Experiment No. 2

Problem Statement - Optimization of genetic algorithm parameter in hybrid genetic algorithm-neural network modelling: Application to spray drying of coconut milk.

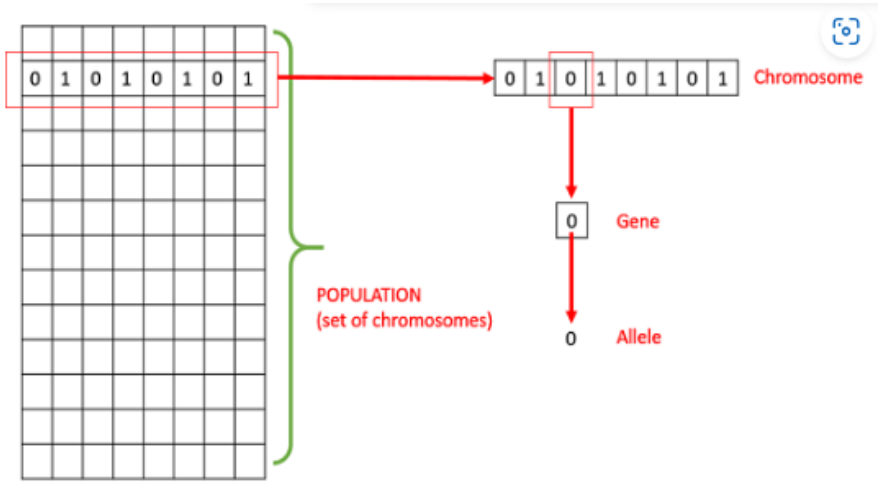
Aim: Optimization of genetic algorithm parameter in hybrid genetic algorithm-neural network modeling: Application to spray drying of coconut milk.

Objective: Optimization of genetic algorithm parameter

Theory:

Genetic Algorithm (GA) is a search-based optimization technique based on the principles of Genetics and Natural Selection. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve. It is frequently used to solve optimization problems, in research, and in machine learning.

Genetic Algorithms are heuristic optimization algorithms inspired by the process of natural selection and genetics. They are widely used for solving optimization problems. In hybrid genetic algorithm-neural network modeling, a genetic algorithm is often used to optimize the weights and biases of a neural network.



Parameters of Genetic Algorithm:

• **Population Size (pop_size)**: The number of individuals in the population.

• **Mutation Rate (mutation_rate):** The probability that a gene (parameter) will be mutated during reproduction.

• **Crossover Rate (crossover_rate):** The probability that two individuals will exchange genetic information during crossover.

• **Selection Method:** The method used to select individuals for reproduction, such as tournament selection or roulette wheel selection.

• Termination Criteria: The condition under which the genetic algorithm terminates, such as reaching a maximum number of generations or achieving a desired fitness level.

Optimization:

Optimization refers to finding the values of inputs in such a way that we get the “best” output values**.** Optimizing genetic algorithm parameters involves finding the combination of parameter values that results in the best performance of the hybrid model. This often requires experimentation and fine-tuning of parameters to achieve optimal results.

Implementation:

Steps for Optimization:

• Initialization: Initialize the genetic algorithm parameters with initial values.

• Create Hybrid Model: Implement the hybrid genetic algorithm-neural network model.

• Parameter Tuning: Experiment with different values of genetic algorithm parameters.

• Evaluation: Evaluate the performance of the hybrid model for each set of parameter values.

• Optimization: Select the parameter values that yield the best performance based on predefined criteria.

• Validation: Validate the optimized parameters on a separate dataset to ensure generalization.

Conclusion: Thus, we have successfully implemented hybrid genetic algorithm.

Questions:

1.What is the inspiration behind genetic algorithms?
2.What are chromosomes, genes, and alleles?
3.What is the genetic algorithm cycle?
4.What is the importance of selection in genetic algorithms?

Code and Output:

Experiment 2

In [12]:

```
import random
from deap import base, creator, tools, algorithms

# Define evaluation function (this is a mock function, replace this with your actual evaluation function)
def evaluate(individual):
    # Here 'individual' represents the parameters for the neural network
    # You'll need to replace this with your actual evaluation function that
    #trains the neural network and evaluates its performance
    # Return a fitness value (here, a random number is used as an example)
    return random.random(),

# Define genetic algorithm parameters
POPULATION_SIZE = 10
GENERATIONS = 5

# Create types for fitness and individuals in the genetic algorithm
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

# Initialize toolbox
toolbox = base.Toolbox()

# Define attributes and individuals
toolbox.register("attr_neurons", random.randint, 1, 100)  # Example: number of neurons
toolbox.register("attr_layers", random.randint, 1, 5)  # Example: number of layers
toolbox.register("individual", tools.initCycle, creator.Individual, (toolbox.attr_neurons, toolbox.attr_layers), n=1)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

# Genetic operators
toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutUniformInt, low=1, up=100, indpb=0.2)
toolbox.register("select", tools.selTournament, tournsize=3)
```

```
# Create initial population
population = toolbox.population(n=POPULATION_SIZE)

# Run the genetic algorithm
for gen in range(GENERATIONS):
    offspring = algorithms.varAnd(population, toolbox, cxpb=0.5, mutpb=0.1)

    fitnesses = toolbox.map(toolbox.evaluate, offspring)
    for ind, fit in zip(offspring, fitnesses):
        ind.fitness.values = fit

    population = toolbox.select(offspring, k=len(population))


best_individual = tools.selBest(population, k=1)[0]
best_params = best_individual

# Print the best parameters found
print("Best Parameters:", best_params)
```

```
Best Parameters: [93, 1]
```