Sparsity and Matrix Estimation

Dimitris Stamos
Slides by Massimiliano Pontil

University College London

Today's Plan

- Sparsity in linear regression
- Formulation as a convex program – Lasso
- Group Lasso
- Matrix estimation problems (Collaborative Filtering, Multi-task Learning, Inverse Covariance, Sparse Coding, etc.)
- Nonlinear extension

L1-regularization

Least absolute shrinkage and selection operator (LASSO):

$$\min_{\|w\|_1 \leq \alpha} \frac{1}{2}\|y - Xw\|_2^2$$

where $\|w\|_1 = \sum_{j=1}^{d} |w_j|$

ℓ_1-norm regularization encourages sparsity

Consider the case $X = I$:

$$\min_{w \in \mathbb{R}^d} \frac{1}{2}\|w - y\|_2^2 + \lambda \|w\|_1$$

Lemma: Let $H_\lambda(t) = (|t| - \lambda)_+ \mathrm{sgn}(t),\ t \in \mathbb{R}$. The solution $\hat{w}$ is given by

$$\hat{w}_i = H_\lambda(y_i), \quad i = 1, \ldots, d$$

Proof: First note that the problem decouples: $\hat{w}_i = \mathrm{argmin}\left\{\frac{1}{2}(w_i - y_i)^2 + \lambda|w_i|\right\}$. By symmetry $\hat{w}_i y_i \geq 0$, thus w.l.o.g. we can assume $y_i \geq 0$. Now, if $\hat{w}_i > 0$ the objective function is differentiable and setting the derivative to zero gives $\hat{w}_i = y_i - \lambda$. Since the minimum is unique we conclude that $\hat{w}_i = (y_i - \lambda)_+$.

Optimality conditions

Directional derivative of f at w in the direction d

$$D^+ f(w; d) := \lim_{\epsilon \to 0^+} \frac{f(w + \epsilon d) - f(w)}{\epsilon}$$

Theorem 1: $\hat{w} \in \arg\min_{w \in \mathbb{R}^d} f(w)$ iff $D^+ f(\hat{w}; d) \geq 0 \ \forall d \in \mathbb{R}^d$

- the directional derivative of a convex function is always well defined and finite
- if f is differentiable then at w then $D^+ f(w; d) = d^\top \nabla f(w)$ and Theorems says that $\hat{w}$ is a solution iff $\nabla f(\hat{w}) = 0$

Optimality conditions (cont.)

If f is convex its subdifferential at w is defined as

$$\partial f(w) = \{u : f(v) \geq f(w) + u^\top (v - w),\ \forall v \in \mathbb{R}^d\}$$

- ∂f is a set-valued function
- the elements of $\partial f(w)$ are called the subgradients of f at w
- intuition: $u \in \partial f(w)$ if the affine function $f(w) + u^\top (v - w)$ is a global underestimator of f

Theorem 2: $\hat{w} \in \arg\min_{w \in \mathbb{R}^d} f(w)$, iff $0 \in \partial f(\hat{w})$

Optimality conditions (cont.)

Theorem 2: $\hat{w} \in \arg\min_{w \in \mathbb{R}^d} f(w)$, iff $0 \in \partial f(\hat{w})$

- if f is differentiable then $\partial f(w) = \{\nabla f(w)\}$ and Theorem 2 says that $\hat{w}$ is a solution iff $\nabla f(\hat{w}) = 0$

Some properties of gradients are still true for subgradients, e.g:

- $\partial(af)(w) = a\partial f(w)$, for all $a \geq 0$
- If f and g are convex then $\partial(f+g)(w) = \partial f(w) + \partial g(w)$

Optimality conditions for Lasso

$$\min \|y - Xw\|_2^2 + \lambda\|w\|_1$$

- by Theorem 2 and the properties of subgradients, w is a optimal solution iff
$$X^\top(y - Xw) \in \lambda\partial\|w\|_1$$
- to compute $\partial\|w\|_1$ use the sum rule and the subgradient of the absolute value: $\partial|t| = \{\mathrm{sgn}(t)\}$ if $t \neq 0$ and $\partial|t| = \{u : |u| \leq 1\}$ if $t = 0$

Case $X = I$: $\hat{w}$ is a solution iff, for every $i = 1, \ldots, d$, $y_i - \hat{w}_i = \lambda\mathrm{sgn}(\hat{w}_i)$ if $\hat{w}_i \neq 0$ and $|y_i - \hat{w}_i| \leq \lambda$ otherwise (verify that these formulae yield the soft thresolding solution on page 4)

General learning method

In generally we will consider optimization problems of the form

$$\min_{w\in\mathbb{R}^d} F(w), \quad \text{where } F(w) = f(w) + g(w)$$

Often f will be a data term: $f(w) = \sum_{i=1}^m E(w^\top x_i, y_i)$, and g a convex penalty function (non necessarily smooth, e.g. the ℓ_1 norm)
We will later discuss a method to solve the above problem under the assumptions that f has some smoothness property and g is "simple", in the sense that the following problem is easy to solve

$$\min_w \frac{1}{2}\|w-y\|^2 + g(w)$$

Group Lasso

Enforce sparsity across a-priori known groups of variables:

$$\min_{W \in \mathbb{R}^d} f(w) + \lambda \sum_{\ell=1}^{N} \|w_{|J_\ell}\|_2$$

where $J_1, \ldots, J_N$ are prescribed subsets of $\{1, \ldots, d\}$

- In the original formulation (Yuan and Lin, 2006) the groups form a partition of the index set $\{1, \ldots, n\}$
- Overlapping groups (Zhao et al. 2009; Jennaton et al. 2010): hierarchical structures such as DAGS
  Example: $J_1 = \{1, 2, \ldots, d\}, J_2 = \{2, 3, \ldots, d\}, \ldots, J_d = \{d\}$

Multi-task learning

- Learning multiple linear regression or binary classification tasks simultaneously
- Formulate as a matrix estimation problem ($W = [w_1, \ldots, w_T]$)

$$\min_{W \in \mathbb{R}^{d \times T}} \sum_{t=1}^{T} \sum_{i=1}^{m} E(w_t^\top x_{ti}, y_{ti}) + \lambda g(W)$$

- Relationships between tasks modeled via sparsity constraints on W
- Few common important variables (special case of Group Lasso):

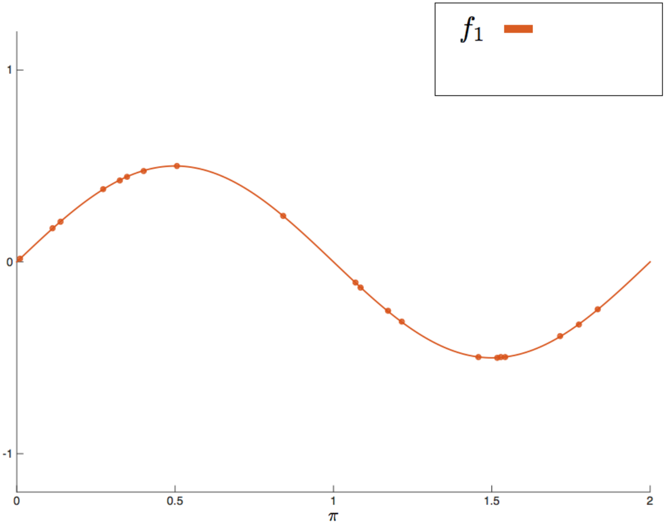
$$g(W) = \sum_{j=1}^{d} \|w^j\|_2$$

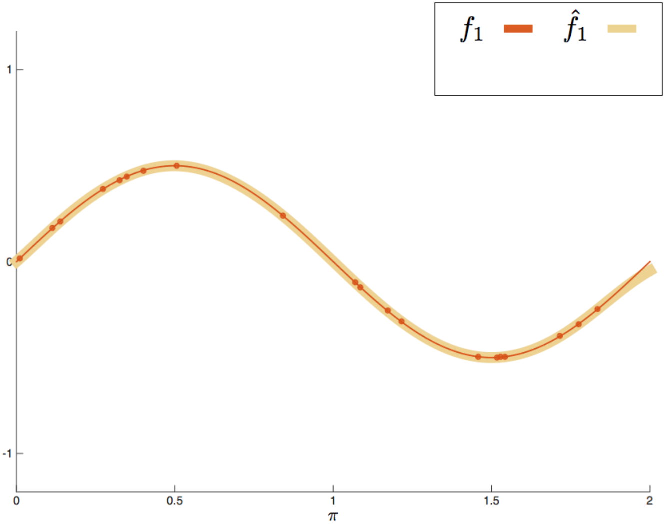
Some references

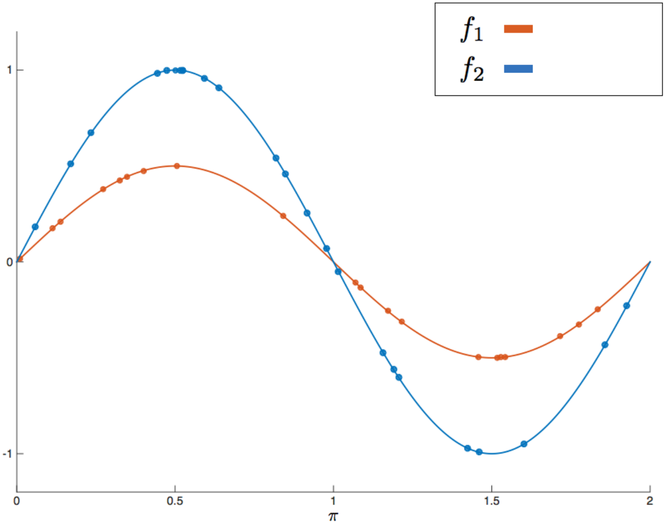
- **Lasso:**
  - P.J. Bickel, Y. Ritov, and A.B. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Annals of Statistics*, 37:1705–1732, 2009.
  - R. Tibshirani. Regression Shrinkage and Selection via the Lasso, *J. Royal Statistical Society B*, 58(1):267–288, 1996.
- **Group Lasso:**
  - M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables *Journal of the Royal Statistical Society, Series B*, 68(1):49–67, 2006.
  - P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 37(6A):3468–3497, 2009.
  - R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. arXiv:0904.3523v2, 2009.
- **Multi-task learning:**
  - A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.
  - G. Obozinski, B. Taskar, and M.I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):1–22, 2010.

Multitask learning (MTL)

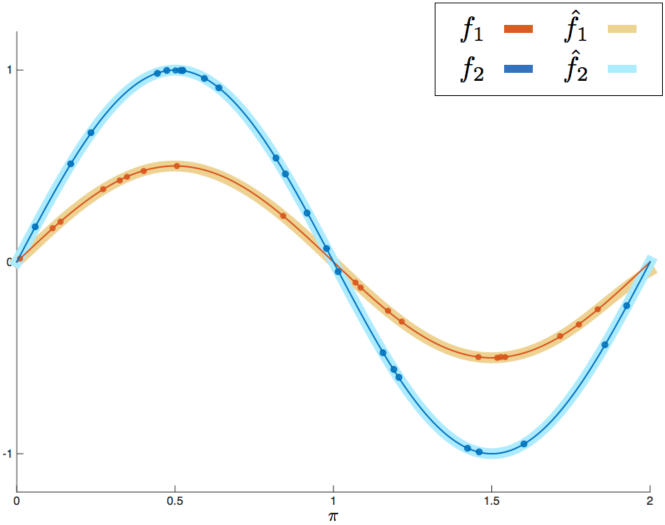
- Framework for solving a collection of related learning problems simultaneously
- When problems (tasks) are closely related, learning in parallel can be more efficient than learning tasks independently
- Example: Learning a set of linear classifiers for related objects (cars, lorries, bicycles)
- Further categorization is possible, e.g. hierarchical models, clustering of tasks

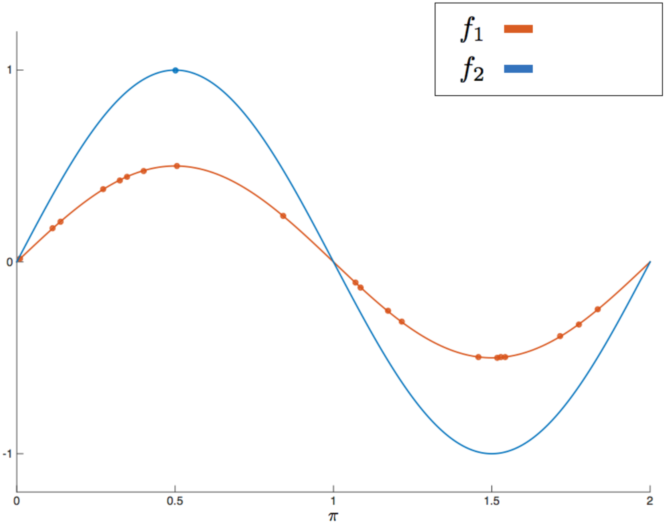

f_1
1
0
-1
0
0.5
1
1.5
2
π

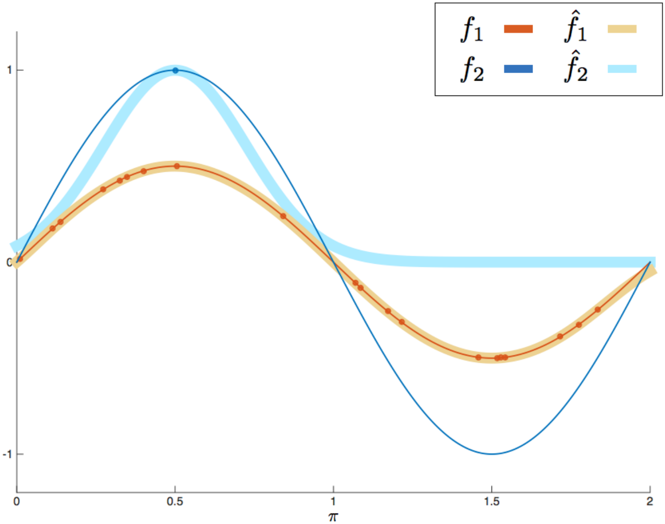

f_1
$\hat{f}_1$
1
0
-1
0
0.5
1
1.5
2
π



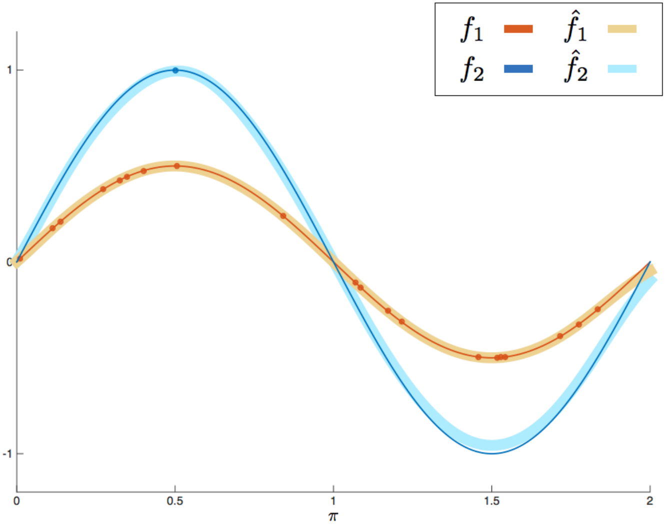
f_1
f_2
1
0
-1
0
0.5
1
1.5
2
π


f_1
$\hat{f}_1$
f_2
$\hat{f}_2$
1
0
-1
0
0.5
1
1.5
2
π


f_1
f_2
1
0
-1
0
0.5
1
1.5
2
π



Without Sharing Information



Sharing Information

Multitask learning

- Fix probability measures $\mu_1, \ldots, \mu_T$ on $\mathbb{R}^d \times \mathbb{R}$
- Draw data: $(x_{t1}, y_{t1}), \ldots, (x_{tn}, y_{tn}) \sim \mu_t, \;\; t = 1, ..., T$
  (in practice n may vary with t)
- Learning method: $\displaystyle \min_{(f_1,...,f_T) \in \mathcal{F}} \;\; \frac{1}{T} \sum_{t=1}^{T} \frac{1}{n} \sum_{i=1}^{n} \ell(y_{ti}, f_t(x_{ti}))$
- $\mathcal{F}$ is a set of vector-valued functions. A standard choice is a ball in a (reproducing kernel) Hilbert space. This provides a means to model interactions between the tasks in that functions with small norm have strongly related components
- Goal is to minimize the multitask error

$$R(f_1, ..., f_T) = \frac{1}{T} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(x,y) \sim \mu_t} \ell(y, f_t(x))$$

Linear MTL

- "task" = "linear model"
  - Regression: $y_{ti} = \langle w_t^*, x_{ti} \rangle + \epsilon_{ti}$
  - Binary classification: $y_{ti} = \operatorname{sign}\langle w_t^*, x_{ti} \rangle \epsilon_{ti}$
- Learning method: $\displaystyle \min_{[w_1,\ldots,w_T] \in \mathcal{S}} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{n} \sum_{i=1}^{n} \ell(y_{ti}, \langle w_t, x_{ti} \rangle)$
- Set $\mathcal{S}$ encourages "common structure" among tasks, e.g. the ball of a matrix norm or other regularizer
- Independent task learning (ITL): $\mathcal{S} = \underbrace{\mathcal{B} \times \cdots \times \mathcal{B}}_{\text{T times}}$

Linear MTL (cont.)

$$\min_{[w_1,\dots,w_T]\in\mathcal{S}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\ell(y_{ti},\langle w_t, x_{ti}\rangle)$$

- Typical scenario: **many tasks** but only **few examples per task**. If $n < d$ we don't have enough data to learn the tasks one by one. However if the tasks are *"related"* and set $\mathcal{S}$ or the associated regularizer captures such relationships in a simple way, learning the tasks *jointly* greatly improves over ITL.

Applications

- **User modelling:**
  - each task is to predict a user's ratings of products
  - the ways different people make decisions about products are related

- **Multiple object detection in scenes:**
  - detection of each object corresponds to a binary classification task: $y_{ti} \in \{-1, 1\}$
  - learning common features enhances performance
  - early work in using multilayer neural networks (now called deep networks) with shared hidden weights

Many more: affective computing, bioinformatics, health informatics, marketing science, neuroimaging, NLP, speech,...

Goal

The multitask error of $W = [w_1, \dots, w_T]$ is

$$R(W) = \frac{1}{T} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(x,y)\sim\mu_t} \ell(y, \langle w_t, x \rangle)$$

It is possible to give bounds on the uniform deviation

$$\sup_{W \in \mathcal{S}} \left\{ R(W) - \frac{1}{T} \sum_{t=1}^{T} \frac{1}{n} \sum_{i=1}^{n} \ell(y_{ti}, \langle w_t, x_{ti} \rangle) \right\}$$

and derive bounds for the excess error

$$R(\hat{W}) - \min_{W \in \mathcal{S}} R(W)$$

Regularizers for linear MTL

$$\min_{[w_1,\ldots,w_T]\in\mathcal{S}} \; \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\ell(y_{ti},\langle w_t, x_{ti}\rangle)$$

Often we drop the constraint and consider penalty methods

$$\min_{w_1,\ldots,w_T} \; \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\ell(y_{ti},\langle w_t, x_{ti}\rangle) + \lambda\Omega(w_1,\ldots,w_T)$$

Regularizers for linear MTL

Different regularizers encourage different types of commonalities between the tasks:

- Variance (or other convex quadratic regularizers) encourages closeness to the mean
- Joint sparsity (or other structured sparsity regularizers) encourages few shared variables
- Trace norm (or other spectral regularizers which promote low rank solutions) encourages few shared features
- More sophisticated regularizers which combine the above, promote clustering of tasks, etc.

Quadratic regularizer

$$\Omega(W) = \sum_{s,t=1}^{T} \langle w_s, E_{st} w_t \rangle$$

where the matrix $E = (E_{st})_{s,t=1}^{T} \in \mathbb{R}^{dT \times dT}$ is positive definite

- Example: let $\gamma \in [0, 1]$ and

$$\Omega_{\mathrm{Var}}(W) = \frac{1}{T} \sum_{t=1}^{T} \|w_t\|^2 + \frac{1-\gamma}{\gamma} \mathrm{Var}(w_1, ..., w_T)$$

  $\gamma = 1$: independent tasks; $\gamma = 0$: identical tasks

- Regularizer favours weight vectors which are close to the mean

Variance regularizer

$$\min_{w_1,\dots,w_T} \frac{1}{Tn}\sum_{t,i}\ell(y_{ti},\langle w_t, x_{ti}\rangle) + \lambda\left(\frac{1}{T}\sum_{t=1}^{T}\|w_t\|^2 + \frac{1-\gamma}{\gamma}\mathrm{Var}(w_1,\dots,w_T)\right)$$

is equivalent to

$$\min_{w_0,u_1\dots,u_T} \frac{1}{Tn}\sum_{t,i}\ell(y_{ti},\langle w_0+u_t, x_{ti}\rangle) + \lambda\left(\frac{1}{\gamma T}\sum_{t=1}^{T}\|u_t\|^2 + \frac{1}{1-\gamma}\|w_0\|^2\right)$$

To see it make the change of variable $w_t = w_0 + u_t$ and minimize over w_0

Variance regularizer (cont.)

$$\min_{w_0, u_1 \ldots, u_T} \frac{1}{Tn} \sum_{t,i} \ell(y_{ti}, \langle w_0 + u_t, x_{ti} \rangle) + \lambda \left(\frac{1}{\gamma T} \sum_t \|u_t\|^2 + \frac{1}{1-\gamma} \|w_0\|^2 \right)$$

We write the above as

$$\min_{w_0} \frac{1}{T} \sum_{t=1}^{T} \min_w \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_{ti}, \langle w, x_{ti} \rangle) + \frac{\lambda}{\gamma} \|w - w_0\|^2 \right\} + \frac{\lambda}{1-\gamma} \|w_0\|^2$$

This makes it apparent that we regularize around some common vector w_0

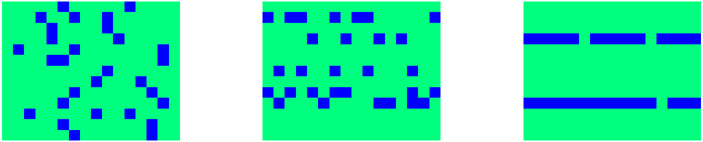
Structured sparsity: few shared variables

- Favour matrices with many zero rows:

$$\|W\|_{2,1} := \sum_{j=1}^{d} \sqrt{\sum_{t=1}^{T} w_{tj}^2}$$

- Special case of **group Lasso** method

Matrices W favoured by different regularizers ($\texttt{green} = 0$, $\texttt{blue} = 1$):



$\#$rows $=$	13	5	2
$\|\cdot\|_{2,1} =$	19	12	8
ℓ_1-norm $=$	29	29	29

Clustered MTL

- T Tasks are clustered into Q groups (e.g. classifiers for cars, lorries, bicycles and pedestrians, dogs..)
- Control mean task magnitude, between cluster variance and within cluster variance

$$\Omega_m(W) = T\|\bar{w}\|^2 = \mathrm{tr} W U W^\top$$
$$\Omega_b(W) = \sum_q T_q \|\bar{w}_q - \bar{w}\|^2 = \mathrm{tr} W (M - U) W^\top$$
$$\Omega_w(W) = \sum_{q, t \in J_q} \|w_t - \bar{w}_q\|^2 = \mathrm{tr} W (I - M) W^\top$$

where $U = 11^\top / T$, and $M_{st} = 1/T_q$ if w_s, w_t in cluster q

The k-Support Norm

- The vector k-support norm is defined by the unit ball

$$\mathbf{conv}\left\{w \in \mathbb{R}^d : \mathbf{card}(w) \leq k, \|w\|_2 \leq 1\right\}$$

- The norm can be written as

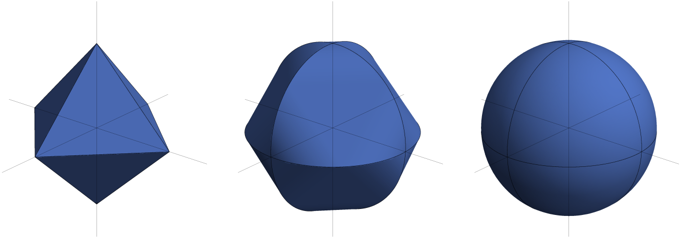
$$\|w\|_{(k)} = \inf\left\{\sum_{g \in \mathcal{G}} \|v_g\|_2 : \operatorname{supp}(v_g) \subset g, \sum_{g \in \mathcal{G}} v_g = w\right\}$$

  where $\mathcal{G}$ is the set of all subsets of $\{1, 2, \ldots, d\}$ of size k.

- Special cases: $k = 1$ (ℓ_1-norm) and $k = d$ (ℓ_2-norm)

Unit Balls

- We recognise the special cases $k = 1$ and $k = d$.



$k = 1$ $\qquad$ $k = 2$ $\qquad$ $k = 3$

- The unit balls suggest the norm is not differentiable for $k \neq d$
- For $k \neq d$ the unit ball has sharp corners at the axes, suggesting the norm may promote sparsity

Some references

- MTL, quadratic and spectral: [Maurer. The Rademacher complexity of linear transformation classes. *COLT*, 2006],

  [Cavallanti, Cesa-Bianchi, and Gentile. Linear algorithms for online multitask classification, *JMLR*, 2010]

- MTL, multitask feature learning: [Maurer and Pontil. Excess risk bounds for multitask learning with trace norm regularization. *COLT*, 2013]

- MTL, few shared variables / group Lasso: [Lounici, Pontil, Tsybakov, van de Geer. Oracle inequalities and optimal inference under group sparsity. *Annals of Statistics*, 2011], [Obozinski, Wainwright, Jordan. Support union recovery in high dimension multivariate regression. *Annals of Statistics*, 2011]

- LTL, multitask feature learning: [Maurer. Transfer bounds for linear feature learning. *Machine Learning*, 2009]

Plan

- Trace norm regularization
- Low rank matrix factorization formulation
- Algorithm and analysis
- Experiments
- Multitask learning with non-linear output constraints
- Ongoing work and planned collaborations

Matrix learning problem

Aim: learn a low rank matrix from data (e.g. known subset of entries)

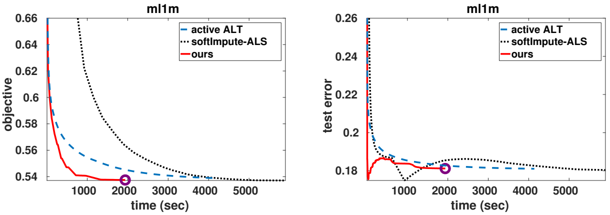
$$\min_{W \in \mathbb{R}^{n \times m}} \ell(W) + \lambda \|W\|_*$$

with ℓ an error term and $\|W\|_*$ the trace norm (sum of singular values)

Approach: solve the related optimization problem

$$\min_{A,B} \ \ell(AB^\top) + \frac{\lambda}{2}(\|A\|_F^2 + \|B\|_F^2)$$

Result: we derive an efficient algorithm that dynamically expands the parameter space, provably converging to a global optimum.



Different settings

$$\min_{W \in \mathbb{R}^{n \times m}} f_\lambda(W), \qquad f_\lambda(W) = \ell(W) + \lambda \|W\|_*$$

- Matrix completion (Srebro & Rennie, 2005). Let $M \in \mathbb{R}^{n \times m}$ be a binary "mask" and $Y \in \mathbb{R}^{n \times m}$ contains the observed entries:

$$\ell(W) = \|M \odot (Y - W)\|_F^2$$

- Multitask learning (Argyriou et al., 2006). Let $w_1, \ldots, w_m$ be the columns of W and (X_j, y_i) the dataset for the j-th task:

$$\ell(W) = \sum_{j=1}^{m} \|y_j - X_j w_j\|^2$$

- Collaborative filtering with attributes (Abernethy et al., 2009). Sampling a function $(x, z) \mapsto z^\top W x$ with noise:

$$\ell(W) = \sum_{i=1}^{N} (y_i - z_i^\top W x_i)^2$$

Matrix learning with PFB

If ℓ is **convex and differentiable**, with Lipschitz continuous gradient we can use **proximal forward-backward** (PFB) (e.g. Bauschke & Combettes, 2017)

$$W_{k+1} = \mathrm{Prox}_{\gamma\lambda\|\cdot\|_*}\big(W_k - \gamma\nabla\ell(W_k)\big), \quad k \in \mathbb{N}$$

PROs.

- Convex Problem $\Rightarrow$ converges to global minimizer
- Convergence rate $O(1/k)$ (faster with accelerated methods)

CONs.

- One SVD per iteration! $O(nm\min(n,m))$ time complexity.
- $O(nm)$ memory requirement even if the solution is low rank.

Factorization based formulation

Variational form for the nuclear norm

$$\|W\|_* = \sum_{i=1}^{n} \sigma_i = \frac{1}{2} \min \left\{ \|A\|_F^2 + \|B\|_F^2 \right\}$$
$$\text{s.t.} \quad r \in \mathbb{N}, \quad A \in \mathbb{R}^{n \times r}, \quad B \in \mathbb{R}^{m \times r}, \quad AB^\top = W$$

$\Rightarrow$ alternative problem in A and B

$$\min_{\substack{A \in \mathbb{R}^{n \times r} \\ B \in \mathbb{R}^{m \times r}}} g_{\lambda,r}(A, B), \qquad g_{\lambda,r}(A, B) = \ell(AB^\top) + \frac{\lambda}{2}(\|A\|_F^2 + \|B\|_F^2)$$

where r is now a *hyperparameter*.

Theorem. *The problems of minimizing f_λ and $g_{\lambda,r}$ are equivalent for sufficently large r.*

Matrix learning with factorization based methods

PROs.

- Smooth functional $\Rightarrow$ we can use any first or second order optimization method, e.g. Gradient Descent (GD)

$$A_{k+1} = A_k - \gamma\big(\nabla \ell(A_k B_k^\top) B_k + \lambda A_k\big)$$
$$B_{k+1} = B_k - \gamma\big(\nabla \ell(A_k B_k^\top)^\top A_k + \lambda B_k\big)$$

- $O(nmr)$ time complexity per iteration
- $O((m+n)r)$ space complexity

CONs.

- Not convex: Guarantees for global optimality?
- In practice r is unknown. How to find it?

Contributions

- We derive an efficient algorithm (both in memory and time) that dynamically expands the parameter space, leveraging the low-rank structure of the problem
- Algorithm implements a strategy to escape saddle points, provably converging to a global optimum
- We provide necessary and sufficient conditions for global optimality, which are efficiently to verify
- Algorithm signifcantly outperforms state of the art

A meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

A meta-algorithm for fast large-scale matrix learning

Let $r = 1,\ \ A'_0 \in \mathbb{R}^n,\ \ B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

A meta-algorithm for fast large-scale matrix learning

Let $r = 1,\ \ A_0' \in \mathbb{R}^n,\ \ B_0' \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A_{r-1}', B_{r-1}') to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.

A meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla \ell(A_r B_r^\top)$ is less than λ: **Stop**.

3. Perform a step in a **descent direction** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$

A meta-algorithm for fast large-scale matrix learning

Let $r = 1,\ \ A'_0 \in \mathbb{R}^n,\ \ B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).
2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.
3. Perform a step in a **descent direction** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$
4. Increase r to $r+1$ and go back to Step 1.

A meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).
2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.
3. Perform a step in a **descent direction** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$
4. Increase r to $r+1$ and go back to Step 1.

This procedure is bound to stop **at most** for $r = \min(n, m)$.
(in practice it stops much earlier)

A criterion for global optimality

Theorem. *Let* $(\bar{A}, \bar{B})$ *be a critical point for* $g_{\lambda,r}$*. Then* $\bar{W} = \bar{A}\bar{B}^\top$ *is a global minimizer for* f_λ *if and only if*

$$\sigma_{\max}\big(\ell(\bar{W})\big) \leq \lambda$$

Remarks:

- In general it is NP-hard to determine whether a critical point of a *non-convex* function is a global minimizer.
- Significantly faster than the full SVD.

A meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla \ell(A_r B_r^\top)$ is less than λ: **Stop**.

3. Perform a step in a **descent direction*** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$

4. Increase r to $r+1$ and go back to Step 1.

This procedure is bound to stop **at most** for $r = \min(n, m)$.
(in practice it stops much earlier)

Escaping from critical points

Theorem. *Let (A, B) be a critical point of $g_{\lambda,r}$ with $\sigma_{\max}(\nabla\ell(AB^\top)) > \lambda$ (i.e. not a global minimizer). If $\mathrm{rank}(A) < r$ or $\mathrm{rank}(B) < r$ then (A, B) is a* **strict saddle** point *(i.e. the corresponding Hessian has at least one negative eigenvalue)*.

- For $r > \min(n, m)$ it is always true that $\mathrm{rank}(A) < r \Rightarrow$ every critical point is either a global minimizer or a strict saddle!
- But *GD does not converge to strict saddle points* (Lee et al. 2016).

Corollary. *For $r > \min(n, m)$, gradient descent applied to $g_{\lambda,r}$* **converges only to global minimizers**.

A meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.

3. Perform a step in a **descent direction*** for $g_{\lambda,r+1}$ from $([A_r\ 0],[B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$

4. Increase r to $r+1$ and go back to Step 1.

This procedure is bound to stop **at most** for $r = \min(n, m)$.
(in practice it stops much earlier)

Escaping from critical points

Corollary. *Let u and v be any two left and right singular vectors of $\nabla \ell(AB^\top)$ associated to a singular values* larger than λ. *Then, for any any $q \in \mathbb{R}^r$ for which $Aq = 0$, we have $Bq = 0$ (and vice-versa). Moreover, if $\operatorname{rank}(A) < r$, then $(uq^\top, vq^\top)$ is a* descent direction *for $g_{\lambda,r}$ at (A, B).*

- If A and B are not full rank, we can explicitly find an escape direction!
- If A and B are full rank, then $[A,\ 0], [B,\ 0]$:
  - is a critical point for $g_{\lambda,r}$
  - is not full rank
- We can "inflate" the problem by one column, from r to $r+1$, and find an escape direction!

A meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla \ell(A_r B_r^\top)$ is less than λ: **Stop**.

3. Perform a step in a **descent direction*** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$

4. Increase r to $r+1$ and go back to Step 1.

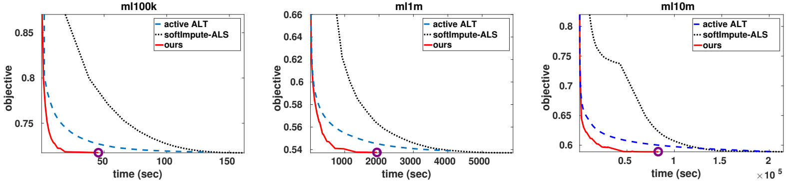
This procedure is bound to stop **at most** for $r = \min(n, m)$.
(in practice it stops much earlier)

Experiments - Movielens

- Movies rated $(1$ to $5)$ by users. Large scale matrix factorization:

$$\ell(W) = \|M \odot (Y - AB^\top)\|_F^2$$

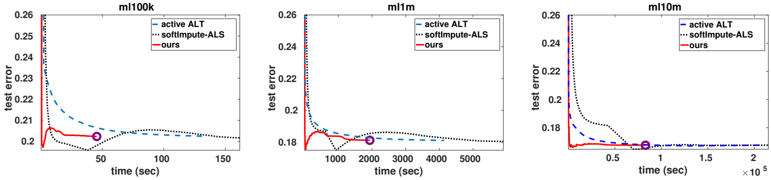
- 50% user ratings for training, 25% validation, 25% test.
- Movielens 100k (ml100k), 1M (ml1m) and 10M (ml10m).



Competitors: Active ALT (Hsieh et al. 2014), ALS Soft-impute (Hastie et al. 2015)

Experiments - Movielens (test error)

Normalized Mean Absolute Error (NMAE): mean of the entry-wise absolute errors normalized by the maximum discrepancy $\max_{i,j}(Y_{ij}) - \min_{i,j}(Y_{ij})$.

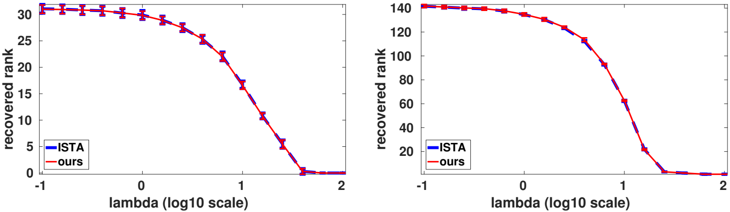


	ml100k			ml1m			ml10m		
	NMAE	time(s)	rank	NMAE	time(s)	rank	NMAE	time(s)	rank
ALT	0.2165	97	93	0.1806	4133	179	0.1670	205023	225
ALS-SI	0.1956	40	16	0.1749	832	31	0.1648	51205	36
Ours	0.1959	2	11	0.1751	39	25	0.1659	3150	41

Experiments - recovered rank

When does the condition for global optimality activate?

- **Synthetic (Left)**: random 100×100 matrix $Y = AB^\top + E$, product of two 100×10 matrices plus Gaussian noise E on the entries.
- **Movielens 100k (Right)**: 943×1682 matrix with 100k available ratings.



Comparison with rank recovered by Proximal method (ISTA).