Projekt III

Filip Axelsson & Julia Holmgren

2021-02-27

Uppgift 1

I denna uppgift ska en agent ta sig från punkten $(0, 0)$ till $(5, 4)$ på rutnätet $S = \{(x, y) \in \mathbf{N}^2; 0 \leq x \leq 5, 0 \leq y \leq 4\}$. Agenten får endast röra sig i detta rutnät och endast gå i en riktining i taget. Det är även inte tillåtet att gå diagonalt. En action a, det vill säga en rörlse, defineras som en vektor ur mänged $A = \{(0, 1), (1, 0), (-1, 0), (0, -1)\}$. Notera även att vid randen kan inte alla actions utföras. Vid varje nytt tillstånd ges en belöning, med andra ord en reward. Vi ska nu impementera en greedy polciy som väljer nästa action som maximeras nästa förväntade reward.

$$\pi(s) = \underset{a \in A(s)}{argmax}\ r(s, a)$$

I de fallen där flera actions som gerenar samma reward kommer det actions som ändrar postionen i y-led väljas. Funktionerna $A(s)$, $r(s, a)$ är givna där $A(s)$ ger möjliga actions i tillstånd s, $r(s, a)$ ger den förväntade rewarden i tillstånd s givet action a. Policyn implementeras genom att loopa över alla actions som generas utav funktionen $A(s)$, och beräkna respektive förväntade reward. Funktionen plockar sedan ut det action som generade det högsta förväntade rewarden.

Härnäst ska värdefunktionen $V^\pi(s)$ för vår givna policy π räknas ut. Vi definerar först Bellman equation:

$$V^\pi(s) = r(s, \pi(s)) + \sum_{s'} V^\pi(s') p(s'|s, a)$$

Vi kan nu beräkna $V^\pi(s)$ på två olika sätt:
Genom att använda låta $T : \mathbf{R}^{|s|} \to \mathbf{R}^{|s|}$. Det gäller att T^π är en linear contraction så ger detta att T^π har en unik fixpunkt som ges $V^\pi, T^\pi(V^\pi) = V^\pi$.

1. Vi löser evkationsystemet $T^\pi(V) = V$
2. Vi använder oss utav policy evaluation:
   $V_0 \in \mathbf{R}^{|s|}, V_{k+1} = T^\pi(V_k) \Rightarrow V_k \to V^\pi$ när $k \to \infty$

I detta fall kommer vi använda oss utav (2) policy evaluation, genom att implementera en loop över den givna funktionen T_pi. Resultatet som visas i *Table 1* som anger de 10 första värderna för värdefunktionen. I plotten nedanför visas vägen som som agenten tar genom att följa den implementerade policyn.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
##Uppgift 1

## A_s(c(0,0)) = Tillåtna actions i (0,0)
A_s <- function(s) {
  A <- list(
    c(0, 1),
    c(1, 0),
    c(0, -1),
    c(-1, 0)
  )
  not_allowed <- c()
  for (i in 1:4) {
    new_s <- s + A[[i]]
    if (new_s[1] < 0 ||
        new_s[2] < 0 ||
        new_s[1] > 5 ||
        new_s[2] > 4 ||
        identical(s, c(5, 4))) {
      not_allowed <- c(not_allowed, i)
    }
  }
  A[not_allowed] <- NULL
  A
}



## Förväntad reward r(s,a)
r <- function(s, a) {
  new_s <- s + a
  if (identical(s, c(5, 4))) {
    0
  } else {
    prod(new_s)
  }
}

## Övergångssannolikheter p(s'|s,a)
p <- function(new_s, s, a) {
  if (identical(s + a, new_s)) {
    1
  } else {
    0
  }
}



##T_pi: R^s -> R^s, linjär operator för policyn pi. Ger endast EN value iteration
T_pi <- function(pi, v_old) {
  all_states <- tibble(x = 0:5) %>%
    crossing(y = 0:4)
```

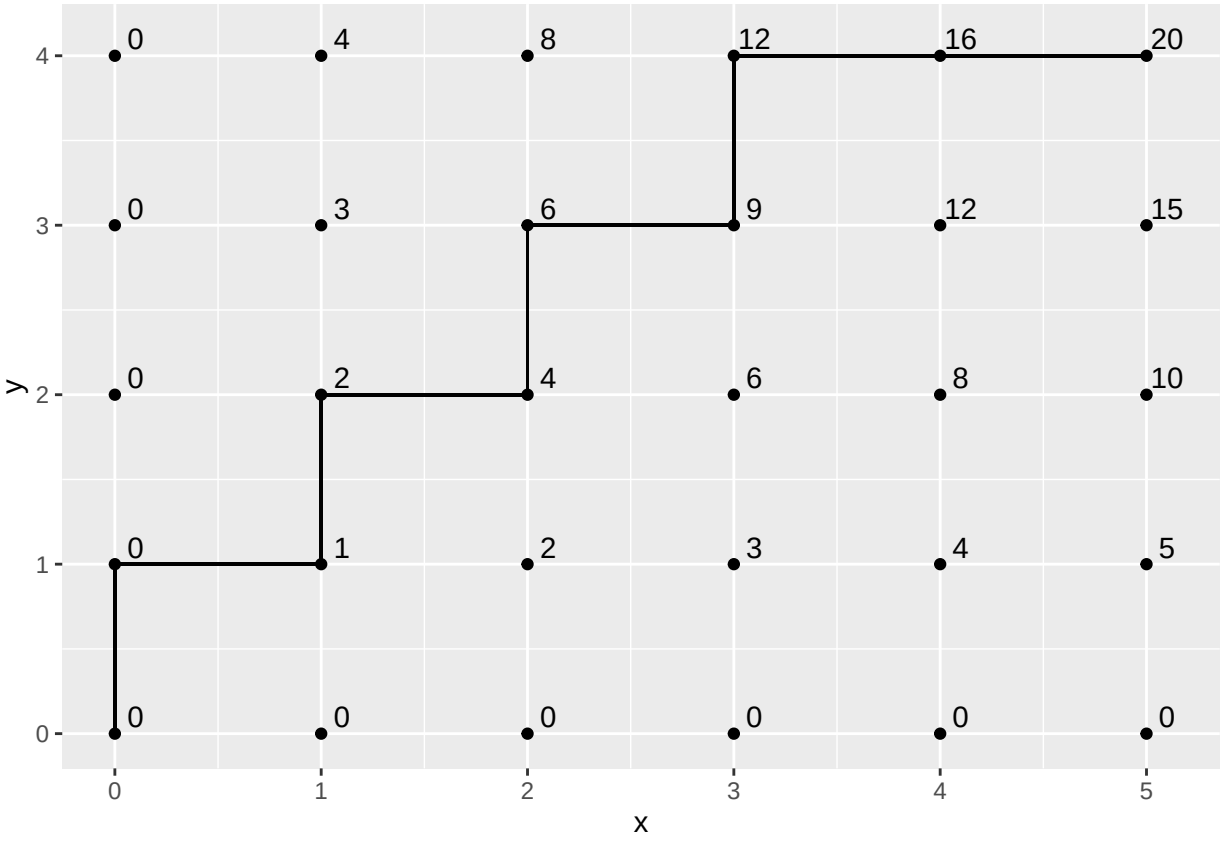
```
  v_new <- c()
  for (i in 1:nrow(all_states)) {
    s <- all_states %>%
      slice(i) %>%
      as.numeric()
    v <- 0
    for (j in 1:nrow(all_states)) {
      s_next <- all_states %>%
        slice(j) %>%
        as.numeric()
      v <- v + v_old[j] * p(s_next, s, pi(s))
    }
    v_new <- c(v_new, v + r(s, pi(s)))
  }
  v_new
}
```

```
policy1 <- function(s) {
  if(s[1] == 5 & s[2] == 4){
    a2 <- c(0,0)
    return(a2)
  }
   actions <- A_s(s)
   r2 = 0
   a2 <- c(0,0)
   for(i in 1:length(actions)){
     a <- as_vector(actions[i])
     r1 <- r(s,a)
   if(r1 > r2){
     r2 <- r1
     a2 <- a
   }
     if(r1 == r2){
       if(a[2]>a2[2]){
       a2 <- a
     }
     }


   }
  return(a2)
}



T_pi_iteration <- function(policy, itterations){
i = 1
v_pi <- rep(0,30)
while(i <= itterations){
v_pi <- T_pi(policy, v_pi)
i = i +1
}
return(v_pi)
}
```

```
v1 <- T_pi_iteration(policy1,30)
```

Table 1: De 10 första uträkningar av värdefunktion i respektive state för policy1

x	y	value
0	0	70
0	1	70
0	2	69
0	3	66
0	4	60
1	0	70
1	1	69
1	2	67
1	3	63
1	4	56



Uppgift 2

Denna uppgift har samma problem som ovan, det vill säga agenten ska ta sig från punkten $(0, 0)$ till punkten $(5, 4)$, dock är det med en annan belöningstruktur. Uppgiften går nu ut på att hitta en optimal policy med avseende på den givna belöningsstrukturen, till exemepl genom value iteration som defineras enligt följande:

$$V_0 \in \mathbf{R}^{|s|}, V_{k+1} = T^*(V_k) \Rightarrow V_k \to V^*, \text{ då } k \to \infty$$

$$Q^*(s,a) = r(s,a) \sum_{s'} + V^*(s')p(s'|s,a)$$

$$\pi^*(s) = \underset{a}{argmax}\ Q^*(s,a)$$

Den arbetsgång som används kan skrivas som följande:

1. Vi formaliserar problemet med en MDP
2. Hitta V^*,Q^* med hjälp utav Bellaman optimality equations
3. Optimal policy $\pi^*(s) = \underset{a}{argmax}\ Q^*(s,a)$

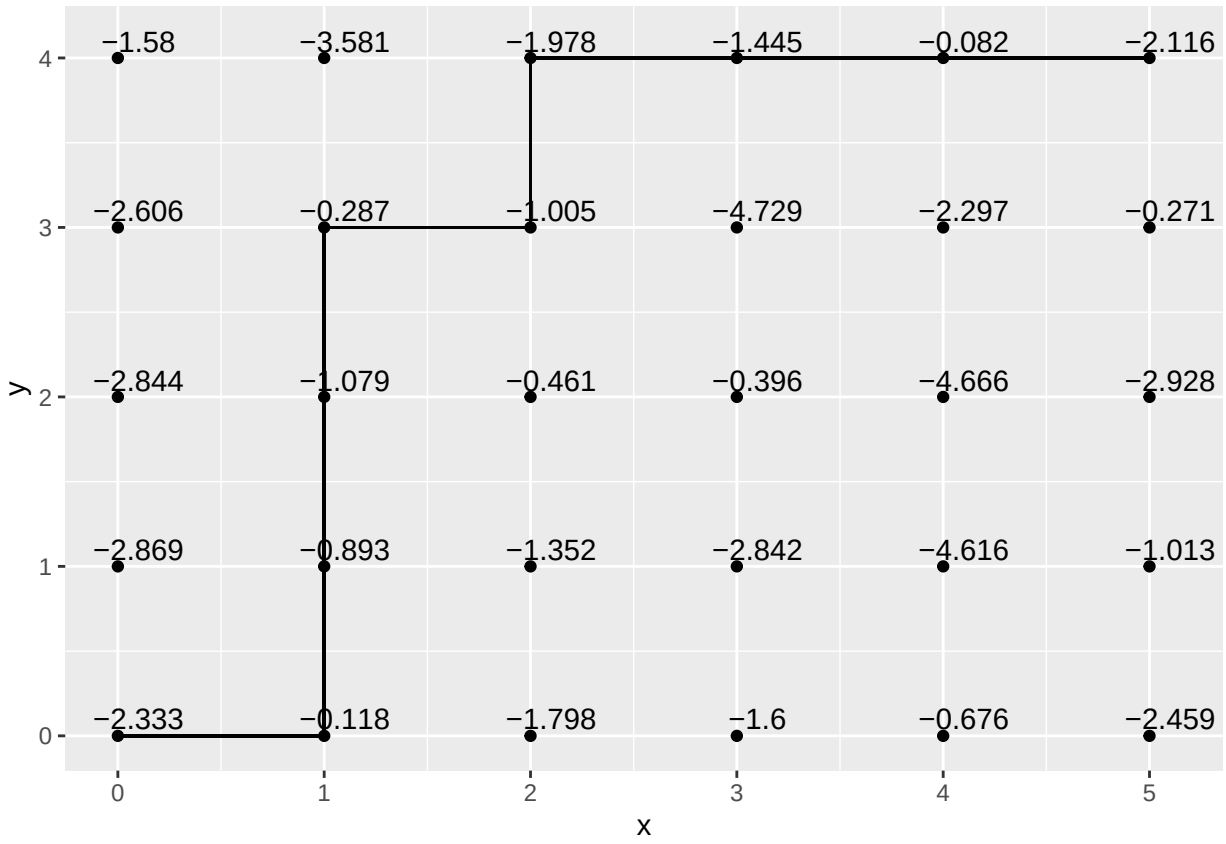
I detta fall tas V^* genom att låta k gå mot oängligheten, detta görs med hjälp av funktionen "T_opti_iteration". Sedan kan den optimala policyn tas fram detta med hjälp utav funktionen "qsa" och "pi_optimal". Nedanför i *Table 2* visas de 10 första värderna för V^* och i figuren nedan visas även agentens rörelsemönster när den optimala policyn används.

```
# skapar V* genom att itterara T_optimal
T_opti_iteration <- function(itterations){
i = 1
v_opti <- rep(0,30)
while(i <= itterations){
v_opti <- T_optimal(v_opti)
i = i +1
}
return(v_opti)
}

v2 <- T_opti_iteration(30)




all_states %>%
  dplyr::select(-reward) %>%
  cbind(v2) %>%
  head(10) %>%
  knitr::kable(
    caption = "De 10 första V* för respektive state",
    col.names = c("x","y","Value")
  )
```

Table 2: De 10 första V* för respektive state

x	y	Value
0	0	-9.002110
0	1	-8.884166
0	2	-7.991383
0	3	-6.912317
0	4	-9.201047
1	0	-8.884166
1	1	-7.991383
1	2	-6.912317
1	3	-6.625765
1	4	-5.620456

```
v_star <- all_states %>%
  cbind(v2)

qsa <- function(s,a) {
  s_new <- s+a
  v3 <- v_star %>%
    filter(x == s_new[1] & y == s_new[2])
  return(r(s,a) + v3$v2 * p(s_new, s, a))
}


pi_optimal <- function(s) {
  if(s[1] == 5 & s[2] == 4){
    a2 <- c(0,0)
    return(a2)
  }
   actions <- A_s(s)
   r2 = -3000
   a2 <- c(0,0)
   for(i in 1:length(actions)){
     a <- as_vector(actions[i])
     r1 <- qsa(s,a)
   if(r1 > r2){
     r2 <- r1
     a2 <- a
   }
     if(r1 == r2){
       if(a[2]>a2[2]){
       a2 <- a
     }
     }


   }
  return(a2)
}
```


−1.58
−3.581
−1.978
−1.445
−0.082
−2.116
−2.606
−0.287
−1.005
−4.729
−2.297
−0.271
−2.844
−1.079
−0.461
−0.396
−4.666
−2.928
−2.869
−0.893
−1.352
−2.842
−4.616
−1.013
−2.333
−0.118
−1.798
−1.6
−0.676
−2.459
y
x
0
1
2
3
4
5