1. **Цель работы**

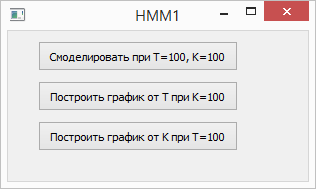
Моделирование наблюдаемых и скрытых последовательностей.

1. **Задание**
2. Смоделировать последовательность скрытых состояний длиной *T=100 (см [3], [4] стр 160-163)*
3. Смоделировать последовательность наблюдаемых состояний длиной *T=100 по* последовательности скрытых состояний *(см [3], [4] стр 160-163)*
4. Представить полученную наблюдаемую последовательность в графическом виде.
5. Привести графики достигнутой точности по параметрам (ρ\_A , ρ\_B) в зависимости от *Т* при *К*=100 и от *К* при *Т*=100.
6. Подготовить по 2 набора (каждый набор включает по К=100) наблюдаемых последовательностей.
7. **Вариант задания**

|  |  |  |  |
| --- | --- | --- | --- |
| *Вариант* | *Алфавит: V* | *Матрица переходных вероятностей А* | *Матрица эмиссей*  *В* |
| 2 |  |  |  |

1. **Описание разработанного программного средства**

Программа разработана средствами Python + Qt5. Предназначена для моделирования скрытых и наблюдаемых последовательностей.



Интерфейс программного средства позволяет пользователю выбирать режим моделирования: Смоделировать при T=100, K=100 (с последующей записью последовательности в файл и представлением полученной последовательности в графическом виде), Построить график от T при K=100 (моделирование производится при различных длинах последовательности Т), Построить график от K при T=100 (моделирование производится при различных К – количество проведения эксперимента). Результатом моделирования последовательность скрытых и наблюдаемых состояний.

1. **Текст программы**

import sys

from PyQt5.QtCore import QObject,Qt

from PyQt5.QtWidgets import QWidget, QLabel, QApplication, QCheckBox, QComboBox, QPushButton, QLineEdit

import matplotlib

import numpy as np

import scipy as sp

matplotlib.use('Qt5Agg')

import matplotlib.pyplot as plt

from functools import reduce

def get\_data(fname):

A = np.array([[i for i in line.split()] for line in open(fname, encoding="utf-8")], dtype = np.double)

return A

def WritingInFile(names, sequences, fileName):

with open(fileName, "w") as file:

for line in sequences:

print(line, file=file)

#моделирование последовательности скрытых состояний

def model\_hiden\_sequence(A, T):

s = 0

Q = [0]

i = 0

for j in range(T - 1):

alpha = np.random.uniform(0,1)

if 0. <= alpha <= A[i][0]:

i = 0

elif A[i][0] <= alpha <= A[i][0] + A[i][1]:

i = 1

elif A[i][0] + A[i][1] <= alpha <= 1:

i = 2

Q.append(i)

return Q

#моделирование последовательности наблюдаемых состояний

#по последовательности скрытых состояний

def model\_observable\_sequence(Q, B, T):

a = 0

O = []

for j in range(T):

i = Q[j]

alpha = np.random.uniform(0,1)

if 0. <= alpha <= B[i][0]:

a = 0

elif B[i][0] <= alpha <= 1:

a = 1

O.append(a)

return O

#подсчет пар для оценки переходной матрицы

def cnt\_pair\_matr\_A(Q, t\_1, t):

cnt = 0

for i in range(1, len(Q)):

if Q[i-1] == t\_1 and Q[i] == t:

cnt += 1

return cnt

#оценка матрицы А

def est\_matr\_A(Q):

est\_A = np.eye(3)

tmp = np.array([[cnt\_pair\_matr\_A(Q, i, j) for j in range(len(est\_A))]

for i in range(len(est\_A))])

tmp1 = np.sum(tmp, axis = 1)

tmp1[tmp1 == 0] = 1

est\_A = list(map(lambda x, y: x / y, tmp, tmp1))

return est\_A

#подсчет пар для оценки матрицы эмиссей

def cnt\_pair\_matr\_B(Q, O, s, a):

cnt = 0

for i in range(len(O)):

if Q[i] == s and O[i] == a:

cnt += 1

return cnt

#оценка матрицы B

def est\_matr\_B(Q, O):

est\_B = np.zeros((3,2))

tmp = np.array([[cnt\_pair\_matr\_B(Q, O, i, j) for j in range(est\_B.shape[1])]

for i in range(est\_B.shape[0])])

tmp1 = np.sum(tmp, axis = 1)

tmp1[tmp1 == 0] = 1

est\_B = list(map(lambda x, y: x / y, tmp, tmp1))

return est\_B

#метод Монте-Карло

def Monte\_Karlo\_method0(T, K):

A = get\_data('A.txt')

B = get\_data('B.txt')

ro\_A = []

ro\_B = []

#моделируем

Q = np.array([model\_hiden\_sequence(A, T) for i in range(K)])

O = np.array(list(map(lambda x: model\_observable\_sequence(x, B, T), Q)))

#оцениваем

est\_A = np.array(list(map(lambda x: est\_matr\_A(x), Q)))

est\_B = np.array(list(map(lambda x, y: est\_matr\_B(x, y), Q, O)))

#усредненная оценка матриц

est\_A\_mean = np.array(list(reduce((lambda a, x: a + x), est\_A))) / K

est\_B\_mean = np.array(list(reduce((lambda a, x: a + x), est\_B))) / K

#норма разности

ro\_A.append(np.linalg.norm(A - est\_A\_mean))

ro\_B.append(np.linalg.norm(B - est\_B\_mean))

WritingInFile(['Q'], Q, 'Q.txt')

WritingInFile(['O'], O, 'O.txt')

return np.array(ro\_A), np.array(ro\_B), Q, O

def Monte\_Karlo\_method(T):

A = get\_data('A.txt')

B = get\_data('B.txt')

ro\_A = []

ro\_B = []

for K in range(1, 101):

#моделируем

Q = np.array([model\_hiden\_sequence(A, T) for i in range(K)])

O = np.array(list(map(lambda x: model\_observable\_sequence(x, B, T), Q)))

#оцениваем

est\_A = np.array(list(map(lambda x: est\_matr\_A(x), Q)))

est\_B = np.array(list(map(lambda x, y: est\_matr\_B(x, y), Q, O)))

#усредненная оценка матриц

est\_A\_mean = np.array(list(reduce((lambda a, x: a + x), est\_A))) / K

est\_B\_mean = np.array(list(reduce((lambda a, x: a + x), est\_B))) / K

#норма разности

ro\_A.append(np.linalg.norm(A - est\_A\_mean))

ro\_B.append(np.linalg.norm(B - est\_B\_mean))

return np.array(ro\_A), np.array(ro\_B), Q, O

def Monte\_Karlo\_method1(K):

A = get\_data('A.txt')

B = get\_data('B.txt')

ro\_A = []

ro\_B = []

for T in range(1, 101):

#моделируем

Q = np.array([model\_hiden\_sequence(A, T) for i in range(K)])

O = np.array(list(map(lambda x: model\_observable\_sequence(x, B, T), Q)))

#оцениваем

est\_A = np.array(list(map(lambda x: est\_matr\_A(x), Q)))

est\_B = np.array(list(map(lambda x, y: est\_matr\_B(x, y), Q, O)))

#усредненная оценка матриц

est\_A\_mean = np.array(list(reduce((lambda a, x: a + x), est\_A))) / K

est\_B\_mean = np.array(list(reduce((lambda a, x: a + x), est\_B))) / K

#норма разности

ro\_A.append(np.linalg.norm(A - est\_A\_mean))

ro\_B.append(np.linalg.norm(B - est\_B\_mean))

return np.array(ro\_A), np.array(ro\_B), Q, O

def plot\_graph(roA, roB):

plt.xlabel('K')

plt.ylabel('ro')

plt.xlim(0,101)

k = np.linspace(1, 101, 100)

f = plt.plot(k, roA)

f1 = plt.plot(k, roB, 'r-')

plt.show()

def plot\_graph1(roA, roB):

plt.xlabel('T')

plt.ylabel('ro')

plt.xlim(0,101)

k = np.linspace(1, 101, 100)

f = plt.plot(k, roA)

f1 = plt.plot(k, roB, 'r-')

plt.show()

def plot\_hist(ydata):

xdata = np.linspace(1, 101, 100)

plt.xlim(0, 100)

plt.bar(xdata, ydata[0])

plt.show()

class Interface(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.Tmax = 100 #длина последовательности

self.Kmax = 100 #кол-во моделирования в методе Монте-Карло

self.ro\_A = 0

self.ro\_B = 0

self.Q = 0

self.O = 0

self.initUI()

def initUI(self):

btn1 = QPushButton('Смоделировать при T=100, K=100', self)

btn1.move(30, 10)

btn1.resize(200,30)

btn1.clicked.connect(self.buttonClicked)

btn2 = QPushButton('Построить график от T при K=100', self)

btn2.move(30, 50)

btn2.resize(200,30)

btn2.clicked.connect(self.buttonClicked1)

btn3 = QPushButton('Построить график от K при T=100', self)

btn3.move(30, 90)

btn3.resize(200,30)

btn3.clicked.connect(self.buttonClicked2)

self.setGeometry(270, 270, 300, 150)

self.setWindowTitle('HMM1')

self.show()

def buttonClicked(self):

self.ro\_A, self.ro\_B, self.Q, self.O = Monte\_Karlo\_method0(self.Tmax, self.Kmax)

plot\_hist(self.Q)

plot\_hist(self.O)

def buttonClicked1(self):

self.ro\_A, self.ro\_B, self.Q, self.O = Monte\_Karlo\_method1(self.Kmax)

plot\_graph1(self.ro\_A, self.ro\_B)

def buttonClicked2(self):

self.ro\_A, self.ro\_B, self.Q, self.O = Monte\_Karlo\_method(self.Tmax)

plot\_graph(self.ro\_A, self.ro\_B)

if \_\_name\_\_ == '\_\_main\_\_':

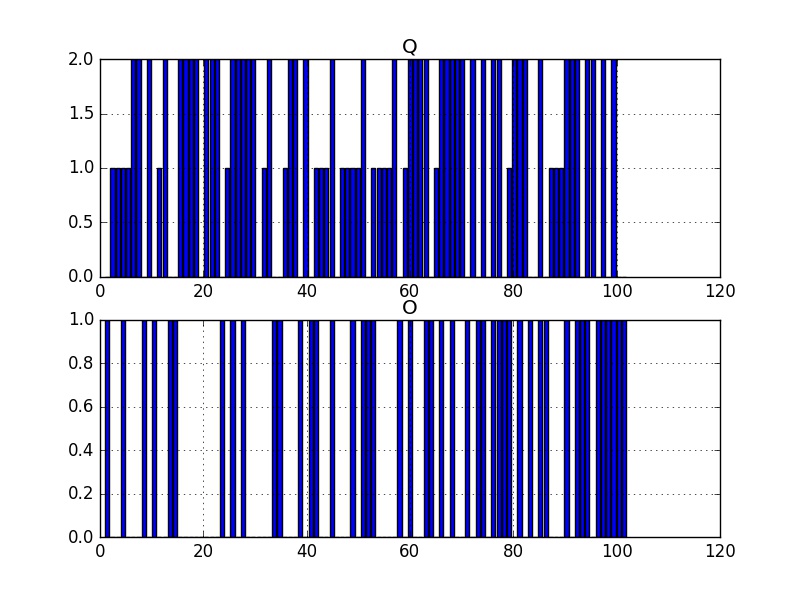
app = QApplication(sys.argv)

ex = Interface()

sys.exit(app.exec\_())

1. **Исследования**

Моделирование последовательностей при T=100, K=100



**Q.txt**

[0 1 1 1 1 2 2 0 2 0 1 2 0 0 2 2 2 2 0 2 2 2 0 1 2 2 2 2 2 0 1 2 0 0 1 2 2 0 2 0 1 1 1 2 0 1 1 1 1 2 0 1 1 1 1 2 0 1 2 2 2 2 0 1 2 2 2 2 2 0 2 0 2 0 2 2 0 1 2 2 2 0 0 2 0 1 1 1 2 2 2 0 2 2 0 2 0 2 0 0]

[0 2 0 2 2 0 0 1 1 2 0 2 0 2 0 0 2 0 1 2 2 2 0 2 2 0 1 1 2 0 1 1 1 1 1 1 2 2 0 0 2 2 2 2 2 2 2 2 2 0 1 1 2 2 2 2 0 0 1 1 2 2 2 0 2 0 1 1 2 0 2 0 0 1 1 1 1 1 1 2 0 0 1 2 0 0 0 1 2 0 1 1 1 2 0 1 1 2 0 2]

[0 1 1 1 2 0 1 1 1 1 1 2 0 0 2 0 2 0 0 0 2 2 0 2 0 0 1 1 1 1 2 2 2 0 2 0 0 0 1 1 2 0 1 2 2 2 2 0 2 0 2 0 2 2 2 2 2 0 2 0 1 2 2 2 2 2 0 1 1 2 2 0 2 0 2 0 1 1 2 2 2 2 0 2 0 2 2 2 0 0 2 2 0 0 1 1 1 2 0 0]

[0 2 2 2 0 0 0 2 2 2 2 0 1 2 2 0 2 0 2 2 0 0 2 2 2 0 1 1 1 1 2 2 2 2 0 1 2 2 2 2 2 0 2 2 0 2 0 1 2 2 0 1 1 1 2 2 0 1 1 2 0 2 0 1 2 0 1 2 0 0 0 0 1 2 2 2 2 2 2 0 0 2 0 2 2 0 2 0 0 0 2 2 0 1 1 1 2 0 1 1] …

**O.txt**

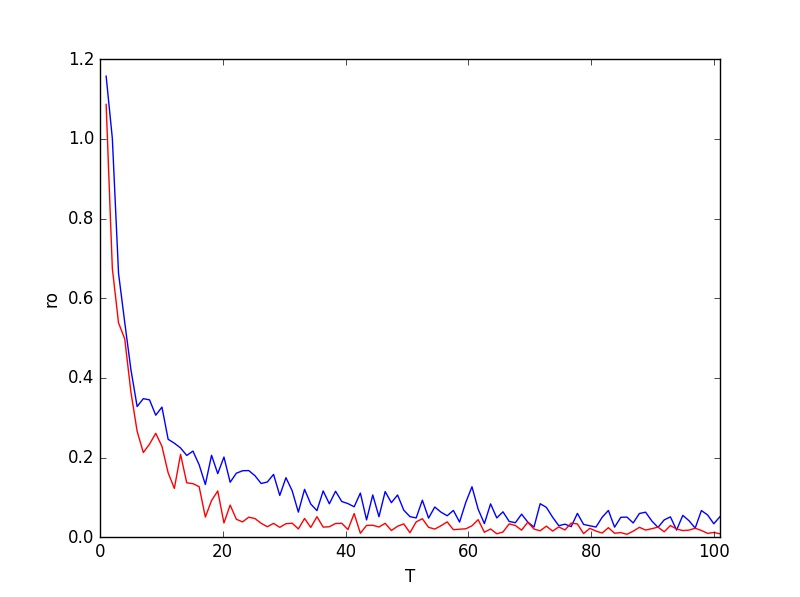
[1 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 1]

[1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 0 0 0 1 1 0 1 1 1 1]

[1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1]

[1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 0 1 1 0 1 0 0 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0] ….

Построить график от T при K=100



Построить график от K при T=100

