FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Vorlesung
Grundlagen adaptiver Wissenssysteme

Prof. Dr. Thomas Gabel
Frankfurt University of Applied Sciences
Faculty of Computer Science and Engineering
tgabel@fb2.fra-uas.de

Die optimale Q-Funktion

Die optimale Q-Funktion

Lernziele

- vollständige Ersetzung des Modells
- praktisch einsetzbare und oft verwendete Lernalgorithmen kennenlernen
- Verfahren, die in Interaktion mit der Umwelt einsetzbar sind

Die optimale Q-Funktion

Überblick

1. Motivation

Die optimale Q-Funktion

Überblick

1. Motivation

2. On-Policy- vs. Off-Policy-Lernen

Die optimale Q-Funktion

Überblick

1. Motivation
2. On-Policy- vs. Off-Policy-Lernen
3. Der Sarsa-Algorithmus

Die optimale Q-Funktion

Überblick

1. Motivation
2. On-Policy- vs. Off-Policy-Lernen
3. Der Sarsa-Algorithmus
4. Der Q-Lernalgorithmus

Die optimale Q-Funktion

Überblick

1. Motivation
2. On-Policy- vs. Off-Policy-Lernen
3. Der Sarsa-Algorithmus
4. Der Q-Lernalgorithmus

Ziel: Modellfreies Lernen

- Typische Lernsituation
  - Steuere einen Prozess, für den kein explizites Modell vorhanden ist
  - nur auf Basis von Beobachtung der Simulation oder des realen Prozesses

Ziel: Modellfreies Lernen

- Typische Lernsituation
  - Steuere einen Prozess, für den kein explizites Modell vorhanden ist
  - nur auf Basis von Beobachtung der Simulation oder des realen Prozesses
- Value Iteration / Policy Iteration benötigen Modell an zwei Stellen:

Ziel: Modellfreies Lernen

- Typische Lernsituation
  - Steuere einen Prozess, für den kein explizites Modell vorhanden ist
  - nur auf Basis von Beobachtung der Simulation oder des realen Prozesses
- Value Iteration / Policy Iteration benötigen Modell an zwei Stellen:
  1. Schätzen des Kostenwerts:

$$V_{k+1}(i) = \min_{a \in A(i)} \sum_{j=0}^{n} p_{ij}(a)(c(i,a) + V_k(j))$$

  2. Bestimmung der Strategie:

$$\pi(i) = \arg\min_{a \in A(i)} \sum_{j=0}^{n} p_{ij}(a)(c(i,a) + V_k(j))$$

Erinnerung: Q-Funktionen (1)

Definition (Zustands-Aktions-Wertfunktion Q)

Für eine gegebene Strategie π ist eine Zustands-Aktions-Wertfunktion $Q^\pi : S \times A \to \mathbb{R}$ definiert als

$$Q^\pi(i, a) := \sum_{j=0}^{n} p_{ij}(a) \left(c(i, a) + Q^\pi(j, \pi(j)) \right)$$

für alle $i \in S = \{0, \ldots, n\}$ und alle $a \in A$.

Erläuterung:

- Eine Q-Funktion schätzt die erwarteten Kosten des Agenten ab, wenn dieser im Zustand i die Aktion a wählen würde und sich danach gemäß Strategie π verhalten würde.

Erinnerung: Q-Funktionen (2)

- Es gilt $V^{\pi}(i) = Q^{\pi}(i, \pi(i))$ (VE8) und daher lässt sich die Q-Funktion (sh. vorige Folie) auch ausdrücken als:

$$Q^{\pi}(i, a) := \sum_{j=0}^{n} p_{ij}(a) \left(c(i, a) + V^{\pi}(j) \right)$$

Erinnerung: Q-Funktionen (2)

- Es gilt $V^\pi(i) = Q^\pi(i, \pi(i))$ (VE8) und daher lässt sich die Q-Funktion (sh. vorige Folie) auch ausdrücken als:

$$Q^\pi(i,a) := \sum_{j=0}^{n} p_{ij}(a)\,(c(i,a) + V^\pi(j))$$

- Eine Strategie π' ist bekanntermaßen gierig (VE7) bezogen auf V^π, wenn

$$\pi'(i) = \arg\min_{a \in A} \sum_{j=0}^{n} p_{ij}(a)\,(c(i,a) + V^\pi(j))$$

- Durch Einsetzen ergibt sich damit: $\pi'(i) = \arg\min_{a \in A} Q^\pi(i,a)$
- Das bedeutet: Auf Basis von einer Q-Funktion ist das gierige Verbessern einer Strategie auch ohne Modell möglich.

Erinnerung: Q-Funktionen (3)

Achtung:

- Diese Erkenntnisse sagen noch nichts darüber aus, wie man eine Q-Funktion ermittelt (Strategiebewertung).

Erinnerung: Q-Funktionen (3)

Achtung:

- Diese Erkenntnisse sagen noch nichts darüber aus, wie man eine Q-Funktion ermittelt (Strategiebewertung).
  - also bspw. die Q-Funktion Q^{π} zur Bewertung der Strategie π
- In VE8 haben wir eine Möglichkeit kennengelernt, wie man eine Q-Funktion auch ohne Modell ermittelt kann.
  - auf Basis von

Erinnerung: Q-Funktionen (3)

Achtung:

- Diese Erkenntnisse sagen noch nichts darüber aus, wie man eine Q-Funktion ermittelt (Strategiebewertung).
  - also bspw. die Q-Funktion Q^{π} zur Bewertung der Strategie π
- In VE8 haben wir eine Möglichkeit kennengelernt, wie man eine Q-Funktion auch ohne Modell ermittelt kann.
  - auf Basis von Monte-Carlo-Methoden

Somit:

- Die Gleichung auf der vorangegangenen Folie gilt für beliebige Strategien π. Insbsd. also auch für die optimale Strategie π^*:

$$Q^*(i,a) := \sum_{j=0}^{n} p_{ij}(a)\,(c(i,a) + V^*(j))$$

- Demzufolge gilt auch: $V^*(i) = \min_{a \in A} Q^*(i,a)$

Erinnerung: Q-Funktionen (3)

Achtung:

- Diese Erkenntnisse sagen noch nichts darüber aus, wie man eine Q-Funktion ermittelt (Strategiebewertung).
  - also bspw. die Q-Funktion Q^{π} zur Bewertung der Strategie π
- In VE8 haben wir eine Möglichkeit kennengelernt, wie man eine Q-Funktion auch ohne Modell ermittelt kann.
  - auf Basis von Monte-Carlo-Methoden

Somit:

- Die Gleichung auf der vorangegangenen Folie gilt für beliebige Strategien π. Insbsd. also auch für die optimale Strategie π^*:

$$Q^*(i,a) := \sum_{j=0}^{n} p_{ij}(a)\,(c(i,a) + V^*(j))$$

- Demzufolge gilt auch: $V^*(i) = \min_{a \in A} Q^*(i,a)$
- Ziel: Iterative Algorithmen finden, die die optimale Q-Funktion Q^* ermitteln können.

Die optimale Q-Funktion

Überblick

1. Motivation
2. On-Policy- vs. Off-Policy-Lernen
3. Der Sarsa-Algorithmus
4. Der Q-Lernalgorithmus

Monte-Carlo versus Temporal Difference

- Im Kapitel zu zeitlichen Differenzmethoden (TD) wurden MC-Verfahren (TD(1)) und TD-Verfahren (TD(0) als Extreme gegenübergestellt.
- Erkenntnisse:
  - Zeitliche Differenzmethoden (TD(λ) mit $\lambda < 1$) bieten gewisse Vorteile gegenüber reinen Monte-Carlo-Verfahren.
    - u.a. niedrigere Schwankungen in den Ergebnissen, keine vollständigen Episoden müssen durchlaufen werden, Aktualisierung schon nach einzelnen Schritten

Kernidee:

- Nutzung von TD-Ideen anstatt von MC-basierter Strategiebewertung, wenn es darum geht, die optimale Strategie komplett ohne Modell zu erlernen.
- Realisierung mit iterativen Algorithmen

Modellfreies Lernen: On-Policy vs. Off-Policy (1)

Grundsätzliche Überlegung:

- Der lernfähige Agent interagiert mit der Umwelt, wählt im Zustand i eine Aktion a und gelangt so unter Erhalt von Kosten $c(i, a)$ in den Folgezustand j.

Modellfreies Lernen: On-Policy vs. Off-Policy (1)

Grundsätzliche Überlegung:

- Der lernfähige Agent interagiert mit der Umwelt, wählt im Zustand i eine Aktion a und gelangt so unter Erhalt von Kosten $c(i, a)$ in den Folgezustand j.
- Wenn es darum geht, eine modellfreie Strategiebewertung für eine gegebene Strategie π vorzunehmen, also Q^π zu ermitteln, so gibt es dafür zwei prinzipielle Möglichkeiten:

Definition (On-Policy-Lernen vs. Off-Strategie-Lernen)

1. **On-Policy:** Der Agent interagiert gemäß der zu bewertenden Zielstrategie mit der Umwelt.

Modellfreies Lernen: On-Policy vs. Off-Policy (1)

Grundsätzliche Überlegung:

- Der lernfähige Agent interagiert mit der Umwelt, wählt im Zustand i eine Aktion a und gelangt so unter Erhalt von Kosten $c(i, a)$ in den Folgezustand j.
- Wenn es darum geht, eine modellfreie Strategiebewertung für eine gegebene Strategie π vorzunehmen, also Q^π zu ermitteln, so gibt es dafür zwei prinzipielle Möglichkeiten:

Definition (On-Policy-Lernen vs. Off-Strategie-Lernen)

1. **On-Policy:** Der Agent interagiert gemäß der zu bewertenden Zielstrategie mit der Umwelt.
2. **Off-Policy:** Der Agent verhält sich gemäß einer anderen Strategie, will aber trotzdem etwas über π lernen (also π bewerten).

Modellfreies Lernen: On-Policy vs. Off-Policy (2)

Frage: Warum kann Off-Policy-Lernen überhaupt nützlich sein?

- um aus den Verhaltensweisen von Menschen oder von anderen Agenten zu lernen
- um Zustandsübergänge, die mit früheren Strategien generiert worden sind ($\pi_1, \pi_2, \ldots$) wiederzuverwenden

Modellfreies Lernen: On-Policy vs. Off-Policy (2)

Frage: Warum kann Off-Policy-Lernen überhaupt nützlich sein?

- um aus den Verhaltensweisen von Menschen oder von anderen Agenten zu lernen
- um Zustandsübergänge, die mit früheren Strategien generiert worden sind ($\pi_1, \pi_2, \ldots$) wiederzuverwenden
- um etwas über die optimale Strategie zu lernen, während gerade eine andere, explorative Strategie verfolgt wird

Modellfreies Lernen: On-Policy vs. Off-Policy (2)

Frage: Warum kann Off-Policy-Lernen überhaupt nützlich sein?

- um aus den Verhaltensweisen von Menschen oder von anderen Agenten zu lernen
- um Zustandsübergänge, die mit früheren Strategien generiert worden sind ($\pi_1, \pi_2, \ldots$) wiederzuverwenden
- um etwas über die optimale Strategie zu lernen, während gerade eine andere, explorative Strategie verfolgt wird
- um gleichzeitig über mehrere Strategien etwas zu lernen (d.h. mehrere Strategien zu bewerten, $Q^{\pi_1}, Q^{\pi_2}, \ldots$ simultan lernen), während aktuell eine andere Strategie verfolgt wird

Modellfreies Lernen: On-Policy vs. Off-Policy (2)

Frage: Warum kann Off-Policy-Lernen überhaupt nützlich sein?

- um aus den Verhaltensweisen von Menschen oder von anderen Agenten zu lernen
- um Zustandsübergänge, die mit früheren Strategien generiert worden sind ($\pi_1, \pi_2, \ldots$) wiederzuverwenden
- um etwas über die optimale Strategie zu lernen, während gerade eine andere, explorative Strategie verfolgt wird
- um gleichzeitig über mehrere Strategien etwas zu lernen (d.h. mehrere Strategien zu bewerten, $Q^{\pi_1}, Q^{\pi_2}, \ldots$ simultan lernen), während aktuell eine andere Strategie verfolgt wird

Nächste Schritte:

- Sarsa-Algorithmus für On-Policy-Lernen
- Q-Lernalgorithmus für Off-Policy-Lernen

Die optimale Q-Funktion

Überblick

Modellfreies On-Policy-Lernen (1)

der optimalen Wertfunktion

Kernideen

- verwende TD-artige Aktualisierung anstatt MC-basierter Strategiebewertung
- d.h. berücksichtige den (aktuellen) Einzelschritt, der als 5-Tupel vorliegt (i, a, c, j, a')

Modellfreies On-Policy-Lernen (1)

der optimalen Wertfunktion

Kernideen

- verwende TD-artige Aktualisierung anstatt MC-basierter Strategiebewertung
- d.h. berücksichtige den (aktuellen) Einzelschritt, der als 5-Tupel vorliegt (i, a, c, j, a')
- Wenn man, wie auch oft in der Literatur, Zustände mit s bezeichnet und von Belohnungen r (rewards) anstatt von Kosten c spricht, so nimmt das 5-Tupel folgende Form an:

$$(s, a, r, s', a')$$

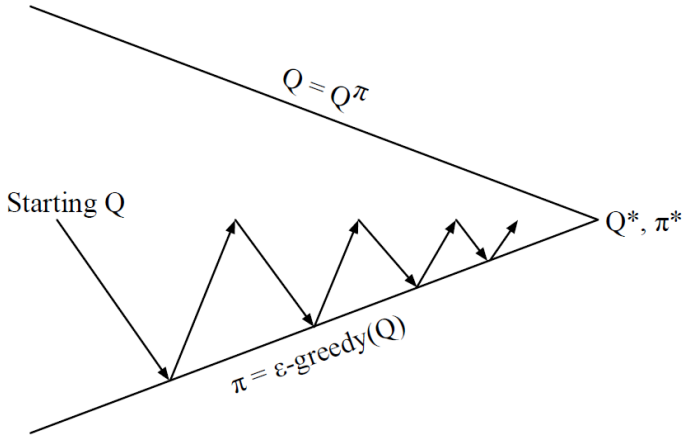
- Diese Buchstaben dienten der Namensgebung beim **SARSA-Algorithmus**.

Modellfreies On-Policy-Lernen (2)

der optimalen Wertfunktion

Kernideen (Fortsetzung)

- verfolge während des Lernens eine ε-gierige Strategie
- führe Aktualisierungsschritte nach jedem Zustandsübergang durch



Zweiphasiges Verfahren,
wobei beide Phasen in jedem Schritt abwechseln

- weiterhin: Strategieverbesserung dank Q-Funktionen ohne Modell möglich
- Strategiebewertung mit Sarsa, d.h. $Q \approx Q^{\pi}$
- Strategieverbesserung in jedem Schritt dank ε-gieriger Auswertung der aktuellen Wertfunktion Q

Der Sarsa-Algorithmus (1)

Sarsa-Algorithmus

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

Der Sarsa-Algorithmus (1)

Sarsa-Algorithmus

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

  Initialisiere Startzustand s_0; $t = 0$

  Wähle Aktion $a_0 := \arg\min_{a \in A} Q_k(s_0, a)$

    oder a_0 gemäss Explorationsstrategie beliebig

  REPEAT *// Schleife für aktuelle Episode*

Der Sarsa-Algorithmus (1)

Sarsa-Algorithmus

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

  Initialisiere Startzustand s_0; $t = 0$

  Wähle Aktion $a_0 := \arg\min_{a \in A} Q_k(s_0, a)$

    oder a_0 gemäss Explorationsstrategie beliebig

  REPEAT *// Schleife für aktuelle Episode*

    Wende a_t auf System an, beobachte s_{t+1} und Kosten $c(s_t, a_t)$

    Wähle Aktion $a_{t+1} := \arg\min_{a \in A} Q_k(s_{t+1}, a)$

      oder a_{t+1} gemäss Explorationsstrategie beliebig

Der Sarsa-Algorithmus (1)

Sarsa-Algorithmus

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

  Initialisiere Startzustand s_0; $t = 0$

  Wähle Aktion $a_0 := \arg\min_{a \in A} Q_k(s_0, a)$

    oder a_0 gemäss Explorationsstrategie beliebig

  REPEAT *// Schleife für aktuelle Episode*

    Wende a_t auf System an, beobachte s_{t+1} und Kosten $c(s_t, a_t)$

    Wähle Aktion $a_{t+1} := \arg\min_{a \in A} Q_k(s_{t+1}, a)$

      oder a_{t+1} gemäss Explorationsstrategie beliebig

    Führe **Lernschritt** durch:

    $Q_{k+1}(s_t, a_t) := (1 - \alpha) Q_k(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma Q_k(s_{t+1}, a_{t+1}))$

Der Sarsa-Algorithmus (1)

Sarsa-Algorithmus

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

  Initialisiere Startzustand s_0; $t = 0$

  Wähle Aktion $a_0 := \arg\min_{a \in A} Q_k(s_0, a)$

    oder a_0 gemäss Explorationsstrategie beliebig

  REPEAT *// Schleife für aktuelle Episode*

    Wende a_t auf System an, beobachte s_{t+1} und Kosten $c(s_t, a_t)$

    Wähle Aktion $a_{t+1} := \arg\min_{a \in A} Q_k(s_{t+1}, a)$

      oder a_{t+1} gemäss Explorationsstrategie beliebig

    Führe **Lernschritt** durch:

    $Q_{k+1}(s_t, a_t) := (1 - \alpha) Q_k(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma Q_k(s_{t+1}, a_{t+1}))$

    $t := t + 1$; $k := k + 1$; passe Lernrate α an

  UNTIL Terminalzustand erreicht

UNTIL Strategie optimal

Der Sarsa-Algorithmus (2)

Bemerkungen:

- Lernrate α muss im Laufe des Lernvorgangs kontinuierlich abgesenkt werden
- Explorationsstrategie kann ε-gierig sein, wobei die Explorationswahrscheinlichkeit im Laufe des Lernvorgangs kontinuierlich abgesenkt werden sollte gemäß der GLIE-Definition aus VE8 (Greedy in the Limit with Infinite Exploration)

Der Sarsa-Algorithmus (2)

Bemerkungen:

- Lernrate α muss im Laufe des Lernvorgangs kontinuierlich abgesenkt werden
- Explorationsstrategie kann ε-gierig sein, wobei die Explorationswahrscheinlichkeit im Laufe des Lernvorgangs kontinuierlich abgesenkt werden sollte gemäß der GLIE-Definition aus VE8 (Greedy in the Limit with Infinite Exploration)
- Nach jedem Schritt wird k um eins erhöht: Nach jedem Zustandsübergang (“zweiphasiges Verfahren”) wird die Q-Funktion aktualisiert.
  - Es wird keine perfekte Strategiebewertung durchgeführt, sondern Bewertung und (gierige) Verbesserung alternieren Schritt für Schritt. $\rightarrow$ k ist weglassbar
  - $Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}))$

Konvergenz von Sarsa

Theorem

Der Sarsa-Algorithmus konvergiert zur optimalen Wertfunktion, d.h. $Q(s,a) \rightarrow Q^*(s,a) \forall s,a$ bzw. $\lim_{k\rightarrow\infty} Q_k(s,a) = Q^*(s,a) \forall s,a$, wenn die folgenden Bedingungen erfüllt sind:

- Die von der Q-Funktion abgeleitete und durch den Agenten verwendete Strategie ist GLIE.
  - Alle Zustands-Aktions-Paare werden unendlich oft besucht und die Strategie konvergiert zu einer deterministischen Strategie.

Konvergenz von Sarsa

Theorem

Der Sarsa-Algorithmus konvergiert zur optimalen Wertfunktion, d.h. $Q(s,a) \rightarrow Q^*(s,a) \forall s,a$ bzw. $\lim_{k\rightarrow\infty} Q_k(s,a) = Q^*(s,a) \forall s,a$, wenn die folgenden Bedingungen erfüllt sind:

- Die von der Q-Funktion abgeleitete und durch den Agenten verwendete Strategie ist GLIE.
  - Alle Zustands-Aktions-Paare werden unendlich oft besucht und die Strategie konvergiert zu einer deterministischen Strategie.
- Die Lernrate α_k erfüllt die Robbins-Monroe-Bedingungen der stochastischen Approximation[a]

$$\sum_{k=0}^{\infty} \alpha_k = \infty \text{ und } \sum_{k=0}^{\infty} \alpha_k^2 < \infty$$

[a] z.B. $\alpha_m = \frac{1}{m}$, wobei mit m die Anzahl der Aktualisierungen für jedes Zustands-Aktions-Paar protokolliert werden muss.

Die optimale Q-Funktion

Überblick

Modellfreies Off-Policy-Lernen (1)

der optimalen Wertfunktion

Kernideen:

- Ziel ist das Lernen der optimalen Wertfunktion Q^*, für die gilt

$$Q^*(i,a) = \sum_{i=0}^{n} p_{ij}(a)\,(c(i,a) + \gamma\, V^*(j))$$

Modellfreies Off-Policy-Lernen (1)

der optimalen Wertfunktion

Kernideen:

- Ziel ist das Lernen der optimalen Wertfunktion Q^*, für die gilt

$$
\begin{aligned}
Q^*(i,a) &= \sum_{i=0}^{n} p_{ij}(a)\,(c(i,a) + \gamma\, V^*(j)) \\
&= \sum_{i=0}^{n} p_{ij}(a) \left(c(i,a) + \gamma \min_{b \in A} Q^*(j,b) \right)
\end{aligned}
$$

Modellfreies Off-Policy-Lernen (1)

der optimalen Wertfunktion

Kernideen:

- Ziel ist das Lernen der optimalen Wertfunktion Q^*, für die gilt

$$\begin{aligned} Q^*(i,a) &= \sum_{i=0}^{n} p_{ij}(a)\,(c(i,a) + \gamma\, V^*(j)) \\ &= \sum_{i=0}^{n} p_{ij}(a)\left(c(i,a) + \gamma \min_{b \in A} Q^*(j,b)\right) \end{aligned}$$

- Bei Sarsa wurde in der Aktualisierungsvorschrift für den aktuellen und für den folgenden Schritt eine Aktion gemäß aktueller ε-gieriger Strategie angenommen.
  $Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}))$

Modellfreies Off-Policy-Lernen (1)

der optimalen Wertfunktion

Kernideen:

- Ziel ist das Lernen der optimalen Wertfunktion Q^*, für die gilt

$$
\begin{aligned}
Q^*(i,a) &= \sum_{i=0}^{n} p_{ij}(a)\,(c(i,a) + \gamma\, V^*(j)) \\
&= \sum_{i=0}^{n} p_{ij}(a)\left(c(i,a) + \gamma \min_{b \in A} Q^*(j,b)\right)
\end{aligned}
$$

- Bei Sarsa wurde in der Aktualisierungsvorschrift für den aktuellen und für den folgenden Schritt eine Aktion gemäß aktueller ε-gieriger Strategie angenommen.
  $Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}))$
- Beim nun vorgestellten Q-Lernen wird a_{t+1} durch eine alternative Aktion a^* ersetzt werden.
  $Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma Q(s_{t+1}, a^*))$

Modellfreies Off-Policy-Lernen (2)

der optimalen Wertfunktion

Aktualisierung der Q-Funktion beim **Q-Learning**

Q-Lernen passt die Wertfunktion für den aktuellen Schritt unter der Annahme an, dass der Agent im Folgeschritt die bestmögliche (gierige) Aktion wählen würde.

Modellfreies Off-Policy-Lernen (2)

der optimalen Wertfunktion

Aktualisierung der Q-Funktion beim **Q-Learning**

Q-Lernen passt die Wertfunktion für den aktuellen Schritt unter der Annahme an, dass der Agent im Folgeschritt die bestmögliche (gierige) Aktion wählen würde.

- Fakt: Aufgrund der Erfordernis nach Exploration wird der Agent dies im Folgezustand jedoch nur mit einer gewissen Wahrscheinlichkeit tun (z.B. mit $1 - \varepsilon$).
- Aktualisierungsvorschrift:

$$Q_{k+1}(i,a) := (1-\alpha)\, Q_k(i,a) + \alpha\,(c(i,a) + \min_{a' \in A(j)} Q_k(j,a'))$$

Modellfreies Off-Policy-Lernen (2)

der optimalen Wertfunktion

Aktualisierung der Q-Funktion beim **Q-Learning**

Q-Lernen passt die Wertfunktion für den aktuellen Schritt unter der Annahme an, dass der Agent im Folgeschritt die bestmögliche (gierige) Aktion wählen würde.

- Fakt: Aufgrund der Erfordernis nach Exploration wird der Agent dies im Folgezustand jedoch nur mit einer gewissen Wahrscheinlichkeit tun (z.B. mit $1 - \varepsilon$).
- Aktualisierungsvorschrift:

$$Q_{k+1}(i,a) := (1-\alpha)\, Q_k(i,a) + \alpha\,(c(i,a) + \min_{a' \in A(j)} Q_k(j,a'))$$

- Bezug zur "alternativen Aktion" (sh. vorige Folie): $a^* = \arg\min_{a' \in A(j)} Q_k(j,a')$

Modellfreies Off-Policy-Lernen (3)

der optimalen Wertfunktion

Bemerkungen:

- Während bei Sarsa ein 5-Tupel $((i, a, c, j, a')$ bzw. (s, a, r, s', a') bzw. $(s_t, a_t, r, s_{t+1}, a_{t+1}))$ erforderlich war,

Modellfreies Off-Policy-Lernen (3)

der optimalen Wertfunktion

Bemerkungen:

- Während bei Sarsa ein 5-Tupel $((i, a, c, j, a')$ bzw. (s, a, r, s', a') bzw. $(s_t, a_t, r, s_{t+1}, a_{t+1}))$ erforderlich war, genügt für eine Aktualisierung der Q-Funktion beim Q-Lernen ein 4-Tupel (i, a, c, j) (bzw. (s, a, r, s') bzw. (s_t, a_t, r, s_{t+1})).

Modellfreies Off-Policy-Lernen (3)

der optimalen Wertfunktion

Bemerkungen:

- Während bei Sarsa ein 5-Tupel $((i, a, c, j, a')$ bzw. (s, a, r, s', a') bzw. $(s_t, a_t, r, s_{t+1}, a_{t+1}))$ erforderlich war, genügt für eine Aktualisierung der Q-Funktion beim Q-Lernen ein 4-Tupel (i, a, c, j) (bzw. (s, a, r, s') bzw. (s_t, a_t, r, s_{t+1})).
- Der Folgezustand j und die Kosten $c(i, a)$ werden durch Beobachtung des Systems (real oder Simulation) bestimmt.
- Q-Lernen repräsentiert ein Verfahren stochastischer Approximation mit Lernrate α.

Der Q-Lernalgorithmus (1)

Algorithmus Q-Learning

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

Der Q-Lernalgorithmus (1)

Algorithmus Q-Learning

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

  Initialisiere Startzustand s_0; $t = 0$

  REPEAT *// Schleife für aktuelle Episode*

Der Q-Lernalgorithmus (1)

Algorithmus Q-Learning

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

  Initialisiere Startzustand s_0; $t = 0$

  REPEAT *// Schleife für aktuelle Episode*

    Wähle Aktion $a_t := \arg\min_{a \in A} Q_k(s_t, a)$

      oder a_t gemäss Explorationsstrategie beliebig

    Wende a_t auf System an, beobachte s_{t+1} und Kosten $c(s_t, a_t)$

Der Q-Lernalgorithmus (1)

Algorithmus Q-Learning

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

  Initialisiere Startzustand s_0; $t = 0$

  REPEAT *// Schleife für aktuelle Episode*

    Wähle Aktion $a_t := \arg\min_{a \in A} Q_k(s_t, a)$

      oder a_t gemäss Explorationsstrategie beliebig

    Wende a_t auf System an, beobachte s_{t+1} und Kosten $c(s_t, a_t)$

    Führe **Lernschritt** durch:

$$Q_{k+1}(s_t, a_t) := (1 - \alpha) Q_k(s_t, a_t) + \alpha \left(c(s_t, a_t) + \gamma \min_{b \in A} Q_k(s_{t+1}, b) \right)$$

Der Q-Lernalgorithmus (1)

Algorithmus Q-Learning

Initialisiere $Q_0(s, a)$ beliebig; $k = 0$

REPEAT *// Schleife, bis Lernziel erreicht*

  Initialisiere Startzustand s_0; $t = 0$

  REPEAT *// Schleife für aktuelle Episode*

    Wähle Aktion $a_t := \arg\min_{a \in A} Q_k(s_t, a)$

      oder a_t gemäss Explorationsstrategie beliebig

    Wende a_t auf System an, beobachte s_{t+1} und Kosten $c(s_t, a_t)$

    Führe **Lernschritt** durch:

$$Q_{k+1}(s_t, a_t) := (1 - \alpha) Q_k(s_t, a_t) + \alpha \left(c(s_t, a_t) + \gamma \min_{b \in A} Q_k(s_{t+1}, b)\right)$$

    $t := t + 1$; $k := k + 1$; passe Lernrate α an

  UNTIL Terminalzustand erreicht

UNTIL Strategie optimal

Der Q-Lernalgorithmus (2)

Erläuterungen:

- Die Lernrate α muss wieder so gewählt werden, dass sie den Bedingungen der stochastischen Approximation genügt, also z.B. wähle α umgekehrt proportional zur Anzahl der Aktualisierungen von Zustands-Aktionspaar (i, a).

Der Q-Lernalgorithmus (2)

Erläuterungen:

- Die Lernrate α muss wieder so gewählt werden, dass sie den Bedingungen der stochastischen Approximation genügt, also z.B. wähle α umgekehrt proportional zur Anzahl der Aktualisierungen von Zustands-Aktionspaar (i, a).
- Q-Learning ist eine Variante des optimistischen $TD(0)$-Algorithmus für den modellfreien Fall (d.h. simulationsbasiertes inkrementelles Ausführen der Value-Iteration-Vorschrift).
- Es existieren auch Varianten von Q-Learning, die die Idee des $TD(\lambda)$-Verfahrens aufgreifen: $Q(\lambda)$.

Konvergenz des Q-Lernens

Voraussetzungen:

- Jedes Paar (i, a) wird unendlich oft besucht
- und es gilt $\sum_{t=0}^{\infty} \alpha_t(i, a) = \infty$
- sowie $\sum_{t=0}^{\infty} \alpha_t(i, a)^2 < \infty$

Konvergenz des Q-Lernens

Voraussetzungen:

- Jedes Paar (i, a) wird unendlich oft besucht
- und es gilt $\sum_{t=0}^{\infty} \alpha_t(i, a) = \infty$
- sowie $\sum_{t=0}^{\infty} \alpha_t(i, a)^2 < \infty$

Diskontierte vs. SKP-Problem

- Wir haben Q-Lernen in seiner diskontierten Version kennengelernt (Diskontierungsfaktor γ mit $\gamma < 1$).
- Q-Lernen funktioniert auch für (undiskontierte) SKP-Probleme; dann ist $\gamma = 1$:

$$Q(i, a) := (1 - \alpha)\, Q(i, a) + \alpha\, (c(i, a) + \min_{a' \in A(j)} Q(j, a'))$$

- Voraussetzung zur Konvergenz bei SKP: Entweder alle Strategien sind erfüllend oder es gibt eine erfüllende Strategie und alle nicht erfüllenden Strategien erzeugen unendliche Kosten.

Zusammenfassung

- Q-Funktion beschreibt Pfadkosten für Zustands-Aktions-Paare.
- Q-Lernen: stochastische Approximation der optimalen Pfadkosten $Q^*(i, a)$ – *Dafür wird kein Modell benötigt!*

Zusammenfassung

- Q-Funktion beschreibt Pfadkosten für Zustands-Aktions-Paare.
- Q-Lernen: stochastische Approximation der optimalen Pfadkosten $Q^*(i, a)$ – *Dafür wird kein Modell benötigt!*
- Optimale Strategie ist durch Greedy-Auswertung der optimalen Q-Funktion

$$\pi^*(i) = \arg\min_{a \in A} Q^*(i, a)$$

gegeben. – *Dafür wird kein Modell benötigt!*

Zusammenfassung

- Q-Funktion beschreibt Pfadkosten für Zustands-Aktions-Paare.
- Q-Lernen: stochastische Approximation der optimalen Pfadkosten $Q^*(i, a)$ – *Dafür wird kein Modell benötigt!*
- Optimale Strategie ist durch Greedy-Auswertung der optimalen Q-Funktion

$$\pi^*(i) = \arg\min_{a \in A} Q^*(i, a)$$

  gegeben. – *Dafür wird kein Modell benötigt!*
- Konvergenzvoraussetzung: Alle Zustands-Aktionspaare werden unendlich oft angepasst $\Rightarrow$ Exploration notwendig!
- Praktische Erwägungen: Gute Ergebnisse können in bestimmten Umgebungen auch mit abweichenden Einstellungen erreicht werden.
  - z.B. konstante Lernrate von $\alpha = 0.1$ oder $\alpha = 1.0$ in deterministischen Umgebungen