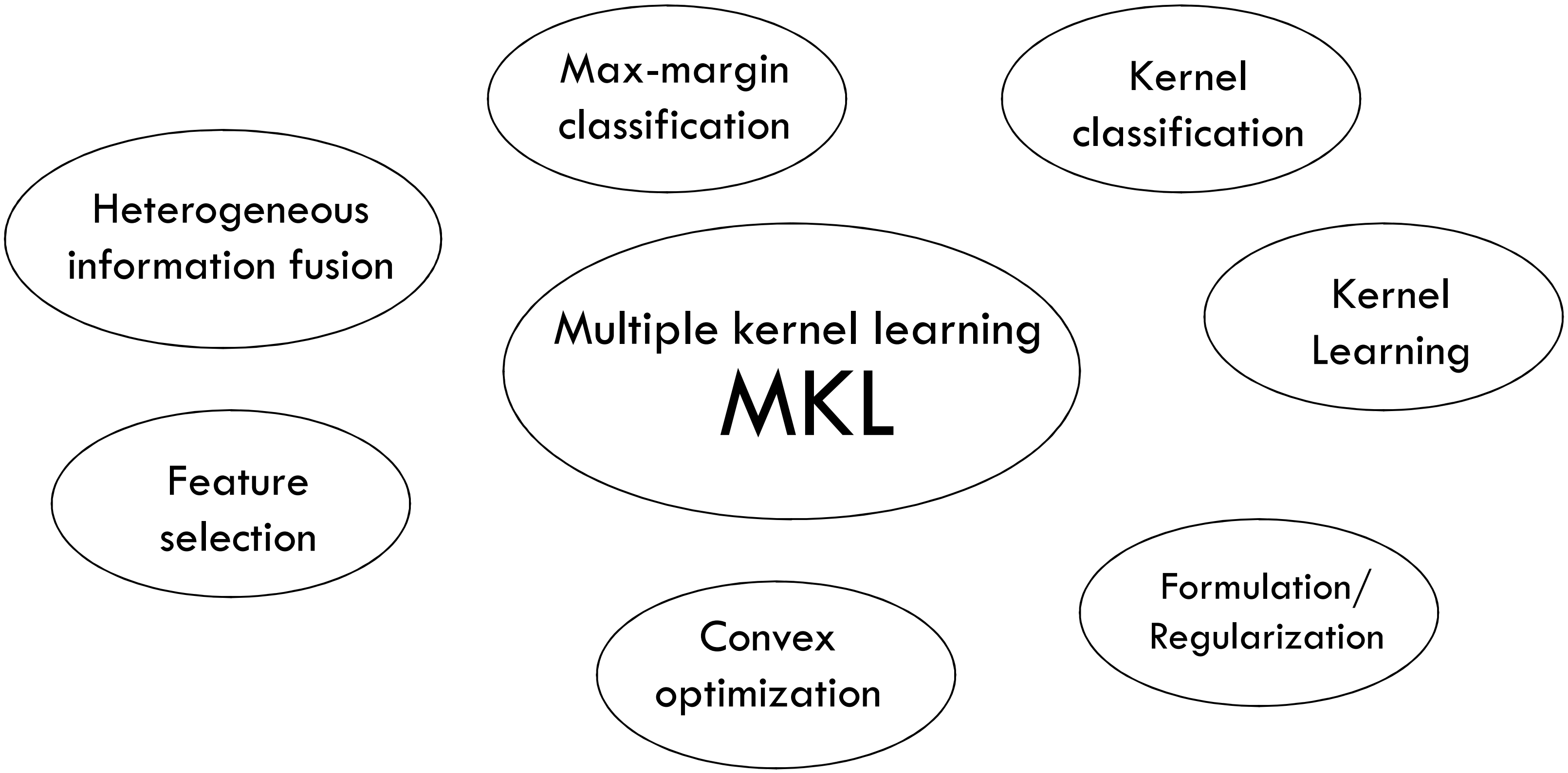
MULTIPLE KERNEL LEARNING

CSE902

Multiple Kernel Learning - keywords



MKL is used when there are heterogeneous sources (representations) of data for the task at hand (we consider classification)

Outline

1. ***Kernel Combination***
   1. ***Heterogeneous Information Fusion***
   2. ***Feature Selection***
2. Linear margin classifiers (SVM)
3. Kernel Classifiers
4. Kernel Learning / Multiple Kernel Learning
   1. MKL Formulation
   2. Sparse vs. Smooth MKL
   3. MKL Optimization
5. Experimental Results from Literature
6. Our Experiments
7. Conclusions

Heterogeneous Information Fusion

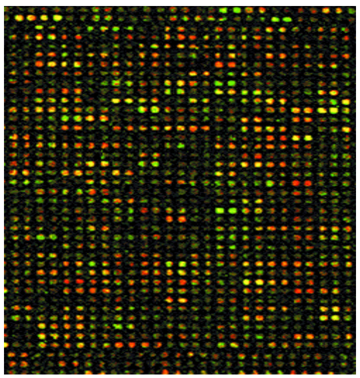
Web page categorization

- Data point = web page
- Sources of information about the webpage:
  - Content:
    - Text
    - Images
    - Structure
    - Sound
  - Relation to other web pages: links → network
  - Users (log data):
    - click behavior
    - origin

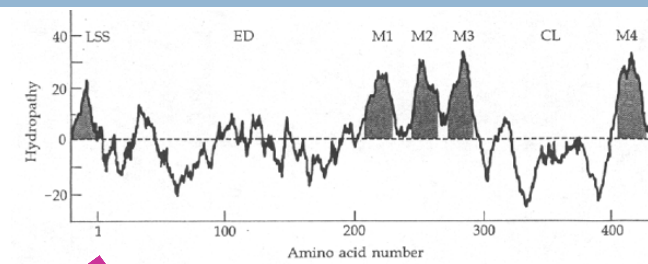
Slide: Lanckriet et al.

Heterogeneous Information Fusion

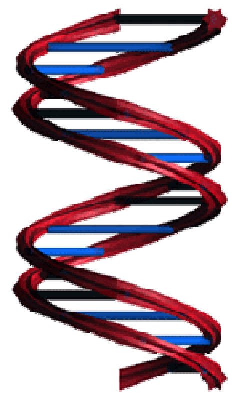
- Bioinformatics



mRNA
expression data

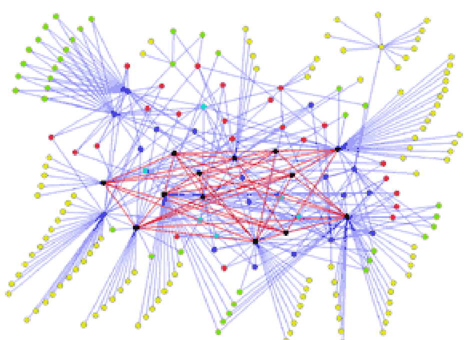


hydrophobicity data

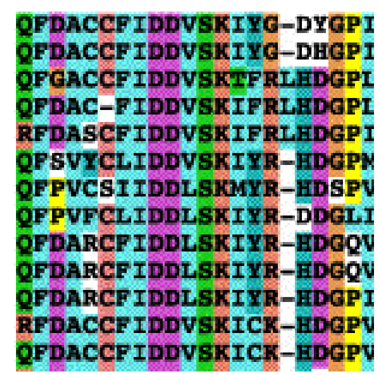


protein-protein
interaction data

sequence data
(gene, protein)



upstream region data
(TF binding sites)



Lanckriet et al.

Feature Selection

Text mining

- Newsgroups data
  - Unstructured text
  - 1.3M Bag-of-Words features
- Log data for suspicious links
  - Structured text
  - 3.2M constant and variable features
- Real Time processing
- Using small number of features for prediction speed

Tsang et al.

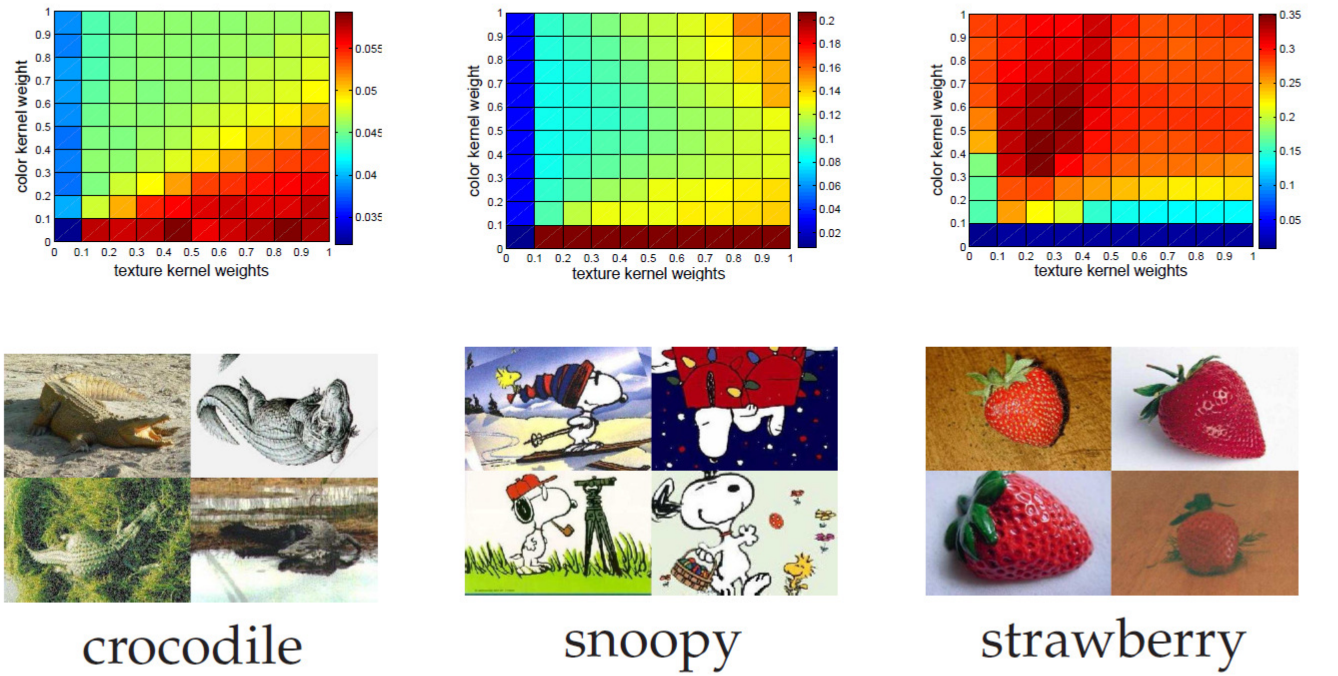
Feature Selection

Image categorization and retrieval

- Hundreds of feature types (SIFT, HOG, GIST)
- Select and combine features for improved prediction accuracy and speed

Tsang et al.

Image Categorization: A Small Experiment



- First row: Surface graphs (influence of different feature combination weights on MAP)
- Second row: 4 examples from each class

Questions

- How to merge different information sources (images, sounds, click data, user feedback ...)
  - Same format (numerical), different scale
  - Different formats (ordinal, categorical, non-vectorial)
- How to eliminate irrelevant or redundant data
- How to combine the feature vectors (feature weighting)
- How to minimize the number of sources to improve computational efficiency (sparsity)

MKL vs Alternatives

- MKL is proposed as an alternative to:
  - Cross validation
  - Feature Selection
  - Metric learning
  - Ensemble methods

MKL vs Alternatives

- MKL is proposed as an alternative to:
  - Cross validation
  - Feature Selection
  - Metric learning
  - Ensemble methods
- Advantages of MKL
  - A technically sound way of combining features
  - Feature combination and classifier training is done simultaneously
  - Good learning bounds
  - Different data formats can be used in the same formulation
  - Non-linear learning

Outline

1. Kernel Combination
   1. Heterogeneous Information Fusion
   2. Feature Selection
2. ***Linear margin classifiers (SVM)***
3. Kernel Classifiers
4. Kernel Learning / Multiple Kernel Learning
   1. MKL Formulation
   2. Sparse vs. Smooth MKL
   3. MKL Optimization
5. Experimental Results from Literature
6. Our Experiments
7. Conclusions

Binary Classification

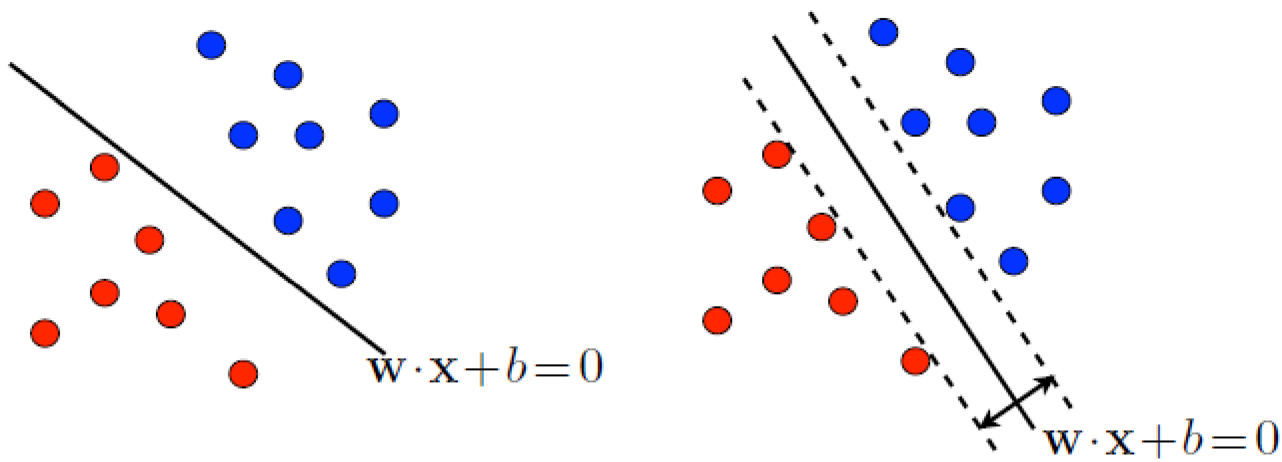
- **Training data**: sample drawn i.i.d. from set $X \subseteq \mathbb{R}^N$ according to some distribution D,

$$S = ((x_1, y_1), \ldots, (x_m, y_m)) \in X \times \{-1, +1\}.$$

- **Problem**: find hypothesis $h : X \mapsto \{-1, +1\}$ in H (classifier) with small generalization error $R_D(h)$.
- **Linear classification**:
  - Hypotheses based on hyperplanes.
  - Linear separation in high-dimensional space.

Cortes et al

Linear Separation

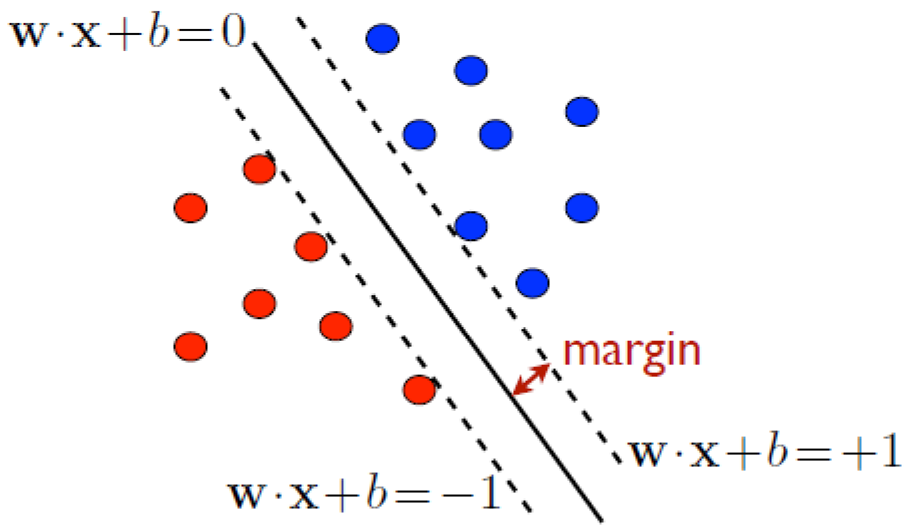


- Classifiers: $H=\{\mathbf{x}\mapsto \mathrm{sgn}(\mathbf{w}\cdot\mathbf{x}+b):\mathbf{w}\in\mathbb{R}^N, b\in\mathbb{R}\}$.

Why is it important to maximize the margin?

Cortes et al.

Margin Classifier



- Canonical hyperplane: $\mathbf{w}$ and b chosen such that for closest points $|\mathbf{w}\cdot\mathbf{x}+b|=1$.
- Margin: $\rho=\min_{\mathbf{x}\in S}\frac{|\mathbf{w}\cdot\mathbf{x}+b|}{\|\mathbf{w}\|}=\frac{1}{\|\mathbf{w}\|}$.

Cortes et al.

Computing the Classifier

- Constrained optimization:

$$\min_{\mathbf{w},b} \; \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to } \; y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, i \in [1, m].$$

- Properties:
  - Convex optimization (strictly convex).
  - Unique solution for linearly separable sample.

Cortes et al.

Computing the Classifier

- Constrained optimization:

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to} \ \ y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, i \in [1, m].$$

- Properties:
  - Convex optimization (strictly convex).
  - Unique solution for linearly separable sample.

What to do when the data is not linearly separable?

Cortes et al.

Computing the Classifier: for Linearly Non-separable Data

- Constrained optimization:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \boxed{\xi_i}$$

Slack variable

$$\text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \ \wedge \ \xi_i \geq 0, i \in [1, m].$$

- Properties:
  - $C \geq 0$ trade-off parameter.
  - Convex optimization (strictly convex).
  - Unique solution.

Cortes et al.

Primal Optimization Problem

$$\min_{\mathbf{w},b,\xi} \boxed{\frac{1}{2}\|\mathbf{w}\|^2} + C \boxed{\sum_{i=1}^{m} \xi_i}$$

$$\boxed{\text{subject to } \; y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \;\wedge\; \xi_i \geq 0, i \in [1, m].}$$

$$\text{Obj. Function} = \overbrace{\text{Regularizer} + \text{error term}} + \text{constraints}$$

- Regularizer: maximize the margin
- Error term: minimize the number of errors
- C: trade-of between overfitting and underfitting

Dual Optimization Problem

- Use Lagrangian to construct the dual problem
  - Constrained optimization:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subject to: } \alpha_i \geq 0 \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, i \in [1, m].$$

  - Solution:

$$h(x) = \operatorname{sgn}\Big(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b \Big),$$

with $b = y_i - \sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i)$ for any SV $\mathbf{x}_i$.

Cortes et al.

Dual Optimization Problem

- Use Lagrangian to construct the dual problem
  - Constrained optimization:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subject to: } \alpha_i \geq 0 \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, i \in [1, m].$$

  - Solution:

$$h(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b\right),$$

with $b = y_i - \sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i)$ for any SV $\mathbf{x}_i$.

Which one to use: Dual or Primal?

Cortes et al.

Dual Optimization Problem

- Use Lagrangian to construct the dual problem
  - Constrained optimization:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subject to: } \alpha_i \geq 0 \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, i \in [1, m].$$

  - Solution:

$$h(x) = \operatorname{sgn}\Big(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b \Big),$$

with $b = y_i - \sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i)$ for any SV $\mathbf{x}_i$.

What to do when linear classifiers do not work?

Cortes et al.

Outline

1. Kernel Combination
   1. Heterogeneous Information Fusion
   2. Feature Selection
2. Linear margin classifiers (SVM)
3. ***Kernel Classifiers***
4. Kernel Learning / Multiple Kernel Learning
   1. MKL Formulation
   2. Sparse vs. Smooth MKL
   3. MKL Optimization
5. Experimental Results from Literature
6. Our Experiments
7. Conclusions

Kernel Classifiers

- Idea:
  - Define $K : X \times X \rightarrow \mathbb{R}$, called kernel, such that:

$$\Phi(x) \cdot \Phi(y) = K(x, y).$$

  - K often interpreted as a similarity measure.
- Benefits:
  - Efficiency: K is often more efficient to compute than Φ and the dot product.
  - Flexibility: K can be chosen arbitrarily so long as the existence of Φ is guaranteed (Mercer's condition).

Cortes et al.

Kernel Trick: An Example

- Polynomial kernels
  - Definition:

$$\forall x, y \in \mathbb{R}^N,\ K(x, y) = (x \cdot y + c)^d, \quad c > 0.$$

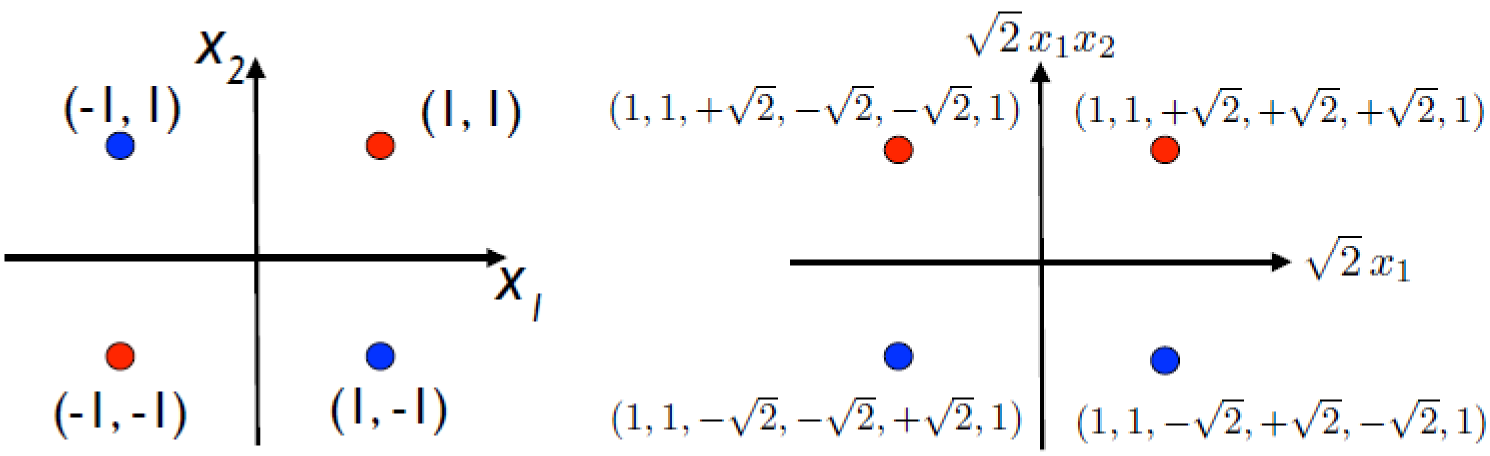
  - Example: for $N = 2$ and $d = 2$,

$$K(x, y) = (x_1 y_1 + x_2 y_2 + c)^2$$
$$= \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \\ \sqrt{2c}\, x_1 \\ \sqrt{2c}\, x_2 \\ c \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}\, y_1 y_2 \\ \sqrt{2c}\, y_1 \\ \sqrt{2c}\, y_2 \\ c \end{bmatrix}.$$

Cortes et al.

XOR Problem: An example

- Use second-degree polynomial kernel with $c = 1$:



Linearly non-separable

Linearly separable by $x_1 x_2 = 0$.

- Data is linearly separable in the transformed feature space (polynomial kernel)

Cortes et al.

Dual Problem with Kernel Function

- Constrained optimization:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{subject to: } 0 \leq \alpha_i \leq C \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, i \in [1, m].$$

- Solution:

$$h(x) = \text{sgn}\Big(\sum_{i=1}^{m} \alpha_i y_i K(x_i, x) + b\Big),$$

with $b = y_i - \sum_{j=1}^{m} \alpha_j y_j K(x_j, x_i)$ for any x_i with $0 < \alpha_i < C$.

Cortes et al.

Dual Problem with Kernel Function

- Constrained optimization:

Inner product replaced by the kernel function

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \boxed{K(x_i, x_j)}$$

$$\text{subject to: } 0 \leq \alpha_i \leq C \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, i \in [1, m].$$

- Solution:

Decision function

$$\boxed{h(x) = \text{sgn}\Big(\sum_{i=1}^{m} \alpha_i y_i K(x_i, x) + b\Big),}$$

with $b = y_i - \sum_{j=1}^{m} \alpha_j y_j K(x_j, x_i)$ for any x_i with $0 < \alpha_i < C.$

Cortes et al.

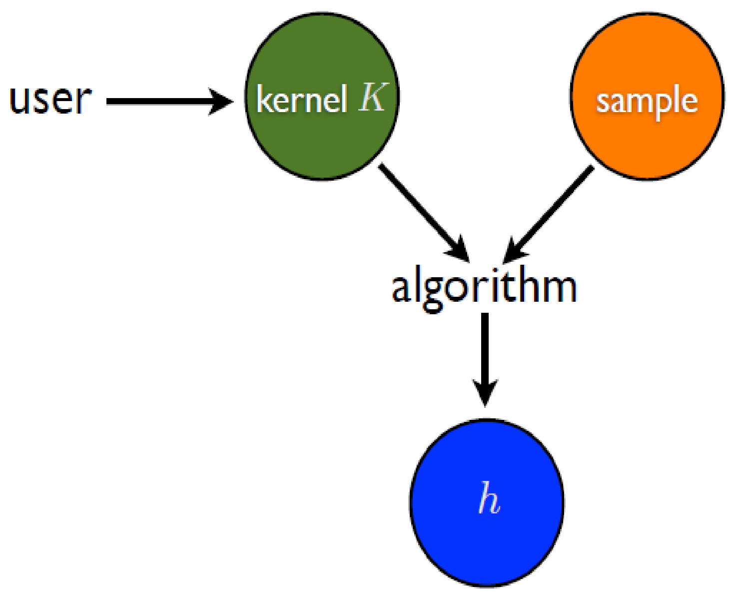
Kernel-based Learning

- The inner product in the feature space (similarity score) is performed implicitly
- Any linear classification method can be extended to nonlinear feature space
- Non-vectorial data can be utilized (as long as kernel matrix is PSD)
- Which kernel to use? How to set the parameters?
- One kernel for each feature type or for all?

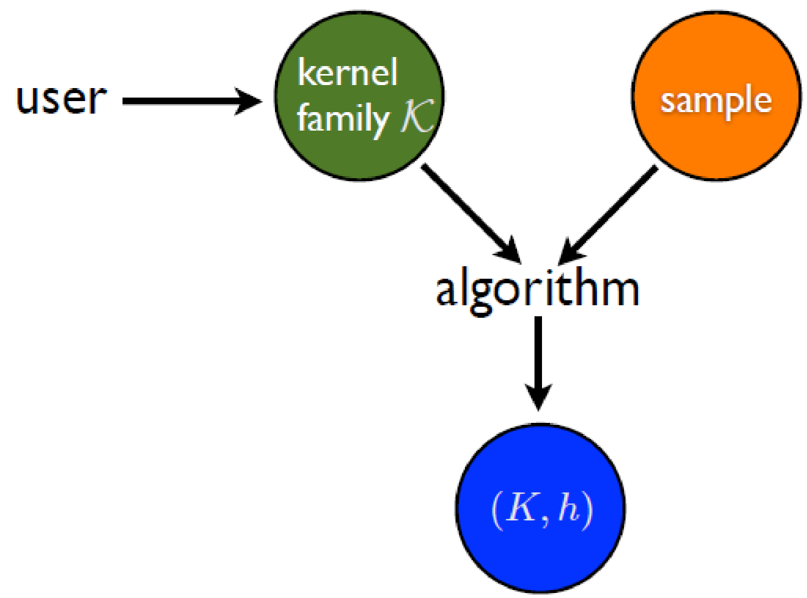
Outline

1. Kernel Combination
    1. Heterogeneous Information Fusion
    2. Feature Selection
2. Linear margin classifiers (SVM)
3. Kernel Classifiers
4. ***Kernel Learning / Multiple Kernel Learning***
    1. ***MKL Formulation***
    2. ***Sparse vs. Smooth MKL***
    3. ***MKL Optimization***
5. Experimental Results from Literature
6. Our Experiments
7. Conclusions

Kernel Learning



standard kernel classifier framework



kernel learning framework

Cortes et al.

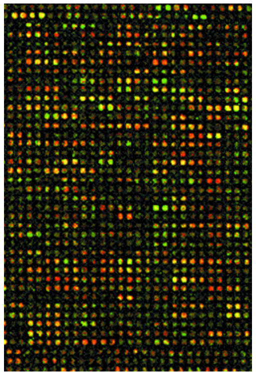
Multiple Kernel Learning (MKL)

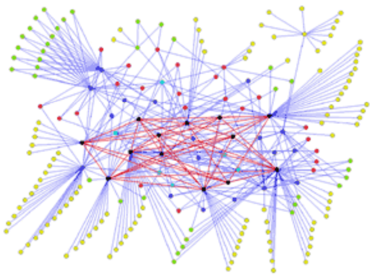
- Multiple kernel learning is a (**parametric**) kernel learning method.

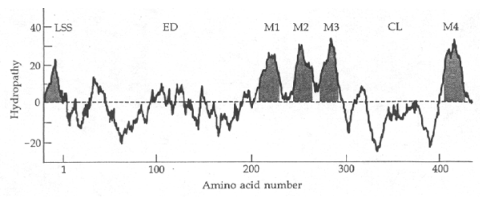
- Some related approaches:
  - Feature selection
  - Metric Learning
  - Ensemble methods

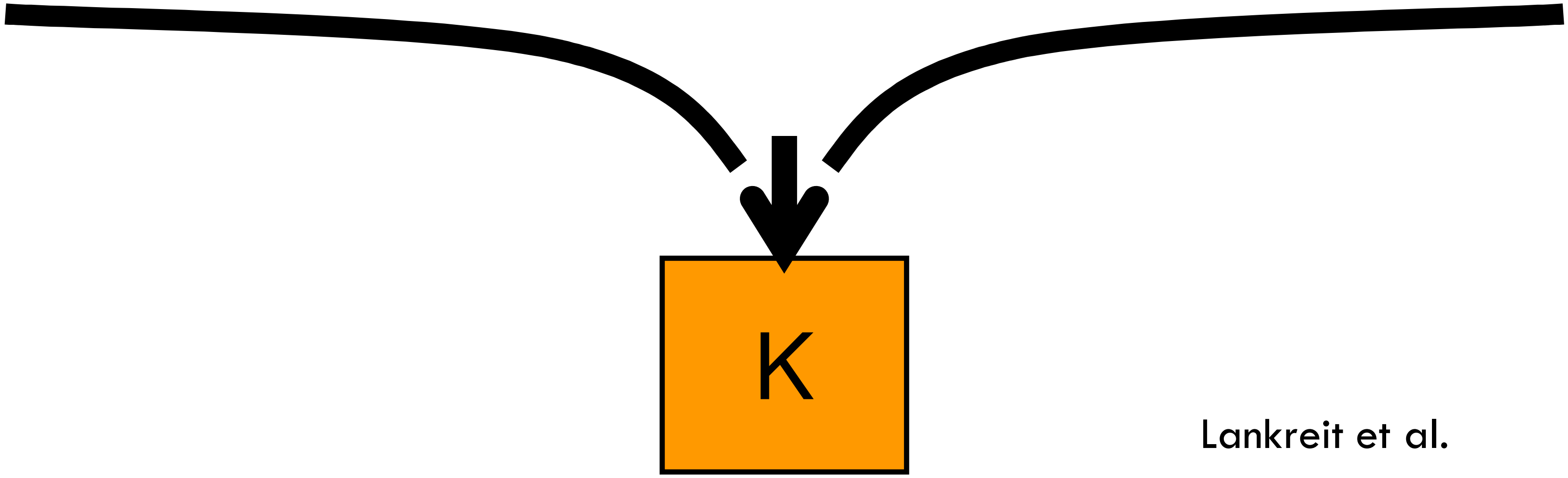
Kernel Learning Framework

- Example: Protein classification with using heterogeneous information sources









Lankreit et al.

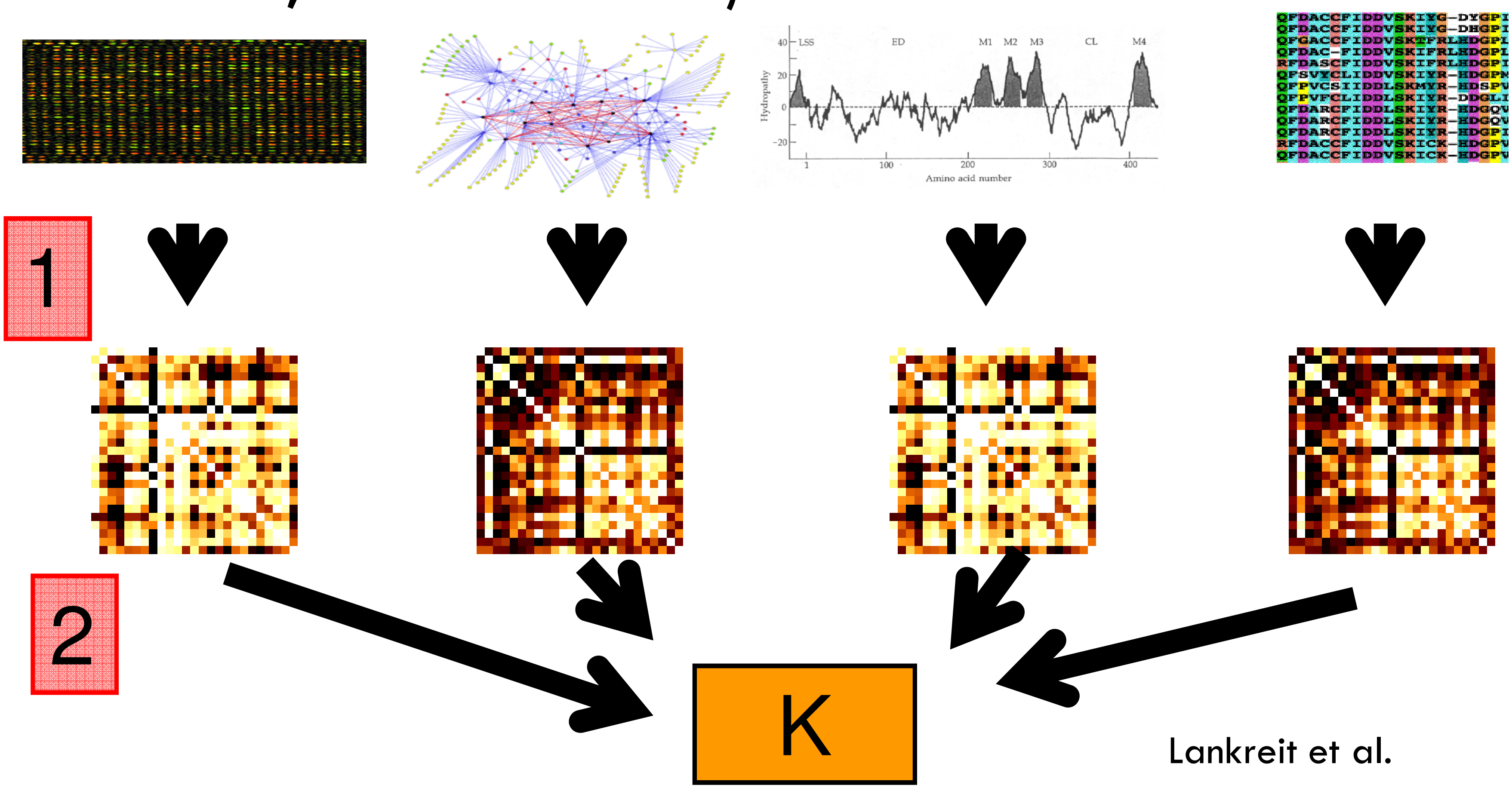
Multiple Kernel Learning

- The overall MKL framework:

1. Extract features from all available sources
2. Construct kernel matrices
    1. Different features
    2. Different kernel types
    3. Different kernel parameters
3. Find the optimal kernel combination and the kernel classifier

Kernel Learning Framework

- Create individual kernels for each source (string kernel, diffusion kernel)



Lankreit et al.

Multiple Kernel Learning

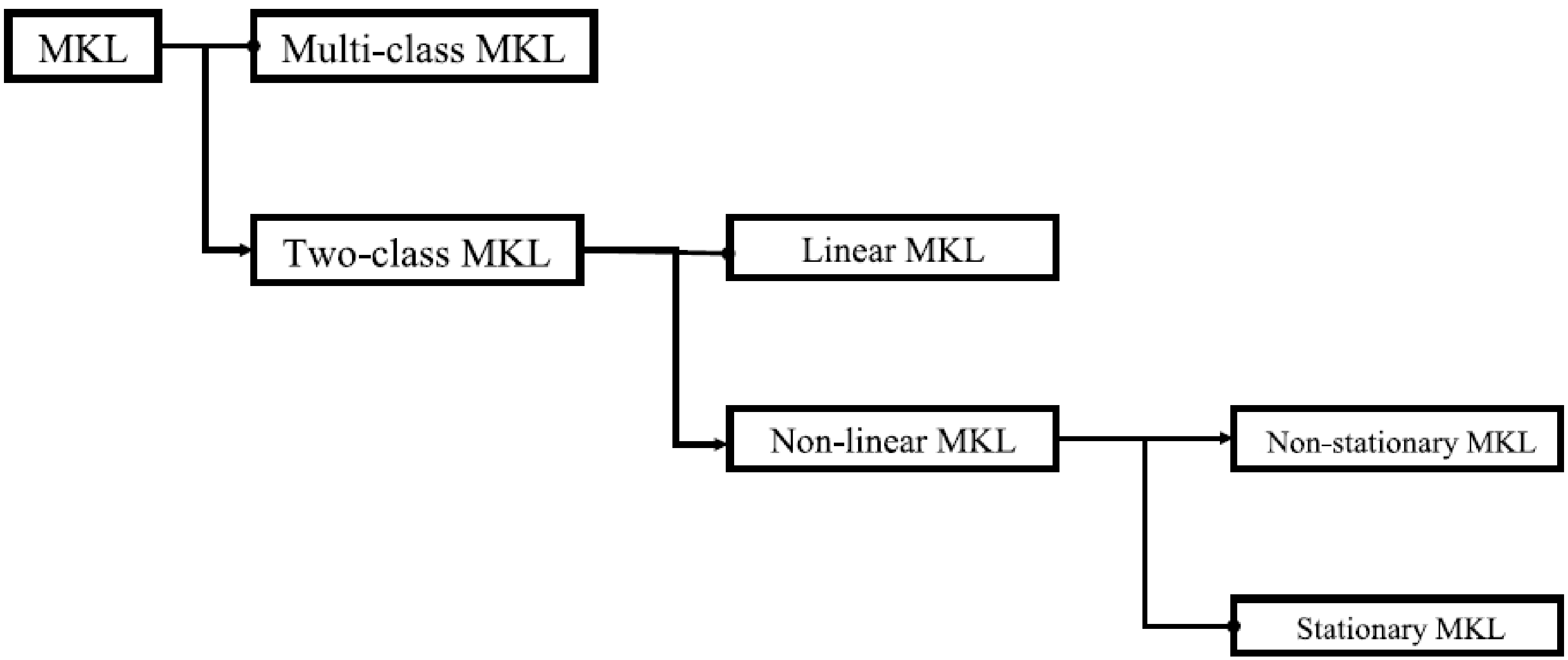
- The overall MKL framework:

1. Extract features from all available sources
2. Construct kernel matrices
    1. Different features
    2. Different kernel types
    3. Different kernel parameters
3. Find the optimal kernel combination and the kernel classifier

How is the optimal combination computed?

Multiple Kernel Learning

- Multiple kernel learning is a **parametric** kernel learning method:



- Our focus is Two-class (binary) MKL

Linear MKL

- Dual MKL formulation

$\circ$: Hadamard product

$$\boxed{\min_{\beta \in \Delta}} \max_{\alpha \in \mathcal{Q}} \widehat{\mathcal{L}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha} \circ \mathbf{y})^\top \boxed{\mathbf{K}(\boldsymbol{\beta})} (\boldsymbol{\alpha} \circ \mathbf{y}),$$

$$\mathbf{K}(\boldsymbol{\beta}) \quad = \sum_{j=1}^{s} \beta_j \mathbf{K}_j$$

- Similar to SVM dual problem, **BUT**:
  - Minimization w.r.t. the kernel coefficient vector β
  - Constraint on the kernel coefficient β
  - (Linearly) Combined kernel instead of a single one.

Regularization on Coefficient Vector

- Regularization:
  - Leads to well-posed problem
  - Affects the level of *sparsity*
  - Can lead to *smooth* objective function (or not)

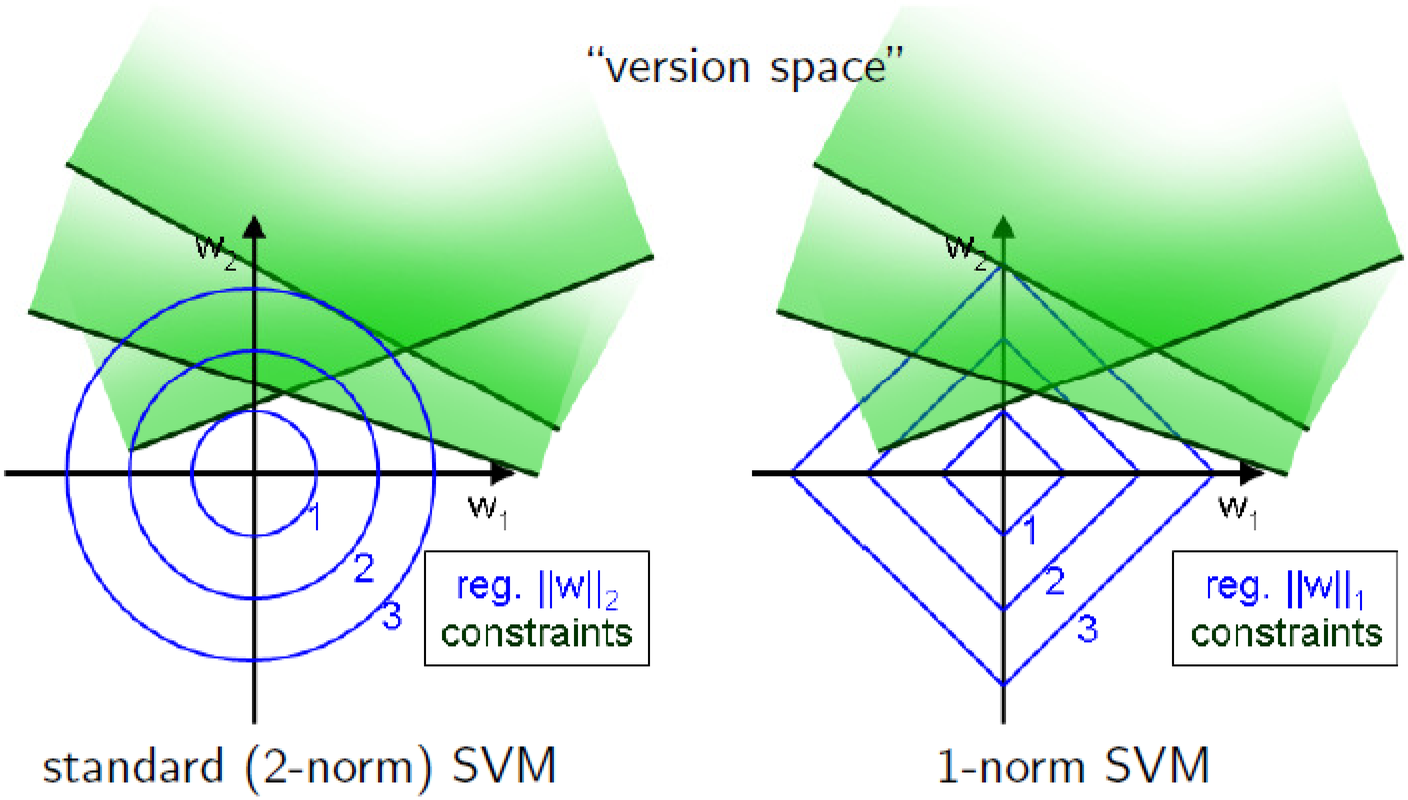
$$\boxed{\Delta_1 = \left\{ \boldsymbol{\beta} \in \mathbb{R}_+^s : \|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{s} |\beta^j| \leq 1 \right\}}.$$

L1-MKL
SPARSE solutions

$$\boxed{\Delta_p = \left\{ \boldsymbol{\beta} \in \mathbb{R}_+^s : ||\boldsymbol{\beta}||_p \leq 1 \right\}}.$$

Lp-MKL
DENSE solutions

Why Does L1-norm Produces Sparse Solutions?

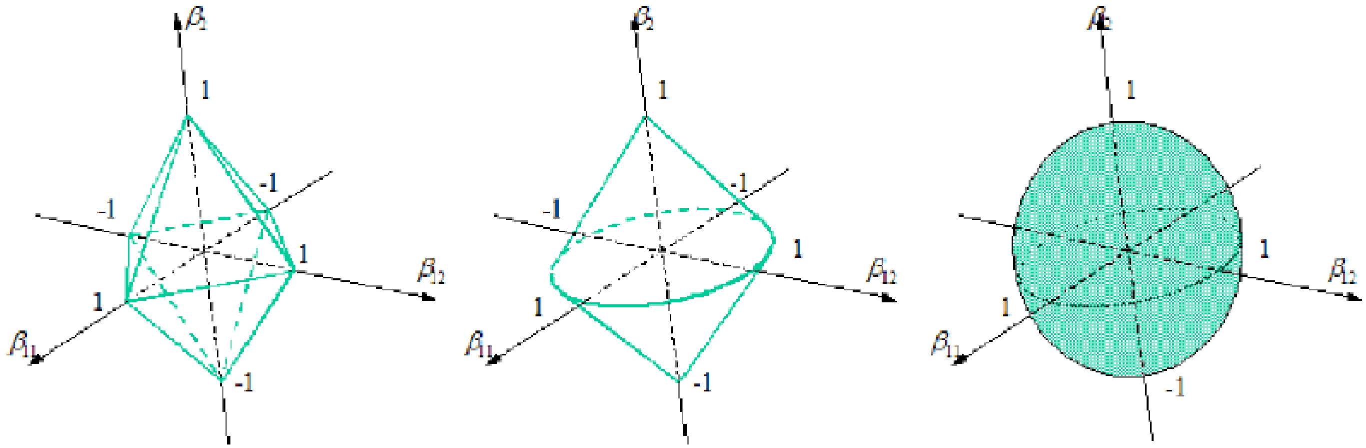


- Feasible solution meets the regularizers at corners

Zien et al.

MKL and Group Lasso

- L1-MKL is equivalent to feature selection with Group-Lasso (each kernel is a group of features)



1-norm SVM, lasso:
1-norm-constraints on all individual features

standard MKL:

- 1-norm-constraints *between* groups (ie kernels)
- 2-norm-constraints *within* feature groups

standard SVM, ridge-regression:
2-norm-constraints on all features

image from M. Yuan, Y. Lin; Journal of the Royal Statistical Society 2006]   Zien et al.

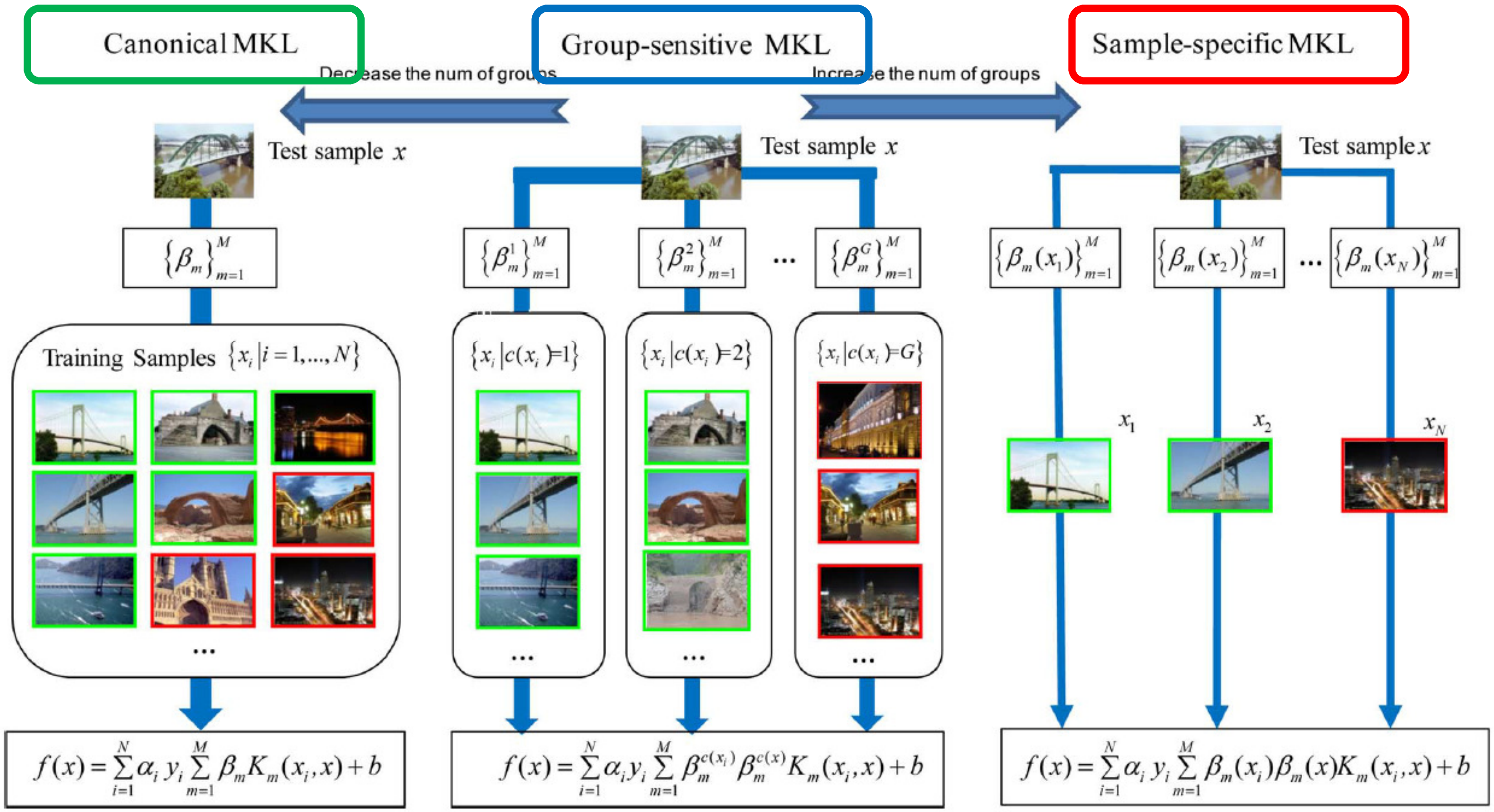
Sparse vs. Dense MKL

- Which one is better?
  - Prediction performance
  - Computational efficiency in training
  - Computational efficiency in prediction

Non-linear MKL

- 2 categories:
  - Stationary: Learn one (and the same) kernel combination for all training samples
  - Non-stationary: Each sample (or group of samples) uses a different kernel combination

Non-stationary MKL



(Left) Canonical MKL: one kernel combination for all; (Middle) Group MKL: one kernel combination for each group; (Right) Sample-MKL: one kernel combination for each sample

Yang et al. 2009

Stationary Non-linear MKL

- Generalized MKL (Varma & Babu 2009)

  Gaussians: $\mathbf{K}_{\boldsymbol{\mu}}(x_i, x_j) = \prod_{k=1}^{p} \exp\big(- \mu_k (x_{ik} - x_{jk})^2 \big)$

- Polynomial kernels (Cortes et al. 2009)

  Polynomials: $\mathbf{K}_{\boldsymbol{\mu},d}(x_i, x_j) = (1 + \sum_{k=1}^{p} \mu_k x_{ik} x_{jk})^d$

- Non-linear MKL methods:
  - Have *high computational cost* due to non-convexity
  - Do NOT have significant *accuracy improvement*

Binary-MKL Optimization

- Dual formulation:

$$\min_{\boldsymbol{\beta}\in\Delta}\max_{\alpha\in\mathcal{Q}}\widehat{\mathcal{L}}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}\circ\mathbf{y}),$$

- *Wrapper approaches* for MKL-L1 and MKL-Lp

- *Direct methods* for regularizers that make the objective function smooth

Wrapper Methods

- 2 alternating steps:
  - Solve SVM for the fixed combined kernel
  - Update the kernel weights for the fixed dual variables

- **Pro**: Off-the-shelf SVM methods can be used

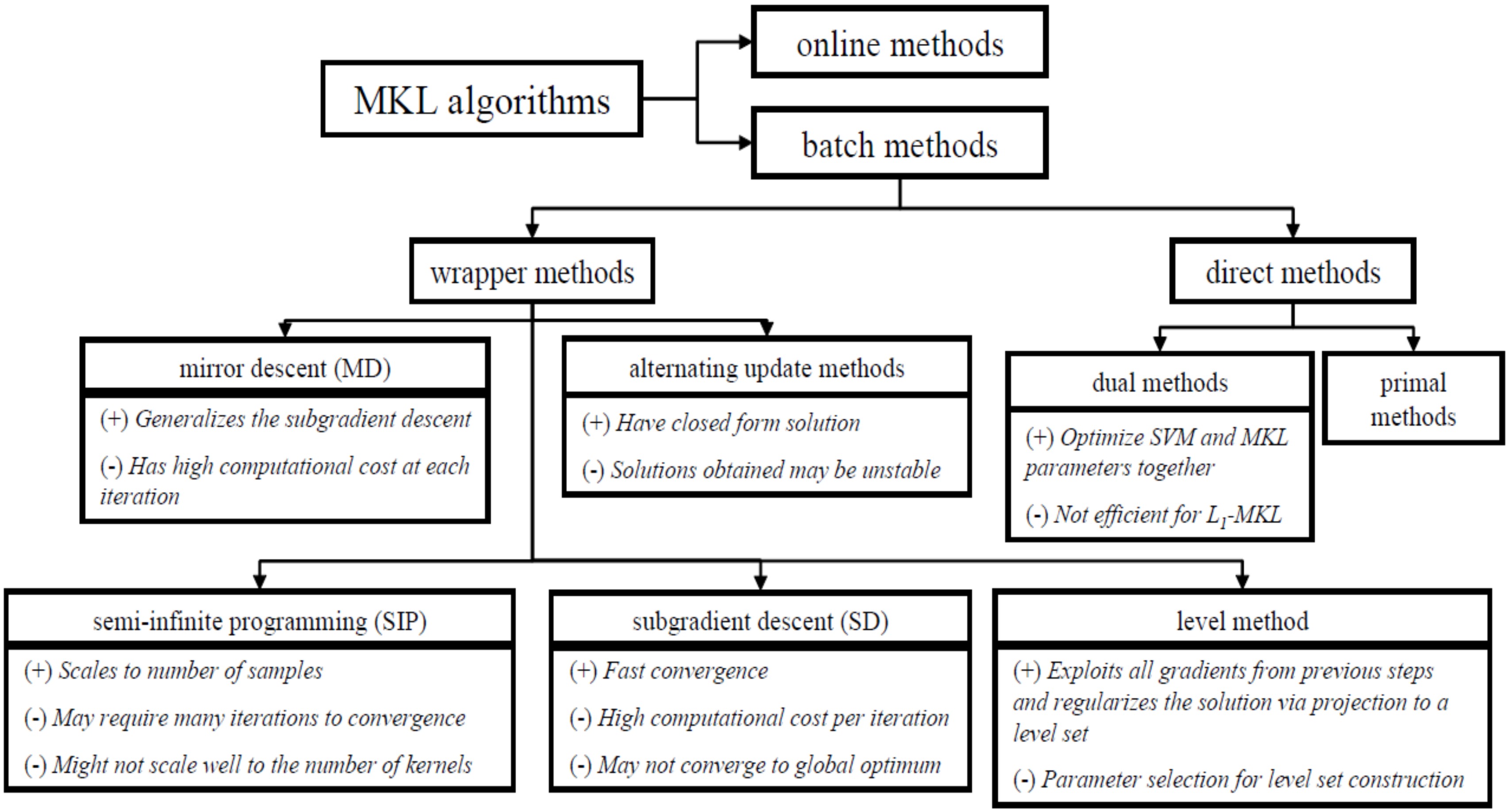
- **Con**: Requires high accuracy in intermediate SVM solutions

Wrapper Methods

- Semi-infinite Programming (SIP) (Sonnenberg et al. 2005)
- Subgradient descend (Rakotomamonjy et al. 2007)
- Level-Method (Xu et al. 2009)
- Mirror-descend (Aflalo et al .2011)
- Alternating update method (Group Lasso) (Xu et al. 2010)

They all solve the same problem: NO accuracy difference when the same regularizer is used!

Binary-MKL Optimization



They all solve the same problem (for the same regularizers!)

MKL-SMO

- Sequential minimization optimization (SMO) is a very efficient optimization tool that is used for SVM

- When $\|\beta\|_p^2$ is used as a regularizer, the optimization problem becomes:

$$\max_{\boldsymbol{\alpha}\in Q} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{8\lambda}\left(\sum_{k=1}^{s}\left[(\boldsymbol{\alpha}\circ\mathbf{y})^\top \mathbf{K}_k(\boldsymbol{\alpha}\circ\mathbf{y})\right]^q\right)^{\frac{2}{q}}$$

- SMO can be used for this formulation because it is **smooth**

Discussions

- The state-of-the-art is linear MKL
  - Non-linear and online MKL approaches are not mature
- Many alternative optimization methods that solve the same problem.
- Is MKL useful at all?
  - Compared to uniform (average) combination
- Which Regularizer should be used?
  - **Sparse** vs **Dense** MKL
- Which optimization method should be selected?

Outline

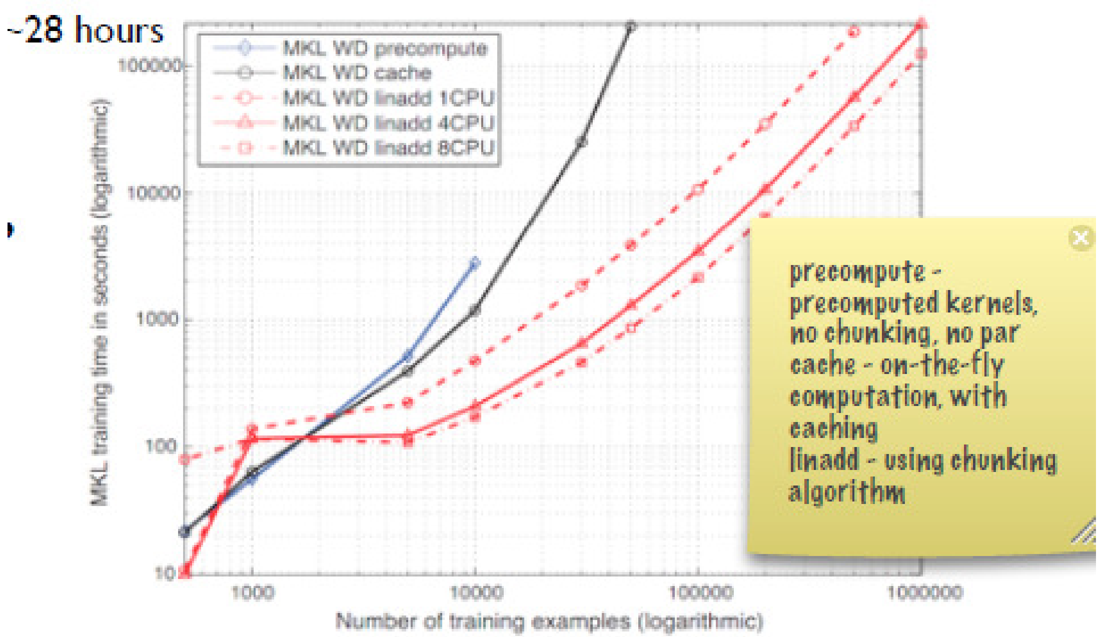
1. Kernel Combination
   1. Heterogeneous Information Fusion
   2. Feature Selection
2. Linear margin classifiers (SVM)
3. Kernel Classifiers
4. Kernel Learning / Multiple Kernel Learning
   1. MKL Formulation
   2. Sparse vs. Smooth MKL
   3. MKL Optimization
5. ***Experimental Results from Literature***
6. Our Experiments
7. Conclusions

Experimental Results

- Let's see the reported results from the literature.
- Since all methods solve the same problem, the evaluation is on:
  - Comparison of accuracy for different norms
  - Comparison of speed for different norms
  - Comparison of speed for different optimization methods

SILP Algorithm- MKL-L1

- 1M samples, 20 base kernels



- 1M samples can be processed in less than 28 hours with caching, chunking, and parallelization.

Cortes et al. Sonnenburg et al. 2006

SimpleMKL vs SILP for MKL-L1

- SimpleMKL Reduced gradient methods

Pima $\ell = 538$ $M = 117$

Algorithm	# Kernel	Accuracy	Time (s)	# SVM eval	# Gradient eval
SILP	11.6 ± 1.0	76.5 ± 2.3	224 ± 37	95.6 ± 13	95.6 ± 13
SimpleMKL	14.7 ± 1.4	76.5 ± 2.6	79.0 ± 13	314 ± 44	24.3 ± 4.8
Grad. Desc.	14.8 ± 1.4	75.5 ± 2.5	219 ± 24	873 ± 147	118 ± 8.7

Sonar $\ell = 146$ $M = 793$

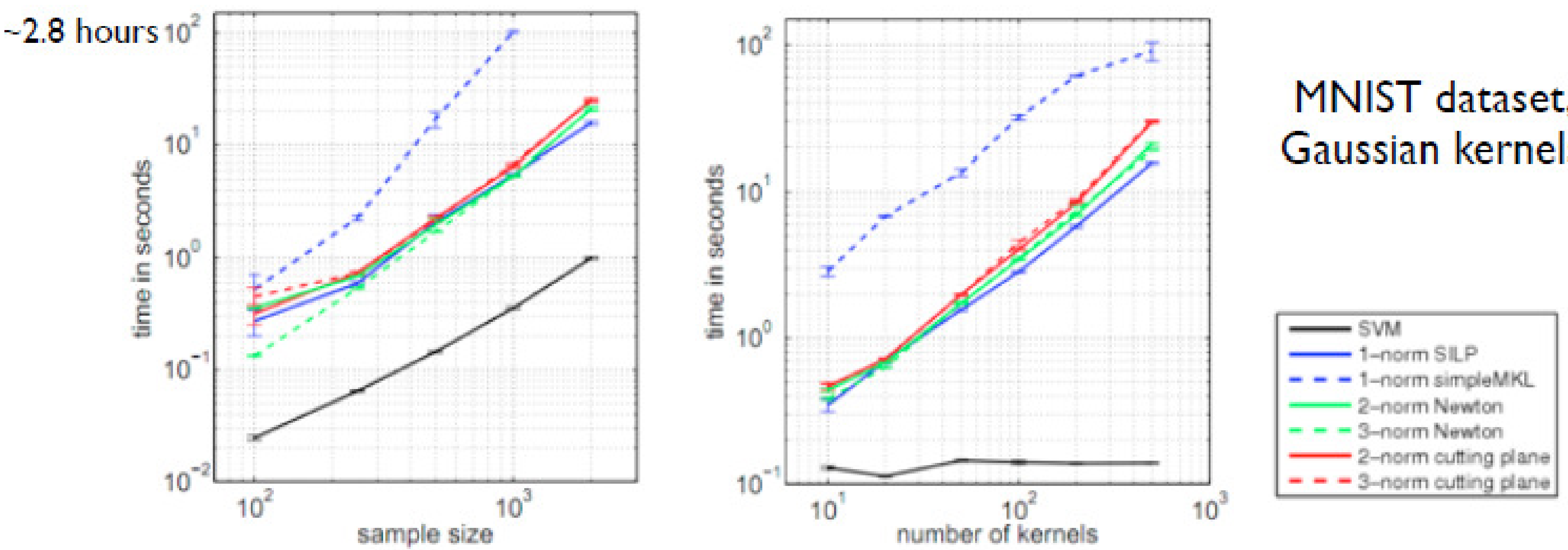
Algorithm	# Kernel	Accuracy	Time (s)	# SVM eval	# Gradient eval
SILP	33.5 ± 3.8	80.5 ± 5.1	2290 ± 864	903 ± 187	903 ± 187
SimpleMKL	36.7 ± 5.1	80.6 ± 5.1	163 ± 93	2770 ± 1560	115 ± 66
Grad. Desc.	35.7 ± 3.9	80.2 ± 4.7	469 ± 90	7630 ± 2600	836 ± 99

- SimpleMKL is faster for small number (hundreds) of samples and large number (hundreds) of kernels.

Cortes et al. Rakotomamonjy al. 2008

MKL-Lp Optimization

- Wrapper method for MKL-Lp: Newton or cutting plane (Kloft 2009)



50 kernels

500 samples

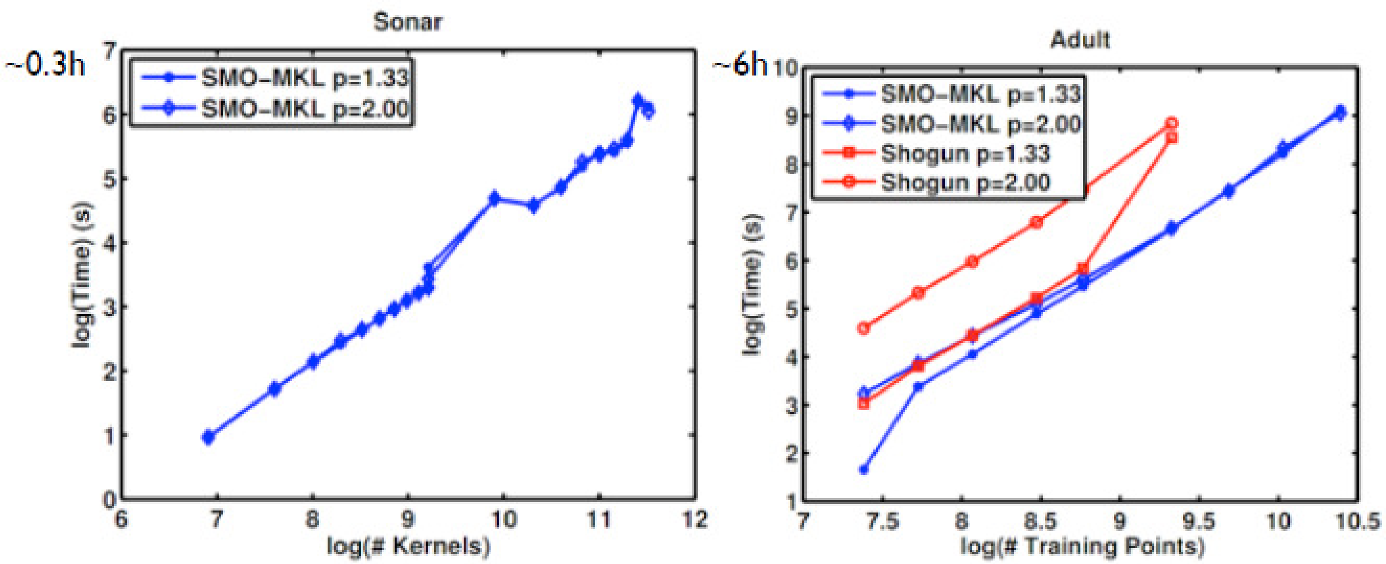
MNIST dataset, Gaussian kernels

- Faster than MKL-L1 (Sonnenburg 2005)

Cortes et al. Kloft et al.

MKL-SMO vs. MKL-SIP (MKL-Lp)

- MKL-SMO is a direct approach
- MKL-SIP is a wrapper method



166 points

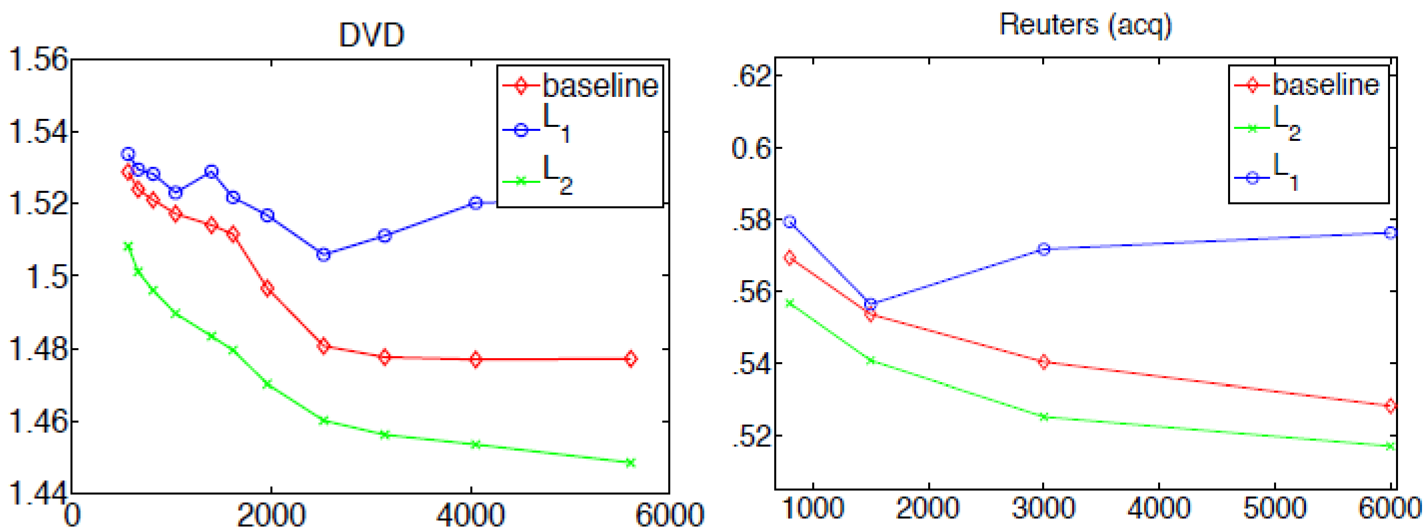
50 kernels

- MKL-SMO is more efficient
- The value of *p* does NOT have a big impact on efficiency

Cortes et al. Wishwanathan et al. 2010

MKL-L1 vs. MKL-L2 (Accuracy)

- Combining kernels on single features – feature selection (Cortes et al. 2009)



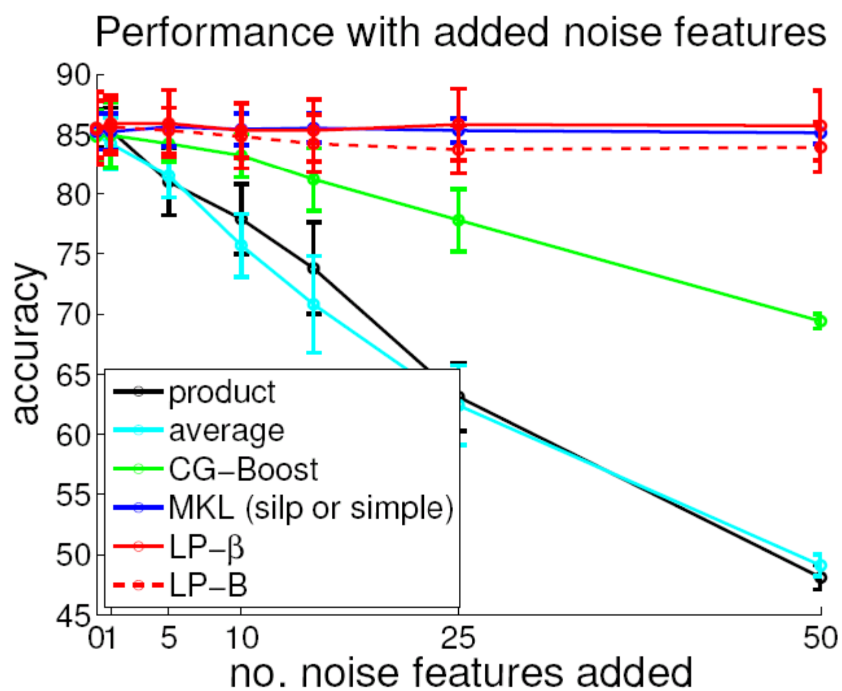
Number of kernels vs. classification error

- Dense combinations are better when using many kernels (MKL-L2 > Average kernel > MKL-L1)

Cortes et al.

MKL-L1 vs. Dense Combination

- Oxford Flowers data set
- Noisy kernels are added



- MKL-L1 is more robust because it can eliminate weak kernels (**contradiction**???)

Cortes et al. Gehler et al.

Learning Bounds for MKL-L1 & MKL-L2

- We have the following theorem:
  - Assume that $K_k(\boldsymbol{x}, \boldsymbol{x}) \leq R^2, \forall k$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, for any $h \in H$
    - L_2 regularization : $R(h) \leq \hat{R}_\rho(h) + O(M^{1/4}\sqrt{\frac{R^2/\rho^2}{n}})$
    - L_1 regularization : $R(h) \leq \hat{R}_\rho(h) + O(\sqrt{(logM)\frac{R^2/\rho^2}{n}})$
- Meaning that, given a sufficiently large number of samples (*n*), *MKL-L1 has a better bound* (more robust) w.r.t. the increased number of kernels.

Theorem from a presentation by AI Lab, University of Geneva et al.

Conclusions So Far

- MKL-L2 is computationally more efficient than MKL-L1
- Dense (smooth) kernel combination performs better than sparse kernel combination
  - Unless there is a large number of noisy kernels
- MKL-L1 does not have advantage over average kernel baseline
- MKL-SMO is the fastest MKL method
  - But NOT available for MKL-L1

Outline

1. Kernel Combination
   1. Heterogeneous Information Fusion
   2. Feature Selection
2. Linear margin classifiers (SVM)
3. Kernel Classifiers
4. Kernel Learning / Multiple Kernel Learning
   1. MKL Formulation
   2. Sparse vs. Smooth MKL
   3. MKL Optimization
5. Experimental Results from Literature
6. ***Our Experiments***
7. Conclusions

Our Additional Experiments

- Data sets
  - Caltech 101: 9,146 images, 101 classes, 48 base kernels
  - VOC2007: 9,963 images, 20 classes, 15 base kernels
  - Subset of ImageNet: 81,738 images, 18 classes, 10 base kernels

Which Wrapper method is the Most Efficient

	Caltech 101 dataset		
	10 training instances per class		
baseline	training	#iter	KerComb
GMKL-L_1	34.6 ± 8.6	38.4 ± 2.0	27.9 ± 7.7
SimpleMKL-L_1	55.7 ± 25.3	17.2 ± 6.8	46.1 ± 22.0
VSKL-L_1	14.1 ± 2.3	38.3 ± 4.3	11.1 ± 1.7
MKL-GL-L_1	21.9 ± 0.8	40.0 ± 0.0	19.5 ± 0.8
MKL-GL-L_2	5.3 ± 0.6	8.8 ± 1.0	4.8 ± 0.6
MKL-GL-L_4	**3.5 ± 0.2**	**5.9 ± 0.4**	3.2 ± 0.2
MKL-Level-L_1	8.0 ± 2.3	33.0 ± 9.5	5.5 ± 1.4
MKL-SIP-L_1	5.4 ± 0.9	39.4 ± 2.6	**2.1 ± 0.3**
MKL-SIP-L_2	**3.8 ± 1.2**	**5.6 ± 0.9**	**2.4 ± 1.1**
MKL-SIP-L_4	**3.3 ± 0.6**	**4.4 ± 0.5**	**1.8 ± 0.6**
	30 training instances per class		
baseline	training	#iter	KerComb
GMKL-L_1	256.7 ± 47.7	38.6 ± 1.8	212.5 ± 42.3
SimpleMKL-L_1	585.6 ± 204.7	19.0 ± 7.5	494.4 ± 174.7
VSKL-L_1	121.9 ± 22.4	36.6 ± 5.1	103.5 ± 17.7
MKL-GL-L_1	197.1 ± 9.1	39.8 ± 1.0	178.3 ± 8.5
MKL-GL-L_2	50.8 ± 5.6	9.3 ± 1.0	46.3 ± 5.2
MKL-GL-L_4	32.5 ± 1.6	5.9 ± 0.3	29.6 ± 1.5
MKL-Level-L_1	63.3 ± 22.1	27.5 ± 11.1	47.9 ± 14.9
MKL-SIP-L_1	44.3 ± 6.1	39.7 ± 2.9	23.2 ± 2.7
MKL-SIP-L_2	30.4 ± 4.2	6.3 ± 1.0	25.2 ± 3.9
MKL-SIP-L_4	**22.6 ± 2.6**	**4.7 ± 0.5**	**18.2 ± 2.1**

- MKL-Lp is more efficient than MKL-L1
- Kernel combination consumes the most time
  - MKL-L1 might have advantage when the number of kernels is very large
- MKL-Lp converges in small number of iterations

Wrapper vs. Direct MKL-Lp

- Performance comparison (training time in sec)
- MKL-SMO (direct MKL-Lp) is faster

		Number of training samples		
Caltech 101		$n = 10$	$n = 20$	$n = 30$
	MKL-SIP	3.6 ±0.2	6.5± 0.3	11.8 ± 0.7
	MKL-SMO	**0.2 ±0.1**	**2.3 ± 0.2**	**3.8 ± 0.5**
VOC 2007		25%	50%	75%
	MKL-SIP	15.5 ± 1.6	145.6 ± 3.9	360.7 ± 8.4
	MKL-SMO	**3.5 ± 0.7**	**14.2± 1.8**	**33.1± 3.0**
		Number of base kernels		
Caltech 101		$K = 48$	$K = 63$	$K = 108$
	MKL-SIP	6.5 ± 0.3	13.6 ± 2.9	19.8 ± 3.4
	MKL-SMO	**2.3 ± 0.2**	**3.2± 0.8**	**6.3± 1.0**
VOC 2007		$K = 15$	$K = 30$	$K = 75$
	MKL-SIP	145.6 ± 3.9	542.0 ± 32.8	1412.1 ± 63.4
	MKL-SMO	**14.2 ± 1.8**	**29.1± 2.8**	**77.8± 10.3**

Classification Performance

Caltech 101

Baseline	Norm	Number of training instances per class 10	20	30
Single		45.3 ± 0.9	55.2 ± 0.9	70.6 ± 0.9
Average		$\mathbf{59.0 \pm 0.7}$	$\mathbf{69.7 \pm 0.6}$	77.2 ± 0.5
MKL-SIP	$p = 1$	53.8 ± 0.6	63.8 ± 0.9	$\mathbf{83.9 \pm 0.7}$
MKL-SIP	$p = 2$	$\mathbf{60.1 \pm 0.6}$	$\mathbf{70.7 \pm 1.0}$	79.1 ± 0.6

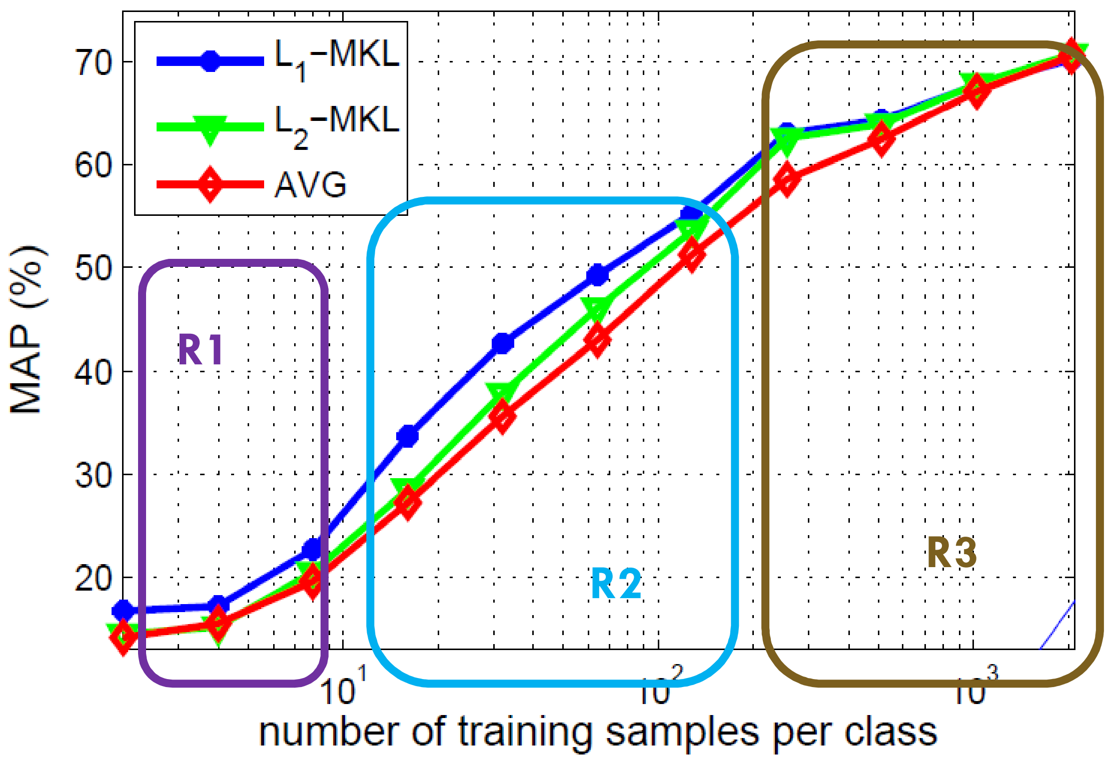
VOC 2007

baseline	Percentage of the samples used for training 1%	25%	50%	75%
Single	$\mathbf{23.4 \pm 0.1}$	44.7 ± 0.8	48.6 ± 0.8	50.0 ± 0.8
Average	21.9 ± 0.5	48.2 ± 0.8	54.5 ± 0.8	57.5 ± 0.8
L_1-MKL	$\mathbf{23.5 \pm 0.7}$	$\mathbf{51.9 \pm 0.4}$	$\mathbf{57.4 \pm 0.4}$	$\mathbf{59.9 \pm 0.9}$
L_2-MKL	22.7 ± 0.4	49.8 ± 0.2	$\mathbf{57.3 \pm 0.2}$	$\mathbf{60.6 \pm 0.5}$

MKL-L1 can perform better when there is a sufficient number of training samples

ImageNet Results

- MKL-L1 is superior going from R1 to R2 (better kernel selection with increased number of samples)



- MKL-L1 has no advantage in R3:
- All base kernels are stronger with increased number of samples.
- Kernel selection has no edge.

Adding Weaker Kernels

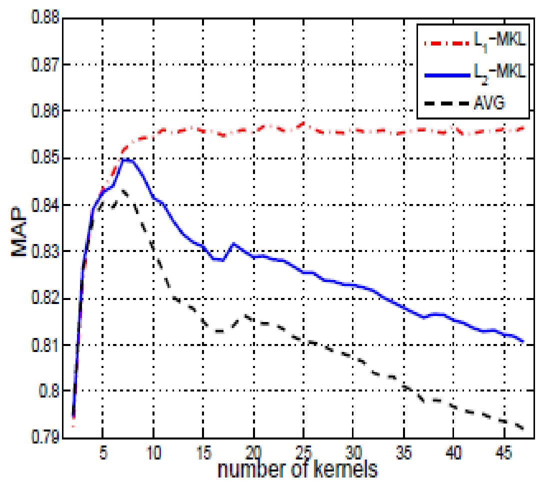


Fig. 4: The change in MAP score with respect to the number of base kernels for the Caltech 101 dataset.

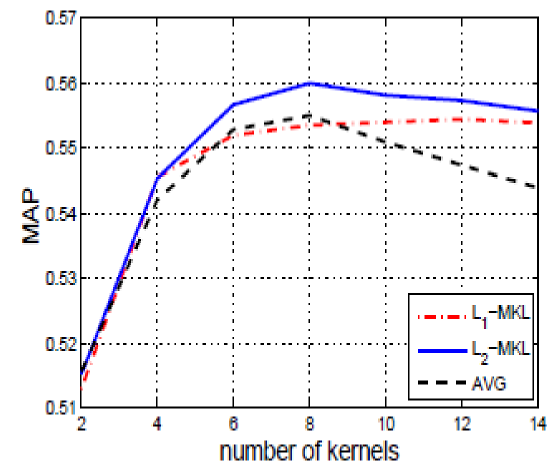


Fig. 5: The change in MAP score with respect to the number of base kernels for the VOC 2007 dataset.

- MKL-L1 is more robust against addition of weak (in terms of accuracy) base kernels

Outline

1. Kernel Combination
    1. Heterogeneous Information Fusion
    2. Feature Selection
2. Linear margin classifiers (SVM)
3. Kernel Classifiers
4. Kernel Learning / Multiple Kernel Learning
    1. MKL Formulation
    2. Sparse vs. Smooth MKL
    3. MKL Optimization
5. Experimental Results from Literature
6. Our Experiments
7. ***Conclusions***

Conclusions I : Comparison of MKL Methods

- Linear MKL is preferred over non-linear
  - Unless you know the exact formulation you need
- MKL-L1 is preferred when:
  - The number of kernels is high (tens or more)
  - Kernel selection is needed
  - The number of training samples is sufficient (hundreds per class)
  - Prediction time is important (sparseness)
- MKL-SMO is the most efficient method (no MKL-L1)
- MKL-SIP is the fastest MKL-L1

Conclusions II: Some Practical Advice

- Multiple MKL packages are available
  - MKL-SMO, Shogun etc.
- Kernel normalization is important
- MKL is scalable for thousands of training samples, hundreds of kernels
  - For more you might need to look into linear methods or kernel approximations
- Average kernel is claimed to perform comparable to sparse MKL
  - Even if it is true, sparseness gives advantage in prediction speed.

Open Problems in MKL

- Improve the computational efficiency
  - Millions of training samples
  - Thousands of base kernels
- Prediction speed
  - Kernel methods are in general not very efficient in prediction (storage and use of support vectors)
- Going beyond linear MKL
  - Online MKL, non-linear MKL, multi-label MKL

MKL Software

- Shogun (MKL-SIP):
  - http://shogun-toolbox.org/
- MKL-SMO:
  - http://research.microsoft.com/en-us/um/people/manik/code/SMO-MKL/download.html
- SimpleMKL (subgradient)
  - http://asi.insa-rouen.fr/enseignants/~arakoto/code/mklindex.html
- Multi-Label MKL: http://www.cse.msu.edu/~bucakser/software.html
- Localized MKL (non-stationary MKL)
  - http://users.ics.aalto.fi/gonen/

References

- S.S. Bucak, R. Jin, A.K. Jain, "Multiple Kernel Learning for Visual Object Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*
- M. Gnen and E. Alpaydin, "Multiple Kernel Learning Algorithms," *Journal of Machine Learning Research, vol. 12,* pp. 2211–2268, 2011.
- S. Vishwanathan, Z. Sun, N. Ampornpunt, and M. Varma, "Multiple Kernel Learning and the SMO Algorithm," in *NIPS 23, 2010, pp. 2361–2369.*
- M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "Lp-norm Multiple Kernel Learning," *Journal of Machine Learning Research, vol. 12, pp. 953–997, 2011.*
- M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *ICML, June 2009, pp. 1065*–1072.
- J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao, "Group- Sensitive Multiple Kernel Learning for Object Categorization," in *ICCV, 2009*

References

- P. V. Gehler and S.Nowozin, “On Feature Combination for Multiclass Object Classification,” in *ICML, 2009*
- C. Cortes, M. Mohri, and A. Rostamizadeh, “Learning Nonlinear Combinations of Kernels,” in *NIPS 22, 2009, pp. 396–404.*
- C. Cortes, M. Mohri, and A. Rostamizadeh, “Generalization Bounds for Learning Kernels,” in *ICML, 2010.*
- A. Rakotomamonjy, F. Bach, Y. Grandvalet, and S. Canu, “SimpleMKL,” *Journal of Machine Learning Research, vol. 9,* pp. 2491–2521, 2008.
- S. Sonnenburg, G. R¨atsch, C. Sch¨afer, and B. Sch¨olkopf, “Large Scale Multiple Kernel Learning,” *Journal of Machine Learning Research, vol. 7, pp. 1531–1565, 2006.*