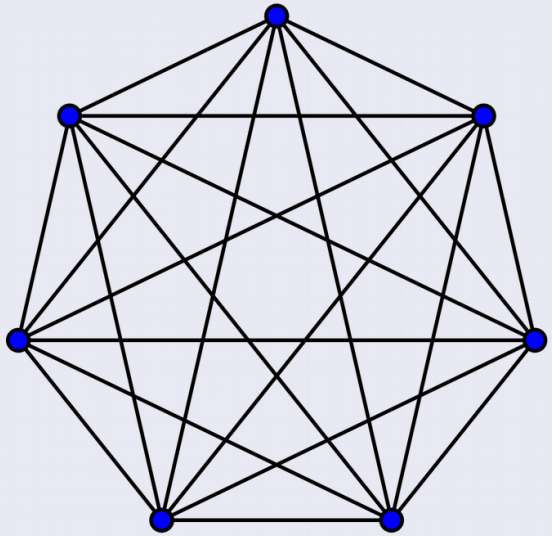
Learning Graph Models from Data (a sketch)

Mircea Andrecut

mircea.andrecut@gmail.com

Contents



General formulation

You are given a "measurement" matrix of m observations and n variables:

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix}$$

Question: Is there a way to infer if variable i causes the behavior of variable j, using only this matrix of measurements?

Network Inference Problem

Goal: identify the links (direction) between nodes.

Data size: $\sim mn$

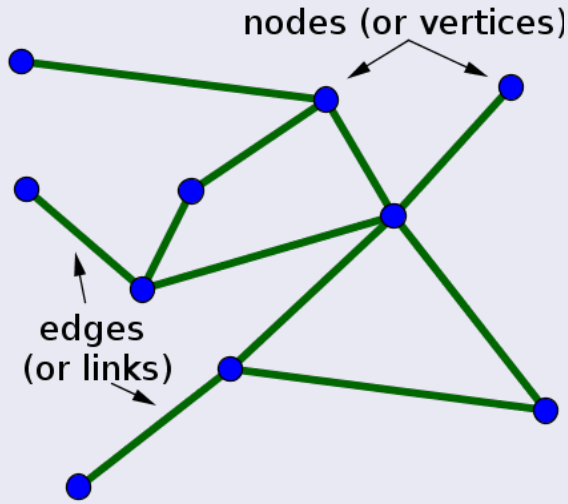
m observations of n variables

Data per unknown: $\alpha \sim mn/n^2 = m/n$

The situation is even worse if we consider all possible interactions:

$\alpha = m/2^n \ll 1$ - Boolean data

$\alpha = m/2^{n^S} \ll 1$ - Continuous data (S entropy)



The problem is *ill conditioned*, since the number of observations is much smaller than the number of possible logic rules describing the interactions.

The *inductive inferences* extracted from such complex interacting networks are only *probable*, given the incomplete data observations:

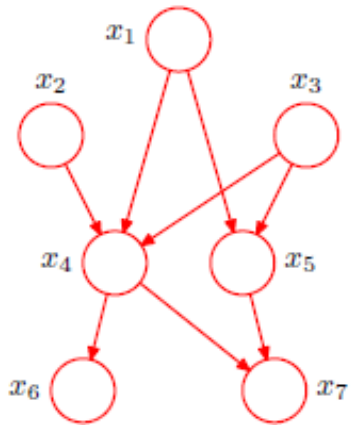
$$P(x) \propto \exp\left[-\sum_j \alpha_j(x_j) - \sum_{i,j} \beta_{ij}(x_i, x_j) - \sum_{ijk} \gamma_{ijk}(x_i, x_j, x_k) - \ldots\right]$$

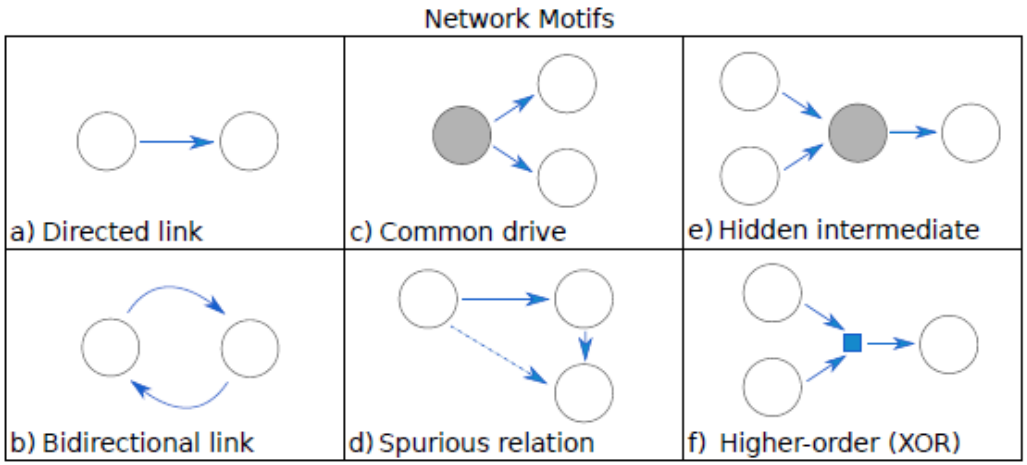
Granger causality

A variable x *Granger causes* variable y if knowledge of past values of both x and y reduces the variance of the prediction error for y.

It can be used to infer directed links if:

1. The cause x occurs before the effect y.
2. The causal series x_t contains unique information about the effect series y_t, that is not available in any other series.





Can we unravel the causality in the network?

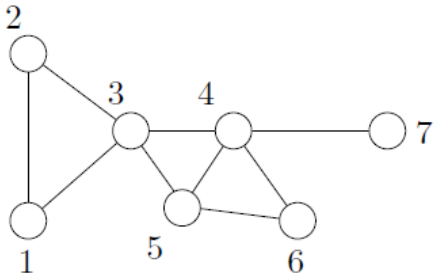
Definitions

Graph = vertex set ($V = \{1, 2, ..., n\}$) + edges ($E \subset V \times V$)

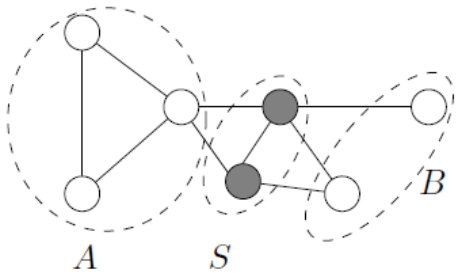
Undirected graphs: no distinction between $(s, t) \in E$ and $(t, s) \in E$.

Directed graphs: distinction between $(s, t) \in E$ and $(t, s) \in E$.

Directed acyclic graphs (DAG): directed graphs without cycles.



Maximal cliques $(123), (345), (456), (47)$



Vertex cutset S

Graph clique: $C \subseteq V \Leftrightarrow (s, t) \in E, \forall s, t \in C$, a fully-connected subset of V.

Maximal clique: it is not strictly contained within any other clique.

The set of all cliques in G: $\mathscr{C}$

Vertex cutset: a subset $S \subset V$ whose removal breaks the graph into pieces.

Factorization and Markov properties

Associate random variables: $X = \{X_1, X_2, ..., X_n\}$ to the vertex set V.

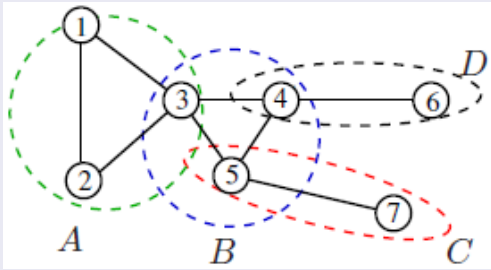
The graph G is a constraint for the random vector X.

Clique compatibility function: $\psi_C(x_C)$, $x_C = \{x_s | s \in C\}$

Factorization property:

A probability distribution $P(X)$ factorizes over G if:

$$\boxed{P(x_1, ..., x_N) = \frac{1}{Z} \prod_{C \in \mathscr{C}} \psi_C(x_C)}$$

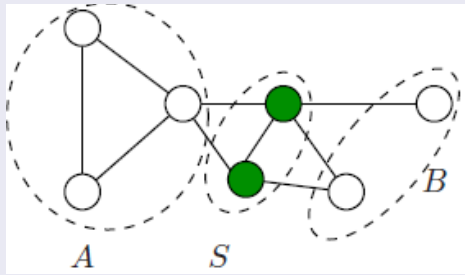


$$P \propto \psi_{123}(x_1, x_2, x_3)\psi_{345}(x_3, x_4, x_5)\psi_{46}(x_4, x_6)\psi_{57}(x_5, x_7)$$

Markov property:

X is a Markov vector w.r.t G if X_A and X_B are *conditionally independent*, when X_S separates A, B

$$X_A \perp\!\!\!\perp X_B \mid X_S \text{ for all cut sets } S \subset V$$

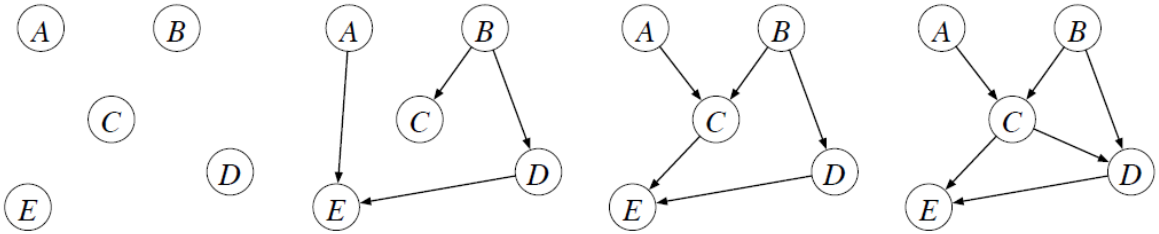


Theorem (Hammersley & Clifford)

The Markov property and the Factorization property are equivalent for $P > 0$.

Learning probabilistic graph models

How can we learn the interaction graph model from the given data set?



Optimization Problems

Constraint optimization (undirected graphs)

- Maximum entropy.
- Constraints: data properties (mean, correlations, marginals).

Best score (directed graphs)

- Bayesian networks.
- Uses a global score to find best graph that fits the data.

Conditional inference

- Neighborhood prediction.
- Constraints: LASSO.

Maximum entropy

Data: The set of random variables $X = \{X_1, ..., X_n\}$, $X \in \mathbb{R}^{m \times n}$.

Goal: Find the simplest probability model $P(X)$ consistent with the data.

Principle of maximum entropy: The probability distribution which best represents the current state of knowledge is the one with largest entropy.

Optimization problem:

$$\boxed{\max_{P(x)} \left\{ S(P(x)) = -\int_x P(x) \ln P(x) dx \right\}}$$

Constraints: $$f(P(x)) = \int_x P(x)dx = 1$$

$$g(x_i) = \int_x x_i P(x)dx = \frac{1}{m}\sum_{j=1}^{m} x_{ji} = \langle x_i \rangle$$

$$h(x_i, x_k) = \int_x x_i x_k P(x)dx = \frac{1}{m}\sum_{k=1}^{m} x_{ji}x_{jk} = \langle x_i x_k \rangle$$

Solution via Lagrange multipliers method:

$$\mathcal{L} = S(P(x)) + \alpha[f(P(x)-1)] + \sum_{i=1}^{n} \beta_i [g(x_i) - \langle x_i \rangle] + \sum_{i,j=1}^{n} \gamma_{ij} [h(x_i, x_j) - \langle x_i x_j \rangle]$$

$$\frac{\partial \mathcal{L}}{\partial P(x)} = 0 \quad \Leftrightarrow \quad \ln(P(x)) = -1 + \alpha + \sum_{i=1}^{n} \beta_i g(x_i) + \sum_{i,j=1}^{n} \gamma_{ij} x_i x_j$$

$$\boxed{P(x) = Z^{-1} \exp[-\mathcal{H}(x, \beta, \gamma)]}$$

$$\mathcal{H}(x, \beta, \gamma) = -\sum_{i=1}^{n} \beta_i g(x_i) - \sum_{i,j=1}^{n} \gamma_{ij} x_i x_j, \quad Z = \exp(1-\alpha)$$

For integral convergence: $\sum_{i,j=1}^{n} \gamma_{ij} x_i x_j = X^T \Gamma X < 0$

The Lagrange coefficients can be found using the constraints:

$$\beta = \Theta\langle x\rangle, \quad \Gamma = -\frac{1}{2}\Theta$$

$\Theta = C^{-1}$ is the precision or concentration matrix

$C_{ij} = \langle x_i x_j\rangle - \langle x_i\rangle\langle x_j\rangle$ is the covariance matrix.

As a consequence, the maximum-entropy distribution is a multivariate Gaussian distribution, determined by $\langle x\rangle$ and Θ:

$$\boxed{P(x;\langle x\rangle, C) = (2\pi)^{-n/2}\det(C)^{-1/2}\exp\left[-\frac{1}{2}(x-\langle x\rangle)^T\Theta(x-\langle x\rangle)\right]}$$

The interaction strength is:

$$\rho_{ij} \equiv \frac{\gamma_{ij}}{\sqrt{\gamma_{ii}\gamma_{jj}}} = \begin{cases} -\frac{\Theta_{ij}}{\sqrt{\Theta_{ii}\Theta_{jj}}} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}.$$

Problem

The covariance matrix C is always rank deficient ($N > M$) or ill conditioned, and thus not invertible $\Theta = C^{-1}$.

One can maximize the penalized Gaussian log likelihood:

$$\boxed{\Theta = \arg\max_{\Theta \succ 0} \left\{ L(P(x)) - \alpha \|\Theta\|_p^p \right\}}$$

where

$$\|\Theta\|_p^p = \sum_{i,j} |\Theta_{ij}|^p, \quad \alpha \geq 0, \quad p = 1, 2$$

$$L(P(x)) = \frac{1}{n} \ln P(x) = \ln \det(\Theta) - \text{trace}(C\Theta)$$

$$\ln \det(\Theta) = \begin{cases} \sum_{i=1}^{n} \ln \lambda_i(\Theta) & \text{if } \Theta \succ 0 \\ -\infty & \text{otherwise} \end{cases}$$

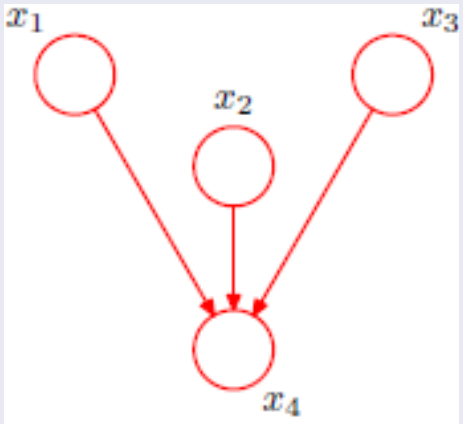
- $p = 2$ (ℓ_2 norm) $\rightarrow$ Ridge regression, dense graph models
- $p = 1$ (ℓ_1 norm) $\rightarrow$ LASSO regression, sparse graph models

Bayesian network

Is a graphical model for representing conditional dependencies between a set of random variables, the structure is a directed acyclic graph.

Edges start from the variables being conditioned on (parents) $Pa(X_i)$ and ending on the conditioned variables (children) X_i:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i))$$



The process of learning Bayesian networks from the data

- *Model selection:* given observed data find the best graph (model) G of relationships between the variables.
- *Parameter fitting:* given a graph G and observed data find the best conditional probabilities for each node.

From any given data most one can hope to learn is equivalence classes of networks, and not individual networks.

Directed graph models

Assume the random variables $X = \{X_1, ..., X_n\}$ with the joint probability distribution:

$$\boxed{P(X_1, ..., X_n; \Theta) = \prod_{i=1}^{n} P(X_i | Pa(X_i); \Theta_i)}$$

$$\boxed{\text{Model } (M) = \text{Graph } (G) + \text{Parameters } (\Theta)}$$

- $G \Leftrightarrow$ specify the parents: $Pa(X_i) = (X_1^{(i)}, ..., X_{K_i}^{(i)}) \subset X$
- $\Theta \Leftrightarrow$ specify the parameters: $\Theta_i = (\theta_0^{(i)}, \theta_1^{(i)}, ..., \theta_{K_i}^{(i)})$

For example, *linear models*:

$$X_i \sim \mathcal{N}(\bar{X}_i, \sigma_i^2), \quad \bar{X}_i \sim \theta_0^{(i)} + \sum_{j=1}^{K_i} \theta_j^{(i)} X_j^{(i)}$$

$\sigma_i, \theta_j^{(i)}$ best fit the data in the Maximum Likelihood Estimation sense.

Bayes' theorem

The posterior probability of the model $M(\Theta)$ given the data D:

$$\boxed{P(M|D) = \frac{P(D|M)P(M)}{P(D)}}$$

- $P(M)$ is the prior probability of M
- $P(D)$ is the probability of D
- $P(D|M)$ is the likelihood:

$$P(D|M) = \int P(D|M(\Theta))P(\Theta|M)d\Theta$$

Assuming that the parameters Θ_i are independent and the models M_i for all X_i are equally likely, the likelihood can be factorized as following:

$$P(D|M) = \prod_{i=1}^{n} \int P(D|M_i(\Theta_i))P(\Theta_i|M_i)d\Theta_i$$

Model selection process

- The space of all models is searched, and each model is scored.
- The model with the best score is the best model fitting the data.

Unsolved problem: there are no magic methods to find the best model.

Example of score functions

- Bayesian Information Criterion: $S_{BIC} = k \ln m - 2 \ln \hat{L}$
- Akaike Information Criterion: $S_{AIC} = 2k - 2 \ln \hat{L}$

k the number of free parameters in the model.
m the number of samples.
$\hat{L}$ is the maximum value of the likelihood function of the model.

For the standard model we have: $\ln \hat{L} = -\frac{m}{2} \ln \sigma^2$, $\sigma^2 = \frac{1}{m} \|X_i - \hat{X}_i\|^2$

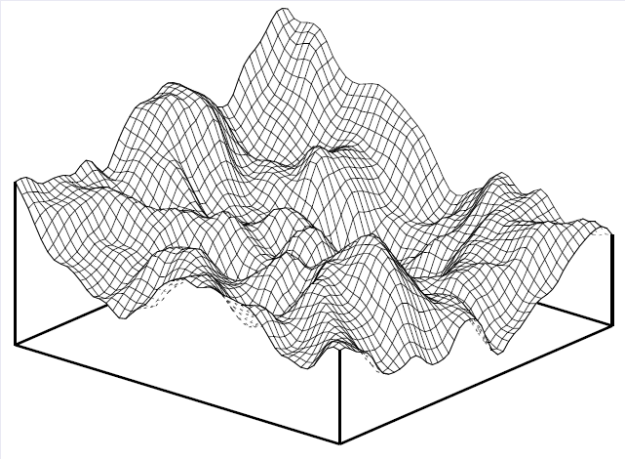
The model with the lowest score $S(\hat{\sigma}^2, K)$ is preferred (parsimony principle)

The score for whole network: $S_{tot}(M) = \sum_{i=1}^{n} S(M_i)$

Greedy search algorithm

1. Start with model M.
2. Change the model (edge deletions, additions, reversals): $M \rightarrow M'$
3. If $score(M') < score(M)$ then accept M'.
4. Repeat.

Finding the model with best score requires global maximum of the likelihood landscape, which is unfeasible since it requires the exhaustive enumeration of all possible graphs.



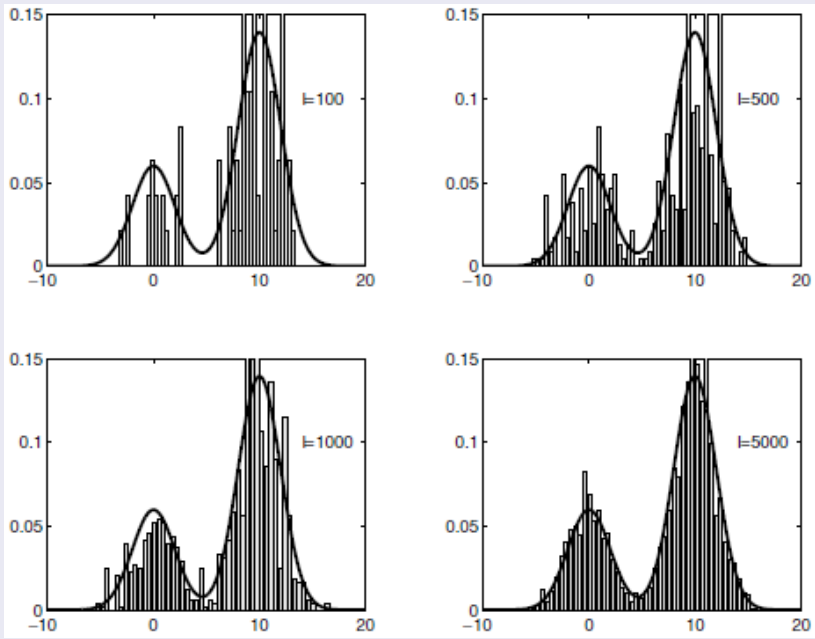
Better solution

- Markov Chain Monte-Carlo to generate samples.
- Simulated Annealing to find the optimum score.

Metropolis-Hastings algorithm

Markov process with a desired stationary distribution $\pi(M) = P(M)$, based on the detailed balance condition: $P(M'|M)P(M) = P(M|M')P(M')$.

Initialize M
for $j = 1 : J$ **do**
  Sample $r \sim \mathcal{U}_{[0,1]}$
  Sample $M' \sim P(M'|M)$
  $\mathcal{A} \leftarrow \min\left\{1, \frac{P(M')P(M|M')}{P(M)P(M'|M)}\right\}$
  if $\mathcal{A} > r$ **then**
    $M \leftarrow M'$
  end if
end for

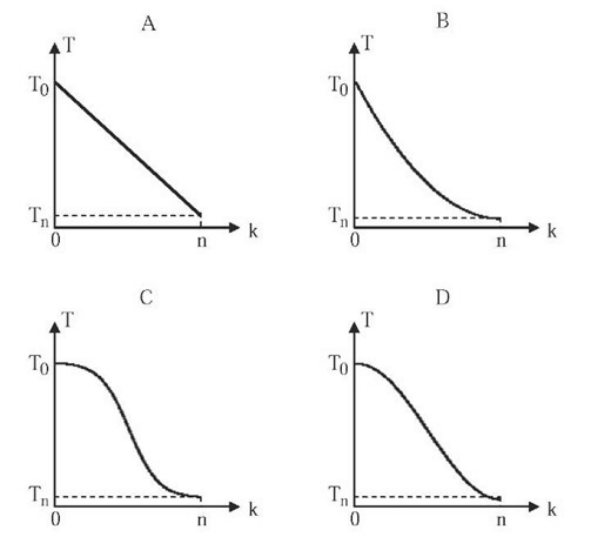


Needs to specify the *source* $P(M'|M)$ and the *target* $P(M)$ distributions.

Simulated Annealing algorithm

Is a metaheuristic modification of MCMC for global optimization:

Initialize M
for $j = 0 : J_{max} - 1$ **do**
  $T \leftarrow temperature(j/J_{max})$ // cooling $\rightarrow$
  Sample $r \sim \mathcal{U}_{[0,1]}$
  Sample $M' \sim P(M'|M)$
  if $P(S(M), S(M'), T) > r$ **then**
    $M \leftarrow M'$
  end if
end for



Acceptance probability:

$$P(S(M), S(M'), T) = \begin{cases} 1 & \text{if } S(M') < S(M) \\ \exp(-(S(M') - S(M))/T) & \text{otherwise} \end{cases}$$

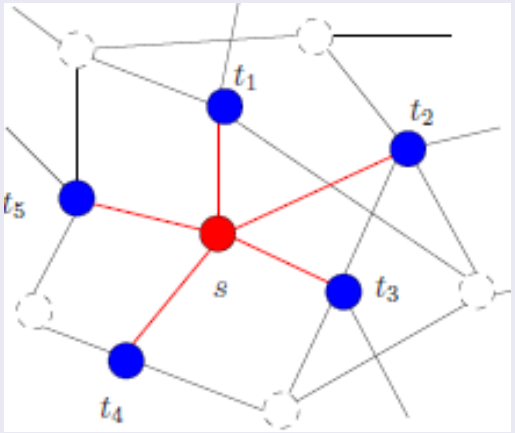
$S(M)$ is the score value.

Markov property

Assume $X = \{X_1, ..., X_n\}$, $X \in \mathbb{R}^{m \times n}$ is a Markov vector.

Define: $X_{\setminus\{s\}} = \{X_t | i \in V_{\setminus s}\} \in \mathbb{R}^{m \times (n-1)}$

The relevant variables to this conditioning are in the neighborhood set: $\mathcal{N}(s) = \{t \in V | (s,t) \in E\}$



$\mathcal{N}(s)$ is the cut set that separates s.

X_s is conditionally independent of $X_{V \setminus \{\mathcal{N}(s) \cup \{s\}\}}$ given $\mathcal{N}(s)$

If we consider the problem of predicting X_s based on $X_{V \setminus \{s\}}$, then the best predictors are a function of only $X_{\mathcal{N}(s)}$:

$$X_s = X_{\setminus\{s\}}^T \beta_s + \varepsilon_s$$

The dependence is captured by β_s, which is a scalar multiple of the corresponding line in the precision matrix $\Theta = C^{-1}$.

Algorithm for graph selection via conditional inference

For each vertex $s = 1, ..., n$:

1. Solve the neighborhood prediction problem using LASSO:

$$\hat{\beta}^{(s)} = \arg\min_{\beta_{(s)}} \left\{ \frac{1}{2} \|x_s - X_{\setminus\{s\}}^T \beta_s\|_2^2 + \lambda \|\beta_s\|_1 \right\}$$

2. Compute the estimate of the neighborhood set $\mathcal{N}(s)$:

$$\hat{\mathcal{N}}(s) = \text{supp}(\hat{\beta}_s)$$

3. Combine the estimates $\hat{\mathcal{N}}(s) | s \in V$ via AND / OR rules to form the graph structure $G = (V, E)$.

AND rule: $(s, t) \in E$ iff $s \in \hat{\mathcal{N}}(t)$ and $t \in \hat{\mathcal{N}}(s)$.

OR rule: $(s, t) \in E$ iff $s \in \hat{\mathcal{N}}(t)$ or $t \in \hat{\mathcal{N}}(s)$.

General linear models

The models are linear in the parameters:

$$y_j = \beta_1 x_{j1} + \beta_2 x_{j1}^2 + \beta_3 x_{j1} x_{j2} + ... + \varepsilon_j, \quad \varepsilon_j \sim \mathcal{N}(0, \sigma^2)$$

Situations where general linear models are not appropriate:

- The range of Y is restricted (e.g. binary, count)
- The variance of Y depends on the mean

Generalized linear models (GLM)

GLMs address both of these issues:

$$\boxed{\eta_j = \beta_1 x_{j1} + ... + \beta_n x_{jn} + ... + \varepsilon_j = \beta^T x_j + \varepsilon_j}$$

by defining the functions:

$$\boxed{\eta_j = g(\mu_j), \quad var(Y_j) = V(\mu_j)}$$

where: $E[Y_j] = \mu_j$, $var(Y_j) = \sigma^2$, and $x_{j0} = 1, j = 1, ..., m$

Standard model

$$Y \in \mathcal{N}(\mu, \sigma^2), \quad g(\mu) = \mu = \beta^T x$$

Logistic model

$$Y \in \{0, 1\}, \quad \mu(x) = \Pr[Y = 1 | X = x]$$

$$g(\mu) = \ln \frac{\mu}{1-\mu} = \ln \frac{\Pr[Y = 1 | X = x]}{\Pr[Y = 0 | X = x]} = \beta^T x$$

$$\mu(x) = \Pr[Y = 1 | X = x] = \frac{\exp(\beta^T x)}{1 + \exp(\beta^T x)}$$

Multi-class logistic model

$$Y \in \{0, 1, ..., K-1\}, \quad \mu_k(x) = \Pr[Y = k | X = x]$$

$$\mu_k(x) = \Pr[Y = k | X = x] = \frac{\exp(\beta_k^T x)}{\sum_{k=0}^{K-1} \exp(\beta_k^T x)}$$

Standard model

Both Ridge and LASSO penalties (elastic net):

$$\min_{\beta} \left\{ \frac{1}{2}\|y - X^T\beta\|_2^2 + \lambda \left[\frac{1}{2}(1-\alpha)\|\beta\|_2^2 + \alpha\|\beta\|_1 \right] \right\}$$

Relative strength of each penalty: $\lambda, \alpha \in [0, 1]$

Algorithms

Statistics and machine learning:

- Coordinate descent
- Least angle regression
- Alternating direction
- Minorization-maximization

Signal processing:

- Gradient descent
- Basis pursuit
- Matching pursuit
- Orthogonal matching pursuit

Etc.

Least Angle Regression

LAR is a popular algorithm for ℓ_2 penalty (ridge regression):

1. Standardize the predictors to have mean zero and unit norm.
2. Start with $\beta_i = 0, i = 1, ..., n$, and residual $r = y$.
3. Find the predictor x_i most correlated with r: $\max_i r^T x_i$.
4. Move β_i in the direction of the sign of its correlation $r^T x_i$, and take the residuals $r = y - \hat{y}$. Stop when some other predictor x_k has as much correlation with r as x_i has.
5. Move both (β_i, β_j) in their joint least squares direction, until some other predictor x_ℓ has as much correlation with the residual r.
6. Continue until all predictors are entered in the model.

Additional step for ℓ_1 penalty (LASSO regression):

6. If a non-zero coefficient β_i hits zero, drop its variable from the active set and recompute the current joint least squares direction.

Coordinate descent - LASSO

Very fast algorithm for LASSO penalty:

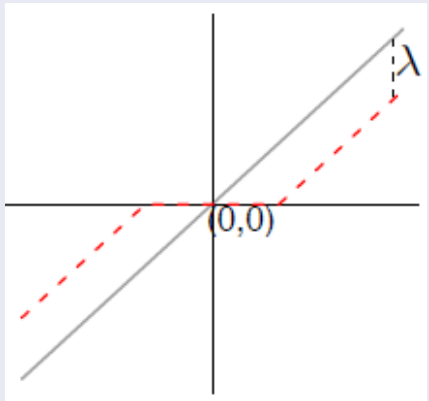
$$\min_{\beta} \left\{ \frac{1}{2} \|y - X^T \beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

Optimizes each parameter separately, holding the others fixed:

$$\beta_i^{t+1} = \arg\min_{\beta_i^t} R(\beta_1^t, ..., \beta_{i-1}^t \beta_i^t \beta_{i+1}^t, ..., \beta_n^t)$$

A cycle of univariate Lasso problems, $i = 0, 1, ..., n$:

- Compute the partial residuals:
  $r_{ji} = y_j - \sum_{k \neq i} x_{jk} \beta_k$
- Compute the coefficient on k predictor:
  $\beta_i^* = \frac{1}{m} \sum_{j=1}^m x_{ji} r_{ji}$
- Update β_k using the soft-thresholding operator:
  $\beta_i \leftarrow S(\beta_i^*, \lambda), \quad S(\beta_i^*, \lambda) = \text{sign}(\beta_i^*)(|\beta_i^*| - \lambda)_+$



Coordinate descent - LASSO + Ridge

For cases where predictors are correlated in groups.

The total penalty function is:

$$P(\alpha, \beta) = \sum_{i=1}^{n} \left[\frac{1}{2}(1-\alpha)\beta_i^2 + \alpha|\beta_i| \right]$$

α creates a compromise between LASSO and Ridge.

The update of β_i changes as following:

$$\beta_i^* = \frac{1}{m} \sum_{j=1}^{m} x_{ji} r_{ji}$$

$$\beta_i \leftarrow \frac{S(\beta_i^*, \lambda\alpha)}{1 + \lambda(1-\alpha)}$$

$$S(\beta_i^*, \lambda\alpha) = \text{sign}(\beta_i^*)(|\beta_i^*| - \lambda\alpha)_+$$

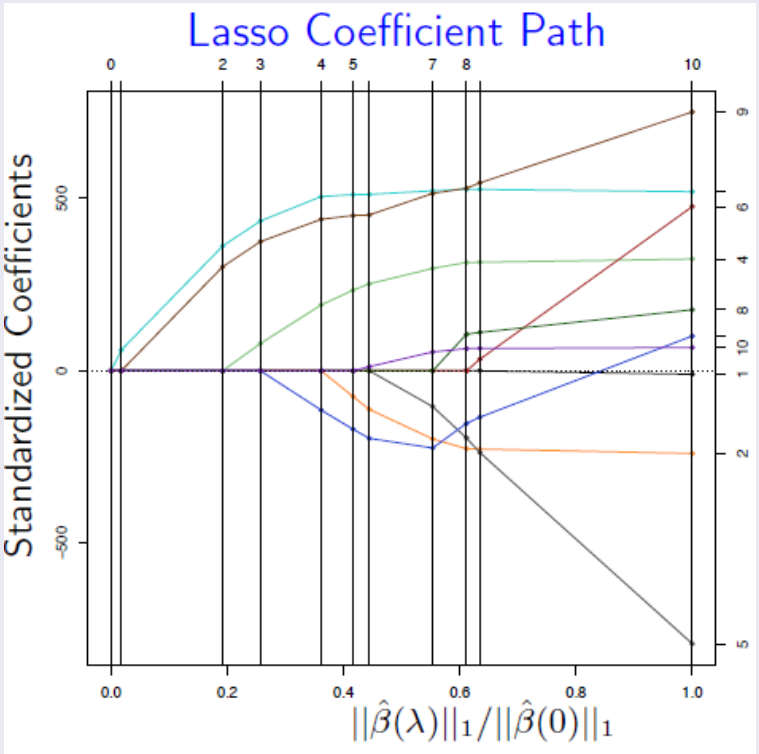
The rest is the same as for LASSO.

Path algorithms

CD and LAR are path algorithms:

$$\hat{\beta}(\lambda) = \arg\min_{\beta_0,\beta} \left\{ \frac{1}{2}\|y - X^T\beta\|_2^2 + \lambda\|\beta\|_1 \right\}$$

One can estimate $\hat{\beta}(\lambda)$ on a grid of λ values, from λ_{max} to λ_{min}.



Logistic regression

Requires the maximization of the likelihood:

$$\max_{\beta} \left\{ L(\beta, y, X) = \prod_{j=1}^{m} \Pr[Y = y_j | X = x_j] \right\}$$

or equivalently the minimization of the negative log-likelihood:

$$\min_{\beta} \{\mathcal{L}(\beta, y, X) = -\ln L(\beta, y, X)\}$$

$$\mathcal{L}(\beta, y, X) = -\sum_{j} \{y_j \ln p(x_j; \beta) + (1 - y_j) \ln(1 - p(x_j; \beta))\}$$

$$= -\sum_{j} \left\{ y_j (\beta^T x_j) - \ln(1 + e^{\beta^T x_j}) \right\}$$

$$p(x_j; \beta) = \frac{\exp(\beta^T x_j)}{1 + \exp(\beta^T x_j)}$$

Logistic regression - Newton algorithm

$$\beta^{new} = \beta^{old} - \left[\frac{\partial^2 \mathcal{L}(\beta)}{\partial\beta\partial\beta^T}\right]^{-1} \frac{\partial \mathcal{L}(\beta)}{\partial\beta}$$

$$\frac{\partial \mathcal{L}(\beta)}{\partial\beta} = -\sum_{j=1}^{m} x_m[y_j - p(x_j;\beta)]$$

$$\frac{\partial^2 \mathcal{L}(\beta)}{\partial\beta\partial\beta^T} = \sum_{j=1}^{m} x_j x_j^T p(x_j;\beta)[1 - p(x_j;\beta)]$$

Logistic regression with penalties

$$\boxed{\min_{\beta}\left\{\mathcal{L}(\beta, y, X) + \lambda\left[\frac{1}{2}(1-\alpha)\|\beta\|_2^2 + \alpha\|\beta\|_1\right]\right\}}$$

Use a combination of Newton outer loop with weighted soft-thresholding.