AI in Logistics

AUTUMN 2021

Christina Imdahl

IE & IS, OPAC

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Schedule of Second Part

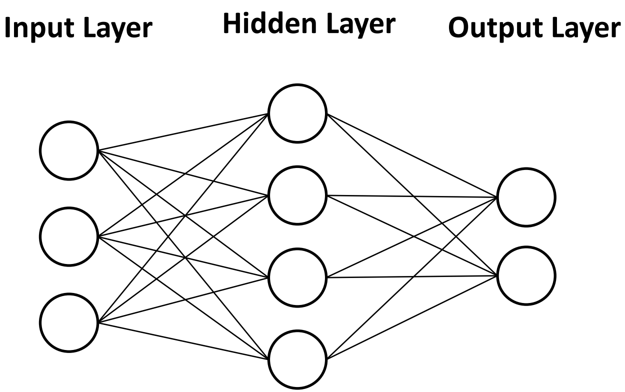
Monday	Tuesday	Wednesday	Thursday	Friday
Oct 4	Oct 5	Oct 6	Oct 7	Oct 8
	Introduction to ML			Case Study Descriptives and Orientation
Oct 11	Oct 12	Oct 13	Oct 14	Oct 15
	Reinforcement Learning – Key Concept			*(Homework: Implement RL)* *Momentum*
Oct 18	Oct 19	Oct 20	Oct 21	Oct 22
	Inventory Management - Heuristics			Case Study Implement a Benchmark
Oct 25	Oct 26	Oct 27	Oct 28	Oct 29
	Wrap-up / Case Study			
Nov 1	Nov 2	Nov 3	Nov 4	Nov 5
Case Presentation				

Objectives of Today

- Understand how neural nets can be used for prediction/function approximation
- First glance to Deep Reinforcement Learning
- Understand the difference between backlog and lost-sales inventory problems

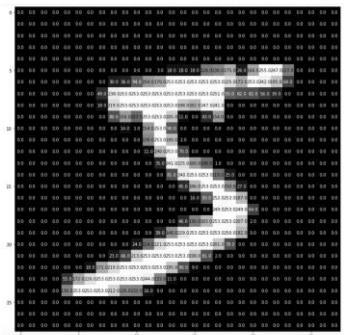
Agenda

Neural Nets



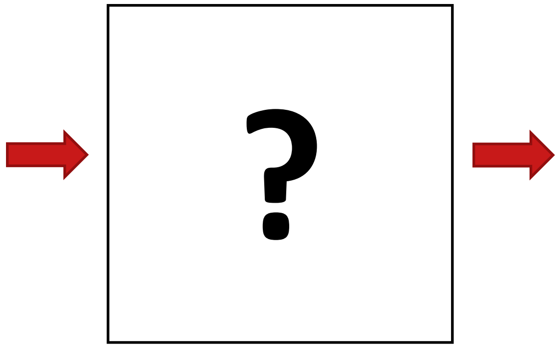
- Neural nets are a supervised machine learning method
- Neural nets consist of:
  - *Input Layers*: intitial data for the NN
  - *Hidden Layers*: intermediate layer between input & output layer
  - *Output Layer*: produces the results for given inputs
- Based on given data, the goal is to predict a certain outcome as best as possible.
- The neural network is trained by minimzing the loss on the training data.

Example: Recognizing Digits



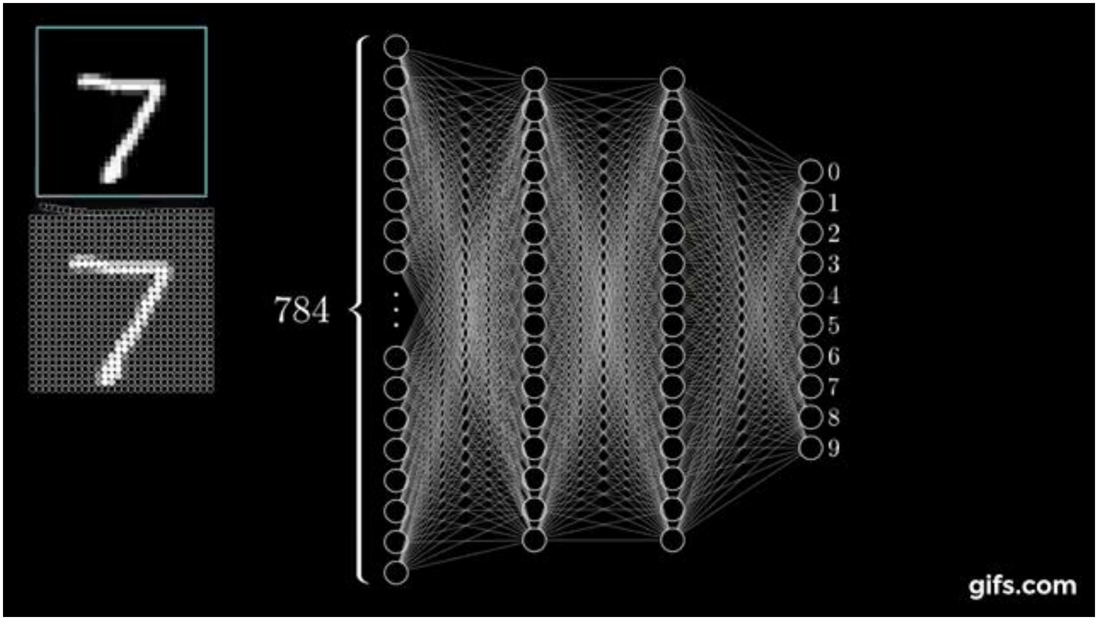
$28\text{x}28 = 784$ fields

Grey scale image $\in [0,1,\dots,255]$

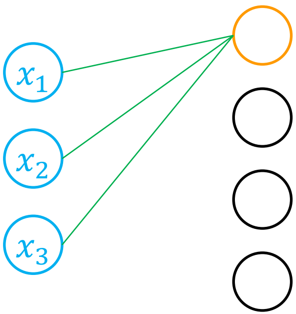


$0 : 0.01$
$1 : 0.03$
$2 : 0.01$
$3 : 0.80$
$4 : 0.04$
$5 : 0.31$
$6 : 0.01$
$7 : 0.08$
$8 : 0.06$
$9 : 0.02$

Neural nets propagates signals over the different layers



Neural Nets - Weights



$$\phi(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + \cdots + w_{1n}x_n + b_1)$$

$$\phi(w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + \cdots + w_{2n}x_n + b_2)$$

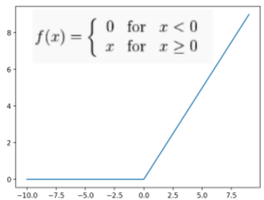
$$\vdots$$

$$\phi(w_{m1}x_1 + w_{m2}x_2 + w_{m3}x_3 + \cdots + w_{mn}x_n + b_m)$$

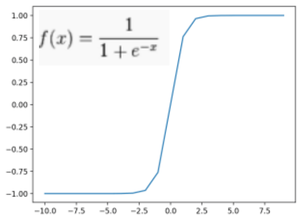
$$\phi\left(\begin{pmatrix} w_{11} & \cdots & w_{1n} \\ w_{21} & \cdots & w_{2n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}\right)$$

Activation functions

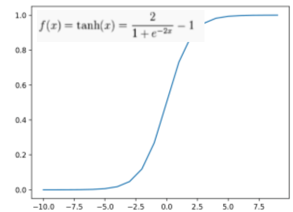
Hidden Layer
(non-linear, differentiable functions, same for all hidden layers)



ReLU

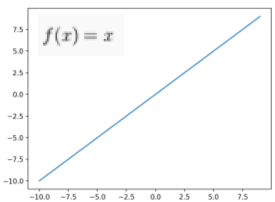


Sigmoid

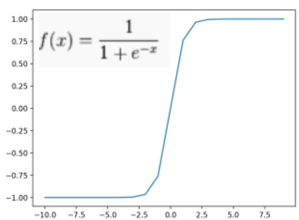


Tanh

Output Layer
(dependent on prediction/estimation task)



Identity



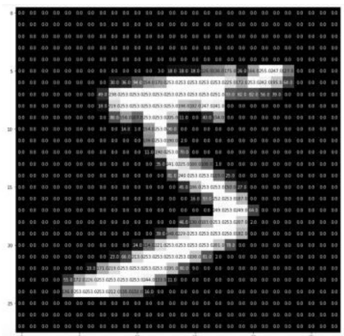
Sigmoid

$$f(x) = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

Softmax

How to estimate the weights?

Trained examples:



$$
\begin{aligned}
0 &: (0.01 - 0)^2 = 0.0001 \\
1 &: (0.03 - 0)^2 = 0.0009 \\
2 &: (0.01 - 0)^2 = 0.0001 \\
3 &: (0.80 - 1)^2 = 0.04 \\
4 &: (0.04 - 0)^2 = 0.0016 \\
5 &: (0.31 - 0)^2 = 0.0961 \\
6 &: (0.01 - 0)^2 = 0.0001 \\
7 &: (0.08 - 0)^2 = 0.0064 \\
8 &: (0.06 - 0)^2 = 0.0036 \\
9 &: (0.02 - 0)^2 = 0.0004
\end{aligned}
$$

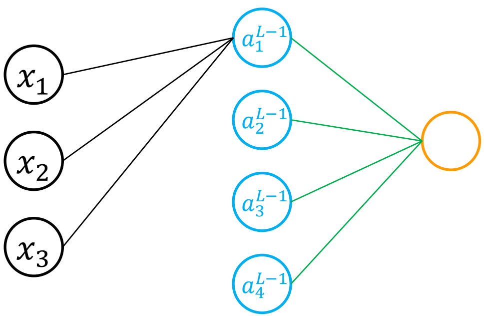
Training Objective : $C(w^1, \ldots, w^L, b^1, \ldots, b^L) = \min(\frac{1}{n}\sum_i (\hat{y} - y)^2)$

(also consider binomial loss, but MSE more relevant for later)

Weights are updated using back-propagation

Training Objective : $\min((\hat{y} - y)^2)$

$$a^L = \phi(w_{11}^L a_1^{L-1} + w_{12}^L a_2^{L-1} + w_{13}^L a_3^{L-1} + \dots + w_{1n}^L a_4^{L-1} + b_1)$$



$w^L \quad a^{L-1} \quad b^L$

$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \phi(z^L)$$

$$C_0 = (a^L - y)^2$$

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L} = a^{L-1} \cdot \phi'(z^L) \cdot 2(a^L - y)$$

Weights are updated using back-propagation

Training Objective : $\min((\hat{y} - y)^2)$

w^{L-1} $\quad a^{L-2}$ $\quad b^{L-1}$

z^{L-1}

w^{L} $\quad a^{L-1}$ $\quad b^{L}$

$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \phi(z^L)$$

$$C_0 = (\mathrm{a^L - y})^2$$

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L} = a^{L-1} \cdot \phi'(z^L) \cdot 2(a^L - y)$$

$$\frac{\partial C_0}{\partial b^L} = \frac{\partial z^L}{\partial b^L}\frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L} = 1 \cdot \phi'(z^L) \cdot 2(a^L - y)$$

$$\frac{\partial C_0}{\partial a^{L-1}} = \frac{\partial z^L}{\partial a^{L-1}}\frac{\partial a^L}{\partial z^L}\frac{\partial C_0}{\partial a^L} = w^L \cdot \phi'(z^L) \cdot 2(a^L - y)$$

Steepest gradient descent

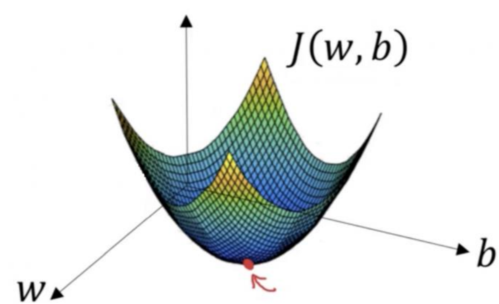
$$\nabla C(w^1, \dots, w^L, b^1, \dots, b^L) = \begin{pmatrix} \frac{\partial C}{\partial w^1} \\ \vdots \\ \frac{\partial C}{\partial w^L} \\ \frac{\partial C}{\partial b^1} \\ \vdots \\ \frac{\partial C}{\partial w^L} \end{pmatrix}$$

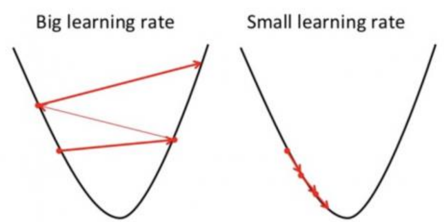
The gradient gives the direction of the steepest increase.

The negative gradient gives the direction of the steepest decline.

Steepest gradient descient:

1. Compute ∇C
2. Take step in - ∇C direction
   $(w_{new}, b_{new}) = (w_{old}, b_{old}) - \alpha \nabla C$
3. Repeat

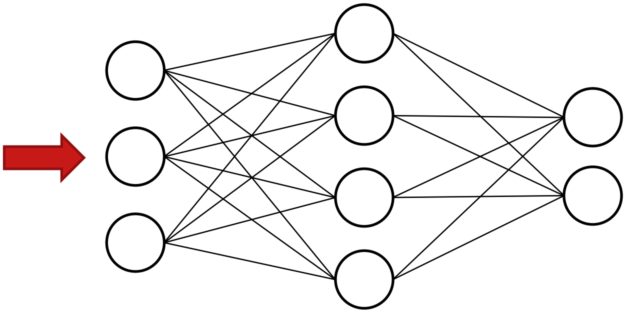




Agenda

DQN - Overview

State s



How to receive the labeled Q-values? (for supervised learning)

How to train the network stable?

Q-Learning and DQN

Q-Learning

$$Q(S_t, A_t) \leftarrow Q(s_t, a_t) + \alpha\big(\underbrace{R_{t+1} + \gamma \max_{a' \in A(s_t)} Q(s_{t+1}, a')}_{\text{Experience}} \underbrace{- Q(s_t, a_t)}_{\text{Expectation}}\big)$$

DQN

Target: $$y = R_{t+1} + \gamma \max_{a' \in A(s_t)} Q(s_{t+1}, a')$$

Estimate: $$\hat{y} = Q(s_t, a_t)$$

Memory replay

Replay memory

$< S_1, A_1, R_2, S_2 >$

$< S_2, A_2, R_3, S_3 >$

$< S_3, A_3, R_4, S_4 >$

$\vdots$

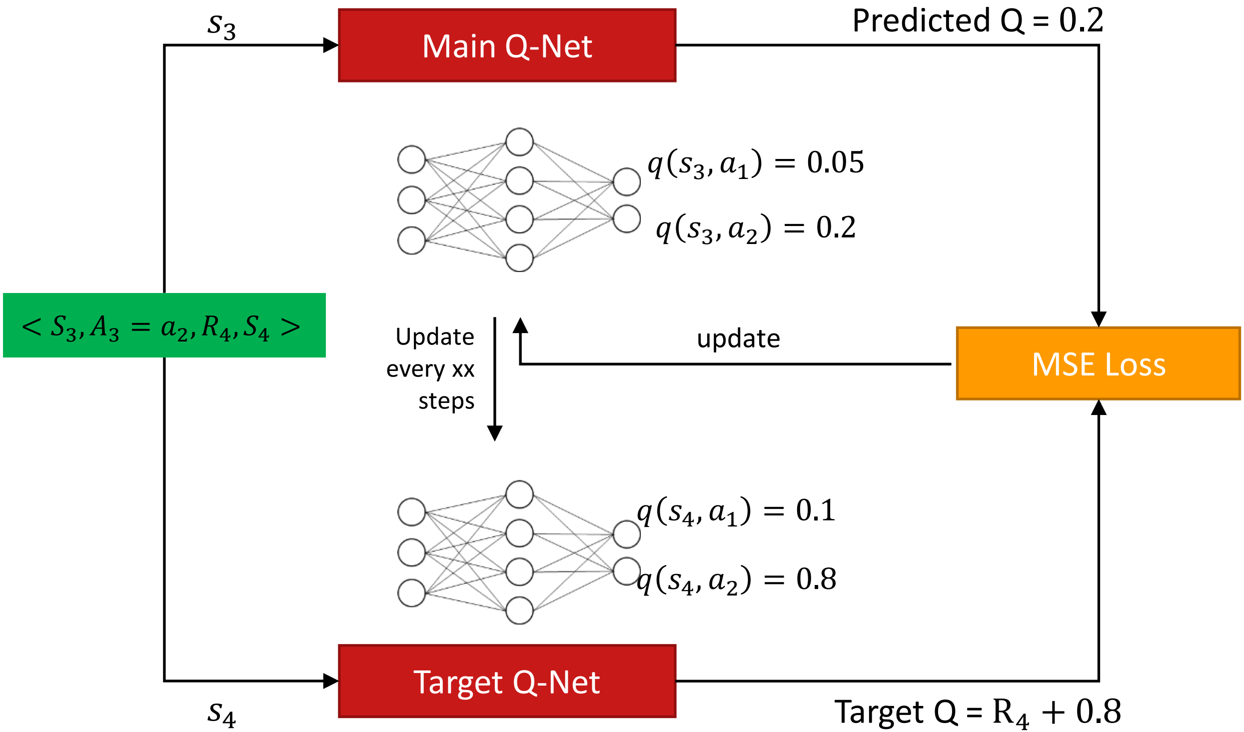
$< S_{N-1}, A_{N-1}, R_N, S_N >$

Select n samples

Minibatch

Optimise neurat net

Target Net



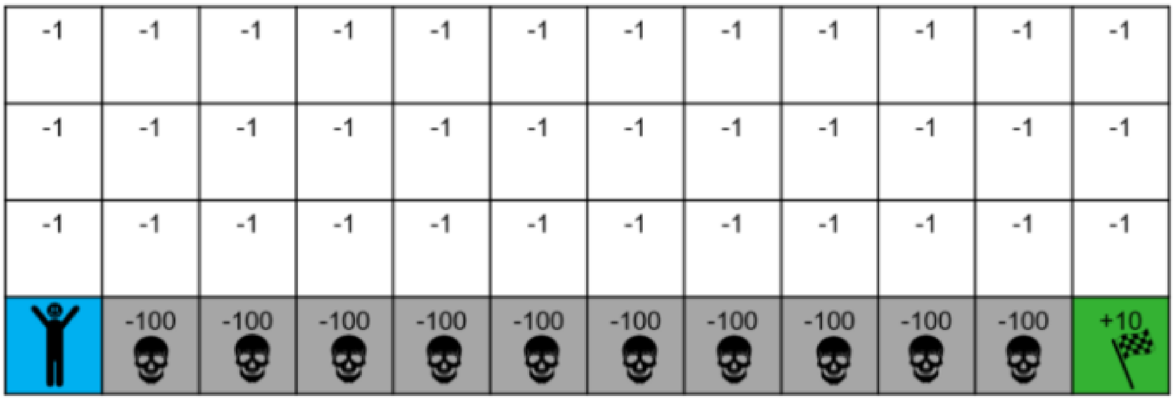
s_3
Main Q-Net
Predicted Q = 0.2
$q(s_3, a_1) = 0.05$
$q(s_3, a_2) = 0.2$
$< S_3, A_3 = a_2, R_4, S_4 >$
Update every xx steps
update
MSE Loss
$q(s_4, a_1) = 0.1$
$q(s_4, a_2) = 0.8$
s_4
Target Q-Net
Target Q = $R_4 + 0.8$

Hyperparameters

- Network architecture (layers, units per layer, activation)

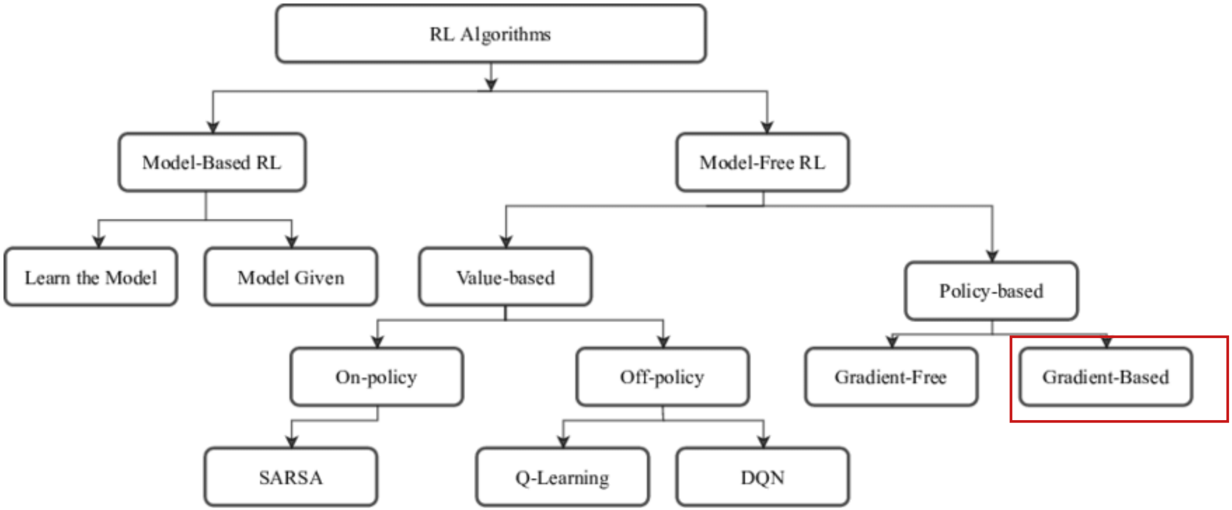
- Learning rate
- Memory size (max, min), batch size
- Loss function
- Target network update frequency
- Discount factor
- Exploration rate

Walking the Cliff

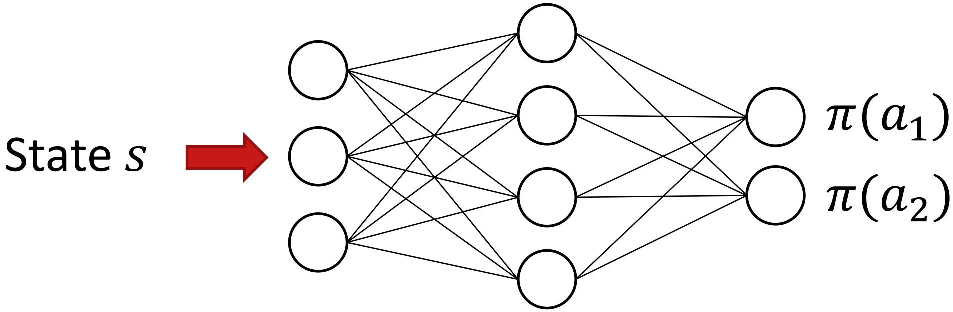


Agenda

Overview of ML Algorithms (incomplete)



REINFORCE (1)



REINFORCE (2)

Update the policy network in the direction of steepest incline:

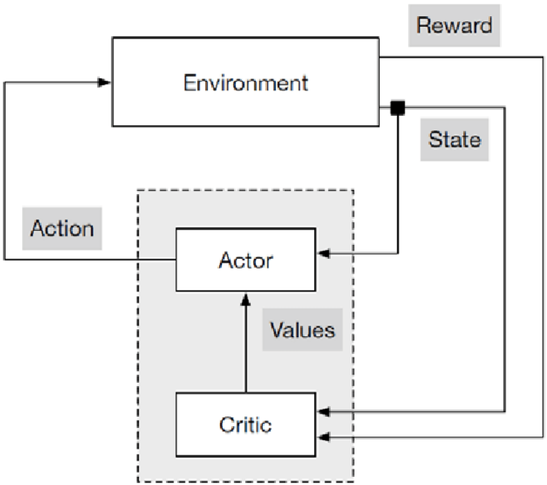
Objective: $\max J(\theta) = \mathbb{E}(\sum_{t=0}^{T-1} \gamma^t r_{t+1} | \pi_\theta)$

Update Rule: $\theta \leftarrow \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$

Gradient: $\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t | s_t)}_{\text{Make action more likely under the current policy}} (\underbrace{\sum_{t'=t+1}^{T} \gamma^{t'-t-1} r_{t'}}_{\text{Weight of trajectories by „how good they were“}})$

More: https://medium.com/@thechrisyoon/deriving-policy-gradients-and-implementing-reinforce-f887949bd63

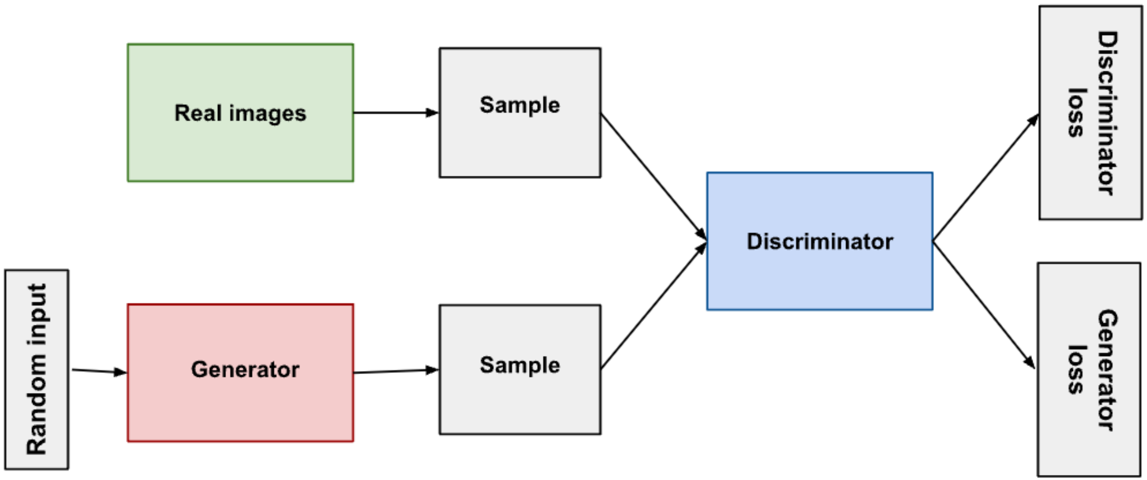
Actor-Critic Methods



- Include Value Function in Reinforce
  $Q(s_t, a_t) \sim \mathbb{E}(\sum_t r^t)$
- Reduces noisyness in estimation
- Critic: Estimates the value function (Q value or state value)
- Actor: updates policy distribution in the direction suggested by the Crtic

Agenda

Generative Adversarial Nets (GAN)



$$\min_G \max_D V(D, G)$$

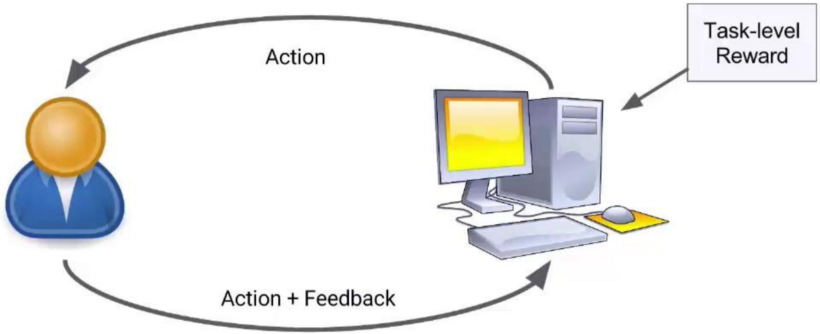
$$V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Imitation Learning

- Inspired by how humans learn
- Used in situations, when rewards are unkown
- Experts state-actions (s_0, a_0^*), (s_1, a_1^*) are used for supervised training
- Learn π_θ by minimizing the loss function $L(a^*, \pi_\theta(s))$

Interactive learning

Interactive Reinforcement Learning: User in the loop



Thomaz, A. L., Hoffman, G., and Breazeal, C. (2005). Real-time interactive reinforcement learning for robots. In AAAI 2005 workshop on human comprehensible machine learning.

Agenda

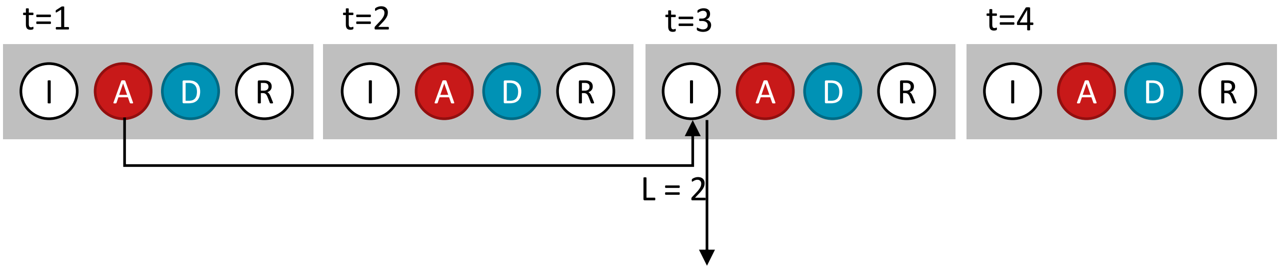
Multi-Period Inventory System with Backlog

- I_{net} On-hand inventory
- B Backorders
- I = $I_{net} - B$ Inventory level
- $T = (T_1, \dots, T_{L-1})$ Pipeline inventory

- $IP = I + \sum T$ Inventory position
- S Basestock level

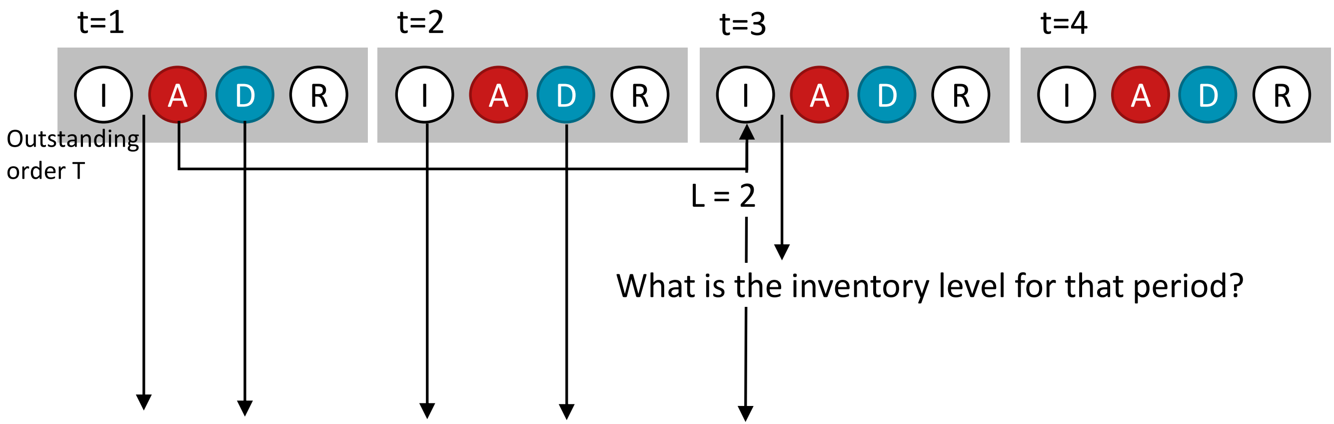
- D Demand
- D_L Demand over the lead time

Inventory Dynamics



What is the inventory level for that period?
What is the target value?

Inventory Dynamics



$$I_3 = I_1 - D_1 + T_1 - D_2 + Q_1 = I_1 + T_1 + Q_1 - D_1 - D_2$$

$$= IP_1 + Q_1 - D_L$$

$$= S - D_L$$

Solving for the optimal basestock level

$$C(S) = h \cdot \mathbb{E}(Left - over) \qquad\qquad + b \cdot \mathbb{E}(Backlog)$$
$$C(S) = h \cdot \int_0^S p(D_L = x) \cdot (S - x)dx + b \cdot \int_S^\infty p(D_L = x) \cdot (x - S)dx$$

Trick: Use

$$b \cdot \int_S^\infty p(D_L = x) \cdot (x - S)dx = b \cdot \int_0^\infty p(D_L = x) \cdot (x - S)dx - b \cdot \int_0^S p(D_L = x) \cdot (x - S)dx$$
$$\int_0^S p(D_L = x) \cdot (x - S)dx = -\int_0^S p(D_L = x) \cdot (S - x)dx$$

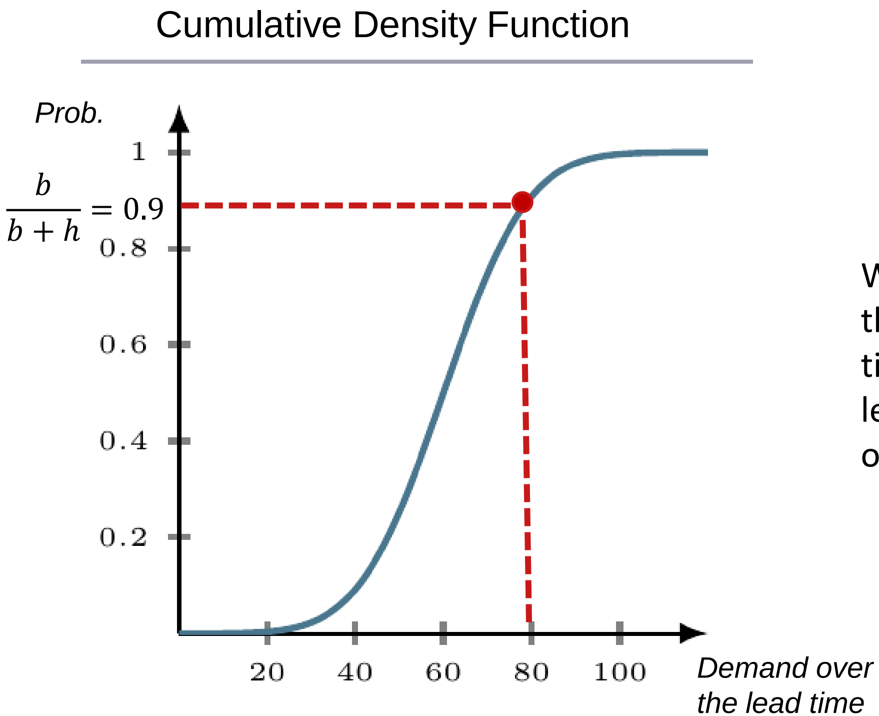
Integration by parts

Set derivative to zero (when no integrals are left)

You get

$$S = F_L^{-1}\left(\frac{b}{b + h}\right)$$

Base-Stock as Service Level

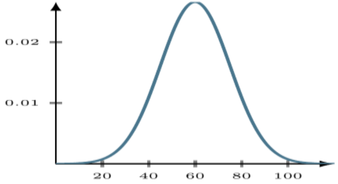


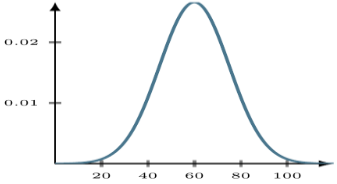
What is the probability that the demand over the lead time exceeds the base-stock level/inventory (probability of stocking out)?

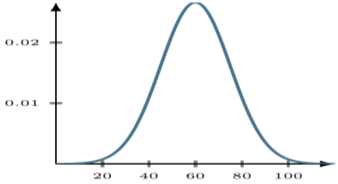
Cumulative demand over several periods

Assume the weekly demand is normally distributed with mean μ and standard deviation σ.

Then, the demand over L weeks is normally distributed with **mean $L\mu$** and **standard deviation $\sigma\sqrt{L}$.**

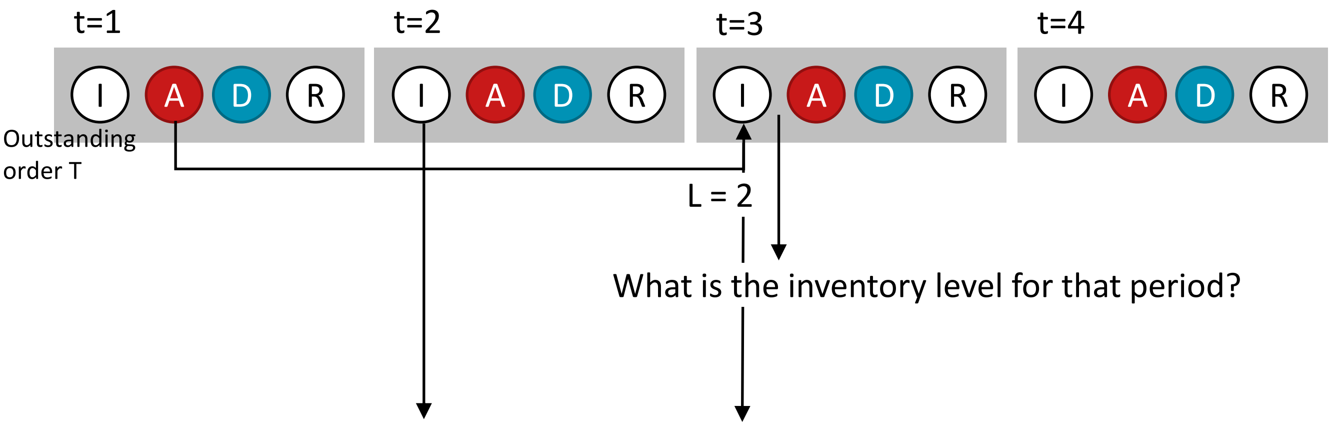






Agenda

Inventory Dynamics



$$I_2 = \max(I_1 - D_1, 0) + T_1$$

$$I_3 = \max(I_2 - D_2, 0) + Q$$

Benchmarking

- Relate to similar problems
- Optimize within a policy class
- Myopic policies
- Approximative policies
- Performance bounds
- Robust policies
- Clearvoyant approach

Myopic policy

The **myopic policy** selects the order quantity Q such that the immediate expected rewards are minimized (one-period costs). The period is typically the period in which the order arrives.

For the lost-sales systems:

$$\mathbb{P}(I_{t+L} - D_{t+L}) \leq \frac{c+h}{b+h}$$

„Clearvoyant“ approach

- With this approach the best possible outcome given the demand realization is evaluated
- „What should I have ordered, would I have known the demand?“
- The achieved solution is better than optimal
- The approach can be used, when no other suitable benchmarks are available.