スライド: http://goo.gl/2nSya

画像のスパース表現の理論と最適化アルゴリズム

冨岡 亮太（東京大学）
tomioka@mist.i.u-tokyo.ac.jp

スパース表現：2つの視点

- データの圧縮
  - 90年代: ウェーブレットにもとづくノイズ除去 [Donoho et al.]
  - 2000年代: 圧縮センシング [Candes et al.]
- モデルの圧縮
  - 90年代: L1正則化付き学習 [Tibshirani et al.]
  - 2000年代: マルチカーネル学習 [Lanckriet, Bach et al.]

データの圧縮の視点から

圧縮センシング

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2 + \lambda\|\boldsymbol{\Psi}\boldsymbol{x}\|_\star$$

y: 観測

x: 復元したい量

A: デザイン行列（観測の過程）

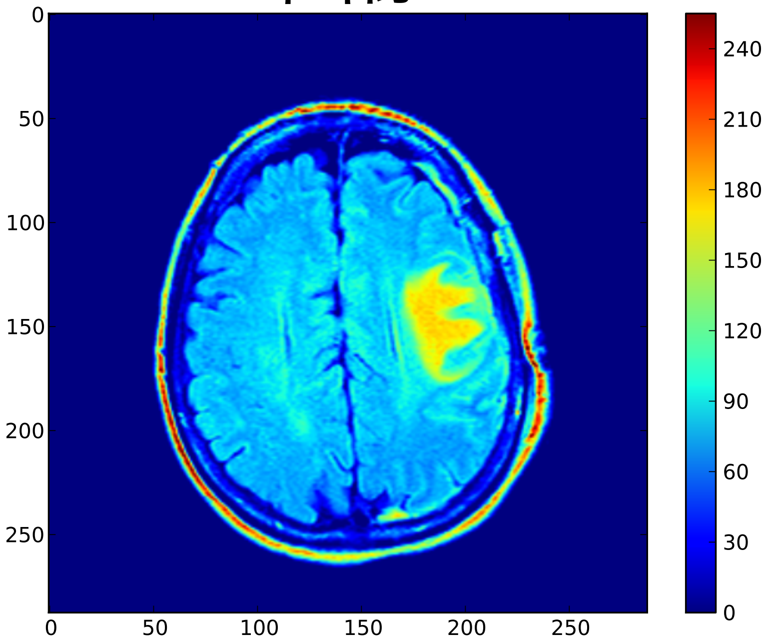
Ψ: スパース基底（例：ウェーブレット変換）

$\|\cdot\|_\star$ ：スパース性ノルム（例：L_1 ノルム）

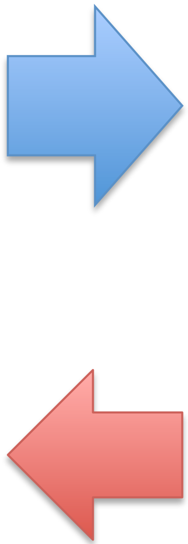
スライド: http://goo.gl/2nSya

圧縮センシング：部分観測からの復元

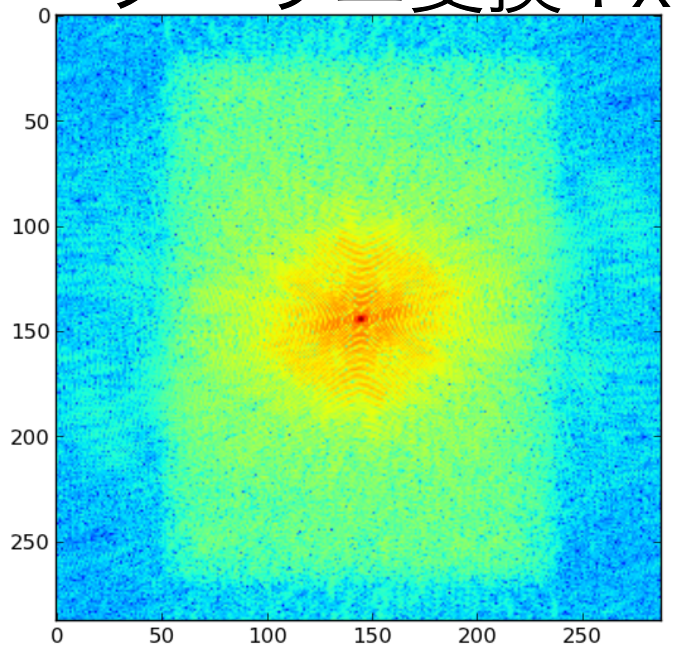
画像 x



観測



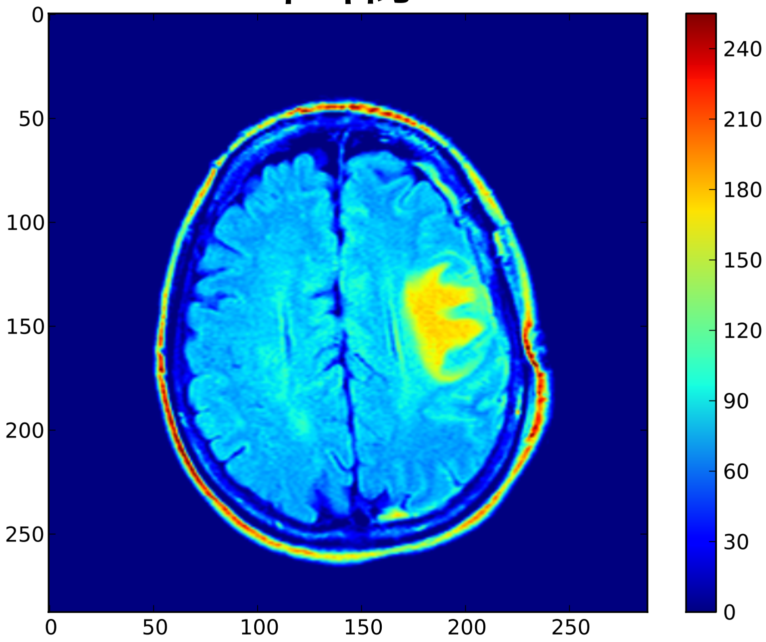
フーリエ変換 Fx



復元

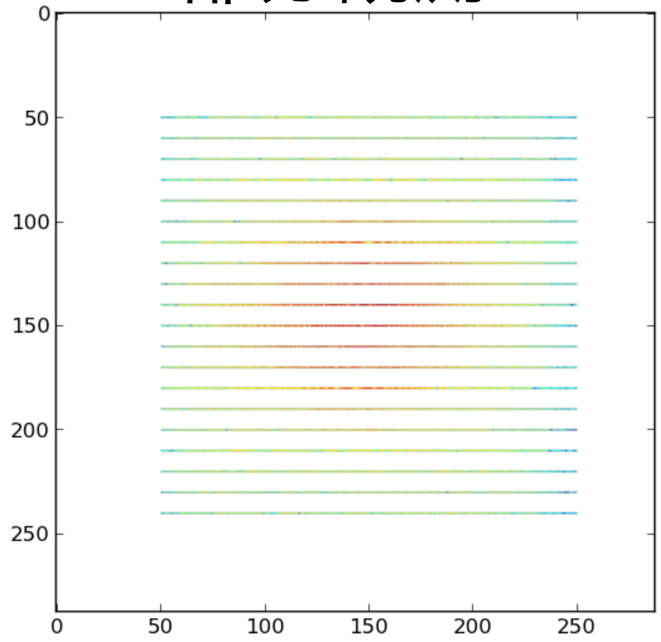
圧縮センシング：部分観測からの復元

画像 x



観測

部分観測 Ax

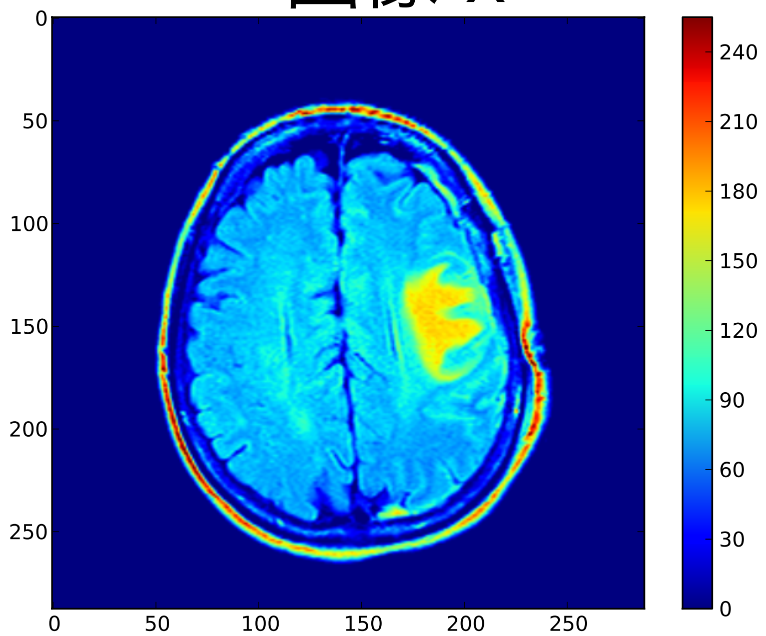


復元？

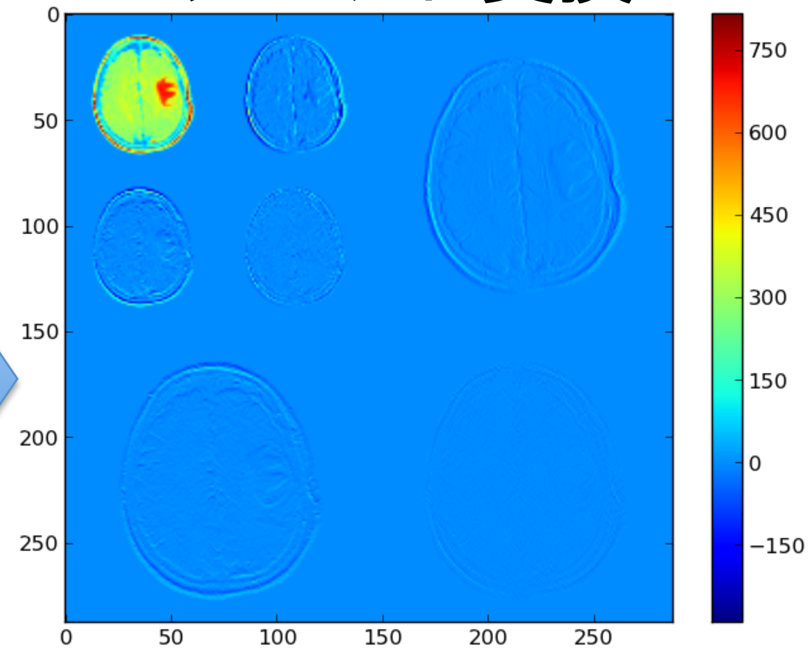
デザイン行列 A
➡ フーリエ変換係数の一部を取り出す操作

MRI画像のスパース性

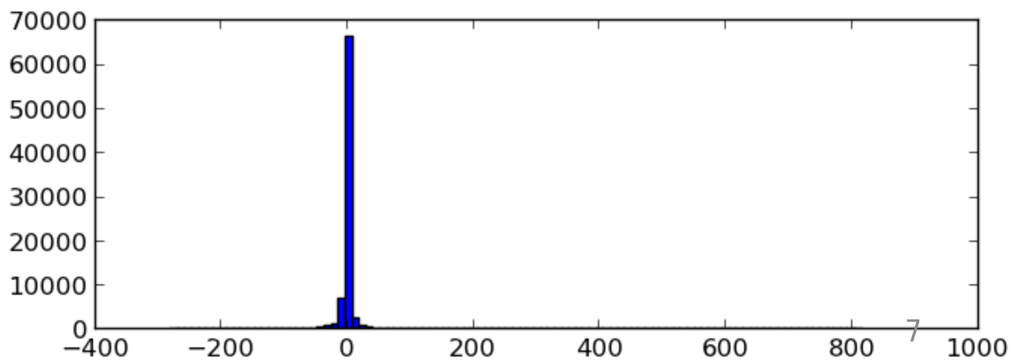
画像 x



ウェーブレット変換 Ψx



スパース基底 Ψ
➡ ウェーブレット変換.
スパース性ノルム
➡ L_1ノルム.



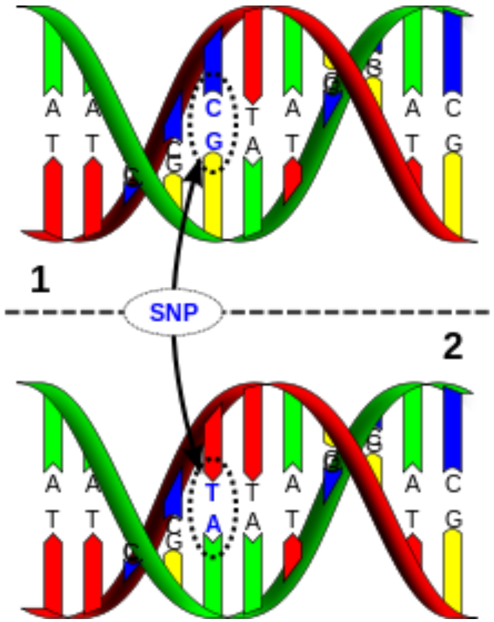
モデルの圧縮の視点から

スライド: http://goo.gl/2nSya

L_1正則化付き学習—SNP (一塩基多型) 解析

- SNPの数 p=500,000
- サンプル数 n=5,000

p ≫ n!!



http://en.wikipedia.org/wiki/Single-nucleotide_polymorphism

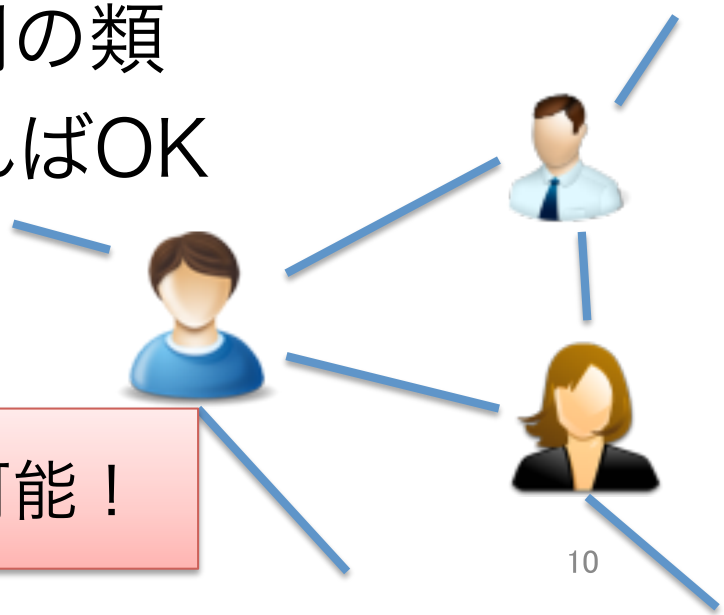
最適化問題: L_1正則化付きロジスティック回帰

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad L(\boldsymbol{y} \circ \boldsymbol{X}\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1$$

ロジスティック損失

カーネル法

- 特徴量の陽な表現―次元の呪い
  - オーバーフィット
  - 計算困難
- カーネル法
  - 特徴量の代わりにサンプル間の類似度（内積）さえ定義できればOK
  - 次元に依存しない理論



多くの学習アルゴリズムはカーネル化可能！

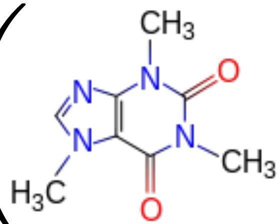
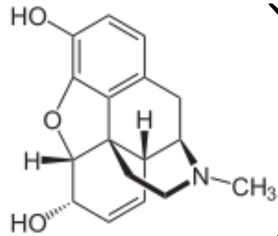
類似度（内積）の表現としてのカーネル

- カーネル関数は任意のオブジェクトの間の内積を表現可能

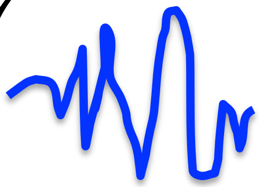
画像間の類似度

$k\Bigg($  ,  $\Bigg)$

化学物質間の類似度

$k\Bigg($  ,  $\Bigg)$

時系列間の類似度

$k\Bigg($  ,  $\Bigg)$

ユーザ間の類似度

$k\Bigg($  ,  $\Bigg)$

スライド: http://goo.gl/2nSya

マルチカーネル学習

[Lanckriet et al. 04; Bach et al. 04; Micchelli & Pontil 05]

- 多数の候補となるカーネルをいかに組み合わせるか？

画像

形状カーネル

$$k_{\mathrm{shape}}(x, x')$$

色カーネル

$$k_{\mathrm{color}}(x, x')$$

テクスチャカーネル

$$k_{\mathrm{texture}}(x, x')$$

$$\begin{aligned} k(x, x') &= d_1 {\color{blue} k_{\mathrm{shape}}(x, x')} \\ &+ d_2 {\color{red} k_{\mathrm{color}}(x, x')} \\ &+ d_3 {\color{green} k_{\mathrm{texture}}(x, x')} \end{aligned}$$

画像認識におけるカーネル選択

- 組み合わせ爆発！
  - Descriptor (SIFT, HueSIFT, etc.)
  - 空間分割、spatial pyramid
  - ヒストグラムに対するカーネル関数
  - カーネル関数のパラメータ
- 1,000 – 10,000 候補カーネル

多数のカーネルにスケールするアルゴリズムが必要

最適化アルゴリズム

解きたい問題

- Type I
  - 圧縮センシングでΨ^{-1}が存在する場合
  - L_1正則化学習、マルチカーネル学習

$$\min_{\boldsymbol{x}} \quad L(\boldsymbol{x}) + \lambda \|\boldsymbol{x}\|_1$$

- Type II
  - 圧縮センシングでΨ^{-1}が存在しない場合

$$\min_{\boldsymbol{x}} \quad L(\boldsymbol{x}) + \lambda \|\boldsymbol{\Psi x}\|_1$$

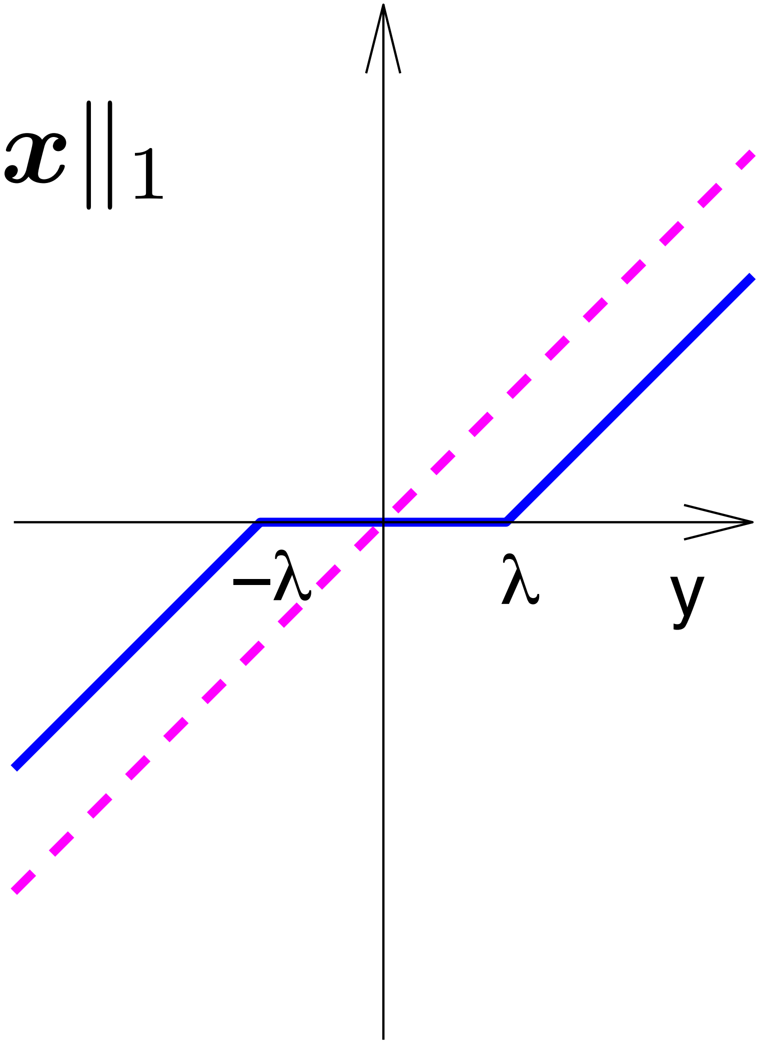
解ける問題

$$\min_{\boldsymbol{x}} \quad \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$$

$$\hat{x}_j = \begin{cases} y_j + \lambda & (y_j < -\lambda), \\ 0 & (-\lambda \le y_j \le \lambda), \\ y_j - \lambda & (y_j > \lambda) \end{cases}$$

$$(j = 1, \ldots, n)$$

$$=: \mathrm{prox}_\lambda(\boldsymbol{y})$$



近接勾配法

- いかに解ける形に帰着するか⇒線形化

$$\hat{\boldsymbol{x}} = \operatorname*{argmin}_{\boldsymbol{x}} \left(L(\boldsymbol{x}) + \lambda \|\boldsymbol{x}\|_1 \right)$$

繰り返しアルゴリズム

$$\boldsymbol{x}^{t+1} = \mathrm{prox}_{\lambda \eta_t} \left(\boldsymbol{x}^t - \eta_t \nabla L(\boldsymbol{x}^t) \right).$$

スライド: http://goo.gl/2nSya

近接勾配法

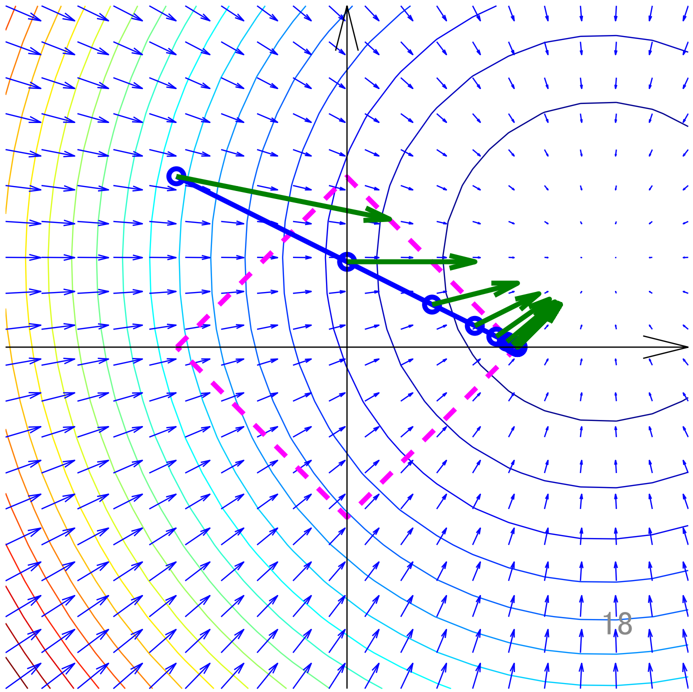
1. 適当にx^0を初期化
2. 収束するまで

$$\boldsymbol{x}^{t+1} = \mathrm{prox}_{\lambda\eta_t}\left(\boldsymbol{x}^t - \eta_t \nabla L(\boldsymbol{x}^t)\right)$$

利点：実装が簡単

欠点：損失項のLのヘシアンの条件数が悪いと遅い

別名：Iterative Shrinkage/Thresholding, Forward-Backward Splitting



近接勾配法—理論的な性質

- 微分可能な関数に対する勾配法と同じ
  - 微分不可能なL_1ノルム項を解析的に扱うため
- L のヘシアンの最小、最大固有値が有限の場合、線形収束
- 最大固有値のみ有限： O(1/k) で収束（k: 反復数)
- $O(1/k^2)$を達成するための加速法が提案されている [Nesterov 2007; Beck & Teboulle 2009]

具体例

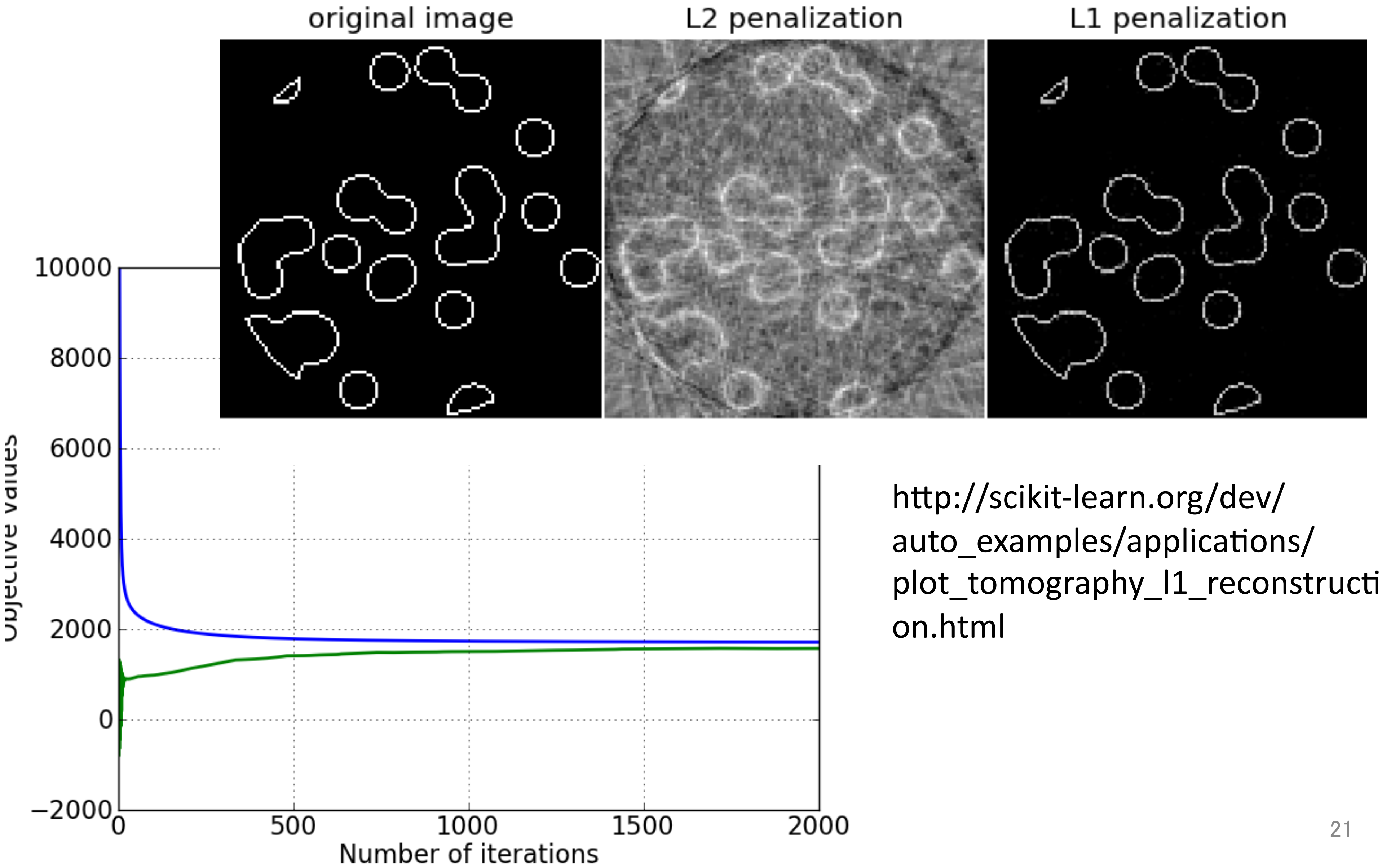
$$\min_{\boldsymbol{x}} \quad \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$$

繰り返しアルゴリズム

$$\boldsymbol{x}^{t+1} = \mathrm{prox}_{\lambda\eta_t}\left(\boldsymbol{x}^t + \eta_t \boldsymbol{A}^\top\left(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^t\right)\right)$$

- 主な計算コストは行列演算A, $A^\top$
  - 圧縮センシングでは A→フーリエ変換、$A^\top$→フーリエ逆変換 (FFTで高速化可能)
- 正則化項はproxさえ計算できれば一般化可能

圧縮センシングへの応用



http://scikit-learn.org/dev/auto_examples/applications/plot_tomography_l1_reconstruction.html

解きたい問題 (type II)

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 + \lambda\|\boldsymbol{\Psi}\boldsymbol{x}\|_1$$

y: 観測

x: 求めたい量

L_1ノルム

A: デザイン行列（観測の過程）

Ψ: スパース基底（例：ウェーブレット変換）

スライド: http://goo.gl/2nSya

交互方向乗数法—Type II のための アルゴリズム

- Type II 最適化問題

$$\min_{\boldsymbol{x}} \quad \underbrace{L(\boldsymbol{x})}_{\text{微分可能}} + \underbrace{\lambda \|\Psi \boldsymbol{x}\|_1}_{\text{微分不可能}}$$

- 問題： prox が容易に計算できない

補助変数を導入して分離

$$\min_{\boldsymbol{x}, \boldsymbol{z}} \quad L(\boldsymbol{x}) + \lambda \|\boldsymbol{z}\|_1,$$

$$\boldsymbol{z} = \boldsymbol{\Psi} \boldsymbol{x}$$

スライド: http://goo.gl/2nSya

拡張ラグランジュ関数

[Hestenes 69; Powell 69]

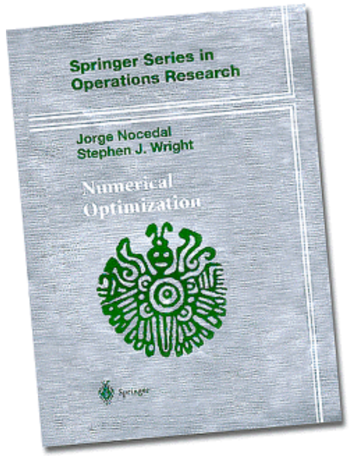
$$\mathcal{L}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\alpha}) = \underbrace{L(\boldsymbol{x}) + \lambda\|\boldsymbol{z}\|_1 + \boldsymbol{\alpha}^\top(\boldsymbol{z}-\boldsymbol{\Psi}\boldsymbol{x})}_{\text{普通のラグランジュ関数 (Boyd “Convex Optimization”)}} + \underbrace{\frac{\eta}{2}\|\boldsymbol{z}-\boldsymbol{\Psi}\boldsymbol{x}\|_2^2}_{\text{ペナルティ項}}$$

重要なポイント：

もとの問題をx, zに関して最小化することは（拡張）ラグランジュ関数の鞍点

$$\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{x},\boldsymbol{z}} \mathcal{L}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\alpha})$$

を見つけることに等しい。



Numerical Optimization (Nocedal & Wright) 2006

スライド: http://goo.gl/2nSya

いかに鞍点を見つけるか？

- 拡張ラグランジュ法：主問題の変数x, zについて*L*を同時最小化

$$
\begin{cases}
(\boldsymbol{x}^{t+1}, \boldsymbol{z}^{t+1}) = \mathrm{argmin}_{\boldsymbol{x},\boldsymbol{z}}\, \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}^t), \\
\boldsymbol{\alpha}^{t+1} = \boldsymbol{\alpha}^t + \eta(\boldsymbol{z}^{t+1} - \boldsymbol{\Psi}\boldsymbol{x})
\end{cases}
$$

結構つらい・・・

双対問題に対する勾配法

- 交互方向乗数法

$$
\begin{cases}
\boldsymbol{x}^{t+1} = \mathrm{argmin}_{\boldsymbol{x}}\, \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}^t, \boldsymbol{\alpha}^t), \\
\boldsymbol{z}^{t+1} = \mathrm{argmin}_{\boldsymbol{z}}\, \mathcal{L}(\boldsymbol{x}^{t+1}, \boldsymbol{z}, \boldsymbol{\alpha}^t), \\
\boldsymbol{\alpha}^{t+1} = \boldsymbol{\alpha}^t + \eta(\boldsymbol{z}^{t+1} - \boldsymbol{\Psi}\boldsymbol{x}^{t+1})
\end{cases}
$$

交互に最適化

双対問題に対する勾配法

交互方向乗数法—理論的な性質

- 収束性：緩い仮定のもとで証明可能 (Boyd 2011 参照)
- 収束レート
  - O(1/k): He & Yuan 2012
  - O(exp(-k)) (線形収束): Deng & Yin 2012
  - 実際は近接勾配法より速い感じ

具体例

$$\min_{\boldsymbol{x}} \quad \frac{1}{2\lambda}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\|\boldsymbol{\Psi}\boldsymbol{x}\|_1$$

拡張ラグランジュ関数

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\alpha})=\frac{1}{2\lambda}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\|\boldsymbol{z}\|_1+\boldsymbol{\alpha}^\top(\boldsymbol{z}-\boldsymbol{\Psi}\boldsymbol{x})+\frac{\eta}{2}\|\boldsymbol{z}-\boldsymbol{\Psi}\boldsymbol{x}\|_2^2$$

繰り返しアルゴリズム

$$\begin{cases}\boldsymbol{x}^{t+1}=\mathrm{argmin}_{\boldsymbol{x}}\left(\frac{1}{2\lambda}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\frac{\eta}{2}\|\boldsymbol{z}^t-\boldsymbol{\Psi}\boldsymbol{x}+\boldsymbol{\alpha}^t\|_2^2\right),\\ \boldsymbol{z}^{t+1}=\mathrm{argmin}_{\boldsymbol{z}}\left(\|\boldsymbol{z}\|_1+\frac{\eta}{2}\|\boldsymbol{z}-\boldsymbol{\Psi}\boldsymbol{x}^{t+1}+\boldsymbol{\alpha}^t\|_2^2\right),\\ \boldsymbol{\alpha}^{t+1}=\boldsymbol{\alpha}^t+(\boldsymbol{z}^{t+1}-\boldsymbol{\Psi}\boldsymbol{x}^{t+1})\end{cases}$$

まとめ

- スパースに表現することの2つの側面
  - データの圧縮
  - モデルの圧縮
- 2つの最適化問題
  - Type I — 正則化項がバラけている
  - Type II — 正則化項がバラけていない
- 最適化アルゴリズム
  - （加速付き）近接勾配法 — 21世紀の勾配法?
  - 交互方向乗数法

お話しできなかった話題

- 拡張ラグランジュ法／交互方向乗数法をType I にも適用できないか?
  - Type I の双対問題は Type II
  - 双対拡張ラグランジュ法 (DAL)
- L_1 正則化以外の正則化項
  - グループL_1 正則化（マルチカーネル学習）
  - トレースノルム正則化
    - Prox作用素が計算できるのでほとんど同じ議論
- より複雑な正則化項
  - Type I、Type II どちらかになっていることが多い