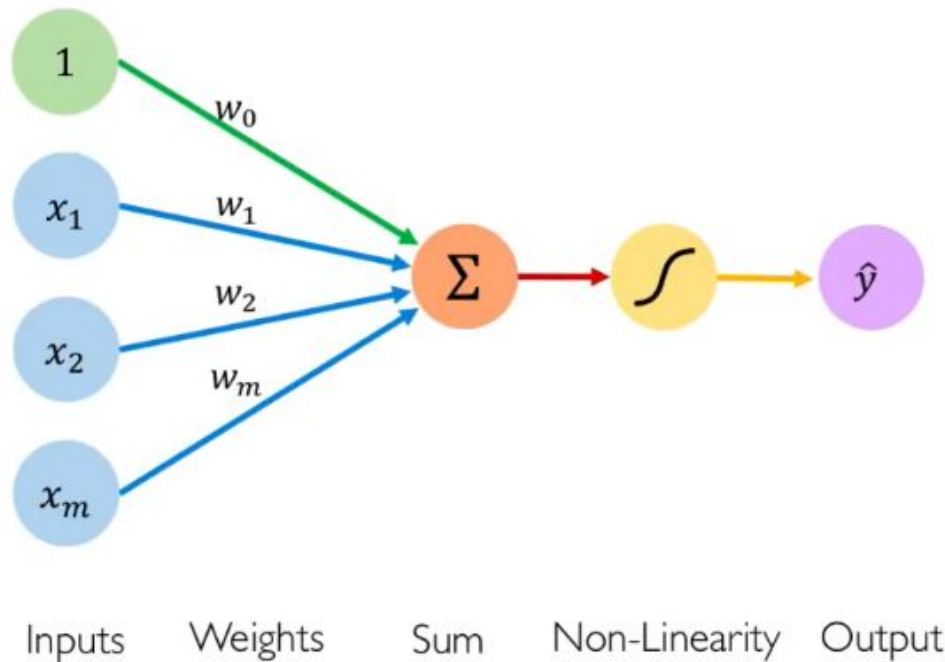


# Алгоритм обратного распространения ошибки для обучения нейронной сети

Подготовил: Тюрин Александр, гр. 22202

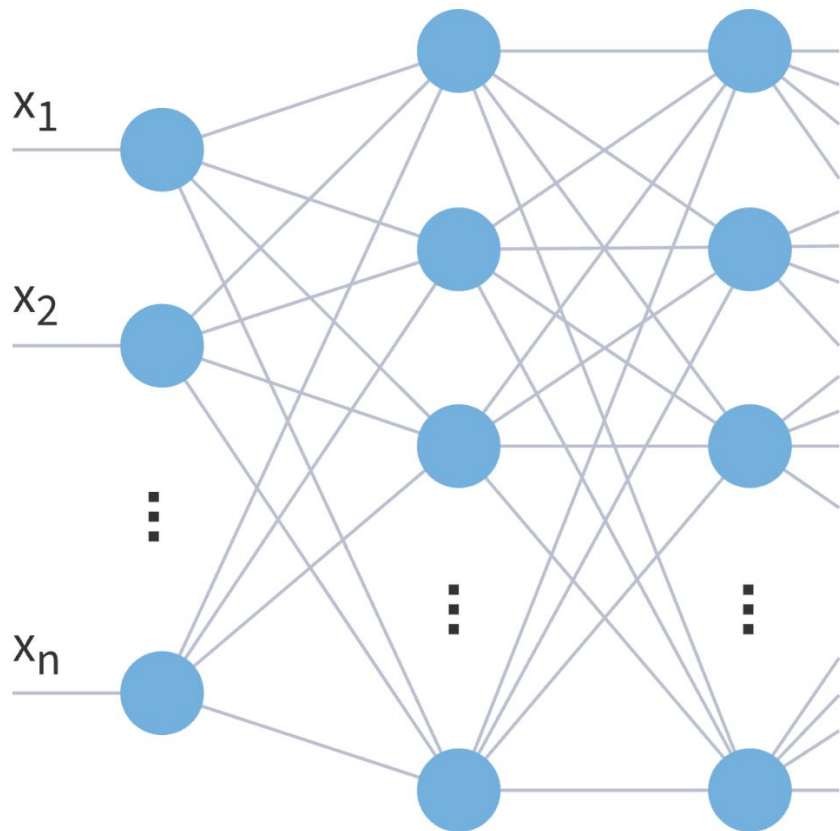


# Искусственные нейронные сети (ANN)



$$\hat{y} = g(w_0 + \mathbf{X}^T \mathbf{W})$$

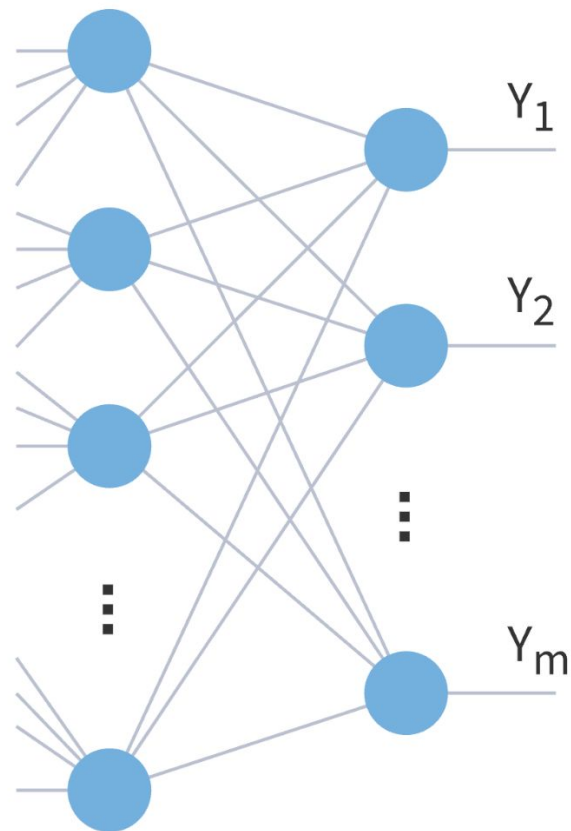
Входной слой

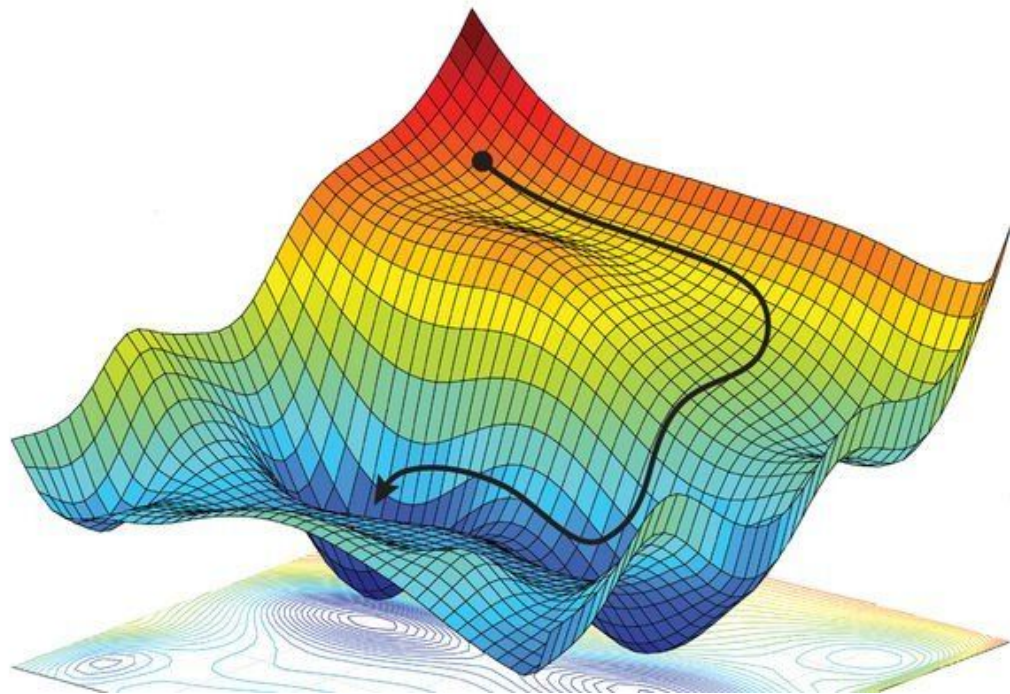


Скрытые слои

⋮  
⋮  
⋮

Выходной слой





$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w}$$

# Проблема до backpropagation: как обучать многослойные сети?

Основные трудности:

- Непонятно, как распределять ошибку между нейронами скрытых слоёв.
- Методы вроде случайного поиска или генетических алгоритмов работали слишком медленно.
- Аналитическое вычисление градиентов для всех весов вручную было крайне трудоёмким.

# Какую проблему решает backpropagation?

1. Эффективное вычисление градиентов
  - Раньше градиенты для каждого веса считали численно (методом конечных разностей), что было очень медленно.
  - Backpropagation позволяет вычислить все градиенты за один проход через сеть.
2. Обучение глубоких сетей
  - До backpropagation многослойные сети почти не обучались из-за сложности обновления весов.
  - Алгоритм автоматически распределяет ошибку по всем слоям, корректируя веса пропорционально их вкладу в ошибку.



Пол Дж. Вербос



Галушкин А. И.



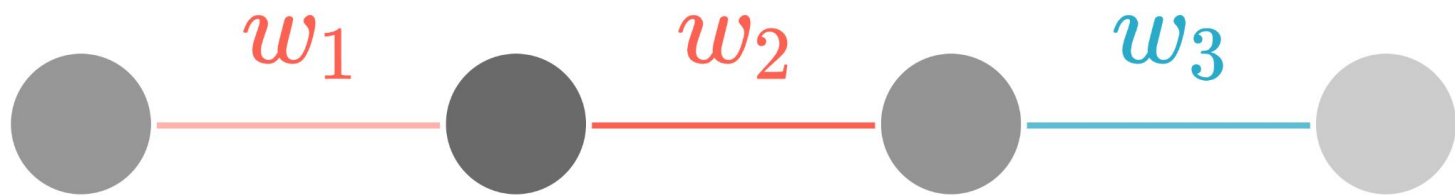
Дэвид Румельхарт

# Алгоритм обратного распространения ошибки

Допустим, у нас есть текущие значения весов  $W$ , и мы хотим совершить шаг SGD по мини-батчу  $X$ . Мы должны сделать следующее:

1. Совершить forward propagation, вычислив и запомнив все промежуточные представления
2. Вычислить все градиенты с помощью backward propagation
3. С помощью полученных градиентов совершить шаг SGD.





$$a^{(L)} = \sigma \left( w^{(L)} a^{(L-1)} + b^{(L)} \right)$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma \left( z^{(L)} \right)$$

$$C_0 = \left( a^{(L)} - y \right)^2$$

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)} \quad \longrightarrow \quad \frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$

$$a^{(L)} = \sigma \left( z^{(L)} \right) \quad \longrightarrow \quad \frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma' \left( z^{(L)} \right)$$

$$C_0 = \left( a^{(L)} - y \right)^2 \quad \longrightarrow \quad \frac{\partial C_0}{\partial a^{(L)}} = 2 \left( a^{(L)} - y \right)$$

$$\frac{\partial C_0}{\partial w^{(L)}} = a^{(L-1)} \sigma' \left( z^{(L)} \right) 2 \left( a^{(L)} - y \right)$$

$$C = \frac{1}{n} \sum_{k=0}^{n-1} C_k$$

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \vdots \\ \frac{\partial C}{\partial w^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{bmatrix}$$

$$\frac{\partial C}{\partial w^{(L)}} = \frac{1}{n} \sum_{k=0}^{n-1} \frac{\partial C_k}{\partial w^{(L)}}$$

$$\frac{\partial C_0}{\partial b^{(L)}} = \frac{\partial z^{(L)}}{\partial b^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

$$\frac{\partial C_0}{\partial w^{(L-1)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

$$\frac{\partial C_0}{\partial w_{jk}^{(L)}} = \frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}}$$

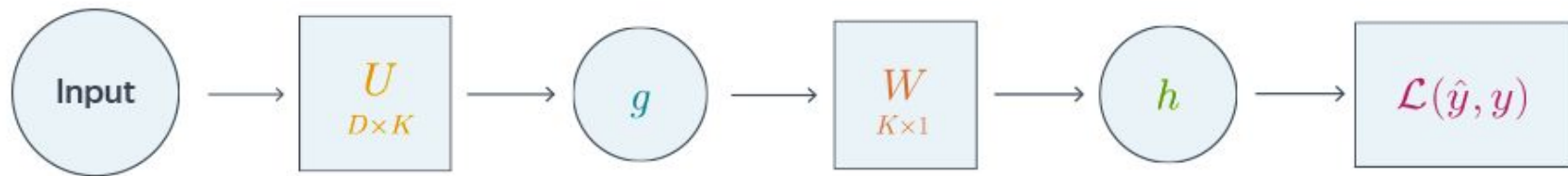
$$\frac{\partial C_0}{\partial a_k^{(L-1)}} = \underbrace{\sum_{j=0}^{n_L-1} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}}}_{\text{Sum over layer L}}$$

$$\delta^{(L)} = \frac{\partial \mathcal{L}}{\partial a^{(L)}} \odot \sigma'(z^{(L)})$$

$$\delta^{(l)} = \left( W^{(l+1)\top} \delta^{(l+1)} \right) \odot \sigma'(z^{(l)})$$

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)} a^{(l-1)\top}$$


$$\frac{\partial \mathcal{L}}{\partial b^{(l)}} = \delta^{(l)}$$



# Проблемы backpropagation

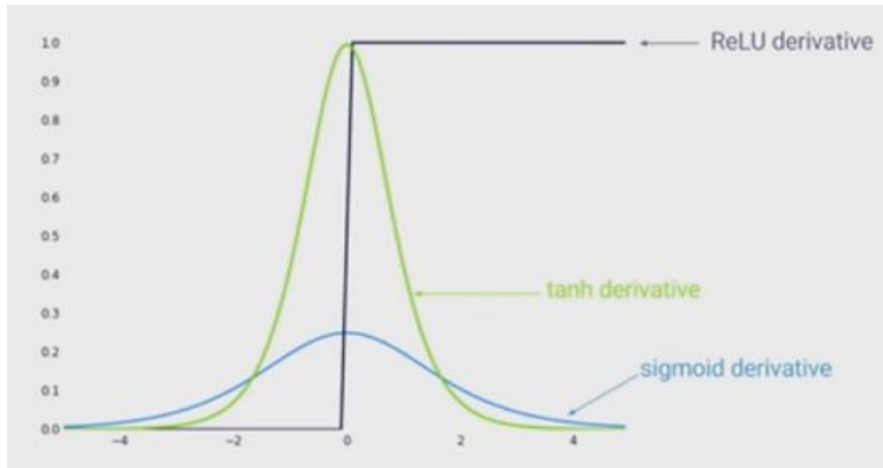
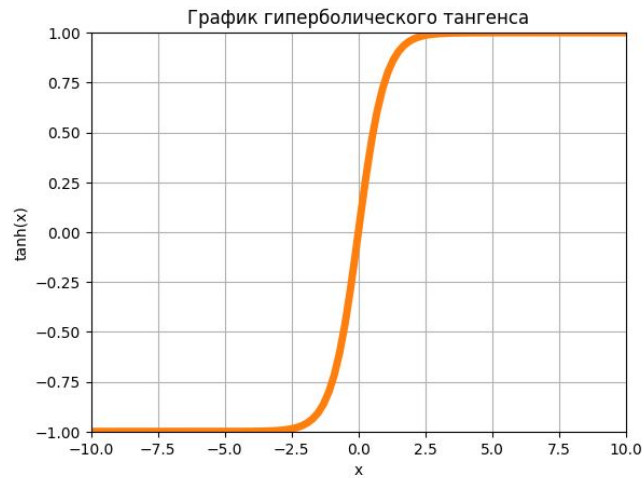
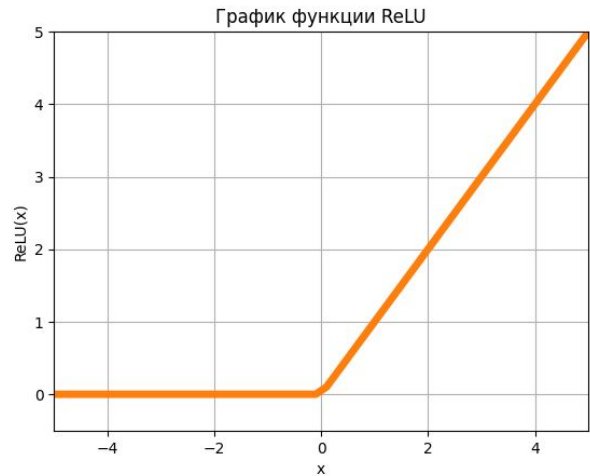
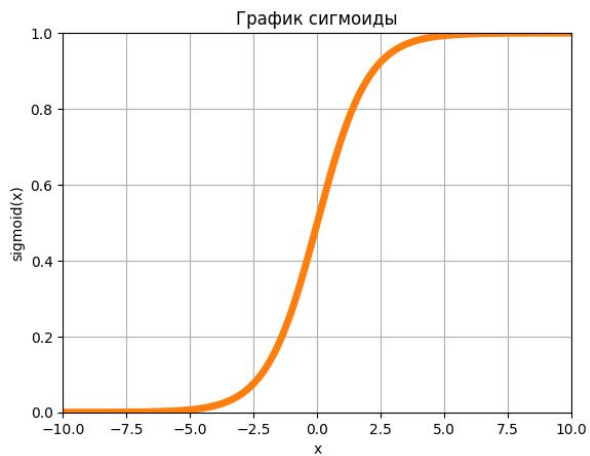
- Затухающие градиенты (Vanishing Gradients).
- Взрывающиеся градиенты (Exploding Gradients).

## Методы решения

- BatchNorm
  - Функции активации (ReLU)
  - Оптимизаторы
- 



# Функции активации



# Применение и значение

- Обучение глубоких нейронных сетей
- Применение в компьютерном зрении
- Обработка естественного языка (NLP)

**Спасибо за внимание!**