AI in Logistics

AUTUMN 2021

Christina Imdahl

IE & IS, OPAC

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Schedule of Second Part

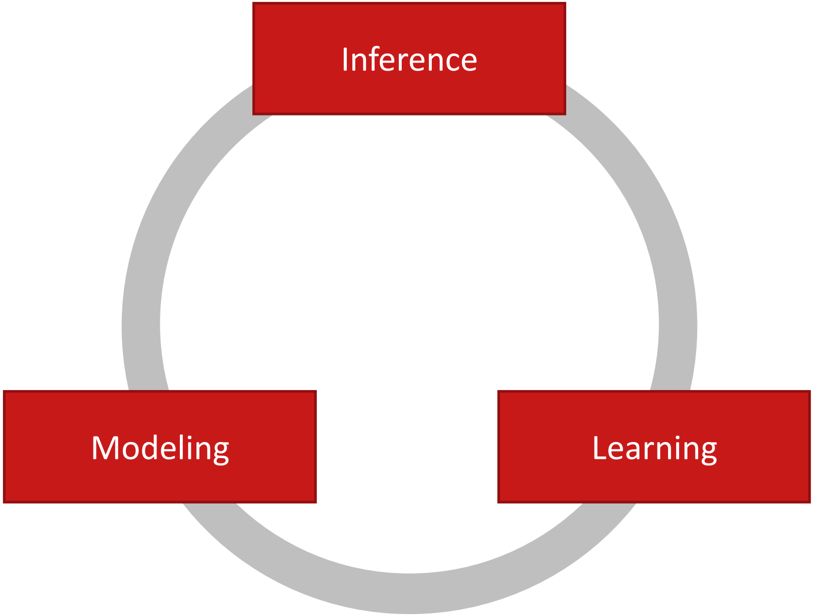
Monday	Tuesday	Wednesday	Thursday	Friday
Oct 4	Oct 5	Oct 6	Oct 7	Oct 8
	Introduction to RL			Case Study Descriptives and Orientation
Oct 11	Oct 12	Oct 13	Oct 14	Oct 15
	Reinforcement Learning – Key Concept			*(Homework: Implement RL)* *Momentum*
Oct 18	Oct 19	Oct 20	Oct 21	Oct 22
	Inventory Management - Heuristics	No Availability	No Availability	No Availability *Case Study Benchmark*
Oct 25	Oct 26	Oct 27	Oct 28	Oct 29
	Wrap-up / Case Study			Case Study
Nov 1	Nov 2	Nov 3	Nov 4	Nov 5
	Submission Case			Case Presentation

Objectives of Today

- Differentiate different ML techniques
- Learn about the basics of Reinforcement Learning
- Understand the interaction between model formulation and learning
- INTUITION on reinforcement learning

Agenda

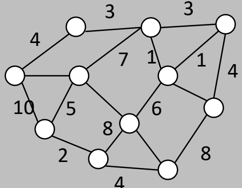
Paradigms



Modelling

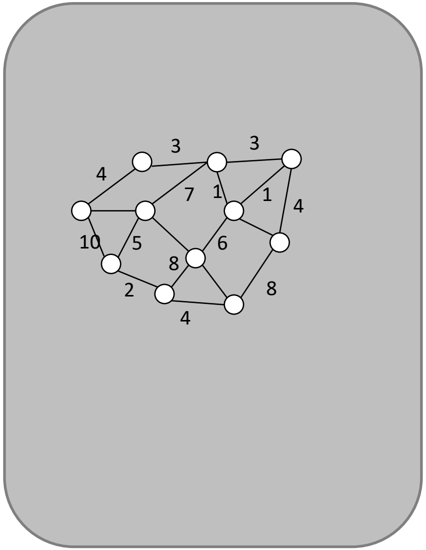
Real World Problem

Mathematical Model

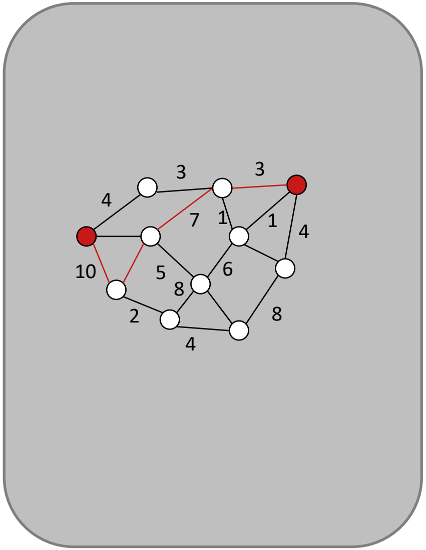


Inference

Full Model

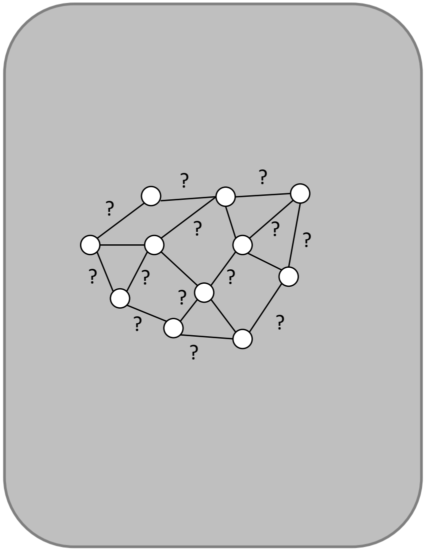


Inference

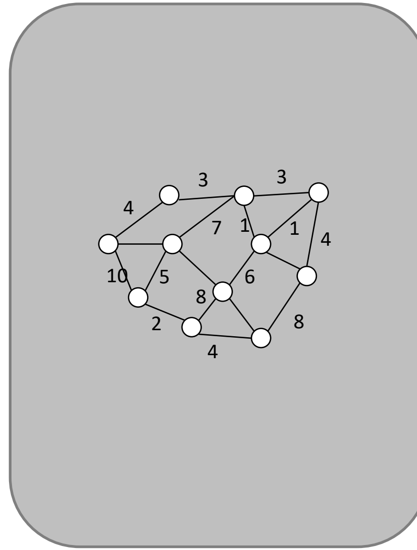


Learning

Model



Learned Model



Overview Machine Learning

Supervised Learning

Predict a value
Input: Labeled Training Data
Output: Prediction Model
Type: Regression/Classification

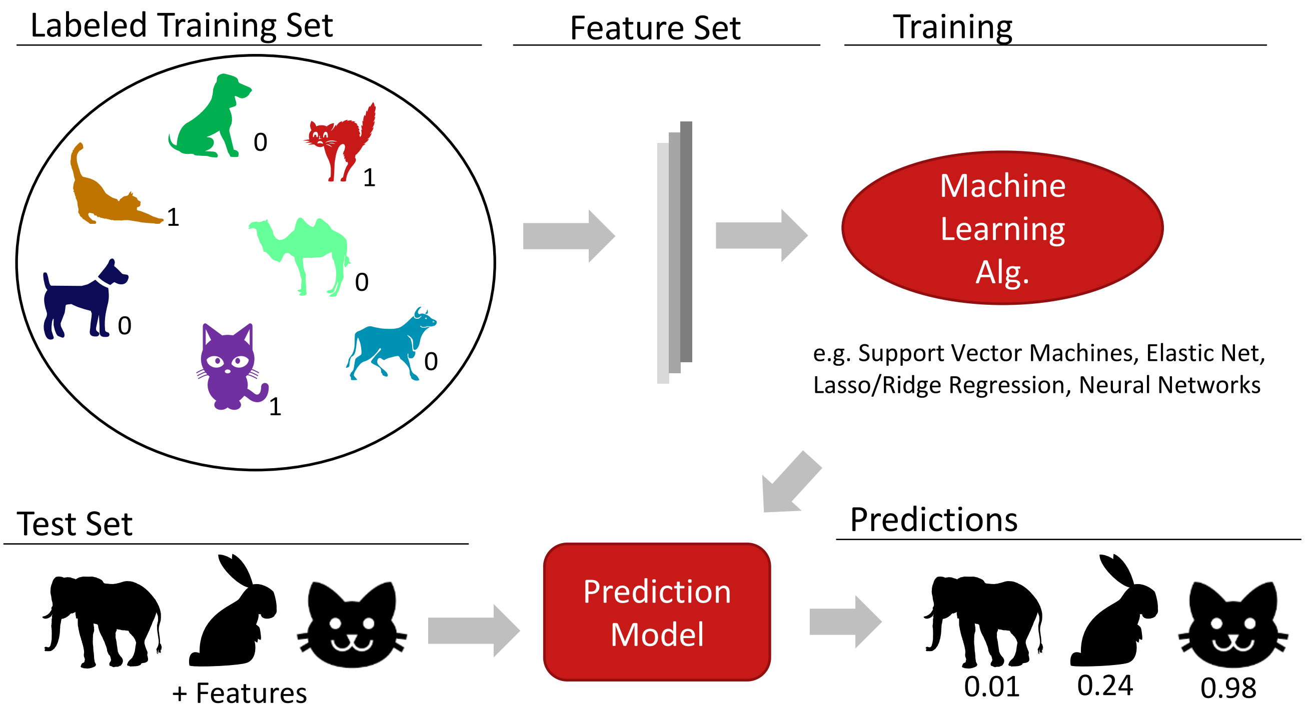
Unsupervised Learning

Identify patterns
Input: Unlabeled Training Data
Output: Classes/Associations
Type: Clustering/Associations

Reinforcement Learning

Find policy to optimize rewards
Input: No Predefined Data
Output: Policy
Type: Reward-based

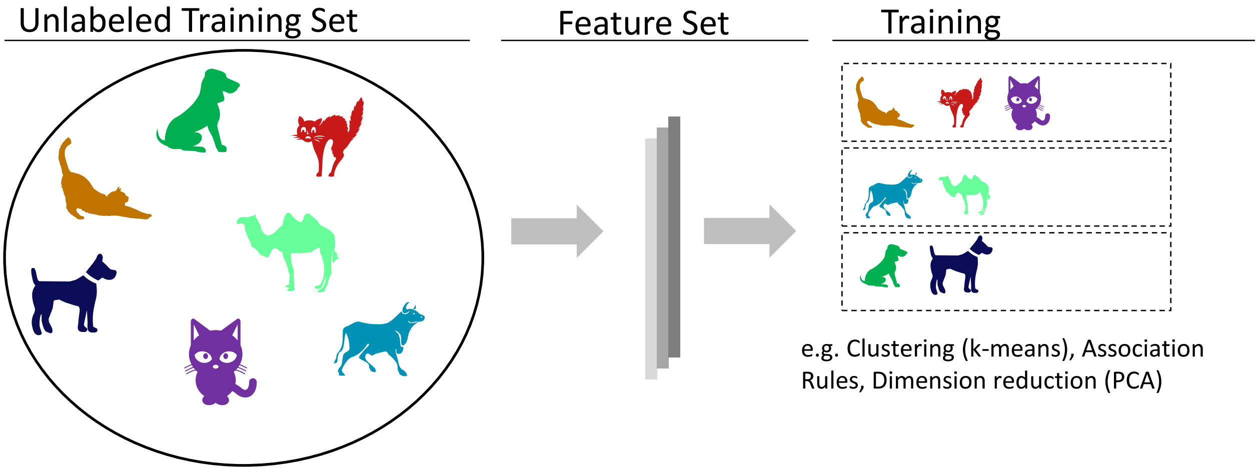
Supervised Learning



OM Applications

- Predicting Demand & Sales
- Predicting Machine Failures
- Predicting Warehouse Operations
- Predicting Decision-Maker Behavior
- Predicting Preferences

Unsupervised Learning

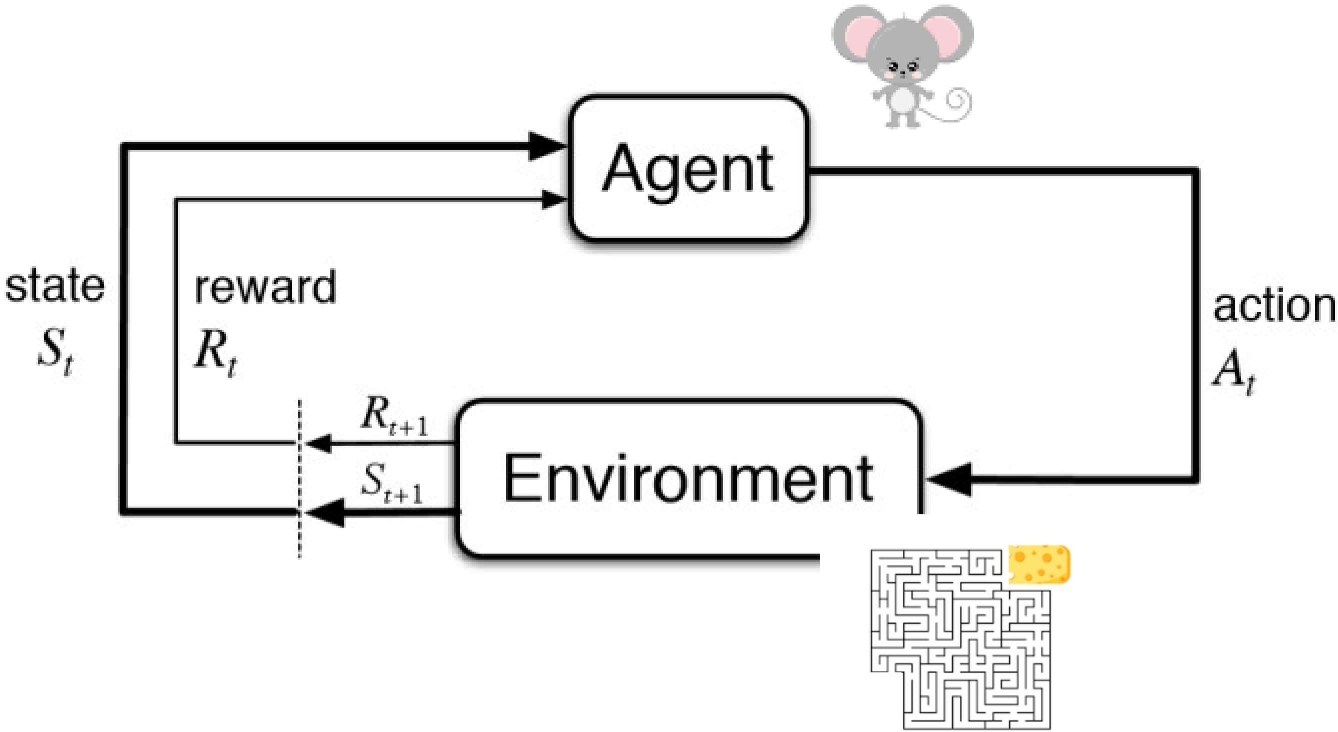


e.g. Clustering (k-means), Association Rules, Dimension reduction (PCA)

OM Applications

- Market Segementation/Targeted Advertisement
- Fraud Detection
- Basket Analysis
- Recommender Engines (Amazon „Other People Bought“, Netflix Movie Recommendation)

Reinforcement Learning



Characteristics of RL

- There is no supervisor, only reward
- Feedback is delayed
- Time matters (sequentiell decision making)
- Actions influence the data the agent receives

RL breaktrough

TU/e

„I thought AlphaGo was based on probability calculation and that it was merely a machine. But when I saw this move, I changed my mind. Surely, AlphaGo is creative.“

-- Lee Sedol

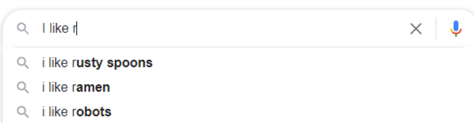
OM Applications

Retail

- Dynamic Pricing
- Anticipatory Shipping
- Item Descriptions
- Fraud Detection

Robotics

- Picking in Warehouses
- Routing Robots
- Manufacturing

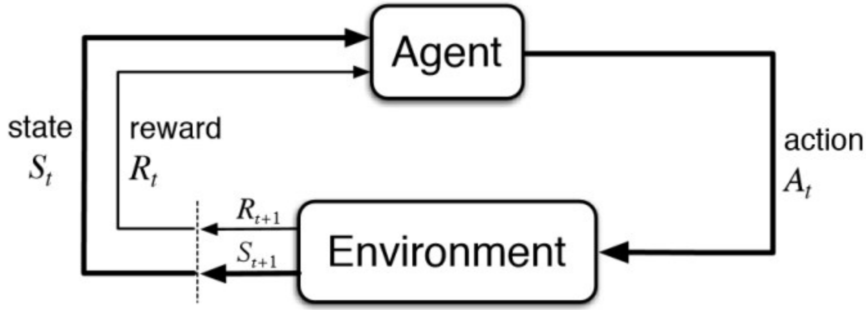


Speech Analysis

- Sentiment Analysis
- Chat robots
- Smart Home Tools

Agenda

Again: Markov Decision Processes



Agent observes state in t:	S_t
Agent takes action in t:	A_t
Agent receives reward R in t+1:	R_{t+1}
Agent moves to the next state:	S_{t+1}

RL Terminology

Policy: The agent‘s behavior given a state a

State-Value Function: how good is a state (s)?

Action-Value Function: how good is a state-action pair (s,a)?

Model: *agent*‘s representation of the environment.

Policy

- Determines the agent‘s behavior given a state
- The policy maps from state to actions
- Deterministc policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = P(A_t = a|S_t = s)$

State-Value function / Value function

- Value of a state under π
- Used to evaluate how good/bad a certain state is
- Used to select between actions
- E.g.:

$$v_\pi(s) = \mathbb{E}_\pi(\sum_{k=0}^{\infty} \gamma^k \underbrace{R_{t+k+1}(s_{t+k}, \pi(a|s_{t+k}), s'_{t+k})}_{\text{Disc. reward of following policy } \pi})$$

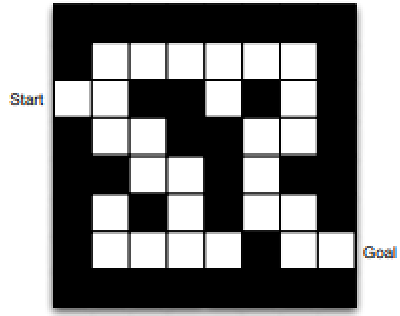
Action-Value function / Q-Function

- Value of an action under π
- Used to evaluate how good/bad a certain state-action pair is
- Used to select between actions
- E.g.:

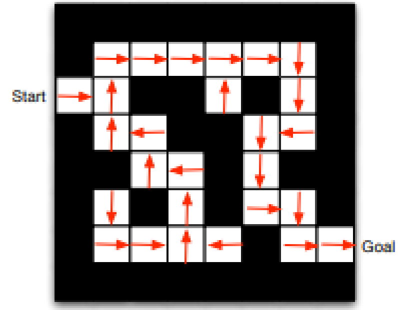
$$q_\pi(a,s) = \underbrace{\mathbb{E}_\pi(R_{t+1}(s_{t+k}, a, s'_{t+k})}_{\text{Immediate reward taking a}} + \underbrace{\sum_{k=1}^{\infty} \gamma^k R_{t+k+1}(s_{t+k}, \pi(a|s_{t+k}), s'_{t+k}))}_{\text{Disc. reward of following policy } \pi}$$

Example: MAZE

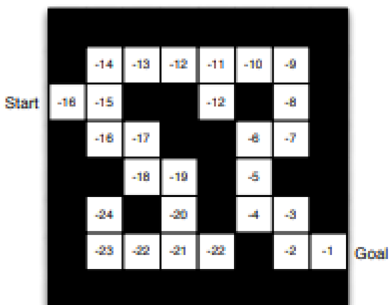
The game



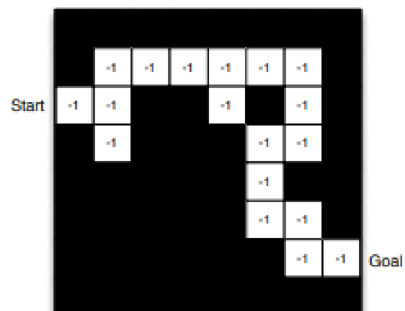
The policy



The value function



The model



Key goal of RL

Determine the policy that maximizes cumulative reward

- Determine the policy directly (policy-based methods)
  - Policy gradient methods, e.g. REINFORCE
- Determine the value function (value-based methods)
  - E.g. Q-Learning, SARSA

Dynamic Programming

- Two methods of dynamic programming can be used to solve MDPs
    - Value Iteration and Policy Iteration

- Let‘s revisit them…

Value Iteration

Update of values based on Bellman Eq.

$$v_{k+1}(s) = \max_a \sum_{s' \in S} P(s'|s,a) \, (\underbrace{R(s,a,s')}_{\text{Immediate Reward}} + \gamma \underbrace{v_{\pi_i,k}(s')}_{\text{Disc. Value of Sucessor State}})$$

1. Randomly initiate value function
2. Update values by above equation until convergence
3. Optimal policy is the action ending in the state with the maximum value

Policy Iteration

1. Randomly initiate policy
2. Policy evaluation: evaluate value function for current policy until convergence

$$v_{\pi_i,k+1}(s) = \sum_{s' \in S} P(s'|s,\pi_i(s))(R(s,\pi_i(s),s') + \gamma v_{\pi_i,k}(s'))$$

3. Policy improvement: improve policy by

$$\pi_{i+1}(s) = \mathrm{argmax}_a \sum_{s' \in S} P(s'|s,a)(R(s,\pi_i(s),s') + \gamma v_{\pi_i}(s')$$

Dynamic Programming Differences and Similarities

- Key methods from dynamic programing can be used to solve MDPs
  - Value iteration and policy iteration

- They assume full knowledge of the underlying MDP (transition probabilities, rewards)

- The *agent* often does not have these (we as programmers may have it)

- The agent shall be able of entering new situations that we don‘t know

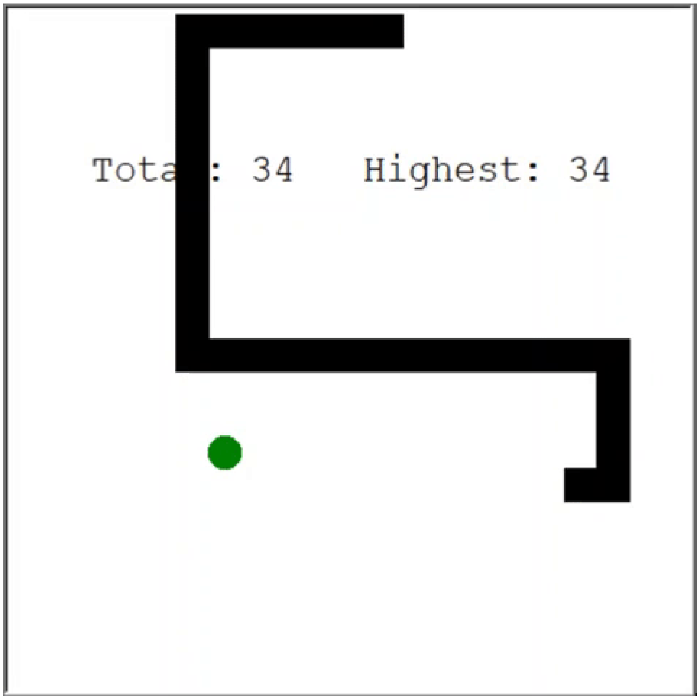
Next Lecture ….

How do we learn when transition probabilities and values are unkown to the agent.

Main idea: Try & Fail & Try….

Agenda

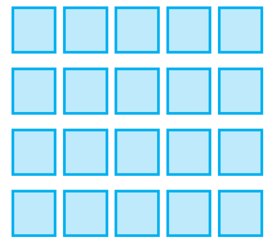
Snake



Discuss with your neighbour:

Actions

Rewards



State space

Snake

Rewards

-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100
-100											-100
-100			10								-100
-100						1					-100
-100					1		-1				-100
-100						-1					-100
-100											-100
-100											-100
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100

Actions

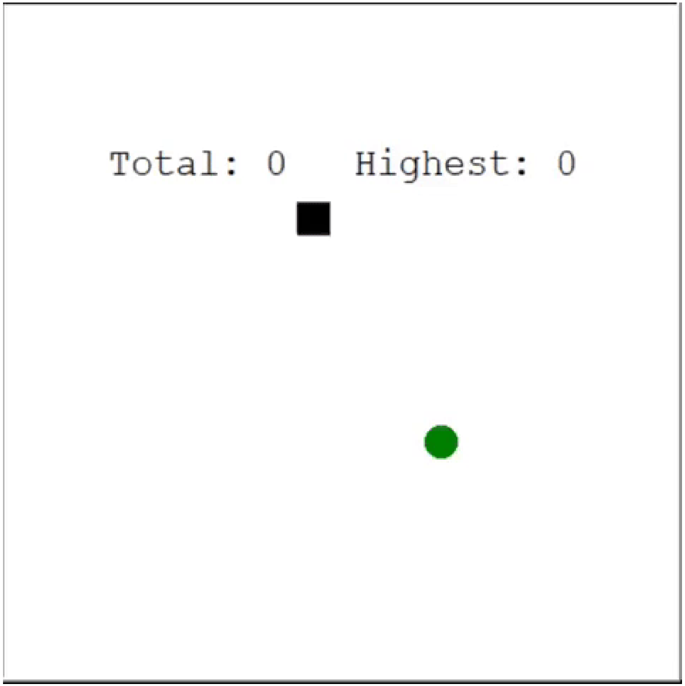
Snake moves up
Snake moves right
Snake moves down
Snake moves left

State

Apple is above the snake
Apple is on the right of the snake
Apple is below the snake
Apple is on the left of the snake
Obstacle directly above the snake
Obstacle directly on the right
Obstacle directly below the snake
Obstacle directly on the left
Snake direction == up
Snake direction == right
Snake direction == down
Snake direction == left

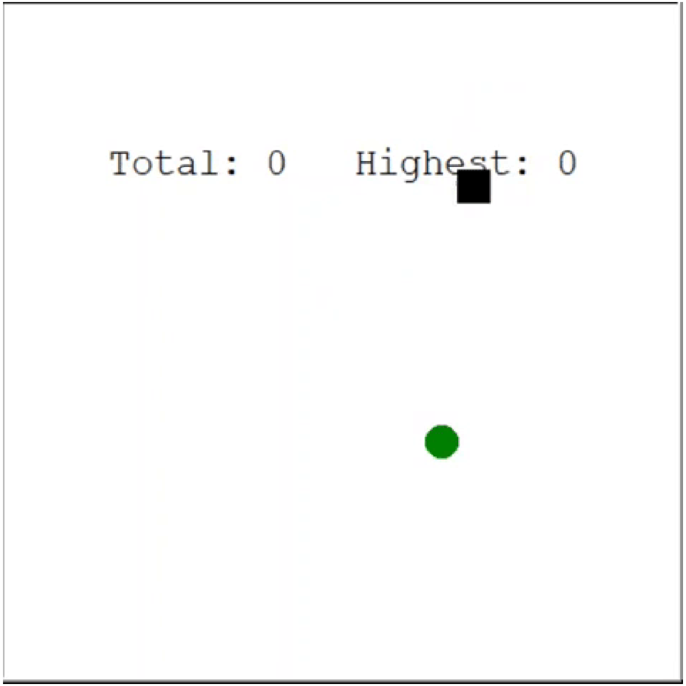
Snake - Learning

Game 1-4



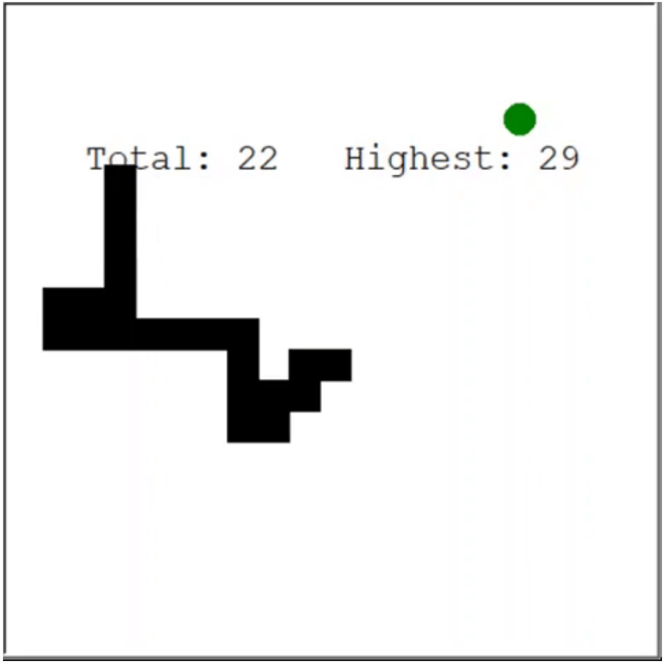
Snake - Learning

Game 7 – First Apple



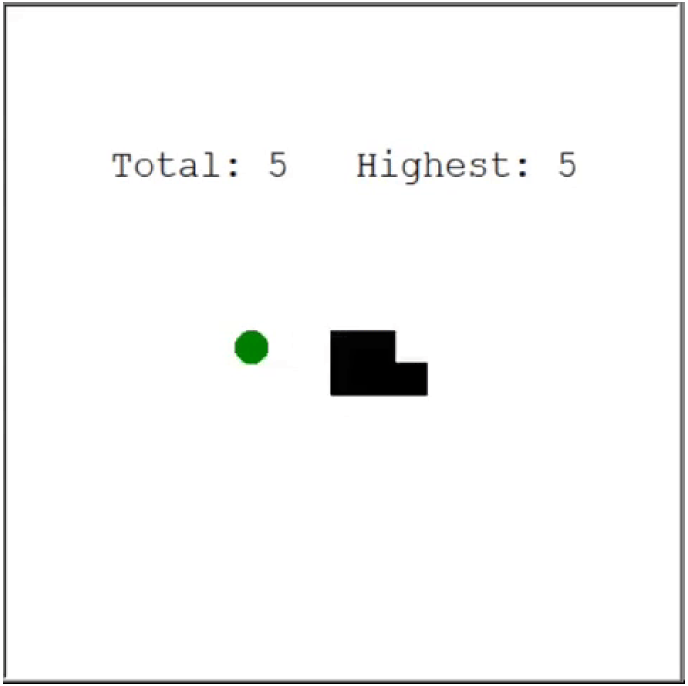
Snake - Learning

Game 13



Snake - Learning

Game 30



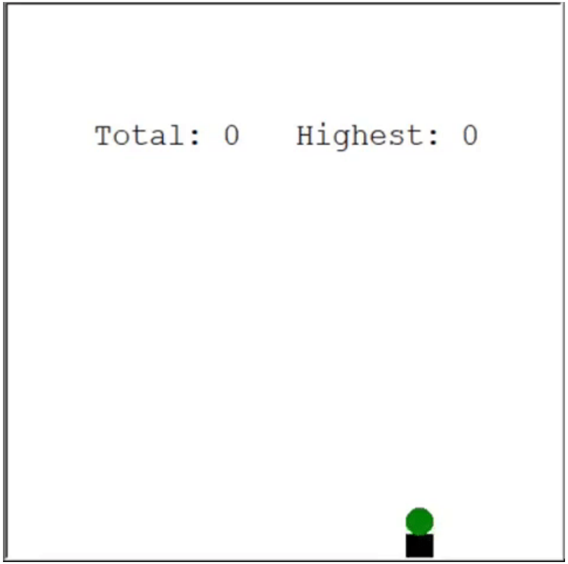
Agenda

Reward 1 – MOVE!

Rewards

-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100
-100											-100
-100			10								-100
-100						1					-100
-100					1		1				-100
-100						1					-100
-100											-100
-100											-100
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100

Result

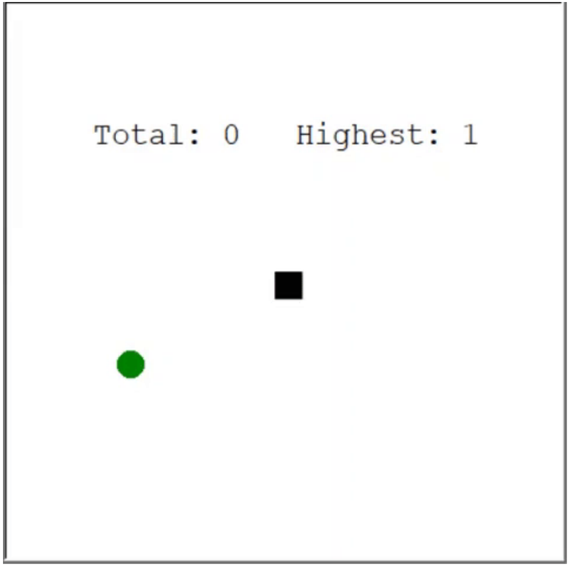


Reward 2 – RUN!

Rewards

-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100
-100											-100
-100			10								-100
-100						-1					-100
-100					-1		-1				-100
-100						-1					-100
-100											-100
-100											-100
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100

Result



Agenda

Different State Space Definitions

1 State (Obstactles = Body/Wall)

Apple is above the snake
Apple is on the right of the snake
Apple is below the snake
Apple is on the left of the snake
Obstacle directly above the snake
Obstacle directly on the right
Obstacle directly below the snake
Obstacle directly on the left
Snake direction == up
Snake direction == right
Snake direction == down
Snake direction == left

2 State - only walls (Obstacles = Wall)

Same as 1 but body location not included

3 State – coordinates

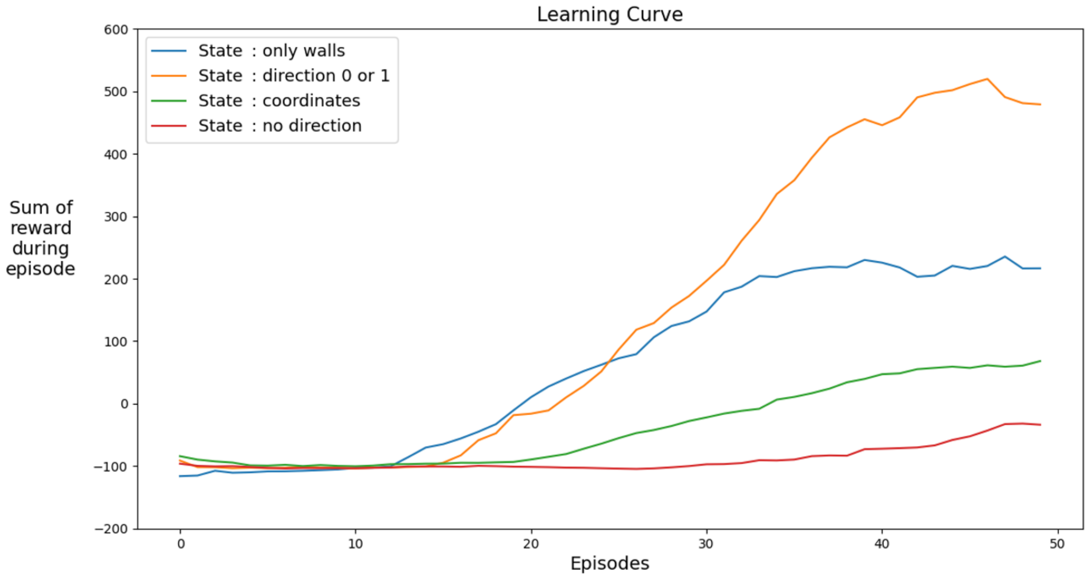
(x,y) Apple
(x,y) Snake

Obstacle directly above the snake
Obstacle directly on the right
Obstacle directly below the snake
Obstacle directly on the left
Snake direction == up
Snake direction == right
Snake direction == down
Snake direction == left

4 State – no direction

Apple is above the snake
Apple is on the right of the snake
Apple is below the snake
Apple is on the left of the snake
Obstacle directly above the snake
Obstacle directly on the right
Obstacle directly below the snake
Obstacle directly on the left

Learning Curve



Agenda

Set-up

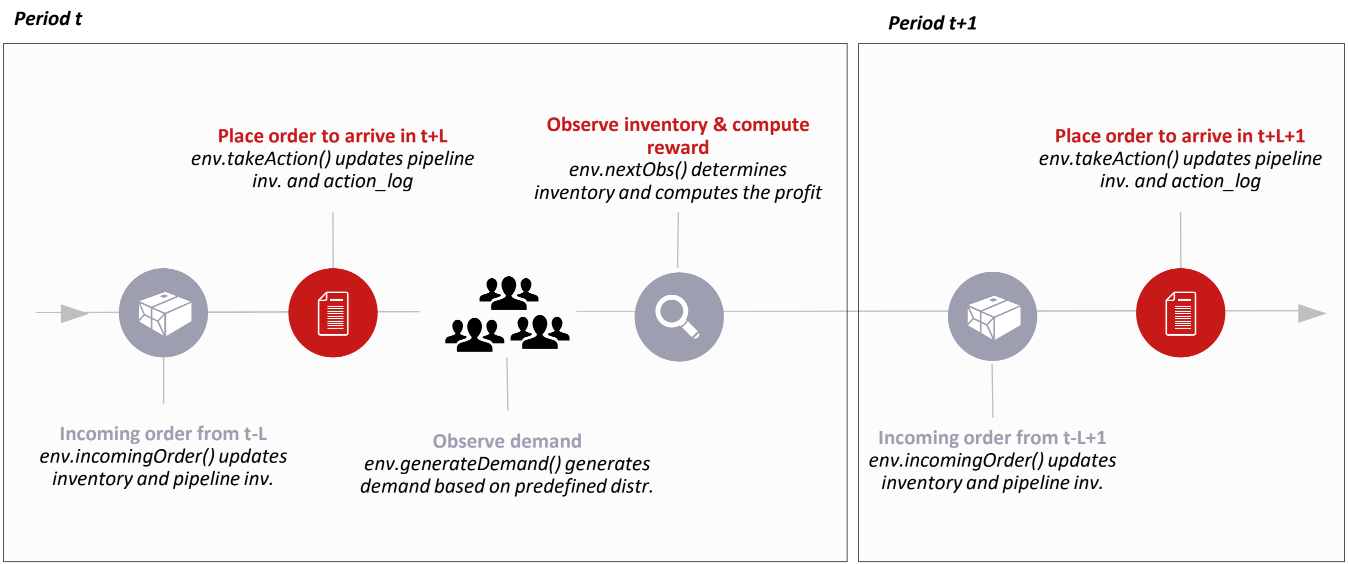
- Provider of bike spare parts (brakes, tires …)

- You start managing the inventory of one item based on part demands

- There is a penalty if you have to much inventory
- There is a penalty if you have to little inventory

- Your objective is to maximize your sales taking into account the costs.

Sequence of events



Two systems to study

Backorder System

- Demand not served at the end of the period can be served the next period
- Optimal policy is a base-stock policy with $S = F^{-1}(\frac{h}{h+b})$
- Can you be equally good?
- How do different parameter/model choices affect your learning?

Lost Sales System

- Demand not served at the end of the period is lost
- Optimal policy is unknown, approximative policies exist
- What is a suitable benchmark?
- Can RL outperform your benchmark?

Acknowledgements

Snake: https://towardsdatascience.com/snake-played-by-a-deep-reinforcement-learning-agent-53f2c4331d36

Snake Code: https://github.com/henniedeharder/snake

RL course by David Silver: https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ&index=1&t=74s

Back-Up Newsvendor

EXAMPLE OF TODAY
THE NEWSSTAND

- YOU are a running a small kiosk and need to decide on the optimal amount of papers at the **beginning of the day**
- You buy newspaper at a **costs of 2€**
- You sell newspaper at a **price of 10€**
- At the end of a day all newspapers are shredded
- You expect 30 customers (with a standard deviation of 10, normally distributed).

NEWSVENDOR MISMATCH COSTS

How much do I order?
Q = 30

Costs: 30 * 2€ = 60€

D<Q
P(D<Q) =F(Q)

Demand = 10

Revenue: 10 * 10€ = 100€
Profit: 100€ - 60€ = 40€

Overstock quantity: 30-10 = 20
Costs of overstocking c_o: 20 * 2€ = 40€

D≥Q
P(D≥Q) =1-F(Q)

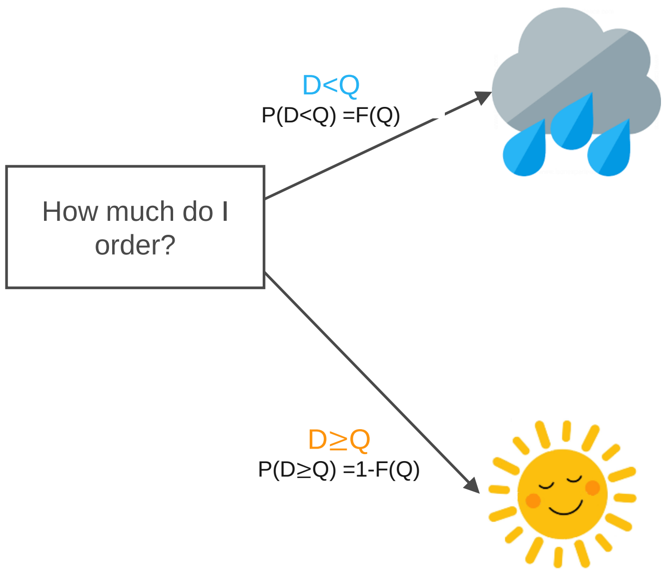
Demand = 50

Revenue: 30 * 10€ = 300€
Profit: 300€ - 60€ = 240€

Understock quantity: 50-30 = 20
Costs of understocking c_u: 20 * 8€ = 160€

$$C(Q;D) = c_o \underbrace{\max(0, Q-D)}_{\text{Overstock quantity}} + c_u \underbrace{\max(0, D-Q)}_{\text{Understock quantity}}$$

NEWSVENDOR MISMATCH COSTS



Overstocking

We need to pay *c* (2€) for every newspaper bought, but not sold.

$$c_o = c$$

Marginal expected costs of one additional unit:

$$c_o \cdot F(Q)$$

Understocking

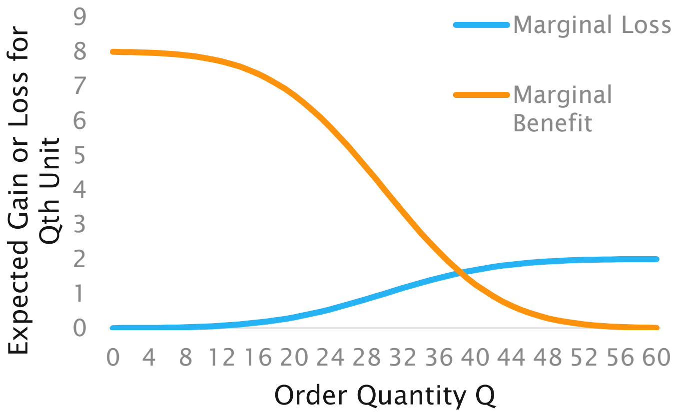
We are losing the margin *p-c (10€-2€)* for every newspaper that we did not sell, because we had too little.

$$c_u = p - c$$

Marginal expected costs of one additional unit:

$$c_u \cdot (1 - F(Q))$$

MARGINAL ANALYSIS: IS IT BENEFICIAL TO ORDER ONE MORE UNIT?



The optimal order quantity is the point where the expected marginal benefit through an order quantity equals the expected marginal loss through an additional order quantity.

$$c_o \cdot F(Q) = c_u \cdot (1 - F(Q))$$

NEWSVENDOR MISMATCH COSTS
OPTIMAL ORDER QUANTITY

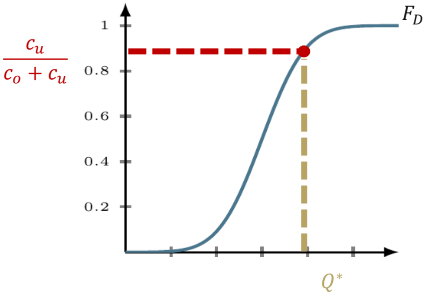
$$C(Q;D) = c_o \underbrace{\max(0, Q - D)}_{\text{Overstock quantity}} + c_u \underbrace{\max(0, D - Q)}_{\text{Understock quantity}}$$

Minimizing the expected mismatch costs via optimization yields a simple formula for the optimal order quantity:

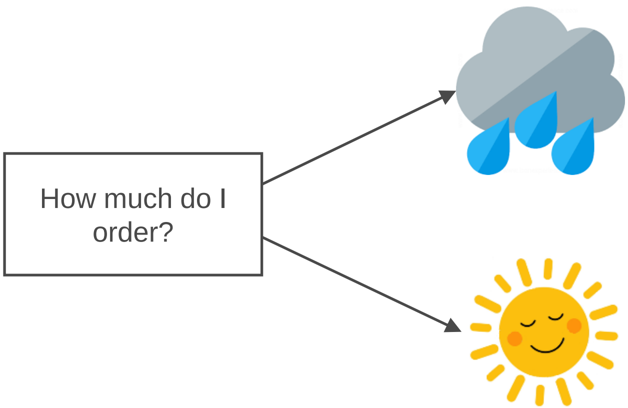
$$F(Q^*) = \frac{c_u}{c_o + c_u}$$

Critical Ratio

= Service level



NEWSVENDOR MISMATCH COSTS



Overstocking

We need to pay *c* (2€) for every newspaper bought, but not sold.

$$c_o = c$$

Understocking

We are losing the margin *p-c (10€-2€)* for every newspaper that we did not sell, because we had too little.

$$c_u = p - c$$

$$F(Q^*) = \frac{c_u}{c_u + c_o} = \frac{p-c}{p-c+c} = \frac{p-c}{p}$$

$$= \frac{10-2}{10} = 80\%$$