GI01/4C55: Supervised Learning
Multi-Task Learning

Luca Baldassarre and Massimiliano Pontil

Department of Computer Science
University College London

Outline

1. Single-Task Review
2. Multi-Task Learning
3. Linear Multi-Task Learning
4. Non-Linear Multi-Task Learning
5. Sparsity and Structure in Multi-Task Learning
6. A simple experiment

Outline

1. Single-Task Review
2. Multi-Task Learning
3. Linear Multi-Task Learning
4. Non-Linear Multi-Task Learning
5. Sparsity and Structure in Multi-Task Learning
6. A simple experiment

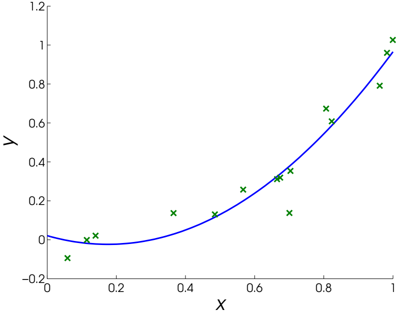
Single-Task Learning

Problem: given a set $\mathbf{z} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq X \times Y$ of i.i.d. input/output examples drawn from a fixed probability distribution, we wish to find a deterministic function

$$f : X \to Y$$

which *best* approximates the probabilistic relation between X and Y, allowing us to *predict* the output for new unseen input examples.

Example:
$y = w_0 + w_1 x + w_2 x^2 + \epsilon$
Goal is to find the parameter vector w
Difficulty: $d \gg n$
High dimensional setting



Regularization Approach

Solution: search within a "large" space of functions for a *"low complexity"* function which fits well the data

$$\min_w \sum_{i=1}^{n} \left(y_i - w^\top \phi(x_i) \right)^2 \quad + \quad \lambda \, \Omega(w)$$

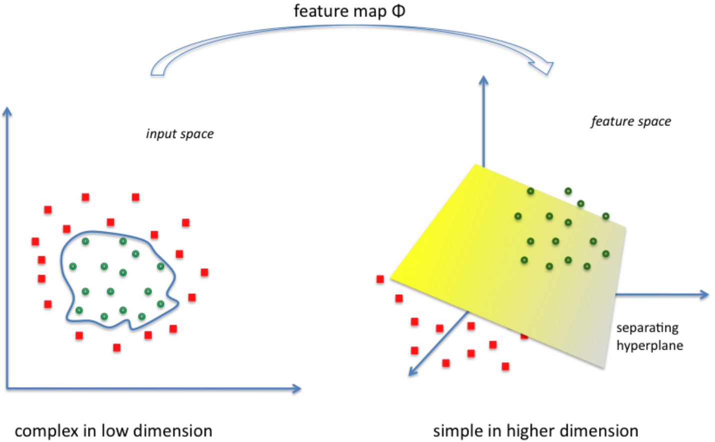
$$\text{Data Error} \quad + \quad \text{Penalty}$$

λ is called the *regularization parameter*: balances trade-off between fitting the data and choosing a simpler estimator.
Learnable and computationally efficient methods based on:

- **Smoothness:** $\Omega =$ weighted 2-norm
  (SVM, kernel methods)
- **Sparsity:** $\Omega =$ number of non-zero coefficients
  (relaxed to 1-norm, Lasso)

Kernel methods

- Use a non-linear feature map $\phi : X \to V$, with potentially $\dim(V) = +\infty$.
- Consider linear functions on the feature space: $f(x) = w^\top \phi(x)$.



Scalar kernels

- The feature map defines a kernel $K : X \times X \to \mathbb{R}$
- $K(x, x') = \phi(x)^\top \phi(x')$.
- Examples:
  - Linear: $K(x, x') = x^\top x'$
  - Polynomial: $K(x, x') = (1 + x^\top x')^d$
  - Gaussian: $K(x, x') = e^{-\frac{||x - x'||^2}{2\sigma^2}}$
- The estimator can be written as

$$f(x) = \sum_{i=1}^{n} a_i K(x, x_i)$$

- Gram matrix: $(K)_{ij} = K(x_i, x_j)$.
- $||f||_K^2 = a^\top K a$.

Outline

1. Single-Task Review
2. Multi-Task Learning
3. Linear Multi-Task Learning
4. Non-Linear Multi-Task Learning
5. Sparsity and Structure in Multi-Task Learning
6. A simple experiment

Multi-Task Learning

- What if we have *multiple* supervised tasks?

$$
\begin{aligned}
f_1 &: X \rightarrow Y \\
f_2 &: X \rightarrow Y \\
&\vdots \\
f_T &: X \rightarrow Y
\end{aligned}
$$

- Typical scenario: many tasks but only *few examples* per task
- If the tasks are related, learning them **jointly** should perform better than learning each task *independently*

Example 1: User Modeling

- Each task is to predict a user's ratings to products

CPU	CD	RAM	$\cdots$	HD	Screen	Price	Rating
1GHz	Y	1GB	$\cdots$	40G	15in	$1000	7
1GHz	N	1.5GB	$\cdots$	20G	13in	$1200	3
1.5GHz	Y	1.5GB	$\cdots$	40G	17in	$1700	5
2GHz	Y	2GB	$\cdots$	80G	15in	$2000	**?**
1.5GHz	N	2GB	$\cdots$	40G	13in	$1800	**?**

- The ways different people make decisions about products are related. *How do we exploit this?*

Example 2: Recommendation Systems

- As above, but now products are discrete objects (e.g. Netflix): ratings of products by different users
- Reformulate as a *matrix completion* problem

7	?	?	9	?
?	2	3	?	5
?	1	?	?	3
5	?	?	?	?
?	?	1	5	?

- How can we fill in the unobserved entries?

Example 3: Object Detection

- Multiple object detection in scenes: detection of each object corresponds to a classification task

- Learning common visual features enhances performance
- Character recognition: very few examples should be needed to recognize new characters

More Applications

Multi-task learning is ubiquitous

- Integration of medical / bioinformatics databases
- Robotics: learn multiple actions
- Networks: different tasks may be distributed over a (social) network
- Finance: predict multiple related stocks

Related Work

- *Neural network approach*: use a hidden layer with few nodes and a set of network weights shared by all the tasks [Baxter 96, Caruana 97, Silver and Mercer 96, etc.]
- *Hierarchical Bayes* [Bakker & Heskes 03, Lenk et al. 96, Xue et al. 07, Yu et al. 05, Zhang et al., 06 etc.]: enforce task relatedness through a common prior probability distribution on the tasks' parameters
- *Related areas*: conjoint analysis, canonical correlation analysis, longitudinal data analysis, seemingly unrelated regression (SUR) in econometrics

Objective and Questions

Learnable and computationally efficient models, which work within a high dimensional setting.

- How to model task structure ?
- What is the multi-task counterpart of smoothness/sparsity assumptions used in single-task learning?
- Pooling data across tasks?

Regularization Approach

- For each task we have a separate training set

$$\mathbf{z}_t = \{(x_{it}, y_{it})\}_{i=1}^{n_t} \subset X \times Y$$

- We can define a combined training set

$$\mathbf{z} = \{(x_i, t_i, y_i)\}_{i=1}^{n} \subset X \times \mathcal{T} \times Y$$

where $n = \sum_{t=1}^{T} n_t$ and $t_i \in \mathcal{T} = \{1, \ldots, T\}$ is the task index.

- We assume the task functions $f_1, \ldots, f_T$ to be related.
- We want to minimize

$$\sum_{t=1}^{T} \sum_{i=1}^{n} (y_{ti} - f_t(x_{ti}))^2 + \lambda\, \Omega(f_1, \ldots, f_T)$$

- The penalty term encodes the relationships among the tasks
- Other loss functions possible (e.g. SVMs)

Outline

1. Single-Task Review
2. Multi-Task Learning
3. Linear Multi-Task Learning
4. Non-Linear Multi-Task Learning
5. Sparsity and Structure in Multi-Task Learning
6. A simple experiment

- $X \subseteq \mathbb{R}^d$
- $f_t(x) = u_t^\top x$, with $u_t \in \mathbb{R}^d$, $t = 1, \ldots, T$.
- Define $u = (u_1^\top, \ldots, u_T^\top)^\top \in \mathbb{R}^{dT}$ and let $\Omega(u) = u^\top E u$.
- E is a $dT \times dT$ *symmetric positive definite* matrix, which captures the relations between the tasks.

$$R(u) = \sum_{t=1}^{T} \sum_{i=1}^{n_t} (y_{ti} - u_t^\top x_{ti})^2 + \lambda u^\top E u.$$

- **Remark.** If E is diagonal and each $d \times d$ block is a multiple of the identity, $u^\top E u = \sum_{t=1}^{T} c_t ||u_t||_2^2$

$$R(u) = \sum_{t=1}^{T} r_t(u_t)$$

  where $r_t(u_t) = \sum_{i=1}^{n_t} (y_{ti} - u_t^\top x_{ti})^2 + \lambda c_t ||u_t||_2^2$.
- The problem decouples and the task are learned **independently**.

Linear Case II

Feature space point of view:

- $f_t(x) = w^\top B_t x$;
- $w \in \mathbb{R}^p$ $(p \geq dT)$ is a common coefficient vector;
- B_t are $p \times d$ matrices which are connected to the matrix E.
- We equivalently have $u_t = B_t^\top w$.
- Since u_t are arbitrary, B_t must be full rank d for any $t = 1, \dots, T$.
- Define the $p \times dT$ matrix $B = (B_1, \dots, B_T)$.
- Assume B is also full rank dT.

Linear Case II

Feature space point of view:

- $f_t(x) = w^\top B_t x$;
- $w \in \mathbb{R}^p$ $(p \geq dT)$ is a common coefficient vector;
- B_t are $p \times d$ matrices which are connected to the matrix E.
- We equivalently have $u_t = B_t^\top w$.
- Since u_t are arbitrary, B_t must be full rank d for any $t = 1, \ldots, T$.
- Define the $p \times dT$ matrix $B = (B_1, \ldots, B_T)$.
- Assume B is also full rank dT.

Linear Multi-Task Kernel:

- The real-valued function $f(x, t) = w^\top B_t x$ has squared norm $w^\top w$.
- The Hilbert space of all such functions has the reproducing kernel

$$Q((x, t), (x', t')) = x^\top B_t^\top B_{t'} x'.$$

- The learning problem can be rewritten as:

$$S(w) = \sum_{t=1}^{T} \sum_{i=1}^{n_t} (y_{it} - w^\top B_t x_{it})^2 + \lambda w^\top w.$$

Linear Case III

$$R(u) = \sum_{t=1}^{T}\sum_{i=1}^{n_t}(y_{ti} - u_t^\top x_{ti})^2 + \lambda u^\top E u$$

$$S(w) = \sum_{t=1}^{T}\sum_{i=1}^{n_t}(y_{it} - w^\top B_t x_{it})^2 + \lambda w^\top w$$

The problems are related $S(w) = R(B^\top w)$

- Given B full rank, let $E = (B^\top B)^{-1}$
- Given E, be A a square root of E ($E = A^\top A$) and let $B = A^\top E^{-1}$
- We then have $u^\top E u = w^\top w$, with $u = B^\top w$.

Proof sketch:

- First case: $u^\top E u = wB(B^\top B)^{-1} B^\top w = w^\top w$.
- Second case: $u^\top E u = wA^\top E^{-1} E E^{-1} A w = w^\top w$.

Multi-Task Estimators

- By the representer theorem

$$w^* = \sum_{t=1}^{T} \sum_{i=1}^{n_t} c_{it} B_t x_{it}$$

- The task functions are then given by

$$\begin{aligned} f_t^*(x) &= \sum_{t'=1}^{T} \sum_{i=1}^{n_t} c_{it} Q((x,t),(x_{it},t')) \\ &= \sum_{i=1}^{n} c_i Q((x,t),(x_i,t_i)) \end{aligned}$$

- **Pooling data across the tasks**:
  Each task depends also on the examples from the other tasks.

Linear Multi-Task Kernels I

Consider the regularizer

$$\Omega(u) = u^\top E u = \sum_{t,t'=1}^{T} u_t^\top u_{t'} G_{tt'}$$

with G a $T \times T$ positive definite matrix.

$$E = \begin{pmatrix} G_{11}\mathbf{I}_d & G_{12}\mathbf{I}_d & \cdots & G_{1T}\mathbf{I}_d \\ G_{21}\mathbf{I}_d & G_{22}\mathbf{I}_d & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ G_{T1}\mathbf{I}_d & G_{T2}\mathbf{I}_d & \dots & G_{TT}\mathbf{I}_d \end{pmatrix} = G \otimes \mathbf{I}_d$$

Due to previous result, $E = (B^\top B)^{-1}$ implies $B^\top B = E^{-1} = G^{-1} \otimes \mathbf{I}_d$. The corresponding Linear Multi-Task Kernel is

$$Q((x,t),(x',t')) = x^\top B_t^\top B_{t'} x' = x^\top x' (G^{-1})_{tt'}$$

since $B_t^\top B_{t'}$ is the (t,t') block of E^{-1}, that is $(G^{-1})_{tt'}\mathbf{I}_d$.

Linear Multi-Task Kernels: Example I

$$B_t^\top = [\sqrt{1-\gamma}\mathbf{I}_d, \underbrace{\mathbf{0}, \dots, \mathbf{0}}_{t-1}, \sqrt{\gamma T}\mathbf{I}_d, \underbrace{\mathbf{0}, \dots, \mathbf{0}}_{T-t}]$$

where $\gamma \in (0,1)$ and $\mathbf{0}$ is the $d \times d$ matrix of all zero entries.

$$B_t^\top B_{t'} = (1-\gamma)\mathbf{I}_d + \gamma T \delta_{tt'} \mathbf{I}_d$$

The Linear Multi-Task Kernel is

$$Q((x,t),(x',t')) = x^\top B_t^\top B_{t'} x' = (1-\gamma+\gamma T \delta_{tt'}) x^\top x'$$

Furthermore

$$\begin{aligned} B^\top B &= [(1-\gamma)\mathbf{1}_T + \gamma T \mathbf{I}_T] \otimes \mathbf{I}_d \\ E = (B^\top B)^{-1} &= [(1-\gamma)\mathbf{1}_T + \gamma T \mathbf{I}_T]^{-1} \otimes \mathbf{I}_d \\ &= \frac{\gamma-1}{\gamma T^2}\mathbf{1}_T + \frac{1}{\gamma T}\mathbf{I}_T \otimes \mathbf{I}_d \end{aligned}$$

Linear Multi-Task Kernels: Example I Cont'd

Recall that if $E = G \otimes \mathbf{I}_d$,

$$u^\top E u = \sum_{t,t'=1}^{T} u_t^\top u_{t'} G_{tt'}.$$

In our case $G = \frac{\gamma-1}{\gamma T^2}\mathbf{1}_T + \frac{1}{\gamma T}\mathbf{I}_T$, hence

$$u^\top E u = \frac{\gamma - 1}{\gamma T^2} \sum_{t,t'=1}^{T} u_t^\top u_{t'} + \frac{1}{\gamma T} \sum_{t=1}^{T} ||u_t||_2^2$$

$$= \frac{1}{T}\left(\sum_{t=1}^{T} ||u_t||_2^2 + \frac{1-\gamma}{\gamma} \sum_{t=1}^{T} ||u_t - \frac{1}{T}\sum_{t'=1}^{T} u_{t'}||_2^2 \right)$$

where γ sets the trade-off between size and variance of the task parameters. ($\gamma = 1$: independent tasks, $\gamma \to 0$: identical tasks)

Graph regularization

- Use symmetric connectivity matrix A to enforce similarities

$$
\begin{aligned}
\Omega(u) &= \frac{1}{2}\sum_{s,t=1}^{T} A_{st}\|u_s - u_t\|^2 + \sum_{t=1}^{T}\|u_t\|^2 A_{tt} \\
&= \sum_{s,t=1}^{T}\left(||u_t||_2^2 A_{st} - u_s^\top u_t A_{st}\right) + \sum_{t=1}^{T}||u_t||_2^2 A_{tt} \\
&= \sum_{t=1}^{T}||u_t||_2^2 \sum_{s=1}^{T}(1+\delta_{st})A_{st} - \sum_{s,t=1}^{d} u_s^\top u_t A_{st} \\
&= \sum_{s,t=1}^{d} u_s^\top u_t L_{st}
\end{aligned}
$$

where $L = D - A$, with $D_{st} = \delta_{st}\left(\sum_{h=1}^{T} A_{sh} + A_{st}\right)$.

$$
Q((x,t),(x',t')) = x^\top x'(L^{-1})_{tt'}
$$

Task Clustering

Cluster tasks in r groups so as to make a partition of the tasks

$$\Omega(u) = \epsilon_1 \sum_{c=1}^{r} \sum_{t \in I(c)} ||u_t - \overline{u}_c||_2^2 + \epsilon_2 \sum_{c=1}^{r} m_c ||\overline{u}_c||_2^2.$$

where

- $I(c)$ is the set of the indexes of the tasks that belong to cluster c;
- $\overline{u}_c$ is the average of the tasks in cluster c;
- m_c is the number of tasks in cluster c.

$$\Omega(u) = \sum_{t,t'=1}^{T} u_t^\top u_{t'} G_{tt'}$$

where G depends on the cluster assignments.

Outline

1. Single-Task Review
2. Multi-Task Learning
3. Linear Multi-Task Learning
4. Non-Linear Multi-Task Learning
5. Sparsity and Structure in Multi-Task Learning
6. A simple experiment

Non-linear Multi-Task Kernels

- Non-linear extension:

$$Q((x,t),(x',t')) = K(x,x')(G^{-1})_{tt'}$$

- Regularizer:

$$\Omega(f_1,\ldots,f_T) = \sum_{t,t'=1}^{T} \langle f_t, f_{t'} \rangle_K \, G_{tt'}$$

- Gram matrix:

$$(Q)_{i,j=1}^{n} = Q((x_i,t_i),(x_j,t_j))$$

- Solution:

$$f(x,t) = \sum_{i=1}^{n} c_i Q((x,t),(x_i,t_i))$$

- **Example.** Regularized Least Squares:

$$c = (Q + n\lambda \mathbf{I})^{-1} y$$

Outline

1. Single-Task Review
2. Multi-Task Learning
3. Linear Multi-Task Learning
4. Non-Linear Multi-Task Learning
5. Sparsity and Structure in Multi-Task Learning
6. A simple experiment

Penalty Function

Define

$$W = \begin{pmatrix} | & & | \\ w_1 & \dots & w_T \\ | & & | \end{pmatrix} = \begin{pmatrix} - \, w^1 \, - \\ \vdots \\ - \, w^d \, - \end{pmatrix}$$

Consider

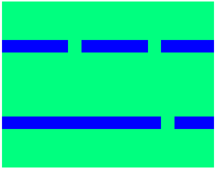
$$\min_W \sum_{t=1}^{T} \sum_{i=1}^{n} (y_{ti} - w_t^\top x_{ti})^2 + \lambda \, \Omega(W)$$

1. Quadratic: encodes closeness of task parameters
2. **Structured sparsity:** few common features

2. Structured Sparsity

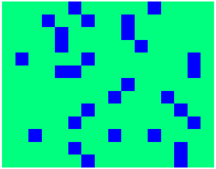
- Favour matrices with many zero rows (few features shared by the tasks)

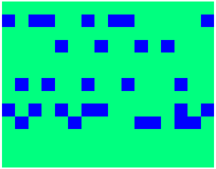
$$\Omega_{\mathrm{s}}(W) = \sum_{j=1}^{d} ||w^j||_2 = \sum_{j=1}^{d} \sqrt{\sum_{t=1}^{T} w_{tj}^2}$$

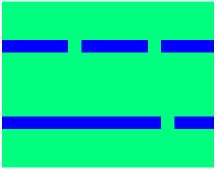


2. Structured Sparsity (cont.)

Compare matrices W favoured by different norms (green $= 0$, blue $= 1$):







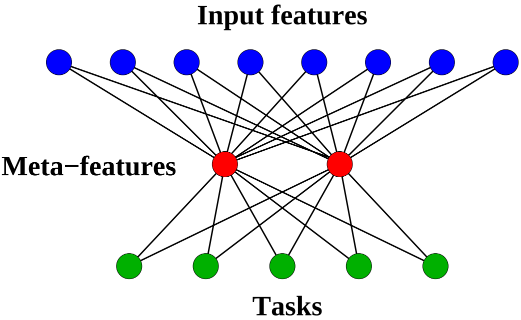
$\#\text{rows}$	$= 13$	5	3
Ω_s	$= 19$	12	8
$\sum_{tj} \lvert w_{tj} \rvert$	$= 29$	29	29

Penalty Function

$$\min_{W} \sum_{t=1}^{T} \sum_{i=1}^{n} (y_{ti} - w_t^\top x_{ti})^2 + \lambda\, \Omega(W)$$

1. Quadratic: encodes closeness of task parameters
2. Structured sparsity: few common features
3. **Spectral:** few common meta-features

- Favour matrices with low rank: $\Omega(W) = \mathrm{rank}(W)$



Intuition: task vectors w_t lie on a *low dimensional* subspace

Spectral Regularization

Recall the SVD of a matrix

$$W = U \, \mathrm{Diag}(\sigma_1, \ldots, \sigma_r) \, V^\top$$

where $U \in R^{d \times r}$ and $V \in R^{T \times r}$ are orthogonal, $r = \min(d, T)$
Approximate the rank with the trace norm:

$$\Omega_{\mathrm{tr}}(W) = \sum_{i=1}^{r} \sigma_i(W)$$

Optimization Perspective: Trace Norm

Express Ω in variational form

$$\Omega_{\mathrm{tr}}(W) = \frac{1}{2} \min_{D \succ 0} \left\{ \mathrm{tr}(W^\top D^{-1} W) + \mathrm{tr}(D) \right\}$$

$$\min_{W,\ D \succ 0} \sum_{t=1}^{T} \sum_{i=1}^{n} (y_{ti} - w_t^\top x_{ti})^2 + \frac{\lambda}{2}\ \mathrm{tr}(W^\top D^{-1} W) + \mathrm{tr}(D)$$

$$\mathrm{tr}(W^\top D^{-1} W) = \sum_{t=1}^{T} w_t^\top D^{-1} w_t = w^\top E w$$

$$E = \begin{pmatrix} D^{-1} & 0 & \cdots & 0 \\ 0 & D^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & D^{-1} \end{pmatrix}$$

Jointly convex problem in W and D.

Related to problem of **learning the kernel**.

Alternating Minimization Algorithm

- W-minimization: solve T independent regularization problems (e.g. SVM, ridge regression, etc.)
- D-minimization: can be solved analytically (via an SVD)

$$D(W) = \frac{(WW^\top)^{\frac{1}{2}}}{\mathrm{tr}(WW^\top)^{\frac{1}{2}}}$$

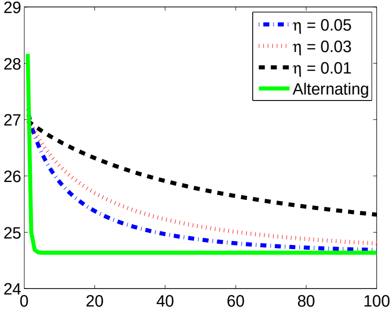
Theorem. By introducing a small perturbation

$$D(W) = \frac{(WW^\top + \varepsilon \mathbf{I})^{\frac{1}{2}}}{\mathrm{tr}(WW^\top + \varepsilon \mathbf{I})^{\frac{1}{2}}}$$

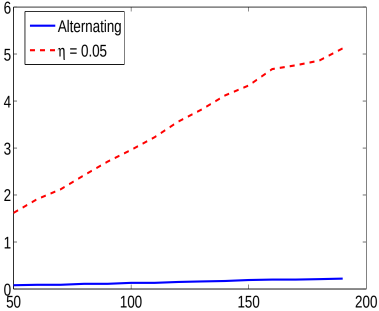
we can show that the algorithm converges to the optimal solution.

Alternating Minimization

Objective function vs. #iterations



Time [s] vs. #tasks



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

Outline

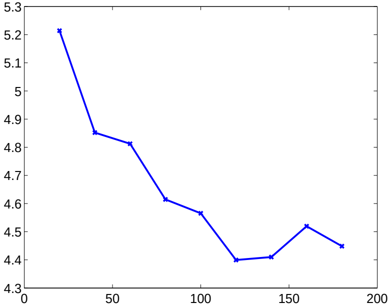
1. Single-Task Review
2. Multi-Task Learning
3. Linear Multi-Task Learning
4. Non-Linear Multi-Task Learning
5. Sparsity and Structure in Multi-Task Learning
6. A simple experiment

Experiment (Computer Survey)

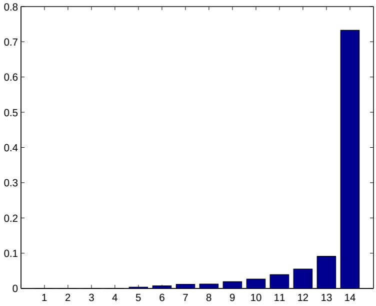
- Consumers' ratings of products [Lenk et al. 96]
- 180 persons (tasks)
- 8 PC models (training examples); 4 PC models (test examples)
- 13 binary input features (RAM, CPU, price etc.) $+$ bias term
- Integer output in $\{0, \ldots, 10\}$ (likelihood of purchase)
- The square loss was used

Experiment (Computer Survey)

Test error vs. #tasks

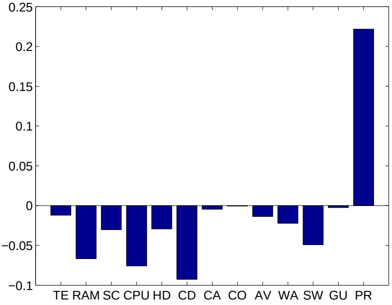


Eigenvalues of matrix D



- Performance improves with more tasks
- A single most important feature shared by everyone

Experiment (Computer Survey)



Method	Test
Independent	15.05
Aggregate	5.52
Structured Sparsity	4.04
Trace norm	3.72
Quadratic + Trace	3.20

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

Extensions / Not-covered

- Convergence rates for the algorithms
- Statistical Analysis
- Additional structured sparsity constraints
- Hierarchical models
- Connections to vector-valued learning and multi-class classification
- Use of unlabeled data / semi-supervised learning
- Other multi-task structures / applications

Material I

[Lenk, DeSarbo, Green, Young] **Hierarchical Bayes conjoint analysis: recovery of partworth heterogeneity from reduced experimental designs.** Marketing Science 1996
[Caruana] **Multi–task learning.** JMLR 1997
[Baxter] **A model for inductive bias learning.** JAIR 2000
[Ben-David, Schuller] **Exploiting task relatedness for multiple task learning.** COLT 2003
[Jebara] **Multi-task feature and kernel selection for SVMs.** ICML 2004
[Torralba, Murphy, Freeman] **Sharing features: efficient boosting procedures for multiclass object detection.** CVPR 2004
[Srebro, Rennie, Jaakkola] **Maximum-margin matrix factorization.** NIPS 2004
[Evgeniou and Pontil] **Regularized multi-task learning.** SIGKDD 2004
[Ando Zhang] **A framework for learning predictive structures from multiple tasks and unlabeled data.** JMLR 2005
[Micchelli and Pontil] **On learning vector-valued functions.** Neural Computation 2005
[Evgeniou, Micchelli, Pontil] **Learning multiple tasks with kernel methods.** JMLR 2005

Material II

[Yu, Tresp, Schwaighofer] **Learning Gaussian processes from multiple tasks.** ICML 2005

[Argyriou, Evgeniou, Pontil] **Multi-task feature learning.** NIPS 2006

[Maurer] **Bounds for linear multi-task learning.** JMLR 2006

[Argyriou, Micchelli, Pontil, Ying] **A spectral regularization framework for multi-task structure learning.** NIPS 2007

[Caponnetto, Micchelli, Pontil, Ying] **Universal mult-task kernels.** JMLR 2008

[Argyriou, Evgeniou, Pontil] **Convex multi-task feature learning.** Mach. Lear. 2008

[Argyriou, Micchelli, Pontil] **When is there a representer theorem? Vector versus matrix regularizers.** JMLR 2009

[Lounici, Pontil, Tsybakov, van de Geer] **Taking advantage of sparsity in multi-task learning.** COLT 2009

[Argyriou, Micchelli, Pontil] **On Spectral Learning.** JMLR 2010