# Inteligentni sistemi 2015-2016

## Seminarska naloga 2

Naloga se izvaja **v parih**. Zagovori bodo potekali v terminu vaj v tednu **18. 1. – 22. 1. 2016.**

Tema druge seminarske naloge je spodbujevano učenje. Želimo naučiti agenta (avtomobil) voziti po večpasovni cesti tako, da ostane na cestišču in se izogne trkom z drugimi vozili. Agent ima na voljo 5 akcij:

| **Akcija** | **Opis** |
|---|---|
| 1 | vozi naravnost (hitrost ostane enaka) |
| 2 | zavij v levo (hitrost ostane enaka) |
| 3 | zavij v desno (hitrost ostane enaka) |
| 4 | povečaj hitrost (vozi naravnost) |
| 5 | zmanjšaj hitrost (vozi naravnost) |

Agentov cilj je prevoziti čim daljši odsek ceste. Vožnja se zaključi, ko se agent dotakne levega ali desnega roba cestišča, trči v drugo vozilo na cesti, ali pa ostane brez goriva.

Na učilnici sta objavljeni datoteki "**simulation.R**" in "**RL.R**". Prva datoteka vsebuje funkcije za učenje agenta ter premikanje in izris objektov na cesti. Te funkcije skrbijo za delovanje celotnega sistema in jih ne spreminjajte (lahko jih popolnoma ignorirate).

Druga datoteka predstavlja glavni program, ki inicializira potrebne spremenljivke, izvede učenje ter sproži simulacijo vožnje z naučeno strategijo. V tej datoteki sta tudi funkciji **getStateDesc** in **getReward**, ki predstavljata vsebinski del seminarske naloge. Vaša naloga je dopolniti implementacijo teh funkcij tako, da bo učenje agenta čim bolj učinkovito.

### *Funkcija getStateDesc*

Funkcija **getStateDesc** se avtomatično kliče ob vsaki spremembi simuliranega okolja. Funkcija prejme podatke o objektih na cesti, ki se nahajajo v vidnem polju agenta (argument ***sceneObjects***). Rezultat funkcije je **opis diskretnega stanja**, ki predstavlja strnjen opis situacije na cesti.

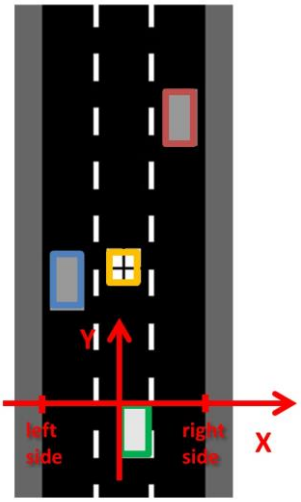
```
getStateDesc <- function(sceneObjects) {
    …

    state
}
```

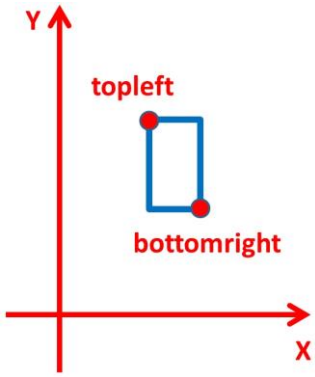
Argument ***sceneObjects*** je dvodimenzionalna tabela (data.frame), ki vsebuje naslednje stolpce: *type*, *xtopleft*, *ytopleft*, *xbottomright*, *ybottomright*, *speed* in *fuel*. Prvi stolpec določa tip objekta in vsebuje naslednje vrednosti:

| Tip objekta | Opis |
|---|---|
| mycar | vozilo, ki ga upravlja agent |
| car | druga vozila na cesti |
| fuel | paket z dodatnim gorivom |
| leftside | levi rob cestišča |
| rightside | desni rob cestišča |

Podatki od drugega do četrtega stolpca določajo položaj objekta na cesti, relativno na položaj agenta (koordinatno izhodišče je postavljeno v zgornji levi rob vozila agenta, kot kaže slika 1). Poimenovanje robnih točk pozicije objekta je prikazano na sliki 2.




Slika 1: Koordinatno izhodišče je postavljeno v zgornji levi rob agenta.




Slika 2: Poimenovanje robnih točk objekta.

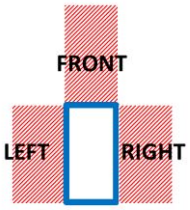
Za situacijo na sliki 1 bi tabela ***sceneObjects*** vsebovala naslednje podatke:

| | type | xtopleft | ytopleft | xbottomright | ybottomright | speed | fuel |
|---|---|---|---|---|---|---|---|
| **1** | mycar | 0 | 0 | 12 | -20 | 5 | 4530 |
| **2** | leftside | -28 | NA | -28 | NA | NA | NA |
| **3** | rightside | 35 | NA | 35 | NA | NA | NA |
| **4** | fuel | -3 | 61 | 9 | 49 | 0 | NA |
| **5** | car | 18 | 117 | 30 | 97 | 4 | NA |
| **6** | car | 24 | 57 | -12 | 37 | 4 | NA |

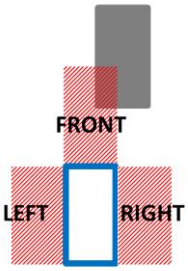
Prva vrstica se nanaša na agenta (vozilo z zelenim robom). Druga in tretja vrstica določata položaj levega oziroma desnega roba cestišča. Četrta vrstica se nanaša na paket goriva (na sliki 1 prikazan kot objekt z rumenim robom). Zadnji dve vrstici opisujeta položaj in hitrost drugih vozil na cesti (na sliki 1 prikazana z rjavim oziroma modrim okvirjem).

Namen funkcije ***getStateDesc*** je, da na podlagi seznama objektov v okolici agenta sestavi opis stanja, ki povzema vso relevantno informacijo o situaciji na cesti. Preprosta začetna ideja je, da v stanje zakodiramo prisotnost drugih vozil v neposredni bližini agenta. Če prostor okoli agenta razdelimo na cone (na primer levo, spredaj, desno), stanje lahko predstavimo kot binarni vektor, ki za vsako cono določa, ali je prazna (vrednost 1) ali se v njej nahaja neko vozilo (vrednost 2). Opis stanja za situacijo na

sliki 3, ko v neposredni bližini agenta ni drugi vozil, bi zapisali kot vektor c(1, 1, 1). Na sliki 4 se pred agentom nahaja drugo vozilo, zato bi v tem primeru stanje zapisali kot c(1, 2, 1).




Slika 3: V okolici agenta ni drugih vozil.




Slika 4: Pred agentom se nahaja drugo vozilo.

Zgoraj predstavljen opis stanja je še vedno nepopoln, saj ne vključuje drugih pomembnih informacij (na primer razdalje do roba cestišča, položaje paketov z gorivom in podobno).

**Vaša naloga je razširiti (ali popolnoma spremeniti) način kodiranja stanj in ga implementirati v funkciji getStateDesc. Množica stanj mora biti končna in diskretna, zato morate upoštevati naslednje omejitve:**

- **vsa stanja morajo biti opisana z vektorjem enake dolžine**
- **vsak element vektorja opisa mora biti pozitivno celo število**

**Zaradi hitrosti in zanesljivosti učenja je zaželeno, da je različnih stanj čim manj!**

## *Funkcija getReward*

Funkcija ***getReward*** se avtomatično kliče med učenjem agenta. Funkcija prejme opis trenutnega stanja (argument ***state***), nazadnje izvedeno akcijo (argument ***action***) ter seznam trkov (argument ***hitObjects***, ki vsebuje oznake tipov objektov, s katerimi je agent trčil ob nazadnje izvedeni akciji; če agent ni trčil z nobenim drugim objektom, je ta vektor prazen). Rezultat funkcije je nagrada (ali kazen), ki jo agent sprejme v opisani situaciji.

```
getReward <- function(state, action, hitObjects) {
    …

    reward
}
```

**Vaša naloga je implementirati funkcijo getReward za smiselno nagrajevanje agenta v fazi učenja. Nagrada mora spodbujati agenta, da izvaja koristne manevre (na primer pobiranje paketov z gorivom) oziroma ga odvračati od negativnih manevrov (na primer trki z drugimi vozili ali padec s cestišča).**

### *Q-učenje*

Ko pripravite implementacije funkcij ***getStateDesc*** in ***getReward***, lahko začnete z učenjem agenta s pomočjo priložene funkcije ***qlearning***, ki se nahaja v datoteki "**simulation.R**". Funkcijo kličemo z naslednjimi argumenti:

```
qlearning(dimStateSpace, gamma = 0.9, maxtrials = 200, maxdistance = 100000)
```

Argument ***dimStateSpace*** je vektor iste dolžine kot opisi stanj, ki jih vrača funkcija ***getStateDesc***. Vsak element vektorja ***dimStateSpace*** določa največjo vrednost, ki jo lahko zavzame istoležni element v opisu stanja.

Ostali argumenti so opcijski in določajo faktor zmanjševanja (argument ***gamma***), maksimalno število poskusov učenja (argument ***maxtrials***) ter najdaljšo prevoženo pot v enem poskusu (argument ***maxdistance***).

Rezultat klica funkcije ***qlearning*** je matrika, ki jo potrebujemo za izvajanje simulacij.

### *Glavni program*

Glavni program začne s klicem funkcije ***initConsts*** za inicializacijo konstant. Funkcija prejme nastavitev za število pasov (argument ***numlanes***, ki je celo število med 2 in 6) in število vozil na cesti pred agentom (argument ***numcars***, ki je celo število med 1 in 10). Nato kličemo funkcijo za učenje agenta, njen rezultat pa podamo kot vhodni argument funkcije za simulacijo dogajanja na cesti.

Na primer, za zgoraj opisani (nepopolni) način kodiranja stanj:

```
initConsts(numlanes=3, numcars=1)
qmat <- qlearning(dimStateSpace = c(2, 2, 2))
simulation(qmat)
```

## Cilj naloge

V okvirju seminarske naloge je potrebno:

- implementirati funkciji ***getStateDesc*** in ***getReward***,
- naučiti agenta voziti v simuliranjem okolju,
- analizirati dosežen uspeh (povprečno prevoženo pot) glede na izbrane parametre (način kodiranja stanja, način podajanja nagrade, število pasov in vozil na cesti, število poskusov učenja, faktor zmanjševanja, …),
- predstaviti ugotovitve v poročilu (dokument v pdf ali doc formatu),
- uspešno zagovoriti nalogo na laboratorijskih vajah.

Na končno oceno seminarske naloge vplivajo inovativnost in elegantnost rešitve, ambicioznost pri raziskovanju problema, argumentacija izbranih postopkov, analiza in predstavitev dobljenih rezultatov.