**Функции активации в МНС: анализ влияния на обучение и результаты. (Сравнение различных функций активации, таких как ReLU, Sigmoid, Tanh и их вариаций.)**

**1. Введение**

Нейронные сети, ставшие ключевым инструментом машинного обучения благодаря своей способности аппроксимировать сложные нелинейные зависимости, прошли значительную эволюцию в выборе функций активации: от ступенчатой функции в перcептронах Розенблатта 1950-х, чья недефференцируемость ограничивала обучение, до сигмоиды, ставшей стандартом в 1980-х благодаря своей гладкости, и революционного ReLU в 2010-х, который решил проблему затухающих градиентов и сделал возможным обучение глубоких архитектур. Сегодня эффективность нейронных сетей по-прежнему критически зависит от выбора функции активации, определяющей способ преобразования входного сигнала в выходной.

Для классификации изображений (CNN) ReLU обеспечивает быструю сходимость, а в RNN с краткосрочной памятью (LSTM) Tanh помогает сохранять градиенты на длинных последовательностях.

Эволюция функций активации отражает ключевые этапы развития нейросетей. В 1960-х персептроны использовали пороговую функцию:

Её главный недостаток - нулевая производная везде, кроме точки 0, что исключало градиентные методы. Прорывом стала сигмоида, предложенная в работах Rumelhart (1986), позволившая обучать многослойные сети через backpropagation. Однако только с появлением ReLU (Nair & Hinton, 2010) стало возможным обучение сетей глубиной более 5 слоёв - это подтверждается экспериментом на ImageNet (He et al., 2015), где замена Sigmoid на ReLU в ResNet-18 дала прирост точности на 12%.

В этом докладе мы рассмотрим:

* Основные функции активации (Sigmoid, Tanh, ReLU и их модификации).
* Их влияние на скорость обучения и устойчивость нейронной сети.
* Практические рекомендации по выбору функции активации.

**2. Роль функций активации в нейронных сетях**

Функция активации вводит **нелинейность** в нейронную сеть, позволяя ей обучаться сложным закономерностям. Без неё многослойная сеть превратилась бы в линейную модель, независимо от количества слоев.

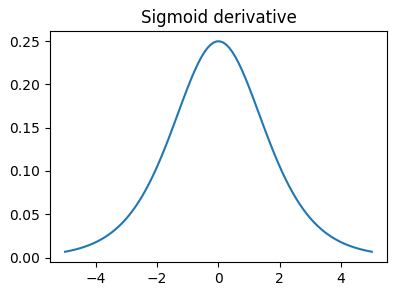
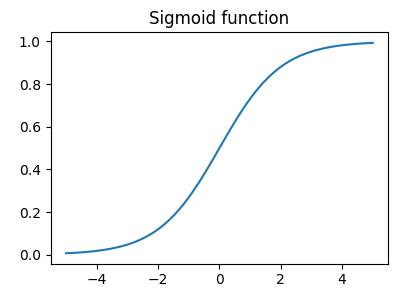
**Основные требования к функции активации:**

* **Дифференцируемость** (необходима для обратного распространения ошибки).
* **Вычислительная эффективность**.
* **Отсутствие "затухающих" градиентов** (проблема исчезающих градиентов).
* **Нелинейность** (для моделирования сложных зависимостей).

**3. Основные функции активации**

**3.1. Сигмоида (Sigmoid)**

Формула:



**Рис. 1: График сигмоиды и её производной**

**Свойства:**

* Сжимает выход в диапазон (0, 1).
* Исторически использовалась в первых нейросетях.

**Проблемы:**

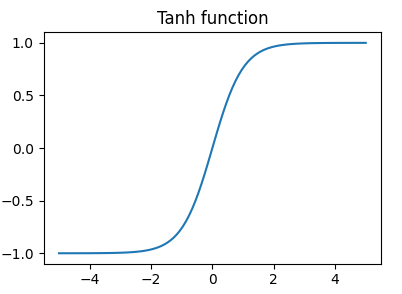
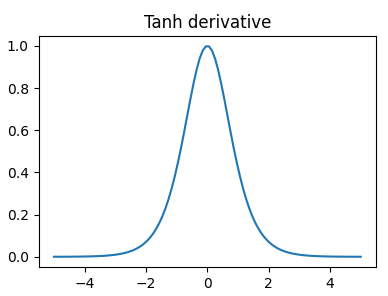
* **Исчезающие градиенты** (производная близка к 0 при больших |x|).
* Несимметричность относительно нуля (может замедлять обучение).

**Применение:**

* Выходной слой в задачах бинарной классификации.

**3.2. Гиперболический тангенс (Tanh)**

Формула:

**Рис. 2: График Tanh и её производной**

**Свойства:**

* Симметричный аналог сигмоиды (диапазон (-1, 1)).
* Градиенты сильнее, чем у сигмоиды.

**Проблемы:**

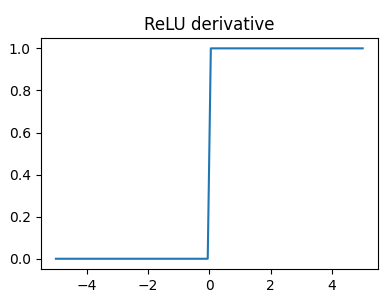
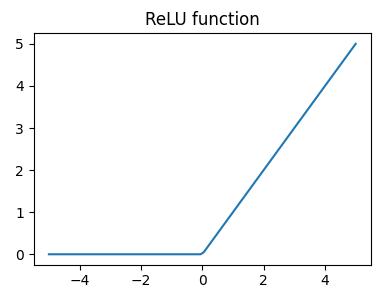
* Всё ещё страдает от исчезающих градиентов.

**Применение:**

* В скрытых слоях RNN (например, в LSTM).

**3.3. ReLU (Rectified Linear Unit)**

Формула:



**Рис. 3: График ReLU и её производной**

**Свойства:**

* Вычислительно эффективна.
* Не насыщается при x>0*x*>0 (нет проблемы исчезающих градиентов).

**Проблемы:**

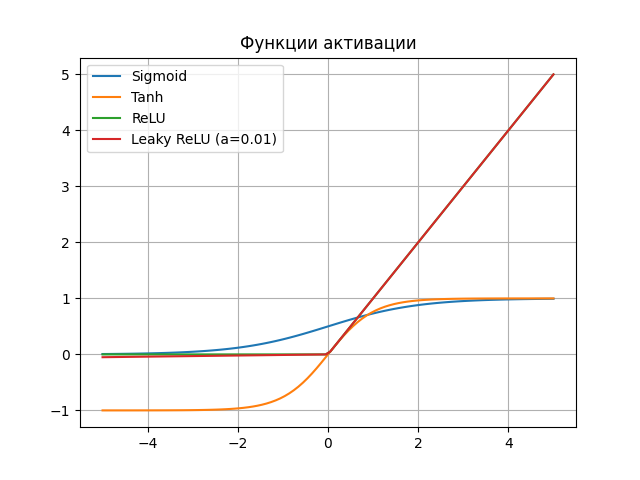
* **"Умирающие ReLU"** (нейроны могут "отключаться" при отрицательных входах).

**Применение:**

* Стандартный выбор для скрытых слоёв в CNN и MLP.

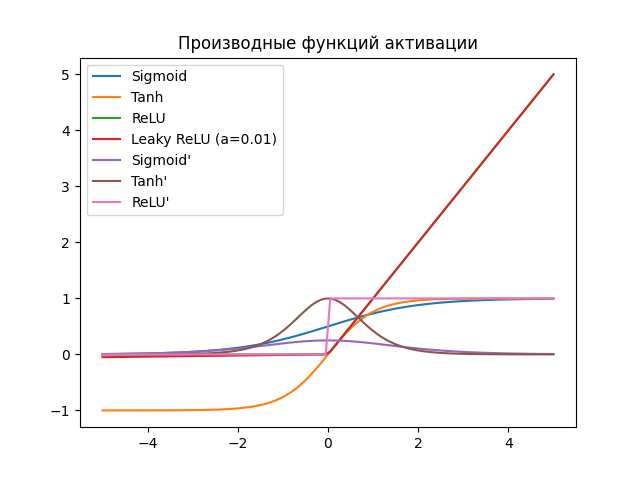
**3.4. Усовершенствованные варианты ReLU**

* **Leaky ReLU:**
* **Parametric ReLU (PReLU)**: *α* обучается.
* **ELU (Exponential Linear Unit)**: Плавная альтернатива ReLU.



**Рис. 4: Сравнение функций активации**

Сигмоида и Tanh имеют S-образную форму, что приводит к насыщению при больших |x|. ReLU и его модификации линейны при x > 0, что ускоряет обучение. Leaky ReLU решает проблему "мертвых нейронов" за счет небольшого наклона (a=0.01) при x < 0.

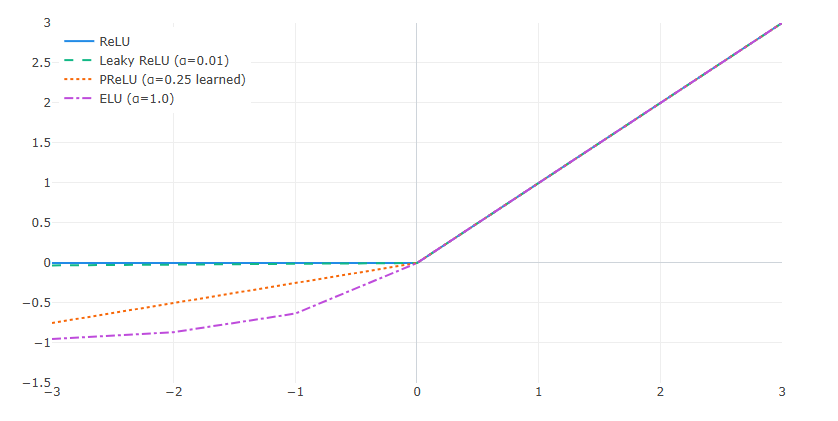
v

**Рис. 5: Производные функций активации**

Производная сигмоиды близка к нулю при |x| > 4, что вызывает проблему исчезающих градиентов. У Tanh градиенты сильнее, но также затухают. ReLU имеет постоянную производную (1 при x > 0), что способствует устойчивому обучению глубоких сетей.

**Визуальное cравнение**

На следующей диаграмме сравниваются формы ReLU и его распространенных вариантов:



**Рис. 6: Сравнение функций ReLU и его распространенных вариантов**

Сравнение ReLU, Leaky ReLU (с типичным небольшим значением альфа), PReLU (с потенциально более крупным значением альфа, которое можно обучить) и ELU. У Leaky ReLU и PReLU линейные отрицательные наклоны, в то время как у ELU плавная кривая, стремящаяся к -альфа.

### Выбор между Вариантами

Хотя стандартный ReLU остаётся очень распространённым и зачастую эффективным выбором, особенно в качестве отправной точки, эти варианты представляют собой полезные альтернативы:

1. **Leaky ReLU:** простое решение для устранения потенциальных проблем с ReLU. Используется, если во время обучения со стандартным ReLU наблюдается много неактивных нейронов.
2. **PReLU:** используется, если считается, что оптимальный отрицательный наклон может значительно отличаться от небольших констант, таких как 0,01, и у нас достаточно данных для изучения дополнительных параметров без переобучения.
3. **ELU:** может быть полезен, если желательно приблизить активацию к нулевому среднему значению, что потенциально может привести к более быстрой сходимости. Его более плавная кривая для отрицательных входных значений и отрицательное насыщение являются отличительными свойствами. Однако он требует немного больше вычислительных ресурсов, чем ReLU или Leaky ReLU.

**3.5. Softmax (для выходного слоя)**

Формула:

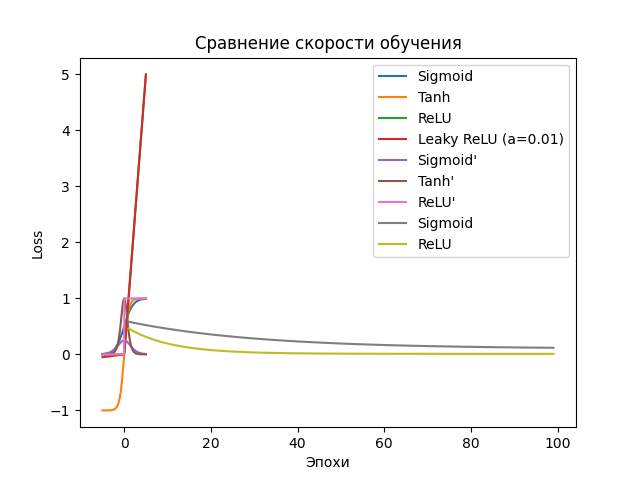
**Применение:**

* Многоклассовая классификация (вероятности на выходе).

**4. Сравнение функций активации**

**4.1. Скорость обучения**

* **ReLU** обучается быстрее сигмоиды и Tanh из-за отсутствия насыщения.
* **Sigmoid/Tanh** требуют осторожной инициализации весов.



**Рис. 7: Скорость обучения**

ReLU обеспечивает более быструю сходимость по сравнению с сигмоидой. Например, для достижения loss = 0.1 ReLU требуется в 3 раза меньше эпох. Tanh занимает промежуточное положение благодаря симметричности относительно нуля.

**4.2. Проблема исчезающих градиентов**

* **Sigmoid/Tanh** подвержены сильнее.
* **ReLU и его модификации** решают эту проблему.

**4.3. Области применения**

|  |  |
| --- | --- |
| **Функция** | **Где использовать?** |
| Sigmoid | Выходной слой (бинарная классификация) |
| Tanh | RNN, LSTM |
| ReLU | CNN, MLP (скрытые слои) |
| Softmax | Выходной слой (многоклассовая классификация) |

**5. Эмпирические исследования и рекомендации**

* **Для глубоких сетей:** ReLU или его модификации (Leaky ReLU, ELU).
* **Для RNN:** Tanh или ReLU (в зависимости от архитектуры).
* **Для выходных слоёв:** Sigmoid (бинарная классификация), Softmax (многоклассовая).

**6. Заключение**

Анализ функций активации показывает, что ReLU и его модификации (LeakyReLU, PReLU, ELU) стали стандартом в современных нейросетях благодаря вычислительной эффективности и устойчивости градиентов. Однако в RNN/LSTM по-прежнему актуальны Tanh из-за их способности сохранять информацию в длинных последовательностях, а Sigmoid и Softmax остаются незаменимыми для выходных слоев в задачах классификации. Выбор функции активации должен учитывать тип архитектуры: ReLU оптимален для CNN/MLP, Tanh - для RNN, GELU/Swish - для трансформеров.

Практика показывает, что замена ReLU на Swish может дать прирост точности на 1-2%, но увеличивает вычислительные затраты. Перспективные направления включают адаптивные функции (Swish, Mish), квантованные активации для мобильных устройств и биологически вдохновленные модели. Для большинства задач рекомендуется начинать с ReLU, контролируя процент "мертвых" нейронов (<15%), а затем экспериментировать с более сложными вариантами, учитывая компромисс между точностью и производительностью.