Proximal Newton Methods for Large-Scale Machine Learning

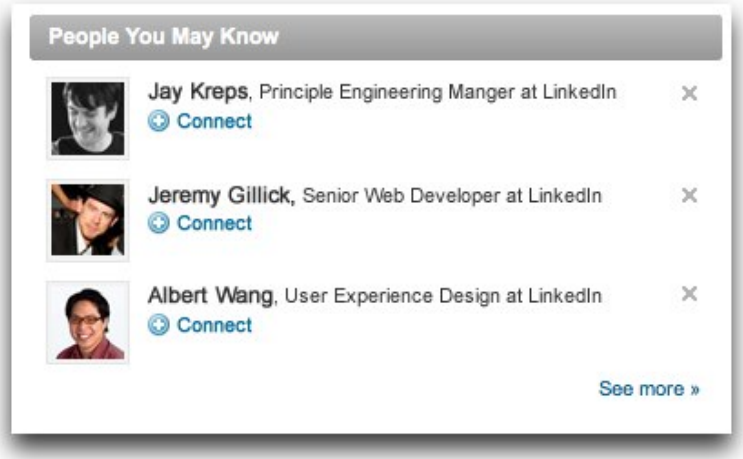
Inderjit Dhillon
Computer Science & Mathematics
UT Austin

ShanghaiTech Symposium on Data Science
June 24, 2015

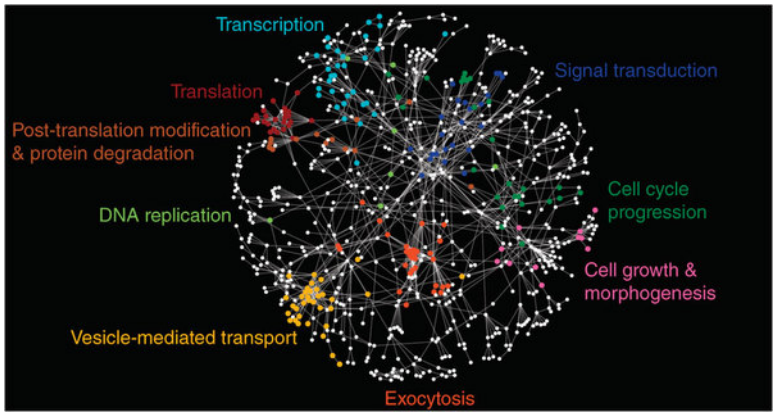
Joint work with C. J. Hsieh, M. Sustik, P. Ravikumar, R. Poldrack, S. Becker, and P. Olsen

Big Data

Link prediction



LinkedIn.



gene-gene network

fMRI

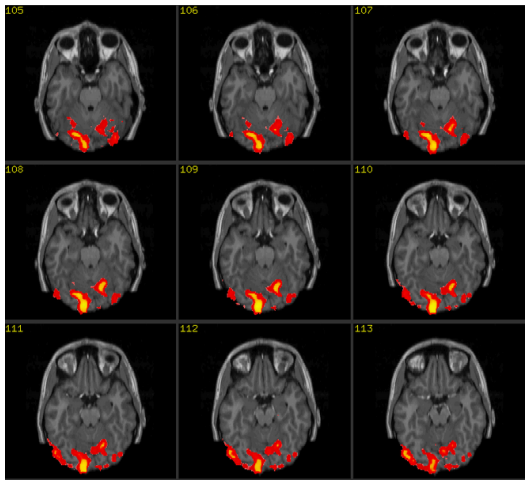
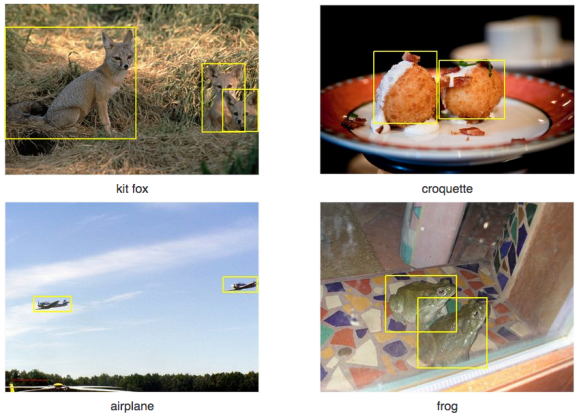
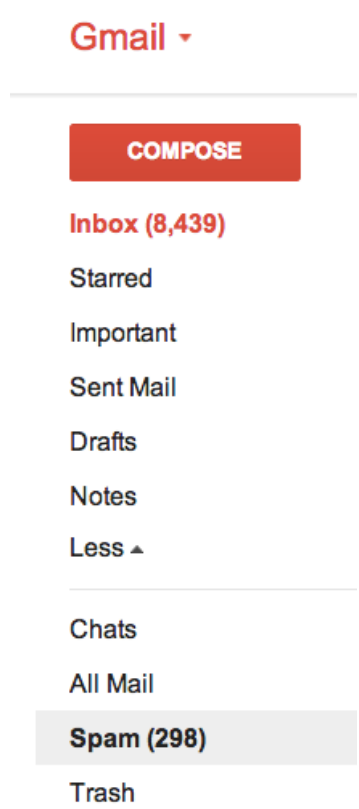


Image classification



Spam classification

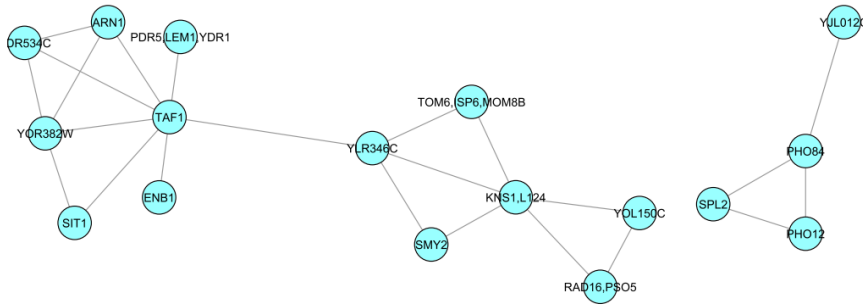


Modern Machine Learning Problems

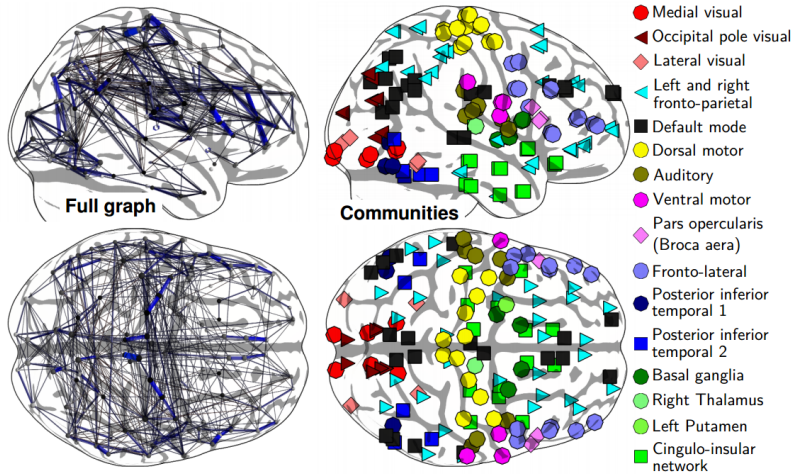
Inverse Covariance Estimation

- Learning graph (structure and/or weights) from data.
- Gene networks, fMRI analysis, stock networks, climate analysis.

Gene network

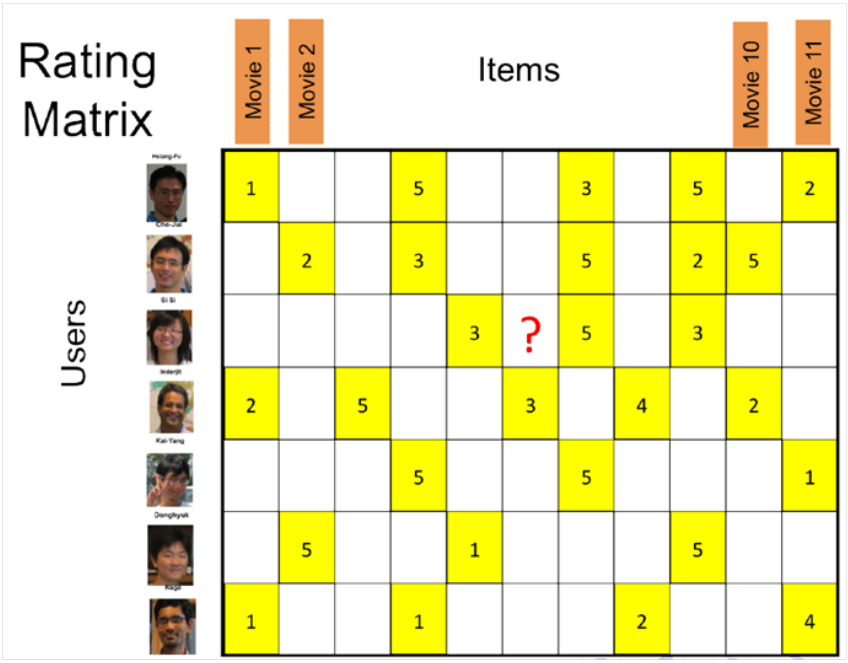


fMRI brain analysis



Matrix Completion

- Missing value estimation for recommender systems.
- Assumption: the underlying matrix is low-rank.



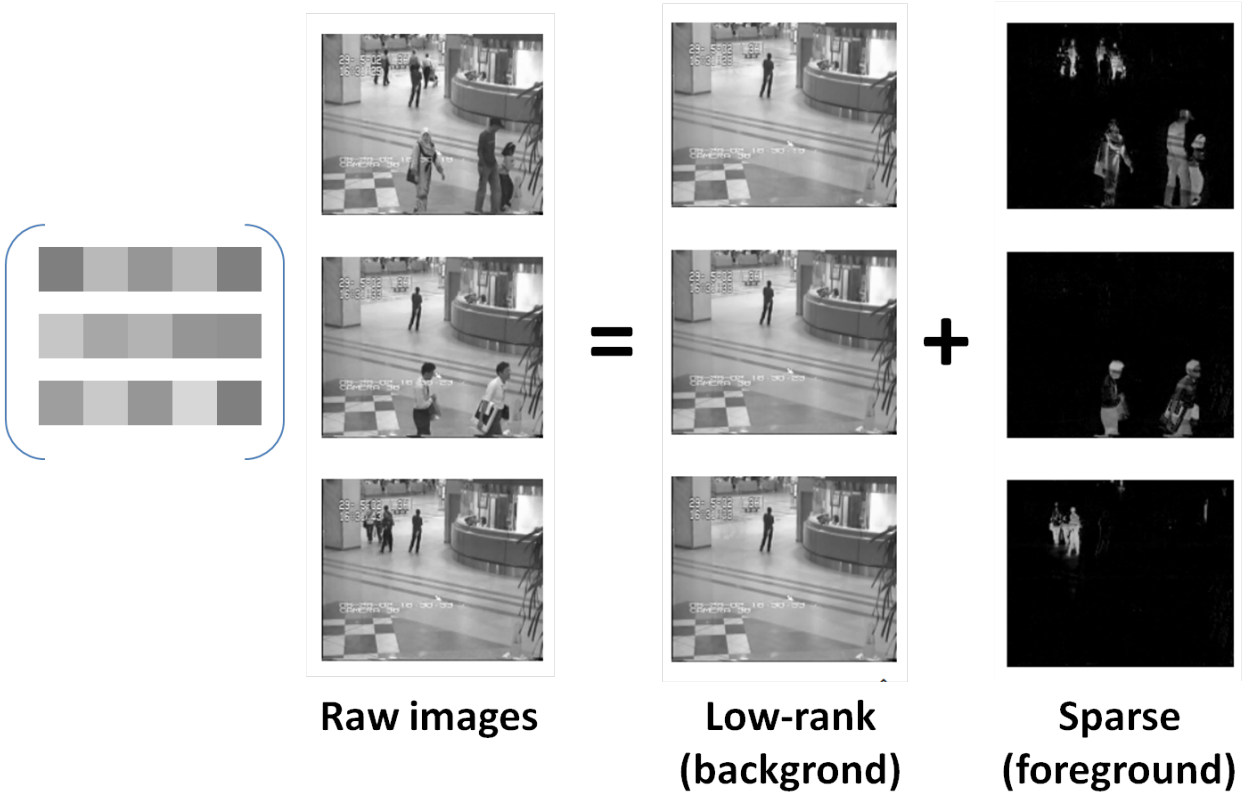
≈

-8.72	0.03	-1.03
-7.56	-0.79	0.62
-4.07	-3.95	2.55
-3.52	3.73	-3.32
-7.78	2.34	2.33
-2.44	-5.29	-3.92
-1.78	1.90	-1.68

-0.07	-0.11	-0.53	-0.46	-0.06	-0.05	-0.53	-0.07	-0.35	-0.19	-0.14
0.13	-0.42	0.45	0.17	-0.25	-0.17	-0.18	0.27	-0.59	0.05	0.14
-0.21	-0.43	-0.23	0.16	0.08	0.17	0.57	-0.39	-0.37	-0.08	-0.15

Robust PCA

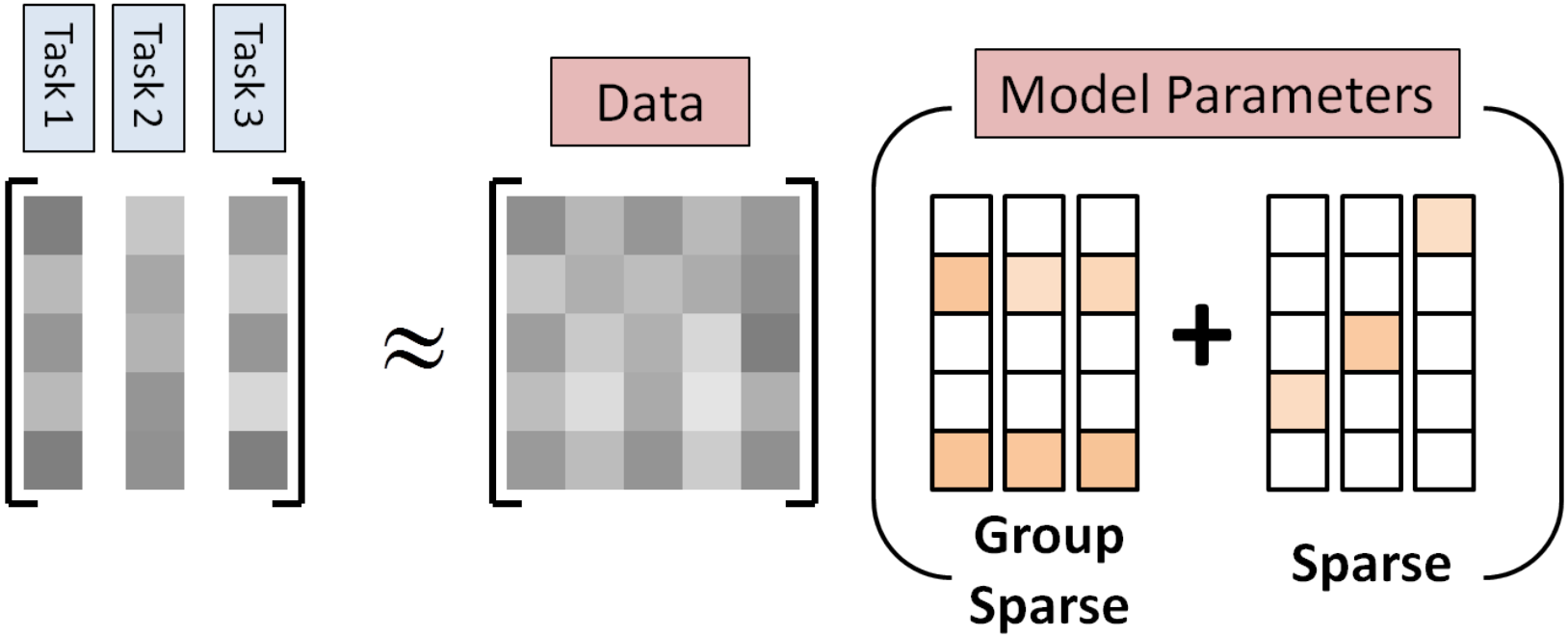
- Image de-noising, face recognition, graphical modeling with latent variables
- Decompose the input matrix as sparse + low-rank matrices.



Figures from (Candès et al, 2009)

Multi-task Learning

- Training T classification tasks together.
- Sparse models: Lasso
- Feature sharing: Group Lasso



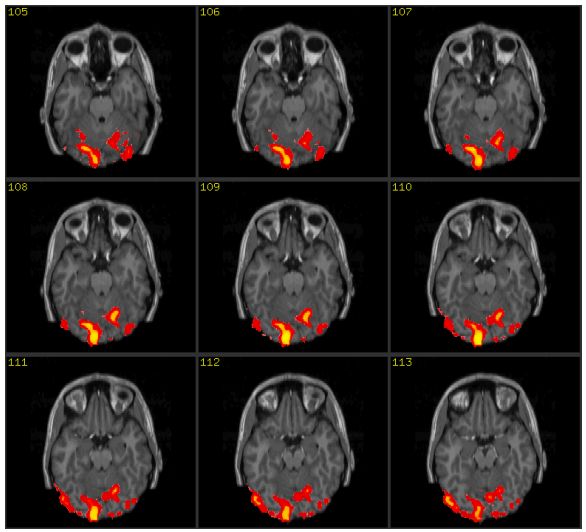
Inverse Covariance Estimation

Estimate Relationships between Variables

- Example: FMRI Brain Analysis.
- Goal: Reveal functional connections between regions of the brain.

(Sun et al, 2009; Smith et al, 2011; Varoquaux et al, 2010; Ng et al, 2011)

Input



Output

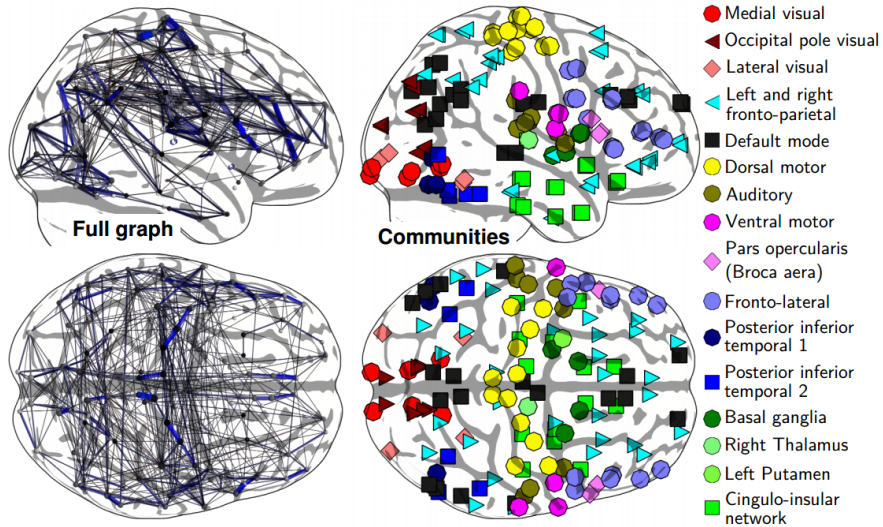


Figure from (Varoquaux et al, 2010)

Estimate Relationships between Variables

- Example: Gene regulatory network discovery.
- Goal: Gene network reconstruction using microarray data.

  (Schafer & Strimmer 2005; Andrei & Kendziorski 2009; Menendez et al, 2010; Yin and Li, 2011)

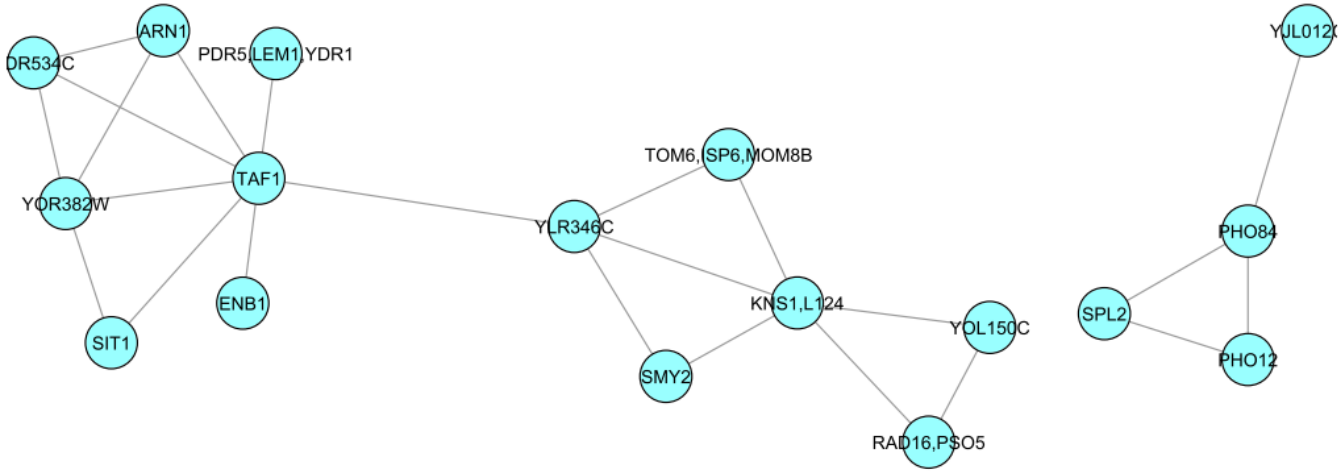


Figure from (Banerjee et al, 2008)

Other Applications

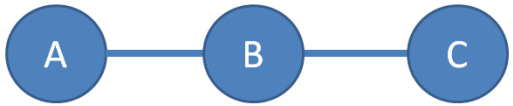
- Financial Data Analysis:
  - Model dependencies in multivariate time series (Xuan & Murphy, 2007).
  - Sparse high dimensional models in economics (Fan et al, 2011).
- Social Network Analysis / Web data:
  - Model co-authorship networks (Goldenberg & Moore, 2005).
  - Model item-item similarity for recommender system(Agarwal et al, 2011).
- Climate Data Analysis (Chen et al., 2010).
- Signal Processing (Zhang & Fung, 2013).
- Anomaly Detection (Ide et al, 2009).

Inverse Covariance Estimation

- Given: n i.i.d. samples $\{\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}\}$, $\boldsymbol{y_i} \in R^p$, $\boldsymbol{y_i} \sim \mathcal{N}(\mu, \Sigma)$,
- Goal: Estimate relationships between variables.
- The sample mean and covariance are given by:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y_i} \quad \text{and} \quad S = \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{y_i} - \hat{\mu})(\boldsymbol{y_i} - \hat{\mu})^T.$$

- Covariance matrix Σ may not directly convey relationships.
- An example – Chain graph: $y(j) = 0.5y(j-1) + \mathcal{N}(0,1)$



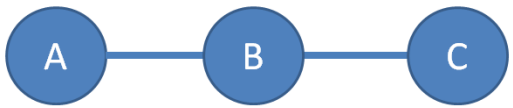
$$\Sigma = \begin{pmatrix} 1.33 & 0.67 & 0.33 \\ 0.67 & 1.33 & 0.67 \\ 0.33 & 0.67 & 1.33 \end{pmatrix}$$

Inverse Covariance Estimation (cont'd)

- Conditional independence is reflected as zeros in Θ $(= \Sigma^{-1})$:

  $\Theta_{ij} = 0 \Leftrightarrow$ $y(i), y(j)$ are conditionally independent given other variables

- Chain graph example: $y(j) = 0.5y(j-1) + \mathcal{N}(0,1)$



$$\Theta = \begin{pmatrix} 1 & -0.5 & 0 \\ -0.5 & 1.25 & -0.5 \\ 0 & -0.5 & 1 \end{pmatrix}$$

- In a GMRF $G = (V, E)$, each node corresponds to a variable, and each edge corresponds to a non-zero entry in Θ.
- Goal: Estimate the inverse covariance matrix Θ from the observations.

Maximum Likelihood Estimator: LogDet Optimization

- Given the n samples, the likelihood is

$$P(\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}; \hat{\boldsymbol{\mu}}, \Theta) \propto \prod_{i=1}^{n} (\det \Theta)^{1/2} \exp\left(-\frac{1}{2} (\boldsymbol{y_i} - \hat{\boldsymbol{\mu}})^T \Theta (\boldsymbol{y_i} - \hat{\boldsymbol{\mu}}) \right)$$

$$= (\det \Theta)^{n/2} \exp\left(-\frac{1}{2} \sum_{i=1}^{n} (\boldsymbol{y_i} - \hat{\boldsymbol{\mu}})^T \Theta (\boldsymbol{y_i} - \hat{\boldsymbol{\mu}}) \right).$$

- The log likelihood can be written as

$$\log(P(\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}; \hat{\boldsymbol{\mu}}, \Theta)) = \frac{n}{2} \log(\det \Theta) - \frac{n}{2} \mathrm{tr}(\Theta S) + \text{constant}.$$

- Maximum likelihood estimator:

$$\Theta^* = \arg\min_{X \succ 0} \{ -\log\det X + \mathrm{tr}(SX) \}.$$

- Problem: when $p > n$, sample covariance matrix S is singular.

L1-regularized inverse covariance selection

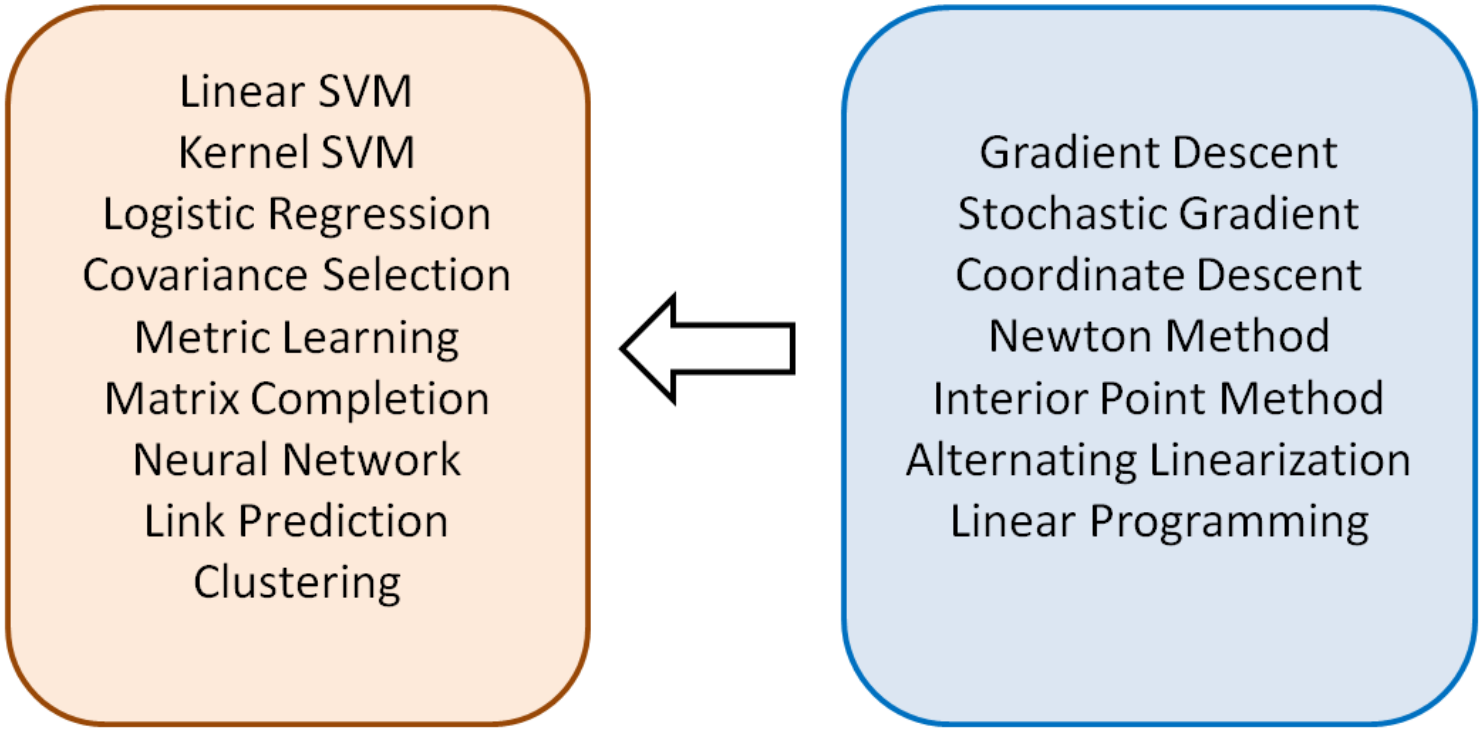
- A sparse inverse covariance matrix is preferred –

  add ℓ_1 regularization to promote sparsity.
- The resulting optimization problem:

$$\Theta = \arg\min_{X \succ 0} \left\{ -\log\det X + \operatorname{tr}(SX) + \lambda \|X\|_1 \right\} = \arg\min_{X \succ 0} f(X),$$

  where $\|X\|_1 = \sum_{i,j=1}^{n} |X_{ij}|$.
- Regularization parameter $\lambda > 0$ controls the sparsity.
- The problem appears hard to solve:
  - Non-smooth log-determinant program.
  - Number of parameters scale quadratically with number of variables.

Optimization in Machine Learning



High-dimensional!

Large number of samples!

Large number of parameters!

Exploiting Structure in Machine Learning Problems

- Regularized Loss Minimization(RLM):

$$\min_{\theta}\{\ \underbrace{g(\theta, X)}_{\text{loss}} + \underbrace{h(\theta)}_{\text{regularization}}\ \} \equiv f(\theta),$$

- Exploiting problem structure:
  - Can we have faster computation?
- Exploiting model structure:
  - Can we avoid un-important search space?
- Exploiting Data distribution:
  - Can we speed up optimization algorithms if we roughly estimate the data distribution?

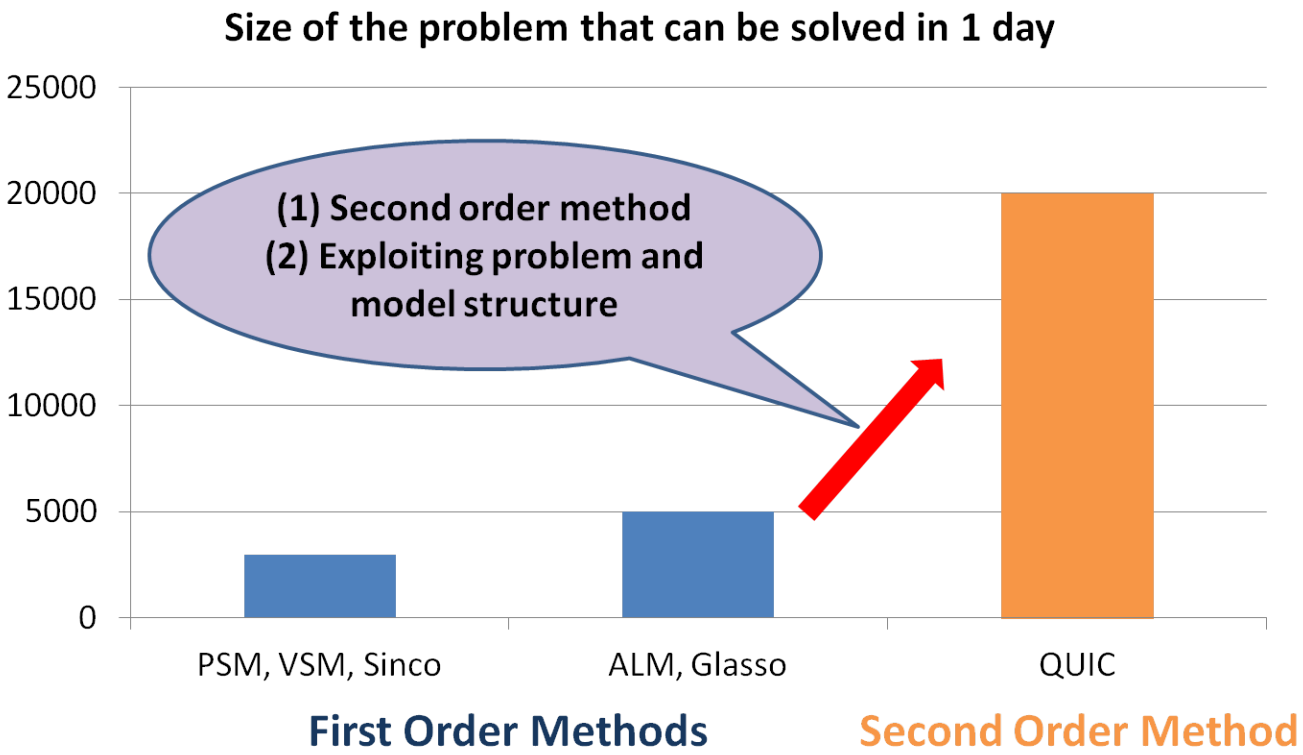
QUIC

QUadratic approximation for Inverse Covariance estimation

Sparse Inverse Covariance Estimation

- p^2 parameters in the non-smooth Log-determinant program:

$$\Theta = \arg\min_{X \succ 0} \{ \underbrace{-\log\det X + \mathrm{tr}(SX)}_{\text{negative log likelihood}} + \lambda \|X\|_1 \}$$

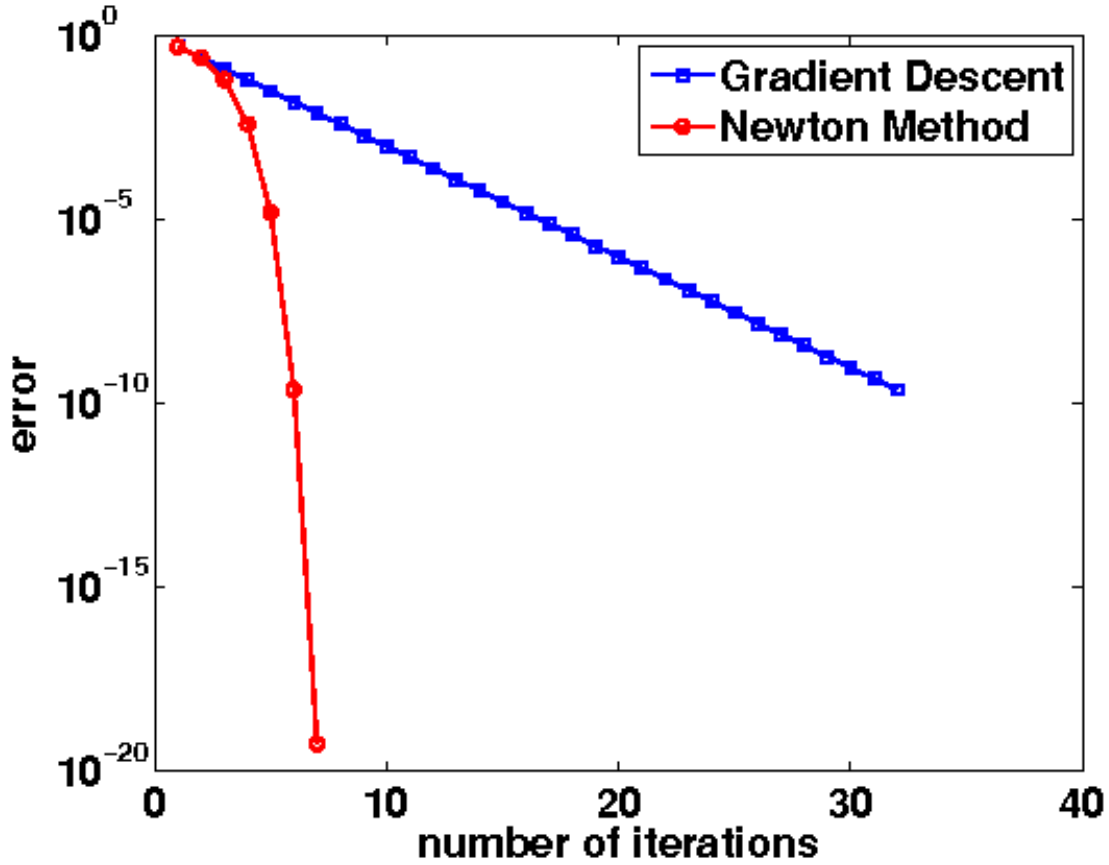


First Order Methods vs Second Order Methods

- Many algorithms have been proposed.
  - Block coordinate descent (Banerjee et al., 2007), (Friedman et al., 2007),
  - Gradient or Accelerate gradient descent (Lu, 2009), (Duchi et al., 2008),
  - Greedy coordinate descent (Scheinberg & Rich., 2009)
  - ADMM (Scheinberg et al., 2009), (Boyd et al., 2011)
- All of them are **first order methods** (only gradient information)
- QUIC: the first **second order method** for this problem (use Hessian information).

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_d} \end{bmatrix}, \quad H = \nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix}$$

First Order Methods vs Second Order Methods



Why do all previous approaches use first order methods?

Second Order Methods for Non-Smooth Functions

- Update rule for Newton method (second order method):

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta(\nabla^2 f(\boldsymbol{x}))^{-1} \nabla f(\boldsymbol{x})$$

- Two difficulties:
  - Time complexity per iteration may be very high.
  - How to deal with non-smooth functions ($\|\boldsymbol{x}\|_1$)?

Second Order Methods for Non-Smooth Functions

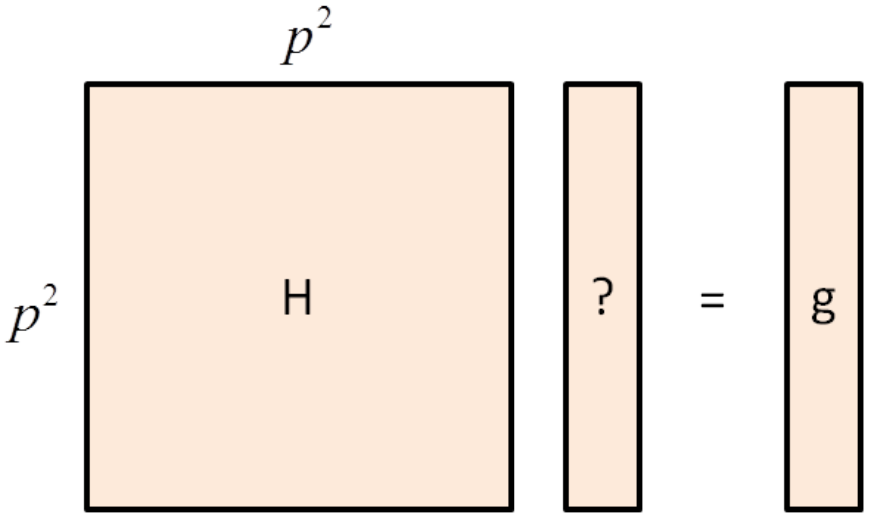
- Update rule for Newton method (second order method):

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta(\nabla^2 f(\boldsymbol{x}))^{-1} \nabla f(\boldsymbol{x})$$

- Two difficulties:
  - Time complexity per iteration may be very high.
  - How to deal with non-smooth functions ($\|\boldsymbol{x}\|_1$)?
- Addressed by (Hsieh, Sustik, Dhillon and Ravikumar, 2011).
- Extensions:
  - Theoretical Analysis: (Lee et al., 2012), (Patrinos and Bemporad., 2013), (Tang and Scheinberg, 2014), (Yen, Hsieh, Ravikumar and Dhillon, 2014), ...
  - Optimization Algorithms: (Olsen et al., 2012), (Dinh et al., 2013), (Treister et al., 2014), (Scheinberg and Tang, 2014), (Zhong et al., 2014), ...

Time Complexity?

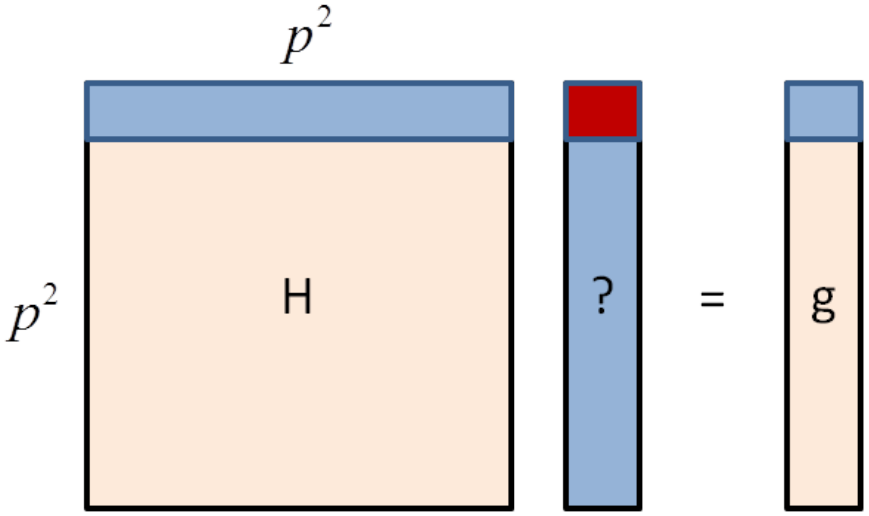
- Newton direction: $H^{-1}\boldsymbol{g}$. (H: Hessian, $\boldsymbol{g}$: gradient)
- Solving the linear system exactly: $O(p^6)$ time.



Appears to be prohibitive to apply a second order method.

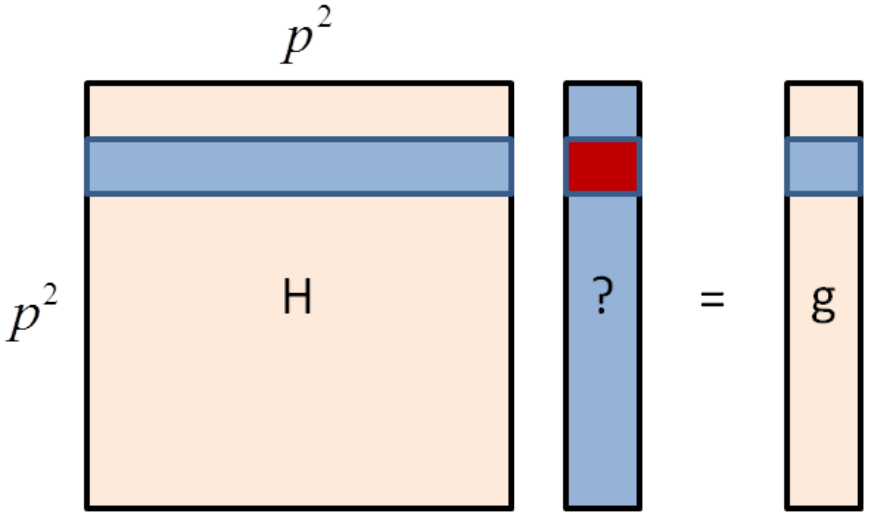
Time Complexity?

- Coordinate descent: each time fits one element of $\boldsymbol{g}$.



Time Complexity?

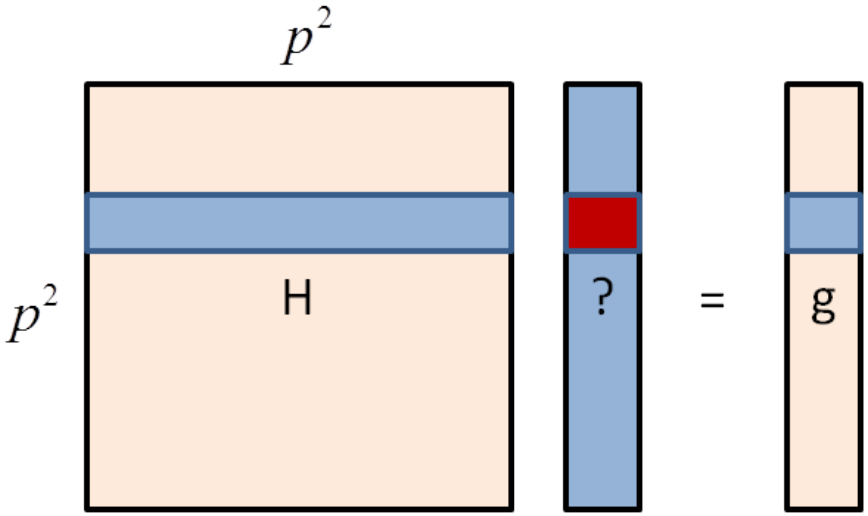
- Coordinate descent: each time fits one element of $\boldsymbol{g}$.



Time Complexity?

- Use iterative solver—coordinate descent

  $O(p^2)$ per coordinate update, $O(p^4)$ for one sweep. $\Rightarrow$ need 40 days for $p = 10^4$ on a 2.83GHz CPU.



Still prohibitive.

Exploiting Problem Structure

Fast Computation of Newton Direction

$$\min_{X \succ 0} \{ \underbrace{-\log\det X + \operatorname{tr}(SX)}_{\text{problem structure}} + \underbrace{\lambda \|X\|_1}_{\text{regularization}} \}$$

Exploiting Problem Structure

- Structure of the Hessian:

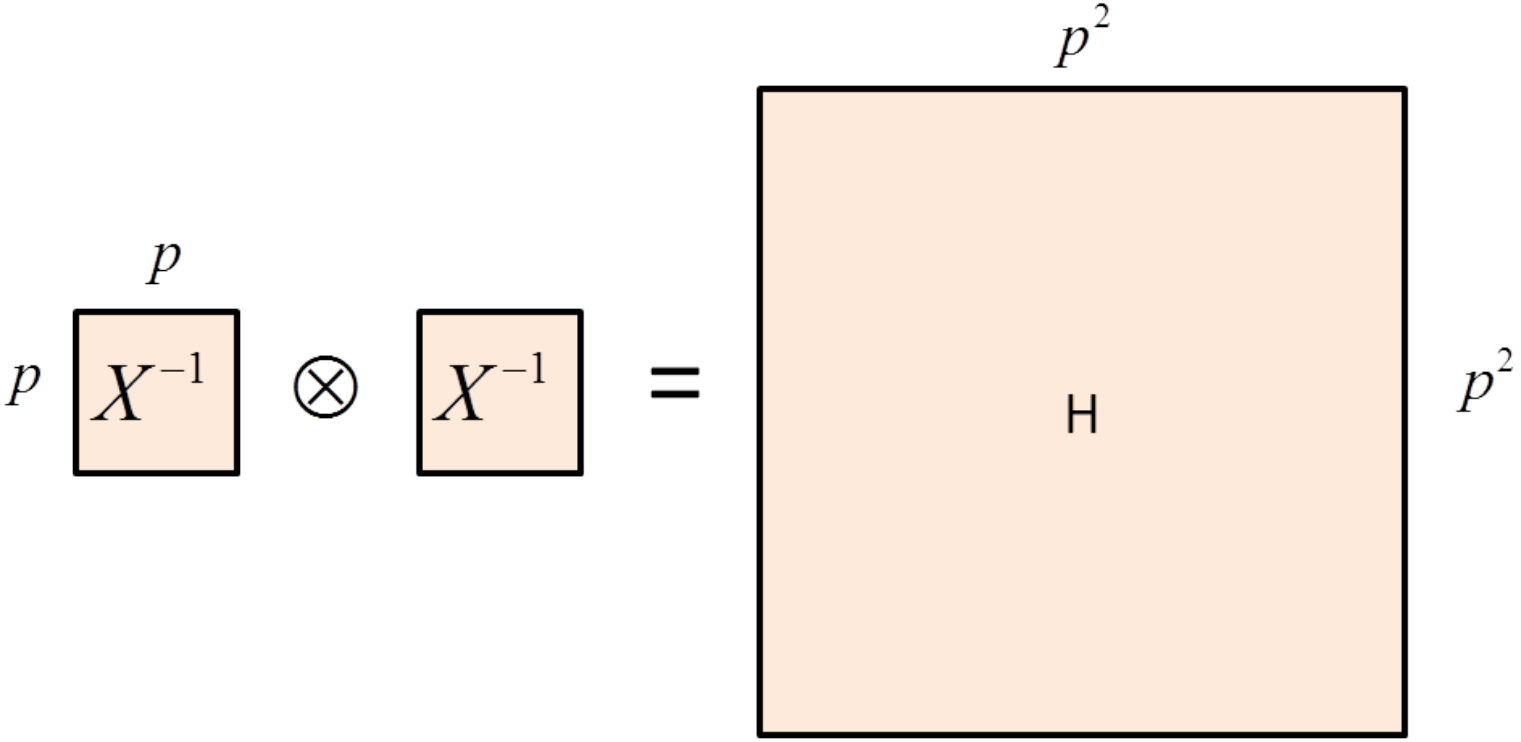
$$H = \frac{\partial^2}{\partial^2 X}(-\log\det X) = X^{-1} \otimes X^{-1}.$$

- The Kronecker product:

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1p}B \\ \vdots & \ddots & \vdots \\ a_{p1}B & \cdots & a_{pp}B \end{pmatrix}$$

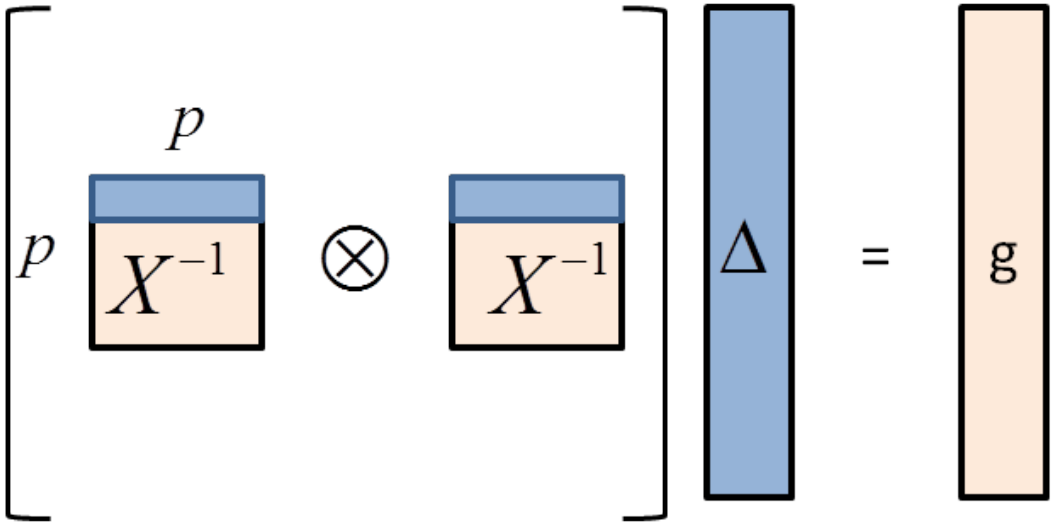
was introduced by G. Zehfuss (1858) and later attributed to L. Kronecker.

Exploiting Problem Structure



The Hessian (p^2 by p^2 matrix) can be represented using p^2 parameters!

Exploiting Problem Structure



- Time complexity: $O(p^2) \Rightarrow O(p)$ for each coordinate update.
  $O(p^4) \Rightarrow O(p^3)$ per sweep.
- At coordinate descent step to update (i, j) element of descent direction:

$$\Big(H \operatorname{vec}(\Delta)\Big)_{ip+j} = \Big((X^{-1} \otimes X^{-1}) \operatorname{vec}(\Delta)\Big)_{ip+j} = \Big(X^{-1} \Delta X^{-1}\Big)_{ij}$$
$$= (X^{-1} U)_{ij},$$

  where $U = \Delta X^{-1}$ is maintained in memory.

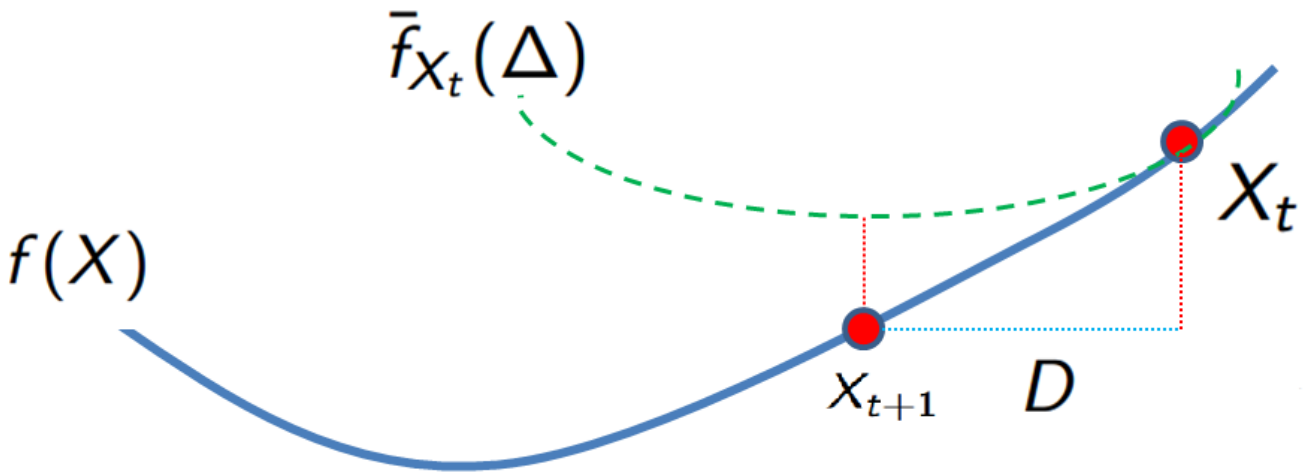
Newton Method for Non-smooth Functions

- Iteration conducts the following updates:
  - Compute the generalized Newton direction

$$D_t = \arg\min_{\Delta} \bar{f}_{X_t}(\Delta)$$

$$= \arg\min_{\Delta} \langle \nabla g(X_t), \mathrm{vec}(\Delta) \rangle + \frac{1}{2} \mathrm{vec}(\Delta)^T \nabla^2 g(X_t) \mathrm{vec}(\Delta) + \lambda \|X_t + \Delta\|_1$$

  - $X_{t+1} \leftarrow X_t + D_t$.



Proximal Newton Method

Proximal Newton for sparse Inverse Covariance estimation

For $t = 0, 1, \ldots$

1. **Compute Proximal Newton direction**: $D_t = \arg\min_{\Delta} \bar{f}_{X_t}(X_t + \Delta)$ (A Lasso problem.)
2. **Line Search**: use an Armijo-rule based step-size selection to get α s.t. $X_{t+1} = X_t + \alpha D_t$ is
   - positive definite,
   - satisfies a sufficient decrease condition $f(X_t + \alpha D_t) \leq f(X_t) + \alpha\sigma\Delta_t$.

Exploiting Problem Structure

Can we scale to $p \approx 20,000$?

- Coordinate descent updates for computing Newton direction.
  - needs $O(p^2)$ storage
  - ~~needs $O(p^4)$ computation per sweep~~
  - needs $O(p^3)$ computation per sweep (by exploiting problem structure)
- Line search (compute determinant using Cholesky factorization).
  - needs $O(p^2)$ storage
  - needs $O(p^3)$ computation

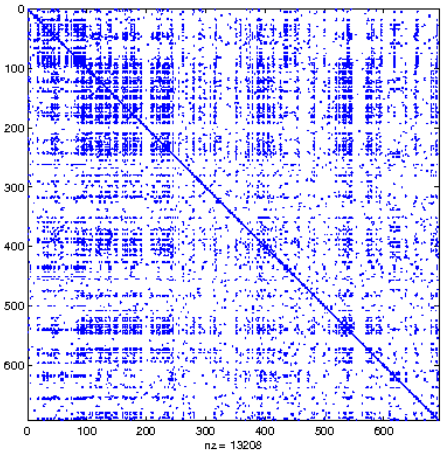
Exploiting Model Structure

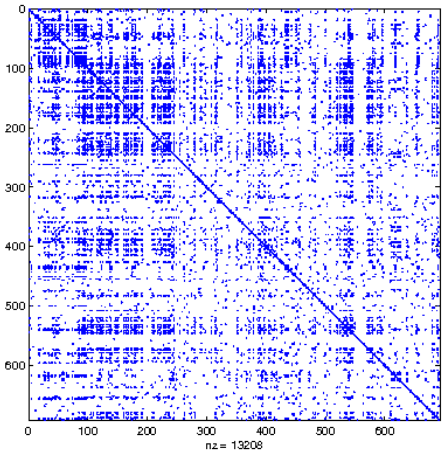
Variable Selection

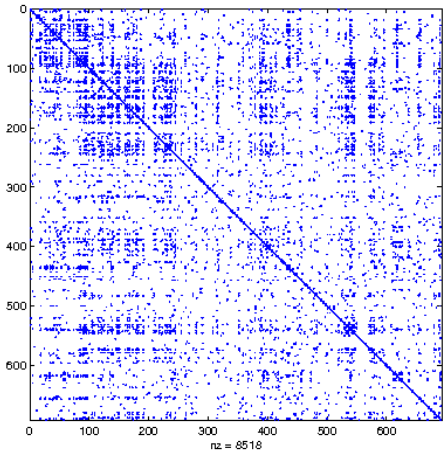
$$\min_{\theta}\{\ \underbrace{-\log\det X + \operatorname{tr}(SX)}_{\text{problem structure}}\ +\ \underbrace{h(\theta)}_{\text{regularization}}\ \} \equiv f(\theta),$$

Sparsity of the Model

ER dataset, $p = 692$, solutions at iteration 1, 3, 5.







Only 2.7% variables become nonzero.

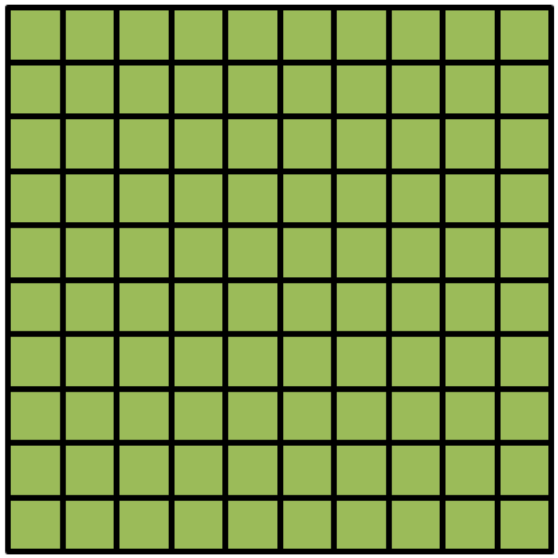
Active Variable Selection

- Strategy: before solving the Newton direction, "guess" the variables that should be updated.
- Fixed set: $S_{fixed} = \{(i,j) \mid X_{ij} = 0 \text{ and } |\nabla_{ij} g(X)| \leq \lambda.\}$.
- Only update the rest of variables (free set).
- Convergence Guaranteed.

  $p = 1869$, number of variables $= p^2 = 3.49$ million. The size of free set drops to $20,000$ very quickly.

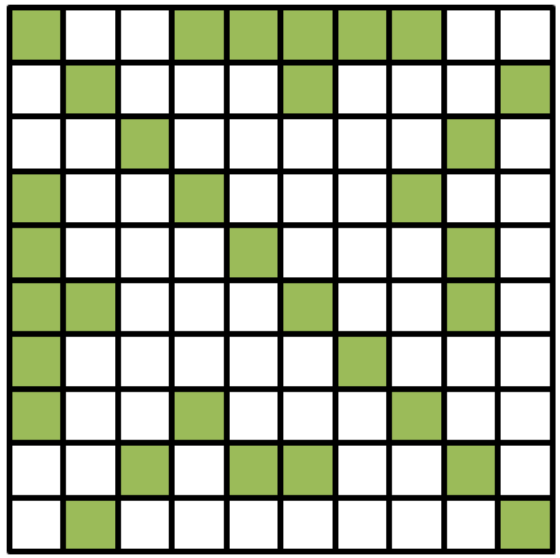
Active Variable Selection

Time complexity: $O(p^3)$ per sweep



Updates on all elements

Time complexity: $O(mp\ )$ per sweep, $m \approx \|X_t\|_0$



Updates on Free Set

Exploiting Model Structure

Can we scale to $p \approx 20,000$?

- Coordinate descent updates for computing Newton direction.
  - needs $O(p^2)$ storage
  - ~~needs $O(p^4)$ computation per sweep~~
  - ~~needs $O(p^3)$ computation per sweep (by exploiting problem structure)~~
  - needs $O(mp)$ computation per sweep, where $m \approx \|X\|_0$ (by exploiting problem and model structure)
- Line search (compute determinant using Cholesky factorization).
  - needs $O(p^2)$ storage
  - needs $O(p^3)$ computation

Algorithm

QUIC: QUadratic approximation for sparse Inverse Covariance estimation

Input: Empirical covariance matrix S, scalar λ, initial X_0.

For $t = 0, 1, \ldots$

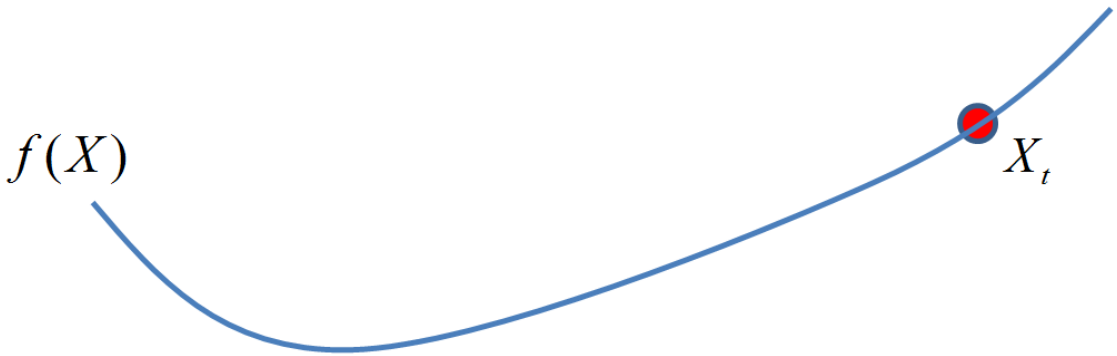
1. Variable selection: select a *free* set of $m \ll p^2$ variables.
2. Use coordinate descent to find descent direction:
   $D_t = \arg\min_\Delta \bar{f}_{X_t}(X_t + \Delta)$ over set of free variables, (A Lasso problem.)
3. Line Search: use an Armijo-rule based step-size selection to get α s.t. $X_{t+1} = X_t + \alpha D_t$ is
   - positive definite,
   - satisfies a sufficient decrease condition $f(X_t + \alpha D_t) \leq f(X_t) + \alpha\sigma\Delta_t$.

   (Cholesky factorization of $X_t + \alpha D_t$)

QUIC can solve $p = 20,000$ in about 2.3 hours.

Algorithm

- Current iterate X_t.

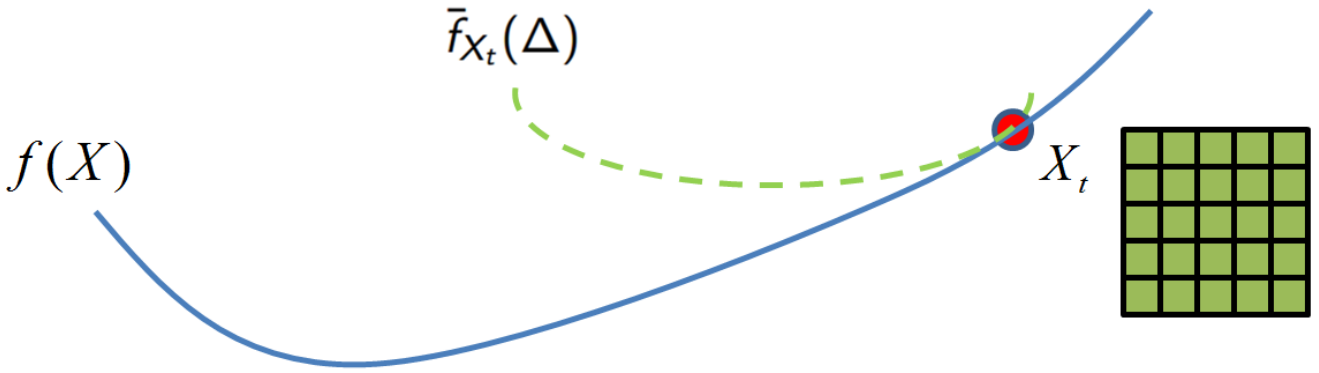


Algorithm

- Form the quadratic approximation $\bar{f}_{X_t}(\Delta)$.

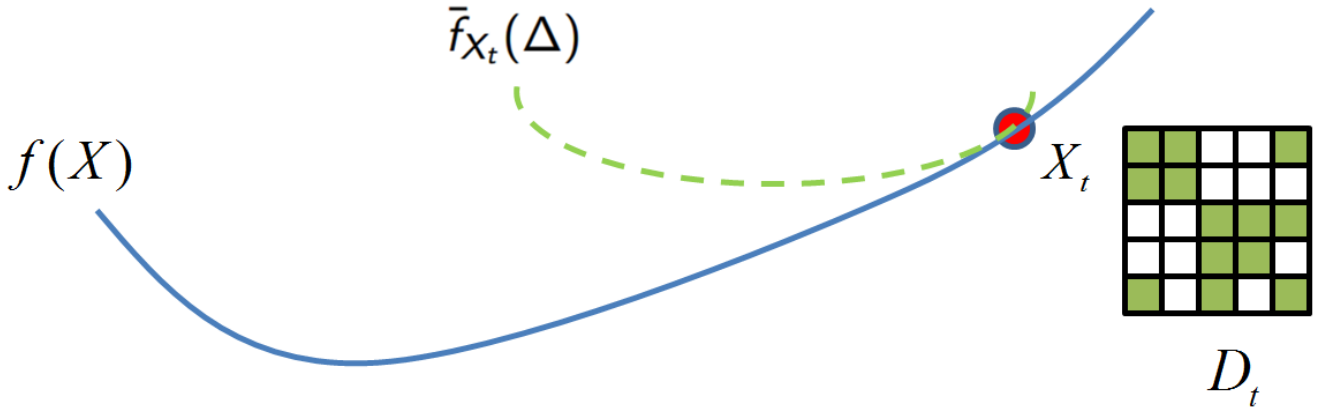
$$f(X) = g(X) + \lambda \|X\|_1$$

$$\bar{f}_{X_t}(\Delta) = f(X_t) + \langle \nabla g(X), \Delta \rangle + \frac{1}{2} \mathrm{vec}(\Delta)^T \nabla^2 g(X) \mathrm{vec}(\Delta) + \lambda \|X + \Delta\|_1$$



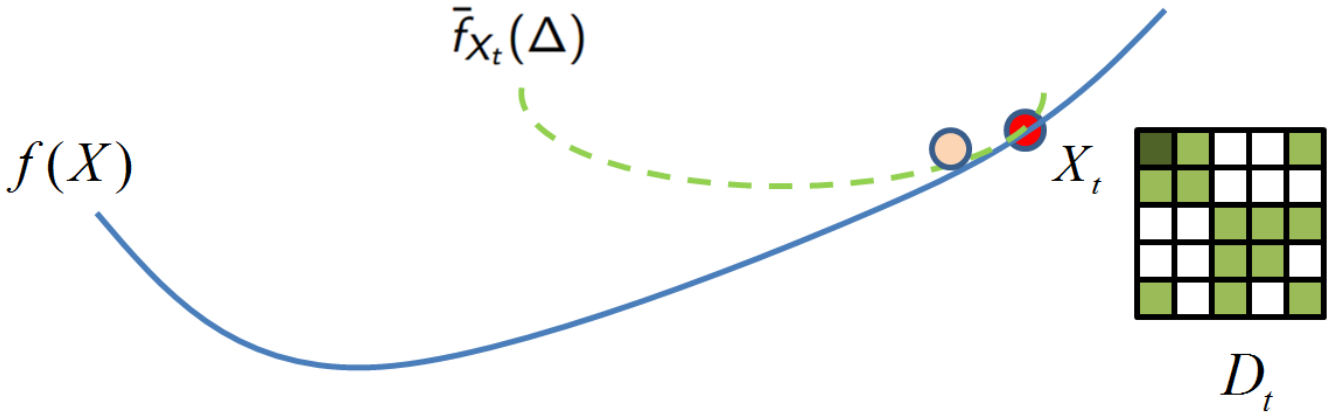
Algorithm

- Form the quadratic approximation $\bar{f}_{X_t}(\Delta)$.
- Free set selection.



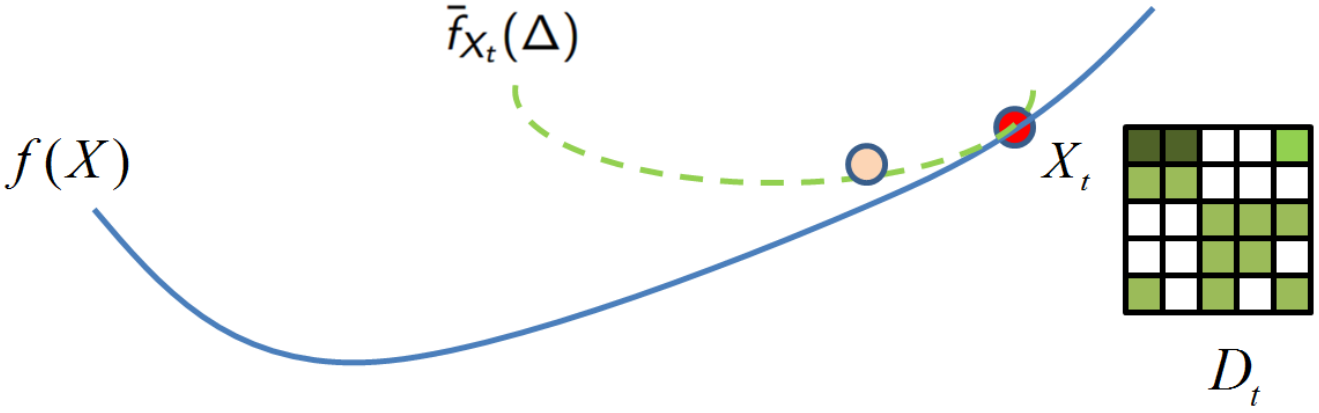
Algorithm

- Form the quadratic approximation $\bar{f}_{X_t}(\Delta)$.
- Free set selection.
- Coordinate descent to minimize $\bar{f}_{X_t}(\Delta)$



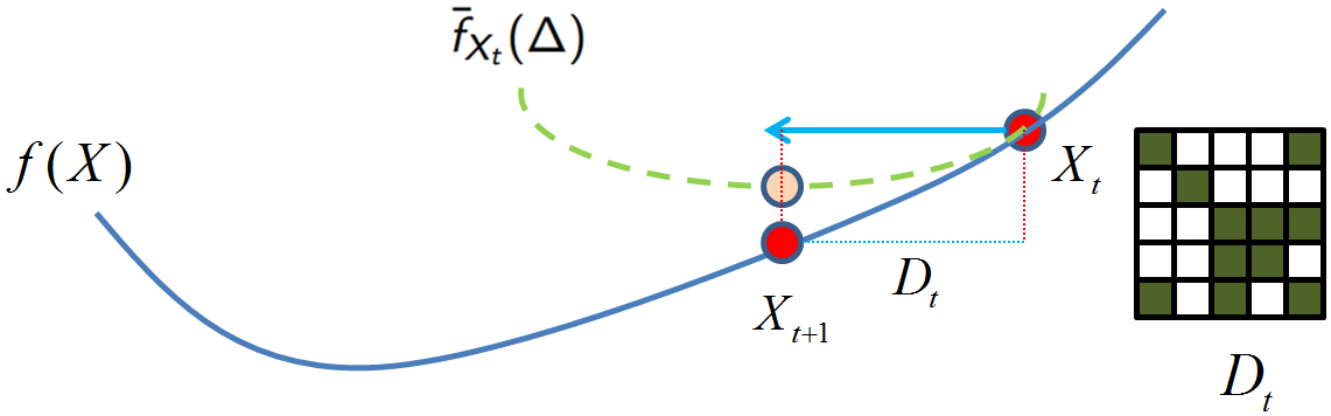
Algorithm

- Form the quadratic approximation $\bar{f}_{X_t}(\Delta)$.
- Free set selection.
- Coordinate descent to minimize $\bar{f}_{X_t}(\Delta)$



Algorithm

- Form the quadratic approximation $\bar{f}_{X_t}(\Delta)$.
- Free set selection.
- Coordinate descent to minimize $\bar{f}_{X_t}(\Delta)$
- $X_{t+1} \leftarrow X_t + D_t$



Theoretical Guarantee

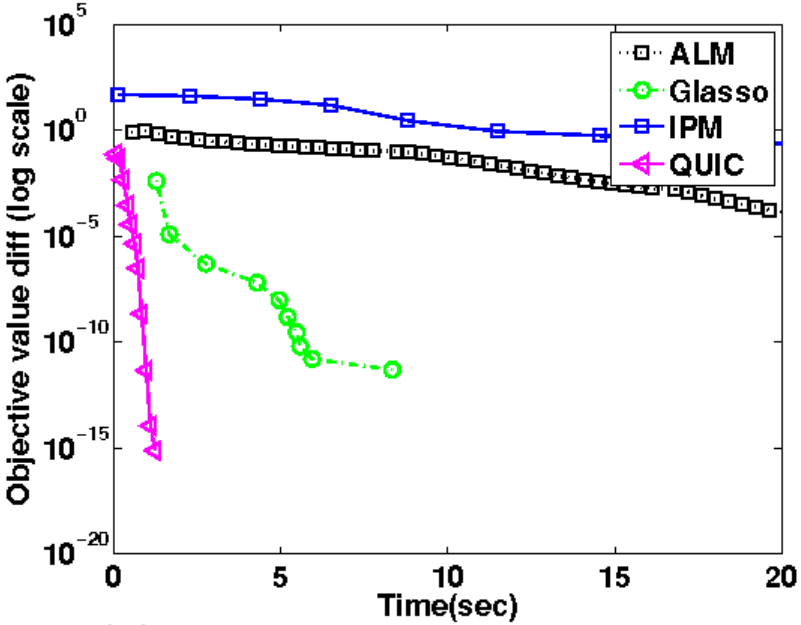
Theorem: Quadratic Convergence Rate for QUIC

Suppose $g(\cdot)$ is strongly convex and the quadratic subproblem at each iteration is solved exactly, QUIC converges to the global optimum with an asymptotic quadratic convergence rate:

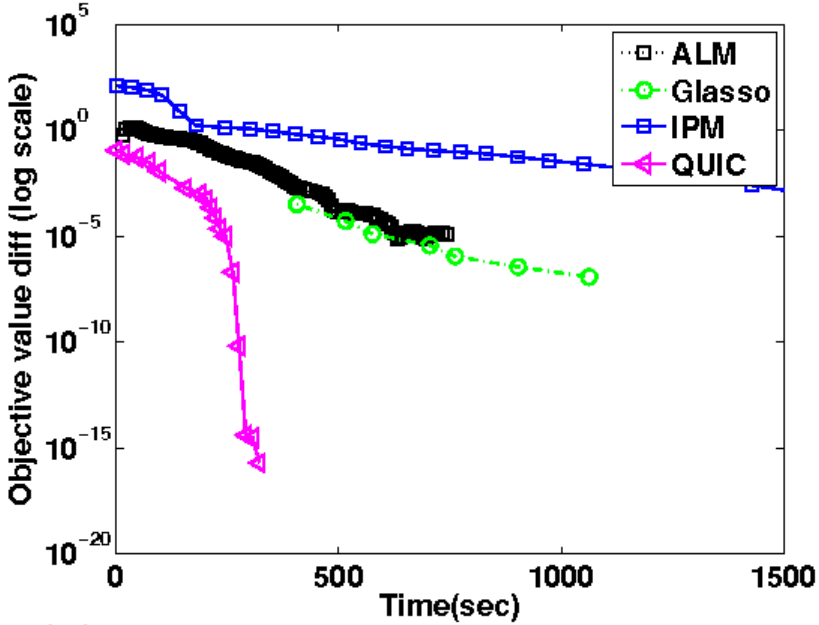
$$\lim_{t \to \infty} \frac{\|X_{t+1} - X^*\|_F}{\|X_t - X^*\|_F^2} = \kappa,$$

where κ is a constant.

QUIC Results: Data from Biology Applications



(a) Time for Estrogen, $p = 692$



(b) Time for hereditarybc, $p = 1,869$

QUIC & DIRTY

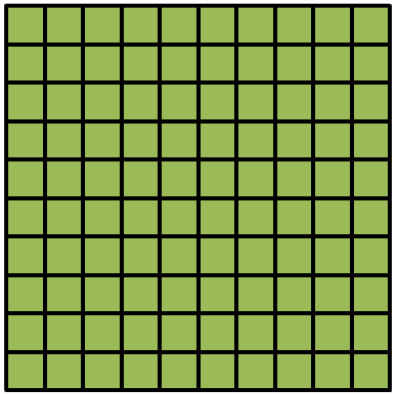
Active Subspace Selection for Dirty Statistical Models

Exploiting Model Structure

$$\min_X \{ \underbrace{\text{Loss}(X; \text{Data})}_{\text{problem structure}} + \underbrace{\text{Regularization}(X)}_{\text{model structure}} \}$$

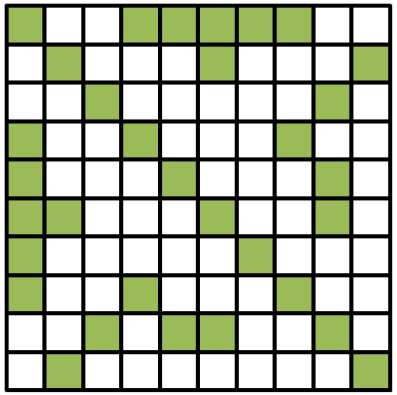
- If $h(X) = \|X\|_1$, the solution is expected to be sparse.
- Active Variable Selection:

Time complexity: $O(p^3)$ per sweep



Updates on all elements

Time complexity: $O(mp^2)$ per sweep, $m \approx \|X_t\|_0$

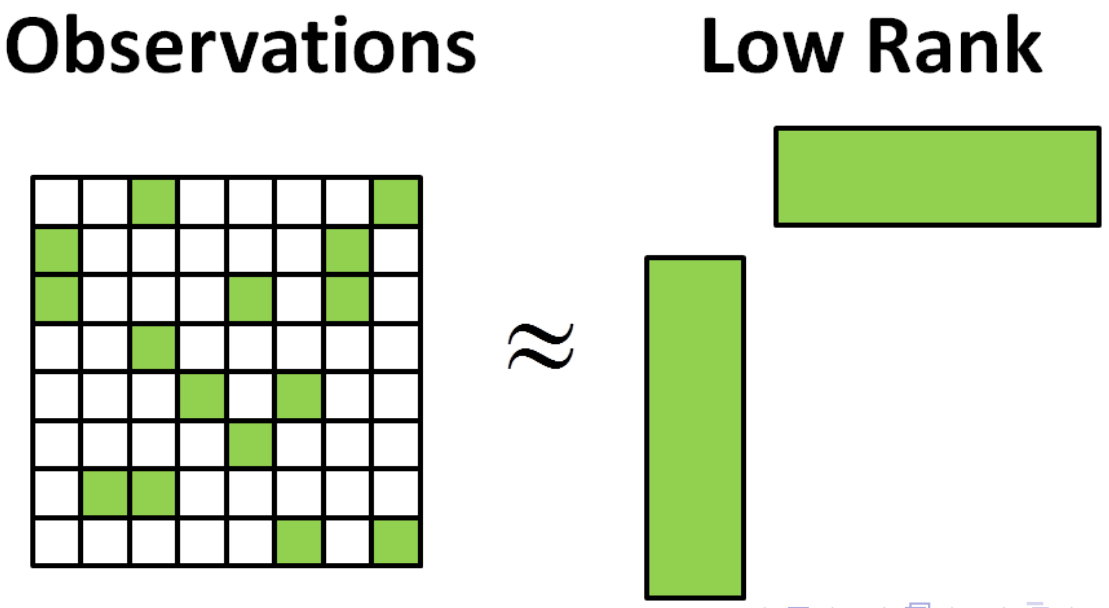


Updates on Free Set

Matrix Completion Problems

$$\min_X \{ \; g(X) \; + \; \lambda \|X\|_* \; \}$$

- $\|X\|_*$: a nuclear norm regularization to promote the low rank structure.
- Active variable selection $\Rightarrow$ Active subspace selection
  (identify the low-dimensional subspace)



Active Subspace Selection

- Given current iterate $X = U\Sigma V^T$, the "active subspace" is:

$$\mathcal{A} = \{\mathbf{u}\mathbf{v}^T \mid \mathbf{u}^T X \mathbf{v} \neq 0 \ \text{ or } \ |\mathbf{u}^T \nabla X \mathbf{v}| > \lambda\}.$$

- Step 1: Active subspace selection:
  - $U_G \Sigma_G V_G^\top = S_\lambda(X - \nabla g(X))$.
  - $U_A := \text{span}(U, U_G)$, $V_A := \text{span}(V, V_G)$.
  - $\mathcal{A} = \{\mathbf{u}\mathbf{v}^T \mid \mathbf{u} \in U_A, \mathbf{v} \in \mathcal{V}_A\}$.
- Step 2: Solve the reduced-sized problem:

$$\underset{S \in \mathbb{R}^{k \times k}}{\text{argmin}} \, \bar{g}_X(U_A S V_A^\top) + \lambda \|S\|_*,$$

  $\bar{g}_X(\cdot)$ is the quadratic approximation of $g(\cdot)$ at current iterate X
- Iterate steps 1 and 2.

- Subspace selection can be generalized to settings where the regularizer is a "decomposable norm" at the solution

Active Subspace Selection for Decomposable Norms

- A norm $\|\cdot\|$ is **decomposable** at $\boldsymbol{x}$ if there is a subspace $\mathcal{T}$ and vector $\boldsymbol{e} \in \mathcal{T}$ such that the sub differential at $\boldsymbol{x}$ is

$$\partial\|\boldsymbol{x}\|_r = \{\boldsymbol{\rho} \in \mathbb{R}^n \mid \Pi_{\mathcal{T}}(\boldsymbol{\rho}) = \boldsymbol{e} \text{ and } \|\Pi_{\mathcal{T}^\perp}(\boldsymbol{\rho})\|_r^* \leq 1\},$$

- Examples: ℓ_1 norm, nuclear norm, group lasso, ...
- **Active subspace selection**:
  - Divide the space into "fixed" subspace and "free subspace":

$$\mathcal{S}_{\text{free}} := [\mathcal{T}(\boldsymbol{\theta})] \cup [\mathcal{T}(\text{prox}_{\lambda_r}(G))], \quad \mathcal{S}_{\text{fixed}} = \mathcal{S}_{\text{free}}{}^{\perp},$$

    where $\mathcal{T}(\cdot)$ is the support of the decomposable norm.
  - Solve the reduced sized problem.

Active Subspace Selection for Dirty Models

- Superposition-structured models or "dirty models":

$$\min_{\{\boldsymbol{\theta}^{(r)}\}_{r=1}^{k}} \left\{ g\left(\sum_r \boldsymbol{\theta}^{(r)} \right) + \sum_r \lambda_r h^{(r)}(\boldsymbol{\theta}^{(r)}) \right\},$$

- Examples:
  - Sparse + low-rank graphical model learning
  - Multi-task learning (sparse + group-sparse or sparse + low-rank).
- Select active subspace for each $\boldsymbol{\theta}^{(r)}$.

Proximal Newton Method

Proximal Newton for Dirty Models (QUIC & DIRTY)

For $t = 0, 1, \ldots$

1. Compute $\bar{\boldsymbol{\theta}} = \sum_{r=1}^{k} \boldsymbol{\theta}^{(r)}$
2. Compute the current gradient $G = \nabla g(\bar{\boldsymbol{\theta}})$
3. Select free set by

$$\mathcal{S}_{\text{free}}^{(r)} := [\mathcal{T}(\boldsymbol{\theta}^{(r)})] \cup [\mathcal{T}(\text{prox}_{\lambda_r}^{(r)}(G))], \quad \mathcal{S}_{\text{fixed}}^{(r)} = \mathcal{S}_{\text{free}}^{(r)\,\perp},$$

4. For $r = 1, \ldots, k$
   - Solve the subproblem

$$\Delta^{(r)} = \underset{\boldsymbol{d} \in \mathbb{R}^n}{\text{argmin}} \langle \boldsymbol{d}, G + \sum_{t \neq r} H\Delta^{(t)} \rangle + \frac{1}{2} \boldsymbol{d}^T H \boldsymbol{d} + \lambda_r \|\boldsymbol{\theta}^{(r)} + \boldsymbol{d}\|_r.$$

5. **Line Search**: use an Armijo-rule based step-size selection to get α s.t. $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha\Delta$ sufficient decrease the objective function value.

Theoretical Guarantee

Theorem: Quadratic Convergence Rate for QUIC & DIRTY

Suppose $g(\cdot)$ is strongly convex and the quadratic subproblem at each iteration is solved exactly, QUIC & DIRTY converges to the global optimum with an asymptotic quadratic convergence rate:

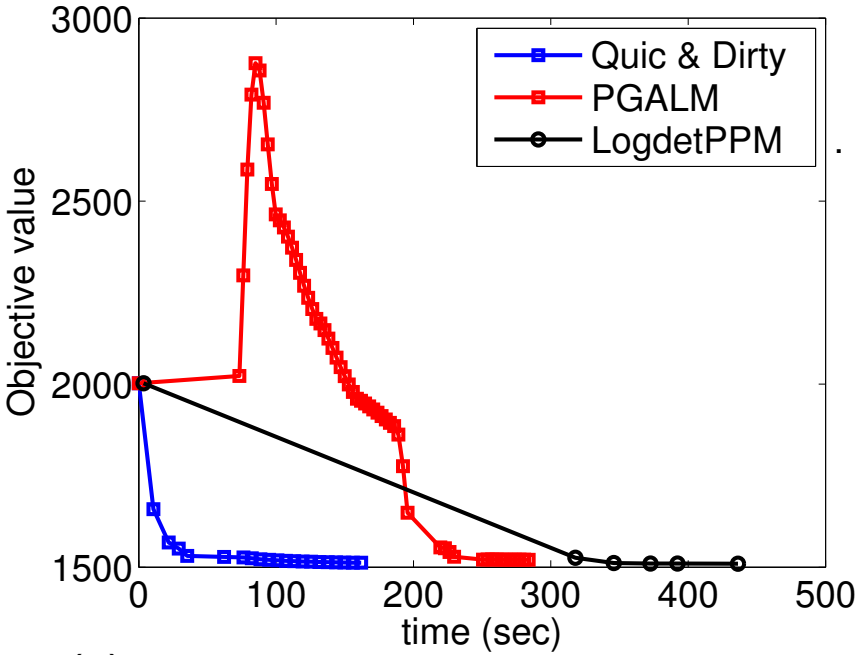
$$\lim_{t \to \infty} \frac{\|X_{t+1} - X^*\|_F}{\|X_t - X^*\|_F^2} = \kappa,$$

where κ is a constant.

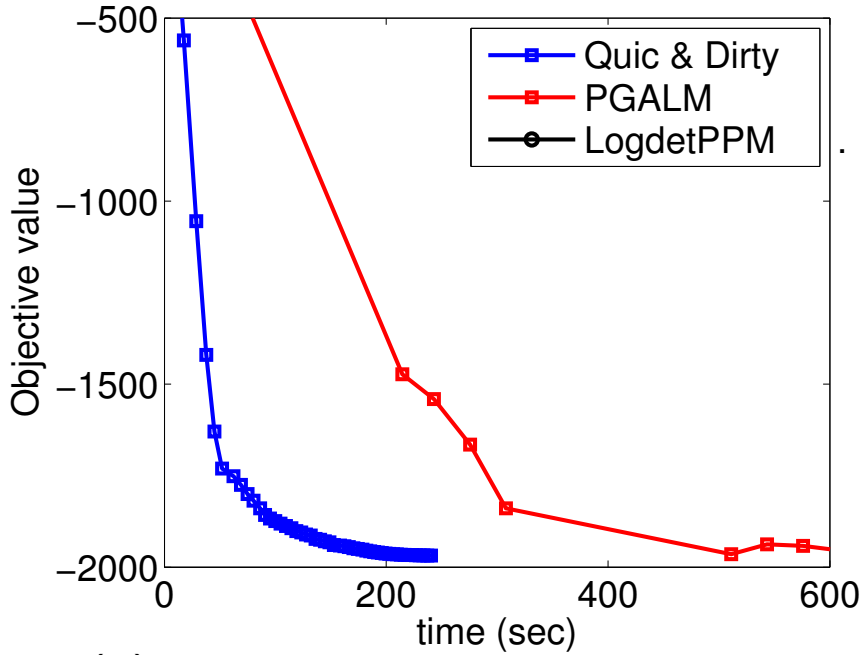
Results: Data from Biology Applications

Sparse + low rank graphical model learning

$$\min_{S,L:L\succeq 0,S-L\succ 0} -\log\det(S-L)+\langle S-L,\Sigma\rangle+\lambda_S\|S\|_1+\lambda_L\,\mathrm{tr}(L).$$



(c) Time for Leukemia, $p = 1,255$



(d) Time for Rosetta, $p = 2,000$

Application: Multi-task Learning

- Multi-task (multi-class) learning with sparse + group sparse structure.

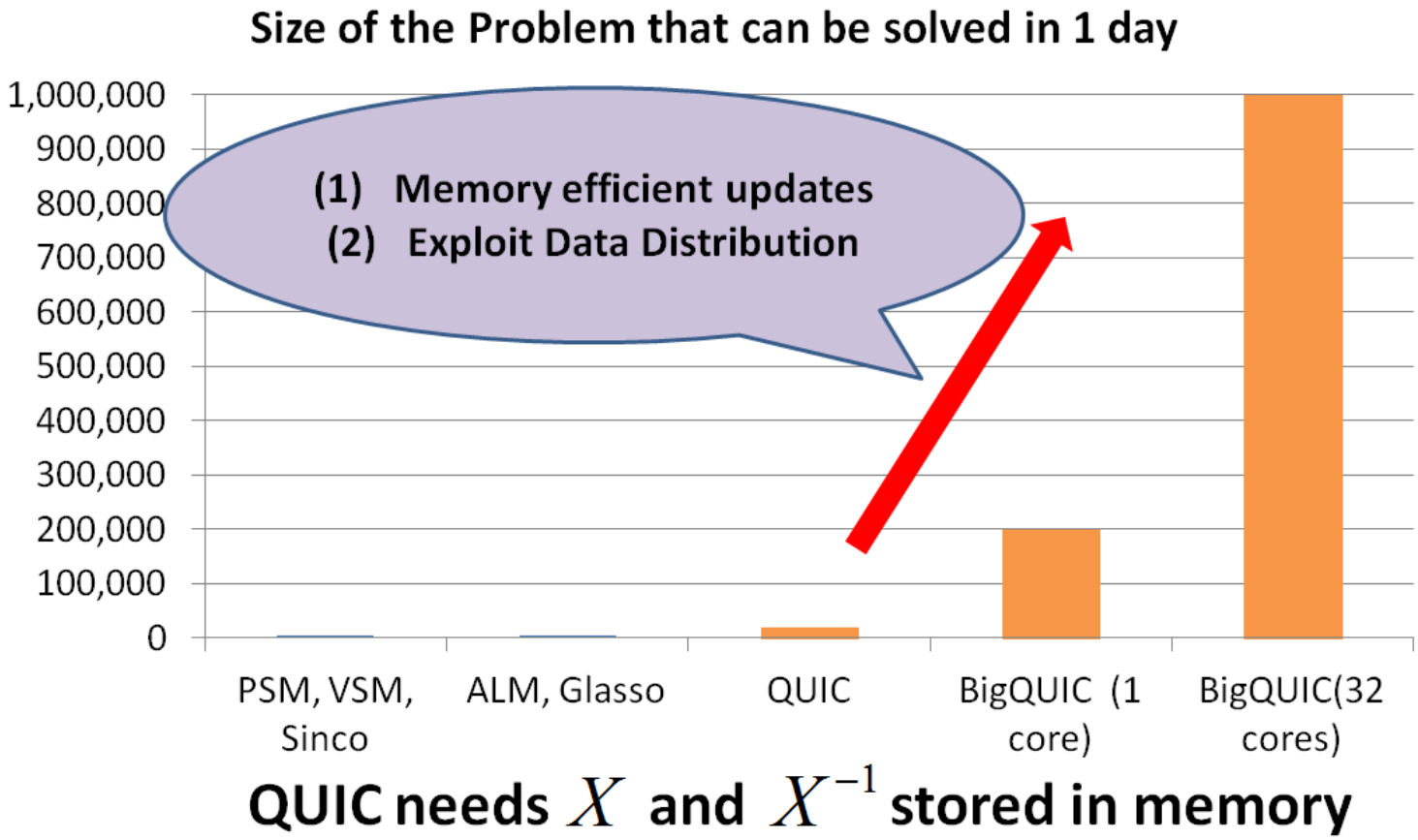
$$\sum_{r=1}^{k} \ell(y^{(r)}, X^{(r)}(S^{(r)} + B^{(r)})) + \lambda_S \|S\|_1 + \lambda_B \|B\|_{1,2},$$

dataset	number of training data	Dirty Models (sparse + group sparse)			Other Models	
		QUIC & DIRTY	proximal gradient	ADMM	Lasso	Group Lasso
USPS	100	7.47% / **0.75s**	7.49% / 10.8s	7.47% / 4.5s	10.27%	8.36%
	400	2.5% / **1.55s**	2.5% / 35.8	2.5% / 11.0s	4.87%	2.93%
RCV1	1000	18.45% / **23.1s**	18.49% / 430.8s	18.5% / 259s	22.67%	20.8%
	5000	10.27% / **87s**	10.27% / 2254s	10.27% / 1191s	13.67%	12.25%

Big & Quic

Sparse Inverse Covariance Estimation with 1 Million Random Variables

Big & Quic


Size of the Problem that can be solved in 1 day
1,000,000
900,000
800,000
700,000
600,000
500,000
400,000
300,000
200,000
100,000
0
(1) Memory efficient updates
(2) Exploit Data Distribution
PSM, VSM, Sinco
ALM, Glasso
QUIC
BigQUIC (1 core)
BigQUIC(32 cores)
QUIC needs X and X^{-1} stored in memory

Difficulties in Scaling QUIC

- Coordinate descent requires X_t and X_t^{-1},
  - needs $O(p^2)$ storage
  - needs $O(mp)$ computation per sweep, where $m = \|X_t\|_0$
- Line search (compute determinant of a big sparse matrix).
  - needs $O(p^2)$ storage
  - needs $O(p^3)$ computation

Line Search

- Given sparse matrix $A = X_t + \alpha D$, we need to
  1. Check its positive definiteness.
  2. Compute $\log\det(A)$.
- Cholesky factorization in QUIC requires $O(p^3)$ computation.
- If $A = \begin{pmatrix} a & \boldsymbol{b}^T \\ \boldsymbol{b} & C \end{pmatrix}$,
  - $\det(A) = \det(C)(a - \boldsymbol{b}^T C^{-1} \boldsymbol{b})$
  - A is positive definite iff C is positive definite and $(a - \boldsymbol{b}^T C^{-1} \boldsymbol{b}) > 0$.
- C is sparse, so can compute $C^{-1}\boldsymbol{b}$ using Conjugate Gradient (CG).
- Time complexity: $T_{CG} = O(mT)$, where T is number of CG iterations.

Difficulties in Scaling QUIC

- Coordinate descent requires X_t and X_t^{-1},
  - needs $O(p^2)$ storage
  - needs $O(mp)$ computation per sweep, where $m = \|X_t\|_0$
- Line search (compute determinant of a big sparse matrix).
  - needs $O(p)$ storage
  - needs $O(mp)$ computation

Back-of-the-envelope calculations

- Co-ordinate descent requires m computations of $\boldsymbol{w}_i^T D \boldsymbol{w}_j$.
- To solve a problem of size $p = 10^4$ (avg degree=10), QUIC takes 12 minutes
- Extrapolate: to solve a problem with $p = 10^6$, QUIC would take $12\text{min} \times 10^4 = 83.33$ days (2.6 days using 32 cores).
- p^2 memory means 8 Terabyte main memory for $p = 10^6$.
- Naive solution that has $O(m)$ memory requirement:
  - Compute $\boldsymbol{w}_i, \boldsymbol{w}_j$ by solving two linear systems by CG,
  - $O(T_{CG})$ computation per coordinate update
  - $O(mT_{CG})$ computation per sweep.
  - Would need 8,333 days to solve the problem for $p = 10^6$ (260 days using 32 cores).

Back-of-the-envelope calculations

- Co-ordinate descent requires m computations of $\boldsymbol{w}_i^T D \boldsymbol{w}_j$.
- To solve a problem of size $p = 10^4$ (avg degree=10), QUIC takes 12 minutes
- Extrapolate: to solve a problem with $p = 10^6$, QUIC would take $12\text{min} \times 10^4 = 83.33$ days (2.6 days using 32 cores).
- p^2 memory means 8 Terabyte main memory for $p = 10^6$.
- Naive solution that has $O(m)$ memory requirement:
  - Compute $\boldsymbol{w}_i, \boldsymbol{w}_j$ by solving two linear systems by CG,
  - $O(T_{CG})$ computation per coordinate update
  - $O(mT_{CG})$ computation per sweep.
  - Would need 8,333 days to solve the problem for $p = 10^6$ (260 days using 32 cores).

- BigQUIC solves problems of size $p = 10^6$ in one day on 32 cores.

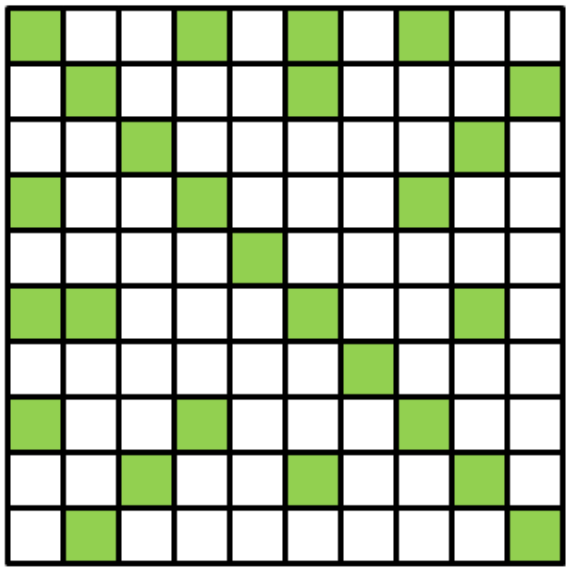
Coordinate Updates with Memory Cache

- Assume we can store M columns of W in memory.
- Coordinate descent update (i, j): compute $\boldsymbol{w}_i^T D \boldsymbol{w}_j$.
- If $\boldsymbol{w}_i, \boldsymbol{w}_j$ are not in memory: recompute by CG:

  $X\boldsymbol{w}_i = \boldsymbol{e}_i$: $O(T_{CG})$ time.

Coordinate Updates with Memory Cache

- Assume we can store M columns of W in memory.
- Coordinate descent update (i, j): compute $\boldsymbol{w}_i^T D \boldsymbol{w}_j$.
- If $\boldsymbol{w}_i, \boldsymbol{w}_j$ are not in memory: recompute by CG:
  $X\boldsymbol{w}_i = \boldsymbol{e}_i$: $O(mT_{CG})$ time.



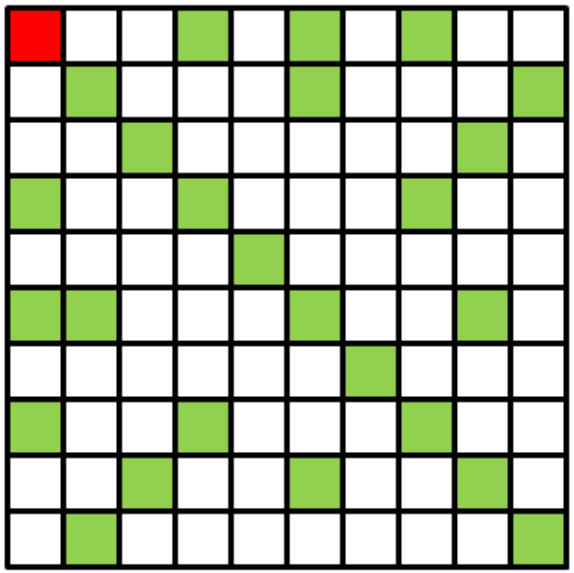
Free Set

Memory Cache
(M=4)

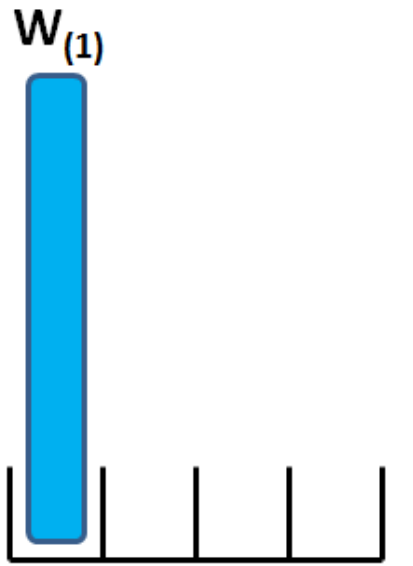
Coordinate Updates with Memory Cache

- Assume we can store M columns of W in memory.
- Coordinate descent update (i, j): compute $\boldsymbol{w}_i^T D \boldsymbol{w}_j$.
- If $\boldsymbol{w}_i, \boldsymbol{w}_j$ are not in memory: recompute by CG:

  $X\boldsymbol{w}_i = \boldsymbol{e}_i$: $O(mT_{CG})$ time.



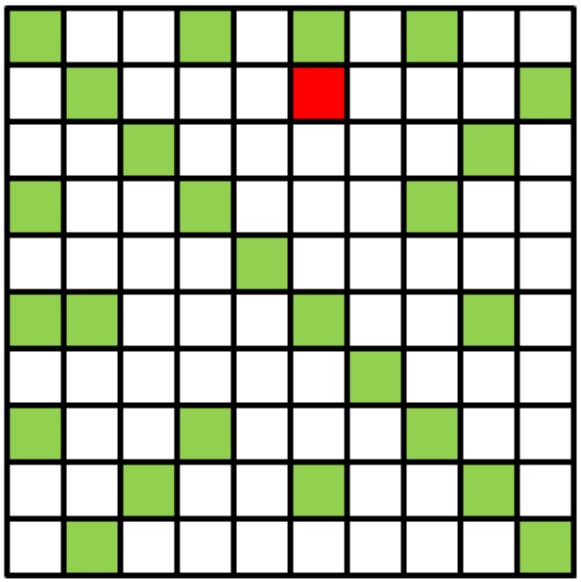
Update (1,1)



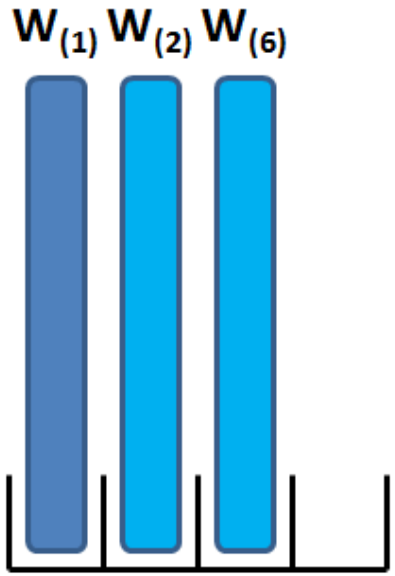
Memory Cache
(M=4)

Coordinate Updates with Memory Cache

- Assume we can store M columns of W in memory.
- Coordinate descent update (i, j): compute $\boldsymbol{w}_i^T D \boldsymbol{w}_j$.
- If $\boldsymbol{w}_i, \boldsymbol{w}_j$ are not in memory: recompute by CG:
  $X\boldsymbol{w}_i = \boldsymbol{e}_i$: $O(mT_{CG})$ time.



Update (2,6)

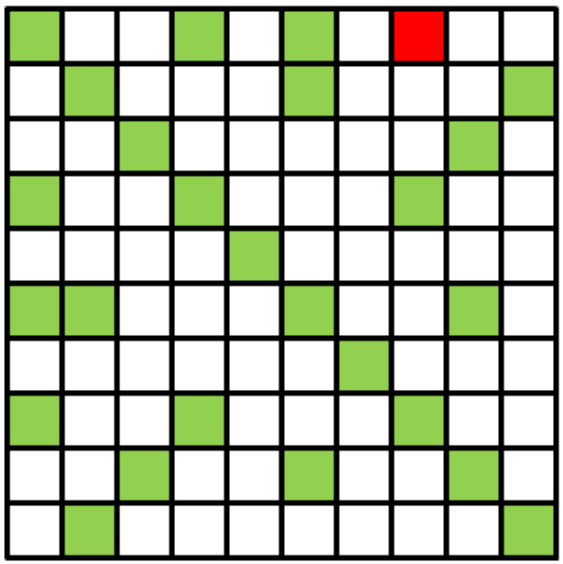


Memory Cache
(M=4)

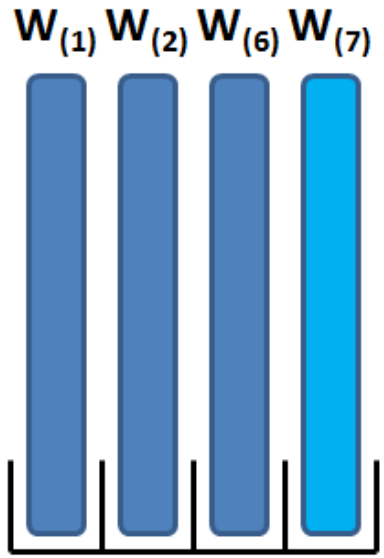
Coordinate Updates with Memory Cache

- Assume we can store M columns of W in memory.
- Coordinate descent update (i, j): compute $\boldsymbol{w}_i^T D \boldsymbol{w}_j$.
- If $\boldsymbol{w}_i, \boldsymbol{w}_j$ are not in memory: recompute by CG:

  $X\boldsymbol{w}_i = \boldsymbol{e}_i$: $O(mT_{CG})$ time.



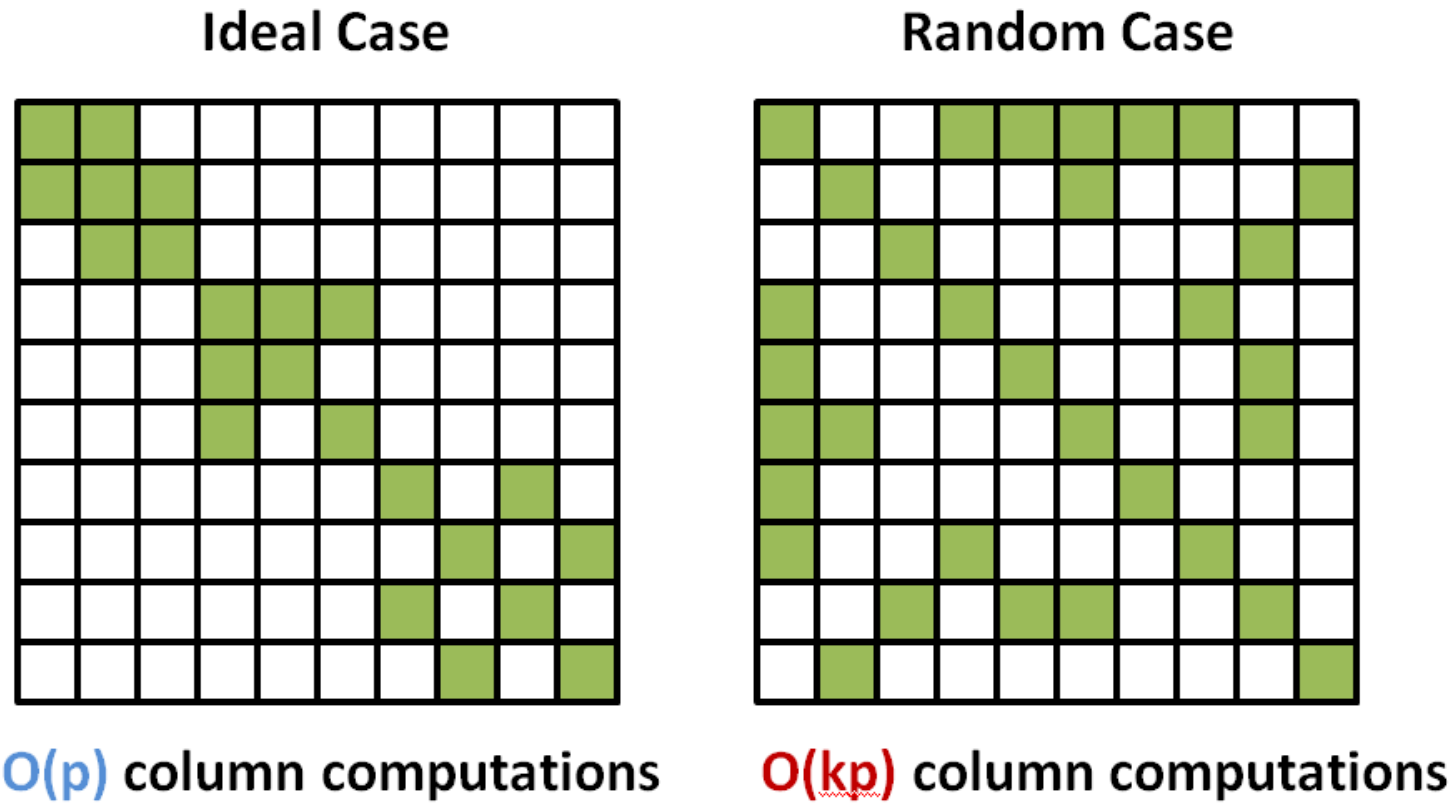
Update (1,7)



Memory Cache
(M=4)

Coordinate Updates – ideal case

- Want to find update sequence that minimizes number of cache misses: probably NP Hard.
- Our strategy: update variables **block by block** (assume $k = p/M$ blocks).



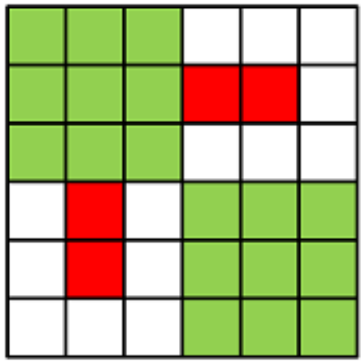
General case: block diagonal + sparse

- If the block partition is not perfect:

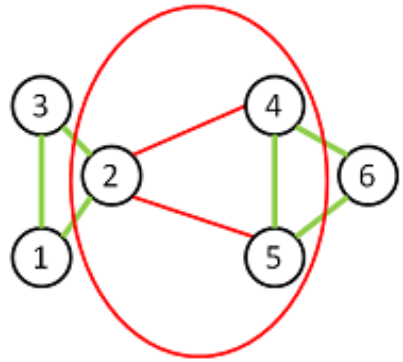
  extra column computations can be characterized by boundary nodes.
- Given a partition $\{S_1, \ldots, S_k\}$, we define boundary nodes as

$$B(S_q) \equiv \{j \mid j \in S_q \text{ and } \exists i \in S_z, z \neq q \text{ s.t. } F_{ij} = 1\},$$

where F is adjacency matrix of the free set.



Free Set



boundary nodes

Graph Clustering Algorithm

- The number of columns to be computed in one sweep is

$$p + \sum_q |B(S_q)|.$$

- Can be upper bounded by

$$p + \sum_q |B(S_q)| \leq p + \sum_{z \neq q} \sum_{i \in S_z, j \in S_q} F_{ij}.$$

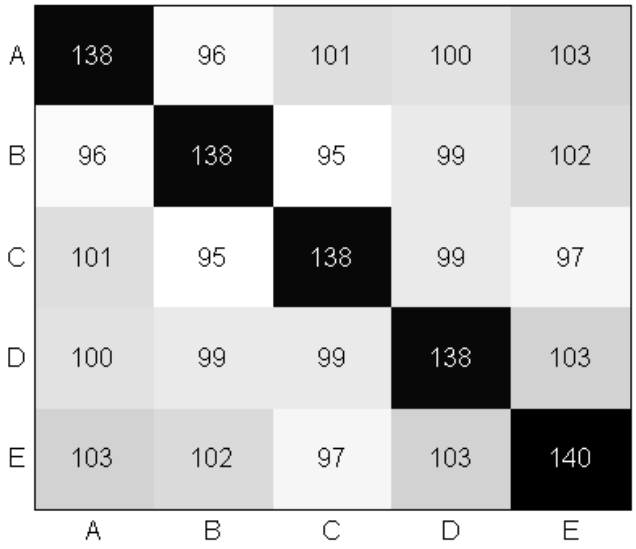
- Minimize the right hand side $\rightarrow$ minimize off-diagonal entries.

  Use Graph Clustering (METIS or Graclus) to find the partition.

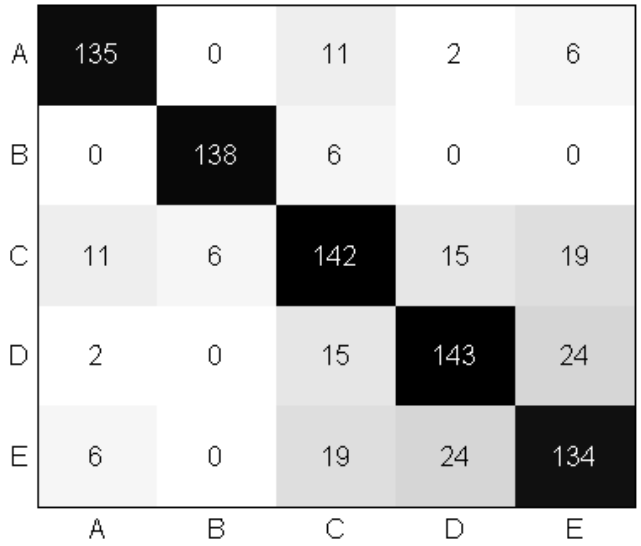
Size of boundary nodes in real datasets

ER dataset with $p = 692$.

Random partition
compute 1687 columns



Partition by clustering
compute 775 columns



$O(kpT_{CG}) \rightarrow O(pT_{CG})$ flops per sweep.

BigQUIC

- Block co-ordinate descent with clustering,
  - needs $O(m + p^2/k)$ storage
  - needs $O(mp)$ computation per sweep, where $m = \|X_t\|_0$
- Line search (compute determinant of a big sparse matrix).
  - needs $O(p)$ storage
  - needs $O(mp)$ computation

Convergence Guarantees for QUIC

Theorem (Hsieh, Sustik, Dhillon & Ravikumar, NIPS, 2011)

QUIC converges quadratically to the global optimum, that is for some constant $0 < \kappa < 1$*:*

$$\lim_{t \to \infty} \frac{\|X_{t+1} - X^*\|_F}{\|X_t - X^*\|_F^2} = \kappa.$$

BigQUIC Convergence Analysis

- Recall $W = X^{-1}$.
- When each $\boldsymbol{w}_i$ is computed by CG ($X\boldsymbol{w}_i = \boldsymbol{e}_i$):
  - The gradient $\nabla_{ij} g(X) = S_{ij} - W_{ij}$ on free set can be computed once and stored in memory.
  - Hessian ($\boldsymbol{w}_i^T D \boldsymbol{w}_j$ in coordinate updates) needs to be repeatedly computed.
- To reduce the time overhead, Hessian should be computed approximately.

BigQUIC Convergence Analysis

- Recall $W = X^{-1}$.
- When each $\boldsymbol{w}_i$ is computed by CG ($X\boldsymbol{w}_i = \boldsymbol{e}_i$):
  - The gradient $\nabla_{ij} g(X) = S_{ij} - W_{ij}$ on free set can be computed once and stored in memory.
  - Hessian ($\boldsymbol{w}_i^T D \boldsymbol{w}_j$ in coordinate updates) needs to be repeatedly computed.
- To reduce the time overhead, Hessian should be computed approximately.

Theorem (Hsieh, Sustik, Dhillon, Ravikumar & Poldrack, 2013)

If $\nabla g(X)$ is computed exactly and $\bar{\eta} I \succeq \hat{H}_t \succeq \eta I$ for some constant $\bar{\eta}, \eta > 0$ at every Newton iteration, then BigQUIC *converges to the global optimum.*

BigQUIC Convergence Analysis

- To have super-linear convergence rate, we require the Hessian computation to be more and more accurate as X_t approaches X^*.
- Measure the accuracy of Hessian computation by $R = [\boldsymbol{r}_1 \ldots, \boldsymbol{r}_p]$, where $\boldsymbol{r}_i = X\boldsymbol{w}_i - \boldsymbol{e}_i$.
- The optimality of X_t can be measured by

$$\nabla^S_{ij} f(X) = \begin{cases} \nabla_{ij} g(X) + \text{sign}(X_{ij})\lambda & \text{if } X_{ij} \neq 0, \\ \text{sign}(\nabla_{ij} g(X)) \max(|\nabla_{ij} g(X)| - \lambda, 0) & \text{if } X_{ij} = 0. \end{cases}$$

Theorem (Hsieh, Sustik, Dhillon, Ravikumar & Poldrack, 2013)

In BigQUIC, *if residual matrix* $\|R_t\| = O(\|\nabla^S f(X_t)\|^\ell)$ *for some* $0 < \ell \leq 1$ *as* $t \to \infty$, *then* $\|X_{t+1} - X^*\| = O(\|X_t - X^*\|^{1+\ell})$ *as* $t \to \infty$.

BigQUIC Convergence Analysis

- To have super-linear convergence rate, we require the Hessian computation to be more and more accurate as X_t approaches X^*.
- Measure the accuracy of Hessian computation by $R = [\boldsymbol{r}_1 \ldots, \boldsymbol{r}_p]$, where $\boldsymbol{r}_i = X\boldsymbol{w}_i - \boldsymbol{e}_i$.
- The optimality of X_t can be measured by

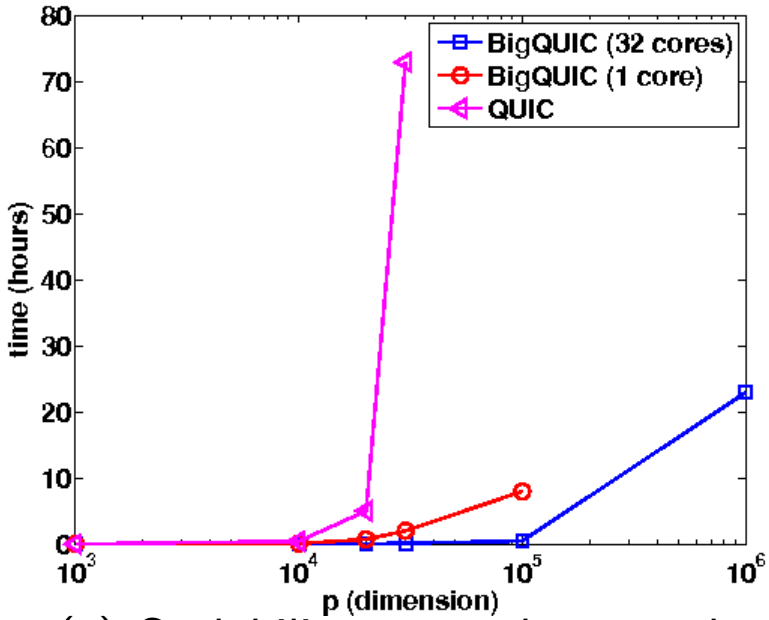
$$\nabla^S_{ij} f(X) = \begin{cases} \nabla_{ij} g(X) + \text{sign}(X_{ij})\lambda & \text{if } X_{ij} \neq 0, \\ \text{sign}(\nabla_{ij} g(X)) \max(|\nabla_{ij} g(X)| - \lambda, 0) & \text{if } X_{ij} = 0. \end{cases}$$

Theorem (Hsieh, Sustik, Dhillon, Ravikumar & Poldrack, 2013)

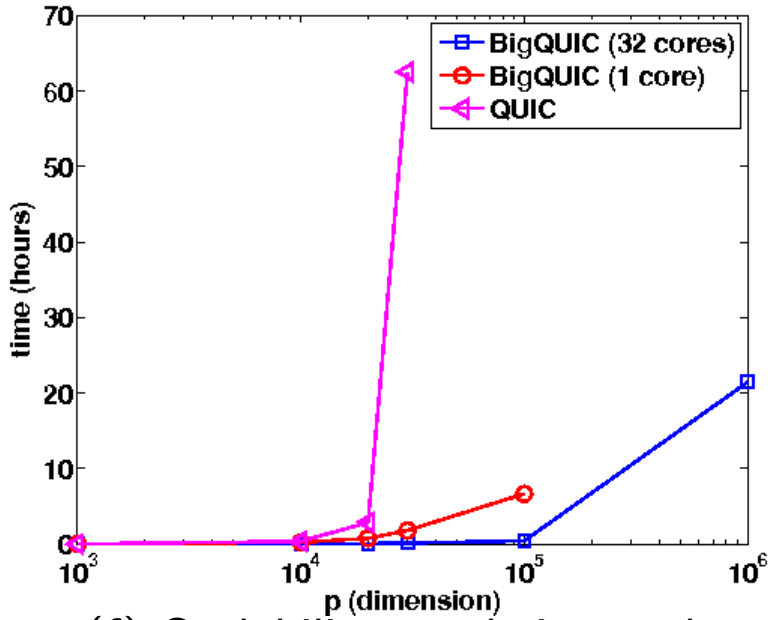
In BigQUIC, *if residual matrix* $\|R_t\| = O(\|\nabla^S f(X_t)\|^\ell)$ *for some* $0 < \ell \leq 1$ *as* $t \to \infty$, *then* $\|X_{t+1} - X^*\| = O(\|X_t - X^*\|^{1+\ell})$ *as* $t \to \infty$.

- When $\|R_t\| = O(\|\nabla^S f(X_t)\|)$, BigQUIC has asymptotic quadratic convergence rate.

Experimental results (scalability)



(e) Scalability on random graph



(f) Scalability on chain graph

Figure: BigQUIC can solve one million dimensional problems.

Experimental results

BigQUIC is faster even for medium size problems.

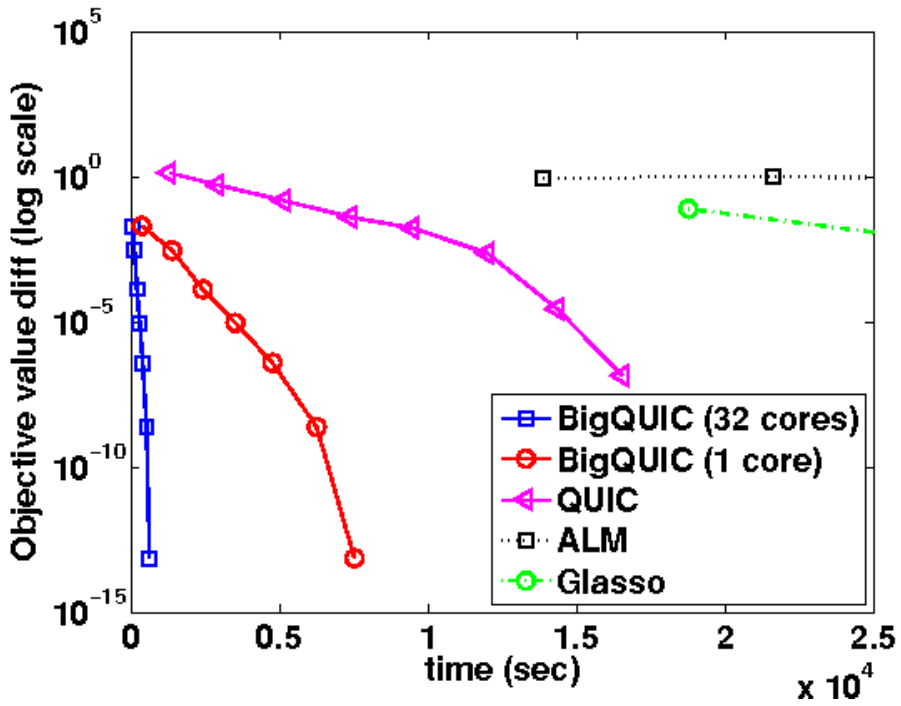
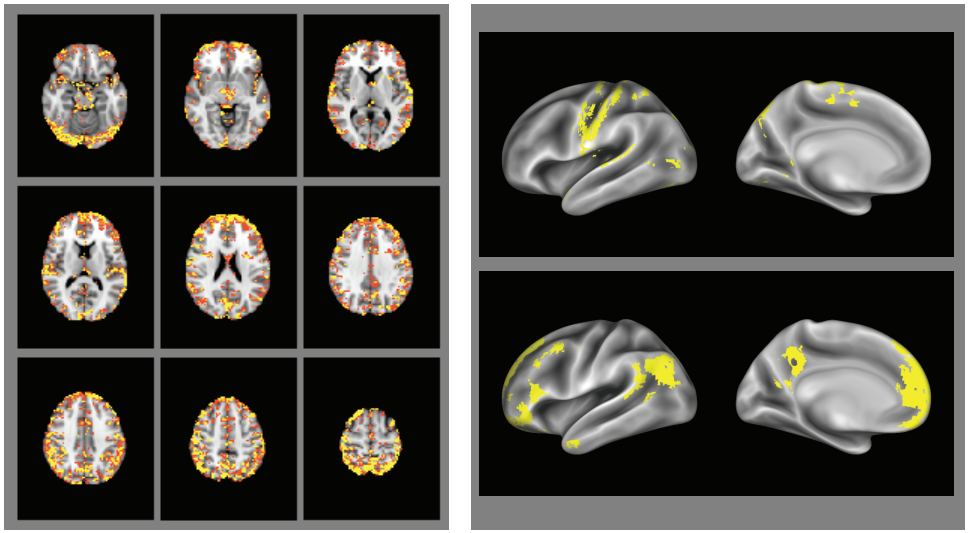


Figure: Comparison on FMRI data with a $p = 20000$ subset (maximum dimension that previous methods can handle).

Results on FMRI dataset

- 228,483 voxels, 518 time points.
- $\lambda = 0.6 \Longrightarrow$ average degree 8, BigQUIC took 5 hours.

  $\lambda = 0.5 \Longrightarrow$ average degree 38, BigQUIC took 21 hours.
- Findings:
  - Voxels with large degree were generally found in the gray matter.
  - Can detect meaningful brain modules by modularity clustering.



Conclusions

- **Proximal Newton method**: a second order method for optimizing a smooth convex function plus a non-smooth regularizer.
- How to develop an efficient proximal Newton method?

  Exploit problem structure and geometry of problem
- For sparse inverse covariance estimation problem, our proposed algorithm (BigQuic) can solve a 1-million dimensional problem in 1-day on a single 32-core machine.
- Main ingredients for BigQuic:
  - Fast coordinate descent solver for computing Newton direction.
  - Avoid un-important updates by **active subspace selection**.
  - Memory efficient coordinate descent updates.
- Quic & Dirty: extension to "Dirty" Statistical Models.

Future Work

- "Effective" order of Newton-like methods for specific problems?
- Multi-core Computation? "Wild" Co-ordinate Descent?
- Prioritized Scheduling (Co-ordinate Descent)?
- Suitability for Distributed for Out-of-core Computing?

- Develop scalable & reliable software
- No dearth of interesting applications!

References

[1] C. J. Hsieh, I. S. Dhillon, P. Ravikumar, S. Becker, and P. Olsen. *QUIC & DIRTY: A Quadratic Approximation Approach for Dirty Statistical Models* . NIPS, 2014.

[2] C. J. Hsieh, M. Sustik, I. S. Dhillon, P. Ravikumar, and R. Poldrack. *BIG & QUIC: Sparse inverse covariance estimation for a million variables* . NIPS (oral presentation), 2013.

[3] C. J. Hsieh, M. Sustik, I. S. Dhillon, and P. Ravikumar. *Sparse Inverse Covariance Matrix Estimation using Quadratic Approximation*. NIPS, 2011.

[4] C. J. Hsieh, I. S. Dhillon, P. Ravikumar, A. Banerjee. *A Divide-and-Conquer Procedure for Sparse Inverse Covariance Estimation*. NIPS, 2012.

[5] P. A. Olsen, F. Oztoprak, J. Nocedal, and S. J. Rennie. *Newton-Like Methods for Sparse Inverse Covariance Estimation*. Optimization Online, 2012.

[6] O. Banerjee, L. El Ghaoui, and A. d'Aspremont *Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data*. JMLR, 2008.

[7] J. Friedman, T. Hastie, and R. Tibshirani. *Sparse inverse covariance estimation with the graphical lasso*. Biostatistics, 2008.

[8] L. Li and K.-C. Toh. *An inexact interior point method for l1-reguarlized sparse covariance selection*. Mathematical Programming Computation, 2010.

[9] K. Scheinberg, S. Ma, and D. Glodfarb. *Sparse inverse covariance selection via alternating linearization methods*. NIPS, 2010.

[10] K. Scheinberg and I. Rish. *Learning sparse Gaussian Markov networks using a greedy coordinate ascent approach*. Machine Learning and Knowledge Discovery in Databases, 2010.