GOETHE UNIVERSITÄT
FRANKFURT AM MAIN

Institut für Informatik, Goethe-Universität Frankfurt am Main

Seminar zur Künstlichen Intelligenz
Sommersemester 2023

Leitung:
Prof. Dr. Manfred Schmidt-Schauß

Erlernen von UND/ODER booleschen Netzwerken

Konstantin Heybrock

Frankfurt am Main
23. Juli 2023

Zusammenfassung

Diese Ausarbeitung dient als Ergänzung zum gleichnamigen Vortrag am 13.07.2023. Gemeinsam sollen sie den Artikel „Boolean Network Learning in Vector Spaces for Genome-wide Network Analysis" (Sato et al. [3]) anschaulich erläutern. In diesem wird das Erlernen von booleschen Netzwerken behandelt und mittels einer Einschränkung skalierbar gemacht.

Inhaltsverzeichnis

1 Einleitung

Ein boolesches Netzwerk versteht man am besten an einem Beispiel. Dazu muss zunächst erklärt werden, für welche Phänomene solche Netzwerke genutzt werden. Neben der Anwendung in der statistischen Physik, werden boolesche Netzwerke vor allem in der Biologie zur Darstellung von genregulatorischen Prozessen genutzt.

1.1 Genregulation

Genregulation ist ein Begriff der Genetik. Er beschreibt die Steuerung der Aktivität der Gene.
Bekanntlich verfügt jede Zelle eines Lebewesens über das gesamte genetische Material des Organismus. Dennoch übernehmen verschiedene Zelltypen sehr verschiedene Aufgaben, an die sie speziell angepasst sind. Dies ist dank Genregulation möglich. Je nach Zelltyp sind andere Bereiche der DNA für die Polymerase zugänglich. Somit ist sichergestellt, dass jede Zelle nur die von ihr benötigten Proteine synthetisiert und keine kostbare Energie vergeudet wird. Gene, die von der Polymerase gelesen werden können, bezeichnet man dabei als aktiv. Abbildung 1 zeigt wie die Genregulation in diesem Fall funktioniert.

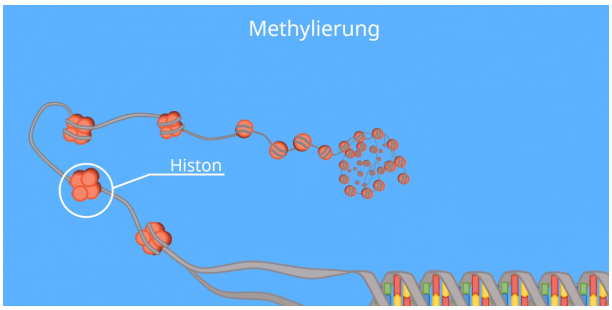


Abb. 1: Um Histone gewickelte DNA. Dies verhindert die Transkription des entsprechenden Abschnitts. Quelle: https://studyflix.de/biologie/genregulation-2647

Auch auf niedrigerer Ebene findet Genregulation statt. So ist nicht jedes Gen, das für eine Zelle generell benötigt wird, dauerhaft aktiv. Auch dies ist wichtig, um wertvolle Energie zu sparen. Am besten schaut man sich dies am (einzelligen) Beispiel des E. coli Bakteriums an.

E. coli Bakterien ernähren sich vorzugsweise von Glucose. Diese können sie unter geringem Aufwand verwerten. Steht allerdings lediglich ein anderer Zucker, wie z.B. die Lactose zur Verfügung, so begnügen sich die Bakterien auch damit. Untersuchungen zeigen, dass im Falle von ausreichender Glucose und Lactose Konzentration, fast ausschließlich Glucose von den Bakterien verstoffwechselt wird. Somit ist sowohl die Glucoseverarbeitung, als auch die Lactoseverarbeitung nahezu binär. Entweder die Bakterien verstoffwechselen Glucose/Lactose oder nicht. Abbildung 2 zeigt die Funktionsweise der Genregulation in diesem Fall.

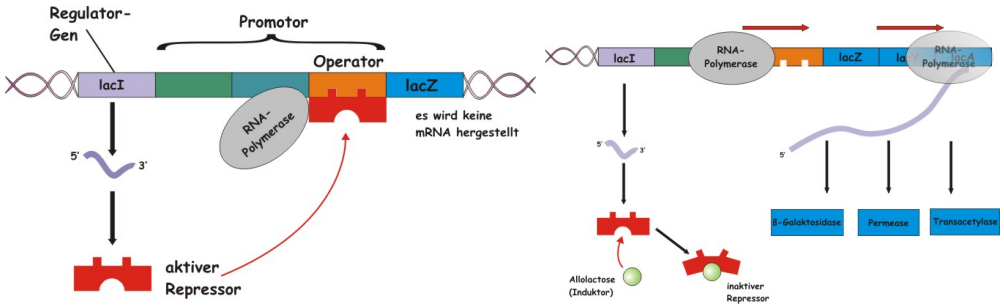
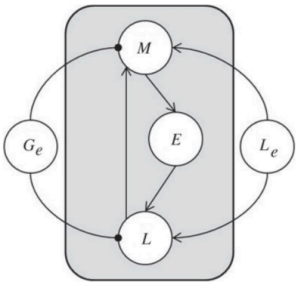


Abb. 2: Lac Operon E. Coli. Links: Aktiver Regulator verhinder Transkription von Laktase. Diese wird zum Abbau von Laktose benötigt. Rechts: In Zelle vorhandene Laktose bindet an Repressor und deaktiviert ihn. Nun kann Laktase produziert werden.
Quelle: http://www.guidobauersachs.de/bc/lac.html

1.2 Boolesche Netzwerke

Die (nahezu) binäre Gestalt der Genregulation ermöglicht es die Dynamik des biologischen Prozesses mittels binärer Variablen und booleschen Funktionen zu beschreiben. Dies geschieht meist in Form eines booleschen Netzwerkes.
Ein boolesches Netzwerk ist ein Übergangssystem endlicher vieler boolescher Variablen $x_1, ..., x_N$, die ihren Zustand in diskreten Zeitschritten verändern. Dabei sind die Variablen im Falle von Genregulation meist Gene, die entweder aktiv oder inaktiv sind.
Beschrieben wird das System durch einen gerichteten Graphen dessen Knoten die Variablen darstellen. Eine Kante von Knoten i nach j symbolisiert einen Einfluss von Variable x_i auf die Zustandsänderung von Variable x_j. Die Zustandsänderung der Variable x_i in einem Zeitschritt wird durch die boolesche Funktion f_i beschrieben. Befindet sich Variable x_i zum diskreten Zeitpunkt t im Zustand $x_i(t)$, so ist ihr Zustand einen Zeitschritt später durch $x_i(t+1) = f_i(x_{k_1}(t), ..., x_{k_m}(t))$ gegeben. Die $k_1, ..., k_m$ sind dabei die Nachbarn von Knoten i. Die Funktionen f_i werden hier als deterministisch und unabhängig vom Zeitpunkt betrachtet.

Der Zustand des gesamten Systems lässt sich über den Vektor $[x_1, .., x_N]^T \in \{0,1\}^N$ darstellen. Abbildung 3 zeigt ein mögliches boolesches Netzwerk für das Beispiel des Lac Operons von E. Coli Bakterien.



Parameter:
G_e = Externe Glukose
L_e = Externe Laktose

Variablen:
L = Interne Laktose
M = mRNA der Lac Gen-Produkte
E = LacY und LacZ Gen-Produkte

$$L(t+1) = f_L(G_e, L_e, E(t), L(t)) = \neg G_e \wedge \Big((L_e \wedge E(t)) \vee (L(t) \wedge \neg E(t)) \Big)$$
$$M(t+1) = f_M(G_e, L_e, L(t)) = \neg G_e \wedge (L(t) \vee L_e)$$
$$E(t+1) = f_E(M(t)) = M(t)$$

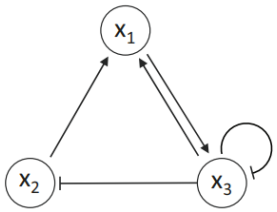
Abb. 3: Boolesches Netzwerk zur Beschreibung des Lac Operons. LacY ist das Transportprotein für Laktose und LacZ ist Laktase, das Enzym, das Laktose umsetzt.

Im Artikel von Sato et al. [3] wird sich mit dem Erlernen solcher Netzwerke beschäftigt. Gegeben ist also ein System an booleschen Variablen (meist Gene) und Daten, die aus zahlreichen Zustandspaaren $\big([x_1(t), .., x_N(t)], [x_1(t+1), .., x_N(t+1)]\big)$ bestehen. Ziel ist es nun Übergangsfunktionen $f_1, ..., f_N$ zu erlernen, zu denen die gegeben Übergänge passen. Also die kausalen Zusammenhänge zwischen Zuständen der Variablen zu verstehen.
Es ist möglich booleschen Funktionen als multinomiale Polynome darzustellen. Unter einigen Einschränkungen ist es dann denkbar mittels Gröbnerbasen-Techniken Lösungen für das Problem zu finden. Aufgrund der exponentiell wachsenden Anzahl an prinzipiell möglichen Zuständen $\big(\big|\{0,1\}^N\big| = 2^N\big)$, ist dieses Vorgehen jedoch nicht skalierbar. Das selbe Problem überträgt sich auch auf die Darstellung boolescher Netzwerke. Es ist nicht möglich allgemeine boolesche Netzwerke mit subexponentiellem Speicheraufwand zu kodieren. Im hier behandelten Artikel wird eine Möglichkeit gefunden, bestimmte boolesche Netzwerke in einer eleganten und effizienten Matrix-Vektor-Form darzustellen. Anschließend können Standardmethoden des überwachten Lernens angewandt werden, um eine Lösungsnäherung (im Sinne einer Verlustfunktion) zu erhalten. Das so gewonnene boolesche Netzwerk wird in der Regel nicht alle Übergänge richtig beschreiben, dennoch liefert dieses Vorgehen in der Anwendung durchaus gute Ergebnisse.

2 Hauptteil

2.1 UND/ODER boolesche Netzwerke

Sato et al. [3] beschränken sich für ihre Untersuchungen auf sogenannte UND/ODER BNs [1]. Diese kennzeichnen sich darüber aus, dass die booleschen Funktionen reine Konjunktionen oder Disjunktionen sind. Z.B. wäre im Beispiel aus Abbildung 3 f_M weder eine reine Konjunktion, noch eine reine Disjunktion. In UND/ODER BNs kann man zwischen UND-Knoten und ODER-Knoten unterscheiden. Ersteres ist ein Knoten mit zug. Übergangsfunktion $f(x_{k_1}, ..., x_{k_m}) = x_{k_1}^\circ \wedge x_{k_2}^\circ \wedge ... \wedge x_{k_m}^\circ$. Dabei ist das Literal x_i° entweder einfach x_i oder $\neg x_i$, hier als pos. bzw. neg. Literale bezeichnet. Abbildung 4 zeigt ein simples UND/ODER BN.



$$\begin{cases} x_1(t+1) = x_2(t) \wedge x_3(t) \\ x_2(t+1) = \neg x_3(t) \\ x_3(t+1) = x_1(t) \vee \neg x_3(t) \end{cases}$$

Abb. 4: Einfaches UND/ODER BN. Quelle [3]

2.2 Matrix-Vektor Darstellung

$$\boldsymbol{Q} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{\theta} = [2\ 1\ 1\ ]^T$$

Abb. 5: Matrix Vektor Darstellung des BNs aus Abb. 4. Quelle [3]

Im Folgenden ist für eine reelle Zahlen b die Schwellfunktion

$$(a)_{\geq b} = \begin{cases} 1 & \text{falls } a \geq b \\ 0 & sonst \end{cases}$$

für alle reellen a definiert. Diese wird auch vektorisiert verwendet. Vektoren werden dick gedruckt dargestellt.
Ein UND/ODER BN stellen Sato et al. [3] über eine binäre $N \times 2N$ Matrix $\mathbf{Q}$ und einen Schwellenvektor $\boldsymbol{\theta} \in \{0, ..., N\}^N$ dar. Dabei dient $\mathbf{Q}$ der Kodierung der vorkommenden Literale jeder Übergangsfunktion (zeilenweise).

Dafür werden für jede der N Übergansfunktionen $2N$ Bits benötigt, da zwischen negativen und positiven Literalen unterschieden werden muss.
Genauer: Sei i ein UND-Knoten mit zugehöriger Konjunktion
$C_i = x^\circ_{k_1} \wedge ... \wedge x^\circ_{k_m}$, dann gilt:

$$\mathbf{Q}(i,j) = \begin{cases} 1 & \text{falls } j \in \{k_1, ..., k_m\} \text{ und } x^\circ_j = x_j \\ -1 & \text{falls } j - N \in \{k_1, ..., k_m\} \text{ und } x^\circ_{j-N} = \neg x_j \\ 0 & \text{sonst} \end{cases}$$

ODER-Knoten werden auf die gleiche Weise als eine Zeile von $\mathbf{Q}$ kodiert. Der Schwellenvektor dient unter anderem dazu den Typ des jeweiligen Knotens zu speichern.
Ist i wieder der UND-Knoten von zuvor, so ist $\boldsymbol{\theta}(i) = m$, wobei m der Anzahl an Knoten mit Einfluss auf den Übergang von Knoten i ist. Ist i ein Oder-Knoten, so wird $\boldsymbol{\theta}(i) = 1$ gesetzt. Entscheidet man sich also Knoten deren Übergangsfunktion nur aus einem Literal besteht, als ODER-Knoten zu bezeichnen, so kann der Typ von Knoten i eindeutig aus $\boldsymbol{\theta}(i)$ abgelesen werden.
Für einen Zustand $\mathbf{s} = [x_1, ..., , x_N]^T$ definieren Sato et al. [3] den Vektor $\mathbf{s}^d = \begin{bmatrix} \mathbf{s} \\ \mathbf{1} - \mathbf{s} \end{bmatrix}$. Der Nachfolgezustand von Zustand $\mathbf{s}$ ist damit einfach durch $\left(\mathbf{Q}\mathbf{s}^d\right)_{\geq \boldsymbol{\theta}}$ gegeben. Eintrags weise erkennt man, dass $\mathbf{Q}(i,:)\mathbf{s}^d$ gerade die Anzahl an wahren Literalen (Wert 1) aus $\{x^\circ_{k_1}, ..., x^\circ_{k_m}\}$ ist. Dazu bedenke man, dass in diesem Skalarprodukt nur die x°_j mit 1 multipliziert werden, die auch in $\{x^\circ_{k_1}, ..., x^\circ_{k_m}\}$ vorkommen. Zudem zählt die erste Hälfte von $\mathbf{Q}(i,:)\mathbf{s}^d$ die positiven, wahren Literale und die zweite Hälfte die negativen, wahren Literale aus $\{x^\circ_{k_1}, ..., x^\circ_{k_m}\}$.
Ist i ein UND-Knoten, so ist $\boldsymbol{\theta}(i) = m$ und $\left(\mathbf{Q}(i,:)\mathbf{s}^d\right)_{\geq m}$ ist genau dann 1, wenn die Anzahl der wahren Literale gleich der gesamten Zahl an Literalen aus $\{x^\circ_{k_1}, ..., x^\circ_{k_m}\}$ entspricht. Dies ist äquivalent zu einer Konjunktion.
Ist i ein ODER-Knoten, so ist $\boldsymbol{\theta}(i) = 1$ und $\left(\mathbf{Q}(i,:)\mathbf{s}^d\right)_{\geq m}$ ist genau dann 1, wenn die Anzahl der wahren Literale mindestens 1 beträgt. Das entspricht einer Disjunktion.

2.3 Das Erlernen von UND/ODER BNs

Die meisten genregulatorischen Prozesse sind nicht so simpel, wie die des Lac Operons der E. Coli Bakterien. In den meisten Situationen ist es für einen Menschen praktisch unmöglich die genaue Übergangsdynamik (die $f_1, ..., f_N$) zu bestimmen.

In seltenen Fällen kann man Variablen (Gene, Proteine, Substrate) identifizieren, die höchst wahrscheinlich an der Genregulation beteiligt sind. Häufig muss/will man aber das gesamte Genom betrachten.
Je nach Situation liefert dies die Variablen. Zudem kann man empirisch beobachten von welchen Zuständen der Variablen in welche übergegangen wird. In diesem Abschnitt wird erklärt, wie Sato et al. [3] in einer solchen Situation, eine Näherung einer möglichen Übergangsdynamik bestimmen. Dabei ist ihr Modell vor allem auf große Anzahlen an Variablen spezialisiert und ermöglicht die Genom weite Betrachtung.

Zu gegebenen Variablen $X_1, ..., X_N$ sind eine $N \times L$ Matrix an Zustandsvektoren $\mathbf{S}_{in} = [\mathbf{s}_1^{in} \dots \mathbf{s}_L^{in}]$ und eine $N \times L$ Matrix an Nachfolgezuständen $\mathbf{S}_{out} = [\mathbf{s}_1^{out} \dots \mathbf{s}_L^{out}]$ gegeben. Es gilt nun eine Matrix $\mathbf{Q}$ und einen Vektor $\boldsymbol{\theta}$ zu finden, sodass $\mathbf{S}_{out} = \left(\mathbf{Q}\mathbf{S}_{in}^d\right)_{\geq \boldsymbol{\theta}}$ gilt. Dabei ist

$\mathbf{S}_{in}^d := \begin{bmatrix} \mathbf{S}_{in} \\ \mathbb{1} - \mathbf{S}_{in} \end{bmatrix} \in \{0,1\}^{2N \times L}$ die dualisierte Version von $\mathbf{S}_{in}$.
Zeilenweise betrachtet, sind dies die Gleichungen: $\mathbf{S}_{out}(i,:) = \left(\mathbf{Q}(i,:)\mathbf{S}_{in}^d\right)_{\geq \boldsymbol{\theta}(i)}$ $\forall i \in \{1, ..., N\}$. Für jede Variable X_i korrespondiert die Gleichung zur Übergangsfunktion f_i ausgewertet auf allen L Daten. Die Natur des Problems erlaubt es diese Gleichungen separat zu lösen, die f_i also unabhängig von einander zu bestimmen.
Sind zu wenige verschiedene Übergänge unter den Daten, so gibt es mit unter sehr viele Lösungen der Gleichung. Umgekehrt gilt auch: Entstammen die Daten in Wirklichkeit keinem UND/Oder Knoten oder sind mit Messfehlern/Rauschen belastet, so muss es keine Lösung der Gleichung geben.
Selbst wenn mindestens eine Lösung existiert, gibt es keine geschlossene Lösungsformel für dieses Problem. Deshalb setzen Sato et al. [3] dazu die Verlustminimierung ein. Dafür wird $\mathbf{Q}(i,:)$ zunächst als reellwertig betrachtet und es werden zwei Verlustfunktionen definiert.

Bezeichne $\mathbf{a}_i := \mathbf{S}_{out}(i,:)$ und $\mathbf{b}_i := \mathbf{Q}(i,:)$, dann sind die Verlustfunktionen definiert durch:

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

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

Dabei bezeichnet F die Frobeniusnorm. Da $\mathbf{b}_i$ ein Zeilenvektor ist, entspricht diese der euklidischen Norm. Zudem sind vektorisierte Operationen mit Pfeilen gekennzeichnet. $\mathbb{1}$ ist eine $2N \times L$ Matrix, die nur aus Einsen besteht.
Man kann zeigen, dass diese Verlustfunktionen für Knoten vom passenden Typen die gewünschten Eigenschaften erfüllen: Sie sind beide nicht negativ und bei bekanntem $\boldsymbol{\theta}$ für den jeweiligen Knotentypen genau dann 0, wenn auf den Daten fehlerfrei operiert wird. Darauf wird im Vortrag genauer eingegangen.
Da die Verlustfunktionen nicht von $\boldsymbol{\theta}$ abhängen, kann zunächst ein $\mathbf{Q}$ Zeilen weise über Minimierung der Verlustfunktionen gefunden werden. Anschließend können die $\boldsymbol{\theta}(i)$ aus $\mathbf{Q}(i,:)$ und dem jeweiligen Knotentypen berechnet werden. Ist i ein ODER-Knoten, so ist $\boldsymbol{\theta}(i) = 1$. Im Fall eines UND-Knotens ist die Bestimmung weniger direkt.
Doch zunächst muss erst geklärt werden, wie der Knotentyp entschieden wird. Eine naheliegende, naive Möglichkeit wäre es $\mathbf{Q}(i,:)$ bezüglicher beider Verlustfunktionen zu minimieren und den Knotentyp nach dem geringeren Verlust zu wählen. Diese Methode ist allerdings sehr unzuverlässig, da in die Minimierung viele Faktoren eingehen. Im Algorithmus des Artikels wird $\mathbf{Q}(i,:)$ zunächst zufällig initialisiert und klassifiziert indem 20 Schwellwerte $\mu_1, ..., \mu_{20}$ gleichmäßig aus $[\min_j(\mathbf{Q}(i,j)), \max_j(\mathbf{Q}(i,j))]$ gewählt. Anschließend wird $\mathbf{Q}(i,:)$ mittels jedem der Schwellwerte binärisiert. Diese seien hier mit $\mathbf{Q}(i,:)_k$ bezeichnet. Zudem werden für jeden dieser 20 Zeilenvektoren $\theta_{k,1} = 1$ und $\theta_{k,2} = ||\mathbf{Q}(i,:)_k||_1$ gewählt. Anschließend werden für alle 40 Fälle gezählt, wie viele Fehler bei der Bestimmung der Übergänge der Daten gemacht werden würden.

Wird die minimale Fehleranzahl für ein $\mathbf{Q}(i,:)_k$ mit $\theta_{k,1}$ erzielt, so wird der Knoten i als ODER-Knoten festgelegt. Ist hingegen $\mathbf{Q}(i,:)_k$ mit $\theta_{k,2}$ die beste Wahl für $\mathbf{Q}(i,:)$ und $\boldsymbol{\theta}(i)$ gewesen, so wird der Knoten als UND-Knoten festgelegt.
Nun wird die passende Verlustfunktion minimiert. Dazu wird das Newton Verfahren zur Bestimmung von Nullstellen verwendet. Die Verlustfunktion wird dabei an der aktuellen Stelle, mittels Taylor, linear approximiert. Anschließend wird die Nullstelle der Approximation in negative Gradientenrichtung analytisch bestimmt. Diese Nullstelle der Approximation stellt den nächsten Iterationsschritt dar.

$$J_i\big(\mathbf{Q}(i,:)_{\text{neu}}\big) \approx J_i\big(\mathbf{Q}(i,:)_{\text{alt}}\big) + \Big(\mathbf{Q}(i,:)_{\text{neu}} - \mathbf{Q}(i,:)_{\text{alt}}\Big) \cdot \nabla J_i\big(\mathbf{Q}(i,:)_{\text{alt}}\big)$$

$$\mathbf{Q}(i,:)_{\text{neu}} = \mathbf{Q}(i,:)_{\text{alt}} - \frac{J_i(\mathbf{Q}(i,:)_{\text{alt}})}{||\nabla J_i(\mathbf{Q}(i,:)_{\text{alt}})||^2} \cdot \nabla J_i\big(\mathbf{Q}(i,:)_{\text{alt}}\big)$$

In jedem Schritt wird $\mathbf{Q}(i,:)$ noch 40 mal binarisiert (ähnlich wie Typbestimmung) und die Genauigkeit auf den Daten bestimmt. Im Endeffekt wird $\mathbf{Q}(i,:)$ nach der höchsten Testgenauigkeit gewählt. Dass in jedem Minimierungsschritt viele verschiedene Binarisierungen getestet werden, zeigt, wie wenig der Algorithmus die Verlustminimerung nutzt.
Zudem wird die Minimierung gegebenen Falls einige Male für leicht veränderte Startwerte durchgeführt. Dies dient einerseits dazu lokealen Minima zu entkommen. Andererseits wird die Typenbestimmung jedes Mal durchgeführt und somit zuverlässiger der passende Knotentyp gefunden.

2.4 Eigene Experimente

In diesem Abschnitt sind die Experimente von Sato et al. [3] mit künstlich generierten Daten nachgestellt.
Dazu wurde ein zufälliges UND/ ODER boolesches Netzwerk mit N Knoten und uniform 1-5 rein zufälligen Argumenten pro Übergangsfunktion erstellt. Die Trainings (L viele) bzw. Testdaten ergeben sich aus rein zufälligen Startzuständen und daraus berechneten Nachfolgezuständen. In einigen Experimenten wurden Bits der Nachfolgezustände mit einer Wahrscheinlichkeit p invertiert, um Messfehler zu simulieren.
Die nachfolgende Tabelle zeigt die Ergebnisse dieser Experimente. An den letzten beiden Zeilen erkennt man, dass bei Knoten, für die es eine Kon- bzw. Disjunktion gibt, die die Daten fehlerfrei erklärt (hier alle), diese sehr schnell gefunden wird. Dies liegt nicht zuletzt daran, dass das Newton Verfahren vergleichsweise schnell gegen Nullstellen (hier ein glob. Minimum) konvergiert.
Gut zu erkennen ist auch, dass wenigen Daten im Vergleich zur Knotenanzahl, zu vielen Lösungen führen, sodass das echte Netzwerk nicht unbedingt gefunden werden kann. Im Artikel wurden selbige Experimente mit $L = 100$ und N bis zu 5000 durchgeführt, allerdings nur die Trainingsdauer und Testgenauigkeit betrachtet. Mit Trainings- bzw. Testgenuaigkeit ist die entsprechende durchschnittliche Genauigkeit der Knoten bezeichnet.

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 gefunden	1	1	2	5	1
maximal benötigte Newtonschritte bei finalem Typ	2	3	3	11	4

Abb. 6: Ergebnisse eigener Experimente für geringe Variablenanzahlen.

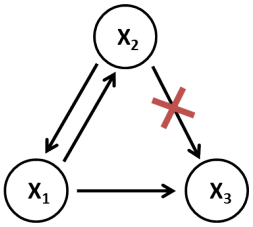
2.5 Experimente mit echten Daten

Sato et al. [3] testeten ihr Modell auch an echten Daten. Dazu wurden N=10928 mRNA Sequenzen der Backhefe (Saccharomyces cerevisiae) identifiziert und deren Konzentration über 3 Zellzyklen hinweg gemessen. Daraus ergaben sich, nach Binarisierung, 40 beobachtete Zustandsübergänge (Datenpunkte). Fokussiert wurde sich dabei auf die Skalierbarkeit des Modells. So wurde nach 29 stündigem Training eine Trainingsgenauigkeit von $87,3\%$ erreicht.
In einem weiteren Experiment testeten sie das Modell mittels 4 facher Kreuzvalidierung (Testblöcke der Größe 10). Dabei erzielten sie eine durchschnittliche Testgenauigkeit von nur $40,9\%$, schlechter als das erwartete Ergebnis bei reinen Zufallsvorhersagen. Die Trainingsgenauigkeit blieb dabei ähnlich wie zuvor.
Mit einigen Maßnahmen gegen Overfitting, hauptsächlich der Beschränkung der ANzahl der Argumente pro Übergangsfunktion auf 4, erzielen Sate et al. [3] eine durchschnittliche Testgenauigkeit bezüglicher Kreuzvalidierung von $84,3\%$. Die Aussagekraft des Experiments ist ohne genauere Betrachtung des Datensatzes und Angaben der Autoren allerdings schwer zu bewerten. Z.B. ist bei nur 40 Übergängen auf fast 11000 Variablen damit zu rechnen, dass für viele Variablen zahlreiche Lösungen existieren. Deshalb wäre es z.B. wünschenswert zu wissen, wie viele der gefundenen Übergangsfunktionen fehlerfrei auf den Daten agieren.
Auch ist unklar ob und wie viele der betrachteten mRNA Sequenzen (nahezu) trivial sind. Damit ist gemeint, dass sie entweder dauerhaft in hoher oder niedriger Konzentration vorkommen. Ist z.B. unter den Laborbedingungen nur Glukose als Nahrungsmittel für die Hefe verfügbar, so ist damit zu rechnen, dass Gene zur Umsetzung von anderen Kohlenhydraten dauerhaft inaktiv sind. Solche Variablen/Gene sind leicht durch z.B. $X_i(t+1) = X_i(t)$ beschrieben und könnten die Ergebnisse des Experiments besser erscheinen lassen, als sie es sind.

3 Probleme

3.1 Korrelation vs. kausaler Zusammenhang

Ein grundlegendes Ziel des entwickelten Modells ist es die Übergangsfunktionen explizit zu erklären (kausale Zusammenhänge finden). Für eine Blackbox Klassifikation gäbe es zahlreiche etablierte Alternativen.
Ein Problem von Klassifikationsalgorithmen allgemein ist, dass die Korrelation zwischen den Variablen genutzt wird, um die Genauigkeit auf den Daten zu maximieren. Aus hoher Korrelation folgt aber nicht unbedingt ein kausaler Zusammenhang. Deshalb bleibt für Variablen ohne eindeutige Lösung fraglich, in wie fern die gefundene Übergangsfunktion wirkliche kausale Zusammenhänge erklärt. Dies gilt insbesondere für den hier behandelten Algorithmus, da nicht alle möglichen Übergangsfunktionen in Betracht gezogen werden.
Das Beispiel zeigt eine Situation, in der mehrere Lösungen für f_3 existieren, sich die Zustandswechsel von X_3 aus der Korrelation mit X_2 erklären lassen, dies aber keinen kausalen Zusammenhang impliziert.



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

$$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \Longrightarrow \quad X_3(t+1) = X_2(t)$$

Abb. 7: Übergänge mittel Korrelation erklären. Oben: Zugrunde liegendes Netzwerk. Dabei gesondert markiert: Kein direkter Einfluss von X_2 auf X_3.

3.2 Daten

Wie im Abschnitt der eigenen Experimente bereits gesehen, bedeuten zu wenige Daten bei zu vielen Variablen (unterbestimmtes Gleichungssystem), dass mit einer Vielzahl an Lösungen zu rechnen ist. Dies ist vor allem für die Anwendung mit echten Lebewesen ein Problem, da die Datengewinnung extrem Aufwändig ist.
Ein weiteres Problem mit den Daten ist die Art ihrer Gewinnung. In der Praxis werden die Zustandsübergänge sequenziell beobachtet. D.h. Nachfolgezustand 1 ist zugleich Startzustand 2 usw.. Dies hat unter anderem eine geringe Variabilität der Daten zu Folge, welche den zuvor genannten Effekt noch verstärkt.
Eine denkbare, aber aufwändige, Lösung wäre es die Zustände künstlich zu verändern. Dazu könnte man die mRNA Sequenzen künstlich synthetisieren und dem Organismus in gewünschter Konzentration zuführen.
Im Beispiel entstehen durch die sequenzielle Datengewinnung nur 2 verschiedene Übergänge, da sie einen Zyklus der Länge 2 bilden.

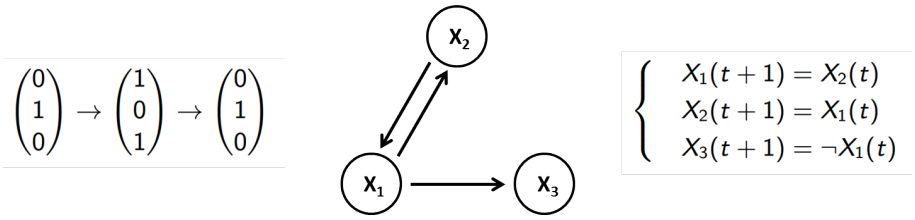
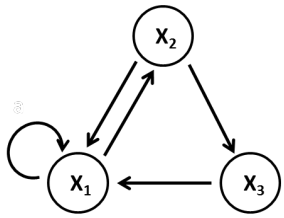


Abb. 8: Problem der sequenziellen Datengewinnung.

3.3 Overfitting

Teilweise schon im vorherigen Abschnitt angesprochen, ist Overfitting ein Problem des Modells.
Bei vielen Variablen (und wenig Daten) besteht eine hohe Chance, dass die Übergänge einer Variable von anderen Variablen nur durch Zufall erklärt werden. Das bedeutet, dass in diesen Fällen häufig viele Nullstellen der Verlustfunktion existieren. Hier würde schon ein einziger Messfehler ausreichen, um die echte Übergangsfunktion aus den Nullstellen zu entfernen. Das verwendete Newton Verfahren konvergiert vergleichsweise schnell gegen eine Nullstelle, sofern diese existiert und nahe genug an ihr gestartet wird. In den Experimenten war dies gut zu beobachten. Deshalb wird höchst wahrscheinlich nicht die echte Übergangsfunktion vom Algorithmus gefunden.
Der zufälligen Erklärungskraft der zahlreichen Variablen kann entgegengewirkt werden, indem die maximale Anzahl an Argumenten der Übergangsfunktion beschränkt wird. Dadurch muss sich auf die wesentlichen Variablen beschränkt werden. Die Autoren nutzen diesen Ansatz in Kombination mit Lernwiederholung für Knoten mit sehr niedriger Trainingsgenauigkeit, da sie in diesen Fällen vermuten, dass die Argumentanzahl zu gering angesetzt war.
Im Beispiel wird die Übergangsfunktion von X_1 betrachtet. Der Messfehler ist rot markiert.



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

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

Lösungen (ohne Einschränkung):

- $X_1(t+1) = X_2(t) \vee \neg X_3(t)$
- $X_1(t+1) = X_1(t) \vee X_2(t) \vee \neg X_3(t)$

Beschränkung auf maximal 1 Argument:

- $X_1(t+1) = \neg X_3(t)$
- eindeutige Übergangsfunktion mit minimaler Fehlerzahl auf Daten

4 Fazit

Das Ziel der Autoren war es ein skalierbares Modell zum Erlernen von booleschen Netzwerken zu entwickeln. Allein wegen der erreichten Skalierbarkeit, ist ihr Modell forschungsrelevant, z.B. für Genom weite Genregulation.
Dennoch bleiben einige Probleme und Fragen offen. So wird die Einschränkung auf UND/ODER boolesche Netzwerke weder logisch noch empirisch legitimiert. Es bleibt also bis auf weiteres Unklar, inwieweit die Ausdrucksfähigkeit des Modells eingeschränkt ist.
Auch die Verwendung des Newton Verfahrens zur Nullstellensuche der Verlustfunktion ist ungewöhnlich und in der Anwendung auf echte Daten fragwürdig. So kann das Verfahren bei Absenz einer Nullstelle nicht konvergieren. Existieren hingegen sehr viele Nullstellen der Verlustfunktion, so ist die Aussagekraft der Daten bereits fragwürdig.
Zudem gehen die Autoren in ihren Experimenten nicht auf das Problem der geringen Datenmenge ein und geben auch keine Messdaten an, mit denen man diesbezüglich Rückschlüsse ziehen könnte.
Außerdem werden Umwelteinflüsse nicht in Betracht gezogen. Diese kann mittels Parametern modelliert werden. Falls nur bestimmte Umwelteinflüsse bei der Datengewinnung verwendet werden, so kann die Einschränkung in den betrachteten Genen sinnvoll sein (Beispiel: Gene der Imunreaktion bei Abwesenheit von Krankheitserregern).
Diese Fragen und Probleme bleibt es zu klären bzw. lösen. Die Autoren bezeichnen ihren Algorithmus selbst noch als simpel und seine Weiterentwicklung ist nur eine Frage der Zeit. Im Punkt der Skalierbarkeit ist der Ansatz von Sato et al. [3] sich auf UND/ODER booleschen Netzwerke zu beschränken ein Erfolg. Vergleichbare Algorithmen [2] ohne diese Einschränkung sind bisher nur für bis zu ca. 100 Variablen nutzbar.

Literatur

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

[2] H. Lähdesmähki, I. Shmulevich, O. Y.-H. On learning gene regulatory networks under the boolean network model. *In Machine Learning 52* (2003), 147—-167.

[3] 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.