Multiple Kernel Learning

(via Semi-Infinite Linear Programming)

Sören Sonnenburg
Fraunhofer FIRST.IDA, Berlin

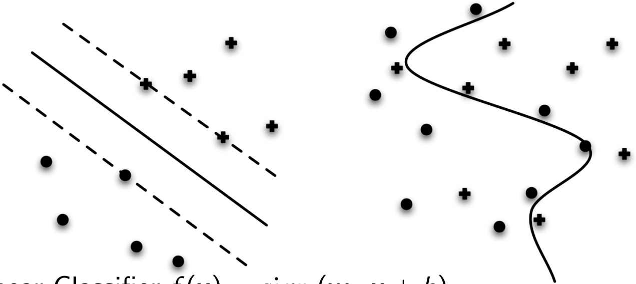
joint work with
Gunnar Rätsch, Christin Schäfer and Bernhard Schölkopf

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik FIRST

Outline

Classification

Given training examples $(\mathbf{x}_i, y_i)_{i=1}^N \in (\mathcal{X}, \{-1, +1\})^N$



- Linear Classifier $f(\mathbf{x}) = \mathrm{sign}\left(\mathbf{w} \cdot \mathbf{x} + b\right)$
- Kernel Machine (e.g. Support Vector Machine), learn weighting $\boldsymbol{\alpha} \in \mathbb{R}^N$ on training examples in kernel feature space $\Phi(\mathbf{x})$

$$f(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{N} y_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b\right),$$

where Kernel $k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$

Classification using Kernel Machines I

Single Kernel

- Kernel Machine (e.g. Support Vector Machine)
  - learn weighting $\boldsymbol{\alpha} \in \mathbb{R}^N$ on training examples $(\mathbf{x}_i, y_i)_{i=1}^N$ in kernel feature space $\Phi(\mathbf{x})$

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{N} y_i \alpha_i \mathrm{k}(\mathbf{x}, \mathbf{x}_i) + b\right)$$

  - where Kernel $k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$
  - still linear classifier $f(\mathbf{x}) = \operatorname{sign}(\mathbf{w} \cdot \mathbf{x} + b)$ in kernel feature space, with weighting $\mathbf{w} = \sum_{i=1}^{N} y_i \alpha_i \Phi(\mathbf{x}_i)$ and examples $\mathbf{x} \mapsto \Phi(\mathbf{x})$

via kernel: non-linear discrimination in input space

Classification using Kernel Machines II

Multiple Kernels

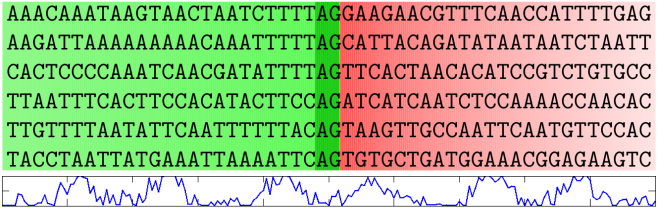
- Linear combination of kernels $\mathrm{k}(\mathbf{x},\mathbf{x}') = \sum_{j=1}^{M} \beta_j \, \mathrm{k}_j(\mathbf{x},\mathbf{x}')$, $\beta_j \geq 0$. Learn $\boldsymbol{\alpha}$ *and* $\boldsymbol{\beta}$. Resulting classifier:

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{j=1}^{M} \beta_j \sum_{i=1}^{N} y_i \alpha_i \mathrm{k}_j(\mathbf{x}, \mathbf{x}_i) + b \right)$$

Learn weighting over training examples α and kernels β

Combining Heterogeneous Data

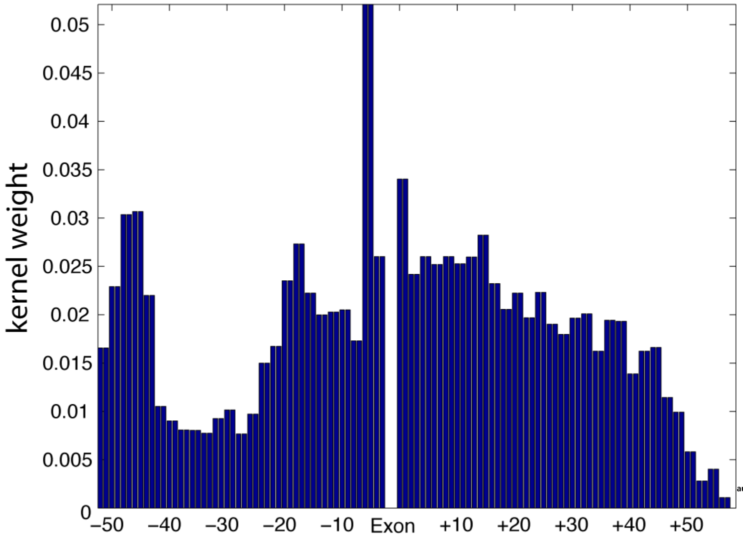
- Consider data from different domains: e.g Bioinformatics features: DNA-strings, binding energies, conservation, structure,...



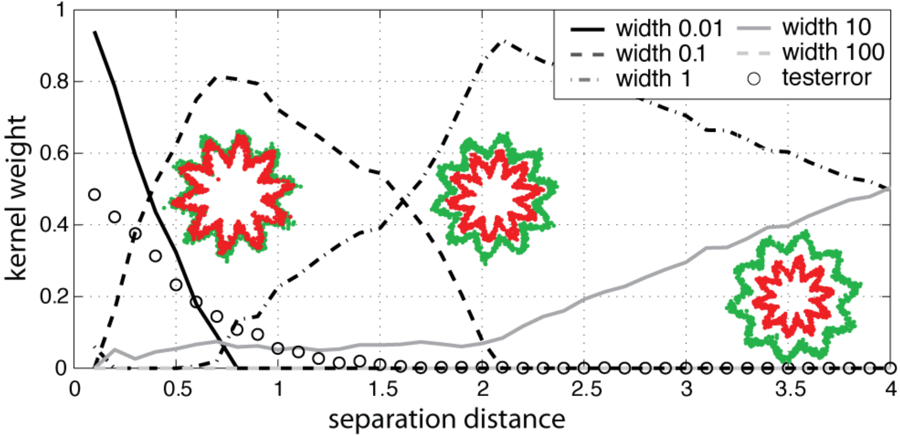
$$\color{red}{k(\mathbf{x}, \mathbf{x}')} =$$
$$\beta_1 \, k_{dna}(\mathbf{x}_{dna}, \mathbf{x}'_{dna}) + \beta_2 \, k_{nrg}(\mathbf{x}_{nrg}, \mathbf{x}'_{nrg}) + \beta_3 \, k_{3d}(\mathbf{x}_{3d}, \mathbf{x}'_{3d}) + \cdots$$

Interpretability

- Bioinformatics: One weight per position in sequence

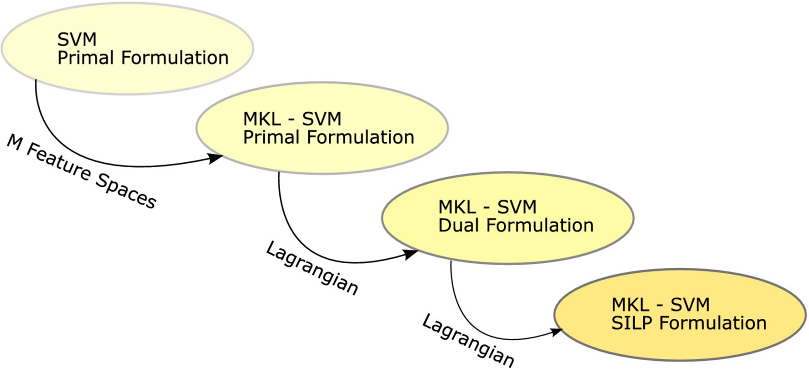


Automated Model Selection



Outline

Derivation



For details see Sonnenburg, Rätsch, Schäfer, Schölkopf 2006

SVM Primal Formulation

$$
\begin{aligned}
\text{min} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\xi_n \\
\text{w.r.t.} \quad & \mathbf{w} \in \mathbb{R}^D, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\
\text{s.t.} \quad & y_i\left(\mathbf{w}^\mathsf{T}\Phi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \forall i = 1, \ldots, N
\end{aligned}
$$

MKL Primal Formulation

$$\min \quad \frac{1}{2}\left(\sum_{j=1}^{M} \beta_j \|\mathbf{w}_j\|_2\right)^2 + C\sum_{i=1}^{N} \xi_n$$

$$\text{w.r.t.} \quad \mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_M),\ \mathbf{w}_j \in \mathbb{R}^{D_j}, \quad \forall j = 1 \ldots M$$

$$\boldsymbol{\beta} \in \mathbb{R}_+^M,\ \boldsymbol{\xi} \in \mathbb{R}_+^N,\ b \in \mathbb{R}$$

$$\text{s.t.} \quad y_i\left(\sum_{j=1}^{M} \beta_j {\mathbf{w}_j}^\top \Phi_j(\mathbf{x}_i) + b\right) \geq 1 - \xi_i,\ \forall i = 1, \ldots, N$$

$$\sum_{j=1}^{M} \beta_j = 1$$

Properties: equivalent to SVM for $M = 1$; solution sparse in "blocks"; each block j corresponds to one kernel

MKL Dual Formulation

Bach, Lanckriet, Jordan 2004:

$$\begin{aligned}
\text{min} \quad & \gamma - \sum_{i=1}^{N} \alpha_i \\
\text{w.r.t.} \quad & \gamma \in \mathbb{R}, \boldsymbol{\alpha} \in \mathbb{R}^N \\
\text{s.t.} \quad & 0 \leq \boldsymbol{\alpha} \leq C, \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& \frac{1}{2} \sum_{r=1}^{N} \sum_{s=1}^{N} \alpha_r \alpha_s y_r y_s K_j(\mathbf{x}_r, \mathbf{x}_s) - \gamma \leq 0, \ \forall j = 1, \ldots, M
\end{aligned}$$

Properties: equivalent to SVM for $M = 1$

The Semi-Infinite Linear Program I

$$
\begin{aligned}
\max \quad & \theta \\
\text{w.r.t.} \quad & \theta \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}_+^M \text{ with } \sum_{j=1}^{M} \beta_j = 1 \\
\text{s.t.} \quad & \sum_{j=1}^{M} \beta_j \left(\underbrace{\frac{1}{2} \sum_{r=1}^{N} \sum_{s=1}^{N} \alpha_r \alpha_s y_r y_s K_j(\mathbf{x}_r, \mathbf{x}_s)}_{=: S_j(\boldsymbol{\alpha})} - \sum_{i=1}^{N} \alpha_i \right) \geq \theta \\
& \text{for all } \boldsymbol{\alpha} \text{ with } 0 \leq \boldsymbol{\alpha} \leq C \text{ and } \sum_{i=1}^{N} y_i \alpha_i = 0
\end{aligned}
$$

Properties:

- linear, optimize over a convex combination of $\boldsymbol{\beta}$
- infinitely many constraints, one for each $0 \leq \boldsymbol{\alpha} \leq C$
- can use standard SVM to identify most violated constraints

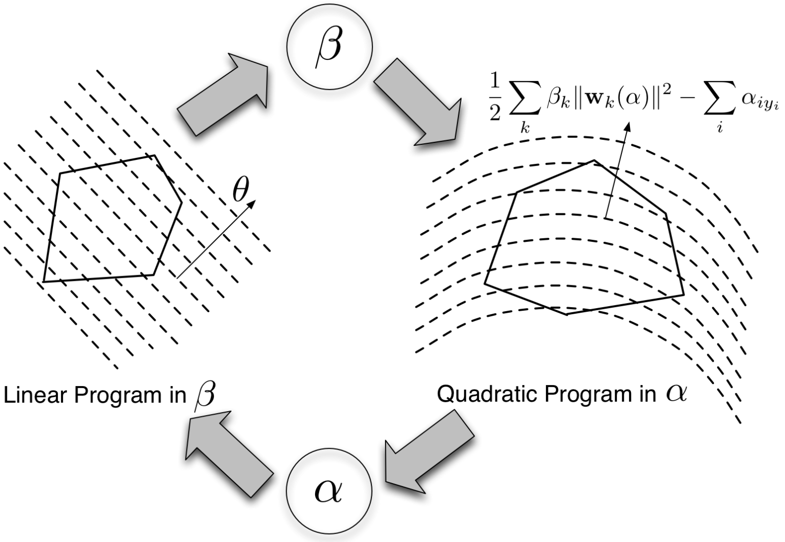
The Semi-Infinite Linear Program II

$$\begin{array}{rl} \max & \theta \\ \text{w.r.t.} & \theta \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}^M_+ \text{ with } \sum_{j=1}^{M} \beta_j = 1 \\ \text{s.t.} & \sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right) \geq \theta \\ & \text{for all } \boldsymbol{\alpha} \text{ with } 0 \leq \boldsymbol{\alpha} \leq C \text{ and } \sum_{i=1}^{N} y_i \alpha_i = 0 \end{array}$$

Solving the SILP:

- Column Generation
  - fast, but no known convergence rate
- Use Boosting like techniques: Arc-GV or AdaBoost*
  - known convergence rate $\mathcal{O}(\log(M)/\varepsilon^2)$
- Chunking like algorithm
  - consider suboptimal SVM solutions: empirically 3-5 times faster

Solving the SILP: Column Generation I



Solving the SILP: Column Generation II

$$
\begin{aligned}
\max \quad & \theta \\
\text{w.r.t.} \quad & \theta \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}_+^M \text{ with } \sum_{j=1}^{M} \beta_j = 1 \\
\text{s.t.} \quad & \sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right) \geq \theta \\
& \text{for all } \boldsymbol{\alpha} \text{ with } 0 \leq \boldsymbol{\alpha} \leq C \text{ and } \sum_{i=1}^{N} y_i \alpha_i = 0
\end{aligned}
$$

- iteratively find most violated constraints, solve linear program with current constraints, ..., till convergence to the global optimum

$$
\sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right) = \frac{1}{2} \sum_{r=1}^{N} \sum_{s=1}^{N} \alpha_r \alpha_s y_r y_s \sum_{j=1}^{M} \beta_j k_j(\mathbf{x}_r, \mathbf{x}_s) - \sum_{i=1}^{N} \alpha_i,
$$

- solved by taking set of most violated constraints into account
- most violated constraints given by SVM solution for fixed $\boldsymbol{\beta}$

Outline

1. Introduction and Motivation
   - When is Multiple Kernel Learning useful?
2. Multiple Kernel Learning
   - Deriving the Semi-Infinite Linear Program
   - MKL SILP Algorithm
3. Applications, Extensions, Outlook
   - Extensions
   - Applications
   - Outlook

Regression

Primal Formulation:

$$\min \quad \frac{1}{2}\left(\sum_{j=1}^{M} \beta_j \|\mathbf{w}_j\|\right)^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

$$\text{w.r.t.} \quad \mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_M), \mathbf{w}_j \in \mathbb{R}^{D_j}, \quad \forall j = 1 \ldots M$$

$$\boldsymbol{\beta} \in \mathbb{R}_+^M, \ \boldsymbol{\xi} \in \mathbb{R}^N, \ \boldsymbol{\xi}^* \in \mathbb{R}_+^N, \ b \in \mathbb{R}$$

$$\text{s.t.} \quad \sum_{j=1}^{M} \beta_j \mathbf{w}_j^{\mathsf{T}} \Phi_j(\mathbf{x}_i) + b \leq y_i + \varepsilon + \xi_i, \ \forall i = 1 \ldots N$$

$$\sum_{j=1}^{M} \beta_j \mathbf{w}_j^{\mathsf{T}} \Phi_j(x_i) + b \geq y_i - \varepsilon - \xi_i^*, \ \forall i = 1 \ldots N$$

$$\sum_{j=1}^{M} \beta_j = 1$$

One Class

Primal Formulation:

$$\min \quad \frac{1}{2}\left(\sum_{j=1}^{M} \beta_j \|\mathbf{w}_j\|_2\right)^2 + C\sum_{i=1}^{N}\xi_i - \rho$$

$$\text{w.r.t.} \quad \mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_M), \mathbf{w}_j \in \mathbb{R}^{D_j}, \quad \forall j = 1 \ldots M$$

$$\boldsymbol{\beta} \in \mathbb{R}_+^M, \ \boldsymbol{\xi} \in \mathbb{R}_+^N$$

$$\text{s.t.} \quad y_i \left(\sum_{j=1}^{M} \beta_j \mathbf{w}_j^{\top} \Phi_j(\mathbf{x}_i)\right) \geq \rho - \xi_i, \forall i = 1, \ldots, N$$

$$\sum_{j=1}^{M} \beta_j = 1$$

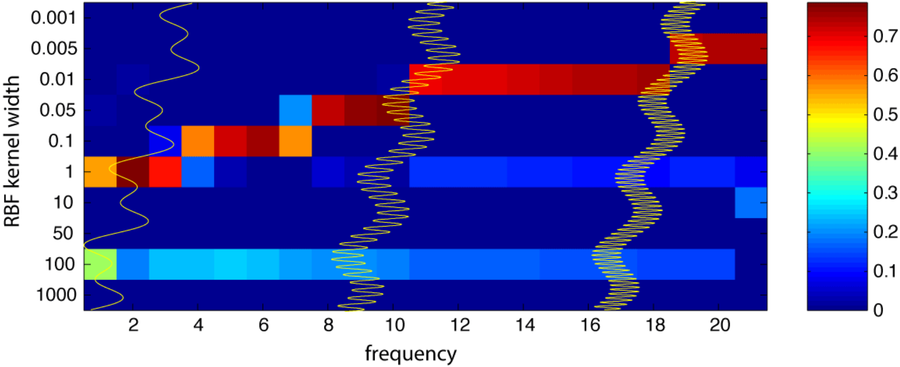
Generalized for arbitrary strictly convex differentiable loss functions (Sonnenburg, Rätsch, Schäfer, Schölkopf 2006)

Multiclass

Primal Formulation (Zien, Ong 2007):

$$
\begin{aligned}
\min \quad & \frac{1}{2}\left(\sum_{j=1}^{M} \beta_j \|\mathbf{w}_j\|_2\right)^2 + C\sum_{i=1}^{N} \xi_n \\
\text{w.r.t.} \quad & \mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_M),\ \mathbf{w}_j \in \mathbb{R}^{k_j}, \quad \forall j = 1 \ldots M \\
& \boldsymbol{\beta} \in \mathbb{R}_+^M,\ \mathbf{s} \in \mathbb{R}^{N \times c},\ \boldsymbol{\xi} \in \mathbb{R}_+^N,\ b \in \mathbb{R} \\
\text{s.t.} \quad & \xi_i = \max_{u \neq y_i} s_{iu},\ s_{iu} \geq 0, \\
& \sum_{j=1}^{M} \beta_j {\mathbf{w}_j}^{\top} \left(\Phi_j(\mathbf{x}_i, y_i) - \Phi_j(\mathbf{x}_i, u)\right) + b_{y_i} - b_u \geq 1 - s_{iu}, \\
& \forall i = 1 \ldots N,\ \forall u = 1 \ldots c
\end{aligned}
$$

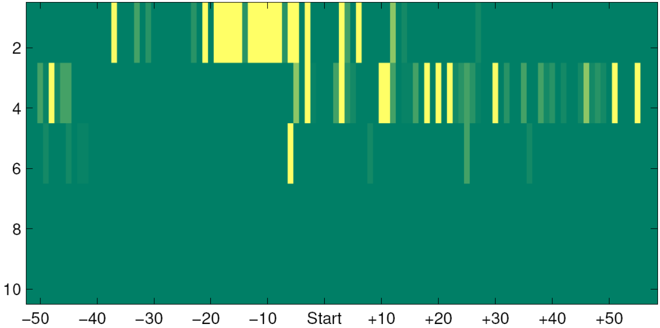
Automated Model Selection - Regression



- $f(x) = \sin(ax) + \sin(bx) + cx$ for varying a
- Support Vector Regression with 10 RBF-Kernels of different width

Knowledge discovery

Feature Extraction



- Support Vector Classification on Bioinformatics problem, distinguish “splice sites” form “fake sites” (aligned DNA sequences)
- One weight β_j per position and per sub-sequence length
- Displayed: Learned weights of 500 kernels

Summary and Outlook

MKL learns convex combination of kernels

$\Rightarrow$ allows (to some extend) for automated model selection

$\Rightarrow$ allows for interpreting SVM result

$\Rightarrow$ matches prior knowledge on real-world bioinformatics problem

- **Simple:** iterative wrapper algorithm around single kernel SVM
- **General:** same technique applicable to a wide range of problems (1-class, 2-class, Multiclass, Regression, ...)
- **Fast:** suitable for large scale problems ($> 100,000$ examples)

Download free source `http://www.shogun-toolbox.org`**.**