Kernel-based Learning and Applications

K.-R. Müller,[1,2] G. Rätsch,[1,2] S. Mika,[1]

K. Tsuda,[4,6] M. Kawanabe,[1] J. Laub[1]

with B. Schölkopf,[4] A. J. Smola,[3] V. Vapnik[5]

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

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

[4] *MPI Tübingen, Germany and* [5] *NEC Research, Princeton, 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

- introduction to Fraunhofer FIRST.IDA
- Support Vector Machines (SVM)
  - basic ideas: VC theory & some bounds
  - kernel feature spaces & "kernel trick"
- applications of SVMs
  - constructing kernels
  - protein superfamilies, splice sites, BCI, OCR
- Further/future interests

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

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

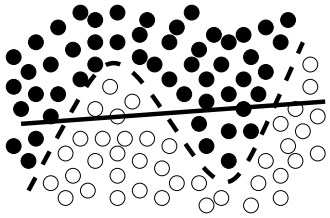
- Law of large numbers: $R_{emp}[f] \to R[f]$ as $N \to \infty$.
  "consistency" of ERM: for $N \to \infty$, ERM should lead to the same result as RM?

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

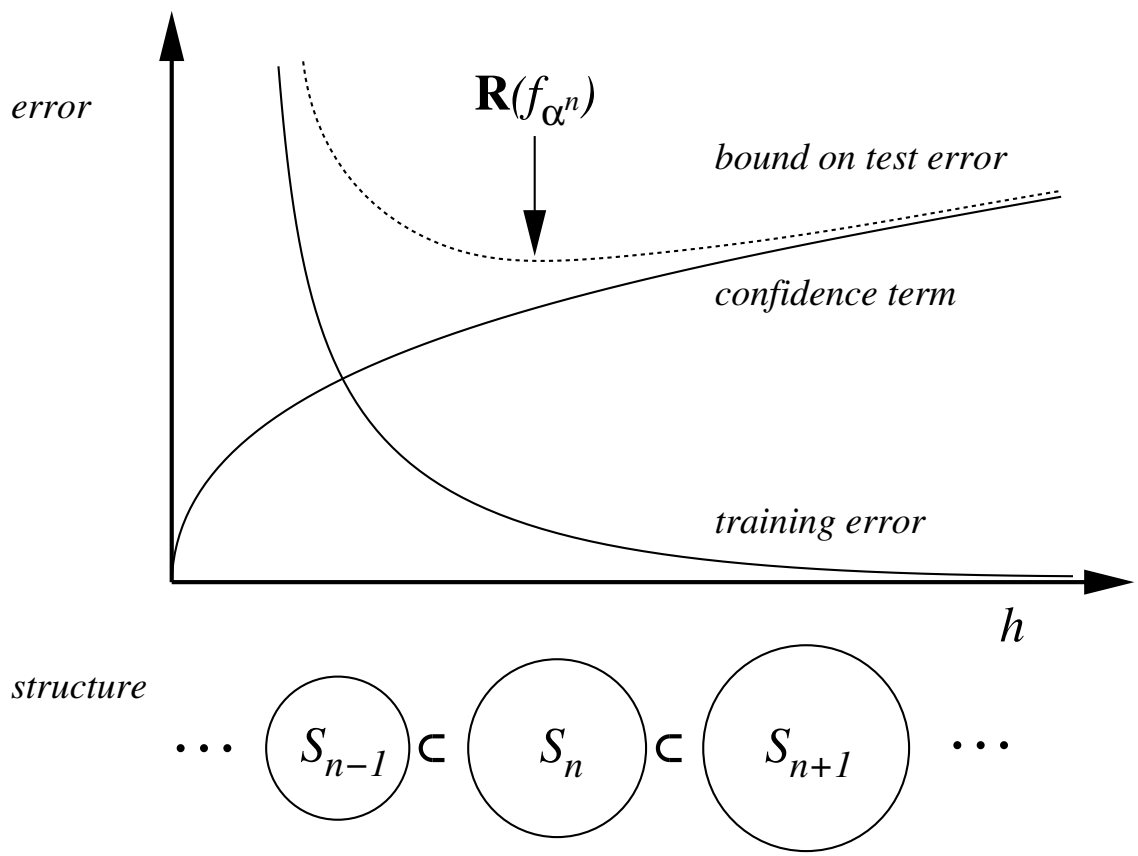
$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log \frac{2N}{d} + 1\right) - \log(\eta/4)}{N}}.$$

- 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



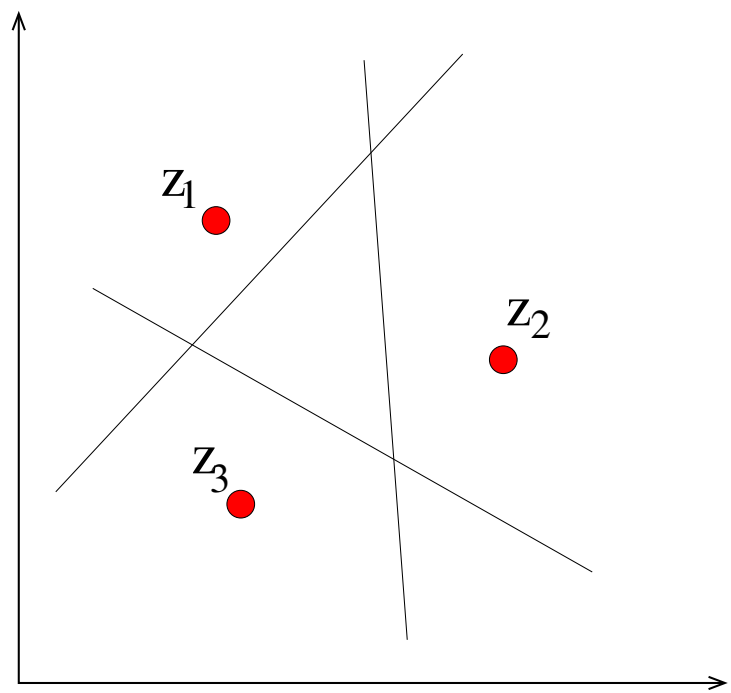
Learning f requires small training error *and* small complexity of the set $\{f_\alpha\}$.

VC-Dimension: Example

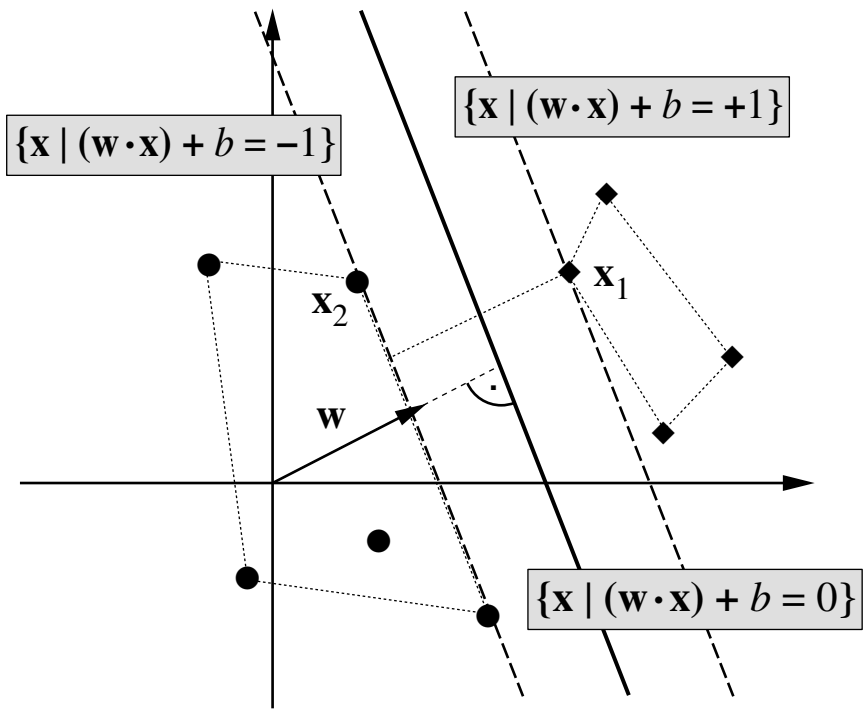
Half-spaces in $\mathbf{R}^2$:

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

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



linear hyperplane classifier



Note:

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

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

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

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

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

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

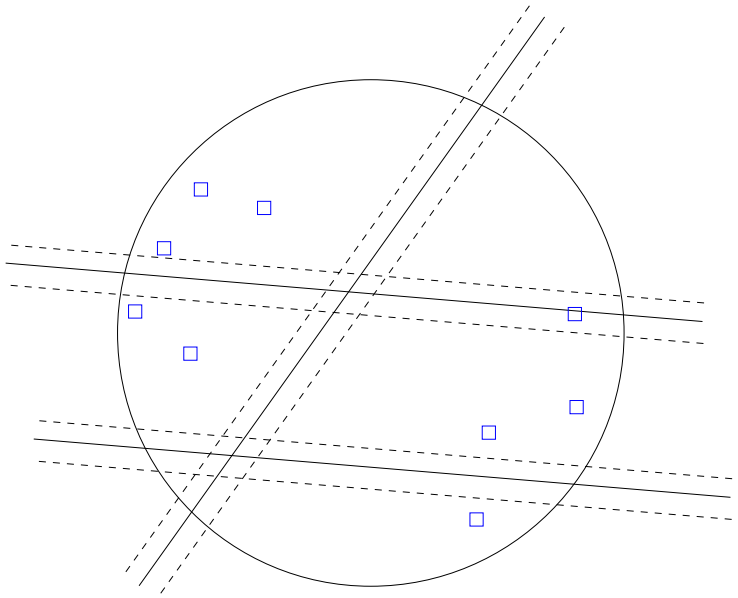
Learn $\tilde{f}$ to construct $f = \tilde{f} \circ \Phi$

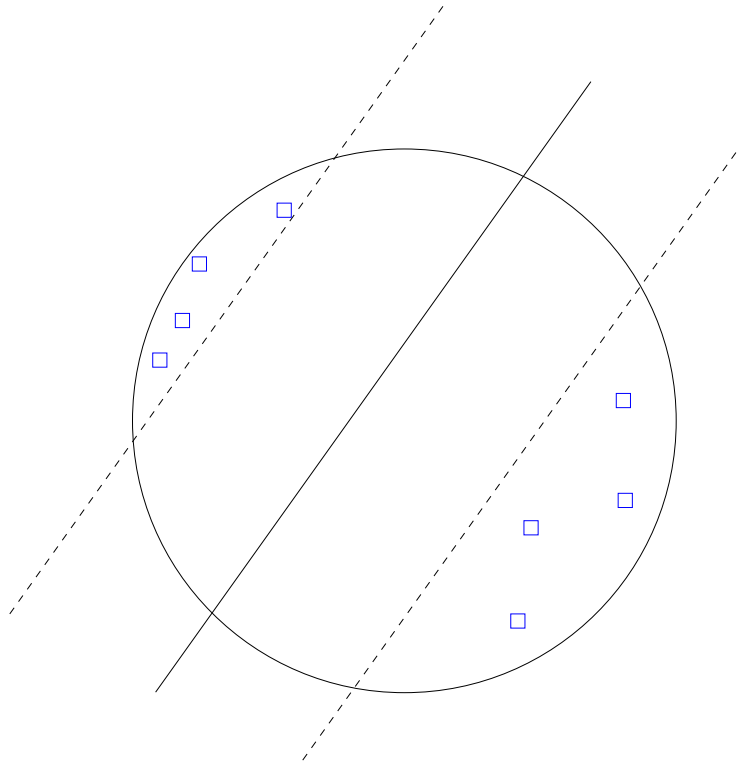
- classical statistics: **harder**, as the data are high-dimensional
- SV-Learning: (in some cases) **simpler**:

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 and "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**:

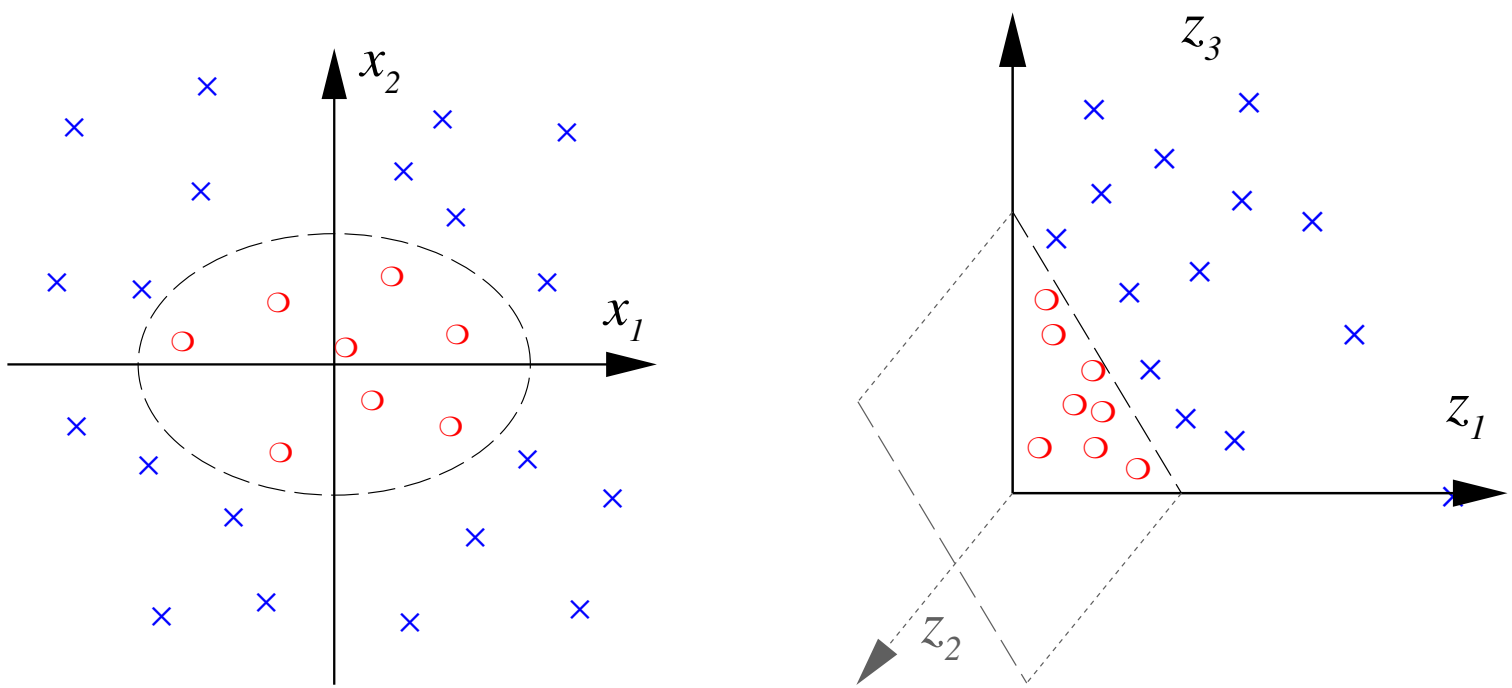
If Φ is chosen such that $\{\tilde{f}\}$ allows small training error *and* has low complexity, then we can guarantee good generalization.

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

Nonlinear Algorithms in Feature Spaces

Example: all second order monomials

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



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:

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

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 $\mathcal{F}$

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

(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

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

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

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

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

Substitute both into L to get the *dual problem*

Dual problem

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

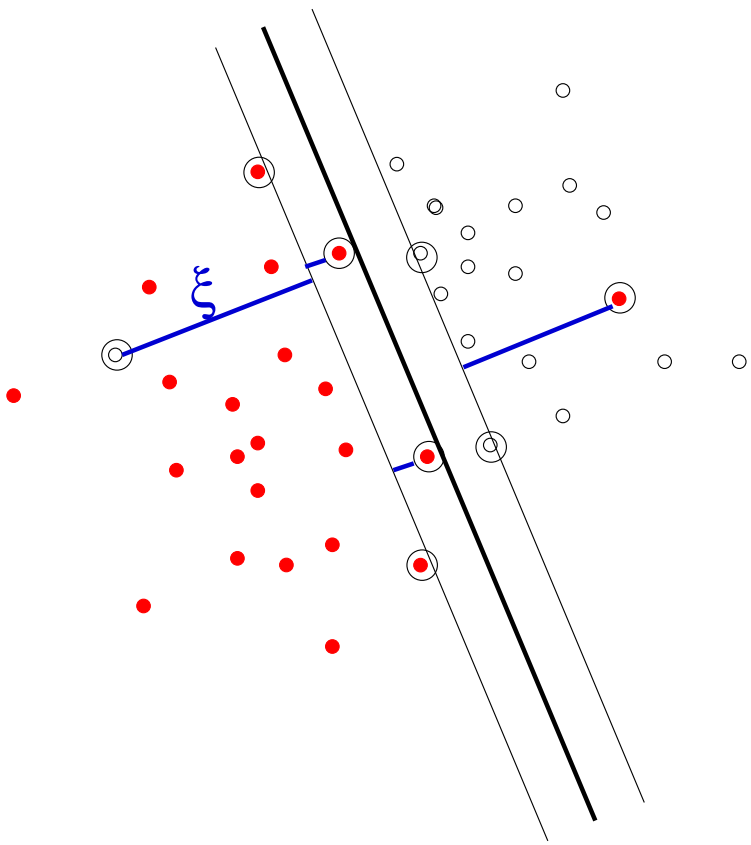
$$\text{subject to} \qquad C \geq \alpha_i \geq 0, \quad i = 1, \ldots, N, \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i y_i = 0.$$

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

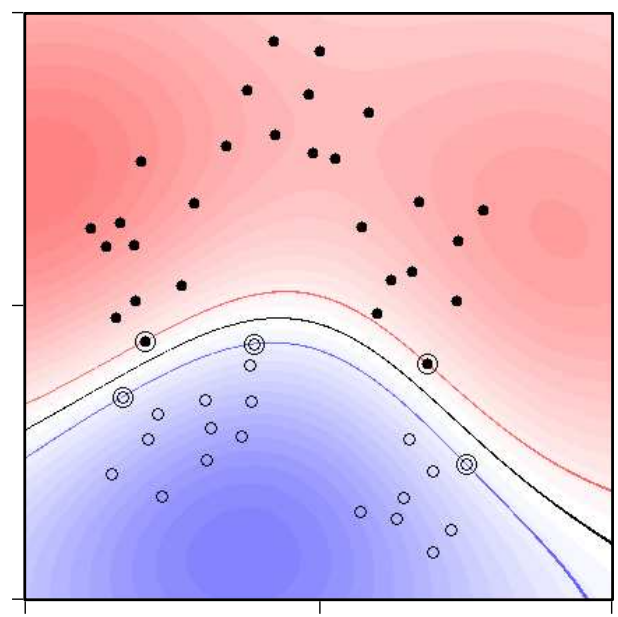
$$y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] > 1 \qquad \Longrightarrow \alpha_i = 0 \longrightarrow \quad \mathbf{x}_i \text{ irrelevant or}$$

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

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 $\mathcal{F}$: $y = \operatorname{sgn}(\mathbf{w} \cdot \Phi(x) + b)$
- 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

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

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

SRM (VC theory)

feature spaces & RKHS

kernel trick

mathematical programs

SV classification & regression

feature extraction → **kernelize it!**

kernel choice

Boosting

Large margins

Bounds, sparsity

APPLICATIONS

OCR, Bioinformatics **(DNA, drug discovery)**

categorization, biomedical data (BCI, EEG),

benchmarks, time-series analysis, power,

quarks, object recognition, textmining

An abstract view: mathematical programs I

- $y = sgn(\mathbf{w}^\top \mathbf{x} + b)$; find $\mathbf{w}$ by solving (cf. Mika et al 2001)

$$\text{SVM:} \quad \min_{w,b,\xi} \; \frac{1}{2}\|w\|_2^2 \; + \; \frac{C}{K}\|\xi\|_2^2$$

$$\text{subject to} \qquad y_k(w^\top x_k + b) \geq 1 - \xi_k \quad \text{for } k = 1, \ldots, K$$

$$\text{LPM:} \quad \min_{w,b,\xi} \; \frac{1}{2}\|w\|_1 \; + \; \frac{C}{K}\|\xi\|_1$$

$$\text{subject to} \qquad y_k(w^\top x_k + b) \geq 1 - \xi_k \quad \text{for } k = 1, \ldots, K$$

$$\text{RFD:} \quad \min_{w,b,\xi} \; \frac{1}{2}\|w\|_2^2 \; + \; \frac{C}{K}\|\xi\|_2^2$$

$$\text{subject to} \qquad y_k(w^\top x_k + b) = 1 - \xi_k \quad \text{for } k = 1, \ldots, K$$

- regularization: limit influence of single patterns: mistrust data!
- use more robust loss functions, e.g. ℓ_1-norm or Huber-loss

Mathematical programs II

- $y = sgn(\mathbf{w}^\top \Phi(\mathbf{x}) + b)$; find $\mathbf{w}$ by solving

$$\text{SVM:} \quad \min_{w,b,\xi} \; \frac{1}{2}\|w\|_2^2 \; + \; \frac{C}{K}\|\xi\|_2^2$$

$$\text{subject to} \qquad y_k(w^\top \Phi(x_k) + b) \geq 1 - \xi_k \quad \text{for } k = 1, \ldots, K$$

$$\text{LPM:} \quad \min_{w,b,\xi} \; \frac{1}{2}\|w\|_1 \; + \; \frac{C}{K}\|\xi\|_1$$

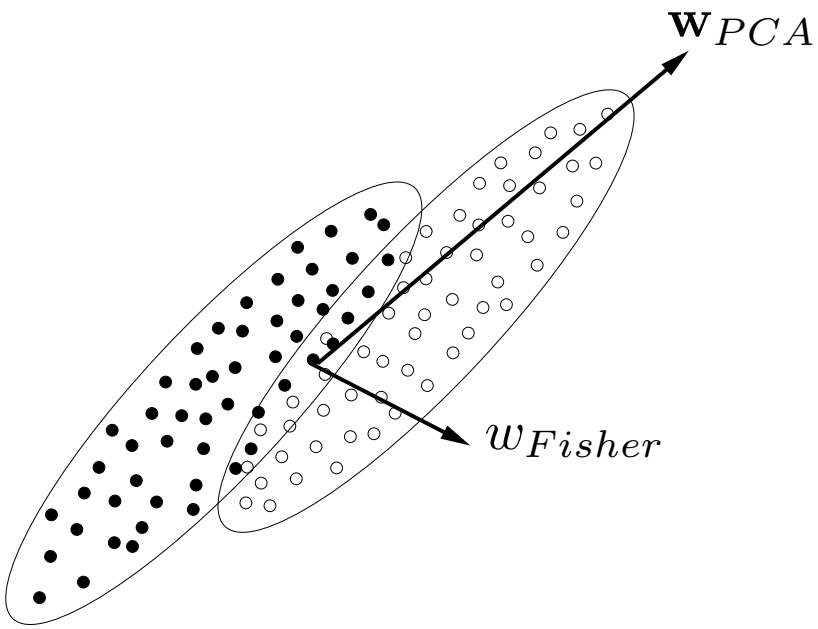
$$\text{subject to} \qquad y_k(w^\top \Phi(x_k) + b) \geq 1 - \xi_k \quad \text{for } k = 1, \ldots, K$$

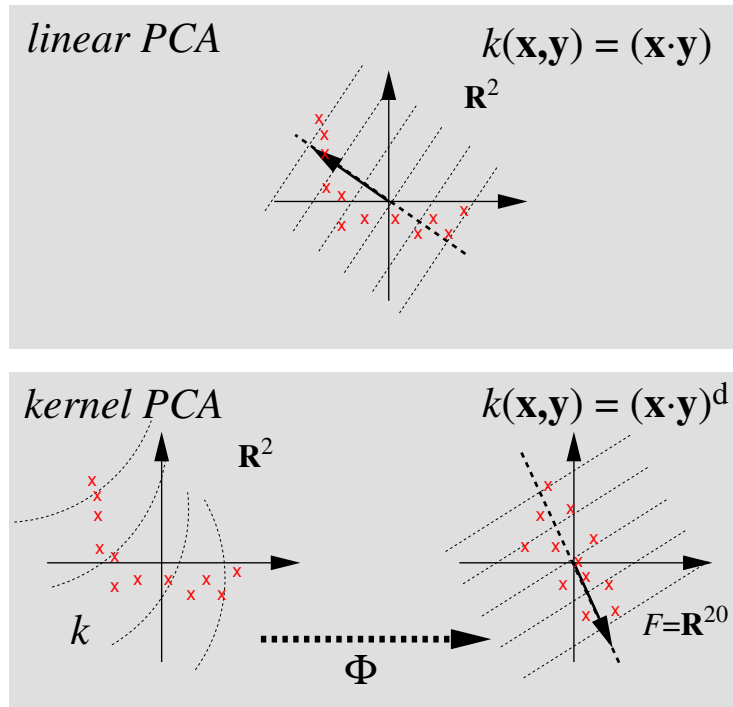
$$\text{RFD:} \quad \min_{w,b,\xi} \; \frac{1}{2}\|w\|_2^2 \; + \; \frac{C}{K}\|\xi\|_2^2$$

$$\text{subject to} \qquad y_k(w^\top \Phi(x_k) + b) = 1 - \xi_k \quad \text{for } k = 1, \ldots, K$$

- LPM: implicit feature selection, sparsity
- remarks: average vs. maximal margin, Boosting, model selection

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, ...)

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 $\lambda \neq 0$, $\mathbf{V} \in \operatorname{span}\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_N)\}$, 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 $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^\top$.

Solve

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

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

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

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 $(1_N)_{ij} := 1/N$:

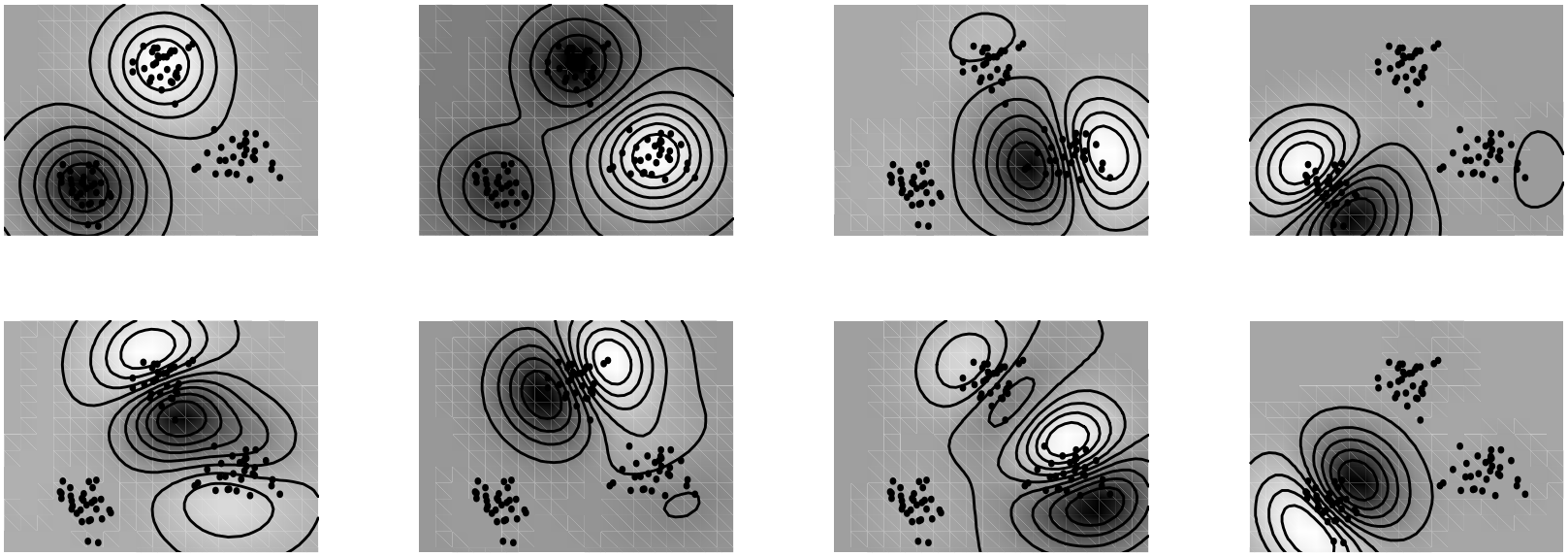
$$\tilde{K}_{ij} = K - 1_N K - K 1_N + 1_N K 1_N.$$

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

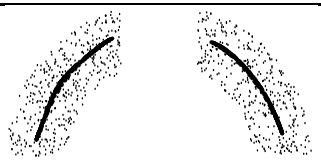
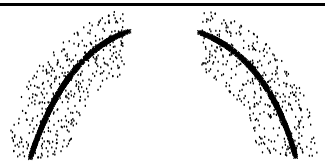
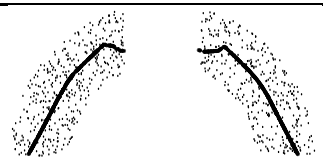
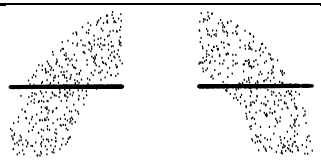
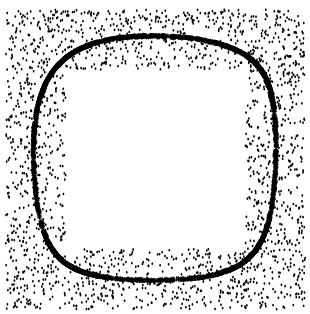
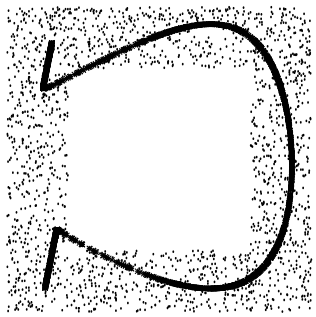
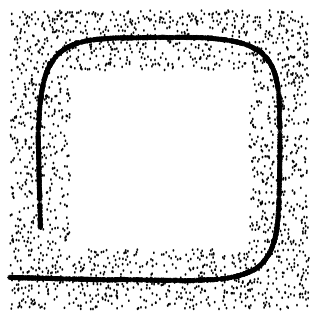
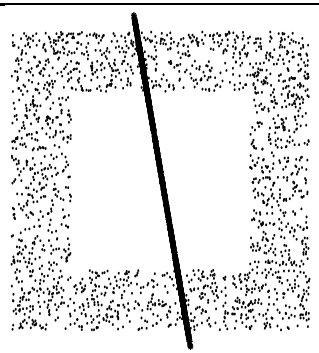
Similar for feature extraction.

Example: RBF Kernel, 8 Principal Components

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

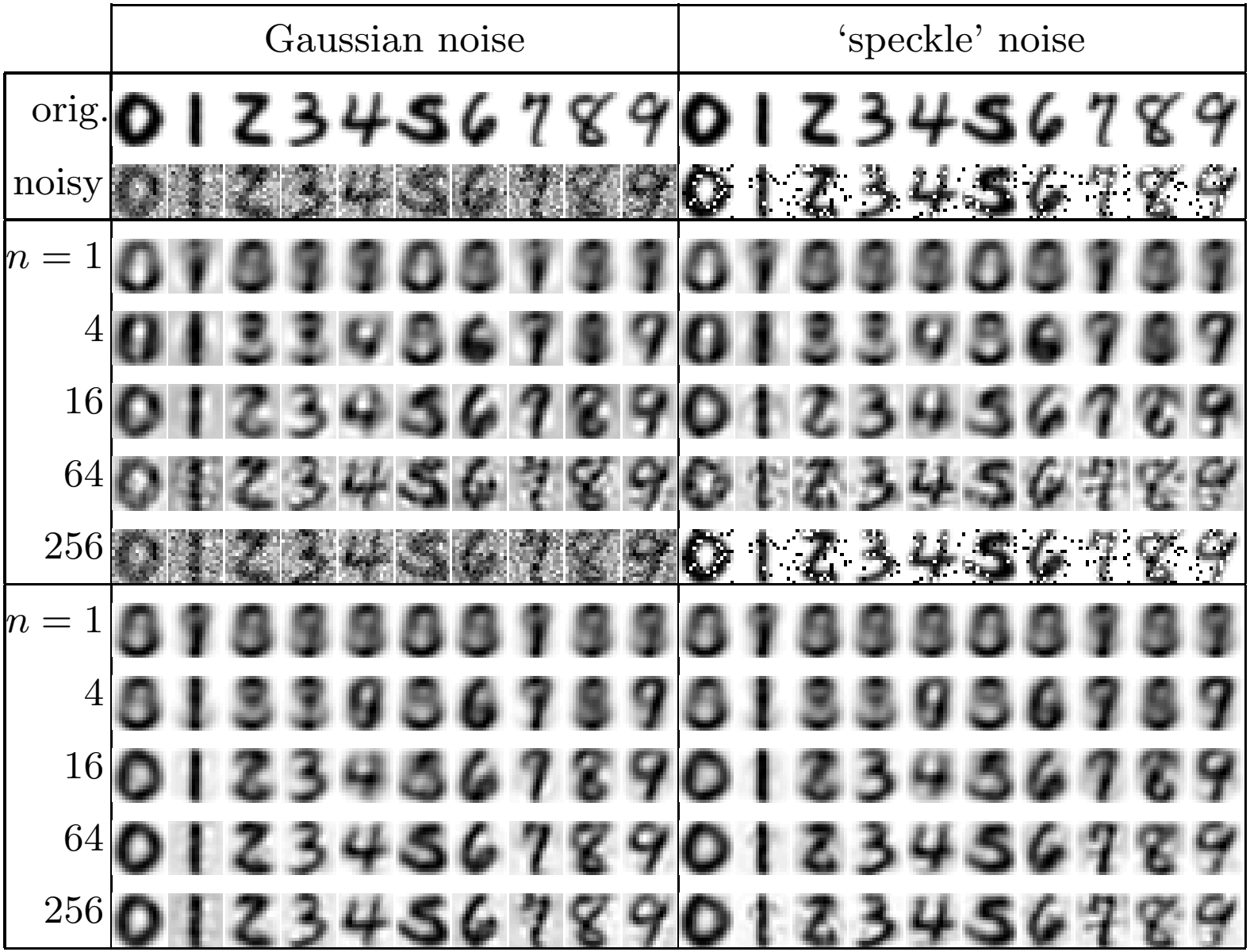


Denoising

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

kernels revisited

- 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

Kernels from Probability Models (J & H 98)

- idea
  - make use of probabilistic modeling for discriminative training (e.g. a HMM model)
  - find natural comparison between objects (of **various** types) that is induced by a generative probability model $P(X|\Theta)$
- Thus: derive a SV-Kernel that reflects this probabilistic model!
- parametric class of models: $P(X|\theta), \theta \in \Theta$ forms a Riemannian manifold M_Θ with a metric: the Fisher Information G

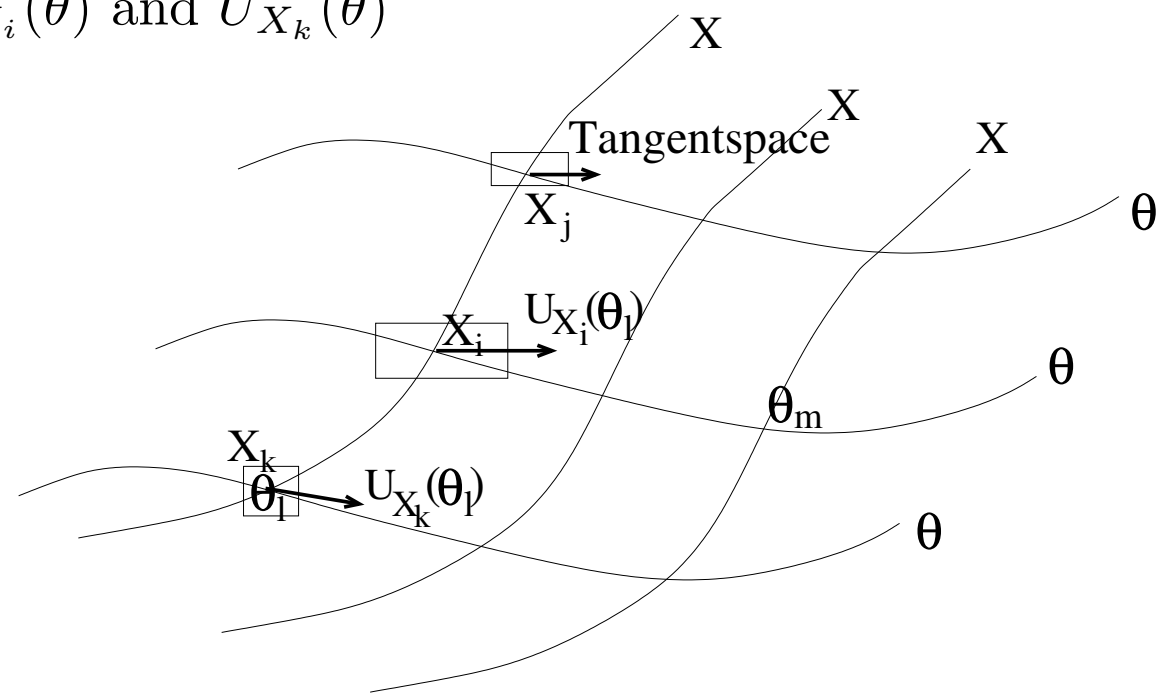
  $$G = E_X\{U_X U_X^\top\}, \quad \text{where} \quad U_X = \nabla_\theta \log P(X|\theta) \quad \text{[Fisher Score]}$$

- "natural" mapping to feature space $\Phi_X = G^{-1} U_X$, so

  $$K(X_i, X_j) \sim \Phi_{X_i} G \Phi_{X_j} = U_{X_i} G^{-1} U_{X_j} \quad \text{for fixed parameter } \theta$$

A geometrical picture

idea: compute similarity between X_i and X_k in terms of probabilistic model with fixed parameters θ_l between $P(X_i|\theta_l)$ and $P(X_k|\theta_l)$ by appropriately computing scalar products of tangent vectors $U_{X_i}(\theta)$ and $U_{X_k}(\theta)$



Protein superfamily classification

- 4541 sequences are hierarchically labeled into 7 classes, 558 folds, 845 superfamilies and 1343 families according to the SCOP scheme
- task: classify top category classes.
- number of sequences: 791, 1277, 1015, 915, 84, 76, 383
- classify as a set of 2-class problems
- train HMMs to obtain $p(\mathbf{x}, \Theta)$ for each class and
- construct Fisher Kernel: $K(X_i, X_j) \sim U_{X_i} G^{-1} U_{X_j}$
- **note**: FK can be improved by TOP kernel (cf. Tsuda et al. 2001 & 2002)

TOP Kernel

- The new feature extractor is described as

$$\boldsymbol{f}_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) := (v(\mathbf{x}, \hat{\boldsymbol{\theta}}), \partial_{\theta_1} v(\mathbf{x}, \hat{\boldsymbol{\theta}}), \ldots, \partial_{\theta_p} v(\mathbf{x}, \hat{\boldsymbol{\theta}}))^\top$$

  where

$$\begin{aligned} v(\mathbf{x}, \boldsymbol{\theta}) &= F^{-1}(P(y = +1|\mathbf{x}, \boldsymbol{\theta})) \\ &= \log(P(y = +1|\mathbf{x}, \boldsymbol{\theta})) - \log(P(y = -1|\mathbf{x}, \boldsymbol{\theta})), \end{aligned}$$

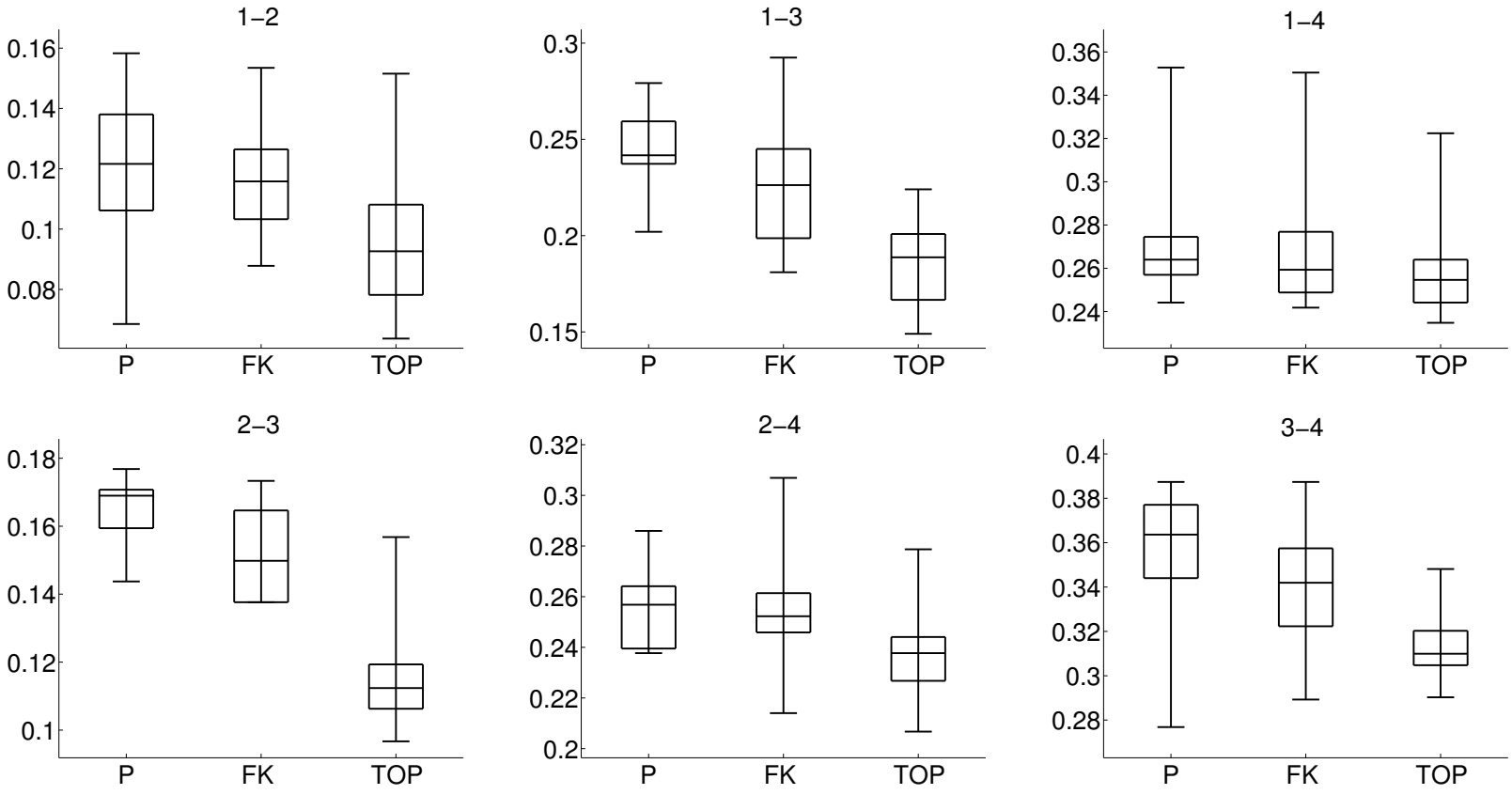
- We refer the inner product in the feature space as the *TOP kernel*

$$K_T(\mathbf{x}, \mathbf{x}') = \boldsymbol{f}_{\hat{\boldsymbol{\theta}}}(\mathbf{x})^\top \boldsymbol{f}_{\hat{\boldsymbol{\theta}}}(\mathbf{x}').$$

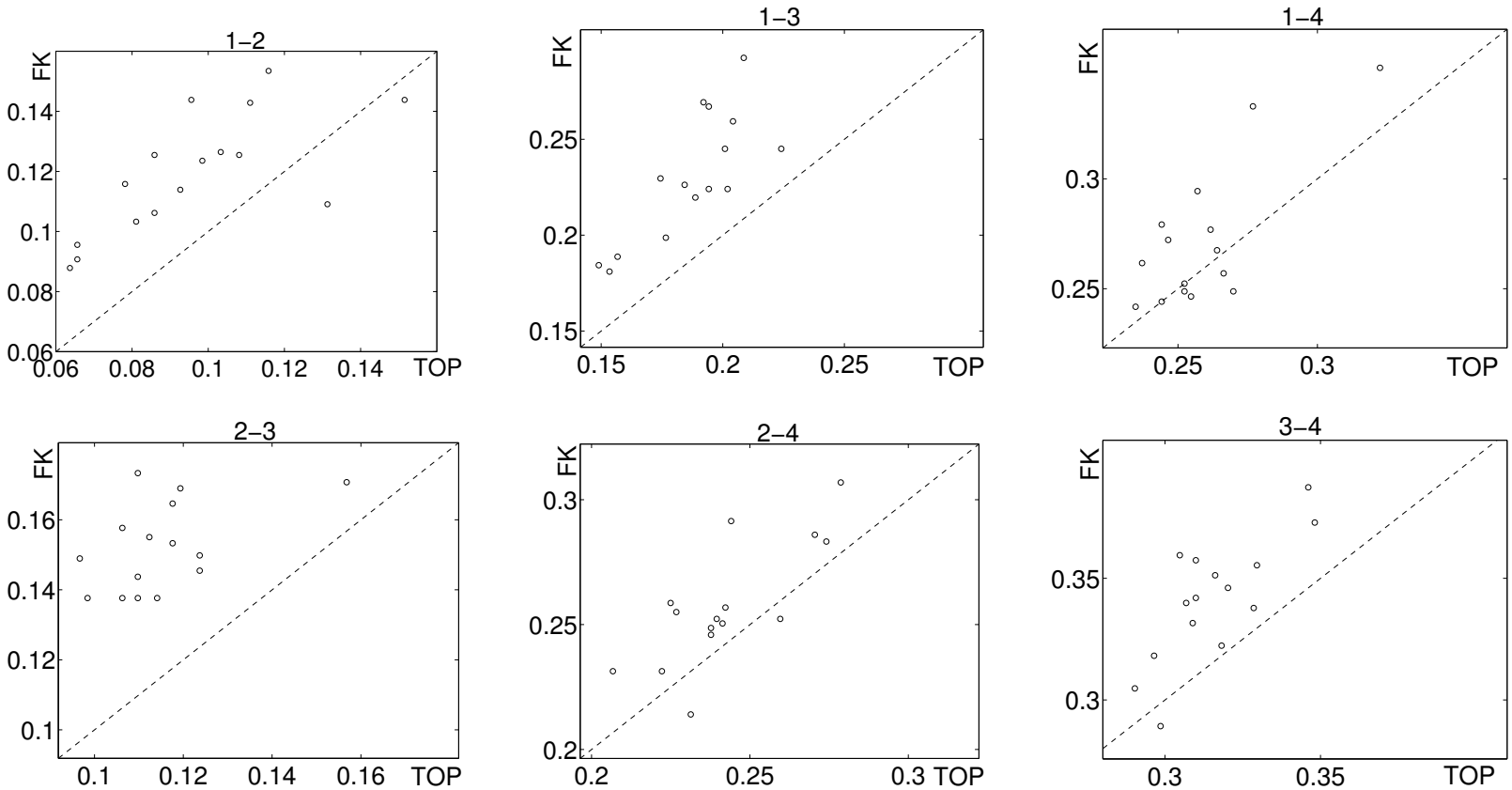
- Plug-in Estimate performs classification by means of the posterior probabilities derived from class conditional likelihoods

$$\hat{y} = \mathrm{sgn}(P(y = +1|\mathbf{x}, \hat{\boldsymbol{\theta}}) - 0.5)$$

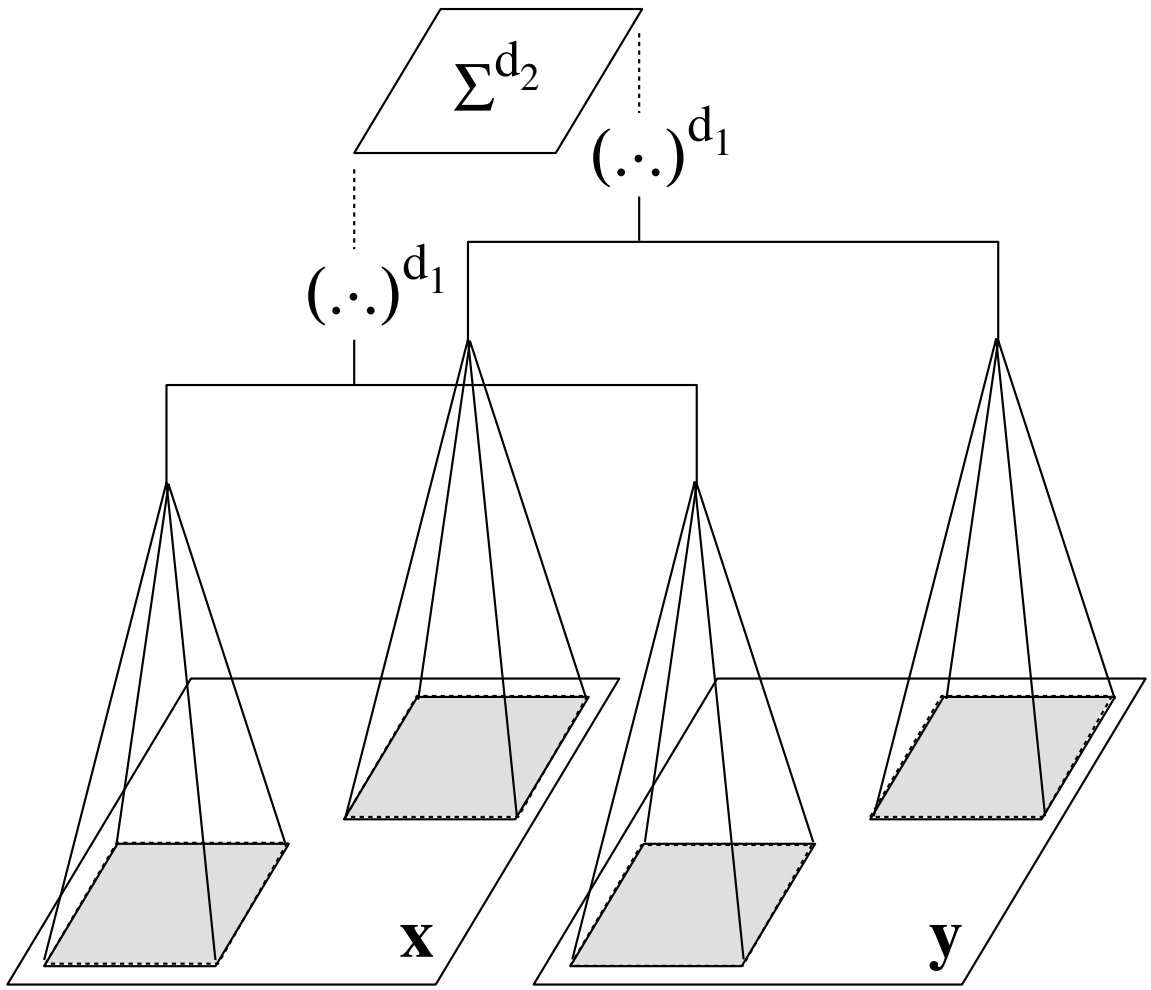
Protein superfamilies



Comparison Plots in Protein Classification



Engineering a kernel for RNA analysis



Application: BCI

Other applications of SVM & Boosting

- automatic music playlist generation (Platt et al. 2002, MSR)
- pedestrian and face recognition (Osuna et al 1998, MIT)
- object recognition (Schölkopf 1998, FhG)
- power monitoring (Rätsch et al. 2000, CRIEPI & FhG)
- management fraud detection (Rätsch & Müller 2001)
- bioinformatics (Jaakola, Haussler et al. 2000, 2001, MIT & UCSC)
- e-mail spam filter (Platt et al. 1999, MSR)
- charm quark detection (Vanerem et al. 1999, CERN & FhG)
- text mining (Joachims 1997, U DO & CMU, Schapire 2000, AT&T)
- nonlinear ICA (Harmeling et al 2002, FhG)
- outlier/novelty detection in high dimensions (Schölkopf et al 2001, MSR, Tax 2001, Delft)

Conclusions & Outlook

- kernel based learning
  - kernelize linear algorithms (Schölkopf, Smola and Müller 1996): Kernel PCA, Kernel Fisher, Clustering, ICA ...
  - mathematical programming machines (LP...)
  - incorporate prior knowledge to get more robust learning machines, e.g. engineer kernels, use invariances
  - unclear: what is better engineered kernel or kernel from probabilistic model
- applications
  - OCR, time-series, questionaires, car industry data
  - DNA/protein analysis, drug discovery, BCI, novelty

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

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