Inleveropgave 1: Model-based Prediction and Control

Planning: een computationeel proces waar we met behulp van een model naar een betere policy zoeken.

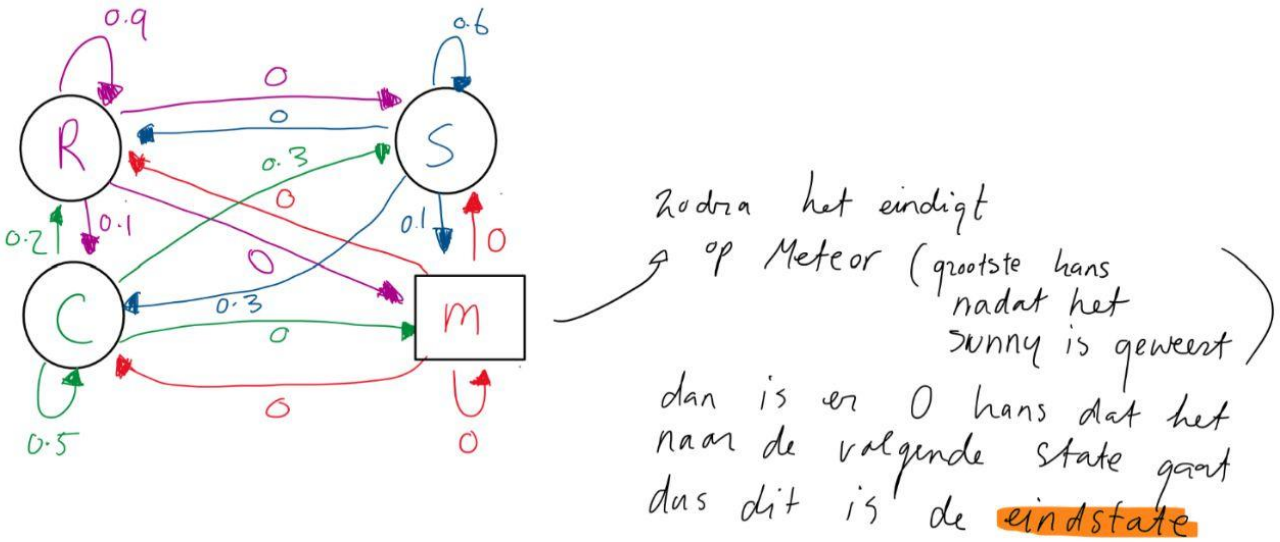
- Bestaat uit twee *deelproblemen:* **Prediction** en **Control.**
- Aan de basis van *Reinforcement Learning* staat het ***MDP*** (Markov Decision Process); het raamwerk voor het beschrijven van de problemen die de agent moet gaan oplossen

Policy: hoe de agent in zijn omgeving handelt.

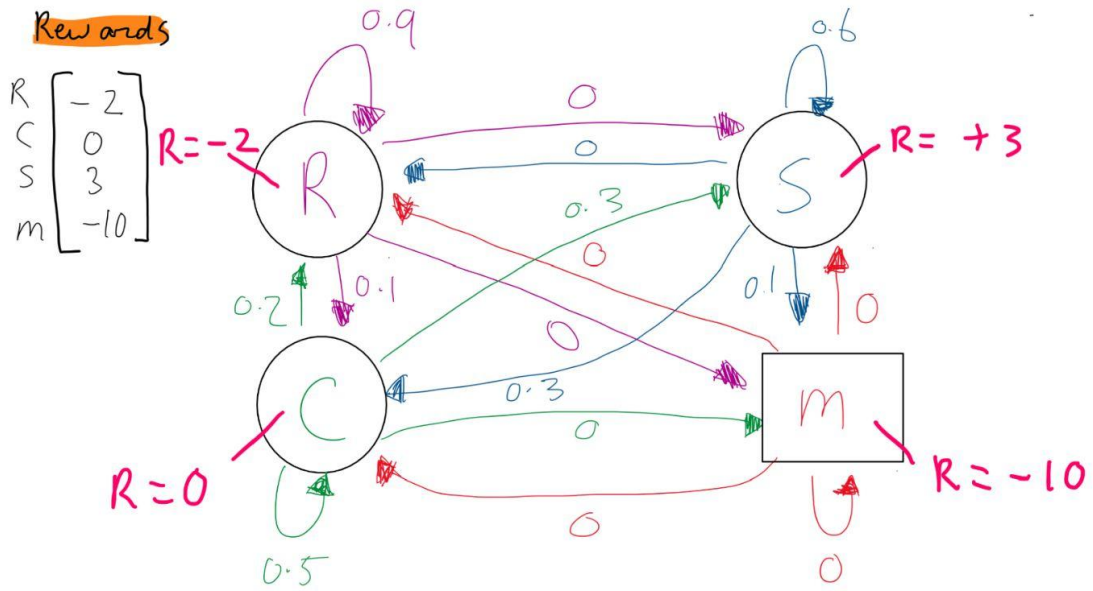
1. Prediction

Prediction gaat over het bepalen van de value-function van een MDP.

1.1. Markov Chain (discrete time)



1.2. Markov Reward Process



1.3. Sampling: Een voorbereiding voor Monte-Carlo Policy Evaluation

Sample 1:

$Y = 1$

C → S(+3) → S(+3) → M (-10)

$G(t) = 0 + (3 * 1) + (3 * 1^2) + (-10 * 1^3) = -4$

Sample 2:

$Y = 1$

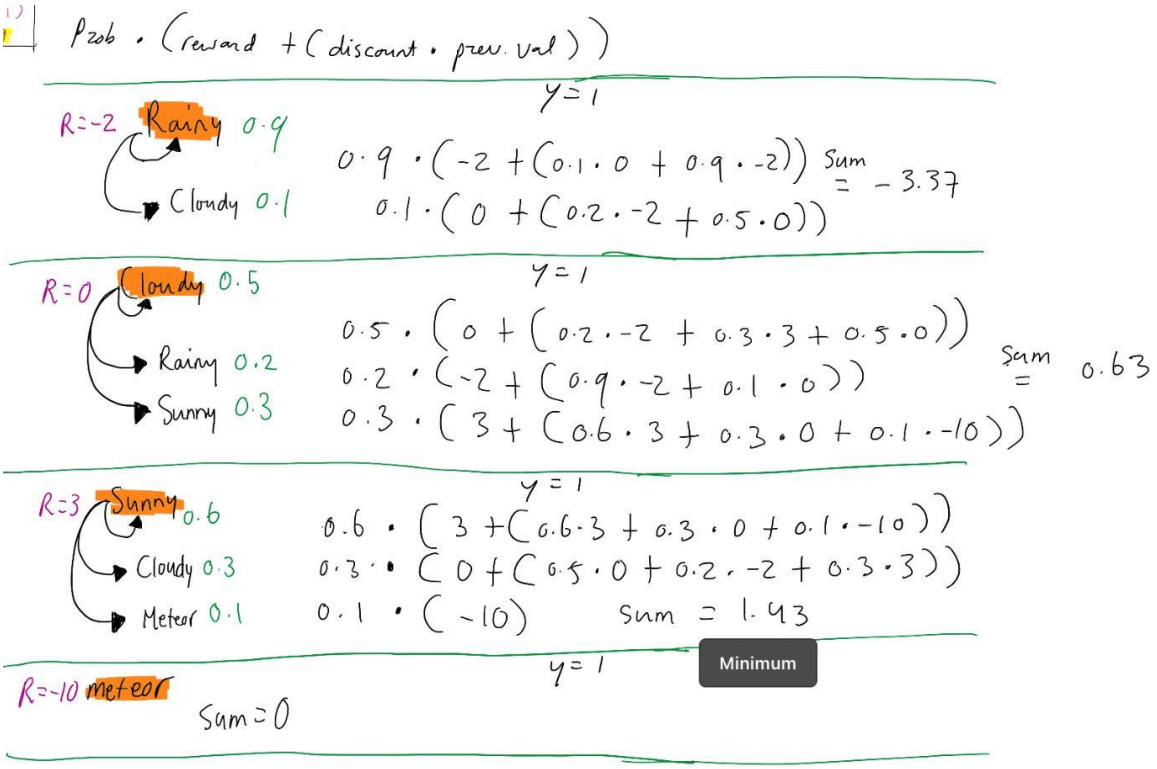
R → R(-2) → C → S(+3) → S(+3) → C → S(+3) → M(-10)

$G(t) = -2 + (0 * 1) + (3 * 1^2) + (3 * 1^3) + (0 * 1^4) + (3 * 1^5) + (-10 * 1^6) = -3$

1.4. De value-function bepalen

- Every state has an entry **V(s)**
- Or every state-action pair s,a has an entry **Q(s,a)**
- Estimate the discounted return for S(t)=s, A(t)=a

V(s)= max a (R(s,a) + y V(s'))
S = {R, S, C, M}
A = {...}



1.5. Zelf onderzoek

- Hoe dichter gamma (discount) bij 1 ligt, hoe dichter de policy zal liggen bij een policy dat de winsten over oneindige tijd optimaliseert. Aan de andere kant zal de waarde-iteratie langzamer convergeren.
- Gamma 1 geeft altijd 'dezelfde' reward terug wat niet optimaal is.

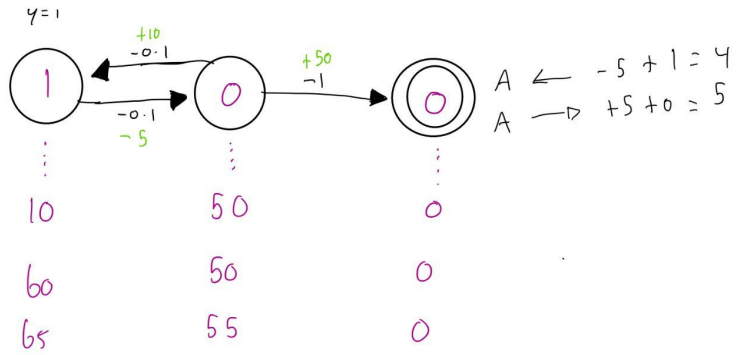
Als y=0 dan is de policy greedy(kiest beste actie voor huidige situatie)

2. Controle met Value Iteration

Bepaal de utility van elke state (value function) van onderstaande MDP, je mag zelf kiezen wanneer je stopt met itereren maar beargumenteer waarom. Te vroeg stoppen is niet goed. Neem een discount factor van γ=1

Value = 0

(vrij onduidelijk)



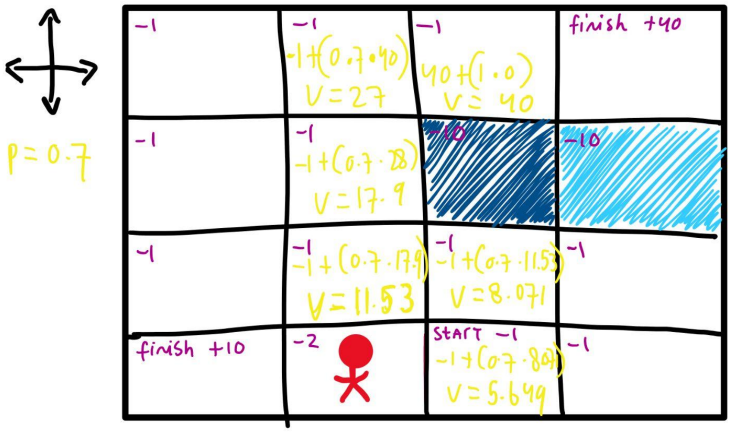
3. Implementatie

Implementeer de doolhof en een agent die op elk legaal veld in de doolhof kan staan en de acties onder, boven, links en rechts kan uitvoeren. De kans dat je de actie uitvoert die je van plan was is altijd **0.7**. Alle andere richtingen op zijn uniform verdeeld over de resterende 3 richtingen. Als je tegen een muur aanloopt blijf je op dezelfde positie staan.

3.1. De betoverde doolhof

(Met de hand uitgewerkt, een mogelijke scenario)

Vanaf het startpunt is dit UP, LEFT, UP, UP, RIGHT, RIGHT




```
-1  -1  -1  40
-1  -1 -10 -10
-1  -1  -1  -1
10  -2  -1  -1
```

3.2. Value iteration

Zie Jupyter Notebook