Application of evolutionary algorithms to solve complex problems in quantitative genetics and bioinformatics

1. Overview

Seek, and you shall find.

Brian Kinghorn

University of New England
Armidale, Australia

Course aims

- To empower participants

Evolutionary algorithms

- String together about 3,000,000,000 nucleotides A T G C
- Number of combinations = $4^{3\ \text{billion}}$ = <error>
- How get that right !? … Impossible ?? But it has happened !
- Lean on that power to solve other complex problems
- Gain insight into the power of evolution

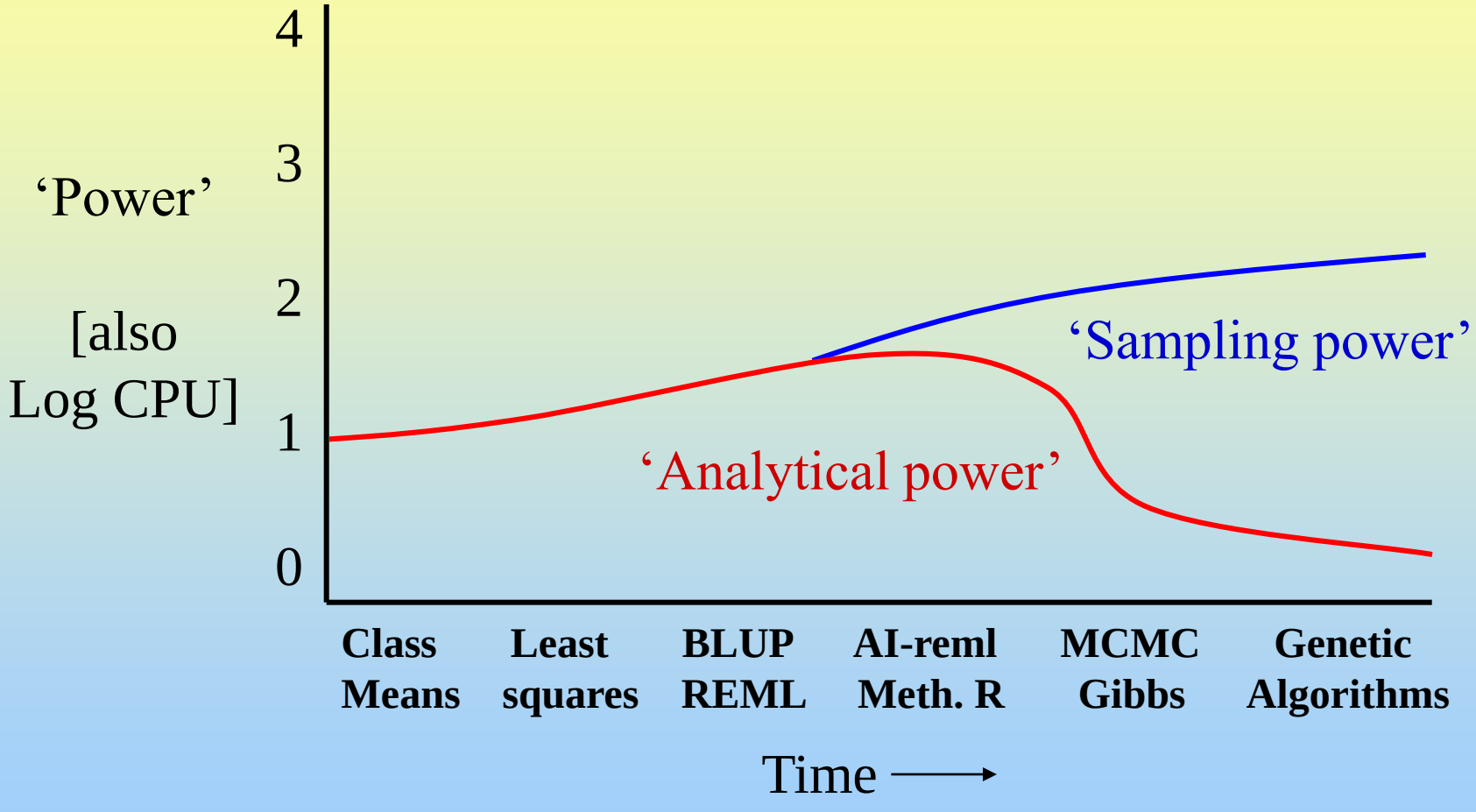
Evolutionary algorithms

- Not a ‘Hard’ science
- Tips and tricks
- Not constrained to how evolution works in biology

Problems NOT for evolutionary algorithms

- Problems where we can ***calculate*** the answer. Eg $\hat{b} = (X'X)^{-1}X'Y$
- Problems where we can ***numerically solve*** for the answer
  eg. IOC BLUP, segregation analysis
- Problems where we can ***intelligently sample*** solutions. Eg. Gibbs sampling
- ... UNLESS the above make assumptions that are violated.
- Evolutionary algorithms can ***find*** solutions for all these problems – but probably not the best route!

Sampling in data analysis ...



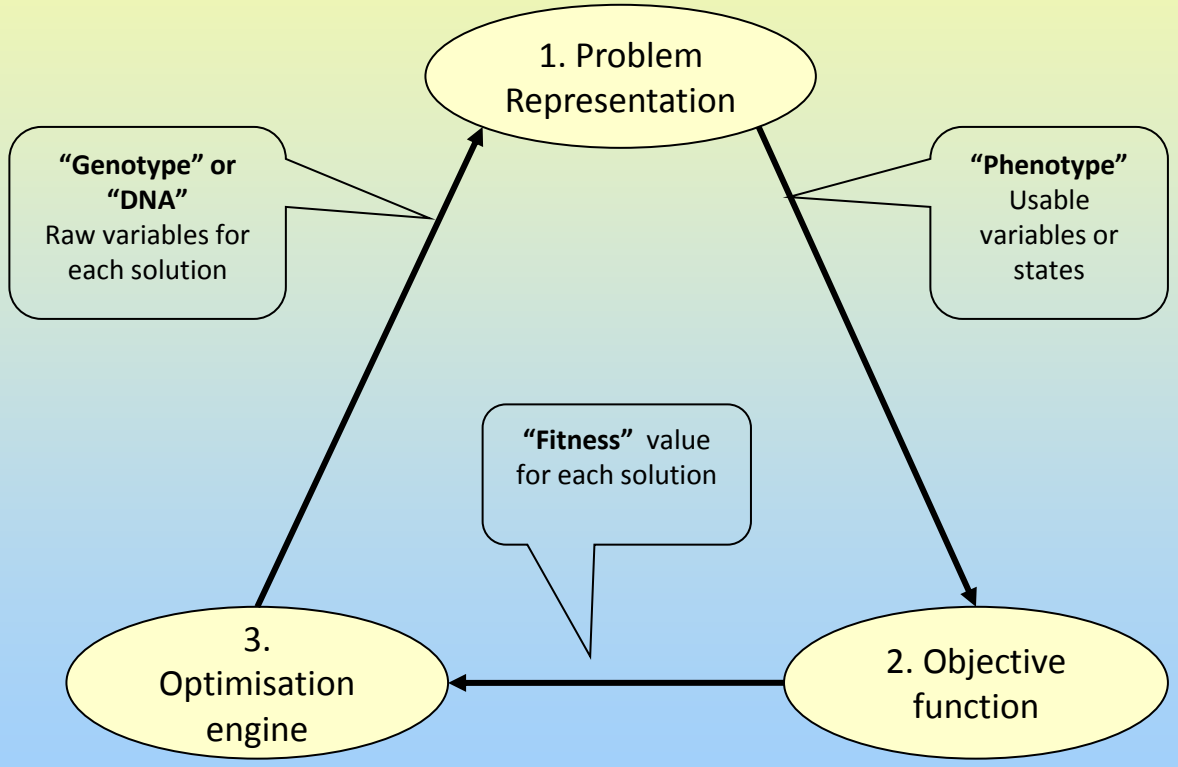
PS. This is a fit, not a plot. Interpret with kindness.

Problems for evolutionary algorithms

Virtually any problem, such as:

- Assignment of individuals to groups
  - Eg. To management groups ; to be tested; to be genotyped; to be selected
- Problems involving thresholds and rules
  - Eg. Supply chain optimisation; animal production models
- Combinatorially tedious problems
  - Eg. Setting up matings; which SNPs to genotype; which SNPs to fit

Architecture of an Evolutionary Algorithm



1. **Problem representation:** Produce the input variables/states ("Phenotypes") from a vector of simple numbers ("Genotype"). Ideally produce only legal solutions to the problem.

2. **Objective function:** Evaluates the "Fitness" of each of these solutions.

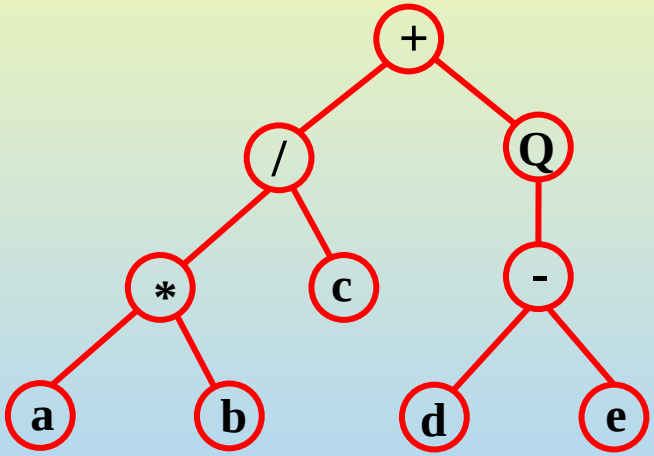
3. **Optimization engine:** Make genotypes of progeny out of the genotypes of parents. It seeks the Genotype that gives the highest fitness.

Architecture of an Evolutionary Algorithm

"Genotype"

+/Q*c-abde

"Phenotype"



$$\frac{a*b}{c}+\sqrt{d-e}$$

1. **Problem representation:** Produce the input variables/states ("Phenotypes") from a vector of simple numbers ("Genotype"). Ideally produce only legal solutions to the problem.

2. **Objective function:** Evaluates the "Fitness" of each of these solutions.

3. **Optimization engine:** Make genotypes of progeny out of the genotypes of parents. It seeks the Genotype that gives the highest fitness.

[Gene Expression Programming will be covered in Chapter 5]

Architecture of an Evolutionary Algorithm

Data	$\frac{a*b}{c}+\sqrt{d-e}$	Error²
4.4	3.5	0.81
3.2	3.1	0.01
5.3	3.4	3.61
6.2	7.4	1.44
7.1	5.9	1.44
1.2	2.1	0.81
	SSE:	8.12

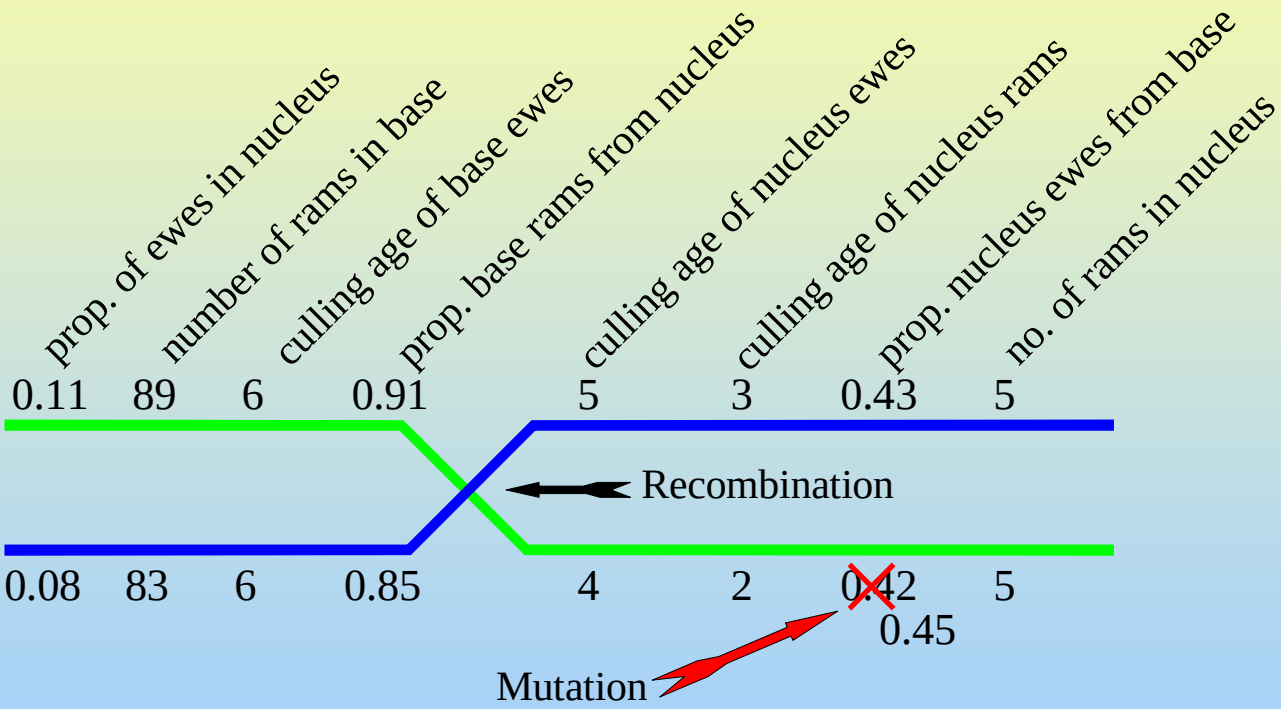
Fitness = -1 * SSE

1. **Problem representation:** Produce the input variables/states ("Phenotypes") from a vector of simple numbers ("Genotype"). Ideally produce only legal solutions to the problem.

2. **Objective function:** Evaluates the "Fitness" of each of these solutions.

3. **Optimization engine:** Make genotypes of progeny out of the genotypes of parents. It seeks the Genotype that gives the highest fitness.

Architecture of an Evolutionary Algorithm



1. **Problem representation:** Produce the input variables/states ("Phenotypes") from a vector of simple numbers ("Genotype"). Ideally produce only legal solutions to the problem.

2. **Objective function:** Evaluates the "Fitness" of each of these solutions.

3. **Optimization engine:** Make genotypes of progeny out of the genotypes of parents. It seeks the Genotype that gives the highest fitness.

Let your computer make you famous


The End