Before start…

- Primo appello sessione estiva: 16/06
- Secondo appello sessione estiva: 01/07
- Terzo appello sessione estiva: 21/07

Prossima settimana soltanto facciamo lezione mercoledì (con Prof. Anselmi)

Introduction to Reinforcement Learning

Outline

- Introduction
  - What is reinforcement learning?
  - Why do we need it?
  - How to?
- Basics of RL
  - Action vs. reward
  - State vs. value
  - Policy
  - Model

Outline

- **Introduction**
  - **What is reinforcement learning?**
  - **Why do we need it?**
  - **How to?**
- Basics of RL
  - Action vs. reward
  - State vs. value
  - Policy
  - Model

Reinforcement learning

Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal.

- Sutton & Barto, 2020

***Reinforcement learning** (**RL**) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward.*

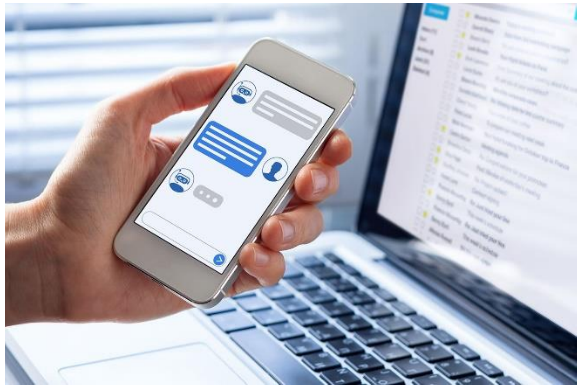
- Wikipedia

Reinforcement learning

- It is a family of problems
  - Sequential decision making

Game playing

Self-driving car



Conversational System

Reinforcement learning

- A typical (narrow) view of the problem formulation

Markov decision process,
to be covered

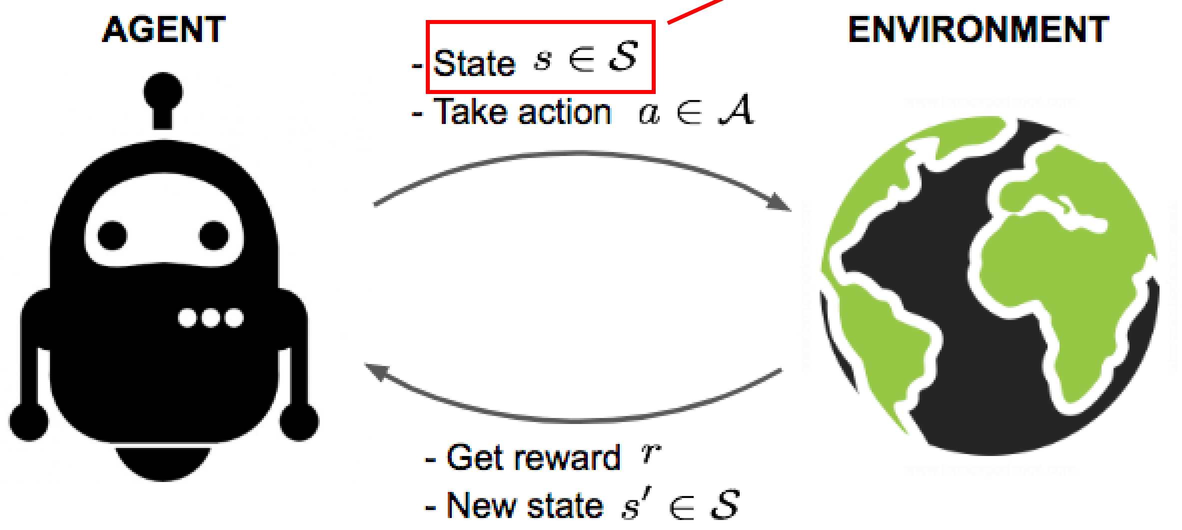
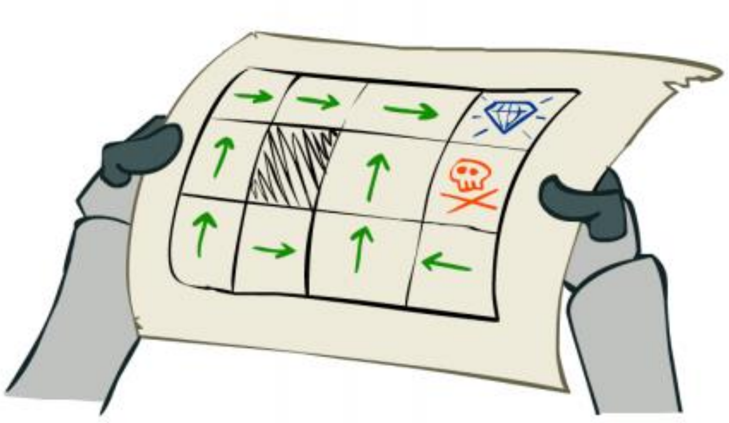


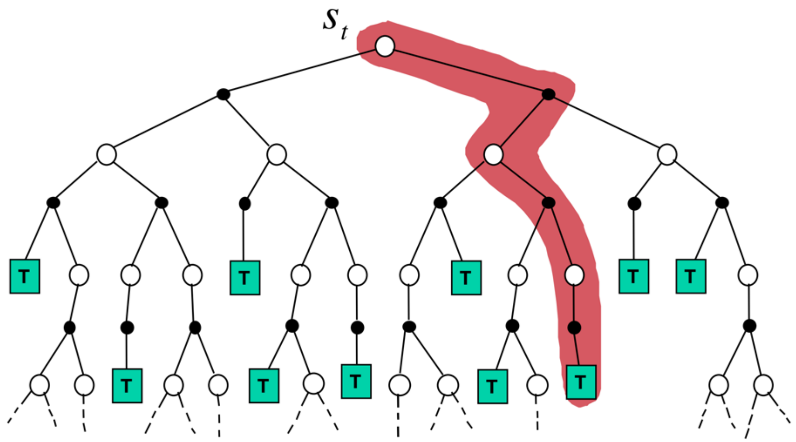
Image credit: Lil'Log

Reinforcement learning

- It is a family of solutions
  - Taking a series of actions to maximum cumulative return



Planning



~~Planning while~~ learning
Reinforcement

Image credit: David Silver, "Model-Free Prediction"

Reinforcement learning

- It is a collection of fields that study such problems and solutions
  - Computer science, psychology, neuroscience, optimization, operations research, and many others

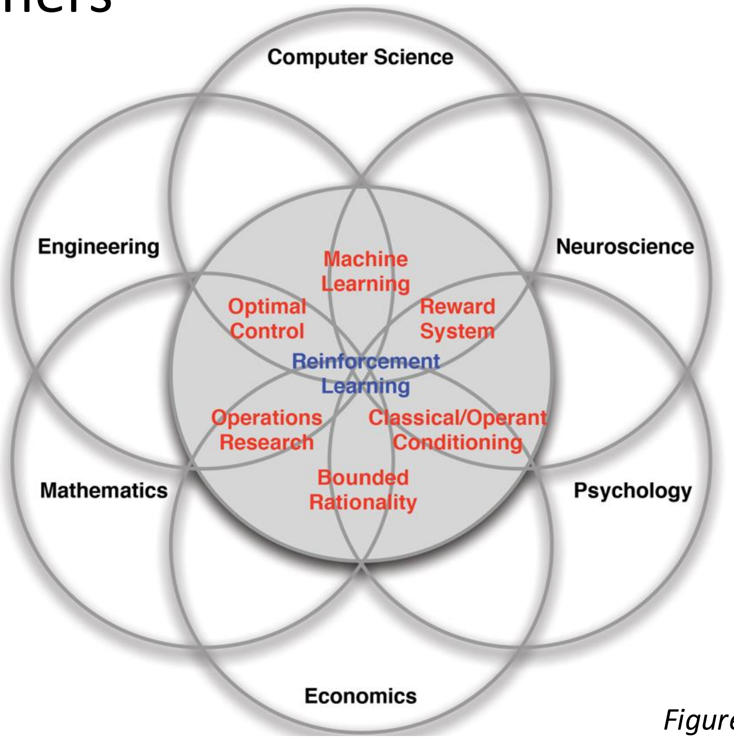


Figure credit: David Silver, "Introduction to RL"

Summary: reinforcement learning

- It is a family of problems
    - Sequential decision making
- It is a family of solutions
    - Planning and learning
- It is a collection of fields that study the problems and solutions

What characterizes Reinforcement Learning (vs other ML tasks)?

✓No supervisor: only a *reward* signal

✓Delayed asynchronous feedback

✓Time matters (sequential data, continual learning)

✓Agent's actions affect the subsequent data it receives (inherent non-stationarity)

Vs. Supervised machine learning

- Classification as an example
  - Training time
    - Input: $\boxed{\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N}$, where $\boldsymbol{x}_n \in R^d$ and $y_n \in [C]$
    - Output: hypothesis $f_\theta(\boldsymbol{x}) \rightarrow y$
    - Goal: $\min_{\theta \in \Theta} \sum_{n=1}^N L(f_\theta(\boldsymbol{x}_n), y_n)$
  - Testing time
    - Apply $f_\theta(\boldsymbol{x}) \rightarrow y$

Do we really have a choice here?

The only decision(s) to make

Are we making any decisions here?

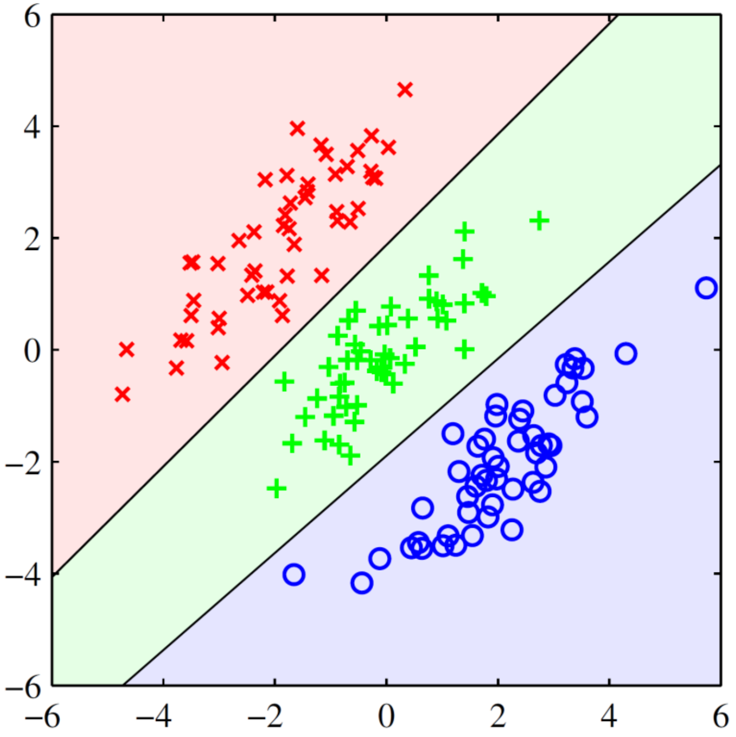


Image credit: Bishop, "Pattern Recognition and Machine Learning"

Vs. Supervised machine learning

Do we really have a choice here?

- <u>Online</u> classification as an example
  - The hypothesis will be immediately tested
    - Input: $\boxed{\{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}}$ where $\boldsymbol{x}_t \in R^d$ and $y_t \in [C]$ arrive <u>sequentially</u>
    - Output: hypothesis $f_{\theta_t}(\boldsymbol{x}) \rightarrow y$ and $\hat{y}$
    - Goal: $\sum_{t=1}^{T-1} \min_{\theta_t \in \Theta} L(f_{\theta_t}(\boldsymbol{x}_{t+1}), y_{n+1})$ ← *Cumulated over T*

Observe y_t & update f_{θ_t} — y_1: $\theta_1 = \text{argmin}_{\theta\in\Theta} L(\hat{y}_1, y_1)$

Predict $\hat{y}_t$ — $f_{\theta_0}(\boldsymbol{x}_1) \rightarrow \hat{y}_1$ $\quad f_{\theta_1}(\boldsymbol{x}_2) \rightarrow \hat{y}_2$ ......

Observe $\boldsymbol{x}_t$ — $\boldsymbol{x}_1$ $\quad \boldsymbol{x}_2$

$1 \qquad\qquad t \qquad\qquad T$

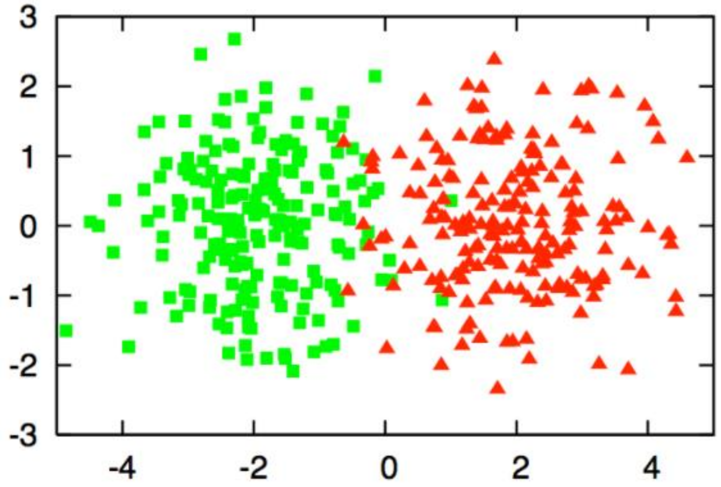
Now are we making sequential decisions to optimize some cumulative metric?

Vs. Semi-supervised machine learning

- Active learning for classification as an example
  - Training time
    - Input: $\{\boldsymbol{x}_n\}_{n=1}^{N}$, where $\boldsymbol{x} \in R^d$
    - Procedure: choose a subset of instances of size T to obtain their labels for model training
    - Output: hypothesis $f_{\theta_{\boxed{T}}}(\boldsymbol{x}) \rightarrow y$
    - Goal: $\boxed{\min E_{P(\boldsymbol{x},y)}[L(f_{\theta_T}(\boldsymbol{x}), y)]}$
  - Testing time
    - Apply $f_{\theta_T}(\boldsymbol{x}) \rightarrow y$

Budget: T queries

This makes us only need to explore!



Now our decisions affect our observations, right!?

Full v.s., partial information

- Full information – in most supervised ML
  - $(\boldsymbol{x}_t, y_t)$ is always given
- Partial information - bandit feedback
  - Only $L(f_\theta(\boldsymbol{x}_t), y_t)$ is provided

$(\boldsymbol{x}_t, \hat{y}_t$=Bear)
$(\boldsymbol{x}_t, \hat{y}_t$=Dog)

Exploitation v.s., exploration

- Unknown unknowns

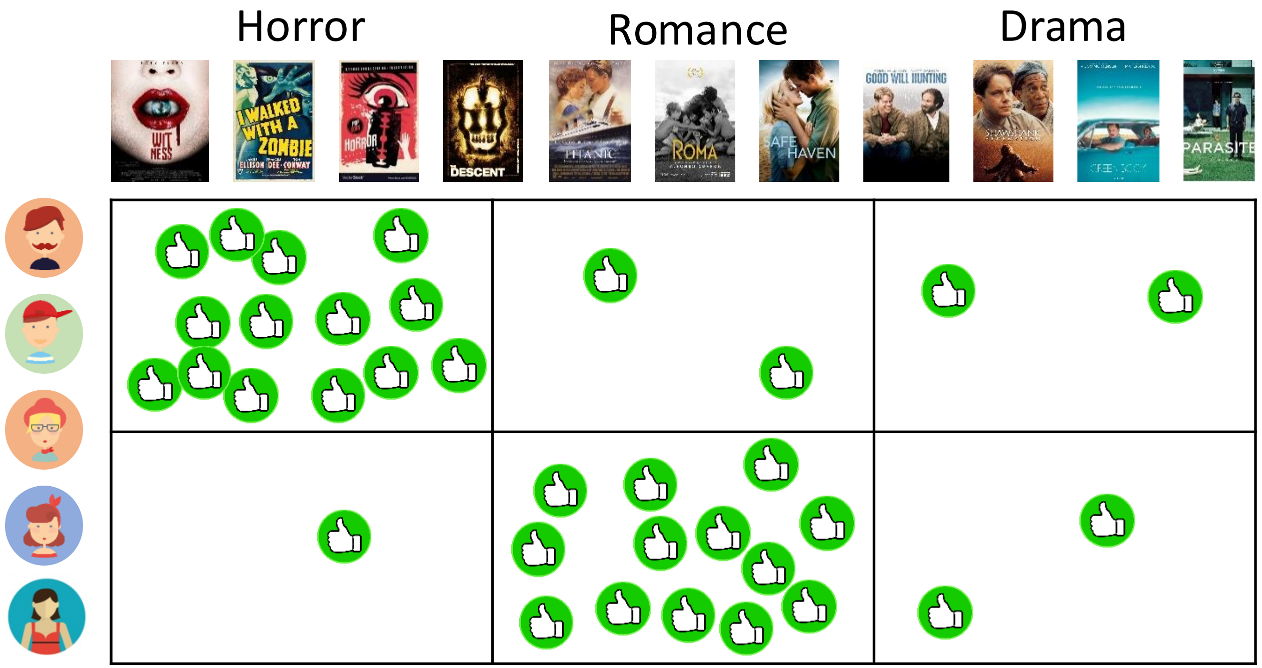


Figure credit: Schnabel et al. 2016 [SSSCJ16]

Matthew effect: we still don't know what we don't know!

A quick summary

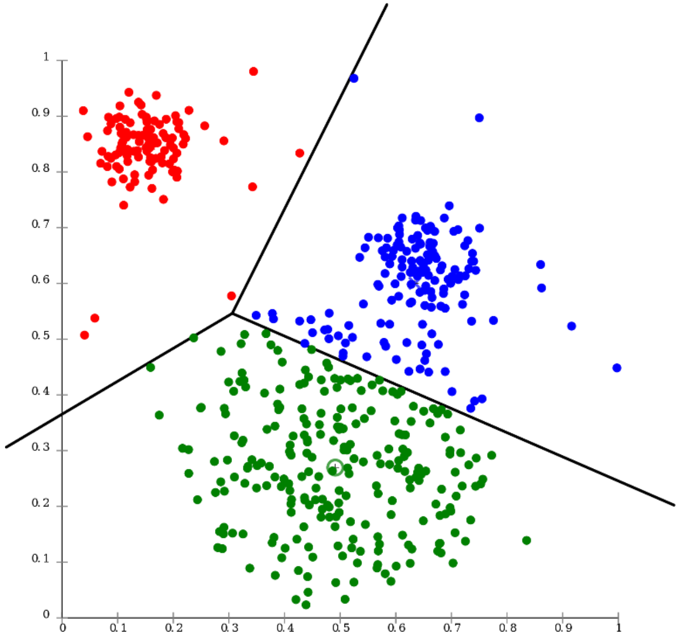
- Reinforcement learning
  - Reward
  - Partial information
  - Delayed consequence
  - Agent’s choice affects the subsequent data it receives, i.e., non-i.i.d.
- Supervised machine learning
  - Ground-truth labels
  - Full information
  - Immediate consequence
  - Pre-determined data distribution, i.e., i.i.d.

Vs. unsupervised learning

- Flat structure clustering as an example
  - Training time
    - Input: $\{\boldsymbol{x}_n\}_{n=1}^{N}$, where $\boldsymbol{x} \in R^d$
    - Output: hypothesis $g_\theta\,(\boldsymbol{x}) \rightarrow k$, where $k \in [K]$
    - Goal: $\min \sum_{k=1}^{K} \sum_{g(x_m)=g(x_n)=k} \boxed{L(\boldsymbol{x}_m, \boldsymbol{x}_n)}$
  - Testing time
    - Apply $g_\theta(\boldsymbol{x}) \rightarrow k$

Your clustering criterion

NO information about the clustering structure at all?!

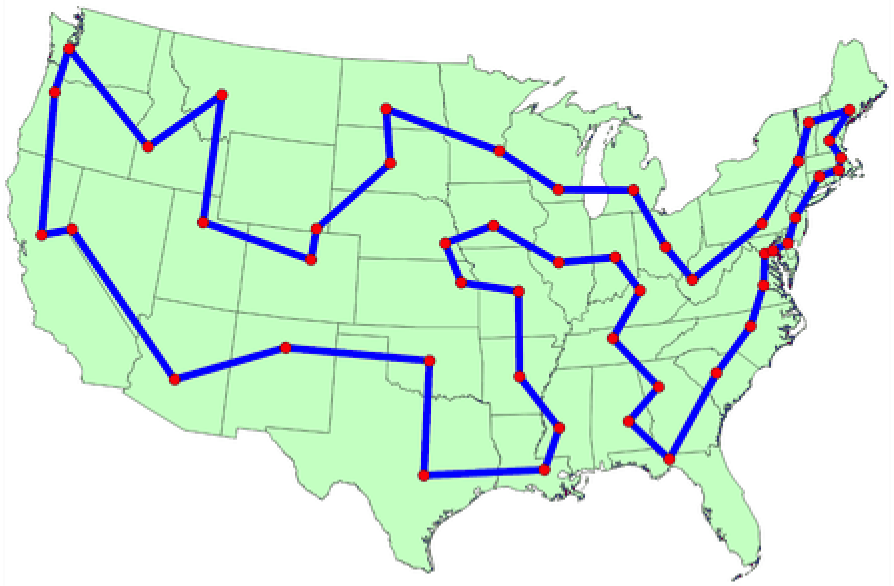


Source: https://en.wikipedia.org/wiki/Cluster_analysis

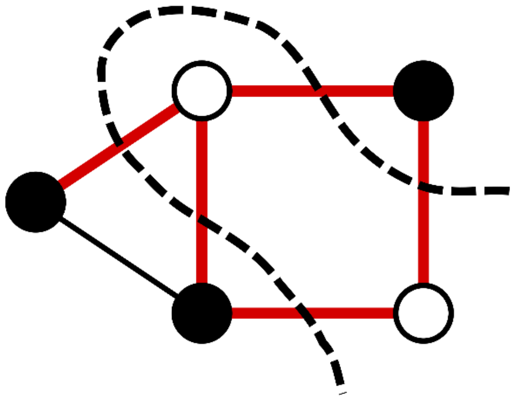
Vs. Combinatorial optimization

- Known environment model
  - A planning problem in RL

It also becomes a learning problem, when functional approximation is needed.



Traveling salesman problem

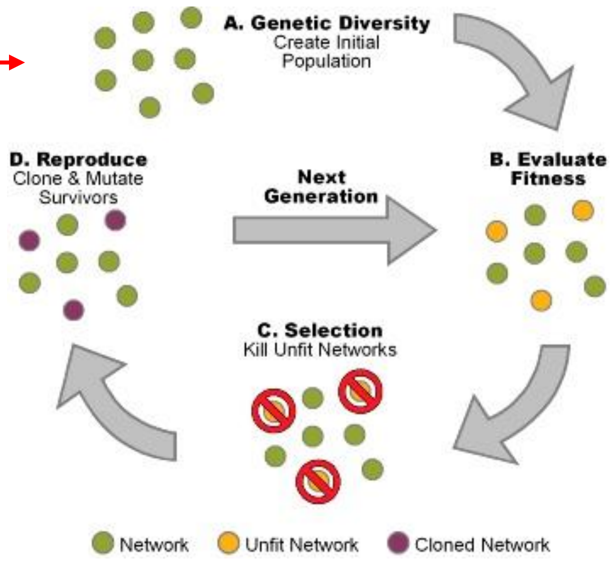


Maximum cut

Vs. Evolutionary methods

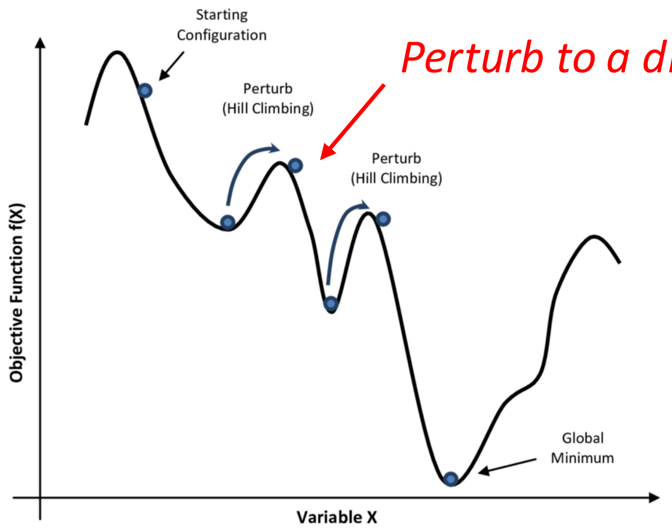
- A family of trial-and-error search solutions
  - Over the population of policies
  - Fail to leverage detailed problem structure

Policies



Genetic programming

Perturb to a different policy



Simulated annealing

Why reinforcement learning

- Sequential decision making is everywhere

2016

1997

Why reinforcement learning

- Sequential decision making is challenging
  - Huge search space

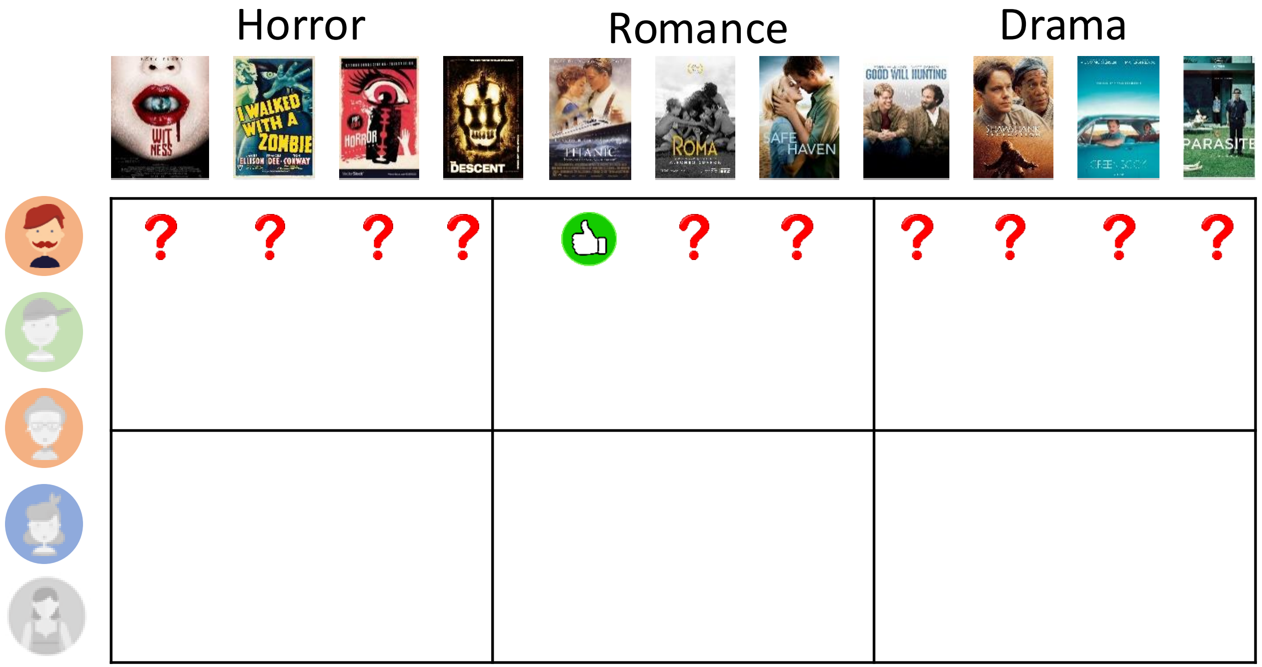
Board size n×n	3^{n^2}	Percent legal	L (legal positions) (A094777)[11]
1×1	3	33.33%	1
2×2	81	70.37%	57
3×3	19,683	64.40%	12,675
4×4	43,046,721	56.49%	24,318,165
5×5	847,288,609,443	48.90%	414,295,148,741
9×9	$4.43426488243 \times 10^{38}$	23.44%	$1.03919148791 \times 10^{38}$
13×13	$4.30023359390 \times 10^{80}$	8.66%	$3.72497923077 \times 10^{79}$
19×19	$1.74089650659 \times 10^{172}$	1.20%	$2.08168199382 \times 10^{170}$

Source: https://en.wikipedia.org/wiki/Go_and_mathematics

Complexity: 10^{50}

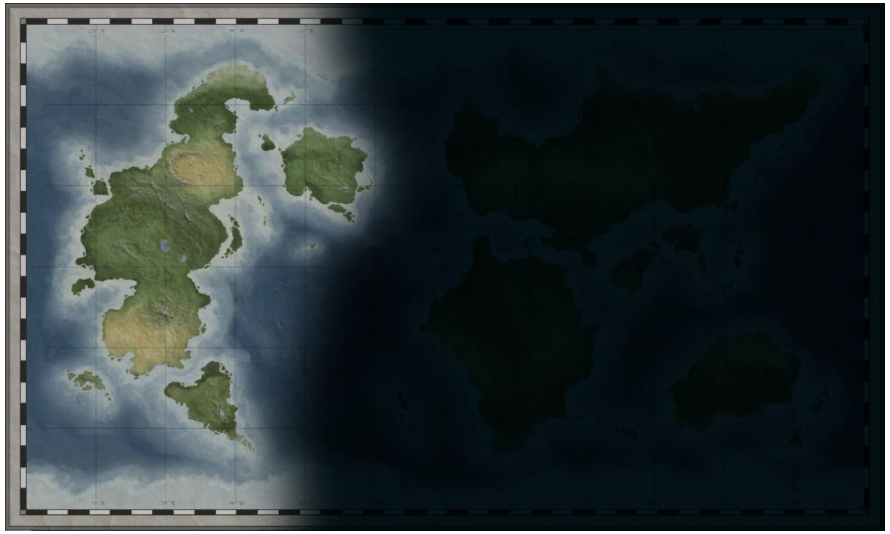
Why reinforcement learning

- Sequential decision making is challenging
  - Huge unknown search space



Why reinforcement learning

- Sequential decision making is challenging
  - Huge unknown search space
    - Supervised ML: generalize to unseen
    - RL: what to generalize



Why reinforcement learning now

- Computational model
  - Deep learning enables sophisticated functional approximation
- Computational power

The system d ... rom
brute force ... y
parallel, RS/6 ... th ***30***
nodes, with ... SC
microproces ... ***ose***
VLSI chess ch

Source: https:/ ... mputer)

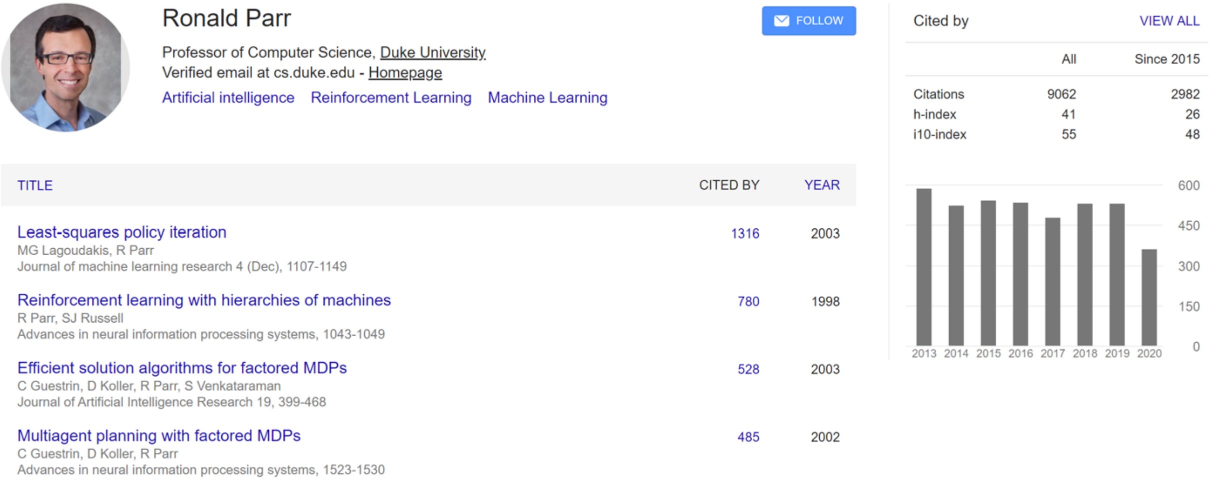
on and strength[61]

		Elo rating	D	
A		3,144[51]	Oc	
A		3,739[51]	Ma	
A		4,858[51]	Ma	
		5,185[51]	Oc	
AlphaZero (20 block)	4 TPUs, single machine	5,018 [62]	Dec 2017	60:40 against AlphaGo Zero (20 block)

Source: https://en.wikipedia.org/wiki/AlphaGo

Why reinforcement learning now

- Low hanging fruits in other machine learning fields have been plucked
  - An argument by Prof. Ronald Parr ☺
- The development in other machine learning fields prepared us
  - Optimizat
  - Deep lear
- Demand frc
  - Self-drivir
  - Conversat

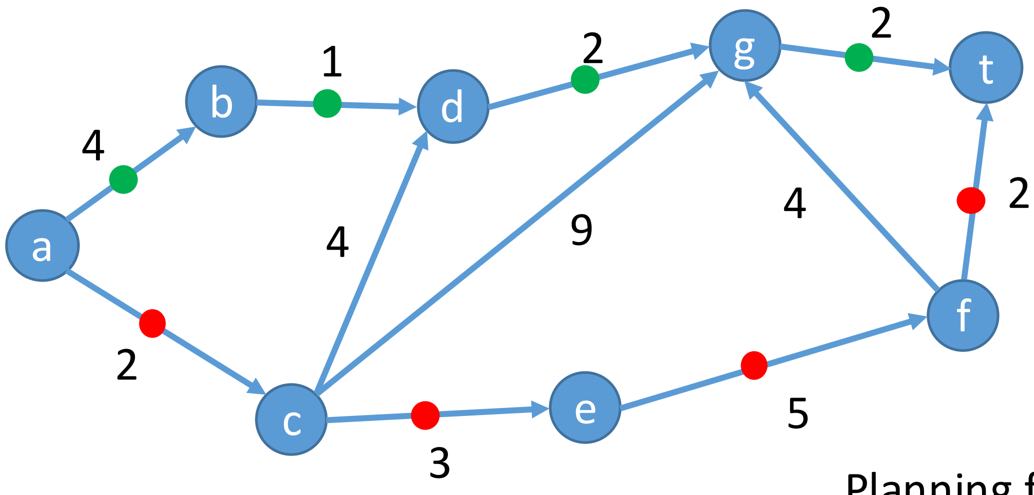


How to do reinforcement learning

- With a known environment
  - Planning - a classical AI search problem

Dijkstra's algorithm:

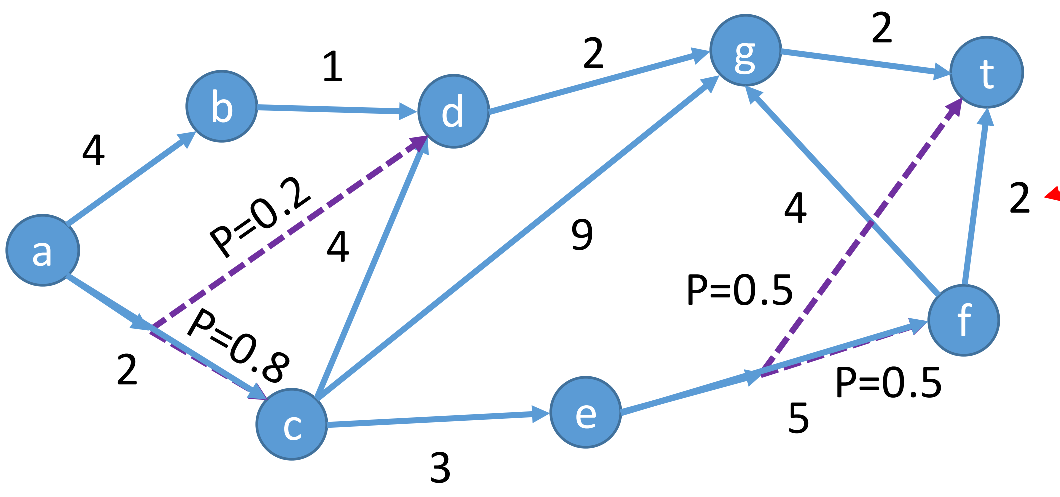
$$\Theta((|V| + |E|) \log |V|)$$



Planning for the long-term is necessary

How to do reinforcement learning

- With a known but stochastic environment
  - Planning

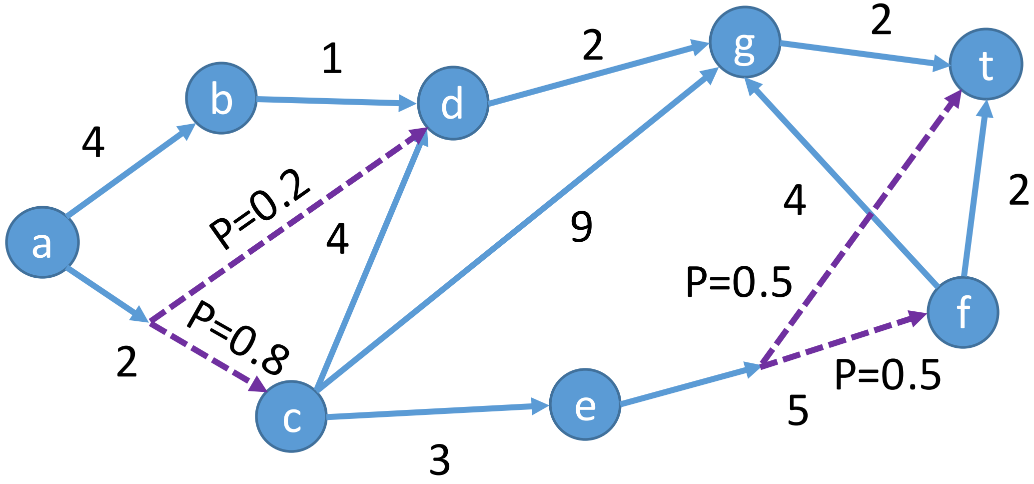


This could also be stochastic!

Example credit: Jiang, UIUC CS-498

How to do reinforcement learning

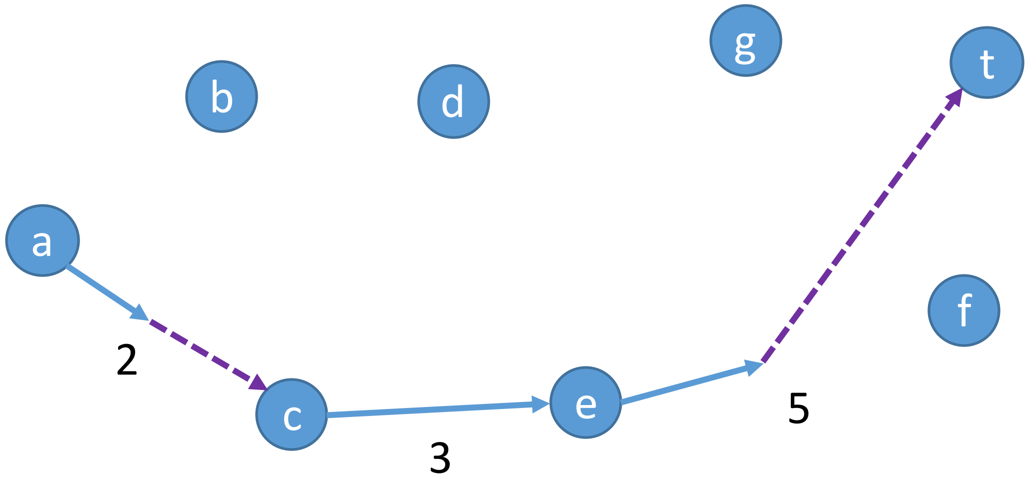
- With an <u>unknown</u> and stochastic environment
  - Planning



Example credit: Jiang, UIUC CS-498

How to do reinforcement learning

- With an <u>unknown</u> and stochastic environment
  - Planning
    - Trial and error

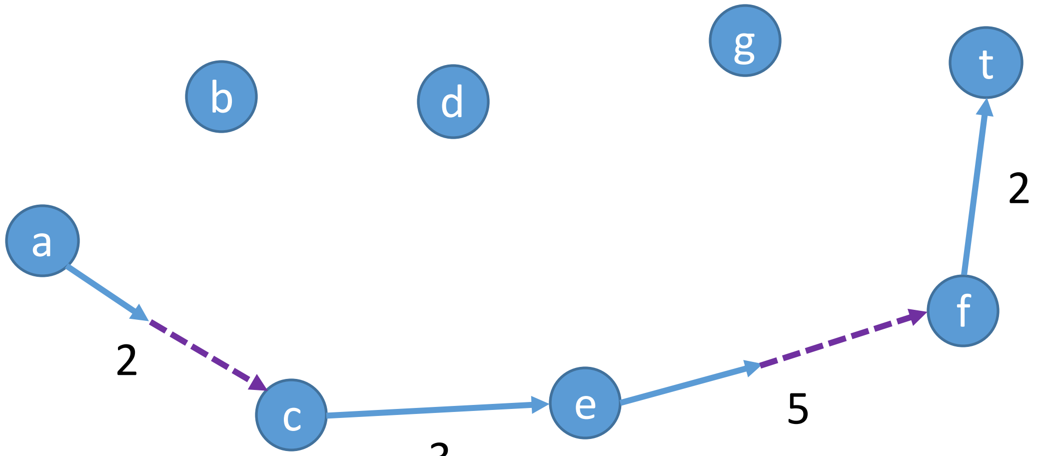


Experience:
$a \to c \to e \to t$

Example credit: Jiang, UIUC CS-498

How to do reinforcement learning

- With an <u>unknown</u> and stochastic environment
  - Planning while learning
    - Trial and error



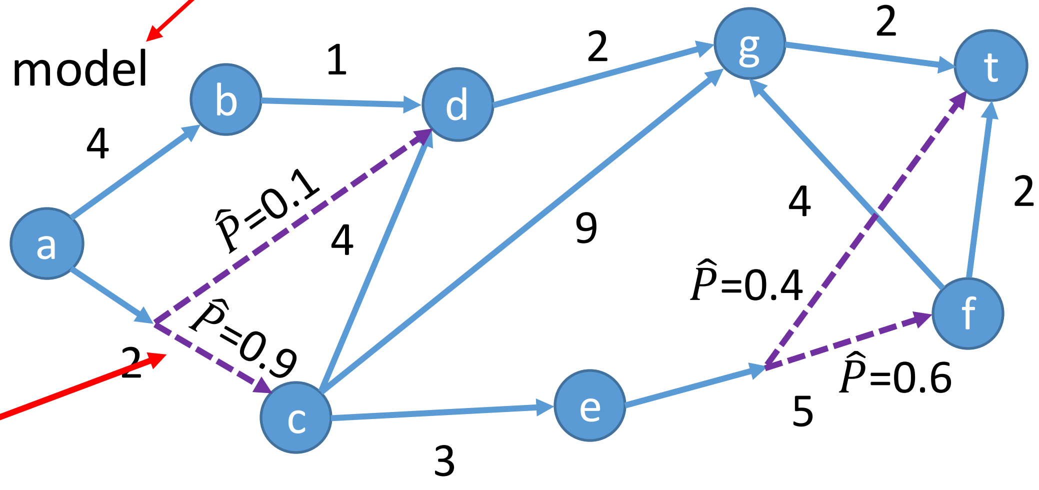
Experience:
$a \to c \to e \to t$
$a \to c \to e \to f \to t$

Example credit: Jiang, UIUC CS-498

How to do reinforcement learning

- With an <u>unknown</u> and stochastic environment
  - Planning while learning
    - Trial and error
  - Model-based
    - Experience -> model

Bias v.s., variance?



Experience:
$a \to c \to e \to t$
$a \to c \to e \to f \to t$

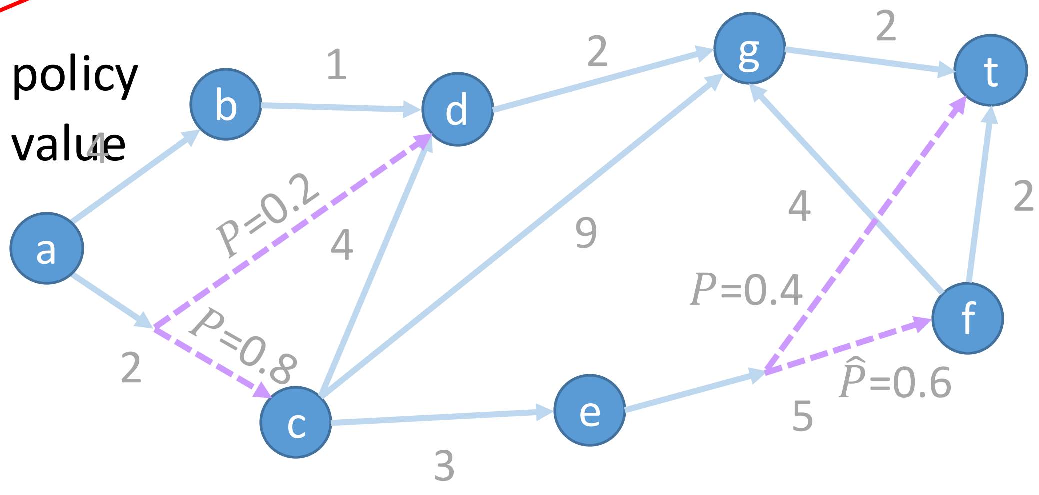
Convergence of the model matters!

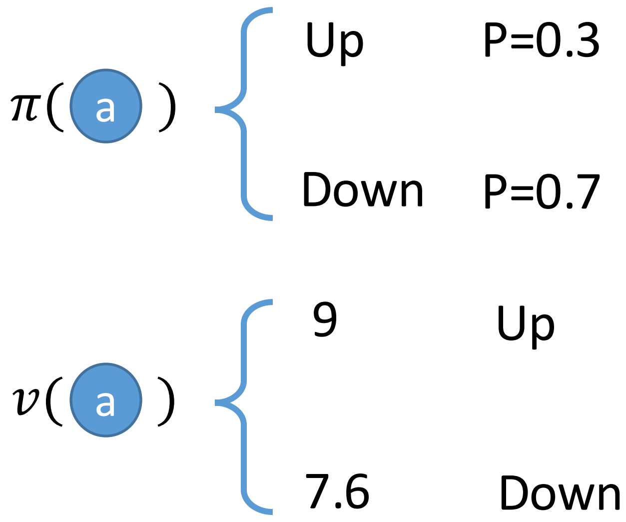
Example credit: Jiang, UIUC CS-498

How to do reinforcement learning

- With an <u>unknown</u> and stochastic environment
  - Planning while learning
    - Trial and error
  - Model-free
    - Experience -> policy
    - Experience -> value

How do we get such experiences matters!
I.e., the explore-exploit trade-off; sometimes it is also the bias-variance trade-off





Experience:
$a \to c \to e \to t$
$a \to c \to e \to f \to t$

Example credit: Jiang, UIUC CS-498

How to do reinforcement learning

- With an unknown and stochastic environment
  - A taxonomy of solutions

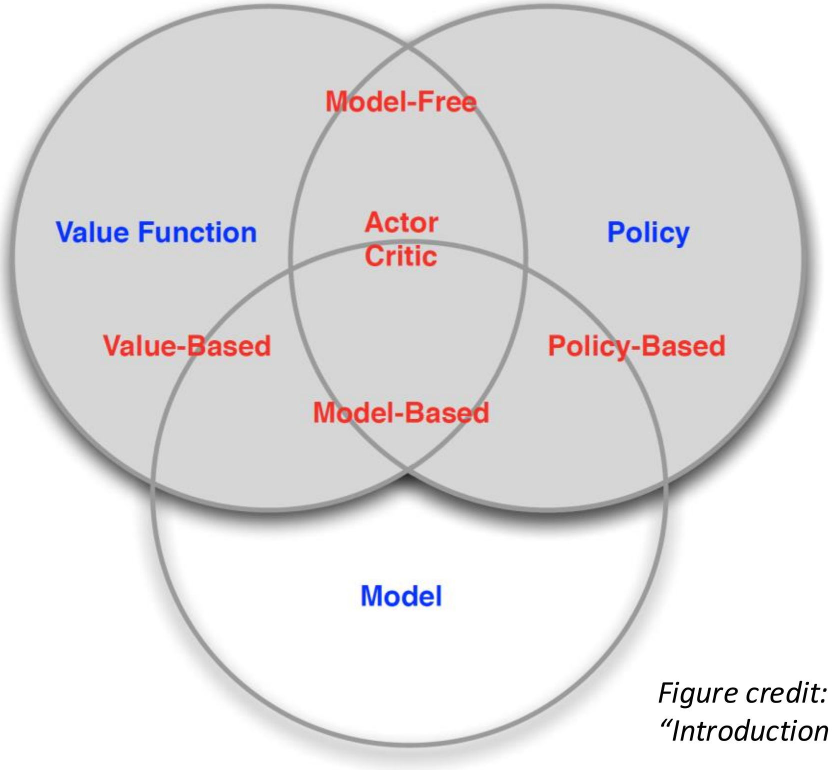


Figure credit: David Silver, "Introduction to RL"

How to do reinforcement learning

- A brief history of reinforcement learning research
    - Planning – originated in optimal control
        - Dated back to 1950s
    - Reinforcement learning – originated in psychology of animal learning
        - Dated back to 1850s

Reinforcement learning in practice is challenging

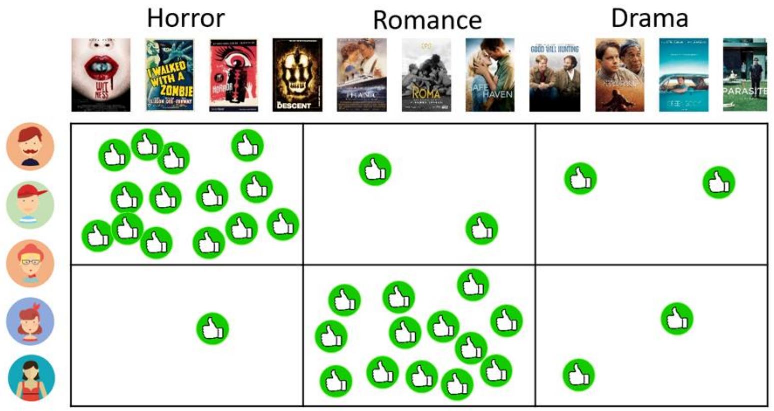
- Learning by trial and error is expensive
  - We need an environment to repeatedly interact with

Environment defined by game rules physics rules

Simulation is easy

Reinforcement learning in practice is challenging

- Learning by trial and error is expensive
  - We need an environment to repeatedly interact with



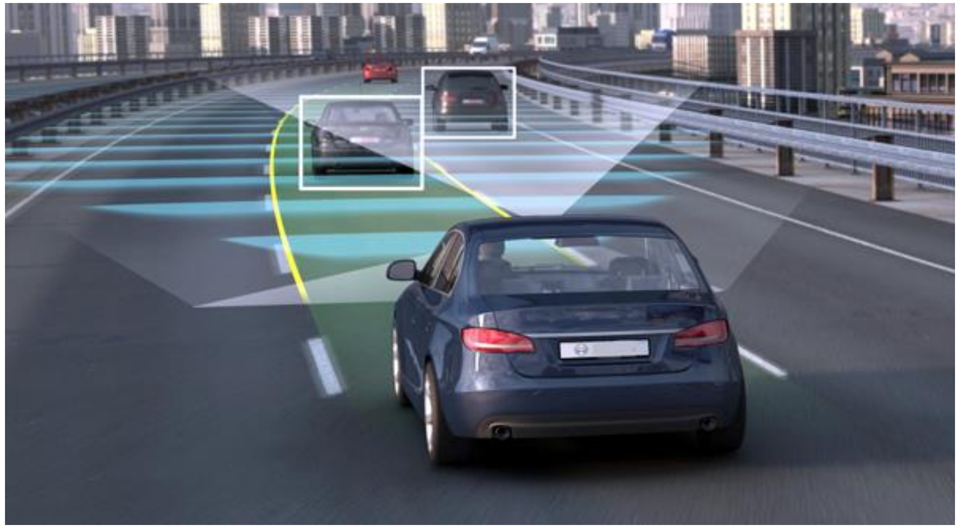
Environment defined by

Users' behaviors

Simulation is hard!

Reinforcement learning in practice is challenging

- There are also safety, privacy, and ethic concerns in exploration
  - We are dealing with unknown unknowns
  - We are learning from explorations

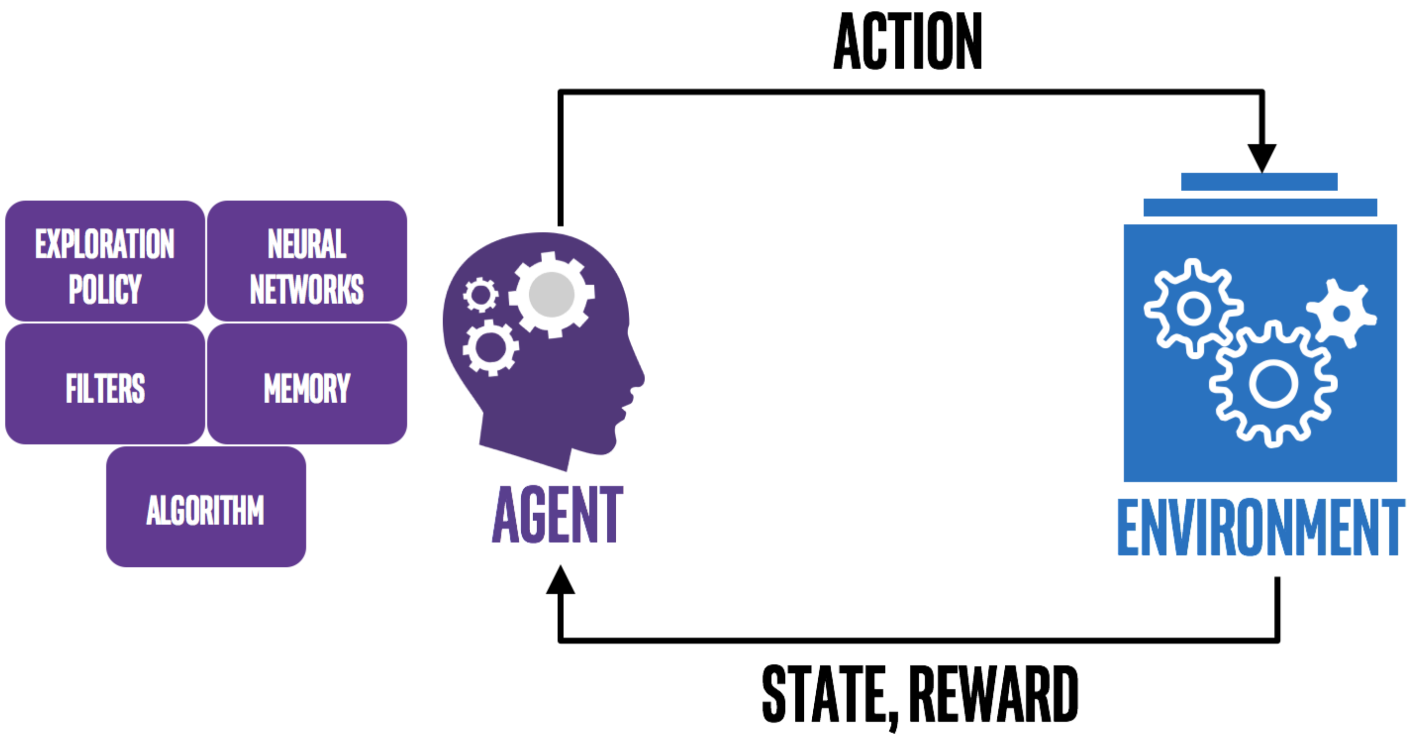


Takeaways

- Reinforcement learning is for sequential decision making
- Reinforcement learning overlaps heavily with different machine learning techniques, but also uniquely differs from them
- Known environment v.s., unknown environment
- Model-based v.s., model-free

References

- Nan Jiang, CS 498 Reinforcement Learning, University of Illinois at Urbana-Champaign.
- Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.
- Sutton & Bartol Reinforcement Learning: An Introduction



Basics of Reinforcement Learning

Outline

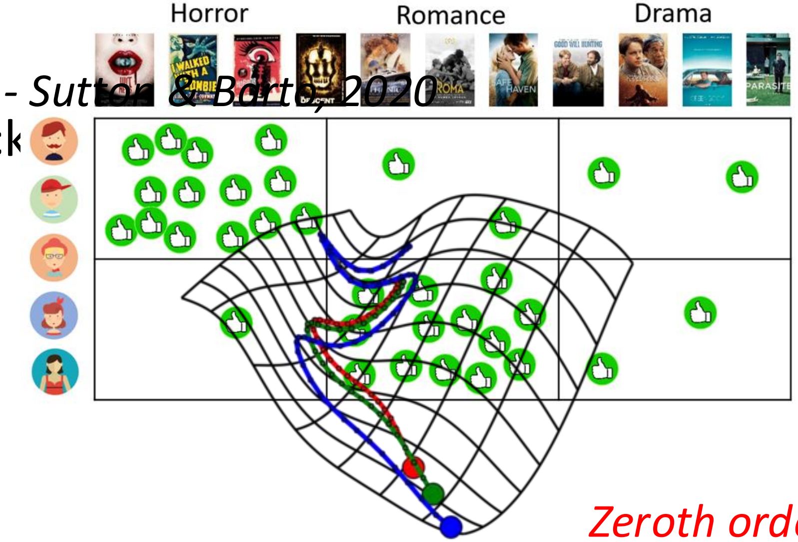
- Introduction
  - What is reinforcement learning?
  - Why do we need it?
  - How to?
- **Basics of RL**
  - **Action vs. reward**
  - **State vs. value**
  - **Policy**
  - **Model**

Action taking in reinforcement learning

- Making a choice out of $\boxed{\text{presented options}}$ ← *Out of agent's control!*
  - Discrete actions
    - Move left or right in Atari Breakout game
    - Recommend an item to a target user
  - Continuous actions
    - Drone/r
    - Model s… a black

Reinforcement learning is learning what to do — how to map situations to actions — so as to maximize a numerical reward signal.

- Sutton & Barto, 2020



Zeroth order optimization

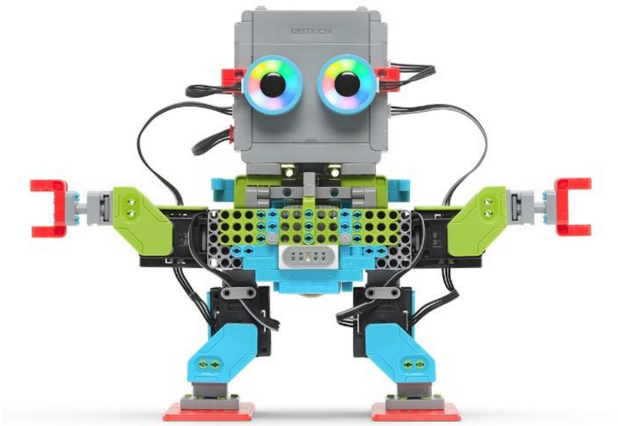
Reward in reinforcement learning

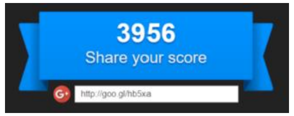
- A scalar feedback signal about the taken action
  - Suggest good/bad <u>immediate</u> consequence of the action
    - Score in Atari game
    - User clicks/purchase in a recommender system
    - Change of black-box function value
  - Delayed feedback
    - GO game
    - Generate a sentence in chat-bot
  - Goal of learning – maximize cumulative rewards
    - Reward hypothesis: "*All goals can be described by the maximization of expected cumulative reward.*"

How to take an action

- With respect to the current observation

Observation o_t





Reward r_t

Action a_t

How to take an action

- With respect to history
  - How did we reach the current observation

o_t

History — Current observation

  - Why do we care about history?
    - In case this has happened before
    - Generalize from history
  - State – a function of history
    - $s_t = \underbrace{f(o_1, a_1, r_1, o_2, a_2, r_2, \ldots \ldots, o_t)}_{\textit{How to construct states?}}$

How to take an action

- To maximize cumulative reward in future

$$\underbrace{o_1, a_1, r_1, o_2, a_2, r_2, \dots \dots, o_t}_{\text{History}}, a_t \quad \underbrace{\quad \quad \quad \quad}_{\text{Future}}$$

History

Future

  - Value function
    - State-action value $v_\pi(s_t, a_t) = \boldsymbol{E}_\pi\left[\sum_{i=t}^{T} \boxed{\gamma^{i-t}} r_{i-t+1}\right]$ *Oftentimes approximation is needed*

With respect to a particular policy!

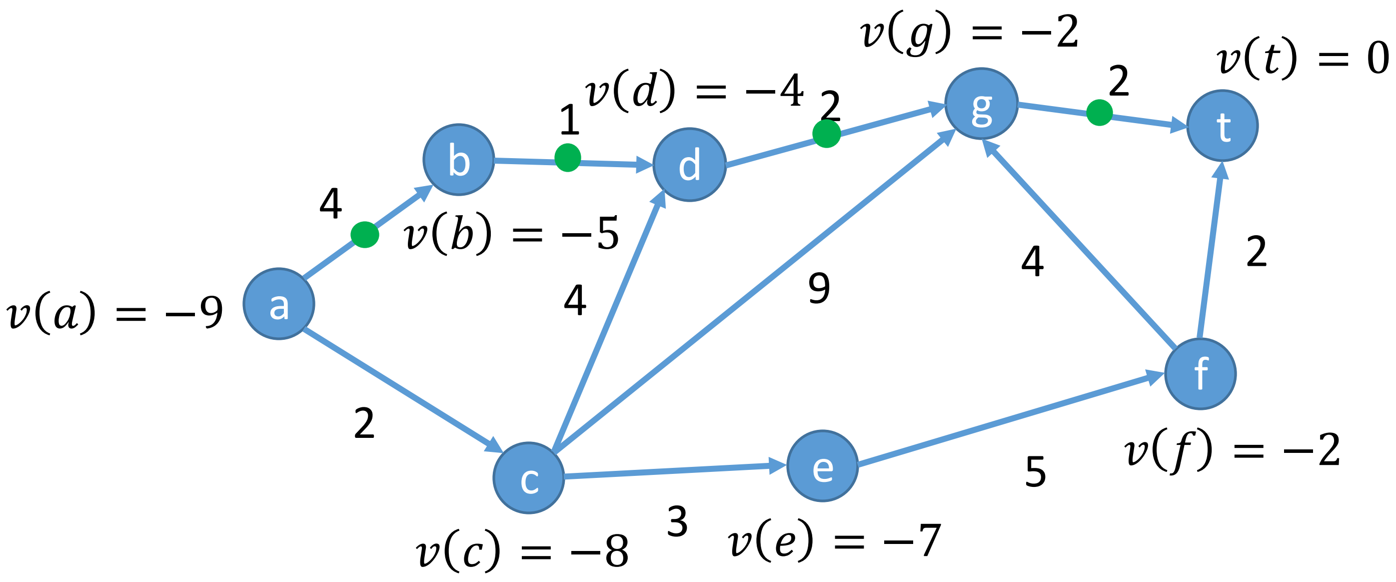
Why do we need this?

    - State value $v_\pi(s_t) = \boldsymbol{E}_{a_t \sim \pi(s_t)}[v_\pi(s_t, a_t)]$
  - Goal: choose an action that leads to a highest value state

Action taking by value function

- Shortest path as an example
  - State:
  - Action:
  - Reward:
  - Value:

Now how should we act?



W.r.t. optimal policy

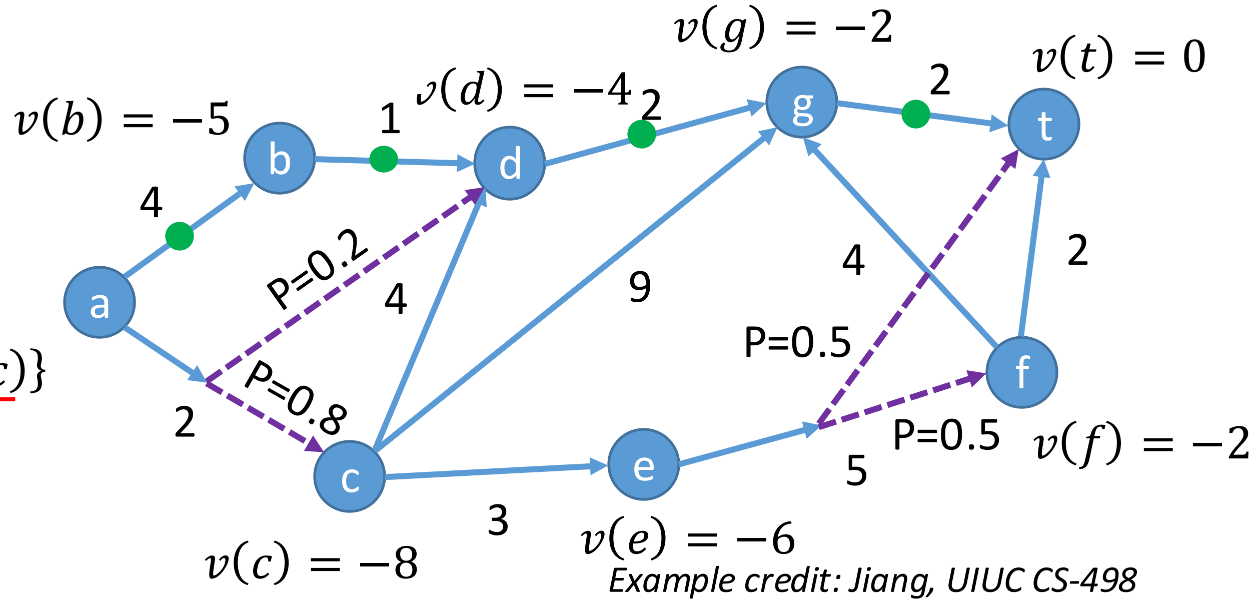
Action taking by value function

- Shortest path as an example
  - State: current node
  - Action: take an outgoing edge
  - Reward: (negative) edge weight
  - Value:

Now how should we act?

$$v(e) = -5 + 0.5 \times v(t) + 0.5 \times v(f) = -6$$

$$v(a) = \max\{-4 + v(b), \underline{-2 + 0.2 \times v(d) + 0.8 \times v(c)}\}$$
$$= -9 \qquad\qquad -9.2$$

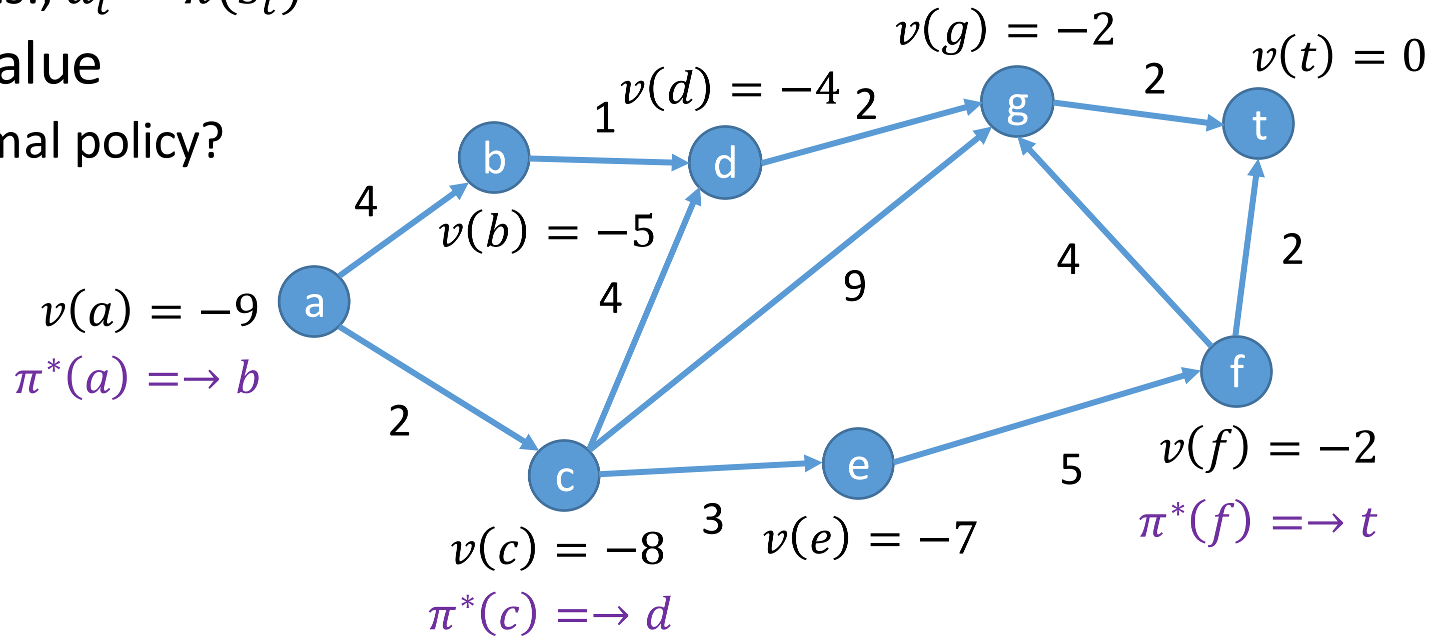


Example credit: Jiang, UIUC CS-498

W.r.t. optimal policy

Policy

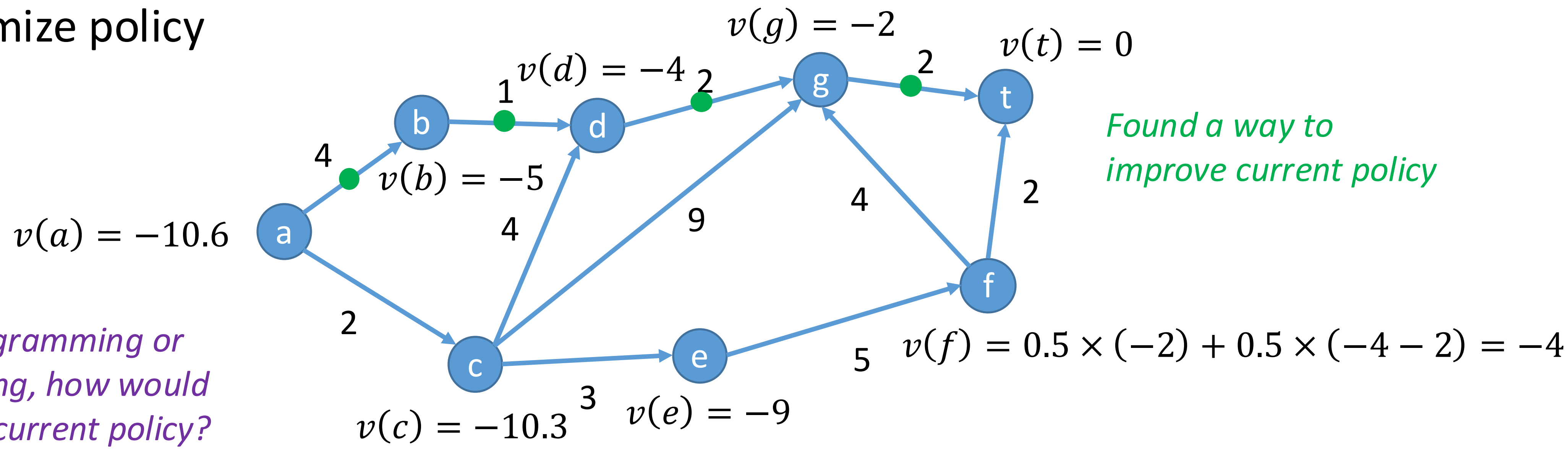
- A mapping from state to action
  - By the agent!
  - Deterministic or stochastic
    - Notation-wise: $a_t = \pi(s_t)$ v.s., $a_t \sim \pi(s_t)$
  - Optimal policy maximizes value
    - Value function gives us optimal policy?



Prediction vs. Control

- Prediction
  - Evaluate value function given a policy
- Control
  - Optimize policy

Sort of E-step vs., M-step?



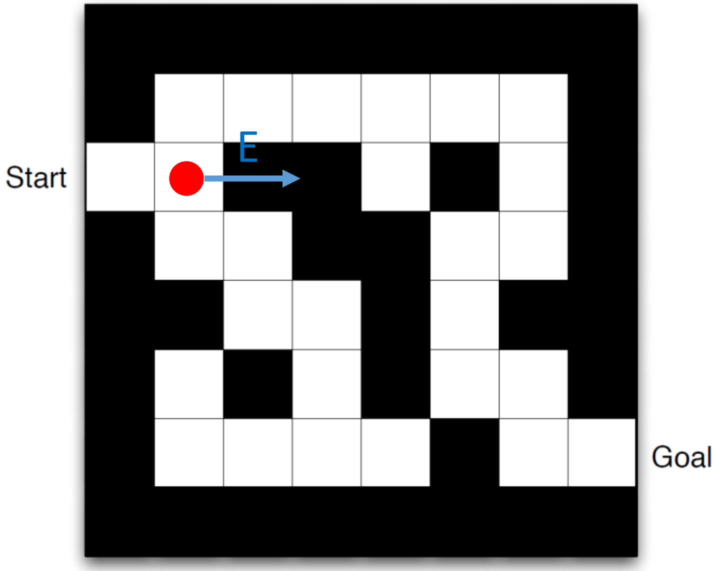
Found a way to improve current policy

Recall genetic programming or simulated annealing, how would they optimize the current policy?

W.r.t. a random policy

Model

- A specification of environment
  - If take an action a_t now,
    - What is the next observation o_t, or the state s_t?
    - What is the reward r_t?



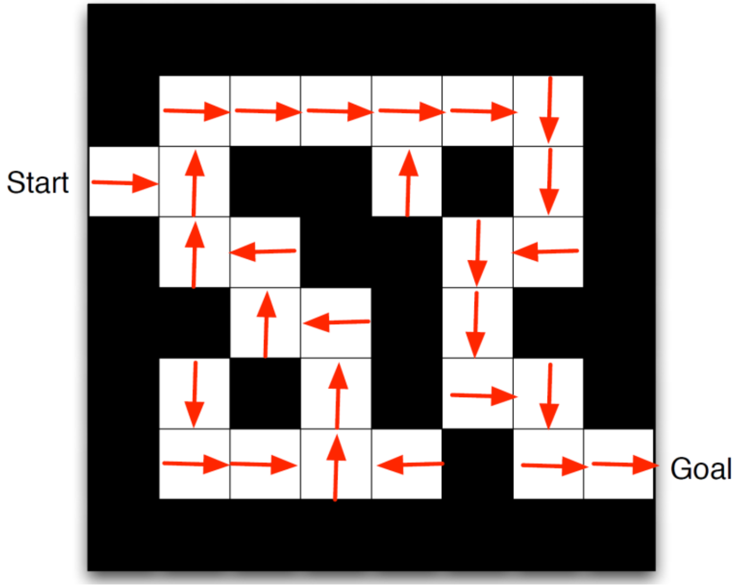
Action: N,S,E,W
Reward: -1
State: current position
Model: configuration of the maze

Example credit: David Silver, "Introduction to RL"

Model

- A specification of environment
  - If take an action a_t now,
    - What is the next observation o_t, or the state s_t?
    - What is the reward r_t?

Optimal policy



Action: N,S,E,W
Reward: -1
State: current position
Model: configuration of the maze

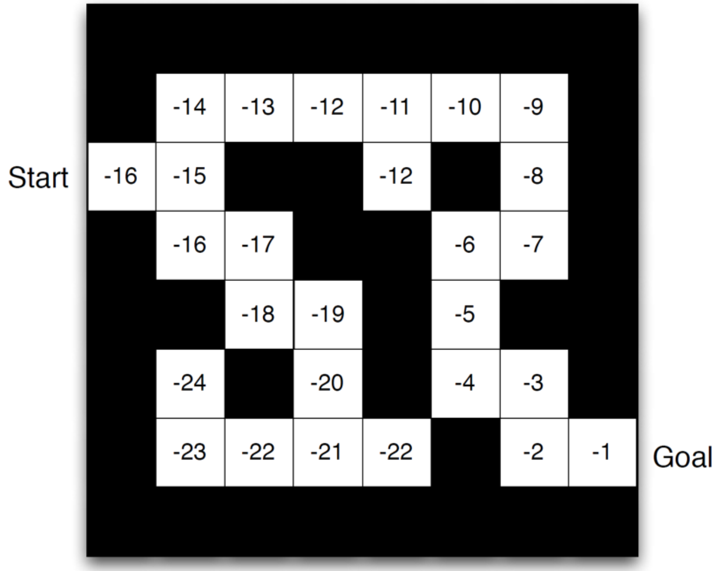
Should be defined for all states!

Example credit: David Silver, "Introduction to RL"

Model

- A specification of environment
  - If take an action a_t now,
    - What is the next observation o_t, or the state s_t?
    - What is the reward s_t?

Value under optimal policy

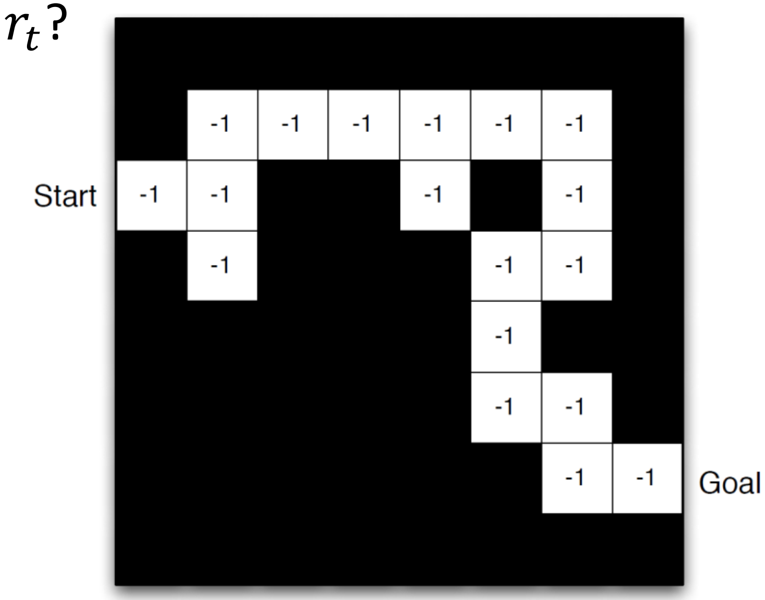


Action: N,S,E,W
Reward: -1
State: current position
Model: configuration of the maze

Example credit: David Silver, "Introduction to RL"

(Estimated) Model

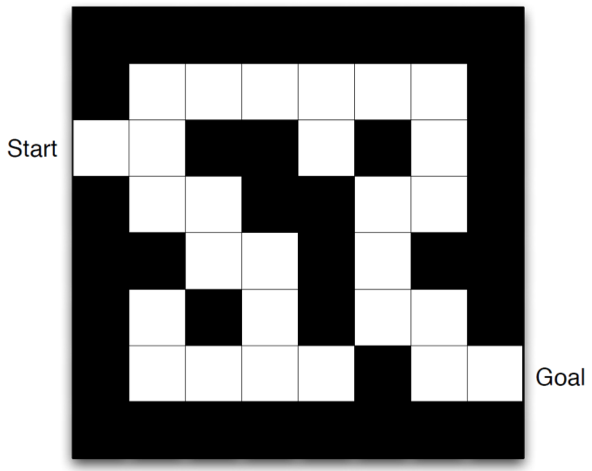
- An agent's perspective of the environment
  - Estimated from history – the learning part
  - If I take an action a_t now,
    - What might be the next observation o_t, or the state s_t?
    - What might be the reward r_t?



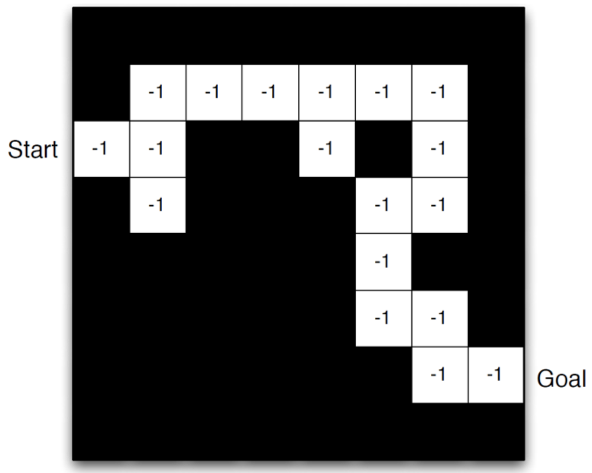
Action: N,S,E,W
Reward: -1 for visited states so far
State: current position
Model: estimated configuration of the maze

Models

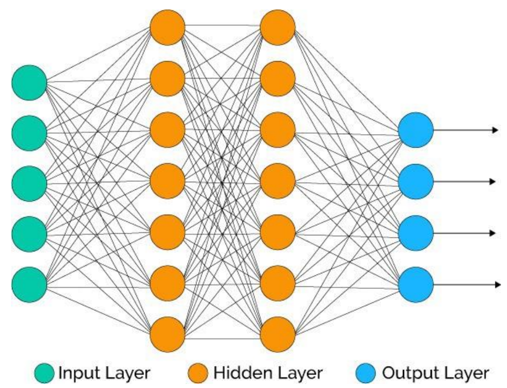
- Environment model
  - Ground-truth construction
  - Might be given sometimes



- Estimated environment model
  - Agent's belief
  - Might not be truthful



- Agent's model
  - The mathematical/ statistical formulation used by the agent for estimation



Takeaways

- RL agents take actions with respect to history/state
- Their goal is to find highest value states
- Model is about the environment, and can be estimated by the agent