# Some Extra Contents

Lecturer : Bijay Mishra

# What kind of data preprocessing do we need before applying data mining algorithm to any dataset?

Data preprocessing techniques are applied before mining. These can improve the overall quality of the patterns mined and the time required for the actual mining.

Some important data preprocessing that must be needed before applying data mining algorithm to any data sets are:

1. **Data cleaning**
2. **Data integration**
3. **Data transformation**
4. **Data reduction**
5. **Data discretization**

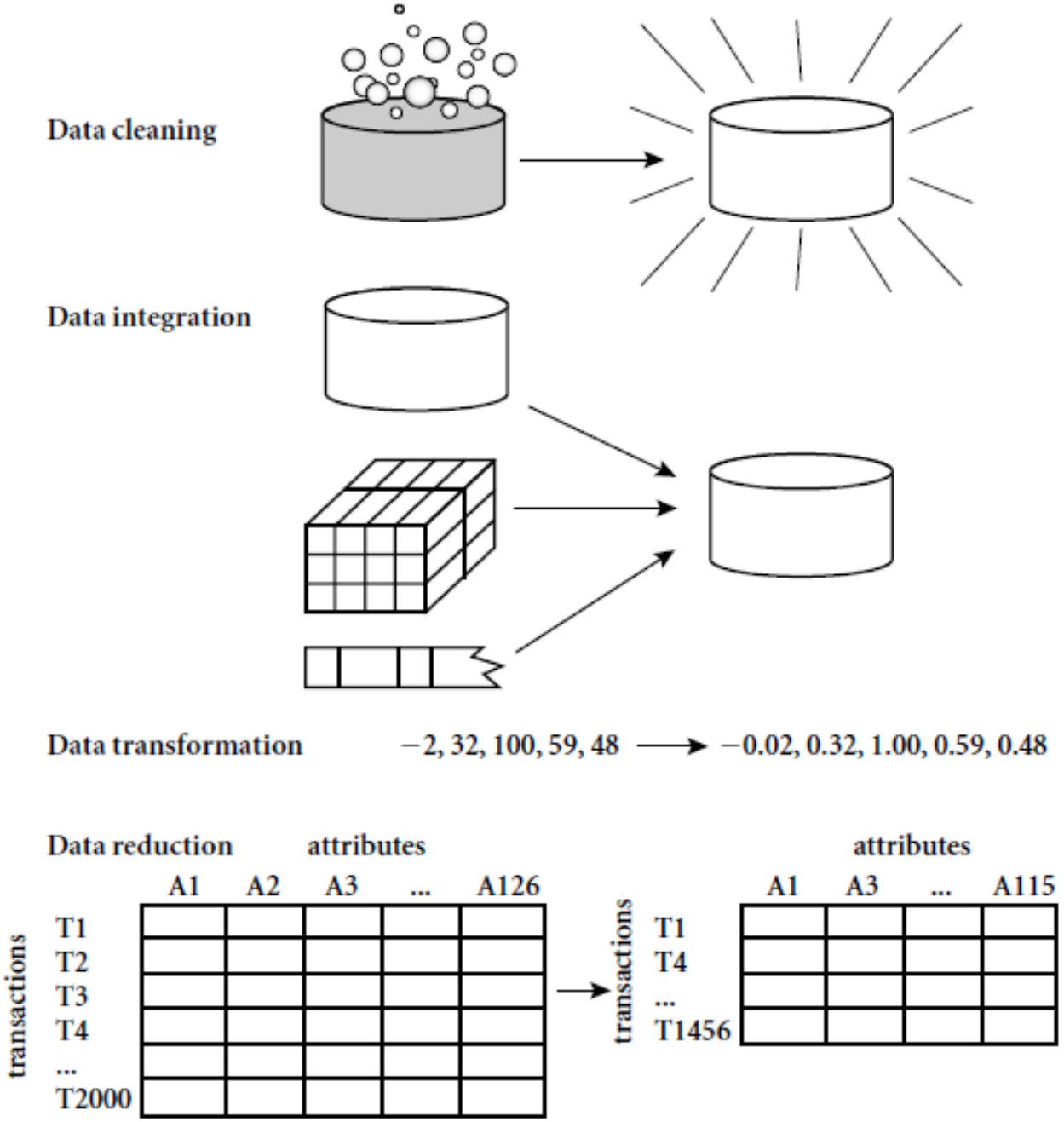



**Figure:** Forms of data preprocessing.

# Major Tasks in Data Preprocessing

| Technique | Purpose |
|---|---|
| **1. Data cleaning** | It can be applied to remove noise and correct inconsistencies in the data. |
| **2. Data integration** | It merges data from multiple sources into a coherent data store, such as a data warehouse. |
| **3. Data transformation** | These are like normalizations. |
| **4. Data reduction** | It can reduce the data size by aggregating, eliminating redundant features, or clustering. |
| **5. Data discretization** | Part of data reduction but with particular importance, especially for numerical data |

# Explain binning method to handle noisy data with example.

**Noise** is a random error or variance in a measured variable. Given a numerical attribute such as, say, *price*, how can we "smooth" out the data to remove the noise?

**Solution:**

**Binning method:** Binning methods smooth a sorted data value by consulting its "neighborhood," i.e. the values around it. The sorted values are distributed into a number of "buckets," or *bins*.

# Binning Methods for Data Smoothing

Let's look at the following data smoothing techniques:

**Sorted data for price (in dollars):** 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

* Partition into (equal frequency) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
* Smoothing by bin means:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
* Smoothing by bin boundaries:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34

In **smoothing by bin means**, each value in a bin is replaced by the mean value of the bin. For example, the mean of the values 4, 8, and 15 in Bin 1 is 9. Therefore, each original value in this bin is replaced by the value 9.

Similarly, **smoothing by bin medians** can be employed, in which each bin value is replaced by the bin median.

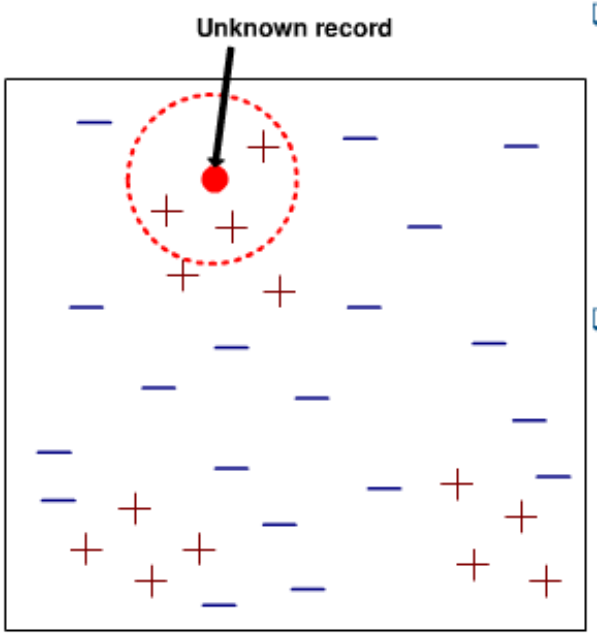
In **smoothing by bin boundaries**, the minimum and maximum values in a given bin are identified as the *bin boundaries*. Each bin value is then replaced by the closest boundary value. In general, the larger the width, the greater the effect of the smoothing.

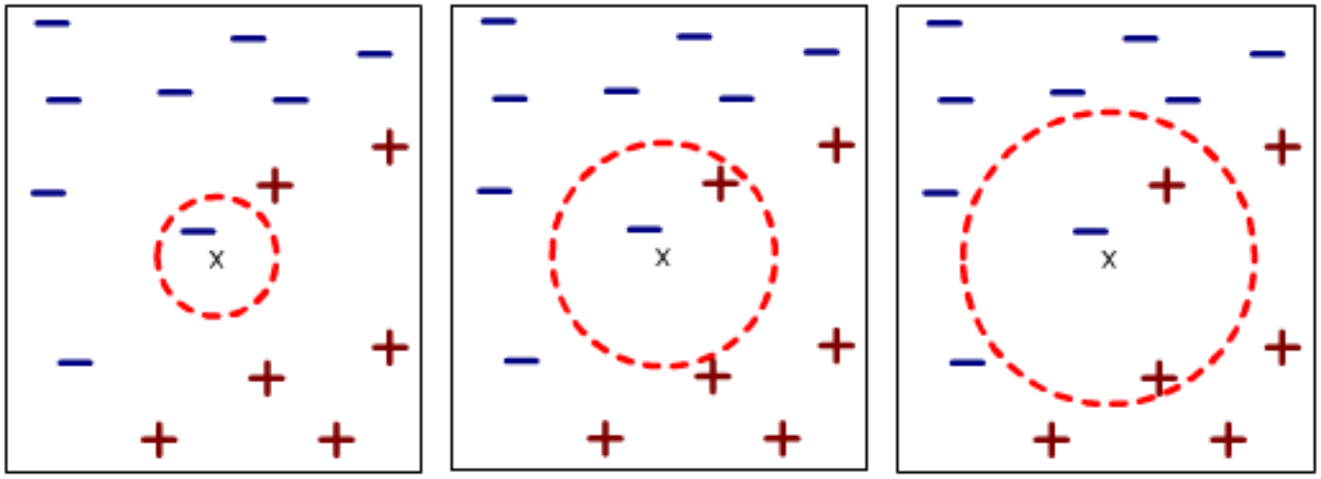
# Simple Discretization Methods: Binning

- Equal-width (distance) partitioning:
  - It divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the attribute, the width of intervals will be: $W = (B-A)/N$.
  - The most straightforward
  - But outliers may dominate presentation
  - Skewed data is not handled well.
- Equal-depth (frequency) partitioning:
  - It divides the range into *N* intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky.

# The *k*-Nearest Neighbor Algorithm

- The *k*-nearest-neighbor method was first described in the early 1950s.
- The *k*-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors (*k* is a positive integer, typically small).
- If *k* = 1, then the object is simply assigned to the class of its nearest neighbor.




(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

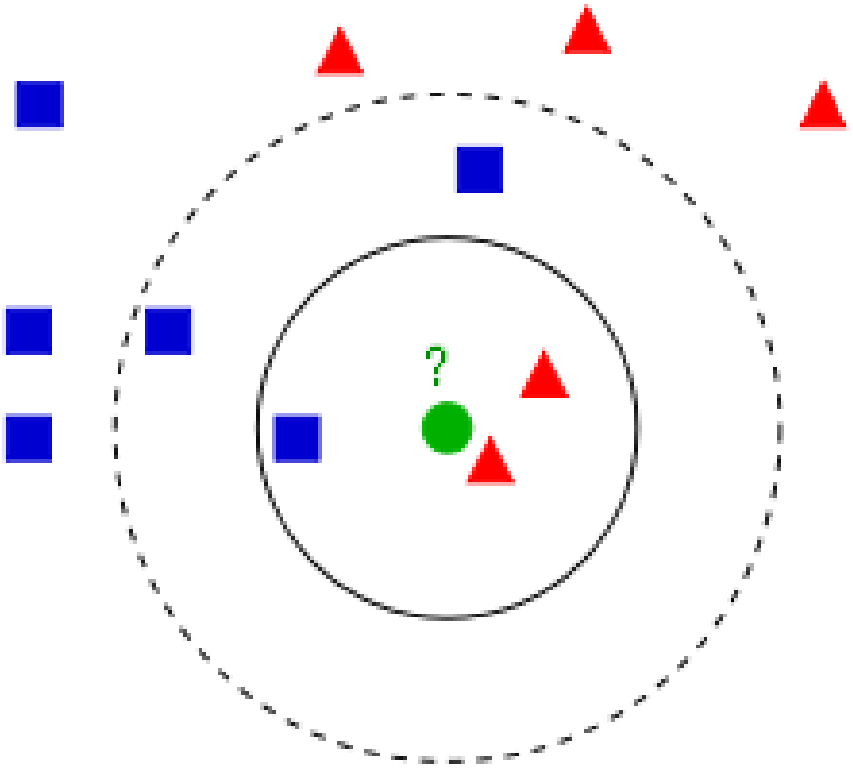



**Figure: Example of *k*-NN classification.**

The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If *k = 3* it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If *k = 5* it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

# Genetic Algorithms

- Genetic Algorithms are *search* and *optimization* techniques based on Darwin's Principle of *Natural Selection*.
- Directed search algorithms based on the mechanics of biological evolution
- John Holland wrote the first book on Genetic Algorithms 'Adaptation in Natural and Artificial Systems' in 1975.
- In 1992 **John Koza** used genetic algorithm to evolve programs to perform certain tasks. He called his method "Genetic Programming".
- Provide efficient, effective techniques for optimization and machine learning applications

# Genetic Algorithm

```
{
    initialize population;
    evaluate population;
    while TerminationCriteriaNotSatisfied
    {
        select parents for reproduction;
        perform recombination and mutation;
        evaluate population;
    }
}
```

# Another Simple Genetic Algorithm

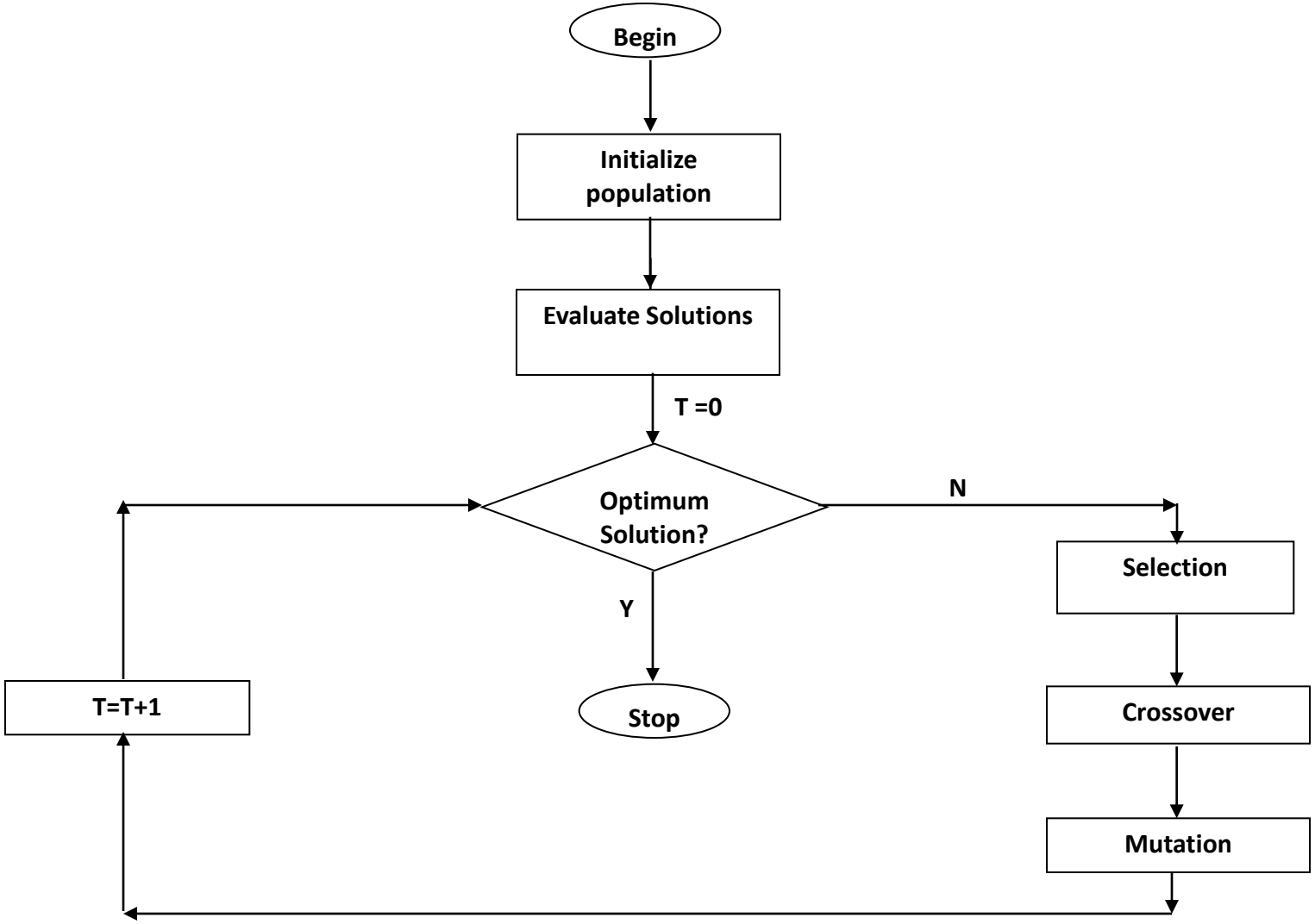
```
Simple_Genetic_Algorithm()
        {
        Initialize the Population;
        Calculate Fitness Function;

        While(Fitness Value != Optimal Value)
                {
                Selection;//Natural Selection, Survival Of Fittest

                Crossover;//Reproduction, Propagate favorable
    characteristics

                Mutation;//Mutation
                Calculate Fitness Function;
                }
         }
```

# Working Mechanism Of Genetic Algorithms



GAs implement optimization techniques by simulating this natural law of evolution in the biological world.

We start with a population of randomly generated solutions. Each of these solutions is evaluated to determine how good or bad it is. In other words to determine how “fit” that solution is. We then check a terminating condition, to see if our solutions are good enough? If yes, we stop. If not, we have to optimize the solutions.

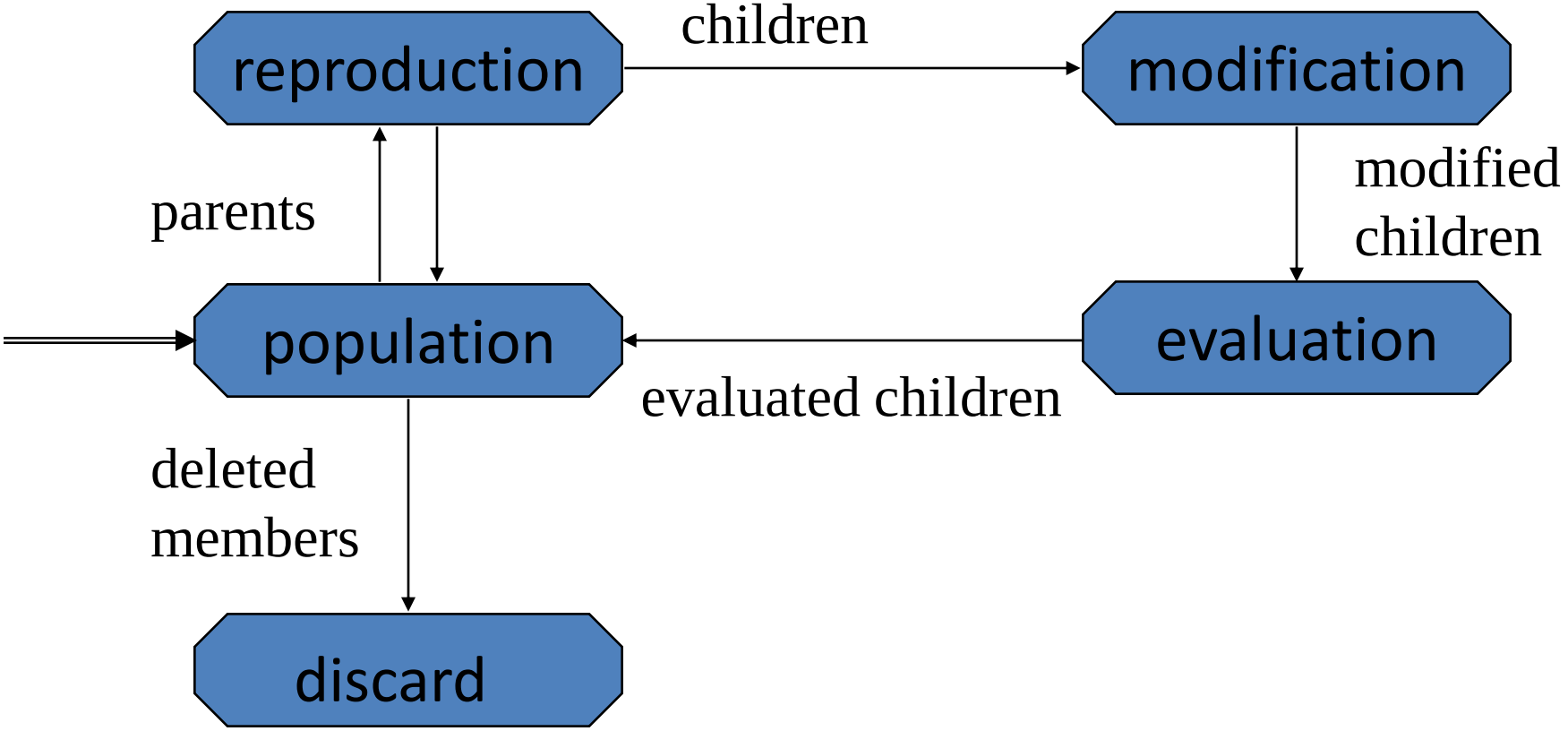
So, we select the best solutions from the initial population (selection). This is similar to “natural selection”.

Then we allow these good solutions to exchange their information, in order to get even better solutions. This step is similar to reproduction among animals or crossover among chromosomes and is called “cross-over”. We may then randomly mutate some small % of the solutions thus obtained after crossover.

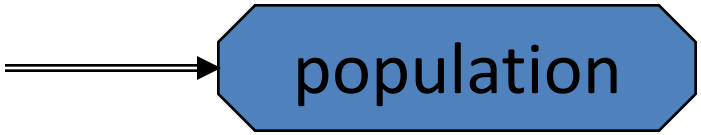
Mutation is very important. It could be a bad thing, it could be a good thing. In the biological sense it mean, making a small change in a gene. In GAs, it means, making a small change to the solution.

Then again, we evaluate each of the solutions, and check the termination condition. As you see , this is an optimization method.

# The Genetic Algorithm Cycle of Reproduction



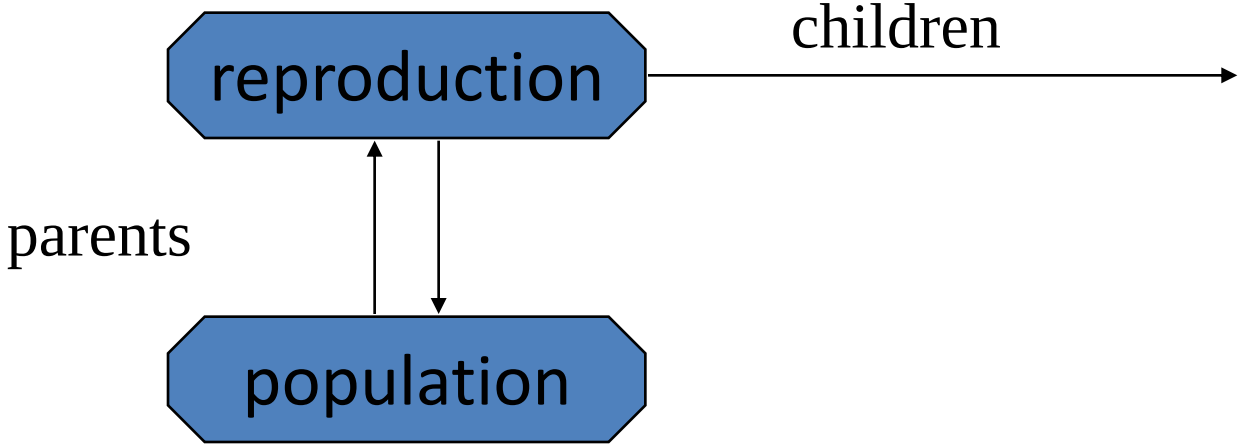
# Population




Chromosomes could be:

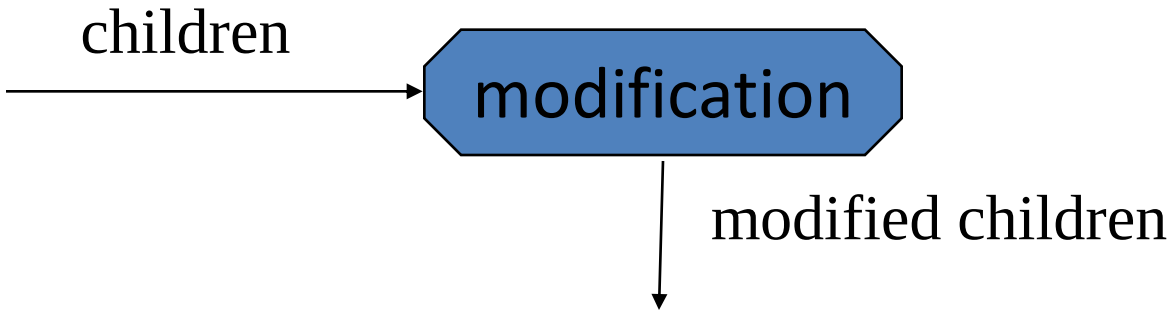
- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

# Reproduction




Parents are selected at random with selection chances biased in relation to chromosome evaluations.

# Chromosome Modification




- Modifications are stochastically triggered
- Operator types are:
  - Mutation
  - Crossover (recombination)

# Crossover Example

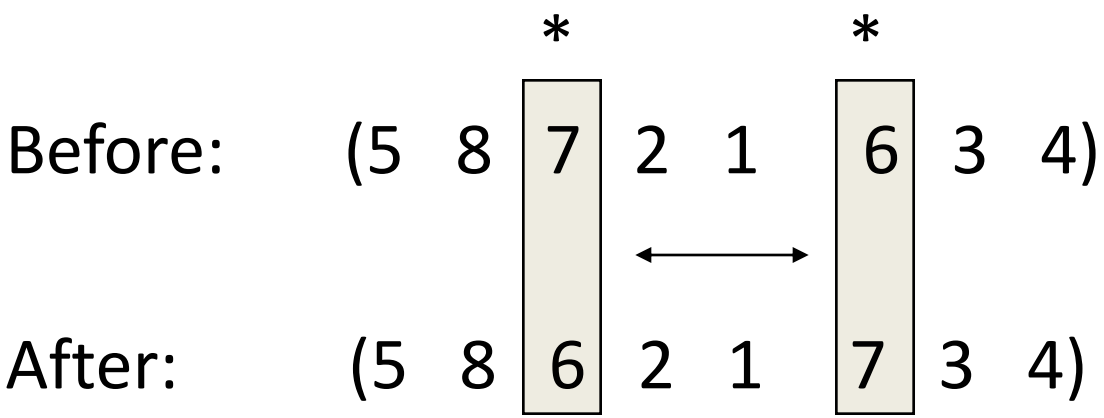
Crossover combines inversion and recombination:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | * | | | | * | | |
| Parent1 | (3 | 5 | 7 | 2 | 1 | 6 | 4 | 8) |
| Parent2 | (2 | 5 | 7 | 6 | 8 | 1 | 3 | 4) |
| Child | (5 | 8 | 7 | 2 | 1 | 6 | 3 | 4) |

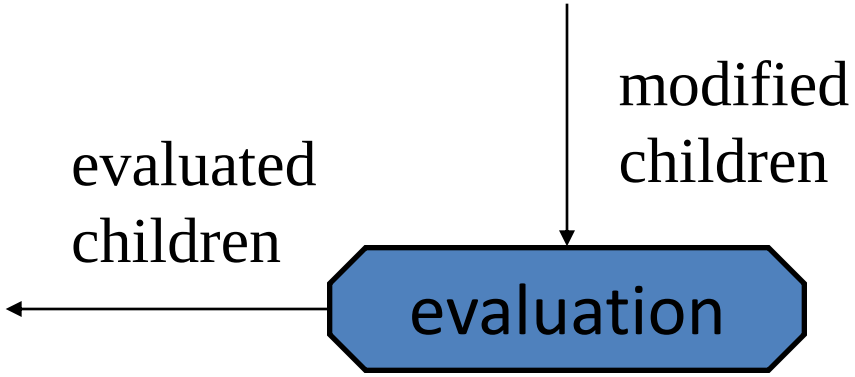
This operator is called the *Order1* crossover.

# Mutation Example

Mutation involves reordering of the list:

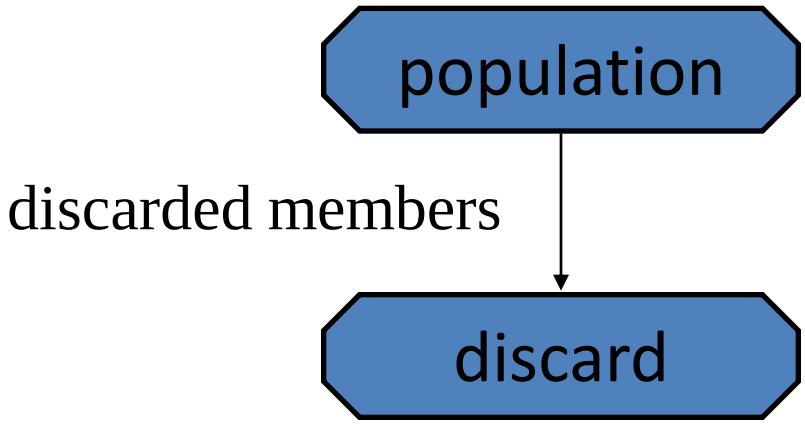


# Evaluation




- The evaluator decodes a chromosome and assigns it a fitness measure
- The evaluator is the only link between a classical GA and the problem it is solving

# Deletion




- *Generational* GA:
  entire populations replaced with each iteration
- *Steady-state* GA:
  a few members replaced each generation

# GA Advantages/Disadvantages

- **Advantages**
  - Easily parallelized
- **Disadvantages**
  - Difficult to understand and explain to end users.
  - Abstraction of the problem and method to represent individuals is quite difficult.
  - Determining fitness function is difficult.
  - Determining how to perform crossover and mutation is difficult.

# Text Indexing Techniques

**Inverted index**

- Maintains two hash- or B+-tree indexed tables:
  - document_table: a set of document records <doc_id, postings_list>
  - term_table: a set of term records, <term, postings_list>
- Answer query: Find all docs associated with one or a set of terms
- + easy to implement
- – do not handle well synonymy and polysemy, and posting lists could be too long (storage could be very large)

**Signature file**

- Associate a signature with each document
- A signature is a representation of an ordered list of terms that describe the document
- Order is obtained by frequency analysis, stemming and stop lists

# More Exercises

(Q) Suppose that the data for analysis include the attribute age. The age values for the data tuples are (in increasing order) : 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

Use smoothing by bin means to smooth the above data, using a bin depth of 3. Illustrate your steps. Comment on the effect of this technique for the given data.

**Solution:**

The following steps are required to smooth the above data using smoothing by bin means with a bin depth of 3.

**Step 1:**
Sort the data. (This step is not required here as the data are already sorted.)

Step 2:
Partition the data into equi-depth bins of depth 3.
Bin 1: 13, 15, 16
Bin 2: 16, 19, 20
Bin 3: 20, 21, 22
Bin 4: 22, 25, 25
Bin 5: 25, 25, 30
Bin 6: 33, 33, 35
Bin 7: 35, 35, 35
Bin 8: 36, 40, 45
Bin 9: 46, 52, 70

Step 3:
Calculate the arithmetic mean of each bin.

Step 4:
Replace each of the values in each bin by the arithmetic mean calculated for the bin.
Bin 1: 142/3, 142/3, 142/3
Bin 2: 181/3, 181/3, 181/3
Bin 3: 21, 21, 21
Bin 4: 24, 24, 24
Bin 5: 262/3, 262/3, 262/3
Bin 6: 332/3, 332/3, 332/3
Bin 7: 35, 35, 35
Bin 8: 401/3, 401/3, 401/3
Bin 9: 56, 56, 56

# Thank you !!!