ЛЕКЦИЯ 8
НЕЙРОННЫЕ СЕТИ И ТЕХНОЛОГИИ ГЛУБОКОГО ОБУЧЕНИЯ

Цель лекции: Изучить принципы построения и обучения искусственных нейронных сетей, устройство искусственного нейрона и персептрона, архитектуру многослойного персептрона, функции активации, прямое и обратное распространение сигнала, причины недообучения и переобучения, а также основные возможности TensorFlow и Keras для разработки моделей глубокого обучения.

Искусственные нейронные сети

Искусственная нейронная сеть (ИНС) — вычислительная модель, состоящая из взаимосвязанных вычислительных элементов — искусственных нейронов. Нейроны получают входные значения, преобразуют их с учётом весовых коэффициентов и передают результат следующим элементам сети.

Нейронные сети используются для классификации, аппроксимации функций, распознавания образов, обработки изображений, текста и последовательностей. В загруженных учебных материалах отдельно рассматриваются искусственный нейрон, синаптические связи, функции активации, однослойные и многослойные сети, обучение с учителем и алгоритм обратного распространения ошибки.

Классические методы	Нейронные сети
Часто требуют ручного проектирования признаков	Способны обучать сложные представления признаков
Могут быть проще для интерпретации	Глубокие модели обычно сложнее интерпретировать
Хорошо работают на многих табличных задачах	Особенно эффективны для изображений, речи, текста и сложных зависимостей
Обычно имеют относительно простую архитектуру	Могут содержать большое число слоёв и параметров

Искусственный нейрон

Искусственный нейрон получает несколько входных сигналов x_1, x_2, …, x_n. Каждому входу соответствует вес w_i. Сначала вычисляется взвешенная сумма входов и добавляется смещение b:

$$z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

После этого к z применяется функция активации f(z), и формируется выход нейрона:

$$y = f(z)$$

Вес показывает, насколько сильно соответствующий вход влияет на результат. Смещение позволяет сдвигать границу решения. В процессе обучения веса и смещения изменяются так, чтобы уменьшить ошибку модели.

Элемент	Назначение
$x_1 \ldots x_n$	Входные признаки
$w_1 \ldots w_n$	Обучаемые веса
b	Смещение
Σ	Взвешенное суммирование
f(z)	Функция активации
y	Выход нейрона

Биологическая аналогия и ограничения аналогии

Термины «нейрон», «синапс» и «сеть» исторически связаны с биологической аналогией. В искусственной модели входные связи сопоставляются с синапсами, а веса — с силой влияния сигналов. Однако искусственный нейрон является математической вычислительной моделью и не воспроизводит полностью работу биологической нервной системы.

Персептрон

Персептрон — одна из простейших моделей искусственного нейрона, применяемая для решения задач классификации. Он вычисляет взвешенную сумму входов и на основании функции активации принимает решение о принадлежности объекта к классу.

Пример. Пусть требуется определить, является ли сетевое событие подозрительным. Входами могут быть количество неуспешных входов, число соединений за короткий интервал и объём переданных данных. Персептрон формирует взвешенную комбинацию признаков и выдаёт решение.

Однослойный персептрон способен решать линейно разделимые задачи. Классический пример ограничения — функция XOR: её классы нельзя разделить одной прямой в двумерном пространстве. Для моделирования нелинейных зависимостей используются многослойные сети.

Слои нейронной сети

Нейроны объединяются в слои. Входной слой получает признаки объекта. Один или несколько скрытых слоёв выполняют промежуточные преобразования. Выходной слой формирует прогноз.

Слой	**Функция**	**Пример**
Входной	Принимает признаки	Параметры сетевого потока
Скрытый	Формирует промежуточные представления	Комбинации признаков поведения
Выходной	Формирует итоговый результат	normal / attack или вероятности классов

Если сеть содержит несколько скрытых слоёв, говорят о глубокой нейронной сети. Термин «глубина» связан прежде всего с количеством последовательных уровней преобразования данных.

Многослойный персептрон MLP

Multilayer Perceptron (***MLP***) — многослойная сеть прямого распространения. Каждый нейрон слоя получает значения от предыдущего слоя, вычисляет взвешенную сумму и применяет функцию активации.

MLP может моделировать нелинейные зависимости благодаря скрытым слоям и нелинейным функциям активации. Он применяется к задачам классификации и регрессии, особенно когда данные представлены фиксированным набором числовых признаков.

Функции активации

Если каждый слой выполнял бы только линейное преобразование, последовательность нескольких таких слоёв всё равно могла бы быть сведена к одному линейному преобразованию. Нелинейные функции активации позволяют сети описывать более сложные зависимости.

Функция	Выражение / идея	Особенность
Ступенчатая	Выдаёт одно из дискретных значений	Исторически используется в простых персептронах
Sigmoid	1 / (1 + e^(-z))	Выход от 0 до 1
tanh	Гиперболический тангенс	Выход от -1 до 1
ReLU	max(0, z)	Широко используется в скрытых слоях глубоких сетей
Softmax	Преобразует набор значений в распределение	Часто используется на выходе многоклассовой классификации

Прямое распространение

При прямом распространении (forward propagation) данные последовательно проходят от входного слоя через скрытые слои к выходу. Каждый слой вычисляет свои значения на основе результатов предыдущего слоя.

Например, сеть получает четыре признака сетевого соединения. Первый скрытый слой формирует промежуточные комбинации этих признаков, второй слой преобразует их далее, а выходной слой выдаёт вероятность принадлежности соединения к классу атаки.

Функция потерь

Для обучения необходимо количественно измерить расхождение между прогнозом модели и правильным ответом. Эту роль выполняет функция потерь (loss function).

Задача	Пример функции потерь	Смысл
Бинарная классификация	Binary cross-entropy	Сравнивает прогноз вероятности с меткой 0/1
Многоклассовая классификация	Categorical / sparse categorical cross-entropy	Оценивает ошибку распределения по классам
Регрессия	Mean Squared Error	Штрафует квадрат отклонения числового прогноза

Обучение нейронной сети

Обучение сети состоит в настройке весов и смещений на обучающей выборке. Для каждого набора входов выполняется прямое распространение, вычисляется ошибка, после чего определяется, как следует изменить параметры, чтобы уменьшить функцию потерь.

Этот процесс повторяется многократно. Один полный проход по обучающей выборке называется эпохой. Данные обычно подаются небольшими группами — mini-batches.

Алгоритм обратного распространения ошибки

Backpropagation — алгоритм вычисления градиентов функции потерь по параметрам многослойной нейронной сети. Ошибка распространяется от выходного слоя к предыдущим слоям, что позволяет определить вклад отдельных весов в итоговую ошибку.

Концептуальная последовательность выглядит так:

1. Выполнить прямое распространение и получить прогноз.
2. Вычислить функцию потерь.
3. Определить градиенты функции потерь относительно параметров выходного слоя.
4. По правилу цепочки вычислить градиенты для предыдущих слоёв.
5. Передать рассчитанные градиенты оптимизатору для обновления параметров.
6. Повторять процесс на следующих mini-batches и эпохах.

Backpropagation не является самостоятельным оптимизатором. Он вычисляет градиенты; изменение параметров выполняет выбранный алгоритм оптимизации.

Градиентный спуск

Градиент показывает направление наиболее быстрого увеличения функции. Чтобы уменьшать ошибку, параметры изменяются в противоположном направлении:

$$w_new = w_old - \eta \cdot \partial L/\partial w$$

Здесь η — learning rate, или скорость обучения. Слишком большое значение может сделать обучение нестабильным, а слишком малое — существенно замедлить его.

Оптимизаторы

Оптимизатор	**Основная идея**
SGD	Обновление параметров по градиенту на отдельных mini-batches
Momentum	Добавляет накопленное направление предыдущих обновлений
RMSprop	Адаптирует масштаб шага для разных параметров
Adam	Сочетает адаптивные оценки моментов градиента и широко используется на практике

Выбор оптимизатора и learning rate относится к настройке модели. Один и тот же оптимизатор не гарантирует лучший результат для всех задач.

Эпоха, batch и итерация

Термин	**Определение**
Epoch	Один полный проход по обучающему набору
Batch	Часть обучающей выборки, обрабатываемая за один шаг
Batch size	Количество объектов в batch
Iteration	Один шаг обновления параметров на batch

Например, если обучающая выборка содержит 10 000 объектов, а batch size равен 100, то одна эпоха содержит примерно 100 итераций.

Недообучение и переобучение

Недообучение (underfitting) возникает, когда модель недостаточно хорошо описывает даже обучающие данные. Причинами могут быть слишком простая архитектура, недостаточное время обучения или неподходящие признаки.

Переобучение (overfitting) возникает, когда модель слишком хорошо подстраивается под обучающую выборку, но хуже работает на новых данных. В загруженных материалах проблемы недообучения, переобучения и валидации выделены как важная часть обучения нейронных сетей.

Состояние	**Train**	**Validation/Test**	**Интерпретация**
Недообучение	Ошибка высокая	Ошибка высокая	Модель не выучила закономерность
Хорошее обобщение	Ошибка низкая	Ошибка сопоставимо низкая	Модель переносит знания на новые данные

Переобучение	Ошибка очень низкая	Ошибка заметно выше	Модель чрезмерно подстроилась под train

Методы уменьшения переобучения

Для повышения способности модели к обобщению применяются различные подходы: увеличение и улучшение обучающих данных, регуляризация, Dropout, ранняя остановка (Early Stopping), уменьшение избыточной сложности сети и контроль качества на валидационной выборке.

Метод	**Идея**
L1/L2-регуляризация	Добавляет штраф за слишком большие параметры
Dropout	Во время обучения временно исключает часть нейронов
Early Stopping	Останавливает обучение, когда качество на validation перестаёт улучшаться
Больше качественных данных	Снижает риск запоминания отдельных объектов
Упрощение архитектуры	Сокращает избыточную ёмкость модели

Глубокое обучение

Глубокое обучение (Deep Learning) — направление машинного обучения, основанное на многослойных нейронных сетях. Глубокие модели способны последовательно формировать всё более сложные представления исходных данных.

Для изображения ранние уровни могут реагировать на простые локальные структуры, а последующие — формировать более сложные признаки. Для текста и последовательностей глубокие архитектуры обучаются представлениям, отражающим контекст и зависимости между элементами.

Основные архитектуры глубокого обучения

Архитектура	**Тип данных / задача**	**Основная идея**
MLP	Табличные и фиксированные числовые признаки	Полносвязные слои
CNN	Изображения и пространственные данные	Локальные свёрточные фильтры
RNN	Последовательности	Передача состояния между шагами
LSTM / GRU	Длинные последовательности	Механизмы управления памятью
Transformer	Текст и другие последовательности	Механизм внимания

Загруженное пособие по ИИ включает MLP и backpropagation в главу о нейронных сетях, а другой учебный материал дополнительно рассматривает CNN, RNN, LSTM и GRU при обучении нейронных сетей.

Свёрточные нейронные сети CNN

CNN применяют свёрточные операции, позволяющие обнаруживать локальные закономерности. Особенно широко они используются при обработке изображений. В задачах информационной безопасности свёрточные модели также могут применяться к специально сформированным матрицам признаков, последовательностям сигналов и другим структурированным представлениям.

Основные компоненты CNN — свёрточные слои, функции активации, операции уменьшения пространственного размера и итоговые слои классификации.

Рекуррентные сети RNN, LSTM и GRU

RNN предназначены для последовательных данных и используют информацию о предыдущих шагах. Это делает их применимыми к временным рядам, тексту, речи и последовательностям событий.

Обычные RNN испытывают трудности при моделировании длинных зависимостей. Архитектуры LSTM и GRU вводят специальные механизмы управления состоянием, позволяющие лучше сохранять существенную информацию на более длинных интервалах.

Нейронные сети в информационной безопасности

Задача	**Возможные входные данные**	**Возможная модель**
Классификация сетевого трафика	Параметры потоков	MLP, CNN
Обнаружение последовательных аномалий	Последовательности событий	RNN, LSTM, GRU
Анализ вредоносных файлов	Статические/динамические признаки	MLP, CNN
Анализ текстов сообщений и журналов	Текстовые последовательности	RNN/LSTM, Transformer

Пример. Для классификации сетевого трафика MLP может получать duration, packets, bytes, failed_logins и другие числовые признаки. Выходной слой формирует вероятность классов normal и attack. Качество оценивается на данных, не использованных при настройке параметров.

TensorFlow и Keras

TensorFlow — программная платформа для построения, обучения и применения моделей машинного и глубокого обучения. Keras предоставляет высокоуровневый интерфейс, позволяющий компактно описывать архитектуру нейронной сети.

В ранее загруженной лекции по инструментам ИИ TensorFlow рассматривается как платформа для операций с тензорами и обучения моделей, а Keras — как высокоуровневый интерфейс создания нейронных сетей. Это соответствует структуре учебного пособия, где TensorFlow и Keras включены в главу о нейронных сетях и глубоком обучении.

Пример нейронной сети в TensorFlow/Keras

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Input(shape=(10,)),
    layers.Dense(32, activation='relu'),
    layers.Dense(16, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

model.compile(
```

```
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=30,
    batch_size=32
)

test_loss, test_accuracy = model.evaluate(X_test, y_test)
```

Разбор примера

Строка / элемент	Назначение
Input(shape=(10,))	Модель получает 10 входных признаков
Dense(32, relu)	Первый скрытый полносвязный слой
Dense(16, relu)	Второй скрытый слой
Dense(1, sigmoid)	Вероятность положительного класса
optimizer='adam'	Алгоритм обновления параметров
binary_crossentropy	Функция потерь бинарной классификации
validation_split=0.2	Часть train используется для контроля обобщения
epochs=30	До 30 проходов по обучающим данным
batch_size=32	32 объекта на один шаг

Пример многоклассовой классификации

```
model = keras.Sequential([
    layers.Input(shape=(20,)),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(4, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Если необходимо различать четыре категории событий, выходной слой содержит четыре нейрона. Softmax формирует распределение оценок по классам, а класс с наибольшей оценкой может быть выбран как итоговый прогноз.

Как анализировать процесс обучения

Во время обучения необходимо отслеживать не только итоговую метрику на test, но и динамику loss и метрик на train и validation. Если train loss продолжает снижаться, а validation loss начинает расти, это типичный признак возможного переобучения.

При сравнении моделей важно использовать одинаковое разбиение данных и не подбирать архитектуру по тестовой выборке. Test должен оставаться независимым источником финальной оценки.

Преимущества и ограничения нейронных сетей

Преимущества	**Ограничения**
Моделируют сложные нелинейные зависимости	Могут требовать много данных
Работают с изображениями, текстом, речью и последовательностями	Обучение может быть вычислительно затратным
Позволяют автоматически формировать представления признаков	Настройка архитектуры и гиперпараметров требует экспериментов
Поддерживают transfer learning	Глубокие модели часто сложнее интерпретировать

Типичные ошибки при разработке нейронной сети

– Считать увеличение числа слоёв гарантией повышения качества.

– Оценивать модель только на обучающей выборке.

– Использовать test для постоянного подбора архитектуры и гиперпараметров.

– Не масштабировать числовые признаки, когда это необходимо для устойчивого обучения.

– Выбирать функцию активации и функцию потерь без учёта типа задачи.

– Игнорировать дисбаланс классов.

– Считать высокую accuracy достаточной для задачи обнаружения редких атак.

– Продолжать обучение после явного ухудшения качества на validation без анализа переобучения.

Сравнение классического машинного и глубокого обучения

Критерий	**Классическое ML**	**Deep Learning**
Признаки	Часто проектируются вручную	Могут обучаться внутри сети
Данные	Эффективно и на относительно небольших табличных наборах	Часто выигрывает при больших объёмах сложных данных
Вычисления	Обычно умеренные	Могут требовать GPU и значительных ресурсов
Интерпретируемость	Часто выше	Часто ниже
Изображения/речь/текст	Нужна специальная подготовка признаков	Глубокие архитектуры особенно востребованы

Основные выводы:

– Искусственный нейрон вычисляет взвешенную сумму входов и применяет функцию активации.

– Персептрон является простейшей нейросетевой моделью и ограничен линейной разделимостью.

– Многослойные сети используют скрытые слои и нелинейные функции активации.

– Обучение включает прямое распространение, вычисление функции потерь, backpropagation и обновление параметров.

– Learning rate, batch size, число эпох и архитектура являются важными параметрами процесса обучения.

– Недообучение и переобучение контролируются по обучающим и валидационным данным.

– Глубокое обучение использует многослойные архитектуры, включая MLP, CNN, RNN/LSTM/GRU и Transformer.

– TensorFlow и Keras позволяют реализовывать и обучать нейронные сети средствами Python.

Контрольные вопросы:

1. Что такое искусственная нейронная сеть?
2. Из каких элементов состоит искусственный нейрон?
3. Какую роль выполняют веса?
4. Для чего используется смещение b?
5. Что такое функция активации?
6. Что представляет собой персептрон?
7. Что означает линейная разделимость?
8. Почему однослойный персептрон не решает XOR?
9. Какие слои содержит многослойная сеть?
10. Что такое MLP?
11. Зачем нейронной сети нелинейные функции активации?
12. Чем sigmoid отличается от tanh?
13. Как работает ReLU?
14. Для чего используется Softmax?
15. Что такое forward propagation?
16. Что такое функция потерь?
17. Какие функции потерь применяются для классификации и регрессии?
18. Что такое эпоха?
19. Что такое batch и batch size?
20. Что такое backpropagation?
21. Какую роль играет правило цепочки?
22. Чем backpropagation отличается от оптимизатора?
23. Что такое градиентный спуск?
24. Как learning rate влияет на обучение?
25. Что такое Adam?
26. Чем недообучение отличается от переобучения?
27. Как обнаружить переобучение по train и validation?

Список источников

1. Учебное пособие «Основы искусственного интеллекта». Глава 8 «Нейронные сети и глубокое обучение»: персептрон, MLP, Backpropagation, TensorFlow и Keras.

2. Искусственный интеллект и машинное обучение. Учебно-методические материалы. Разделы «Искусственные нейронные сети», «Персептроны», «Обучение искусственных нейронных сетей»: искусственный нейрон, функции активации, однослойные и многослойные сети, обратное распространение ошибки, CNN, RNN, LSTM, GRU, недообучение и переобучение.

3. Лекция 2 «Инструменты и платформы искусственного интеллекта». Разделы TensorFlow, Keras и PyTorch.