Chapter 6: Regulatory Networks

6.3 Boolean Network Models

Prof. Yechiam Yemini (YY)
Computer Science Department
Columbia University

Overview

- Boolean network models
- Sample applications
- Kaufmann’s theory of evolution
- Learning (reverse engineering) Boolean nets
Intro To Boolean Networks

Example: Thresholding Gene Expression

- Boolean model: discretize expressions to on/off model

![Diagram showing gene expression over time with on/off states for gene 1, gene 2, and gene 3.]

time (min)
Boolean Network Model

- A Boolean Network Model:
  - Nodes represent transcription factors
  - Edges represent regulatory input
  - Boolean gates (input functions) represent gene expression

\[
\begin{align*}
  f_A(A,B,C) &= A \text{ OR } C \\
  f_B(A,B,C) &= A \text{ AND } C \\
  f_C(A,B,C) &= \neg A \text{ OR } B
\end{align*}
\]

Dynamics

- Network State: \(X=(A,B,C,\ldots)\) is a Boolean vector
- State evolution: \(X(t+1)=f(X(t))=(f_A(X(t)), f_B(X(t)), \ldots)\)
  - E.g., \(X(t+1)=(\text{A OR C, A AND C, (NOT A) OR B})\)
  - \((0,1,1)\rightarrow(1,0,1)\)

This is discrete time synchronous dynamics
- State transitions occur through concurrent gates firings

Attractors
Noisy (Stochastic) Dynamics

- If gene-gates “fire” randomly
- The network becomes asynchronous
- The dynamics landscape changes

<table>
<thead>
<tr>
<th>X(t)</th>
<th>X(t+δ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>001</td>
</tr>
<tr>
<td>001</td>
<td>011</td>
</tr>
<tr>
<td>010</td>
<td>001,000,011</td>
</tr>
<tr>
<td>011</td>
<td>101,111,001</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>101</td>
<td>110,111,100</td>
</tr>
<tr>
<td>110</td>
<td>101,111,100</td>
</tr>
<tr>
<td>111</td>
<td>111</td>
</tr>
</tbody>
</table>

Example: Boolean Repressilator

- Repressilator has three repressors in a loop:
  - $f_A(A,B,C)=A \text{ OR } C$
  - $f_B(A,B,C)=A \text{ AND } C$
  - $f_C(A,B,C)=(\text{NOT } A) \text{ OR } B$

Stable attractor
More Generally

- **Boolean Network:**
  - A digraph $G=<V,E>$; nodes = genes, edges=regulation
  - For each node assign a Boolean function over ingress neighbors

- **Attractors & cycles describe dynamics of expression**

- **Learning (reverse engineering; identification):**
  - Extract Boolean network model from expression levels

---

**Example: Regulation of Drosophila Patterns**


- **Fundamental question:** how do gene’s regulate spatial patterns?
Segmentation Is Regulated By A Cascade

- Genes are activated in precise temporal order
- Use regulatory interaction to coordinate development functions

Segmentation Is Regulated By A Cascade

- Pair rule genes initiate stripes
  - eve, ftz...
- Segment polarity genes control anterior/posterior structure
  - Engrailed, wingless

- Giant (gt) & kruppel (kr) control stripe 2

- knirps (kni) & hunchback (hb) control stripes 3-7
A Boolean Network Model

\[ en_{i+1} = (W G_{i+1,1} \text{ or } W G_{i+1,2}) \text{ and not } S L P_i \]

\[ hh_{i+1} = E N_i \text{ and not } C I R_i \]

\[ p t c_{i+1} = C I A_i \text{ and not } E N_i \text{ and not } C I R_i \]

\[ c l_{i+1} = \text{not } E N_i \]

\[ E N_{i+1} = e n_i \]

\[ W G_{i+1} = w g_i \]

\[ C I R_{i+1} = c l_i \]

\[ H H_{i+1} = h h_i \]

\[ w g_{i+1} = (C I A_i \text{ and } S L P_i \text{ and not } C I R_i) \text{ or } \]

\[ \{w g_i \text{ and } (C I A_i \text{ or } S L P_i) \text{ and not } C I R_i \} \]

Either of the activators can counter mRNA decay.

\[ P T C_i^{+1} = p t c_i \text{ or } (P T C_i \text{ and not } H H_{i+1,1} \text{ and not } H H_{i+1,1}) \]

Free PTC does not decay.

Coordinated Regulation

Cadigan, Nussa, Genes & Development 11, 3256 (1997)
The Steady State (Attractor)

The model reproduces the wild type steady state

Compute Attractor Expression

Possible stable patterns

- wild type
- broad
- lethal
- double wg
- displaced
- displaced, 2 wg

The latter states have very small probability.
Mutations (Perturbations)

*Kauffman’s Model*

*wg, en or hh mutations are lethal*

*ptc mutation broadens the stripes*

The *wg, en and hh* stripes broaden, regardless of initial state.
Kauffman’s Model [60’s, 93]

- Study Boolean networks to describe evolution
- BN: a graph of “genes” each with a random Boolean function
  - N=# of nodes; k=connectivity
- BN traverses trajectories over the hypercube \([0,1]^n\)
- Converges to best fit response to random inputs

- Trajectories: series of state transitions
- Attractors: repeating trajectories
- Basin of Attraction: all states leading to an attractor

Evolution of Boolean Networks

- Nature evolves an ensemble of networks
  - Mutations change connectivity/gene-transition-function
- Genes select best-fitness transition functions
- What happens if k is large (e.g., k=N-1)?
  - \(X(t+1)\) is uncorrelated with \(X(t)\)
  - The number of attractors is very small; cycles are huge with period of some \(2^{0.5N}\)
  - Most genes would be oscillating
  - Network is very sensitive to small perturbations
- Need to keep k small
  - K=1: too small; gene’s do not interact
  - K=2: large number of attractors \(\sim N^{0.5}\); avg cycle \(N^{0.5}\)
Learning Boolean Nets
(“Reverse Engineering” “Identification”)

The Challenge

- Discretization: Given expression profiles vector X(t)
  - Set expression thresholds \( \tau \) (how?)
  - Extract a time-state map \( S(t) \); compute state transitions map \( M(x) \)

- Learning:
  - Given: state transition map \( M \)
  - Compute: a Boolean vector function \( f \) such that \( M(x) = f(x) \)
Akutsu Algorithm (99)

- Brute force search for $f$
- Fix $k$, and consider networks of max degree $k$
  - For each gene $i$, and for each subset of $k$ ingress genes find all functions $f_i$ that are compatible over this ingress set for all $\{S(r)\}$
  - i.e., $S'(r) = f_i(S'(r-1))$ where $S'$ is the restriction of $S$ to the ingress set
  - For $k$ fixed: $O(k^2 n^{k+1} m)$; if $k$ is not fixed, learning is NP complete.

Notes
- Works for small $k$...does not handle noise...
- Later improvements handle noise

Suppose the Network Graph is Known

- Given $M$ and $G$, computing $f$ is simple:
  - The truth table for $f_i(X)$ obtains by projecting $M$
  - The network graph $G$ guides the projections

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$f_1(X) = X_1 \text{ OR } X_2$
How Do We Find The Network Graph?

An intuitive approach:
- \( f(X_1, X_2, X_3) \) depends on \( X_i \) iff \( f(1, X_2, X_3) \neq f(0, X_2, X_3) \) for some \( (X_2, X_3) \)
- We call such values \( <X_i, X_2, X_3, f> \) “dependency”

An Intuitive Algorithm

- Repeat for all \( X_i \) and \( f_k \):
  - Scan \( M \) to find a dependency of \( f_k \) on \( X_i \); if found then add an \( X_i \to f_k \) edge to \( G \)
  - Else (no dependency found) then \( f_k \) is independent of \( X_i \)
REVEAL (98 Liang)

- Compute network graph from mutual information measure
- Base theory:
  - Let \( <X,Y> \) be an \(<input, output>\) stream
  - Consider \( H(Y) \), the entropy of \( Y \), and \( M(X,Y) \), the mutual information of \( X \) and \( Y \)
  - If \( M(Y,X) = H(Y) \) then \( X \) determines \( Y \) uniquely

\[
H(X) = -\sum p_i \log(p_i)
\]

- \( p_i \) is the probability that a random element of data stream \( X \) is \( i \)
- \( M(X, Y) = H(X) + H(Y) - H(X,Y) \)

REVEAL Algorithm

- Step 1: compute state transition \(<input, output>\) table

<table>
<thead>
<tr>
<th>Input stream</th>
<th>Output stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_{i+1} )</td>
<td>( A_i )</td>
</tr>
<tr>
<td>( B_{i+1} )</td>
<td>( B_i )</td>
</tr>
<tr>
<td>( C_{i+1} )</td>
<td>( C_i )</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 1 0</td>
</tr>
</tbody>
</table>
Step 2a: Compute Entropies

<table>
<thead>
<tr>
<th>Input stream value</th>
<th>Output stream value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_{i-1}</td>
<td>B_{i-1}</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ P(A_i = 0) = \frac{2}{8} = 0.25 \]
\[ P(A_i = 1) = \frac{6}{8} = 0.75 \]

H(A_i) = -((0.25)\log(0.25) + (0.75)\log(0.75)) = 0.81

Step 2a: Compute Entropies

note: \( \lim_{x \to 0} x \cdot \log(x) = 0 \), therefore in the left-hand limit, \((0)\log(0) = 0\).

H(A_i) = -((0.25)\log(0.25) + (0.75)\log(0.75)) = 0.81
H(B_i) = -((0.75)\log(0.75) + (0.25)\log(0.25)) = 0.81
H(C_i) = -((0.5)\log(0.5) + (0.5)\log(0.5)) = 1
H(A_{i-1}) = H(B_{i-1}) = H(C_{i-1}) = -((0.5)\log(0.5) + (0.5)\log(0.5)) = 1
H(A_{i-1}, C_{i-1}) = -((0.25)\log(0.25) + (0.25)\log(0.25) + (0.25)\log(0.25) + (0.25)\log(0.25)) = 2
H(C_i, A_{i-1}) = -((0.5)\log(0.5) + (0.5)\log(0.5)) = 1
H(A_i, A_{i-1}, C_{i-1}) = -((0.25)\log(0.25) + (0.25)\log(0.25) + (0.25)\log(0.25) + (0.25)\log(0.25)) = 2
H(B_i, A_{i-1}, C_{i-1}) = -((0.25)\log(0.25) + (0.25)\log(0.25) + (0.25)\log(0.25) + (0.25)\log(0.25)) = 2

......

......
Step 2b: Compute Network

- First compute mutual information

  (I) $M(A_i, [A_{i-1}, C_{i-1}]) = H(A_i) + H(A_{i-1}, C_{i-1}) - H(A_i, A_{i-1}, C_{i-1}) = 0.81 + 2 - 2 = 0.81$
  
  $= H(A_i)$, therefore $A_{i-1}$ and $C_{i-1}$ determine $A_i$

  (II) $M(B_i, [A_{i-1}, C_{i-1}]) = H(B_i) + H(A_{i-1}, C_{i-1}) - H(B_i, A_{i-1}, C_{i-1}) = 0.81 + 2 - 2 = 0.81$
  
  $= H(B_i)$, therefore $A_{i-1}$ and $C_{i-1}$ determine $B_i$

  (III) $M(C_i, A_{i-1}) = H(C_i) + H(A_{i-1}) - H(C_i, A_{i-1}) = 1 + 1 - 1 = 1$
  
  $= H(C_i)$, therefore $A_{i-1}$ determines $C_i$

- Use this to determine network graph

Step 3: Compute Boolean Functions

- Consider only network dependencies

<table>
<thead>
<tr>
<th>$A_{i-1}$</th>
<th>$C_{i-1}$</th>
<th>$A_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$A_{i-1}$</th>
<th>$C_{i-1}$</th>
<th>$B_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$A_{i-1}$</th>
<th>$C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$A_i = A_{i-1}$ OR $C_{i-1}$

$B_i = A_{i-1}$ AND $C_{i-1}$

$C_i = \neg A_{i-1}$
Consider the following expression scenario of 4 genes:

A threshold at 0.6 yields: 0010 → 1011 → 1001 → 1000 → 0100 → 0010
- Note: the intermediate state 0000 of the transition 1000→0000→0100 is ignored

The transition map is partial
- Can admit multiple Boolean net models
- Exercise: find 2 distinct Boolean net models

Exercise: use REVEAL to compute a network model

Computing A Partial Dependency Graph

Dependency means that $g_4$ must appear in any expression of $f_1$;
For a partial map $f_1$, may require other genes without depending on them.
Sensitivity Considerations & Noisy Maps

Consider again the 4 genes example

Different thresholds yield different Boolean dynamics:
- $Tr_{1/2/3/4}=0.6$: $0010 \rightarrow 1011 \rightarrow 1001 \rightarrow 1000 \rightarrow 0100 \rightarrow 0010 \ldots$
- $Tr_{1/2/3}=0.2$, $Tr_{4}=0.8$: $0010 \rightarrow 1010 \rightarrow 1011 \rightarrow 1000 \rightarrow 0110 \rightarrow 0100 \rightarrow 0110 \rightarrow 0010 \ldots$ (non-deterministic)
- $Tr_{1/2/3/4}=0.8$: $0010 \rightarrow 0011 \rightarrow 1000 \rightarrow 0000 \rightarrow 0100 \rightarrow 0000 \rightarrow 0010 \ldots$

Research Questions

- Extend the intuitive algorithm to handle partial noisy maps
- Extend REVEAL to handle partial noisy maps
- Probabilistic Boolean net models?
  - Max likelihood training...EM...?
- SVM based models... Boolean kernel machines...?
How Good Are Boolean Models?

Advantages

- Provide good \textit{qualitative} interpretation of regulation
- Particularly important for switching behaviors
  - Phage lysis...sporulation...Drosophila patterns...
  - Such systems are "robust" wrt exact expression values
- Useful connection with evolutionary behaviors

Disadvantages

- Boolean abstraction is poor fit to real expression data
- Cannot model important features:
  - Amplification of a signal; subtraction and addition of signals
  - Handling smoothly varying environmental parameter (e.g. temperature, nutrients)
  - Temporal performance behavior (e.g. cell cycle period)
  - Negative feedback control (Boolean model oscillates vs. stabilize)
<table>
<thead>
<tr>
<th>A Variety of Regulatory Network Models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finite-field models:</strong> ( X(t+1)=p(X) )</td>
</tr>
<tr>
<td>- ( p ) is a polynomial over finite field</td>
</tr>
<tr>
<td>- Generalizes the Boolean model</td>
</tr>
<tr>
<td><strong>Differential equations models:</strong> describe ( \frac{dX}{dt}=f(X) )</td>
</tr>
<tr>
<td>- ( f ) describes non-linear control of change by neighbors</td>
</tr>
<tr>
<td><strong>Linear model:</strong> ( X(t+1)=W X+ B )</td>
</tr>
<tr>
<td>- ( W ) is a weight matrix; linear approximation near steady state</td>
</tr>
<tr>
<td><strong>Neural network models:</strong> ( x_i(t)=\sigma(WX_{\text{Neighbors}(i)}+B) )</td>
</tr>
<tr>
<td>- Sigmoid non-linearity can be trained through gradient algorithm</td>
</tr>
<tr>
<td>- Comes with a learning algorithm</td>
</tr>
<tr>
<td><strong>Bayesian network models</strong>…</td>
</tr>
</tbody>
</table>