

Глубинное обучение

Дмитрий Никулин

24 марта 2021

Лекция 1: вводная

Я

Agenda

- О том, каким будет курс
- Немного истории и важные тренды
- От регрессии к нейросетке
- Нейросетки - конструктор Lego
- Запускаем свою первую нейросетку



Правила игры

Про пары

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

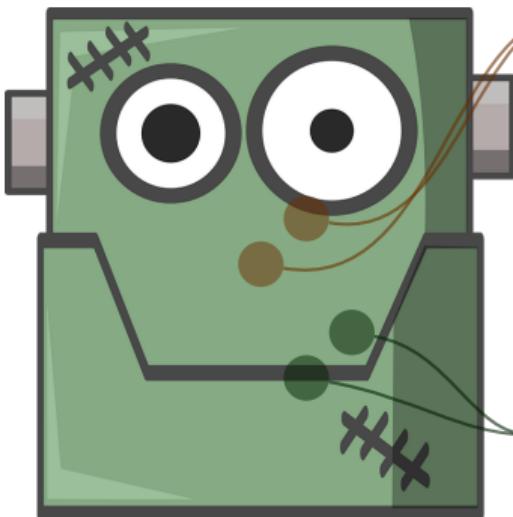
Про источники

- лекции и задачки на 95% заимствованы с предыдущей итерации ДПО от [Филиппа Ульянкина](#)
- ноутбуки для семинаров и домашек частично берём из [курса ШАДа](#) и [курса на майноре ИАД](#)

Что посмотреть про нейронки

- Если ничего не знаете про машинное обучение, смотрите [вводный курс от Яндекса и МФТИ](#). Для тех, кто мало знает про ML.
- Курс по DL для студентов ФКН ВШЭ. Для тех, кто хочет посмотреть, как читают курс по DL на ФКН для бакалавров.
- Курс нейронок, который читают в ШАДе 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 ШАДе.
- Бесплатный курс по DL от Samsung AI Center. Вводный курс в нейросетки на PyTorch. Для тех, кто хочет упор на компьютерное зрение.

Структура курса (примерно)



- Базовая часть курса:
- От регрессии к нейросети
- Введение в PyTorch
- 50 оттенков градиентного спуска, backpropagation
- Эвристики и приёмы для обучения сеток

- Разные подобласти:
- Свёрточные сети, локализация, сегментация, перенос стиля
- Автокодировщики, генеративные модели
- Рекуррентные нейронные сети
- Введение в NLP, эмбеддинги, автопереводы, трансформеры
- Обучение с подкреплением

Немного истории



Первый заход

- 1956 — Дартмутский семинар, море оптимизма
- 1958 — Перцептрон Розенблатта
- 1966 — ALPAC report, после которого угас интерес к автоматическому переводу между русским и английским
- 1969 — Марвин Минский и Сеймур Пейперт опубликовали книгу «Перцептроны» 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 (1980-е)

- 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 мнение консилиума врачей в 65% случаев, лучше любого врача)
- 1980-е — алгоритм обратного распространения ошибки (backpropagation) позволил обучать сети за линейное время
- Ренессанс нейронных сетей

Зима близко

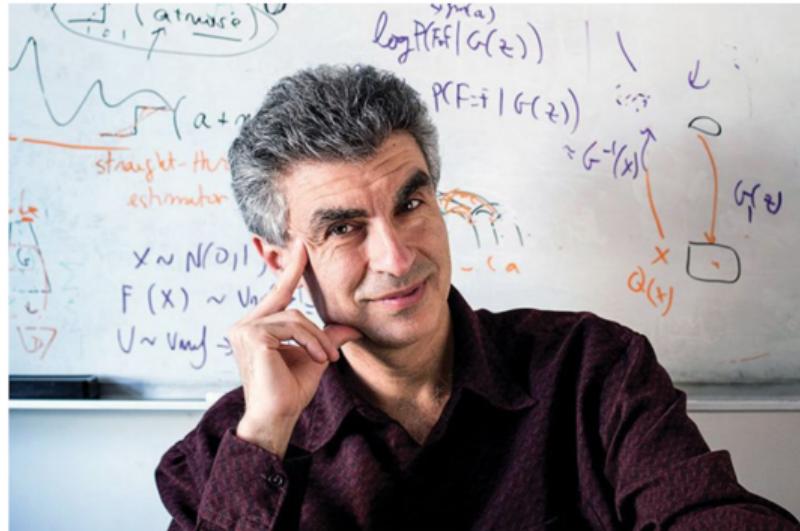
- Новая волна оптимизма
- Экспертная система XCON сэкономила компании DEC около 40 миллионов долларов за 6 лет
- Завышенные ожидания снова лопнули
- 1990-е — ударными темпами развивается классическое машинное обучение



Революция (2005-2006)



Джеффри Хинтон
Geoffrey Hinton
(университет Торонто)

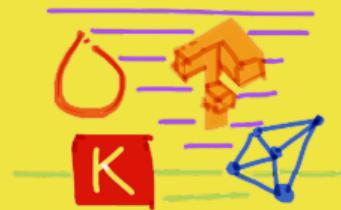


Йошуа Бенджио
Yoshua Bengio
(университет Монреаля)

Революция

- 2005--2006: группы Хинтона и Бенджио 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!

Интерлюдия: фреймворки



Фреймворки

theano



Монреальский
университет (2007)

Static Computational
Graph



Google

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

Static Dynamic
Computational Graph

P Y T H O N



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



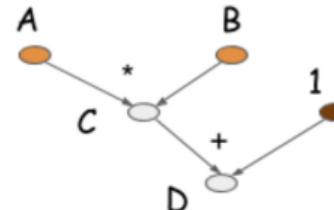
Сначала задали граф вычислений,
а потом уже считали

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 т.п.)
- Довольно сложно искать ошибки из-за того, что сначала задаётся граф вычислений

Andrej Karpathy, Head of AI @ Tesla npo Tensorflow (2017)



Andrej Karpathy 
@karpathy

...

I've been using PyTorch a few months now and I've never felt better. I have more energy. My skin is clearer. My eye sight has improved.

9:56 PM · May 26, 2017 · Twitter Web Client

430 Retweets 68 Quote Tweets 1,832 Likes

<https://twitter.com/karpathy/status/868178954032513024>

Denis Vorotyntsev, DS @ Unity про Tensorflow (2021)



Data Scientist
@dsunderhood

Нейронки на работе я пишу на tf, по своим проектам - на торче. У tf есть красивая обертка - keras. Именно она привлекает много новичков. Однако если ты лишишь что-то сложнее классификации ирисов, то кривая сложности взмывает до небес моментально. Сейчас поговорим про tf.

Translate Tweet

11:28 AM - Mar 16, 2021 · Twitter Web App

2 Retweets 3 Quote Tweets 44 Likes



Data Scientist @dsunderhood · Mar 16

Replying to @dsunderhood
Тф 1 с его сессиями и графами был создан не для людей. Моя коллега говорит что пишет на tf1 - это ходить по минному полю из граблей с заезженными тапами. Дебагинг сеток отступался как класс. Я проявляю научиться писать на tf 1 в уроках от Ng.



Data Scientist @dsunderhood · Mar 16

В его курсе по ДЛ нужно было заполнить ячейкой чтобы выполнить задание. Это я когда увидел with tf.Session() ахах: bla bla bla.



Data Scientist @dsunderhood · Mar 16

Я пытаюсь понять tf1, но горякое введение было слишком высок.



Data Scientist @dsunderhood · Mar 16

Переход на tf2 сделал фреймворк похожим на нормальный, но только "похожий". Главное отличие от торча - то как собирается вычислительный граф. В торче граф динамичен, его можно менять на лету. В tf граф статичен - один раз создаем и баста.



Data Scientist @dsunderhood · Mar 16

В tf можно включить eager execution, что делает граф динамичным, но вы моментально просидите по скорости.



Data Scientist @dsunderhood · Mar 16

Как разница в типе графа влияет на обучение нейронок? Это ускоряет сложные, жирные лайтней, но часто сваивает руки. Акумуляция градиентов, обучение разных кусков модели с разным пр. постепенный фриз/анфриз модели - в торче все это реализовать довольно легко.



Data Scientist @dsunderhood · Mar 16

В tensorflow - сложно или невозможно.



Data Scientist @dsunderhood · Mar 16

К примеру акум градиентов в можно сделать только через кастом тренинг луп. На стаковерфлоу один человек предложил устанавливать пр=0 в момент аккумуляции и пр=0 когда нужен айден: that's how you do grad acum



Data Scientist @dsunderhood · Mar 16

Обучение разных кусков модели с разным пр - ок, можно сделать через кастом тренинг луп в классе модели, но такая возможность появилась только в tf2.



Data Scientist @dsunderhood · Mar 16

Вобщем все можно переписать на кастом тренинг луп как в голове торча, но очень не хочется переписывать большой кусок лайтней чтобы проверить одну идею.



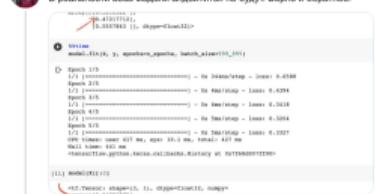
Data Scientist @dsunderhood · Mar 16

Фреймворк - это вообще забей. Все с веесами должны соревноваться компонентами. Нет компонента - все работает, зеркнит все слои, закомпайлиз и анфризнут пару слов. model.summary() выдаст что есть trainable веса.



Data Scientist @dsunderhood · Mar 16

В реальности веса модели определяются не будут. Верно и обратное.



Data Scientist @dsunderhood · Mar 16

Обычный подход с айденом моделей на лету через кастом не работает. Компилияция модели в кастоме нигде. Хотчеч постепенно инфильтрат? Статьи тренинг через кастом, анфриз, компайл, перезапуск.



Data Scientist @dsunderhood · Mar 16

Отступы с одной граблей ты моментально наступаешь на другую как учитывается номер строки при этих плюсах? Не поедет ли у меня пр шадулеры?



Data Scientist @dsunderhood · Mar 16

Иногда нужно проиниц константы в атрибутах модуля. Тф их просто удирает. Потому что это не tf гарифмы. Пишут или делают отдельный класс с атрибутом модуля.

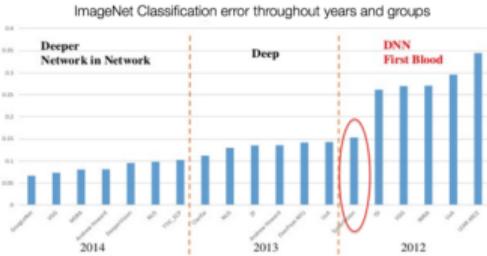
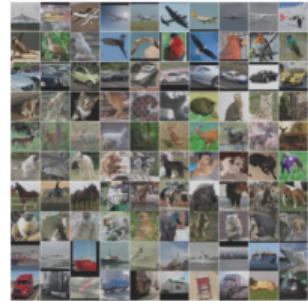


<https://twitter.com/dsunderhood/status/1371739689409974278>

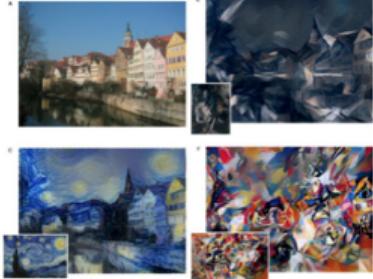
Brave new world

- 2012: аспирант Хинтона, Алекс Крижевский, бьёт в два раза рекорд по точности классификации изображений с помощью глубокой нейросети AlexNet
- 2013: DeepMind выкатывают алгоритм DQN, который учит нейросетевых агентов играть в игры Atari, местами лучше человека
- 2015: точность классификации изображений нейросетями превосходит человеческую
- 2016: алгоритм AlphaGo от DeepMind выигрывает у одного из сильнейших игроков в го
- 2018: появляется универсальная модель для обработки текстов, BERT

ImageNet



Пабло Пикассо



Винсент Ван Гог

Василий Кандинский



AlphaGo 4:1



© Lay and You

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

23/03/2016, 20:32

© Try and You

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

24/03/2018, 08:59

утро

вечер

↑
original



Новая зима близко — ???



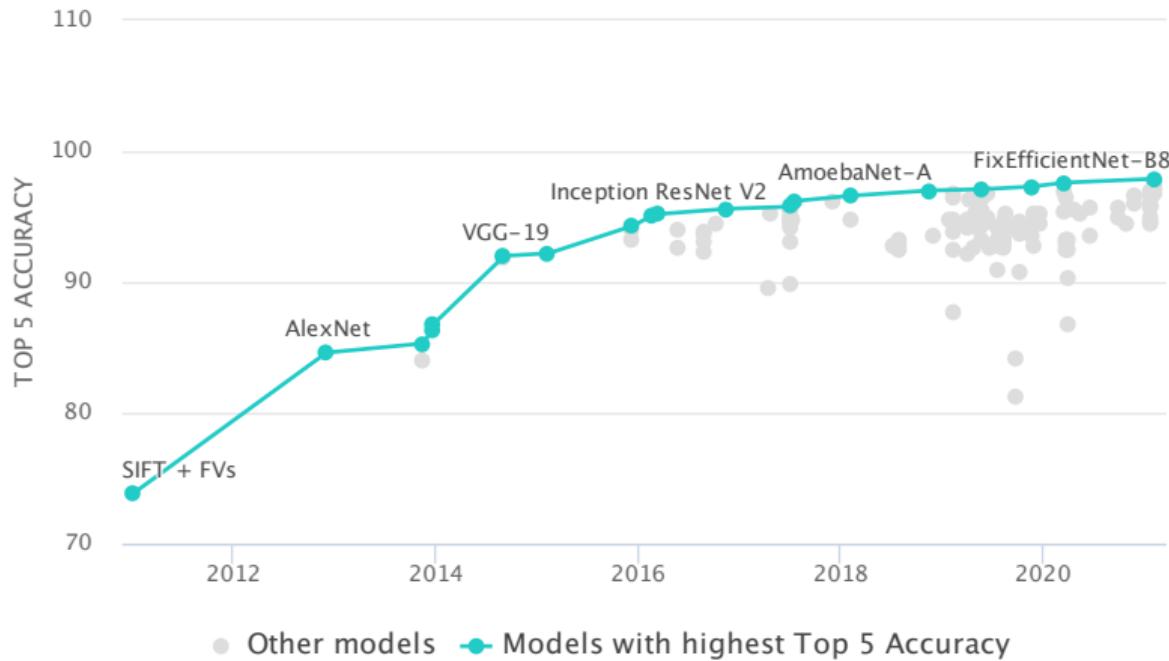
или нет ...

- 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

Важные тренды

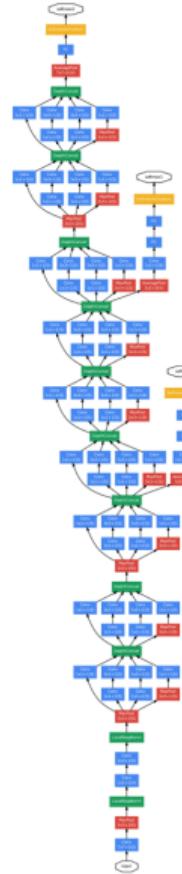
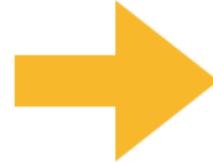
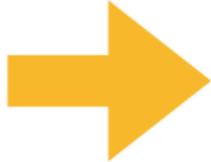
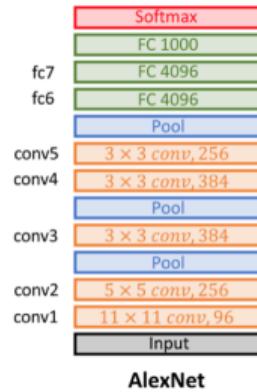


1. Точность сетей растёт



<https://paperswithcode.com/sota/image-classification-on-imagenet>

2. Сложность сетей растёт



3. Объёмы данных растут



4,855,710,815

Internet Users in the world



1,843,754,381

Total number of Websites



253,761,269,351

Emails sent **today**

14 марта 2021



7,336,754,080

Google searches **today**



7,069,408

Blog posts written **today**



762,038,383

Tweets sent **today**



7,196,539,758

Videos viewed **today**
on YouTube



85,848,513

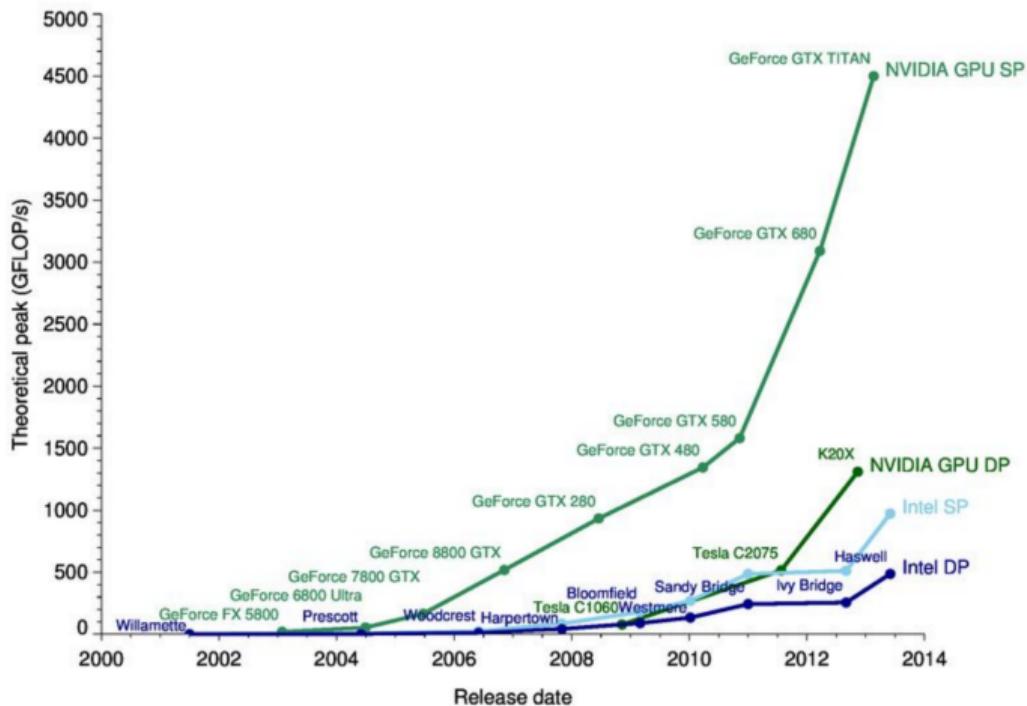
Photos uploaded **today**
on Instagram



150,422,117

Tumblr posts **today**

4. Вычислительные мощности растут



4. Почему это возможно?



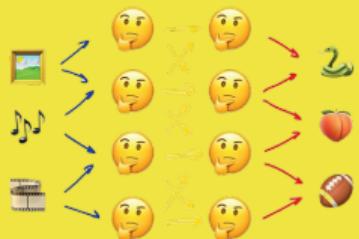
FINAL FANTASY XV
ファイナルファンタジー XV



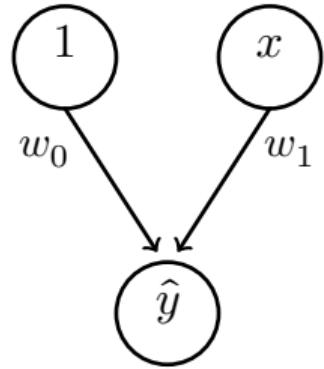
QUANTUM BREAK

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

От регрессии к нейросетке

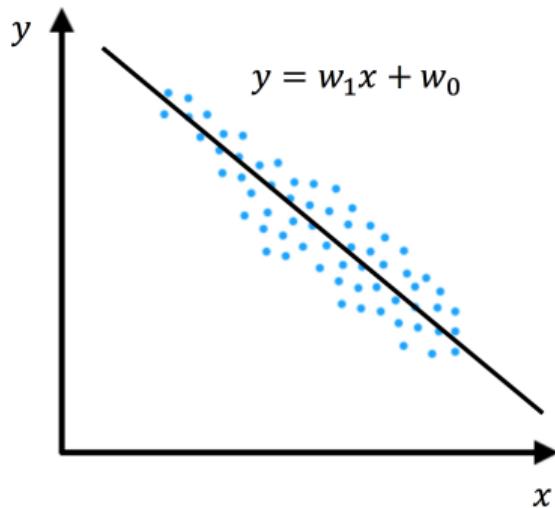


Линейная регрессия

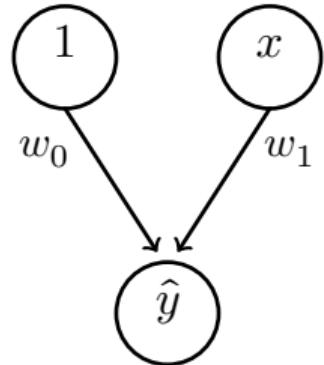


$$y_i = w_0 + w_1 \cdot x_i$$

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



Линейная регрессия



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

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

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

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

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

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

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:

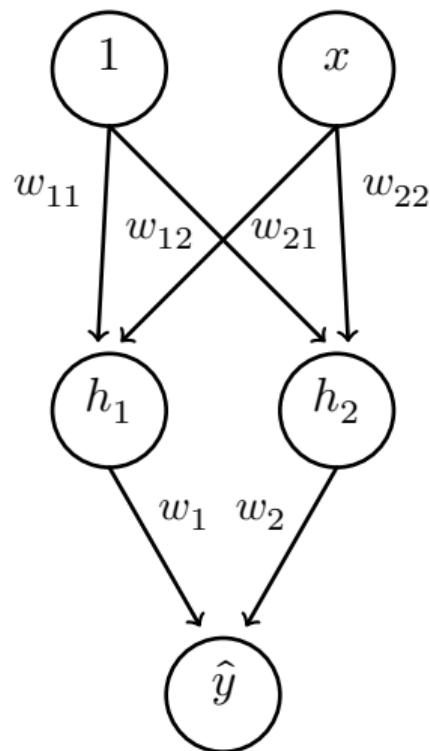
$$MAE(w) = \frac{1}{n} \sum_{i=1}^n |w^T x_i - y_i|$$

- 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...



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

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

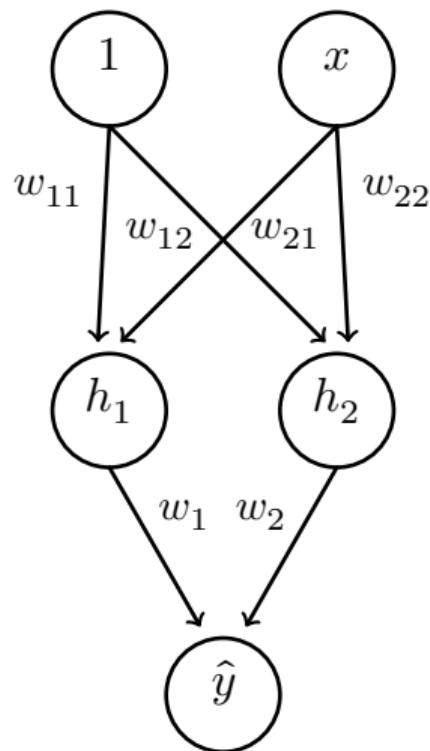
$$y_i = w_1 \cdot h_{1i} + w_2 \cdot h_{2i}$$

$$h = X \cdot W_1$$

$$y = h \cdot W_2$$

Норм идея?

А что если...

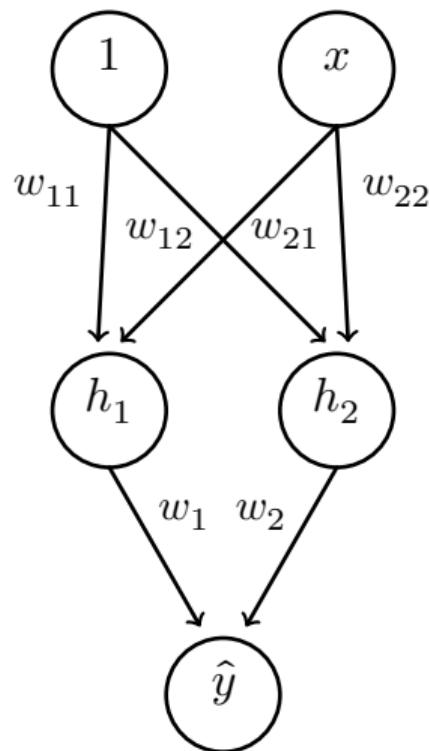


$$\begin{aligned} y &= w_1 \cdot h_1 + w_2 \cdot h_2 = \\ &= w_1 \cdot (w_{11} + w_{21} \cdot x) + w_2 \cdot (w_{12} + w_{22} \cdot x) = \\ &= \underbrace{(w_1 w_{11} + w_2 w_{12})}_{\gamma_1} + \underbrace{(w_1 w_{21} + w_2 w_{22})}_{\gamma_2} x \end{aligned}$$

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:

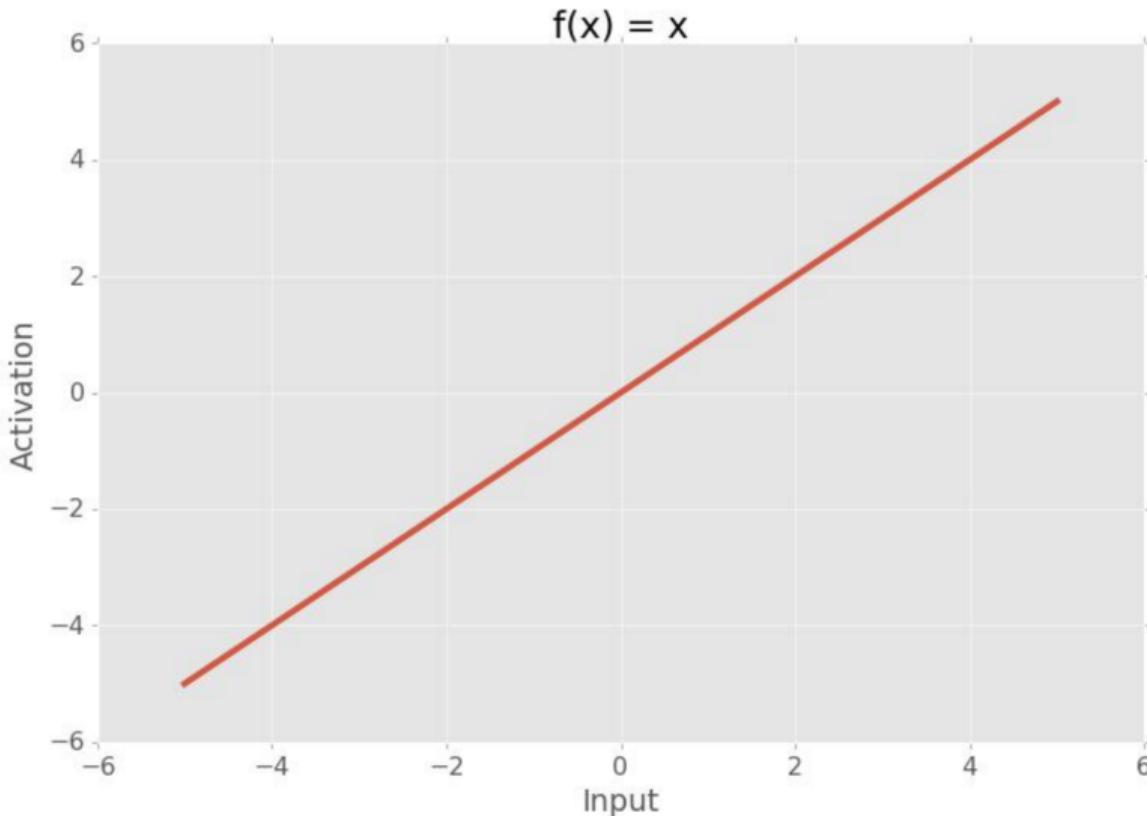
$$h_{1i} = w_{11} + w_{21} \cdot x_i$$

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

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

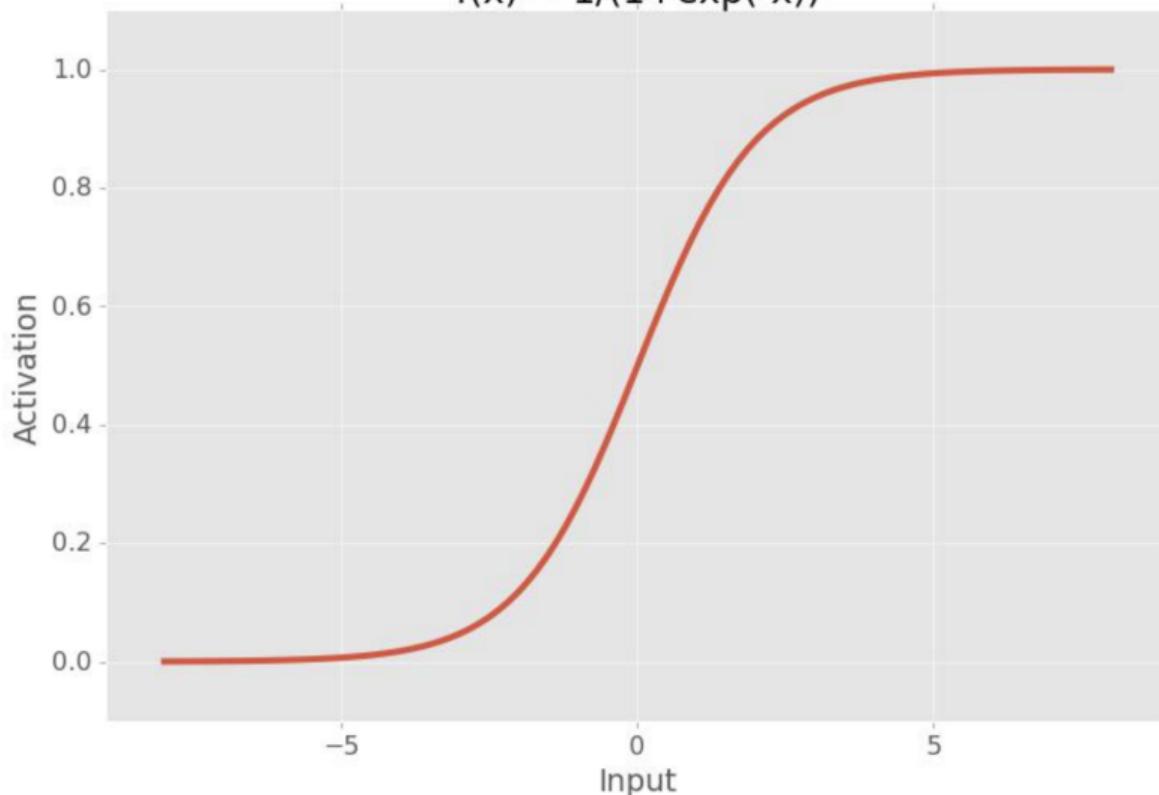
$$y = h \cdot W_2 = f(X \cdot W_1) \cdot W_2 \neq X \cdot A$$

Почему нельзя взять такую функцию?

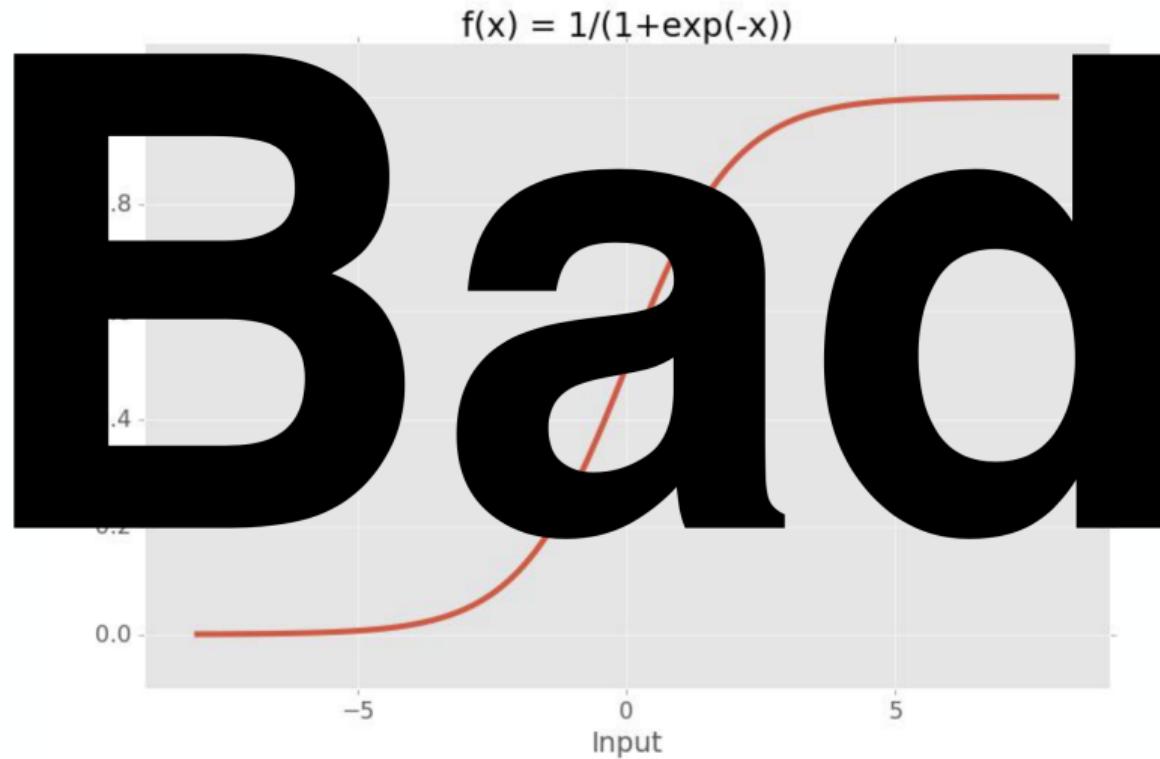


Сигмоида

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



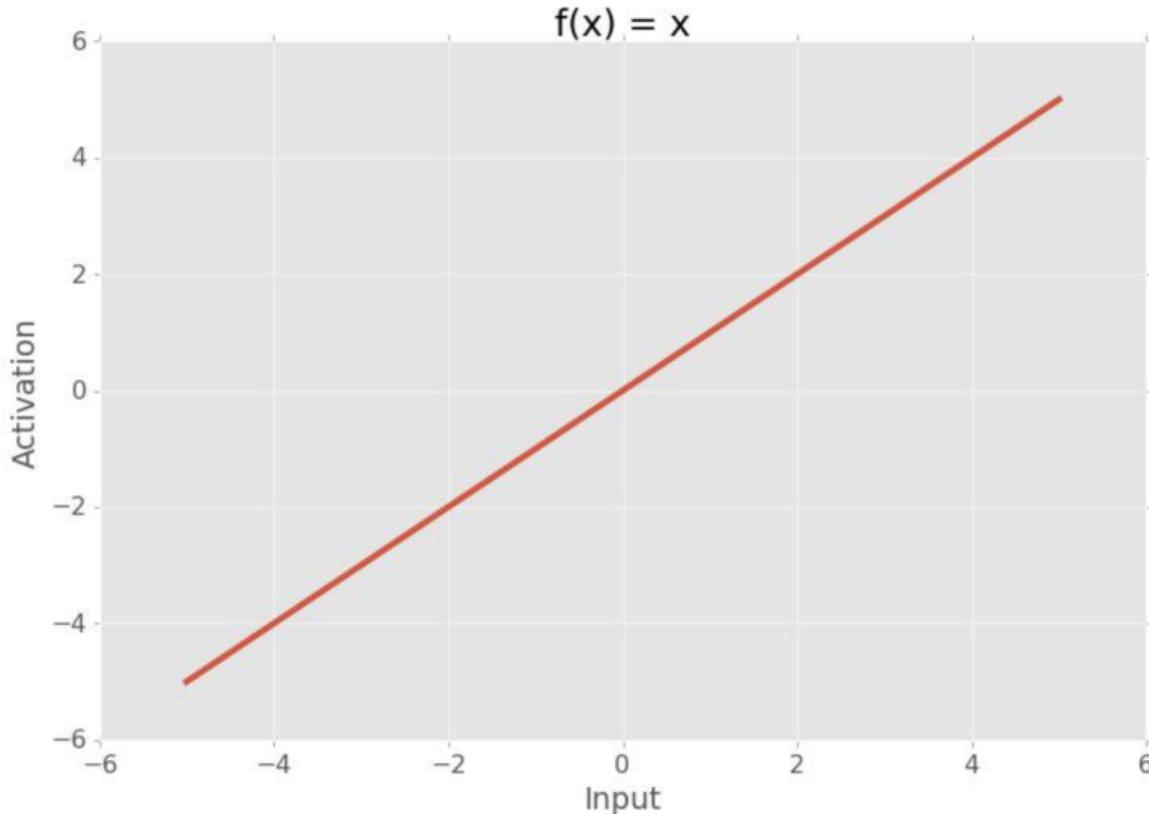
Сигмоида



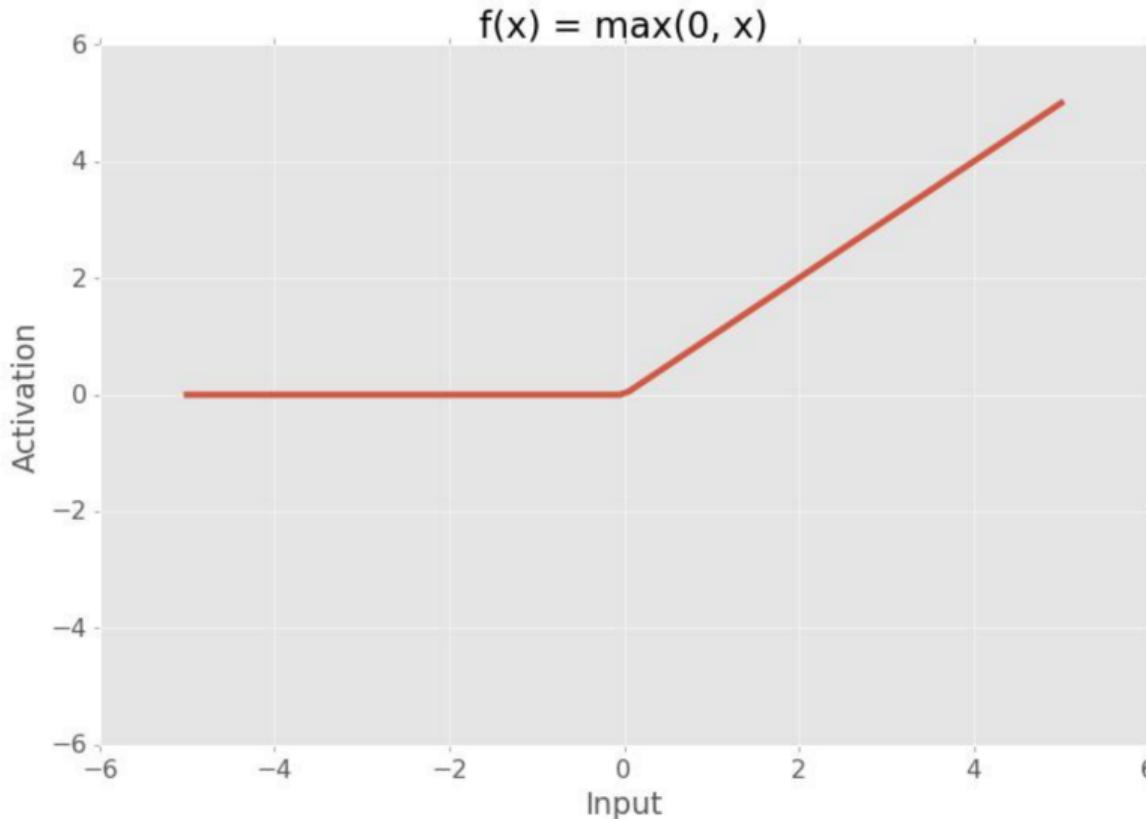
Сигмоида

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

От линейной активации ...

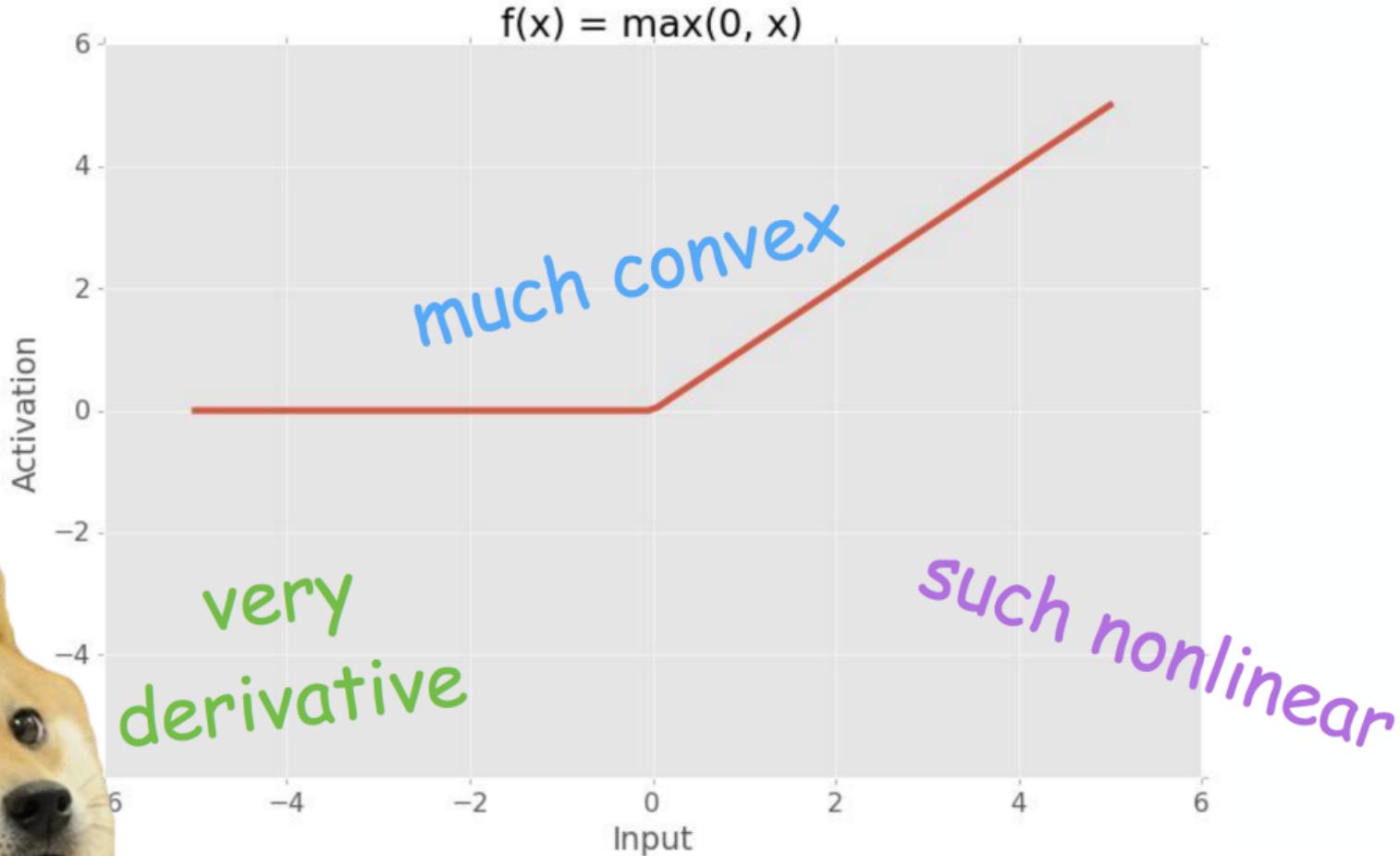


... к нелинейной



ReLU

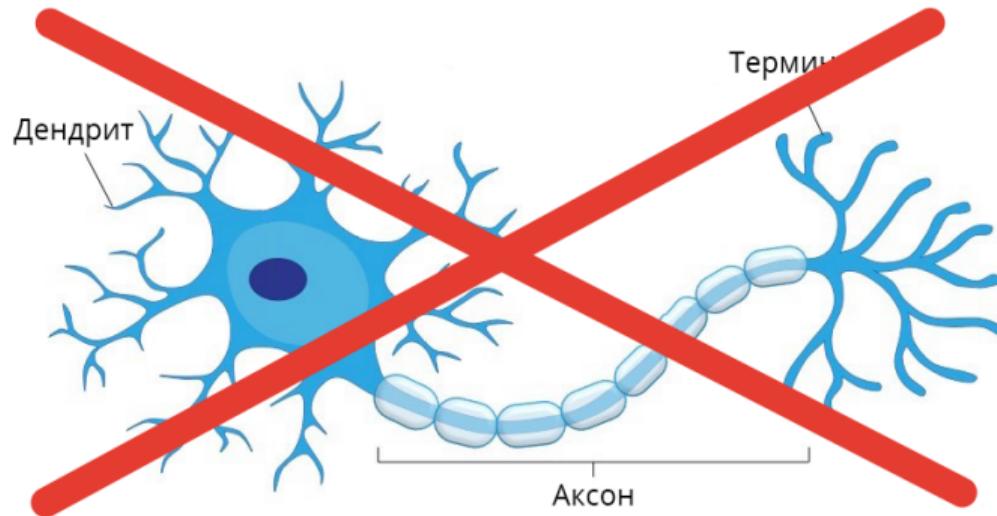
$$f(x) = \max(0, x)$$



Перцепtron



Нейрон

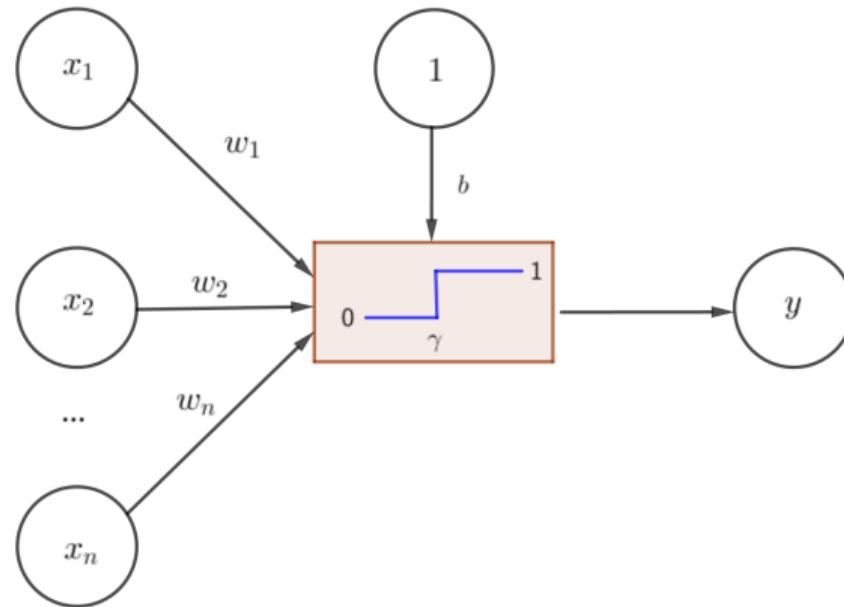


Хватит нам врать!

Нейрон

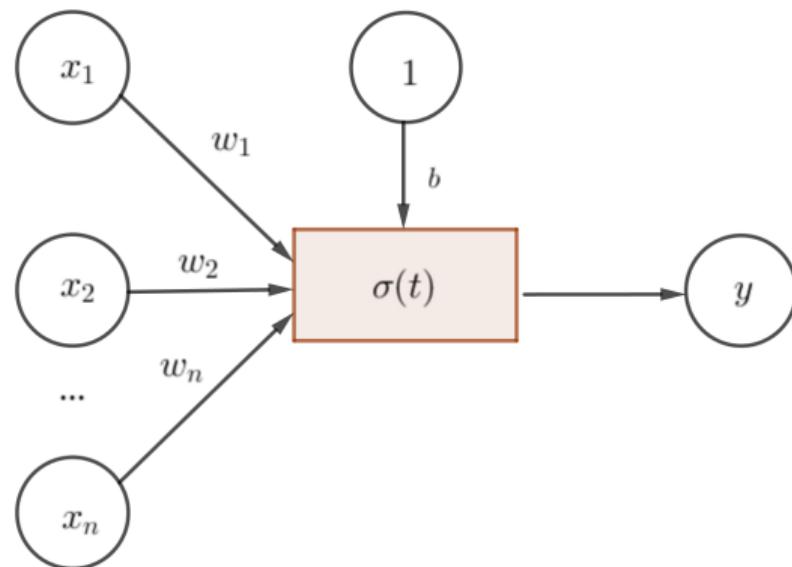
- Термин нейронная сеть пришёл из биологии, его придумали 70 лет назад
- Оказалось, что мозг устроен гораздо сложнее
- Продуктивнее думать про нейросетки как про вычислительные графы

Персептрон Розенблатта (1958)



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

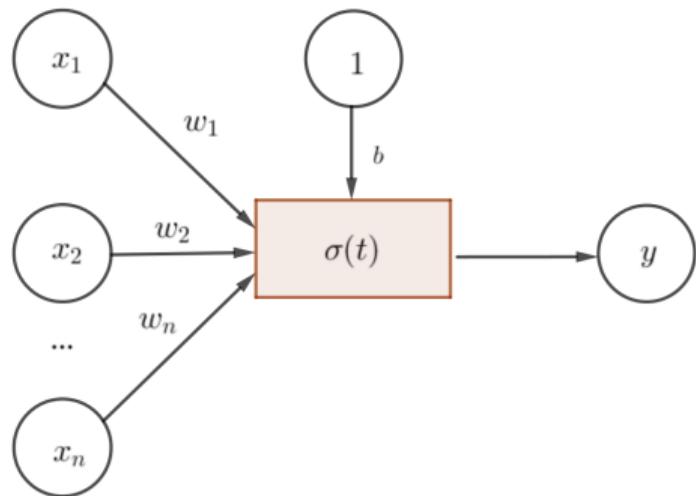
Функция активации



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

Линейная регрессия

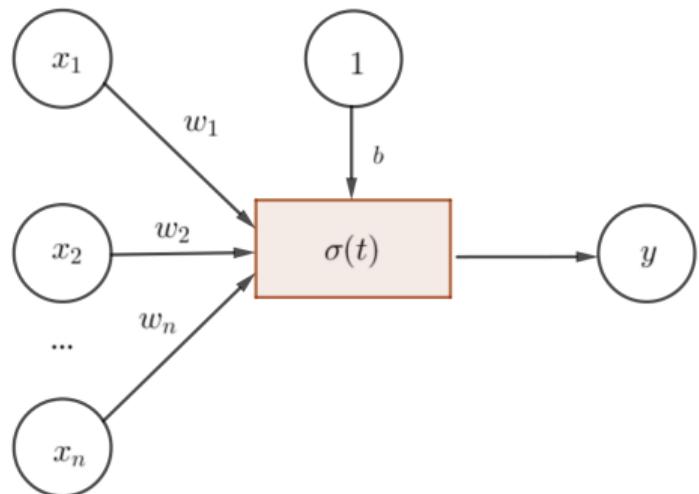
Нейрон с линейной функции активации — это линейная регрессия...



$$\sigma(t) = t$$

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

Логистическая регрессия

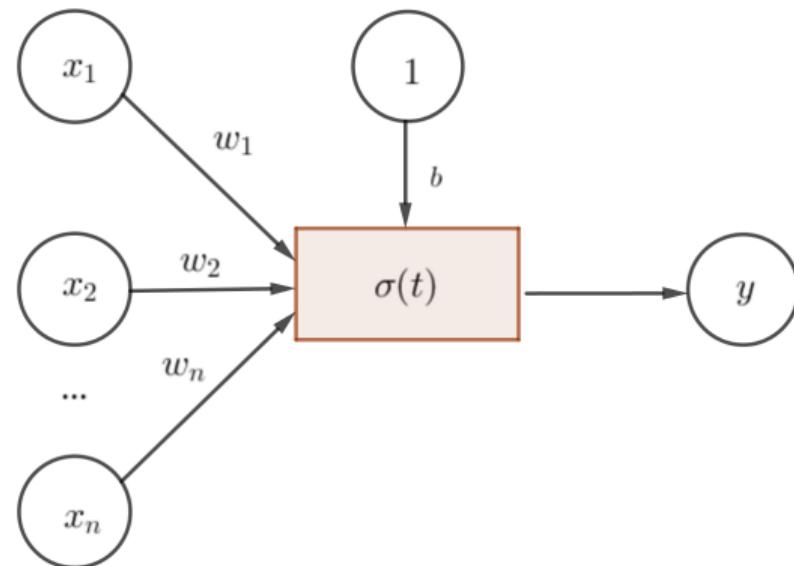


Нейрон с сигмоидом в качестве функции активации — это логистическая регрессия...

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

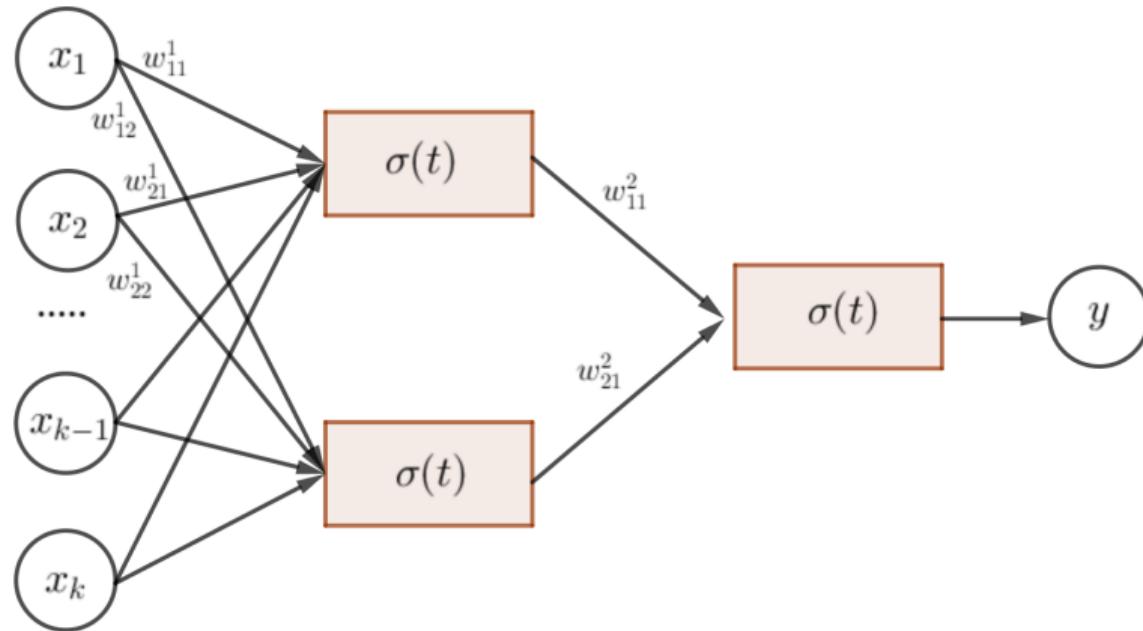
$$P(y = 1 | x) = \sigma(w_0 + w_1 \cdot x_1 + \dots + w_n \cdot x_n)$$

Матричный вид



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

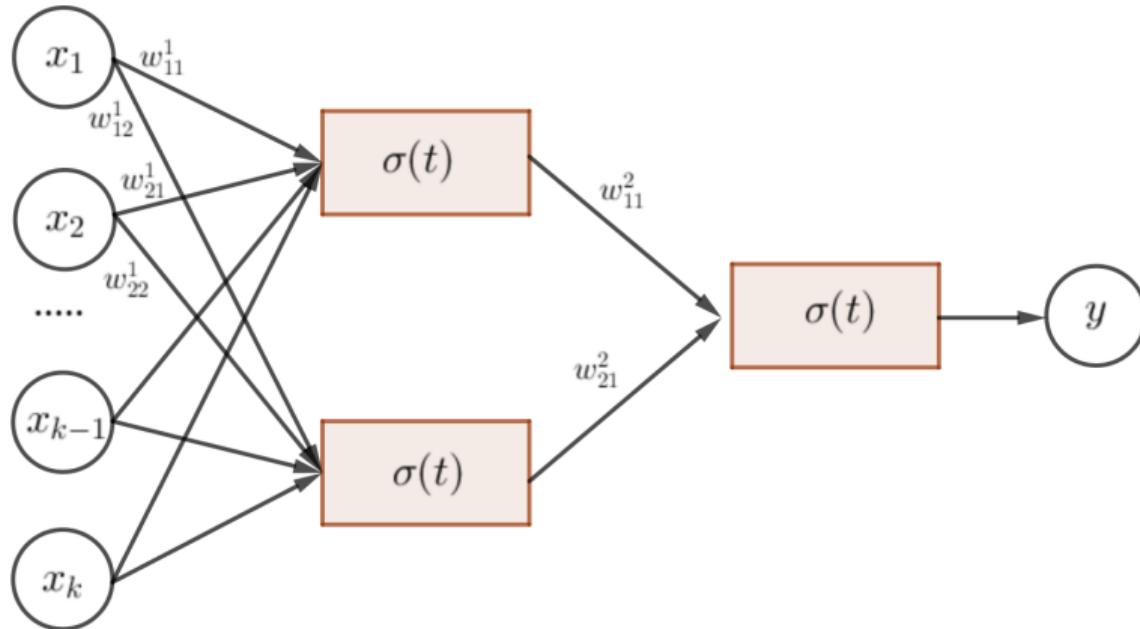
Две регрессии скрепили третьей



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

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

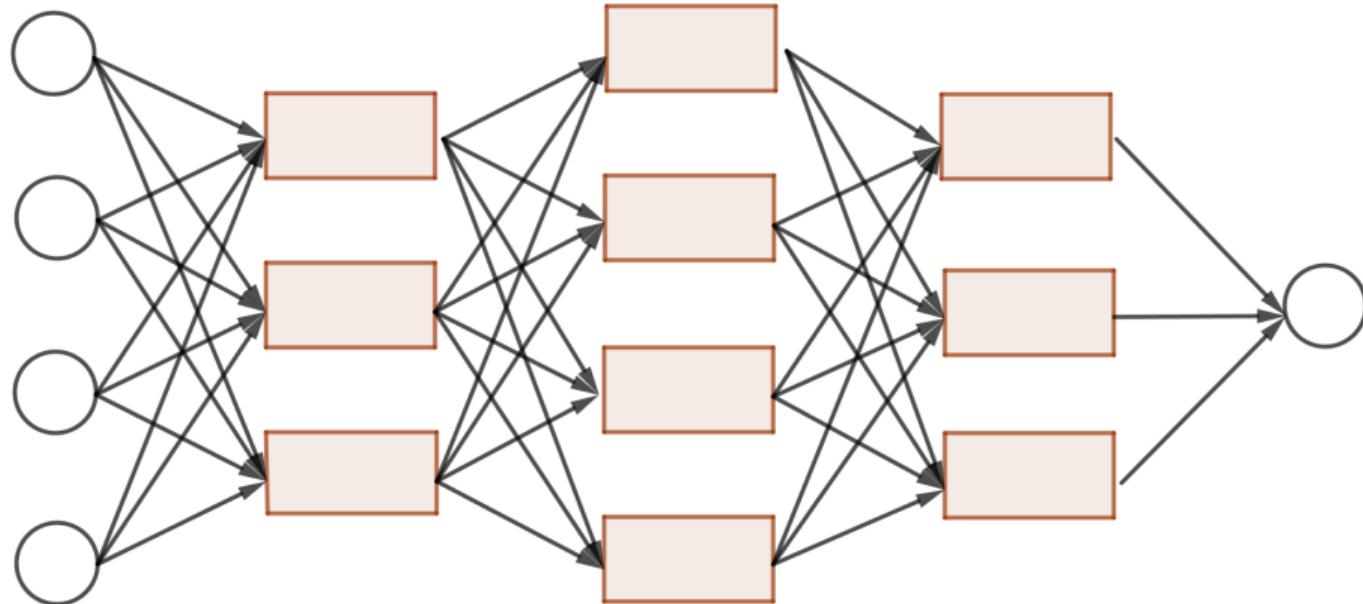
MLP (multi-layer perceptron)



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

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

Армия из регрессий



The Perceptron Convergence Theorem (Rosenblatt, 1965)

- Любая непрерывная и ограниченная функция может быть сколь угодно точно аппроксимирована нейронной сетью с одним скрытым слоем с нелинейной функцией активации нейрона.
- Любая функция может быть сколь угодно точно аппроксимирована нейронной сетью с двумя скрытыми слоями с нелинейной функцией активации нейрона.
- Что ещё можно пожелать?

Графическое доказательство теоремы:

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

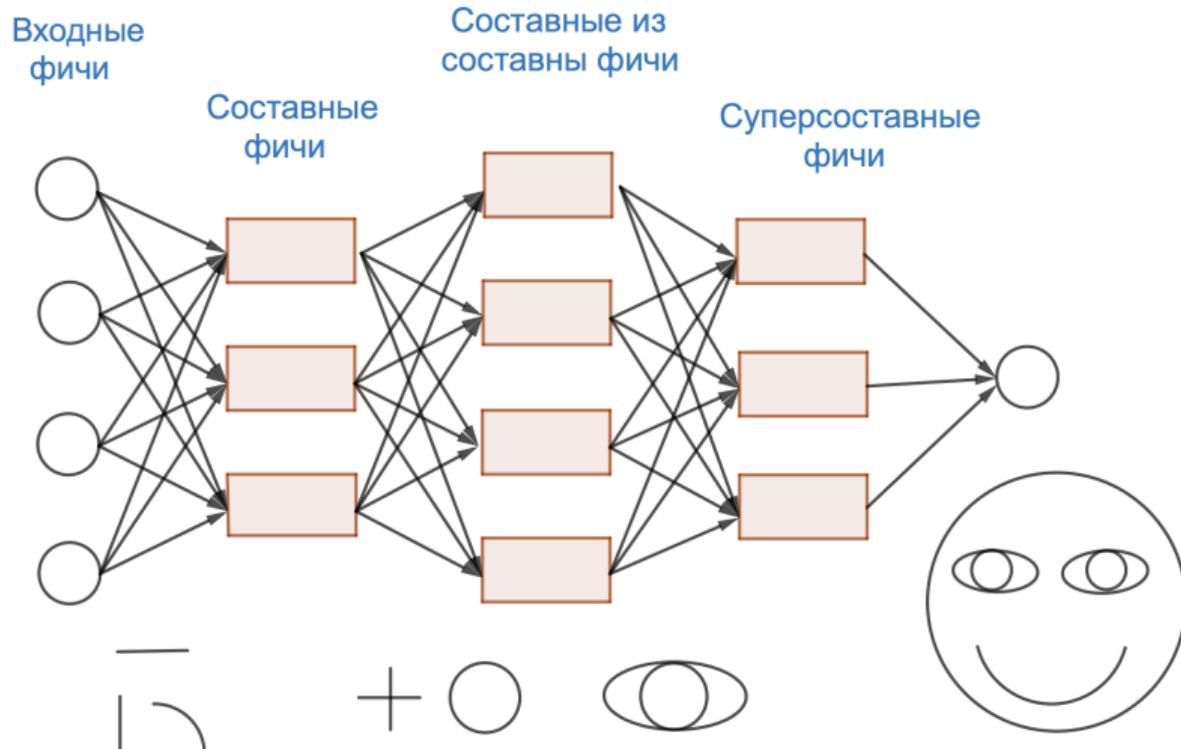
A photograph taken from underwater looking up at the surface. The water is a deep blue, and numerous bright, white sunbeams penetrate the surface, creating a starburst effect. The surface is slightly wavy.

Going Deeper

Мотивация

- Перцептрон 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



Ещё одна аналогия

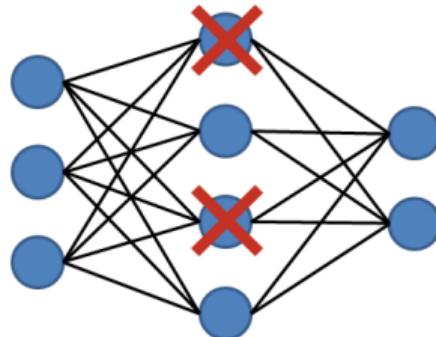


Нейросети — конструктор LEGO



Слои бывают разными

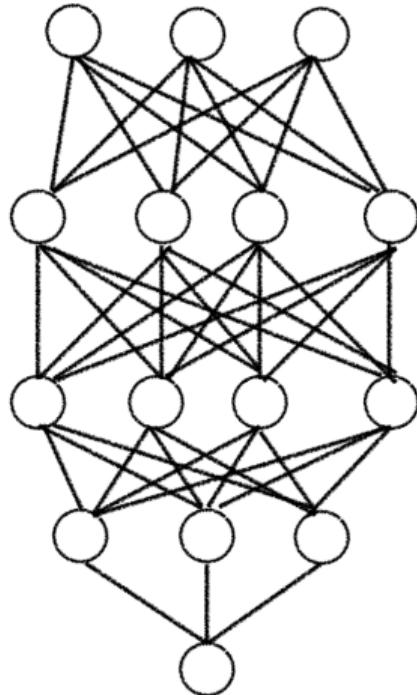
- Слой, который просто взвешивает входы называется **полносвязным**.
- Слои бывают очень разными. Например, **Dropout**: с вероятностью p отключаем нейрон. Такой слой препятствует переобучению и делает нейроны более устойчивыми к случайным возмущениям.



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	$\begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$	$\begin{cases} f(x) + \alpha, & x < 0 \\ 1, & x \geq 0 \end{cases}$

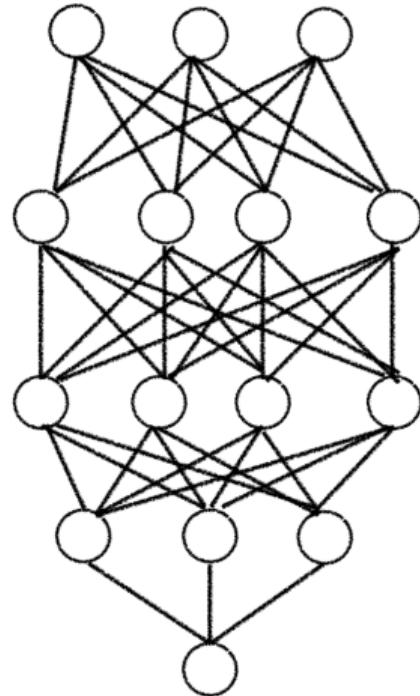
Архитектуры бывают разными



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	

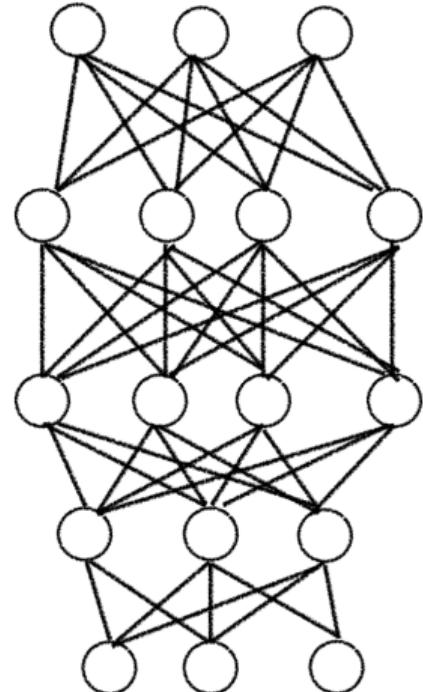
Каждый слой — просто функция, каждая сетка — конструктор LEGO

Персептра



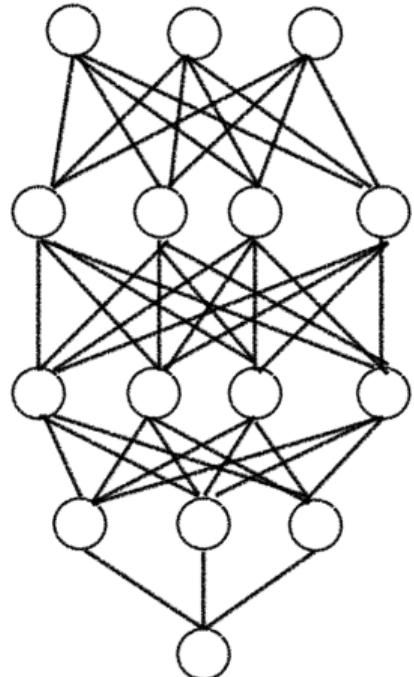
Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$
Output	

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



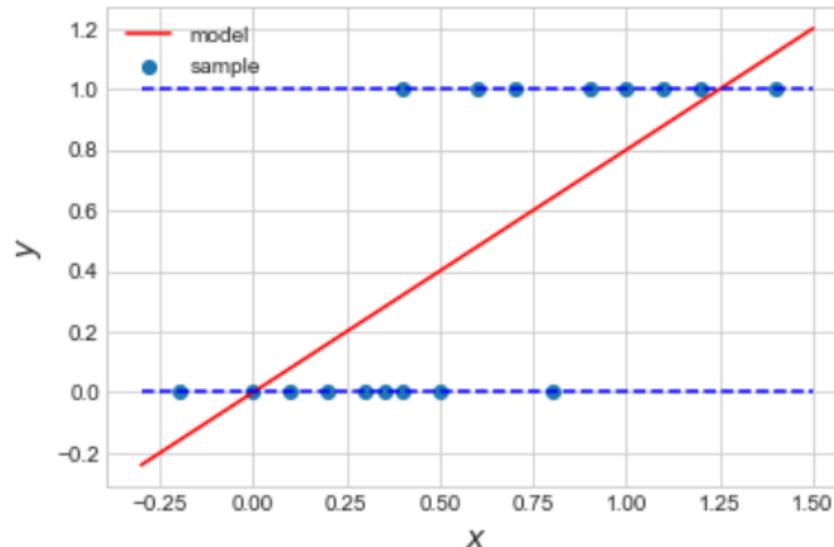
Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$
Sigmoid	$\sigma(x)$
Output	

Подробнее про классификацию

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

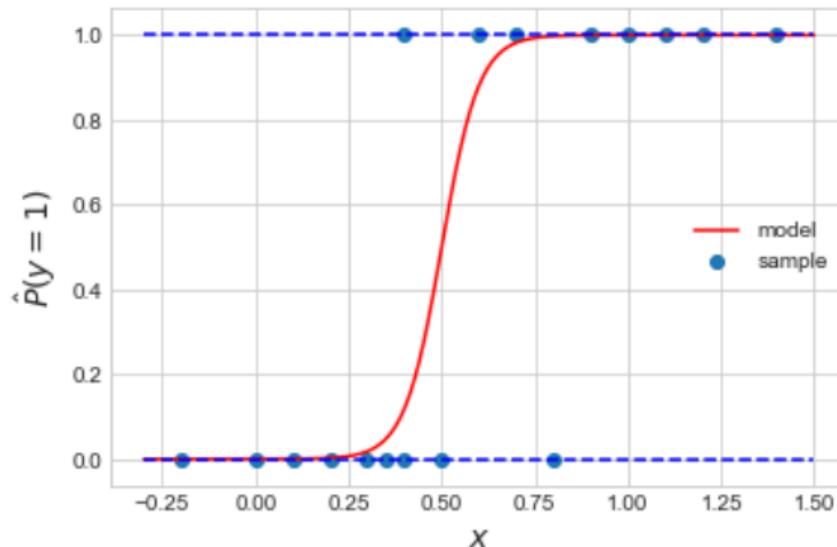
Подробнее про классификацию

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



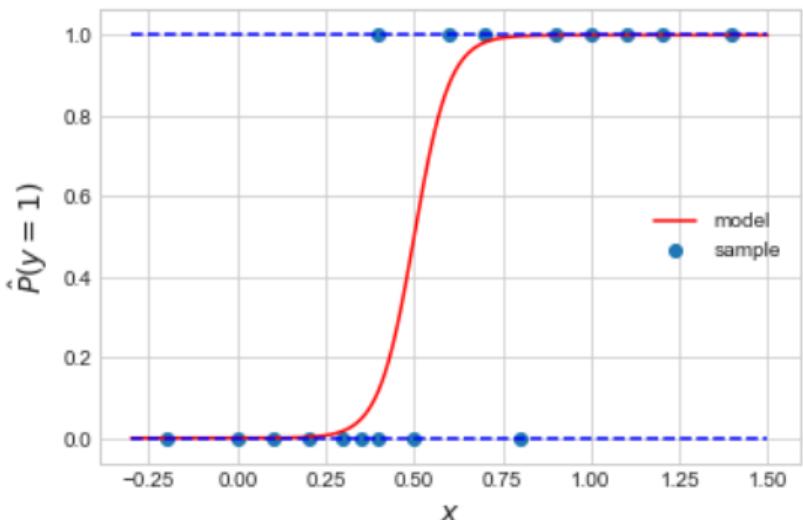
Подробнее про классификацию

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



Подробнее про классификацию

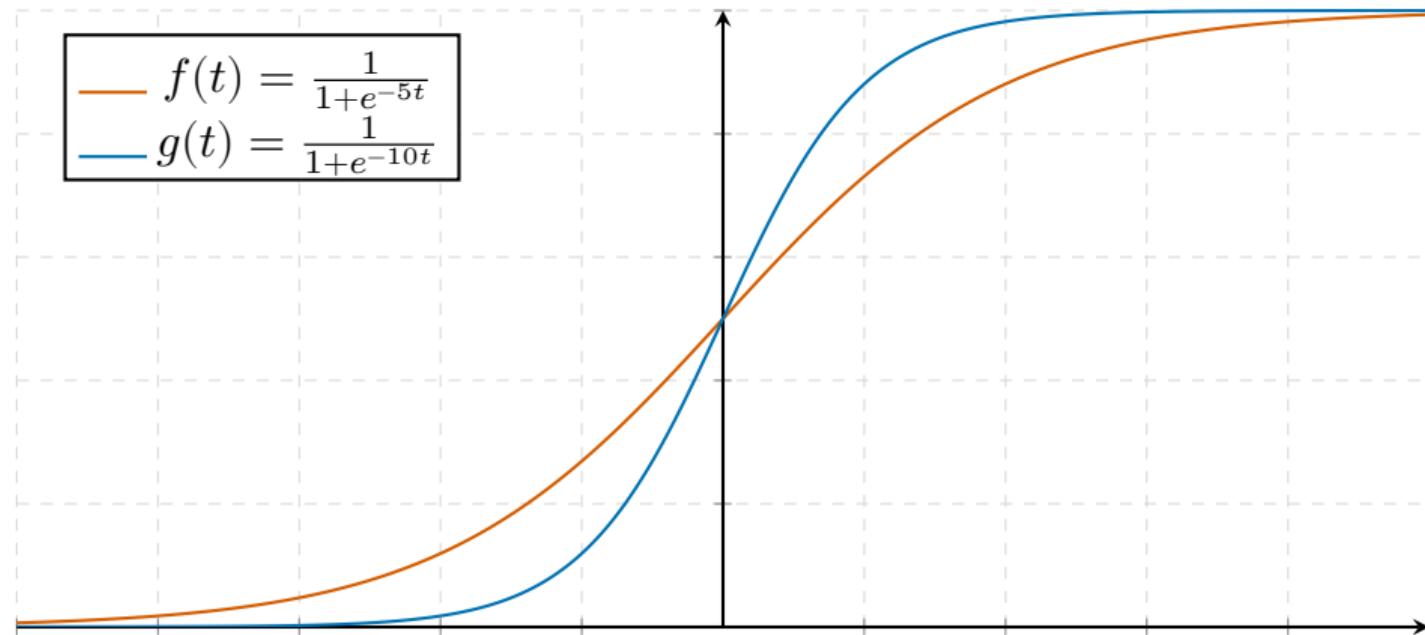
- $y \in \{0, 1\}$ — целевая переменная, X — признаки
- Модель: $P(y = 1 | w) = F(w_0 + w_1 x_1 + w_2 x_2 + \dots + w_k x_k)$



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

$$F(t) = \sigma(t) = \frac{1}{1 + e^{-t}} = \frac{e^t}{1 + e^t}$$
$$\sigma'(t) = \sigma(t) \cdot (1 - \sigma(t))$$

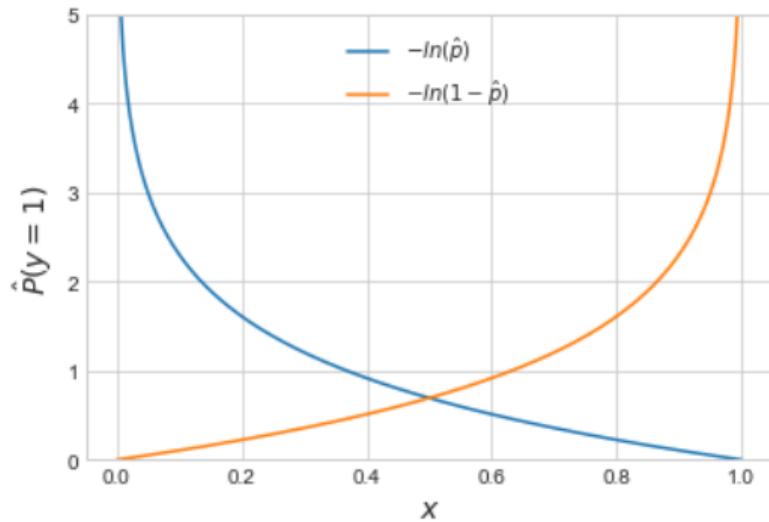
Сигмоида



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:

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



Пример: два класса



$$\begin{array}{c} t \qquad \sigma(t) \qquad \text{logloss} \\ \Rightarrow \qquad \qquad \qquad \end{array}$$

Input

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

$\Rightarrow 10 \Rightarrow 0.99 \Rightarrow - (1 \cdot \ln 0.99 + (1 - 1) \cdot \ln(1 - 0.99))$



$$\begin{array}{c} t \qquad \sigma(t) \qquad \text{logloss} \\ \Rightarrow \qquad \qquad \qquad \end{array}$$

Input

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

$\Rightarrow -1 \Rightarrow 0.26 \Rightarrow - (0 \cdot \ln 0.26 + (1 - 0) \cdot \ln(1 - 0.26))$

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

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

$\Rightarrow -\ln 0.99$



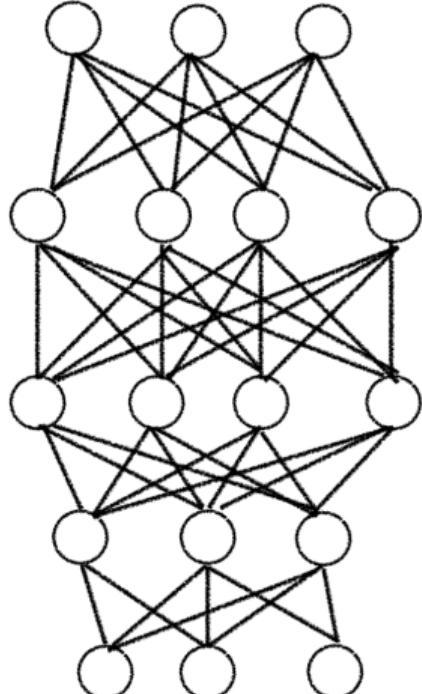
\Rightarrow

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

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

$\Rightarrow -\ln 0.74$

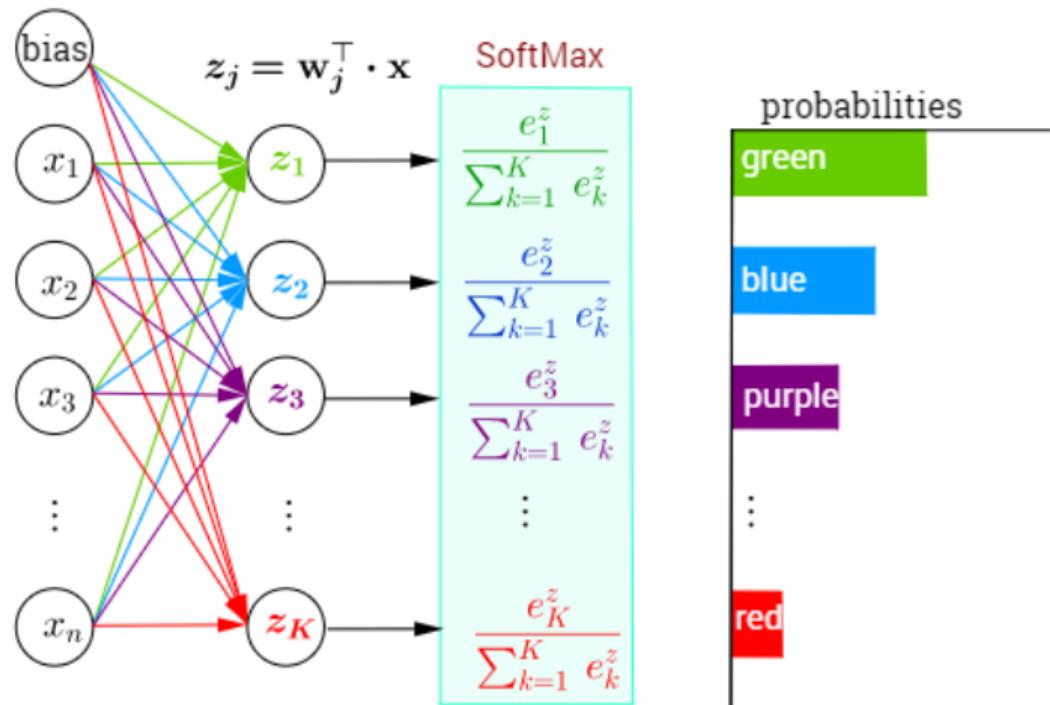
Мультиклассификация



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, $y \in 1, \dots, K$

$$\begin{aligned} & (w_1^T x, \dots, w_K^T x) \\ & \Downarrow \\ & (e^{z_1}, \dots, e^{z_K}) \\ & \Downarrow \\ & \left(\frac{e^{z_1}}{\sum_{k=1}^K e^{z_k}}, \dots, \frac{e^{z_K}}{\sum_{k=1}^K e^{z_k}} \right) \end{aligned}$$

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	$XW + b$

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



\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

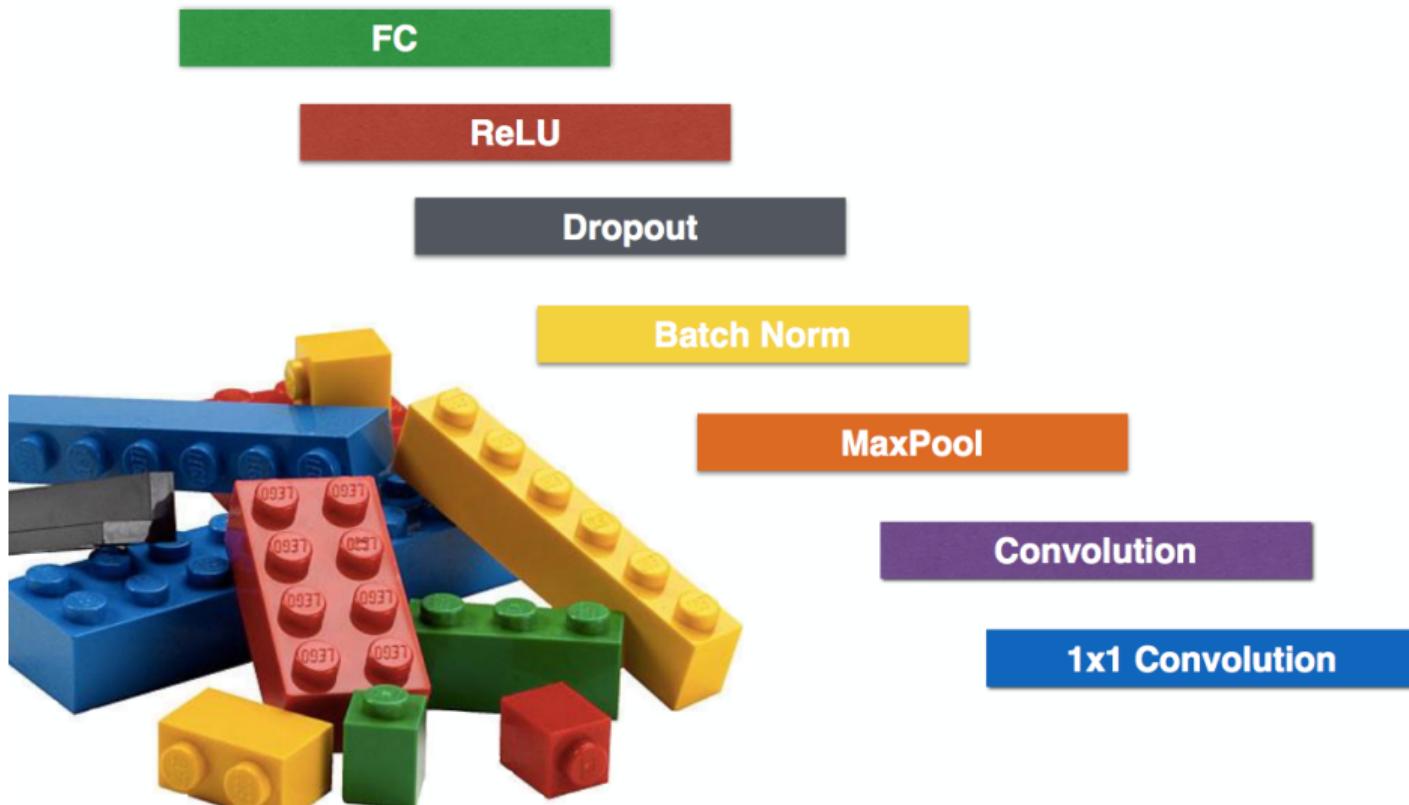
- При поиске Softmax мы ищем экспоненты, в памяти компьютера может произойти переполнение из-за больших чисел
- Если добавить ко всем входам нейронки одинаковую константу, значение Softmax не изменится:

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

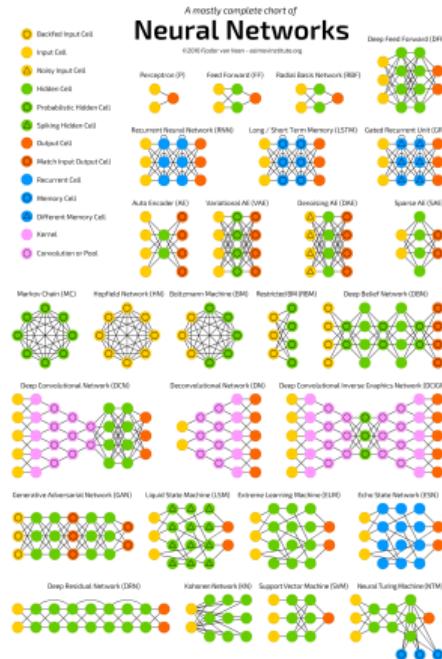
- Обычно считают устойчивый к переполнению Softmax:

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

Нейросети - конструктор LEGO



Не нейросеть, а граф вычислений



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

Запускаем свою первую нейросеть!