Model-based Prediction and Control

Stan Meyberg
Adaptive systems

April 20, 2021

1 Prediction

1.1 Markov Chain

Teken een Markov Chain van 4 states aan de hand van de state transition probability matrix $< S, P >$ in figuur 1. Er is geen specifieke beginstate. Er is wel een specifieke eindstate (deze is aangegeven met een dubbele ring).

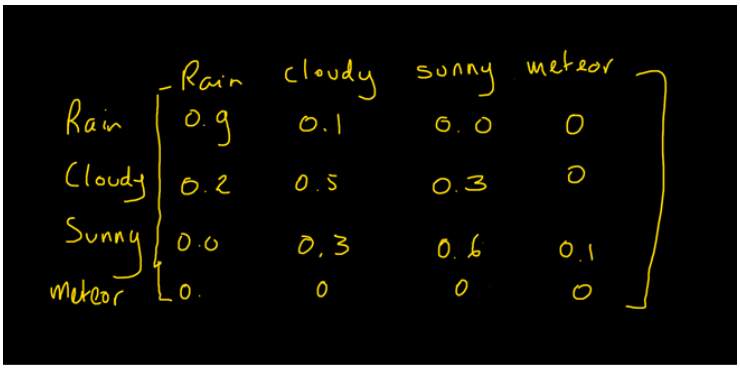

Figure 1: State transition probability matrix met 4 states

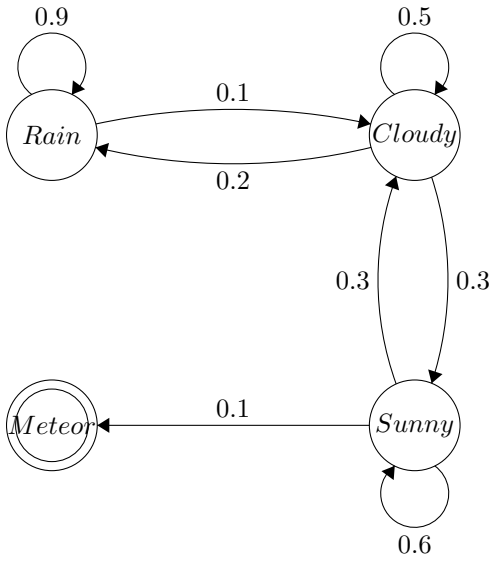

Figure 2: Uitwerking van Markov Chain

1.2 Markov Reward Process

Maak van de Markov Process een Markov Reward Process. De states (van links naar rechts) hebben respectievelijk rewards $[-2, 0, 3, -10]$

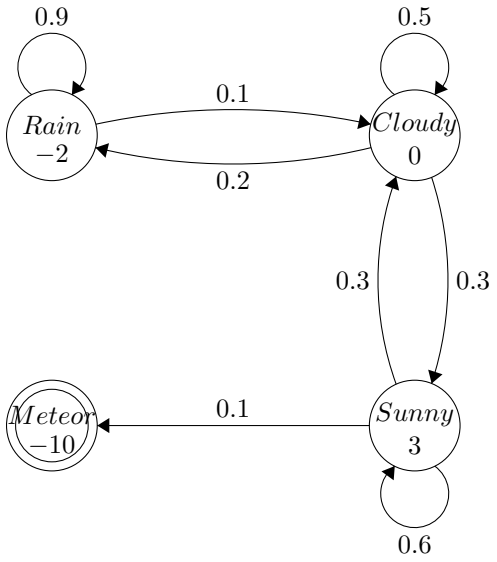

Figure 3: Uitwerking van Markov Chain

1.3 Sampling. Een voorbereiding voor Monte-Carlo Policy Evaluation

Pak twee mogelijke samples van je MRP en leg uit wat de return G_t was voor elke sample. De formule voor de return is als volgt:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Waarbij $\gamma = 1$. Hieronder worden er twee samples weergegeven waarbij voor iedere sample de return wordt berekend.

1.3.1 Sample I

Sample I is als volgt: [Cloudy, Rain, Cloudy, Sunny, Meteor]. De return van deze sample is:

$$\begin{aligned} G_t &= 0 + (1^1 \times -2) + (1^2 \times 0) + (1^3 \times 3) + (1^4 \times -10) \\ G_t &= 0 + -2 + 0 + 3 + -10 \\ G_t &= -9 \end{aligned}$$

1.3.2 Sample II

Sample II is als volgt: [Sunny, Sunny, Cloudy, Sunny, Sunny, Meteor]. De return van deze sample is:

$$\begin{aligned} G_t &= 3 + (1^1 \times 3) + (1^2 \times 0) + (1^3 \times 3) + (1^4 \times 3) + (1^5 \times -10) \\ G_t &= 3 + 3 + 0 + 3 + 3 - 10 \\ G_t &= 2 \end{aligned}$$

1.4 De value-function bepalen

Bepaal nu voor alle states wat de value is na 2 iteraties. De value voor alle states worden geïnitialiseerd op 0. Gebruik hiervoor de Bellman expectation equation met $\gamma = 1$. De Bellman Expectation Equation gaat als volgt:

$$V(s) = E[G_t | S_t = s]$$

Deze kan omgeschreven worden naar:

$$V(s) = E[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]$$

Na de initialisatie zien de states er als volgt uit:

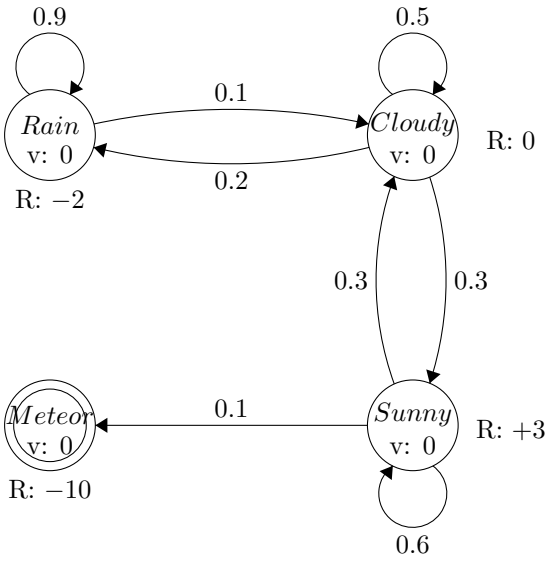

Figure 4: States met values bij initalisatie ($t = 0$)

Met behulp van de Bellman expectation equation kan de value van de states in de volgende iteratie berekend worden:

1.4.1 Iteratie I

$$
\begin{aligned}
v(Rain) &= 0.9 \times (-2 + 1 \times 0) + 0.1 \times (0 + 1 \times 0) = -1.8 \\
v(Cloudy) &= 0.2 \times (-2 + 1 \times 0) + 0.5 \times (0 + 1 \times 0) + 0.3 \times (3 + 1 \times 0) = 0.5 \\
v(Sunny) &= 0.3 \times (0 + 1 \times 0) + 0.6 \times (3 + 1 \times 0) + 0.1 \times (-10 + 1 \times 0) = 0.8 \\
v(Meteor) &= 0
\end{aligned}
$$

Na de eerste iteratie ziet de MRP er als volgt uit:

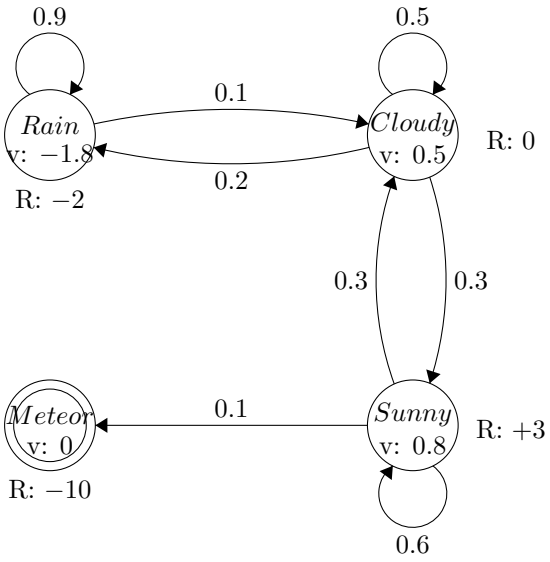

Figure 5: MRP na iteratie 1

1.4.2 Iteratie II

$$
\begin{aligned}
v(Rain) &= 0.9 \times (-2 + 1 \times -1.8) + 0.1 \times (0 + 1 \times 0.5) = -3.37 \\
v(Cloudy) &= 0.2 \times (-2 + 1 \times -1.8) + 0.5 \times (0 + 1 \times 0.5) + 0.3 \times (3 + 1 \times 0.8) = 0.63 \\
v(Sunny) &= 0.3 \times (0 + 1 \times 0.5) + 0.6 \times (3 + 1 \times 2.28) + 0.1 \times (-10 + 1 \times -1) = 1.43 \\
v(Meteor) &= 0
\end{aligned}
$$

Na de tweede iteratie ziet de MRP er als volgt uit:

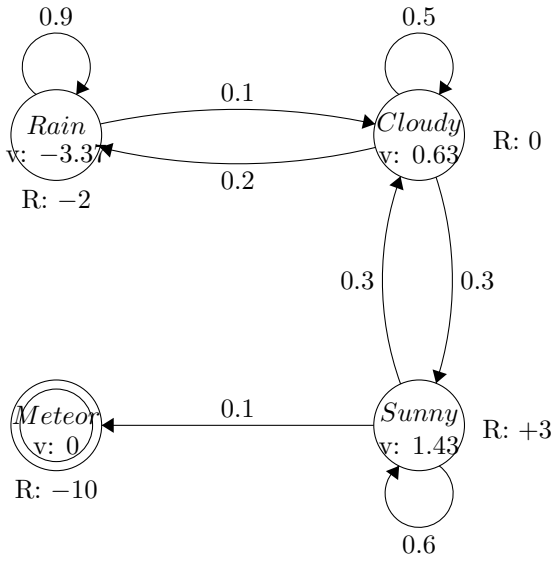

Figure 6: MRP na iteratie 2

1.5 Discount factor

Noem twee problemen die je mogelijk hebt met $\gamma = 1$.

1. Wanneer er sprake is van een cyclische Markov Reward Process, bestaat de kans dat de reward oneindig wordt. Hierdoor kan deze waarde dan ook niet geëvalueerd worden.

2. Met een discount factor van 1 wordt oneindig de toekomst in gekeken en tellen deze rewards dan ook even hard mee. Hierbij kan dus het lange-termijn denken beloond worden. Echter, dit is alleen handig wanneer het model zelf niet veranderd. Wanneer het model kan veranderen heeft dit oneindige vooruit kijken veelal geen zin, omdat het model binnen een aantal stappen weer veranderd kan zijn waardoor de lange-termijn reward niet meer accuraat kan zijn.

2 Control met Value Iteration

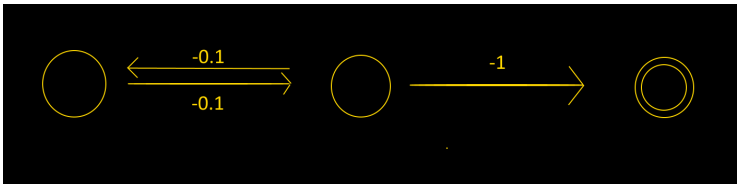

Figure 7: Gegeven MDP

Hieronder zijn de eerste vier stappen van value iteration te zien over de MDP uit figuur 7. Tijdens de iteraties wordt de hoogste utility gepakt van de aanliggende states om te gebruiken in de berekening van de nieuwe utility van de huidige state. In de onderstaande iteraties valt te zien dat na de derde iteratie de values convergeren (wanneer er vergeleken wordt met de vierde iteratie). Om deze reden is ervoor gekozen om te stoppen met itereren na de vierde iteratie.

Iteratie I

$v(s_0) = -0.1 + (1 \times 0) = -0.1$
$v(s_1) = 0.5 \times (-0.1 + (1 \times 0)) + 0.5 \times (-1 + (1 \times 0)) = -0.55$
$v(s_2) = 0$

Iteratie II

$v(s_0) = -0.1 + (1 \times -0.55) = -0.65$
$v(s_1) = -1.0 + (1 \times 0) = -1.0$
$v(s_2) = 0$

Iteratie III

$v(s_0) = -0.1 + (1 \times -1.0) = -1.1$
$v(s_1) = -1.0 + (1 \times 0) = -1.0$
$v(s_2) = 0$

Iteratie IV

$v(s_0) = -0.1 + (1 \times -1.0) = -1.1$
$v(s_1) = -1.0 + (1 \times 0) = -1.0$
$v(s_2) = 0$