Лекция 9

Здравствуйте, уважаемые слушатели!
Тема нашей лекции – Нейронные сети

План лекции:
1. Введение в нейронные сети
2. Основы архитектуры нейронных сетей
3. Искусственный нейрон и функции активации
4. Основные архитектуры нейронных сетей
5. Заключение

1. Введение в нейронные сети

Нейронные сети — это мощные вычислительные модели, вдохновленные биологическими нейронами, которые предназначены для решения сложных задач машинного обучения, таких как классификация изображений, обработка естественного языка и прогнозирование временных рядов. Современные нейронные сети стали основой для глубокого обучения, где они используются для построения сложных архитектур с множеством слоев. Основной идеей нейронных сетей является обучение на основе предоставленных данных, что позволяет сети выявлять сложные взаимосвязи и зависимости.

Эффективность нейронных сетей обеспечивается способностью адаптироваться к данным, автоматически обучаясь на входных данных и оптимизируя свои параметры для улучшения предсказаний. Они состоят из большого количества искусственных нейронов, объединенных в слои, каждый из которых передает информацию следующему слою, что делает нейронные сети особенно эффективными для решения нелинейных задач.

2. Основы архитектуры нейронных сетей

Архитектура нейронной сети определяет ее структуру и взаимодействие между нейронами. Основные элементы нейронных сетей включают:

- **Входной слой**: Первый слой, принимающий данные для обработки. Количество нейронов в этом слое соответствует количеству признаков во входных данных.
- **Скрытые слои**: Слои, находящиеся между входным и выходным, в которых происходит обработка и обучение сети. Эти слои могут быть линейными или нелинейными в зависимости от функции активации, и их количество определяет глубину сети.
- **Выходной слой**: Последний слой, который выдает итоговое предсказание сети. Его структура зависит от задачи — например, один нейрон для бинарной классификации и несколько нейронов для многоклассовой.

Важнейшими параметрами нейронных сетей являются веса и смещения, которые обучаются на данных и играют ключевую роль в передаче информации между нейронами. Основная цель обучения — оптимизировать веса и смещения так, чтобы сеть минимизировала ошибку на тестовой выборке.

3. Искусственный нейрон и функции активации

Искусственный нейрон, или персептрон, является основным строительным блоком нейронной сети. Каждый нейрон принимает один или несколько входов, умножает их на соответствующие веса, добавляет смещение и применяет функцию активации для получения результата. В общем виде модель нейрона выглядит так:

y=f(∑i=1nwixi+b)y = f \left(\sum_{i=1}^n w_i x_i + b \right)y=f(i=1∑nwixi+b)

где:

- wiw_iwi — веса,
- xix_ixi — входные данные,
- bbb — смещение,
- fff — функция активации, которая придает нейрону нелинейность.

3.1 Функции активации

Функции активации играют ключевую роль в нейронных сетях, поскольку они придают модели способность обучаться нелинейным зависимостям. Наиболее распространенные функции активации:

- **Сигмоидная функция**: Принимает значения от 0 до 1 и полезна для задач бинарной классификации.

σ(x)=11+e−x\sigma(x) = \frac{1}{1 + e^{-x}}σ(x)=1+e−x1

- **Гиперболический тангенс (tanh)**: Принимает значения от -1 до 1, часто используется для уменьшения градиентного затухания.

tanh(x)=ex−e−xex+e−xtanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}tanh(x)=ex+e−xex−e−x

- **ReLU (Rectified Linear Unit)**: Переход к 0 для отрицательных значений и линейная зависимость для положительных. ReLU является популярной функцией в современных нейронных сетях, так как помогает справляться с проблемой затухающих градиентов.

ReLU(x)=max(0,x)ReLU(x) = \max(0, x)ReLU(x)=max(0,x)

- **Leaky ReLU**: Вариант ReLU, который позволяет небольшие отрицательные значения, помогая справляться с проблемой «мертвых нейронов».

LeakyReLU(x)=max(0.01x,x)Leaky ReLU(x) = \max(0.01x, x)LeakyReLU(x)=max(0.01x,x)

4. Обучение нейронной сети

Обучение нейронной сети представляет собой процесс настройки параметров (весов и смещений) для минимизации функции потерь. Этот процесс включает несколько ключевых компонентов:

4.1 Функция потерь

Функция потерь измеряет, насколько близко предсказания сети к истинным значениям. Наиболее распространенные функции потерь:

- **Среднеквадратическая ошибка (Mean Squared Error, MSE)**: Используется для задач регрессии.

MSE=1n∑i=1n(yi−y^i)2MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2MSE=n1i=1∑n(yi−y^i)2

- **Бинарная кросс-энтропия**: Для задач бинарной классификации.

L=−1n∑i=1nyilog(y^i)+(1−yi)log(1−y^i)L = - \frac{1}{n} \sum_{i=1}^n y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)L=−n1i=1∑nyilog(y^i)+(1−yi)log(1−y^i)

- **Категориальная кросс-энтропия**: Применяется для многоклассовых задач.

L=−∑i=1n∑j=1kyijlog(y^ij)L = - \sum_{i=1}^n \sum_{j=1}^k y_{ij} \log(\hat{y}_{ij})L=−i=1∑nj=1∑kyijlog(y^ij)

4.2 Метод обратного распространения ошибки

Метод обратного распространения ошибки (backpropagation) используется для вычисления градиентов функции потерь относительно параметров сети. Этот алгоритм включает два основных шага:

1. **Прямое распространение**: Входные данные проходят через сеть для получения предсказания.
2. **Обратное распространение**: Градиенты функции потерь вычисляются по отношению к весам и смещениям, начиная с выходного слоя и двигаясь к входному.

Обратное распространение совместно с методом градиентного спуска позволяет нейронной сети корректировать свои параметры в направлении уменьшения функции потерь.

4.3 Градиентный спуск

Градиентный спуск — это метод оптимизации, используемый для минимизации функции потерь путем корректировки параметров сети. Основные варианты градиентного спуска:

- **Пакетный градиентный спуск (Batch Gradient Descent)**: Использует весь тренировочный набор данных для вычисления градиента.
- **Стохастический градиентный спуск (Stochastic Gradient Descent, SGD)**: Обновляет параметры на каждом образце, что увеличивает скорость, но делает процесс обучения менее стабильным.
- **Мини-пакетный градиентный спуск (Mini-batch Gradient Descent)**: Компромисс между пакетным и стохастическим методами, использует небольшие подмножества данных для обновления параметров.

5. Основные архитектуры нейронных сетей

Существует множество архитектур нейронных сетей, каждая из которых подходит для определенных задач. Рассмотрим наиболее популярные из них.

5.1 Полносвязные нейронные сети (Fully Connected Networks, FCN)

Полносвязные сети состоят из слоев, каждый нейрон которых соединен с каждым нейроном следующего слоя. Полносвязные сети эффективны для задач, где входные данные имеют фиксированную размерность. Однако они требуют значительных вычислительных ресурсов и могут легко переобучаться на больших наборах данных.

5.2 Сверточные нейронные сети (Convolutional Neural Networks, CNN)

CNN используются в задачах, связанных с изображениями и видеоданными. В отличие от полносвязных сетей, CNN применяют операции свертки, которые извлекают пространственные зависимости в данных, делая их эффективными для обработки двумерных и трехмерных данных.

Основные элементы CNN:

- **Сверточные слои**: Применяют фильтры к входным данным, что позволяет выделять особенности, такие как края и текстуры.
- **Pooling слои**: Уменьшают размерность данных, что помогает сократить вычислительные ресурсы и предотвратить переобучение.

- **Полносвязные слои**: Используются в конце сети для классификации извлеченных признаков.

5.3 Рекуррентные нейронные сети (Recurrent Neural Networks, RNN)

RNN эффективны для анализа последовательных данных, таких как текст и временные ряды. В отличие от CNN, RNN имеют механизмы памяти, что позволяет учитывать предыдущие элементы последовательности. Однако стандартные RNN страдают от проблемы затухающих и взрывных градиентов.

Популярные модификации RNN:

- **LSTM (Long Short-Term Memory)**: Содержит ячейки памяти, которые позволяют сохранять долгосрочные зависимости.
- **GRU (Gated Recurrent Unit)**: Упрощенный вариант LSTM, который также решает проблему долгосрочной зависимости, но требует меньше вычислительных ресурсов.

5.4 Генеративно-состязательные сети (Generative Adversarial Networks, GAN)

GAN состоят из двух сетей: генератора, который создает новые данные, и дискриминатора, который пытается отличить реальные данные от созданных. GAN используются в задачах генерации изображений, преобразования стилей и создания искусственных данных. Основная идея GAN заключается в конкуренции между генератором и дискриминатором, что позволяет обучить генератор создавать реалистичные данные.

6. Регуляризация и предотвращение переобучения

Нейронные сети подвержены переобучению, особенно при работе с большими объемами данных и сложными архитектурами. Для решения этой проблемы используются методы регуляризации:

- **Dropout**: Исключает случайные нейроны во время обучения, снижая зависимость сети от определенных нейронов.
- **L2-регуляризация**: Добавляет штраф на веса, чтобы предотвратить их избыточное увеличение.
- **Ранняя остановка (Early Stopping)**: Останавливает обучение, если ошибка на валидационном наборе начинает увеличиваться, что указывает на переобучение.

7. Применение нейронных сетей

Нейронные сети находят широкое применение в самых разных областях:

- **Компьютерное зрение**: Обработка изображений, распознавание лиц и объектов, медицинская диагностика.
- **Обработка естественного языка**: Машинный перевод, классификация текста, анализ тональности.
- **Прогнозирование временных рядов**: Прогнозирование цен на финансовых рынках, анализ временных данных в метеорологии.
- **Генерация контента**: Создание изображений, генерация текста, преобразование стилей.

8. Заключение

Нейронные сети представляют собой фундаментальный инструмент в машинном обучении и глубоких нейронных архитектурах. Способность обучаться сложным паттернам и обрабатывать большие объемы данных делает нейронные сети незаменимыми для многих современных приложений.

Литературы:

1. Машинное обучение: основы, алгоритмы и практика применения, Уатт Дж. 2022 стр. 76-85
2. Прикладное машинное обучение и искусственный интеллект для инженеров, Просиз Джеф - 2023 стр. 75-86