Formal Computational Skills

Lecture 10: Optimisation

Today: Optimisation also known as Search

Why? Lots of problems can be formulated as a set of parameters which must be found or explored to generate a behaviour/approximate a function etc

However: People also interested in search procedures themselves and especially evolutionary theory

Will focus on former (though also applicable to latter) and take view that each set (instantiation) of parameters defines a potential solution to the problem and we have a function which judges its performance and an (iterative) procedure to generate new solutions

Eg: neural network defined by its weights. Optimise weights until outputs match targets as judged by $(\text{output-target})^2$ via backprop

There are therefore 3 main elements:

1. The space of potential parameters ie all the potential solutions

2. The error function used to judge solution performance

3. The iterative procedure used to generate new solutions from old solutions

What we care about (in general) are: length of time taken to generate good solutions, how good solutions are and what regions of search space (ie potential solutions) our procedure allows us to explore

Will talk about:

- Search spaces
- Search space properties
- Different classes of search procedure
- Optimisation when evaluation is noisy
- (little bit of) statistical testing

Will not talk about:

- How to evaluate search space properties (not enough time and it varies on a case-to-case basis)
- Specific algorithms and operators: too many variants and will just give the general idea
- Solution Encoding – again not enough time
- Details of how statistical properties are evaluated

May lapse into GA type language at times

Search Space

The Search Space is the set of all possible solutions

eg 2D binary string: 4 solutions, 2D real valued: a 2d plane

Thus search space is N-dimensional where N is (maximum!) number of parameters used

How we move through the search space is defined by our search procedure

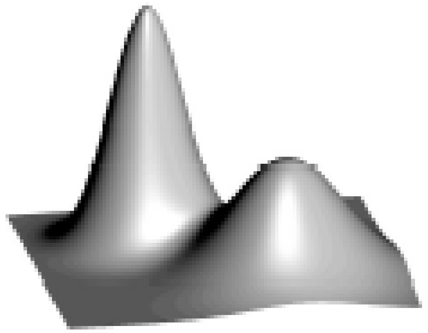
Search space can therefore be seen to be a connected graph where connected points are those that can be reached by the search procedure from current position in one iteration and depending on operators will be more likely to get to certain destinations

In general: If operators are well designed should be able to get to all nearby areas of space

Search Space as Landscape

In GAs referred to as fitness landscape:

Often search space viewed as an N+1 dimensional landscape where extra dimension is the solution performance (fitness) eg 2D real valued genotype gives us nice landscape below



Can be a useful metaphor but **REMEMBER** picture can be **<u>very</u>** different in high dimensions

Eg GasNet search-space, average of 200 dimensions: are there really any local optima (peaks and valleys)? And genotype has variable length - ???

Error functions

Need an error function (fitness function) to evaluate solution's performance

Clearly of central importance: if there is little chance of differentiating between good and bad solutions, the optimisation process cannot hope to succeed.

Essentially it defines the search space and the type of solution one achieves

Eg: if error is $E=(\text{output}-\text{target})^2$ neural network will favour solutions that try to get outputs to match large targets as a 1% error for a target of 100 (ie $E=1^2=1$) will completely swamp a 100% error in a target of size 0.1 ($E=0.1^2=0.01$)

Answer? Could use $E=((\text{output}-\text{target})/\text{target})^2$ but depends on what characteristics we desire from our network output

Ideally, there should be smooth paths in the search space leading to optimal (or good) solutions but in reality may not be possible

Basically one should ensure that there is a gradient for evolution to follow and avoid having large local optima

However, this process is generally post-hoc eg making a fitness function for a GA

1. Design fitness function
2. Population gets stuck in local optimum
3. Design new fitness function which avoids local optimum, population gets stuck again.
4. Repeat ad nauseum till you regret ever criticising wonderful gradient-descent techniques and wonder why you bothered with nonsense evolutionary methods etc etc etc

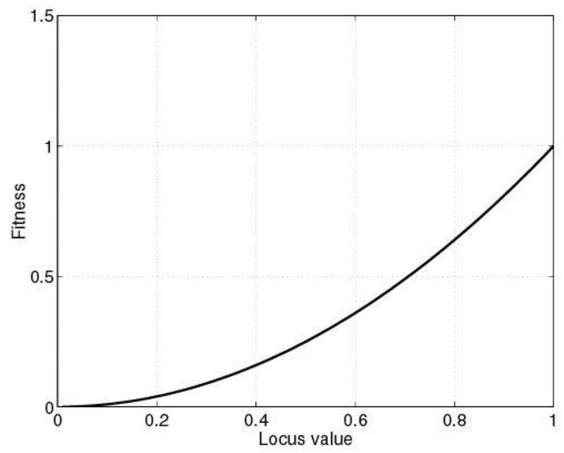
Epistasis, ruggedness and local optimality

If fitness dependent on a non-linear combination of the solution parameters (genotype loci), the solution (genotype) is said to be *epistatically-linked*.

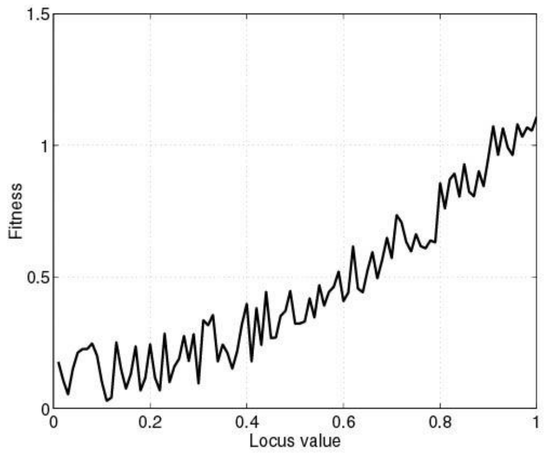
Ie individual locus fitnesses are dependent on the context of other loci values and inter-locus interactions

This will generally be the case for complex problems eg ANN robot controllers

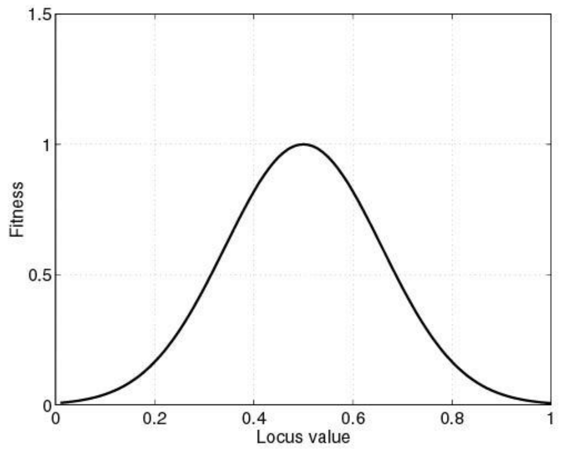
Epistatically-linked solutions/genotypes give rise to the two major properties of search/fitness landscapes thought to influence search dynamics, *ruggedness* and *local optimality*.



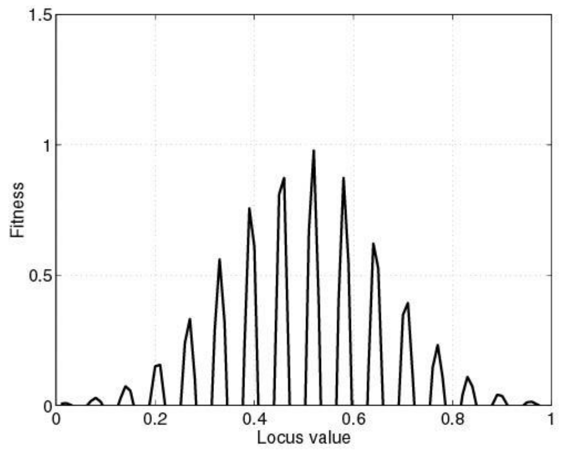
Smooth vs rugged



Ruggedness is regarded as similar to evaluation/fitness noise, where direction to good solutions may be obscured by local noise



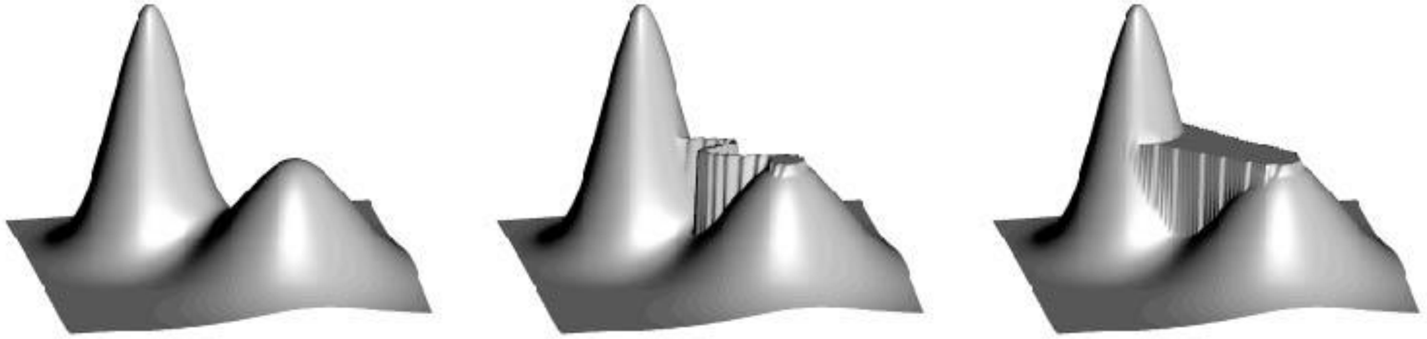
Global vs local optima



By contrast, local optimality is typically thought of in more global terms, with landscapes containing numbers of *deceptive* peaks

However no rigorous distinction between the two properties

Neutrality



Neutral landscapes are where one can move to points of equal fitness: moving along a neutral network

optimisation in landscapes with high levels of neutrality is characterised by periods when performance does not increase interspersed by short periods of rapid fitness increase (punctuated equilibria etc)

Adaptive evolution on neutral landscapes has shown that populations tend to move to areas of space which have more neutral neighbours ie the neutral evolution of robustness

Neutrality may be useful in escaping from local optima but in high dimensional spaces, hard to tell if one is moving neutrally or hovering around a local optimum

Indeed, often difficult to measure **ANY** of the aforementioned properties

One major reason is that in many spaces vast majority of scores are 0 so properties say space is homogeneous and 0

However, if networks evolve faster they must, in some sense, be making the space of solutions easier to search in (smoother? More densely packed with good solutions? Less local optima? More neutral?? Other properties?)

Analysis will hopefully tell us what the search space properties are like and what features of our networks are 'good' for search

Optimisation Procedures

Various flavours of search procedure, but mainly iterative ie:

1. Start with random solution
2. Generate a new one based on the current solution and its performance
3. Repeat till success or boredom

Various types but distinctions blurry (and my own). Here's a taste:

1. Gradient-based: calculate gradient of error wrt the adjustable parameters. Generate new solution by taking step in the direction of negative gradient eg $w \to w - k\, dE/dw$. Can augment with 2nd order info, momentum etc. But often can't calculate gradient so...

2. Hill-climbers: modify (mutate) current solution. Is it better than current one? If so keep solution, if not mutate again. Various modifications – accept neutral moves, accept downwards with a certain probability (simulated annealing). Can be biased by starting position so try multi-start which leads us to ...

3. Population-based such as a GA: maintain a population of solutions. New population generated either as a whole or one at a time. New individuals can be based on combinations of several of the current members of the current population. Can be easy to implement, don't need much knowledge of the problem, solves some of the problems of hill-climbers and can be robust due to population. **NOT** a panacea for all search problem ills: introduce a whole host of other problems (as do hill-climbers etc) such as representational issues and problems of getting good operators.

Movement/Neighbourhood

By specifying how new individuals are created, search process also specifies what new solutions can be reached from current one (its neighbourhood) and how likely each is to be generated

Eg: 3D binary string 000, mutation is 1 bit flip/individual, 3 destinations, 100, 010, 001, all with probability = 1/3

3D integer valued string range [0, 99]: 50 50 50, 0.1 probability of mutating each bit. If bit mutated, moves a gaussian distance with mean 0 sd 5. Can get to ANY point in space, BUT probability of 0 0 0 is a lot less than probability of 48 50 50 (around 0.015)

3D integer valued string as before but also prob 0.01 of adding a new variable to the string … ?????

Noisy Evaluations

To complicate things further, also often have noisy fitness, due to eg not being evaluated over exactly the same conditions (robot fitnesses are often highly dependent on the initial conditions). Alternatively environment could be a source of noise

Typically this noise will obscure differences between the performances of neighbouring solutions (although sometimes the noise can be helpful to eg allow search to escape from local optima, or to smooth the search space)

Usually use scores taken over a small subset of different conditions as potential sample sets for training are often huge

Question is, how can we say which solution better than the other: can we say use average score??

Hypothesis Testing

Suppose we test 2 robot controllers A and B 10 times. A is better than B 8 times. Which is better?

Assume the null hypothesis H_0: controllers are identical and differences can be explained by random fluctations.

So calculate the probability of getting observed results (or more extreme ones) assuming controllers identical (and a certain data distribution). Call this probability p, the significance level

If we reject H_0 based on the evidence then we accept H_1 the alternative hypothesis that A is better than B.

However we can't be sure that we are right and only know that there is a probability p of us being wrong

Eg if 2 contollers are identical, P(A>B) = 0.5. Assume binomial data distribution

There are 2^{10} = 1024 different outcomes each equally likely

trial	1	2	3	4	5	6	7	8	9	10	Probability
winner	A	A	A	A	A	A	A	A	A	A	1/1024
winner	B	A	A	A	A	A	A	A	A	A	1/1024
winner	A	B	A	A	A	A	A	A	A	A	1/1024
winner	A	A	B	A	A	A	A	A	A	A	1/1024

1 trial where A wins all 10

10 trials where A wins 9 and B wins 1

45 trials where A wins 8 and B wins 2

So 56 different ways of getting our result or more extreme one given identical controllers. Probability of each is 1/1024.

Therefore probability of our result p = 56/1024 = 0.055

Is this a significant difference?

Depends on your viewpoint or how exact you need to be: basically roughly 5% of the time we run this experiment we would get this result if controllers identical

ie if you accept the difference you will be wrong 1 in 20 times

Guidelines:

- pick a significance level beforehand
- say what p was and let readers decide
- more trials etc

Problems of biased reporting: we publish the one experiment that worked and not the 5 that didn’t

Types of Test

Different tests appropriate for different situations/tasks

Just described (vague) chi-squared test assuming a binomial distribution. Proper one uses a smooth distribution and can be expanded to use scores and expected scores so more info

t-test does similar but tests differences in means given scores

Both assume Gaussian distributed data: often not the case eg if scores are capped at 1. Must use rank-based/parametric methods (eg Wilcoxon) where scores are ranked in order of size:

eg A: 1 0.80 1 0.95 ranked scores: A = 2 8 2 4

B: 1 0.85 0.87 0.88 B = 2 7 6 5

if scores tied, assign mean rank to all points (other procedures possible). Test has less power as we are losing info. Can use more complex tests if we know we have data with heavy tails

Summary

This lecture:

- discussed the concept of search space and its properties
- looked at some search procdures, their merits and their behaviours
- discussed the problems noisy evaluations
- introduced hypothesis testing as a way of analysing data

And that’s it!!!!

Merry Christmas.

Questionnaires

Please fill in course assessment forms on-line: lecturers and uni DO use them (both to improve courses and to evaluate lecturers)

Where they say constructive comments do (if you like) what you want me to **stop** doing, **start** doing and **continue** doing, or what you **enjoyed** and what could be **improved**

Also important for me if you have any ideas about the peer assessment side: I think it works but I'm not sure and could well be improved

Feel free to be critical but try to be constructive with it. More comments the better (but numbers also useful if you have no comments).