Domain Transfer for Reinforcement Learning Agents

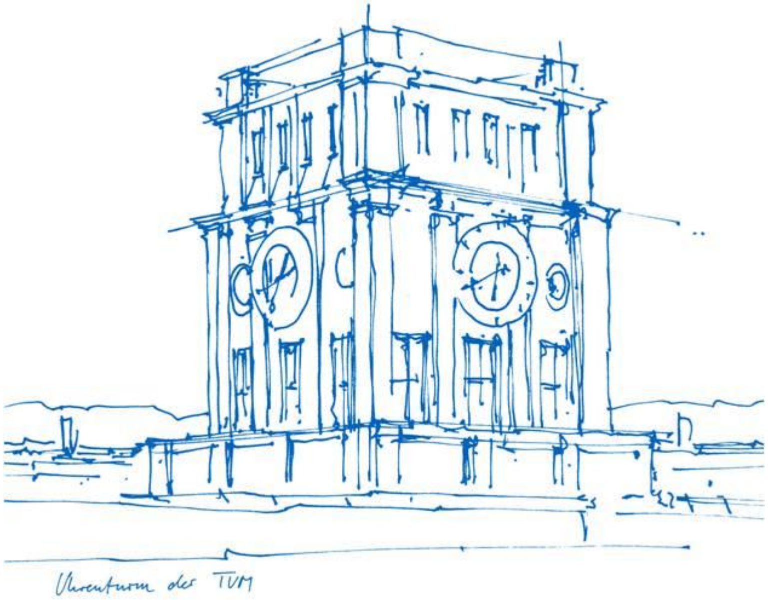
Final Presentation - PreciBake

Ahmad Qasim, Murat Can Karacabey
Abdul Moeed, Sebastian Oehme

Mentors: M. Sundholm, H. Belhassan
TUM-Mentor: M. Rauchensteiner

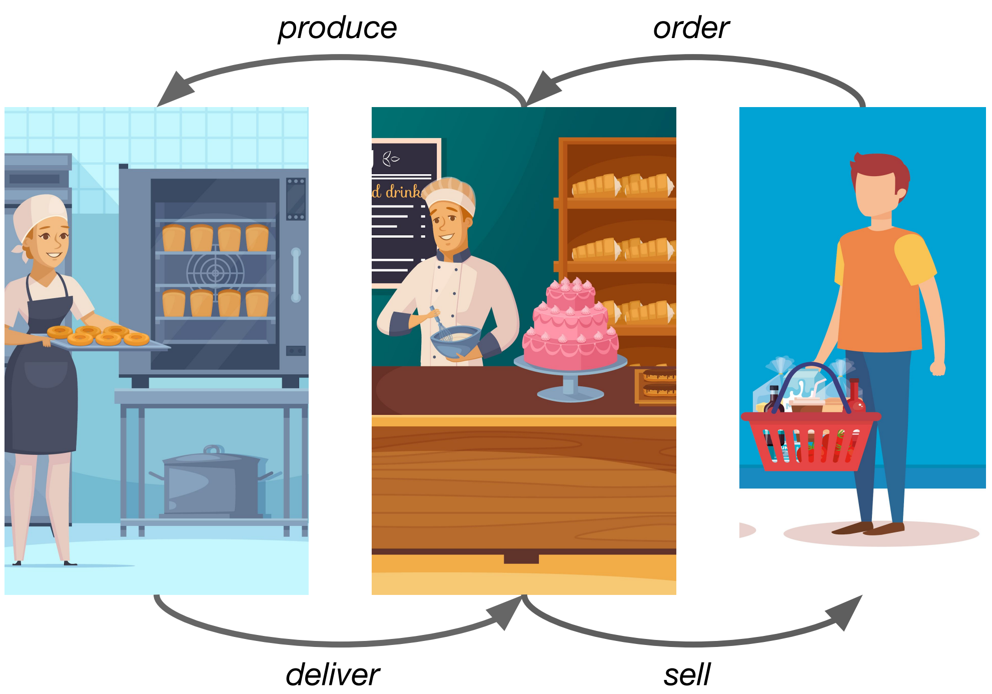
Data Innovation Lab
Department of Mathematics
Technische Universität München

München, 17. February 2020

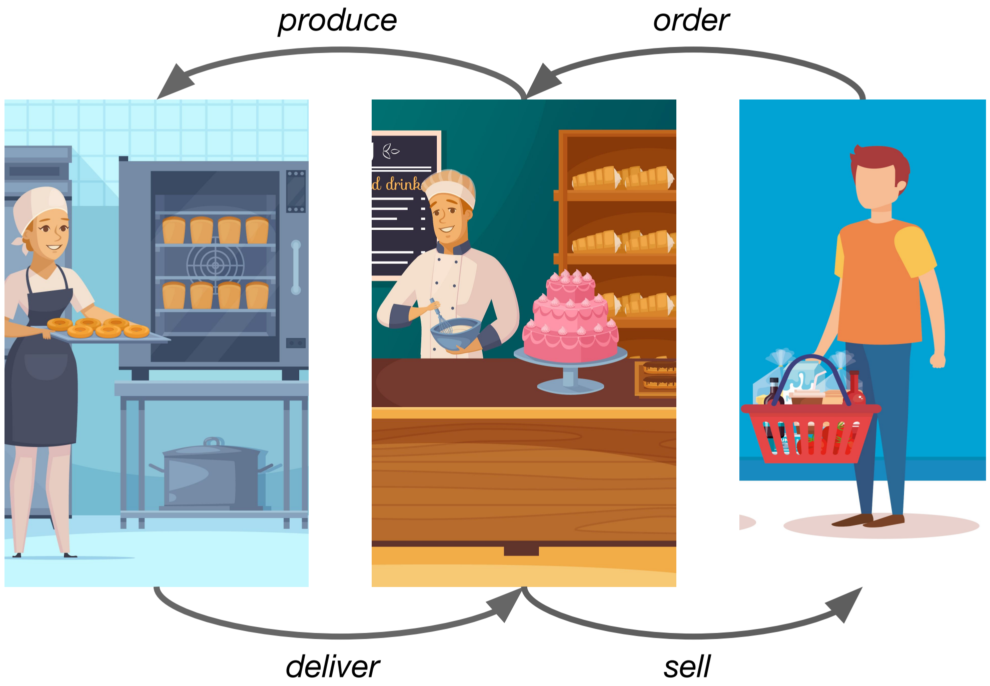


Images: freepik.com

Inventory Control



Inventory Control



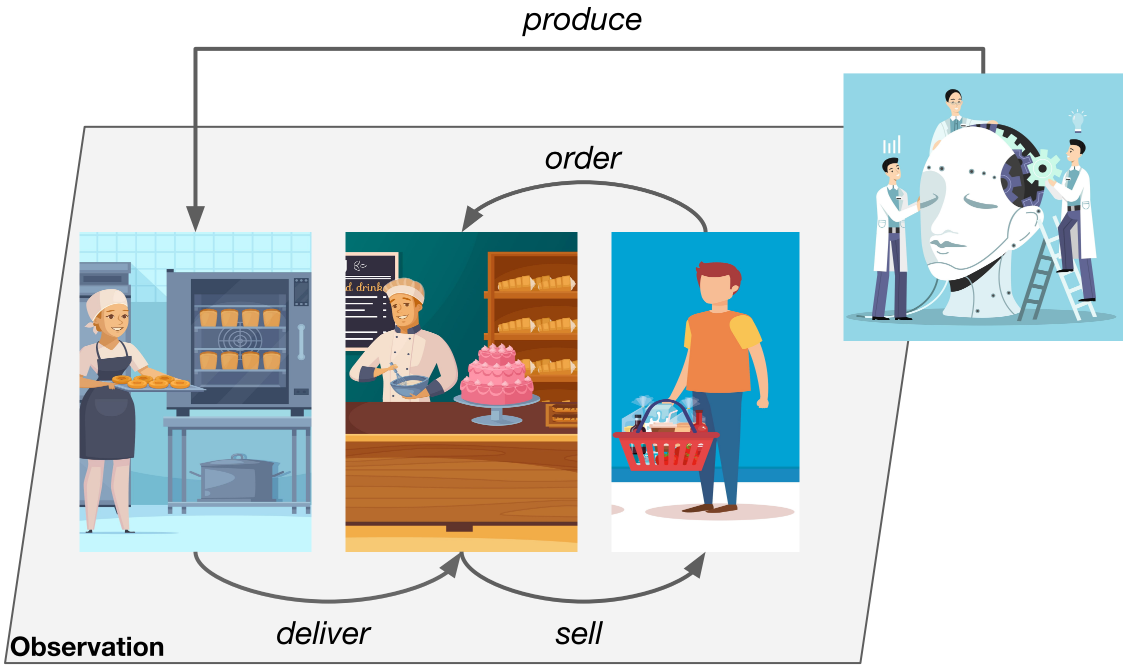
Main goal:

- Maximize product sales

Sub-goals:

- Minimize waste
- Maximize freshness
- Minimize waiting time for customer

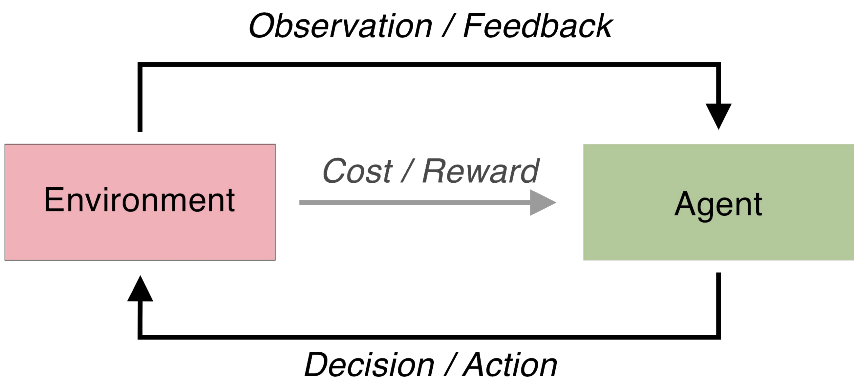
Inventory Control with Reinforcement Learning



Create an agent that

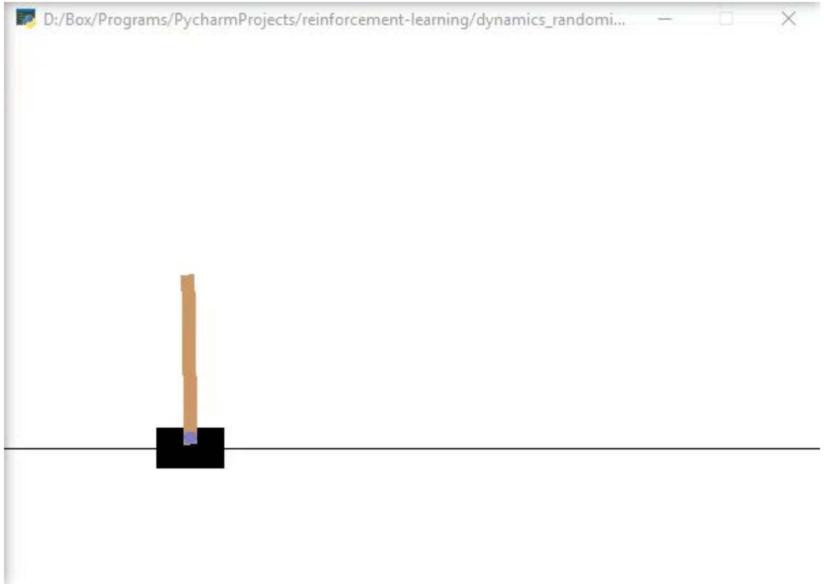
- Satisfies problem constraints
- Safely operates in real world after training on simulation (domain transfer)

Reinforcement Learning (RL)



On each ***episode***, agent must take ***actions*** that maximize expected ***reward*** for each ***state***

Example - RL

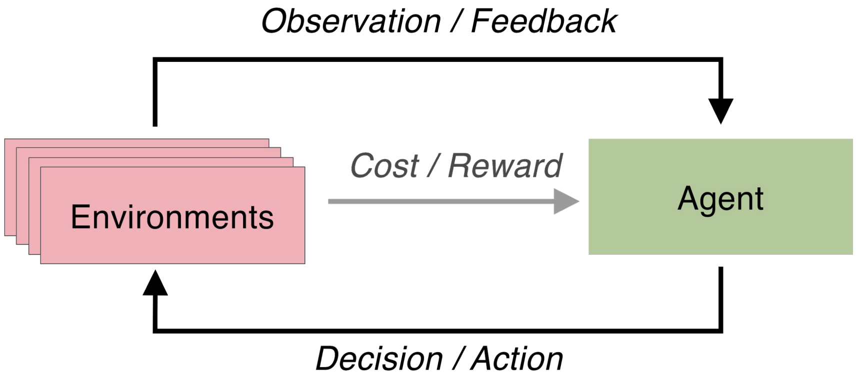


Cartpole - OpenAI Gym

Task: Balance pole on cart

- ***State***
  - Cart position, velocity
  - Pole angle
- ***Actions***
  - *Move left or right*
- ***Reward***
  - *+1 for each time step*
  - *0 when pole falls*

RL with Dynamics Randomization

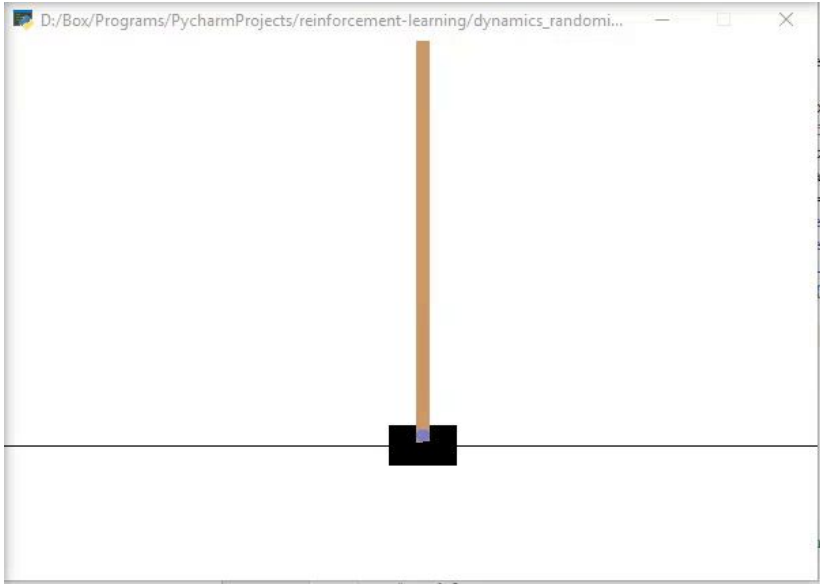


Idea: To generalize to new environments, sample a different environment on each episode

Practice: Environment is defined by certain parameters, sample different values of these parameters on each episode

Data Innovation Lab - WS 2019/20
Department of Mathematics
Technical University of Munich

Dynamics Randomization - Example



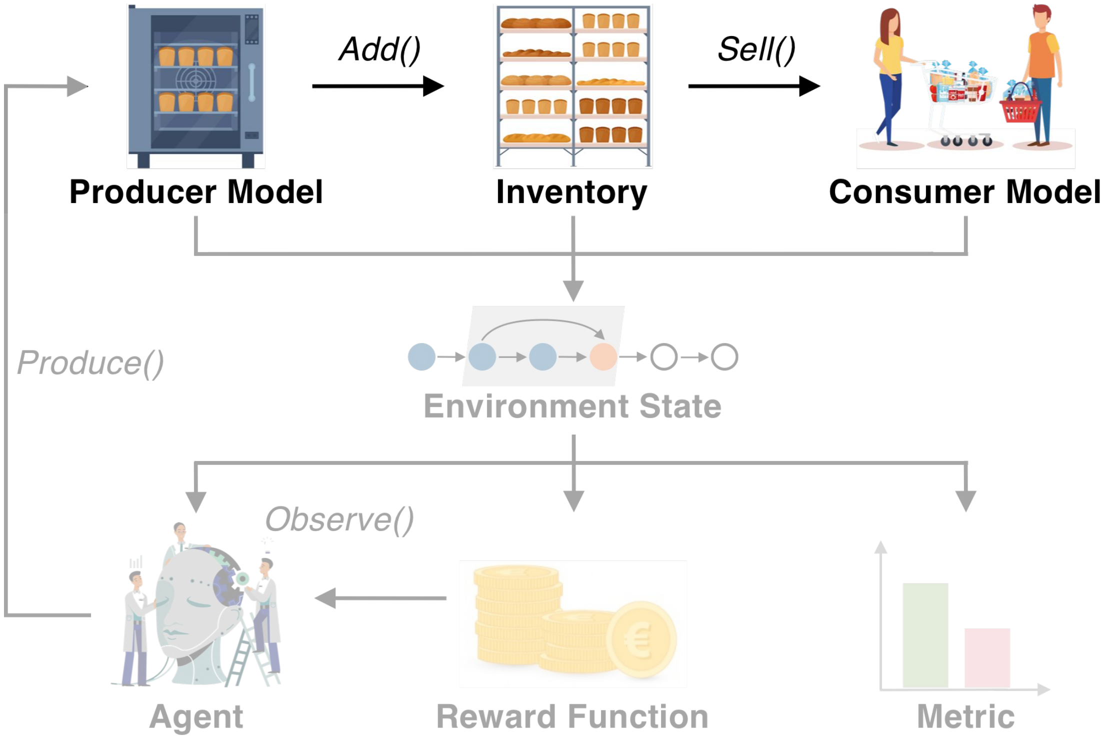
Cartpole - OpenAI Gym

Per episode, sample

- ***Pole Length:***[0.3, 0.6]
- ***Pole Mass:*** [0.5, 1.5]

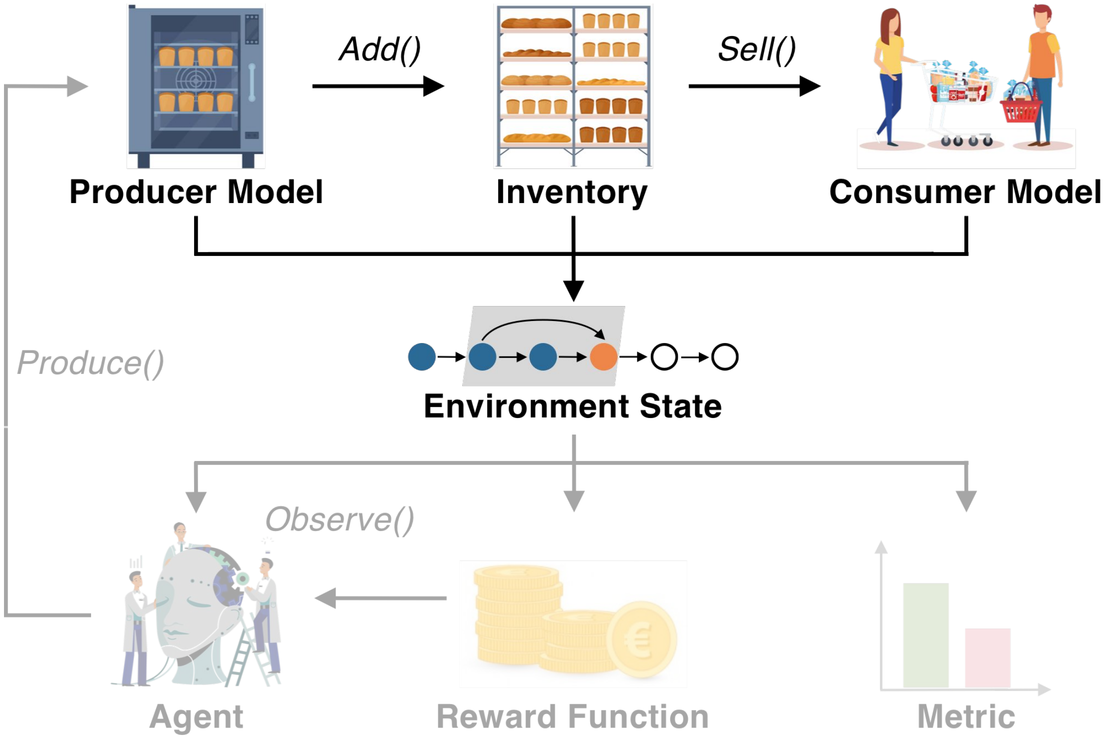
Environment

- **Producer Model**
  - Time-shift
  - Quantity limit
- **Inventory**
  - Storage limit
  - Freshness
- **Consumer Model**
  - Stochastic



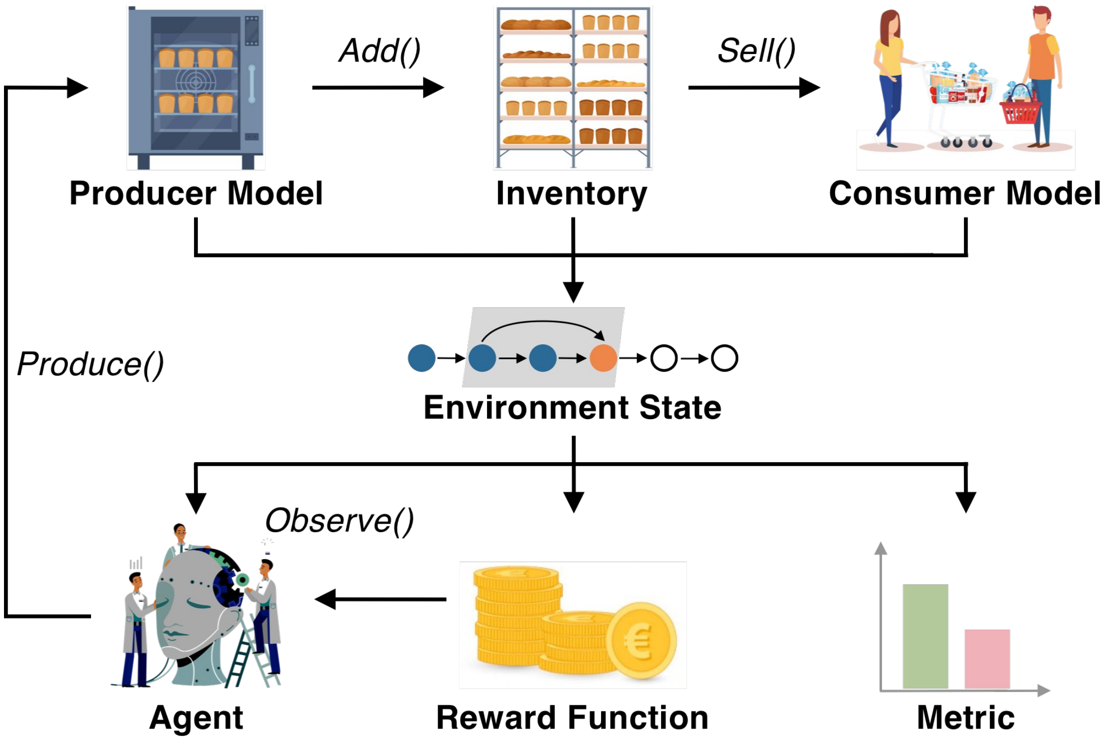
Environment

- **Environment State**
  - Production queue
  - Order queue
  - Inventory
    - Number of the items
    - Age of the items



Environment

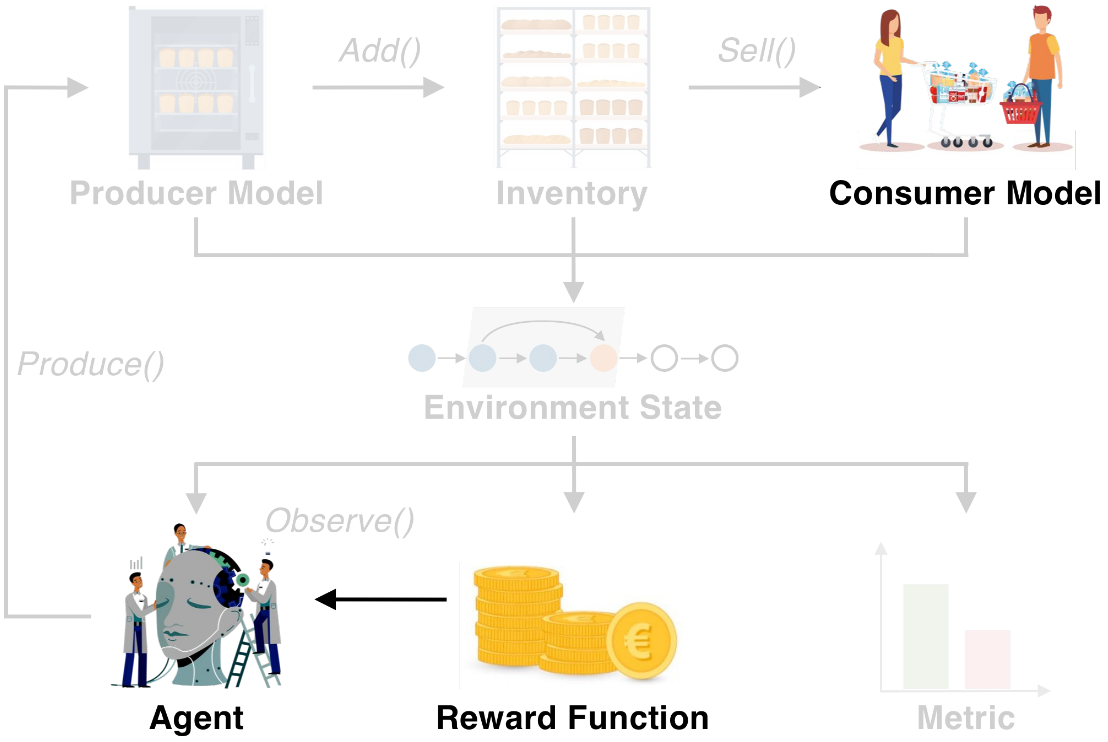
- **Agent**
  - Decides on what to produce
- **Reward**
  - Agent-specific evaluation of the state
- **Metric**
  - Common evaluation of the agents' performances



Data Innovation Lab - WS 2019/20
Department of Mathematics
Technical University of Munich

Environment - Our Task

- **Main task:**
  - Create a robust agent that can perform well in varying demand trends
- **Requires:**
  - An expressive consumer model



Consumer Model

- Simulate behavior of giving orders
- Requirements for the model
    - Expressibility
    - Flexibility
- Problem
    - No real world data available

Consumer Model

- Simulate behavior of giving orders
- Requirements for the model
  - Expressibility
  - Flexibility
- **Problem**
  - **No real world data available**

→

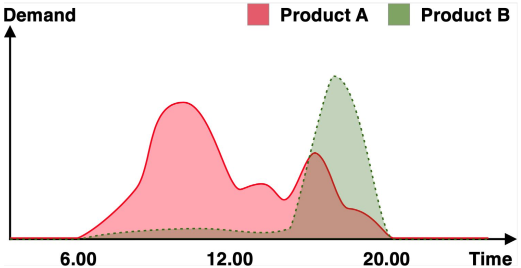
Solution approaches

1. Learn a model from a public dataset
2. Use a generative model to generate data

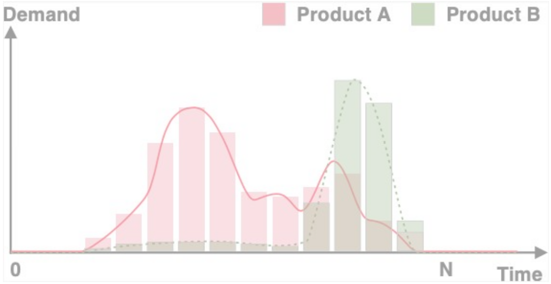
Data Innovation Lab - WS 2019/20
Department of Mathematics
Technical University of Munich

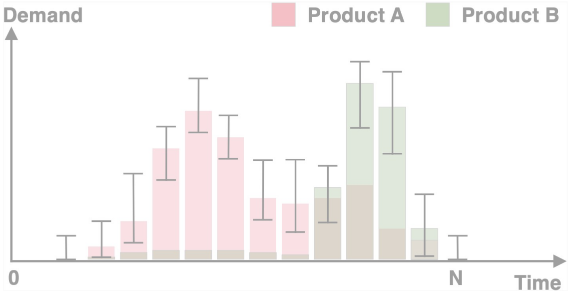
Generative Consumer Model

A generative model that is based on Poisson Distribution



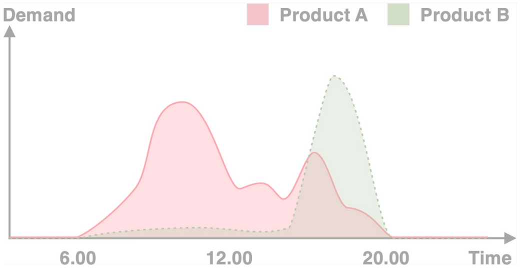
- Example expected demand

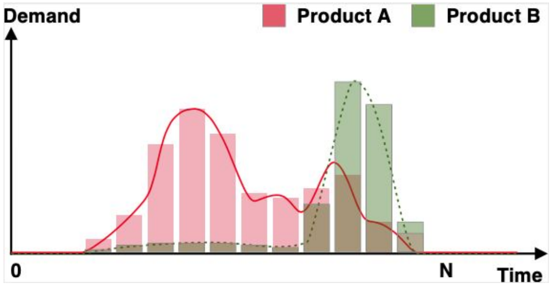


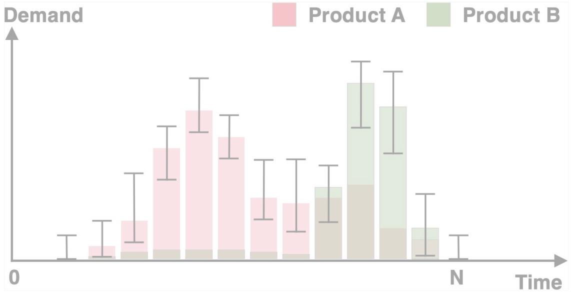


Generative Consumer Model

A generative model that is based on Poisson Distribution





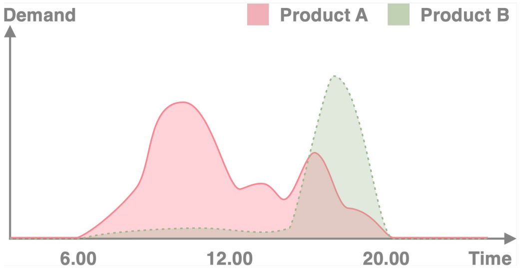


- Example expected demand

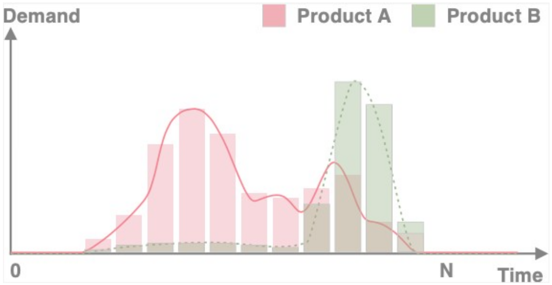
- Create bins with means
- Generate demand

Generative Consumer Model

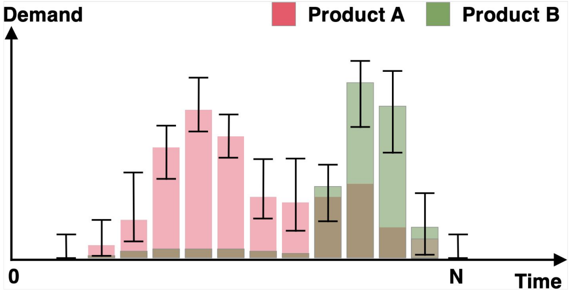
A generative model that is based on Poisson Distribution



- Example expected demand



- Create bins with mean
- Generate demand

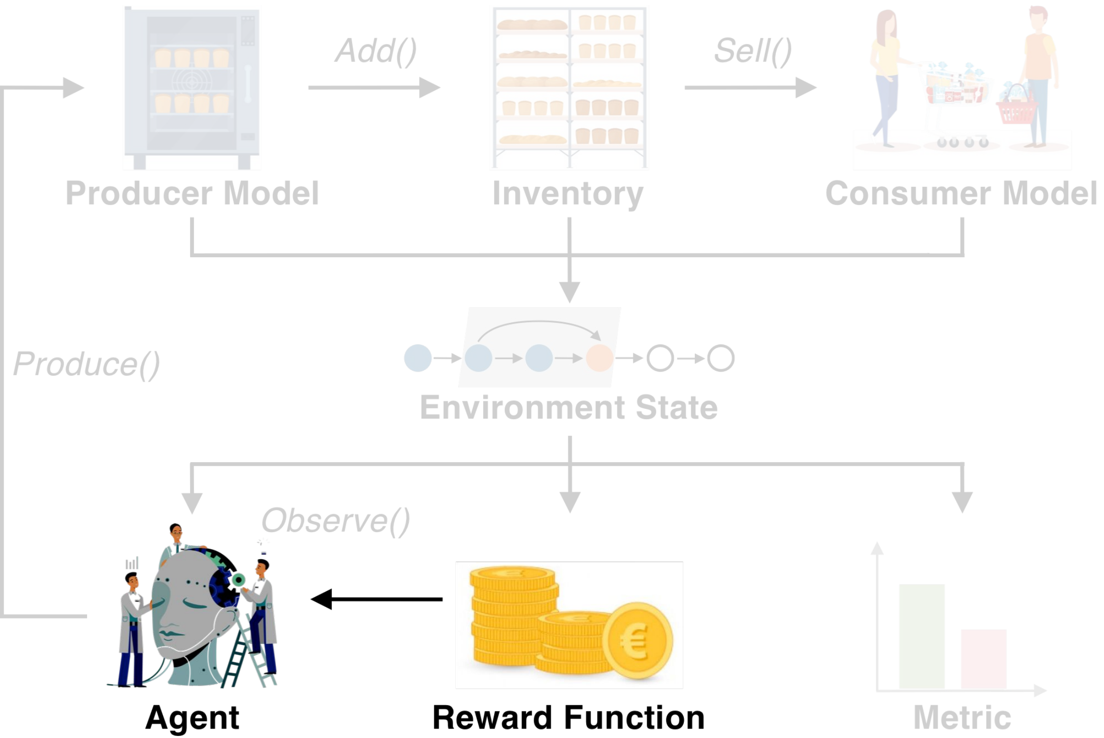


- Vary means for dynamics randomization

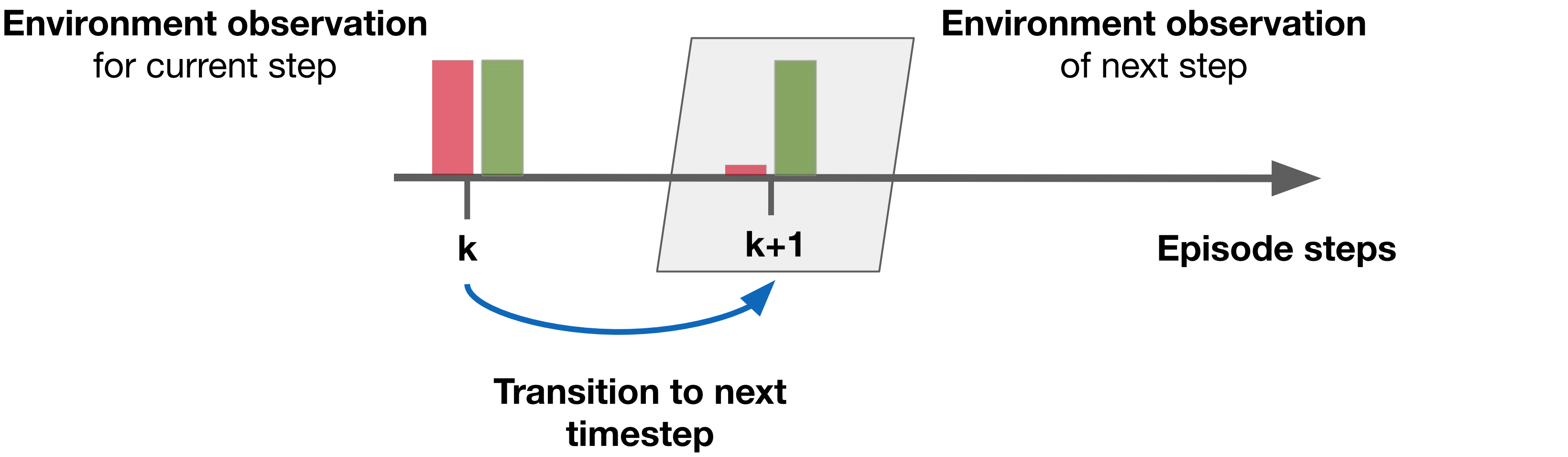
Data Innovation Lab - WS 2019/20
Department of Mathematics
Technical University of Munich

Environment

- **Agent** has to choose
  - Product type
  - Product quantity



Agents Overview



Agents

Baseline

- Benchmark for comparison
- Simulates a real bakery decision process
- Fill inventory if below threshold

Deep Deterministic Policy Gradient (DDPG)

- Model-free
- High-performing RL algorithm
- Using Deep RL to estimate best action
- Reward engineering

Dynamic Programming (DP) with Demand Prediction

- Model-based
- Adapted from D. Bertsekas
- Interpretable and follows the principle of optimality
- Cost function

Agents

Baseline

- Benchmark for comparison
- Simulates a real bakery decision process
- Fill inventory if below threshold

Deep Deterministic Policy Gradient (DDPG)

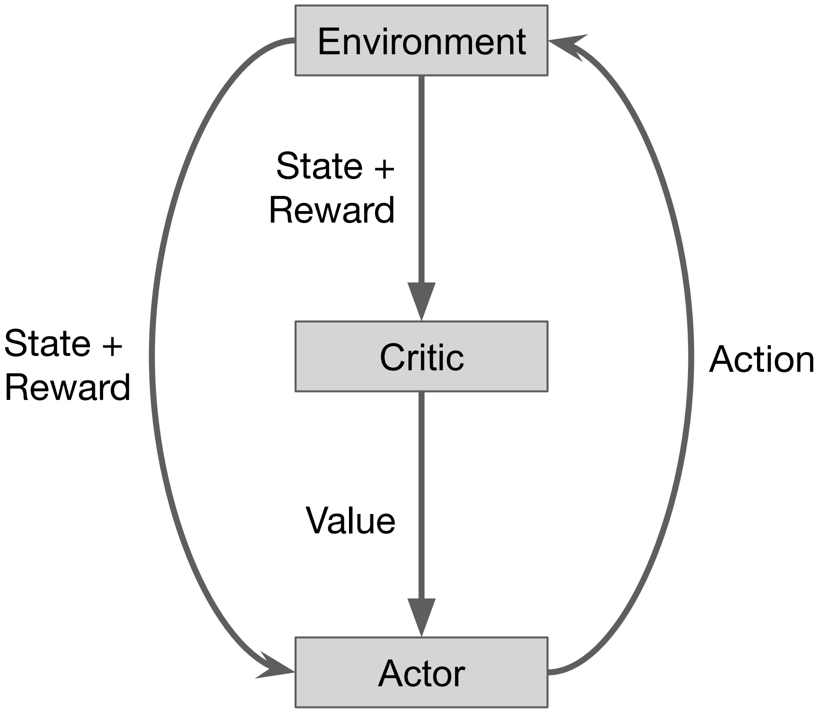
- Model-free
- High-performing RL algorithm
- Using Deep RL to estimate best action
- Reward engineering

Dynamic Programming (DP) with Demand Prediction

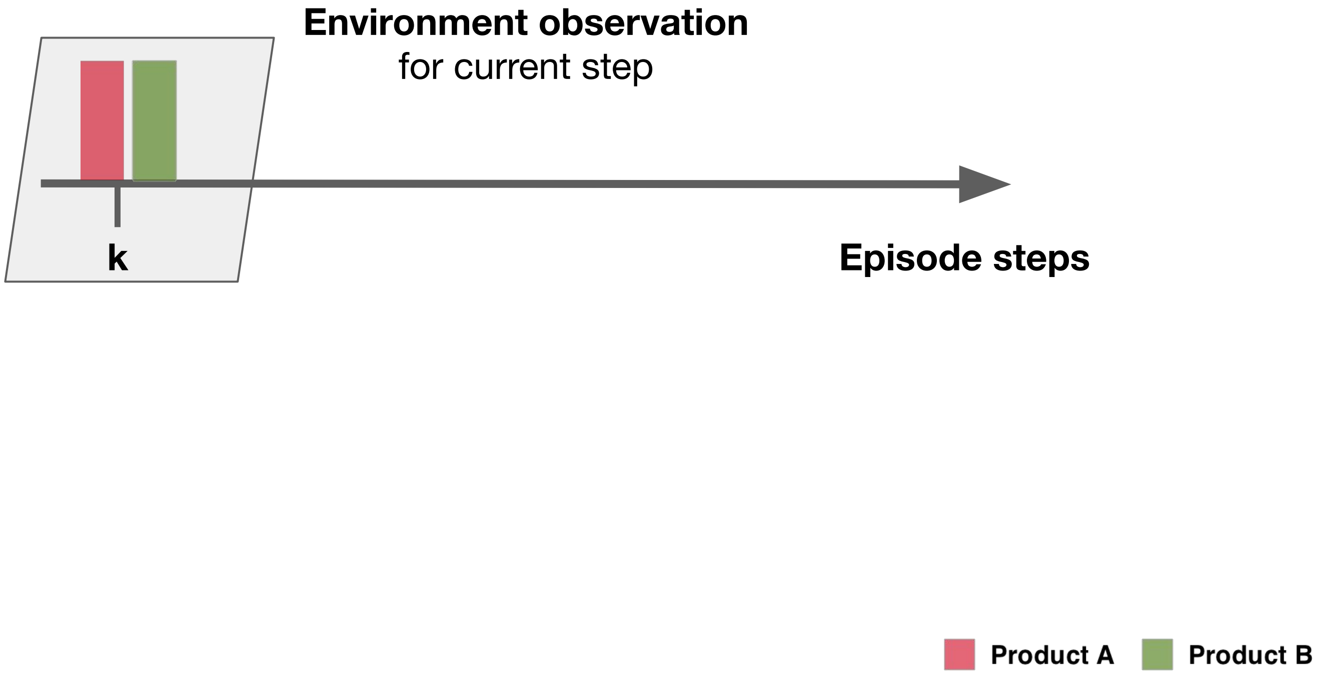
- Model-based
- Adapted from D. Bertsekas
- Interpretable and follows the principle of optimality
- Cost function

DDPG

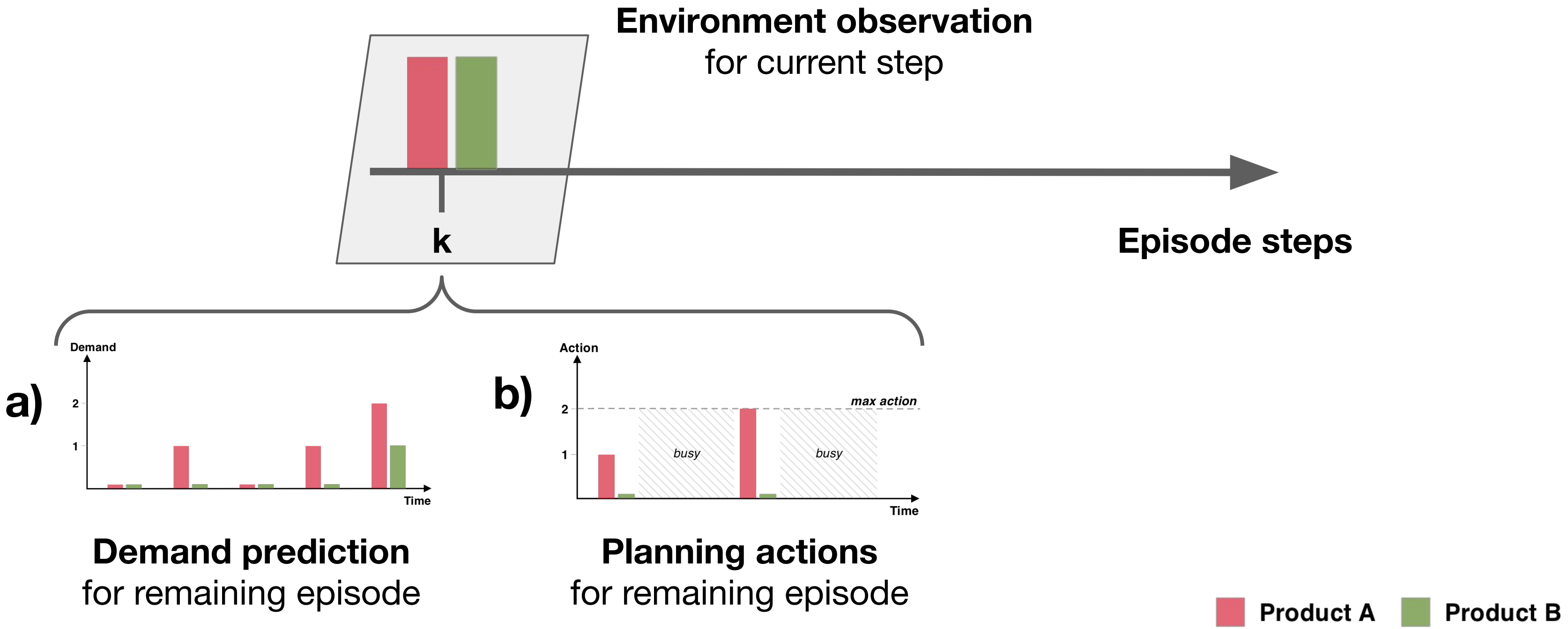
- Environment has a state and returns a reward
- Critic estimates the value of state
- Actor generates actions
- Challenges:
  - Several hyper-parameters to tune
  - Reward engineering



DP Agent overview



DP Agent overview



Demand Model

- Predicts future consumer orders
- DP agent acts based on it
- DP agent's performance, highly dependent on demand model
  - ➢ Tried different approaches

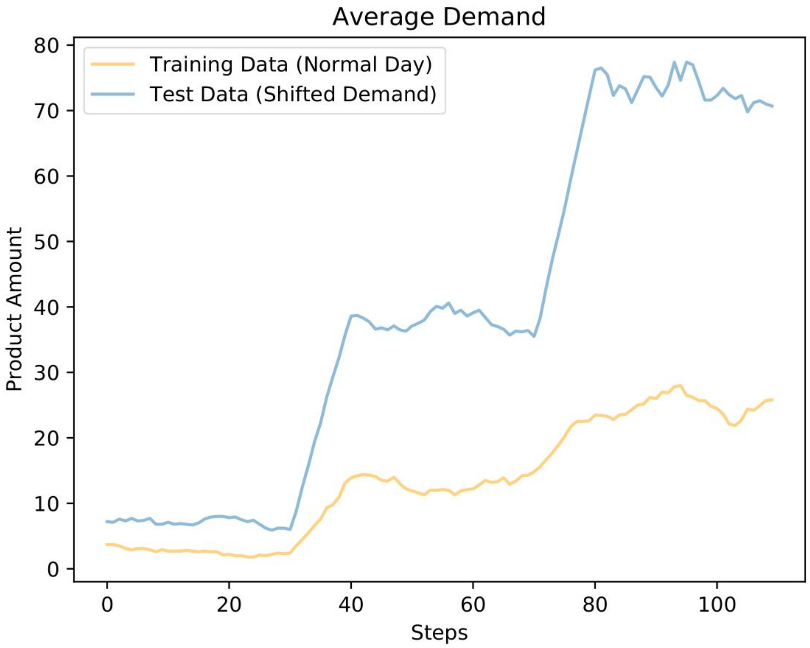
Demand Model

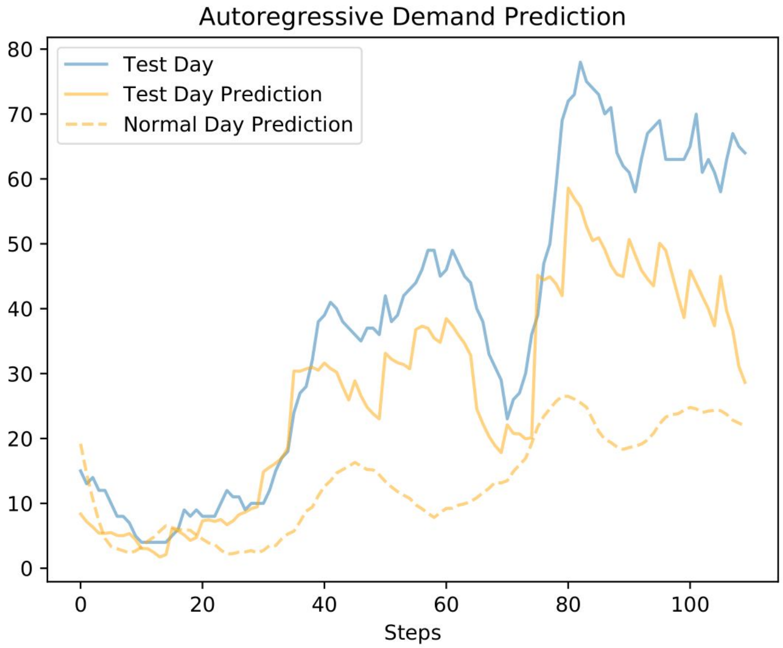
- Predicts future consumer orders
- DP agent acts based on it
- DP agent’s performance, highly dependent on demand model
  - ➢ Tried different approaches

Final model: Autoregressive Model (AR)

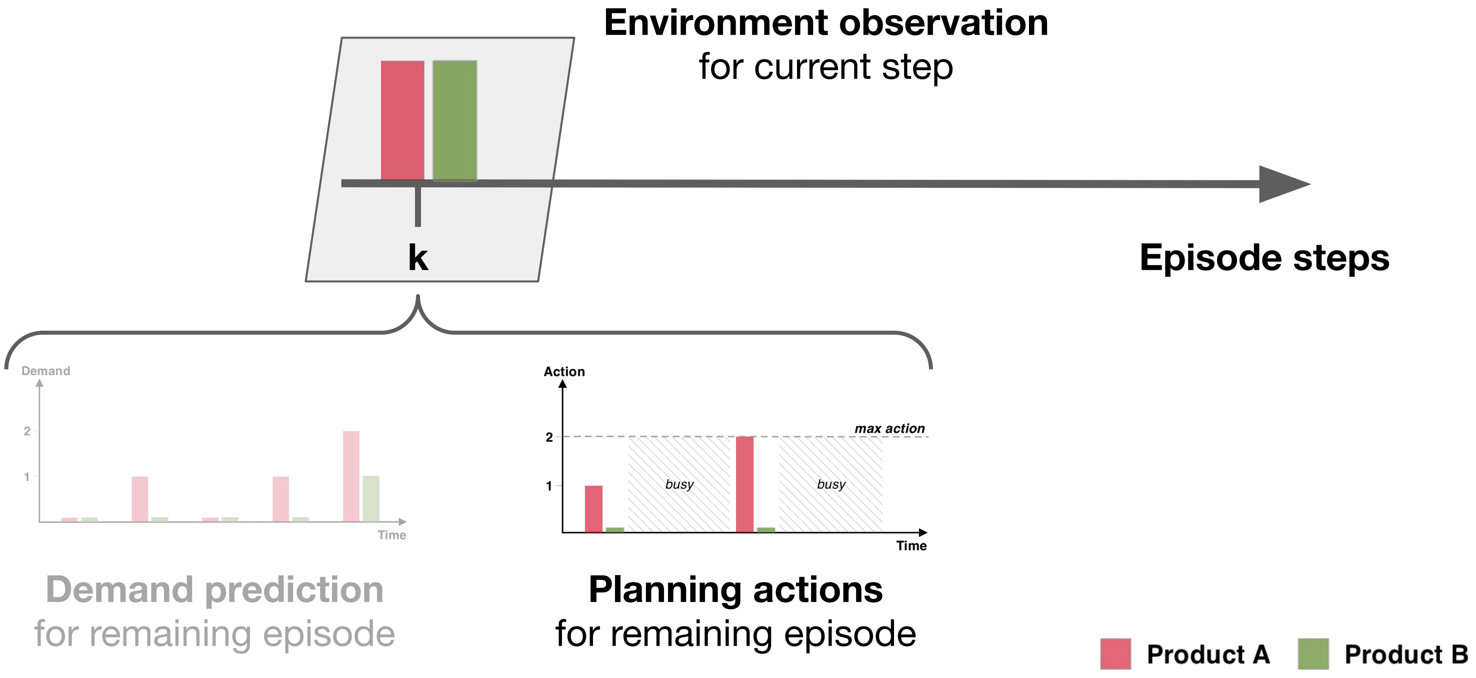
- predicts based on real consumer orders
- self-adjusts

Demand Model - AR





DP Agent


Environment observation
for current step
k
Episode steps
Demand
2
1
Time
Demand prediction
for remaining episode
Action
2
1
max action
busy
busy
Time
Planning actions
for remaining episode
Product A
Product B

Dynamic Programming

- **Discrete-time equation**

$$\tilde{x}^i_{k+1} = \tilde{x}^i_k + d^i_k - w^i_k$$

  - Inventory $\tilde{x}^i_k$
  - Delivery d^i_k
  - Demand w^i_k

- **Incorporated**
  - Production time-shift
  - Multi-products

Dynamic Programming Algorithm

- **Discrete-time equation**

  $$\tilde{x}^i_{k+1} = \tilde{x}^i_k + d^i_k - w^i_k$$

  - Inventory $\tilde{x}^i_k$
  - Delivery d^i_k
  - Demand w^i_k

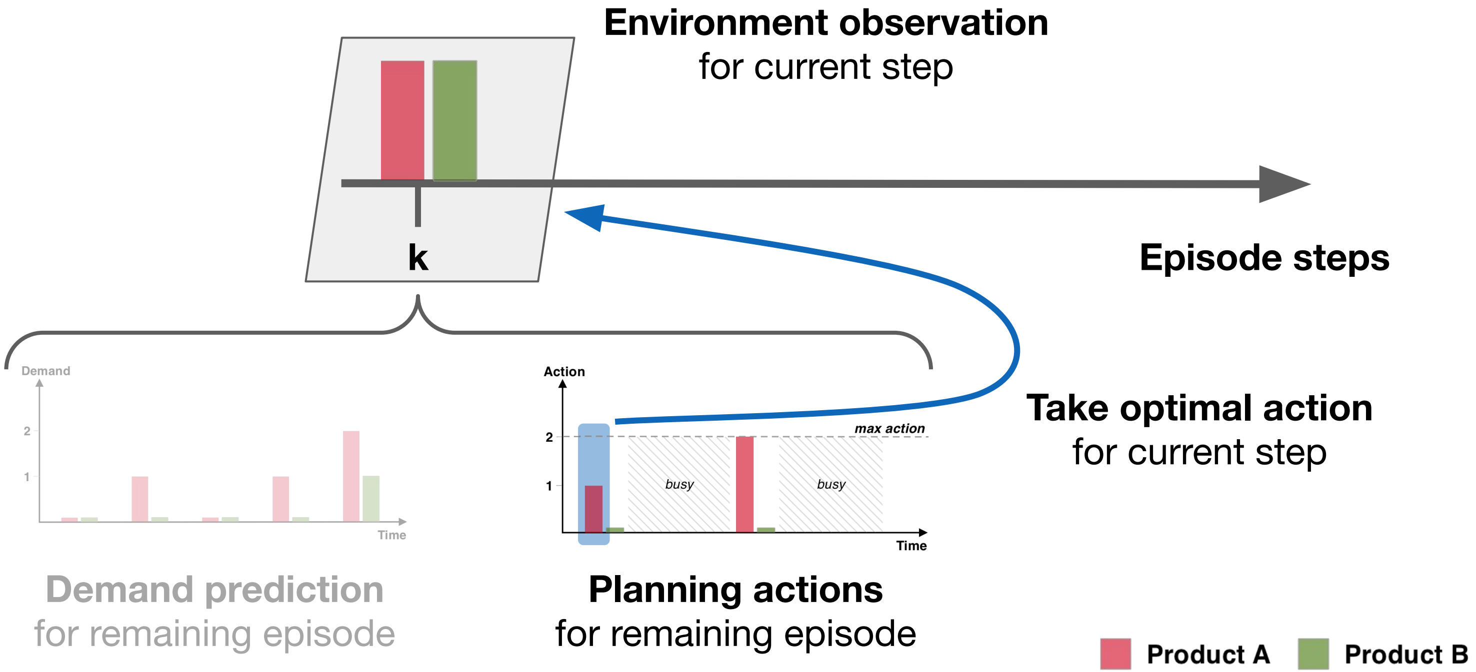
- cost-to-go g_k

- **Inventory Control**
  - Fixed demand to the prediction: $\overline{w}^i_k$
  - Solve deterministic problem:

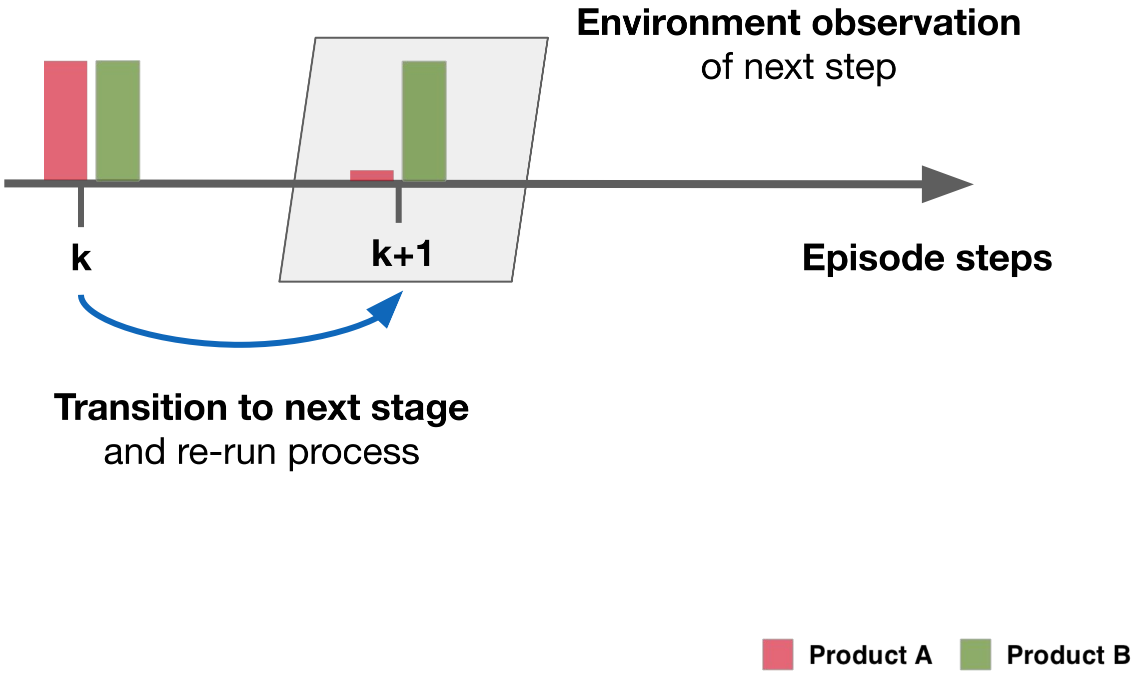
  $$\text{minimize } g_N(\tilde{x}^i_N) + \sum_{k}^{N-1} g_k(\tilde{x}^i_k, d^i_k, \overline{w}^i_k)$$

- **Uses *Principle of Optimality*** (Bellmann)
  - Sequentially calculate optimal costs of tail subproblems
  - Going from shorter to longer problems

DP Agent


Environment observation
for current step
k
Episode steps
Demand
2
1
Time
Action
max action
2
1
busy
busy
Time
Take optimal action
for current step
Demand prediction
for remaining episode
Planning actions
for remaining episode
Product A
Product B

DP Agent


Environment observation
of next step
k
k+1
Episode steps
Transition to next stage
and re-run process
Product A
Product B

Results - Test cases

- In-distribution
  - ➢ Similar consumer demand (distribution) as training
- Out-of-distribution (domain transfer to new day / location)
  - ➢ **+50%** consumer demand — **“Busy”**
  - ➢ **-50%** consumer demand — **“Idle”**
  - ➢ **[-50%,+50%]** consumer demand — **“Chaotic”**

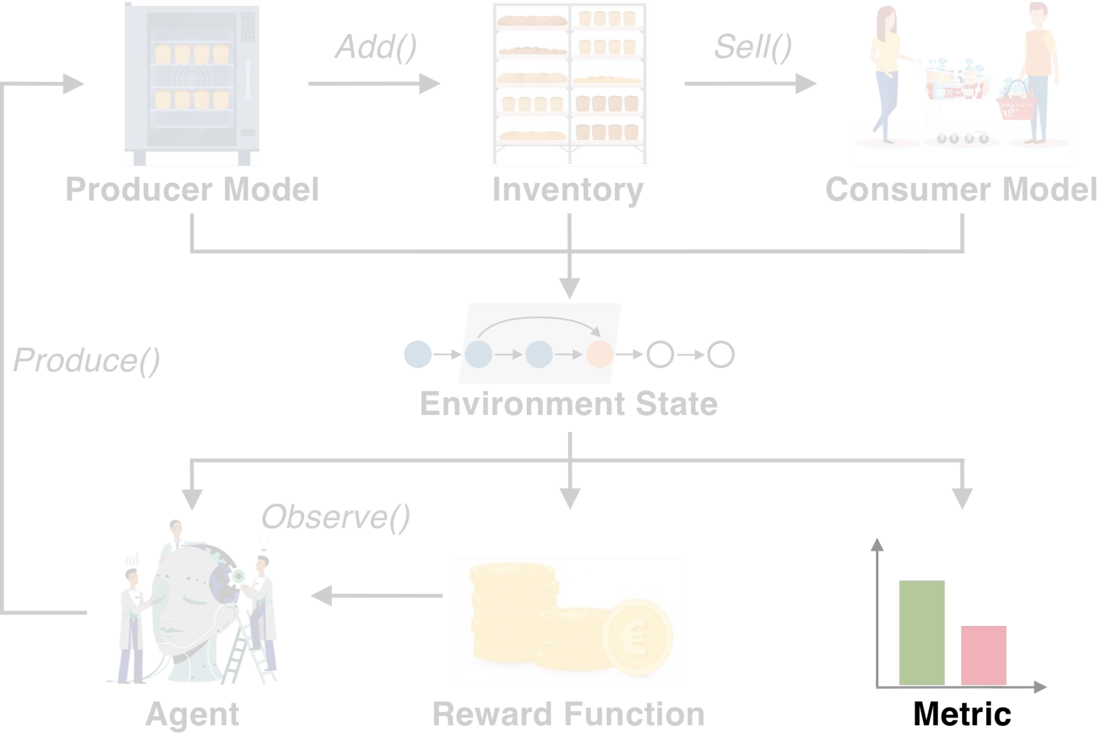
Metrics

Product sales: s_{ratio}

sales out of total orders

Freshness: p_{ratio}
(reciprocal)

average age of the products



Results

	Sales				***Freshness***			
	Similar demand	**Higher mean demand (+50%)**	**Lower mean demand (-50%)**	**Random mean ([-50%,+50%])**	**Similar demand**	**Higher mean demand (+50%)**	**Lower mean demand (-50%)**	**Random mean ([-50%,+50%])**
Baseline								
DDPG								
DDPG w/ DR								
DP w/ AR								
DP w/ Oracle								

Color scale for *sales* ratio

0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000

Color scale for *freshness* (reciprocal) ratio

0.000	1.625	3.250	4.875	6.500	8.125	9.750	11.375	13.000

Data Innovation Lab - WS 2019/20
Department of Mathematics
Technical University of Munich

Results

Sales

	Similar demand	Higher mean demand (+50%)	Lower mean demand (-50%)	Random mean ([-50%,+50%])
Baseline	0.5503	0.4121	0.8188	0.5867
DDPG				
DDPG w/ DR				
DP w/ AR				
DP w/ Oracle				

Freshness

	Similar demand	Higher mean demand (+50%)	Lower mean demand (-50%)	Random mean ([-50%,+50%])
Baseline	5.3800	4.0300	10.1690	6.4300
DDPG				
DDPG w/ DR				
DP w/ AR				
DP w/ Oracle				

Color scale for *sales* ratio

0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000

Color scale for *freshness* (reciprocal) ratio

0.000	1.625	3.250	4.875	6.500	8.125	9.750	11.375	13.000

Baseline - Produces more than required, products less fresh

Results

	Sales				Freshness			
	Similar demand	Higher mean demand (+50%)	Lower mean demand (-50%)	Random mean ([-50%,+50%])	Similar demand	Higher mean demand (+50%)	Lower mean demand (-50%)	Random mean ([-50%,+50%])
Baseline	0.5503	0.4121	0.8188	0.5867	5.3800	4.0300	10.1690	6.4300
DDPG								
DDPG w/ DR								
DP w/ AR								
DP w/ Oracle	***0.6716***	***0.488***	***0.9521***	***0.6347***	3.5250	2.7700	2.4500	3.6650

Color scale for *sales* ratio

0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000

Color scale for *freshness* (reciprocal) ratio

0.000	1.625	3.250	4.875	6.500	8.125	9.750	11.375	13.000

DP with Oracle - Upper bound

Results

	Sales				*Freshness*			
	Similar demand	**Higher mean demand (+50%)**	**Lower mean demand (-50%)**	**Random mean ([-50%,+50%])**	**Similar demand**	**Higher mean demand (+50%)**	**Lower mean demand (-50%)**	**Random mean ([-50%,+50%])**
Baseline	0.5503	0.4121	0.8188	0.5867	5.3800	4.0300	10.1690	6.4300
DDPG	0.4412	0.3316	0.7343	0.4222	2.5700	1.5399	6.5000	2.1500
DDPG w/ DR	0.2512	0.1902	0.4055	0.2597	***1.6750***	***0.9650***	***1.9200***	***1.2600***
DP w/ AR								
DP w/ Oracle	***0.6716***	***0.488***	***0.9521***	***0.6347***	3.5250	2.7700	2.4500	3.6650

Color scale for *sales* ratio

0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000

Color scale for *freshness* (reciprocal) ratio

0.000	1.625	3.250	4.875	6.500	8.125	9.750	11.375	13.000

DDPG, DDPG with Dynamics Randomization - Produces less than required

Results

	Sales				***Freshness***			
	Similar demand	**Higher mean demand (+50%)**	**Lower mean demand (-50%)**	**Random mean ([-50%,+50%])**	**Similar demand**	**Higher mean demand (+50%)**	**Lower mean demand (-50%)**	**Random mean ([-50%,+50%])**
Baseline	0.5503	0.4121	0.8188	0.5867	5.3800	4.0300	10.1690	6.4300
DDPG	0.4412	0.3316	0.7343	0.4222	2.5700	1.5399	6.5000	2.1500
DDPG w/ DR	0.2512	0.1902	0.4055	0.2597	***1.6750***	***0.9650***	***1.9200***	***1.2600***
DP w/ AR	0.5649	0.4013	0.8934	0.5736	3.2736	1.6421	13.0000	3.6578
DP w/ Oracle	***0.6716***	***0.488***	***0.9521***	***0.6347***	3.5250	2.7700	2.4500	3.6650

Color scale for *sales* ratio

0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000

Color scale for *freshness* (reciprocal) ratio

0.000	1.625	3.250	4.875	6.500	8.125	9.750	11.375	13.000

DP with Autoregressive (AR) Prediction - Balances well between both metrics

Results

	Sales				***Freshness***			
	Similar demand	**Higher mean demand (+50%)**	**Lower mean demand (-50%)**	**Random mean ([-50%,+50%])**	**Similar demand**	**Higher mean demand (+50%)**	**Lower mean demand (-50%)**	**Random mean ([-50%,+50%])**
Baseline	0.5503	0.4121	0.8188	0.5867	5.3800	4.0300	10.1690	6.4300
DDPG	0.4412	0.3316	0.7343	0.4222	2.5700	1.5399	6.5000	2.1500
DDPG w/ DR	0.2512	0.1902	0.4055	0.2597	***1.6750***	***0.9650***	***1.9200***	***1.2600***
DP w/ AR	0.5649	0.4013	0.8934	0.5736	3.2736	1.6421	13.0000	3.6578
DP w/ Oracle	***0.6716***	***0.488***	***0.9521***	***0.6347***	3.5250	2.7700	2.4500	3.6650

Color scale for *sales* ratio

0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000

Color scale for *freshness* (reciprocal) ratio

0.000	1.625	3.250	4.875	6.500	8.125	9.750	11.375	13.000

Overall - good performance, improvements possible

Summary

- **Brief Overview**
  - Explore the available methods for domain transfer
  - Create a consumer model suitable for the task
  - Create a demand model that predicts the demands
  - Implement Model-free (DDPG) and Model-based (DP) agents
  - Compare agents with the baseline
- **Future Work**
  - Improvement in consumer model with learning-based methods (e.g. LSTM)
  - Approximate Dynamic Programming in case the state space is high

Thank you for your attention! Questions?

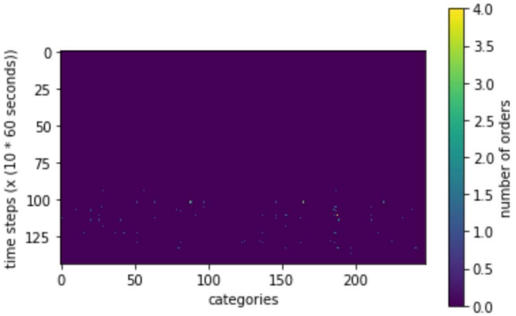
Image kochrezepte.at

Data Innovation Lab - WS 2019/20
Department of Mathematics
Technical University of Munich

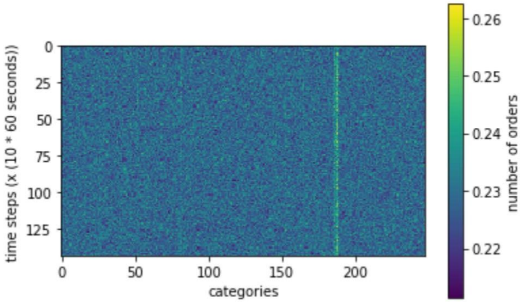
Backup - VAE

Take away Data

Training

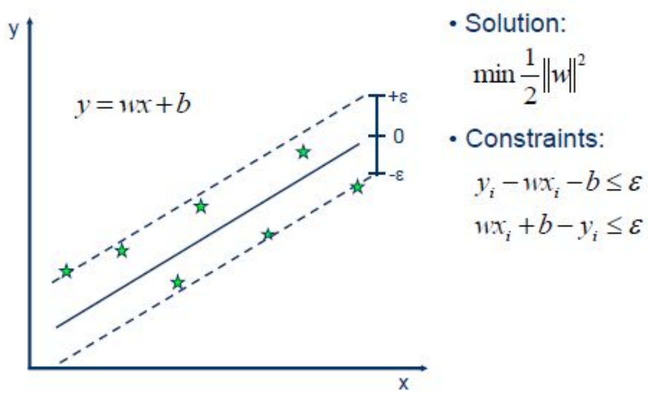


Generated



Backup - Support Vector Regression

- Kernel Function maps data to higher dimensions
- At higher dimensions linear separation is possible
- A hyperplane between the data points
- A soft margin of tolerance for the hyperplane
- Minimizes the MSE (mean squared error)



AQ

Backup - Autoregressive Model

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \epsilon_t$$

Here, Φ are the parameters, X are the observations, ε is white noise and c is the constant

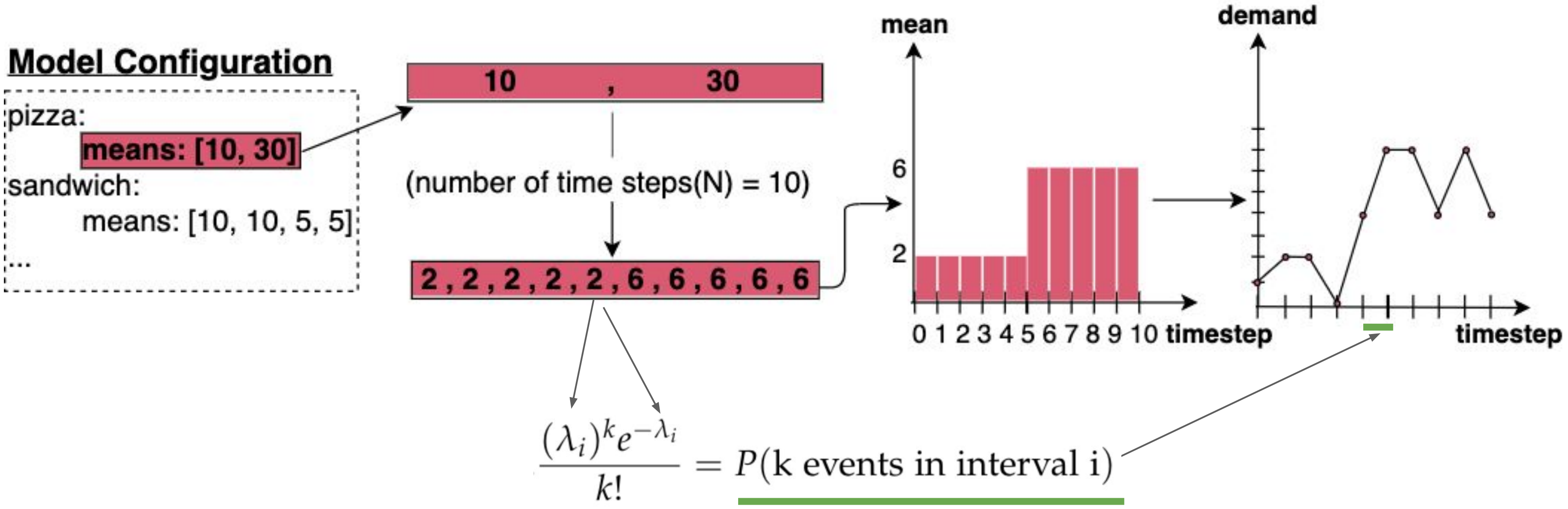
- The output depends on the previous observations X
- The parameters Φ are adjusted
- Goal is to minimize MSE (mean squared error)

AM

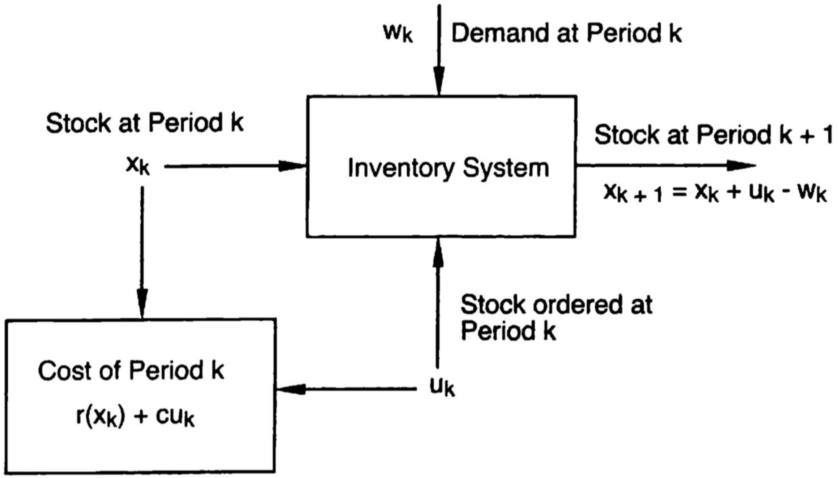
Backup - Consumer models

- **Average**
  - For given time ***t***, take average of all orders at this time
- **Random Nearest-Neighbor**
  - For given time ***t***, go to nearest time ***t****, and sample from all orders in ***t**** in a uniform random manner
- **Linear Regression / Support Vector Regression**
  - For ***n*** products, train ***n*** linear or support vector models

Backup - Poisson Consumer Model



Backup - Original from D. Bertsekas



- Original example from D. Bertsekas

- Discrete-time system

$$x_{k+1} = f_k(x_k, u_k, w_k) = x_k + u_k - w_k$$

- Cost function that is additive over time

$$E\left\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)\right\}$$
$$= E\left\{\sum_{k=0}^{N-1}\left(cu_k + r(x_k + u_k - w_k)\right)\right\}$$

$$J_\pi(x_0) = E\left\{R(x_N) + \sum_{k=0}^{N-1}\Big(r(x_k) + c\mu_k(x_k)\Big)\right\}$$

$\mu_k(x_k) =$ amount that should be ordered at time k if the stock is x_k

Backup - Metric

$$s_{ratio} = \frac{o_f}{o_f + o_u}$$

$$p_{ratio} = \frac{\sum_{i=0}^{N} a_i}{N}$$

o_f = number of fulfilled orders,

o_u = number of unfulfilled orders

a_i = age of all products at time-step i in the inventory,

N = number of time-steps.