Лабораторная работа 3. Функции активации

Что такое функция активации

Значение текущего нейрона вычисляется на основе значений предыдущих нейронов, умноженных на коэффициенты. Такие коэффициенты называют **весами**, а процесс вычисления нового значения – **взвешенной суммой.**

$$y = \sum_{i=1}^{n}(w_i \cdot x_i) + b$$

где w_i – веса (коэффициенты), x_i – значения входов для текущего нейрона и b – некоторое смещение (bias). Упрощенная запись:

$$y = Wx + b$$

Результат y может принимать любые значения $(-\infty; +\infty)$. Перед нами возникает две проблемы:

1. Нейрон не понимает, активирован он или нет. Нам нужно задать границы, в которых нейрон будет активироваться
2. В более сложных задачах требуется использовать несколько слоев. Но комбинацией линейных функций является **линейная функция**:

$$y = W_2(W_1x + b_1) + b_2$$
$$y = Wx + b$$

   Сколько бы слоев мы не сделали – модель остается линейной. Другими словами такую многослойную модель можно представить однослойной.

Однако только **нелинейные** функции позволяют нейронным сетям решать нетривиальные задачи с использованием нескольких слоев.
Для того, чтобы модель стала нейлинейной, на выходе каждого слоя добавляют **функцию активации.**

- **Функция активации** – это математическая функция, которая преобразует выходной сигнал нейрона и добавляет модели нелинейность. Использование функции активации также называют добавлением **слоя активации.**

Функции активации

№	Название	Обозначение	Формула	Название для кода
1	Функция Хэвисайда (ступенчатая функция)	θ	$\theta(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$	`heaviside`
2	Rectified Linear Unit	$ReLU$	$ReLU(x) = max(0, x)$	`relu`
3	Leaky ReLU	$LReLU$	$LReLU(x) = \begin{cases} x, x \geq 0 \\ \alpha \cdot x, x < 0 \end{cases}$	`leaky_relu`
4	Жесткий гиперболический тангенс	$thHard$	$thHard(x) = \begin{cases} -1, x < -1 \\ x, x \in [-1; 1] \\ 1, x > 1 \end{cases}$	`hard_tanh`
5	Нормированный арктангенс	$arctg_{norm}$	$arcth_{norm}(x) = \frac{2}{\pi} \cdot arctg(x)$	`norm_atan`
6	Плавная функция знака	$SoftSign$	$SoftSign(x) = \frac{x}{1+\|x\|}$	`soft_sign`
7	Сигмоида (логистическая функция)	σ	$\sigma(x) = \frac{1}{1+e^{-x}}$	`sigmoid`
8	Swish (Sigmoid Linear Unit)	$SiLU$	$SiLU(x) = x \cdot \sigma(x)$	`swish`
9	SoftPlus	$SoftPlus$	$SoftPlus(x) = ln(1 + e^x)$	`soft_plus`
10	Гиперболический тангенс	th	$th(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	`tanh`
11	Exponential Linear Unit	ELU	$ELU(x) = \begin{cases} x, x \geq 0 \\ \alpha(e^x - 1), x < 0 \end{cases}$	`elu`
12	Continuously Differentiable Exponential Linear Unit	$CELU$	$CELU(x) = \begin{cases} x, x \geq 0 \\ \alpha(e^{x\alpha} - 1), x < 0 \end{cases}$	`celu`

Комментарии к функциям:

- Для функции $LReLU$ (№3) α – малая константа (например, 0.01)
- Для функции $SiLU$ (№8) σ – сигмоида (то есть функция №7)
- Для функций ELU и $CELU$ (№11 и №12) α – положительное число (например, 1)

Задание 1

Реализуйте все 12 функций активации в Python. Например:

```
def heavyside(x):
    ...
    return y
```

Варианты для следующих заданий

Вариант	1-ая функция	2-ая функция
1	5	11
2	2	7
3	6	9
4	1	12
5	4	8
6	3	10
7	6	7
8	1	8
9	5	10
10	2	11
11	4	12
12	3	9
13	1	7
14	6	12
15	2	9
16	5	8
17	3	11
18	4	10
19	6	8
20	1	10
21	5	7
22	2	12
23	3	8
24	4	11
25	1	9

Вариант	1-ая функция	2-ая функция
26	6	11
27	2	10
28	5	12
29	3	7
30	4	9
31	1	11
32	6	10
33	2	8
34	5	9
35	3	12
36	4	7

Задание 2

1. Аналитически вычислите производные функций активации, соответствующих вашему варианту.
2. Реализуйте данные производные функции на Python

Задание 3

1. Постройте графики для функций, соответствующих вашему варианту, а также их производных на интервале $x \in [-5; 5]$
2. Для каждой функции активации по графику определить ее область значений

Задание 4

1. Сгенерируйте входной массив x размером 10^6 случайных чисел из стандартного нормального распределения
2. Замерьте время выполнения (используйте `time.perf_counter()`) при вычислении функции поэлементно (цикл `for`) и при использовании векторизованных операций `numpy`
3. Повторите замеры 10 раз и усредните
4. Постройте столбчатую диаграмму времени выполнения для каждой функции и каждого способа

5. Сравните, какая из двух функций, соответствующих вашему варианту, работает быстрее при вычислении функции поэлементно (цикл `for`) и при использовании векторизованных операций `numpy`