Network-based Penalized Regression

Wei Pan[1]
(joint work with Chong Luo[1], Xiaotong Shen[2])
[1]Division of Biostatistics, School of Public Health
[2]School of Statistics
University of Minnesota

Beijing, China
June 18, 2010

Outline

- Problem
- Review: Existing penalized methods
- New methods
- Discussion

Introduction

- Problem: linear model

$$Y = \sum_{i=1}^{p} X_i \beta_i + \epsilon, \;\; E(\epsilon) = 0, \tag{1}$$

  Feature: large p, small n.

- Q: variable selection; prediction
- Example 1: Li and Li (2008); Pan, Xie & Shen (2010)
  Y: clinical outcome, e.g. survival time;
  X_i: expression level of gene i.
- Example 2: eQTL analysis, Pan (2009)
- Typical approaches: ignore any relationships among X_i's.
- In our applications: genes are related ...
  e.g. as described by a network:

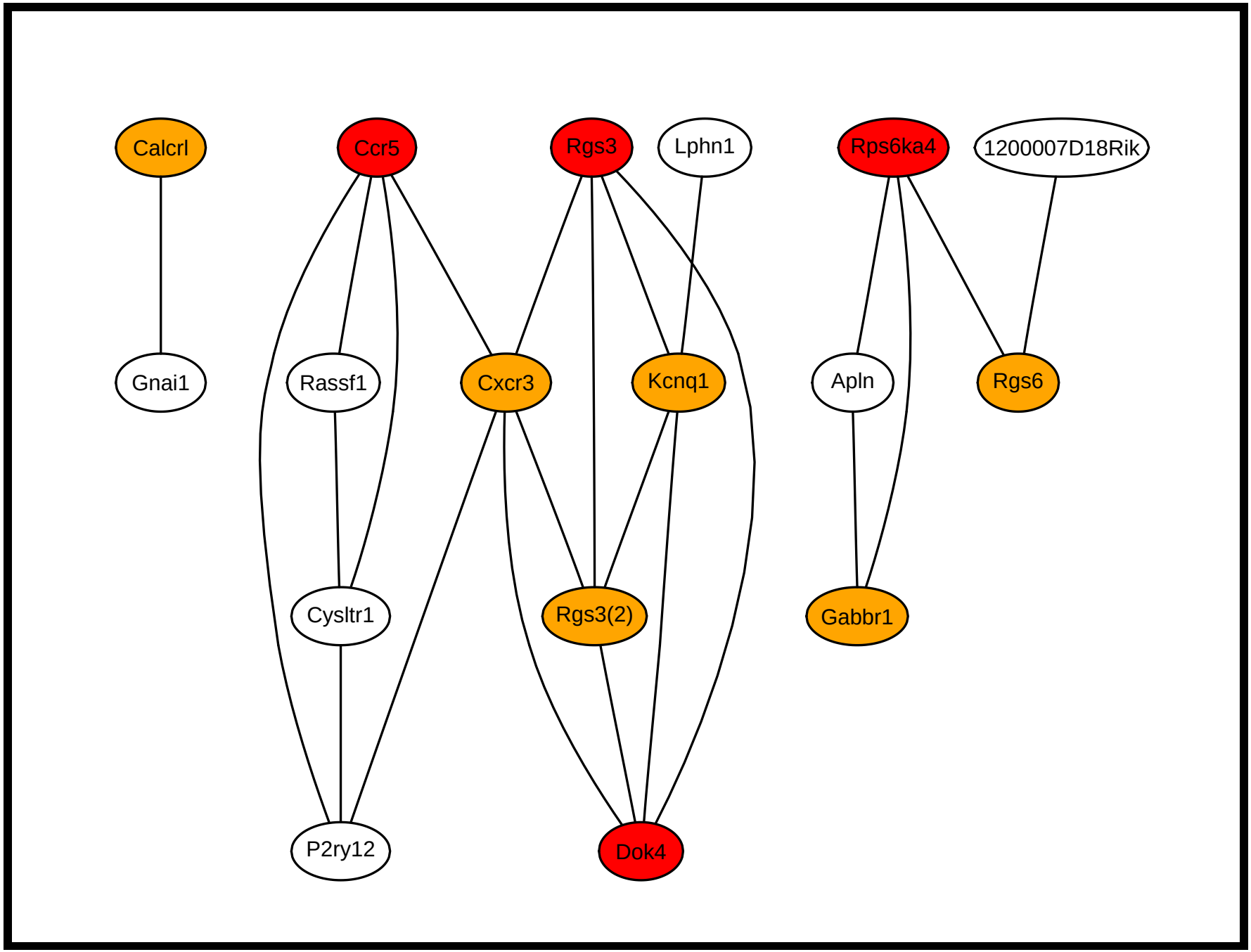
Calcrl
Ccr5
Rgs3
Lphn1
Rps6ka4
1200007D18Rik
Gnai1
Rassf1
Cxcr3
Kcnq1
Apln
Rgs6
Cysltr1
Rgs3(2)
Gabbr1
P2ry12
Dok4

Figure 1:

- **Network assumption/prior**: if two genes $i \sim j$ in a network, then $|\beta_i| \approx |\beta_j|$, or $|\beta_i|/w_i \approx |\beta_j|/w_j$.
- Goal: utilize the above assumption/prior.
- How?

Review: Existing Methods

- Penalized methods: for "large p, small n"

$$\hat{\beta} = \arg\min_{\beta} L(\beta) + p_\lambda(\beta),$$

- Lasso (Tibshirani 1996):

$$p_\lambda(\beta) = \lambda \sum_{k=1}^{p} |\beta_k|.$$

  Feature: variable selection; some $\hat{\beta}_k = 0$.

- Elastic net (Zou and Hastie 2005)

$$p_\lambda(\beta) = \lambda \sum_{k=1}^{p} |\beta_k| + \lambda_2 \sum_{k=1}^{p} \beta_k^2.$$

  But ...

- A network-based penalty of Li and Li (2008):

$$p_\lambda(\beta) = \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sum_{i \sim j} \left(\frac{\beta_i}{\sqrt{d_i}} - \frac{\beta_j}{\sqrt{d_j}} \right)^2, \qquad (2)$$

  d_i: degrees of node i;
  Feature: two λ's and two terms for diff purposes ...
  Problem: if β_i and β_j have diff signs ...

- A modification by Li and Li (2010):

$$p_\lambda(\beta) = \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sum_{i \sim j} \left(\frac{sgn(\tilde{\beta}_i)\beta_i}{\sqrt{d_i}} - \frac{sgn(\tilde{\beta}_j)\beta_j}{\sqrt{d_j}} \right)^2, \quad (3)$$

  $\tilde{\beta}_j$: an initial estimate based on Enet; a 2-step procedure.

- A class of network-based penalties of Pan, Xie and Shen (2010):

$$p_\lambda(\beta;\gamma,w)=\lambda 2^{1/\gamma'}\sum_{i\sim j}\left(\frac{|\beta_i|^\gamma}{w_i}+\frac{|\beta_j|^\gamma}{w_j}\right)^{1/\gamma} \quad (4)$$

- w_i: smooth what?
  1) $w_i=d_i^{(\gamma+1)/2}$: smooth $|\beta_i|/\sqrt{d_i}$, as in Li and Li;
  2) $w_i=d_i$: smooth $|\beta_i|$
  Some theory under simplified cases.

- Feature: each term is an L_γ norm, $\gamma\geq 1$
  $\Longrightarrow$ **grouped** variable selection!
  $\Longrightarrow$ tend to realize $\hat\beta_i=\hat\beta_j=0$ if $i\sim j$!
  see Yuan and Lin 2006; Zhao et al 2007

- γ: a larger γ smoothes more;

- $\gamma = \infty$:

$$p_\lambda = \lambda \sum_{i \sim j} \max\left(\frac{|\beta_i|}{\sqrt{d_i}}, \frac{|\beta_j|}{\sqrt{d_j}} \right)$$

  maximally forces $|\hat{\beta}_i|/\sqrt{d_i} = |\hat{\beta}_j|/\sqrt{d_i}$ if $i \sim j$!

- Computational algorithm of Pan et al (2010):
  Generalized boosted lasso (GBL) (Zhao and Yu 2004);
  provided *approximate* solution paths.

- Use CV to choose tuning parameters, e.g. λ.

- Some simulation results:
  PMSE: prediction mean squared error for Y;
  q_1: # false zeros ($\beta_i \neq 0$ but $\hat{\beta}_i = 0$);
  q_0: # true zeros ($\beta_i = 0$ and $\hat{\beta}_i = 0$);
  $n = 50$, $p = p_1 + p_0 = 44 + 66$

Set-up	Methods	PMSE	q_1	q_0
1	Lasso	166.6 (32.9)	20.1 (2.5)	53.9 (6.4)
	Enet	164.3 (29.3)	10.6 (9.2)	31.4 (24.0)
	Li&Li	154.6 (28.3)	5.0 (7.6)	15.1 (21.2)
	$\gamma = 2$	138.1 (32.3)	3.2 (3.7)	60.0 (5.4)
	$\gamma = 8$	**132.0** (35.8)	3.2 (4.3)	60.0 (4.8)
	$\gamma = \infty$	162.9 (46.6)	7.3 (5.9)	56.6 (6.8)
2	Lasso	**160.8** (39.0)	30.2 (4.0)	61.1 (4.2)
	Enet	161.1 (45.5)	29.0 (8.5)	57.8 (15.1)
	Li&Li	161.7 (44.7)	26.0 (11.7)	52.1 (22.3)
	$\gamma = 2$	161.2 (44.3)	16.8 (8.2)	61.3 (5.1)
	$\gamma = 8$	169.9 (57.4)	19.6 (10.1)	60.2 (7.5)
	$\gamma = \infty$	186.0 (67.6)	23.6 (10.0)	61.0 (7.4)

- Conclusion of Pan et al (2010): best for variable selection, but not necessarily in prediction (PMSE).
  A surprise: $\gamma = \infty$ did not work well!
- Why?

		$\beta_1 = 5$			$\beta_2 = 1.58$		
Set-up	Methods	Mean	Var	MSE	Mean	Var	MSE
1	Lasso	5.28	8.69	8.69	1.43	2.43	2.42
	Enet	3.79	4.76	6.18	1.82	1.86	1.90
	Li&Li	5.00	1.69	1.67	1.74	1.33	1.34
	$\gamma = 2$	3.82	1.02	2.41	1.51	1.29	1.28
	$\gamma = 8$	3.47	0.79	3.12	1.50	1.02	1.02
	$\gamma = \infty$	2.13	1.33	9.57	1.64	2.08	2.06
		$\beta_1 = 5$			$\beta_2 = -1.58$		
2	Lasso	2.54	4.31	10.31	0.13	0.34	3.25
	Enet	2.87	4.85	9.32	0.16	0.41	3.44
	Li&Li	2.88	3.97	8.43	0.16	0.43	3.45
	$\gamma = 2$	1.37	0.79	14.00	0.22	0.28	3.53
	$\gamma = 8$	1.07	0.80	16.22	0.24	0.36	3.67
	$\gamma = \infty$	0.47	0.46	20.98	0.23	0.39	3.65

New Methods

- Q1: What is the comparative performance of GBL?
  GBL provides only *approximate* solution paths.
- Pan et al (2010): for a general γ, non-linear programming.
  Special case: $\gamma = \infty$, quadratic programming
- Use CVX package in Matlab!

Set-up	Methods	PMSE	q_1	q_0
1	Lasso	166.6 (32.9)	20.1 (2.5)	53.9 (6.4)
	Enet	164.3 (29.3)	10.6 (9.2)	31.4 (24.0)
	Li&Li	154.6 (28.3)	5.0 (7.6)	15.1 (21.2)
	$\gamma = 2$	138.1 (32.3)	3.2 (3.7)	60.0 (5.4)
	$\gamma = 8$	**132.0** (35.8)	3.2 (4.3)	60.0 (4.8)
	$\gamma = \infty$	162.9 (46.6)	7.3 (5.9)	56.6 (6.8)
	QP, $\gamma = \infty$	**126.6** (32.8)	1.1 (2.6)	56.1 (12.0)
2	Lasso	**160.8** (39.0)	30.2 (4.0)	61.1 (4.2)
	Enet	161.1 (45.5)	29.0 (8.5)	57.8 (15.1)
	Li&Li	161.7 (44.7)	26.0 (11.7)	52.1 (22.3)
	$\gamma = 2$	161.2 (44.3)	16.8 (8.2)	61.3 (5.1)
	$\gamma = 8$	169.9 (57.4)	19.6 (10.1)	60.2 (7.5)
	$\gamma = \infty$	186.0 (67.6)	23.6 (10.0)	61.0 (7.4)
	QP, $\gamma = \infty$	**143.1** (27.7)	9.5 (7.0)	51.6 (15.0)

		$\beta_1 = 5$			$\beta_2 = 1.58$		
Set-up	Methods	Mean	Var	MSE	Mean	Var	MSE
1	Lasso	5.28	8.69	8.69	1.43	2.43	2.42
	Enet	3.79	4.76	6.18	1.82	1.86	1.90
	Li&Li	5.00	1.69	1.67	1.74	1.33	1.34
	$\gamma = 2$	3.82	1.02	2.41	1.51	1.29	1.28
	$\gamma = 8$	3.47	0.79	3.12	1.50	1.02	1.02
	$\gamma = \infty$	2.13	1.33	9.57	1.64	2.08	2.06
	QP, $\gamma = \infty$	3.34	0.67	3.42	1.58	1.12	1.65
		$\beta_1 = 5$			$\beta_2 = -1.58$		
2	Lasso	2.54	4.31	10.31	0.13	0.34	3.25
	Enet	2.87	4.85	9.32	0.16	0.41	3.44
	Li&Li	2.88	3.97	8.43	0.16	0.43	3.45
	$\gamma = 2$	1.37	0.79	14.00	0.22	0.28	3.53
	$\gamma = 8$	1.07	0.80	16.22	0.24	0.36	3.67
	$\gamma = \infty$	0.47	0.46	20.98	0.23	0.39	3.65
	QP, $\gamma = \infty$	1.31	0.74	14.35	0.32	0.59	4.19

- Conclusion: better prediction, but still severely biased coef estimates!

Problem is not (likely) computational

- Q2: How to reduce (or eliminate) the bias?
- Tried ideas similar to adaptive Lasso, relaxed Lasso, an adaptive non-convex penlaty (TLP) ...
  BUT none worked!
  Why?
  To achieve two goals: variable selection and grouping
- New method: a 2-step procedure; similar to Li and Li (2010):
- Step 1: same as before,

$$p_\lambda = \lambda \sum_{i \sim j} \max\left(\frac{|\beta_i|}{\sqrt{d_i}}, \frac{|\beta_i|}{\sqrt{d_i}}\right)$$

- Step 2: force $\beta_i = \beta_j = 0$ if $\tilde{\beta}_i = \tilde{\beta}_j = 0$ and $i \sim j$, then use the

fused Lasso penalty:

$$p_\lambda = \lambda \sum_{i \sim j} \left| \frac{sgn(\tilde{\beta}_i)\beta_i}{\sqrt{d_i}} - \frac{sgn(\tilde{\beta}_j)\beta_j}{\sqrt{d_j}} \right|$$

- Use CVX package in Matlab!
  Both steps involve QP.
- A problem: depends on Step 1.
- Ideally in Step 2:

$$p_\lambda = \lambda \sum_{i \sim j} \left| \frac{|\beta_i|}{\sqrt{d_i}} - \frac{|\beta_j|}{\sqrt{d_j}} \right|$$

  but non-convex ...

Set-up	Methods	PMSE	q_1	q_0
1	Lasso	166.6 (32.9)	20.1 (2.5)	53.9 (6.4)
	Enet	164.3 (29.3)	10.6 (9.2)	31.4 (24.0)
	Li&Li	154.6 (28.3)	5.0 (7.6)	15.1 (21.2)
	$\gamma = 2$	138.1 (32.3)	3.2 (3.7)	60.0 (5.4)
	$\gamma = 8$	**132.0** (35.8)	3.2 (4.3)	60.0 (4.8)
	$\gamma = \infty$	162.9 (46.6)	7.3 (5.9)	56.6 (6.8)
	QP, $\gamma = \infty$	**126.6** (32.8)	1.1 (2.6)	56.1 (12.0)
	2-step, $\gamma = \infty$	**87.5** (17.6)	1.2 (2.7)	60.5 (11.9)
2	Lasso	**160.8** (39.0)	30.2 (4.0)	61.1 (4.2)
	Enet	161.1 (45.5)	29.0 (8.5)	57.8 (15.1)
	Li&Li	161.7 (44.7)	26.0 (11.7)	52.1 (22.3)
	$\gamma = 2$	161.2 (44.3)	16.8 (8.2)	61.3 (5.1)
	$\gamma = 8$	169.9 (57.4)	19.6 (10.1)	60.2 (7.5)
	$\gamma = \infty$	186.0 (67.6)	23.6 (10.0)	61.0 (7.4)
	QP, $\gamma = \infty$	**143.1** (27.7)	9.5 (7.0)	51.6 (15.0)
	2-step, $\gamma = \infty$	**130.2** (27.7)	10.2 (7.5)	56.1 (15.5)

		$\beta_1 = 5$			$\beta_2 = 1.58$		
Set-up	Methods	Mean	Var	MSE	Mean	Var	MSE
1	Lasso	5.28	8.69	8.69	1.43	2.43	2.42
	Enet	3.79	4.76	6.18	1.82	1.86	1.90
	Li&Li	5.00	1.69	1.67	1.74	1.33	1.34
	$\gamma = 2$	3.82	1.02	2.41	1.51	1.29	1.28
	$\gamma = 8$	3.47	0.79	3.12	1.50	1.02	1.02
	$\gamma = \infty$	2.13	1.33	9.57	1.64	2.08	2.06
	QP, $\gamma = \infty$	3.34	0.67	3.42	1.58	1.12	1.65
	2-step, $\gamma = \infty$	**5.00**	0.56	**0.56**	**1.49**	0.60	**0.60**
		$\beta_1 = 5$			$\beta_2 = -1.58$		
2	Lasso	2.54	4.31	10.31	0.13	0.34	3.25
	Enet	2.87	4.85	9.32	0.16	0.41	3.44
	Li&Li	2.88	3.97	8.43	0.16	0.43	3.45
	$\gamma = 2$	1.37	0.79	14.00	0.22	0.28	3.53
	$\gamma = 8$	1.07	0.80	16.22	0.24	0.36	3.67
	$\gamma = \infty$	0.47	0.46	20.98	0.23	0.39	3.65
	QP, $\gamma = \infty$	1.31	0.74	14.35	0.32	0.59	4.19
	2-step, $\gamma = \infty$	**3.09**	1.35	**4.98**	**0.31**	1.06	4.62

Discussion

- Penalty **and** computational algorithm matter!
- Can be extended to SVM (Zhu, Pan & Shen 2009, 2010);
- Develop more efficient penalties & algorithms (with X Shen);
- Relax the smoothness assumption:
  New assumption: neighboring genes are more likely to participate or not participate at the same time; no assumption on the smoothness of regression coefficients.
  Bayesian approaches (Moni and Li 2009; Li and Zhang 2009; Tai, Pan & Shen 2010)
  Penalized approaches?
- Penalty **and** computational algorithm matter!

Acknowledgement: This research was supported by NIH.

You can download our papers from
http://www.biostat.umn.edu/rrs.php

Thank you!