Learning

Chapter 4

What is Learning?

“Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task (or tasks drawn from the same population) more effectively the next time.” --Herbert Simon

"Learning is constructing or modifying representations of what is being experienced." -- Ryszard Michalski

"Learning is making useful changes in our minds." --Marvin Minsky

Types of Learning

The strategies for learning can be classified according to the amount of inference the system has to perform on its training data. In increasing order we have

1. Rote learning – the new knowledge is implanted directly with no inference at all, e.g. simple memorisation of past events, or a knowledge engineer's direct programming of rules elicited from a human expert into an expert system.

2. Supervised learning – the system is supplied with a set of training examples consisting of inputs and corresponding outputs, and is required to discover the relation or mapping between then, e.g. as a series of rules, or a neural network.

3. Unsupervised learning – the system is supplied with a set of training examples consisting only of inputs and is required to discover for itself what appropriate outputs should be, e.g. a *Kohonen Network* or *Self Organizing Map*.

Early expert systems relied on rote learning, but for modern AI systems we are generally interested in the supervised learning of various levels of rules.

Need for Learning

As with many other types of AI system, it is much more efficient to give the system enough knowledge to get it started, and then leave it to learn the rest for itself. We may even end up with a system that learns to be better than a human expert.

The ***general learning approach*** is to generate potential improvements, test them, and discard those which do not work. Naturally, there are many ways we might generate the potential improvements, and many ways we can test their usefulness. At one extreme, there are model driven (top-down) generators of potential improvements, guided by an understanding of how the problem domain works. At the other, there are data driven (bottom-up) generators, guided by patterns in some set of training data.

Machine Learning

Branch of AI that use algorithms to allow computer to evolve behaviours based on data collected from database or gathered through sensors

Focuses on prediction, based on known properties learned from the training data

The performance is usually evaluated with respect to the ability to reproduce known knowledge

Machine Learning

As regards machines, we might say, very broadly, that a machine learns whenever it changes its structure, program, or data (based on its inputs or in response to external information) in such a manner that its expected future performance improves.

Some of these changes, such as the addition of a record to a data base, fall comfortably within the province of other disciplines and are not necessarily better understood for being called learning. But, for example, when the performance of a speech-recognition machine improves after hearing several samples of a person's speech, we feel quite justified in that case saying that the machine has learned.

Machine learning usually refers to the changes in systems that perform tasks associated with artificial intelligence (AI).

Such tasks involve recognition, diagnosis, planning, robot control, prediction, etc.

The changes might be either enhancements to already performing systems or synthesis of new systems.

Concept of Learning

Environment

- It refers to the nature and quality of the information given to the learning element
- The nature of information depends on its level
  - High level: information is abstract and deals with broad class of problems
  - Low level: information is detailed and deals with single problem
- Quality of information involves noise free, ordered and reliable information

Concept of Learning

Learning Element

- Acquire new knowledge through learning element
- Learning may be of any type discussed above

Problem Generator/ Knowledge Base

- Stores the information about the problems and solutions are suggested
- Knowledge Base should be
  - Expressive: Knowledge should be represented in easy and understandable way
  - Modifiable: Must be easy to update or add new data in the knowledge base
  - Extendibility: the knowledge base should have feature to change its structure that should be well defined

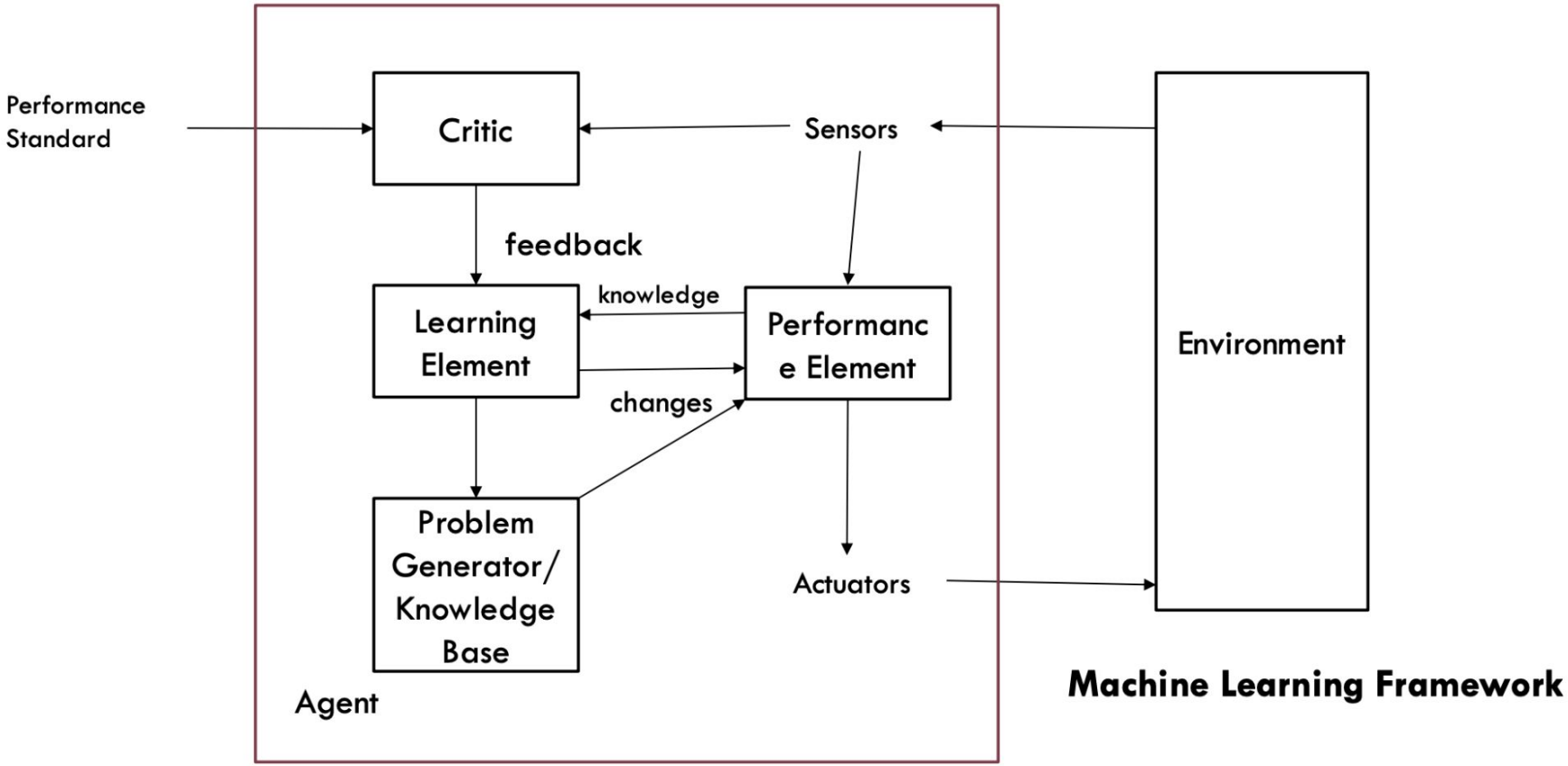
Concept of Learning

Performance Element

- This part analyses how complex the learning is and how learning is being performed
- Complexity depends on the type of task to be performed It must send feedback to the learning system as well so that evaluation of overall performance could be done
- The learning element should have access to all internal actions of the performance elements

Sensors and Actuators

- Sensors collect information from environment
- Actuators implement the suggested action



Machine Learning Framework

Rote Learning

Rote learning is a technique which focuses on memorization

It avoids understanding the inner complexities and inferences of the subject that is being learned and instead focuses on memorizing the material

The major practice involved in rote learning is repetition

Learning by explanation

Explanation-based learning (EBL) uses a domain theory to construct an explanation of the training example, usually a proof that the example logically follows from the theory

Using this proof the system filters the noise, selects only the relevant to the proof aspects of the domain theory, and organizes the training data into a systematic structure

This makes the system more efficient in later attempts to deal with the same or similar examples.

Basic Concept of EBL

- Target concept.
  - The task of the learning system is to find an effective definition of this concept.
  - Depending on the specific application the target concept could be a classification, theorem to be proven, a plan for achieving goal, or heuristic to make a problem solver more efficient.
- Training example.
  - This is an instance of the target concept
- Domain theory.
  - Usually this is a set of rules and facts representing domain knowledge.
  - They are used to explain how the training example is an instance of the target concept.
- Operationality criteria.
  - Some means to specify the form of the concept definition.

Learning by Analogy

Analogy is a powerful inference tool

Our language and reasoning are laden with analogy

Example: Last month share market was a roller coaster.

Bill is like a fire engine

So, AI must be able to grasp analogy for learning easily

It is used in different learning strategies like learning by advice taking, learning in problem solving, etc.

Learning by Analogy

Analogy is a powerful inference tool

Our language and reasoning are laden with analogy

Example: Last month share market was a roller coaster.

Bill is like a fire engine

So, AI must be able to grasp analogy for learning easily

It is used in different learning strategies like learning by advice taking, learning in problem solving, etc.

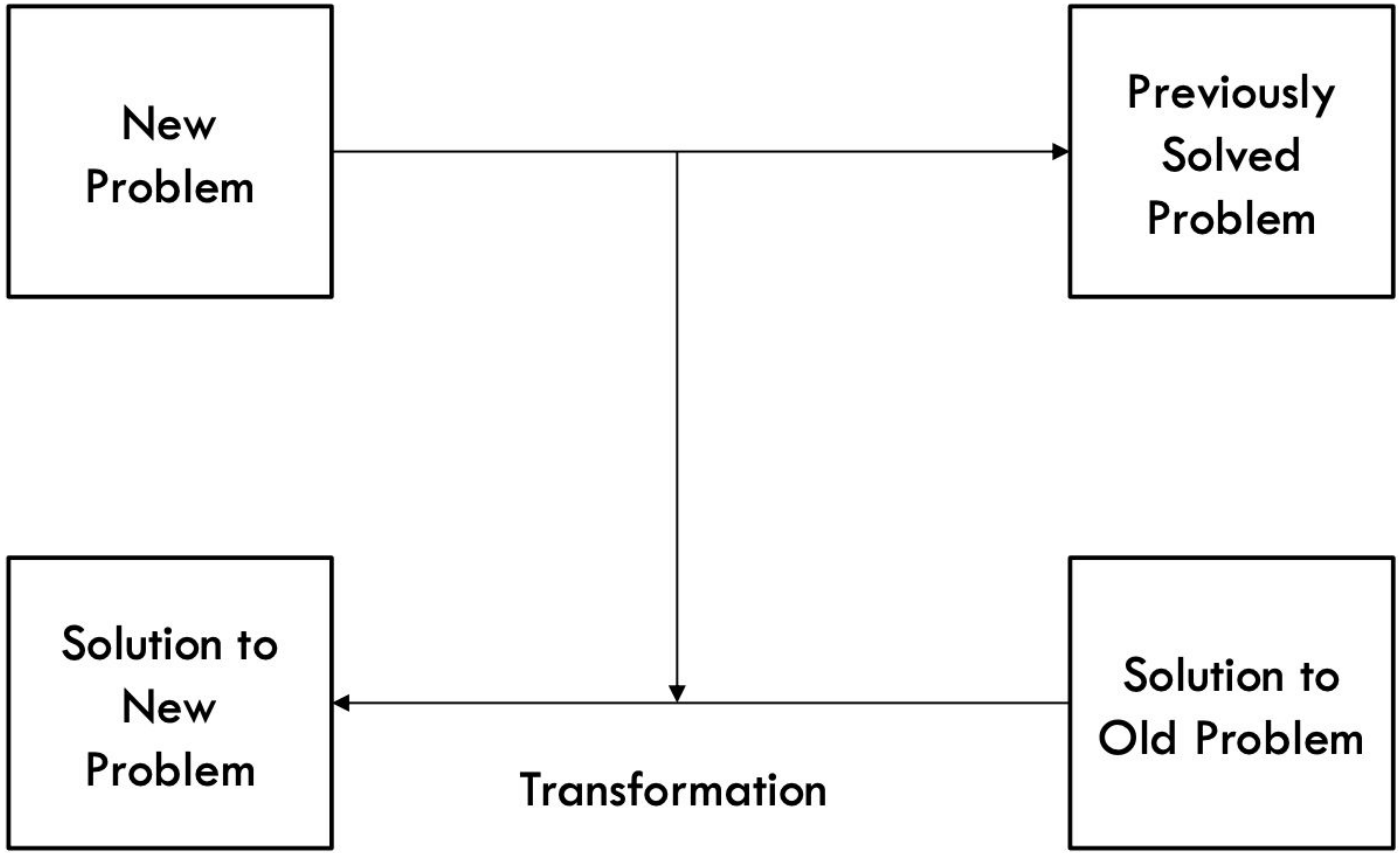
Learning by analogy

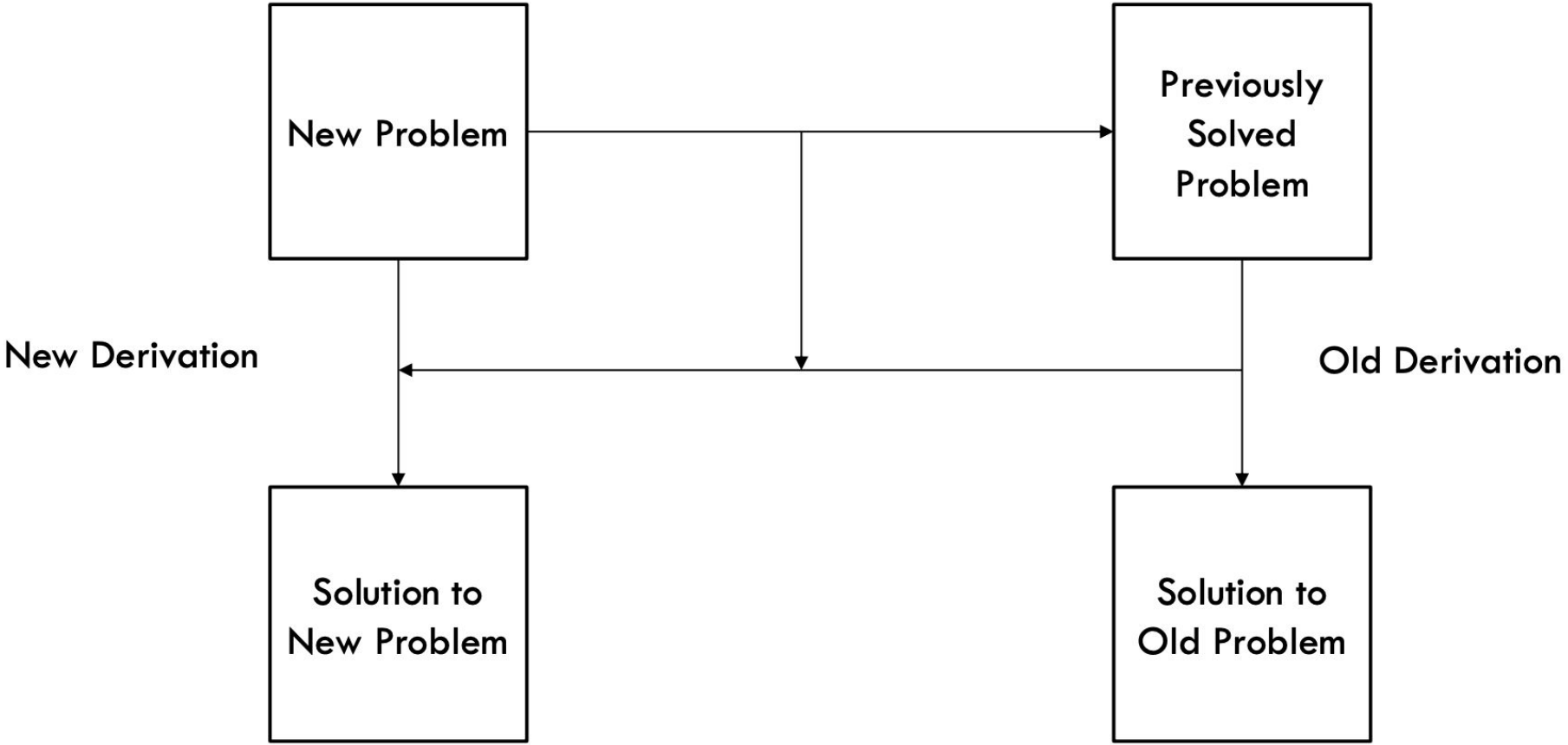
Two methods of Analogy Problem solving are:

Transformational Analogy : focuses on final solution

Derivational Analogy: focuses on process of problem solving

Transformational Analogy




New Problem
Previously Solved Problem
New Derivation
Old Derivation
Solution to New Problem
Solution to Old Problem

Inductive learning

- Simplest form: learn a function from examples

- Idea:

Given:

- *f*: the *target function*
- Examples of *f* where an *example* is a pair $(x, f(x))$ (training data) and examples might have noise

Problem:

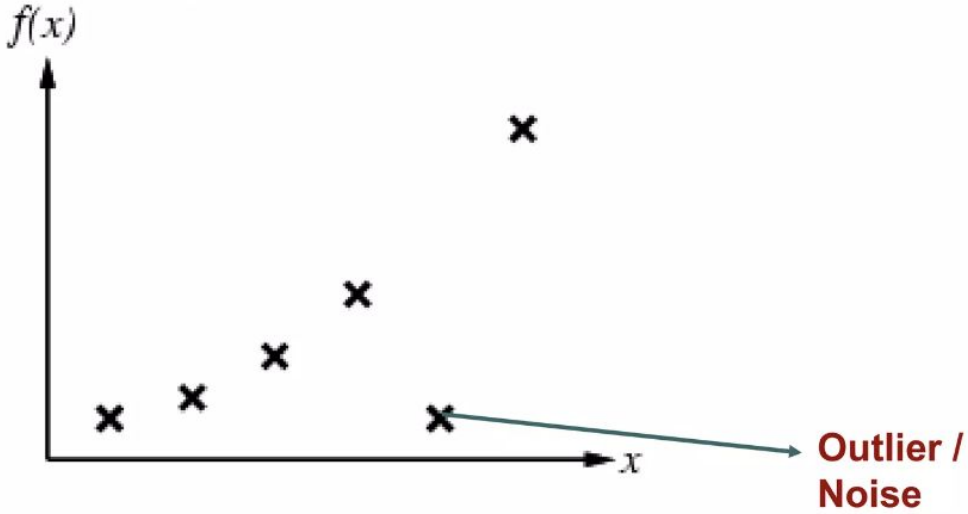
- find a *hypothesis h* such that $h \approx f$ (*h* is mostly consistent with *f*)

(This is a highly simplified model of real learning:

- Ignores prior knowledge
- Assumes that there are no missing examples)

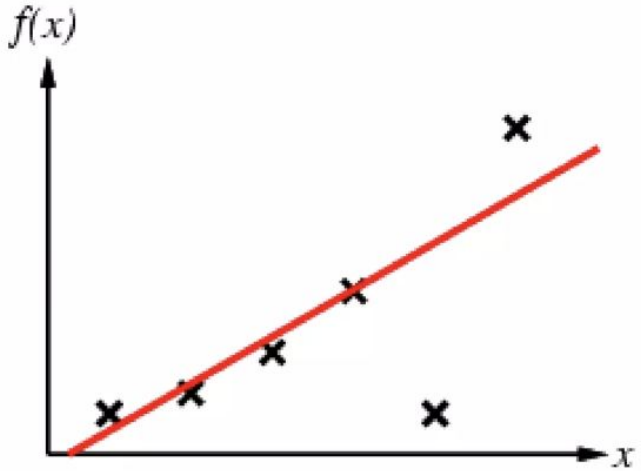
Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is ***a consistent hypothesis*** if it agrees with *f* on all examples)
- E.g., curve fitting:



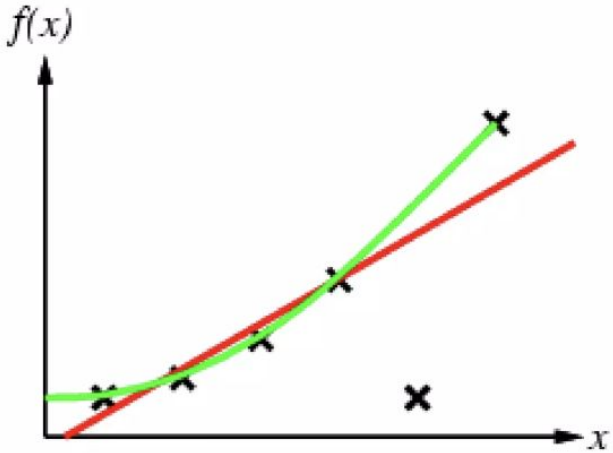
Inductive learning method (cont'd)

- Construct/adjust *h* to agree with *f* on training set
- (*h* is *consistent* if it agrees with *f* on all examples)
- E.g., curve fitting:



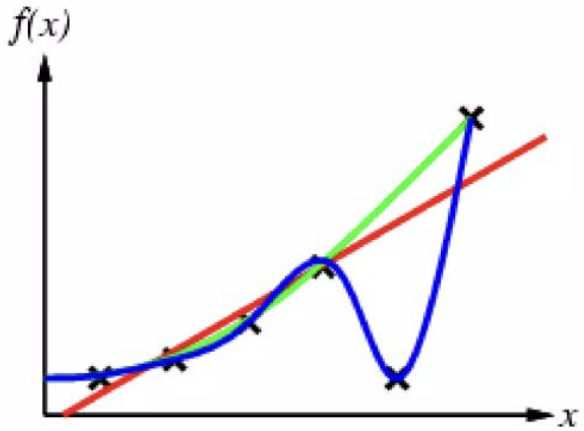
Inductive learning method (cont'd)

- Construct/adjust *h* to agree with *f* on training set
- (*h* is *consistent* if it agrees with *f* on all examples)
- E.g., curve fitting:



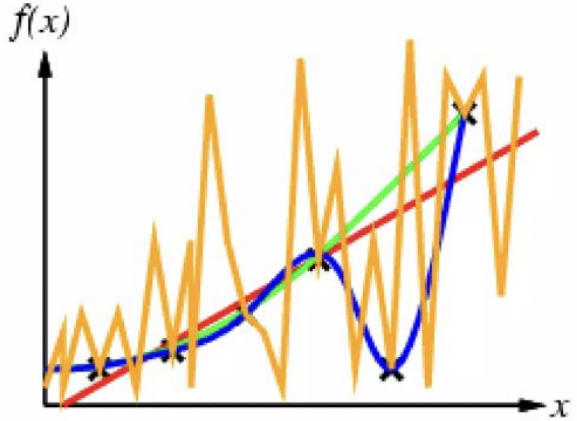
Inductive learning method (cont'd)

- Construct/adjust *h* to agree with *f* on training set
- (*h* is *consistent* if it agrees with *f* on all examples)
- E.g., curve fitting:



Inductive learning method (cont'd)

- Construct/adjust *h* to agree with *f* on training set
- (*h* is *consistent* if it agrees with *f* on all examples)
- E.g., curve fitting:



Genetic Algorithm

Algorithm: An algorithm is a sequence of instructions to solve a problem. Most of the algorithms are static.

A Genetic Algorithm(GA) is adaptive (dynamic) a model of machine learning algorithm that derives its behavior from a metaphor of some of the mechanisms of evolution in nature.

GA: Background

Evolution can be seen as a process leading to the maintenance of a population's ability to survive and reproduce in a specific environment.

This ability is called evolutionary fitness.

Evolutionary fitness can also be viewed as a measure of the organism's ability to anticipate changes in its environment.

The fitness, or the quantitative measure of the ability to predict environmental changes and respond adequately, can be considered as the quality that is optimised in natural life

Evolutionary Computation

Evolutionary Computation stimulates evolution on a computer.

The result of such simulations is a sense of optimisation algorithms

Optimisation iteratively improves the quality of solutions until an optimal, or near-optimal, solution is found

The evolutionary approach is based on computational models of natural selection and genetics.

We call them evolutionary computation ,an umbrella term that combines genetic algorithms, evolution strategies and genetic programming

GA - Simulation of Natural Evolution

All methods of evolutionary computation simulate natural evolution by creating a population of individuals, evaluating their fitness, generating a new population through genetic operations, and repeating this process a number of times.

We focus on Genetic Algorithm as most of the other algorithms can be viewed as variations of genetic algorithms.

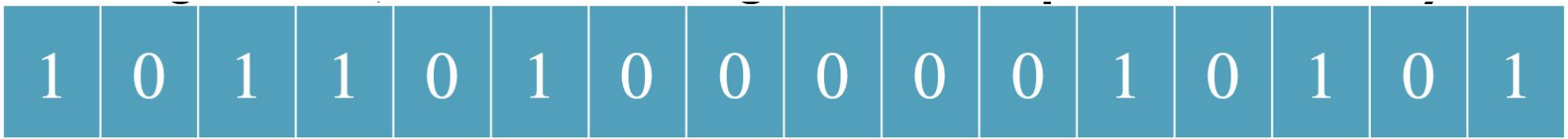
Genetic Algorithm

In early 1970s John Holland introduced the concept of genetic algorithm

His aim was to make computers do what nature does.

Holland was concerned with algorithms that manipulate strings of binary digits

Each artificial “chromosomes” consists of a number of “genes”, and each gene is represented by 0 or 1

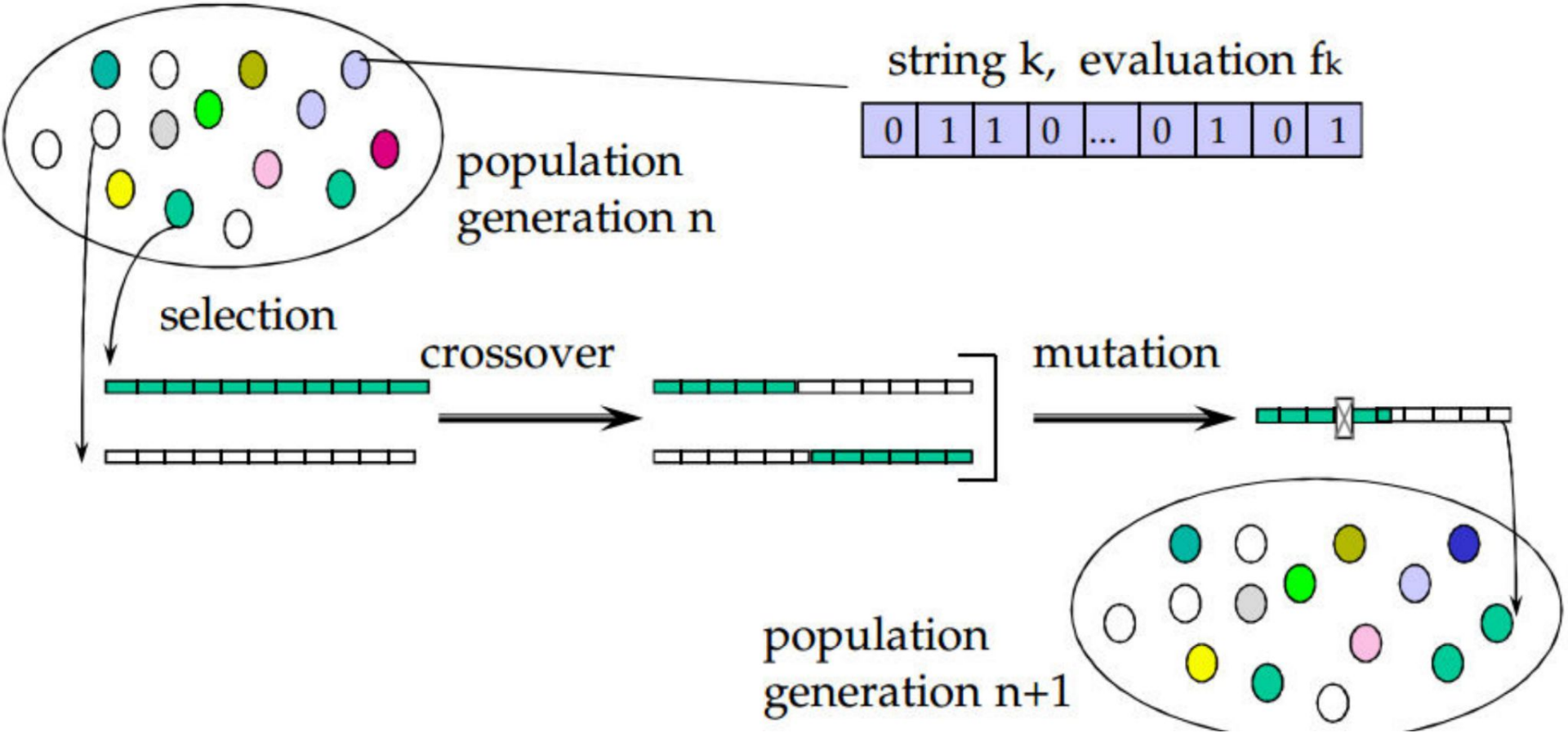


Genetic Algorithm

Two mechanisms link a GA to the problem it is solving : Encoding and Evaluation
The GA uses a measure of fitness of individual chromosomes to carry out reproduction.

As reproduction takes place, the crossover operator exchanges part of two single chromosomes and he mutation operator changes the gene value in some randomly chosen location of the chromosome

Basic Genetic Algorithm



Genetic algorithm is majorly used for 2 purposes-

1. Search

2. Optimisation

Genetic algorithms use an iterative process to arrive at the best solution.

Finding the best solution out of multiple best solutions (best of best).

Compared with Natural selection, it is natural for the fittest to survive in comparison with others.

- Evolution usually starts from a population of randomly generated individuals in the form of iteration. (Iteration will lead to a new generation).
- In every iteration or generation, the fitness of each individual is determined to select the fittest.
- Genome fittest individuals selected are mutated or altered to form a new generation, and the process continues until the best solution has reached.

The process terminates under 2 scenarios-

1. When maximum number of generations have been created
2. Fitness level reached is sufficient.

Relating it to the Optimisation scenario, we need to identify the **Genetic Representation** of our solution domain or business problem we need to solve. Evaluation criteria i.e., **Fitness Function** to decide the worth of a solution.

For example:

- We need to maximize the profit (Fitness Function) by increasing sales (Genetic representation) of the product.
- We need to find the best model hyperparameters (Fitness function) for the classification algorithms i.e., Fine-tuning to yield the best prediction
- Optimum number of feature (fitness function) selection for building the machine learning model (Genetic representation).

The process can be broadly divided as following:

1. Initialisation:

Randomly generate a population with multiple chromosomes.

Gene is the smallest unit and can be referred to as a set of characteristics (variables).

We aim to join the Genes to obtain the Chromosomes(solution).

The chromosome itself represents one candidate solution abstractly.

The generation of Chromosome is user-defined (combination of numbers between 0 and 5 or only binary numbers).

2. Defining the fit function:

Now we need to define the evaluation criteria for best chromosomes(solution). Each chromosome is assigned with a fitness score by the fitness function, which represents the goodness of the solution.

Let’s say the fitness function is the sum of all the genes.

Hence, the chromosome with the maximum sum is the fittest.

In our case, the chromosome has a sum of 12.

3. Selection:

Selecting the top 2 fittest chromosomes for creating the next generation. These will act as parents to generate offspring for the next generation which will naturally inherit the strong features.

Two pairs of individuals (**parents**) are selected based on their fitness scores.

Other chromosomes are dropped. Here are some of the methods of parent selection-

1. Roulette Wheel Selection
2. Rank Selection
3. Steady State Selection
4. Tournament Selection
5. Elitism Selection

4. Crossover:

Crossover is the equivalent of two parents having a child. Each chromosome contributes a certain number of genes to the new individual.

Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.

1. Single point crossover.
2. k-point crossover ($k \geq 1$)
3. Uniform crossover.

5. Mutation:

To avoid the duplicity(crossover generates offspring similar to parents) and to enhance the diversity in offspring we perform mutation. The mutation operator solves this problem by changing the value of some features in the offspring at random.

These steps are repeated until the termination criteria is met.

When to apply Genetic Algorithm:

- There are multiple local optima
- The objective function is not smooth (so derivative methods cannot be applied)
- Number of parameters is very large
- Objective function is noisy or stochastic

Advantages of Genetic Algorithm:

- Concept is easy to understand
- Modular, separate from application
- Answer gets better with time
- Inherently parallel; easily distributed
- Genetic algorithms work on the Chromosome, which is an encoded version of potential solutions' parameters, rather the parameters themselves.
- Genetic algorithms use fitness score, which is obtained from objective functions, without other derivative or auxiliary information

Disadvantages:

- Genetic Algorithms might be costly in computational terms since the evaluation of each individual requires the training of a model.
- These algorithms can take a long time to converge since they have a stochastic nature.