# Machine Learning

## Evolutionary Algorithms

## Machine Learning in Nature

Many machine learning algorithms are based on natural systems:

- Gradient Descent/Hill Climbing
- Genetic Algorithms/Genetic Programming
- Artificial Neural Networks
- Ant Colony Optimization
- Artificial Immune Systems
- Swarm Intelligence



"I'm fascinated by the idea that genetics is digital. A gene is a long sequence of coded letters, like computer information. Modern biology is becoming very much a branch of information science."

- Richard Dawkins

## Genetic Algorithms

## GA Overview

- Developed: USA in the 1970's
- Early names: J. Holland, K. DeJong, D. Goldberg
- Typically applied to discrete optimization
- Modeled after Darwinian natural selection and population genetics
- Method itself is problem independent
- Method is extremely robust



## GA Overview (2)

Holland's original GA is now known as the simple genetic algorithm (SGA)

Other GAs use different:

- Representations
- Mutations
- Crossovers
- Selection mechanisms



## GA Vocabulary

Population

The collection of all candidate solutions

Individual

Individual = Chromosome in simple GAs.

Chromosome

A collection of features or genes

Gene

One piece or feature of the chromosome

## GA Vocabulary (2)

Allele
The value of a gene

Genotype
The encoding of an individual

Phenotype
The manifestation of the individual in its environment

Fitness Function
How good is the phenotype or solution

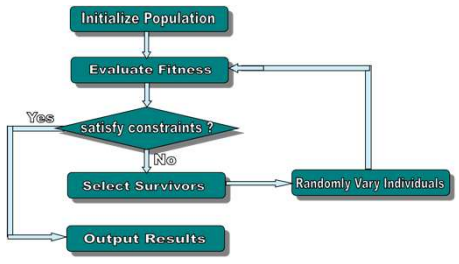


## Big Picture Idea

- Maintain a population of individuals
  - Individual is feasible solution to problem
  - Each individual is characterized by a fitness function
- Higher fitness is better solution
  - Based on their fitness, parents are selected to reproduce offspring for a new generation
  - Fitter individuals have more chance to reproduce



## Conceptual Algorithm



## Encoding

- Parameters of the solution (genes) are concatenated to form a string (chromosome)
- All kind of alphabets can be used for allele (numbers, characters), but generally a binary alphabet is used
- Generally many different encodings for the parameters of a solution are possible
- **Good coding is probably the most important factor for the performance of a GA**



## Algorithm

Generate initial population;

Apply fitness function;

Repeat

Create next population

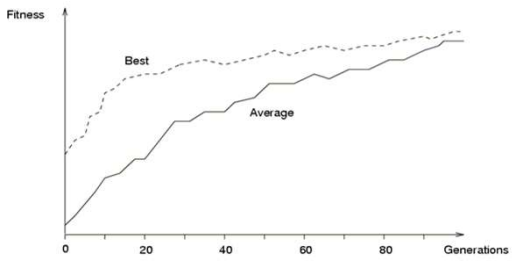
- Selection
- Recombination
- Mutation

Apply Fitness function

until population has converged or solution found



## Convergence



## Problem Independence

Reproduction mechanisms have no knowledge of the problem to be solved

Only link between the genetic algorithm and the problem are:

- Chromosome encoding
- Fitness function



## Example 1: Maximum Span

**Problem**: Schedule *n* jobs on *m* processors such that the maximum span is minimized.

**Chromosomes**: A *n*-vector **x** such that $\mathbf{x}_i = 1$ to *m*

**Fitness**: the maximal span of all processors



## Example 2: Optimization

**Problem**: Find the value in the interval [1-*n*] that maximizes some objective function *f*.

**Chromosome**: a number between 1-*n*

**Fitness Function**: *f*(*n*)

## Example 3: TSP

**Problem**: Find the lowest cost Hamiltonian circuit in a graph of *n* cities.

**Chromosome**: a permutation of *n* cities

**Fitness**: length of tour



## GA Operators

The main operations of the GA are:

- Selection
- Recombination
- Mutation

You can create your own novel methods of each for whatever encoding you design.



## Selection

**Main idea**: better fit individuals have a higher chance to make it into subsequent generations.

Well-Known Selection Types:

- Rank Order Selection
- Roulette Wheel Selection
- Tournament Selection

## Roulette Wheel Selection

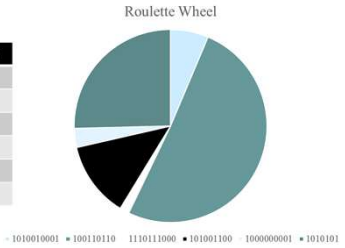
The probability of selection is proportional to the fitness of the individual with relation to the fitness of the entire population.

$$p(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)}$$



## Roulette Wheel Selection (2)

| Chromosome | Fitness |
|---|---|
| 1010010001 | 4 |
| 0100110110 | 32 |
| 1110111000 | 1 |
| 0101001100 | 8 |
| 1000000001 | 2 |
| 0001010101 | 16 |






## Roulette Wheel Selection (3)

A new generation of individuals is created by "spinning" the roulette wheel one time for each population member.

High fitness chromosomes such as 0100110110 are likely to be selected multiple times, low fitness ones such as 1110111000 are likely to be eliminated.

## Rank Order Selection

Best individual is ranked $n$, worst individual is ranked 1. The probability of selection is:

$$f(i) = \frac{rank(i)}{n\,(n\,-1)}$$



## Rank Order Selection (2)

| Chromosome | Fitness |
|---|---|
| 1010010001 | 4 |
| 0100110110 | 32 |
| 1110111000 | 1 |
| 0101001100 | 8 |
| 1000000001 | 2 |
| 0001010101 | 16 |

| Rank | Rank/30 | Selection |
|---|---|---|
| 3 | 3/30 | 10% |
| 6 | 6/30 | 20% |
| 1 | 1/30 | 3.3% |
| 4 | 4/30 | 13.3% |
| 2 | 2/30 | 6.7% |
| 5 | 5/30 | 16.7% |



## Tournament Selection

In Tournament Selection, $n$ individuals are selected at random from the population. The $m$ most fit individuals from the $n$ are selected for the next generation.

## Tournament Selection (2)

If $n = 3$ and $m = 2$, we would perform this selection three times to get six individuals for a replacement population.

| Chromosome | Fitness |
|---|---|
| 1010010001 | 4 |
| 0100110110 | 32 |
| 1110111000 | 1 |
| 0101001100 | 8 |
| 1000000001 | 2 |
| 0001010101 | 16 |



## Recombination

**Main idea**: Individuals chosen by selection algorithm are mated together so that genes are exchanged.

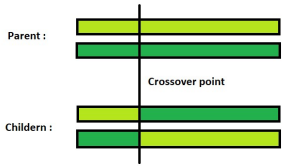
Well-Known Recombination Operators:

- Single Point Crossover
- *N*-Point Crossover
- Uniform Crossover
- Ordered List Crossover



## Single Point Crossover

In **single point crossover**, a random locus (position) along the chromosome is selected and alleles after the crossover point are exchanged.



## Single Point Crossover (2)

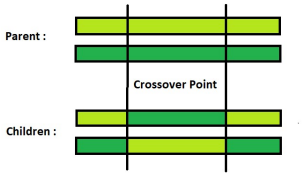
| | |
|---|---|
| Chromosome1 | 11011\|00100110110 |
| Chromosome2 | 11011\|11000011110 |
| Offspring1 | 11011\|11000011110 |
| Offspring2 | 11011\|00100110110 |

Single Point Crossover



## *N*-Point Crossover

***N*-point crossover** is a generalization of the single point crossover operator in which *n* randomly chosen loci are selected. Alternating segments between those loci are then swapped to produce the offspring.






## N-Point Crossover (2)

| | |
|---|---|
| Chromosome1 | 11011\|00100\|110110 |
| Chromosome2 | 10101\|11000\|011110 |
| Offspring1 | 11011\|11000\|110110 |
| Offspring2 | 10101\|00100\|011110 |

Two Point Crossover

## Uniform Crossover

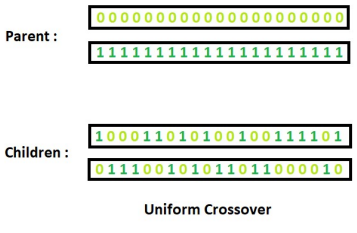
In a **uniform crossover**, the chromosome is not divided into segments but rather each gene is examined separately with a percentage chance for exchange.

The crossover between two good solutions may not always yield a better or as good a solution.

If parents are good, the probability of the child being good is high.



## Uniform Crossover (2)

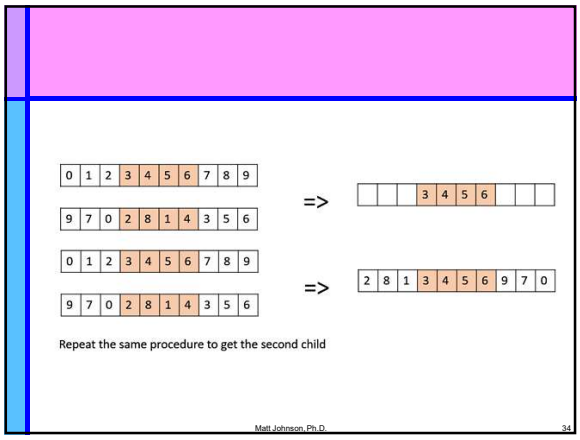





## Order Crossover

Davis's **order crossover** can be used when positional ordering is used in a chromosome.

1. Create two random crossover points in the parent and copy the segment between them from the first parent to the first offspring.
2. Now, starting from the second crossover point in the second parent, copy the remaining unused numbers from the second parent to the first child, wrapping around the list.
3. Repeat for the second child with the parent's role reversed




## Mutation Operators

**Main Idea:** occasionally alter a piece of the chromosome.

Well-Known Mutation Operators:

- Flip Bit
- Swap
- Scramble
- Inversion
- Uniform



## Bit Flip Mutation

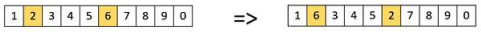
In **bit flip mutation**, we select one or more random bits and flip them. This is used for binary encoded GAs

0 0 1 1 0 1 0 0 1 0 => 0 0 1 0 0 1 0 0 1 0

**Random resetting** is an extension of the bit flip for the integer representation. In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

## Swap Mutation

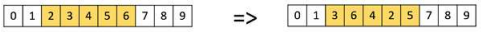
In **swap mutation**, we select two positions on the chromosome at random, and interchange the values. This is common in permutation based encodings.






## Scramble Mutation

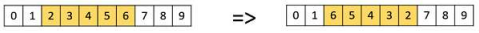
**Scramble mutation** is also popular with permutation representations. In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.






## Inversion Mutation

In **inversion mutation**, we select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.



## Uniform Mutation

**Uniform mutation** is used for real-valued chromosomes in the range [a, b]. The selected gene is replaced with a uniform random number within the [a, b] interval.



## Rate of Operations

Operators are applied with a certain probability. Typically these rates are:

- Selection Rate – high
- Crossover Rate – in between
- Mutation Rate – extremely low



## CASE STUDY: TSP Problem

There are issues unique to this (or any problem).

For TSP:

1. The best fitness has *lowest* value.
2. The recombination and mutation operators must create valid circuits.
3. Chromosomes cannot use binary genes effectively, as it is ordering that's important.

## Chromosomes

There are 20 cities in the distance matrix.

Since it's a circuit, your can always pick the same city as the start of each tour (e.g. Oakland).

Chromosomes must be valid tours, so each needs to be a permutation of the remaining 19 cities (numbers 1 through 19?)



## Initial Chromosomes

One way of generating random chromosomes:

Let list *m* contain numbers 1 – 19.
Let chromosome *c* be empty.
Repeat
Randomly chose a member *v* of *m*
  Remove *v* from *m*
  Append *v* to end of *c*
until *m* is empty



## Selection for TSP

Since lower fitness values are better than higher, Roulette Wheel Selection is difficult to use.

Either tournament selection or rank order selection could work. Experiment!

## Recombination for TSP

There are many different methods for crossover that deal with ordering.

In addition to Davis's method, one possibility is Partially Mapped Crossover or PMX [Goldberg and Lingle, 1985]



## Mutation for TSP

**Swap Mutation** seems like a great fit!