

Probability Concepts and its Applications in Reinforcement Learning

Dr. Sriparna Saha
Associate Professor
Department of Computer Science and Engineering
Indian Institute of Technology Patna, Bihar, India

With special thanks to
Martha and Adam, University of Alberta - Fundamentals of Reinforcement Learning
Fei-Fei Li & Justin Johnson & Serena Yeung - CS 231 Reinforcement Learning

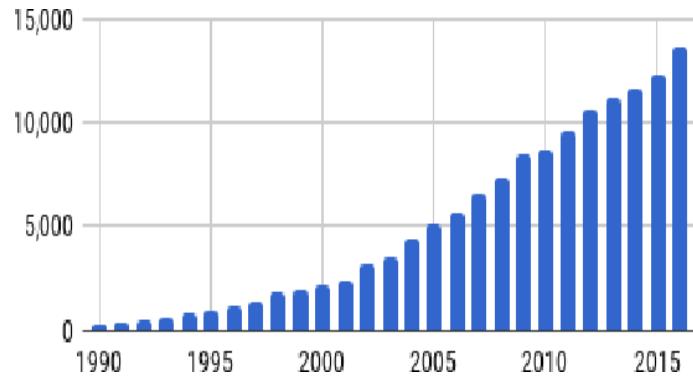
Some Quotes and Statistics

“ I can not disguise my satisfaction that it plays with a very dynamic style, much like my own “ - Garry Kasparov, former World Chess Champion

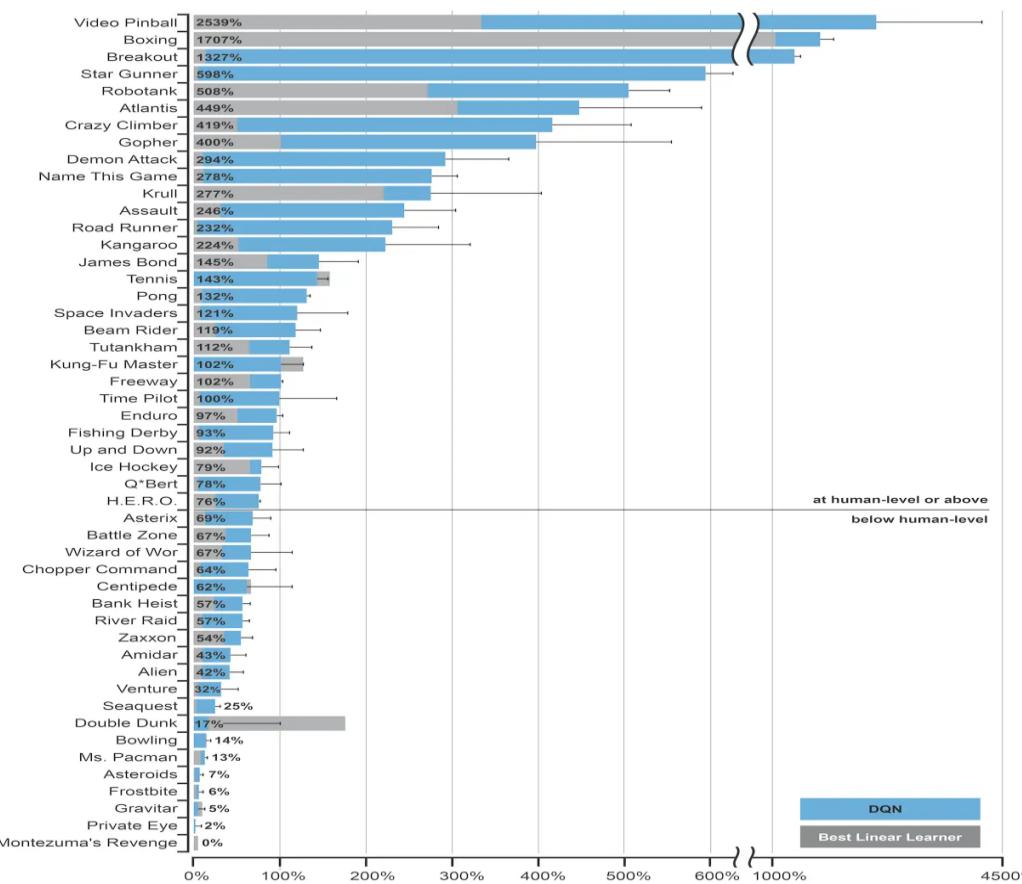
“Deep meta reinforcement learning will be the future of artificial intelligence which aims to build a single model to master a wide variety of tasks. Thus each model will be capable to perform a wide range of complex tasks” - Sudharsan Ravichandiran, Hands-On Reinforcement Learning with Python



In March 2016, AlphaGo defeated Lee Sedol (the strongest player of the last decade with an incredible 18 world titles)



Growth of published reinforcement learning papers, Henderson et al., 2019



Amazingly, DQN achieved human-level performance in almost half of the 50 games to which it was applied; Mnih et al., 2015

Resources

Text Book

Reinforcement Learning: An Introduction

Link

Reinforcement Learning

An Introduction
second edition

Richard S. Sutton and Andrew G. Barto



Some Pioneer Scientists in Reinforcement Learning Field



Rich Sutton, Google Deepmind
“Father of Reinforcement Learning”



David Silver, Google Deepmind
Deep Reinforcement Learning (DQN)

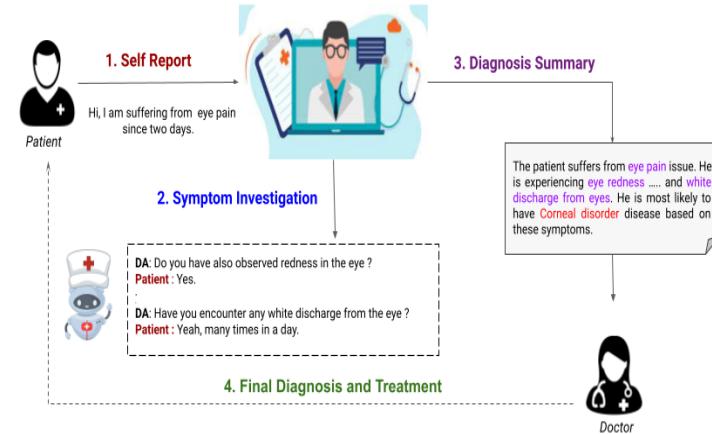


Csaba Szepesvári, Google Deepmind
Monte-Carlo tree search algorithm

Our Vision

- **Development of an Intelligent Health Assistant**

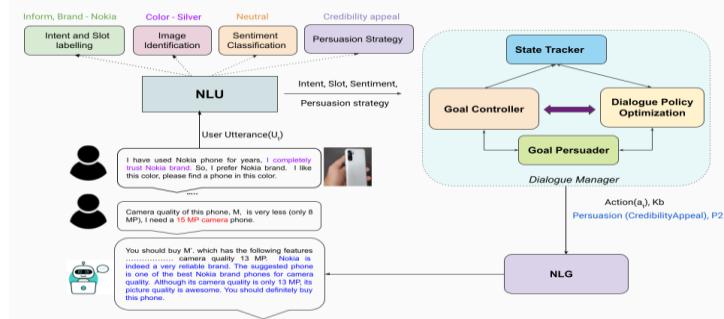
- Doctor : population (shortage of 7.2 millions medical workers \Rightarrow 12.9 millions in the upcoming decade, WHO, 2013)
- Doctors Accessibility in Rural areas
- An AI based Assistant to Doctors



- **Towards Building Multimodal, Multi-intent Dialogue Agent**

Humans \longleftrightarrow Computers : *Task oriented Dialogue Assistant*

- ❖ Task accomplishment is key priority for both users and agent
- ❖ Adversarial Situations : Dynamic Goal, Task Goal Unavailability



Automation, Programming & Machine Learning

Automation Task \Rightarrow Model (Rules) \Rightarrow Machine

Manual effort \Rightarrow Machine

Repetitive, Instructive tasks

E.g. : Automatic Floor Cleaner



6 5 3 1 8 7 2 4

Programming

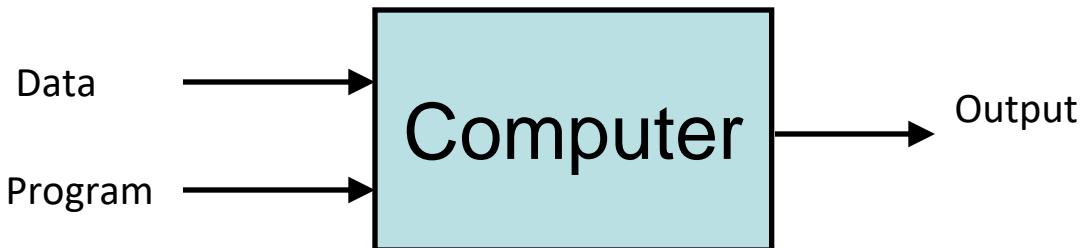
Task automation using computer programs

Problem solution \Rightarrow Set of Instructions

E.g. : Searching, Sorting

Machine Learning : Teaching a task to an autonomous agent using previous experiences (data).
Let the data do the work

Programming Vs Machine Learning



Simple task , Output can easily be expressed in terms of input ;

Example : Task (Square of a number)
$$Y = f(x)$$
$$= x^2$$

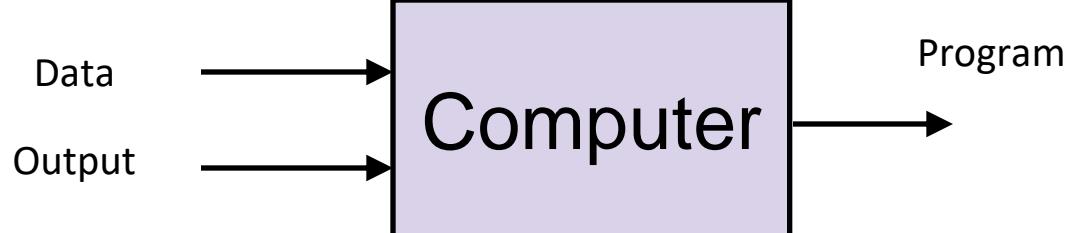
Inference : $125 \Rightarrow$ Computer $\Rightarrow 15625$

Complex task, Huge amount of available data

Example : Task (?)
Data, Output : $(2, 5), (3, 11), \dots$
 $(100, 10099)$

$$Y = g(x)$$
$$= x (x + 1) - 1$$

Inference : $125 \Rightarrow 15624$



Magic ?

No, more like gardening

- **Seeds** = Algorithms
- **Nutrients** = Data
- **Gardener** = You
- **Plants** = Programs



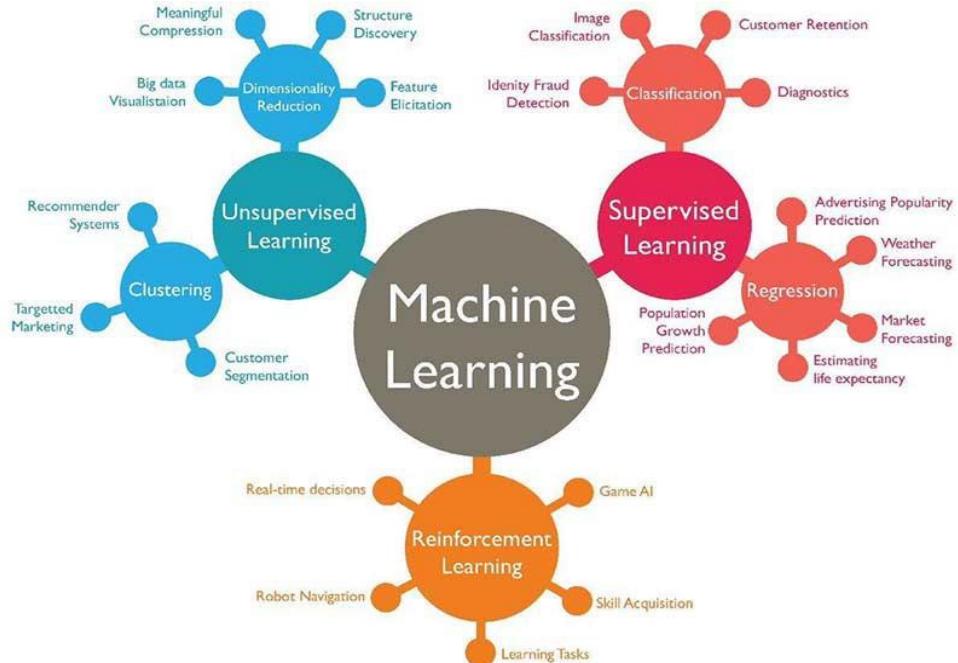
Machine Learning Techniques

1) Supervised (inductive) Learning

1) Unsupervised Learning

1) Semi-Supervised Learning

1) Reinforcement Learning



Supervised Learning

Data : (Input, Output)

(X, Y)

Target : Learn a function, 'f' that maps X → Y

$$f^* = \min \text{Error } (Y_v, f(X_v))$$

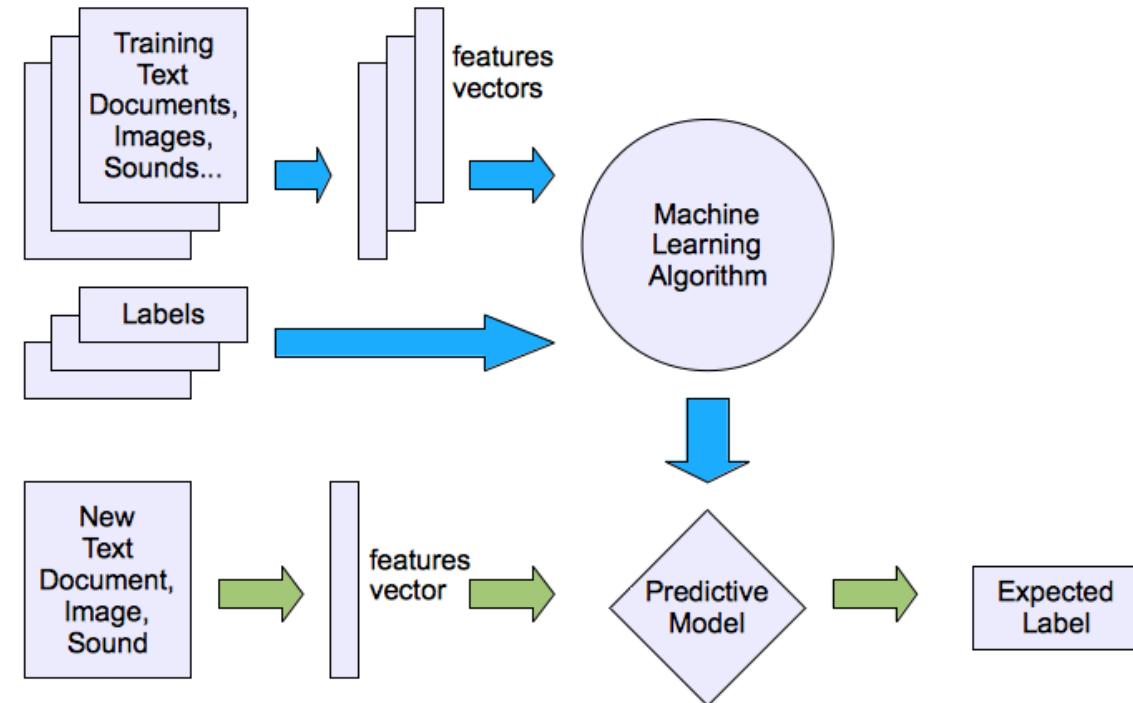
Examples :

- Classification
- Regression
- Object detection
- Image Captioning



Model
→ **Cat**

Supervised Learning

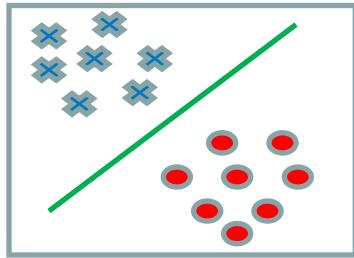


Supervised learning techniques

- **Linear classifier** (numerical functions)
- **Parametric** (Probabilistic functions)
 - Naïve Bayes, Hidden Markov models (HMM), Probabilistic graphical models
- **Non-parametric** (Instance-based functions)
 - K -nearest neighbors, Kernel regression, Local regression
- **Non-metric** (Symbolic functions)
 - Classification and regression tree (CART), decision tree
- **Aggregation**
 - Bagging (bootstrap + aggregation), Adaboost, Random forest

Supervised learning techniques

Linear classifier



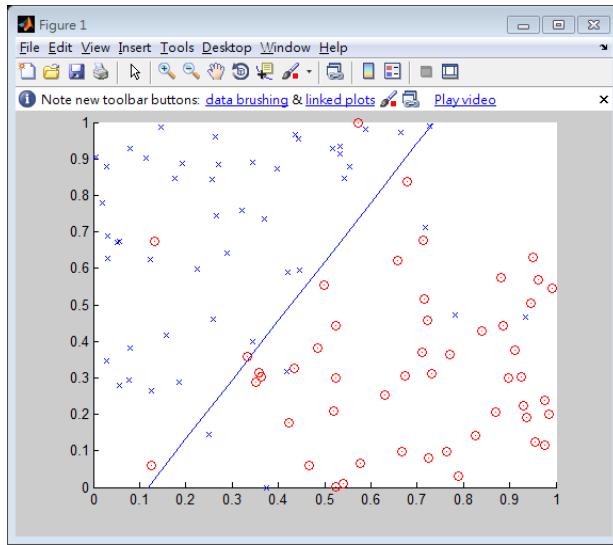
$$g(x_n) = \text{sign}(w^T x_n)$$

where w is an d -dim vector (learned)

Techniques:

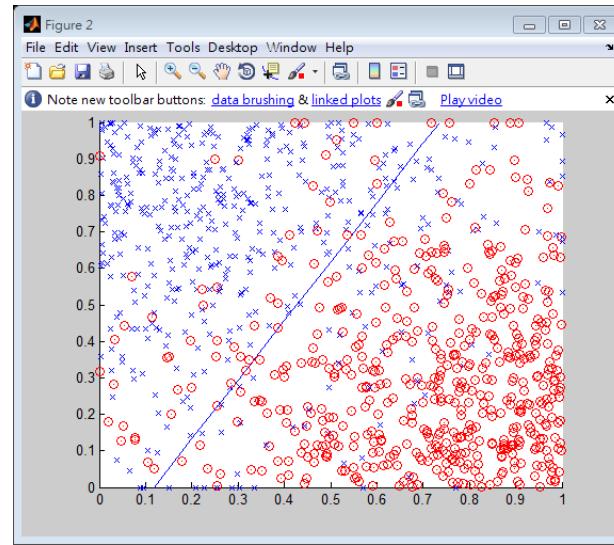
- Perceptron
- Logistic regression
- Support vector machine (SVM)
- Ada-line
- Multi-layer perceptron (MLP)

Using perceptron learning algorithm(PLA)



Training

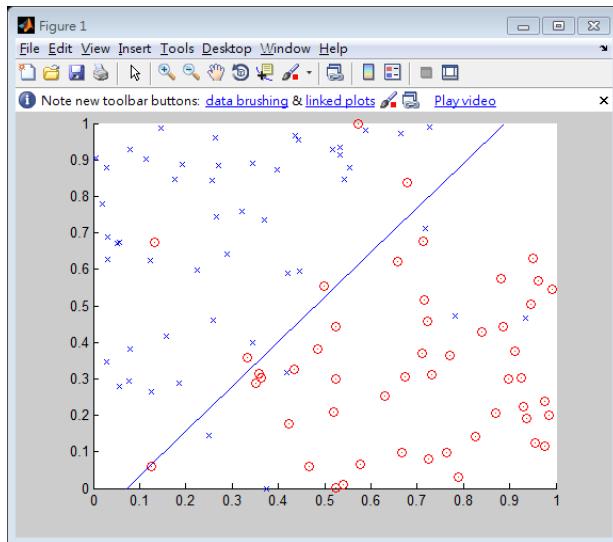
Error rate: 0.10



Testing

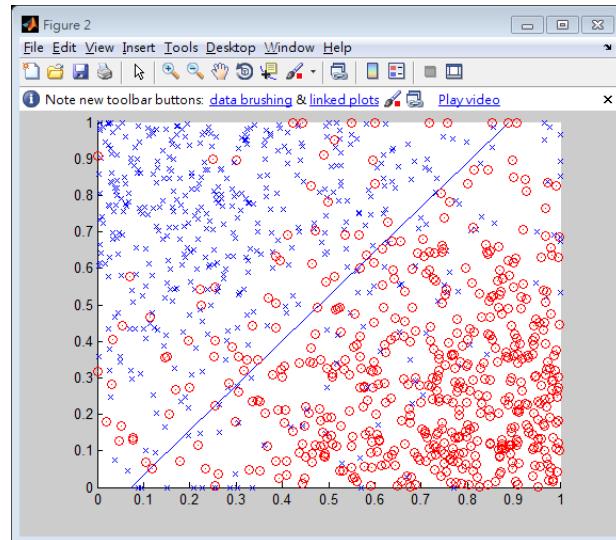
Error rate: 0.156

Using logistic regression



Training

Error rate: 0.11

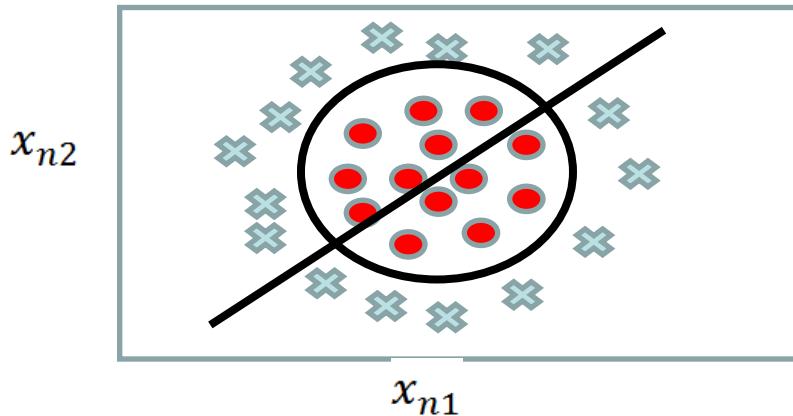


Testing

Error rate: 0.145

Supervised learning techniques

- Non-linear case



$$\begin{aligned}x_n &= [x_{n1}, x_{n2}] \\&\quad \downarrow \\x_n &= [x_{n1}, x_{n2}, \color{red}{x_{n1} * x_{n2}}, \color{red}{x_{n1}^2}, \color{red}{x_{n2}^2}] \\g(x_n) &= \text{sign}(w^T x_n)\end{aligned}$$

- Support vector machine (SVM):
 - Linear to nonlinear: **Feature transform** and **kernel function**

Unsupervised Learning

Data : X

Just data ; no label

Target : Learn hidden structure of data

Sample : $x_1, x_2, x_3, \dots, x_n$

$(x_i, x_j, x_m \dots), (x_a, x_b, x_c \dots) \dots, (x_p, x_q \dots)$

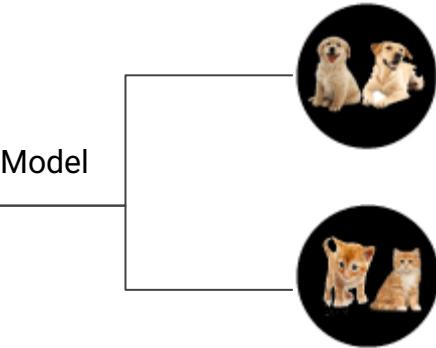
Objective : Min $\sum^K \sum^{\text{point}} d(x_{\text{point}}, C_k)$

Examples :

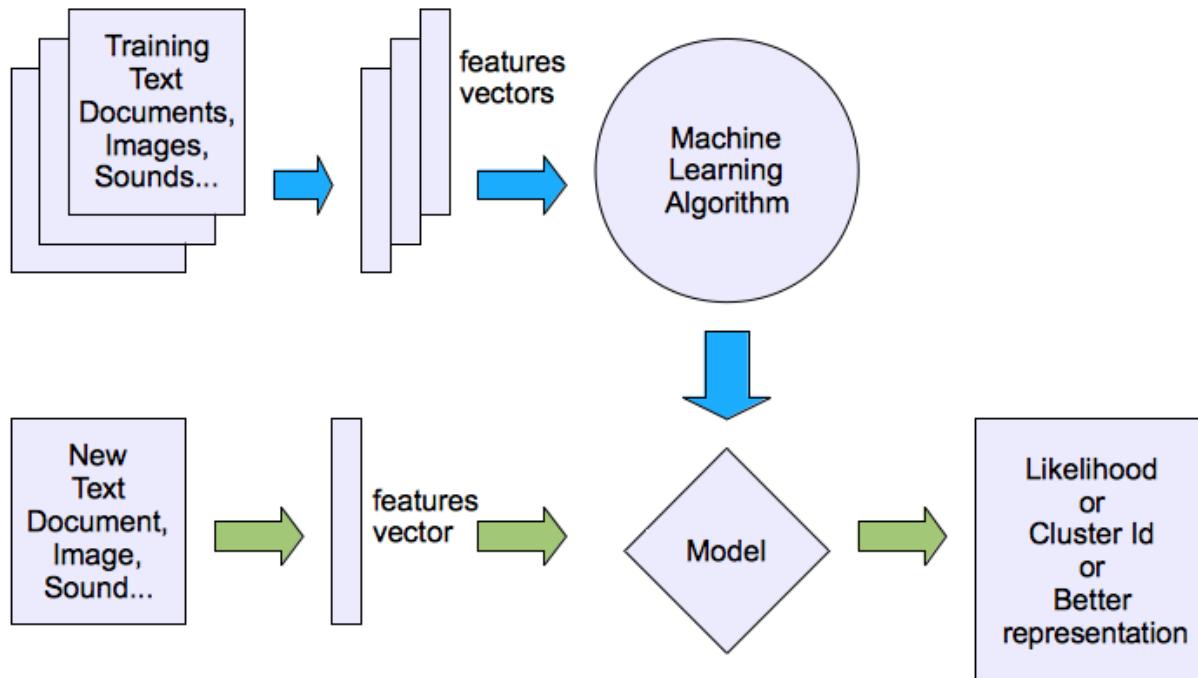
- Clustering
- Dimensionality reduction
- Feature learning



Unlabeled data



Unsupervised Learning



Unsupervised learning techniques

Unsupervised learning categories and techniques

- Clustering
 - K-means clustering
 - Spectral clustering
- Dimensionality reduction
 - Principal component analysis (PCA)
 - Factor analysis

Why “Learn”?

- Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- There is no need to “learn” to calculate payroll
- Learning is used when:
 - Human expertise does not exist (navigating on Mars)
 - Humans are unable to explain their expertise (speech recognition)
 - Solution changes in time (routing on a computer network)
 - Solution needs to be adapted to particular cases (user biometrics)

What learning means ?

- *Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.*
- Example in retail:
People who bought “Da Vinci Code” also bought “The Five People You Meet in Heaven”
www.amazon.com
- **Build a model that is a good and useful approximation to the data.**

Machine learning and our focus

- Human learn from past experiences. However, a computer does not have “experiences”.
- **A computer system learns from data**, which represent some “**past experiences**” of an application domain.
- **Our focus:** learn a target function that can be used to predict the values of a discrete class attribute, e.g., **approve** or **not-approved**, and **high-risk** or **low risk**.

Classification Example

- **Data:** A set of data records (also called examples, instances or cases) described by
 - *k* attributes: $A_1, A_2, \dots A_k$.
 - a class: Each example is labelled with a pre-defined class.
- **Goal:** To learn a classification model from the data that can be used to predict the classes

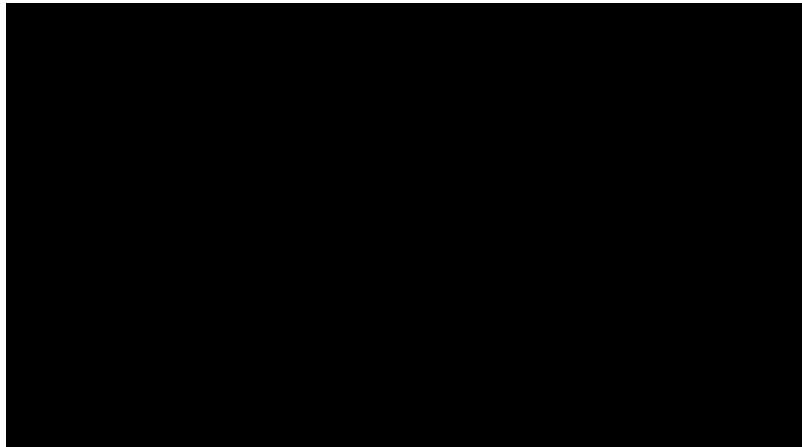
ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging
- [Your favorite area]

Reinforcement Learning

A machine learning technique that enables an agent (model) to learn in an interactive environment by trial and error using feedback from its own actions and experiences.



A reinforcement learning agent uses the data it generates itself

Agent \Rightarrow Environment : (experience, feedback) \Rightarrow Learning

Reinforcement Learning

RL = learning how to act (map situations to decisions), RL = Real Life

Task : Set of decisions for observed set of environment states

Goal : Learn how to take decisions (actions) in order to maximize reward



Some examples

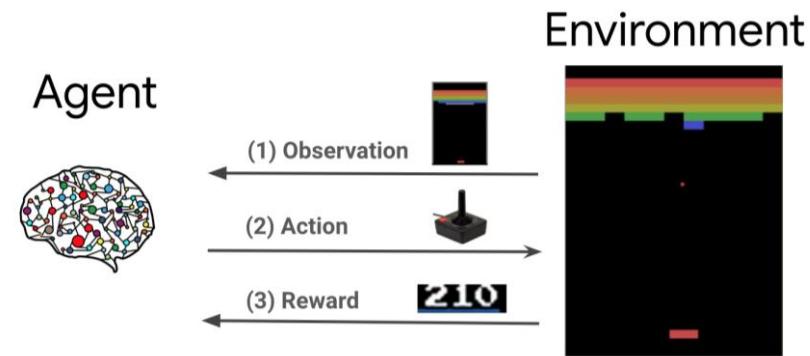
Atari Game

State : Raw pixel inputs of the game state

Action : Game controls e.g. Left, Right, Up, Down

Reward: Score increase/decrease at each time step

Objective : Learn an optimal function $(f : S \rightarrow A)$ that maximizes reward.



Some examples

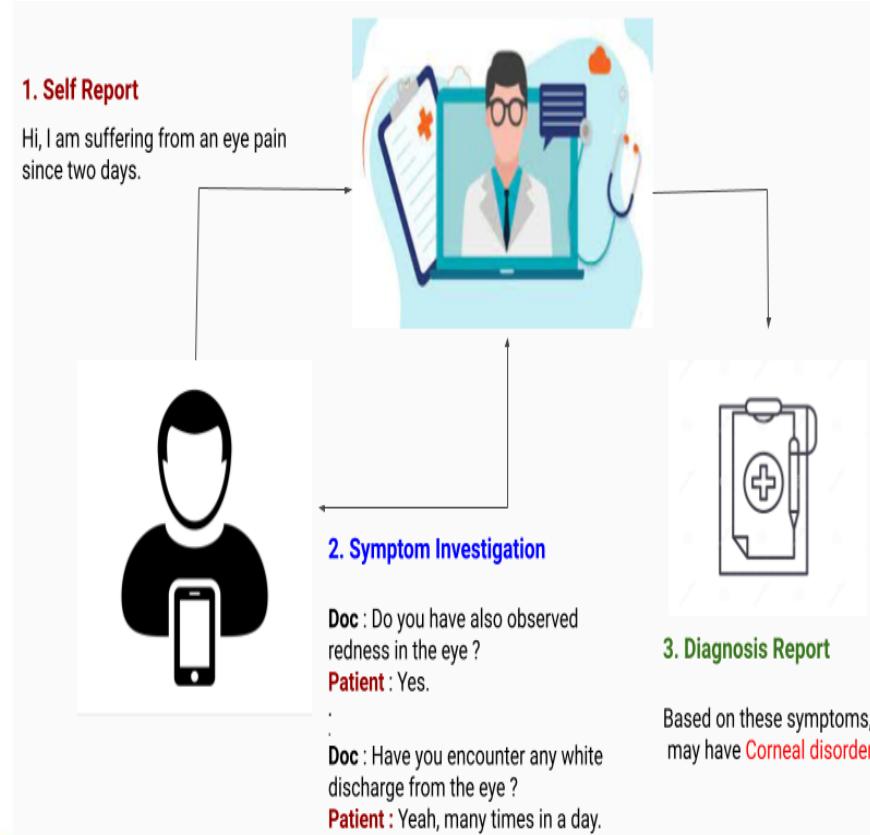
Symptom Investigation

State : Patient self report and other observed symptoms

Action : Inspect a new symptom from symptom space

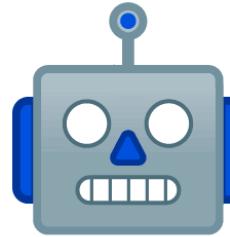
Reward: If it request a relevant symptom / diagnose patient correctly.

Objective : Learn an optimal symptom investigation function that collects adequate set of symptoms for accurate diagnosis



Key Elements of Reinforcement Learning

- Agent
- Environment
- Action
- Observation / State
- Reward



Agent



Environment

Gulcehre et al., 2021, RL Unplugged

Policy (π) : A function that maps observation to an action i.e., $S \Rightarrow A$

Steps for Reinforcement Learning

- I. The agent observes an input state

$$S_t = \text{Statetracker}(\text{Environment})$$

- I. An action is determined by a decision making function (policy)

$$a_t = \text{policy}(S_t)$$

- I. The action is performed

$$S_{t+1} = \text{Environment}(a_t)$$

- I. The agent receives a scalar reward or reinforcement from the environment

$$r_t = \text{Environment}(a_t, S_t)$$

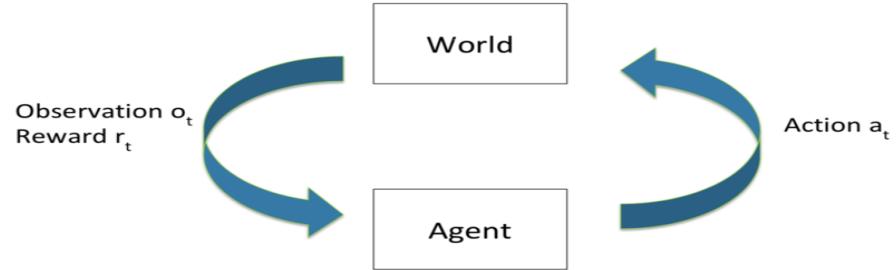
- I. Information about the reward given for that state / action pair is recorded

$$e_t = (S_t, a_t, S_{t+1}, r_t, \text{done})$$

* done : 1 (terminal state) and 0 (non terminal state)

Characteristics of Reinforcement Learning

- Sequential Decision Problem



- Exploration and Exploitation

Information about world (Exploration) and Greedy behavior as per gained Information (Exploitation)

- Reward Hypothesis

All goals can be described by the maximisation of expected cumulative reward

Exploration Exploitation Trade-Off

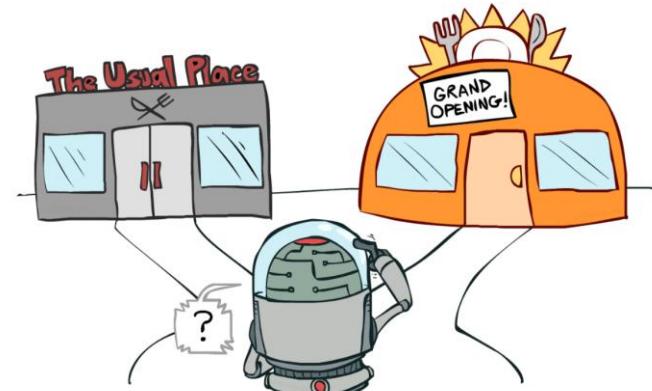
- Exploration finds more information about the environment
- Exploitation exploits known information to maximise reward
- Both Important : Exploration and Exploitation

Examples :

- 1) Restaurant

Exploitation Go to your favourite restaurant

Exploration Try a new restaurant



Reward

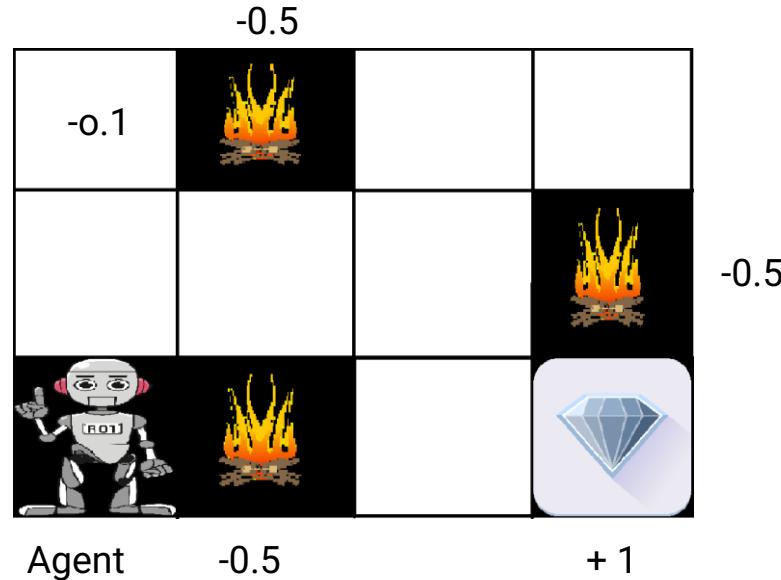
- Can be an arbitrary number e.g. -1, 0.45
- Set by the environment or algorithm designer

Agent action is determined by cumulative reward

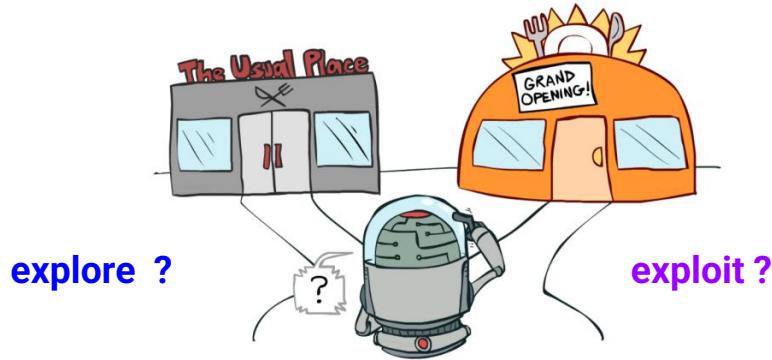
Return of a state

$$G_t = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \dots \quad \sum_{k \geq 0} \gamma^t \cdot r_{t+k+1}$$

- $\gamma = 0$ means you only care about reward now. 1 means we care about the future the same as now.



Exploration Exploitation - ϵ greedy



Action at time(t)

$$\left\{ \begin{array}{ll} \max Q_t(a) & \text{with probability } 1-\epsilon \\ \text{any action } (a) & \text{with probability } \epsilon \end{array} \right.$$

How can we mathematically formalize the RL problem?

Markov Decision Process (MDP)

- “The future is independent of the past given the present”
- **Markov Assumption** : Current state completely characterizes the state of the environment and it is sufficient to select an appropriate action.

$$P(A | S_t) = P(A | S_t, S_{t-1}, S_{t-2}, S_{t-3}, \dots, S_0)$$

A MDP is defined by (S, A, R, P, γ)

S : Set of possible states (State space)

A : Set of possible actions (Action space)

R : Reward function $(S \times A \rightarrow R)$

P : Transition probability, i.e., distribution over next state given (state, action) $P = (S_{t+1} | S_t, a_t)$

γ : Discount factor $(0,1)$



How can we mathematically formalize the RL problem?

Markov Decision Process (MDP)

Policy (π) : A policy is a function from S to A that specifies what action to take in each state

$$a = \pi(S)$$

Optimal Policy (π^*) : A policy that always takes best action for a given state, leading to gain of highest possible reward from the state.

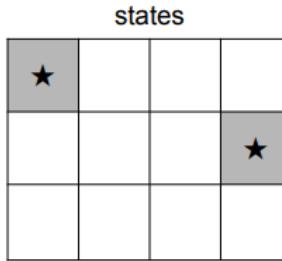
$$a^* = \pi^*(S)$$

Objective : find policy π^* that maximizes cumulative discounted reward:

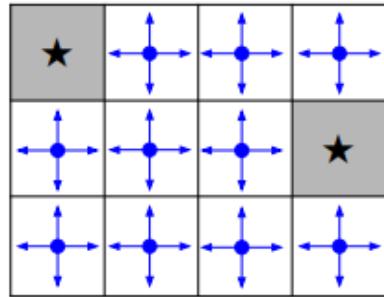
$$\pi^* = \arg \max_{\pi} [\sum_{t \geq 0} \gamma^t * r_t | \pi]$$

A simple MDP: Grid World

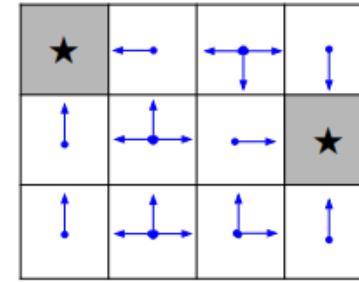
actions = {
1. right →
2. left ←
3. up ↑
4. down ↓
}



Set a negative “reward”
for each transition
(e.g. $r = -1$)



Random Policy



Optimal Policy

Value function and Q-Value function

RL Trajectory : $s_0, a_0, r_0, s_1, a_1, r_1, \dots$

- How good is a state?

Value function : Expected cumulative reward from following the policy from state s

$$V^\pi(s) = E [\sum_{t \geq 0} \gamma^t * r_t \mid S_0 = s, \pi]$$

- How good is a state-action pair?

Q Value function : The Q-value function at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy:

$$Q^\pi(s, a) = E [\sum_{t \geq 0} \gamma^t * r_t \mid S_0 = s, a_0 = a, \pi]$$

Policy

A policy (Π) is a function that maps state to action

$$a = \Pi(S)$$

- A policy is the agent's behavior

$$a_t = \operatorname{argmax}_i P(A_i | S_t = S, \Pi_\Theta); \quad A : \text{Action Space}$$

- Stochastic Policy

$$a = \Pi(S)$$

$$= P(A_t = a | S_t = S)$$

Objective : Learn a policy that can serve the underlying task appropriately; e.g: Ticket reservation

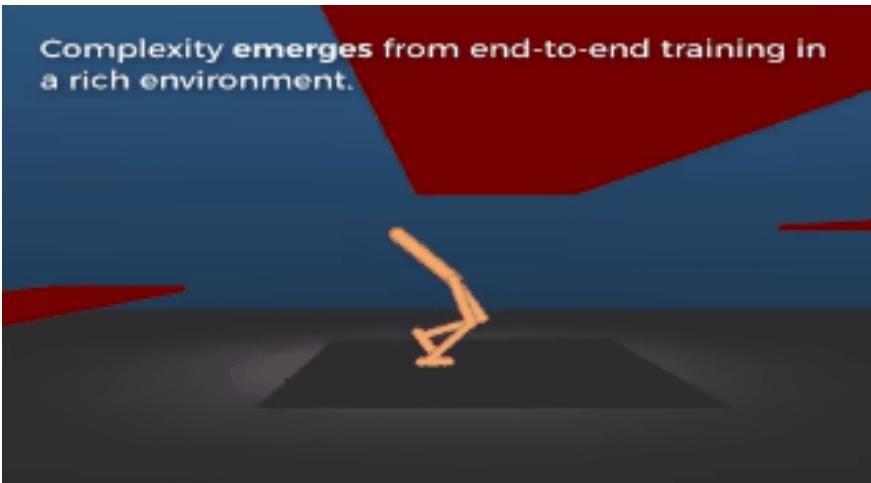
How to learn a Policy ?

$$\begin{aligned} a_{t+1} &= \Pi_\Theta(S_t) \\ &= \operatorname{argmax}_i Q(S_t, A_i | \Pi_\Theta) \\ &= \operatorname{argmax}_i (r_t + \operatorname{argmax}_{a_{(t+2)}} Q_{a_{(t+2)}}(S_{t+1}, a_{t+2})) \\ &= \operatorname{argmax}_i (r_t + V(S_{t+1})) \end{aligned}$$

Task : Learn a Q state - action value / State value function

- Tabular Q learning
- Deep Reinforcement Learning

Applications



DeepMind 2017: Locomotion End-to-End



OpenAI 2017: Self-Play

Applications



Google 2016: Large-scale Interaction

Agent observation raw pixels



Multi-agent collaboration, DeepMind 2018: Quake III

Agents can play with human teammates

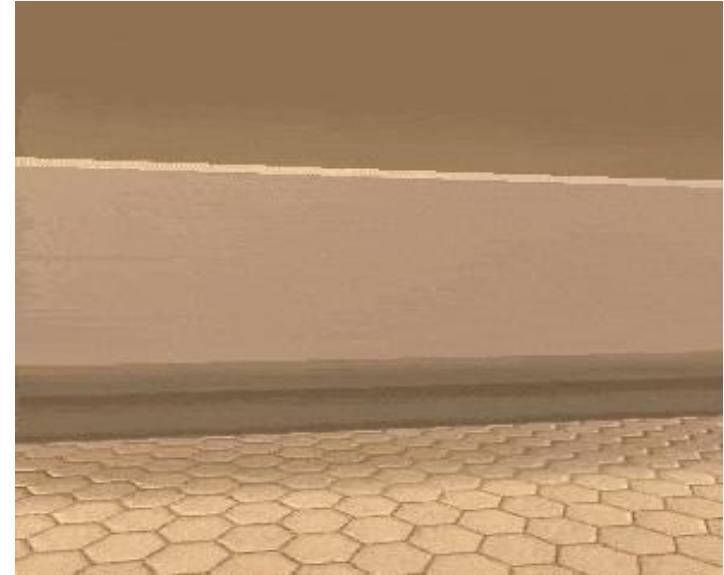


Indoor map overview

Applications



Chaplot et al 2017: Task-Orientated Natural Language Grounding



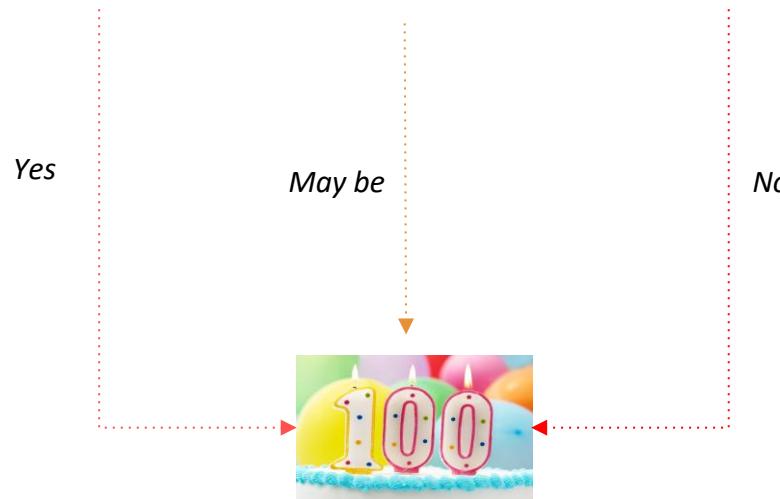
M Tank 2019: <https://github.com/TheMTank/cups-rl/>

*Brush up of
Probability
Concepts*

Probability



How likely is it that you will live more than 100 years?



The Oldest Living Man On Earth
Swami Sivananda (125 years +)

What is the amount of certainty?

Probability

“Chance of occurrence of an event”

Will it rain Today / Should I go with an umbrella ?

No

Why ?

$T = 10^\circ \text{ C}$, Humidity < K, ...

With these conditions, only one time, it rained in last five years

Probability

Randomness is fundamental to the universe

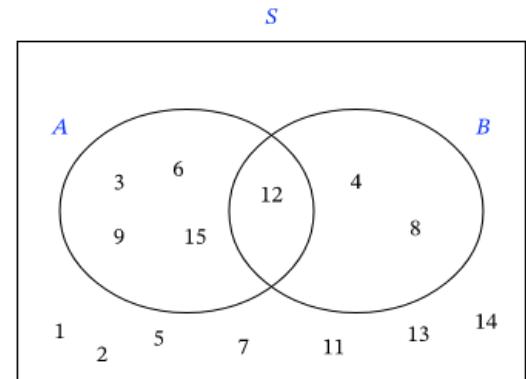
Can you ever be sure about the outcome of tossing a coin / throwing a dice ?

- Probability theory is a mathematical framework for quantifying our uncertainty about the world
- The ratio of the number of favorable outcomes to the total number of outcomes of an event

$$\text{Probability}(e) = n \text{ (Favorable Outcomes)} / n \text{ (Total Outcomes)}$$

Key Terms

- **Experiment :** A trial or an operation conducted to produce an outcome is called an experiment.
- **Sample Space (Ω) :** Ω is the space of possible outcomes
- **Event (e) :** A subset of the sample space
- **Exhaustive Event :** if at least one of the events must occur at any time



Why Probability

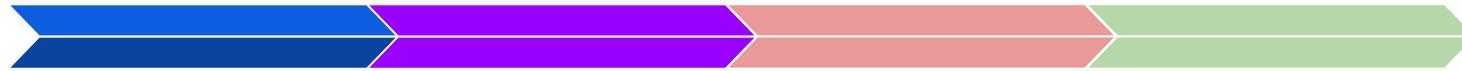


Table Lookup

Rule based Systems

Statistical Machine Learning

Neural Network

Task : Sentiment Classification

Sentence : I am glad to get the opportunity.

Table lookup and Rule based System

Table Lookup

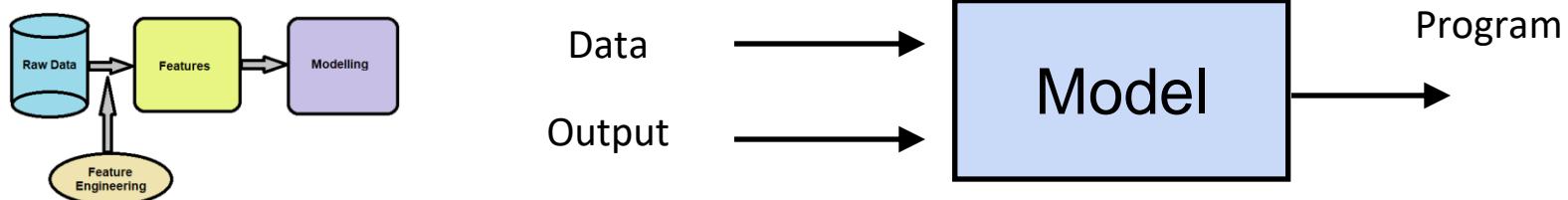
- Searching - existing knowledge base
- Always fails if it does find an exact match
- Very hard to scale up

Rule based System

- Decision according to set of rules
- If a sentence contains 'good' \Rightarrow Positive
- Huge no. of rules but still fails to achieve a nominal performance

Learn from Data

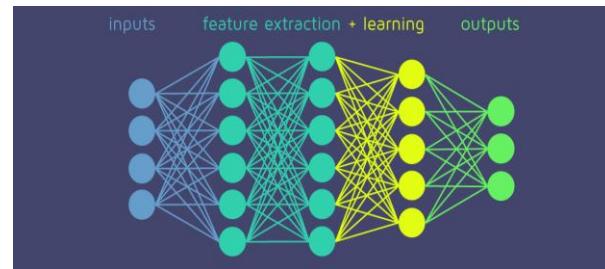
Data + Classifier > Human decision maker



But depends on human engineered features !!!

Deep Learning

End to end systems; essential properties learnt at intermediate layers



Machine Learning and Probability

- Machine Learning \Rightarrow Model based on data (Previous experience)

[Decision based data]

Will it rain today = $1 / K$

[Only one time out of K days, it rained with the given conditions]

- Prediction needs to have associated confidence

confidence = probability

An Example

A six-sided die is rolled twice. What is the probability that the sum of the numbers is at least 10?

Sample Space (S) : (1, 1) (1, 2) (6, 4), (6, 5), (6, 6)

Favorable Events : (4,6), (5,5), (6,4), (5,6), (6,5) or (6,6)

$$\text{Required Probability} = n(\text{Sum} \geq 10) / n(S)$$

$$= 6/36$$

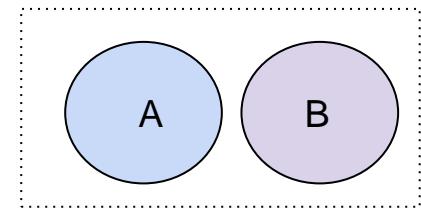
$$= 1/6$$

Axioms of Probability

1. For any event A, we have $P(A) \geq 0$
1. $P(S) = 1$
1. If the events A_1, A_2, \dots are pairwise disjoint, then

$$P(A_1 \cup A_2 \cup \dots) = P(A_1) + P(A_2) + \dots$$

1. $P(A') = 1 - P(A)$
1. If A and B are two events in an experiment such that $A \subset B$, then $P(B-A) = P(B) - P(A)$

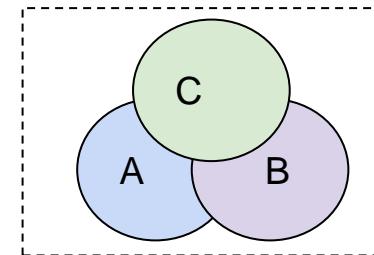
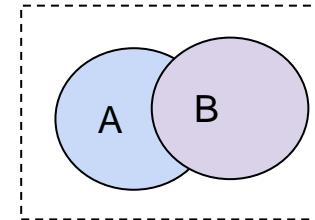


Some Important Propositions and Corollaries

- If the sample space S is finite, say $S = \{a_1, \dots, a_n\}$, then

$$P(a_1) + P(a_2) + \dots + P(a_n) = 1.$$

- $P(A) = 1 - P(\bar{A})$ for any event A
- $P(A) \leq 1$ for any event A
- If $A \subseteq B$, then $P(A) \leq P(B)$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A \cap B) - P(A \cap C) - P(B \cap C) + P(A \cap B \cap C)$



Some Important Propositions and Corollaries

- ❖ Two events, A and B are independent

Occurrence of one event does not affect the chances of the other.

$P(A) = x$, whatever be $P(B)$ and vice-versa

$$P(A \cap B) = P(A) \cdot P(B)$$

- ❖ Chain Rule

$$P(A \cap B) = P(A|B) \cdot P(B)$$

$$P(A \cap B \cap C) = P(A|B \cap C) \cdot P(B|C) \cdot P(C)$$

An Example

Consider an example of rolling a die. If A is the event ‘the number appearing is odd’ and B be the event ‘the number appearing is a multiple of 3’, Verify whether these two events are independent.

Solution :

Sample Space = {1, 2 , 3 , 4 , 5, 6}

A = {1, 3, 5}

B = {3, 6}

$P(A) = 3/6 = 1/2$ and $P(B) = 2/6 = 1/3$

$P(A | B) = P(A \cap B) / P(B) = 1/6 / 1/3 = 1/2$

$\Rightarrow P(A | B) = P(A); \Rightarrow P(A \cap B) = P(A). P(B)$

which implies that the occurrence of event B has not affected the probability of occurrence of the event A .

Some Important Propositions and Corollaries

Distributive laws:

$$(A \cap B) \cup C = (A \cup C) \cap (B \cup C) \text{ and } (A \cup B) \cap C = (A \cap C) \cup (B \cap C)$$

De Morgan's Laws:

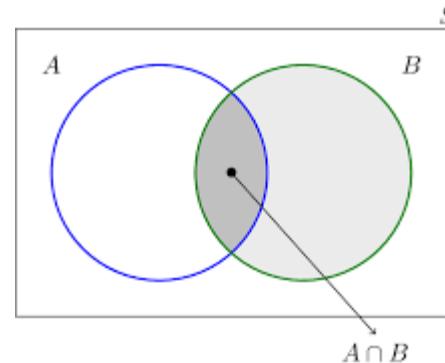
$$\text{i. } (A \cup B)' = A' \cap B'$$

$$\text{ii. } (A \cap B)' = A' \cup B'$$

Conditional Probability

“ Probability of an event (A) given the knowledge that an event B has already occurred”

$$P(A/B) = P(A \cap B) / P(B)$$



Sentiment of an utterance (message) Vs Sentiment of an utterance (message) given its emotion information

An Example

2 blue and 3 red marbles are in a bag.

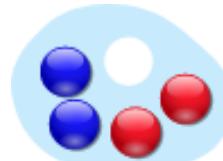
What are the chances of getting a blue marble?

the chance is $2 / 5$

But after taking one out the chances change!

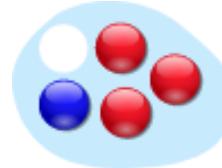
So the next time:

if we got a red marble before

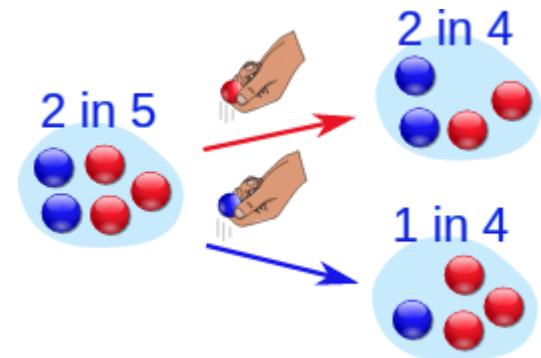


The chance of a blue marble next is $2/4$

if we got a blue marble before



The chance of a blue marble next is $1/4$



An example

In a card game, suppose a player needs to draw two cards of the same suit in order to win. Of the 52 cards, there are 13 cards in each suit. Suppose first the player draws a heart. Now the player wishes to draw a second heart. what is the probability of picking a second heart ?

Since one heart has already been chosen, there are now 12 hearts remaining in a deck of 51 cards. So the conditional probability $P(\text{Draw second heart}|\text{First card a heart}) = 12/51$

Bayes' Theorem



"Probability is orderly opinion ... inference from data is nothing other than revision of such opinion in the light of relevant new information" - Thomas Bayes

Bayes' theorem describes probability of an event based on the condition of occurrence of other events.

$$P(h | D) = P(D | h) P(h) / P(D)$$

Where $P(h)$ = prior probability

$P(D | h)$ = likelihood $P(D)$ = evidence

$P(h | D)$ = posterior probability = Likelihood * Prior / Evidence

An Example

A man is known to speak the truth 2 out of 3 times. He throws a die and reports that the number obtained is a four. Find the probability that the number obtained is actually a four.

Solution

Let

A : the event that the man reports that number four is obtained.

E1 : the event that four is obtained

E2 : the event that four is not obtained

$$P(E1) = \frac{1}{6}, P(E2) = 1 - \frac{1}{6} = \frac{5}{6}$$

$P(A | E1)$: Probability that man reports four and it is actually a four = $\frac{2}{3}$

$P(A | E2)$: Probability that man reports four and it is not a four = $\frac{1}{3}$

By using Bayes' theorem, probability that number obtained is actually a four,

$$\begin{aligned} P(E1 | A) &= P(E1) \cdot P(A | E1) / P(A) = P(E1) \cdot P(A|E1) / \{ P(E1) \cdot P(A|E1) + P(E2) \cdot P(A|E2) \} \\ &= (\frac{1}{6} \cdot \frac{2}{3}) / \{(\frac{1}{6} \cdot \frac{2}{3}) + (\frac{5}{6} \cdot \frac{1}{3})\} \\ &= 2/7 \end{aligned}$$

Bayes' Theorem

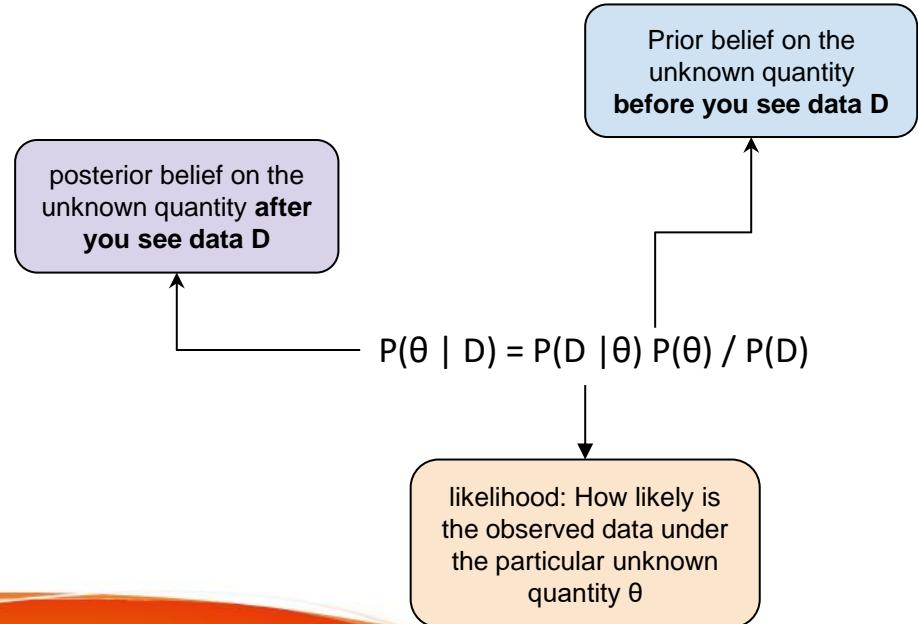
- Difference between frequentism and Bayesianism : Frequentism is based on frequencies of events, Bayesianism is based on our knowledge of events.
- There are plenty of applications of the Bayes' Theorem in the real world
 - Naive Bayes' Classifiers
 - Bayesian Parameter Estimation

Bayes Rule in Machine Learning

D: data (evidence)

θ : unknown quantities

- e.g., model parameters



An Example

Suppose 15 men out of 300 men and 25 women out of 1000 are good orators. An orator is chosen at random. Find the probability that a male person is selected. Assume that there are equal numbers of men and women?

Solution :

Assume

There be 1000 men and 1000 women

E1 and E2 be the events of choosing a man and a woman respectively

$$P(E1) = 1000/2000 = 1/2, \text{ and } P(E2) = 1000/2000 = 1/2$$

Let E be the event of choosing an orator

$$P(E|E1) = 50/1000 = 1/20, \text{ and } P(E|E2) = 25/1000 = 1/40 \quad [\because 300 \text{ men} \rightarrow 15 \text{ op}, \therefore 1000 \text{ men} \rightarrow 50 \text{ op}]$$

Probability of selecting a male person given that the person selected is a good orator

$$\begin{aligned} P(E1 | E) &= P(E|E1) * P(E1) / P(E|E1) * P(E1) + P(E|E2) * P(E2) \\ &= (1/2 * 1/20) / \{(1/2 * 1/20) + (1/2 * 1/40)\} \\ &= 2/3 \end{aligned}$$

Random Variables

A random variable

- denotes something about which we are uncertain
- perhaps the outcome of a randomized experiment

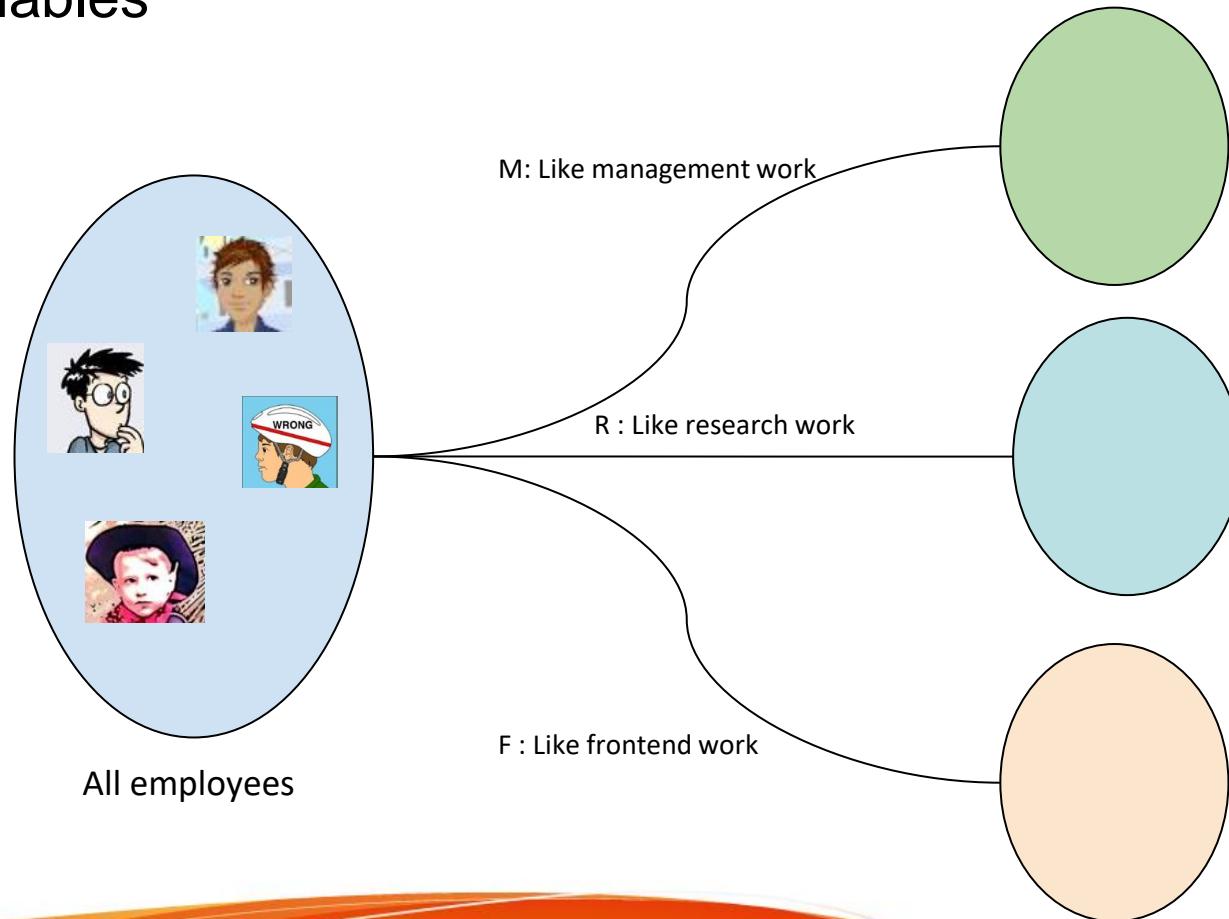
Examples

A = True, if a randomly drawn person from the class is Bengali.

B = Two randomly chosen students from the class have same no. of characters in their surnames

$P(X)$: the fraction of possible worlds in which X is true

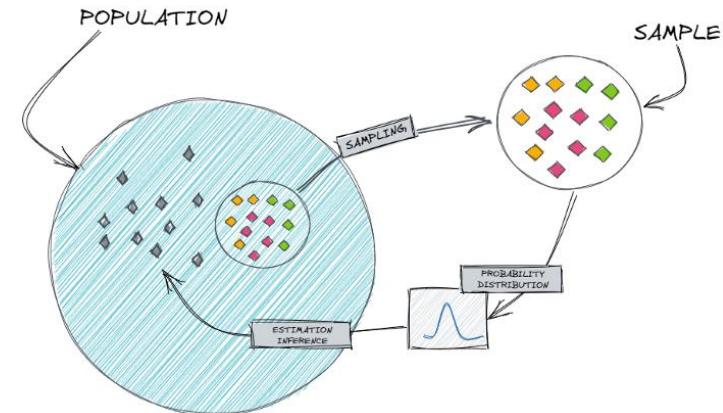
Random Variables



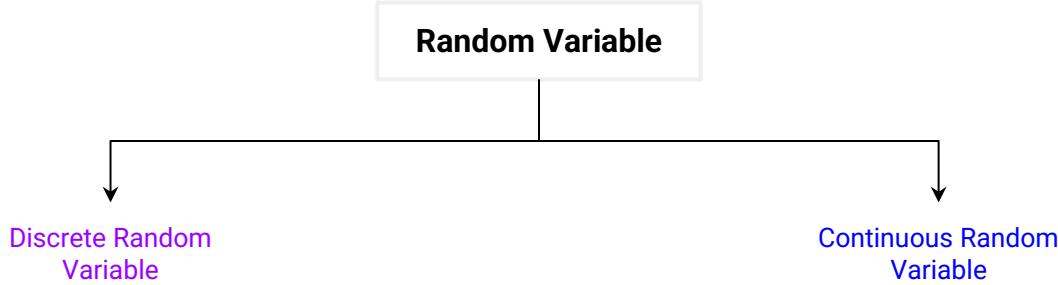
Probability Distributions

To describe the likelihood of each possible value of a random variable x , $x \sim P(x)$ to indicate that x is a random variable which is drawn from a probability distribution $P(x)$.

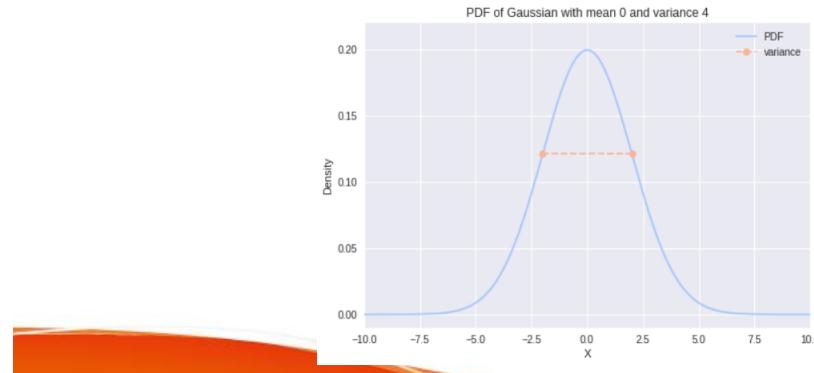
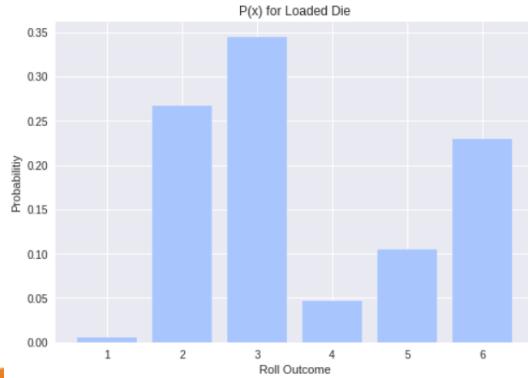
- $P : A \rightarrow R$, It maps an input space A to the sample space to a real number, namely the probability
- To calculate confidence intervals for parameters and to calculate critical regions for hypothesis tests.



Types of Random Variables



- Sample space : Discrete set
- PMF : Mapping of each value in the variable's sample space to a probability.
- Sample space : Infinite set
- PDF : Mapping of each value in the variable's sample space to a probability.



Probability Mass Function (PMF)

- Probability distribution over a discrete random variable.
- $P : A \rightarrow [0, 1]$

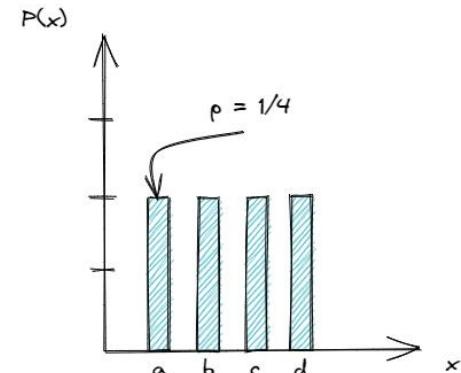
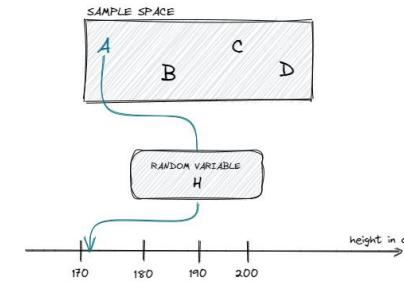
Some PMFs

- Binomial Distribution

$$f(k; p) = p^k (1 - p)^{1-k} \quad \text{for } k \in \{0, 1\}$$

- Binomial Distribution

$$f(k; n, p) = \binom{n}{k} p^k (1 - p)^{n-k}$$



Probability Density Function (PDF)

- ❖ Probability distribution over a continuous random variable
- ❖ it maps an infinite sample space to relative likelihood values
- ❖ Characteristics

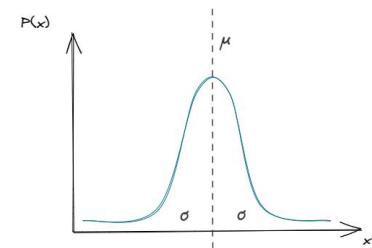
$$(1) \quad p(x) \geq 0$$

$$(2) \quad \int p(x) \delta x = 1$$

Some PDFs

Gaussian Distribution

$$\mathcal{N}(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right)$$



Standard Normal Distribution

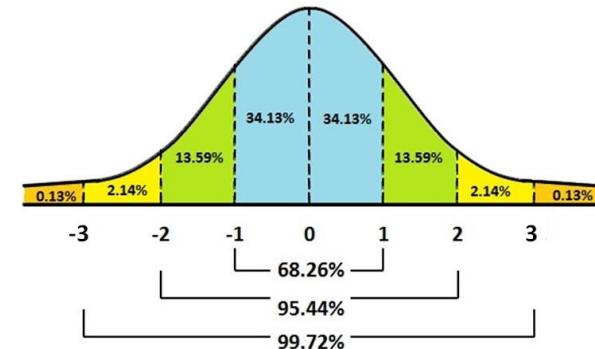
Normal Distribution

$$y = \frac{1}{\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

The simplest case of the normal distribution, known as the Standard Normal Distribution $\mu = 0, \sigma = 1$

$$f(Z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

Where
 $-\infty < z < \infty$



The total area under the normal curve is equal to 1.

Some Important Distributions

Distribution	PDF or PMF	Mean	Variance
$Bernoulli(p)$	$\begin{cases} p, & \text{if } x = 1 \\ 1 - p, & \text{if } x = 0. \end{cases}$	p	$p(1 - p)$
$Binomial(n, p)$	$\binom{n}{k} p^k (1 - p)^{n-k} \text{ for } 0 \leq k \leq n$	np	npq
$Geometric(p)$	$p(1 - p)^{k-1} \text{ for } k = 1, 2, \dots$	$\frac{1}{p}$	$\frac{1-p}{p^2}$
$Poisson(\lambda)$	$e^{-\lambda} \lambda^x / x! \text{ for } k = 1, 2, \dots$	λ	λ
$Uniform(a, b)$	$\frac{1}{b-a} \quad \forall x \in (a, b)$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
$Gaussian(\mu, \sigma^2)$	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	μ	σ^2
$Exponential(\lambda)$	$\lambda e^{-\lambda x} \quad x \geq 0, \lambda > 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$

Expectation

- Weighted average of the possible values that X can take, each value being weighted according to the probability of that event occurring.

for discrete random variable

$$E(X) = \sum v_i P(X = v_i)$$

for continuous random variable

$$E(X) = \int_{-\infty}^{+\infty} xf(x) dx$$

Some Properties

$$1. \quad E(X + Y) = E(X) + E(Y)$$

$$1. \quad E(aX) = a E(X)$$

$$1. \quad E(XY) = E(X).E(Y)$$

$$1. \quad E(aX + b) = a E(X) + b$$

An Example

Determine the mean and variance of the random variable X having the following probability distribution.

$X = x$	1	2	3	4	5	6	7	8	9	10
$P(x)$	0.15	0.10	0.10	0.01	0.08	0.01	0.05	0.02	0.28	0.20

Solution

$$\begin{aligned}\text{Mean of the random variable } X &= E(X) = \sum_x x P_X(x) \\ &= (1 \times 0.15) + (2 \times 0.10) + (3 \times 0.10) + (4 \times 0.01) + (5 \times 0.08) + (6 \times 0.01) + \\ &\quad (7 \times 0.05) + (8 \times 0.02) + (9 \times 0.28) + (10 \times 0.20)\end{aligned}$$

$$E(X) = 6.56$$

$$\begin{aligned}E(X^2) &= \sum_x x^2 P_X(x) \\ &= (1^2 \times 0.15) + (2^2 \times 0.10) + (3^2 \times 0.10) + (4^2 \times 0.01) + \\ &\quad (5^2 \times 0.08) + (6^2 \times 0.01) + (7^2 \times 0.05) + (8^2 \times 0.02) + \\ &\quad (9^2 \times 0.28) + (10^2 \times 0.20). \\ &= 50.38\end{aligned}$$

$$\begin{aligned}\text{Variance of the Random Variable } X &= V(X) = E(X^2) - [E(X)]^2 \\ &= 50.38 - (6.56)^2 \\ &= 7.35\end{aligned}$$

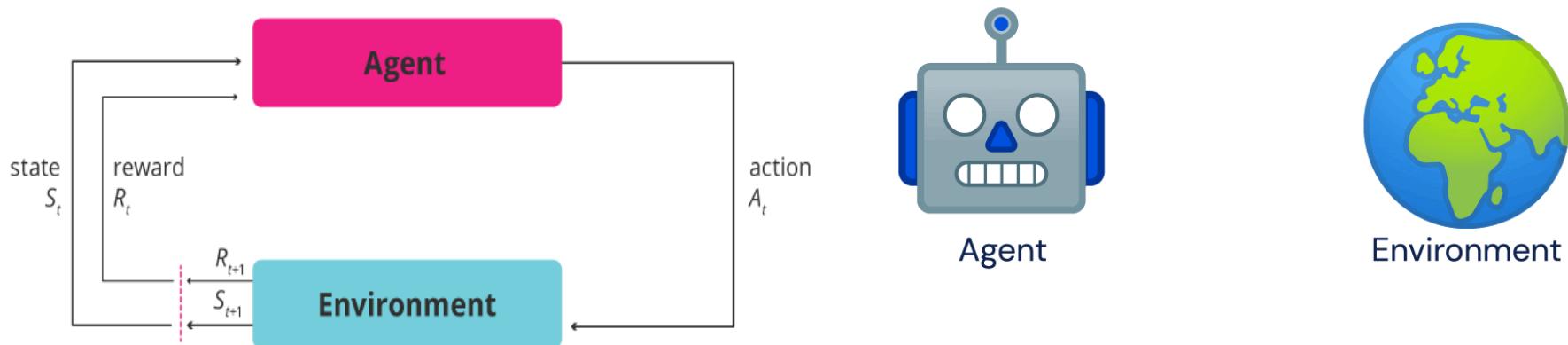
Markov Decision Process (MDP)

Quick Recap

RL = learning how to act (map situations to decisions), RL = Real Life

Task : Set of decisions for observed set of environment states

Goal : Learn how to take decisions (actions) in order to maximize reward



Gulcehre et al., 2021, RL Unplugged

Markov Decision Process (MDP)

“The future is independent of the past given the present”



Markov Assumption : Current state completely characterises the state of the environment and it is sufficient to select an appropriate action.

$$P(A | S_t) = P(A | S_t, S_{t-1}, S_{t-2}, S_{t-3}, \dots, S_0)$$



Andrey Markov
(1856-1922)

Markov Decision Process (MDP)

A MDP is defined by (S, A, R, P, γ)

S : Set of possible states (State space)

A : Set of possible actions (Action space)

R : Reward function $(S \times A \rightarrow R)$

P : Transition probability, i.e., distribution over next state given (state, action) :

$P(S_{t+1} | S_t, a_t)$

γ : Discount factor $(0,1)$

MDP vs HMM

- Like an HMM, we have States s , beginning with initial state s_0
- Unlike an HMM, we also have Actions a
 - Each state s has actions $A(s)$ available from it
- Unlike an HMM, the Transition model $P(s' | s, a)$ depends on both the state you are in, and the action you perform

Some examples

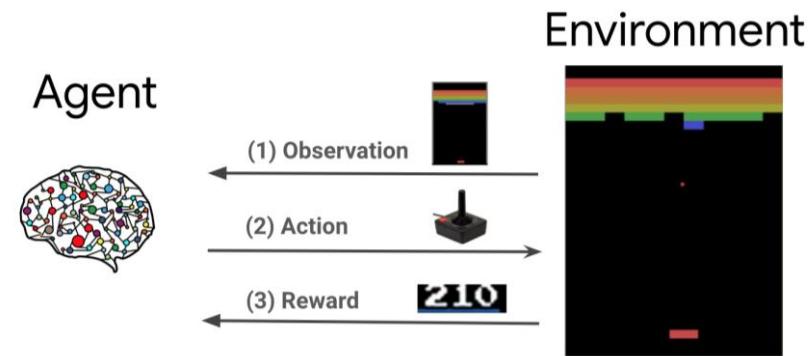
Atari Game

State : Raw pixel inputs of the game state

Action : Game controls e.g. Left, Right, Up, Down

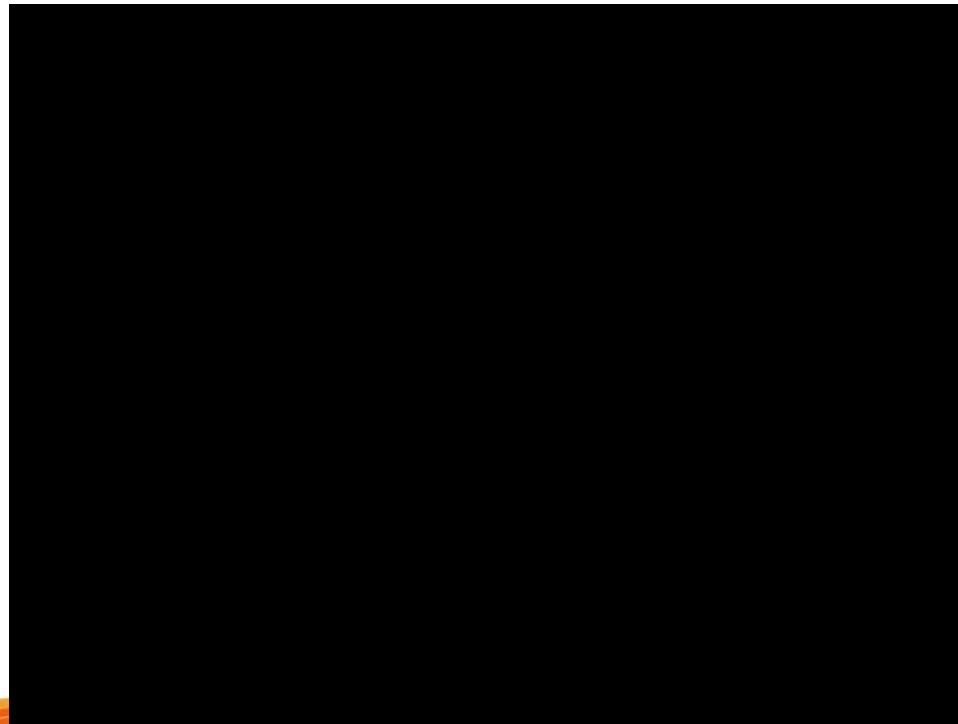
Reward: Score increase/decrease at each time step

Objective : Learn an optimal function $(f : S \rightarrow A)$ that maximizes reward.



Some examples

Atari Game

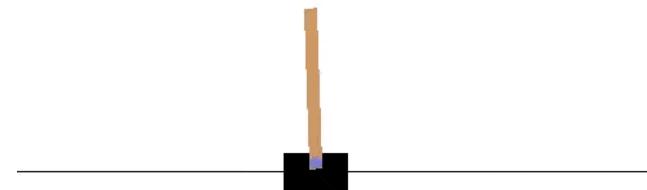


Some examples

CartPole-v1

State : (position, velocity, angular position, angular velocity)

Action : +1 and -1, which translates to a movement of either left or right

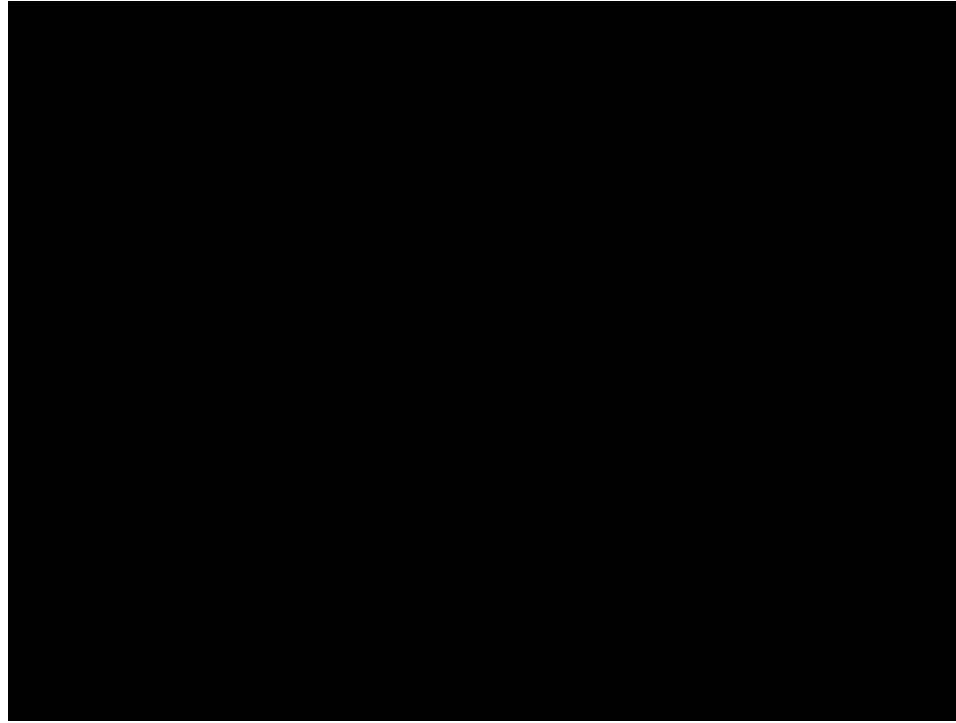


Reward: +1 is provided for every timestep if pole remains upright

Objective : The pendulum starts upright, and the goal is to prevent it from falling over.

Some examples

CartPole-v1



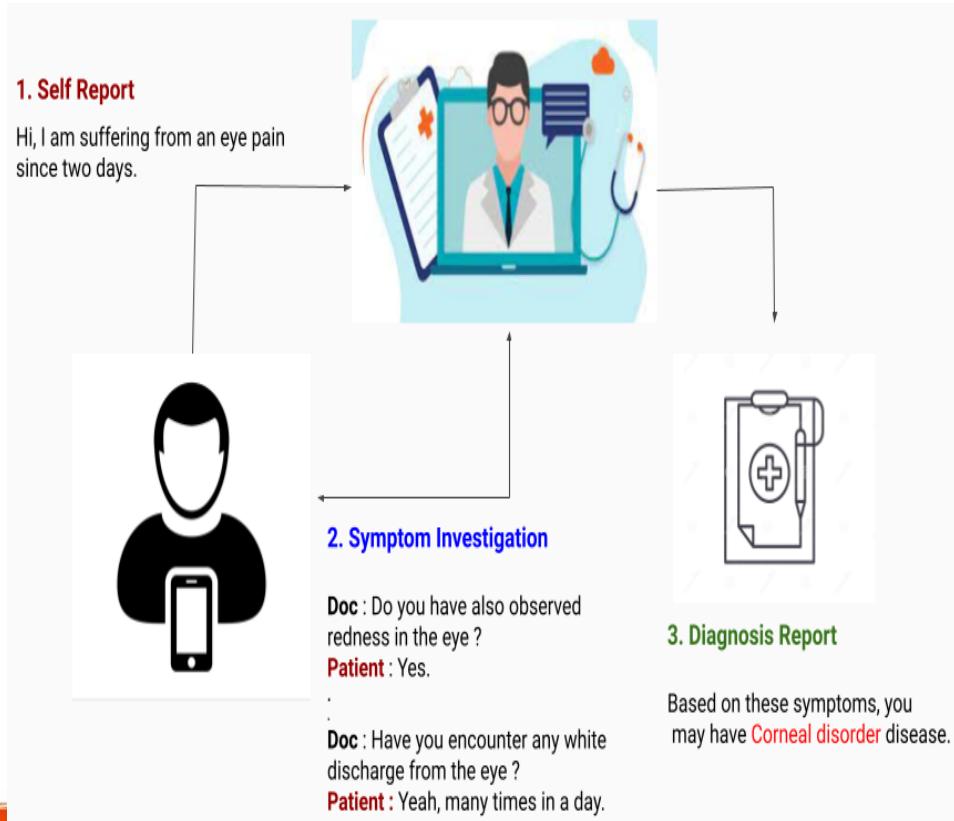
Some examples

State : Patient self report and other observed symptoms

Action : Inspect a new symptom from symptom space

Reward: If it requests a relevant symptom / diagnose patient correctly.

Objective : Learn an optimal symptom investigation function that collects adequate set of symptoms for accurate diagnosis



Chess

State : Current pixel

Action : Valid move

Reward: +1(win), -1(lose), 0(draw)

> **Delayed Reward** : reward only for actually winning, not for achieving subgoals, e.g., taking opponent's pieces or gaining control of the center of board

Objective : win the game



Reward : What Sutton & Barto wrote in 1998:

“... the reward signal is not the place to impart to the agent prior knowledge about how to achieve what we want it to do. For example, a chess-playing agent should be rewarded only for actually winning, not for achieving subgoals such taking its opponent's pieces or gaining control of the center of the board. If achieving these sorts of subgoals were rewarded, then the agent might find a way to achieve them without achieving the real goal. For example, it might find a way to take the opponent's pieces even at the cost of losing the game. The reward signal is your way of communicating to the robot what you want it to achieve, not how you want it achieved.”

Mountain-Car

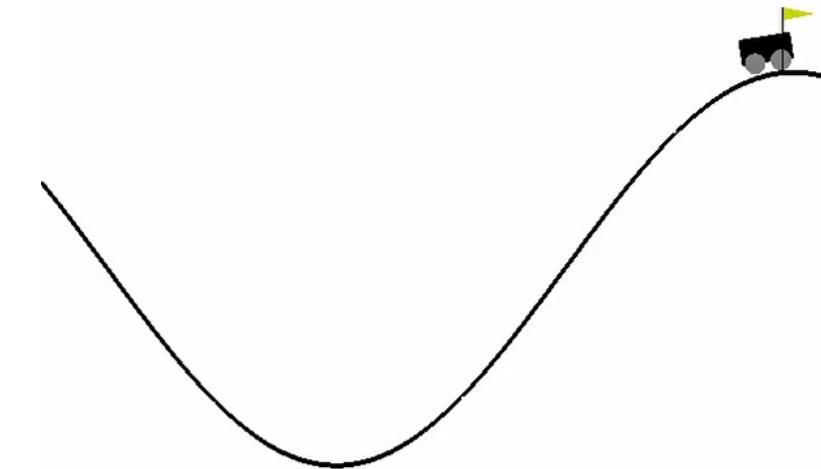
State : Position and velocity

Action : accelerate forward, accelerate backward, do nothing

Reward:

- reward = -1 for every time step, until car reaches the top
- reward = 1 at the top, 0 otherwise $\gamma < 1$

Objective : Goal



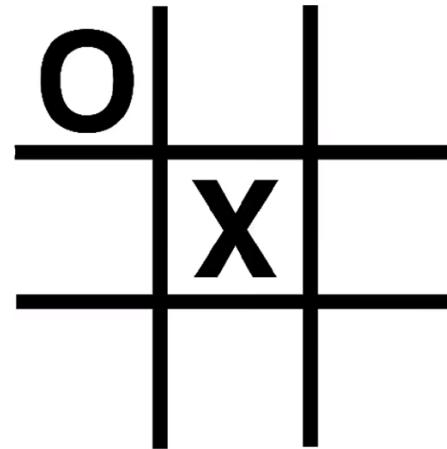
Tic tac toe

State : positions of X's and O's on the board

Action : move for X

Reward: win/lose/tie the game (+1 / -1/0)

Objective : Win the game



Reward Hypothesis

- Agent goal: maximize cumulative reward
- **Hypothesis:** All goals can be described by the maximization of expected cumulative reward
- Examples
 - Fly stunt maneuvers in a helicopter:
 - +ve reward for following desired trajectory
 - -ve reward for crashing
 - Backgammon
 - +/-ve reward for winning/losing a game
 - Make a humanoid robot walk:
 - +ve reward for forward motion
 - -ve reward for falling over
 - Play many different Atari games:
 - +/-ve reward for increasing/decreasing score

Policy

Policy (π) : A policy is a function from S to A that specifies what action to take in each state

$$a = \pi(S)$$

Optimal Policy (π^*) : A policy that always takes best action for a given state, leading to gain of highest possible reward from the state.

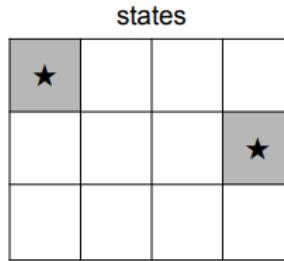
$$a^* = \pi^*(S)$$

Objective : find policy π^* that maximizes cumulative discounted reward:

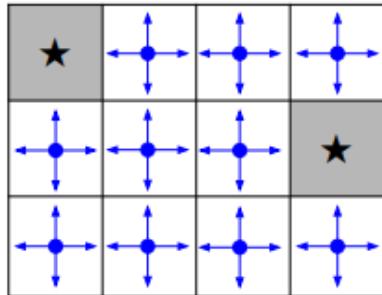
$$\pi^* = \arg \max_{\pi} [\sum_{t \geq 0} \gamma^t * r_t | \pi]$$

A simple MDP: Grid World

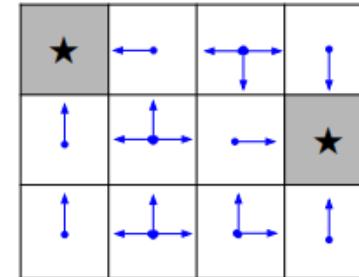
actions = {
1. right →
2. left ←
3. up ↑
4. down ↓
}



Set a negative “reward”
for each transition
(e.g. $r = -1$)



Random Policy



Optimal Policy

Dynamic of MDP

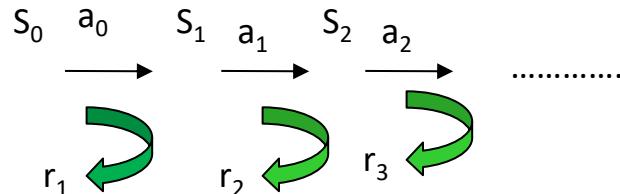
We start in some state s_0 , and get to choose some action $a_0 \in A$



The state of the MDP transitions to some successor state s_1 , drawn according to $s_1 \sim P(s_0, a_0)$



We choose another action a_1 as per the new observation state s_1

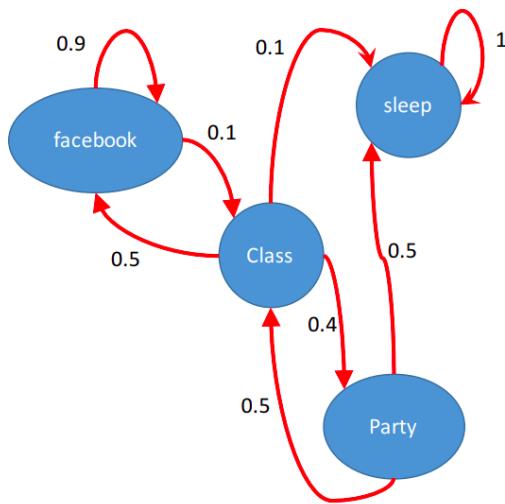


Objective : Our goal in reinforcement learning is to choose actions over time so as to maximize the expected value of the total payoff.

$$R(s_0, a_0) + \gamma \cdot R(s_1, a_1) + \gamma^2 \cdot R(s_2, a_2) + \dots$$

An Example

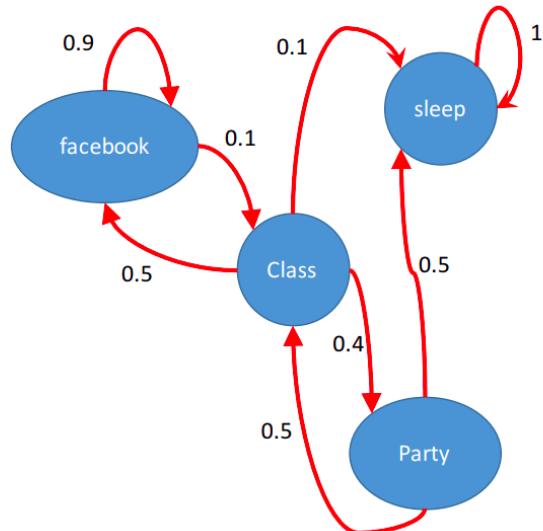
Markov Process: the student chain



	FB	C	P	S
FB	0.9	0.1		
C	0.5		0.4	0.1
P		0.5		0.5
S				1

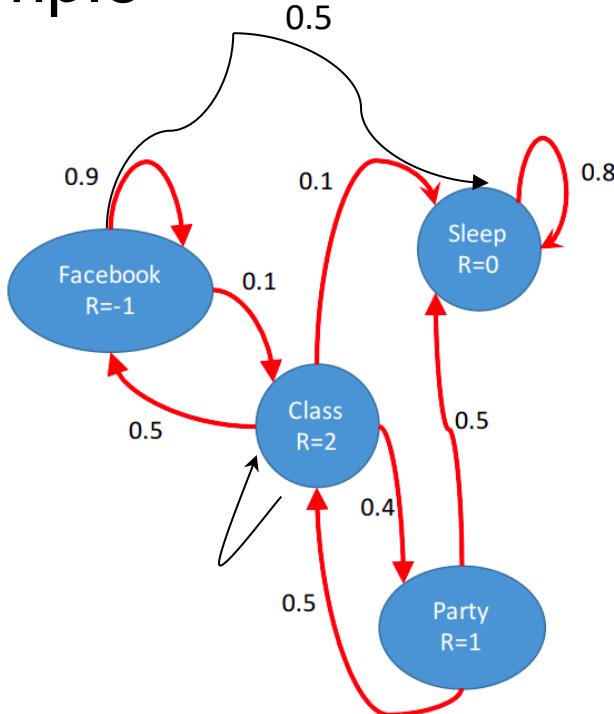
An Example

Example: the student chain



- Example of episodes (random walks):
 - C C Fb Fb C P S
 - C Fb Fb Fb Fb C P S

An Example



- Example of episode (random walks): discount factor = $\frac{1}{2}$

- C C Fb Fb S

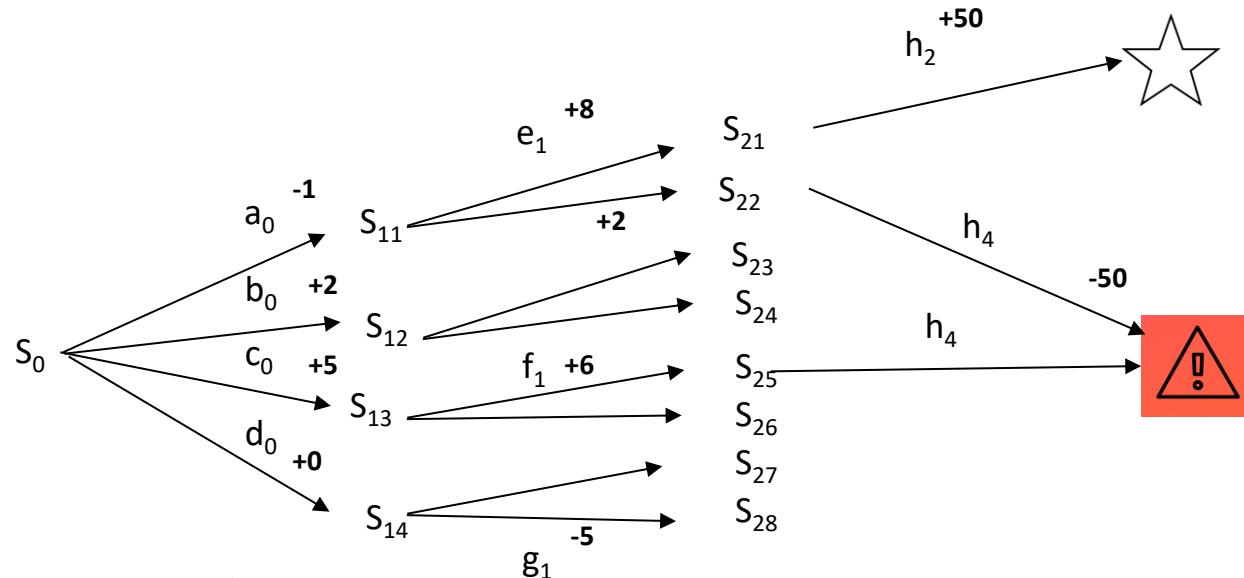
$$\begin{aligned} \text{total reward} &= \\ G_1 &= 2 + 2 * \frac{1}{2} + (-1) * \left(\frac{1}{2}\right)^2 + \\ &(-1) * 0.5^3 + 0 = 2 + 1 - \frac{1}{4} - \frac{1}{8} \\ &= 3 - 0.365 = 2.635 \end{aligned}$$

- C Fb Fb Fb Fb C P S

$$G_1 = ?$$

How to learn an Optimal Policy ?

Policy is nothing but a function that tells which action should be picked
 $a = \pi(S)$



$S_0 \Rightarrow a_0 \Rightarrow S_{11} \Rightarrow e_1 \Rightarrow S_{21} \Rightarrow h_2 \Rightarrow \star$

$$\text{gain} = -1 + 8 + 50 = 57$$

$S_0 \Rightarrow c_0 \Rightarrow S_{13} \Rightarrow f_1 \Rightarrow S_{25} \Rightarrow h_4 \Rightarrow \Delta$

$$\text{gain} = +5 + 6 - 50 = -39$$

Agent should care about Cumulative reward rather than immediate reward !

How to learn an Optimal Policy ?

$$\begin{aligned} a_{t+1} &= \Pi_\Theta(S_t) \\ &= \operatorname{argmax}_i Q(S_t, A_i | \Pi_\Theta) \\ &= \operatorname{argmax}_i (r_t + Q_{a_{(t+2)}}(S_{t+1}, a_{t+2})) \\ &= \operatorname{argmax}_i (r_t + V(S_{t+1})) \end{aligned}$$

Thus, we either need to learn an **Optimal Q function, $Q(S,a)$** or **State-Value function, $V(S)$**

Machine Learning Categories

Supervised learning	Unsupervised learning	Reinforcement learning
Input/Output pairs	Input only	Input & Critic
Learning phase vs. acting phase	learning phase v.s. acting phase	learning and acting simultaneously—online learning
		Explicitly explore the world and learn by trial and error

What is reinforcement learning

An autonomous agent should learn to choose optimal actions in each state to achieve its goals

The agent learns how to achieve that goal by trial and error interactions with the environment

Example: Learning to ride a bike

suppose in the first trial, the RL system begins riding the bicycle and performs a series of actions that result in the bicycle being tilted 45 degrees to the right

At this point, the RL system has not only learned that turning the handle bars right or left when tilted 45 degrees right is bad but “that the state of being tilted 45 degrees to the right is bad.

RL as a function approximation method

Learning the controller policy is very similar to function approximation, except

Delayed reward: in RL, the trainer provides only a sequence of immediate reward values

=> Facing the problem of temporal credit assignment problem

Exploration and Exploitation:

Exploration is to collect new information

Exploitation of what it has already learned to maximize the cumulative rewards

in RL, the agents influence the distribution of training examples by the action sequences it chooses

RL Vs. Other Function Approximator

Partially observable states:

In many practical scenarios the sensors only provide partial information (like the camera in-front of a robot)

Solution–Considering previous observations along with the current sensor data

Life-long Learning:

Unlike the function approximation task, in RL, robots need to learn many tasks simultaneously plus online learning process forever

RL Convergence

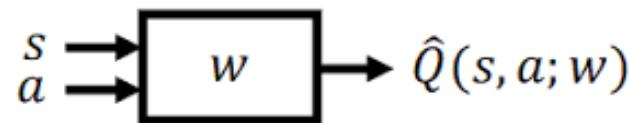
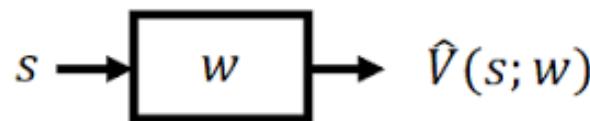
Three conditions of convergence

- Deterministic Markov Decision Process
- Immediate positive bounded reward
- Agent selects every agent-action pairs infinitely often

Q function and State-value function

V (S) : The state value function tells us the value for being in some state when following some policy.

Q (S, a) : The action value function tells us the value of taking an action in some state when following a certain policy.



Relation between State-value function and Q-Value function

Q (S_t, a_t) value function : Quality of action (a) as per observed state (S)

Value (S_t) function : Return /gain from the observed state (S)

Key difference : In Q (S_t, a_t), action at t^{th} time step (a_t) is already determined $v_*(s) = \max_a q_*(s, a)$

$$\begin{aligned} v_*(s) &= \max_{a(s)} q_{\pi*}(s, a) \\ &= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')] \end{aligned}$$

$$q_*(s, a) = \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} q_*(s', a')]$$

How to learn an Optimal Policy ?

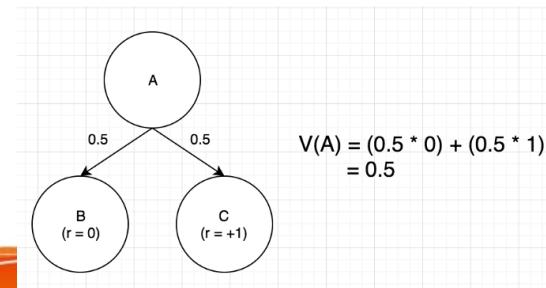
- How good is a state (S) ?

Expected cumulative reward from state, S

Intuition :

- *good state : If the state, S , is more likely to lead to a path that ends up to a goal state.*
- *bad state : If the state S , is more likely to lead to a path that would not be end up at one of goal states.*

How good or bad (intensity / subjectivity) ?



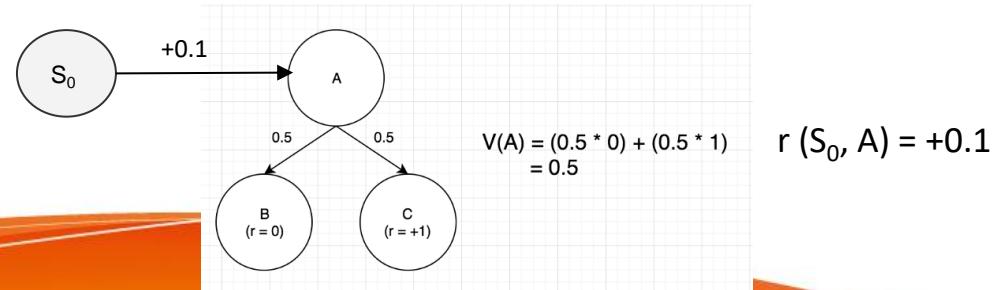
State Value function

Value function : Expected cumulative reward from following the policy from state (S)

- The Value Function represents the value for the agent to be in a certain state.
- Expected return G_t from a given state

$$V^\pi(s) = E [\sum_{t \geq 0} \gamma^t * r_t | S_0 = s, \pi]$$

Reward Vs Value function : A reward is immediate. Value function measures potential future rewards we may get from being in this state s.

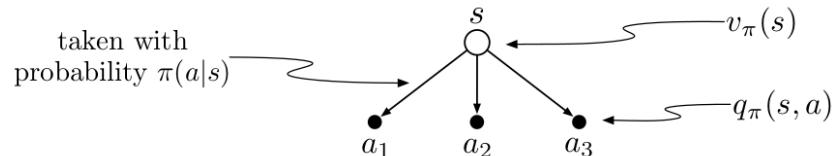


Q State -Action Value function

Q State - Action Value function :

$Q(S, a)$: Relevance of action, a , for state, S

$$= r_{\text{immediate}} + \gamma * \arg \max_{a'} Q(S', a')$$



$$Q^\pi(s_t, a_t) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | s_t, a_t]$$

State-action value Cumulative reward state-action pair

$$= E_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_t \mid s_t = s, a_t = a \right]$$

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0

	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017

Actions
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Q State -Action Value function



Thomas et al., 2018

	\leftarrow	\rightarrow	\uparrow	\downarrow
Start	0	0	0	0
Small cheese	0	0	0	0
Nothing	0	0	0	0
2 small cheese	0	0	0	0
Death	0	0	0	0
Big cheese	0	0	0	0

$$r(\text{start, right}) = 1$$

$$\text{New } Q(\text{start, right}) = Q(\text{start, right}) + \alpha[\Delta Q(\text{start, right})]$$

$$\Delta Q(\text{start, right}) = R(\text{start, right}) + \gamma \max Q'(\text{1cheese, } a') - Q(\text{start, right})$$

$$\Delta Q(\text{start, right}) = 1 + 0.9 * \max(Q'(\text{1cheese, left}), Q'(\text{1cheese, right}), Q'(\text{1cheese, down})) - Q(\text{start, right})$$

$$\Delta Q(\text{start, right}) = 1 + 0.9 * 0 - 0 = 1$$

$$\text{New } Q(\text{start, right}) = 0 + 0.1 * 1 = 0.1$$

	\leftarrow	\rightarrow	\uparrow	\downarrow
Start	0	0.1	0	0
Small cheese	0	0	0	0
Nothing	0	0	0	0
2 small cheese	0	0	0	0
Death	0	0	0	0
Big cheese	0	0	0	0

Bellman Equation

In reinforcement learning, we want the agent to be able to relate the value of the current state to the value of future states, without waiting to observe all future rewards.

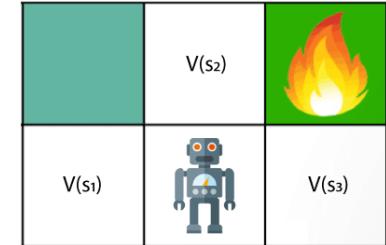
The **Bellman equation** is one way to formalize this connection between the value of a state and future possible states.



Richard Bellman
(1856-1922)

$$V(s) = \max [R(s,a) + \gamma * V(s')]$$

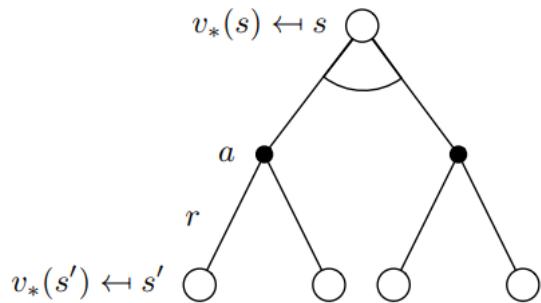
$$\begin{aligned} v(s) &= \mathbb{E} [G_t \mid S_t = s] \\ &= \mathbb{E} [R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s] \\ &= \mathbb{E} [R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) \mid S_t = s] \\ &= \mathbb{E} [R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E} [R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \end{aligned}$$



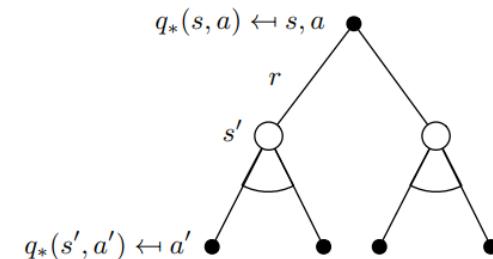
Bellman Optimality Equation

“ The Optimal Value Function is recursively related to the Bellman Optimality Equation “

$$q_*(s, a) = E \left[R_{t+1} + \gamma \max_{a'} q_*(s', a') \right]$$



Bellman Optimality Equation for State-Value function

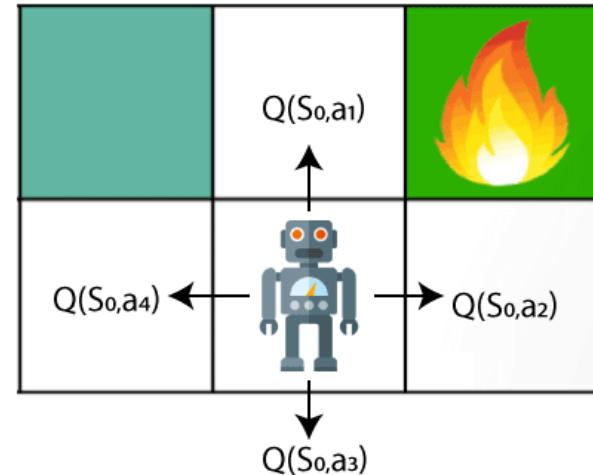


Bellman Optimality Equation for State-Action Value function

Bellman Equation

Bellman Equation in terms of Q - value function

$$\begin{aligned} q_{\pi}(s, a) &= \mathbb{E}_{\pi}[R_t + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma \mathbb{E}_{\pi}[G_{t+1} | S_{t+1} = s']] \\ &= \sum_{s', r} p(s', r|s, a) [r + \gamma \sum_a \pi(a|s) q_{\pi}(s', a')] \end{aligned}$$



An Example

s1	s2	s3	
	s10	s11	s12

R=+1

R=-1

The agent will try to remember the preceding steps that it has taken to reach the final step.



V=1	V=1	V=1	
V=1			

But what will the agent do if he starts moving from the block, which has 1 value block on both sides ?

	V=1	V=1	
V=1			

An Example

s1	s2	s3	
s5		s7	
	s10	s11	s12

R=+1
R=-1

$$\begin{aligned}
 V(s_5) &= \max [R(s,a) + \gamma V(s')] \\
 &= 0 + 0.9 \times 0.81 \\
 &= 0.73
 \end{aligned}$$

$$\begin{aligned}
 V(s_9) &= \max [R(s,a) + \gamma V(s')] \\
 &= 0 + 0.9 \times 0.71 \\
 &= 0.66
 \end{aligned}$$

$$\begin{aligned}
 V(s_3) &= \max [R(s,a) + \gamma V(s')], \\
 &= 1 + \gamma \times 0 \\
 &= 1
 \end{aligned}$$

$$\begin{aligned}
 V(s_2) &= \max [R(s,a) + \gamma V(s')], \text{ here } \gamma = 0.9 (\text{lets}), \\
 &= 0 + 0.9 \times 1 \\
 &= 0.9
 \end{aligned}$$

$$\begin{aligned}
 V(s_1) &= \max [R(s,a) + \gamma V(s')] \\
 &= 0 + 0.9 \times 0.9 = 0.81
 \end{aligned}$$

V=0.81	V=0.9	V=1	
s1	s2	s3	
V=0.73			
s5	s6	s7	
V=0.66			
s9	s10	s11	s12

↓
V=0.66

Role of Discount factor (γ)

- Keep the total reward/total cost finite
 - useful for infinite horizon problems
- Intuition (economics):
 - Money today is worth more than money tomorrow.

Total reward: $r_1 + \gamma \cdot r_2 + \gamma^2 \cdot r_3 + \dots$

Total cost: $c_1 + \gamma \cdot c_2 + \gamma^2 \cdot c_3 + \dots$

Policy Learning : Reward Maximization

The goal of intelligence is to maximize the cumulative sum of a single received number:

$$\text{reward} = \text{pleasure} - \text{pain}$$

General Artificial Intelligence = reward

maximization

Exploration Vs Exploitation

Exploration: choose actions that visit new states in order to obtain more data for better learning.

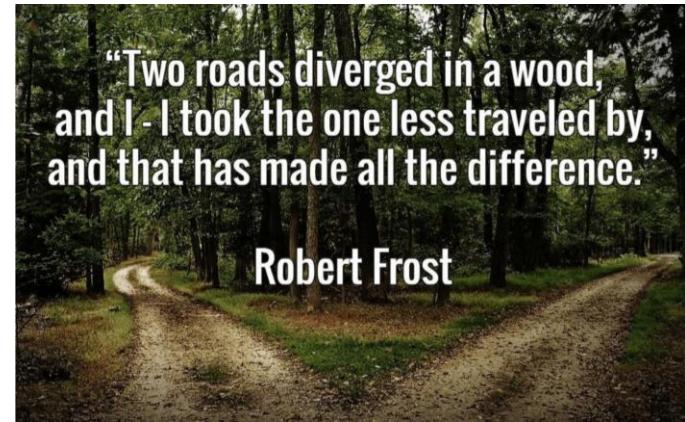
Exploitation: choose actions that maximize the reward given current learnt model

ϵ -greedy

- With ϵ prob take an action randomly
- With prob $1-\epsilon$ take the current greedy action

Lower ϵ over time

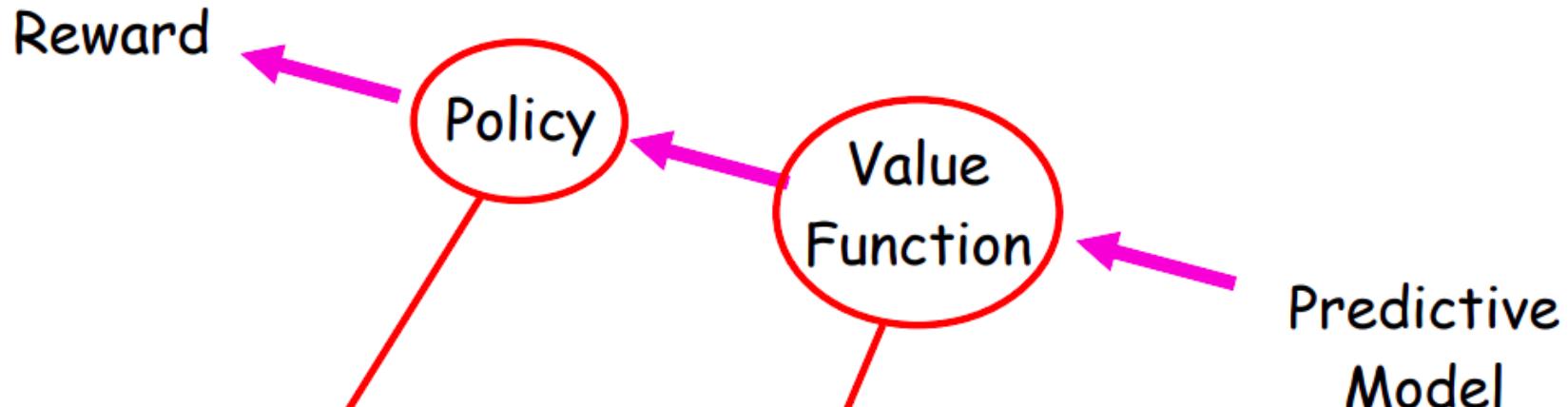
- Increase exploitation as more learning has happened



“Two roads diverged in a wood,
and I - I took the one less traveled by,
and that has made all the difference.”

Robert Frost

Reward, Value Function and Policy

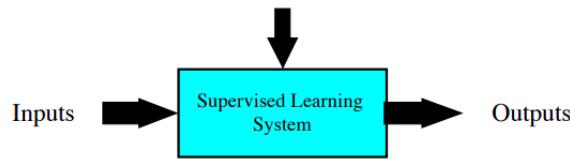


A learned, time-varying prediction of imminent reward
Key to all efficient methods for finding optimal policies

SL Vs RL

Supervised Learning

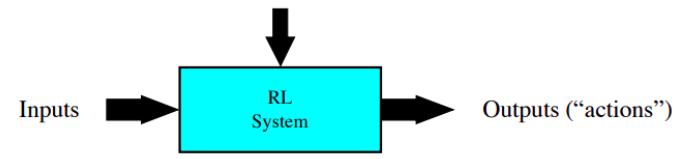
Training Info = desired (target) outputs



Error = $(\text{target output} - \text{actual output})$

Reinforcement Learning

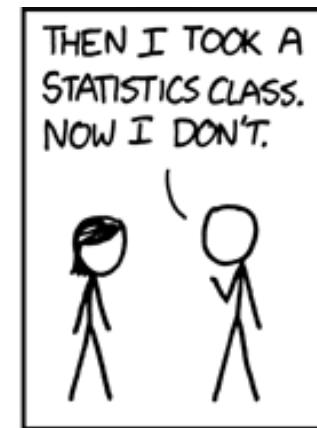
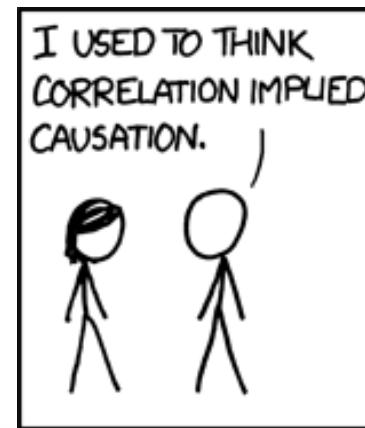
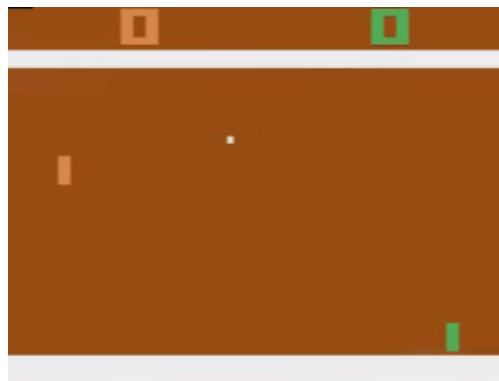
Training Info = evaluations (“rewards” / “penalties”)



Objective: get as much reward as possible

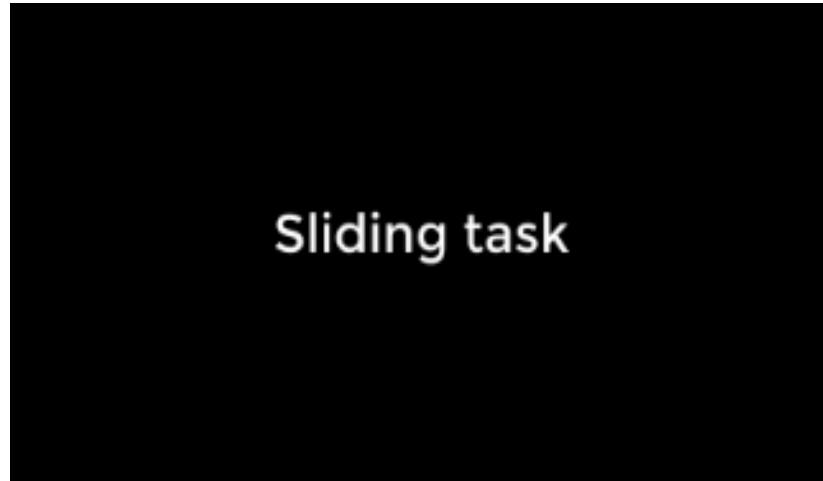
Credit Assignment

- Pong, you hit the paddle, 20 frames later you earn a point by it passing the opponent's paddle.
- What actions contributed to earning this point?
- Supervised Learning: instant reward/supervision. RL: generally delayed rewards.



Sparse, noisy and delayed rewards

- “Reward shaping” requires an insane amount of engineering
- Many other ways of solving this e.g. better exploration, sub-goals, etc



OpenAI 2017: Hindsight Experience Replay

Reward hacking

- The agent maximizes your reward function without the desired behaviour
- All it knows is what gives it reward. This might not be aligned with exactly how the human wants the agent to get the reward



OpenAI 2016: Faulty Reward Functions

How to design a good reward function

Crafting reward functions for reinforcement learning models is not easy

An illustration

- Reward is an way of encapsulating domain and problem's objective into learning model.
- Be sure to understand what your reward function is doing. Analyze model's performance for some smaller time of training.
- **Negative Reward** Negative rewards incentivize you to get done as quickly as possible
- **Reward Shaping**

Reward shaping is a big deal. Gradual feedback and let it know it's getting better when it is reaching close to the underlying objective. It helps it learn a lot faster



Reward Model 1

if success +1 else 0

Reward Model 2

*if success +1
elif fail 0
else d(S', G) // S' : next state*

G : Goal

Applications

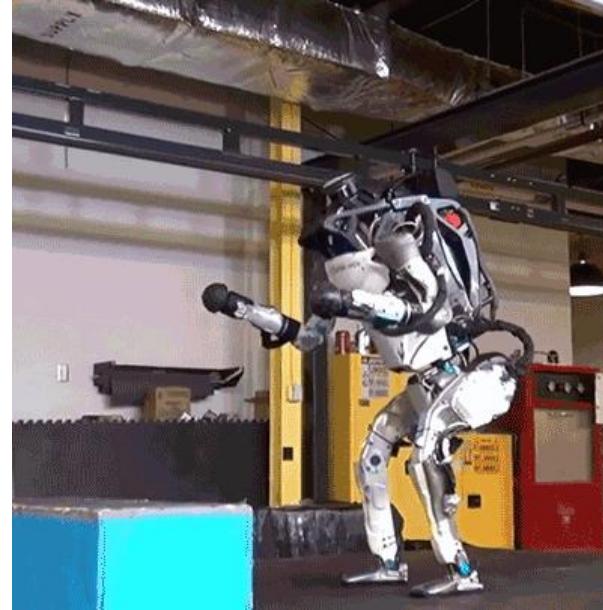
Many important problems are MDPs....

- Robot path planning
- Travel route planning
- Elevator scheduling
- Bank customer retention
- Autonomous aircraft navigation
- Manufacturing processes
- Network switching & routing

Applications

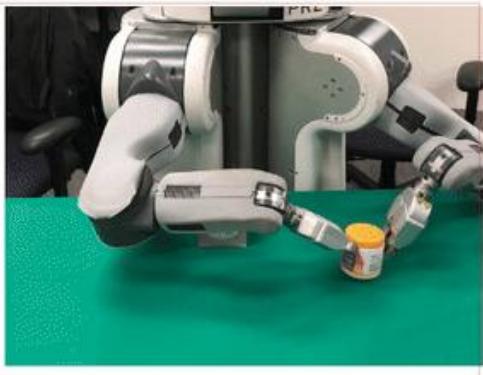
“Autonomous helicopter flight via Reinforcement Learning”

Ng (Stanford), Kim, Jordan, & Sastry (UC Berkeley) 2004



Boston Dynamics 2017: What's new, Atlas?

Applications



BAIR 2017: One-shot Imitation from Watching Videos



BAIR 2018: Learning Acrobatics by Watching YouTube (Imitation Learning + Inverse RL)

Chat-bot



Hello!! How can
I help you ??



.....



Anytime

My Computer
Isn't working.



.....



Great!!!! that
solved my problem



Dialogue: Definition

- A formal discussion of a topic in speech or writing.
- A connected series of utterances; a text or conversation.
- Exchange of ideas or opinions on a particular issue, with a view to reaching an amicable agreement.

Characteristics of Dialogue



Who's Turn It Is ?

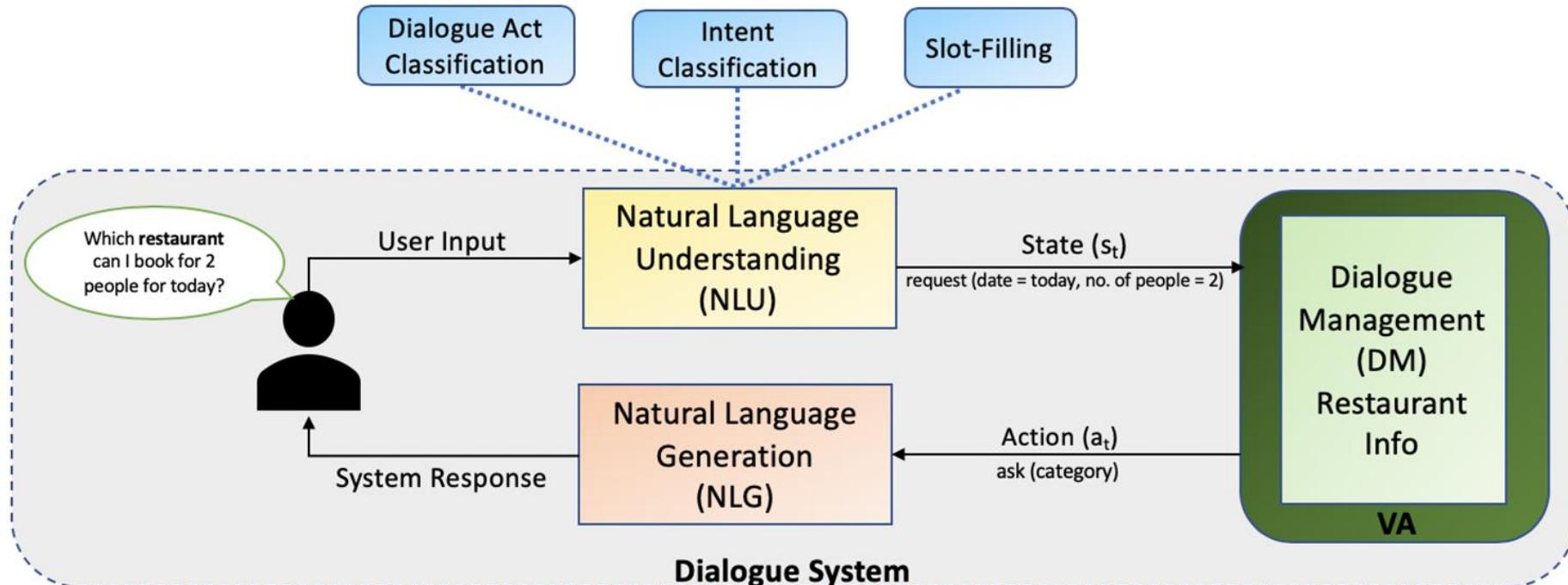


What Are The Intentions ?



How to Reply Back ?

Dialogue System



DIALOGUE MANAGEMENT STRATEGY

- The main function of the dialogue management component is to control the flow of the dialogue. This involves the following task:
 - Determining whether sufficient information has been elicited from the user in order to take a suitable action
 - Execute an action
 - Communicate with an application
 - Communicate with the user

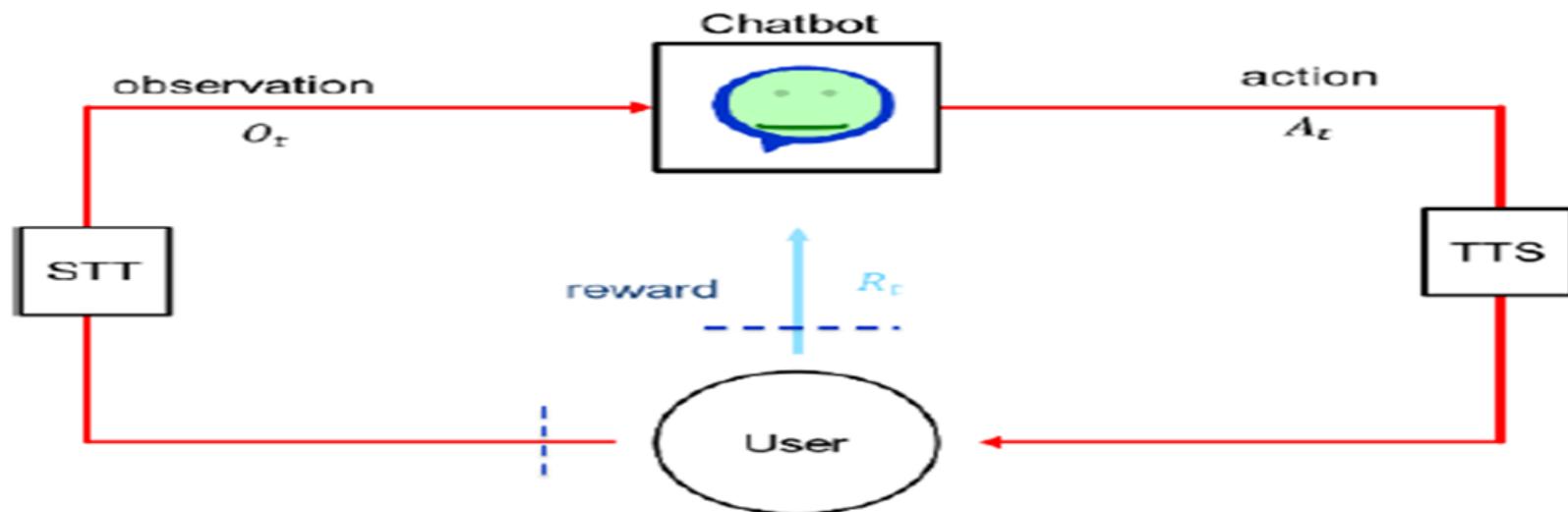
WHY RL ?

- Currently, the dialogue management strategy employed is completely **RULE-BASED**
- Hence, the inclusion of RL :
- Minimize the use of hand-crafting
- Data driven
- To exploit the recent advances in Reinforcement Learning and Deep Learning
- End to end synthesis
- Create an infrastructure that will be used across different domains

A RL CHATBOT

- The RL Chatbot takes an action (a_t) at time t which is converted into speech by text-to-speech (TTS) unit
- The user's response, in the form of speech is converted to text by speech-to-text (STT) unit
- The output of the STT unit i.e. the observation (o_t) is received by the chatbot
- The user may either generate a reward signal (r_t) explicitly or it is derived from the user's response
- The task of the RL Chatbot is to respond in a way so as to maximize its reward

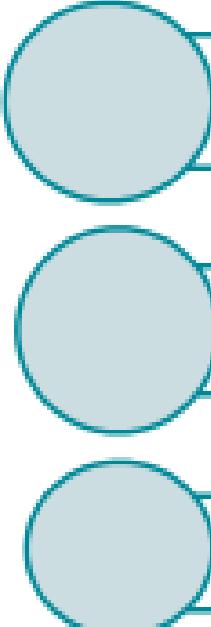
A RL Chat-bot



RL

- Thus, any RL setup is mathematically described in terms of an abstraction called the Markov Decision Process (MDP)
- A MDP model is described by $M = \langle S; A; R; P; \text{Rho} \rangle$, where
 - S is the finite state space
 - A is the finite action space (corresponding to VA's action)
 - R is the reward model
 - P is the transition model
 - Rho is the initial state distribution
- The idea is to implement RL for learning dialogue strategies using the Q-learning algorithm
- The main idea in Q-learning is that we can iteratively approximate the Q-function using the Bellman Equation (via Dynamic programming but still slow)
- Hence, the Q-function will be approximated using a neural network (incorporation of Deep RL)

Problem Statement



Developing a *Hierarchical Dialogue Management Strategy* for Task-Oriented Virtual Agent

For managing *Multi-Intent Conversations*

Using Reinforcement Learning (Deep RL)

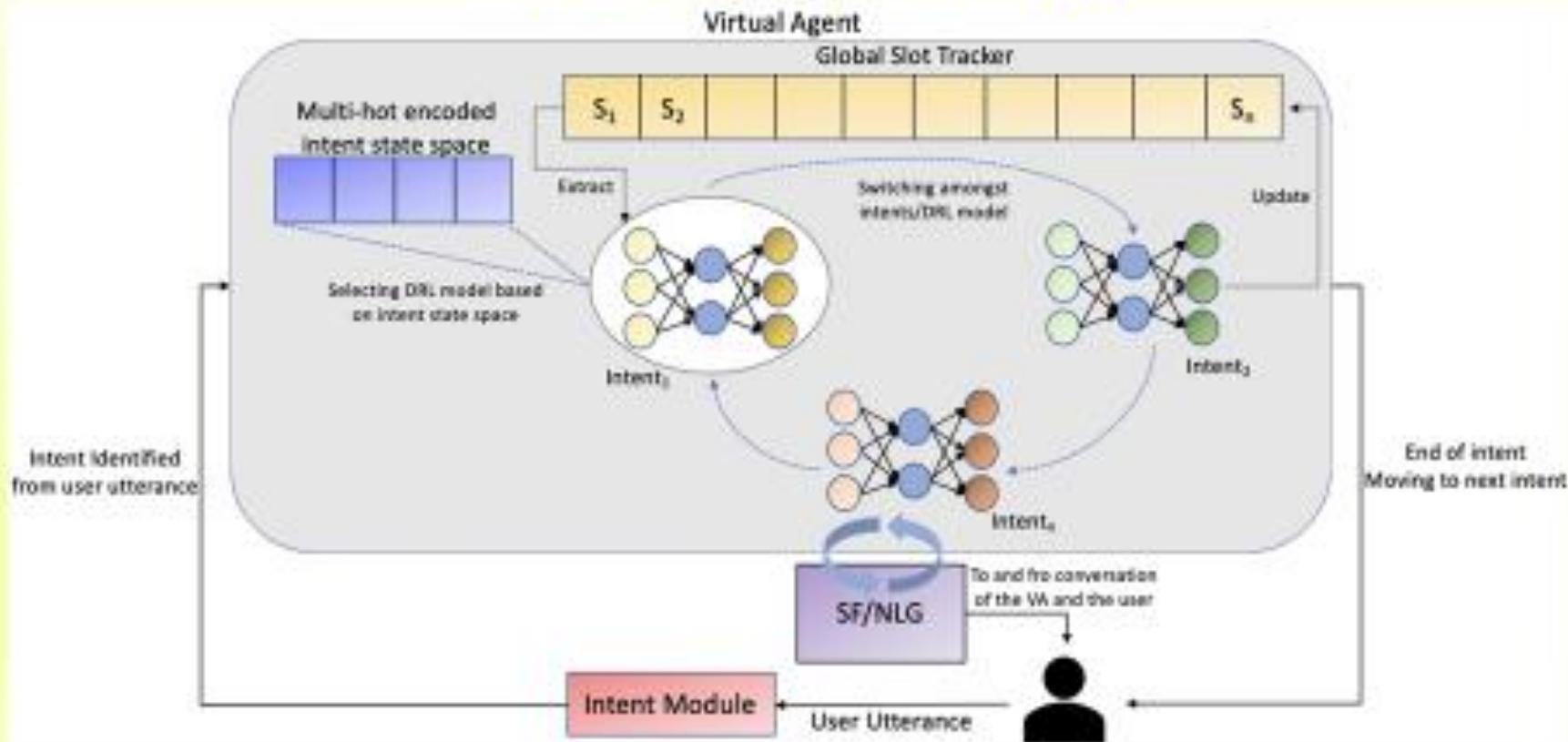
What is Multi-Intent ?

Intent captures the main communicative intention of the user utterances.

Multi-intent captures more than one intentions present in an user utterance.

For eg., "*Show me flights and cheapest fares from Pittsburgh to Denver*" has multiple intents which are **flight** and **airfare**

Proposed Methodology



Proposed MDP

- A generic architecture of MDP is used. It finds its applicability in any domain having n number of intents and m number of slots.
- **State Space :** At any given time, the state space consists of a pair of states (S_1, S_2) which are described as follows :
 - (S_1) : It is a state which is a tuple of n variables corresponding to the number of intents in a particular domain.
 - It keeps track of the current intent (output from the IC module) being serviced by the VA in the form of one hot encoded vector.
 - This in turn invokes the appropriate RL model corresponding to the intent being served. State (S_1) represents the top-level hierarchy of the DM.

Proposed MDP

- (S_2) : For each of the intents, (S_2) is a tuple of variables of the length equal to the number of slots required to be filled up for that particular intent.
 - This state represents the bottom-level hierarchy of the DM.
 - These slot variables in turn correspond to confidence scores of different slots which are the probability values outputted from the SF module representing the confidence of the module in predicting different slot labels.
- **Action Space** : The action space constitutes of 21 unique actions categorized in three different classes, i.e., *Ask*, *Reask/Confirm* and *Salutation* such as *askClass*, *reaskGCity* and so on.

Reward Model

- Different reward function at different time-step/ for different actions in conversation
 - For any action except terminating action

$$R(s, a, s') = (w_1 * (||\vec{NS}||_1 - ||\vec{CS}||_1) - w_2)$$

- $||\vec{NS}||_1$ state is the summation of the confidence scores of all state variables in the state vector 's' obtained after taking an action 'a' in state 's'.
- $||\vec{CS}||_1$ state is the summation of confidence scores for all state variables in state vector 's'
- w_1 is used to encourage the agent to act in a way as to increase confidence on the acquired slots
- w_2 is used to discourage unnecessary questions and iterations with the user.
- $w_2 = 1$ and $w_1 = 8$

Reward Model

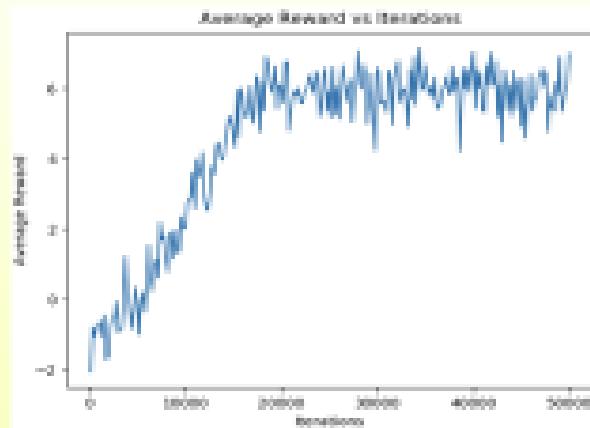
- For terminating action,
 - The **condition** is that how many slots has a confidence score above a threshold say 0.7, the condition is satisfied when all slots match the criteria.
 - If the condition is not satisfied , the reward function is :

$$R(s,a,s') = -w_1 * (||\overrightarrow{EV}||_1 - ||\overrightarrow{CS}||_1)$$

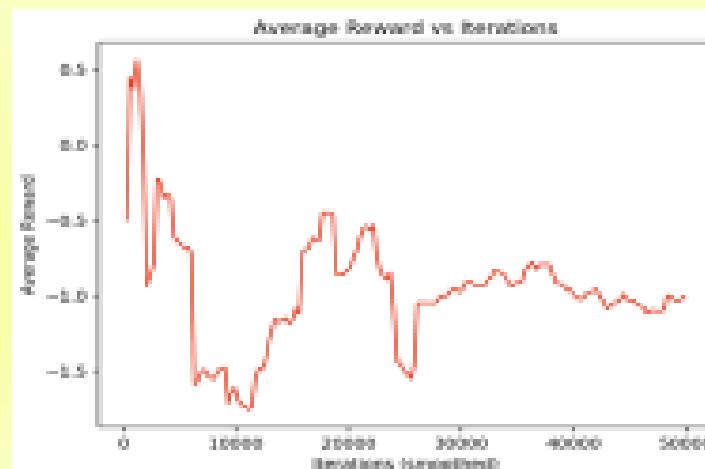
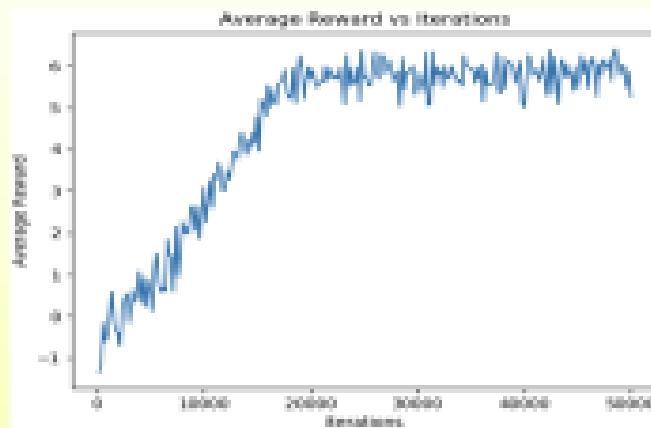
- $||\overrightarrow{EV}||_1$ is the summation of maximum expected confidence scores of different slots.
- If the condition is satisfied :

$$R(s,a,s') = w_1 * ||\overrightarrow{CS}||_1$$

Results (Learning Curves)



For different
MDPs



For Flat RL
Agent

VERBALIZATION OF THE PROPOSED SYSTEM

System Log : No intent

S₁ : [0 0 0 0 0]

Agent > "Hello How can I help you ?"

User > "I would like to travel from pittsburgh to denver"

System Log : Intent Changed to 'flight'

S₁ : [1 0 0 0 0]

S₂ : [0.98 0.97 0. 0. 0.]

Agent [askClass] > "Please specify the class of flight"

User > "I want to travel in economy class"

S₁ : [1 0 0 0 0]

S₂ : [0.98 0.97 0. 0. 0.94]

Agent [askDateTime] > "Specify the time and date"

User > "On fourth january at morning"

S₁ : [1 0 0 0 0]

S₂ : [0.98 0.97 0.87 0.99 0.94]

Agent [closing_conversation] > "The flights from pittsburgh to denver on fourth january at morning via economy class are JW456, GF678.

Do you need anything else ?"

User [Yes/No] > "Show me the cheapest fares"

System Log : Intent Changed to 'airfare'

S₁ : [0 1 0 0 0]

S₂ : [0.98 0.97 0. 0.87 0.94]

Agent [askRtrip] > "Round trip or one way fare ?"

User > "round trip"

S₁ : [0 1 0 0 0]

S₂ : [0.98 0.97 0. 0.99 0.87 0.94]

Agent [closing_conversation] > "The cheapest fare of the flight from pittsburgh to denver on fourth january and round trip via economy class is \$1000 for JW456.

Do you need anything else ?"

User [Yes/No] > "No Thanks"

System Log : No Intent

Exit

VA IN HINDI

- We intend to exploit the fact that even though there is a language difference, but the semantic relation of a specific task co-exist irrespective of any language barriers.
- Thus, we have reused the same DM module trained once and by replacing its supporting module such as NLU (specifically IC and SF) and NLG by its corresponding *Hindi* counterpart, created a VA which can serve the user in that specific language.
- The IC and SF module has been trained on the semantic tagging corpus developed in *Hindi* for the ATIS dataset
- Following this similar approach, the DM module can be reused for any other language for the related tasks given that there is an availability of a semantic tagging corpus for the task in that given language.

VA IN HINDI

System Log : No intent

S₁ : [0 0 0 0 0]

Agent > "उत्तम उत्तर प्रणाली में आपका स्वागत है"

User > "जमीनी परिवहन की क्या सुविधा है"

System Log : Intent changed to 'ground_service'

S₁ : [0 0 0 1 0] S₂ : [0. 0.]

Agent [askGcityTrantype] > "कृपया जमीनी सेवा के स्पष्ट और परिवहन प्रणाली को निश्चित करें"

User > "डेंवर में निम्नमें लिमोसिन सेवा है"

S₁ : [0 0 0 1 0] S₂ : [0.945 0.876]

Agent [closing_conversation] > "हाँ निम्नलिखित सेवा डेंवर में उपलब्ध हैं। क्या अपने कुछ और जल्दी बाहर है?"

User[हाँ/नहीं] > "हाँ मुझे उड़ाने दिलाएँ"

System Log : Intent changed to 'flight'

S₁ : [1 0 0 0 0] S₂ : [0. 0. 0. 0. 0.]

Agent [askDeptArr] > "आप कहा से कहा तक की उड़ान भरना चाहते हैं?"

User > "मैं बोस्टन से डेंवर तक उड़ान भरना चाहता हूँ"

S₁ : [1 0 0 0 0] S₂ : [0.998 0.988 0. 0. 0.]

Agent [askClass] > "कृपया अपनी उड़ान की श्रेणी नियन्त्रण करें"

User > "मैं व्यावसायिक श्रेणी में उड़ान भरना चाहता हूँ"

S₁ : [1 0 0 0 0] S₂ : [0.998 0.988 0. 0. 0.997]

Agent [askDateTime] > "कृपया तारीख और समय नियन्त्रण करें"

User > "मैं बाज़ जनवरी को सुबह में उड़ान भरना चाहता हूँ"

S₁ : [1 0 0 0 0] S₂ : [0.998 0.988 0.932 0.976 0.997]

Agent [closing_conversation] > "आपकी बोस्टन से डेंवर के लिए बाज़ जनवरी को सुबह की व्यावसायिक श्रेणी की उड़ानें हैं A, B, C। क्या अपने कुछ और जल्दी बाहर है?"

User[हाँ/नहीं] > "नहीं धन्यवाद"

System Log : No intent

Exit

Summary

- MDP defined by states, actions, transition model, reward function
- The “solution” to an MDP is the policy: what do you do when you’re in any given state
- The Bellman equation tells the utility of any given state
- The agent should have the chance to try (and fail) enough times : This is impossible if episode takes too long, e.g., reward=“obtain a great Ph.D.”

Module 3: Tabular methods and Q-networks

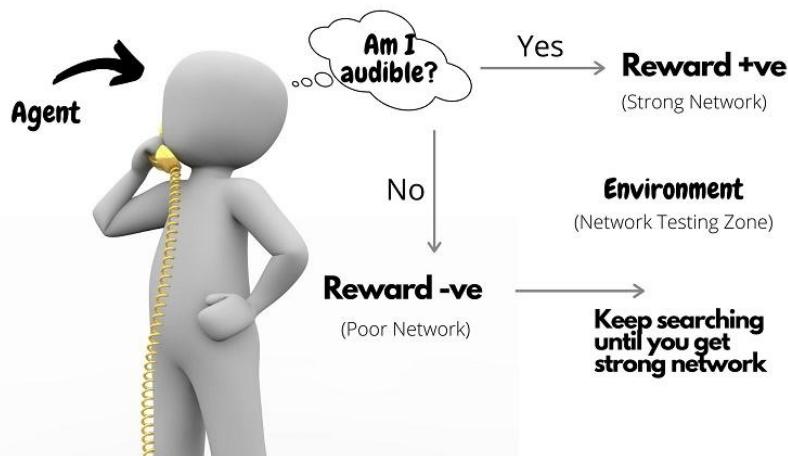
Deep Q-networks (DQN, DDQN, Dueling DQN, Prioritised Experience Replay)

Quick Recap

RL = learning how to act (map situations to decisions), **RL** = Real Life

Task : Set of decisions for observed set of environment states

Goal : Learn how to take decisions (actions) in order to maximize reward



Policy

Policy (π) : A policy is a function from S to A that specifies what action to take in each state

$$a = \pi(S)$$

Optimal Policy (π^*) : A policy that always takes best action for a given state, leading to gain of highest possible reward from the state.

$$a^* = \pi^*(S)$$

Objective : find policy π^* that maximizes cumulative discounted reward:

$$\pi^* = \arg \max_{\pi} [\sum_{t \geq 0} \gamma^t * r_t | \pi]$$

How to learn an Optimal Policy ?

$$\begin{aligned} a_{t+1} &= \Pi_\Theta(S_t) \\ &= \operatorname{argmax}_i Q(S_t, A_i | \Pi_\Theta) \\ &= \operatorname{argmax}_i (r_t + \operatorname{argmax}_{a_{(t+2)}} Q_{a_{(t+2)}}(S_{t+1}, a_{t+2})) \\ &= \operatorname{argmax}_i (r_t + V(S_{t+1})) \end{aligned}$$

Thus, we either need to learn an **Optimal Q function, $Q(S,a)$** or **State-Value function, $V(S)$**

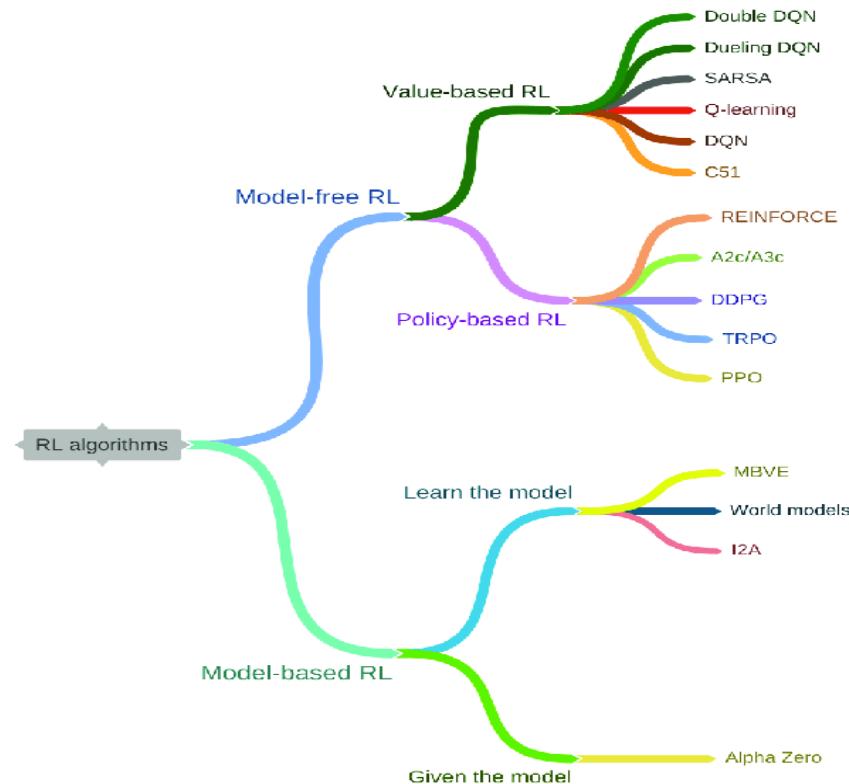
Reinforcement Learning Algorithms

- Model-free and Model-based RL

whether the agent has access to (or learns) a model of the environment

What to learn ?

- policies, either stochastic or deterministic,
- action-value functions (Q-functions),
- value functions,
- and/or environment models



Approaches To Reinforcement Learning

Value-based RL

Estimate the optimal value function $Q^*(s, a)$

Policy-based RL

Search directly for the optimal policy π^*

Model-based RL

Build a model of the environment

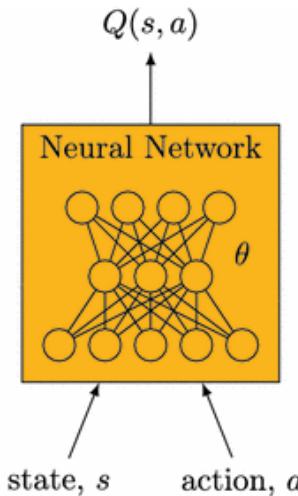
Value-based and Policy-based RL Algorithms

Value-based RL

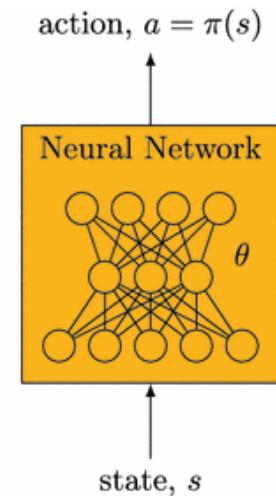
In a value-based Reinforcement Learning method, we learn a state-action $Q(s,a)$ / value function $V(s)$. The agent is expecting a long-term return of the current states under policy π .

Policy-based RL

In a policy-based RL method, we try to come up with such a policy that the action performed in every state helps you to gain maximum reward in the future.



(a) Value-Based DRL



(b) Policy-Based DRL

Q Learning

- Q-Learning is a basic form of Reinforcement Learning which uses Q-values (also called action values) to iteratively improve the behavior of the learning agent.
- It is a model-free, off-policy reinforcement learning that will find the best course of action, given the current state of the agent.

Bellman Equation is used to determine the value of a particular state and deduce how good it is to be in/take that state.

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \underbrace{\left(\underbrace{r_t + \gamma \cdot \max_a Q(s_{t+1}, a)}_{\substack{\text{reward} \\ \text{discount factor} \\ \text{estimate of optimal future value}}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)}_{\text{temporal difference}}$$

new value (temporal difference target)

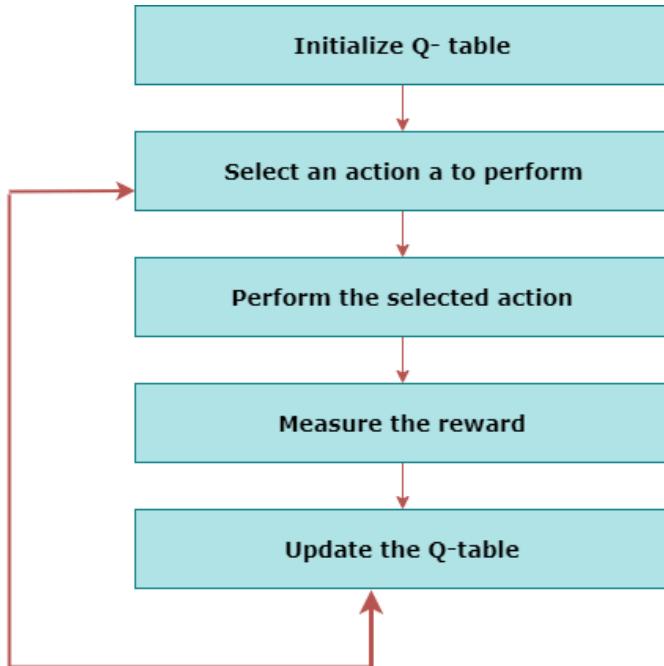
Important Terms in Q-Learning

- **States:** The State, S, represents the current position of an agent in an environment.
- **Action:** The Action, A, is the step taken by the agent when it is in a particular state.
- **Rewards:** For every action, the agent will get a positive or negative reward.
- **Episodes:** When an agent ends up in a terminating state and can't take a new action.
- **Q-Values:** Used to determine how good an Action, A, taken at a particular state, S, is. $Q(A, S)$.
- **Temporal Difference:** A formula used to find the Q-Value by using the value of current state and action and previous state and action.

A policy maps a state to an action

$$a = \pi(s)$$

Q Learning



Initialized

Q-Table		Actions					
States	0	South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
		0	0	0	0	0	0
States	327	·	·	·	·	·	·
		0	0	0	0	0	0
States	499	·	·	·	·	·	·
		0	0	0	0	0	0

Training

Q-Table		Actions					
States	0	South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
		0	0	0	0	0	0
States	328	·	·	·	·	·	·
		-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
States	499	·	·	·	·	·	·
		9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Q Learning

Initialize $Q(s, a)$ arbitrarily

Repeat (for each episode):

 Initialize s

 Repeat (for each step of episode):

 Choose a from s using policy derived from Q

 Take action a , observe r, s'

 Update

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

$s \leftarrow s'$;

 Until s is terminal

An Illustration

Action	Fetching	Sitting	Running
Start	0	0	0
Idle	0	0	0
Wrong Action	0	0	0
Correct Action	0	0	0
End	0	0	0



Action	Fetching	Sitting	Running
Start	0	1	0
Idle	0	0	0
Wrong Action	0	0	0
Correct Action	0	0	0
End	0	0	0

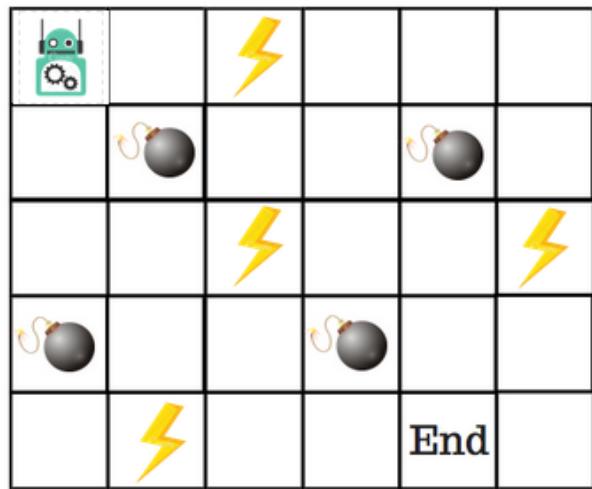


Action	Fetching	Sitting	Running
Start	5	7	10
Idle	2	5	3
Wrong Action	2	6	1
Correct Action	54	34	17
End	3	1	4



Action	Fetching	Sitting	Running
Start	0	1	0
Idle	0	0	0
Wrong Action	0	0	0
Correct Action	0	34	0
End	0	0	0

An Illustration

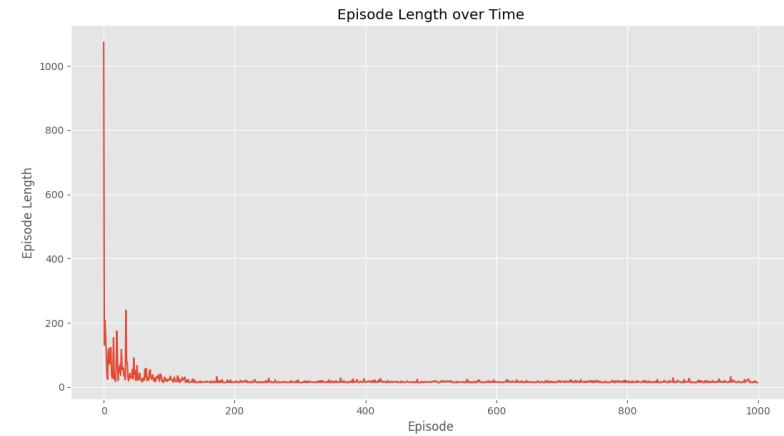
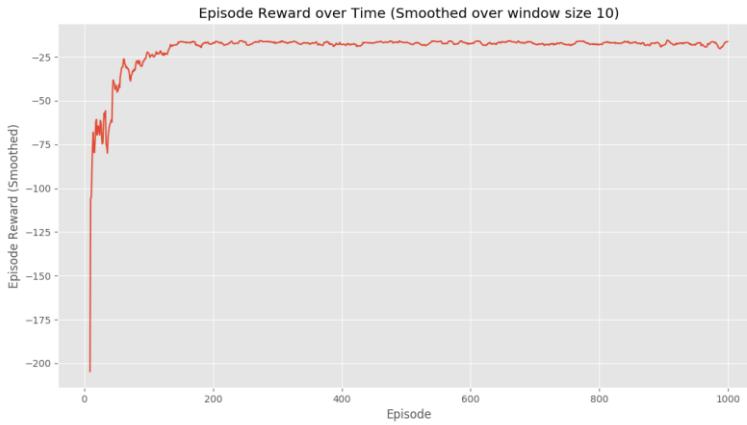


Actions : ↑ → ↓ ←

Start	0	0	0	0
Nothing / Blank	0	0	0	0
Power	0	0	0	0
Mines	0	0	0	0
END	0	0	0	0

Characteristics of Q Learning

- The discount factor (γ) determines the importance of future rewards.
 - $\gamma = 0$, will make the agent "myopic" (or short-sighted) by only considering current rewards
 - $\gamma = 1$ will make it strive for a long-term high reward



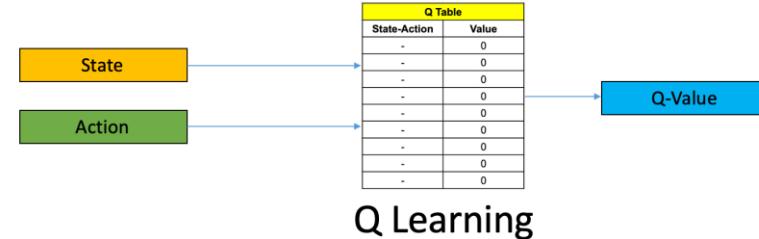
Deep Reinforcement Learning

Why Deep Reinforcement Learning ?

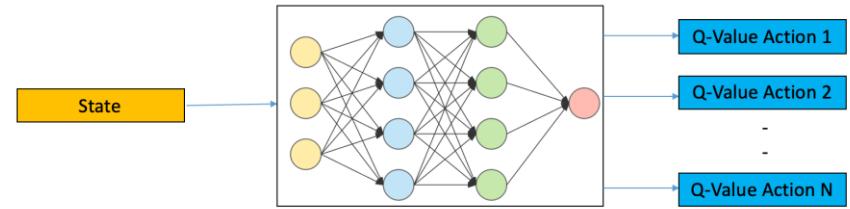
- Too many states to store in memory
- Too slow to learn the values of each state separately
- We need to generalise what we learn across states.

Use deep neural networks to represent

- Value function
- Policy
- Model

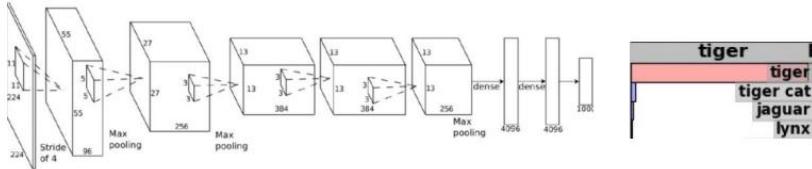


Q Learning



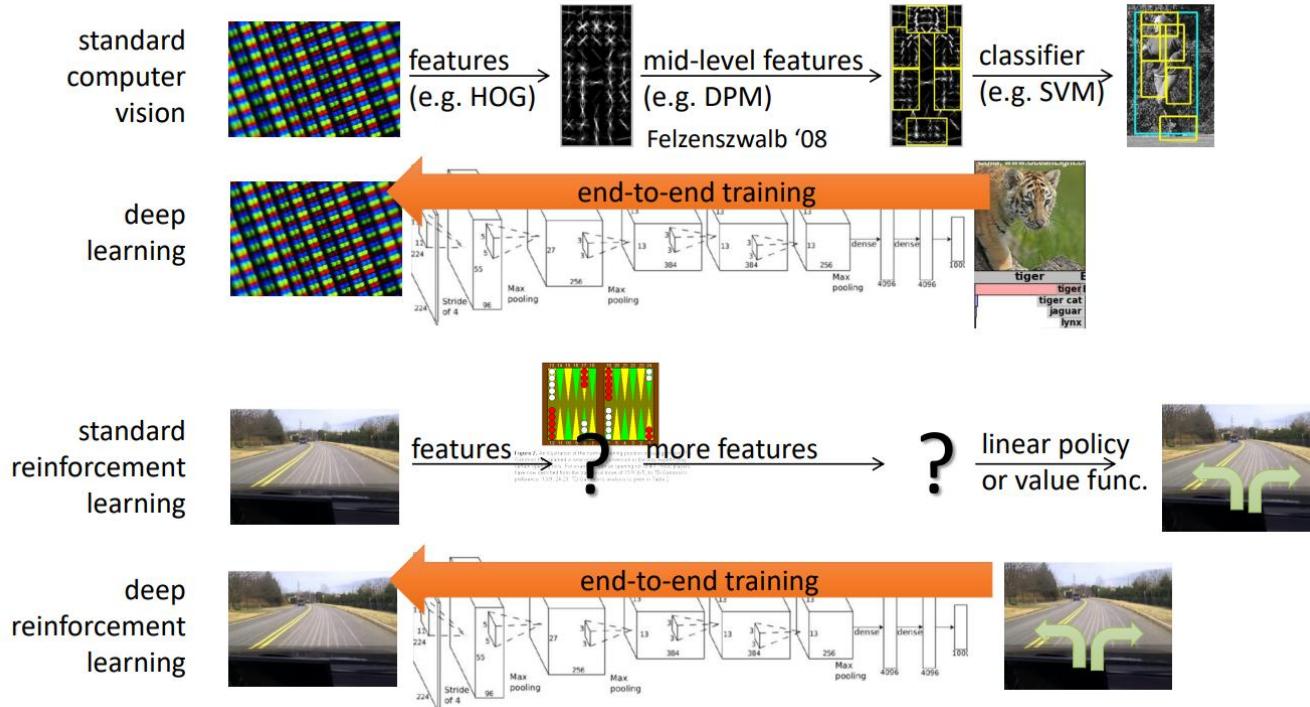
Deep Q Learning

Why Deep Reinforcement Learning ?

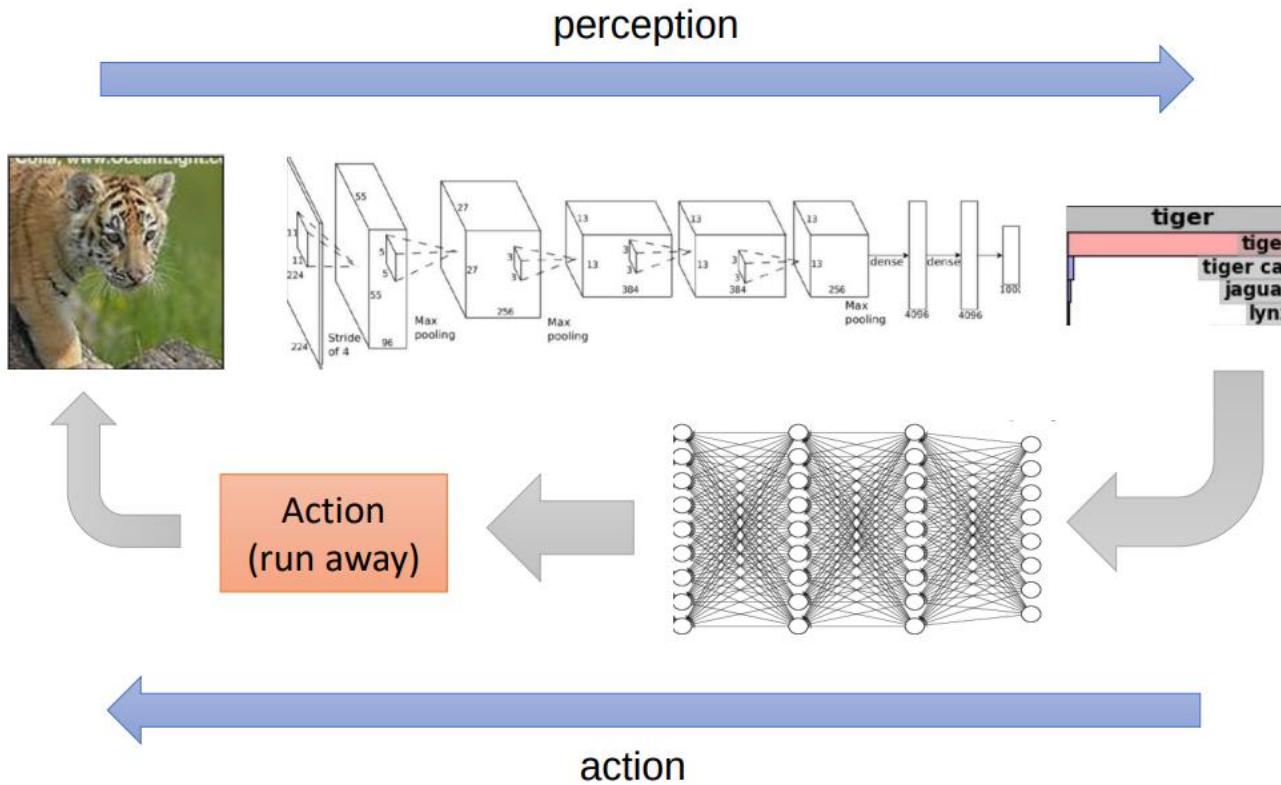


1. Advances in deep learning
2. Advances in reinforcement learning
3. Advances in computational capability

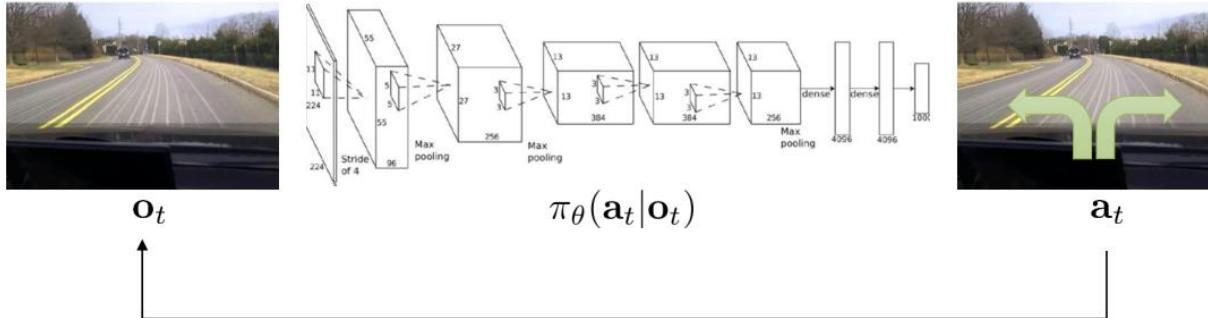
Why Deep Reinforcement Learning ?



Why Deep Reinforcement Learning ?



Deep Reinforcement Learning



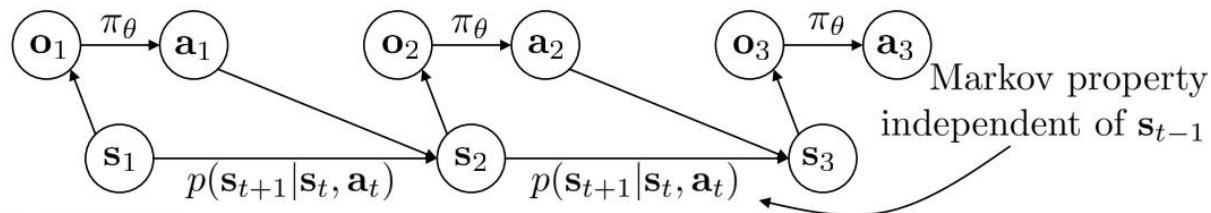
\mathbf{s}_t – state

\mathbf{o}_t – observation

\mathbf{a}_t – action

$\pi_\theta(\mathbf{a}_t | \mathbf{o}_t)$ – policy

$\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)$ – policy (fully observed)

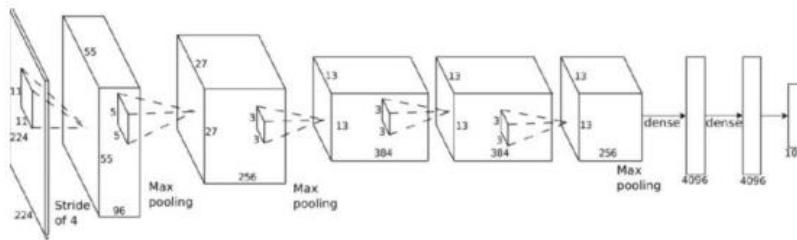


Images: Bojarski et al. '16, NVIDIA

DRL vs Imitation Learning



\mathbf{o}_t



$\pi_\theta(\mathbf{a}_t | \mathbf{o}_t)$



\mathbf{a}_t



\mathbf{o}_t
 \mathbf{a}_t

training data

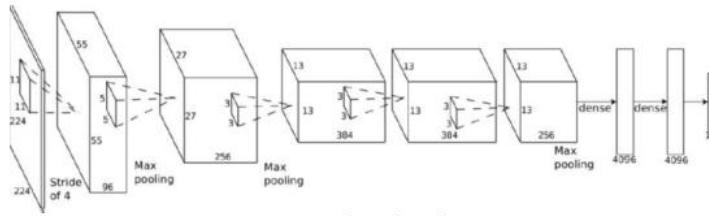
supervised learning

$\pi_\theta(\mathbf{a}_t | \mathbf{o}_t)$

Reward Function



\mathbf{o}_t



\mathbf{a}_t

which action is better or worse?

$r(\mathbf{s}, \mathbf{a})$: reward function

tells us which states and actions are better



high reward



low reward

\mathbf{s} , \mathbf{a} , $r(\mathbf{s}, \mathbf{a})$, and $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ define
Markov decision process

Some Applications



robotics



autonomous driving



language & dialogue
(structured prediction)

business operations



finance



Atari games:



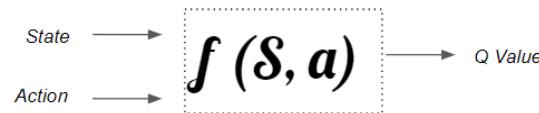
Real-world robots:



Beating Go champions:

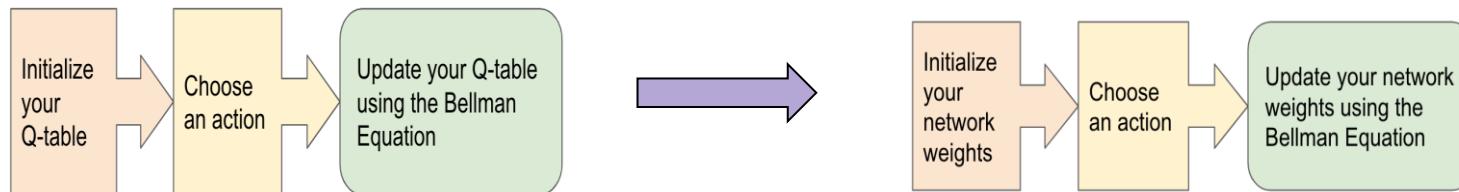
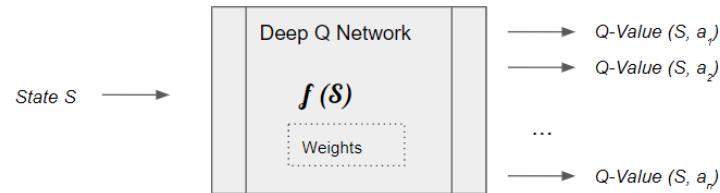
Deep Q Network (DQN)¹

Q Network



Neural Nets are the best Function Approximators

Steps



¹ Paper : www.nature.com/articles/nature14236 Code : sites.google.com/a/deepmind.com/dqn/

Deep Q Network

Q-Table

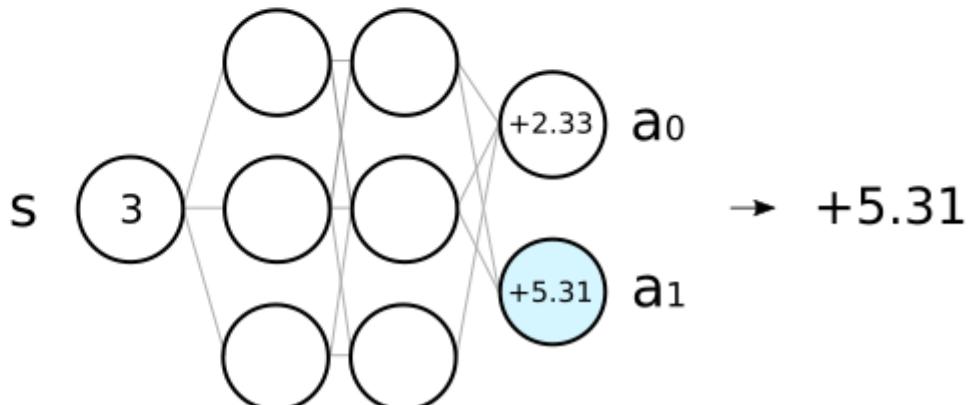
$Q(s, a) \rightarrow Q(3, 1) \rightarrow$

	s_0	s_1	s_2	s_3	s_4	
a_0	+4.21	+3.24	+1.84	+2.33	+3.73	
a_1	+2.53	+7.44	+3.34	+5.31	+6.22	

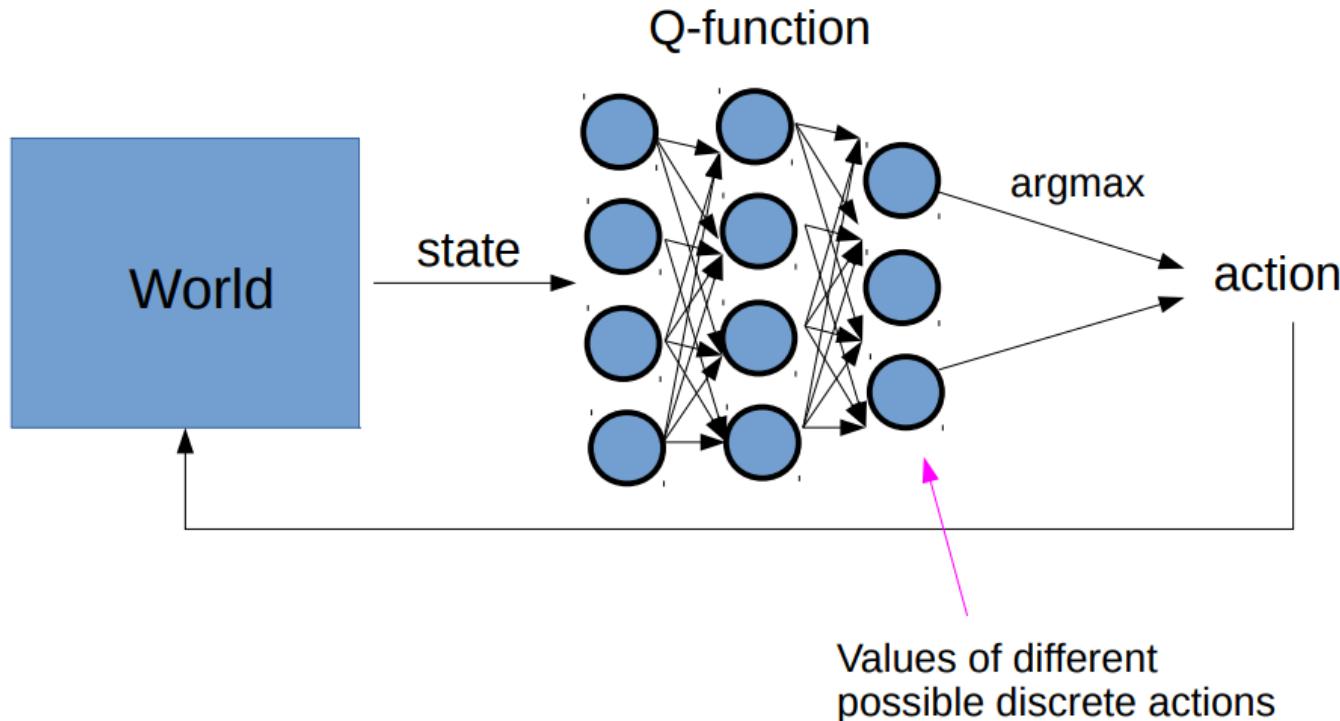
→ +5.31

DQN

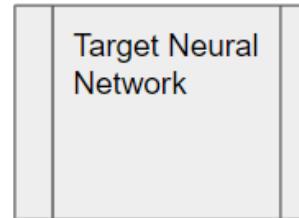
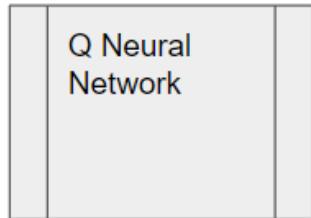
$Q(s, a) \rightarrow Q(3, 1) \rightarrow$



Deep Q Network



DQN Architecture Components



Experience Replay interacts with the environment to generate data to train the Q Network

Deep Q Network

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function \hat{Q} with weights $\theta^- = \theta$

For episode = 1, M **do**

 Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

For $t = 1, T$ **do**

 With probability ε select a random action a_t

 otherwise select $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$

 Execute action a_t in emulator and observe reward r_t and image x_{t+1}

 Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

 Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

 Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

 Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

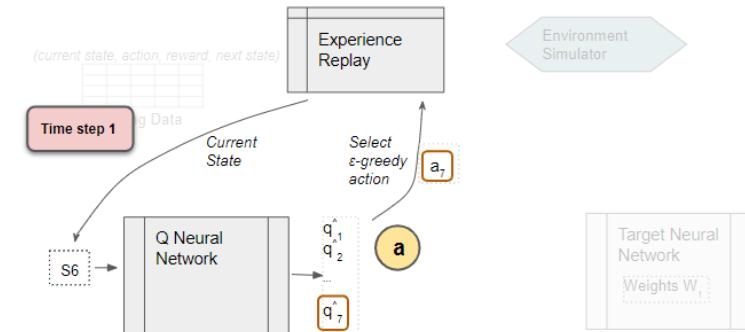
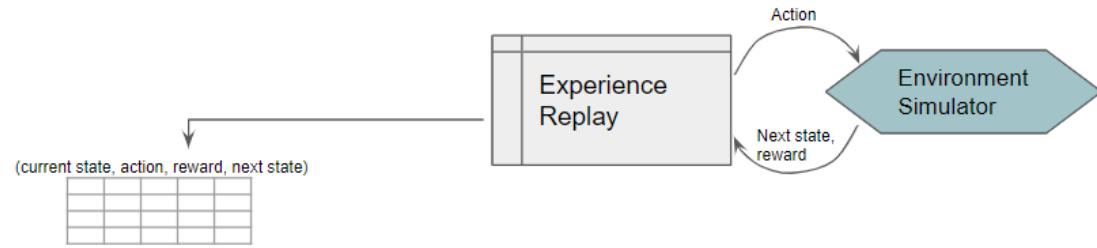
 Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ

 Every C steps reset $\hat{Q} = Q$

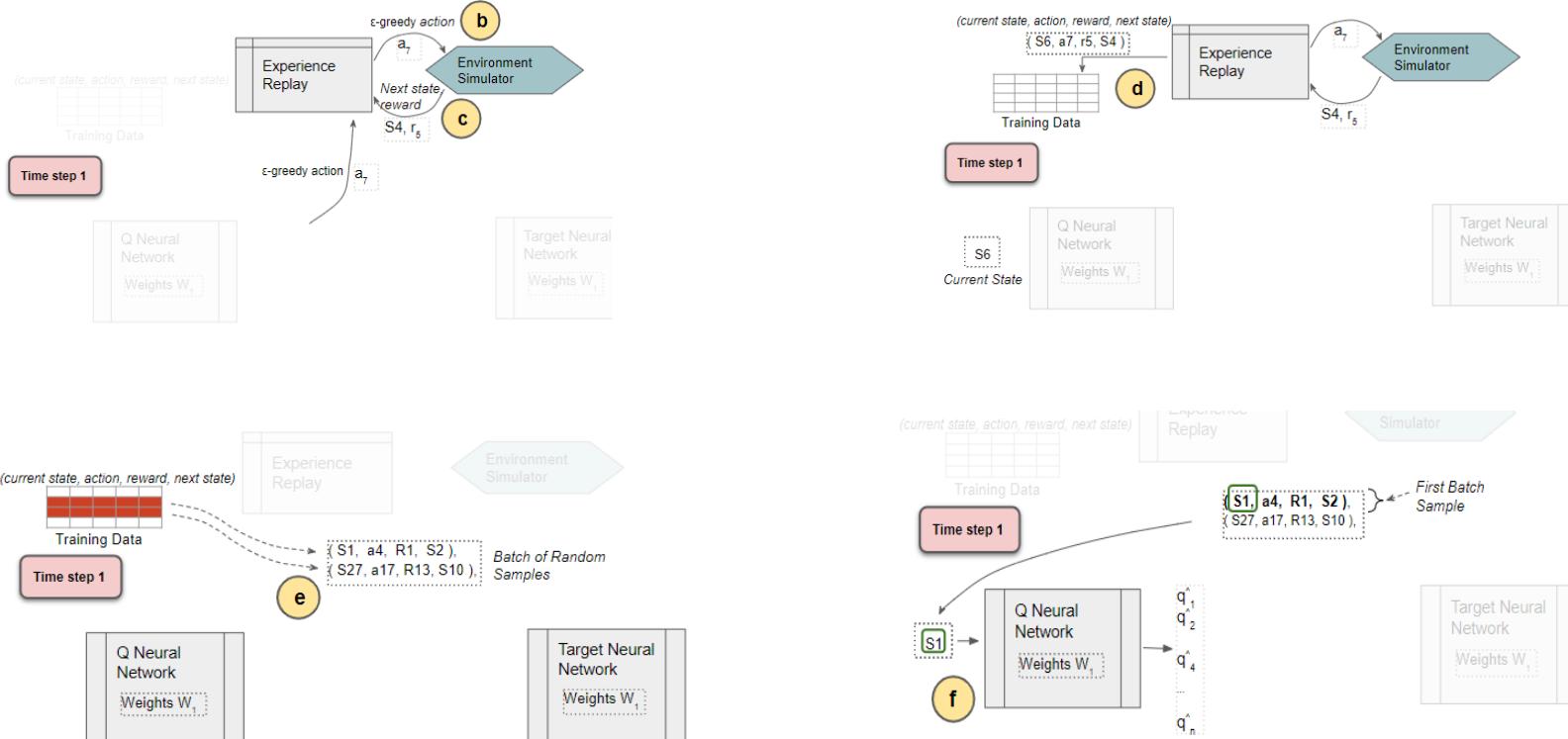
End For

End For

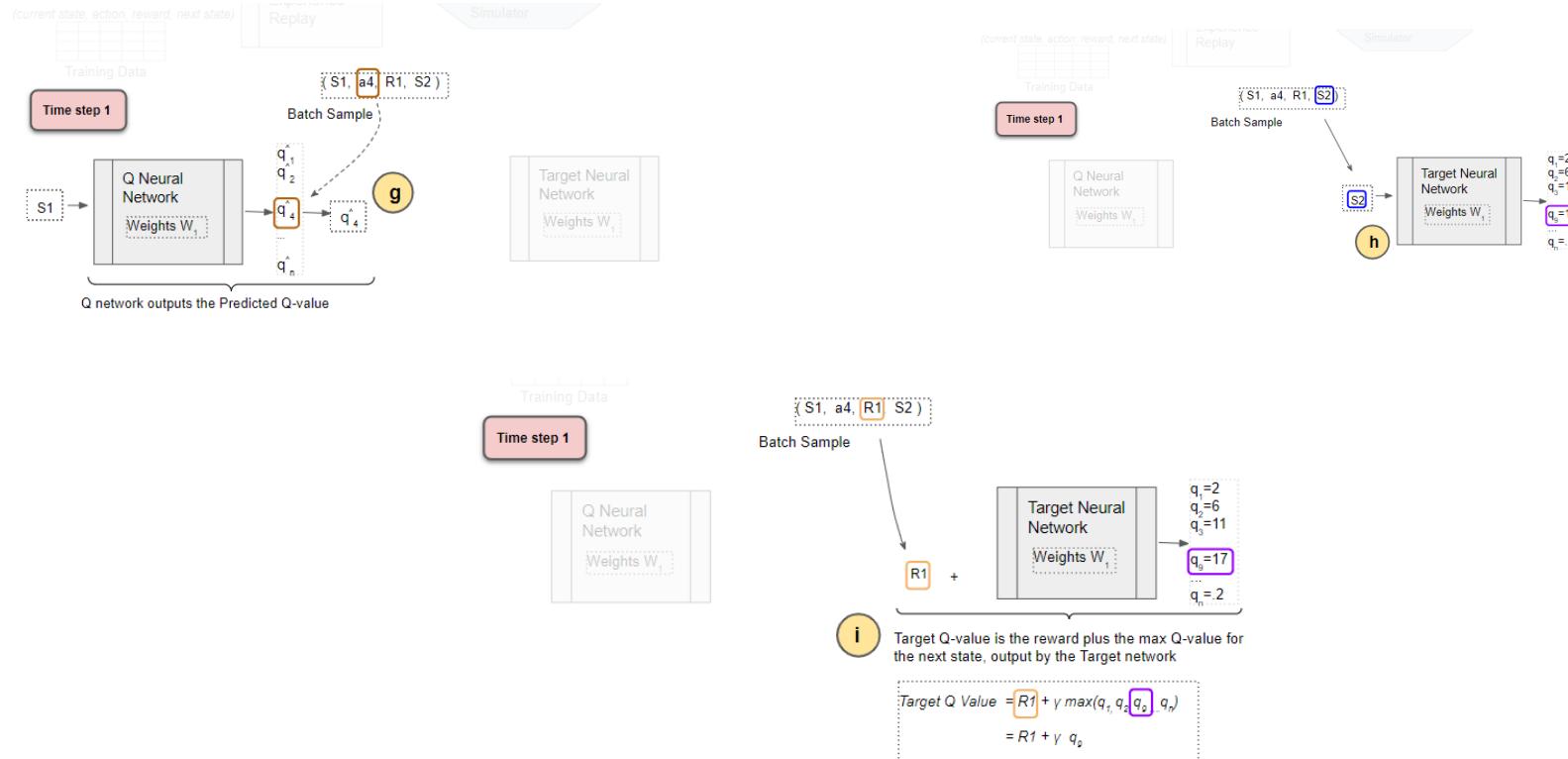
DQN



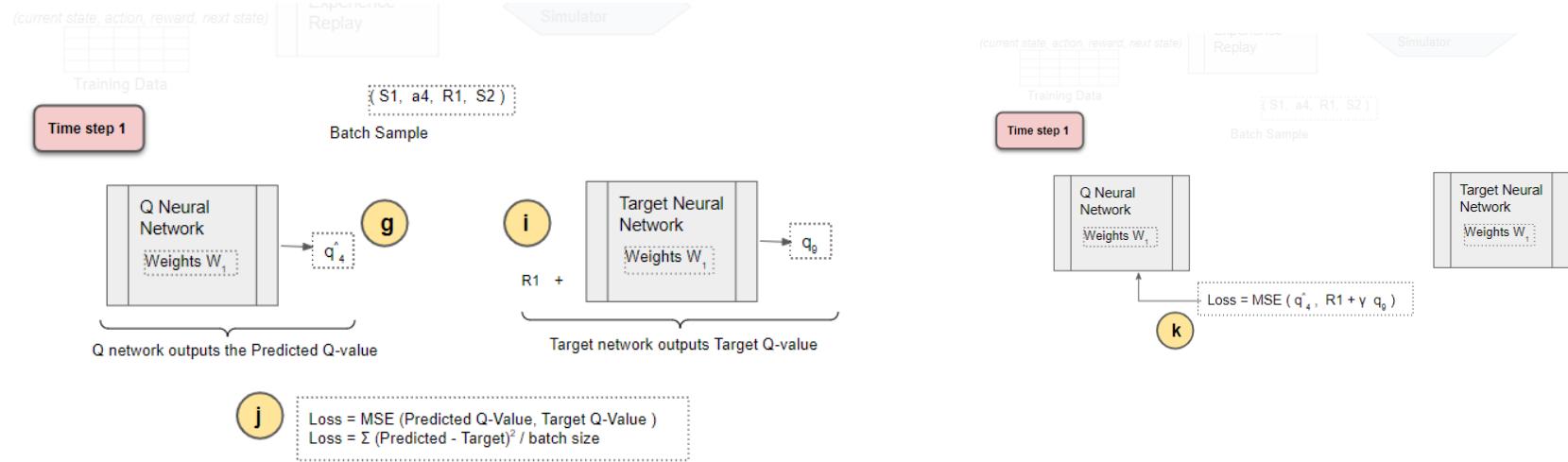
DQN



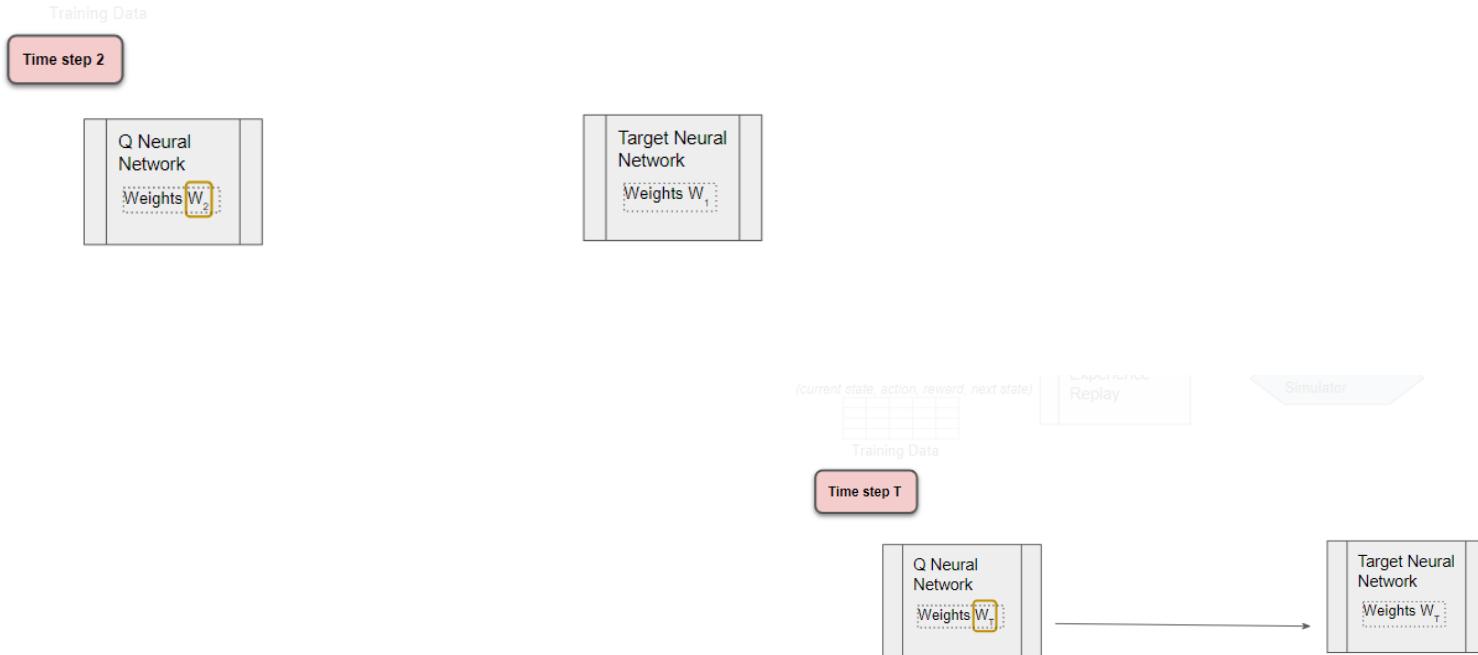
DQN



DQN



DQN



An Application : Dialogue Policy Learning

Task Oriented Dialogue System



Could you help me find a train to **Cambridge** on **Wednesday**?



Sure! What station would you like to leave from? And when would you like to depart?



London King's Cross. I was wondering if there are any trains that **arrive by 3pm**.

Slot Description	Value
Train destination	Cambridge
Train departure	London King's Cross
Time the train should arrive by	3pm
Time the train should leave by	(unspecified)
Day the train should run	Wednesday
Intent Description	Status
Check train schedules	
Book a train ticket	Active
Reschedule a train ticket	

Steiner et al., 2022

Open ended Dialogue System

Persona

I bought my first home.
I love to barbecue.
I live in Springfield.
I'm a writer.

Persona

I weight 300 pounds.
I am not healthy.
I am a man.
I like The Godfather.



Hello how are you, I am new to the Springfield area.



Hi! Seen any good movies lately?



I have been to the movies.



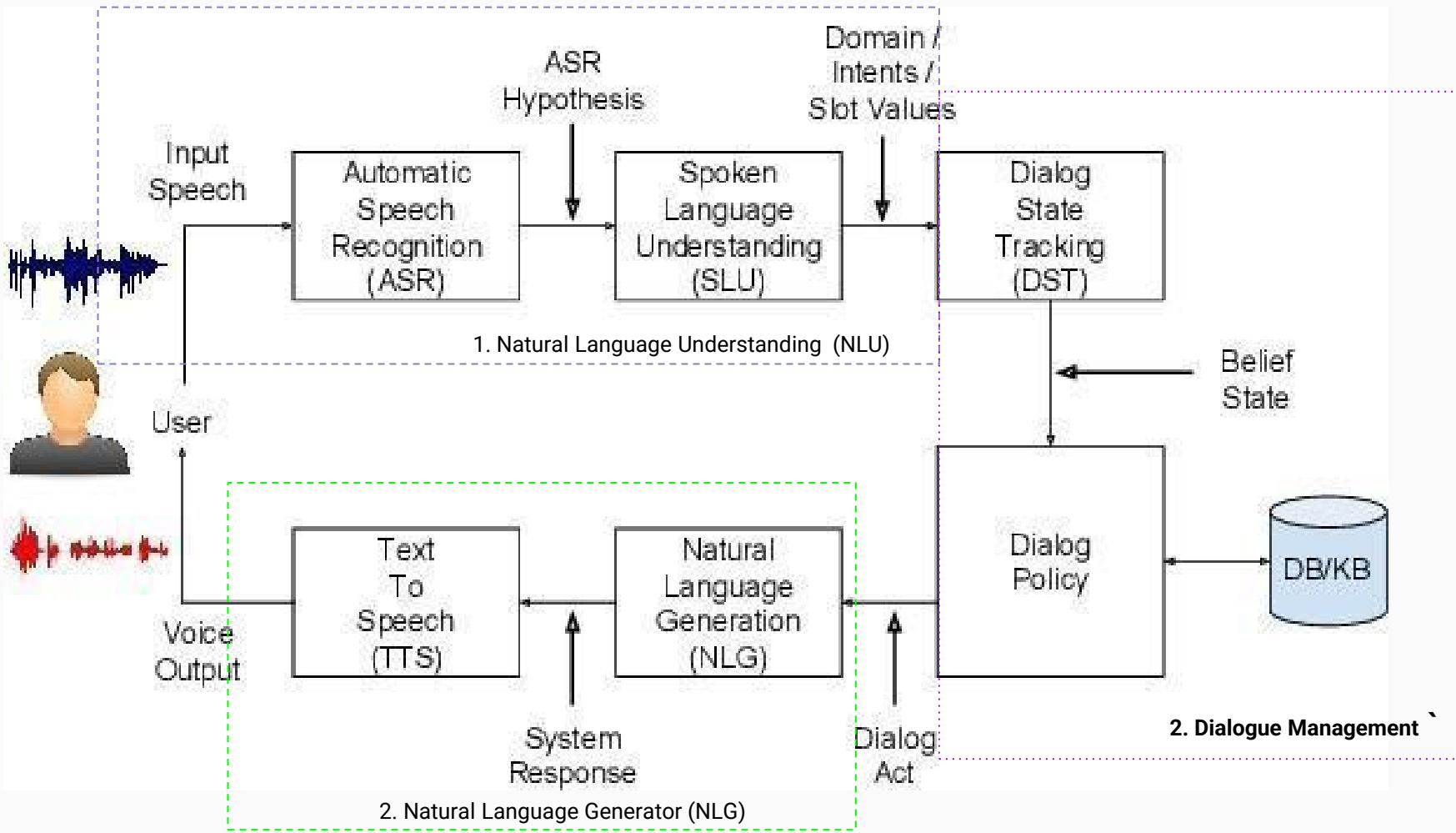
I love The Godfather, one of my favorites! Was that filmed?



I don't believe so. I don't watch movies more of a writer.

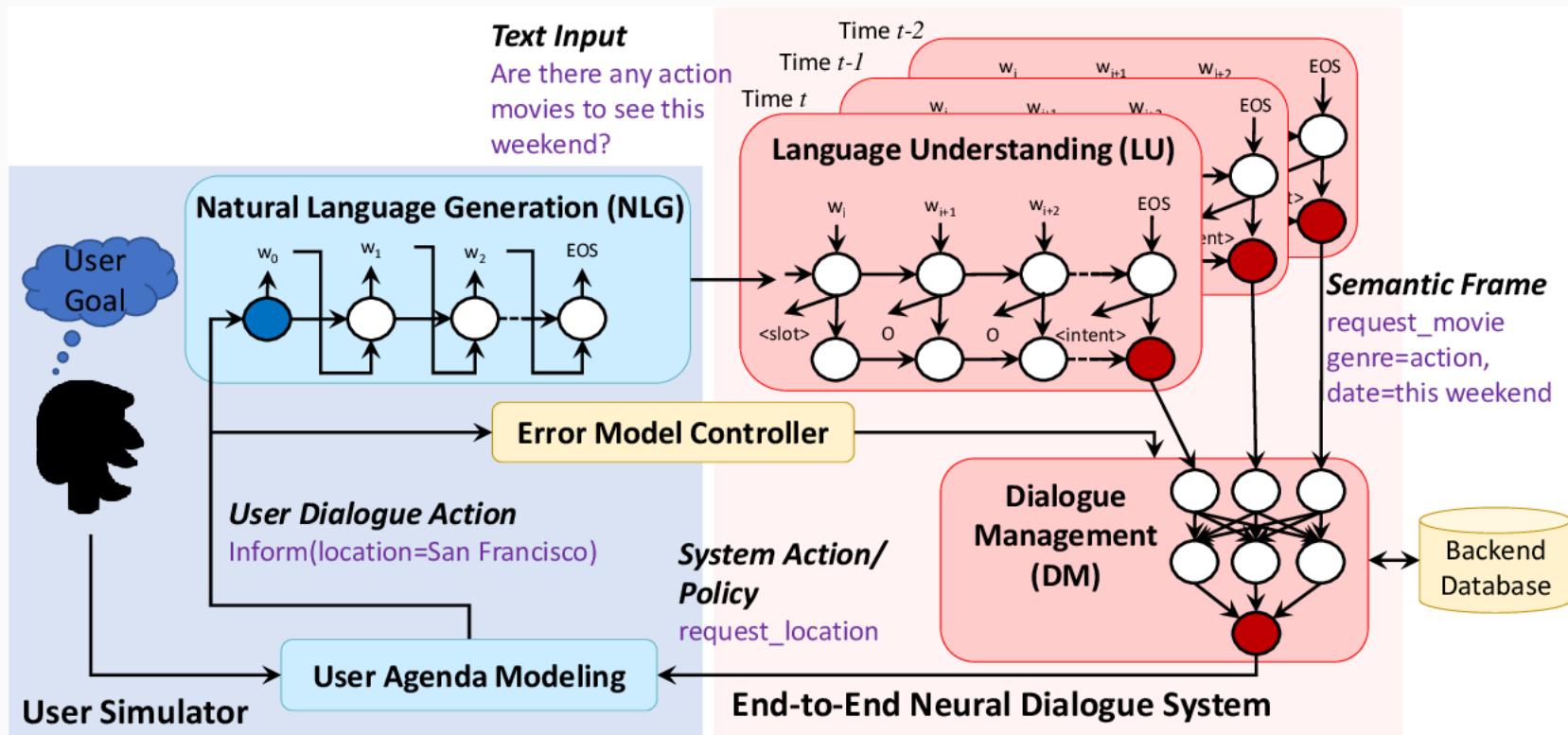


What do you write? Any diet books ? I am not very healthy.



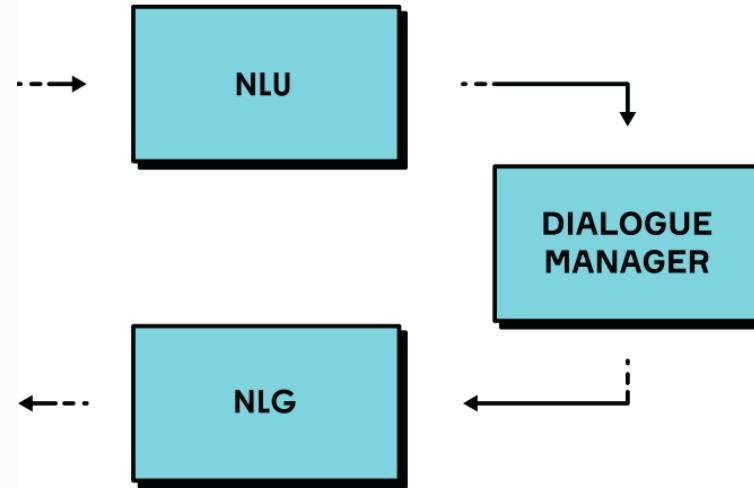
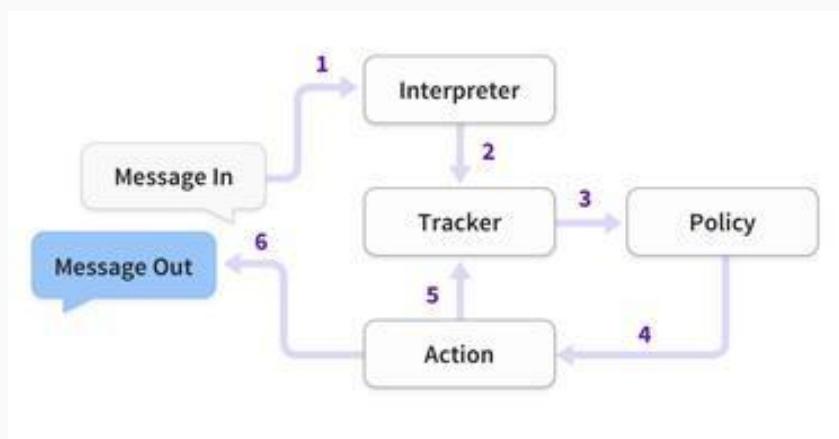
Framework for End to End Dialogue System

Task Oriented Dialogue System



Dialogue Management

- Central module of Dialogue System
- Decision maker
 - Given dialogue context , what assistant should say



Multimodal Dialogue Policy Learning for Dynamic and Co-operative Goal Setting

Problem : Goal unavailability ; existing task oriented agents completely flatter in case of goal unavailability scenarios. Text input and task goal (phone specification)

Idea : In real word, agents do not give up in goal unavailability scenarios, they find a very close and servable goal and persuade end users for the new goal.

User : Hi, can you help me in buying a mobile ?.

Agent : Sure, what kind of features do you want in your phone?

User : It should have 32 GB internal ram and at least 13 MP camera Quality.

Agent : Is that all features that you need?

User : And color should be gold.

Agent : Congrats, there is a perfect phone for you. You should buy this samsung galaxy C7. It has 16 MP camera quality with 32 GB internal capacity.



[Preq(), sentiment = neutral]

[Sreq(), sentiment = neutral]

[Inform(), sentiment = neutral]

[SpDone(), sentiment = neutral]

[Inform(), sentiment = neutral]

[Result(r, G),sentiment = neutral]

User : No, I do not like this color, I want something in this color ?



[Feedback(), sentiment = negative]

Agent : Can you please confirm that you want phone in rose gold color ?

User : Yes, you are right, I love this color.

Agent : Then, surely you will like Le max 2 by LeEco, which contains 32 GB Internal ram with 21 MP camera quality.



[Confirm(), sentiment = neutral]

[Confirm(), sentiment = positive]

User : Yes, it is nice, how much does it cost ?

Agent : It costs only 350 euro.

User : Oh, it too costly for me. I need a phone within 250 euro.

Agent : No issue, you should buy Le pro3 elite which costs only 280 euro. It has highly rated camera quality of 16 MP with 32 GB internal capacity. It is highly recommended as per your requirements.



User : Its really cool, please book it for me. Thanks

Agent : You are most welcome. Thanks for buying from us.

[Feedback(), sentiment = negative]

[Result(r'', G''), sentiment = neutral]

[BookingReq(), sentiment = positive]

[done() , sentiment = neutral]

Contribution & Problem Formulation

Contributions :

- We propose a novel framework for Dialogue Policy Learning, which utilizes a **multimodal data enabled state representation** and new module called **Goal Manager for handling users' dynamic goals and goal unavailability scenarios**.
- A **unique Markov Decision Process (MDP)** formulation with a multi-modal semantic state space has been proposed, which incorporates three different reward functions namely task based, sentiment and persona based reward.
- A **large-scale annotated multi-modal dialogue dataset** for goal shift and cooperative goal setting has been created.

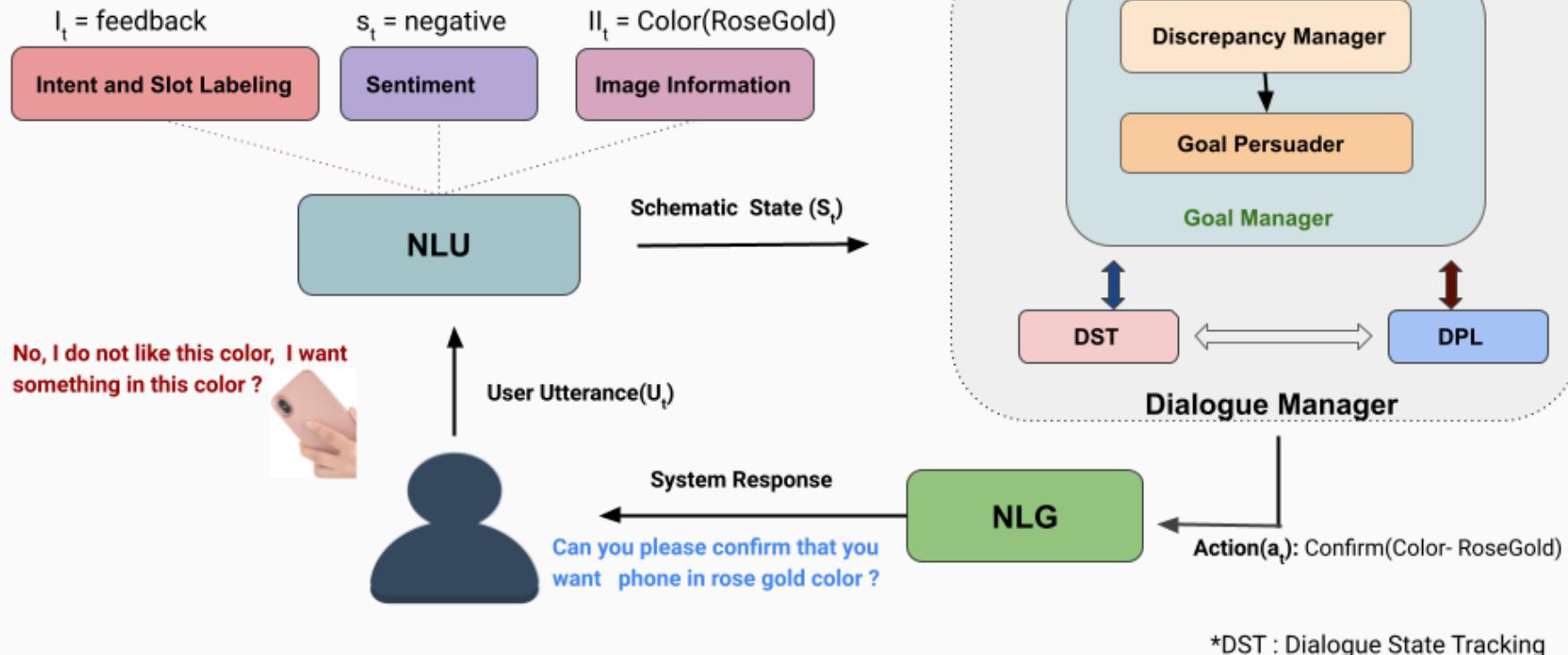
Problem : To learn an optimal dialogue policy (π^*), which estimates probability distribution over action space for a given dialogue context.

MDP (S, a, S', r)

$$a^* = \operatorname{argmax}_i P(a_i | S_t, GS_t, \pi^*)$$

$$\pi^* = \{\pi | V_\pi(s) \geq V_{\pi'}(s) \text{ for } \forall s \in S\}$$

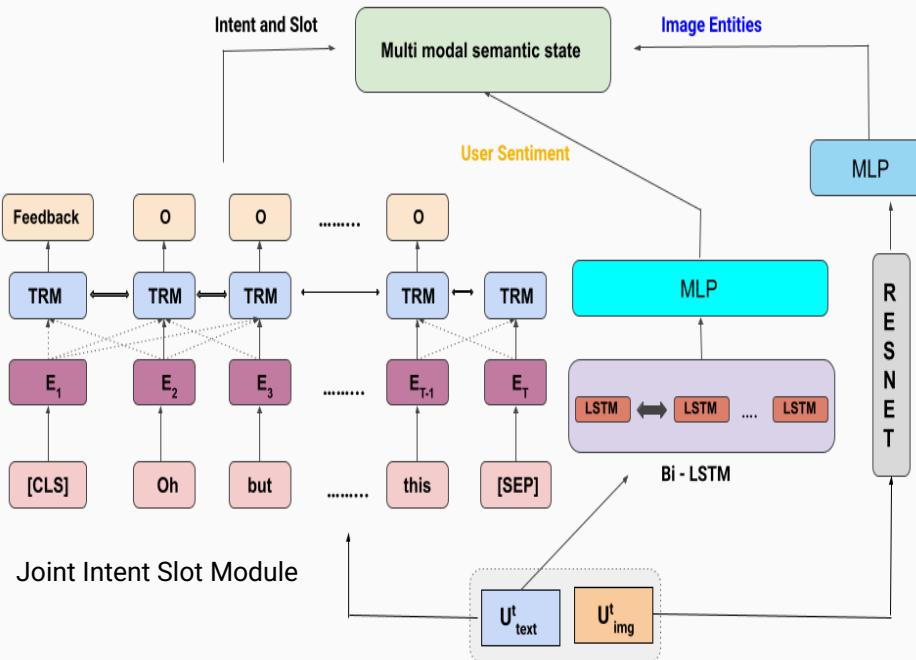
Proposed Model



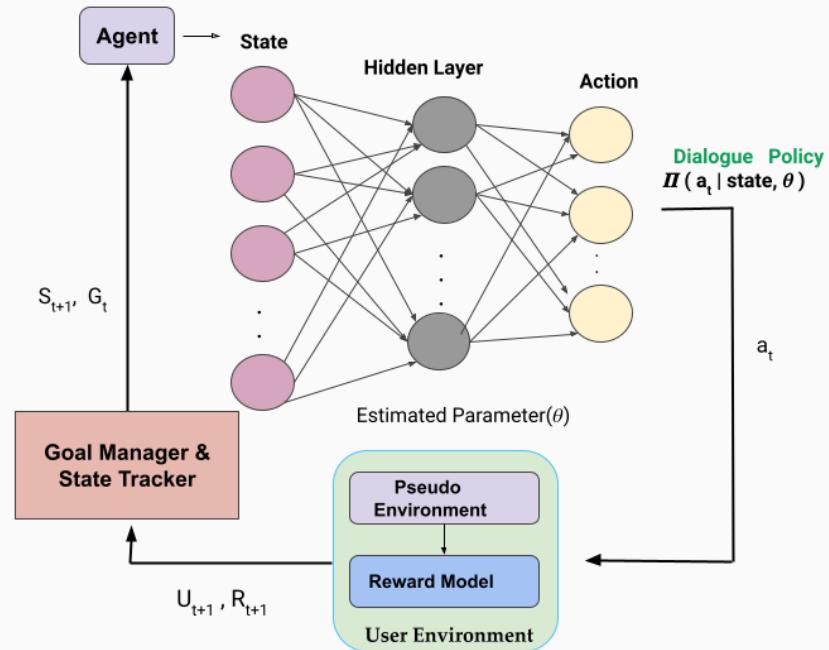
*DST : Dialogue State Tracking
*DPL : Dialogue Policy Learning

Proposed end to end multi modal framework for dynamic and co-operative goal setting

Architecture of NLU and Dialogue Management



Extraction and representation of multi modal utterance semantic



Dialogue policy learning setting with incorporated GM module

$$\Pi : [S, G] \rightarrow A$$

$$a^* = \operatorname{argmax}_a Q^*([S, G], a); \quad a \in A$$

$$Q^*([S, G], a) = r + \operatorname{argmax}_{a'} Q^*([S', G'], a') \quad a' \in A$$

$$Q^*([S, G], a) = r + V([S', G'])$$

Reward Model

- Reinforcing task specific behavior i.e., slot filling and task accomplishment using Task based Reward (TR)

$$TR = \begin{cases} +wtr_1 * (MaxLenLimit - n) & \text{if success} \\ -wtr_2 * MaxLenLimit & \text{if failure} \\ wtr_3 * (|LSlt' - LSlt|) & \text{if } (|LSlt' - LSlt|) \\ -wtr_4 & \text{otherwise} \end{cases}$$

- Imparting user adaptiveness behavior sensed and guided through user sentiment/Sentiment Reward (SR)

$$SR = \begin{cases} wsr_1 * (s - 1) & \text{if } s < 0.5 \\ wsr_2 * (4 - p) * (1 - s) & \text{if } s < 0.5, D=-1, p \leq 3 \\ wsr_3 * p * (s - 1) & \text{if } s < 0.5, D=-1, p > 3 \\ s & \text{if } s = 0.5 \text{ (neutral)} \\ wsr_4 * (s - 0.5) & \text{otherwise (positive)} \end{cases}$$

- Persona based Reward (PR) aims to provide persona aware persuasive argument for alleviating goal conflicts

$$PR = \begin{cases} +wpr_1(3 - p) & \text{if } D = -1, P_{ij} \in G, PT \in P_{ij} \\ -wpr_2 & \text{if } D = -1, P_{ij} \in G, PT \notin P_{ij} \\ p * (s - 0.5) & \text{Otherwise} \end{cases}$$

Dataset

Entries	Value
Domain	Electronic
Task	Phn Info, PhnBooking
Total No. of Dialogues	1000
Total No. of Utterances	8335
No. of Samples in KB	2697
No. of Attributes(Slots) in KB	18
Examples of Attributes	Brand, Ram,color etc.
Image Categories	Color, Brand, Style, Shape, Type

Entries	Values
Total no. of dialogues	1000
Total no. of utterances	8335
No. of instances in KB	2697
No. of slots	18

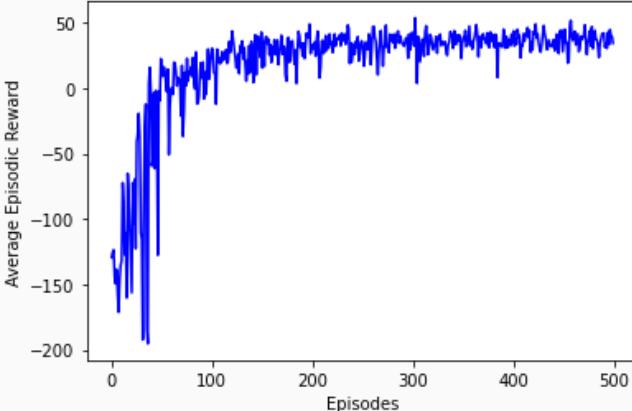
Categories	Count
Color	417
Brand	326
Keypad/Touchpad	438
Slide	555
Shape type	125
Total	1861



Image category	List of attributes
Brand	Apple, Huawei, MOTO, Samsung
Color	Black, Blue, Glacier white, Rose gold, Silver, Yellow
Type	Flip, Slide
Style	Keypad
Shape	Landscape

Images have been collected from GSMAreana and open source platforms.

Results



Learning Curve : Avg. Episodic Reward over Episodes

Task	Accuracy	F1-Score
Intent Classification	93.11%	0.874
Slot Tagging	87.39%	0.866

Performance of Intent and Slot Module

Model	Accuracy(%)	F1 - Score
Last layer + 1 FC	0.5814	0.573
Last layer + 2 FC	0.6129	0.6016
Last layer + 3 FC	0.6477	0.6328

Performance of Image Identification Module (ResNet 152 model has been used for image feature extraction)

Agent	Success rate	Dialogue length	Average Reward
Random Agent	0.002	17.24	-240.41
Rule Agent	0.000	14.00	-234.26
HDRL-M	0.071	15.10	-1.34*
UDP	0.468	14.81	-72.50
UPDP _{TR}	0.672	11.04	31.55
UPPD _{TR+PR}	0.766	9.92	35.47
MM-UPPD _{TR+PR}	0.784	8.76	36.28
MM-UPPD_{TR+PR+SR}	0.793	8.56	37.30

Comparative study of performances of different baselines and the proposed Multi-Modal Unified Persona aware Persuasive Dialogue Policy(MM-UPPPDP)

Findings and Analysis

- The proposed agent continues to serve users even in case of goal unavailability and persuades them on some persona aligned feature.
- TR + SR + PR \Rightarrow Task oriented, user adaptive and persuasive behavior
- Intent and Slot Module Failure
User - It should have 16 GB internal ram, Predicted Slot: Ram - 16 GB.
- Image Identification Error



Predicted : Yellow, Tag : Apple



Predicted : Keypad, Tag : Slide

Double DQN

Problems with DQN

- **Network Stability** : Same network for action selection and policy evaluation
- **Overestimation** : In the beginning of the training we don't have enough information about the best action to take. Therefore, taking the maximum q value (which is noisy) as the best action to take can lead to false positives. If non-optimal actions are regularly given a higher Q value than the optimal best action, the learning will be complicated

Double DQN

To Remove upward bias caused by $\max_a Q(s, a, w)$

- Current Q-network w is used to **select** actions
- Older Q-network w^- is used to **evaluate** actions

Action evaluation: w^-

$$l = \left(r + \gamma \underbrace{\max_{a'} Q(s', a', \mathbf{w}, \mathbf{w}^-)}_{\text{Action selection: } w} - Q(s, a, \mathbf{w}) \right)^2$$

Action selection: w

The original algorithm in “Double Q-learning” (Hasselt, 2010)

Algorithm 1 Double Q-learning

```
1: Initialize  $Q^A, Q^B, s$ 
2: repeat
3:   Choose  $a$ , based on  $Q^A(s, \cdot)$  and  $Q^B(s, \cdot)$ , observe  $r, s'$ 
4:   Choose (e.g. random) either UPDATE(A) or UPDATE(B)
5:   if UPDATE(A) then
6:     Define  $a^* = \arg \max_a Q^A(s', a)$ 
7:      $Q^A(s, a) \leftarrow Q^A(s, a) + \alpha(s, a)(r + \gamma Q^B(s', a^*) - Q^A(s, a))$ 
8:   else if UPDATE(B) then
9:     Define  $b^* = \arg \max_a Q^B(s', a)$ 
10:     $Q^B(s, a) \leftarrow Q^B(s, a) + \alpha(s, a)(r + \gamma Q^A(s', b^*) - Q^B(s, a))$ 
11:   end if
12:    $s \leftarrow s'$ 
13: until end
```

The original Double Q-learning algorithm uses two independent estimates $Q^{\{A\}}$ and $Q^{\{B\}}$.

With a 0.5 probability, we use estimate $Q^{\{A\}}$ to determine the maximizing action, but use it to update $Q^{\{B\}}$. Conversely, we use $Q^{\{B\}}$ to determine the maximizing action, but use it to update $Q^{\{A\}}$.

By doing so, we obtain an unbiased estimator $Q^{\{A\}}(\text{state}, \text{argmax}Q^{\{\text{next state, action}\}})$ for the expected Q value and inhibit bias.

- “Deep Reinforcement Learning with Double Q-learning” (Hasselt et al., 2015),

Algorithm 1 : Double Q-learning (Hasselt et al., 2015)

Initialize primary network Q_θ , target network $Q_{\theta'}$, replay buffer \mathcal{D} , $\tau \ll 1$

for each iteration **do**

for each environment step **do**

Observe state s_t and select $a_t \sim \pi(a_t, s_t)$

Execute a_t and observe next state s_{t+1} and reward $r_t = R(s_t, a_t)$

Store (s_t, a_t, r_t, s_{t+1}) in replay buffer \mathcal{D}

for each update step **do**

sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$

Compute target Q value:

$$Q^*(s_t, a_t) \approx r_t + \gamma \operatorname{argmax}_{a'} Q_{\theta'}(s_{t+1}, a')$$

Perform gradient descent step on $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$

Update target network parameters:

$$\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$$

Clipped Double Q-learning, in “Addressing Function Approximation Error in Actor-Critic Methods” (Fujimoto et al., 2018).

Algorithm 1 : Clipped Double Q-learning (Fujimoto et al., 2018)

```
Initialize networks  $Q_{\theta_1}$ ,  $Q_{\theta_2}$ , replay buffer  $\mathcal{D}$ 
for each iteration do
    for each environment step do
        Observe state  $s_t$  and select  $a_t \sim \pi(a_t, s_t)$ 
        Execute  $a_t$  and observe next state  $s_{t+1}$  and reward  $r_t = R(s_t, a_t)$ 
        Store  $(s_t, a_t, r_t, s_{t+1})$  in replay buffer  $\mathcal{D}$ 
    for each update step do
        sample  $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$ 
        Compute target Q value:
            
$$Q^*(s_t, a_t) \approx r_t + \gamma \min_{i=1,2} Q_{\theta_i}(s_{t+1}, \operatorname{argmax}_{a'} Q_{\theta_i}(s_{t+1}, a'))$$

        Perform gradient descent step on  $(Q^*(s_t, a_t) - Q_{\theta}(s_t, a_t))^2$ 
        Update target network parameters:
            
$$\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$$

```

- In Clipped Double Q-learning, we follow the original formulation of Hasselt 2015. We have two independent estimates of the true Q value. Here, for computing the update targets, we take the minimum of the two next-state action values produced by our two Q networks; When the Q estimate from one is greater than the other, we reduce it to the minimum, avoiding overestimation.

Double DQN

DQN Target value

$$y_t^{DQN} = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta_t^-)$$

Rewrite it to Double Q form

$$y_t^{DQN} = r_{t+1} + \gamma Q(s_{t+1}, \arg \max_{a'} Q(s_{t+1}, a'; \theta_t^-); \theta_t^-)$$

Double Q learning Target

$$y_t^{Double} = r_{t+1} + \gamma Q(s_{t+1}, \arg \max_{a'} Q(s_{t+1}, a'; \theta_t); \theta_t')$$

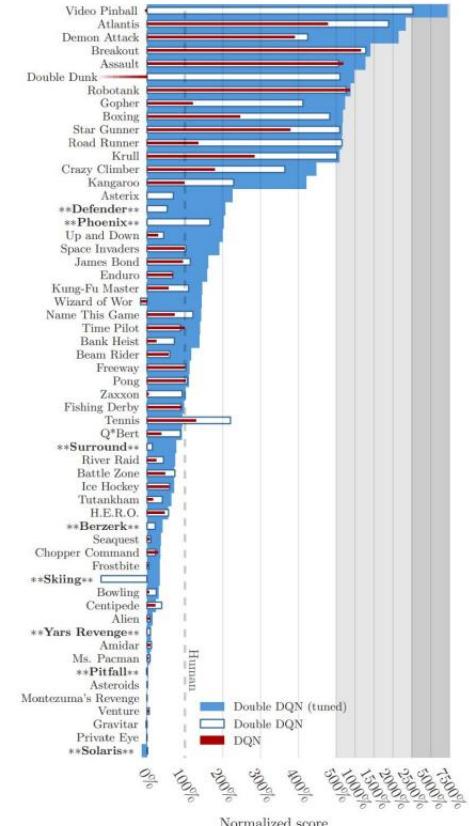
In double Q learning, two set of weights are maintained, one to determine the action selected by greedy policy and another to determine its Q value.

However, for DQN, only the offline set of weight is used to both choose the action and determine the target value. This can leads to overoptimism problem.

Double DQN

- Clearly outperform DQN, without additional computation cost or tuning.
- And the tuned version is even better

	DQN	Double DQN	Double DQN (tuned)
Median	47.5%	88.4%	116.7%
Mean	122.0%	273.1%	475.2%



Dueling DQN

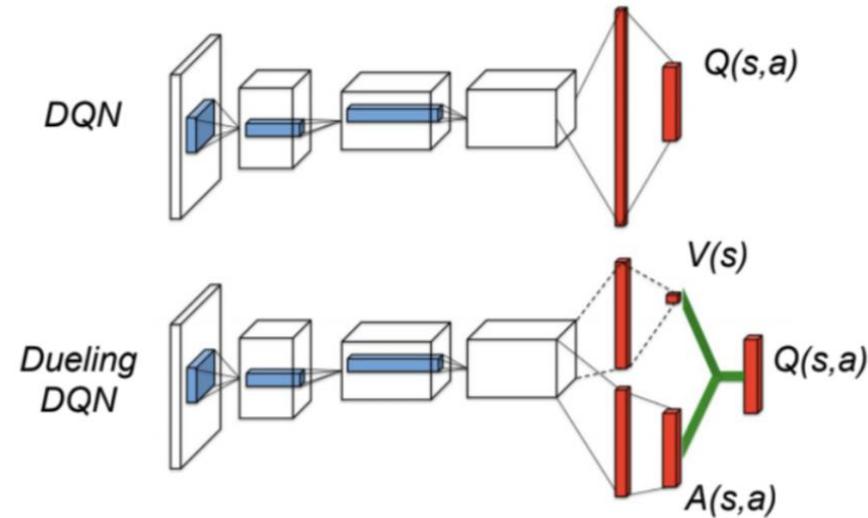
This algorithm splits the Q-values in two different parts, the value function $V(s)$ and the advantage function $A(s, a)$.

- Action-independent **value function** $V(s, v)$
- Action-dependent **advantage function** $A(s, a, w)$

$$Q(s, a) = V(s) + A(s, a)$$

$V(s)$: how much reward we will collect from state s

$A(s, a)$: how much better one action is compared to the other actions.



- The **Advantage** is a quantity obtained by subtracting the Q-value, by the V-value:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

Recall that the Q value represents the value of choosing a specific action at a given state, and the V value represents the value of the given state regardless of the action taken. Then, intuitively, the Advantage value shows how advantageous selecting an action is relative to the others at the given state

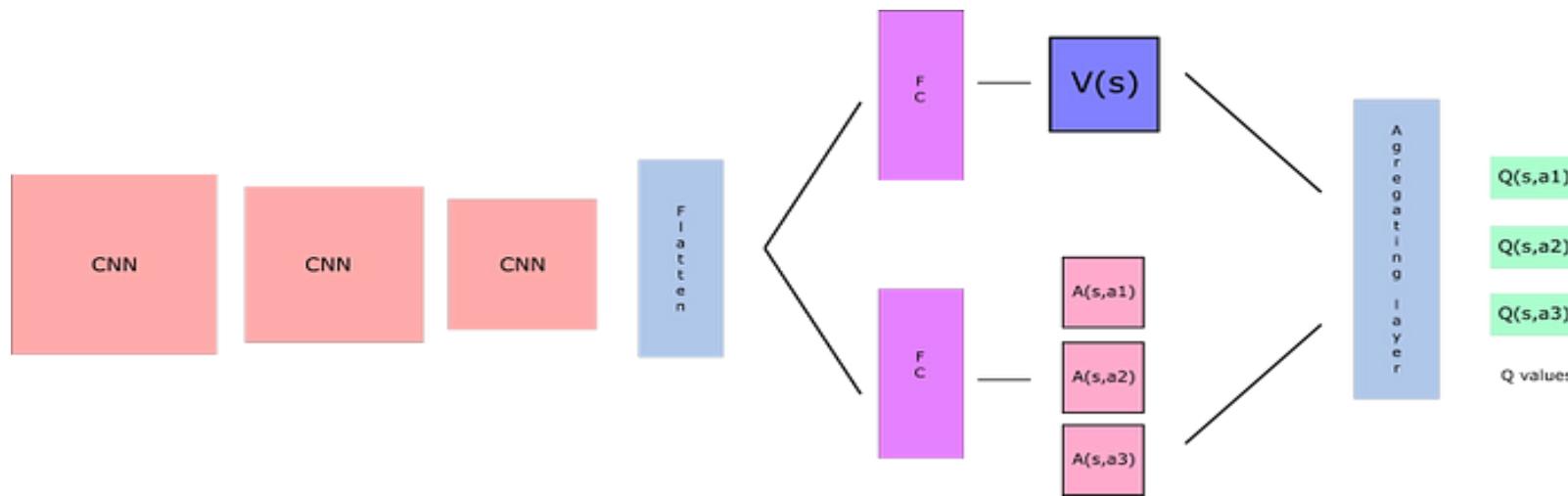
- **What Changes & Motivation**
- (Wang et al.) presents the novel *dueling architecture* which explicitly separates the representation of state values and state-dependent action advantages via two separate streams.
- The key motivation behind this architecture is that for some games, it is unnecessary to know the value of each action at every timestep. The authors give an example of the Atari game Enduro, where it is not necessary to know which action to take until collision is imminent.



- By explicitly separating two estimators, the dueling architecture can learn which states are (or are not) valuable, without having to learn the effect of each action for each state. Like the Enduro example, this architecture becomes especially more relevant in tasks where actions might not always affect the environment in meaningful ways.

Architecture

- Like the standard DQN architecture, we have convolutional layers to process game-play frames. From there, we split the network into two separate streams, one for estimating the state-value and the other for estimating state-dependent action advantages. After the two streams, the last module of the network combines the state-value and advantage outputs.



- It seems intuitive to just sum the two values, as immediately obvious from the definition of the advantage value:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha)$$

However, the authors present two issues about this method:

1. It is problematic to assume that V and A give reasonable estimates of the state-value and action advantages, respectively. Naively adding these two values can, therefore, be problematic.
2. The naive sum of the two is “unidentifiable,” in that given the Q value, we cannot recover the V and A uniquely. It is empirically shown in Wang et al. that this lack of identifiability leads to poor practical performance.

Therefore, the last module of the neural network implements forward mapping shown below:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + (A(s, a; \theta, \alpha) - \max_{a' \in |\mathcal{A}|} A(s, a'; \theta, \alpha))$$

- which will force the Q value for the maximizing action to equal V, solving the identifiability issue.
- Alternatively, as used in Wang et al.'s experiments, we can also use:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + (A(s, a; \theta, \alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a'; \theta, \alpha))$$

$$a^* = \arg \max_{a' \in \mathcal{A}} Q(s, a'; \theta, \alpha, \beta)$$

- **Training**
- Because the dueling architecture shares the same input-output interface with the standard DQN architecture, the training process is identical. We define the loss of the model as the mean squared error:

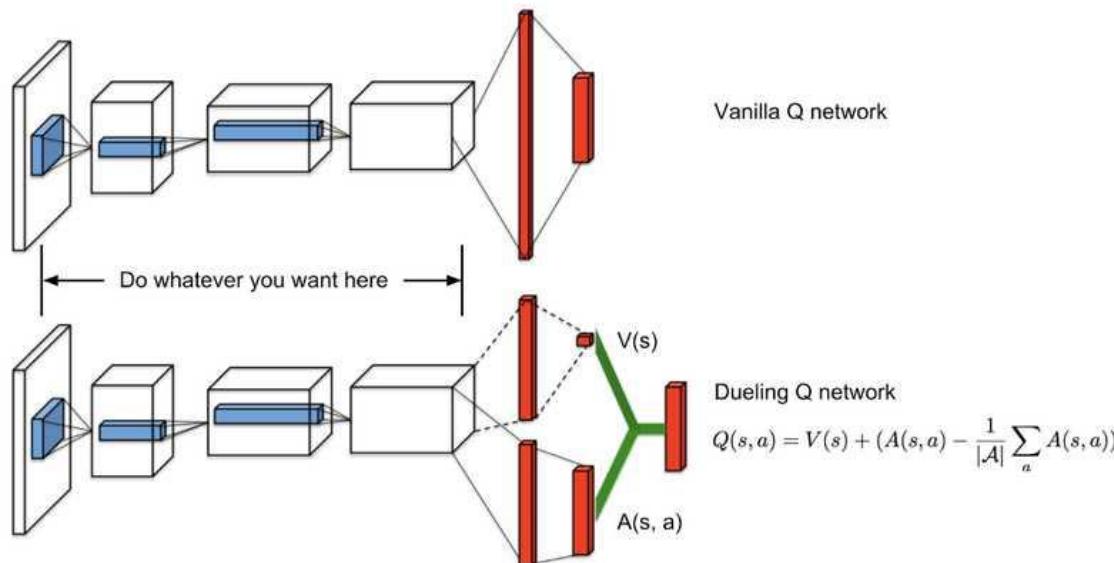
$$L(\theta) = \frac{1}{N} \sum_{i \in N} (Q_\theta(s_i, a_i) - Q'_\theta(s_i, a_i))^2$$

where $Q'_\theta = R(s_t, a_t) + \gamma \max_{a'_i} Q_\theta(s'_i, a'_i)$ (TD error)

and take the gradient descent step to update our model parameters.

Dueling DQN

The Dueling DQN also brings the concept of advantage or the incrementally higher utility of taking an action over the base state's absolute value.



DDQN and Dueling DQN

- Double DQN
 - Combining DQN and Double Q-learning to solve overoptimism problems for Q values
 - Provide a solid theoretical analysis of overestimation error bound in tradition Q learning
 - Demonstrate large estimation error in DQN and how DDQN fixes it and improve the performance using Atari games
 -
- Dueling DQN
 - Propose a decoupled estimator architecture for state value and action advantages, to replace previous single stream Q value estimator
 - The new architecture can be used together with many existing RL methods
 - In Atari games, Dueling DQN outperforms DDQN, and with prioritized replay, it is the SOTA in ALE benchmark

Experience Replay

- At each time step:
 - Take action a_t according to *epsilon-greedy policy*
 - Store experience $(s_t, a_t, r_{t+1}, s_{t+1})$ in *replay memory buffer*
 - Randomly sample *mini-batch* of experiences from the buffer

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
...
$s_t, a_t, r_{t+1}, s_{t+1}$

Prioritized Experience Replay

'DQN samples transitions from the replay buffer uniformly'

However, we should pay less attention to samples that is already close to the target. Thus, we should sample transitions that have a large target gap

- Based on the error value above (pick transitions with higher error more frequently)
- Rank them according to the error value and select them by rank (pick the one with higher rank more often)

Prioritized Experience Replay Memory¹

Prioritized Experience Replay is a type of experience replay in reinforcement learning where we In more frequently replay transitions with high expected learning progress, as measured by the magnitude of their temporal-difference (TD) error.

Let i be the index of the i -the tuple of experience (s_i, a_i, r_i, s_{i+1})

- p_i for new tuples is set to 0
- Priority of a tuple i is proportional to DQN error

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \mathbf{w}^-) - Q(s_i, a_i; \mathbf{w}) \right|$$

¹ Paper : <https://arxiv.org/abs/1511.05952v4>, Code : https://github.com/labmlai/annotated_deep_learning_paper_implementations

Prioritized Experience Replay Memory

Weight experience according to ``surprise'' (or error)

Store experience in priority queue according to DQN error

$$\left| r + \gamma \max_{a'} Q(s', a', w^-) - Q(s, a, w) \right|$$

Stochastic Prioritization

p_i is proportional to
DQN error

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

α determines how much prioritization is used, with $\alpha = 0$ corresponding to the uniform case.

Introduction to Policy-based Methods



Related Publications

A. Tiwari, **S. Saha**, P. Bhattacharyya (2022), ``A Knowledge Infused Context Driven Dialogue Agent for Disease Diagnosis using Hierarchical Reinforcement Learning", **Knowledge Based Systems (impact factor:8.038)** .

A. Tiwari, T. Saha, **S. Saha**, S. Sengupta, A. Maitra, R. Ramnani, P. Bhattacharyya (2021), ``A Persona Aware Persuasive Dialogue Policy for Dynamic and Co-operative Goal Setting", **Expert Systems with Applications (impact factor: 6.954, h5-index: 118)**.

A. Tiwari, T. Saha, **S. Saha**, S. Sengupta, A. Maitra, R. Ramnani, and P. Bhattacharyya(2021), ``Multi-Modal Dialogue Policy Learning for Dynamic and Co-operative Goal Setting" in **International Joint Conference on Neural Networks (IJCNN) 2021**, 18-22 July 2021

T. Saha, D. Gupta, **S. Saha**, P. Bhattacharyya (2021), ``A Unified Dialogue Management Strategy for Multi-Intent Dialogue Conversations in Multiple Languages", **ACM Transactions on Asian and Low-Resource Language Information Processing**

A. Tiwari, T. Saha, **S. Saha**, S. Sengupta, A. Maitra, R. Ramnani, P. Bhattacharyya (2021), ``A Dynamic Goal Adapted Task Oriented Dialogue Agent", **Plos One (h5 index: 175, impact factor: 2.74)**

T. Saha, N. Priya, **S. Saha** and P. Bhattacharyya (2021), ``A Transformer based Multi-task Model for Domain Classification, Intent Detection, and Slot-Filling" in **International Joint Conference on Neural Networks (IJCNN) 2021**, 18-22 July 2021.

T. Saha, S. Chopra, **S. Saha**, P. Bhattacharyya and Dr. P. Kumar (2021), ``A Large-Scale Dataset for Motivational Dialogue System: An Application of Natural Language Generation to Mental Health", in **International Joint Conference on Neural Networks (IJCNN) 2021**, 18-22 July 2021

T. Saha, A. Upadhyaya, **S. Saha**, P. Bhattacharyya (2021), ``Towards Sentiment and Emotion aided Multi-modal Speech Act Classification in Twitter", in **NAACL-HLT 2021**, June 6-11, 2021

T. Saha, S. Chopra, **S. Saha** and P. Bhattacharyya (2020), ``Reinforcement learning based personalized neural response generation", in **International Conference on Neural Information Processing (ICONIP) 2020**, 18-22 November, 2020

N. Priya, A. Tiwari and **S. Saha**(2021), ``Context Aware Joint Modeling of Domain Classification, Intent Detection and Slot Filling with Zero-shot Intent Detection Approach", in **28th International Conference on Neural Information Processing (ICONIP-2021)**, December 8 - 12, 2021, BALI, Indonesia

T. Saha, A. Patra, **S. Saha** and P. Bhattacharyya (2020), `` Towards Emotion-aided Multi-modal Dialogue Act Classification", In **ACL 2020**, July 5-10, 2020, Seattle, Washington

References

- Li, X., Chen, Y. N., Li, L., Gao, J., & Celikyilmaz, A. (2017, November). End-to-End Task-Completion Neural Dialogue Systems. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (pp. 733-743).
- Shi, W., & Yu, Z. (2018, July). Sentiment Adaptive End-to-End Dialog Systems. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1509-1519).
- Wang, X., Shi, W., Kim, R., Oh, Y., Yang, S., Zhang, J., & Yu, Z. (2019). Persuasion for good: Towards a personalized persuasive dialogue system for social good. arXiv preprint arXiv:1906.06725.
- Shi, W., Wang, X., Oh, Y. J., Zhang, J., Sahay, S., & Yu, Z. (2020, April). Effects of persuasive dialogues: testing bot identities and inquiry strategies. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (pp. 1-13).
- Tang, K. F., Kao, H. C., Chou, C. N., & Chang, E. Y. (2016, December). Inquire and diagnose: Neural symptom checking ensemble using deep reinforcement learning. In NIPS Workshop on Deep Reinforcement Learning.
- Kao, H. C., Tang, K. F., & Chang, E. (2018, April). Context-aware symptom checking for disease diagnosis using hierarchical reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).
- Wei, Z., Liu, Q., Peng, B., Tou, H., Chen, T., Huang, X. J., ... & Dai, X. (2018, July). Task-oriented dialogue system for automatic diagnosis. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 201-207).
- Liao, K., Liu, Q., Wei, Z., Peng, B., Chen, Q., Sun, W., & Huang, X. (2020). Task-oriented dialogue system for automatic disease diagnosis via hierarchical reinforcement learning. arXiv preprint arXiv:2004.14254.
- Rasmussen, K., Belisario, J. M., Wark, P. A., Molina, J. A., Loong, S. L., Cotic, Z., ... & Car, J. (2014). Offline eLearning for undergraduates in health professions: a systematic review of the impact on knowledge, skills, attitudes and satisfaction. Journal of global health, 4(1).
- Ramakrishnan, N., Vijayaraghavan, B. K. T., & Venkataraman, R. (2020). Breaking barriers to reach farther: A call for urgent action on tele-ICU services. Indian Journal of Critical Care Medicine: Peer-reviewed, Official Publication of Indian Society of Critical Care Medicine, 24(6), 393.
- Fox, S., & Duggan, M. (2013). Health online 2013. Health, 2013, 1-55.
- Conversational AI Market by Component (Platform and Services), Type (IVA and Chatbots), Technology (ML and Deep Learning, NLP, and ASR), Application, Deployment Mode (Cloud and On-premises), Vertical, and Region - Global Forecast to 2025." Markets and Markets, June 2020. Retrieved from: https://www.researchanmarkets.com/reports/5136158/conversational-ai-market-by-component-platform?utm_source=GNOM&utm_medium=PressRelease&utm_code=m3d7t3&utm_campaign=1426757+-+Global+Conversational+AI+Market+Analysis+2020-2025&utm_exec=joca220prd

How to learn an Optimal Policy ?

- How good is a state (S) ?

Expected cumulative reward from state, S

Intuition :

- **good state** : If the state, S , is more likely to lead to a path that ends up to a goal state.
- **bad state** : If the state S , is more likely to lead to a path that would not be end up at one of goal states.

How good or bad (intensity / subjectivity) ?

$$\begin{aligned} \text{Value } (S_{11}) &= p(S_{11} \rightarrow S_{21}) * 8 + p(S_{11} \rightarrow S_{22}) * 2 \\ &+ 50 \quad (\text{Assuming } \gamma = 1) \end{aligned}$$

$$\begin{aligned} &= 0.9 * 8 + 0.1 * 2 + 50 \\ &= 57.4 \end{aligned}$$

An Example

0.00	0.00	0.00	1.00
0.00		0.00	-1.00
0.00	0.00	0.00	0.00

VALUES AFTER 1 ITERATIONS

0.00	0.00	0.72	1.00
0.00		0.00	-1.00
0.00	0.00	0.00	0.00

VALUES AFTER 2 ITERATIONS

$\gamma = 0.9$, two terminal states with $R = +1$ and -1

An Example

0.00 ▶	0.52 ▶	0.78 ▶	1.00
0.00 ▶		▲ 0.43	-1.00
0.00 ▶	0.00 ▶	0.00	0.00 ▼

VALUES AFTER 3 ITERATIONS

0.37 ▶	0.66 ▶	0.83 ▶	1.00
▲ 0.00		▲ 0.51	-1.00
0.00 ▶	0.00 ▶	0.31	◀ 0.00

VALUES AFTER 4 ITERATIONS

An Example

0.51 ↗	0.72 ↗	0.84 ↗	1.00
0.27 ↑		0.55 ↑	-1.00
0.00 ↑	0.22 ↗	0.37 ↑	0.13 ←

VALUES AFTER 5 ITERATIONS

0.64 ↗	0.74 ↗	0.85 ↗	1.00
0.57 ↑		0.57 ↑	-1.00
0.49 ↑	0.43 ←	0.48 ↑	0.28 ←

VALUES AFTER 100 ITERATIONS

An example

0.64 ▶	0.74 ▶	0.85 ▶	1.00
▲ 0.57		▲ 0.57	-1.00
▲ 0.49	◀ 0.43	▲ 0.48	◀ 0.28

VALUES AFTER 1000 ITERATIONS