THE UNIVERSITY OF
SYDNEY

Lecture 9: Monte Carlo integration and combinatorial optimisation

STAT5003

Pengyi Yang

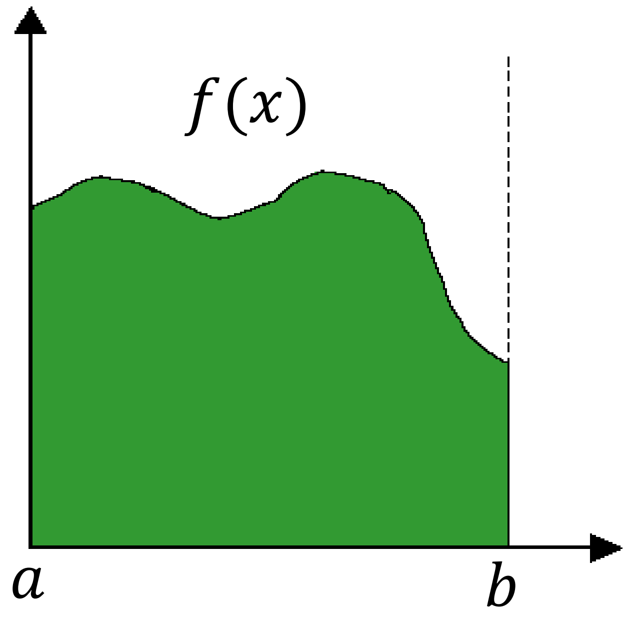
What is Monte Carlo Methods?

- Monte Carlo are a *class* of computational methods
- can be applied to a wide range of problems
- the key aspect of Monte Carlo methods are that they rely on *repeated random sampling*
- they generally provide approximate solutions (not exact solutions)
- they are used in cases where analytical or numerical solutions don't exist or are too difficult to implement

- In this lecture, we will demonstrate Monte Carlo methods for
  - Function integration
  - Global optimisation

Monte Carlo for numerical integration

- A lot of problems can be converted into integration problems.
- Monte Carlo integration is the most common application of Monte Carlo methods.
- This is applied when analytical solution is difficult to find or doesn't exist.



$$\int_a^b f(x)dx = ?$$

Expectation of functions

The expected value of a random variable, x, is defined as:

$$E[x] = \int_A x \cdot p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} x_i$$

The expected value of a function, $f(x)$, is defined as:

$$E[f(x)] = \int_A f(x) \cdot p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

where $x_i \sim p(x)$

Formulation of Monte Carlo integration

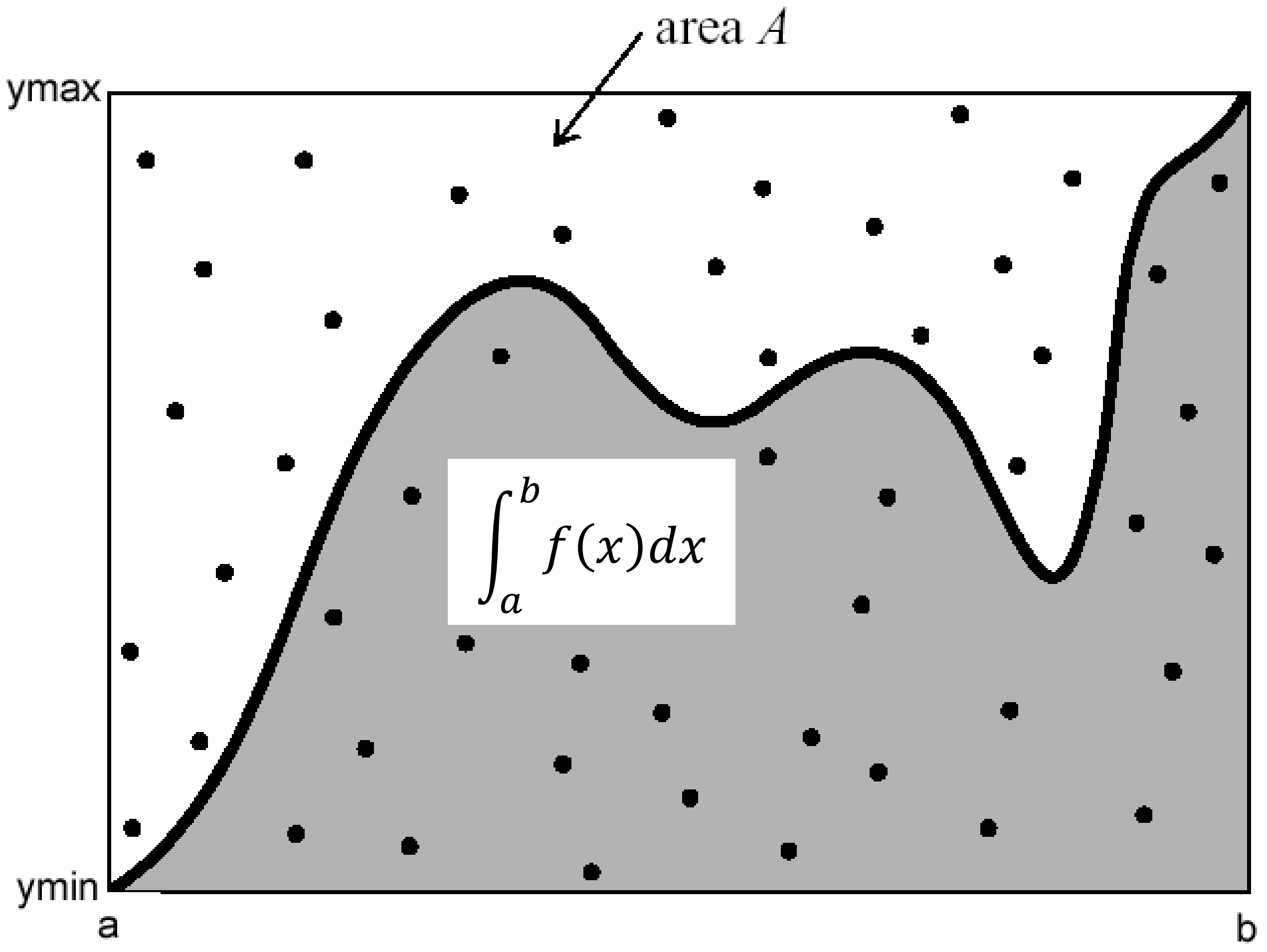
- If we can substitute $g(x) = f(x)p(x)$

$$E[f(x)] = \int_A g(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} \frac{g(x_i)}{p(x_i)}$$

where $x_i \sim p(x)$

- Key idea: generate random points $x_i \sim p(x)$, evaluate the function $g(x)$ at these random points in the integration domain *A*, and sum up the answers
- $x_i \sim p(x)$ is often set to be an uniform distribution or normal distribution

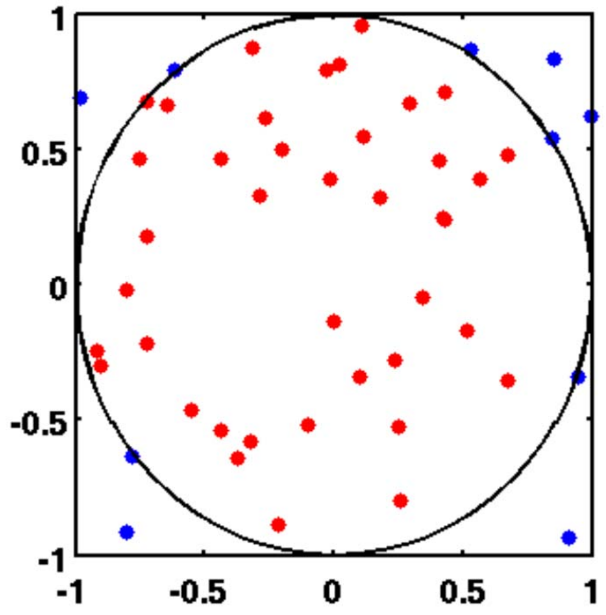
Visually



$$\int_a^b f(x)dx$$

Demonstration

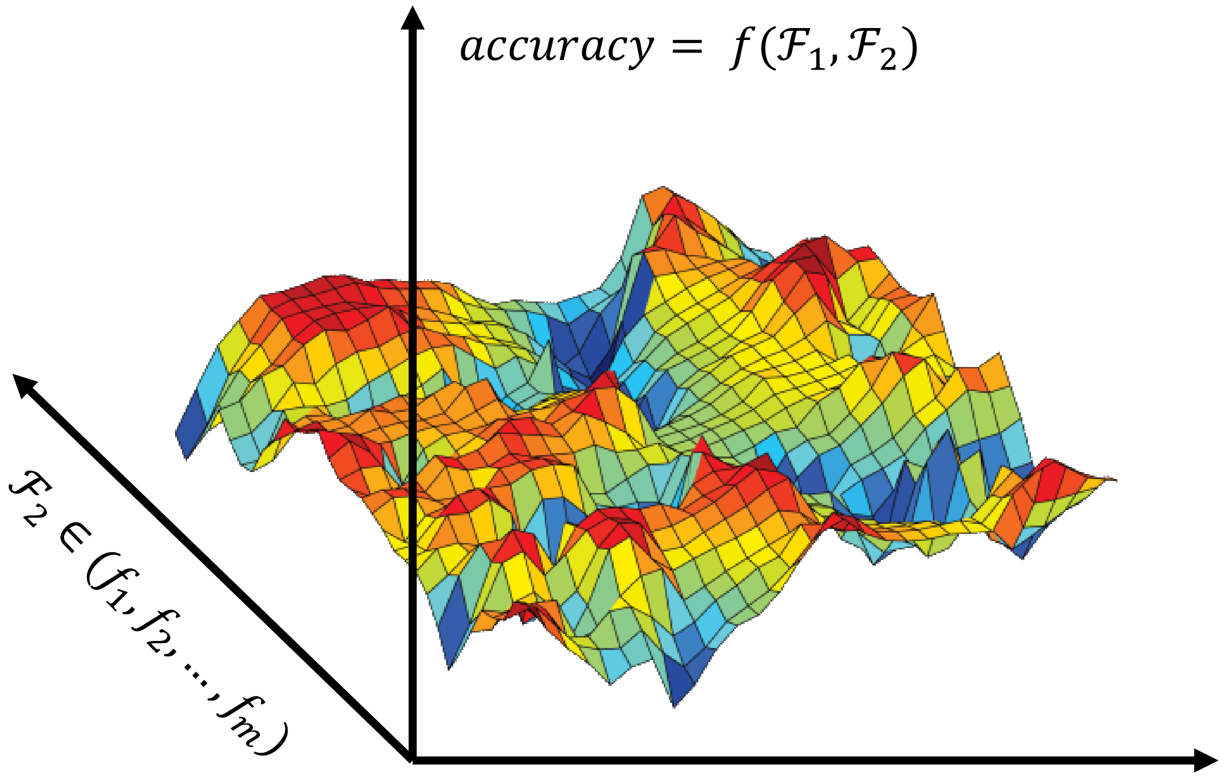
Another example: calculating π



- Draw *N* points within *A* (a unit square space) uniformly at random
- Count the *R* points for which $x^2 + y^2 < 1$
- Compute the ratio *R/N*, which converges towards $\pi/4$ as *N* increases

Monte Carlo optimisation

for feature selection


$accuracy = f(\mathcal{F}_1, \mathcal{F}_2)$
$\mathcal{F}_2 \in (f_1, f_2, \dots, f_m)$

$\mathcal{F}_1 \in (f_1, f_2, \dots, f_m)$

$\mathcal{F}_1 \neq \mathcal{F}_2$

Optimise accuracy function

$$\begin{tikzcd}
\mathcal{F}_1 = f_2, \mathcal{F}_2 = f_8 \arrow[dr] & \\
\mathcal{F}_1 = f_{18}, \mathcal{F}_2 = f_6 \arrow[r] & \boxed{accuracy = f(\mathcal{F}_1, \mathcal{F}_2)} \\
\mathcal{F}_1 = f_{22}, \mathcal{F}_2 = f_{17} \arrow[ur] & \\
\mathcal{F}_1 = f_3, \mathcal{F}_2 = f_{14} \arrow[uur] & \\
\vdots &
\end{tikzcd}$$

Demonstration

Important points

- One needs to pay special attention to separating feature selection from classification model validation. Two layers of data partitioning is often required to ensure this (recall the double layer cross-validation described in last lecture).

- Monte Carlo methods are very slow to converge.

- Heuristic algorithms such as Genetic Algorithm (GA) is faster than pure Monte Carlo optimisation and can loosely be considered as an “improved” Monte Carlo-based methods.

Improve from pure Monte Carlo optimisation: Genetic Algorithms

- A **genetic algorithm** (or **GA**) is a search technique used in computing to find true or approximate solutions to optimisation and search problems.

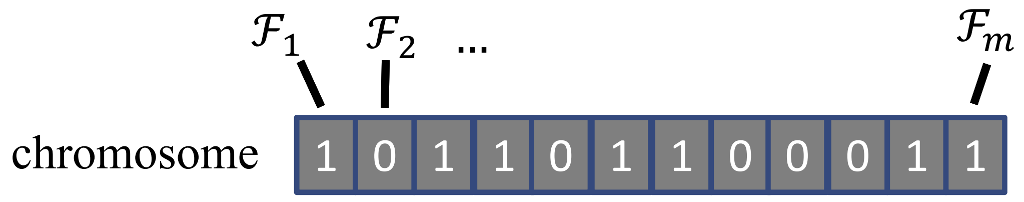
- GA are categorised as global search heuristics and can loosely be considered as an "improved" Monte Carlo-based methods because it also involves stochastic processes.

- Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination).

Coding a problem in GA

- Genetic algorithms are implemented as a computer simulation in which a population of abstract representations (called chromosomes) of candidate solutions (called individuals) to an optimisation problem evolves toward better solutions.

- Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also commonly used.



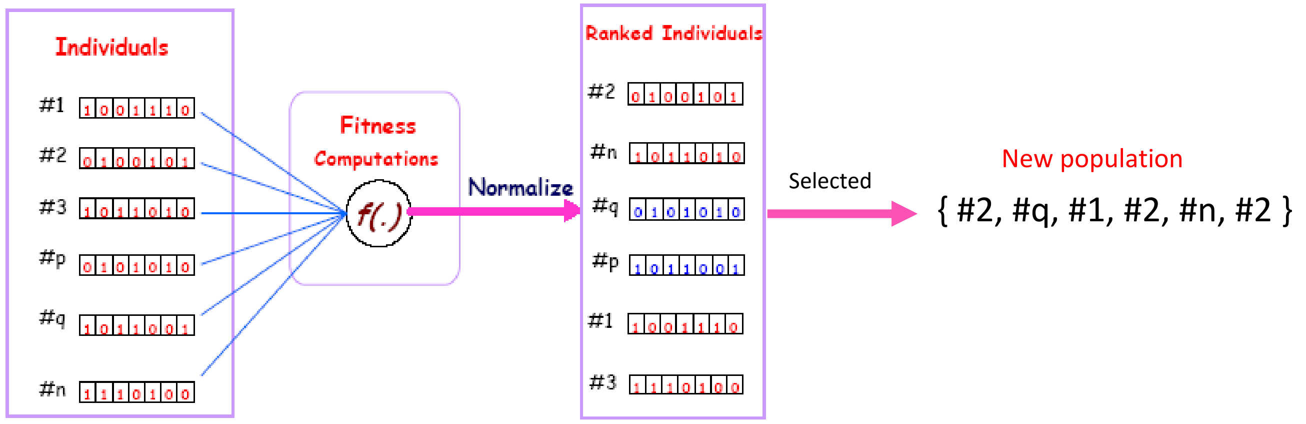
1: selected
0: excluded

GA in brief

- The evolution usually starts from a population of randomly generated individuals and happens in generations.

1 0 1 1 0 1 1 0 0 0 1 1   0 0 1 0 1 1 1 0 0 1 0 1   …

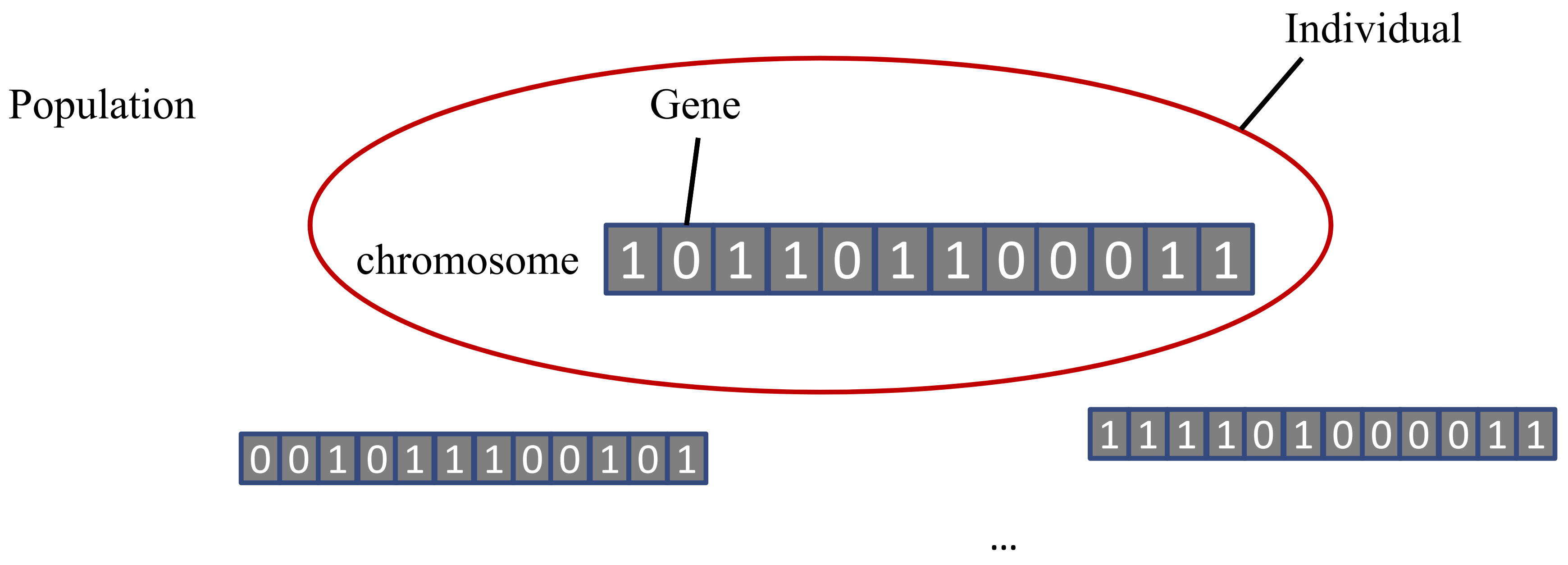
- In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified (crossover and mutated) to form a new population.



- The new population is then used in the next iteration of the algorithm.
- Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

Key terms

- **Chromosome** - Blueprint for an *individual*
- **Gene** - Possible aspect (*features*) of a *chromosome*
- **Individual** - Any possible solution
- **Population** - Group of all *individuals*
- **Search Space** - All possible solutions to the problem



GA Requirements

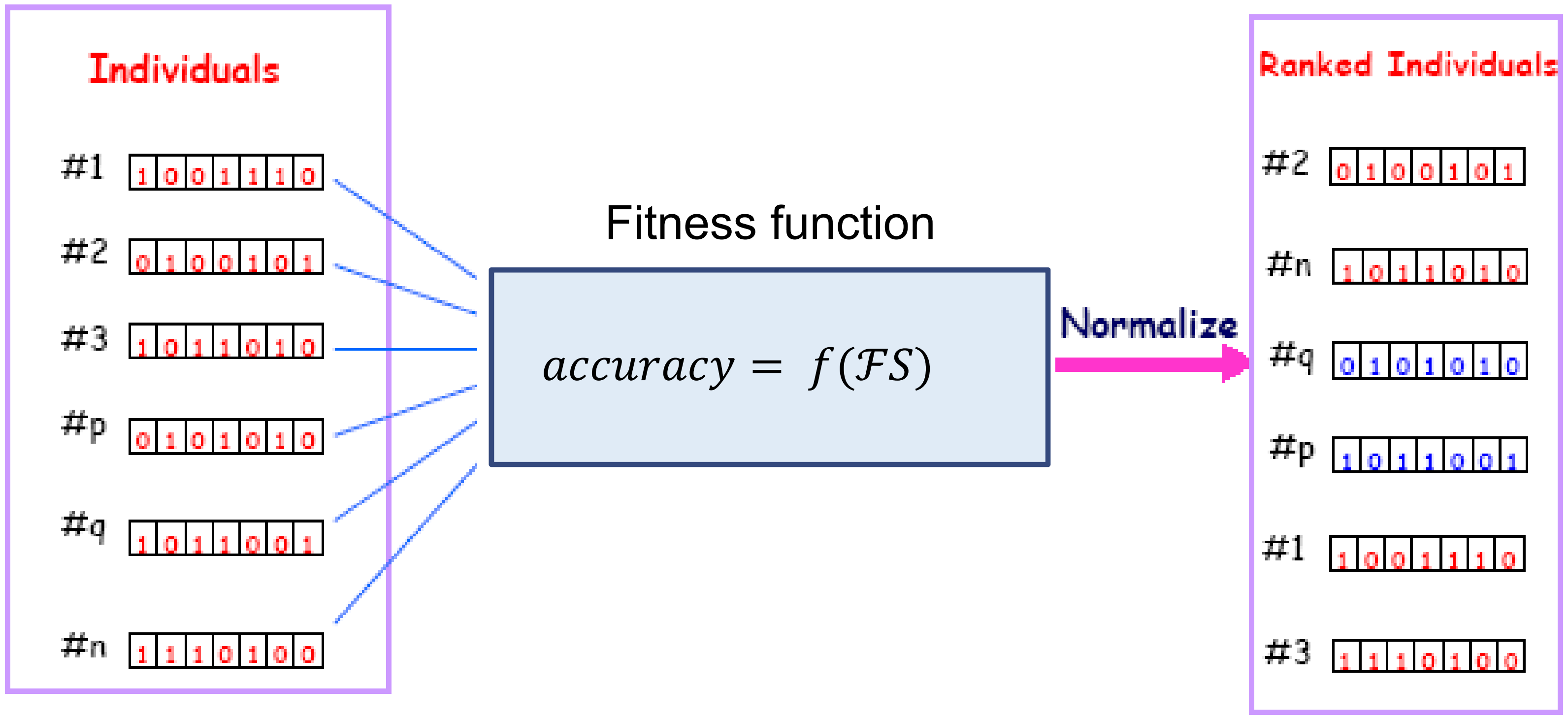
- A typical genetic algorithm requires two things to be defined:
  - A chromosome representation of the solution domain, and
  - A fitness function to evaluate the solution domain. (e.g. in the feature selection problem we want to maximize the classification accuracy or some sort of metric of a classifier)
- A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not an object is included.
- The main property that makes these genetic representations convenient is to set the chromosome to a fixed size which allows them to be easily aligned due to their fixed size and therefore facilitates simple crossover operation.
- However, variable length representations may also be used.

More on chromosome representation

Chromosomes could also be represented as:

– Bit strings	(0, 1, 0, 1, ... 1, 1, 0, 0)
– Real numbers	(43.2, -33.1, ... 0.0, 89.2)
– Permutations of element	(E11, E3, E7, ... E1, E15)
– Lists of rules	(R1, R2, R3, ... R22, R23)
– Program elements	(genetic programming)
– ... any data structure ...	

Defining a fitness function



Descriptive procedure of GA

The most common type of genetic algorithm works like this:

- A population is created with a group of individuals created randomly.
- The individuals in the population are then evaluated.
- The evaluation function is provided by the user and gives the individuals a score based on how well they perform at the given task.
- Two individuals are then selected based on their fitness, the higher the fitness, the higher the chance of being selected.
- These individuals then “reproduce” to create one or more offspring, after which the offspring are mutated randomly.
- This continues until a suitable solution has been found or a certain number of generations have passed, depending on the needs of the user.

Key component of GA in details: Initialization

- **Initialization**
  - Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions.
  - Traditionally, the population is generated randomly, covering the entire range of possible solutions (the *search space*).
  - Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

0	0	1	0	1	1	1	0	0	1	0	1

1	1	1	1	0	1	0	0	0	0	1	1

…

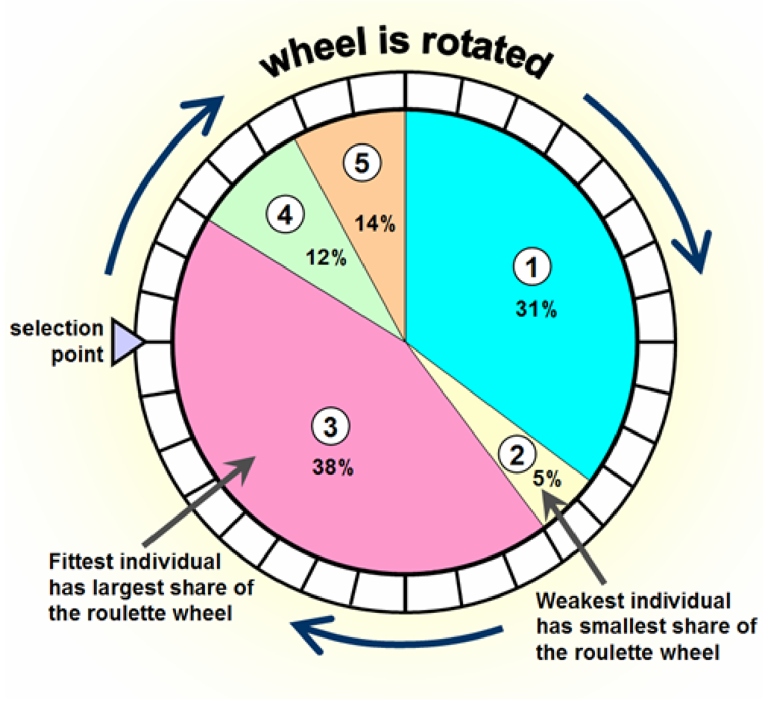
Key component of GA in details: Fitness evaluation and selection

- **Fitness evaluation and selection**
  - During each successive generation, a proportion of the existing population is selected to breed a new generation.
  - Individual solutions are selected through a *fitness-based* process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected.
  - Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population, preventing premature convergence on poor solutions. Popular selection methods include *roulette wheel selection* and *tournament selection*.

Roulette wheel selection

- In roulette wheel selection, individuals are given a probability of being selected that is directly proportionate to their fitness.

- Two individuals are then chosen randomly based on these probabilities and to produce offspring.


wheel is rotated
5
14%
4
12%
1
31%
selection point
3
38%
2
5%
Fittest individual has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel

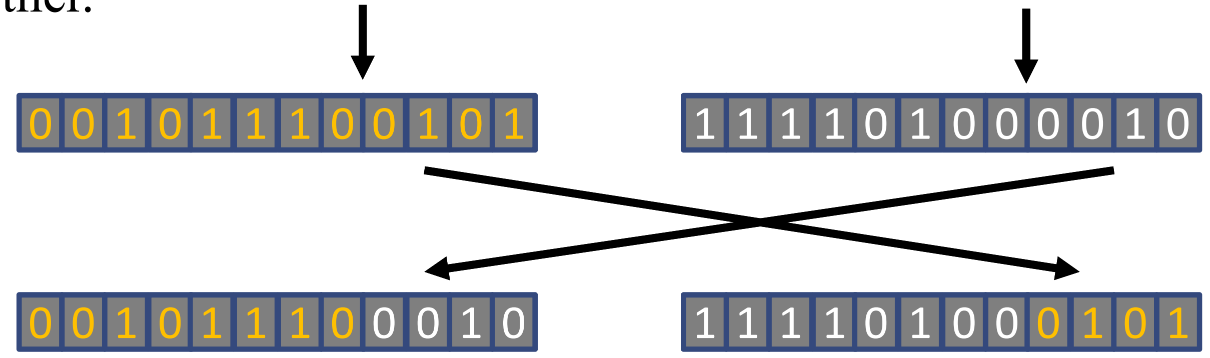
Key component of GA in details: Reproduction

- **Reproduction**
  - The next step is to generate a second generation of solutions from those selected through genetic operators:

    crossover and ***mutation***.
  - For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously.
  - By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each child, and the process continues until a new population of solutions of appropriate size is generated.
  - These processes ultimately result in the next generation population of chromosomes that is different (and likely improves) from the initial generation.

Crossover

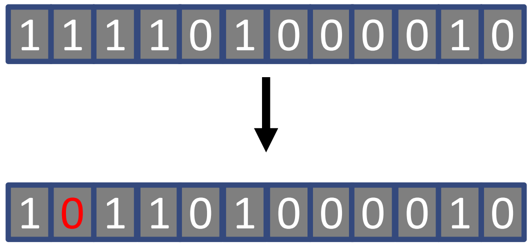
- the most common type is single point crossover. In single point crossover, you choose a locus at which you swap the remaining genes from on parent to the other.



- As you can see, the children take one section of the chromosome from each parent.
- The point at which the chromosome is broken depends on the randomly selected crossover point.
- This particular method is called single point crossover because only one crossover point exists. Sometimes only child 1 or child 2 is created, but oftentimes both offspring are created and put into the new population.
- Crossover does not always occur, however. Sometimes, based on a set probability, no crossover occurs and the parents are copied directly to the new population. The probability of crossover occurring is usually set from 60% to 70%.

Mutation

- After selection and crossover, you now have a new population full of individuals.
- Some are directly copied, and others are produced by crossover.
- In order to ensure that the individuals are not all exactly the same, you allow for a small chance of mutation.
- You loop through all the genes of all the individuals, and if that gene is selected for mutation, you can either change it by a small amount or replace it with a new value. The probability of mutation is usually between 1 and 2 tenths of a percent.
- Mutation is fairly simple. You just change a randomly selected gene from 0 to 1 or the other way around. Mutation is, however, important to ensuring genetic diversity within the population.

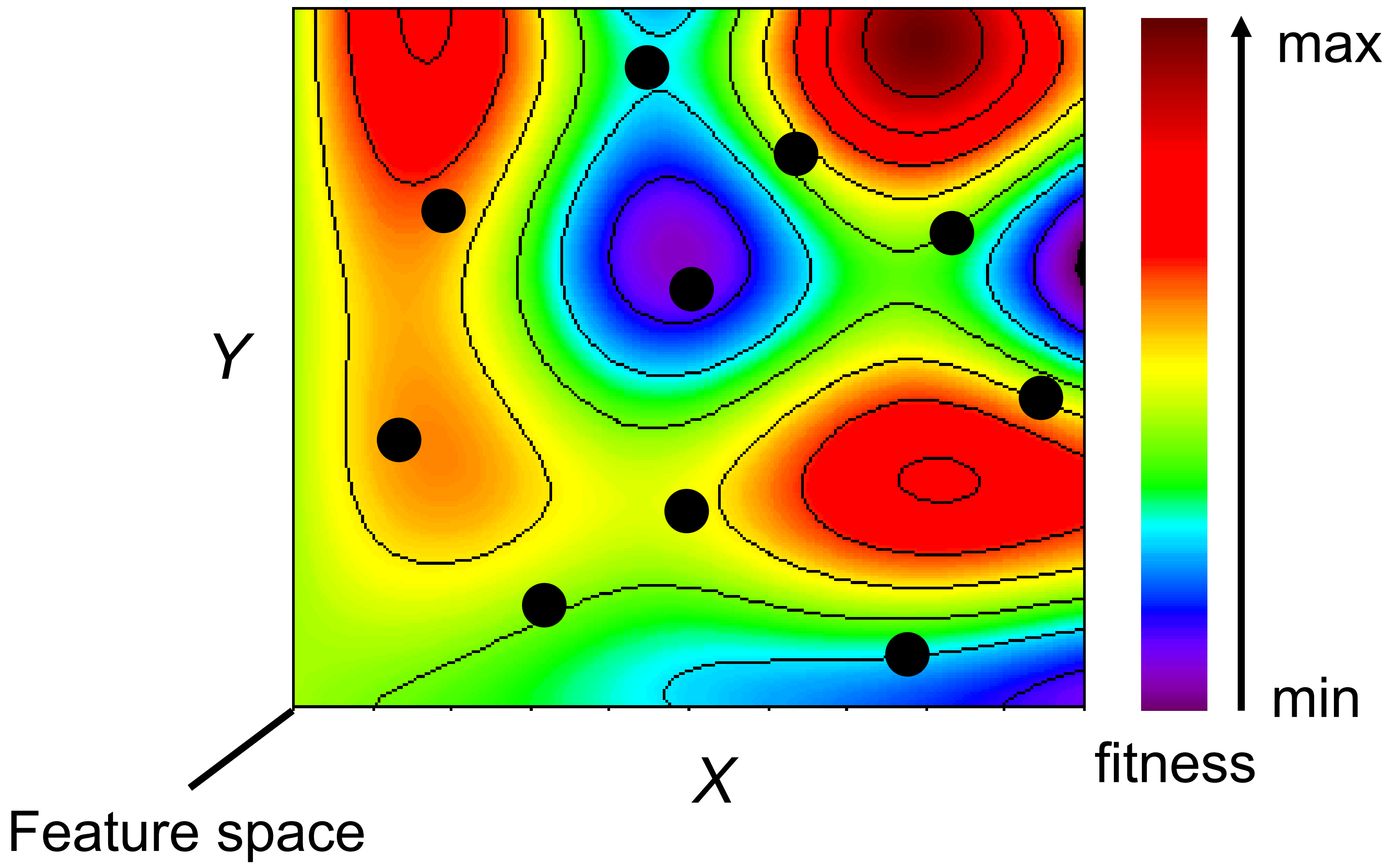


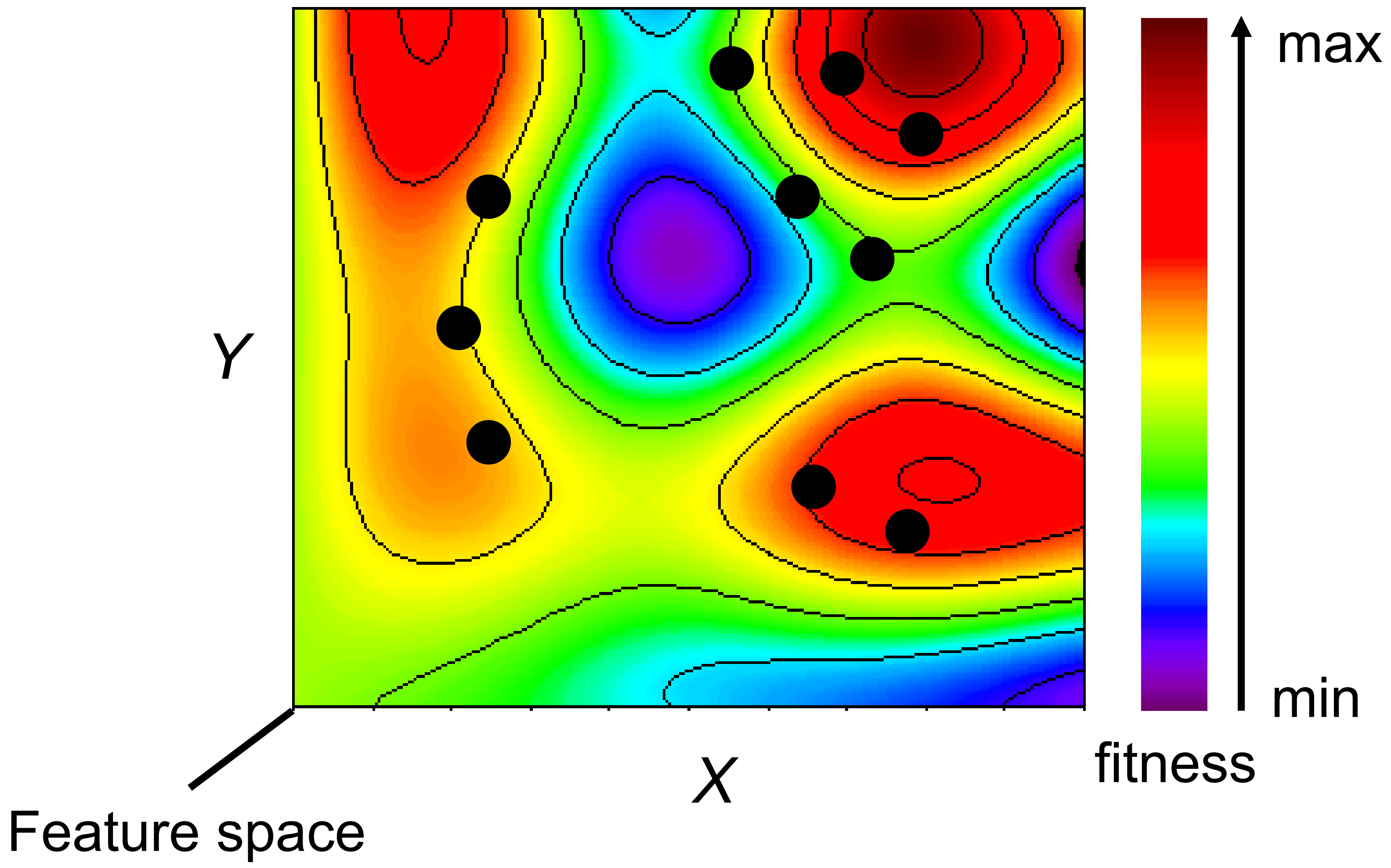
Termination

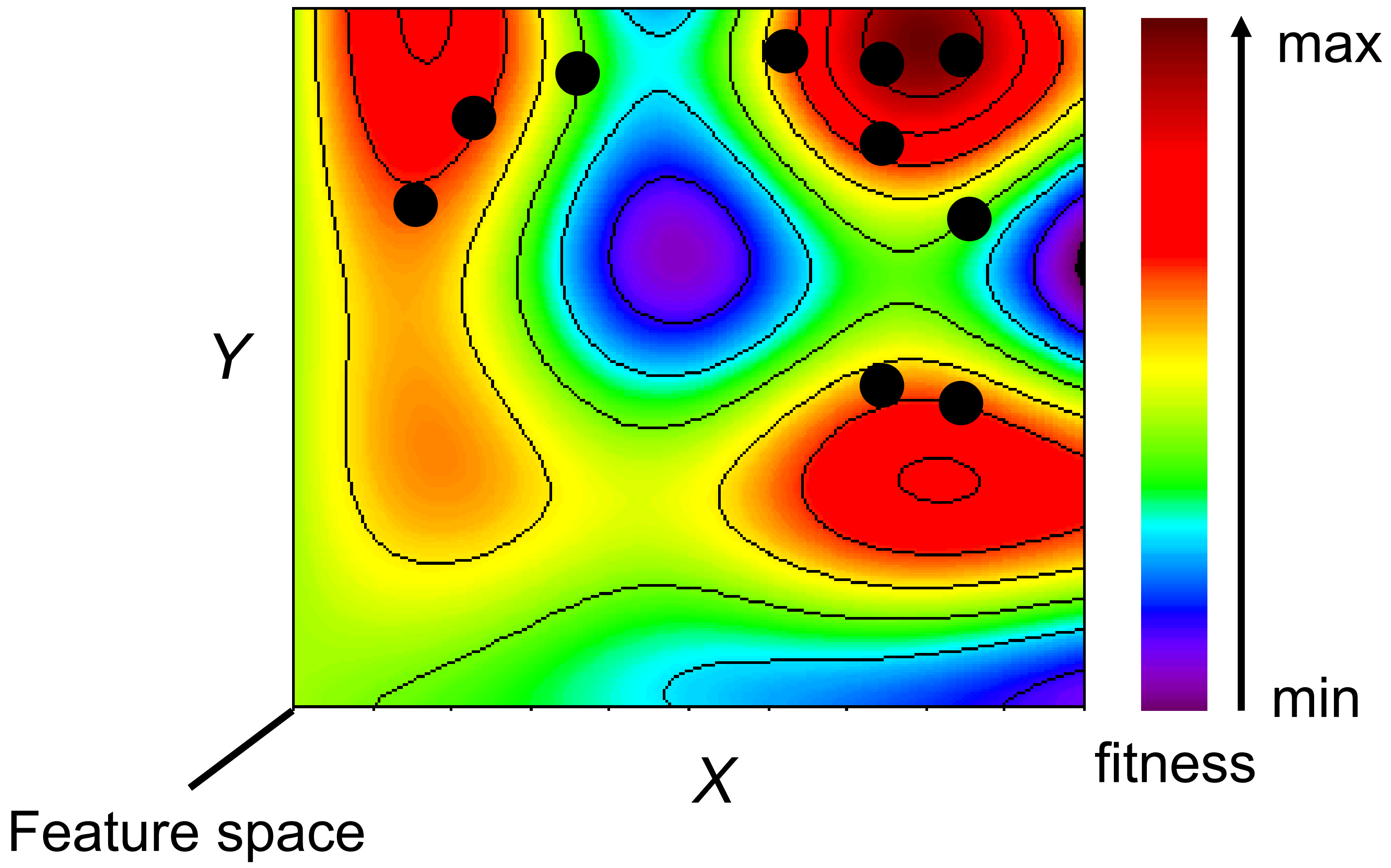
- GA generational process is repeated until a termination condition has been reached.
- Common terminating conditions are:
  - A solution is found that satisfies minimum criteria
  - Fixed number of generations reached
  - Allocated budget (computation time/money) reached
  - The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
  - Manual inspection
  - Any combinations of the above

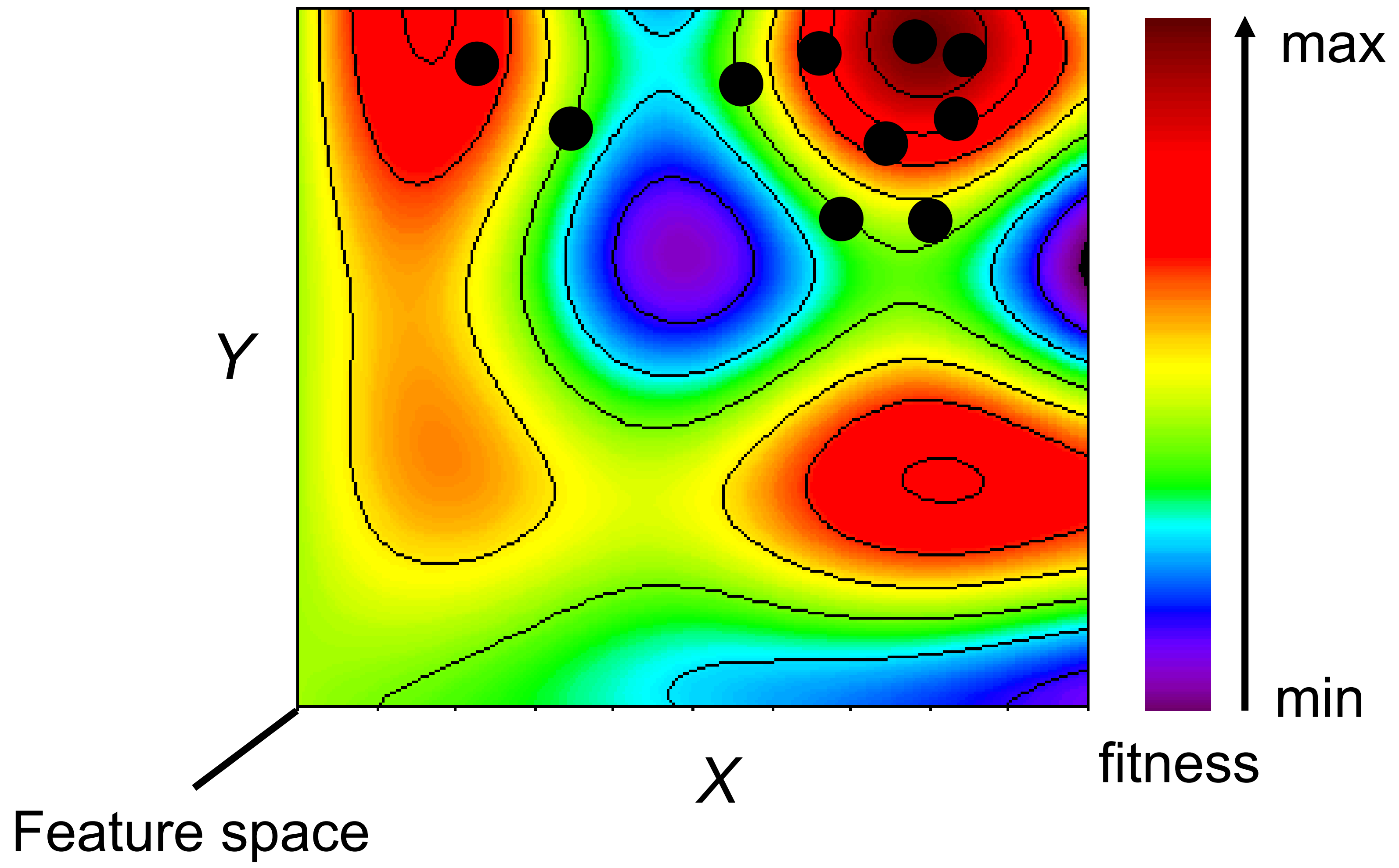
GA summary in pseudo-code

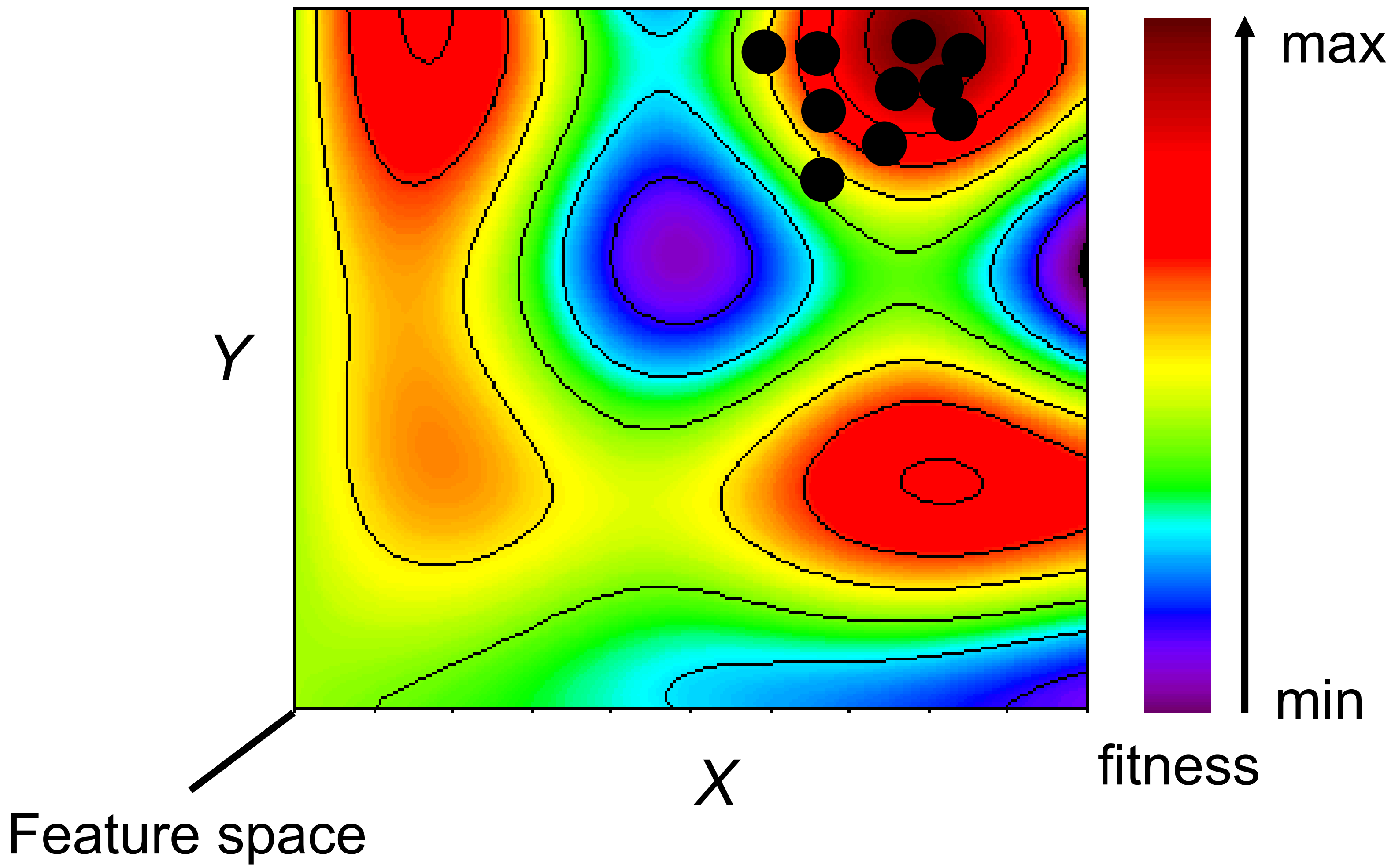
```
Choose initial population

Repeat

    Evaluate the fitness of each individual in the population

    Select best-ranking individuals to reproduce

    Breed new generation through crossover and mutation

Until <terminating condition>
```


Y
X
max
min
fitness
Feature space

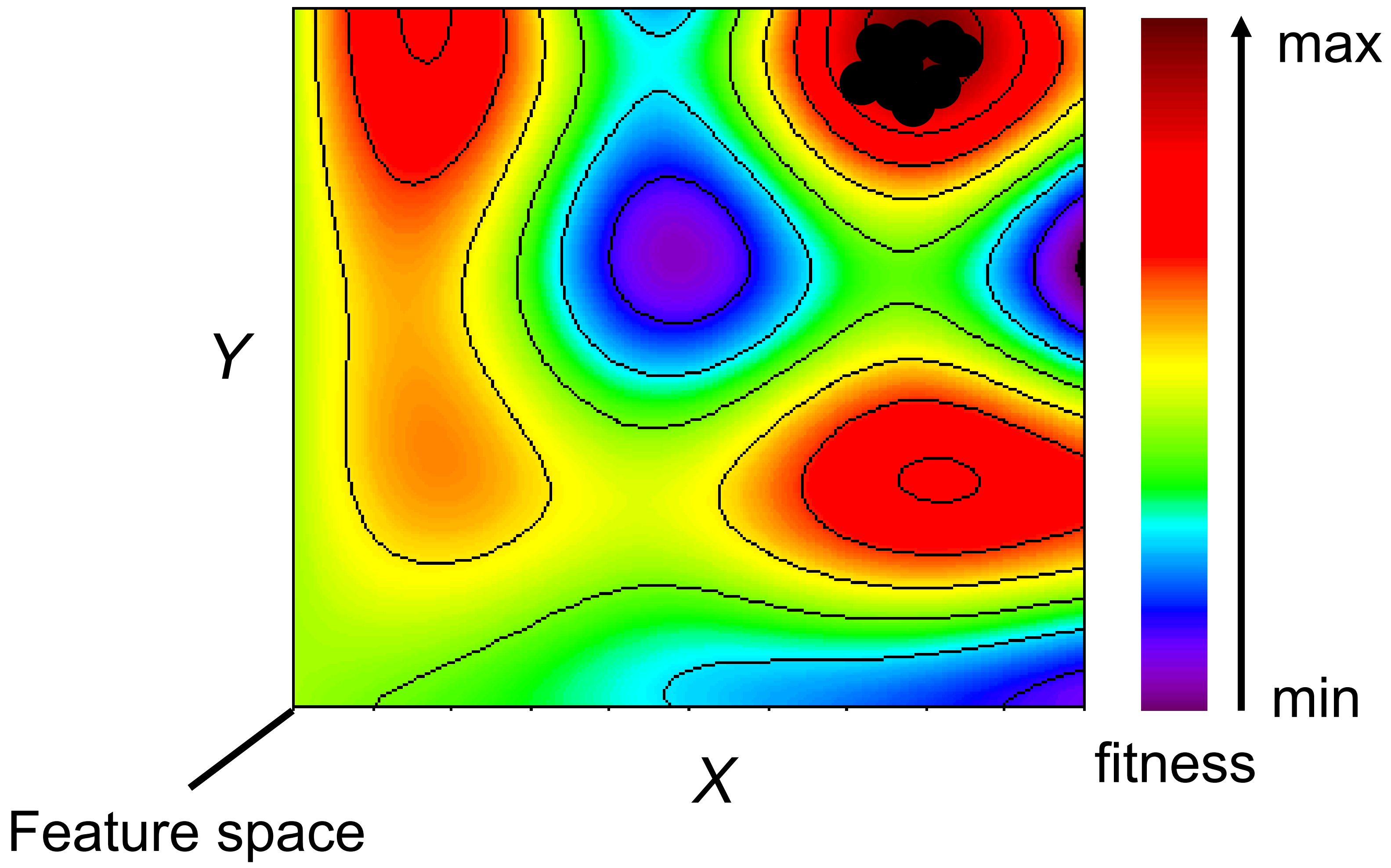

max
min
fitness
Y
X
Feature space


max
min
fitness
Y
X
Feature space


Y
X
max
min
fitness
Feature space


max
min
fitness
Y
X
Feature space

Demonstration


max
min
fitness
Y
X
Feature space