Bewertendes Feedback

- Aktionen **bewerten** anstatt instruieren durch Vorgabe korrekter Aktionen
- Reines bewertendes Feedback hängt allein von der gewählten Aktion ab.
  Reines instruiertes Feedback hängt gar nicht von der gewählten Aktion ab.
- Überwachtes Lernen ist instruierend; Optimierung ist bewertend
- **Assoziativ** vs. **Nichtassoziativ**
  - Assoziativ: Eingaben werden auf Ausgaben abgebildet; lerne die beste Ausgabe **für jede** Eingabe
  - Nichtassoziativ: "lerne" (finde) eine beste Ausgabe
- n-armiger Bandit (zumindest wie wir ihn betrachten):
  - Nichtassoziativ
  - Bewertendes Feedback

Dieser Teil ist aus "Reinforcement Learning: An Introduction", Richard S. Sutton and Andrew G. Barto

Der n-armige Bandit

- Wähle wiederholt eine aus n Aktionen; jede Auswahl wird **Spiel** genannt

- Nach jedem Spiel a_t erhält man einen Reward r_t, wobei

$$E\left\langle r_t | a_t \right\rangle = Q^*(a_t)$$

  Dies sind unbekannte **Aktionsbewertungen**
  Verteilung von r_t hängt nur von a_t ab

- Ziel ist es den Reward auf lange Sicht zu maximieren, z.B. über 1000 Spiele

  Um die Aufgabe des n-armigen Banditen zu lösen,
  muss man eine Reihe von Aktionen **erkunden (explore)**
  und die besten von ihnen ausnutzen (exploit).

Das Exploration/Exploitation Problem

- Angenommen man definiert Schätzungen
  $Q_t(a) \approx Q^*(a)$ **Schätzungen für Aktionsbewertungen**

- Die $greedy$ Aktion zum Zeitpunkt t ist

$$\begin{aligned} a_t^* &= \arg\max_a Q_t(a) \\ a_t &= a_t^* \Rightarrow exploitation \\ a_t &\neq a_t^* \Rightarrow exploration \end{aligned}$$

- Man kann nicht immer erkunden, aber auch nicht immer ausnutzen

- Man kann das Erkunden nie stoppen, sollte es aber reduzieren

$Action - Value$ **Methoden**

- Methoden die ausschließlich Schätzungen für Aktionsbewertungen betrachten. Angenommen im t-ten Spiel wurde Action $a\, k_a$-mal gewählt, was die $Rewardsr_1, r_2, ...,_{k_a}$ produziert hat, dann

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}$$

  "Mittlerer $Reward$"

-

$$\lim_{k_a \to \infty} Q_t(a) = Q^*(a)$$

$\epsilon - greedy$ Aktionsauswahl

- $greedy$ Aktionsauswahl

$$a_t = a_t^* = \arg\max_a Q_t(a)$$

- ϵ-greedy Aktionsauswahl:

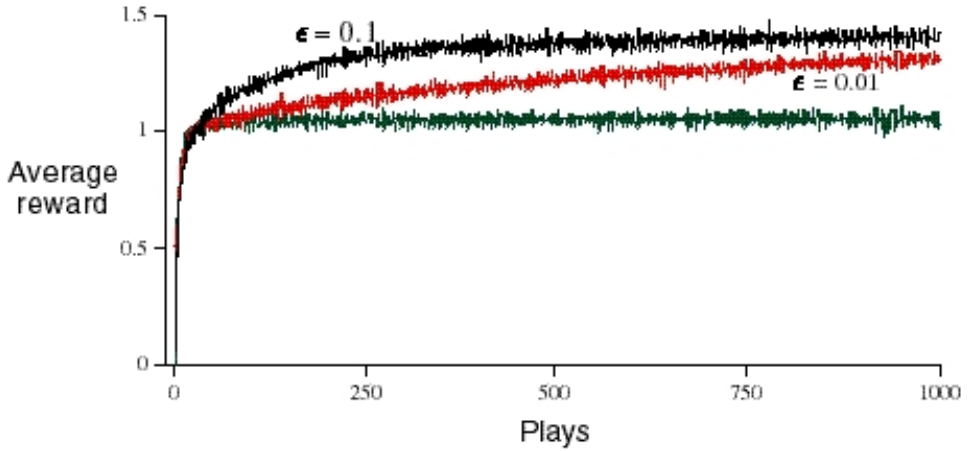
$$a_t = \begin{cases} a_t^* & \text{mit Wahrschinlichkeit} \quad 1-\epsilon \\ & \text{Zufallsaktion mit Wahrscheinlichkeit} \quad \epsilon \end{cases}$$

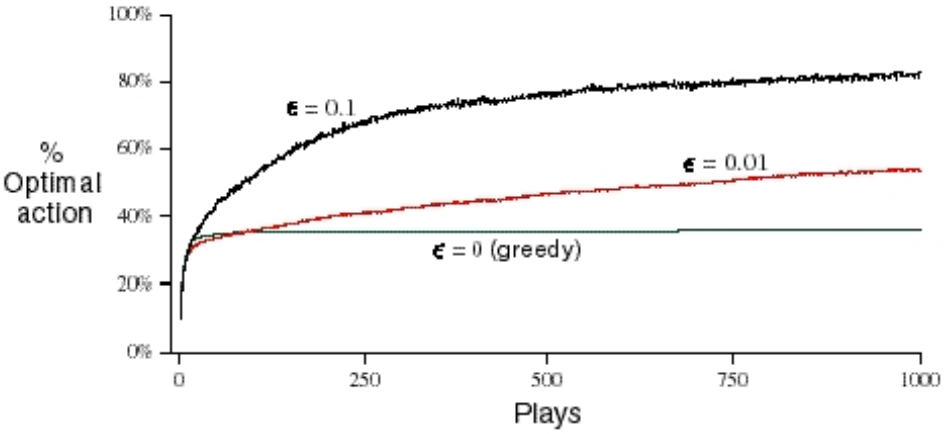
...der einfachste Weg um $exploration$ und $exploitation$ zu behandeln.

10 -armige Testumgebung

- $n = 10$ mögliche Aktionen
- Jedes $Q^*(a)$ wird zufällig aus einer Normalverteilung gewählt: $\eta(0, 1)$
- Jedes r_t ist auch normalverteilt: $\eta(Q^*(a_t), 1)$
- 1000 Spiele
- Wiederhole alles 2000 mal und mittel die Ergebnisse

ϵ -greedy Methode für die 10 armige Testumgebung





Softmax Aktionsauswahl

- $Softmax$ Aktionsauswahl-Methode staffelt Aktionswahrscheinlichkeiten nach geschätzten Werten.

- Die gebräuchlichste $softmax$ -Methode benutzt eine Gibbs-, oder Boltzmann-Verteilung:
  Wähle Aktion a im Spiel t mit Wahrscheinlichkeit

$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}},$$

  wobei τ die "Temperatur" ist

Binäre Banditenaufgabe

Angenommen es gibt nur **zwei** Aktionen: $a_t = 1$ oder $a_t = 2$
und nur **zwei** $Rewards : r_t = Erfolg$ oder $r_t = Fehler$

Dann könnte man ein **Ziel** oder eine **Soll-Aktion** definieren:

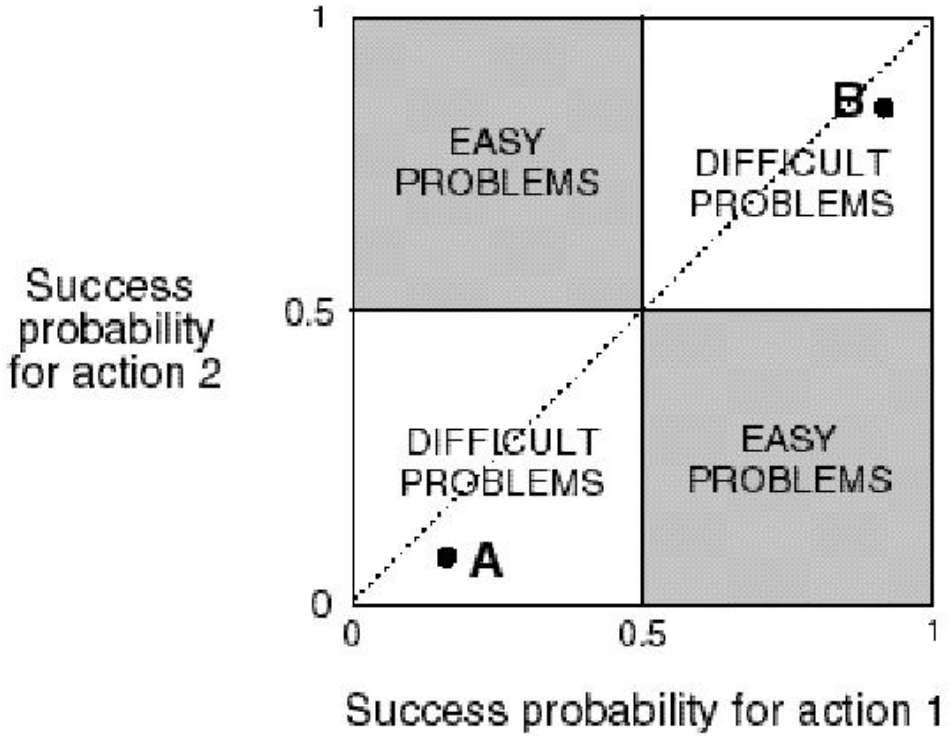
$$d_t = \begin{cases} a_t & \text{wenn} \quad Erfolg \\ \text{Die andere Aktion} & \text{wenn} \quad Fehler \end{cases}$$

und immer die Aktion spielen, die am öftesten das Ziel war

Dies ist ein **überwachter Algorithmus**
Er funktioniert gut bei deterministischen Problemen. . .

Zufallsraum

Der Raum aller möglichen binären Banditenaufgaben:



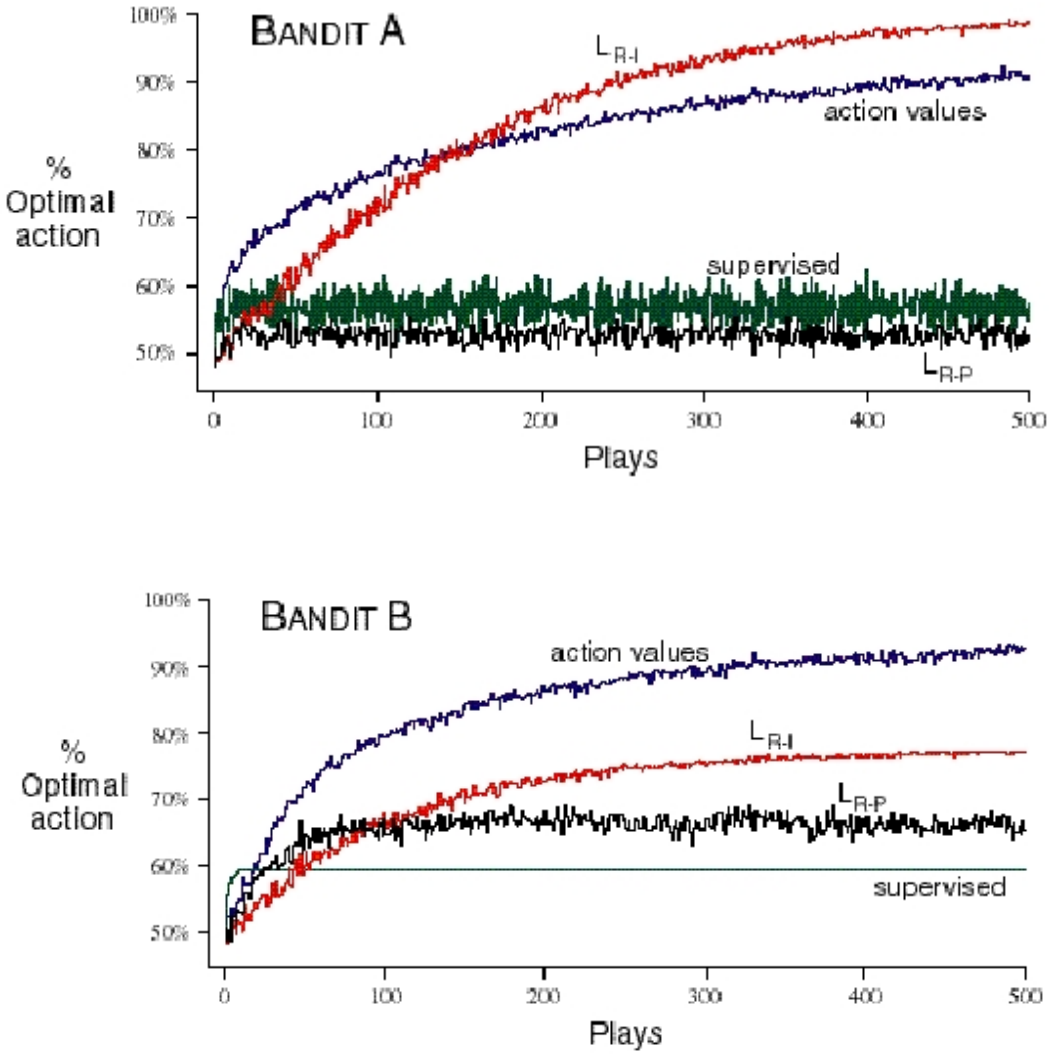
Linear Learning Automata

Sei $\pi_t(a) = Pr\{a_1 = a\}$ der einzige zu adaptierende Parameter

L_{R-1} (Linear, reward -inaction)
Bei $Erfolg : \pi_{t+1}(a_t) = \pi_t(a_t) + \alpha(1 - \pi_t(a_t)) \quad 0 < \alpha < 1$
(die anderen Aktionswahrsch. werden angepasst, so dass sie sich zu 1 summieren)
Bei $Misserfolg$: keine Änderung

L_{R-P} (Linear, reward -penalty)
Bei $Erfolg : \pi_{t+1}(a_t) = \pi_t(a_t) + \alpha(1 - \pi_t(a_t)) \quad 0 < \alpha < 1$
(die anderen Aktionswahrsch. werden angepasst, so dass sie sich zu 1 summieren)
Bei $Misserfolg : \pi_{t+1}(a_t) = \pi_t(a_t) + \alpha(0 - \pi_t(a_t)) \quad 0 < \alpha < 1$

Performanz der binären Banditenaufgaben A und B



Inkrementelle Implementierung

Erinnern wir uns an die Bewertungsmethode des mittleren $Rewards$:
Der Durchschnitt der ersten k $Rewards$ ist (bei Vernachlässigung der Abhängigkeit von a):

$$Q_k = \frac{r_1 + r_2 + \cdots + r_k}{k}$$

Kann dies inkrementell aufgebaut werden (ohne alle $Rewards$ zu speichern)?

Wir könnten eine laufende Summe und einen Zähler benutzen,
oder equivalent:

$$Q_{k+1} = Q_k + \frac{1}{k+1}\left[r_{k+1} - Q_k\right]$$

Dies ist eine übliche Form für $update$-Regeln:

$$NeueSchätzung = AlteSchätzung + Schrittweite\ [Ziel\ -\ AlteSchätzung\ ]$$

Nicht -stationäre Probleme

Q_k als $Reward$-Durchschnitt zu nehmen ist angemessen für ein stationäres Problem,
d.h., wenn sich kein $Q^*(a)$ über die Zeit ändert.

Aber nicht für ein nicht-stationäres Problem.

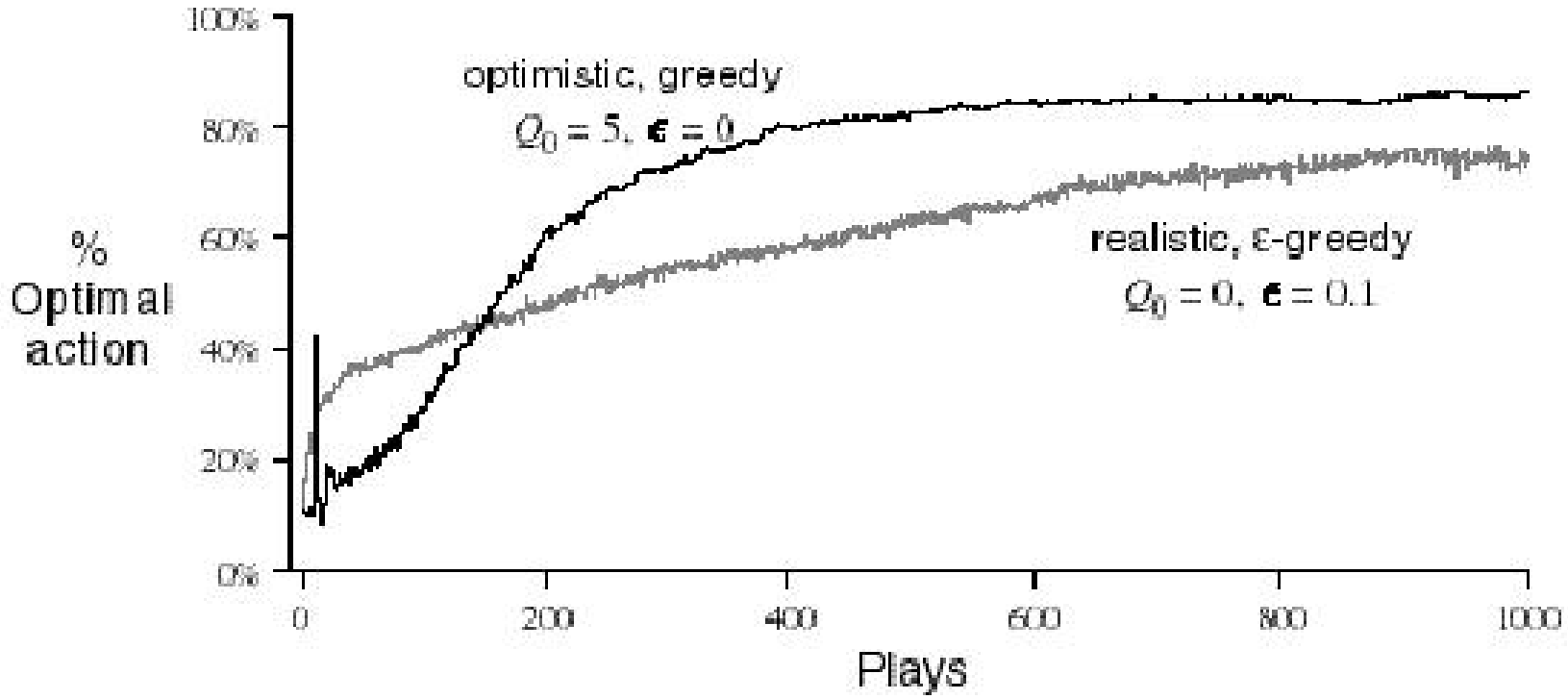
Besser in dem nicht-stationären Fall ist:

$$\begin{aligned} Q_{k+1} = & \quad Q_k + \alpha \left[r_{k+1} - Q_k \right] \quad fuer\ konstantes\ \alpha, 0 < \alpha \leq 1 \\ = & \quad (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha (1-\alpha)^{k-i} r_i \end{aligned}$$

exponential, recency-weighted average

Optimistische Initialwerte

- Alle bisherigen Methoden hängen von $Q_0(a)$ ab , d.h., sie sind ***biased***.
- Angenommen wir initialisieren die Aktionswerte **optimistisch**, z.B. in der 10-armigen Testumgebung: $Q_0(a) = 5$ für alle a



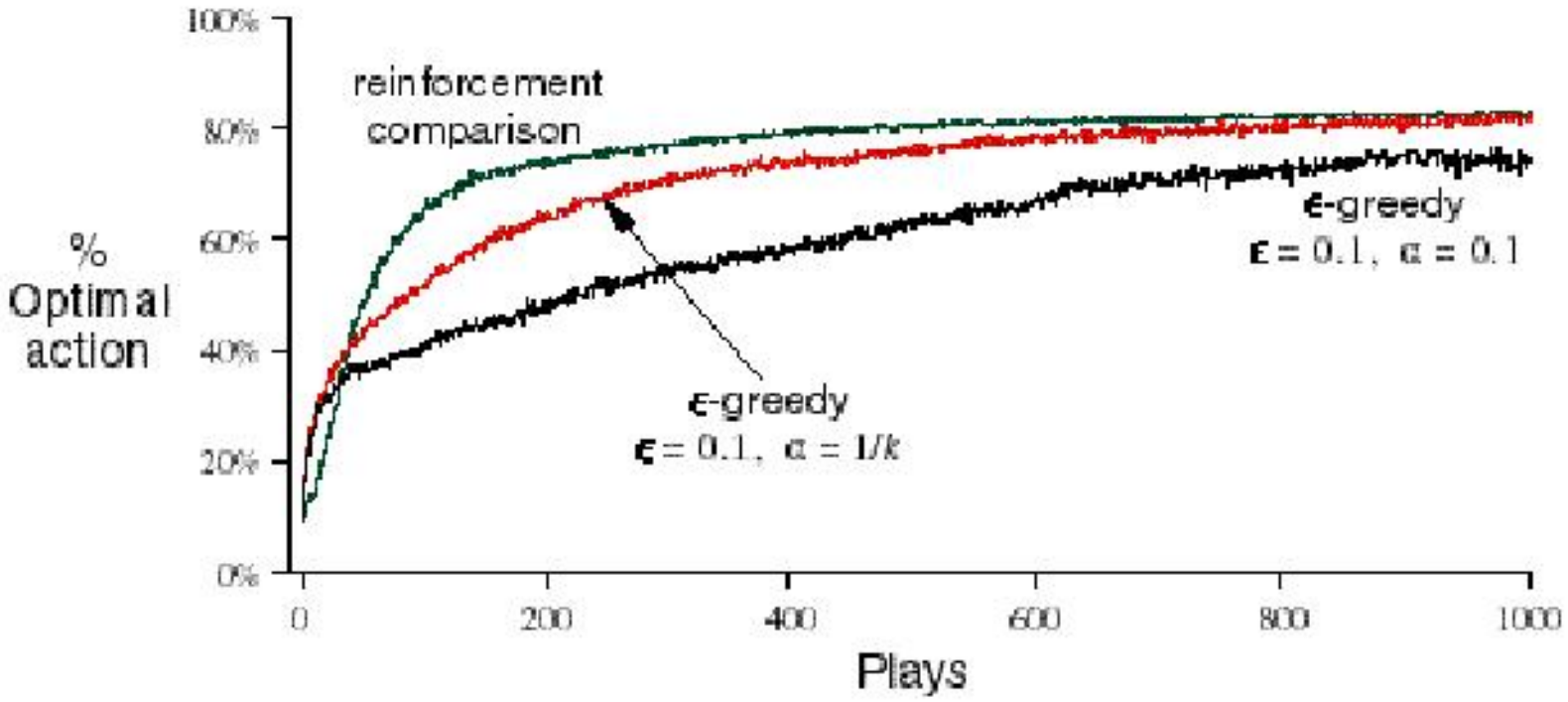
Reinforcement -Vergleich

- Vergleiche Rewards mit einem Referenz - $Reward \quad \bar{r}_t$, z.B. ein $Reward$-Durchschnitt aller erhaltenen $Rewards$
- Verstärke oder schwäche die gewählte Aktion in Abhängigkeit von $r_t - \bar{r}_t$ ab
- Sei $p_t(a)$ die **Präferenz** für Aktion a
- Präferenzen legen Aktions-Wahrscheinlichkeiten fest, z.B., durch die Gibbs Verteilung:

$$\pi_t(a) = Pr\{a_t = a\} = \frac{e^{p_t(a)}}{\sum_{b=1}^{n} e^{p_t(b)}}$$

- Dann: $p_{t+1}(a_t) = p_t(a) + [r_t - \bar{r}_t] \ \ und \ \bar{r}_{t+1} = \bar{r}_t + \alpha \, [r_t - \bar{r}_t]$

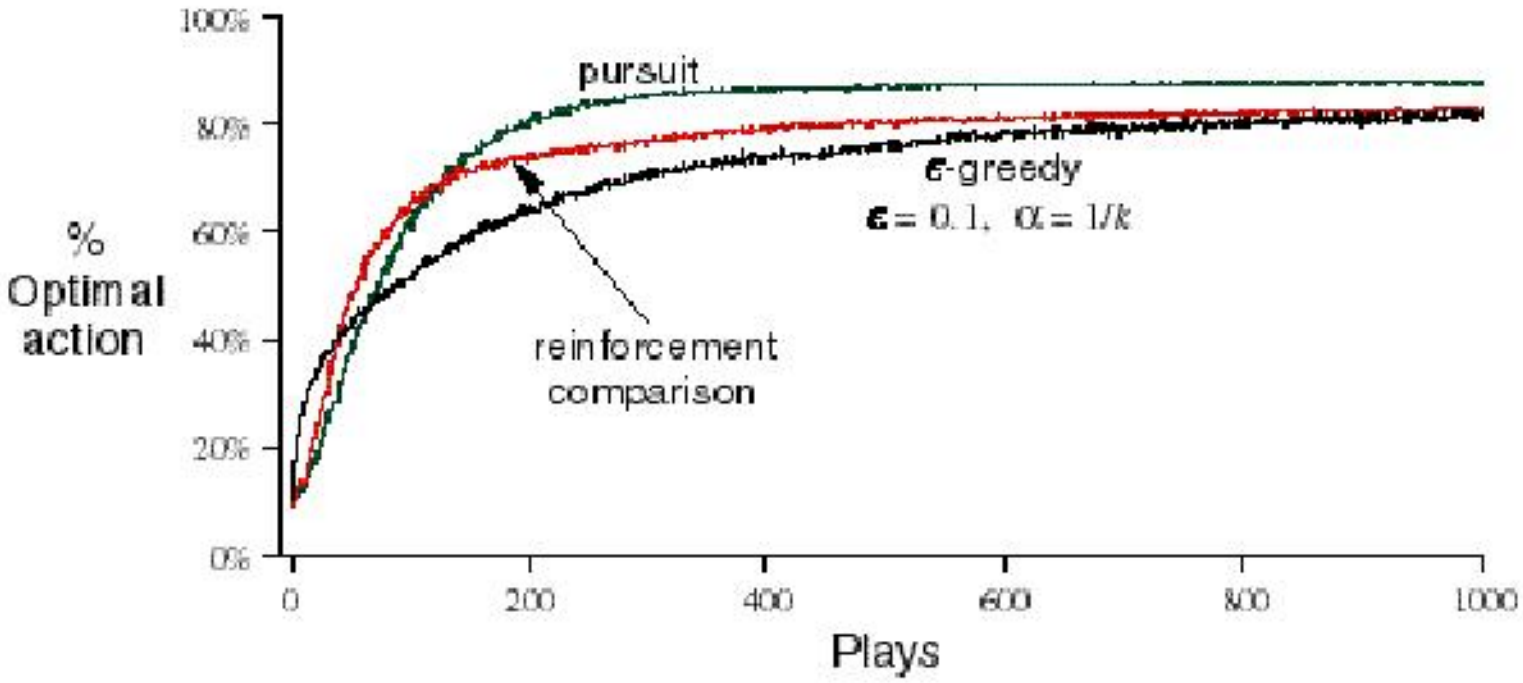
Performanz einer Reinforcement -Vergleich Methode



Pursuit Methoden

- Behandeln sowohl Schätzungen von Aktionswerten sowie Aktions-Präferenzen
- "Verfolge" immer die $greedy$ -Aktion, d.h. mache die $greedy$ -Aktion wahrscheinlicher für die Aktionsauswahl
- Aktualisiere nach dem t -ten Spiel die Aktionswerte um Q_{t+1} zu erhalten
- Die neue greedy -Aktion ist $a^*_{t+1} = \arg\max_a Q_{t+1}(a)$
- Dann: $\pi_{t+1}(a^*_{t+1}) = \pi_t(a^*_{t+1}) + \beta \left[1 - \pi_t(a^*_{t+1})\right]$
  und die Wahrscheinlichkeiten der anderen Aktionen werden reduziert um die Summe von 1 zu erhalten.

Performanz einer Pursuit Methode



Schlussfolgerungen

- Dies sind alles sehr einfache Methoden
  - aber sie sind kompliziert genug - wir werden auf ihnen aufbauen
  - Ideen für Verbesserungen:
  - Abschätzung von Unsicherheiten . . . Intervall-Abschätzung
  - Approximierung von $Bayes\,optimal\,solutions$
  - Gittens indices (klassische Lösung für n-armige Banditen zur Abwägung von $Exploration$ und $Exploitation$)

- Das komplette RL Problem bietet einige Lösungsideen . . .