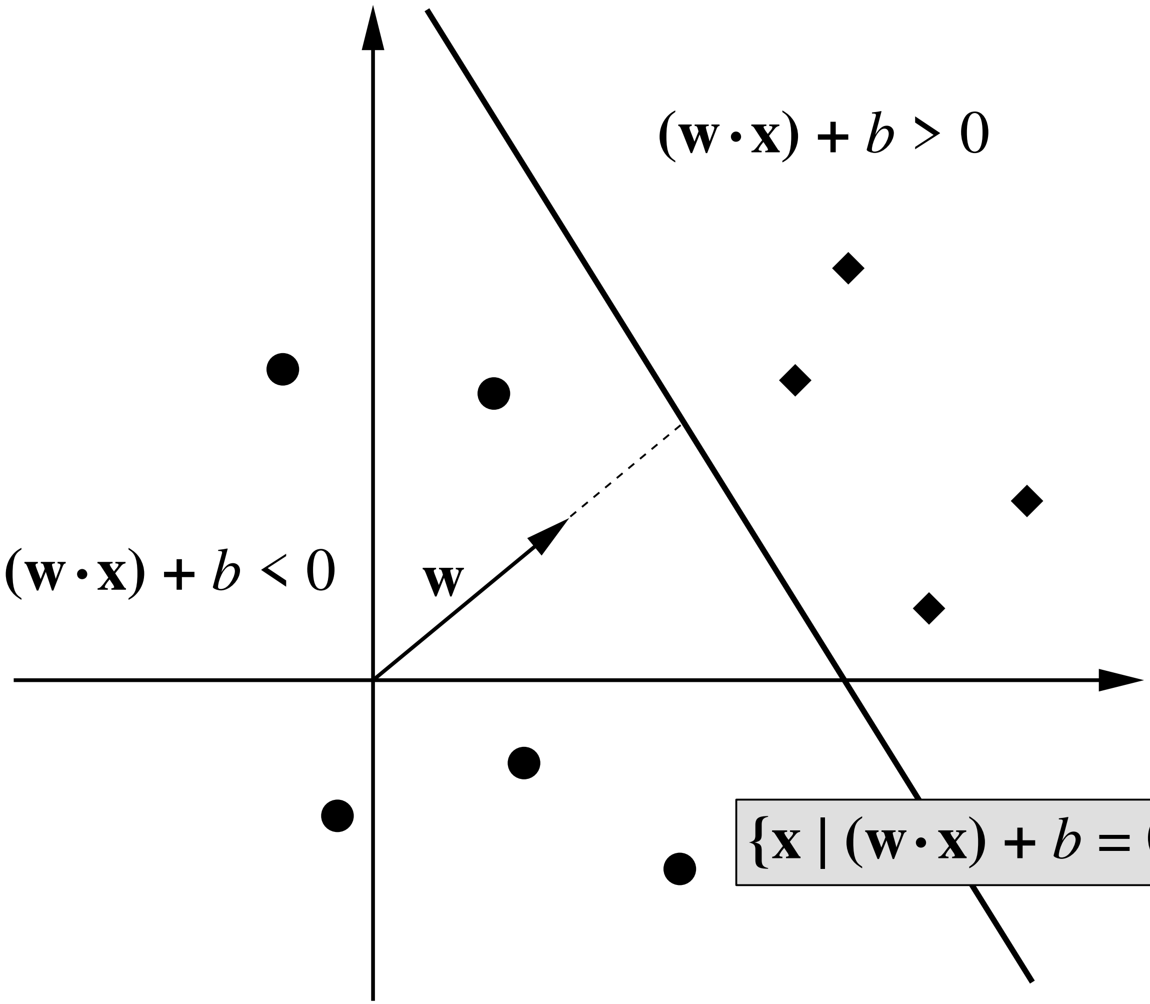
Note that in this case,

$$f(\mathbf{x}) = k(\mathbf{s}, \mathbf{x}) \qquad g(\mathbf{x}) = k(\mathbf{t}, \mathbf{x})$$

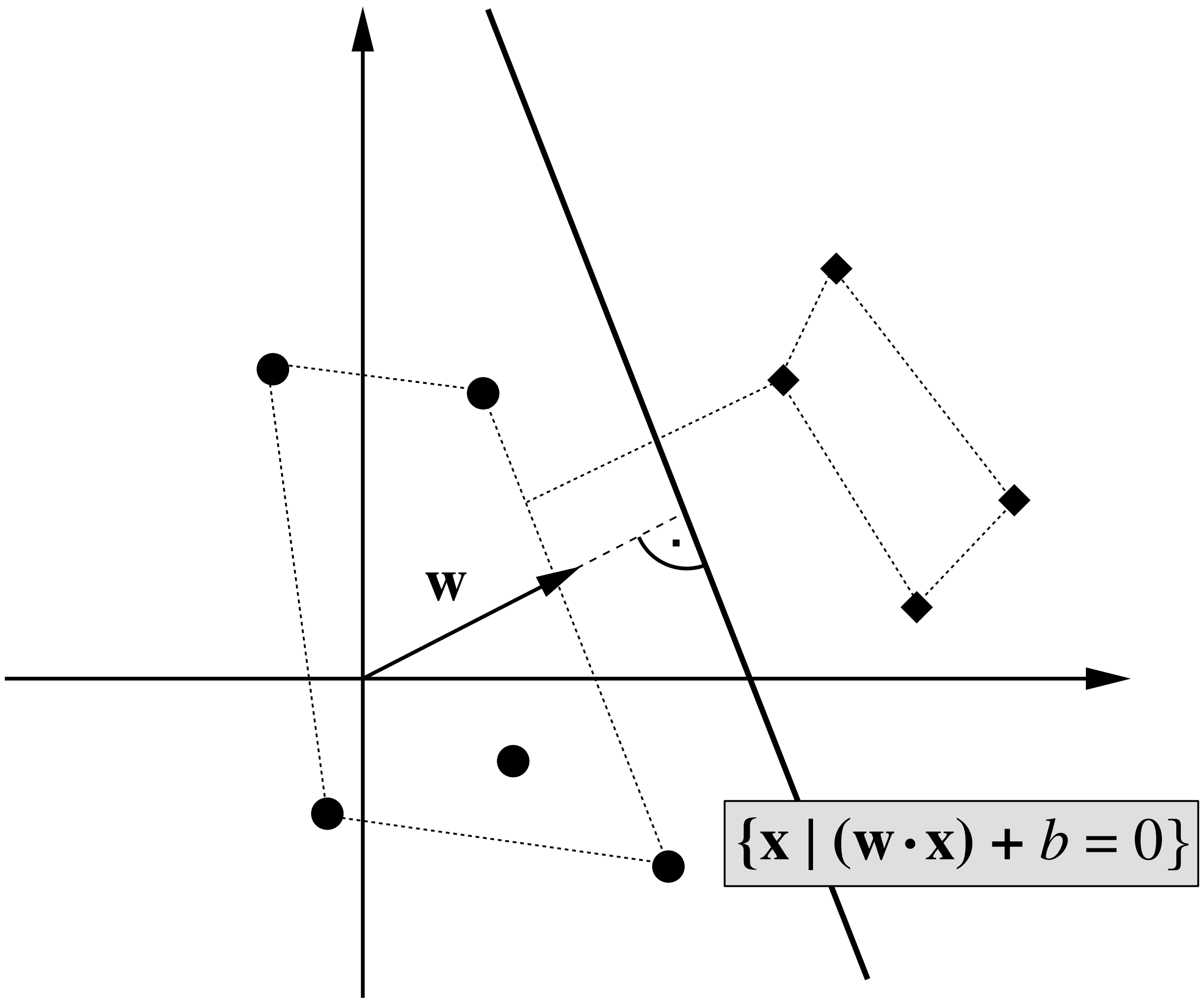
Support Vector Classifiers: Overview

- optimal hyperplane classifiers
  - have maximal "margin" and minimal VC-dimension
  - can be constructed by solving a quadratic program
  - are defined by a small subset of the training set ("SVs")
- generalization to nonseparable problems
- generalization to nonlinear problems using kernels
- applications
  - comparison to classical RBF networks; OCR results
  - methods for incorporating domain knowledge
  - methods for speeding up the SV expansion evaluation
  - training methods

Separating Hyperplane



Optimal Separating Hyperplane *(Vapnik and Lerner, 1963)*



Eliminating the Scaling Freedom *(Vapnik, 1995)*

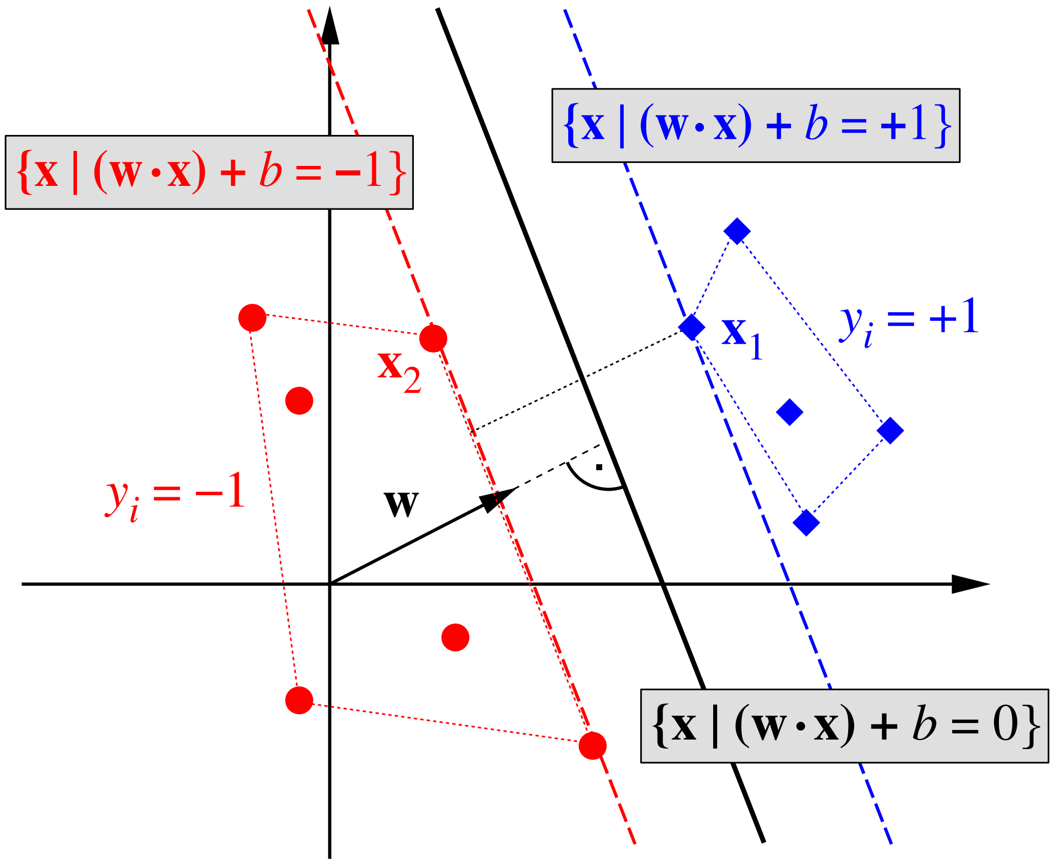
Note: if $c \neq 0$, then

$$\{\mathbf{x} | (\mathbf{w} \cdot \mathbf{x}) + b = 0\} = \{\mathbf{x} | (c\mathbf{w} \cdot \mathbf{x}) + cb = 0\}.$$

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

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

Canonical Optimal Hyperplane



Note:

$$(\mathbf{w}\cdot\mathbf{x}_1) + b = +1$$
$$(\mathbf{w}\cdot\mathbf{x}_2) + b = -1$$

$$\Rightarrow \quad (\mathbf{w}\cdot(\mathbf{x}_1-\mathbf{x}_2)) = 2$$

$$\Rightarrow \quad \left(\frac{\mathbf{w}}{\|\mathbf{w}\|}\cdot(\mathbf{x}_1-\mathbf{x}_2)\right) = \frac{2}{\|\mathbf{w}\|}$$

VC Dimension of Hyperplane Decision Functions

Theorem (Vapnik, 1995): Assume that the hyperplane $(\mathbf{w} \cdot \mathbf{x}) + b = 0$ is in canonical form w.r.t. $X^* = \{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$. The set of decision functions $f(\mathbf{x}, \mathbf{w}, b) = \mathrm{sign}\{(\mathbf{x} \cdot \mathbf{w}) + b\}$ defined on X^* and satisfying the constraint $\|\mathbf{w}\| \leq A$ has a VC–dimension satisfying

$$h \leq [R^2 A^2] + 1.$$

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

- control h by varying A.
- independent of the dimensionality of the space
- caveat: structure in SRM should be defined a priori (Shawe-Taylor et al., 1996)

Formulation as an Optimization Problem

Hyperplane with maximum margin: minimize

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

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

$$y_i \cdot [(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 \quad \text{for } i = 1 \ldots \ell$$

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

Lagrange Function

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

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i \left(y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1\right).$$

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

- if a constraint is violated, then $y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 < 0 \longrightarrow$
  - α_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 $y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 > 0$, then $\alpha_i = 0$: otherwise, L could be increased by decreasing α_i *(KKT conditions)* (e.g. Bertsekas, 1995)

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.

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

and

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

Substitute both into L to get the *dual problem*

The Support Vector Expansion

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

where for all $i = 1, \ldots, \ell$ either

$y_i \cdot [(\mathbf{w} \cdot \mathbf{x}_i) + b] > 1 \quad \Longrightarrow \alpha_i = 0 \longrightarrow \mathbf{x}_i$ irrelevant

or

$y_i \cdot [(\mathbf{w} \cdot \mathbf{x}_i) + b] = 1$ (*on* the margin) $\longrightarrow \mathbf{x}_i$ "Support Vector"

The solution is determined by the examples on the margin.

Thus

$$\begin{aligned} f(\mathbf{x}) &= \operatorname{sgn}\left((\mathbf{x} \cdot \mathbf{w}) + b\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{\ell} \alpha_i y_i (\mathbf{x} \cdot \mathbf{x}_i) + b\right). \end{aligned}$$

Why it is Good to Have Few SVs

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

Theorem (Vapnik and Chervonenkis, 1974): Denote $\#\mathrm{SV}(\ell)$ the number of SVs obtained by training on ℓ examples randomly drawn from $P(\mathbf{x}, y)$, and E the expectation. Then

$$E(\mathrm{Prob}(\text{test error})) \leq \frac{E(\#\mathrm{SV}(\ell))}{\ell - 1}$$

A Mechanical Interpretation *(Burges and Schölkopf, 1997)*

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

Then the solution is mechanically stable:

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

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

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

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

subject to

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

Both the final decision function and the function to be maximized are expressed in dot products $\longrightarrow$ can use kernels.

Nonseparable Problems *(Cortes and Vapnik, 1995)*

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

Modify the constraint to

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

with

$$\xi_i \geq 0$$

("*soft margin*") and minimize

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

for some $C > 0$.

The Soft Margin Dual

Write down a Lagrangian, substitute the conditions that

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

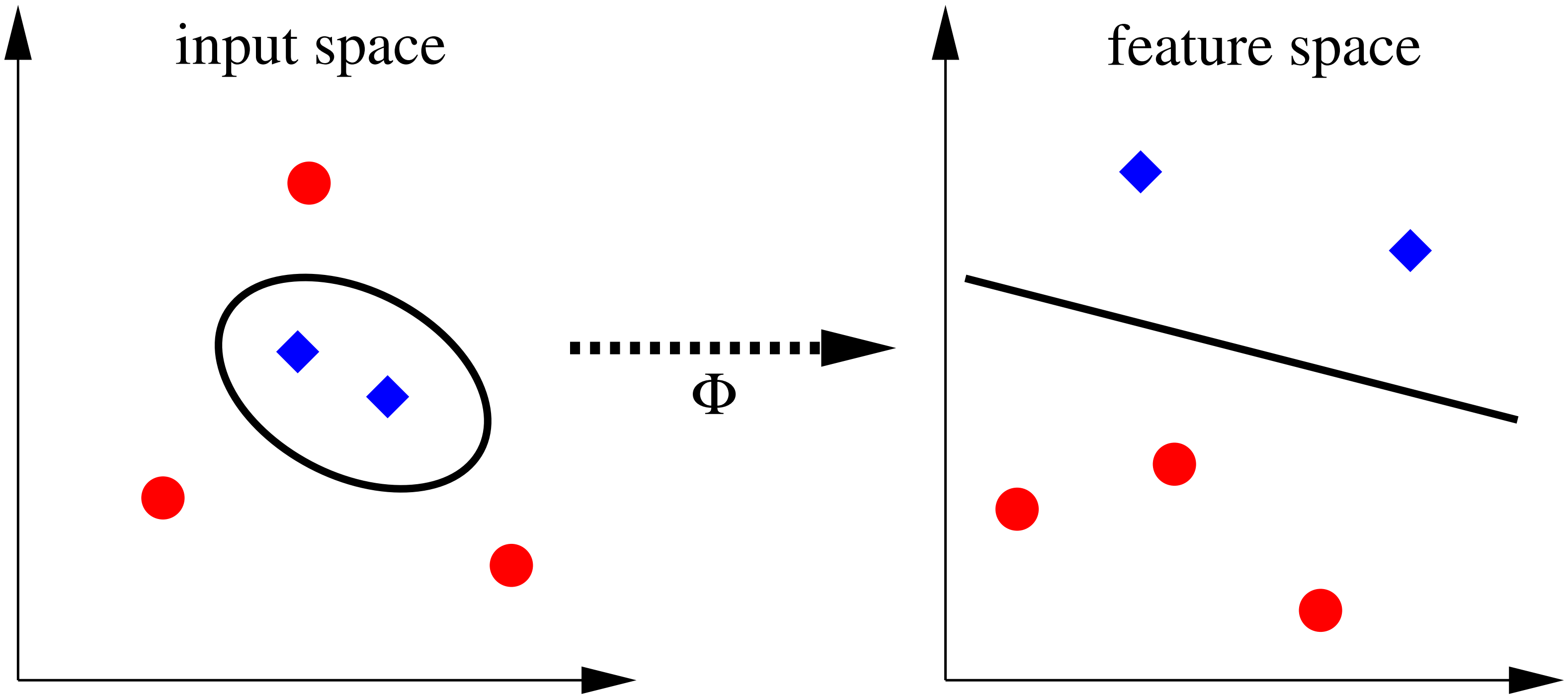
$\longrightarrow$ Maximize

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

subject to

$$C \geq \alpha_i \geq 0, \quad i = 1, \ldots, \ell, \quad \text{and} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0.$$

Nonlinear SV Machines: the Idea



Nonlinear SV Machines *(Boser, Guyon, and Vapnik, 1992)*

Maximize

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

subject to

$$C \geq \alpha_i \geq 0, \quad i = 1, \ldots, \ell, \quad \text{and} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0.$$

Decision function:

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

Using kernels, we solve the classification problem in the high-dimensional space F.

Computing the Offset b

For all SVs (i.e. $\alpha_i > 0$), the constraint becomes an equality:

$$y_i \cdot \left(\sum_{j=1} y_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) + b \right) = 1 - \xi_i.$$

If moreover $\alpha_i < C$, we have (again by KKT) $\xi_i = 0$, hence

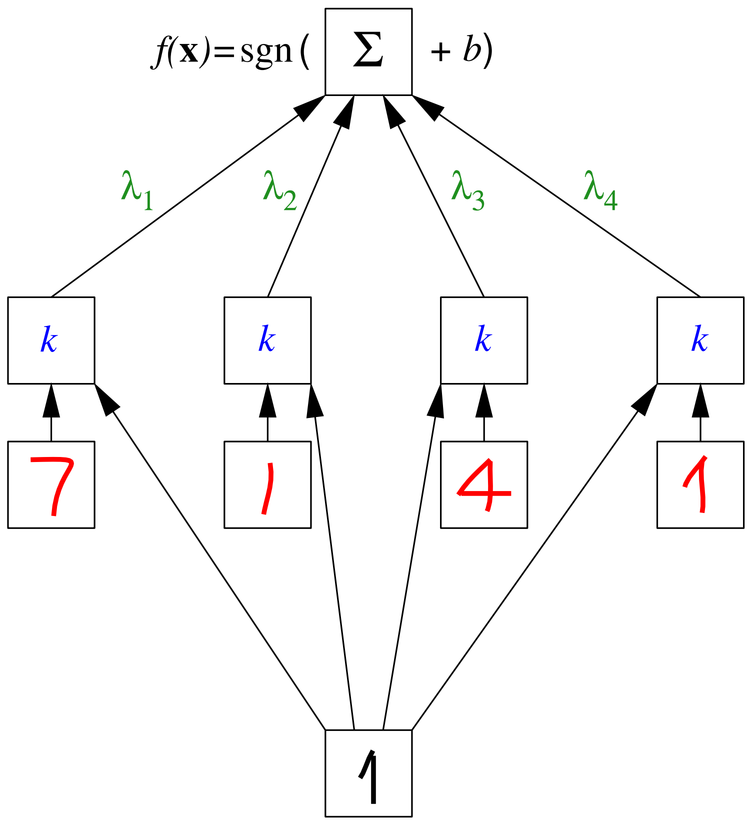
$$y_i \cdot \left(\sum_{j=1} y_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) + b \right) = 1,$$

hence

$$b = y_i - \sum_{j=1}^{\ell} y_j \alpha_j k(\mathbf{x}_i, \mathbf{x}_j).$$

In practice, one can average over all examples with multipliers satisfying $0 < \alpha_i < C$.
Alternatively, one can compute b from the corresponding double-dual variable.

The Support Vector Machine



classification

$f(\mathbf{x}) = \mathrm{sgn}\,(\Sigma\, \lambda_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b)$

weights

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

$k(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)^{\mathrm{d}}$

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

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

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

input vector $\mathbf{x}$

Multi-Class Problems

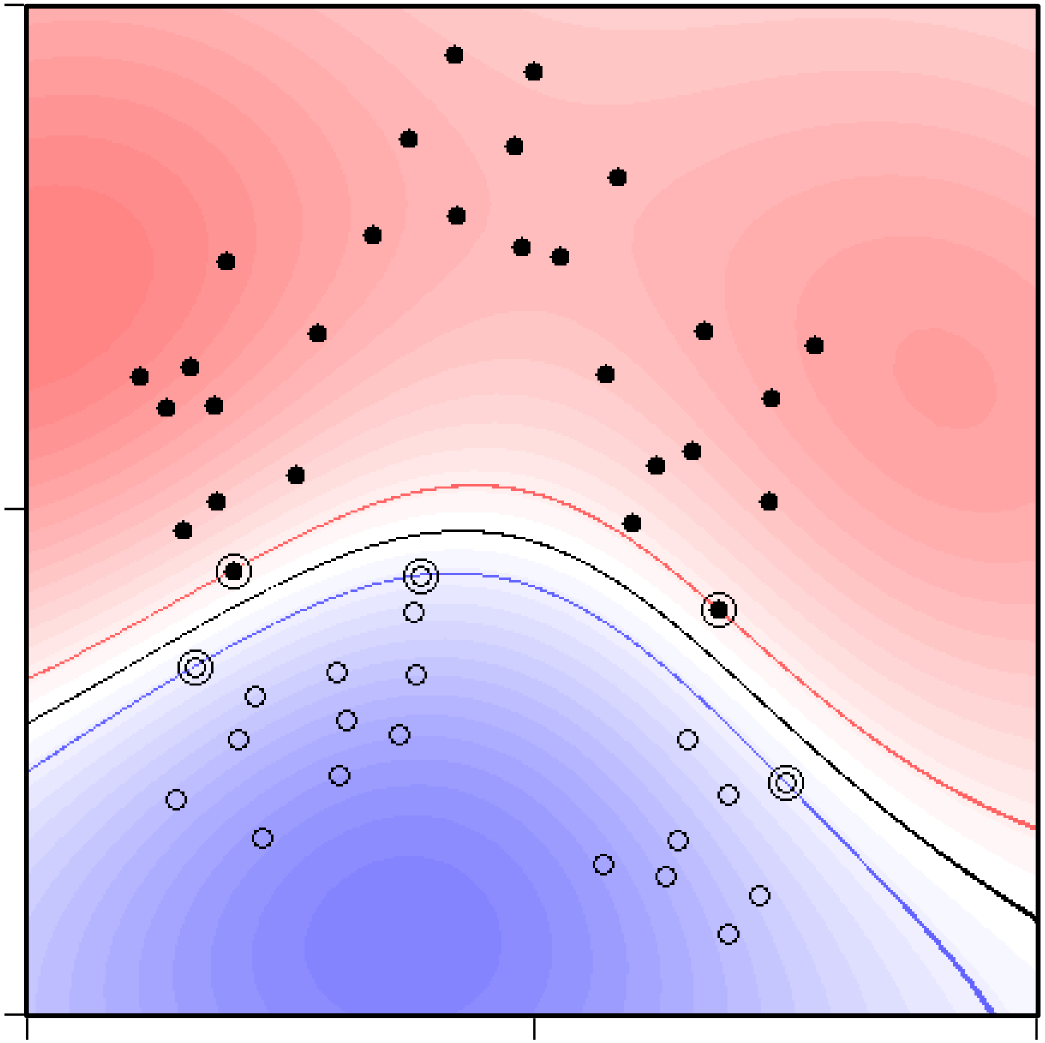
Simple approach: for n classes, train n binary recognizers $f^1, \ldots, f^n$, each to separate one class from the rest.
Combine them to give the n-class output

$$\operatorname{argmax}_{j=1,\ldots,n} g^j(\mathbf{x}), \quad \text{where } g^j(\mathbf{x}) = \sum_{i=1}^{\ell} y_i \alpha_i^j \cdot k(\mathbf{x}, \mathbf{x}_i) + b^j$$

The values $g^j(\mathbf{x})$ can also be used for reject decisions, for instance by considering the difference between the maximum and the second highest value as a measure of confidence.

More sophisticated methods for directly training multi-class SVMs are being researched (e.g. Weston and Watkins, 1998; Bennett, 1998).

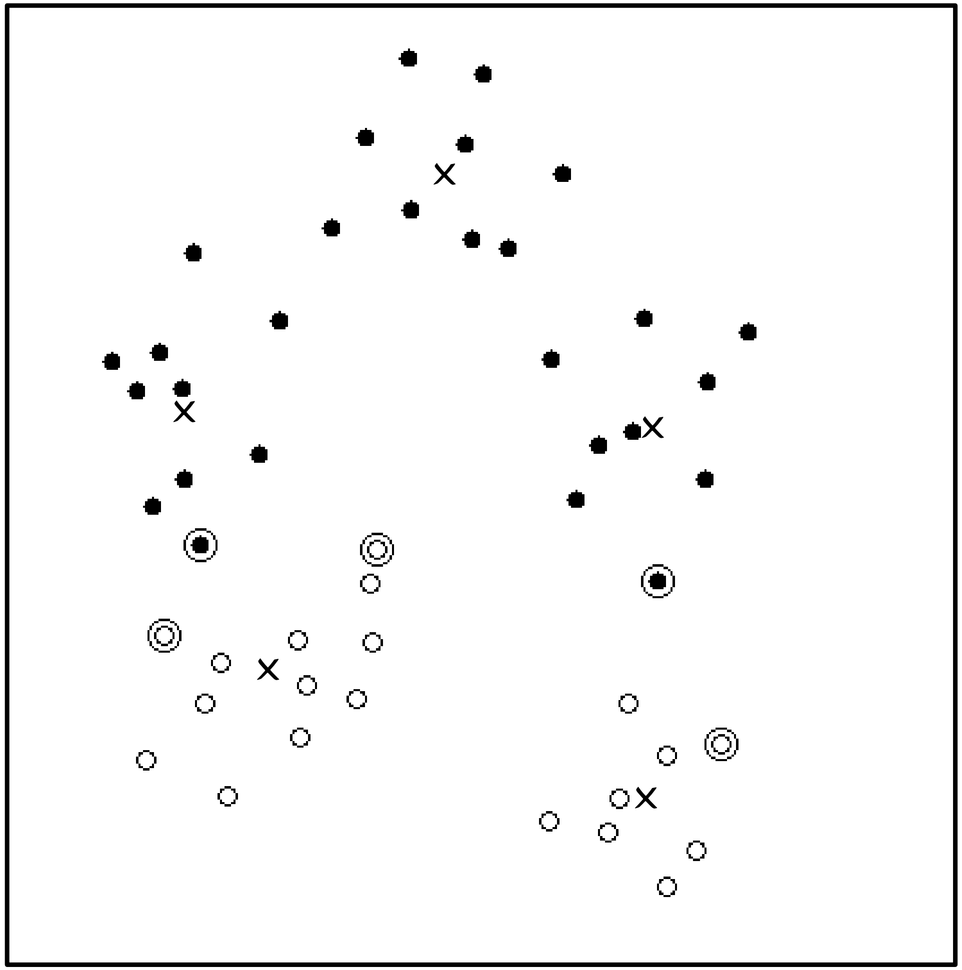
Toy Example: $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2)$



Domain: $[-1, 1]^2$

B. Schölkopf, 7/1998

RBF Center Selection



Comparison to Classical RBF machines

USPS Database	Classification Error Rate		
	classical RBF	RBF with SV centers	full SV machine
Training ($\ell = 7291$)	1.7%	0.0%	0.0%
Test ($\bar{\ell} = 2007$)	6.7%	4.9%	4.2%

Interpretation: Recall that $\quad R[f] \leq R_{emp}[f] + \Phi(h/\ell)$.

SV training does two things:

- minimize R_{emp} by picking the centers (SV) which are crucial for the task
- minimize Φ by minimizing h (via $\|\mathbf{w}\|$)

Results *(Schölkopf, Burges, and Vapnik, 1995)*

On a handwritten digit recognition task (US Postal Service, 7291 examples, 16×16), we constructed three types of SV classifiers:

Polynomial classifiers:	$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$
RBF classifiers:	$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2/(2\sigma^2)\right)$
Neural networks:	$k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \cdot (\mathbf{x} \cdot \mathbf{y}) - \Theta)$

- they use almost the same ($\approx 90\%$) small sets ($\approx 4\%$ of the data base) of Support Vectors.
- all three systems perform well ($\approx 4\%$ error)

C4.5 decision tree	16%	(Cortes and Vapnik, 1995)
LeNet1	5%	(LeCun et al., 1989)
SV	4%	
VSV	3%	
Humans; Tangent Distance	2.5%	(Simard et al., 1993)

Further SV Pattern Recognition Applications

- Text categorization (Joachims, 1997)
- Face detection (Osuna, Freund, and Girosi, 1997b)
- Object recognition (Blanz et al., 1996)
- High-Energy Physics (charm tagging)

In all cases, state-of-the-art performances.

Selecting the Kernel Degree

Recall that $h \leq [R^2 A^2] + 1$, where R is the radius of the smallest ball containing the training set, and $\|\mathbf{w}\| \leq A$.

- Compute R: minimize R^2 subject to $\|\mathbf{x}_i - \mathbf{x}^*\|^2 \leq R^2$. Introduce a Lagrangian, differentiate by $\mathbf{x}^*$, R,...

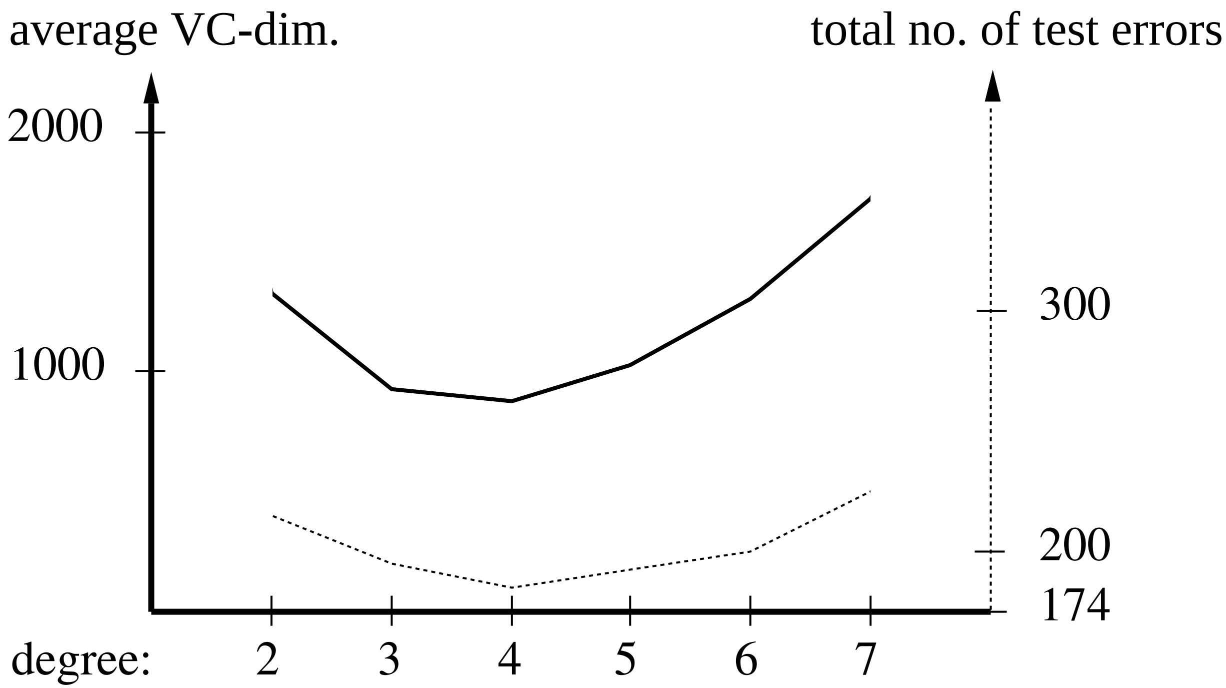
  Dual: maximize $\Sigma_i \lambda_i \cdot (\mathbf{x}_i \cdot \mathbf{x}_i) - \Sigma_{i,j} \lambda_i \lambda_j \cdot (\mathbf{x}_i \cdot \mathbf{x}_j)$ subject to $\Sigma_i \lambda_i = 1,\ \lambda_i \geq 0$ (Schölkopf, Burges, and Vapnik, 1995).
- $\|\mathbf{w}\|$ is computed by the SV algorithm

For degrees $d \geq 2$, the USPS problem is separable. Hence, minimizing

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

$\Longleftrightarrow$ minimizing the VC-dimension h.

Results



average VC-dim.
total no. of test errors
2000
1000
300
200
174
degree:
2
3
4
5
6
7

What Characterizes a Pattern?

- a set of possible instances
- instances are related to each other by *transformations*

Additional knowledge about the structure of the set of possible instances, i.e. about $P(\mathbf{x}, y)$, is called *prior knowledge*.

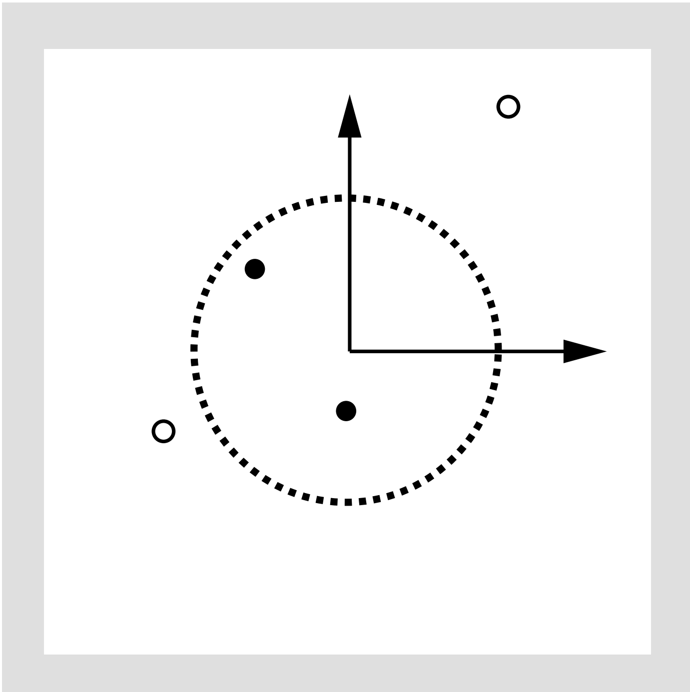
Examples: handwritten character deformations, 3–D object transformations

Learning vs. Prior Knowledge

Task: predict the value of an unknown function at a given point.

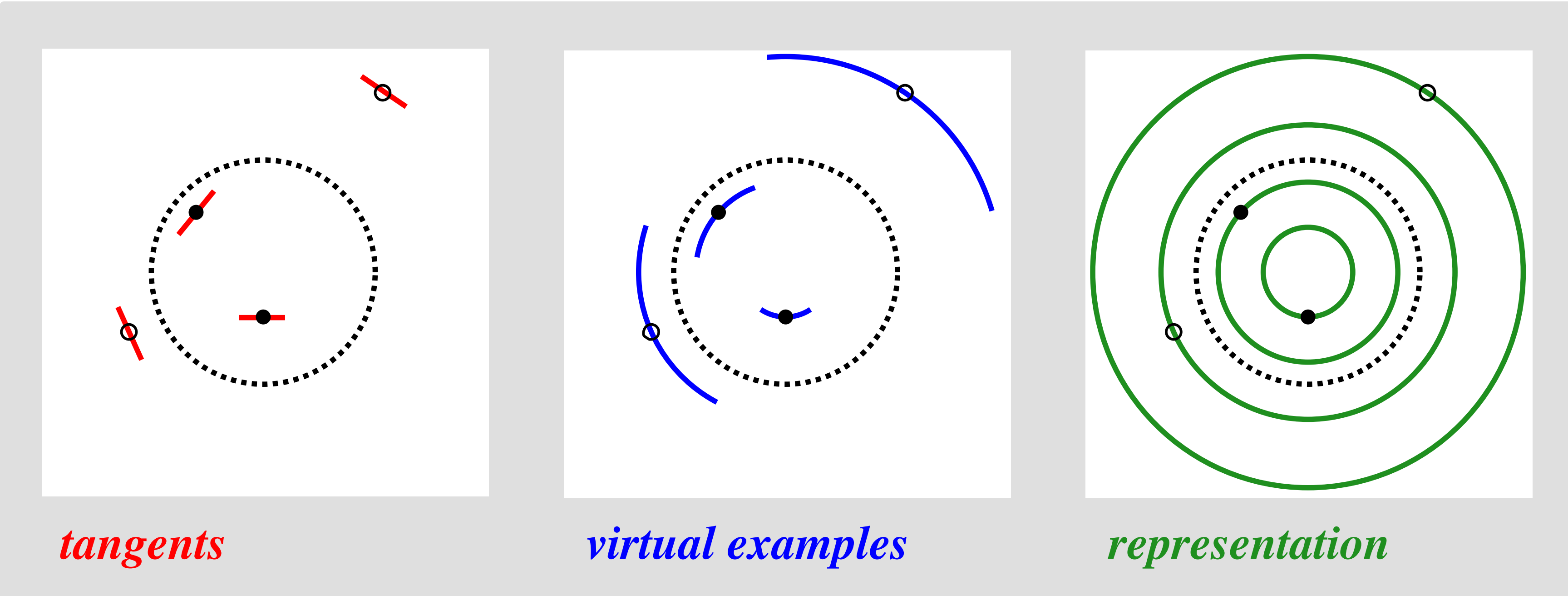
Domain:	Physics	Statistics
Example:	predict the position of a falling rock	Optical Character Recognition
Observations	(position, time)	(pattern, class)
No. of observations:	few	many
Prior knowledge:	much: dynamical model	some: invariances etc.

Incorporating Prior Knowledge: a Toy Example



Prior information: classification function is invariant under rotation about the origin.

Incorporating Prior Knowledge: 3 Approaches

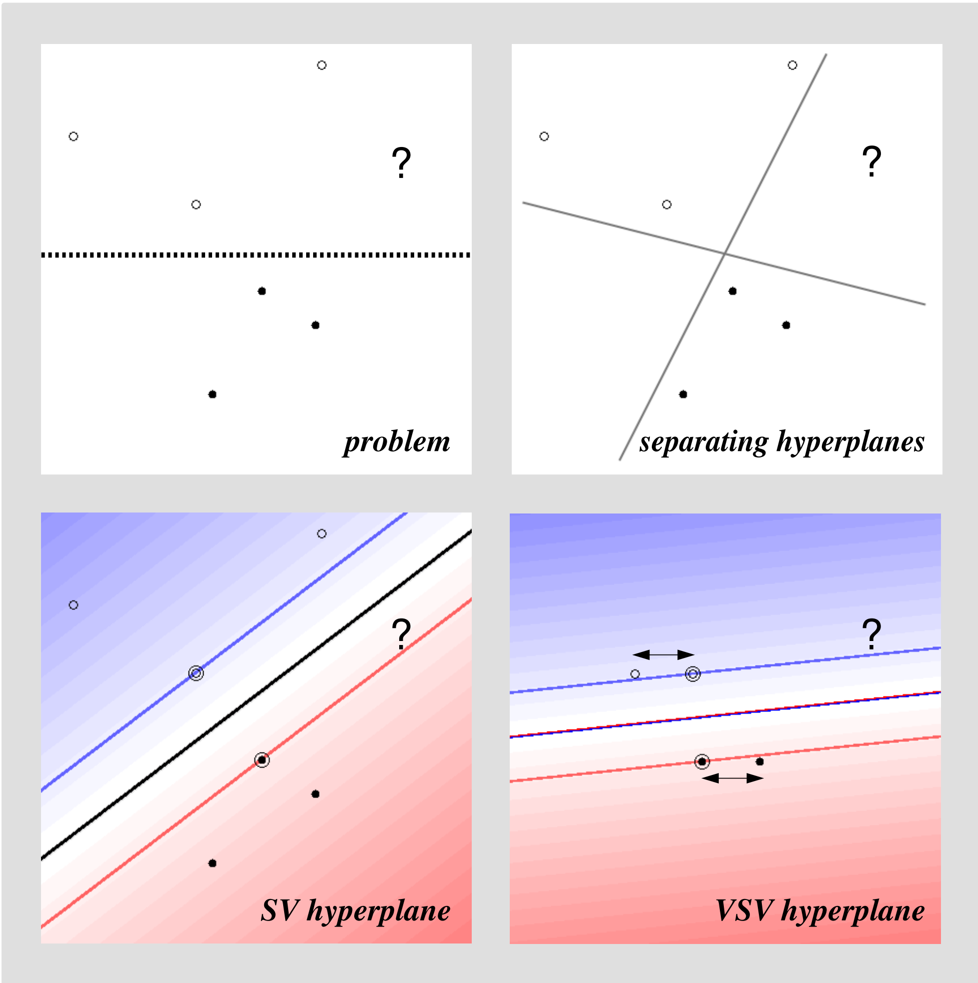


1. left: incorporating a regularizer to learn tangent values
2. middle: generating "virtual examples" from training examples
3. right: changing the representation of the data by first mapping each example to its norm

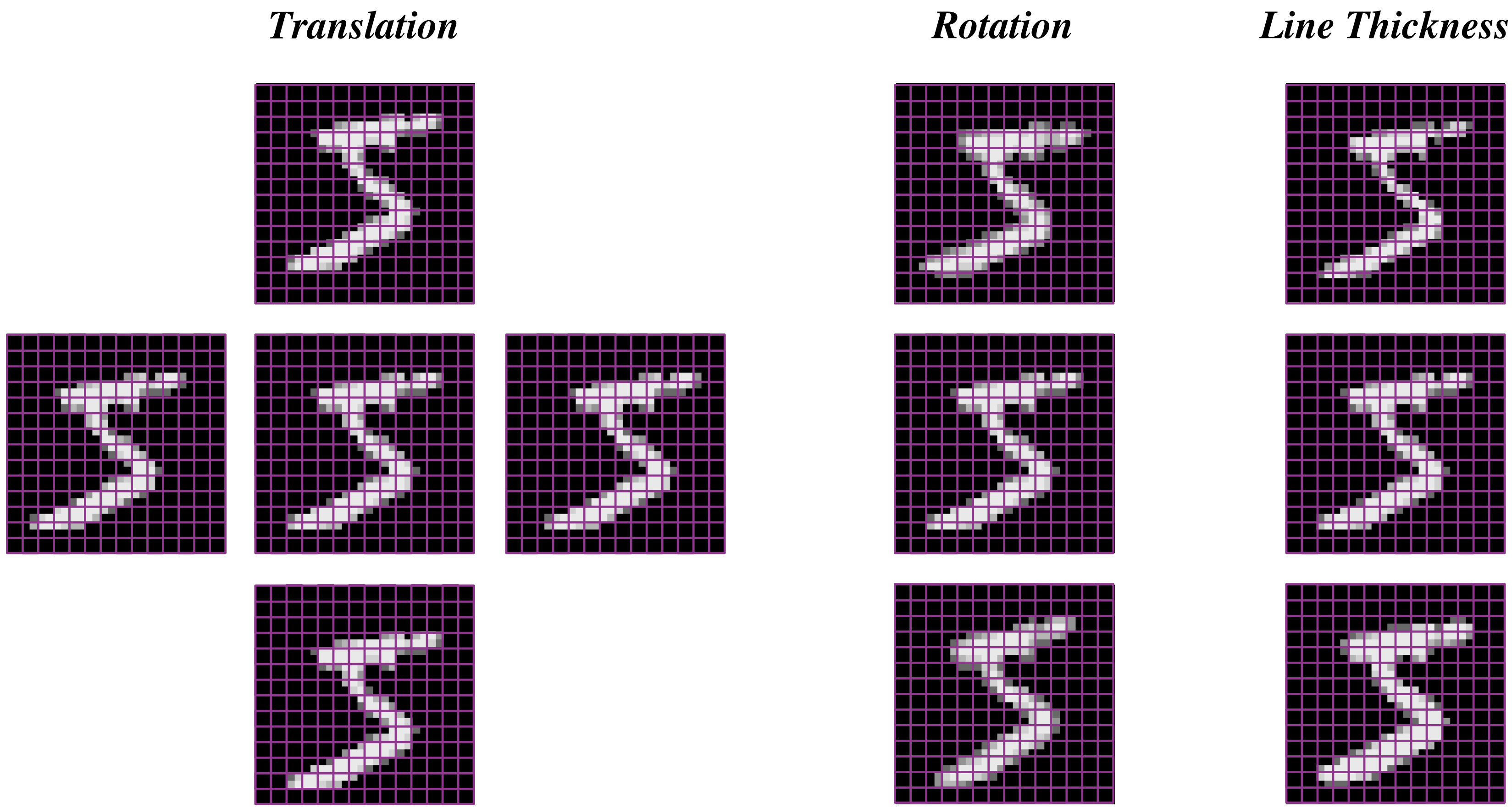
The Virtual SV Method

(Schölkopf et al., 1996a)

Prior knowledge:
classification function is invariant under horizontal translation.



Examples of Invariance Transformations in Optical Character Recognition



(e.g. Drucker, Schapire, and Simard, 1993)

Virtual SV Results: Translation Invariance

- Best single classifier on the MNIST benchmark set of handwritten characters (60000 examples, 28×28). Better than: *LeNet5*, *Tangent Distance*. So far slightly worse than a *boosted ensemble of Neural Nets:* 0.8% vs. 0.7% error.
- USPS benchmark: 3.2% error.

classifier trained on	size	av. no. of SVs	test error
full training set	7291	274	4.0%
overall SV set	1677	268	4.1%
Virtual SV set	8385	686	3.2%
Virtual patterns from full DB	36455	719	3.4%

Invariant Hyperplanes *(Schölkopf et al., 1998d)*

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

$$g(\mathbf{x}) := \sum_{i=1}^{\ell} v_i (B\mathbf{x} \cdot B\mathbf{x}_i) + b.$$

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

$$\frac{1}{\ell} \sum_{j=1}^{\ell} \left(\frac{\partial}{\partial t}\Big|_{t=0} g(\mathcal{L}_t \mathbf{x}_j) \right)^2 .$$

This corresponds to an SV optimization after preprocessing with

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

where

$$C = \frac{1}{\ell} \sum_{j=1}^{\ell} \left(\frac{\partial}{\partial t}\Big|_{t=0} \mathcal{L}_t \mathbf{x}_j \right) \left(\frac{\partial}{\partial t}\Big|_{t=0} \mathcal{L}_t \mathbf{x}_j \right)^{\top} .$$

The Tangent Covariance Matrix

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

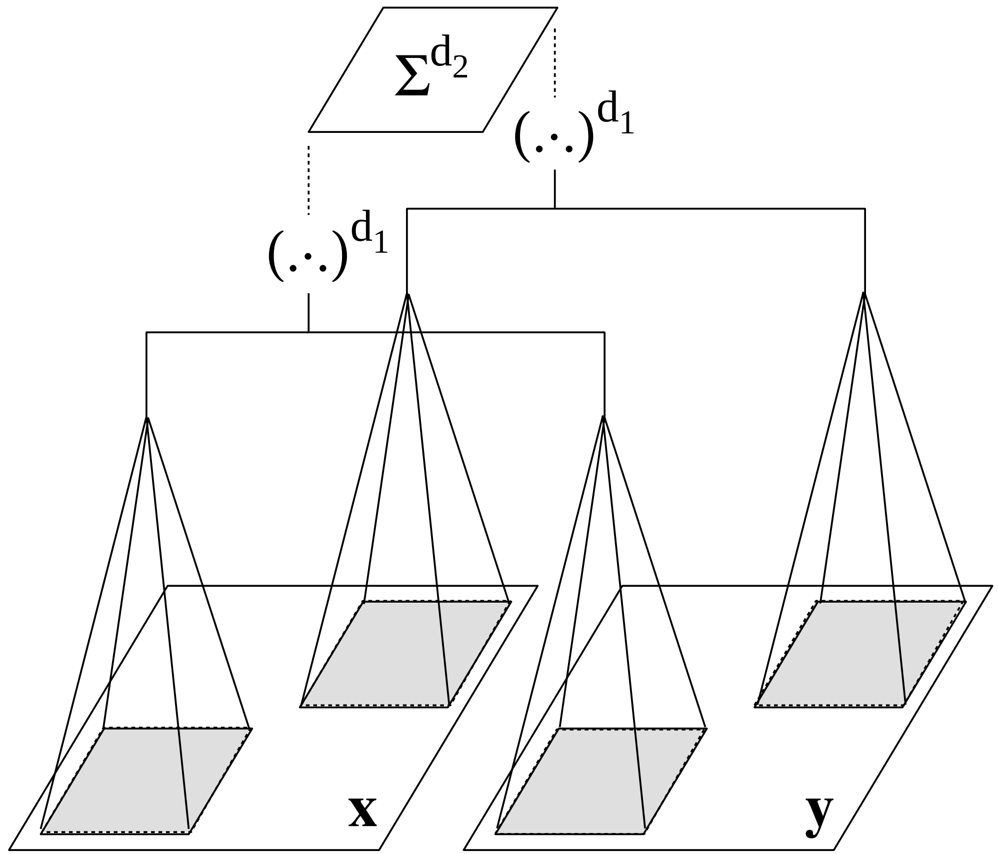
Preprocessing of $\mathbf{x}$:

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

1. project $\mathbf{x}$ onto the Eigenvectors of C
2. divide by the square roots of the Eigenvalues, i.e.: the directions of main variance of $\pm\frac{\partial}{\partial t}|_{t=0}\mathcal{L}_t\mathbf{x}$ are scaled back

- in practice, use $C_\lambda := (1-\lambda)C + \lambda I$

Kernels Local in the Image



Local products of degree d_1, global products of degree d_2, overall degree $d_1 \cdot d_2$.

Summary: Incorporating Prior Knowledge

Subset ($\ell = 5000$) of the MNIST OCR set.

Classifier	Test Error / %
SV	4.0
Virtual SV, with translations	2.8
Invariant hyperplane, $\lambda = 0.4$	3.6
semi-local kernel $k_9^{2,2}$	3.1
purely local kernel $k_9^{4,1}$	3.4
Virtual SV with $k_9^{2,2}$	2.0

Speeding up the decision rule *(Burges and Schölkopf, 1997)*

Approximate

$$\Psi = \sum_{i=1}^{\ell} \alpha_i y_i \Phi(\mathbf{x}_i)$$

by

$$\Psi' = \sum_{i=1}^{N_z} \gamma_i \Phi(\mathbf{z}_i),$$

with $N_z \ll \ell$: Minimize

$$\rho = \|\Psi - \Psi'\|^2$$

(ρ can be expressed in terms of K by using $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) = K(\mathbf{x}, \mathbf{y})$ — "Reduced Set Method")

On the MNIST set, a speed-up by a factor 50 resulted only in a slight decrease in accuracy from 1.0% to 1.1%.

Alternative Reduced Set Methods *(Osuna and Girosi, 1998)*

- reformulate the primal optimization problem, including some penalty terms to ensure that the solution is maximally sparse
- run SV regression on the data $(\mathbf{x}_i, f(\mathbf{x}_i))$, where $\mathbf{x}_i$ are the SVs, and

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

Chunk Training

- naive approach: the complexity of maximizing

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

  scales with the third power of the training set size ℓ

- 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 (Vapnik, 1979)
- in fact, one can use chunks which do not even contain all SVs. Maximize over these sub-problems. In particular, throw out SVs that hit the upper boundary (Osuna, Freund, and Girosi, 1997a).
- the extreme case: by making the sub-problems very small (just two points), one can solve them analytically (Platt, 1998).

SV Machines: Crucial Aspects

Computational The training of an SV machine comprises a (possibly large) convex quadratic program with simple constraints.

Statistical The kernel method allows to reduce a large class of learning machines to separating hyperplanes in some reproducing kernel space. For these, capacity can be controlled independent of dimensionality.

Further Kernel Algorithms

- SV regression
  - cost functions
  - margin maximization
  - optimization problems
- regression applications
  - Boston housing benchmark
  - time series prediction
- kernel PCA
  - derivation of the eigenvalue problem
  - applications: feature extraction, de-noising

SV Regression: ε-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

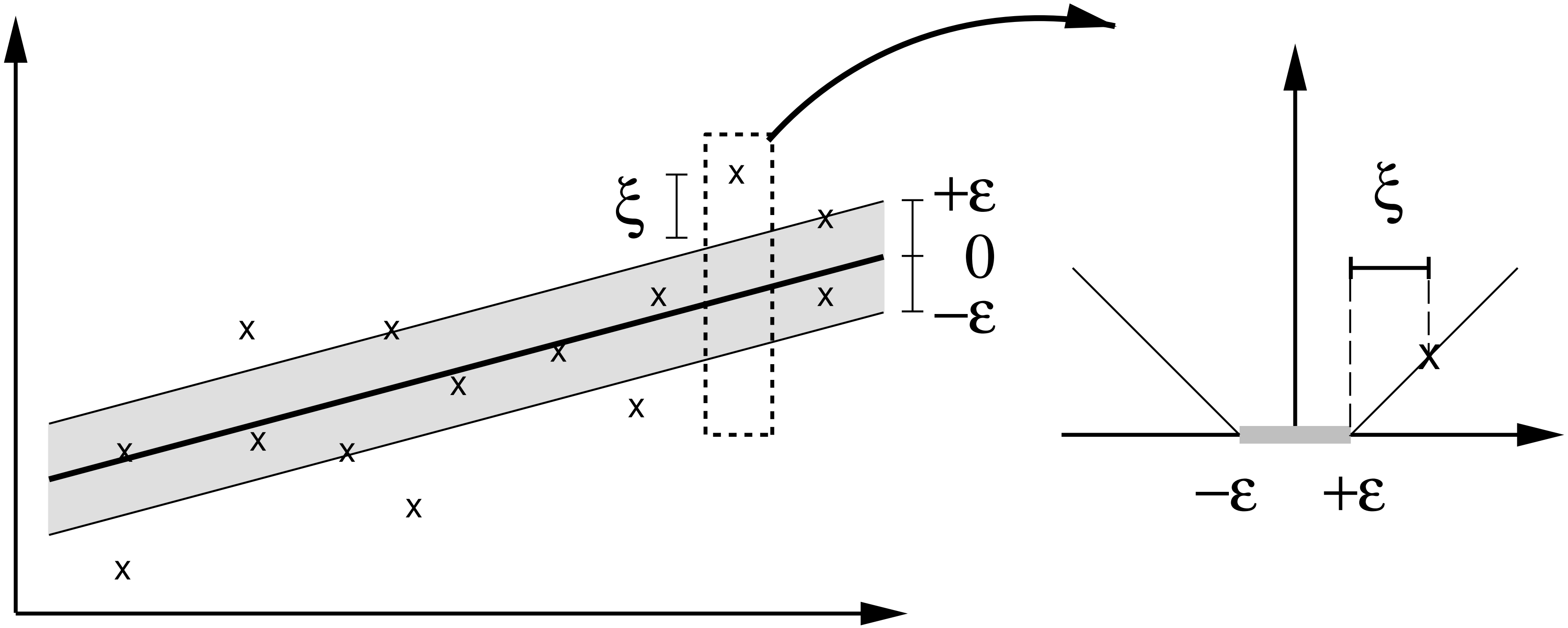
ε-Insensitive Loss:

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

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

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

ε-SV Regression Estimation: The Picture



B. Schölkopf, 7/1998

SV Regression as Margin Maximization

Definition. Let $(E, \|.\|_E)$, $(F, \|.\|_F)$ be normed spaces, and $C \subset E$. The ε-margin of a function $f : C \to F$ is

$$m_\varepsilon(f) := \inf\{\|\mathbf{x} - \mathbf{y}\|_E \colon \mathbf{x}, \mathbf{y} \in C, \|f(\mathbf{x}) - f(\mathbf{y})\|_F \geq 2\varepsilon\}.$$

- $m_\varepsilon(F) > 0$ for all $\varepsilon > 0$ if and only if f is uniformly continuous.
- $M_\varepsilon \geq \frac{2\varepsilon}{L}$ if f is Lipschitz-continuous with constant L
- for linear functions $f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b$, we have $m_\varepsilon(f) = \frac{2\varepsilon}{\|\mathbf{w}\|}$. Minimizing $\|\mathbf{w}\|$ thus amounts to maximizing $m_\varepsilon(f)$.
- for $C = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, the margin $m_1(f) = \frac{2}{\|\mathbf{w}\|}$ is the margin of the canonical hyperplane.

Formulation as an Optimization Problem

Estimate a linear regression

$$f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b$$

with precision ε by minimizing

$$\begin{array}{ll} \text{minimize} & \tau(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \\ \text{subject to} & ((\mathbf{w} \cdot \mathbf{x}_i) + b) - y_i \le \varepsilon + \xi_i \\ & y_i - ((\mathbf{w} \cdot \mathbf{x}_i) + b) \le \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0 \end{array}$$

for all $i = 1, \ldots, \ell$.

Dual Problem, In Terms of Kernels

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

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

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

The regression estimate takes the form

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

Boston Housing Benchmark

- 506 examples, 13-dimensional.
- 100 runs, with 25 randomly selected test points.
- training set is split into actual training set and validation set (80 points) for selecting ε, C, and kernel parameters
- ftp://ftp:ics.uci.com/pub/machine-learning-databases/housing

Results (MSE):

- Bagging regression trees: 11.7 (Breiman, 1994)
- ε-SV regression: 7.6 (Stitson et al., 1998)

Extensions of the SV Algorithm

- separate regularization and function class
- more general cost functions
- conditionally positive definite kernels
- automatic ε-tube

(Smola and Schölkopf, 1998a; Smola, Schölkopf, and Müller, 1998; Schölkopf, Bartlett, Smola, and Williamson, 1998a)

SVM vs. Regularization Theory *(Smola et al., 1998)*

The problem of minimizing the regularized risk

$$R_{reg}[f] = R_{emp}[f] + \lambda \|Pf\|^2,$$

for

$$f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b$$

leads to a QP with matrix $KD^{-1}K$, where
$D_{ij} := ((Pk)(\mathbf{x}_i, .) \cdot (Pk)(\mathbf{x}_j, .))$ and $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

This is equivalent to the SVM if $K = D$, which holds if k is Green's function of P^*P:

$$\begin{aligned} ((Pk)(\mathbf{x}_i, .) \cdot (Pk)(\mathbf{x}_j, .)) &= (k(\mathbf{x}_i, .) \cdot (P^*Pk)(\mathbf{x}_j, .)) \\ &= (k(\mathbf{x}_i, .) \cdot \delta_{\mathbf{x}_j}(.)) = k(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

ν-SV Regression *(Schölkopf et al., 1998a)*

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

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

Proposition. If ν-SVR leads to the solution $\bar{\varepsilon}, \bar{\mathbf{w}}, \bar{b}$, then ε-SVR with ε set a priori to $\bar{\varepsilon}$, and the same value of C, has the solution $\bar{\mathbf{w}}, \bar{b}$.

Proof. If we minimize the primal objective function, then fix ε and minimize only over the remaining variables, the solution does not change.

"Only" difference to ε-SVR: in ν-SVR, ε is computed automatically.

ν-SV Regression: Dual Problem

Maximize

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

subject to

$$\begin{aligned} \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) &= 0 \\ 0 \leq \alpha_i^{(*)} &\leq C \\ \sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) &\leq C\ell\nu. \end{aligned}$$

The Meaning of ν

Theorem. Assume $\varepsilon > 0$. The following statements hold:

(i) ν is an upper bound on the fraction of errors.

(ii) ν is a lower bound on the fraction of SVs.

(iii) Suppose the data were generated iid from a distribution $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$ with $P(y|\mathbf{x})$ continuous. With probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of errors.

Proof Idea. *Ad (i)* For errors, we have $\alpha_i^{(*)} = C$. The constraint on $\Sigma_i(\alpha_i + \alpha_i^*)$ then implies (i). *Ad (ii)* Observe that for $\varepsilon > 0$, $\Sigma_i(\alpha_i + \alpha_i^*) = C\ell\nu$. This gives a lower bound on the number of nonzero multipliers. *Ad (iii)* Asymptotically, all SVs lie outside the tube.

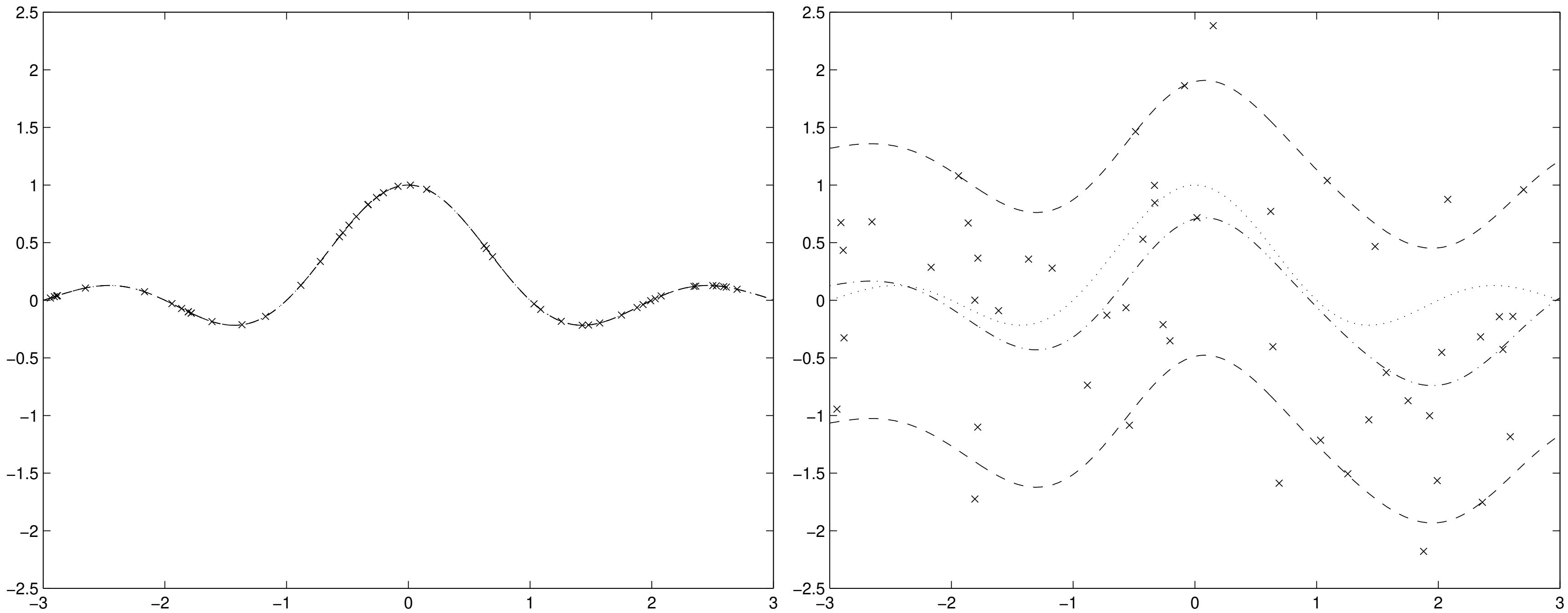
Toy Examples: Estimating a Noisy Sinc Function

$\nu = 0.2$

ℓ	10	50	100	200	500	1000	1500	2000
ε	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

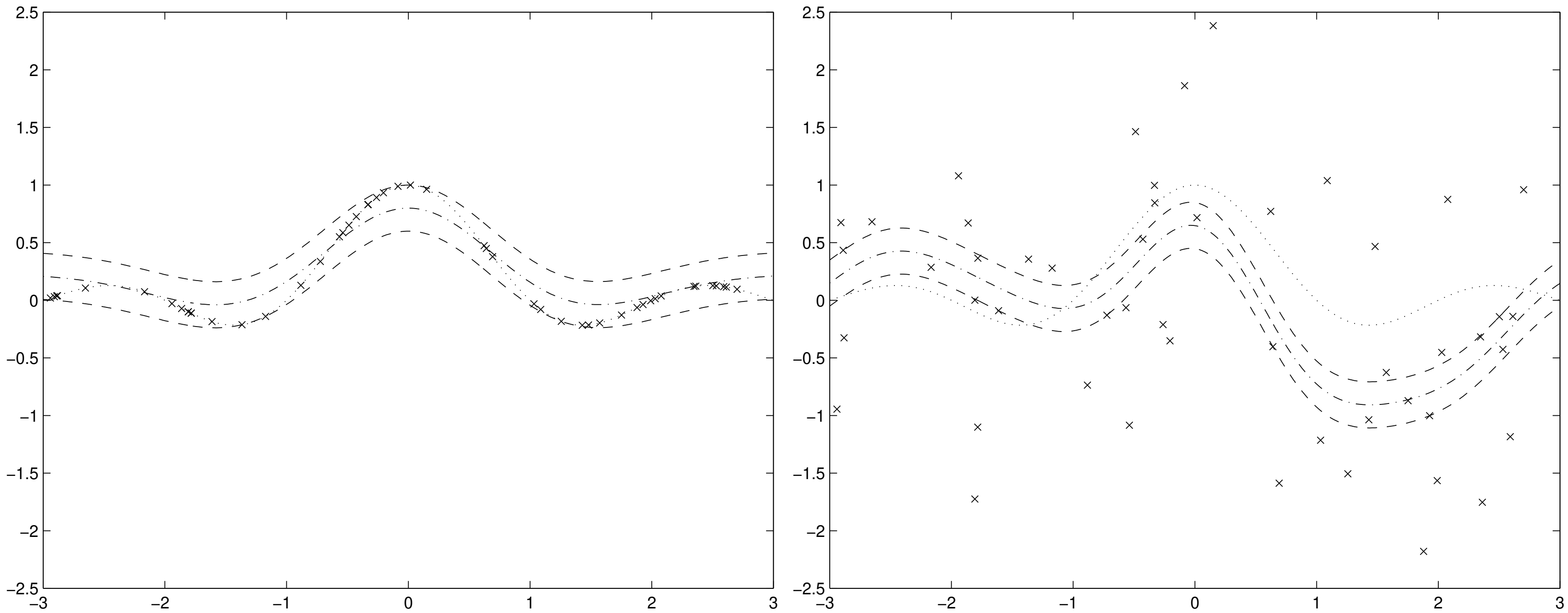
- automatically computed ε largely independent of ℓ
- asymptotics consistent with theorem

ν-SV-Regression: Automatic Tube Tuning



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

ε-SV-Regression, Run on the Same Data



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

Boston housing benchmark

ν-SVR	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
automatic ε	2.6	1.7	1.2	0.8	0.6	0.3	0.0	0.0	0.0	0.0
MSE	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

ε-SVR	0	1	2	3	4	5	6	7	8	9	10
MSE	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 σ chosen as in (Schölkopf et al., 1997)

Parametric Error Models

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

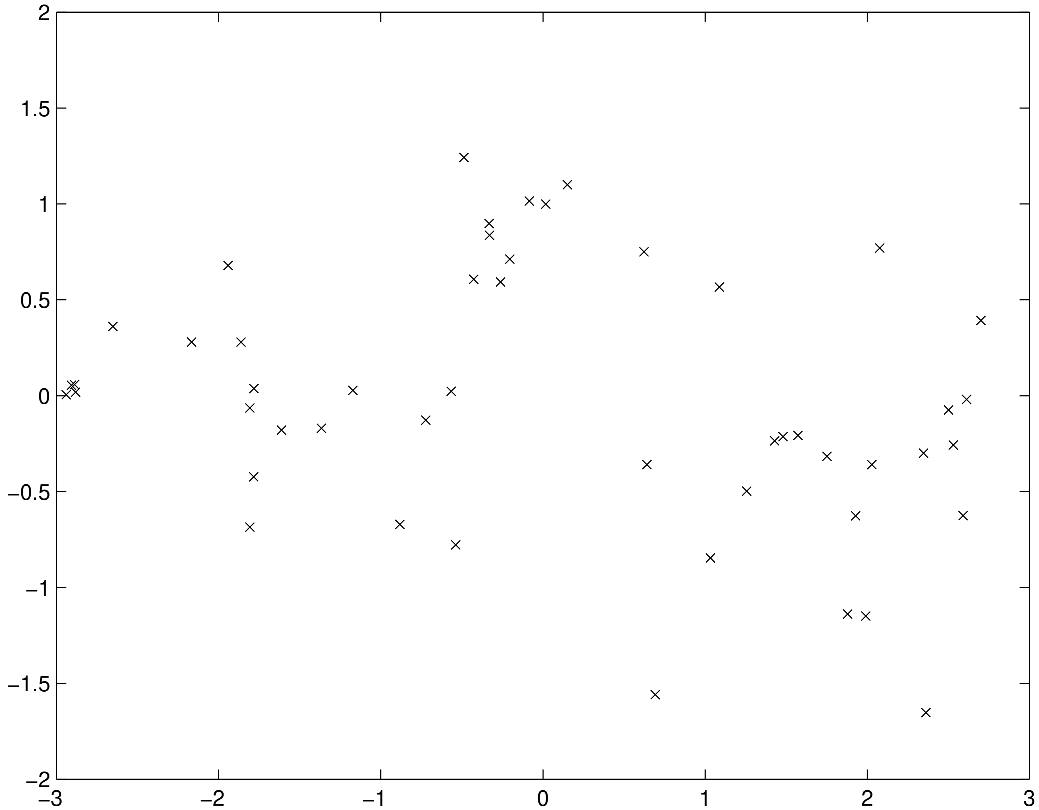
$$\tau(\mathbf{w}, \boldsymbol{\xi}^{(*)}, \varepsilon) = \|\mathbf{w}\|^2/2 + C \cdot \left(\nu\ell\varepsilon + \sum_{i=1}^{\ell}(\xi_i + \xi_i^*)\right)$$

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

This leads to the "usual" dual, with the modified last constraint

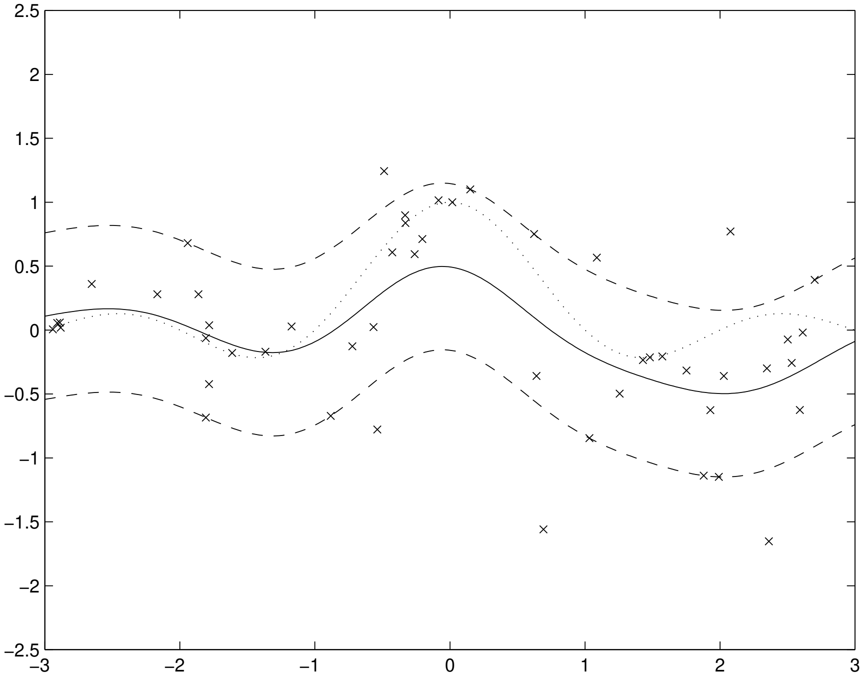
$$\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*)\zeta(\mathbf{x}_i) \leq C\ell\nu.$$

Toy Example: Some Noisy Data

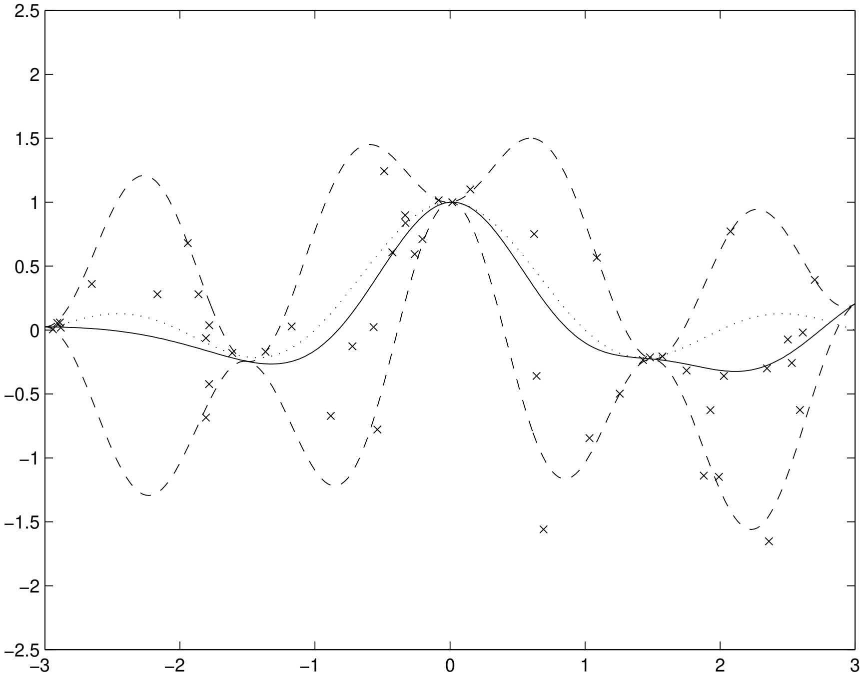


Assumption: we have prior knowledge indicating that the noise is modulated by $\zeta(x) = \sin^2((2\pi/3)x)$.

Toy Example, II



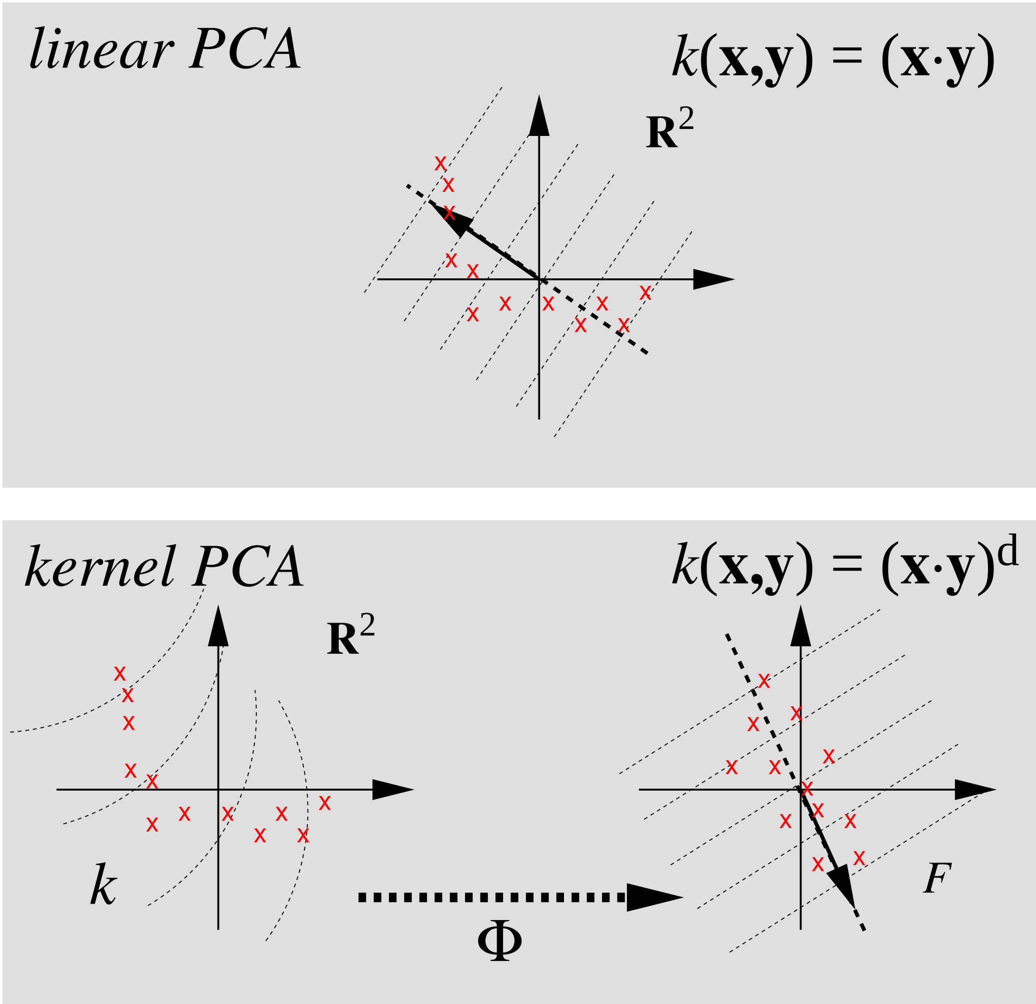
constant-radius tube



parametric model using $\zeta(x)$

Kernel PCA

(Schölkopf, Smola, and Müller, 1998e)



PCA in High-dimensional Feature Spaces

$\mathbf{x}_1, \ldots, \mathbf{x}_\ell, \quad \Phi : \mathbf{R}^N \to F,\ \mathbf{x} \mapsto \mathbf{X}, \quad C = \frac{1}{\ell}\sum_{j=1}^{\ell} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^\top$

Eigenvalue problem

$$\lambda \mathbf{V} = C\mathbf{V} = \frac{1}{\ell}\sum_{j=1}^{\ell}(\Phi(\mathbf{x}_j)\cdot\mathbf{V})\Phi(\mathbf{x}_j).$$

For $\lambda \neq 0$, $\mathbf{V} \in \mathrm{span}\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_\ell)\}$, thus

$$\Longleftrightarrow \lambda(\Phi(\mathbf{x}_k)\cdot\mathbf{V}) = (\Phi(\mathbf{x}_k)\cdot C\mathbf{V}) \text{ for all } k = 1, \ldots, \ell$$

$$\mathbf{V} = \sum_{i=1}^{\ell}\alpha_i\Phi(\mathbf{x}_i)$$

Nonlinear PCA as an Eigenvalue Problem

Define an $\ell \times \ell$ matrix

$$K_{ij} := (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j)$$

to get

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

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_\ell)^\top$.
Solve

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

$\longrightarrow (\lambda_k, \boldsymbol{\alpha}^k)$

$$(\mathbf{V}^k \cdot \mathbf{V}^k) = 1 \Longleftrightarrow \lambda_k (\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k) = 1$$

Feature extraction

Compute projections on the Eigenvectors

$$\mathbf{V}^k = \sum_{i=1}^{M} \alpha_i^k \Phi(\mathbf{x}_i)$$

in F:

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

$$\begin{aligned}(\mathbf{V}^k \cdot \Phi(\mathbf{x})) &= \sum_{i=1}^{M} \alpha_i^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) \\ &= \sum_{i=1}^{M} \alpha_i^k k(\mathbf{x}_i, \mathbf{x})\end{aligned}$$

Centering in F

Center the data in F:

$$\tilde{\Phi}(\mathbf{x}_i) := \Phi(\mathbf{x}_i) - \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(\mathbf{x}_i)$$

For $\tilde{\Phi}(\mathbf{x}_i)$, everything works fine.

Express $\tilde{K}$ in terms of K, using $(1_\ell)_{ij} := 1/\ell$:

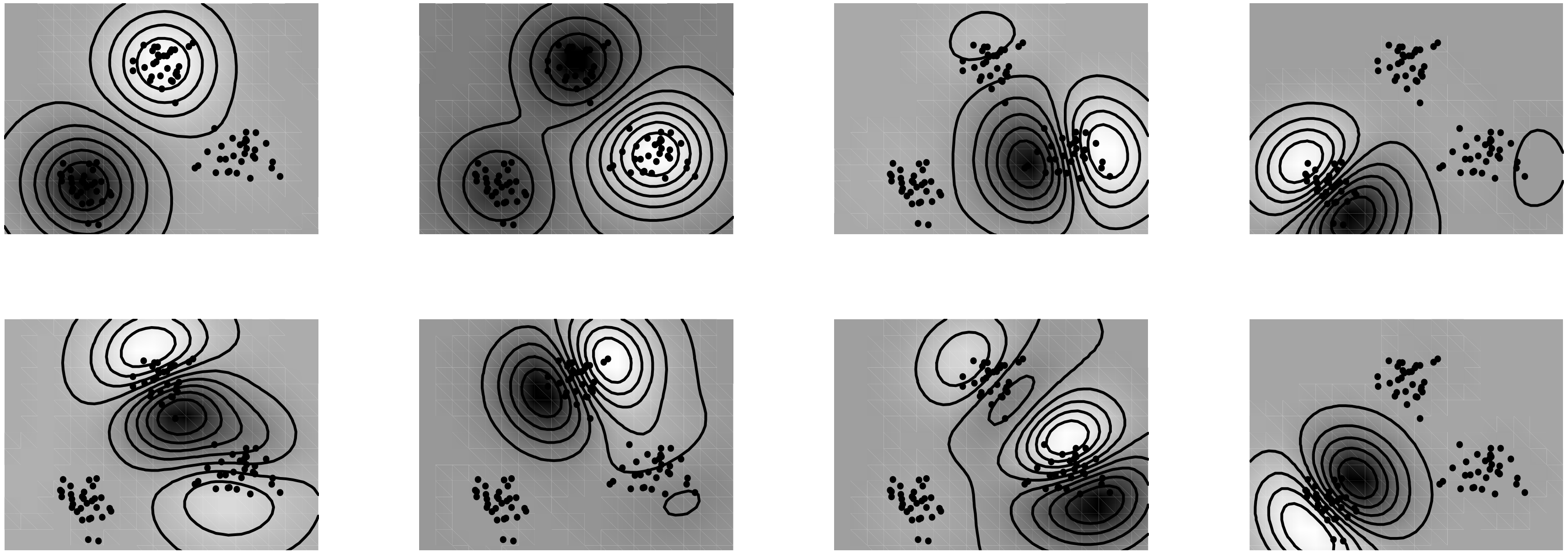
$$\tilde{K}_{ij} = K - 1_\ell K - K 1_\ell + 1_\ell K 1_\ell.$$

Compute $\tilde{K}$ and solve the Eigenvalue problem.

Similar for feature extraction.

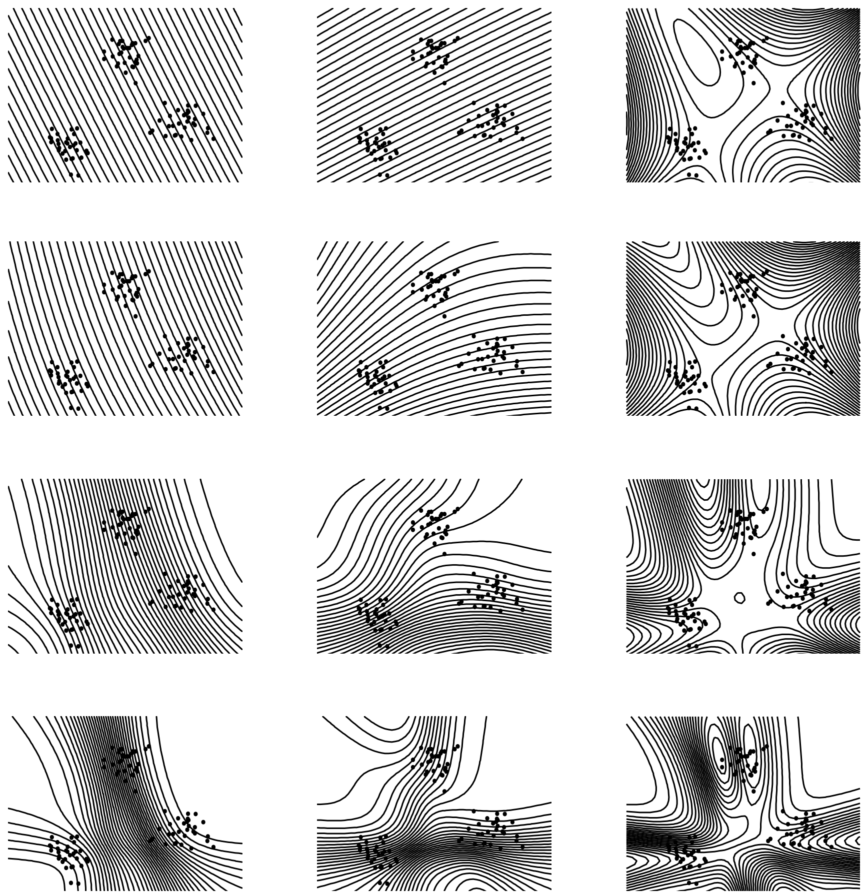
Toy Example: RBF Kernel, 8 Principal Components

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{0.1}\right)$$



Matlab code available from http://svm.first.gmd.de

Gradual Transition from Linear PCA to Kernel PCA



$\kappa = 0.1$

$\kappa = 1$

$\kappa = 5$

$\kappa = 10$

$$k(\mathbf{x}, \mathbf{y}) = \tanh\left(\kappa(\mathbf{x} \cdot \mathbf{y}) + 1\right)$$

Handwritten Digit Feature Extraction

USPS data, polynomial kernel $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$ with $d = 1, \ldots, 7$.
Linear SV machine (hyperplane) trained on kernel PCA features.

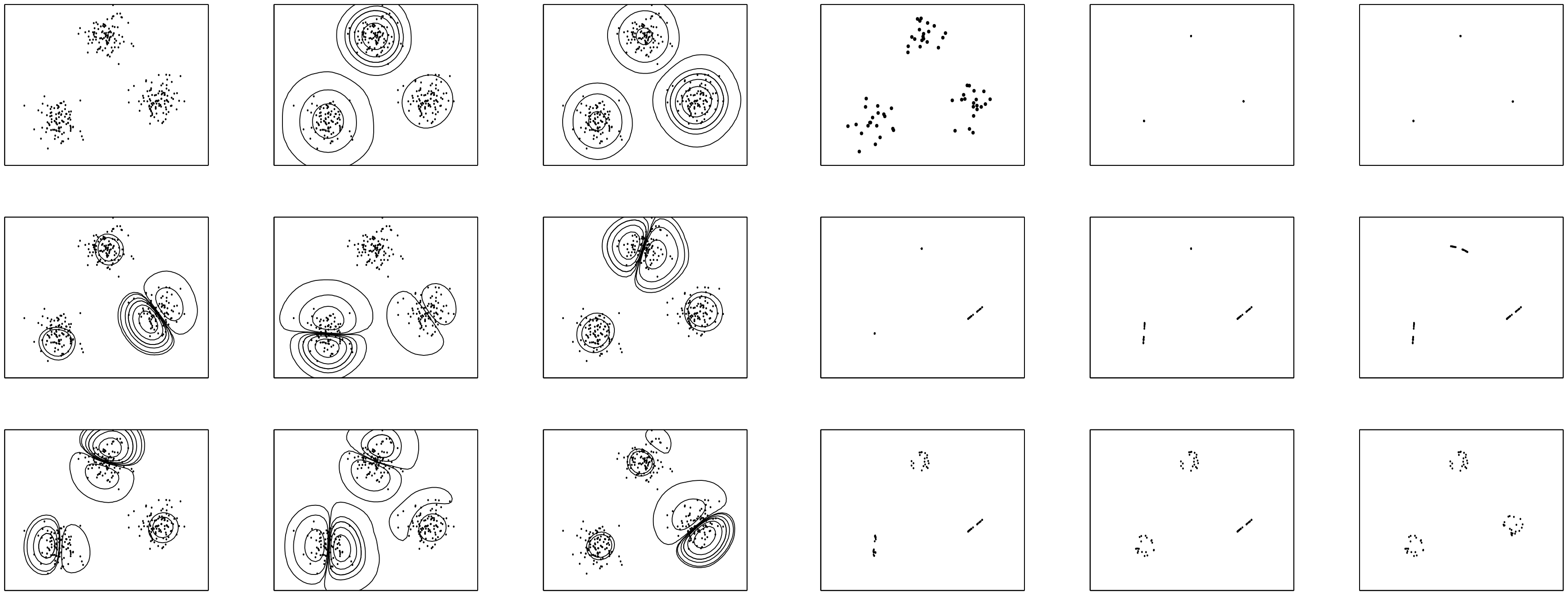
# of components	Test Error Rate for degree						
	1	2	3	4	5	6	7
64	8.8	7.3	6.8	6.7	6.7	7.2	7.5
128	8.6	5.8	5.9	6.1	5.8	6.0	6.8
256	8.7	5.5	5.3	5.2	5.2	5.4	5.4
512	n.a.	4.9	4.6	4.4	5.1	4.6	4.9
1024	n.a.	4.9	4.3	4.4	4.6	4.8	4.6
2048	n.a.	4.9	4.2	4.1	4.0	4.3	4.4

Bottomline: better than linear PCA for corresponding numbers of features, even better when freedom to use *more* features is used.

Kernel PCA Denoising

(Schölkopf et al., 1998c)

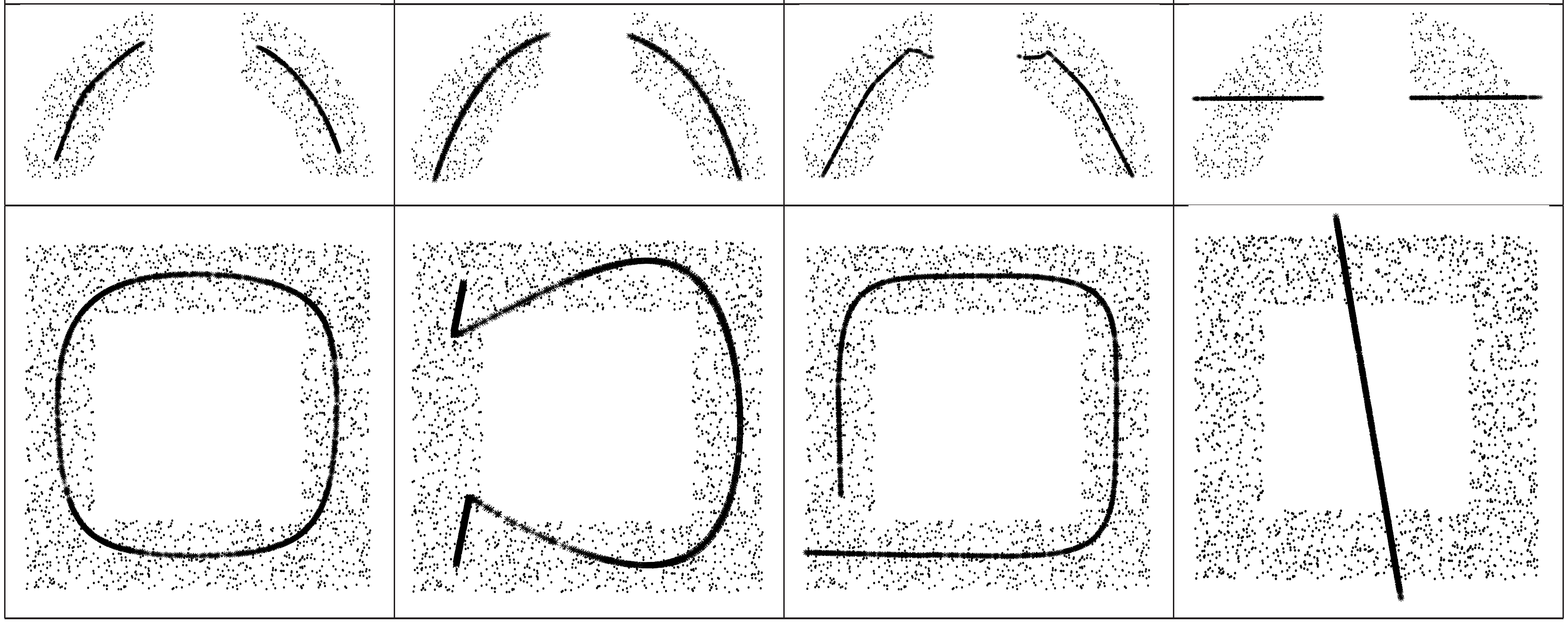
Idea: in feature space, we discard higher-order principal components, and compute approximate pre-images.



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

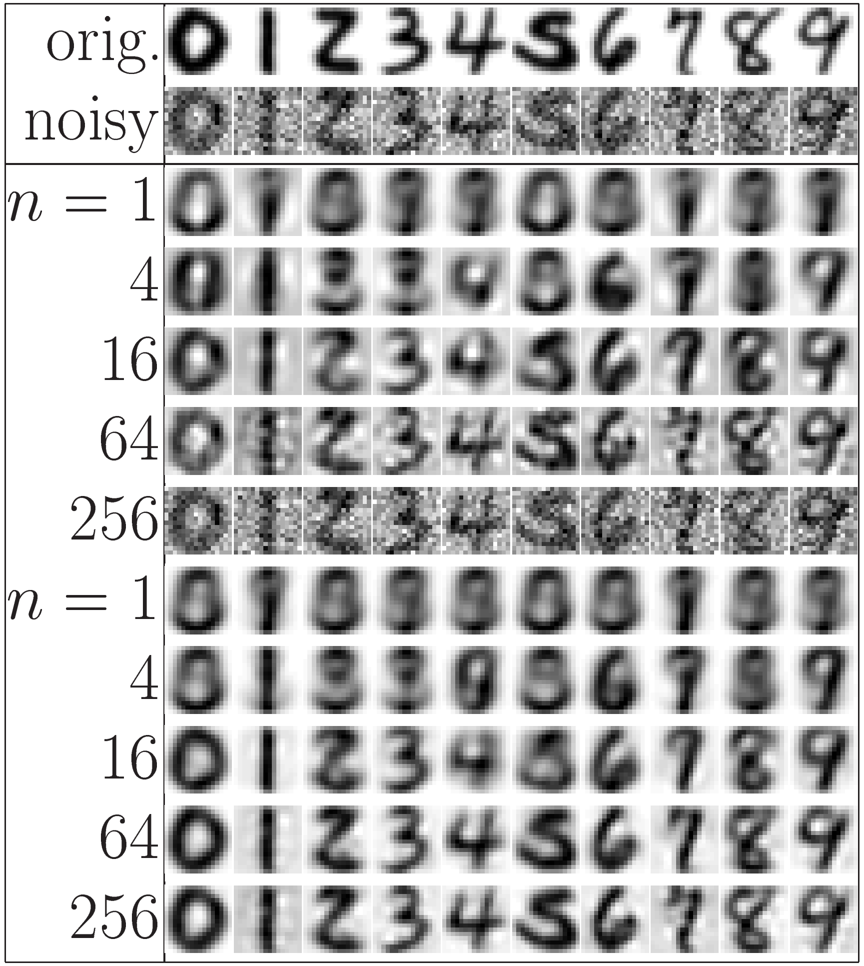
Comparison of Different Algorithms

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



(Hastie and Stuetzle, 1989; Diamantaras and Kung, 1996)

Denoising of USPS Handwritten Digits



Gaussian noise ($\sigma = 0.5$)

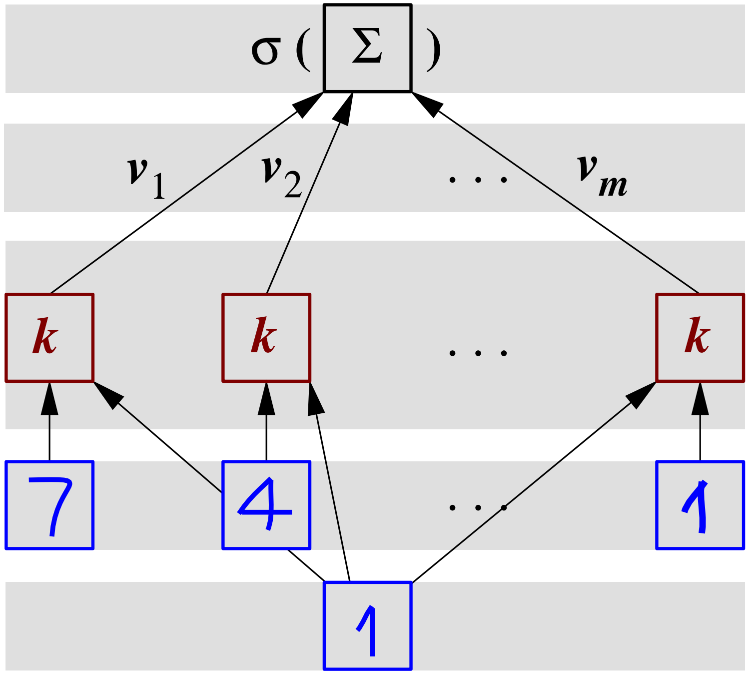
linear PCA reconstruction using $n = 1, 4, 16, 64, 256$ components

kernel PCA reconstruction using $n = 1, 4, 16, 64, 256$ components

Other Kernel Algorithms

- Clustering (Schölkopf, Smola, and Müller, 1996b; Graepel and Obermayer, 1998)
- Similarity-based algorithms
- Kernel Perceptron variants
- ICA

SV Learning Architecture



output $\sigma\left(\Sigma\, v_i\, k\,(\mathbf{x},\mathbf{x}_i)\right)$

weights

comparison: e.g. $k(\mathbf{x},\mathbf{x}_i)=(\mathbf{x}\cdot\mathbf{x}_i)^d$
$k(\mathbf{x},\mathbf{x}_i)=\exp(-\|\mathbf{x}-\mathbf{x}_i\|^2/c)$
$k(\mathbf{x},\mathbf{x}_i)=\tanh(\kappa(\mathbf{x}\cdot\mathbf{x}_i)+\theta)$

patterns $\mathbf{x}_1 \ldots \mathbf{x}_m$

input pattern $\mathbf{x}$

Further Details...

This tutorial cannot explain all details, or cover all contributions to SV research. For more exhaustive treatments, see

- Vapnik (1995) *(SVMs and learning theory)*
- Schölkopf, Burges, and Smola (1998b) *(Proceedings of the 1997 NIPS workshop)*
- Burges (1998) *(SV pattern recognition tutorial)*
- Smola and Schölkopf (1998b) *(SV regression tutorial)*

Many of the papers can be downloaded from `svm.first.gmd.de`.

Conclusion

- *Kernels:*
  - carry out linear methods in high-dimensional spaces with a computational cost independent of their dimensionality
  - every algorithm that can be formulated in dot products could benefit from this technique
- *SV machines:*
  - extract SVs characterizing a given task
  - achieve high accuracies, especially using the VSV method
- *Kernel PCA:*
  - extracts features leading to high classification accuracies
  - is carried out by solving an Eigenvalue problem

References

K. Bennett. Combining support vector and mathematical programming methods for induction. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1998. In press.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

V. Blanz, B. Schölkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3D models. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks — ICANN'96*, pages 251 – 256, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112.

B. E. Boser, I. M. Guyon, and V. N. 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, 1992. ACM Press.

L. Breiman. Bagging predictors. Technical Report 421, Department of Statistics, UC Berkeley, 1994. ftp://ftp.stat.berkeley.edu/pub/tech-reports/421.ps.Z.

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.

C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):1–47, 1998.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

K. I. Diamantaras and S. Y. Kung. *Principal Component Neural Networks*. Wiley, New York, 1996.

H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705 – 719, 1993.

T. Graepel and K. Obermayer. Fuzzy topographic kernel clustering. In W. Brauer, editor, *Proceedings of the 5th GI Workshop Fuzzy Neuro Systems '98*, pages 90 – 97, 1998.

T. Hastie and W. Stuetzle. Principal curves. *JASA*, 84:502 – 516, 1989.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, LS VIII, University of Dortmund, 1997.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541 – 551, 1989.

E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. AI Memo 1602, Massachusetts Institute of Technology, 1997a.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proc. Computer Vision and Pattern Recognition '97*, pages 130–136, 1997b.

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. To appear.

J. Platt. Fast training of svms using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1998. In press.

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*. AAAI Press, Menlo Park, CA, 1995.

B. Schölkopf, P. Bartlett, A. Smola, and R. Williamson. Support vector regression with automatic accuracy control. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, Berlin, 1998a. Springer Verlag. In press.

B. Schölkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks — ICANN'96*, pages 47 – 52, Berlin, 1996a. Springer Lecture Notes in Computer Science, Vol. 1112.

B. Schölkopf, C.J.C. Burges, and A.J. Smola. *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1998b. In press.

B. Schölkopf, S. Mika, A. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction *via* approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, Berlin, 1998c. Springer Verlag. In press.

B. Schölkopf, P. Simard, A. 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*, Cambridge, MA, 1998d. MIT Press.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max-Planck-Institut für biologische Kybernetik, 1996b.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299 – 1319, 1998e.

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 Trans. Sign. Processing*, 45:2758 – 2765, 1997.

J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. A framework for structural risk minimization. In *COLT*, 1996.

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*, pages 50–58, San Mateo, CA, 1993. Morgan Kaufmann.

A. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 1998a. In press.

A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 1998b. Invited paper, in press.

A. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 1998. In press.

M. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with ANOVA decomposition kernels. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1998. to appear.

V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

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).

V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24, 1963.

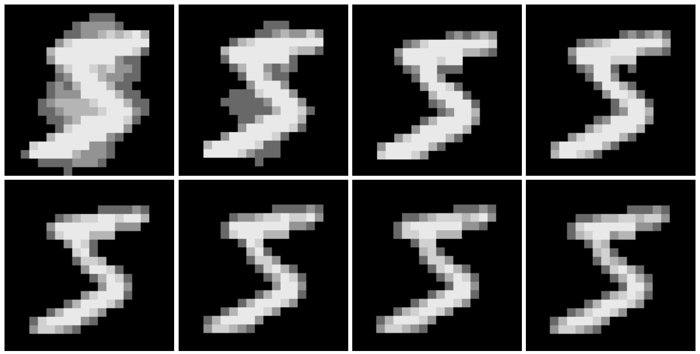
J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, Egham, TW20 0EX, UK, 1998.

Invariant Hyperplane Preprocessing: Results

Subset of the MNIST database. Lowest test error with $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^4$, using various levels of smoothing: 4.0%.

Classification error rates for $k(\mathbf{x}, \mathbf{y}) = (B_\lambda \mathbf{x} \cdot B_\lambda \mathbf{y})^4$, with the invariant hyperplane preprocessing matrix $B_\lambda = C_\lambda^{-\frac{1}{2}}$.

λ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
error rate in %	4.2	3.8	3.6	3.6	3.7	3.8	3.8	3.9	3.9	4.0



USPS Results

polynomial: $k(\mathbf{x}, \mathbf{y}) = ((\mathbf{x} \cdot \mathbf{y})/256)^d$

d	1	2	3	4	5	6	7
raw error/%	8.9	4.7	4.0	4.2	4.5	4.5	4.7
av. # of SVs	282	237	274	321	374	422	491

RBF: $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2/(256\, c)\right)$

c	4.0	2.0	1.2	0.8	0.5	0.2	0.1
raw error/%	5.3	5.0	4.9	4.3	4.4	4.4	4.5
av. # of SVs	266	240	233	235	251	366	722

NN: $k(\mathbf{x}, \mathbf{y}) = 1.04 \tanh(2(\mathbf{x} \cdot \mathbf{y})/256 + \Theta)$

$-\Theta$	0.8	0.9	1.0	1.1	1.2	1.3	1.4
raw error/%	6.3	4.8	4.1	4.3	4.3	4.4	4.8
av. # of SVs	206	242	254	267	278	289	296

MNIST SV Set Overlap

polynomial: $k(\mathbf{x}, \mathbf{y}) = ((\mathbf{x} \cdot \mathbf{y})/784)^d$

d	2	3	4	5	6	7
# of test errors	163	147	135	131	127	127
# of SVs	994	1083	1187	1292	1401	1537

RBF: $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2/(784\ c)\right)$

c	1	0.75	0.6	0.5	0.4	0.3
# of test errors	147	145	145	141	137	134
# of SVs	1061	1118	1179	1264	1308	1460

sigmoid: $k(\mathbf{x}, \mathbf{y}) = 1.04 \tanh(2(\mathbf{x} \cdot \mathbf{y})/784 + \Theta)$

$-\Theta$	1.3	1.4	1.5	1.6	1.7	1.8
# of test errors	139	138	138	141	145	144
# of SVs	1137	1162	1194	1211	1223	1217

	Polyn	RBF	tanh	intersection	shared by 2	union
no. of SVs	1187	1179	1194	1054	124	1328
% intersection	89	89	88	100	–	–