

Multi-Task Learning and Matrix Regularization

Andreas Argyriou

Department of Computer Science
University College London

Collaborators

- T. Evgeniou (INSEAD)
- R. Hauser (University of Oxford)
- M. Herbster (University College London)
- A. Maurer (Stemmer Imaging)
- C.A. Micchelli (SUNY Albany)
- M. Pontil (University College London)
- Y. Ying (University of Bristol)

Main Themes

- Machine learning
- Convex optimization
- Sparse recovery

Outline

- Multi-task learning and related problems
- Matrix learning and an alternating algorithm
- Extensions of the method
- Multi-task representer theorems
- Kernel hyperparameter learning; convex kernel learning

Supervised Learning (Single-Task)

- m examples are given: $(x_1, y_1), \dots, (x_m, y_m) \in \mathcal{X} \times \mathcal{Y}$
- Predict using a function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Want the function to generalize well over the whole of $\mathcal{X} \times \mathcal{Y}$
- Includes regression, classification etc.
- Task = probability measure on $\mathcal{X} \times \mathcal{Y}$

Multi-Task Learning

- Tasks $t = 1, \dots, n$
- m examples per task are given: $(x_{t1}, y_{t1}), \dots, (x_{tm}, y_{tm}) \in \mathcal{X} \times \mathcal{Y}$
- Predict using functions $f_t : \mathcal{X} \rightarrow \mathcal{Y}$, $t = 1, \dots, n$
- When the tasks are related, learning the tasks *jointly* should perform better than learning each task *independently*
- Especially important when *few data points* are available *per task* (small m); in such cases, independent learning is not successful

Multi-Task Learning (contd.)

- One goal is to learn what *structure* is common across the n tasks
- Want simple, interpretable models that can explain multiple tasks
- Want good generalization on the n given tasks but also on new tasks (*transfer learning*)
- Given *a few examples* from a new task t' , $\{(x_{t'1}, y_{t'1}), \dots, (x_{t'l}, y_{t'l})\}$, want to learn $f_{t'}$ using just the learned task structure

Learning Theoretic View: Environment of Tasks

- Environment = probability distribution on a set of learning tasks
[*Baxter, 1996*]
- To sample a task-specific sample from the environment
 - draw a function f_t from the environment
 - generate a sample $\{(x_{t1}, y_{t1}), \dots, (x_{tm}, y_{tm})\} \in (\mathcal{X} \times \mathcal{Y})^m$
using f_t
- Multi-task learning means learning *a common hypothesis space*

Learning Theoretic View (contd.)

- Baxter's results:
 - As n (#tasks) increases, m (#examples per task needed) decreases as $O(\frac{1}{n})$
 - Once we have learned a hypothesis space \mathcal{H} , we can use it to learn a new task drawn from the same environment; the sample complexity depends on the log-capacity of \mathcal{H}
- Other results:
 - Task relatedness due to input transformations: improved multi-task bounds in some cases [*Ben-David & Schuller, 2003*]
 - Using common feature maps (bounded linear operators): error bounds depend on Hilbert-Schmidt norm [*Maurer, 2006*]

Multi-Task Applications

- Multi-task learning is ubiquitous
- Human intelligence relies on transferring learned knowledge from previous tasks to new tasks
- E.g. character recognition (very few examples should be needed to recognize new characters)
- Integration of medical / bioinformatics databases

Multi-Task Applications (contd.)

- Marketing databases, collaborative filtering, recommendation systems (e.g. Netflix); task = product preferences for each person

Description				
Closure	Type of winery	Type of wine	Price	Your rating
Metacork	International	Blush red	\$25	
Metacork	Mid-sized regional	Dry white	\$20	
Traditional cork	Small boutique	Dry red	\$20	
Screwcap	International	Dry red	\$30	
Metacork	Small boutique	Aromatic white	\$30	
Traditional cork	International	Dry white	\$15	
Screwcap	Large national	Blush red	\$20	
Synthetic cork	International	Aromatic white	\$20	

Related Problems

- Sparse coding (some images share common basis images)
- Vector-valued / structured output
- Multi-class problems
- Regression with grouped variables, multifactor ANOVA in statistics; (selection of groups of variables)
- Multi-task learning is a *broad problem*; no single method can solve everything

Learning Multiple Tasks with a Common Kernel

- Let $\mathcal{X} \subseteq \mathbb{R}^d, \mathcal{Y} \subseteq \mathbb{R}$ and let us learn n linear functions

$$f_t(x) = \langle w_t, x \rangle \quad t = 1, \dots, n$$

(we ignore nonlinearities for the moment)

- Want to impose *common structure / relatedness* across tasks
- Idea: use *a common linear kernel* for all tasks

$$K(x, x') = \langle x, D x' \rangle$$

(where $D \succ 0$)

Learning Multiple Tasks with a Common Kernel

- For every $t = 1, \dots, n$ solve

$$\min_{w_t \in \mathbb{R}^d} \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \langle w_t, D^{-1} w_t \rangle$$

- Adding up, we obtain the equivalent problem

$$\min_{w_1, \dots, w_n \in \mathbb{R}^d} \sum_{t=1}^n \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \sum_{t=1}^n \langle w_t, D^{-1} w_t \rangle$$

Learning Multiple Tasks with a Common Kernel

- For multi-task learning, we want to *learn the common kernel* from a *convex set* of kernels:

$$\inf_{\substack{w_1, \dots, w_n \in \mathbb{R}^d \\ D \succ 0, \text{tr}(D) \leq 1}} \sum_{t=1}^n \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \text{tr}(W^\top D^{-1} W) \quad (\mathcal{MTL})$$

$$\sum_{t=1}^n \langle w_t, D^{-1} w_t \rangle$$

↑

- We denote $W = \begin{pmatrix} | & & | \\ w_1 & \dots & w_n \\ | & & | \end{pmatrix}$

Learning Multiple Tasks with a Common Kernel

- *Jointly convex* problem in (W, D)
- The constraint $\text{tr}(D) \leq 1$ is important
- Fixing W , the optimal $D(W)$ is

$$D(W) \propto (WW^\top)^{\frac{1}{2}}$$

($D(W)$ is usually not in the feasible set because of the inf)

- Once we have learned \hat{D} , we can *transfer* it to learning of a new task t'

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^m E(\langle w, x_{t'i} \rangle, y_{t'i}) + \gamma \langle w, \hat{D}^{-1}w \rangle$$

Alternating Minimization Algorithm

- Alternating minimization over W (supervised learning) and D (unsupervised “correlation” of tasks).

Initialization: set $D = \frac{I_{d \times d}}{d}$

while convergence condition is not true **do**

for $t = 1, \dots, n$ learn w_t *independently* by minimizing

$$\sum_{i=1}^m E(\langle w, x_{ti} \rangle, y_{ti}) + \gamma \langle w, D^{-1}w \rangle$$

end for

 set $D = \frac{(WW^T)^{\frac{1}{2}}}{\text{tr}(WW^T)^{\frac{1}{2}}}$

end while

Alternating Minimization (contd.)

- Each w_t step is a regularization problem (e.g. SVM, ridge regression etc.)
- It does *not* require computation of the (pseudo)inverse of D
- Each D step requires an SVD; this is usually the most costly step

Alternating Minimization (contd.)

- The algorithm (with some perturbation) *converges* to an optimal solution

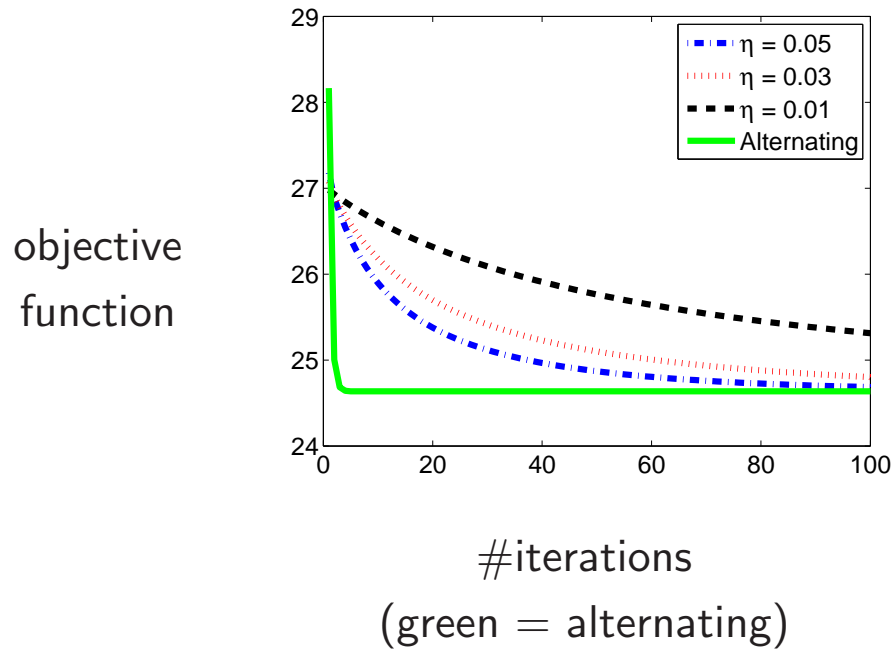
$$\min_{\substack{w_1, \dots, w_n \in \mathbb{R}^d \\ D \succ 0, \text{tr}(D) \leq 1}} \sum_{t=1}^n \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \text{tr} (D^{-1}(WW^\top + \varepsilon I)) \quad (\mathcal{R}_\varepsilon)$$

Theorem. *An alternating algorithm for problem $(\mathcal{R}_\varepsilon)$ has the property that its iterates $(W^{(k)}, D^{(k)})$ converge to the minimizer of $(\mathcal{R}_\varepsilon)$ as $k \rightarrow \infty$.*

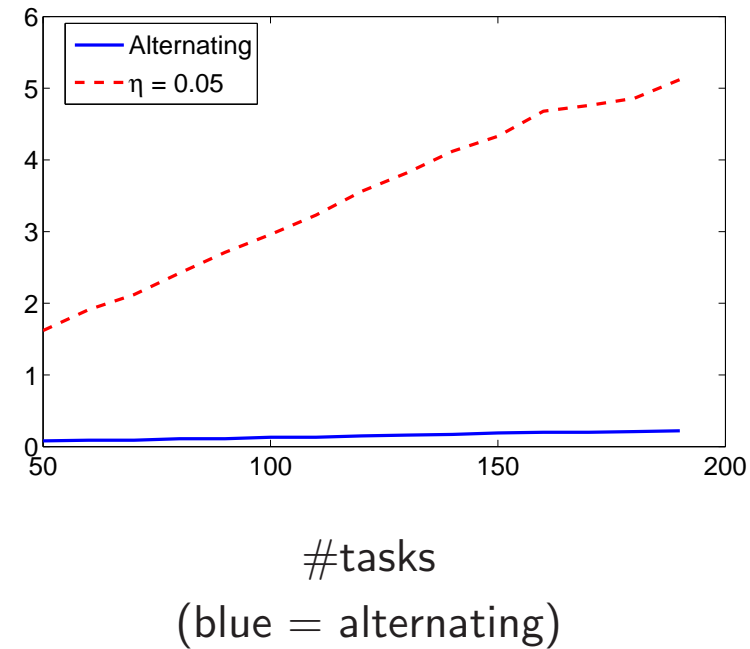
Theorem. *Consider a sequence $\varepsilon_\ell \rightarrow 0^+$ and let (W_ℓ, D_ℓ) be the minimizer of $(\mathcal{R}_{\varepsilon_\ell})$. Then any limiting point of the sequence $\{(W_\ell, D_\ell)\}$ is an optimal solution of the problem (\mathcal{MTL}) .*

- Note: the starting value of D does not matter

Alternating Minimization (contd.)



seconds



- Compare computational cost with a gradient descent approach ($\eta :=$ learning rate)

Alternating Minimization (contd.)

- Typically fewer than 50 iterations needed in experiments
- At least *an order of magnitude* fewer iterations than gradient descent (but cost per iteration is larger)
- Scales better with the number of tasks
- Both methods require SVD (costly if d is large)
- Alternative algorithms: SOCP methods [*Srebro et al. 2005, Liu and Vandenberghe 2008*], gradient descent on matrix factors [*Rennie & Srebro 2005*], singular value thresholding [*Cai et al. 2008*]

Trace Norm Regularization

- Eliminating D in optimization problem (\mathcal{MTL}) yields

$$\min_{W \in \mathbb{R}^{d \times n}} \sum_{t=1}^n \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \|W\|_{tr}^2 \quad (\mathcal{TR})$$

The *trace norm* (or nuclear norm) $\|W\|_{tr}$ is the sum of the singular values of W

- There has been recent interest in trace norm / rank problems in *matrix factorization, statistics, matrix completion etc.* [Cai et al. 2008, Fazel et al. 2001, Izenman 1975, Liu and Vandenberghe 2008, Srebro et al. 2005]

Trace Norm vs. Rank

- Problem (\mathcal{TR}) is a convex relaxation of the problem

$$\min_{W \in \mathbb{R}^{d \times n}} \sum_{t=1}^n \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \text{rank}(W)$$

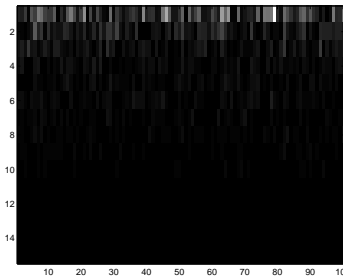
- NP-hard problem (at least as hard as Boolean LP)
- Rank and trace norm correspond to L_0 , L_1 on the vector of singular values
- Multi-task intuition: we want the task parameter vectors w_t to lie on a *low dimensional* subspace

Connection to Group Lasso

- Problem (\mathcal{MTL}) is equivalent to

$$\min_{\substack{A \in \mathbb{R}^{d \times n} \\ U \in \mathbb{R}^{d \times d}, U^\top U = I}} \sum_{t=1}^n \sum_{i=1}^m E(\langle a_t, U^\top x_{ti} \rangle, y_{ti}) + \gamma \|A\|_{2,1}^2$$

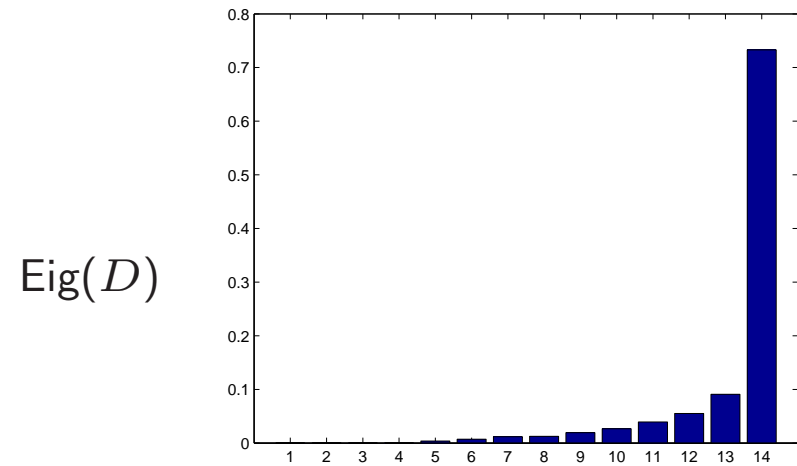
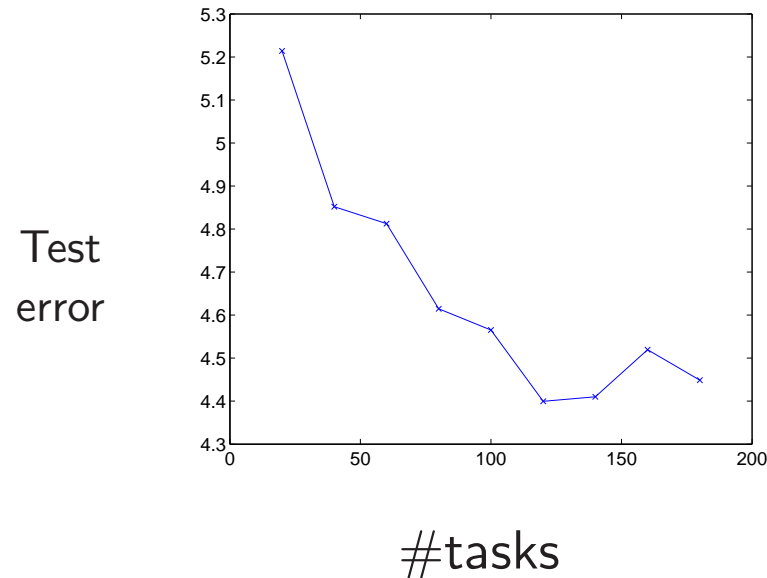
where $\|A\|_{2,1} := \sum_{i=1}^d \sqrt{\sum_{t=1}^n a_{it}^2}$



Experiment (Computer Survey)

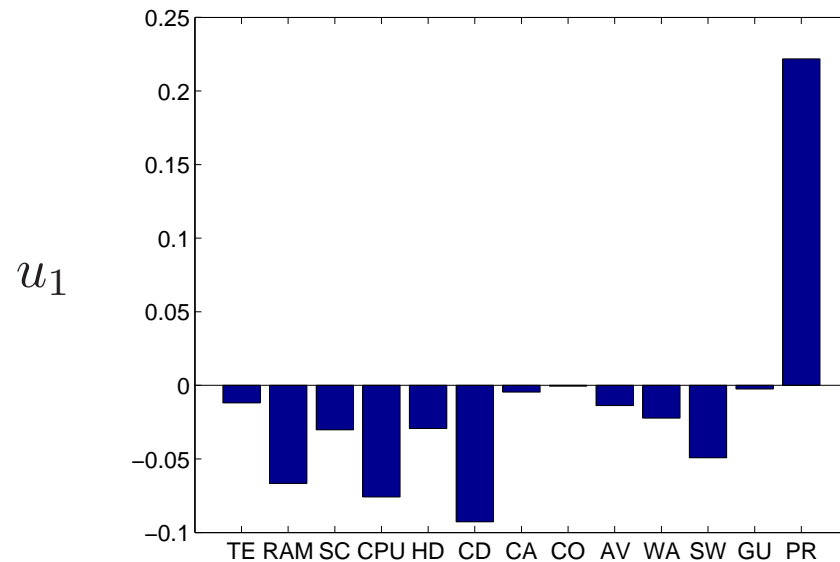
- Consumers' ratings of products [Lenk et al. 1996]
- 180 persons (tasks)
- 8 PC models (training examples); 4 PC models (test examples)
- 13 binary input variables (RAM, CPU, price etc.) + bias term
- Integer output in $\{0, \dots, 10\}$ (likelihood of purchase)
- The square loss was used

Experiment (Computer Survey)



- Performance improves with more tasks
(for learning of the tasks independently, error = 16.53)
- A single most important feature shared by all persons

Experiment (Computer Survey)



Method	RMSE
Alternating Alg.	1.93
Hierarchical Bayes [Lenk et al.]	1.90
Independent	3.88
Aggregate	2.35
Group Lasso	2.01

- The most important feature (eigenvector of D) weighs *technical characteristics* (RAM, CPU, CD-ROM) vs. *price*

Spectral Regularization

- Generalize (\mathcal{MTL}):

$$\inf_{\substack{w_1, \dots, w_n \in \mathbb{R}^d \\ D \succ 0, \text{tr}(D) \leq 1}} \sum_{t=1}^n \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \text{tr}(W^\top F(D)W)$$

where F is a *spectral* matrix function, $f : (0, +\infty) \rightarrow (0, +\infty)$,

$$F(U \Lambda U^\top) = U \text{diag}[f(\lambda_1), \dots, f(\lambda_d)] U^\top$$

or

$$\min_{W \in \mathbb{R}^{d \times n}} \sum_{t=1}^n \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \Omega(W)$$

- It can be shown that $\Omega(W)$ is a function of the singular values of W

Spectral Regularization (contd.)

- In particular, if $f(\lambda) = \lambda^{1-\frac{2}{p}}$, $p \in (0, 2]$, we have

$$\Omega(W) = \|W\|_p^2$$

where $\|W\|_p$ is the Schatten L_p (pre)norm of the singular values of W

Theorem. *The regularizer $\text{tr}(W^\top F(D) W)$ is jointly convex if and only if $\frac{1}{f}$ is matrix concave of order d , that is,*

$$\mu \left(\frac{1}{F} \right) (A) + (1 - \mu) \left(\frac{1}{F} \right) (B) \preceq \left(\frac{1}{F} \right) (\mu A + (1 - \mu)B)$$

for all $A, B \succ 0$, $\mu \in [0, 1]$

- Spectral problems appear also in graph applications, control theory etc.

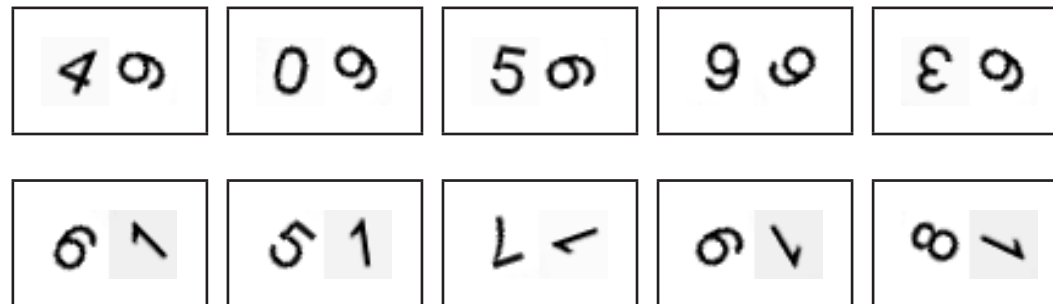
Learning Groups of Tasks

- Assume *heterogeneous* environment, i.e. K low dimensional subspaces
- Learn a partition of tasks in K groups

$$\inf_{\substack{D_1, \dots, D_K \succ 0 \\ \text{tr}(D_k) \leq 1}} \sum_{t=1}^n \min_{k=1}^K \min_{w_t \in \mathbb{R}^d} \left\{ \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \langle w_t, D_k^{-1} w_t \rangle \right\}$$

- The representation learned is $(\hat{D}_1, \dots, \hat{D}_K)$; we can transfer this representation to easily learn a new task
- *Non-convex* problem; we use stochastic gradient descent

Experiment (Character Recognition - Projection on Image Halves)



6 vs. 1 task (on the right half)

- Binary classification tasks on 28×56 images
- One half of the image contains the relevant character, the other half contains a randomly chosen character
- Two groups of tasks (with probabilities 50-50%): projection on left half - projection on right half

Experiment (Character Recognition - Projection on Image Halves)

- Training set contains pairs of *alphabetic* characters
- 1000 tasks, 10 examples per task
- Wish to obtain a representation that represents rotation invariance on either half of the image
- Wish to transfer this representation to pairs of *digits*

Experiment (Character Recognition - Projection on Image Halves)

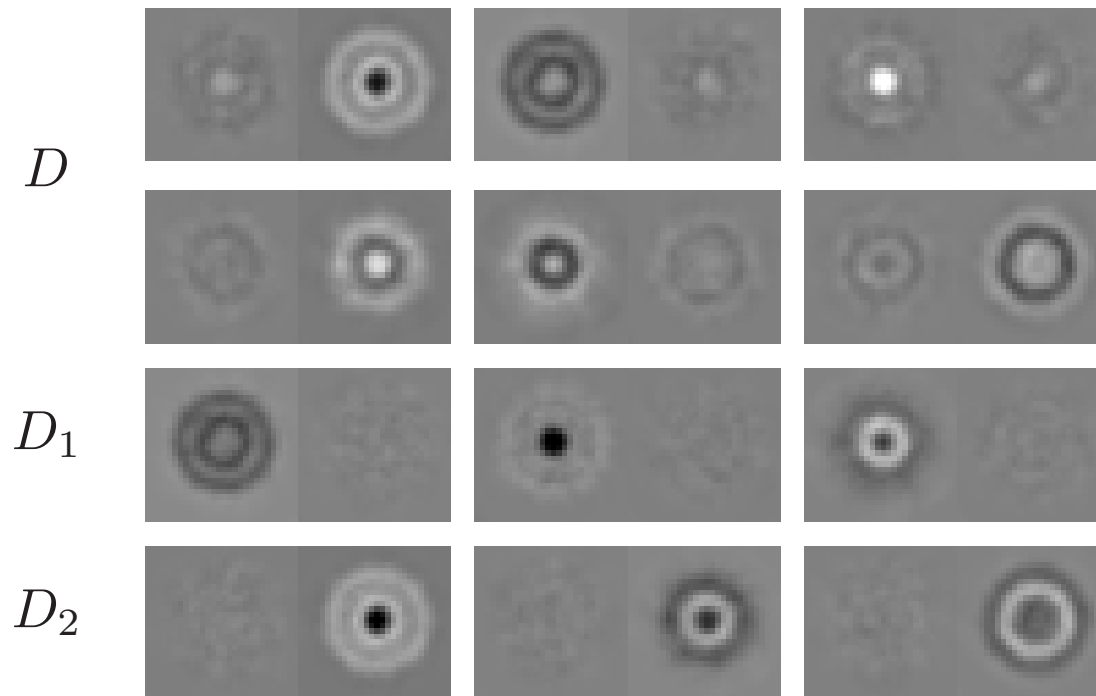
Independent	$K = 1$	$K = 2$
0.27	0.036	0.013

Transfer error for different methods

	D_1	D_2
All digits (left & right)	48.2%	51.8%
Left	99.2%	0.8%
Right	1.4%	98.6%
Training data	50.7%	49.3%

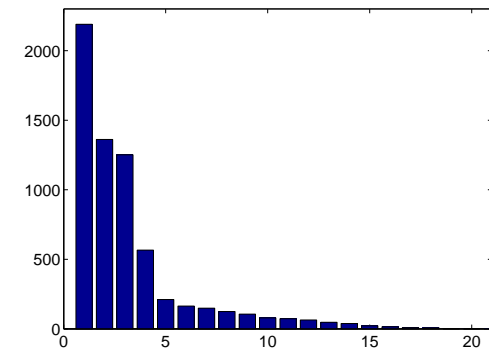
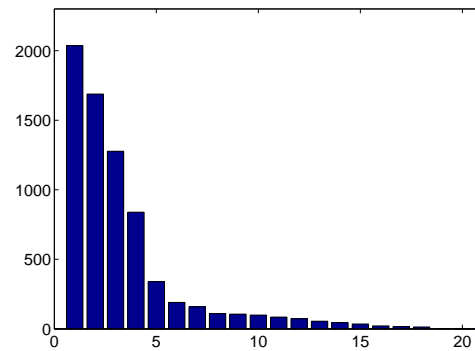
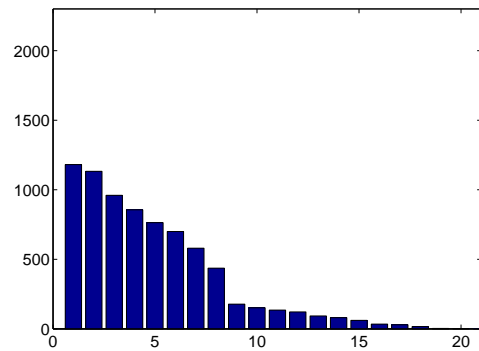
Assignment of tasks in groups (with $K = 2$)

Experiment (Character Recognition - Projection on Image Halves)



Dominant eigenvectors of D (for $K = 1$) and D_1, D_2 (for $K = 2$)

Experiment (Character Recognition - Projection on Image Halves)



Spectrum of D learned with $K = 1$ (left)
Spectra of D_1 (middle) and D_2 (right) learned with $K = 2$

Representer Theorems

- All previous formulations satisfy a *multi-task representer theorem*

$$\hat{w}_t = \sum_{s=1}^n \sum_{i=1}^m c_{si}^{(t)} x_{si} \quad \forall t \in \{1, \dots, n\} \quad (\mathcal{R.T.})$$

Consequently, a nonlinear *kernel* can be used

- *All tasks* are involved in this expression (unlike the single-task representer theorem \Leftrightarrow Frobenius norm regularization)
- Generally, consider any matrix optimization problem of the form

$$\min_{w_1, \dots, w_n \in \mathbb{R}^d} \sum_{t=1}^n \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \Omega(W)$$

Representer Theorems (contd.)

- **Definitions:**

\mathbf{S}_+^n = the positive semidefinite cone

The function $h : \mathbf{S}_+^n \rightarrow \mathbb{R}$ is matrix nondecreasing, if

$$h(A) \leq h(B) \quad \forall A, B \in \mathbf{S}_+^n \quad \text{s.t. } A \preceq B$$

Theorem. *Rep. thm. (R.T.) holds if and only if there exists a matrix nondecreasing function $h : \mathbf{S}_+^n \rightarrow \mathbb{R}$ such that*

$$\Omega(W) = h(W^\top W) \quad \forall W \in \mathbb{R}^{d \times n}$$

(under differentiability assumptions)

Representer Theorems (contd.)

Corollary. *The standard rep. thm. for single-task learning ($n = 1$),*

$$\hat{w} = \sum_{i=1}^m c_i x_i$$

holds if and only if there exists a nondecreasing function $h : \mathbb{R}_+ \rightarrow \mathbb{R}$ such that

$$\Omega(w) = h(\langle w, w \rangle) \quad \forall w \in \mathbb{R}^d$$

- Sufficiency of the condition has been known [*Kimeldorf & Wahba, 1970, Schölkopf et al., 2001* etc.]

Implications

- “Kernelization”
- In single-task learning, the choice of h does not matter essentially
- However, in multi-task learning, the choice of h is important (since \preceq is a partial ordering)
- Many valid regularizers: Schatten L_p norms $\|\cdot\|_p$, rank, orthogonally invariant norms, norms of type $W \mapsto \|WM\|_p$ etc.
- In matrix learning, kernels and sparsity can be exploited in the same model

Connection to Learning the Kernel

- Recall problem (\mathcal{MTL})

$$\min_{D \succ 0, \text{tr}(D) \leq 1} \min_{w_1, \dots, w_n \in \mathbb{R}^m} \sum_{t=1}^n \sum_{i=1}^m E(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \langle w_t, D^{-1} w_t \rangle$$

$(\mathcal{MTL}) \iff$ learning a common kernel $K(x, x') = \langle x, D x' \rangle$ within the convex hull of an **infinite** number of *linear kernels*

- Extends formulation of [Lanckriet et al. 2004] (single task)

$$\min_{K \in \mathcal{K}} \min_{c \in \mathbb{R}^m} \sum_{i=1}^m E((\mathbf{K}c)_i, y_i) + \gamma \langle c, \mathbf{K}c \rangle$$

in which \mathcal{K} was a *polytope* (convex hull of a finite set of kernels)

A General Framework for Learning the Kernel

- Convex set \mathcal{K} is generated by *basic kernels*: $\mathcal{K} = \text{conv}(\mathcal{B})$
- Example 1: *Finite set* of basic kernels (aka “multiple kernel learning”)
- Example 2: *Linear* basic kernels

$$B(x, x') = \langle x, Dx' \rangle$$

where $D \in$ bounded, convex set (e.g. (\mathcal{MTL}))

- Example 3: *Gaussian* basic kernels

$$B(x, x') = e^{-\langle x-x', \Sigma^{-1}(x-x') \rangle}$$

where Σ belongs in a convex subset of the p.s.d. cone

Learning the Kernel and Structured Sparsity

- Interpretation of LTK in the feature space
[*Bach et al. 2004, Micchelli & Pontil 2005*]

$$\min_{v_1, \dots, v_N \in \mathbb{R}^m} \sum_{i=1}^m E \left(\sum_{j=1}^N \langle v_j, \Phi_j(x_i) \rangle, y_i \right) + \gamma \left(\sum_{j=1}^N \|v_j\| \right)^2$$

- Group Lasso in the feature space
- The $\|\cdot\|_{2,1}$ norm tends to favor a small number of feature maps / kernels in the solution

Why Learn Kernels in Convex Sets?

- Data fusion (e.g. in bioinformatics)
- Kernel hyperparameter learning (e.g. Gaussian kernel parameters)
- Multi-task learning
- Metric learning
- Semi-supervised learning (learning the graph)
- Efficient alternative to cross validation; exploits the power of convex optimization

Properties of the Solution of LTK

- Formulation for learning the kernel

$$\min_{K \in \mathcal{K}} \min_{c \in \mathbb{R}^m} \sum_{i=1}^m E((\mathbf{K}c)_i, y_i) + \gamma \langle c, \mathbf{K}c \rangle \quad (\mathcal{LTK})$$

where $\mathcal{K} = \text{conv}(\mathcal{B})$

- I.e. solutions of (\mathcal{LTK}) are of the form $\hat{K} = \sum_{i=1}^N \hat{\lambda}_i \hat{B}_i$, where

$$\hat{\lambda}_i \geq 0, \sum_{i=1}^N \hat{\lambda}_i = 1, \hat{B}_i \in \mathcal{B}$$

- Any solution (\hat{c}, \hat{K}) of (\mathcal{LTK}) is a *saddle point* of a minimax problem

Properties of the Solution of LTK (contd.)

Theorem. (\hat{c}, \hat{K}) solves (\mathcal{LTK}) if and only if

$$1. \langle \hat{c}, \hat{\mathbf{B}}_i \hat{c} \rangle = \max_{B \in \mathcal{B}} \langle \hat{c}, \mathbf{B} \hat{c} \rangle \quad i = 1, \dots, N$$

2. \hat{c} is the solution to

$$\min_{c \in \mathbb{R}^m} \sum_{i=1}^m E((\hat{\mathbf{K}}c)_i, y_i) + \gamma \langle c, \hat{\mathbf{K}}c \rangle$$

Moreover, there exists a solution involving *at most $m + 1$ kernels*:

$$\hat{K} = \sum_{i=1}^{m+1} \hat{\lambda}_i \hat{B}_i$$

A General Algorithm for Learning the Kernel

- Incrementally builds an estimate of the solution, $K^{(k)} = \sum_{i=1}^k \lambda_i K^{(i)}$

Initialization: Given an initial kernel $K^{(1)}$ in the convex set \mathcal{K}

while convergence condition is not true **do**

1. Compute $\hat{c} = \operatorname{argmin}_{c \in \mathbb{R}^m} \sum_{i=1}^m E((\mathbf{K}^{(k)}c)_i, y_i) + \gamma \langle c, \mathbf{K}^{(k)}c \rangle$

2. Find a basic kernel $\hat{B} \in \operatorname{argmax}_{B \in \mathcal{B}} \langle \hat{c}, \mathbf{B} \hat{c} \rangle$

3. Compute $K^{(k+1)}$ as the optimal convex combination of \hat{B} and $K^{(k)}$

end while

A General Algorithm for Learning the Kernel (contd.)

Theorem. *There exists a limit point of the LTK algorithm and any limit point is a solution of (\mathcal{LTK}) .*

- Step 1 is standard regularization (SVM, ridge regression etc.)
- Step 2 is the hardest: tractable for e.g. finite \mathcal{B} or MTL; non-convex for e.g. Gaussian kernels
- Step 3 is convex in one variable (requires solving a few regularization problems)
- Some non-convex cases (e.g. few-parameter Gaussians) are solvable

Character Recognition Experiments

- MNIST classification of digits.
- **1st experiment:** Gaussian basic kernels with one parameter σ .
- Compared with
 - continuously parameterized + local search
 - finite grid of basic kernels
 - SVM
- **2nd experiment:** Gaussian basic kernels with two parameters σ_1, σ_2 (left, right halves of images).
- Compared with varying finite grids.

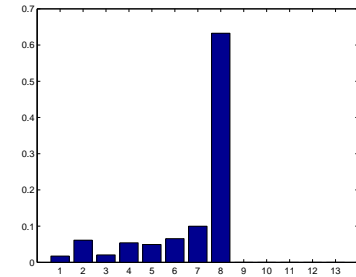
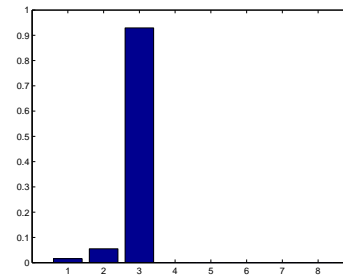
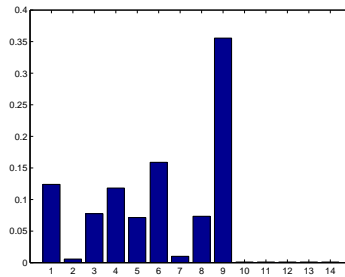
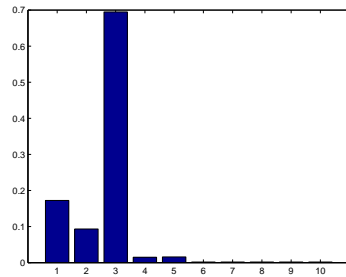
Character Recognition Experiments (contd.)

Task	Method											
	LTK	local	finite	SVM	LTK	local	finite	SVM	LTK	local	finite	SVM
	$\sigma \in [75, 25000]$				$\sigma \in [100, 10000]$				$\sigma \in [500, 5000]$			
odd-even	6.5	6.6	18.0	11.8	6.5	6.6	10.9	8.6	6.5	6.5	6.7	6.9
3 vs. 8	3.7	3.8	6.9	6.0	3.9	3.8	4.9	5.1	3.6	3.8	3.7	3.8
4 vs. 7	2.7	2.5	4.2	2.8	2.4	2.5	2.7	2.6	2.3	2.5	2.6	2.3

Task	LTK	5 × 5	10 × 10	LTK	5 × 5	10 × 10	LTK	5 × 5	10 × 10
	$\sigma \in [75, 25000]$			$\sigma \in [100, 10000]$			$\sigma \in [500, 5000]$		
	odd-even	5.8	15.8	11.2	5.8	10.1	6.2	5.8	6.8
3 vs. 8	2.7	6.5	5.1	2.5	4.6	2.5	2.6	3.5	2.5
4 vs. 7	1.8	3.9	2.9	1.7	2.7	2.0	1.8	2.0	1.8

- Using two parameters improves performance
- Continuous vs. finite grid
 - more efficient
 - robust wrt. parameter range
 - does not overfit

Character Recognition Experiments (contd.)



Learned kernel coefficients. From left to right: odd vs. even (1 and 2 kernel params.), 3 vs. 8 and 4 vs. 7 (2 params.)

Learning the Graph in Semi-Supervised Learning

- We can also apply LTK when there are *few* labeled data points
- A number of graphs are given, e.g. generated using k -NN, exponential decay weights etc.
- To exploit the structure of each graph, the Gram matrices \mathbf{B}_i are taken to be the pseudoinverses of the *graph Laplacians*

Conclusion

- Multi-task learning is ubiquitous; exploiting task relatedness can enhance learning performance significantly
- Proposed an *alternating algorithm* to learn tasks that lie on a *common subspace*; this algorithm is simple and efficient
- Nonlinear kernels can be introduced via a *multi-task representer theorem*; also proposed *spectral* and *non-convex matrix learning*
- Multi-task learning can be viewed as an instance of *learning combinations of infinite kernels*
- Generally, we can use a greedy incremental algorithm to learn combinations of (finite or infinite) kernels

Future Work

- Convergence rates for the algorithms proposed
- Task relatedness can be of many types (hierarchical features, input transformation invariances etc.)
- Convex relaxation techniques for the non-convex formulations presented
- Algorithms and representer theorems for special types of norms
- Related problems (sparse coding, structured output, metric learning etc.)