17 Reverse engineering polynomial dynamical systems

Recall our goal: Construct PDS's from a given set of data:

Input states $\vec{s}_1, \dots, \vec{s}_m \in \mathbb{F}^n$
Output states $\vec{t}_1, \dots, \vec{t}_m \in \mathbb{F}^n$ with $F(\vec{s}_i) = \vec{t}_i$.

Goal: Construct the model space $F_1 \times F_2 \times \dots \times F_n$ of all PDS's $f = (f_1, \dots, f_n)$ that fit the data.

That is, $f(\vec{s}_i) = (f_1(\vec{s}_i), \dots, f_n(\vec{s}_i)) = (t_{i1}, \dots, t_{in}) = \vec{t}_i$.

Any such f is called a model.

Subproblem: For each j, find all polynomials f_j satisfying
$f_j(\vec{s}_1) = t_{1j}$, $f_j(\vec{s}_2) = t_{2j}$, ..., $f_j(\vec{s}_m) = t_{mj}$.

Let $p = \text{char } \mathbb{F}$ = smallest k s.t. $\underbrace{1 + 1 + \dots + 1}_{k \text{ times}} = 0$.

Fact: p must be prime (if $|\mathbb{F}| < \infty$).

Fermat's little theorem: $a^p \equiv a \pmod p$, or equivalently,
for each $a \in \mathbb{F}$, $a^p = a$.

2

Corollary: $x_i^p = x_i$.

Thus, all polynomials have maximum degree $p-1$.

Technically, this means they are in the quotient ring

$$R := \mathbb{F}[x_1, \ldots, x_n] / \langle x_1^p - x_1, \ldots, x_n^p - x_n \rangle.$$

Let $F_j = \{ f_j \in R : f_j(\vec{s}_i) = \vec{t}_i \text{ for all } i = 1, \ldots, n \}$.

= set of polynomials that fit the data for node j.

Theorem $\boxed{F_j = f_j + I}$ where $I = \{ f : f(\vec{s}_i) = 0 \ \forall i = 1, \ldots, m \}$

= set of polys. that vanish on each $\vec{s}_i$.

So, $F_j = \{ f_j + h : h \in I \}$ where f_j is any one particular poly. that fits the data.

Recall: Compare this to:

- Solving $A\vec{x} = \vec{b}$: $\vec{x} = \vec{x}_p + NS(A)$
- Solving a linear ODE: $y = y_p + y_h$

What is I?

Define $I(\vec{s}_i) = \langle x_1 - s_{i1}, \ldots, x_n - s_{in} \rangle$

$= \{ g_1(\vec{x})(x - s_{i1}) + \ldots + g_n(\vec{x})(s_n - s_{in}) \}$

$=$ all polys f_i s.t. $f_i(\vec{s}_i) = 0$.

Then $I = \bigcap_{i=1}^{m} I(\vec{s}_i)$.

To find I: Use a computational algebra software package, like Sage.

* How to Find f_j:

There are many algorithms (e.g, Lagrange interpolation).

Here's one (based on the Chinese Remainder theorem for rings).

Output: Function $f_j(\vec{x})$ s.t. $f_j(\vec{s}_i) = t_{ij}$ for each $i = 1, \ldots, m$.

Algorithm: For each $j = 1, \ldots, n$, we'll construct a function $r_j(\vec{x})$

s.t. $r_j(\vec{x}) = \begin{cases} 1 & \vec{x} = \vec{s}_j \\ 0 & \vec{x} \neq \vec{s}_j \end{cases}$.

This works: $r_j(\vec{x}) = \prod_{\substack{k=1 \\ k \neq j}}^{m} b_{jk}(\vec{x})$

where $b_{jk}(\vec{x}) = (s_{j,\ell} - s_{k,\ell})^{p-2}(x_\ell - s_{k,\ell})$,

and ℓ is the first coordinate s.t. $\vec{s}_j \neq \vec{s}_k$.

Then, define $f_j(\bar{x}) = t_{ij}\vec{r}_i(x)$

$$= t_{1j}\vec{r}_1(\vec{x}) + t_{2j}\vec{r}(\vec{x}) + \cdots + t_{mj}\vec{r}(\vec{x})$$

Example: Consider the 3-node system over $\mathbb{Z}_5$ s.t.

$\bar{s}_1 = (2,0,0) \longrightarrow (4,3,1) = \bar{t}_1$

$\bar{s}_2 = (4,3,1) \longrightarrow (3,1,4) = \bar{t}_2$

$\bar{s}_3 = (3,1,4) \longrightarrow (0,4,3) = \bar{t}_3$.

$$200 \downarrow 431 \downarrow 314 \downarrow 043$$

This is called a time series.

Note: $\bar{s}_1$ differs from $\bar{s}_2$ & $\bar{s}_3$ in the $\boxed{l=1}$ coordinate, so the same l will work for f_1, f_2, f_3.

Find $f_1(x_1, x_2, x_3)$

First, compute the r-polynomials.

$r_1(\bar{x}) = b_{12}(\bar{x})\, b_{13}(\bar{x})$

$b_{12}(\bar{x}) = (s_{11} - s_{21})^3(x_1 - s_{21}) = (2-4)^3(x_1 - 4) = -8(x_1+1) = 2x_1+2$

$b_{13}(\bar{x}) = (s_{11} - s_{31})^3(x_1 - s_{31}) = (2-3)^3(x_1-3) = -x_1+3 = 4x_1+3$

$r_1(\bar{x}) = b_{12}(\bar{x})\, b_{13}(\bar{x}) = 3x_1^2 + 4x_1 + 1$

$\underline{\underline{r_2(\vec{x})}} = b_{21}(\vec{x})\, b_{23}(\vec{x})$

$b_{21}(\vec{x}) = (s_{21} - s_{11})^3 (x_1 - s_{11}) = (4-2)^3 (x_2 - 2) = 8(x_1+3) = 3x_1+4$

$b_{23}(\vec{x}) = (s_{21} - s_{31})^3 (x_1 - s_{31}) = (4-3)^3 (x_1 - 3) = x_1 + 2$

$r_2(\vec{x}) = b_{21}(\vec{x})\, b_{23}(\vec{x}) = (3x_1+4)(x_1+2) = 3x_1^2 + 3$

$\underline{\underline{r_3(\vec{x})}} = b_{31}(\vec{x})\, b_{32}(\vec{x})$

$b_{31}(\vec{x}) = (s_{31} - s_{11})^3 (x_1 - s_{11}) = (3-2)^3 (x_1 - 2) = x_1 + 3$

$b_{32}(\vec{x}) = (s_{31} - s_{21})^3 (x_1 - s_{21}) = (3-4)^3 (x_1 - 4) = -(x_1 - 4) = 4x_1 + 4$

$r_3(\vec{x}) = b_{31}(\vec{x})\, b_{32}(\vec{x}) = (x_1+3)(4x_1+4) = 4x_1^2 + x_1 + 2$

$f_1(x_1, x_2, x_3) = t_{11}\, r_1(\vec{x}) + t_{12}\, r_2(\vec{x}) + t_{13}\, r_3(\vec{x})$

$= 4(3x_1^2 + 4x_1 + 1) + 3(3x_1^2 + 3) + 0(4x_1^2 + x_1 + 2)$

$= \boxed{x_1^2 + x_1 + 3}$

Since the same $\ell = 1$ works for f_2 & f_3,

$f_2(x_1, x_2, x_3) = t_{21}\, r_1(\vec{x}) + t_{22}\, r_2(\vec{x}) + t_{23}\, r_3(\vec{x})$

$= 3(3x_1^2 + 4x_1 + 1) + 1(3x_1^2 + 3) + 4(4x_1^2 + x_1 + 2)$

$= \boxed{3x_1^2 + x_1 + 4}$

$$f_3(x_1, x_2, x_3) = t_{31}\, r_1(\bar{x}) + t_{32}\, r_2(\bar{x}) + t_{33}(\bar{x})$$

$$= 1(3x_1^2+4x_1+1) + 4(3x_1^2+3) + 3(4x_1^2+x_1+2)$$

$$= \boxed{2x_1^2+2x_1+4}$$

Our "particular solution" is

$$f = (f_1, f_2, f_3) = (x_1^2+x_1+3,\ 3x_1^2+x_1+4,\ 2x_1^2+2x_1+4)$$

The model space $F_1 \times \dots \times F_n = f + (I \times \dots \times I)$

$$= (f_1 + I, \dots, f_n + I).$$

<u>Model selection</u> The next task is to pick the "best" model from the model space, Ideally, one with "predictive power."

<u>One approach</u> (sketch): Given a set $f_j + I$, find a polynomial that has no terms in I.

That is, compute the remainder of f_j upon division of elts in I

Recall that this is "well-defined" if we have a Gröbner basis.

Fix a monomial ordering

Compute a Gröbner basis $\mathcal{G}$

Compute the remainder of f_j upon division by all elts. in I.

This is called the <u>normal form</u> of f_j w.r.t. $\mathcal{G}$, denoted

$NF(f_j, \mathcal{G})$.

<u>Output</u>: $f = (NF(f_1, \mathcal{G}), \ldots, NF(f_n, \mathcal{G}))$.