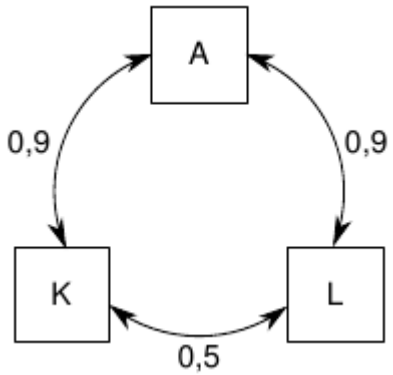
Übungen zur Vorlesung
Grundlagen der Künstlichen Intelligenz
Sommersemester 2016

Institut für Informatik 4
Priv.-Doz. Dr. V. Steinhage
Friedrich-Ebert-Allee 144
53113 Bonn
Email: steinhage@cs.uni-bonn.de
WWW: http://net.cs.uni-bonn.de/ivs/

Blatt 8 (9 Punkte)

Abgabe durch Hochladen auf der eCampus-Seite bis Sonntag, 12.06.2015, 10:00 Uhr, in Gruppen von 2-3 Personen.

Aufgabe 8.1: MDP: Entwurf und Anwendung von *Value Iteration* **$(2 + 2 = 4)$**



Ihr Agent startet in A. In K liegt ein Schlüssel, den Ihr Agent bei einem Besuch mitnimmt. Dieser Schlüssel passt auf das Schloss in L. Gelangt Ihr Agent mit dem Schlüssel dorthin, schließt er das Schloss auf und findet dahinter eine Belohnung von $+1$. Mit Erreichen von L terminiert der Agent. Alle anderen Schritte verursachen jeweils Kosten von $-0,1$. Der Wechsel zwischen zwei Orten gelingt nur mit einer gewissen Wahrscheinlichkeit, die Sie der obigen Zeichnung entnehmen können, im anderen Fall bleibt der Agent im gleichen Ort.

a) Finden Sie eine Modellierung dieses Problems, in der diese Nutzenfunktion separierbar ist und der Nutzen eines Zustands nicht mehr von der vergangenen Historie abhängt. Stellen Sie diese Modellierung als Graph dar, bei dem die Knoten die Zustände und deren Nutzenwerte enthalten. Es sollte also fünf Knoten geben mit den Zuständen $s_0 = (A, -S)$, $s_1 = (K, S)$, $s_2 = (L, -S)$, $s_3 = (A, S)$ und $s_4 = (L, S)$, wobei S beschreibt, dass der Agent den Schlüssel besitzt. Verbinden Sie die Zustandsknoten *vollständig* durch *alle möglichen* Zustandsübergange bzw. Aktionen als gerichtete Kanten. Schreiben Sie die entsprechenden Übergangswahrscheinlichkeiten an alle Kanten.

b) Führen Sie den Algorithmus VALUE ITERATION für *eine* Iteration auf ihrem Entwurf durch: berechnen Sie so die Nutzen $U_1(s_0)$, $U_1(s_1)$, $U_1(s_2)$, $U_1(s_3)$, $U_1(s_4)$ und geben Sie die *policy*$_1$ der Zustände s_0,...,s_3 nach dieser ersten Iteration an.
Hinweis 1: Es gilt $U_0(s_0) = U_0(s_1) = U_0(s_2) = U_0(s_3) = -0.1$ und $U_0(s_4) = +1$.
Hinweis 2: Sollte die Value-Iteration für einen Zustand mehrere Nachfolgezustände mit max. erwarteten Nutzen ergeben, wird die Policy für diesen Zustand enstpr. viele Aktionen vorschlagen (z.B.: *gehe_zu_L* oder *gehe_zu_A*).

Beachten Sie, dass von Aufgaben 8.2 und 8.3 nur eine auszusuchen ist, die abgegeben werden kann/muss. Es wird dementsprechend nur eine Aufgabe korrigiert und bepunktet. Bei Abgabe beider Aufgaben bitte kenntlich machen, welche bewertet werden soll.

Aufgabe 8.2: Entscheidungsbäume: Attributsauswahl (entweder hier $2 + 1 = 3$)

Gegeben sei folgende Trainingsmenge über acht Trainingsbeispielen, die jeweils über die drei Booleschen Attributen A, B und C beschrieben sind, mit ihren Booleschen Antwortwerten.

A	B	C	Antwort
Nein	Nein	Nein	Nein
Nein	Nein	Ja	Nein
Nein	Ja	Nein	Ja
Nein	Ja	Ja	Nein
Ja	Nein	Nein	Ja
Ja	Nein	Ja	Ja
Ja	Ja	Nein	Ja
Ja	Ja	Ja	Nein

a) Erstellen Sie einen vollständigen Entscheidungsbaum nach der in Vorlesung 13 beschriebenen Methode des Decision-Tree-Learning mit schrittweiser Auswahl der Attribute nach größtem Informationsgewinn. Gehen Sie davon aus, dass im ersten Schritt das Attribut A den größten Gewinn (GAIN) hat und somit als erstes Attribut ausgewählt wird. Fahren Sie ausgehend von diesem Punkt fort, den Algorithmus anzuwenden, indem Sie für *jeden* zu bestimmenden Entscheidungsknoten explizit die Gain-Werte für die möglichen Attribute rechnen.

b) Verwenden Sie nun zusätzlich die beiden kombinierten Attribute (B = JA $\wedge$ C = JA) und (A = NEIN $\wedge$ B = NEIN) und geben Sie einen optimalen Entscheidungsbaum an, der *ausschließlich* auf diesen beiden kombinierten Attributen basiert. (Der Baum beginnt also nicht mit der Wahl von A als erstem Attribut wie in Aufgabenteil a).) Sie müssen hier die Gain-Werte für die möglichen Attributauswahlen nicht wieder explizit berechnen.

Aufgabe 8.3: Programmieraufgabe: Entscheidungsbäume (oder hier 3)

Vorbereitung: Laden Sie bitte das ZIP-Archiv decision.zip herunter von unserer eCampus-Seite unter Kursunterlagen >> Python und AIMA Python >> Decision Trees. Das ZIP-Archiv enthält den Ordner *decision* mit dem Skript *decisionS.py* sowie dem vorgegebenen Restaurant-Trainingsdatensatz *restaurant.feat*. Fügen Sie den Ordner *decision* auf der gleichen Ebene wie den Ordner *aima* ein.

Aufgabe: Der Trainingsalgorithmus im vorgegeben Skript *decisionS.py* wählt in der Funktion *choose_attribute* die Attribute auf triviale Weise aus. Bitte erstellen Sie auf der Basis des gegebenen Skripts *decisionS.py* ein neues Skript *decision.py*, welches die Auswahl der Attribute nach Gain und Remainder umsetzt. Bitte erzeugen Sie mit Hilfe Ihres neu erstellten Skripts

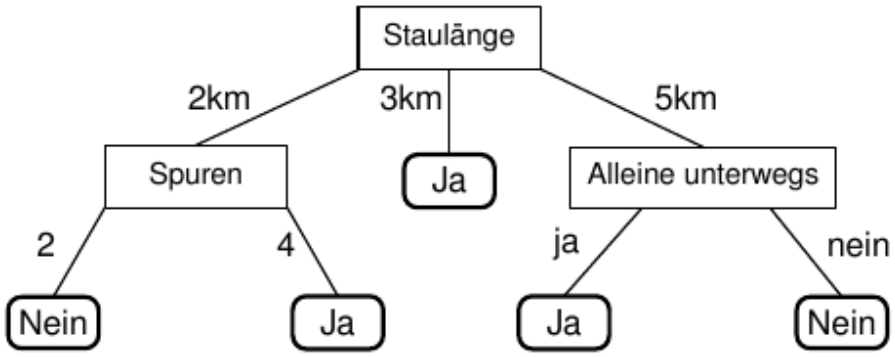
decision.py den Entscheidungsbaum für das Restaurantbeispiel auf der Basis des Restaurant-Trainingsdatensatzes *restaurant.feat* und reichen Sie auf jeden Fall einen Screenshot des erzeugten Entscheidungsbaums ein. Eine Visualisierung des Entscheidungsbaums ist über *Show Decisiontree* der AIMA-Py-GUI erzeugbar.

Aufgabe 8.4: Entscheidungsbäume und -listen **$(1 + 1 = 2)$**

Instanz	Alleine unterwegs	Staulänge	Spuren	Tank voll	Abfahrt ?
#1	JA	5 KM	2	NEIN	JA
#2	NEIN	3 KM	3	NEIN	JA
#3	JA	3 KM	2	NEIN	JA
#4	JA	2 KM	2	NEIN	NEIN
#5	NEIN	5 KM	2	JA	NEIN
#6	JA	2 KM	4	NEIN	JA
#7	JA	2 KM	2	NEIN	NEIN

Stellen Sie sich vor, Sie fahren gerade nach Köln und hören im Radio, dass kurz vor Ihnen ein Stau ist. Wie entscheiden Sie sich, ob Sie von der Autobahn abfahren, um den Stau zu umgehen? In der obigen Tabelle sind einige Attribute aufgeführt, die Ihre Entscheidung beeinflussen könnten, und einige Beispieldaten von verschiedenen (fiktiven) Autofahrern.

Der resultierende Entscheidungsbaum habe folgende Gestalt:



a) Was passiert, wenn Sie Ihren Entscheidungsbaum auf ein Problem anwenden, bei dem die Staulänge 2 km beträgt und es 3 Spuren gibt? Tip: Schauen Sie sich dazu noch einmal genau die vier Fälle an, die im Algm. DECISION-TREE-LEARNING behandelt werden.

b) Konstruieren Sie nun eine Entscheidungsliste für das oben gegebene Problem mit möglichst kurzen Tests (also möglichst kleinem *k* bzgl. der Sprachklasse *k*-DL). Verwenden Sie für Ihre Lösung *Tank voll* als ersten Test. Um Aufzählungen zu umgehen, berücksichtigen Sie auch Tests von Attributwerten über Relationen wie „$<$“, „$>$“, „$\leq$ “ und „$\geq$“ mit Hilfe eines Vergleichswertes (s. Vorl 13, Folie 43) wie z.B. „*Spuren* > 1“.