# Learning the Structure of Mixed Graphical Models

**Jason Lee** with Trevor Hastie, Michael Saunders, Yuekai Sun, and Jonathan Taylor

Institute of Computational & Mathematical Engineering
Stanford University

June 26th, 2014

# Examples of Graphical Models

- Pairwise MRF.

$$p(y) = \frac{1}{Z(\Theta)} \exp\left(\sum_{(r,j)\in E(G)} \phi_{rj}(y_r, y_j)\right)$$

- Multivariate gaussian distribution (Gaussian MRF)

$$p(x) = \frac{1}{Z(\Theta)} \exp\left(-\frac{1}{2}\sum_{s=1}^{p}\sum_{t=1}^{p} \beta_{st} x_s x_t + \sum_{s=1}^{p} \alpha_s x_s\right)$$

# Mixed Graphical Model

- Want a simple joint distribution on $p$ continuous variables and $q$ discrete (categorical) variables.
- Joint distribution of $p$ gaussian variables is multivariate gaussian.
- Joint distribution of $q$ discrete variables is pairwise mrf.
- Conditional distributions can be estimated via (generalized) linear regression.
- What about the potential term between a continuous variable $x_s$ and discrete variable $y_j$?

# Mixed Model - Joint Distribution

$$p(x, y; \Theta) = \frac{1}{Z(\Theta)} \exp\left( \sum_{s=1}^{p} \sum_{t=1}^{p} -\frac{1}{2} \beta_{st} x_s x_t + \sum_{s=1}^{p} \alpha_s x_s + \sum_{s=1}^{p} \sum_{j=1}^{q} \rho_{sj}(y_j) x_s + \sum_{j=1}^{q} \sum_{r=1}^{q} \phi_{rj}(y_r, y_j) \right)$$

# Properties of the Mixed Model

- Pairwise model with 3 type of potentials: discrete-discrete, continuous-discrete, and continuous-continuous. Thus has $O((p+q)^2)$ parameters.
- $p(x|y)$ is a gaussian with $\Sigma = B^{-1}$ and $\mu = B^{-1}\left(\sum_j \rho_{sj}(y_j)\right)$.
- Conditional distribution of $x$ have the same covariance regardless of the values taken by the discrete variables $y$. Mean depends additively on the values of discrete variables $y$.
- Special case of Lauritzen's mixed graphical model.

# Related Work

- Lauritzen proposed the conditional Gaussian model
- Fellinghauer et al. (2011) use random forests to fit the conditional distributions. This is tailored for mixed models.
- Cheng, Levina, and Zhu (2013) generalize to include higher order edges.
- Yang et al. (2014) and Shizhe Chen, Witten, and Shojaie (2014) generalize beyond Gaussian and categorical.

# Outline

**Parameter Learning**

Structure Learning

Experimental Results

# Pseudolikelihood

- Log-likelihood: $\ell(\Theta) = \log p(x^i; \Theta)$. Derivative is $\hat{T}(x, y) - E_{p(\Theta)}[T(x, y)]$ where $T$ are sufficient statistics. This is hard to compute.
- Log-pseudolikelihood: $\ell_{PL}(\Theta) = \sum_s \log p(x_s^i | x_{\setminus s}^i; \Theta)$
- Pseudolikelihood is an asymptotically consistent approximation to the likelihood by using product of the conditional distributions.
- Partition function cancels out in the conditional distribution, so gradients of the log-pseudolikelihood are cheap to compute.

# Conditional Distribution of a Discrete Variable

For a discrete variable $y_r$ with $L_r$ states, its conditional distribution is a multinomial distribution, as used in (multiclass) logistic regression. Whenever a discrete variable is a predictor, each level contributes an additive effect; continuous variables contribute linear effects.

$$p(y_r|y_{\setminus r},,x;\Theta) = \frac{\exp\left(\sum_s \rho_{sr}(y_r)x_s + \phi_{rr}(y_r,y_r) + \sum_{j\neq r}\phi_{rj}(y_r,y_j)\right)}{\sum_{l=1}^{L_r}\exp\left(\sum_s \rho_{sr}(l)x_s + \phi_{rr}(l,l) + \sum_{j\neq r}\phi_{rj}(l,y_j)\right)}$$

This is just **multinomial logistic regression**.

$$p(y_r = k) = \frac{\exp\left(\alpha_k^T z\right)}{\sum_{l=1}^{L_r}\exp\left(\alpha_l^T z\right)}$$

Continuous variable $x_s$ given all other variables is a gaussian distribution with a linear regression model for the mean.

$$p(x_s|x_{\setminus s}, y; \Theta) = \frac{\sqrt{\beta_{ss}}}{\sqrt{2\pi}} \exp\left(\frac{-\beta_{ss}}{2}\left(\frac{\alpha_s + \sum_j \rho_{sj}(y_j) - \sum_{t\neq s} \beta_{st} x_t}{\beta_{ss}} - x_s\right)^2\right)$$

This can be expressed as **linear regression**

$$E(x_s|z_1, \ldots, z_p) = \alpha^T z = \alpha_0 + \sum_j z_j \alpha_j \tag{1}$$

$$p(x_s|z_1, \ldots, z_p) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x_s - \alpha^T z)^2\right) \text{ with } \sigma = 1/\beta_{ss} \tag{2}$$

# Two more parameter estimation methods

Neighborhood selection/Separate regressions.

- Each node maximizes its own conditional likelihood $p(x_s|x_{\setminus s})$. Intuitively, this should behave similar to the pseudolikelihood since the pseudolikelihood jointly minimizes $\sum_s -\log p(x_s|x_{\setminus s})$.
- This has twice the number of parameters as the pseudolikelihood/likelihood because the regressions do not enforce symmetry.
- Easily distributed.

Maximum Likelihood

- Believed to be more statistically efficient
- Computationally intractable.

# Outline

Parameter Learning

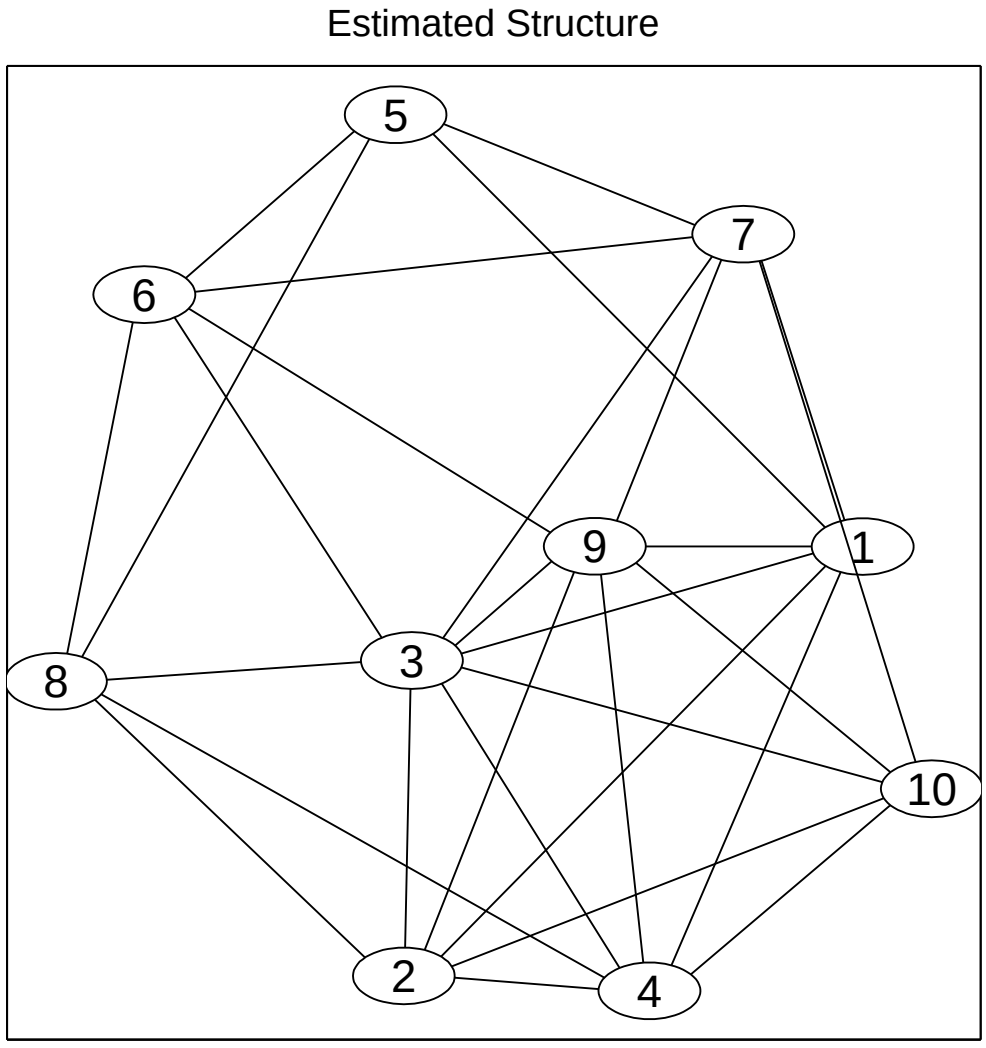
**Structure Learning**

Experimental Results

# Sparsity and Conditional Independence

- Lack of an edge $(u, v)$ means $X_u \perp X_v | X_{\setminus u,v}$ ($X_u$ and $X_v$ are conditionally independent.)
- Means that parameter block $\beta_{st}$,$\rho_{sj}$, or $\phi_{rj}$ are 0.
- Each parameter block is a different size. The continuous-continuous edge are scalars, the continuous-discrete edge are vectors and the discrete-discrete edge is a table.

# Structure Learning



# Parameters of the mixed model

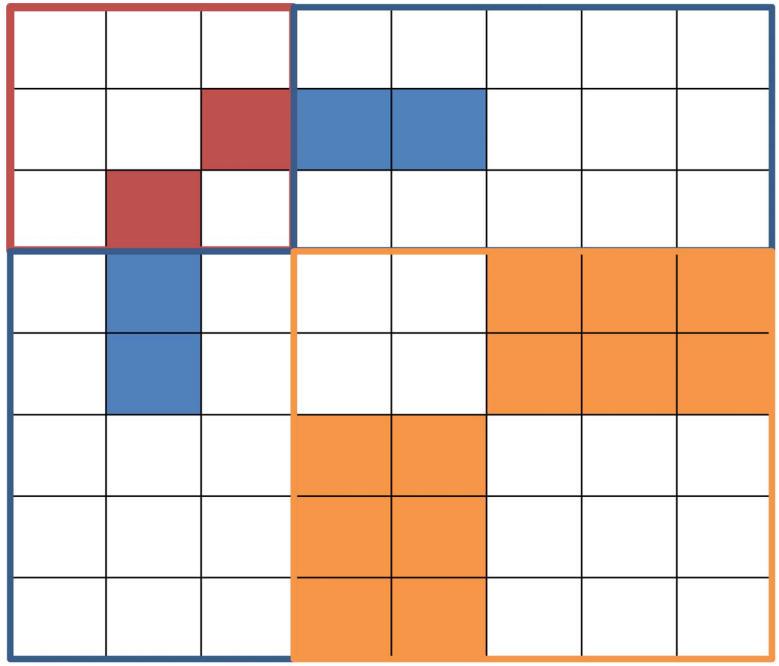


**Figure:** $\beta_{st}$ shown in red, $\rho_{sj}$ shown in blue, and $\phi_{rj}$ shown in orange. The rectangles correspond to a group of parameters.

# Regularizer

$$\min_\Theta \ell_{PL}(\Theta) + \lambda \left( \sum_{s,t} w_{st} \|\beta_{st}\| + \sum_{s,j} w_{sj} \|\rho_{sj}\| + \sum_{r,j} w_{rj} \|\phi_{rj}\| \right)$$

- Each edge group is of a different size and different distribution, so we need a different penalty for each group.
- By KKT conditions, a group is non-zero iff $\left\| \frac{\partial \ell}{\partial \theta_g} \right\| > \lambda w_g$. Thus we choose weights

$$w_g \propto \mathbf{E}_0 \left\| \frac{\partial \ell}{\partial \theta_g} \right\|.$$

# Optimization Algorithm: Proximal Newton method

- $g(x) + h(x) :=$
  $\min_\Theta \ell_{PL}(\Theta) + \lambda \left( \sum_{s,t} \|\beta_{st}\| + \sum_{s,j} \|\rho_{sj}\| + \sum_{r,j} \|\phi_{rj}\| \right)$
- First-order methods: proximal gradient and accelerated proximal gradient, which have similar convergence properties as their smooth counter parts (sublinear convergence rate, and linear convergence rate under strong convexity).
- Second-order methods: model smooth part $g(x)$ with quadratic model. Proximal gradient is a linear model of the smooth function $g(x)$.

# Proximal Newton-like Algorithms

- Build a quadratic model about the iterate $x_k$ and solve this as a subproblem.

$$x_+ = \operatorname{argmin}_u \, g(x) + \nabla g(x)^T (u - x) + \frac{1}{2t}(u - x)^T H (u - x) + h(u)$$

---

**Algorithm 1** A generic proximal Newton-type method

---

**Require:** starting point $x_0 \in \operatorname{dom} f$

1: **repeat**
2: Choose an approximation to the Hessian $H_k$.
3: Solve the subproblem for a search direction:
$\Delta x_k \leftarrow \arg\min_d \nabla g(x_k)^T d + \frac{1}{2} d^T H_k d + h(x_k + d)$.
4: Select $t_k$ with a backtracking line search.
5: Update: $x_{k+1} \leftarrow x_k + t_k \Delta x_k$.
6: **until** stopping conditions are satisfied.

---

# Why are these proximal?

### Definition (Scaled proximal mappings)

Let $h$ be a convex function and $H$, a positive definite matrix. Then the scaled proximal mapping of $h$ at $x$ is defined to be

$$\text{prox}_h^H(x) = \arg\min_y \; h(y) + \frac{1}{2}\|y - x\|_H^2.$$

The proximal Newton update is

$$x_{k+1} = \text{prox}_h^{H_k}\left(x_k - H_k^{-1}\nabla g(x_k)\right)$$

and analogous to the proximal gradient update

$$x_{k+1} = \text{prox}_{h/L}\left(x_k - \frac{1}{L}\nabla g(x_k)\right)$$

# A classical idea

**Traces back to:**

- Projected Newton-type methods
- Cost-approximation methods

**Popular methods tailored to specific problems:**

- `glmnet`: lasso and elastic-net regularized generalized linear models
- LIBLINEAR: $\ell_1$-regularized logistic regression
- QUIC: sparse inverse covariance estimation

- Theoretical analysis shows that this converges quadratically with exact Hessian and super-linearly with BFGS (Lee, Sun, and Saunders 2012).
- Empirical results on structure learning problem confirms this. Requires very few derivatives of the log-partition.
- If we solve subproblems with first order methods, only require proximal operator of nonsmooth $h(u)$. Method is very general.
- Method allows you to choose how to solve the subproblem, and comes with a stopping criterion that preserves the convergence rate.
- PNOPT package:
  `www.stanford.edu/group/SOL/software/pnopt`

# Statistical Consistency

Special case of a more general model selection consistency theorem.

**Theorem (Lee, Sun, and Taylor 2013)**

1. $\left\|\hat{\Theta} - \Theta^\star\right\|_F \leq C\sqrt{\frac{|A| \log |G|}{n}}$
2. $\hat{\Theta}_g = 0$ *for* $g \in I$.

$|A|$ *is the number of active edges, and* $I$ *is the inactive edges. Main assumption is a generalized irrepresentable condition.*

# Outline

Parameter Learning

Structure Learning

**Experimental Results**

# Synthetic Experiment

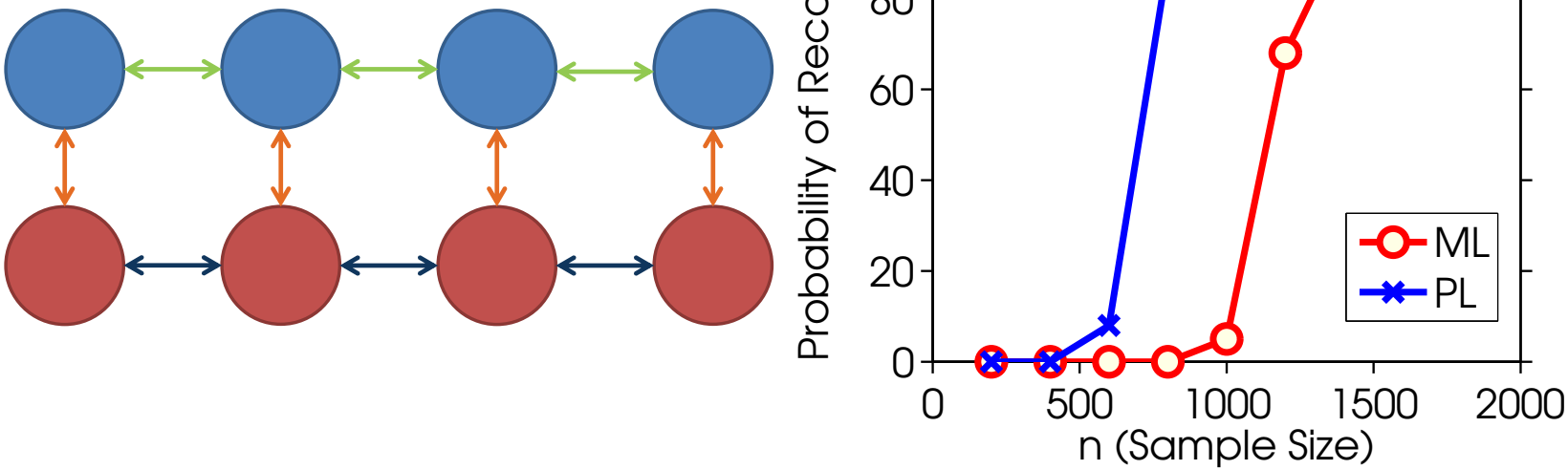



**Figure:** Blue nodes are continuous variables, red nodes are binary variables and the orange, green and dark blue lines represent the 3 types of edges. Plot of the probability of correct edge recovery at a given sample size ($p + q = 20$). Results are averaged over 100 trials.

# Survey Experiments

- The survey dataset we consider consists of 11 variables, of which 2 are continuous and 9 are discrete: age (continuous), log-wage (continuous), year(7 states), sex(2 states),marital status (5 states), race(4 states), education level (5 states), geographic region(9 states), job class (2 states), health (2 states), and health insurance (2 states).
- All the evaluations are done using a holdout test set of size $100,000$ for the survey experiments.
- The regularization parameter $\lambda$ is varied over the interval $[5 \times 10^{-5}, .7]$ at 50 points equispaced on log-scale for all experiments.

# Comparing Against Separate Regressions

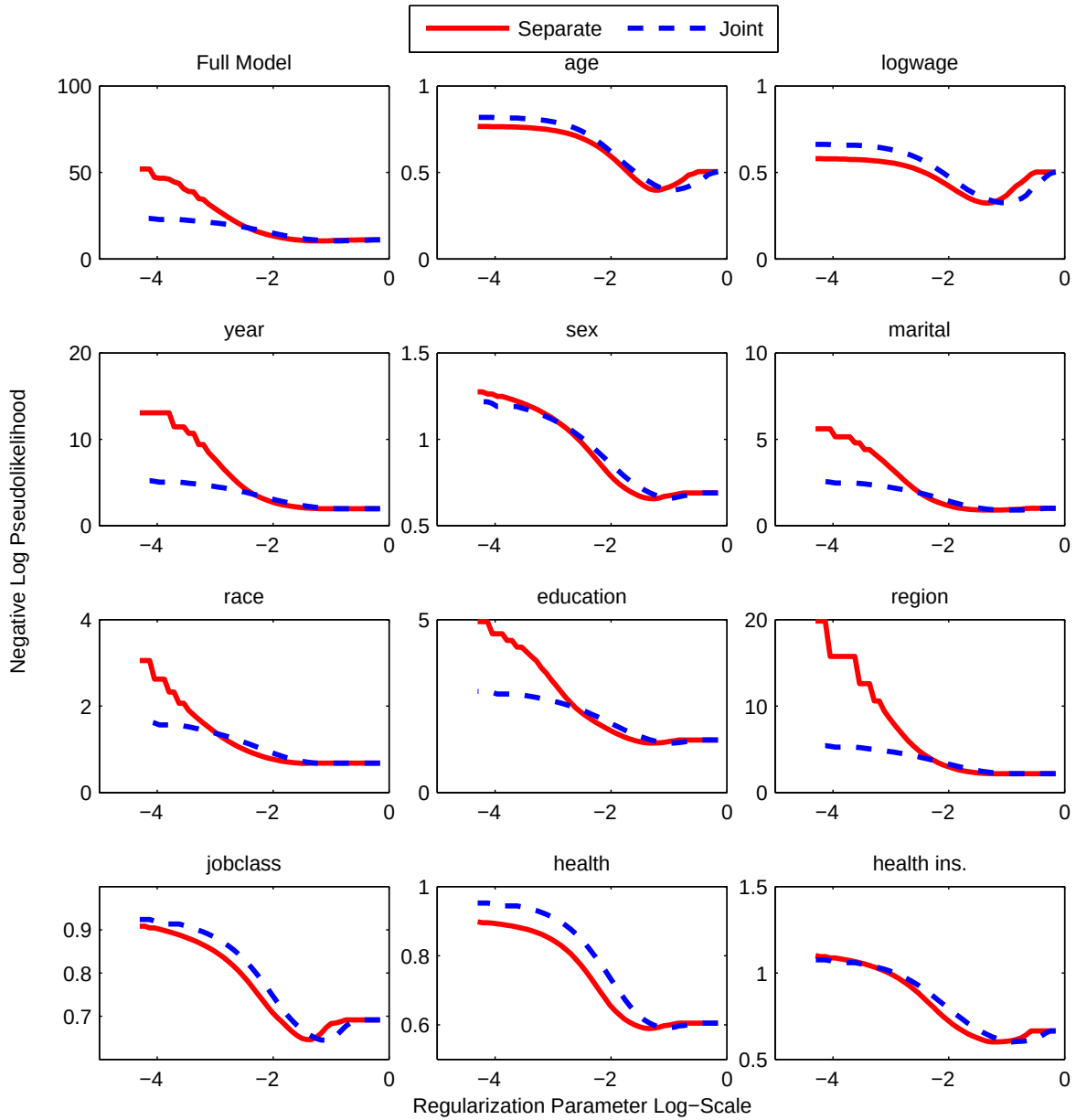



**Figure:** Separate Regression vs Pseudolikelihood $n = 100$.

# Comparing Against Separate Regressions

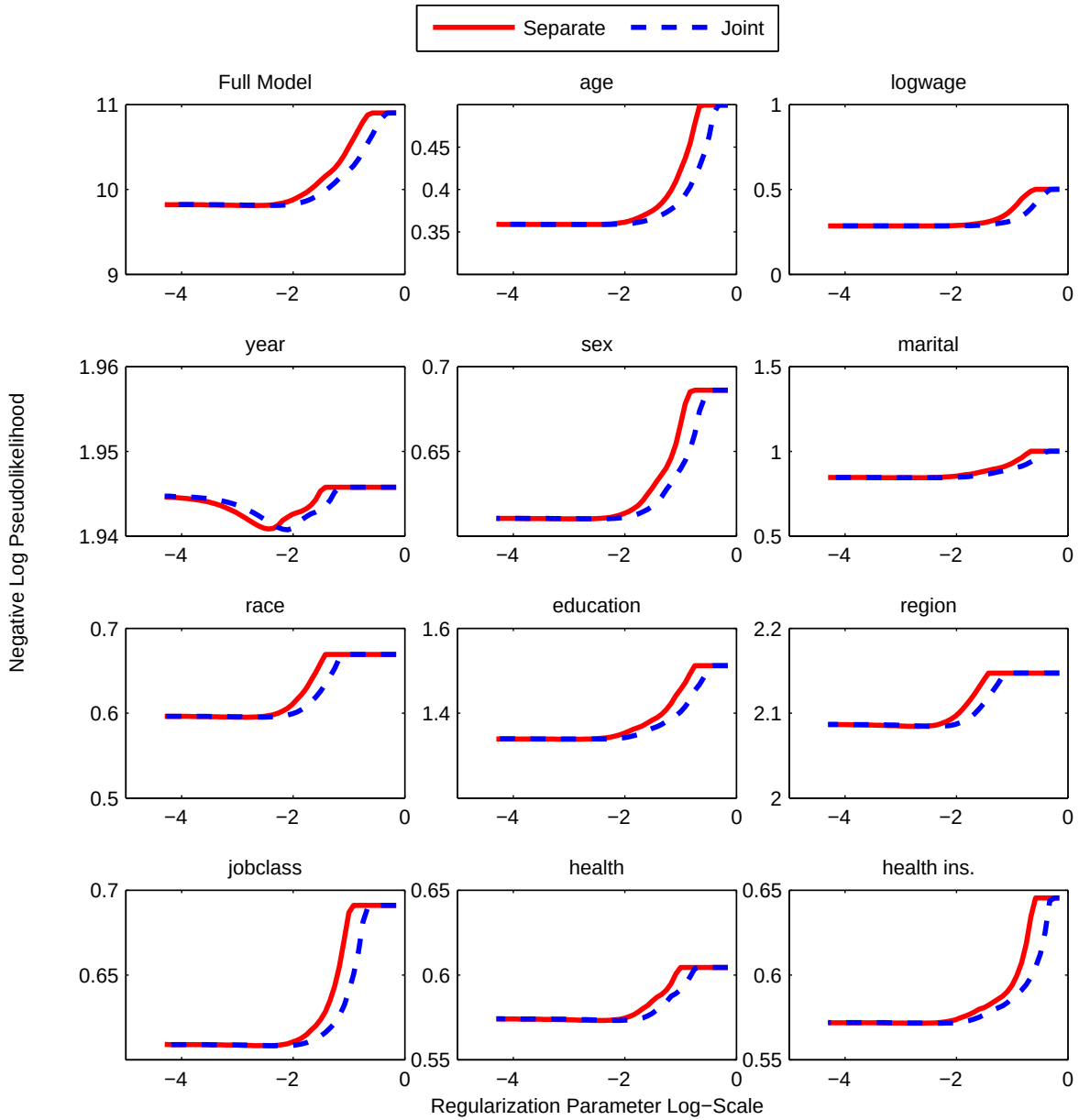



**Figure:** Separate Regression vs Pseudolikelihood $n = 10000$.

# What do we lose from using the Pseuodolikelihood?

- We originally motivated the pseudolikelihood as a computational surrogate to the likelihood.
- Pseudolikelihood is consistent.
- For small models, we can compute maximum likelihood estimates and compare it against the pseudolikelihood.

# MLE vs. MPLE on Conditional Model

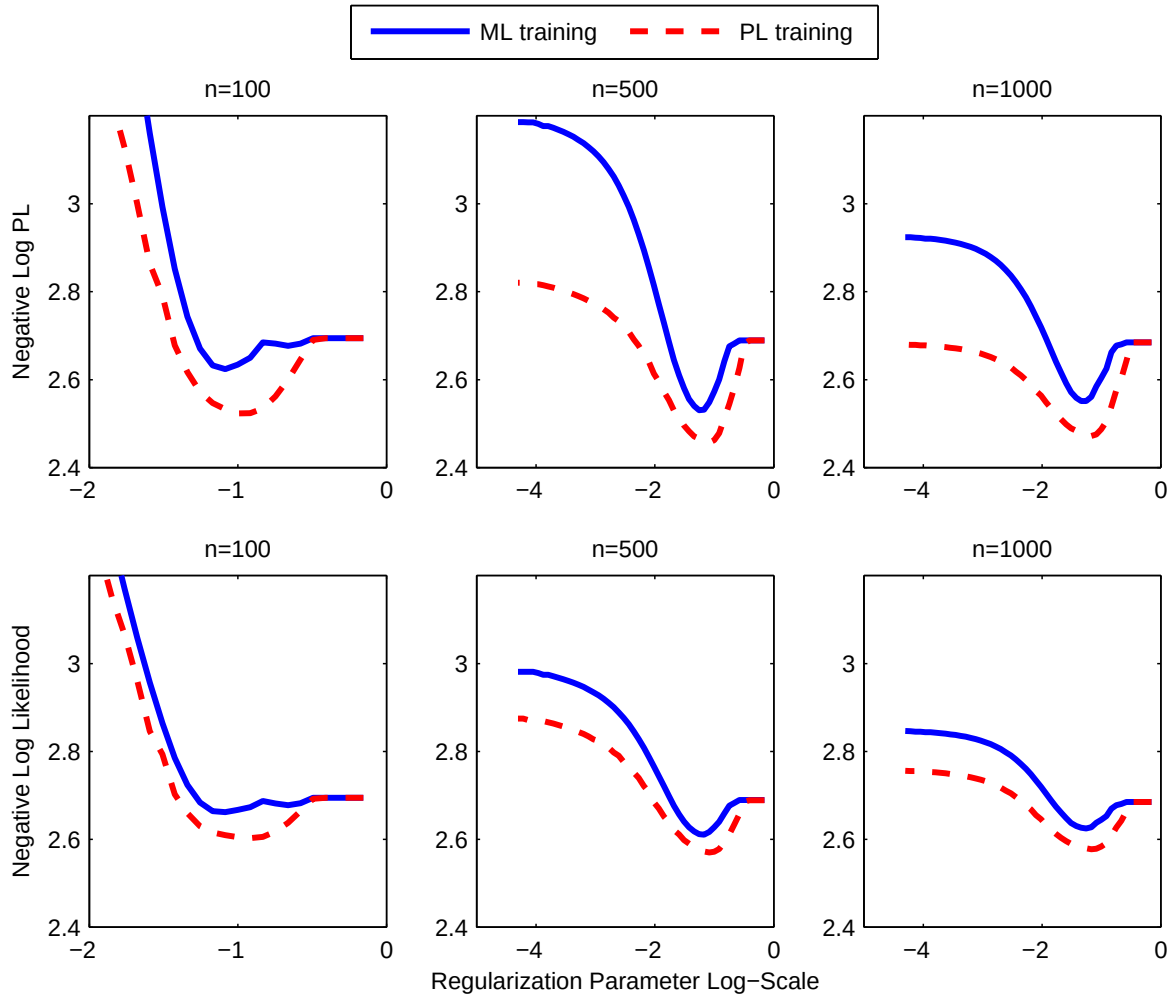



**Figure:** Maximum Likelihood vs Pseudolikelihood. $y$-axis for top row is the negative log pseudolikelihood. $y$-axis for bottom row is the negative log likelihood. Pseudolikelihood outperforms maximum likelihood across all the experiments.

# MLE vs. MPLE

- We expect PL to do better when evaluated on test negative log PL and ML to do better when evaluated on test negative log likelihood.
- Asymptotic theory also suggests that ML is better.
- Theory does not apply to misspecified models and finite sample regime.

# Conclusion

- Defined a new pairwise graphical model over gaussian and discrete variables.
- Used the pseudolikelihood for tractable inference
- Group sparsity to enforce an edge-sparse graphical model.
- Fast learning method using proximal Newton. Theoretical analysis of proximal Newton algorithm.
- Theoretical analysis in high-dimensional regime for general exponential families.

Thanks for listening!

# Solving the subproblem

$$\begin{aligned}\Delta x_k &= \arg\min_d \ \nabla g(x_k)^T d + \frac{1}{2} d^T H_k d + h(x_k + d) \\ &= \arg\min_d \ \hat{g}_k(x_k + d) + h(x_k + d)\end{aligned}$$

Usually, we must use an iterative method to solve this subproblem.

- Use proximal gradient or coordinate descent on the subproblem.
- A gradient/coordinate descent iteration on the subproblem is much cheaper than a gradient iteration on the original function $f$, since it does not require a pass over the data. By solving the subproblem, we are more efficiently using a gradient evaluation than gradient descent.
- $H_k$ is commonly a L-BFGS approximation, so computing a gradient takes $O(Lp)$. A gradient of the original function takes $O(np)$. The subproblem is independent of $n$.

# Inexact Newton-type methods

**Main idea:** no need to solve the subproblem exactly only need a good enough search direction.

- We solve the subproblem approximately with an iterative method, terminating (sometimes very) early
- number of iterations may increase, but computational expense per iteration is smaller
- many practical implementations use inexact search directions

# What makes a stopping condition good?

We should solve the subproblem more precisely when:

1. $x_k$ is close to $x^\star$, since Newton's method converges quadratically in this regime.
2. $\hat{g}_k + h$ is a good approximation to $f$ in the vicinity of $x_k$ (meaning $H_k$ has captured the curvature in $g$), since minimizing the subproblem also minimizes $f$.

# Early stopping conditions

For regular Newton's method the most common stopping condition is

$$\|\nabla \hat{g}_k(x_k + \Delta x_k)\| \leq \eta_k \|\nabla g(x_k)\| .$$

Analogously,

$$\underbrace{\left\|G_{(\hat{g}_k+h)/M}(x_k + \Delta x_k)\right\|}_{\text{optimality of subproblem solution}} \leq \eta_k \underbrace{\left\|G_{f/M}(x_k)\right\|}_{\text{optimality of } x_k}$$

Choose $\eta_k$ based on how well $G_{\hat{g}_k+h}$ approximates $G_f$:

$$\eta_k \sim \frac{\left\|G_{(\hat{g}_{k-1}+h)/M}(x_k) - G_{f/M}(x_k)\right\|}{\left\|G_{f/M}(x_{k-1})\right\|}$$

**Reflects the Intuition:** solve the subproblem more precisely when

- $G_{f/M}$ is small, so $x_k$ is close to optimum.
- $G_{\hat{g}+h} - G_f \approx 0$, means that $H_k$ is accurately capturing the curvature of $g$.

# Convergence of the inexact prox-Newton method

- Inexact proximal Newton method converges superlinearly for the previous choice of stopping criterion and $\eta_k$.
- In practice, the stopping criterion works extremely well. It uses approximately the same number of iterations as solving the subproblem exactly, but spends much less time on each subproblem.

# Sparse inverse covariance (Graphical Lasso)

Sparse inverse covariance:

$$\min_{\Theta} \; -\text{logdet}(\Theta) + \mathbf{tr}(S\Theta) + \lambda\|\Theta\|_1$$

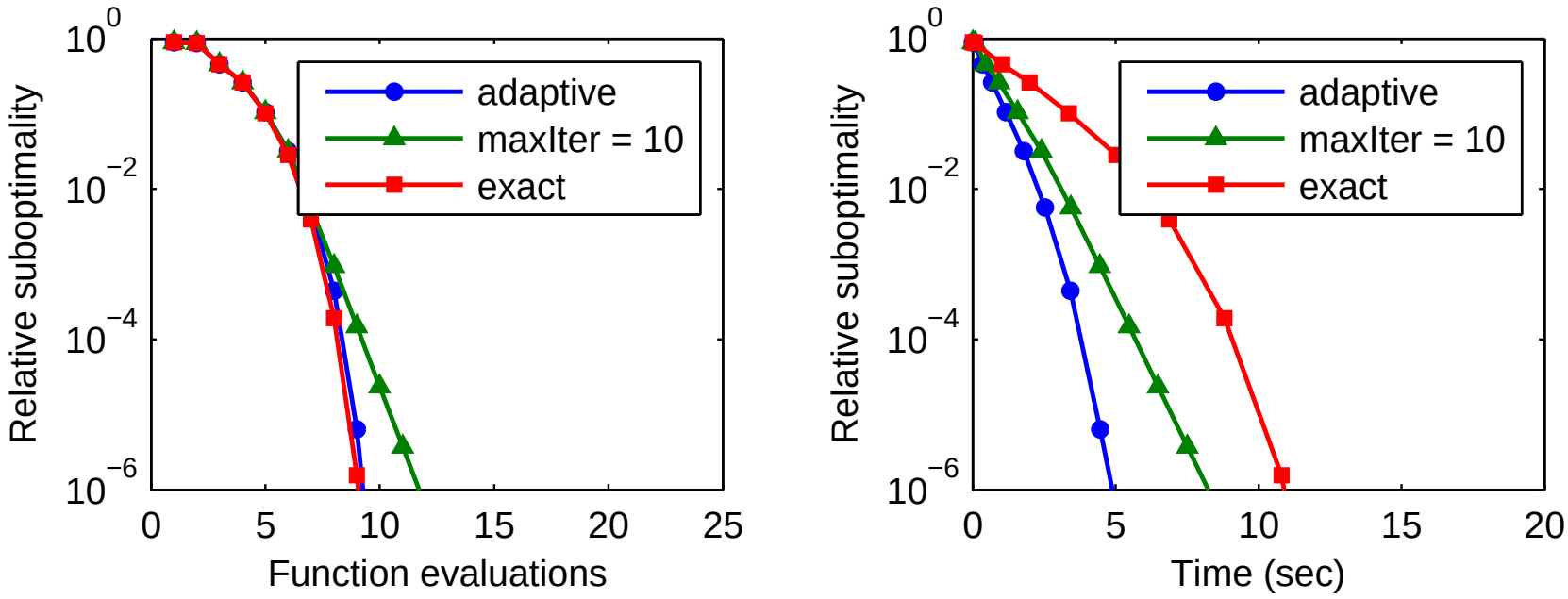
- $S$ is a sample covariance, and estimates $\Sigma$ the population covariance.

$$S = \sum_{i=1}^{p}(x_i - \mu)(x_i - \mu)^T$$

- $S$ is not of full rank since $n < p$, so $S^{-1}$ doesn't exist.
- Graphical lasso is a good estimator of $\Sigma^{-1}$

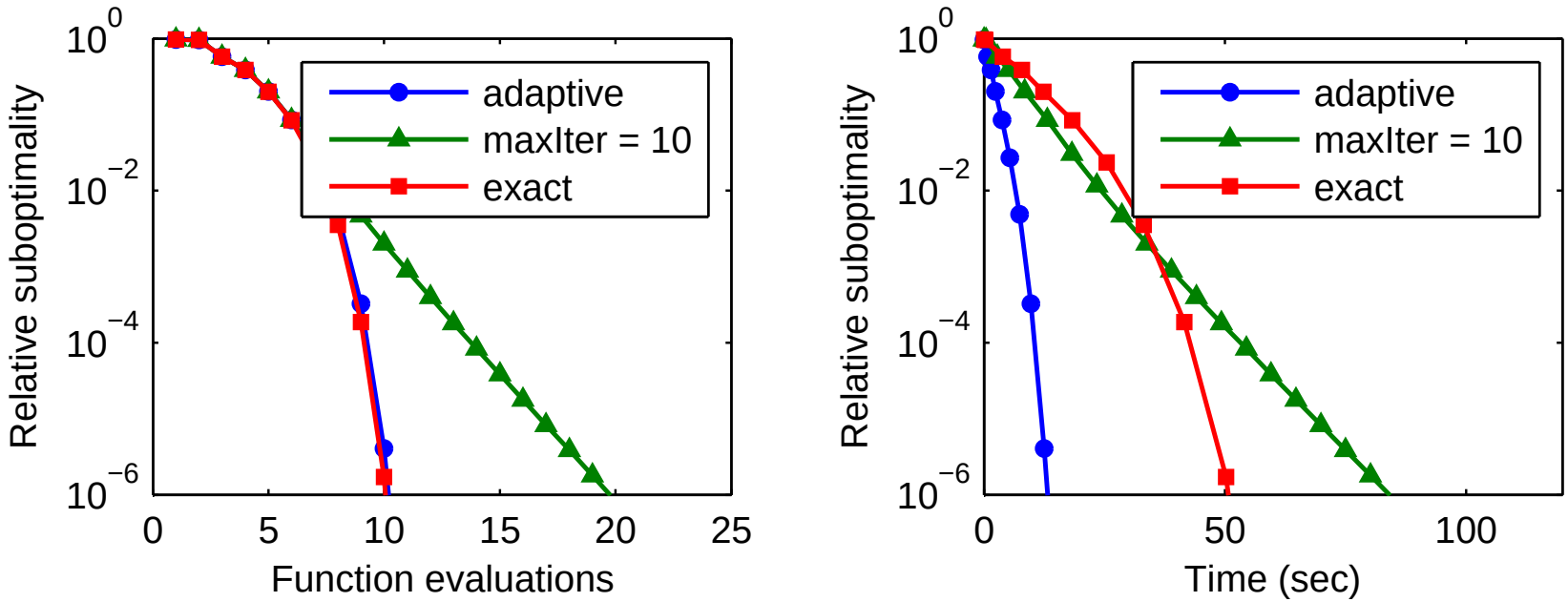
# Sparse inverse covariance estimation

**Figure:** Proximal BFGS method with three subproblem stopping conditions (Estrogen dataset $p = 682$)



# Sparse inverse covariance estimation

**Figure:** Leukemia dataset $p = 1255$



# Another example

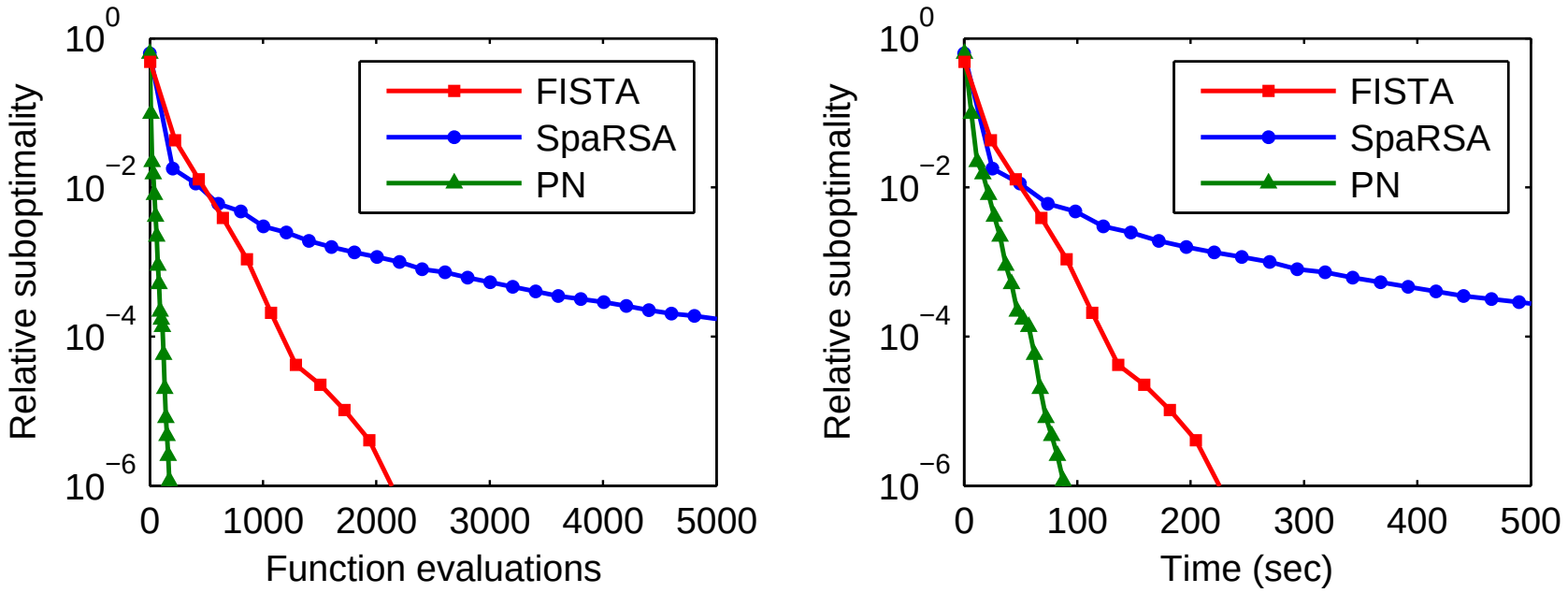
**Sparse logistic regression**

- training data: $x^{(1)}, \ldots, x^{(n)}$ with labels $y^{(1)}, \ldots, y^{(n)} \in \{0, 1\}$
- We fit a sparse logistic model to this data:

$$\underset{w}{\text{minimize}} \ \frac{1}{n} \sum_{i=1}^{n} -\log(1 + \exp(-y_i w^T x_i)) + \lambda \|w\|_1$$

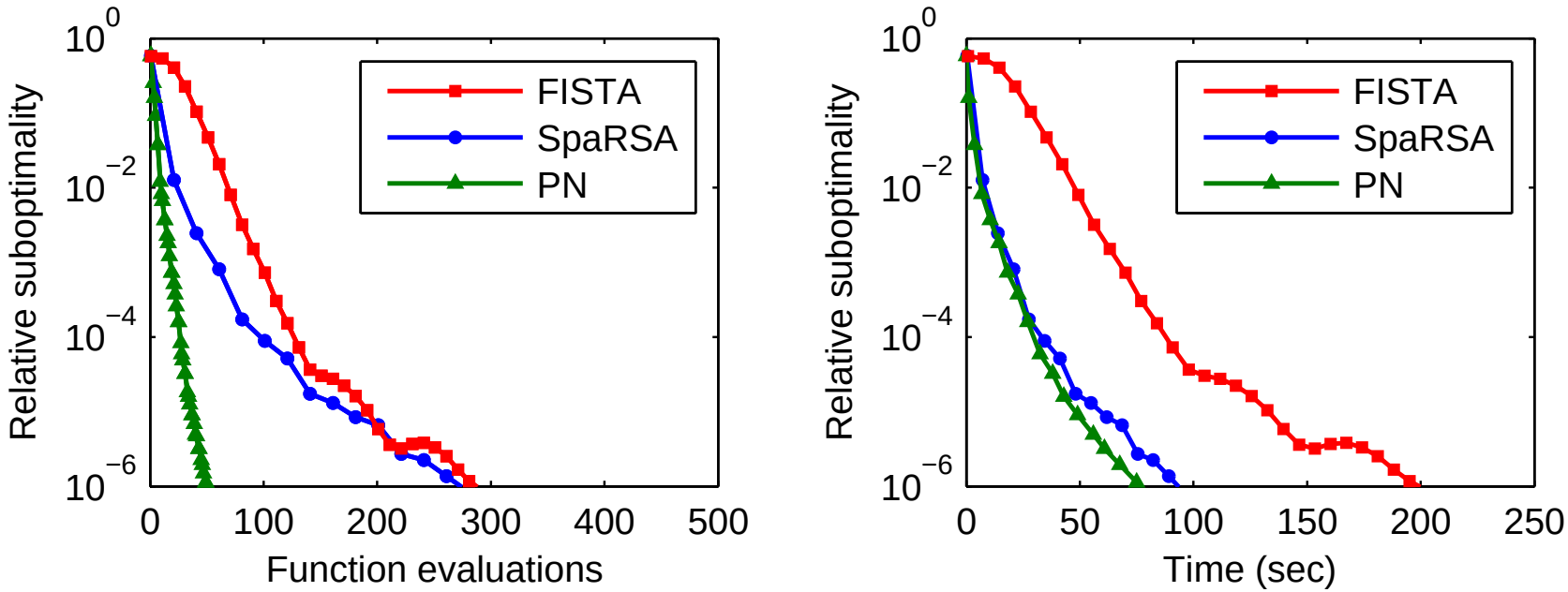
# Sparse logistic regression

**Figure:** Proximal L-BFGS method vs. FISTA and SpaRSA (`gisette` dataset, $n = 5000$, $p = 6000$ and dense)



# Sparse logistic regression

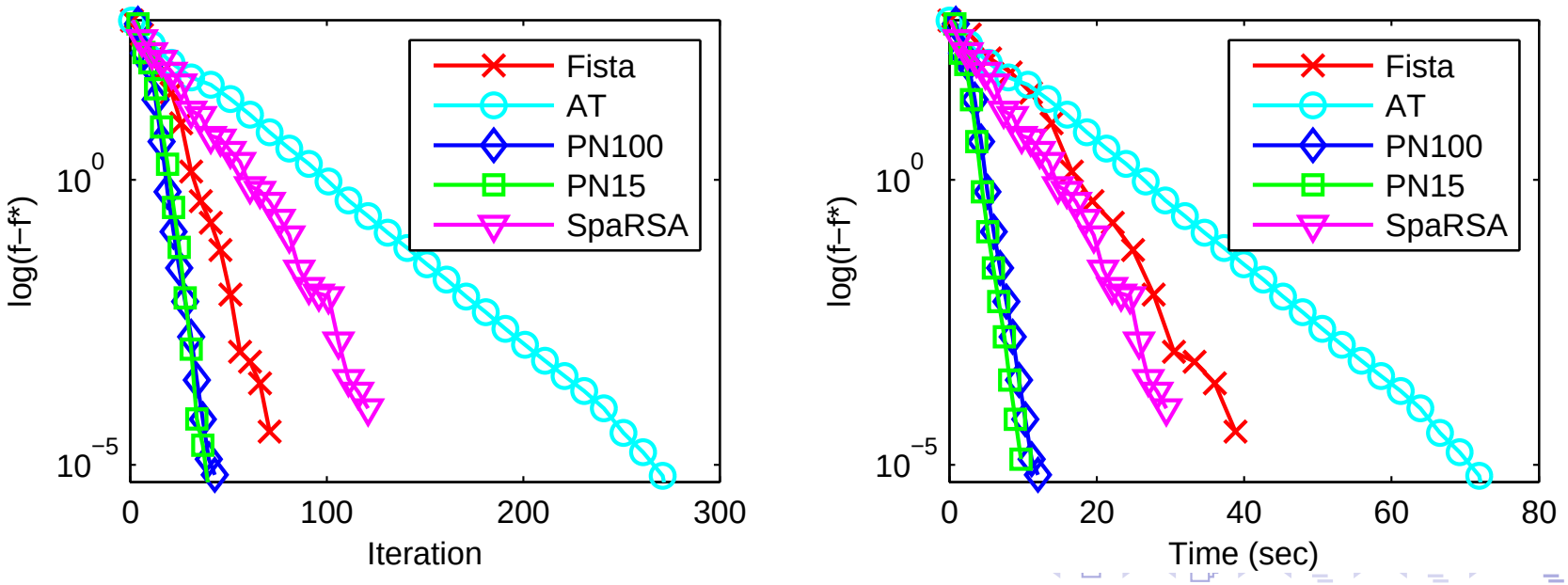
**Figure:** `rcv1` dataset, $n = 47,000$, $p = 542,000$ and 40 million nonzeros



# Markov random field structure learning

$$\begin{aligned}\underset{\theta}{\text{minimize}} \; & - \sum_{(r,j)\in E} \theta_{rj}(x_r, x_j) + \log Z(\theta) \\ & + \sum_{(r,j)\in E} \left(\lambda_1 \|\theta_{rj}\|_2 + \lambda_F \|\theta_{rj}\|_F^2\right).\end{aligned}$$

**Figure:** Markov random field structure learning



# Summary of Proximal Newton

Proximal Newton-type methods

- converge rapidly near the optimal solution, and can produce a solution of high accuracy
- are insensitive to the choice of coordinate system and to the condition number of the level sets of the objective
- are suited to problems where $g$, $\nabla g$ is expensive to evaluate compared to $h$, $\text{prox}_h$. This is the case when $g(x)$ is a loss function and computing the gradient requires a pass over the data.
- "more efficiently uses" a gradient evaluation of $g(x)$.