Evolutionary Fuzzy Systems

Evolutionary fuzzy systems are hybrid fuzzy systems where evolutionary optimization algorithms are used to optimize/adapt membership functions and/or rules of a fuzzy system. The evolutionary methods operate by representing the optimization parameters via a gene-like structure and subsequently utilizing the basic mechanisms of Darwinian natural selection to find a population of superior parameters.

Evolutionary Methods

The (human) brain that created “the wheel, New York, wars and so on” (after Douglas Adams)
The evolution mechanism that created the human brain (after Darwin et al.)
Evolution has optimized biological processes; therefore Adoption of the evolutionary paradigm to computation and other problems can help us find optimal solutions.

Flavors of Evolutionary Computing

- **Evolutionary Programming**
  L. Fogel 1962 (San Diego, CA):
- **Genetic Algorithms**
  J. Holland 1962 (Ann Arbor, MI):
- **Evolution Strategies**
  I. Rechenberg & H.-P. Schwefel 1965 (Berlin, Germany):
- **Genetic Programming**
  J. Koza 1989 (Palo Alto, CA):

Genotype vs. Phenotype

- *Living* organisms have both a genotype and a phenotype
- Genotype is underlying genetic coding
- Phenotype is expression of that coding given environmental influences

http://evonet.dcs.napier.ac.uk/

http://www.it.bond.edu.au/jmontgomery/pres
### The Concept of Natural Selection

- Limited number of resources
- Competition results in struggle for existence
- Success depends on fitness --
  - fitness of an individual: how well-adapted an individual is to their environment. This is determined by their genes (blueprints for their physical and other characteristics).
- Successful individuals are able to reproduce and pass on their genes

### When changes occur ...

- Previously “fit” (well-adapted) individuals will no longer be best-suited for their environment
- Some members of the population will have genes that confer different characteristics than “the norm”. Some of these characteristics can make them more “fit” in the changing environment.

### Major Agents of Genetic Change in Individuals

- Mutation in genes
  - may be due to various sources (e.g. UV rays, chemicals, etc.)

### Major Agents of Genetic Change in Individuals

- Recombination (Crossover)
  - occurs during reproduction -- sections of genetic material exchanged between two chromosomes

### Why use evolution as a model for solving computational problems?

- Require search through many possibilities to find a solution
  - (e.g. search through sets of rules for one set that best predicts the ups and downs of the financial markets)
  - Search space too big -- search won’t return within our lifetimes
  - These types of problems are better solved using a parallel approach
- Require algorithm to be adaptive or to construct original solution
Why Evolution is good for solving Optimization Problems

- Evolution is in effect a method of searching for the best (optimal) solution from a great number of possibilities
  - Possibilities -- all individuals
  - Best solution -- the most "fit" or well-adapted individual
- Evolution is a parallel process
  - Testing and changing of numerous species and individuals occur at the same time (or, in parallel)
- Evolution can be seen as a method that designs new (original) solutions to a changing environment

Genetic Algorithms

- Individuals compete for the opportunity to reproduce
- During reproduction, individuals exchange genetic material/information
- Consequently, individuals with new features are introduced into population
- The most "fit" individuals (by some metric) survive to reproduce

GA: Encoding the Problem

- Have to come up with a method of representing an individual.
- Representation of an individual can be using discrete values (binary, integer, or any other system with a discrete set of values).

Example: binary representation.

```
1 0 1 0 0 0 1 1
```

Gene

http://evonet.dcs.napier.ac.uk/
GA: Encoding the Problem

**Genotype:**
- 8 bits
- 1 0 1 0 0 0 1 1

**Phenotype:**
- Integer
- Real Number
- Schedule
- ...
- Something else?

Phenotype could be integer numbers

Genotype: 1 0 1 0 0 0 1 1

Phenotype: 163

\[ 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 128 + 32 + 2 + 1 = 163 \]

Phenotype could be Real Numbers
e.g. a number between 2.5 and 20.5 using 8 binary digits

Genotype: 1 0 1 0 0 0 1 1

Phenotype: 13.9609

\[ x = 2.5 + \frac{163}{256} (20.5 - 2.5) = 13.9609 \]

Phenotype could be a Schedule
e.g. 8 jobs, 2 time steps

Genotype: 1 0 1 0 0 0 1 1

Job | Time Step
---|---
1 | 2
2 | 1
3 | 2
4 | 1
5 | 1
6 | 1
7 | 2
8 | 2

Basic Genetic Algorithm

- **Step 1.** Generate a random population of \( n \) chromosomes.
- **Step 2.** Evaluate fitness of each individual. If target fitness is reached – terminate.
- **Step 3.** Repeat until \( n \) children have been produced
  - Choose 2 parents based on fitness proportional selection
  - Apply crossover and mutation operators to copies of the parents
  - Produce new chromosomes / offsprings
- **Step 4.** Replace parent population with offspring population. Continue from Step 2.
Basic Genetic Algorithm: Example 1

Generation n

<table>
<thead>
<tr>
<th>Population</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001000</td>
<td>17</td>
</tr>
<tr>
<td>00010101</td>
<td>15</td>
</tr>
<tr>
<td>00110000</td>
<td>12</td>
</tr>
<tr>
<td>01011110</td>
<td>11</td>
</tr>
<tr>
<td>01101100</td>
<td>9</td>
</tr>
<tr>
<td>11010001</td>
<td>5</td>
</tr>
<tr>
<td>11100010</td>
<td>4</td>
</tr>
<tr>
<td>11111101</td>
<td>1</td>
</tr>
</tbody>
</table>

Selection

<table>
<thead>
<tr>
<th>Population</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>20</td>
</tr>
<tr>
<td>00111000</td>
<td>13</td>
</tr>
<tr>
<td>01011011</td>
<td>13</td>
</tr>
<tr>
<td>01101001</td>
<td>10</td>
</tr>
<tr>
<td>10100101</td>
<td>9</td>
</tr>
<tr>
<td>11001010</td>
<td>8</td>
</tr>
<tr>
<td>11100111</td>
<td>1</td>
</tr>
</tbody>
</table>

Generation n+1

<table>
<thead>
<tr>
<th>Population</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>20</td>
</tr>
<tr>
<td>00111110</td>
<td>13</td>
</tr>
<tr>
<td>01011001</td>
<td>13</td>
</tr>
<tr>
<td>01101110</td>
<td>10</td>
</tr>
<tr>
<td>10110011</td>
<td>9</td>
</tr>
<tr>
<td>11010101</td>
<td>8</td>
</tr>
<tr>
<td>11101111</td>
<td>1</td>
</tr>
</tbody>
</table>

Basic Genetic Algorithm: Example 2

Generation n

Reproduction

<table>
<thead>
<tr>
<th>Population</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>10</td>
</tr>
<tr>
<td>1011</td>
<td>10</td>
</tr>
<tr>
<td>0100</td>
<td>10</td>
</tr>
<tr>
<td>1001</td>
<td>10</td>
</tr>
</tbody>
</table>

Crossover

<table>
<thead>
<tr>
<th>Population</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>01101101</td>
<td>8</td>
</tr>
<tr>
<td>01001101</td>
<td>8</td>
</tr>
<tr>
<td>10110101</td>
<td>8</td>
</tr>
<tr>
<td>11011001</td>
<td>8</td>
</tr>
</tbody>
</table>

Mutation

<table>
<thead>
<tr>
<th>Population</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>11010101</td>
<td>8</td>
</tr>
<tr>
<td>10111010</td>
<td>8</td>
</tr>
<tr>
<td>11011101</td>
<td>8</td>
</tr>
<tr>
<td>11100010</td>
<td>8</td>
</tr>
</tbody>
</table>

Generation n+1

<table>
<thead>
<tr>
<th>Population</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>10</td>
</tr>
<tr>
<td>1011</td>
<td>10</td>
</tr>
<tr>
<td>0100</td>
<td>10</td>
</tr>
<tr>
<td>1001</td>
<td>10</td>
</tr>
</tbody>
</table>

Basic Genetic Algorithm: Fitness

Fitness characterizes the fit of a specimen to a particular problem.

Example: two strings $g_1 = (010 001)$ and $g_2 = (110 001)$ encode X and Y as 3-bit integers.

If fitness function is: $f(X,Y) = X + Y$, then

- $f(g_1) = 3$;
- $f(g_2) = 7$;

Basic Genetic Algorithm: Selection

There many methods of selection

- The likelihood of any individual becoming a parent is directly proportional to its fitness

Genetic Operators: Recombination

- Need one or more recombination operators for our representation.

- Some important factors are:
  - The child should inherit something from each parent. If this is not the case then the operator is a mutation operator.
  - The recombination operator should be designed in conjunction with the representation so that recombination is not always catastrophic.
  - Recombination should produce valid chromosomes
Genetic Operators: Recombination

Whole Population:

Each chromosome is cut into n pieces which are recombined. (Example for n=1)

\[ \begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \]

parents

\[ \begin{array}{cccccccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{array} \]

offspring

http://evonet.dcs.napier.ac.uk/

Genetic Operators: Mutation

Mutation controls diversity of the population

Some important factors are:
- At least one mutation operator should allow every part of the search space to be reached
- The size of mutation is important and should be controllable.
- Mutation should produce valid chromosomes

Genetic Operators: Mutation

before
\[ \begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array} \]

after
\[ \begin{array}{cccccccc}
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
\end{array} \]

Mutation usually happens with probability $p_m$ for each gene

Factors controlling GA evolution

- Choice of the fitness function
- Population size (number of individuals)
- Number of generations to be run
- Probabilities of:
  - Crossover
  - Mutation

Dynamic Evolution

- Genetic algorithms can adapt to a dynamically changing search space
- Seek out the moving maximum via a parasitic fitness function
  - as the chromosomes adapt to the search space, so does the fitness function
Example

- “Moving Optimum”

Evolutionary Programming

- First developed by Lawrence Fogel in 1966 for use in pattern learning
- Real-valued approach to problem encoding
- Models only the behavioral linkage between parents and their offspring, rather than seeking to emulate specific genetic operators from nature such as the encoding of behavior in a genome and recombination by genetic crossover.

Evolution Strategies

- Similar to Evolutionary Programming
- Developed by Ingo Rechenberg
- Self-adaptation of evolution parameters "Evolution window"
- Individuals (potential solutions) are encoded by a set of real-valued variables (phenotype evolution).

Comparison: GA, EP and ES

<table>
<thead>
<tr>
<th></th>
<th>Genetic Algorithms</th>
<th>Evolution Strategies</th>
<th>Evolutionary Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation</td>
<td>Binary-valued</td>
<td>Real-valued</td>
<td>Real-valued</td>
</tr>
<tr>
<td>Mutation</td>
<td>Minor operator</td>
<td>Main operators</td>
<td>Main operators</td>
</tr>
<tr>
<td>Recombination</td>
<td>Main operator</td>
<td>Minor operator</td>
<td>None</td>
</tr>
<tr>
<td>Self-adaptation</td>
<td>None</td>
<td>Standard deviation</td>
<td>Meta-EP</td>
</tr>
<tr>
<td>Selection</td>
<td>Probability</td>
<td>Deterministic</td>
<td>Probability</td>
</tr>
<tr>
<td>Initial Application</td>
<td>Adaptive system</td>
<td>Parameter optimization</td>
<td>Artificial intelligence</td>
</tr>
</tbody>
</table>

Evolution Strategies: Encoding

- Individuals are encoded as vectors of real numbers (object parameters)
  \[ \mathbf{op} = (o_1, o_2, o_3, \ldots, o_m) \]
- Each individual vector can be accompanied by a strategy vector, detailing how to apply mutations
  \[ \mathbf{sp} = (s_1, s_2, s_3, \ldots, s_n) \]
- Strategy vector is also subject to mutation and recombination
- These two parameters constitute the individual’s chromosome
1. Generate some random individuals
2. Select the $p$ best individuals based on fitness
3. Use these $p$ individuals to generate $c$ children. Children are produced by adding a zero-mean Gaussian random variable to each phenotype of an individual; recombination is also used
4. Go to step 2, until the desired result is achieved (i.e. little difference between generations)

ES: Fitness Functions

- Need a method for determining if one solution is more optimal than another
- Mathematical formula
- Main difference from genetic algorithms is that only the most fit individuals are allowed to reproduce (elitist selection)

ES: Forming the Next Generation

- Number of individuals selected to be parents ($p$)
  - too many: lots of persistent bad traits
  - too few: stagnant gene pool
- Total number of children produced ($c$)
  - limited by computer resources
  - more children $\Rightarrow$ faster evolution

ES: Mutation

- Needed to add new genes to the pool
  - optimal solution cannot be reached if a necessary gene is not present
  - bad genes filtered out by evolution
- Random changes to the chromosome
  - object parameter mutation
  - strategy parameter mutation
    - changes the step size used in object parameter mutation
- Often, a Gaussian/normal distribution $N(0,s)$ is used, where $0$ is the mean value, $s$ is the standard deviation and $o'_i = o_i + N(0,s_i)$ for each parameter

ES: Mutation

- affected by the strategy parameters (another vector of real numbers)
  $$op = (8, 12, 31, \ldots ,5)$$
  $$op_{mutated} = (8.2, 11.9, 31.3, \ldots , 5.7)$$
  $$sp = (.1, .3, .2, \ldots ,.5)$$
ES: Recombination
- Similar to crossover of genetic algorithms
- Equal probability of receiving each parameter from each parent
  
  $$(8, 12, 31, \ldots, 5) (2, 5, 23, \ldots, 14)$$

$$\Rightarrow$$

$$(2, 12, 31, \ldots, 14)$$

ES: Intermediate Recombination
- Often used to adapt the strategy parameters
- Each child parameter is the mean value of the corresponding parent parameters
  
  $$(8, 12, 31, \ldots, 5) (2, 5, 23, \ldots, 14)$$

$$\Rightarrow$$

$$(5, 8.5, 27, \ldots, 9.5)$$

ES: Evolution Process
- $p$ parents produce $c$ children in each generation
- Four types of processes:
  - $p,c$
  - $p/r,c$
  - $p+c$
  - $p/r+c$

ES: Evolution Process $p,c$
- $p$ parents produce $c$ children using mutation only (no recombination)
- The fittest $p$ children become the parents for the next generation
- Parents are not part of the next generation
- $c \geq p$
- $p/r,c$ is the above with recombination

ES: Evolution Process $p+c$
- $p$ parents produce $c$ children using mutation only (no recombination)
- The fittest $p$ individuals (parents or children) become the parents of the next generation
- $p/r+c$ is the above with recombination

ES: Example - Hill Climbing
- 2-dimensional search space
- 3rd dimension is the fitness
- Goal: find the global maximum (most fit solution in the space)
- Avoid local maxima
Example: Rugged Terrain

- More of a challenge to optimize
- Easy to get stuck in the many local maxima
- Need to adjust mutation and crossover rates

ES: Climbing Strategy

- 1.5 - ES
- Hill climbing in action...

ES: Evolution of a Two-Phase Jet Nozzle

- Goal: obtain maximum thrust
- Nozzle represented by a series of conical segments
- Segments evolved by (1+1) ES
- Unexpected shape!

Genetic Programming

- Evolves a population of computer programs represented by trees
- J. Koza 1989 (Palo Alto, CA):

\[ \sin(x) + \sqrt{x^2 + y} \]
Genetic Programming: Representation
- Consists of a terminal set and function set
- Both sets should satisfy closure and sufficiency requirements.
  - Closure: each of the functions in the function set is able to accept as its arguments any value and data-type that may possible be returned by some other function or terminal.
  - Sufficiency: there should be a solution in the space of all possible programs constructed from the specified function and terminal sets.

GP: Prefix Notation
- GP trees and the corresponding expressions can be represented in prefix notation.
- In this notation, functions always precede their arguments.

$$\max(x^2, x+3y)$$

$$\max (\ast x x) (+ x (\ast 3 y))$$

- If all functions have a fixed arity, the brackets become redundant in prefix-notation expressions.

$$\max (* x x) (+ x (* 3 y))$$

$$\max * x x + x * 3 y$$

GP: The Terminal Set
- The terminal set may consist of
  - The program’s external inputs (e.g. x, y),
  - 0-arity functions (e.g. rand(), go_left()),
  - Numerical constants (e.g. 0.1, 3, π).

GP: The Function Set
- For many other problems, the primitive set includes specialized functions and terminals.
  - If the goal is to program a robot to mop the floor
    Function set = \{moving, turning, swishing the mop\}
  - If the goal is the automatic creation of a controller
    Function set = \{integrators, differentiators, leads, lags, gains\}
    Terminal set = \{reference signal, plant output\}
  - If the goal is the synthesis of analog electrical circuits
    Function set = \{transistors, capacitors, resistors\}

GP: Fitness
- Fitness can be measured in terms of:
  - The amount of error between its output and the desired output,
  - The amount of time (fuel, money, etc.) required to bring a system to a desired target state,
  - The accuracy of the program in recognizing patterns or classifying objects into classes,
  - The payoff that a game-playing program produces,
  - The compliance of a structure with user-specified design criteria,

The fitness measure is, for many practical problems, multi-objective, i.e. it combines two or more different elements that are often in competition with one another.
**GP: Basic Algorithm**

1. Randomly create an initial population of programs from the available primitives.
2. Iterate the following sub-steps until the termination criterion is satisfied:
   - Execute each program and evaluate its fitness.
   - Select one or two program(s) from the population with a probability based on fitness to participate in genetic operations.
   - Create new individual program(s) by applying genetic operations with specified probabilities.
3. Return the best-so-far individual

**GP: Selection**

- Genetic operators are applied to individual(s) that are *probabilistically selected based on fitness*.
- Better individuals are favoured over inferior individuals.
- The most commonly employed methods for selecting individuals are *tournament selection* and *fitness-proportionate selection*.

**GP: Crossover**

- Given two parents crossover randomly selects a crossover point in each parent tree and swaps the sub-trees rooted at the crossover points.

**GP: Mutation**

- *Mutation* randomly selects a mutation point in a tree and substitutes the sub-tree rooted there with a randomly generated sub-tree.
- Mutation is sometimes implemented as crossover between a program and a newly generated random program ("headless chicken" crossover).

**GP: An Example**

- **Goal**: to automatically create a computer program whose output is equal to the values of the quadratic polynomial \( x^2 + x + 1 \) in the range from \(-1\) to \(+1\).
- **Step 1 – Definition of the Terminal Set:**
  - The problem is to find a mathematical function of one independent variable, so the terminal set must include \( x \).
  - In order to evolve any necessary coefficients, the terminal set also includes numerical constants.
  - That is: \( T = \{ x; \Re \} \), where \( \Re \) denotes constant numerical terminals in some range (e.g \([-5.0, +5.0]\)).

**Step 2 – Definition of the Function Set:**

- One possible choice consists of the four ordinary arithmetic functions of addition, subtraction, multiplication, and division:
  \[ \mathbb{F} = \{ +, -, *, \% \} \].
- To avoid run-time errors, the division function \( \% \) is protected: it returns a value of 1 when division by 0 is attempted, but otherwise returns the quotient of its two arguments.

---

Step 3 – Definition of the Fitness Function:
- The fitness of a particular individual in the population must reflect how closely the output of an individual program comes to $x^2+x+1$.
- The fitness measure could be defined as the value of the integral of the errors between the value of the individual mathematical expression and $x^2+x+1$.
- Often it is not possible to analytically compute the value of the integral, which is then numerically approximated using dozens or hundreds of different values of the independent variable $x$.

Step 4 – Fixing GP Parameters:
- Population size: 4 (typically thousands or millions of individuals).
- Crossover probability: 50% (commonly about 90%).
- Reproduction probability: 25% (typically about 8%).
- Mutation probability: 25% (usually about 1%)
- Architecture-altering operation probability: 0% (frequently around 1%).

Step 5 – Termination Criterion:
- A reasonable termination criterion for this problem is that the run will continue from generation to generation until the fitness (error) of some individual gets below 0.01.
- Often a maximum number of generations is also used as an additional stopping criterion.

The fitness of each of the four randomly created individuals of generation 0 is equal to the area between two curves.

Initial population of four randomly created individuals of generation 0

**GP: An Example**

**GP: An Example - Run**
Summary of GP’s Human-competitive Results

- 23 instances where GP has duplicated the functionality of a previously patented invention, infringed a previously patented invention, or created a patentable new invention.
- 15 instances where GP has created an entity that either infringes or duplicates the functionality of a previously patented 20th-century invention.
- 6 instances where GP has done the same with respect to an invention patented after January 1, 2000.
- 2 instances where GP has created a patentable new invention (general-purpose controllers).

Regression is a technique used to interpret experimental data. It consists in finding the coefficients of a prefixed function such that the resulting function best fits the data. If the fit is not good then the experimenter has to try with a different function until a good model for the data is found. The problem of symbolic regression consists in finding a good function (with its coefficients) that fits well the data points.
Key issues

Exploration vs Exploitation
- Exploration = sample unknown regions
  - Too much exploration = random search, no convergence
- Exploitation = try to improve the best-so-far individuals
  - Too much exploitation = local search only … convergence to a local optimum

Genetic diversity
- differences of genetic characteristics in the population
- loss of genetic diversity = all individuals in the population look alike
- snowball effect
- convergence to the nearest local optimum
- in practice, it is irreversible

Which method is better:
No free lunch theorem

In 1997, Wolpert & Macready compared black-box optimisation algorithms using probability theory…
- proved that across all possible functions, all such algorithms perform equally well;
- or rather: any one technique will perform well on some problems, really badly on others and no better than random search on the rest

So should we just give up now?
Not at all, identify which techniques work well on different problems

EC: Domains of Application
- Numerical, Combinatorial Optimisation
- System Modeling and Identification
- Planning and Control
- Engineering Design
- Data Mining
- Machine Learning
- Artificial Life
- Evolving neural networks

EC: Performance
- Acceptable performance at acceptable costs on a wide range of problems
- Intrinsic parallelism (robustness, fault tolerance)
- Superior to other techniques on complex problems with
  - lots of data, many free parameters
  - complex relationships between parameters
  - many (local) optima

EC: Disadvantages
- No guarantee for optimal solution within finite time
- Weak theoretical basis
- May need parameter tuning
- Often computationally expensive, i.e. slow
- May be stuck at local minima
- Difficult to find an encoding for the problem
- Difficult to define a valid fitness function
EC: Advantages

- Widely applicable
- Low development & application costs
- Easy to incorporate other methods
- Solutions are interpretable (unlike NN)
- Can be run interactively, accommodate user proposed solutions
- Provide many alternative solutions

Sources for this presentation

- [http://evonet.dcs.napier.ac.uk/](http://evonet.dcs.napier.ac.uk/)
- [http://members.shaw.ca/julie.leung/cpsc533/main.htm](http://members.shaw.ca/julie.leung/cpsc533/main.htm)
- [http://homepage.sunrise.ch/homepage/pglaus/index.htm](http://homepage.sunrise.ch/homepage/pglaus/index.htm)
- [http://faculty.virginia.edu/tip203](http://faculty.virginia.edu/tip203)
- [http://members.shaw.ca/julie.klung/cpsc533/main.html](http://members.shaw.ca/julie.klung/cpsc533/main.html)
- [http://www.nightflight.com/foldoc-bin/foldoc.cgi?evolutionary+programming](http://www.nightflight.com/foldoc-bin/foldoc.cgi?evolutionary+programming)
- [http://faculty.virginia.edu/tip203](http://faculty.virginia.edu/tip203)