Pacman Labboek

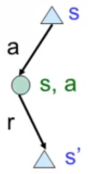
Evelyne van Oers, Ruben van Heusden, Daniël Groen, Daniëlle Bijl

June 2016

1 Introductie

In dit project zal de computer leren hoe het pacman efficient kan spelen, met gebruik van machine learning.

Onze hypothese is dat met behulp van Q-learning, de computer kan leren om pacman efficient te spelen. Q-learning maakt gebruik van twee matrices: een Q-matrix en een Reward matrix. De reward matrix staat voor de reward r die een agent krijgt voor het uitvoeren van een actie a van één staat s naar de andere staat s'. De Q-matrix is het geheugen van de agent, die na iedere reward voor een actie geupdated wordt met kennis.



De gebruikte software is python 3, in een 32 bit linux ubuntu virtual machine.

Het project is opgedeeld in een aantal stappen:

1. het maken van een kleine pacman omgeving
2. het implementeren van q-learning
3. het maken van een volledig pacman bord waar pacman kan leren

In eerste instantie moet een kleine omgeving gebouwd worden die de regels van pacman hanteert. Dit houdt in: pacman en spookjes kunnen niet door muren lopen, aangeraakt worden door een spookje is nadelig, en het pakken van voedsel is voordelig.

De tweede stap is het implementeren van Q-learning in de kleine omgeving. De hypothese is dat Q-learing goed werkt voor een pad-probleem zoals pacman, maar dat het lastiger zal worden naarmate het bord groter wordt. Het idee is dus om het bord stap voor stap uit te breiden.

De laatste stap is het maken van een volledig pacman bord met alle tegenstanders, waar de Q-learning op toegepast wordt. De hypothese is dat wanneer alles werkt op kleine schaal, dit met een aantal aanpassingen ook op grote schaal zal werken.

2 Voortgangsrapportage

Maandag 20 Juni

Op maandag hebben wij in python de benodigde code voor pacman gemaakt, en een klein 2 bij 4 bord voor het uitvoeren van tests. Classes voor alle onderdelen in pacman zijn gemaakt met definities van wat deze onderdelen kunnen doen:

- Pacman
- Ghosts
- Dots
- Achtergrond zoals paden en blokjes

De ghosts bewegen richting pacman door de afstand tussen pacman en henzelf te schatten en deze te minimalizeren. We wilden de beweging van de ghosts gericht maken, maar ook simpel. Hierom hebben we ervoor gekozen dat de ghosts hun afstand tot pacman berekenen met behulp van pythagoras en deze afstand willen verkleinen. Dit werkt in vrijwel alle gevallen naar behoren.

We hebben ons ook verdiept in Q-learning, en hebben dit toegepast op het kleine 2 bij 4 bord met het gebruik van een tabel met Q-waardes.

Woensdag 22 Juni

Vandaag hebben we ons verder verdiept in Q-learning. We zijn van mening dat Q-learning met gebruik van een tabel niet practisch zal zijn om pacman te spelen op een redelijk speelveld. Een redelijk speelveld is een speelveld met de volgende specificaties:

- Een veld van $3 * 3 = 9$ vakjes

- Één ghost als tegenstander
- Minimaal één dot dat pacman op moet pakken om te winnen.

We hebben het hier al over meer dan 9^2 states waar een tabel voor gemaakt moet worden met waarden. Dit achtten wij niet practisch. In plaats van het opstellen van een grote tabel gaat onze interesse uit naar approximate Q-learning, waarbij niet de states, maar specifieke features tijdens het leren in acht genomen worden. Een feature is een bepaald kenmerk in states van een wereld die relevant kunnen zijn voor de agent om zijn doel te bereiken. Approximate Q-learning updated de weging van een bepaalde feature om de Q-waarde te berekenen zodat deze zo dicht mogelijk bij de echte reward waarde komt te liggen. Het voordeel hiervan is dat pacman in verschillende contexten een 'goede' state van een 'slechte' state kan onderscheiden door kennis te gebruiken over gelijksoortige states.

De gekozen features zijn:

- De afstand tot de dichtst bijzijnde dot
- De afstand tot de ghost.

Er is gekozen voor deze features om het probleem simpel te houden. Ook zijn wij van mening dat voor de kleine schaal van het bord dat gebruikt word, meerdere features niet nodig zijn.

Donderdag 23 Juni

Op deze dag hebben we het grotere 3 bij 3 bord gemaakt en de features geïmplementeerd.
Voor de closest dot feature hebben we besloten breadth first search te gebruiken om dichtst bijzijnde dot te vinden. Anders vind je niet per sé de beste dot. Met breadth first search vindt het op de snelste manier de dichtst bijzijnde dot, door vanaf pacman de aangrenzende vakjes te onderzoeken.

Voor de ghost distance wordt pythagoras gebruikt om de afstand van pacman tot de ghost te leren.

Met Q-learning moet pacman leren hoe voordelig het is om naar voedsel te lopen, en welke afstand tot een ghost houden voordelig is.

Breadth first search was goed in verschillende paden vergelijken: echter konden wij de lengte van de paden niet krijgen. Dit was een groot probleem, want voor de feature waarde mogen de pad waardes niet te veel schommelen. De pad

lengte werd gebruikt om voor een lineair verband te krijgen tussen de waarden van een pad.

Dit probleem hebben wij opgelost door het invoegen van een method die het pad langs gaat met breadth first search, en zo de beste volgende state vindt. Deze beste volgende state wordt opgeslagen in het solution path, tot het doel bereikt is.

Vrijdag 24 Juni

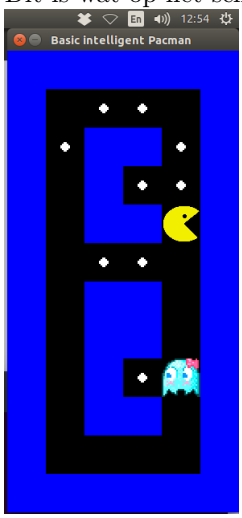
Vandaag hebben we nog gekeken naar de mogelijkheid om onze code aan te passen zodat de beslissingen van pacman gemaakt worden per choice point in plaats van per node. Dit zou een hoop schelen in de berekeningen die gemaakt worden. Als de hoeveelheid berekeningen geminimaliseerd worden, is de verwachting dat onze code ook op het originele, grote pacman bord kan werken. Hier is echter nog meer tijd voor nodig.

3 Discussie

De resultaten van dit project zijn goed, al zou het beter zijn geweest als we aan het begin meer informatie op hadden gezocht voor we begonnen. Dit omdat er dan misschien tijd was geweest om uit te vinden hoe we de machine learning onafhankelijk maken. Aan het eind van dit project werkt alles naar behoren. We werken echter wel met gehardcode feature weights. De bedoeling met approximate Q-learning is dat deze feature weights na veel leren door de machine juist afgestemd worden, maar er was geen tijd meer voor om dit te implementeren. Dit zou kunnen komen omdat wij het werk onderschat hebben. We zouden waarschijnlijk nog een extra week nodig hebben om meer over approximate Q-learning te weten te komen, en hoe we dit implementeren in onze pacman environment. Zowel de features automatisch door machine learning optimaal af laten stellen, alsmede pacman op het originele, grote bord te laten werken.

4 Appendix

Dit is wat op het scherm te zien is:



De code kan gedownload worden van:
http://www.mediafire.com/download/kuby172d2b92og9/Pacman_Code.py

http://www.mediafire.com/download/nkc82l0bjzbbxox/q-learning.zip

Of op de algemene site:
http://www.evevon.github.io