Нейросети: основы нейронных сетей, архитектуры нейронных сетей

Цель занятия

После освоения темы:

- вы познакомитесь с терминологией, используемой в нейросетях, и архитектурами нейронных сетей;
- узнаете основные библиотеки Python, используемые для работы с нейросетями;
- сможете написать программу для вычисления результата сигмоидальной функции активации для заданных входных данных;
- сможете написать код, реализующий свертку изображения и фильтра;
- сможете реализовать простую нейросеть, состоящую из одного входного слоя, одного скрытого слоя и одного выходного слоя.

План занятия

Используемые термины

Нейронная сеть — алгоритм машинного обучения, который использует математические функции, моделирующие работу мозга, чтобы извлекать закономерности из данных.

Нейрон в нейронной сети — элементарная единица обработки информации, которая принимает входные данные, обрабатывает их и генерирует выходные данные.

Глубокое обучение (Deep Learning) — подмножество нейронных сетей, которые имеют множество скрытых слоев и позволяют извлекать более сложные закономерности из данных.

Сверточная нейросеть (Convolutional Neural Network, CNN) — одна из самых популярных и мощных архитектур нейронных сетей, применяемых в обработке изображений, видео и звука.

Рекуррентные сверточные нейронные сети (Recurrent Convolutional Neural Network, RCNN) — комбинация сверточных и рекуррентных слоев, которая позволяет моделировать пространственные и временные зависимости в данных.

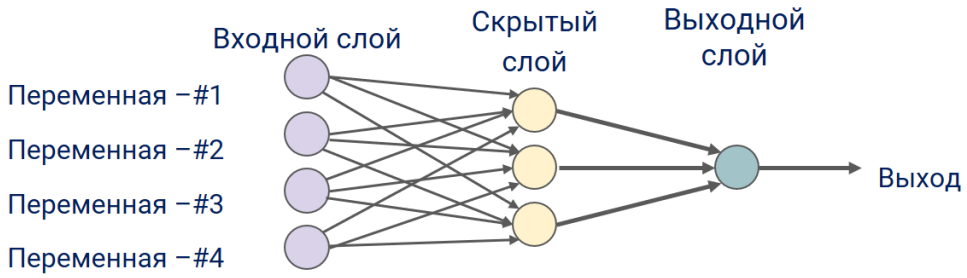
Конспект занятия

1. Основы нейронных сетей

Нейронная сеть — это алгоритм машинного обучения, который использует математические функции, моделирующие работу мозга, чтобы извлекать закономерности из данных.

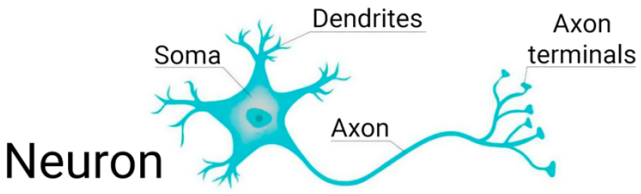
Нейронные сети состоят из нейронов, которые получают входные данные, производят вычисления и передают результаты следующим нейронам.

Нейроны объединены в слои, где каждый слой выполняет определенные вычисления над данными подобно разделам мозга человека.

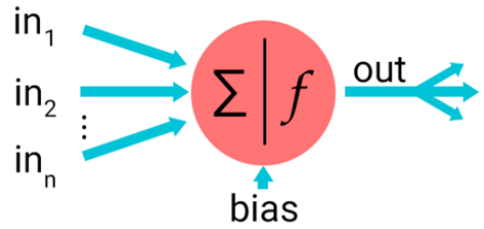


Нейрон в нейронной сети — это элементарная единица обработки информации, которая принимает входные данные, обрабатывает их и генерирует выходные данные.

Нейрон назван так в честь биологической нейронной системы, которая обрабатывает информацию в головном мозге живых организмов. Биологический нейрон состоит из тела клетки и дендритов. Они принимают входные сигналы от других нейронов и аксонов, которые передают выходные сигналы другим нейронам.



Идея использования нейронов в нейронных сетях заключается в том, чтобы имитировать работу биологических нейронов и использовать их для обработки информации. Каждый нейрон в нейронной сети принимает входные данные от других нейронов через свои входы и выполняет вычисления, используя веса. Затем передает выходные данные другим нейронам через свои выходы.



В одном слое нейросети может быть много нейронов, потому что каждый нейрон выполняет свою функцию в обработке входных данных.

Множество нейронов позволяет производить следующие функции:

- Извлечение различных признаков и их совместная обработка.
- Изучение более сложных закономерностей в данных.

  Каждый нейрон может быть настроен на обнаружение в данных определенных признаков (цвет, текстура, форма или ориентация) и обработку данных совместно с другими нейронами.
- Повышение точности предсказаний нейросети.

  Каждый нейрон может вносить свой вклад в принятие решений.

Нейросети отличаются от других алгоритмов машинного обучения тем, что могут обучаться на неструктурированных данных — изображении, аудио или тексте.

Deep Learning (глубокое обучение) — это подмножество нейронных сетей, которые имеют множество скрытых слоев и позволяют извлекать более сложные закономерности из данных.

Deep Learning обычно требует больших объемов данных и вычислительных ресурсов для обучения. Но может достичь значительных результатов в областях компьютерного зрения и естественного языка.

Библиотеки Python для работы с нейросетями:

- **TensorFlow** — одна из самых популярных и используемых библиотек для обучения нейросетей. TensorFlow позволяет создавать и обучать нейросети любой сложности, включая глубокое обучение.
- **Keras** — это высокоуровневая библиотека для обучения нейросетей, которая позволяет быстро создавать и обучать модели. Keras имеет простой и интуитивно понятный интерфейс, может использоваться с различными бэкендами.

- **PyTorch** — библиотека, которая позволяет создавать и обучать нейросети глубокого обучения. PyTorch имеет динамический граф вычислений, что позволяет создавать модели более гибко и эффективно.
- **Caffe** — это библиотека, которая была разработана для обработки изображений и видео. Caffe имеет оптимизированные функции для сверточных нейронных сетей и может работать с большими объемами данных.

2. Архитектуры нейронных сетей

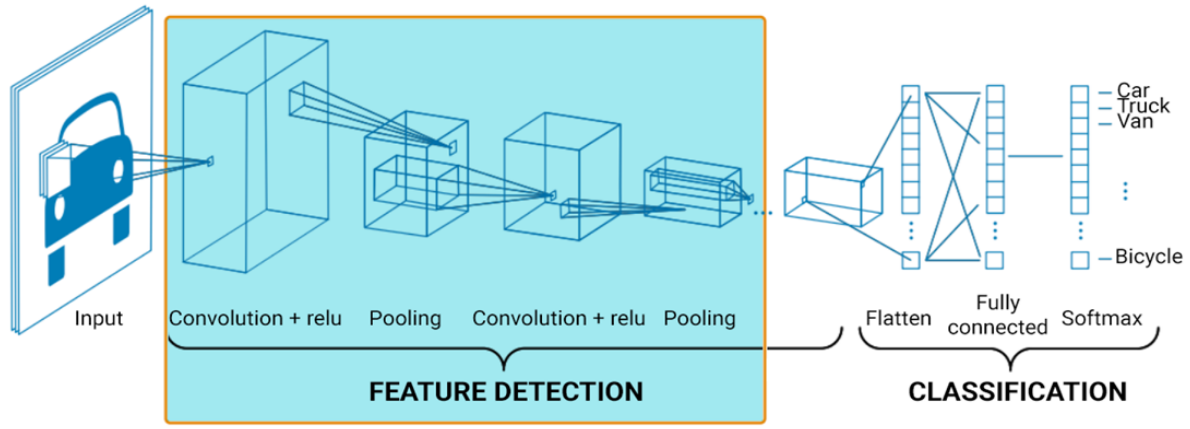
Сверточные сети

Сверточная нейросеть (Convolutional Neural Network, CNN) — одна из самых популярных и мощных архитектур нейронных сетей, применяемых в обработке изображений, видео и звука.

Ее главная идея: использовать свойства локальности и инвариантности для более эффективной обработки изображений и других типов данных, имеющих структуру.

Архитектура сверточной нейросети включает несколько типов слоев:

- слой входных данных (Input layer);
- сверточный слой (Convolutional layer);
- слой активации (Activation layer);
- слой объединения (Pooling layer);
- полносвязный слой (Fully Connected layer);
- слой нормализации (Normalization layer);
- слой отсева (Dropout layer).



Обзор основных слоев:

Слой	**Функция**	**Параметры**	**Входные данные**	**Выходные данные**
Входной слой	Принимает входные данные	Размер входных данных	Изображение или другие данные	Изображение или другие данные
Сверточный слой	Выделяет признаки изображения	Количество фильтров, размер фильтров, шаг свертки	Изображение	Карта признаков
Слой активации	Применяет нелинейность	Функция активации	Карта признаков	Карта признаков
Слой объединения	Уменьшает размерность	Размер объединенного окна, шаг объединения	Карта признаков	Уменьшенная карта признаков
Полно-связный слой	Классифицирует признаки	Количество нейронов	Вектор признаков	Вектор выходных значений
Слой нормали-зации	Нормализует данные	Тип и параметры нормализации	Вектор признаков	Нормализованный вектор признаков

Слой отсева	Уменьшает переобучение	Вероятность отключения нейронов	Вектор признаков	Вектор признаков

Свойство локальности: нейросеть способна обрабатывать данные, фокусируясь на небольших участках данных (например, на пикселях изображения), а не на данных в целом. Это позволяет нейросетям обнаруживать локальные шаблоны и особенности: края, формы, текстуры и т. д.

Свойство инвариантности: нейросеть распознает объекты, даже если они немного изменены — повернуты, масштабированы или искажены. Это позволяет нейросетям обобщать знания о классификации объектов на более широкий набор данных, что делает их более универсальными и применимыми на практике.

Применение сверточных сетей:

- классификация изображений;
- детектирование объектов;
- сегментация изображений;
- распознавание лиц;
- определение эмоций;
- обработка естественного языка;
- распознавание речи и т. д.

Преимущества:

- эффективная обработка больших объемов данных;
- способность к автоматическому извлечению признаков;
- обучение без учителя и возможность распараллеливания для ускорения обучения .

Недостатки:

- сложность в настройке параметров модели;
- требование большого количества данных для обучения.

Пример задачи свертки. Предположим, что у нас есть изображение размером 5x5 пикселей и фильтр (kernel) размером 3x3 пикселя. Мы хотим применить свертку между изображением и фильтром, чтобы получить карту признаков (feature map) размером 3x3.

Для выполнения свертки мы размещаем фильтр на изображении и перемещаем его по всем позициям, применяя ко всем пикселям изображения.

На каждой позиции мы производим операцию умножения элементов фильтра на соответствующие пиксели изображения, а затем суммируем результаты. Затем это значение записывается в позицию карты признаков, соответствующую позиции фильтра на изображении.

Для иллюстрации работы алгоритма приведем код. Решать задачу свертки мы будем с помощью библиотеки `NumPy`:

```
import numpy as np
```

Задаем изображение и фильтр:

```
image = np.array([[1, 2, 0, 0, 3],
                  [0, 1, 2, 3, 1],
                  [0, 0, 1, 2, 3],
                  [3, 2, 1, 2, 0],
                  [1, 0, 3, 1, 2]])
filter = np.array([[1, 0, 1],
                   [0, 1, 0],
                   [1, 0, 1]])
```

Применяем свертку между изображением и фильтром:

```
feature_map = np.zeros((3, 3))
for i in range(3):
     for j in range(3):
     feature_map[i][j] = np.sum(image[i:i+3, j:j+3] * filter)
print(feature_map)
```

Рекуррентные нейросети

Рекуррентная нейросеть (Recurrent Neural Network, RNN) используется в обработке последовательных данных: текстов, временных рядов и аудиофайлов.

Идея рекуррентной нейросети заключается в том, что каждый элемент последовательности входных данных обрабатывается вместе с предыдущими элементами с использованием общих весовых коэффициентов.

Архитектура рекуррентной нейросети включает один или несколько слоев, которые обеспечивают обработку последовательных данных и обновляют скрытое состояние на каждом шаге.

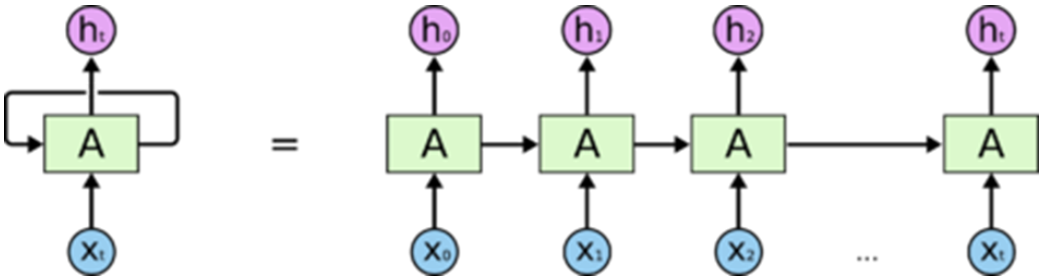
В RNN используется вектор состояния, который передается через все элементы последовательности и служит для сохранения контекста входных данных.

Типы RNN:

- простая рекуррентная нейросеть (Simple RNN);
- LSTM (Long Short-Term Memory);
- GRU (Gated Recurrent Unit).

Основной компонент RNN — рекуррентный слой. Он состоит из повторяющихся блоков нейронов, которые передают информацию от предыдущего шага обработки к следующему. В каждом блоке используется функция активации, которая позволяет сети захватывать зависимости между последовательными элементами.

Архитектура RNN:



На диаграмме показаны четыре повторяющихся блока, но в реальности их количество может быть гораздо больше.

В каждом блоке RNN входные данные (*x*) передаются на вход скрытого слоя (*h*), а затем смешиваются с выходом предыдущего шага (*h*-1) для создания нового скрытого состояния (*h*). Затем этот выход (*h*) используется для генерации следующего выхода (*y*) и передается на следующий блок.

В RNN обычно используются два типа архитектур: **однонаправленные** и **двунаправленные**. В однонаправленных RNN информация движется только в одном направлении — от прошлого к будущему. В двунаправленных RNN информация передается и в прошлое, и в будущее, что позволяет сети захватывать более сложные зависимости в последовательностях данных.

Применение рекуррентных нейросетей:

- языковое моделирование;
- машинный перевод;
- распознавание речи;
- генерация текста;
- финансовые прогнозы;
- медицинские данные и т. д.

Преимущества:

- обработка последовательных данных различной длины;
- возможность использования предыдущего контекста для прогнозирования будущих значений;
- способность к обучению без учителя.

Недостатки:

- сложность в обучении и настройке параметров модели;

- проблемы с градиентным затуханием, которые могут возникать при обработке длинных последовательностей.

Современные архитектуры

Рассмотрим наиболее популярные архитектуры, которые применяются в современных нейросетевых продуктах.

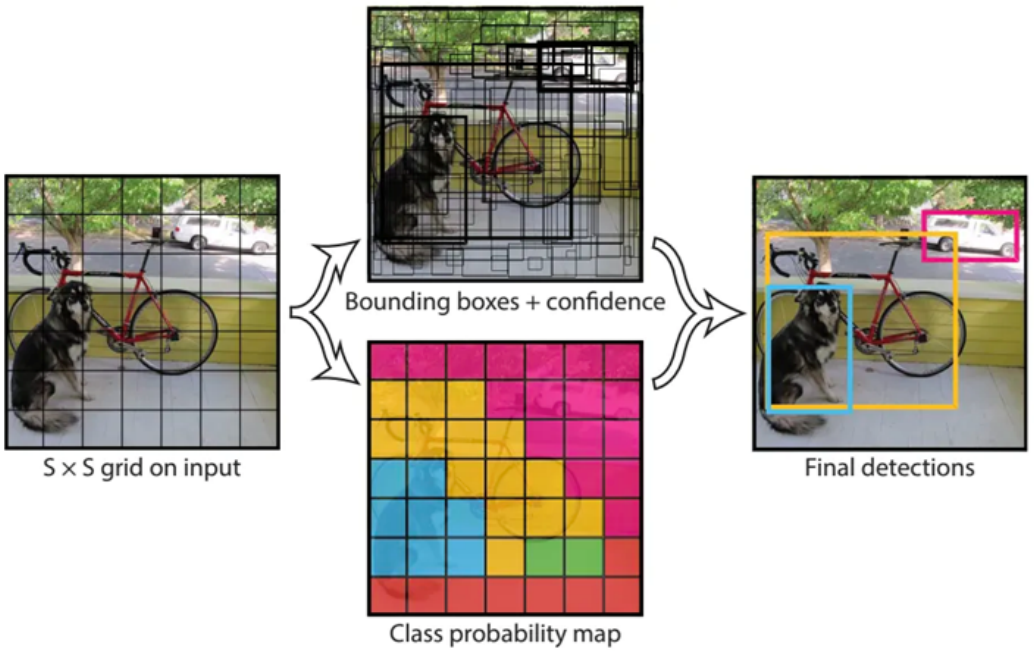
Автоэнкодер (Autoencoder) используется для обучения без учителя. Позволяет извлекать скрытые представления из данных, уменьшать размерность и восстанавливать исходные данные.

Применение:

- Сжатие изображений. В продукте Google Photos Google использует автоэнкодеры для сжатия изображений. Это позволяет сохранять на маленьких устройствах большое количество фотографий и экономить место на серверах Google.
- Устранение шума из аудио. Dolby использует автоэнкодеры в своих продуктах для устранения шума из аудиосигналов, например, в наушниках с шумоподавлением.

Рекуррентные сверточные нейронные сети (Recurrent Convolutional Neural Network, RCNN) — комбинация сверточных и рекуррентных слоев, позволяющая моделировать пространственные и временные зависимости в данных.

Применение: YOLO (You Only Look Once) — приложение для распознавания объектов в реальном времени.



Трансформер (Transformer) используется для обработки последовательностей данных: текстов и аудиофайлов. Достигает высоких результатов в задачах машинного перевода, генерации текстов и прочих.

Главным продуктам является нейросеть GPT от OpenAI — языковая модель, которая использует трансформеры для генерации текста. Она может быть использована для создания различного контента — от статей и новостных сообщений до литературы и музыки.

U-Net используется для сегментации изображений. Имеет особенную структуру, позволяющую сохранять информацию о пространственном контексте входного изображения.

Применение:

- **DeepLabCu** — программное обеспечение для обработки видео, использующее нейросеть U-Net для отслеживания движения объектов на видео и выделения их контуров.

- **Окклюзионная терапия с использованием масок в области офтальмологии** — метод лечения заболеваний глаз, который использует нейросеть U-Net, чтобы создать маски для затенения определенных областей глаза. Этот метод может быть использован в лечении заболеваний, например, амблиопии.
- **Лечение рака**. Некоторые исследования показали, что нейросеть U-Net может быть использована для обнаружения опухолей и анализа их характеристик. Это помогает улучшить диагностику и лечение рака.

Дополнительные материалы для самостоятельного изучения

1. 10 библиотек Python для машинного обучения и искусственного интеллекта - UPROGER | Программирование
2. TensorFlow
3. Keras: Deep Learning for humans
4. PyTorch
5. Caffe | Deep Learning Framework (berkeleyvision.org)
6. Kevin Rexis Velasco, YOLO (You Only Look Once)
7. Cere Labs, Understanding U-Net Architecture For Image Segmentation