Praktikum Künstliche Intelligenz

Aufgabenstellung zum 3. Meilenstein
Markov-Ketten und Probabilistik
SLAM

Prof. Dr. Heiner Klocke
Dipl. Inform. Alex Maier
Dipl. Inform. Sascha Schewe

Wintersemester 2014

11 Probalistische Inferenz in Markov-Modellen

Inferenzaufgaben

- **Filtern und Überwachen**. Aufgabe: Belief-State des neuen Zustands berechnen. **A-posterior-Verteilung** über den aktuellen Zustand berechnen, wenn die Evidenzen bekannt sind. Zustandsabschätzung, in welchen Zustand ist die Welt jetzt. $P(X_t | e_{1:t})$

  Abschätzen der Gegenwart $\mathbf{X_t}$:

  Bsp.: Berechnung der **heutigen** Regenwahrscheinlichkeit.

- **Vorhersage**. Blick in die Zukunft $\mathbf{X_{t+k}}$. Abschätzung, was kann in Zukunft passieren. $P(X_{t+k} | e_{1:t})$ $k > 0$

- **Glättung (smoothing)**. Verbessern der Vergangenheit $\mathbf{x_k}$ durch mehr Evidenzen. $P(X_k | e_{1:t})$ $0 \leq k < t$

Filtern

$$P(X_{t+1} | e_{1:t+1}) = f(e_{t+1}, P(X_t | e_{1:t})$$

Agent muss für ein bekanntes Filterergebnis bis zur Zeit t das Ergebnis X_{t+1} aus den neuen Evidenzen e_{t+1} berechnen.

Der Filter-Prozess

$P(X_{t+1} | e_{1:t+1}) = f(e_{t+1}, P(X_t | e_{1:t}))$

$$P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1})$$ Aufteilung der Evidenz

$$= \alpha P(e_{t+1} | X_{t+1}, e_{1:t}) \cdot P(X_{t+1} | e_{1:t})$$ Bayes-Regel

$$= \alpha \overset{②}{P}(e_{t+1} | X_{t+1}) \cdot \overset{①}{P}(X_{t+1} | e_{1:t})$$ Sensor-Markov-Annahme

zweiphasige Vorhersage ①,②

(kausal) (diagnostisch)

weiter rekursiv absteigen

Vorhersage des nächsten Zustandes

→ Aktualisierung mit der neuen Evidenz kommt direkt aus dem Sensormodell.

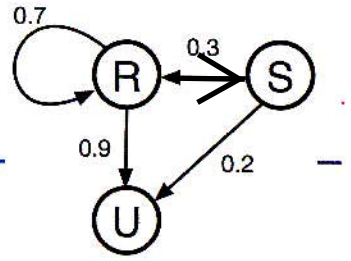
$$P(X_{t+1} | e_{1:t+1}) = \alpha \cdot P(e_{t+1} | X_{t+1}) \cdot \sum_{x_t} P(X_{t+1} | x_t, e_{1:t}) P(x_t | e_{1:t})$$

← Konditionierung des aktuellen Zustands x_t

$$= \alpha \cdot P(e_{t+1} | X_{t+1}) \cdot \sum_{x_t} P(X_{t+1} | x_t) \cdot P(x_t | e_{1:t})$$

hier rekursiv weiter

Filterprozess am Regenschirmbeispiel



Berechne $P(R_2 | u_{1:2})$

Tag 0: Keine Beobachtung, nur a-priori-Glauben: $P(R_0) = \langle 0.5; 0.5 \rangle$

Tag 1: Regenschirm erscheint, $u_1 = \text{true}$.
Voraussage von $t=0$ auf $t=1$

$$P(R_1) = \sum_{r_0} P(R_1 | r_0) P(r_0)$$

Regen / kein Regen

$$= \langle 0.7; 0.3 \rangle \times 0.5 + \langle 0.3; 0.7 \rangle \times 0.5 = \langle 0.5; 0.5 \rangle$$

Aktualisierungsschritt:

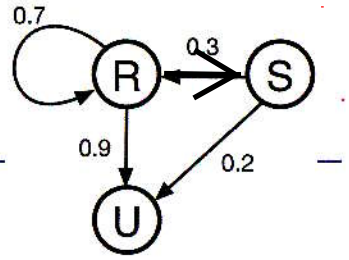
$R_1 = t$ $R_2 = f$

$$P(R_1 | u_1) = \alpha P(u_1 | R_1) \cdot P(R_1) = \alpha \langle 0.9; 0.2 \rangle \langle 0.5; 0.5 \rangle$$

Regen / ¬Regen

$$= \alpha \langle 0.45; 0.1 \rangle \approx \langle \underbrace{0.818}_{= 0.818 \cdot \alpha}; \underbrace{0.182}_{= 0.1 \cdot \alpha} \rangle \qquad \alpha = \frac{1}{0.45 + 0.1} = 1.82$$

Filterprozess am Regenschirmbeispiel (2/2)



Tag 2: Regenschirm erscheint, d.h. $u_2 = \text{true}$

Voraussage: $t=1 \rightarrow t=2$

$$P(R_2|u_1) = \sum_{r_1} P(R_2|r_1) \cdot P(r_1|u_1)$$

$$= \langle 0.7; 0.3\rangle \times 0.818 + \langle 0.3; 0.7\rangle \times 0.182 \approx \langle 0.617; 0.373\rangle$$

Aktualisierung:

$$P(R_2|u_1,u_2) = \alpha P(u_2|R_2) P(R_2|u_1)$$

$$= \alpha \langle 0.9; 0.2\rangle \langle 0.627; 0.373\rangle$$

$$= \alpha \langle 0.565; 0.075\rangle \approx \langle 0.883; 0.117\rangle$$

$$\alpha = \frac{1}{0.565+0.075}$$

Aufgabe 1 (Praktikum) 12.12.14

Führen Sie den Filterprozess bis zum Tag $t=100$ fort.
Natürlich nicht von Hand, sondern mit einem Programm.

1.2 SLAM (*Simultaneous* L*ocalization and* M*apping*)

(„Enorm wichtiger Kontext auf 'Empfehlung' vereinzelter Querulanten entfernt" – zutiefst verletzter Sascha)

Ihre Aufgabe besteht darin,

1. einen Roboter zu bauen der sich frei auf einer 2-Dimensionalen Oberfläche bewegen, seine Umgebung per Abstandssensor ausmessen und seine eigene Ausrichtung ermitteln kann (Vorbedingungen für SLAM).
   Hinweise zur Konstruktion: der Ultraschallsensor sollte möglichst frei drehbar montiert werden, damit der Roboter sich nicht unnötig selbst drehen muss (Bewegungsungenauigkeiten), ein eventueller Kompasssensor sollte mindestens 20cm von jedem Motor entfernt montiert werden (magnetische Interferenz). Ein mögliches, leicht zu erweiterndes Grundmodell findet sich unter:
   http://www.nxtprograms.com/explorer/index.html

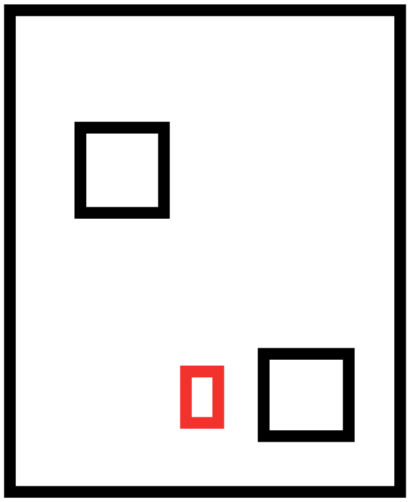
2. diesen Roboter von Ihrem Rechner aus zu steuern, seine Sensordaten auszuwerten, daraus eine Karte (repräsentative 2D-Map, bspw. mit der eigenen Position in rot und derzeit bekannten Umgebungsfeatures in schwarz) zu erstellen und diese, sowie seine eigene Position, während des Mappings darzustellen.
   Hinweise zur Implementierung: Verwenden Sie bei der Implementierung der atomaren Verhalten, wie Messen, Bewegen usw. das Behavior Based Programming:
   http://www.lejos.org/nxt/nxj/tutorial/Behaviors/BehaviorProgramming.htm
   Weitere hilfreiche Klassen und Funktionen finden Sie in den folgenden Packages:
   lejos.robotics.localization
   lejos.robotics.mapping
   lejos.robotics.navigation
   lejos.robotics.objectdetection
   lejos.robotics.pathfinding
   lejos.robotics.subsumption

3. eine kleine, abgegrenzte, unbekannte Umgebung mit Hindernissen möglichst schnell und vollständig zu mappen.

4. ihre Entscheidungen (inklusive Roboterdesign, Algorithmen, eventuelle Filter etc.) zu dokumentieren und zu begründen, ohne in ausschweifendes ''Geschwafel'' zu verfallen.

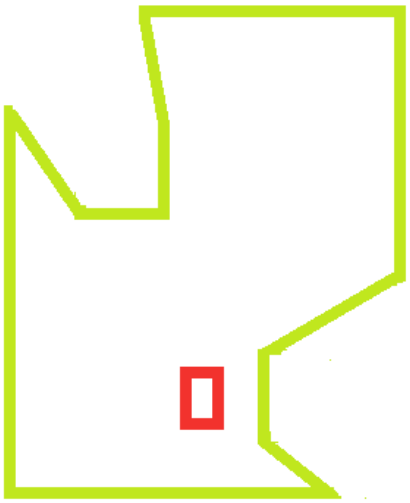
Beispiel:

Reale Umgebung (Künstlerische Darstellung)

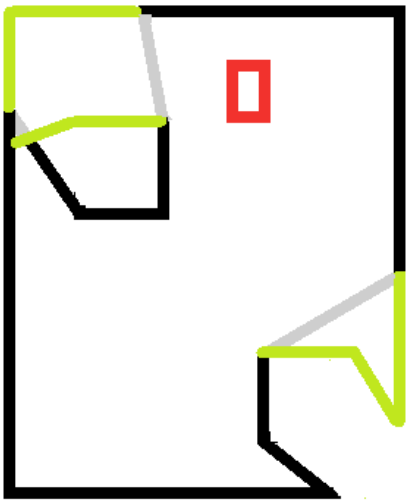


Karte (Ausgangssituation)

Karte (1. Iteration)



Karte (2. Iteration)



... ETC PP