1. Задание

1. Найти оценки параметров модели по данным, смоделированным в л/р3 (алгорим Баум-Велша, [4] стр 145-156)
2. Отобразить результаты оценивания в таблице:

Начальное приближение параметров модели	Оценки параметров модели	Достигнутая точность по параметрам (ρ_A, ρ_B)	Кол-во итераций $(iter)$	Достигнутая точность по значению невязки функции правдоподобия: $\lvert lnL(O \vee \lambda)^{iter} - lnL(O \vee \lambda)^{iter-1} \rvert$

Начальное приближение параметров модели выбирать как минимум три раза:

1) Близкими к истинным параметрам
2) Равными истинным параметрам
3) Далекими от истинных параметров

2. Вариант задания

Вариант	*Алфавит: V*	*Матрица переходных вероятностей A*	*Матрица эмиссей B*
2	{0,1}	$\begin{pmatrix} 0,3 & 0,3 & 0,4 \\ 0 & 0,5 & 0,5 \\ 0,5 & 0 & 0,5 \end{pmatrix}$	$\begin{pmatrix} 0,2 & 0,8 \\ 0,8 & 0,2 \\ 0,5 & 0,5 \end{pmatrix}$

3. Текст программы

```
import numpy as np
import scipy as sp
from functools import reduce
import matplotlib.pyplot as plt

def get_data(fname, type):
    O = np.array([[i for i in line.split()] for line in open(fname, encoding="utf-8")],
dtype=type)
    return O

def get_data1(fname, type):
    O = np.array([i for i in open(fname, encoding="utf-8").readline().split()],
dtype=type)
    return O

def WritingInFile(names, sequences, fileName):
    with open(fileName, "w") as file:
        for line in sequences:
            print(line, file=file)

#прямой ход
def forward_path(O, pi, A, B, T, N, K):
    alpha_k = []
    for k in range(K):
        alpha = np.zeros((T, N))
        alpha[0, :] = pi * B[:, O[k, 0]]
        for t in range(1, T):
            for j in range(N):
                tmp = np.zeros(N)
                for i in range(N):
                    tmp[i] = alpha[t - 1, i] * A[i, j]
                alpha[t, j] = tmp.sum() * B[j, O[k, t]]
        alpha_k.append(alpha)
    return np.array(alpha_k)



#обратный ход
def backward_path(O, pi, A, B, T, N, K):
```

```
 beta_k = []
 for k in range(K):
     beta = np.zeros((T, N))
        beta[T - 1, :] = 1
        for t in range(T - 2, -1, -1):
            for i in range(N):
                tmp = np.zeros(N)
                for j in range(N):
                    tmp[j] = beta[t + 1, j] * A[i, j] * B[j, O[k, t + 1]]
                beta[t, i] = tmp.sum()
        beta_k.append(beta)
    return np.array(beta_k)

#вычисление гамма
def calculate_gamma(alpha, beta, T, N, K):
    gamma_k = []
    for k in range(K):
        gamma = np.zeros((T, N))
        for t in range(T):
            for i in range(N):
                gamma[t, i] = alpha[k, t, i] * beta[k, t, i]
            sum_all = gamma[t, :].sum()
            gamma[t, :] = gamma[t, :] / sum_all
        gamma_k.append(gamma)
    return np.array(gamma_k)

#вычисление кси
def calculate_ksi(O, alpha, beta, A, B, T, N, K):
    ksi_k = []
    for k in range(K):
        ksi = np.zeros((T, N, N))
        for t in range(T - 1):
            for i in range(N):
                for j in range(N):
                    ksi[t, i, j] = alpha[k, t, i] * A[i, j] * beta[k, t + 1, j] * B[j,
O[k, t + 1]]
            sum_all = ksi[t, :, :].sum()
            ksi[t, :, :] = ksi[t, :, :] / sum_all
        ksi_k.append(ksi)
    return np.array(ksi_k)

#EM-алгоритм (вычисление оценок параметров модели)
def estimate_parameter(O, pi_0, A_0, B_0, T, N, M, K):
    alp = forward_path(O, pi_0, A_0, B_0, T, N, K)
    bet = backward_path(O, pi_0, A_0, B_0, T, N, K)
    gam = calculate_gamma(alp, bet, T, N, K)
    ksi = calculate_ksi(O, alp, bet, A_0, B_0, T, N, K)
#оценка начальных состояний
    est_pi = np.sum(gam[:, 0, :], axis=0) / K
#оценка переходной матрицы
    est_A_k = np.zeros((K, N, N))
    for k in range(K):
        for i in range(N):
            denom = gam[k, :-1, i].sum()
            for j in range(N):
                est_A_k[k, i, j] = ksi[k, :-1, i, j].sum() / denom

    est_A = np.sum(est_A_k, axis=0) / K
#оценка матрицы эмиссей
    est_B_k = np.zeros((K, N, M))
    for k in range(K):
        for i in range(N):
            denom = gam[k, :, i].sum()
            for j in range(M):
                numer = gam[k, :, i][O[k] == j].sum()
                est_B_k[k, i, j] = numer / denom
    est_B = np.sum(est_B_k, axis=0) / K
    return est_pi, est_A, est_B

#вычисление функции правдоподобия
```

```
def log_likelihood(O, pi, A, B, T, N, K):
    alp = forward_path(O, pi, A, B, T, N, K)
    L = []
    for k in range(K):
        l = np.zeros((T, N))
        for t in range(T):
            for i in range(N):
                l[t, i] = alp[k, t, i]
            sum_all = l[t, :].sum()
        L.append(sum_all)
    lnL = np.sum(np.log(L))
    return lnL

#условие выхода
def iter_exit(O, pi_old, A_old, B_old, pi_new, A_new, B_new, T, N, K):
    old = log_likelihood(O, pi_old, A_old, B_old, T, N, K)
    new = log_likelihood(O, pi_new, A_new, B_new, T, N, K)
    exit = abs(old - new)
    if exit > 1e-3:
        return False, exit
    else:
        return True, exit

#итерационный алгоритм Баума-Уэлша
def baum_welch(O, pi, A, B, T, N, M, K):
    iter = 0
    exit = False
    max_iter = 100
    ex = []
    temp = []
    temp.append(log_likelihood(O, pi, A, B, T, N, K))
    while exit == False:
        iter += 1
        new_pi, new_A, new_B = estimate_parameter(O, pi, A, B, T, N, M, K)
        exit, tmp = iter_exit(O, pi, A, B, new_pi, new_A, new_B, T, N, K)
        temp.append(log_likelihood(O, new_pi, new_A, new_B, T, N, K))
        if iter > max_iter:
            exit = True
        ex.append(tmp)
        pi, A, B = new_pi, new_A, new_B
    plt.xlabel('iter')
    plt.ylabel('lnl')
    it = np.linspace(0, iter, iter)
    f = plt.plot(it, np.array(ex1))
    plt.show()
    WritingInFile(['iter'], np.array([iter]), 'iter.txt')
    WritingInFile(['ex'], ex, 'ex.txt')
    return pi, A, B, ex

def ro_lambd():
    Q = get_data('Q.txt', np.int)
    O = get_data('O.txt', np.int)
    A_0 = []
    B_0 = []
    pi_0 = []
    A_0.append(get_data('A1.txt', np.double))
    B_0.append(get_data('B1.txt', np.double))
    pi_0.append(get_data1('pi1.txt', np.int))
    A_0.append(get_data('A2.txt', np.double))
    B_0.append(get_data('B2.txt', np.double))
    pi_0.append(get_data1('pi2.txt', np.double))
    A_0.append(get_data('A3.txt', np.double))
    B_0.append(get_data('B3.txt', np.double))
```

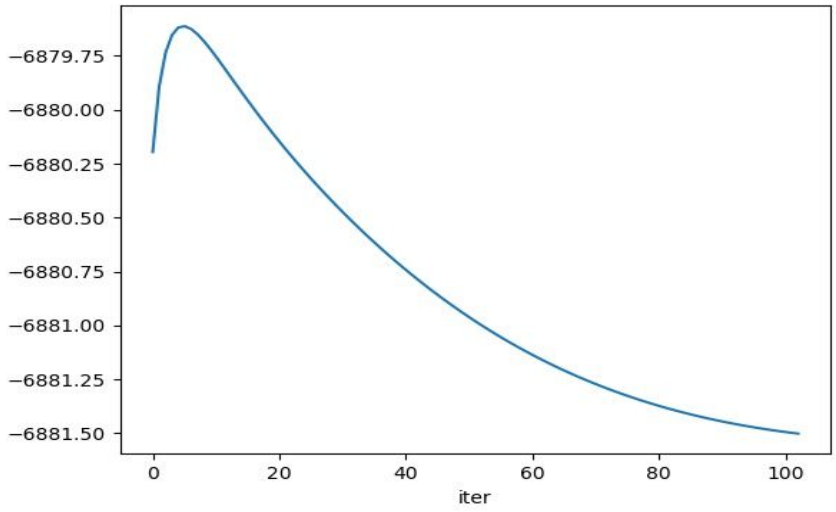
```
    pi_0.append(get_data1('pi3.txt', np.double))
    A_0.append(get_data('A4.txt', np.double))
    B_0.append(get_data('B4.txt', np.double))
    pi_0.append(get_data1('pi4.txt', np.double))
    A_0.append(get_data('A5.txt', np.double))
    B_0.append(get_data('B5.txt', np.double))
    pi_0.append(get_data1('pi5.txt', np.double))
    est = []
    lnL = []
#прогон алгоритма из разных приближений и выбор оценок параметров по максимальному
#логарифму функции правдоподобия
    for i in range(5):
        est_pi, est_A, est_B, ex = baum_welch(O, np.array(pi_0[i]), np.array(A_0[i]),
np.array(B_0[i]), 100, 3, 2, 100)
        est.append([est_pi, est_A, est_B])
        lnL.append(log_likelihood(O, est_pi, est_A, est_B, 100, 3, 100))
    maxlnL = lnL.index(np.max(np.array(lnL)))
    est_param = est[maxlnL]
    WritingInFile(['maxlnL'], np.array([np.max(np.array(lnL))]), 'maxlnL.txt')
    WritingInFile(['est_pi'], est_param[0], 'est_pi.txt')
    WritingInFile(['est_A'], est_param[1], 'est_A.txt')
    WritingInFile(['est_B'], est_param[2], 'est_B.txt')
    ro_A = np.linalg.norm(A_0[maxlnL] -  est_param[1])
    ro_B = np.linalg.norm(B_0 [maxlnL] - est_param[2])
    WritingInFile(['roA'], np.array([ro_A]), 'roA.txt')
    WritingInFile(['roB'], np.array([ro_B]), 'roB.txt')
    return est_param

ro_lambd()
```

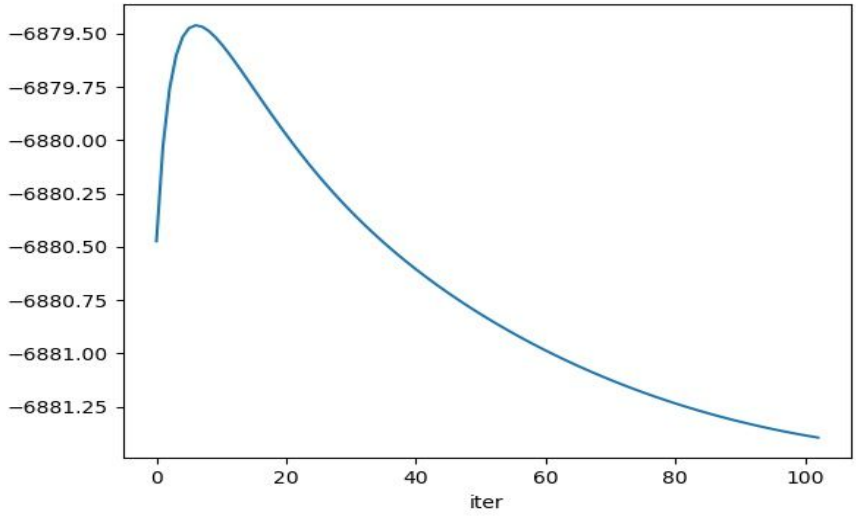
4. Исследования

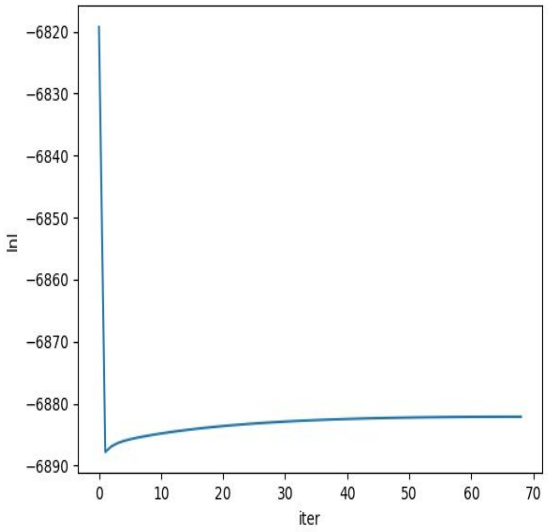
Для различных начальных приближений

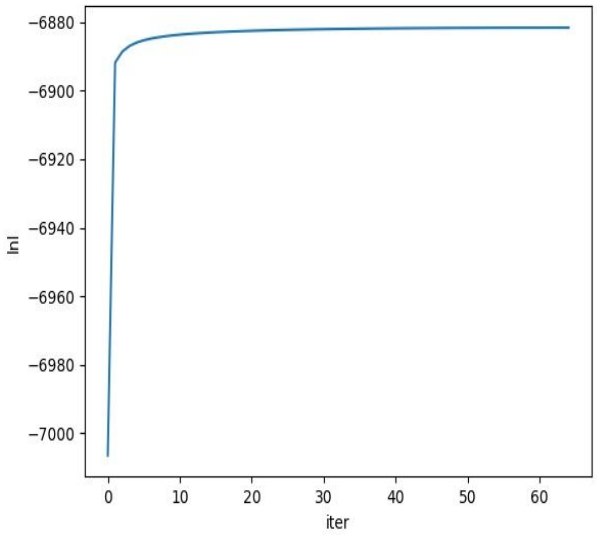
№	Начальное приближение параметров модели	Оценки параметров модели	Достигнутая точность по параметрам (ρ_A , ρ_B)	Кол-во итераций ($iter$)	Достигнутая точность по значению невязки функции правдоподобия: $\lvert lnL(O \vee \lambda)^{iter} - lnL(O \vee \lambda)^{iter-1} \rvert$
1	$\pi = (1\ 0\ 0)$ $A = \begin{matrix} 0{,}3 & 0{,}3 & 0{,}4 \\ 0 & 0{,}5 & 0{,}5 \\ 0{,}5 & 0 & 0{,}5 \end{matrix}$ $B = \begin{matrix} 0{,}2 & 0{,}8 \\ 0{,}8 & 0{,}2 \\ 0{,}5 & 0{,}5 \end{matrix}$	$\pi = (1\ 0\ 0)$ $A = \begin{matrix} 0{,}3223236 & 0{,}3623742 & 0{,} \\ 0 & 0{,}4852506 & 0{,} \\ 0{,}4864534 & 0 & 0{,} \end{matrix}$ $B = \begin{matrix} 0{,}27866 & 0{,}7226301 \\ 0{,}6391175 & 0{,}3628825 \\ 0{,}6007258 & 0{,}4992742 \end{matrix}$	0.123271 0.252465	101	0.0036643
2	$\pi = (0{,}925\ 0{,}025\ 0{,}05)$	$\pi = (0{,}928518349\ 0{,}001148165$	0.125817 0.261109	101	0.005179

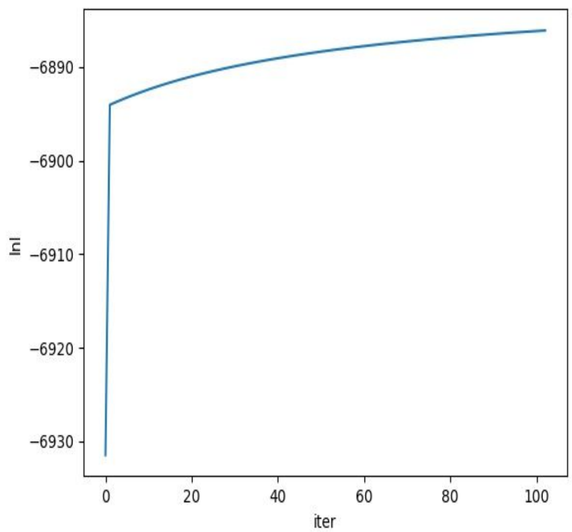
	$A=\begin{pmatrix} 0.365 & 0.3 & \\ 0.005 & 0.5 & \\ 0.435 & 0.005 & \end{pmatrix}$ $B=\begin{pmatrix} 0.135 & 0.865 \\ 0.865 & 0.135 \\ 0.435 & 0.565 \end{pmatrix}$	$A=\begin{pmatrix} 0.3253698 & 0.3627656 & \\ 0.0042136 & 0.4547175 & \\ 0.4784825 & 0.0075977 & \end{pmatrix}$ $B=\begin{pmatrix} 0.2741632 & 0.8 \\ 0.6383054 & 0.3616946 \\ 0.5012929 & 0.4987071 \end{pmatrix}$			
3	$\pi=(0.5,0.5)$ $A=\begin{pmatrix} 0.6 & 0.1 & 0.3 \\ 0.2 & 0.2 & 0.6 \\ 0.7 & 0.1 & 0.2 \end{pmatrix}$ $B=\begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \\ 0.3 & 0.8 \end{pmatrix}$	$\pi=(0.154199086002,0.845800991$ $A=\begin{pmatrix} 0.5614322 & 0.1398389 & \\ 0.2882221 & 0.1677524 & \\ 0.5792607 & 0.1647248 & \end{pmatrix}$ $B=\begin{pmatrix} 0.6028826 & 0.3971174 \\ 0.1821952 & 0.8178048 \\ 0.3676101 & 0.6323899 \end{pmatrix}$	0.194036283978 0.3331186628713	67	0.0009105


−6879.75
−6880.00
−6880.25
−6880.50
−6880.75
−6881.00
−6881.25
−6881.50
0
20
40
60
80
100
iter


−6879.50
−6879.75
−6880.00
−6880.25
−6880.50
−6880.75
−6881.00
−6881.25
0
20
40
60
80
100
iter


−6820
−6830
−6840
−6850
−6860
−6870
−6880
−6890
lnl
0
10
20
30
40
50
60
70
iter


−6880
−6900
−6920
−6940
−6960
−6980
−7000
lnl
0
10
20
30
40
50
60
iter



Для различных начальных приближений (в качестве оценки параметров модели выбираются такие оценки, при которых логарифмическая функция правдоподобия максимальна)

№	Начальное приближение параметров модели	Оценки параметров модели	Достигнутая точность по параметрам (ρ_A , ρ_B)	Кол-во итераций $(iter)$	Достигнутая точность по значению невязки функции правдоподобия: $\lvert lnL(O \vee \lambda)^{iter} - lnL(O \vee \lambda)^{iter-1} \rvert$
1	$\pi=(1,0,0)$ $A=\begin{pmatrix} 0.3 & 0.3 & 0.4 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix}$ $B=\begin{pmatrix} 0.2 & 0.8 \\ 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix}$	$\pi=(0.998851831448, 0.00114816…$ $A=\begin{pmatrix} 0.325698 & 0.3627966 & … \\ 0.0042136 & 0.4547175 & … \\ 0.4784925 & 0.0075977 & … \end{pmatrix}$	0.125817087958 0.261109540218	101	0.0293165844805
2	$\pi=(0.95,0.05,0)$ $A=\begin{pmatrix} 0.35 & 0.3 & … \\ 0.05 & 0.5 & … \\ 0.45 & 0.05 & \end{pmatrix}$				

	0.125 0.875 B= 0.875 0.125 0.425 0.575	0.274632 0.725368 B= 0.633054 0.366946 0.601229 0.398771			
3	π=(0.5,0.5) 0.6 0.1 0.3 A= 0.2 0.2 0.6 0.7 0.1 0.2 0.8 0.2 B= 0.2 0.8 0.3 0.8				
4	π=(0.3,0.4,0.3) 0.4 0.4 0.2 A= 0.1 0.4 0.5 0.6 0.1 0.3 0.3 0.7 B= 0.9 0.1 0.45 0.55				
5	π=(0,0,1) 0.25 0.25 0. A= 0.35 0.5 0. 0.45 0.05 0. 0.5 0.5 B= 0.5 0.5 0.5 0.5				