Recall Evolutionary Algorithms

- Start with a population of candidate solutions to a problem you are interested in
  - Each solution is encoded in a representation that consists of a set of atomic features
- Evaluate them (assigning a fitness)
- Select higher fitness solutions
- Apply genetic operators to generate a new population
- Repeat

EA Theory

- Consider any given feature $H$
  $$m(H, t + 1) = m(H, t)S(H)R(H)$$
  - Where $m(H, t)$ is the number of solutions containing $H$
  - $S$ is a gain related to the average fitness of solutions containing $H$
  - $R$ is a term related to the creation and destruction of $H$ by the genetic operators
- Note that $H$ can be any subset of atomic features that can exist in a solution

Linkage

- $R(H)$ is critical to EA success
- If only atomic features can have consistent selection gain, only simple functions can be solved
- For harder functions, particular atomic features must be “held together” (linked) giving high $R(H)$ for optimal solutions to be found

Perspective

- EC algorithms can be seen as being based on probability distributions
- over combinations of features
- So, one EC-derived approach to optimization is as follows...

Generalized Probabilistic Model Building Optimization Algorithm

- Generate a random population of feature combinations (solutions)
- Repeat the following until some measure convergence is reached:
  - Evaluate the fitness of this population
  - Apply a selection operator to reduce this population to combinations that have “high fitness”
  - Using the remaining solutions, refine a probabilistic model of which combinations of features are associated with high fitness
- Based on the selected population and the probabilistic model, generate a new population
Generalized Probabilistic Model Building Optimization Algorithm (2)

- Repeat the following until some measure of model convergence is met:
  - Generate a "random" population of feature combinations (solutions)
  - Evaluate the fitness values for this population
  - Apply a selection operator to reduce this population to combinations that have "high fitness"
  - Using the remaining solutions, refine a probabilistic model of which combinations of features are associated with high fitness
- Use the model to determine parameters, such as population size, for the next steps
- Generate a random population of feature combinations (solutions)
- Repeat the following until some measure convergence is reached:
  - Evaluate the fitness of this population
  - Apply a selection operator to reduce this population to combinations that have "high fitness"
  - Based on this population and the probabilistic model, generate a new population

My Mutual Information Linkage Learning Scheme

- Is of the second type
- Recall...

The (Linkage Learning Part of the) Algorithm, In Brief

- Until the halting criteria is met:
  - Generate a random population, and evaluate fitness
  - Select a highly fit subpopulation
  - For this selected subpopulation, update proportions of each atomic feature, and each pair of features
  - Based on these proportions, calculate mutual information for each feature pair
  - Plot these mutual information values on a histogram
  - Use standard (Gaussian mixture) histogram thresholding to "split off" feature pairs that are strongly dependent, and link them for later use in the GA

After this algorithm is run

- We have a linkage graph, where an arc between parameter x and parameter y means they have a fitness-altering relationship as a pair
- With appropriate recombination operator design, this means the probability $R(H)$ for any feature $H$ defined by a combination of parameters is proportional to those parameter’s connectivity in the linkage graph

In the generalized algorithm

- The step
  - “Based on this population and the probabilistic model, generate a new population”
- Draws on the linkage graph as the probabilistic model, to bias the propagation of features $R(H)$
- And on the population as a distribution of the parameter values that are propagate in these feature combinations

Another Variation

- The Enhanced Compact Genetic Algorithm (ECGA)
- Let’s imagine keeping a marginal probability model of the parameter values in a population
- And generate new populations based on this model
- That is, we estimate the probabilities $P(X|\vec{x})$ for each of a set of non-overlapping subsets of parameters
- But how do we determine the best model of this sort?
An MDL-like score for models

\[ C = (\text{Model Size}) + (\text{Model Entropy}) = M + H \]

Where \( N \) is the population size, \( S \) is the set of partitions of parameters, and \( K_i \) is the cardinality of the parameter \( i \)

\[ M = \log(N) \sum_{S \in S} \prod_{i \in S} K_i \]

Where \( H(S) \) is the entropy of the estimated distribution over partition \( S \)

We can use this as a criteria to minimize in model building

We can use greedy search or simulated annealing etc. and this metric to find good models

The Bayesian Optimization Algorithm (BOA)

- In this case, the probabilistic model is a Bayesian Network
- BNs are powerful, popular representations of distributions over multiple variables
- A BN is a directed graph, where each node represents a variable, and the “parents” of the node are the variables on which this variable depends
- Each node “contains” a model of the appropriate conditional probability

Bayesian Networks

- Are powerful because of the inference that can be performed with them
- In particular, given actual variables (or probabilities of variables) from outside the network, we can determine marginal probabilities and likelihoods of other variables
- The processes involved are non-trivial, but worth investigation elsewhere

Structural and Parameter Learning in BNs

- Structural Learning is determining which connections are necessary in the network
  - Note that this is a discrete, combinatorial optimization problem
- Parameter Learning is determining the appropriate probability distributions, given the network structure
  - This is generally a continuous (and often smooth) optimization problem
  - Note that the number of parameters involved increases exponentially with the number of incoming connections to a node
  - Thus, structural learning has a massive impact on parameter learning
- These two aspects of learning appear in almost every type of knowledge representation

Structural and Parameter Learning

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Structural Elements</th>
<th>Parameters</th>
<th>Active Node</th>
<th>Structural Learning</th>
<th>Parameter Learning</th>
<th>Parameter Space</th>
<th>Structural Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Net</td>
<td>Activator and Connectors</td>
<td>Weights</td>
<td>Hidden Layer</td>
<td>Update at</td>
<td>Update at</td>
<td>System or pruning</td>
<td></td>
</tr>
<tr>
<td>Gaussian Net</td>
<td>Nodes and Connections</td>
<td>Continuous Parameters</td>
<td>Mean and Variance of Output Nodes</td>
<td>Update from Data</td>
<td>Other None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boltzmann Systems</td>
<td>Rules</td>
<td>Active Nodes</td>
<td>Parameters</td>
<td>Update from Data</td>
<td>Other By Hand</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision Trees</td>
<td>Nodes</td>
<td>Active Node</td>
<td>Parameters</td>
<td>Update at Active Node</td>
<td>Threshold Limitation</td>
<td>Unknown Limitation</td>
<td></td>
</tr>
<tr>
<td>Rule-based Systems</td>
<td>Branch Functions</td>
<td>Rules with Significant Output</td>
<td>Antecedents</td>
<td>Confident or Confused</td>
<td>Exclusion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In BOA

- We use a greedy or annealing search, as in ECGA
- We can also use MDL, or a number of other metrics, including the Bayesian-Dirichlet metric

The Bayesian-Dirichlet Metric

\[
p(D, B | \xi) = p(B | \xi) \prod_{x} \frac{m'(x) + m'(x, \pi_{x})}{m'(x, \pi_{x})!} \prod_{x} \frac{m'(x) + m'(x, \pi_{x})}{m'(x) + m'(x, \pi_{x})!}
\]

- \( D \) is the data we have
- \( m \) is a number of samples with particular values
- \( m' \) is a prior number
- \( \pi \) indicates parents

BOA Results

Take Home Messages

- We can define a class of optimization algorithms that
  - Use probabilistic models to define a probabilistic neighbourhood structure over combinations of parameters
  - This could be a relationship between a representation and operators
  - Or a literal probabilistic model
  - Learning models involves a structural learning element (which parameters depend on which) and a parameter element (the probabilities of parameter combinations)
    - In a regular GA,
      - the structural learning is done by the human GA designer, specifying the representation and operators
      - Probability distributions are otherwise represented by the population
    - In a linkage learning GA
      - Structural learning can either be done beforehand
      - Or interleaved in the GA process