Dynamical systems and ecological modeling

Matt Guay

Maryland Mathematical Modeling Contest

October 9th, 2014

Ecological dynamics

- Interactions of organisms in natural environments - pretty complicated.
- To understand some facet of an ecosystem, we can use mathematical models.
- Objective: A good model is simple (abstracts away irrelevant detail) but not too simple (captures phenomena of interest).
- First step: Know what you want to model, and what you don't!

Predators and prey

- Start simple! Build the simplest possible model before worrying about elaborations.
- For us: One predator species, one prey.
- We'll investigate two possible models:
- Ordinary differential equation model: Track only population sizes that evolve according to an ODE.
- Discrete dynamic model: Explicitly simulate a number of organisms moving, predating, reproducting, etc.
- Both can be understood as types of dynamical systems.

Dynamical systems

- For our purposes, a state space $\mathcal{X}$ is a finite collection of state variables $\boldsymbol{x} = (x_i)_{i=1}^{M}$, each taking values in a (discrete or continuous) domain S (i.e. $\mathcal{X} = S^M$).
- Dynamical systems are functions on a state space which change with time $t \in \mathcal{T}$. The time domain $\mathcal{T}$ may be continuous ($\mathcal{T} = \mathbb{R}$) or discrete ($\mathcal{T} = \mathbb{N}$).
- A **dynamical system** on this state space evolves according to an evolution function $\Phi : \mathcal{X} \times \mathcal{T} \to \mathcal{X}$ obeying certain properties. See e.g. Wikipedia for the full definition.
- Important specific examples include autonomous ODEs, autonomous difference equations, and cellular automata.

Ordinary differential equations

- Continuous-time, continuous-space dynamical systems form a subset of ordinary differential equations. In this case, $\mathcal{X} = \mathbb{R}^M$, $\mathcal{T} = \mathbb{R}$, and the state variables evolve as a function $\boldsymbol{x}(t)$ in $\mathbb{R}^M$ satisfying the ODE

$$\dot{\boldsymbol{x}} = F(\boldsymbol{x})$$
$$\boldsymbol{x}(0) = \boldsymbol{x}_0$$

  for a continuous (or better) function F and initial state $\boldsymbol{x}_0$.
- This ODE is autonomous since F does not depend on t.

Ecological ODEs

- ODEs are most easily applied to modeling statistics of ecological populations.
- Example: Track population sizes, no other details of animal populations.
- We will consider a two-population model, keeping track of two population sizes: the Lotka-Volterra model.
- This is simple and analytically tractable, but abstracts heavily away from actual ecology.

Lotka-Volterra history

- First developed by Vito Volterra ca. 1926 to explain the variances in fish catches in the Adriatic Sea.
- Four important model assumptions:
  - The prey population grows exponentially in the absence of predation.
  - The predator population decreases exponentially in the absence of prey.
  - Predators reduce prey population growth rate, proportional to both the predator and prey populations.
  - Prey increases the predator population growth rate, proportional to both the predator and prey populations.

Lotka-Volterra equation

- Two state variables $(x, y) \in \mathbb{R}^2$. Prey population is x, Predator population is y.
- Four parameters: α - prey growth rate. β - prey predation effect. γ - predator population decay rate. δ - predator predation effect.
- **Lotka-Volterra equation** (LVE): Given initial x_0 and y_0, $x(t)$ and $y(t)$ satisfy

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \delta xy - \gamma y.$$

Lotka-Volterra equation

- One can use the various tools of dynamical systems theory to analyze the behavior of $x(t)$ and $y(t)$. Spoiler: They oscillate, or $y(t) \to 0$ and $x(t)$ grows unboundedly, or both populations go to 0.
- More important for mathematical modeling is the ability to numerically solve the equations. LVE generalizations can remain numerically solvable even when not analytically tractable (more effects, more species, etc.).

Solving LVE numerically

- Numerical ODE solvers are a fantastically useful set of tools available for most programming languages.
- For an arbitrary ODE, numerical simulation may be extremely difficult. There is tons of literature on the ways to efficiently numerically solve different classes of ODEs.
- For the M3C/MCM, knowing how to use these tools is crucial to building an ODE model.
- **Model example**: Solving LVE using *ode45* and MATLAB.

And now for a

- MATLAB break -

ODE model weaknesses

- List some!

Discrete modeling approaches

- We can revisit the way we model predator-prey interactions, and simulate the organisms instead of tracking population statistics.
- ODE models look at dynamics in a low-dimensional state space (in the LVE case, $\mathbb{R} \times \mathbb{R}$)
- Idea: Let the state space correspond to a high-dimensional physical space (sorta).
- States are discrete objects in physical space (sorta).
- Discrete time corresponding to iterative state updates.
- Simplest approach here: **cellular automata** (CA).

Cellular automata

- Underlying state space $\mathcal{X}$ is a grid of M spatial "cells" $\boldsymbol{x} = \{x_i\}_{i=1}^{M}$ (though other spatial graphs work, too).
- The possible states are a small, finite set S, e.g. $S = \{0, 1\}$, $S = \{\text{red, blue, green}\}$, $S = \{\text{fox, rabbit}\}$.
- Time variable $t \in \mathcal{T} = \mathbb{N}$.
- State evolution can be defined by a discrete difference equation, but it is often useful to use a transition map instead.

Cellular automata

- Transition maps: in general,

$$\boldsymbol{x}(t+1) = F(\boldsymbol{x}(t)),$$

  where the transition map F is a function (deterministic CA) or a stochastic process (stochastic CA) taking values in S.
- Commonly, $x_i(t+1)$ depends only on $x_i(t)$ and $x_j(t)$ for x_j in a neighborhood $N(x_i)$ of x_i.

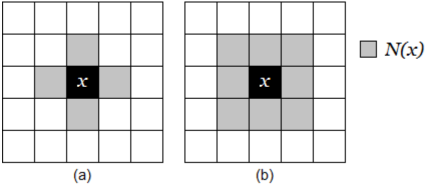


Figure: Two common neighborhoods; (a) Von Neumann and (b) Moore.

A predator-prey CA model

Assumptions:

1. Two "animals": fox (predator) and rabbit (prey).
2. Foxes can *move*, *die*, *eat*, and *reproduce* with some probabilities.
3. Rabbits can *move*, *die*, and *reproduce* with some probabilities.

Setup:

- State space is an $N \times N$ grid.
- States are *empty* (E), *fox*, (F), *rabbit* (R).
- The transition map is stochastic and best described algorithmically, using Moore neighborhoods.

The transition map

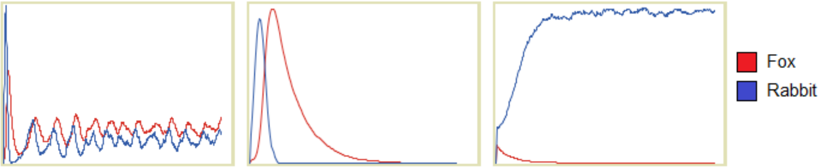
Adapted from Hawick & Scogings, 2010

For each time step $t+1$
  For each cell x (chosen in random order)
    Choose y in neighborhood $N(x)$ at random
    If $u_t(x) = F$ and $u_t(y) = R$
      $u_{t+1}(x) = E$, $u_{t+1}(y) = F$ with probability ϵ_f (fox eats rabbit)
    Else if $u_t(x) = R$ and $u_t(y) = F$
      $u_{t+1}(x) = F$, $u_{t+1}(y) = E$ with probability ϵ_r (rabbit eaten by fox)
    Else if $u_t(x) = F[R]$ and $u_t(y) = E$
      $u_{t+1}(x) = E$ with probability $\delta_f[\delta_r]$ (die)
      $u_{t+1}(x) = F[R]$, $u_{t+1}(y) = F[R]$ with probability $\rho_f[\rho_r]$ (reproduce)
      $u_{t+1}(x) = E$, $u_{t+1}(y) = F[R]$ with probability $\mu_f[\mu_r]$. (move)

A predator-prey CA model

Things to note

- The mathematical formalism is instructive in general, but an algorithmic description is more useful for most models.
- Lots of parameters! ϵ_f, ϵ_r for eating, δ_f, δ_r for dying, ρ_f, ρ_r for reproduction, μ_f, μ_r for moving.
- Parameter values dictate system dynamics. Extinction of one or both species, or cyclic population growth and decline (a la Lotka-Volterra) are all possible.



(Start simulation now)

Model strengths and weaknesses

Strengths:

- Captures more aspects of population dynamics than Lotka-Volterra.
- CA's allow simple, well-chosen rules to generate complex behaviors.
- Easy to program.
- Large numbers of parameters mean behavior can be tailored to known data.
- Easy to modify for better model fidelity.

Model strengths and weaknesses

Weaknesses:

- Still fails to capture many aspects of predator-prey dynamics.
- High-dimensional state spaces mean analytic results are difficult to produce.
- Simulation via cellular automata is usually inductive rather than deductive.
- May be computationally intractable for large domains.

Elaborations on this model

- Instead of a grid, consider an automaton on a general graph, for better spatial fidelity.
- Create a more complicated food web by adding additional possible CA states.
- Investigate agent-based models instead of CA models.
- Rules can be made to vary in space and/or time.

And now...

Contest Tips 1

Contest necessities

- Everyone should have a computer to work on.
- Look for a (reasonably) comfortable working space ahead of time.
- Software to write up your solution (LaTeX)
- A programming language at least one (preferably two) teammates can use.
- Be able to learn, quickly!

Suggested timeline

- **Before contest begins**: Coordinate! Know where you'll meet, exchange email addresses and phone numbers. Know when teammates won't be available
- **Friday**: Problem is put online at 5PM. Time for research. Do as much background research on the problem as you can. Start outlining at least two possible modeling approaches.
- **Saturday**: Keep doing background research. Choose a modeling approach, start programming an implementation. Start writing. Suggested: 2 working on the model, 1 writing.
- **Sunday**: Both implementation and writing should be in full swing. By Sunday night, 2 people should be writing. Don't go to sleep.
- **Monday**: Solution is due at 10AM sharp. Plan to finish by 9AM.

Solution paper structure

- Abstract
- Title page, table of contents
- Problem description
- Model description (including proposed solution)
- Model assumptions
- Results
- Model strengths and weaknesses
- Conclusion
- Code appendix
- Works cited

Finding and using documents

- Obvious starting places: Google, Google Scholar. Research papers > random websites.
- http://www.lib.umd.edu/ may have access to papers you can't get on Google Scholar.
- Investigate references in papers you've already found.
- Google Scholar also lets you see who has cited a given paper (super helpful).
- Keep a running bibliography, even of papers you aren't sure you'll use. You can trim it at the end.

Finding and using software

- You'll likely need new software, or software libraries during the competition.
- Use existing code when possible. Don't write your own unless you have to!
- Finding and using new software/code means knowing how to search effectively.
- Look for documentation or help pages.