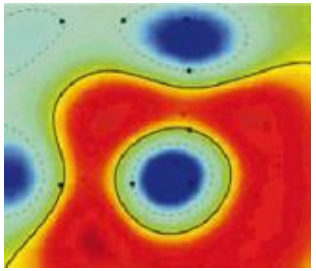
Kernel Methods

Introduction to SVMs, KPCA, RDE



Lecture by Klaus-Robert Müller, TUB 2016

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

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

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

(introduce slack variables if training data not separated correctly)

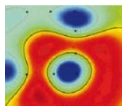
$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i \cdot ((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) - 1\right).$$

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

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

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

Substitute both into L to get the *dual problem*



Dual Problem

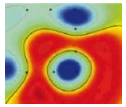
$$\text{maximize} \qquad W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

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

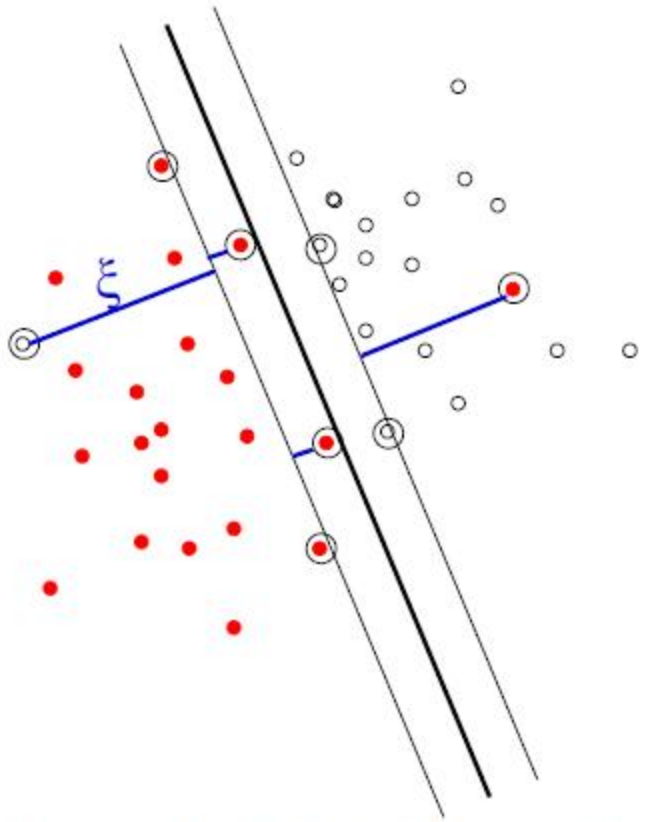
Note: solution determined by training examples (SVs) on /in the margin. Remark: duality gap.

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

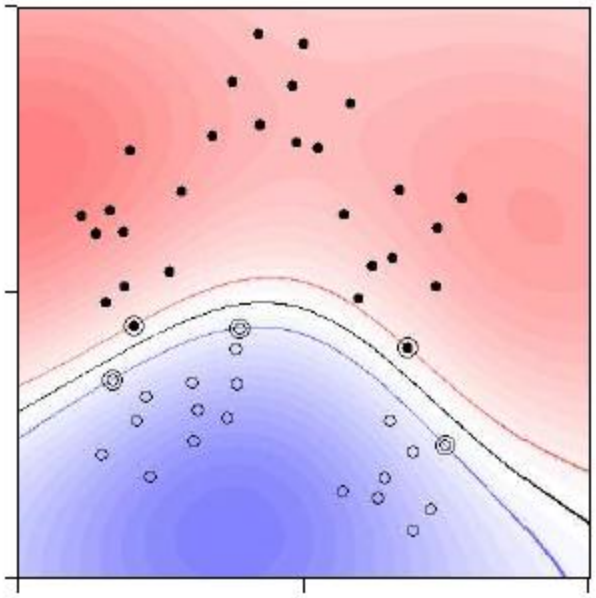
$$y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] = 1 \quad (on \ /\text{in margin}) \longrightarrow \quad \mathbf{x}_i \text{ Support Vector}$$




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"

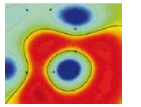
$$
\begin{aligned}
f(\mathbf{x}) &= \operatorname{sgn}(\mathbf{w} \cdot \Phi(\mathbf{x}) + b) \\
&= \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i \, \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b\right) \\
&= \operatorname{sgn}\left(\sum_{i \in \#\mathrm{SVs}} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i)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

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

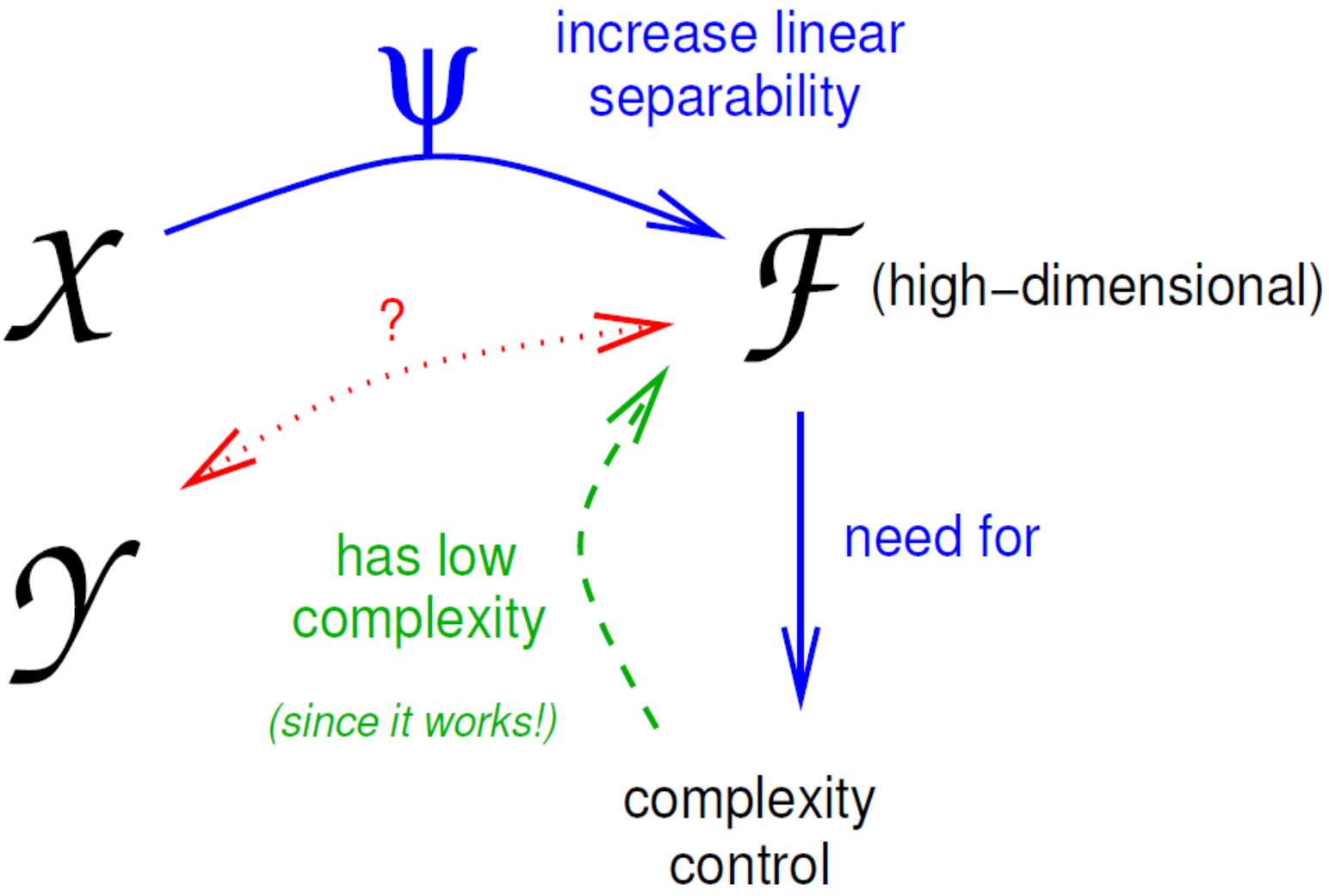
So far…

- 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?

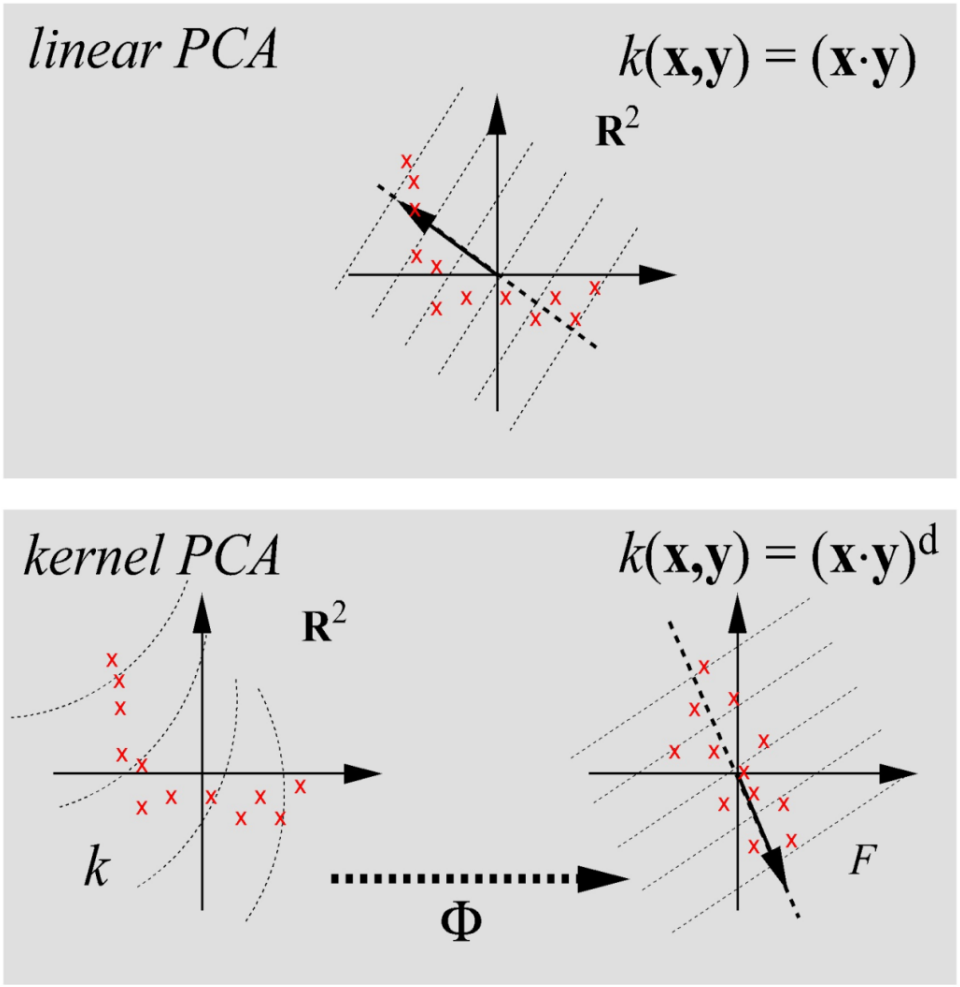


Learning in Kernel Spaces



[cf. Braun, Buhmann, Müller 2008]

Example: Kernel PCA



[cf. Schölkopf, Smola, Müller 98]

Kernel PCA and the 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,

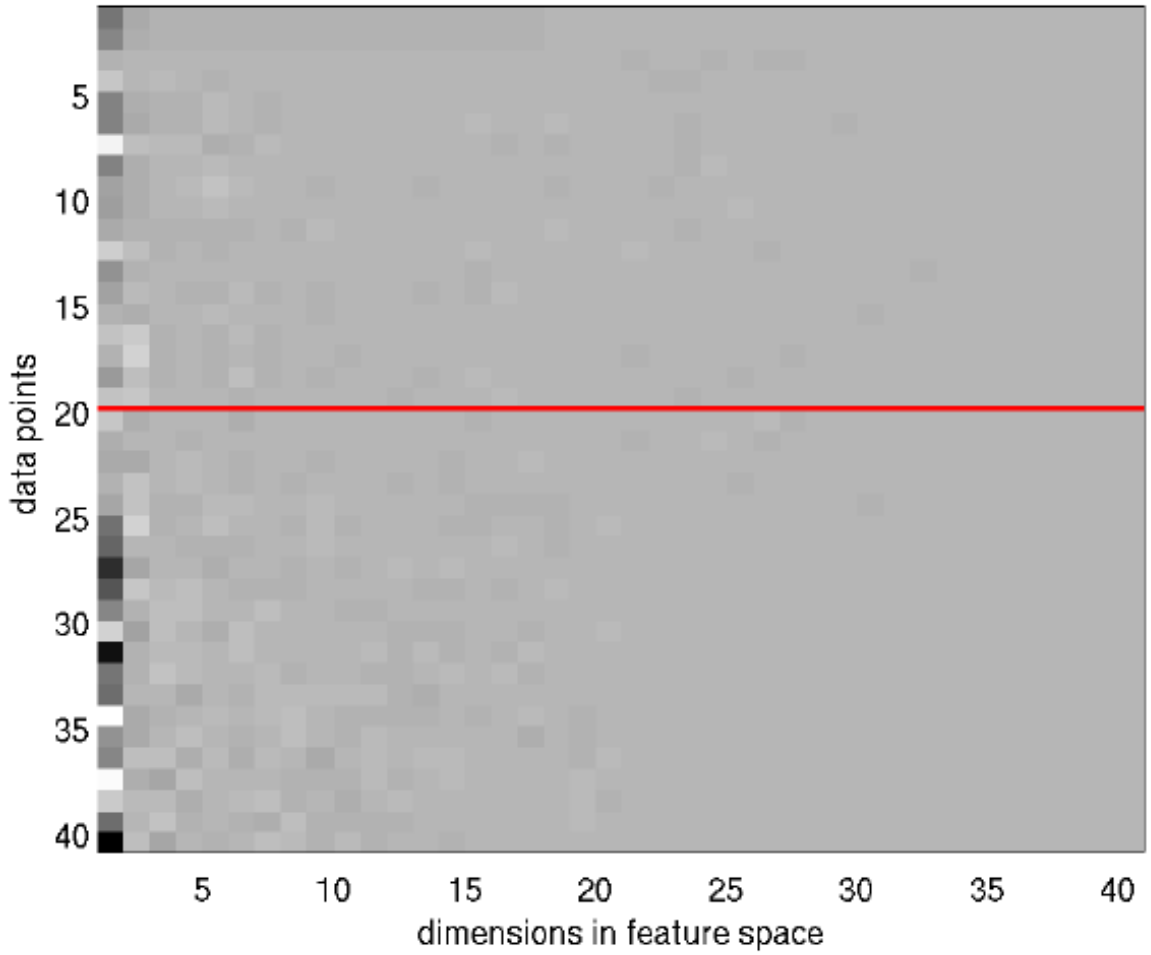
$$\mathbf{F} = \mathbf{U}\mathbf{L}^{1/2}$$

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



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

Note that variance of data becomes smaller and smaller.

Variances in Feature Space (unsupervised learning)

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

kPCA 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

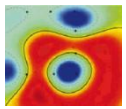
$$s_i = \mathbf{u}_i{}^\top \mathbf{y},$$

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

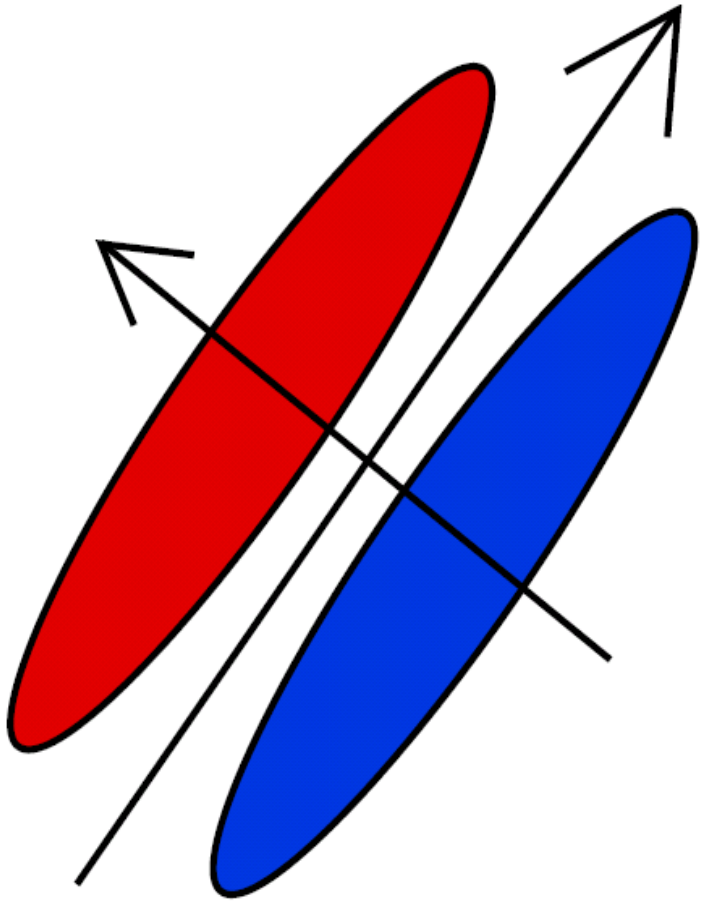
$$\Pi_m \mathbf{y} = \sum_{i=1}^{m} \mathbf{u}_i \mathbf{u}_i{}^\top \mathbf{y}.$$

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

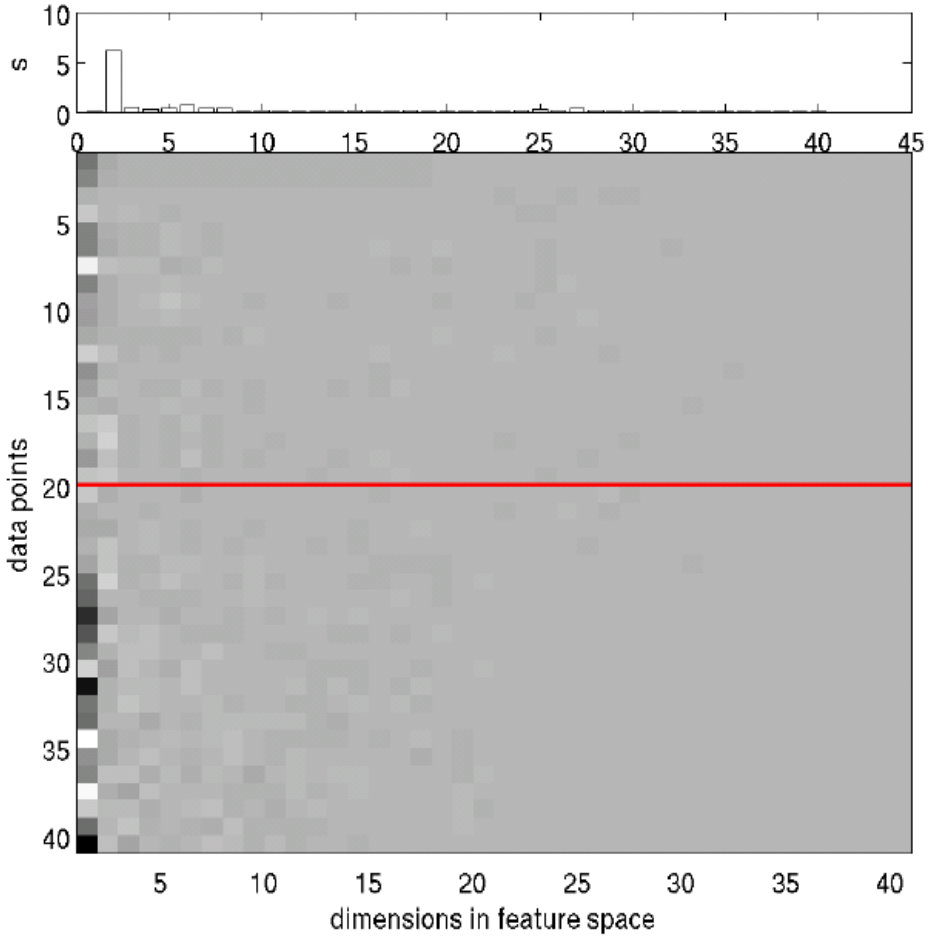

[cf. Braun, Buhmann, Müller 08]

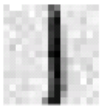
Class Information and large PCA directions

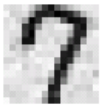


Large PCA directions need not be informative!

Zip data revisited: location of label information





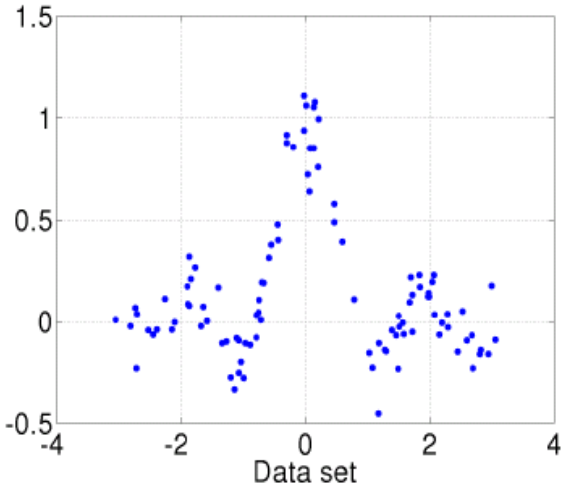


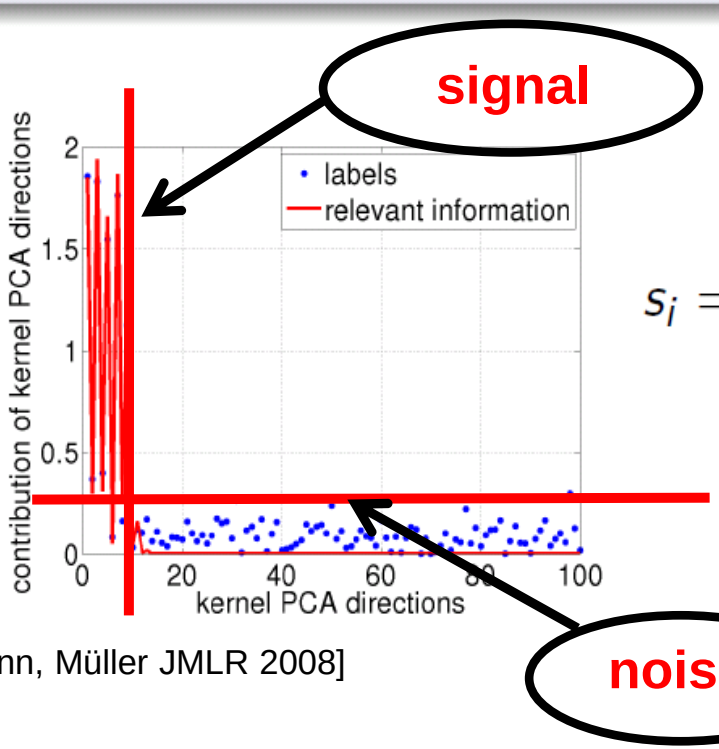
Information about class membership concentrated in direction 2, almost absent from dimensions with small variance!

A 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.



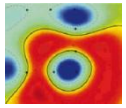


$$s_i = \mathbf{u}_i^\top \mathbf{y},$$

[cf. Braun, Buhmann, Müller JMLR 2008]

Outline

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.



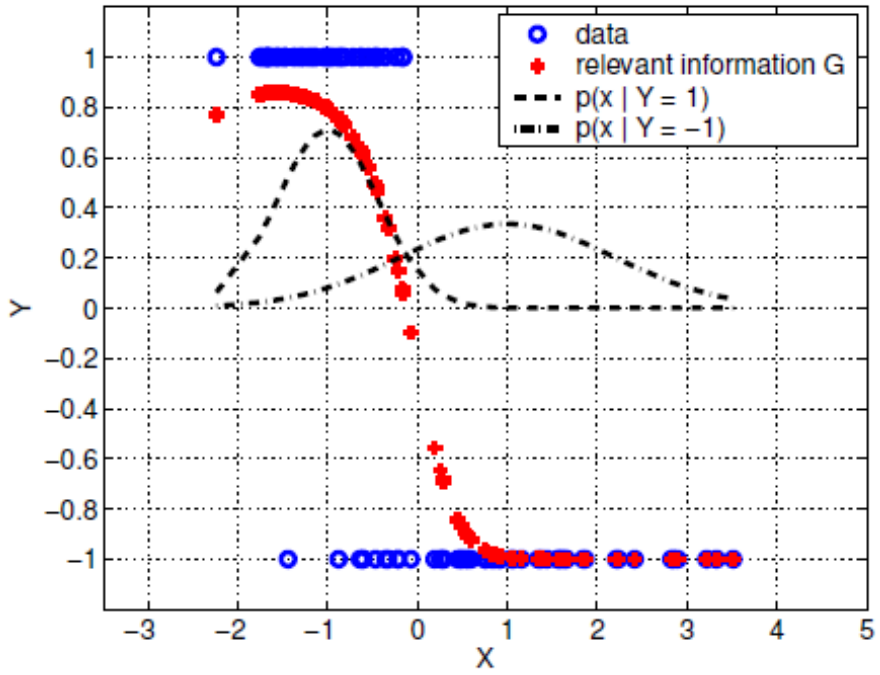
Relevant Information

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

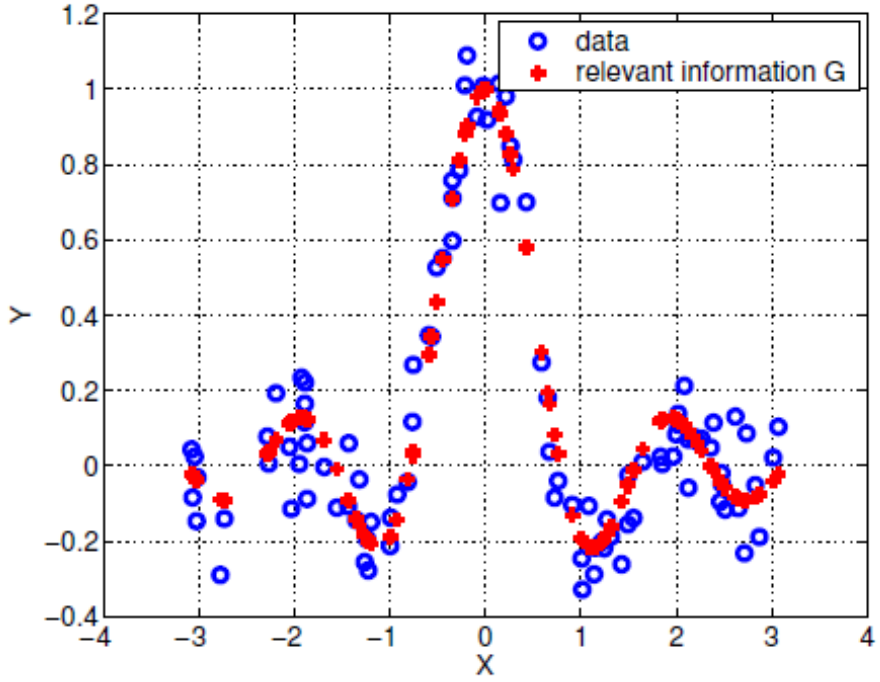
$$Y_i = g(X_i) + \varepsilon_i, \qquad \text{"outputs"} = \text{"smooth part"} + \text{"noise"}$$

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

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



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 $\mathbf{K}$ 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 II

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

The asymptotic setting:

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

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

  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 III

finite sample setting

$$\mathbf{K}f(a) = \frac{1}{N}\sum_{j=1}^{N} k(a, X_j) f(X_j)$$

u_i eigenvector of $\mathbf{K}$

$s_i = {u_i}^\top G$

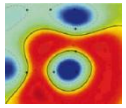
$\rightsquigarrow$

asymptotic setting

$$T_k f(s) = \int_{\mathcal{X}} k(s,t) f(t) P(dt)$$

ψ_i eigenfunction of T_k

$\sigma_i = \langle \psi_i, g \rangle$



Discussion of asymptotic setting

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

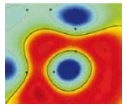
Assumption

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

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

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

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


A 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.

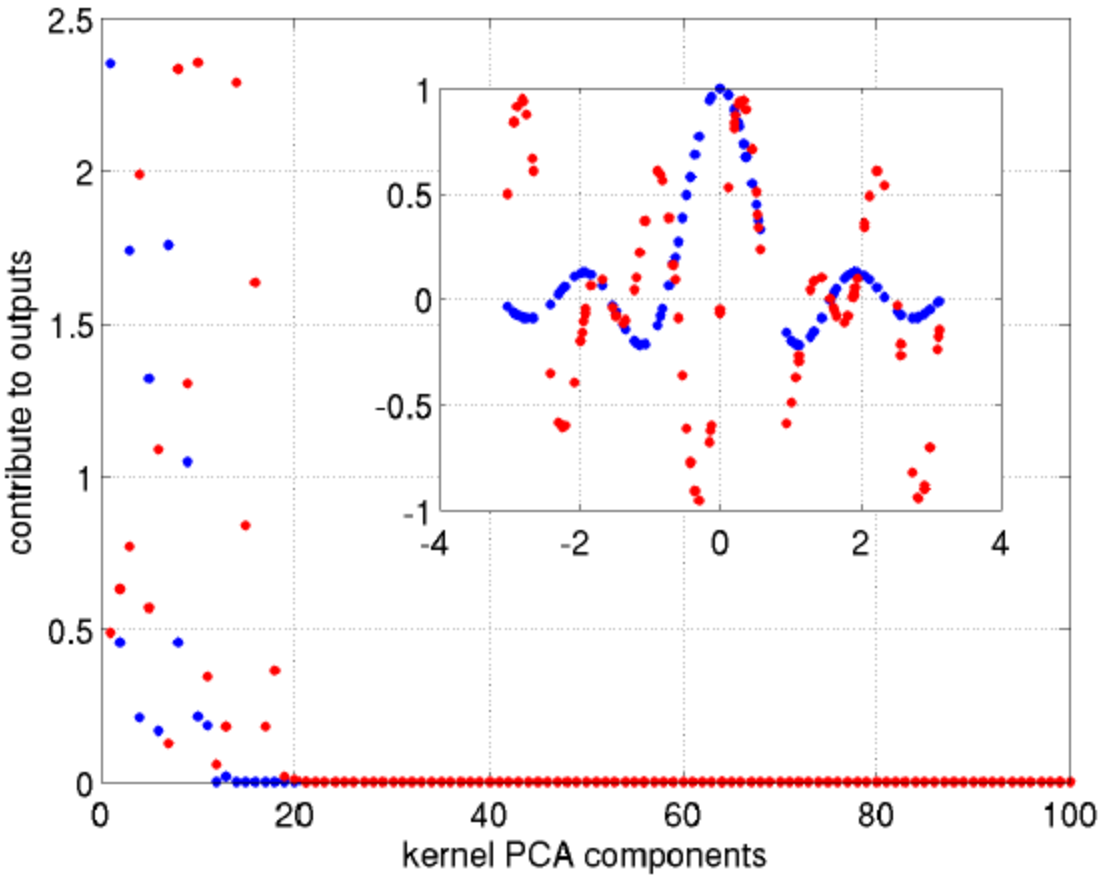
Theorem

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,

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

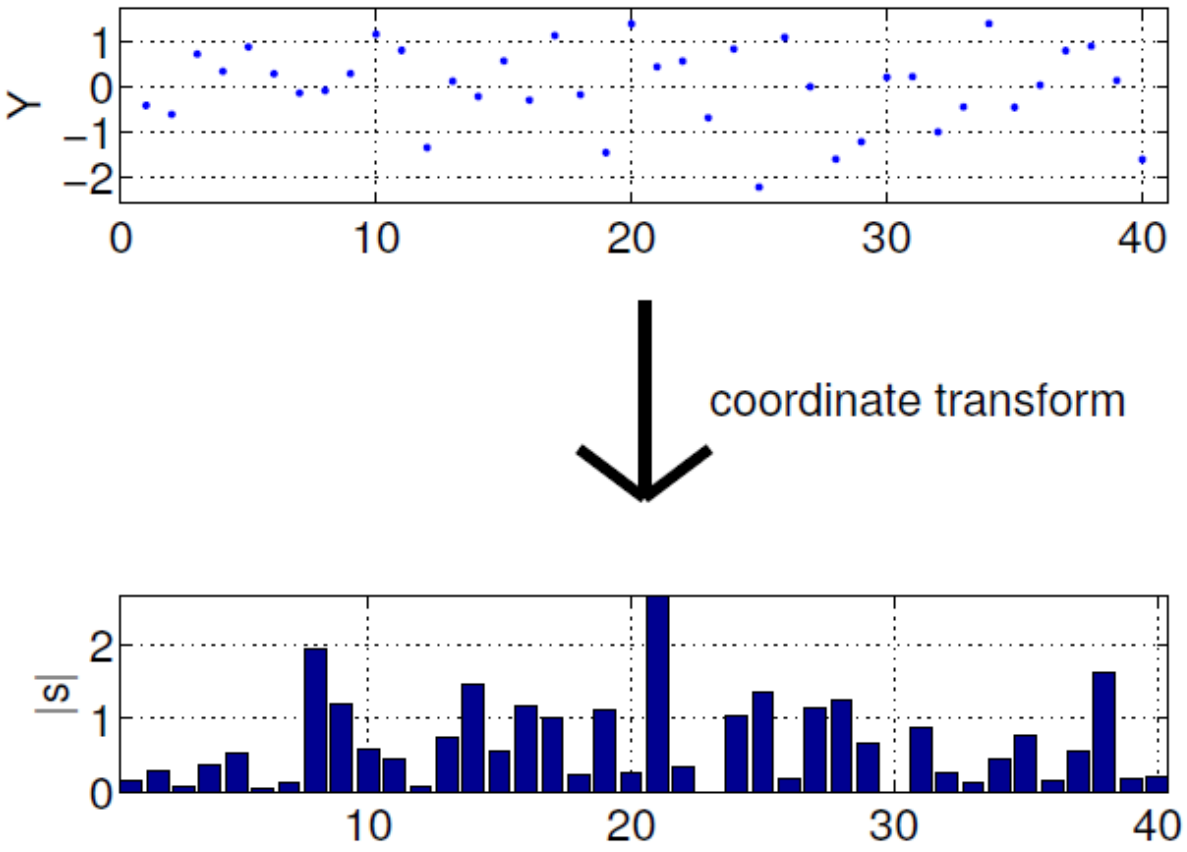
[cf. Braun, Buhmann, Müller JMLR 2008]

Example



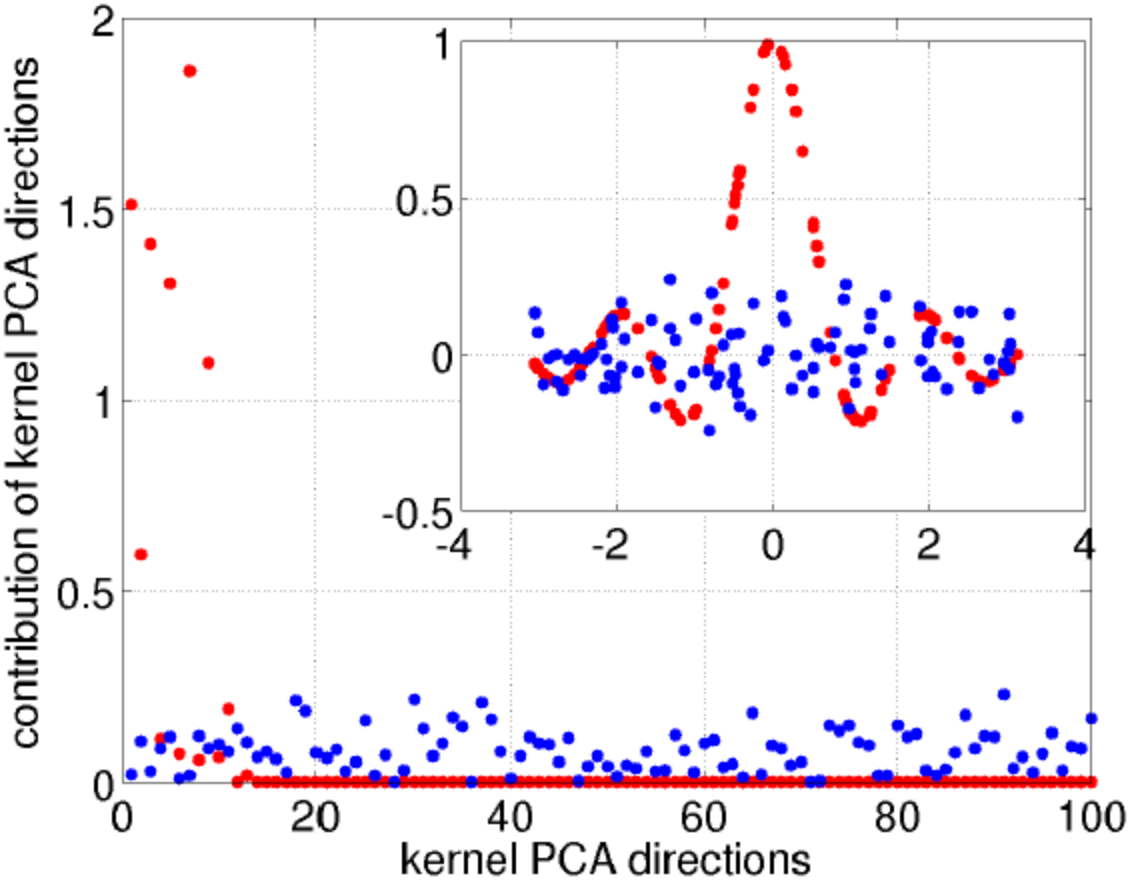
(rbf-kernel with $w = 1$, $\mathrm{sinc}(x)$ function and $\cos(x)\sin(5x)$.)

Location of zero-mean noise



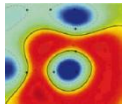
Since $\mathbf{U}^\top$ is a random rotation, noise stays the same.

Example

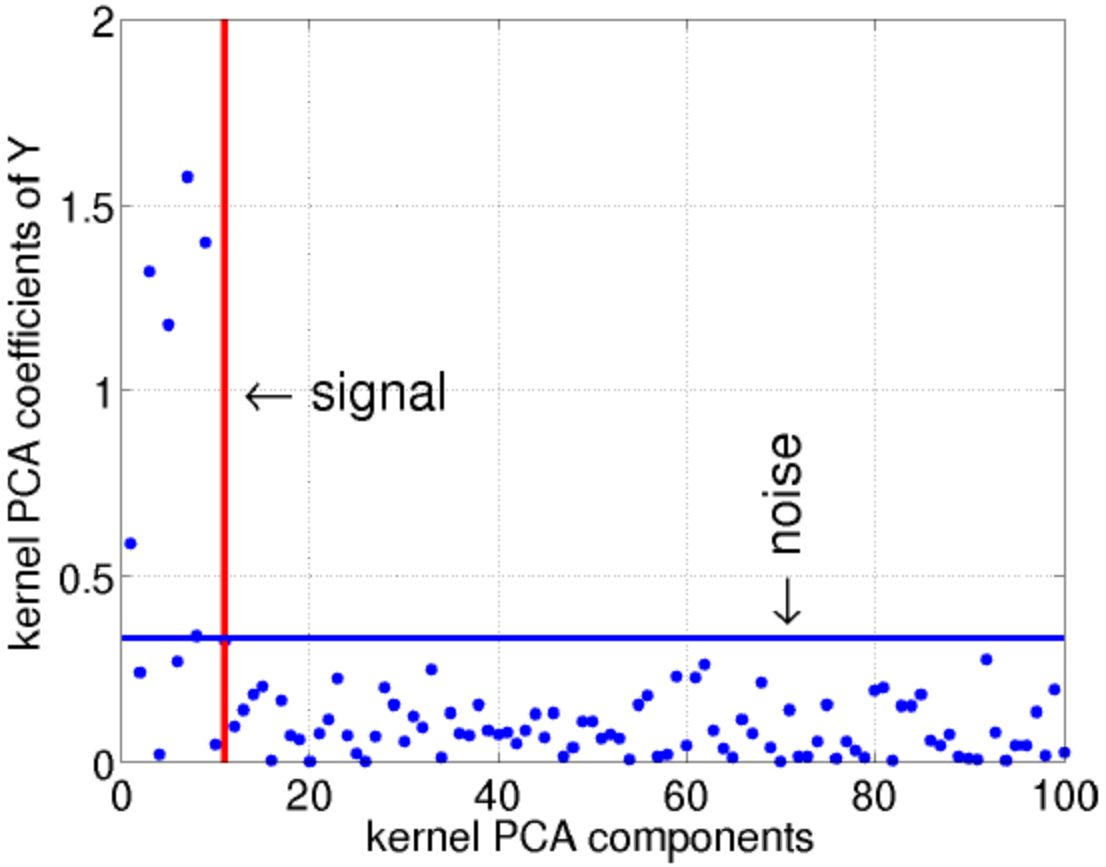

contribution of kernel PCA directions
kernel PCA directions
0
20
40
60
80
100
0
0.5
1
1.5
2
-4
-2
0
2
4
-0.5
0.5
1

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.
- Distinguishing between complex and noisy data sets.



Estimating dimensionality: fitting a 2 component model



Find cut-off dimension which separates the two parts.

Estimating dimensionality: fitting a 2 component model

Assumption:

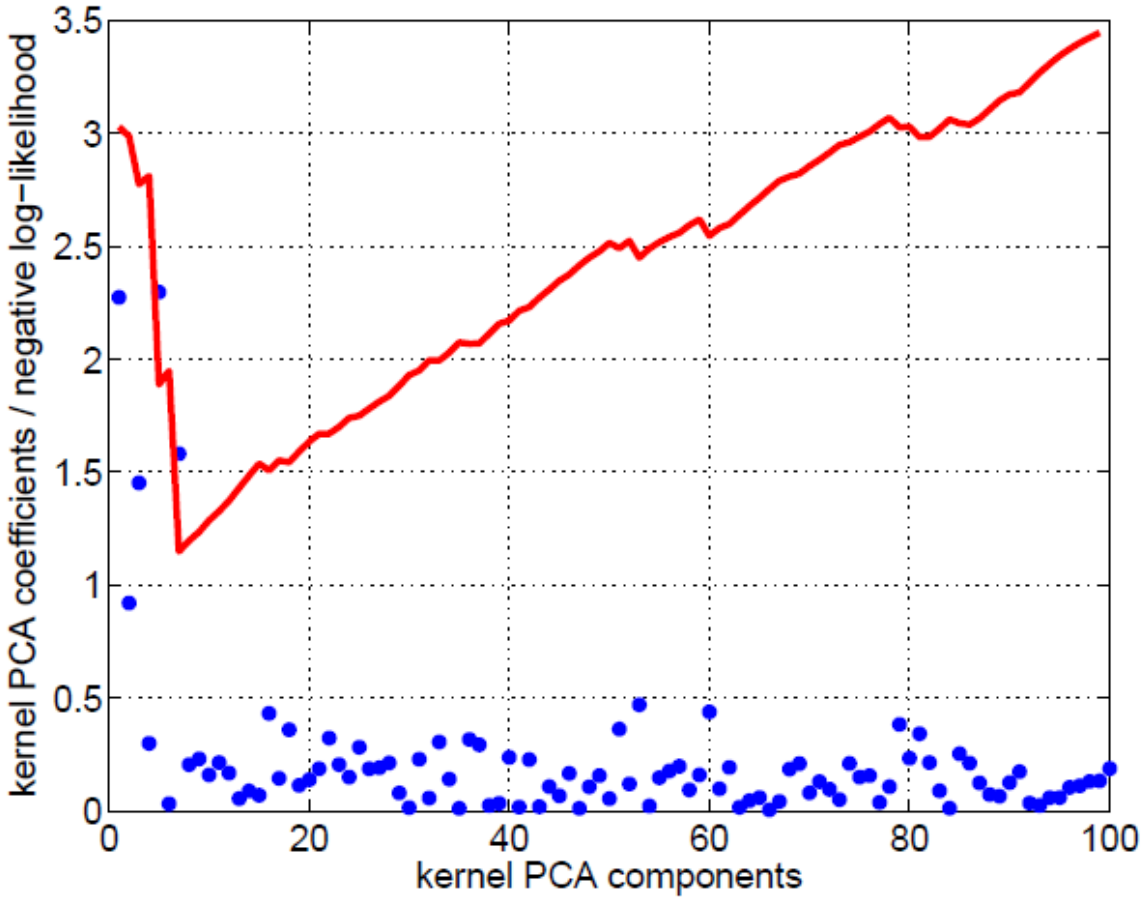
$$s_i \sim \begin{cases} \mathcal{N}(0, \sigma_1^2) & 1 \leq i \leq d \\ \mathcal{N}(0, \sigma_2^2) & d < i \leq n \end{cases}$$

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 dimensionality: fitting a 2 component model



The resulting log-likelihoods.

Estimating the noise level

Project labels $\mathbf{y}$ to subspace of estimated dimensionality $\hat{d}$

$$\hat{\mathbf{y}} = \sum_{i=1}^{\hat{d}} \mathbf{u}_i \mathbf{u}_i^\top \mathbf{y}.$$

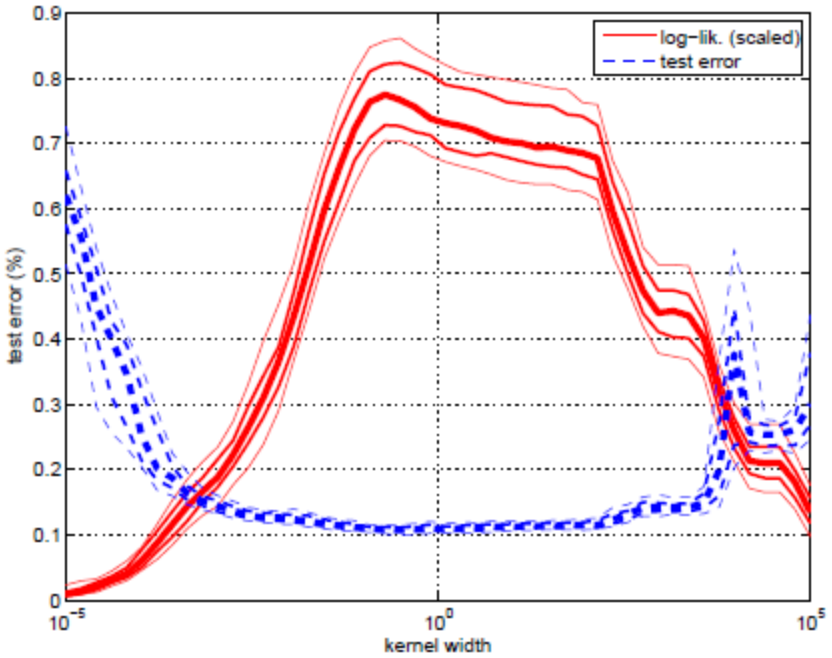
Estimate noise level by comparing projected labels and original labels using loss function L

$$\hat{err} = \frac{1}{N} \sum_{i=1}^{N} L(\hat{\mathbf{y}}, \mathbf{y}).$$

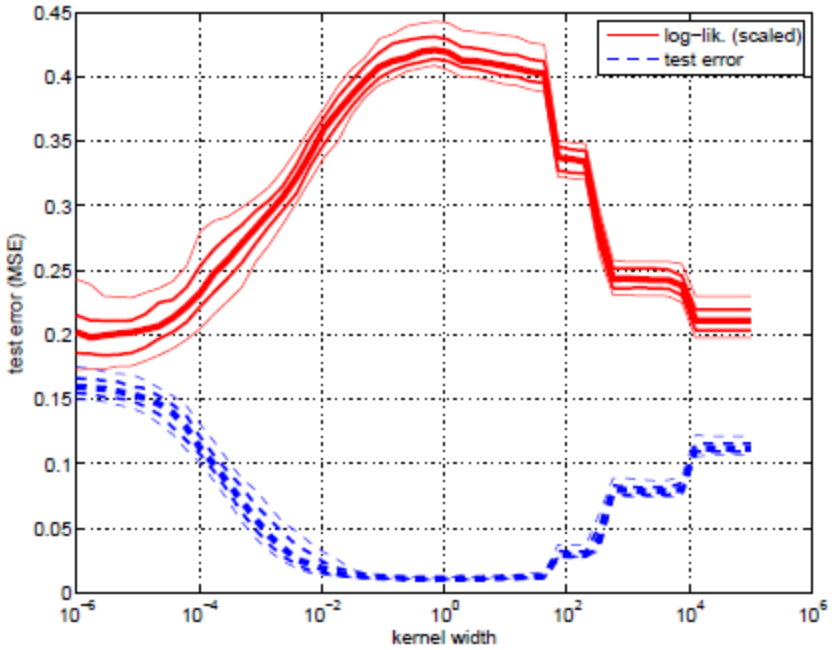
Modelselection

Idea: Use kernel which separates noise from data best.

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

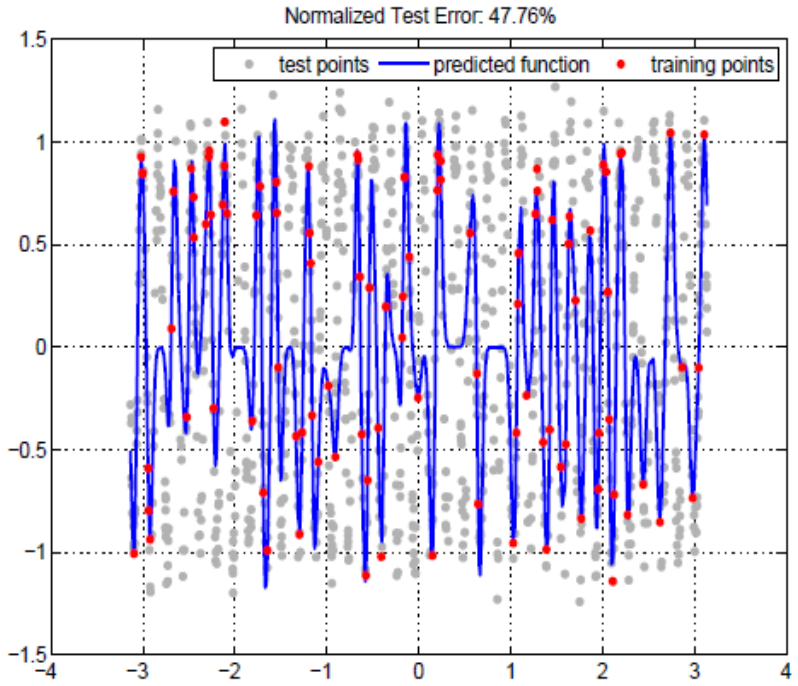


classification
(banana)

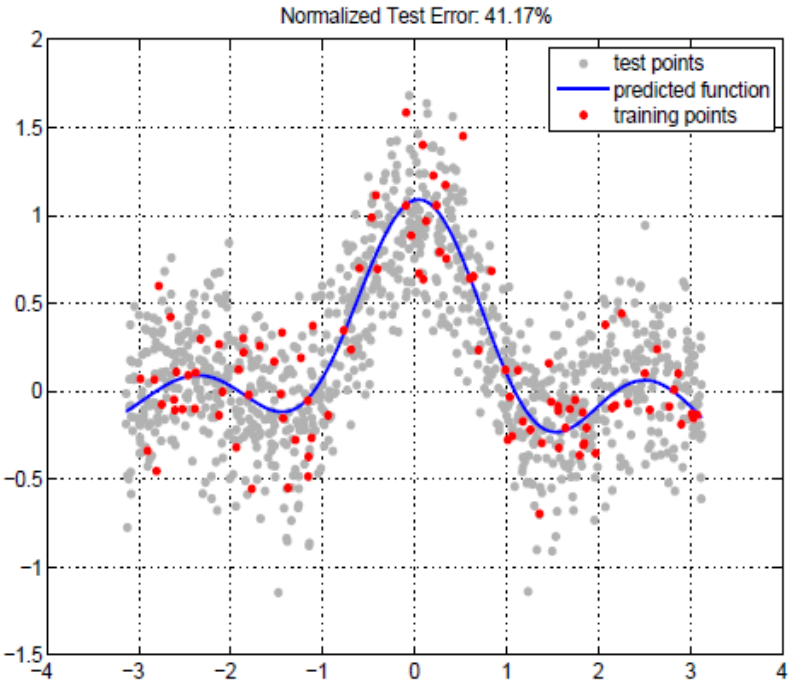


regression
(noisy sinc)

Noisy vs. complex data

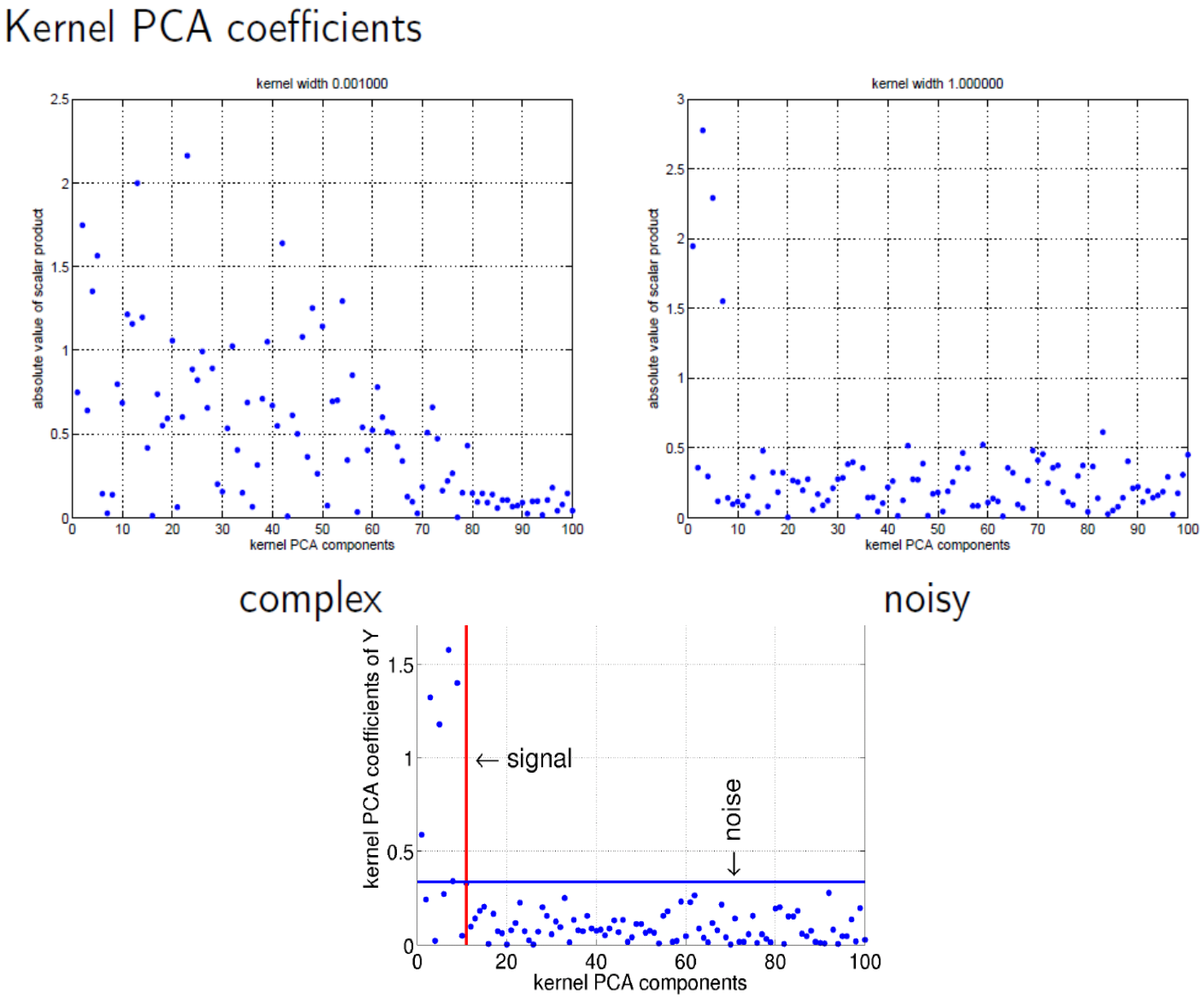


complex



noisy

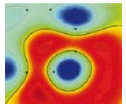
Noisy vs. complex data II



Noisy vs. complex data III

Estimated Dimensionalities distinguish both cases clearly.

data set	dimensions	est. noise-level
complex data set	50	15.84%
noisy data set	7	26.97%



Benchmark Data

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

kPCR: (kernel) least-squares on the denoised data

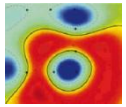
KRR: kernel ridge regression

SVM: support vector machines

Benchmark Data Sets: categorizing data

	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

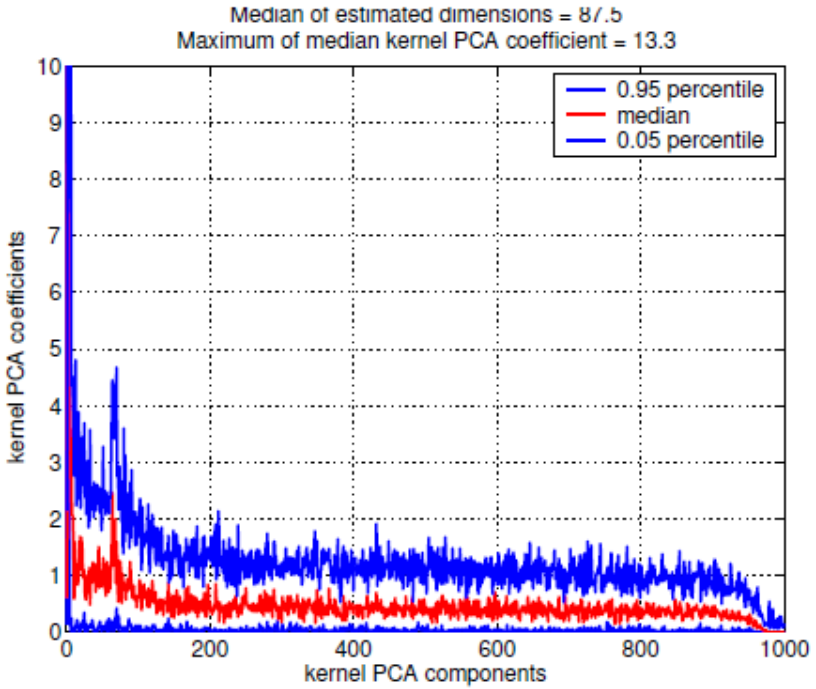
Genes are not encoded in one piece on the DNA, but in multiple parts.

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



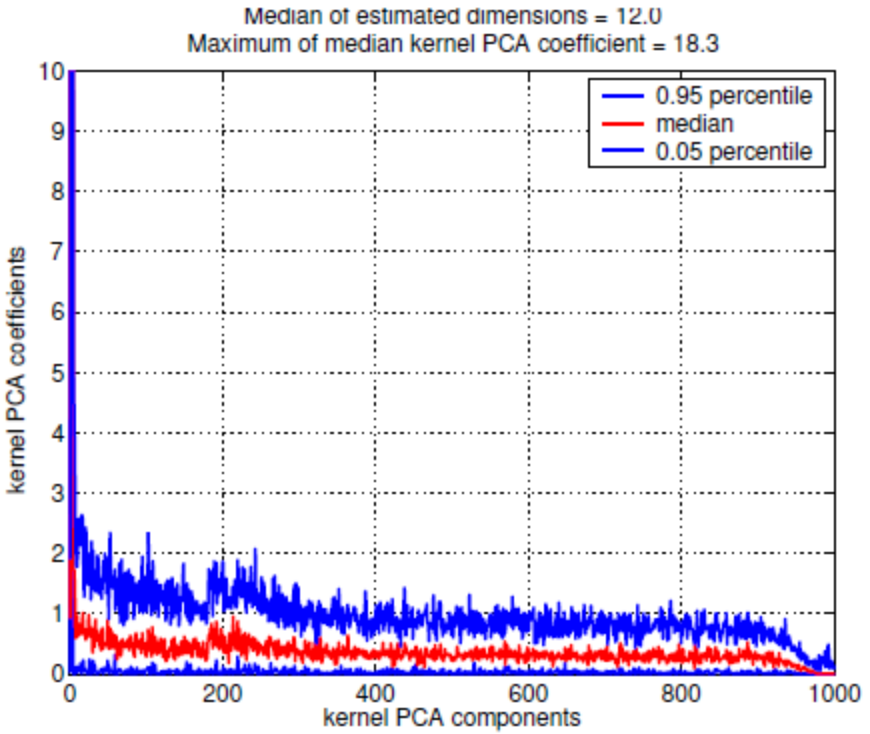
Dimensionality 87, test error $12.9 \pm 0.9\%$.

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

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

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.

Domain specific weighted degree kernel

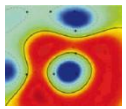
Weighted degree kernel is defined as

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

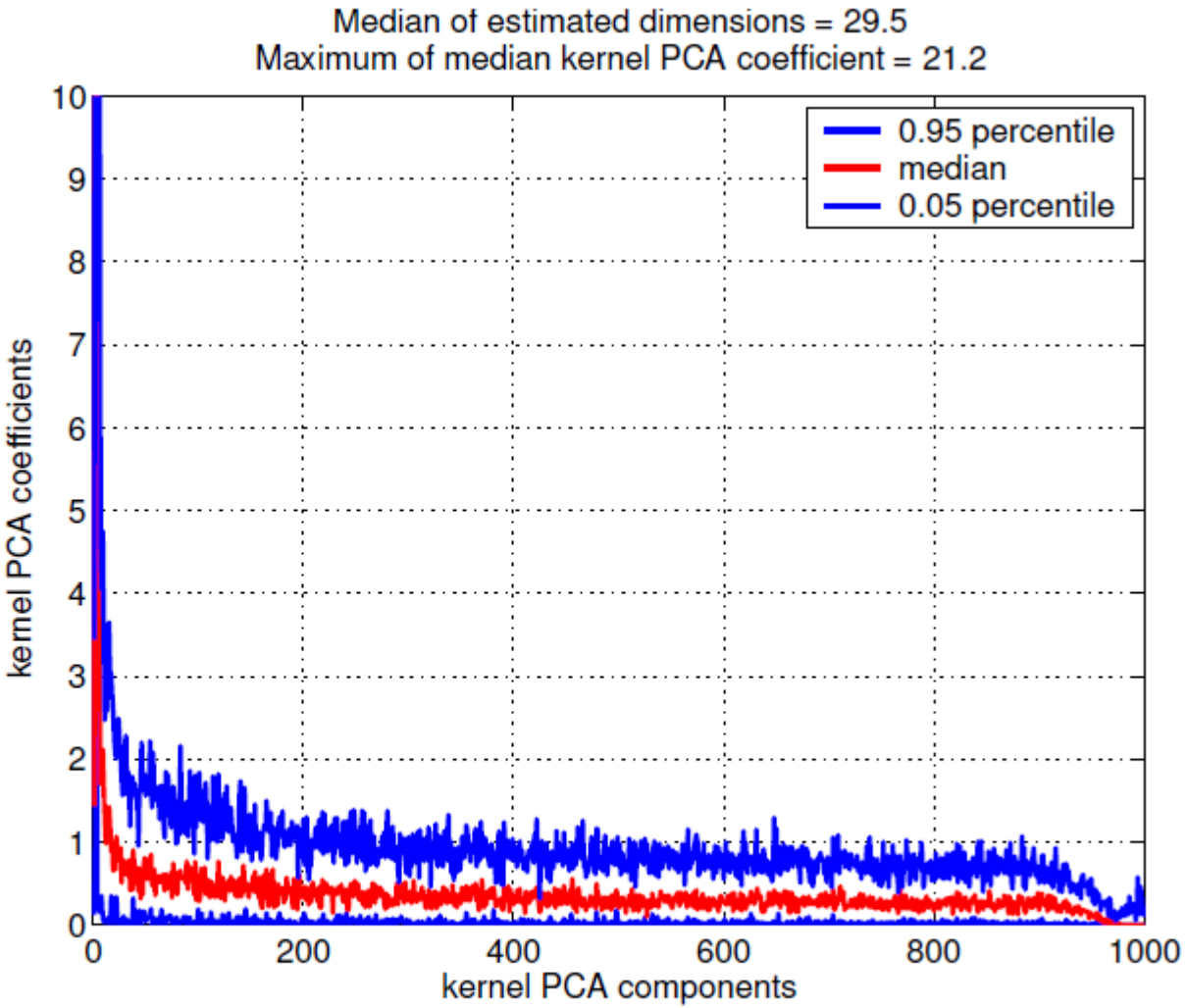
with:

$$u_{j,i}(x) = x_i x_{i+1} \dots x_{i+j-1} \quad \text{(subword of length } j \text{ starting at } i\text{)}$$

$$w_j = d - j + 1 \quad \text{(longer matches get lower weights)}$$

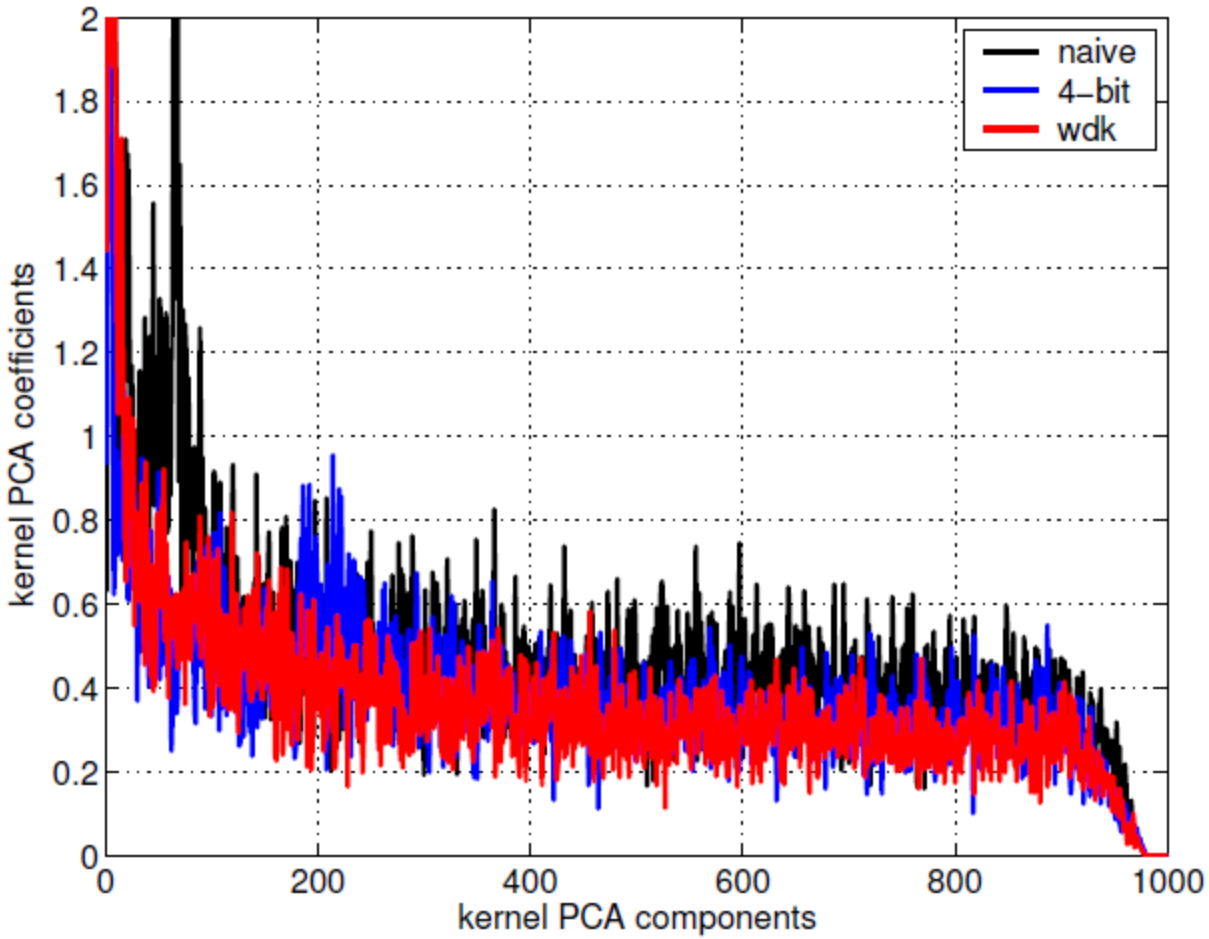


Domain specific weighted degree kernel

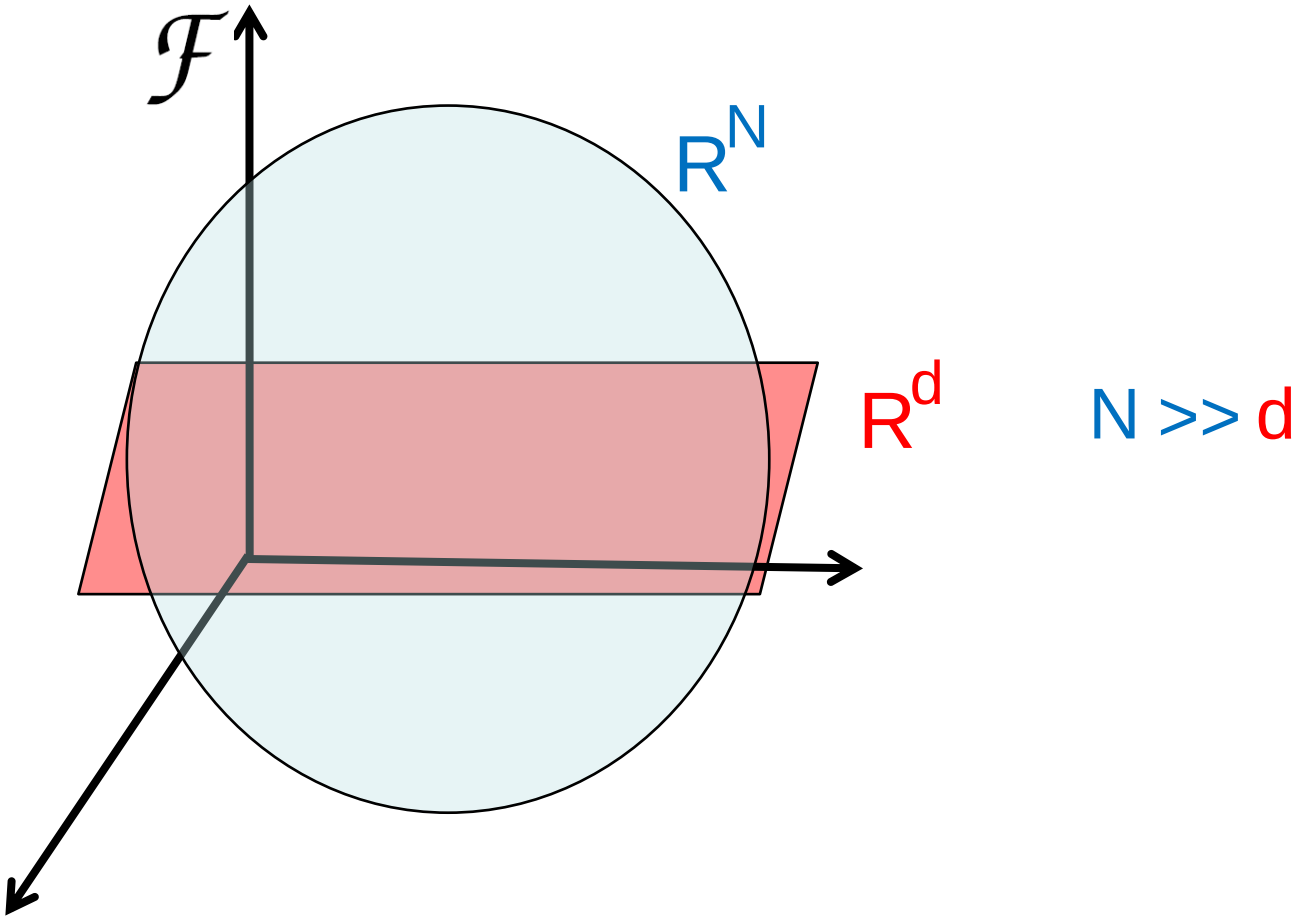


Dimensionality 29, test error $5.5 \pm 0.7\%$

The three spectra compared

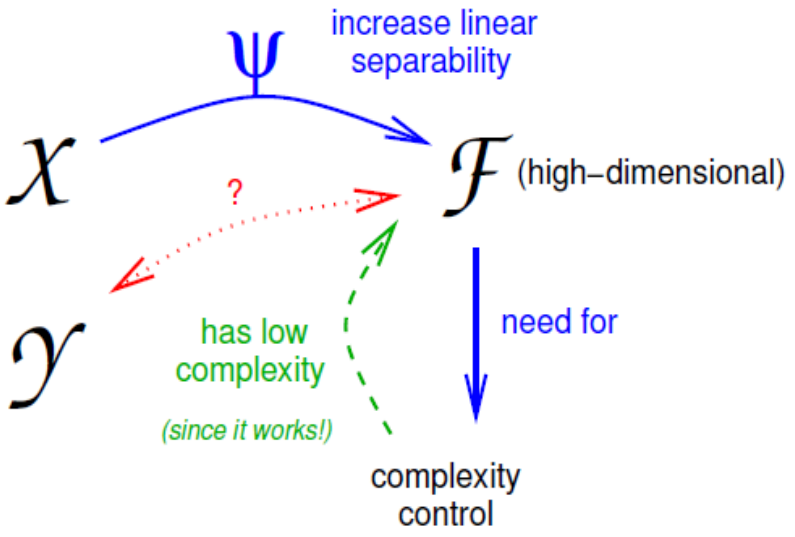


SVM: a cartoon



Some insights on kernel methods

- 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.



Representation is what matters!