Нейросеть

Нейросеть — это математическая модель и её программное или аппаратное воплощение, которая имитирует работу нервных сетей живого организма, в частности человеческого мозга. Она состоит из множества связанных между собой искусственных нейронов, которые обмениваются информацией и обучаются на данных, чтобы решать сложные задачи: распознавать образы, классифицировать данные, прогнозировать и принимать решения. Нейросеть устроена слоями: входной принимает данные, скрытые слои их обрабатывают, а выходной слой выдает результат. Современные нейросети применяются в генерации текстов и изображений, распознавании речи, медицине, финансах и других областях, они могут самостоятельно обучаться и адаптироваться, становясь мощным инструментом анализа и автоматизации.

Нейрон в искусственной нейронной сети — это простейшая вычислительная единица, которая принимает несколько входных сигналов, обрабатывает их и генерирует один выходной сигнал. Каждый вход умножается на определённый вес (весовой коэффициент), после чего все взвешенные входы суммируются. Полученная сумма затем передаётся в функцию активации, которая преобразует её в итоговое значение выхода нейрона.

Устройство искусственного нейрона можно описать так:

– входы: несколько значений, которые нейрон принимает;

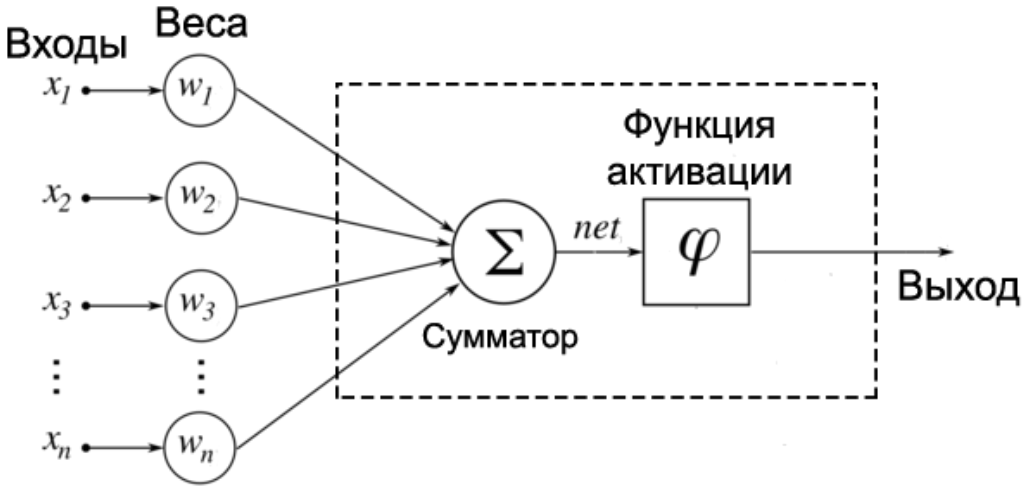
– веса: числовые коэффициенты, отражающие важность каждого входного сигнала;

– сумматор: агрегирует взвешенные входы в единую сумму;

– функция активации: нелинейная функция, применяемая к сумме, формирующая выход нейрона;

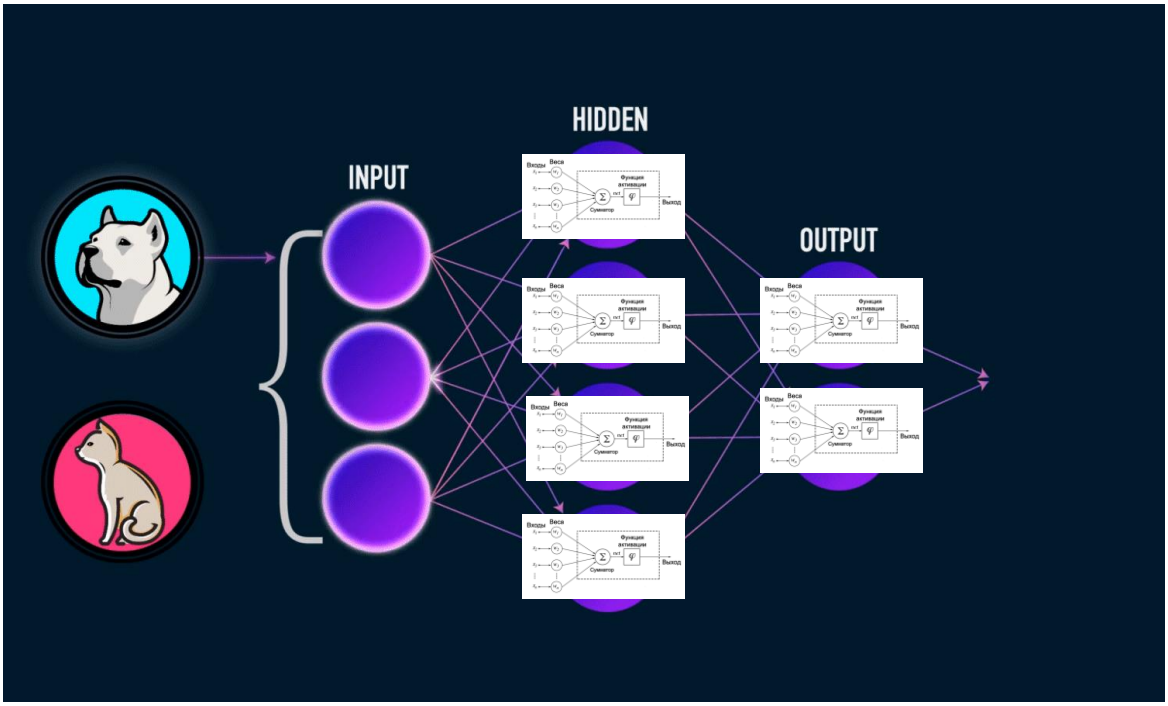
– выход: результат работы нейрона, который передаётся дальше другим нейронам или на выход сети.

Схематически нейрон можно представить так:



По структуре искусственный нейрон — упрощённая модель биологического нейрона, который в живом организме состоит из тела клетки с ядром, и отростков: дендритов (принимают сигналы) и аксона (передаёт сигнал). Искусственные нейроны также связаны между собой синапсами, или связями с весами, обеспечивающими обучение сети. Нейрон — это своего рода "черный ящик" с несколькими входами и одним выходом, внутри которого происходит преобразование входных данных с помощью весов и функции активации для формирования выходного сигнала.

Понимание организации нейрона позволяет представить нейросеть в целом таким образом, как показано на рисунке:



В работе нейросети выделяют прямой и обратный проходы. Прямой проход (forward pass) — это этап работы нейросети, когда входные данные подаются в сеть и проходят через все слои последовательно, от входного к выходному. В результате на выходном слое получается предсказание или ответ сети, основанный на текущих весах и параметрах. На этом этапе происходит преобразование данных и вычисление выходного сигнала без изменения весов.

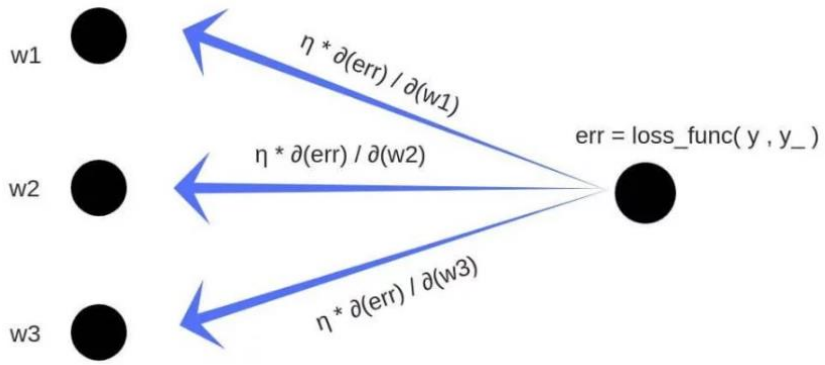
Обратный проход (backward pass) — это этап обучения сети, когда рассчитывается ошибка предсказания на выходе (разница между фактическим и желаемым результатом), и эта ошибка распространяется назад по слоям сети. При этом с помощью метода обратного распространения ошибки происходит корректировка весов нейронов для уменьшения ошибки на будущих проходах. Весовые коэффициенты обновляются, чтобы улучшить точность сети,

используя градиентный спуск и производные функции активации. Так, прямой проход отвечает за вычисление выхода нейросети по входу, а обратный проход — за обучение нейросети посредством корректировки её параметров на основе ошибки предсказания

При обратном проходе применяется метод обратного распространения ошибки. Основа метода обратного распространения ошибки — это применение цепного правила из математического анализа для вычисления градиентов функции потерь (ошибки) по параметрам нейронной сети (весам). По сути, в обратном проходе сети происходит распространение ошибки от выходного слоя к входным слоям, при этом вычисляются частные производные ошибки по каждому весу. Эти градиенты используются для корректировки весов с помощью метода градиентного спуска, чтобы минимизировать ошибку сети.

Идея метода такова: сначала сеть делает прогноз по входным данным (прямой проход), затем вычисляется ошибка прогноза, после чего ошибка «распространяется» обратно через слои сети (обратный проход). В этом процессе веса обновляют по направлению антиградиента функции ошибки для приближения к её минимуму. Благодаря этому методу нейросеть обучается подстраивать параметры так, чтобы улучшить качество предсказаний.

Метод обратного распространения ошибки связывает обучение нейросети с оптимизацией функции потерь через вычисление градиентов с помощью цепного правила, что позволяет эффективно и последовательно корректировать все веса сети.



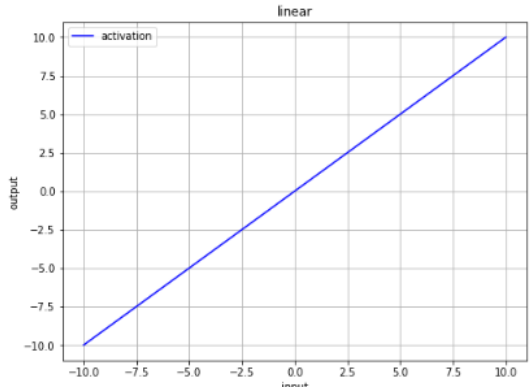
Функции активации нейросетей

Линейная функция

Самая простая функция активации – линейная. График функции и ее производной представлены на рисунках ниже

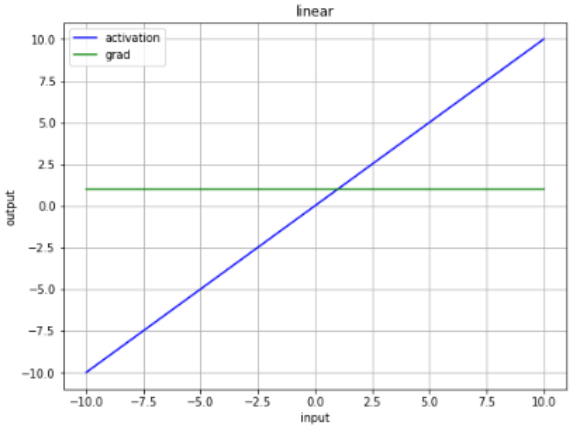
Линейная функция представляет собой прямую линию, и она пропорциональна входу.

$$A(x) = x$$



Производная такой функции по x равняется:

$$\frac{\partial A}{\partial x} = 1$$



Для этой функции производная всегда одинакова в любой точке.

А это значит, что градиент никак не связан с входом (x). Градиент - постоянный вектор, а спуск идет по **постоянному** градиенту. Если есть ошибочное предсказание, то изменения, сделанные обратным распространением ошибки, тоже постоянны вне зависимости от входа.

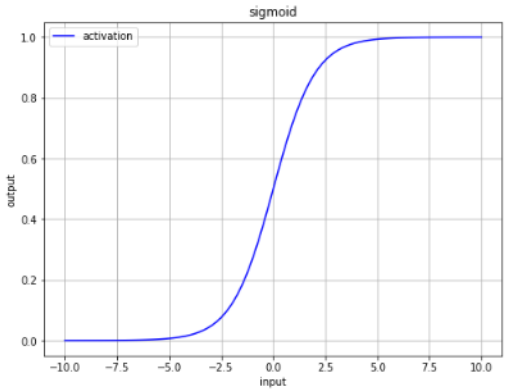
Нет разницы, сколько в сети слоев, т.к. финальная функция активации в последнем слое будет просто линейной функцией от входных параметров на первом слое, соответственно любое количество слоев можно заменить одним. Нет смысла ставить подряд слои с линейной активацией.

Сигмоида

Сигмоида – нелинейная активационная функция

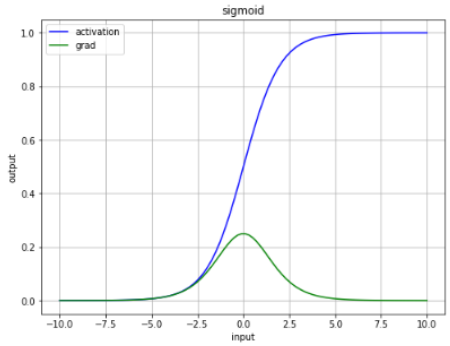
Сигмоида - одна из самых частых активационных функций в нейросетях.

$$A(x) = \frac{1}{1 + e^{-x}}$$



Производная сигмоиды

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$



Сигмоида — нелинейная функция, а комбинация таких функций производит тоже нелинейную функцию. Соответственно проблема линейной функция здесь неактуальна.

В диапазоне значений сигнала от -2.5 до 2.5 значения активации меняются очень быстро, т.е. любое изменение значения сигнала в этой области повлечет существенное изменение значения активации.

Сигмоида идеально подходит для задач бинарной классификации. Она стремится привести значения к одной из сторон кривой, по итогу получаем четкие границы предсказания

У сигмоиды фиксированный диапазон значений функции — [0,1], а линейная функция обладает диапазоном (−∞, +∞). А значит в случае с сигмоидой не будет ошибок с большими значениями активаций

Недостатки

При приближении к концам сигмоиды значения активации очень слабо изменяются. Это означает, что градиент в таких областях имеет очень мальнькие значения. А из-за этого рождается проблема затухающих градиентов

Градиент получается настолько маленьким или же вообще нулевым, что обучение замораживается или же идет очень медленно.

Тангенс

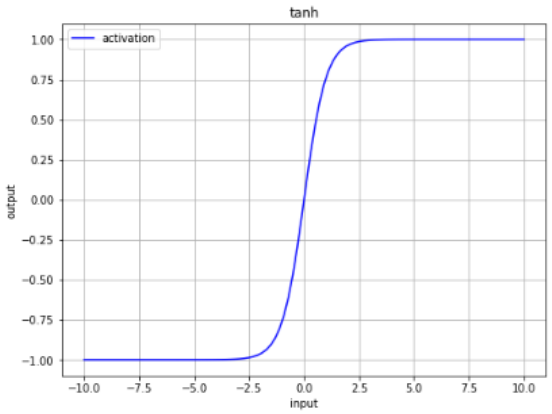
Гиперболический тангенс очень похож на сигмоиду, только производная у него чуть больше, но все остальные плюсы и минусы остаются.

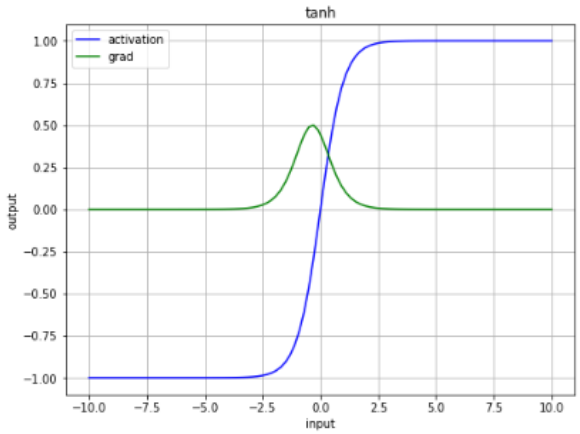
Производная тангенса

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} = 2\sigma(2x) - 1$$

$$\partial \tanh(z) = \frac{1}{ch^2(x)} = \frac{1}{(\frac{e^x + e^{-x}}{2})^2}$$

$$ch(x) = \frac{e^x + e^{-x}}{2}$$





Достоинства:

1. Тангенс - нелинейная функция, можем обучать подряд много слоев.
2. В диапазоне значений сигнала от -2.5 до 2.5 значения активации меняются очень быстро. А значит, что любое изменение значения сигнала в этой области повлечет существенное изменение значения активации.
3. Тангенс идеально подходит для задач бинарной классификации. Он стремится привести значения к одной из сторон кривой.
4. У тангенса фиксированный диапазон значений функции — [-1,1]. А значит не будет ошибок с большими значениями аткиваций

Недостатки:

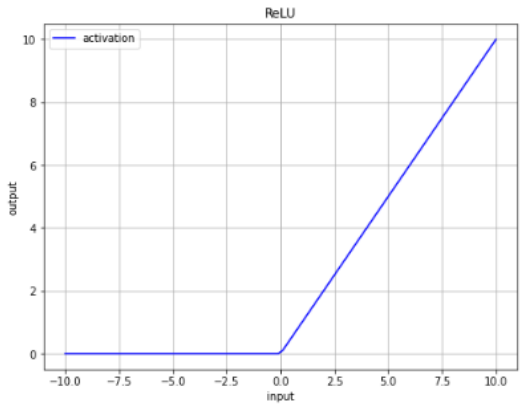
При приближении к концам тангенса значения активации очень слабо изменяются. Имеем проблему затухающих градиентов.

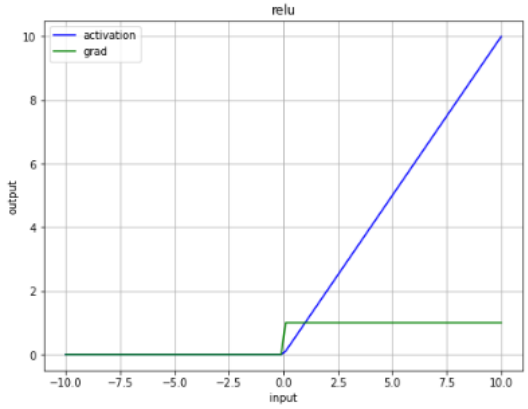
ReLU

ReLU (Rectified Linear Unit) - это нелинейная функция активации.

Производная

$$A(x) = max(0, x)$$





Если имеем большую нейронную сеть с огромным количеством нейронов, то использование сигмоиды или гиперболического тангенса повлекут за собой активацию буквально всех нейронов. Чтобы сеть была легче, необходимо, чтобы некоторые нейроны не были активированы. ReLu может это сделать. Например, если у нас есть нейросеть со случайно инициализированными весами, то 50% активаций равны 0 по ReLU.

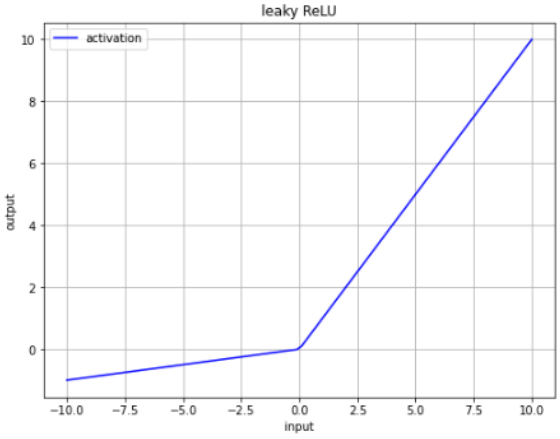
Но и тут вознимает проблема. Градиент на этой части равен 0. А значит, веса сети не будут изменяться во время обучения.

Но существуют модификации ReLU, которые помогают эту проблему решить.

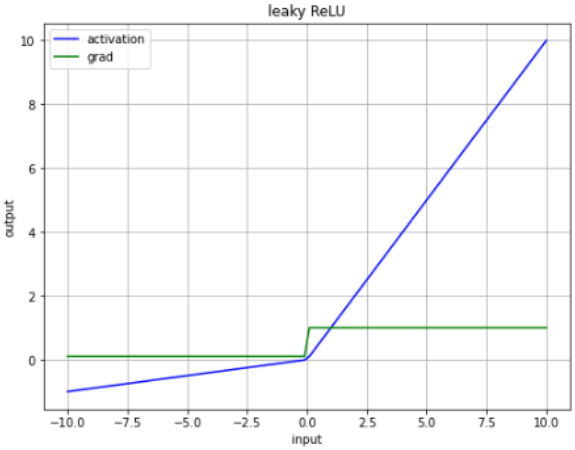
Leaky ReLU

Для решения проблемы нулевого градиента имеет смысл заменить горизонтальную часть функции на линейную

$$A(x) = max(0.1x, x)$$



Производная



Как выбрать функцию активации?

Для выбора функции активации можно пользоваться следующими рекомендациями

Функция активации	**Достоинства**	**Недостатки**
linear	1. Меньше времени на вычисления	1. Линейная функция 2. Нефиксированный диапазон значений функции — [-∞,+∞]
Sigmoid	1. Нелинейная функция 2. В диапазоне значений сигнала от -2.5 до 2.5 значения активации меняются очень быстро 3. Фиксированный диапазон значений функции — [0,1]	1. Затухают градиенты
Tanh	1. Нелинейная функция 2. В диапазоне значений сигнала от -2.5 до 2.5 значения активации меняются очень быстро 3. Фиксированный диапазон значений функции — [-1,1]	1. Затухают градиенты
ReLU	1. Меньше времени на вычисления 2. Меньше вес у сети 3. Нет проблемы затухающих градиентов	1. Нулевые градиенты у сигналов, которые меньше 0 2. Нефиксированный диапазон значений функции
Leaky ReLU	1. Нет проблемы нулевых градиентов	1. Нужно подбирать значение alpha 2. Нефиксированный диапазон значений функции