Gene Network Inference using Minimum Description Length Principle

ECEN 647 PROJECT

REPO LINK: HTTPS://GITHUB.COM/RKAPR/MDL

Overview

- Defining the problem
- Results
- Algorithm Details: NML, MDL, Kolmogorov Structure function
- Algorithm summary
- Limitations

Defining the problem

Gene @ t	Boolean Function (t-1)
Gene1	((!Gene2 & !Gene3) | Gene10)
Gene2	(!Gene9 & Gene6 & !Gene8)
Gene3	((Gene9 & Gene10) | (Gene1 & Gene10))
Gene4	(!Gene2 | (!Gene5 & !Gene6))
Gene5	((!Gene1 & Gene6) | (Gene1 & !Gene4 & !Gene6))
Gene6	((!Gene6 & !Gene10) | (Gene6 & !Gene1 & Gene10))
Gene7	(!Gene8 | (Gene6 & !Gene7))
Gene8	((!Gene4 & !Gene7) | (Gene4 & Gene1 & Gene7))
Gen9	((!Gene9 & !Gene6) | (!Gene2 & !Gene6) | (Gene9 & Gene6))
Gene10	((Gene4 & Gene5) | (Gene7 & Gene5) | (Gene7 & Gene4))

Results

Gene of Interest	Actual Predictors	Estimated Predictors Ts=10,Ns=10	Estimated Predictors Ts=10,Ns=100
Gene1	2,3,10	2,7	2,3,10
Gene2	6,8,9	NULL	NULL
Gene3	1,9,10	10	9,10
Gene4	2,5,6	2	2
Gene5	1,4,6	1,2,4	1,4,6
Gene6	1,6,10	6,10	1,6,10
Gene7	6,7,8	8	2,8
Gene8	1,4,7	1,4,7	1,4,7
Gene9	2,6,9	2	2,6,9
Gene10	4,5,7	4,5,7	4,5,7

Algorithm Details: Notations

- Consider the network gene by gene. Gene $\mathrm{y_i}$ has regulator set $\lambda_i = \{\mathrm{i}_1, \dots, \mathrm{i}_{\mathrm{k_i}}\}$ regulators, i ∈ {1,…,g} where g is the total number of genes in the network.
- $y_{i,t} = f_i(y_{i_1,t-1}, \dots, y_{i_{k_i},t-1})$ for t = {1,…,n}
- Since we are considering only one gene at a time, dropping the subscript i and writing in matrix form: $\mathrm{Y} = f(X) \oplus \varepsilon$ with $\mathrm{P}(\varepsilon_t = 1) = \theta$ denotes probability of binary error $\varepsilon \in \{0,1\}^n$.
- Assuming θ depends on the regressor vectors X, it can take one of 2^k possible values denoted by set Θ.
- The goal is to estimate the set of regulators λ_i for each gene. The functions f_i , and hence the network will be deterministic given the regulators and the expression patterns.
- Use MDL to estimate these λ_i using binomial regression.
- The model class is defined by $\mathcal{M} = \{P(y; f, x, \Theta) = \theta^{(1-y \oplus f(x))}(1-\theta)^{(y \oplus f(x))}\}$, where x is a row of X.

Algorithm Details: NML

- We look for a distribution q(Y) over all possible binary strings of length n such that the ideal codelength log(1/q(Y)) for a particular string is as close as possible to the ideal codelength $\log(1/P(Y; f, X, \Theta, \lambda))$ if we knew the parameters.
- Under NML we choose q which minimizes the difference between the two codelengths for the worst possible string Y:

$$q(Y) = min_q max_Y \frac{P(Y; f, X, \Theta, \lambda)}{q(Y)}$$

- This minimizing distribution is given by q(Y)= $P(Y; f, X, \Theta, \lambda)/C_n$ where

$$C_n = \sum_{k=0}^{n} \binom{n}{k} \left(\frac{k}{n}\right)^k \left(1 - \frac{k}{n}\right)^{n-k}$$

Algorithm Details: NML

- For Boolean model class under consideration, the maximized likelihood is given by

$$P(\mathbf{y};\lambda,\hat{f},\mathbf{X},\hat{\Theta}) = \prod_{l:\mathbf{b}_l\in\mathbf{X}} \left(\frac{m_{l_1}}{m_l}\right)^{m_{l_1}} \left(1-\frac{m_{l_1}}{m_l}\right)^{m_l-m_{l_1}},$$

- And the normalizing constant is given by

$$C_{m_l} = \sum_{i=0}^{m_l} \binom{m_l}{i} \left(\frac{i}{m_l}\right)^{i} \left(1-\frac{i}{m_l}\right)^{m_l-i}.$$

- So the universal NML model is

$$\hat{P}(\mathbf{y}) = \frac{P(\mathbf{y};\lambda,\hat{f},\mathbf{X},\hat{\Theta})}{\prod_{l:\mathbf{b}_l\in\mathbf{X}} C_{m_l}},$$

- And the codelength is given by :

$$-\log\hat{P}(\mathbf{y}) = \sum_{l:\mathbf{b}_l\in\mathbf{X}} \left[m_l h\left(\frac{m_{l_1}}{m_l}\right) + \log C_{m_l}\right],$$

- Here bl are the unique regressors in X, ml number of times bl appears in X and ml1 is number of times y(bl) is 1.

Algorithm Details: MDL

- Need to estimate model codelength as a measure of model complexity and residual codelength as a measure of data fitting.
- Model codelength is cost of specifying the regulator genes plus the cost of storing the probability table.
- Assuming g total genes

- log(g) bits are needed to store k, the number of regulators,
- To prefer codelength with smaller indegrees, the authors put upper limit on #bits to store k, with log(k + 1)+ log(1+ln(g)).
- Log(C(n,k)) bits to store index of set among all possible sets of size k

- Total model codelength is L = -log($\hat{P}(Y)$) + min{g, log(C(n,k))+log(k + 1)+ log(1+ln(g))}

Algorithm Details: Kolmogorov Structure Function

- To limit number of computations, the authors suggest using a computable form of Kolmogorov structure function with model length:

$$L_M(\mathbf{y},\lambda,d) = \sum_{l:\mathbf{b}_l\in\mathbf{X}} \log C_{m_l} + \frac{w}{2}\log\frac{w\pi}{2d} + L_\lambda,$$

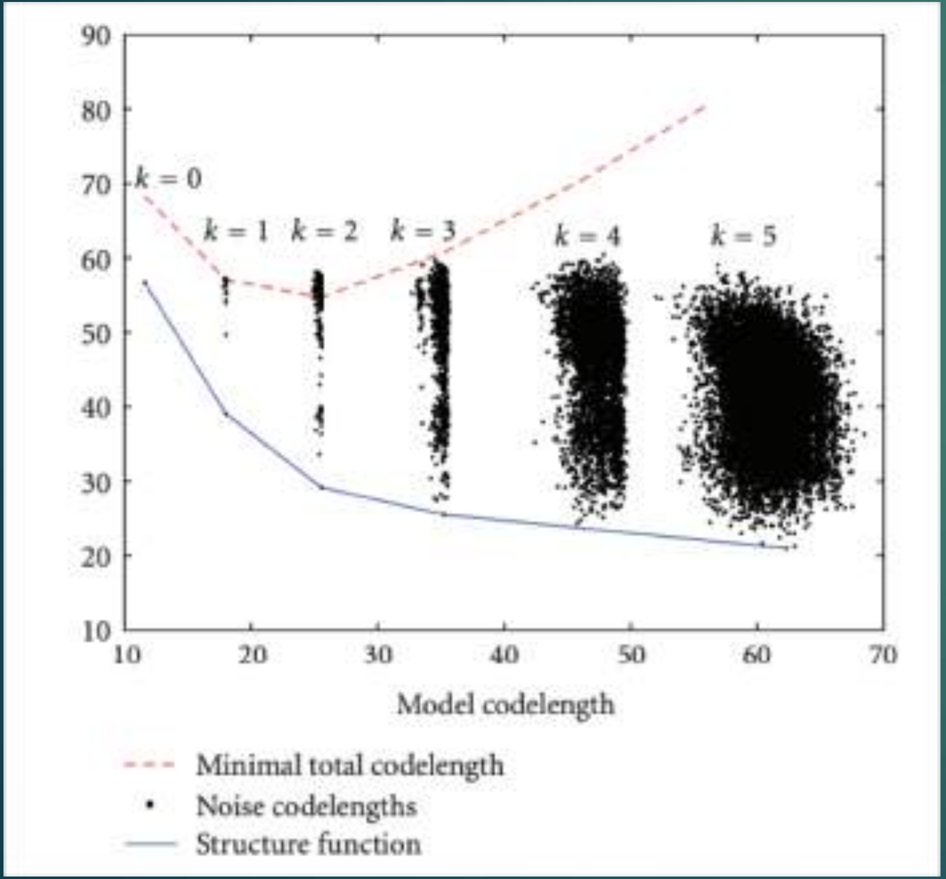
- And noise codelength:

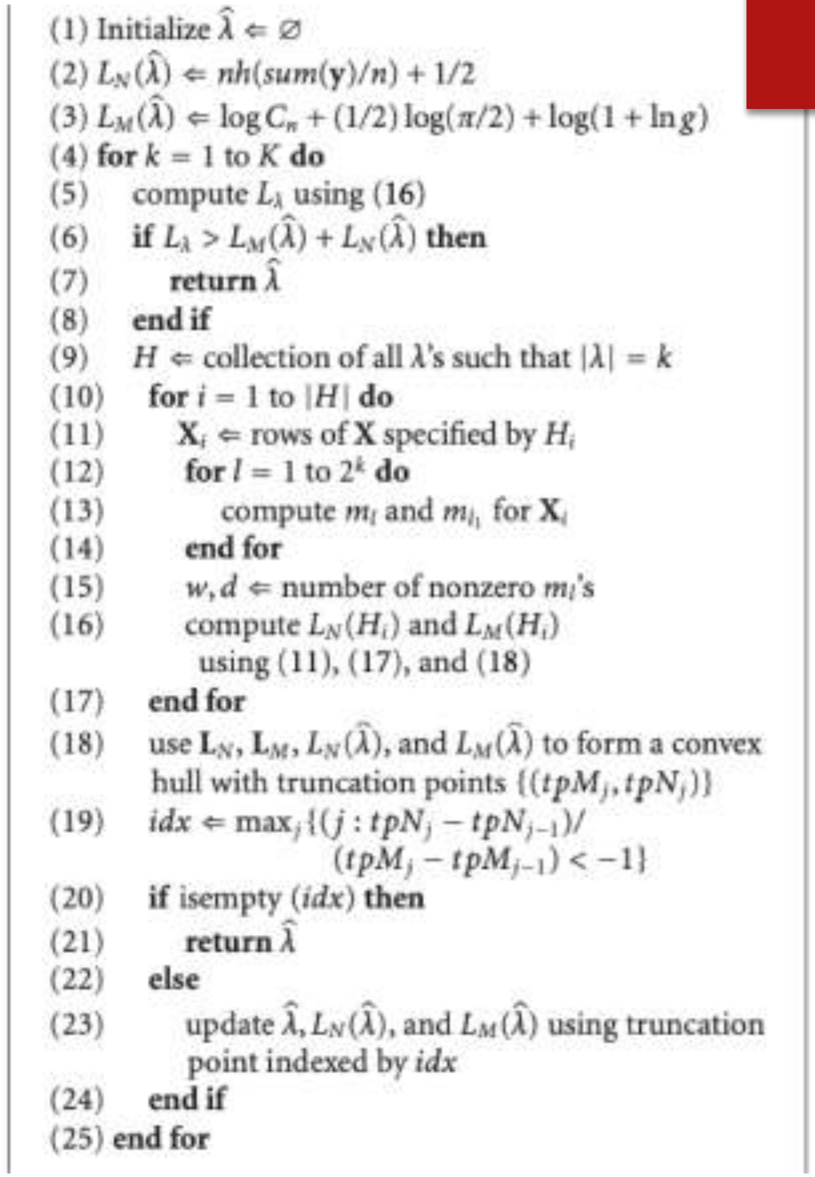
$$L_N(\mathbf{y},\lambda,d) = -\log P(\mathbf{y};\lambda,\hat{f},\mathbf{X},\hat{\Theta}) + \frac{d}{2}.$$

- Then the regulator set is selected as the point at which the slope of following function of model length drops below -1:

$$h_{\mathbf{y}}(\alpha) = \min_{\lambda,d}\{L_N(\mathbf{y},\lambda,d) : L_M(\mathbf{y},\lambda,d) \leq \alpha\}.$$

Algorithm Summary





```
(1) Initialize $\hat{\lambda} \Leftarrow \varnothing$
(2) $L_N(\hat{\lambda}) \Leftarrow nh(sum(\mathbf{y})/n) + 1/2$
(3) $L_M(\hat{\lambda}) \Leftarrow \log C_n + (1/2)\log(\pi/2) + \log(1 + \ln g)$
(4) for $k = 1$ to $K$ do
(5)     compute $L_{\lambda}$ using (16)
(6)     if $L_{\lambda} > L_M(\hat{\lambda}) + L_N(\hat{\lambda})$ then
(7)         return $\hat{\lambda}$
(8)     end if
(9)     $H \Leftarrow$ collection of all $\lambda$'s such that $|\lambda| = k$
(10)     for $i = 1$ to $|H|$ do
(11)         $\mathbf{X}_i \Leftarrow$ rows of $\mathbf{X}$ specified by $H_i$
(12)         for $l = 1$ to $2^k$ do
(13)             compute $m_l$ and $m_{l_1}$ for $\mathbf{X}_i$
(14)         end for
(15)         $w, d \Leftarrow$ number of nonzero $m_l$'s
(16)         compute $L_N(H_i)$ and $L_M(H_i)$
              using (11), (17), and (18)
(17)     end for
(18)     use $\mathbf{L}_N$, $\mathbf{L}_M$, $L_N(\hat{\lambda})$, and $L_M(\hat{\lambda})$ to form a convex
         hull with truncation points $\{(tpM_j, tpN_j)\}$
(19)     $idx \Leftarrow \max_j\{(j : tpN_j - tpN_{j-1})/$
                 $(tpM_j - tpM_{j-1}) < -1\}$
(20)     if isempty ($idx$) then
(21)         return $\hat{\lambda}$
(22)     else
(23)         update $\hat{\lambda}$, $L_N(\hat{\lambda})$, and $L_M(\hat{\lambda})$ using truncation
             point indexed by $idx$
(24)     end if
(25) end for
```

Limitations

- Need to pre-select genes of interest, can’t be used for more than tens of genes
- It is very difficult to observe all network state changes from real data, we don’t know the perfect sampling time point. Possible to miss fast state transitions and slower transitions may spread-out over multiple sample points. This error needs to be considered.
- Genes don’t update synchronously, possible to miss fast gene transitions and slower transitions may spread-out over multiple sample points. This may results in observing states which are not expected from the synchronous update of Boolean data.
- The algorithm doesn’t take into account the order of state changes, it estimates the network based on just the states values.

References

- Normalized Maximum Likelihood Models for Boolean Regression with Application to Prediction and Classification in Genomics,
Ioan Tabus, Jorma Rissanen, Jaakko Astola
- Inference of Gene Regulatory Networks Based on a Universal Minimum Description Length, John Dougherty, Ioan Tabus & Jaakko Astola