курс «Глубокое обучение»

Нейронные сети

Александр Дьяконов

10 февраля 2020 года

План

Простейшая нейросеть – 1 нейрон

Функции активации

(линейная, пороговая, сигмоида, гиперболический тангенс, softmax, LeakyReLU, ELU, Maxout)

Функциональная выразимость нейрона

Теорема об универсальной аппроксимации

Сеть прямого распространения

Обучение

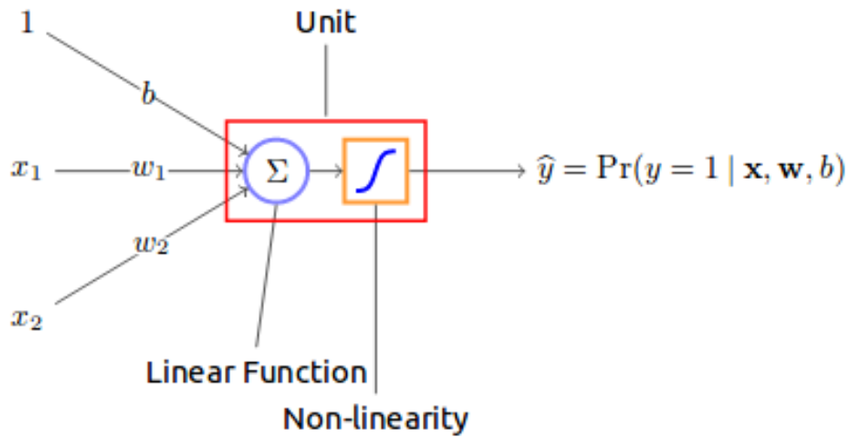
Функции ошибки

Производные на компьютере

Проблема затухания градиента

Обратное распространение градиента

Простейшая нейросеть – 1 нейрон



Разделяющая поверхность – линейная

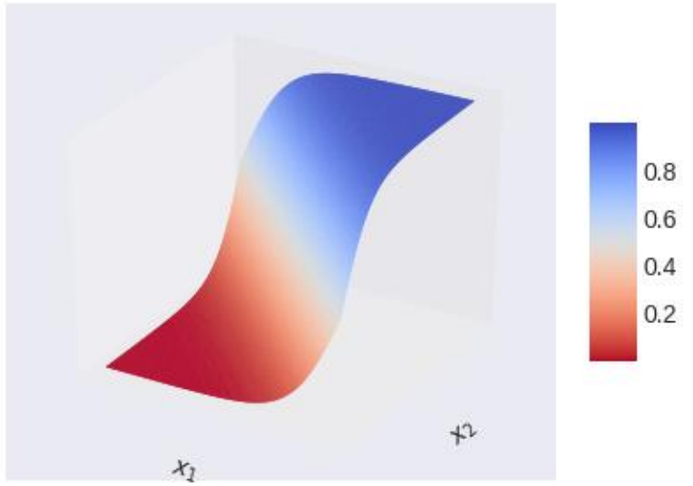
$$a(x) = b + w_1 x_1 + \ldots + w_n x_n = \sum_{t=0}^{n} w_t x_t$$

$$h(x) = \sigma(a(x))$$

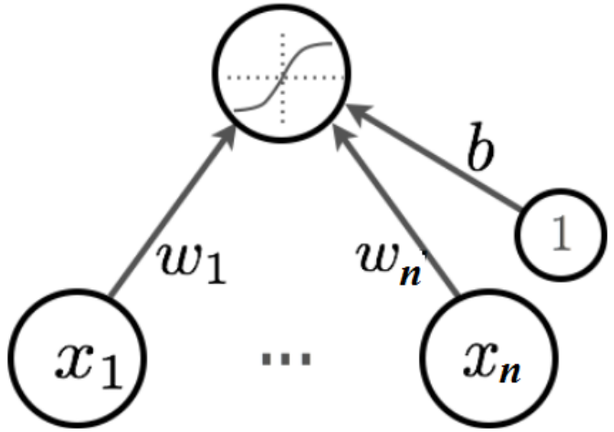
b – **смещение**

σ – **функция активации**

w_t – **веса связей**



Линейные модели – нейросети!



Линейная регрессия

$$a(x) = b + w_1 x_1 + \ldots + w_n x_n$$

Логистическая регрессия

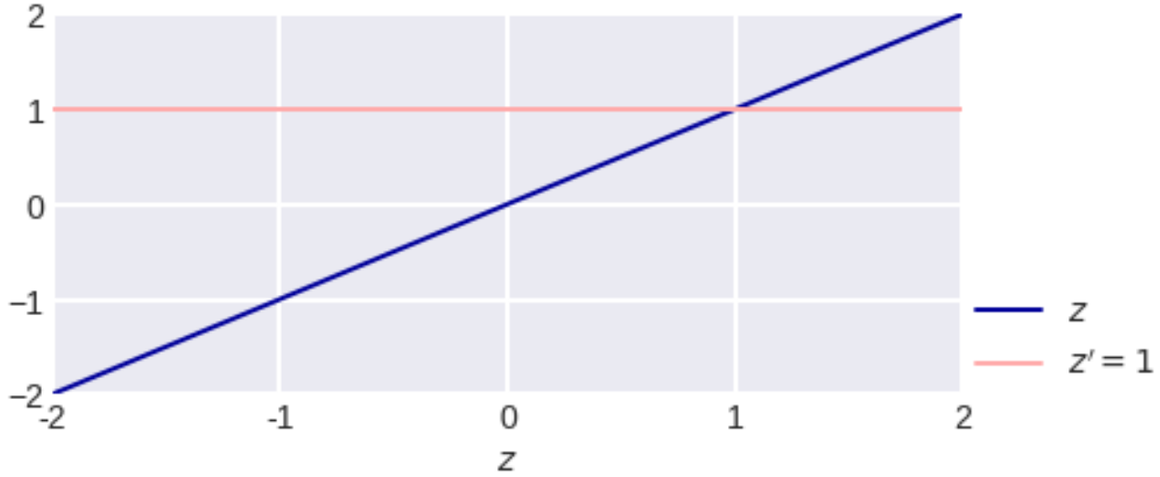
$$a(x) = \sigma(b + w_1 x_1 + \ldots + w_n x_n)$$

Линейный классификатор

$$a(x) = \text{th}(b + w_1 x_1 + \ldots + w_n x_n)$$

Функции активации

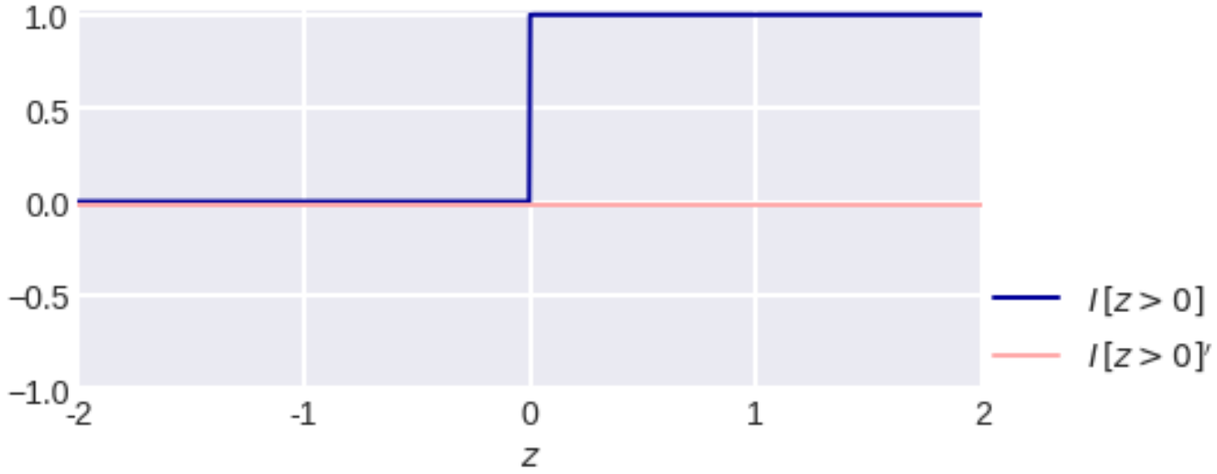
Тождественная функция (линейная / linear activation function)



$$f(z) = z$$

$$\frac{\partial f(z)}{\partial z} = 1$$

Пороговая функция (threshold function)

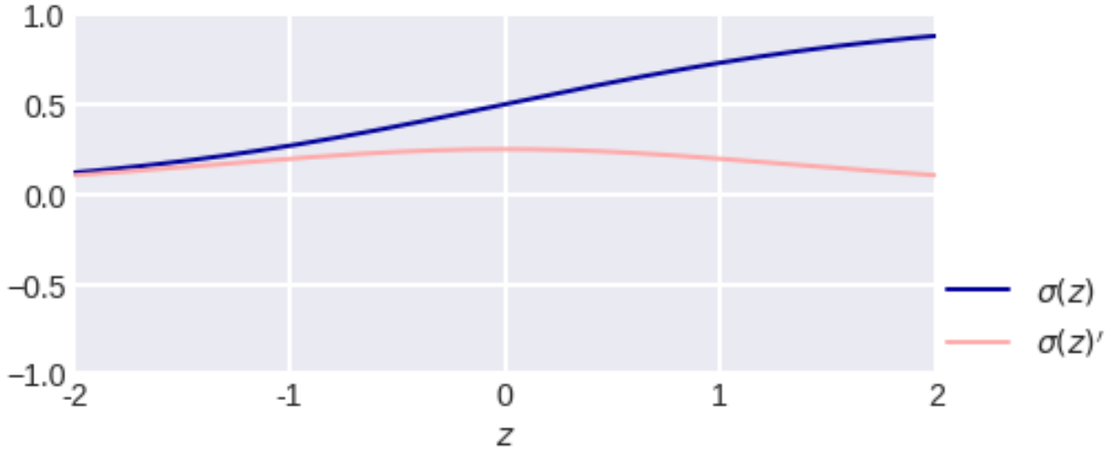


$$\text{th}(z) = I[z > 0]$$

$$\frac{\partial \text{th}(z)}{\partial z} = 0$$

Функции активации

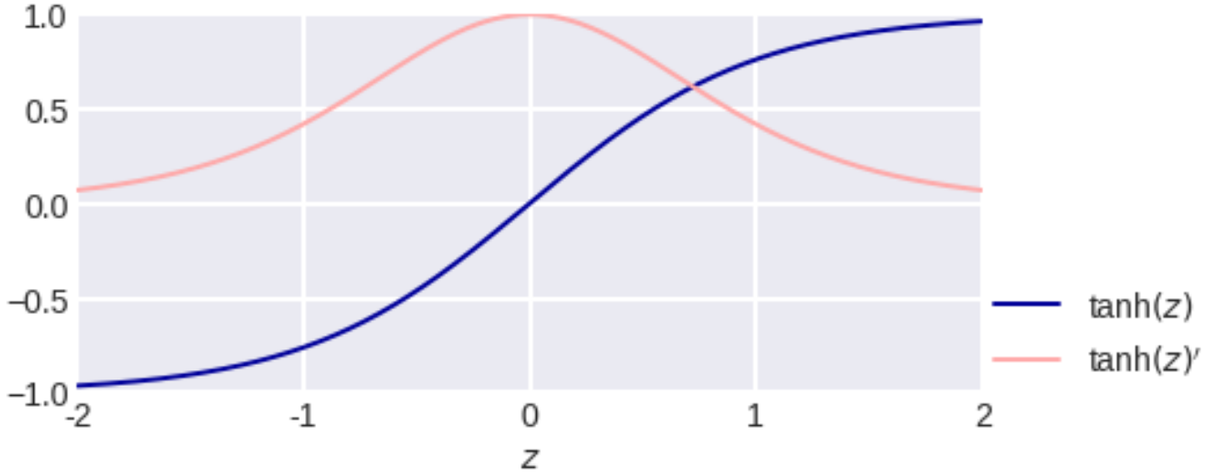
Сигмоида (sigmoid activation function)



$$\sigma(z) = \frac{1}{1+e^{-z}} \in (0,1)$$

$$\frac{\sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z)) > 0$$

Гиперболический тангенс (hyperbolic tangent)



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

$$\frac{\partial \tanh(z)}{\partial z} = 1 - \tanh^2(z)$$

Функции активации
в задачах классификации

$$\text{softmax}(z_1,\ldots,z_k) = \frac{1}{\sum_{t=1}^{k}\exp(z_t)}(\exp(z_1),\ldots,\exp(z_k))^{\text{T}}$$

сумма выходов = 1

выходы интерпретируются как вероятности

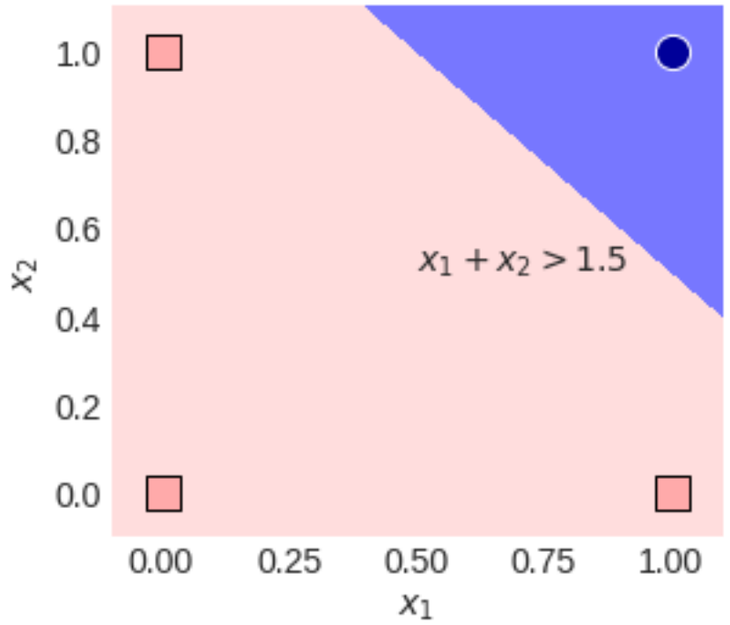
$$[0.5,\ 0.5,\ 0.1,\ 0.7] \to [0.257,\ 0.257,\ 0.172,\ 0.314]$$

$$[-1.0,\ 0,\ 1.0,\ 0,\ -1.0] \to [0.07,\ 0.18,\ 0.5,\ 0.18,\ 0.07]$$

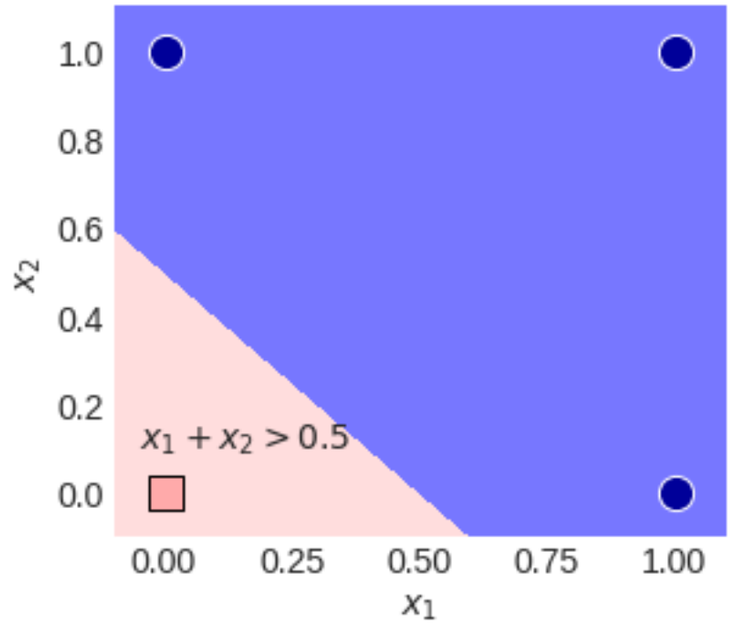
$$[1.0,\ 1.0,\ 1.0,\ 2.0,\ 1.0] \to [0.15,\ 0.15,\ 0.15,\ 0.4,\ 0.15]$$

Что может один нейрон

Логическое И



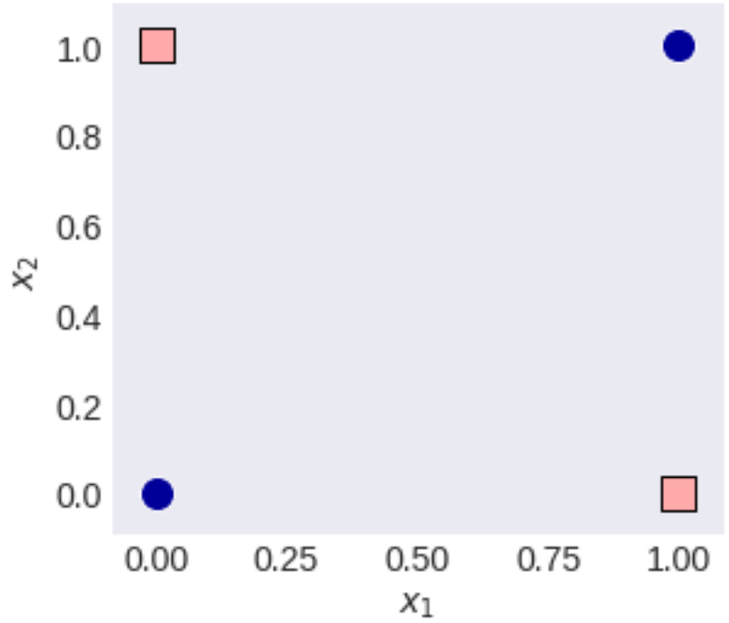
Логическое ИЛИ



Для простоты – пороговая функция активации

Что НЕ может один нейрон

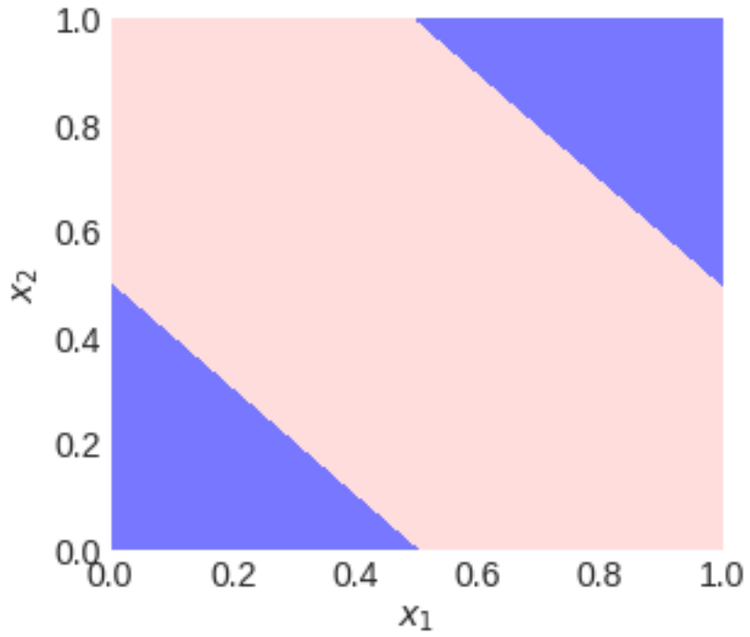
Исключающее ИЛИ

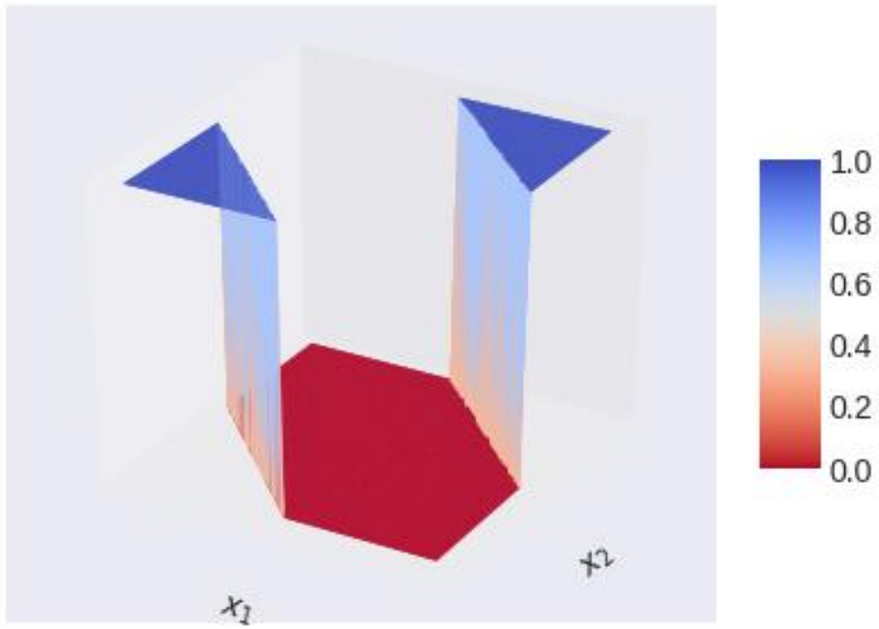


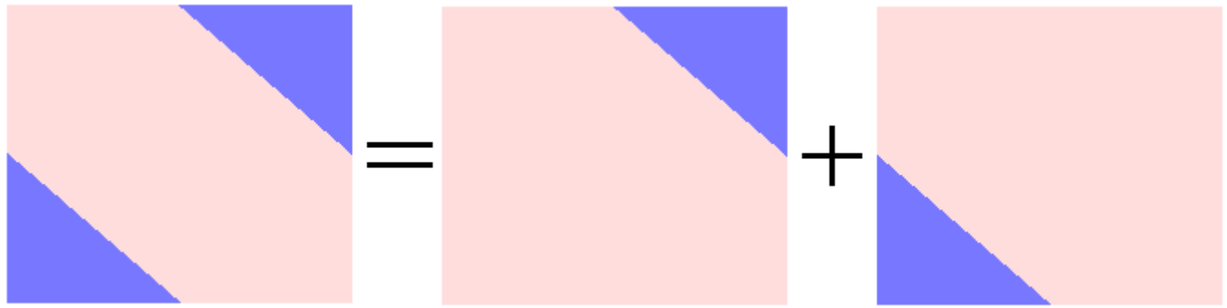
$$\text{th}(\text{th}(x_1 + x_2 - 1.5) + \text{th}(-x_1 - x_2 + 0.5) - 0.5)$$

$$\text{th}(\text{th}(0+0-1.5) + \text{th}(-0-0+0.5) - 0.5) = \text{th}(0+1-0.5) = 1$$
$$\text{th}(\text{th}(0+0-1.5) + \text{th}(-0-1+0.5) - 0.5) = \text{th}(0+0-0.5) = 0$$
$$\text{th}(\text{th}(0+0-1.5) + \text{th}(-1-0+0.5) - 0.5) = \text{th}(0-0-0.5) = 0$$
$$\text{th}(\text{th}(1+1-1.5) + \text{th}(-1-1+0.5) - 0.5) = \text{th}(1+0-0.5) = 1$$

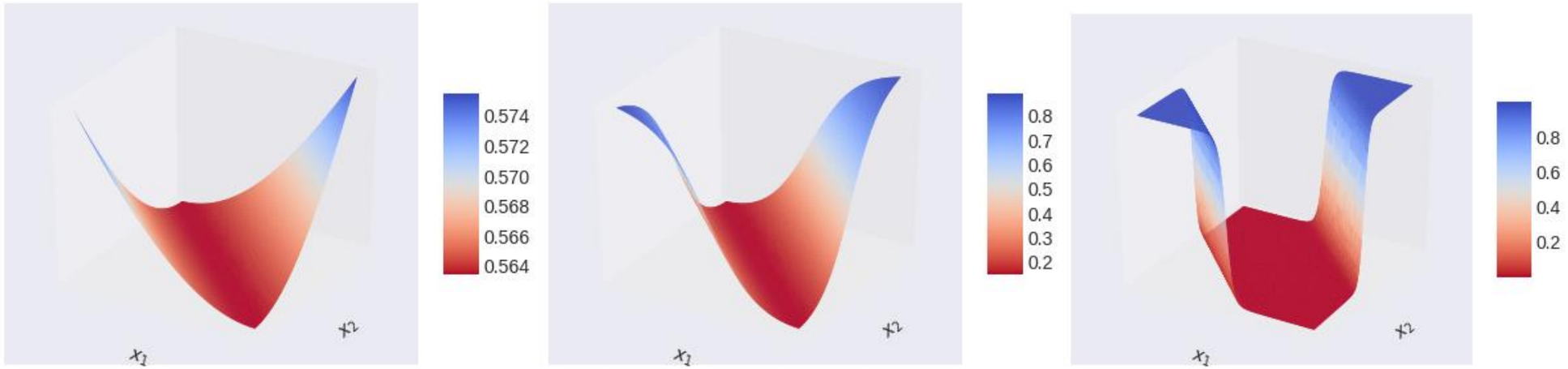
Что НЕ может один нейрон







Сигмоида стремится к пороговой функции



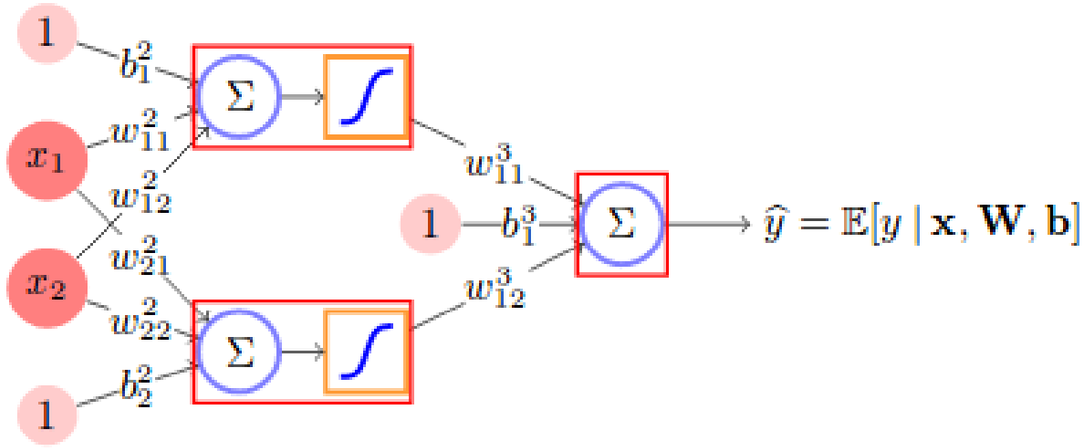
$$\sigma_c(\sigma_c(x_1 + x_2 - 1.5) + \sigma_c(-x_1 - x_2 + 0.5) - 0.5)$$

$$\sigma_c(z) = \frac{1}{1 + e^{-cz}}$$

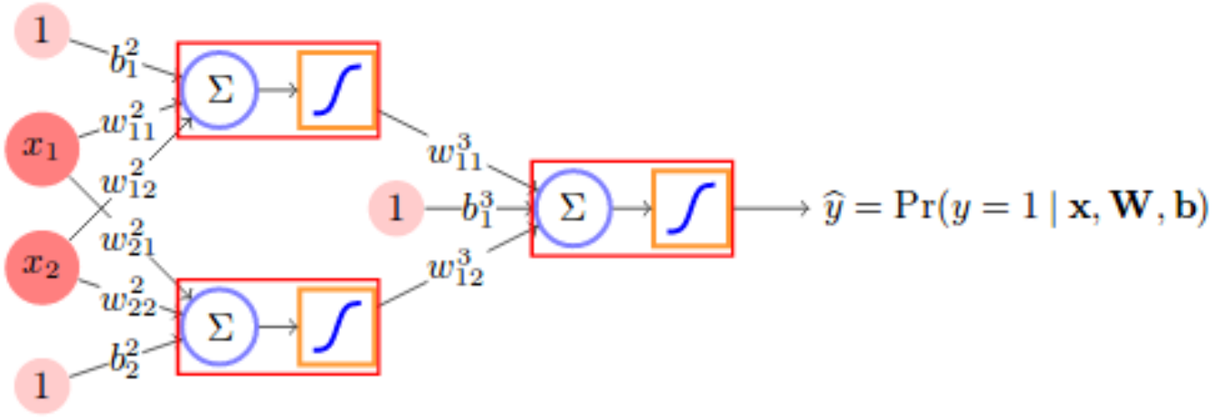
- **Сигмоиду проще обучать – дифференцируемая**
  - **Есть возможность получать «вероятности»**

Двуслойная нейронная сеть

Регрессия



Классификация



Такой нейронной сети хватит…

Теорема об универсальной аппроксимации [Hornik, 1991]

Любую непрерывную функцию можно с любой точностью приблизить нейросетью глубины 2 с сигмоидной функцией активации на скрытом слое и линейной функции на выходном слое

Нейросеть глубины два с фиксированной функцией активации в первом слое и линейной функцией активации во втором может равномерно аппроксимировать (м.б. при увеличении числа нейронов в первом слое) любую непрерывную функцию на компактном множестве тогда и только тогда, когда функция активации неполиномиальная.

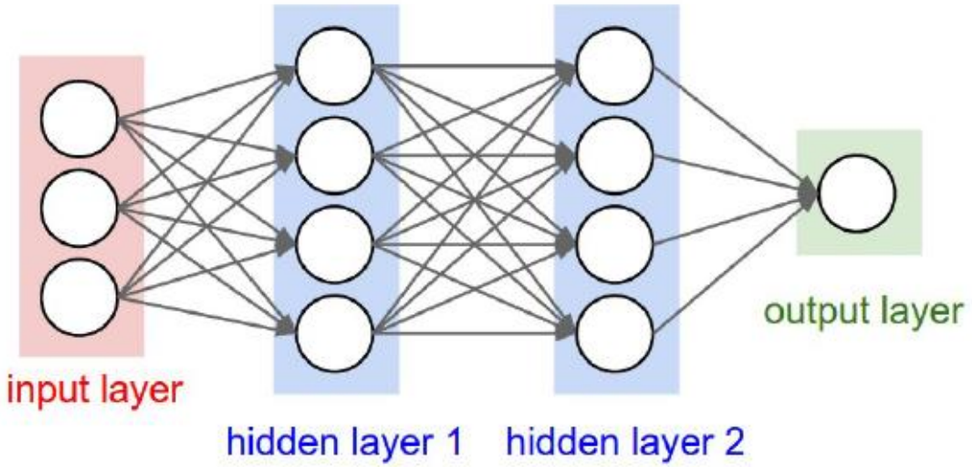
http://www2.math.technion.ac.il/~pinkus/papers/neural.pdf

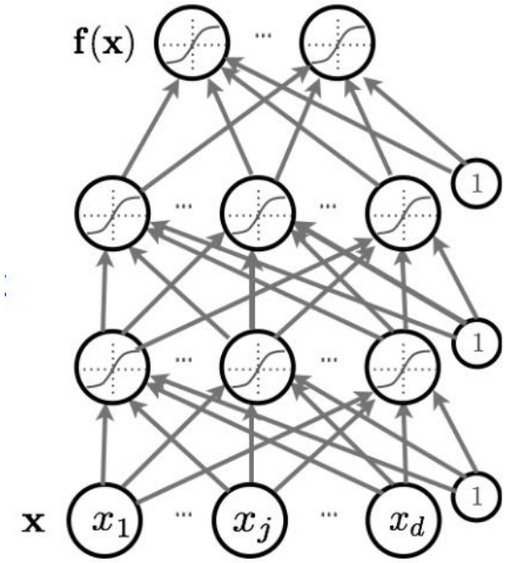
Более того, функция активации м.б. любая (неполиномиальная)!

Но…

- **много нейронов (неизвестно сколько)**
- **экспоненциальные веса**
- **сложность обучения**

Многослойная нейронная сеть – пример нелинейной модели





Ориентированный граф вычислений

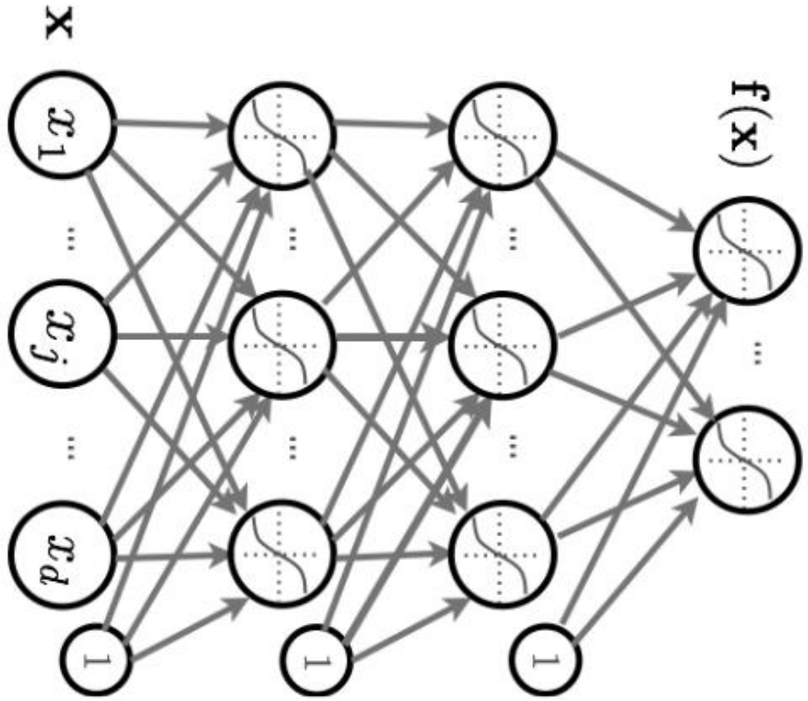
Вершины – переменные или нейроны

Рёбра – зависимости

Сеть прямого распространения – Feedforward Neural Network

(т.е. нет циклов)

все нейроны предыдущего слоя связаны с нейронами следующего



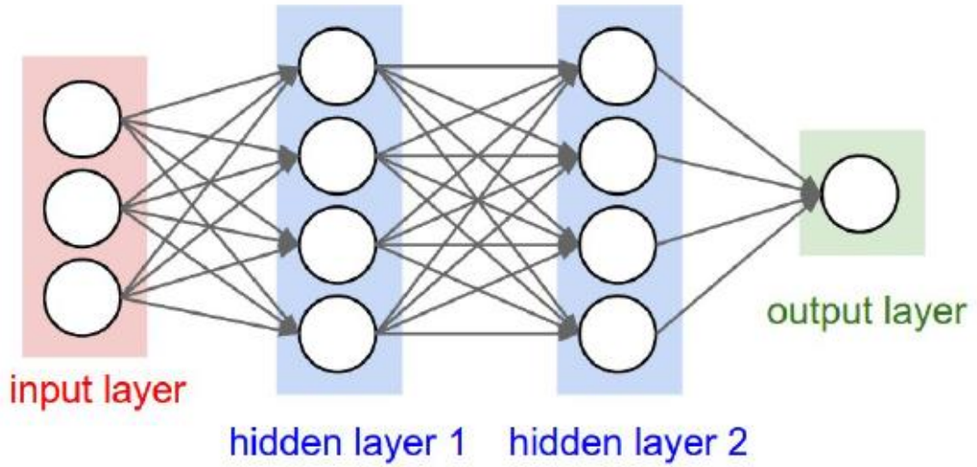
входной слой

один или несколько скрытых слоёв

выходной слой

Важная аналогия

Глубокая НС – последовательное преобразование признакового пространства



$$\varphi_k(W_k \cdot \ldots \cdot \varphi_2(W_2 \cdot \varphi_1(W_1 \cdot x)))$$

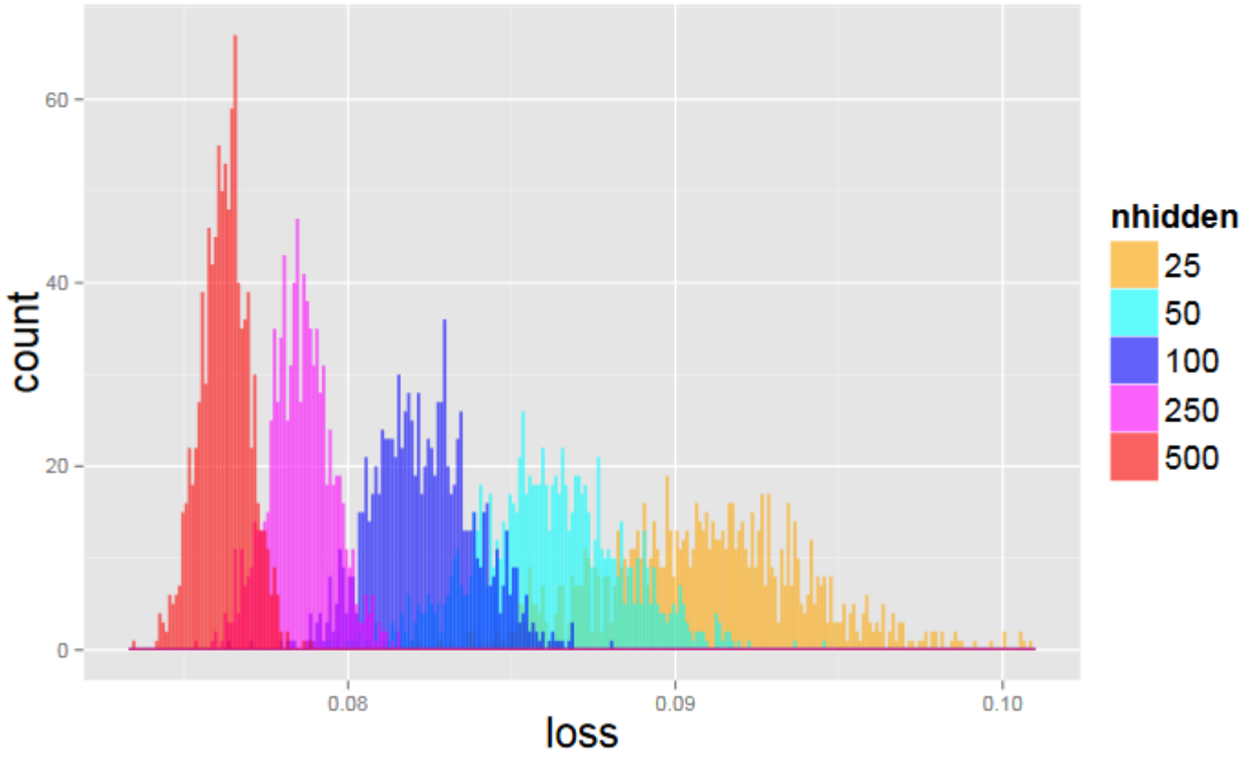
иногда чуть другая запись!

Сейчас наука DL, в основном, как правильно представлять (преобразовывать) признаковые пространства

увидим потом и в таких моделях, как кодировщик

нельзя просто преобразовывать… надо что-то ещё требовать

Зачем нужны глубокие нейронные сети



Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, Yann LeCun «The Loss Surfaces of Multilayer Networks» 2015, https://arxiv.org/abs/1412.0233

Обучение

Как принято…

минимизация регуляризованного эмпирического риска

$$\frac{1}{m}\sum_{t=1}^{m} L(a(x_i \mid w), y_i) + \lambda R(w) \to \min_w$$

Задача оптимизации невыпуклая!

«Настройка» нейронной сети – получение весов w

Метод стохастического градиента

$$w^{(t+1)} = w^{(t)} - \eta \nabla[L(a(x_i \mid w^{(t)}), y_i) + \lambda R(w^{(t)})]$$

т.к. очень много слагаемых… и так быстрее;)

Метод стохастического градиента

1. Случайная инициализация весов $w^{(0)} \sim \text{norm}(0, \sigma^2)$

2. Цикл по t до сходимости

2.1. Выбираем случайный объект x_i

2.2. Вычисляем градиент $\nabla[L(a(x_i \mid w^{(t)}), y_i) + \lambda R(w^{(t)})]$

2.3. Адаптация весов $w^{(t+1)} = w^{(t)} - \eta \nabla[L(a(x_i \mid w^{(t)}), y_i) + \lambda R(w^{(t)})]$

почему такая инициализация?

Функции ошибки

Классификация – logloss (CrossEntropyLoss)

$$L((a_1,\ldots,a_l),y) = -\log\frac{\exp(a_y)}{\sum_{j=1}^{l}\exp(a_j)} = -a_y + \log\sum_{j=1}^{l}\exp(a_j)$$

– Часто при реализации делают так:

$$-a_y + \max\{a_j\} + \log\left(\sum_{j=1}^{l}\exp(a_j - \max\{a_j\})\right)$$

Обратное распространение (Backpropagation)

Идея: вычисление производной сложной функции

$$\nabla f(w, g(w), h(w)) = \frac{\partial f}{\partial w} + \frac{\partial f}{\partial g}\nabla g(w) + \frac{\partial f}{\partial h}\nabla h(w)$$

Автоматическое дифференцирование

Прямое распространение

$$x, w \to f(x, w, g(x, w), h(x, w))$$

вычисление ответов, функции ошибки

Обратное распространение

$$x, w, \nabla g, \nabla h \to \nabla f$$

вычисление градиентов

Производные на компьютере

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind
«Automatic differentiation in machine learning: a survey» 2015-2018
https://arxiv.org/abs/1502.05767

Функции активации

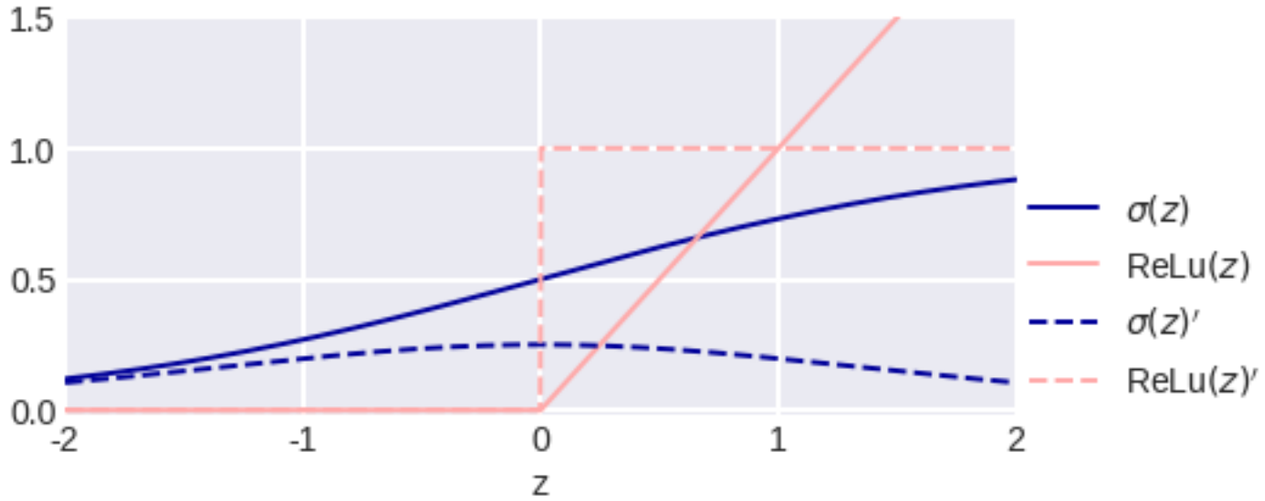
Проблема – затухание градиента

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

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

$$\mathrm{ReLU}(z) = \max(0, z)$$

$$\frac{\partial \mathrm{ReLU}(z)}{\partial z} = I[z > 0]$$



Rectified Linear Unit

Что плохого в сигмоиде

- «убивает» градиенты
- выходы не отцентрированы (легко устранить $\rightarrow$ tanh)
- вычисление экспоненты всё-таки дорого…

Ещё функции активации

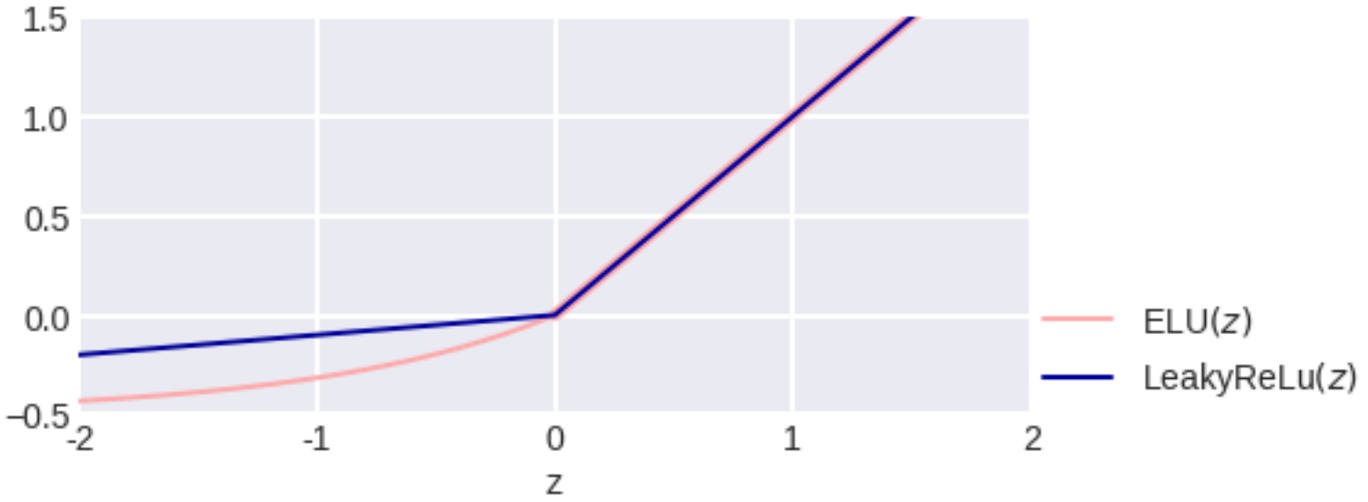
$$\text{LeakyReLU}(z) = \max(0.1z, z)$$

Exponential Linear Unit

$$\text{ELU}(z) = \begin{cases} z, & z \geq 0, \\ \alpha(e^z - 1), & z < 0. \end{cases}$$

Scaled Exponential Linear Unit

$$\text{SELU}(z) = \lambda \text{ELU}(z)$$



Ещё функции активации

Gaussian Error Linear Unit

(Google's BERT and OpenAI's GPT-2)

$$\mathrm{GELU}(z) = \frac{z}{2}\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}(z + \alpha z^3)\right)\right)$$

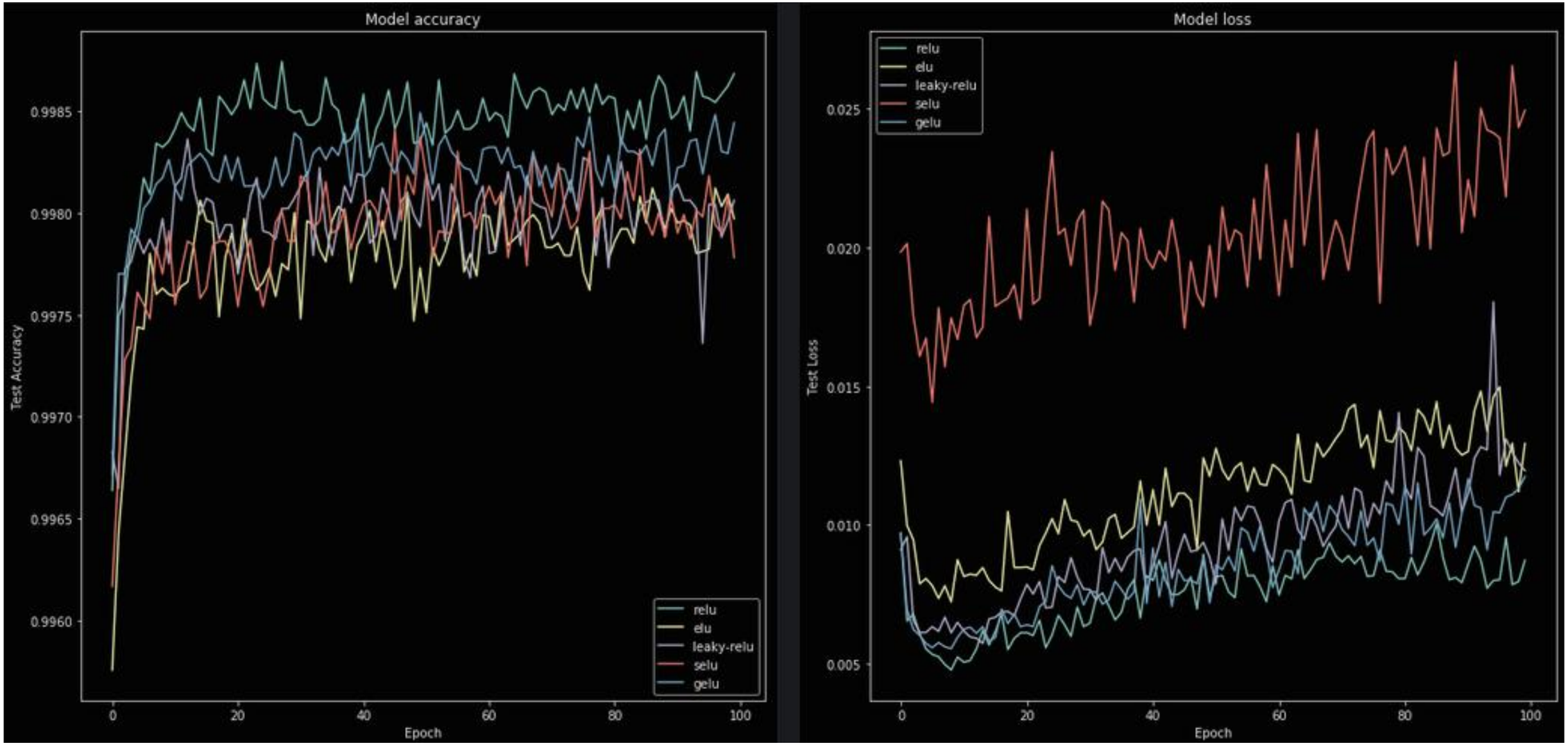
$$\alpha = 0.044715$$

Maxout

$$\mathrm{Maxout}(z) = \max(w^{\mathrm{T}} z + w_0, v^{\mathrm{T}} z + v_0)$$

не совсем функция активации, т.к. есть параметры

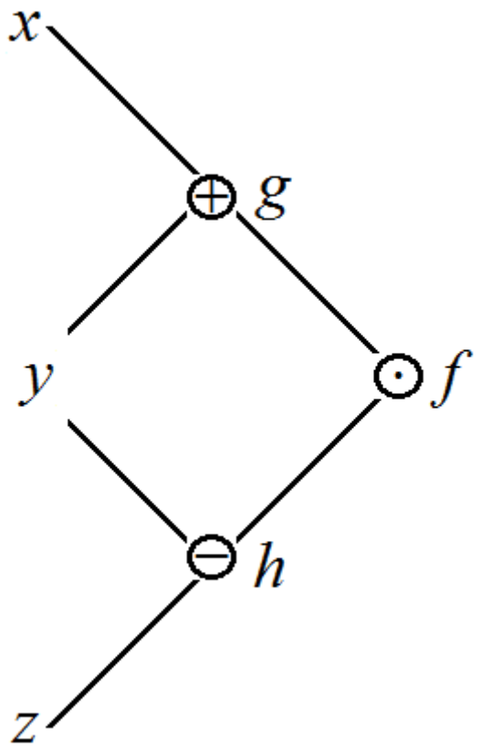
Функции активации



https://mlfromscratch.com/activation-functions-explained/

Вычисление градиента

$$f = (x+y)\cdot(y-z)$$



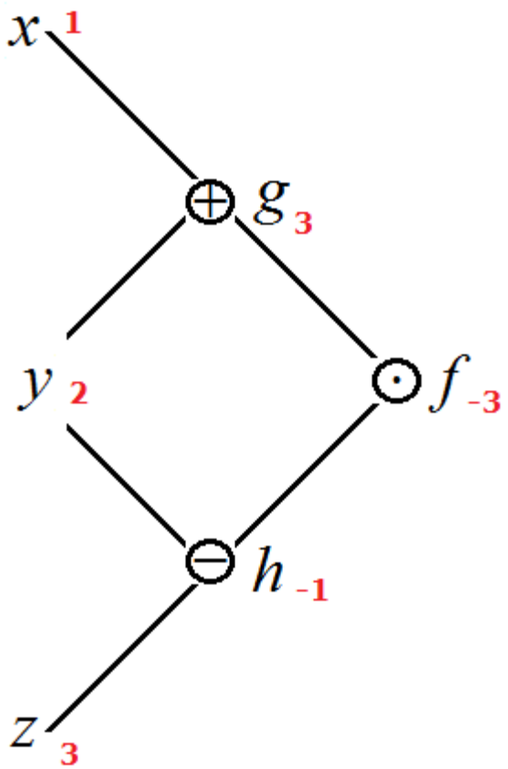
$$f(x,y,z) = \underbrace{(x+y)}_{g(x,y)} \cdot \underbrace{(y-z)}_{h(y,z)}$$

Как проводится вычисление функции?

$$x, y, z = 1, 2, 3$$

Вычисление градиента

$$f = (x+y)\cdot(y-z)$$



$$f(x,y,z) = \underbrace{(x+y)}_{g(x,y)} \cdot \underbrace{(y-z)}_{h(y,z)}$$

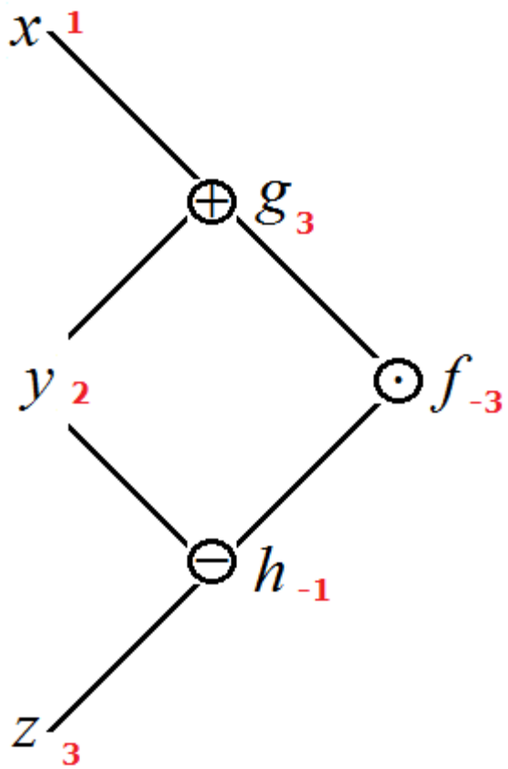
Как проводится вычисление функции?

$$x, y, z = 1, 2, 3$$

«Прямой ход»

Вычисление градиента

$$f = (x+y)\cdot(y-z)$$



$$f(x,y,z) = \underbrace{(x+y)}_{g(x,y)} \cdot \underbrace{(y-z)}_{h(y,z)}$$

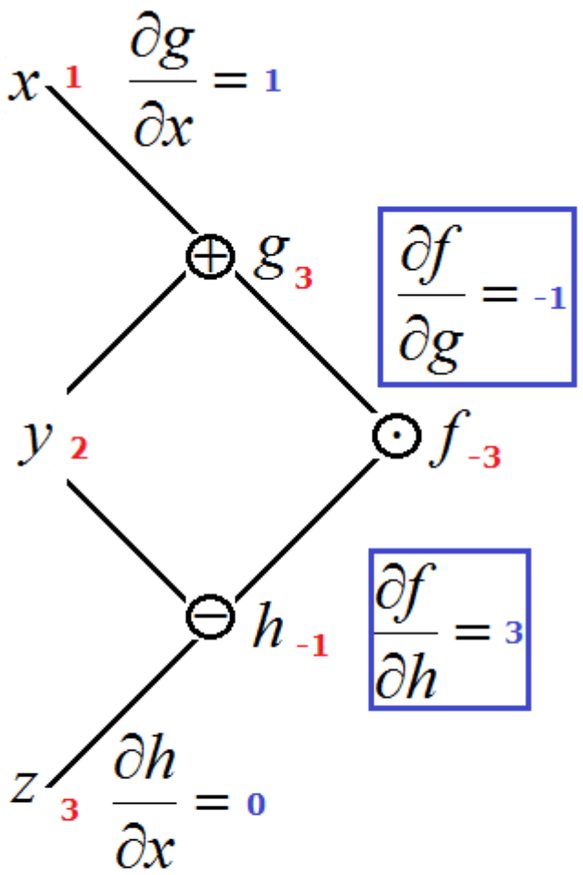
Как проводится вычисление производных?

$$\frac{\partial f}{\partial g} = h, \frac{\partial f}{\partial h} = g$$

$$\frac{\partial g}{\partial x} = 1, \frac{\partial h}{\partial x} = 0$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} + \frac{\partial f}{\partial h}\frac{\partial h}{\partial x}$$

Вычисление градиента



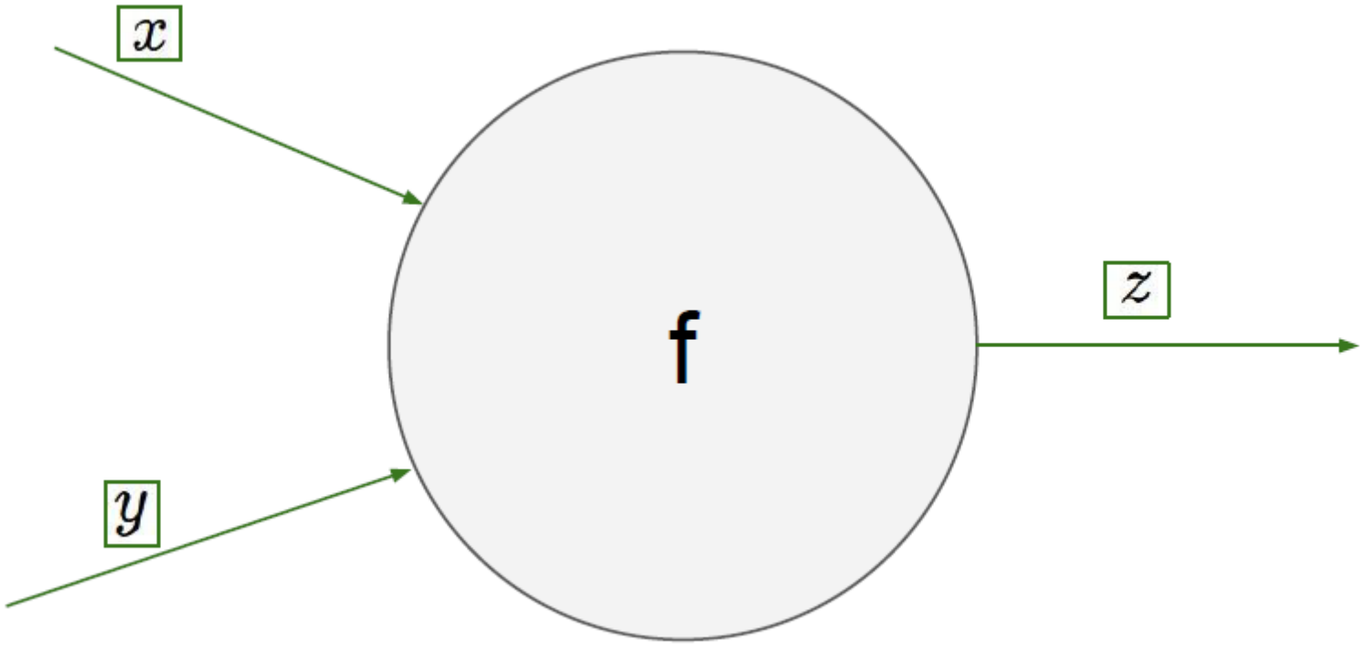
$$\frac{\partial f}{\partial x} = \boxed{\frac{\partial f}{\partial g}}\frac{\partial g}{\partial x} + \boxed{\frac{\partial f}{\partial h}}\frac{\partial h}{\partial x}$$

$$f(x,y,z) = \underbrace{(x+y)}_{g(x,y)} \cdot \underbrace{(y-z)}_{h(y,z)}$$

Как проводится вычисление производных?

«Обратный ход»

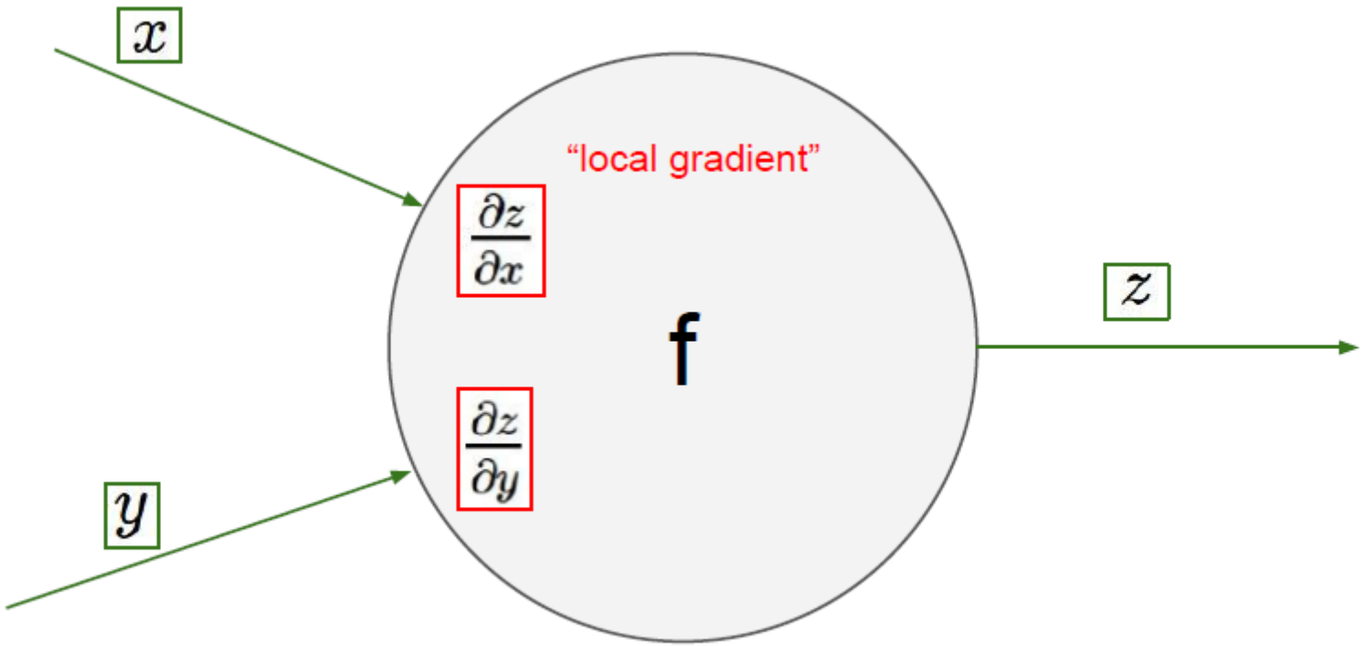
Обратное распространение градиента



http://cs231n.stanford.edu/2017/index.html

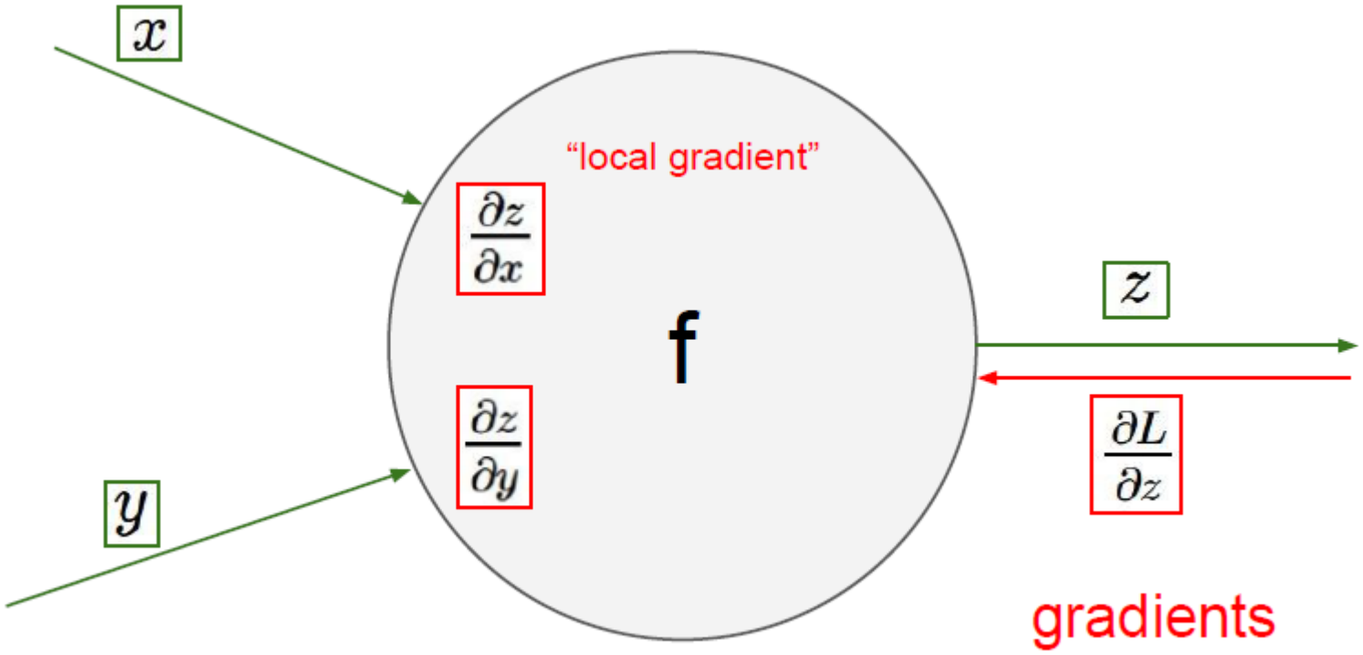
Обратное распространение = SGD + дифференцирование сложных функций

Обратное распространение градиента



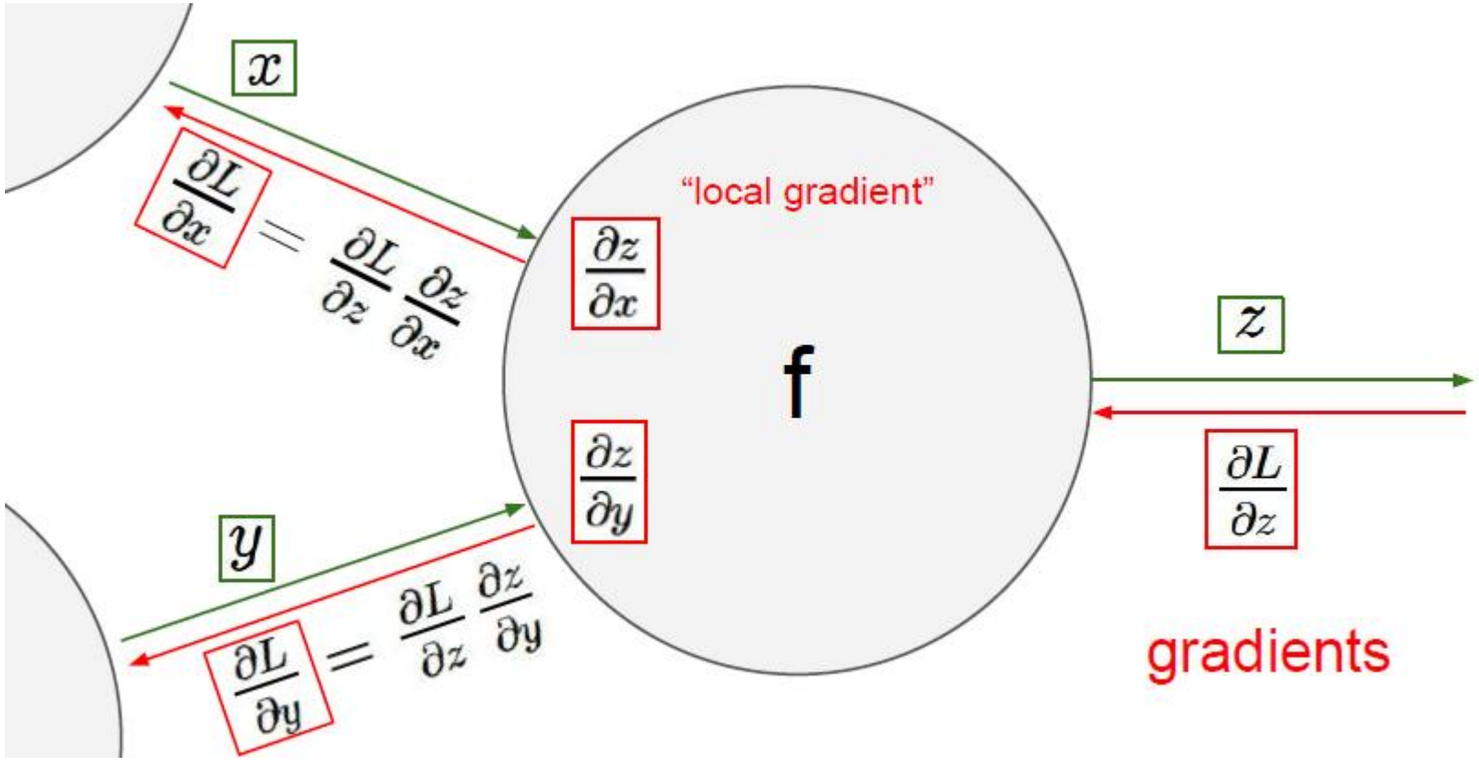
http://cs231n.stanford.edu/2017/index.html

Обратное распространение градиента



http://cs231n.stanford.edu/2017/index.html

Обратное распространение градиента



http://cs231n.stanford.edu/2017/index.html

Итог

НС – нелинейное обобщение линейных алгоритмов

- последовательное преобразование признакового пространства
- ансамбль алгоритмов
- суперпозиция «логистических регрессий»

высокая функциональная выразимость

обучение градиентными методами

Дальше: как делать эффективное обучение?