Desiging and combining kernels: some lessons learned from bioinformatics

Jean-Philippe Vert
Jean-Philippe.Vert@mines-paristech.fr

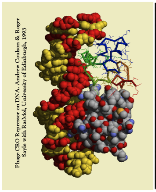
Mines ParisTech & Institut Curie

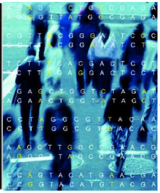
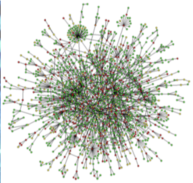
NIPS MKL workshop, Dec 12, 2009.

Kernels are very popular in bioinformatics

Why?

- Many problems can be approached by kernels methods (classification, regression, feature construction, ...)
- Many data with particular structures
  $\rightarrow$ Kernel design
- Need to integrate heterogeneous data
  $\rightarrow$ Kernel combination


Phage CRO Repressor on DNA. Andrew Coulson & Roger Sayle with RasMol, University of Edinburgh, 1993

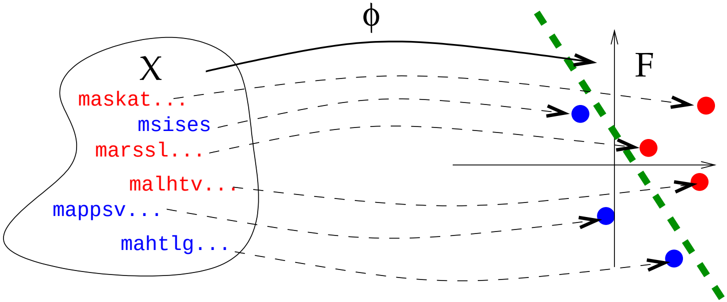



Outline

Outline

What is a GOOD kernel?

- Leads to good performances
- Mathematically valid
- Fast to compute
- Interpretable model (?)



How to MAKE a good kernel?

3 main ideas

1. Define good features

$$K(x,x') = \Phi(x)^\top \Phi(x')$$

2. Define a good metric

$$d(x,x') = \sqrt{K(x,x) + K(x',x') - 2K(x,x')}$$

3. Define a good functional penalty

$$\min_{f \in \mathcal{H}} \left\{ R(f) + \lambda \| f \|_{\mathcal{H}}^2 \right\}$$

How to MAKE a good kernel?

3 main ideas

1. Define good features

$$K(x,x') = \Phi(x)^\top \Phi(x')$$

2. Define a good metric

$$d(x,x') = \sqrt{K(x,x) + K(x',x') - 2K(x,x')}$$

3. Define a good functional penalty

$$\min_{f \in \mathcal{H}} \left\{ R(f) + \lambda \| f \|_{\mathcal{H}}^2 \right\}$$

How to MAKE a good kernel?

3 main ideas

1. Define good features

$$K(x,x') = \Phi(x)^\top \Phi(x')$$

2. Define a good metric

$$d(x,x') = \sqrt{K(x,x) + K(x',x') - 2K(x,x')}$$

3. Define a good functional penalty

$$\min_{f \in \mathcal{H}} \left\{ R(f) + \lambda \| f \|_{\mathcal{H}}^2 \right\}$$

Idea 1: define good features

Motivation

- Estimate a function $f(x) = w^\top \Phi(x)$
- A good feature is more important than a good algorithm!

Examples

- Explicit feature computations
  - substring or subgraph indexation
  - Fisher kernel $\Phi(x) = \nabla_\theta \log P_\theta(x)$
- Implicit feature construction + kernel trick
  - Walk-based graph kernels
  - Mutual information kernels $K(x, x') = \int P_\theta(x) P_\theta(x') d\theta$

Caveats

- One good feature among too many irrelevant ones may not be enough with L_2 regularization

Idea 1: define good features

Motivation

- Estimate a function $f(x) = w^\top \Phi(x)$
- A good feature is more important than a good algorithm!

Examples

- Explicit feature computations
  - substring or subgraph indexation
  - Fisher kernel $\Phi(x) = \nabla_\theta \log P_\theta(x)$
- Implicit feature construction + kernel trick
  - Walk-based graph kernels
  - Mutual information kernels $K(x, x') = \int P_\theta(x) P_\theta(x') d\theta$

Caveats

- One good feature among too many irrelevant ones may not be enough with L_2 regularization

Idea 1: define good features

Motivation

- Estimate a function $f(x) = w^\top \Phi(x)$
- A good feature is more important than a good algorithm!

Examples

- Explicit feature computations
  - substring or subgraph indexation
  - Fisher kernel $\Phi(x) = \nabla_\theta \log P_\theta(x)$
- Implicit feature construction + kernel trick
  - Walk-based graph kernels
  - Mutual information kernels $K(x, x') = \int P_\theta(x) P_\theta(x') d\theta$

Caveats

- One good feature among too many irrelevant ones may not be enough with L_2 regularization

Example: string kernel with substring indexation

Index the feature space by fixed-length strings, i.e.,

$$\Phi(\mathbf{x}) = (\Phi_u(\mathbf{x}))_{u \in \mathcal{A}^k}$$

where $\Phi_u(\mathbf{x})$ can be:

- the number of occurrences of u in $\mathbf{x}$ (without gaps) : spectrum kernel (Leslie et al., 2002)
- the number of occurrences of u in $\mathbf{x}$ up to m mismatches (without gaps) : mismatch kernel (Leslie et al., 2004)
- the number of occurrences of u in $\mathbf{x}$ allowing gaps, with a weight decaying exponentially with the number of gaps : substring kernel (Lohdi et al., 2002)

Idea 2: define a good metric

Motivation

- A kernel defines a Hilbert metric
  $d(x,x') = \sqrt{K(x,x) + K(x',x') - 2K(x,x')}$
- The functions we can learn are smooth w.r.t this metric

$$\left| f(x) - f(x') \right| \leq \| f \|_{\mathcal{H}} d(x,x')$$

Examples

- Edit distances for strings or graphs, local alignment of biological sequences, graph matching distances
- MAMMOTH distance between protein 3D structures

Caveats

- Most "good" distances are not Hilbertian

Idea 2: define a good metric

Motivation

- A kernel defines a Hilbert metric
  $d(x,x') = \sqrt{K(x,x) + K(x',x') - 2K(x,x')}$
- The functions we can learn are smooth w.r.t this metric

$$\left| f(x) - f(x') \right| \leq \| f \|_{\mathcal{H}} d(x,x')$$

Examples

- Edit distances for strings or graphs, local alignment of biological sequences, graph matching distances
- MAMMOTH distance between protein 3D structures

Caveats

- Most "good" distances are not Hilbertian

Idea 2: define a good metric

Motivation

- A kernel defines a Hilbert metric
  $d(x,x') = \sqrt{K(x,x) + K(x',x') - 2K(x,x')}$
- The functions we can learn are smooth w.r.t this metric

$$\left| f(x) - f(x') \right| \leq \parallel f \parallel_{\mathcal{H}} d(x,x')$$

Examples

- Edit distances for strings or graphs, local alignment of biological sequences, graph matching distances
- MAMMOTH distance between protein 3D structures

Caveats

- Most "good" distances are not Hilbertian

Example: local alignment kernel

How to compare 2 protein sequences?

$$\mathbf{x}_1 = \texttt{CGGSLIAMMWFGV}$$
$$\mathbf{x}_2 = \texttt{CLIVMMNRLMWFGV}$$

Find a good alignment π:

```
CGGSLIAMM----WFGV
|...|||||....||||
C---LIVMMNRLMWFGV
```

Two non-Hilbertian metrics

$$SW(\mathbf{x},\mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x},\mathbf{y})} s(\pi).$$

$$K_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = \log \sum_{\pi \in \Pi(\mathbf{x},\mathbf{y})} \exp\left(\beta s\left(\mathbf{x},\mathbf{y},\pi\right)\right),$$

Example: local alignment kernel

How to compare 2 protein sequences?

$\mathbf{x}_1$ = CGGSLIAMMWFGV

$\mathbf{x}_2$ = CLIVMMNRLMWFGV

Find a good alignment π:

```
CGGSLIAMM----WFGV
|...|||||....||||
C---LIVMMNRLMWFGV
```

Two non-Hilbertian metrics

$$SW(\mathbf{x},\mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x},\mathbf{y})} s(\pi).$$

$$K_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = \log \sum_{\pi \in \Pi(\mathbf{x},\mathbf{y})} \exp\left(\beta s\left(\mathbf{x},\mathbf{y},\pi\right)\right),$$

Idea 3: define a good penalty function

Motivation

- The kernel constrains the set of functions over which we optimize (balls in RKHS).
- We may first define a penalty we like, then find the associated kernel.

Examples

- graph Laplacian over gene networks
- cluster kernel for protein remote homology detection

Caveats

- Some penalties may not be RKHS norms (eg, total variation to estimate piecewise constant functions)

Idea 3: define a good penalty function

Motivation

- The kernel constrains the set of functions over which we optimize (balls in RKHS).
- We may first define a penalty we like, then find the associated kernel.

Examples

- graph Laplacian over gene networks
- cluster kernel for protein remote homology detection

Caveats

- Some penalties may not be RKHS norms (eg, total variation to estimate piecewise constant functions)

Idea 3: define a good penalty function

Motivation

- The kernel constrains the set of functions over which we optimize (balls in RKHS).
- We may first define a penalty we like, then find the associated kernel.

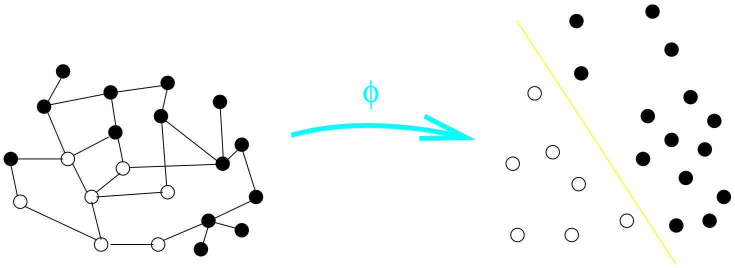
Examples

- graph Laplacian over gene networks
- cluster kernel for protein remote homology detection

Caveats

- Some penalties may not be RKHS norms (eg, total variation to estimate piecewise constant functions)

Example : Kernel on a graph



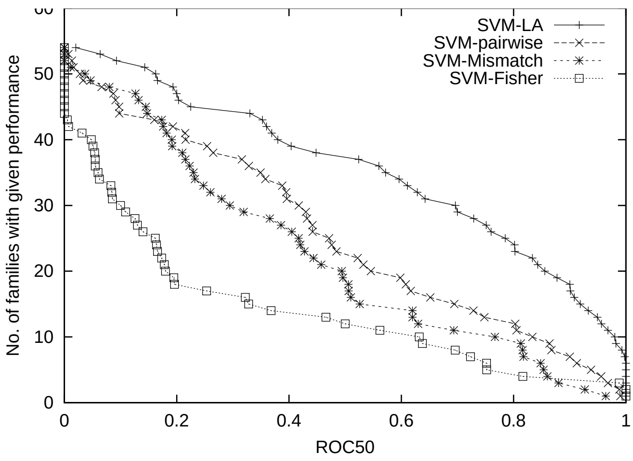
Laplacian-based kernel

The set $\mathcal{H} = \left\{ f \in \mathbb{R}^m : \sum_{i=1}^m f_i = 0 \right\}$ endowed with the norm:

$$\Omega(f) = \sum_{i \sim j} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j) \right)^2$$

is a RKHS whose reproducing kernel is the pseudo-inverse of the graph Laplacian.

The choice of kernel makes a difference



Performance on the SCOP superfamily recognition benchmark.

Outline

Motivation

- We can imagine plenty of kernels for a given application
  - different kernels for the same data (e.g., different string kernels)
  - kernels for different types of data (e.g., integrating string and 3D structures for protein classification)
- Which one to use?
- Perhaps we can combine them to make better than each one individually?

Sum kernels

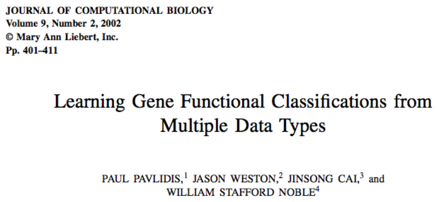
- Consider p kernels $K_1, \ldots, K_p$
- Form the sum (eg, Pavlidis et al., 2002):

$$K = \sum_{i=1}^{p} K_i \,.$$

- Equivalently, concatenate the features of the different kernels
- Equivalently, work in the RKHS $\mathcal{H} = \mathcal{H}_1 \oplus \ldots \oplus \mathcal{H}_p$ with

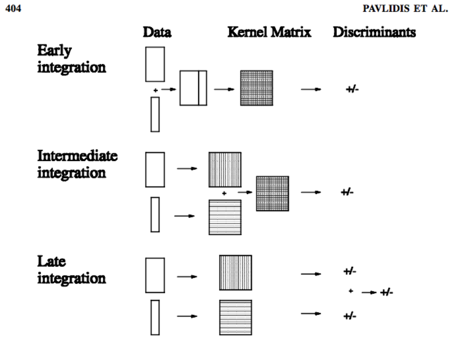
$$\| f \|_{\mathcal{H}}^2 = \inf_{f = f_1 + \ldots + f_p} \sum_{i=1}^{p} \| f_i \|_{\mathcal{H}_i}^2 \,.$$

Some early work

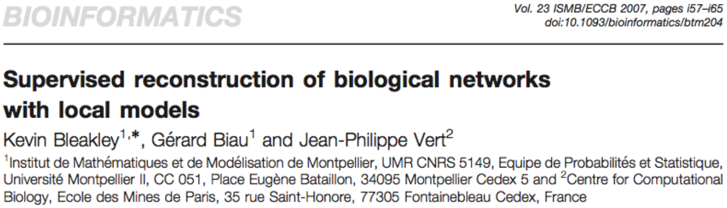

JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 9, Number 2, 2002
© Mary Ann Liebert, Inc.
Pp. 401–411

Learning Gene Functional Classifications from Multiple Data Types

PAUL PAVLIDIS,[1] JASON WESTON,[2] JINSONG CAI,[3] and WILLIAM STAFFORD NOBLE[4]


404 PAVLIDIS ET AL.

Huge improvements can be observed



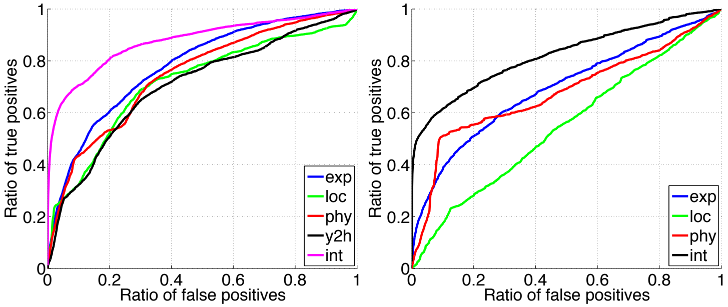
BIOINFORMATICS

Vol. 23 ISMB/ECCB 2007, pages i57–i65
doi:10.1093/bioinformatics/btm204

Supervised reconstruction of biological networks with local models

Kevin Bleakley[1,*], Gérard Biau[1] and Jean-Philippe Vert[2]

[1]Institut de Mathématiques et de Modélisation de Montpellier, UMR CNRS 5149, Equipe de Probabilités et Statistique, Université Montpellier II, CC 051, Place Eugène Bataillon, 34095 Montpellier Cedex 5 and [2]Centre for Computational Biology, Ecole des Mines de Paris, 35 rue Saint-Honore, 77305 Fontainebleau Cedex, France



Multiple kernel learning (MKL)

- Form the convex combination:

$$K = \sum_{i=1}^{p} \eta_i K_i \,.$$

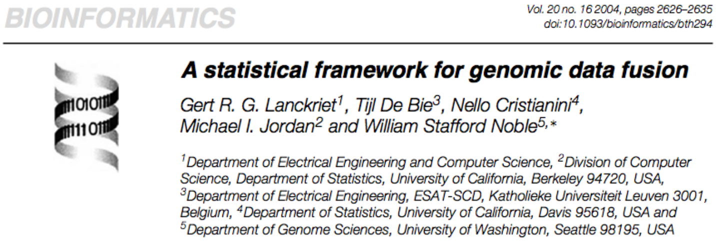
where the weights are chosen to minimize the following convex function under the constraint $tr(K) = 1$ (Lanckriet et al., 2003):

$$h(K) = \inf_{f \in \mathcal{H}_K} \left\{ R(f) + \lambda \| f \|_{\mathcal{H}_K} \right\}$$

- Equivalently, work in the space $\mathcal{H} = \mathcal{H}_1 + \ldots + \mathcal{H}_p$ with non-Hilbertian group L_1 norm (Bach et al., 2004):

$$\| f \|_{\mathcal{H}} = \inf_{f = f_1 + \ldots + f_p} \sum_{i=1}^{p} \| f_i \|_{\mathcal{H}_i} \,.$$

Example: Lanckriet et al. (2004)



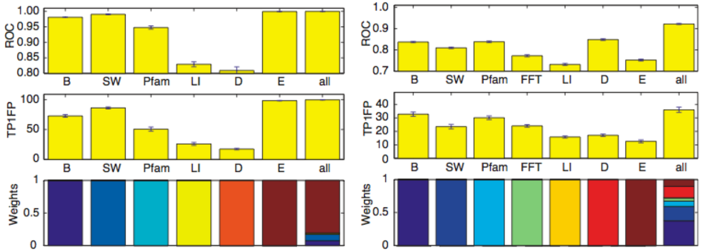
BIOINFORMATICS

Vol. 20 no. 16 2004, pages 2626–2635
doi:10.1093/bioinformatics/bth294

A statistical framework for genomic data fusion

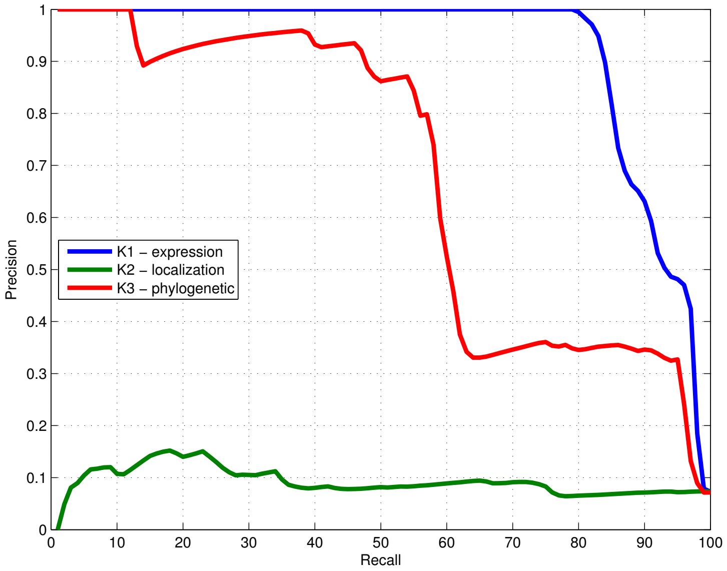
Gert R. G. Lanckriet[1], Tijl De Bie[3], Nello Cristianini[4], Michael I. Jordan[2] and William Stafford Noble[5,*]

[1]Department of Electrical Engineering and Computer Science, [2]Division of Computer Science, Department of Statistics, University of California, Berkeley 94720, USA, [3]Department of Electrical Engineering, ESAT-SCD, Katholieke Universiteit Leuven 3001, Belgium, [4]Department of Statistics, University of California, Davis 95618, USA and [5]Department of Genome Sciences, University of Washington, Seattle 98195, USA

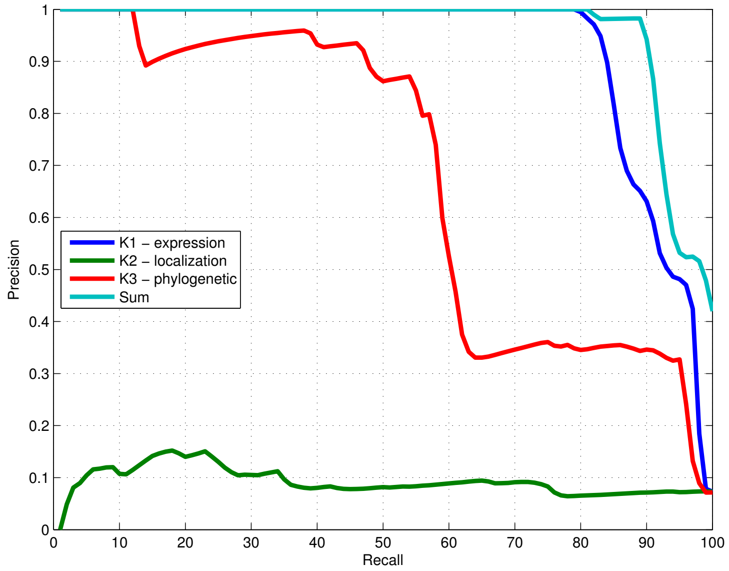


(A) Ribosomal proteins (B) Membrane proteins

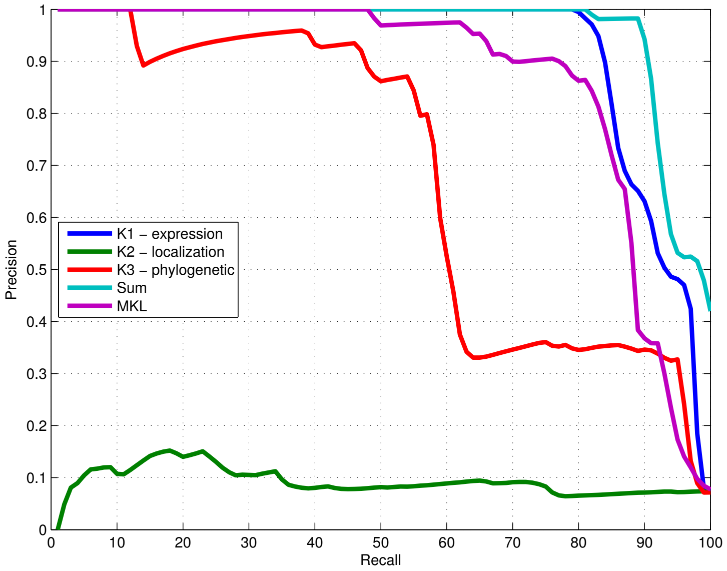
MKL or sum kernel for protein network inference?



MKL or sum kernel for protein network inference?

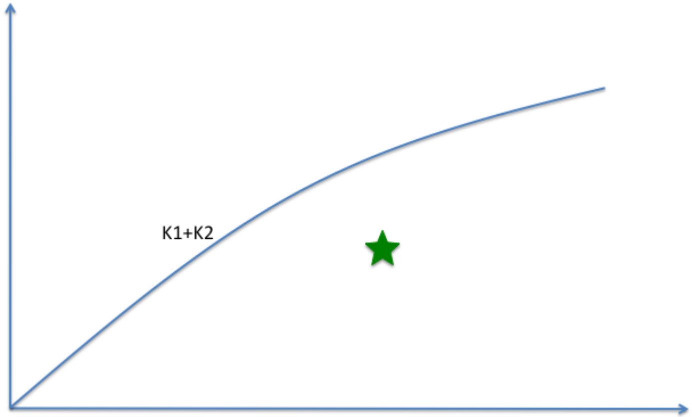


MKL or sum kernel for protein network inference?

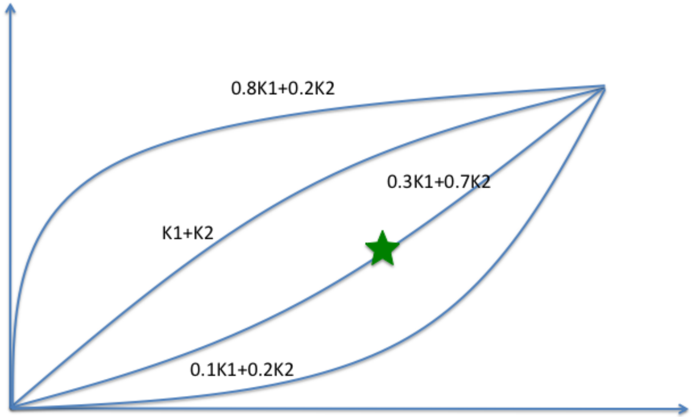


Why MKL does not estimate a good kernel combination

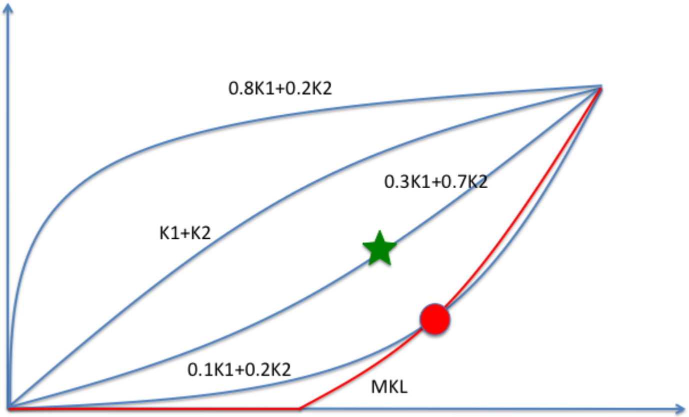
Why MKL does not estimate a good kernel combination



Why MKL does not estimate a good kernel combination

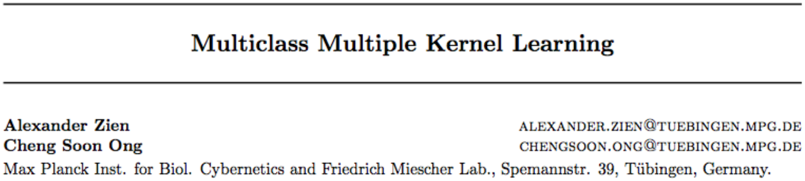


Why MKL does not estimate a good kernel combination



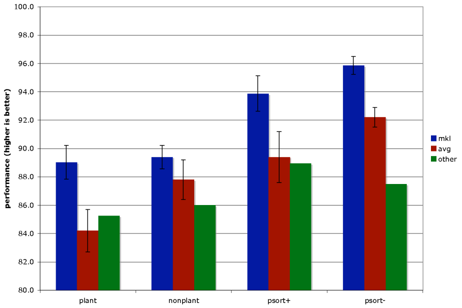
Sometimes MKL works

Subcellular protein classficiation from 69 kernels



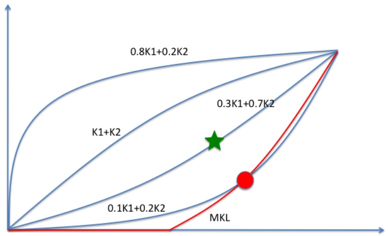
Multiclass Multiple Kernel Learning

Alexander Zien ALEXANDER.ZIEN@TUEBINGEN.MPG.DE
Cheng Soon Ong CHENGSOON.ONG@TUEBINGEN.MPG.DE
Max Planck Inst. for Biol. Cybernetics and Friedrich Miescher Lab., Spemannstr. 39, Tübingen, Germany.



MKL or sum kernel?

- Sum is simpler and works better to combine well-engineered kernels (eg, for data integration).
- In spite of its misleading name, MKL is better suited for kernel selection than for weight optimization (ℓ_2 vs ℓ_1). Useful to select among large sets of kernels.
- We would love to be able to select the "optimal" linear combination of a few kernels



Outline

Conclusion

- Are kernels popular and useful in bioinformatics?
  $\rightarrow$ YES
- Is kernel design useful?
  $\rightarrow$ YES, and we have many tricks for that
- Is kernels combination useful for performance?
  $\rightarrow$ YES, and the sum kernel does a good job
- Is MKL useful?
  $\rightarrow$ Hardly yet, but it offers the promising possibility to work with MANY kernels and emphasize INTERPRETABILITY
- Do we want to learn good linear combinations?
  $\rightarrow$ YES