MMKI-Praktikum SS06: Aufgabe 3D

Die Aufgaben 3D und 4D schließen thematisch an die Aufgaben aus 2D an. Ziel dieses Aufgabenblattes ist die Anwendung eines maschinellen Lernensverfahrens. Dazu sollen die zwei Teilaufgaben aus Aufgabe 2D mit Hilfe eines solchen selbstgewählten Verfahrens gelöst werden.

Prinzipiell besteht die Lösung jeder Lernaufgabe aus zwei Schritten:

1. Der Lernprozess, in dem das Lernverfahren eine Struktur lernt/optimiert, die die Aufgabe lösen soll.
2. Der Test, bei dem der Controller unter Anwendung der gelernten Struktur die Aufgabe löst.

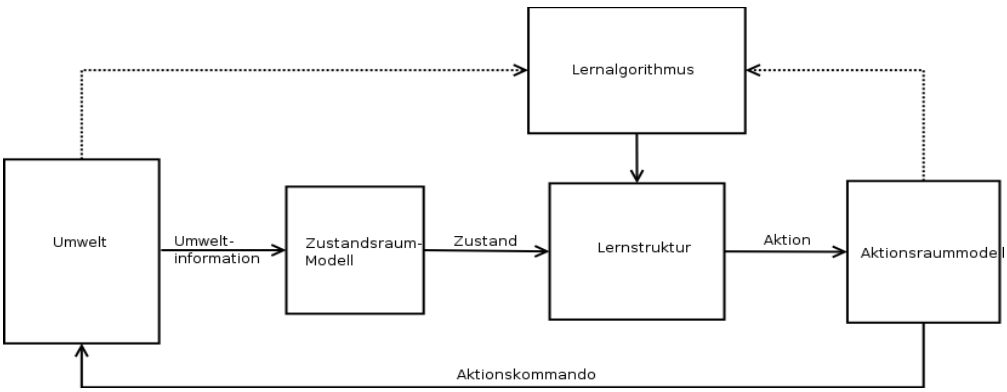
Aufgabe:

a)

- Bringen Sie den Simloid in die bekannte Ausgangslage (alle Gelenke mit dem PID-Conroller fixiert, beide Arme zeigen senkrecht nach unten), sobald alle Gelenke ihre Zielstellung erreicht haben, schalten Sie die Reibung für das rechte Armgelenk (und nur für dieses) mit dem friction-Kommando aus.
- Programmieren Sie einen Controller, der unter Anwendung eines maschinellen Lernverfahrens den rechten Arm des Simloid mindestens einmal bis über die obere Senkrechte hinaus bewegt – also eine Umdrehung schafft. Das Verfahren soll deterministisch funktionieren, d.h. bei wiederholter Ausführung aus der Initialposition in angemessener Zeit gegen die Lösung konvergieren.
- Es gelten dieselben Einschränkungen bzgl. des maximalen torque, wie im letzten Aufgabenblatt – d.h. nach der Initialisierung sollen nur noch (torque 2 <x>) Kommandos abgesendet werden, der Betrag des Torques soll 30 nicht überschreiten.
- Sie können aus zwei Optimierungskriterien auswählen:
  - Minimieren Sie die Zeit bis zum Erreichen der Senkrechten aus der Ausgangslage.
  - Minimieren Sie den maximal verwendeten torque-Betrag.
    (Selbstverständlich sollte das Erreichen des eigentlichen Ziels (Arm ist oben, bzw. Überschlag) noch in vertretbarer Zeit geschehen)

b)

- Die Initialisierung erfolgt analog wie in unter a), mit dem Unterschied, dass der rechte Arm ungefähr senkrecht nach oben stehen soll. Variieren Sie dazu während des Lernprozesses die Initialposition zwischen ca. -5° und +5° um den oberen Totpunkt. (Wer will kann zusätzlich danach auch noch einige Takte (ohne Reibung und mit ausgeschaltetem PID-Controller) warten, damit sich auch eine initiale Winkelgeschwindigkeit einstellt)
- Programmieren Sie einen Controller, der unter Anwendung eines maschinellen Lernverfahrens den rechten Arm des Simloid aus dieser (probabilistischen) Ausgangslage stabilisiert und möglichst ruhig in der Senkrechte balanciert. Die Lernaufgabe gilt als gelöst, sobald der Controller aus beliebiger Ausgangslage des gegebenen Bereiches den Arm senkrecht bewegt und für mindest 10 Simulationssekunden senkrecht hält.
- Es gelten dieselben Einschränkungen bzgl. des maximalen torque, wie im letzten Aufgabenblatt – d.h. nach der Initialisierung sollen nur noch (torque 2 <x>) Kommandos abgesendet werden, der Betrag des Torques soll 30 nicht überschreiten.
- Mögliche Optimierungskriterien sind:
  - Minimieren Sie (nach der Stabilisierungsphase) die maximale Auslenkung aus der Null-Lage.
  - Minimieren Sie die durch den Controller benutzte Energie (Zeitintegral der torque-Beträge)



Komponenten des Lernprozesses

Abgabeformat:

Unabhängig von der Wahl des konkreten Lernverfahrens sind von Ihnen einige Entscheidungen zu treffen, die sie im schriftlichen Teil ihrer Lösung rechtfertigen sollten. (Bitte kurz und wesentlich!)

- Beschreiben Sie kurz ihr Lernverfahren, auch mit Bezug auf die konkrete Aufgabenstellung. Sofern Sie sich für ein besonders neues oder trickreiches Verfahren entschieden haben, erläutern Sie es näher.
- Beschreiben und begründen Sie den von Ihnen modellierten Zustandsraum. Welche Zustandsvariablen (Dimensionen) gibt es, wie werden diese berechnet, wie ist der Zustandsraum aufgebaut? Verwenden Sie einen diskreten Zustandsraum, ist natürlich auch die Form der Diskretisierung wichtig.
- Beschreiben und begründen Sie den von Ihnen modellierten Aktionsraum. Welche Aktionstypen oder konkrete Aktionen gibt es, wie werden diese ggf. in die nativen Simloid-Kommandos umgerechnet? Verwenden Sie einen diskreten Aktionsraum, ist wiederum die Form der Diskretisierung relevant.
- Beschreiben und begründen Sie die von Ihnen durch den Lernprozesses gelernte Struktur. Dies kann zum Beispiel ein Entscheidungsbaum, ein Neuronales Netz, ein Fuzzy-Controller oder im Falle von Reinforcement Learning eine beliebige Repräsentation der Wertfunktion sein (z.B. LookUp-Tabelle).
- Sollten Sie, was durchaus erwünscht ist, verschiedene Verfahren und Modellierungen probiert haben, um zu einer Lösung zu gelangen, beschreiben Sie kurz ihre Herangehensweise und die Auswirkungen ihrer Änderungen.

Um den Lernerfolg zu dokumentieren, sind für beide Aufgabenteile je 2 graphische Plots abzugeben:

- Für die Lernphase ist ein Diagramm zu erstellen, welches ein geeignetes Maß für den Lernerfolg gegen die Zeit oder die Anzahl der Takte oder Episoden aufträgt. Die Zeitskale soll bei Null starten und bis zum Erreichen eines hinreichend stabilen Verhaltens gehen.
- Der gelernte Controller soll unter Anwendung der gelernten Struktur fünf mal ausgehend vom ggf. probabilistischen Initialzustand aufgerufen werden, um die korrekte Lösung zu dokumentieren.
  Für diese Episoden soll jeweils der Gelenkwinkel des rechten Schultergelenks in ein gemeinsames Diagramm gezeichnet werden. (x-Achse ist wieder die Zeit in s oder Takten)

Abzugeben sind ferner die Quellcodes der Programme, sowie zwei Startskripte:

1. Lernphase: Start des Simloid und des Controllers, der die Struktur lernt.
2. Testphase: Start des Simloid und des Controllers mit der gelernten Struktur.

Bemerkung: Lern- und Testcontroller können durchaus auch zwei verschiedene binaries sein.

Hinweise:

Wie in der Übung erwähnt, sind die beiden Aufgaben Standardbeispiele des Reinforcement Learning (RL) und auf diese Weise recht einfach lösbar.

Folgendes sollten Sie aber in jedem Fall feststellen:
Die Größe und die genaue Modellierung des Zustands- und des Aktionsraumes haben signifikanten Einfluß auf den Lernerfolg und die Lerngeschwindigkeit (in beide Richtungen)!
Wenn ihr Lernsystem zwar etwas lernt (konvergiert), aber nicht das, was es soll, ist in aller Regel das von Ihnen gewählte feedback-Signal ungünstig gewählt (z.B. reward beim RL oder die Fitness-Funktion bei EA) → Experimentieren Sie!

Sinvolle Literatur:

- Tom Mitchell, *Machine Learning* : (breit, allgemein und gut verständlich)
- Sutton / Barto, *Reinforcement Learning: An Introduction* : (sehr gut und detailliert)
  - online: http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html
- http://www.cs.ualberta.ca/~sutton/RL-FAQ.html

Weitere Ressourcen:

- Reinforcement Learning Software and Stuff
  - http://www.cs.ualberta.ca/~sutton/software.html
- RL-Demos and Implementation
  - http://www-anw.cs.umass.edu/rlr/domains.html

- RL Toolbox (sehr mächtige C/C++ Bibliothek zum Reinforcement Learning)
  - http://www.igi.tugraz.at/ril-toolbox/general/overview.html
- The Use of **Java** in Machine Learning (Java soll ja kein Nachteil sein)
  - http://www.developer.com/java/other/article.php/10936_1559871_1

DemoClient / RL++:

Um denjenigen, die den Aufwand einer eigenen Implementierung scheuen und die vorerst mit RL-Verfahren arbeiten wollen, einen einfachen Einstieg zu ermöglichen, haben wir den bekannten SampleClient mit Hilfe der bereits erwähnten Bibliothek *RL++* um Lernfähigkeiten erweitert.
Dieser neue SampleClient ist aber wie gesagt nur als Anregung zu verstehen und muss keineswegs benutzt werden!

Zur Benutzung muss zuerst die Bibliothek installiert werden. Es empfiehlt sich, die Installtion ohne root-Rechte in ein gesondertes Verzeichnis (zum Beispiel `~/rlpp` in ihrem home-Verzeichnis), so kann die Bibliothek später durch Löschen des Verzeichnisses einfach deinstalliert werden. Laden Sie sich das entsprechende tar.bz2-File runter und führen Sie folgende Kommandos aus:

```
# tar xjf rl++.tar.bz2;
# cd RLPP;
# aclocal; autoconf; autoheader; automake;
# ./configure --prefix <hier sollte Ihr RLPP-Zielverzeichnis stehen>;
# make; make install;
```

Zur weiteren Benutzung sollte noch eine Umgebungsvariable auf den Ort der Installation gesetzt werden, z.B. in der tcsh:

```
# setenv RL_PP_ROOT <RLPP-Zielverzeichnis>;
```

Zum Übersetzen des Clients steht ein build-Skript im selben Verzeichnis bereit. Das Beispiel benutzt als Lernverfahren „Q-Learning“ und als Repräsentation der ValueFunction bzw. (des Zustands/Aktionsraumes) sogenannte „TileCodings“. Der Client soll lernen, den Arm ausgehend von einer Stellung von 220 möglichst ruhig nach unten zu halten.
Wenn Sie das Beispiel ausprobieren, werden Sie feststellen, dass er in kurzer Zeit ein Verhalten lernt, aber das eigentlich gewünschte Verhalten relativ schlecht umsetzt.
Ein Grund dafür ist, dass hier ausschliesslich die Winkelinformation in den Zustand eingeht. (Mit Sicherheit finden Sie einen geeigneteren Zustandsraum!) Die in der Übung erwähnten Schnittstellen zur Bibliothek sind kommentiert.

Wenn Sie den SampleClient bzw. RL++ verwenden wollen, sollten Sie in jedem Fall die entsprechende Diplomarbeit von Michael Gollin lesen (Kapitel 3 und 4.), um die einzelnen Möglichkeiten und die Bedeutung der Parameter kennenzulernen. Das Mountain-Car- und das Windy-Gridworld-Problem sind bereits als Beispiele in verschiedenen Versionen in der Bibliothek enthalten.

weitere Hinweise:

Wie bereits mehrfach erwähnt, ist das Winkelmodell der Dynamixels unvollständig und unstetig. Insbesondere gibt es einen 60° breiten Bereich, der nur durch die Werte 0 bzw. 1023 beschrieben ist. Je nach Lernexperiment muss dies bei der Modellierung des Zustands- und Aktionsraumes geeignet berücksichtigt werden.

Es ist erlaubt alles zu nutzen, was als Umweltinformation zur Verfügung steht. So ist es zum Beispiel auch möglich, aus den Beschleunigungssensoren Rückschlüsse auf den Zustand des Bioloid oder einzelner Teile des Bioloid zu treffen – und somit den „blinden“ Bereich des Armgelenks auszugleichen.