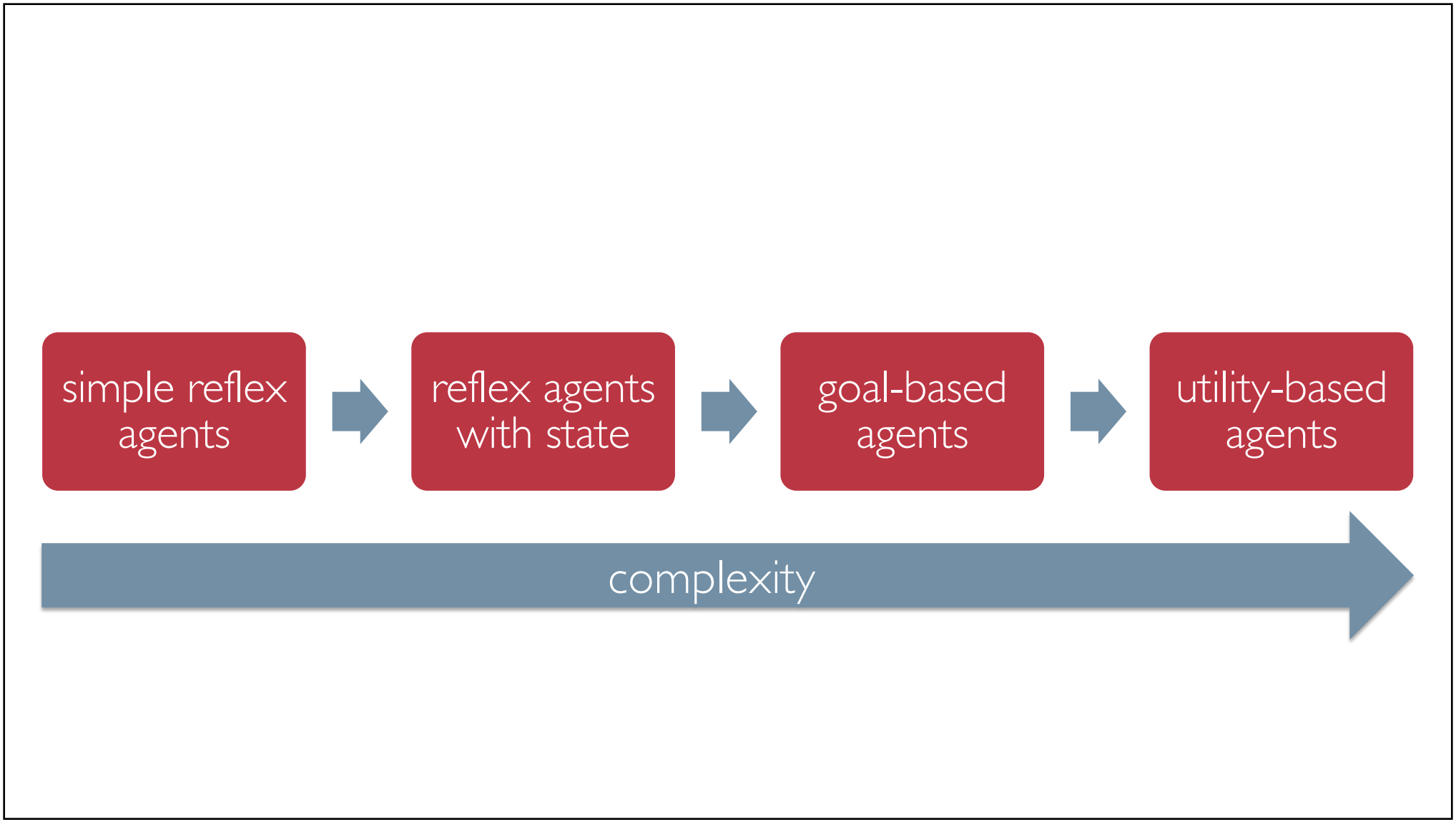
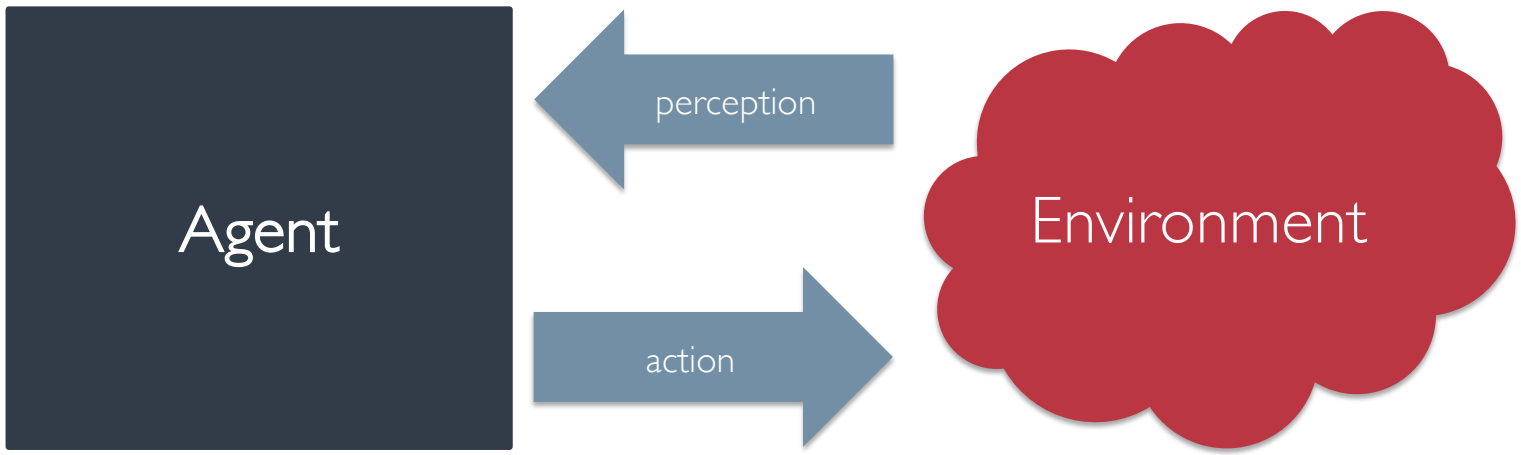
Learning Agents

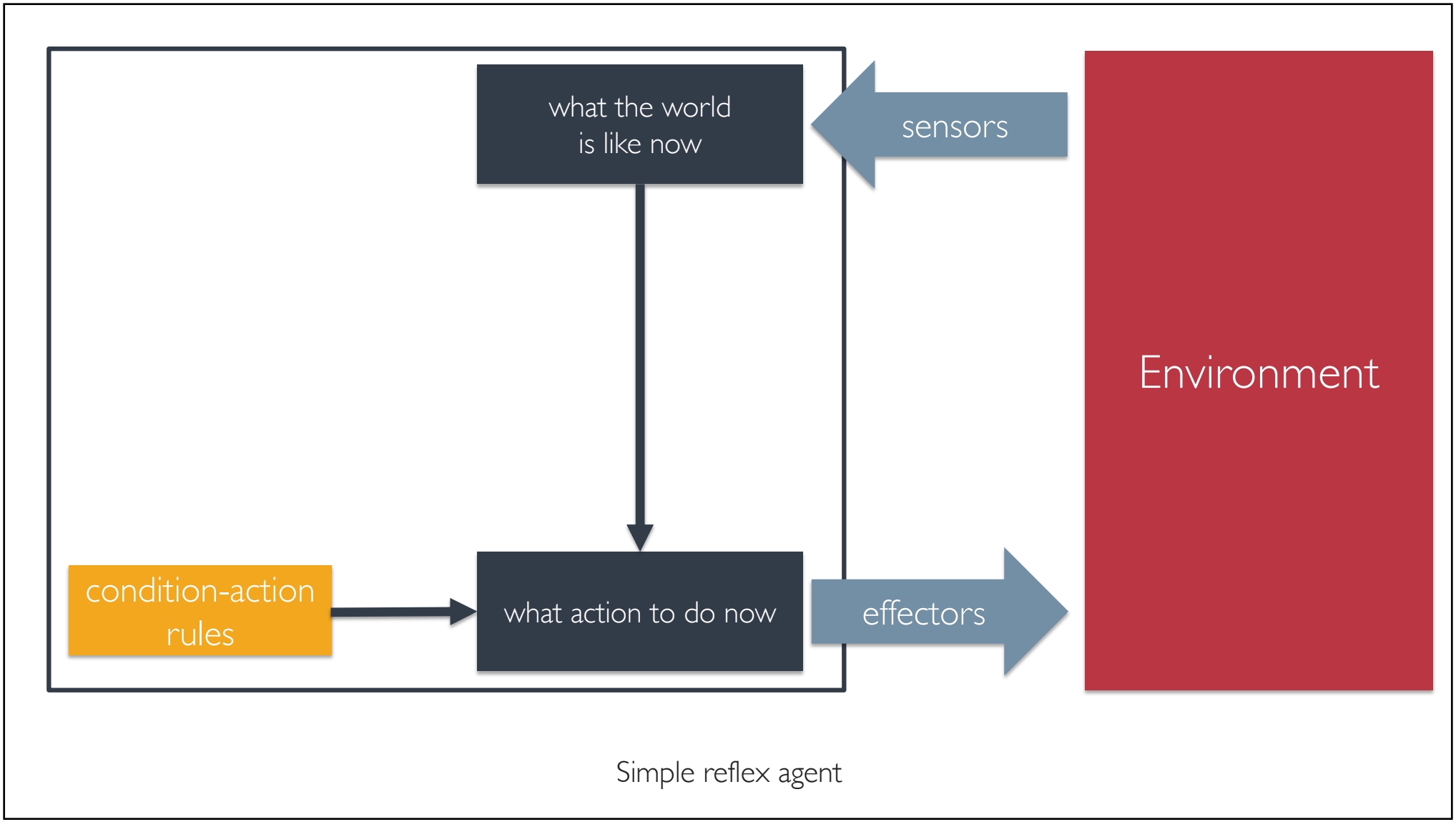
Foundations of Artificial Intelligence

Francesco Amigoni & Pierluca Lanzi

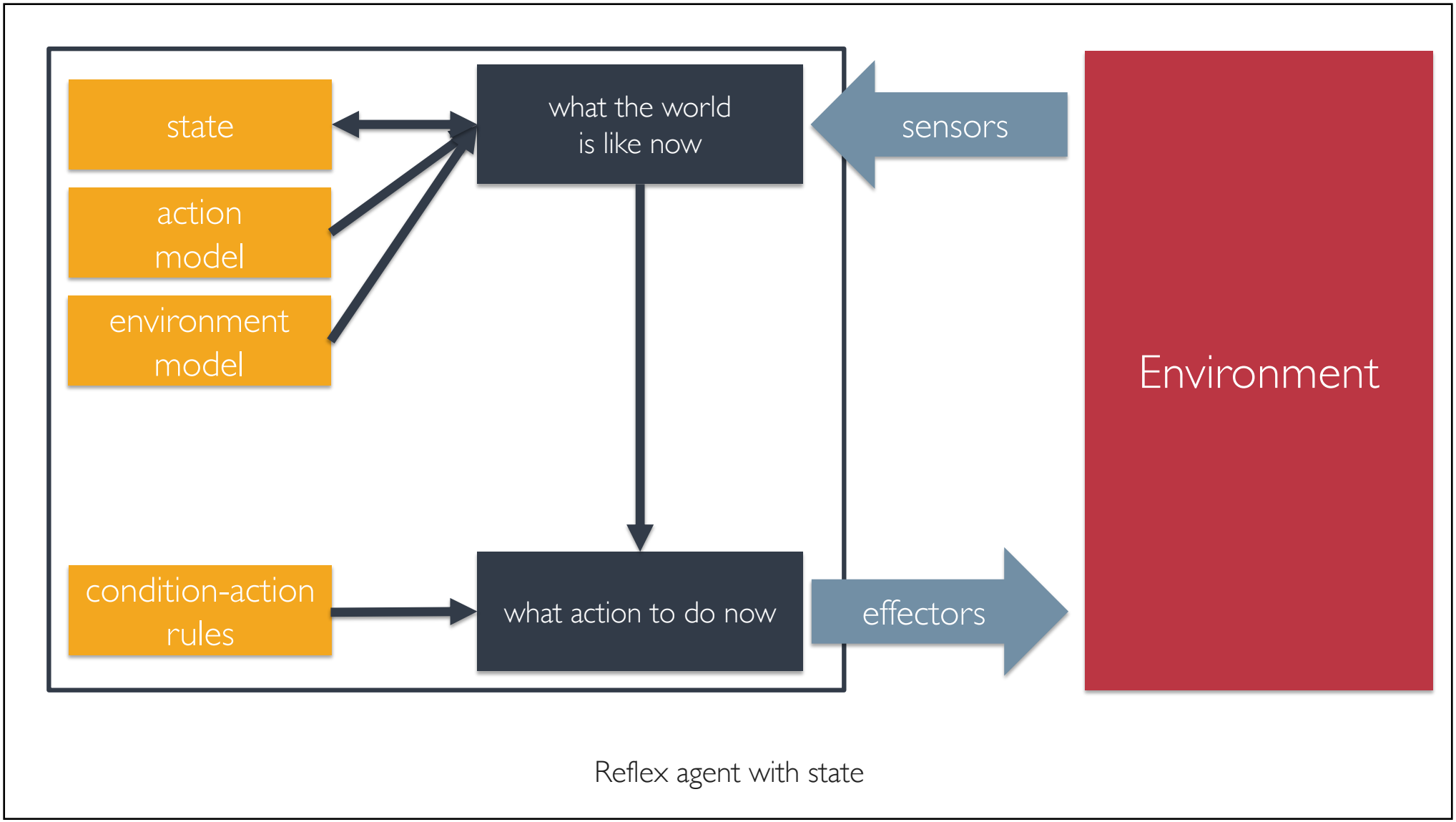
recap


simple reflex agents
reflex agents with state
goal-based agents
utility-based agents
complexity

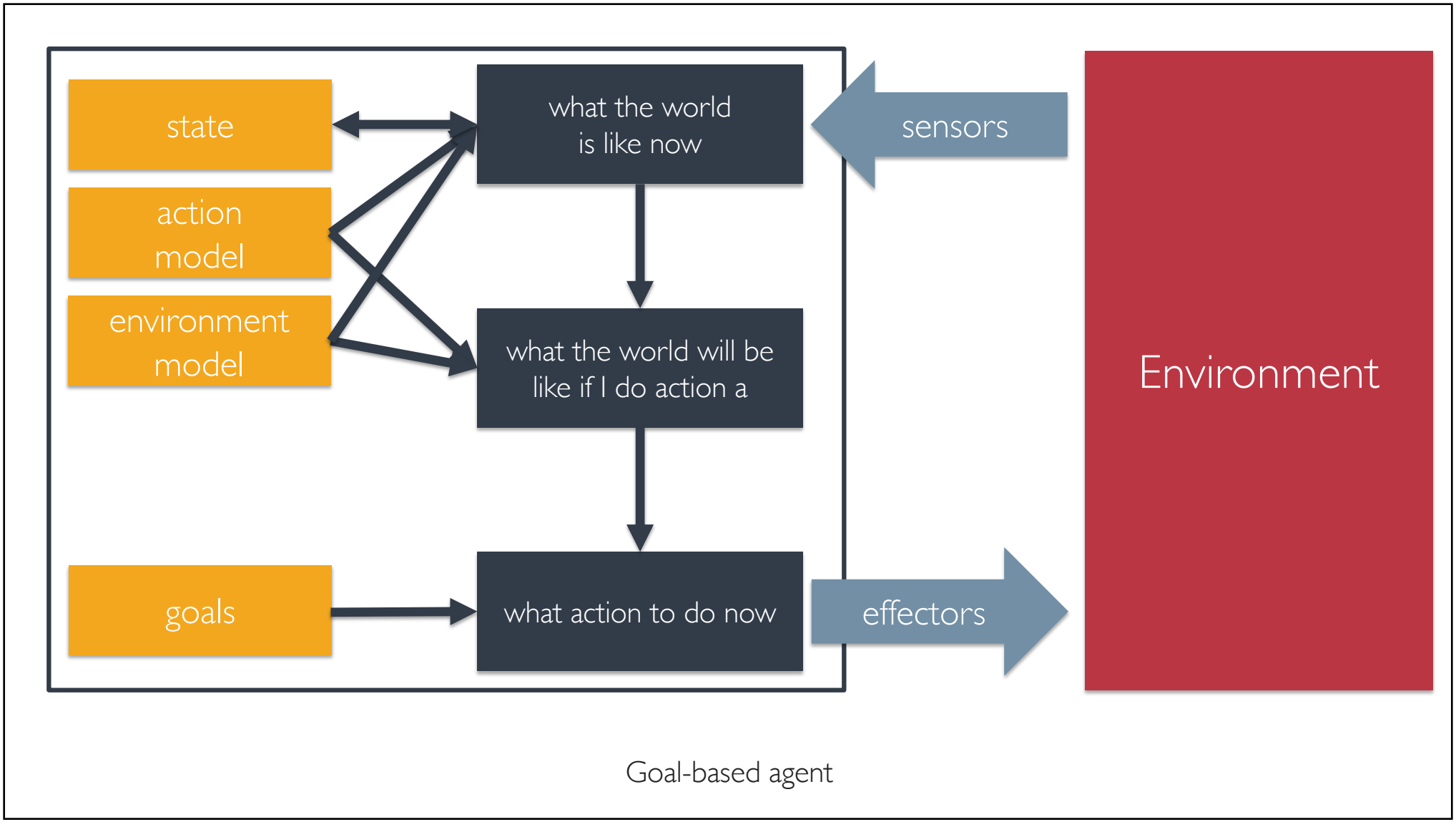

Agent
perception
action
Environment



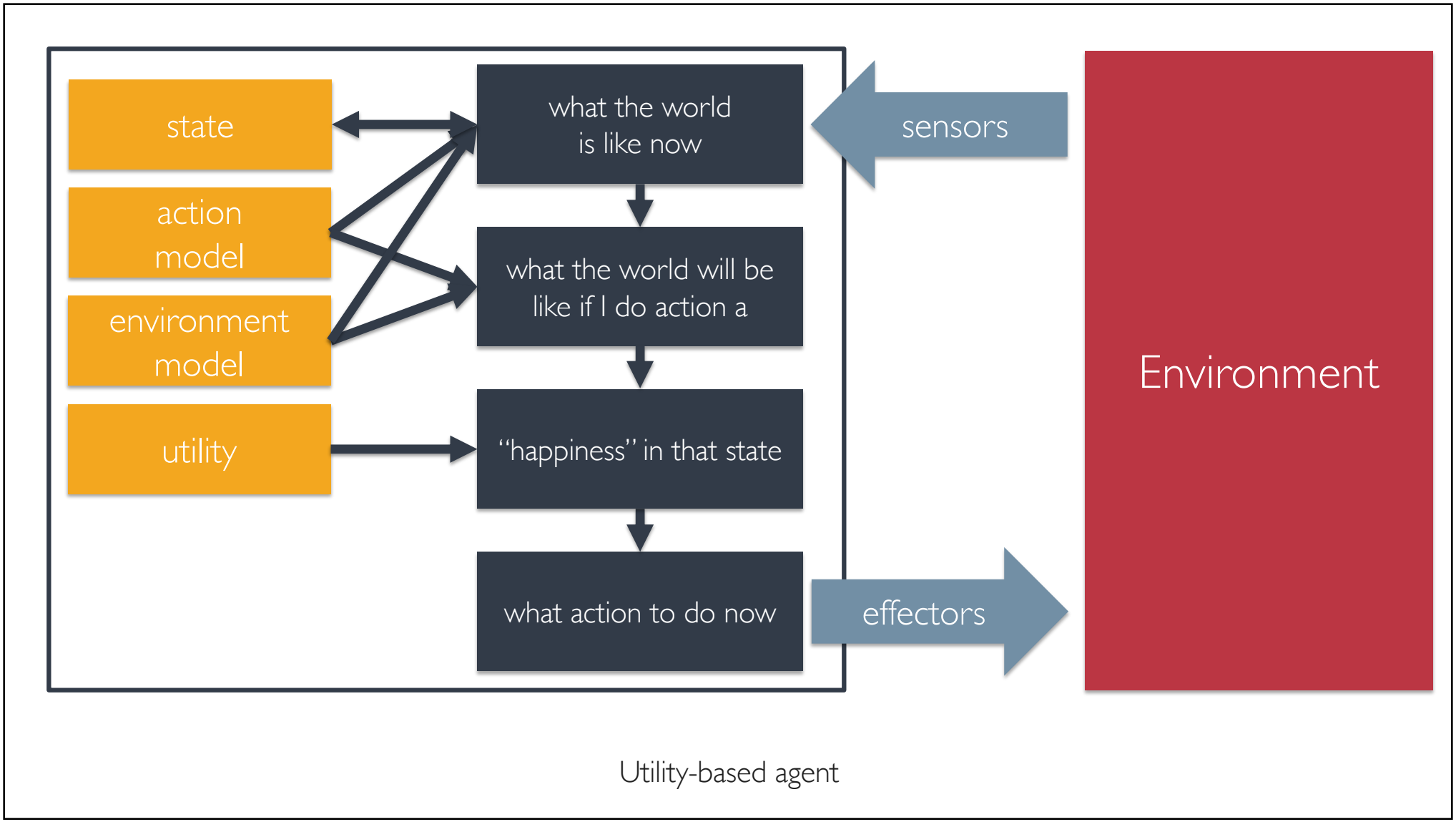
Simple reflex agent



Reflex agent with state



Goal-based agent



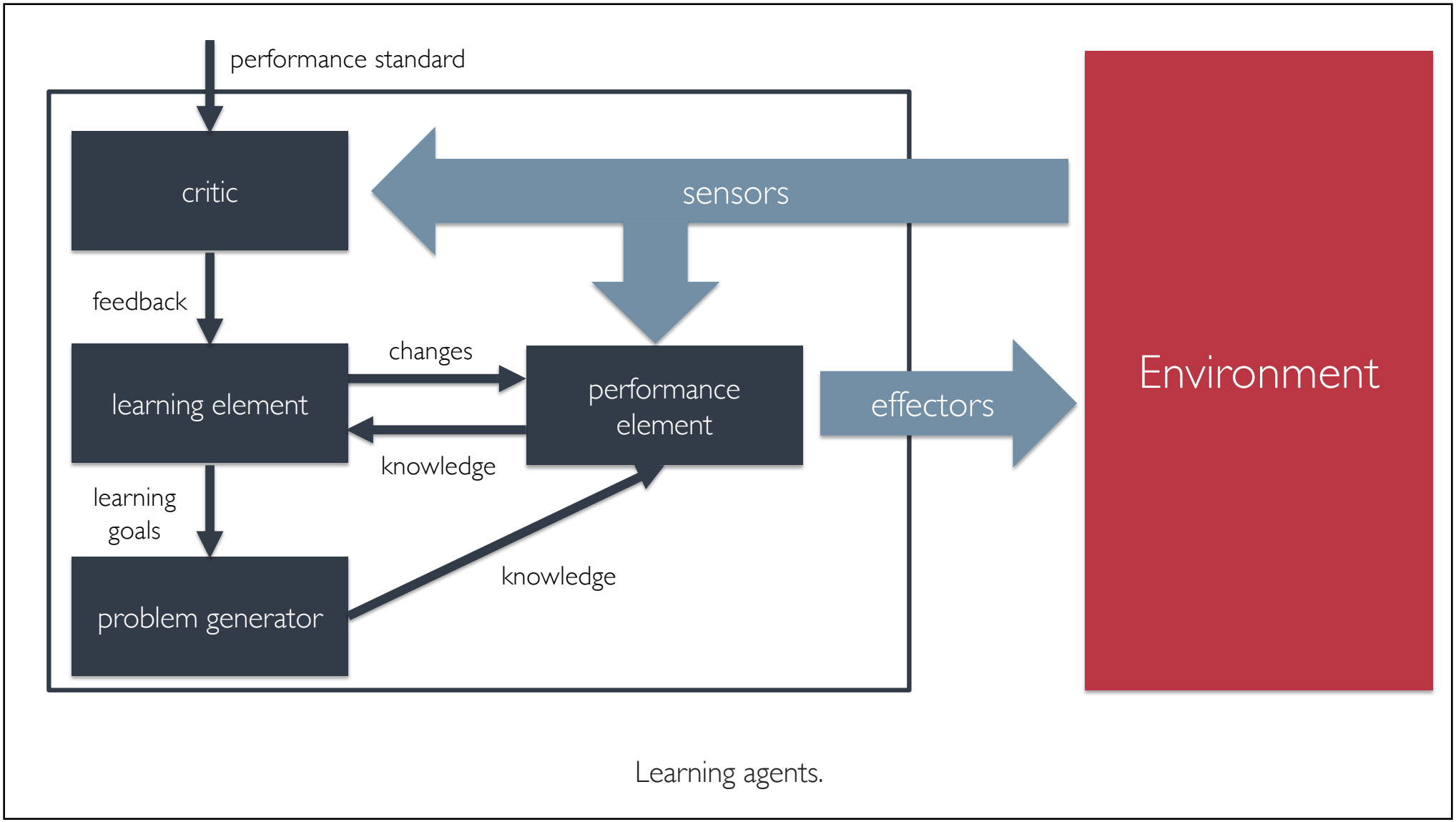
Utility-based agent

learning agents

Learning allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow.

~~"How am I going to get it to learn this?"~~

"What kind of performance element will my agent use to do this once it has learned how?"



Learning agents.

Learning Element

It is responsible for making improvements.

Performance Element

It selects external actions.

Critic

It provides feedback on the agent is doing and determines how the performance element should be modified to improve future performance.

Problem Generator

It suggests exploratory actions that will lead to new and informative experiences.

The performance element analyses the sensor inputs and decide what action to perform.

Thus it is what we have previously considered to be the entire agent.

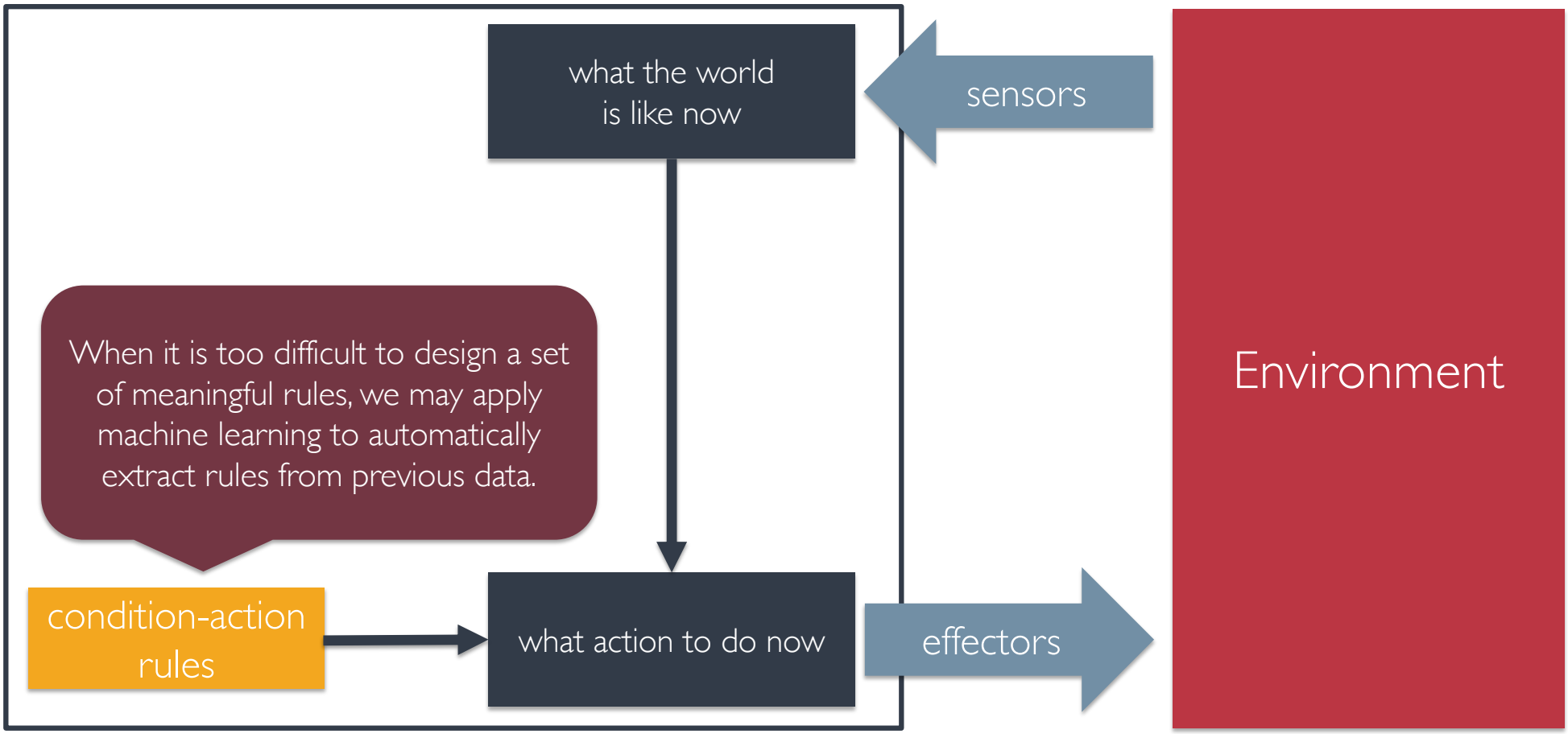
how can a computer program learn?

Machine Learning

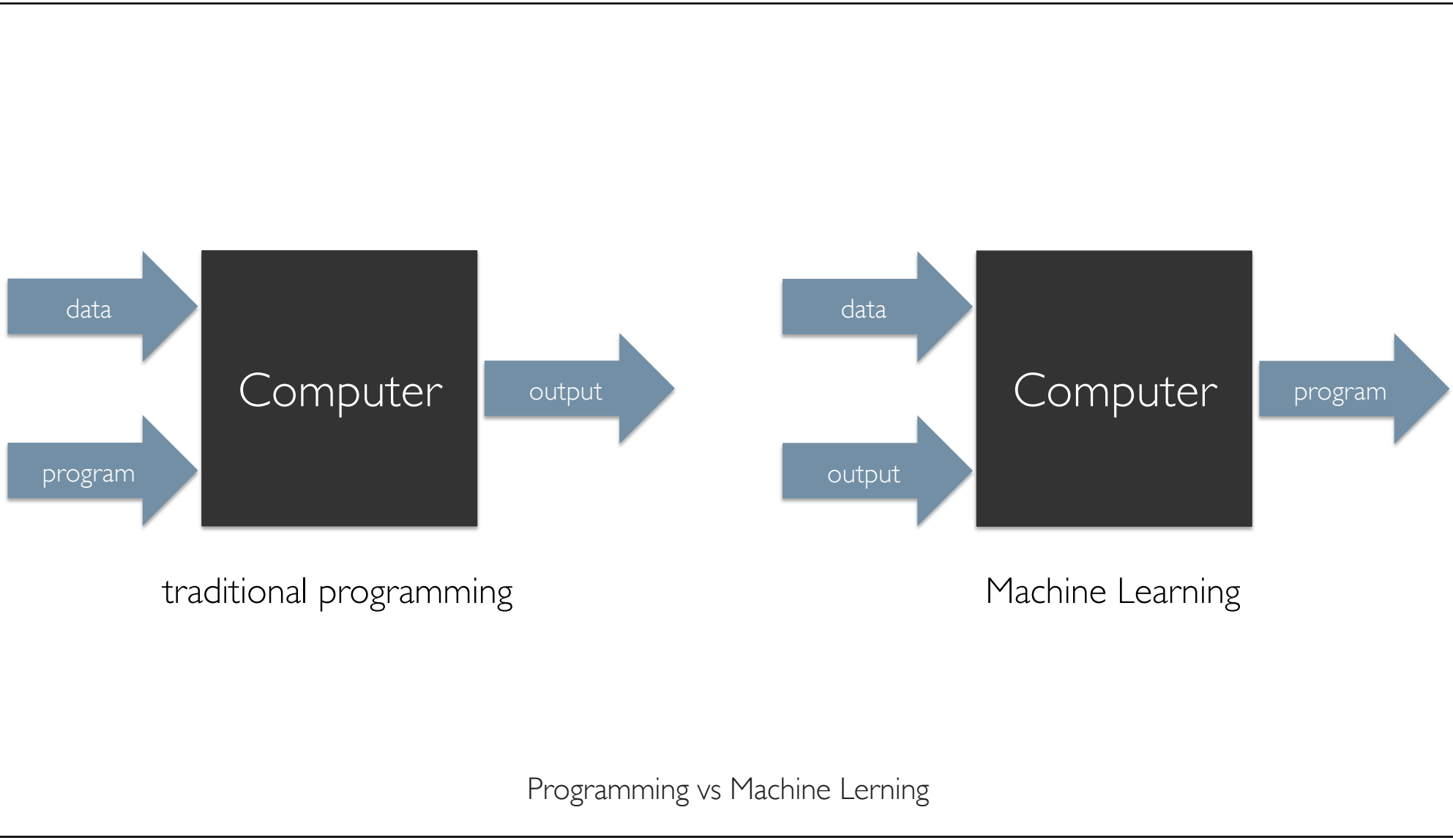
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, improves with experience E" Mitchell (1997)

Machine Learning

- It is an area of Artificial Intelligence focused on building algorithms capable of learning, extracting knowledge from experience.
- Machine Learning algorithms extract knowledge, they cannot create it.
- The goal is to build programs that can make informed decisions on new unseen data



Example of application of machine learning to extract rules for a simple reflex agent.



Programming vs Machine Lerning

Machine Learning Applications

- Computer vision and robotics
- Speech recognition
- Biology and medicine
- Finance
- Information retrieval, Web search, ...
- Entertainment and Videogames
- Space exploration
- Education
- …

Suppose we have the experience we collected encoded as a dataset

$D = x_1, \ldots, x_N$

Supervised Learning

Given a set of desired outputs $y_1, \ldots, y_N$ it learns to produce the correct output for a new set of unseen data points.

- Desired output (labels, numbers)
- Direct feedback
- Predict future/outcome

Unsupervised Learning

Exploits regularities in D to build a representation for reasoning or prediction.

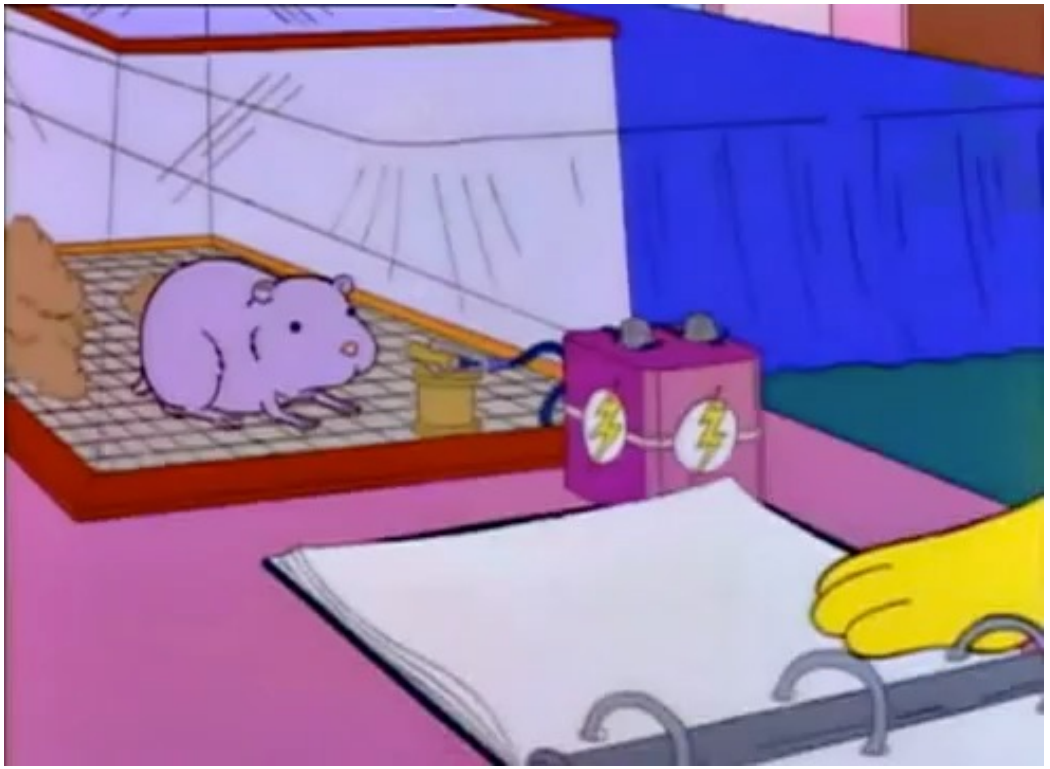
- No desired output to predict
- No feedback about predictions
- "find hidden structure"

Reinforcement Learning

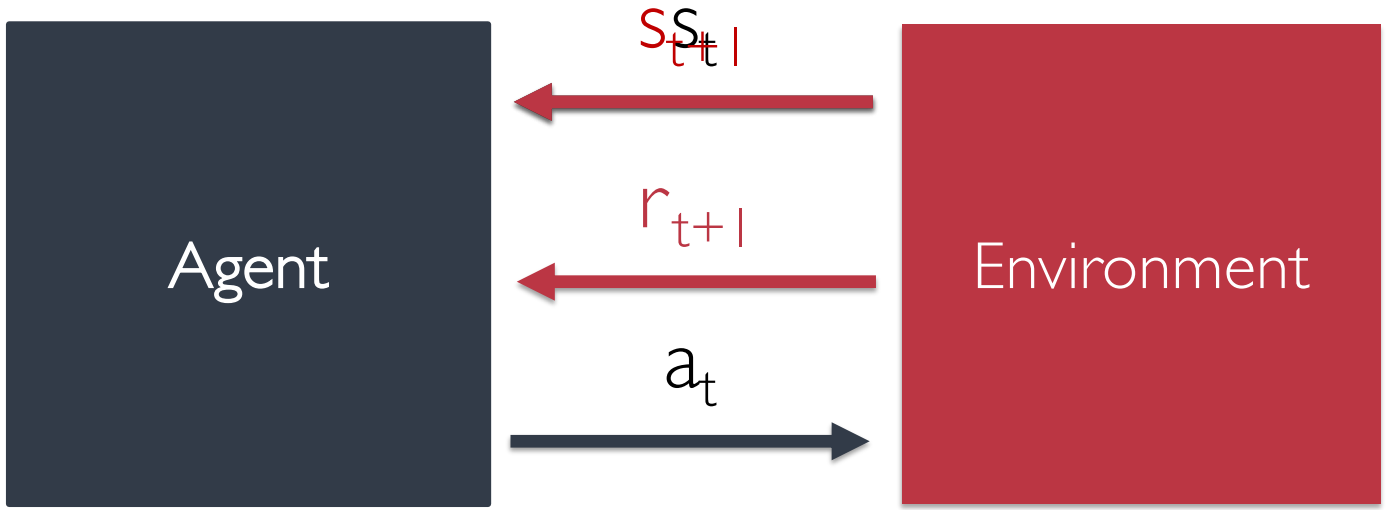
It performs actions $a_1, \ldots, a_N$ that affect the environment, and receiving rewards $r_1, \ldots, r_N$ it learn to maximize its long- term reward

- Decision process (actions)
- Reward system (immediate)
- Long term expectation
- Learn sequence of actions

reinforcement learning



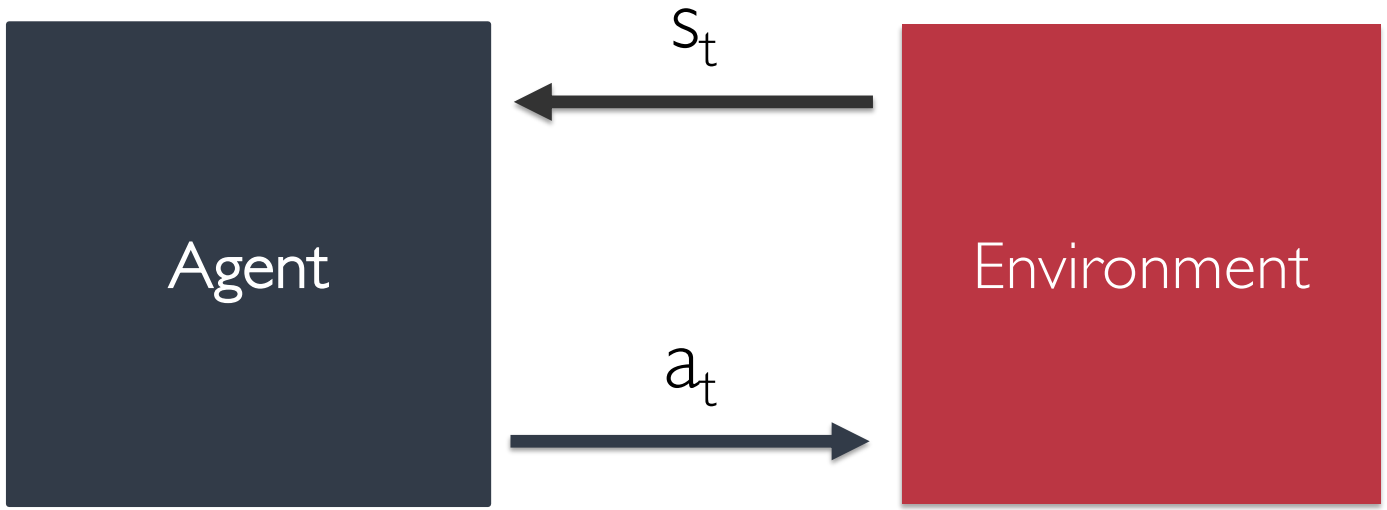
The Simpsons Season S4E16 – Duffles
https://www.imdb.com/title/tt0763029/



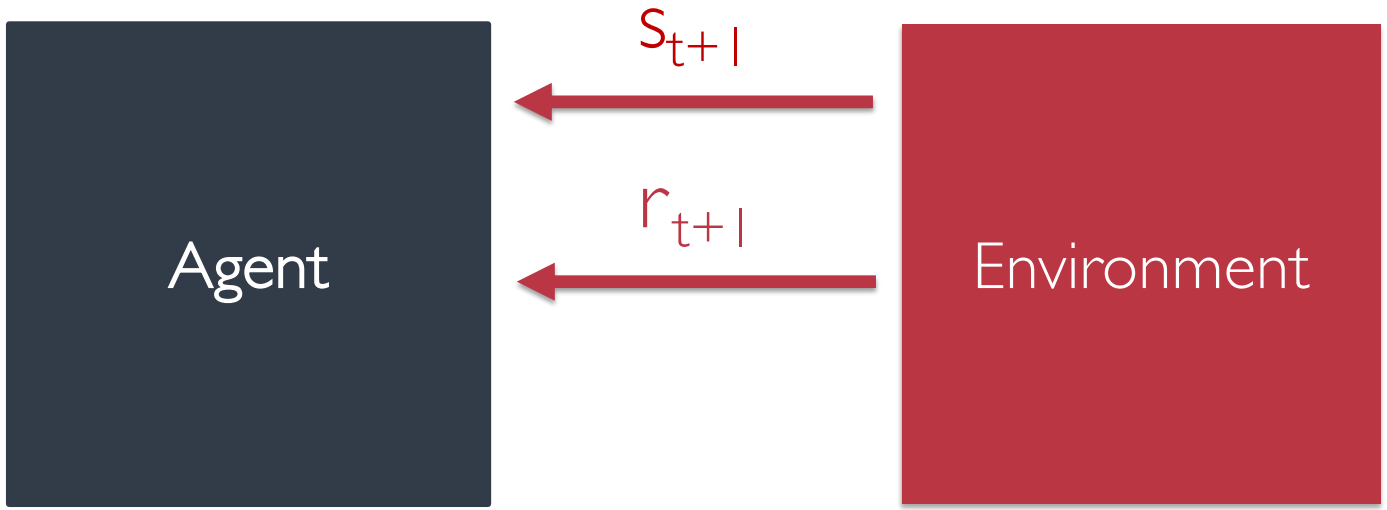
At time t, the agent perceives the environment to be in state s_t and decides to perform action a_t as a result in the next time step t+1 the environment state changes to s_{t+1} and the agent receives a reward r_{t+1}

In Sequential Decision Making …

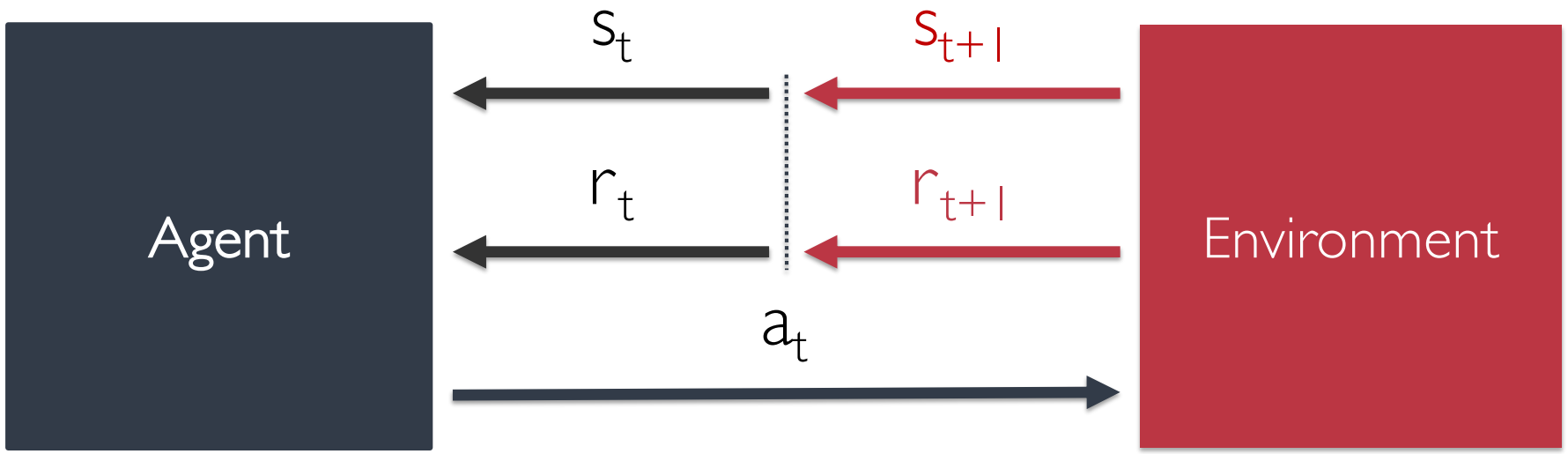
- … we take a sequence of decisions (or actions) to reach the goal
- … the optimal actions are context-dependent
- … we generally do not have examples of correct actions
- … actions may have long-term consequences
- … short-term consequences of optimal actions might seem negative

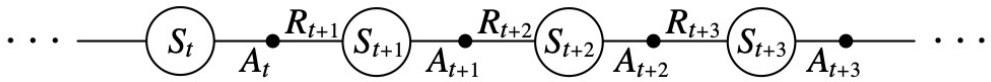


At time t, the agent perceives the environment to be in state s_t and decides to perform action a_t



As a result, in the next time step t+1 the environment state changes to s_{t+1} and the agent receives a reward r_{t+1}





Agent-environment interaction in reinforcement learning.

The agent's goal is to maximize
the total amount of reward received

How much future reward will it get when it performs a_t
in s_t and then continues to do its best from there on?

What is the expected payoff from s_t and a_t?

The agent needs to compute an action-value function mapping state-action pairs to expected payoffs

$$Q(s_t, a_t) \rightarrow \text{payoff}$$

or a state-value functions mapping to expected payoffs

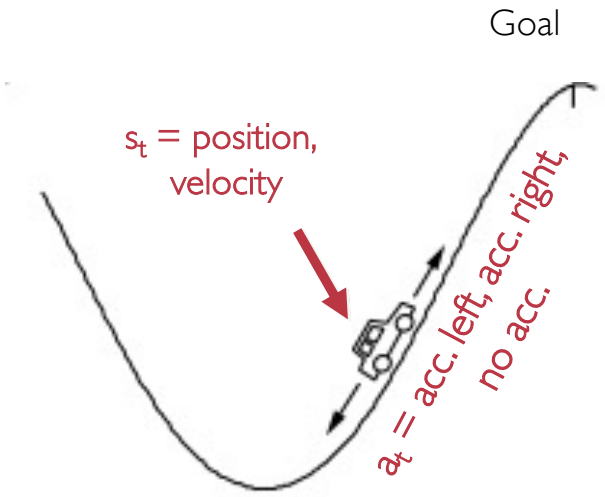
$$V(s_t) \rightarrow \text{payoff}$$

Reinforcement learning assumes
that $Q(s_t,a_t)$ is represented as a table

But the real world is complex,
the number of possible inputs can be huge!

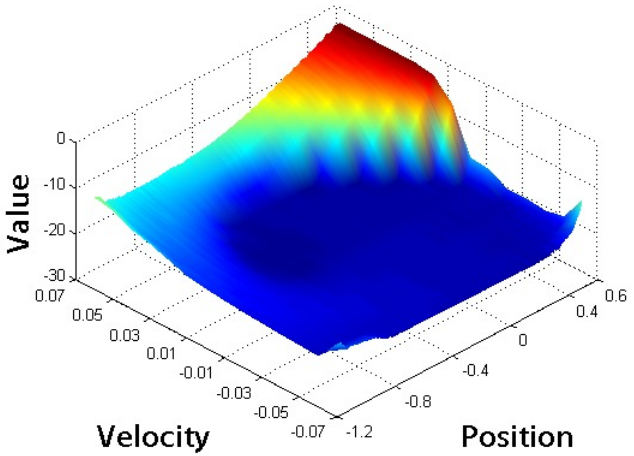
We cannot afford to compute an exact $Q(s_t,a_t)$
(more about this later)

r_t = 0 when goal is reached,
-1 otherwise



Task: drive an underpowered car up a steep mountain road

Value Function
$V(s_t)$



action selection

Action Selection & Policy

- At each time step, the agent must decide what action to take in step $\mathbf{t}$ based on its current evaluation of the expected payoff in $\mathbf{s_t}$ using a policy function
- At any given point in time, a policy $\boldsymbol{\pi(st)}$ select what actions the agent should perform
- The policy defines the behavior of an agent based on its payoff evaluation
- The policy can be deterministic or stochastic

Deterministic & Stochastic Policies

- Deterministic policy
  - In the simplest case the policy can be modeled as a function $\pi: S \rightarrow A$
  - For example, the policy might simply select the action with the largest expected payoff
  - This type of policy can be conveniently represented using a table

- Stochastic policy
  - It maps each state to a probability distribution over the actions $\pi: S, A \rightarrow R$
  - $\pi(s, a)$ returns the probability of selecting a in s
  - Since $\pi(s, a)$ is a probability distribution, it always return value greater or equal to zero and the sum over all the actions is 1
  - A stochastic policy can be used to represent also a deterministic policy

Exploration-Exploitation Dilemma

- To obtain a lot of reward, the agent must prefer actions that it has tried in the past and found to lead to high payoff
- However, to discover such actions, it has to try actions that it has not selected before
- The agent needs to find a trade-off between the exploration of new actions and the exploitation promising actions
- This is called exploration-exploitation dilemma

Greedy Policy

For each state, it deterministically selects an action with maximal value

ε-Greedy Policy

With probability ε it performs a random action, with probability
1 - ε it performs the action promising the highest payoff

the environment

The Environment

- The environment must satisfy the Markov property.
- The next state s_{t+1} and reward r_{t+1} depend only on the current state s_t and action a_t
- The environment can thus be modeled as a Markov Decision Process (MDP) that has a one-step dynamic described by the probability distribution $p(s_{t+1}, r_{t+1} \mid s_t, a_t)$
  - $p: S \times R \times S \times A \rightarrow [0,1]$
  - $\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1 \quad \forall s \in A \; \forall a \in A(s)$

expected payoff

Expected Payoff in Continuing and Episodic Tasks

- In reinforcement learning, the agent has to maximize the reward it receives in the long run

$$G_t \doteq r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_{t+k} + \cdots \stackrel{?}{=} \infty$$

- To provide an upper bound to the payoff, we introduce a discount the future rewards by a factor γ $(0 < \gamma < 1)$:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{k-1} R_{t+k} + \cdots < \infty$$

- Thus, the expected reward to maximize will be defined as:

$$\mathbb{E}[G_t] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] \leq R_{max} \frac{1}{1-\gamma}$$

The reward hypothesys

"That all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)." (Rich Sutton)

http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html

Goal and Rewards

- In reinforcement learning, the agent learns how to maximize the expected future payoff.
- We must design a reward function that adequately represents our learning goal
- Examples
  - Goal-reward representation returns 1 when the goal is reached, 0 otherwise
  - Action-penalty representation returns -1 when the state is not the goal, 0 once the goal is reached
- Challenges to reward hypothesis
  - How to represent risk-sensitive behavior?
  - How to capture diversity in behavior?

the value function

The action-value function $Q(s_t, a_t)$ estimates the expected future payoff when performing action a_t in state s_t

The state-value function $V(s_t)$ estimates the expected future payoff starting from s_t(in the former case)

They can be both decomposed as the sum of the immediate reward received r_{t+1} and the future rewards

Bellman Expectation Equation

- The state–value function can again be **decomposed** into immediate reward plus discounted value of successor state:

$$V(s) = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1}) | s_t = s]$$

- The action-value function can be similarly decomposed:

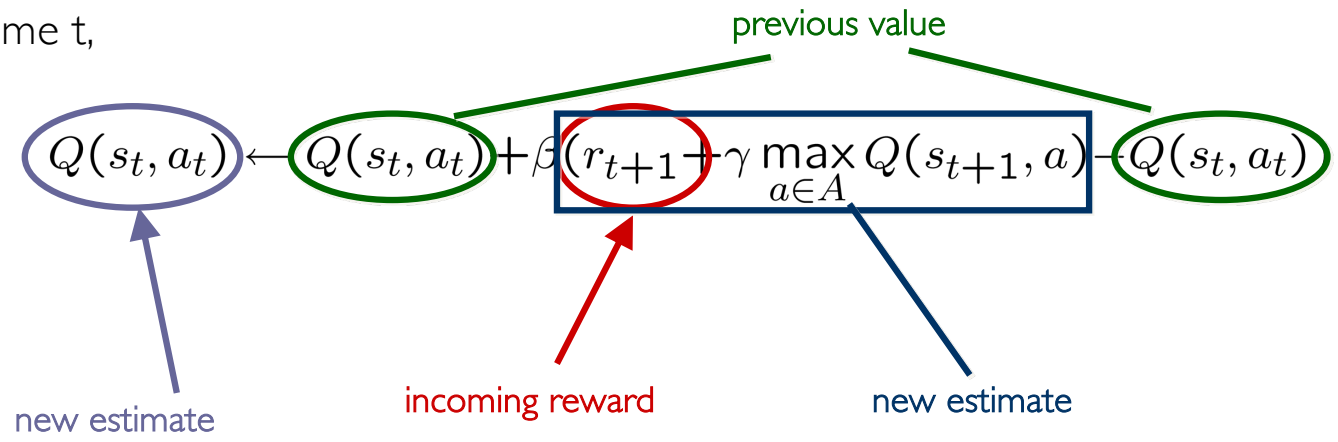
$$Q(s, a) = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1}) | s_t = s, a_t = a]$$

incoming reward

value of the next state

Q-learning

- At the beginning, the table $Q(\cdot, \cdot)$ is initialized with random values
- At time t,



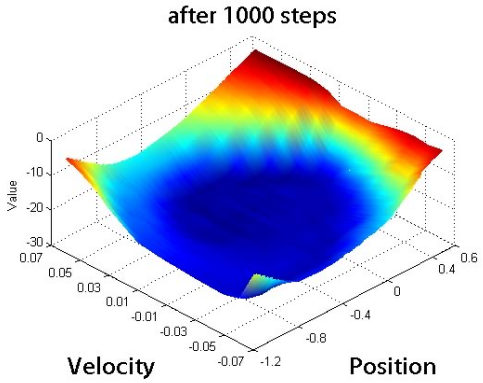
- The parameters are the discount factor γ, the learning rate β, the action selection strategy

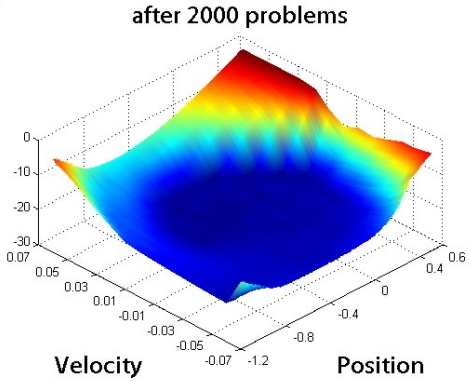
Q-learning

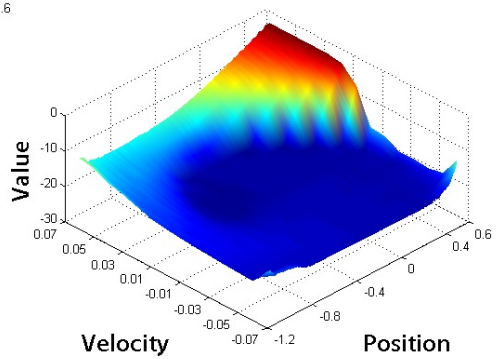
- At the beginning, the table $Q(\cdot, \cdot)$ is initialized with random values
- At time t,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \beta(r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t))$$

- Parameters
  - Discount factor γ
  - Learning rate β
  - Action selection strategy $\pi(s_t, a_t)$; ε-Greedy is the most common choice during learning but sufficient exploration must be guarantee to tackle the exploration-exploitation dilemma







Exact representation infeasible

Approximation mandatory

The function is unknown, it is learnt online from experience

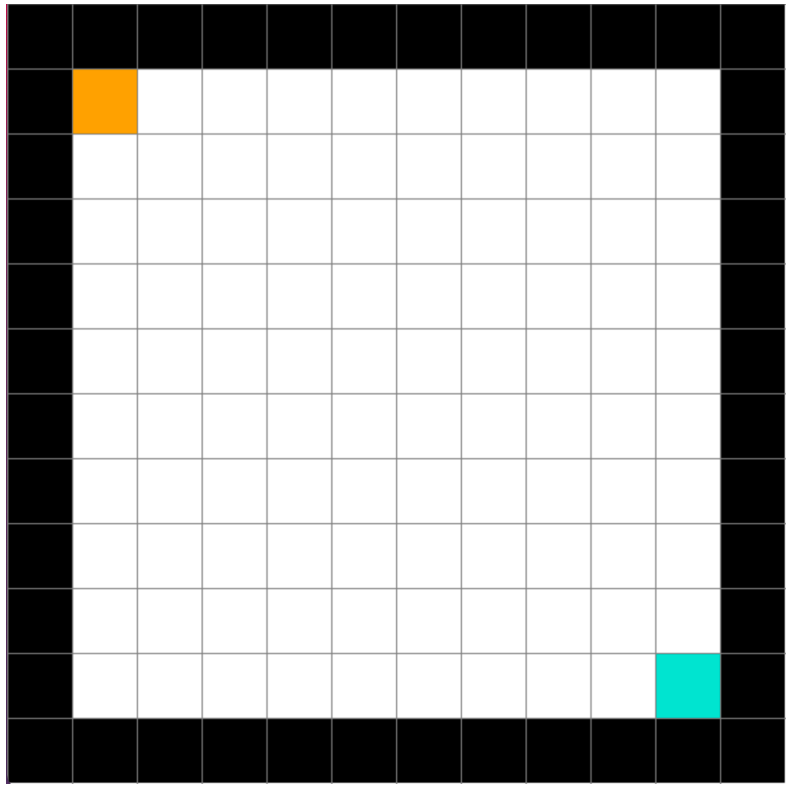
Tabular representation is infeasible in practice and approximators must be used for interesting problems

Reinforcement learning computes an unknown value function while also trying to approximate it

Approximator works on intermediate estimates
While also providing information for the learning

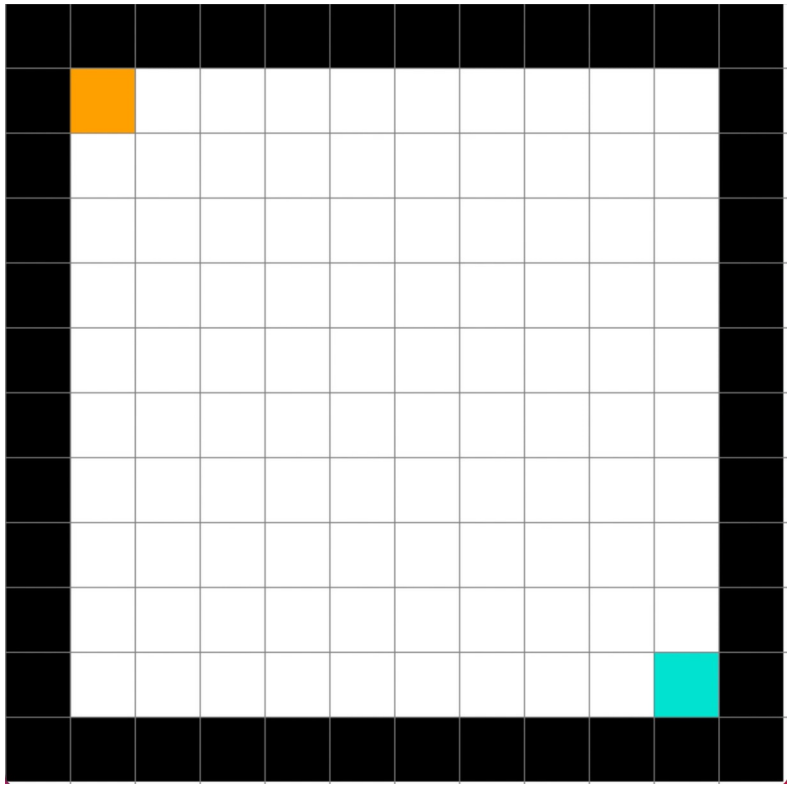
Convergence is not guaranteed

search? learning?



Simple empty environment with one start position (yellow) and one goal position (blue)

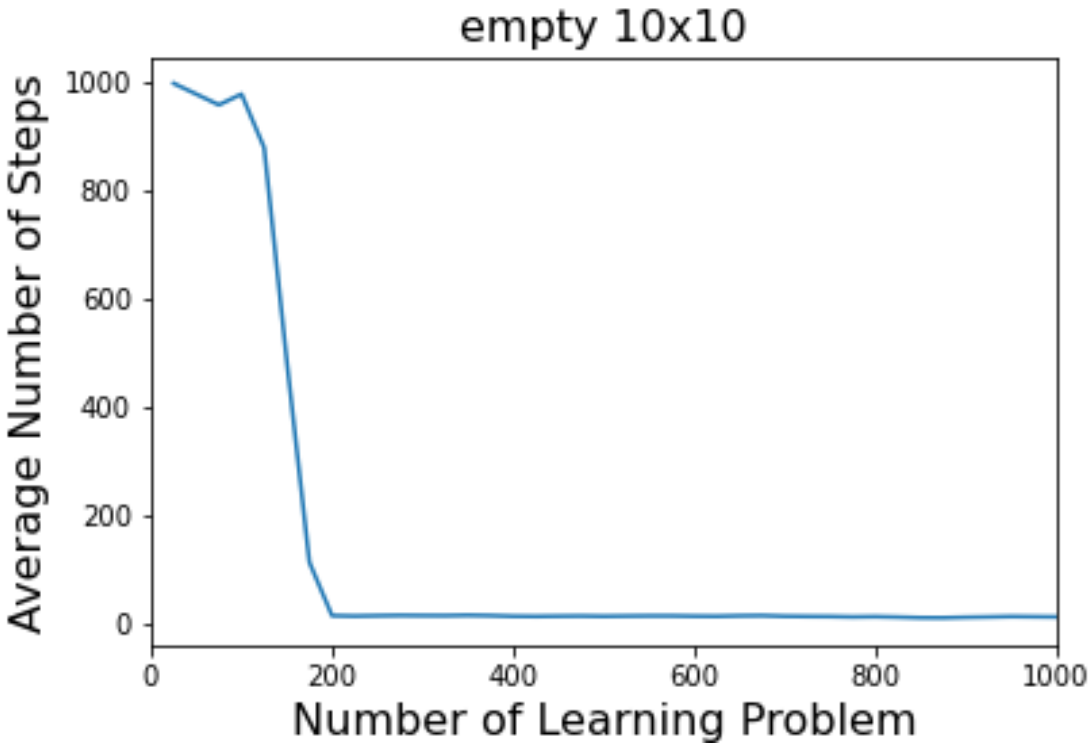
solution using a search algorithm



when applying reinforcement learning
we need to select a reward function

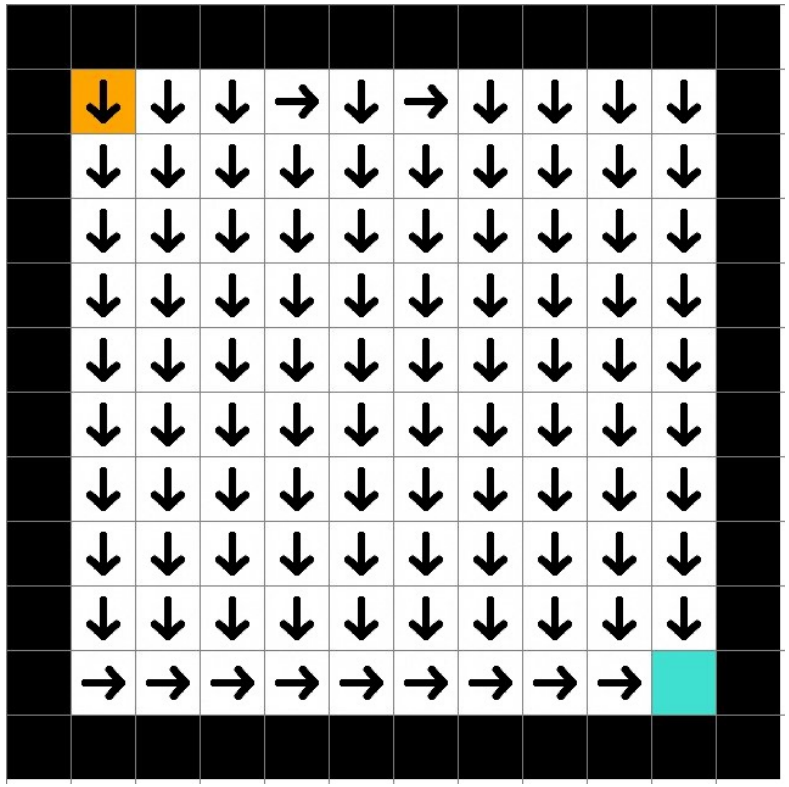
let's keep it simple, zero everywhere,
except when we reach the goal

when we reach the goal, we receive one



Let's measure the performance as the average number of steps in the last 100 problems.

search finds a solution, reinforcement learning an action value function

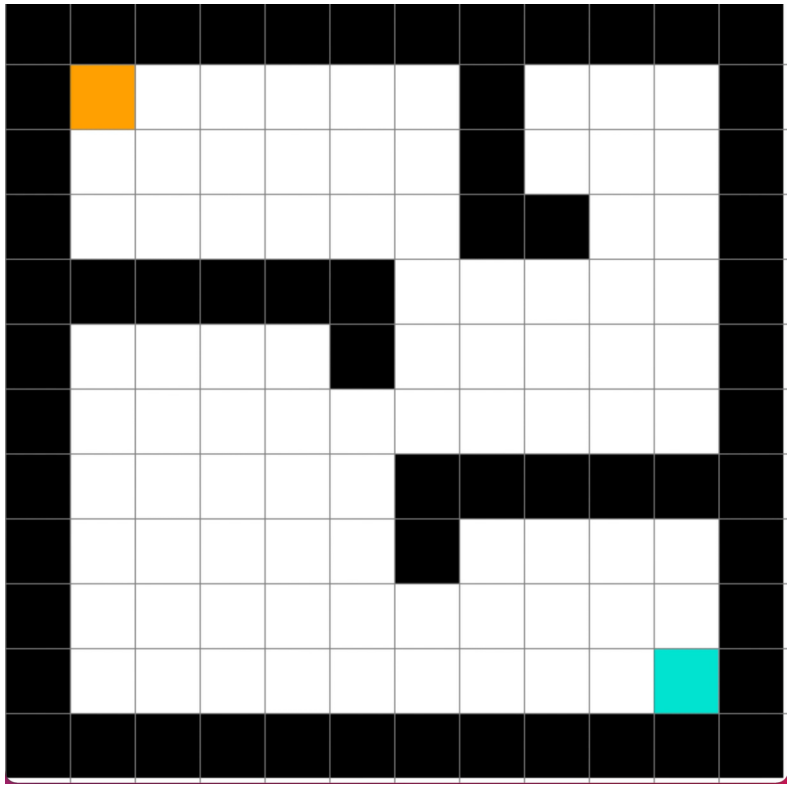


The best action for every position based on the action-value function.

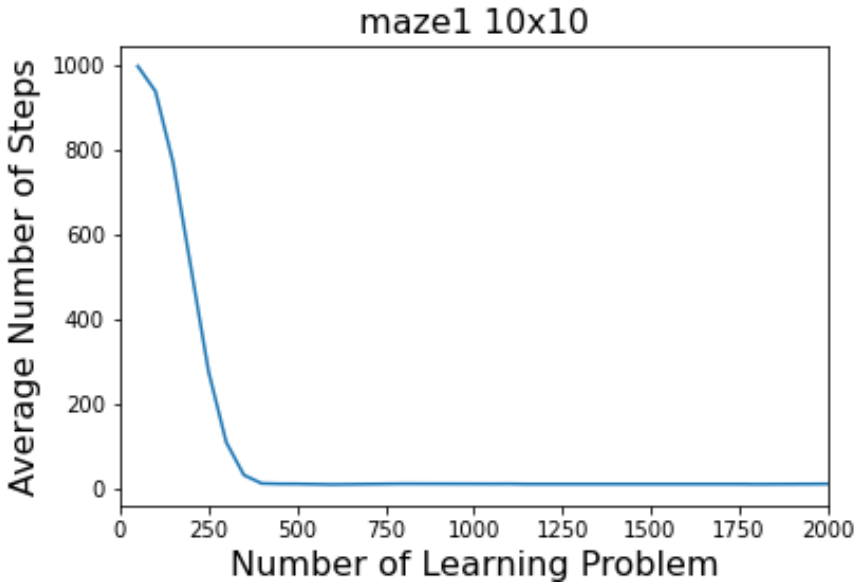
State	Up	Right	Down	Left
(1, 2)	0.397	0.440	0.440	0.000
(1, 3)	0.418	0.463	0.463	0.000
(1, 4)	0.440	0.488	0.488	0.000
(1, 5)	0.463	0.513	0.513	0.000
(1, 6)	0.488	0.540	0.540	0.000
(1, 7)	0.513	0.569	0.569	0.000
(1, 8)	0.540	0.599	0.599	0.000
(1, 9)	0.569	0.630	0.630	0.000
(1, 10)	0.599	0.663	0.000	0.000
(2, 1)	0.000	0.440	0.440	0.397
(2, 2)	0.418	0.463	0.463	0.418
(2, 3)	0.440	0.488	0.488	0.440
(2, 4)	0.463	0.513	0.513	0.463
(2, 5)	0.488	0.540	0.540	0.488
(2, 6)	0.513	0.569	0.569	0.513
...	...	...	...	...

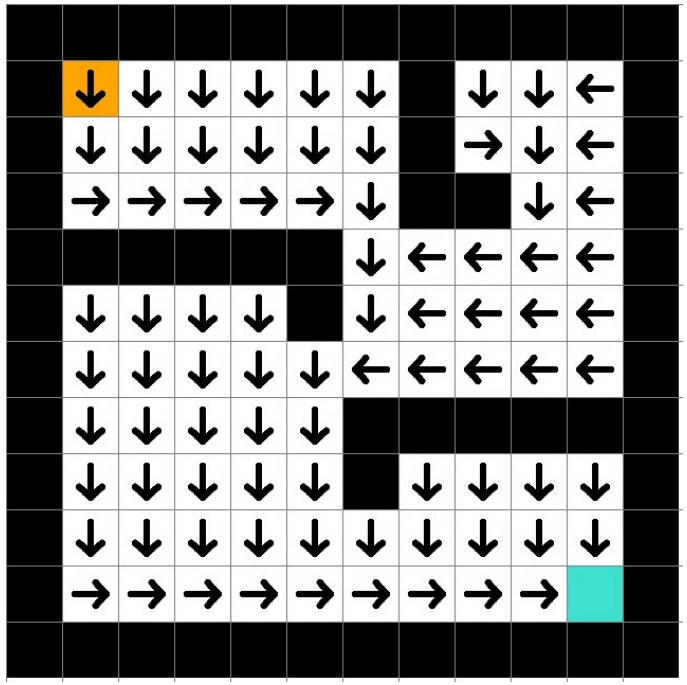
The computed action value function (first rows only).

another example



Solution using A*





Solution using reinforcement learning.

State	Up	Right	Down	Left
(1, 2)	0.358	0.397	0.397	0.000
(1, 3)	0.377	0.418	0.418	0.000
(1, 4)	0.397	0.440	0.440	0.000
(1, 5)	0.418	0.463	0.463	0.000
(1, 6)	0.440	0.488	0.000	0.000
(1, 8)	0.000	0.397	0.397	0.000
(1, 9)	0.377	0.418	0.377	0.000
(1, 10)	0.397	0.397	0.000	0.000
(2, 1)	0.000	0.397	0.397	0.358
(2, 2)	0.377	0.418	0.418	0.377
(2, 3)	0.397	0.440	0.440	0.397
(2, 4)	0.418	0.463	0.463	0.418
(2, 5)	0.440	0.488	0.488	0.440
(2, 6)	0.463	0.513	0.000	0.463
(2, 8)	0.000	0.000	0.418	0.377
...	...	...	...	...

The computed action value function (first rows only).

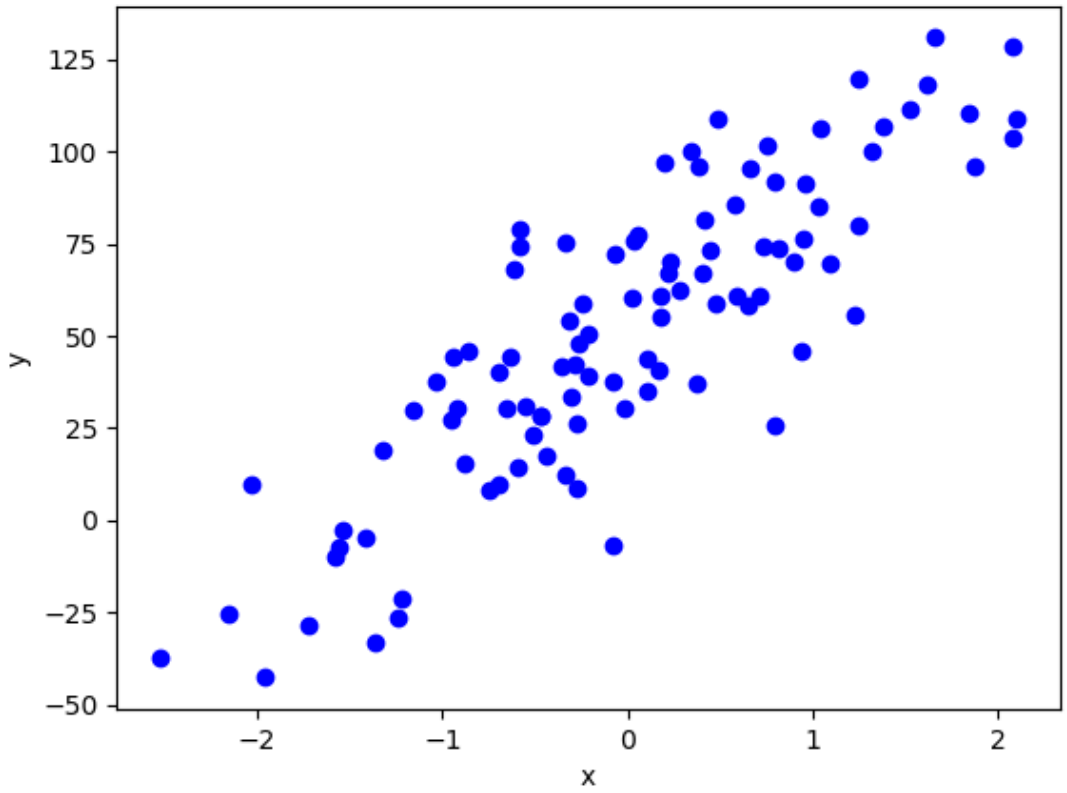
supervised learning

Supervised Learning

Given a set of inputs-output pairs $(x_1, y_1)\ldots, (x_N, y_N)$ it learns to produce the correct output for a new set of unseen data points.

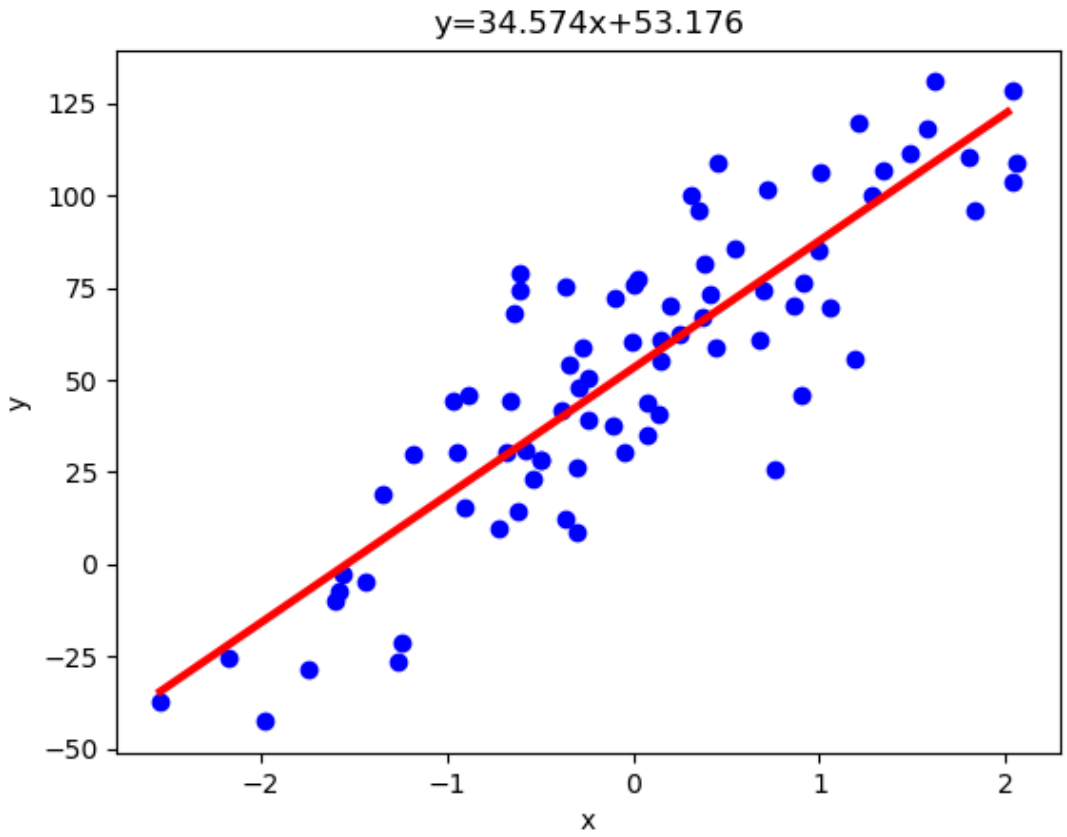
When the target values are real values, we have a regression problem

When they are discrete or symbolic, we have a classification problem



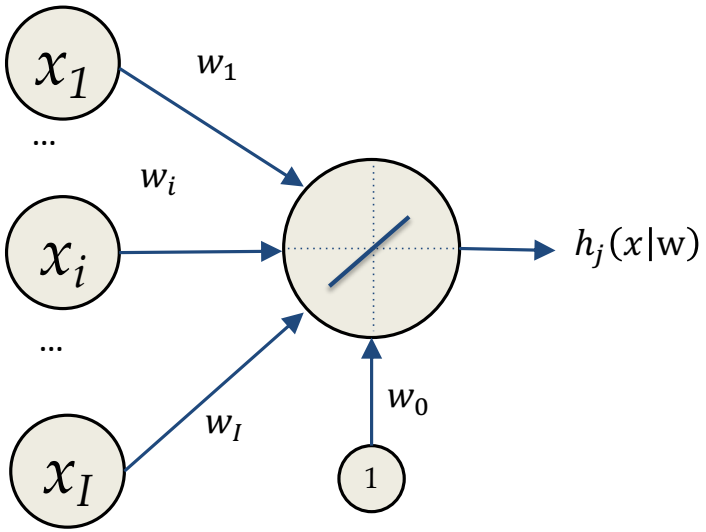
(x,y) inputs-output pairs

We can use a simple linear model



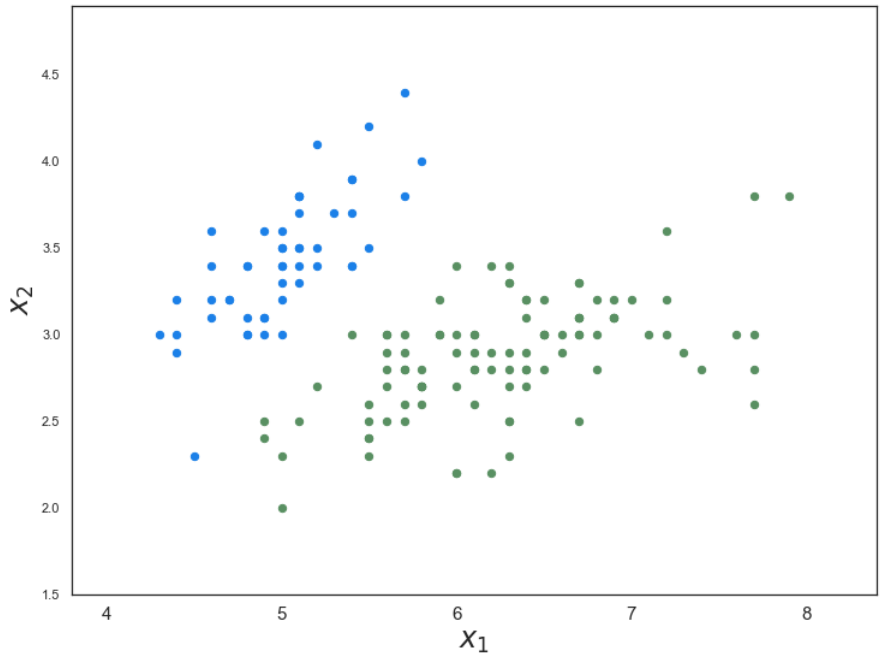
Example data fitted with a simple linear regression model

How can we compute a regression model?

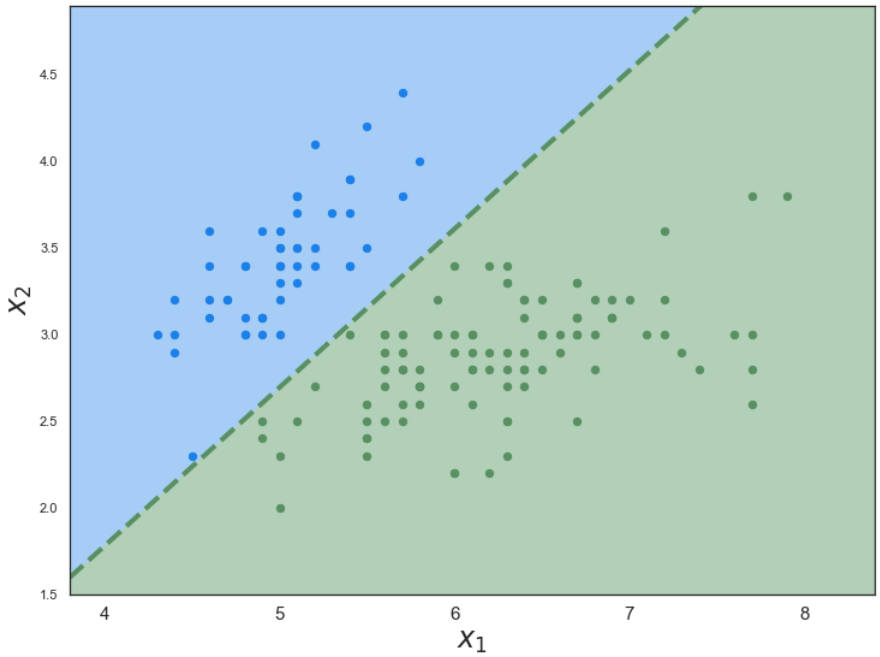


$$h_j(x|\mathrm{w},\mathrm{b}) = \mathrm{h_j}\left(\Sigma_{i=1}^{I} w_i \cdot x_i - b\right) = \mathrm{h_j}\left(\Sigma_{i=0}^{I} w_i \cdot x_i\right) = (w^T x)$$

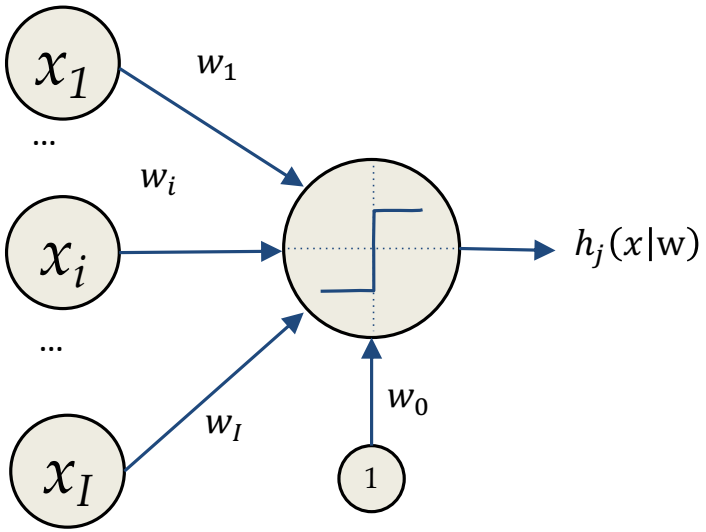
Computation in an artificial neurons



(x,y) inputs-output pairs with x is bidimensional vector identified by the two coordinates x_1 and x_2 while y identifies the points' color (blue or green).



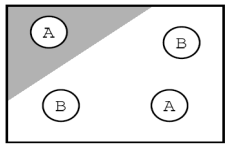
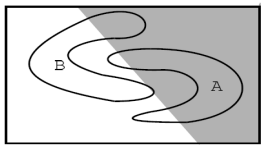
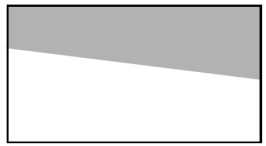
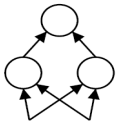
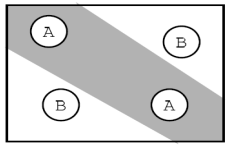
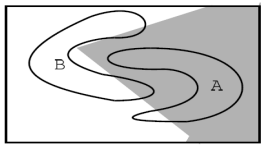
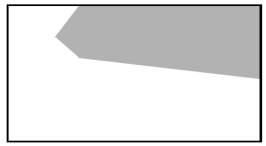
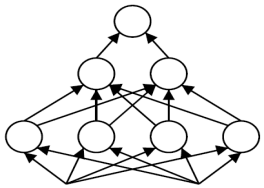
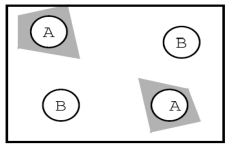
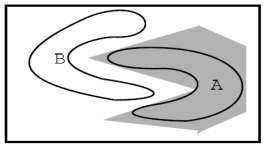
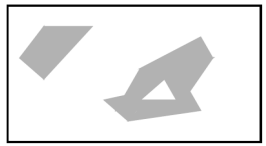
Classification model.



$$h_j(x|\mathrm{w},\mathrm{b}) = \mathrm{h_j}\left(\Sigma_{i=1}^{I} w_i \cdot x_i - b\right) = \mathrm{h_j}\left(\Sigma_{i=0}^{I} w_i \cdot x_i\right) = h_j(w^T x)$$

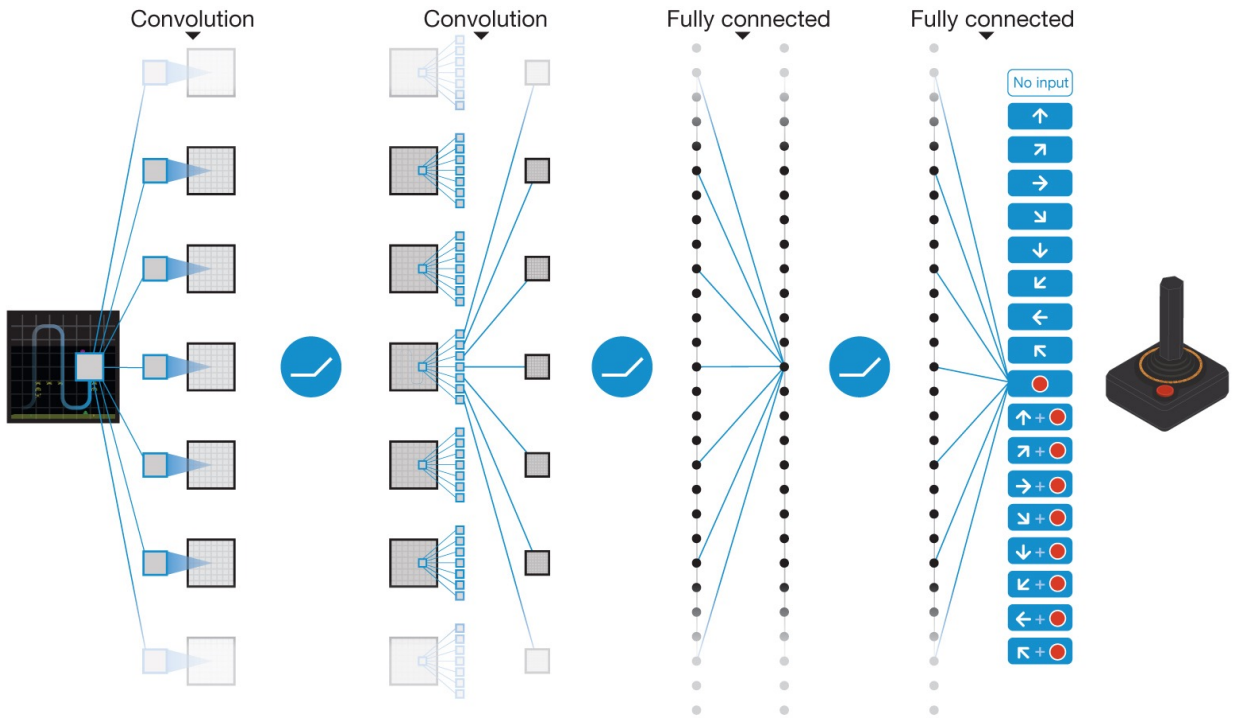
Computation in an artificial neurons for binary classification.

We Can Combine Simple Neurons to Do More …

Topology	Type of Decision Region	XOR Problem	Classes with Meshed Regions	Most General Region Shapes
	Half bounded by hyperplanes			
	Convex Open or Clesed Regions			
	Arbitrary Regions (Complexity limited by the number of nodes)			

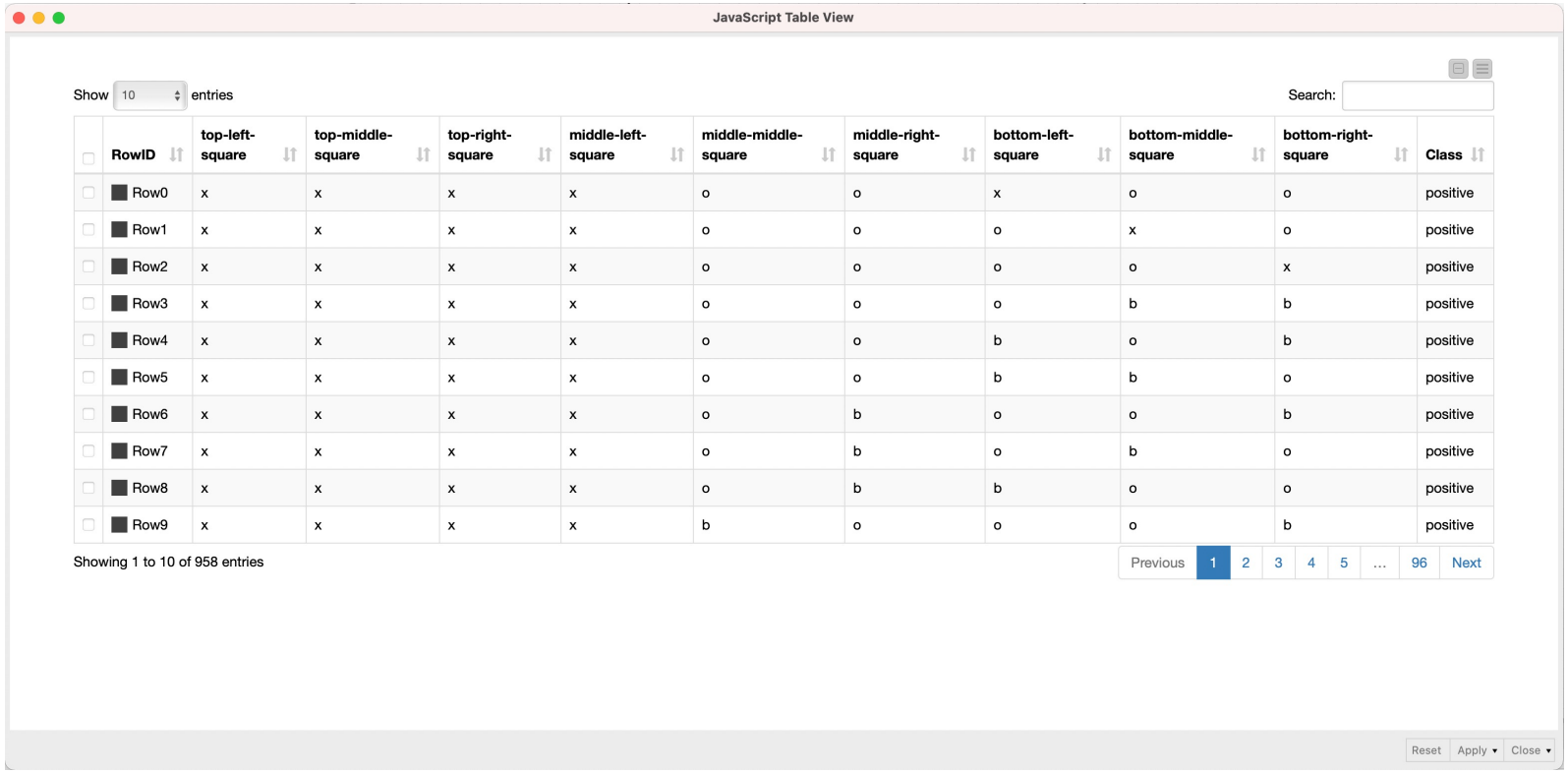
For instance, you can use a complex network to compute the action-value function for a very complex task

V Mnih *et al. Nature* **518**, 529-533 (2015) doi:10.1038/nature14236
https://www.nature.com/articles/nature14236



V Mnih *et al. Nature* **518**, 529-533 (2015) doi:10.1038/nature14236
https://www.nature.com/articles/nature14236

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/img/kites_detections_output.jpg



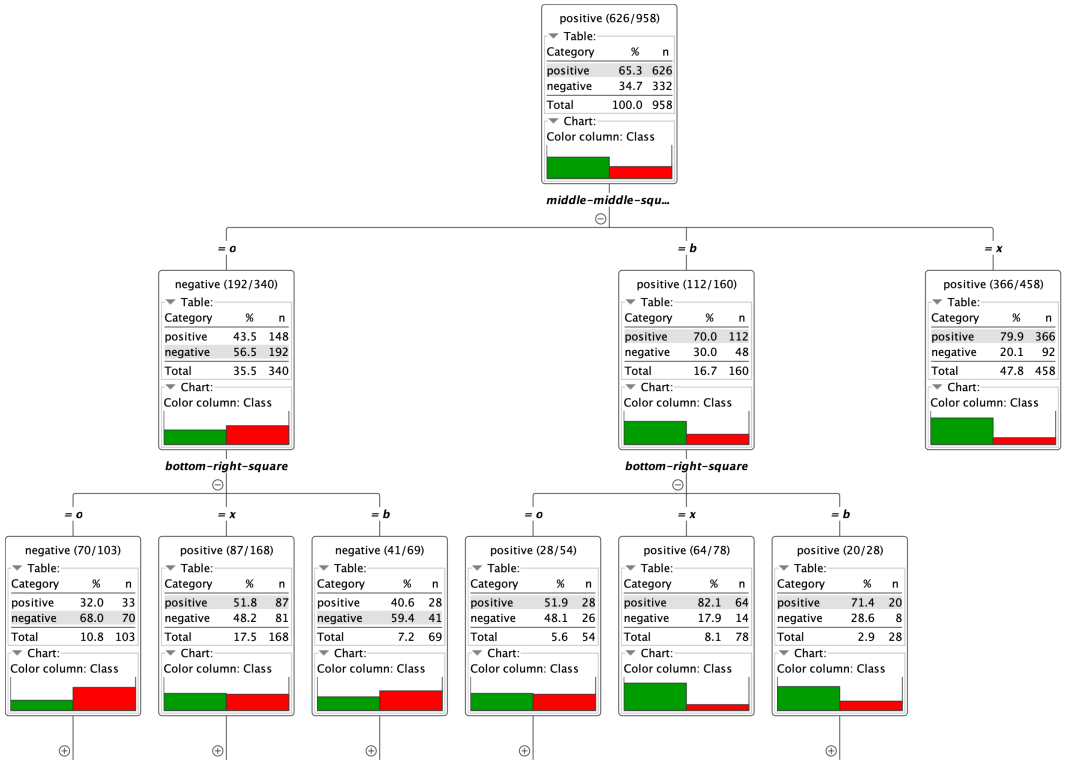
Tic-tac-toe ending positions classified accordingly to the victory of player X (positive, X wins, negative it loses)

Row ID	Rule	Record...	Numbe...
Row1	$top-left-square$ = "x" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "o" => "positive"	48	33
Row2	$top-left-square$ = "o" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "o" => "negative"	50	50
Row3	$top-left-square$ = "b" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "o" => "negative"	5	5
Row4	$bottom-middle-square$ = "o" AND $bottom-left-square$ = "x" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	24	18
Row5	$bottom-middle-square$ = "x" AND $bottom-left-square$ = "x" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "positive"	38	38
Row6	$bottom-middle-square$ = "b" AND $bottom-left-square$ = "x" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	13	7
Row7	$top-right-square$ = "x" AND $bottom-left-square$ = "o" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "positive"	27	21
Row8	$top-right-square$ = "o" AND $bottom-left-square$ = "o" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	30	30
Row9	$top-right-square$ = "b" AND $bottom-left-square$ = "o" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	3	3
Row10	$middle-right-square$ = "o" AND $bottom-left-square$ = "b" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	11	10
Row11	$middle-right-square$ = "b" AND $bottom-left-square$ = "b" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	4	3
Row12	$middle-right-square$ = "x" AND $bottom-left-square$ = "b" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "positive"	18	14
Row13	$top-middle-square$ = "x" AND $bottom-middle-square$ = "o" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "positive"	10	9
Row14	$top-middle-square$ = "o" AND $bottom-middle-square$ = "o" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "negative"	7	7
Row15	$top-middle-square$ = "b" AND $bottom-middle-square$ = "o" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "positive"	3	3
Row16	$bottom-middle-square$ = "x" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "negative"	16	11
Row17	$bottom-middle-square$ = "b" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "positive"	16	11
Row18	$top-left-square$ = "o" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "negative"	5	5
Row19	$top-left-square$ = "b" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "negative"	12	12
Row20	$top-left-square$ = "x" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "b" => "positive"	42	28
Row21	$top-left-square$ = "o" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "b" => "negative"	4	4
Row22	$top-left-square$ = "b" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "b" => "negative"	8	8
Row23	$bottom-right-square$ = "x" AND $middle-middle-square$ = "b" => "positive"	78	64
Row24	$top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "b" => "positive"	20	20
Row25	$top-left-square$ = "o" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "b" => "negative"	8	8
Row26	$top-left-square$ = "b" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "b" => "positive"	0	0
Row27	$middle-middle-square$ = "x" AND TRUE => "positive"	458	366

Classification rules for the tic-tac-toe dataset.

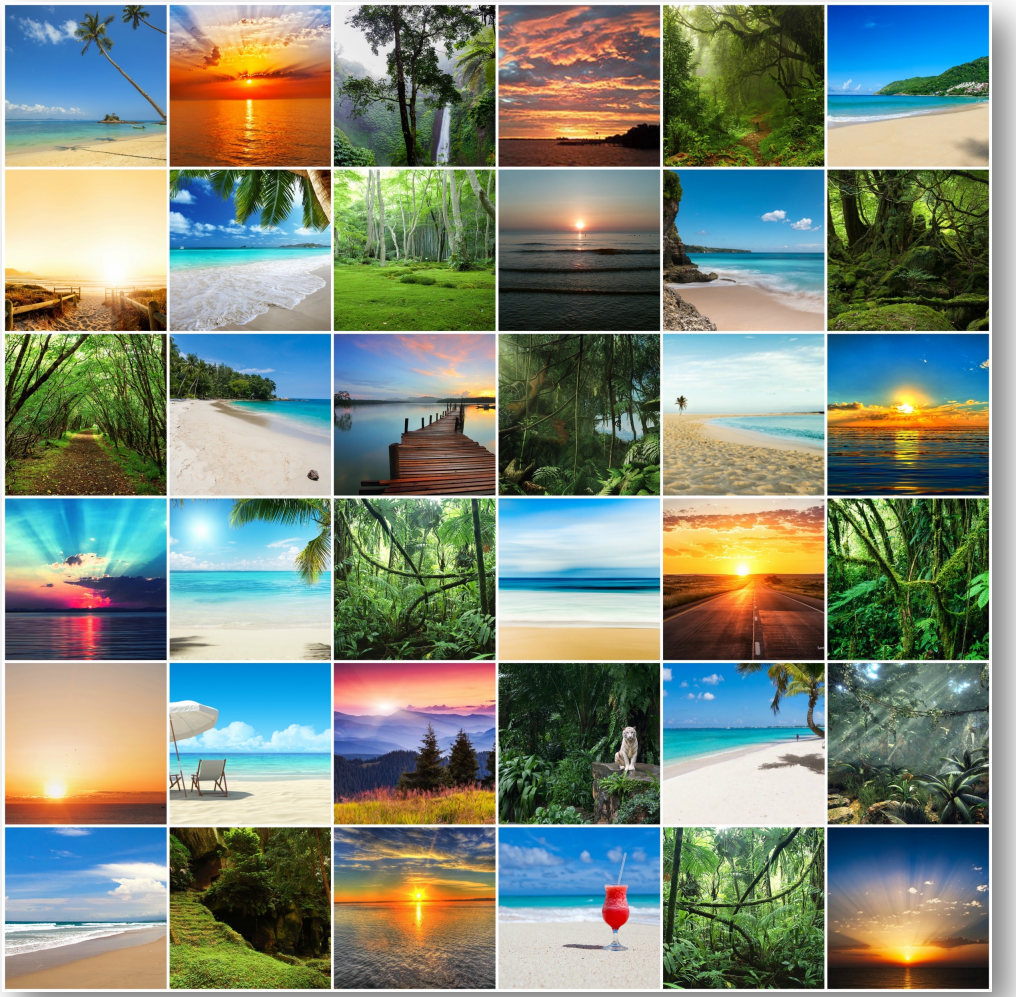
Row ID	Rule	Record...	Numbe...
Row1	$top-left-square$ = "x" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "o" => "positive"	48	33
Row2	$top-left-square$ = "o" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "o" => "negative"	50	50
Row3	$top-left-square$ = "b" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "o" => "negative"	5	5
Row4	$bottom-middle-square$ = "o" AND $bottom-left-square$ = "x" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	24	18
Row5	$bottom-middle-square$ = "x" AND $bottom-left-square$ = "x" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "positive"	38	38
Row6	$bottom-middle-square$ = "b" AND $bottom-left-square$ = "x" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	13	7
Row7	$top-right-square$ = "x" AND $bottom-left-square$ = "o" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "positive"	27	21
Row8	$top-right-square$ = "o" AND $bottom-left-square$ = "o" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	30	30
Row9	$top-right-square$ = "b" AND $bottom-left-square$ = "o" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	3	3
Row10	$middle-right-square$ = "o" AND $bottom-left-square$ = "b" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	11	10
Row11	$middle-right-square$ = "b" AND $bottom-left-square$ = "b" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "negative"	4	3
Row12	$middle-right-square$ = "x" AND $bottom-left-square$ = "b" AND $bottom-right-square$ = "x" AND $middle-middle-square$ = "o" => "positive"	18	14
Row13	$top-middle-square$ = "x" AND $bottom-middle-square$ = "o" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "positive"	10	9
Row14	$top-middle-square$ = "o" AND $bottom-middle-square$ = "o" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "negative"	7	7
Row15	$top-middle-square$ = "b" AND $bottom-middle-square$ = "o" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "positive"	3	3
Row16	$bottom-middle-square$ = "x" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "negative"	16	11
Row17	$bottom-middle-square$ = "b" AND $top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "positive"	16	11
Row18	$top-left-square$ = "o" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "negative"	5	5
Row19	$top-left-square$ = "b" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "o" => "negative"	12	12
Row20	$top-left-square$ = "x" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "b" => "positive"	42	28
Row21	$top-left-square$ = "o" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "b" => "negative"	4	4
Row22	$top-left-square$ = "b" AND $bottom-right-square$ = "o" AND $middle-middle-square$ = "b" => "negative"	8	8
Row23	$bottom-right-square$ = "x" AND $middle-middle-square$ = "b" => "positive"	78	64
Row24	$top-left-square$ = "x" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "b" => "positive"	20	20
Row25	$top-left-square$ = "o" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "b" => "negative"	8	8
Row26	$top-left-square$ = "b" AND $bottom-right-square$ = "b" AND $middle-middle-square$ = "b" => "positive"	0	0
Row27	$middle-middle-square$ = "x" AND TRUE => "positive"	458	366

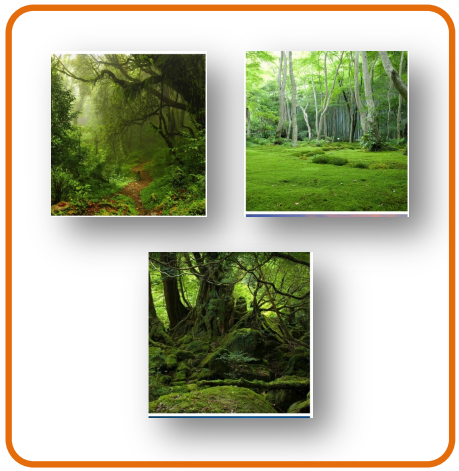
Classification rules for the tic-tac-toe dataset.

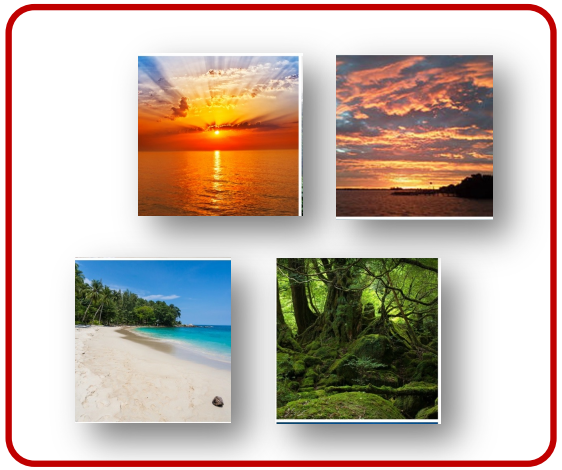


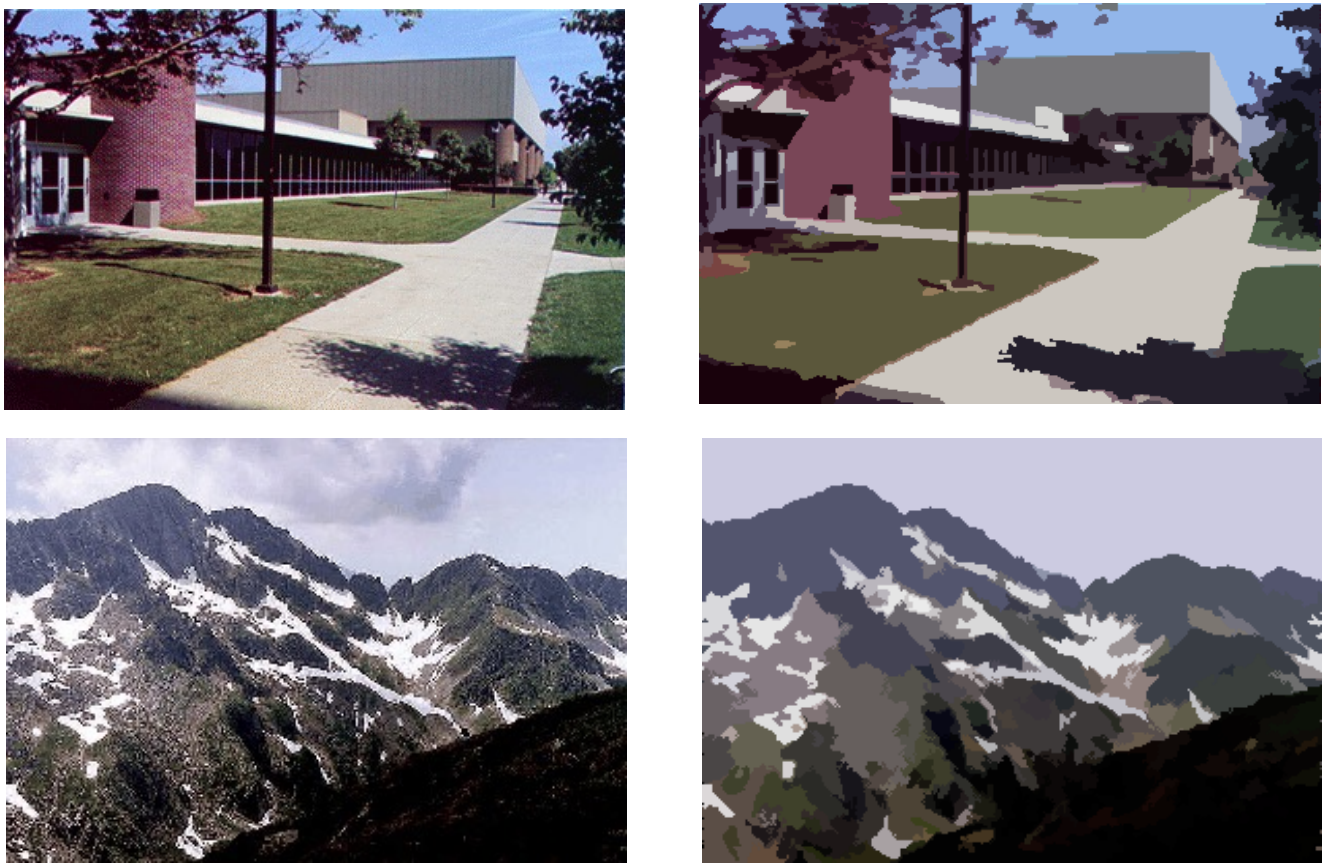
Decision tree model for the same problem.

unsupervised learning

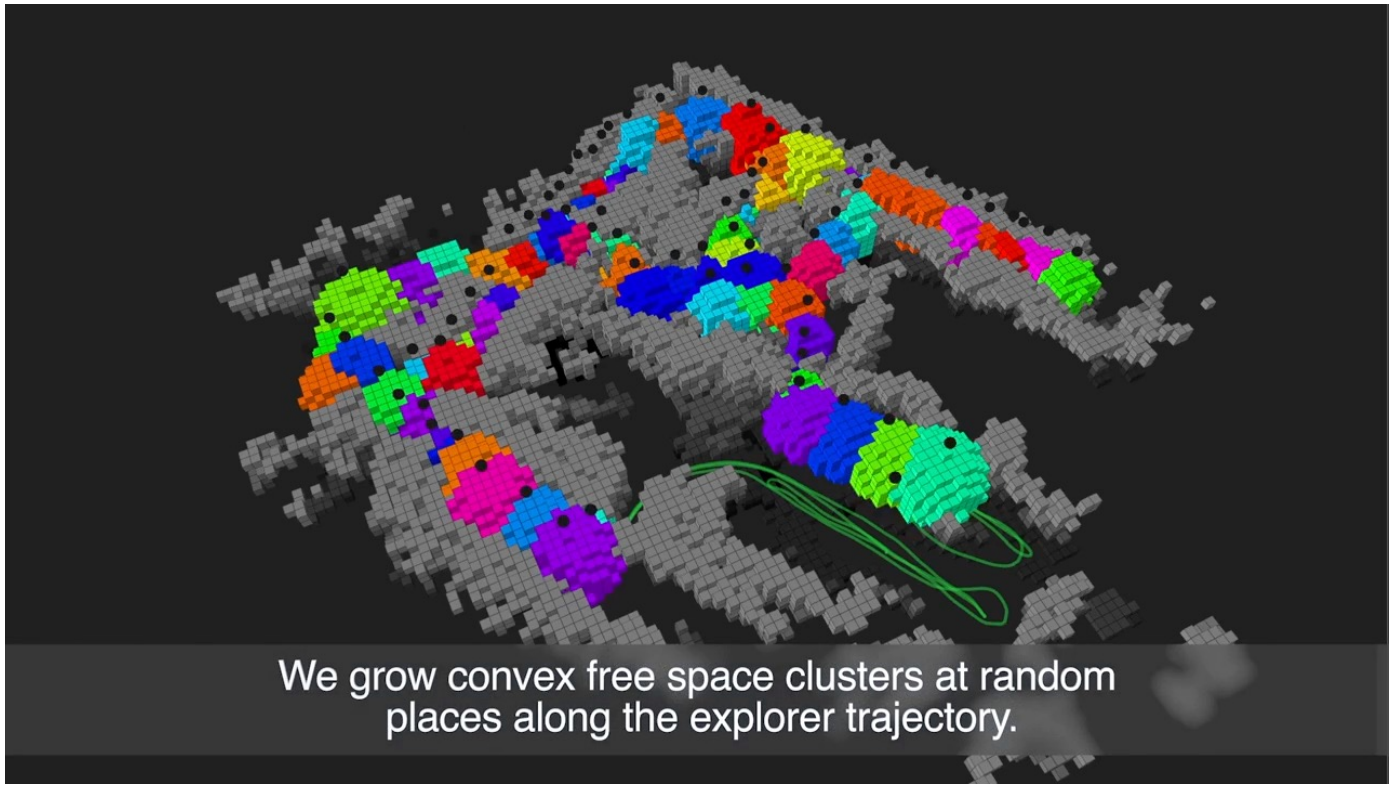






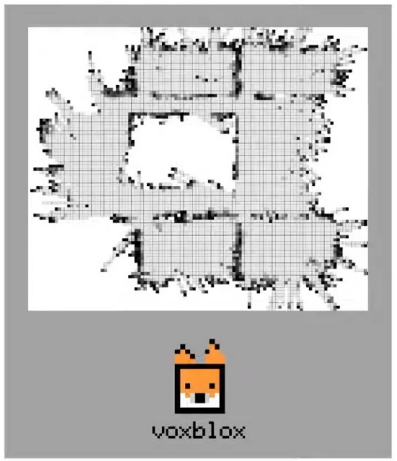


D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

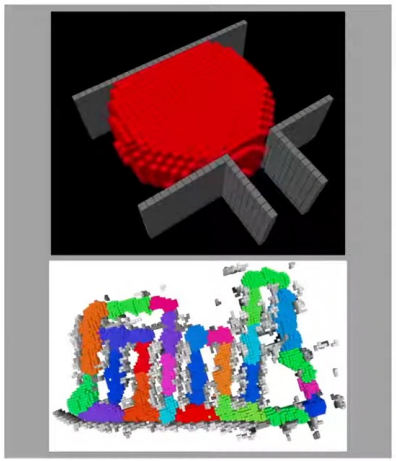


https://www.youtube.com/watch?v=UokjxSLTcd0

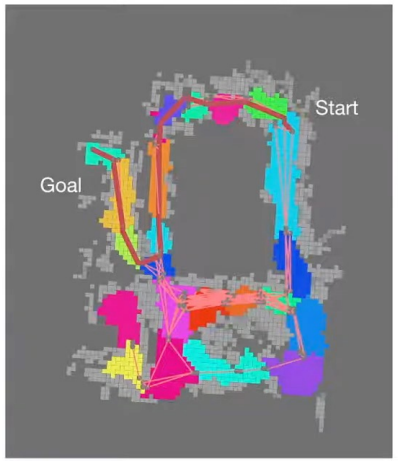
Free space from landmarks



Clustering algorithm

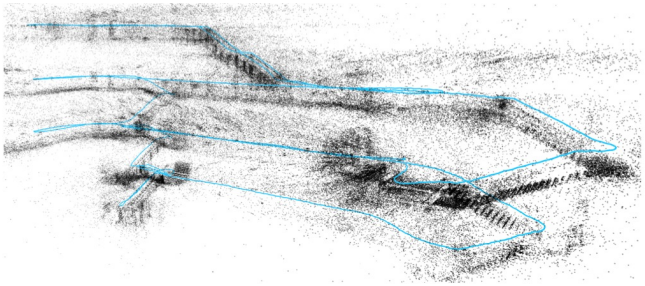


Path Planning

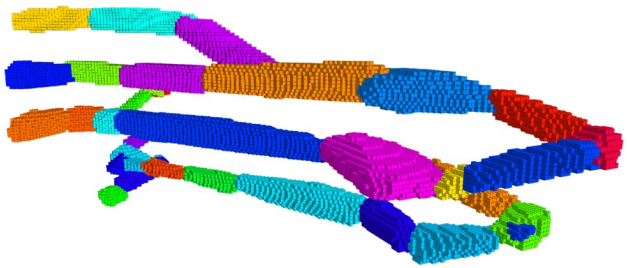


 Wed AM Pod U.2

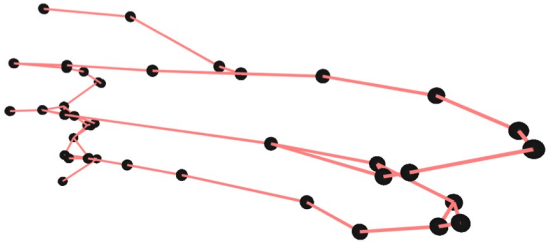
https://www.youtube.com/watch?v=BJwxyuTFg-g



(a) Visual SLAM map.



(b) Topological map (convex voxel clusters).



(c) Navigation graph.

Fabian Blöchliger, Marius Fehr, Marcin Dymczyk, Thomas Schneider, Roland Siegwart.
Topomap: Topological Mapping and Navigation Based on Visual SLAM Maps. https://arxiv.org/abs/1709.05533