BIG & QUIC: Sparse Inverse Covariance Estimation for a Million Variables

Cho-Jui Hsieh
The University of Texas at Austin

NIPS
Lake Tahoe, Nevada
Dec 8, 2013

Joint work with M. Sustik, I. Dhillon, P. Ravikumar and R. Poldrack

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)
- $p = 228,483$ voxels.

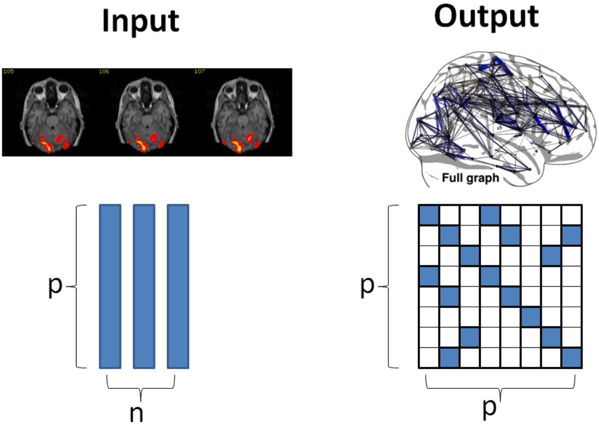


Figure from (Varoquaux et al, 2010)

Other Applications

- Gene regulatory network discovery:

  (Schafer & Strimmer 2005; Andrei & Kendziorski 2009; Menendez et al, 2010; Yin and Li, 2011)
- 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 $\{\mathbf{y_1}, \ldots, \mathbf{y_n}\}$, $\mathbf{y}_i \in R^p$, $\mathbf{y_i} \sim \mathcal{N}(\mu, \Sigma)$,
- An example – Chain graph: $y_j = 0.5y_{j-1} + \mathcal{N}(0, 1)$

$$\Sigma = \begin{pmatrix} 1.33 & 0.67 & 0.33 & 0.17 \\ 0.67 & 1.33 & 0.67 & 0.33 \\ 0.33 & 0.67 & 1.33 & 0.67 \\ 0.17 & 0.33 & 0.67 & 1.33 \end{pmatrix}, \; \Sigma^{-1} = \begin{pmatrix} 1 & -0.5 & 0 & 0 \\ -0.5 & 1.25 & -0.5 & 0 \\ 0 & -0.5 & 1.25 & -0.5 \\ 0 & 0 & -0.5 & 1 \end{pmatrix}$$

- Conditional independence is reflected as zeros in Σ^{-1}:

$\Sigma^{-1}_{ij} = 0 \Leftrightarrow$ y_i and y_j are conditionally independent given other variables.

L1-regularized inverse covariance selection

- Goal: Estimate the inverse covariance matrix in the high dimensional setting: p(# variables) $\gg n$(# samples)
- Add ℓ_1 regularization – a sparse inverse covriance matrix is preferred.
- The ℓ_1-regularized Maximum Likelihood Estimator:

$$\Sigma^{-1} = \arg\min_{X \succ 0} \Big\{ \underbrace{-\log\det X + \operatorname{tr}(SX)}_{\text{negative log likelihood}} + \lambda \|X\|_1 \Big\} = \arg\min_{X \succ 0} f(X),$$

where $\|X\|_1 = \sum_{i,j=1}^{n} |X_{ij}|$.

L1-regularized inverse covariance selection

- Goal: Estimate the inverse covariance matrix in the high dimensional setting: p(# variables) $\gg$ n(# samples)
- Add ℓ_1 regularization – a sparse inverse covriance matrix is preferred.
- The ℓ_1-regularized Maximum Likelihood Estimator:

$$\Sigma^{-1} = \arg\min_{X \succ 0} \big\{ \underbrace{-\log\det X + \operatorname{tr}(SX)}_{\text{negative log likelihood}} + \lambda \|X\|_1 \big\} = \arg\min_{X \succ 0} f(X),$$

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

Scalability

- Block coordinate ascent (Banerjee et al, 2007), Graphical Lasso (Friedman et al, 2007).
- VSM, PSM, SINCO, IPM, PQN, ALM (2008-2010).

  ALM solves $p = 1000$ in 300 secs.
- QUIC: Newton type method (Hsieh et al, 2011)

  Solves $p = 1000$ in 10 secs, $p = 10,000$ in half hour.
- All the above methods require $O(p^2)$ memory, cannot solve problems with $p > 30,000$.
- Need for scalability: FMRI dataset has more than $220,000$ variables

Scalability

- Block coordinate ascent (Banerjee et al, 2007), Graphical Lasso (Friedman et al, 2007).
- VSM, PSM, SINCO, IPM, PQN, ALM (2008-2010).

  ALM solves $p = 1000$ in 300 secs.
- QUIC: Newton type method (Hsieh et al, 2011)

  Solves $p = 1000$ in 10 secs, $p = 10,000$ in half hour.
- All the above methods require $O(p^2)$ memory, cannot solve problems with $p > 30,000$.
- Need for scalability: FMRI dataset has more than $220,000$ variables
- BigQUIC (2013):

  $p = 1,000,000$ (1 trillion parameters) in 22.9 hrs with 32 GBytes memory (using a single machine with 32 cores).

Our innovations

- Main Ingredients:
  1. Second-order Newton-like method (QUIC)
     $\longrightarrow$ quadratic convergence rate.
  2. Memory-efficient scheme using block coordinate descent (BigQUIC)
     $\longrightarrow$ scale to one million variables.
  3. Approximate Hessian computation (BigQUIC)
     $\longrightarrow$ super-linear convergence rate.

QUIC – proximal Newton method

- Split smooth and non-smooth terms: $f(X) = g(X) + h(X)$, where

$$g(X) = -\log\det X + \mathrm{tr}(SX) \text{ and } h(X) = \lambda\|X\|_1.$$

- Form quadratic approximation for $g(X_t + \Delta)$:

$$\begin{aligned}\bar{g}_{X_t}(\Delta) = &\mathrm{tr}((S - W_t)\Delta) + (1/2)\,\mathrm{vec}(\Delta)^T (W_t \otimes W_t)\,\mathrm{vec}(\Delta) \\ &- \log\det X_t + \mathrm{tr}(SX_t),\end{aligned}$$

where $W_t = (X_t)^{-1} = \frac{\partial}{\partial X} \log\det(X) \mid_{X=X_t}$.

- Define the generalized Newton direction:

$$D_t = \arg\min_{\Delta} \bar{g}_{X_t}(\Delta) + \lambda\|X_t + \Delta\|_1.$$

- Solve by coordinate descent (Hsieh et al, 2011) or other methods (Olsen et al, 2012).

Coordinate Descent Updates

- Use coordinate descent to solve:

$$\arg\min_{D}\{\bar{g}_X(D) + \lambda\|X + D\|_1\}.$$

- Closed form solution for each coordinate descent update:

$$D_{ij} \leftarrow -c + \mathcal{S}(c - b/a, \lambda/a),$$

where $\mathcal{S}(z, r) = \text{sign}(z)\max\{|z| - r, 0\}$ is the soft-thresholding function, $a = W_{ij}^2 + W_{ii}W_{jj}$, $b = S_{ij} - W_{ij} + \mathbf{w}_i^T D\mathbf{w}_j$, and $c = X_{ij} + D_{ij}$.

- The main cost is in computing $\mathbf{w}_i^T D\mathbf{w}_j$,

  where $\mathbf{w}_i, \mathbf{w}_j$ are i-th and j-th columns of $W = X^{-1}$.

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 $X_t + \alpha D_t$)

Difficulties in Scaling QUIC

Consider the case that $p \approx 1$million, $m = \|X_t\|_0 \approx 50$million.

- Coordinate descent requires X_t and $W = X_t^{-1}$,
  - needs $O(p^2)$ storage
  - needs $O(mp)$ computation per sweep, where $m = \|X_t\|_0$
- Line search (compute determinant using Cholesky factorization).
  - needs $O(p^2)$ storage
  - needs $O(p^3)$ computation

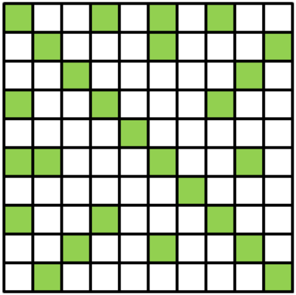
Coordinate Updates with Memory Cache

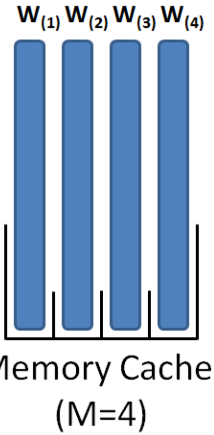
- Assume we can store M columns of W in memory.
- Coordinate descent update (i, j): compute $\mathbf{w}_i^T D \mathbf{w}_j$.
- If $\mathbf{w}_i, \mathbf{w}_j$ are not in memory: recompute by CG:

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

Coordinate Updates with Memory Cache

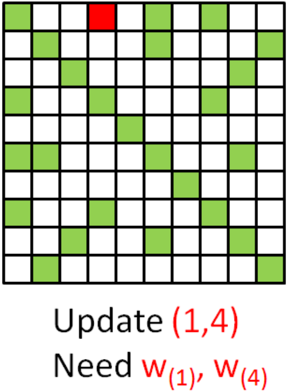
$\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4$ stored in memory.

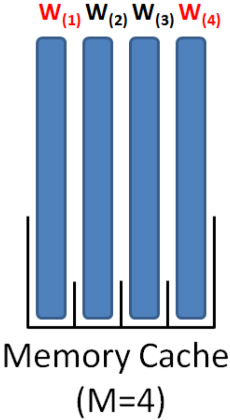




Coordinate Updates with Memory Cache

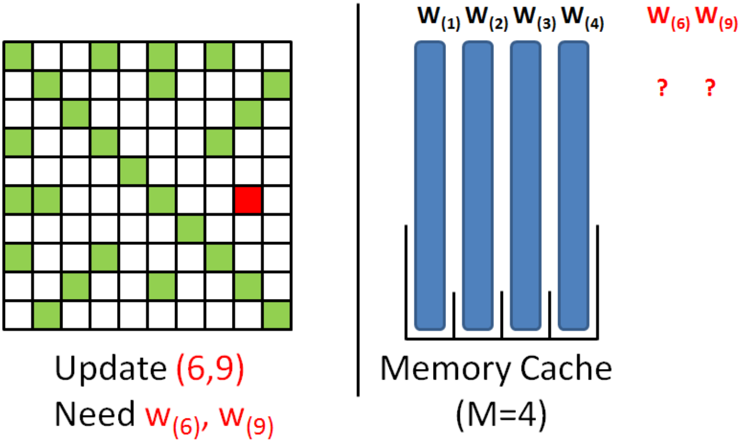
Cache hit, do not need to recompute $\mathbf{w}_i, \mathbf{w}_j$.





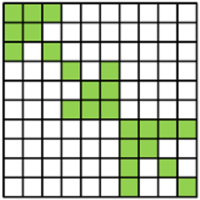
Coordinate Updates with Memory Cache

Cache miss, recompute $\mathbf{w}_i, \mathbf{w}_j$.



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**.
- The ideal case: there exists a partition $\{S_1, \ldots, S_k\}$ such that all free sets are in diagonal blocks:



Free Set

- Only requires p column evaluations.

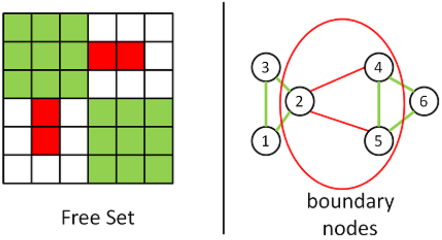
General case: block diagonal + sparse

- 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.



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

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

- Use Graph Clustering (METIS or Graclus) to find the partition.
- Example: on fMRI dataset ($p = 0.228$ million) with 20 blocks,

  random partition: need 1.6 million column computations.

  graph clustering: need 0.237 million column computations.

BigQUIC

- Block co-ordinate descent with clustering,
  - needs $O(p^2) \rightarrow O(m + p^2/k)$ storage
  - needs $O(mp) \rightarrow O(mp)$ 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 $\log\det(A)$.
- Our approach computes $\log\det(A)$ in $O(mp)$ time.
- Cholesky factorization in QUIC requires $O(p^3)$ computation.
- If $A = \begin{pmatrix} a & \mathbf{b}^T \\ \mathbf{b} & C \end{pmatrix}$,
  - $\det(A) = \det(C)(a - \mathbf{b}^T C^{-1}\mathbf{b})$
  - A is positive definite iff C is positive definite and $(a - \mathbf{b}^T C^{-1}\mathbf{b}) > 0$.
- C is sparse, so can compute $C^{-1}\mathbf{b}$ using Conjugate Gradient (CG).
- Time complexity: $T_{CG} = O(mT)$, where T is number of CG iterations.

BigQUIC

- Block co-ordinate descent with clustering,
  - needs $O(p^2) \rightarrow O(m + p^2/k)$ storage
  - needs $O(mp) \rightarrow O(mp)$ computation per sweep, where $m = \|X_t\|_0$
- Line search (compute determinant of a big sparse matrix).
  - needs $O(p^2) \rightarrow O(p)$ storage
  - needs $O(p^3) \rightarrow O(mp)$ computation

Algorithm

BigQUIC

Input: Samples Y, scalar λ, initial X_0.

For $t = 0, 1, \ldots$

1. Variable selection: select a *free* set of $m \ll p^2$ variables.
2. **Construct a partition by clustering.**
3. Run **block coordinate descent** to find descent direction: $D_t = \arg\min_\Delta \bar{f}_{X_t}(X_t + \Delta)$ over set of free variables.
4. 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 $f(X_t + \alpha D_t) \leq f(X_t) + \alpha\sigma\Delta_t$.

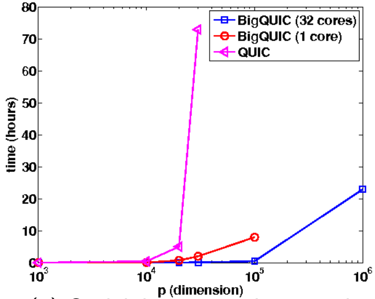
   (Schur complement with conjugate gradient method.)

BigQUIC Convergence Analysis

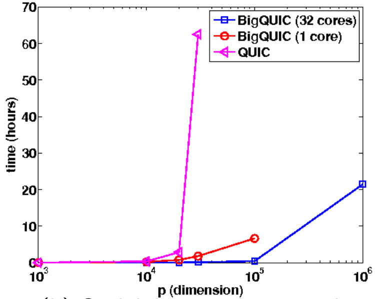
- Recall $W = X^{-1}$.
- When each $\mathbf{w}_i$ is computed by CG ($X\mathbf{w}_i = \mathbf{e}_i$):
  - The gradient $\nabla_{ij} g(X) = S_{ij} - W_{ij}$ on free set can be computed once and stored in memory.
  - Hessian ($\mathbf{w}_i^T D \mathbf{w}_j$ in coordinate updates) needs to be repeatedly computed.
- To reduce the time overhead, Hessian should be computed approximately.
- **Theorem:** the convergence rate is quadratic if $\|X\hat{\mathbf{w}}_{\mathbf{i}} - \mathbf{e}_{\mathbf{i}}\| = O(\|\nabla^S f(X_t)\|)$, where

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

Experimental results (scalability)



(a) Scalability on random graph



(b) Scalability on chain graph

Figure: BigQUIC can solve one million dimensional problems.

Experimental results

BigQUIC is faster even for medium size problems.

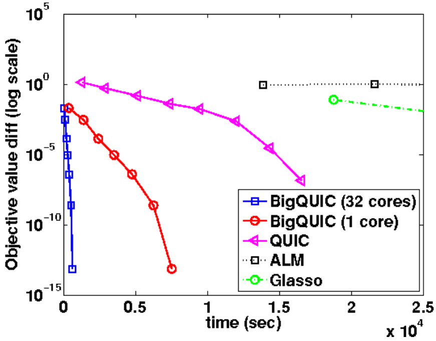
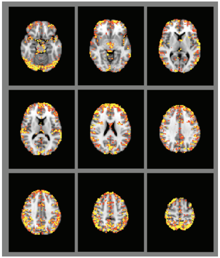


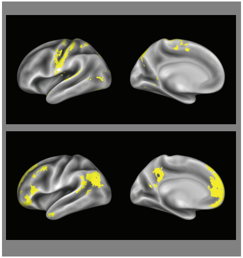
Figure: Comparison on FMRI data with a $p = 20000$ subset (maximum dimension that previous methods can handle).

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

- BigQUIC: Memory efficient quadratic approximation method for sparse inverse covariance estimation.
- Our contributions:
  - Computing Newton direction:
    - Coordinate descent $\rightarrow$ **block coordinate descent with clustering.**
    - Memory complexity: $O(p^2) \rightarrow O(m + p^2/k)$.
    - Time complexity: $O(mp) \rightarrow O(mp)$.
  - Line search (computing determinant of a big sparse matrix)
    - Cholesky factorization $\rightarrow$ **Schur complement with conjugate gradient method.**
    - Memory complexity: $O(p^2) \rightarrow O(p)$.
    - Time complexity: $O(p^3) \rightarrow O(mp)$.
  - **Inexact Hessian computation with super-linear convergence.**

References

[1] 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.
[2] C. J. Hsieh, M. Sustik, I. S. Dhillon, and P. Ravikumar. *Sparse Inverse Covariance Matrix Estimation using Quadratic Approximation*. NIPS, 2011.
[3] C. J. Hsieh, I. S. Dhillon, P. Ravikumar, A. Banerjee. *A Divide-and-Conquer Procedure for Sparse Inverse Covariance Estimation*. NIPS, 2012.
[4] P. A. Olsen, F. Oztoprak, J. Nocedal, and S. J. Rennie. *Newton-Like Methods for Sparse Inverse Covariance Estimation*. Optimization Online, 2012.
[5] O. Banerjee, L. El Ghaoui, and A. d'Aspremont *Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data*. JMLR, 2008.
[6] J. Friedman, T. Hastie, and R. Tibshirani. *Sparse inverse covariance estimation with the graphical lasso*. Biostatistics, 2008.
[7] L. Li and K.-C. Toh. *An inexact interior point method for l1-reguarlized sparse covariance selection*. Mathematical Programming Computation, 2010.
[8] K. Scheinberg, S. Ma, and D. Glodfarb. *Sparse inverse covariance selection via alternating linearization methods*. NIPS, 2010.
[9] K. Scheinberg and I. Rish. *Learning sparse Gaussian Markov networks using a greedy coordinate ascent approach*. Machine Learning and Knowledge Discovery in Databases, 2010.