A Fast Augmented Lagrangian Algorithm for Learning Low-Rank Matrices

Ryota Tomioka, Taiji Suzuki, Masashi Sugiyama, Hisashi Kashima

2010/8/18
ICML2010読む会

お話したいこと

* 低ランク行列学習はいろいろ応用がある
* 提案法 (M-DAL) は速くて理論的な性能保証のある実用的なアルゴリズム

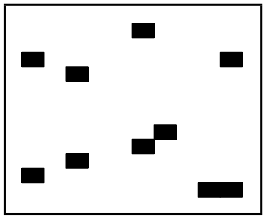
ソフトウェア：

http://www.ibis.t.u-tokyo.ac.jp/RyotaTomioka/Softwares/SupportPageICML10

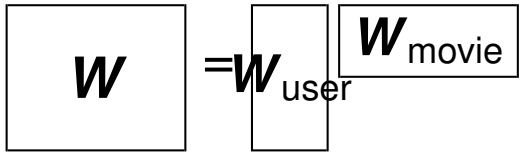
低ランク学習

1. Matrix completion [Srebro et al. 05; Abernethy et al. 09] (collaborative filtering, link prediction)

$$\boldsymbol{Y} = \boldsymbol{W}$$

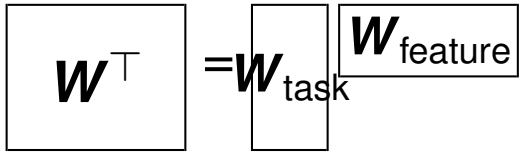


Part of $\boldsymbol{Y}$ is observed



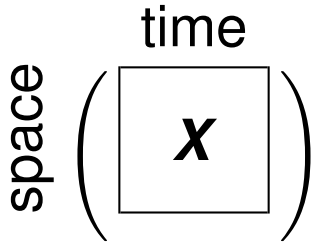
2. Multi-task learning [Argyriou et al., 07]

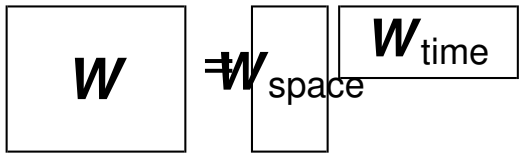
$$\boldsymbol{y} = \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b} \qquad \boldsymbol{W}^\top = \begin{bmatrix} \boldsymbol{w}_{\text{task 1}}{}^\top \\ \boldsymbol{w}_{\text{task 2}}{}^\top \\ \vdots \\ \boldsymbol{w}_{\text{task R}}{}^\top \end{bmatrix}$$



3. Predicting over matrices (classification/regression) [Tomioka & Aihara, 07]

$$y = \langle \boldsymbol{W}, \boldsymbol{X} \rangle + b$$





低ランク化と特徴抽出

$$f(X) = \langle W, X \rangle + b$$

$$= \mathrm{Tr}\left(W^\top X\right) + b$$

$$= \mathrm{Tr}\left(\sum_{j=1}^{r} \sigma_j \boldsymbol{v}_j \boldsymbol{u}_j{}^\top X\right) + b$$

$$= \sum_{j=1}^{r} \sigma_j \mathrm{Tr}\left(\boldsymbol{v}_j \boldsymbol{u}_j{}^\top X\right) + b$$

$$= \sum_{j=1}^{r} \sigma_j \mathrm{Tr}\left(\boldsymbol{u}_j{}^\top X \boldsymbol{v}_j\right) + b$$

$$\left(W = \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j{}^\top\right)$$

特徴抽出

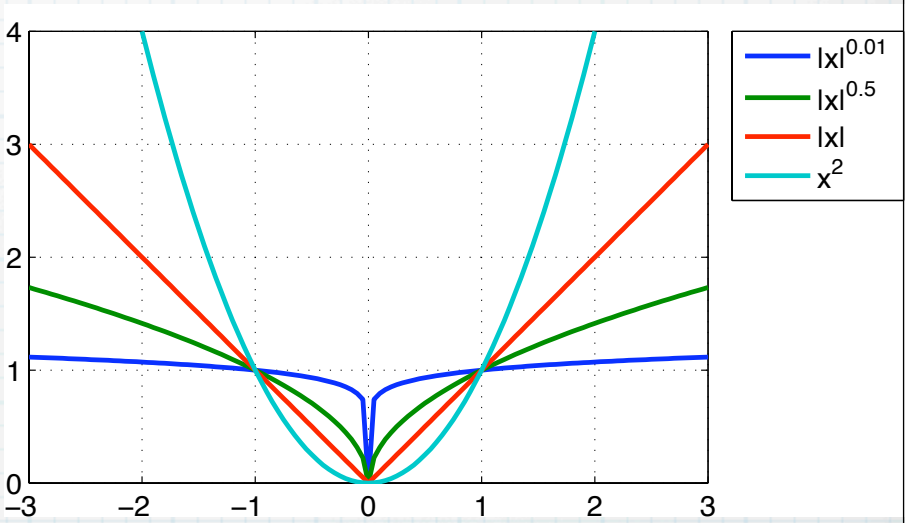
時間フィルタ

空間フィルタ

行列のランクの凸近似：トレースノルム

- L1-ノルム：ベクトルの非ゼロ成分の個数の凸近似

$$\|\boldsymbol{w}\|_1 = \sum_{j=1}^{n} |w_j|$$



- トレースノルム：行列のランク（＝非ゼロ特異値の数）の凸近似

$$\|\boldsymbol{W}\|_* = \sum_{j=1}^{r} \sigma_j(\boldsymbol{W})$$

$\sigma_j(\boldsymbol{W})$ は W の j 番目に大きい特異値

最適化問題(従来法)

$$\underset{\boldsymbol{W}\in\mathbb{R}^{R\times C}}{\text{minimize}} \quad L(\boldsymbol{W}) + \lambda\|\boldsymbol{W}\|_*$$

損失項(データ)　正則化項

- Proximal (accelerated) gradient [Ji & Ye, 09]
  - 途中の解も低ランク
  - データのスケーリングが悪いと遅い
  - 収束性は O(1/k^2) : First-order black-box の方法としては最適
- 内点法 [Tomioka & Aihara, 07]
  - スケーリングにロバスト
  - 収束性は O(exp(-k))

スケーリングにロバストで途中の解が低ランク
なアルゴリズムが欲しい! → M-DAL(提案法)

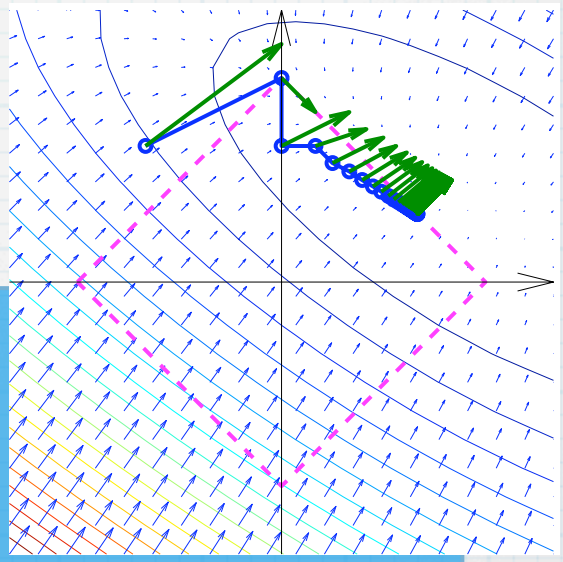
従来法: Proximal Gradient

1. W^0を適当に初期化. 適当に$\eta_1, \eta_2, \ldots$ を選ぶ.

2. 停止基準が満たされるまで

$$W^{t+1} := \operatorname*{argmin}_{\boldsymbol{W}} \left(\underbrace{\langle \boldsymbol{W}, \nabla L(\boldsymbol{W}^t) \rangle}_{\text{ロス項の線形近似}} + \underbrace{\lambda \|\boldsymbol{W}\|_*}_{\text{正則化項}} + \underbrace{\frac{1}{2\eta_t} \|\boldsymbol{W} - \boldsymbol{W}^t\|_{\text{fro}}^2}_{\text{Proximity 項}} \right)$$

$$= \underbrace{\mathrm{ST}_{\lambda\eta_t}}_{\text{縮小}} \left(\underbrace{\boldsymbol{W}^t - \eta_t \nabla L(\boldsymbol{W}^t)}_{\text{勾配ステップ}} \right)$$



Spectral Soft-threshold:

$$\mathrm{ST}_\lambda(\boldsymbol{W}) = \boldsymbol{U} \max(\boldsymbol{S} - \lambda \boldsymbol{I}, 0) \boldsymbol{V}^\top$$

ただし、特異値分解

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$$

良い点:
- シンプル

悪い点:
- スケールにロバストでない

最適化問題(提案法)

$$\underset{\boldsymbol{W}\in\mathbb{R}^{R\times C}}{\text{minimize}} \quad f_\ell(\boldsymbol{A}\mathrm{vec}(\boldsymbol{W})) + \lambda\|\boldsymbol{W}\|_*$$

損失項　　正則化項

データ

- f_ell は損失関数(ロジスティックなど)でそれなりに性質がよいとする
- A はデザイン行列(サンプル数x未知変数の数)で性質が悪いかも
- f_ell と A を区別する → A (データ) の性質によらない収束性の評価

例:多入力多出力回帰

$$L(\boldsymbol{W}) = \sum_{i=1}^{m} \|\boldsymbol{y}_i - \boldsymbol{W}\boldsymbol{x}_i\|^2$$

$$= \|\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{X}\|_{\mathrm{fro}}^2$$

$$= \left\|\mathrm{vec}(\boldsymbol{Y}) - \underbrace{(\boldsymbol{X}^\top \otimes \boldsymbol{I}_R)}_{\text{デザイン行列 A}}\mathrm{vec}(\boldsymbol{W})\right\|^2 \quad \text{ロス関数}$$

$$\left(\mathrm{vec}(AXB) = (B^\top \otimes A)\mathrm{vec}(X)\right)$$

クロネッカー積

* 他のロスでも同様

提案法: Dual Augmented-Lagrangian

1. W^0を適当に初期化. $\eta_1<\eta_2<\ldots$ を選ぶ
2. 停止基準が満たされるまで

$$\boldsymbol{W}^{t+1} := \operatorname*{argmin}_{\boldsymbol{W}} \left(\underbrace{f_\ell(\boldsymbol{A}\mathrm{vec}(\boldsymbol{W}))}_{\text{ロス項(線形近似なし)}} + \underbrace{\lambda\|\boldsymbol{W}\|_*}_{\text{正則化項}} + \underbrace{\frac{1}{2\eta_t}\|\boldsymbol{W}-\boldsymbol{W}^t\|^2}_{\text{Proximity 項}} \right)$$

$$= \mathrm{ST}_{\lambda\eta_t}\left(\boldsymbol{W}^t + \eta_t \boldsymbol{A}^\top \boldsymbol{\alpha}^{t+1}\right)$$

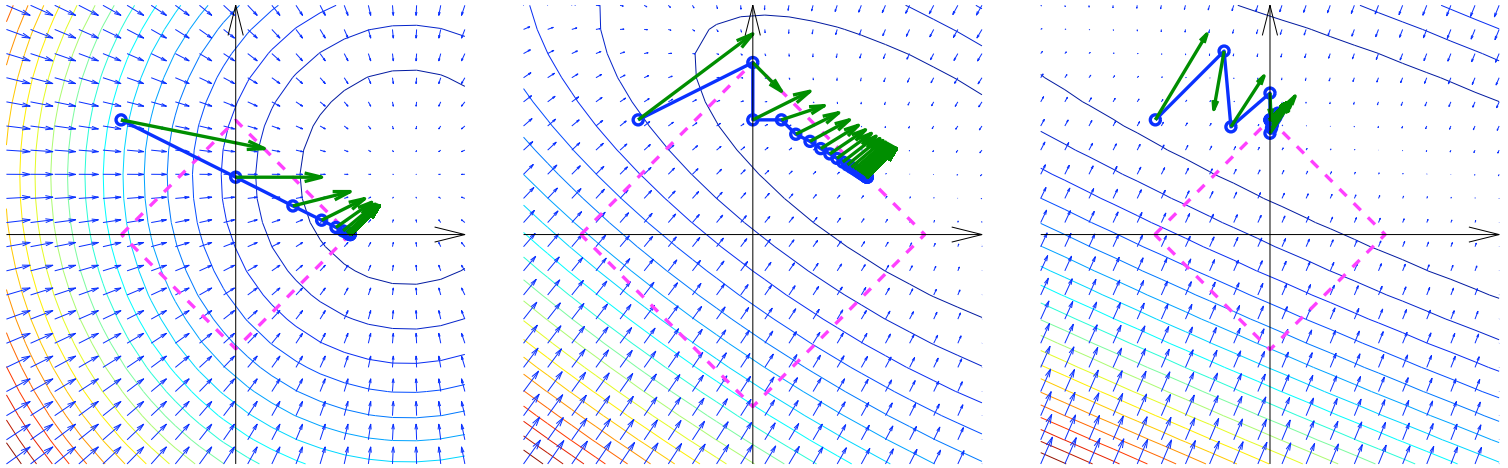
ただし $\boldsymbol{\alpha}^{t+1} = \operatorname{argmin} \varphi_t(\boldsymbol{\alpha})$

- $\phi_t(\alpha)$ の最小化は易しい(微分可能)
- ステップサイズη_tは増加列
- 導出は Fenchel 双対を使う
- 名前の由来: 双対側での Augmented Lagrangian 法なので [Rockafellar 76]

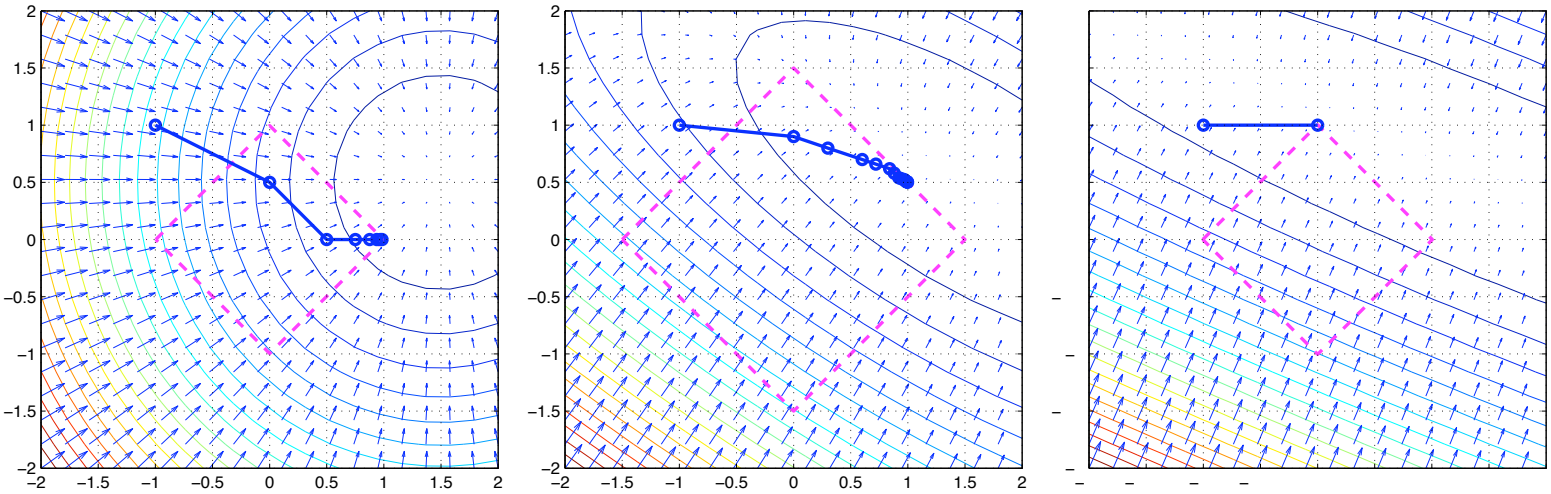
数値例

スケーリングが悪いほどDALが有利

Proximal Gradient



DAL



定理（収束レート）

- W*: 目的関数 f(W)（ロスと正則化項の和）を最小化する元
- W^t: DAL アルゴリズムで生成された解. ただし,$\phi_t(\alpha)$の最小化は以下の条件で止める:

$$\|\nabla\varphi_t(\boldsymbol{\alpha}^{t+1})\| \leq \sqrt{\frac{\gamma}{\eta_t}}\|\boldsymbol{W}^{t+1} - \boldsymbol{W}^t\|_{\mathrm{fro}}$$

- $1/\gamma$ は損失関数の微分 ∇f_ell のリプシッツ定数
- 仮定:

$$f(\boldsymbol{W}^{t+1}) - f(\boldsymbol{W}^*) \geq \sigma\|\boldsymbol{W}^{t+1} - \boldsymbol{W}^*\|_{\mathrm{fro}}^2 \, (t = 0, 1, 2, \ldots)$$

- 結果:超1次収束

$$\|\boldsymbol{W}^{t+1} - \boldsymbol{W}^*\|_{\mathrm{fro}} \leq \frac{1}{\sqrt{1 + 2\sigma\eta_t}}\|\boldsymbol{W}^t - \boldsymbol{W}^*\|_{\mathrm{fro}}$$

定理の味わいどころ

- ロス関数の性質 → 定数γ
  - 停止基準に影響
- デザイン行列の性質 → 定数 σ
  - 停止基準に無関係
  - 正であれば超1次収束が成立（小さくてもよい）
- ステップサイズ η_t
  - 大きいほど収束は速い
  - 大きいほど内部最小化は大変

正則化項の一般化

- マルチ行列学習

$$\phi_\lambda(\boldsymbol{W}) = \lambda \sum_{k=1}^{K} \|\boldsymbol{W}^{(k)}\|_* = \lambda \left\| \begin{pmatrix} \boldsymbol{W}^{(1)} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \boldsymbol{W}^{(K)} \end{pmatrix} \right\|_*$$

  - 大きい行列を作る必要なし
  - 複数の情報源からの情報統合に使える

- 一般のスペクトル正則化

$$\phi_\lambda(\boldsymbol{W}) = \sum_{j=1}^{r} g_\lambda(\sigma_j(\boldsymbol{W}))$$

  - g_λは原点でゼロの凸関数で,以下の操作が計算できる:

$$\mathrm{ST}_{g_\lambda}(\sigma_j) = \operatorname*{argmin}_{x \in \mathbb{R}} \left(g_\lambda(x) + \frac{1}{2}(x - \sigma_j)^2 \right)$$

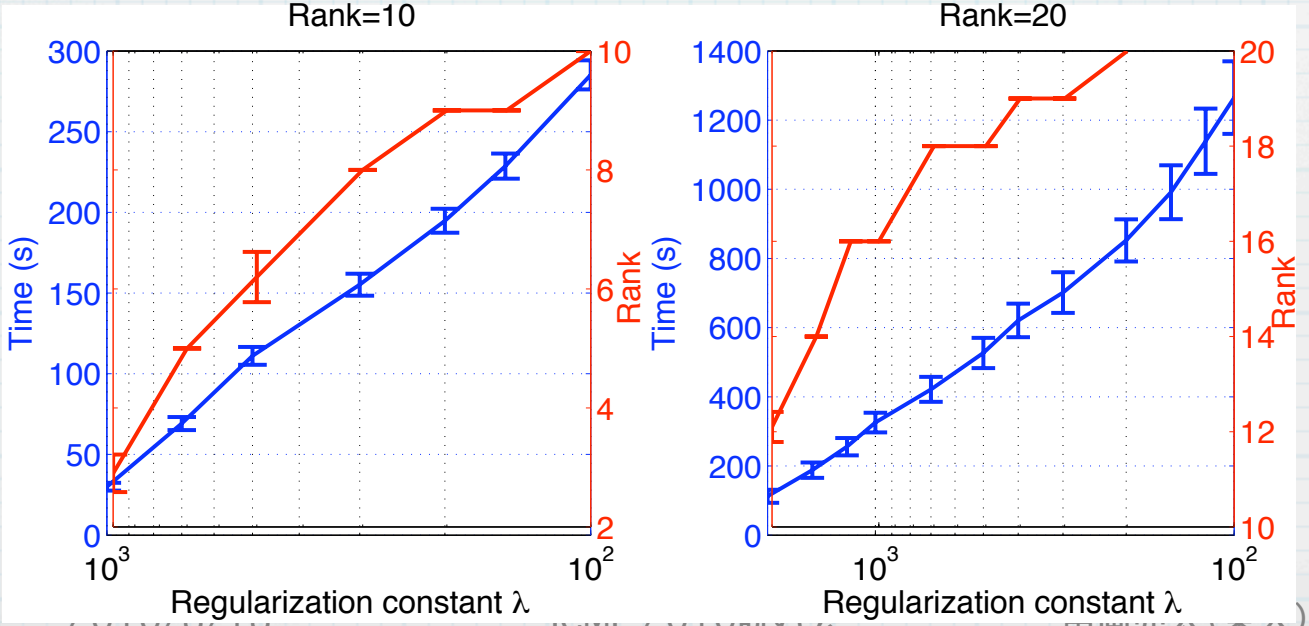
実験1: 行列穴埋め

- 特徴:大規模&構造のあるデザイン行列A
- 真の行列W* : 10,000 x 10,000 (要素数1億) 低ランク
- 観測:ランダムに m 個を選ぶ

➡ 内部目的関数$\phi_t(\alpha)$の最小化に**擬似ニュートン法**を使う

➡ 10,000x10,000行列をメモリに保持する必要なし

m=1,200,000　　m=2,400,000



実験1：ランク=10（詳細）

Rank=10, #observations m=1,200,000

λ	time (s)	#outer	#inner	rank	S-RMSE
1000	33.1 (±2.0)	5 (±0)	8 (±0)	2.8 (±0.4)	0.0158 (±0.0024)
700	77.1 (±5.6)	11 (±0)	18 (±0)	5 (±0)	0.0133 (±0.0008)
500	124 (±7.2)	17 (±0)	28 (±0)	6.4 (±0.5)	0.0113 (±0.0015)
300	174 (±8.0)	23 (±0)	38.4 (±0.84)	8 (±0)	0.00852 (±0.00039)
200	220 (±9.9)	29 (±0)	48.4 (±0.84)	9 (±0)	0.00767 (±0.00031)
150	257 (±9.9)	35 (±0)	58.4 (±0.84)	9 (±0)	0.00498 (±0.00026)
100	319 (±11)	41 (±0)	70 (±0.82)	10 (±0)	0.00743 (±0.00013)

* 内部反復数はたかだか外部反復数の2倍程度

➡ 内部反復の停止基準は厳密過ぎない

実験1：ランク=20（詳細）

Rank=20, #observations m=2,400,000

λ	time (s)	#outer	#inner	rank	S-RMSE
2000	112 (±19)	6 (±0)	15.1 (±1.0)	12.1 (±0.3)	0.011 (±0.002)
1500	188 (±22)	11 (±0)	24.1 (±1.0)	14 (±0)	0.0094 (±0.001)
1200	256 (±25)	15 (±0)	31.1 (±1.0)	16 (±0)	0.0090 (±0.0008)
1000	326 (±29)	19 (±0)	38.1 (±1.0)	16 (±0)	0.0073 (±0.0007)
700	421 (±36)	24 (±0)	48.1 (±1.0)	18 (±0)	0.0065 (±0.0004)
500	527 (±44)	29 (±0)	57.1 (±1.0)	18 (±0)	0.0042 (±0.0003)
400	621 (±48)	34 (±0)	66.1 (±1.0)	19 (±0)	0.0044 (±0.0002)
300	702 (±59)	38.5 (±0.5)	74.1 (±1.5)	19 (±0)	0.0030 (±0.0003)
200	852 (±61)	43.6 (±0.5)	83.9 (±2.3)	20 (±0)	0.0039 (±0.0001)
150	992 (±78)	48.4 (±0.7)	92.5 (±1.5)	20 (±0)	0.0024 (±0.0002)
120	1139 (±94)	53.4 (±0.7)	102 (±1.5)	20 (±0)	0.0016 ($\pm 6 \times 10^{-5}$)
100	1265 (±105)	57.7 (±0.8)	109 (±2.4)	20 (±0)	0.0013 ($\pm 8 \times 10^{-5}$)

実験2: 行列の上の教師付き判別

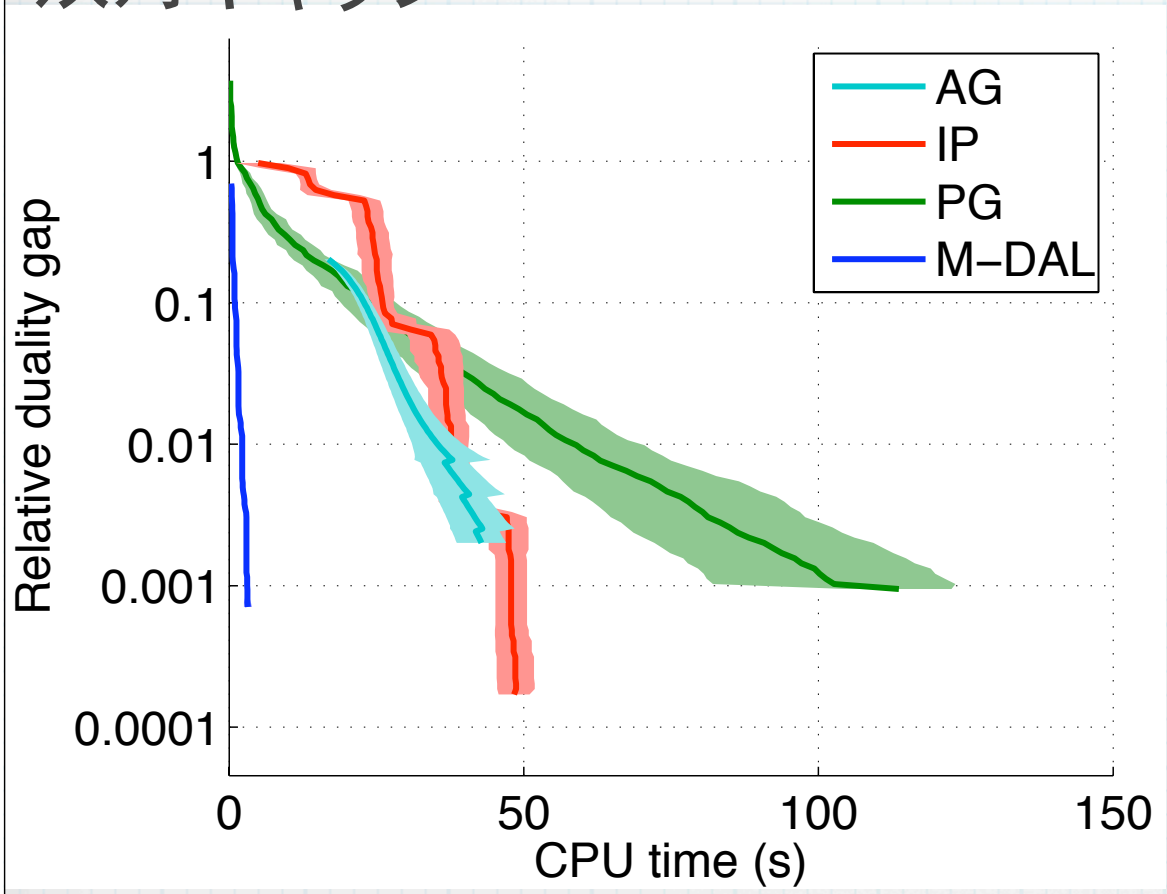
- 真の行列 W*: 64x64, rank=16
- サンプル数 m=1,000
- ロジスティック損失:

$$f_\ell(\boldsymbol{z}) = \sum_{i=1}^{m} \log(1 + \exp(-y_i z_i))$$

- デザイン行列A: 1000x64^2 (dense)
  - 各訓練データはウィッシャート分布からサンプル
- 中規模でdense → 内部目的関数$\phi_t(\alpha)$の最小化にニュートン法を使う

実験2: 結果

双対ギャップ

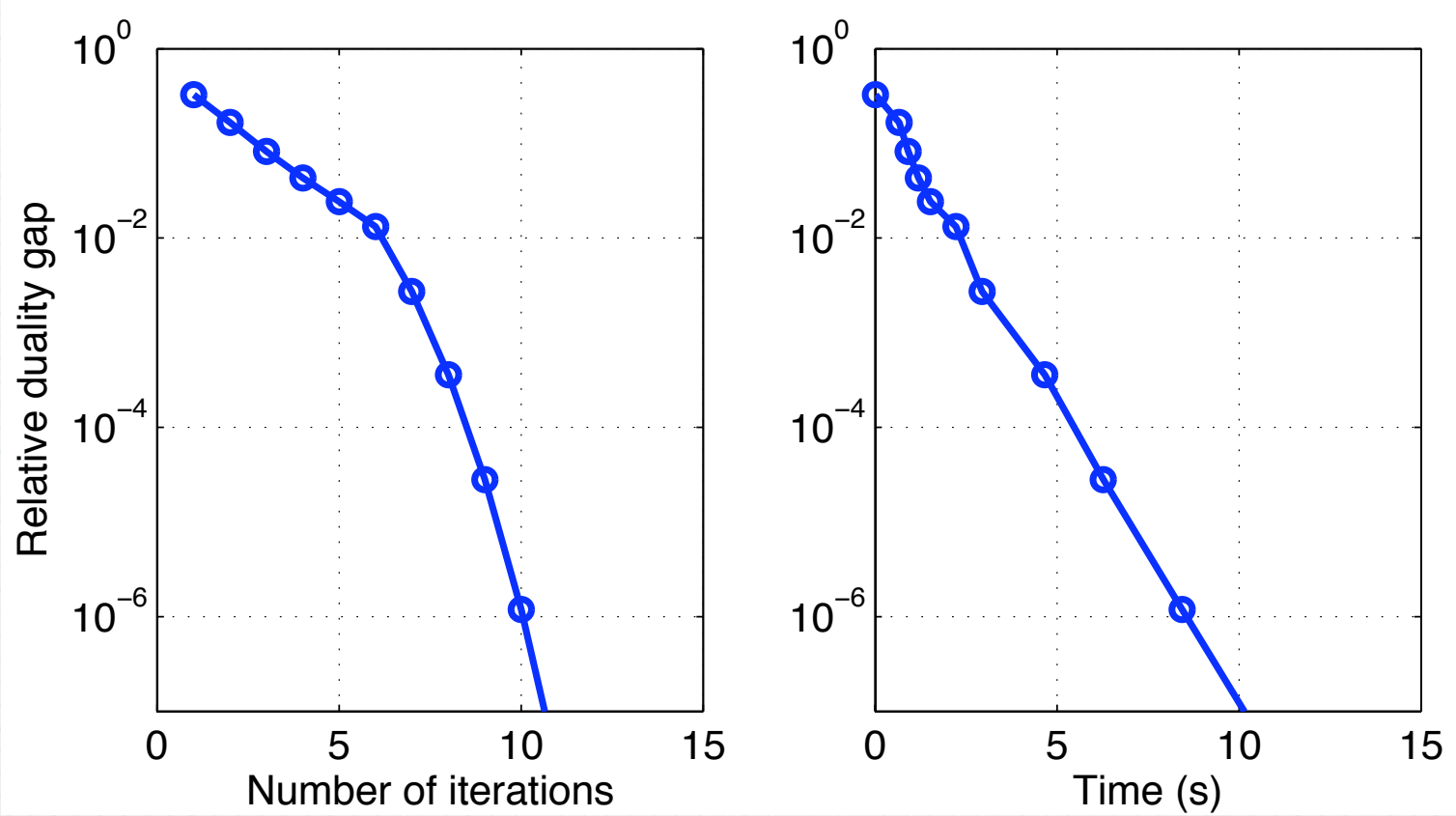


計算時間

- M-DAL (提案手法)
- IP (内点法)
- PG (Projected Gradeint)
- AG (Accelerated Gradient) [Ji & Ye, 09] : Proximal Gradient を速くしたもの

超一次収束

双対ギャップ



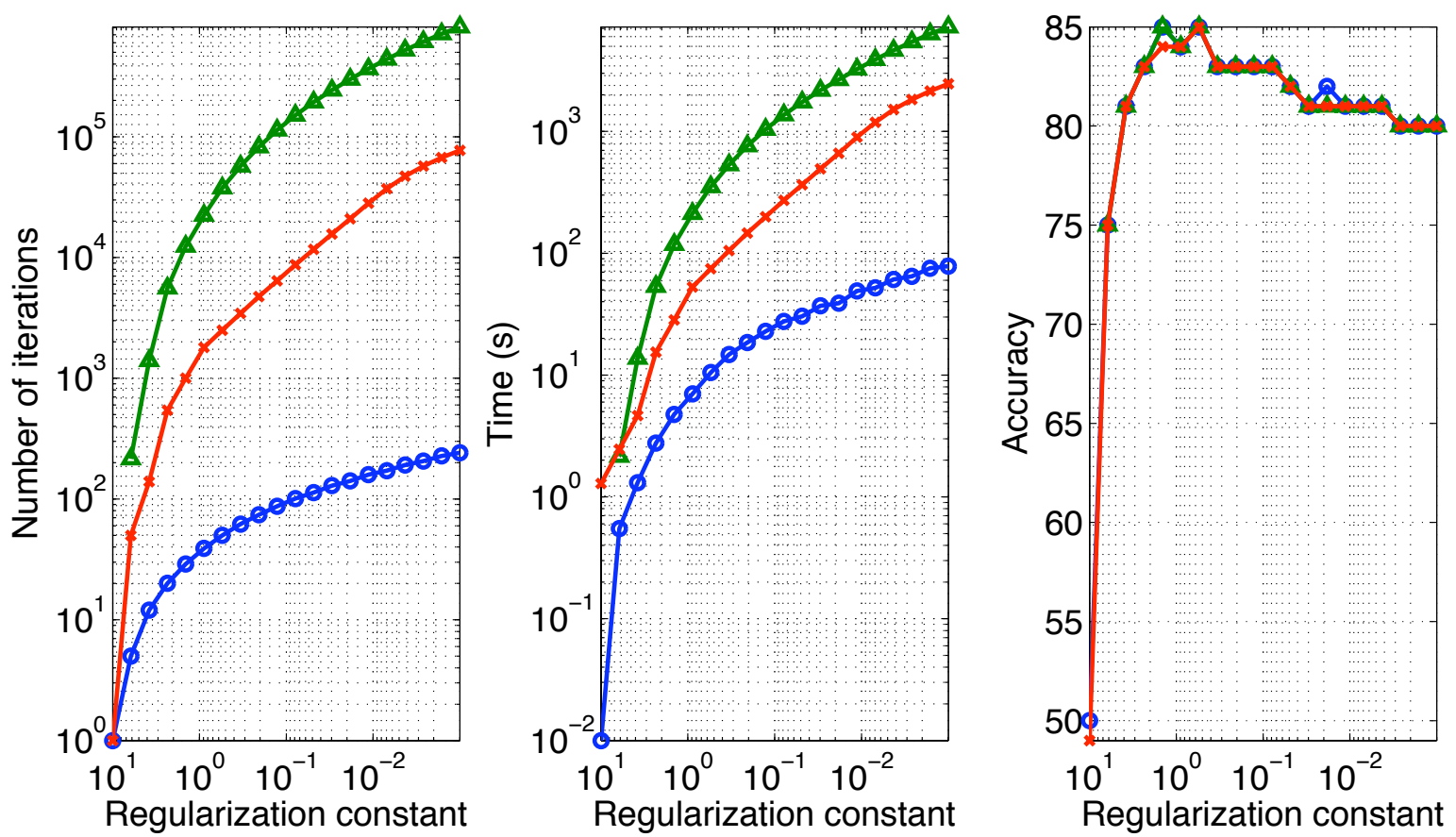
反復数　　計算時間

実験3: BCI

* BCI comptition 2003 dataset IV
* タスク: 次に右／左のどちらの手の指でキーを叩くか予測する
* データ: 多次元脳波データ: 28チャネルx50時間点 (長さ500ms)
* それぞれの訓練データを3つの行列に加工
  * 1次成分 (<20Hz) : 28x50行列
  * 2次成分 (alpha 7-15Hz): 28x28行列(共分散)
  * 2次成分 (beta 15-30Hz): 28x28行列(共分散)
* 訓練データ数 316. テストデータ数100.

BCI: 結果

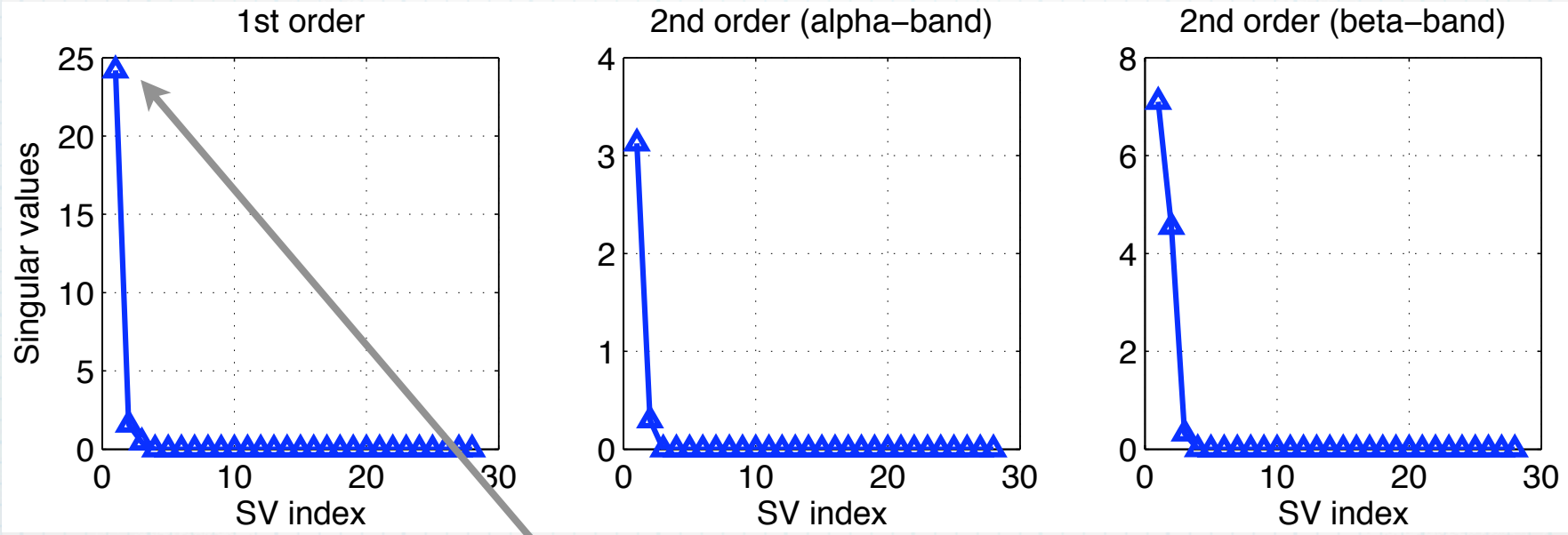
-o- M-DAL (提案法), -△- PG, -x- AG [Ji&Ye 09]

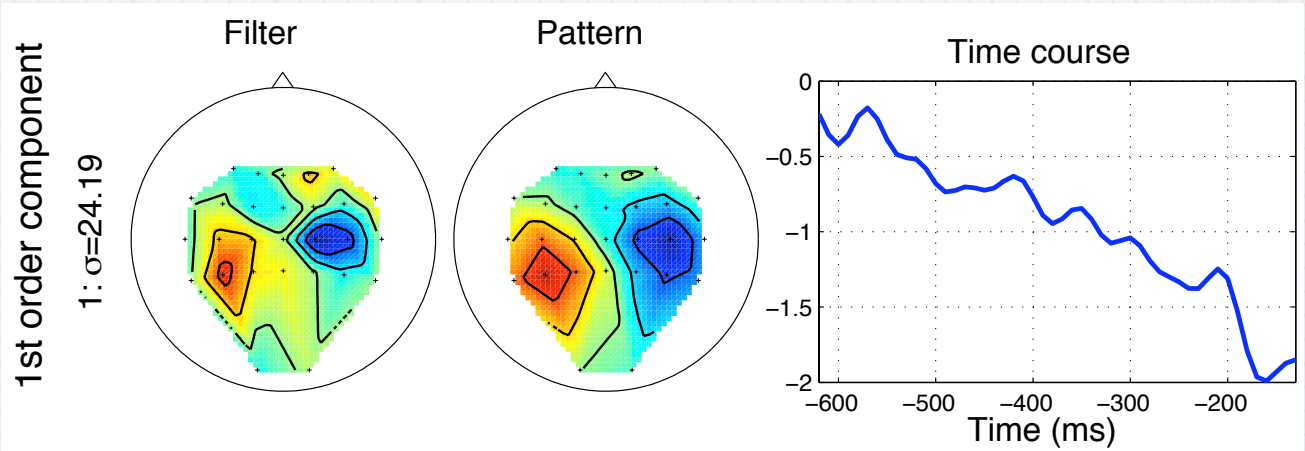


* 外部停止基準:双対ギャップ <=1e-3
* 注:上の計算時間は正則化パスの計算のコスト

BCI: 結果

特異値


1st order
2nd order (alpha−band)
2nd order (beta−band)
Singular values
SV index
SV index
SV index


1st order component
1: σ=24.19
Filter
Pattern
Time course
Time (ms)

まとめ

- トレースノルム(とその拡張)は低ランク行列の学習を凸最適化で行うための鍵
- トレースノルムは行列におけるL1正則化に対応
- DALは途中の解を低ランクで保持することが可能
- DAL は非漸近的に超1次収束する

➡ 反復数が非常に小さい

- 特異値に関して分解可能な正則化項+微分可能な損失関数

➡ 1反復あたりの計算量が小さい

- 低ランク行列の推定／学習だけでなく複数情報源の統合も可能

ICMLのフィードバック

- Matrix Factorization は流行り気味?
  - セッション1a: Topic Models and Matrix Factorization
  - セッション2a: Matrix Factorization and Recommendation
- スパース／MKL／最適化はぽつぽつ