M E N A C E

Wie man einem Computer beibringt, Tic-Tac-Toe zu spielen

Autor: Uwe Härtel
Alle Rechtschreibfehler sind Absicht ;o)

1 Was ist MENACE?

Die Abkürzung MENACE bedeutet **M**atchbox **E**ducable **N**oughts **A**nd **C**rosses **E**ngine. Es handelt sich also offenbar um eine Maschinerie, die mit Hilfe von Streichholzschachteln das Weltweit bekannte Kringel-und-Kreuze-Spielchen erlernen kann.
Erdacht wurde dieses Verfahren von einem gewissen Donald Michie im Jahr 1961, der nach einem zweitägigen Tic-Tac-Toe-Marathon in über 200 Spielen einen fast perfekten maschinellen Meister dieses Spiels erschaffen hatte.
Ich stelle mir es grauenvoll vor, zwei Tage lang am Stück mit knapp 300 Streichholzschachteln permanent Tic-Tac-Toe zu spielen, aber Michie hat mit diesem Experiment einen Grundstein auf dem Gebiet des maschinellen Lernens bzw. später der künstlichen Intelligenz gelegt.
Was mich als absoluten Mathematik-Muffel an diesem System so faszinierte, ist dessen geniale Einfachheit. Es sind keine komplizierten Formelkonstrukte notwendig, die sowieso kein normaler Mensch verstehen würde. Das Prinzip lässt sich meines Erachtens auf jedes Brettspiel anwenden, wenn es nicht zu komplex ist und es lässt sich leicht in jeder beliebigen Programmiersprache umsetzen.

1.1 Funktionsweise

Zunächst einmal wird für jede gültige Spielstellung eine Streichholzschachtel benötigt. In dieser befinden sich verschiedenfarbige Murmeln, welche die leeren Felder des Spielfelds symbolisieren. So würde z.B. also die Schachtel, die das leere Spielfeld am Anfang jeder Partie repräsentiert, 9 unterschiedlich gefärbte Murmeln enthalten.
Zum ausführen eines Spielzuges wird nun zufällig eine Murmel aus der Schachtel genommen und das X bzw. O an die Stelle gesetzt, die dieser Murmelfarbe zugeordnet ist.
Beispielsweise stehe eine rote Murmel für das linke untere Eckfeld. Die Maschine entnimmt selbige also aus der Schachtel und malt ihr X in jene Ecke.

Um später aus der Erfahrung (Stichwort „trial and error“) zu lernen, muss sich die Maschine merken, welche Streichholzschachteln verwendet und welche Murmeln daraus entnommen wurden.
Nach dem Spielende vollzieht sich der eigentliche Lernprozess nach folgendem Prinzip:

Fall 1, die Maschine gewinnt.
Es wird eine weitere Murmel der gleichen Farbe, wie die zuvor gezogene, in die jeweilige Schachtel hinzugefügt. (positive reinforcement)

Fall 2, die Maschine verliert
Die Anzahl der Murmeln der gezogenen Farbe in der Schachtel wird um eine reduziert, d.h. die gezogene Murmel wird einbehalten. (negative reinforcement)

Fall 3, das Spiel endet unentschieden
Die Anzahl der Murmeln bleibt unverändert, die gezogene Murmel wird also lediglich wieder in die Schachtel zurückgelegt.

Was passiert nun im folgenden Spiel? Nehmen wir mal an, die Maschine aus obigen Beispiel hat gewonnen, somit liegen nun 2 rote Murmel in der Zündholzschachtel für das leere Spielfeld. Die Wahrscheinlichkeit, das bei der folgenden Zufallsauswahl wieder eine rote Murmel gezogen wird, hat sich also von 1 : 9 auf 2 : 9 verdoppelt!
Die Maschine hat „gelernt“, das dieser Zug zum Sieg führen kann und wendet ihn immer öfter an.
Und so setzt sich das ganze fort, bis schließlich nach einer gewissen Anzahl Spiele alle Partien unentschieden enden, vorausgesetzt, der menschliche Gegenspieler macht keine Fehler.
Wer mal den 80er Jahre Kultfilm „War games“ gesehen hat, wird mir zustimmen, das der Supercomputer, der ja drauf und dran war, ein Weltweiten Atomkrieg auszulösen, genau nach diesem Prinzip Tic-Tac-Toe gespielt hat.

2 Umsetzung in 3D Tic-Tac-Toe

Natürlich habe ich nicht Zigtausend Zündholzschachteln und Millionen Murmeln in meiner Wohnung angehäuft. ;o)
Der Schachtelstapel für mein Spiel befindet sich in der Datenbank „Menace.mdb“, die man sich z.B. mit MS Access anschauen kann.

Das größte Problem ist, das ich nicht weiß, wie viele Spielsituationen es bei der dreidimensionalen Variante des Spiels gibt. Meine Recherche ergab, das dass „normale“ Tic-Tac-Toe mit 9 Feldern insgesamt 2201 gültige Spielstellungen hat. Dies bestätigte sich auch in meinen Testläufen. Fasst man die auf Grund der Symmetrie existierenden Duplikate durch entsprechende Spiegelungen und Rotationen um die zwei Achsen zusammen, bleiben eben jene rund 300 individuelle Stellungen übrig. (eigentlich mehr, aber da bei Michie die Maschine jedes Spiel beginnen durfte, musste nur für jeden zweiten Zug eine Zündholzschachtel angelegt werden)
Wie verhält es sich nun im 3D-Bereich? 2201 hoch 3 wäre zu einfach oder?? Erste Testläufe ließen bereits nach kurzer Zeit die Anzahl der Streichholzschachteln auf ca. 36000 Stück auflaufen!!
Aufgrund meiner mangelnden Kenntnisse und meiner doch recht massiven Abneigung gegenüber derartiger Mathematischer Probleme, sah ich mich nicht in der Lage, die Anzahl der möglichen Spielstellungen für 27 Felder zu berechnen. Wenn jemand mathematisch bewandert ist und dies Zufälligerweise schon mal Spaßeshalber ausgerechnet hat oder berechnen kann, möge er mir eine Mail schreiben, ich bin jederzeit sehr an diesem Thema interessiert.
Mein Programm berücksichtigt auf jeden Fall die Spiegelung jeder Situation um die 3 Achsen, was ja die unbekannte Anzahl der Spielstellungen zumindest auf 1/8 ihrer selbst reduzieren dürfte. Die Berechnung der virtuellen Rotationen war mir allerdings dann doch zu anstrengend ;o)

2.1 Aufbau der Datenbank

Jede Zeile der Datenbank entspricht einer Zündholzschachtel. Jeder Datensatz enthält die Spalten F1 – F27, welche die Felder des Spielfeldes darstellen. Hier werden also

meine Murmeln in 27 verschiedenen Farben abgelegt – nur das ich auf das Bunte und die Murmeln zu simplen Gewichtungen reduziert habe.
F1 entspricht dem Spielfeld links unten vorne (x = 1, y = 1, z = 1)
F27 entspricht dem Spielfeld rechts oben hinten (x = 3, y = 3, z =3)
Wenn der Computer seinen Zug ausführen möchte, wird zunächst die Datenbank durchsucht, ob bereits eine Streichholzschachtel der aktuellen Spielstellung existiert.
Hierbei muss die Suche insgesamt 4 mal erfolgen, um auch die gespiegelten Abbilder der Spielstellung zu berücksichtigen.
Ist die Suche erfolglos, wird ein neuer Datensatz angelegt, es wird also eine neue Schachtel für die Stellung zum Stapel hinzugefügt.
Das Abbild der Spielstellung wird als einfacher String zusammengesetzt, von links nach rechts, von unten nach oben und von vorne nach hinten.
Wird der Computer fündig, muss er nun eine der virtuellen Murmeln auswählen, um seinen Zug festzulegen. Das geschieht nach dem Prinzip: Schütte erst mal alles auf einen Haufen und such dann zufällig eine aus. Hierzu wird jedem der 27 Felder entsprechend der Gewichtung ein bestimmter Zahlenbereich zugeordnet.
Ein Beispiel:
F1 hat 15, F2 hat 25, F3 hat 0 und F4 hat 10 Murmeln
Summe 50 Murmeln
Nun erfolgt die Klassifizierung:
F1 – Alle Zahlen von 1 bis 15
F2 – Alle Zahlen von 16 bis 40
F3 – nix
F4 – alle Zahlen von 41 bis 50
Hernach muss sich der Rechner lediglich noch eine Zufallszahl zwischen 1 und 50 ausdenken und das entsprechende Feld für den Spielzug kann zugeordnet werden. Deutlich zu sehen ist das der Bereich für F2 der größte ist, somit ist auch die Wahrscheinlichkeit am höchsten, das F2 erneut ausgewählt wird.

Der Computerspieler macht also seinen Zug und merkt sich die Nummer der verwendeten Zündholzschachtel / Zeilennummer in der Datenbank sowie die Nummer des Feldes, auf das er gesetzt hat. Am Schluss des Spiels hat er maximal 14 Schachteln, die er dann mittels des oben beschriebenen reinforcement auffüllt.

2.2 Unterschiede zum originalen MENACE von Michie

Zunächst einmal hatte wohl im Originalversuch die Maschine stets bei jedem Spiel den ersten Zug, so das sie sich nur die ungeraden Züge zum lernen merken musste. Dies halbiert zwar sicherlich die Trainingszeit, jedoch halte ich es einfach für unfair.
Des weiteren habe ich für jedes Feld einen Sicherheitsbestand von einer Murmel festgelegt, um „Irrtümer“ beim lernen zu vermeiden. Eine Wahrscheinlichkeit für ein Feld von 0 bedeutet, das dieses Feld bereits belegt ist und deshalb nicht zur Wahl steht, in allen anderen Fällen kann die Wahrscheinlichkeit zwar sehr klein aber nicht 0 werden.

2.3 Und? Funktioniert das ganze nun?

Ich habe das System ein Training von 50.000 Testspielen durchlaufen lassen. Zu Beginn des Trainings lag die Gewinnquote des Computerspielers bei ca. 50%, er hat also jedes zweite Spiel gegen den Dummy verloren.
Die weitere Entwicklung im Verlauf des Trainings war wie folgt:

Gewinnquote nach 11.000 Spielen 64,1%
Nach 12.000 Spielen 65,4%
Nach 25.000 Spielen 72,3%
Nach 50.000 Spielen 81,0%

Die Zahlen sprechen für sich, MENACE funktioniert ganz hervorragend. Bemerkenswert hierbei ist, das kein einziges von den 50.000 Spielen unentschieden ausgegangen ist. Dies liegt offenbar schon allein an der Tatsache, das die Besetzung des Feldes in der Mitte aller Achsen (X,Y,Z = 2) bereits 13 Gewinnmöglichkeiten offeriert, was ein Blocken eigentlich unmöglich macht. Der Computer bestätigte hat dies bestätigt, die Tests zeigten bereits nach kurzer Zeit ein massives Ansteigen der Wahrscheinlichkeit für Feld F14 und nach 10.000 Testspielen eröffnete er schon fast jedes Spiel mit eben diesem Feld. Die Frage nach dem zwangsläufigen unentschiedenen Ausgang der Spiele lasse ich erst mal offen, vielleicht lässt sich sie sich auch auf mathematischen Wege lösen...
Auf alle Fälle war der Computer nach den 50.000 Spielen extrem Spielstark geworden, eigentlich war er beinahe unschlagbar, zumindest, wenn er beginnen durfte.

3 Weiterführende Links

Wer sich mit dem Thema MENACE bzw. maschinellem Lernen auseinander setzen möchte, sollte den einen oder anderen Blick auf diese Webseiten werfen, die mir auch bei meiner Recherche sehr geholfen haben:

http://www.everything2.com/index.pl?node=menace

http://www.adit.co.uk/html/menace_simulation.html

http://www.ke.informatik.tu-darmstadt.de/lehre/ws05/mldm/rl.pdf
(sehr schöne Erklärung, aber merkwürdiges Deutsch-Englisch-Gemisch)

http://www.vb-helper.com/links_artificial_intelligence.html
(Downloadmöglichkeit für verschiedene Simulationen)

„Lernen, Lernen, nochmals Lernen!“
(Lenin)