


Санкт-Петербургский государственный университет
Прикладная математика, информатика и искусственный интеллект

Отчет по учебной практике 2 (научно-исследовательской работе)
(семестр 4)

Марковские цепи и моделирование случайных сценариев.

Выполнила:

Барабашева Анастасия Дмитриевна,

группа 22.Б04-мм 

Научный руководитель:

Доктор физико-математических наук, доцент

Голяндина Нина Эдуардовна.

Кафедра статистического моделирования

Санкт-Петербург
2024

I. Введение

В ходе работы я познакомилась марковскими цепями по книге «Конечные цепи Маркова» авторы Кемени, Снелл. Я изучила необходимую теорию и выполнила упражнения, связанные с различными типами марковских процессов. Также построила марковскую модель интересного мне процесса, оценила характеристики при помощи формул и моделирования. Основная часть работы состоит из двух частей: теоретической и творческой. В теоретической части приведены основные определения и теоремы, которые использовались при решении задач, сами упражнения и их решения вместе с объяснениями и кодом. В творческой части приведено описание данных, по которым строилась матрица состояний, логика построения и основные результаты. Код и граф из творческой части находятся в приложении.

II. Основная часть

1. Марковские процессы

Основные определения и теоремы:

Определение 1. Конечным марковским процессом называется конечный стохастический процесс такой, что:

для любого высказывания p , истинность которого зависит лишь от исходов экспериментов до n -го,

$$\mathbb{P}\{f_n = s_j \mid (f_{n-1} = s_i) \wedge p\} = \mathbb{P}\{f_n = s_j \mid f_{n-1} = s_i\}$$

(предполагается, что $f_{n-1} = s_i$ и p совместимы).

Определение 2. Переходные вероятности на n -м шаге, которые будем обозначать $p_{ij}^{(n)}$, это

$$p_{ij}^{(n)} = \mathbb{P}\{f_n = s_j \mid f_{n-1} = s_i\}.$$

Определение 3. Конечной цепью Маркова называется конечный марковский процесс, для которого переходные вероятности $p_{ij}(n)$ не зависят от n . В этом случае они будут обозначаться p_{ij} . Элементы U называются состояниями.

Определение 4. Переходной матрицей цепи Маркова называется матрица P с элементами p_{ij} . Вектором начальных вероятностей (или начальным

распределением) называется вектор $\pi_0 = \{p_i^0\} = \{\mathbb{P}\{f_0 = s_i\}\}$.

Теорема 1. Пусть для данного марковского процесса p — какое-нибудь высказывание, истинность которого зависит от экспериментов после n -го. Тогда

$$\mathbb{P}\{f_n = s_j \mid (f_{n+1} = s_i \wedge p)\} = \mathbb{P}\{f_n = s_j \mid f_{n+1} = s_i\}.$$

Все состояния распадаются на классы эквивалентных состояний. Два состояния принадлежат одному классу эквивалентности, если они «сообщаются», т.е. если из одного состояния можно попасть в другое, и наоборот. Возникающее при этом частичное упорядочение показывает возможные направления, в которых может развиваться процесс. Минимальные элементы этого частичного упорядочения представляют особый интерес.

Определение 5. Минимальные элементы частичного упорядочения классов эквивалентности называются эргодическими множествами. Остальные элементы называются невозвратными множествами. Состояния, входящие в невозвратное множество, называются невозвратными состояниями. Состояние эргодического множества называют эргодическими (или возвратными) состояниями.

Так как при любом частичном упорядочении конечного множества должен быть по крайней мере один минимальный элемент, то в любой цепи Маркова найдется хотя бы одно эргодическое множество. Однако наличие невозвратных множеств не обязательно. Так бывает, если вся цепь состоит из одного эргодического множества или же имеется несколько эргодических множеств, не сообщающихся друг с другом. Если процесс выходит из невозвратного множества, то он никогда уже не может вернуться в это множество. В то же время, если он когда-нибудь попадет в эргодическое множество, то он никогда не сможет его покинуть. В частности, если эргодическое множество состоит лишь из одного состояния, то это состояние таково, что, попав в него, уже нельзя из него выйти. Такое состояние называется поглощающим. Поскольку из такого состояния нельзя перейти ни в какое другое состояние, следующая теорема полностью характеризует поглощающие состояния.

Теорема 2. Состояние s_i поглощающее тогда и только тогда когда, когда $p_{ii} = 1$.

Если имеется только один циклический подкласс, то мы называем класс эквивалентности регулярным, в противном случае мы говорим, что этот класс циклический. Если класс эквивалентности регулярен, то по прошествии достаточного времени процесс может оказаться в любом из состояний класса.

Типы цепей:

1. Цепи без невозвратных множеств. Если такая цепь состоит более чем из одного эргодического множества, то между этими множествами нет абсолютно никакого взаимодействия. Значит, мы имеем две или более изолированные цепи Маркова, объединенные вместе. Эти цепи можно изучать по отдельности, и поэтому, не теряя общности, мы можем предположить, что вся цепь представляет единственное эргодическое множество. Цепи, состоящие из единственного эргодического множества, называются эргодическими цепями.

1-А. Эргодическое множество регулярно. В этом случае цепь называется регулярной цепью Маркова. Как мы только что видели, в этом случае все достаточно высокие степени матрицы P должны быть положительными. Следовательно, из какого бы состояния процесс ни исходил, после достаточно большого количества шагов он может находиться в любом состоянии.

1-В. Эргодическое множество циклично. В этом случае цепь называется циклической цепью Маркова. Такая цепь имеет некоторый период d ($d > 1$), и ее состояния подразделяются на d циклических множеств. При данном начальном положении цепь движется по циклическим множествам в определенном порядке, возвращаясь в множество, содержащее начальное состояние, через d шагов. Мы также знаем, что по прошествии достаточно времени процесс может находиться в любом состоянии циклического множества, соответствующего выбранному моменту.

2. Цепи с невозвратными множествами. В таких цепях процесс движется в направлении к эргодическим множествам. Вероятность того, что процесс находится в одном из эргодических множеств, стремится к единице. Кроме того, попав в эргодическое множество, процесс не может его покинуть. Поэтому естественно классифицировать такие цепи по типу их эргодических множеств.

2-А. Каждое эргодическое множество состоит из единственного состояния. Такие цепи называются поглощающими цепями. В этом случае процесс рано или поздно останавливается в каком-то одном (поглощающем) состоянии. Этот тип процессов можно также охарактеризовать тем, что все эргодические состояния являются поглощающими.

2-В. Все эргодические множества регулярны, но не все они состоят из одного элемента.

2-С. Все эргодические множества цикличны.

2-Д. Имеются как циклические, так и регулярные эргодические мно-

жества.

Естественно, в каждом из этих классов можно производить дальнейшую классификацию, учитывая количество имеющихся эргодических множеств. Особый интерес представляет вопрос о том, имеется ли одно или несколько эргодических множеств.

Цепь, все состояния которой являются поглощающими, называется поглощающей цепью.

Удобно придать матрице P несколько иной канонический вид, объединив все эргодические состояния в одну группу и все невозвратные состояния — в другую группу. (Пусть, скажем, имеется s невозвратных и $r - s$ эргодических состояний.) Тогда каноническая форма будет

$$P = \begin{pmatrix} S & 0 \\ R & Q \end{pmatrix}$$

Здесь снова область 0 составлена целиком из нулей. Подматрица Q (размерности $s \times s$) описывает поведение процесса до выхода из множества невозвратных состояний, подматрица R размерности $s \times (r - s)$ отвечает переходам из невозвратных в эргодические состояния, а матрица S размерности $(r - s) \times (r - s)$ относится к процессу после достижения им эргодического множества. Степень Q стремится к 0. Значит, при возведении матрицы P во всё большие степени, все элементы последних s столбцов стремятся к 0. Если цепь поглощающая, матрица примет вид:

$$P = \begin{pmatrix} I & 0 \\ R & Q \end{pmatrix}$$

Определение 6. Для поглощающей цепи Маркова фундаментальной матрицей называется матрица $N = (I - Q)^{-1}$.

Теорема 3. $\{M_i[n_j]\} = N$ при $s_i, s_j \in T$.

Теорема 4. $\{D_i[n]\} = N_2 = N(2N_{dg} - I) - N_{sq}$ при $s_i, s_j \in T$.

Определение 7. Обозначим через n_j функцию, равную общему числу моментов времени, проводимых процессом в s_j (эта величина определена только для невозвратных состояний). Пусть также i_k^j есть функция, равная 1, если процесс после k шагов находится в s_j , и равная 0 в противном случае. Теперь мы дадим вероятностную интерпретацию матрицы N . Пусть T — множество невозвратных состояний. Теорема 5. $\{M_i[n_j]\} = N$, где $s_i, s_j \in T$.

Теорема 5. Если P — регулярная переходная матрица, то

- (i) Степени P^n стремятся (при $n \rightarrow \infty$) к вероятностной матрице A .

- (ii) Каждая строка матрицы A представляет один и тот же вероятностный вектор $\alpha = \{a_1, a_2, \dots, a_n\}$, т.е. $A = \xi\alpha$.
- (iii) Все компоненты α положительны.

Теорема 6. Если P — регулярная переходная матрица, а A и α — те же, что и в теореме 5, то

- (a) Для любого вероятностного вектора π последовательность векторов πP^n сходится к вектору α при $n \rightarrow \infty$.
- (b) Вектор α — единственный вероятностный вектор, для которого $\alpha P = \alpha$.
- (c) $PA = AP = A$.

Перейдем к выполнению упражнений.

Упражнение 1.

Условие:

Производится серия опытов, в каждом из которых подбрасывают две правильные монеты. Обозначим через s_1 , выпадение двух гербов, через s_2 — выпадение герба и решетки и через s_3 — выпадение двух решеток.

- (a) Найдите матрицу переходных вероятностей.
- (b) Если при данном бросании получились два герба, то какова вероятность, что два герба выпадут через три бросания?
- (c) Классифицируйте состояния.

Решение:

(a)

	s_1	s_2	s_3
s_1	0.25	0.5	0.25
s_2	0.25	0.5	0.25
s_3	0.25	0.5	0.25

(b) Вероятность перехода из любого состояния в состояние s_1 (ГГ) равно 0.25.

(c) Из любого состояния можем попасть в любое, значит эргодическая цепь. Возвращение из каждого состояния в каждое возможно за 1 шаг, НОД = 1, значит $d = 1$, период цепи равен 1, тип цепи 1-А.

Упражнение 2.

Условие:

Производится такая последовательность испытаний. В первом опыте бросается правильная монета. Если в $n - 1$ -м опыте выпал герб, то в n -м опыте бросается правильная монета; если n -м опыте выпала решетка, то в $n - 1$ -м опыте бросается такая монета, для которой вероятность выпадения герба равна $1/n$. Каковы переходные вероятности? С каким процессом мы имеем дело?

Решение:

Матрица переходов:

	P	Г
P	$\frac{n-1}{n}$	$\frac{1}{n}$
Г	0.5	0.5

По определению:

Конечным марковским процессом называется конечный стохастический процесс такой, что:

для любого высказывания p , истинность которого зависит лишь от исходов экспериментов до n -го,

$$\mathbb{P}\{f_n = s_j \mid (f_{n-1} = s_i) \wedge p\} = \mathbb{P}\{f_n = s_j \mid f_{n-1} = s_i\}$$

Согласно определению, мы должны проверить, что вероятность перехода в n -м шаге зависит только от состояния на $(n - 1)$ -м шаге и не зависит от предшествующих состояний (или от высказываний p , истинность которых зависит лишь от исходов экспериментов до n -го). Проверим, зависят ли эти вероятности от предыдущих состояний или только от текущего состояния S_{n-1} .

1. Если $S_{n-1} = \Gamma$, то $P(S_n = \Gamma)$ и $P(S_n = P)$ равны $\frac{1}{2}$ и зависят только от текущего состояния S_{n-1} .

2. Если $S_{n-1} = P$, то $P(S_n = \Gamma)$ и $P(S_n = P)$ равны $\frac{1}{n}$ и $1 - \frac{1}{n}$ соответственно, и зависят только от текущего состояния S_{n-1} .

Таким образом, вероятность перехода в состояние S_n зависит только от состояния S_{n-1} и не зависит от предшествующих состояний или высказываний p , истинность которых зависит лишь от исходов экспериментов до n -го шага. Следовательно, процесс является марковским процессом.

Упражнение 3.

Условие:

Согласно «Введению в конечную математику» в стране Оз никогда не бывает подряд двух ясных дней. Если сегодня ясно, то завтра будет плохая погода (снег или дождь с равной вероятностью). Если сегодня снег (или дождь), то погода на следующий день не изменится с вероятностью $1/2$. Если все-таки она изменится, то лишь в половине случаев будет ясно. В текущем упражнении разрешены только 2 состояния: хорошая погода и плохая погода. Покажите, что процесс по-прежнему остается цепью Маркова, найдите его переходную матрицу.

Решение:

По определению:

Конечным марковским процессом называется конечный стохастический процесс такой, что:

для любого высказывания p , истинность которого зависит лишь от исходов экспериментов до n -го,

$$\mathbb{P}\{f_n = s_j \mid (f_{n-1} = s_i) \wedge p\} = \mathbb{P}\{f_n = s_j \mid f_{n-1} = s_i\}$$

Для наших состояний и переходов:

- Если $S_n = X$, то S_{n+1} зависит только от S_n и всегда переходит в $S_{n+1} = \Pi$ с вероятностью 1.
- Если $S_n = \Pi$, то вероятность перехода в S_{n+1} зависит только от текущего состояния S_n и распределяется как $P(S_{n+1} = 0) = \frac{3}{4}$ и $P(S_{n+1} = 1) = \frac{1}{4}$.

Так как переходные вероятности зависят только от текущего состояния, а не от предшествующих состояний или других событий, процесс обладает марковским свойством.

Матрица переходов:

	X	Π
X	0	1
Π	0.25	0.75

Упражнение 4.

Условие:

Семеро мальчиков играют в мяч. Первый мальчик всегда бросает мяч второму. Второй мальчик с равными вероятностями бросает мяч третьему

или седьмому. Третий мальчик оставляет мяч у себя, когда он к нему попадает. Четвертый мальчик всегда бросает шестому. Пятый мальчик с равными вероятностями бросает четвертому, шестому или седьмому мальчику. Шестой мальчик всегда бросает мяч четвертому. Седьмой мальчик бросает мяч первому или четвертому с равными вероятностями.

- (a) Напишите переходную матрицу P .
- (b) Классифицируйте состояния.
- (c) Представьте P в канонической форме.
- (d) Объясните, что в данном случае означает попадание цепи в конце концов в одно из эргодических множеств.
- (e) Пусть мяч у пятого мальчика. Найдите математическое ожидание и дисперсию числа раз, когда мяч достается седьмому мальчику. Найдите также математическое ожидание и дисперсию времени до достижения эргодического множества.

Решение: (a) Переходная матрица.

Состояния:

1. Мяч у первого мальчика
2. Мяч у второго мальчика
- ...
7. Мяч у седьмого мальчика

Переходная матрица P будет выглядеть следующим образом:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 \end{pmatrix}$$

(b) Классификация состояний.

- Поглощающее состояние: 3
- Эргодическое множество: $\{4, 6\}$
- В состояние 5 можно попасть только начав из него.

(с) Каноническая форма матрицы P .

Каноническая форма переходной матрицы P будет выглядеть следующим образом:

$$P = \begin{pmatrix} S & 0 \\ R & Q \end{pmatrix}$$

где Q - это матрица поведения процесса до выхода из множества невозвратных состояний, R - эта матрица отвечает переходам из невозвратных в эргодическое состояние, и S - относится к процессу после достижения им эргодического множества.

$$Q = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & 0 \end{pmatrix}$$

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \end{pmatrix}$$

Если нам неважно, какое эргодическое состояние было достигнуто, то матрица примет вид:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \end{pmatrix}$$

(d) Объяснение попадания в эргодическое множество.

Попадание цепи в одно из эргодических множеств означает, что процесс со временем будет находиться только в одном из этих множеств состояний и не покинет его. В данной задаче это означает, что мяч либо будет переходить между состояниями $\{4, 6\}$, либо попадет в поглощающее состояние 3. В итоге спустя время мяч будет у кого-то из $\{3, 4, 6\}$.

(e) Ожидание и дисперсия числа раз, когда мяч достается седьмому мальчику.

Если мяч находится у пятого мальчика, то можно вычислить математическое ожидание и дисперсию числа раз, когда мяч достается седьмому мальчику, используя фундаментальную матрицу N .

$$N = (I - Q)^{-1}$$

$$N = \begin{pmatrix} \frac{4}{3} & \frac{4}{3} & 0 & \frac{2}{3} \\ \frac{1}{3} & \frac{4}{3} & 0 & \frac{2}{3} \\ \frac{2}{9} & \frac{2}{9} & 1 & \frac{4}{9} \\ \frac{2}{3} & \frac{2}{3} & 0 & \frac{4}{3} \end{pmatrix}$$

Изучаем процесс на пути из 5 в 7. (3-ая строка и 4-ая строка)

$M_5[n_7] = 4/9$ - 3-ая строка 4-ый столбец.

Для нахождения дисперсии найдем N_2 - матрица, где каждый элемент является квадратом соответствующего элемента матрицы N .

$$N_2 = N(2N_{dg} - I) - N_{sq}$$

$$N_{dg} = \begin{pmatrix} \frac{4}{3} & 0 & 0 & 0 \\ 0 & \frac{4}{3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{4}{3} \end{pmatrix}$$

$$N = \begin{pmatrix} \frac{16}{9} & \frac{16}{9} & 0 & \frac{4}{9} \\ \frac{1}{9} & \frac{16}{9} & 0 & \frac{4}{9} \\ \frac{4}{81} & \frac{4}{81} & 1 & \frac{16}{81} \\ \frac{4}{9} & \frac{4}{9} & 0 & \frac{16}{9} \end{pmatrix}$$

$$N_2 = \begin{pmatrix} \frac{4}{9} & \frac{4}{9} & 0 & \frac{6}{9} \\ \frac{4}{9} & \frac{4}{9} & 0 & \frac{6}{9} \\ 0.321 & 0.321 & 1 & 0.543 \\ \frac{6}{9} & \frac{6}{9} & 0 & \frac{4}{9} \end{pmatrix}$$

Теперь найдем величину, показывающую сколько шагов нужно сделать для попадания в эргодическое множество (т.е. полное время, которое процесс проводит в невозвратных состояниях).

M_5 = сумма элементов строки 3 матрицы (соответствует состоянию 5) $N = \frac{2}{9} + \frac{2}{9} + 1 + \frac{4}{9} = 17/9 = 1.778$

D_5 = сумма элементов строки 3 матрицы (соответствует состоянию 5) $N_2 = 0.321 * 2 + 1 + 0.543 = 2.185$.

Вычисления матриц проводились кодом на Python, приведенном ниже.

```
import numpy as np

Q = np.array([
    [0, 1, 0, 0],
    [0, 0, 0, 0.5],
    [0, 0, 0, 1/3],
    [0.5, 0, 0, 0]
])

I = np.eye(Q.shape[0])
N = np.linalg.inv(I - Q)
print(N)

diag_matrix = np.diag(np.diag(N))

N_2 = np.dot(N, 2 * diag_matrix - I) - N**2
print(N_2)
```

Упражнение 5.

Условие:

Рассмотрим следующую переходную матрицу цепи Маркова:

$$P = \begin{pmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ \frac{3}{4} & 0 & \frac{1}{4} \\ 0 & 1 & 0 \end{pmatrix}$$

- (a) Регулярна ли цепь?
- (b) Найдите α , A и Z .
- (c) Найдите M и M_2 .
- (d) Найдите матрицу ковариаций.
- (e) Пользуясь методами поглощающих цепей, найдите математическое ожидание времени первого достижения состояния s_1 , исходя из s_3 . Проверьте результат, сравнив его с (c).

Решение:

- (a) Если при некотором N все элементы матрицы P^N отличны от 0, то матрица регулярна. Поищем такое N .

$$P^2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ 0.375 & 0.5 & 0.125 \\ 0.75 & 0 & 0.25 \end{pmatrix}$$

$$P^3 = \begin{pmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ 0.5625 & 0.25 & 0.1875 \\ 0.375 & 0.5 & 0.125 \end{pmatrix}$$

$N = 3$ - цепь регулярна.

- (b) Для нахождения вектора $\alpha = (a_1, a_2, a_3)$ ищем решение $\alpha * P = \alpha$:

$$\begin{cases} a_1 + a_2 + a_3 = 1 \\ a_1 = 1/2 * a_1 + 3/4 * a_2 \\ a_2 = 1/3 * a_1 + a_3 \\ a_3 = a_2 \end{cases}$$

Решение $\alpha = (\frac{1}{2}, \frac{1}{3}, \frac{1}{6})$, $A = \epsilon * a$.

$$P^3 = \begin{pmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \end{pmatrix}$$

$$P^2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ 0.5 & 0.333 & 0.1667 \\ 0.5 & 0.333 & 0.1667 \end{pmatrix}$$

$$Z = (I - (P - A))^{-1}$$

$$Z = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{6} & \frac{7}{9} & \frac{1}{18} \\ \frac{-1}{3} & \frac{4}{9} & \frac{8}{9} \end{pmatrix}$$

Вычисление матрицы проводились кодом на Python, приведенном ниже.

```
import numpy as np

P = np.array([
    [1/2, 1/3, 1/6],
    [3/4, 0, 1/4],
    [0, 1, 0]
])

A = np.array([
    [1/2, 1/3, 1/6],
    [1/2, 1/3, 1/6],
    [1/2, 1/3, 1/6]
])

I = np.eye(3)
Z = np.linalg.inv(I - (P - A))
print(Z)
```

(с) Матрица средних времен достижения: $M = (I - Z + E * Z_{dg})D$, здесь Z_{dg} - матрица, составленная из диагональных элементов матрицы Z , E - матрица, все элементы которой равны 1, матрица D по диагонали имеет

элементы, равные $1/a_i$ из вектора α , остальные нули.

$$M = \begin{pmatrix} 2 & \frac{7}{6} & \frac{16}{5} \\ \frac{5}{3} & 3 & 5 \\ \frac{8}{3} & 1 & 6 \end{pmatrix}$$

$M_2 = M(2 * Z_{dg} * D - I) + 2(ZM - E(ZM)_{dg}) - M_{sq}$ - матрица дисперсий.

$$M_2 = \begin{pmatrix} 2 & 7\frac{7}{9} & 33\frac{7}{9} \\ 5\frac{7}{9} & 2 & 33\frac{5}{9} \\ 5\frac{7}{9} & 5\frac{5}{9} & 22 \end{pmatrix}$$

Вычисления матриц проводились кодом на Python, приведенном ниже.

```
import numpy as np

Z = np.array([
    [1, 0, 0],
    [1/6, 7/9, 1/18],
    [-1/3, 4/9, 8/9]
])

E = np.ones((3, 3))

Z_dg = np.diag(np.diag(Z))

D = np.array([
    [2, 0, 0],
    [0, 3, 0],
    [0, 0, 6]
])

I = np.eye(3)
M = np.dot(I - Z + np.dot(E, Z_dg), D)
print(M)
M_2 = np.dot(M, (2 * np.dot(Z_dg, D) - I)) +
      2 * (np.dot(Z, M) - np.diag(np.diag(np.dot(Z, M)))) - M**2

print(M_2)
```

(d) Найдём матрицу ковариаций.

$$C_{ij} = a_i \cdot z_{ij} + a_j \cdot z_{ji} - a_i \delta_{ij} - a_i \dot{a}_j$$

$$\delta_{ij} = 0 \quad \text{when } i \neq j$$

$$\delta_{ij} = 1 \quad \text{when } i = j$$

$$C_{11} = \frac{1}{4}$$

и тд

$$C = \begin{pmatrix} \frac{1}{4} & -\frac{1}{9} & -0.1389 \\ -\frac{1}{9} & 0.074 & 0.037 \\ -0.1389 & 0.037 & 0.102 \end{pmatrix}$$

Диагональные элементы матрицы C дают предельные дисперсии для времен пребывания в каждом состоянии. Вычисление матрицы проводились кодом на Python, приведенном ниже.

```
import numpy as np
```

```
Z = np.array([
    [1, 0, 0],
    [1/6, 7/9, 1/18],
    [-1/3, 4/9, 8/9]
])
```

```
alpha = np.array([1/2, 1/3, 1/6])
```

```
n = Z.shape[0]
C = np.zeros((n, n))
```

```
for i in range(n):
    for j in range(n):
        delta_ij = 1 if i == j else 0
        C[i, j] = alpha[i] * Z[i, j] + alpha[j] * Z[j, i] -
            alpha[i] * delta_ij - alpha[i] * alpha[j]
```


print (C)

(е) Считаем, что матрица уже приведена к каноническому виду. Цепь без невозвратных состояний, имеющая единственное эргодическое множество, $s = 0$.

$$N = (I - Q)^{-1} = 1$$

1 и есть мат.ожидание из s_3 в s_1 . Если находимся в состоянии s_3 , то среднее время первого достижения s_1 это $8/3$. (элемент (3,1) из M). Результаты получились разные: 1 и $8/3$.

Упражнение 6.

Общее (конечное) случайное блуждание определяется следующим образом. Состояния перенумеровываются s_0, s_1, \dots, s_n . Если процесс находится в s_i , то он переходит в s_{i-1} с вероятностью q_i , остаётся в s_i с вероятностью r_i и переходит в s_{i+1} с вероятностью p_i . (Здесь $p_i + q_i + r_i = 1, q_0 = 0, p_n = 0$.)

- (а) При каких условиях случайное блуждание эргодично?
- (b) Из уравнения $\alpha P = \alpha$ выведите методом математической индукции, что $a_{i+1}q_{i+1} = a_i p_i$.
- (с) Докажите, что эргодическое случайное блуждание обратимо.
- (d) Найдите формулу для неподвижного вектора α .

Решение: Составим левый участок матрицы. Получается r_i на диагонали, p_i над диагональю, q_i под диагональю.

	s_0	s_1	s_2	s_3
s_0	r_0	p_0	0	0
s_1	q_1	r_1	p_1	0
s_2	0	q_2	r_2	p_2
s_3	0	0	q_3	r_3

- (а) Видно, что такими шагами можно попасть из любого состояния в любое.
 $p_i > 0$ и $q_i > 0$.
НОД длин всех путей, возвращающихся в одно и тоже состояние, должен быть 1,
значит $r_i > 0$.

(b)

$$\alpha P = \alpha$$

$$a_{i+1}q_{i+1} = a_i p_i$$

База индукции: рассмотрим $i = 0$.

Тогда $a_0 = r_0 a_0 + q_1 a_1$ (из $\alpha P = \alpha$)

По условию $r_0 + p_0 = 1$ (и $q_0 = 0$)

Значит $p_0 = 1 - r_0$. $a_0(1 - r_0) = q_1 a_1$, т.е. $a_0 p_0 = q_1 a_1$. Шаг индукции:

Предположим, что для некоторого i справедливо $a_{i+1}q_{i+1} = a_i p_i$.

Докажем, что это справедливо и для $i+1$. Из $\alpha P = \alpha$ имеем: (для $i+1$).

$a_{i+1} = a_i p_i + r_{i+1} a_{i+1} + a_{i+2} q_{i+2}$. Из предположения индукции $a_{i+1} q_{i+1} =$

$a_i p_i$, значит $a_{i+1} = a_{i+1} q_{i+1} + r_{i+1} a_{i+1} + a_{i+2} q_{i+2}$, $a_{i+1}(1 - q_{i+1} - r_{i+1}) =$

$a_{i+2} q_{i+2}$. Из условия $p_{i+1} + q_{i+1} + r_{i+1} = 1$, значит $p_{i+1} = -q_{i+1} - r_{i+1} + 1$.

Получим $a_{i+2} q_{i+2} = a_{i+1} p_{i+1}$

(с) Докажем, что эргодическое случайное блуждание обратимо. Нужно доказать, что $P = \hat{P}$, т.е. $\hat{P} = \frac{a_j p_{ji}}{a_i}$

$$p_{ij} = \frac{a_i p_{ij}}{a_i}$$

Матрица так устроена, что есть 3 случая:

- $p_{ij} = p_{ji} = 0$ (выполняется)
- $p_{ij} = p_{ji}$, если $j = i$, т.е. $p_{ii} a_i = a_i p_{ii}$ (выполняется)

В случае $|j - i| = 1$, выполняется благодаря пункту (b)

$$a_{i+1}q_{i+1} = a_i p_i$$

(d) Знаем, что $\sum a_i = 1$ и $a_{i+1}q_{i+1} = a_i p_i$.

$$a_1 q_1 = a_0 p_0, \text{ т.е. } a_1 = a_0 \frac{p_0}{q_1}$$

$$a_2 q_2 = a_1 p_1, \text{ т.е. } a_2 = a_1 \frac{p_1}{q_2} = a_0 \frac{p_1 p_2}{q_2 q_1}$$

и так далее.

В общем случае формула $a_i = a_0 \prod_{j=0}^{i-1} \frac{p_{j+1}}{q_j}$.

Найдём a_0 :

$$a_0 + \sum_{i=1}^n a_0 \prod_{j=0}^{i-1} \frac{p_{j+1}}{q_j} = 1$$

$$a_0 \left(1 + \sum_{i=1}^n \prod_{j=0}^{i-1} \frac{p_{j+1}}{q_j} \right) = 1 \quad \Rightarrow \quad a_0 = \frac{1}{1 + \sum_{i=1}^n \prod_{j=0}^{i-1} \frac{p_{j+1}}{q_j}}$$

Итак,

$$a_i = \frac{\prod_{j=0}^{i-1} \frac{p_{j+1}}{q_j}}{1 + \sum_{k=1}^n \prod_{j=0}^{k-1} \frac{p_{j+1}}{q_j}}$$

2. Построение и моделирование марковской модели

Основная идея: имеются данные о действиях пользователя внутри онлайн-курса. Хотим оценить сколько шагов пользователь тратит на то, чтобы попасть в одно из поглощающих состояний и какая вероятность завершения курса конкретным пользователем.

Данные о действиях пользователя включают в себя id пользователя, название "шага совершенного пользователем, и время этого шага – так для каждого пользователя можно было представить его действия в виде последовательности состояний-узлов, следующих друг за другом. Узлы цепочки представляют события, происходящие с нашим пользователем. Данные собраны с курса, который проходили пользователи, поэтому состояния там такие:

- «start General course» (в матрице - sGc),
- «successful class» (в матрице - sc),
- «Dropped General» (в матрице - DG),
- «level up» (в матрице - lu),
- «Finished course» (в матрице - Fc),
- «fell asleep» (в матрице - fa),
- «App session» (в матрице - As),
- «level down» (в матрице - ld).

Из них финальные состояния это: (они же и являются поглощающими) «Finished course», «Dropped General», «fell asleep».

Матрица переходов заполнялась по данным из датасета (ссылка на датасет в списке литературы), затем высчитывался начальный вектор. Вычисления производились на языке программирования Python. Матрица состояний:

	sGc	sc	DG	lu	Fc	fa	As	ld
sGc	0.000000	0.827221	0.000000	0.027710	0.000000	0.000000	0.125509	0.019560
sc	0.000000	0.797008	0.019580	0.008538	0.010657	0.042755	0.115106	0.006356
DG	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
lu	0.000000	0.854460	0.018779	0.009390	0.000000	0.032864	0.028169	0.056338
Fc	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
fa	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
As	0.049709	0.171356	0.005534	0.002720	0.001501	0.000281	0.766929	0.001970
ld	0.000000	0.867925	0.000000	0.094340	0.006289	0.000000	0.012579	0.018868

Матрица состояний, приведенная к канонической форме:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.827 & 0.028 & 0.125 & 0.020 \\ 0.020 & 0.01 & 0.04 & 0 & 0.8 & 0 & 0 & 0 \\ 0.02 & 0 & 0.03 & 0 & 0.85 & 0.01 & 0.03 & 0.06 \\ 0.01 & 0.002 & 0.0003 & 0.05 & 0.17 & 0.003 & 0.77 & 0.002 \\ 0 & 0.006 & 0 & 0 & 0.868 & 0.094 & 0.013 & 0.019 \end{pmatrix}$$

После этого в коде составлялся граф процесса, который приведен в приложении 2. Далее была найдена фундаментальная матрица, рассчитаны математическое ожидание и дисперсия до поглощения. Затем было проведено моделирование, благодаря которому получили новые значения дисперсии, математического ожидания и вероятностей попадания в поглощающие состояния. Ниже приведены основные результаты.

Характеристики, полученные расчетом по формулам.

Вероятности попадания в поглощающие состояния:

Finished course: 0.148431

Dropped General: 0.294281

fell asleep: 0.557288

Мат. ожидание и дисперсия:

Expected Time to Absorption: 21.62065082733475

Variance of Time to Absorption: 441.12230685916813.

Характеристики, полученные моделированием.

Вероятности попадания в поглощающие состояния:

Finished course: 0.1487

Dropped General: 0.2932

fell asleep: 0.5581

Мат. ожидание и дисперсия:

Expected Time to Absorption: 21.58539

Variance of Time to Absorption: 440.34006854790005.

Разность характеристик (по модулю) :

Expected Time to Absorption: 0.03526082733474922

Variance of Time to Absorption: 0.7822383112680882

Probabilities of Reaching Each End State:

Finished course: 0.0003

Dropped General: 0.0011

fell asleep: 0.0008.

Можем сделать вывод о том, что формулы точны для изучения марковских процессов.

III. Заключение

В ходе выполнения работы я укрепила знания в области теории вероятностей, познакомилась с марковскими процессами и их устройством, изучила теоретические марковских цепей. Выполнила несколько упражнений на свойства, их доказательства и расчет характеристик марковских цепей. Составила марковскую цепь по интересному мне процессу, моделированием проверила формулы для расчета характеристик марковских процессов, нарисовала граф процесса и научилась моделировать марковский процесс

на языке программирования Python.

IV. Список литературы

1. авторы Кемени, Снелл. Конечные цепи Маркова.
2. ссылка на dataset.

V. Приложение

Обработка данных

```

import numpy as np
import pandas as pd

filename = 'worklist.csv'
sep1 = ';'
start_state = 'start General course'
end_states = ['Finished course', 'Dropped General', 'fell asleep']
steps = 1000

# Чтение данных без заголовков
df = pd.read_csv(filename, sep=sep1, header=None)

# Задание имен столбцов вручную
df.columns = ['student_id', 'state', 'event_time']

# Преобразование времени событий и сортировка
df['event_time'] = df['event_time'].str.replace(',', '.').astype(float)
df = df.sort_values(by=['student_id', 'event_time'])

states = df['state'].unique().tolist()

# Создание матрицы переходов
calc_matrix = np.zeros((len(states), len(states)), dtype=int)

for i in range(len(df) - 1):
    if df.iloc[i]['student_id'] == df.iloc[i + 1]['student_id']:
        from_state = df.iloc[i]['state']
        to_state = df.iloc[i + 1]['state']
        calc_matrix[states.index(from_state), states.index(to_state)] += 1

for end_state in end_states:
    calc_matrix[states.index(end_state), :] = 0

TransitionMatrix = np.zeros_like(calc_matrix, dtype=float)

for i in range(len(calc_matrix)):
    row_sum = calc_matrix[i].sum()
    if row_sum > 0:
        TransitionMatrix[i] = calc_matrix[i] / row_sum

# Обработка конечных состояний
for i in range(len(TransitionMatrix)):
    if TransitionMatrix[i].sum() == 0:
        TransitionMatrix[i, i] = 1

# Начальный вектор
InitialVector = np.zeros(len(states))
InitialVector[states.index(start_state)] = 1

# Расчет вероятностей
ProbabilityMatrix = np.linalg.matrix_power(TransitionMatrix, steps)
ProbabilityVector = np.dot(InitialVector, ProbabilityMatrix)

# Округление вероятностей
ProbabilityVector = np.round(ProbabilityVector, 2)
Result = dict(zip(states, ProbabilityVector))
print('\nProbability Vector:\n', Result)

# Вероятностная матрица
ProbabilityMatrix = np.linalg.matrix_power(TransitionMatrix, steps)

```



```

Probability Vector:
{'start General course': 0.0, 'successful class': 0.0, 'Dropped General': 0.29, 'level up': 0.0, 'Finished course': 0.15, 'fell as

```

Делаем вывод по вероятностям:

Finished course: 0.148431

Dropped General: 0.294281

fell asleep: 0.557288

Граф

Создание графа

```
!apt-get install graphviz
!pip install pydot pillow
```



```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
graphviz is already the newest version (2.42.2-6).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
Requirement already satisfied: pydot in /usr/local/lib/python3.10/dist-packages (1.4.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (9.4.0)
Requirement already satisfied: pyparsing>=2.1.4 in /usr/local/lib/python3.10/dist-packages (from pydot) (3.1.2)
```

```
import networkx as nx
```

```
G = nx.MultiDiGraph()
```

```
for i in range(len(TransitionMatrix)):
    for j in range(len(TransitionMatrix)):
        if round(TransitionMatrix[i][j], 2) != 0:
            G.add_edge(states[i], states[j], weight=round(TransitionMatrix[i][j], 2), label="{:.02f}".format(TransitionMatrix[i][j]))
```

```
# Добавление размеров (вероятностей) к узлам графа
```

```
for i in range(len(TransitionMatrix)):
    G.add_node(states[i], size=int(ProbabilityVector[i] * 100))
```

```
# Сохранение графа в файл .gexf
```

```
nx.write_gexf(G, "Graph.gexf")
```

Сохранение графа

```
import pydot
```

```
import fitz
```

```
from PIL import Image
```

```
# Сохранение графа в файл .dot
```

```
nx.drawing.nx_pydot.write_dot(G, "Graph.dot")
```

```
# Преобразование графа в PDF
```

```
(graph,) = pydot.graph_from_dot_file('Graph.dot')
```

```
graph.write_pdf('Graph.pdf')
```

Расчет по формулам

```
transient_states = [state for state in states if state not in end_states]
```

```
Q = np.zeros((len(transient_states), len(transient_states)))
```

```
for i, state1 in enumerate(transient_states):
```

```
    for j, state2 in enumerate(transient_states):
```

```
        Q[i, j] = TransitionMatrix[states.index(state1), states.index(state2)]
```

```
R_matrix = np.zeros((len(transient_states), len(end_states)))
```

```
for i, state1 in enumerate(transient_states):
```

```
    for j, state2 in enumerate(end_states):
```

```
        R_matrix[i, j] = TransitionMatrix[states.index(state1), states.index(state2)]
```

```
# Находим матрицу фундаментальных состояний
```

```
I = np.eye(len(transient_states))
```

```
N = np.linalg.inv(I - Q)
```

```
# Математическое ожидание времени до поглощения
```

```
expected_times = N.sum(axis=1)
```

```
# Дисперсия времени до поглощения
```

```
var_times = (2 * N - I).dot(expected_times) - expected_times**2
```

```
# Результаты
```

```
expected_time_to_absorption = dict(zip(transient_states, expected_times))
```

```
variance_time_to_absorption = dict(zip(transient_states, var_times))
```

```
print('\nExpected Time to Absorption:\n', expected_time_to_absorption['start General course'])
```

```
print('\nVariance of Time to Absorption:\n', variance_time_to_absorption['start General course'])
```



```
Expected Time to Absorption:
21.62065082733475
```

```
Variance of Time to Absorption:
```


441.12230685916813

Моделирование

```
import numpy as np

num_simulations = 100000 # Количество симуляций

# Функция для моделирования процесса
def simulate_chain(transition_matrix, states, start_state, end_states, max_steps=1000):
    current_state = start_state
    time = 0
    while current_state not in end_states and time < max_steps:
        current_index = states.index(current_state)
        next_state = np.random.choice(states, p=transition_matrix[current_index])
        current_state = next_state
        time += 1
    return current_state, time # Возвращаем достигнутое состояние и время

# Счетчики для конечных состояний
end_state_counts = {end_state: 0 for end_state in end_states}
times_to_absorption = []

# Моделирование симуляций
for _ in range(num_simulations):
    end_state, time = simulate_chain(TransitionMatrix, states, start_state, end_states, steps)
    end_state_counts[end_state] += 1
    times_to_absorption.append(time)

# Расчет вероятностей попадания в каждое конечное состояние
probabilities = {end_state: count / num_simulations for end_state, count in end_state_counts.items()}

# Расчет математического ожидания и дисперсии
expected_time = np.mean(times_to_absorption)
variance_time = np.var(times_to_absorption)

# Вывод результатов
print('\nExpected Time to Absorption:', expected_time)
print('Variance of Time to Absorption:', variance_time)
print('\nProbabilities of Reaching Each End State:')
for end_state, probability in probabilities.items():
    print(f'{end_state}: {probability:.4f}')
```



```
Expected Time to Absorption: 21.58539
Variance of Time to Absorption: 440.34006854790005

Probabilities of Reaching Each End State:
Finished course: 0.1487
Dropped General: 0.2932
fell asleep: 0.5581
```

Итог

Ниже приведена разность расчетных значений со значениями, полученными моделированием.

```
print('\nExpected Time to Absorption:', expected_time_to_absorption['start General course'] - expected_time)
print('Variance of Time to Absorption:', variance_time_to_absorption['start General course'] - variance_time)

print('\nProbabilities of Reaching Each End State:')
i=0
for end_state, probability in probabilities.items():
    print(f'{end_state}: {B[0][i]-probability:.4f}')
    i+=1
```



```
Expected Time to Absorption: 0.03526082733474922
Variance of Time to Absorption: 0.7822383112680882

Probabilities of Reaching Each End State:
Finished course: -0.0003
Dropped General: 0.0011
fell asleep: -0.0008
```