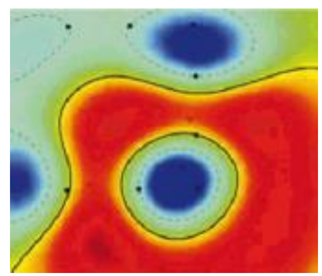
Kernel Methods

Introduction to SVMs, KPCA, RDE



Lecture by Klaus-Robert Müller, TUB 2024

Basic ideas in learning theory

Three scenarios: regression, classification & density estimation.
Learn f from examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathbb{R}^n \times \mathbb{R}^m \text{ or } \{\pm 1\}, \quad \text{generated from } P(\mathbf{x}, y),$$

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

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

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

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

Basic ideas in learning theory II

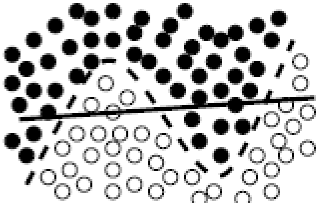
- Law of large numbers: $R_{emp}[f] \to R[f]$ as $N \to \infty$.
  "*consistency*" of ERM: for $N \to \infty$, ERM should lead to the same result as RM?

- No: *uniform* convergence needed (Vapnik) $\to$ VC theory.
  Thm. [classification] (Vapnik 95): with a probability of at least $1 - \eta$,

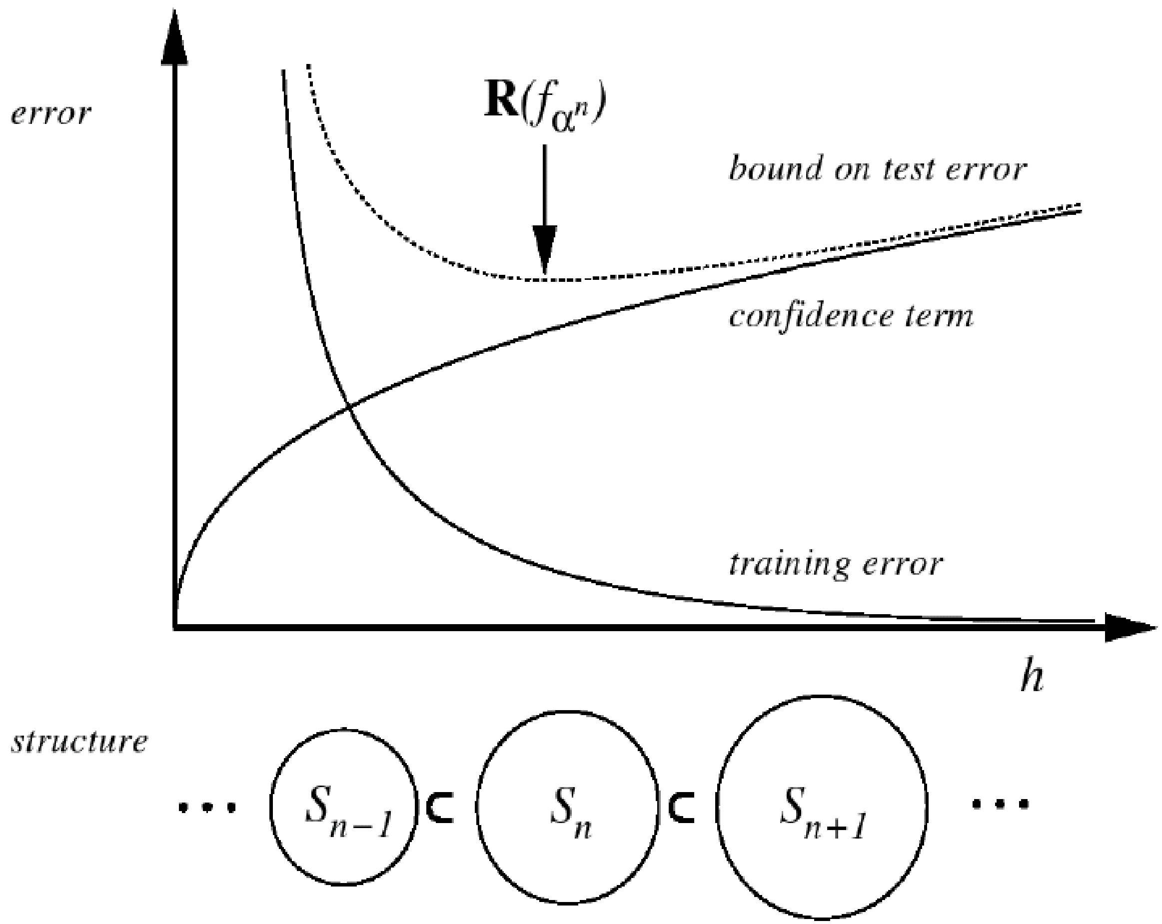
$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log \frac{2N}{d} + 1\right) - \log(\eta/4)}{N}}.$$

- Structural risk minimization (SRM): introduce structure on set of functions $\{f_\alpha\}$ & minimize RHS to get low risk! (Vapnik 95)

- d is VC dimension, measuring complexity of function class



Structural Risk Minimization: the picture



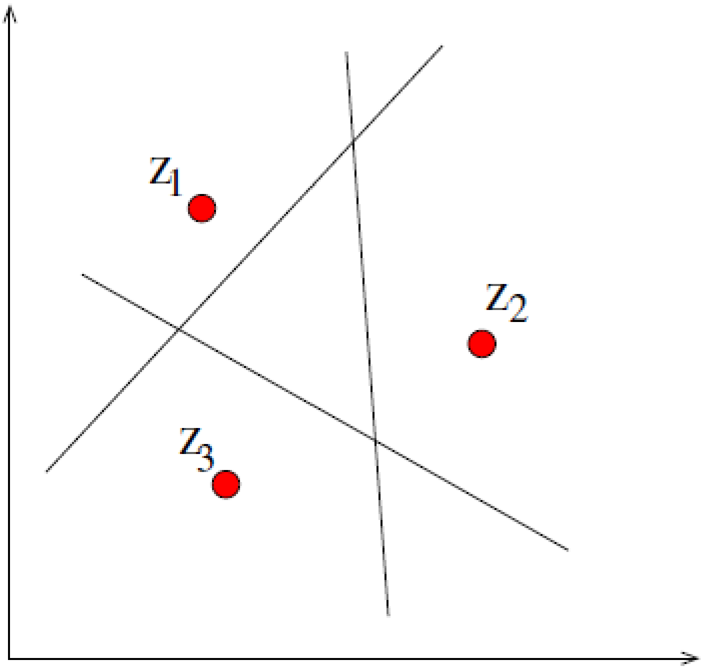
Learning f requires small training error *and* small complexity of the set $\{f_\alpha\}$.

VC Dimensions: an examples

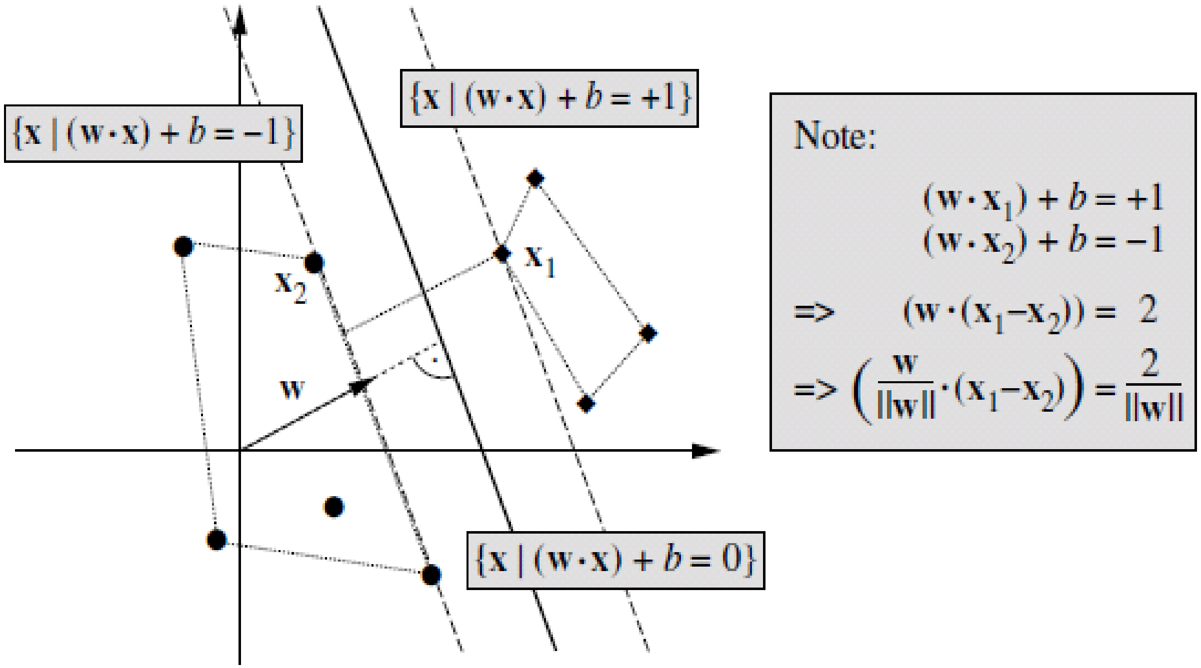
Half-spaces in $\mathbf{R}^2$:

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

- Clearly, we can shatter three non-collinear points.
- But we can never shatter four points.
- Hence the VC dimension is $d = 3$
- in n dimensions: VC dimension is $d = n + 1$



Linear Hyperplane Classifier



- hyperplane $y = \mathrm{sgn}\,(\mathrm{w} \cdot \mathrm{x} + b)$ in canonical form if $\min_{\mathrm{x}_i \in X} |(\mathrm{w} \cdot \mathrm{x}_i) + b| = 1.$, i.e. scaling freedom removed.
- larger margin $\sim 1/\|\mathrm{w}\|$ is giving better generalization $\rightarrow$ LMC!

VC Theory applied to hyperplane classifiers

- Theorem (Vapnik 95): For hyperplanes in canonical form VC–dimension satisfying

$$d \leq \min\{[R^2\|\mathbf{w}\|^2] + 1, n + 1\}.$$

Here, R is the radius of the smallest sphere containing data. Use d in SRM bound

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log\frac{2N}{d} + 1\right) - \log(\eta/4)}{N}}.$$

- maximal margin = minimum $\|\mathbf{w}\|^2$ $\rightarrow$ good generalization, i.e. low risk, i.e. optimize

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

- independent of the dimensionality of the space!

Feature Spaces & curse of dimensionality

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

$$\begin{aligned} \Phi : \mathbf{R}^N &\rightarrow F \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}) \\ \text{where } N &\ll \dim(F). \end{aligned}$$

to get data $(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_N), y_N) \in F \times \mathbf{R}^M$ or $\{\pm 1\}$.

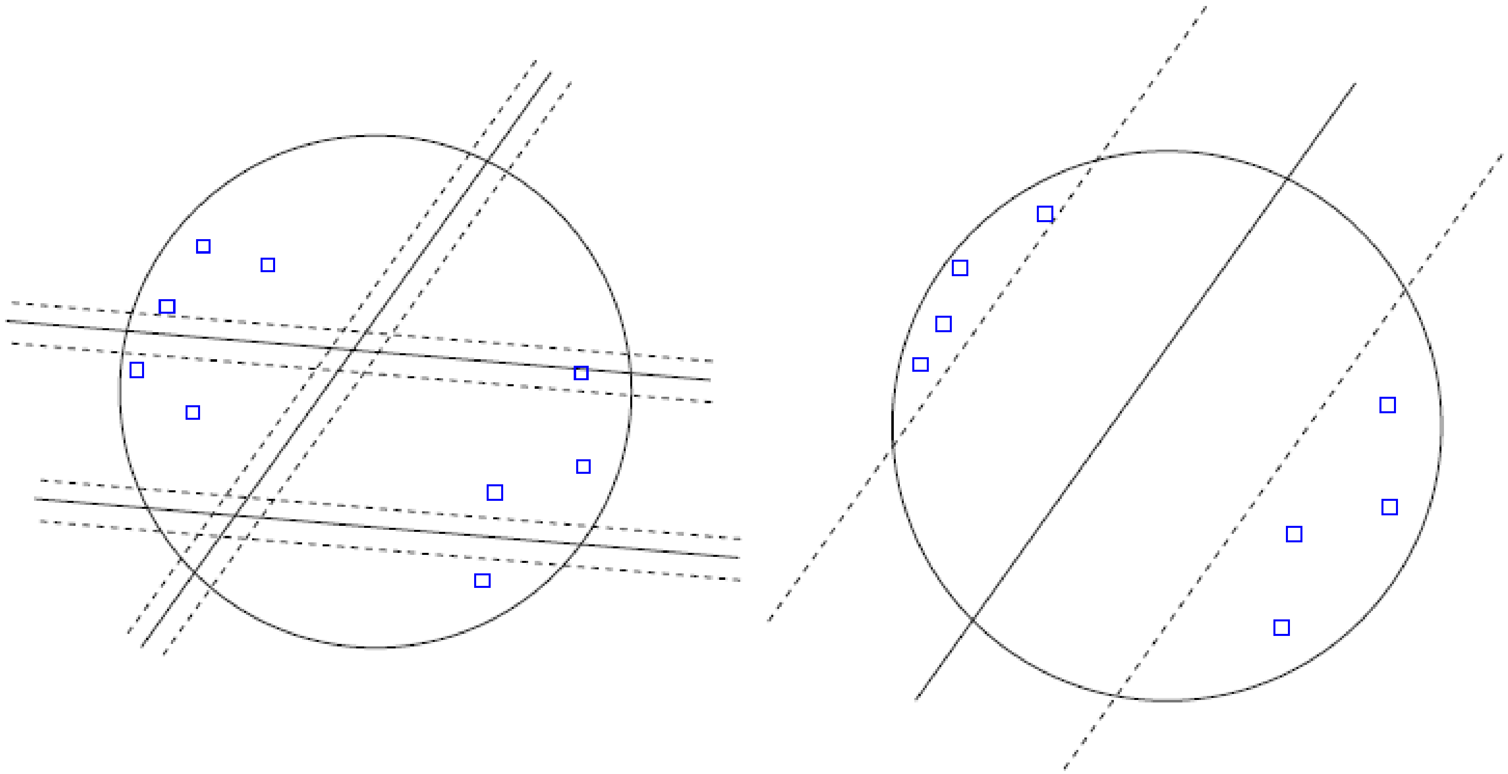
Learn $\tilde{f}$ to construct $f = \tilde{f} \circ \Phi$

- 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 complexity, then we can guarantee good generalization.

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

Margin Distributions – large margin hyperplanes



Feature Spaces & curse of dimensionality

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

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

to get data $(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_N), y_N) \in F \times \mathbf{R}^M$ or $\{\pm 1\}$.

Learn $\tilde{f}$ to construct $f = \tilde{f} \circ \Phi$

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

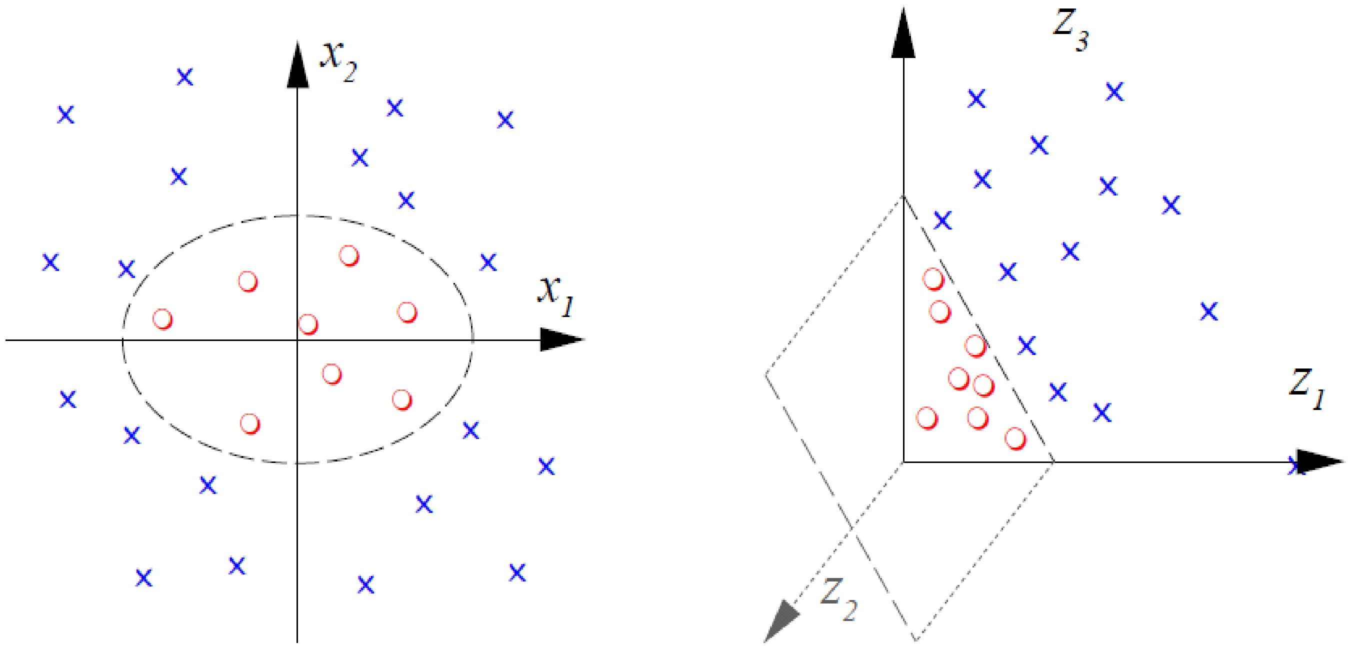
If Φ is chosen such that $\{\tilde{f}\}$ allows small training error *and* has low complexity, then we can guarantee good generalization.

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

Nonlinear Algorithms in Feature Space

Example: all second order monomials

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



The kernel trick: an example

(cf. Boser, Guyon & Vapnik 1992)

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

- Scalar product in (**high dimensional**) feature space can be computed in $\mathbf{R}^2$!
- works only for Mercer Kernels $k(\mathbf{x}, \mathbf{y})$

Kernology

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

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

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

Kernology II

Examples of common kernels:

$$
\begin{array}{rrcl}
\text{Polynomial} & k(\mathbf{x},\mathbf{y}) & = & (\mathbf{x}\cdot\mathbf{y}+c)^d \\
\cancel{\text{Sigmoid}} & \cancel{k(\mathbf{x},\mathbf{y})} & \cancel{=} & \cancel{\tanh(\kappa(\mathbf{x}\cdot\mathbf{y})+\Theta)} \\
\text{RBF} & k(\mathbf{x},\mathbf{y}) & = & \exp\left(-\|\mathbf{x}-\mathbf{y}\|^2/(2\,\sigma^2)\right) \\
\text{inverse multiquadric} & k(\mathbf{x},\mathbf{y}) & = & \dfrac{1}{\sqrt{\|\mathbf{x}-\mathbf{y}\|^2+c^2}}
\end{array}
$$

Note: kernels correspond to regularization operators (a la Tichonov) with regularization properties that can be conveniently expressed in Fourier space, e.g. Gaussian kernel corresponds to general smoothness assumption (Smola et al 98)!

A RKHS representation of $\mathcal{F}$

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

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

$$\langle \tilde{\Phi}(\mathbf{x}), \tilde{\Phi}(\mathbf{y}) \rangle = k(\mathbf{x}, \mathbf{y}), \quad \text{i.e. require} \quad \langle k(\mathbf{x}, .), k(\mathbf{y}, .) \rangle = k(\mathbf{x}, \mathbf{y}).$$

For a Mercer kernel $k(\mathbf{x}, \mathbf{y}) = \sum_j \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{y})$, 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

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

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

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

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

Hyperplane $y = \mathbf{sgn}\left(\mathbf{w} \cdot \Phi(x) + b\right)$ in $\mathcal{F}$

$$
\begin{array}{ll}
\text{min} & \|\mathbf{w}\|^2 \\
\text{subject to} & y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] \geq 1 \quad \text{for } i = 1 \ldots N
\end{array}
$$

(i.e. training data separated correctly, otherwise introduce slack variables).

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

obtain unique α_i by QP (no local minima!): dual problem

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

$$\text{i.e.} \qquad \sum_{i=1}^{N} \alpha_i y_i = 0 \qquad \text{and} \qquad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i).$$

Substitute both into L to get the *dual problem*

Hyperplane in $\mathcal{F}$ with slack variables: SVM

$$\min \quad \|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i{}^p$$

$$\text{subject to} \quad y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \quad \text{for } i = 1 \ldots N$$

(introduce slack variables if training data not separated correctly)

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

obtain unique α_i by QP (no local minima!): dual problem

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

$$\text{i.e.} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i).$$

Substitute both into L to get the *dual problem*

Dual Problem

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

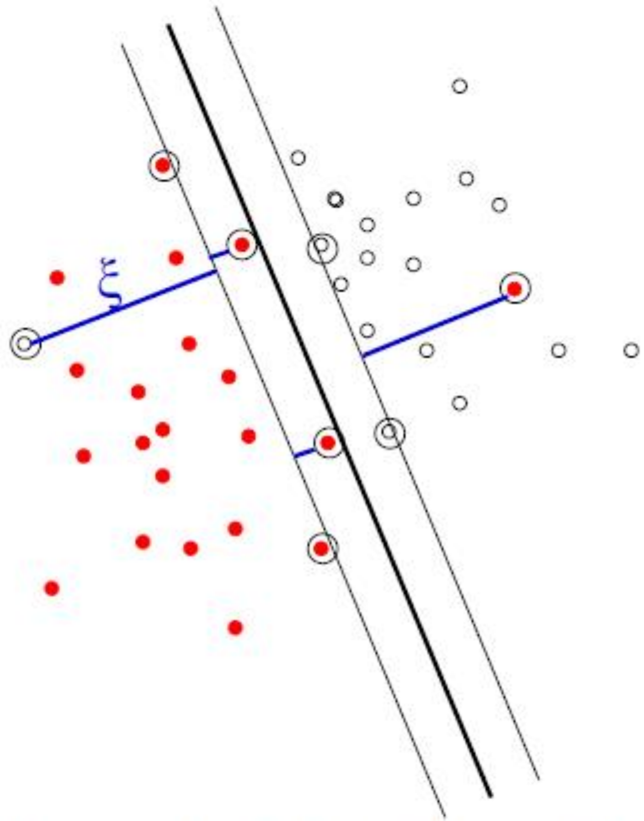
$$\text{subject to} \qquad C \geq \alpha_i \geq 0, \quad i = 1, \ldots, N, \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i y_i = 0.$$

Note: solution determined by training examples (SVs) on /in the margin. Remark: duality gap.

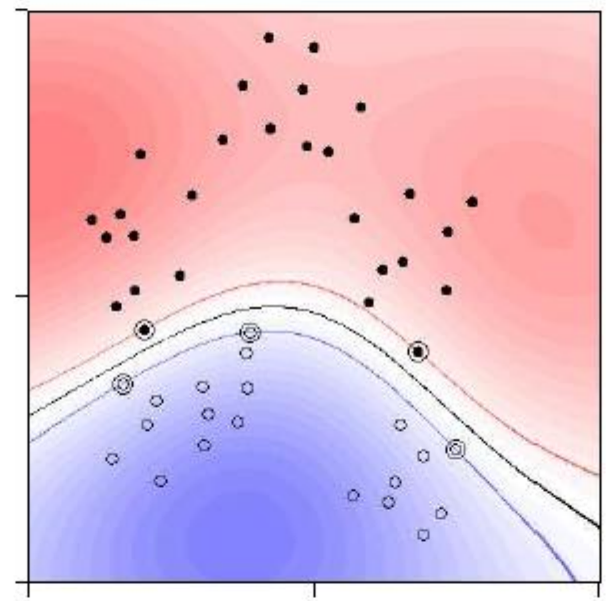
$$y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] > 1 \qquad \Longrightarrow \alpha_i = 0 \longrightarrow \quad \mathbf{x}_i \text{ irrelevant or}$$

$$y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] = 1 \quad (on\ /\text{in margin}) \longrightarrow \quad \mathbf{x}_i \text{ Support Vector}$$

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



linear SV with slack variables



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

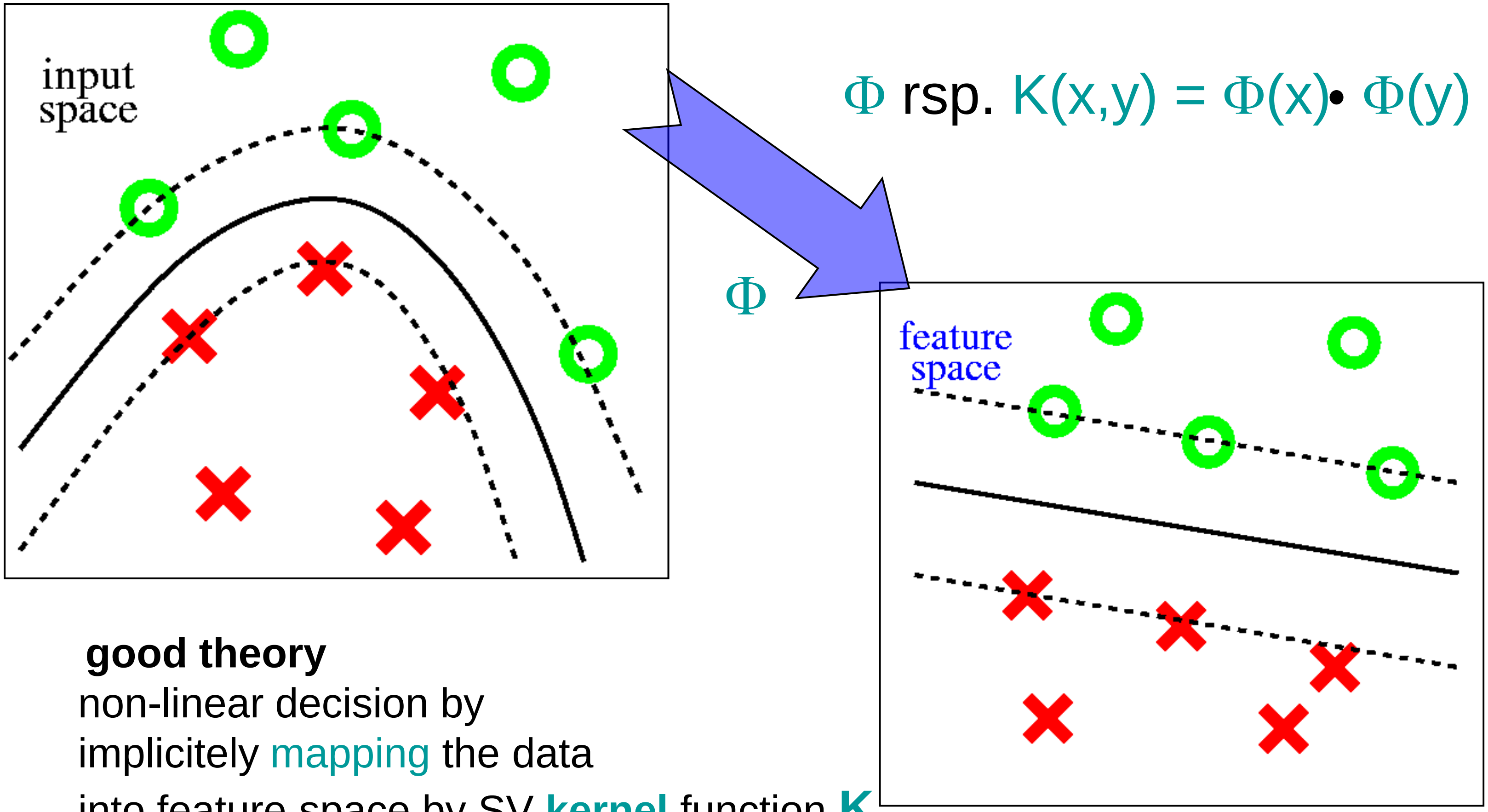
Kernel Trick

- Saddle Point: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$.
- Hyperplane in $\mathcal{F}$: $y = \operatorname{sgn}\left(\mathbf{w} \cdot \Phi(x) + b\right)$
- putting things together "kernel trick"

$$
\begin{aligned}
f(\mathbf{x}) &= \operatorname{sgn}\left(\mathbf{w} \cdot \Phi(\mathbf{x}) + b\right) \\
&= \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i \, \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b\right) \\
&= \operatorname{sgn}\left(\sum_{i \in \#\mathrm{SVs}} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b\right) \qquad \textbf{sparse!}
\end{aligned}
$$

- trick: $k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$, i.e. **never use Φ: only k!!!**

Support Vector Machines in a nutshell



Φ rsp. $K(x,y) = \Phi(x) \bullet \Phi(y)$

good theory
non-linear decision by
implicitely mapping the data
into feature space by SV **kernel** function **K**

Kernels ...

- kernels hold key to learning problem.
- chosing kernels ...
  - Mercer condition (ℓ_2 integrability & positivity)
  - kernel reflects prior (Smola, Schölkopf & Müller 98, Girosi 98)
  - approximating LOO bounds give good model selection results (Tsuda et al. 2001, Vapnik & Chapelle 2000)
- So: **engineer** an appropriate kernel from prior knowledge! (Jaakola and Haussler 1998, Watkins 2000, Zien et al 2000, recently a large body of interesting work)
- And: use **careful** model selection to find appropriate kernel parameters, i.e. chose appropriate degree of polynomial or bandwidth of Gaussian kernel

Digestion: Use of kernels

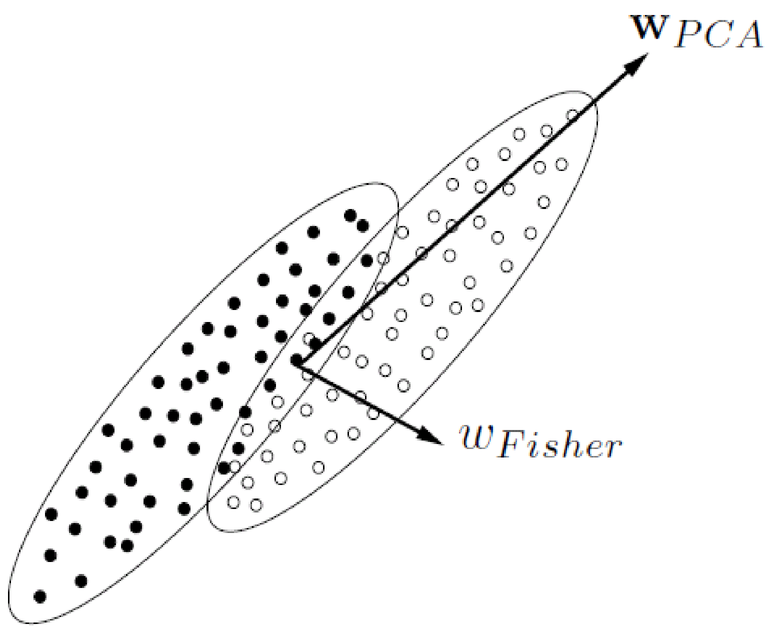
- Question: What makes kernel methods (e.g. SVM) perform well?
- Answer:
  - In the first place: a good idea/theory.
  - But also: The kernel
- Using kernels, we work explicitly in extremely high dimensional spaces (RKHS) with interesting features for themselves (depending on the kernel) [SSM et al. 98]
- Common choices: Gaussian kernel $\exp(-\|\boldsymbol{x}-\boldsymbol{y}\|^2/c)$ or polynomial kernel $(\boldsymbol{x}\cdot\boldsymbol{y})^d$.
- Almost any linear algorithm can be transformed to feature space. [SSM et al. 98]
- With suitable regularization it outperforms its linear counterpart. [Mika et al. 02]

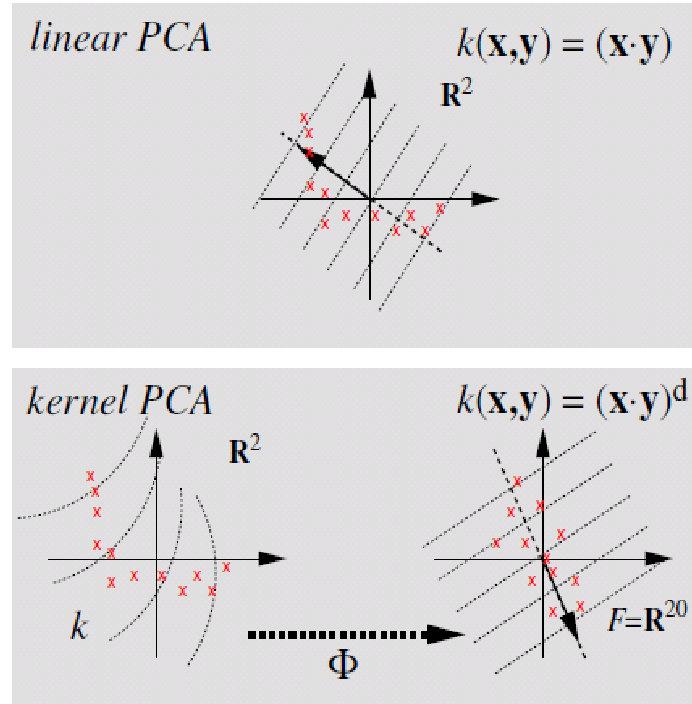
[Zien et al. 00, Tsuda et al. 02, Sonnenburg et al. 05]

- The kernel can be adopted to specific tasks, e.g. using prior knowledge

Kernels for graphs, trees, strings etc.

Remark: Kernelizing linear algorithms



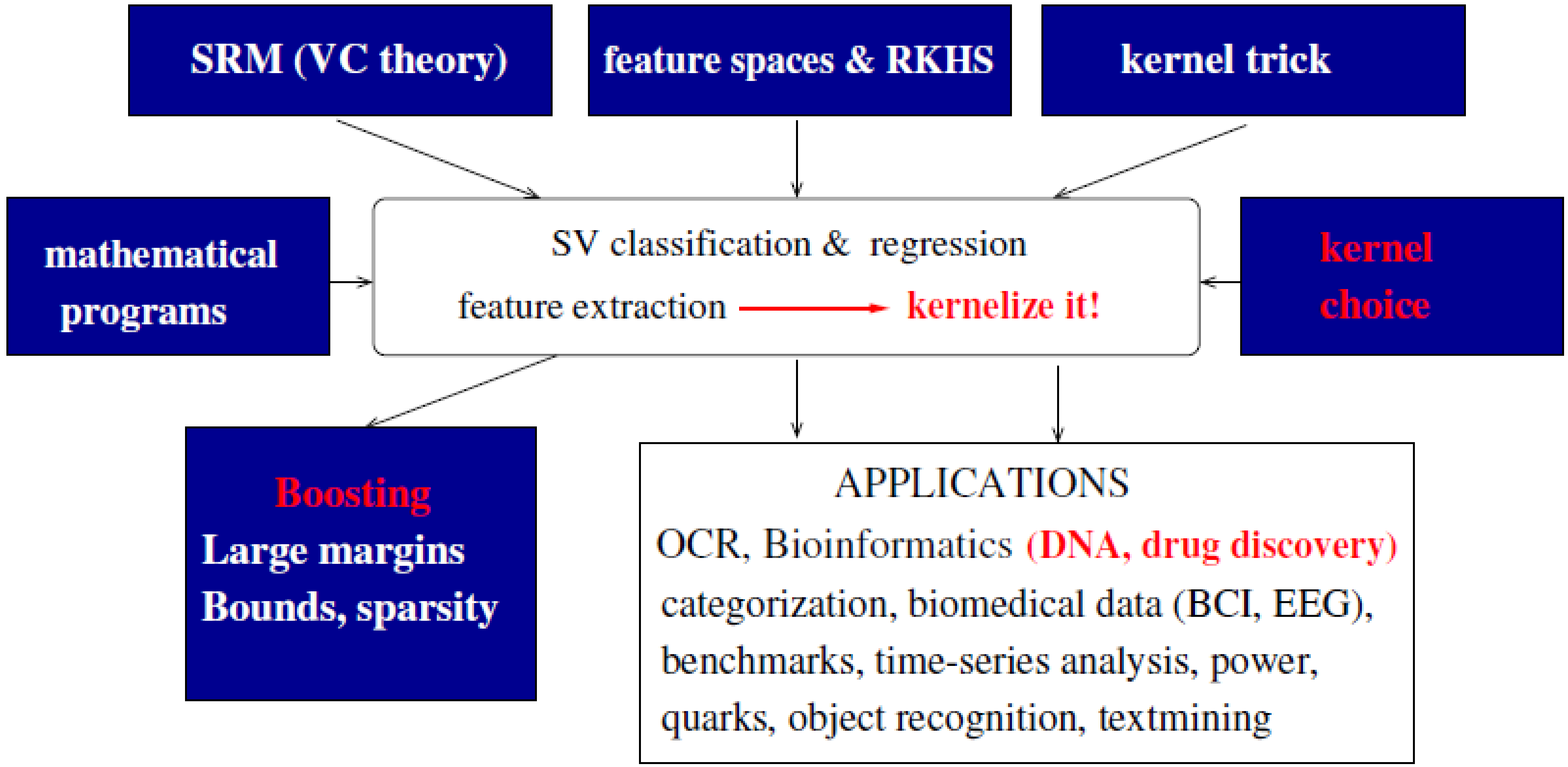


(cf. Schölkopf, Smola and Müller 1996, 1998, Schölkopf et al 1999, Mika et al, 1999, 2000, 2001, Müller et al 2001, Harmeling et al 2003, ...)

Digestion

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log\frac{2N}{d}+1\right)-\log(\eta/4)}{N}}$$

$$K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$$



Part II

Implementation Issues: working set methods

matrix notation: Let $\boldsymbol{\alpha} = (\alpha_1, \ldots \alpha_M)^\top$, let $\mathbf{y} = (y_1, \ldots, y_M)^\top$, let H be the matrix with the entries $H_{ij} = y_i y_j \,\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)$, and let $\mathbf{1}$ denote the vector of length M consisting of all 1s.

dual SVM Problem becomes:

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top H \boldsymbol{\alpha}, && (1)\\
\text{subject to} \quad & \mathbf{y}^\top \boldsymbol{\alpha} = 0, && (2)\\
& \boldsymbol{\alpha} - C\mathbf{1} \leq 0, && (3)\\
& \boldsymbol{\alpha} \geq 0. && (4)
\end{aligned}
$$

Implementation Issues: working set methods II

$\boldsymbol{\alpha}_B$ of the variables in the working set at a current iteration and $\boldsymbol{\alpha}_N$ remaining variables. H is thus partitioned as $H = \begin{bmatrix} H_{BB} & H_{BN} \\ H_{NB} & H_{NN} \end{bmatrix}$, at each iteration, is obtained:

$$\max_{\boldsymbol{\alpha}} \quad (\mathbf{1}_B^\top - \boldsymbol{\alpha}_N^\top H_{NB})\boldsymbol{\alpha}_B - \frac{1}{2}\boldsymbol{\alpha}_B^\top H_{BB}\boldsymbol{\alpha}_B, \tag{5}$$

$$\text{subject to} \quad \mathbf{y}_B^\top \boldsymbol{\alpha}_B = -\mathbf{y}_N \boldsymbol{\alpha}_N, \tag{6}$$

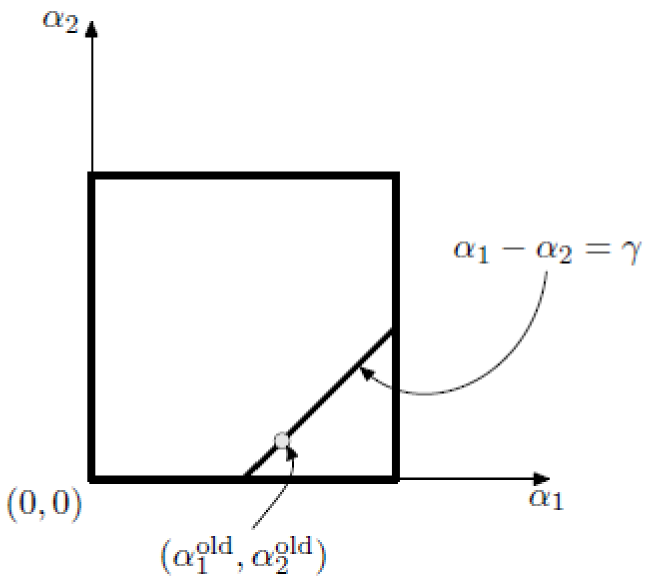
$$\boldsymbol{\alpha}_B - C\mathbf{1}_B \leq 0, \tag{7}$$

$$\boldsymbol{\alpha}_B \geq 0. \tag{8}$$

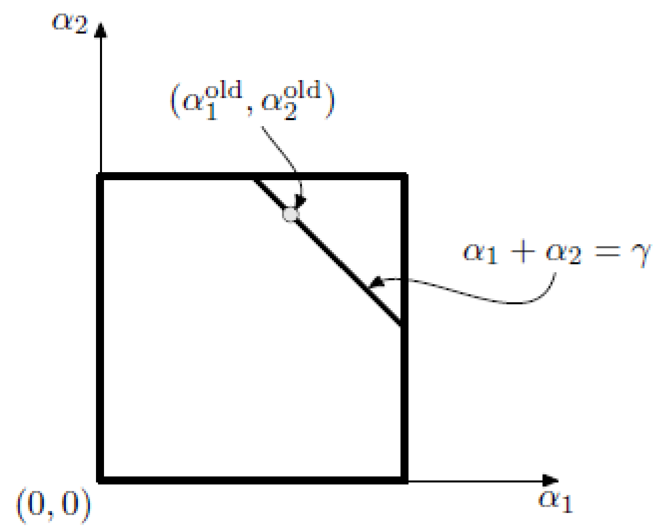
Usuall small working set, iteration is carried out until KKT conditions, are satisfied to the required precision ϵ. monitor gap.

John Platt's SMO

- extreme: use only two points in working set and compute optimal solution *analytically*



$y_1 \neq y_2$



$y_1 = y_2$

SMO continued

eliminating α_1 yields update rule for α_2:

$$\alpha_2^{\text{new}} = \alpha_2^{\text{old}} - \frac{y_2(E_1 - E_2)}{\eta}, \tag{9}$$

where

$$E_1 = \textstyle\sum_{j=1}^{M} y_j \alpha_j \, \mathrm{k}(\mathbf{x}_1, \mathbf{x}_j) + b - y_1, \tag{10}$$

$$E_2 = \textstyle\sum_{j=1}^{M} y_j \alpha_j \, \mathrm{k}(\mathbf{x}_2, \mathbf{x}_j) + b - y_2, \tag{11}$$

$$\eta = 2\,\mathrm{k}(\mathbf{x}_1, \mathbf{x}_2) - \mathrm{k}(\mathbf{x}_1, \mathbf{x}_1) - \mathrm{k}(\mathbf{x}_2, \mathbf{x}_2). \tag{12}$$

Next, the bound constraints should be taken care of. Depending on the geometry, one computes the following lower and upper bounds on the value of the variable α_2:

$$L = \begin{cases} \max\left(0, \alpha_2^{\text{old}} - \alpha_1^{\text{old}}\right), & \text{if } y_1 \neq y_2, \\ \max\left(0, \alpha_2^{\text{old}} + \alpha_1^{\text{old}} - C\right), & \text{if } y_1 = y_2, \end{cases}$$

$$H = \begin{cases} \min\left(C, C + \alpha_2^{\text{old}} - \alpha_1^{\text{old}}\right), & \text{if } y_1 \neq y_2, \\ \min\left(C, \alpha_2^{\text{old}} + \alpha_1^{\text{old}}\right), & \text{if } y_1 = y_2. \end{cases}$$

The constrained optimum is then found by clipping the unconstrained optimum to the ends of the line segment:

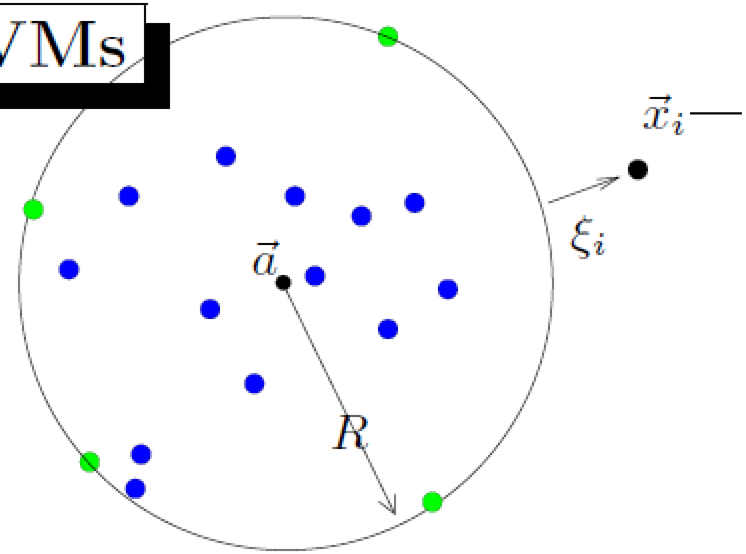
$$\alpha_2^{\text{new}} := \begin{cases} H, & \text{if } \alpha_2^{\text{new}} \geq H, \\ L, & \text{if } \alpha_2^{\text{new}} \leq L, \\ \alpha_2^{\text{new}}, & \text{otherwise.} \end{cases}$$

Finally, the value of α_1^{new} is computed:

$$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + y_1 y_2 (\alpha_2^{\text{old}} - \alpha_2^{\text{new}}). \tag{13}$$

- Use heuristics to choose examples

One-Class SVMs



Fitting a hypersphere around the data

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{M} \alpha_i \,\mathrm{k}(\mathbf{x}_i, \mathbf{x}_i) - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j \,\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j), \tag{14}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, \; i = 1, \ldots, M,$$

$$\sum_{i=1}^{M} \alpha_i = 1.$$

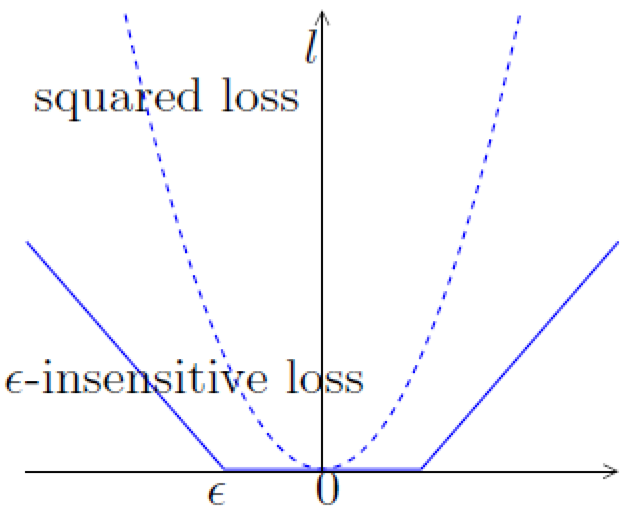
new object belongs to target class? (cf. Tax 01, Schölkopf et al. 01)

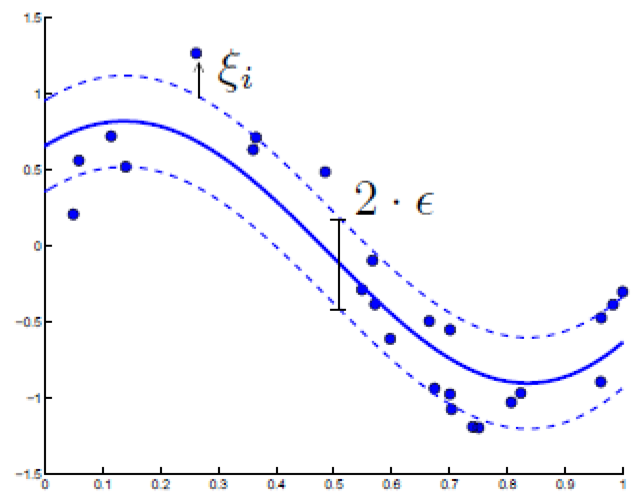
$$f(\mathbf{x}) = \mathrm{sign}(R^2 - \mathrm{k}(\mathbf{x}, \mathbf{x}) + 2 \sum_{i} \alpha_i \,\mathrm{k}(\mathbf{x}, \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j \,\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)). \tag{15}$$

SVMs for Regression

$$\ell(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2,$$

$$\ell(f(\mathbf{x}), y) = \begin{cases} |f(\mathbf{x}) - y| - \epsilon, & \text{if } |f(\mathbf{x}) - y| > \epsilon, \\ 0, & \text{otherwise.} \end{cases}$$





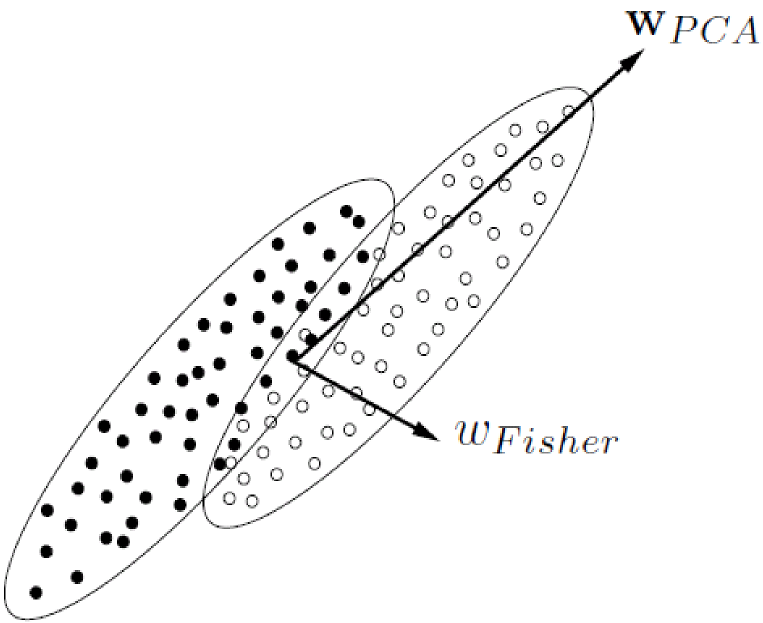
(cf. Vapnik 95, Smola and Schölkopf 02)

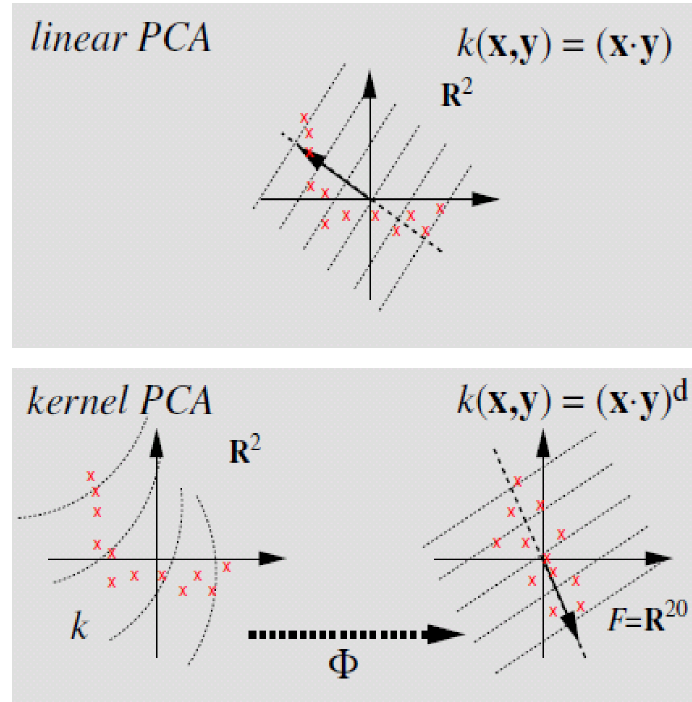
SVMs for Regression II

The primal formulation for the SVR

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}^{(*)}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{M}(\xi_i + \xi_i^*), \\ \text{subject to} \quad & ((\mathbf{w}^\top \mathbf{x}_i) + b) - y_i \leq \epsilon + \xi_i, \\ & y_i - ((\mathbf{w}^\top \mathbf{x}_i) + b) \leq \epsilon + \xi_i^*, \\ & \xi_i^{(*)} \geq 0, \quad i = 1, \ldots, M. \end{aligned}$$

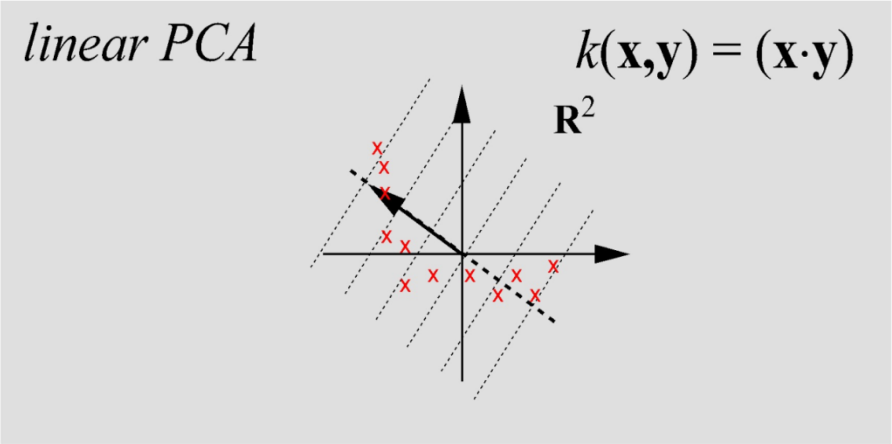
Remark: Kernelizing linear algorithms

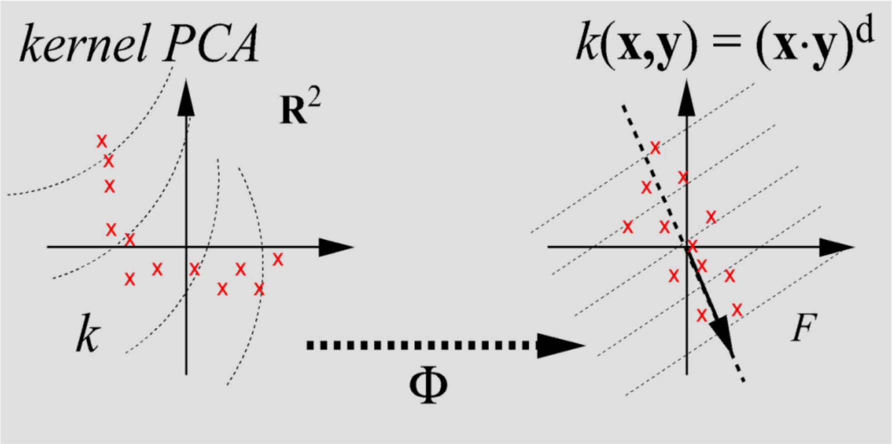




(cf. Schölkopf, Smola and Müller 1996, 1998, Schölkopf et al 1999, Mika et al, 1999, 2000, 2001, Müller et al 2001, Harmeling et al 2003, ...)

Kernel PCA





[cf. Schölkopf, Smola, Müller 98]

PCA in high dimensional feature spaces

$$\mathbf{x}_1, \ldots, \mathbf{x}_N, \qquad \Phi : \mathbb{R}^D \rightarrow F, \qquad C = \frac{1}{N}\sum_{j=1}^{N} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^\top$$

Eigenvalue problem

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

For $\lambda \neq 0$, $\mathbf{V} \in \mathrm{span}\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_N)\}$, thus $\mathbf{V} = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i)$.

Multiplying with $\Phi(\mathbf{x}_k)$ from the left yields

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

Nonlinear PCA as an Eigenvalue problem

Define an $N \times N$ matrix

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

to get

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

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

Solve

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

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

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

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 ("kernel PCA components")

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

Centering in Feature Space

Center the data in F:

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

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

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

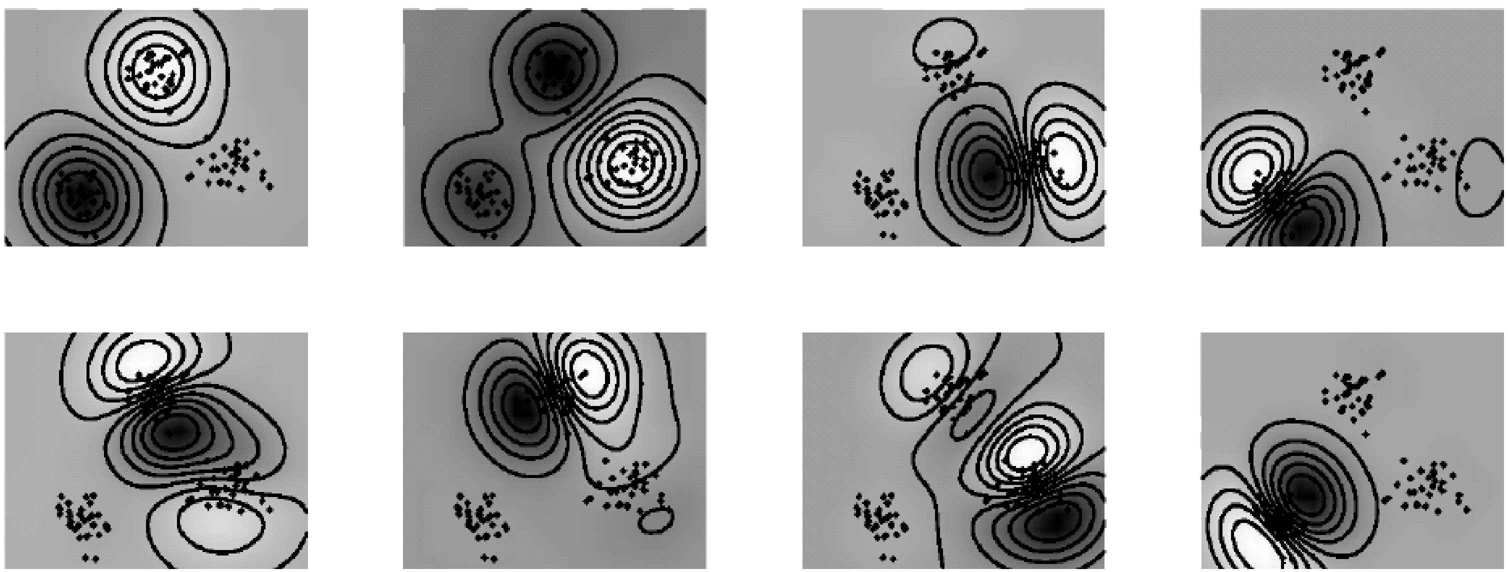
$$\tilde{K}_{ij} = K - 1_N K - K 1_N + 1_N K 1_N.$$

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

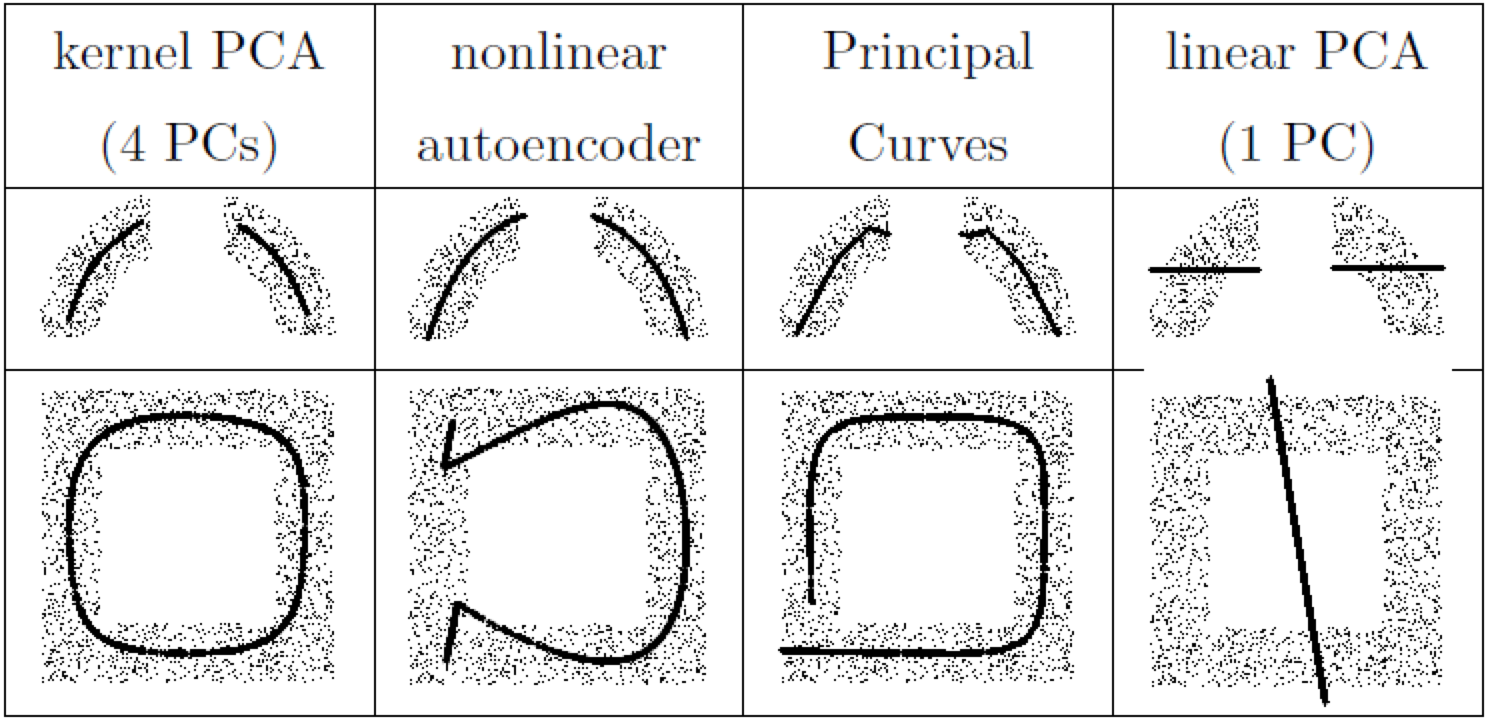
Similar for feature extraction.

Example: 8 kPCA components with RBF kernel

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



Denoising



Principal curves: Hastie & Stützle, 1989

Nonlinear autoencoder: e.g. Kramer, 1991

Denoising II