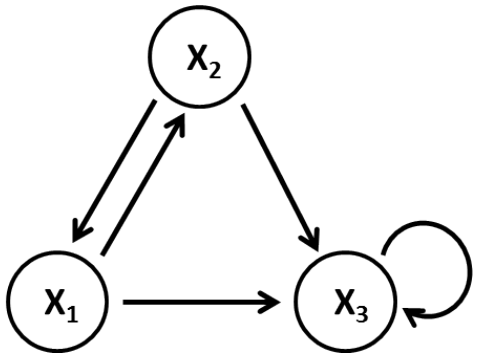
Skalierbares Erlernen Boolescher Netzwerke



Konstantin Heybrock

13 Juli 2023

Genregulation

- Steuerung der Aktivität von Genen
- Zustände von Genen (fast) binär
- aktives Gen kann transkribiert werden
  ↪ kodiertes Protein wird gebildet
- inaktives Gen wird nicht transkribiert
  ↪ kodiertes Protein wird nicht gebildet
- steuerbare Proteinsynthese spart Energie
- essenziell für alle Lebensformen
- Unteschiede zwischen Ein-/Mehrzellern

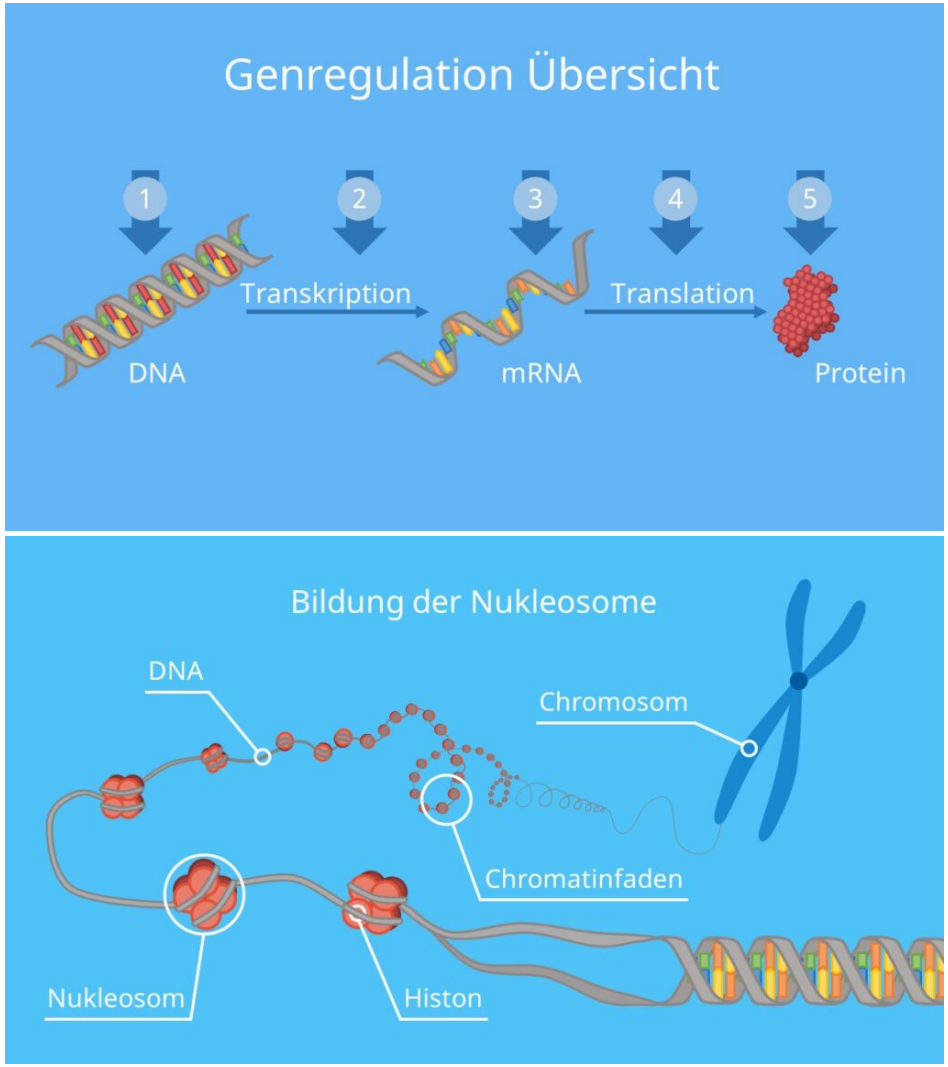


Abb. 1: Ebenen der Genregulation. Abb. 2: DNA-Methylierung
Quelle: https://studyflix.de/biologie/genregulation-2647

Repräsentation durch Boolesches Netzwerk

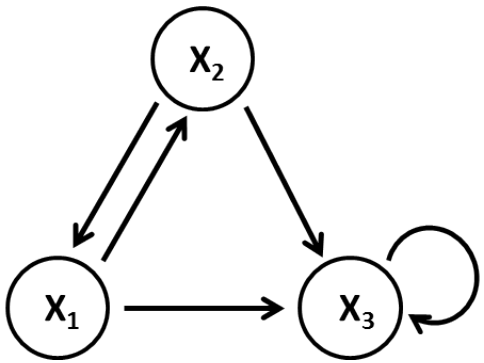
Definition

Boolesches Netzwerk.
Für $N \in \mathbb{N}$ seien $X_1(t), ..., X_N(t)$ boolesche Variablen, veränderlich in diskreten Zeitschritten.
Zu jeder Variable X_i sei f_{X_i} die zugehörige boolesche Übergangsfunktion. D.h. es gilt:

$$X_i(t+1) = f_{X_i}\left(X_{k_1}(t), ..., X_{k_m}(t)\right) \quad \forall t \in \mathbb{N}$$

Zustandswechsel passieren dabei simultan und sind deterministisch.

- Variablen werden oft als Knoten bezeichnet und das Netzwerk als Graph dargestellt
- Zustand des Netzwerks zum Zeitpunkt t: $\left(X_1(t) \ \ldots \ X_N(t)\right)^T \in \{0,1\}^N$



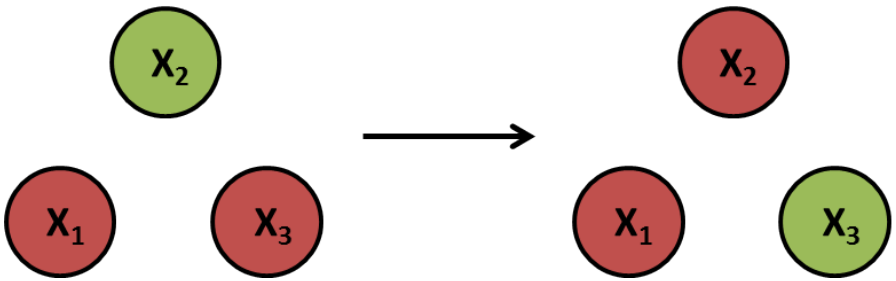
$$\begin{cases} X_1(t+1) = \neg X_2(t) \\ X_2(t+1) = X_1(t) \\ X_3(t+1) = \neg X_3(t) \wedge \left(X_1(t) \vee X_2(t)\right) \end{cases}$$

Zielsetzung

Situation:

- betrachte Lebewesen mit all seinen Genen $(X_1, ..., X_N)$
- Zusammenhänge zwischen Zuständen von Genen $(f_{X_1}, ..., f_{X_N})$ unbekannt
- Zustände aller Gene messbar (mRNA Konzentration)
- Zustandsänderung beobachtbar

Beobachtbar



Unbekannt

$$\begin{cases} X_1(t+1) = \neg X_2(t) \\ X_2(t+1) = X_1(t) \\ X_3(t+1) = \neg X_3(t) \wedge \big(X_1(t) \vee X_2(t)\big) \end{cases}$$

Zielsetzung

Ziel: Erlernen eines Booleschen Netzwerks zur Beschreibung der Genregulation aller Gene aus den Beobachtungen.

Anforderungen:

- aktive/inaktive Gene des nächsten Zeitschritts aus aktuellem Zustand vorhersagen
- Zusammenhänge zwischen Genen sollen explizit erklärt werden
  $\Rightarrow$ Kein reines Klassifikationsproblem (Black box Vorhersage des Nachfolgezustands)

Problem:

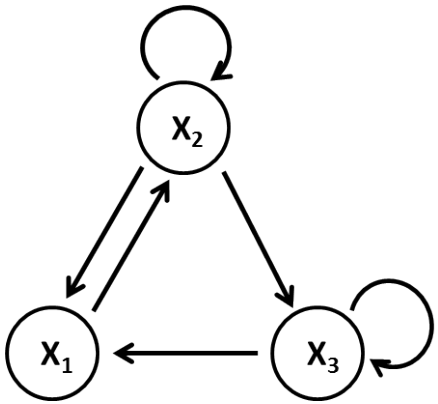
- Bisher effizienteste Kodierung boolescher Funktion f mit bis zu N Argumenten: binäre 2×2^N Matrix
  $\Rightarrow$ Modell hätte exponentiell viele Parameter
- bereits simple Lebensformen wie z.B. die Backhefe besitzen tausende Gene

Modellierung: Der Modellraum

Definition

Und/Oder Boolesches Netzwerk.
Ein Und/Oder Boolesches Netzwerk ist ein Boolesches Netzwerk, dessen Übergangsfunktionen reine Konjunktionen oder Disjunktionen sind.

- beschränke Suche auf Modellraum $\mathcal{M} = \left\{ M : M \text{ ist Und/Oder BN auf } X_1, ..., X_N \right\}$
- echte Einschränkung der Ausdrucksfähigkeit
- erzeugt eigene Probleme (siehe später)
- empirisch legitimiert durch Häufigkeit von Und/Oder Knoten in Genregulation

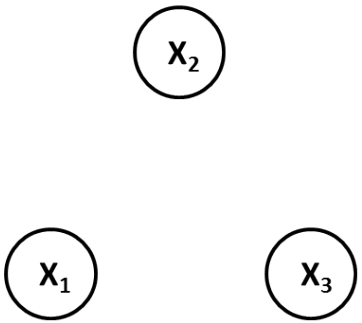


$$\begin{cases} X_1(t+1) = \neg X_2(t) \vee X_3(t) \\ X_2(t+1) = X_1(t) \vee \neg X_2(t) \vee \neg X_3(t) \\ X_3(t+1) = \neg X_3(t) \wedge X_2(t) \end{cases}$$

Modellierung: Kodierung der Und/Oder BNs

Matrix Q:

- binäre $N \times 2N$ Matrix
- Zeile i kodiert Argumente von i-ter Übergangsfunktion f_{X_i}
- $2N$ Spalten da Argumente negativ oder positiv eingehen können
- bisher keine Kodierung des Knotentyps



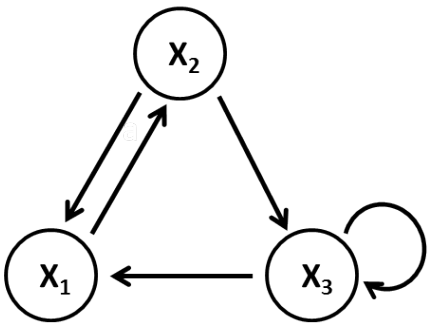
$$\begin{cases} X_1(t+1) = \neg X_2(t) \square X_3(t) \\ X_2(t+1) = X_1(t) \\ X_3(t+1) = \neg X_3(t) \square X_2(t) \end{cases}$$

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Modellierung: Kodierung der Und/Oder BNs

Vektor θ:

- Schwellwertvektor aus $\{1, ..., N\}^N$
- Eintrag i kodiert Knotentyp von X_i
- Übergangsfunktion f_{X_i} Disjunktion $\Rightarrow \theta(i) = 1$
- f_{X_i} Konjunktion $\Rightarrow \theta(i) = \sum_j \mathbf{Q}(i,j)$
  - Anzahl der Argumente von f_{X_i}
- $\theta(i)$ gibt an wie viele Argumente wahr sein müssen, damit f_{X_i} wahr ist



$$\begin{cases} X_1(t+1) = \neg X_2(t) \wedge X_3(t) \\ X_2(t+1) = X_1(t) \\ X_3(t+1) = \neg X_3(t) \vee X_2(t) \end{cases}$$

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \theta = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

Modellierung: Kodierung der Und/Oder BNs

- gegeben Zustand $\boldsymbol{s}_{\text{in}} = (x_i \dots x_N)^T$, berechne Nachfolgezustand $\boldsymbol{s}_{\text{out}} = \left(\boldsymbol{Q} \cdot \boldsymbol{s}_{\text{in}}^d\right)_{\geq \theta}$
  - Eintrag $\left(\boldsymbol{Q} \cdot \boldsymbol{s}_{\text{in}}^d\right)(i) = \#\text{Argumenten von } f_{x_i} \text{ wahr in } \boldsymbol{s}_{\text{in}}$
  - Vergleich mit $\theta(i)$ besagt, ob genug Argumente wahr sind
- Modellraum $\mathcal{M} = \left\{(\mathbf{Q}, \boldsymbol{\theta}) \;\middle|\; \mathbf{Q} \in \{0,1\}^N,\ \boldsymbol{\theta}(i) \in \{1, m_i\}\right\}$
- $\left|\mathcal{M}\right| \sim 3^{N^2} 2^N$
- Berechnen der Vorhersagengenauigkeit (Accuracy) auf den Daten für alle Wahlen von Q und θ zu aufwändig

Netzwerk Erlernen: Übersicht

Daten:

- $\mathbf{S}_{in}^{d}$ dualisierte Eingabezustände
- $\mathbf{S}_{\text{out}}$ Daten der Nachfolgezustände

Ziel: Erlernen von $\mathbf{Q}$ und θ,
sodass $\mathbf{S}_{\text{out}} = \left(\mathbf{Q} \cdot \mathbf{S}_{in}^{d}\right)_{\geq \theta(i)}$ gilt.

Knotenweise:

- entspricht Erlernen der f_{X_i} einzeln
- $\mathbf{a}_i := \mathbf{S}_{\text{out}}(i,:) \in \{0,1\}^L$
- $\mathbf{b}_i := \mathbf{Q}(i,:) \in \{0,1\}^{2N}$
- Suche $\mathbf{b}_i$, $\theta(i)$ sodass
  $\mathbf{a}_i = \left(\mathbf{b}_i \cdot \mathbf{S}_{in}^{d}\right)_{\geq \theta(i)}$ gilt

$$\mathbf{S}_{in}^{d} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad 2N \times L$$

$$\mathbf{S}_{\text{out}} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \qquad N \times L$$

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}}_{\substack{\text{L Nachfolge-} \\ \text{zustände von } X_1}} = \Big(\underbrace{\begin{bmatrix} ? & ? & ? & ? & ? & ? \end{bmatrix}}_{\text{kodiert } f_{X_1}} \cdot \mathbf{S}_{in}^{d}\Big)_{\geq ?}$$

Die Verlustfunktionen

- definiere Verlustfunktionen auf $\mathbb{R}^{2N}$ mit gewünschten Eigenschaften
- suche reellen Vektor $\mathbf{b}_i$, der den Verlust minimiert (möglicherweise zu 0)
  - dazu rate den Knotentyp zunächst geschickt
- binarisiere das erlernte $\boldsymbol{b}_i$ zu $\boldsymbol{b}_i^* \in \{0,1\}^{2N}$
- bestimme $\boldsymbol{\theta}(i)$ aus Knotentyp und erlerntem $\mathbf{b}_i$

$$
\begin{aligned}
J_i^{\text{and}}(\mathbf{b}_i) \quad = \quad & \Big((\mathbf{1} - \mathbf{a}_i) \cdot \big(\mathbf{1} - \vec{\min}\big\{ \mathbf{1},\, \mathbf{b}_i \cdot (\mathbb{1} - \mathbf{S}_{in}^d) \big\} \big) \Big) \\
& + \frac{1}{2} \Big(\mathbf{a}_i \cdot \big((\mathbf{b}_i \vec{\cdot}\, \mathbf{b}_i) \cdot (\mathbb{1} - \mathbf{S}_{in}^d) \big) \Big) \\
& + \frac{1}{2} \Big|\Big| \mathbf{b}_i \vec{\cdot}\, (\mathbf{1} - \mathbf{b}_i) \Big|\Big|_F^2
\end{aligned}
$$

$$
\begin{aligned}
J_i^{\text{or}}(\mathbf{b}_i) \quad = \quad & \Big(\mathbf{a}_i \cdot \big(\mathbf{1} - \vec{\min}\big\{ \mathbf{1},\, \mathbf{b}_i \cdot \mathbf{S}_{in}^d \big\} \big) \Big) \\
& + \frac{1}{2} \Big((\mathbf{1} - \mathbf{a}_i) \cdot \big((\mathbf{b}_i \vec{\cdot}\, \mathbf{b}_i) \cdot \mathbf{S}_{in}^d \big) \Big) \\
& + \frac{1}{2} \Big|\Big| \mathbf{b}_i \vec{\cdot}\, (\mathbf{1} - \mathbf{b}_i) \Big|\Big|_F^2
\end{aligned}
$$

Eigenschaften der Verlustfunktionen

Proposition

Macht das Modell für Knoten i $\left(\mathbf{b}_i, \theta(i)\right)$ *keine Fehler auf den Daten, so ist der Verlust 0. Ist der Verlust für ein* $\mathbf{b}_i$ *gleich 0, so lassen sich die gegebenen Übergänge von Knoten i mit der Kon- bzw. Disjunktion gegeben durch* $(\mathbf{b}_i, \theta(i))$ *korrekt beschreiben.*

Seien $\mathbf{b}_i \in \{0,1\}^{2N}$ *und* $\theta(i) \in \left\{1, \sum_j \mathbf{b}_i(j)\right\}$, *sowie* $\mathbf{a}_i$ *und* $\mathbf{S}_{in}^d$ *Zustandsdaten wie zuvor, dann gilt:*

$$\mathbf{a}_i = \left(\mathbf{b}_i \cdot \mathbf{S}_{in}^d\right)_{\geq \theta(i)} \iff J_i(\mathbf{b}_i) = 0$$

Beweisidee für den Fall $\theta(i) = 1$:

- jeder Summand der Verlustfunktionen ist nicht negativ
- $\left|\left|\mathbf{b}_i \overset{\rightarrow}{\cdot} (\mathbf{1} - \mathbf{b}_i)\right|\right|_F^2 = 0 \Leftrightarrow \mathbf{b}_i$ binär
- $\left(\mathbf{a}_i \cdot \left(\mathbf{1} - \overrightarrow{\min}\left\{\mathbf{1},\ \mathbf{b}_i \cdot \mathbf{S}_{in}^d\right\}\right)\right) = \#$fälschlich vorhergesagte 0er
- $\left((\mathbf{1} - \mathbf{a}_i) \cdot \left((\mathbf{b}_i \overset{\rightarrow}{\cdot} \mathbf{b}_i) \cdot \mathbf{S}_{in}^d\right)\right) = \#$fälschlich vorhergesagte 1er

Bemerkung: Nullstelle je nach Daten nicht eindeutig/existent

Eigenschaften bei Absenz von Optimal-Lösung

- verursacht durch Messfehler/Rauschen oder Daten nicht von Und/Oder Knoten
- Verlustfunktionen besitzen keine Nullstellen
- Lokale Minima in der Regel nicht binär
- Summand 1 der Verlustfunktion bestraft zu kleine Einträge von $\mathbf{b}_i$, die weniger Fehler verursachen, wenn sie 1 wären
- Summand 2 bestraft zu große Einträge von $\mathbf{b}_i$, die weniger Fehler verursachen, wenn sie 0 wären
- Summand 3 bestraft betragsmäßig große Einträge
  - wichtig damit durch Summand 1 nicht zu beliebig großen Werten führt
- $\boldsymbol{b}_i$ nahe am globalen Minimum $\Rightarrow$ sinnvoll binarisierbar

Newtonschritt	b_i	Verlust
zufälliger Start	$\begin{bmatrix}0,91 & 0,79 & 0,06 & 0,84 & 0,64 & 0,72\end{bmatrix}$	41,3
1	$\begin{bmatrix}0,64 & 0,22 & 0,03 & 0,43 & 0,60 & 0,55\end{bmatrix}$	36,4
2	$\begin{bmatrix}0,37 & 0,00 & 0,06 & 0,07 & 0,53 & 0,34\end{bmatrix}$	10,8
Minimum	$\begin{bmatrix}0,08 & 0,02 & 0,08 & 0,07 & 0,83 & 0,08\end{bmatrix}$	2,58
echter Knoten	$\begin{bmatrix}0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$	14

Der Trainingsalgorithmus

Algorithm 1 for learning an AND/OR BN as matrix

Input: $N \times L$ binary matrices $\boldsymbol{S}_{in}, \boldsymbol{S}_{out}$
Output: $N \times 2N$ binary matrix $\boldsymbol{Q}$, N dim. threshold vector $\boldsymbol{\theta}$
s.t. $\boldsymbol{S}_{out} = (\boldsymbol{Q}\boldsymbol{S}_{in}^{d})_{\geq \boldsymbol{\theta}}$ where $\boldsymbol{S}_{in}^{d} = [\boldsymbol{S}_{in}; (\mathbb{1} - \boldsymbol{S}_{in})]$

for $i = 1$ **to** N **do**
  randomly initialize $2N$ dim. row vector $\boldsymbol{b}_i$
  for $p = 1$ **to** max_try **do**
    determine node_type (AND,OR) of node i
    for $q = 1$ **to** max_itr **do**
      compute J_i (1),(2) and $\boldsymbol{J}_{i\_Jacob}$ (3),(4)
      update $\boldsymbol{b}_i$ by (5)
      threshold $\boldsymbol{b}_i$ to a binary vector $\boldsymbol{b}_i^*$
      set $\boldsymbol{Q}(i,:) = \boldsymbol{b}_i^*$
      if node_type = AND **then**
        put $\boldsymbol{\theta}(i) = \|\boldsymbol{b}_i^*\|_1$
      else
        $\boldsymbol{\theta}(i) = 1$
      end if
      compute $\epsilon_i = \|\boldsymbol{a}_i - (\boldsymbol{b}_i^* \boldsymbol{S}_{in}^{d})_{\geq \boldsymbol{\theta}(i)}\|_1$
      if $\epsilon_i = 0$ **then**
        exit p-loop
      end if
    end for
    $\boldsymbol{b}_i = 0.5 \cdot \boldsymbol{b}_i + 0.5 \cdot \Delta$ $\%\Delta \sim$ U(0,1)
  end for
end for
return $\boldsymbol{Q}$ and $\boldsymbol{\theta}$

Eigene Experimente

- generiertes Netzwerk mit uniform 1-5 Argumenten pro Übergangsfunktion
- $\mathbf{S}_{in}^{d} \in \{0,1\}^{2N \times L}$ uniform verteilt (L=100)
- $\mathbf{S}_{\text{out}}$ daraus berechnet
  - realistischer: Mit einem Zustand starten und L mal Nachfolgezustand berechnen
- Training mit max-try=10 und max-itr=100

N	3	10	100	1000	N=100, L=1000
Trainingsdauer (s)	0,01	0,06	1,4	176	2,8
Trainingsgenauigkeit (%)	100	100	100	100	100
Testgenauigkeit (%)	100	99,5	98,7	97,5	100
korrekte Knoten (%)	100	99	65	44,9	100
maximale Versuche bis finaler Knotentyp	1	1	2	5	1
max Newtonschritte bei finalem Knotentyp	2	3	3	11	4

Eigene Experimente: Newton vs. Gradient Descent

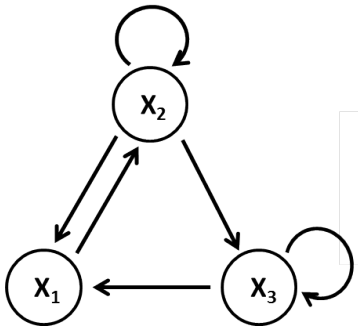
- in Abwesenheit einer Nullstelle konvergiert Newton Verfahren i. A. nicht
- $b_i \leftarrow b_i - \frac{J_i}{||\nabla J_i||^2} \nabla J_i$
- häufig: Springt um lokales Minimum (oder mehrere) umher
- im Experiment ca. 10% der Bits von $\mathbf{S}_{\text{out}}$ invertiert

Newtonschritt	b_i	Verlust
48	$\begin{bmatrix} 0,26 & 0,18 & 0,25 & 0,45 & 0,50 & 0,33 \end{bmatrix}$	47,2
49	$\begin{bmatrix} 0,25 & 0,31 & 0,31 & 0,56 & 0,48 & 0,37 \end{bmatrix}$	7,2
50	$\begin{bmatrix} 0,08 & -0,01 & 0,06 & 0,19 & 0,24 & 0,08 \end{bmatrix}$	5,7
51	$\begin{bmatrix} 0,26 & 0,18 & 0,25 & 0,45 & 0,50 & 0,33 \end{bmatrix}$	47,2
Minimum	$\begin{bmatrix} 0,22 & 0,23 & 0,26 & 0,49 & 0,50 & 0,26 \end{bmatrix}$	3,68
echter Knoten	$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$	

Newton vs. Gradient Descent

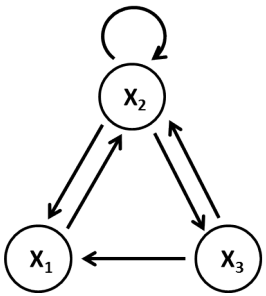
- $N = 3$, $L = 100$, Rauschen$= 0,2$
- für größere N keine signifikanten Unterschiede gefunden

Netzwerk:



$$\begin{cases} X_1(t+1) = \neg X_2(t) \vee X_3(t) \\ X_2(t+1) = X_1(t) \vee \neg X_2(t) \vee \neg X_3(t) \\ X_3(t+1) = \neg X_3(t) \wedge X_2(t) \end{cases}$$

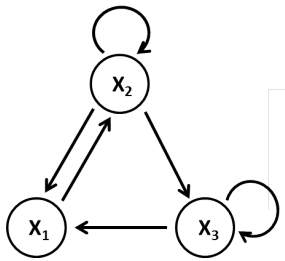
Newton:



$$\begin{cases} X_1(t+1) = \neg X_2(t) \vee X_3(t) \\ X_2(t+1) = X_1(t) \vee X_2(t) \vee X_3(t) \vee \neg X_1(t) \vee \neg X_2(t) \vee \neg X_3(t) \\ X_3(t+1) = X_2(t) \end{cases}$$

- falls Nullstelle vorhanden, schnellere Konvergenz
- weniger Hyperparameter

Gradient Descent:



$$\begin{cases} X_1(t+1) = \neg X_2(t) \vee X_3(t) \\ X_2(t+1) = X_1(t) \vee \neg X_2(t) \vee \neg X_3(\\ X_3(t+1) = \neg X_3(t) \wedge X_2(t) \end{cases}$$

- erreicht niedrigere Verlustwerte

Experiment mit echten Daten

Backhefe (Saccharomyces cerevisiae):

- erste Eukaryote (Mehrzeller) mit vollständig entschlüsseltem Genom
- Pilz mit 6275 Genen
  - im Artikel $N = 10.928$ mRNA Sequenzen identifiziert

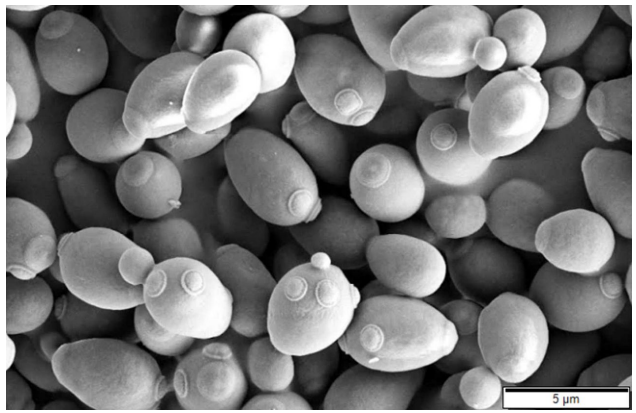


Abb. 3: Saccharomyces cerevisiae. Quelle: https://de.wikipedia.org/wiki/Backhefe

Datensatz E-MTAB01:

- 41 beobachtete Zustände (40 Übergänge) der 10.928 mRNA Sequenzen
- Zeitspanne: 3 Zellzyklen
- Tabelle zeigt Ergebnis der Autoren Sato et al. [1]
- mittlere Testgenauigkeit bei 4 facher Kreuzvalidierung aber nur 40,9%
- dies legt nahe, dass Überanpassung an Daten stattfindet

Datensatz	Zeit (h)	Trainingsgenauigkeit (%)
E-MTAB01	28,9	87,3

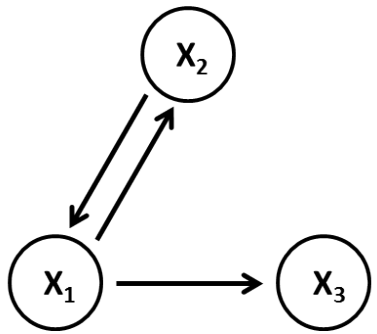
Experimente mit echten Daten

- mit einigen Verbesserungen: mittlere Testgenauigkeit = 84,3%
- bei akzeptablen Knoten (ca. 6600): mittlere Testgenauigkeit = 94,5%
- im Umkehrschluss für restliche Knoten: mittlere Testgenauigkeit ca. 68,1%
- nicht angegeben ob viele der akzeptablen Knoten trivial sind
  - z.B. $X_i(t) = 1 \; \forall t$ und $X_i(t+1) = X_i$

Probleme: Daten

- einseitige Daten enthalten wenig Information über bestimmte Zustandswechsel
  - einseitige Daten entstehen u. a. durch sequenzielle Datengewinnung (siehe Beispiel)
- Variablen sind abhängig von Umwelteinflüssen oder evolutive Überreste
  - z.B. Gene, die Proteine zur Imunabwehr produzieren, könnten unter Laborbedingungen permanent inaktiv sein
- Variablen, die auf Daten fast nur einen Zustand annehmen, sollten ignoriert werden
  - andernfalls verfälschen sie die Genauigkeit in positive Richtung
- Verhältnis von Variablen zu Daten groß entspricht unterbestimmten Gleichungssystem
  - es sind viele Lösungen/Nullstellen zu erwarten

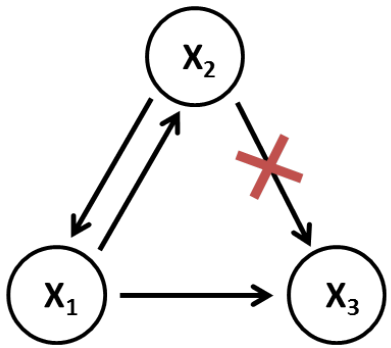
$$\begin{pmatrix}0\\1\\0\end{pmatrix} \rightarrow \begin{pmatrix}1\\0\\1\end{pmatrix} \rightarrow \begin{pmatrix}0\\1\\0\end{pmatrix}$$



$$\begin{cases} X_1(t+1) = X_2(t) \\ X_2(t+1) = X_1(t) \\ X_3(t+1) = \neg X_1(t) \end{cases}$$

Probleme: Korrelation vs. kausaler Zusammenhang

- Allgemeines Problem von Klassifikation: Ergebnis resultiert aus Korrelation der Variablen. Diese muss keinen kausalen Zusammenhang bedeuten.
- erklärt teilweise wieso Testgenauigkeit $98,7\%$ aber nur $44,9\%$ richtige Übergangsfunktionen möglich ist

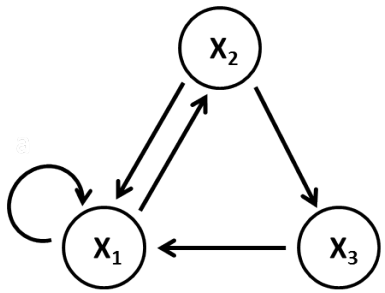


$$\begin{cases} X_1(t+1) = X_2(t) \\ X_2(t+1) = X_1(t) \\ X_3(t+1) = \neg X_1(t) \end{cases}$$

$$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \to \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \to \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \implies X_3(t+1) = X_2(t)$$

Overfitting

- bei vielen Variablen (und wenig Daten) besteht hohe Chance, dass Übergänge einer Varibale von anderer Variable durch Zufall erklärt werden
- zudem kann Hinzufügen/Wegnehmen einer Variable der Formel auf den Daten keine Auswirkung hat



$$\{ \quad X_1(t+1) = \neg X_3(t)$$

$$\begin{pmatrix}0\\1\\1\end{pmatrix} \to \begin{pmatrix}1\\0\\1\end{pmatrix} \quad \begin{pmatrix}0\\0\\1\end{pmatrix} \to \begin{pmatrix}0\\1\\0\end{pmatrix} \quad \begin{pmatrix}1\\1\\0\end{pmatrix} \to \begin{pmatrix}1\\1\\0\end{pmatrix}$$

- $\begin{pmatrix}1\\1\\1\end{pmatrix} \to \begin{pmatrix}0\\1\\1\end{pmatrix}$
- Lösungen:
  - $X_1(t+1) = X_2(t) \vee \neg X_3(t)$
  - $X_1(t+1) = X_1(t) \vee X_2(t) \vee \neg X_3(t)$
- beschränke maximalen Argumente auf 1: $X_1(t+1) = \neg X_3(t)$

Lösungsansätze

- Overfitting mit Beschränken der Argumentenzahl pro Funktion entgegenwirken
- Overfitting mit Auswahl des Modells nach Testgenauigkeit entgegenwirken
- größere Datensätze erstellen
- auf Gene mit viel Variabilität beschränken
  - hier nicht getan, um Skalierbarkeit zu demonstrieren
- Ergebnisse nur zur Erklärung akzeptabler Knoten nutzen
- Variablen mit niedriger Testgenauigkeit und bekannten, wenigen Argumenten: Andere Verfahren für diese Variablen nutzen
  - dadurch könnten nicht Und/Oder Knoten modelliert werden

Quellen

Literatur

[1] A. Melkman, T. T. Determining a singleton attractor of an and/or boolean network in o(1.587n) time. *Information Processing Letters 110* (2010), 565–569.

[2] Taisuke Sato, K. Boolean network learning in vector spaces for genome-wide network analysis. *In Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning 18* (2021), 560—-569.