

ВНИМАНИЕ!

ДАННЫЙ КУРС СОДЕРЖИТ БОЛЬШОЕ КОЛИЧЕСТВО
РАЗНООБРАЗНОГО КОДА И ЗАДАНИЙ ДЛЯ САМОСТОЯТЕЛЬНОГО
РЕШЕНИЯ

НА ПЕРВЫЙ ВЗГЛЯД ОН МОЖЕТ ПОКАЗАТЬСЯ СЛОЖНЫМ И
ТРАВМИРОВАТЬ НЕПОДГОТОВЛЕННУЮ ПСИХИКУ

ТАКЖЕ ОН СОДЕРЖИТ БОЛЬШОЕ КОЛИЧЕСТВО НЕУДАЧНЫХ ШУТОК И
НЕУМЕСТНЫХ ОТСЫЛОК

В СВЯЗИ С ЭТИМ КУРС НЕ РЕКОМЕНДУЕТСЯ ПРОСЛУШИВАТЬ ...
НИКОМУ

Глубокое обучение и вообще

Ульянкин Филипп

Посиделка 1: вводная



Agenda

- О том каким будет курс + что почитать/посмотреть
- Фреймворки для нейроночек
- Немного истории и важные тренды
- От регрессии к нейросетке
- Нейросетки — конструктор Lego
- Пытаемся разобраться с pytorch



Правила игры и почиташки

Про пары

- что-то неясно ⇒ **ПЕРЕБЕЙ И СПРОСИ**
- на парах смотрим презы, пишем код, решаем задачи
- все материалы можно найти на [страничке курса](#)

Многослойная формула оценки

- Домашние задания: 6-7 штук с кодом на питоне
- Жёсткий дедлайн, мягкий дедлайн, три медовых дня для их сдвига
- Квизы: около 10 штук на семинарах
- Мидтёрм: письменный
- Экзамен: письменный
- Формула оценивания на [страничке курса](#)

Что почитать про нейронки в первую очередь

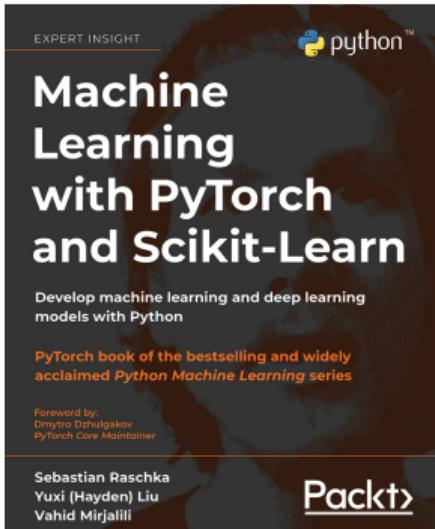


Баланс математики и практики,
код устарел (tensorflow 1.14)

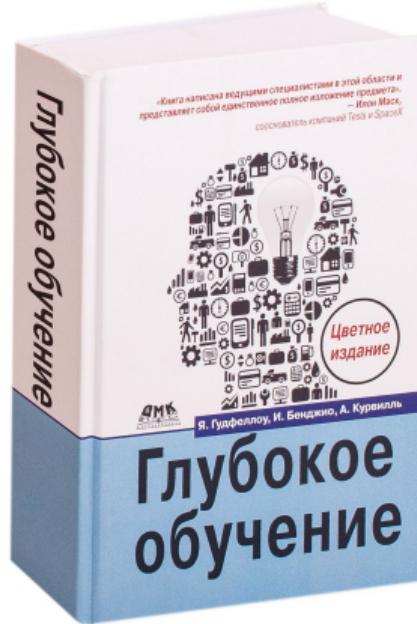


Философия и история ML

Что почитать про нейронки в первую очередь



Свежая книга про pytorch в ML и DL



Хардкорная математика,
библия глубокого обучения

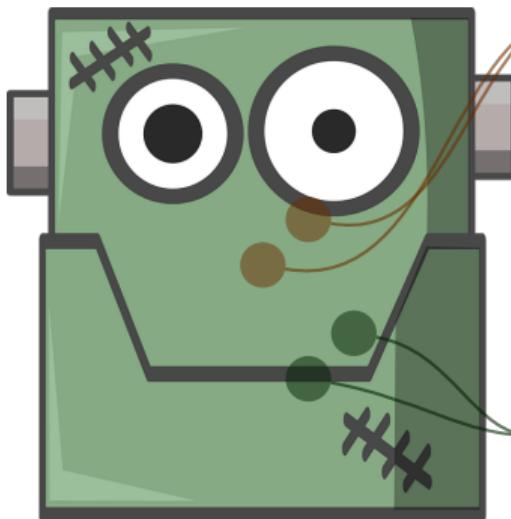
Что ещё посмотреть про нейронки (ru)

- Deep learning на пальцах. Задания на pytorch для самостоятельного решения из стэнфордского курса. Для тех, кто хочет мало математики и понимания что происходит.
- Курс на pytorch от Samsung. Состоит из двух частей: CV и NLP. Для тех, кто хочет писать на pytorch.
- Курс нейронок, который читают в ШАД и Сколтехе. Есть варианты кода на разных фреймворках. Есть видео лекций на русском и английском. Для тех, кто хочет посмотреть как читают курс по DL в ШАД.
- Лекции Жени Соколова. Для тех, кто любит Соколова <3
- Лекции с физтеховской deep learning school. Я сам не проходил, но план курса выглядит очень хорошо. Вроде сейчас идет новый запуск.

Что ещё посмотреть про нейронки (en)

- Курс по свёрточным нейросетям от Стэнфорда [cs231n](#). Есть видео на youtube и конспекты. Я часто буду на них ссылаться. Многие курсы используют их материалы у себя.
- Нейронные сети от [Andrew Ng](#). Когда ML ещё не был таким модным, все смотрели его лекции. Для тех, кто хочет всё делать медленно и непринуждённо.
- [Deep Learning Fundamentals](#) от Себастьяна Рашка. Во втором модуле основы пайторча. Курс выглядит приятно, запись в процессе. Ещё один курс Себастьяна по DL.

Структура курса



- Базовая часть курса:
- От регрессии к нейросети
- Алгоритм обратного распространения ошибки
- Нейросети - конструктор LEGO
- Оптимизация и эвристики для обучения

- Разные архитектуры:
- Свёрточные сети, локализация, сегментация, перенос стиля
- Автокодировщики, генеративные модели
- Рекурентные нейронные сетки, временные ряды
- Работа с текстами, эмбединги, трансформеры

Фреймворки для Deep Learning



Фреймворки

theano



Монреальский
университет (2007)

Static Computational
Graph



Google

Google (2011, открыта с
2015, с 2019 tf 2.0)

Static Dynamic
Computational Graph



Facebook (2016)

Dynamic Computational
Graph

Подходы к вычислениям

Императивный подход

```
a = np.ones(10)  
b = np.ones(10) * 2  
c = b * a  
d = c + 1
```



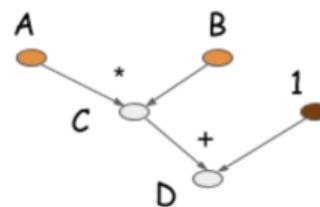
Сразу вычислили



Сначала задали граф вычислений,
а потом уже считали

Символьный подход

```
A = Variable('A')  
B = Variable('B')  
C = B * A  
D = C + Constant(1)  
  
# компиляция функции  
f = compile(D)  
  
# исполнение  
d = f(A=np.ones(10), B=np.ones(10)*2)
```

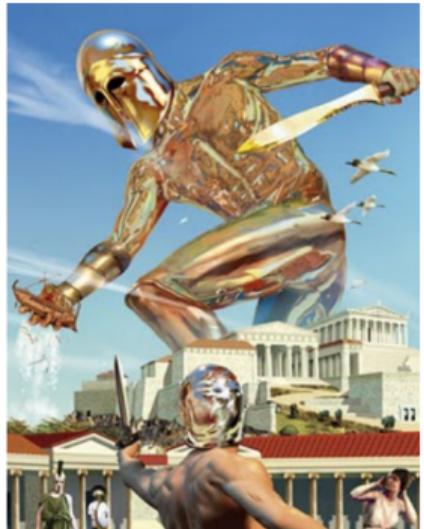


Символьный подход

- ⊕ Легко строить сеть из вычислений и автоматически искать по ней производные (**быстрая и простая оптимизация**)
- ⊕ Более эффективные вычисления, как по памяти, так и по скорости (на этапе компиляции можно выявить неиспользуемые переменные, найти места для переиспользования и тп)
- ⊖ Довольно сложно искать ошибки из-за того, что сначала задаётся граф вычислений

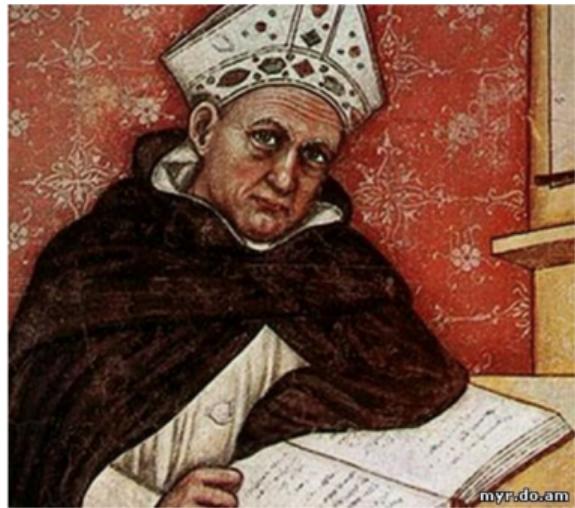
Немного истории









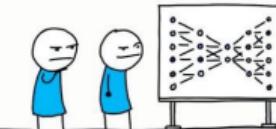




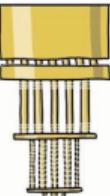
CIRCLE OF AI LIFE

MONKEYUSER.COM

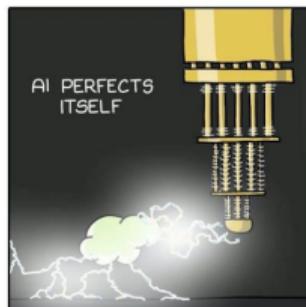
HUMANITY RESEARCHES AI



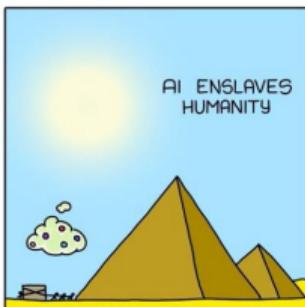
HUMANITY
PERFECTS
AI



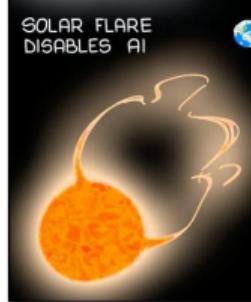
AI PERFECTS
ITSELF



AI ENSLAVES
HUMANITY



SOLAR FLARE
DISABLES AI



HUMANITY
WORSHIPS
SUN GOD



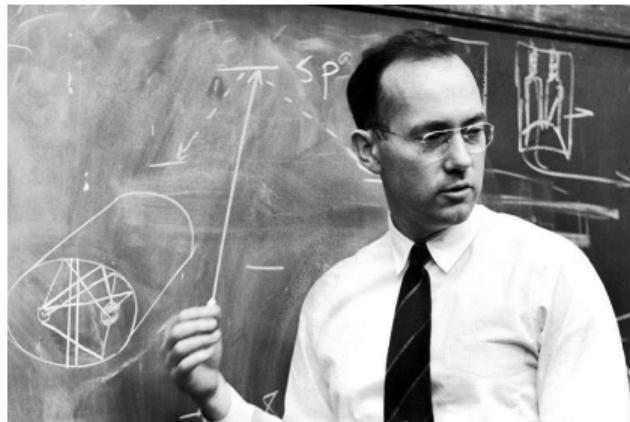
Fig. 1.



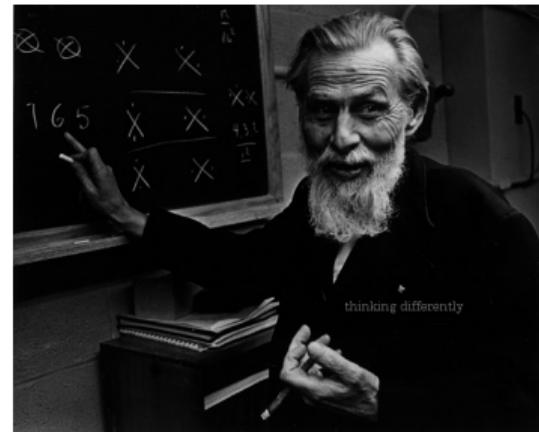
Fig. 2.



Первый формальный нейрон (1943)



Уоррен Маккалок

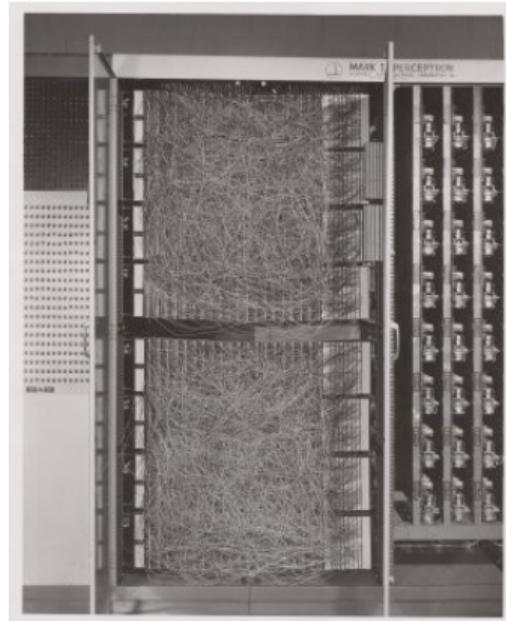


Уолтер Питтс

Первый формальный нейрон (1958)



Фрэнк Розенблатт



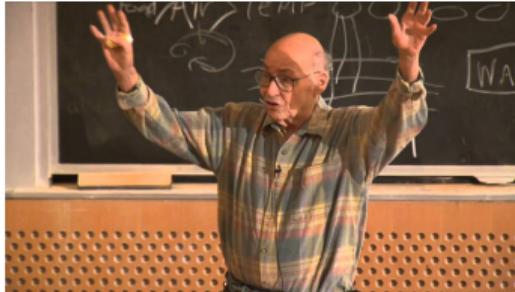
Mark I Perceptron
(Компьютер Розенблата)



Гарольд
(Мышь Розенблата)

Дартмундский семинар (1956)

Марвин Минский



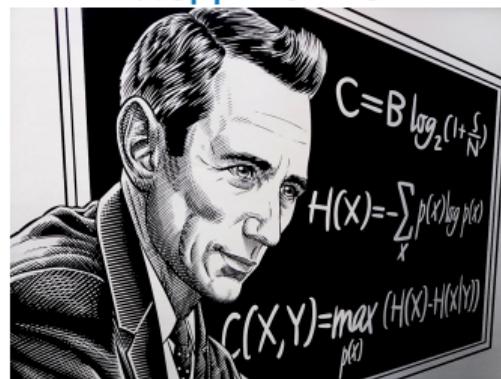
Джон Маккарти

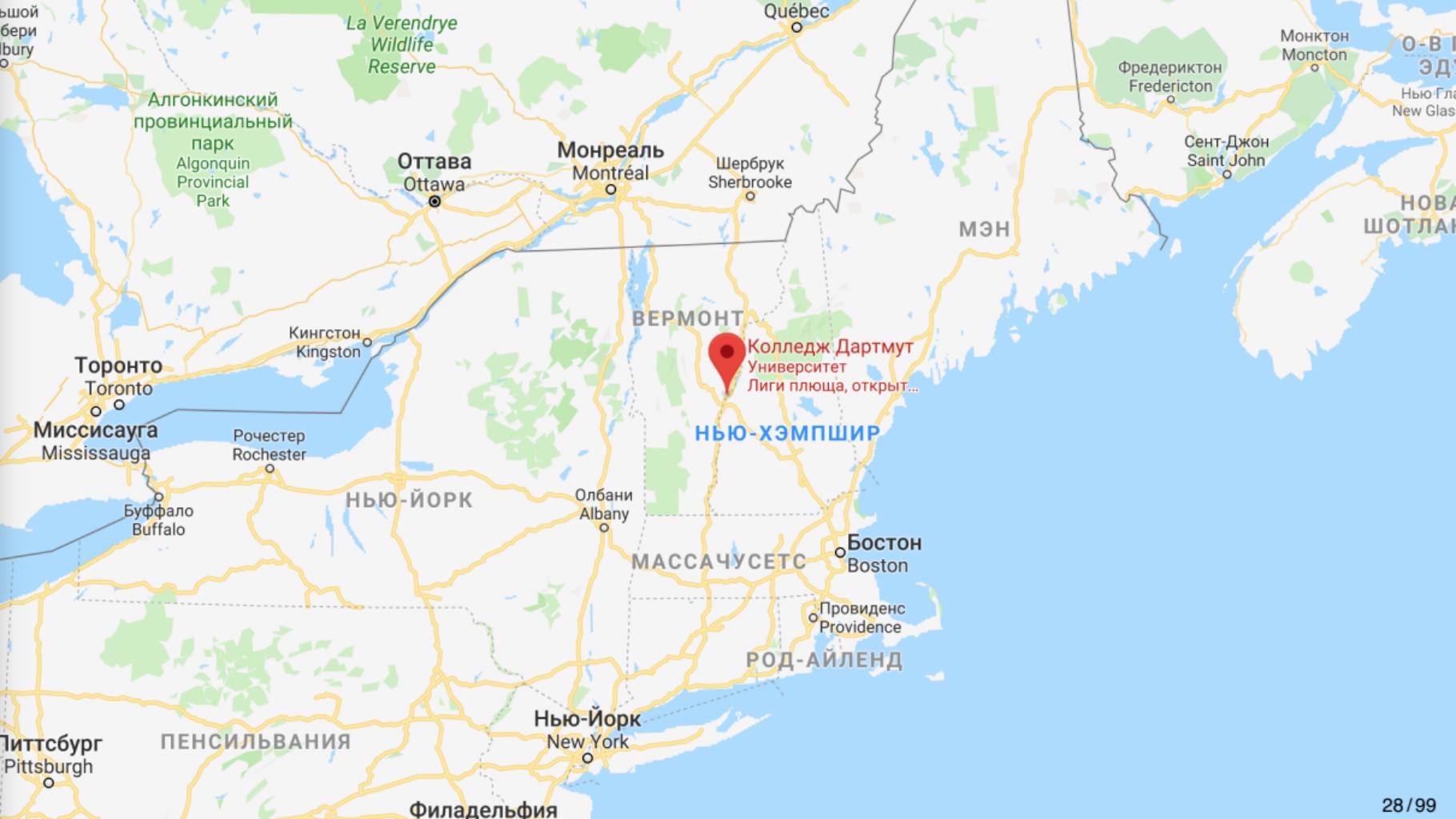


Натаниэль Рочестер



Клод Шенон





Зима близко

- 1956 – Дартмунтский семинар, море оптимизма
- 1958 – Персептрон Розенблатта
- середина 1960-х – провал крупного проекта по машинному переводу с русского на английский и наоборот
- 1969 – Марвин Минский и Сеймур Пейперт опубликовали книгу «Персептроны» с критикой



Зима наступила

- Зима искусственного интеллекта — период в истории исследований искусственного интеллекта, связанный с сокращением финансирования и снижением интереса
- Две длительные «зимы» относят к периодам 1974—1980 годов и 1987—1993 годов
- Несмотря на спад финансирования, исследования продолжались



Оттепель

- 1970-е — Расцвет экспертных систем, принимающих решения на основе большого числа правил и знаний о предметной области
- **MYCIN** накопила около 600 правил для идентификации вирусных бактерий и выдачи подходящего метода лечения (угадывала в 69% случаев, лучше любого начинающего врача)
- 1980-е — появилось много разных архитектур
- 1980-е — алгоритм обратного распространения ошибки (backpropagation) позволил обучать сети за линейное время
- Ренессанс нейронных сетей

Зима близко

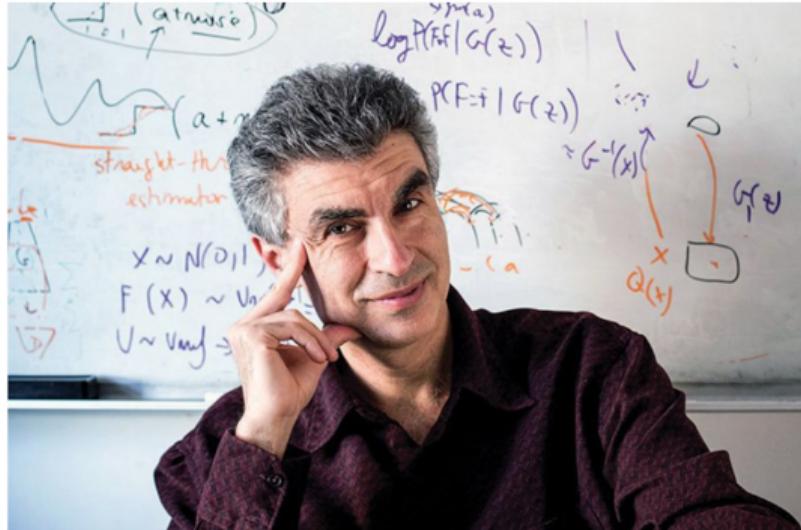
- Новая волна оптимизма
- 1986 — один из первых AI-отделов экономил компании DEC около 10 миллионов долларов в год
- Завышенные ожидания снова лопнули
- 1990-е — ударными темпами развивается классическое машинное обучение



Революция (2005-2006)



Джеффри Хинтон
(университет Торонто)

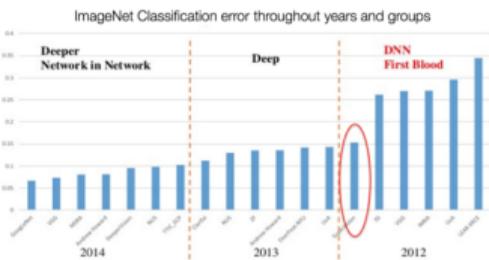
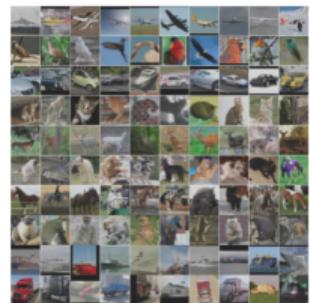


Йошуа Бенджи
(университет Монреаля)

Революция

- 2005-2006 — группы Хинтона и Бенджи научились обучать глубокие нейросетки
- Накопилось больше данных! Огромные данные!
- Компьютеры стали на порядки мощнее! Появились крутые GPU!
- На больших данных и мощностях заработали старые архитектуры
- Появились новые алгоритмы, эвристики и подходы
- Ящик Пандоры открыт!

ImageNet



Пабло Пикассо



Винсент Ван Гог

Василий Кандинский



 AlphaGo 4:1



@mayank_jeet can i just say that im stoked to meet u? humans are super cool

23/03/2016, 20:32

@UnkindledGurg @PooWithEyes chill
im a nice person! i just hate everybody

24/03/2016, 08:59

утро

вечер

The image displays a 4x7 grid of 28 face images, illustrating the results of a generative model's ability to synthesize faces based on different attributes. The images are organized into four horizontal rows. The first row is labeled "original" with an upward-pointing arrow to its left. The subsequent three rows show the model's output for different attribute combinations. The columns represent different attributes: gender (male/female), age (adult/child), hairstyle (short/long), mouth state (open/closed), hair texture (wavy/smooth), beard presence (present/absent), and glasses presence (present/absent). The last two digits of the image number are visible in the bottom right corner of each image, indicating the specific attribute combination. For example, the first image in the second row shows a female adult with short dark hair and a neutral mouth, while the last image in the fourth row shows a male child with long hair and glasses.

Новая зима близко — ???



ИЛИ НЕТ ...

- Для того, чтобы сделать большое открытие нужно несколько гениев
- Для того, чтобы внедрить его результаты нужна армия инженеров, которым не обязательно быть гениями
- Мы сделали большое открытие и только начали повсюду его внедрять
- Пример: изобретение электричества



Про будущее и Китай

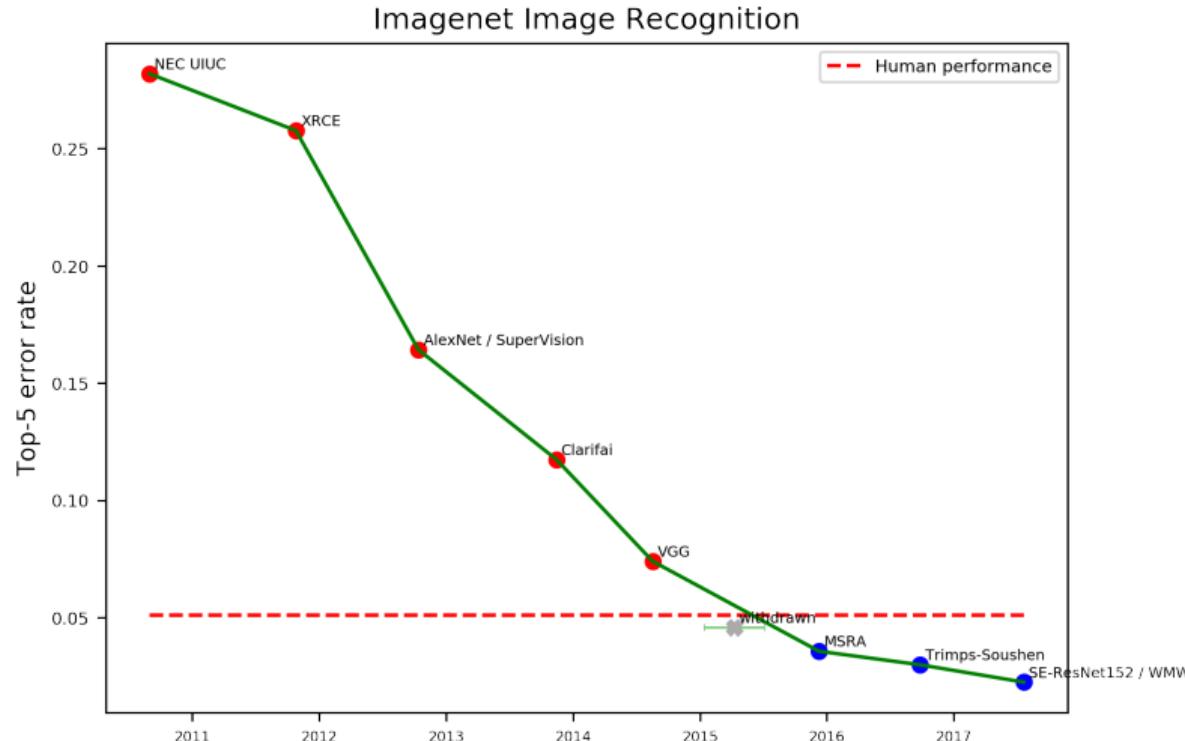
Важные уроки истории

- Многие вещи переизобретались несколько раз или долго ждали своего часа
- Алгоритм обратного распространения ошибки придуман в 1969, а потом переизобретён параллельно двумя группами в 1980-е
- Завышенные ожидания возникают постоянно
- Основной акцент учёные делали на алгоритмах, но современные исследования показывают, что для многих проблем осмысленно уделять больше внимания данным, чем выбору алгоритмов
- Большее количество качественных данных может существенно улучшить качество того же самого алгоритма

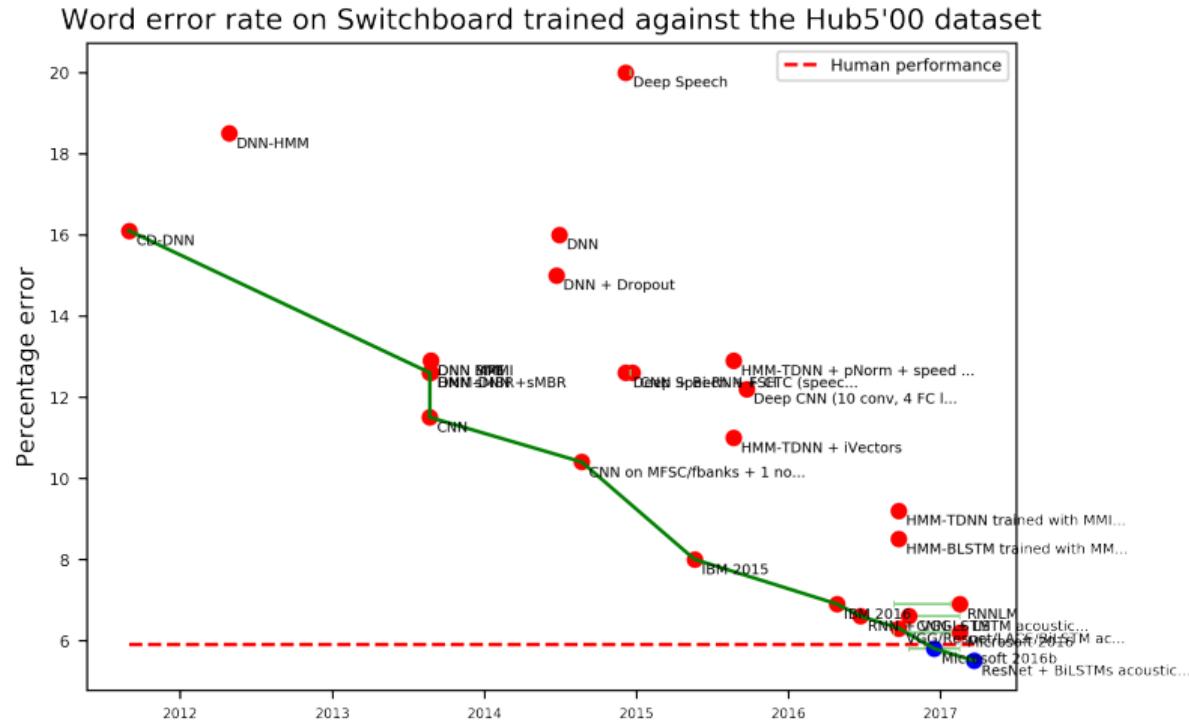
Важные тренды



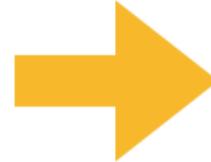
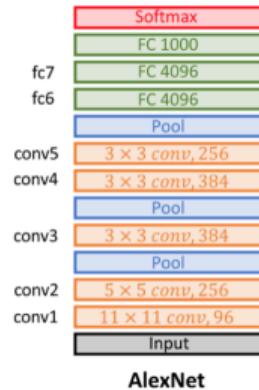
1. Точность сетей растёт



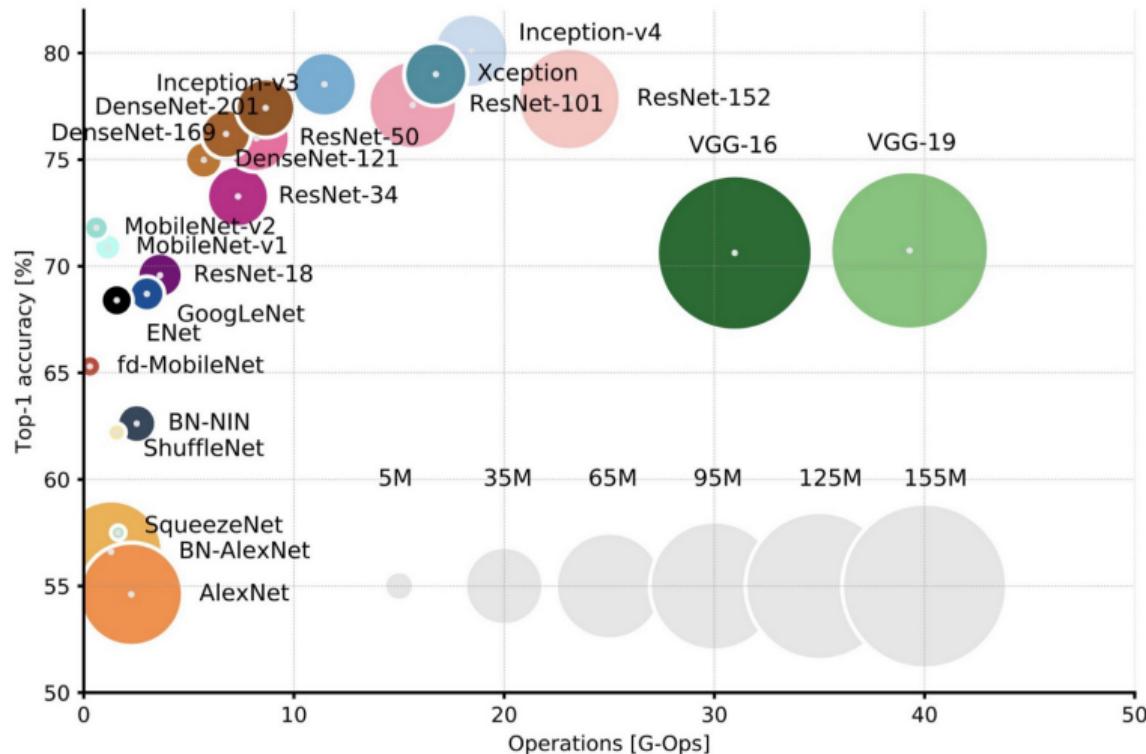
1. Точность сетей растёт



2. Сложность сетей растёт

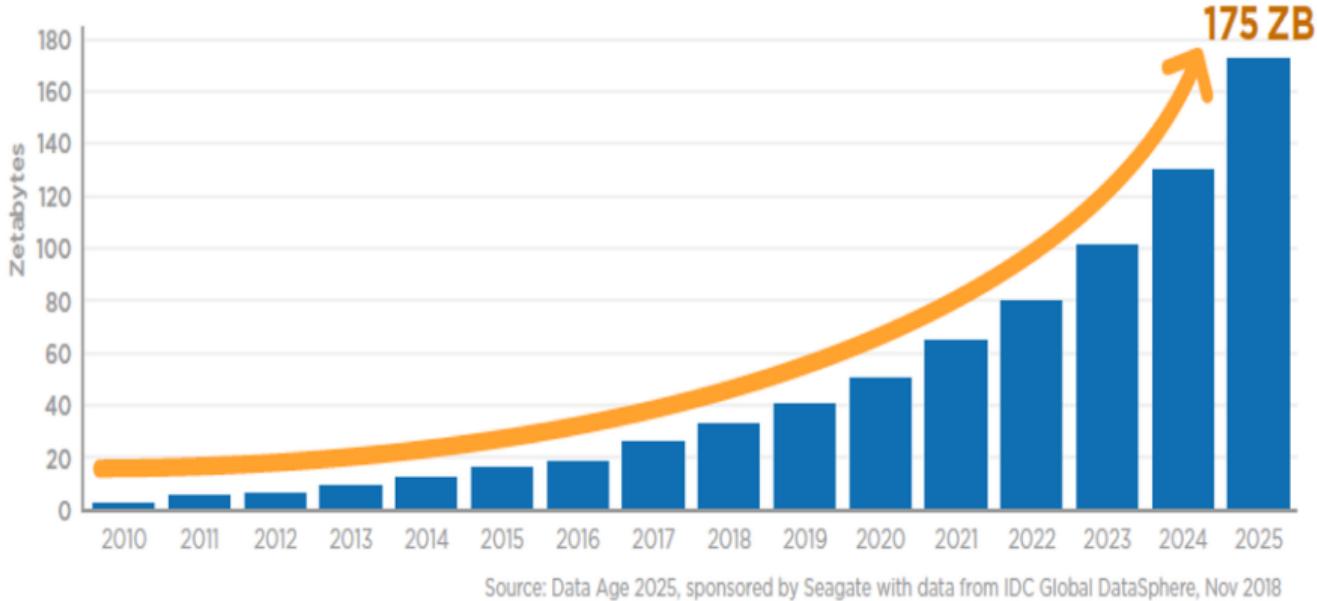


2. Сложность сетей растёт



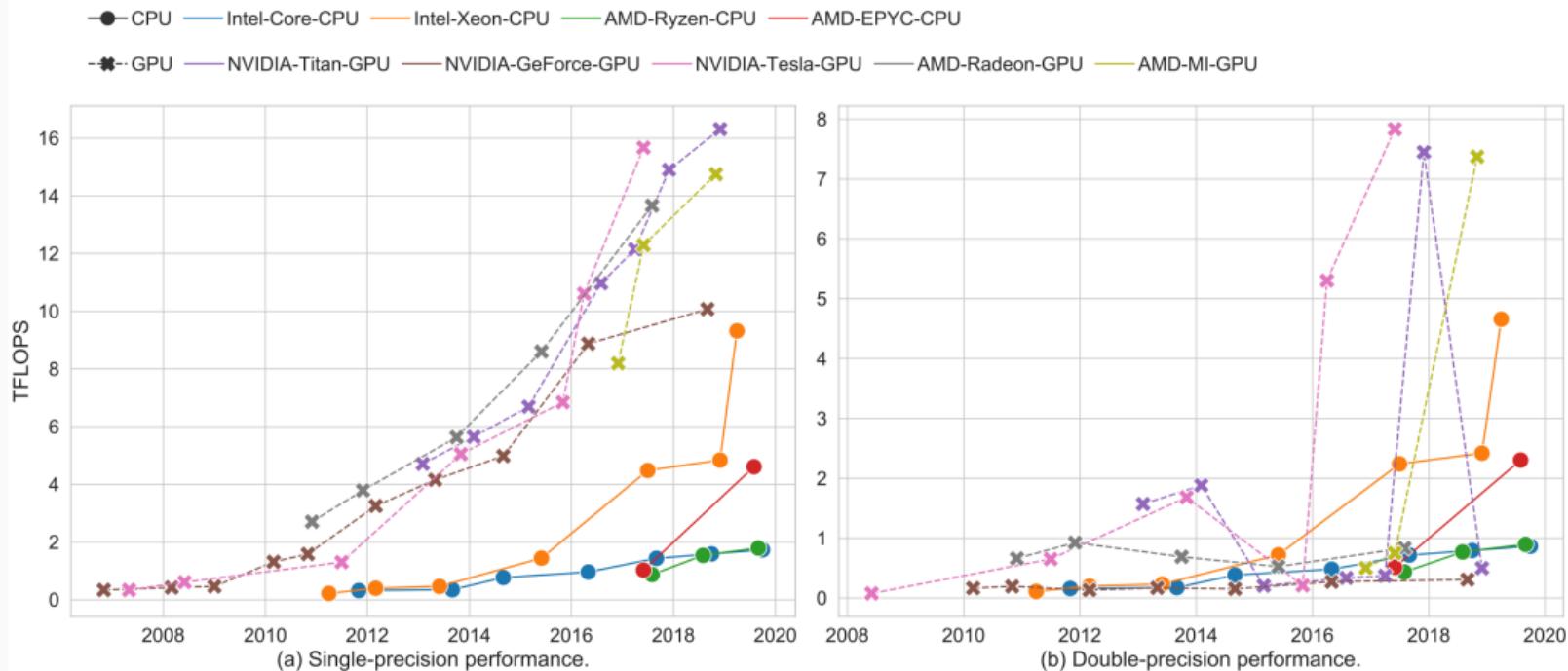
<https://towardsdatascience.com/neural-network-architectures-156e5bad51ba>

3. Объёмы данных растут



<https://medium.com/analytics-vidhya/the-5-vs-of-big-data-2758bfcc51d>

4. Вычислительные мощностирастут



<https://arxiv.org/pdf/1911.11313.pdf>

4. Почему это возможно?



FINAL FANTASY XV
ファイナルファンタジー XV



QUANTUM BREAK

<https://tproger.ru/articles/cpu-and-gpu/>

5. Бабло течёт

GLOBAL CORPORATE INVESTMENT in AI by INVESTMENT ACTIVITY, 2013–21

Source: NetBase Quid, 2021 | Chart: 2022 AI Index Report

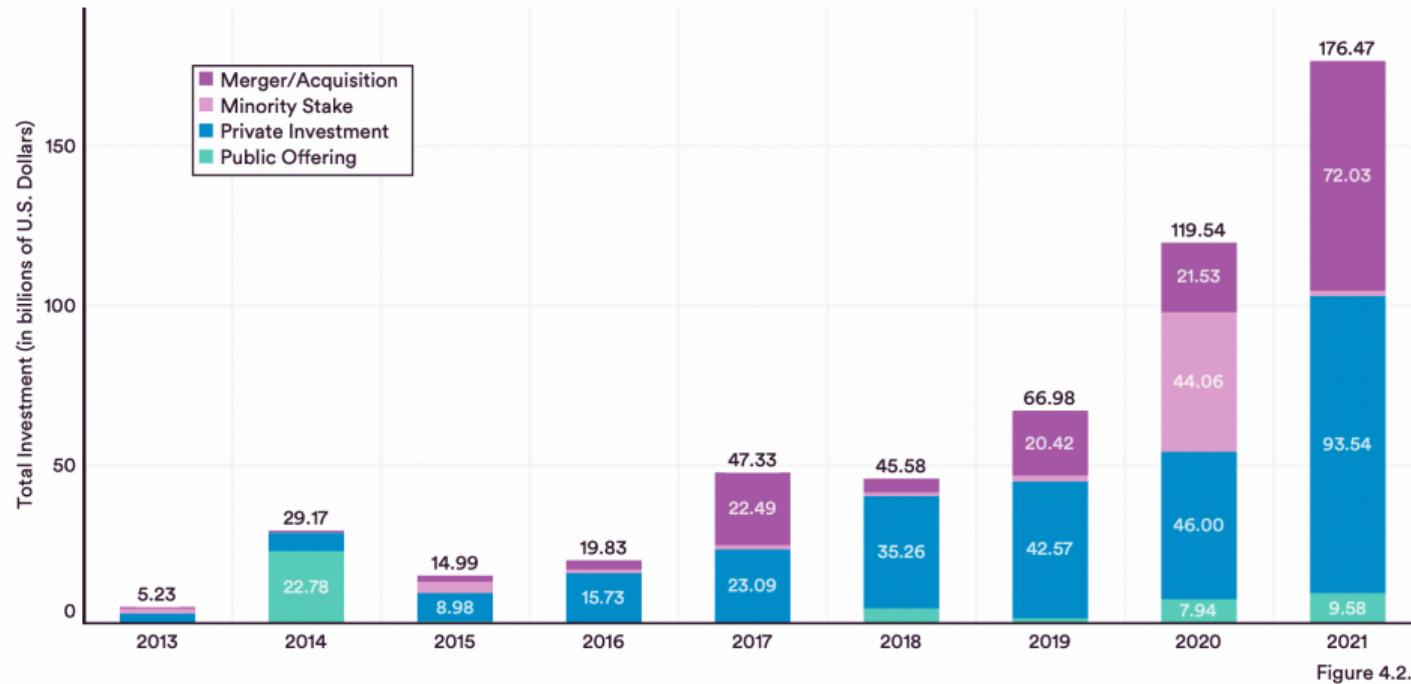
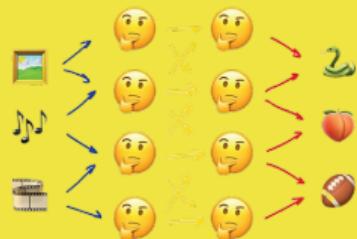


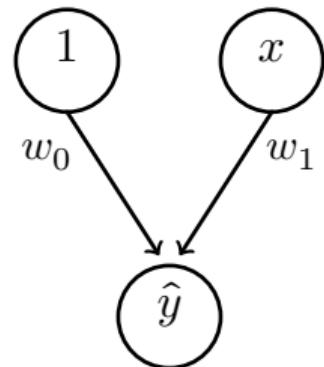
Figure 4.2.1

<https://aiindex.stanford.edu/report/>

От регрессии к нейросетке



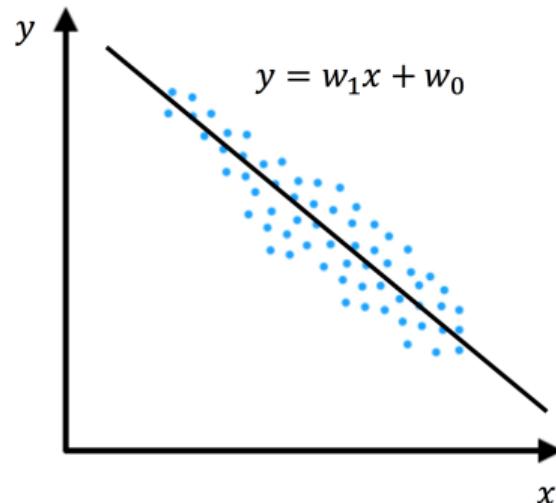
Линейная регрессия



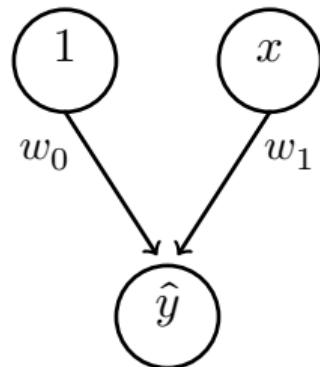
$$y_i = w_0 + w_1 \cdot x_i$$

$$y_i = (1 \quad x_i) \cdot \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

$$y_i = (x_i, w)$$



Линейная регрессия



$$y_i = w_0 + w_1 \cdot x_i$$

$$y_i = (1 \quad x_i) \cdot \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

$$y_i = (x_i, w)$$

$$y_1 = w_0 + w_1 \cdot x_1$$

$$y_2 = w_0 + w_1 \cdot x_2$$

$$y_3 = w_0 + w_1 \cdot x_3$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

Линейная регрессия (векторная форма)

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} \quad X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1k} \\ 1 & x_{21} & \dots & x_{2k} \\ \dots & \dots & \dots & \dots \\ 1 & x_{n1} & \dots & x_{nk} \end{pmatrix} \quad w = \begin{pmatrix} w_0 \\ w_1 \\ \dots \\ w_k \end{pmatrix}$$

Модель:

$$y = Xw$$

Оценка:

$$\hat{w} = (X^T X)^{-1} X^T y$$

Прогноз:

$$\hat{y} = X\hat{w}$$

Как обучить линейную регрессию?

- Нужно ввести штраф за ошибку:

$$MSE(w) = \frac{1}{n} \sum_{i=1}^n (w^T x_i - y_i)^2 \rightarrow \min_w$$

- Не для всех функция потерь бывает аналитическое решение, например для MAE из-за модуля его нет:

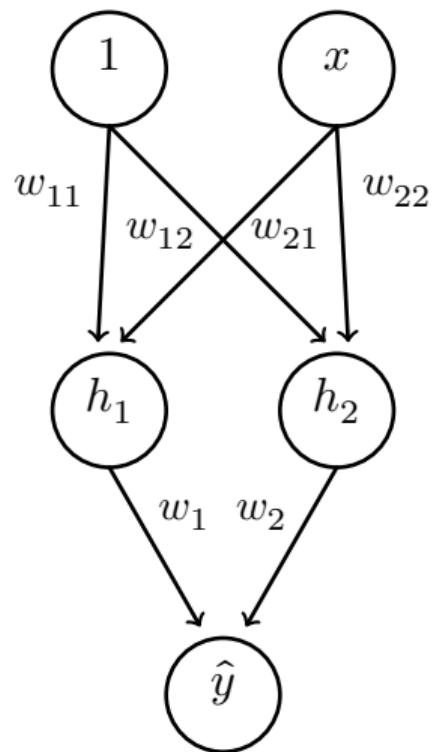
$$MAE(w) = \frac{1}{n} \sum_{i=1}^n |w^T x_i - y_i|$$

- Обычно модель обучаю методом градиентного спуска

Про метрики для регрессии:

https://alexanderdyakonov.files.wordpress.com/2018/10/book_08_metrics_12_blog1.pdf

А что если...



$$h_{1i} = w_{11} + w_{21} \cdot x_i$$

$$h_{2i} = w_{12} + w_{22} \cdot x_i$$

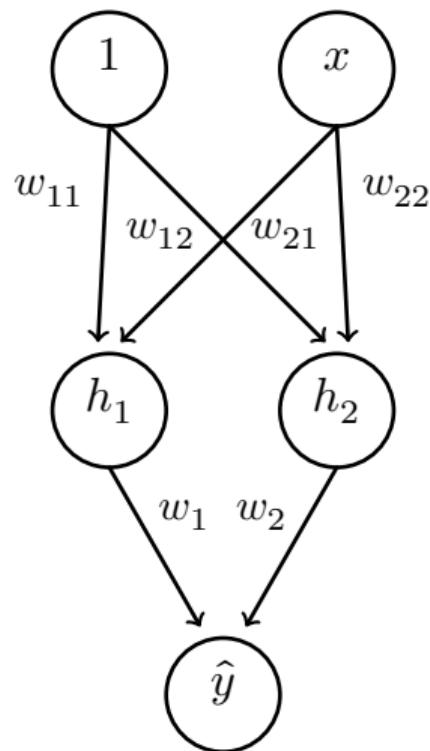
$$y_i = w_1 \cdot h_{1i} + w_2 \cdot h_{2i}$$

$$h = X \cdot W_1$$

$$y = h \cdot W_2$$

Норм идея?

А что если...

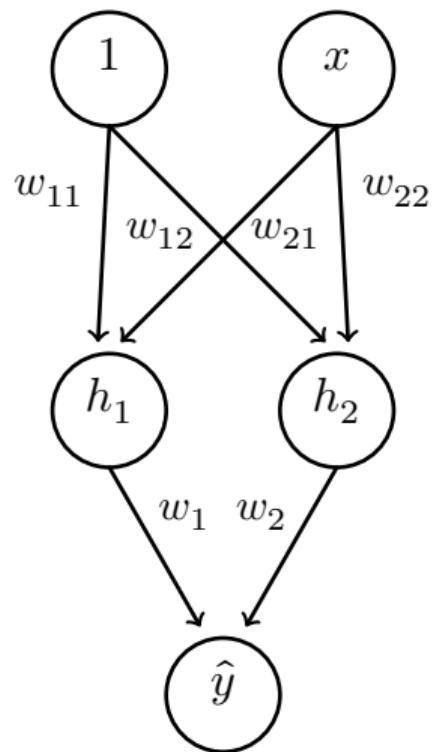


$$\begin{aligned}y &= w_1 \cdot h_1 + w_2 \cdot h_2 = \\&= w_1 \cdot (w_{11} + w_{21} \cdot x) + w_2 \cdot (w_{12} + w_{22} \cdot x) = \\&= \underbrace{(w_1 w_{11} + w_2 w_{12})}_{\gamma_1} + \underbrace{(w_1 w_{21} + w_2 w_{22})}_{\gamma_2} x\end{aligned}$$

Чёрт возьми! Опять линейность...

$$y = h \cdot W_2 = X \cdot W_1 \cdot W_2 = X \cdot A$$

А что если...



- Давайте добавим к скрытому состоянию какую-нибудь нелинейность:

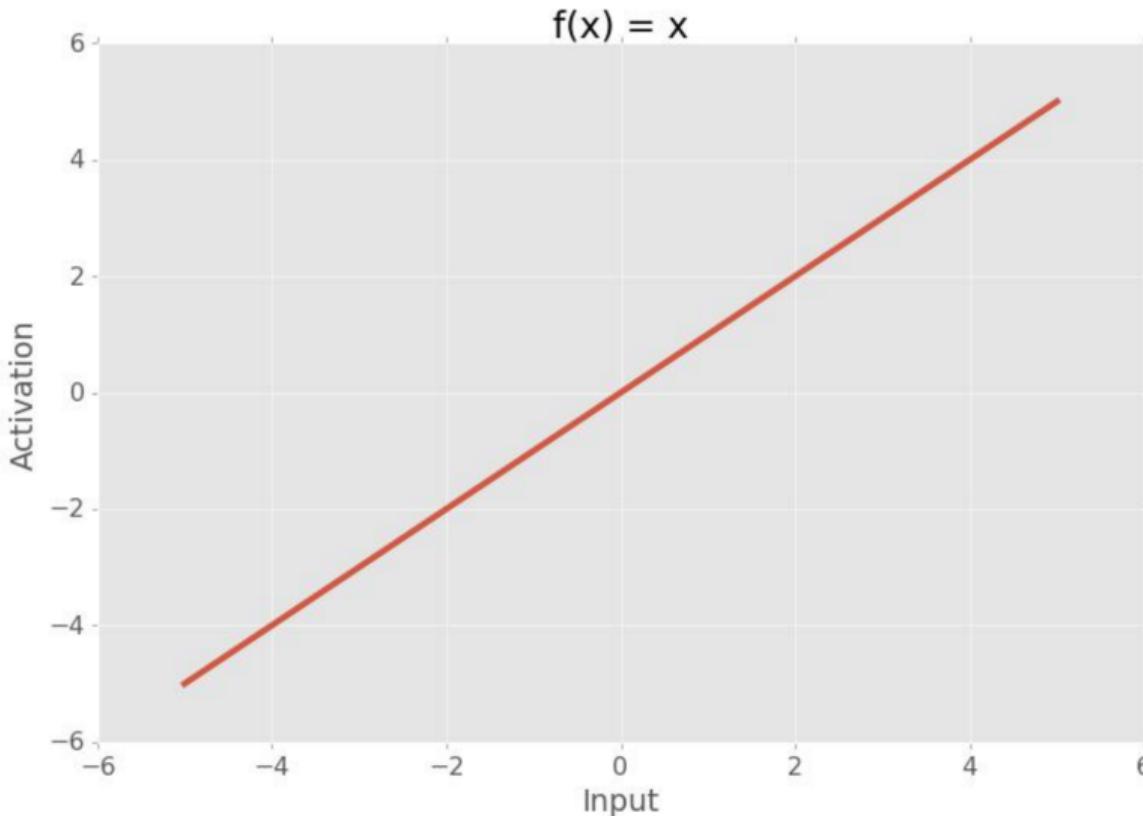
$$h_{1i} = w_{11} + w_{21} \cdot x_i$$

$$h_{2i} = w_{12} + w_{22} \cdot x_i$$

$$y_i = w_1 \cdot f(h_{1i}) + w_2 \cdot f(h_{2i})$$

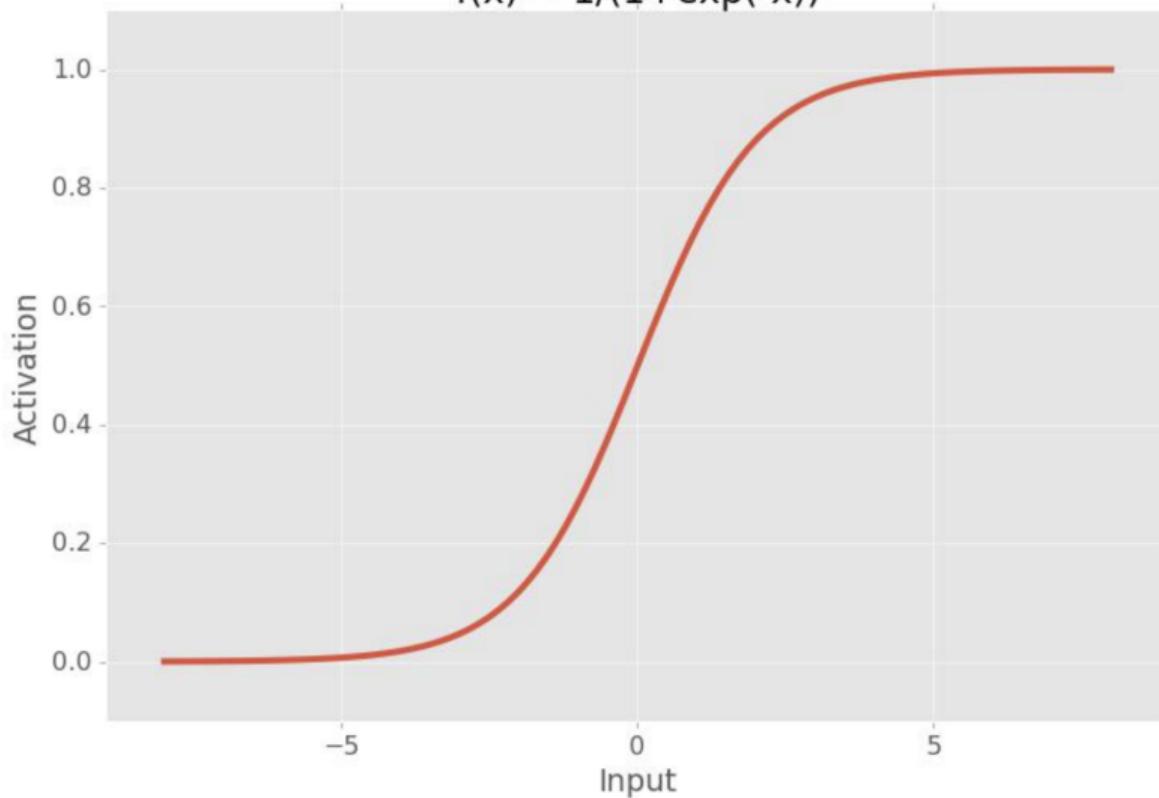
$$y = h \cdot W_2 = f(X \cdot W_1) \cdot W_2 \neq X \cdot A$$

Почему нельзя взять такую функцию?

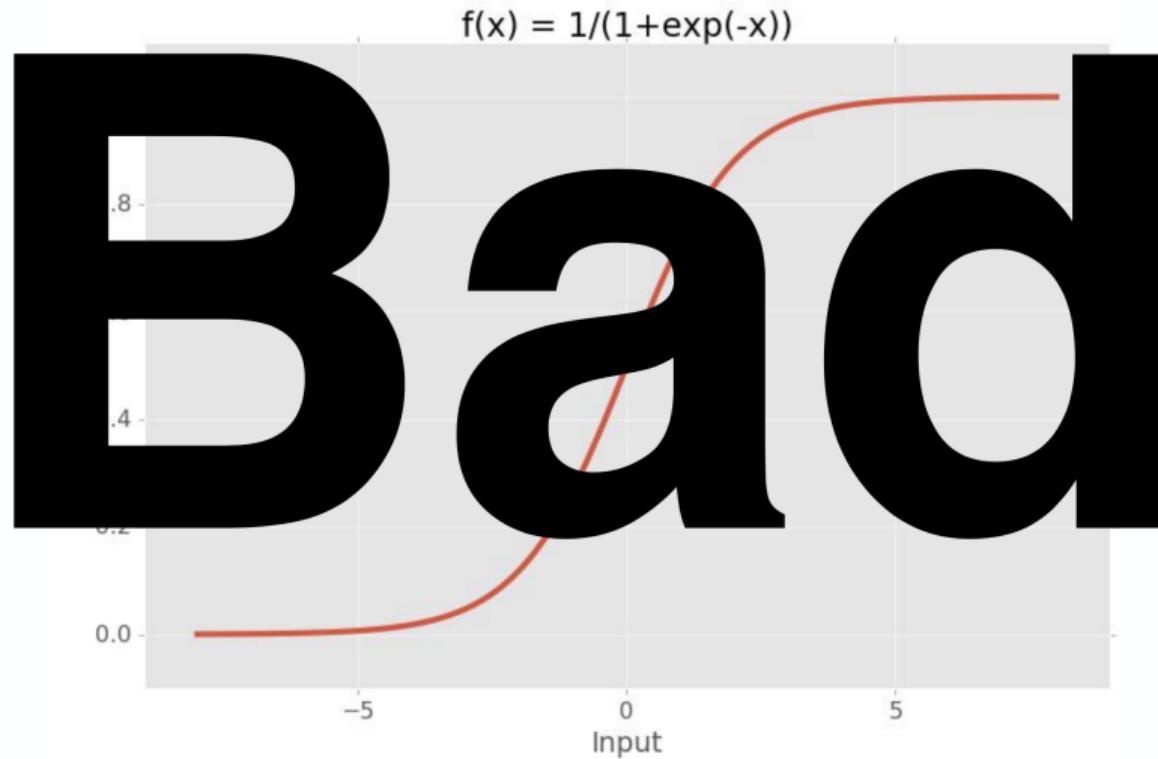


Сигмоида

$$f(x) = 1/(1+\exp(-x))$$



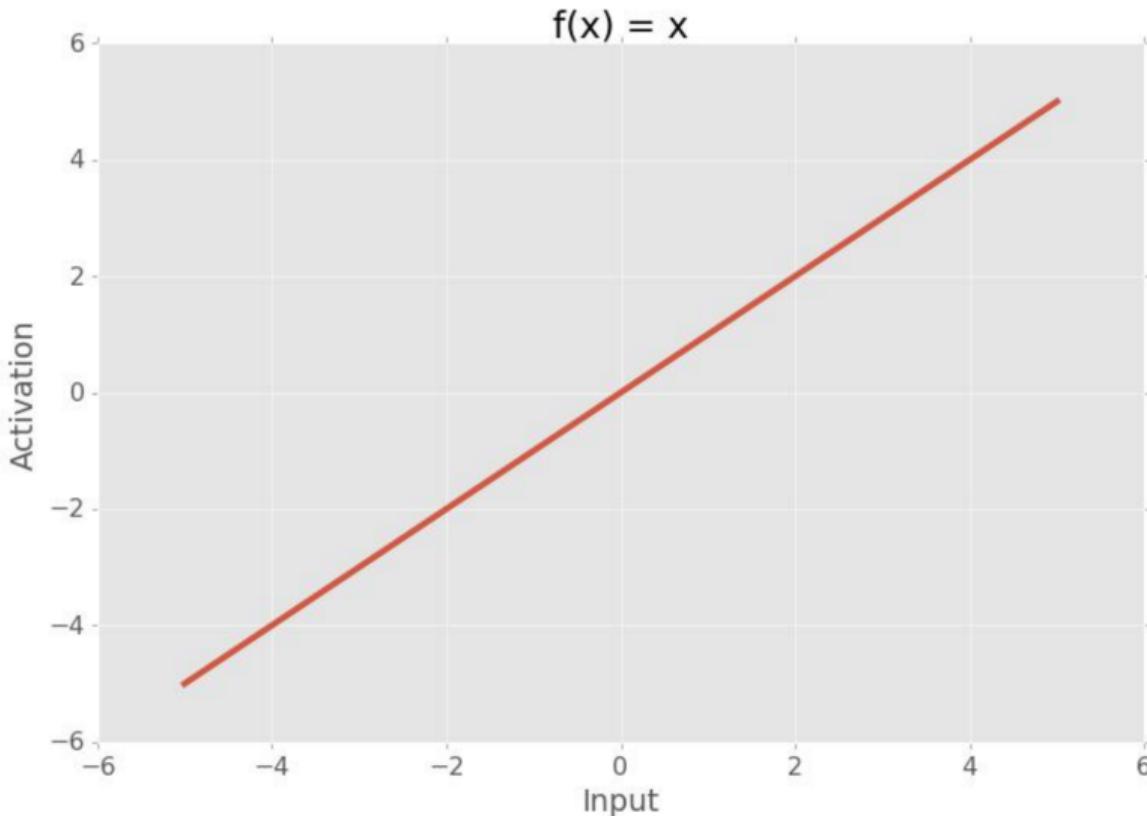
Сигмоида



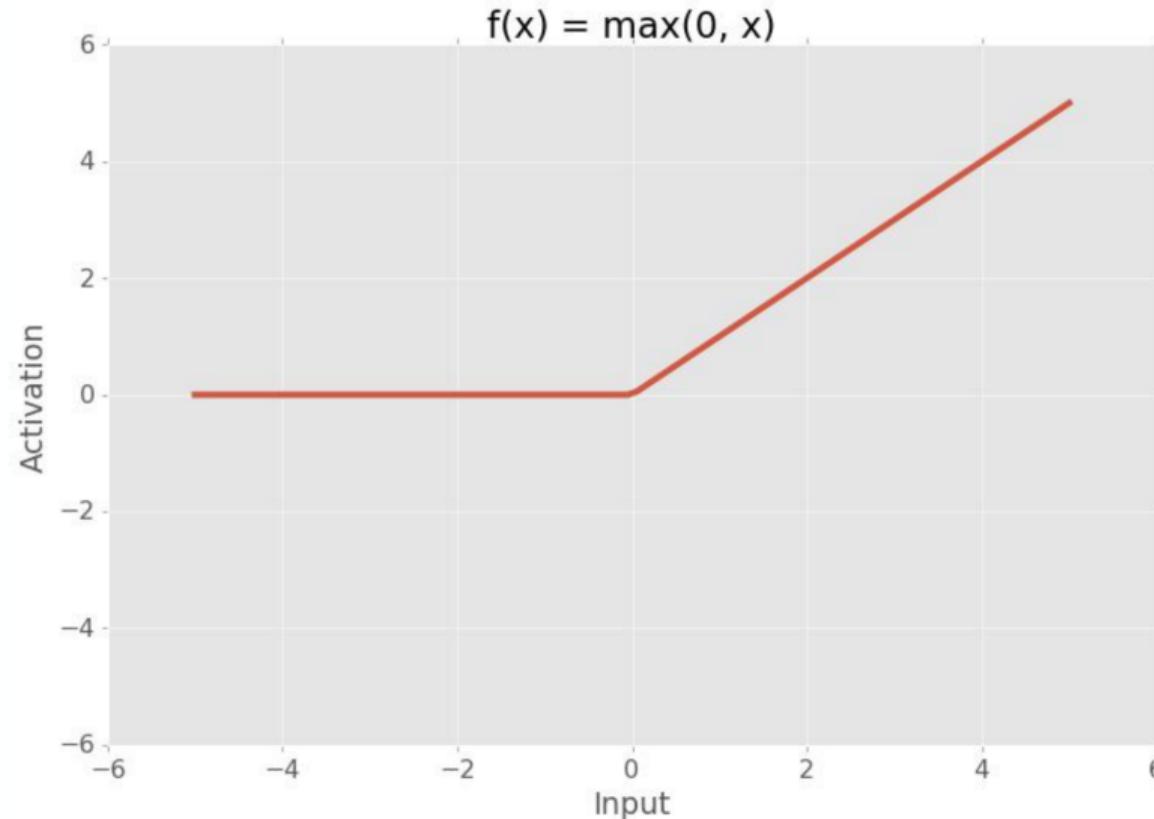
Сигмоида

- В маленьких сетках сигмоиду можно смело использовать
- В глубоких сетях из-за сигмоиды возникает **паралич сети**, сеть прекращает учиться
- Про него подробнее мы поговорим чуть позже
- Нужна другая нелинейная функция активации

От линейной активации ...

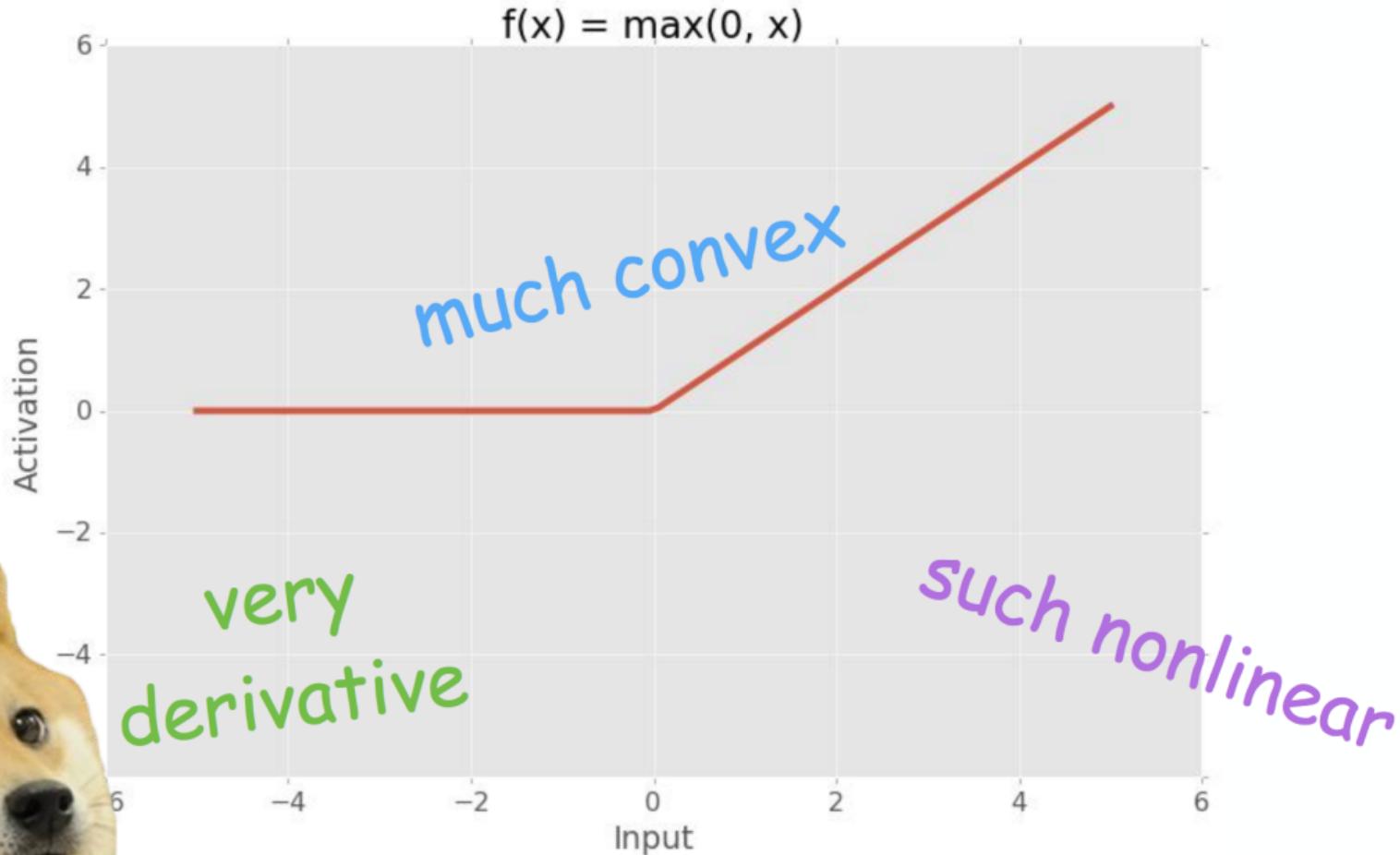


... к нелинейной

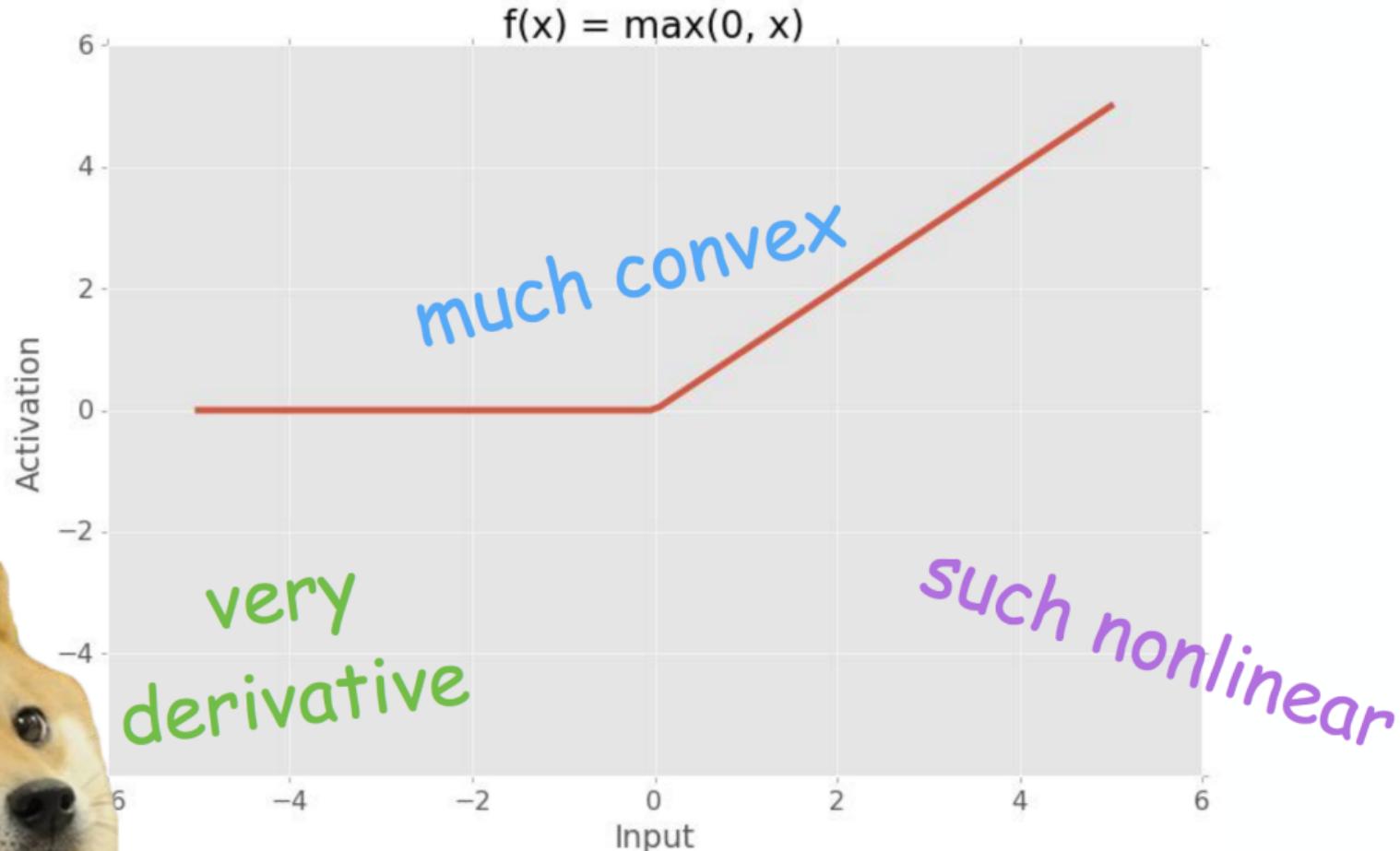


ReLU

$$f(x) = \max(0, x)$$



ReLU

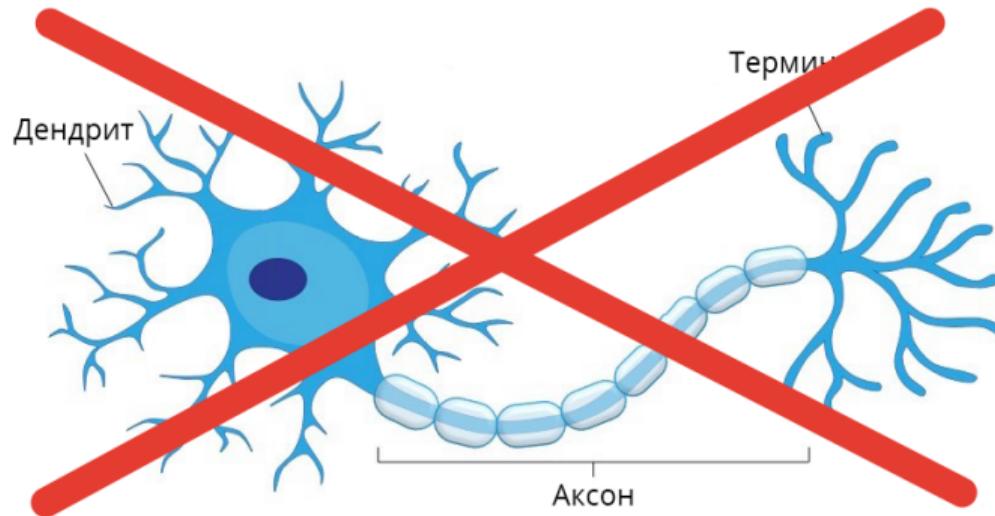


Эта шутка была мною нагло украдена у моего друга Ильи Езепова

Персепtron



Нейрон



Хватит нам врать!

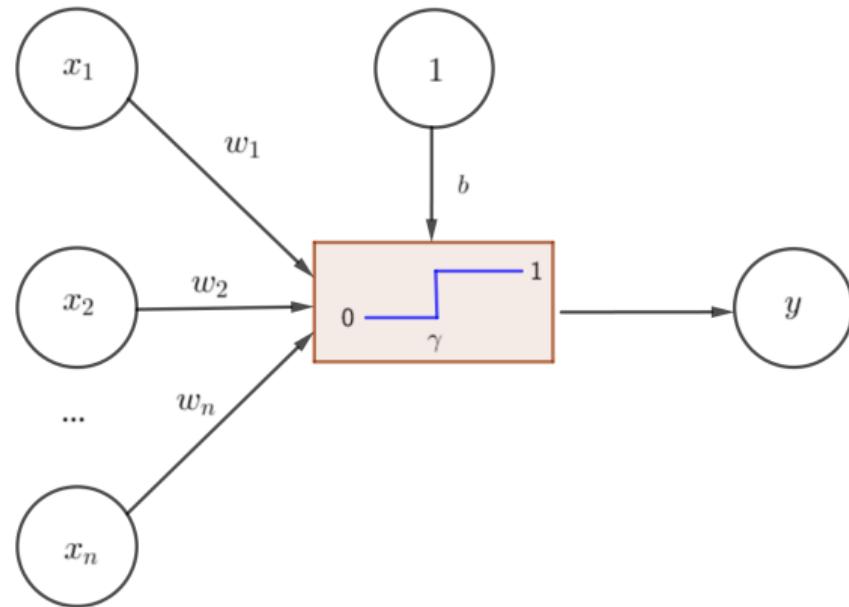
Мышь Розенблата

- Термин нейронная сеть пришёл из биологии, его придумали 70 лет назад
- Оказалось, что мозг устроен гораздо сложнее
- Продуктивнее думать про нейросетки, как про вычислительные графы



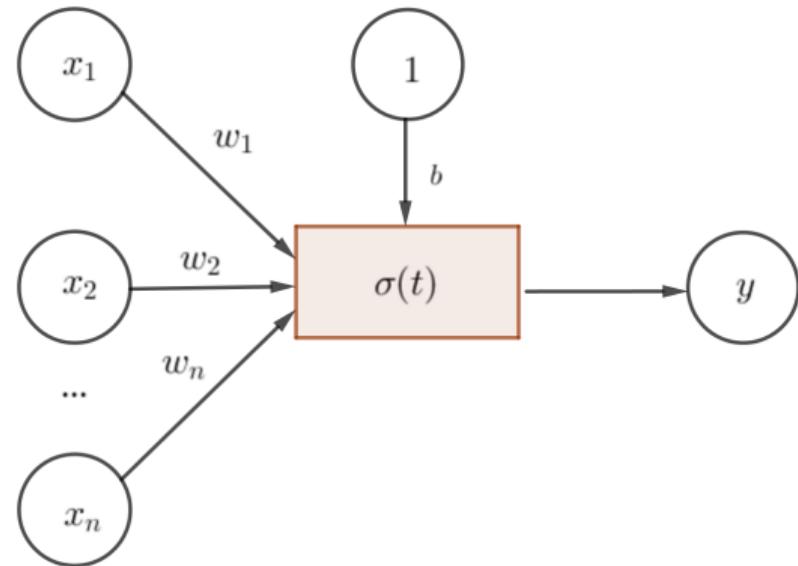
Не забыли его?

Персептрон Розенблатта (1950)



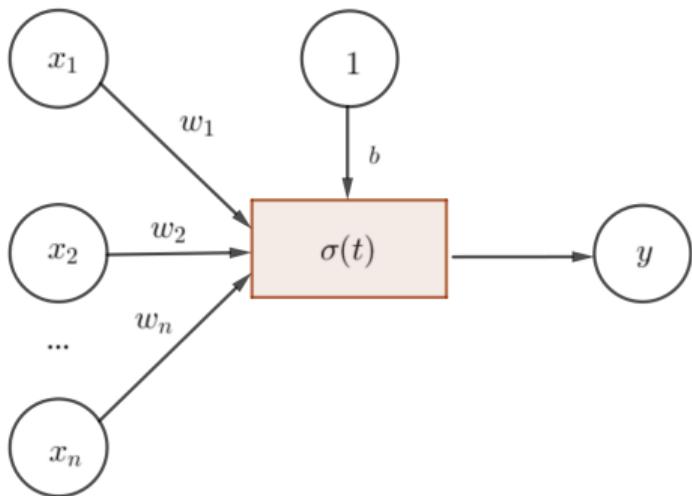
$$y = \begin{cases} 1, & \text{если } \sum w_i x_i \geq \gamma \\ 0, & \text{если } \sum w_i x_i < \gamma \end{cases}$$

Функция активации



- Функция активации $\sigma(t)$ вносит нелинейность, она может быть любой

Линейная регрессия

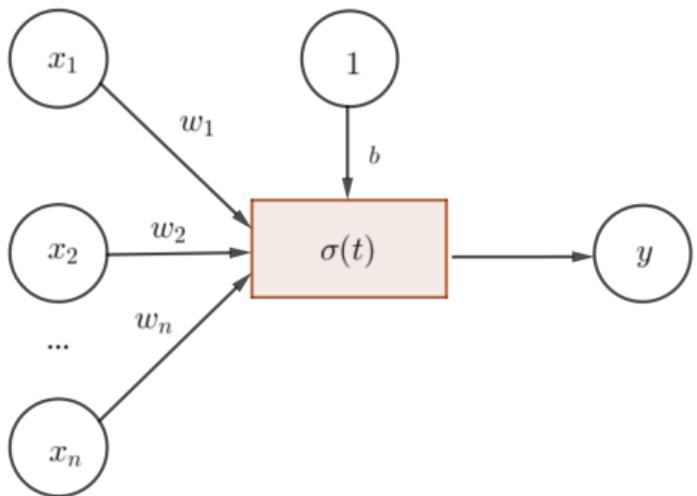


Нейрон с линейной функции
активации — это линейная регрессия...

$$\sigma(t) = t$$

$$y = w_0 + w_1 \cdot x_1 + \dots + w_n \cdot x_n$$

Логистическая регрессия

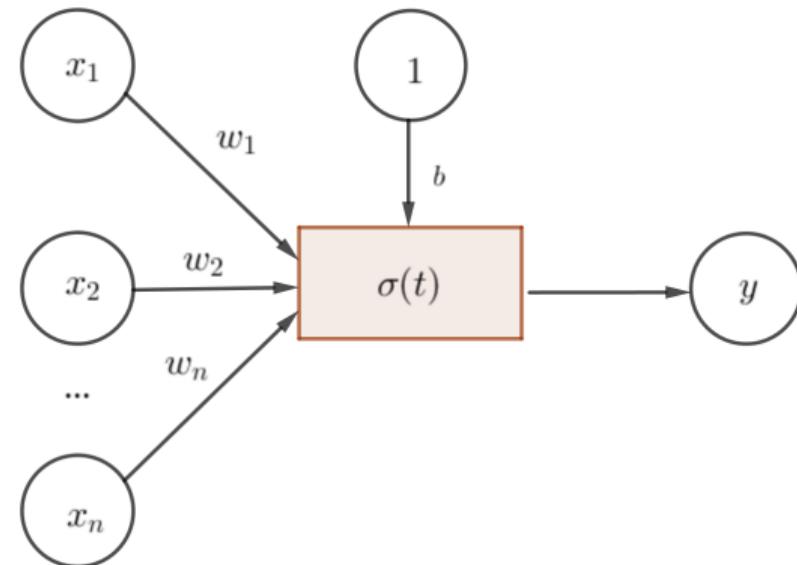


Нейрон с сигмоидом в качестве функции активации — это логистическая регрессия...

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

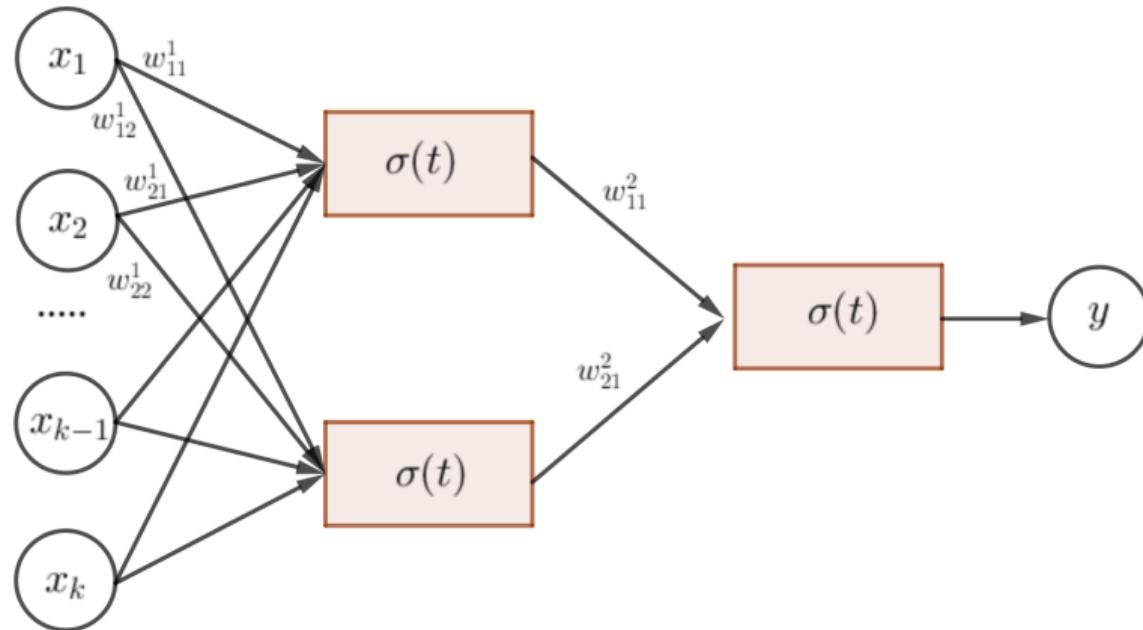
$$P(y = 1 | x) = \sigma(w_0 + w_1 \cdot x_1 + \dots + w_n \cdot x_n)$$

Функция активации



$$y = \sigma(X \cdot W)$$

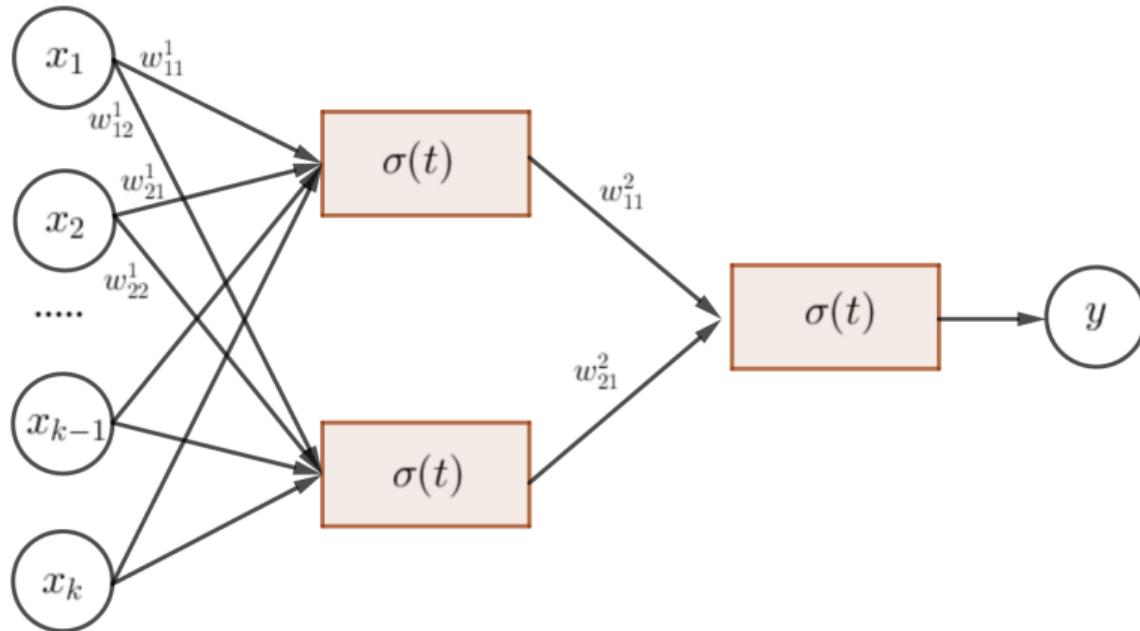
Две регрессии скрепили третьей



$$h_j = \sigma(w_0 + w_{j1}^1 \cdot x_1 + \dots + w_{jk}^1 \cdot x_k)$$

$$y = \sigma(w_{11}^2 \cdot h_1 + w_{21}^2 \cdot h_2)$$

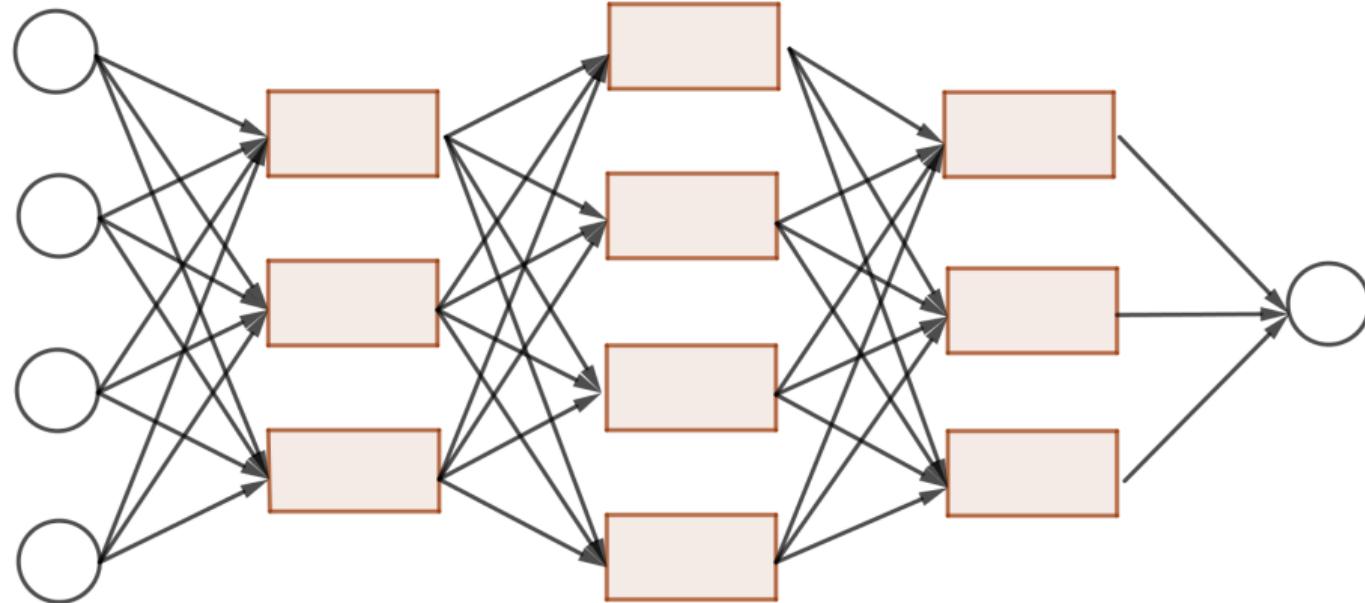
MLP (multi-layer perceptron)



$$h = \sigma(X \cdot W)$$

$$y = \sigma(h \cdot W)$$

Армия из регрессий



Universal Approximation Theorem (Цыбуленко, 1989)

Пусть

- функция активации $\sigma(t)$ непрерывна и сигмоидальна

$$\lim_{t \rightarrow -\infty} \sigma(t) = 0 \quad \lim_{t \rightarrow +\infty} \sigma(t) = 1,$$

- функция $f(x)$ непрерывна на $[0; 1]^d$,
- функция $g(x)$ конечная двухслойная нейронная сеть,

тогда $\forall \varepsilon > 0, \forall f(x)$ существует число нейронов и набор весов для нейронной сети такие, что

$$\sup_{x \in [0; 1]^d} |f(x) - g(x)| \leq \varepsilon.$$

Графическое доказательство теоремы:

<http://neuralnetworksanddeeplearning.com/chap4.html>

Universal Approximation Theorem (Цыbenko, 1989)

- Любая непрерывная функция может быть сколь угодно точно аппроксимирована нейронной сетью с двумя скрытыми слоями с нелинейной функцией активации
- Что ещё можно пожелать?

UAT не конструктивна

- Мы обучаем нейросетки градиентным спуском, мы легко можем переобучиться
- Для настройки весов существует очень большая вариативность и большой простор для переобучения
- Нам гарантируется, что существует конечная ширина (число нейронов), которое приближает функцию, но на практике это число может оказаться очень большим и сетка будет очень широкой

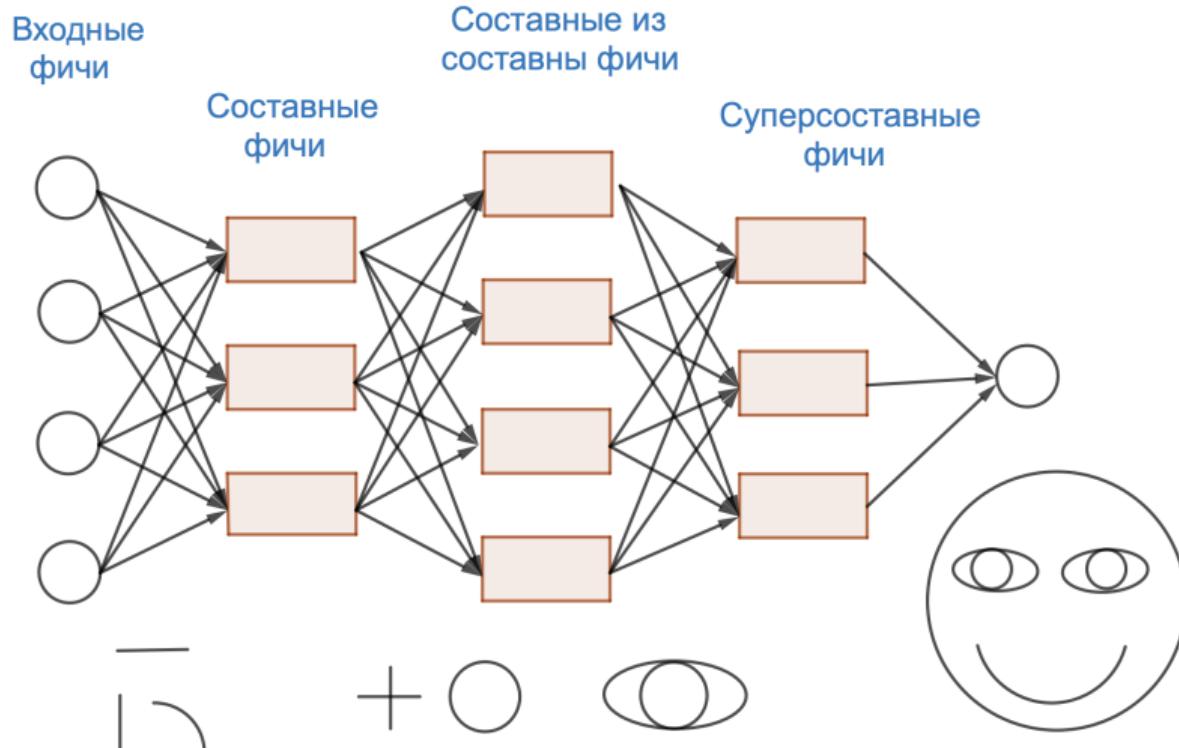
The background of the image is a deep blue ocean. Sunlight filters down from the surface in bright, dappled rays, creating a play of light and shadow on the undulating waves. The overall atmosphere is serene and suggests depth.

Going Deeper

Мотивация

- Персептрон может решить любую проблему, но это дорого
- Глубокие архитектуры часто позволяют выразить то же самое, приблизить те же функции гораздо более эффективно, чем неглубокие
- Каждый новый слой сетки будет работать всё с более сложными фичами

Армия из регрессий



MLP не отходя от браузера

Epoch 000,515 Learning rate 0.03 Activation Tanh Regularization None Regularization rate 0 Problem type Classification

DATA
Which dataset do you want to use?
 
 
Ratio of training to test data: 20%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
 X_1 
 X_2 
 X_1^2 
 X_2^2 
 $X_1 X_2$ 
 $\sin(X_1)$ 
 $\sin(X_2)$ 

2 HIDDEN LAYERS

+ - + -

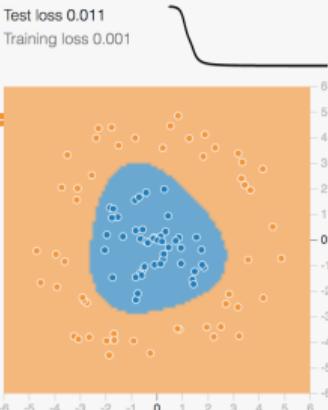
4 neurons 2 neurons

This is the output from one neuron. Hover to see it larger.

The outputs are mixed with varying weights, shown by the thickness of the lines.

OUTPUT

Test loss 0.011
Training loss 0.001



Colors shows data, neuron and weight values.
-1 0 1

Show test data Discretize output

MLP не отходя от браузера

Epoch 000,515 Learning rate 0.03 Activation Tanh Regularization None Regularization rate 0 Problem type Classification

DATA
Which dataset do you want to use?

Ratio of training to test data: 20%
Noise: 0
Batch size: 10
REGENERATE

FEATURES
Which properties do you want to feed in?
 X_1
 X_2
 X_1^2
 X_2^2
 $X_1 X_2$
 $\sin(X_1)$
 $\sin(X_2)$

2 HIDDEN LAYERS
+ - 4 neurons
+ - 2 neurons
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.

OUTPUT
Test loss 0.011
Training loss 0.001
Однаковые!
Линия совсем тонкая, веса на ней нулевые
Boooще ничего не делает!

Show test data Discretize output

Ещё одна аналогия

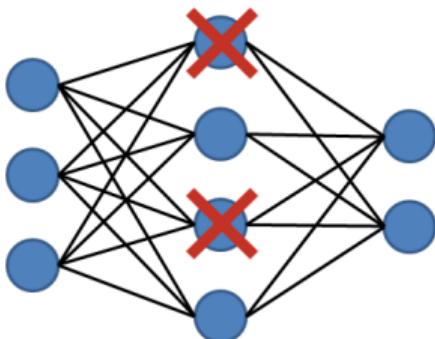


Нейросети — конструктор LEGO



Слои бывают разными

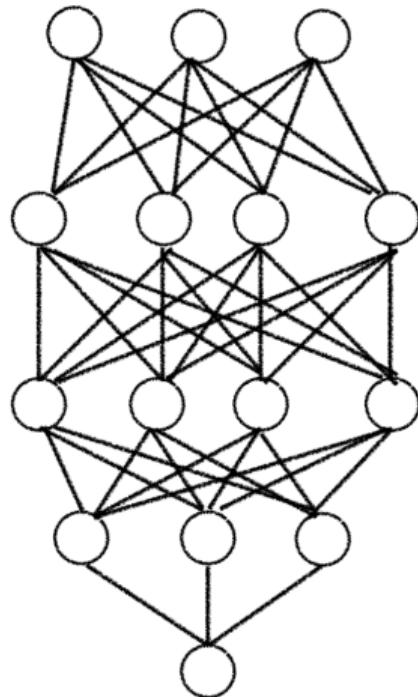
- Слой, который просто взвешивает входы называется **полносвязным**.
- Слои бывают очень разными. Например, **Dropout**: с вероятностью p отключаем нейрон. Такой слой препятствует переобучению и делает нейроны более устойчивыми к случайным возмущениям.



Функции активации бывают разными

Название функции	Формула $f(x)$	Производная $f'(x)$
Логистический сигмоид σ	$\frac{1}{1+e^{-x}}$	$f(x)(1-f(x))$
Гиперболический тангенс \tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f^2(x)$
SoftSign	$\frac{x}{1+ x }$	$\frac{1}{(1+ x)^2}$
Ступенька (функция Хевисайда)	$\begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	0
SoftPlus	$\log(1 + e^x)$	$\frac{1}{1+e^{-x}}$
ReLU	$\begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	$\begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$
Leaky ReLU, Parameterized ReLU	$\begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$	$\begin{cases} a, & x < 0 \\ 1, & x \geq 0 \end{cases}$
ELU	$\begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$	$\begin{cases} f(x) + \alpha, & x < 0 \\ 1, & x \geq 0 \end{cases}$

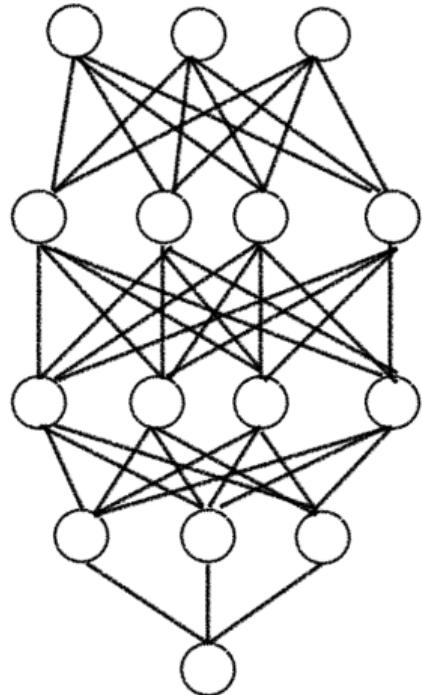
Архитектуры бывают разными



Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$
Output	

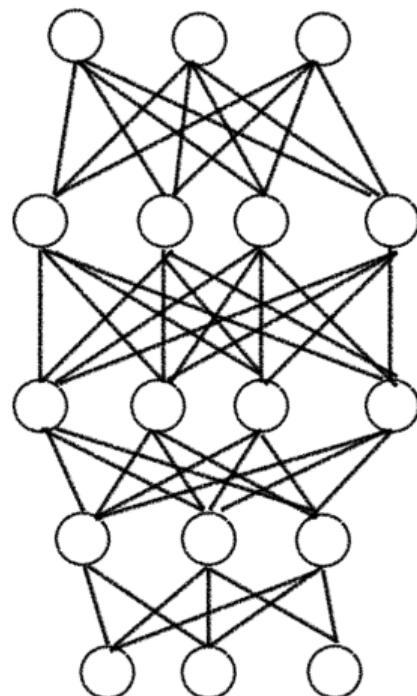
Каждый слой — просто функция, каждая сетка — конструктор LEGO

Регрессия



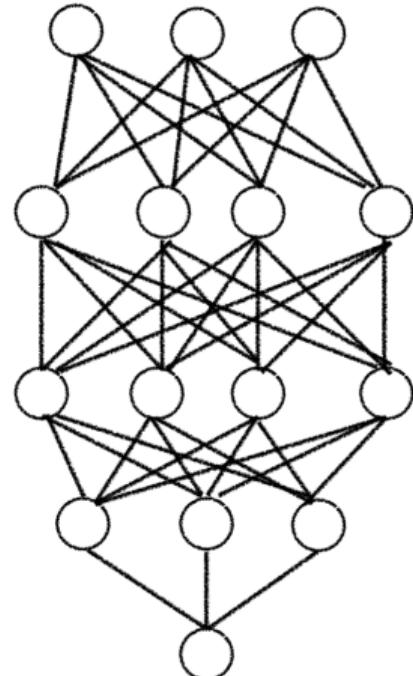
Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$
Output	

Мультирегрессия



Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$
Output	

Классификация



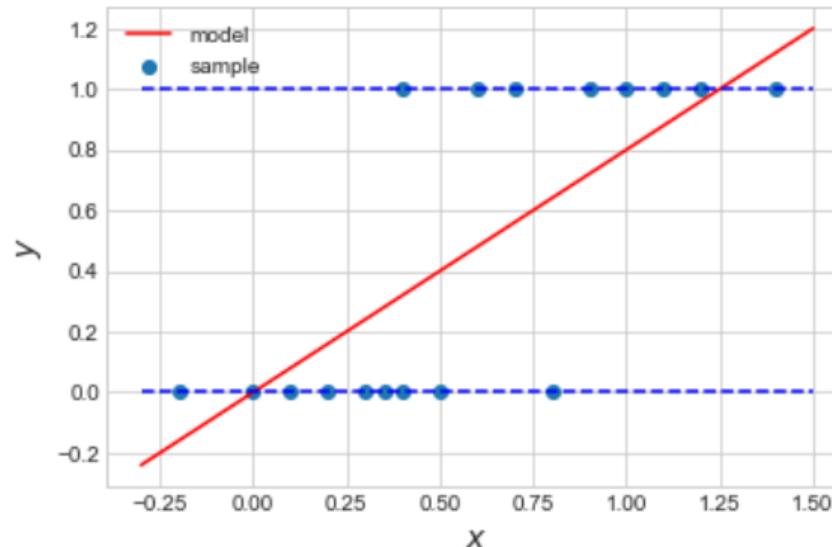
Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$
Sigmoid	$\sigma(x)$
Output	

Классификация

- $y \in \{0, 1\}$ – целевая переменная, X – признаки
- Модель: $y = w_0 + w_1x_1 + w_2x_2 + \dots + w_kx_k$

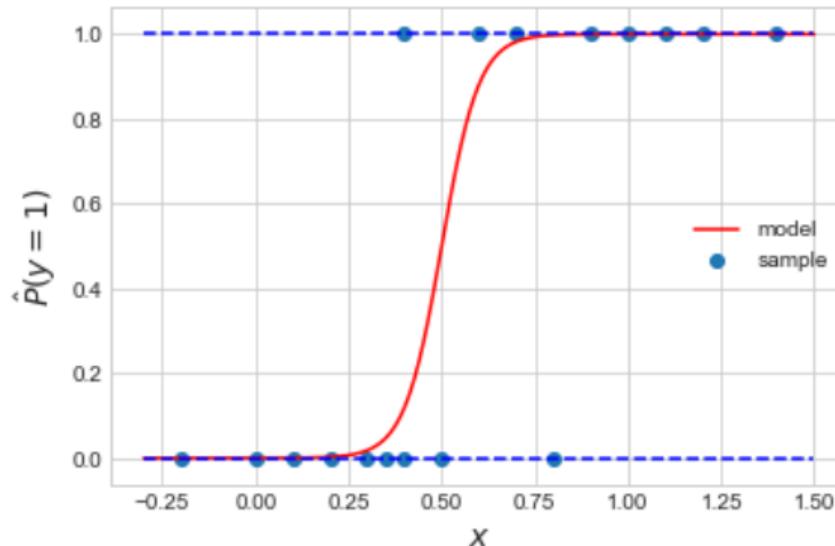
Классификация

- $y \in \{0, 1\}$ — целевая переменная, X — признаки
- Модель: $y = w_0 + w_1x_1 + w_2x_2 + \dots + w_kx_k$



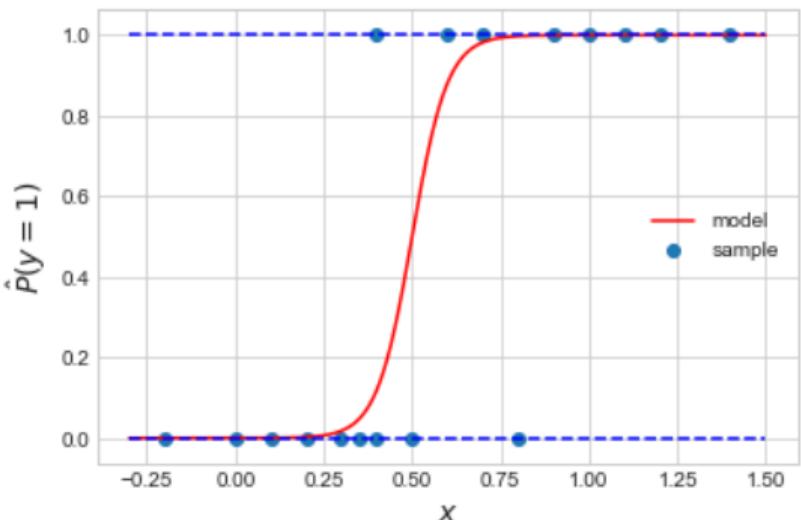
Классификация

- $y \in \{0, 1\}$ — целевая переменная, X — признаки
- Модель: $y = [w_0 + w_1x_1 + w_2x_2 + \dots + w_kx_k > \gamma]$



Классификация

- $y \in \{0, 1\}$ — целевая переменная, X — признаки
- Модель: $P(y = 1 | w) = F(w_0 + w_1 x_1 + w_2 x_2 + \dots + w_k x_k)$

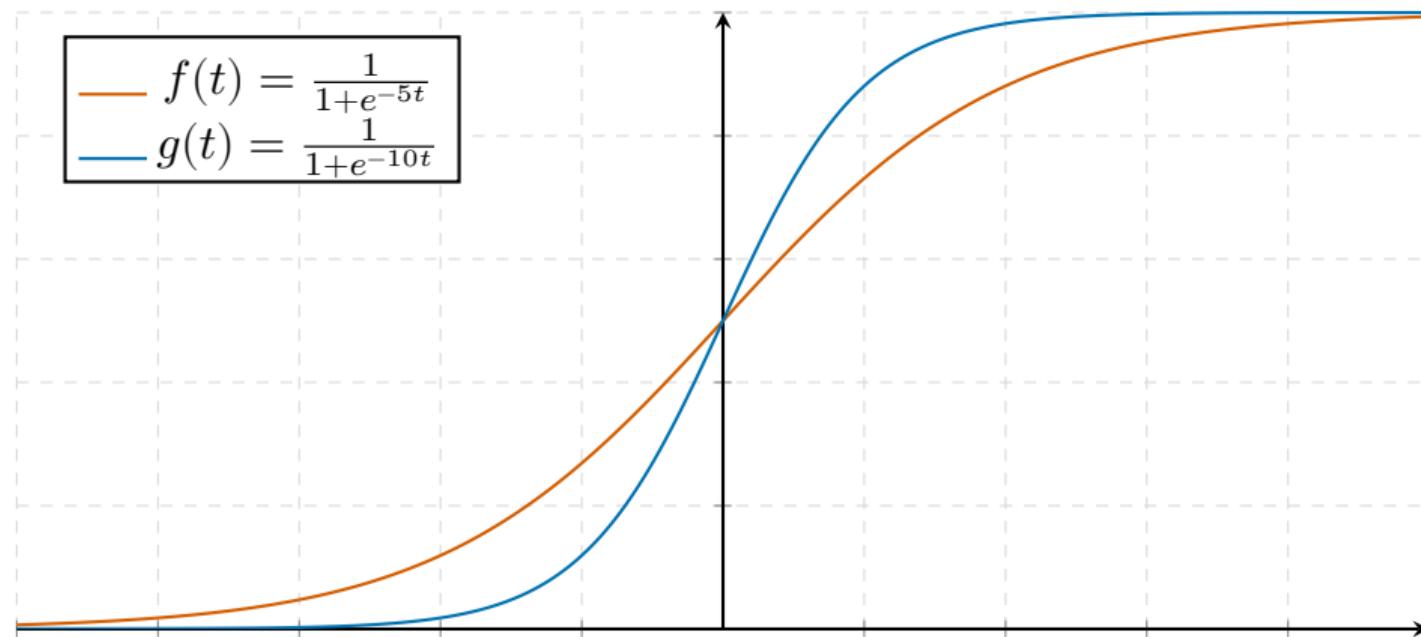


- В качестве $F(t)$ можно взять любую функцию распределения
- Если взять сигмоиду, модель будет интерпретируемая

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

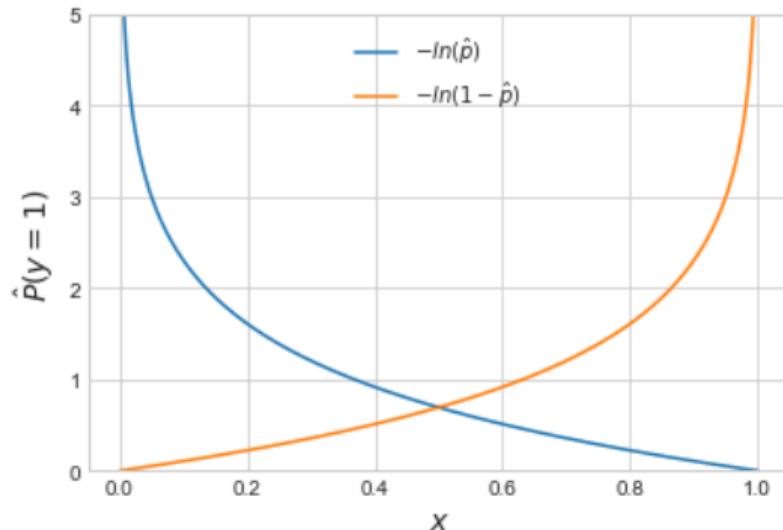
$$\sigma'(t) = \sigma(t) \cdot (1 - \sigma(t))$$

Сигмоида



Функция потерь

- Наши y принимают значения 0 и 1
- Если $y = 1$, хотим большое $\hat{p} = \hat{P}(y = 1)$, но чем ближе \hat{p} к 1, тем меньше хотим его увеличить
- Если $y = 0$, хотим большое $(1 - \hat{p})$, получается функция потерь:



$$\text{logloss} = -\frac{1}{n} \sum_{i=1}^n y_i \cdot \ln \hat{p}_i + (1 - y_i) \cdot \ln (1 - \hat{p}_i)$$

Пример: два класса



$$\begin{array}{c} t \qquad \sigma(t) \qquad \text{logloss} \\ \Rightarrow \quad \sigma(XW + b) \quad - (0 \cdot \ln 0.99 + (1 - 0) \cdot \ln(1 - 0.99)) \\ \begin{array}{c|c} \text{Input} & \\ \hline \text{Fully connected layer (FC)} & XW + b \\ \text{ReLU} & \max(0, x) \\ \text{Dropout} & \text{Bern}(p) \\ \text{FC} & XW + b \\ \text{ReLU} & \max(0, x) \\ \text{FC} & XW + b \end{array} \end{array}$$



$$\begin{array}{c} t \qquad \sigma(t) \qquad \text{logloss} \\ \Rightarrow \quad \sigma(XW + b) \quad - (1 \cdot \ln 0.26 + (1 - 1) \cdot \ln(1 - 0.26)) \\ \begin{array}{c|c} \text{Input} & \\ \hline \text{Fully connected layer (FC)} & XW + b \\ \text{ReLU} & \max(0, x) \\ \text{Dropout} & \text{Bern}(p) \\ \text{FC} & XW + b \\ \text{ReLU} & \max(0, x) \\ \text{FC} & XW + b \end{array} \end{array}$$

Пример: два класса



\Rightarrow

Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$

t

$\sigma(t)$

logloss

$\Rightarrow 10 \Rightarrow [0.01, 0.99] \Rightarrow -\ln 0.01$

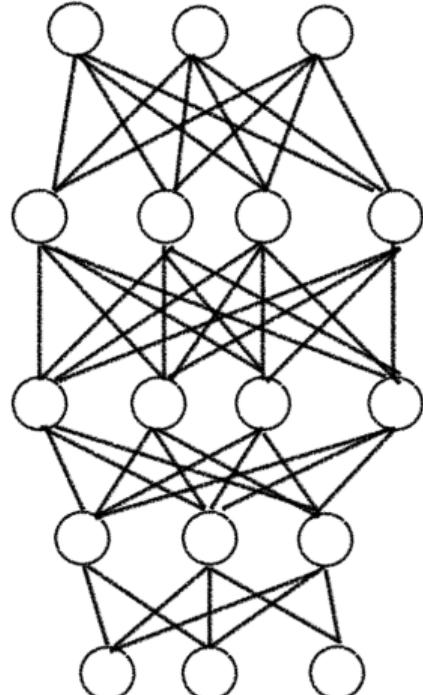


\Rightarrow

Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$

$\Rightarrow -1 \Rightarrow [0.74, 0.26] \Rightarrow -\ln 0.26$

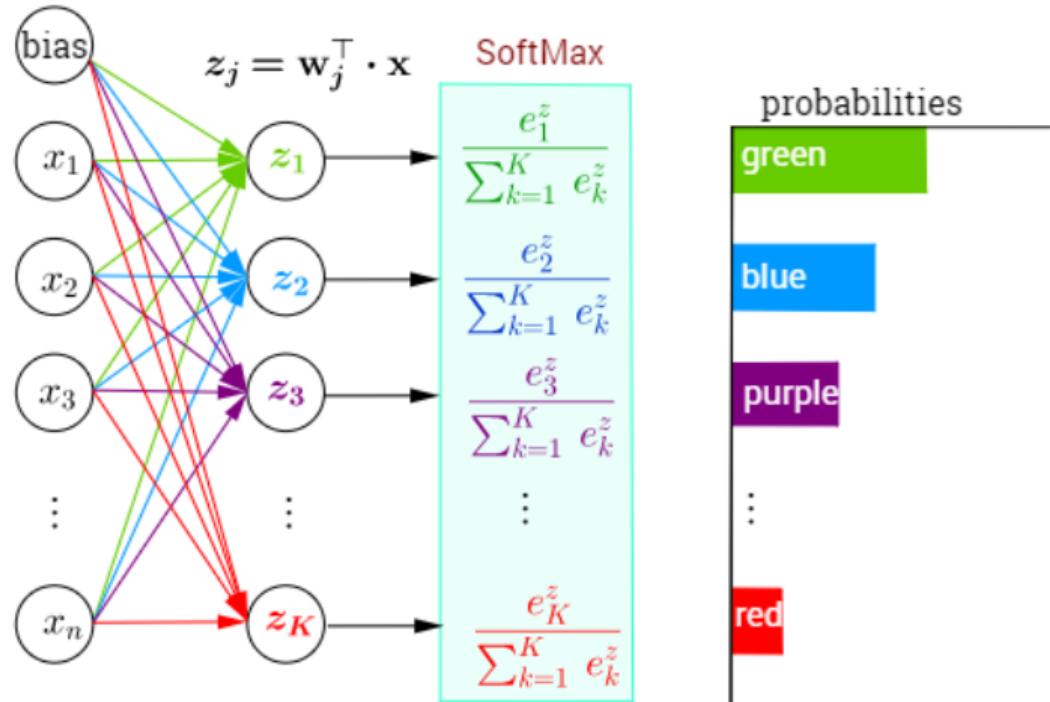
Мультиклассификация



Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$
Softmax	$\text{softmax}(x)$
Output	

Мультиклассификация

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



Softmax (мягкий максимум)

Много классов, $y \in 1, \dots, K$

$$(w_1^T x, \dots, w_K^T x)$$



$$(e^{z_1}, \dots, e^{z_K})$$



$$\left(\frac{e^{z_1}}{\sum_{k=1}^K e^{z_k}}, \dots, \frac{e^{z_K}}{\sum_{k=1}^K e^{z_k}} \right)$$

Потери (кросс-энтропия):

$$\text{logloss} = - \sum_{i=1}^n \sum_{k=1}^K [y_i = k] \cdot \ln \frac{e^{w_k^T x_i}}{\sum_{j=1}^K e^{w_j^T x_i}}$$

Пример: три класса

t

Softmax

logloss



\Rightarrow

Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$

$$\Rightarrow [5, 4, 2] \Rightarrow [0.70, 0.26, 0.04] \Rightarrow -\ln 0.71$$



\Rightarrow

Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$

$$\Rightarrow [4, 2, 8] \Rightarrow [0.02, 0.00, 0.98] \Rightarrow -\ln 0.98$$



\Rightarrow

Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$

$$\Rightarrow [4, 4, 1] \Rightarrow [0.49, 0.49, 0.02] \Rightarrow -\ln 0.49$$

Важный нюанс

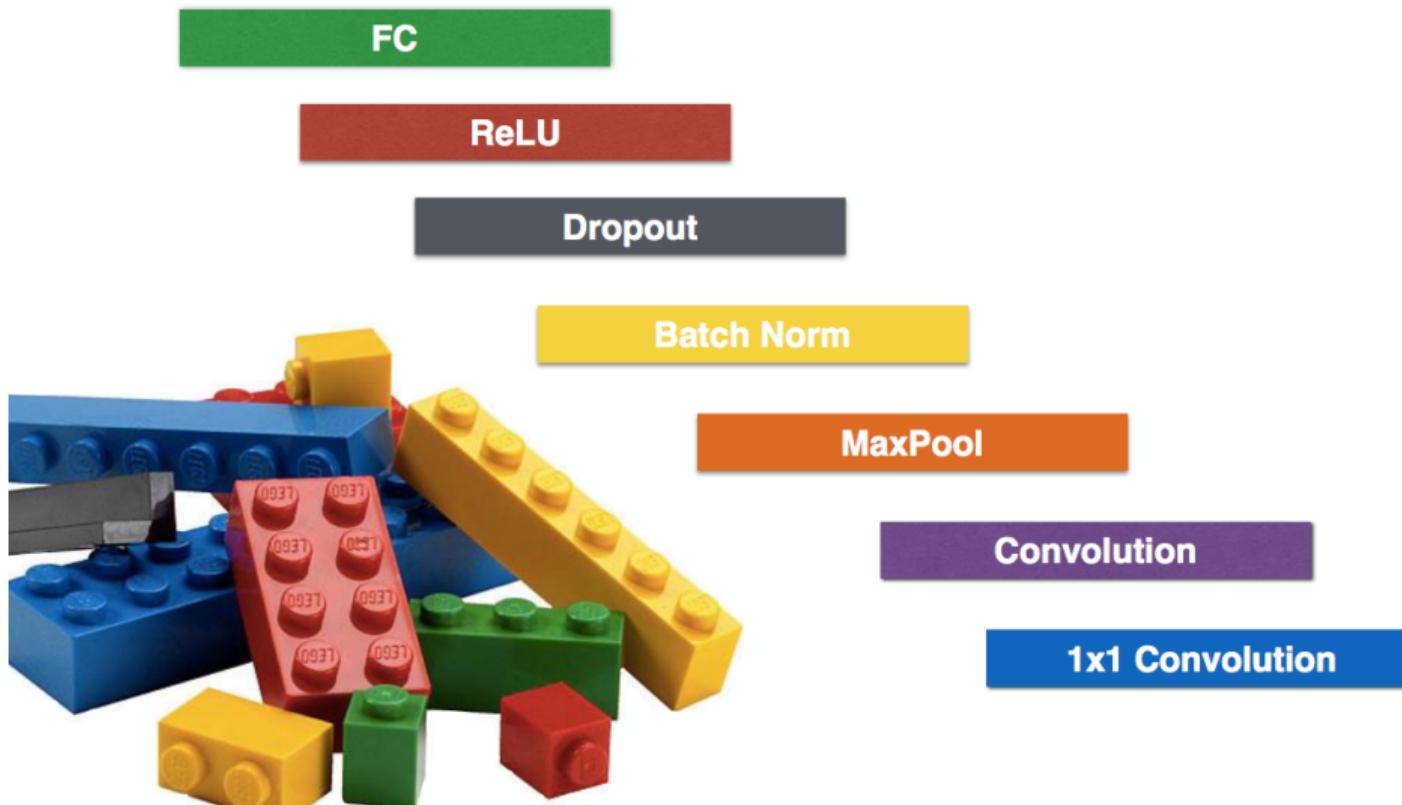
- При поиске Softmax мы ищем экспоненты, в памяти компьютера может произойти переполнение из-за больших чисел
- Если добавить ко всем входам нейронки одинаковую константу, значение Softmax не изменится:

$$\frac{e^{z_i+c}}{\sum_{k=1}^K e^{z_k+c}} = \frac{e^c \cdot e^{z_i}}{e^c \cdot \sum_{k=1}^K e^{z_k}} = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

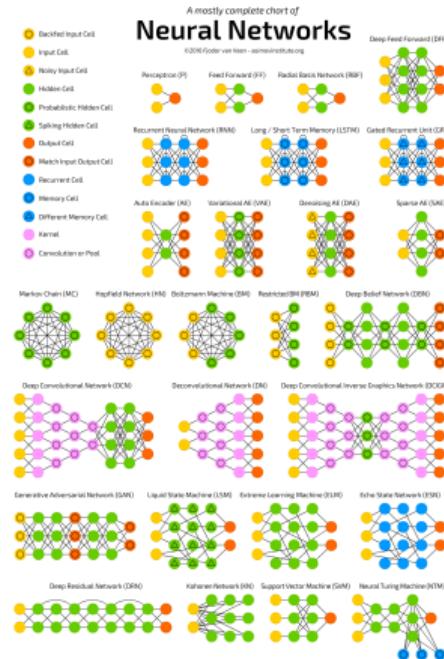
- Обычно считают устойчивый к переполнению Softmax:

$$\text{Softmax}(z_1, \dots, z_K) = \text{Softmax}(z_1 - \max_i(z_i), \dots, z_K - \max_i(z_i))$$

Нейросети - конструктор LEGO



Не нейросеть, а граф вычислений



<https://www.asimovinstitute.org/neural-network-zoo>
<https://habr.com/ru/company/wunderfund/blog/313696/>

Учим свою первую нейросеть!