Denoising and Dimension Reduction in Feature Space

Algorithms in Complex Systems
September 24–26, 2007, EURANDOM, Eindhoven, NL

Klaus-Robert Müller, Mikio L. Braun
Technische Universität Berlin, Germany
and Fraunhofer Institute FIRST, Berlin, Germany

Joint work with

Claudia Sannelli\textsuperscript{1}, Joachim Buhmann\textsuperscript{2}

\textsuperscript{1} Technische Universität Berlin, Germany
\textsuperscript{2} ETH Zürich, Switzerland
Basic Ideas of Statistical Learning Theory I

Three scenarios: regression, classification & density estimation.

Learn $f$ from examples

$$(x_1, y_1), \ldots, (x_N, y_N) \in \mathbb{R}^N \times \mathbb{R}^M \text{ or } \{\pm 1\}, \quad \text{generated from } P(x, y),$$

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

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

is minimal (Risk Minimization (RM)).

**Problem**: $P$ is unknown. $\rightarrow$ need an induction principle.

*Empirical risk minimization (ERM)*: replace the average over $P(x, y)$ by an average over the training sample, i.e. minimize the training error

$$R_{emp}[f] = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} |f(x_i) - y_i|^2$$
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.
  
  Theorem (Vapnik 95): with a probability of at least \( 1 - \eta \),
  
  \[
  R[f] \leq R_{emp}[f] + \sqrt{\frac{d (\log \frac{2N}{d} + 1) - \log(\eta/4)}{N}}.
  \]

- 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
• hyperplane \( y = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b) \) in canonical form if 
\[
\min_{\mathbf{x}_i \in \mathcal{X}} |(\mathbf{w} \cdot \mathbf{x}_i) + b| = 1, \text{ i.e. scaling freedom removed.}
\]
• larger margin \( \sim 1/||\mathbf{w}|| \) is giving better generalization \( \rightarrow \) LMC!
Theorem (Vapnik 95): For hyperplanes in canonical form VC–dimension satisfying

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

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 \( \|w\|^2 \) \rightarrow good generalization, i.e. low risk, i.e. optimize

\[ \min \|w\|^2 \]

- independent of the dimensionality of the space!
Feature Spaces and “Curse of Dimensionality”

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

\[ \Phi : \mathbb{R}^N \rightarrow F \]

\[ x \mapsto \Phi(x) \]

where \( N \ll \text{dim}(F) \).

To get data \((\Phi(x_1), y_1), \ldots, (\Phi(x_N), y_N) \in F \times \mathbb{R}^M \) 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 \( \Phi \) 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.
Example: all second order monomials

\[ \Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \]
\[ (x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \]
Kernel “Trick”: An Example

(cf. Boser, Guyon & Vapnik 1992)

\[
(\Phi(x) \cdot \Phi(y)) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(y_1^2, \sqrt{2} y_1 y_2, y_2^2)^\top
\]
\[
= (x \cdot y)^2
\]
\[
=: k(x, y)
\]

- Scalar product in \textbf{(high dimensional)} feature space can be computed in \(\mathbb{R}^2\)!
- works only for Mercer Kernels \(k(x, y)\).
If $k$ is a continuous kernel of a positive integral operator on $L_2(\mathcal{D})$ (where $\mathcal{D}$ is some compact space),

$$\int f(x)k(x,y)f(y) \, dx \, dy \geq 0, \text{ for } f \neq 0,$$

it can be expanded as

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

with $\lambda_i > 0$, and $N_F \in \mathbb{N}$ or $N_F = \infty$. In that case

$$\Phi(x) := \begin{pmatrix} \sqrt{\lambda_1} \psi_1(x) \\ \sqrt{\lambda_2} \psi_2(x) \\ \vdots \end{pmatrix}$$

satisfies $(\Phi(x) \cdot \Phi(y)) = k(x, y)$. 
Examples of common kernels:

Polynomial  \( k(x, y) = (x \cdot y + c)^d \)

RBF  \( k(x, y) = \exp \left( -\frac{\|x - y\|^2}{2 \sigma^2} \right) \)

inverse multiquadric  \( k(x, y) = \frac{1}{\sqrt{\|x - y\|^2 + c^2}} \)

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 hypothesis 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

- \( \psi \) increases linear separability
- \( X \) maps to \( \mathcal{F} \) (high-dimensional)
- \( Y \) has low complexity
  - (since it works!)
- Need for complexity control

Basic Ideas
The Kernel “Trick”

- \( F \) (high-dimensional)
- Has low complexity (since it works!)
- Need for complexity control
Kernelizing Linear Algorithms—PCA

\[ w_{\text{PCA}} \]

\[ w_{\text{Fisher}} \]

\[ k(x,y) = (x \cdot y) \]

\[ k(x,y) = (x \cdot y)^d \]

\[ F = \mathbb{R}^{20} \]

\( \Phi \)

PCA in High-Dimensional Feature Spaces

\[ x_1, \ldots, x_N, \quad \Phi : \mathbb{R}^D \rightarrow F, \quad C = \frac{1}{N} \sum_{j=1}^{N} \Phi(x_j)\Phi(x_j)^\top \]

Eigenvalue problem

\[ \lambda \mathbf{V} = C \mathbf{V} = \frac{1}{N} \sum_{j=1}^{N} (\Phi(x_j) \cdot \mathbf{V})\Phi(x_j). \]

For \( \lambda \neq 0 \), \( \mathbf{V} \in \text{span}\{\Phi(x_1), \ldots, \Phi(x_N)\} \), thus \( \mathbf{V} = \sum_{i=1}^{N} \alpha_i \Phi(x_i) \).

Multiplying with \( \Phi(x_k) \) from the left yields

\[ N\lambda (\Phi(x_k) \cdot \mathbf{V}) = (\Phi(x_k) \cdot C \mathbf{V}) \quad \text{for all} \; k = 1, \ldots, N \]
Nonlinear PCA as an Eigenvalue Problem

Define an $N \times N$ matrix

$$K_{ij} := (\Phi(x_i) \cdot \Phi(x_j)) = k(x_i, x_j)$$

to get

$$N \lambda K \alpha = K^2 \alpha$$

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

Solve

$$N \lambda \alpha = K \alpha$$

$\rightarrow (\lambda_k, \alpha_k)$

$$(V^k \cdot V^k) = 1 \iff N \lambda_k (\alpha^k \cdot \alpha^k) = 1$$
Feature Extraction

Compute projections on the Eigenvectors

\[ \mathbf{V}^k = \sum_{i=1}^{M} \alpha_i^k \Phi(x_i) \]

in \( F \):

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

\[ f_k(x) = (\mathbf{V}^k \cdot \Phi(x)) = \sum_{i=1}^{M} \alpha_i^k (\Phi(x_i) \cdot \Phi(x)) \]

\[ = \sum_{i=1}^{M} \alpha_i^k k(x_i, x) \]
Example: RBF Kernel, 8 Principal Components

\[ k(x, y) = \exp \left( -\frac{\|x-y\|^2}{0.1} \right) \]
Kernel PCA components ("features") allow us to construct a feature mapping and look at the data points in feature map:

Let \( K = k(x_i, x_j) \), and let \( K = ULU^\top \) be the eigendecomposition of \( K \).

Then,

\[
F = UL^{1/2}
\]

is a matrix such that

\[
FF^\top = UL^{1/2}L^{1/2}U^\top = K.
\]

This means that the rows of \( F \) are the transformed input points such that their scalar products are \( k(x_i, x_j) \).
Example: ZIP Data Set

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

Note that variance of data becomes smaller and smaller.
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
In a supervised setting, the goal is to predict outputs $y_i$ (class labels, real numbers) from inputs $x_i$.

Contributions of kernel PCA components can be computed by

$$s_i = u_i^\top y,$$

with $u_i$ eigenvector of kernel matrix $K$, $y = (y_1, \ldots, y_N)$ vector of outputs.

Projection of outputs to first $m$ kernel PCA components given by

$$\Pi_m y = \sum_{i=1}^{m} u_i u_i^\top y.$$

Ideally, $|s_i|$ is large only for a few directions.
Large PCA directions need not be informative!
Information about class membership concentrated in direction 2, almost absent from dimensions with small variance!
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.
Define “relevant information” by separating the noise from the outputs:

\[ Y_i = g(X_i) + \varepsilon_i, \]

\[ g(x) = E(Y|X = x), \]

\[ G = (g(X_1), \ldots, g(X_N)). \]

“outputs” = “smooth part” + “noise”

The (population) relevant information

The (sample) relevant information
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

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

- This operator has eigenvectors $u_i$, which are also the kernel PCA components.
- The contribution of the $i$th 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 quantities 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 $\psi_i$ 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

<table>
<thead>
<tr>
<th>finite sample setting</th>
<th>asymptotic setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\textbf{K}f(a) = \frac{1}{N} \sum_{j=1}^{N} k(a, X_j)f(X_j)$</td>
<td>$T_k f(s) = \int_{\mathcal{X}} k(s, t) f(t) P(dt)$</td>
</tr>
<tr>
<td>$u_i$ eigenvector of $\textbf{K}$</td>
<td>$\psi_i$ eigenfunction of $T_k$</td>
</tr>
<tr>
<td>$s_i = u_i^\top G$</td>
<td>$\sigma_i = \langle \psi_i, g \rangle$</td>
</tr>
</tbody>
</table>
Discussion of the Asymptotic Setting

Under the following assumption, asymptotic coefficients $\sigma_i$ 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 $g = T_k h$):

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

$$\Rightarrow \sigma_i = \lambda_i \alpha_i = O(\lambda_i).$$

(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,

\[
\frac{1}{\sqrt{N}} |u_i^\top G| < 2l_i a_r c_i (1 + O(rN^{-1/4}))
\]

\[
+ ra_r \Lambda_r O(1) + T_r + \sqrt{A T_r} O(N^{-1/4}) + ra_r \sqrt{\Lambda_r} O(N^{-1/2}),
\]

where

- \( c_i \): measures size of the eigenvalue cluster around \( l_i \)
- \( a_r = \sum_{i=1}^{r} |\alpha_i| \): measure for size of the first \( r \) components
- \( \Lambda_r \): sum of all eigenvalues smaller than \( \lambda_r \)
- \( A \): supremum norm of \( g \)
- \( T_r \): error of projecting \( g \) to the space spanned by the first \( r \) eigenfunctions
Proof Sketch (1/5)

Decomposition

Decompose scalar product by truncating the kernel function and the function \( g \):

\[
g \rightsquigarrow \tilde{g} = \sum_{\ell=1}^{r} \alpha_\ell \lambda_\ell \psi_\ell, \quad k(x, y) \rightsquigarrow \tilde{k}(x, y) = \sum_{j=1}^{r} \lambda_j \psi_j(x) \psi_j(y).
\]

Analogously, \( K \rightsquigarrow \tilde{K}, \ u_i \rightsquigarrow \tilde{u}_i \), etc. Then:

\[
\frac{1}{\sqrt{N}} |u_i^\top g(X)| \leq \frac{1}{\sqrt{N}} |u_i^\top \tilde{g}(X)| + \frac{1}{\sqrt{N}} ||g(X) - \tilde{g}(X)||
\]

and

\[
\frac{1}{\sqrt{N}} |u_i^\top \tilde{g}(X)| \leq \sum_{\ell=1}^{r} |\alpha_\ell| \sum_{j=1}^{r} (u_i^\top \tilde{u}_j) \left[ \frac{1}{\sqrt{N}} \lambda_\ell \psi_\ell(X)^\top \tilde{u}_j \right].
\]
Proof Sketch (2/5)
The degenerate kernel case

Bound scalar product between sample vector of eigenfunction and eigenvector of truncated kernel matrix:

\[ \tilde{l}_j \tilde{u}_j = \tilde{K} \tilde{u}_j = \tilde{\Psi} \tilde{\Lambda} \tilde{\Psi}^\top \tilde{u}_j \Rightarrow \tilde{l}_j \tilde{\Psi}^+ \tilde{u}_j = \tilde{\Lambda} \tilde{\Psi}^\top \tilde{u}_j \]

with \( \tilde{\Psi}_{ij} = \psi_j(X_i)/\sqrt{N} \). Taking norms,

\[
\frac{1}{\sqrt{N}} \lambda \psi(X) ^\top \tilde{u}_j \leq ||\tilde{\Lambda} \tilde{\Psi}^\top \tilde{u}_j|| \leq \tilde{l}_j ||\tilde{\Psi}^+||.
\]
Proof Sketch (3/5)
Bounding the truncation error

Bound on scalar product between eigenvectors of original kernel matrix $K$ and kernel matrix $\tilde{K}$ using truncated kernel.

With $\tilde{E} = K - \tilde{K}$, and \textit{sin-theta-theorem}

$$|u_i^\top \tilde{u}_j| \leq \min \left( \frac{\|\tilde{E}\|}{|l_i - \tilde{l}_j|}, 1 \right) =: \omega_{ij}$$

Combining with the previous bound, we will be interested in bounding

$$\sum_{j=1}^{r} \tilde{l}_j \omega_{ij} = \sum_{j \in J(l_i)} \tilde{l}_j \omega_{ij} + \sum_{j \notin J(l_i)} \tilde{l}_j \omega_{ij} \leq 2|J(l_i)|l_i + 2r\|\tilde{E}\|$$

with

$$J(l_i) = \{1 \leq j \leq r \mid l_i/2 \leq \tilde{l}_j \leq 2l_i\}.$$
Proof Sketch (4/5)
Bounding the truncation error for the function

By the law of large numbers,

\[
\frac{1}{n} \|g(\mathbf{X}) - \tilde{g}(\mathbf{X})\|^2 \to \|g - \tilde{g}\|^2_{L(P_X)} = \sum_{j=r+1}^{\infty} \alpha_j^2 \lambda_j^2 =: \tilde{T}^2.
\]

If \( g \) is bounded by \( F \),

\[
\text{Var}_{P_X}((g - \tilde{g})^2) \leq \|g - \tilde{g}\|_{\infty}^2 \|g - \tilde{g}\|^2 = F^2 \tilde{T}^2
\]

and by the Chebychev-inequality we get

\[
\frac{1}{\sqrt{N}} \|g(\mathbf{X}) - \tilde{g}(\mathbf{X})\| \leq \tilde{T} + \sqrt{F \tilde{T}(n\delta)^{-1/4}}.
\]
Proof Sketch (5/5)

Summary

1. Eigenfunction sample vectors and “truncated” eigenvectors
\[
\frac{1}{\sqrt{N}} \lambda_\ell \psi_\ell(X)^\top \tilde{u}_j \leq \tilde{l}_j \|\tilde{\Psi}^+\|.
\]

2. Perturbation of kernel truncation
\[
\frac{1}{\sqrt{N}} |u_i^\top \tilde{g}(X)| \leq \|\tilde{\Psi}^+\| \sum_{\ell=1}^{r} |\alpha_\ell| (2|J(l_i)l_i + 2r\|\tilde{E}\|).
\]

3. Truncation of the function \( g \)
\[
\frac{1}{\sqrt{N}} \|g(X) - \tilde{g}(X)\| \leq \tilde{T} + \sqrt{F \tilde{T}} (n\delta)^{-1/4}.
\]
\[
\frac{1}{\sqrt{N}} |u_i^\top g(X)| \leq 2l_i |J(l_i)| \|\tilde{\alpha}\|_1 \|\tilde{\Psi}^+\| + 2r \|\tilde{E}\| \|\tilde{\alpha}\|_1 \|\tilde{\Psi}^+\| + \tilde{T} + \sqrt{F \tilde{T}} (n\delta)^{-1/4}.
\]
An Example

(rbf-kernel with $w = 1$, 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
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.
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

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

\[ \Rightarrow \text{choose } d \text{ which minimizes } - \log \ell(d). \]
Estimating the Dimensionality
Fitting a Two-Component Model

The resulting log-likelihoods.
**Idea:** Use kernel which separates noise from data best.

\[ \leadsto \text{choose kernel such that log-likelihood value at } \hat{d} \text{ is maximal.} \]
Benchmark Data Sets

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

kPCR: (kernel) least-squares on the denoised data
KRR: kernel ridge regression
SVM: support vector machines
### Benchmark Data Sets: Categorizing Data Sets

<table>
<thead>
<tr>
<th></th>
<th>low noise</th>
<th>high noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>low dimensional</td>
<td>banana, thyroid, waveform</td>
<td>breast-cancer, diabetes flare-solar, german heart, titanic</td>
</tr>
<tr>
<td>high dimensional</td>
<td>image, ringnorm</td>
<td>splice</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Aminoacid</th>
<th>Encoded as</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
</tr>
</tbody>
</table>

Dimensionality 87, test error 12.9 ± 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

<table>
<thead>
<tr>
<th>Aminoacid</th>
<th>Encoded as</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(0, 0, 0, 1)</td>
</tr>
<tr>
<td>C</td>
<td>(0, 0, 1, 0)</td>
</tr>
<tr>
<td>G</td>
<td>(0, 1, 0, 0)</td>
</tr>
<tr>
<td>T</td>
<td>(1, 0, 0, 0)</td>
</tr>
</tbody>
</table>

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

All aminoacids are comparably far from one another. But only fixed positions are compared.
A Domain Specific Kernel: Weighted Degree Kernel

Weighted degree kernel is defined as

\[ k(x, x') = \sum_{j=1}^{d} w_j \sum_{i=1}^{N-d} 1\{u_{j,i}(x) = u_{j,i}(x')\} \]

with:

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

\[ w_j = d - j + 1 \quad \text{(longer matches get lower weights)} \]
A Domain Specific Kernel: Weighted Degree Kernel

Median of estimated dimensions = 29.5
Maximum of median kernel PCA coefficient = 21.2

Dimensionality 29, test error 5.5 ± 0.7%
The Three Spectra Compared

The diagram compares the kernel PCA components and coefficients for three different methods: naive, 4-bit, and wdk. The graph is labeled with axes for kernel PCA components and kernel PCA coefficients, with values ranging from 0 to 2 along both axes. The methods are distinguished by color: black for naive, blue for 4-bit, and red for wdk.
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.

\[
\begin{align*}
\mathcal{X} &\xrightarrow{\psi} \mathcal{F} \\
\mathcal{Y} &\xrightarrow{\text{increase linear separability}} \\
\end{align*}
\]

\*Complexity control\*

\*Need for\* (since it works!)