

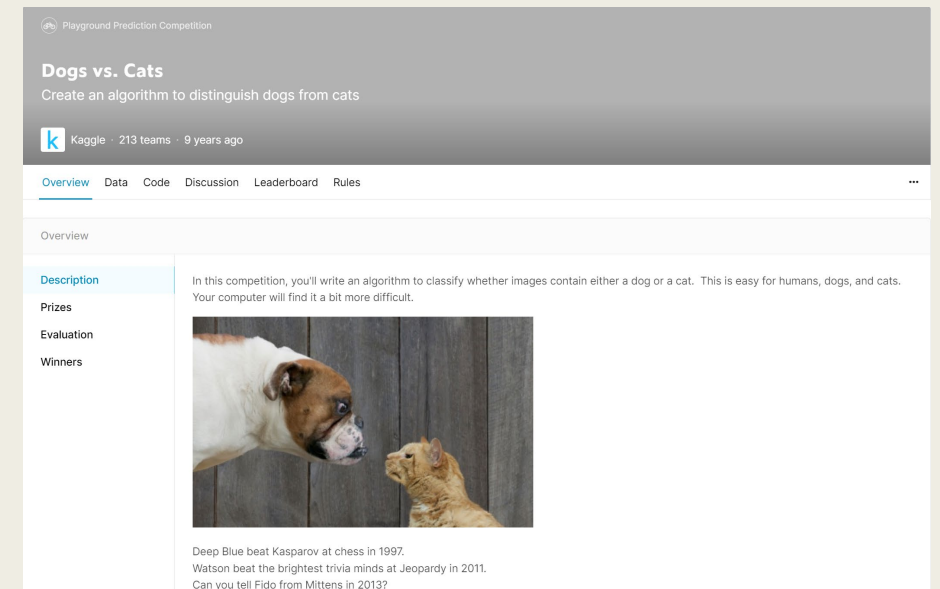


НЕЙРОННЫЕ СЕТИ

Что такое нейрон. Функция активации. Полносвязный слой.
MLP

Глубинное обучение

- Классическое машинное обучение: у нас есть данные, **признаки**, целевая переменная
- Но что делать, когда непонятно, какие признаки брать?
- Мы можем работать с текстом и картинками при помощи алгоритмов классического МО, но это неудобно и обычно не дает хороших результатов
- Какие признаки нужны, чтобы отличить кошку от собаки?



Глубинное обучение

- Классическое NLP. Мы можем генерировать текст с помощью цепей Маркова, вспомним NLTK:

long , from one to the top - mast , and no coffin and went out a sea captain -- this peaking of the whales . , so as to preserve all his might had in former years abounding with them , they toil with their lances , strange tales of Southern whaling . at once the bravest Indians he was , after in vain strove to pierce the profundity . ? then ?" a levelled flame of pale , And give no chance , watch him ; though the line , it is to be gainsaid . have been 'long , from one to the top - mast , and no coffin and went out a sea\ncaptain -- this peaking of the whales . , so as to preserve all his\nmight had in former years abounding with them , they toil with their\nlances , strange tales of Southern whaling . at once the bravest\nIndians he was , after in vain strove to pierce the profundity . ?\nthen ?" a levelled flame of pale , And give no chance , watch him ;\nthough the line , it is to be gainsaid . have been'

Глубинное обучение

- Современное NLP. Мы можем обучить модель на огромном количестве данных, вспомним [GPT](#):

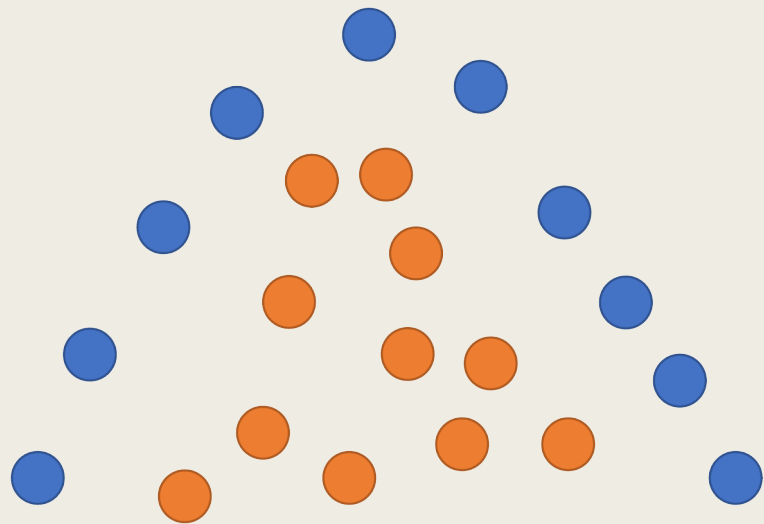
Данная работа представляет собой исследование на материале нескольких периодов. В центре внимания автора находится проблема изменения отношения между структурой и функциями синтаксических конструкций в языках, в том числе на материале русского и китайского. Автор проводит анализ семантических и морфологических параметров синтаксических конструкций в русском языке*.

* сгенерировано на материале аннотаций к дипломам ;)

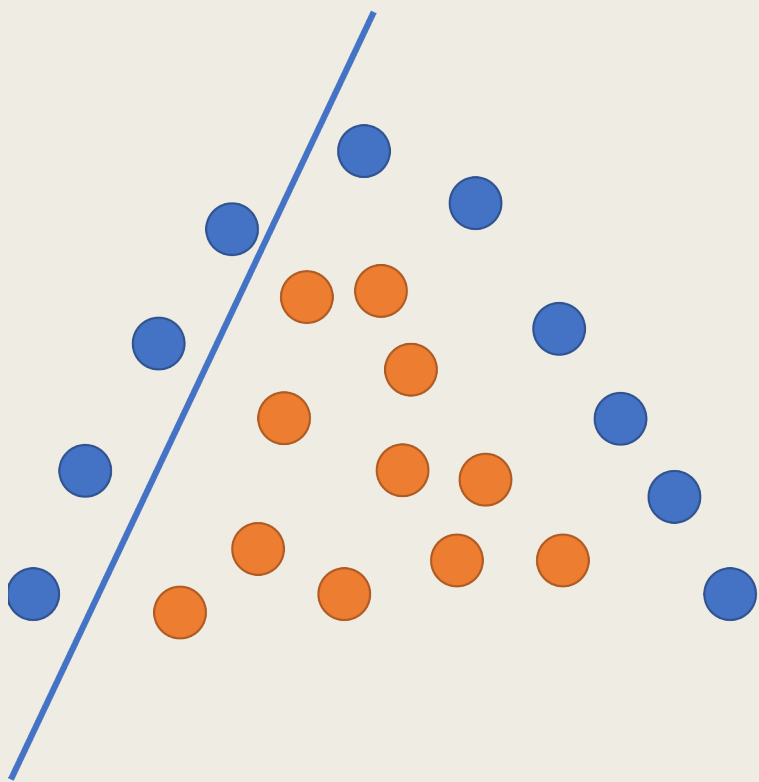
Преимущества и недостатки НС

- + они более качественные
- + они умеют сами извлекать признаки из данных
- + они умеют работать со сложными данными
- + они могут работать с нелинейными зависимостями
- они тяжеловесные и долго работают
- они не интерпретируемые: они не просто устанавливают признаки сами какие хотят, но и не сообщают нам об этом

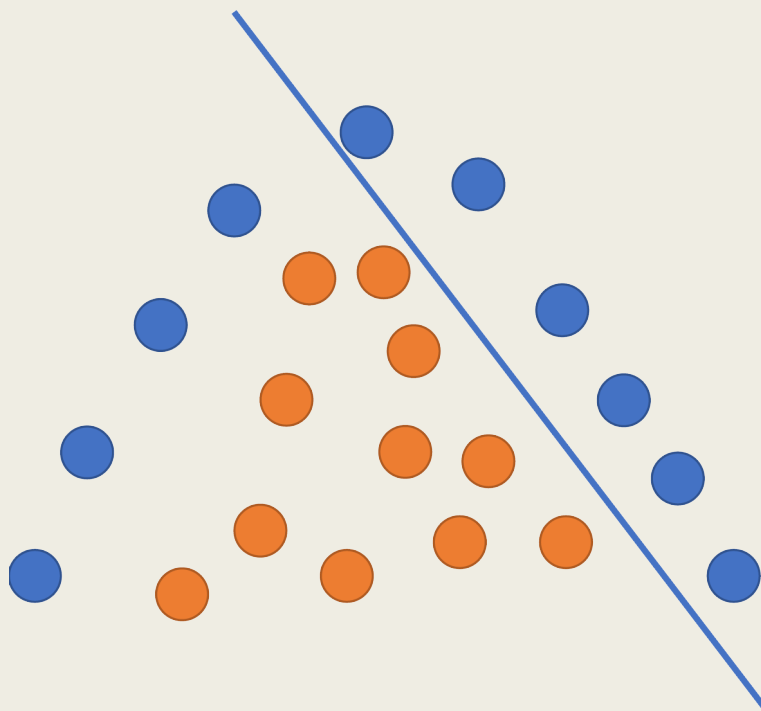
Нелинейные закономерности



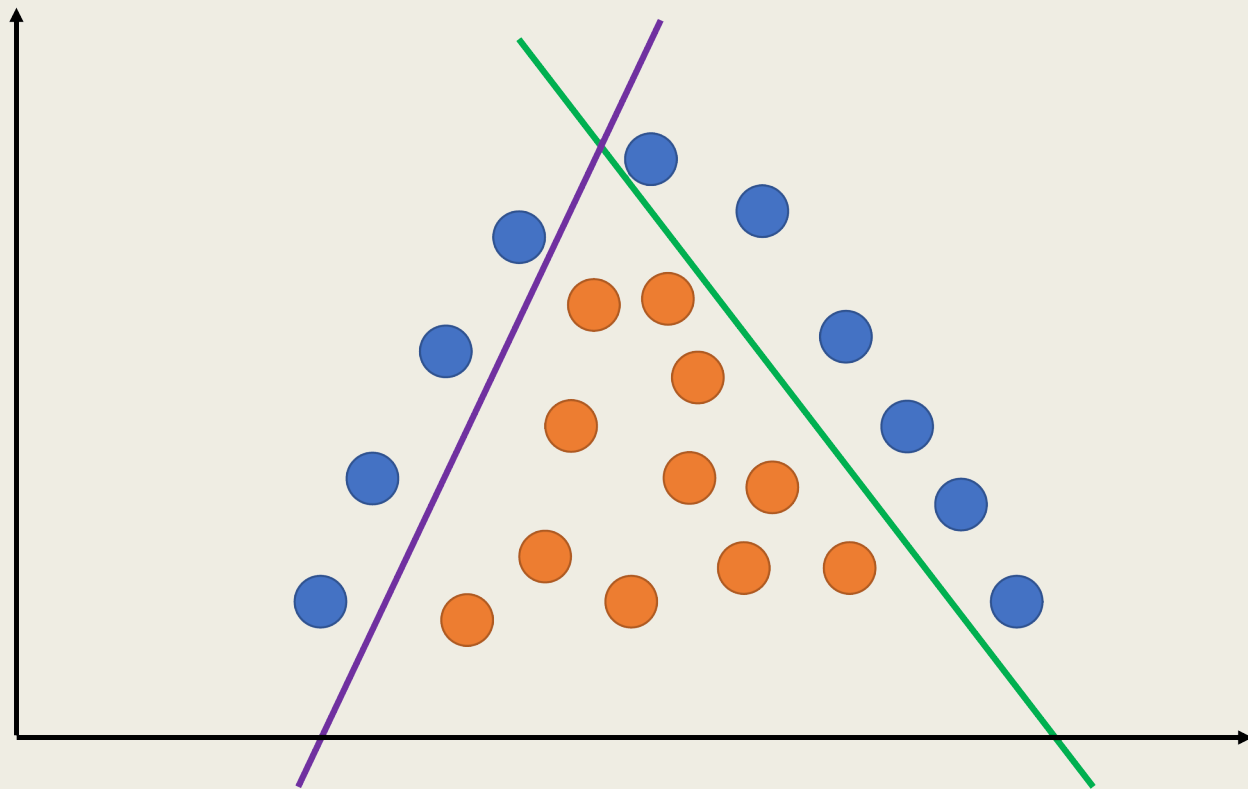
Нелинейные закономерности



Нелинейные закономерности



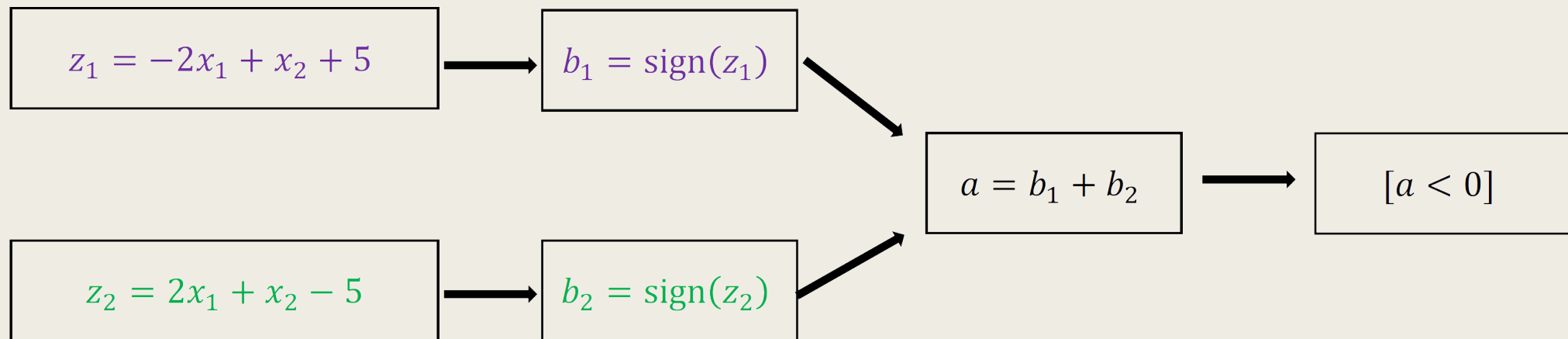
Нелинейные закономерности



$$b_1(x) = \text{sign}(-2x_1 + x_2 + 5)$$

$$b_2(x) = \text{sign}(2x_1 + x_2 - 5)$$

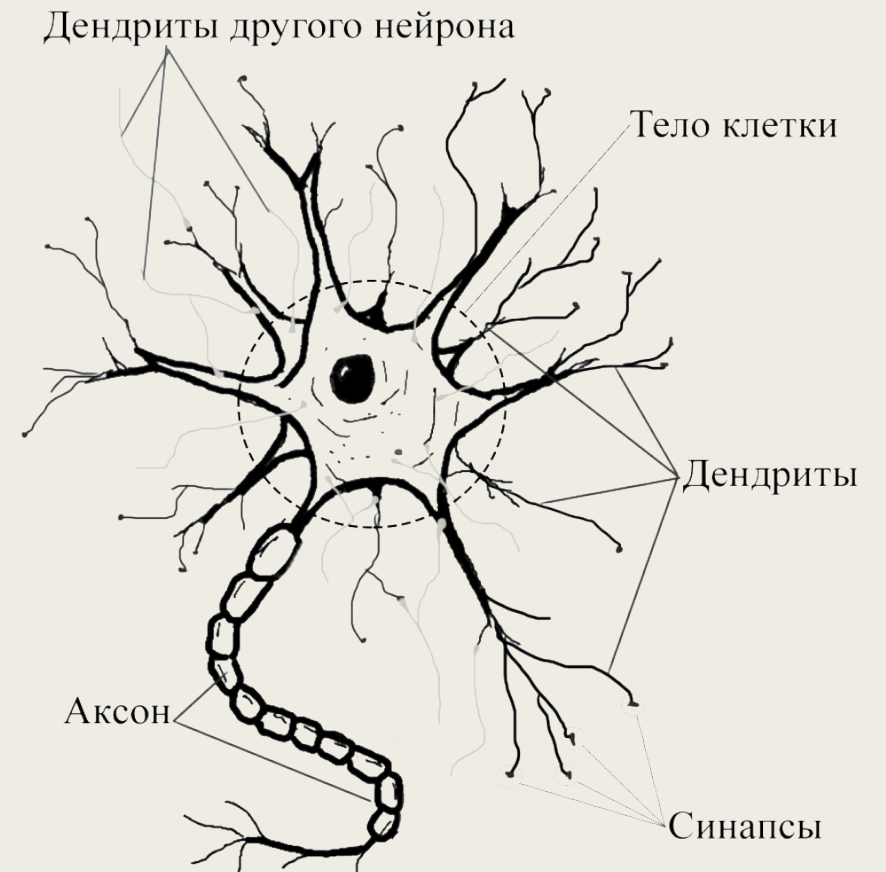
Нелинейные закономерности



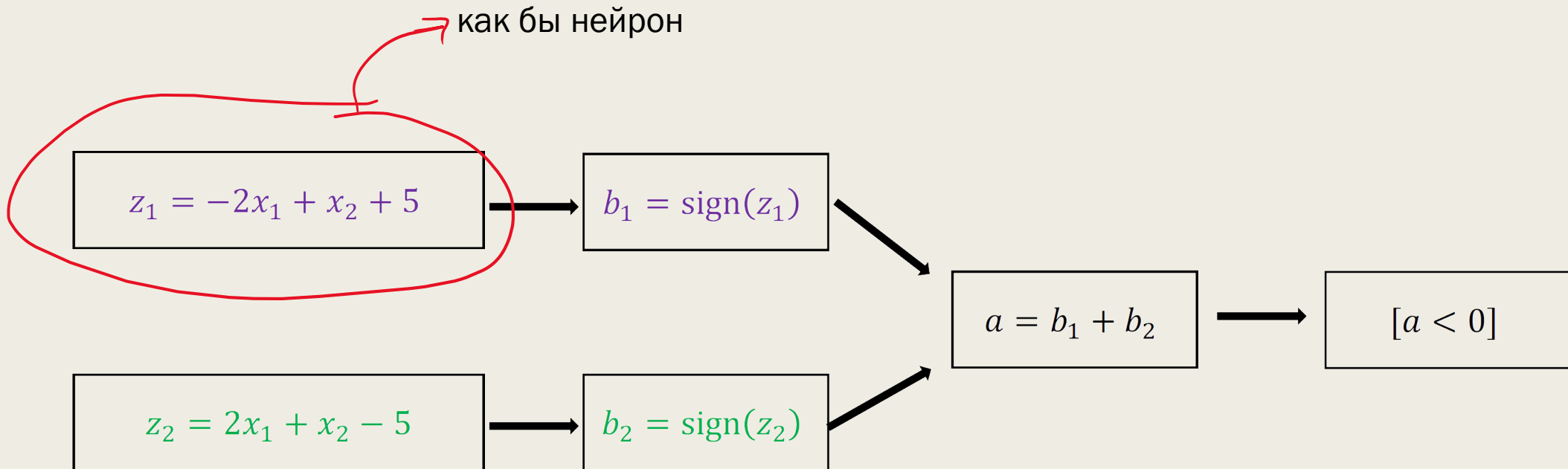
Идея нейрона

- Модель МакКаллока-Питтса
- Дендриты другого нейрона подают сигналы
- Синапсы их усиливают или ослабляют
- В ядре (теле клетки) эти сигналы складываются
- Аксон перерабатывает их и посылает дальше

$$\sum w_j x_j, \text{ ты?}$$

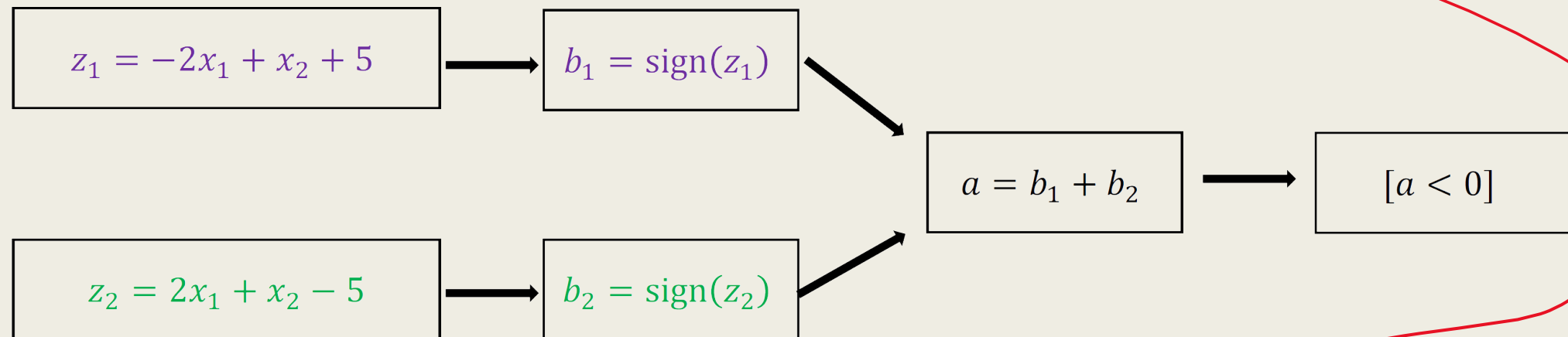


Идея нейрона



Идея нейрона

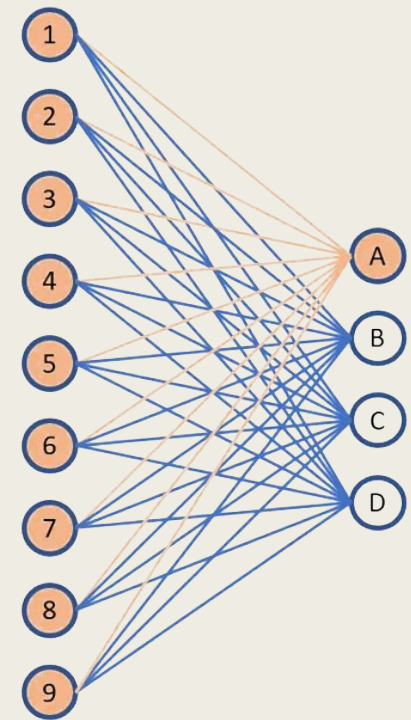
как бы нейронная сеть



Полносвязный слой (Fully connected layer)

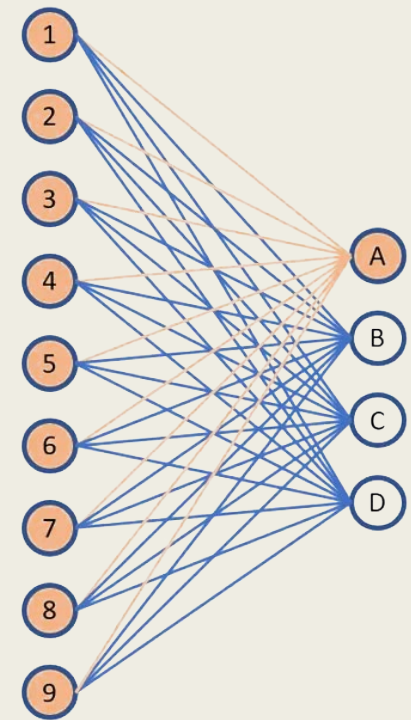
- На входе n чисел
- На выходе m чисел
- Каждый выход – линейная модель над входами

$$z_j = \sum_{i=1}^n w_{ji} x_i + b_i$$



Полносвязный слой (Fully connected layer)

- t линейных моделей, в каждой $n + 1$ параметров
- Итого в одном слое немного больше tn параметров
- Это очень много!
- Нужно много данных для обучения

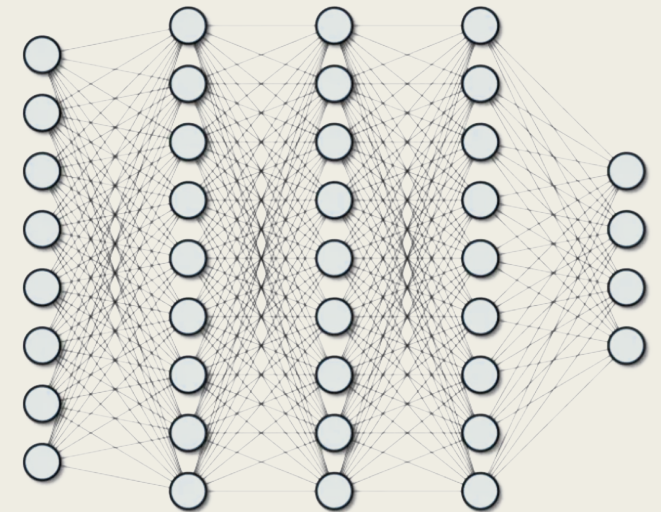


Нелинейность

- Как объединить слои в модель?
- Рассмотрим два полносвязных слоя:

$$s_k = \sum_{j=1}^m v_{kj} z_j + c_k = \sum_{j=1}^m v_{kj} \sum_{i=1}^n w_{ji} x_i + \sum_{j=1}^m v_{kj} b_j + c_k$$

- z_j – наши выходы первого слоя
- v_{kj} – веса второго слоя, c_k – его свободные коэффициенты
- s_k – выходы второго слоя
- Можно подставить формулу для первого слоя

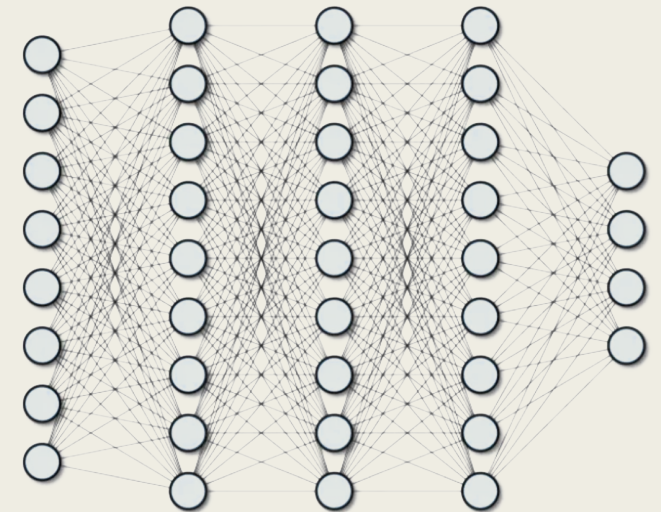


Нелинейность

- Как объединить слои в модель?
- Рассмотрим два полносвязных слоя:

$$\begin{aligned} s_k &= \sum_{j=1}^m v_{kj} z_j + c_k = \sum_{j=1}^m v_{kj} \sum_{i=1}^n w_{ji} x_i + \sum_{j=1}^m v_{kj} b_j + c_k = \\ &= \sum_{j=1}^m \left(\sum_{i=1}^n v_{kj} w_{ji} x_i + v_{kj} b_j + \frac{1}{m} c_k \right) \end{aligned}$$

- Получается, два полносвязных слоя ничем не лучше одного

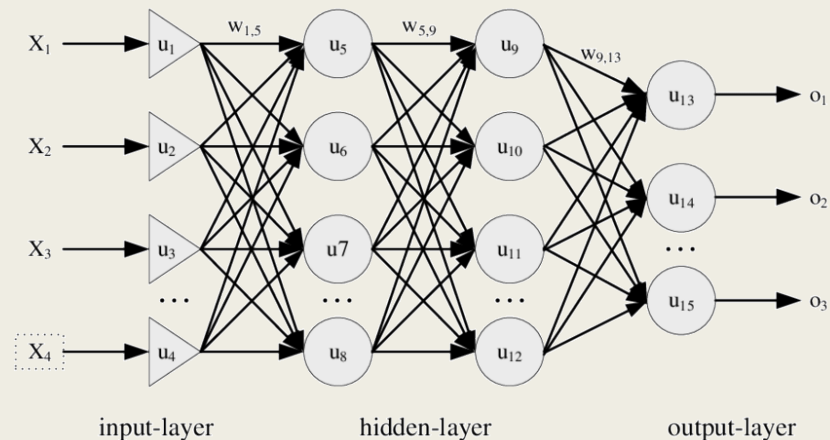


Нелинейность

- Вариант: добавлять какую-нибудь нелинейную функцию после каждого слоя (и применять ее ко всем выходам всех нейронов):



- Такая конструкция называется **многослойный перцептрон**
- Побаловаться можно онлайн [тут](#)



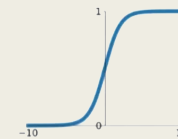
Нелинейность

- Какие функции использовать?
- Например, сигмоиду или ReLU: у всех есть свои плюсы и минусы
- Тысячи их!

Activation Functions

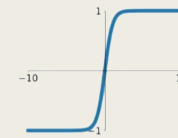
Sigmoid

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



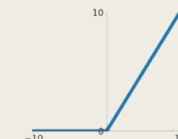
tanh

$$\tanh(x)$$



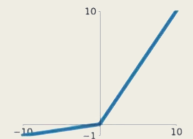
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

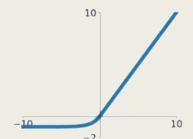


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

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



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?

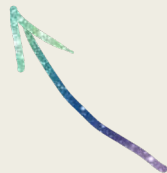
Графическое доказательство теоремы:

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

ПОДХОДЫ К ВЫЧИСЛЕНИЯМ

■ Императивный подход

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



сразу вычислили

■ 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)
```



сначала задали граф вычислений,
а потом уже вычислили

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
- Реализуем нейронные сети как графы вычислений

Фреймворки

theano



Монреальский
университет (2007)

Static Computational
Graph



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

Static Dynamic
Computational Graph

PYTORCH



Facebook (2016)

Dynamic Computational
Graph

Сначала обёртка для Theano, потом
для Tensorflow, сейчас фактически
часть Tensorflow



[почему мы выбираем
торч](#)