Übungsblatt 5

Machine Learning (WS 16/17) 13. Dezember 2016

Stefan Edelkamp

Sämtliche Aufgaben sind von der Gruppe selbstständig zu lösen. Die Verwendung von Hilfsmitteln und Quellen außerhalb der Vorlesungsunterlagen gilt es in expliziter Weise zu dokumentieren.
Abgabe ist am Donnerstag, den 12.1.2017.

1. Recherchieren Sie, was ein Partially Observable Markov Decision Process (POMDP) ist! Erklären Sie kurz worum es dabei geht und welches Verfahren vom letzten Übungszettel einen wesentlichen Gedanken des POMDPs teilt! Begründen Sie! (5 P)

2. Bei welchen Problemen/Methoden im Machine Learning kann der EM Algorithmus angewendet werden. Erklären Sie die Anwendungsweise! (5 P)

1 Markov Decision Process (MDP)

Im Zusammenhang mit der Testreihe bezüglich des Blutalkoholwertes mit dem Kommilitonen kam es vor, dass dieser gefährliche Wege aus der Kneipe nach Hause genommen hat. Um sein Leben signifikant zu verlängern, schreiben Sie ihm ein Programm, das einen Markov Decision Process verwendet, um seinen Heimweg sicher zu gestalten. Für einen Prototypen verwenden Sie eine einfache Karte mit der Position des Kommilitonen, seinem Zuhause und der Weser (s. Abbildung 1). Aus einer weiteren lustigen Testreihe haben Sie berechnet, in welche Richtung der volltrunkene Kommilitone laufen möchte im Vergleich zu der Richtung, in die er tatsächlich läuft: Zu 80% läuft er in die korrekte Richtung, zu jeweils 7% in eine Richtung die um 90ř abweicht, und zu 6% in die entgegengesetzte Richtung. Es besteht nicht die Möglichkeit, dass der Kommilitone auf einer Stelle verharrt, es sei denn, er läuft gegen eine Wand, dann bleibt er ganz sicher (zu 100%) an der gleichen Position. Wir unterscheiden die Richtungen Hoch, Runter, Links, Rechts auf der Karte. Die Belohnung ist für jeden Schritt -0.01 (da wir einen kurzen Weg bevorzugen), für das erreichen des Zustands Weser -15, da der Kommilitone das vermutlich nicht überleben wird und die sichere Heimat hat eine Belohnung von 1 (lieber nicht nach Hause finden als in der Weser zu enden).

2	-0.01	-0.01	-0.01	Haus (1)
1	-0.01	X	-0.01	Weser (-15)
0	Kommilitone	-0.01	-0.01	-0.01
y/x-Position	0	1	2	3

Abbildung 1: Karte mit den Belohnungen (Die Positionsangaben sind KEINE Belohnungen!)

1. Implementieren Sie den *Value Iteration* Algorithmus und berechnen Sie über 100 Iterationen über alle Felder den Utility Wert der einzelnen Positionen! (15 P)

2. Geben Sie die beste Aktion für jeden Zustand an! Dokumentieren Sie die Berechnung! (5 P)

3. Geben Sie bis auf Hunderstel genau den größten Discountwert γ an, bei dem sich die Policy ändert! In welchem Punkt ändert sich die Policy zuerst? Erklären Sie das Ergebnis! Nur in diesem Aufgabenteil wird ein Discountwert angenommen. (5 P)

4. Was passiert, wenn Sie die negative Belohnung für den Zustand Weser noch wesentlich verstärken bzw. wesentlich verringern? (5 P)

5. Was passiert, wenn bei einem Reward für die Bahngleise von -1000 als Abbruchbedingung statt 100 Iterationen, die maximale Änderung der Utilitywerte $\delta < 0.02$ bzw. $\delta < 0,002$ gefordert wird? Ändern Sie ihre Implementierung entsprechend, testen sie die beiden Abbruchbedingungen und erklären Sie die Auswirkungen auf die Policy. (5 P)

6. Wie kann *Value Iteration* optimiert werden? Geben Sie zwei mögliche Verbesserungen an und erklären Sie die Vor- und Nachteile. (5 p)

2 Perzeptron

Um einen intelligenten Super Computer zu entwickeln wollen wir statt eines herkömmlichen Arbeitsspeichers ein Neuronales Netz einsetzen. Für die Entwicklungs eines Prototyps verwenden wir das Konzept des künstlichen Perzeptrons um bekannte logische Operatoren zu entwickeln. Das besondere an unserem Super Computer soll sein, dass nicht nur binäre Werte verwendet werden sollen, sondern reel-wertige. Als erstes wollen wir einen Versuch mit der Disjunktion (Oder-Verknüpfung) durchführen. Eine Wahrheitstabelle zu dem herkömmlichen Oder-Operator ist in Abb. 3 zu sehen.

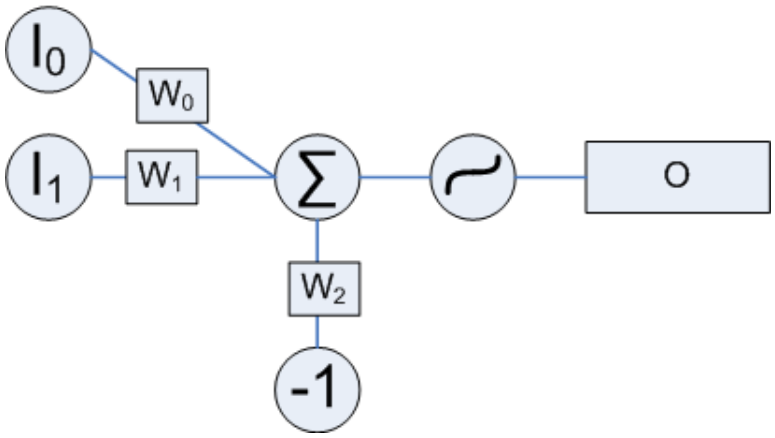

Abbildung 2: Das Perzeptron für den Super Computer

Maschinelles Lernen Übungsblatt 5

I_0	I_1	O
0	0	0
1	0	1
0	1	1
1	1	1

Abbildung 3: Wahrheitstabelle der Disjunktion

$$\sigma(y) = \frac{1}{1 + e^{-y}} \tag{1}$$

$$\frac{d\sigma(y)}{dy} = \sigma(y) * (1 - \sigma(y)) \tag{2}$$

1. Trainieren Sie den *Super Oder Operator*: Verwenden Sie das Verfahren des stochastischen Gradientenabstiegs (Online-Training), um ein Perzeptron mit zwei Eingabeneuronen und einem Bias-Neuron mit dem Wert -1 (s. Abb. 2) entsprechend den Referenzdaten in Abb. 3 zu trainieren! Es soll die Lernrate 10.0 und die initialen Gewichte $w_0 = w_1 = w_2 = 0$ verwendet werden! Dokumentieren Sie jede Gewichtsänderung mit allen Zwischenschritten (inkl. Fehler, Gewichtsanpassung etc.)! Trainieren Sie soviele Epochen, bis die Summe der **quadrierten** Fehler den Wert 0.1 unterschreitet! Als Aktivierungsfunktion soll die Sigmoid (oder auch logistische) Funktion verwendet werden (s. Gleichung 1 und die Ableitung in Gleichung 2)! Erklären Sie das Verfahren und beschreiben Sie jeden Schritt! (10 P)

2. Um die Disjunktion unseres Super-Computers zu testen geben wir als Eingabe (a) $I_0 = I_1 = 0.5$ und (b) $I_0 = I_1 = 0.1$ ein. Zu welchem Ergebnis kommt der *Super Oder Operator* in diesen Fällen? Entsprechen die Ergebnisse Ihren Vermutungen (in Bezug zu den Referenzdaten)? Begründen Sie! (5 P)

3. Erklären Sie den Einfluss der Lernrate auf den Prozess des Gradientenabstiegs! Was passiert bei einer sehr kleinen Lernrate? Welches Problem kann bei einer zu großen Lernrate auftreten? (5 P)