Лабораторная работа

«Распознавание цифр при помощи нейронных сетей».

Содержание

Структура классического перцептрона

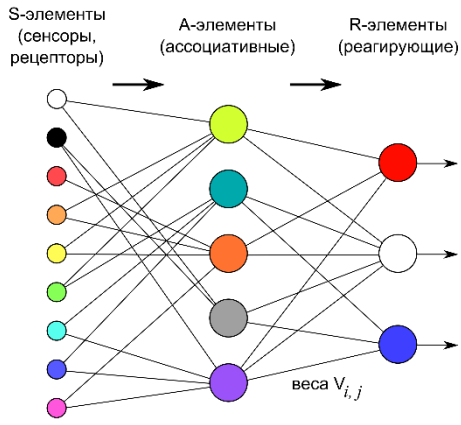
Перцептрон является одним из самых простых видов нейронных сетей, и его структура включает в себя следующие компоненты:

- **Входной слой:** слой, который принимает на вход входные данные. Эти данные могут быть представлены в виде вектора или матрицы.
- **Выходной слой:** слой, который выдает результат работы перцептрона. В зависимости от задачи, решаемой перцептроном, выходной слой может иметь различное количество нейронов.
- **Веса:** каждый нейрон в перцептроне имеет свой вес, который определяет важность входных сигналов для нейрона. Веса являются настраиваемыми параметрами перцептрона и обучаются в процессе обучения.
- **Сумматор:** компонент перцептрона, который выполняет суммирование взвешенных входных сигналов с весами.
- **Функция активации:** функция, которая нормализует выход сумматора, чтобы выдать выходной сигнал перцептрона.

Важно отметить, что перцептрон может иметь сколько угодно скрытых слоев между входным и выходным слоями, в зависимости от сложности задачи. Каждый из скрытых слоев выполняет аналогичные вычисления, используя свои веса и функции активации.

Общая формула для вычисления выходного сигнала перцептрона выглядит следующим образом:

$$y = f(w1x1 + w2x2 + ... + wnxn)$$



где:

- y - выходной сигнал перцептрона
- f - функция активации
- w - весовой коэффициент нейрона
- x - входной сигнал нейрона
- n - количество входов нейрона

Таким образом, перцептрон является простой моделью, которая может быть использована для решения различных задач машинного обучения, таких как классификация или регрессия.

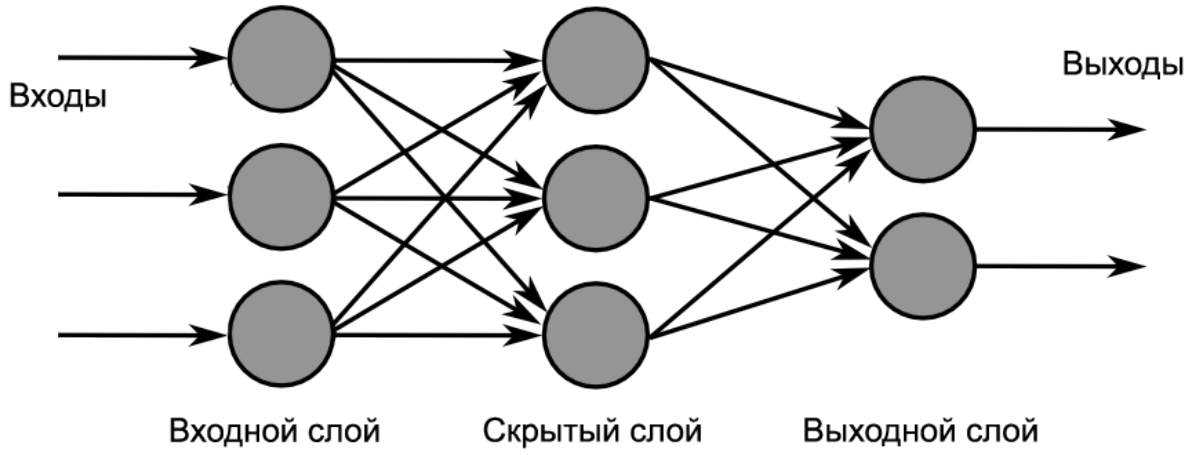
Описание процесса прямого распространения в нейронной сети

Процесс прямого распространения (forward propagation) в нейронной сети - это процесс передачи входных данных через несколько слоев нейронов, где каждый слой состоит из множества нейронов, соединенных с предыдущим слоем.

Для начала, каждый нейрон входного слоя принимает некоторое значение и передает его далее. Затем, каждый нейрон в скрытом слое получает входные данные из предыдущего слоя и выполняет линейную операцию, где каждое входное значение умножается на соответствующий вес и затем складываются сдвигом (bias). Таким образом, нейрон вычисляет взвешенную сумму входных данных, что приводит к созданию нового значения, которое передается в функцию активации.

Функция активации применяется к каждому полученному значению, чтобы вычислить выходные данные нейрона, которые передаются дальше в следующий слой. Функции активации могут быть различными, например, можно использовать сигмоиду, гиперболический тангенс, ReLU и многие другие.

Процесс повторяется для каждого слоя скрытых нейронов, пока не достигнется выходной слой. В выходном слое обычно используется функция активации softmax для получения вероятностных значений, которые представляют собой итоговый вывод модели.



В математической записи, процесс прямого распространения может быть представлен в виде формулы:

$z^l=W^l*a^{(l-1)}+b^l$

$a^l=f(z^l)$

где:

- l - номер слоя,

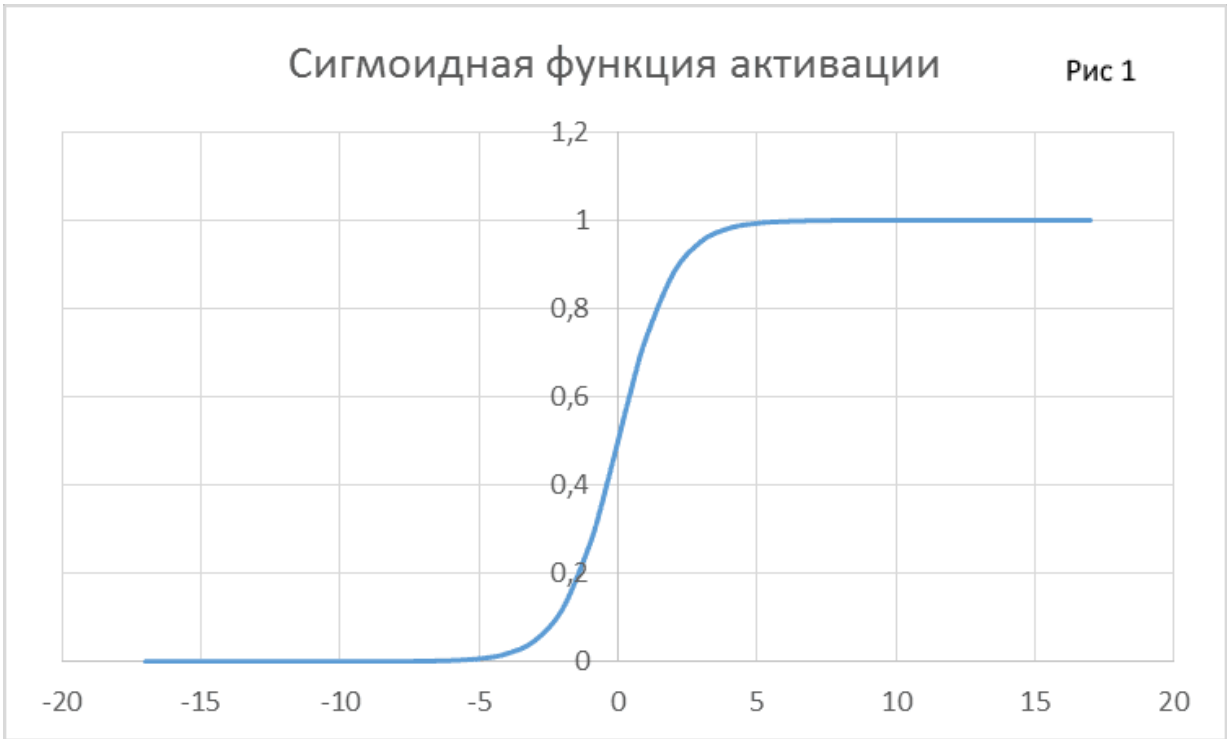
- W - матрица весов,
- a - вектор выходных значений слоя (активации),
- b - вектор смещений,
- z - вектор суммированных входных значений слоя,
- f - функция активации.

Функции активации

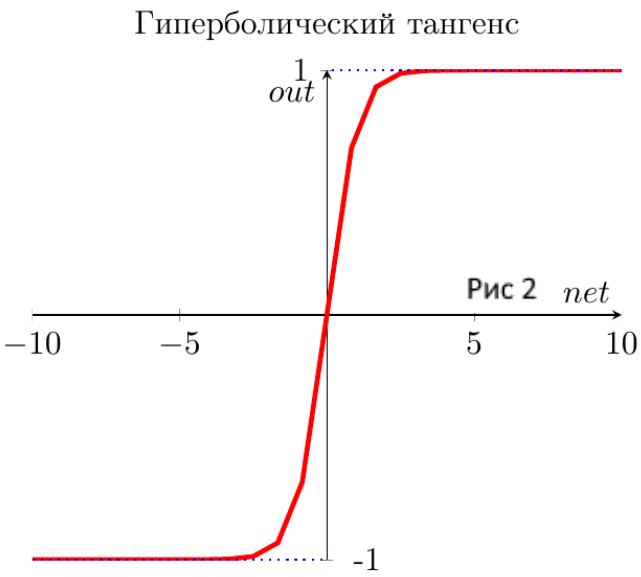
Функции активации - это математические функции, которые используются в нейронных сетях для добавления нелинейности в модель. Они применяются к результату линейного преобразования на предыдущем слое и определяют выходной сигнал нейрона. Функции активации являются необходимыми для решения более сложных задач, таких как классификация изображений или обработка естественного языка.

Вот некоторые из наиболее распространенных функций активации:

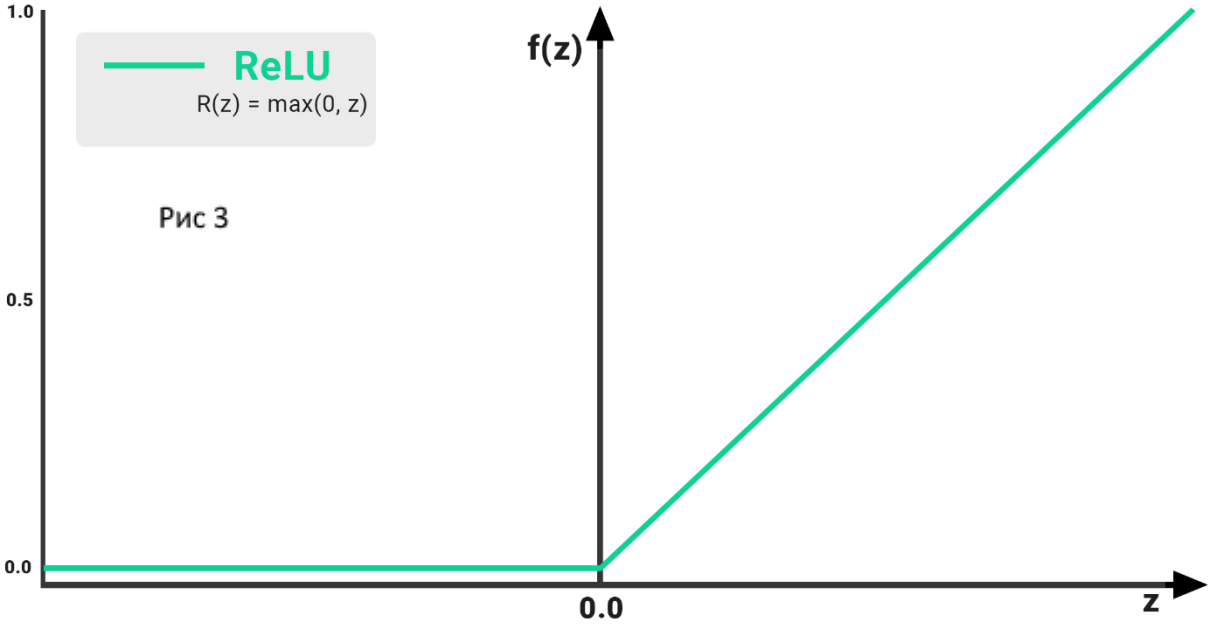
1. **(sigmoid):** *f(x) = 1 / (1 + exp(-x))*. Она выглядит как S-образная кривая и используется для бинарной классификации.



2. **Гиперболический тангенс (tanh):** *f(x) = (exp(x) - exp(-x)) / (exp(x) + exp(-x))*. Эта функция также имеет S-образную форму, но ее выходные значения находятся в диапазоне [-1, 1]. Она часто используется в задачах регрессии.



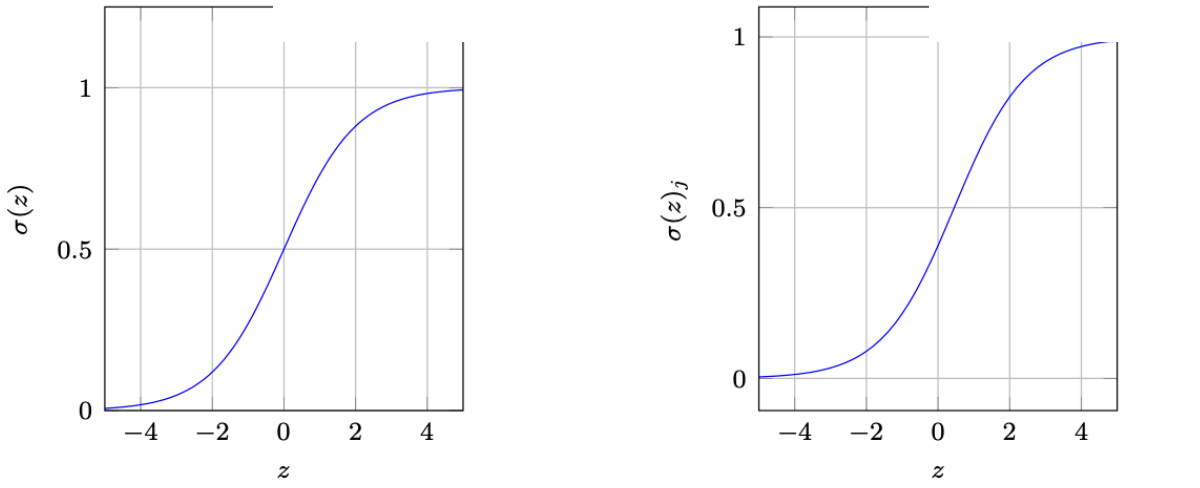
3.**ReLU (Rectified Linear Unit):** *f(x) = max(0, x)*. Она имеет вид прямой линии с наклоном 1 для x>0 и равна 0 для x<=0.



ReLU очень популярна в нейронных сетях, потому что она быстро сходится и не имеет проблемы затухания градиента.

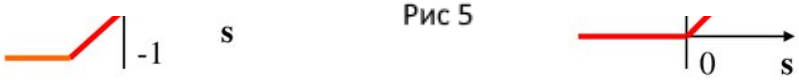
4. **Softmax:** *f(x) = exp(x) / sum(exp(x))*. Эта функция применяется в последнем слое нейронной сети для многоклассовой классификации. Она преобразует выходные значения нейронов в вероятности для каждого класса.

Рис 4



(a) Sigmoid activation function. (b) Softmax activation function.

Рис 5



5. **Линейная функция (Linear):** *f(x) = x*. Она используется в задачах регрессии, когда необходимо предсказывать непрерывные значения.

Выбор функции активации зависит от конкретной задачи и особенностей данных. Они могут быть комбинированы в разных комбинациях в разных слоях нейронной сети для достижения лучшей производительности.

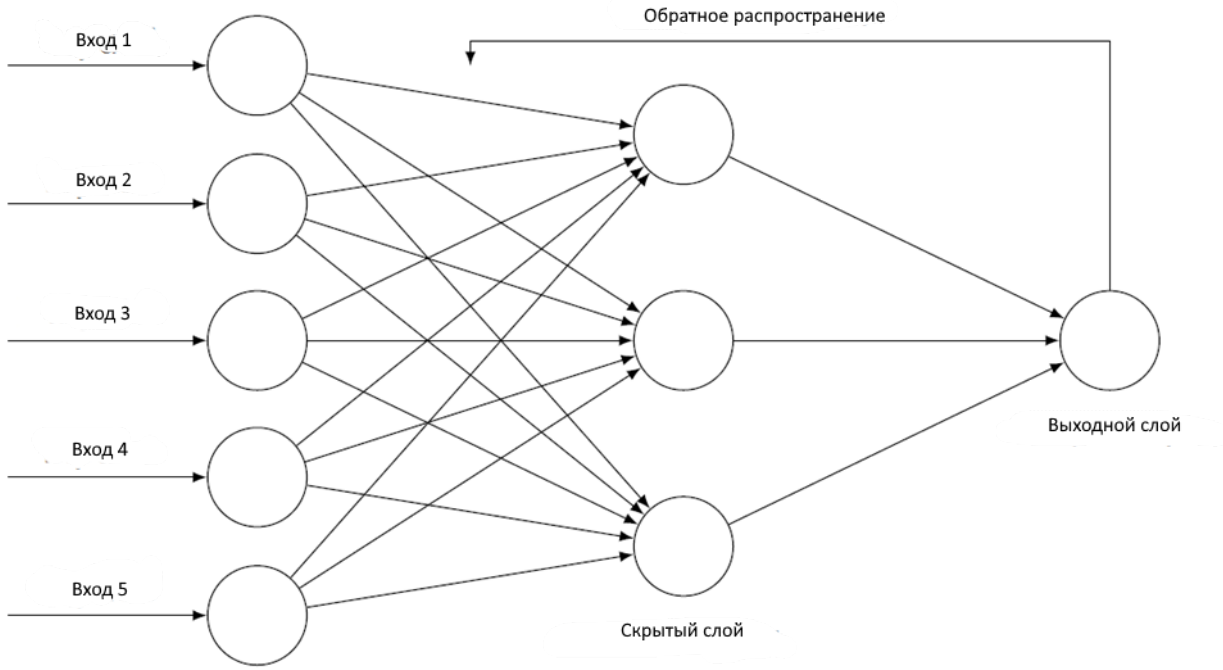
Описание процесса обратного распространения (Backpropagation)

Backpropagation - это алгоритм, используемый для обучения многослойных нейронных сетей, который позволяет определить, как изменение весов и смещений (bias) сети влияет на ее ошибку (loss function). Он основывается на методе градиентного спуска, который минимизирует функцию потерь с помощью последовательного изменения весов в направлении обратного градиента.

Процесс обратного распространения начинается с вычисления ошибки на выходном слое сети. Эта ошибка сравнивает выходное значение сети с ожидаемым значением (в случае задачи классификации, например, это может быть вектор, в котором только один элемент равен 1, а все остальные - 0).

Затем ошибка "распространяется" обратно через сеть, начиная с последнего слоя и до первого. В каждом слое сети вычисляется градиент функции потерь по входным данным и параметрам слоя (весам и смещениям). Для этого используется цепное правило дифференцирования, которое позволяет выразить градиент входной функции через градиент выходной функции.

На каждой итерации обратного распространения веса и смещения в слое корректируются в направлении антиградиента функции потерь. Коэффициент, на который умножается антиградиент, называется скоростью обучения (learning rate).



Таким образом, обратное распространение позволяет оптимизировать веса и смещения в нейронной сети, минимизируя ошибку (функцию потерь) на обучающей выборке. Однако необходимо следить за переобучением сети, что может произойти, если она будет слишком хорошо подстроена под обучающую выборку, и не сможет обобщить свои знания на новые данные.

Оптимизаторы в машинном обучении

Оптимизаторы - это методы, которые используются для обновления весов и смещений в процессе обучения нейронных сетей. Их основная задача - минимизировать функцию потерь, оптимизируя параметры модели. Различные оптимизаторы имеют свои преимущества и недостатки и могут быть эффективными в различных типах задач и архитектур.

Некоторые из наиболее популярных оптимизаторов:

Стохастический градиентный спуск (SGD) - это один из наиболее простых и широко используемых оптимизаторов. Он обновляет параметры модели, используя градиент функции потерь по каждому параметру.

Adaptive Moment Estimation (Adam) - это метод, который адаптивно изменяет скорость обучения на основе первых и вторых моментов градиента.

Root Mean Square Propagation (RMSprop) - это метод, который адаптивно изменяет скорость обучения на основе квадратного корня из среднеквадратичного градиента.

Momentum - это метод, который добавляет некоторый "импульс" к обновлению параметров модели на основе предыдущих градиентов.

Adagrad - это метод, который адаптивно изменяет скорость обучения для каждого параметра на основе истории градиентов.

Adadelta - это метод, который адаптивно изменяет скорость обучения и размер шага для каждого параметра.

Nadam - это метод, который сочетает Adam и Momentum.

Каждый из этих оптимизаторов имеет свои уникальные характеристики, которые могут быть эффективны в различных ситуациях. Выбор оптимального оптимизатора зависит от архитектуры модели, типа задачи и данных, а также от индивидуальных предпочтений и опыта разработчика.

Постановка задачи в нейронной сети

Постановка задачи в нейронной сети зависит от конкретной задачи, которую необходимо решить. В общем случае можно выделить две основные задачи - это задачи регрессии и классификации.

Задача регрессии заключается в предсказании некоторого числового значения на основе входных данных. Например, можно решать задачу предсказания цены на недвижимость на основе её характеристик, таких как площадь, количество комнат, расположение и т.д. В этом случае в качестве выхода нейронной сети будет выступать некоторое числовое значение, которое будет предсказывать цену недвижимости.

Задача классификации заключается в определении принадлежности входных данных к определенному классу. Например, можно решать задачу классификации изображений на

основе их содержимого. В этом случае в качестве выхода нейронной сети будет выступать вероятность принадлежности каждому классу.

Задача выбора типа нейронной сети и её архитектуры также зависит от конкретной задачи и характеристик входных данных. Например, для решения задачи классификации изображений наиболее распространенной архитектурой является сверточная нейронная сеть, которая позволяет обрабатывать входные данные в виде изображений. Для решения задачи регрессии часто используются полносвязные нейронные сети.

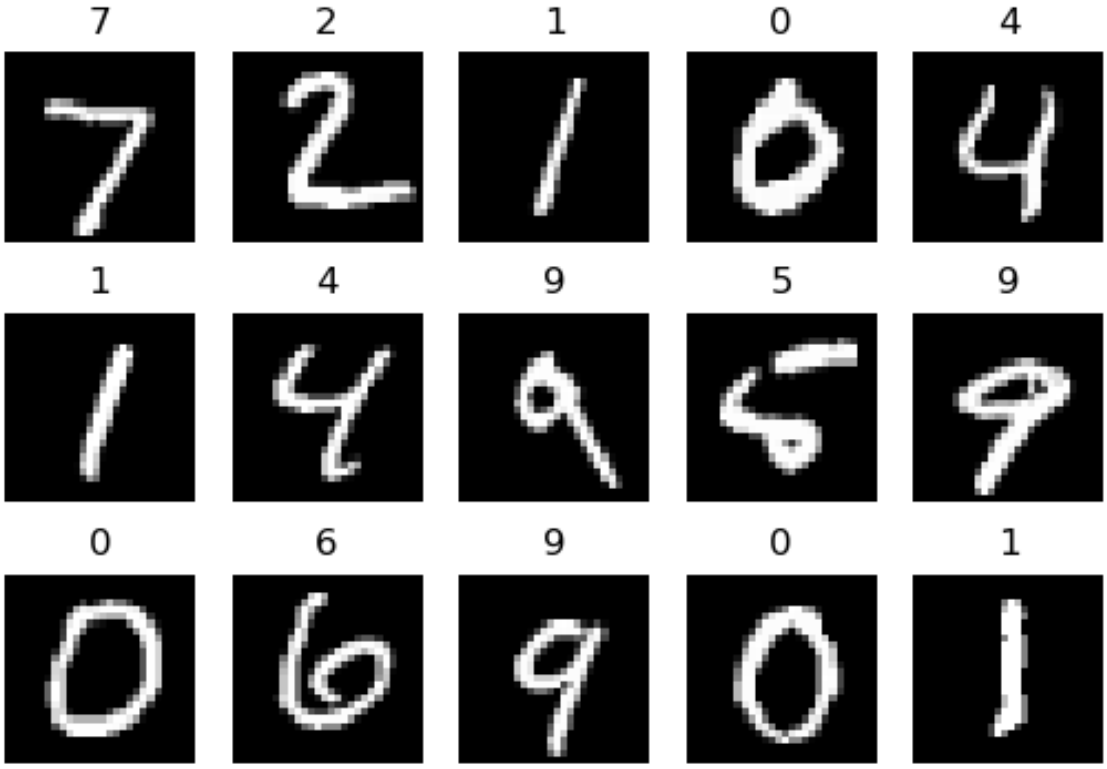
База данных MNIST 28x28

База данных MNIST является одной из самых популярных баз данных в области компьютерного зрения и машинного обучения. Она состоит из рукописных изображений цифр от 0 до 9, размеченных соответствующими метками. Размер изображений составляет 28 на 28 пикселей. Всего в базе данных 60 000 тренировочных изображений и 10 000 тестовых изображений.

Структура базы данных MNIST представляет собой набор изображений в формате MNIST, где каждое изображение имеет размер 28 на 28 пикселей и представлено в виде одномерного массива длиной 784 элемента. Каждый элемент массива содержит значение яркости пикселя в диапазоне от 0 до 255. Кроме того, каждому изображению

соответствует метка от 0 до 9, которая указывает на то, какую цифру изображает данное изображение.

Для работы с базой данных MNIST можно использовать специальные библиотеки, такие как TensorFlow или PyTorch. Библиотеки содержат встроенные функции для загрузки базы данных и подготовки ее к использованию в обучении нейронной сети.



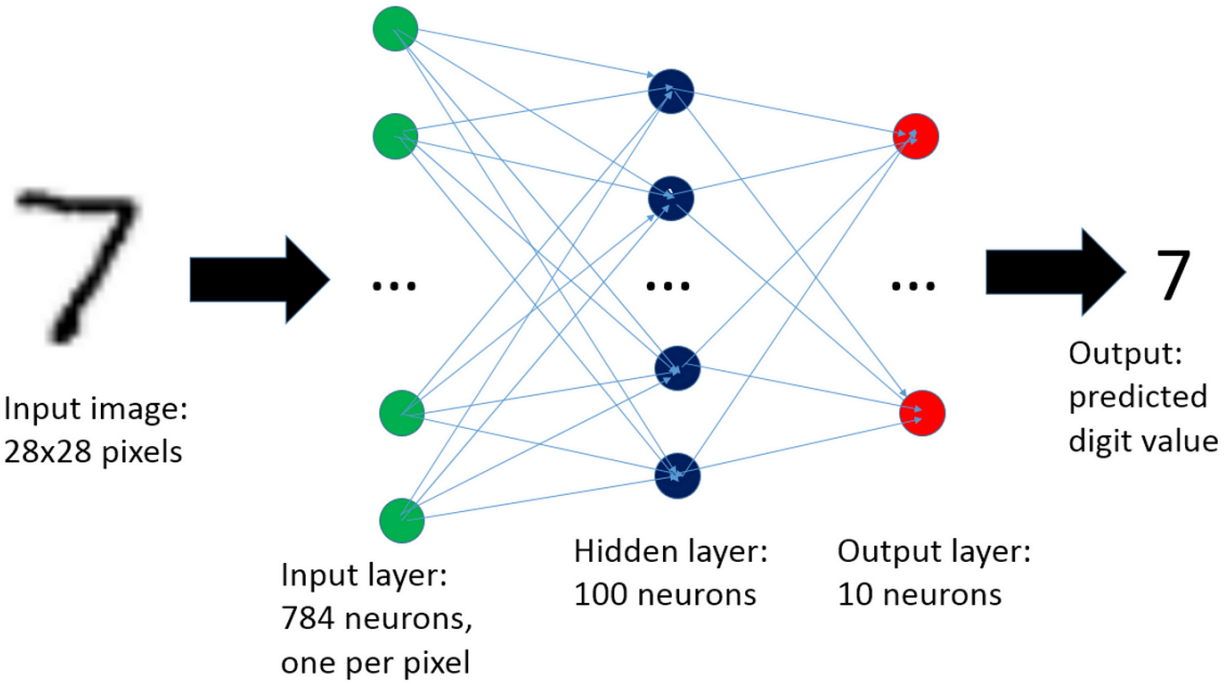
Постановка целей, задачи для распознания цифр

Постановка задачи распознавания цифр на базе данных MNIST включает в себя следующие цели:

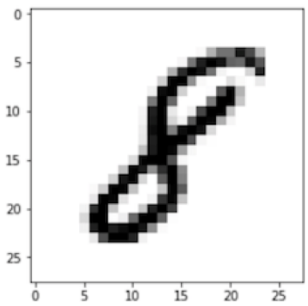
1. Предобработка данных: база данных MNIST состоит из рукописных изображений цифр размером 28x28 пикселей, которые необходимо привести к удобному для обработки формату. Для этого обычно используют методы нормализации и стандартизации, которые позволяют привести значения пикселей к диапазону [0, 1] или [-1, 1], соответственно.

2. Обучение нейронной сети: для распознавания цифр на базе MNIST используется многослойный персептрон (MLP) - один из наиболее распространенных типов нейронных сетей. Обучение MLP происходит с помощью обратного распространения ошибки (backpropagation), в процессе которого происходит оптимизация весов сети для минимизации ошибки классификации.
3. Оценка качества модели: после обучения необходимо оценить качество модели на тестовых данных, которые не были использованы в процессе обучения. Для этого используют метрики качества, такие как точность (accuracy), точность и полнота (precision and recall) и F-мера (F1-score).

Целью задачи распознавания цифр на базе данных MNIST является разработка модели, которая сможет распознавать рукописные цифры с высокой точностью.



Google Colab

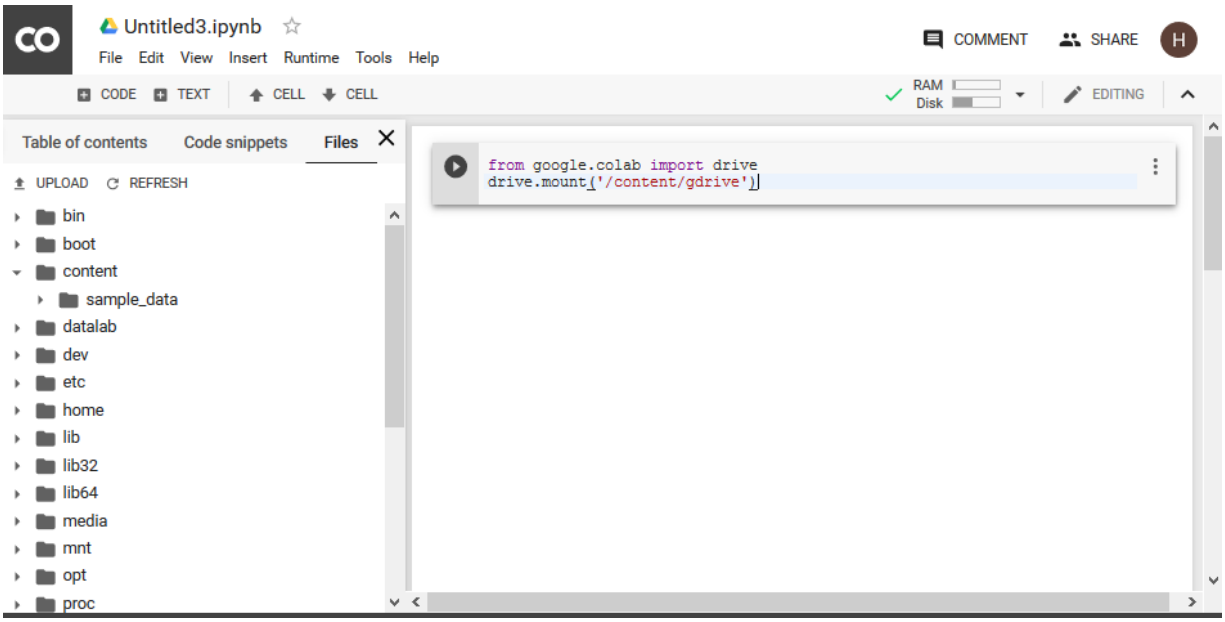


Google Colab (Colaboratory) - это бесплатный веб-сервис, который предоставляет возможность запускать и создавать Jupyter-ноутбуки в облаке. Colab был создан Google, и он позволяет пользователям работать с Python и другими языками программирования для анализа данных и машинного обучения.

Colab позволяет создавать, запускать и сохранять Jupyter-ноутбуки в облаке Google, что позволяет быстро и легко подключаться к мощным вычислительным ресурсам, таким как графические процессоры (GPU) и тензорные процессоры (TPU), без необходимости устанавливать соответствующее оборудование на своем компьютере.

Для использования Colab необходима только учетная запись Google и доступ в Интернет, что делает его доступным для широкой аудитории пользователей. Кроме того, Colab имеет богатый функционал для работы с данными и библиотеками машинного обучения, такими как TensorFlow, Keras, PyTorch, OpenCV и многими другими.

Colab также позволяет совместную работу с другими пользователями, обмен результатами работы через Google Drive, а также интеграцию с другими сервисами Google, такими как BigQuery, Google Sheets и Google Cloud Storage.



Практика в Google Colab - это возможность использовать мощь вычислительных ресурсов Google и удобство работы в облачной среде для создания, обучения и тестирования нейронных сетей.

Для начала работы с Google Colab необходимо создать учетную запись Google и зайти на страницу colab.research.google.com. Далее нужно создать новый ноутбук, выбрав язык программирования и оборудование для выполнения вычислений.

Google Colab предоставляет доступ к множеству библиотек и фреймворков для машинного обучения, таких как TensorFlow, PyTorch, Keras и др. Эти библиотеки могут быть установлены и использованы в ноутбуке с помощью команды !pip install.

В Google Colab можно загружать свои наборы данных или использовать уже готовые базы данных, например, MNIST, CIFAR-10 и т.д. Также можно использовать готовые модели и нейронные сети, предварительно обученные на больших наборах данных, для решения своих задач.

Важной особенностью Google Colab является возможность совместной работы над одним ноутбуком с другими пользователями. Это позволяет обмениваться опытом и знаниями, а также быстро решать возникающие проблемы и исправлять ошибки.

В целом, Google Colab является удобной и мощной средой для работы с машинным обучением и глубоким обучением, которая позволяет легко создавать, обучать и тестировать нейронные сети, а также совместно работать над проектами с другими пользователями