Data requirements of reverse-engineering algorithms

Winfried Just

Department Mathematics, Ohio University,
Athens, Ohio, U.S.A.

This material is based on work done while the author was visiting the Mathematical Biosciences Institute (MBI) at OSU in Columbus, Ohio. The author acknowledges support by the National Science Foundation under Agreement No. 0112050.

The problem

- **Collecting data** on biochemical networks remains **expensive**; reverse-engineering algorithms typically rely on **small data sets.**

- This makes reverse-engineering problems **vastly underdetermined.**

- Reverse-engineering algorithms usually **select and return one model** that is consistent with the data.

Our goal

Develop a theory of data requirements for the most popular reverse-engineering algorithms.
Ideally, such a theory would:

- Predict the probability that a given algorithm infers the correct, or an approximately correct, model of the network from a given data set.

- Provide guidelines for the most promising choice of input parameters.

Outline of the talk

- **Part 1:** General considerations

- **Part 2:** Proof of principle: Theorems about the Laubenbacher-Stigler algorithm

Why a lot of data may be needed

- Validation of the modeling paradigm (stochastic *vs.* continuous *vs.* discrete).

- Dealing with noise in the data (this is essentially statistics).

- **Model selection under the assumption of a suitable modeling paradigm and clean data.**

One modeling paradigm

- There are n chemical species with concentrations $\vec{x} = [x_1, \ldots x_n]^T$.

- $x_i \in F$, for some finite set F.

- Concentration levels change simultaneously in discrete time steps so that $x_i(t+1) = h_i(\vec{x}(t))$.

One reverse-engineering problem

Given a data set

$$D = \{< \bar{x}(t), y(t) >: \ t \in [m]\}$$

such that $y(t) = h_i(\vec{x}(t))$ for all $t \in [m]$, find the unknown function h_i.

The function h_i will henceforth be called the i-**th regulatory function;** a reverse-engineering algorithm returns a **model** h_i^* of h_i consistent with the data.

We will **not** in general assume that the data are time series data.

A caveat

It is not clear whether the h_i's should be reverse-engineered one at a time.

Quality of the solution

- **Perfectionist:** Accept h_i^* only if $h_i^* = h_i$.

- **Topologist:** Accept h_i^* if it depends on the same input variables as h_i.

- **Approximist:** Accept h_i^* if it correctly predicts the network response to most concentration vectors that have not been studied in the lab.

No Free Lunch Theorem

For a data set D of size m there are $|F|^{|F|^n - m}$ different solutions h_i^* for the reverse engineering problem. Model selection is (at least implicitly) based on a prior distribution $\ominus$ of biologically plausible functions h_i. If $\ominus$ is the uniform prior, then no reverse-engineering algorithm can do better than randomly picking a model h_i^*, at least if the experimental protocol by which the data were obtained is ignored.

Some cheaper lunches exist

Even with a uniform prior Θ, one can sometimes slightly increase the chances of finding the correct model if the data collection protocol is known.

How come? If data were collected from systems that had been running unperturbed for a long time, one can assume that data input vectors are part of an attractor.

Using data collection protocol information

- Reverse-engineering all h_i simultaneously may be beneficial.

- A formalism for data collection protocols should indicate which knockout, overexpression, or perturbation experiments are feasible.

- Guidelines for data collections may result as an additional bonus.

In the absence of a theory

We imagine an experimenter who randomly samples (with replacement from a uniform distribution) data inputs $\vec{x}(t)$ from F^n and takes measurements of the system response $y(t)$. Initial data sets $D_m = \{< \vec{x}(t), y(t) >: t \in [m]\}$ are analyzed with the same algorithm. Let $\lambda(h_i)$ be the smallest m such that the algorithm returns $h_i^* = h_i$. We can prove esitmates for the expected value $E(\lambda(h_i))$.

Towards meaningful priors

- Performance of reverse-engineering algorithms should be assessed in the context of a biologically meaningful prior $\ominus$.

- Regulatory functions tend to **depend on relatively few inputs.**

- Boolean gene regulatory functions tend to be so-called **nested canalyzing functions.**

Two algorithms

- The **LS-algorithm** of Laubenbacher and Stigler, 2004. A computational algebra approach to reverse engineering of gene regulatory networks. *J. Theor. Biol.* 229, 523–537.

- The **LS-algorithm with preprocessing** of Jarrah, Laubenbacher, Stigler, and Stillman, 2007. Reverse engineering of polynomial dynamical systems. *Adv. in Appl. Math.* to appear.

Input parameters

In these algorithms, F is assumed to be a finite field, such as $\mathbb{F}_2 = \{0, 1\}$. The algorithms use so-called *term orders* of monomials in the polynomial ring $F[x_1, \ldots, x_n]$ as input parameters. The most important types of term orders are **lex orders** and **graded orders.** In the LS-algorithm, the term order is entirely user-defined, in the LS-algorithm with preprocessing, the user specifies the type of term order (*e.g.*, lex or graded) and the algorithm determines the specific term order based on the data.

The LS-algorithm with graded orders

Theorem 1. *Let* $F = \{0, 1\}$ *and let* h_i *be a nested canalyzing function that depends on* k *variables. If the LS-algorithm is run with a* **randomly chosen graded** *term order, then* $E(\lambda(h_i))$ *is* $\Theta(n^k)$*. If the LS-algorithm is run with an* **optimally chosen graded** *term order, then* $E(\lambda(h_i))$ *is* $\Omega(n^{k-1})$*.*

The LS-algorithm with lex orders

Theorem 2. *Suppose h_i depends on exactly $k > 0$ variables.*

(i) If the LS-algorithm is run with a **randomly chosen lex** *order, then $E(\lambda(h_i))$ is $\Omega(c^n)$ for some constant $c > 1$.*

(ii) If the LS-algorithm is run with an **optimally chosen lex** *order, then $E(\lambda(h_i))$ is $O(|F|^k k \ln |F|)$.*

The LS-algorithm with preprocessing

Theorem 3. *Suppose h_i depends on at most k variables. If the LS-algorithm with preprocessing is run for lex orders, then $E(\lambda(h_i))$ is $O(|F|^{2k}\, 2k\, ln\, n)$. Moreover, if h_i is nested canalyzing, then $E(\lambda(h_i))$ is $O(|F|^{k+1}\, (k+1)\, ln\, n)$.*