Neurorobotics

Module 1: Background and Foundations

Lecture 6: Reinforcement Learning and Prediction.
Braitenberg Vehicle 4; Markov decision processes; reinforcement learning; prediction

David Vernon
Carnegie Mellon University Africa

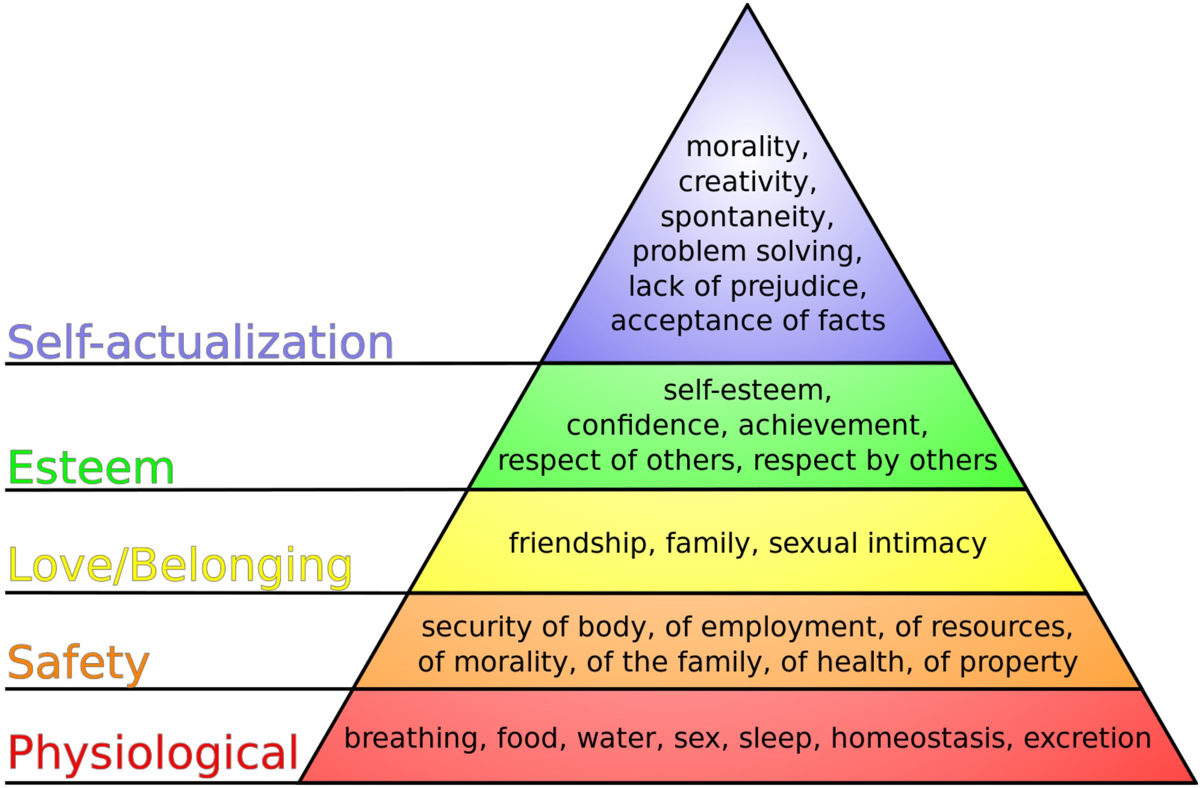
www.vernon.eu

Prologue

- Unsupervised learning is possible
  - When robots actively explore and sense the environment
  - Without receiving rewards or
  - Without seeking a goal

- Alternative forms of learning are needed when time and energy is limited
  - Need to choose what information to encode (i.e. learn) in order to survive and thrive
  - This choice is often based on a value system
    - Derived from evolution in an ecological niche
    - Derived from some cultural trait
    - Ranging from desire for food and shelter to adherence to moral codes and social norms

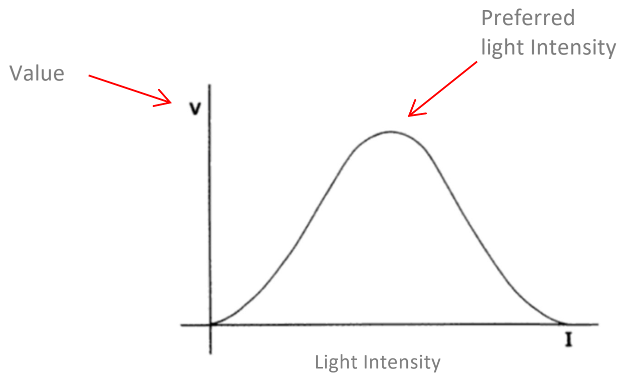
Maslow’s Hierarchy of Needs

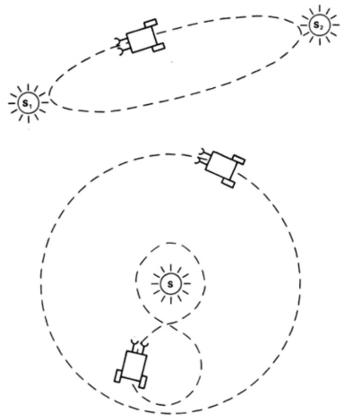


https://commons.wikimedia.org/wiki/File:Maslow%27s_hierarchy_of_needs.svg

Braitenberg Vehicle 4

- Braitenberg vehicles 2 and 3
  - Two opposing value systems
    - Attraction to light
    - Repulsion by light
- Braitenberg vehicle 4
  - (Slightly) more sophisticated value system
    - Preference for a specific amount of light
    - Strive to stay in a desired range





Markov Decision Processes

- How do learning mechanisms use value to acquire information?
- Markov decision process (MDP)
  - Formalization of the problem of using a reward
    - To learn the best action to take
    - For a particular environmental state

Markov Decision Processes

Markov decision process (MDP)

1. An agent can be in just one state at any give time

- The state is a unique descriptor of the environmental conditions relevant to the agent and the task
  - For example, location in the environment
  - For example, image of the environment taken by a robot's camera
- The state space should be carefully selected for the task
  - Minimum amount of complexity required to capture the desired effects

Markov Decision Processes

Markov decision process (MDP)

2. There is a defined set of actions available in a given state

   - The agent chooses a single action from that set

     – For example: steer left or steer right

Markov Decision Processes

Markov decision process (MDP)

3. When an action is taken, the outcome, i.e. the next state, is not certain

   - The transition from one state to another may be probabilistic

   - These probabilities can be learned through exploration

Markov Decision Processes

Markov decision process (MDP)

4. There may be a payoff associated with each state

- The payoff drives the agent to learn actions that maximize accumulated reward
- Payoffs
  - Rewards
  - Penalties
  - Positive of negative numeric values
  - Sparse (e.g., single global payoff for achieving a goal)
  - Dense (e.g., penalties for hitting obstacles, rewards for achieving intermediate tasks)
- Sparse payoffs make learning good actions more challenging
  - Requires large amount of exploration to reach a rewarding state

Markov Decision Processes

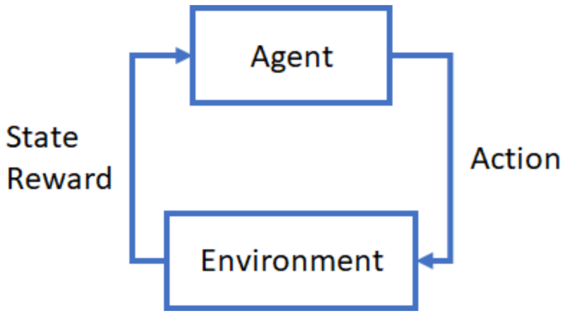
Markov decision process (MDP)

5. A policy defines a mapping from states to actions

   - A policy describes a strategy for selecting actions
   - Initial policy might be random (leading to random behavior)
   - As the agent learns transitions and rewards, the policy improves
   - Ideally, an optimal policy is learned

Markov Decision Processes

- Markov decision process (MDP)
  - The agent can only be in a single state
  - The agent only uses information from the current state to decide what action to take
    - The action is dependent only on the current state (Markov property)
    - The agent performs the action and potentially receives a reward from the environment
    - The agent transitions to new state
    - The process repeats until some termination state is reached



Reinforcement Learning

- MDPs provide a framework for formalizing problems in reinforcement learning
  - Learning how to maximize rewards while performing a task in the environment
  - Agent explores the environment in an unsupervised manner
    - To discover which actions lead to high value (maximum long term cumulative reward)
- Two types of reinforcement learning
  1. Model-free reinforcement learning
  2. Model-based reinforcement learning

Reinforcement Learning

Model-free reinforcement learning

- Does not create a model of the environment
  - Does not learn probabilities of state transitions
- Policy: chose the best action for a given state
  - Represent which states are good and bad
  - Not the reason why they are good or bad

Reinforcement Learning

Model-based reinforcement learning

- Creates an explicit model of the environment
  - Can be in the form of state transition probabilities
- May lead to a better policy and explanation of actions
  - Cost of learning
  - Cost of storing transition probabilities
  - Depends on complexity of the environment

Reinforcement Learning

There is evidence for both model-free and model-based reinforcement learning in the brain

Model-free Reinforcement Learning

Q-learning algorithm

- Q value is a measurement that relates
  - The cumulative reward
  - For taking the given action
  - In the given state
- There is a Q value for every possible state and action pair

Model-free Reinforcement Learning

Q-learning algorithm

- The Q values may be initialized to zero or random values
- As the agent explores the environment, its state changes with every action at each timestep
  - For each state transition from time step t to $t+1$
  - the Q value for each state-action pair is updated using the following rule:

Maximum Q value that could be reached in the next state by taking some action a

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

Updated Q value at time $t+1$

State at time t

Selected action at time t

Current Q value at time t

Learning rate $(0\text{-}1)$

Reward received at time t

Parameter to discount future reward: tradeoff between obtaining an immediate reward or waiting to get a larger reward in the future

Current Q value at time t

Model-free Reinforcement Learning

Q-learning algorithm

- The Q values may be initialized to zero or random values
- As the agent explores the environment, its state changes with every action at each timestep
  - For each state transition from timestep t to $t+1$
  - the Q value for each state-action pair is updated using the following rule:

Current Q value at time t

Temporal difference: the increase in Q value arising from a transition from the current state to the next state

$$Q^{new}(s_{t,}a_t) = \underbrace{Q(s_t, a_t)}_{} + \underbrace{\alpha(r_t + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t))}_{}$$

Model-free Reinforcement Learning

Q-learning algorithm

- The Q values may be initialized to zero or random values
- As the agent explores the environment, its state changes with every action at each timestep
  - For each state transition from timestep t to $t+1$
  - the Q value for each state-action pair is updated using the following rule:

Difference between the potential next Q value (given some action) and the current Q value (given the last action taken)

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha(\underbrace{r_t + \gamma max_a Q(s_{t+1}, a)}) - \underbrace{Q(s_t, a_t)})$$

Model-free Reinforcement Learning

Q-learning algorithm

- The Q values may be initialized to zero or random values
- As the agent explores the environment, its state changes with every action at each timestep
  - For each state transition from timestep t to $t+1$
  - the Q value for each state-action pair is updated using the following rule:

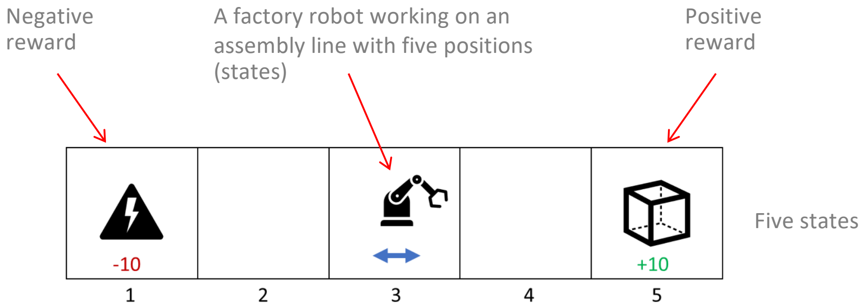
Combination of the reward gained at the current time and the best existing Q value for the next state (found by iterating through all possible actions that can be taken at state s_{t+1}) and discounted to lessen the weight of future rewards (future rewards are less predictable)

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha\left(\overbrace{r_t + \gamma max_a Q(s_{t+1}, a)} - Q(s_t, a_t)\right)$$

Model-free Reinforcement Learning

Q-learning algorithm example

- The robot can move left or right on the assembly line
  - Five positions / states: 1 – 5
  - Rightmost position / state
    - Positive reward
    - Position to be in to pick up an object
  - Leftmost position / state
    - Negative reward
    - Might be another robot at that position already



Q table after first transition from state 4 to state 5

	1	2	3	4	5
Left	N/A	0	0	0	0
Right	0	0	0	10	N/A

Q table after first transition from state 5 to state 4

	1	2	3	4	5
Left	N/A	0	0	0	5
Right	0	0	0	10	N/A

Fully trained Q table

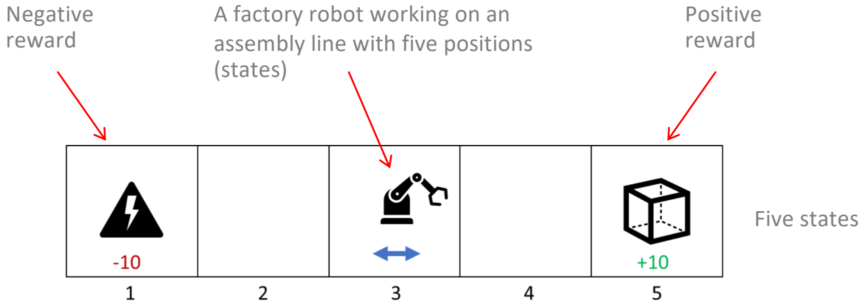
	1	2	3	4	5
Left	N/A	-9.33	1.33	2.67	1.33
Right	1.33	2.67	5.33	10.67	N/A

Model-free Reinforcement Learning

Q-learning algorithm example

Q value table

- One row for each action
- One column for each state
- Each cell captures $Q(s_t, a_t)$
- Each cell is Initialized to zero



Q table after first transition from state 4 to state 5

	1	2	3	4	5
Left	N/A	0	0	0	0
Right	0	0	0	10	N/A

Q table after first transition from state 5 to state 4

	1	2	3	4	5
Left	N/A	0	0	0	5
Right	0	0	0	10	N/A

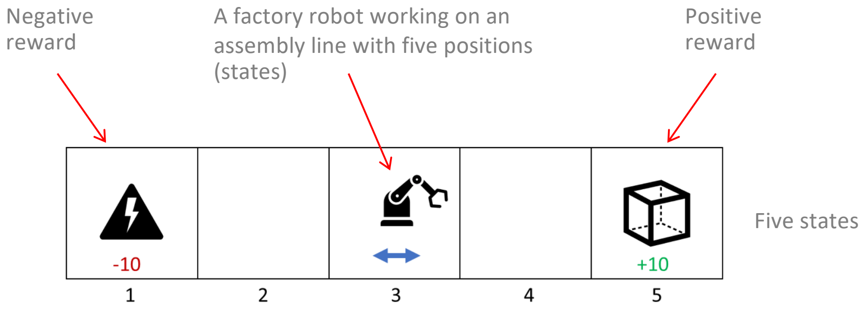
Fully trained Q table

	1	2	3	4	5
Left	N/A	-9.33	1.33	2.67	1.33
Right	1.33	2.67	5.33	10.67	N/A

Model-free Reinforcement Learning

Q-learning algorithm example

- The robot starts in the state/position 3
- It wanders around randomly
  - Applies the Q value update equation at each timestep
  - The Q values change only if a reward is received
  - At some point in this random walk, the robot moves to a location (state / position) where it receives a reward



Q table after first transition from state 4 to state 5

	1	2	3	4	5
Left	N/A	0	0	0	0
Right	0	0	0	10	N/A

Q table after first transition from state 5 to state 4

	1	2	3	4	5
Left	N/A	0	0	0	5
Right	0	0	0	10	N/A

Fully trained Q table

	1	2	3	4	5
Left	N/A	-9.33	1.33	2.67	1.33
Right	1.33	2.67	5.33	10.67	N/A

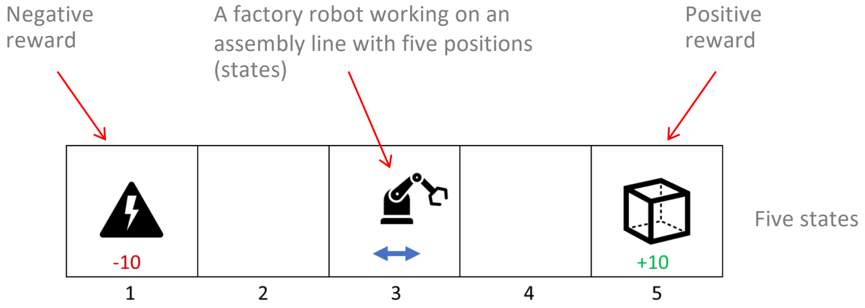
Model-free Reinforcement Learning

Q-learning algorithm example

- Let $\alpha = 1$ and $\gamma = 0.5$
- Assume the robot is in state 4 for the first time
- If it then moves right to state 5, the Q value table is updated as follows

$$Q^{new}(4,'right') \leftarrow \; 0 + 1(10 + .5 * max_a \, (Q(4,a)) - Q(4,a_t))$$
$$Q^{new}(4,'right') \leftarrow \; 0 + 1(10 + .5 * \; 0 - 0)$$
$$Q^{new}(4,'right') \leftarrow 10.$$

Should be $Q(5,a)$
because it is the next state S_{t+1}
reached by taking action '$right$'
This is either 0 or N/A



Q table after first transition from state 4 to state 5

	1	2	3	4	5
Left	N/A	0	0	0	0
Right	0	0	0	10	N/A

Q table after first transition from state 5 to state 4

	1	2	3	4	5
Left	N/A	0	0	0	5
Right	0	0	0	10	N/A

Fully trained Q table

	1	2	3	4	5
Left	N/A	-9.33	1.33	2.67	1.33
Right	1.33	2.67	5.33	10.67	N/A

Model-free Reinforcement Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    → Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    → $S \leftarrow S'$
  until S is terminal

The maximum Q value over all possible actions in the next state S'

R. S. Sutton and A. G. Barto, Reinforcment Learning – An Introduction, MIT Press, 2018.

Model-free Reinforcement Learning

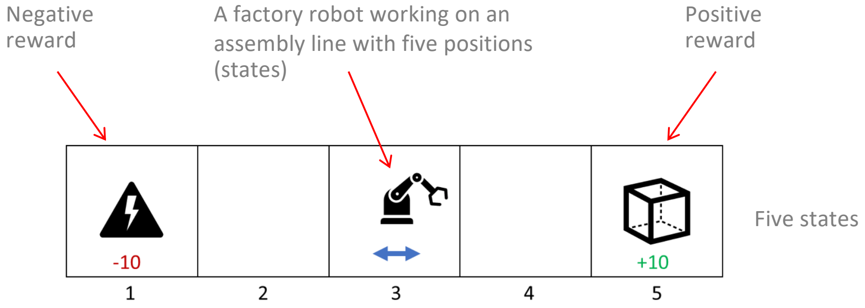
Q-learning algorithm example

- If, in state 5, it then moves left to state 4, the Q value table is updated as follows

$$Q^{new}(5,'left') \leftarrow 0 + 1(0 + .5 * max_a\,(Q(5,a)) - Q(5,a_t))$$
$$Q^{new}(5,'left') \leftarrow 0 + 1(0 + .5 * 10 - 0)$$
$$Q^{new}(5,'left') \leftarrow 5.$$

Should be $Q(4,a)$ because it is the next state S_{t+1} reached by taking action '$left$'

This is either 0 or 10



Q table after first transition from state 4 to state 5

	1	2	3	4	5
Left	N/A	0	0	0	0
Right	0	0	0	10	N/A

Q table after first transition from state 5 to state 4

	1	2	3	4	5
Left	N/A	0	0	0	5
Right	0	0	0	10	N/A

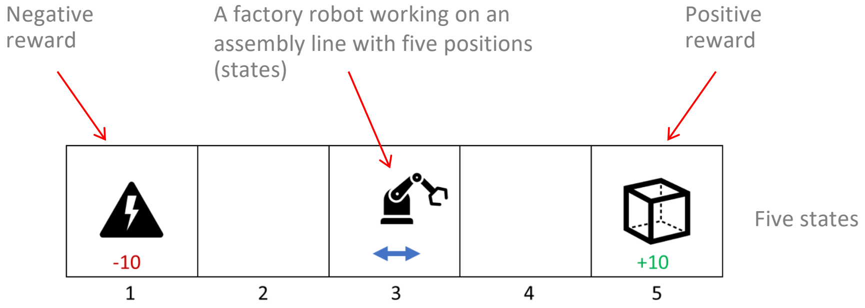
Fully trained Q table

	1	2	3	4	5
Left	N/A	-9.33	1.33	2.67	1.33
Right	1.33	2.67	5.33	10.67	N/A

Model-free Reinforcement Learning

Q-learning algorithm example

- Eventually, after training
  - The Q table will have Q values that increase as the robot moves right towards the positive reward
  - The Q table will have Q values that decrease as the robot moves left towards the negative reward
  - The robot can select which movements to make based on the Q values of each possible action, given a state



Q table after first transition from state 4 to state 5

	1	2	3	4	5
Left	N/A	0	0	0	0
Right	0	0	0	10	N/A

Q table after first transition from state 5 to state 4

	1	2	3	4	5
Left	N/A	0	0	0	5
Right	0	0	0	10	N/A

Fully trained Q table

	1	2	3	4	5
Left	N/A	-9.33	1.33	2.67	1.33
Right	1.33	2.67	5.33	10.67	N/A

Model-based Reinforcement Learning

- Model-free RL
  - State transitions are deterministic
  - Each action causes a definite state transition
- Model-based RL
  - State transitions are probabilistic
  - Each action causes a transition to a state with a certain probability

Model-based Reinforcement Learning

Value iteration: a model-based RL algorithm

- Maintains a representation of
  - State values
  - State transitions
  - Rewards
- For each triplet (s_t, a, s_{t+1}), store
  - Transition probability $T(s_t, a_t, s_{t+1})$
  - Reward value $R(s_t, a_t, s_{t+1})$

Model-based Reinforcement Learning

Value iteration: a model-based RL algorithm

Learn the value of each state $V(s)$

$$V^{new}(s_{t+1}) = max_a(\sum_{s'} T(s_t, a, s')[R(s_t, a, s') + \gamma V(s')])$$

Sum over all possible states

Range of possible states

The new value of a state is determined by selecting the action that maximizes the sum of the product of the transition probabilities and the sum of the reward for that transition and the discounted value of transition to that state

Model-based Reinforcement Learning

Value iteration: a model-based RL algorithm

- Learn the value of each state
  - Unclear from the text how this value is used to determine the action policy
  - Q-learning represents this policy explicitly as the Q value of each action for all states $Q(s_t, a_t)$
  - How does value iteration represent an action policy …
    as the V value of each action for all possible state transitions $V(s_t, a_t, s_t)$???
  - Need to investigate further

Prediction

- In model-based RL, state transition probabilities allows an agent to predict future states
- Prediction allows agents to minimize cost and maximize gains over long periods of time
- However, predictions are inherently uncertain
- We need to be able to quantify the degree of uncertainty and use this when selecting actions
- Entropy, H, is a measure of uncertainty

$$H = -\sum_{i} p_i log_2(p_i)$$

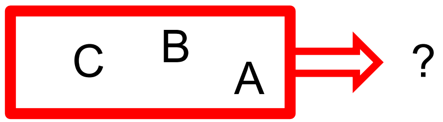
Aside: Information Theory

Measures of uncertainty

- Information and uncertainty are technical terms that describe any process that selects one or more objects from a set of objects

- Suppose we have a device that can produce 3 symbols, A, B, or C

  - Wait for the next symbol … uncertain about which symbol will be produced



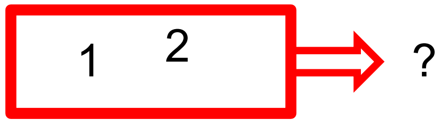
  - Once a symbol appears, and we see it, our uncertainty decreases … we have received some information

- Information is a decrease in uncertainty

Information Theory

Measures of uncertainty

- How should information be measured?
  - For the three-symbol device: “uncertainty of 3 symbols”?
- Consider a second device



  - “uncertainty of 2 symbols”?
- What happens when we combine them as one device:
  - Six symbols: A1, A2, B1, B2, C1, C2 ... uncertainty of 6 symbols
  - Not good: we would like our measure of information to be additive

Information Theory

Measures of uncertainty

- We can do this by using logs
  - $\log(3) + \log(2) = \log(3 \times 2) = \log(6)$
  - The base of the log determines the units of uncertainty

    $\log_2$ units are bits (from 'binary')
    $\log_3$ units are trits (from 'trinary')
    $\log_e$ units are nats (from 'natural logarithm') … usually use $\ln(x)$ for $\log_e(x)$
    $\log_{10}$ units are Hartleys, after an early worker in the field

Information Theory

Measures of uncertainty

- If a device produces one symbol, we are uncertain by $\log_2(1) = 0$ bits ...
  because there is no uncertainty about what the device will do next

- If a device produces two symbols (with equal probability),
  we are uncertain by $\log_2(2) = 1$ bit

- Candidate formula for uncertainty is $\log_2(M)$,
  where M is the number of symbols

$$\begin{aligned}\log_2(M) &= -\log_2(M^{-1})\\ &= -\log_2(1/M)\\ &= -\log_2(P)\end{aligned}$$

P is the probability that any symbol appears

Information Theory

Measures of uncertainty

- Now take the probability of the symbols appearing into account

  Generalize for the probabilities of the M individual symbols, P_i

$$\sum_{i=1}^{M} P_i = 1$$

- The surprise we get when we see the i^{th} type of symbol is sometimes called "surprisal": the degree of uncertainty about an outcome i

$$u_i = -\log_2(P_i) \qquad = \log_2(1/P_i)$$

Information Theory

- Measures of uncertainty
  - Uncertainty is the average surprisal for an infinite string of symbols produced by the device
  - Let's find the average for a string of length N that has an alphabet of M symbols
    - Suppose the i^{th} kind of symbol appears N_i times, then

$$N = \sum_{i=1}^{M} N_i$$

Information Theory

Measures of uncertainty

The average surprisal for the N symbols is given by

$$\frac{\sum_{i=1}^{M} N_i u_i}{\sum_{i=1}^{M} N_i}$$

Equivalently:

$$\sum_{i=1}^{M} \frac{N_i}{N} u_i$$

This is based on the frequentist definition of probability $P(\mathrm{E}) = \lim_{n \to \infty} \frac{n_{\mathrm{E}}}{n}$

Substituting in the term for probability

$$H = \sum_{i=1}^{M} P_i u_i$$

This is Shannon's famous 1948 definition of uncertainty (in bits per symbol)

H is called Entropy

Hence:

$$H = -\sum_{i=1}^{M} P_i \log_2 P_i$$

Information Theory

- Suppose there are three coins
  - A fair coin with equal probability of landing heads or tails
  - A weighted coin that lands on heads 90% of the time
  - A weighted coin that lands on heads, 10% of the time
- The uncertainty, i.e., the entropy, associated with these three probability distributions is

Entropy in nats

$$H_1 = -(.5(ln(.5)) + .5(ln(.5))) = .693$$

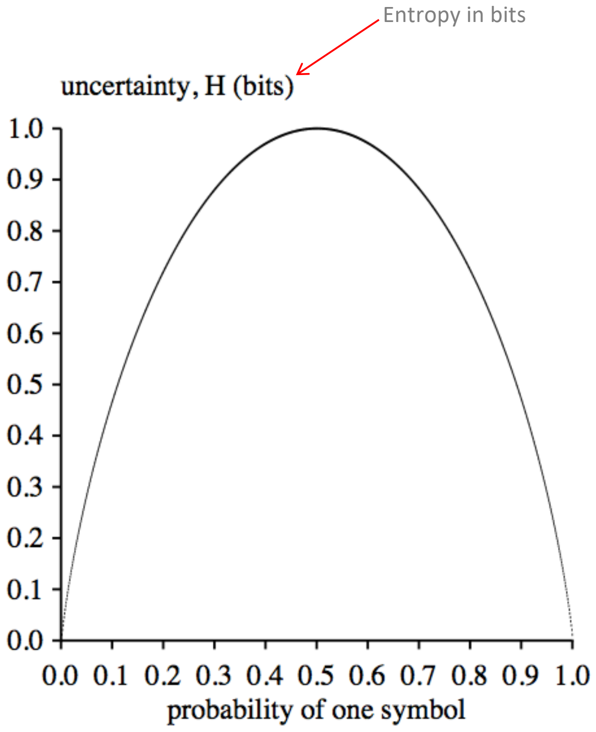
$$H_2 = -(.9(ln(.9)) + .1(ln(.1)) = .325$$

$$H_3 = -(.1(ln(.1)) + .9(ln(.9)) = .325$$

Three different probability distributions

Information Theory

Measures of uncertainty



H for the case of two symbols

Value-based Action Selection

- The softmax function is a way of selecting actions based on a probability distribution of the expected rewards

- If a collection of available actions has a quantity attached to each action, e.g., as a reward value, this can be converted into a probability distribution as follows

The temperature β: a parameter that regulates a tradeoff between exploration and exploitation:
small values lead to a flatter distribution and hence more exploration
large values lead to distributions with more peaks and hence more exploitation

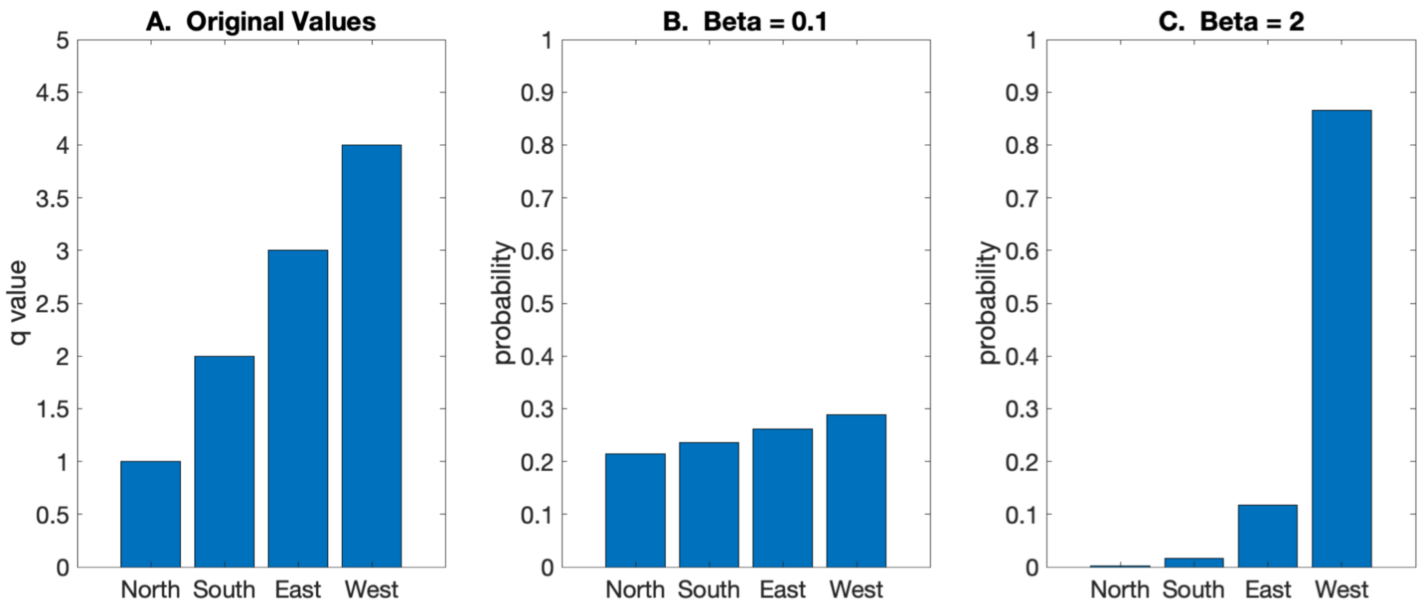
The quantity or value associated with action a

$$p_a = \frac{\exp(\beta q_a)}{\sum_{i=1}^{N} \exp(\beta q_i)}$$

Probability of taking and action a

sum over all actions

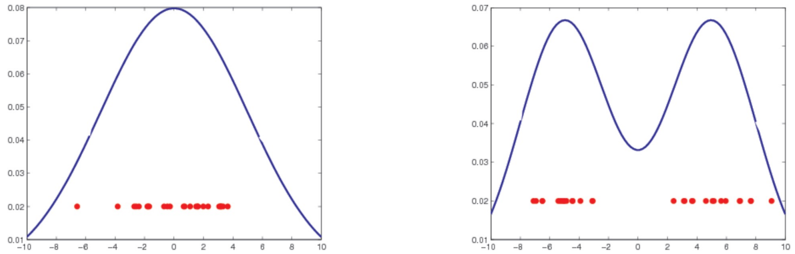
Value-based Action Selection



Value-based Action Selection

- Softmax coverts quantities of a distribution to probability distribution
- An action can then be chosen by sampling that distribution
- Question: how do you sample a distribution?

"When we say we sample (from) a distribution, we mean that we choose some discrete points, with likelihood defined by the distribution's probability density function"



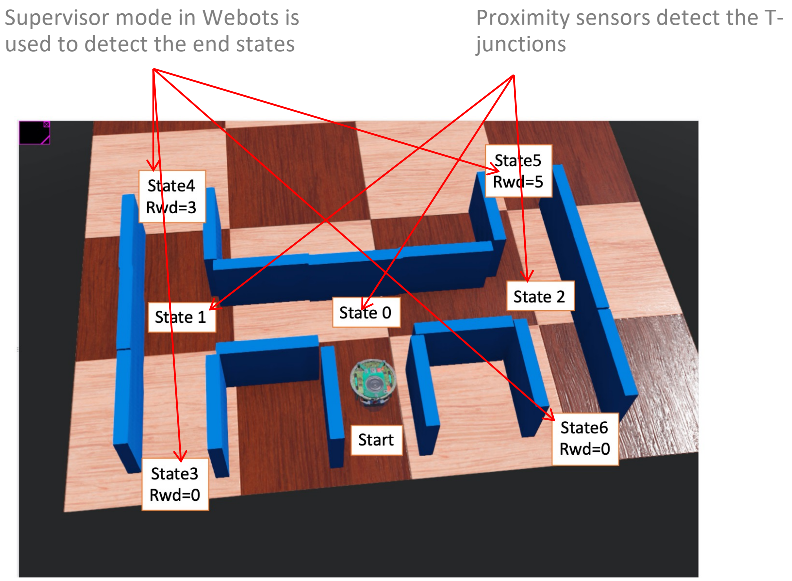
Samples (red dots) drawn from two different distributions

https://www.usna.edu/Users/cs/crabbe/SI475/current/particleFilter/particleFilter.pdf

Model-free Reinforcement Learning

Q-learning algorithm demonstration on Webots

- An e-Puck robot explores a double-T maze
  - States 0, 1, and 2 are decision points at T-junctions
    - The robot can turn left
    - The robot can turn right
  - States 3, 4, 5, and 6 are endpoints
    - The robot receives a reward of 0, 3, or 5, depending on the location
    - Q learning is used to learn the appropriate actions at each location / state



Model-free Reinforcement Learning

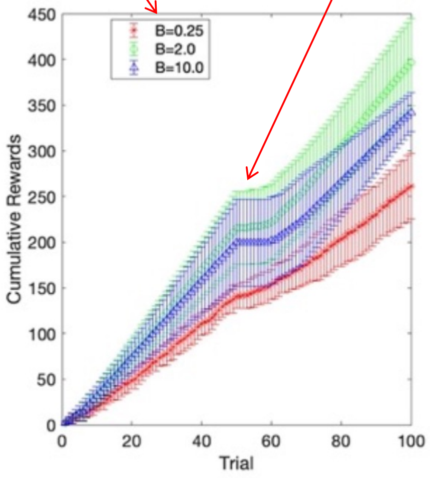
Q-learning algorithm demonstration on Webots

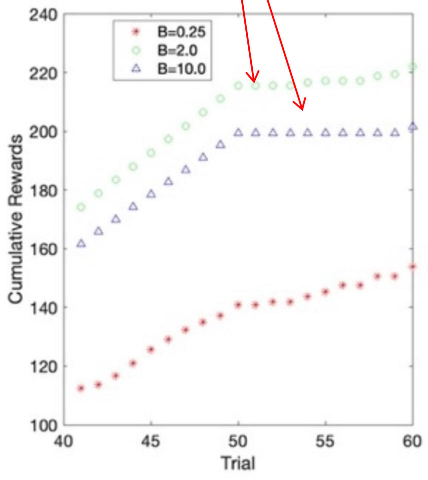
- An e-puck robot explores a double-T maze
  - States 0, 1, and 2 are decision points at T-junctions
    - The robot can turn left
    - The robot can turn right
  - States 3, 4, 5, and 6 are endpoints
    - The robot receives a reward of 0, 3, or 5, depending on the location
    - Q learning is used to learn the appropriate actions at each location / state

Higher value of β result in more reward before the reward locations change but diminished ability to adapt afterwards

Softmax temperature β

The reward locations are changed for trials 51 - 100





Mean cumulative reward in the ten trials leading up to and the ten trials following the change in the location of the rewards

Reading

Hwu, T. and Krichmar, J. (2022). Neurorobotics: Connecting the Brain, Body and Environment, MIT Press.

Chapter 4, Sections 4.1 - 4.5, pp. 63 - 74.

For a refresher on discrete probability and information theory, refer to the following lecture notes.

http://vernon.eu/STI/Lecture%2006%20-%20Discrete%20Probability.pdf