h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

5. Machine Learning

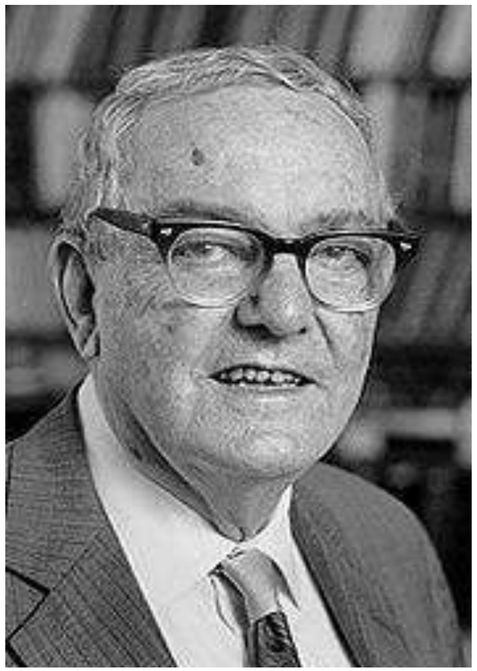
Artificial Intelligence and Neural Network

Dr. Udaya Raj Dhungana
Assist. Professor
Pokhara University, Nepal
Guest Faculty
Hochschule Darmstadt University of Applied Sciences, Germany
E-mail: udaya@pu.edu.np and udayas.epost@gmail.com

Overview

- What is machine learning?
- Applications of machine learning
- Learning Agent
- Learning Agent Components
- Machine Learning Approaches
- Forms of Learning
- Learning Algorithms
- Learning Models

What is Machine Learning?



Herbert Simon
Turing Award 1975
Nobel Prize in Economics 1978

- **Herbert Alexander Simon**:
  - "Learning is any process by which a system improves performance from experience."
  - "Machine Learning is concerned with computer programs that automatically improve their performance through experience."

What is Machine Learning?

- **Marvin Minsky, 1986**

Learning is making useful changes in the workings of our minds.

- **Mitchell, 1997**

A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**.

- Learning = Improving with experience at some task
  - –Improve over task *T*,
  - –With respect to performance measure, *P*
  - –Based on experience, *E*.

What is Machine Learning?

- Learning denotes changes in a system that enable the system to do the same task more efficiently next time.
- Learning is an important feature of "Intelligence".

Why Machine Learning?

- Develop systems that can automatically adapt and customize themselves to individual users.
  - Personalized news or mail filter
- Discover new knowledge from large databases (***data mining***).
  - Market basket analysis (e.g. diapers and beer)
- Ability to mimic human and replace certain monotonous tasks which require some intelligence.
  - like recognizing handwritten characters
- Develop systems that are too difficult/expensive to construct manually because they require specific detailed skills or knowledge tuned to a specific task (knowledge engineering bottleneck).

Why now?

- Huge amount of available data (especially Internet)
- Increasing computational power
- Growing progress in available algorithms and theory developed by researchers
- Increasing support from industries

Machine Learning Applications

Learning Agent

A **learning agent** is a tool in AI that is capable of learning from its experiences.

It starts with some basic knowledge and is then able to act and adapt autonomously, through learning, to improve its own performance.

Unlike intelligent agents that act on information provided by a programmer, learning agents are able to perform tasks, analyze performance, and look for new ways to improve on those tasks - all on their own.

Learning Agent

Example:

Imagine you've decided to start driving for Uber. It's a Friday night and you're running your typical route from the nightlife in San Francisco to many of the hotels away from the downtown area. You've become accustomed to taking the Cupertino Street via Apple headquarters, exit to pick up your Engineers, but tonight you opt for Mountain View Road via Google headquarters for a change of pace. To your surprise, you discover that Mountain View Road is not only quicker, but you can avoid the accidents prone intersection at Cupertino Street and other Bay Area street . You decide that, in the future, you'll be sure to take Mountain View Road.

Learning Agent Components

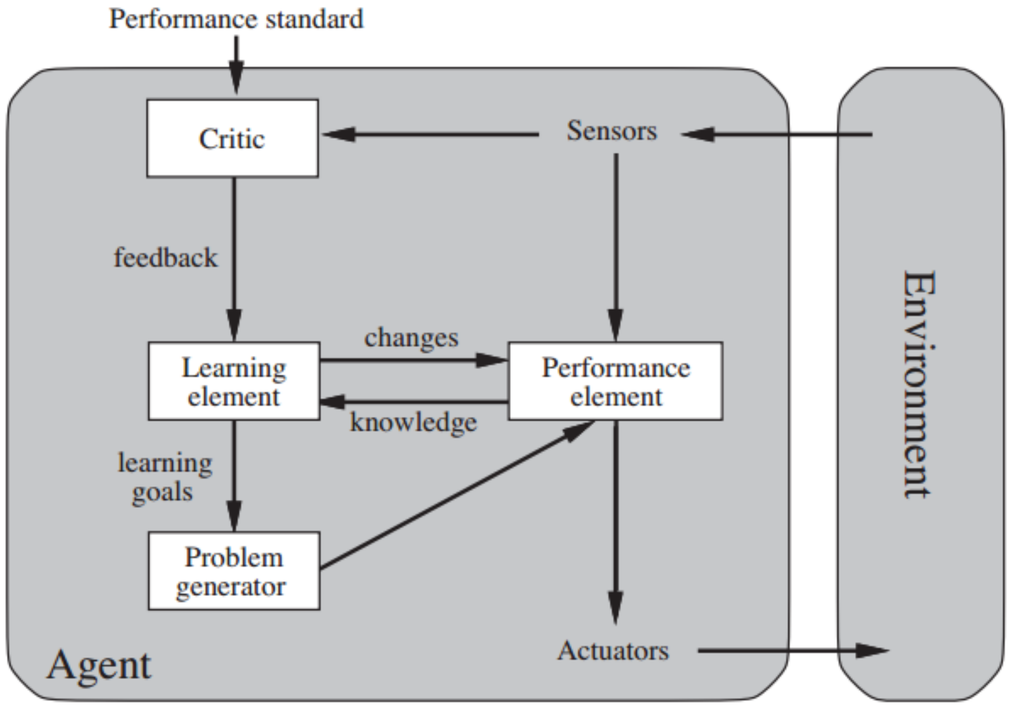
A learning agent is made up of four basic components:

1. The **performance element**

- chooses what action to take; it later shifts to a new action based on feedback and suggestions for improvement

2. The **critic element**

- determines the outcome of the action and gives feedback (**Monitors results of performance, provides feedback to learning element**)



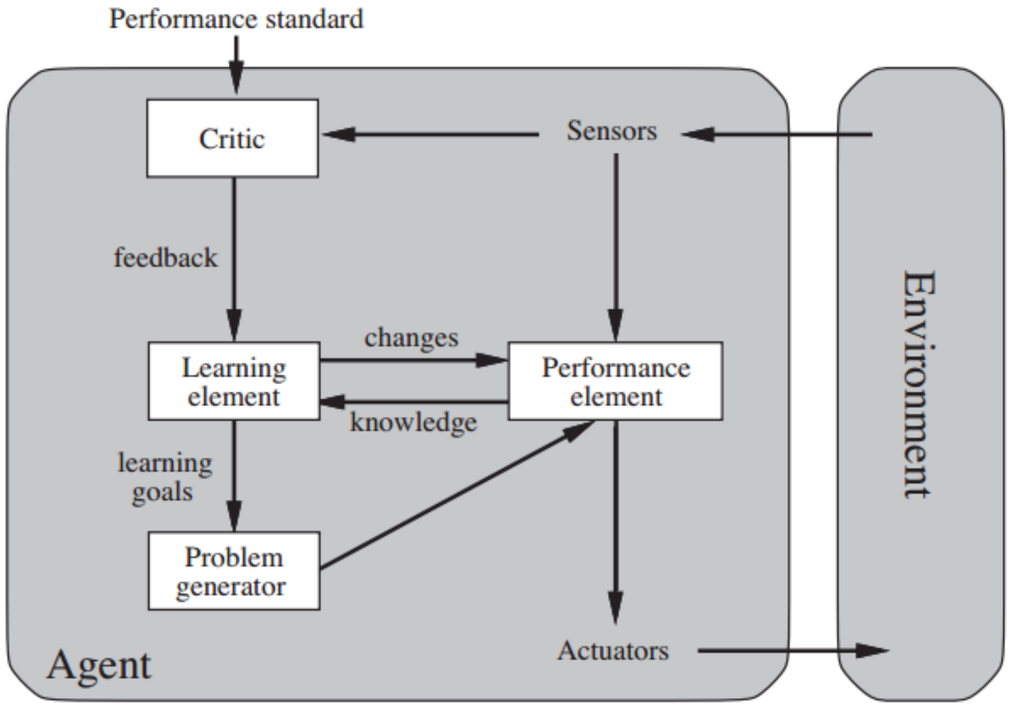
Learning Agent Components

3. The **learning element**

- takes the feedback from the critic element and figures out how to make the action better next time (**Adds knowledge, makes an improvement to the system**)

4. The **problem generator**

- is tasked with developing new experiences for the learning agent to try; this is the piece that helps the agent continue to learn



Machine Learning Approaches

Depending on the nature of the "signal" or "feedback" available to the learning system:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Machine Learning Approaches

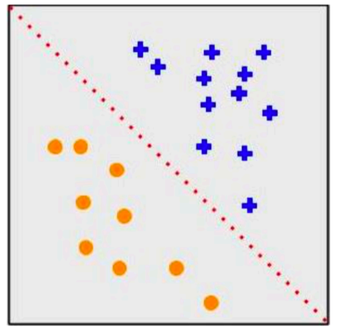
- Supervised Learning algorithms
  - build a mathematical model of a set of data (training data) that contains both:
    - example inputs and their desired outputs
  - the goal is to learn a general rule that maps inputs to outputs.
  - Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs
  - Examples:
    - Classification and regression algorithms

Machine Learning Approaches

- Supervised Learning algorithms
  - Learn through examples of which we know the desired output (what we want to predict).
  - *Is this a cat or a dog?*
  - *Are these emails spam or not?*
  - *Predict the market value of houses, given the square meters, number of rooms, neighborhood, etc.*

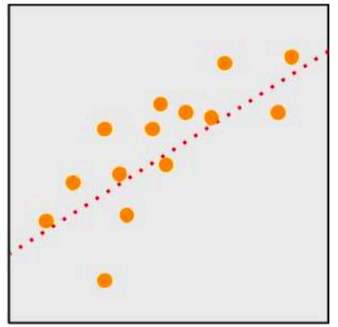
Classification

Output is a **discrete** variable (e.g., cat/dog)



Regression

Output is **continuous** (e.g., price, temperature)



Machine Learning Approaches

- Unsupervised Learning
  - No labels are given to the learning algorithm, leaving it on its own to find structure in its input.
  - Unsupervised learning can be a goal in itself
  - discover hidden patterns in data.
  - Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points.
  - learn from test data that has not been labeled, classified or categorized.
  - identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data.
  - Example:
    - Clustering

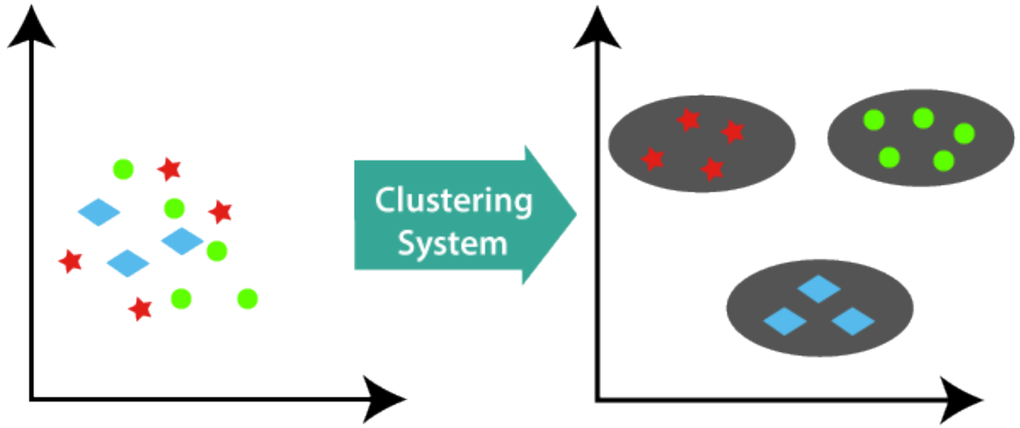
Machine Learning Approaches

- Unsupervised Learning

There is no *desired* output. Learn something about the data. *Latent* relationships.

I have photos and want to put them in 20 groups.

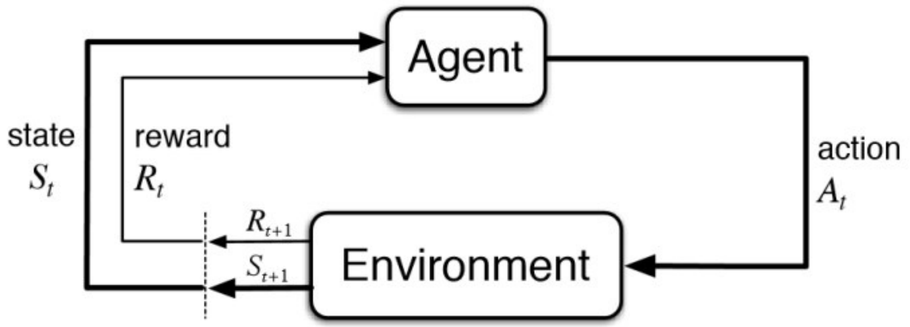
I want to find anomalies in the credit card usage patterns of my customers.



Useful for learning structure in the data (clustering), hidden correlations, reduce dimensionality, etc.

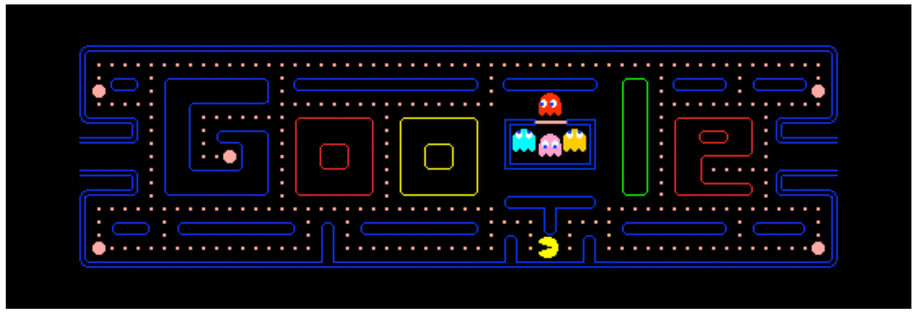
Machine Learning Approaches

- Reinforcement Learning
  - A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent).
  - As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.
  -



Machine Learning Approaches

- Reinforcement Learning
  - An agent interacts with an environment and watches the result of the interaction.
  - Environment gives feedback via a positive or negative reward signal.
  - Example:
    - Let’s take the game of PacMan where the goal of the agent (PacMan) is to eat the food in the grid while avoiding the ghosts on its way. The grid world is the interactive environment for the agent. PacMan receives a reward for eating food and punishment if it gets killed by the ghost (loses the game). The states are the location of PacMan in the grid world and the total cumulative reward is PacMan winning the game.



Forms of Learning

- Rote Learning
- Learning from examples
- Explanation-based learning
- Learning by analogy
- Learning by taking advice

Rote Learning

- Rote Learning is basically *memorisation*.
  - Saving knowledge so it can be used again.
  - Retrieval is the only problem.
  - No repeated computation, inference or query is necessary.
- A simple example of rote learning is *caching*
  - Store computed values (or large piece of data)
  - Recall this information when required by computation.
  - Significant time savings can be achieved.
  - Many AI programs (as well as more general ones) have used caching very effectively.

Rote Learning

- Samuel's Checkers program employed rote learning:
  - A minimax search was used to explore the game tree.
  - Time constraints do not permit complete searches.
  - It *records* board positions and scores at search ends.
  - Now if the same board position arises later in the game the stored value can be recalled and the end effect is that more deeper searched have occurred.

Rote Learning

- Rote learning is basically a simple process. However it does illustrate some issues that are relevant to more complex learning issues.
  1. Organisation
  2. Generalisation
  3. Stability of the Environment

Rote Learning

- Organisation:
  - access of the stored value must be faster than it would be to recompute it. Methods such as hashing, indexing and sorting can be employed to enable this.
  - *E.g* Samuel's program indexed board positions by noting the number of pieces.

Rote Learning

- Generalisation:
  - The number of potentially stored objects can be very large. We may need to generalise some information to make the problem manageable.

  - *E.g* Samuel's program stored game positions only for white to move. Also rotations along diagonals are combined.

Rote Learning

- Stability of the Environment:
  - Rote learning is not very effective in a rapidly changing environment.
  - If the environment does change then we must detect and record exactly what has changed -- *the frame problem*.

Rote Learning

- Learning by memorization;
  - which avoids understanding the inner complexities the subject that is being learned;

  - Rote learning instead focuses on memorizing the material so that it can be recalled by the learner exactly the way it was read or heard

- One-to-one mapping from inputs to stored representation;
  - Store computed value when computation is more complex so that no need to recompute. This save time and increase performance.

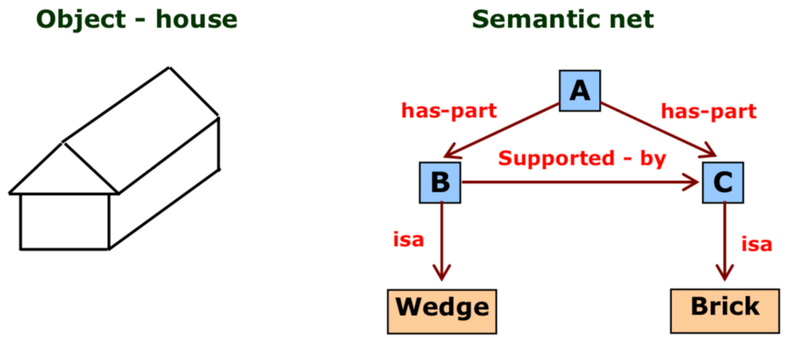
Learning from examples- Induction

- A process of learning by example.
- The system tries to induce a general rule from a set of observed instances. The learning methods extract rules and patterns out of massive data sets.
- This learning processes belong to supervised learning, does classification and constructs class definitions, called induction or concept learning.
- The techniques used for constructing class definitions (or concept leaning) are :
  - Winston's Learning program
  - Version Spaces
  - Decision Trees

Winston Learning

- Winston (1975) described a Blocks World Learning program.
- This program operated in a simple blocks domain.
- The goal is to construct representation of the definition of concepts in the blocks domain.

Concept of a house



- Node **A** represents entire structure, which is composed of two parts : node **B**, a Wedge, and node **C**, a Brick.
- Links in network include supported-by**,** has-part, and isa**.**

https://wiki.eecs.yorku.ca/course_archive/2014-15/W/6339/_media/winston.pdf

Winston Learning

Winston's program:

Winston's program followed 3 basic steps in concept formulation:

- Select one known instance of the concept. Call this the concept definition.
- Examine definitions of other known instance of the concept. Generalize the definition to include them.
- Examine descriptions of near misses. Restrict the definition to exclude these.

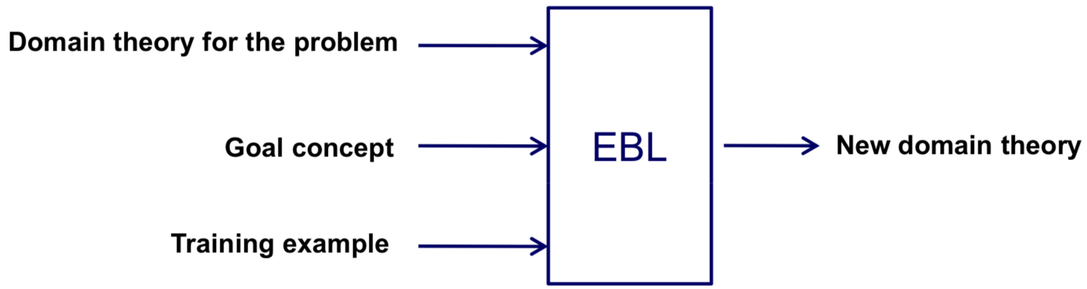
Both steps 2 and 3 of this procedure rely heavily on comparison process by which similarities and differences between structures can be detected.

Winston's program can be similarly applied to learn other concepts such as "ARCH".

https://proftriumph.com/winstons-learning/

Explanation-based Learning

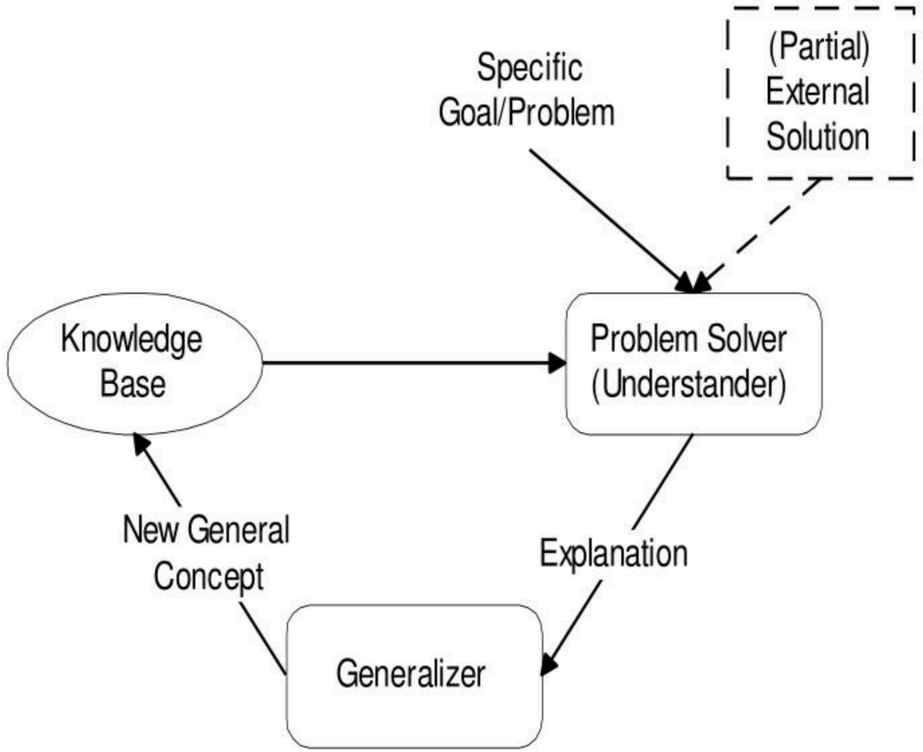
- Human learn quite a lot from one example.
  - by examining particular situations and relating them to the background knowledge in the form of known general principles.
  - This kind of learning is called "Explanation Based Learning (EBL)"
- EBL is abstracting a general concept from a particular training example.
- EBL is a technique to formulate general concepts on the basis of a specific training example.
- EBL analyses the specific training example in terms of domain knowledge and the goal concept.
- The result of EBL is an explanation structure, that explains why the training example is an instance of the goal concept.
- The explanation-structure is then used as the basis for formulating the general concept.



Explanation-based Learning Architecture

They are the 4 different kinds of input that EBL algorithm accepts.

1. **Training example ()** - Situation description, facts .
2. **Goal concept** - a high level description of what the program is supposed to learn .
3. **Domain theory/knowledge** - (Inference rules) a set of rules that describe relationships between objects and actions in a domain
4. **Operational criterion** - description of which concepts are usable.

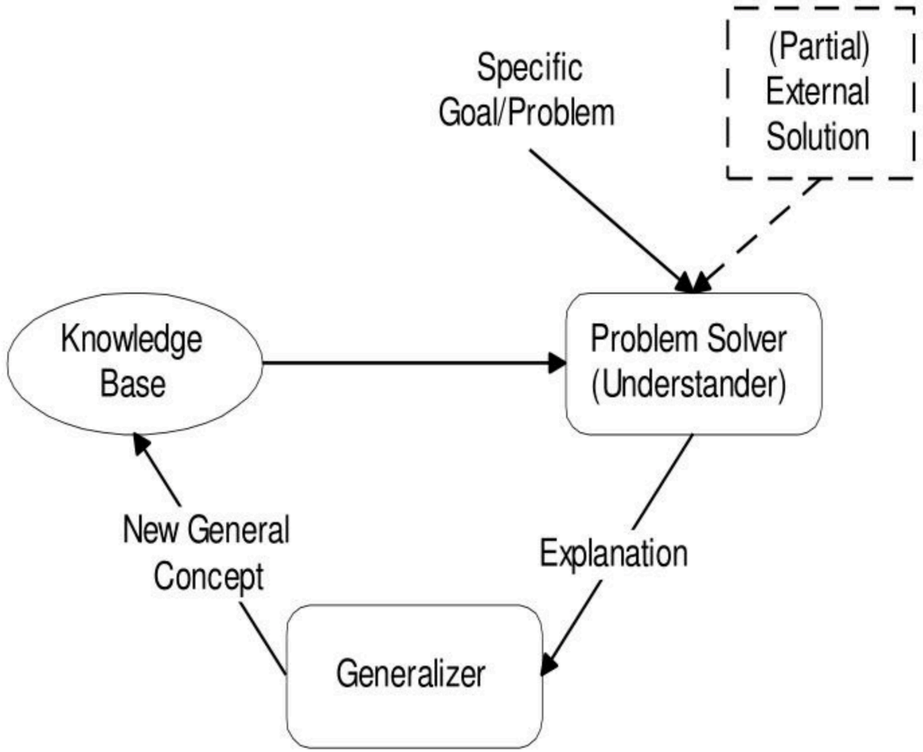


Explanation-based Learning Architecture

From the 4 different kinds of inputs, EBL algorithm computes a generalisation of the training example that is sufficient not only to describe the goal concept but also satisfies the operational criterion in the following two steps:

1. **Explanation**- the domain theory is used to prune away all unimportant aspects of the training example with respect to the goal concept.

2. **Generalization**- the explanation is generalised as far possible while still describing the goal concept.



Explanation-based Learning Architecture

Example : "CUP" Generalization problem

Goal Concept

lift-able(x) ^ stable(x) ^ open-vessel(x) ↔ cup(x)

Training example

colour(Obj23, Blue) ^ has-part(Obj23, Handle16) ^ has-part(Obj23, Bottom19) ^
owner(Obj23, Ralph) ^ has-part(Obj23, Concavity12) ^ is(Obj23, Light) ^
is(Ralph, Male) ^ isa(Handle16,Handle) ^ isa(Bottom19, Bottom) ^
is(Bottom19, Flat) ^ isa(Concavity12, Concavity) ^
is(Concavity12, Upward-Pointing)

Domain theory

has-part(x,y) ^ isa(y,Concavity) ^ is(y, Upward-Pointing) → open-vessel(x)
is(x, Light) ^ has-part(x,y) ^ isa(y,Handle) → liftable(x)
has-part(x,y) ^ isa(y, Bottom) ^ is(y,Flat) → stable(x)

Operationality Criterion

The concept definition is expressed in terms of structural features.
The first stage of the EBL process is to show why the training example is an example of a cup. This represents a proof.

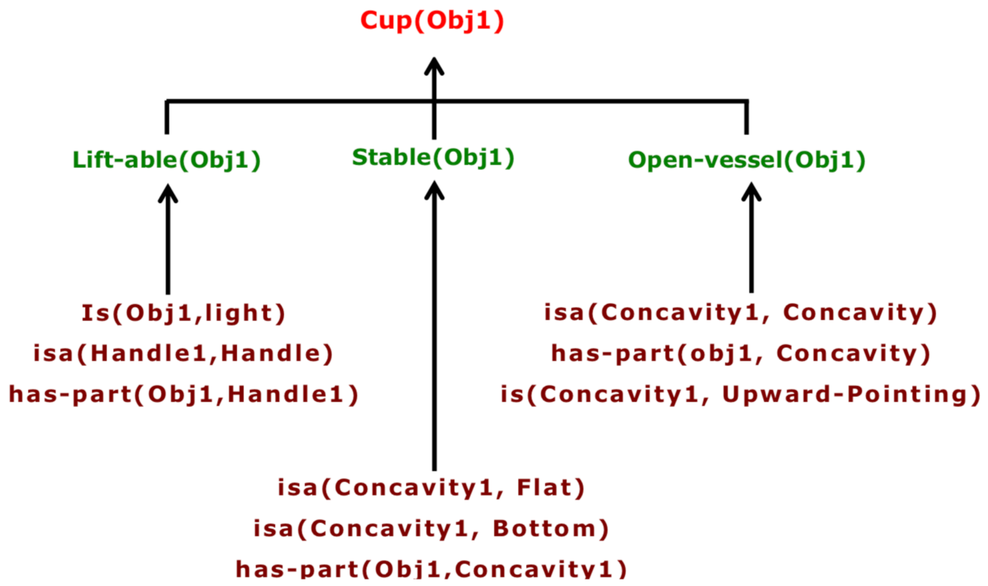
This proof is expressed only in terms of the operationality criterion and irrelevant details discarded relating to the Owner and Color.

Explanation structure of the cup.

Explanation-based Learning Architecture

Explanation Structure of the cup

The structure below represents the first stage of the EBL process. It proofs why the training example is an example of a cup.



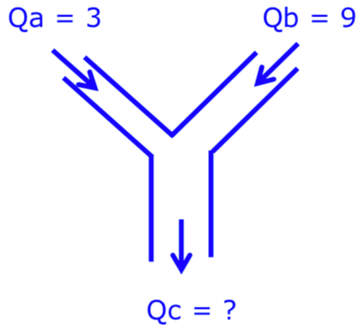
Proof tree generalization using a goal regression technique.

The above proof is now generalized by using a goal regression technique. In this example replacing the constants with variables gives the required generalization of the cup.

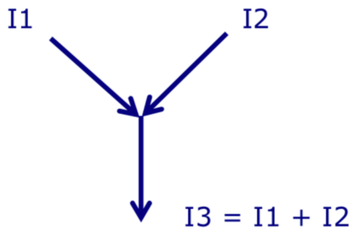
has-part(x, y) U isa(y, Concavity) U is(y, Upward-Pointing) U
has-part(x, z) U isa(z, Bottom) U is(z, Flat) U has-part(x,w) U
isa(w, Handle) U is(x, Light)

Learning by Analogy

- acquiring new knowledge about an input entity by transferring it from a known similar entity.
- transforms the solutions of problems in one domain to the solutions of the problems in another domain by discovering analogous states and operators in the two domains.
- Example: Infer by analogy the hydraulics laws that are similar to Kirchoff's laws.



Hydraulic Problem



Kirchoff's First Law

Learning by Analogy

- Analogy involves a complicated mapping between what might appear to be two dissimilar concepts.
- There are two methods of analogical problem methods studied in AI:
    - Transformational Analogy
    - Derivational Analogy

Learning by Analogy

- Transformational Analogy: Look for a similar solution and *copy* it to the new situation making suitable substitutions where appropriate.
- Example: Geometry:- If you know about lengths of line segments and a proof that certain lines are equal then we can make similar assertions about angles.

We know that lines RO = NY and angles AOB = COD
We have seen that RO + ON = ON + NY - additive rule.
So we can say that angles AOB + BOC = BOC + COD
So by a transitive rule line RN = OY
So similarly angle AOC = BOD

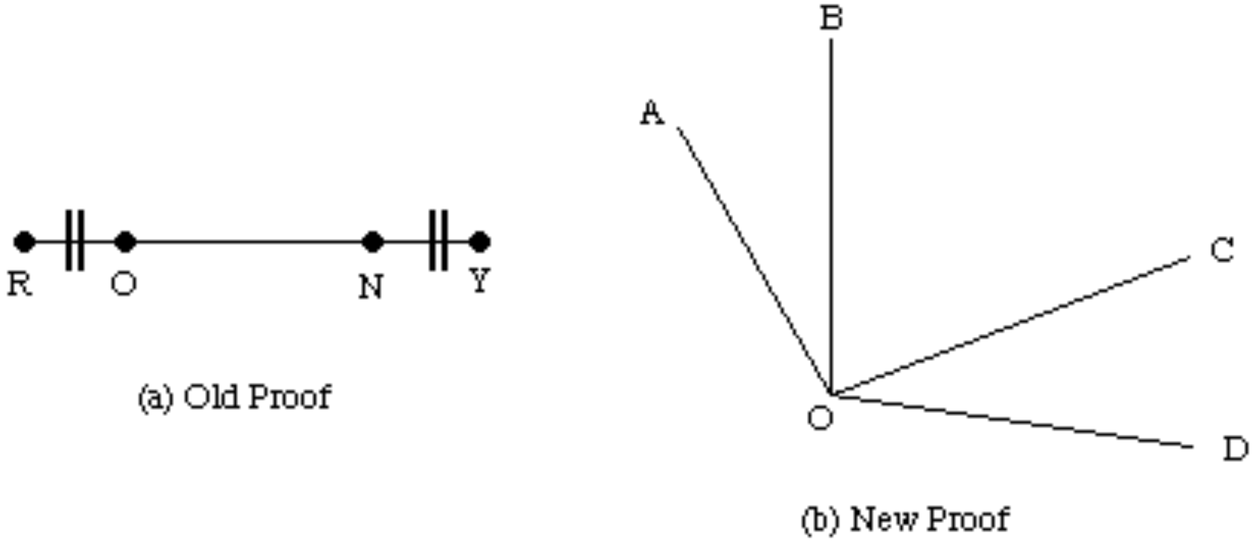


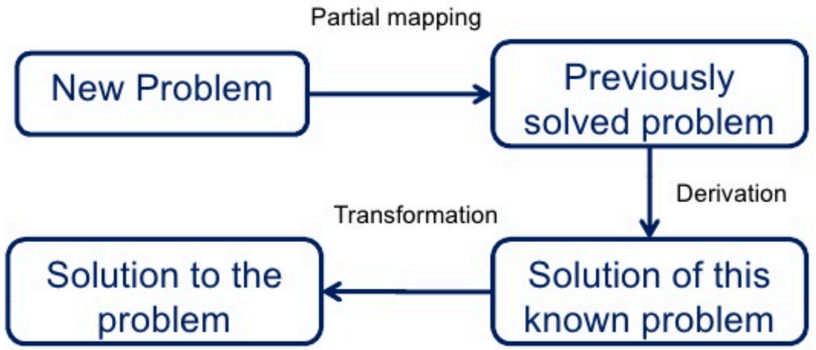
Fig. Transformational Analogy Example

Learning by Analogy

- Derivational Analogy:
  - Transformational analogy does not look at how the problem was solved -- it only looks at the final solution.
  - The *history* of the problem solution - the steps involved - are often relevant.
  - Carbonell (1986) showed that derivational analogy is a necessary component in the transfer of skills in complex domains:
    - In translating Pascal code to LISP -- line by line translation is no use. You will have to *reuse* the major structural and control decisions.
    - One way to do this is to *replay* a previous derivation and modify it when necessary.
    - If initial steps and assumptions are still valid copy them across.
    - Otherwise alternatives need to found -- best first search fashion.

Learning by Analogy

- Other Examples:
  - Learn how to hold a cup and then learn to hold objects with a handle

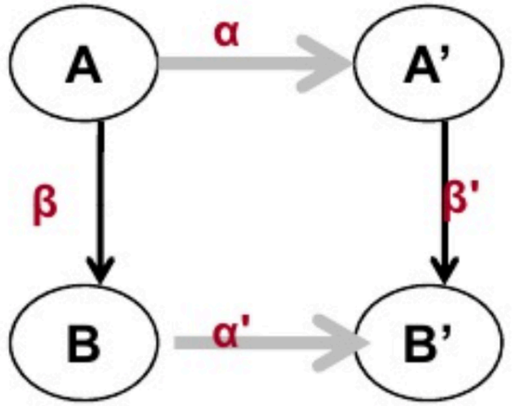


Learning by Analogy

A is similar to A' according to α

If I have B, can I get B'?

- Learn the causality relationship β
- Transform α to α'
- Get B' according to B and α'



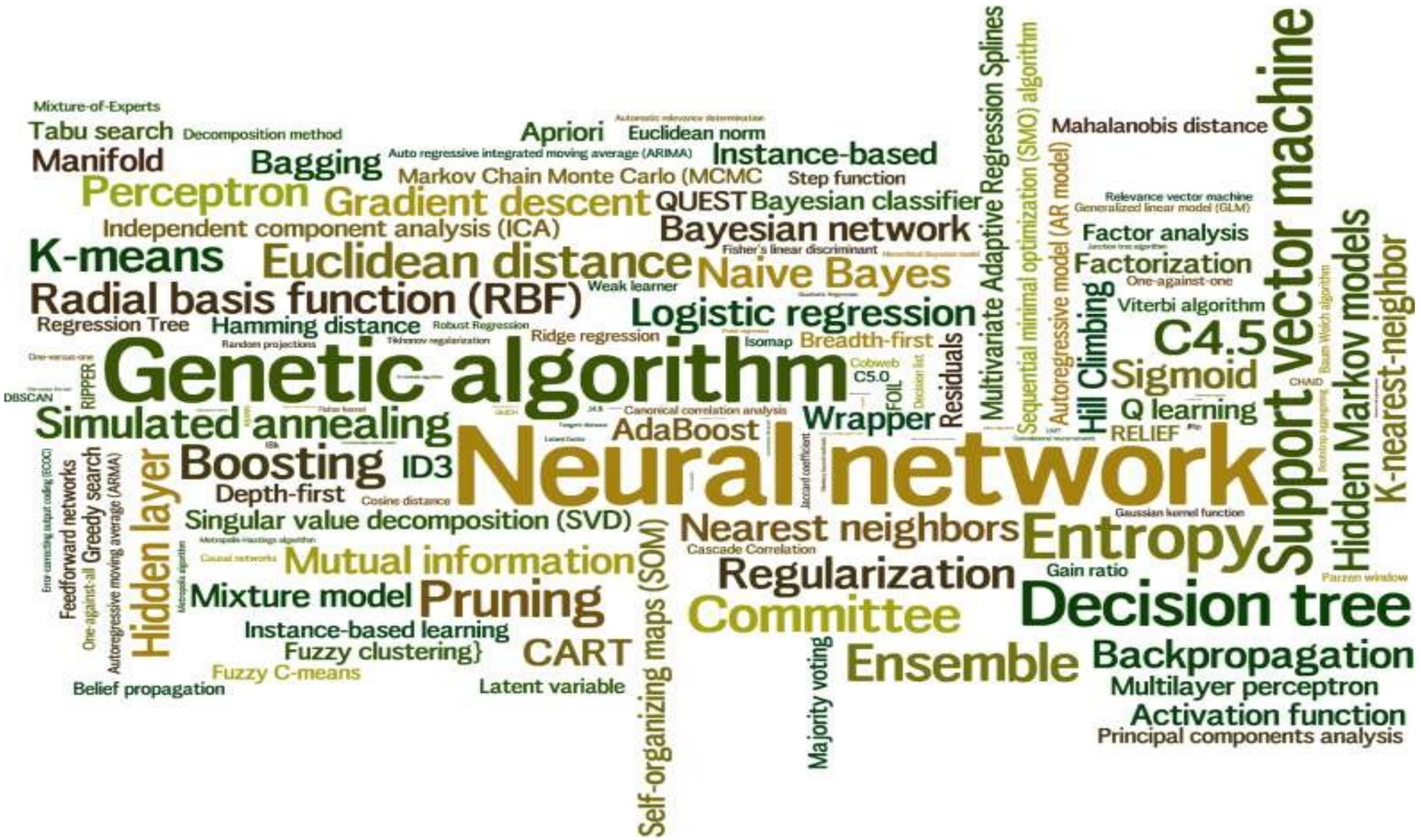
Learning by taking Advice

- Similar to rote learning but **advice** may need to be operationalized
- In this type of learning, a programmer writes a program to give some instructions to perform a task to the computer. Once it is learned (i.e. programmed), the system will be able to do new things.
- Also, there can be several sources for taking advice such as humans(experts), internet etc.
- However, this type of learning has a more necessity of inference than rote learning.
- As the stored knowledge in knowledge base gets transformed into an operational form, the reliability of the knowledge source is always taken into consideration.

Learning by taking Advice

- The idea of advice taking in AI based learning was proposed as early as 1958 (McCarthy).
- However very few attempts were made in creating such systems until the late 1970s. E.g. Expert systems
- There are two basic approaches to advice taking:
  - Take high level, abstract advice and convert it into rules that can guide performance elements of the system. *Automate all aspects of advice taking*
  - *Develop sophisticated tools* such as knowledge base editors and debugging. These are used to aid an expert to translate his expertise into detailed rules. Here the expert is an *integral* part of the learning system. Such tools are important in *expert systems* area of AI.

Learning Algorithms



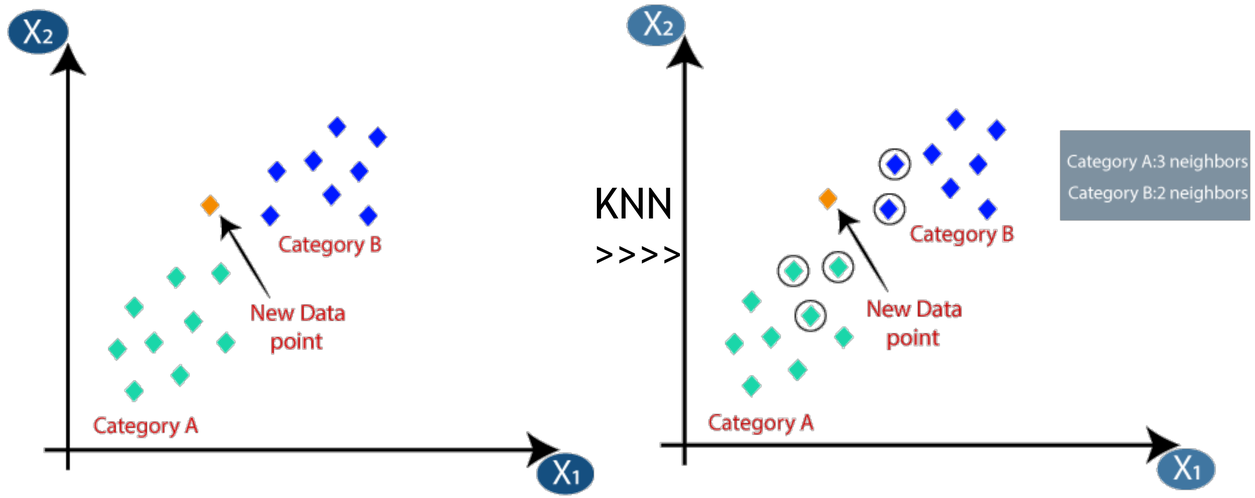
K- Nearest Neighbor Algorithm

- Supervised Learning technique.
- Assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- Mostly it is used for the Classification problems but can be used for regression as well.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

K-Nearest neighbor algorithm

- Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories.
- **Algorithm:**
  - **Step-1:** Select the number K of the neighbors
  - **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
  - **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
  - **Step-4:** Among these k neighbors, count the number of the data points in each category.
  - **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
  - **Step-6:** Our model is ready.



Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$ A(x1, y1) and B(x2, y2)

Learning Models

- **Artificial Neural Network**
- Decision Tree
- Support Vector Machine
- Regression Analysis
- Bayesian Networks
- Genetic Algorithms
- Boltzmann Machines

Genetic Algorithm

- Genetic Algorithms(GAs) are adaptive heuristic search algorithms.
- They are inspired by Charles Darwin's theory of natural evolution.
- This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.
- They are commonly used to generate high-quality solutions for optimization problems and search problems.
- This algorithm is important because it solves difficult problems that would take a long time to solve.

See an Example at https://towardsdatascience.com/genetic-algorithm-explained-step-by-step-65358abe2bf

Genetic Algorithm

- The process of natural selection starts with the selection of fittest individuals from a population.
- Each individual is represented as a string of character/integer/float/ bits. This string is analogous to the Chromosome.
- They produce offspring which inherit the characteristics of the parents and will be added to the next generation.
- If parents have better fitness, their offspring will be better than parents and have a better chance at surviving.
- This process keeps on iterating and at the end, a generation with the fittest individuals will be found.
- This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

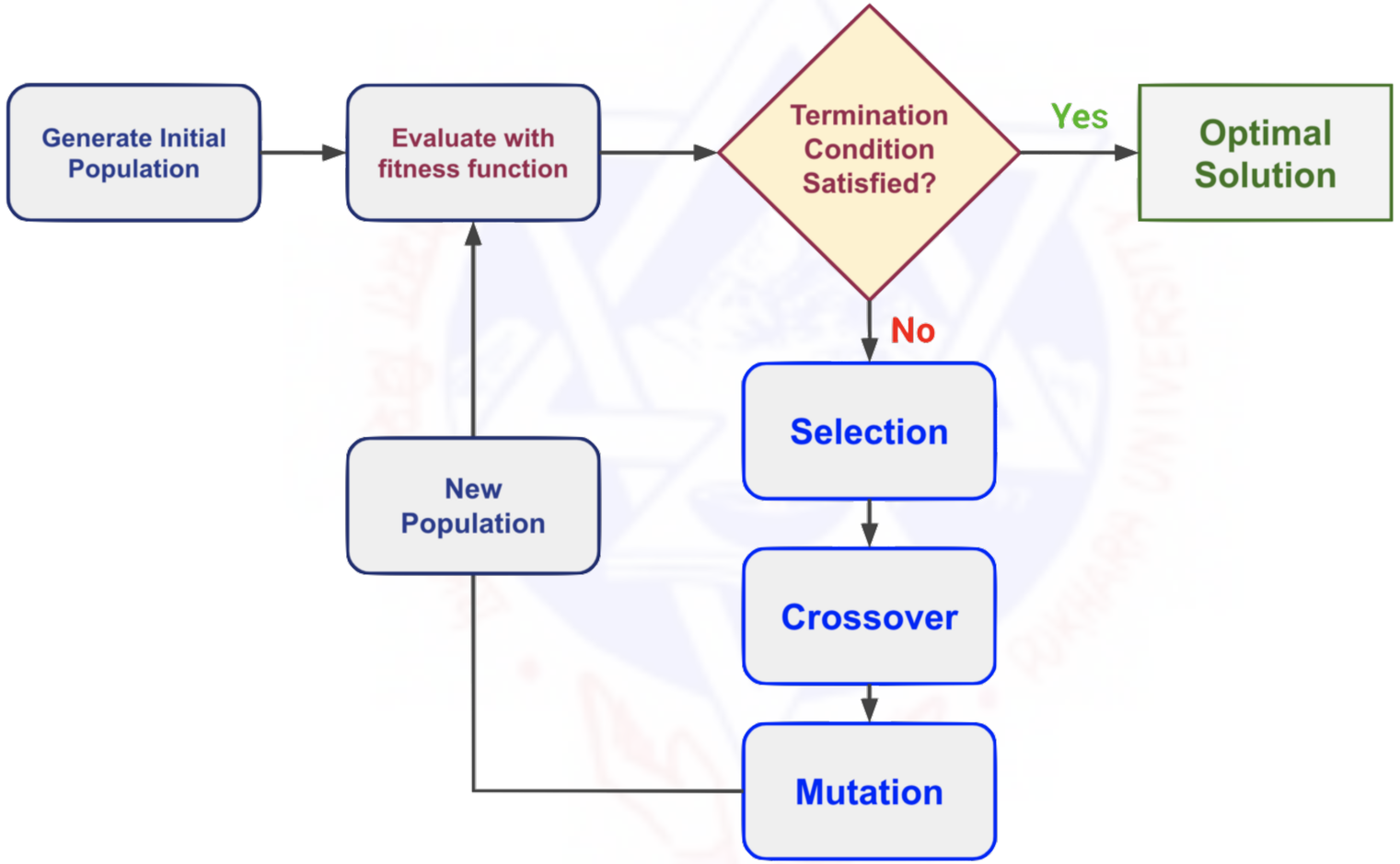
See an Example at https://towardsdatascience.com/genetic-algorithm-explained-step-by-step-65358abe2bf

Phases in Genetic Algorithm

- Genetic algorithms follow the following phases to solve complex optimization problems:
  - Initialization
  - Fitness assignment
  - Selection
  - Reproduction
    - Crossover
    - Mutation
  - Termination

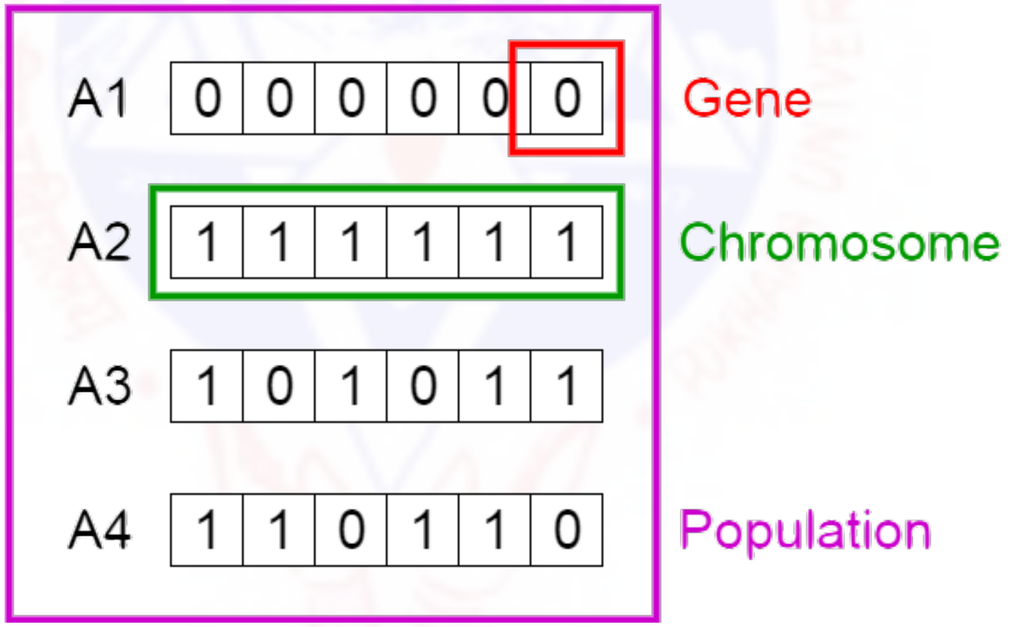
See an Example at https://towardsdatascience.com/genetic-algorithm-explained-step-by-step-65358abe2bf

Phases in Genetic Algorithm



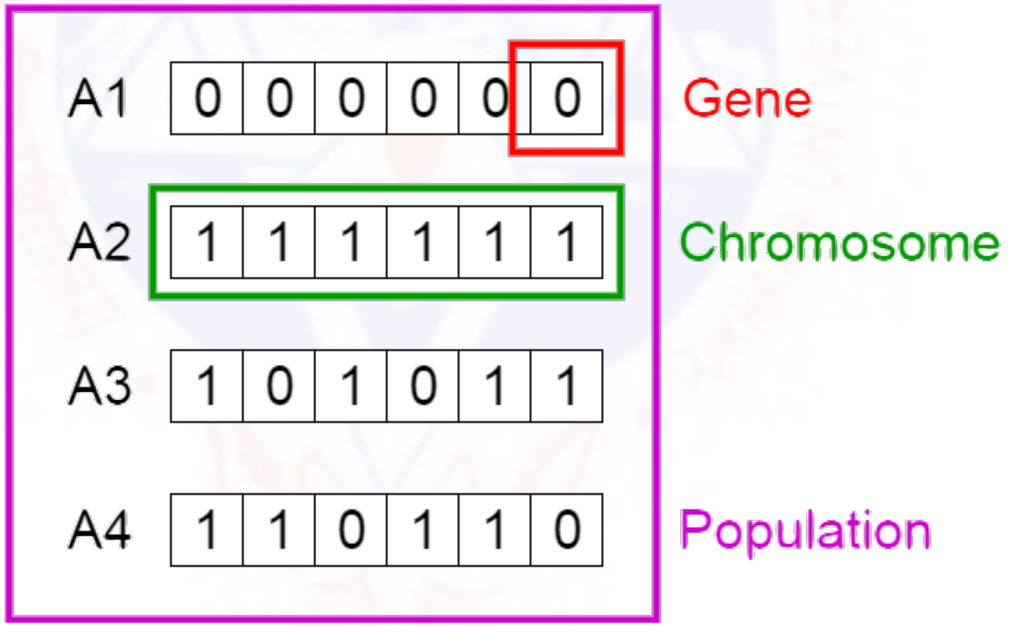
Phases in Genetic Algorithm

- Initial Population:
  - The process begins with a set of individuals which is called a **Population**. Each individual is a solution to the problem you want to solve.
  - An individual is characterized by a set of parameters (variables) known as **Genes**. Genes are joined into a string to form a **Chromosome** (solution).



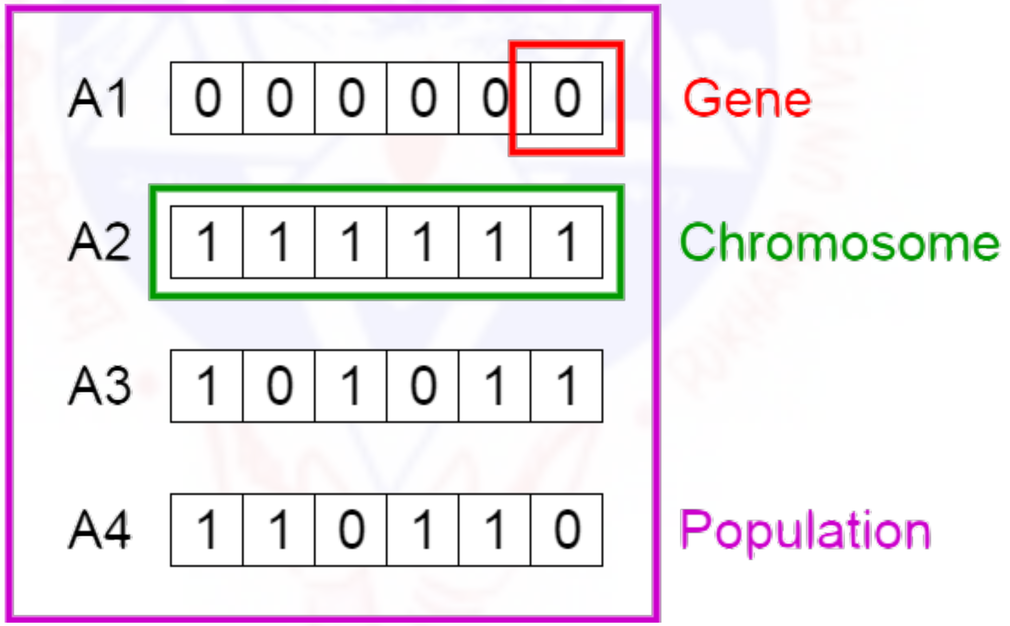
Phases in Genetic Algorithm

- Initial Population:
  - In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet.
  - Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.



Phases in Genetic Algorithm

- Initial Population:
  - This initial population consists of all the probable solutions to the given problem.
  - The most popular technique for initialization is the use of random binary strings



Phases in Genetic Algorithm

- Fitness Function:
  - We need to define the evaluation criteria for best chromosomes. This is called fitness function.
  - The **fitness function** determines how fit an individual is (the ability of an individual to compete with other individuals).
  - It gives a **fitness score** to each individual.
  - The probability that an individual will be selected for reproduction is based on its fitness score.
  - Better the fitness score, the higher the chances of being chosen for reproduction.
  - The individuals having better fitness scores are given more chance to reproduce than others.
  - The individuals with better fitness scores are selected who mate and produce **better offspring** by combining chromosomes of parents.

Phases in Genetic Algorithm

- Selection:
  - The idea of **selection** phase is to select the fittest individuals and let them pass their genes to the next generation
  - Pairs of individuals (**parents**) are selected based on their fitness scores.
  - These individuals pass on their genes to the next generation
  - The main objective of this phase is to establish the region with high chances of generating the best solution to the problem (better than the previous generation).
  - The genetic algorithm uses the fitness proportionate selection technique to ensure that useful solutions are used for recombination

Phases in Genetic Algorithm

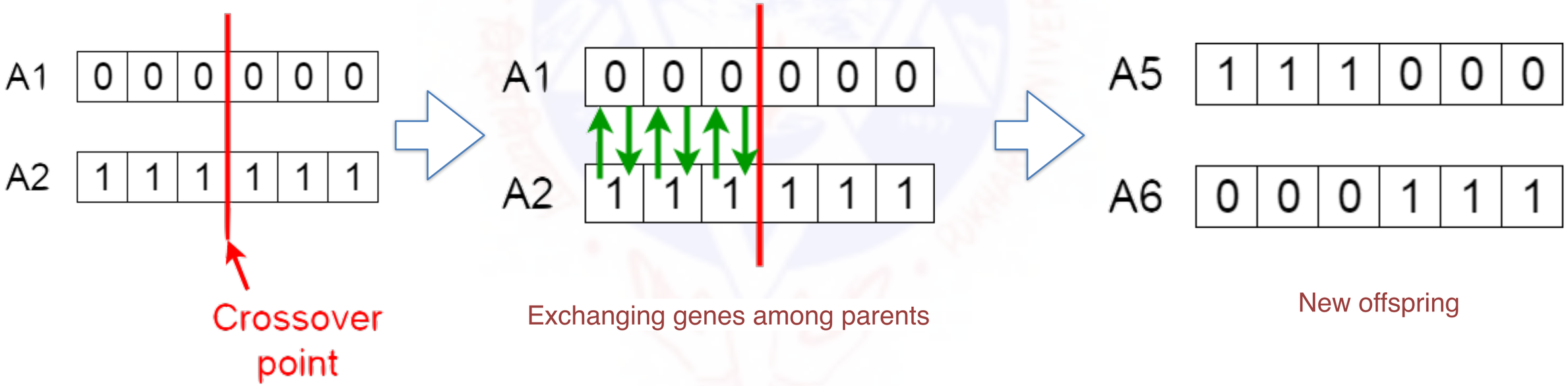
- Reproduction:
  - This phase involves the creation of a child population. The algorithm employs following operators that are applied to the parent population:
  - Crossover:
  - Mutation:

Phases in Genetic Algorithm

- Crossover:
  - **Crossover** is the most significant phase in a genetic algorithm.
  - This operator swaps the genetic information of two parents to reproduce an offspring.
  - It is performed on parent pairs that are selected randomly to generate a child population of equal size as the parent population.
  - For each pair of parents to be mated, a **crossover point** is chosen at random from within the genes.

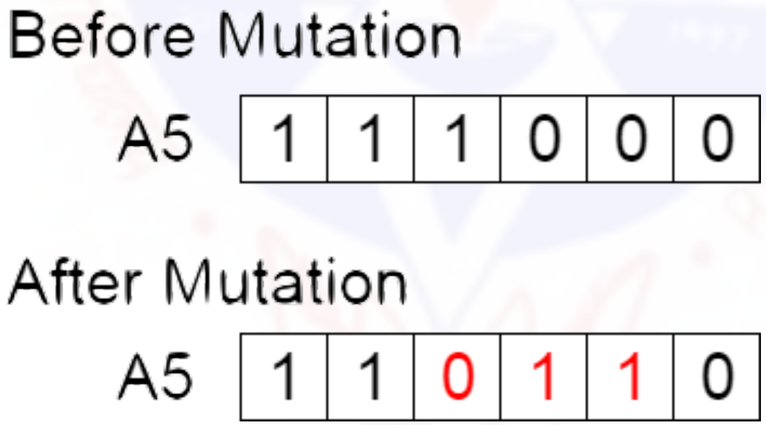
Phases in Genetic Algorithm

- Crossover:
  - For example, consider the crossover point to be 3 as shown below.
  - **Offspring** are created by exchanging the genes of parents among themselves until the crossover point is reached.
  - The new offspring are added to the population.



Phases in Genetic Algorithm

- Mutation:
  - This operator adds new genetic information to the new child population.
  - This is achieved by flipping some bits in the chromosome.
  - Mutation solves the problem of local minimum and enhances diversification.
  - In certain new offspring formed, some of their genes can be subjected to a **mutation** with a low random probability.



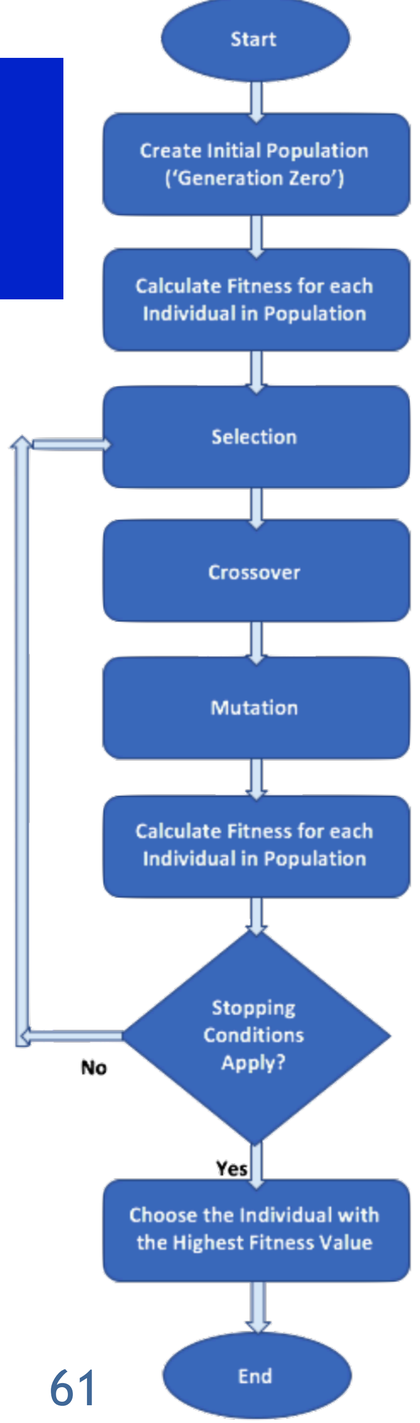
Phases in Genetic Algorithm

- Termination:
  - The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation).
  - Then it is said that the genetic algorithm has provided a set of solutions to our problem.
  - The algorithm will terminate after the threshold fitness solution has been attained.
  - It will identify this solution as the best solution in the population.

Phases in Genetic Algorithm

Genetic Algorithm:

1. Randomly initialize populations p
2. Determine fitness of population
3. Until convergence, repeat the following steps:
    A. Select parents from population
    B. Crossover and generate new population
    C. Perform mutation on new population
    D. Calculate fitness for new population



Advantages of Genetic Algorithm

- Parallelism
- Global optimization
- A larger set of solution space
- Requires less information
- Provides multiple optimal solutions
- Probabilistic in nature
- Genetic representations using chromosomes

Disadvantages of Genetic Algorithm

- They are not effective in solving simple problems.
- Lack of proper implementation may make the algorithm converge to a solution that is not optimal.
- The quality of the final solution is not guaranteed.
- Repetitive calculation of fitness values may make some problems to experience computational challenges.
- The fitness function identification is a limitation.
- 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.

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 (may not be predicted precisely)

Applications of Genetic Algorithm

- **Transport**: Genetic algorithms are used in the traveling salesman problem to develop transport plans that reduce the cost of travel and the time taken. They are also used to develop an efficient way of delivering products.
- **DNA Analysis**: They are used in DNA analysis to establish the DNA structure using spectrometric information.
- **Aircraft Design:** They are used to develop parametric aircraft designs. The parameters of the aircraft are modified and upgraded to provide better designs.
- **Economics:** They are used in economics to describe various models such as the game theory, cobweb model, asset pricing, and schedule optimization.

Applications of Genetic Algorithm

- **Optimization Problem:** One of the best examples of the optimization problems is the travel salesman problem which uses GA. Other optimization problems such as job scheduling, sound quality optimization GAs are widely used.
- **Immune system model:** GAs are used to model various aspects of the immune system for individual gene and multi-gene families during evolutionary time.
- **Machine Learning:** GAs have been used to solve problem-related to classification, prediction, create rules for learning and classification.

Questions

What is machine learning?
What are different applications of machine learning? List 10 of them.
What is a learning agent? Describe different components of learning agent.
What are different machine learning approaches? Describe in brief.
What is supervised learning? Explain.
What is unsupervised learning? Explain.
What is reinforcement learning? Explain.
What are different forms of Learning?
What is rote learning? What is the difference between rote learning and learning by taking example?
Explain about learning by examples.
Explain about explanation based learning.
List some learning algorithms.
What are different learning models?
Explain the genetic algorithms in detail.

h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

THANK YOU

End of Chapter