Interview Task Report

Tra My Nguyen

November 2025

1 Problem formulation

- Each system can be defined as $X_{t+1} = \dot{X}_t + X_t$ where $X_t = (x_{1t}, x_{2t}, ..., x_{nt})$ with n variables
- The governing equations can be defined as $\dot{X}_t = Ak(X_t)$ where
  - k is a kernel transformation resulting in $X_k = (\theta_1(X), \theta_2(X), ..., \theta_m(X))$
  - A is a matrix of numeric values of shape (n, m)
- So the problem can be broken down into 2 tasks: finding the kernel k and the matrix A that captures the evolution of the systems, which is in itself a time series $X_{t+1} = Ak(X_t) + X_t = Ak(X_t)$

  System 1 has $k_1 = (x, y, xy)$ and $A_1 = \begin{pmatrix} 2 & 0 & -0.1 \\ 0 & -0.5 & 0.075 \end{pmatrix}$

  System 2 has $k_2 = (1, x, y, z, xy, zy, y^3)$ and $A_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & -0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0.25 & 0 & 1 & 0 & 0 & -2 \end{pmatrix}$
- **Program synthesis** can be used to find k and we can perform regression on each form of k to find matching matrix A (manual implementation)
- **Linear regression with lagged features** to fit model to train data and choose k and A that give score $r > 0.99$ on test data (use sklearn)

2 Implementation

Components of synthesizer, choices of component? bottom up enumeration with observational equivalence using system evolution data as examples

- program bank: each program is a 3-tuple (set of forms (each form is either a variable or a product of variables), size of program, round of discovery) e.g. `{'x', 'x*y'}` represents $k = (x, xy)$ of size 5
- dict of basic operations
  - variables, no input required

  - `add` adds new terms to set, because coeffs will be learned later, so keep only the simplest forms, requires 2 inputs
    e.g. `add({'x'},{'x','y'}) = {'x','y'}`
  - `multiply` broadcasts product of sets, requires 2 inputs
    e.g. `multiply({'x'},{'x','y'}) = {'x*x','x*y'}`

- iterator calls combinator to get new programs, fit new programs to data and check if any program can satisfy min test score, stop if found, start new round if not
- combinator finds and combines
  - **combinations** of programs in latest round together
  - **products** of programs in latest round and combinations of programs in older rounds
- redundancy check
  - new program already exists in program bank
  - new program is subset of a program in bank (because kernel of higher dimension already fitted to data and unable to satisfy min score)

3 Experiment

Data generation, selection, how many?

- values of variables of 2 systems tend to get really big (order of $\sim 10e^{100}$) or really small ($\sim 10e^{-100}$) really soon for a lot of initial points, at around time step 8-15
- data for regression should contain examples from multiple evolutions that last less than 8 time steps
- 50 inputs for each system, 80% train data and 20% test data
- perform linear regression with correct kernel before trying program synthesis to verify if data is reasonable

Results

- initial points randomized from uniform distribution $[-0.5, 0.5]$
- System 1, inputs = evolutions of 5 initial points over 10 time steps, ground truth found by the synthesizer

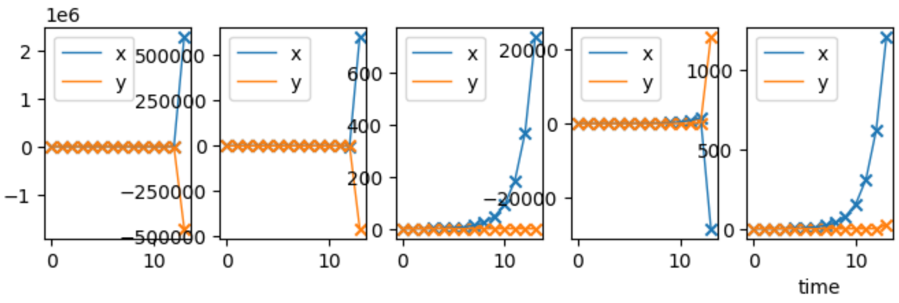

Figure 1: System 1 evolution

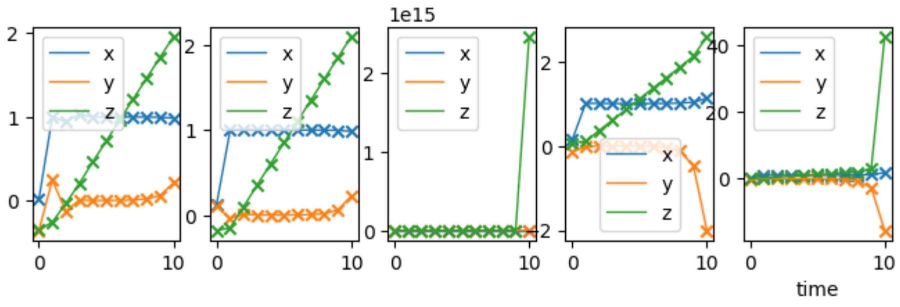

Figure 2: System 2 evolution

- System 2, inputs = evolutions of 10 initial points over 5 time steps, synthesizer found some other equations

  e.g. $k = (x, xy, xz, yz, z)$ of size 13 and

  $$A = \begin{pmatrix} 0 & -0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0.29347892 & 0.10717984 & -0.21424233 & -0.6936943 & 1.17737706 \end{pmatrix}$$

  gives test score of $\simeq 0.99$

  e.g. $k = (x, x^2, xy, xz, y, yz, z)$ of size 13 and

  $$A = \begin{pmatrix} 0 & 0 & -0.25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0.21880134 & 0.02375237 & 0.10258972 & 0.05411208 & -0.33997678 & 0.05588454 & 0.95088318 \end{pmatrix}$$

  gives test score of $\simeq 0.999$

- Synthesizer fails to find ground truth or equations that gives min test score 0.9999 within 3 rounds, at round 4, there are $\simeq$ 3 millions combinations to test, cannot finish running after 45 mins of runtime

Discarded ideas

- program `c` representing a constant as a basic operation
- e.g. `add({'x'},{'y'})` returns `{'x+y'}`
- favor programs of bigger size to skip over some combinations

  e.g. all programs size 5 are subsets of at least one program result of combinations of programs size 3 (can be proven), all programs size 5 are 'contained' in size 7 $\Rightarrow$ no need to consider programs size 5 and skip to combinations of programs size 7 result of combinations of programs size 3

  problem missing programs size 7 from products of program size 1 and 5

4 Think further

Missing data

- if enough data, delete input data before and output data after missing data
- various methods of imputation or interpolation to fill in the missing data

Noisy data

- if noise is really small like in given example, linear regression should still work well, though there might be bigger effect on terms of higher degree, then input should include only evolutions with very few time steps to avoid big values of variables like aforementioned
- if bigger noise, use a more robust regression model e.g. ridge regression, lasso regression, etc.