Praca-licencjacka-chomikuj-17

Uniwersytet Jagiello«ski w Krakowie

Wydzia³ Fizyki, Astronomii i Informatyki Stosowanej

Mateusz Jancarz

Nr albumu: 1107719

Uczenie maszynowe z j¦zykiem Python

Praca magisterska na kierunku Informatyka

Praca wykonana pod kierunkiem
dra hab. Andrzeja Kapanowskiego
Instytut Fizyki

Kraków 2015

1:1130949268

O±wiadczenie autora pracy

wiadom odpowiedzialno±ci prawnej o±wiadczam, »e niniejsza praca dy-plomowa zosta ª a napisana przeze mnie samodzielnie i nie zawiera tre±ci uzy-skanych w sposób niezgodny z obowi¡zuj¡cymi przepisami.

O±wiadczam równie», »e przedstawiona praca nie by ª a wcze±niej przed-miotem procedur zwi¡zanych z

uzyskaniem tytuªu zawodowego w wy»szej uczelni.

Kraków, dnia Podpis autora pracy

O±wiadczenie kieruj¡cego prac¡

Potwierdzam, »e niniejsza praca zostaªa przygotowana pod moim kierun-kiem i kwalikuje si¦ do przedstawienia jej w post¦powaniu o nadanie tytuªu zawodowego.

Kraków, dnia Podpis kieruj¡cego prac¡

1

2:7021885621

Sk²adam serdeczne podzi¦kowania Panu doktoro-wi habilitowanemu Andrzejowi Kapanowskiemu za udzielon¡ pomoc oraz zaanga»owanie w powstanie tej pracy.

3:7534317148

Streszczenie

Niniejsza praca przedstawia aplikacj¦, która symuluje gie²d¦ zak²adów sportowych, oraz gracza-bota wykorzystuj¡cego uczenie maszynowe do zwi¦k-szania skuteczno±ci swojej gry.

Pocz¡tek pracy zawiera wprowadzenie do dziedziny uczenia maszynowego oraz przybli»a jeden z przyk²adów

jego zastosowania w robotyce. Opisane zostaj¡ narz¦dzia wykorzystane w projekcie, w których sk³ad wchodzi j¦zyk Python oraz wybrane biblioteki, takie jak NumPy i Matplotlib.
Omówiono uczenie ze wzmocnieniem, jako nowe podej±cie w obszarze uczenia si¦ maszyn. Opisano wszystkie sk³adowe potrzebne do denicji tego typu algorytmów. Przedstawiono matematyczny model tego paradygmatu, czyli proces decyzyjny Markowa.

Zaprezentowano dwa popularne algorytmy z klasy zwanej metodami ró»-nic czasowych: Q-learning oraz Sarsa. Zostaj¡ one przedstawione w mo»li-we kompletnej wersji uwzgl¦dniaj¡cej wykorzystanie tzw. ±ladów aktywno±ci, które w teorii pomagaj¡ przyspieszy¢ proces nauki.

Stworzono implementacj¦ wymienionych algorytmów, na podstawie któ-rej zbudowana jest symulacja gry na gie³dzie. Przy pomocy przeprowadzo-nych testów sprawdzono skuteczno±¢ zaprogramowanej symulacji. Szczegó³o-wo opisana jest rola oraz dobór warto±ci poszczególnych parametrów nauki w kontek±cie rozwi¡zywanego problemu. Wspomnian¡ aplikacj¦ uzupe³niaj¡ programy pomocnicze s³u»¡ce do wizualizacji danych gie³dowych na wykre-sach oraz generowania sztucznych sekwencji zmian kursu.

S³owa kluczowe: sztuczna inteligencja, uczenie maszynowe, uczenie ze wzmoc-nieniem, metody ró»nic czasowych, algorytm O-learning, algorytm Sarsa, ±la-dy aktywno±ci, proces decyzyjny Markowa

4:2908627240

English title: Machine learning with Python

Abstract

This thesis presents the application simulating sports betting markets and a bot player that performs transactions on markets and utilizes machine learning for improving it's performance.

In the beginning, the introduction to machine learning is provided with examples of it's eective usage in robotics. Development tool set used is presented, which contains Python language, NumPy and Matplotlib modules among the others. Reinforcement learning is presented as a novel approach in machine learning. Mathematic model of the approach is explained with Markov decission process.

Two concrete algorithms of temporal dierence learning paradigm are presented: Q-learning and Sarsa. The description contains possibly complex versions, which contain usage of eligibility traces that, in theory, help to improve learning process eciency.

Implemenation of mentioned algorithms is developed and used to build simulation of trading on betting exchange. Eectiveness of the application is examined via various experiments. The role and selection of all the learning parameter values is described in a context of the problem being solved. The application is supplemented with other tools built that are used to visualize exchange markets data and generate artificial price change sequences.

Keywords: artificial intelligence, machine learning, reinforcement learning, temporal dierences learning, Q-learning algorithm, Sarsa algorithm, eligibi-lity traces, Markov decision process

5:1122368467

Spis tre±ci

7:9025885810

Spis tabel

4

8:3249205555

Spis rysunków

5

9:2901961289

Listingi

6

10:2263799020

1. Wst¦p

Tematem niniejszej pracy jest Uczenie maszynowe z j¦zykiem Python . Praca stanowi wprowadzenie do tematyki uczenia maszynowego jako jednego z fragmentów szerokiego obszaru sztucznej inteligencji. Wspomniany w jej ty-tule j¦zyk Python zastosowano w celu prostego oswajania si¦ z poznawanymi algorytmami.

1.1. Cele i za³o»enia

Celem pracy jest przedstawienie praktycznego zastosowania maszynowego uczenia si¦. Stworzono w jej ramach aplikacj¦, która symuluje gr¦ na gie³dzie zak³adów sportowych. Zaprogramowany gracz-bot przeprowadza transakcje, a dzi¦ki wbudowanemu modu³owi ucz¡cemu stara si¦ w miar¦ zdobywane-go do±wiadczenia zwi¦kszа¢ skuteczno±¢ swojej gry. Ca³o±¢ projektu zosta³a zbudowana przy u»yciu j¦zyka Python.

Celem po±rednim mo»na okre±li¢ faz¦ poznawania mo»liwo±ci uczenia ma-szynowego ze wzmocnieniem oraz implementacj¦ konkretnych algorytmów nauki.

1.2. Znaczenie uczenia maszynowego

Ludzko±¢ produkuje w dzisiejszych czasach ogromn¡ ilo±¢ danych. Dzi¦-ki powszechnemu dost¦powi do sieci Internet tworzenie, gromadzenie oraz przesy³anie danych jest bardzo proste i tanie. Danych ju» jest du»o, a ich ilo±¢ ca³y czas ro±nie. Statystyki podaj¡, »e oko³o 90% z nich zosta³o wy-tworzonych w ostatnich kilku latach [1]. Samo posiadanie nawet ogromnej ilo±ci informacji mo»e dzisiaj nie gwarantowa¢ przewagi konkurencyjnej w biznesie, wi¦c coraz wi¦kszy wp³yw ma ich skuteczne wykorzystanie. Surowe dane s¡ cz¦sto niewiele warte, a liczy si¦ wiedza z nich pozyskana. Fakt ten jest motorem nap¦dowym do rozwoju dziedzin zwi¡zanych z przetwarzaniem ogromnych ilo±ci danych w celu formowania nietrywialnych wniosków z nich p³yn¡cych.

Nic dziwnego, »e o poj¦ciach takich jak data mining, big data czy sztucz-na inteligencja jest coraz g³o±niej. W±ród nich jest równie» termin uczenie maszynowe. Zyskuje ono na popularno±ci zw³aszcza w ostatniej dekadzie, mimo »e zosta³o zaproponowane ju» w roku 1959 przez Arthura Samuela[2]. Zdeniowa³ on uczenie maszynowe jako dziedzin¦ wiedzy, która przekazuje

komputerom zdolno±¢ nauki bez potrzeby zaprogramowania ich wprost . Warto wspomnie¢, »e a» do oko³o po³owy lat dziewi¦¢dziesi¡tych na rynku prawie nie istnia³y przypadki komercyjnego zastosowania uczenia maszynowego [3].

11:4800164380

Pierwszym tego typu projektem uznaje si¦ oprogramowanie ucz¡ce si¦ gra¢ w warcaby stworzone w 1959 przez wspomnianego Arthura Samuela, ówczesne-go pracownika rmy IBM [4]. Dzisiaj zastosowa« jest wiele, poczynaj¡c od systemów rekomendacyjnych w sklepach internetowych, a ko«cz¡c na opro-gramowaniu do przetwarzania j¦zyka naturalnego czy steruj¡cego robotem.

1.3. Potrzeba zastosowania maszynowego uczenia si¦

Sztuczna inteligencja jest szeroko stosowana w robotyce. Wykorzystuje si¦ j¡ do tworzenia mo»liwie autonomicznych jednostek zdolnych do samo-dzielnej adaptacji do ±rodowiska, a tak»e przeró»nych czynników losowych. Cz¦sto za przykªad bierze si¦ zdolno±¢ zwierz¡t do radzenia sobie w przy-padku odniesienia ran. Analogiczna sytuacja mo»e si¦ przydarzy¢ w ±wiecie robotów, dlatego wa»nym jest, aby mogªy si¦ one do nich w odpowiedni sposób dostosowywa¢.

Nie sposób jest przewidzie¢ wszystkie tego typu sytuacje, dlatego opty-malnym rozwi¡zaniem mo»e by¢ przekazanie robotom zdolno±ci do samo-dzielnej nauki radzenia sobie z nimi. Obiecuj¡cym podej±ciem do tego typu przypadków jest uczenie ze wzmocnieniem , czyli nauka na bazie interakcji agenta ze ±rodowiskiem. Czasami okre±lane jest ono jako sztuczna inteligen-cja w mikroskali, poniewa» wykorzystywane w tym paradygmacie algoryt-my musz¡ reagowa¢ w sposób autonomiczny, aby wykona¢ okre±lony cel [5]. Ciekawym przykªadem zastosowania uczenia ze wzmocnieniem jest 6-no»ny robot kroczący, który zostaª zaprogramowany tak, aby radzi¢ sobie z szerok¡ gam¡ urazów mechanicznych [6].

Twórcy robota bazuj¡ na hipotezie, w której zwierz¦ta rozumiej¡ prze-strze« dost¦pnych zachowa« oraz ich potencjaª na podstawie posiadanego do±wiadczenia i podejmuj¡ inteligentne wybory dotycz¡ce prób testuj¡cych te zachowania w celu ich ewaluacji. Zdeniowali oni map¦ zachowa« do war-to±ci, która pozwala przewidywa¢ skuteczno±¢ tysi¦cy ró»nych zachowa«.

Na tej podstawie wymaganiem wst¦pnym w budowie tego rodzaju syste-mu staje si¦ jedynie precyzyjne zdeniowanie sko«czonej przestrzeni stanów i zachowa« oraz okre±lenie celu (w tym przypadku wydajnego poruszania si¦), a tak»e ustalenie strategii do testowania par stan-zachowanie. Dzi¦ki temu robot sam jest w stanie nauczy¢ si¦ takich zachowa«, które s¡ opty-malne do osi¡gni¦cia celu. Nie zmienia si¦ to nawet w przypadku urazów mechanicznych, bo w razie ich wyst¡pienia robot sam dostosowuje sposób chodzenia, aby zrekompensowa¢ poniesione straty. Schemat to obrazuj¡cy zostaª przedstawiony na rysunku 1.1.

Rysunek 1.1: Schemat testowania ró»nych zachowa« przez robota kroczi¡ce-go.

ródªo: [6]

1.4. Zawarto±¢ pracy

Rozdziaª 1 zawiera wprowadzenie do tematyki opisywanej w pracy magi-sterskiej. W rozdziale 2 opisano podstawowe narz¦dzia wykorzystane do stwo-rzenia projektu praktycznego. Krótkie wprowadzenie do uczenia maszynowe-go i uczenia si¦ ze wzmocnieniem zawarto w rozdziaªach 3 i 4. Projektowanie aplikacji przedstawiono w rozdziale 5. Rozdziaª ten zawiera równie» opis do-meny, w której zastosowanie ma tworzona aplikacja, czyli gieªdy zakªadów sportowych. Pod koniec tej cz¦±ci pracy przedstawiono wyniki testów zapro-gramowanej symulacji. Rozdziaª 6 stanowi podsumowanie wyników pracy. W dodatku A jest zawarty szczegóªowy opis konguracji i sposobu uruchamiania projektu, dodatek B natomiast zawiera kod [1]ródªowy zaimplementowanych algorytmów.

Literatura z dziedziny maszynowego uczenia si¦ jest bardzo bogata. Do-brym wprowadzenie do tematu jest ksi¡»ka Cichosza [7]. Zastosowanie Pytho-na do budowy systemów ucz¡cych si¦ (ale bez uczenia si¦ ze wzmocnieniem) jest opisane przez Richerta i Coelho [8]. Przewodnikiem po temacie uczenia si¦ ze wzmocnieniem mo»e by¢ ksi¡»ka Suttona i Barto [9].
W projekcie praktycznych zastosowano nast¦puj¡ce narz¦dzia 1:
Python 2.7 [10],
NumPy 1.9 [11],

Matplotlib 1.4 [12].

1Podane wersje byªy wykorzystywane w procesie implementacji. Aplikacja zostaªa przetestowana równie» na starszych wersjach, które wymieniono w sekcji A.1.

13:7413238631

2. Opis wykorzystywanych narz¦dzi

Ka»dy wi¦kszy projekt programistyczny bazuje na pewnym zestawie go-towych narz¦dzi takich jak biblioteki, modu³y, standardy protoko³ów itp. W tym rozdziale zostan¡ opisane podstawowe narz¦dzia informatyczne przydat-ne przy pracy nad projektem z dziedziny uczenia maszynowego. Punktem wyj±cia b¦dzie oczywi±cie sam j¦zyk Python.

2.1. Wprowadzenie do j¦zyka Python

Na wst¦pie tego rozdzia³u warto wyja±ni¢ powód wyboru g³ównego narz¦-dzia znajduj¡cego si¦ w tytule pracy, czyli j¦zyka Python. J¦zyk ten nie bez powodu jest okre±lany jako wykonywalny pseudokod . Kod w nim zapisany jest czytelny, a dzi¦ki niewielkiemu narzutowi sk³adniowemu relatywnie zwi¦z³y. Spora cz¦±¢ podstawowych elementów j¦zyka jest wbudowana w jego sk³ad-ni¦, a przyk³adem mo»e by¢ tutaj inicjalizacja niektórych struktur danych, np. list lub s³owników:

Listing 2.1: Inicjalizacja wybranych struktur danych w j¦zyku Python.

```
e m p t y _ l i s t = [ ]
# l u b l i s t ( )
```

```
empty_tuple = ( )
#
l u b
t u p l e ( )
```

```
empty_dict = {}
#
l u b
d i c t ( )
```

empty_set = set ( )

Kolejnym, równie wa»nym powodem wyboru tego j¦zyka jest jego popu-larno±¢. J¦zyk jest udost¦pniany na licencji open source, co znacznie u³atwia spo³eczno±ci aktywny wk³ad w jego rozwój poprzez tworzenie nowych narz¦-dzi. Mimo bogatej biblioteki standardowej powsta³o wiele specjalistycznych bibliotek zewn¦trznych. Wi¦kszo±¢ z tych najpopularniejszych dotyczy za-gadnie« naukowych, w których to Python jest szczególnie popularny.

Dobrym przyk³adem jest modu³ NumPy, który pomaga m.in. w prze-prowadzaniu z³o»onych oblicze« na tablicach. Mocn¡ stron¡ tej biblioteki jest fakt, »e cz¦±¢ deniowanych operacji wykonywana jest w czasie procesu kompilacji, co niew¡tpliwie pomaga w wydajnym przeprowadzaniu oblicze«. Dobrym uzupe³nieniem jest modu³ Matplotlib, który s³u»y do generowania wykresów. Oba wymienione narz¦dzia b¦d¡ stosowane w tej pracy. Ich opis znajduje si¦ w kolejnych sekcjach rozdzia³u.

Programowanie w j¦zyku Python jest równie» proste w porównaniu do in-nych j¦zyków. Typy sprawdzane s¡ dynamicznie, a za zarz¡dzanie pami¦ci¡ odpowiada (podobnie jak np. w j¦zyku Java) garbage collector . Wymienione zalety stanowi¡ jednak przyczyn¦ stosunkowo niskiej wydajno±¢ w porówna-niu do j¦zyków kompilowanych.

14:2944556849

numpy.array(),

Popularno±¢ j¦zyka zapocz¡tkowa³a wersja 2.0 wydana w 2000 roku. Twór-cy j¦zyka zdecydowali si¦ na

stworzenie nowej ga ª ¦zi, czego efektem by ª o wydanie w roku 2008 wersji 3.0. Mimo na pozór jedynie subtelnych (cho¢ wielu) ró»nic, wersja ta jest w znacz¡cej mierze niekompatybilna z poprzed-ni¡. Pomimo rozwoju nowej ga ª ¦zi, wcze±niejsza wersja jest nadal wspierana i wci¡» bardzo popularna. W pracy b ¦dzie wykorzystywana w ª a±nie ona, a konkretnie wersja 2.7.

Praca z j ¦zykiem wymaga interpretera t ª umacz¡cego wysokopoziomowe instrukcje j ¦zyka na kod maszynowy w czasie wykonywania programu. Stano-wi on dodatkowo swoisty pomost mi ¦dzy j ¦zykiem, a systemem operacyjnym, co pozwala ª atwo tworzy¢ oprogramowanie bez wzgl ¦du na platform ¦. Inter-preter jest dost ¦pny dla wszystkich popularnych systemów, a z niektórymi z nich (np. dystrybucjami z rodziny Linux) jest dostarczany.

2.2. Korzystanie z NumPy

NumPy [11] jest pakietem oferuj¡cym wielowymiarowe macierze, funkcje dzia ª aj¡ce na macierzach, narz ¦dzia algebry liniowej, transformat ¦ Fouriera, liczby losowe, narz ¦dzia do integracji kodu z C/C++ i Fortranem. Macie-rze powinny by¢ tworzone za pomoc¡ nast ¦puj¡cych funkcji:

numpy.zeros(), numpy.ones(), numpy.empty(), numpy.arange().

2.2.1. Podstawowe operacje

Podstawowe operacje na macierzach przedstawia listing 2.2.

Listing 2.2: Sesja interaktywna z u»yciem modu ª u NumPy.

```
>>> import
numpy
```

```
>>> a = numpy . a r a n g e ( 6 )
# odmiana
r a n g e ( )
```

```
>>> a
```

```
#instancja klasy
numpy.ndarray


array([0, 1, 2, 3, 4, 5])



>>> b=a.reshape(2,
3)



>>> b
#inny widok
na te
same dane









array([[0, 1, 2],



[3,
4,
5]])



>>> b[1][2]
=
9
```

```
# zmiana
w i d o c z n a  w a  i  b


>>> c = a . r e s h a p e ( 3 ,
2 ) . copy ( )
# prawdziwa  k o p i a


>>> b . ndim



# l i c z b a
wymiarow  ( 2  a x e s )


2
```

```
>>> b . shape
```

```
( 2 ,  3 )
```

```
>>> b . dtype

#
t y p  danych ,
np . i n t ,
f l o a t , complex


dtype ( ' i n t 6 4 ' )
```

```
>>> a
2
#
d z i a l a n i a  wykonywane
e l e m e n t w i s e


a r r a y ( [ 0 ,
1 ,

4 ,
9 , 1 6 , 8 1 ] )
```

```
>>> a
5
```

```
# zwracana
j e s t
nowa  m a c i e r z


a r r a y ( [ 0 ,
5 , 1 0 , 1 5 , 2 0 , 4 5 ] )



>>> a + 1









a r r a y ( [ 1 ,
2 ,

3 ,
4 ,
5 , 1 0 ] )
```

15:1053700557

2.2.2. Indeksowanie

Silnı stronı NumPy sı ró»norodne sposoby dost¦pu do macierzy. Mo»liwe jest u»ycie obiektu macierzy jako indeksu przykªadowo w celu jej przeltro-wania.

Listing 2.3: Indeksowanie w NumPy.

```
>>> a [ numpy . a r r a y ( [ 3 ,
4 , 5 ] ) ]
# wskazujemy  wybrane  i n d e k s y
```

```
a r r a y ( [ 3 , 4 , 9 ] )
```

```
>>> a > 3
#
warunek
p r z e n o s i
s i e
na  e l e m e n t y
```

```
a r r a y ( [ F a l s e ,
F a l s e ,
F a l s e , F a l s e ,
True ,
True ] , dtype= bool )


>>> a [ a >
3 ]
```

```
a r r a y ( [ 4 , 9 ] )
```

```
>>> a [ a >
3 ] = 3   # wybieramy
e l e m e n t y  do
wymiany


>>> a
```

```
a r r a y ( [ 0 ,   1 , 2 , 3 , 3 ,   3 ] )
```

2.2.3. Brakuj¡ce warto±ci

Przetwarzane dane czasem mog¡ zawiera¢ nieprawidªowe warto±ci, które mo»emy oznaczy¢ z u»yciem staªej numpy.NAN. NumPy pozwala wygodnie przetwarza¢ takie dane za pomoc¡ specjalnych metod.

Listing 2.4: Obsªuga brakuj¡cych warto±ci w NumPy.

```
# Symulujemy dane z nieprawidlowymi wartosciami.

>>> x = numpy.array([1,
2,
numpy.NAN, 3,
4])


>>> x




array([ 1.,
2., nan,
3.,
4.])



>>> numpy.isnan(x)




array([False,
False, True,
False, False], dtype=bool)


>>> x[~numpy.isnan(x)]
#
zaprzeczenie
logiczne macierzy


array([ 1.,
2., 3.,
4.])
```

```
>>> numpy . mean ( x [ ~numpy . i s n a n ( x ) ] )
```

```
2 . 5
```

2.2.4. Funkcje matematyczne

Ponadto biblioteka udost¦pnia szereg funkcji matematycznych, którymi mo»na z ªatwo±ci¡ przetwarza¢ caªe tablice.

Listing 2.5: Funkcje matematyczne w NumPy.

```
>>> x = numpy . a r a n g e ( 4 ,
dtype= f l o a t )
# u s t a l o n y
t y p  danych
```

```
>>> x
```

```
array([ 0., 1.,
2.,

3.])
```

```
>>> numpy . exp ( x )
```

```
array([ 1.
,

2.71828183,

7.3890561 ,
20.08553692])
```

```
>>> numpy . s q r t ( x )
```

```
array([ 0.
,
1.
,
1.41421356,
1.73205081])



>>> numpy.sin(x)









array([ 0.
, 0.84147098,
0.90929743,
0.14112001])



>>> y = numpy.array([5,
6, 7, 8])

#na bazie
listy Pythona
```

```
>>> numpy . add ( x ,
y )
# d z i a l a n i e
macierzowe
```



```
a r r a y ( [ 5 ,  7 ,
9 , 1 1 ] )

>>> numpy . dot ( x ,
y )
# i l o c z y n  s k a l a r n y

4 4 . 0
```

```
>>> x . sum( ) ,  y . min ( ) ,
y .max( )

( 6 . 0 ,  5 ,  8 )
```

2.3. Korzystanie z Matplotlib

Matplotlib [12] jest to pythonowa biblioteka do tworzenia wykresów 2D i 3D w ró»nych formatach. Rysunek 2.1 prezentuje mo»liwo±ci biblioteki. Na listingu 2.6 przedstawiono sposób wygenerowania prostego wykresu funkcji liniowej.

Listing 2.6: Sesja interaktywna z u»yciem Matplotlib.

```
>>>
import m a t p l o t l i b . p y p l o t a s
p l t

# p a k i e t p y p l o t

>>> x = range ( 1 0 )

>>>
y = [ 2  i +1 for
i  in
range ( 1 0 ) ]

>>>
p l t . s c a t t e r ( x ,
y )

#
Matlab  i n t e r f a c e

>>>
p l t . t i t l e ( " Funkcja
l i n i o w a  y
= 2
x
+ 1" )
>>> p l t . x l a b e l ( "x" )
>>> p l t . y l a b e l ( "y" )

>>> p l t . g r i d ( )
>>> p l t . show ( )
```

Rysunek 2.1: Przykªad wykresów wygenerowanych za pomocj biblioteki Matplotlib.

ródªo: [12]

17:5924874201

2.4. Korzystanie z PyMongo

PyMongo jest bibliotekj j¦zyka Python uªatwiajicj prac¦ z bazami da-nych wykorzystujicymi system MongoDB. Realizacj¦ poªiczenia z bazj da-nych oraz przeprowadzanie podstawowych operacji przy pomocy biblioteki PyMongo przedstawia listing 2.7.

Listing 2.7: Korzystanie z moduªu PyMongo.

```
from pymongo import  MongoClient


#
O t w a r c i e  p o l a c z e n i a  z
serwerem
na  o k r e s l o n y m  h o s c i e  i  p o r c i e .
c l i e n t =
MongoClient ( ' l o c a l h o s t ' ,
2 7 0 1 7 )




#
Dostep
do  o b i e k t u  b a z y .


db = c l i e n t . t e s t _ d a t a b a s e


#
Dostep
do  o k r e s l o n e j
k o l e k c j i .


u s e r s _ c o l l e c t i o n = db . u s e r s _ c o l l e c t i o n




new_user
= { ' f i r s t n a m e '
: ' Mateusz ' ,


' l a s t n a m e ' : ' J a n c a r z ' ,




' hometown ' : ' Krakow ' }
```

```
#
Dodanie obiektu (uzytkownika)
do
kolekcji.

users_collection.insert_one(new_user)


#Pozyskanie listy wszystkich
uzytkownikow.
all_users=users_collection.find()
#
Pozyskanie
listy
wszystkich
uzytkownikow z Krakowa.


users_from_krakow =
users_collection.find({'hometown': 'Krakow'})




#
Pozyskanie
liczby
uzytkownikow z Krakowa.

users_from_krakow . count ( )
```

18:2352046012

3. Uczenie maszynowe

Maszynowe uczenie si¦ jest to dziedzina interdyscyplinarna, czerpi¡ca g³ównie z informatyki oraz statystyki. Jako jeden z elementów sk³adowych sztucznej inteligencji skupia si¦ na budowie oraz analizie sztucznych syste-mów zdolnych do uczenia si¦ na podstawie danych [7].

3.1. Denicja uczenia si¦

W procesie uczenia si¦ mo»na wyró»ni¢ kilka kluczowych elementów, które formu³uj¡ wymagania w stosunku do dzia³a« systemu, który mo»na okre±li¢ jako ucz¡cy si¦. Pierwszym z nich jest zmiana. Nie ka»da jednak zmiana w systemie mo»e mie¢ charakter uczenia si¦. Jako kontrprzyk³ad mo»na tutaj poda¢ zjawisko zapominania. Zmian¡ o jak¡ nam w tym przypadku chodzi jest taka, która w nosi popraw¦ do dzia³ania systemu. Taka korzystna zmia-na jest czynnikiem wymaganym, ale nie wystarczaj¡cym. Wymagan¡ cech¡ poprawy jest jej autonomiczno±¢, a wi¦c ¹ród³o jej pochodzenia ma by¢ w samym systemie. System, który si¦ uczy sam zmienia si¦ na lepsze. Za nauk¦ nie mo»na uzna¢ dodanie nowej funkcjonalno±ci do systemu, albo popraw¦ ju» istniej¡cej. Tak¡ - mimo, »e korzystn¡ zmian¦ - nie mo»emy uzna¢ za nauk¦. Zmiana cho¢ autonomiczna, nie mo»e by¢ spowodowana samoistnie czy te» losowo, ale pod wp³ywem czynników, b¡d¹ okoliczno±ci zewn¦trznych. Taki zewn¦trzny wp³yw mo»emy wtedy nazwa¢ do±wiadczeniem zdobywanym przez system ucz¡cy si¦.

Denicja uczenia si¦, do której doprowadzi³y powy»sze rozwa»ania, mo»e brzmie¢ nast¦puj¡co [7]:

Denicja 1. Uczeniem si¦ systemu jest ka»da autonomiczna zmiana w sys-temie na podstawie do±wiadcze«, która doprowadzi do poprawy jako±ci jego dzia³ania.

3.2. Proces uczenia si¦ w kontek±cie programów komputerowych

Potrzeba uczenia si¦ programów komputerowych bynajmniej nie jest zwi¡-zana z in»yneri¡ oprogramowania i ch¦ci¡ zmniejszenia wysi ku projektan-tów czy programistów. Motywacja do tworzenia tego typu systemów wynika z trudno±ci w rozwi¡zywanych przez nie problemach, które cz¦sto wymagaj¡ niekonwencjonalnego podej±cia. W przypadku niektórych stawianych proble-mów bardziej opaca si¦ stworzy¢ algorytm daj¡cy maszynie zdolno±¢ do nauki rozwi¡zywania problemu zamiast zaprogramowania jej wprost. Mo»e

to wynika¢ na przyk ad z nik ej wiedzy lub wr¦cz jej braku o ±rodowisku pracy systemu.

Programy ucz¡ce si¦ wykorzystuj¡ abstrakcyjny, w pewien sposób spara-metryzowany algorytm wykonywania zadania. Uczenie si¦ wówczas polega na takiej analizie danych historycznych, aby te parametry odpowiednio dobra¢ i w konsekwencji przekszta ci¢ w odpowiedni algorytm konkretny, rozwi¡-zuj¡cy problem stawiany przez konstruktora systemu. Ilustruje to schemat 3.1.

Rysunek 3.1: Uczenie si¦ jako konkretyzacja algorytmu.

ród o: opracowanie w asne na podstawie: [7]

Uzyskane w wyniku nauki parametry okre±la si¦ w zale»no±ci od rodzaju wykonywanego zadania wiedz¡ lub umiej¦tno±ci¡. Cz¦sto s¡ one okre±lane jako hipoteza, podkre±laj¡c fakt, i» najcz¦±ciej nie mo»na ich uzna¢ jako nie-zawodne i mog¡ wymaga¢ rewizji [7].

20:9777229923

3.3. Rodzaje uczenia maszynowego

Rozwijajıc temat uczenia maszynowego nale»y na samym poczıtku upo-rzıdkowa¢ dotychczasowı wiedz¦ o nim i przedstawi¢ ró»ne jego rodzaje, aby zna¢ wachlarz mo»liwo±ci, z których mo»emy wybiera¢ konkretne metody najlepiej pasujıce do charakterystyki rozwiızywanego problemu.
Szereg algorytmów uczenia si¦ mo»na podzieli¢ wedªug kilku kryteriów. Najcz¦±ciej spotyka si¦ podziaª bioröcy pod uwag¦ tzw. informacj¦ trenujıcı. Przyjmijmy, »e system uczıcy si¦ ma za zadanie poprzez obserwacj¦ pewnych danych wej±ciowych generowa¢ na ich podstawie odpowiedzi, a uczenie si¦ jest procesem zwi¦kszania efektywno±ci tego zabiegu. W takim przypadku infor-macja trenujıca instruuje w pewien sposób ucznia co do wªa±ciwego sposobu odpowiadania. Kryterium w tym podziale jest to, czy taka informacja jest dost¦pna dla systemu - innymi sªowy - czy ucze« posiada nauczyciela. Z tego wzgl¦du wymienia si¦:

uczenie z nadzorem (ang. supervised learning),
uczenie bez nadzoru (ang. unsupervised learning).

Dodatkowo wyró»nia si¦ trzeciı grup¦, o której wspomniano ju» we wst¦-pie pracy, czyli uczenie ze

wzmocnieniem, okre±lane równie» jako nauk¦ po-przez interakcj¦. Ten sposób nauki jest znacznie bardziej skupiony na celu nauki ni» pozosta ª e podej±cia. Równie» istnieje w nim zewn¦trzne ¹ród ª o in-formacji trenuj¡cej, ma ono jednak charakter raczej krytyka ni» nauczyciela

[13].Pe ª ni on rol¦ warto±ciuj¡c¡ dla prób testuj¡cych agenta ucz¡cego si¦. Ucze« otrzymuje on od niego wzmocnienia uzale»nione od skutków podej-mowanych dzia ª a«. System stara si¦ maksymalizowa¢ warto±ci wzmocnie« w d ª ugim horyzoncie czasowym, co formalnie ma reprezentowa¢ zwi¦kszanie efektywno±ci wykonywanych przez niego zada«.

21:4851053491

4. Uczenie si¦ ze wzmocnieniem

W tym paradygmacie uczenia wyst¦puje dynamiczna interakcja z otocze-niem. Ucze« obserwuje stany ±rodowiska, wykonuje akcje, obserwuje efekty tych akcji przez otrzymywanie nagród (warto±ci wzmocnienia). Zadaniem ucznia jest znalezienie strategii wybierania akcji, która prowadzi do dªugo-terminowej maksymalizacji nagród [7].

Podstawy teoretyczne uczenia si¦ ze wzmocnieniem s¡ zwi¡zane z proce-sami decyzyjnymi Markowa i metodami programowania dynamicznego. Zo-stan¡ przedstawione potrzebne poj¦cia z tej dziedziny korzystaj¡c z prostego modelu rynku (gieªdy).

4.1. Zadanie uczenia si¦ ze wzmocnieniem

Jak wspominano kluczowym aspektem uczenia ze wzmocnieniem jest na-uka strategii wybierania akcji. Jej konstruowanie odbywa si¦ na podstawie bezpo±rednich interakcji ze ±rodowiskiem metod¡ prób i bª¦dów. Informacja trenuj¡ca ma charakter warto±ciuj¡cy (oceniaj¡cy) jako±¢ działania ucznia, a nie instruuj¡cy go, w jaki sposób ma działa¢. Scenariusz uczenia si¦ ze wzmocnieniem mo»na przedstawi¢ za pomoc¡ abstrakcyjnego algorytmu jak na rysunku 4.1.

Rysunek 4.1: Schemat procesu uczenia si¦ ze wzmocnieniem.

ródªo: [14]

Na listingu 4.1 uogólniony algorytm uczenia si¦ ze wzmocnieniem zapisa-ny w pseudokodzie.

22:1047899171

wªasno±ci¡ Markowa

t na-

Listing 4.1: Algorytm uczenia si¦ ze wzmocnieniem.

d l a w s z y s t k i c h kroków c z a s u t wykonaj :

o b s e r w u j a k t u a l n y s t a n xt ;

w y b i e r z akcj¦ at do wykonania w
s t a n i e
xt ;

wykonaj akcj¦ at ;

o b s e r w u j wzmocnienie rt i nast¦pny s t a n
xt+1 ;

ucz si¦ na p o s t a w i e do±wiadczenia
hxt, at, rt, xt+1i .

rodowisko jest przez ucznia niekontrolowalne . Ucze« powinien maksyma-lizowa¢ swoje nagrody dªugoterminowo. Najcz¦±ciej przyjmuje si¦, »e ucze« powinien maksymalizowa¢ zdyskontowan¡ sum¦ otrzymanych nagród

E
" ∞
γtrt# ,
(4.1)

X

t=0

gdzie γ ∈ [0, 1] jest wspóªczynnikiem dyskontowania
a rt to wzmocnienia w
kolejnych chwilach czasu t. Wspóªczynnik γ jest miar¡ "dalekowzroczno±ci"
ucznia.

W przypadku uczenia si¦ ze wzmocnieniem mo»emy mie¢ do czynienia z ró»nymi trybami nauki zale»nymi od sposobu modykacji strategii decy-zyjnej ucznia. My skupimy si¦ na trybie inkrementacyjnym , gdzie strategia decyzyjna mo»e podlega¢ modykacji w ka»dym kroku czasu pod wpªywem pozyskiwanego do±wiadczenia.

4.2. Procesy decyzyjne Markowa

Modelem matematycznym zadania uczenia si¦ ze wzmocnieniem jest pro-blem decyzyjny Markowa [7].

Denicja 2. (Proces decyzyjny Markowa) Proces decyzyjny Markowa jest

zdenowany jako czwórka hX, A, ρ, δi, przy czym
X jest sko«czonym zbiorem stanów,
A jest sko«czonym zbiorem akcji,
ρ jest funkcj¡ wzmocnienia,
δ jest funkcj¡ przej±¢ stanów.

We wcze±niejszej notacji denicja oznacza, »e w ka»dym kroku czasu

groda rt (liczba rzeczywista) jest realizacj¡ zmiennej losowej ρ(xt, at), a stan
xt+1 jest realizacj¡ zmiennej losowej δ(xt, at). Zgodnie z

funkcje ρ i δ nie zale»¡ od historii. Wygodnie jest wprowadzi¢ oznaczenia:

R(x, a) = E[ρ(x, a)],
(4.2)
Pxy(a) = Pr[δ(x, a) = y].
(4.3)

19

23:3887376069

Denicja 3. (Strategia)

4.3. Strategie

Strategi¡ dla procesu decyzyjnego Markowa jest dowolna funkcja π : X → A.

Podana denicja okre±la strategi¦ stacjonarn¡ i deterministyczn¡ , co uprasz-cza teori¦. W praktyce algorytmy uczenia si¦ ze wzmocnieniem wykorzystuj¡ z regu³y strategie niestacjonarne i niedeterministyczne, które pozwalaj¡ na prac¦ w ±rodowiskach zmiennych i nie do ko«ca przewidywalnych. System po-s³uguje si¦ strategi¡ π, je»eli w ka»dym kroku t wykonuje on akcj¦ at = π(xt).

Do oceniania strategii s³u»¡

funkcja warto±ci

i funkcja warto±ci akcji .

Denicja 4. (Funkcja warto±ci)
cja warto±ci ze wzgl¦du na strategi¦

Dla procesu decyzyjnego Markowa funk-

π jest dla ka»dego stanu $x \in X$ okre±lona

nast¦puj¡co:

" ∞

#

V π(x) = Eπ

X γtrt

t=0

Denicja 5. (Funkcja warto±ci akcji) Dla procesu decyzyjnego Markowa funkcja warto±ci akcji ze wzgl¦du na strategi¦ π jest dla ka»dej pary (x, a),

x ∈ X, a ∈ A okre±lona nast¦puj¡co:
Qπ(x, a) = Eπ "ρ(x, a) +
∞
γtrt | x0 = x, a0 = a# .
(4.5)

X

t=1

Warto zauwa»y¢, »e V π(x) = Qπ(x, π(x)), czyli Qπ zawiera wi¦cej informacji ni» V π.

4.4. Model rynku

Rozwa»my prosty model rynku, który jest ±rodowiskiem dla procesu de-cyzyjnego Markowa. Istnieje mo»liwo±¢ kupowania b¡d¹ sprzeda»y jednego pakietu akcji. Jego cena mo»e przyjmowa¢ dwie umowne warto±ci: 1 lub 2.

Warto dodatkowo uwzgl¦dni¢ tendencj¦ zmian kursu (rosn¡ca ↑, malej¡ca ↓, bez zmian), dlatego w zbiorze X okre±lamy cztery stany. Mo»na powie-

dzie¢, »e w okre±leniu stanu jest zakodowana historia si¦gaj¡ca jeden krok w przeszªo±¢.

X = {1 , 1 ↓, 2 , 2 ↑} (zbiór stanów),
(4.6)
A = {0, 1} (zbiór akcji),
(4.7)
gdzie a = 0 oznacza sprzedaj lub nie kupuj , a = 1 znaczy kupuj
lub nie

sprzedawaj. Rozkªady prawdopodobie«stw przej±¢ nie zale»¡ od akcji i s¡ podane w tablicy 4.1.

Warto±ci oczekiwane nagród s¡ okre±lone poni»ej, przy czym staªa K > 0 okre±la ró»nic¦ warto±ci pakietu akcji pomi¦dzy poziomami kursu 1 i 2:

(1

p)K
dla x = 1
↓

i a = 1,

pK−

dla x = 1

i a = 1,

R(x, a) =

(1
p)K
dla
x = 2

i
a = 1,
(4.8)

–

–

↑

pK
dla
x

i

a
,

−

= 2

0

w pozostaªych przypadkach .

20

24:4825372726

Tabela 4.1: Rozkªad prawdopodobie«stwa przej±¢ mi¦dzy stanami. Przej±cia nie zale»¡ od akcji. Je»eli p = 1, to ±rodowisko jest deterministyczne i mamy

powtarzajicy si¦ ci¡g przej±¢ (1 ↓, 1 , 2 ↑, 2). Je»eli p = 0.5, to mamy niedeterministyczne przeskoki mi¦dzy cenami 1 i 2.

Pxy
y = 1 ↓
y = 1
y = 2 ↑
y = 2
x = 1 ↓
0
p
1 − p
0
x = 1
0
1 − p
p

0
x = 2 ↑
1 − p
0
0
p
x = 2
p
0
0
1 − p

Tabela 4.2: Strategie dla modelu rynku.

Stany
π1
π2
π3
π4
1 ↓
0
1
1
0
1
0
1
1
1
2 ↑
0
1
0
1
2
0
1
0
0

Badane strategie zamieszczono w tabeli 4.2. Analiz¦ modelu warto roz-poczi¢ od najprostszych strategii.

Strategia π1: Ta strategia oznacza brak zaanga»owania w kupowanie pa-kietu akcji. Wyznaczamy funkcj¦ warto±ci i funkcj¦ warto±ci akcji.

V π1 (2 ↑) = V π1
(2) = V π1 (1 ↓)
= V π1 (1) = 0.

(4.9)
$Q\pi 1 (1 \downarrow, 0) =$
0,
$Q\pi 1 (1 \downarrow, 1)$
$= (1 - p)K,$
(4.10)
$Q\pi 1 (1\ , 0) =$
0,
$Q\pi 1 (1\ , 1)$
$= pK,$
(4.11)
$Q\pi 1 (2 \uparrow, 0) =$
0,
$Q\pi 1 (2 \uparrow, 1)$
$= -(1 - p)K,$
(4.12)
$Q\pi 1 (2\ , 0) =$
0,
$Q\pi 1 (2\ , 1)$
$= -pK.$
(4.13)

Zauwa»my, »e $Q\pi 1 (x, a)$ odtwarza $R(x, a)$.

Strategia $\pi 2$: Ta strategia polega na sta³ym trzymaniu kupionego pakietu akcji. Obliczamy funkcj¦ warto±ci.

$V \pi 2 (1 \downarrow)$
$=$
$p\gamma V \pi 2 (1\ ) + (1 - p)[K + \gamma V \pi 2 (2 \uparrow)],$
(4.14)

$V \pi 2 (1\ )$
$=$
$(1 - p)\gamma V \pi 2 (1\ ) + p[K + \gamma V \pi 2 (2 \uparrow)],$
(4.15)

$V \pi 2 (2 \uparrow)$
$=$
$p\gamma V \pi 2 (2\ ) + (1 - p)[-K + \gamma V \pi 2 (1 \downarrow)],$
(4.16)

$V \pi 2 (2\ )$
$=$
$(1 - p)\gamma V \pi 2 (2\ ) + p[-K + \gamma V \pi 2 (1 \downarrow)].$
(4.17)

W przypadku deterministycznym $p = 1$ mamy proste rozwi¦zanie:

$V \pi 2 (1 \downarrow) = \gamma V \pi 2 (1\ ) = -V \pi 2 (2 \uparrow) = -\gamma V \pi 2 (2\ ) =$

γK

.
(4.18)

1 + γ2

21

25:1037133857

mo»emy obliczy¢ z

Dla przypadku p = 0.5 otrzymujemy:

V π2 (1 ↓) = V π2 (1) = −V π2 (2 ↑) = −V π2 (2) =
K

.
(4.19)

2

Obliczamy funkcj¦ warto±ci akcji.

$$Q\pi 2\,(1\downarrow, 0) = V\,\pi 2\,(1\downarrow) - (1-p)K, \tag{4.20}$$

$$Q\pi 2\,(1\downarrow, 1) = V\,\pi 2\,(1\downarrow), \tag{4.21}$$

$$Q\pi 2\,(1\ , 0) = V\,\pi 2\,(1\ ) - pK, \tag{4.22}$$

$$Q\pi 2\,(1\ , 1) = V\,\pi 2\,(1\ ), \tag{4.23}$$

$$Q\pi 2\,(2\uparrow, 0) = V\,\pi 2\,(2\uparrow) + (1-p)K, \tag{4.24}$$

$$Q\pi 2\,(2\uparrow, 1) = V\,\pi 2\,(2\uparrow), \tag{4.25}$$

$$Q\pi 2\,(2\ , 0) = V\,\pi 2\,(2\ ) + pK, \tag{4.26}$$

Qπ2 (2 , 1)
=
V π2
(2).

(4.27)

Strategia π3: Ta strategia jest optymalna, co mo»na sprawdzi¢ tworz¡c stra-tegi¦ zach ªann¡ na bazie strategii π1 lub te» obserwuj¡c funkcje warto±ci akcji.

Funkcje warto±ci dla stanów przy stosowaniu strategii π3 uk ªadu równa«.

V π3 (1 ↓)
=
pγV π3
(1) + (1 − p)[K + γV π3
(2 ↑)],
(4.28)
V π3 (1)
=
(1 − p)γV π3
(1) + p[K + γV π3
(2 ↑)],
(4.29)
V π3 (2 ↑)
=
pγV π3
(2) + (1 − p)γV π3
(1 ↓),

(4.30)
V π3 (2)
=
(1 − p)γV π3
(2) + pγV π3
(1 ↓).

(4.31)
W szczególnym przypadku p = 1 rozwi¡zanie jest proste:

V π3 (2 ↑) = γV π3 (2) = γ2V π3 (1 ↓) = γ3V π3 (1) =

γ3K

.
(4.32)

1 − γ4

Równie» w przypadku p = 0.5 istnieji proste zwi¡zki:

V π3 (1) = V π3 (1) =
(2 − γ)K
,

(4.33)

↓

4(1
−
γ)

V π3 (2 ↑) = V π3 (2) =

γK

.

(4.34)

4(1 − γ)

Obliczamy funkcj¦ warto±ci akcji.

Qπ3 (1 ↓, 0)
=
V π3 (1 ↓) − (1 − p)K,

(4.35)

Qπ3 (1 ↓, 1)
=
V π3 (1 ↓),

(4.36)

$$Q\pi 3(1\ ,0) = V\pi 3(1\ ) - pK,$$

(4.37)

$$Q\pi 3(1\ ,1) = V\pi 3(1\ ),$$

(4.38)

$$Q\pi 3(2\ 个,0) = V\pi 3(2\ 个),$$

(4.39)

$$Q\pi 3(2\ 个,1) = V\pi 3(2\ 个) - (1-p)K,$$

(4.40)

Qπ3 (2 , 0) =
V π3 (2),

(4.41)

Qπ3 (2 , 1)
=
V π3 (2) − pK.

(4.42)

26:1109728721

4.5. Klasy metod uczenia si¦ ze wzmocnieniem

Wyró»nia si¦ trzy podstawowe klasy metod do rozwi¡zywania problemów uczenia si¦ ze wzmocnieniem:

programowanie dynamiczne, metody Monte Carlo, oraz metody ró»nic czasowych. Ka»da z klas ma swoje wady oraz za-lety. Programowanie dynamiczne wymaga ±cisªej denicji modelu ±rodowiska. Metody Monte Carlo nie maj¡ takiego wymagania, ponadto nie nadaj¡ si¦ do rozwi¡zywania problemów inkrementacyjnie, co powoduje, »e s¡ bardziej wymagaj¡ce pami¦ciowo. Metody ró»nic czasowych jako b¦d¡ce stosunkowo nowym podej±ciem w uczeniu ze wzmocnieniem ª¡cz¡ zalety obu wymienio-nych grup.

4.6. Metody ró»nic czasowych

Stosunkowo nowym podej±ciem w uczeniu si¦ ze wzmocnieniem s¡ tzw. metody ró»nic czasowych (ang. Temporal Dierences , TD). Jest to klasa me-tod dla predykcji w wieloetapowych problemach predykcyjnych, czyli takich, w których poprawno±¢ przewidywa« nie mo»e by¢ zwerykowana natych-miast, lecz dopiero po pewnej liczbie kroków [13].

W tym paradygmacie podczas uczenia si¦ funkcji warto±ci zaczynamy z arbitralnie zainicjowanej wielko±ci V0 i pewnej strategii π. Operacja aktuali-zacji funkcji warto±ci V dla stanu xt w tej metodzie wygl¡da nast¦puj¡co:

$$V(x_t) \leftarrow V(x_t) + \alpha[r_t + \gamma V(x_{t+1}) - V(x_t)]. \quad (4.43)$$

Wyró»nia si¦ dwa sposoby implementacji algorytmów TD:

on-policy (ze strategii) - gdzie rozwa»ana jest warto±¢ najlepszej akcji dla obranej strategii;

o-policy (bez strategii) - gdzie rozwa»ana jest warto±¢ najlepszej akcji ignoruj¡c aktualn¡ strategi¦.

4.6.1. Algorytm Q-learning

Popularn¡ implementacj¡ podej±cia TD jest algorytm Q-learning zapro-ponowany przez C. Watkinsa w 1989. Algorytm uczy si¦ funkcji warto±ci

akcji Q, co sugeruje jego nazwa. Okre±la si¦ go jako o-policy, gdzie bezpo-±rednio o wyborze akcji a w stanie x decyduje warto±¢ Q(x, a). Ucze« mo»e

posªugiwa¢ si¦ strategi¡ inn¡ ni» ta, której si¦ uczy. Algorytm Q-learning jest zaprezentowany na listingu 4.2.

Listing 4.2: Algorytm Q-learning.

```
z a i n i c j a l i z u j
Q(x, a) d l a w s z y s t k i c h stanów
d l a w s z y s t k i c h
kroków c z a s u
t wykonaj :
o b s e r w u j a k t u a l n y s t a n xt ;
w y b i e r z akcj¦at na b a z i e
xt i Q ;
wykonaj akcj¦at ;

o b s e r w u j wzmocnienie rt i nast¦pny s t a n
```

wyznacz
$\leftarrow r_t + \gamma \max_a Q(x_{t+1}, a) - Q(x_t, a_t)$;

a k t u a l i z u j $Q(x_t, a_t) \leftarrow Q(x_t, a_t) + \alpha$.

x i a k c j i a ;

xt+1 ;

23

27:1151913784

st¦pujıcı posta¢:

Action, Reward, State, Action

W ka»dy kroku czasowym aktualizowana jest warto±¢ funkcji warto±ci akcji Q. Formuªa do jej wyliczania wyglıda nast¦pujıco:

$$Q(x_t, a_t) \leftarrow Q(x_t, a_t) + \alpha[r_t + \gamma \max_a Q(x_{t+1}, a) - Q(x_t, a_t)], \qquad (4.44)$$

gdzie Q to warto±¢ akcji, x to stan ±rodowiska, t jest krokiem czasowym (kolejnı iteracji), $\alpha \in [0, 1]$ jest współczynnikiem uczenia si¦, a $\gamma \in [0, 1]$ to współczynnik dyskontowania nagród.

Parametr γ reguluje znaczenie nagród krótkoterminowych w stosunku

do tych otrzymywanych w dªu»szym horyzoncie czasowym. Im ni»sza jest warto±¢ parametru, tym mniejsze znaczenie majı nagrody dªugoterminowe,

warto±¢ $\gamma = 1$ natomiast traktuje wszystkie wzmocnienia jednakowo, nieza-le»nie od momentu ich otrzymania.

Bardzo cz¦sto przy prezentacji tej formuªy parametr α jest pomijany,

z racji ustalenia jego warto±ci na 1. Taka posta¢ rzeczy jest odpowiednia dla ±rodowisk o charakterze stacjonarnym, czyli niezmiennym w czasie. W innym przypadku nale»y rozwa»y¢ zmniejszenie jego warto±ci, powodujıc u agenta cz¦±ciowe zapominanie zdobywanej wiedzy.

4.6.2. Algorytm Sarsa

Bardzo bliski algorytmowi Q-learning jest algorytm Sarsa (ang. State,). Reguªa aktualizacji funkcji Q ma tutaj na-

$$Q(x_t, a_t) \leftarrow Q(x_t, a_t) + \alpha[r_t + \gamma Q(x_{t+1}, a_{t+1}) - Q(x_t, a_t)], \quad (4.45)$$

w której nie wyst¦puje maksymalizacja warto±ci akcji w nast¦pnym stanie, ale warto±¢ akcji, jaka faktycznie w tym stanie zostanie wybrana do wy-konania. Zwykle u»ywa si¦ do uczenia funkcji warto±ci akcji strategii, której algorytm si¦ uczy, co zalicza algorytm Sarsa do kategorii on-policy. Szczagóªy algorytmu przedstawia listing 4.3.

Listing 4.3: Algorytm Sarsa.

```
z a i n i c j a l i z u j
Q(x, a)
d l a  w s z y s t k i c h  stanów  x  i  a k c j i  a ;

d l a  w s z y s t k i c h
kroków
c z a s u  t wykonaj :

je»eli t = 0 t o :
```

o b s e r w u j a k t u a l n y s t a n x0 ;
w y b i e r z akcj¦ a0 na p o d s t a w i e x0 i Q ;
wykonaj akcj¦
at ;

o b s e r w u j
wzmocnienie
rt i nast¦pny s t a n xt+1 ;
w y b i e r z akcj¦
at+1 na
p o d s t a w i e xt+1 i Q ;
wyznacz
← rt + γQ(xt+1, at+1) − Q(xt, at) ;

a k t u a l i z u j Q(xt, at) ← Q(xt, at) + α .

4.6.3. lady aktywno±ci - TD(λ)

Metody ró»nic czasowych mog¡ by¢ parametryzowane wspóªczynnikiem ±wie»o±ci λ ∈ [0, 1]. W standardowej wersji algorytmu ma on warto±¢ ze-
ro, wi¦c wªa±ciwym jest okre±lenie go jako TD(0). Warto zatem rozwa»y¢

28:8568436848

λ> 0, gdy» mo»e to w wielu przypadkach przyspieszy¢ proces uczenia si¦.

Takie podej±cie okre±la si¦ mianem ±ladów aktywno±ci (ang. eligibility tra-ces). Zastosowanie tej metody jest szczególnie wa»ne w przypadkach, kiedy wzmocnienia s¡ w pewien sposób odroczone, czyli nie mog¡ zosta¢ przekazane w ka»dym kroku czasowym.

Zaktualizowana formuªa 4.43 dla dowolnej warto±ci λ > 0 prezentuje si¦ nast¦puj¡co:

V (xt) ← V (xt) + α[rt + γV (xt+1) − V (xt)]ex(t).
(4.46)

Jak wida¢ nasze równanie to teraz iloczyn poprzedniej wersji przez no-wy parametr ex(t) zwany ±ladem aktywno±ci stanu x w kroku t, gdzie x

jest stanem, dla którego w danym momencie nast¦puje aktualizacja funkcji warto±ci. Gªówn¡ zmian¡ jest bowiem fakt, »e reguªa jest stosowana nie dla pojedynczego stanu, ale dla wszystkich stanów w danym kroku czasowym.

4.6.4. Q(λ)

Znanych jest kilka implementacji ±ladów aktywno±ci dla algorytmu Q-learning. Najpopularniejszym rozwi¡zaniem jest zaproponowane przez Watkinsa. Wy-korzystuje ono podej±cie podobne jak w 4.46 z t¡ ró»nic¡, »e ±cie»ka ±ladów aktywno±ci jest ucinana w momencie pierwszego wyboru akcji drog¡ eksplo-racji - zakªadaj¡c, »e posªugujemy si¦ strategi¡ ε-zachªann¡. Czasem jednak

nie stosuje si¦ tego wykluczenia, co mo»e przynie±ć korzy±ci w przypadkach, kiedy ucze« posªuguje si¦ strategi¡ zachªann¡ w znacz¡cej wi¦kszo±ci przy-padków.

Rozszerzona o ±lady aktywno±ci wersja algorytmu Q-learning prezentuje si¦ w pseudokodzie nast¦puj¡co:

Listing 4.4: Algorytm Q-learning lambda.

```
z a i n i c j a l i z u j        Q(x, a) o r a z e(x, a) ← 0 d l a w s z y s t k i c h x i a .

powtórz
d l a  ka»dego
e p i z o d u :




z a i n i c j a l i z u j x , a ;









powtórz d l a  ka»dego kroku w
e p i z o d z i e :




wykonaj a , o b s e r w u j r , x0 ;




w y b i e r z  a0
d l a  x0
na p o d s t a w i e  Q ;
```

$a* \leftarrow \text{argmax}_b Q(x0, b)$;

$\leftarrow r + \gamma Q(x0, a*) - Q(x, a)$;

$e(x, a) \leftarrow e(x, a) + 1$;

d l a w s z y s t k i c h x i a :

) + α
e(x, a) ;

Q(x, a) ← Q(,x, tao

je±li a0
= a∗
$e(x, a) \leftarrow \gamma\lambda e(x, a)$,

w
przeciwnym.
r a z i e e(s, a) ← 0 ;

$x \leftarrow x0$; $a \leftarrow a0$

do
momentu , a»
x j e s t
stanem terminalnym .

5. Projektowanie aplikacji

Rozdziaª po±wi¦cony jest projektowi tworzonemu w ramach tej pracy. Po-cz¡tek przedstawia zastosowanie uczenia maszynowego w bran»y inwestycyj-nej. Nast¦pnie zaprezentowany jest temat gieªd zakªadów sportowych, czyli domeny tworzonej aplikacji. W kolejnej cz¦±ci opisane s¡ elementy skªadaj¡ce si¦ na projekt oraz przedstawione s¡ szczegóªy implementacyjne. Ko«cówka rozdziaªu zawiera testy zaprogramowanej symulacji.

5.1. Zastosowanie maszynowego uczenia si¦ w konstruowaniu strategii gieªdowych

Sztuczna inteligencja mimo, »e maªymi krokami, to sukcesywnie wkra-cza do naszego codziennego »ycia. Jedn¡ z dziedzin, w których naprawd¦ mo»e ona rozwin¡¢ skrzydªa jest bran»a inwestycyjna, a konkretnie gieªdy papierów warto±ciowych i inne podobne jej rynki nansowe. Odpowiednie jej zastosowanie pomaga w przeprowadzaniu tzw. spekulacji gieªdowych, czyli przewidywania zmian cenowych, co ma prowadzi¢ do skutecznego inwesto-wania kapitaªu.

Jest to idealne ±rodowisko do zastosowania elementów sztucznej inteli-gencji. W tej chwili jest to obszar w peªni skomputeryzowany, a dost¦p do danych gieªdowych w formie cyfrowej jest powszechny, a w niektórych przy-padkach znacz¡co uªatwiony. Wiele podmiotów oferuje bowiem dost¦p do swoich danych oraz usªug poprzez udost¦pnianie wªasnego API (ang. Appli-cation Programming Interface), które upraszcza tworzenie wªasnych narz¦dzi.

Te i inne czynniki powoduj¡, »e obecnie okoªo 84% wszystkich transakcji gieªdowych jest przeprowadzanych bez czynnego udziaªu czªowieka. W 1960 roku ±redni czas przetrzymywania zakupionej akcji wynosiª 4 lata. W roku 2000 warto±¢ ta spadªa do 8 miesi¦cy. Wedªug statystyk z czerwca 2014 czas ten to 20 sekund, a obecnie ju» poni»ej 10 [15].

Istniej¡ ró»ne podej±cia w budowaniu tego typu systemów wykorzystuj¡c uczenie maszynowe. Tradycyjnym rozwi¡zaniem jest przewidywanie przyszªej ceny zasobu na podstawie warto±ci obserwowanych w przeszªo±ci. Wedªug tak stworzonych prognoz generowane s¡ sygnaªy kupna, b¡d¹ sprzeda»y, które potem przekazywane s¡ do komponentu odpowiedzialnego za przeprowadza-nie transakcji gieªdowych. W publikacji [16] wymienia si¦ kilka wad takiego rozwi¡zania, m.in. brak mo»liwo±ci uwzgl¦dnienia kosztów przeprowadzanych transakcji. Gªówn¡ trudno±ci¡ w tym podej±ciu jest sformuªowanie prognoz przy uwzgl¦dnieniu wszystkich istotnych czynników wpªywaj¡cych na wyso-ko±¢ kursu. Model tego rozwi¡zania przedstawiono na ilustracji 5.1.

Rysunek 5.1: Model systemu wykorzystuj¡cego nauk¦ przez prognozowanie.

ródªo: [16]

Wydaje si¦, »e pozornie prostszym w implementacji oraz skuteczniejszym rozwi¡zaniem jest podej±cie do problemu wykorzystuj¡c opisywane w pracy uczenie ze wzmocnieniem. Eliminuje to skomplikowany proces przewidywania przyszªych cen i powoduje, »e zachowanie agenta oraz jego skuteczno±¢ zale»¡ wyª¡cznie od wykonywanych przez niego transakcji. Znika zatem wymaganie uwzgl¦dnienia wprost wszystkich czynników, które s¡ w tym podej±ciu repre-zentowane przy pomocy wzmocnie« otrzymywanych przez agenta. Schemat 5.2 przedstawia ogóln¡ architektur¦ takiego systemu.

Rysunek 5.2: Model systemu wykorzystuj¡cego uczenie przez wzmocnienie.

ródªo: [16]

Korzystaj¡c z tych wniosków autor postanowiª stworzy¢ implementacj¦ opisywanego systemu uwzgl¦dniaj¡c generowanie mo»liwe trafnych sygna-

ªów kupna oraz sprzeda»y na podstawie do±wiadczenia zdobywanego pod-czas gry. Aplikacja nie obejmuje mechanizmu przeprowadzania rzeczywistych transakcji, natomiast symuluje go stosuj¡c pewne uproszczenie. Zaª o»ono w nim, »e ka»dorazowy sygnaª spowoduje automatyczny zakup, co w rzeczywi-stych warunkach nie jest zapewnione.

5.2. Gieªdy zakªadów sportowych

Dane, z których korzysta aplikacja, to rzeczywiste dane pochodz¡ce z tzw. gieªdy zakªadów sportowych. Gieªd¦ tak¡ mo»na okre±li¢ jako podmiot zajmuj¡cy si¦ przyjmowaniem zakªadów pieni¦»nych zwi¡zanych z ró»nymi dyscyplinami sportowymi, czyli tzw. bukmacher. W przeciwie«stwie jednak do tradycyjnych jednostek stronami handlu nie jest gieªda i gracz, ale sami gracze. Gieªda jest jedynie po±rednikiem i udost¦pnia odpowiednie mechani-zmy do przeprowadzana transakcji. Mechanizm ten przypomina uproszczon¡ form¦ znan¡ np. z gieªd papierów warto±ciowych.

Zakªady mog¡ by¢ skªadane za danym rezultatem, b¡d¹ przeciwko, co w teorii prowadzi do tego, »e mo»na je odprzedawa¢. Stwarza to mo»liwo±¢ spekulacji gieªdowej i czerpanie zysków bez wzgl¦du na ko«cowy rezultat wydarzenia sportowego.

Przed opisem mechanizmów gieªdowych warto zdeniowa¢ kilka mniej lub bardziej formalnych poj¦¢:

wydarzenie: Dowolne wydarzenie sportowe, w ramach którego mo»na zakupi¢ zakªad, np. mecz piªkarski, walka bokserska itp.

rynek: W ogólnym rozumieniu jest to obszar gieªdy, w ramach które-go gracze handluj¡ na ±ci±le okre±lonych zasadach. W ramach ka»dego z wydarze« na gieªdzie udost¦pniony jest szereg rodzajów rynków.

opcja: Rezultat danego wydarzenia mo»liwy do obstawienia, spo±ród kil-ku na danym rynku. Termin opcja b¦dzie stosowane wymiennie z okre-±leniem rezultat.

kierunek zakªadu: Typ zakªadu jaki mo»na zakupi¢ na danej opcji. Mo»liwe jest zagranie za dan¡ opcj¡ lub jej przeciwie«stwem (równoznacz-ne z obstawieniem ka»dego innego rezultatu). Zakªad za okre±lany b¦dzie poj¦ciem back, a przeciw jako lay.

Przykªady dla niektórych wprowadzonych poj¦¢ przedstawia tabela 5.1.

Tabela 5.1: Przykªady dla wprowadzonych poj¦ç dla meczu piªkarskiego.

Wydarzenie	Rynki	Opcje
wygra Real Madrid	Kurs meczowy	
wygra FC Barcelona		
Real Madrid vs FC Barcelona		
mecz zako«czy si¦ remisem		
Poni»ej/powy»ej 2.5 gola	padnie poni»ej 2.5 goli	
	padnie powy»ej 2.5 goli	

32:4574631179

5.2.1. Handel na gieªdzie

Ka»dy rynek posiada z góry okre±lonı liczb¦ mo»liwych rezultatów. Cena obstawienia jednego z nich jest równa odwrotno±ci prawdopodobie«stwa jego zaj±cia. Ich suma dla wszystkich rezultatów wynosi wi¦c w ka»dym momencie w przybli»eniu 1. Jak zostaªo wspomniane mo»liwe jest zagranie za danym rezultatem (wygrana w przypadku jego wystıpienia) lub przeciwko (wygrana w przypadku ka»dego innego rezultatu). Gra polega na wystawieniu oferty na danej opcji za okre±lonı stawk¦ z odpowiednim kierunkiem. Oferta pozostaje na rynku a» do przyj¦cia jej przez innego gracza lub do momentu zamkni¦cia rynku bıd¹ wycofania jej przez wystawiajıcego. Oferta mo»e zosta¢ przyj¦ta przez innego uczestnika gry, je±li my±li on dokªadnie odwrotnie i wystawia on zakªad po tej samej cenie, ale w odwrotnym do kierunku naszej oferty.

Aby lepiej zrozumie¢ opisany mechanizm, zaprezentowany zostanie przy-kªad opisanej transakcji zaczerpni¦ty z artykuªu . Gracz 1 przewiduje, »e

pewne wydarzenie sportowe zako«czy si¦ rezultatem ω. Stawia on zakªad typu back za stawk¦ µ. Aby to uczyni¢ akceptuje aktualnie najlepszı ocze-kujıcı ofert¦ lay, bıd¹ proponuje swojı cen¦ ρ. Gracz 2, który uwa»a, »e wydarzenie nie zako«czy si¦ rezultatem ω mo»e przyjı¢ ofert¦ gracza 1, bıd¹ wystawi¢ wªasnı. W przypadku przyj¦cia istniejıcej oferty gracz 2 wystawia zakªad typu lay za cen¦ ρ. Cena tego zakªadu to wtedy ρ - 1 razy liczba

jednostek, za którı gra. Po zako«czeniu wydarzenia, którego dotyczy rynek zyski obu graczy b¦dı wyglıda¢ nast¦pujıco:

p(gracz 1) =

µ(ρ − 1)
je±li ω to rezultat wydarzenia,
(5.1)

−
µ
w przeciwnym wypadku.

p(gracz 2) =
−µ(ρ − 1)
je±li ω to rezultat wydarzenia,
(5.2)

µ
w przeciwnym wypadku.

Istnieje mo»liwo±¢ odsprzeda»y zakupionego zasobu po korzystnej (lub te» nie) cenie. W tym celu nale»y wystawi¢ ponownie ofert¦ na tej samej opcji w przeciwnym kierunku. Od ró»nicy kursów obu transakcji b¦dzie zale»e¢ ewentualny zysk lub strata. Ich warto±ci mo»liwe s¡ do wyliczania za pomoc¡ równania

µ(ρ1
− ρ2)
je±li pierwszy zakªad to lay oraz

ωto nie jest ko«cowy rezultat,

p =

µ(ρ2

ρ1)
je±li pierwszy zakªad to back oraz

to ko«cowy rezultat

−

ω

,

0w przeciwnym wypadku.
(5.3)
gdzie p oznacza prot (strat¦ je±li warto±¢ ujemna), µ stawk¦, czyli sum¦ zainwestowanych jednostek pieni¦»nych w zakªad, ρ1 to cena zakupu zakªadu, a ρ2 to cena jego odsprzeda»y.

Niemniej jednak, je±li wynik wydarzenia b¦dzie ró»ny od tego, który byª przedmiotem transakcji, gracz nic nie zyska, ale te» nic nie straci. Po»¡danym zabiegiem w tym przypadku jest podziaª zysku lub straty mi¦dzy wszystkie

opcje. Prot lub strata si¦ zmniejszy, ale za to nie b¦dzie zale»ny od ko«co-wego rezultatu wydarzenia.

Aby tego dokona¢ nale»y stawk¦ sprzeda»y powi¦kszy¢ dodatkowo o nie-wielk¡ jej cz¦±¢ wyliczaj¡ zgodnie z wzorem 5.4.
µ2
=
ρ2

(5.4)

µ
ρ1
je±li pierwszy zakªad to
back ,

µρ1
je±li pierwszy zakªad to
lay .

ρ2

Warto±¢ protu po takiej operacji mo»na wyliczy¢ przy pomocy poni»-szych równa«.
p =
µ(ρ2

2
ρ1)

(5.5)

µ(ρ1
−ρ2)
je±li pierwszy zakªad to back ,

ρ1

.

ρ

–

je±li pierwszy zakªad to lay

Jak mo»na zaobserwowa¢, ró»nice w cenach podejmowanych zakªadów ozna-czaj¡ prot. Skuteczno±¢ w prognozowaniu tendencji zmian kursu jest klu-czowa w czerpaniu regularnych zysków na tego typu rynkach.

Opisane procesy zostaªy przedstawione tylko w celu informacyjnym. Two-rzona aplikacja nie obejmuje mechanizmów przeprowadzania transakcji na gieªdzie, umo»liwia jedynie symulacj¡ takiej gry na rynkach archiwalnych. Znaj¡c przedstawione mechanizmy gieªdowe oraz wykorzystuj¡c stworzony moduª do uczenia zostaªa zaprogramowana symulacja samodzielnego gracza-bota zdolnego do podwy»szania skuteczno±ci swojej gry na podstawie zdobywane-go do±wiadczenia.

5.3. Charakterystyka danych gieªdowych

Posiadane dane dotycz¡ ponad 400 rynków typu Kurs meczowy dla me-czów piªkarskich rozgrywanych w przeci¡gu miesi¦cy marzec-czerwiec w po-pularnych ligach europejskich. Dane te zawieraj¡ mi¦dzy innymi nazw¦ wy-darzenia sportowego, dat¦ jego rozpocz¦cia czy sum¦ przetransferowanych na rynku pieni¦dzy. Najwa»niejszym ich elementem z punktu widzenia rozwi¡zy-wanego problemu jest lista zmian kursu dla

wszystkich opcji w ramach rynku. Rozkªad tych zmian pochodzi z okresu do pi¦ciu dni wstecz od rozpocz¦cia meczu.

Dane przechowywane s¡ w formacie JSON (JavaScript Object Notation), cz¦sto wykorzystywanym do reprezentacji danych. Jego struktur¦ stanowi¡ obiekty zªo»one z par klucz-warto±¢. Format pozwala tworzy¢ struktury za-gnie»d»one skªadaj¡ce si¦ z wielu obiektów, a tak»e list. Jego format przypo-mina literaªy sªowników oraz list w j¦zyku Python. Ka»dy rynek reprezen-towany jest przez pojedynczy obiekt JSON. Przykªadow¡ struktur¦ takiego obiektu przedstawia listing 5.1. Opis pól obiektu rynkowego zostaª zawarty w tabeli 5.2.

Dane archiwizowane s¡ w bazie danych opartej na systemie MongoDB, który pozwala przechowywa¢ je w postaci obiektów JSON. Alternatywnym sposobem ich zapisu s¡ pliki tekstowe.

34:9276009858

Listing 5.1: Przykªadowy obiekt rynku.

```
{
"_id" : "1.118452104",
"marketStartTime" : ISODate("2015-05-05T20:45:00.000Z"),
"competitionName" : "UEFA Champions League",
"eventName" : "Juventus v Real Madrid",
"totalMatched" : 4099833.45,
"marketName" : "Match Odds",
"collectStartTime" : ISODate("2015-04-30T21:43:49.152Z"),
"runners" : [
{
"runnerName": "Juventus",
"runnerId": "2423",
"oddChanges" : [
```

```
{
"collectDelay" : 0.00,
"oddLink" : 3.25
},
{
"collectDelay" : 623.60,
"oddLink" : 3.15
},
{
"collectDelay" : 3366.69,
"oddLink" : 3.20
}
]
},
{
"runnerName": "Real Madrid",
"runnerId": "2426",
"oddChanges": [
{
"collectDelay" : 0.00,
"oddLink" : 2.58
},
{
"collectDelay" : 451.60,
"oddLink" : 2.60
```

```
},
{
"collectDelay" : 30959.29,
"oddLink" : 2.62
}
]
},
{
"runnerName": "The Draw",
"runnerId": "58805",
"oddChanges": [
{
"collectDelay" : 0.00,
"oddLink" : 3.35
},
{
"collectDelay" : 2411.50,
"oddLink" : 3.30
},
{
"collectDelay" : 2432.60,
"oddLink" : 3.35
}
]
}
]
```

}

31

35:2603630528

Tabela 5.2: Opis pól w strukturze obiektu reprezentuj¡cego rynek.

Nazwa pola Opis przechowywanej warto±ci
_id Identykator rynku w systemie gie³dowym.
marketStartTime Czas rozpocz¦cia wydarzenia, którego dotyczy rynek.
competitionName Nazwa rozgrywek, w ramach których przeprowadzane
jest wydarzenie sportowe, którego dotyczy rynek.
eventName Nazwa wydarzenia sportowego.
totalMatched ¡czna kwota przetranserowanych na rynku pieni¦dzy

(warto±¢ podana w walucie Euro).

marketName
Typ rynku.

collectStartTime
Czas rozpocz¦cia zbierania danych na temat rynku.

runners
Lista zawieraj¡ca informacje na temat wszystkich opcji

na rynku.

runnerName
Nazwa opcji.

runnerId
Identykator opcji w systemie gie³dowym.

oddChanges
Lista obiektów zawieraj¡cych informacje o zmianach

kursu opcji. Ka»dy obiekt dotyczy jednej zmiany.

Ró»nica momentu kolejnej zmiany kursu i czasu, w któ-

collectDelay
rym rozpocz¦to zbieranie informacji na temat rynku

(pole collectStartTime) podana w sekundach.

oddLink
Cena opcji po zmianie.

5.4. Elementy projektu

Projekt aplikacji zosta³ podzielony na trzy modu³y: ml_algos, ml_simulation,
ml_utils.

5.4.1. Modu³ z implementacjami algorytmów uczenia si¦

W module ml_algos zosta³a zawarta implementacja omawianych w tej pra-cy algorytmów ró»nic czasowych Q-learning i Sarsa obejmuj¡ca mechanizm ±ladów aktywno±ci. Celowo zosta³a oddzielona od kodu symulacji gie³dowej, aby mog³a by¢ w prosty sposób zastosowana do innych problemów, nie tylko tych opisywanych w tej pracy. Bardziej szczegó³owy opis korzystania z tego modu³u zawarto w sekcji 5.6.

36:9550634483

ml_utils

Implementacja powsta³a na bazie algorytmów zawartych w ksi¡»ce [9]. W obecnej formie mo»e ona by¢ u»yta do nauki w ±rodowiskach o charakterze stochastycznym, dla których dodatkowo istnieje trudno±¢ w zdeniowaniu modelu. Dzi¦ki zastosowaniu ±ladów aktywno±ci algorytmy dobrze spisuj¡ si¦ w przypadkach, gdy konsekwencje podejmowanych decyzji nie s¡ mo»liwe do ocenienia w ka»dym kroku nauki.

5.4.2. Modu³ narz¦dzi pomocniczych
Modu³ zawiera wszelkie narz¦dzia pomocnicze, z których ko-

rzystaj¡ pozosta³e cz¦±ci aplikacji. Dotycz¡ one g³ównie wczytywania da-nych u»ywanych w symulacji oraz ich przetwarzania i wizualizacji. Elementy
modu³u to common_utils, data_loading, data_visualizer, graph_generator, message,
simulation_data_generator , simulation_utils.

Skrypt data_loading udost¦pnia dwie klasy pozwalaj¡ce na wczytywanie danych z pliku JSON b¡d¹ bazy

danych MongoDB. Dodatkowo mie±ci si¦ tam klasa odpowiedzialna za aktualizacj¦ lokalnej bazy danych o dane znajduj¡ce si¦ w bazie zdalnej, w której gromadzone s¡ wszystkie dane udost¦pniane przez gie²d¦ i wykorzystywane w aplikacji.

Skrypt data_visualizer to aplikacja okienkowa, której zadaniem jest wizuali-zacja danych na wykresach. Zosta²a stworzona w celu werykacji poprawno±ci zbieranych danych oraz obrazowania zmienno±ci kursu na rynkach dla u²a-twienia pracy nad tworzon¡ symulacj¡. Szczegó²owy opis aplikacji znajduje si¦ sekcji 5.5.

W skryptach common_utils oraz simulation_utils zgrupowano szereg metod statycznych kolejno ogólnych, dotycz¡cych zapisu do pliku czy przetwarzania tekstów komunikatów oraz tych zwi¡zanych z symulacj¡, które pomagaj¡ g²ównie w przetwarzaniu danych.

Kolejny skrypt w tym module to simulation_data_generator , który s²u»y do generowania sztucznych danych rynkowych o okre±lonym modelu. Tego typu uproszczone dane s¡ przydatne do analizy symulacji i dopasowywania parametrów algorytmu nauki. Opis jego uruchamiania zawarty jest sekcji A.3.
Skrypt graph_generator przejmuje ca²kowicie odpowiedzialno±¢ generowa-nia wykresów, a message zawiera sta²e tekstowe wykorzystywane w projekcie.

5.4.3. Modu² symulacji

Modu² ml_simulation jest odpowiedzialny za przeprowadzenie procesu sy-mulacji. Nie obejmuje on dzia²a« zwi¡zanych z uczeniem si¦, do tego celu wy-korzystuje osobny modu² ml_algos. Zadaniem tego modu²u jest dostarczenie danych do symulacji, koordynacja symulacji o zdeniowanych parametrach, nast¦pnie udost¦pnienie wyniku w postaci tekstowej, a w niektórych przypad-kach tak»e wykresu. Skrypty wchodz¡ce w sk²ad modu²u to: merged_simulation,
simulation, simulation_launcher, simulation_result.

G²ówny skrypt tego modu²u to simulation, w którym zawarte s¡ klasy

SingleMarketSimulation oraz AllMarketsSimulation odpowiedzialne kolejno za prze-prowadzenie symulacji gry na jednym rynku oraz za koordynacj¦ tego procesu dla wszystkich jednostek rynkowych.

37:8454732675

Do uruchamiania symulacji s²u»y skrypt simulation_launcher. Pe²na in-strukcja dotycz¡ca konguracji oraz u»ytkowania znajduje si¦ w sekcji A.5.

Skrypt simulation_result jest odpowiedzialny za przetworzenie rezultatu sy-mulacji oraz wy±wietlenie jej wyniku. Poniewa» algorytmy zawieraj¡ pewien czynnik losowy zaistnia²a potrzeba uruchamiania ich wielokrotnie oraz u±red-niania rezultatów, w celu oceny efektywno±ci algorytmów. Za zebranie wyni-ków wielu symulacji oraz ich u±rednienie odpowiada skrypt merged_simulation.

5.5. Wizualizacja danych

Pierwszi z aplikacji stworzonych w ramach niniejszej pracy jest narz¦dzie pomocnicze pozwalaji ce na przedstawienie danych rynkowych na wykresach. Aplikacja udost¦pnia interfejs graczny, który zosta ª zbudowany przy pomo-cy standardowej biblioteki j¦zyka Python - Tkinter, natomiast do generowa-nia wykresów wykorzystano modu ª Matplotlib.

Interfejs aplikacji dzieli si¦ na dwie cz¦±ci. Po lewej stronie umieszczona jest lista wydarze« sportowych zwi¡zanych z dost¦pnymi danymi rynkowymi oraz kontrolki obs ªuguj ce funkcjonalno±¢ aplikacji. Poni»ej ich lista:

Kontrolki typu radio do wyboru typu ¹ród ªa danych. W opublikowanej wersji jedynym dost¦pnym rodzajem jest plik JSON.

Przycisk aktualizacji lokalnej bazy danych danymi z bazy zdalnej, do której bezpo±rednio spªywaji dane gie ªdowe. Opcja jest domy±lnie zablo-kowana.

Lista wyboru wªa±ciwo±ci, po której ma by¢ sortowana lista rynków oraz przycisk odwracania kolejno±ci.

Pole checkbox, którego wª¡czenie powoduje uwzgl¦dnienie czasu zmiany ceny na wykresach.

Pole checkbox, którego wª¡czenie powoduje wy±wietlenie wska¹nika wst¦g Bollingera na wykresach.

Przycisk do generowania obecnie wy±wietlanych wykresów do plików gra-cznych.

Po wybraniu nazwy wydarzenia, którego dotyczy rynek w prawym panelu wy±wietlaj si¦ wykresy dotycz ce danych ze wszystkich opcji. Instrukcja uruchamiania aplikacji zosta ªa zawarta w sekcji A.4. Rysunek 5.3 przedstawia okno aplikacji.

Rysunek 5.3: Okno aplikacji do wizualizacji danych gieªdowych na wykresach.

ródªo: opracowanie wªasne

5.6. Korzystanie z implementacji algorytmu nauki

Przygotowana symulacja wykorzystuje implementacje dwóch algorytmów z rodziny metod ró»nic czasowych - Q-learning i Sarsa. Caªo±¢ obsªugi pro-cesu nauki znajduje si¦ w skrypcie td_learning.py. Zawarta jest w nim deni-cja klasy gªównej TDAgent. Aby przeprowadzi¢ proces nauki nale»y utworzy¢ obiekt tej klasy za pomoc¡ udost¦pnionego konstruktora, a nast¦pnie wy-woªywa¢ na nim zdeniowane metody wedªug scenariusza, który zostanie opisany w dalszej cz¦±ci tej sekcji. Przykªadow¡ konstrukcj¦ obiektu klasy TDAgent przedstawia listing 5.2:

35

39:4155231952

e_tracing

Listing 5.2: Przykªadowa konstrukcja obiektu agenta.

```
1
import
numpy
```

```
2
import
ml_project . ml_algos . t d _ l e a r n i n g  a s  td
```

```
3
```

```
4
q = numpy . a r r a y ( [
```

```
5
[ 0 . 0 ,  0 . 0 ,  0 . 0 ] ,
```

```
6
[0.0, 0.0, 0.0],
7
[0.0, 0.0, 0.0],
8
[0.0, 0.0, 0.0]
9
])
10
11
e_tracing = td.ETrace(tracing_type=td.ETracingMethod.WATKINS,
12
factor=0.4)
13
14
qa = td.TDAgent(q, learning_rate=0.4,
```

```
15
discount_factor=0.9,
16
chance_to_explore =0.05,
17
e_tracing=e_tracing,
18



alg=td.TDAlgorithm.Q_LEARNING,
19
terminal_states=[])
```

Jak wida¢ konstruktor przyjmuje szereg argumentów. Pierwszym z nich jest macierz warto±ci dla par stan-akcja, która technicznie mo»e by¢ list¡ dwuwymiarow¡ lub obiektem typu numpy.ndarray modu³u NumPy. Ka»dy z wierszy oznacza stan w jakim mo»e znale¹¢ si¦ agent w trakcie nauki. Ko-lumny natomiast oznaczaj¡

zakres akcji jakie mo»e wykona¢ agent. Z punktu widzenia modu ªu ucz¡cego nie ma znaczenia jakie s¡ to stany oraz akcje. S¡

one traktowane kolejno jako liczby z zakresu [0, m − 1] oraz [0, n − 1], gdzie m to liczba wierszy, a n to liczba kolumn we wspomnianej macierzy. Liczby

w poszczególnych komórkach oznaczaj¡ warto±¢ akcji podejmowanej w okre-±lonym stanie. Je±li w którymkolwiek ze stanów nie ma mo»liwo±ci podj¦cia pewnej akcji, to warto±¢ dla tej pary ma by¢ ustalona na sta ª¡ numpy.NAN. Standardowo macierz powinna by¢ inicjalizowana zerami, które w procesie nauki b¦d¡ aktualizowane nowymi warto±ciami.
Kolejne parametry konstruktora to:

learning_rate - warto±¢ α ze wzorów 4.44 oraz 4.45,
discount_factor - warto±¢ γ ze wzorów 4.44 oraz 4.45,
chance_to_explore - warto±¢ ε w strategii ε-zach ªannej.

Argument odpowiada za metod¦ ±ladów aktywno±ci. Jest on

opcjonalny i mo»e przyjmowa¢ warto±¢ None, w przeciwnym wypadku nale»y przekaza¢ obiekt klasy ETrace. W linii 11 powy»szego listingu pokazano przy-k ªadow¡ konstrukcj¦. Klasa pe ªni rol¦ opakowania dla dwóch warto±ci - me-
tody ±ladów aktywno±ci oraz wspó ªczynnika ±wie»o±ci λ. Pierwszy parametr

podajemy jako liczb¦ ca ªkowit¡: 0 - metoda Watkinsa, 1 - metoda naiwna. W celach pomocniczych stworzono klas¦ ETracingMethod, która zawiera sta ªe

z tymi warto±ciami. Drugi parametr to liczba rzeczywista z przedzia ªu [0, 1], która ustala wspomniany wspó ªczynnik λ.

Kolejny parametr konstruktora okre±la algorytm nauki. Przyjmuje war-to±¢ 0 dla Q-learning lub 1 dla Sarsa. Obie sta ªe s¡ okre±lone w klasie
TDAlgorithm.

40:2782461479

Po stworzeniu obiektu typu TDAgent nale»y zdeniowa¢ w nast¦puj¡cy

sposób stan pocz¡tkowy agenta:

```
td_agent.set_initial_state(0)
```

Na tym ko«czy si¦ etap inicjalizacji agenta. W tym momencie mo»liwe jest rozpocz¦cie iteracyjnego procesu nauki. Na listingu 5.6 przedstawiono proces skªadajicy si¦ z 10 kroków.

```
1
for step in xrange(10):

2
chosen_action = td_agent.choose_action_to_perform()

3

4reward = generate_reward(chosen_action)

5
td_agent.provide_reward_for_last_action(reward)
6

7
next_state = determine_next_state(chosen_action)
```

8
td_agent . go_to_next_state (n e x t _ s t a t e)

Na początku kaŻdej iteracji należy wywołać metodę choose_action_to_perform , która wybiera akcję do podjęcia w tym kroku czasowym oraz zwraca liczbę całkowitą oznaczającą identykator akcji. Następnym krokiem jest wykona-nie tej akcji oraz odebranie wzmocnienia ze środowiska. W listingu dzieje się to w metodzie generate_reward, za której denicję odpowiada uŻytkownik opisywanego moduału. Dla przykładu moŻe się w niej znaleźć wywołanie po-lecenia zrobienia kroku przez robota i wygenerowanie nagrody na podstawie tego, na jaką odległość po jego wykonaniu przybliŻył się do okreŚlonego celu. W przypadku tworzonej symulacji nagroda jest generowana tylko po wykona-niu akcji zamknięcia zakładu i jest równa wartości otrzymanego zysku bądź straty.

Otrzymane wzmocnienie przekazywane jest do agenta za poŚrednictwem metody provide_reward_for_last_action . Oprócz wzmocnienia należy ustalić no-wy stan, w którym znajdzie się agent po wykonaniu akcji. W formie liczby całkowitej przekazujemy go do agenta metodą go_to_next_state. Jak widać w procesie nauki biblioteka bierze na siebie tylko część odpo-wiedzialności. Sprawą naturalną jest, Że uŻytkownik z niej korzystający jest zobowiązany do zaprogramowania rzeczywistej interakcji ze Środowiskiem i odbioru wzmocnień. W dodatku implementacja nie przewiduje denicji mo-delu Środowiska, czego powodem jest wymóg ustalania kaŻdorazowo nowych stanów agenta po wykonywaniu akcji.

37

41:2065204612

5.7. Projektowanie symulacji

5.7.1. Denicja stanów

W procesie deniowania stanów dla algorytmu nauki należy uwzględnić zarówno stany gracza uzależnione od

tego, czy posiada on zakªad oraz stany w jakich mo»e znale¹¢ si¦ ±rodowisko, w tym przypadku rynek gieªdowy, a ±ci±lej cena danej opcji. Po ich ustaleniu rzeczywisty zbiór stanów ma posta¢ iloczynu kartezja«skiego obydwu zbiorów.
W przypadku agenta zestaw stanów zdeniowano nast¦puj¡co:

brak pozycji,
pozycja przewiduj¡ca spadek kursu,

pozycja przewiduj¡ca wzrost kursu.

Kwestia denicji stanów ±rodowiska jest bardziej zªo»ona. Naturalnym podej±ciem wydaje si¦ by¢ zdeniowanie zakresu stanów na bazie cen, które przyjmowaªa dana opcja od chwili otwarcia rynku. Przykªadowo dla opcji, której cena na przestrzeni czasu prezentuje si¦ jak na wykresie 5.4, przedziaª

ten obejmowaªby zakres [1.48, 1.62], czyli warto±ci od minimum do maksi-

mum ceny. Daje to ª¡cznie 15 stanów ±rodowiska, wi¦c tworz¡c iloczyn kar-tezja«ski zbioru stanów agenta oraz zbioru stanów ±rodowiska otrzymujemy ª¡cznie 45 stanów. Liczba ta z pewno±ci¡ wydaje si¦ by¢ za wysoka, maj¡c na uwadze fakt, »e agent dla optymalnego dziaªania musi przetestowa¢ akcje podejmowane w ka»dym ze stanów.

Rysunek 5.4: Demonstracja sekwencji zmian ceny jednej z opcji rynkowych na wykresie.

ródªo: opracowanie wªasne

W tym podej±ciu mo»na dostrzec inne problemy jak np. zale»no±¢ zbioru stanów od opcji, co rodzi np. trudno±¢ w porównywaniu rezultatów symulacji mi¦dzy rynkami. Dodatkowo cz¦stym zabiegiem byªoby powi¦kszanie zbioru stanów o nowe warto±ci w przypadku zmian ceny na niespotykane dot¡d warto±ci. Przy ka»dorazowym doªo»eniu stanu warto±¢ akcji w nim podejmo-wanych byªaby zerowa, czyli agent nie posiadaªby o niej »adnej wiedzy.

42:8019304974

Rysunek 5.5: Wst¦gi Bollingera na wykresie zmiany ceny.

ródªo: opracowanie wªasne

Warto w tym przypadku zastosowa¢ tzw. wska¹niki analizy technicznej, które powstaªy z my±li o opisywaniu stanów rynków nansowych. Jednym z nich si wst¦gi Bollingera oparte na zmienno±ci cen. Zakªada on, »e cena porusza si¦ w obr¦bie trzech wst¦g:
±rodkowej - b¦d¡cej n-okresow ±redni ruchomi,

górnej - b¦d¡cej k-krotno±ci n-okresowego odchylenia standardowego po-wy»ej ±rodkowej wst¦gi,

dolnej - b¦d¡cej k-krotno±ci n-okresowego odchylenia standardowego po-
ni»ej ±rodkowej wst¦gi.

Na wykresie przedstawiono opisane wst¦gi dla sekwencji cen pewnej opcji rynkowej. Stany rynku mo»na okre±li¢ na podstawie tego, w którym z prze-dziaªów wyznaczonych przez wst¦gi mo»e znale¹¢ si¦ cena. Zdeniowane zo-staªy 4 stany:
cena powy»ej górnej wst¦gi,
cena mi¦dzy górn i ±rodkow wst¦gi,

cena mi¦dzy ±rodkow i doln wst¦gi,

cena poni»ej dolnej wst¦gi.

5.7.2. Denicja akcji

W przypadku denicji zbioru akcji mo»liwych do wykonania przez agenta warto zwróci¢ uwag¦ na akcj¦ biern, czyli brak interakcji z gieªd w danym kroku czasowym. Zdaniem autora warto rozró»ni¢ t¦ akcj¦ w

zale»no±ci od tego czy gracz aktualnie posiada zakªad.
Zdeniowany nast¦pujcy zestaw akcji:
brak interakcji z gieªdj nie posiadajc zakªadu (akcja bierna),
kupno zakªadu przewidujc spadek ceny (zakªad typu back),

kupno zakªadu przewidujc wzrost ceny (zakªad typu lay),

odsprzeda» (zamkni¦cie) zakªadu,

utrzymywanie zakªadu (akcja bierna).

Nale»y mie¢ na wzgl¦dzie fakt, »e niektórych akcji nie mo»na przeprowa-

dzi¢ w pewnych stanach. Dla przykªadu nie mo»na odsprzeda¢ zakªadu je±li

wcze±niej nie zostaª on przez nas zakupiony lub kupi¢ zakªad posiadajc ju» jeden.

5.7.3. Denicja macierzy warto±ci Q

Na podstawie zdeniowanych wcze±niej stanów oraz akcji mo»na ustali¢ poczjtkowj posta¢ macierzy Q, która zostanie przekazana do moduªu uczce-go. Ka»demu z trzech stanów agenta nale»y przyporzjdkowa¢ cztero-elementowny zbiór stanów ±rodowiska. Poni»ej fragment tak uformowanej listy stanów:

brak pozycji i cena powy»ej górnej wst¦gi,
pozycja przewidujca spadek kursu i cena powy»ej górnej wst¦gi,

pozycja przewidujca wzrost kursu i cena powy»ej górnej wst¦gi,

brak pozycji i cena mi¦dzy górnj i ±rodkowj wst¦gi,

pozycja przewidujca spadek kursu i cena mi¦dzy górnj i ±rodkowj wst¦gi itd.

Dokªadajc do tego zbiór akcji otrzymujemy macierz 12 ×5 zaprezento-wanj poni»ej.

NAN
NAN
NAN
0.0
0.0

0.0
0.0001
0.0001
N AN
N AN

N AN
N AN
N AN
0.0
0.0

0.0
0.0001
0.0001
NAN
NAN

N AN
N AN
N AN
0.0
0.0

Q =

N AN
N AN
N AN
0.0
0.0

(5.6)

0.0
0.0001
0.0001
NAN
NAN

NAN
NAN
NAN
0.0
0.0

N AN
N AN
N AN
0.0
0.0

0.00.0001 0.0001 N AN N AN
NAN NAN NAN 0.0
0.0

N AN N AN N AN
0.0
0.0

Nale»y zwróci¢ uwag¦ na kolumny 2. i 3., przechowuj¡ce warto±ci akcji kolejno: zagraj na spadek , zagraj na wzrost dla poszczególnych stanów. Nieze-rowa warto±¢ ma zach¦ci¢ agenta do gry ju» od pocz¡tku interakcji z gieªdi. ZakªadajÈc, »e gracz nie posiada »adnej wiedzy na temat danego rynku nie ma uzasadnienia dla jakiejkolwiek zwªoki przy rozpocz¦ciu procesu nauki. Liczba jest niewielka, aby nie wpªywa¢ na zachowanie agenta w dªu»szym horyzoncie czasowym. Dla warto±ci akcji, których nie mo»na wykona¢ w danym stanie zostaªa ustalona warto±¢ NAN (Not a Number).

5.7.4. Proces nauki

Po przekazaniu macierzy rozpoczynamy iteracyjny proces nauki analo-giczny do tego zaprezentowanego na listingu 5.6 dla ka»dej ceny z posiadanej sekwencji cen danej opcji. Za wzmocnienie uznajemy rezultat poczynionej transakcji, czyli otrzymany zysk, b¡d¹ poniesion¡ strat¦. Wynosi ono dla

44:4932809389

ka»dej akcji 0, prócz tej, odpowiedzialnej za zamkni¦cie zakªadu (kolumna 4 macierzy Q).

Za wyliczenie zysku odpowiada metoda zaimplementowana na podstawie wzoru 5.5. Stawka, któr¡ dysponuje gracz wynosi domy±lnie 10 jednostek. Metoda uwzgl¦dnia koszty prowizji ponoszone tylko w przypadku zyskownych transakcji, wynosz¡ce do 5% protu.

Agent inicjalizowany jest ze stanem brak pozycji i w ka»dym kroku usta-lany na podstawie wykonywanych

przez agenta akcji. Stan ±rodowiska jest ustalany w ka»dym kroku zgodnie z tym w jakim przedziale wyznaczonym ze pomoc¡ wst¦g Bollingera znajduje si¦ aktualna cena opcji.

5.8. Testy symulacji

5.8.1. Znaczenie i dobór parametrów algorytmu

Na przykªadzie wykonanych testów symulacji przedstawiona zostanie isto-ta parametrów algorytmów z równa« 4.44 oraz 4.45. Testy zostaªy wykonane z u»yciem sztucznych danych o modelu sinusoidy z kilkoma okresami. W ka»dym przypadku zastosowano algorytm Q-learning bez u»ycia ±ladów ak-tywno±ci. Niektóre charakterystyczne rezultaty symulacji zostaªy zaprezen-towane na wykresach. Zawieraj¡ one informacje:

o sekwencji zmian ceny bez uwzgl¦dnienia czasu zmiany - (czarna linia),
sygnaªy kupna zakªadu typu back (niebieskie punkty),

sygnaªy kupna zakªadu typu lay (czerwone punkty),

poziom salda, poczynaj¡c od 0 (zielona linia).

5.8.1.1. Prawdopodobie«stwo wykonania akcji eksploruj¡cej ε

Jako pierwszy zostanie omówiony parametr ε, od którego zale»na jest stosowana przez algorytm strategia ε-zachªanna. Oznacza ona podejmowanie

decyzji najkorzystniejszych w aktualnym kroku czasowym, robi¡c co jaki± czas wyj¡tki wykonuj¡c tzw. akcje eksploruj¡ce, które maj¡ przyspieszy¢ po-wi¦kszanie si¦ zakresu przetestowanych akcji. Wspomniany parametr oznacza prawdopodobie«stwo z jakim taka decyzja mo»e zosta¢ wybrana.

Ustalenie warto±ci tego parametru na warto±¢ 1 oznacza w praktyce caª-kowicie losowy dobór akcji w ka»dym kroku. Obrazuje to wykres , na którym saldo konta gracza nie wykazuje »adnych tendencji w jedn¡ b¡d¹ drug¡ stro-n¦.

45:5893297861

Rysunek 5.6: Rezultat symulacji wykorzystuj¡cej sztuczne dane z warto±ci¡ parametru ε ustawion¡ na 1.

ródªo: opracowanie wªasne

Parametr przyjmuj¡cy za± warto±¢ 0 skazuje algorytm na tendencyjno±¢ uzale»nion¡ od decyzji podejmowanych w pocz¡tkowej fazie nauki. W ka»-dym kroku czasowym agent wybiera zawsze akcj¦ z najwy»sz¡ warto±ci¡. Jak wida¢ na wykresie 5.7 macierz warto±ci akcji wygl¡daªa w pewnym momencie tak, »e najbardziej opªacaln¡ decyzj¡ z punktu widzenia strategii okazaªo si¦ zaprzestanie gry na gieªdzie, co te» agent uczyniª patrz¡c na utrzymuj¡cy si¦ poziom salda od okolic 580 kroku.

Rysunek 5.7: Rezultat symulacji wykorzystuj¡cej sztuczne dane z warto±ci¡ parametru ε ustawion¡ na 0.

ródªo: opracowanie wªasne

46:3595574801

Niska warto±¢ tego parametru gwarantuje dobre wyniki w pocz¡tkowej fazie nauki, ale mniej optymalne w dªu»szym horyzoncie czasowym, poniewa» ewaluacja akcji nast¦puje wolniej w miar¦ jak zaw¦»a si¦ dobór akcji. Dla testowanego modelu za warto±¢ bliski optymalnej mo»na uzna¢ 0.1.

5.8.1.2. Wspóªczynnik uczenia si¦ α

Wspóªczynnik uczenia si¦ α wpªywa na to, jak szybko nowa wiedza b¦dzie

nadpisywa¢ starsze informacje. Warto±¢ 0 spowoduje brak jakiejkolwiek na-uki, warto±¢ 1, »e agent b¦dzie braª pod uwag¦ jedynie najbardziej aktualn¡ wiedz¦ i taka te» warto±¢ b¦dzie optymalna dla ±rodowisk deterministycz-nych. Problemy o charakterze stochastycznym cz¦sto wymagaj¡ niewielkiej warto±ci parametru. Dla rozwa»anej sekwencji danych za bliski optymalnej mo»na uzna¢ warto±¢ 0.9, co w pewnym stopniu udowadnia wykres 5.8.

Rysunek 5.8: Rezultat symulacji wykorzystuj¡cej sztuczne dane z warto±ci¡ parametru α ustawion¡ na 0.9.

ródªo: opracowanie wªasne

5.8.1.3. Współczynnik dyskontowania nagród γ

Parametr γ okre±la stopie« wa»no±ci przyszªych nagród. Warto±¢ 1 powo-

duje, »e wszystkie nagrody, niezale»nie od momentu otrzymania b¦d¡ trak-towane jednakowo. Zerowa warto±¢ przyczyni si¦ do tego, »e rozwa»ane b¦-d¡ jedynie te akcje, które bezpo±rednio prowadz¡ do otrzymywania nagród. Mo»e mie¢ to niezbyt dobre konsekwencje, zwªaszcza w przypadkach, kie-dy nagrody przynosz¡ tylko niektóre z akcji. Wykres 5.9 prezentuje ciekawy przypadek z tym zwi¡zany.

47:9948298928

Rysunek 5.9: Rezultat symulacji wykorzystuj¡cej sztuczne dane z warto±ci¡ parametru γ ustawion¡ na 0.

Aby zrozumie¢, co spowodowaªo taki obrót spraw, nale»y przyjrze¢ si¦ ko«cowej postaci macierzy warto±ci Q, któr¡ prezentuje listing 5.3. Jak wida¢ warto±ci niezerowe s¡ obecne jedynie w kolumnie czwartej, przechowuj¡cej warto±ci akcji zamkni¦cia zakªadu, która jako jedyna gwarantuje nagrody. Zgodnie z istot¡ wspóªczynnika dyskontowania inne akcje nie byªy warto-±ciowane w czasie dziaªania algorytmu. Dlatego te» akcje otwarcia zakªadu w ró»nych kierunkach s¡ podejmowana równie cz¦sto, poniewa» maj¡ w ka»-dym momencie tak¡ sam¡ warto±¢. Fakt ten mo»na stwierdzi¢ na podstawie ilo±ci sygnaªów kupna - wystarczy porówna¢ ilo±¢ niebieskich i czerwonych punktów. Nieco gorzej wida¢ to na wykresie salda, gdzie momentami jego poziom wzrasta, ale prawie za ka»dym razem nast¦puje to po relatywnie krótkim utrzymywaniu zakªadu. Pó¹niej jednak wida¢ tendencj¦ do nietraf-nego wyboru mi¦dzy akcjami zamkni¦cia oraz utrzymania zakªadu. Agent w przypadku wybrania akcji kupna zakªadu, dla którego akcja zamkni¦cia posiada warto±¢ ujemn¡ ma zawsze do wyboru:
zamkni¦cie zakªadu (Qa < 0),
utrzymywanie zakªadu (Qa = 0).

Wybierana jest prawie zawsze akcja druga jako posiadaj¡ca wy»sz¡ war-to±¢. Taki zakªad jest zamykany tylko w dwóch przypadkach: wyst¡pienia polecenia wykonania akcji eksploruj¡cej b¡d¹ zmiany stanu ±rodowiska, tj. przedziaªu (wyznaczonego przez wst¦gi Bollingera) w jakim znajduje si¦ ce-na opcji. Zaistnienie tego drugiego mo»na stwierdzi¢ na podstawie kroków czasowych w jakich wyst¦puj¡ sygnaªy kupna. Skupiaj¡ si¦ one okolicach lokalnych minimów i maksimów funkcji. Wtedy dla nienotowanego wcze±niej stanu ceny gieªdowej warto±¢ akcji zamkni¦cia oraz utrzymania zakªadu ma t¦ samo, zerow¡ warto±¢.

Kiedy agent podejmuje zakªad, dla którego zamkni¦cia zwykle przynosiªy zysk (wedªug warto±ci Q), ma on nast¦puj¡ce mo»liwo±ci:
zamkni¦cie zakªadu (Qa > 0),

utrzymywanie zakªadu (Qa < 0).

Agent zawsze zamyka zakªad, poniewa» wedªug strategii jest to optymalne. Tym samym nigdy nie pozwala rosn¡¢ zyskom, co robi zawsze w przypadku zakupu potencjalnie mniej warto±ciowych zakªadów.

Listing 5.3: Ko«cowa posta¢ macierzy warto±ci Q dla symulacji na sztucznych danych uruchomionej z warto±ci¡ parametru γ ustawion¡ na 0.

```
[ [
0 .
0 .
0 .
nan
nan ]
```

[
nan
nan
nan
−0.0097
0 .]

[
nan
nan
nan
0 . 0 0 0 6
0 .]

[
0 .
0 .
0 .
nan
nan]

[
nan
nan
nan
−0.0087
0 .]

[
nan
nan
nan
0 . 0 0 0 8
0 .]

[
0 .
0 .
0 .
nan
nan]

[
nan
nan
nan
−0.0002
0 .]

[
nan
nan
nan
−0.0097
0 .]

[
0 .
0 .
0 .
nan
nan]

[
nan
nan
nan 0 . 0 0 1 7
0 .]

[
nan
nan

nan
−0.0196
0 .]]

5.8.1.4. Optymalny algorytm i parametry

Na koniec tej cz¦±ci testów skonfrontowano ze sob¡ oba algorytmy, czyli Q-learning i Sarsa równie» w wersjach ze ±ladami aktywno±ci. Tabela prezen-tuje u±rednione wyniki z 10 symulacji przeprowadzonych dla ka»dego zestawu parametrów.

Tabela 5.3: Zestawienie rezultatów symulacji wykorzystuj¡cej sztuczne dane rynkowe.

Wersja
Ogólny
Zyskowno±¢
Liczba
Zysk na
algorytmu1
zysk (j.)2
transakcji (%)3
transakcji
transakcj¦ (j.)

Q-learning
0.47
85.3
3.7
10.0
468.60
0.00088

Sarsa
0.96
74.1
2.4
23.5
332.70
0.00304

Q-learning ±. a.
0.91
83.4
4.3
12.3
421.50
0.00216

Sarsa ±. a.
0.44
71.3
2.8
25.9
428.60
0.00101

1Warto±ci parametrów: ε: 0.1, α: 0.9, γ: 1.0, λ: 0.4, typ ±. a.: Watkinsa.

2 Uzyskany w przypadku gry stawk¡ równ¡ 10 jednostek.

3 Udziaª transakcji przynosz¡cych kolejno: zauwa»alny zysk, ±ladowy zysk lub strat¦, za-uwa»aln¡ strat¦.

Najwa»niejszymi parametrami s¡ ogólny zysk oraz zysk na transakcj¦. W przypadku obu z nich algorytm Sarsa bez u»ycia ±ladów aktywno±ci góru-je nad reszt¡. Gracz z niego korzystaj¡cy handluje rzadziej od reszty, przy czym ogólny zysk jest najwy»szy. Swoistym paradoksem jest tutaj fakt, »e równie» cz¦±ciej podejmowaª on bª¦dne decyzje w porównaniu do obu wersji algorytmu Q-learning.

49:6432998399

5.8.2. Testy symulacji na rzeczywistych danych

Tabela 5.4 przedstawia wyniki symulacji dla ró»nych wersji algorytmów. Dla ka»dej z nich zaprezentowano u±rednione wyniki z 5 przeprowadzonych symulacji z u»yciem wszystkich posiadanych danych rynkowych. Uwzgl¦dnio-no tylko te opcje, dla których liczba zaobserwowanych zmian ceny wynosiªa przynajmniej 100, a ±rednia cena nie przekraczaªa warto±ci 2.7. Pierwsze kry-terium miaªo na celu wyeliminowanie relatywnie maªo popularnych rynków. Wymaganie co do ±redniej ceny wprowadzono dlatego, »e wy»sze kursy mog¡ generowa¢ nieproporcjonalnie wysokie zyski lub straty, co mogªoby w du»ym stopniu zaciemni¢ rezultat symulacji. Z tej przyczyny gracz nie podjiª gry jedynie na niewielkiej cz¦±ci. Warto±ci wszystkich parametrów dobrano na podstawie wykonywanych testów i s¡ one bliskie optymalnym. Warto zwróci¢ uwag¦ na warto±¢ parametru ±wie»o±ci α, która jest znacznie ni»sza od tej,

która byªa optymalna w przypadku poprzedniego modelu danych. Wskazuje ona na znacznie bardziej losowy charakter zmienno±ci ceny w przypadku rzeczywistych danych gieªdowych.

Tabela 5.4: Zestawienie rezultatów symulacji gry na gieªdzie zakªadów spor-towych wykorzystuj¡cej archiwalne dane.

3

5

Wersja algorytmu1
Ogólny zysk (j.)2

Zyskowno±¢ transakcji (%)

Zyskowno±¢ rynków (%)4

Liczba transakcji na rynek
Zysk na transakcj¦ (j.)

Q-learning
2034.4
70.3
12.3
17.5
80.6
0.0
19.4
129.2
0.0391

Sarsa
1526.1
66.5

15.0
18.5
80.3
0.0
19.7
108.9
0.0348

Q-learning ±. a.
2377.7
70.9
13.7
15.4
87.7
0.0
12.3
141.4
0.0417

Sarsa ±. a.
1676.1
65.6
17.4
17.1
86.4
0.0
13.6
114.7
0.0362

1Warto±ci parametrów: ε: 0.05, α: 0.1, γ: 1.0, λ: 0.4, typ ±. a.: Watkinsa.

2 Uzyskany w przypadku gry stawk¡ równ¡ 10 jednostek.
3 Udziaª transakcji przynosz¡cych kolejno: zauwa»alny zysk, ±ladowy zysk lub strat¦, za-

uwa»aln¡ strat¦.
4 Udziaª rynków, na których transakcje ogóªem przyniosªy kolejno: zauwa»alny zysk, ±la-

dowy zysk lub strat¦, zauwa»aln¡ strat¦.
5 rednia liczba transakcji na rynek.

Z zaª¡czonego zestawienia wynika, »e u»ycie ±ladów aktywno±ci faktycz-nie wpªywa na polepszenie jako±ci procesu nauki. Lepiej poradziª sobie z

50:2823832810

zadaniem algorytm Q-learning, co wskazuje, »e dla tego typu danych jego formuªa aktualizacji jest bardziej odpowiednia. Nie do±¢, »e przeprowadzaª on transakcje zauwa»alnie cz¦±ciej ni» inne, to ±redni zysk na transakcj¦ byª wy»szy. Warto porówna¢ w przypadku ka»dego algorytmu ró»nice udziaªów z kolumn 3. i 4. Informuje ona na to, »e graczom faktycznie udaje si¦ zwi¦kszy¢ skuteczno±¢ w czasie gry na rynku.

51:6345434864

6. Podsumowanie

Celem pracy byªo zbudowanie aplikacji symulujicej gr¦ na gieªdzie zakªa-dów sportowych z wykorzystaniem algorytmów uczenia si¦ ze wzmocnieniem. Przedstawiono w sposób formalny ogólne zasady, wedªug których dziaªaj¡ konkretne algorytmy tego typu. Przybli»ono proces decyzyjny Markowa ja-ko model matematyczny zadania uczenia ze wzmocnieniem. W jego ramach zdeniowano autorski uproszczony model gieªdy, podkre±laj¡c niedetermini-styczny charakter tego ±rodowiska.

Maj¡c na wzgl¦dzie rozwi¡zywany problem skupiono si¦ na klasie metod ró»nic czasowych jako stosunkowo nowym podej±ciu w tym rodzaju uczenia si¦ maszyn. Przedstawiono mo»liwie rozbudowane wersje algorytmów kon-kretnych - Q-learning oraz Sarsa. Uwzgl¦dniono w nich wspóªczynnik ucze-
nia si¦ α, którego zmniejszona warto±¢ odgrywa wa»n¡ rol¦ w ±rodowiskach

o du»ym stopniu losowo±ci. Opisano ±lady aktywno±ci jako remedium dla problemu odroczonych nagród (ang. delayed rewards).

Przy u»yciu j¦zyka Python oraz narz¦dzi pomocniczych zaimplemento-wano wymienione algorytmy. Stworzony moduª nauki zostaª wykorzysta-ny do zbudowania symulacji przeprowadzania transakcji gieªdowych przez gracza-bota. By to osi¡gn¡¢ szczególowo zdeniowano charakter ±rodowiska pracy agenta ucz¡cego si¦ oraz mo»liwe interakcje.

Skuteczno±¢ nauki zbadano na rzeczywistych danych gieªdowych oraz na wygenerowanych uproszczonych sekwencjach zmian kursu. Przeprowadzono testy porównuj¡ce ró»ne wersje algorytmów. Równie» na ich podstawie okre-±lono rol¦ oraz dobór poszczególnych parametrów nauki. Wyniki przeprowa-dzanych symulacji prezentowano nie tylko w formie statystyk, ale tak»e w postaci wykresów. Mimo wprowadzonych uproszcze« mo»na stwierdzi¢, »e zastosowane algorytmy bardzo dobrze poradziªy sobie z tym zadaniem. W miar¦ post¦pu prac zauwa»ono potencjalne usprawnienia, które mogªyby si¦ przyczyni¢ do uzyskania jeszcze lepszych wyników.

Relatywnie prostym do wprowadzenia ulepszeniem byªaby zmiana warto-±ci parametrów takich jak α i λ w miar¦ post¦powania procesu nauki, gdy»
obecnie zainicjowane warto±ci pozostaj¡ niezmienne. Do rozwa»enia byªo-by równie» u»ycie bardziej zªo»onej polityki wyboru akcji ni» zastosowana

ε-zachªanna, np. strategii Boltzmanna. Bardziej zªo»onym ulepszeniem byªo-

by daleko id¡ce rozbudowanie procesu nauki agenta. W obecnym rozwi¡zaniu obejmuje on gr¦ tylko i wyª¡cznie na jednym rynku. W tej chwili wi¦c za ka»-dym razem gr¦ rozpoczyna nowy, niedo±wiadczony acz skory do nauki gracz. Naturalnym usprawnieniem byªoby stworzenie mechanizmu czerpania ogólnej i uniwersalnej wiedzy, pozwalaj¡cego wyci¡ga¢ wnioski z dªugookresowej gry na wielu rynkach.

J¦zyk Python dobrze sprawdziª si¦ jako narz¦dzie do implementacji opi-

52:5040200935

sywanych algorytmów. Stworzyª on jednak»e pewne ograniczenia zwi¡zane z wydajnym przeprowadzanych symulacji. Dla ka»dego z rynków zgodnie z u»ywanym modelem nauki mo»e ona by¢ wykonana niezale»nie, czego nast¦p-stwem byªo podj¦cie próby zrównoleglenia tego procesu. Niestety wykorzy-stywany interpreter j¦zyka Python uniemo»liwia rozwi¡zanie tego problemu w tradycyjnym podej±ciu, czyli posªugujc si¦ w¡tkami (ang. threads). Dzieje si¦ to z powodu zastosowania tzw. Global Interpreter Lock (GIL), który jest swoistym zabezpieczeniem przed wykonywaniem wielu w¡tków jednocze±nie. Jest to niestety jeden z mankamentów najpopularniejszej implementacji j¦-zyka Python (nieobecny np. w Jython czy IronPython) i mimo, »e istniej¡ rozwi¡zania zast¦pcze, to wedªug autora nie na tyle atrakcyjne, aby je zasto-sowa¢.

Autor pracy jest ±wiadom faktu, »e niniejsza praca jest jedynie wst¦pem do jak»e rozlegªego obszaru uczenia maszynowego, niemniej jednak stanowi zach¦t¦ do prowadzenia dalszych bada« w tej dziedzinie.

53:1063815326

data_visualizer.py

simulation_launcher.py

SOURCE_PATH

A. Uruchamianie projektu

Rozdziaª zawiera peªn¡ instrukcj¦ konguracji i u»ytkowania wszystkich aplikacji stworzonych w ramach tej pracy.

A.1. Wymagania

Wymagania co do ±rodowiska uruchomieniowego zostaªy zdeniowane na podstawie tego, na jakich systemach operacyjnych oraz przy u»yciu których wersji zastosowanych narz¦dzi udaªo si¦ uruchomi¢ wszystkie aplikacje w ra-mach projektu. W przypadku tych drugich zostaªy podane najstarsze wersje, dla których nie stwierdzono problemów.

system operacyjny z rodziny Windows lub Linux (wraz ze ±rodowiskiem gracznym),

interpreter j¦zyka Python w wersji 2.7 (wymagana jego ±cie»ka w zmiennej ±rodowiskowej PATH),
biblioteka NumPy w wersji 1.6,

biblioteka Matplotlib w wersji 1.1.0.

Dodatkowym wymaganiem jest dodanie do zmiennej systemowej PY-THONPATH ±cie»ki z aplikacji. Mo»na to wykona¢ otwierajic terminal od-powiedni dla posiadanego systemu operacyjnego oraz wykonujic polecenie

podane poni»ej. Przykªady te zakªadaji, »e to katalog nad-

rz¦dny wzgl¦dem folderu ml_project:
w systemie Linux:

```
SOURCE_PATH=/home/mgr−j a n c a r z / s r c /main
e x p o r t PYTHONPATH=${PYTHONPATH} : ${SOURCE_PATH}
```

w systemie Windows:

```
set SOURCE_PATH=C: \ mgr−j a n c a r z \ s r c \main
set PYTHONPATH=%PYTHONPATH%;%SOURCE_PATH%
```

A.2. Konguracja ¹ródªa danych rynkowych

Aplikacje oraz mogi korzysta¢ z

dwóch typów ¹ródeª z danymi rynkowymi - bazy danych MongoDB oraz pliku JSON. Poniewa» pierwszy typ wymaga dodatkowej instalacji systemu bazodanowego wraz z aplikacj¡ dostarczone s¡ pliki z danymi. Domy±lnie wykorzystywane s¡ 2 pliki:

54:4857909766

marketData−all.json - plik zawierajicy rzeczywiste archiwalne dane doty-cz¡ce 438 rynków sportowych,

marketData−mock.json - plik ze sztucznymi danymi generowanymi skryptem data_generator.py (jego opis znajduje si¦ w sekcji A.3).

Konguracja dotyczi¡ca ¹ródªa danych oraz innych ustawie« dla obu apli-kacji zawarta jest w pliku settings . cfg zaprezentowanym na listingu A.1.

Listing A.1: Domy±lna zawarto±¢ pliku konguracyjnego settings . cfg .

```
[DB A c c e s s ]

db_server_ip =

db_port =

db_name =

[ C o n f i g u r a t i o n ]

allow_db_as_data_source = F a l s e

data_source_type = f

d a t a _ s o u r c e _ f i l e n a m e = marketData −a l l . j s o n

;          d a t a _ s o u r c e _ f i l e n a m e = marketData −mock . j s o n

a l p h a = 0 . 4

gamma = 1 . 0

e p s i l o n = 0 . 1

a l g = Q

markets_to_process = −1
simulations_to_run = 1

time_steps_to_omit = 1
```

Nale»y zwróci¢ uwag¦ na poni»sze zmienne:

allow_db_as_data_source : Ustala mo»liwo±¢ wykorzystywania bazy danych jako ¹ródªa danych (dotyczy aplikacji data_visualizer.py). Domy±lnie usta-wiona na False i tak¡ warto±¢ nale»y pozostawi¢.
data_source_type: Typ wykorzystywanego ¹ródªa danych. Dozwolone s¡

warto±ci: d (baza danych) oraz f (plik JSON). Nale»y pozostawi¢ domy±l-n¡ warto±¢ zmiennej.

data_source_lename: Nazwa pliku z danymi, z którego korzysta aplika-cja, je±li ustawiony typ ¹ródªa danych to plik. Przewidziane warto±ci tej

zmiennej to marketData−all.json lub marketData−mock.json.

A.3. Uruchamianie generatora sztucznych danych

Skrypt simulation_data_generator.py pozwala generowa¢ sztuczne dane ryn-kowe. Po jego uruchomieniu w pliku marketData−mock.json zapisywany jest 1
obiekt z danymi. Mo»liwe jest wygenerowanie danych w jednym z trzech pre-deniowanych modeli danych. Typ modelu ustawiany jest za pomoc¡ atrybu-

tu −m/−−model, natomiast liczb¦ warto±ci do wygenerowania ustala atrybut −n/−−number (domy±lna warto±¢ to 2000).

55:5419200416

result . txt .

settings . cfg .

Obsªugiwane warto±ci atrybutu modelu danych:

L/l - funkcja liniowa malej¡ca,
G/g - rozkªad Gaussa,
S/s - sinusoida.

Dane zostaj¡ wygenerowane po wywoªaniu poni»szego polecenia:

```
python simulation_data_generator.py [opcjonalne argumenty]
```

A.4. Aplikacja wizualizacji danych

Szczegóªowy opis aplikacji znajduje si¦ w sekcji 5.5. Program uruchamiany jest za pomoc¡ polecenia:

```
python data_visualizer.py
```

A.5. Konguracja i uruchamianie symulacji nauki

Aplikacja symulujica gr¦ na gieªdzie zakªadów sportowych korzysta z kon-

guracji zawartej w pliku Ustawione w niej warto±ci zmiennych

mogi by¢ nadpisywane za pomoc¡ parametrów uruchomieniowych skryptu. Zmienne zwi¡zane ze ¹ródªami danych zostaªy przedstawione w sekcji A.2. W tabeli A.1 znajduje si¦ opis pozostaªych zmiennych oraz atrybutów skryptu.
Symulacja uruchamiana jest za pomoc¡ polecenia:

```
python simulation_launcher.py [opcjonalne argumenty]
```

Po zako«czeniu symulacji jej rezultat zostaje wydrukowany w konsoli sys-temowej oraz traa do pliku

56:8355344197

Tabela A.1: Dost¦pne zmienne konguracyjne oraz atrybuty uruchomieniowe skryptu simulation_launcher.py.

Zmienna Atrybut	Opis
−h, −−help Wy±wietlenie instrukcji u»ytkowania	skryptu.
alpha −a, −−alpha Ustawienie wspóªczynnika uczenia si¦	α ∈ [0, 1].

gamma
−g, −−gamma
Ustawienie wspóªczynnika dyskontowa-

nia nagród γ ∈ [0, ∞].

epsilon
−e, −−epsilon
Ustawienie prawdopodobie«stwa podej-

mowania decyzji losowych (tzw. eksplo-

rujicych) ε ∈ [0, 1].

alg
−−alg
Konguracja wykorzystywanego w

symulacji
algorytmu
nauki. Warto±ć

zmiennej lub atrybutu to ciąg znakowy z

nast¦pujicymi elementami skªadowymi:

Znak 1. - implementacja algorytmu

Q/q (Q-learning) S/s (Sarsa)

Znak 2. - metoda ±ladów aktywno±ci

(opcjonalnie)

W/w (Watkinsa) N/n (naiwna)

Kolejne znaki - warto±¢ liczbowa wspóª-

czynnika ±wie»o±ci (opcjonalnie)

Liczba rzecz. z przedziaªu [0,1]

Przykªadowe warto±ci atrybutu/zmien-

nej:

Q sN0.1 Qw0.42

markets_to_process
−m, −−markets
Liczba rynków gieªdowych do rozegrania

w ramach symulacji. Warto±¢ -1 oznacza

wszystkie dost¦pne rynki.

simulations_to_run
−s, −−sims
Liczba symulacji do przeprowadzenia.

time_steps_to_omit
−o, −−omit
Liczba caªkowita oznaczaj¡ca liczb¦ kro-

ków czasowych do pomijania. Warto±¢

wi¦ksza ni» 1 przyspiesza symulacj¦, ale

powoduje, »e jest mniej precyzyjna.

−−cong
Polecenie
skorzystania
ze
spe-

cjalnego
pliku
konguracyjnego

simulation_cong.json, w którym mo»na w

prosty sposób zdeniowa¢ wiele zesta-

wów parametrów symulacji. Powoduje

pomini¦cie wszystkich innych atrybutów

z wyj¡tkiem −−ds oraz −h/−−help.

57:1127751836

import import

B. Kod ¹ródªowy

W tym dodatku zamieszczono fragment pliku td_learning.py, w którym znajduje si¦ implementacja algorytmów Q-learning oraz Sarsa. Plik zawiera denicj¦ klasy TDAgent reprezentuj¡cej agenta w procesie nauki.

Kluczowym fragmentem skryptu s¡ metody __learn_ql oraz __learn_sarsa, w których wyliczany jest skªadnik , czyli bª¡d TD potrzebny do aktualizacji

macierzy warto±ci Q. Jej aktualizacja odbywa si¦ w metodzie __update_q_matrix na podstawie wyliczonego skªadnika.

Listing B.1: Implementacja algorytmów Q-learning oraz Sarsa zawarta w
module td_learning.

```
# - - c o d i n g : u t f -8 - -
__author__ = ' Mateusz J a n c a r z '

numpy random


c l a s s TDAlgorithm :
"""
Klasa opakowujaca s t a l e z i d e n t y f i k a t o r a m i algorytmow .
"""
Q_LEARNING = 0
SARSA = 1


c l a s s ETracingMethod :
"""
Klasa opakowujaca s t a l e z i d e n t y f i k a t o r a m i metod sladow a k t y w n o s c i .
```

"""
WATKINS = 0
NAIVE = 1

c l a s s ETrace :
"""
Klasa r e p r e z e n t u j a c a typ sladow a k t y w n o s c i .
: param t r a c i n g _ t y p e : i d e n t y f i k a t o r metody
: param f a c t o r : w a r t o s c w s p o l c z y n n i k a s w i e z o s c i
"""

def __init__ (s e l f , t r a c i n g _ t y p e , f a c t o r) :
s e l f . type = t r a c i n g _ t y p e
s e l f . f a c t o r = f a c t o r

c l a s s TDAgent :

54

58:1250649176

"""

Klasa
r e p r e z e n t u j a c a
a g e n t a
u c z a c e g o
s i e metodami
r o z n i c
czasowych .
: param
q : poczatkowa
p o s t a c
m a c i e r z y
w a r t o s c i
Q

: param

l e a r n i n g _ r a t e : wskaznik zapamietywania

: param
chance_to_explore : prawdopodobienstwo
p o d j e c i a a k c j i e k s p l o r u j a c e j
: param
d i s c o u n t _ f a c t o r :
w s p o l c z y n n i k
dyskontowania

: param
e _ t r a c i n g : o b i e k t
r e p r e z e n t u j a c y typ sladow
a k t y w n o s c i
: param
a l g : i d e n t y f i k a t o r algorytmu
k o n k r e t n e g o

: r a i s e
V a l u e E r r o r : wyrzucany
w przypadku bledow w a l i d a c j i
parametrow

"""

def

```
__init__(self, q,
learning_rate=0.1,
chance_to_explore=0.25,
discount_factor=0.8,
e_tracing=None,
alg=TDAlgorithm.Q_LEARNING):

self.Q = TDAgent.__validated_q_matrix(q)

for arg in (learning_rate, chance_to_explore, discount_factor):
if not isinstance(arg, (int, long, float)) and not (0<=arg<=1):
raise ValueError('Learning rate and chance to explore must be a'
'number in [0,1].')

if not isinstance(discount_factor,
(int, long, float)) and discount_factor<0:
raise ValueError('Discount factor must be a positive number.')



if e_tracing is not None and not isinstance(e_tracing, ETrace):
raise ValueError(

'Argument e_tracing should be an instance of ETrace class.')
else:
self.e = numpy.array(self.Q)

self.e[~numpy.isnan(self.e)] = 0.0

self.Q = self.__validated_q_matrix(q)

self.epsilon = chance_to_explore
self.gamma = discount_factor
self.alpha = learning_rate

self.current_state = None
self.current_action = None
self.action_in_next_state = None
self.last_action_reward = None

self.e_tracing = e_tracing
self.alg = alg

@staticmethod
def __validated_q_matrix(array):
"""
Sprawdza wymiary wejsciowej tablicy nastepnie zwraca jako numpy.ndarray.
```

: param a r r a y : t a b l i c a (dwuwymiarowa l i s t a l u b numpy . ndarray)
: r e t u r n : : r a i s e V a l u e E r r o r : wyrzucany w przypadku bledow w a l i d a c j i
"""

55

59:1612088087

def

def

def

```
is_not_list = array is None or
not (


isinstance(array, numpy.ndarray)
or isinstance(array, list))

if is_not_list:

must be a two-dimensional list'

raise ValueError('Initial Q
array


'or numpy.ndarray.')
if not len(array)>0:
raise ValueError('Provided array     is empty.')

q = numpy.array(array)









if
len(q.shape)<2:
```

```
raise ValueError('Provided array has wrong format.')


return
q









set_initial_state(self, state):









"""
```

Ustawia
s t a n
poczatkowy a g e n t a .

: param
s t a t e :
i n d e k s
s t a n u

"""

```
if
not
isinstance(state, (int, long))
or
\
```

```
not
0 <=
state
<
self.Q.shape[0]:
```

```
raise
ValueError('Initial
state identifier
must
be
a positive
'
```

```
'integer
that
is
lower than
Q
```

```
matrix
rows  number
'




' ( which
i s
'
+
s t r ( s e l f .Q. shape [ 0 ] )
+ ' ) . ' )

i f
s e l f . c u r r e n t _ s t a t e
i s  None :









s e l f . c u r r e n t _ s t a t e = s t a t e









__get_possible_actions ( s e l f ,
s t a t e=None ) :
```

"""

Zwraca
l i s t e
mozliwych do
wykonania
a k c j i
d l a
o k r e s l o n e g o
s t a n u .

: param
s t a t e :
i n d e k s
s t a n u

: r e t u r n : l i s t a z i n d e k s a m i a k c j i

"""

state = self.current_state
if
state
is
None else
state

return
(~numpy.isnan(self.Q[state])).nonzero()[0]

__get_most_valuable_action(self,
state=None):

"""

Zwraca
i n d e k s
a k c j i
z
na j wy zs za (l u b
l o s ow a w
przypadku
rownych

w a r t o s c i) w a r t o s c i a
Q
d l a
podanego
l u b
o b e c n i e
przyjmowanego s t a n u .

: param
s t a t e :
i n d e k s
s t a n u

```
:return: indeks akcji

"""

in_state = self.current_state
if state
is
None else
state

available_actions = self.Q[in_state]
```

most_val_actions = numpy . argwhere (
a v a i l a b l e _ a c t i o n s == numpy . nanmax (
a v a i l a b l e _ a c t i o n s))

return random . c h o i c e (most_val_actions) [0] i f len (most_val_actions) > 1 e l s e most_val_actions [0] [0]

def __choose_action_e_greedily (s e l f , s t a t e=None) :
"""

Zwraca i n d e k s
a k c j i
z g o d n e j z
p o l i t y k a
e −z a c h l a n n a wzgledem w a r t o s c i Q
d l a podanego
l u b o b e c n i e
przyjmowanego
s t a n u .
: param s t a t e :
i n d e k s
s t a n u

: r e t u r n : i n d e k s a k c j i

"""

i n _ s t a t e = s t a t e i f
s t a t e
i s
not
None
e l s e s e l f . c u r r e n t _ s t a t e

```
c h o o s e _ b e s t _ a c t i o n =
s e l f . e p s i l o n
< random . random ( )
```

```
def
```

```
def
```

```
def
```

```
return self.__get_most_valuable_action(in_state)\ifchoose_best_action\
else random.choice(self.__get_possible_actions(in_state))

choose_action_to_perform(self):
"""
Zwraca i ustawia indeks akcji optymalnej pod wzgledem wartosci
Qzgodnie ze strategia e-zachlanna.

:return: indeks akcji
"""
self.last_action_reward=None
self.current_action=self.__choose_action_e_greedily()
return self.current_action

provide_reward_for_last_action(self, reward):
"""

Umozliwia przekazanie do obiektu ucznia wzmocnienia otrzymanego po wykonaniu akcji.
:param reward: wartosc wzmocnienia
"""
if not isinstance(reward,
(int, long,
float)):
raise ValueError('Reward must be
a number.')
self.last_action_reward=
reward

__learn_ql(self,
next_state):

"""



Wylicza i zwraca
jeden ze
skladnikow
aktualizacji dla algorytmu

Q-learning.
:param next_state: indeks stanu
:return: wartosc skladniku delta
"""
max_action_value=numpy.nanmax(

[self.Q[next_state, action] for action inself.__get_possible_actions(next_state)])
```

```
delta=self.last_action_reward+ self.gamma   max_action_value − \
self.Q[self.current_state,   self.current_action]

return delta

def __learn_sarsa(self, next_state):
"""
Wylicza i        zwraca jeden ze skladnikow    aktualizacji        dla algorytmu SARSA.
:param next_state:       indeks stanu
:return:     wartosc skladniku       delta
"""
self.action_in_next_state= self.__choose_action_e_greedily(
next_state)
```

57

61:9166568761

```
delta=self.last_action_reward+ self.gamma   \
self.Q[next_state,   self.action_in_next_state] −\
self.Q[self.current_state,   self.current_action]

return delta

def __update_q_matrix(self,        delta):
"""
Aktualizuje macierz Q.
:param delta:       wartosc skladniku       delta
"""
if      self.e_tracing    is not None:

self.e[self.current_state, self.current_action]+= 1.
most_val_action = self.__get_most_valuable_action()


for x in xrange(self
for y in xrange(
```

```
.Q. shape [ 0 ] ) :

s e l f .Q. shape [ 1 ] ) :


s e l f .Q[ x , y ] += d e l t a       s e l f . a l p h a  s e l f . e [ x , y ]

i f       s e l f . c u r r e n t _ a c t i o n == most_val_action :
s e l f . e [ x , y ] = s e l f . gamma        s e l f . e _ t r a c i n g . f a c t o r

e l s e :
i f       s e l f . e _ t r a c i n g . type == ETracingMethod .WATKINS:
s e l f . e [ x , y ] = 0 .

e l s e :
s e l f .Q[ s e l f . c u r r e n t _ s t a t e ,   s e l f . c u r r e n t _ a c t i o n ] \
+= d e l t a      s e l f . a l p h a

def go_to_next_state ( s e l f , n e x t _ s t a t e ) :
"""
Umozliwia z d e f i n i o w a n i e stanu , w ktorym ma s i e       z n a l e z c a g e n t
po wykonaniu a k c j i . Dodatkowo przeprowadza c z y n n o s c i z w i a z a n e
z wartosciowaniem      a k c j i .
: param n e x t _ s t a t e :       i n d e k s s t a n u
: r a i s e         V a l u e E r r o r : wyrzucany w przypadku braku w c z e s n i e j s z e g o
wywolania metody p r o v i d e _ r e w a r d _ f o r _ l a s t _ a c t i o n ( )
"""
i f       s e l f . l a s t _ a c t i o n _ r e w a r d      i s None :
r a i s e V a l u e E r r o r ( ' Reward f o r t h e     l a s t     a c t i o n was not p r o v i d e d . ' )

s e l f . c u r r e n t _ a c t i o n = s e l f . c u r r e n t _ a c t i o n \
i f       s e l f . a c t i o n _ i n _ n e x t _ s t a t e  i s None \
e l s e   s e l f . a c t i o n _ i n _ n e x t _ s t a t e

d e l t a = s e l f . __learn_ql ( n e x t _ s t a t e ) \
i f       s e l f . a l g == TDAlgorithm .Q_LEARNING \
e l s e   s e l f . __learn_sarsa ( n e x t _ s t a t e )

s e l f . __update_q_matrix ( d e l t a )
s e l f . c u r r e n t _ s t a t e = n e x t _ s t a t e
```

62:5085030937

Bibliograa

[1]Science Daily, Big Data, for better or worse: 90% of world's data generated over last two years, 2013, http://www.sciencedaily.com/releases/2013/05/130522085217.htm

[2]Wikipedia, Machine learning, 2015, https://en.wikipedia.org/wiki/Machine_learning .
[3]Tom M. Mitchell, The Discipline of Machine Learning , School of Computer
Science Carnegie Mellon University Pittsburgh, 2006.
[4] Wikipedia, Arthur Samuel, 2015,
https://en.wikipedia.org/wiki/Arthur_Samuel .

[5]Michael L. Littman, Reinforcement learning improves behaviour from evalu-ative feedback, Nature Insight, vol. 521, 2015.
[6]Antoine Cully, Je Clune, Danesh Tarapore, Jean-Baptiste Mouret,
Robots that can adapt like animals , Nature Insight, vol. 521, 2015.
[7]Pawe ª Cichosz,

Systemy ucz¡ce si¦ , Wydawnictwa Naukowo-Techniczne, Warszawa, 2000.
[8]Willi Richert, Luis Pedro Coelho,
Building Machine Learning Systems with Python , Packt Publishing, 2013.
[9]Richard S. Sutton, Andrew G. Barto,
Reinforcement Learning: An Introduction , MIT Press, Cambridge, 1998.

[10]Python Programming Language - Ocial Website, http://www.python.org/ .

[11]Pakiet NumPy, 2015, http://www.numpy.org/ .

[12]Biblioteka Matplotlib, 2015, http://matplotlib.org/ .

[13]Systemy Eksperckie , Zak ªad Elektrotechniki i Informatyki Uniwersytetu Rze-
szowskiego,

http://www.neurosoft.edu.pl/tkwater/tk/se.pdf
[14] Schemat uczenia ze wzmocnieniem - ilustracja,

https://web.stanford.edu/group/pdplab/pdphandbook/handbookch10.

html

[15]FuturistSpeaker, Artıcial Intelligence will be Crashing the Stock Market in 3, 2, 1..., 2015,

http://www.futuristspeaker.com/2014/06/ artificial-intelligence-will-be-crashing-the-stock-market-in-3-2-1/
[16]John Moody, Lizhong Wu, Yuansong Liao, Matthew Saell

Performance functions and reinforcement learning for trading systems and

portfolios, Oregon Graduate Institute of Science and Technology Department of Computer Science and Engineering, Portland, OR, 1997.
[17] Juan M. Alberola, Ana Garcia-Fornes, Agustin Espinosa

Price Prediction in Sports Betting Markets , Departament de Sistemes Infor-màtics i Computació, Universitat Politècnica de València, Camí de Vera s/n. 46022, València, Spain

63:2237029441