Ответы к экзамену по нейронным сетям

Дисциплина «Нейронные сети»

Единый конспект по вопросам

Содержание

Ответы подготовлены по материалам из директории `tyugashev/`. Если в материалах курса вопрос раскрыт слабо или отсутствует отдельный термин, это отмечено в ответе. В исходном списке вопросов отсутствует номер 13, поэтому нумерация сохранена как в документе `вопросы-2026-нейросети.docx`.

Основы искусственного интеллекта и машинного обучения

1. Понятие ИИ. Разные подходы к определению искусственного интеллекта. Определение из Национальной стратегии РФ

Искусственный интеллект можно понимать по-разному, потому что само слово «интеллект» не имеет одного строгого технического смысла. В самом общем виде ИИ - это область информатики, которая создает системы, способные решать задачи, обычно требующие человеческого мышления: распознавать образы, понимать и порождать текст, делать выводы, учиться по данным, планировать действия и адаптироваться к ситуации.

В материалах курса подчеркивается практическая сторона: когда сегодня говорят «искусственный интеллект», часто имеют в виду системы машинного обучения. Их ключевая особенность - адаптивность, то есть способность не только выполнять заранее прописанный алгоритм, но и подстраивать поведение на основе данных и опыта.

Существует несколько подходов к определению ИИ:

- имитационный подход - система считается интеллектуальной, если ее поведение похоже на поведение человека; классический пример - тест Тьюринга;
- когнитивный подход - ИИ пытается моделировать процессы человеческого мышления;
- рациональный подход - интеллектуальная система должна выбирать разумные действия для достижения цели;
- инженерный подход - ИИ рассматривается как набор технологий для решения прикладных задач: классификации, прогнозирования, генерации текста, управления, распознавания речи и изображений.

В Национальной стратегии развития искусственного интеллекта РФ ИИ определяется как комплекс технологических решений, который позволяет имитировать когнитивные функции человека, включая самообучение и поиск решений без заранее заданного алгоритма, и получать при решении конкретных задач результаты, сопоставимые с результатами интеллектуальной деятельности человека. Для экзамена важно запомнить две части этого определения: ИИ имитирует когнитивные функции и дает практически полезный результат, сопоставимый с человеческим.

Источники: `нейронные-сети-май-2026.doc`, введение, с. 3-4; раздел «Основы машинного обучения», с. 5-6; `нейронные-сети-май-2026.doc`, раздел «Философские, этические и социальные проблемы нейросетей», с. 166; Указ Президента РФ от 10.10.2019 N 490 «О развитии искусственного интеллекта в Российской Федерации».

2. Подходы к построению интеллектуальных систем. Логический подход. Статистические методы машинного обучения. Искусственные нейронные сети. Сильные и слабые стороны подходов

В курсе выделяется несколько способов добиться гибкого поведения программы. Самый простой путь - непосредственно запрограммировать варианты поведения через условия вида `если - то - иначе`. Такой подход понятен, но плохо масштабируется: при изменении предметной области приходится менять код, заново отлаживать программу и внедрять новую версию.

Логический, или символический, подход строит интеллектуальную систему как сочетание машины вывода и базы знаний. В базе знаний хранятся факты и правила, например продукционные правила вида «если выполнены условия, то сделать вывод». Сильная сторона такого подхода - объяснимость: можно найти конкретное правило, из-за которого система сделала вывод. Слабая сторона - знания нужно формализовать вручную, а это трудно, дорого и не всегда возможно, особенно если эксперт сам не осознает все правила своей деятельности.

Статистические методы машинного обучения исходят из другой идеи: вместо ручного задания правил система восстанавливает зависимость по эмпирическим данным. Есть обучающая выборка - набор примеров «вход - правильный ответ» или просто набор объектов без ответов. Алгоритм

ищет закономерности и затем применяет их к новым объектам. Сильные стороны - работа с большими данными, применимость к классификации, регрессии, прогнозированию, поиску ассоциаций. Слабые стороны - зависимость от качества данных, риск переобучения, вычислительная сложность и не всегда очевидная интерпретация результата.

Искусственные нейронные сети - частный, но сегодня наиболее важный класс методов машинного обучения. Их идея - грубо воспроизвести принцип работы нервной системы: большое число простых элементов соединены весами, а обучение состоит в подборе этих весов. Сильные стороны ИНС - способность автоматически выделять признаки, решать задачи компьютерного зрения, обработки текста, речи, генерации изображений и программного кода. Слабые стороны - потребность в больших данных и вычислительных ресурсах, сложность объяснения решения, ошибки и галлюцинации, необходимость подбирать архитектуру и гиперпараметры.

Для экзамена удобно сравнить подходы так: логический подход хорошо объясняет, но плохо учится; статистическое машинное обучение хорошо извлекает закономерности из данных, но зависит от выборки; нейросети дают лучшие результаты на сложных неструктурированных данных, но часто работают как «черный ящик».

Источники: `нейронные-сети-май-2026.doc`, раздел «Основы машинного обучения», с. 5-8; раздел «Нерешенные вопросы ИНС», с. 166-168.

3. История искусственных нейросетей

История искусственных нейронных сетей начинается с попытки формально описать работу нервной клетки. В 1943 году Мак-Каллок и Питтс предложили модель искусственного нейрона. В 1949 году Дональд Хебб сформулировал правило обучения: если два нейрона часто возбуждаются совместно, связь между ними усиливается. Эта идея легла в основу обучения по Хеббу.

В 1957 году Розенблатт разработал персептрон - классическую раннюю ИНС. Его аппаратная реализация «Марк-1» могла распознавать некоторые буквы английского алфавита. В этот период на нейросети возлагали большие надежды, но затем Минский и Пейперт показали принципиальные ограничения простых однослойных персептронов. Интерес к направлению снизился - это можно считать одним из первых «зимних» периодов нейросетевого подхода.

Новый подъем связан с появлением многослойных сетей и алгоритмов их обучения. Важную роль сыграл метод обратного распространения ошибки: идея распространения ошибки от выходов к входам была описана в 1970-е годы и получила развитие в 1986 году в работах Румельхарта, Хинтона и Вильямса. В 1982 году Хопфилд предложил сети ассоциативной памяти, в 1984 году Кохонен - самоорганизующиеся карты для кластеризации и визуализации.

В 1988 году Ян Лекун предложил сверточные нейронные сети для распознавания рукописных цифр. В 1997 году Хохрайтер и Шмидхубер предложили LSTM - рекуррентную архитектуру, способную лучше учитывать долгосрочные зависимости. Примерно с 2012 года началась эпоха бурного внедрения глубокого обучения: появились большие данные, выросла производительность GPU, стали развиваться сверточные, рекуррентные и гибридные архитектуры.

В обработке текста ключевым переломом стал 2017 год, когда появилась архитектура трансформер. На ней основаны BERT, GPT и современные большие языковые модели. В материалах курса отдельно отмечается цепочка качественных скачков: BERT -> GPT-3 -> InstructGPT -> ChatGPT -> DeepSeek R1.

Источники: `нейронные-сети-май-2026.doc`, раздел «История искусственных нейронных сетей», с. 25-27; раздел «Архитектура трансформер», с. 73-75; `история-gpt.doc`.

4. Классические задачи машинного обучения и их разновидности. Плюсы и проблемы машинного обучения

Машинное обучение решает задачу восстановления зависимости по данным. Есть множество объектов, по каждому собраны признаки, и требуется построить процедуру, которая по новому объекту выдает ответ. Ответ может быть классом, числом, последовательностью, действием или найденной структурой в данных.

Классические задачи обучения с учителем:

- классификация - выбор класса из конечного множества, например «кошка» или «собака»;
- регрессия - предсказание числа или числового вектора, например цены квартиры;
- прогнозирование - предсказание будущих значений временного ряда;
- seq2seq - преобразование последовательности в последовательность, например машинный перевод;
- преобразование изображений, речи, текста и мультимодальных данных.

Задачи обучения без учителя включают кластеризацию, поиск ассоциативных правил, обнаружение выбросов, сокращение размерности и заполнение пропущенных значений. Отдельно выделяются частичное обучение, когда ответы известны только для части объектов, обучение с подкреплением, где агент получает награды от среды, и динамическое обучение, когда данные поступают потоком.

Плюсы машинного обучения состоят в том, что оно позволяет находить закономерности в больших и сложных данных, решать задачи, для которых невозможно явно написать все правила, и переносить обученную модель на новые примеры. Поэтому МО применяется в медицине, экономике, геологии, робототехнике, компьютерном зрении, распознавании речи, офисной автоматизации и анализе текстов.

Проблемы машинного обучения - переобучение, зависимость от качества обучающей выборки, вычислительная стоимость, необходимость подбирать признаки и параметры, а также трудность объяснения результата у сложных моделей. Если модель обучена на неправильных или смещенных данных, она может показывать хорошую точность на тесте, но плохо работать в реальной эксплуатации.

Источники: `нейронные-сети-май-2026.doc`, раздел «Классические задачи машинного обучения», с. 10-15; раздел «Обучение нейросети с учителем», с. 40-43.

Архитектуры искусственных нейронных сетей

5. Формальный нейрон. Функция активации. Виды функций активации

Формальный нейрон - это искусственный элемент нейросети, который принимает несколько входных сигналов, умножает их на веса, суммирует и преобразует результат с помощью функции активации. С математической точки зрения он реализует скалярную функцию векторного аргумента.

Если входы обозначить как x_i, веса как w_i, а смещение как b, то суммарное возбуждение нейрона записывается так:

$$s = b + \sum_{i=1}^{n} x_i w_i.$$

Выход нейрона равен:

$$y = f(s),$$

где f - функция активации. Она определяет, как нейрон реагирует на суммарный входной сигнал. Без нелинейной активации многослойная сеть сводилась бы к одному линейному преобразованию и не могла бы эффективно описывать сложные зависимости.

В материалах курса перечислены пороговые и сигмоидальные функции активации:

- единичный скачок - выход равен 0 до порога и 1 после порога;
- линейный порог - похож на скачок, но имеет небольшой линейный участок;
- логистическая сигмоида - гладкая S-образная функция со значениями от 0 до 1;
- гиперболический тангенс - похож на сигмоиду, но имеет область значений от −1 до 1.

Для современных глубоких сетей также часто используют ReLU, LeakyReLU, GELU и softmax. ReLU обнуляет отрицательные значения и оставляет положительные, поэтому хорошо подходит для скрытых слоев. Softmax обычно применяют на выходе классификатора или языковой модели, чтобы превратить логиты в вероятности классов или токенов.

Источники: `нейронные-сети-май-2026.doc`, раздел «Структура формального нейрона», с. 30-33; разделы с примерами Keras/PyTorch, с. 98-122; `устройство-gpt.docx`.

6. Основные архитектуры современных искусственных нейронных сетей

Архитектура нейронной сети - это способ соединения нейронов и слоев. Один формальный нейрон сам по себе решает только простую задачу, а реальные системы строятся как большие сети с разными типами связей.

К основным архитектурам относятся:

- полносвязные сети прямого распространения - каждый слой передает сигналы следующему, часто каждый нейрон слоя связан со всеми нейронами следующего слоя;
- сверточные сети - используют свертки и подвыборку, особенно эффективны в компьютерном зрении;
- рекуррентные сети - имеют обратные связи и обрабатывают последовательности с учетом предыстории;
- LSTM и GRU - улучшенные рекуррентные сети с вентилями, предназначенные для долгосрочных зависимостей;
- трансформеры - архитектуры на механизме самовнимания, лежащие в основе BERT, GPT и современных больших языковых моделей;
- сети ассоциативной памяти, например сети Хопфилда;
- самоорганизующиеся карты Кохонена - сети обучения без учителя для кластеризации и визуализации.

В учебном пособии также упоминаются новые направления, например KAN - сети Колмогорова-Арнольда, где подбираются функции на связях между нейронами. Это скорее современное исследовательское направление, чем базовая архитектура для экзамена.

Выбор архитектуры зависит от задачи. Для табличных признаков часто достаточно полносвязной сети. Для изображений обычно используют CNN. Для текстов, речи и временных рядов исторически применяли RNN, LSTM и GRU, а сегодня ведущую роль заняли трансформеры. Для генерации текста и работы с длинным контекстом трансформер обычно предпочтительнее классической рекуррентной сети.

Источники: `нейронные-сети-май-2026.doc`, раздел «Архитектуры нейронных сетей», с. 34-37; раздел «Применение современных нейросетей», с. 57-76.

7. Полносвязная глубокая нейронная сеть прямого распространения - особенности

Полносвязная глубокая нейронная сеть прямого распространения - это многослойная сеть, в которой сигнал идет только от входа к выходу, без обратных связей. Между входным и выходным слоями находятся скрытые слои. Если каждый нейрон предыдущего слоя соединен со всеми нейронами следующего слоя, сеть называется полносвязной.

Особенности такой сети:

- данные проходят последовательно: входной слой -> скрытые слои -> выходной слой;
- в каждом слое выполняется линейное преобразование и нелинейная функция активации;
- обратных связей нет, поэтому сеть не хранит внутреннее состояние последовательности;
- вход обычно имеет фиксированную размерность;
- чем больше скрытых слоев, тем глубже сеть и тем более сложные зависимости она может аппроксимировать.

Полносвязные сети удобны как базовая архитектура: их легко понять, реализовать и обучать методом обратного распространения ошибки. Они подходят для табличных данных, простых классификаторов и регрессионных задач. В сверточных сетях полносвязные слои часто стоят в конце и используют найденные признаки для классификации.

Главный недостаток полносвязной сети - большое число параметров. Если подать на вход изображение напрямую, каждый пиксель будет связан со многими нейронами, что ведет к огромному числу весов и риску переобучения. Такая сеть плохо использует структуру данных: для изображения важна локальность пикселей, для текста - порядок слов. Поэтому для изображений лучше CNN, а для последовательностей - RNN/LSTM/GRU или трансформеры.

Источники: `нейронные-сети-май-2026.doc`, раздел «Архитектуры нейронных сетей», с. 34-36; разделы с примерами Keras/PyTorch, с. 98-122.

8. Нейросети с обратными связями. Классификация. Примеры, сферы использования

Нейросети с обратными связями - это сети, в которых информация может передаваться не только от входа к выходу, но и обратно: с последующих слоев на предыдущие или от состояния сети в предыдущий момент времени к состоянию в следующий момент. Такие сети называют рекуррентными.

В материалах курса выделяются следующие варианты сетей с обратными связями:

- слоисто-циклические - слои замкнуты в кольцо, выходной слой передает сигналы входному;
- слоисто-полносвязные - внутри слоя есть обмен сигналами, а затем результат передается дальше;
- полносвязанно-слоистые - нейроны принимают сигналы как от своего слоя, так и от последующих слоев;
- частично-рекуррентные сети Элмана и Жордана;
- LSTM и GRU как современные модификации рекуррентных сетей.

Главная идея RNN - память о предшествующей последовательности. При обработке текста, речи или временных рядов отдельный элемент часто нельзя понять без контекста. Например, в тексте важно, какие слова были раньше, а в прогнозе курса валют важна история значений.

Сферы применения рекуррентных сетей - машинный перевод, генерация текстов, распознавание речи, анализ временных рядов, музыка, рукописный ввод, робототехника и анализ активности человека. Однако у классических RNN есть серьезные проблемы: последовательная обработка плохо параллелится, обучение медленное, а при длинных цепочках возникает затухание или взрыв градиентов. Поэтому на практике классические RNN часто заменяют LSTM, GRU или трансформерами.

Источники: `нейронные-сети-май-2026.doc`, раздел «Архитектуры нейронных сетей», с. 34-36; раздел «Рекуррентные нейронные сети», с. 67-72.

9. Архитектуры нейросетей LSTM и GRU

LSTM, или long short-term memory, - это разновидность рекуррентной нейронной сети, созданная для работы с долгосрочными зависимостями. Обычная RNN теоретически может учитывать длинный контекст, но на практике ранняя информация постепенно искажается, а при обучении возникает затухание градиента. LSTM решает эту проблему с помощью специальной ячейки памяти и вентилей.

В LSTM-блоке есть несколько управляющих вентилей. Входной вентиль контролирует, какая новая информация попадет в память. Вентиль забывания определяет, какую старую информацию сохранить или удалить. Выходной вентиль управляет тем, какая часть состояния памяти будет использована для формирования выхода. Благодаря этому LSTM может удерживать важный контекст на длинных промежутках.

GRU, или gated recurrent unit, - более простая архитектура того же семейства. Она также использует вентили, но вычислений в ней меньше. У ячейки GRU есть текущее входное значение, предыдущее состояние и новое состояние. В материалах курса отмечается, что GRU похожа по сути на LSTM, но работает быстрее, поэтому применяется достаточно часто.

Разница для экзамена:

- LSTM сложнее, имеет отдельное состояние памяти и несколько вентилей;
- GRU проще, обычно быстрее обучается и требует меньше вычислений;
- обе архитектуры предназначены для последовательностей и долгосрочных зависимостей;
- обе применяются в текстах, речи, временных рядах и других данных, где порядок элементов важен.

Источники: `нейронные-сети-май-2026.doc`, раздел «Рекуррентные нейронные сети», с. 67-72; раздел «Нейросети для анализа естественного языка», с. 134-141.

10. Сверточные нейронные сети. История. Основные идеи, сферы применения

Сверточные нейронные сети, или CNN, - это архитектуры, специально приспособленные к данным с локальной пространственной структурой. Прежде всего это изображения. До широкого применения нейросетей для компьютерного зрения человек вручную придумывал признаки: линии, углы, цвета, текстуры. CNN позволяют подать изображение на вход сети, а признаки сеть выделяет сама в процессе обучения.

Исторически сверточные сети связаны с Яном Лекуном, который в 1988 году использовал их для распознавания рукописных цифр. Новый крупный скачок произошел в 2012 году, когда сеть Крижевского и Хинтона резко снизила ошибку распознавания на ImageNet. С этого момента сверточные сети стали базовой архитектурой компьютерного зрения.

Главная операция CNN - свертка. Небольшое ядро свертки проходит по изображению как фильтр, умножает значения пикселей на веса и формирует карту признаков. На ранних слоях сеть выделяет простые признаки: границы, линии, переходы яркости. На более глубоких слоях появляются сложные признаки: части объектов, морды животных, колеса, лица и т.д. После сверточных слоев обычно применяют pooling - подвыборку, которая уменьшает размер карты признаков и оставляет наиболее важную информацию.

Преимущества CNN - меньше параметров по сравнению с полносвязной сетью для изображений, более быстрый подбор весов, меньшая склонность к переобучению, способность строить иерархию признаков. На выходе часто используются полносвязные слои для классификации или регрессии.

Сферы применения: медицинские изображения, распознавание лиц, видеонаблюдение, беспилотный транспорт, робототехника, классификация и сегментация изображений, обработка снимков МРТ, КТ, рентгена. В материалах также отмечено, что CNN хуже подходят для последовательных данных и временных рядов, где порядок во времени важнее локальной пространственной структуры.

Источники: `нейронные-сети-май-2026.doc`, раздел «История искусственных нейронных сетей», с. 25-27; раздел «Сверточные сети», с. 57-66.

11. Использование нейросетей для обработки текста. Виды решаемых задач, применяемые архитектуры

Обработка естественного языка, или NLP, находится на пересечении искусственного интеллекта и математической лингвистики. В материалах курса выделяются две большие группы задач: анализ текста и синтез текста. Анализ отвечает за понимание или извлечение информации, синтез - за порождение нового текста или речи.

К задачам NLP относятся:

- распознавание речи и перевод речи в текст;
- реферирование и аннотирование;
- информационный поиск;
- классификация текстов по темам;
- анализ тональности отзывов;
- выделение именованных сущностей и фактов;
- чат-боты и диалоговые системы;
- машинный перевод;
- генерация текста;
- синтез речи.

Нейросеть не может напрямую работать с текстом как со строкой. Сначала текст нужно очистить, разбить на токены, привести слова к нужной форме при необходимости и представить в числовом виде. Используются разные подходы: bag of words, one-hot encoding, TF-IDF, семантические модели, word2vec, fastText, GloVe, а также обучаемые Embedding-слои.

Для обработки текста применялись разные архитектуры. Одномерные CNN могут анализировать локальные сочетания слов как одномерную структуру. RNN, LSTM и GRU учитывают порядок слов и предысторию последовательности. ELMo использовала две LSTM в разных направлениях, чтобы учитывать контекст слева и справа. BERT и GPT основаны на трансформерах: они используют механизм внимания и сегодня являются основой большинства сильных языковых моделей.

Для экзамена главное: текст сначала превращается в числа, затем модель учит связи между словами или токенами. Старые подходы опирались на частоты и простые векторы, затем появились word embeddings, рекуррентные сети, а современный стандарт - трансформеры и большие языковые модели.

Источники: `нейронные-сети-май-2026.doc`, раздел «Нейросети для анализа естественного языка», с. 134-141; `устройство-gpt.docx`; `история-gpt.doc`.

12. Архитектура «трансформер»

Трансформер - это архитектура нейронной сети, предложенная в 2017 году в работе «Attention is All You Need». Ее главное отличие от рекуррентных сетей состоит в том, что она не передает контекст по цепочке от слова к слову. Вместо этого она использует механизм самовнимания, который позволяет каждому токену учитывать другие токены последовательности.

Ключевая идея самовнимания: для каждого токена строятся три представления - Query, Key и Value. Query можно понимать как «что я ищу», Key - «по чему меня можно найти», Value - «какую информацию передать». Вес внимания между токенами вычисляется через сходство Query и Key, затем применяется softmax, и результат используется для взвешенной суммы Value.

Упрощенная формула внимания:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V.$$

В классическом трансформере есть кодировщик и декодировщик. Кодировщик строит представление входной последовательности. Декодировщик генерирует выходную последовательность, используя уже сгенерированное начало и информацию кодировщика. Такая схема хорошо подходит для машинного перевода. В GPT используется в основном декодерная часть с причинной маской: модель не может смотреть в будущие токены и предсказывает следующий токен по предыдущим.

Типичный блок трансформера включает:

- векторизацию токенов и добавление позиционной информации;
- слой самовнимания, часто многоголовый;
- остаточные соединения, помогающие обучать глубокую сеть;
- нормализацию;
- полносвязный MLP-блок для обработки признаков каждого токена;
- выходной линейный слой и softmax для получения вероятностей.

Преимущество трансформера - возможность учитывать дальние связи в тексте и лучше использовать параллельные вычисления, чем RNN. Поэтому трансформеры стали основой BERT, GPT, ChatGPT, Claude, Qwen, DeepSeek, YandexGPT, GigaChat и других современных языковых моделей.

Источники: `нейронные-сети-май-2026.doc`, раздел «Архитектура трансформер», с. 73-75; `устройство-gpt.docx`; `для-занятия-30-04-2026-gpt-perplp.docx`; `история-gpt.doc`.

Современные языковые модели и практика применения

14. Использование современных языковых моделей ИИ. Базовые принципы промпт-инжиниринга

Современные языковые модели - это большие нейросети на архитектуре трансформера, обученные предсказывать следующий токен по предыдущему контексту. В материалах GPT прямо сравнивается с «T9 на стероидах»: модель не думает словами в человеческом смысле, а строит вероятностное продолжение текста на основе огромного опыта обучения.

Путь данных в GPT выглядит так: текст -> токены -> номера токенов -> векторы -> позиционная информация -> блоки трансформера -> логиты для словаря -> softmax -> вероятности следующего токена. После выбора токена он добавляется к контексту, и процесс повторяется. Поэтому GPT называется авторегрессионной моделью.

Языковые модели используют для ответов на вопросы, перевода, суммаризации, объяснения кода, генерации программ, анализа документов, поиска идей, стилизации текста, подготовки учебных материалов и работы в роли виртуальных помощников. При этом нужно помнить о рисках: модель может галлюцинировать, выдавать устаревшую информацию, уверенно ошибаться или неправильно понять задачу.

Базовые принципы промпт-инжиниринга:

- формулировать запрос конкретно и однозначно;
- задавать роль модели, если это помогает задаче;
- давать контекст и целевую аудиторию;
- указывать формат, стиль, объем и ограничения ответа;
- прикладывать источник, если ответ должен опираться только на него;
- приводить примеры желаемого ответа - one-shot или few-shot;
- использовать пошаговое уточнение, если первая версия не подходит;
- просить модель улучшить сам промпт, когда задача плохо формализована.

Хороший промпт должен быть понятен и человеку, и модели. Если задача требует точности, нужно прямо запретить выдумывание: «используй только приложенный источник», «если данных нет - скажи, что данных нет». В этом смысле промпт-инжиниринг постепенно переходит в контекст-инжиниринг: важен не один запрос, а полный набор инструкций, примеров, документов и ограничений.

Источники: `нейронные-сети-май-2026.doc`, раздел «Введение в промпт-инжиниринг и контекст-инжиниринг», с. 160-164; `устройство-gpt.docx`; `для-занятия-30-04-2026-gpt-perplp.docx`; `история-gpt.doc`.

15. Вайбкодинг и будущее ИТ. Ваше мнение

Термин «вайбкодинг» в материалах курса отдельно не раскрыт, но в них есть близкая тема - автоматизированная генерация исходных текстов программ по постановке задачи на естественном языке. Вайбкодинг можно понимать как стиль разработки, при котором человек описывает желаемое поведение программы, интерфейс или исправление, а большая языковая модель генерирует код, тесты, объяснения и варианты реализации.

Мое мнение: вайбкодинг не отменяет инженерную профессию, но меняет ее центр тяжести. Рутинное написание шаблонного кода, перенос между языками, генерация CRUD-частей, подготовка тестовых набросков и рефакторинг будут все сильнее автоматизироваться. В материалах курса прямо отмечено, что ИИ уже помогает генерировать «скелет» приложения, переводить код с одного языка на другой и разгружать программиста от рутинных задач.

Но разработка ПО не сводится к набору кода. Инженер отвечает за постановку задачи, архитектуру, ограничения, безопасность, проверку корректности, сопровождение, интеграцию с реальными системами и ответственность за последствия. Модель может написать убедительный, но ошибочный код, не знать скрытых требований, нарушить безопасность, неправильно обработать крайние случаи или использовать неподходящую библиотеку.

Поэтому будущее ИТ, скорее всего, не «программисты исчезнут завтра», а «слабый программист без понимания системы станет менее конкурентоспособен». Ценность будет смещаться к тем, кто умеет формулировать требования, читать и проверять сгенерированный код, проектировать архитектуру, писать тесты, контролировать качество данных и работать с ИИ как с ускорителем. Вайбкодинг полезен как инструмент быстрого прототипирования, но опасен, если заменяет инженерную проверку ощущением «выглядит нормально».

Краткий экзаменационный вывод: ИИ станет постоянным инструментом программиста, резко ускорит часть разработки, но ответственность за качество, архитектуру и безопасность останется за человеком или организацией, внедряющей систему.

Источники: `нейронные-сети-май-2026.doc`, раздел «Нейрострудники», с. 154-160; раздел «Введение в промпт-инжиниринг и контекст-инжиниринг», с. 160-164. Термин «вайбкодинг» в материалах курса явно не раскрыт, ответ дополнен общетеоретическим пояснением.

Обучение нейронных сетей и качество данных

16. Методы обучения искусственных нейронных сетей. Обучение однослойной сети

Обучение нейронной сети - это настройка весов ее связей. В материалах курса подчеркивается: нейросеть не программируют подробными правилами решения задачи, а обучают, то есть подбирают веса так, чтобы сеть давала нужные ответы.

Процесс обучения обычно включает два этапа:

1. Весам присваивают начальные значения, часто небольшие случайные.
2. Затем веса поэтапно корректируются так, чтобы улучшать функционал качества.

Один полный проход по обучающей выборке называется эпохой.

Основные виды обучения:

- обучение с учителем - есть правильные ответы, сеть сравнивает свой выход с ними и исправляет веса;
- обучение без учителя - правильных ответов нет, сеть сама ищет структуру в данных, например кластеры;
- обучение с подкреплением - агент действует в среде и получает награду или штраф;
- динамическое обучение - данные поступают потоком, а модель дообучается по мере работы.

Обучение однослойной сети удобно рассматривать на примере персептрона. Пусть входы принимают значения 0 или 1, а сеть должна распознавать цифру. Если ответ правильный, веса не меняются. Если сеть выдала 1, хотя должна была выдать 0, нужно уменьшить веса активных входов. Если сеть выдала 0, хотя должна была выдать 1, нужно увеличить веса активных входов. В общем виде правило можно записать так:

$$w_i \leftarrow w_i + \eta(d - y)x_i,$$

где d - правильный ответ, y - ответ сети, x_i - входной сигнал, η - скорость обучения. Для простого персептрона это означает: усиливать связи, которые помогают правильному ответу, и ослаблять связи, которые привели к ошибке.

Источники: `нейронные-сети-май-2026.doc`, раздел «Обучение нейронной сети», с. 39-40; раздел «Обучение нейросети с учителем», с. 40-43; пример обучения персептрона, с. 46-56.

17. Методы обучения искусственных нейронных сетей. Проблемы глубокого обучения. Метод обратного распространения ошибки

Для обучения нейросетей применяются методы оптимизации. В материалах курса перечислены методы локальной оптимизации с частными производными первого порядка, включая градиентный спуск и сопряженные градиенты, методы второго порядка, например метод Ньютона и Левенберга-Маркардта, а также стохастические методы.

Главная идея обучения с учителем: у сети есть вход и правильный выход. Сеть делает прогноз, считается ошибка, затем веса корректируются так, чтобы ошибка уменьшалась. Для глубоких сетей это сложнее, чем для однослойного персептрона, потому что правильные выходы скрытых слоев неизвестны. Нельзя напрямую сказать каждому скрытому нейрону, каким должен быть его ответ.

Метод обратного распространения ошибки решает эту проблему. Сначала выполняется прямой проход: входные данные идут от входного слоя к выходному, и сеть получает прогноз. Затем считается функция потерь. После этого ошибка распространяется обратно - от выходного слоя к предыдущим слоям. По цепному правилу вычисляется вклад каждого веса в итоговую ошибку, и веса обновляются в направлении уменьшения ошибки.

Краткий алгоритм:

1. Инициализировать веса случайными малыми значениями.
2. Подать обучающий пример на вход и выполнить прямой проход.

3. Посчитать ошибку выходного слоя.
4. Распространить ошибку назад по слоям.
5. Вычислить поправки весов через градиенты.
6. Обновить веса.
7. Повторять до приемлемой ошибки или до остановки обучения.

Проблемы глубокого обучения: переобучение, локальные минимумы, паралич сети, затухание и взрыв градиентов, высокая вычислительная стоимость, чувствительность к архитектуре и гиперпараметрам. Для борьбы с ними используют нормализацию, dropout, остаточные связи, подбор оптимизаторов, раннюю остановку, больше данных и перенос обучения с предобученных моделей.

Источники: `нейронные-сети-май-2026.doc`, раздел «Обучение нейросети с учителем», с. 40-43; раздел «Рекуррентные нейронные сети», с. 67-72; разделы с примерами Keras/PyTorch, с. 98-122.

18. Использование градиентов в глубоком обучении. Проблемы метода, включая затухание градиентов

Градиент показывает, как изменится функция потерь при небольшом изменении параметра модели. Если функция потерь - это ошибка сети, то градиент подсказывает, в какую сторону нужно изменить веса, чтобы уменьшить ошибку. В глубоком обучении градиенты вычисляются методом обратного распространения ошибки.

Упрощенное правило градиентного спуска:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w},$$

где w - вес, L - функция потерь, η - скорость обучения. Если градиент большой, вес меняется сильнее; если маленький - слабее.

В глубокой сети градиент для ранних слоев получается как произведение многих производных по цепному правилу. Поэтому возникают две классические проблемы. Если многие множители меньше 1, произведение быстро стремится к нулю - это затухание градиента. Ранние слои почти не получают сигнала ошибки и плохо обучаются. Если многие множители больше 1, произведение резко растет - это взрыв градиента. Тогда веса начинают изменяться слишком сильно, обучение становится нестабильным.

Простая иллюстрация:

$$0.5^{20} \approx 0.000001,$$

то есть после многих слоев сигнал почти исчезает. А если множитель равен 2, то:

$$2^{20} = 1048576,$$

и сигнал становится чрезмерно большим. Именно поэтому в материалах курса при рекуррентных сетях говорится: если веса меньше 1, возникает затухающий градиент, если больше 1 - взрывной.

Проблемы градиентного метода также включают попадание в локальные минимумы, чувствительность к скорости обучения, зависимость от масштаба признаков, шумность стохастического градиента и необходимость дифференцируемых операций. На практике помогают ReLU/LeakyReLU/GELU, нормализация, остаточные соединения, clipping градиента, Adam и другие адаптивные оптимизаторы, корректная инициализация весов.

Источники: `нейронные-сети-май-2026.doc`, раздел «Обучение нейросети с учителем», с. 40-43; раздел «Рекуррентные нейронные сети», с. 67-72; `устройство-gpt.docx`.

19. Как качество данных, используемых при обучении, влияет на качество решения нейросетями поставленных задач? Приведите примеры

Качество данных напрямую определяет качество нейросети. Сеть обучается не тому, что разработчик имел в виду, а тем закономерностям, которые реально присутствуют в обучающей выборке. Если

данные неполные, смещенные, шумные или неправильно размеченные, модель будет воспроизводить эти ошибки.

В материалах курса приведен показательный пример с распознаванием танков. Сеть вроде бы успешно обучалась отличать танки на фотографиях, но позже выяснилось, что все изображения танков были сняты на одинаковом фоне. Модель «научилась» распознавать фон, а не танк. Это пример ложной корреляции: модель выбрала самый простой признак, который помогал на обучающей выборке, но не отражал суть задачи.

Основные проблемы данных:

- недостаточный объем - сеть не видит разнообразия реальных случаев;
- плохая разметка - правильные ответы содержат ошибки;
- смещение выборки - одни группы объектов представлены лучше других;
- утечка данных - в обучении случайно появляется информация, которой не будет при эксплуатации;
- шум и выбросы - модель начинает учитывать случайные ошибки;
- несовпадение обучающих и реальных данных - в эксплуатации распределение другое.

Примеры. Если медицинская система обучалась только на данных пациентов из одной страны, она может хуже работать на пациентах другой страны, потому что отличаются образ жизни, генетика, протоколы обследований и структура болезней. В материалах курса упоминается проблемный опыт IBM Watson в медицине и ситуация, когда обучение на американских пациентах плохо переносится на индийскую клинику. В компьютерном зрении плохое освещение, однотипный фон или отсутствие редких классов приведут к ошибкам. В языковых моделях некачественные тексты и устаревшие данные повышают риск галлюцинаций и неверных ответов.

Для экзамена главный вывод: данные должны быть репрезентативными, корректно размеченными, достаточно разнообразными и похожими на реальные условия применения. Иначе даже хорошая архитектура даст плохой результат.

Источники: `нейронные-сети-май-2026.doc`, раздел «Обучение нейросети с учителем», с. 40-43; раздел «Сверточные сети», с. 57-66; раздел «Нерешенные вопросы ИНС», с. 166-170.

20. Каковы основные параметры, которые необходимо подбирать при обучении и использовании искусственных нейронных сетей?

При обучении нейросети подбирают не только веса. Веса меняются автоматически в процессе обучения, а разработчик выбирает архитектуру, гиперпараметры, функцию потерь, способ подготовки данных и параметры использования модели.

Основные параметры архитектуры:

- число слоев;
- число нейронов или каналов в слоях;
- типы слоев: полносвязные, сверточные, рекуррентные, LSTM, GRU, Transformer-блоки;
- функции активации: sigmoid, tanh, ReLU, LeakyReLU, GELU, softmax;
- наличие dropout, нормализации, residual-связей;
- размерность эмбеддингов для текста;
- длина входной последовательности или контекстного окна.

Основные параметры обучения:

- обучающая, валидационная и тестовая выборки;
- функция потерь, например MSE или кросс-энтропия;
- оптимизатор: SGD, RMSprop, Adam и др.;
- скорость обучения;
- размер мини-выборки, или batch size;
- число эпох;
- способ инициализации весов;
- регуляризация и ранняя остановка;

- метрики качества: accuracy, F1, precision, recall, MAE и т.д.

При использовании обученной модели тоже есть параметры. Для классификации выбирают порог принятия решения. Для языковых моделей подбирают температуру, top-k, top-p, максимальную длину ответа, системную инструкцию, контекст и формат вывода. Для компьютерного зрения могут задаваться пороги уверенности и параметры постобработки.

В материалах курса прямо указывается, что подбор лучшей нейросети - это во многом задача оптимизации: нужно выбирать число слоев, виды слоев, функции активации и гиперпараметры. В практических заданиях предлагается менять длину вектора представления слов, число нейронов рекуррентного слоя, число рекуррентных слоев, оптимизатор, число эпох и размер мини-выборки.

Источники: `нейронные-сети-май-2026.doc`, разделы с примерами Keras/PyTorch, с. 98-122; раздел «AutoML - ИИ совершенствует себя», с. 145-154; контрольные вопросы главы 3, с. 165.