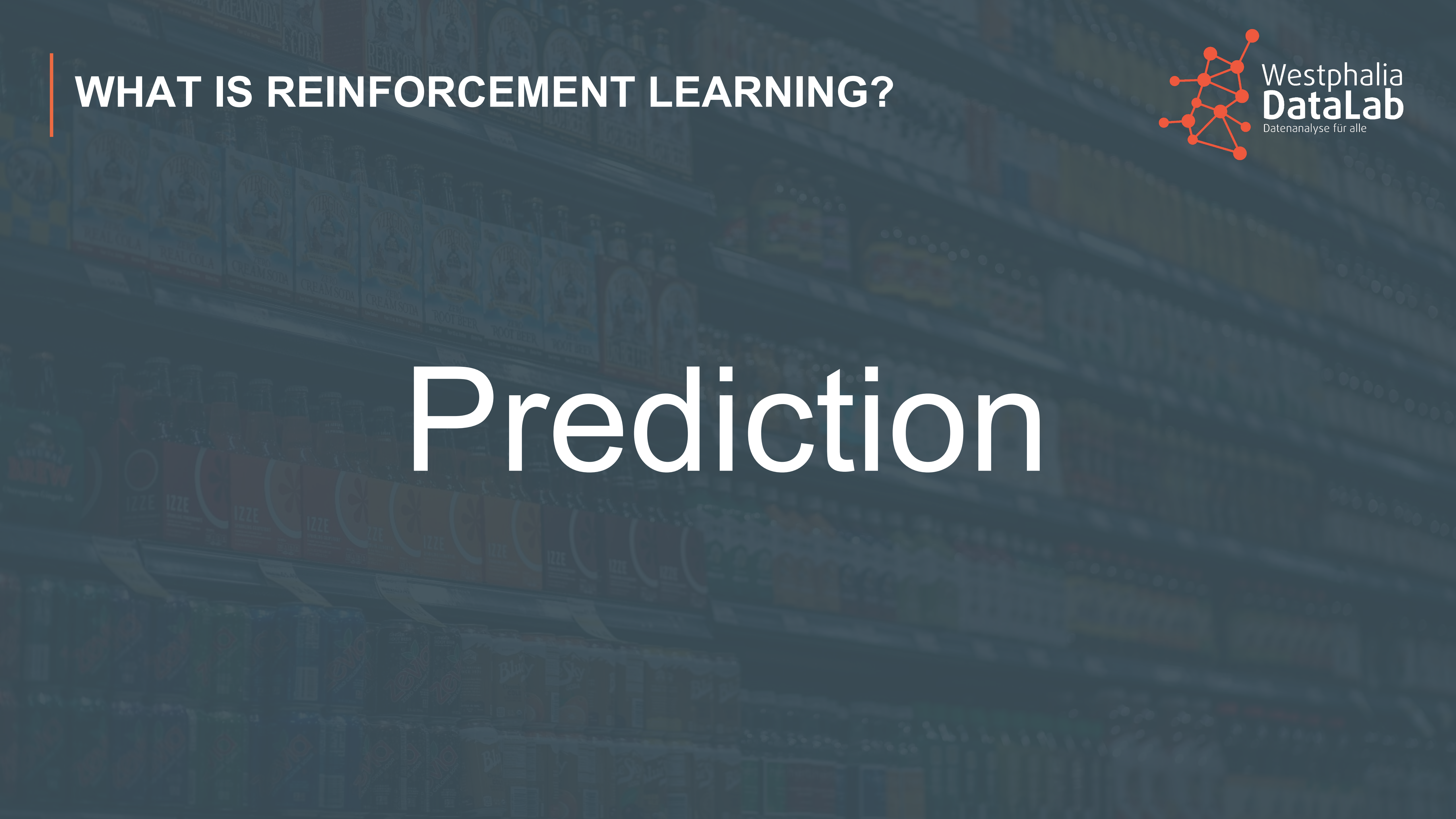
Westphalia DataLab
Datenanalyse für alle

# Reinforcement Learning in Logistics – A Warehouse Management Case Study

David Middelbeck | Head of Product | Westphalia DataLab GmbH

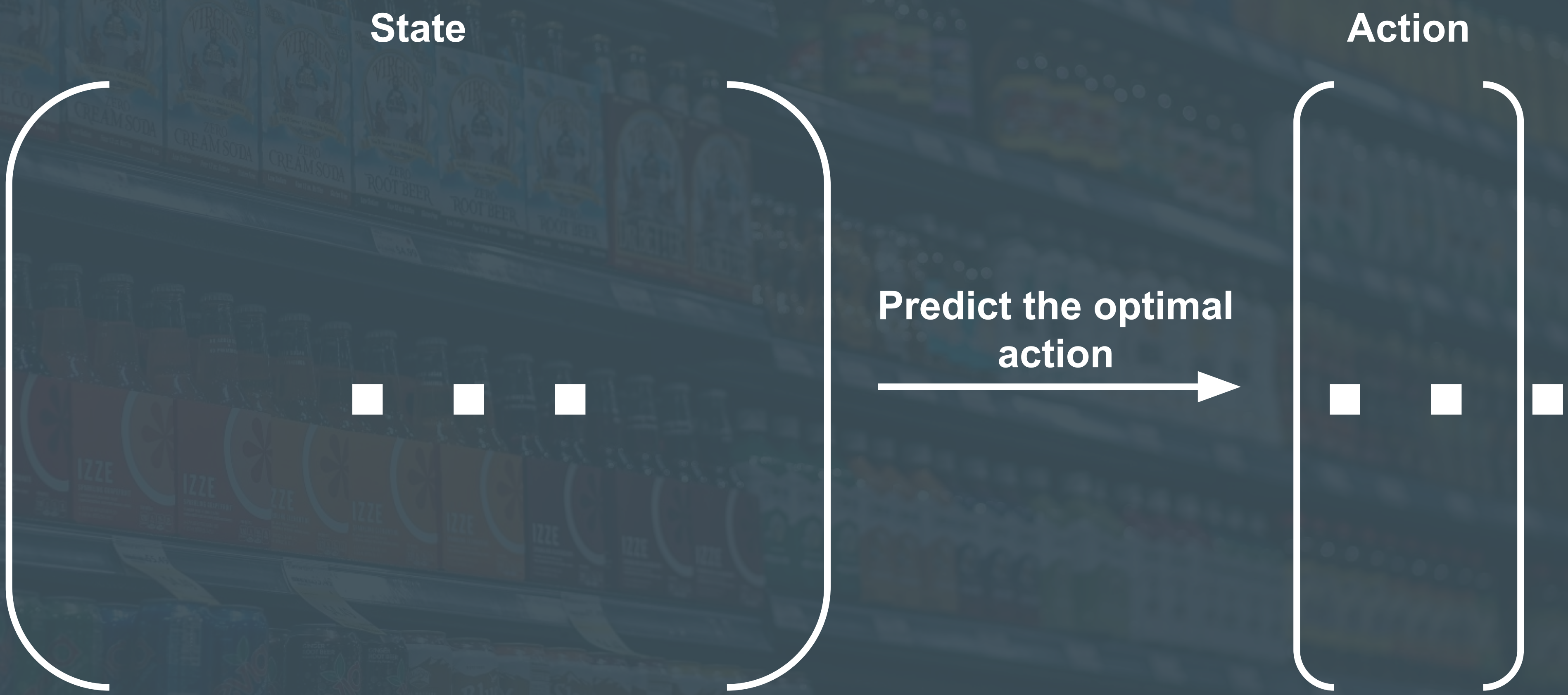
Marcus Cramer | Head of Analytics | Westphalia DataLab GmbH



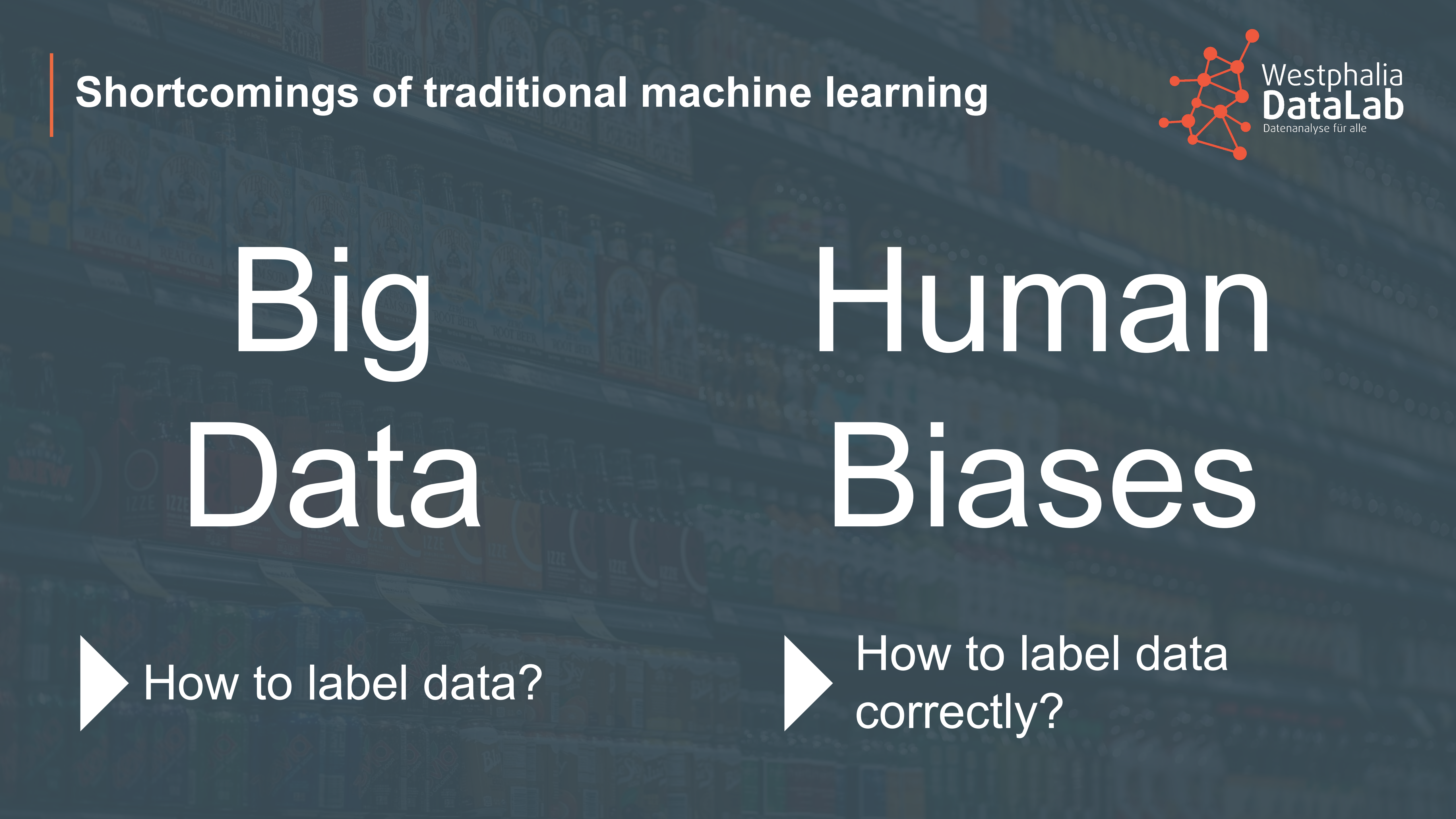
# WHAT IS REINFORCEMENT LEARNING?

## Prediction

# Reinforcement Learning – Just a fancy phrase for supervised learning?



# Shortcomings of traditional machine learning



## Big Data

- How to label data?

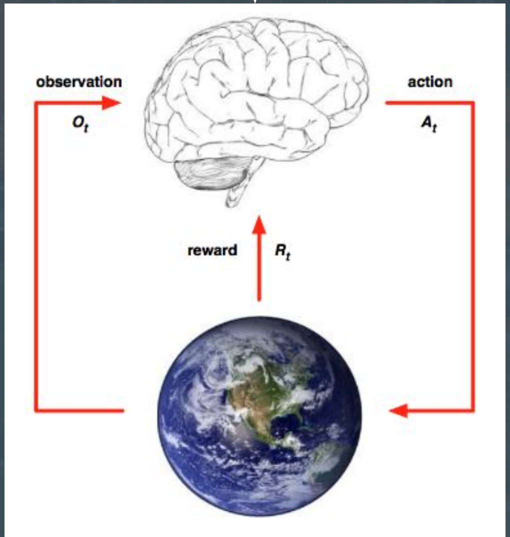
## Human Biases

- How to label data correctly?

# Reinforcement learning – Evolution in silicon

## How does evoluation work?

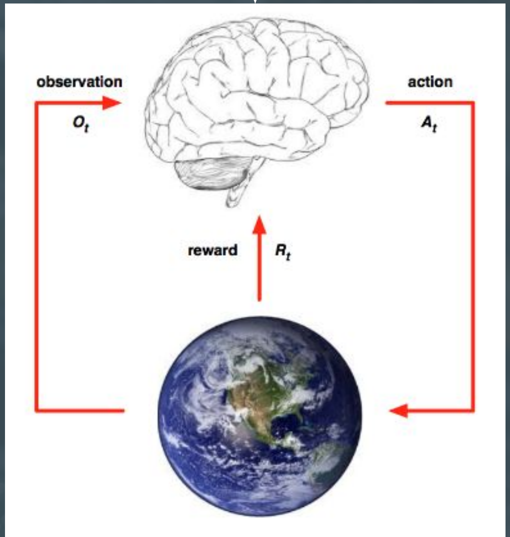
Evolution



Learning

## How does reinforcement learning work?

Development
+ tuning



Learning

*Image sources: http://www.effectiveengineer.com/blog/five-key-skills-of-successful-programmers, UCL course on RL*

# Key difference between reinforcement learning and supervised learning

Westphalia DataLab
Datenanalyse für alle

Label data

▼

Train

▼

Profit

Define world

▼

Train

▼

Profit

# EXAMPLES OF RL USAGE

RL qualifies as a solution method for different kinds of problems

## Natural Movement

This is Google's DeepMind AI teaching itself how to walk

https://www.youtube.com/watch?feature=oembed&v=gn4nRCC9TwQ

## Complex Games

https://www.youtube.com/watch?feature=oembed&v=HT-UZkiOLv8

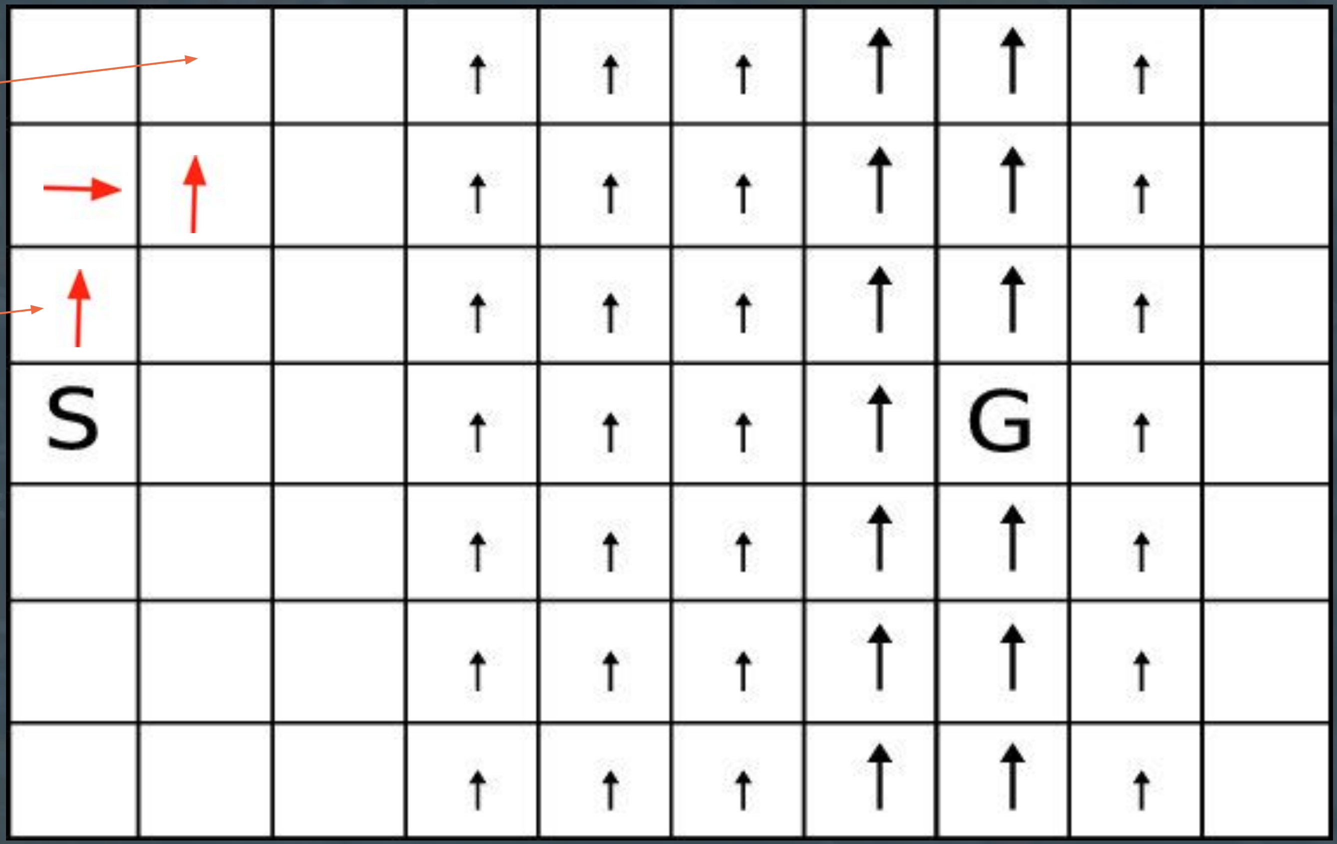
## Autonomous transportation

https://www.youtube.com/watch?feature=oembed&v=VCdxqn0fcnE

# A PRACTICAL EXAMPLE

Windy Gridworld: Reach the goal G asap starting in S

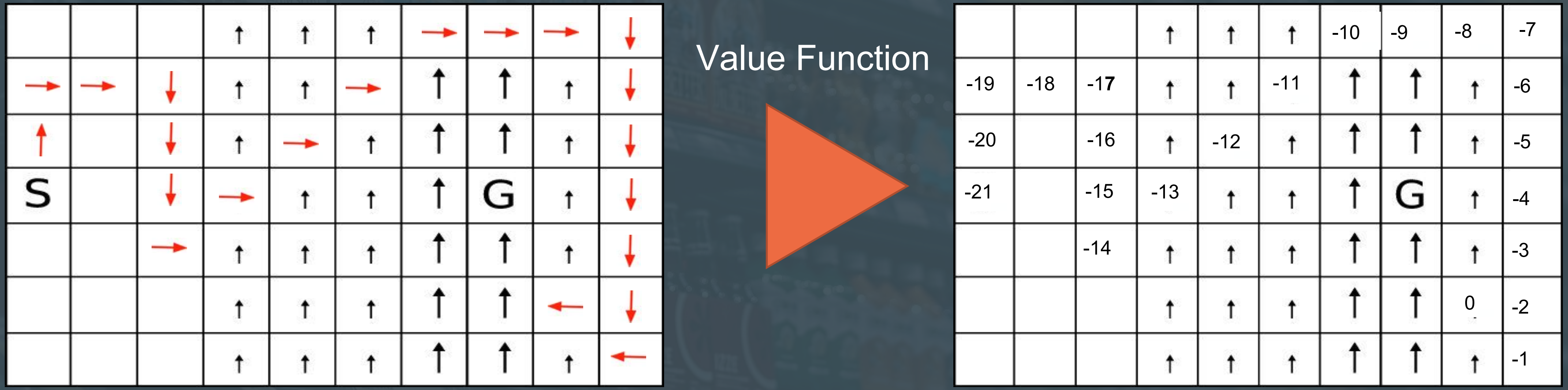
- The state is given by the actual position on the grid
- An action is given by a movement in one of the 4 directions
- Entering a field with a small arrow moves the agent one step upward, 2 steps for the big arrows
- You cannot leave the grid
- The Agent shall be punished for taking too many steps



*Reward per step = -1*

# Approach 1: Determine state values and go to the best next state

Westphalia DataLab
Datenanalyse für alle



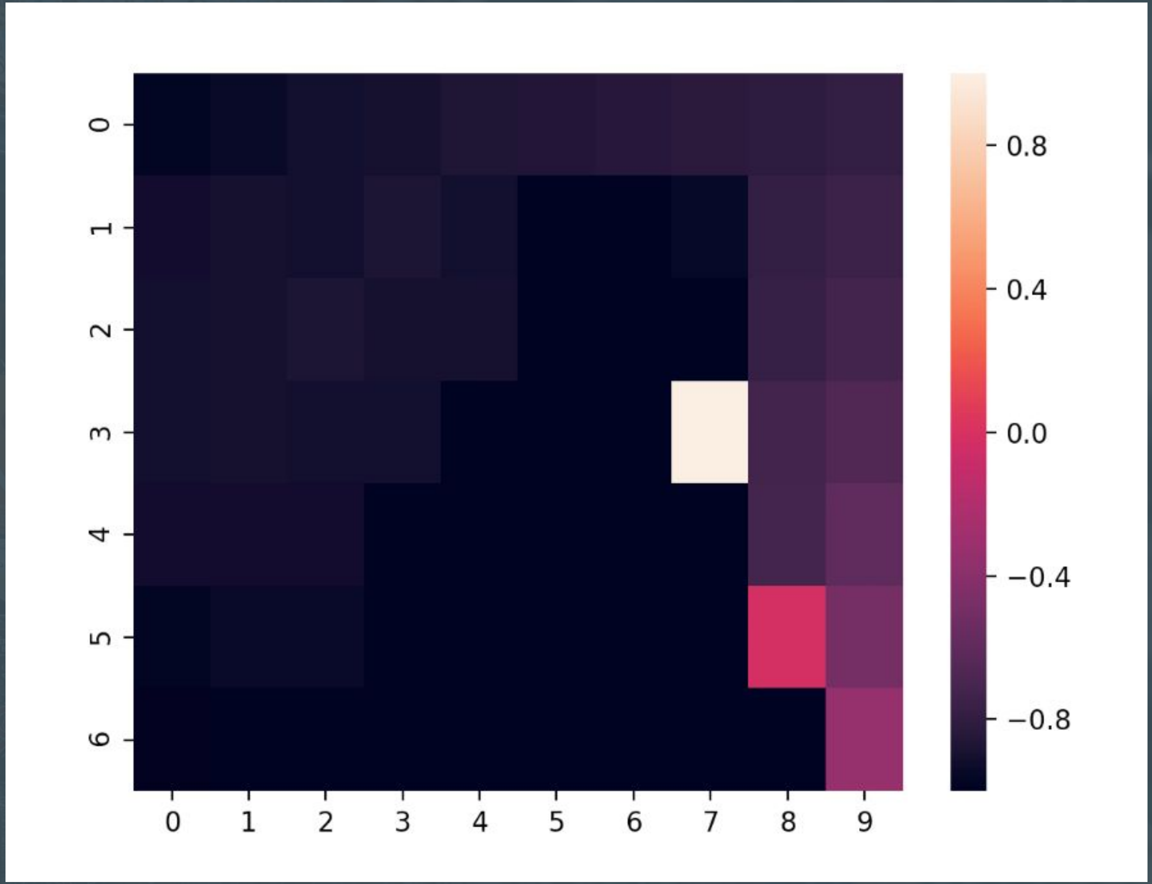
|  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  |  |  | ↑ | ↑ | ↑ | → | → | → | ↓ |
| → | → | ↓ | ↑ | ↑ | → | ↑ | ↑ | ↑ | ↓ |
| ↑ |  | ↓ | ↑ | → | ↑ | ↑ | ↑ | ↑ | ↓ |
| S |  | ↓ | → | ↑ | ↑ | ↑ | G | ↑ | ↓ |
|  |  | → | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↓ |
|  |  |  | ↑ | ↑ | ↑ | ↑ | ↑ | ← | ↓ |
|  |  |  | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ← |

Value Function

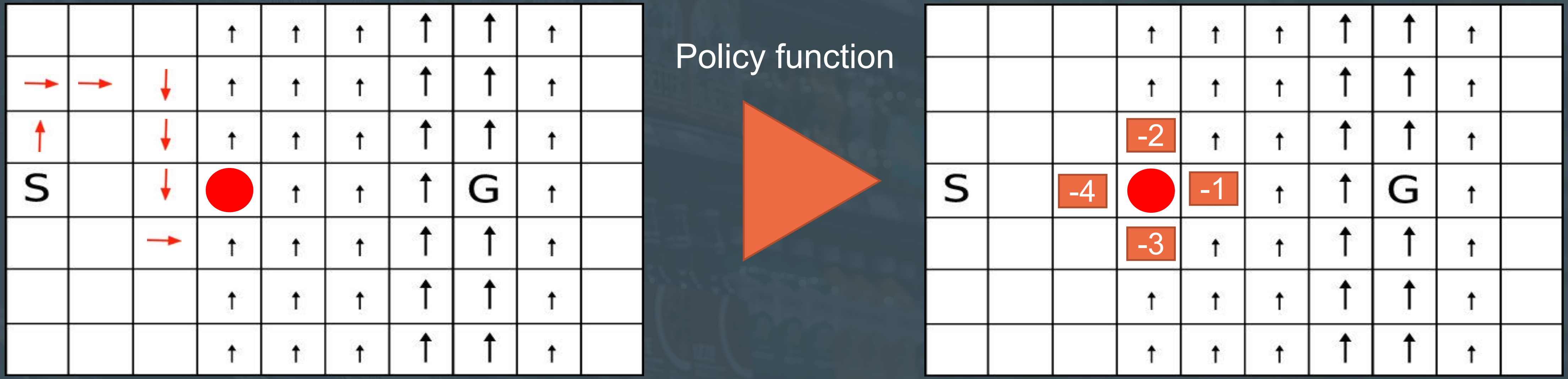
|  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  |  |  | ↑ | ↑ | ↑ | -10 | -9 | -8 | -7 |
| -19 | -18 | -17 | ↑ | ↑ | -11 | ↑ | ↑ | ↑ | -6 |
| -20 |  | -16 | ↑ | -12 | ↑ | ↑ | ↑ | ↑ | -5 |
| -21 |  | -15 | -13 | ↑ | ↑ | ↑ | G | ↑ | -4 |
|  |  | -14 | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | -3 |
|  |  |  | ↑ | ↑ | ↑ | ↑ | ↑ | 0 | -2 |
|  |  |  | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | -1 |

# Approach 1: Determine state values and go to the best next state

## State values after 1000 iterations



# Approach 2: Determine best actions (= policy) directly



# Live Demo

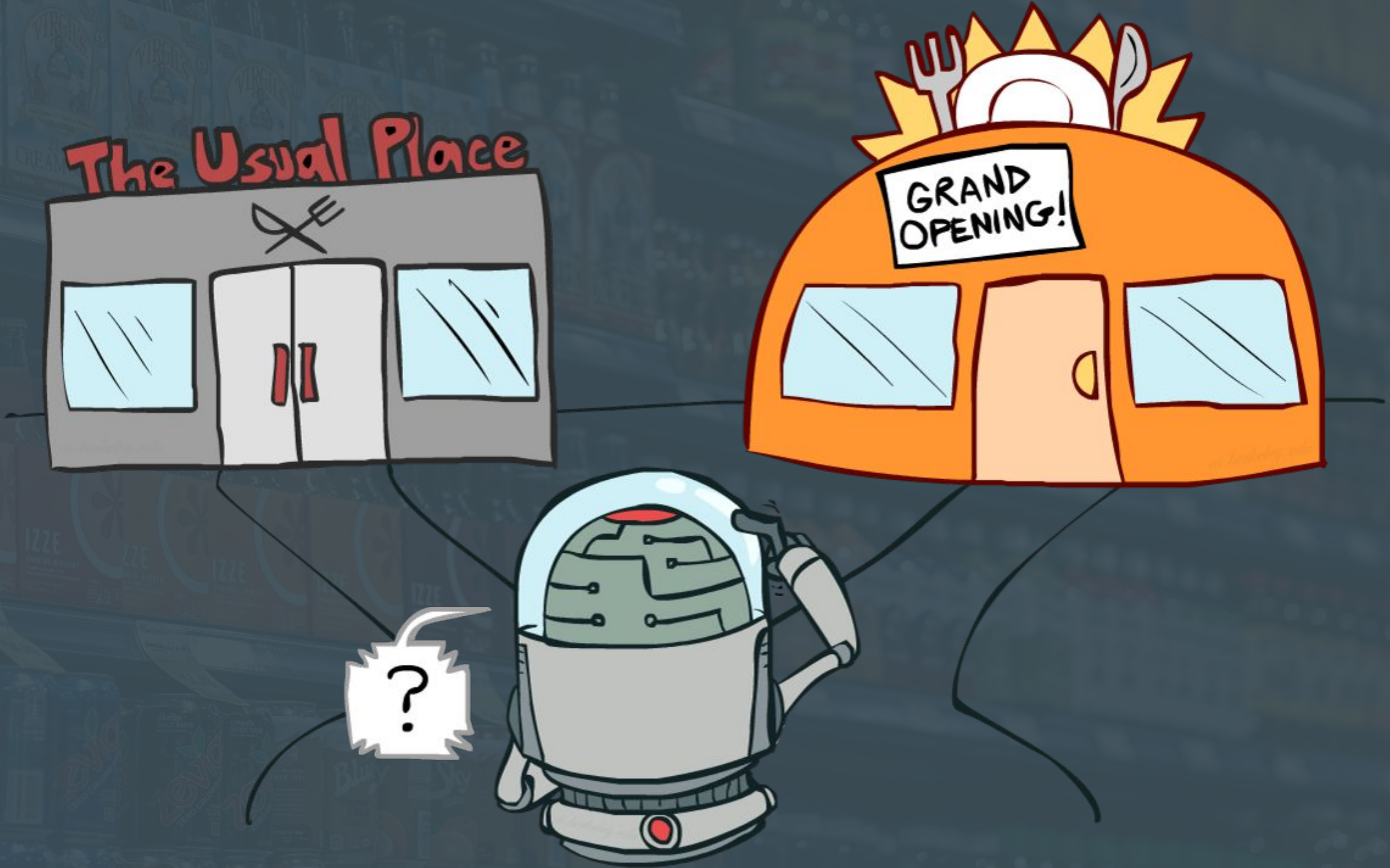
Westphalia
DataLab
Datenanalyse für alle

Try it at home!

https://github.com/MarcusCramer91/WindyGridworld

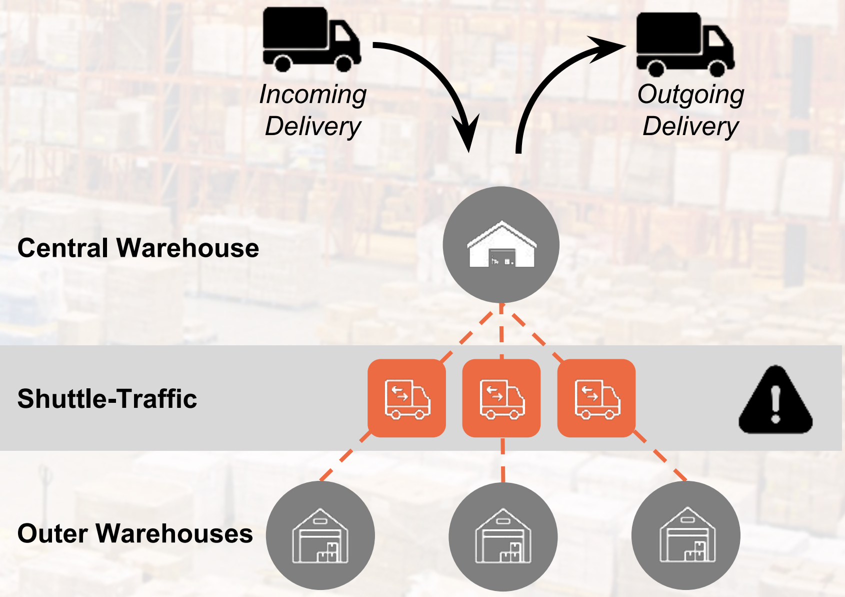
# CHALLENGES IN REINFORCEMENT LEARNING

Westphalia DataLab
Datenanalyse für alle



*Source: Google Images*

# EXAMPLE PROJECT



*Incoming Delivery*

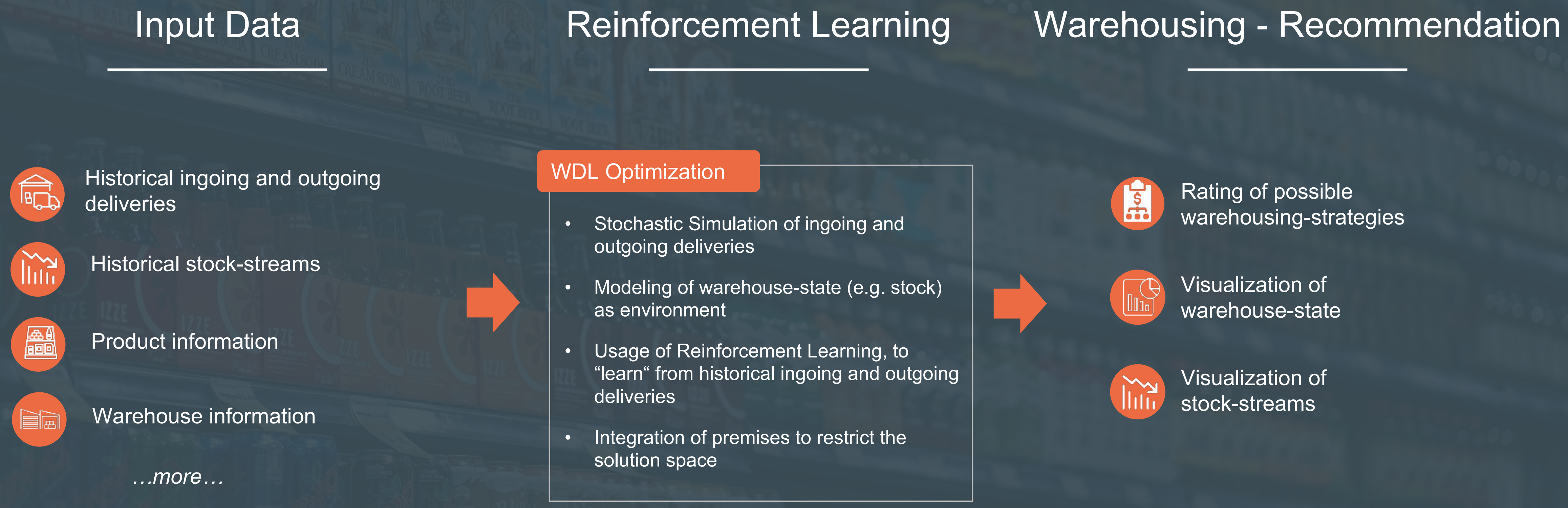
*Outgoing Delivery*

**Central Warehouse**

**Shuttle-Traffic**

**Outer Warehouses**

# PROCESS OF WAREHOUSING OPTIMIZATION



## Input Data

- Historical ingoing and outgoing deliveries
- Historical stock-streams
- Product information
- Warehouse information

*…more…*

## Reinforcement Learning

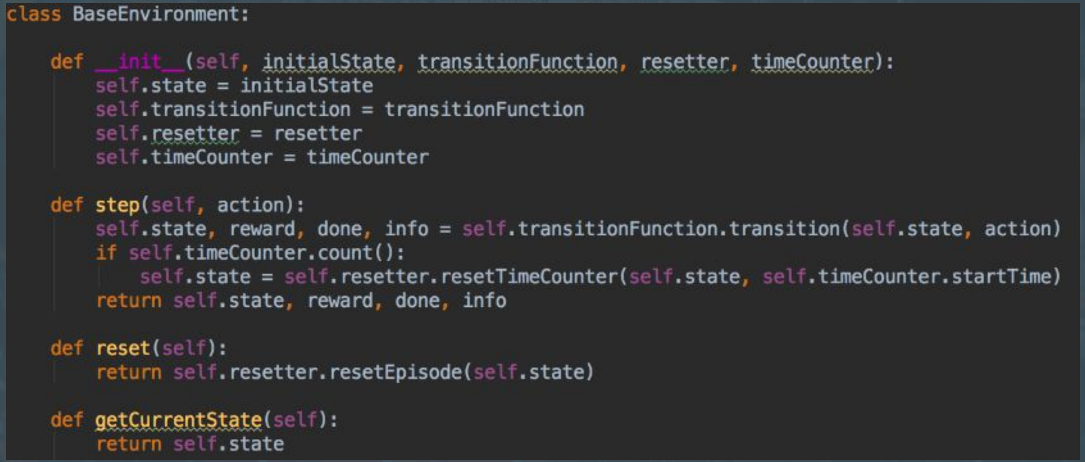
**WDL Optimization**

- Stochastic Simulation of ingoing and outgoing deliveries
- Modeling of warehouse-state (e.g. stock) as environment
- Usage of Reinforcement Learning, to “learn“ from historical ingoing and outgoing deliveries
- Integration of premises to restrict the solution space

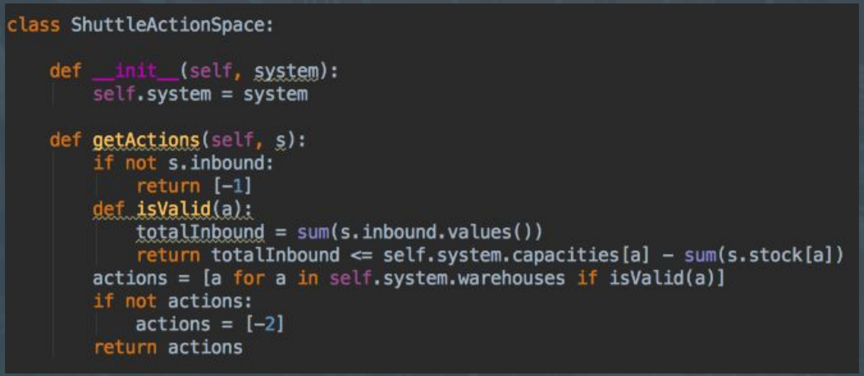
## Warehousing - Recommendation

- Rating of possible warehousing-strategies
- Visualization of warehouse-state
- Visualization of stock-streams

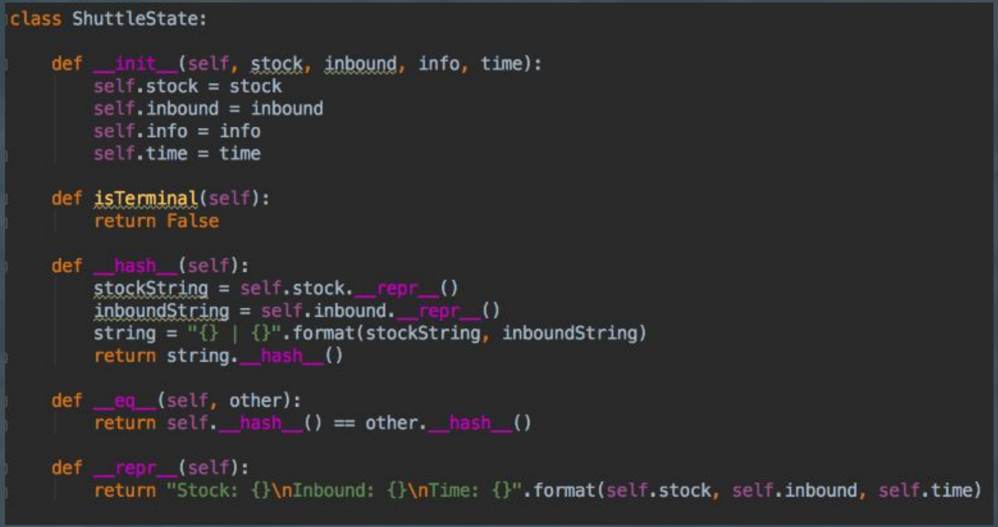
# A LITTLE BIT OF CODE…



```python
class BaseEnvironment:

    def __init__(self, initialState, transitionFunction, resetter, timeCounter):
        self.state = initialState
        self.transitionFunction = transitionFunction
        self.resetter = resetter
        self.timeCounter = timeCounter

    def step(self, action):
        self.state, reward, done, info = self.transitionFunction.transition(self.state, action)
        if self.timeCounter.count():
            self.state = self.resetter.resetTimeCounter(self.state, self.timeCounter.startTime)
        return self.state, reward, done, info

    def reset(self):
        return self.resetter.resetEpisode(self.state)

    def getCurrentState(self):
        return self.state
```



```python
class ShuttleActionSpace:

    def __init__(self, system):
        self.system = system

    def getActions(self, s):
        if not s.inbound:
            return [-1]
        def isValid(a):
            totalInbound = sum(s.inbound.values())
            return totalInbound <= self.system.capacities[a] - sum(s.stock[a])
        actions = [a for a in self.system.warehouses if isValid(a)]
        if not actions:
            actions = [-2]
        return actions
```



```python
class ShuttleState:

    def __init__(self, stock, inbound, info, time):
        self.stock = stock
        self.inbound = inbound
        self.info = info
        self.time = time

    def isTerminal(self):
        return False

    def __hash__(self):
        stockString = self.stock.__repr__()
        inboundString = self.inbound.__repr__()
        string = "{} | {}".format(stockString, inboundString)
        return string.__hash__()

    def __eq__(self, other):
        return self.__hash__() == other.__hash__()

    def __repr__(self):
        return "Stock: {}\nInbound: {}\nTime: {}".format(self.stock, self.inbound, self.time)
```

# RL CHALLENGES IN ACTION

- Huge State Space:
  The huge amount of possible states makes the problem extremely complicated

- Local Optima:
  If the problem of “Exploration vs. Exploitation“ is not solved well, the algorithm tends to stay in a local optimum

- States in general:
  A good representation of the problem in the states is essential for the algorithms success. Not having enough information hurts the algorithms knowledge, while having insufficient information increases complexity unnecessary

- Time:
  As for all projects, time is one of the most important factors, especially if you want to apply RL

# Starting points for the interested scholar

## Excellent lecture of David Silver (DeepMind) on the topic

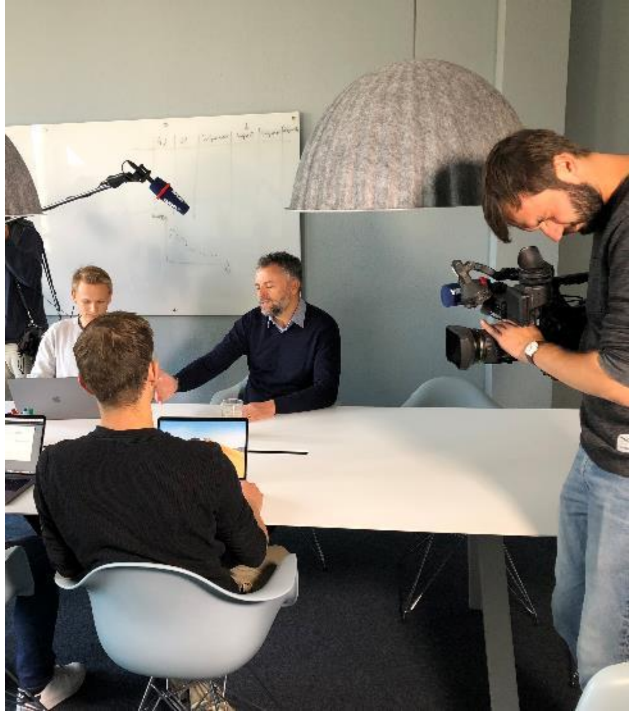
https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PL7-jPKtc4r78-wCZcQn5IqyuWhBZ8fOxT

## Library with lots of test problems (e.g. ATARI games)

https://gym.openai.com/

## Library with lots of implemented RL algorithms

https://tensorforce.readthedocs.io/en/latest/



# CONNECT WITH US @WDL!

David Middelbeck (Head of Product)
+49 151 689 698 85
middelbeck@westphalia-datalab.com

Marcus Cramer (Head of Analytics)
49 (0)151 689 698 79
cramer@westphalia-datalab.com

…OR AT:

>. TechLabs

# THE GRIDWORLD EXAMPLE

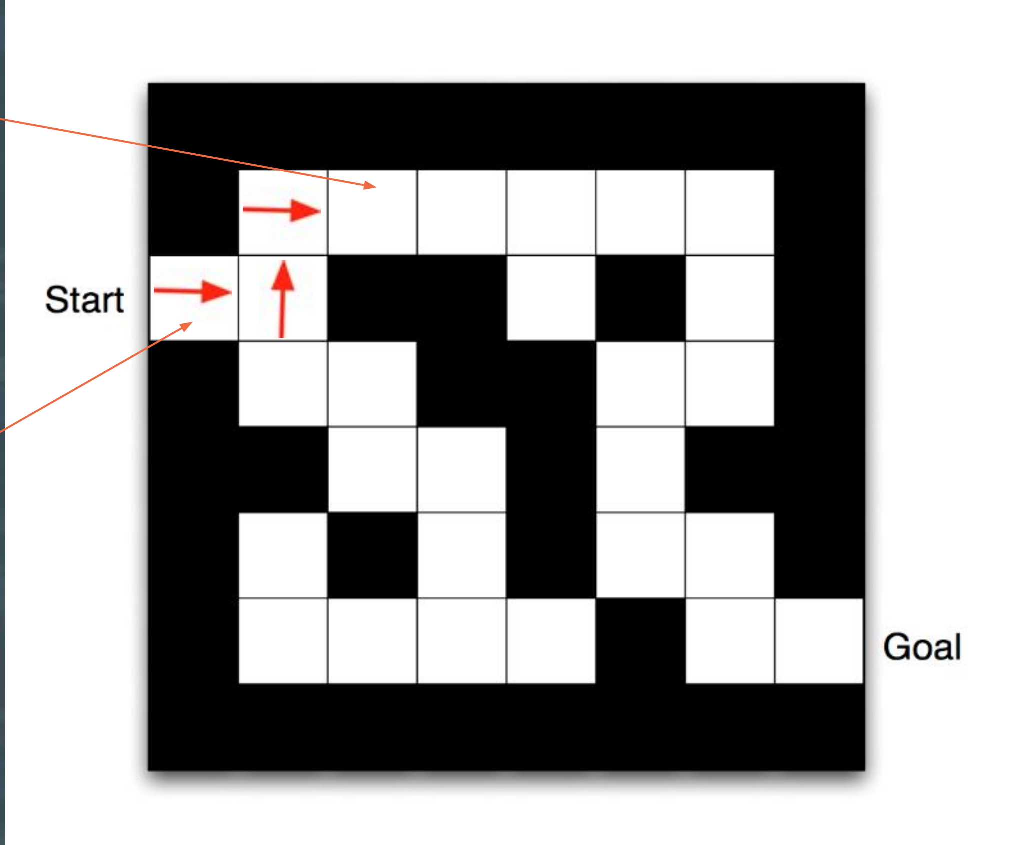
Reach the goal asap

Westphalia DataLab
Datenanalyse für alle

- The state is given by the actual position on the grid

- An action is given by a movement in one of the 4 directions

- The Agent shall be punished for taking too many steps

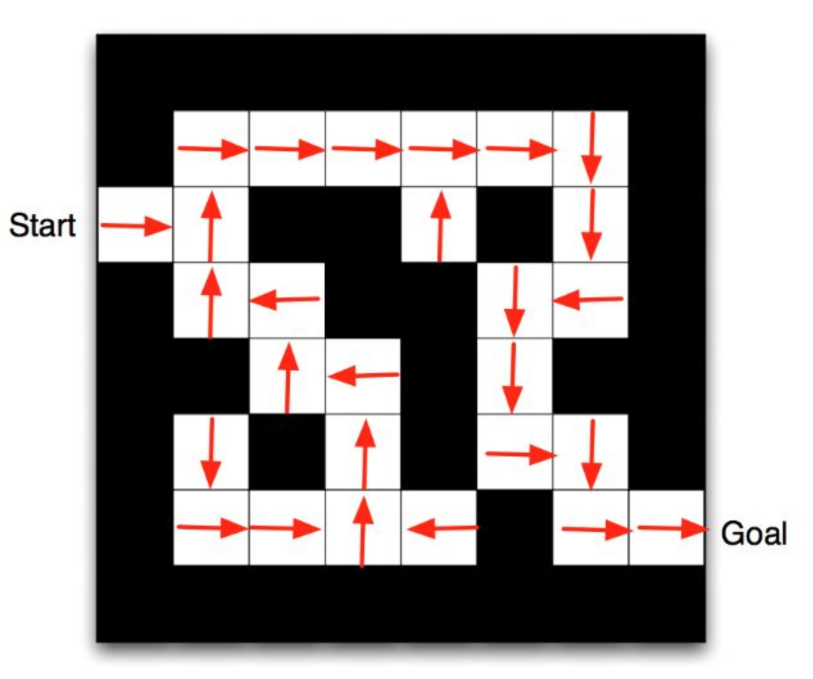


*Reward per step = -1*

*Source: UCL course on RL*

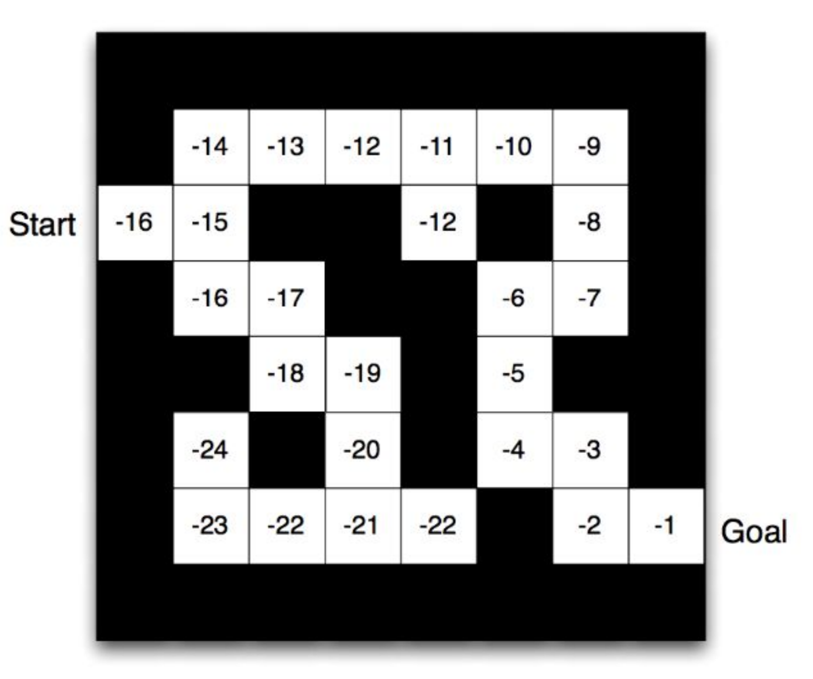
# MAJOR COMPONENTS OF THE AGENT

Policy



The optimal policy gives us the best action for each state

Value Function



|  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  |  | -14 | -13 | -12 | -11 | -10 | -9 |  |  |
| Start | -16 | -15 |  |  | -12 |  | -8 |  |  |
|  |  | -16 | -17 |  |  | -6 | -7 |  |  |
|  |  |  | -18 | -19 |  | -5 |  |  |  |
|  |  | -24 |  | -20 |  | -4 | -3 |  |  |
|  |  | -23 | -22 | -21 | -22 |  | -2 | -1 | Goal |

Expected reward for each state considering usage of the optimal policy

*Source: UCL course on RL*