



RETTUNGSWAGEN DISPATCH BOT

Deborah Burri

Fachhochschule Nordwestschweiz, FHNW
Olten, 31.12.2025

Impressum:

Deckblatt: Symbolbild: <https://insurando.ch/blog/gesundheit/wer-bezahlt-die-kosten-fuer-den-krankenwagen/>

Inhaltsverzeichnis

| | |
|---|---|
| 1. Einleitung | 1 |
| 2. Motivation und Themenwahl | 1 |
| 3. Theoretische Grundlagen des Reinforcement Learnings | 2 |
| 4. Problemdefinition | 2 |
| 5. Design des eigenen Environments | 3 |
| 6. Reward-Design und Reward Shaping | 3 |
| 7. Vorgehen und experimentelle Umsetzung | 3 |
| 7.1. Iteratives Vorgehen und Analyse des Lernverhaltens | 3 |
| 7.2. Vergleich mit Random Policy | 4 |
| 7.3. Einsatz von Seeds und Robustheitsanalyse | 4 |
| 8. Was habe ich gelernt? | 5 |
| 9. Reflexion | 5 |
| 10. Fazit | 5 |
| 11. Eigenständniserklärung | 6 |
| 12. Verzeichnisse | 7 |
| 12.1. Abbildungsverzeichnis | 7 |

1. Einleitung

Die effiziente Disposition von Rettungswagen ist eine zentrale Aufgabe im Rettungswesen, bei der zeitkritische Entscheidungen in Notfallsituationen getroffen werden müssen. In Notfällen ist es entscheidend, innerhalb kürzester Zeit den geeigneten Rettungswagen auszuwählen und diesen zum Einsatzort zu senden. Dabei spielen sowohl die Entfernung als auch die aktuelle Verfügbarkeit der Fahrzeuge und die Priorität des Notfalls eine wesentliche Rolle.

Die zunehmende Komplexität moderner Rettungssysteme macht deutlich, dass starre regelbasierte Ansätze nur begrenzt geeignet sind, um solche dynamischen Entscheidungssituationen optimal zu bewältigen. Reinforcement Learning bietet hier einen vielversprechenden Ansatz, da ein Agent durch wiederholte Interaktion mit einer Umgebung lernen kann. Ziel dieser Arbeit ist es daher, einen Reinforcement-Learning-Agenten zu entwickeln und zu evaluieren, der die Disposition von Rettungswagen in einer simulierten Umgebung übernimmt.

2. Motivation und Themenwahl

Die Wahl des Themas basiert auf einem grundlegenden Interesse am medizinischen Bereich sowie an der Frage, wie künstliche Intelligenz Entscheidungsprozesse in sicherheitskritischen Domänen unterstützen kann. Im Rahmen des Fachs *Mensch-KI-Automation* wurde unter anderem das Thema Triage behandelt, wodurch ein vertieftes Verständnis für prioritätsbasierte Entscheidungen in Notfallsituationen entstand. Diese Inhalte lieferten einen wichtigen Impuls für die Auseinandersetzung mit der Rettungswagen-Disposition als konkreten Anwendungsfall.

Darüber hinaus erscheint das Thema besonders geeignet, um zentrale Konzepte des Reinforcement Learnings praktisch umzusetzen und zu reflektieren. Die Kombination aus gesellschaftlicher Relevanz, technischer Komplexität und persönlichem Interesse bildete somit die Grundlage für dieses Projekt.

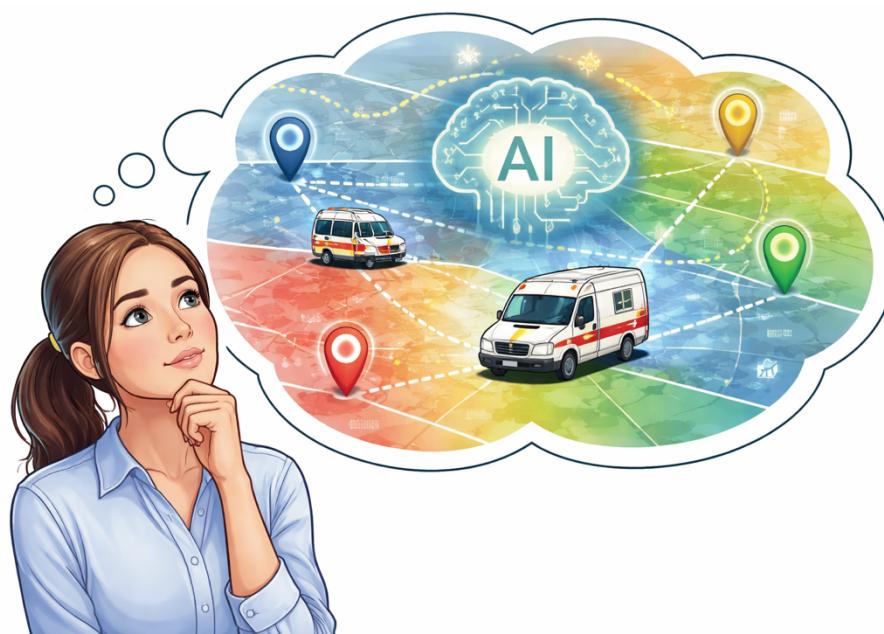


Abb. 1: Themenwahl RTW und RL-Bot (Generierung mit ChatGPT)

3. Theoretische Grundlagen des Reinforcement Learnings

Reinforcement Learning beschreibt eine Lernmethode, bei dem ein Agent durch Interaktion mit einer Umgebung lernt, um eine optimale Handlungsstrategie zu entwickeln. In jedem Zeitschritt beobachtet der Agent einen Zustand, wählt eine Aktion und erhält als Rückmeldung einen numerischen Reward. Ziel ist es, den erwarteten kumulierten Reward über eine Trainingsepisode hinweg zu maximieren.

Das betrachtete Problem lässt sich als Markov Decision Process (MDP) modellieren, bestehend aus einem Zustandsraum, einem Aktionsraum, einer Übergangsfunktion, einer Reward-Funktion sowie einem Diskontrfaktor. Der Zustandsraum wird dabei durch eine MultiDiscrete-Observation beschrieben, welche relevante Systeminformationen wie Positionen und Auslastungszustände der Einsatzfahrzeuge sowie Merkmale eingehender Notrufe umfasst. Zur Verwendung eines tabellarischen Lernverfahrens wird dieser mehrdimensionale Zustandsraum auf einen eindeutigen eindimensionalen Index abgebildet, der als Zugriffsschlüssel für die Q-Funktion dient. (Siehe Abb. 2)

```
def obs_to_index(obs, env: EmergencyDispatchEnv) -> int:
    n0 = env.num_zones
    n1 = env.max_busy + 1
    n2 = env.num_zones
    n3 = env.max_busy + 1
    n4 = env.num_zones + 1
    n5 = 4

    a0, a1, a2, a3, a4, a5 = obs

    index = (
        a0
        + n0 * (
            a1
            + n1 * (
                a2
                + n2 * (
                    a3
                    + n3 * (
                        a4
                        + n4 * a5
                    )
                )
            )
        )
    )
    return int(index)
```

Abb. 2: MDP-Zyklus (VSCode)

Für die vorliegende Arbeit wurde Q-Learning als Lernverfahren gewählt. Q-Learning ist ein modellfreier Reinforcement-Learning-Ansatz, bei dem eine Q-Funktion gelernt wird, die den erwarteten diskontierten langfristigen Reward eines Zustand-Aktions-Paares approximiert. Die optimale Policy ergibt sich durch die Auswahl derjenigen Aktion, die für einen gegebenen Zustand den höchsten Q-Wert aufweist.

Während des Trainings wird eine ϵ -greedy-Strategie eingesetzt, um Exploration und Exploitation auszubalancieren. (Siehe Abb. 3) Dadurch wird sichergestellt, dass der Agent sowohl neue Aktionen ausprobiert als auch bereits bekannte, erfolgreiche Strategien nutzt.

```
if np.random.rand() < epsilon:
    action = env.action_space.sample()
else:
    action = int(np.argmax(Q[state]))
```

Abb. 3: ϵ -greedy-Strategie (VSCode)

4. Problemdefinition

Die Rettungswagen-Disposition stellt ein dynamisches Entscheidungsproblem dar, bei dem mehrere konkurrierende Ziele berücksichtigt werden müssen. In der simulierten Umgebung existieren mehrere Zonen sowie zwei Rettungswagen, die sich zwischen diesen Zonen bewegen können. Zu zufälligen Zeitpunkten treten Notrufe auf, die jeweils einer Zone zugeordnet sind und eine bestimmte Priorität besitzen. Der Agent muss entscheiden, welcher Rettungswagen einem Notruf zugewiesen wird. Diese Entscheidung beeinflusst nicht nur den aktuellen Einsatz, sondern auch die zukünftige Einsatzbereitschaft der Fahrzeuge. Das Problem ist somit nicht rein kurzfristig lösbar, sondern erfordert eine Strategie, die langfristige Konsequenzen berücksichtigt.

5. Design des eigenen Environments

Für das Projekt wird ein eigenes Environment auf Basis der Gymnasium-Schnittstelle implementiert. Gymnasium stellt dabei lediglich eine standardisierte API zur Verfügung, während die eigentliche Modellierung des Problems vollständig eigenständig erfolgt. Das Environment definiert den Zustandsraum, den Aktionsraum, die Übergangsdynamik sowie die Reward-Funktion.

Der Zustandsraum umfasst die Position und den Belegungsstatus der beiden Rettungswagen sowie die Zone und Priorität eines aktiven Notrufs. Die Übergangsdynamik beschreibt unter anderem die Bewegung der Fahrzeuge, die Dauer von Einsätzen sowie das Auftreten neuer Notrufe. Durch diese Eigenimplementierung konnte das Szenario gezielt an die Fragestellung angepasst werden.

6. Reward-Design und Reward Shaping

Die Definition der Reward-Funktion erweist sich als einer der kritischsten Aspekte des Projekts. Der Environment-Reward bestraft ineffiziente Entscheidungen, etwa durch lange Distanzen zwischen Rettungswagen und Einsatzort oder durch Verzögerungen bei hochprioritären Einsätzen. Hohe Prioritäten führen dabei zu stärkeren negativen Rewards, da Verzögerungen in solchen Fällen als besonders kritisch bewertet werden.

In frühen Trainingsphasen zeigt sich, dass der Agent kaum lernt. Ursache hierfür ist ein zu spärliches Reward-Signal, das dem Agenten nur wenig Orientierung bietet. Um dieses Problem zu lösen, wird potentialbasiertes Reward Shaping eingesetzt. (Siehe Abb. 4). Die verwendete Potentialfunktion bewertet Zustände anhand der minimalen Distanz eines freien Rettungswagens zum aktuellen Notruf, gewichtet nach dessen Priorität. Dadurch erhält der Agent kontinuierliches Feedback, das den Lernprozess erheblich beschleunigt, ohne die optimale Policy zu verändern.

```
def shaping_potential(obs, env: EmergencyDispatchEnv) -> float:
    a0_pos, a0_busy, a1_pos, a1_busy, call_zone, call_priority = obs

    # Fall: kein aktiver Notruf
    if call_zone == env.num_zones:
        return 0.0

    distances = []

    # Nur freie RTWs berücksichtigen
    if a0_busy == 0:
        distances.append(abs(a0_pos - call_zone))
    if a1_busy == 0:
        distances.append(abs(a1_pos - call_zone))

    if not distances:
        # Kein freier RTW verfügbar -> kein shaping
        return 0.0

    best_dist = min(distances)

    # Gewichtung nach Priorität
    priority_weight = 1.0 + float(call_priority)

    # Kleinere Distanz => höherer Wert
    return -best_dist * priority_weight
```

Abb. 4: Reward Shaping (VSCode)

7. Vorgehen und experimentelle Umsetzung

7.1. Iteratives Vorgehen und Analyse des Lernverhaltens

Zu Beginn des Projekts wird ein grober Projektplan erstellt und die Machbarkeit des Vorhabens analysiert. Ein erstes Code-Grundgerüst wird mithilfe von ChatGPT generiert und anschliessend schrittweise erweitert. Durch die Analyse der Lernkurven wird früh erkannt, dass der Agent zunächst nur sehr begrenzt lernt. Diese Beobachtung führt zu einer intensiven Auseinandersetzung mit dem Reward-Design und letztlich zur Einführung von Reward Shaping als zentralen Verbesserungsschritt.

7.2. Vergleich mit Random Policy

Um die Leistungsfähigkeit des Agenten objektiv bewerten zu können, wird zusätzlich eine Random Policy implementiert. Diese wählt in jedem Zustand eine Aktion zufällig und berücksichtigt weder die Distanz zum Einsatzort noch die Priorität des Notrufs oder die Verfügbarkeit der Rettungswagen. Die Random Policy dient somit als Baseline, die das Verhalten eines nicht-intelligenten Systems repräsentiert.

Der Vergleich zwischen der gelernten Greedy-Policy des Q-Learning-Agenten und der Random Policy zeigt deutlich, dass der Agent signifikant bessere Ergebnisse erzielt. Während die Random Policy über die Trainigsepisoden hinweg überwiegend negative Rewards akkumuliert, erreicht die gelernte Policy deutlich höhere durchschnittliche Rewards. Dieser Unterschied belegt, dass der Agent tatsächlich eine sinnvolle Dispatch-Strategie erlernt hat und nicht lediglich zufällige Entscheidungen trifft. Der Vergleich mit der Random Policy ist daher ein wichtiger Bestandteil der Evaluation und unterstreicht die Wirksamkeit des gewählten Lernansatzes.

7.3. Einsatz von Seeds und Robustheitsanalyse

Da Reinforcement Learning stark von Zufallsprozessen beeinflusst wird, werden mehrere Trainingsläufe mit unterschiedlichen Zufalls-Seeds durchgeführt. Seeds bestimmen den Verlauf der Zufallszahlen und beeinflussen unter anderem die Exploration des Agenten sowie die Generierung von Notrufen. Durch ein Multi-Seed-Experiment kann gezeigt werden, dass der Agent über verschiedene Startbedingungen hinweg konsistent lernt. Die Lernkurven weisen über alle Seeds hinweg einen ähnlichen Verlauf auf, was auf ein robustes Lernverhalten hindeutet. (Siehe Abb. 5)

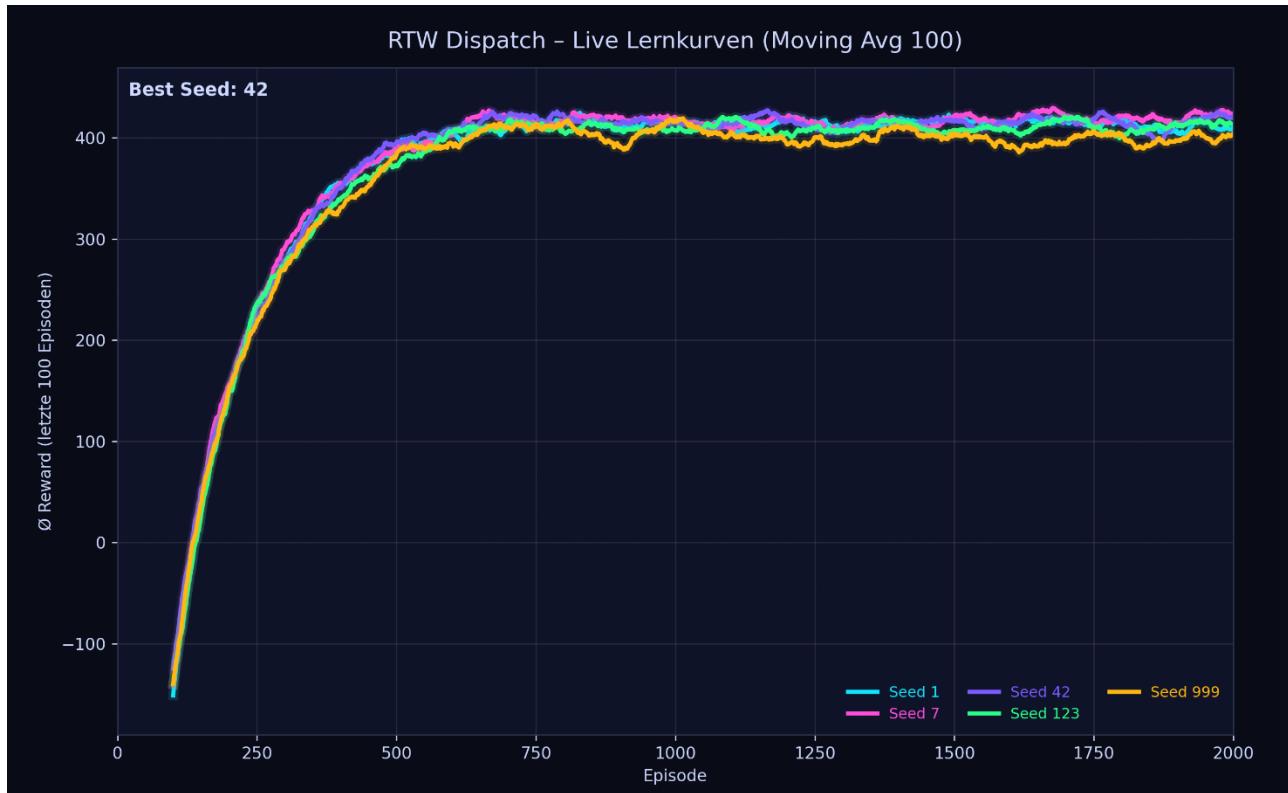


Abb. 5: RTW-Dispatch - Lernkurven (Python Plot)

8. Was habe ich gelernt?

Im Rahmen des Projekts werden zahlreiche zentrale Konzepte des Reinforcement Learnings praktisch erprobt und vertieft. Dazu gehört speziell das Verständnis von Policies, die Begründung von Aktionen auf Basis von Q-Werten sowie die Bedeutung von Seeds für reproduzierbare Experimente. Besonders deutlich wird, dass das Reward-Design einen entscheidenden Einfluss auf den Lernerfolg hat und oftmals wichtiger ist als die Wahl des eigentlichen Lernalgorithmus.

9. Reflexion

Rückblickend hätte ein noch experimentellerer Ansatz, etwa der Vergleich mit regelbasierten Dispatch-Strategien, zusätzliche Erkenntnisse liefern können. Für zukünftige Arbeiten wäre es zudem interessant, das Environment zu erweitern, beispielsweise durch eine grössere Anzahl an Rettungswagen oder realistischere Einsatzdynamiken. Auch der Einsatz von Deep Reinforcement Learning zur Approximation der Q-Funktion stellt einen möglichen nächsten Schritt dar.

10. Fazit

In dieser Arbeit wird ein Reinforcement-Learning-Agent zur dynamischen Disposition von Rettungswagen entwickelt und evaluiert. Durch die Implementierung eines eigenen Environments, gezieltes Reward Shaping sowie den Vergleich mit einer Random Policy konnte gezeigt werden, dass der Agent in der Lage ist, eine effiziente und robuste Dispatch-Strategie zu erlernen. Die Ergebnisse verdeutlichen das Potenzial von Reinforcement Learning für komplexe, dynamische Entscheidungsprobleme im medizinischen Kontext.

11. Eigenständigkeitserklärung

Ich, Deborah Burri, habe die Arbeit mit dem Titel «Rettungswagen Dispach Bot» selbst und selbständig verfasst und erkläre hiermit,

- dass ich alle verwendeten Hilfsmittel (KI-Assistenzsysteme wie Chatbots [z.B. ChatGPT], Übersetzungs- [z.B. DeepL], Paraphrasier- [z.B. Scispace]) oder Programmierapplikationen [z.B. Github Copilot] deklariert habe;
- dass ich sämtliche von mir verwendeten Materialien wie Bilder oder Grafiken in der Arbeit mit Quellenangaben kenntlich gemacht habe, oder dass diese Materialien von mir selbst erstellt wurden;
- dass ich mir bewusst bin, dass die Arbeit auf Plagiate und auf Drittautorschaft menschlichen oder technischen Ursprungs (künstliche Intelligenz) überprüft werden kann bei Verdacht, aber auch ohne besonderen Anlass.

12. Verzeichnisse

12.1. Abbildungsverzeichnis

| | |
|---|---|
| Abb. 1: Themenwahl RTW und RL-Bot (Generierung mit ChatGPT) | 1 |
| Abb. 2: MDP-Zyklus (VSCode) | 2 |
| Abb. 3: ϵ -greedy-Strategie (VSCode) | 2 |
| Abb. 4: Reward Shaping (VSCode) | 3 |
| Abb. 5: RTW-Dispatch - Lernkurven (Python Plot) | 4 |