­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Лабораторна робота №3

Прийняття рішень в умовах ризику

з курсу «МАТЕМАТИЧНІ МЕТОДИ ДОСЛІДЖЕННЯ ОПЕРАЦІЙ»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 14

Виконав студент гр. КНз-2

Чалий Михайло

­­

Львів 2014

## Мета роботи

Ознайомитись з методами прийняття рішень в умовах, коли вибір деякої стратегії пов’язаний з певним набором станів середовища з визначеною ймовірністю.

## Короткі теоретичні відомості

Задача прийняття рішень в умовах ризику виникає в тих випадках, коли з кожною стратегією *xi*, що обирається, пов’язана ціла множина різноманітних результатів *Oj* з відомими ймовірностями . В цьому випадку, модель задачі може бути представлена у вигляді такої матриці:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | O1 | O2 | … | Om | | х1 | *u(x1,O1)* | *u(x1,O2)* | *…* | *u(x1,Om)* | | х2 | *u(x2,O1)* | *u(x2,O2)* | *…* | *u(x2,Om)* | | … | *…* | *…* | *…* | *…* | | хn | *u(xn,O1)* | *u(xn,O2)* | *…* | *u(xn,Om)* | |  |

Тут *u(xi,Oj)* – корисність результату *Oj* при використанні стратегії *xi*.

Якщо відомі ймовірності , то можливо вивести очікувану корисність для кожної стратегії:

|  |  |
| --- | --- |
|  |  |

Вочевидь, в якості оптимальної стратегії слід обирати ту, для якої очікувана корисність максимальна. У разі, коли елементи матриці (3.1) є від’ємними числами (тобто виступають у ролі витрат), в якості найкращої стратегії обирається стратегія, для якої очікувані витрати мінімальні.

Даний критерій оптимальності носить назву «критерій Байеса-Лапласа» і застосовується при відомих значеннях ймовірностей, або є підстави вважати, що ймовірності виникнення кожного з результатів приблизно однакові.

## Завдання

4.1 Обрати варіант та дані температури («Додаток Б»).

4.2 Визначити найкращу стратегію при поверненні протягом одного з 12-ти місяців за умови, що ймовірність повернення в кожен з місяців однакова.

4.3 Визначити найкращу стратегію за при поверненні протягом одного сезону за наданих наборів ймовірностей повернення у кожен з місяців (табл. 3.6).

Таблиця 3.6 – Набори значень ймовірностей

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №№ | Назва набору | Місяць, Pi | | | | | | | | | | | | |
| 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 |
| 1 | Зима | 1/3 | 1/3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/3 |
| 2 | Весна | 0 | 0 | 1/3 | 1/3 | 1/3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | Літо | 0 | 0 | 0 | 0 | 0 | 1/3 | 1/3 | 1/3 | 0 | 0 | 0 | 0 |
| 4 | Осінь | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1/3 | 1/3 | 1/3 | 0 |

4.4 Визначити найкращу стратегію при поверненні протягом одного з 12-ти місяців за умови, що ймовірність повернення взимку втричі більша за інші місяці.

4.5 Визначити найкращу стратегію при поверненні протягом одного з 12-ти місяців за умови, що ймовірність повернення залежить від кількості днів у місяці (рік вважати не високосним).

4.6 Визначити найкращу стратегію при поверненні протягом одного з 12-ти місяців за умови, що ймовірність повернення в кожен з місяців однакова, а початкова вартість речей з номерами № 2, 4, 9, 15, 16 (табл. 3.1) зменшилася втричі.

Таблиця Б.1 – Середньомісячна температура міст світу

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Місто | І | ІІ | ІІІ | ІV | V | VI | VII | VIII | IX | X | XI | XII |
| 14 | Урумчі (Китай) | -13 | -11 | -1 | +11 | +18 | +23 | +25 | +24 | +18 | +9 | -2 | -10 |

## Рішення

Лістінг l2\_3.py

# coding=utf8

import operator

from utils import \*

from collections import namedtuple

class ClothesSet(object):

def \_\_init\_\_(self, name, temp\_min, temp\_max, clothes, weight):

self.name = name

self.temp\_min = temp\_min

self.temp\_max = temp\_max

self.clothes = clothes

self.weight = weight

def get\_price(self):

return self.weight \* 10;

def \_\_repr\_\_(self):

return self.name + ": " + str(self.temp\_min) + '..' + str(self.temp\_max) + ': ' + ', '.join(self.clothes)

class ClothesItem(object):

def \_\_init\_\_(self, name, weight, price):

self.name = name

self.weight = weight

self.price = price

class MonthCalculation(object):

def \_\_init\_\_(self, tag, clothes\_set, clothes\_diff, added\_price, full\_price):

self.tag = tag

self.clothes\_set = clothes\_set

self.clothes\_diff = clothes\_diff

self.added\_price = added\_price

self.full\_price = full\_price

def \_\_repr\_\_(self):

return "(Diff: " + str(self.clothes\_diff) + "; Added price: " + str(self.added\_price) + ")"

class ClothesSolver(object):

def \_\_init\_\_(self, all\_clothes\_sets, all\_clothes, temps):

self.all\_clothes\_sets = all\_clothes\_sets

self.all\_clothes = all\_clothes

self.temps = temps

def \_calculate\_clothes\_diff(self, clothes\_set, temp):

applicable\_set = next((x for x in self.all\_clothes\_sets if x.temp\_min < temp <= x.temp\_max), None)

return [ x for x in applicable\_set.clothes if not x in clothes\_set.clothes ]

def \_calculate\_added\_price\_of\_clothes(self, clothes):

def get\_price\_for\_item(clothes\_item):

return next(x for x in self.all\_clothes if x.name == clothes\_item).price

return sum(get\_price\_for\_item(x) + 2.0 for x in clothes)

def calculate\_options(self, monthes):

def calulate\_monthes(clothes\_set):

def calulate\_month(m):

temp = self.temps[m]

clothes\_diff = self.\_calculate\_clothes\_diff(clothes\_set, temp)

added\_price = self.\_calculate\_added\_price\_of\_clothes(clothes\_diff)

full\_price = clothes\_set.get\_price() + added\_price

return MonthCalculation(m, clothes\_set, clothes\_diff, added\_price, full\_price)

return [calulate\_month(m) for (m,\_) in monthes]

return [ (cs, calulate\_monthes(cs)) for cs in self.all\_clothes\_sets ]

def find\_best\_strategy(self, monthes):

probabilities = [p for (\_, p) in monthes]

def expected\_favor(calculations):

return sum(-calculation.full\_price \* probabilities[calculation\_index]

for calculation\_index, calculation in enumerate(calculations))

options = self.calculate\_options(monthes)

expected\_favors = [(clothes\_set, expected\_favor(calculations))

for (clothes\_set, calculations) in options]

return sorted(expected\_favors, key=operator.itemgetter(1), reverse=True)

## Результат

Див l2\_3.html

## Висновки

Ознайомився з методами прийняття рішень в умовах, коли вибір деякої стратегії пов’язаний з певним набором станів середовища з визначеною ймовірністю.