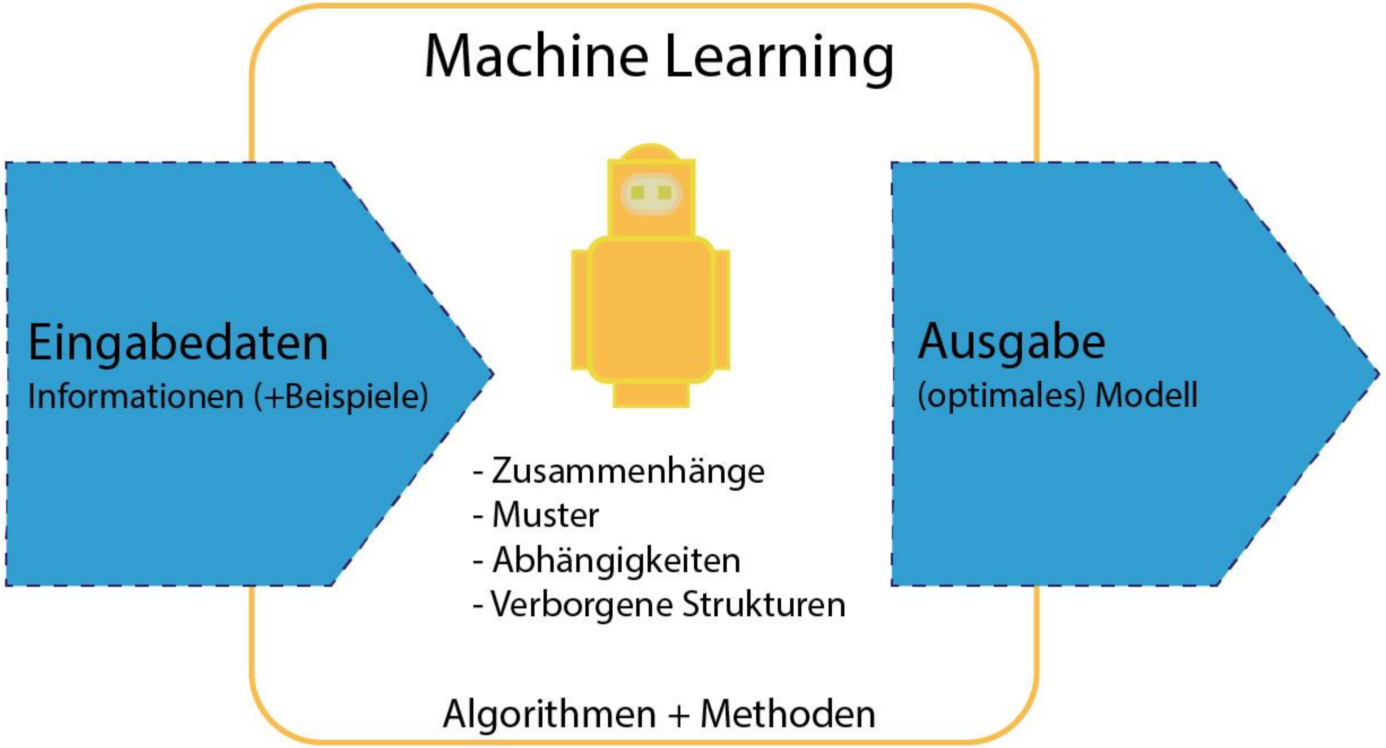
Lernverfahren im Machine Learning mit Fokus auf Reinforcement Learning

ONTIME –

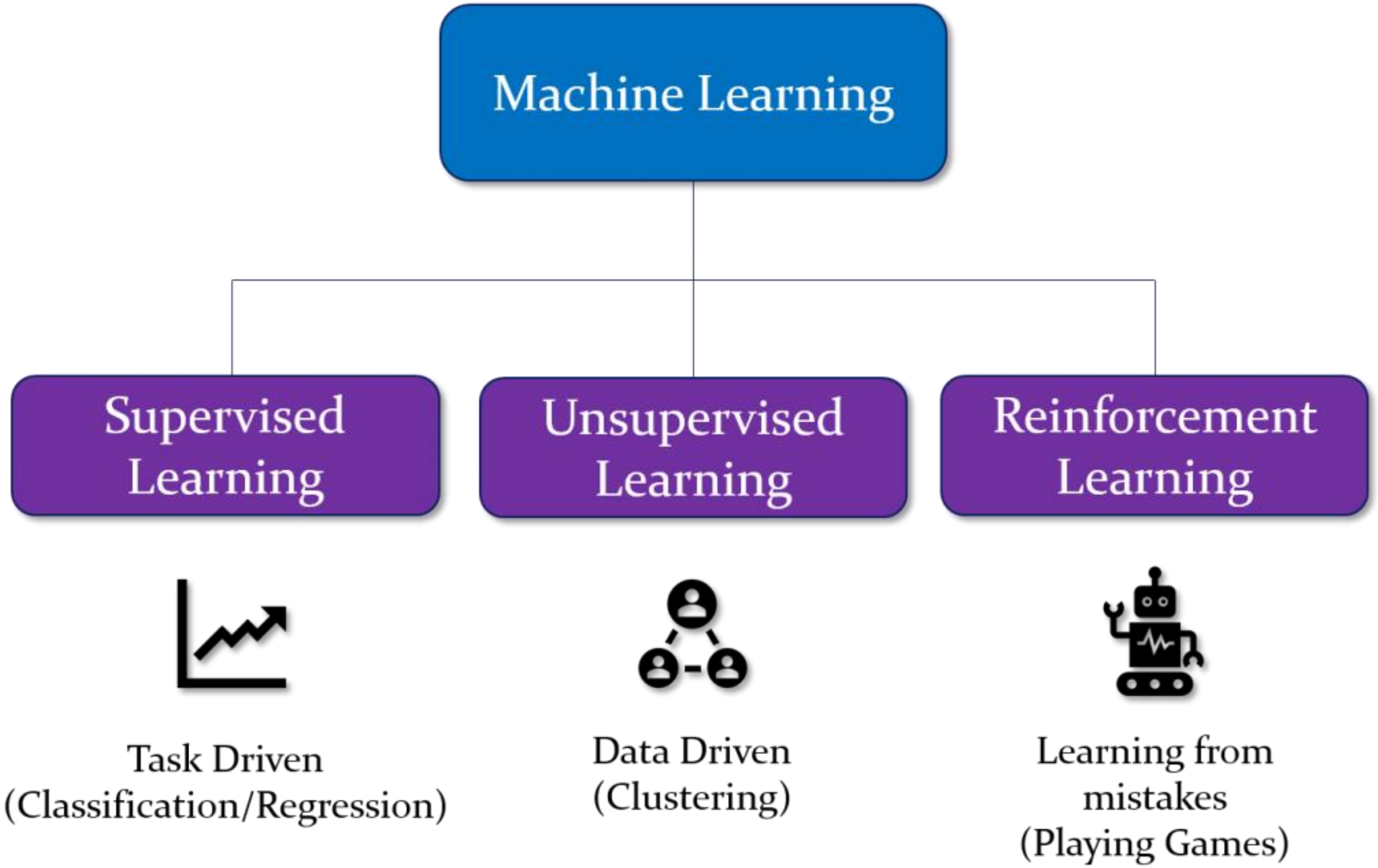
PUNCTUALITY AS A SERVICE

Inhaltsverzeichnis

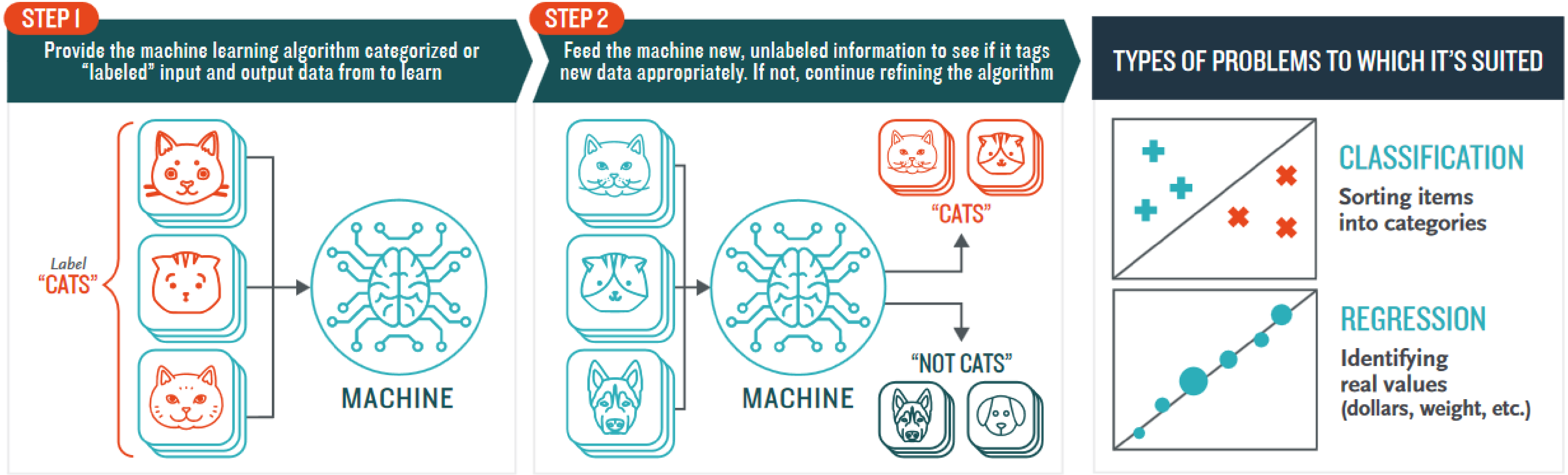
Machine Learning



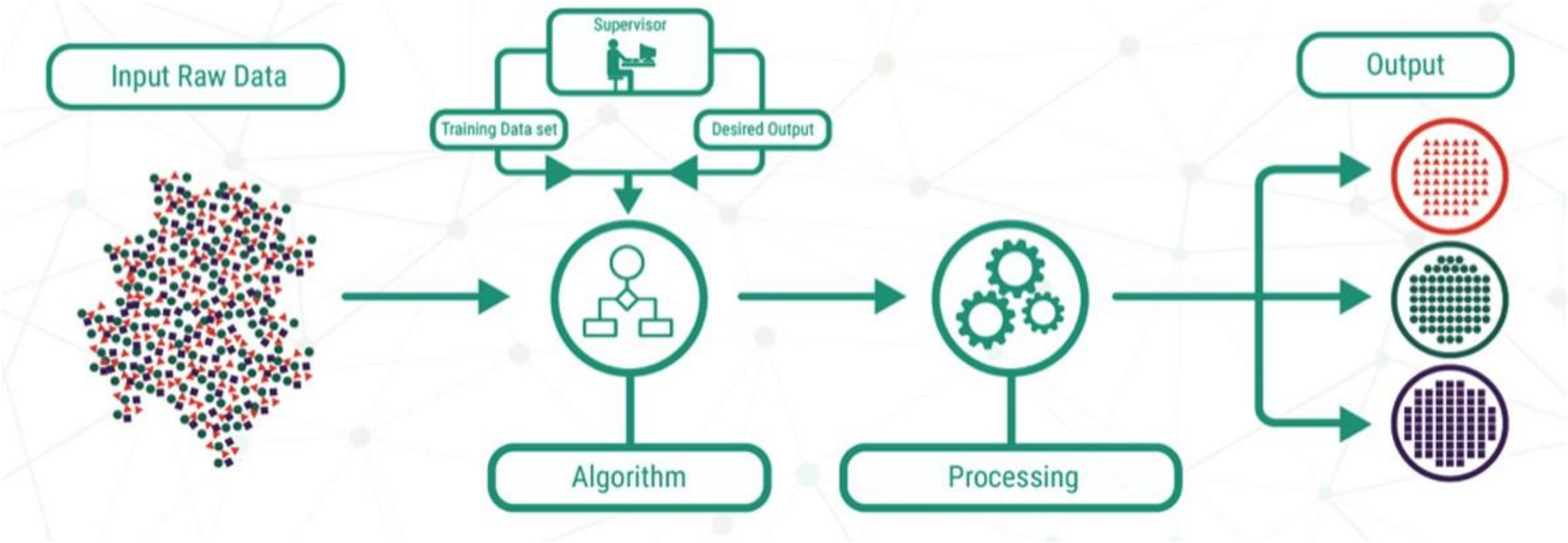
Lernverfahren



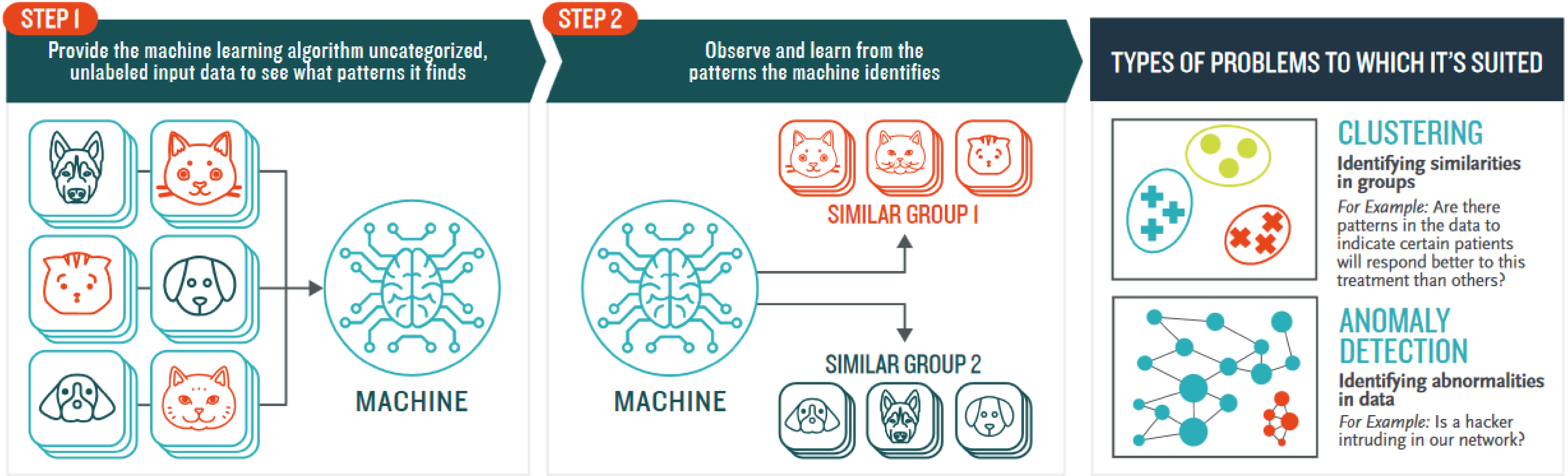
SL – Use Case


STEP 1
Provide the machine learning algorithm categorized or "labeled" input and output data from to learn
Label
"CATS"
MACHINE
STEP 2
Feed the machine new, unlabeled information to see if it tags new data appropriately. If not, continue refining the algorithm
"CATS"
MACHINE
"NOT CATS"
TYPES OF PROBLEMS TO WHICH IT'S SUITED
CLASSIFICATION
Sorting items into categories
REGRESSION
Identifying real values (dollars, weight, etc.)

SL - Verfahren

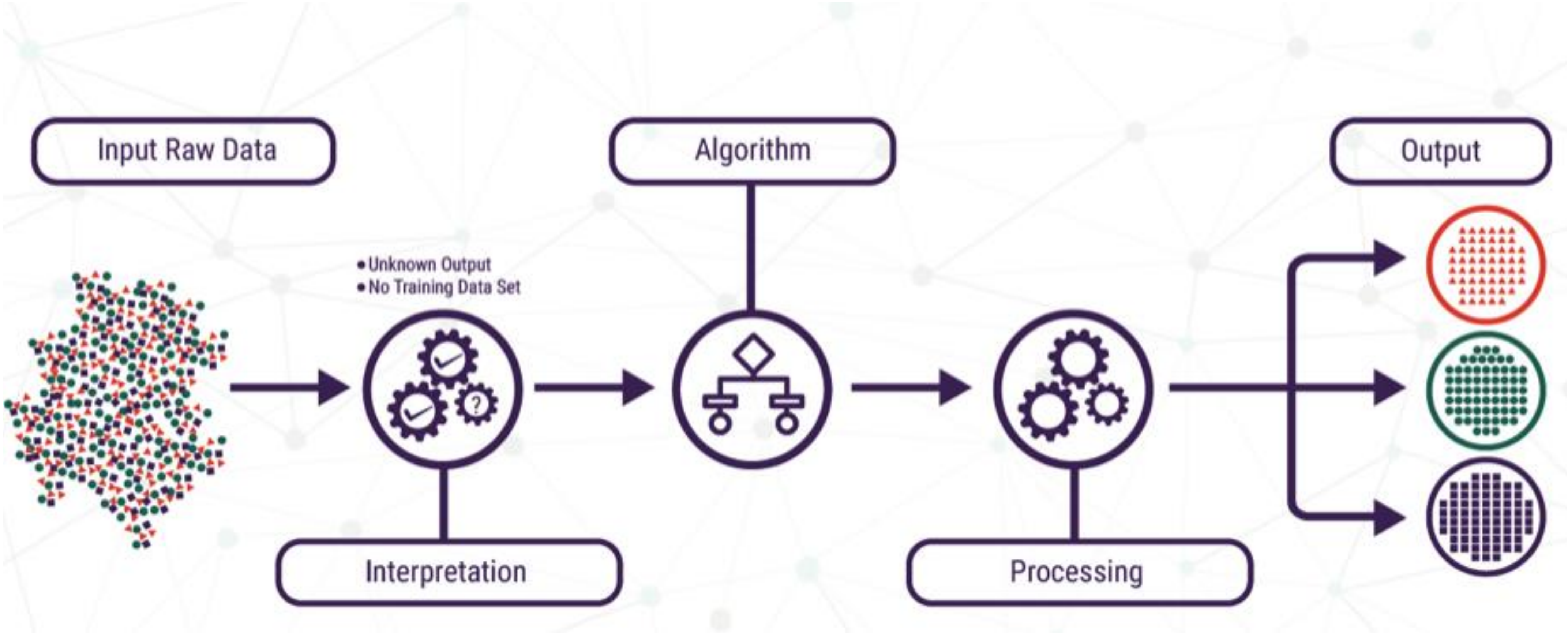


UL – Use Case

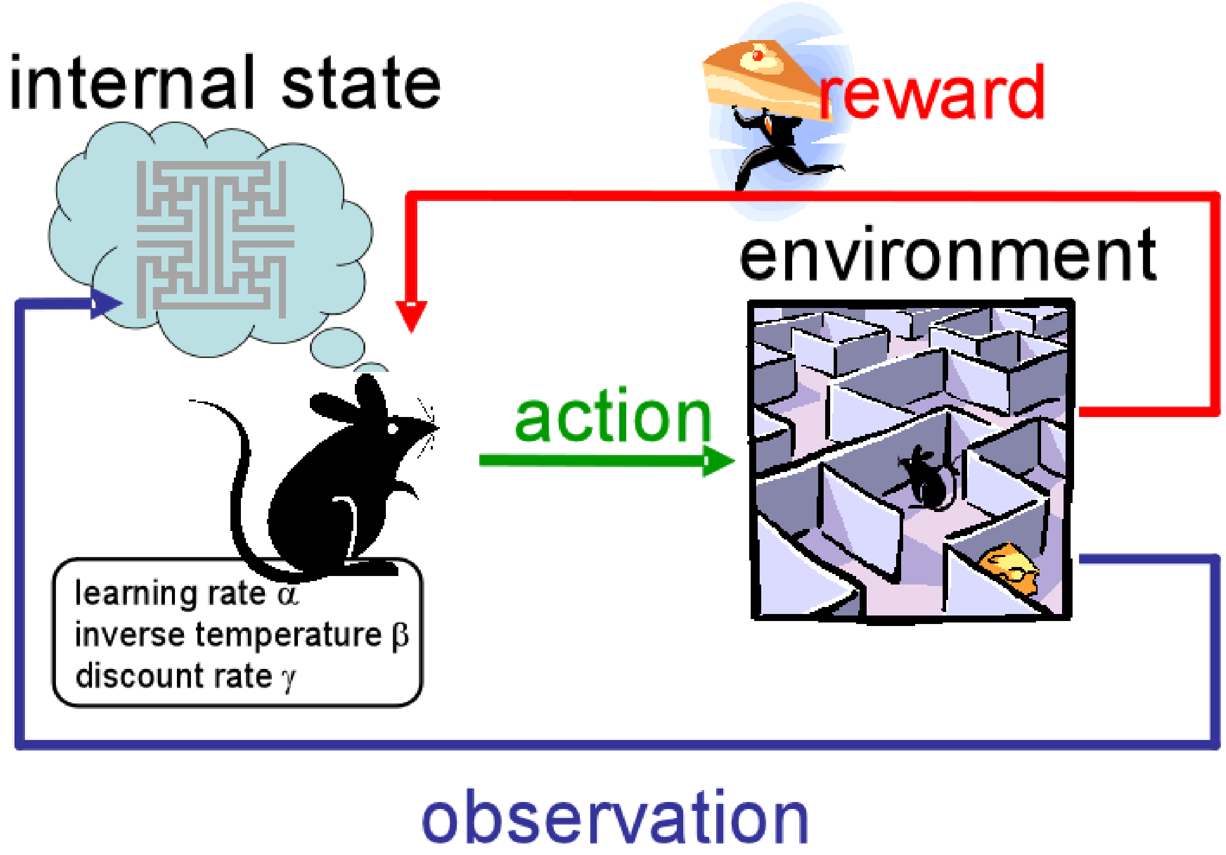


STEP 1
Provide the machine learning algorithm uncategorized, unlabeled input data to see what patterns it finds
MACHINE
STEP 2
Observe and learn from the patterns the machine identifies
SIMILAR GROUP 1
MACHINE
SIMILAR GROUP 2
TYPES OF PROBLEMS TO WHICH IT'S SUITED
CLUSTERING
Identifying similarities in groups
For Example: Are there patterns in the data to indicate certain patients will respond better to this treatment than others?
ANOMALY DETECTION
Identifying abnormalities in data
For Example: Is a hacker intruding in our network?

UL - Verfahren

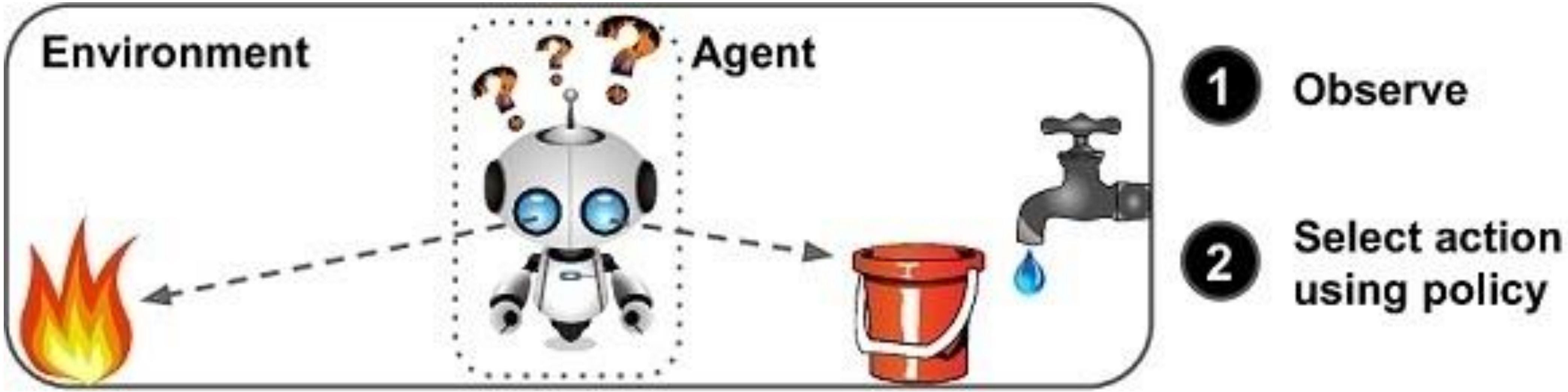


RL – Use Case

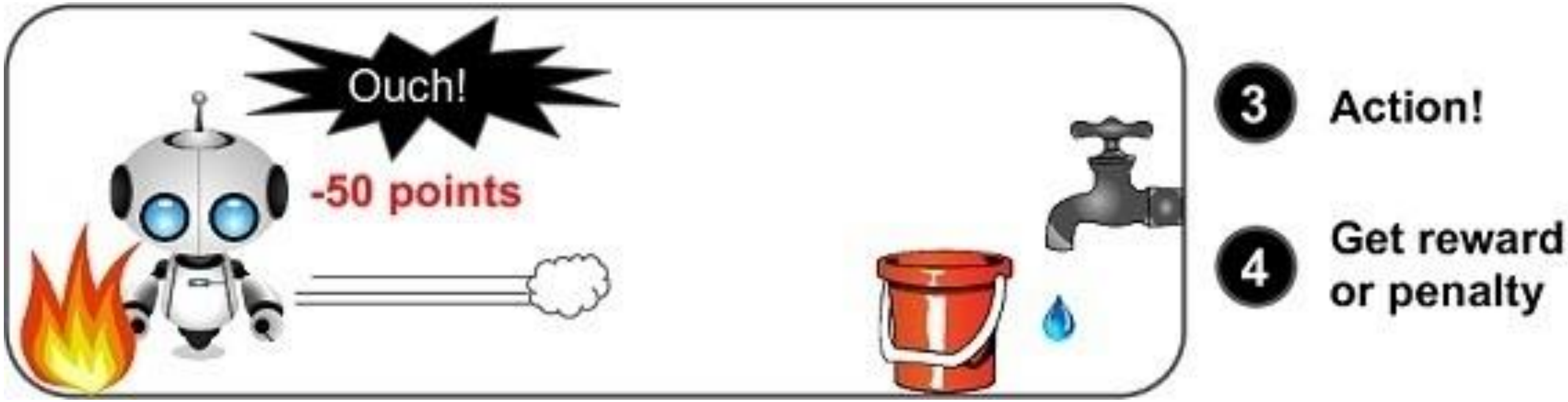

internal state
reward
environment
action
learning rate α
inverse temperature β
discount rate γ
observation

RL - Verfahren

RL - Beispiel



RL - Beispiel

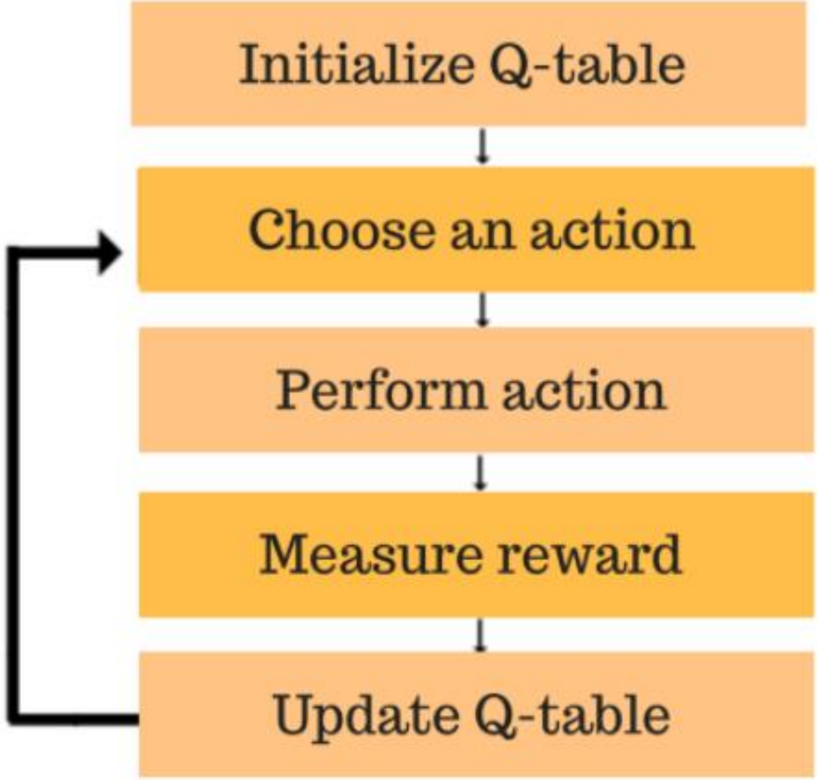


RL - Beispiel

= bad!
...
Next time avoid it.
5
Update policy (learning step)
6
Iterate until an optimal policy is found

Inhaltsverzeichnis

Q-Learning


Initialize Q-table
Choose an action
Perform action
Measure reward
Update Q-table

Q-Learning

Bellman-Gleichung:

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

Q-Learning

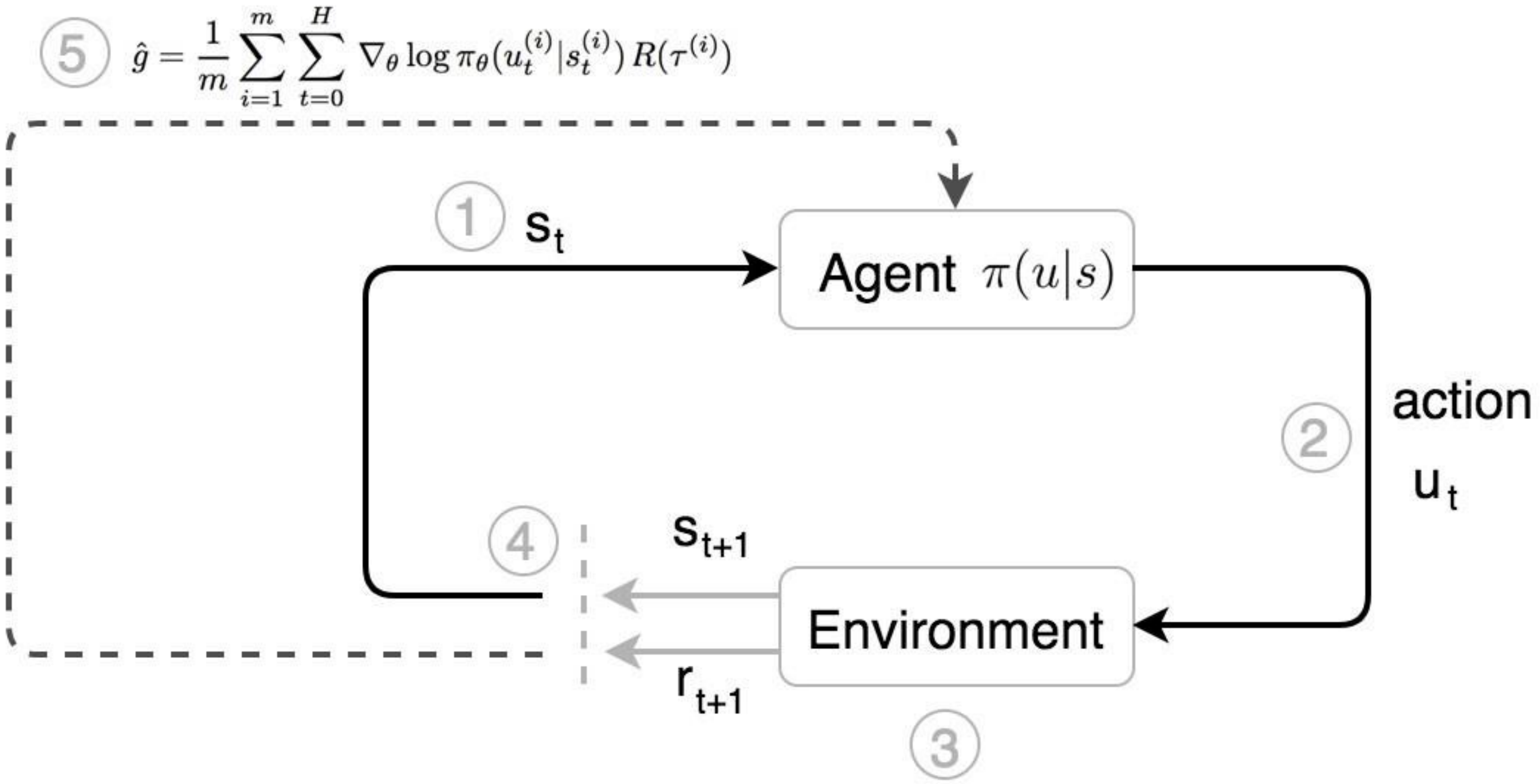
Game Board:

Current state (*s*): 0 0 0
0 1 0

Q Table: γ = 0.95

	0 0 0 1 0 0	0 0 0 0 1 0	0 0 0 0 0 1	1 0 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0
↑	0.2	0.3	1.0	-0.22	-0.3	0.0
↓	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
→	0.21	0.4	-0.3	0.5	1.0	0.0
←	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

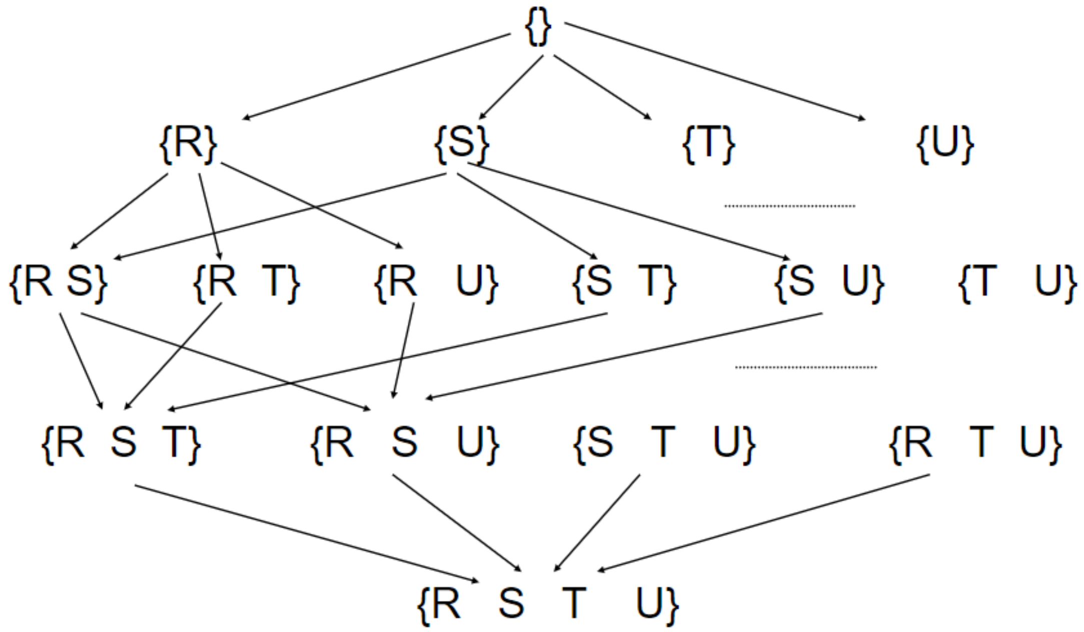
Policy Gradient



Policy Gradient

- Maximierung des erwarteten Rewards nach einer Trajektorie τ von Schritten: $J(\theta) = \mathbb{E}_{\pi}[r(\tau)]$
- Update der Parameter θ mithilfe von Gradient Descent: $\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$
- Wahrscheinlichkeit für eine Aktion wird ermittelt und in Abhängigkeit von dem Reward erhöht oder erniedrigt → Aktionen mit hoher Wahrscheinlichkeit werden infolgedessen statistisch häufiger ausgewählt als Aktionen mit niedriger Wahrscheinlichkeit
- Stochastisches Verfahren, welches sich kontinuierlich für Aktionen vorhersagt
  ← → Q-Learning: Deterministisches Verfahren, welches den erwarteten Reward für eine Aktion vorhersagt und sich diskret für den höchsten Reward entscheidet

Dynamische Programmierung



- Algorithmische Möglichkeit Optimierungsprobleme zu lösen, indem das Problem in viele gleichartige Teilprobleme aufgeteilt wird – Bedingung: Optimale Lösung des Problems setzt sich aus der optimalen Lösung der Teilprobleme zusammen

- Lösungen der kleinsten Teilprobleme werden ermittelt und abgespeichert →Ergebnisse werden einerseits für ähnliche Teilprobleme verwendet und andererseits zur Lösung des nächstgrößeren Problems → Kostspielige Rekursionen werden vermieden

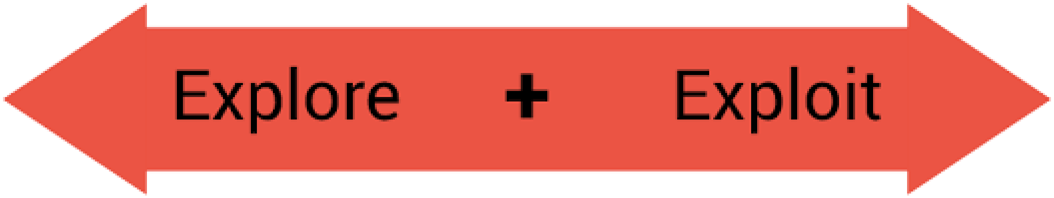
Mehrarmige Banditen Problem

- Klassisches Problem im Reinforcement Learning
- An einem k-armigen Banditen bzw. an k einarmigen Banditen sollen n Spiele gespielt werden
- Jedem Arm wird eine Zufallsvariable zugeordnet
- Ziel: Gesamtgewinn maximieren

- Annahmen:
  - Unabhängigkeit
  - Stationarität
  - Unterschiedliche Erwartungswerte
  - Gleiche Standardabweichung

Exploration-Exploitation-Dilemma

Alle Arme mehrfach ausprobieren, um zuverlässig herausfinden zu können, welcher der Beste ist

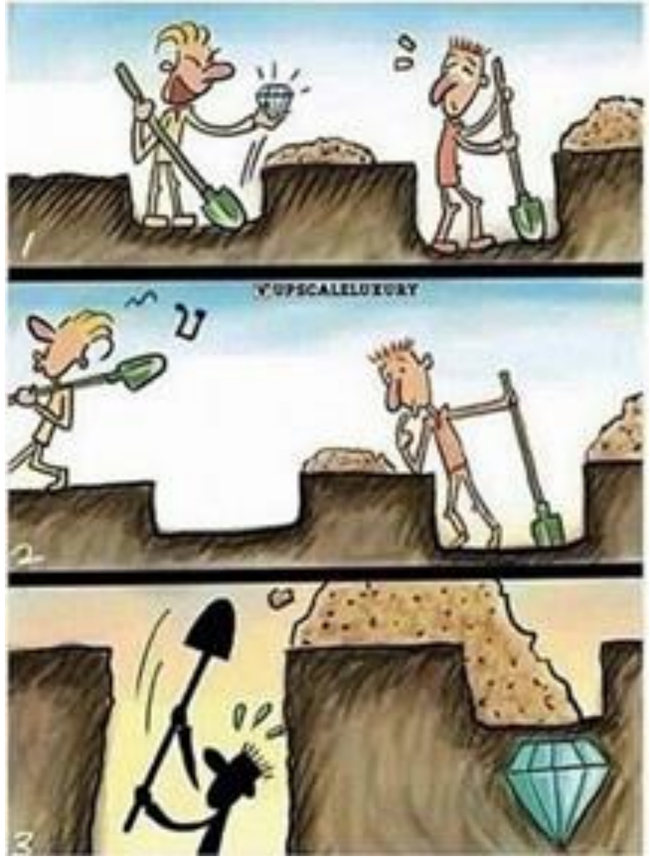


Besten Arm besonders häufig spielen, um den Gewinn zu maximieren

Probleme reiner Exploration

- Es werden viele verschiedene Wege ausprobiert, anstatt den Besten durchzuziehen
- Häufig werden auch Arme mit kleineren Erwartungswerten betätigt
- Erzielte Gewinn nähert sich immer mehr dem Mittelwert an anstatt dem Maximalgewinn

Probleme reiner Exploitation



- Schon nach wenigen Spielen wird der beste Arm ausgewählt
- Es besteht hierbei die Gefahr, dass es sich nur um den scheinbar besten Arm handelt, falls dieser zu Beginn zufälligerweise besser performt hat als der tatsächlich beste Arm
- Letztendliche Gewinn fällt kleiner aus als der maximal mögliche Gewinne

Bestandteile der Algorithmen

1. **Initialisierung**: Simulation, Zufallsvariablen und Datenelemente müssen vorbereitet werden, sodass eine Spieldurchführung, Speicherung der Ergebnisse und Berechnung der nächsten Entscheidungen möglich ist
2. **Schleife über n Spiele:**
   1. **Selektion:** Auswahl des Armes, der als nächstes gespielt wird
   2. **Spiel ausführen:** Spieldurchführung am ausgewählten Arm mithilfe eines Zufallsgenerators mit der Zufallsvariable X_i sowie der Speicherung der Ergebnisse
   3. **Update:** Berechnung neuer Größen auf Basis des neuen Ergebnisses, damit für das nächste Spiel wieder eine neue Entscheidung in der Selektion getroffen werden kann
3. **Aufbereitung und Ausgabe der Ergebnisse**

Simulation

- 9 – armiger Bandit (k=9)

- Jeder Arm hat eine andere Zufallsvariable

- Höchster Erwartungswert: 1,6 – Arm 9

- 2. Höchster Erwartungswert: 1,4 – Arm 8

- Mittlerer Erwartungswert über alle Arme: 0,8

- Anzahl an Spiele: 2700 (n=2700)

- **Algorithmen:** Random-, Greedy-, ε-First-, ε-Greedy, ε-Decreasing-Algorithmus

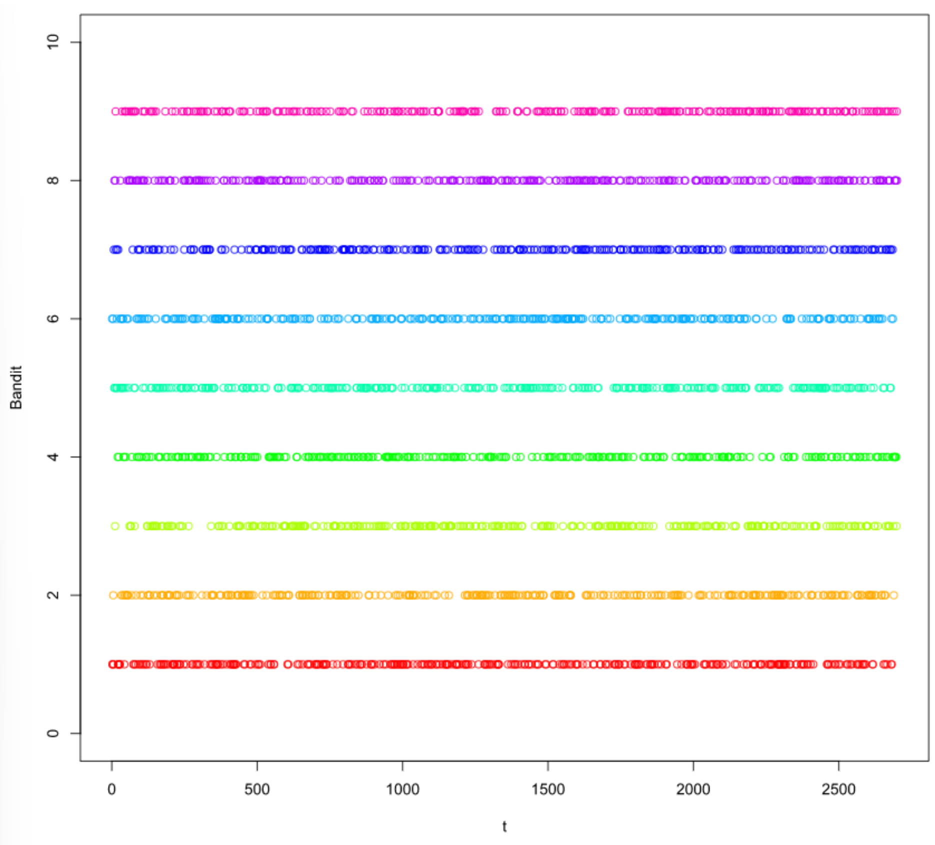
Random-Algorithmus

- Komplet zufällige Auswahl des nächsten Armes
- Nur Untersuchung der einzelnen Arme
- Keine Nutzung des gewonnen Wissens aus der Untersuchung der Arme

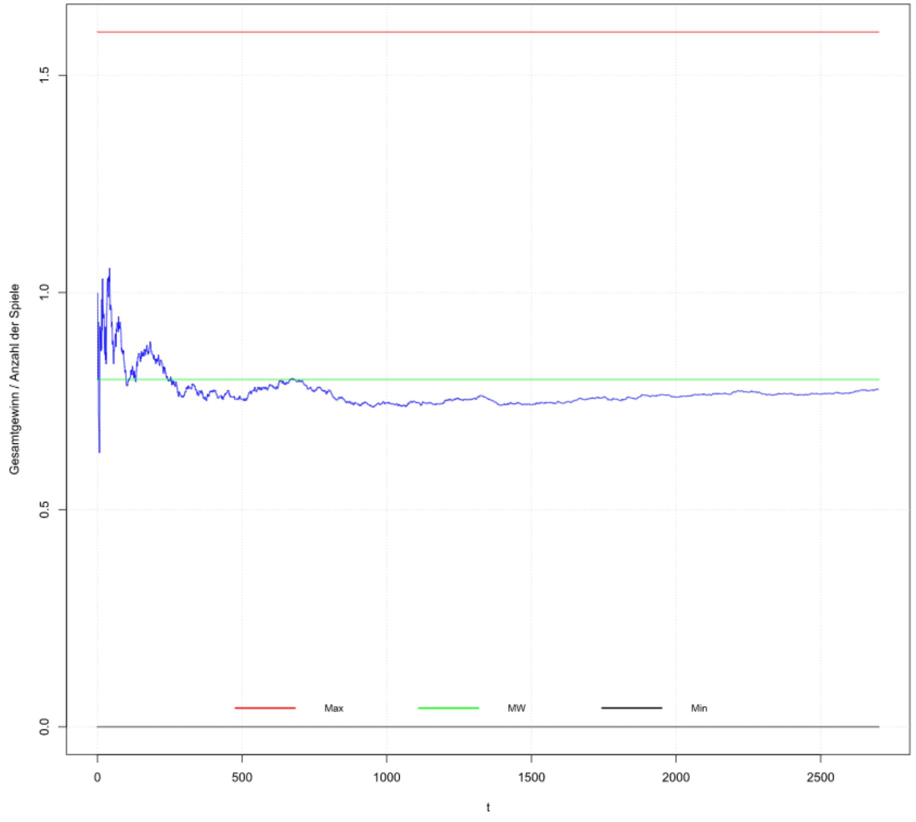
→ Reine Exploration

Random-Algorithmus

Nutzung der Banditenarme



Entwicklung des Gesamtgewinnes



Greedy-Algorithmus

- Initial wird jeder Arm einmal untersucht

- Erzielte Mittelwert für jeden Arm berechnet

- Arm mit dem höchsten Mittelwert (höchste Gewinnwahrscheinlichkeit) wird ab jetzt immer ausgewählt

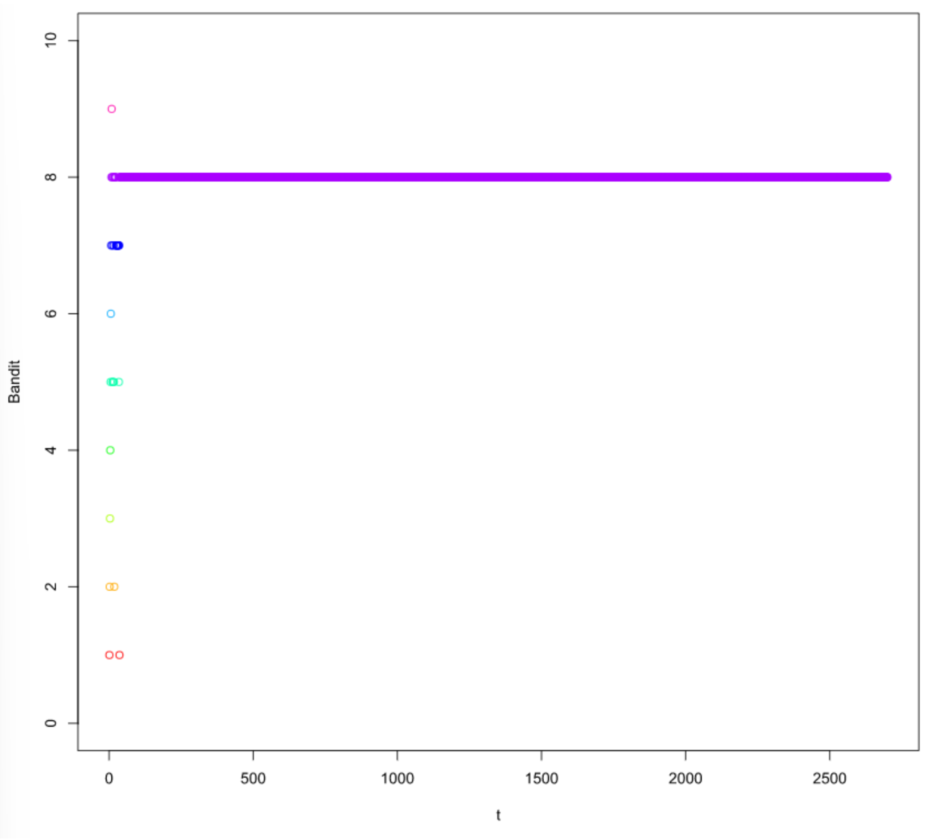
- Mittelwerte werden bei jedem Spiel aktualisiert

- Arm wird ausschließlich dann gewechselt, wenn sein Mittelwert unter den Mittelwert eines anderen Armes rutscht

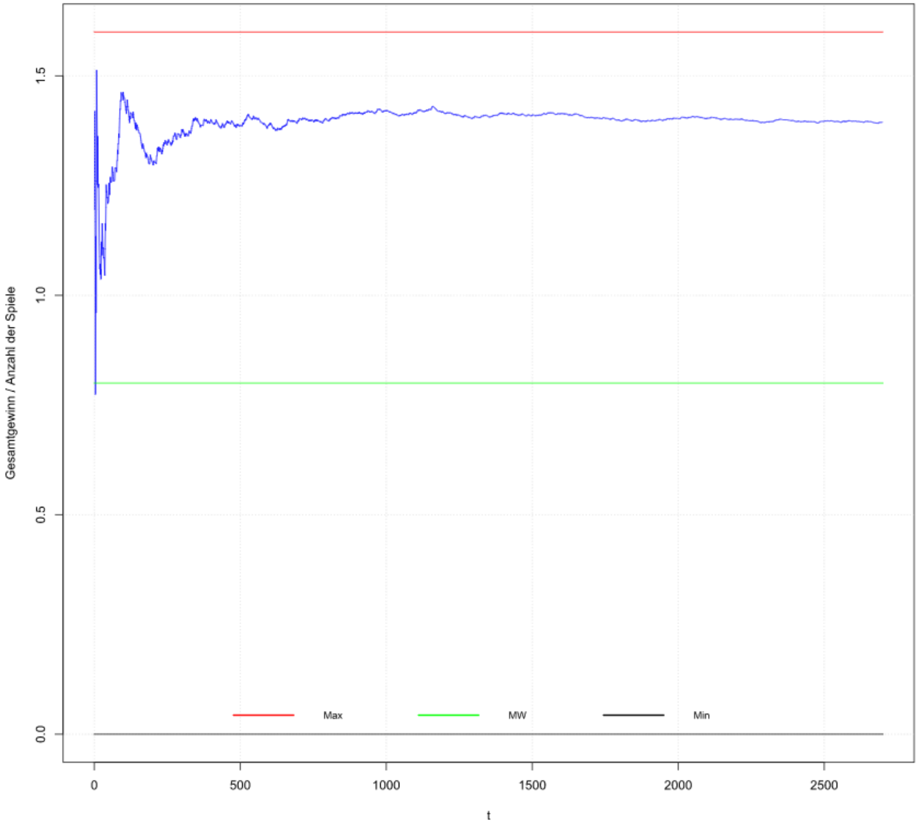
→ Reine Exploitation

Greedy-Algorithmus

Nutzung der Banditenarme



Entwicklung des Gesamtgewinnes



ε-First-Algorithmus

- ein ε zwischen 0 und 1 wird definiert, z.B. ε = 0,1

- die ersten ε*n Spiele werden nach dem Random-Algorithmus (Exploration) durchgeführt

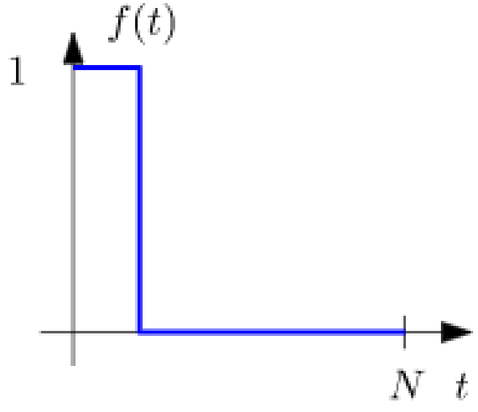
- die folgenden (1- ε)*n Spiele werden nach dem Greedy-Algorithmus (Exploitation) durchgeführt

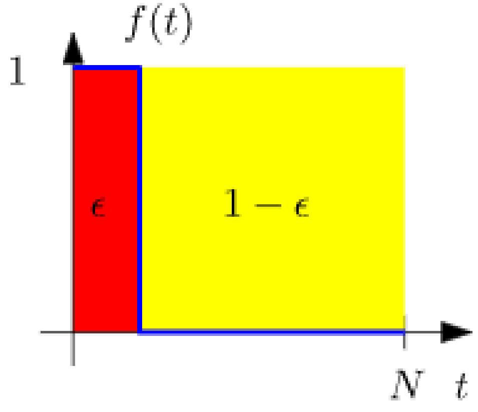
→**Trade-Off zwischen Exploration und Exploitation**

- Grenzfälle:
  - ε=0 – Reine Exploitation
  - ε=1 – Reine Exploration

- Problem: Welche Größe für ε?

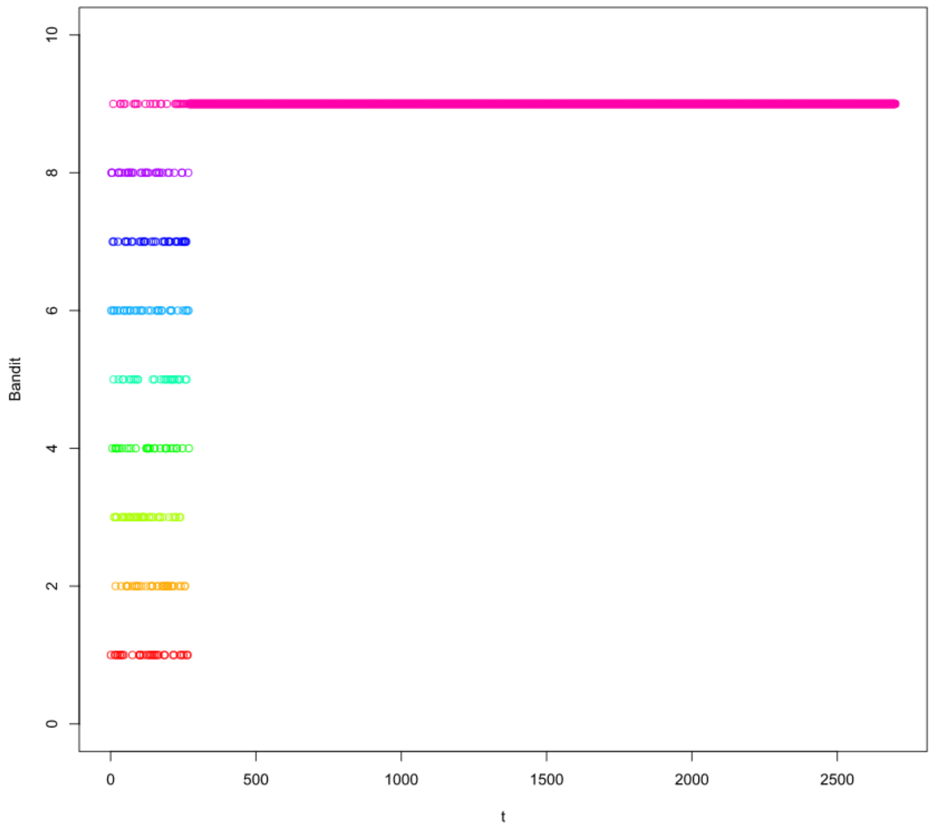
- Simulation: ε = 0,1



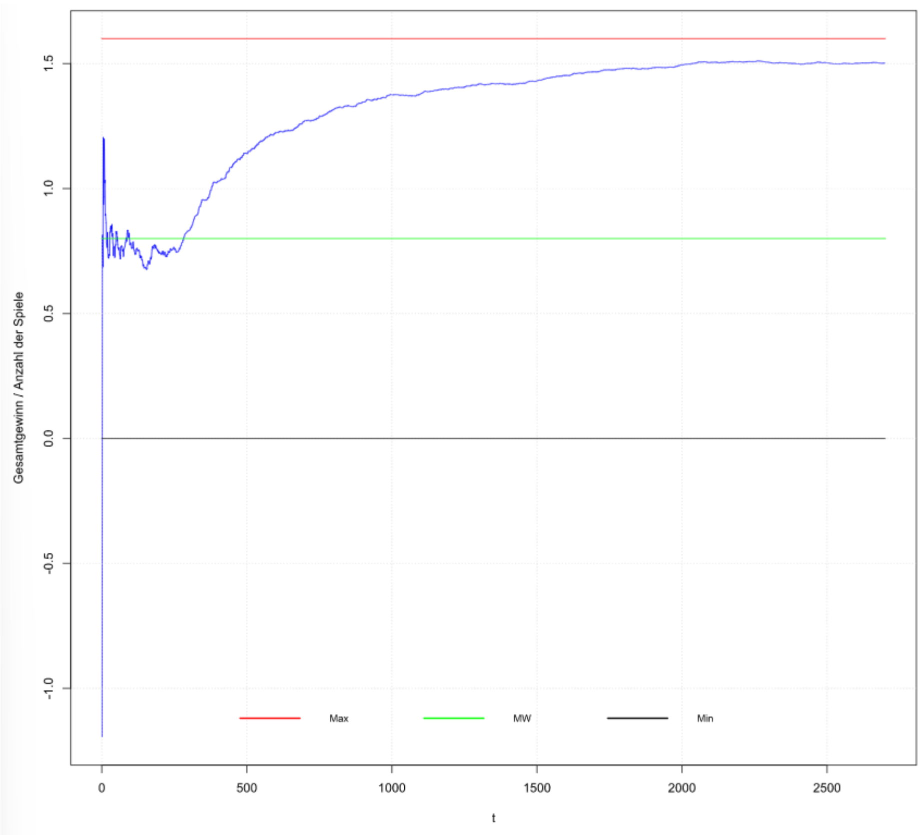


ε-First-Algorithmus

Nutzung der Banditenarme



Entwicklung des Gesamtgewinnes



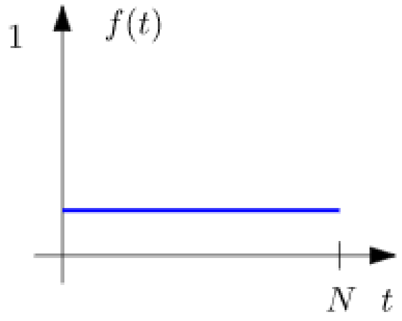
ε-Greedy-Algorithmus

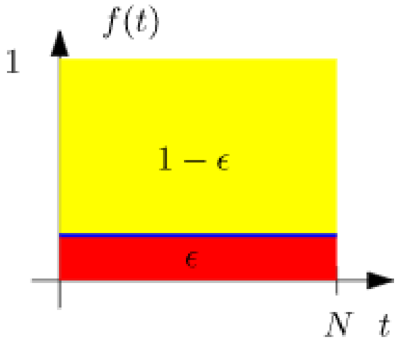
- ε entscheidet bei jedem Spiel aufs neue, ob eine Exploration oder eine Exposition durchgeführt wird mit der entsprechenden Wahrscheinlichkeit

- sinnvoll bei Nicht-stationären Problemen, bei denen sich die Erfolgswahrscheinlichkeit ändern kann → passt sich leichter an zeitliche Veränderungen an als der ε-First-Algorithmus

- Nachteil: Algorithmus führt auch noch sehr spät Explorationsphasen durch, die eigentlich unsinnig sind

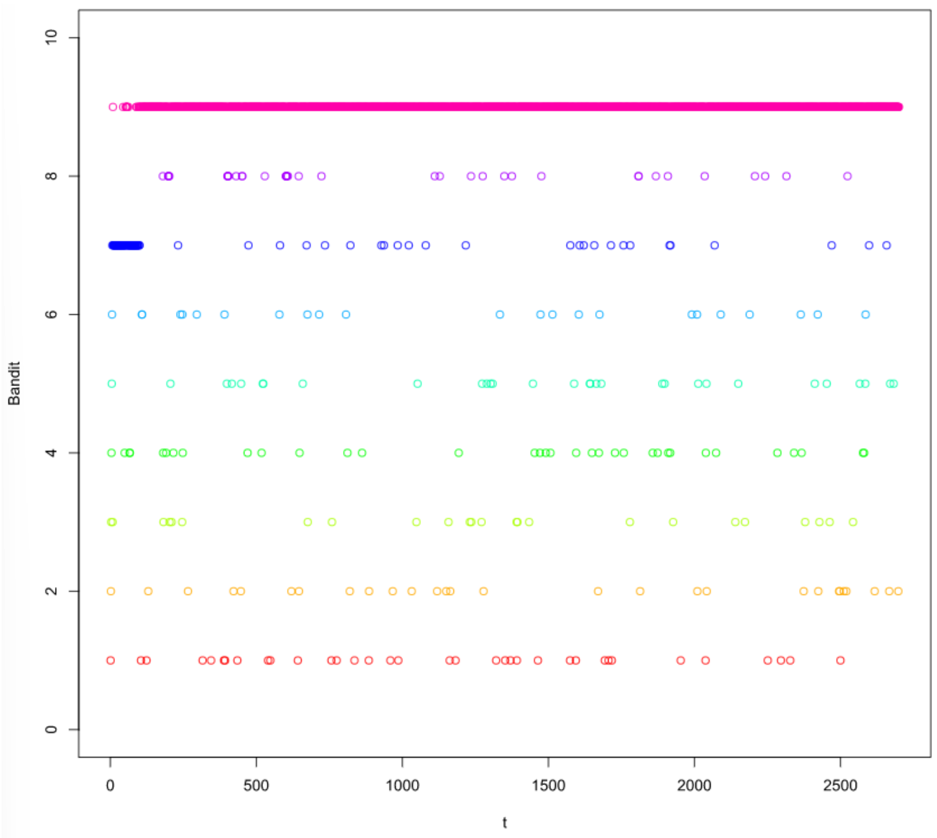
→ Trade-Off zwischen Exploration und Exposition



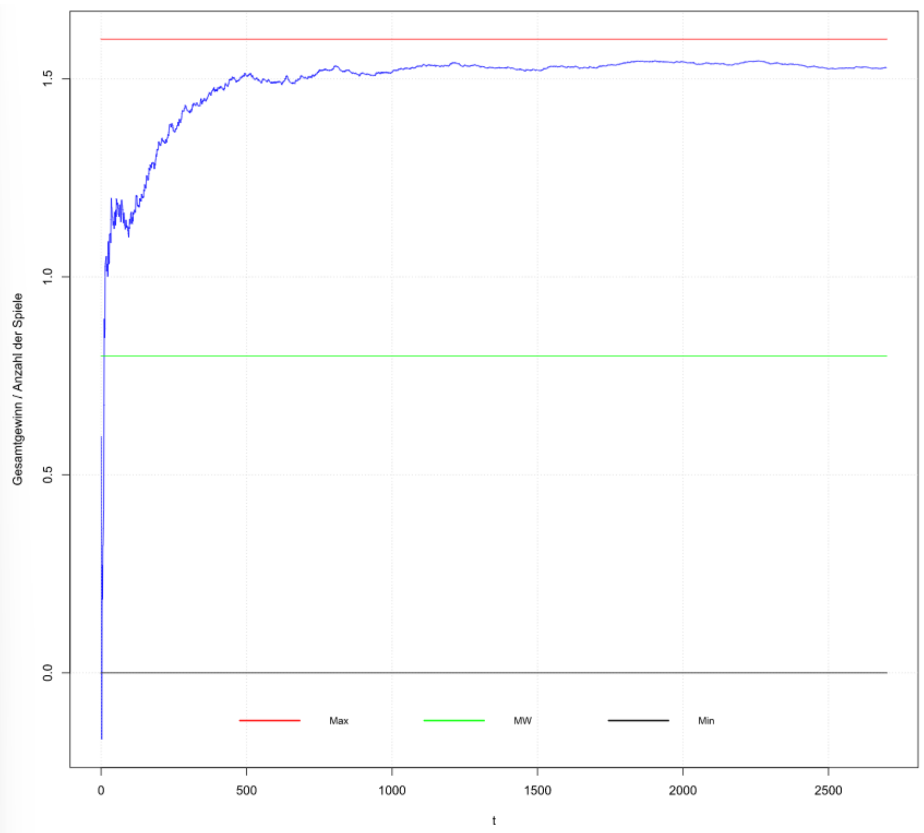


ε-Greedy-Algorithmus

Nutzung der Banditenarme



Entwicklung des Gesamtgewinnes



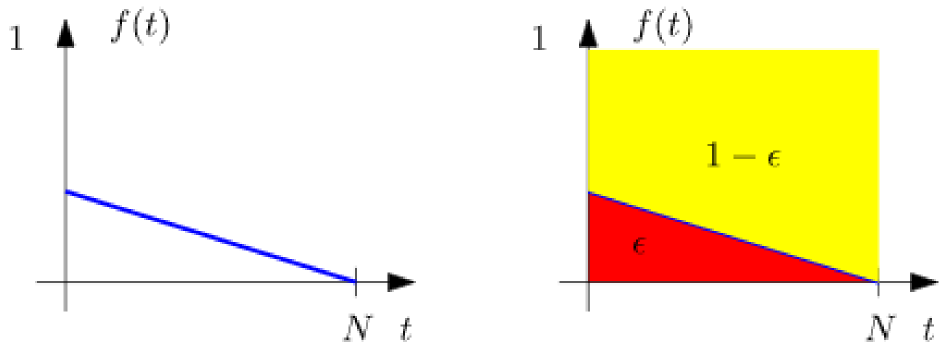
ε-Decreasing-Algorithmus

- Kombination aus ε-First- und ε-Greedy-Algorithmus

- eine monoton fallende Funktion wird definiert, die angibt, dass zu Beginn viel exploriert wird, während zum Ende hin immer weniger exploriert wird

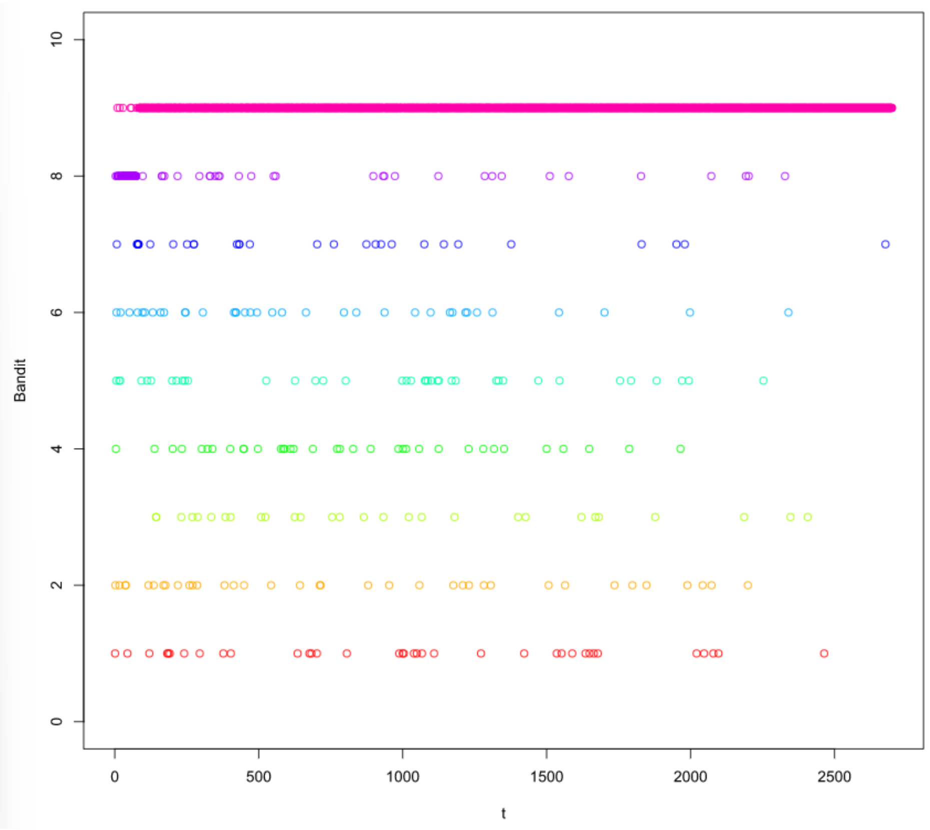
- ε gibt den übergreifenden Anteil der Explorationsspiele im Vergleich zu den Expositionsspielen an

→Trade-Off zwischen Exploration und Exploitation

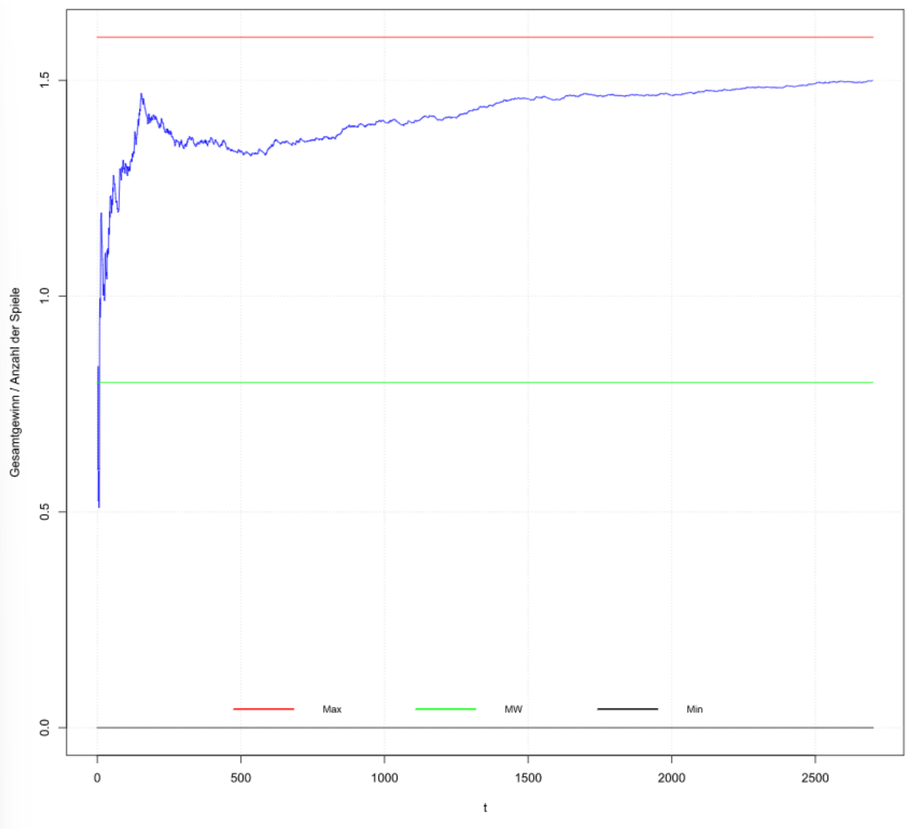


ε-Decreasing-Algorithmus

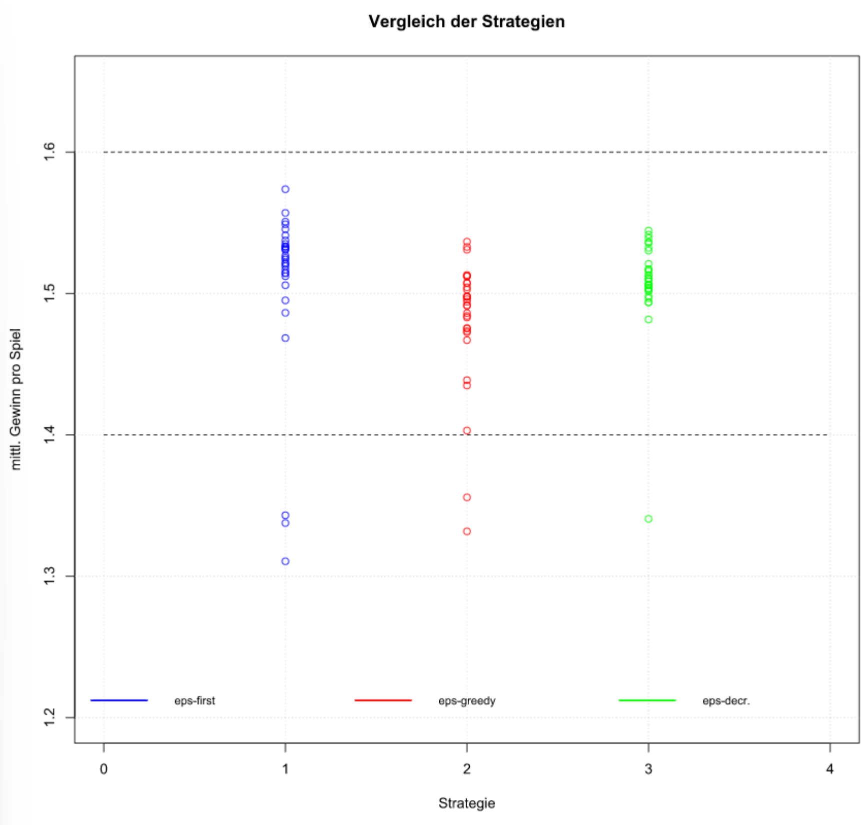
Nutzung der Banditenarme



Entwicklung des Gesamtgewinnes



Vergleich der ε-Algorithmen



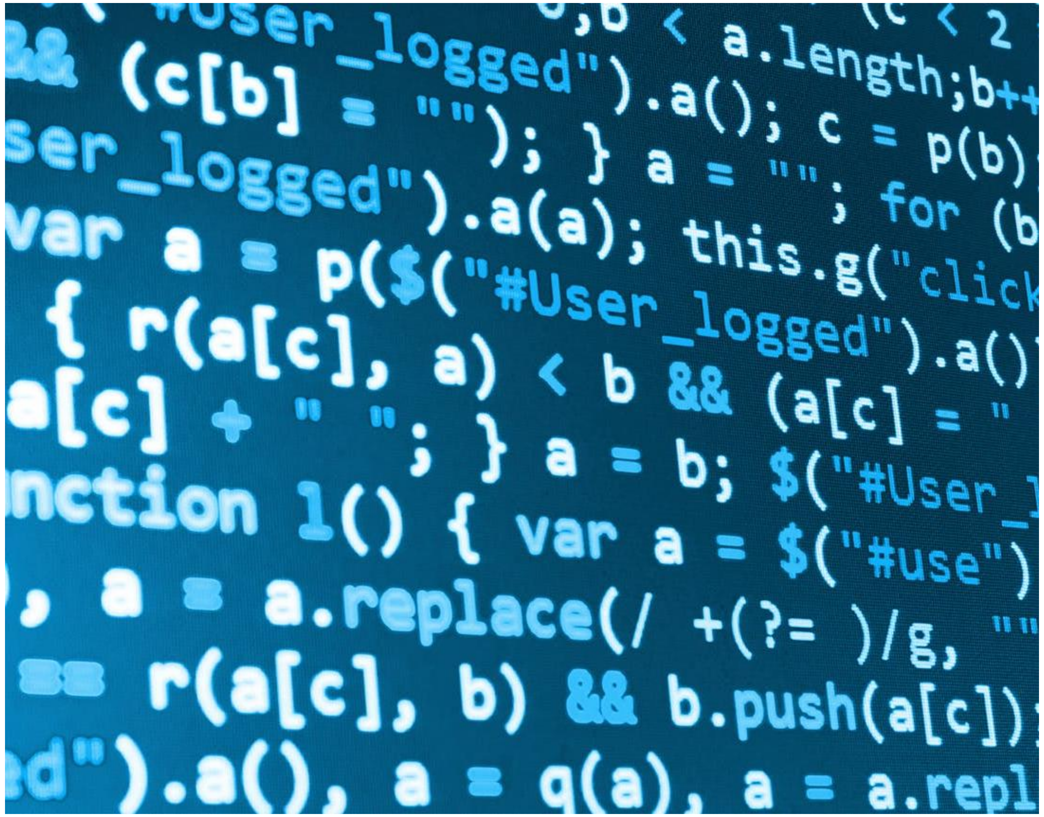
- Algorithmen liegen alle sehr nah bei einander

- ε-First-Algorithmus erreicht die besten Ergebnisse, da es sich um ein stationäres Problem handelt

- ε-First-Algorithmus hat teilweise allerdings auch die schlechtestes Ergebnisse → Dieser Effekt tritt auf, wenn der optimale Arm zu Beginn ungewöhnlich schlecht performt

- bei dem ε-Decreasing-Algorithmus wird dieser Effekt reduziert

Weitere Lösungsalgorithmen



- Upper Confidence Bounds Algorithmus

- Thompson Sampling

- Monte-Carlo-Algorithmus

Inhaltsverzeichnis