Particle Filtering with Polynomial Functors

Eric Redekopp

March 1, 2024

1 PARTICLE FILTERING

Particle Filtering is a Bayesian statistical method for tracking the likely state of an underlying system over time given some observed data. We use particle filtering at the CEPHIL laboratory to model epidemiological systems. The underlying system is represented as a system dynamics (SD) model, which is essentially a system of stochastic differential equations representing the expected change in the system over time. Particle filtering allows us to run many instances of the model at once, and to propagate forward those instances that most closely match the observed data while replacing those that match poorly.

2 DYNAMICAL SYSTEMS IN POLY

Poly gives us a powerful and expressive mathematical language to express dynamical systems and their trajectory over time. A state system represented as a map between polynomial functors has the form $\phi : Sy^S \to p$. S represents the set of all possible system states, and p is another polynomial representing the system's "interface"; that is, the outputs given from the system state at each time point and the possible subsequent inputs given a system state. Since Sy^S is a polynomial comonoid, we can "step through" the system states by repeatedly taking the substitution product of the system with itself.

3 PARTICLE FILTERING IN POLY

We can consider the entire particle filtering algorithm to be a state system and express it as a polynomial morphism. This polynomial has a rather ugly form, which I will introduce here and explain in detail throughout this paper.

$$(List \triangleleft S^N) R \Delta_N S^N y^{(List \triangleleft S^N) R \Delta_N S^N} \xrightarrow{\phi} y^O \tag{3.1}$$

Where:

- S is the set of all possible system states of the underlying SD model.
- N is the number of particles in the filter.
- $(List \triangleleft S^N)$ is the polynomial $List(S^N) = 1 + S^N + (S^N)^2 + (S^N)^3 + \ldots$ representing lists containing N-tuples of states.
- R represents all of the states of a stateful random number generator[1].
- Δ_N is the set of all discrete probability distributions over a set of size N. In this case, we consider the probability at each position to be a particle's weight.
- S^N is an N-tuple of system states or, equivalently, a function $N \to S$.
- O is the set of all possible values of the observed data.

Let's break this down into more manageable pieces. First, consider the component of the system that updates the state for all N at each time step. This sub-system has the form:

$$S^N y^{S^N} \xrightarrow{\psi} y \tag{3.2}$$

$$\psi^\sharp_s : \bullet \mapsto \{d(s_n) | n \in 1, \ldots, N\} \tag{3.3}$$

Where s is the N-tuple of current system states and d is a user-defined stochastic state update function. This is essentially the regular state update function for state systems in Poly, but extended to N simultaneous systems instead of a single one.

Next, consider the component of the system that resamples particles based on the observed data. This sub-system has the form:

$$R\Delta_N S^N y^{R\Delta_N S^N} \xrightarrow{\Psi} y^O \tag{3.4}$$

$$\Psi^\sharp_{r,w,s} : o \mapsto (r^{+N}, w', s') \tag{3.5}$$

$$w' = \text{normalize}(\{w_{\theta(w)_n} \times p(o | s_{\theta(w)_n}) | n \in 1, \ldots, N\}) \tag{3.6}$$

$$s' = \{s_{\theta(w)_n} | n \in 1, \ldots, N\} \tag{3.7}$$

where r = the current state of the random number generator, w = the current weights of the particles $\in \Delta_N$, normalize is the function that normalizes all weights to sum to 1, $\theta(w)$ is the function that chooses an ancestor particle for each updated particle based on its state, and $p(o|s_i)$ is the likelihood function giving the probability of seeing the observed data given the state of particle i.

Finally, the portion $(List \triangleleft S^N)$ keeps track of the particle states at each timepoint for future analysis. These sub-systems combined give us our full system and, most importantly, our map $\phi^\sharp$

$$\phi^\sharp_{l,r,w,s} : o \mapsto ([l; s], r', w', s') \tag{3.8}$$

I am running out of time to explain further, however this full system is essentially the combination of all the above systems, with the slight modification that s' is the updated states of

[1] This might be too much detail for practical purposes, but I'm leaving it in here because my notes from the research week have been considering this to be part of the state and I don't want to make any changes today.

the selected ancestor particles and not simply the updated state of each particle as shown in the state update sub-system.

To run the full particle filter, assuming we have k timepoints of data, we use the combined system $\phi^{\triangleleft k}$ which is the system composed with itself k times. The input to this system is the entire dataset, and after the full run, we take the particle data over time from the list for analysis. There might be some additional work to be done formalizing the idea of an "output" for this system, such as adding a halting state that gives the list of particle values over time. However, that is a task for another day.

4 Future Work

Given additional time, I would like to expand this paper into a more complete and detailed description of this algorithm and how it works as a polynomial morphism. I hope to implement this idea in the future within the Algebraic Julia framework. Further, I would like to expand this idea to Particle MCMC by defining a Poly-based implementation of MCMC and finding a way to use this particle filter as a likelihood function for it. More broadly, I am very interested in using Poly to represent complex algorithms as tensor products of simpler polynomial morphisms to give them a strong mathematical basis and to allow the modular construction of complex programs through small, easy-to-understand components.