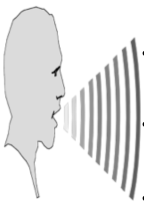
EEL5840: Elements of Machine Intelligence

Rocko

Announcements



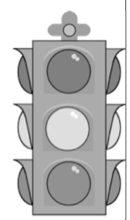
- Announcements:
  - > Tentative 1st Exam Date
    - Thur. Oct. 8th in class
- Today's Handouts in WWW:
  - > Outline Class 14
- Web Site
  - > www.mil.ufl.edu/eel5840
  - > Software and Notes
  - > XLISP Documentation

EEL5840: Elements of Machine Intelligence

Rocko

Today's Menu

- Introduction to Genetic Algorithms (GAs)

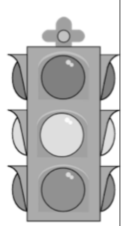


EEL5840: Elements of Machine Intelligence

Rocko

Today's Menu

- Introduction to Genetic Algorithms (GAs)



EEL5840: Elements of Machine Intelligence

Rocko

Introduction to Genetic Algorithms

- What are Genetic Algorithms (GAs)?
  - > Genetic Algorithms (GAs), are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured, yet randomized information exchange.
  - > In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; with an occasional new part tried for good measure.
  - > GAs are not a simple random walk through the state space; they exploit historical information to speculate on new search points with expected improved performance.
  - > They were introduced by John Holland, at the U. of Michigan, 1975, in his paper *Adaptation in Natural and Artificial Systems.*


Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

- What are Genetic Algorithms (GAs)? - Continued-
  - > Two goals in Holland's research
    - – To abstract and rigorously explain adaptive processes in natural systems
    - – To design artificial software that retains the important mechanisms in natural systems
  - > The Central theme in GA research is *ROBUSTNESS*
  - > Robustness is concerned with the balance between efficiency and efficacy **(**the ability to produce a desired amount of a desired effect) necessary for survival in many different environments.
  - > Higher levels of adaptation means that existing systems will perform for longer periods of time
  - > GAs are theoretically and empirically proven to provide robust search in complex state spaces


Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

- Robustness of Traditional Optimization/Search Methods
  - > Calculus-Based Methods
    - – Indirect - solve the usually non-linear set of equations resulting from setting the gradient of the objective function equal to zero.
    - – Direct - seek local optima by hopping on the function and moving in a direction related to the local gradient, i.e., hill-climbing.
  - > Problems with calculus-based methods
    - – They are local in scope, that is, the optima they seek are the best in a neighborhood of the current point . Improvements are obtained via random restart and trickery (heuristics).
    - – Calculus methods depend on the existence of the gradient (well defined derivative/slope values). They often require stationarity.


Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

- Robustness of Traditional Optimization/Search Methods
  - > Enumerative Schemes
    - – Defined over a finite search space, or a discrete infinite search space. The search algorithm starts looking at objective function values at every point in the space, one at a time, e.g., Dynamic Programming (Bellman 1961)
  - > Problems with enumerative schemes
    - – The curse of dimensionality: algorithms and methods that break down on problems of moderate size and complexity.
  - > Random Search
    - – Random search of the problem state space while saving the best answers
  - > Problems with random search
    - – Efficiency considerations—in the long run these methods cannot be expected to do better than enumeration schemes

We must be careful to separate random search algorithms from the randomized techniques used in GAs. GAs are search procedures that use random choice as a tool to guide a highly explorative search through the coding of a parameter space


Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

- The Goals of Optimization
  - > Optimization is to study how to describe and attain what is best once one knows how to measure and alter what is bad or good
  - > Optimization is the quantitative study of optima and methods for finding them.
  - > Optimization seeks to improve performance toward some optimal point(s) thus,
    - – we seek interim performance, the improvement process
    - – we seek optimal point(s) or convergence (the destination itself)

It would be nice to be perfect: meanwhile we can only strive to improve

The most important goal of optimization is improvement

Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

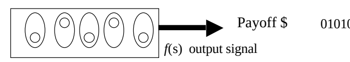
- How are GAs Different
  - > GAs work with a coding of the parameter set, not the parameters themselves
  - > GAs search from a population of points, not a single point
  - > GAs use payoff (an objective function) information, not derivatives or other auxiliary knowledge
  - > GAs use probabilistic transition rules, not deterministic rules

Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

- How are GAs Different
  - > *GAs work with a coding of the parameter set, not the parameters themselves*
    The natural parameter set of the optimization problem must be coded as a finite length string of symbols over a finite alphabet. GAs exploit coding similarities in a very general way; as a result they are largely unconstrained by the limitations of other methods (e.g., continuity, stationarity, existence of derivatives, etc.)

  Example:



Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

- How are GAs Different
  - > GAs search from a population of points, not a single point.
    Moving pt to pt in search spaces that are multi-modal (that have many optima) is a perfect prescription for locating false peaks. GAs on the other hand work from a rich database of points simultaneously, climbing many peaks in parallel, thereby reducing the probability of finding a false peak (weaker local minimum/maximum) as compared to pt-to-pt methods. Thereafter generate successive populations of strings...

    "There is safety in numbers"

  Example:

  Flip an unbiased coin to obtain an initial population of strings for the black box problem, say, n=4 strings, e.g.,

  01101 11000 01000 10011

Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

- How are GAs Different
  - > GAs use payoff (an objective function) information, not derivatives or other auxiliary knowledge
    Gradient search requires derivatives (calculated analytically or numerically) in order to climb the current peak (or alternatively access to most if not all the tubular parameters). GAs are blind. They only require payoff values associated with individual strings. GAs attempt to develop broadly based schemes by ignoring auxiliary information.

  Example

  Let the fitness values of the n=4 strings be given by:

  $f(01101)=169$ $f(11000)=576$ $f(01000)=64$ $f(10011)=361$

Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

- How are GAs Different
  - > GAs use probabilistic transition rules, not deterministic rules GAs do not use simple random search but rather use probability as a guide toward likely improvement.
- Illustration of a Simple GA Algorithm
  - > Mechanics: simple copying and swapping of partial strings to yield simplicity of operation and power of effect
  - > Operators: Three typical operations include
    - – Reproduction
    - – Crossover
    - – Mutation
  - > The idea is to take the initial population and generate successive populations that (we hope) improve (they get more fit) over time.

Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

> Reproduction

The process in which individual strings are copied according to their objective function values, *f* (this function is also known as the fitness function). Intuitively, we can think of the function *f* as some measure of profit, utility, or goodness which we are interested in maximizing. Copying strings according to their fitness values means that strings with a higher *f* value have a higher probability to contributing to one or more offspring in the next generation. This implements survival of the fittest.

A simple algorithmic reproduction operator is to implement a biased roulette wheel where each current string in the popu-lation has a roulette wheel slot sized in proportion to its fitness.

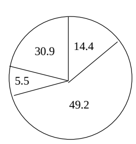
Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

Example:

Summing up the fitness values for the sample strings, we can calculate a percentage of population total fitness and use it as our roulette wheel slot values.

String	Fitness	% of Total
01101	169	14.4
11000	576	49.2
01000	64	5.5
10011	361	30.9
Total	1170	100.0



Rocko

EEL5840: Elements of Machine Intelligence
Introduction to Genetic Algorithms

> Crossover

Two steps: first select members of the newly reproduced string set are mated at random, second each pair of strings undergoes crossing over as follows: an integer position *k* along the string is selected uniformly at random between 1 and the string length less 1 (i. e. the interval [1, *l*-1]) Two new strings are created by swapping all characters between positions *k*+1 and *l* inclusively.

Example:

Suppose strings A_2 and A_4 are selected for mating. Choosing a random number between 1 and 4 (*l*=5) we get 3. The resulting crossover yields:

A_2 = 110|00 A_4 = 100|11
A_2' =110|11 A_4' =100|00

Rocko

EEL5840: Elements of Machine Intelligence

Introduction to Genetic Algorithms

> Why should this work?

Populations of *n* strings over some appropriate alphabet represent complete ideas or prescriptions for performing a particular task. Substrings within each string (idea) contains various notions of what is important or relevant to the task. Viewed this way, the population contains not just a sample of *n* ideas; rather, it contains a multitude of notions and rankings of those notions for task performance.

Genetic algorithms exploit ruthlessly this wealth of information by (1) reproducing high-quality notions according to their performance and (2) crossing these notions with many other high performance notions from other strings. The action of crossover with previous reproduction speculates on new ideas constructed from the high-performance building blocks (notions) of past trials.

This is the process of innovation and it obeys Martin's Law.

Rocko

EEL5840: Elements of Machine Intelligence

Introduction to Genetic Algorithms

> Mutation

The mutation operator plays a secondary role in simple GAs. We note that the frequency of mutation to obtain good results in empirical GA studies is on the order of one mutation per thousand bit transfers. It is usually implemented by a random toggling of a bit position in an otherwise fit string.

Mutation is needed because, even though reproduction and crossover effectively search and recombine extant notions, occasionally they may become overzealous and lose some potentially useful genetic material (a 1 or 0 in a particular bit position). In artificial systems, the mutation operator protects against such an irrecoverable loss (a sort of insurance policy).

Rocko

EEL5840: Elements of Machine Intelligence

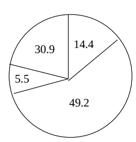
Introduction to Genetic Algorithms

Example:

Suppose we want to maximize $f(x) = x^2$, $0 \le x \le 31$

Start with a population at random. Flip a coin twenty times to obtain n=4 binary strings. Calculate fitness and percentages.

String	Fitness	% of Total	n*%	n'
01101	169	14.4	0.58	1
11000	576	49.2	1.97	2
01000	64	5.5	0.22	0
10011	361	30.9	1.23	1
Total	1170	100.0	4.00	4



Spin the wheel four times to obtain a new generation. We expect n*.144 of string 1, n*.492 of string 2, etc. Suppose we obtained the n' column...

Rocko

EEL5840: Elements of Machine Intelligence

Introduction to Genetic Algorithms

> Crossover

1. The strings are mated randomly using coin tosses to pair off the potential happy couples
2. Mated string couples cross over, using coin tosses to select the crossing sites.

Mating Pool after reproduction	Mate (Random)	Crossover site (Random)	New Population	f
0110|1	2	4	01100	144
1100|0	1	4	11001	625
11|000	4	2	11011	729
10|011	3	2	10000	361
			Total	1754

Mutation is performed on a bit by bit basis with a probability of p_m=0.001 With 20 bits transferred we expect 0.02 bits to undergo mutation $\therefore$ no mutation

EEL5840: Elements of Machine Intelligence

Rocko

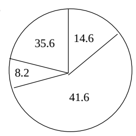
Introduction to Genetic Algorithms

Example: (2nd iteration)

Suppose we want to maximize $f(x) = x^2$, $0 \le x \le 31$

Start with a population at random. Flip a coin twenty times to obtain n=4 binary strings. Calculate fitness and percentages.

String	Fitness	% of Total	n*%	n'
01110	144	8.2	0.32	0
11001	625	35.6	1.42	1
11011	729	41.6	1.67	2
10000	256	14.6	0.59	1
Total	1754	100.0	4.00	4



Spin the wheel four times to obtain a new generation. We expect n*.082 of string 1, n*.356 of string 2, etc. Suppose we obtained the n' column...

EEL5840: Elements of Machine Intelligence

Rocko

Introduction to Genetic Algorithms

> Crossover

1. The strings are mated randomly using coin tosses to pair off the potential happy couples
2. Mated string couples cross over, using coin tosses to select the crossing sites.

Mating Pool after reproduction	Mate (Random)	Crossover site (Random)	New Population	f
11|001	2	2	11011	729
11|011	1	2	11001	625
110|11	4	3	11000	576
100|00	3	3	10011	256
			Total	2186

Mutation is performed on a bit by bit basis with a probability of p_m=0.001 With 20 bits transferred we expect 0.02 bits to undergo mutation $\therefore$ no mutation

EEL5840: Elements of Machine Intelligence

Rocko

The End!