МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ:
Проректор по учебной работе
_____ О.Г. Локтионова
«_5_» _12_____ 2025 г.

ТЕСТИРОВАНИЕ И ОЦЕНКА КАЧЕСТВА СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Методические указания к проведению
лабораторных и практических занятий
для студентов направления подготовки
09.04.01 Информатика и вычислительная техника

Курск 2025 г.

УДК 004.8
Составитель Е.Н. Иванова

Рецензент
Доцент кафедры программной инженерии,
кандидат технических наук
Т.Н. Конаныхина

Тестирование и оценка качества систем искусственного интеллекта : методические указания к проведению лабораторных и практических занятий для студентов направления подготовки 09.04.01 Информатика и вычислительная техника / Юго-Зап. гос. ун-т; сост.: Е.Н. Иванова. – Курск, 2025. – 14 с. – Библиограф.: с. 14.

Содержатся цель каждого занятия, краткая теория. Приводятся примерные задания для выполнения и их реализация на лабораторных и практических занятиях.

Методические указания соответствуют требованиям программ, утвержденным учебно-методическим объединением по направлению Информатика и вычислительная техника.

Предназначены для студентов очной и заочной форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60x84 1/16.
Усл.печ.л. Уч.-изд.л. . Тираж 20 экз. Заказ . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Лабораторная работа №1

Практическое занятие №1

Метрики сложности программ искусственного интеллекта

Цель работы:

Изучить применимость классических метрик сложности (Мак-Кейба, Джилба, метрик потока данных) для оценки программ, реализующих алгоритмы искусственного интеллекта (ИИ), и выполнить их сравнительный анализ.

Теоретическая часть

Оценка сложности традиционного ПО часто основывается на анализе управляющих структур и потоков данных. Однако программы ИИ (нейронные сети, агентные системы, алгоритмы поиска) обладают спецификой:

– декларативность. Часть логики может быть представлена данными (например, веса нейросети, правила базы знаний), что снижает ценность метрик, анализирующих только код;

– недетерминизм. Поведение системы может зависеть от случайных факторов (например, epsilon-жадная стратегия в обучении с подкреплением);

– сложность данных. Основная сложность может скрываться не в потоке управления, а в структуре и размерности обрабатываемых данных.

Тем не менее, классические метрики могут быть полезны для оценки сложности реализации вспомогательного кода, алгоритмов предобработки и логики принятия решений, не основанной на обучении.

Задание

В качестве объекта исследования выступает программный мо-

дуль, реализующий один из алгоритмов (согласно варианту).

Варианты заданий:

1. Агентная система. Реализация простого рефлекторного агента для навигации на сетке (среда "чистка пылесоса").

2. Алгоритм поиска. Реализация алгоритма A* (A-star) для поиска пути на графе.

3. Древо решений. Реализация алгоритма построения и использования дерева решений (например, классификатора на основе ID3).

4. Генетический алгоритм. Реализация базового цикла генетического алгоритма (селекция, скрещивание, мутация).

5. Логический вывод. Реализация простого механизма логического вывода на продукционных правилах (forward chaining).

Необходимо:

1. Разработать программу согласно варианту.

2. Выделить в программе часть, отвечающую за логику ИИ (ядро алгоритма), и вспомогательную часть (ввод данных, инициализация, вывод).

3. Для каждой из двух частей (ядро ИИ и вся программа целиком) рассчитать следующие метрики:

– цикломатическое число Мак-Кейба (Z). Построить управляющий граф для каждой функции/метода и рассчитать:

$$Z(G) = E - N + p,$$

где E – число ребер (ребро соединяет узел *m* с узлом *n,* если оператор *n* следует сразу за оператором *m*);

N – число узлов (операторов программы);

p – число компонентов связности графа.

– метрика Джилба. Рассчитать насыщенность условными операторами:

$$C_L = C_A / N$$

где C_A – число операторов вида IF – THEN – ELSE;

N – количество операторов.

– метрика Чепина. Проанализировать список ввода-вывода ядра ИИ. Разбить переменные на группы (*P*, *M*, *C*, *T*) и рассчитать *Q*:

$$Q = \alpha_1 P + \alpha_2 M + \alpha_3 C + \alpha_4 T,$$

где *P* – вводимые переменные для расчетов и для обеспечения вы-

вода. Примером может служить используемая в программах лексического анализатора переменная, содержащая строку исходного текста программы, т.е. сама переменная не модифицируется, а только содержит исходную информацию;

M – модифицируемые, или создаваемые внутри программы переменные;

C – переменные, участвующие в управлении работой программного модуля (управляющие переменные);

T – не используемые в программе ("паразитные") переменные;

$\alpha_1, \alpha_2, \alpha_3, \alpha_4$ – весовые коэффициенты.

Весовые коэффициенты использованы для отражения различного влияния на сложность программы каждой функциональной группы. По мнению автора метрики, наибольший вес, равный трем, имеет функциональная группа *C*, так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом:

$\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3, \alpha_4 = 0{,}5$.

– метрика спена. Спен – это число утверждений, содержащих данный идентификатор, между его первым и последним появлением в тексте программы. Следовательно, идентификатор, появившийся *n* раз, имеет спен, равный *n*-1. При большом спене усложняется тестирование и отладка. Усреднённый по всем идентификаторам, спен даёт сложность программы. Выбрать 2-3 ключевые переменные, управляющие логикой ИИ, и вычислить их спен. Усреднённый по всем идентификаторам, спен даёт сложность программы.

– уровень комментированности. Рассчитать:

$F = N_{KOM} / N_{CTP}$,

где N_{KOM} – количество комментариев в программе;

N_{CTP} – количество строк или операторов исходного текста.

Этапы выполнения

На лабораторном занятии:

1. Разработать программу, оформив код согласно требованиям.

На практическом занятии:

2. Построить управляющий граф для ключевых функций (например, функции выбора действия агентом или функции оценки пути в A*).

3. Выполнить подсчет метрик, заполнив сравнительную итоговую таблицу.

4. Провести сравнительный анализ полученных значений для ядра ИИ и вспомогательного кода. Сделать вывод о том, какие метрики лучше подходят для оценки сложности именно алгоритмической части ИИ, а какие – для оценки качества кода в целом.

Содержание отчета

1. Титульный лист.
2. Цель работы и текст задания.
3. Текст программы.
4. Управляющие графы для анализируемых функций.
5. Таблица со значениями всех рассчитанных метрик для ядра ИИ и всей программы.
6. Анализ полученных результатов.
7. Выводы о применимости классических метрик сложности к программам ИИ.

Лабораторная работа №2

Практическая работа №2

Тестирование программ искусственного интеллекта

Цель работы:

Освоить методы тестирования программного обеспечения, реализующего алгоритмы искусственного интеллекта, с учетом специфики их работы (недетерминизм, обучение, работа с данными).

Теоретическая часть

Тестирование систем ИИ сталкивается с проблемой "оракула" – сложно заранее определить единственно правильный результат работы. Например, для игры в ГО существует множество выигрышных ходов. Поэтому тестирование ИИ требует комбинации классических и специализированных подходов.

Тестирование "черного ящика" (функциональное). Проверка соответствия поведения системы спецификации на наборе входных данных.

Тестирование "белого ящика" (структурное). Проверка внутренних путей исполнения кода.

Метаморфное тестирование. Если для конкретного входа сложно определить правильный выход, можно проверить, сохраняются ли определенные соотношения между входами и выходами. Например, если перемешать порядок записей в обучающей выборке, точность модели после обучения должна измениться незначительно (в пределах допустимого).

Тестирование на основе сценариев. Проверка работы ИИ в специально сконструированных ситуациях (пограничные случаи, аномалии данных).

Регрессионное тестирование моделей. Проверка того, что точность модели не упала ниже установленного порога после внесения изменений в код или данные.

Задание

Для программы, разработанной в лабораторной работе №1 (или для новой программы согласно вариантам ниже), требуется разработать и применить тесты, использующие разные стратегии.

Варианты заданий:

1. Классификатор. Программа, классифицирующая ирисы Фишера на основе простейшего алгоритма (например, по ближайшему соседу или пороговым значениям).

2. Рекомендательная система. Программа, рекомендующая фильм на основе предпочтений пользователя по простым правилам (например, "если нравится жанр X, то предложить фильмы этого жанра с рейтингом > Y").

3. Игровой ИИ. Программа, реализующая алгоритм "минимакс" для игры "Крестики-нолики".

4. Обработка текста. Программа, определяющая тональность короткого сообщения (позитивная/негативная) на основе словаря тонов.

Необходимо:

1. Подготовить тесты по методу "черного ящика":

– применить метод эквивалентного разбиения для входных данных. Создать таблицу с допустимыми и недопустимыми классами;

– применить анализ граничных значений. Создать тесты для граничных ситуаций;

– (для детерминированных алгоритмов) применить анализ причинно-следственных связей, построив таблицу истинности для ключевых решающих правил.

2. Подготовить тесты по методу "белого ящика":

– построить блок-схему для фрагмента кода, реализующего логику ИИ (содержащего условные операторы);

– разработать тесты для обеспечения минимум 50% покрытия решений (переходов) и покрытия условий.

3. Применить метод метаморфного тестирования:

– сформулировать 2-3 метаморфных соотношения для вашей

программы (например, для классификатора: "если к объекту добавить шум, его класс не должен измениться, если шум мал");

– реализовать и выполнить тесты, проверяющие эти соотношения.

Этапы выполнения

На практическом занятии:

1. Изучить код программы и ее функциональные требования.
2. Выделить входные и выходные данные.
3. Поэтапно разработать тесты для каждого из трех указанных методов.

На лабораторном занятии:

4. Выполнить тестирование программы, зафиксировав результаты в таблицах.
5. Проанализировать, какие типы потенциальных ошибок были обнаружены или не обнаружены каждым из методов.

Содержание отчета

1. Титульный лист, цель работы.
2. Краткое описание программы и ее алгоритма.
3. Раздел "Тестирование черного ящика": таблицы с классами эквивалентности, граничными значениями и причинно-следственными связями; таблица с тестами и результатами.
4. Раздел "Тестирование белого ящика": блок-схема фрагмента кода; таблица с тестами, покрываемыми путями и результатами.
5. Раздел "Метаморфное тестирование": список метаморфных соотношений; таблица с исходными и последующими тестами и результатами проверки соотношений.
6. Выводы по работе, где сравнивается эффективность разных методов тестирования для данного класса программ ИИ.

Лабораторная работа №3

Практическая работа №3

Оценка надёжности программ искусственного интеллекта

Цель работы:

Изучить подходы к оценке надежности программного обеспечения, реализующего алгоритмы искусственного интеллекта, и выполнить сравнительный анализ классических и специализированных моделей.

Теоретическая часть

Понятие "отказ" для системы ИИ сложнее, чем для традиционной программы. Отказом может считаться не только "падение" (crash) программы, но и:

–·неприемлемое качество: точность классификации упала ниже допустимого порога;

– небезопасное поведение: автономный агент принял решение, приведшее к нежелательным последствиям в симулированной среде;

– деградация со временем: модель, работающая с данными, перестала быть адекватной (concept drift).

Поэтому оценка надежности ИИ включает в себя как классический анализ частоты сбоев, так и анализ стабильности качества.

Классическая модель Джелинского-Моранды предполагает, что интенсивность обнаружения ошибок пропорциональна числу оставшихся ошибок в коде. Может быть применена для оценки надежности реализации алгоритма (ошибки программиста).

Модель Миллса основана на методе "посева" ошибок. В контексте ИИ может быть адаптирована для оценки ненайденных проблем с данными или граничных случаев.

Валидационная модель обеспечивает оценку надежности через анализ точности на отложенной тестовой выборке и доверительных интервалов.

Анализ чувствительности – оценка того, как сильно меняется выход системы при малых возмущениях входа (шум, искажения).

Задание

Для программы, разработанной в лабораторной работе №1 или №2, необходимо выполнить две части исследования: оценить надежность самой программы как кода и оценить стабильность работы алгоритма ИИ.

Этапы выполнения:

1. Выполнить оценку надежности кода (Модель Миллса)

На лабораторном занятии:

1. Искусственно внести в программный код (например, в функции обработки данных) L=5 простых ошибок (например, неверное граничное условие, опечатка в формуле). Задокументировать их.

2. Провести серию тестов (можно использовать тесты из ЛР №2).

На практическом занятии:

3. Предположим, что в процессе тестирования было найдено m внесенных ошибок и v собственных. Используя формулу:

$$N = L \cdot v / m,$$

оценить первоначальное число собственных ошибок N для трех сценариев.

4. Рассчитать вероятность P (доверительный уровень) для утверждения о том, что в программе было не более R собственных ошибок, используя формулы:

$P = L / (L + R + 1)$, если обнаружены все ошибки,

$P = C_L^{l-1} / C_{L+R+1}^{l+R}$, если обнаружено l из L ошибок.

2. Выполнить оценку надежности и стабильности алгоритма ИИ

На лабораторном занятии:

1. Подготовить данные. Сгенерировать три массива входных данных $\{X_i\}$ объемом 30 элементов (например, значения признаков

для классификации), распределенных по разным законам:

– равномерное распределение;

– экспоненциальное распределение;

– рэлеевское распределение.

2. Провести эксперимент. Для каждого из 30 наборов данных зафиксировать результат работы программы (например, выданный класс или действие) и считать его "правильным" (или определить степень его приемлемости). Если результат не совпадает с ожидаемым (или качество низкое), это считается "отказом". Получить массив X_i – количество успешных срабатываний между "отказами".

На практическом занятии:

3. Применить модель Джелинского-Моранды:

– для каждого из трех массивов $\{X_i\}$ оценить первоначальное число "отказов" (ошибок качества) в программе – B;

– провести оценку, используя 100% (30 элементов), 80% (24 элемента) и 60% (18 элементов) данных, чтобы оценить стабильность прогноза.

– если B оказалось больше числа проанализированных отказов n, оценить среднее время X_j до обнаружения следующих 2-3 отказов.

Содержание отчета

1. Титульный лист, цель работы.

2. Оценка надежности кода (Модель Миллса): таблица с протоколом внесенных ошибок; таблица расчетов N и P для разных сценариев обнаружения. Анализ результатов.

3. Оценка надежности и стабильности алгоритма ИИ (Джелинского-Моранды):

– краткое описание методики сбора данных об "отказах";

– таблицы с исходными массивами $\{X_i\}$ для трех распределений;

– таблица с оценками B для разных законов распределения и разного объема данных $(3\times 3 = 9$ значений$)$. Анализ стабильности оценок;

– расчет прогнозируемого времени до следующих отказов.

4. Общий вывод по работе, сопоставляющий надежность кода и надежность алгоритма, а также влияние входных данных на этот показатель

Список использованных источников

1. Турнецкая Е. Л., Агроновский А. В. Программная инженерия. Тестирование и контроль качества программного обеспечения : учебное пособие для вузов. – СПб. : Лань, 2025. – 169 с.

2. Кейнер К., Бах Д., Петтикорд Б. Тестирование программного обеспечения: контекстно ориентированный подход / Пер. с англ. – СПб. : Питер, 2025. – 350 с.

3. Василькова А. Н., Воробей А. В., Медведев О. С., Прудник А. М. Технологии оценки качества программного обеспечения : пособие / под ред. Т. В. Казак. – Минск : БГУИР, 2025. – 85 с.

4. Черников Б. В., Поклонов Б. Е. Оценка качества программного обеспечения: практикум: учебное пособие для вузов / под ред. Б. В. Черникова. – Москва : Инфра-М: Форум, 2022. – 400 с.

5. Принципы верификации и тестирования моделей искусственного интеллекта : учебное пособие для вузов. – СПб. : Лань, 2025. – 160 с.

6. ГОСТ Р 56920-2024. Системная и программная инженерия. Тестирование программного обеспечения. Общие положения.

7. ГОСТ Р 56921. Системная и программная инженерия. Тестирование программного обеспечения. Часть 2. Процессы тестирования

8. ГОСТ Р 56922. Системная и программная инженерия. Тестирование программного обеспечения. Часть 3. Документация тестирования.

9. ГОСТ Р 71998-2025. Информационные технологии. Требования и оценка качества систем и программного обеспечения. Определение качества ИТ-услуг.

10. ПНСТ 965-2024. Системная и программная инженерия. Тестирование программного обеспечения. Часть 11. Тестирование систем искусственного интеллекта.