Stat 521A
Lecture 26

Structure learning in UGMs

- Dependency networks
- Gaussian UGMs
- Discrete UGMs

Dependency networks

- A simple way to learn a graph is to regress each node on all others, $p(x_i | x_{-i})$
- If the full conditionals are sparse, this gives rise to a sparse graph
- Heckerman et al used classification trees to do variable selection
- Meinshausen & Buhlman proved that if you use lasso, the method is a consistent estimator of graph structure
- Wainwright et al extended the proof to L1 penalized logistic regression

Problem with depnets

- Although one can recover the structure, the params of the full conditionals need not correspond to any consistent joint
- To estimate params given the graph can be computationally hard (esp for discrete variables)
- Only give a point estimate of the structure*

* Parent fusion project

Bayesian inference for GGMs

- If we use decomposable graphical models, we can use the hyper inverse wishart as a conjugate prior, and hence compute p(D|G) analytically
- Problem reduces to discrete search
- Can use MCMC, MOSS, etc
- For non-decomposable models, have to approximate p(D|G) eg by BIC. Have to compute MLE for every neighboring graph! *
- See work by Adrian Dobra.

* Derive analog of structural EM to speed this up – nips project, anyone?

Graphical lasso

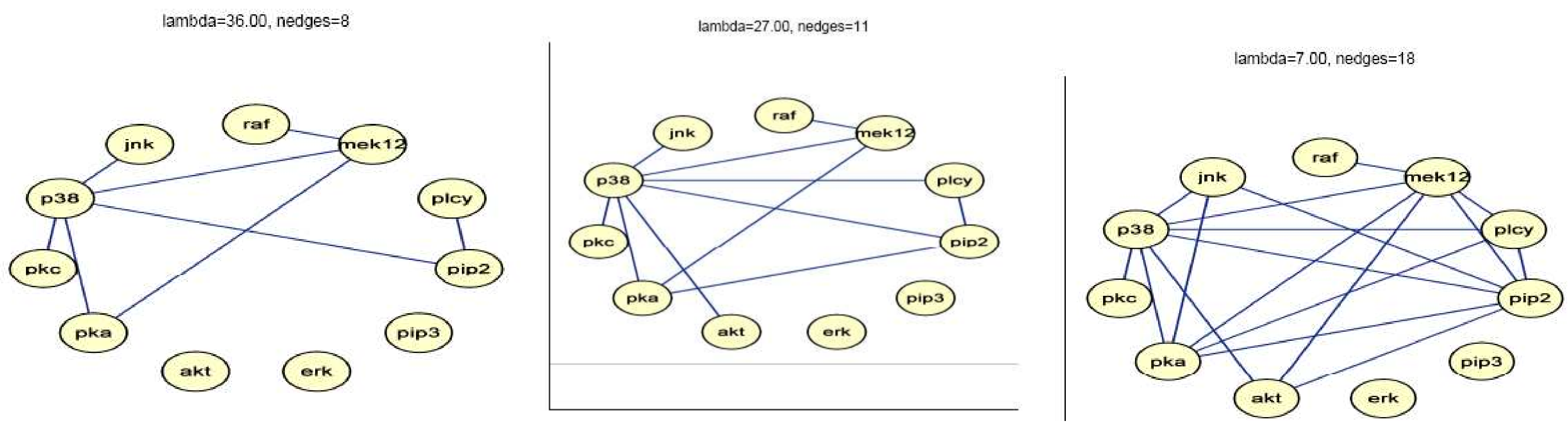
- We can estimate parameters and structure for GGMs simultaneously by optimizing

$$f(\Omega) = \log\det\Omega - \mathrm{tr}(S\Omega) - \lambda||\Omega||_1$$

$$e\ ||\Omega||_1 = \sum_{j,k} |\omega_{jk}|$$

- Convex
- Can solve in O(#iter d^4) time by solving a sequence of lasso subproblems

Example


lambda=36.00, nedges=8
jnk
raf
mek12
p38
plcy
pkc
pip2
pka
pip3
akt
erk
lambda=27.00, nedges=11
jnk
raf
mek12
p38
plcy
pkc
pip2
pka
pip3
akt
erk
lambda=7.00, nedges=18
raf
jnk
mek12
p38
plcy
pkc
pip2
pka
pip3
akt
erk

MLE params for GGM

- Consider first the problem of estimating Ω given known zeros (absent edges)

$$\ell_C(\Omega) = \log\det\Omega - \mathrm{tr}(\mathbf{S}\Omega) - \sum_{(j,k)\notin E(G)} \gamma_{jk}\Omega_{jk}$$

- Setting gradient to zero gives

$$\Omega^{-1} - \mathbf{S} - \Gamma = 0 \qquad \mathbf{w}_{12} - \mathbf{s}_{12} - \gamma_{12} = 0$$

Let j be a specific node in group 1. Then if $G_{j2} \neq 0$, then $\gamma_{j2} = 0$, so $w_{j2} = s_{j2}$. In other words, edges that are not constrained to be zero must have an MLE covariance equal to the empirical covariance.

- Consider this partition

$$\begin{pmatrix} \mathbf{W}_{11} & \mathbf{w}_{12} \\ \mathbf{w}_{12}^T & w_{22} \end{pmatrix} \begin{pmatrix} \Omega_{11} & \omega_{12} \\ \omega_{12}^T & \omega_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & 0 \\ 0^T & 1 \end{pmatrix}$$

$$\mathbf{w}_{12} = -\mathbf{W}_{11}\omega_{12}/\omega_{22} = \mathbf{W}_{11}\beta$$

e $\beta \stackrel{\text{def}}{=} -\omega_{12}/\omega_{22}$.

$$\mathbf{W}_{11}\beta - \mathbf{s}_{12} - \gamma_{12} = 0$$

Cont'd

- We have $\mathbf{W}_{11}\beta - \mathbf{s}_{12} - \gamma_{12} = \mathbf{0}$
- Dropping the zeros $\mathbf{W}_{11}^{*}\beta^{*} - \mathbf{s}_{12}^{*} = \mathbf{0}$
- Can recover Ω from weights using $\omega_{12} = -\beta_{12}\omega_{22}$
- To find w_22, use block inversion lemma

$$\omega_{22} = (\mathbf{W}/\mathbf{W}_{11})^{-1} = (w_{22} - \mathbf{w}_{12}^{T}\mathbf{W}_{11}^{-1}\mathbf{w}_{12})^{-1}$$

Now $\mathbf{W}_{11}^{-1}\mathbf{w}_{12} = (\mathbf{W}_{11}^{*})^{-1}\mathbf{s}_{12}^{*} = (\beta, \mathbf{0})$, since $\mathbf{w}_{12} = \mathbf{s}_{12}$ in all locations that are not constrained to be zero. Similarly, $w_{22} = s_{22}$. Hence

$$\frac{1}{\omega_{22}} = s_{22} - \mathbf{w}_{12}^{T}\beta \qquad (3.82)$$

code

```
W = S; % W = inv(precMat)
precMat = zeros(p,p);
beta = zeros(p-1,1);
iter = 1;
converged = false;
normW = norm(W);
while ~converged
  for i = 1:p
    % partition W & S for i
    noti = [1:i-1 i+1:p];
    W11 = W(noti,noti);
    w12 = W(noti,i);
    s22 = S(i,i);
    s12 = S(noti,i);

    % find G's non-zero index in W11
    idx = find(G(noti,i));  % non-zeros in G11
    beta(:) = 0;
    beta(idx) = W11(idx,idx) \ s12(idx);

    % update W
    w12 = W11 * beta;
    W(noti,i) = w12 ;
    W(i,noti) = w12';

    % update precMat (technically only needed on last iteration)
    p22 = max([0  1/(s22 - w12'*beta)]);  % must be non-neg
    p12 = -beta * p22;
    precMat(noti,i) = p12 ;
    precMat(i,noti) = p12';
    precMat(i,i) = p22;
  end
  converged =  convergenceTest(norm(W), normW) || (iter > maxIter);
  normW = norm(W);
  iter = iter + 1;
end
```

ggmFitHtf in pmtk (by Baback Moghaddam)

Example

Let us now give a worked example of this algorithm. Let the input be the following adjacency matrix, representing the cyclic structure, $X_1 - X_2 - X_3 - X_4 - X_1$, and empirical covariance matrix:

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} 10 & 1 & 5 & 4 \\ 1 & 10 & 2 & 6 \\ 5 & 2 & 10 & 3 \\ 4 & 6 & 3 & 10 \end{pmatrix} \tag{3.83}$$

After 3 iterations we converge to the following MLE:

$$\mathbf{\Sigma} = \begin{pmatrix} 10.00 & 1.00 & \mathbf{1.31} & 4.00 \\ 1.00 & 10.00 & 2.00 & \mathbf{0.87} \\ \mathbf{1.31} & 2.00 & 10.00 & 3.00 \\ 4.00 & \mathbf{0.87} & 3.00 & 10.00 \end{pmatrix}, \quad \mathbf{\Omega} = \begin{pmatrix} 0.12 & -0.01 & \mathbf{0} & -0.05 \\ -0.01 & 0.11 & -0.02 & \mathbf{0} \\ \mathbf{0} & -0.02 & 0.11 & -0.03 \\ -0.05 & \mathbf{0} & -0.03 & 0.13 \end{pmatrix} \tag{3.84}$$

Graphical lasso

$$f(\mathbf{\Omega}) = \log \det \mathbf{\Omega} - \mathrm{tr}(\mathbf{S}\mathbf{\Omega}) - \lambda||\mathbf{\Omega}||_1 \qquad \lambda_{jj} \geq 0, \lambda_{jk}^{max} = |\hat{\Sigma}_{jk}|$$

The basic idea is very similar to the method in Section 3.3.7, except we replace the least squares subproblem with a lasso subproblem. The analog of the gradient equation (3.75) is the following:

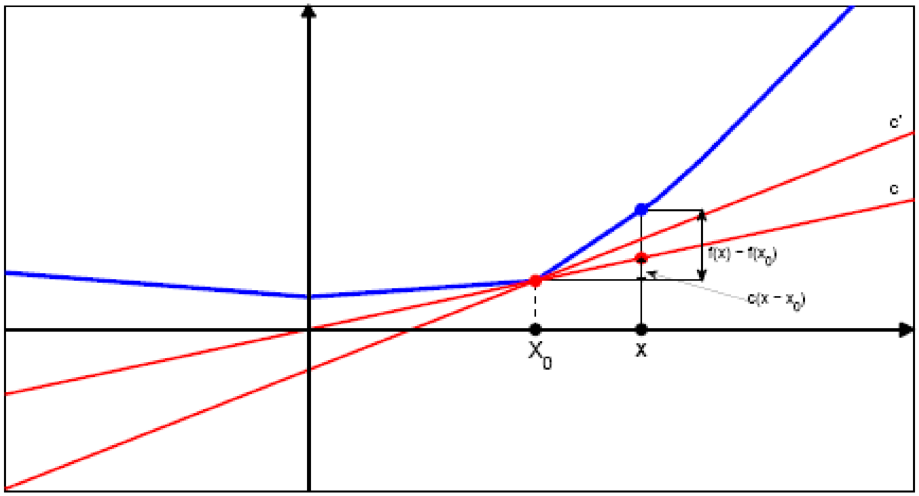
$$\mathbf{\Omega}^{-1} - \mathbf{S} - \lambda \mathrm{Sign}(\mathbf{\Omega}) = \mathbf{0} \tag{3.86}$$

As discussed in Section ??, we must replace the gradient with the subgradient, due to the non differentiable penalty term. So we define $\mathrm{Sign}(\omega_{jk}) = \mathrm{sign}(\omega_{jk})$ if $\omega_{jk} \neq 0$, and $\mathrm{Sign}(\omega_{jk}) \in [-1, 1]$ otherwise. The analogous result to Equation 3.79 is

$$\mathbf{W}_{11}\boldsymbol{\beta} - \mathbf{s}_{12} + \lambda \mathrm{Sign}(\boldsymbol{\beta}) = \mathbf{0} \tag{3.87}$$

since $\boldsymbol{\beta}$ and $\boldsymbol{\omega}_{12}$ have opposite signs.

Subgradients



We can generalize the notion of derivative to handle this case as follows. We define a **subderivative** of a function $f : \mathcal{I} \rightarrow \mathbb{R}$ at a point x_0 to be a scalar c such that

$$f(x) - f(x_0) \geq c(x - x_0) \forall x \in \mathcal{I} \tag{29.84}$$

where $\mathcal{I}$ is some interval containing x_0. See Figure 29.16. We define the *set* of subderivatives as the interval $[a, b]$ where a and b are the one-sided limits

$$a = \lim_{x \rightarrow x_0^-} \frac{f(x) - f(x_0)}{x - x_0}, \quad b = \lim_{x \rightarrow x_0^+} \frac{f(x) - f(x_0)}{x - x_0} \tag{29.85}$$

The set $[a, b]$ of all subderivatives is called the **subdifferential** of the function f at x_0 and is denoted $\partial f(x)|_{x_0}$. For example, the subdifferential of the absolute value function $f(x) = |x|$ is

$$\partial f(x) = \begin{cases} \{-1\} & \text{if } x < 0 \\ [-1, 1] & \text{if } x = 0 \\ \{+1\} & \text{if } x > 0 \end{cases} \tag{29.86}$$

If the function is everywhere differentiable, then $\partial f(x) = \{\frac{df(x)}{dx}\}$. By analogy to the standard calculus result, one can show that the point $\hat{x}$ is a local minimum of f iff $0 \in \partial f(x)$.

Graphical lasso

$$f(\mathbf{\Omega}) = \log\det\mathbf{\Omega} - \text{tr}(\mathbf{S}\mathbf{\Omega}) - \lambda||\mathbf{\Omega}||_1$$

The basic idea is very similar to the method in Section 3.3.7, except we replace the least squares subproblem with a lasso subproblem. The analog of the gradient equation (3.75) is the following:

$$\mathbf{\Omega}^{-1} - \mathbf{S} - \lambda\text{Sign}(\mathbf{\Omega}) = \mathbf{0} \tag{3.86}$$

As discussed in Section ??, we must replace the gradient with the subgradient, due to the non differentiable penalty term. So we define $\text{Sign}(\omega_{jk}) = \text{sign}(\omega_{jk})$ if $\omega_{jk} \neq 0$, and $\text{Sign}(\omega_{jk}) \in [-1, 1]$ otherwise. The analogous result to Equation 3.79 is

$$\mathbf{W}_{11}\beta - \text{s}_{12} + \lambda\text{Sign}(\beta) = \mathbf{0} \tag{3.87}$$

since β and ω_{12} have opposite signs.

This is equivalent to a lasso problem. To see this, consider the objective

$$J(\beta) = \frac{1}{2}(\text{y} - \mathbf{Z}\beta)^T(\text{y} - \mathbf{Z}\beta) + \lambda||\beta||_1 \tag{3.88}$$

Setting the gradient to zero we get

$$\mathbf{Z}^T\mathbf{Z}\beta - \mathbf{Z}^T\text{y} + \lambda\text{Sign}(\beta) = \mathbf{0} \tag{3.89}$$

We see that $\mathbf{Z}^T\text{y}$ is similar to s_{12} (namely an estimate of the covariance between target and inputs), and that $\mathbf{Z}^T\mathbf{Z}$ gets replaced by $\mathbf{W}_{11}$, which represents correlation amongst the current inputs.

Shooting (coord desc for lasso)

We now present a **coordinate descent** algorithm called **shooting** [Fu98] for solving the unconstrained lasso problem:

$$J(\mathbf{w}, \lambda) = RSS(\mathbf{w}) + \lambda \sum_{j=1}^{d} |w_j| \tag{17.36}$$

Besides being simple and fast, this method yields additional insight into why an L1 regularizer results in a sparse solution.

We can compute the partial derivative of the lasso objective function wrt a particular parameter, say w_k as follows. One can show (Exercise 17) that

$$\frac{\partial}{\partial w_k} RSS(\mathbf{w}) = a_k w_k - c_k \tag{17.37}$$

$$a_k = 2 \sum_{i=1}^{n} x_{ik}^2 \tag{17.38}$$

$$c_k = 2 \sum_{i=1}^{n} x_{ik}(y_i - \mathbf{w}_{-k}^T \mathbf{x}_{i,-k}) \tag{17.39}$$

$$= 2 \sum_{i=1}^{n} \left[x_{ik} y_i - x_{ik} \mathbf{w}^T \mathbf{x}_i + w_k x_{ik}^2 \right] \tag{17.40}$$

where $\mathbf{w}_{-k} = \mathbf{w}$ without component k, and similarly for $\mathbf{x}_{i,-k}$. We see that c_k is (proportional to) the correlation between the k'th feature $\mathbf{x}_{:,k}$ and the residual due to the other features, $\mathbf{r}_{-k} = \mathbf{y} - \mathbf{X}_{:,-k}\mathbf{w}_{-k}$; if this correlation is zero, then feature k would be orthogonal to the residual, and we couldn't reduce the RSS by updating w_k. Hence the magnitude of c_k is an indication of how relevant feature k is for predicting $\mathbf{y}$ (relative to the other features and the current parameters).

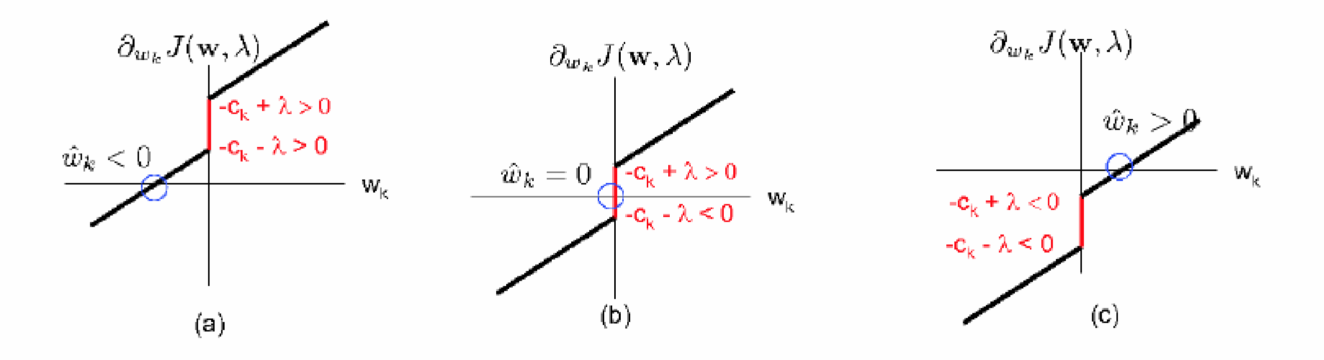
Shooting cont'd

The L1 penalty function is not differentiable, so we need to compute the **subdifferential** (see Section 29.6.1) rather than the standard differential. This is given by

$$\partial_{w_k} J(\mathbf{w}, \lambda) = (a_k w_k - c_k) + \lambda \partial_{w_k} ||\mathbf{w}||_1 \quad (17.41)$$

$$= \begin{cases} \{a_k w_k - c_k - \lambda\} & \text{if } w_k < 0 \\ [-c_k - \lambda, -c_k + \lambda] & \text{if } w_k = 0 \\ \{a_k w_k - c_k + \lambda\} & \text{if } w_k > 0 \end{cases} \quad (17.42)$$

This subdifferential is a piecewise linear function of w_k. Since $a_k > 0$, it is sloping up and to the right, except it has a vertical "kink" in it at $w_k = 0$, spanning the range $[-c_k - \lambda, -c_k + \lambda]$: see Figure 17.6. Depending on the value of c_k, the solution to $\partial_{w_k} J(\mathrm{w}, \lambda) = 0$ can occur at 3 different values of w_k, as follows:



Soft thresholding

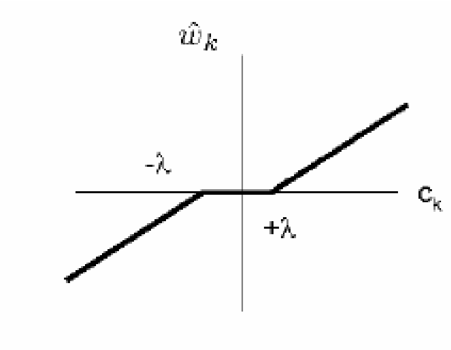
1. $c_k < -\lambda$, so the feature is strongly negatively correlated with the residual. In this case, the subgradient is zero at $\hat{w}_k = \frac{c_k+\lambda}{a_k} < 0$.

2. $c_k \in [-\lambda, \lambda]$, so the feature is only weakly correlated with the residual. In this case, the subgradient is zero at $\hat{w}_k = 0$. Thus if the correlation is not less than λ, we set the corresponding coefficient to 0.

3. $c_k > \lambda$, so the feature is strongly positively correlated with the residual. In this case, the subgradient is zero at $\hat{w}_k = \frac{c_k-\lambda}{a_k} > 0$.

In summary, we have

$$\hat{w}_k(c_k) = \begin{cases} (c_k+\lambda)/a_k & \text{if } c_k < -\lambda \\ 0 & \text{if } c_k \in [-\lambda, \lambda] \\ (c_k-\lambda)/a_k & \text{if } c_k > \lambda \end{cases} \quad (17.43)$$



$$\hat{w}_k = \text{soft}\left(\frac{c_k}{a_k}; \frac{\lambda}{a_k}\right)$$

$$\text{soft}(a;\delta) = \text{sign}(a)\max\{0, |a|-\delta\} = \text{sign}(a)\left(|a|-\delta\right)_+$$

Lasso vs ridge vs subset selection

For orthonormal features, we have explicit solns

the lasso solution as follows (using the fact that $a_k = 2$ and $\hat{w}_k^{OLS} = c_k/2$)

$$\hat{w}_k^{lasso} = \text{sign}(\hat{w}_k^{OLS}) \left(|\hat{w}_k^{OLS}| - \frac{\lambda}{2} \right)_+ \tag{17.46}$$

By contrast, the ridge estimate would be

$$\hat{w}_k^{ridge} = \frac{\hat{w}_k^{OLS}}{1+\lambda} \tag{17.47}$$

which does not force sparsity. If we pick the best K features using subset selection, the parameter estimate is as follows

$$\hat{w}_k^{SS} = \begin{cases} \hat{w}_k^{OLS} & \text{if rank}(|w_k|) \leq K \\ 0 & \text{otherwise} \end{cases} \tag{17.48}$$

Graphical lasso with shooting

$$f(\mathbf{\Omega}) = \log\det \mathbf{\Omega} - \mathrm{tr}(\mathbf{S}\mathbf{\Omega}) - \lambda||\mathbf{\Omega}||_1$$

The basic idea is very similar to the method in Section 3.3.7, except we replace the least squares subproblem with a lasso subproblem. The analog of the gradient equation (3.75) is the following:

$$\mathbf{\Omega}^{-1} - \mathbf{S} - \lambda \mathrm{Sign}(\mathbf{\Omega}) = \mathbf{0} \tag{3.86}$$

As discussed in Section ??, we must replace the gradient with the subgradient, due to the non differentiable penalty term. So we define $\mathrm{Sign}(\omega_{jk}) = \mathrm{sign}(\omega_{jk})$ if $\omega_{jk} \neq 0$, and $\mathrm{Sign}(\omega_{jk}) \in [-1, 1]$ otherwise. The analogous result to Equation 3.79 is

$$\mathbf{W}_{11}\boldsymbol{\beta} - \mathbf{s}_{12} + \lambda \mathrm{Sign}(\boldsymbol{\beta}) = \mathbf{0} \tag{3.87}$$

since $\boldsymbol{\beta}$ and $\boldsymbol{\omega}_{12}$ have opposite signs.

This is equivalent to a lasso problem. To see this, consider the objective

$$J(\boldsymbol{\beta}) = \frac{1}{2}(\mathbf{y} - \mathbf{Z}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{Z}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_1 \tag{3.88}$$

Setting the gradient to zero we get

$$\mathbf{Z}^T\mathbf{Z}\boldsymbol{\beta} - \mathbf{Z}^T\mathbf{y} + \lambda \mathrm{Sign}(\boldsymbol{\beta}) = \mathbf{0} \tag{3.89}$$

We see that $\mathbf{Z}^T\mathbf{y}$ is similar to $\mathbf{s}_{12}$ (namely an estimate of the covariance between target and inputs), and that $\mathbf{Z}^T\mathbf{Z}$ gets replaced by $\mathbf{W}_{11}$, which represents correlation amongst the current inputs.

One simple way to solve this lasso problem is to use coordinate descent, known as the **shooting algorithm** (see Section ??). To apply this to the current problem, let $\mathbf{V} = \mathbf{W}_{11}$. (Recall $\mathbf{W} = \mathbf{\Sigma}$.) Then the update for $\boldsymbol{\beta}$ becomes

$$\beta_j := S_\lambda\left(s_{12j} - \sum_{k \neq j} V_{kj}\beta_k\right)/V_{jj} \tag{3.90}$$

where S is the soft-threshold operator

$$S_t(x) = \mathrm{sign}(x)\max(0, |x| - t) \tag{3.91}$$

We can implement this in a way which is very similar to Listing ??. The only change is to replace the line `beta(idx) = W11(idx,idx) \ s12(idx)` with the code shown below.

Discrete UGMs

- Computing Z and hence the likelihood is intractable unless the graph is decomposable
- Hence Bayesian methods “never” used
- Even search and score is inefficient

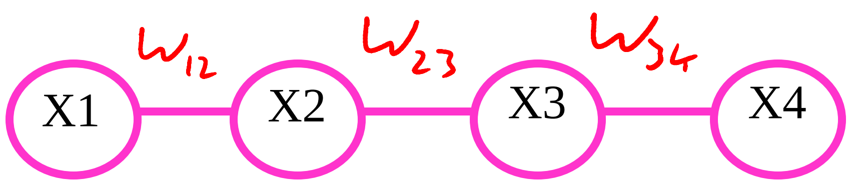
Ising models

- Analogous to GGM for binary data

$$\mathcal{N}(\mathbf{x}|\mathbf{K}) = \frac{1}{Z(\mathbf{K})} \exp(-\tfrac{1}{2} \sum_{j,k} K_{j,k} x_j x_k), \quad x_j \in \mathbb{R}$$

$$p(\mathbf{x}|\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp(\sum_{j,k} W_{jk} x_j x_k), \quad x_j \in \{-1, +1\}$$

$$\mathbf{W} = \begin{pmatrix} W_{11} & W_{12} & 0 & 0 \\ W_{21} & W_{22} & W_{23} & 0 \\ 0 & W_{32} & W_{33} & W_{34} \\ 0 & 0 & W_{43} & W_{44} \end{pmatrix}$$



$$X_j \perp X_{-j} | X_{N_j}$$

Markov property

$w_{jk} \geq 0$	attractive (ferro magnet)
$w_{jk} \leq 0$	repuslive (anti ferro magnetic)
w_{jk} mixed sign	frustrated system

Besag, Hammersley & Clifford, Geman & Geman

Glasso for Ising models (Banerjee)

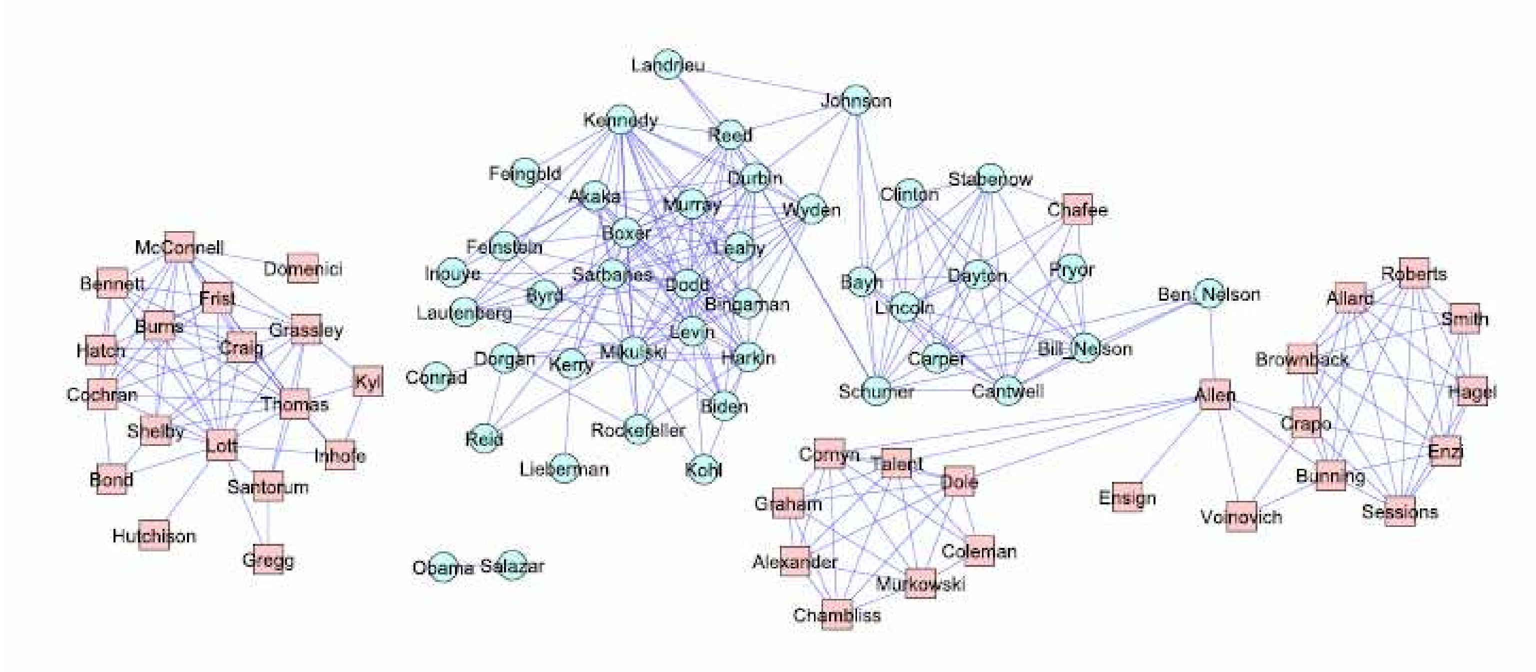
$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z}\exp[\sum_{i=1}^{d-1}\sum_{j=i+1}^{d} W_{ij}x_i x_j]$$

$$Z = \sum_{\mathbf{x}\in\{-1,+1\}^d} \exp[\sum_{i=1}^{d-1}\sum_{j=i+1}^{d} W_{ij}x_i x_j]$$

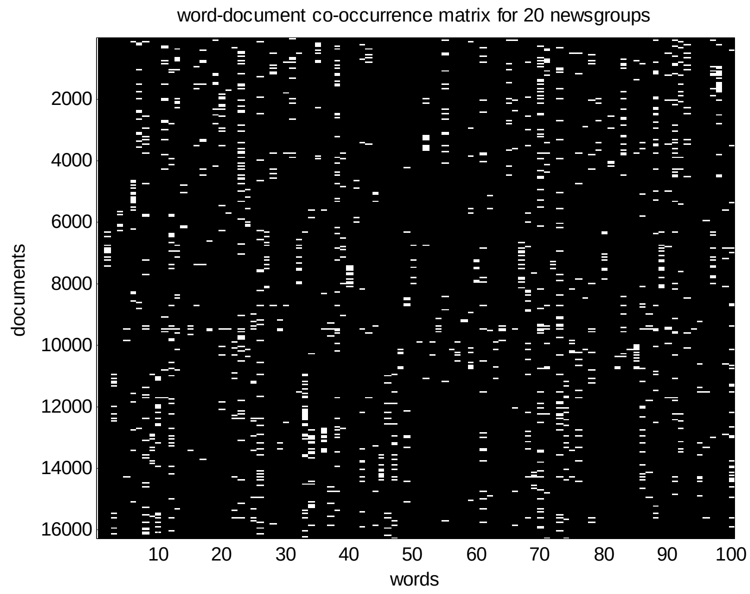
Convex relaxation of matrix permanent to matrix determinant

$$\hat{\mathbf{W}} = \text{graphicalLasso}(\text{Cov}(\mathbf{X}) - \lambda\mathbf{I} + \frac{1}{3}\mathbf{I},\ \ \lambda)$$

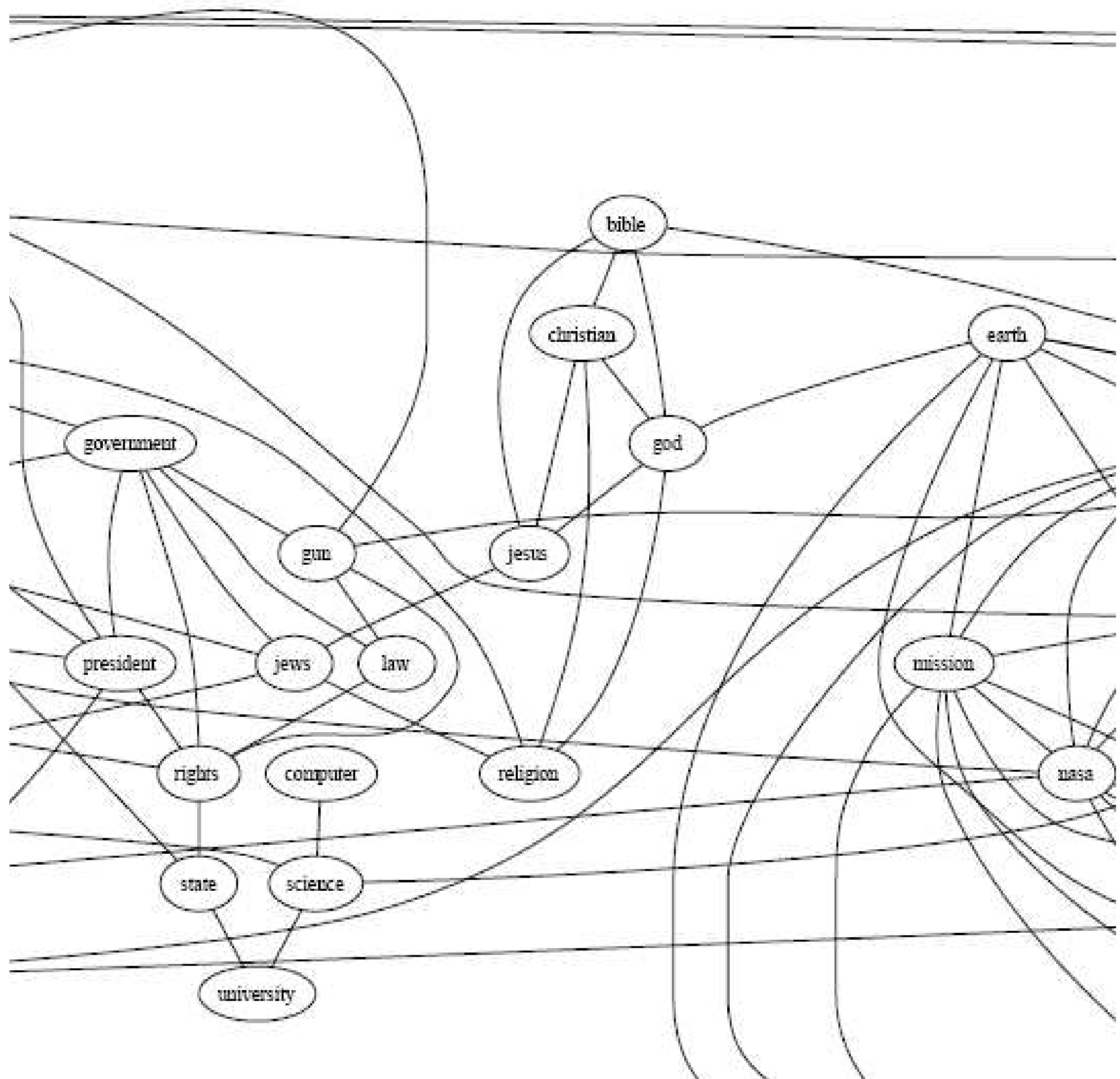
Senate voting data



20 newsgroups



n=16,000,   d=100



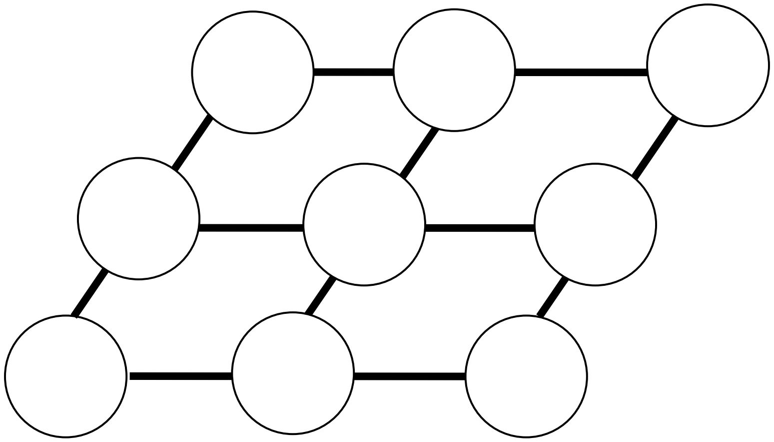
Courtesy Mark Schmidt

Markov random fields

- Markov random fields for $y_j \in \{1,\ldots,K\}$

$$p(\mathbf{y}|\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp(\sum_{j,k} \mathbf{w}_{jk}^T \mathbf{f}_{jk}(y_j, y_k)) \propto \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}))$$

y_j	y_k	$\mathbf{f}_{jk}(y_j, y_k)$
1	1	$(1,0,0,0,0,0,0,0,0)$
1	2	$(0,1,0,0,0,0,0,0,0)$
1	3	$(0,0,1,0,0,0,0,0,0)$
2	1	$(0,0,0,1,0,0,0,0,0)$
$\ldots$		
3	3	$(0,0,0,0,0,0,0,0,1)$



Parameter *vector* on each edge

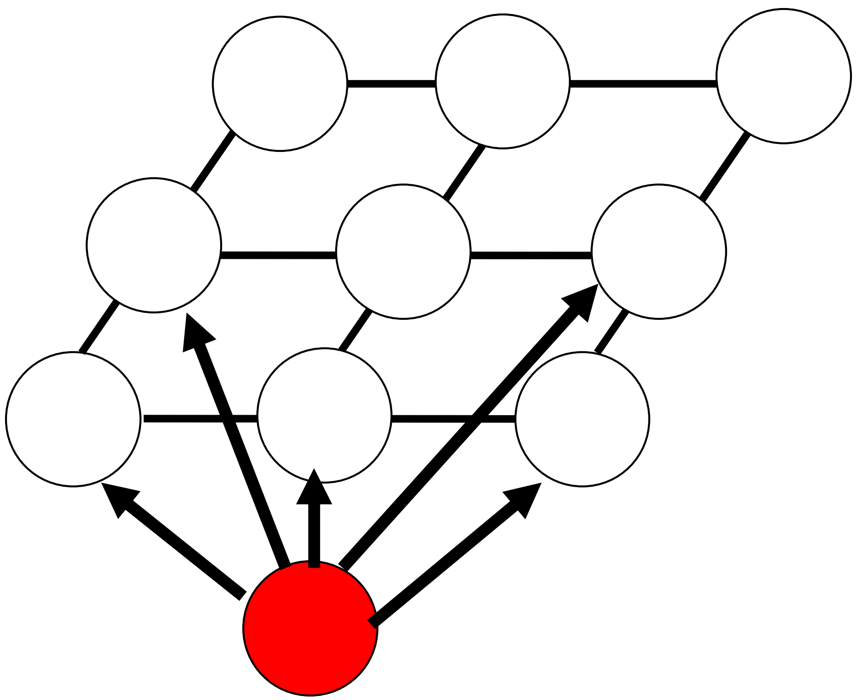
- No longer a 1:1 mapping between G and W

Conditional random fields

- CRFs are a conditional density model

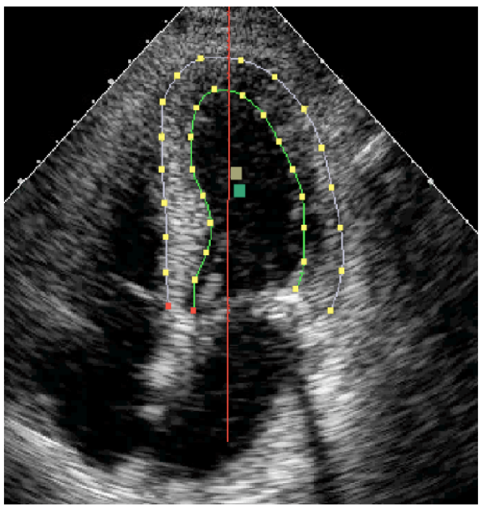
$$p(\mathbf{y}|\mathbf{x}, \mathbf{W}, \mathbf{V}) = \frac{1}{Z(\mathbf{W}, \mathbf{V}, \mathbf{x})} \exp(\sum_{j,k} \mathbf{w}_{j,k}^T \mathbf{f}_{jk}(y_j, y_k, \mathbf{x}) + \sum_j \mathbf{v}_j^T \mathbf{g}_j(y_j, \mathbf{x}))$$

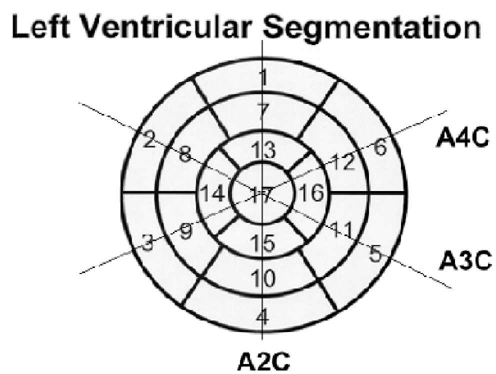
- No longer a 1:1 mapping between G and W

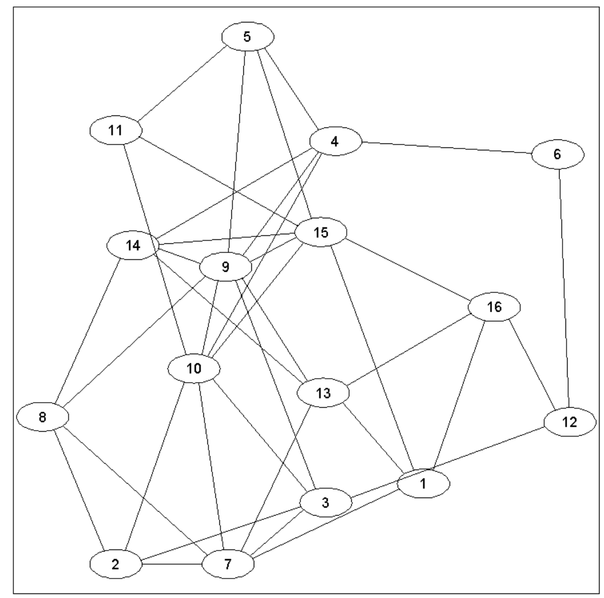


Heart wall abnormality data

- d=16, n=345, $y_j \in \{0,1\}$ representing normal or abnormal segment, x_j in R^{100} representing features derived from image processing







“Structure Learning in Random Fields for Heart Motion Abnormality Detection”
Mark Schmidt, Kevin Murphy, Glenn Fung, Romer Rosales.
CVPR 2008.

Group L1 regularization

- Solution: penalize groups of parameters, one group per edge

$$
\begin{aligned}
J(\mathbf{w}, \mathbf{v}) &= -\log \sum_i p(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w}, \mathbf{v}) + \lambda_2 ||\mathbf{v}||_2^2 + \lambda_1 \sum_g ||\mathbf{w}_g||_p \\
||\mathbf{w}||_2 &= \sqrt{\sum_k w_k^2} \\
||\mathbf{w}||_\infty &= \max_k |w_k|
\end{aligned}
$$

Group lasso

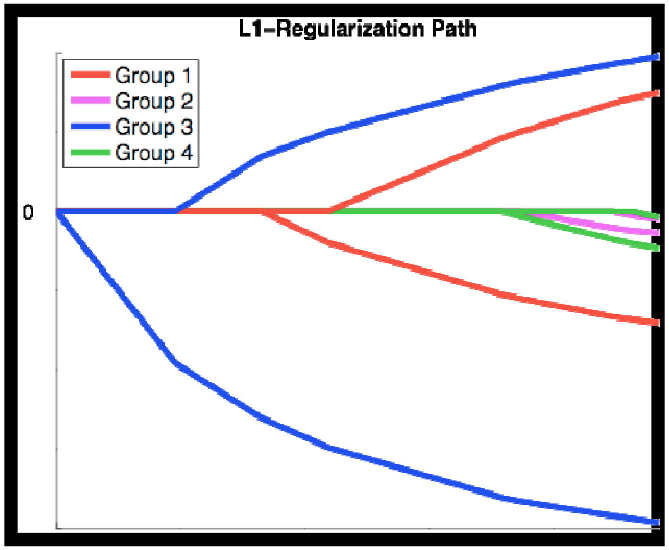
- Sometimes we want to select groups of parameters together (e.g., when encoding categorical inputs)

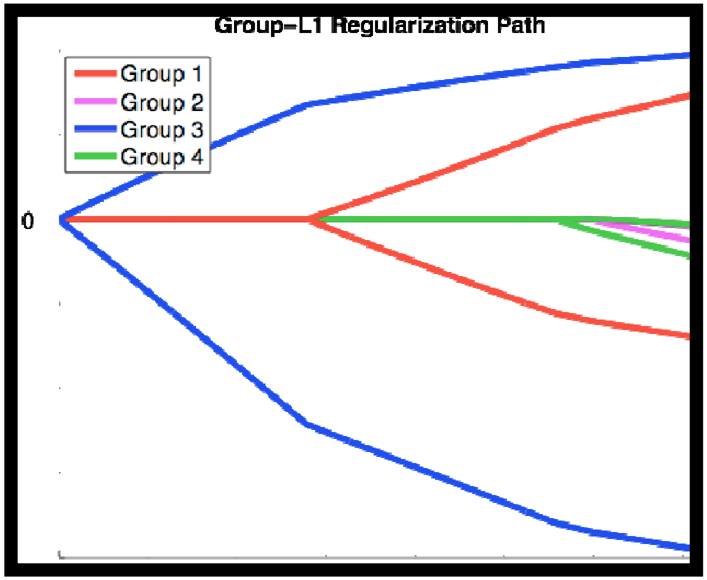
$$\hat{\mathbf{w}} = \arg\min RSS(\mathbf{w}) + \lambda R(\mathbf{w})$$

$$R(\mathbf{w}) = \sum_g ||\mathbf{w}_g||_2 = \sum_g \sqrt{\sum_{j \in g} w_{gj}^2}$$

$$R(\mathbf{w}) = \sum_g ||\mathbf{w}_g||_\infty = \sum_g \max_{j \in g} |w_{gj}|$$

Still convex, but much harder to optimize…





Group L1 for graphs

- Penalize groups of parameters, one group per edge

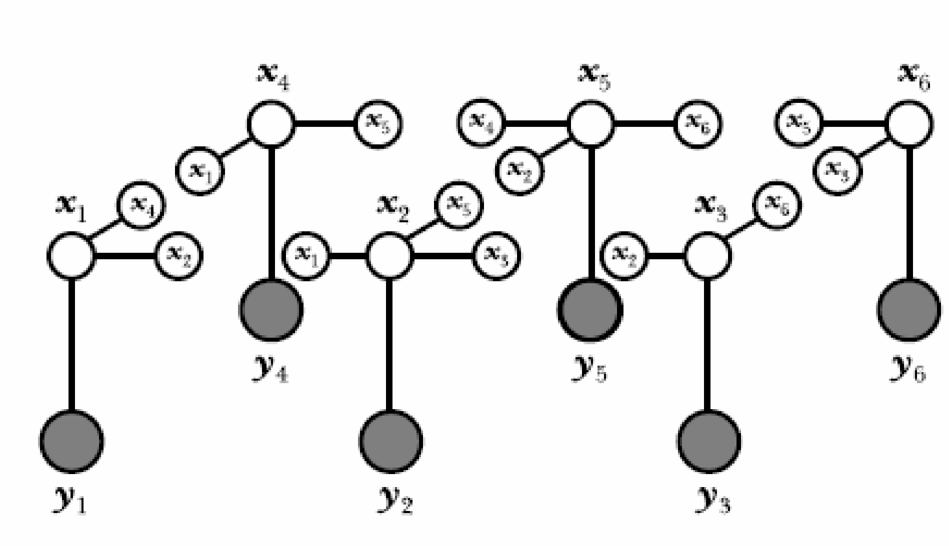
$$
\begin{aligned}
J(\mathbf{w}, \mathbf{v}) &= -\log \sum_i p(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w}, \mathbf{v}) + \lambda_2 ||\mathbf{v}||_2^2 + \lambda_1 \sum_g ||\mathbf{w}_g||_p \\
||\mathbf{w}||_2 &= \sqrt{\sum_k w_k^2} \\
||\mathbf{w}||_\infty &= \max_k |w_k|
\end{aligned}
$$

- Issues
  - How deal with intractable log-likelihood? Use PL (Schmidt) or LBP (Lee & Koller)
  - How handle non-smooth penalty functions? (Projected gradient or projected quasi newton)

Pseudo likelihood

- PL is locally normalized

$$
\begin{aligned}
L(\mathbf{W}) &= \prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{W}) = \prod_{i=1}^{n} \frac{1}{Z(\mathbf{W})} \exp(\sum_j \sum_k x_{ij} W_{jk} x_{ik}) \\
PL(\mathbf{W}) &= \prod_{i=1}^{n} \prod_{j=1}^{d} p(x_{ij}|\mathbf{x}_{i,n_i}, \mathbf{w}_{j,:}) \\
&= \prod_j \prod_i \frac{1}{Z(\mathbf{w}_j, \mathbf{x}_{i,N_j})} \exp(x_{ij} \sum_k W_{jk} x_{ik}) \\
Z(\mathbf{w}_j, \mathbf{x}_{N_j}) &= \sum_{x_j \in \{-1,+1\}} \exp(x_j \sum_{k \in N_j} W_{jk} x_k)
\end{aligned}
$$



Constrained formulation

- Convert penalized negative log pseudo likelihood

$$f(\mathbf{w},\mathbf{v}) = -\log \sum_i PL(\mathbf{y}_i|\mathbf{x}_i,\mathbf{v},\mathbf{w}) + \lambda_2 ||\mathbf{v}||_2^2$$

$$\min_{\mathbf{w},\mathbf{v}} = f(\mathbf{w},\mathbf{v}) + \lambda_1 \sum_g ||\mathbf{w}_g||_p$$

- into constrained form

$$L(\boldsymbol{\alpha},\mathbf{w},\mathbf{v}) = f(\mathbf{w},\mathbf{v}) + \lambda_1 \sum_g \alpha_g$$

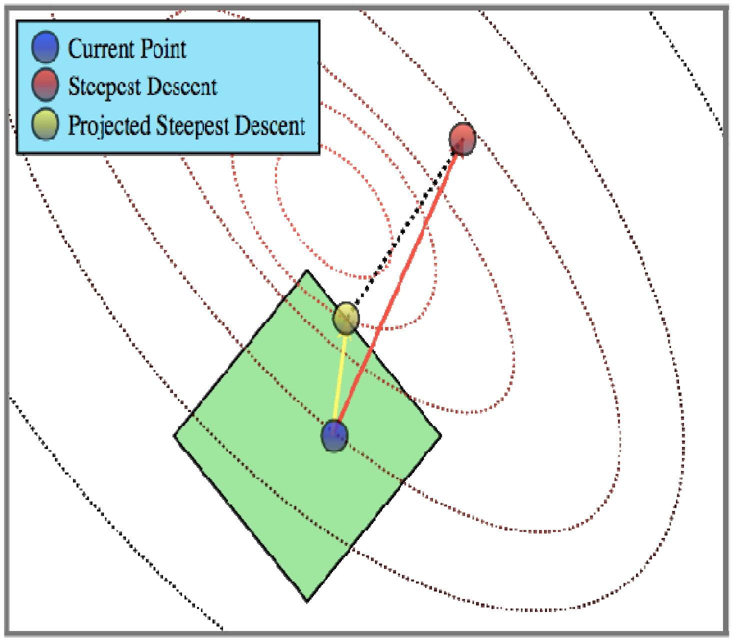
$$\min_{\boldsymbol{\alpha},\mathbf{w},\mathbf{v}} = L(\boldsymbol{\alpha},\mathbf{w},\mathbf{v}) \text{ st } \forall g . \alpha_g \geq ||\mathbf{w}_g||_p$$

Desiderata for an optimizer

- Must handle $\binom{d}{2}$ groups (d = 16 in our application, so 120 groups)
- Must handle 100s features per group
- Cannot use second-order information (Hessian too expensive to compute or store) – so interior point is out
- Must converge quickly

Projected gradient method

- At each step, we perform an efficient projection onto the convex constraint set



$$
\begin{aligned}
\mathbf{x}_k &= (\boldsymbol{\alpha}, \mathbf{w})_k \\
\mathbf{x}_{k+1} &= t\Pi_{S_p}(\mathbf{x}_k - \beta \mathbf{g}_k) \\
\mathbf{g}_k &= \nabla f(\mathbf{x})_{\mathbf{x}_k} \\
\Pi_{\mathcal{S}}(\mathbf{x}) &= \arg\min_{\mathbf{x}^* \in \mathcal{S}} ||\mathbf{x} - \mathbf{x}^*||_2 \\
\mathcal{S}_p &= \{\mathbf{x} : \forall g. \alpha_g \geq ||\mathbf{w}_g||_p\}
\end{aligned}
$$

Project each group separately.

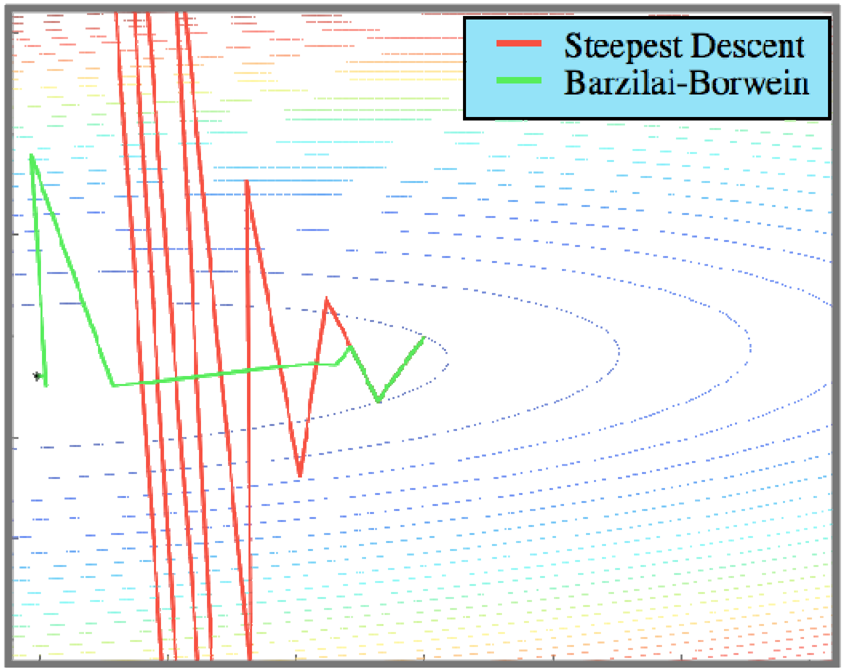
Takes O(N) time for p=2,
O(N log N) time for p=∞,

Where N = #params per group.

van der Berg, Schmidt, Friedlander, Murphy; Duchi et al.

Spectral step size

- Gradient descent can be slow
- Barzilai and Borwein proposed the following stepsize, which in some cases enjoys super-linear convergence rates

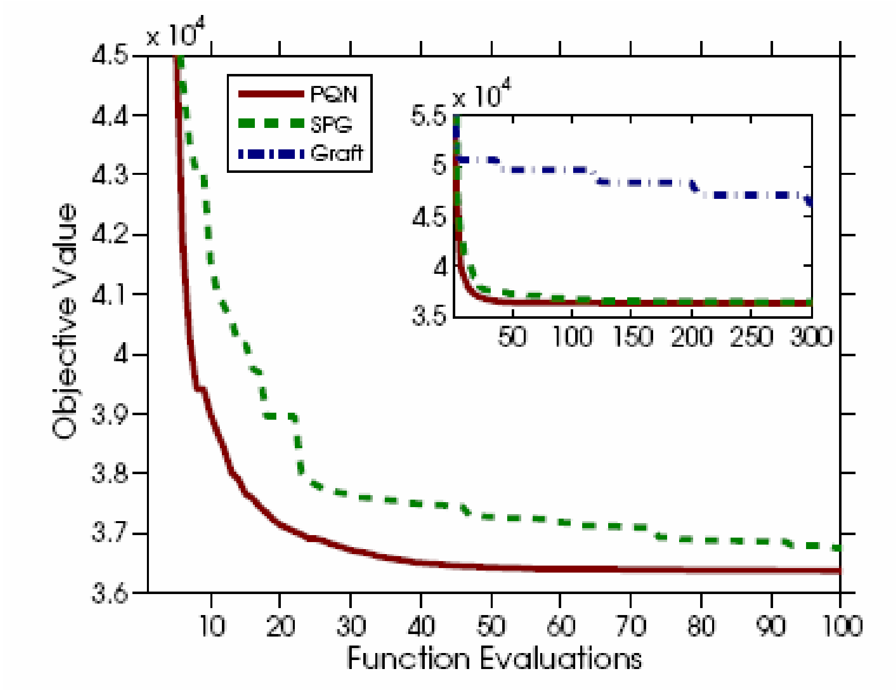


$$\begin{aligned} \mathbf{x}_{k+1} &= t\Pi(\mathbf{x}_k - \beta_k \mathbf{g}_k) \\ \mathbf{g}_k &= \nabla f(\mathbf{x})|_{\mathbf{x}_k} \\ \beta_{k+1} &= \frac{(\mathbf{x}_k - \mathbf{x}_{k-1})^T(\mathbf{x}_k - \mathbf{x}_{k-1})}{(\mathbf{x}_k - \mathbf{x}_{k-1})^T(\mathbf{g}_k - \mathbf{g}_{k-1})} \end{aligned}$$

t chosen using non-monotone Armijo line search

Projected quasi Newton

- Use LBFGS in outer loop to create a constrained quadratic approximation to objective
- Use spectral projected gradient in inner loop to solve subproblem



"Optimizing Costly Functions with Simple Constraints:
A Limited-Memory Projected Quasi-Newton Algorithm",
Mark Schmidt, Ewout van den Berg, Michael P. Friedlander, and Kevin Murphy,
AI/Stats 2009

Experiments

- We compared classification accuracy on synthetic 10-node CRF and real 16-node CRF.
- For each node, we compute the max of marginal using exact inference

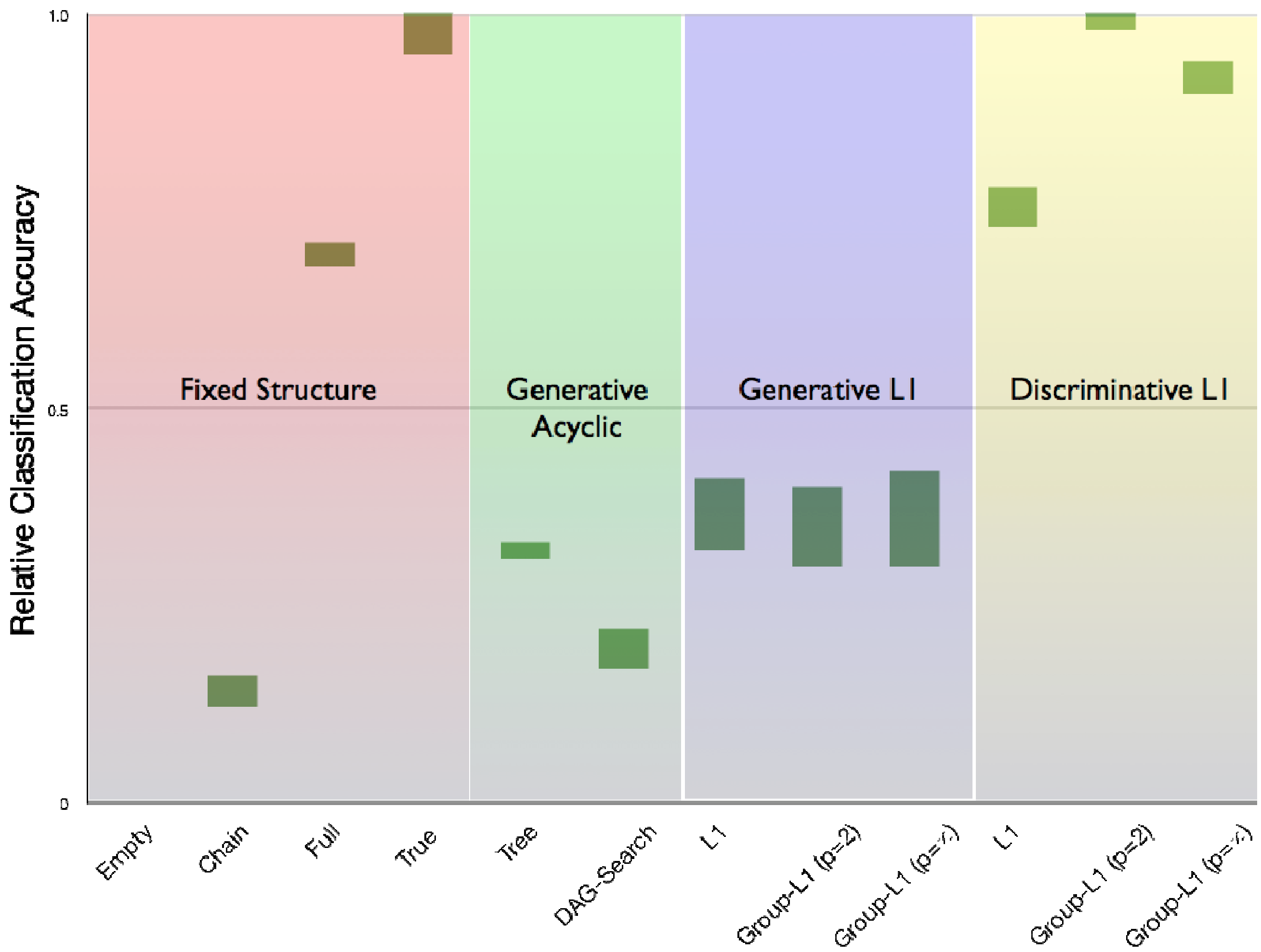
$$\hat{y}_j = \arg\max p(y_j|\mathbf{x}, \mathbf{w}, G)$$

- First learn (or fix) G, then learn w given G
  - Empty, chain, full, true
  - Best DAG (greedy search), best tree (Chow-Liu)
  - max p(y|w) $||w||_1$, $||w||_2$, $||w||_\infty$
- Jointly learn G and w
  - Max p(y|x,w,v) $||w||_1$, $||w||_2$, $||w||_\infty$

Results on synthetic data

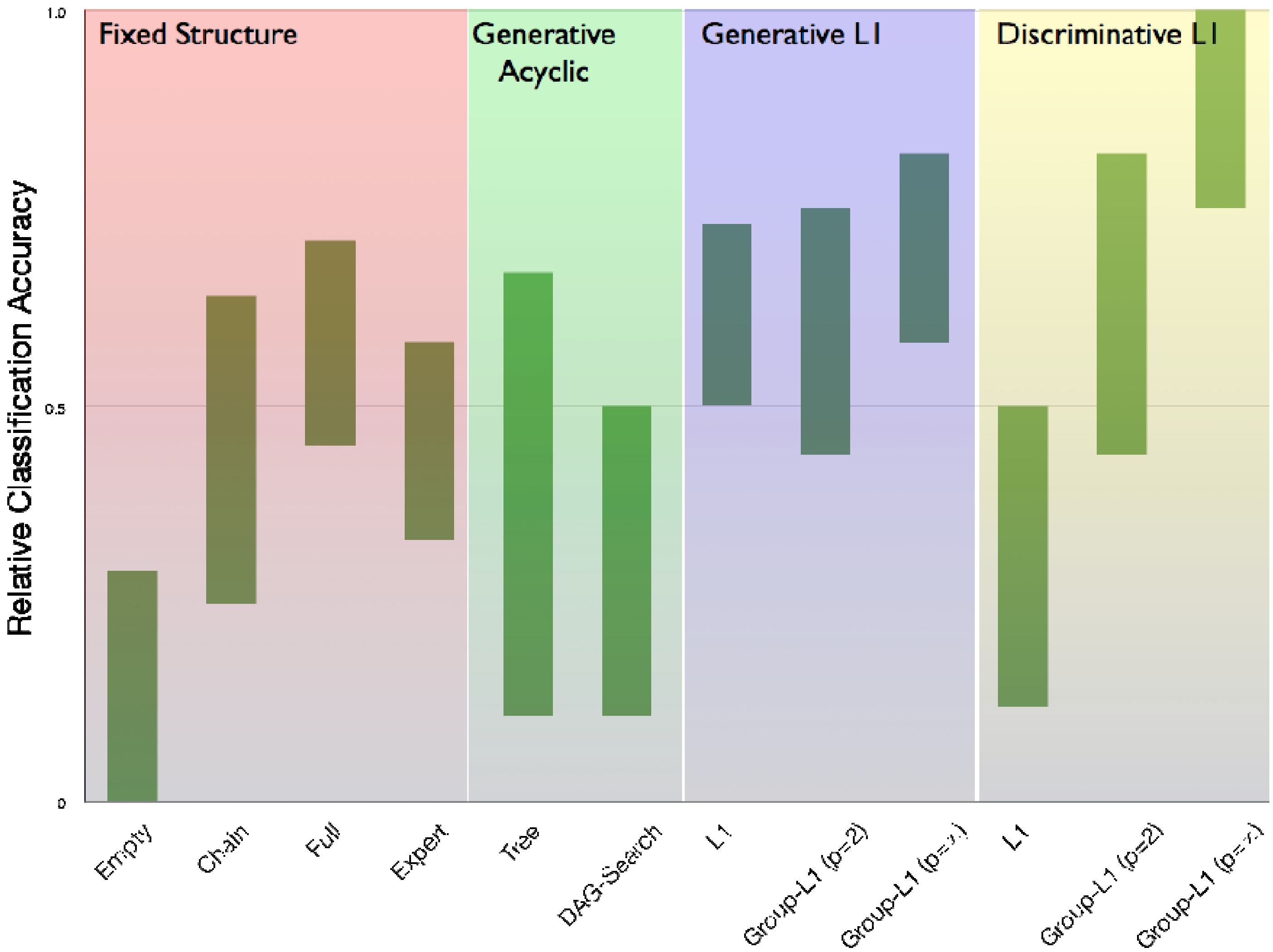
- d=10, n=500 train, 1000 test

90% confidence interval derived from 10 random trials



Results on heart data

90% confidence interval derived from 10-fold cross validation



Incremental feature addition

- Lee, Ganapathi & Koller compute gradient and expectations using LBP instead of PL
- They greedily add features according to their expected gain (change in penalized loglik)
- Initially the graph is sparse so LBP is accurate, but degrades over time

Della Pietra

Can use Gibbs sampling + IS corrections
Della Pietra, Della Pietra, Lafferty, PAMI
1997

```
m, r, xevo, ijjiir, b, to, jz, gsr, wq, vf, x, ga,
msmGh, pcp, d, oziVlal, hzagh, yzop, io, advzmxnv,
ijv_bolft, x, emx, kayerf, mlj, rawzyb, jp, ag,
ctdnnnbg, wgdw, t, kguv, cy, spxcq, uzflbbf,
dxtkkn, cxwx, jpd, ztzh, lv, zhpkvnu, l^, r, qee,
nynrx, atze4n, ik, se, w, lrh, hp+, yrqyka'h,
zcngotcnx, igcump, zjcjs, lqpWiqu, cefmfhc, o, lb,
fdcY, tzby, yopxmvk, by, fz,, t, govyccm,
ijyiduwfzo, 6xr, duh, ejv, pk, pjw, l, fl, w
```

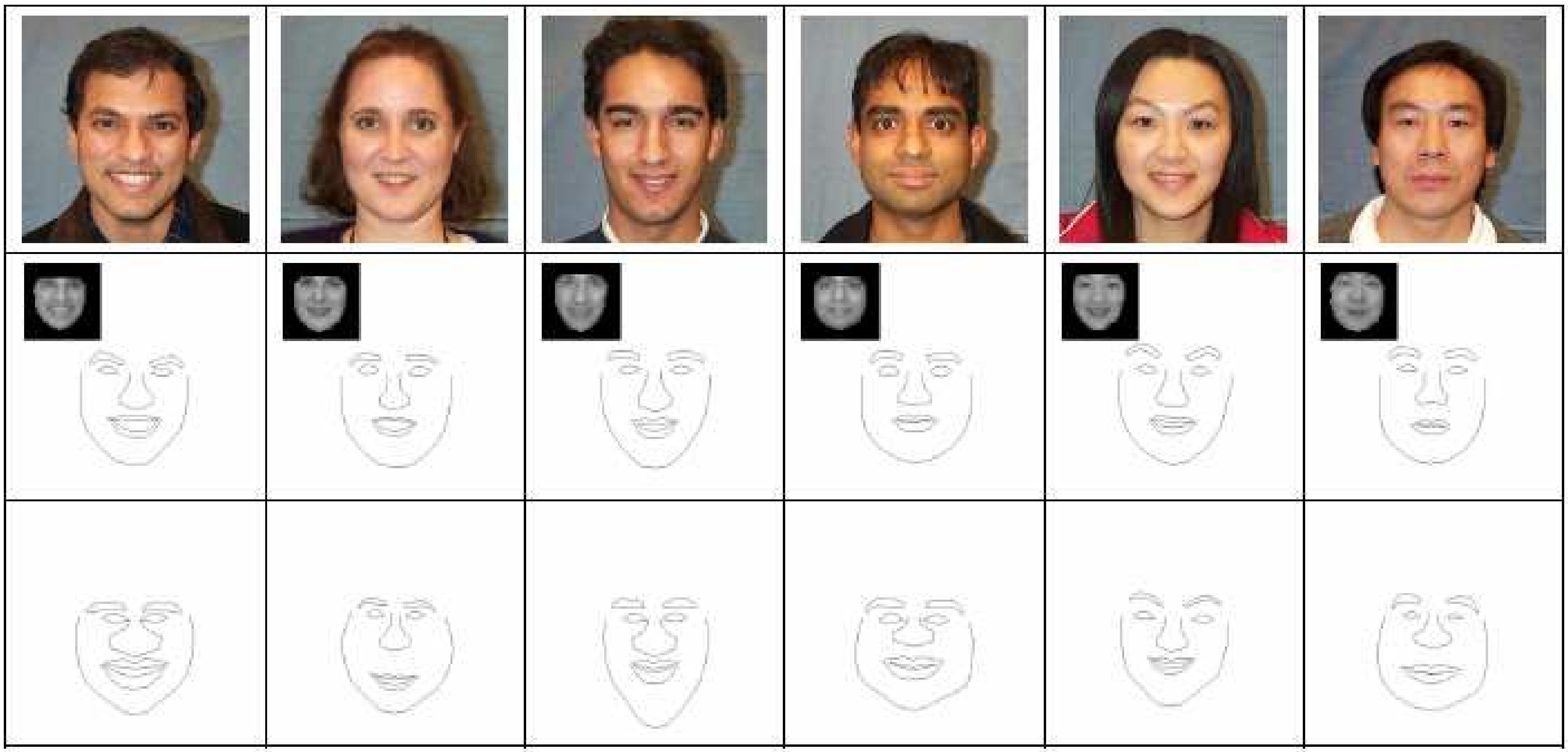
The second most important feature, according to the algorithm, i
that two adjacent lower-case characters are extremely common
The second-order field now becomes

$$p(\omega) = \frac{1}{Z} e^{\sum_{i \sim j} \lambda_{[a-z][a-z]} \chi_{[a-z][a-z]}(\omega_{ij}) + \sum_i \lambda_{[a-z]} \chi_{[a-z]}(\omega_i)}$$

The first 1000 features that the algorithm induces include the strings s>, <re, ly>, and ing>, where the character “<” denotes beginning-of-string and the character “>” denotes end-of-string. In addition, the first 1000 features include the regular expressions [0-9] [0-9] (with weight 9.15) and [a-z] [A-Z] (with weight −5.81) in addition to the first two features [a-z] and [a-z] [a-z]. A set of strings obtained by Gibbs sampling from the resulting field is shown here:

```
was, reaser, in, there, to, will, ,, was, by,
homes, thing, be, reloverated, ther, which,
conists, at, fores, anditing, with, Mr., proveral,
the, ,, ***, on't, prolling, prothere, ,, mento,
at, yaou, 1, chestraing, for, have, to, intrally,
of, qut, ., best, compers, ***, cluseliment, uster,
of, is, deveral, this, thise, of, offect, inatever,
thifer, constranded, stater, vill, in, thase, in,
youse, menttering, and, ., of, in, verate, of, to
```

Maxent models of faces



Use importance sampling to reweight the Gibbs samples when evaluating feature gain

C. Liu and S.C. Zhu and H.Y. Shum, ICCV 2001