Evolutionary Algorithms
And
Genetic Encodings
Lab Assignments: 35% (ISE-483), 25% (SSIE-583)

- Complete 4/5 assignments based on algorithms presented in class
  - Lab 4: April 12th (Wednesday after Spring break)
    - Evolutionary Algorithms, (Lab 4 in Brightspace Assignments)
    - Due April 24th
  - Lab 5: May 1st
    - Ant Clustering Algorithm, (Lab 5 in Brightspace Assignments)
    - Due May 8th

SSIE – 583 -Presentation and Discussion: 35%

- Present and lead the discussion of an article related to the class materials
  - Enginet students post/send video or join by Zoom
- All presentations completed?
Class Book
  - Chapters 1 and 4.

Lecture notes
- Chapter 1: What is Life?
- Chapter 2: The logical Mechanisms of Life
- Chapter 3: Formalizing and Modeling the World
- Chapter 4: Self-Organization and Emergent Complex Behavior
- Chapter 5: Reality is Stranger than Fiction
- Chapter 6: Von Neumann and Natural Selection
  - posted online @ http://informatics.indiana.edu/rocha/i-bic

Papers and other materials
- Optional
    - Chapter 2, 7, 8
    - Chapter 3, sections 3.1 to 3.5
    - Chapters 10, 11, 14 – Dynamics, Attractors and chaos
- Fogel, Owens and Walsh (1966)
  - *Artificial Intelligence through simulated evolution*. Wiley.
  - Evolution of finite-state machines
- John Koza (1992) at Stanford University
Evolving computer programs to perform a task
- No strict genotype-phenotype mapping
- LISP programs
  - Can be expressed in the form of parse trees

```
(DEFUN AREA-OF-CIRCLE ()
  (SETF R 45)
  (SETF PI 3.1415)
  (* PI (* R R)))
```
the workings

1) Choose a pool of possible functions and terminals
   - Setting up a language of description
2) Generate Random population of trees (programs)
   - Must be syntactically correct (parsing)
   - Size is usually restricted
3) Evaluate Fitness Function for each tree
   - Desired I/O
   - Simplicity, speed
4) Reproduce next generation with variation
   - Trees with higher fitness value reproduce with higher probability
5) Go back to 3)
crossover

- Choose random point in each parent’s tree
- Exchange subtrees beneath to produce offspring
  - Allows size of program to increase or decrease

\[ \text{PI}/\{[(\text{PI}/\text{PI})/(\text{PI}/\text{PI})]\} \]

\[ \text{R+}(\sqrt{\text{PI} \times \text{D}}) \]

\[ \text{R+}\{\sqrt{\text{PI} \times [(\text{PI}/\text{PI})/(\text{PI}/\text{PI})]}\} \]
mutation

- Choose random point in a tree
- Replace subtree beneath with random tree

\[ R+ (\sqrt{\pi} \cdot D) \]

\[ R+ \{ \sqrt{\pi} \cdot [\text{MAX}(A, B) - \pi] \} \]
crossover demo
Mutation demo
Genetic programming

Architecture-altering operations
1) Generate Random population of tree/programs
2) Evaluate Fitness Function for each program
   - Desired I/O, simplicity, speed
3) Reproduce next generation
   - Selection by fitness
   - Variation
     - crossover and mutation
   - Fill new population
4) Go back to 2) until stop criteria is met
   - Desired fitness
   - Specified number of generations
   - Convergence
### Genetic programming applications

- **Optimal control**
- **Planning**
- **Symbolic regression**
  - Fit real data
    - Example: Uncover laws of physics
  - Binary Classification
- **Software Tool**
  - Eureqa: [https://www.creativemachineslab.com/eureqa.html](https://www.creativemachineslab.com/eureqa.html)

- **Robot strategies**
  - Robocup

- **Evolvable hardware**
Genetic programming

- Optimal control
- Planning
- Symbolic regression
  - Fit real data
    - Example: Uncover laws of physics
- Binary Classification

**Eureqa Models**

The *Eureqa Models* tab provides access to model blueprints for Eureqa generalized additive models (Eureqa GAM), Eureqa regression, and Eureqa classification models. These blueprints use a proprietary Eureqa machine learning algorithm to construct models that balance predictive accuracy against complexity.

Examples of applications include:

- **Robot strategies**
  - Robocup
- **Evolvable hardware**
A symbolic regression tool

Eureqa: https://www.creativemachineslab.com/eureqa.html

Candidate models

\[
\begin{align*}
\frac{dx}{dt} &= -2y^2 + \log(x) \\
\frac{dy}{dt} &= -x + \frac{y}{6} \\
\frac{dx}{dt} &= -\sin(y) \\
\frac{dy}{dt} &= -y^{1.8} + \log(x) \\
\frac{dx}{dt} &= -x^2 + \frac{y}{4x} \\
\end{align*}
\]

Candidate tests

The inference process generates several different candidate symbolic models that match sensor data collected while performing previous tests. It does not know which model is correct.

Inference Process

The inference process physically performs an experiment by setting initial conditions, perturbing the hidden system and recording time series of its behavior. Initially, this experiment is random; subsequently, it is the best test generated in step c.
A symbolic regression tool

Eureqa: https://www.creativemachineslab.com/eureqa.html

Genetic Programming in Python, with a scikit-learn inspired API:

gplearn

https://gplearn.readthedocs.io/

One general law, leading to the advancement of all organic beings, namely, multiply, vary, let the strongest live and the weakest die.

—Charles Darwin, On the Origin of Species (1859)

Initial Conditions (actuators)

The inference process physically performs an experiment by setting initial conditions, perturbing the hidden system and recording time series of its behavior. Initially, this experiment is random; subsequently, it is the best test generated in step c.
Types of encoding

- Binary encodings
  - Typically fixed-length
- Many-letter encoding
  - Larger alphabet (e.g. graph-generation grammars)
- Real-valued encodings
  - Genes take real values
- Tree Encodings
  - Genetic programming
- Indirect Encodings
  - Modeling Phenotype development or post-transcription processes
    - L-Systems, Dynamical systems, evolutionary robotics
Cellular automata

homogenous lattice of state-determined systems

Cellular Automata

Density Task
(a.k.a majority classification problem)
- Lattices of 149 Binary Cells (599, 999)
- Rules of Radius 3 (7 Cells in Neighborhood)
- Task: Organize to
  - All 1's if Initial Configuration (IC) has more 1 Cells
  - All 0's if IC has more 0 Cells

Possible neighborhood states

Possible CA transition functions

\[ K^{[N]} = 2^7 = 128 \]
\[ K^{K^{[N]}} = 2^{128} \approx 3.4 \times 10^{38} \]
Cellular automata

encoding in GA with binary encoding

Possible neighborhood states $K^{|N|} = 2^7 = 128$

<table>
<thead>
<tr>
<th>Neighbor State</th>
<th>00000000</th>
<th>01000000</th>
<th>10100000</th>
<th>01100000</th>
<th>10010000</th>
<th>01010000</th>
<th>00110000</th>
<th>01110000</th>
<th>10001000</th>
<th>01001000</th>
<th>10101000</th>
<th>11101000</th>
<th>00011000</th>
<th>01011000</th>
</tr>
</thead>
</table>

Used in the evolutionary search by GA (elite selection)

Traditional Genetic Algorithm

Genotype

Variation

Selection

Phenotype

Pop of rules

- 010010101100100
- 010010101100100
- 010010101100100
- 010010101100100
- 010010101100100
- 010010101100100
Evolving CA rules

With genetic algorithms

- Das, Mitchell and Crutchfield
  - Used Genetic Algorithm to evolve rules for this task

典型结果：块扩张

Regular domains

{0+}

{10+}

{1+}

Particles

to evolve photos with numerical encodings
In genetic algorithms

1) Genotypes contain real or integer values
   1) Crossover is performed in the same way
   2) Mutation assigns a random number in a given interval

2) More computationally demanding for Reals
3) Attention to crossover points
   1) Conversion to binary avoids crossover issues, but longer genotypes

Agent Chromosome/Genotype (Population of $p$ agents)
Evolutionary algorithms to optimize neural networks

- **Capabilities (not in gradient-based ANN)**
  - Generation of ANN building blocks
    - Activation functions
  - Hyperparameter optimization
  - Architecture and learning algorithm search and optimization
- **Massive Parallelism**
  - Population of solutions

---

evolving morphologies and robots with indirect encodings

Karl Sim’s simulations and The Golem Project

http://demo.cs.brandeis.edu/golem/

http://www.youtube.com/watch?v=oCXzcPNsqGA
Gene expression programming

- Proposed by Candida Ferreira
- Program trees are encoded in fixed-length linear genotypes
- Genotypes
  - Open-reading frame architecture
    - Stop signal not necessarily at end of genotype
  - Non-coding genes are possible
    - Can include genetic operators
  - Genes contain two types of symbols
    - Functions (only at the head) and terminals
  - Multigenic solutions
    - Assembled from non-coding operations between various open-reading frames

"Once a Darwinian process gets going in a world, it has an open-ended power to generate surprising consequences: us, for example" Richard Dawkins
readings

- **Class Book**
    - Chapter 7

- **Lecture notes**
  - Chapter 1: What is Life?
  - Chapter 2: The logical Mechanisms of Life
  - Chapter 3: Formalizing and Modeling the World
  - Chapter 4: Self-Organization and Emergent Complex Behavior
  - Chapter 5: Reality is Stranger than Fiction
  - Chapter 6: Von Neumann and Natural Selection
    - posted online @ casci.binghamton.edu/academics/i-bic

- **Papers and other materials**
  - Optional
      - Chapter 5, 7.7, 8.3.1, 8.3.6,