🐭➔🧀 Arbeitsblatt: Wie Maschinen lernen (Reinforcement Learning)

❓ Teil A: Für die Mittelstufe

Die Idee: Stell dir vor, du möchtest einer Computer-Maus beibringen, den Weg durch ein Labyrinth zum Käse zu finden. Du sagst ihr aber nicht, wo sie langgehen soll. Stattdessen nutzt du ein Belohnungssystem: Findet sie den Käse, bekommt sie viele Punkte (+20). Läuft sie gegen eine Wand (schwarzer Block), kommt sie einfach nicht weiter und verschwendet Zeit. Die Maus merkt sich mit der Zeit, welche Wege zum Ziel führen und welche Sackgassen sind.

Aufgaben:

1. **Der Plan:** Die Maus steht vor einer Kreuzung. Oben ist der Käse 3 Felder entfernt, aber ein schwarzer Block versperrt den direkten Weg. Rechts ist der Käse 5 Felder entfernt, aber der Weg ist frei. Welchen Weg wird die Maus nach einigem Training wählen und warum?

2. **Das Ziel:** Du verschiebst den Käse im Editor plötzlich auf die andere Seite des Labyrinths. Warum läuft die Maus zuerst noch in die alte Richtung, bevor sie nach einer Weile den neuen Weg lernt?

❓ Teil B: Für die Oberstufe

Das Konzept: Reinforcement Learning basiert auf dem Zusammenspiel von einem **Agenten** (Maus) und seiner **Umgebung** (Gitterwelt). Der Agent führt **Aktionen** aus und erhält Rückmeldungen. Ziel ist die Maximierung des **kumulativen Rewards**. In dieser Simulation gibt es keine negativen Fallen, sondern nur "Hindernisse". Der Lerneffekt entsteht dadurch, dass nur der Pfad zum Ziel positive Werte in die Umgebung "ausstrahlt".

Aufgaben:

1. **Exploration vs. Exploitation:** Beobachte die Mäuse bei einem niedrigen Bellman-Horizont (Slider links). Warum ist das "sinnlose" Umherirren am Anfang notwendig, damit die KI später eine perfekte Policy-Map (Pfeil-Karte) erstellen kann?

2. **Diskontierung (γ):** Stell den Gamma-Slider auf einen sehr niedrigen Wert (z.B. 0.5). Beobachte die Q-Tabelle. Warum werden die Zahlenwerte zum Rand hin nun viel schneller klein (nahe 0), als wenn Gamma auf 0.99 steht?

🧑‍🏫 Teil C: Lehrer-Infobox (Hintergrundwissen)

1. Die Bellman-Gleichung in einfacher Form: Mit Belohnung, aber keine Fallen

In dieser App nutzen wir eine positive Bestärkung. Die Gleichung lautet:

$$V(s) \;=\; \max\left[\, R(s,a) \;+\; \gamma \cdot V(\mathrm{s}')\,\right]$$

Da es keine Fallen gibt, ist R (Belohnung) fast überall 0, außer beim Käse (R=20). Das Wissen über die 20 Punkte wird durch den Slider ("Horizont") schrittweise auf die benachbarten Felder übertragen. Ein Feld ist also nur deshalb etwas wert, weil es "weiß", dass sein Nachbar näher am Käse liegt.

Idee dahinter: Der Wert von “jetzt” = (sofortige Belohnung) + (abgezinst) * (Wert von “danach”)

Diskontierung (γ): γ ∈ [0,1] steuert, wie stark zukünftige Belohnungen zählen: kleines γ → kurzsichtig, γ nahe 1 → langfristig.

Warum „max“? Q-Learning bewertet die aktuelle Aktion danach, wie gut man ab dem nächsten Zustand optimal weiterhandeln könnte (beste Folgeentscheidung).

Labyrinth-Effekt: Trifft die Maus das Ziel, wird zuerst vor allem der letzte Schritt vor dem Ziel aufgewertet

„Belohnung wandert zurück“: In späteren Durchläufen werden dann auch frühere Schritte größer, weil deren nächste Zustände inzwischen höhere Q-Werte haben. So breitet sich der Wert vom Ziel rückwärts über den Pfad aus.

Vorwärts vs. rückwärts: Die Maus erlebt vorwärts, aber die Wertinformation propagiert rückwärts im Effekt, weil jedes Update den Wert von „danach“ in „jetzt“ einbaut.

Warum nicht sofort alles? Weil Q-Learning lokal pro Übergang updatet; daher braucht es mehrere Updates/Episoden, bis frühe Entscheidungen stark profitieren.

2. Insgesamt 16 Labyrinthe auf einmal: Die sogenannten Policy-Map

Warum zeigen wir 16 Mazes gleichzeitig?

- **Die Erkenntnis:** Die Schüler sollen sehen, dass die KI nicht *einen* Weg auswendig lernt. Sie lernt eine **Strategie für jeden Ort**.
- **Die Visualisierung:** Die Pfeile (Policy) zeigen die Richtung des stärksten Wert-Gefälles an. Es ist wie ein Höhenmodell: Der Käse ist der Gipfel, und die Mäuse laufen einfach immer "bergauf". Schwarze Blöcke sind unpassierbare Schluchten ohne Werte.

3. Der Lerneffekt (Slider)

Der Slider simuliert die **Informations-Propagation**. In einem echten neuronalen Netz braucht es tausende Versuche, bis der Reward vom Ziel bis zum Startpunkt "durchgereicht" wurde. Unser Slider kürzt diesen zeitaufwendigen Prozess ab, damit die Schüler sofort sehen können, wie sich das "Wissens-Feld" ausbreitet.