



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение
высшего образования

«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФГБОУ ВО «ИГУ»

Факультет бизнес-коммуникаций и информатики

Кафедра естественнонаучных дисциплин

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине Б1.В.ДВ.01.02 Основы вычислительной математики

направление подготовки 09.03.03 Прикладная информатика

направленность (профиль) Прикладная информатика (разработка программного обеспечения)

Одобен
УМК факультета бизнес-коммуникаций
и информатики

Разработан в соответствии с ФГОС ВО

с учетом требований проф. стандарта

Председатель УМК

М.Г. Синчурина
ФИО, должность, ученая степень, звание

подпись, печать

Разработчики:



(подпись)

доцент

(занимаемая должность)

Е.А. Лутковская

(инициалы, фамилия)

Цель фонда оценочных средств. Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «Б1.В.ДВ.01.02 Основы вычислительной математики». Перечень видов оценочных средств соответствует рабочей программе дисциплины.

Фонд оценочных средств включает контрольные материалы для проведения текущего контроля (в следующих формах: практическое задание, тест, проект) и промежуточной аттестации в форме вопросов и заданий к зачету.

Структура и содержание заданий – задания разработаны в соответствии с рабочей программой дисциплины «Б1.В.ДВ.01.02 Основы вычислительной математики».

1. Паспорт фонда оценочных средств

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-1 ППК-Р1 Способен разрабатывать и отлаживать программный код	ПК-1 ППК-Р1.1 Выполняет формализацию и алгоритмизацию поставленных задач для разработки программного кода	• ПК-1.1 ППК-Р1.1. З-2. Знает нотации и программное обеспечение для графического отображения алгоритмов • ПК-1.1 ППК-Р1.1. У-1. Умеет использовать методы и приемы формализации и алгоритмизации поставленных задач
	ПК-1 ППК-Р1.2 Формализует задачу ИТ отрасли в язык естественнонаучных дисциплин	• ПК-1.6 ППК-Р1.2. З-1. Знает основные разделы математики, применяемые для анализа и моделирования непрерывных процессов и дискретных систем в прикладных задачах • ПК-1.6 ППК-Р1.2. У-1. Умеет выбирать адекватный математический аппарат для формализованного описания сущностей и отношений на основе ТЗ и бизнес-требований
	ПК-1 ППК-Р1.3 Осуществляет обоснованный выбор методов и алгоритмов для программной реализации формальной математической модели	• ПК-1.7 ППК-Р1.3. З-1. Знает основные классы методов программной реализации моделей и критерии выбора алгоритмов
	ПК-1 ППК-Р1.4 Разрабатывает программный код с использованием языков программирования	• ПК-1.2 ППК-Р1.4 З-1. Знает синтаксис выбранного языка программирования, особенности программирования на этом языке, стандартные библиотеки языка программирования • ПК-1.2 ППК-Р1.4 У-2. Умеет использовать выбранную среду программирования
	ПК-1 ППК-Р1.5 Оформляет программный код в соответствии с установленными требованиями	• ПК-1.3 ППК-Р1.5 З-1. Знает нормативно-технические документы (стандарты и регламенты), определяющие требования к оформлению программного кода
	ПК-1 ППК-Р1.6 Работает с системой управления версиями программного кода	• ПК-1.4 ППК-Р1.6. З-1. Знает возможности используемой системы управления версиями и вспомогательных инструментальных программных средств
	ПК-1 ППК-Р1.7 Проверяет и отлаживает программный код	• ПК-1.5 ППК-Р1.7 З-1. Знает методы и приемы отладки программного кода • ПК-1.5 ППК-Р1.7 З-2. Знает типы и форматы сообщений об ошибках, предупреждений • ПК-1.5 ППК-Р1.7 У-2. Умеет отлаживать программный код на уровне программных модулей

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-19 Способен осуществлять профессиональную деятельность в условиях командного взаимодействия, саморазвития и профессионального лидерства (УНК-01, УНК-02, УНК-03, УНК-04)	ПК-19 .1 УНК-01.01 Осуществляет коммуникации в профессиональной деятельности	• ПК-19.1 УНК-01.01 У-3 Способен формулировать и понимать технологические и бизнес-требования • ПК-19.1 УНК-01.01 З-1 Знает устоявшуюся в отрасли и компании терминологию
	ПК-19 .2 УНК-02.01 Принимает участие в групповом взаимодействии в ходе профессиональной деятельности	• ПК-19.2 УНК-02.01 У-4. Умеет давать конструктивную обратную связь • ПК-19.2 УНК-02.01 З-1. Знает основы командной работы, роли и ответственности каждого участника."
	ПК-19 .3 УНК-03.01 Демонстрирует системное мышление при решении задач профессиональной деятельности	• ПК-19.3 УНК-03.01 У-1. Умеет анализировать задачи, учитывая взаимосвязи между их компонентами • ПК-19.3 УНК-03.01 З-1. Знает основные принципы и методы анализа и синтеза систем
	ПК-19 .4 УНК-03.02 Осуществляет анализ и планирование деятельности	• ПК-19.4 УНК-03.02 З-1. Знает методы анализа деятельности и показатели эффективности
	ПК-19 .5 УНК-02.02 Проявляет лидерство и осуществляет наставничество	• ПК-19.5 УНК-02.02 У-3. Умеет делиться знаниями и опытом, оказывать поддержку и осуществлять наставничество коллегам
	ПК-19 .6 УНК-04.01 Планирует и осуществляет самообучение	• ПК-19.6 УНК-04.01 .У-1. Умеет самостоятельно определять области для профессионального развития и формулировать цели обучения • ПК-19.6 УНК-04.01 .З-1. Знает современные методы и ресурсы для самостоятельного обучения • ПК-19.6 УНК-04.01 .З-3. Знает основы саморегуляции и мотивации для поддержания постоянного профессионального роста"
	ПК-19 .7 УНК-01.02 Демонстрирует владение профессиональной культурой	• ПК-19.7 УНК-01.02 У-4. Способен демонстрировать аккуратность, пунктуальность и ответственность в выполнении своих обязанностей

2. Показатели и критерии оценивания компетенций, шкалы оценивания

2.1. Показатели и критерии оценивания компетенций

№ п/п	Раздел, тема	Код индикатора компетенции	Наименование ОС	
			ТК	ПА
1	Комплексные числа	ПК-1 ППК-Р1.3	Пз	Пз

№ п/п	Раздел, тема	Код индикатора компетенции	Наименование ОС	
			ТК	ПА
2	Предел, непрерывность, ряды	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-19 .7, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-19 .3	Тест, Пз, Проект	Тест, Пз
3	Производные и интегралы	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7	Пз, Проект	Пз
4	Обыкновенные дифференциальные уравнения первого порядка	ПК-1 ППК-Р1.4, ПК-19 .7, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.5	Тест, Пз	Тест, Пз
5	Обыкновенные дифференциальные уравнения второго порядка	ПК-19 .7, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5	Тест, Пз	Тест, Пз
6	Системы дифференциальных уравнений	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7	Проект, Пз	Пз
7	Численные методы решения дифференциальных уравнений	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5	Пз	Пз
8	Матрицы	ПК-1 ППК-Р1.4, ПК-19 .7, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.5	Тест, Пз, Проект	Тест, Пз
9	Системы линейных уравнений	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-19 .7	Тест, Пз	Тест, Пз
10	Линейные операторы	ПК-1 ППК-Р1.4, ПК-19 .7, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.5	Тест, Пз	Тест, Пз

2.2. Критерии оценивания результатов обучения для текущего контроля успеваемости и промежуточной аттестации

Оценочное средство	Критерии оценивания	Шкала оценивания
Практическое задание	Задание выполнено верно. Выбран оптимальный путь решения. Присутствует развернутое описание алгоритма решения	Отлично
	Задание выполнено верно. Допущены негрубые логические ошибки при описании алгоритма решения. Отсутствуют пояснения к решению задания	Хорошо
	Ход решения задания верный, но допущены ошибки приведшие к неправильному ответу	Удовлетворительно
	В работе получен неверный ответ, связанный с грубыми ошибками допущенными в ходе решения, либо решение отсутствует полностью	Неудовлетворительно

Оценочное средство	Критерии оценивания	Шкала оценивания
Тест	Студентом даны правильные ответы на 91-100% заданий	Отлично
	Студентом даны правильные ответы на 81-90% заданий	Хорошо
	Студентом даны правильные ответы на 71-80% заданий	Удовлетворительно
	Студентом даны правильные ответы менее чем на 70% заданий	Неудовлетворительно
Проект	Проект студентом завершён в полном объеме. Для естественнонаучного проекта представлена работоспособная практическая часть, правильно выполнены и обоснованы необходимые расчёты. Реализация практической части проекта соответствует техническому заданию или заданию преподавателя. В проекте обозначена актуальность выбора темы с опорой на анализ предметной области. Студент способен сформулировать и обосновать практическую значимость своей работы. Подготовлена презентация результатов работы. Студент ориентируется во всех этапах разработки проекта, уверенно отвечает на вопросы аудитории. Способен аргументированно обосновать концепцию проекта и выбор инструментов для разработки проекта. Студент способен выделить достоинства и недостатки своей работы и предложить способы устранения недостатков	Отлично
	Проект студентом в целом завершён. Выполнены ключевые задачи. Для естественнонаучного проекта практическая часть в целом работоспособна, но есть мелкие неустранённые недостатки, необходимые расчеты в целом выполнены верно, но есть небольшие замечания. Реализация проекта в целом соответствует техническому заданию или заданию преподавателя. В проекте обозначена, но недостаточно обоснована актуальность темы. Практическая значимость работы просматривается, студент в целом может её сформулировать. Подготовлена презентация результатов работы. Студент ориентируется в этапах разработки проекта, но отвечает не на все вопросы аудитории. Способен обосновать выбор инструментов для реализации проекта. Студент способен выделить достоинства и недостатки своей работы, но не может предложить способы устранения последних	Хорошо
	Проект студентом завершён только в основных пунктах задания. Выполнены только ключевые задачи с недостатками. Для естественнонаучного проекта практическая часть работоспособна не вполне, есть существенные неустранённые недостатки, необходимые расчеты выполнены с ошибками. Реализация проекта частично соответствует техническому заданию или заданию преподавателя. В проекте обозначена, но не обоснована актуальность темы. Практическая значимость работы сформулирована слабо. Подготовлена презентация результатов работы. Студент слабо ориентируется в этапах разработки проекта, отвечает только на некоторые вопросы аудитории. Плохо обосновывает выбор инструментов для реализации проекта. Студент способен выделить достоинства и недостатки своей работы после серии наводящих вопросов, но не может предложить способы устранения недостатков	Удовлетворительно
	Проект студентом не завершён. Неполностью выполнены или не выполнены совсем ключевые задачи. Для естественнонаучного проекта практическая часть не работоспособна или не начата, есть существенные неустранённые недостатки, необходимые расчеты выполнены с грубыми ошибками. Реализация проекта не соответствует техническому заданию или заданию преподавателя. В проекте необозначена, и не обоснована актуальность темы. Практическая значимость работы не сформулирована. Плохо подготовлена презентация результатов работы. Студент почти не ориентируется в этапах разработки проекта, не отвечает на вопросы аудитории. Плохо обосновывается выбор инструментов для реализации проекта. Студент не способен выделить достоинства и недостатки своей работы даже после серии наводящих вопросов	Неудовлетворительно

2.3. Оценочные средства для текущего контроля (примеры)

2.3.1. Материалы для компьютерного тестирования обучающихся

Общие критерии оценивания

Процент правильных ответов	Оценка
91% – 100%	5 (отлично)
81% – 90%	4 (хорошо)
71% – 80%	3 (удовлетворительно)
Менее 70%	2 (неудовлетворительно)

Соответствие вопросов теста индикаторам формируемых и оцениваемых компетенций

№ вопроса в тесте	Код индикатора компетенции
1	ПК-1 ППК-Р1.4
2	ПК-1 ППК-Р1.4, ПК-19 .7
3	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-19 .7
4	ПК-1 ППК-Р1.4
5	ПК-1 ППК-Р1.4, ПК-19 .7
6	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-19 .7
7	ПК-19 .7
8	ПК-1 ППК-Р1.4, ПК-19 .7
9	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5
10	ПК-19 .7
11	ПК-1 ППК-Р1.4, ПК-19 .7
12	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-19 .7
13	ПК-19 .7
14	ПК-1 ППК-Р1.4
15	ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-19 .7
16	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-19 .7
17	ПК-19 .7
18	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5
19	ПК-19 .7
20	ПК-19 .7
21	ПК-1 ППК-Р1.4, ПК-19 .7

Ключ ответов

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
1	b
2	b, c, d
3	e
4	1a, 2b
5	транспонирует, Транспонирует, Транспонирует., транспонирует.

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
6	a
7	d
8	a, b
9	1a, 2b
10	2
11	f
12	a, b
13	d
14	c
15	b
16	a
17	a
18	c
19	b
20	маклорена, Маклорена, МакЛорена, Мак Лорена, мак Лорена
21	a, c, d

Перечень тестовых вопросов

№ 1. Задание с единичным выбором. Выберите один правильный ответ.

Команда python `E = pr.eye(3)` создает...

- a. матрицу третьего порядка из всех единиц
- b. матрицу третьего порядка из нулей с единицами по главной диагонали
- c. матрицу третьего порядка из всех нулей

№ 2. Задание с множественным выбором. Выберите 3 правильных ответа.

Какие библиотеки нужно подключить на Python, чтобы решать системы линейных уравнений?

- a. matplotlib
- b. numpy
- c. scipy
- d. sympy

№ 3. Задание с единичным выбором. Выберите один правильный ответ.

Какие библиотеки нужно подключить на Python, чтобы явно вычислять производные?

- a. Math
- b. Scikit-learn
- c. Mathplotlib
- d. NumPy
- e. Sympy
- f. SciPy

№ 4. Задание на соответствие. Соотнесите элементы двух списков.

Соотнесите какой библиотеке python соответствует данная функция

1. A.eigenvals() a. sympy
2. linalg.eig(a) b. numpy

№ 5. Задание открытой формы. Введите ответ.

Что сделает команда Python $B1 = B.T$ с матрицей B ? (одно слово)

№ 6. Задание с единичным выбором. Выберите один правильный ответ.

Функция `dsolve(eq, func, hint='separable')` на Python лучше решает дифференциальные уравнения какого типа?

- a. с разделяющимися переменными
b. линейные
c. Бернулли
d. однородные

№ 7. Задание с единичным выбором. Выберите один правильный ответ.

Радиус сходимости степенного ряда находится как предел на бесконечности

- a. отношения коэффициента (n+1)-го члена к коэффициенту n-го
b. модуля отношения коэффициента (n+1)-го члена к коэффициенту n-го
c. отношения n-го члена к (n+1)-му
d. модуля отношения коэффициента n-го члена к коэффициенту (n+1)-го
e. отношения коэффициента n-го члена к коэффициенту (n+1)-го
f. отношения (n+1)-го члена к n-му

№ 8. Задание с множественным выбором. Выберите 2 правильных ответа.

Если на python нужно вычислить левосторонний предел, то четвертым аргументом в функции `limit` нужно поставить

- a. '-'
b. `dir='-'`
c. `dir='left'`
d. `\left`

№ 9. Задание на соответствие. Соотнесите элементы двух списков.

Соотнесите какой библиотеке python соответствует данная функция

1. `linsolve()` a. sympy
2. `linalg.solve()` b. numpy

№ 10. Задание открытой формы. Введите ответ.

Сколько среди перечисленных задач являются правильной постановкой «задачи Коши»? Укажите число.

- $yy'' = (y')^2, y(0) = 1$
○ $y'' = e^{-2x}, y(0) = 1, y(2) = e^{-4}$
○ $e^{-y} dx + (1 - xe^{-y}) dy = 0$
○ $(y'')^2 + (y')^2 = 1, y(0) = 1, y'(0) = 2$

№ 11. Задание с единичным выбором. Выберите один правильный ответ.

Какие библиотеки нужно подключить на Python, чтобы решать дифференциальные уравнения?

- a. Scikit-learn

- b. Mathplotlib
- c. SciPy
- d. Math
- e. NumPy
- f. Sympy

№ 12. Задание с множественным выбором. Выберите 2 правильных ответа.

Какие библиотеки нужно подключить на Python, чтобы работать с числовыми матрицами?

- a. numpy
- b. sympy
- c. math
- d. scipy

№ 13. Задание с единичным выбором. Выберите один правильный ответ.

У функции $y = (1 + 5x)/(3 + x)$ какая вертикальная асимптота?

- a. $x = 1/5$
- b. $x = -1/5$
- c. $y = 5$
- d. $x = -3$
- e. $x = 3$

№ 14. Задание с единичным выбором. Выберите один правильный ответ.

Какая команда python правильно выполнит умножение матриц с подключением библиотеки SymPy?

- a. $A@B$
- b. `numpy.dot(A, B)`
- c. $A*B$
- d. `numpy.matmul(A,B)`

№ 15. Задание с единичным выбором. Выберите один правильный ответ.

Если на python вычисляется предел на бесконечности, то третьим аргументом в функции limit нужно поставить

- a. $\backslash inf$
- b. oo
- c. 0
- d. $\backslash infty$

№ 16. Задание с единичным выбором. Выберите один правильный ответ.

Если $\lim_{x \rightarrow 0} (\frac{abs((x-2)*(x-7))}{(x^3-9x^2+14x)})$ равен бесконечности и $\lim_{x \rightarrow 0} (\frac{abs((x-2)*(x-7))}{(x^3-9x^2+14x)})$ равен бесконечности, то у функции

- a. в точке 0 разрыв II рода
- b. в точке 2 устранимый разрыв I рода
- c. в точке 0 неустраняемый разрыв I рода
- d. в точке 2 разрыв II рода
- e. в точке 0 устранимый разрыв I рода

f. в точке 2 неустраняемый разрыв I рода

№ 17. Задание с единичным выбором. Выберите один правильный ответ.

Если радиус сходимости степенного ряда равен бесконечности, значит

- a. ряд сходится при всех x
- b. ряд сходится безусловно
- c. ряд расходится
- d. ряд сходится условно

№ 18. Задание с единичным выбором. Выберите один правильный ответ.

Какую функцию на python лучше использовать для аналитического решения дифференциального уравнения второго порядка?

- a. Функция `Cauchy_k()`.
- b. функция `odeint` (из `scipy.integrate`)
- c. Функция `dsolve` библиотеки `SymPy`.

№ 19. Задание с единичным выбором. Выберите один правильный ответ.

Наклонные асимптоты имеют вид

- a. $y=b$
- b. $y = kx + b$
- c. $x=a$

№ 20. Задание открытой формы. Введите ответ.

Данный код на Python `import sympy x = sympy.symbols('x') func = cos(x) x0=0 print((func).series(x, x0, 10))` выдаст разложение функции косинуса в какой ряд (фамилия)?

№ 21. Задание с множественным выбором. Выберите 3 правильных ответа.

Какие библиотеки нужно подключить на Python, чтобы работать с матрицами линейных операторов?

- a. `SciPy`
- b. `matplotlib`
- c. `sympy`
- d. `numpy`

2.3.2. Практические задания для оценки компетенции «ПК-1 ППК-Р1.3»

№ 1. Комплексные числа.

На Python найти все значения $\sqrt{1+i}$.

№ 2. Пределы.

На Python вычислить предел: $\lim_{x \rightarrow -\infty} (-x^2 - 3x - 7) * e^{2x}$.

№ 3. Исследование функции.

На Python найдите асимптоты графика функции:
$$-\frac{2x^2 - 5x - 4}{x - 1}$$

№ 4. Производная.

На Python построить график функции $f(x)$ с указанием точек экстремума, точек

$$f(x) = \frac{x^2 - 16}{5(x + 5)}$$

перегиба и асимптот:

№ 5. *Функции многих переменных.*

С помощью Python провести касательную плоскость и нормаль к поверхности:

$$z = \sqrt{x^2 + y^2} - xy$$

в точке M(3;4; -7).

№ 6. *Функции многих переменных.*

$$z = x^3 + xy + y^2.$$

На Python найти экстремумы функции Нарисовать функцию и убедиться, что экстремум найден верно.

№ 7. *Интегралы.*

$$y = 2x, y = -x^2 + 7x - 6.$$

На Python найти площадь фигуры, ограниченной линиями Нарисовать график.

№ 8. *Дифференциальные уравнения первого порядка.*

Найти решение дифференциального уравнения $x dy = (x^4 - y) dx$, удовлетворяющее начальному условию $y(1) = 1$.

№ 9. *Дифференциальные уравнения второго порядка.*

$$xy'' - (x+1)y' - 2(x-1)y = x^3 e^x,$$

На Python найти общее решение уравнения если известно одно частное решение соответствующего однородного уравнения

$$e^{-x^2}$$

№ 10. *Системы дифференциальных уравнений.*

На Python решить задачу Коши: $dx/dt = 2x - 5y + 3$, $dy/dt = 5x - 6y + 1$, $x(0) = 6, y(0) = 5$.

№ 11. *Численное решение дифференциальных уравнений.*

На Python найти численное решение задачи Коши системы уравнений $y_1' = xy_1 + y_2, y_2' = y_1 - y_2, y_1(0) = 1, y_2(0) = 0$ на отрезке $[0; 2]$.

№ 12. *Матрицы.*

$$A^{10}, \quad A = \begin{pmatrix} 1 & 1 \\ -1 & 3 \end{pmatrix}.$$

Вычислить на Python если

№ 13. *Матрицы.*

С помощью Python представить вектор $f = (46, 46, 92, 0)$ как линейную комбинацию векторов $a = (0, 1, 1, 3)$, $b = (0, 2, 3, 0)$, $c = (2, 1, 1, 1)$, $d = (0, 3, 0, 4)$.

№ 14. *Решение систем линейных уравнений.*

На Python найти общее и базисное решения системы линейных уравнений

$$\begin{cases} x - 2y + 2z = -3, \\ 3x - y + 3z = 1, \\ 2x + y + z = 4. \end{cases}$$

№ 15. *Линейные операторы.*

$$A = \begin{pmatrix} 1 & -1 \\ -6 & 2 \end{pmatrix}$$

Дана матрица . с помощью Python определите, какой из векторов $x_1 =$

(1,2), $x_2 = (0,3)$, $x_3 = (0,0)$ является собственным для матрицы A?

2.3.3. Практические задания для оценки компетенции «ПК-1 ППК-Р1.4»

№ 16. Пределы.

$$\lim_{x \rightarrow -\infty} (-x^2 - 3x - 7) * e^{2x}.$$

На Python вычислить предел:

№ 17. Исследование функции.

$$-\frac{2x^2 - 5x - 4}{x - 1}$$

На Python найдите асимптоты графика функции:

№ 18. Производная.

На Python построить график функции $f(x)$ с указанием точек экстремума, точек

$$f(x) = \frac{x^2 - 16}{5(x + 5)}$$

перегиба и асимптот:

№ 19. Функции многих переменных.

С помощью Python провести касательную плоскость и нормаль к поверхности:

$$z = \sqrt{x^2 + y^2} - xy$$

в точке $M(3;4; -7)$.

№ 20. Функции многих переменных.

$$z = x^3 + xy + y^2.$$

На Python найти экстремумы функции Нарисовать функцию и убедиться, что экстремум найден верно.

№ 21. Интегралы.

На Python найти площадь фигуры, ограниченной линиями $y = 2x$, $y = -x^2 + 7x - 6$.

Нарисовать график.

№ 22. Дифференциальные уравнения первого порядка.

Найти решение дифференциального уравнения $x dy = (x^4 - y) dx$, удовлетворяющее начальному условию $y(1) = 1$.

№ 23. Дифференциальные уравнения второго порядка.

$$xy'' - (x + 1)y' - 2(x - 1)y = x^3 e^x,$$

На Python найти общее решение уравнения если известно одно частное решение соответствующего однородного уравнения

$$e^{-x^2}$$

№ 24. Системы дифференциальных уравнений.

На Python решить задачу Коши: $dx/dt = 2x - 5y + 3$, $dy/dt = 5x - 6y + 1$, $x(0) = 6$, $y(0) = 5$.

№ 25. Численное решение дифференциальных уравнений.

На Python найти численное решение задачи Коши системы уравнений $y_1' = xy_1 + y_2$, $y_2' = y_1 - y_2$, $y_1(0) = 1$, $y_2(0) = 0$ на отрезке $[0; 2]$.

№ 26. Матрицы.

$$A^{10}, \quad \text{если} \quad A = \begin{pmatrix} 1 & 1 \\ -1 & 3 \end{pmatrix}.$$

Вычислить на Python

№ 27. Матрицы.

С помощью Python представить вектор $f = (46, 46, 92, 0)$ как линейную комбинацию векторов $a = (0, 1, 1, 3)$, $b = (0, 2, 3, 0)$, $c = (2, 1, 1, 1)$, $d = (0, 3, 0, 4)$.

№ 28. Решение систем линейных уравнений.

На Python найти общее и базисное решения системы линейных уравнений

$$\begin{cases} x - 2y + 2z = -3, \\ 3x - y + 3z = 1, \\ 2x + y + z = 4. \end{cases}$$

№ 29. Линейные операторы.

$$A = \begin{pmatrix} 1 & -1 \\ -6 & 2 \end{pmatrix}$$

Дана матрица A . с помощью Python определите, какой из векторов $x_1 = (1, 2)$, $x_2 = (0, 3)$, $x_3 = (0, 0)$ является собственным для матрицы A ?

2.3.4. Практические задания для оценки компетенции «ПК-1 ППК-Р1.5»

№ 30. Пределы.

$$\lim_{x \rightarrow -\infty} (-x^2 - 3x - 7) * e^{2x}.$$

На Python вычислить предел:

№ 31. Исследование функции.

$$-\frac{2x^2 - 5x - 4}{x - 1}$$

На Python найдите асимптоты графика функции:

№ 32. Производная.

На Python построить график функции $f(x)$ с указанием точек экстремума, точек

$$f(x) = \frac{x^2 - 16}{5(x + 5)}$$

перегиба и асимптот:

№ 33. Функции многих переменных.

С помощью Python провести касательную плоскость и нормаль к поверхности:

$$z = \sqrt{x^2 + y^2} - xy$$

в точке $M(3; 4; -7)$.

№ 34. Функции многих переменных.

На Python найти экстремумы функции $z = x^3 + xy + y^2$. Нарисовать функцию и убедиться, что экстремум найден верно.

№ 35. Интегралы.

На Python найти площадь фигуры, ограниченной линиями $y = 2x$, $y = -x^2 + 7x - 6$. Нарисовать график.

№ 36. Дифференциальные уравнения первого порядка.

Найти решение дифференциального уравнения $x dy = (x^4 - y) dx$, удовлетворяющее начальному условию $y(1) = 1$.

№ 37. Дифференциальные уравнения второго порядка.

На Python найти общее решение уравнения $xy'' - (x + 1)y' - 2(x - 1)y = x^3 e^x$, если известно одно частное решение соответствующего однородного уравнения e^{-x^2}

№ 38. Системы дифференциальных уравнений.

На Python решить задачу Коши: $dx/dt = 2x - 5y + 3$, $dy/dt = 5x - 6y + 1$, $x(0) = 6, y(0) = 5$.

№ 39. Численное решение дифференциальных уравнений.

На Python найти численное решение задачи Коши системы уравнений $y_1' = xy_1 + y_2$, $y_2' = y_1 - y_2$, $y_1(0) = 1, y_2(0) = 0$ на отрезке $[0; 2]$.

№ 40. Матрицы.

Вычислить на Python A^{10} , если $A = \begin{pmatrix} 1 & 1 \\ -1 & 3 \end{pmatrix}$.

№ 41. Матрицы.

С помощью Python представить вектор $f = (46, 46, 92, 0)$ как линейную комбинацию векторов $a = (0, 1, 1, 3)$, $b = (0, 2, 3, 0)$, $c = (2, 1, 1, 1)$, $d = (0, 3, 0, 4)$.

№ 42. Решение систем линейных уравнений.

На Python найти общее и базисное решения системы линейных уравнений
$$\begin{cases} x - 2y + 2z = -3, \\ 3x - y + 3z = 1, \\ 2x + y + z = 4. \end{cases}$$

№ 43. Линейные операторы.

Дана матрица $A = \begin{pmatrix} 1 & -1 \\ -6 & 2 \end{pmatrix}$. с помощью Python определите, какой из векторов $x_1 = (1, 2)$, $x_2 = (0, 3)$, $x_3 = (0, 0)$ является собственным для матрицы A ?

2.3.5. Практические задания для оценки компетенции «ПК-19 .3»

№ 44. ряды.

$$\sum_{n=2}^{\infty} \frac{(x-4)^n}{\sqrt{3n-4}}$$

На Python найдите радиус и интервал сходимости ряда:

2.3.6. Практические задания для оценки компетенции «ПК-19 .4»

№ 45. ряды.

$$\sum_{n=2}^{\infty} \frac{(x-4)^n}{\sqrt{3n-4}}$$

На Python найдите радиус и интервал сходимости ряда:

2.3.7. Практические задания для оценки компетенции «ПК-19 .5»

№ 46. ряды.

$$\sum_{n=2}^{\infty} \frac{(x-4)^n}{\sqrt{3n-4}}$$

На Python найдите радиус и интервал сходимости ряда:

2.3.8. Практические задания для оценки компетенции «ПК-1 ППК-Р1.6»

№ 47. Функции многих переменных.

На Python найти экстремумы функции $z = x^3 + xy + y^2$. Нарисовать функцию и убедиться, что экстремум найден верно.

№ 48. Интегралы.

На Python найти площадь фигуры, ограниченной линиями $y = 2x, y = -x^2 + 7x - 6$.
Нарисовать график.

2.3.9. Практические задания для оценки компетенции «ПК-1 ППК-Р1.7»

№ 49. *Функции многих переменных.*

На Python найти экстремумы функции $z = x^3 + xy + y^2$. Нарисовать функцию и убедиться, что экстремум найден верно.

№ 50. *Интегралы.*

На Python найти площадь фигуры, ограниченной линиями $y = 2x, y = -x^2 + 7x - 6$.
Нарисовать график.

2.3.10. Практические задания для оценки компетенции «ПК-1 ППК-Р1.1»

№ 51. *Численное решение дифференциальных уравнений.*

На Python найти численное решение задачи Коши системы уравнений $y_1' = xy_1 + y_2, y_2' = y_1 - y_2, y_1(0) = 1, y_2(0) = 0$ на отрезке $[0; 2]$.

2.3.11. Практические задания для оценки компетенции «ПК-1 ППК-Р1.2»

№ 52. *Численное решение дифференциальных уравнений.*

На Python найти численное решение задачи Коши системы уравнений $y_1' = xy_1 + y_2, y_2' = y_1 - y_2, y_1(0) = 1, y_2(0) = 0$ на отрезке $[0; 2]$.

2.3.12. Проекты для оценки компетенции «ПК-1 ППК-Р1.1»

№ 53. *Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.*

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском. Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{\text{linear}} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{\text{linear}} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости. Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 54. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.

3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для

создания практически полезного продукта.

2.3.13. Проекты для оценки компетенции «ПК-1 ППК-Р1.2»

№ 55. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском. Градиентный спуск с контролем сходимости по изменению функции потерь. `tol` — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости. Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последо-

вательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 56. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):
# Функция для генерации рекомендаций на основе схожести пользователей
pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):
# Функция для отображения рекомендаций пользователю через интерфейс
pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.14. Проекты для оценки компетенции «ПК-1 ППК-Р1.3»

№ 57. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последо-

вательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском """ Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). """ Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 58. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.15. Проекты для оценки компетенции «ПК-1 ППК-Р1.4»

№ 59. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском. Градиентный спуск с контролем сходимости по изменению функции потерь. `tol` — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости. Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь —

это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютона, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 60. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.16. Проекты для оценки компетенции «ПК-1 ППК-Р1.5»

№ 61. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском. Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{\text{linear}} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{\text{linear}} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости. Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 62. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.17. Проекты для оценки компетенции «ПК-1 ППК-Р1.6»

№ 63. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском """ Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). """ Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 64. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.18. Проекты для оценки компетенции «ПК-1 ППК-Р1.7»

№ 65. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое

изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском. Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{\text{linear}} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{\text{linear}} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости. Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 66. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

— Историю просмотров пользователя

- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.19. Проекты для оценки компетенции «ПК-19 .1»

№ 67. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском """ Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). """ Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 68. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):
    # Функция для вычисления косинусного сходства между двумя
    # пользователями
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):
    # Функция для генерации рекомендаций на основе схожести пользователей
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):
    # Функция для отображения рекомендаций пользователю через интерфейс
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.20. Проекты для оценки компетенции «ПК-19 .2»

№ 69. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}$ $k=0 \rightarrow \infty$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском `"""` Градиентный спуск с контролем сходимости по изменению функции потерь. `tol` — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). `"""` Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 70. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет

предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.21. Проекты для оценки компетенции «ПК-19 .3»

№ 71. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском """ Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). """ Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной

сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 72. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.22. Проекты для оценки компетенции «ПК-19 .4»

№ 73. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}$ $k=0\infty$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная,

квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском. """ Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). """ Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости. Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 74. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.23. Проекты для оценки компетенции «ПК-19 .5»

№ 75. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском """ Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). """ Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность

потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{\text{linear}} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{\text{linear}} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 76. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов
```

```
pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.24. Проекты для оценки компетенции «ПК-19 .6»

№ 77. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}$ $k=0\infty$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском """ Градиентный спуск с контролем сходимости по изменению функции потерь. tol — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). """ Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 78. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    # пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.

3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

2.3.25. Проекты для оценки компетенции «ПК-19 .7»

№ 79. Кейс: контроль сходимости градиентного спуска при обучении линейной регрессии.

Контекст. При обучении модели линейной регрессии методом градиентного спуска мы итеративно обновляем веса по правилу:

$$w_{k+1} = w_k - \eta \nabla L(w_k),$$

где $L(w)$ — функция потерь (например, MSE), η — шаг (learning rate). Последовательность значений функции потерь $\{L(w_k)\}_{k=0}^{\infty}$ образует числовой ряд (точнее, последовательность), и нас интересует, сходится ли она к некоторому пределу (локальному минимуму) и насколько быстро.

В терминах исследования рядов это означает:

Сходимость: $L(w_k) \rightarrow L^*$ при $k \rightarrow \infty$.

Скорость сходимости: насколько быстро убывает разность $|L(w_k) - L^*|$ (линейная, квадратичная и т.п.).

Критерий остановки: когда можно считать, что сходимость достигнута (ряд «почти сошёлся»). На практике нельзя ждать бесконечного числа итераций, поэтому используют критерии остановки, основанные на поведении «хвоста» последовательности: малое изменение функции потерь или градиента между соседними шагами. Это и есть применение идей сходимости рядов.

Проблема без контроля сходимости. Если learning rate слишком большой, последовательность $L(w_k)$ может расходиться или осциллировать (не сходиться). Если слишком маленький — обучение будет крайне медленным. Без формального критерия остановки алгоритм либо недообучится, либо будет тратить лишние ресурсы.

Решение. Реализовать мониторинг сходимости по значениям функции потерь и градиентов, а также адаптивную стратегию изменения learning rate на основе поведения «хвоста» ряда.

Задание: Практическая реализация на Python. Командная работа.

1. Базовая линейная регрессия с градиентным спуском. Градиентный спуск с контролем сходимости по изменению функции потерь. `tol` — порог для критерия остановки (насколько «сошёлся» ряд значений потерь). Критерий остановки: если изменение потерь очень мало, считаем, что ряд сошёлся

2. Анализ сходимости: проверка скорости убывания «хвоста». Чтобы понять, насколько быстро сходится последовательность потерь, можно исследовать, как убывает

разность между соседними значениями. Здесь в качестве ошибки используем разность потерь относительно последнего значения (не забудьте про защиту от деления на ноль). Оценка коэффициента линейной сходимости: Если $C_{linear} < 1$, это говорит о линейной сходимости; чем ближе к 0, тем быстрее. Если $C_{linear} \geq 1$, сходимость может отсутствовать или быть очень медленной.

3. Адаптивный подбор learning rate на основе сходимости Идея: если «ряд» значений потерь не убывает достаточно быстро (или даже растёт), уменьшаем learning rate. Если потери не уменьшаются — ряд «расходится» или осциллирует

Связь с теорией рядов и практическая польза. Исследование сходимости здесь — это анализ поведения последовательности значений функции потерь: сходится ли она, насколько быстро, и когда можно остановиться. Критерий Коши (фундаментальность последовательности) на практике заменяется на проверку малого изменения между соседними членами: $|L_k - L_{k-1}| < \epsilon$. Оценка скорости сходимости позволяет сравнивать разные методы оптимизации и подбирать гиперпараметры. Адаптивные стратегии (как в примере выше) — это прямое применение результатов анализа сходимости для улучшения алгоритма. Где ещё это применимо в ML и разработке ПО? Стохастический градиентный спуск (SGD) — анализ сходимости по эпохам и батчам, контроль осцилляций. Методы второго порядка (Ньютон, BFGS) — исследование квадратичной сходимости вблизи минимума. Регуляризация и ранняя остановка — критерий остановки по сходимости валидационной ошибки. Итерационные алгоритмы в численных методах (решение СЛАУ, собственные значения) — контроль сходимости итерационных процессов. Разработка библиотек ML — встроенные механизмы мониторинга сходимости, логирование, автоматические предупреждения о проблемах. Такой подход делает обучение моделей более предсказуемым, устойчивым и ресурсоэффективным, а код — более «математически обоснованным».

№ 80. Кейс: Разработка системы рекомендаций для онлайн-кинотеатра.

Описание задачи:

Ваша команда разрабатывает систему рекомендаций для онлайн-кинотеатра, которая будет предлагать пользователям фильмы и сериалы на основе их предпочтений. Система должна учитывать:

- Историю просмотров пользователя
- Рейтинги, поставленные просмотренным фильмам
- Жанры, которые пользователь предпочитает

Технические задачи:

1. Сбор и предобработка данных о пользователях и фильмах.
2. Применение методов линейной алгебры для нахождения схожести пользователей.
3. Разработка модели машинного обучения для рекомендаций.
4. Создание пользовательского интерфейса для отображения рекомендаций.

Функции на Python:

1. Сбор данных:

```
def fetch_user_data(user_id):  
    # Функция для получения данных о пользователе из базы данных  
    pass
```

Скопировать

2. Предобработка данных:

```
def preprocess_data(data):  
    # Функция для очистки данных, создания матрицы пользователей и фильмов  
    pass
```

Скопировать

3. Вычисление схожести:

```
def calculate_similarity(user1, user2, data_matrix):  
    # Функция для вычисления косинусного сходства между двумя  
    пользователями  
    pass
```

Скопировать

4. Модель рекомендаций:

```
def recommend_movies(user, user_matrix, movie_matrix):  
    # Функция для генерации рекомендаций на основе схожести пользователей  
    pass
```

Скопировать

5. Отображение рекомендаций:

```
def display_recommendations(user_id, recommended_movies):  
    # Функция для отображения рекомендаций пользователю через интерфейс  
    pass
```

Скопировать

Этапы выполнения:

1. Команда собирает данные о пользователях и фильмах.
2. Данные очищаются и преобразуются в формат, пригодный для анализа.
3. С помощью функции

```
calculate_similarity
```

вычисляется схожесть пользователей.

4. Модель рекомендаций

```
recommend_movies
```

генерирует список рекомендованных фильмов.

5. Интерфейс отображает рекомендации пользователю.

Этот кейс позволяет объединить знания линейной алгебры, машинного обучения и разработки программного обеспечения, а также применить навыки работы с Python для создания практически полезного продукта.

3. Промежуточная аттестация

3.1. Методические материалы, определяющие процедуру оценивания знаний, умений, навыков и опыта деятельности

Зачет является заключительным этапом процесса формирования компетенций обучающегося при изучении дисциплины и имеет целью проверку и оценку знаний обучающегося по теории, и применению полученных знаний, умений и навыков при решении практических задач.

Зачет проводится по расписанию, сформированному учебно-методическим управлением, в сроки, предусмотренные календарным учебным графиком. Зачет принимается

преподавателем, ведущим лекционные занятия.

Зачет проводится только при предъявлении обучающимся зачетной книжки и при условии выполнения всех контрольных мероприятий, предусмотренных учебным планом и рабочей программой дисциплины. Обучающимся на зачете представляется право выбрать один из билетов. Время подготовки к ответу составляет 30 минут. По истечении установленного времени обучающийся должен ответить на вопросы билета. Результаты зачета оцениваются по четырехбалльной системе и заносятся в зачетно-экзаменационную ведомость и зачетную книжку. В зачетную книжку заносятся только положительные оценки. Подписанный преподавателем экземпляр ведомости сдается не позднее следующего дня в деканат.

В случае неявки обучающегося на зачет в зачетно-экзаменационную ведомость делается отметка «не явка». Обучающиеся, не прошедшие промежуточную аттестацию по дисциплине, должны ликвидировать академическую задолженность в установленном локальными нормативными актами порядке.

3.2. Вопросы к зачету

№	Вопрос	Код компетенции
1.	Использование библиотеки SymPy для символьных вычислений для решения задач математического анализа	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-19 .1, ПК-19 .2, ПК-19 .3, ПК-19 .4, ПК-19 .5, ПК-19 .6, ПК-19 .7
2.	Функция dsolve (из библиотеки SymPy) для решения дифференциального уравнения первого порядка в Python	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
3.	Функция scipy.integrate.solve_ivp для решения дифференциального уравнения первого порядка в Python	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
4.	Функция scipy.integrate.odeint для решения дифференциального уравнения первого порядка в Python	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
5.	Функция scipy.integrate.solve_ivp для решения задачи Коши (задачи с начальными условиями) для дифференциального уравнения второго порядка в Python	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
6.	Функция odeint (из scipy.integrate) для решения дифференциального уравнения второго порядка в Python	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
7.	Функция dsolve (из библиотеки SymPy) для решения дифференциального уравнения второго порядка в Python	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7

№	Вопрос	Код компетенции
8.	Использование функции <code>scipy.integrate.solve_ivp</code> для решения систем дифференциальных уравнений	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
9.	Использование функции <code>scipy.integrate.solve_bvp</code> для решения систем дифференциальных уравнений	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
10.	Использование функции <code>odeint</code> библиотеки SymPy для решения систем дифференциальных уравнений	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
11.	Функции <code>ode()</code> и <code>odeint()</code> модуля <code>scipy.integrate</code> для численного решения уравнений и систем обыкновенных дифференциальных уравнений первого порядка.	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
12.	Использование библиотеки Mathplotlib для создания двумерной графики при численном решении дифференциальных уравнений.	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
13.	Использование библиотеки NumPy для работы с матрицами.	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
14.	Использование библиотеки SymPy для работы с матрицами.	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
15.	Использование библиотеки NumPy для решения систем линейных уравнений.	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
16.	Использование библиотеки SymPy для решения систем линейных уравнений.	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
17.	Использование библиотеки SciPy для решения систем линейных уравнений.	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7

№	Вопрос	Код компетенции
18.	Использование библиотеки NumPy для работы с матрицами линейных операторов.	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
19.	Использование библиотеки SymPy для работы с матрицами линейных операторов.	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7
20.	Использование библиотеки SciPy для работы с матрицами линейных операторов.	ПК-1 ППК-Р1.4, ПК-19 .7

3.3. Тематика курсовых работ

По данной дисциплине выполнение курсовых проектов (работ) не предусматривается.

3.4. Материалы для компьютерного тестирования обучающихся

Общие критерии оценивания

Процент правильных ответов	Оценка
91% – 100%	5 (отлично)
81% – 90%	4 (хорошо)
71% – 80%	3 (удовлетворительно)
Менее 70%	2 (неудовлетворительно)

Соответствие вопросов теста индикаторам формируемых и оцениваемых компетенций

№ вопроса в тесте	Код индикатора компетенции
1	ПК-1 ППК-Р1.4
2	ПК-1 ППК-Р1.4, ПК-19 .7
3	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-19 .7
4	ПК-1 ППК-Р1.4
5	ПК-1 ППК-Р1.4, ПК-19 .7
6	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-19 .7
7	ПК-19 .7
8	ПК-1 ППК-Р1.4, ПК-19 .7
9	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5
10	ПК-19 .7
11	ПК-1 ППК-Р1.4, ПК-19 .7
12	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-19 .7
13	ПК-19 .7
14	ПК-1 ППК-Р1.4
15	ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-19 .7
16	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-19 .7
17	ПК-19 .7

№ вопроса в тесте	Код индикатора компетенции
18	ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5
19	ПК-19 .7
20	ПК-19 .7
21	ПК-1 ППК-Р1.4, ПК-19 .7

Ключ ответов

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
1	b
2	b, c, d
3	e
4	1a, 2b
5	транспонирует, Транспонирует, Транспонирует., транспонирует.
6	a
7	d
8	a, b
9	1a, 2b
10	2
11	f
12	a, b
13	d
14	c
15	b
16	a
17	a
18	c
19	b
20	маклорена, Маклорена, МакЛорена, Мак Лорена, мак Лорена
21	a, c, d

Перечень тестовых вопросов

№ 1. Задание с единичным выбором. Выберите один правильный ответ.

Команда `python E = pr.eye(3)` создает...

- a. матрицу третьего порядка из всех единиц
- b. матрицу третьего порядка из нулей с единицами по главной диагонали
- c. матрицу третьего порядка из всех нулей

№ 2. Задание с множественным выбором. Выберите 3 правильных ответа.

Какие библиотеки нужно подключить на Python, чтобы решать системы линейных уравнений?

- a. `matplotlib`
- b. `numpy`

c. scipy

d. sympy

№ 3. Задание с единичным выбором. Выберите один правильный ответ.

Какие библиотеки нужно подключить на Python, чтобы явно вычислять производные?

a. Math

b. Scikit-learn

c. Mathplotlib

d. NumPy

e. Sympy

f. SciPy

№ 4. Задание на соответствие. Соотнесите элементы двух списков.

Соотнесите какой библиотеке python соответствует данная функция

1. A.eigenvals() a. sympy

2. linalg.eig(a) b. numpy

№ 5. Задание открытой формы. Введите ответ.

Что сделает команда Python $B1 = B.T$ с матрицей B ? (одно слово)

№ 6. Задание с единичным выбором. Выберите один правильный ответ.

Функция `dsolve(eq, func, hint='separable')` на Python лучше решает дифференциальные уравнения какого типа?

a. с разделяющимися переменными

b. линейные

c. Бернулли

d. однородные

№ 7. Задание с единичным выбором. Выберите один правильный ответ.

Радиус сходимости степенного ряда находится как предел на бесконечности

a. отношения коэффициента (n+1)-го члена к коэффициенту n-го

b. модуля отношения коэффициента (n+1)-го члена к коэффициенту n-го

c. отношения n-го члена к (n+1)-му

d. модуля отношения коэффициента n-го члена к коэффициенту (n+1)-го

e. отношения коэффициента n-го члена к коэффициенту (n+1)-го

f. отношения (n+1)-го члена к n-му

№ 8. Задание с множественным выбором. Выберите 2 правильных ответа.

Если на python нужно вычислить левосторонний предел, то четвертым аргументом в функции `limit` нужно поставить

a. '-'

b. `dir='-'`

c. `dir='left'`

d. `\left`

№ 9. Задание на соответствие. Соотнесите элементы двух списков.

Соотнесите какой библиотеке python соответствует данная функция

1. `linsolve()` a. sympy

2. `linalg.solve()` b. numpy

№ 10. Задание открытой формы. Введите ответ.

Сколько среди перечисленных задач являются правильной постановкой «задачи Коши»? Укажите число.

- $yy'' = (y')^2, y(0) = 1$
- $y'' = e^{-2x}, y(0) = 1, y(2) = e^{-4}$
- $e^{-y}dx + (1 - xe^{-y})dy = 0$
- $(y'')^2 + (y')^2 = 1, y(0) = 1, y'(0) = 2$

№ 11. Задание с единичным выбором. Выберите один правильный ответ.

Какие библиотеки нужно подключить на Python, чтобы решать дифференциальные уравнения?

- a. Scikit-learn
- b. Matplotlib
- c. SciPy
- d. Math
- e. NumPy
- f. SymPy

№ 12. Задание с множественным выбором. Выберите 2 правильных ответа.

Какие библиотеки нужно подключить на Python, чтобы работать с числовыми матрицами?

- a. numpy
- b. sympy
- c. math
- d. scipy

№ 13. Задание с единичным выбором. Выберите один правильный ответ.

У функции $y = (1 + 5x)/(3 + x)$ какая вертикальная асимптота?

- a. $x = 1/5$
- b. $x = -1/5$
- c. $y = 5$
- d. $x = -3$
- e. $x = 3$

№ 14. Задание с единичным выбором. Выберите один правильный ответ.

Какая команда python правильно выполнит умножение матриц с подключением библиотеки SymPy?

- a. $A @ B$
- b. `numpy.dot(A, B)`
- c. $A * B$
- d. `numpy.matmul(A, B)`

№ 15. Задание с единичным выбором. Выберите один правильный ответ.

Если на python вычисляется предел на бесконечности, то третьим аргументом в функции limit нужно поставить

- a. ∞
- b. ∞
- c. 0
- d. ∞

№ 16. Задание с единичным выбором. Выберите один правильный ответ.

Если $\lim_{x \rightarrow 0} \frac{\text{abs}((x-2)*(x-7)/(x^3-9*x^2+14*x))}{x}$ равен бесконечности и $\lim_{x \rightarrow 0} \frac{\text{abs}((x-2)*(x-7)/(x^3-9*x^2+14*x))}{x}$ равен бесконечности, то у функции

- a. в точке 0 разрыв II рода
- b. в точке 2 устранимый разрыв I рода
- c. в точке 0 неустранимый разрыв I рода
- d. в точке 2 разрыв II рода
- e. в точке 0 устранимый разрыв I рода
- f. в точке 2 неустранимый разрыв I рода

№ 17. Задание с единичным выбором. Выберите один правильный ответ.

Если радиус сходимости степенного ряда равен бесконечности, значит

- a. ряд сходится при всех x
- b. ряд сходится безусловно
- c. ряд расходится
- d. ряд сходится условно

№ 18. Задание с единичным выбором. Выберите один правильный ответ.

Какую функцию на python лучше использовать для аналитического решения дифференциального уравнения второго порядка?

- a. Функция `Cauchy_k()`.
- b. функция `odeint` (из `scipy.integrate`)
- c. Функция `dsolve` библиотеки `SymPy`.

№ 19. Задание с единичным выбором. Выберите один правильный ответ.

Наклонные асимптоты имеют вид

- a. $y=b$
- b. $y = kx + b$
- c. $x=a$

№ 20. Задание открытой формы. Введите ответ.

Данный код на Python `import sympy` `x = sympy.symbols('x')` `func = cos(x)` `x0=0` `print((func).series(x, x0, 10))` выдаст разложение функции косинуса в какой ряд (фамилия)?

№ 21. Задание с множественным выбором. Выберите 3 правильных ответа.

Какие библиотеки нужно подключить на Python, чтобы работать с матрицами линейных операторов?

- a. `SciPy`
- b. `matplotlib`
- c. `sympy`
- d. `numpy`

3.5. Практические задания для оценки компетенции «ПК-1 ПК-Р1.3»

№ 1. Комплексные числа.

На Python вычислите значение выражения и представьте результат в виде $a + bi$

$$(3-4i)(-3+5i)/(1-4i)'$$

№ 2. Пределы.

$$\lim_{x \rightarrow \infty} \frac{e^{4x}-1}{1-e^{5x}}$$

На Python вычислить предел

№ 3. Исследование функции.

$$f(x) = \frac{2+3x}{x+1}$$

На python найдите асимптоты графика функции

№ 4. ряды.

$$\sum_{n=1}^{\infty} \frac{n^2+12}{n^2-n+23}$$

На Python исследуйте ряд на сходимость:

№ 5. Производная.

$$y = x^3 - 3x^2 + x + 4.$$

С помощью Python исследовать на экстремум функцию

№ 6. Интегралы.

$$\int_0^{\infty} e^{-x} dx$$

С помощью Python исследовать на сходимость интеграл

№ 7. Дифференциальные уравнения первого порядка.

$$y' + 6y^2 = -\frac{1}{x^2}$$

На Python решить задачу Коши $y(1) = 3$.

№ 8. Дифференциальные уравнения второго порядка.

$$x^3 y'' + x^2 y' = 1.$$

На Python решить уравнение

№ 9. Системы дифференциальных уравнений.

На Python решить систему уравнений $dx/dt = 7x + 3y$, $dy/dt = 6x + 4y$.

№ 10. Численное решение дифференциальных уравнений.

На Python найти численное решение задачи Коши $y'' + 4xy' + (4x^2 + 2)y = 0$, $y(0) = y'(1) = 0$ на отрезке $[0; 2.5]$.

№ 11. Матрицы.

С помощью Python найдите четверную степень матрицы

$$\begin{pmatrix} 1 & 2 \\ 5 & 0 \end{pmatrix}.$$

№ 12. Матрицы.

Найти на Python ранг матрицы, составленной из векторов, и определить, являются ли векторы $a = (0, 1, 1, 3)$, $b = (0, 2, 3, 0)$, $c = (2, 1, 1, 1)$ и $d = (-2, 2, 3, 2)$ линейно независимыми.

№ 13. Решение систем линейных уравнений.

На Python найти общее решение однородной системы линейных уравнений

$$\begin{cases} x - 2y + z + w = 0, \\ 2x + y + 2z + 3w = 0, \\ 2x - 5y + 2z - w = 0. \end{cases}$$

выписать фундаментальную систему решений.

№ 14. *Линейные операторы.*

$$A = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}$$

С помощью Python привести матрицу к диагональному виду и указать матрицу перехода Т.

3.6. Практические задания для оценки компетенции «ПК-1 ПК-Р1.4»

№ 15. *Пределы.*

$$\lim_{x \rightarrow \infty} \frac{e^{4x} - 1}{1 - e^{5x}}$$

На Python вычислить предел

№ 16. *Исследование функции.*

$$f(x) = \frac{2+3x}{x+1}$$

На python найдите асимптоты графика функции

№ 17. *ряды.*

$$\sum_{n=1}^{\infty} \frac{n^2 + 12}{n^2 - n + 23}$$

На Python исследуйте ряд на сходимость:

№ 18. *Производная.*

$$y = x^3 - 3x^2 + x + 4.$$

С помощью Python исследовать на экстремум функцию

№ 19. *Интегралы.*

$$\int_0^{\infty} e^{-x} dx$$

С помощью Python исследовать на сходимость интеграл

№ 20. *Дифференциальные уравнения первого порядка.*

$$y' + 6y^2 = -\frac{1}{x^2}$$

На Python решить задачу Коши , $y(1) = 3$.

№ 21. *Дифференциальные уравнения второго порядка.*

$$x^3 y'' + x^2 y' = 1.$$

На Python решить уравнение

№ 22. *Системы дифференциальных уравнений.*

На Python решить систему уравнений $\frac{dx}{dt} = 7x + 3y$, $\frac{dy}{dt} = 6x + 4y$.

№ 23. *Численное решение дифференциальных уравнений.*

$$y'' + 4xy' + (4x^2 + 2)y = 0,$$

На Python найти численное решение задачи Коши $y(0) =$
 $y'(1) = 0$ на отрезке $[0; 2.5]$.

№ 24. *Матрицы.*

С помощью Python найдите четверную степень матрицы

$$\begin{pmatrix} 1 & 2 \\ 5 & 0 \end{pmatrix}.$$

№ 25. *Матрицы.*

Найти на Python ранг матрицы, составленной из векторов, и определить, являются ли

векторы $a = (0, 1, 1, 3)$, $b = (0, 2, 3, 0)$, $c = (2, 1, 1, 1)$ и $d = (-2, 2, 3, 2)$ линейно независимыми.

№ 26. Решение систем линейных уравнений.

На Python найти общее решение однородной системы линейных уравнений

$$\begin{cases} x - 2y + z + w = 0, \\ 2x + y + 2z + 3w = 0, \\ 2x - 5y + 2z - w = 0. \end{cases}$$

выписать фундаментальную систему решений.

№ 27. Линейные операторы.

$$A = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}$$

С помощью Python привести матрицу к диагональному виду и указать матрицу перехода T .

3.7. Практические задания для оценки компетенции «ПК-1 ППК-Р1.5»

№ 28. Пределы.

$$\lim_{x \rightarrow \infty} \frac{e^{4x} - 1}{1 - e^{5x}}$$

На Python вычислить предел

№ 29. Исследование функции.

$$f(x) = \frac{2+3x}{x+1}$$

На python найдите асимптоты графика функции

№ 30. ряды.

$$\sum_{n=1}^{\infty} \frac{n^2 + 12}{n^2 - n + 23}$$

На Python исследуйте ряд на сходимость:

№ 31. Производная.

С помощью Python исследовать на экстремум функцию $y = x^3 - 3x^2 + x + 4$.

№ 32. Интегралы.

$$\int_0^{\infty} e^{-x} dx$$

С помощью Python исследовать на сходимость интеграл

№ 33. Дифференциальные уравнения первого порядка.

$$y' + 6y^2 = -\frac{1}{x^2}$$

На Python решить задачу Коши $y(1) = 3$.

№ 34. Дифференциальные уравнения второго порядка.

$$x^3 y'' + x^2 y' = 1.$$

На Python решить уравнение

№ 35. Системы дифференциальных уравнений.

На Python решить систему уравнений $\frac{dx}{dt} = 7x + 3y$, $\frac{dy}{dt} = 6x + 4y$.

№ 36. Численное решение дифференциальных уравнений.

На Python найти численное решение задачи Коши $y'' + 4xy' + (4x^2 + 2)y = 0$, $y(0) = y'(1) = 0$ на отрезке $[0; 2.5]$.

№ 37. Матрицы.

С помощью Python найдите четвертую степень матрицы

$$\begin{pmatrix} 1 & 2 \\ 5 & 0 \end{pmatrix}.$$

№ 38. Матрицы.

Найти на Python ранг матрицы, составленной из векторов, и определить, являются ли векторы $a = (0,1,1,3)$, $b = (0,2,3,0)$, $c = (2,1,1,1)$ и $d = (-2,2,3,2)$ линейно независимыми.

№ 39. Решение систем линейных уравнений.

На Python найти общее решение однородной системы линейных уравнений

$$\begin{cases} x - 2y + z + w = 0, \\ 2x + y + 2z + 3w = 0, \\ 2x - 5y + 2z - w = 0. \end{cases}$$

выписать фундаментальную систему решений.

№ 40. Линейные операторы.

$$A = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}$$

С помощью Python привести матрицу к диагональному виду и указать матрицу перехода T .

3.8. Практические задания для оценки компетенции «ПК-19.7»

№ 41. Интегралы.

$$\int_0^{\infty} e^{-x} dx$$

С помощью Python исследовать на сходимость интеграл

3.9. Практические задания для оценки компетенции «ПК-1 ППК-Р1.1»

№ 42. Численное решение дифференциальных уравнений.

На Python найти численное решение задачи Коши $y'' + 4xy' + (4x^2 + 2)y = 0$, $y(0) = y'(1) = 0$ на отрезке $[0; 2.5]$.

3.10. Практические задания для оценки компетенции «ПК-1 ППК-Р1.2»

№ 43. Численное решение дифференциальных уравнений.

На Python найти численное решение задачи Коши $y'' + 4xy' + (4x^2 + 2)y = 0$, $y(0) = y'(1) = 0$ на отрезке $[0; 2.5]$.