B313 Machine Learning 2 - SS 25
Fachbereich 4: Informatik, Kommunikation und Wirtschaft
HTW Berlin - Hochschule für Technik und Wirtschaft Berlin

Beleg 1

Dies ist der erste Beleg des Kurses. Wie im Moodle angegeben, reichen Sie bitte für diesen Beleg ein `ipynb` oder `py` und ein `pdf` ein. Der Beleg soll die folgenden Inhalte vermitteln:

- ein erstes einfaches Beispiel aus dem Bereich des reinforcement learnings implementieren und verschiedene Strategien zur Lösung des Problems erkunden
- hierbei wiederholen wir/lernen wir grundlegende python und numpy Funktionen sowie plot-Funktionen kennen
- wir lernen wichtige grundlegende Begriffe und Denkweisen aus dem reinforcement learning kennen

Je nach Vorkenntnis kann der Beleg eine hohe Schwierigkeit aufweisen - lassen Sie sich davon nicht entmutigen. Insbesondere in diesem Fall, empfehle ich Ihnen die Teilnahme an der Übung in der wir uns geleitet durch den Beleg arbeiten werden und umfassende Hilfestellung möglich ist. Falls Sie den Beleg alleine bearbeiten möchten, können Sie eine ausführlichere Perspektive auf das Problem im Buch Stutton, Brato: Reinforcement Learning erhalten.

Aufgabe 1: k-armed bandit

In dieser Übung wollen wir den k-armed bandit - eine vereinfachte Slotmaschine mit k Hebeln implementieren und uns mit Strategien befassen diesen automatisiert bestmöglich zu spielen. Das Beispiel ist dem Buch Stutton, Brato: Reinforcement Learning entnommen und die Aufgaben an die darin enthaltene Diskussion angelehnt.

Am k-armed bandit können k verschiedene Aktionen durchgeführt werden (jeweils einer der k vielen Hebel betätigt werden.) Hierzu werden zunächst k viele Zufallszahlen $q_*(1), q_*(2), ..., q_*(k)$, nach einer Standardnormalverteilung $\mathcal{N}(0, 1)$ (Gaußverteilung mit Mittelwert 0 und Varianz 1) gezogen. Wird eine Aktion $a \in \{0, 1, \ldots, k-1\}$ ausgeführt, so erhalten wir einen zufälligen *reward* R der aus der Verteilung $\mathcal{N}(q_*(a), 1)$ gezogen wird.

Wir wollen nun Strategien entwickeln und untersuchen, an dieser Maschine zu spielen und dabei den größten reward zu erhalten - ohne dass wir die $q_*(j)$ kennen.

a) Implementieren Sie den k-armed bandit. Implementieren Sie hierfür die Funktion

```python
def get_bandit_function(k):
    '''
    Parameters:
    k : int - number of armes for the bandit

    Returns:
    callable - the bandit function bandit_function which can be called via bandit_function(action)
        which returns a random reward for selecting the action-th arm of the bandit
    list - list of centers of the different arms of the bandits
    '''

    # draw k gaussian random numbers q_i as centers
    # define a function which expects a parameter action and returns a random number from N(q_i,1)
    # return that function and a list of the q_i's
```

b) Spielen Sie für $k = 10$ für mindestens $N_e = 500$ Episoden jeweils für $N = 1000$ Schritte stets an einem zufällig ausgewählten Hebel und mitteln den in jedem Schritt erhaltenen reward über alle N_e Episoden. Für jede Episode soll ein neuer k-armed bandit erzeugt werden. Dies sollte Ihnen am Ende N Zahlen geben. Stellen Sie diese in einem Plot dar.

Aufgabe 2: Erwarteter Reward

Um an dem k-armed bandit möglichst gewinnbringend zu spielen, müssen wir anfangen aus den durchgeführten Spielen zu lernen.

Wir werden das Spiel für einen festen k-armed bandit wie oben wiederholt spielen (N Schritte), den erhaltenen reward beobachten und versuchen besonders lukrative Hebel (d.h. Hebel denen ein großes $q_*(j)$ zugeordnet ist) zu finden und bevorzugt zu wählen.

Hierfür führen wir die Zahlen $Q_t(a)$ die zu einem Zeitpunkt t für eine Aktion a den von uns erwarteten reward darstellt. Wir wählen zu Beginn: $Q_1(a) = Q_0$ für alle $a = 0, 1, \ldots, k-1$ mit zunächst $Q_0 = 0$. Danach wählen wir:

$$Q_t(a) = \frac{\text{Summe der rewards, die wir für Aktion } a \text{ bisher erhalten haben}}{\text{Anzahl wie oft wir Aktion } a \text{ bisher gewählt haben}}$$

wobei falls a bisher nicht gewählt wurde, $Q_t(a) = Q_0$ bleibt.

Um $Q_t(a)$ zu bestimmen, müssen wir uns **nicht** alle bisherigen rewards merken. Falls wir für den Hebel a die bisherigen rewards $R_1, R_2, \ldots, R_{n-1}$ und nun den neuen reward R_n erhalten haben können wir den neuen Wert von $Q_t(a)$ bestimmen via

$$Q_t(a) = Q_{t-1}(a) + \frac{1}{n}(R_n - Q_{t-1}(a)) \quad \text{falls} \quad A_t = a.$$

D.h. wir müssen uns nur die erwarteten rewards $Q_t(a)$ sowie die Anzahl, wie oft wir die Aktion a ausgeführt haben $N_t(a)$ merken.

a) Implementieren Sie eine Funktion `update_Q_and_N`. Diese soll die Parameter `Q`, `N`, `r` und `a` erwarten, wobei `Q` die Liste der aktuellen Werte für die erwarteten rewards Q_t, `N` die Liste der Anzahlen N_t, wie oft jede Aktion a bisher vorkam, r der aktuell erhaltene reward und a die aktuelle gewählte Aktion ist. Die Funktion `update_Q_and_N` soll die aktualisierten Listen Q und N mit den Werten Q_{t+1} und N_{t+1} zurückgeben, wobei

$$N_{t+1}(b) = \begin{cases} N_t(b) + 1 & \text{falls} \quad b = \texttt{a} \\ N_t(b) & \text{sonst} \end{cases}$$

$$Q_{t+1}(b) = \begin{cases} Q_t(b) + \frac{1}{N_{t+1}(b)}(\texttt{r} - Q_t(b)) & \text{falls} \quad b = \texttt{a} \\ Q_t(b) & \text{sonst} \end{cases}$$

Aufgabe 3: Greedy and epsilon-greedy

Um den k-armed bandit zu spielen können wir nun verschiedene Strategien wählen.

1) Greedy: Die greedy Strategie ist es, stets den Hebel mit dem maximal erwarteten reward zu nutzen, also

$$A_t = \underset{a \in \{0,1,\ldots,k-1\}}{\arg\max} \; Q_t(a).$$

Falls mehrere Q's maximal sind, sollten wir zufällig einen der Kandidaten auswählen.

2) ϵ-greedy: Für die ϵ-greedy Strategie wird zu Beginn ein $0 \leq \epsilon \leq 1$ festgelegt. Dann wird vor jedem Zug eine Zufallszahl z aus dem Intervall $[0, 1]$ gezogen. Falls $z < \epsilon$ ist, wird für A_t eine zufällige Aktion aus $\{0, 1, \ldots, k-1\}$ ausgewählt. Falls $z \geq \epsilon$ wird die greedy Wahl getroffen aus 1).

a) Implementieren Sie die Funktion

```
def eps_greedy(bandit_function,eps,Q_0=0,k=10,N=1000):
  '''
  Parameters:
  bandit_function : callable - function of the k-armed bandit, where bandit_function(action)
      returns the reward of pulling the k-th lever (action = 0,1,..,k-1)
  eps : float - epsilon of the eps-greedy strategy. Chooses with probability eps the
      exploration and with 1-eps the greedy strategy
  Q_0 : float - initial vale for the expected reward
  k : int - number of arms of the bandit
  N : int - number of rounds to play

  Returns:
  np.array - Array of the encountered rewards in sequence
  np.array - Array of the expected Q values
  np.array - Array of times, the different levers have been pulled
  '''
```

b) Spielen Sie erneut für $k = 10$ mindestens $N_e = 500$ viele Episoden für die greedy Strategie ($\epsilon = 0$) und bestimmen Sie den durchschnittlichen reward zu jedem Schritt $t = 1, \ldots, 1000$ und zeichnen Sie diesen in das oben erstellte Diagramm mit ein.

c) Verfahren Sie genauso für $\epsilon = 0.01$, $\epsilon = 0.1$ und $\epsilon = 0.5$.

d) Verfahren Sie noch einmal so für die greedy Strategie mit $Q_0 = 5$.

e) Speichern Sie das erhaltene Diagramm (falls nicht übersichtlich genug in 2 Diagrammen).

Aufgabe 4: Konstante Schrittweite

Implementieren Sie eine alternatives Update für die $Q_t(a)$'s bei dem statt dem Faktor $\frac{1}{N_{t+1}(a)}$ eine feste Schrittweite α verwendet wird und testen Sie verschiedene α Werte in der Update Regel (Hinweis: die Schrittweite sollte (viel) kleiner als 1 gewählt werden). Erstellen Sie analoge Diagramme zu denen in Aufgabe 3 mit der konstanter Schrittweite.

Aufgabe 5: Diskussion

Diskutieren Sie Ihre Resultate aus den vorherigen Aufgaben kurz an Hand der erhaltenen Grafik in einem pdf (max 3 Seiten mit Grafik(en)). Diskutieren Sie hierbei die folgenden Punkte ggf. mit eigenen Grafiken

- Warum erhalten Sie das gezeigte Ergebnis für die zufällige Strategie vom Anfang
- Gehen Sie auf das Zusammenspiel zwischen Erkundung (exploration - neue Aktionen wählen) und Ausnutzen (exploitation - die beste verfügbare Aktion wählen) in Hinblick auf die folgenden Fragen ein:
  - Mit welchen Methoden erhalten wir am schnellsten hohen rewards?
  - Welche Methoden liefert auf lange Sicht die besten rewards?
  - Welche Methode scheint insgesamt die besten Resultate zu liefern und warum?

  - Warum liefert die greedy Strategie mit $Q_0 = 5$ besser Resultate als mit $Q_0 = 0$ und was würden Sie erwarten, falls $Q_0 = -5$ gewählt wird?

- Diskutieren Sie kurz, welchen Einfluss eine konstante Schrittweite hier hat - wie verändert sie das Ergebniss und was erhalten Sie, wenn die Schrittweite sehr groß ($\alpha \approx 1$) oder sehr klein ($\alpha \ll 10^{-5}$) gewählt wird und warum?

Aufgabe 6: Extra (optional)

- Bestimmen Sie in allen oben betrachteten Strategien, wie oft in einem Zug über die verschiedenen Episoden hinweg, die optimale Aktion gewählt wurde und fertigen Sie ein ähnliches Diagramm an wie für den gemittelten reward.