KNOWLEDGE ACQUISITION

Chapter 4

What is Learning? Some Quotations

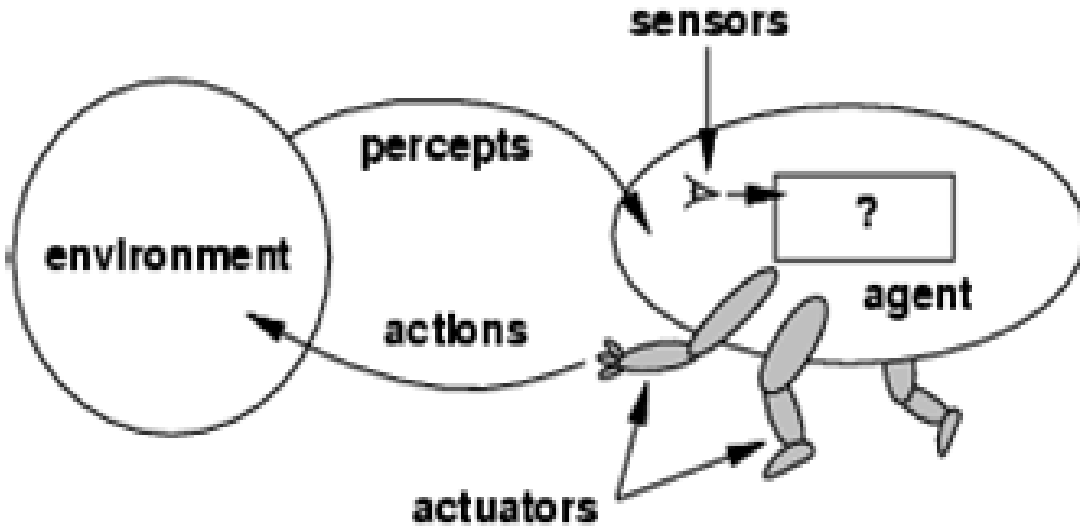
- **Herbert Simon, 1983**
  - Learning denotes changes in a system that enable a system to do the same task more efficiently the next time.
- **Marvin Minsky, 1986**
  - Learning is making useful changes in the workings of our minds.
- **Ryszard Michalski, 1986**
  - Learning is constructing or modifying representations of what is being experienced.
- **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

- Learning plays a vital role in knowledge acquisition.
- Making a machine intelligent would require making it to learn.

Learning Agents

- An **agent** is an entity that is capable of **perceiving** and doing **action**.
- An agent can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators**.
- In computer science an **agent is a software agent** that assists users and acts in performing computer-related tasks.



Learning Agents

Environment → / Agent ↓	Sensors	Actuators
Human agent	Eyes, ears, etc.	Legs, hands, mouth
Robotic agent	Cameras, IR range finders	Motors
Software agent	Key strokes, file contents	Displays to screen, writes files

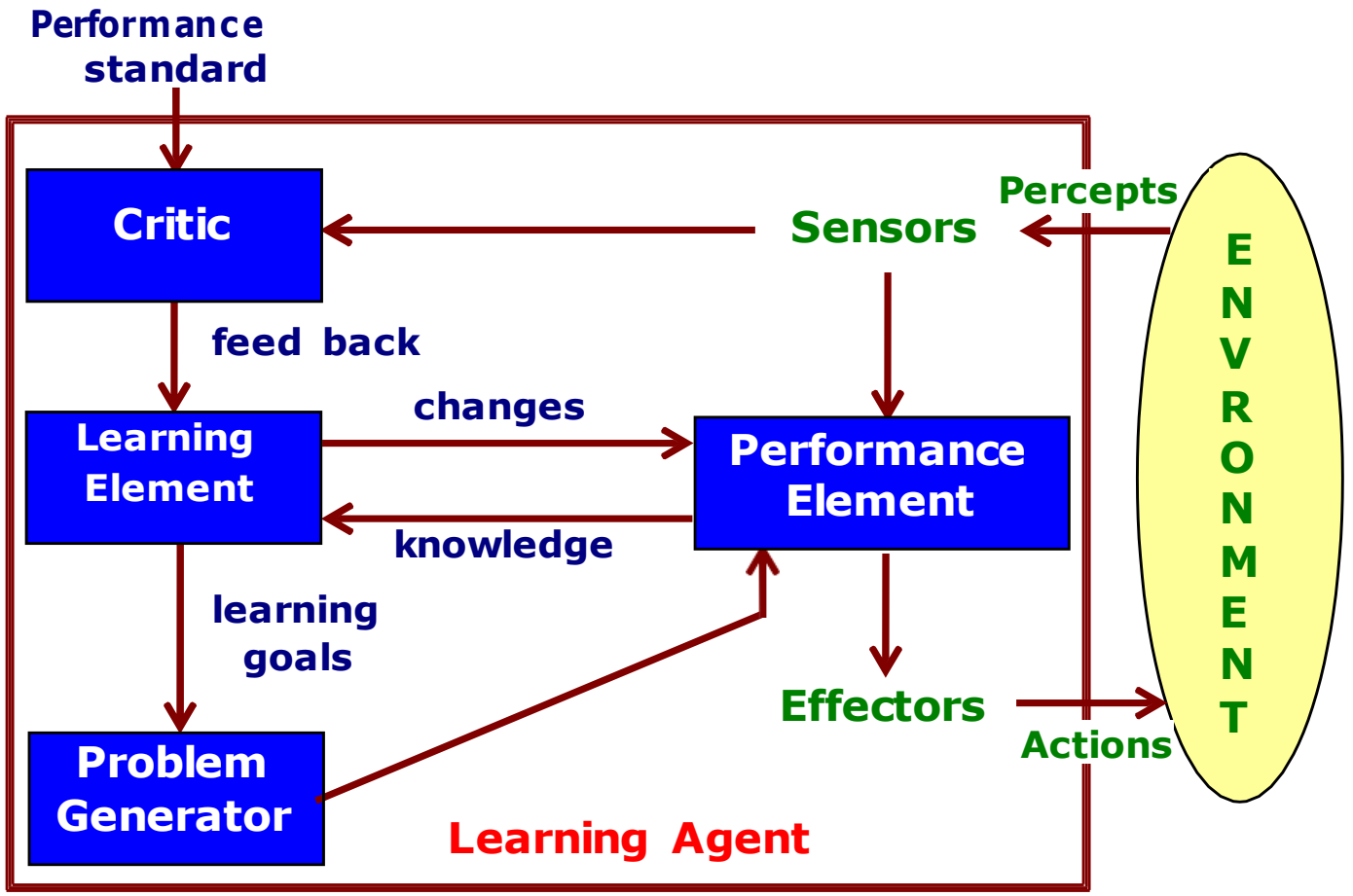
Intelligent Agent

- In artificial intelligence, the term used for agent is an intelligent agent.
- Learning is an important feature of “Intelligence”.
- Percept : agent's perceptual inputs
- Percept sequence : history of everything the agent has perceived
- Agent function : describes agent’s behavior
- Agent program : implements agent's function

Learning agent consist of four main components

- Learning element,
- Performance element,
- Critic, and
- Problem generator

Components of a Learning System



Components of a Learning System

- **Performance Element:** The Performance Element is the agent itself that acts in the world. It takes in percepts and decides on external actions.
- **Learning Element:** It responsible for making improvements, takes knowledge about performance element and some feedback, determines how to modify performance element.
- **Critic:** Tells the Learning Element how agent is doing (success or failure) by comparing with a fixed standard of performance.
- **Problem Generator:** Suggests problems or actions that will generate new examples or experiences that will aid in training the system further.

Example: Automated Taxi on City Roads

- **Performance Element:** Consists of knowledge and procedures for driving actions. E.g., turning, accelerating, braking are performance element on roads.
- **Learning Element:** Formulates goals. E.g., learn rules for braking, accelerating, learn geography of the city.
- **Critic:** Observes world and passes information to learning element. E.g. , quick right turn across three lanes of traffic, observe reaction of other drivers.
- **Problem Generator:** Try south city road.

Paradigms of Machine Learning

- **Rote Learning:** Learning by memorization;
  - one-to-one mapping from inputs to stored representation;
  - association-based storage and retrieval.
- **Induction:** Learning from examples;
  - a form of supervised learning,
  - uses specific examples to reach general conclusions;
  - concepts are learned from sets of labeled instances.
- **Clustering:** Discovering similar group;
  - unsupervised,
  - inductive learning in which natural classes are found for data instances, as well as ways of classifying them.
- **Analogy:** Determine correspondence between two different representations that come from inductive learning in which a system transfers knowledge from one database into another database of a different domain.

Paradigms of Machine Learning

- **Discovery**: Learning without the help from a teacher; Learning is both inductive and deductive. It is deductive if it proves theorems and discovers concepts about those theorems. It is inductive when it raises conjectures (guess). It is unsupervised, specific goal not given.
- **Genetic Algorithms:** Inspired by natural evolution; In the natural world, the organisms that are poorly suited for an environment die off, while those well-suited for it prosper. Genetic algorithms search the space of individuals for good candidates. The "goodness" of an individual is measured by some fitness function. Search takes place in parallel, with many individuals in each generation.
- **Reinforcement:** Learning from feedback (+ve or -ve reward) given at end of a sequence of steps. Unlike supervised learning, the reinforcement learning takes place in an environment where the agent cannot directly compare the results of its action to a desired result. Instead, it is given some reward or punishment that relates to its actions. It may win or lose a game, or be told it has made a good move or a poor one. The job of reinforcement learning is to find a successful function using these rewards.

Rote Learning

- Most elementary form of learning
- Rote learning technique avoids understanding the inner complexities but focuses on memorizing the material so that it can be recalled by the learner exactly the way it was read or heard
- **Learning by Memorization** avoids understanding the inner complexities of 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.

Example

- Samual's (1963) checkers program was designed to store the chess moves played by its creator and thus used to learn the game.
- It learned to play checkers to such a good extent that ultimately it was able to beat its own creator.

Limitations of Rote Learning

- It fails when complexities increase in the stored information and it becomes difficult to access the right information at the right time.

Learning from Example: 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.
- The learning processes belong to supervised learning, do classification and construct class definitions, called induction or concept learning.
- From the perspective of inductive learning, we are given input samples (x) and output samples (f(x)) and the problem is to estimate the function (f).
- Specifically, the problem is to generalize from the samples and the mapping to be useful to estimate the output for new samples in the future.

Example

- For example, having seen many cats, all of which have tails, one might conclude that all cats have tails.
- This is an unsound step of reasoning but it would be impossible to function without using induction to some extent.
- In many areas it is an explicit assumption.
- There is scope of error in inductive reasoning, but still it is a useful technique that has been used as the basis of several successful systems.

Concept Learning

- One major subclass of inductive learning is concept learning.
- This takes examples of a concept and tries to build a general description of the concept.
- Very often, the examples are described using attribute-value pairs.

Example

- The example of inductive learning given here is that of a fish. Look at the table below:

	herring	cat	dog	cod	Whale
Swims	Yes	No	No	Yes	Yes
Has fins	Yes	No	No	Yes	Yes
Has lungs	No	Yes	Yes	No	Yes
Is a fish	Yes	No	No	Yes	No

- In this example, there are various ways of generalizing from examples of fish and non-fish.
- The simplest description can be that a fish is something that does not have lungs.
- No other single attribute would serve to differentiate the fish.
- The two very common inductive learning algorithms are version spaces and ID3 (Iterative Dichotomiser 3 - Decision Trees).

Some Practical Examples of Induction

- **Credit risk assessment**.
  - The x is the properties of the customer.
  - The f(x) is credit approved or not.
- **Disease diagnosis**.
  - The x are the properties of the patient.
  - The f(x) is the disease they suffer from.
- **Face recognition**.
  - The x are bitmaps of peoples faces.
  - The f(x) is to assign a name to the face.
- **Automatic steering**.
  - The x are bitmap images from a camera in front of the car.
  - The f(x) is the degree the steering wheel should be turned.

Decision Trees

- A Decision Tree takes as input an object given by a set of properties, output a Boolean value (yes/no decision).
- Each internal node in the tree corresponds to test of one of the properties.
- Branches are labelled with the possible values of the test.
- **Aim:** Learn goal concept (goal predicate) from examples
- **Learning element:** Algorithm that builds up the decision tree.
- **Performance element:** decision procedure given by the tree

Example

- Problem to wait for a table at a restaurant. A decision tree decides whether to wait or not in a given situation.
- Attributes:

(a) Alternate: alternative restaurant nearby

(b) Bar: bar area to wait

(c) Fri/Sat: true on Fridays and Saturdays

(d) Hungry: whether we are hungry

(e) Patrons: how many people in restaurant (none, some, or full)

(f) price: price range (£, ££, £££)

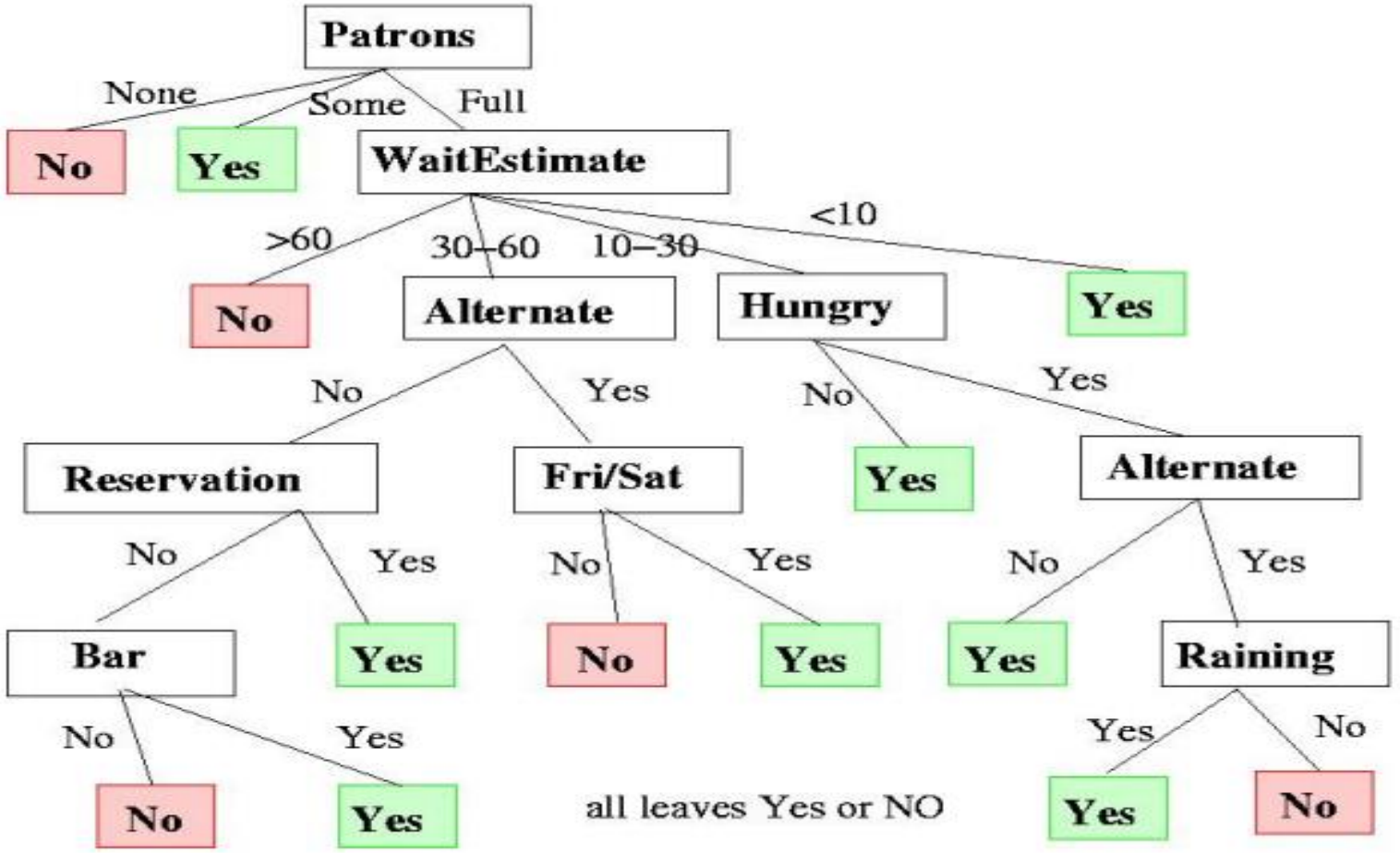
(g) raining: raining outside

(h) reservation: whether we made a reservation

(i) type: kind of restaurant (French, Italian, Thai, or Burger)

(j) WaitEstimate: estimated wait (6o)

Original Decision Tree



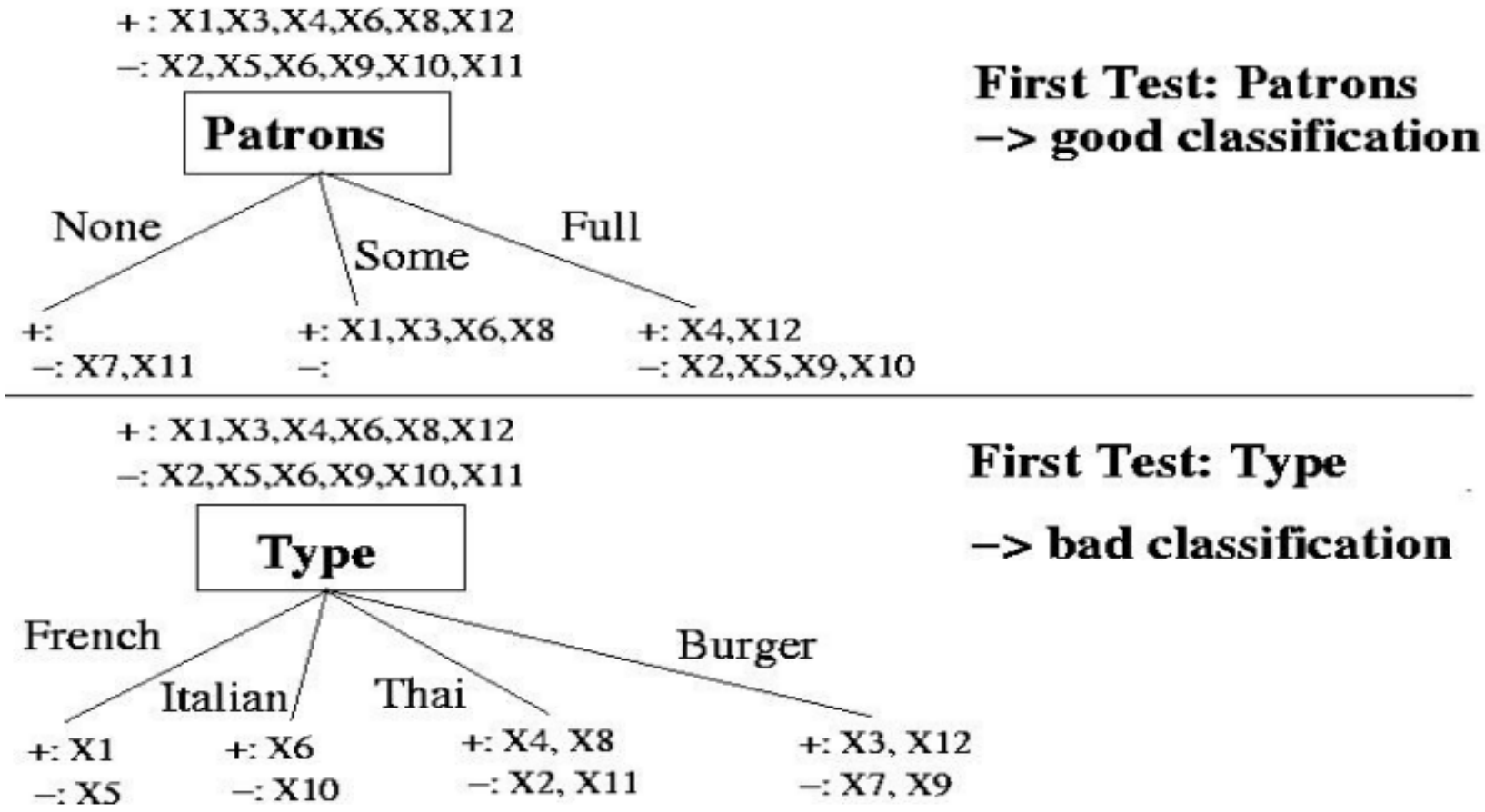
Some Examples

Ex	Attributes										Goal
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	Wait
X_1	Yes	No	No	Yes	Some	£££	No	Yes	French	0-10	Yes
X_2	Yes	No	No	Yes	Full	£	No	No	Thai	30-60	No
X_3	No	Yes	No	No	Some	£	No	No	Burger	0-10	Yes
X_4	Yes	No	Yes	Yes	Full	£	Yes	No	Thai	10-30	Yes
X_5	Yes	No	Yes	No	Full	£££	No	Yes	French	>60	No
X_6	No	Yes	No	Yes	Some	££	Yes	Yes	Italian	0-10	Yes
X_7	No	Yes	No	No	None	£	Yes	No	Burger	0-10	No
X_8	No	No	No	Yes	Some	££	Yes	Yes	Thai	0-10	Yes
X_9	No	Yes	Yes	No	Full	£	Yes	No	Burger	>60	No
X_{10}	Yes	Yes	Yes	Yes	Full	£££	No	Yes	Italian	10-30	No
X_{11}	No	No	No	No	None	£	No	No	Thai	0-10	No
X_{12}	Yes	Yes	Yes	Yes	Full	£	No	No	Burger	30-60	Yes

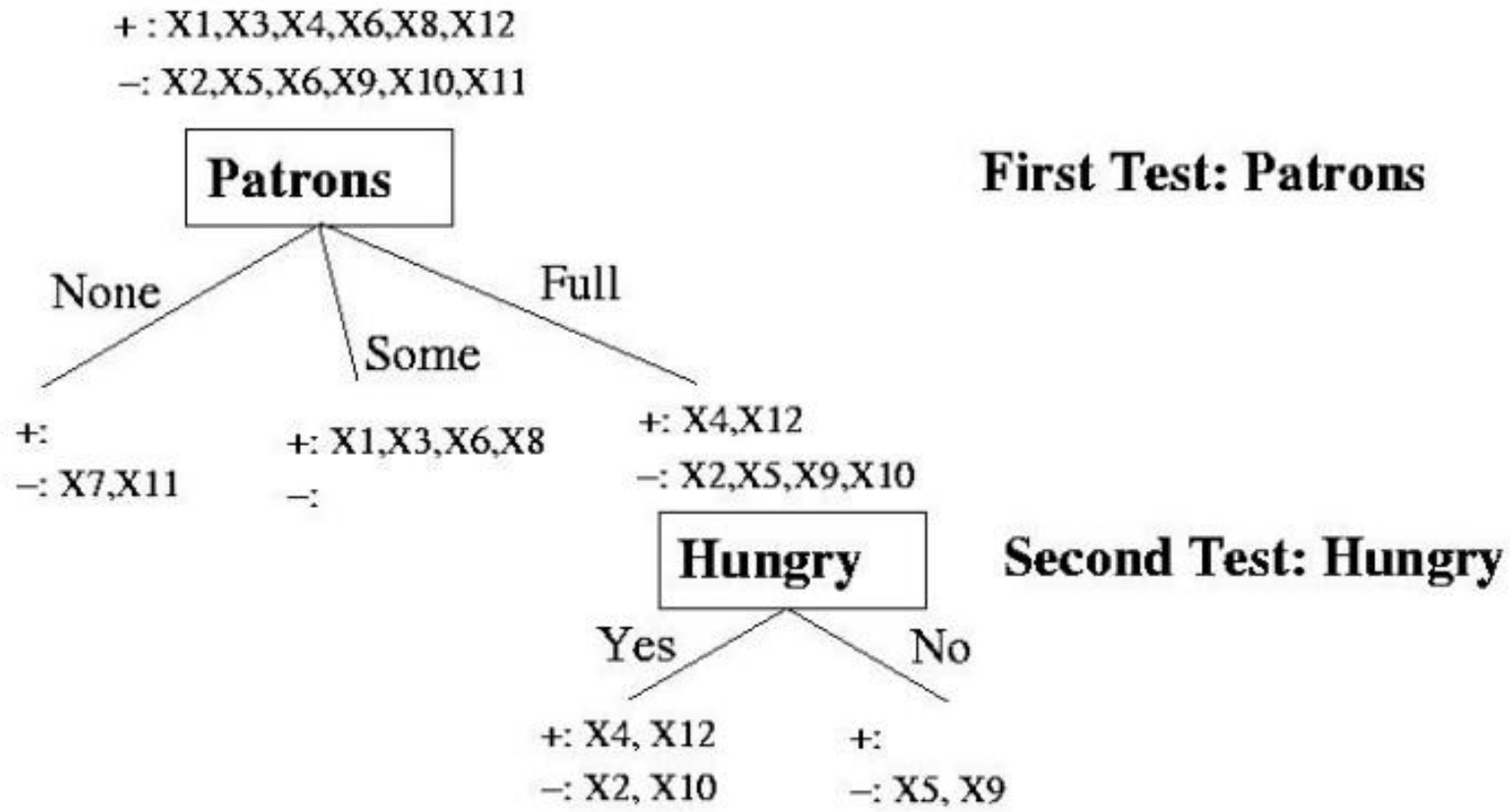
Different Solutions

- Trivial solution: Construct decision tree that has one path to a leaf for each example
- Given the examples o.k., but else bad
- Finding smallest decision trees is intractable, hence heuristic decisions: test the most important attribute first. Most important = makes most difference to the classification of an example.
- A perfect attribute divides the examples into sets that contain only instances of one class
- A really useless attribute leaves the example sets with roughly the same proportion of instances of all classes as the original set
- Short paths in the trees, small trees
- Compare splitting the examples by testing on attributes (cf. Patrons, Type)

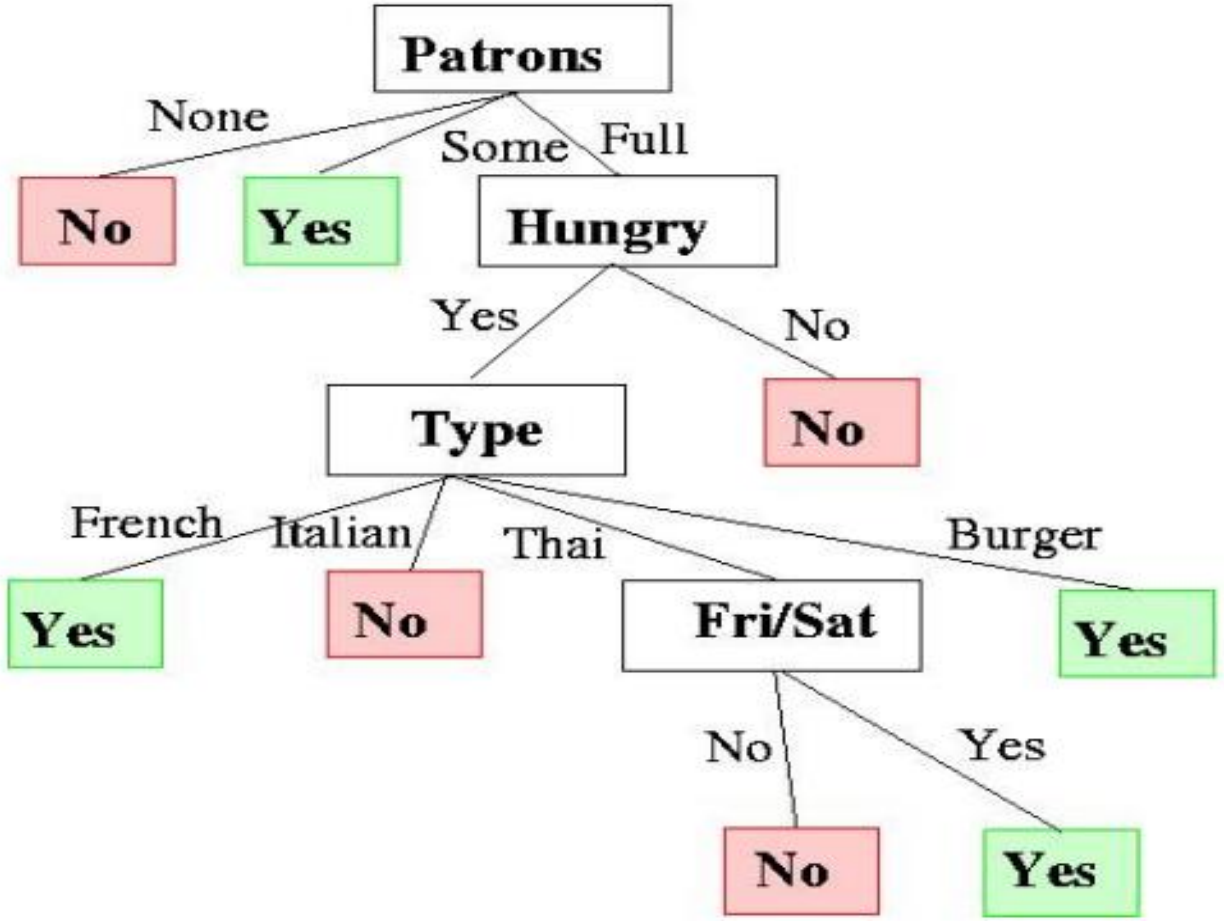
Selecting Best Attributes



Selecting Best Attributes (Continued)



Generated Decision Tree



Discussion of the Result

Comparison of original tree and learned tree:

- Trees differ.
- Learned tree is smaller (no test for raining and reservation, since all examples can be classified without them).
- Detects regularities (waiting for Thai food on weekends).
- Can make mistakes (e.g. case where the wait is < 10, but the restaurant is full and not hungry).

Assessing the Performance

- Collect a large set of examples.
- Divide it into two disjoint sets: training set and test set.
- Learn algorithm with training set and generate hypothesis H.
- Measure percentage of examples in test set that are correctly classified by H.
- Repeat steps 1 to 4 for different sets.

Reinforcement Learning (RL)

- Reinforcement Learning(RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.
- Though both supervised and reinforcement learning use mapping between input and output, unlike supervised learning where feedback provided to the agent is correct set of actions for performing a task, reinforcement learning uses rewards and punishment as signals for positive and negative behavior.
- As compared to unsupervised learning, reinforcement learning is different in terms of goals. While the goal in unsupervised learning is to find similarities and differences between data points, in reinforcement learning the goal is to find a suitable action model that would maximize the total cumulative reward of the agent.

The Idea

- The idea behind Reinforcement Learning is that an agent will learn from the environment by interacting with it and receiving rewards for performing actions.

The Idea

- Learning from interaction with the environment comes from our natural experiences.
- Imagine you're a child in a living room. You see a fireplace, and you approach it.

The Idea

- It's warm, it's positive, you feel good *(Positive Reward +1).* You understand that fire is a positive thing.

The Idea

- But then you try to touch the fire.
- Ouch! It burns your hand *(Negative reward -1)*.
- You've just understood that fire is positive when you are a sufficient distance away, because it produces warmth. But get too close to it and you will be burned.

The Idea

- That's how humans learn, through interaction.
- Reinforcement Learning is just a computational approach of learning from action.

The Reinforcement Learning Process

Example

- Let's imagine an agent learning to play Super Mario Bros as a working example. The Reinforcement Learning (RL) process can be modeled as a loop that works like this:
  - Our Agent receives **state So** from the **Environment** (In our case we receive the first frame of our game (state) from Super Mario Bros (environment))
  - Based on that **state So,** agent takes an **action Ao** (our agent will move right)
  - Environment transitions to a **new state S1** (new frame)
  - Environment gives some **reward R1** to the agent (not dead: +1)
  - This RL loop outputs a sequence of **state, action and reward.**
  - The goal of the agent is to maximize the expected cumulative reward.

The central idea of the Reward Hypothesis

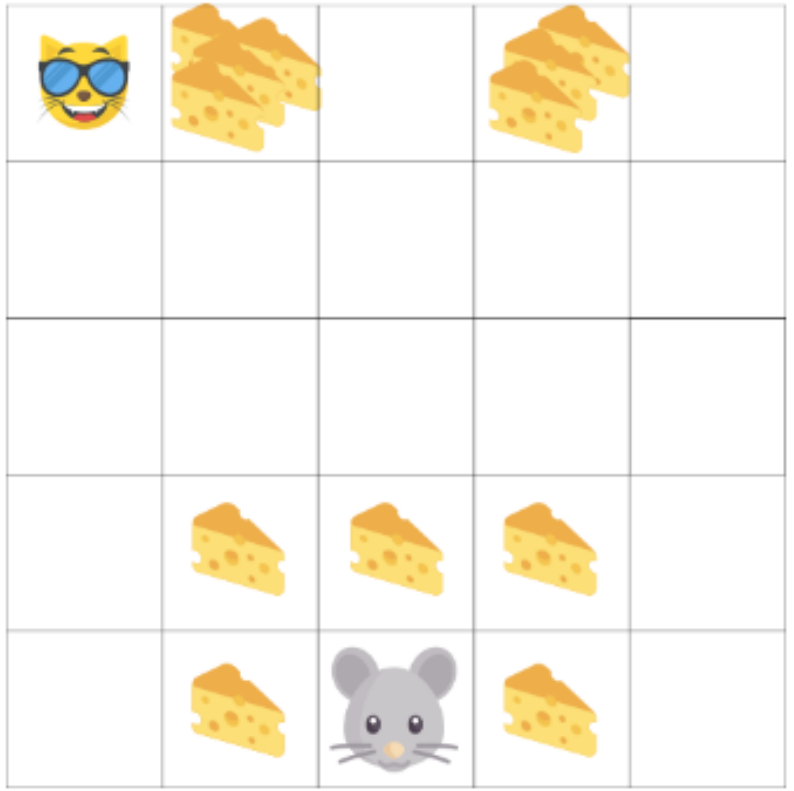
- Why is the goal of the agent to maximize the expected cumulative reward?
- Well, Reinforcement Learning is based on the idea of the reward hypothesis. All goals can be described by the maximization of the expected cumulative reward.
- **That's why in Reinforcement Learning, to have the best behavior, we need to maximize the expected cumulative reward.**

Cumulative Reward

- The cumulative reward at each time step t can be written as:
- $G_t = R_{t+1} + R_{t+2} + \ldots$
- Which is equivalent to:
- $G_t = \sum_{k=0}^{T} R_{t+k+1}$

The Problem

- However, in reality, we can't just add the rewards like that. The rewards that come sooner (in the beginning of the game) are more probable to happen, since they are more predictable than the long term future reward.
- Let say your agent is this small mouse and your opponent is the cat. Your goal is to eat the maximum amount of cheese before being eaten by the cat.
- As we can see in the diagram, it's more probable to eat the cheese near us than the cheese close to the cat (the closer we are to the cat, the more dangerous it is).
- As a consequence, the reward near the cat, even if it is bigger (more cheese), will be discounted. We're not really sure we'll be able to eat it.



Discount The Rewards

- To discount the rewards, we proceed like this:
  - We define a discount rate called gamma. It must be between 0 and 1.
  - The larger the gamma, the smaller the discount. This means the learning agent cares more about the long term reward.
  - On the other hand, the smaller the gamma, the bigger the discount. This means our agent cares more about the short term reward (the nearest cheese).
- Our discounted cumulative expected rewards is:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} where \gamma \in [0, 1)$$

$$R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots$$

Discount The Rewards

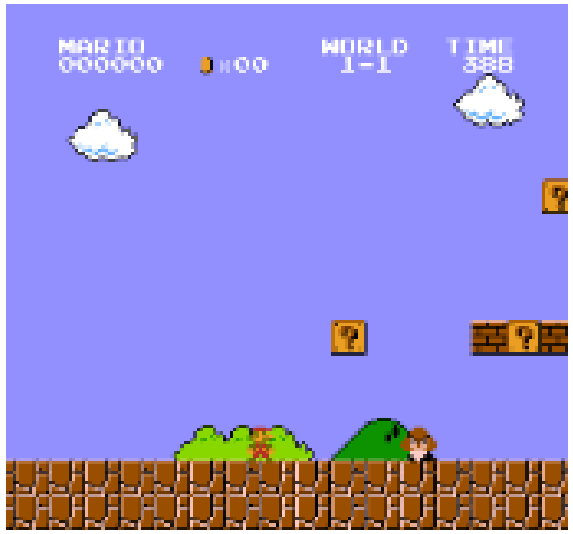
- To be simple, each reward will be discounted by gamma to the exponent of the time step.
- As the time step increases, the cat gets closer to us, so the future reward is less and less probable to happen.

Episodic or Continuing Tasks

- A task is an instance of a Reinforcement Learning problem. We can have two types of tasks:
  - episodic and
  - continuous.

Episodic Tasks

- In this case, we have a starting point and an ending point **(a terminal state). This creates an episode**: a list of States, Actions, Rewards, and New States.
- For instance think about Super Mario Bros, an episode begin at the launch of a new Mario and ending: when you're killed or you're reach the end of the level.



Beginning of a new episode

Continuous Tasks

- **These are tasks that continue forever (no terminal state).** In this case, the agent has to learn how to choose the best actions and simultaneously interacts with the environment.
- For instance, an agent that do automated stock trading. For this task, there is no starting point and terminal state. **The agent keeps running until we decide to stop him.**

Monte Carlo vs TD Learning methods

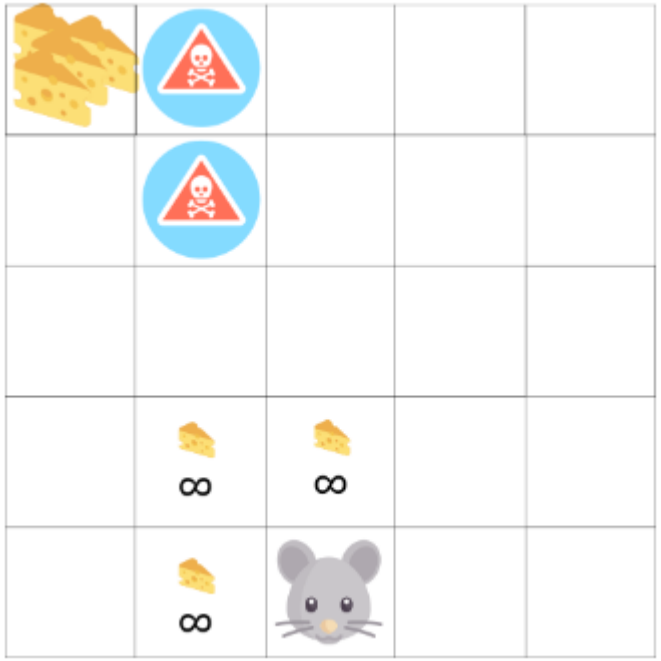
- We have two ways of learning:
- Collecting the rewards **at the end of the episode** and then calculating the **maximum expected future reward**: *Monte Carlo Approach*
- Estimate **the rewards at each step**: *Temporal Difference Learning*

Exploration/Exploitation Trade-off

- **Exploration** is finding more information about the environment.
- **Exploitation** is exploiting known information to maximize the reward.
- Remember, the goal of our RL agent is to maximize the expected cumulative reward. However, we can fall into a common trap.

Example

- In this game, our mouse can have an infinite amount of small cheese (+1 each). But at the top of the maze there is a gigantic sum of cheese (+1000).
- However, if we only focus on reward, our agent will never reach the gigantic sum of cheese. Instead, it will only exploit the nearest source of rewards, even if this source is small (exploitation).
- But if our agent does a little bit of exploration, it can find the big reward.
- This is what we call the exploration/exploitation trade off. We must define a rule that helps to handle this trade-off.



Example

- If we take the maze environment:
- We always start at the same starting point.
- We terminate the episode if the cat eats us or if we move > 20 steps.
- At the end of the episode, we have a list of State, Actions, Rewards, and New States.
- The agent will sum the total rewards Gt (to see how well it did).
- It will then update V(st) based on the formula above.
- Then start a new game with this new knowledge.
- By running more and more episodes, **the agent will learn to play better and better.**