An Introduction to Kernel-based Learning

K.-R. Müller[1,2], G. Rätsch[1,3], S. Mika[1], A. Ziehe[1], M. Kawanabe[1], J. Laub[1], S. Harmeling[1], with B. Schölkopf[4], A. J. Smola[3], V. Vapnik[5]

[1] *FHG FIRST, Berlin and* [2] *University of Potsdam, Germany*

[3] *Australian National University, Canberra, Australia*

[4] *Barnhill Tech., Georgia, USA and* [5] *AT&T Research Labs, NJ, USA*

[6] *AIST Computational Biology Research Center, Tokyo, Japan*

[1,2] **http://www.first.gmd.de/persons/Mueller.Klaus-Robert.html**

http://www.kernel-machines.org and **http://www.boosting.org**

Content of this talk

- kernel based learning
  - basic ideas: VC theory & some bounds
  - kernel feature spaces & "kernel trick"
  - kernelizing the perceptron: Support Vector Machines (SVM)
- applications of kernel based learning
  - kernel PCA
  - kernel ICA (Stefan Harmeling's talk)

Goal: how to classify in infinite dimensional spaces and how to beat the "curse of dimensionality", kernelizing

Basic ideas of statistical learning theory I

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

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

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 of statistical learning theory II

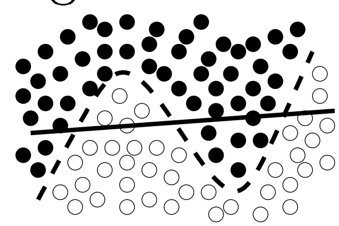
- Law of large numbers: $R_{emp}[f] \to R[f]$ as $N \to \infty$.
  "consistency" of ERM: for $N \to \infty$, ERM should lead to the same result as RM?

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

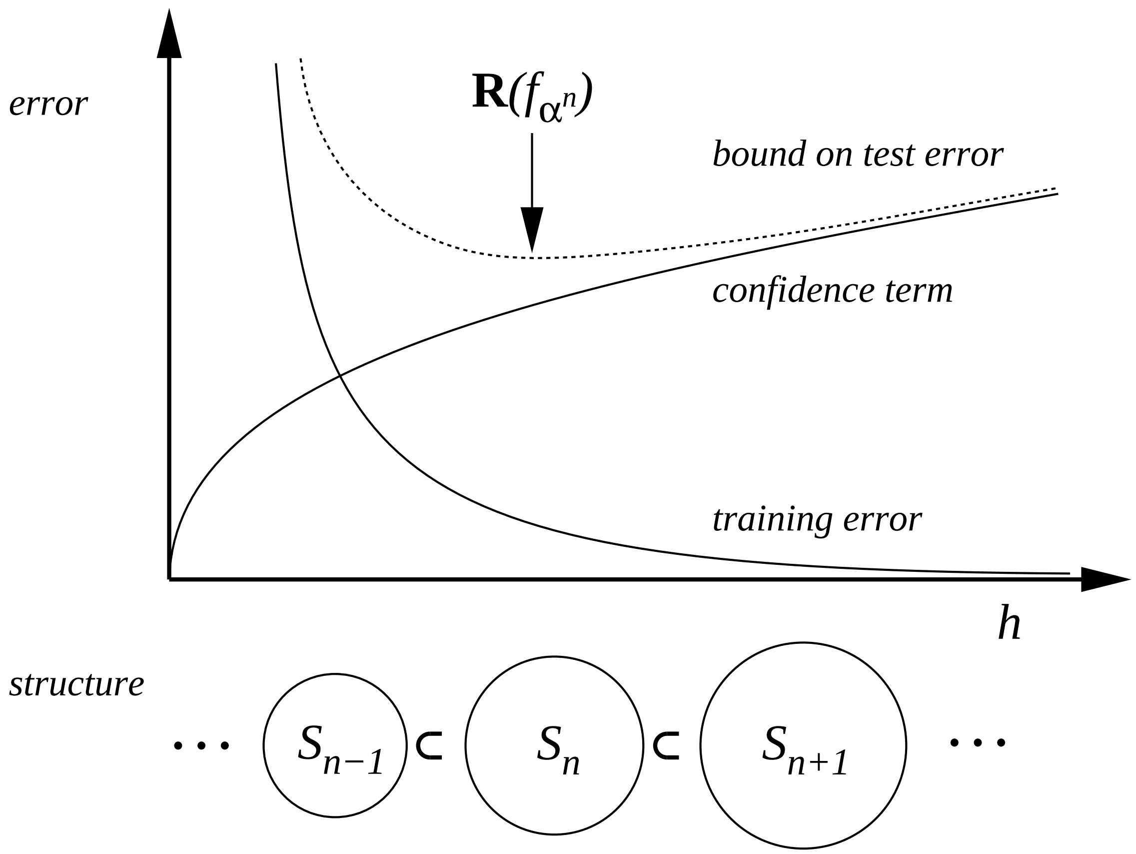
$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log \frac{2N}{d} + 1\right) - \log(\eta/4)}{N}}.$$

- 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

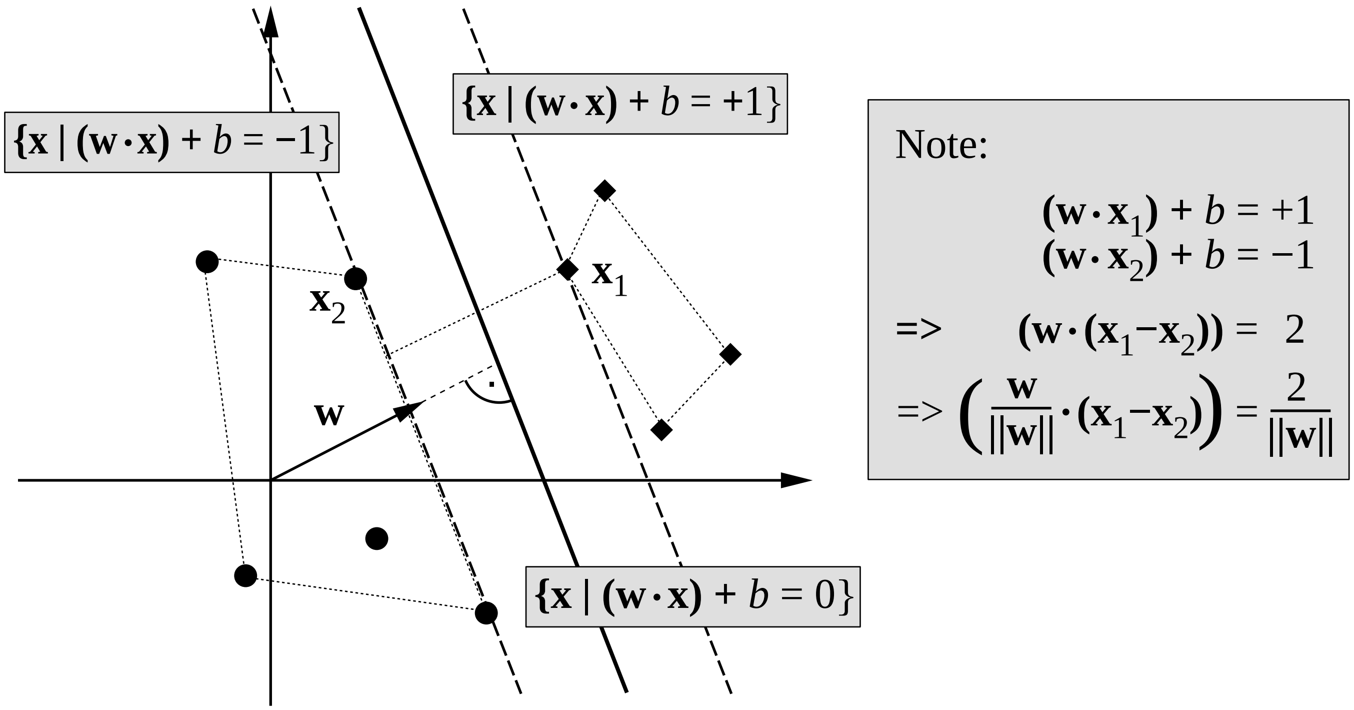


SRM: the picture



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

linear hyperplane classifier



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

Applying VC theory to hyperplanes

- **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 $\|\mathbf{w}\|^2$ → good generalization, i.e. low risk, i.e. optimize

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

- **independent of the dimensionality of the space!**

Feature Spaces and "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**:

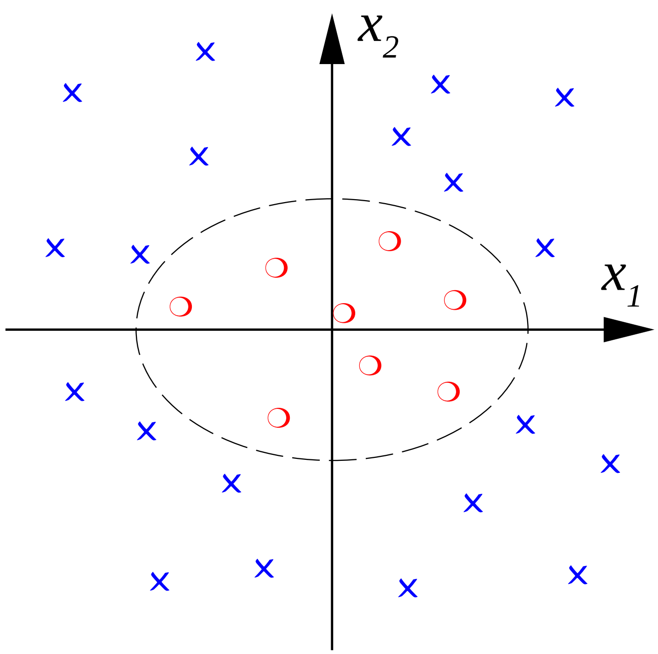
If Φ is chosen such that $\{\tilde{f}\}$ allows small training error *and* has low complexity, then we can guarantee good generalization.

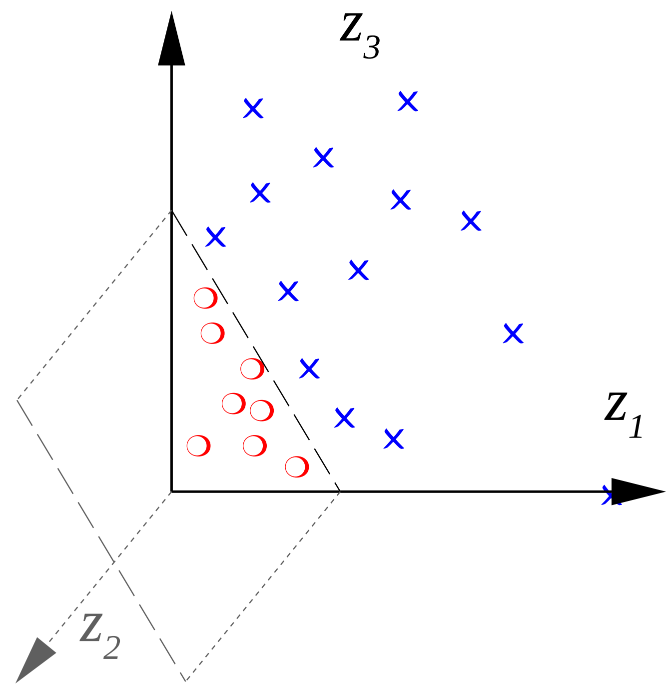
The *complexity* matters, not the *dimensionality* of the space.

Nonlinear Algorithms in Feature Spaces

Example: all second order monomials

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





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 I

[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:

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

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, see L 3)!

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 $\mathcal{F}$

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

(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

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

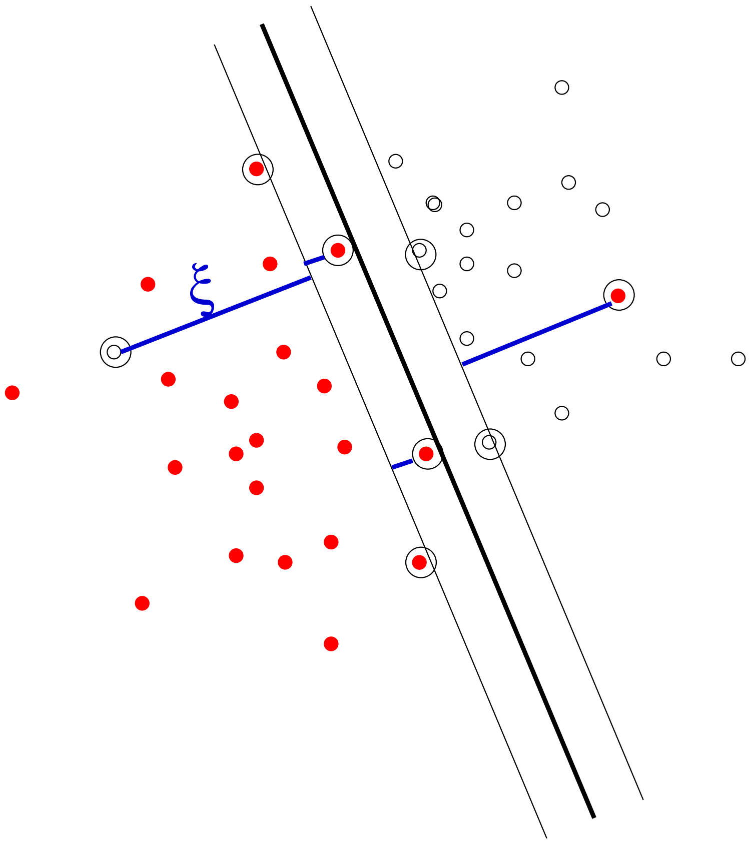
subject to $$C \geq \alpha_i \geq 0, \quad i = 1, \ldots, N, \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i y_i = 0.$$

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

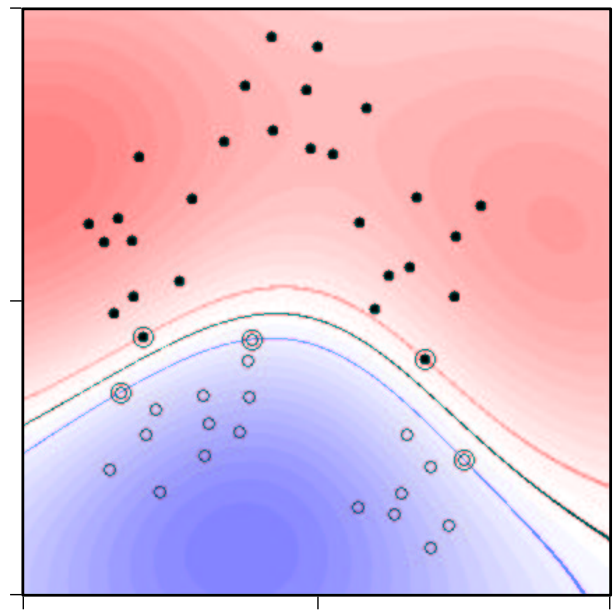
$$y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] > 1 \quad \Longrightarrow \alpha_i = 0 \longrightarrow \quad \mathbf{x}_i \text{ irrelevant or}$$

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

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 $\mathcal{F}$: $y = \operatorname{sgn}(\mathbf{w} \cdot \Phi(x) + b)$
- putting things together "kernel trick"

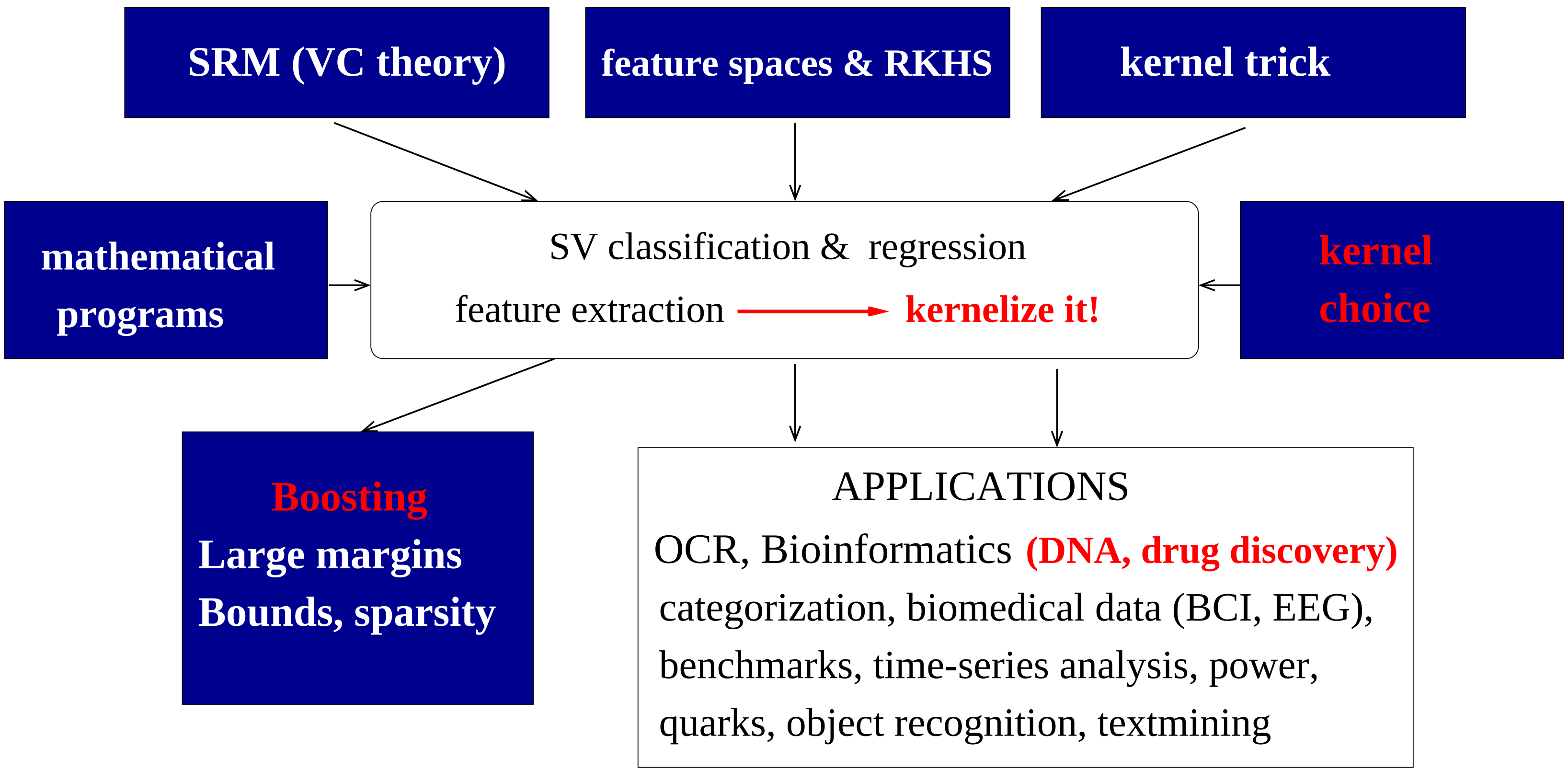
$$
\begin{aligned}
f(\mathbf{x}) &= \operatorname{sgn}(\mathbf{w} \cdot \Phi(\mathbf{x}) + b) \\
&= \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i \, \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b \right) \\
&= \operatorname{sgn}\left(\sum_{i \in \#\mathrm{SVs}} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i)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

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

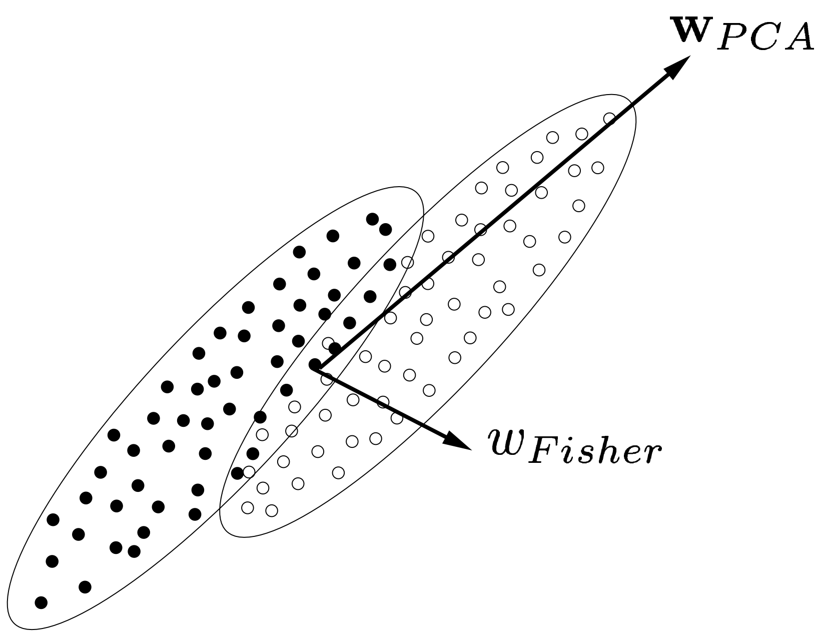
Digestion

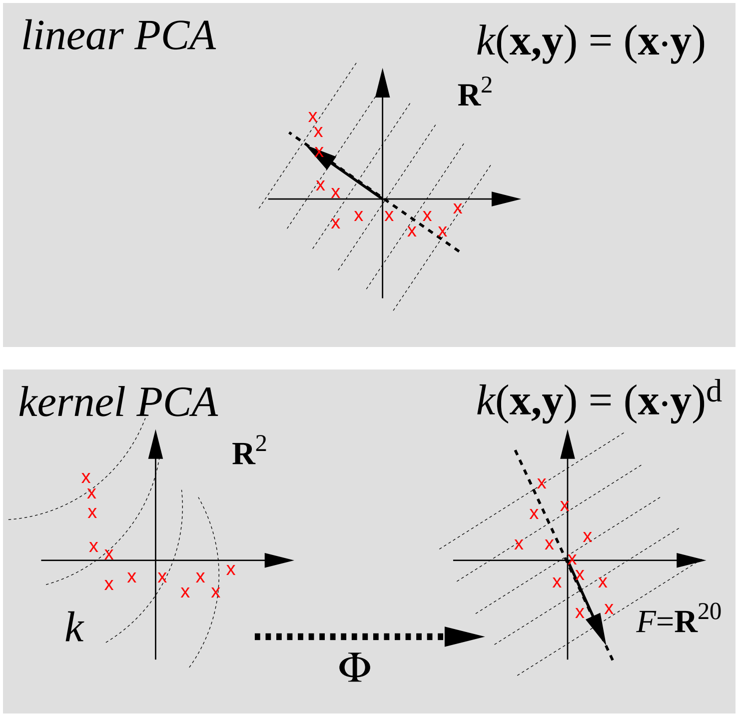
$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log\frac{2N}{d}+1\right) - \log(\eta/4)}{N}}$$

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



Kernelizing linear algorithms





Kernel ICA (Stefan's talk)

PCA in High-dimensional Feature Spaces

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

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

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

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

Solve

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

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

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

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

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

Centering in F

Center the data in F:

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

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

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

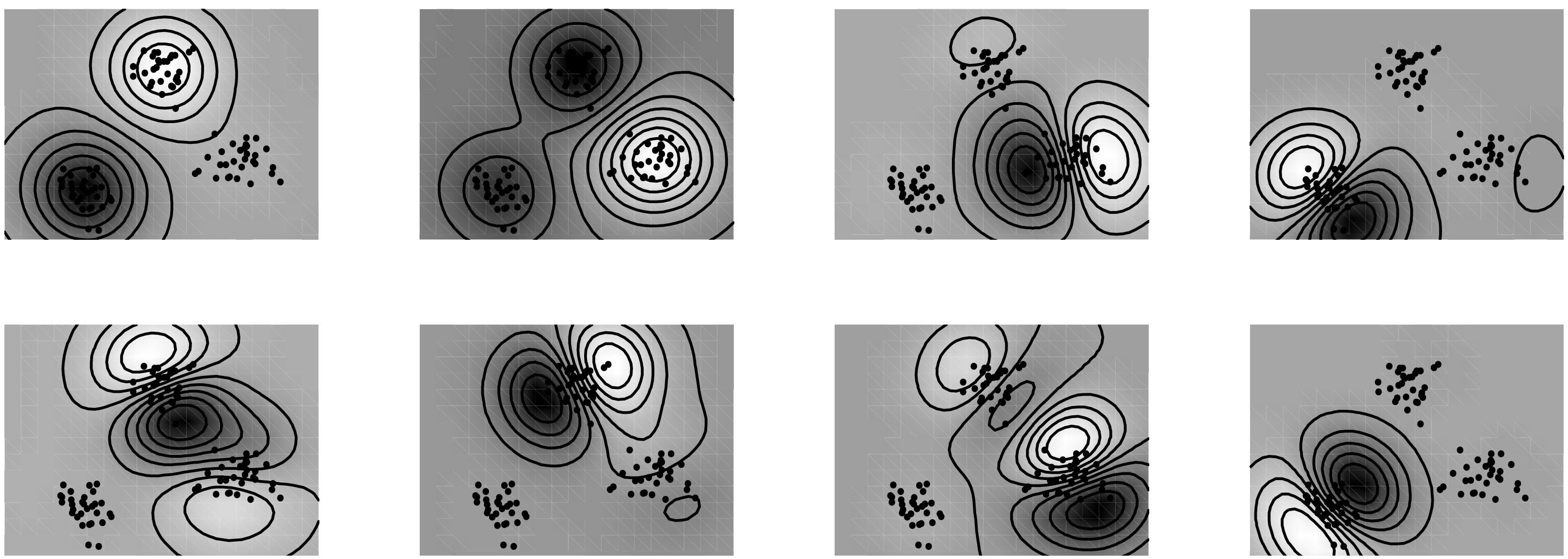
$$\tilde{K}_{ij} = K - 1_N K - K 1_N + 1_N K 1_N.$$

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

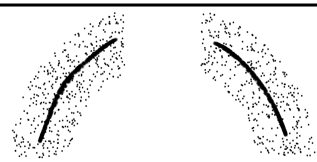
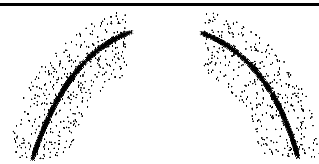
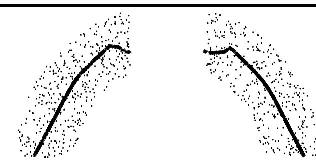
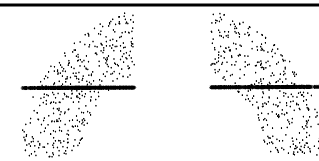
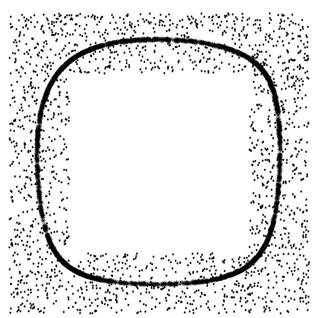
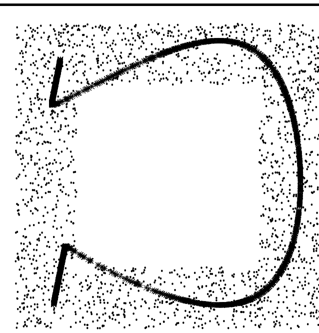
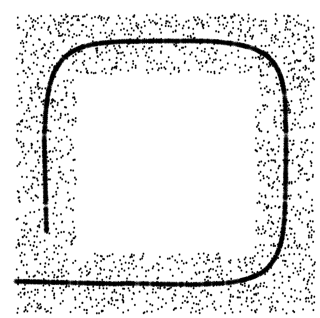
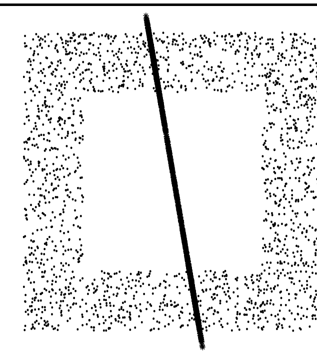
Similar for feature extraction.

Example: RBF Kernel, 8 Principal Components

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

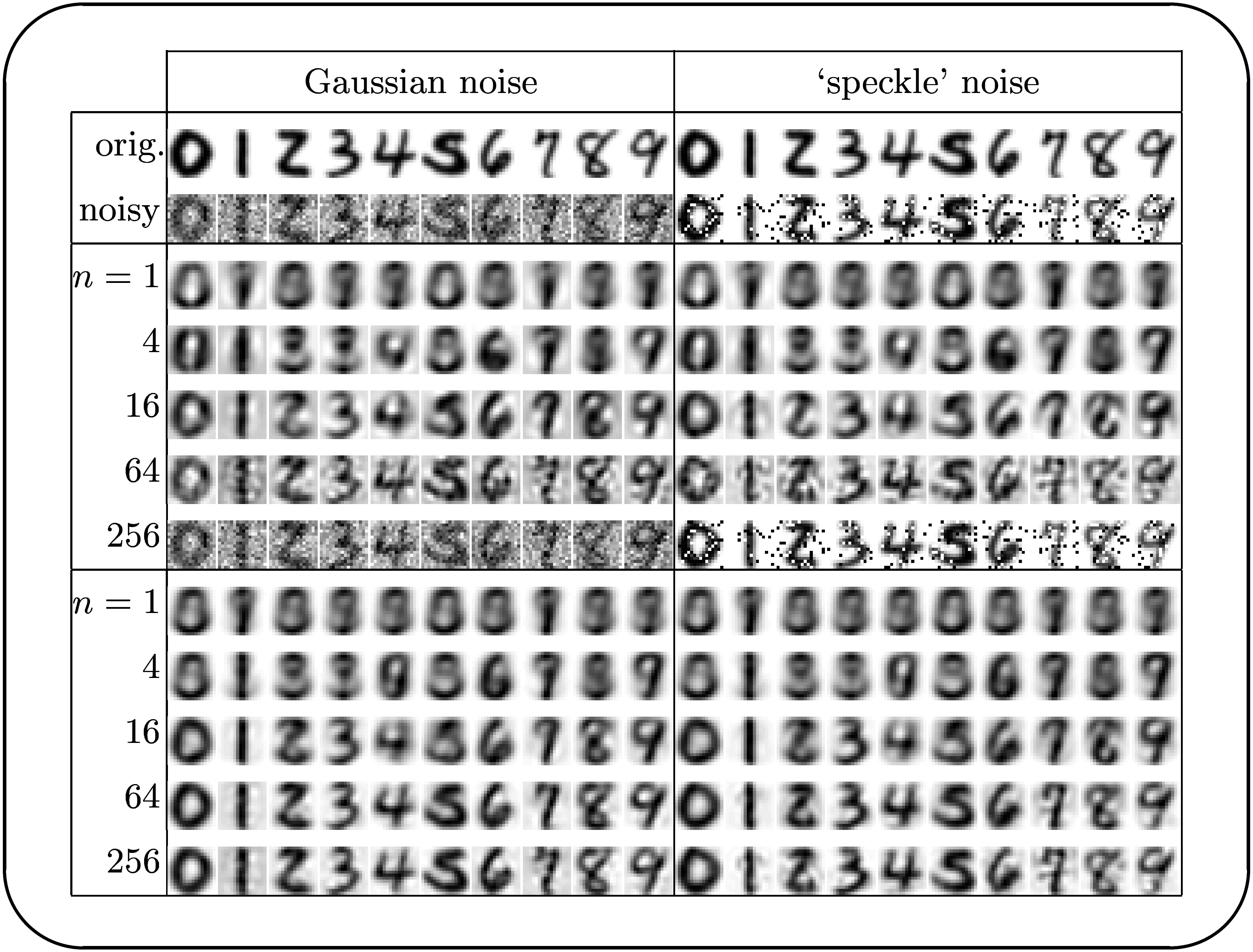


Denoising

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

Principal curves: Hastie & Stützle, 1989

Nonlinear autoencoder: e.g. Kramer, 1991


Gaussian noise
‘speckle’ noise
orig.
noisy
n = 1
4
16
64
256
n = 1
4
16
64
256

Conclusions & Outlook

- kernelize linear algorithms (Schölkopf, Smola and Müller 1996): Kernel PCA, Kernel Fisher, Clustering, ICA ...
- applications
  - OCR, time-series, finance,
  - DNA/protein analysis, drug discovery, BCI

`Support Vector Homepage:` **http://www.kernel-machines.org**

(former **http://svm.first.gmd.de**)