# INTRODUCTION TO

# REINFORCEMENT LEARNING

**PRESENTED BY : GROUP 1**

# AGENDAX

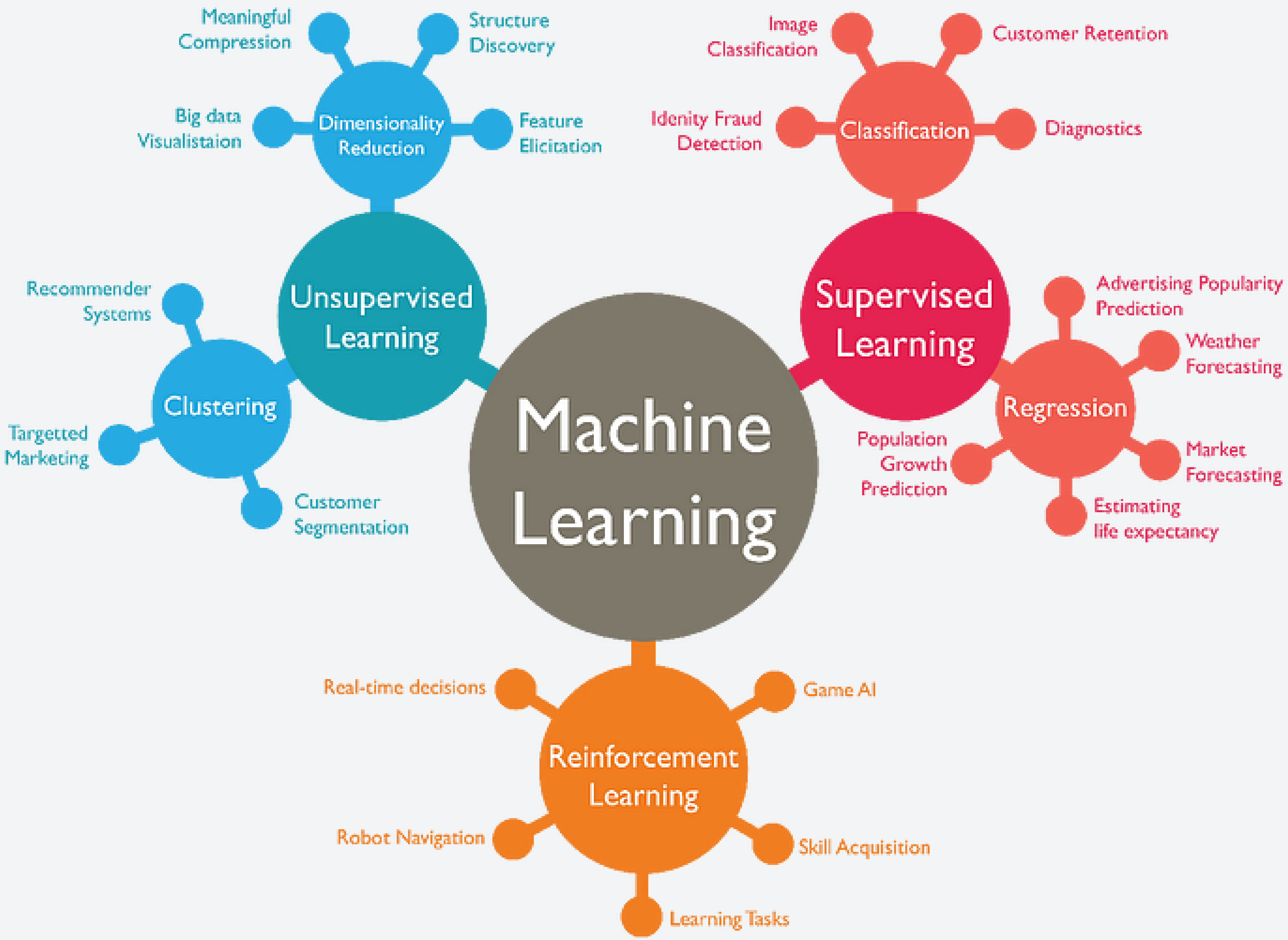
01 OVERVIEW OF REINFORCEMENT LEARNING

02 KEY CONCEPTS

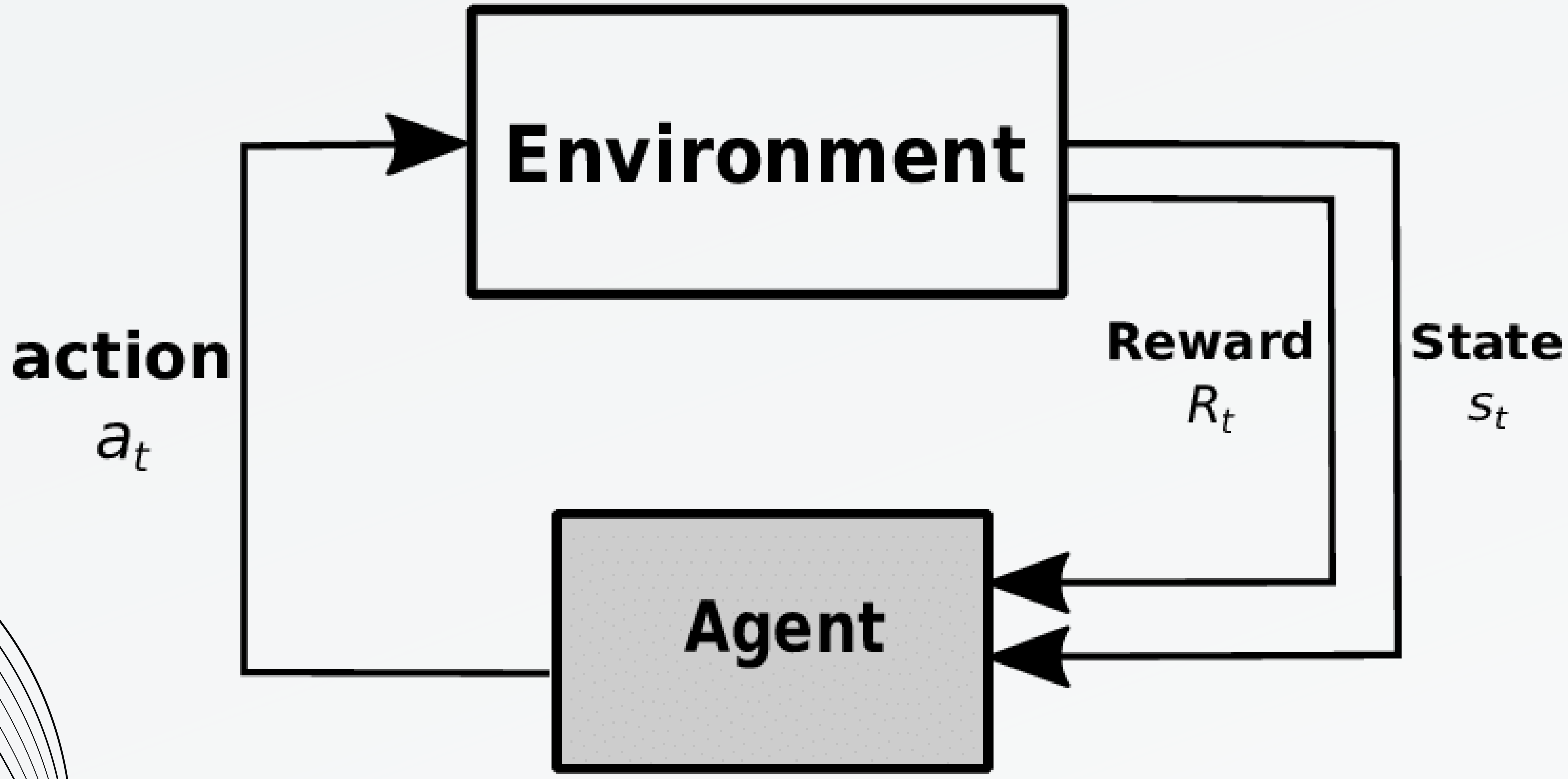
03 COMPONENTS OF RL

04 EXAMPLES

05 PRACTICAL EXAMPLE


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

# INTRODUCTION TO REINFORCEMENT LEARNING




The goal is to train system based on interaction with dynamic environment.
Environment provides feedback to the system in terms of rewards and punishment.
The system can then use the reinforcement learning to learn a series of actions that maximize its reward.
An example is a chess-playing engine.

# KEY CONCEPTS

## AGENT

The entity making decisions in the environment, The agent's objective is typically to maximize a cumulative reward signal over time.

## ENVIRONMENT

the external system with which the agent interacts. It is a crucial component of the RL framework and plays a key role in shaping the agent's learning process.

## ACTION

a decision or move that an agent can take in a given state of the environment. Actions are the means by which the agent interacts with and influences the environment

## REWARD

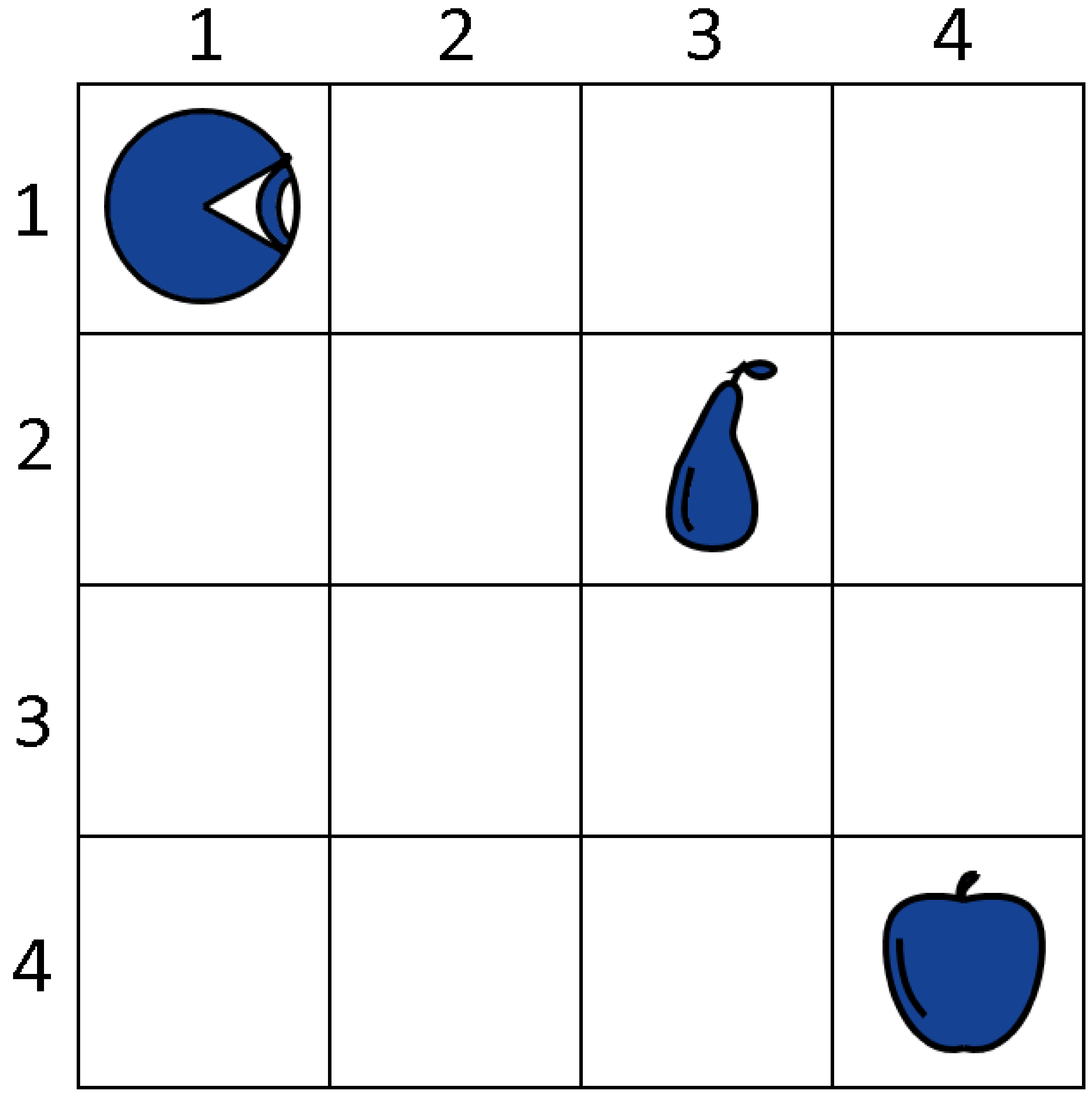
The reward is a crucial element in the RL framework, serving as feedback to the agent about the quality of its actions and guiding the learning process.

# COMPONENTS OF RL

**Policy**: Strategy or plan that the agent uses to determine its actions.

In this example, an agent has to forage food from the environment in order to satisfy its hunger. It then receives rewards on the basis of the fruit it eat

The action space, in this example, consists of four possible behaviors: A=up, dwon, left, right

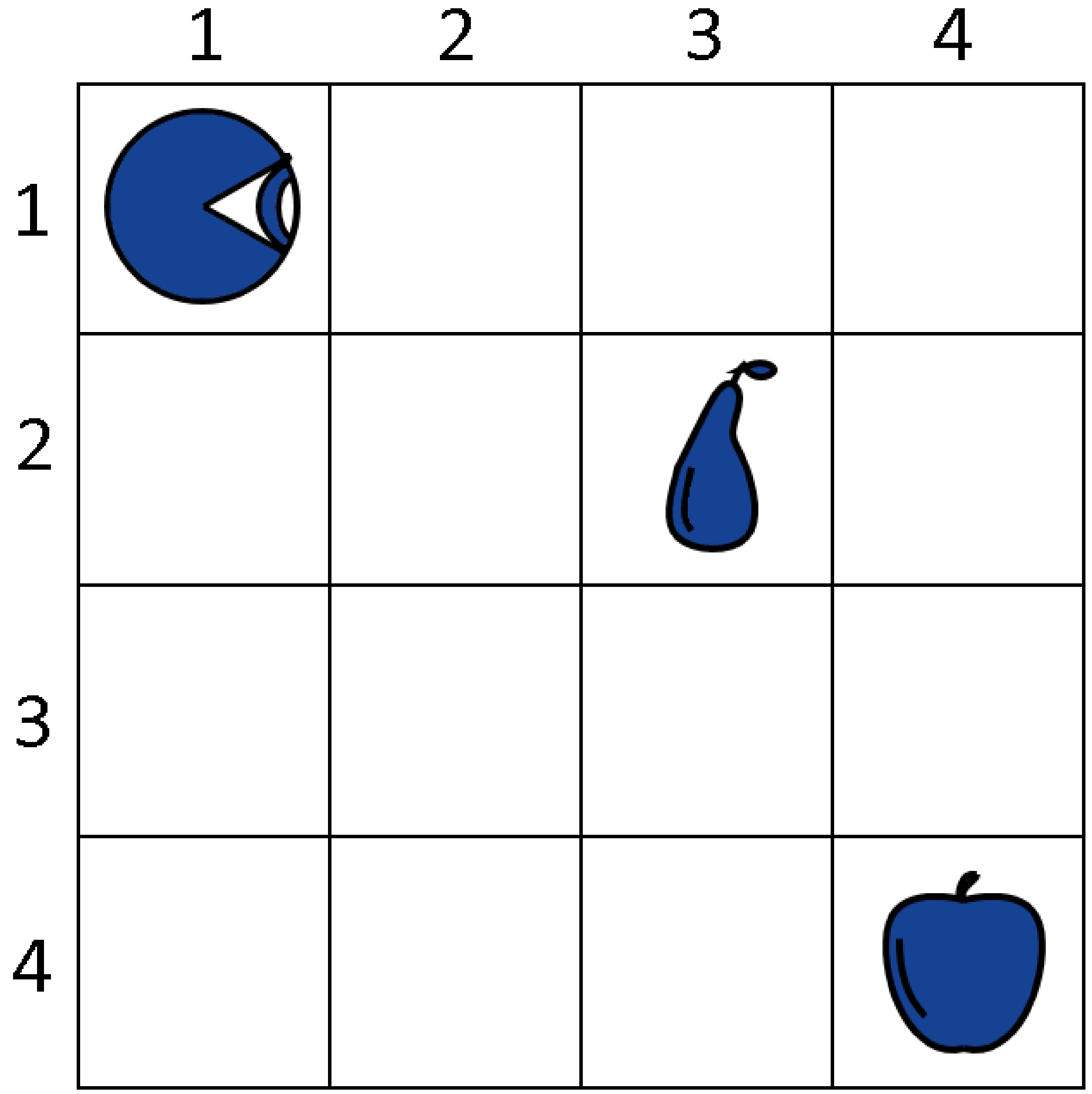


# COMPONENTS OF RL

**Policy**

The reward function R thus looks like this:
R(nothing)=-1
R(apple)=+10
R(pear)=+5

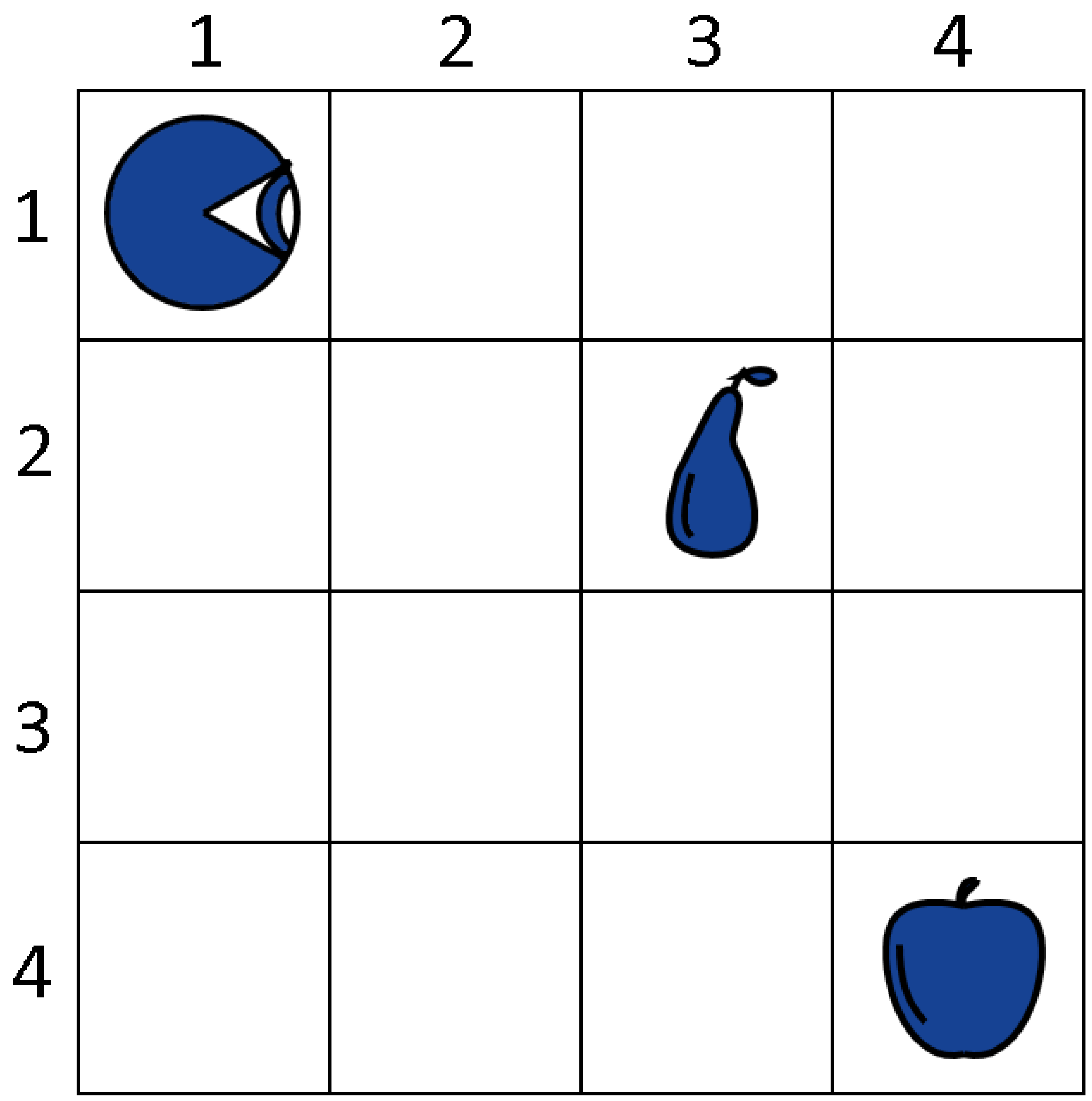
The simulation runs for an arbitrary finite number of time steps but terminates early if the agent reaches any fruit.



# COMPONENTS OF RL

**Policy**

The reward function  is defined as follows:
If it's in an empty cell, the agent receives a negative reward of -1, to simulate the effect of hunger.
If instead, the agent is in a cell with fruit, in this case,  for the pear (2,3) and for the apple(4,4), it then receives a reward of +5 and +10, respectively.



# COMPONENTS OF RL

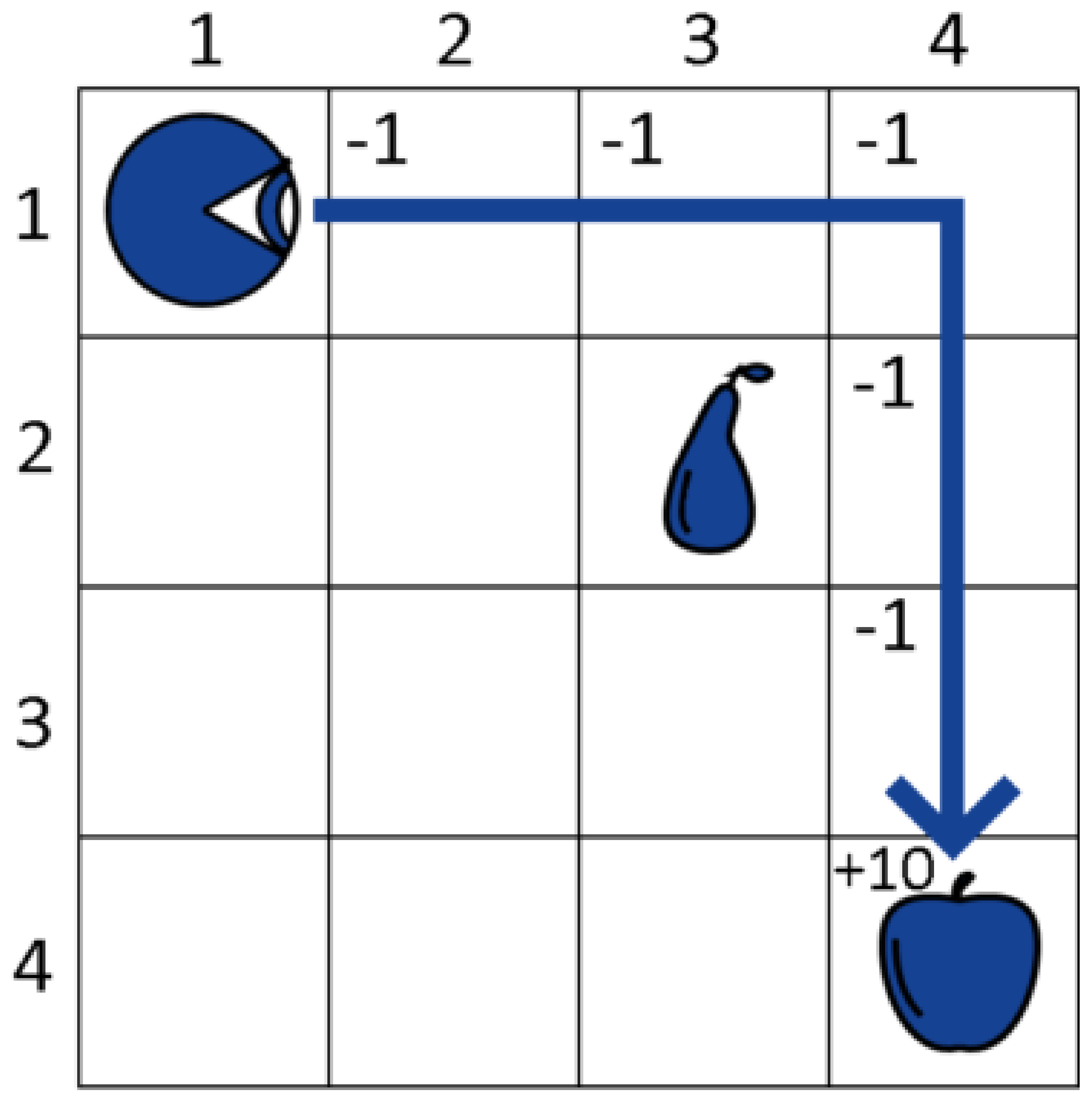
**Policy-> Evaluation**

The agent then has to select between the two policies. By computing the utility function U over them, the agent obtains:

U(P1)=-1-1-1+5=3

U(P2)=-1-1-1-1-1+10=5

The evaluation of the policies suggests that the utility is maximized with P2,
which then the agent chooses as its policy for this task.



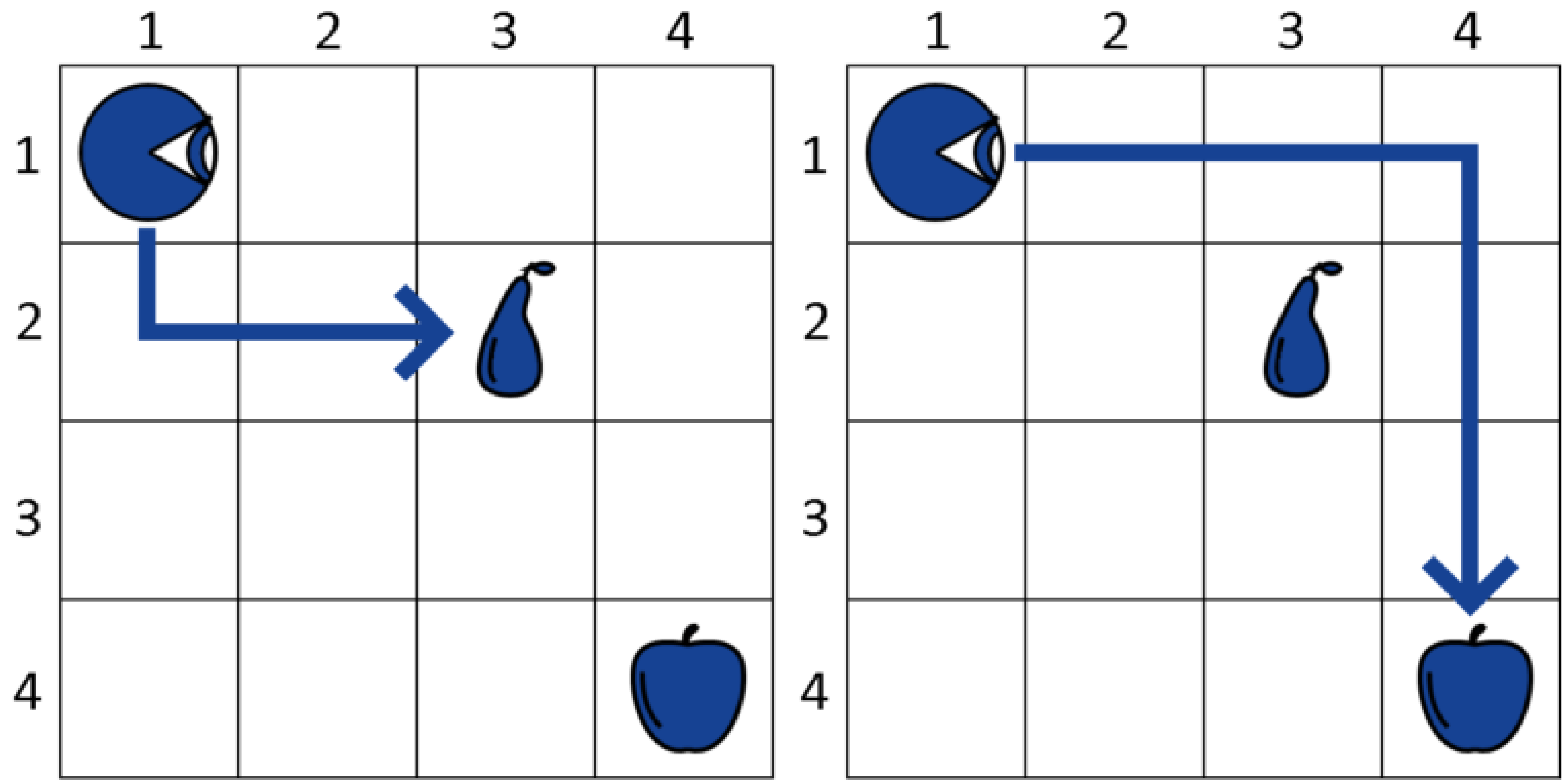
# COMPONENTS OF RL

**Policy-> Evaluation**

The agent then considers two policies p1 and p2. If we simplify slightly the notation, we can indicate a policy as a sequence of actions starting from the state of the agent at the initial state S0:

- P1=down->right->right---> PEAR
- P2=right->right->right->down->down->down->APPLE

P1




P2

# COMPONENTS OF RL

**Value Function**: Represents the expected cumulative future rewards for a given state or state-action pair.

It's often useful to know the value of a state, or state-action pair.
By value, we mean the expected return if you start in that state or state-action pair.
A function that estimates how good it is for the agent to be in a given state

# COMPONENTS OF RL

**Type of value function**

**State-Value Function (V(s)):**

The value of a state is the expected return starting from that state; depends on the agent's policy

**Action-Value Function (Q(s, a)):**

The value of taking an action in a state under the policy π is the expected return starting from that state, taking that action, and thereafter following

# MODEL-FREE VS MODEL-BASED RL

One of the most important branching points in an RL algorithm is the question of whether the agent has access to (or learns) a model of the environment.
By a model of the environment, we mean a function which predicts state transitions and rewards.

**Model-Free Learning :**

- Learn from interacting with the world, Sample reward and Transition function by interacting
- Seek to learn the consequences of their actions through experience ; carry out an action multiple times and adjust the policy for optimal rewards, based on the outcomes.
- Called direct Methods
- Tend to be easier to implement and tune.

.

# MODEL-FREE VS MODEL-BASED RL

**Model-Based Learning :**

- Learn from the model instead of interacting with the world
- The main upside to having a model is that **it allows the agent to plan** by thinking ahead, seeing what would happen for a range of possible choices.
- Called indirect Methods

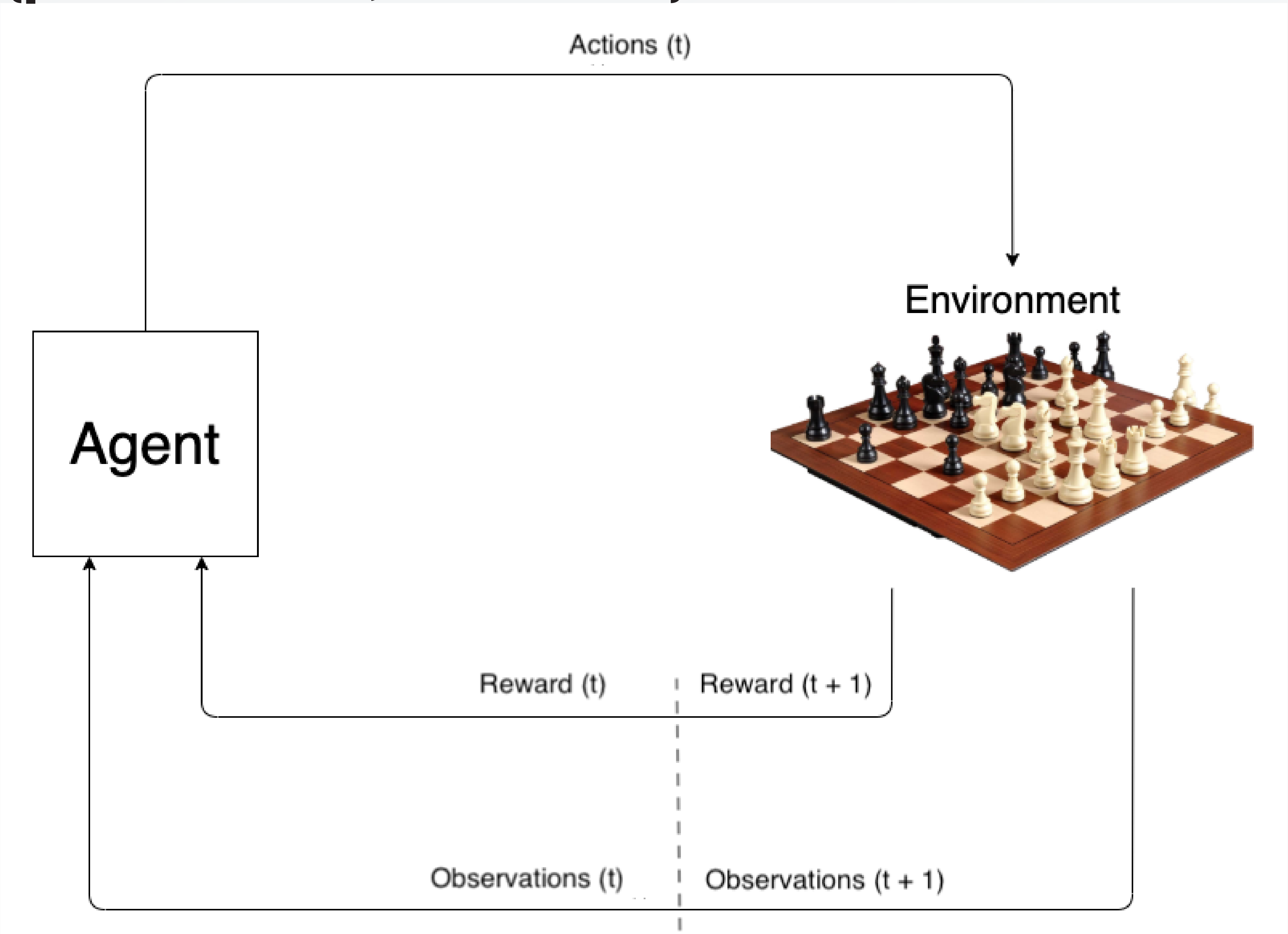
.

# COMPARISON/EVALUATION

| s/n | Model-Free | Model-Based |
|---|---|---|
| 1 | rewards are not accounted for (since this is automated, reward = 1) | rewards are accounted for |
| 2 | no modelling (no decision policy is required) | modelling is required (policy network) |
| 3 | this doesn’t require the use of initial states to predict the next state | this requires the use of initial states to predict the next state using the policy network |
| 4 | the rate of missing the ball with respect to time is zero | the rate of missing the ball with respect to time approaches zero |

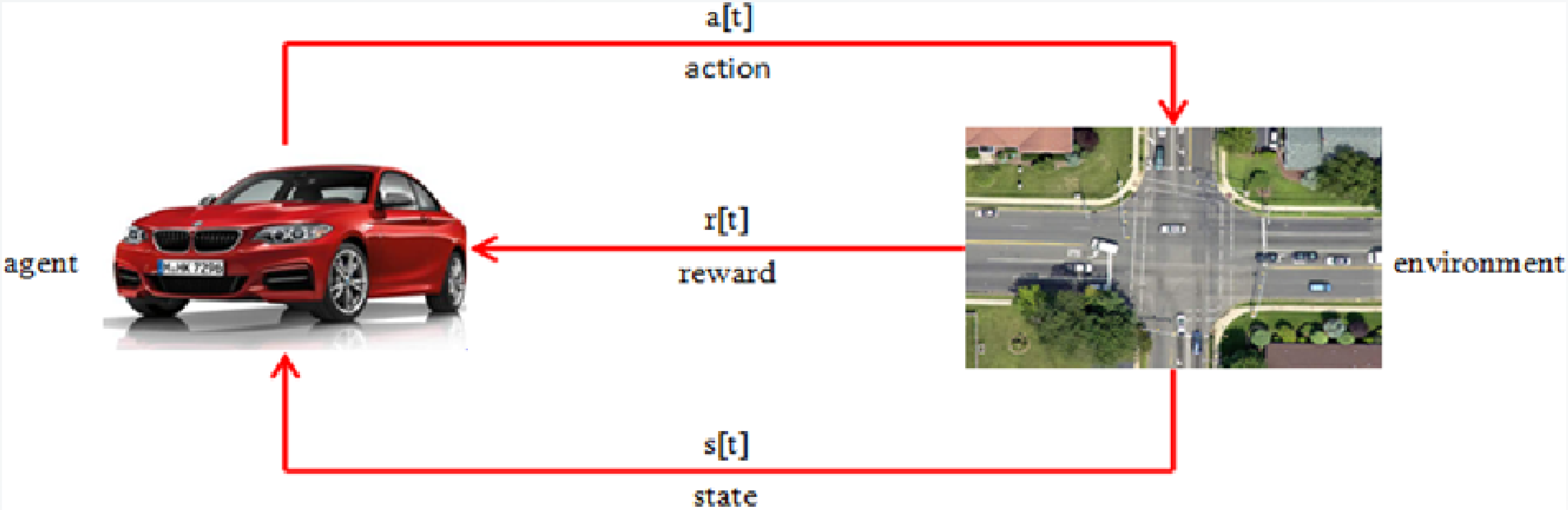
# EXAMPLES OF USAGE

**Learn games (pacman- chess, tennis table):**



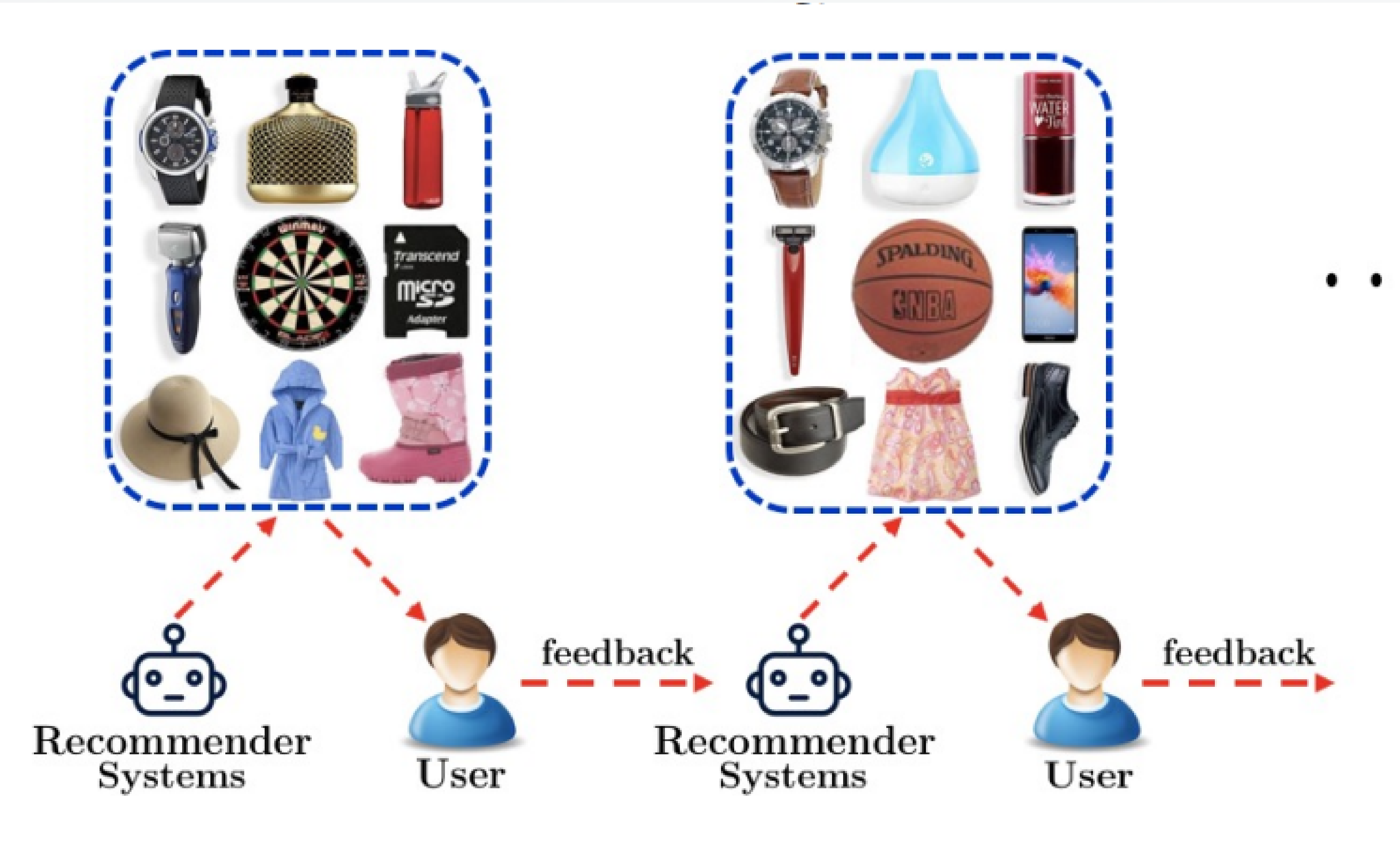
# EXAMPLES OF USAGE

Self driving Cars



# EXAMPLES OF USAGE

Recommendation Systems



# PRACTICAL EXAMPLE