**Веса** – это свободные параметры, наши настройки нейронной сети, изменяя, которые, мы меняем ее поведение и результат работы

**Смещение (bias)** — это параметр, который позволяет нейрону сдвигать свой выход на постоянное значение.

**Ошибка или функция потерь (cost/loss)** – это насколько предсказания нейронной сети отличаются от истинных значений. Нам нужно минимизировать это значение, найти **точку перегиба**

**Prediction –** это значение, предсказанное моделью на основе входных данных

**Target –** это ответ, желаемое значение, которое модель пытается предсказать

**1.Что такое нейронная сеть? Из чего состоит? Для чего нужна?**

* **Что такое нейронная сеть?**

**Что нужно сказать:** Нейронная сеть **с точки зрения математики** – это математическая модель, уравнение, формула. Мы знаем какие у нас есть входные данные и нам необходимо с ними работать. **С точки зрения программирования** – это алгоритм или же программа, которая выполняет какую-то задачу.

**Чего не стоит говорить:** что нейронная сеть — это человеческий мозг. Нейронная сеть нужна для решения сложных задач.

* **Для чего нужна?**

Нейронные сети нужны для решения задач, у которых нет алгоритма решения. Большая часть задач, которые решают нейронные сети это задачи по нахождению соответствия

Нейронные сети нужны для обработки и интерпретации сложных данных, обучения на примерах и выявления закономерностей или трендов, которые могут быть неочевидны для человека или традиционных алгоритмов обработки данных.

* **Из чего состоит?**

У нас есть несколько **слоев** (входной, скрытые (бизнес-логика)), выходной) и **нейроны**.

Нейронная сеть состоит из слоев нейронов: входного слоя, одного или нескольких скрытых слоев и выходного слоя. Каждый нейрон в этих слоях связан с другими и обрабатывает информацию, используя веса и функции активации.

**2.Виды нейронных сетей и особенности их применения:**

* **Полносвязные нейронные сети** (Fully Connected Networks):

1. Применяются для структурированных табличных данных.
2. Каждый нейрон одного слоя соединён с каждым нейроном следующего слоя, что обеспечивает сложное комбинирование признаков.

* **Сверточные нейронные сети** (Convolutional Neural Networks - CNNs):

1. Идеальны для анализа изображений и видео.

**Основные особенности:**

1. Уменьшение количества входов: используются фильтры для уменьшения размерности данных, сохраняя при этом важные признаки.
2. Позиционная инвариантность: способность распознавать объекты независимо от их положения в изображении.
3. Локальное восприятие: каждый фильтр рассматривает только небольшую область изображения, что помогает распознавать локальные паттерны.

* **Рекуррентные нейронные сети** (Recurrent Neural Networks - RNNs):

1. Подходят для работы с последовательными данными, такими как тексты, временные ряды, аудио.

**Особенность:**

1. Способность сохранять информацию из предыдущих шагов, что позволяет учитывать контекст в данных.

* **Генеративно-состязательные сети** (Generative Adversarial Networks - GANs):

1. Используются для генерации новых данных, которые похожи на обучающую выборку.
2. Состоят из двух частей: генератора, создающего данные, и дискриминатора, оценивающего их реалистичность.

**3.Что такое конфигурация нейронной сети?**

**Конфигурация нейронной сети** — это ее структура, включающая количество и тип слоев, количество нейронов в каждом слое, типы функций активации и другие настройки, которые определяют ее поведение и способность к обучению.

* Виды и количество слоев, количество нейронов в слое
* Формат входных и выходных данных (количество входов и выходов). Ответ нейронной сети — это массив из двух чисел (одно число большое другое маленькое, процентная вероятность) грубо говоря это можно назвать one-hot vector
* Задаем связи между слоями и определяем направление движения информации
* Активационная функция

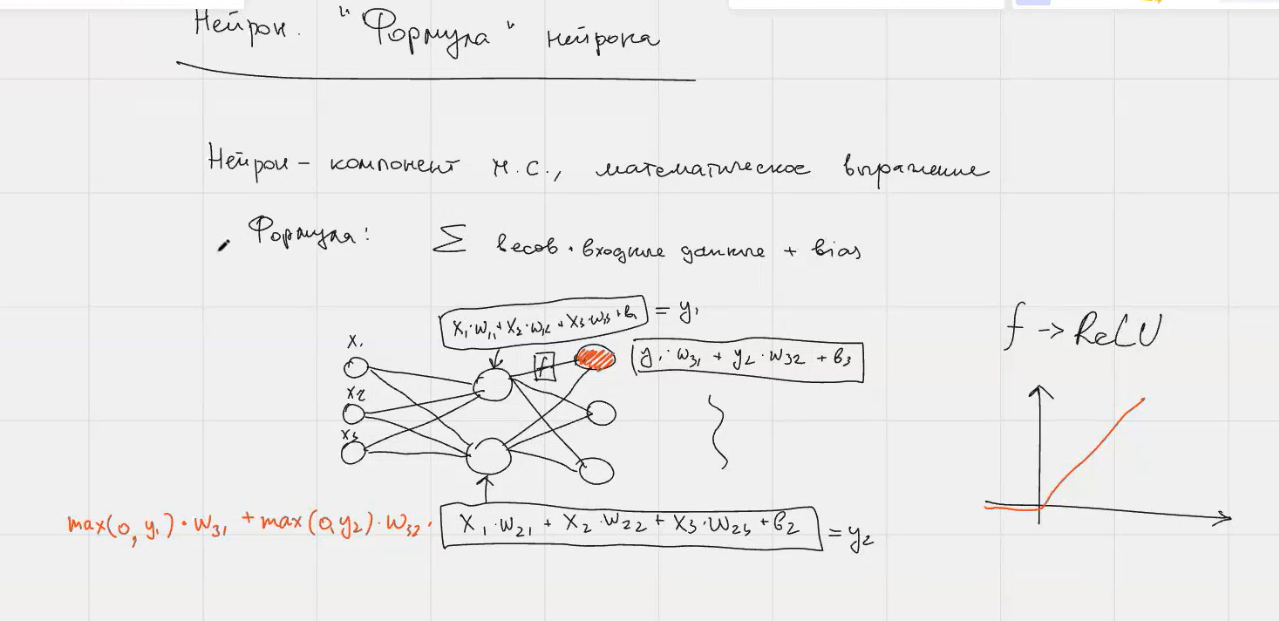
**Вот несколько ключевых понятий, связанных с конфигурацией нейронной сети:**

* **Слои**: Нейронные сети обычно состоят из нескольких слоев. Входной слой получает входные данные, скрытые слои выполняют промежуточные вычисления, а выходной слой генерирует окончательные результаты.
* **Нейроны**: Нейроны представляют собой базовые вычислительные блоки в нейронной сети. Каждый нейрон принимает входные данные, умножает их на веса, суммирует результаты, применяет функцию активации и передает выходные данные следующему слою.
* **Веса и смещения**: Веса связывают нейроны в разных слоях и определяют влияние каждого входа на выход нейрона. Смещения представляют собой константы, которые добавляются к взвешенной сумме входов перед применением функции активации.
* **Функции активации**: Функции активации определяют выход нейрона в ответ на входные данные. Они добавляют нелинейность в модель, что позволяет сети обучаться более сложным зависимостям.
* **Архитектура**: Архитектура нейронной сети определяет, как слои соединены между собой. Например, в одном слое могут быть все нейроны соединены с каждым нейроном следующего слоя (полносвязная архитектура), или нейроны могут быть организованы в определенные паттерны (например, в сверточных слоях для обработки изображений).

**4.Что такое нейрон? “Формула” нейрона?**

**Нейрон –** это атомарный компонент нейронной сети, математическое выравнивание.

Нейрон в контексте нейронных сетей — это математическая функция, которая принимает входные данные, взвешивает их, применяет функцию активации и передает результат дальше. Формула нейрона обычно выглядит как

Сумма весов \* на входные данные + баес / f(x)=активация(∑(weights⋅inputs)+bias).  


Также важно понимать, что есть еще функция активации, через которые эти данные будут пропущены.  
f -> ReLu – это говорит о том, что любые отрицательные значения превращаются в ноль, а любые положительные значения остаются такими как они есть. И тогда мы должны писать максимальное значение между нулем и y1 \* w31

**5.Объяснение диаграммы нейронной сети**

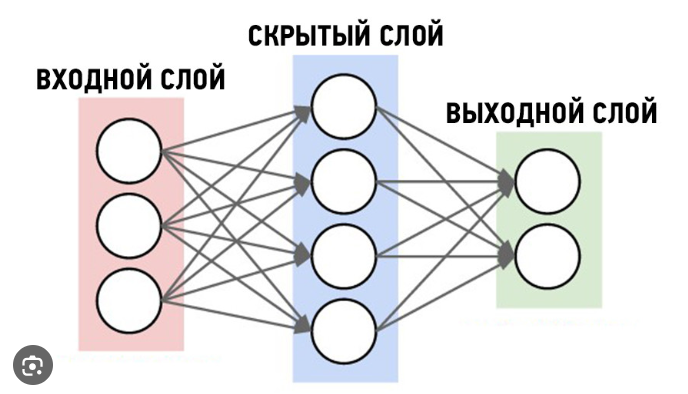
****

Диаграмма нейронной сети показывает, как данные перемещаются от входного слоя через скрытые слои к выходному слою, и как нейроны в этих слоях взаимодействуют друг с другом.

* **Что предоставляют собой стрелки?**

Например, есть у нас три входа и будет один нейрон, к которому эти входы подключаются, смотря на эту картинку мы можем сказать четкую формулу x1\*w1+x2\*w2+x3\*w3. Это действие называется **взвешиванием**, то есть умножение числа на какой-то вес, каждый из переходов (стрелок) означает умножение на какое-то число. Понимая это, мы и получаем формулу **x1\*w1+x2\*w2+x3\*w3.** (линейная комбинация) к этой сумме часто добавляется смещение (bias)

Веса в начале обучения инициализируются случайно и корректируются в процессе обучения для минимизации ошибкиНаш граф (стрелочка от числа к нейрону) является **направленным** и **взвешенным**Также применяется какая-то активационная функция (это тоже важно сказать)

**6.Что такое внутренние и внешние параметры нейронной сети?**

**Внешние параметры** – это данные, поступающие на вход нейронной сети. Данные, которые пользователь отправляет в нейронную сеть.

**Требования к внешним данным**

* Данные должны быть **единообразны** (в идеале эти данные должны быть нормализованы)  
  **Нормализация входных данных** – это подразумевает, что данные будут находиться в строго заданном диапазоне и чаще всего этот диапазон от 0 до 1. Мы просто берем и делим эти данные на среднее значение этих данных
* Данные должны быть **сбалансированы**. Данные, которые мы считываем должны быть в равном количестве, если мы считываем цифры, то не должно быть больше троек, потому что в процентном соотношении нейронная сеть будет чаще видеть тройки.

**Внутренние параметры** – это непосредственно наши веса и баесы, которые мы подставляем.

**Требования к внутренним параметрам**

* Изначально равны **рандомному** значению, кроме 0 (потому что мы все перемножаем и если мы будем умножать на ноль мы и получим ноль)

**7.В чем заключается обучение нейронной сети?**

**Обучение нейронной сети** — это процесс настройки внутренних параметров (весов и смещений) таким образом, чтобы минимизировать разницу между предсказанными и реальными данными на основе предоставленного набора обучающих данных.

**Цель обучения нейронной сети**

* Минимизация ошибки путем подбора внутренних параметров (весов и баеса)

**Процесс обучения**

1. Forword pass - для начала мы исходные данные отправляем в нейронную сеть и получаем ее ответ
2. Определяем ошибку нейронной сети (loss/cost) – находим функцию ошибки (prediction – target)^2/ (ответ нейронной сети – правильный ответ)^2 . То есть узнаем разницу между правильным ответом и ответом нейронной сети.
3. Вычисляем градиенты (частные производные) – определяем в какую сторону было смещение (это можно сделать с помощью **угла коэффициента касательной в точке** (k=y2-y1)/(x2-x1) или с помощью производной (скорость изменения функции) k = (разница по y)/(разница по x)
4. Вычисляем новое значение свободного параметра bновое = bстарое – m \* **0.1**

**Learning Rate** (скорость обучения **0.1**) – если скорость обучения слишком высока, сеть может перескакивать через оптимальные решения; если слишком низка — обучение будет происходить очень медленно.

**8.Чем отличаются данные для обучения от данных для тестирования?**

Грубо говоря это те же данные, желательно чтобы они были отличимы от данных, которые мы уже использовали. Изначально часть данных для тестированная мы берем с данных для обучения.

* **Смысл их разделять?** Чтобы мы посмотрели, насколько нейросеть обучена, количество правильных/неправильных ответов. Чтобы наша нейросеть впервые столкнулась с этими параметрами, и мы посмотрели ее работу.

**9.Функция ошибки (Loss/Cost function)**

**Loss= (prediction – target)^2**

**Функция ошибки (Loss или Cost)** измеряет разницу между предсказаниями нейронной сети и реальными данными. Она помогает определить, насколько хорошо сеть выполняет задачу, и используется для корректировки весов и смещений сети в процессе обучения. Чем меньше значение функции ошибки, тем точнее предсказания сети.

Loss или Cost – это функция большого количества переменных

Loss (w1, w2, w3,w4 … bias1, bias2, bias3…) для того чтобы оптимизировать эту ошибку, которая зависит от большого количества переменных мы находим частную производную.

**Значение производных:** производные функции ошибки по весам и смещениям (частные производные) используются для определения направления, в котором нужно изменять эти параметры, чтобы уменьшить ошибку, т.е., как быстро и в каком направлении нужно "шагать" для улучшения модели.

**Обновление параметров:** процесс обновления параметров сети (весов и смещений) на основе производных функции ошибки — это то, что обычно называется "обучением" модели.

**10.Каким образом происходит подбор внутренних параметров при обучении нейронной сети?**

Подбор внутренних параметров происходит через процесс обучения, где с помощью алгоритма обратного распространения ошибки и оптимизации (часто используя градиентный спуск) веса и смещения корректируются для минимизации функции ошибки.  
Нужно сделать больший упор на математику, почти всегда при вычислении должна находиться активационная функция   
Наша ошибка будет равна – Loss= (prediction – target)^2

Model/target – результат вычислений Активационной функции от результата нейронной сети, а результат нейронной сети зависит от весов и баеса

**11.Что такое градиентный спуск? Для чего он нужен?**

**Градиентный спуск (частные производные)** — это метод оптимизации, который используется для нахождения минимума функции ошибки путем постепенной корректировки параметров сети в направлении наибольшего уменьшения ошибки.

Передвижение по параболе. Парабола отражает нашу ошибку, и мы перемещались по этой параболе с какой-то точки вниз мы находили минимальное значение ошибки.  
Парабола — это только пример функции ошибки, для очень примитивной нейронной сети (вырожденной нейронной сети)  
**Вырожденная нейронная сеть** – это сеть, которая не содержит весов, мы рассматривали один только баес. На практике такая нейронная сеть не будет применяться, из-за того, что будет всего один параметр, гибкость такой сети будет минимальной.

* **Почему именно парабола** – чтобы избавиться от отрицательных значений и выйти в ноль, при помощи возведения в квадрат. Получаем график квадратичной функции, то есть параболу.
* **Каким образом мы изменяем тот параметр, который мы обучаем в процессе обучения –** мы находим то, что мы называем градиентами, то есть частную производную, которую мы находим по функции ошибки по одной из переменных, которая на эту ошибку влияет.

Например, у нас есть два входа (x1 и x2) и был один нейрон, формула **(x1 \* w1 + x2 \* w2 + b)**. Из этой формулы на ошибку влияют **веса** и **баес**. Когда мы будем выполнять градиентный спуск, одно из действий которое мы выполняем в процессе поиска правильных значений для весов мы будем находить частные производные.  
**x1 и x2 не влияют на ошибку**, поскольку входные параметры не могут изменяться, и мы не можем сказать, что от них зависит ошибка. Эти параметры мы откуда-то взяли (измеряли, выяснили и так далее)

* **Почему именно производная?** Помогает найти скорость изменения функции. Производная дает понять, насколько сильно изменяется функция. Функция может убывать или возрастать. Мы можем сказать, не видя графика, где функция убывает или возрастает. Если y увеличивается значит функция возрастает, если уменьшается значит убывает. **Что это нам дает?** Мы ищем точку перегиба, то место, где производная меняет свой знак (перед этой точкой у нас производная уменьшалась, а после этой точки начинает возрастать, или наоборот)
* **Почему именно частные производные? Частные производные –** они измеряют как функция многих переменных меняется в ответ на изменения только одной из этих переменных, а все остальные остаются постоянными. Потому что Loss – это функция большого количества переменных. Loss – зависит от (w1, w2, b) и частные производные нам надо находить по весам и баесу. Писк этих частных производных и изменения этих значений и будет градиентным спуском. Сначала мы определяем по какой переменной мы хотим найти производную остальные переменные фиксируем берем их как константы
* **Почему градиентный спуск?** Потому что, если посмотреть на параболу (! Для функции трех переменных это будет уже не парабола, а четырехмерный график, который мы не можем отобразить), но идея остается. У нас есть парабола, которая зависит от b (мы берем какое-то случайное значение) то мы получаем какую-то случайную точку на одной из ветвей. И по мере нахождения градиента, мы должны оказаться в какой-то более низкой точке. Если мы все делаем правильно, то с каждой такой итерацией ошибка должна уменьшаться.

**12.Что влияет на время обучения и качество работы нейронной сети?**

* **Размер данных:** больше данных может увеличивать время обучения, но улучшать качество.
* **Сложность сети:** больше слоев и нейронов может улучшить способность учиться, но замедлить обучение.
* **Гиперпараметры:** Настройки, как скорость обучения и количество эпох, влияют на скорость и результат.
* **Вычислительные ресурсы:** Мощные процессоры или GPU ускоряют обучение.

**13.Каким образом можно улучшить качество работы нейронной сети?**

* Количество параметров
* Структура нейронной сети (тип нейронной сети, разные нейронные сети подходят для решения разных задач)
* Сбалансированность данных
* Количество циклов\эпох обучения. Нельзя перетренировать нейронную сеть с конкретными параметрами. Мы можем увеличить количество эпох но до какого-то предела, иначе мы получим нейронную сеть которая идеально распознает данные для обучения, но плохо распознает данные для тестирования

Улучшить качество работы сети можно через настройку гиперпараметров, увеличение объема и разнообразия обучающих данных, использование регуляризации для предотвращения переобучения, а также применение техник улучшения обучения, таких как аугментация данных или изменение архитектуры сети.

**14.Что такое функция активации? Для чего она нужна? Какие активационные функции бывают?**

**Активационная функция в нейронной сети —** это правило, которое помогает каждому нейрону решить, насколько сильно он должен активироваться, или насколько сильным должен быть его сигнал. Это как фильтр, который принимает входящие сигналы, обрабатывает их и решает, передавать ли сигнал дальше и как сильно. Это помогает нейронной сети учиться на сложных данных и делать полезные предсказания.

**Картинка -> карта свойств (feature map) -> активационная функция (ReLU) -> (max-pool)/(mean-pool) -> двумерный массив сплющивается в одномерный и отправляется в нейронную сеть.**

вход –> сверточная часть (CNN) –> полносвязная нейронная сеть (FC) –> выход.

Когда мы обрабатываем картинки, например, с крестиками и ноликами, изображение проходит через несколько этапов. Сначала оно попадает на 'карту свойств' **(feature map)**, где выделяются важные черты. Затем применяем активационную функцию, например **ReLU**, которая помогает определить, какие черты наиболее важны. После этого используем **'max-pooling**' для выбора наибольшего значения в области (или **'mean-pooling'** для вычисления среднего), чтобы упростить и уменьшить размер данных. Затем двумерный массив данных сплющивается в одномерный и отправляется в полносвязную нейронную сеть **(FC),** где происходит дальнейший анализ и классификация.

* **Для чего это надо?** Весь процесс позволяет сети эффективно уменьшить количество анализируемых пикселей, например, с 36 до 4, упрощая задачу и повышая скорость работы.   
    
  **Это позволяет анализировать, к примеру не 36 пикселей, а всего на всего 4**.
* **Популярные активационные функции**

1. **Сигмоид** (Sigmoid): преобразует значения в диапазон от 0 до 1. Часто используется в последнем слое для бинарной классификации.
2. **Гиперболический тангенс** (Tanh): преобразует значения в диапазон от -1 до 1. Это центрированная версия сигмоида.
3. **ReLU** (Rectified Linear Unit): предоставляет линейный выход для всех положительных входов и 0 для всех отрицательных.
4. **Softmax**: часто используется в последнем слое многоклассовых классификационных задач. Преобразует вектор входных значений в распределение вероятностей.

* **Откуда появился фильтр?**

Изначально нейронная сеть применяет рандомные фильтры, в процессе применения какие-то фильтры срабатывают лучше, чем другие. Методом подбора