

Evolutionary Computation – Terminology and Metaphors

- Evolutionary Computation – Evolutionary Strategies, Evolutionary Programming, Genetic Algorithms, Genetic Programming, Insect metaphors (swarms and colonies), Artificial immune systems – represent a set of search algorithms with various different representations and therefore capabilities.
- All emphasize a population based methodology in which multiple candidate solutions are maintained.¹
- As per any other approach to machine learning decisions need to be made regarding representation, credit assignment and cost function.
- The processes employed for credit assignment tends to differ from that assumed in most other ML frameworks in that:
 - It is necessary to determine who gets to contribute to generating the next set of candidate solutions;
 - More than one individual might be involved in contributing material to candidate solutions.;
 - Representations assumed (and therefore free parameters) need not be numerical.

In the following we concentrate on introducing the basic definitions and context for describing the process of credit assignment as a ‘search’ over a space of candidate solutions.

¹ The emphasis on maintaining multiple candidate solutions makes EC appropriate for speedups under ‘embarrassingly parallel’ computational architectures, most recently in the form of Graphics Processor Units (Langdon, 2011).

Language and Search Spaces

Any EC approach will require,

- A *language* from which to compose solutions
 - the composition of *potential solutions* from this language represents the space of candidate solutions or *search space* (also synonymous with *error surface*).
 - The *search space* is therefore an enumeration of *all possible* combinations of the language.
- Such a search space may be *finite, infinite, continuous* or *discrete*.
- Selection of an appropriate language for the problem at hand will have a significant influence on your ability to define a *well-posed* (or *ill-posed*) search problem.
- It must be possible to *evaluate* any individual in terms of a specific *fitness value* and therefore provide the basis for *relative ranking* to other candidate solutions.
- *Fitness* – where this incorporates the application *cost function* – measures the degree to which desired characteristics are possessed by different candidate solutions.
- There are no smoothness constraints on the fitness function.
 - Conversely, greedy processes for credit assignment would generally require this in order to efficiently sample the search space.

Search Space Structure

Consider the case of an optimisation problem.

The objective is to *minimize* (alternatively *maximize*) fitness, f .

Global optima are the candidate solution(s), x , in the search space, S , for which f is minimized, such that

$$x^* \in S^* \Leftrightarrow f(x^*) = \min_{x \in S} (f(x))$$

In practice we are generally happy if we find the ‘level set’ or,

$$L_{f^* + \epsilon} = \{ x \in S \mid f(x) \leq f(x^*) + \epsilon \}$$

for $\epsilon > 0$

Limitations are that we

- i) know the ϵ which matches the noise levels in the data;
- ii) have knowledge of global optimum x^* (in both formulations!).

Neighbourhoods and Local optima

The search space is given structure by considering the connectivity in neighbouring candidate solutions.

Consider a discrete space:

A **variation operator**, M , starting at any point $x(t) \in S$ creates a ‘move’ to a new point $x(t + 1) \in S$

In EC systems, such an operator is usually stochastic in nature to ensure that *different* neighbours, $x(t + 1)$, are found for the *same* start point, $x(t)$.

All neighbouring points, $x(t + 1)$, generated from a single application of the **variation operator**, M , from initial point, $x(t)$, form the *neighbourhood* of $x(t)$.

Landscapes

We’ve gone to a lot of trouble to provide definitions for local and global optima, *WHY??*

- *Search spaces* summarize the *neighbourhood* structures;
- Solutions correspond to points in such a *landscape*,
 - The ‘height’ of candidate solutions provides a relative value of merit for the candidate.
- The significance of such spaces, however, is closely tied to the influence they have on the *variation operator(s)*.
 - Gradient decent is dominated by such surfaces.

Desirable properties of the search technique are therefore that,

- 1) Must not be caught or trapped by a neighbourhood of the *search process*;
- 2) Obtains solutions within a ‘small’ number of steps (bearing in mind that we are attempting to provide solutions with minimal *a priori* knowledge).

Representation

Individuals or *chromosomes* making up the population of candidate solutions are **not directly** members of the search space, S , but the **representation space**, R .

That is, a *chromosome* is described in terms of a *representation* which, when evaluated, corresponds to a point in the *search space*.

From the Evolutionary Computation perspective there are several reasons for this,

- 1) Biologists distinguish between *genotype* – specification of an organism – and its observable characteristics – the *phenotype*.

Search space, S , is often referred to as the *phenotype*.

Members of the representation space, R , are *genotypes*, *genomes*, *chromosomes* or *individuals*.

(We'll tend to use 'individual.')

- 2) *Variation operators* are usually defined with respect to the *representation space*, not the *search space*.
- 3) *Selection operators* can / do make use of information from the search space cf, 'fitter' solutions are more likely to survive / reproduce.

The mapping between *representation* and *search spaces*, g ,

$$g : R \rightarrow S$$

is equivalent to evaluating individual $x(t)$ to express as a specific fitness value.

- 4) *Fitness* provides a *point* in *search space* to associate with the quality of an individual.
 - The suitability of an individual's representation **biases** the efficiency with which solutions are found.
 - one typical problem is that multiple different individuals may map to the same point in *representation space*.

All representations are generally considered to be composed from *genes*.

Genes take on one of a fixed set of values – the *alleles*.

Operators

Two forms of operator: *Population* (or *selection*) and *Variation*.

Variation Operators²

- Often problem specific and fall into 3 general cases;
 - Mutation, Crossover (recombination), and local search.
- 1) Mutation Operators
 - Operate on a single individual, the *parent*, producing a single new individual, the *child*.
 - *Strategy* parameters act to direct this (e.g. probability of application or variance).
 - Basic operation of mutation is **exploration** or to,
 - generate new individuals whose representation (potentially) does not currently exist in the population.
- 2) Crossover or Recombination
 - Typically, but not always, based on 2 parents, resulting in as many children.
 - Emphasizes **exploitation** of material from current population.
 - Assumes material necessary for an optimal solution currently exists in the population.
 - Strategy parameters again act to control the level of exploitation (e.g. probability of application)
- 5) Local Search
 - Stochastic operators not particularly efficient at fine tuning parameters;
 - Attempt to resolve this by switching into gradient decent at an appropriately chosen point e.g., (Renders and Flasse, 1996).

Population Operators

- Independent of the problem;
- Define processes for *selection*, *replacement* and in the case of multi-population models, *migration* and *deme* replacement.

Basic Objective (of Population Operators)

- Select the individuals to contribute to the next population;
- At initialisation we have a random sampling of candidate solutions.

² Sometimes also referred to as ‘search operators’.

- As candidates are evaluated information is retained regarding where the search effort should be directed.
 - This is through the action of the population based operators, thus biasing,
 - Selection of individuals;
 - Manner in which children are created;
 - Formulation of a new population.
- Efficiency of the process is dependent on our ability to utilize partial information gleaned from the,
 - Distribution of individuals (and their fitness) across the search landscape;
 - Suitability of the variation operators and representation to the problem.

Summary

- *Population* acts as the memory structure.
- *Population operators* direct the population sampling process,
 - Darwin's survival of the fittest – population is of fixed size, enforcing a competition;
 - Selection is *biased* by the fitness of individuals – the degree of greediness associated with of the selection of individuals for inclusion in the next population is terms ***selection pressure***.
- *Variation operators* direct the creation of new individuals.
- Mapping between *representation* and *search space* has the potential to make the problem more difficult or easy to solve.
- The very *nature of variation operators* generates *extensive redundancy* in individuals.

Pragmatics

- Search Spaces are a metaphor used to visualize a concept;
 - Tend to break-down when you move away from individuals expressing anything other than a number in an optimisation problem.
 - Emphasizes a microscopic view to the problem.
 - Both GA and GP emphasize problem solving through the discovery of suitable problem specific “building blocks” (will discuss how much this actually holds up in practice later ☺).
 - Variation operators might not be symmetric.

- How might you represent this as a search space?
- No Free Lunch Theorems (NFL)
 - NFL Theorem proves that no search algorithm is superior on average across all possible problems (Walpert, Macready, 1997).
 - Implication
 - Any machine learning technique (deterministic or stochastic) might well perform better on one problem, but there is always a set of problems for which random sampling is better.
 - Pragmatics
 - Only interested in solving some specific classes of problem. Thus, as long as the machine learning technique performs better over these, we are happy!

Additional reading

- Sections 1.3 – 1.5 of course text.
 - Melanie Mitchell – An Introduction to Genetic Algorithms.
- Nicholas J. Radcliffe (1997) *Introduction – Theoretical Foundations and Properties of Evolutionary Computations*. Handbook of Evolutionary Computation, Section B2.1, Oxford University Press
- Jean-Michel Renders, Stéphane P. Flasse (1996) *Hybrid Methods using Genetic Algorithms for Global Optimization*. IEEE Transactions on Systems, Man, and Cybernetics, 26(2) pp 243-258
- William B. Langdon, Riccardo Poli (2002) Fitness Landscapes. Foundations of Genetic Programming, Chapter 2, Springer-Verlag
- William B. Langdon (2011) Graphics Processing Units and Genetic Programming: An overview. Soft Computing, in press.
- David H. Wolpert, William G. Macready (1997), *No Free Lunch Theorems for Optimization*. IEEE Transactions on Evolutionary Computation, 1(1), pp. 67-82