Learning in Kernel Feature Spaces

Klaus-Robert Müller

Technische Universität Berlin, Germany
and Fraunhofer Institute FIRST, Berlin, Germany

Joint work with

Mikio L. Braun[1], Joachim Buhmann[2]

[1] Technische Universität Berlin, Germany
[2] ETH Zürich, Switzerland

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Basic Ideas of Statistical Learning Theory I

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

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

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

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

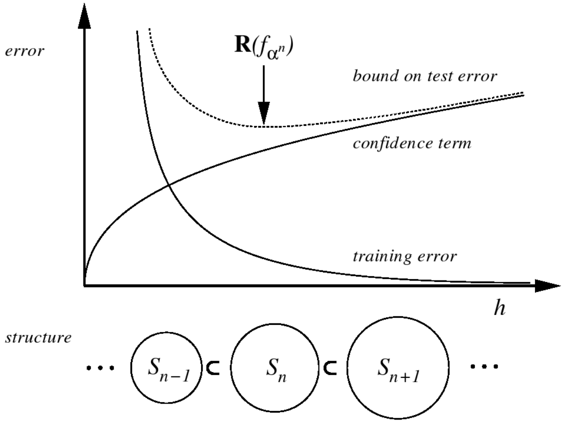
Basic Ideas of Statistical Learning Theory II

- Law of large numbers: $R_{emp}[f] \rightarrow R[f]$ as $N \rightarrow \infty$.
  "consistency" of ERM: for $N \rightarrow \infty$, ERM should lead to the same result as RM?
- No: *uniform* convergence needed (Vapnik) → VC theory.
  Theorem (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


error
$\mathbf{R}(f_{\alpha^n})$
bound on test error
confidence term
training error
h
structure
$\cdots S_{n-1} \subset S_n \subset S_{n+1} \cdots$

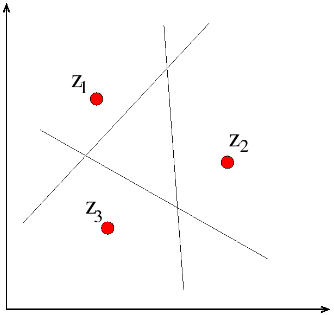
Learning f requires small training error *and* small complexity of the set $\{f_\alpha\}$.

VC-Dimension: Example

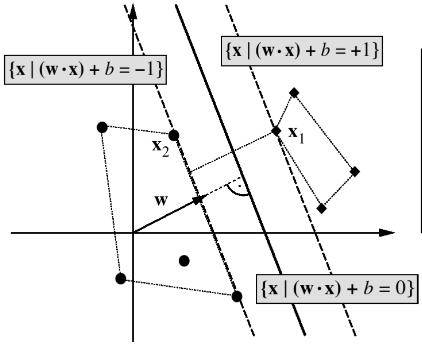
Half-spaces in $\mathbb{R}^2$:

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

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



Linear Hyperplane Classifier


$\{\mathbf{x} \mid (\mathbf{w}\cdot\mathbf{x}) + b = -1\}$
$\{\mathbf{x} \mid (\mathbf{w}\cdot\mathbf{x}) + b = +1\}$
$\{\mathbf{x} \mid (\mathbf{w}\cdot\mathbf{x}) + b = 0\}$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{w}$

Note:

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

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

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

- hyperplane $y = \operatorname{sgn}(\mathbf{w}\cdot\mathbf{x} + b)$ 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

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

to get data $(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_N), y_N) \in F \times \mathbb{R}^m$ 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**:

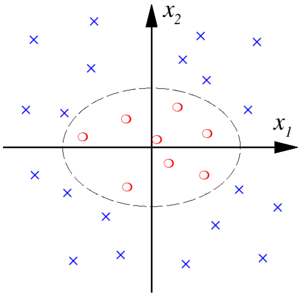
If Φ is chosen such that $\{\tilde{f}\}$ allows small training error *and* has low complexity, then we can guarantee good generalization.

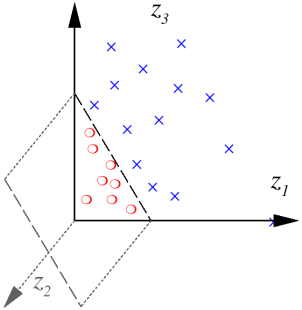
The *complexity* matters, not the *dimensionality* of the space.

Nonlinear Algorithms in Feature Spaces

Example: all second order monomials

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





Kernel "Trick": An Example

(cf. Boser, Guyon & Vapnik 1992)

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

- Scalar product in (**high dimensional**) feaure space can be computed in $\mathbb{R}^2$!
- 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

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

with $\lambda_i > 0$, and $\dim(F) \in \mathbb{N}$ or $\dim(F) = \infty$. 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:

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

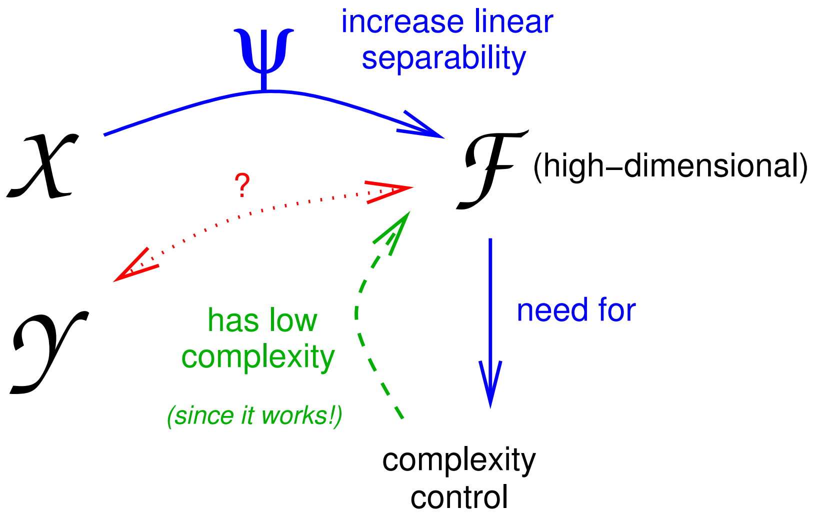
Note: kernels correspond to regularization operators (a la Tichonov) with regularization properties that can be conveniently expressed in Fourier space, e.g. Gaussian kernel corresponds to general smoothness assumption (Smola et al 98)!

A Preliminary Summary

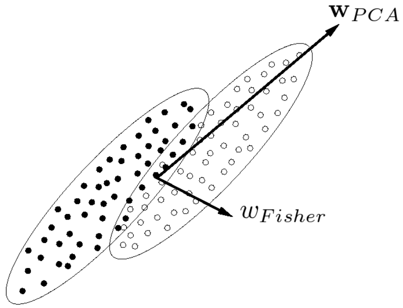
- Statistical learning theory tells us: we need to restrict the complexity of our hypthesis class and trade-off error vs. complexity.
- For large margin hyperplanes, VC-dimension is independent of dimensionality of space
- Kernels can be used to preprocess data to increase discriminative power.

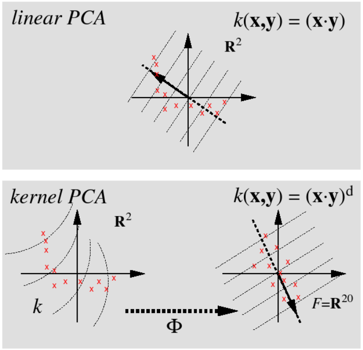
Still, it is not fully clear why any of this works: How do kernels find useful non-linear preprocessings, which also realize a large margin?

Learning in Kernel Spaces



Kernelizing Linear Algorithms—PCA




linear PCA
$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})$
$\mathbf{R}^2$
kernel PCA
$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^d$
$\mathbf{R}^2$
k
Φ
$F=\mathbf{R}^{20}$

(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

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

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 $\lambda \neq 0$, $\mathbf{V} \in \text{span}\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_N)\}$, 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

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

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

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

Feature Extraction

Compute projections on the Eigenvectors

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

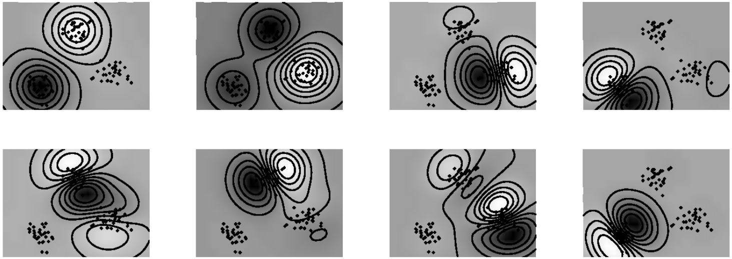
in F:

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

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

Example: RBF Kernel, 8 Principal Components

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



Empirical Kernel Map

Kernel PCA components ("features") allow us to construct a feature mapping and look at the data points in feature map:

Let $\mathbf{K} = k(\mathbf{x}_i, \mathbf{x}_j)$, and let $\mathbf{K} = \mathbf{U}\mathbf{L}\mathbf{U}^\top$ be the eigendecomposition of $\mathbf{K}$.

Then,

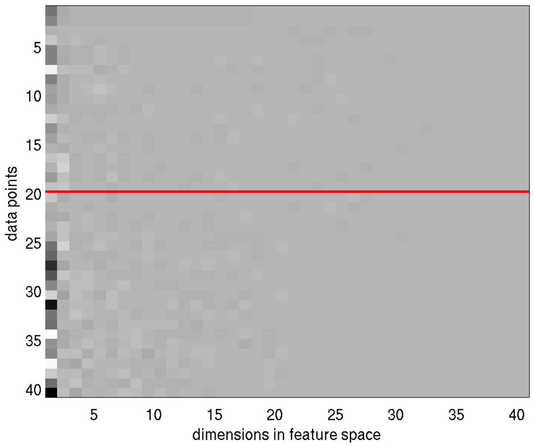
$$\mathbf{F} = \mathbf{U}\mathbf{L}^{1/2}$$

is a matrix such that

$$\mathbf{F}\mathbf{F}^\top = \mathbf{U}\mathbf{L}^{1/2}\mathbf{L}^{1/2}\mathbf{U}^\top = \mathbf{K}.$$

This means that the rows of $\mathbf{F}$ are the transformed input points such that their scalar products are $k(\mathbf{x}_i, \mathbf{x}_j)$.

Example: ZIP Data Set



Columns are dimensions in feature space, rows are data points.

Note that variance of data becomes smaller and smaller.

Variances in Feature Space

Principal values (variances) are given by the eigenvalues of the kernel matrix.

Both sample and population eigenvalues typically decay quickly!

Theorem Bounds on the eigenvalues[1]

Individual eigenvalues:

$$|l_i - \lambda_i| \leq \lambda_i C(r,N) + E(r,N)$$

with $C(r,N) \to 0$ for $N \to \infty$, $E(r,N) \to 0$ for $r \to \infty$.
Tail sums of eigenvalues:

$$\left| \sum_{i=d}^{n} l_i - \sum_{i=d}^{\infty} \lambda_i \right| \leq C' \sqrt{\sum_{i=d}^{\infty} \lambda_i + E'}$$

[1] Blanchard et al., *Statistical Properties of Kernel Principal Component Analysis*, Machine Learning, 2006
Braun, *Accurate Bounds for the Eigenvalues of the Kernel Matrix*, JMLR, 2006

Kernel PCA and the Outputs

- In a supervised setting, the goal is to predict outputs y_i (class labels, real numbers) from inputs $\mathbf{x}_i$.
- Contributions of kernel PCA components can be computed by

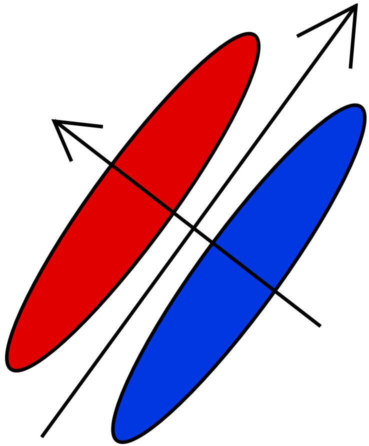
$$s_i = \mathbf{u}_i^\top \mathbf{y},$$

  with $\mathbf{u}_i$ eigenvector of kernel matrix $\mathbf{K}$, $\mathbf{y} = (y_1, \ldots, y_N)$ vector of outputs.
- Projection of outputs to first m kernel PCA components given by

$$\Pi_m \mathbf{y} = \sum_{i=1}^{m} \mathbf{u}_i \mathbf{u}_i^\top \mathbf{y}.$$

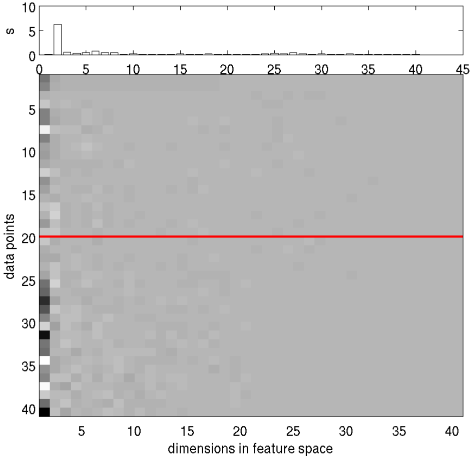
Ideally, $|s_i|$ is large only for a few directions.

Class Information and Large PCA Directions



Large PCA directions need not be informative!

Location of the Label Information

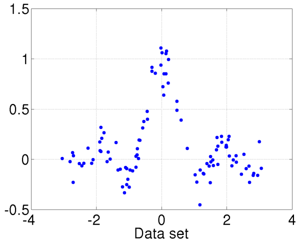


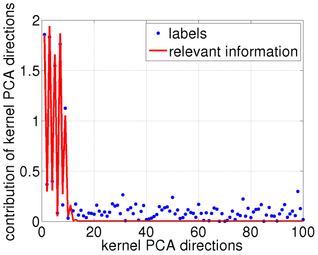
Information about class membership concentrated in direction 2, almost absent from dimensions with small variance!

Theoretical Result

Result

If we assume that the learning problem can be represented by the kernel asymptotically,
then the relevant information about the Y is contained in the leading kernel PCA directions up to a small error.





Overview

1. Define “relevant information”.
2. Consider asymptotic setting, introduce assumption, derive result for asymptotic setting.
3. Derive bound for contribution of kernel PCA direction for finite sample setting.
4. Consider noise in the labels.

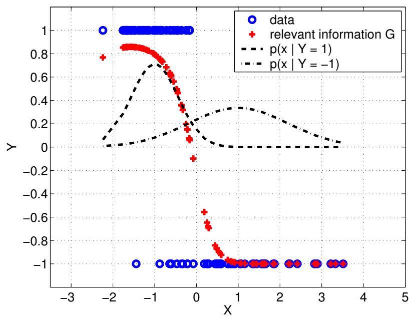
Relevant Information

Define "relevant information" by separating the noise from the outputs:

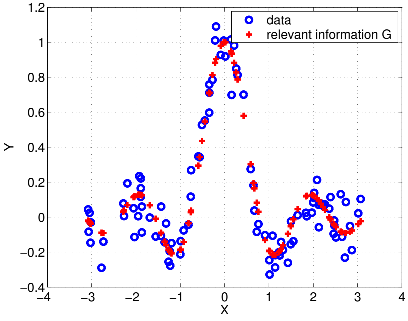
$$Y_i = g(X_i) + \varepsilon_i, \qquad \text{"outputs" = "smooth part" + "noise"}$$

$$g(x) = E(Y|X = x), \qquad \text{the (population) relevant information}$$

$$G = (g(X_1), \ldots, g(X_N)). \qquad \text{the (sample) relevant information}$$



classification



regression

The Finite Sample and the Asymptotic Setting

The question reduces to approximation of integral operators by Monte carlo integration:

Finite sample setting:

- The kernel matrix **K** defines a linear operator via

$$[\mathbf{K}v]_i = \sum_{j=1}^{N} k(X_i, X_j) v_j, \qquad \text{also for } v_j = f(X_j).$$

- This operator has eigenvectors u_i, which are also the kernel PCA components.
- The contribution of the ith component u_i to the relevant information in the labels G is given by

$$s_i = u_i^\top G.$$

The Finite Sample and the Asymptotic Setting

For $N \to \infty$, these quantites converge to their asymptotic counterparts:

The asymptotic setting:

- The kernel matrix (properly scaled) converges to an integral operator:

$$\mathbf{K}f(a) = \frac{1}{N}\sum_{j=1}^{N} k(a, X_j)f(X_j) \to T_k f(a) = \int_{\mathcal{X}} k(a,b)f(b)P_X(dt),$$

  where P_X is the probability measure generating the X_j.
- The eigenvectors converge to the eigenfunctions ψ_i of T_k.
- the contributions s_i converge to the scalar products with $g(x) = E(Y|X = x)$:

$$s_i = \frac{1}{\sqrt{N}}|u_i{}^\top G| \to |\langle\psi_i, g\rangle| = \int_{\mathcal{X}} \psi_i(x)g(x)P_X(dx).$$

The Finite Sample and the Asymptotic Setting

finite sample setting		asymptotic setting
$\mathbf{K}f(a) = \frac{1}{N}\sum_{j=1}^{N} k(a, X_j)f(X_j)$ u_i eigenvector of $\mathbf{K}$ $s_i = u_i{}^\top G$	$\rightsquigarrow$	$T_k f(s) = \int_{\mathcal{X}} k(s,t)f(t)P(dt)$ ψ_i eigenfunction of T_k $\sigma_i = \langle \psi_i, g \rangle$

Discussion of the Asymptotic Setting

Under the following assumption, asymptotic coefficients σ_i decay at rate $O(\lambda_i)$:

Assumption

Kernel and data set match in the following sense:
g asymptotically representable by T_k, (exists h such that $g = T_k h$):

$$\rightsquigarrow g(x) = \sum_{i=1}^{\infty} \lambda_i \alpha_i \psi_i(x)$$

$\rightsquigarrow$ $$\boxed{\sigma_i = \lambda_i \alpha_i = O(\lambda_i).}$$

(Note: Constant unspecified, depends on fit between kernel and data set.)

Equivalence of Finite Sample and Asymptotic Setting

Theorem

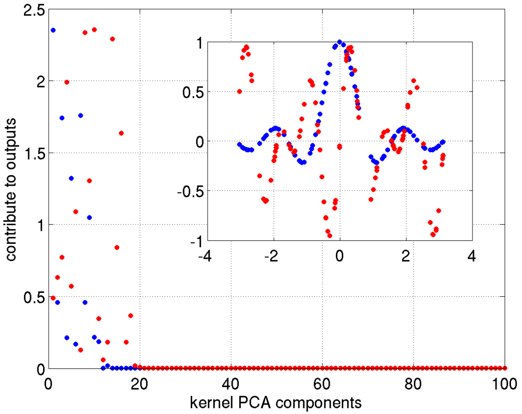
Let $g(x) = \sum_{i=1}^{\infty} \alpha_i \lambda_i \psi_i(x)$, $G = (g(X_1), \ldots, g(X_N))$. Then, with high probability,

$$\frac{1}{\sqrt{N}} |u_i^\top G| < 2 l_i a_r c_i (1 + O(rN^{-1/4})) + r a_r \Lambda_r O(1) + T_r + \sqrt{A T_r} O(N^{-1/4}) + r a_r \sqrt{\Lambda_r} O(N^{-1/2}),$$

where

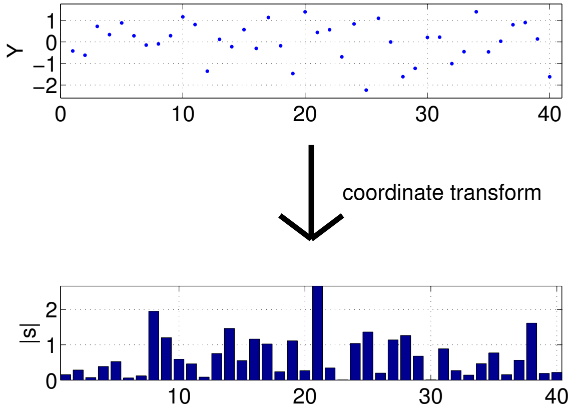
c_i:	measures size of the eigenvalue cluster around l_i
$a_r = \sum_{i=1}^{r} \lvert\alpha_i\rvert$:	measure for size of the first r components
Λ_r:	sum of all eigenvalues smaller than λ_r
A:	supremum norm of g
T_r:	error of projecting g to the space spanned by the first r eigenfunctions

An Example



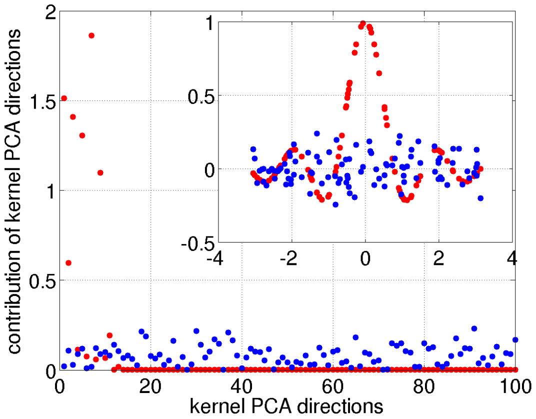
(rbf-kernel with $w = 1$, $\text{sinc}(x)$ function and $\cos(x)\sin(5x)$.)

The Location of Zero-Mean Noise



Since $\mathbf{U}^\top$ is a random rotation, noise stays the same.

Example

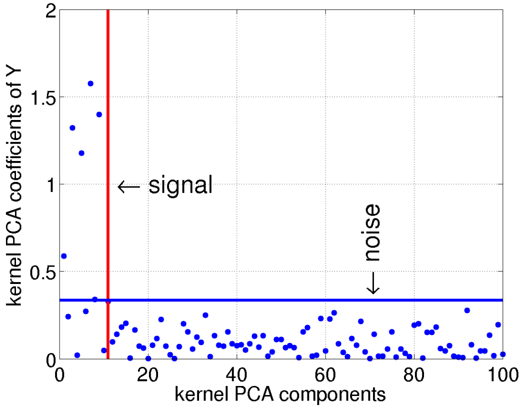

contribution of kernel PCA directions
kernel PCA directions
0
0.5
1
1.5
2
0
20
40
60
80
100
-0.5
0
0.5
1
-4
-2
0
2
4

Applications

- Estimating the dimensionality of the data set given a kernel.
- Denoising the labels, estimating the amount of noise in the labels.
- Model selection among kernels.

Estimating the Dimensionality

Fitting a Two-Component Model



Find cut-off dimension which separates the two parts.

Estimating the Dimensionality
Fitting a Two-Component Model

Assumption:

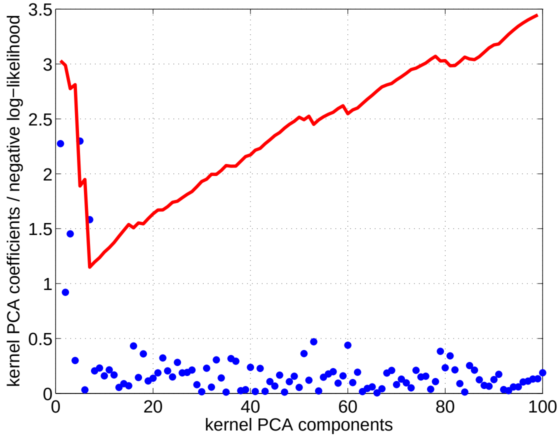
$$s_i \sim \begin{cases} \mathcal{N}(0, \sigma_1^2) & 1 \leq i \leq d \\ \mathcal{N}(0, \sigma_2^2) & d < i \leq n \end{cases}$$

The negative log-likelihood is proportional to

$$-\log \ell(d) \sim \frac{d}{n} \log \frac{1}{d} \sum_{i=1}^{d} s_i^2 + \frac{n-d}{n} \log \frac{1}{n-d} \sum_{i=d+1}^{n} s_i^2 .$$

$\rightsquigarrow$ choose d which minimizes $-\log \ell(d)$.

Estimating the Dimensionality
Fitting a Two-Component Model

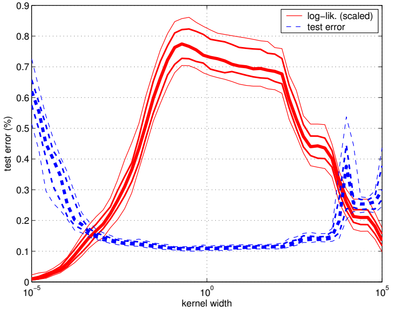


The resulting log-likelihoods.

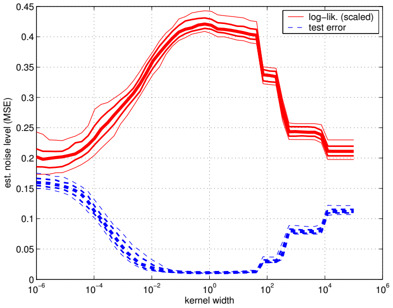
Model Selection

Idea: Use kernel which separates noise from data best.

$\rightsquigarrow$ choose kernel such that log-likelihood value at $\hat{d}$ is maximal.



classification
(banana)



regression
(noisy sinc)

Benchmark Data Sets

data set	dim	dim (cv)	est. error rate	kPCR	KRR	SVM
banana	24	26	8.8 ± 1.5	11.3 ± 0.7	10.6 ± 0.5	11.5 ± 0.7
breast-cancer	2	2	25.6 ± 2.1	27.0 ± 4.6	26.5 ± 4.7	26.0 ± 4.7
diabetis	9	9	21.5 ± 1.3	23.6 ± 1.8	23.2 ± 1.7	23.5 ± 1.7
flare-solar	10	10	32.9 ± 1.2	33.3 ± 1.8	34.1 ± 1.8	32.4 ± 1.8
german	12	12	22.9 ± 1.1	24.1 ± 2.1	23.5 ± 2.2	23.6 ± 2.1
heart	4	5	15.8 ± 2.5	16.7 ± 3.8	16.6 ± 3.5	16.0 ± 3.3
image	272	368	1.7 ± 1.0	4.2 ± 0.9	2.8 ± 0.5	3.0 ± 0.6
ringnorm	36	37	1.9 ± 0.7	4.4 ± 1.2	4.7 ± 0.8	1.7 ± 0.1
splice	92	89	9.2 ± 1.3	13.8 ± 0.9	11.0 ± 0.6	10.9 ± 0.6
thyroid	17	18	2.0 ± 1.0	5.1 ± 2.1	4.3 ± 2.3	4.8 ± 2.2
titanic	4	6	20.8 ± 3.8	22.9 ± 1.6	22.5 ± 1.0	22.4 ± 1.0
twonorm	2	2	2.3 ± 0.7	2.4 ± 0.1	2.8 ± 0.2	3.0 ± 0.2
waveform	14	23	8.4 ± 1.5	10.8 ± 0.9	9.7 ± 0.4	9.9 ± 0.4

kPCR: (kernel) least-squares on the denoised data
KRR: kernel ridge regression
SVM: support vector machines

Benchmark Data Sets: Categorizing Data Sets

	low noise	**high noise**
low dimensional	banana, thyroid, waveform	breast-cancer, diabetis flare-solar, german heart, titanic
high dimensional	image, ringnorm	splice

- Splice data set seems most promising for more model selection.
- On "high noise, low dimensional" data sets, data seems to be intrinsically very noisy.

Application: Kernel Design for Splice Site Detection

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
TTGTTTTAATATTCAATTTTTTACAGTAAGTTGCCAATTCAATGTTCCAC
CTGTATTCAATCAATATAATTTTCAGAAACCACACATCACAATCATTGAA
TACCTAATTATGAAATTAAAATTCAGTGTGCTGATGGAAACGGAGAAGTC
```

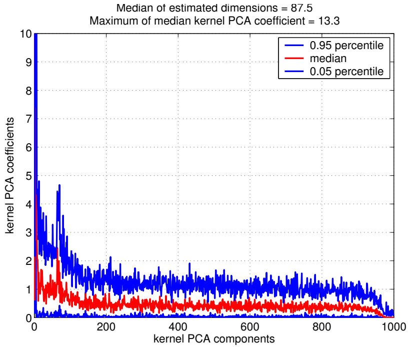
Genes are not encoded in one piece on the DNA, but in multiple parts.

Splice sites indicate where a coding region ends.

First, the whole protein sequence is built from the DNA, then special enzymes "cut out" the non-coding regions based on the splice cites.

Naive Encoding

Aminoacid	Encoded as
A	0
C	1
G	2
T	3



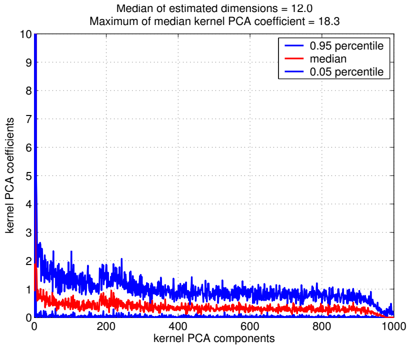
Dimensionality 87, test error $12.9 \pm 0.9\%$.

Using an rbf kernel, over 100 resamples of the data.

Main problem: A, C appear more similar than A, T.

A Better Encoding

Aminoacid	Encoded as
A	(0, 0, 0, 1)
C	(0, 0, 1, 0)
G	(0, 1, 0, 0)
T	(1, 0, 0, 0)



Dimensionality 11, test error $7.6 \pm 0.7\%$.

All aminoacids are comparably far from one another. But only fixed positions are comapred.

A Domain Specific Kernel: Weighted Degree Kernel

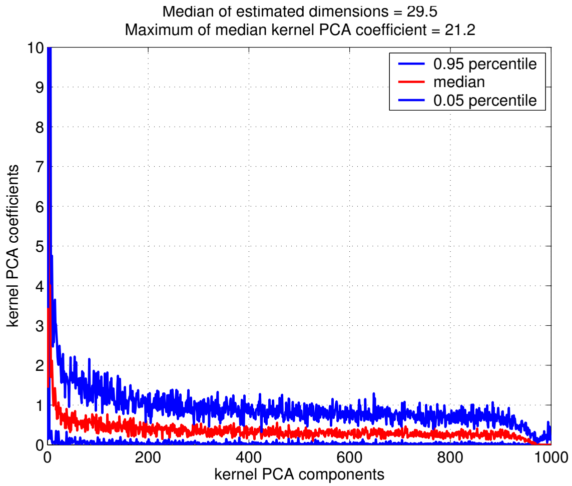
Weighted degree kernel is defined as

$$k(x, x') = \sum_{j=1}^{d} w_i \sum_{i=1}^{N-d} 1_{\{u_{j,i}(x)=u_{j,i}(x')\}}$$

with:

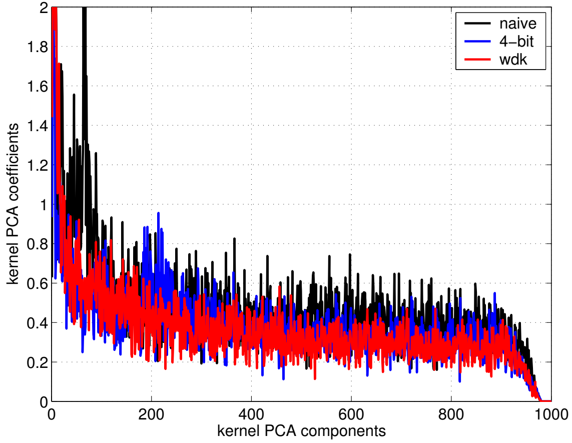
$$u_{j,i}(x) = x_i x_{i+1} \dots x_{i+j-1} \qquad \text{(subword of length } j \text{ starting at } i\text{)}$$
$$w_j = d - j + 1 \qquad \text{(longer matches get lower weights)}$$

A Domain Specific Kernel: Weighted Degree Kernel



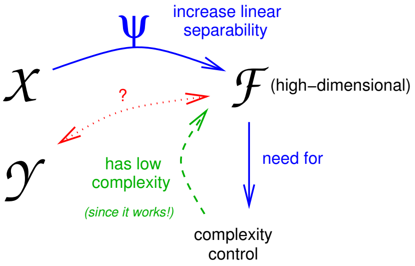
Dimensionality 29, test error $5.5 \pm 0.7\%$

The Three Spectra Compared



Summary

- Clarify role of embedding through the kernel in terms of effective dimensionality of the data in feature space.
- Theoretical contribution to better understanding of kernel methods.
- New diagnosis tool for model selection.
- Future work: effective dimensionality dependend learning bounds.


Ψ
increase linear separability
$\mathcal{X}$
$\mathcal{F}$
(high–dimensional)
?
$\mathcal{Y}$
has low complexity
(since it works!)
need for
complexity control