

Лекция 1. Введение

Глубинное обучение

Антон Кленицкий

О курсе

Пройдем путь от самых основ до современных моделей глубокого обучения:

- Что такое нейросети и как их обучать
- Основные архитектуры - сверточные, рекуррентные, трансформеры
- Приложения - Computer Vision, NLP
- Генеративные модели

Организационные вопросы

- Материалы курса:
<http://github.com/fintech-dl-hse/course>
- Формула оценки: $\sum_i O_{hw_i}$
- Telegram чат
- Форма обратной связи
- Задавайте вопросы!

Литература

- *Глубокое обучение. Погружение в мир нейронных сетей.*
С. Николенко, А. Кадурин, Е. Архангельская.
- *Глубокое обучение.* Я. Гудфеллоу, Й. Бенджио, А. Курвиль.
Английская версия:
<http://www.deeplearningbook.org/>.
- *Understanding deep learning.* S. Prince
<http://udlbook.github.io/udlbook/>
- *Dive into Deep Learning.* A. Zhang, Z. Lipton, M. Li, A. Smola <http://d2l.ai/>

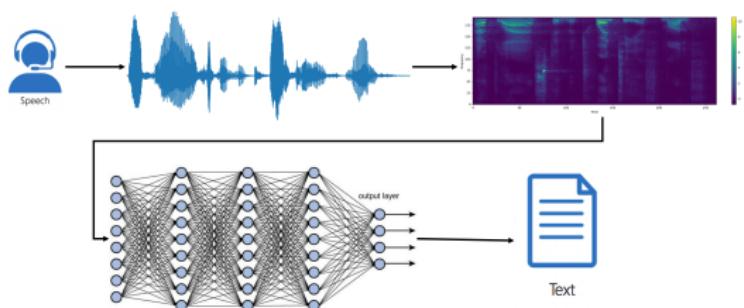
Дополнительные материалы

- CS231n - DL for CV (Stanford)
- CS224n - NLP with DL (Stanford)
- http://github.com/yandexdataschool/Practical_DL
- http://github.com/yandexdataschool/nlp_course
- <http://dlcourse.ai/>
- Pytorch Tutorials
- <http://kaggle.com>

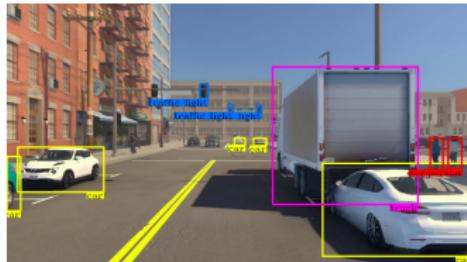
Приложения Deep Learning

Обработка аудио

- Speech-to-text
- Text-to-speech
- Voice cloning
- Синтез музыки



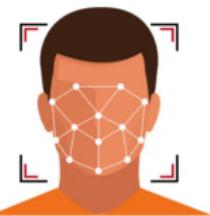
Computer Vision



Object detection



Segmentation



Face recognition



Style Transfer

Natural Language Processing



Машинный перевод



Яндекс.Алиса

Диалоговые системы



ChatGPT

Programming

Github Copilot / OpenAI Codex

Technical preview

Your AI pair programmer

The screenshot shows a dark-themed code editor interface for GitHub Copilot. At the top, there are tabs for files: `fetch_pic.js`, `push_to_git.py`, `d3_scale.js`, `fetch_stock.js`, and `material_ui.js`. The main code editor area contains the following JavaScript code:

```
1 const fetchNASAPictureOfDay = () => {
2   return fetch('https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY', {
3     method: 'GET',
4     headers: {
5       'Content-Type': 'application/json',
6     },
7   })
8     .then(response => response.json())
9     .then(json => {
10       return json;
11     });
12 }
```

A blue button labeled "Copilot" is visible at the bottom left of the code editor. Below the code editor, the GitHub Copilot logo (a white octocat icon) and the text "GitHub Copilot" are displayed.

Text-to-image

DALL-E / Stable Diffusion / Midjourney / Kandinskiy

Text Prompt an armchair in the shape of an avocado....

AI Generated
images



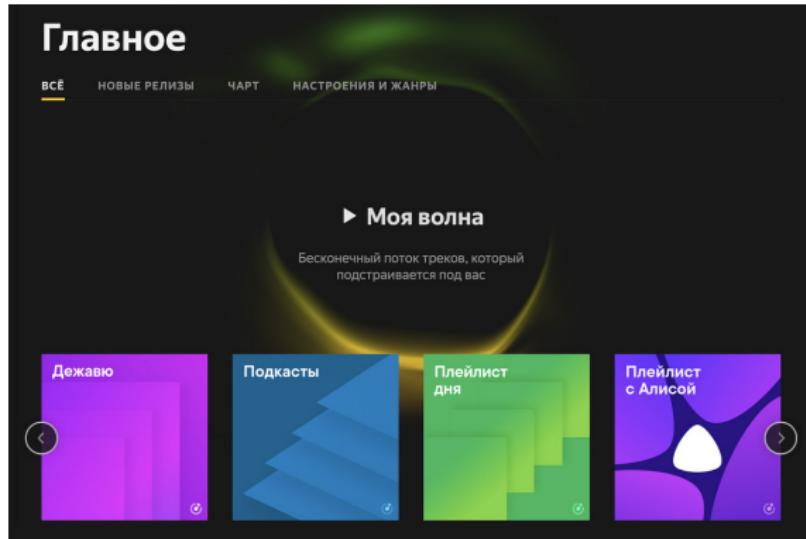
Поиск



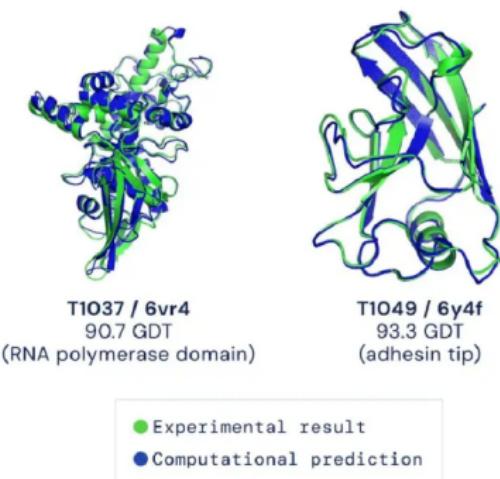
Введите поисковый запрос или URL



Рекомендательные системы



AlphaFold - предсказание пространственной структуры белка



Deep Reinforcement Learning



Atari games (2013)



AlphaGo (2016)



AlphaStar (2019)

Есть ли задачи, где DL не лучший вариант?

Есть ли задачи, где DL не лучший вариант?

Да!

Табличные данные - (пока что?) царство градиентного бустинга

Tabular Data
columns = attributes for those observations

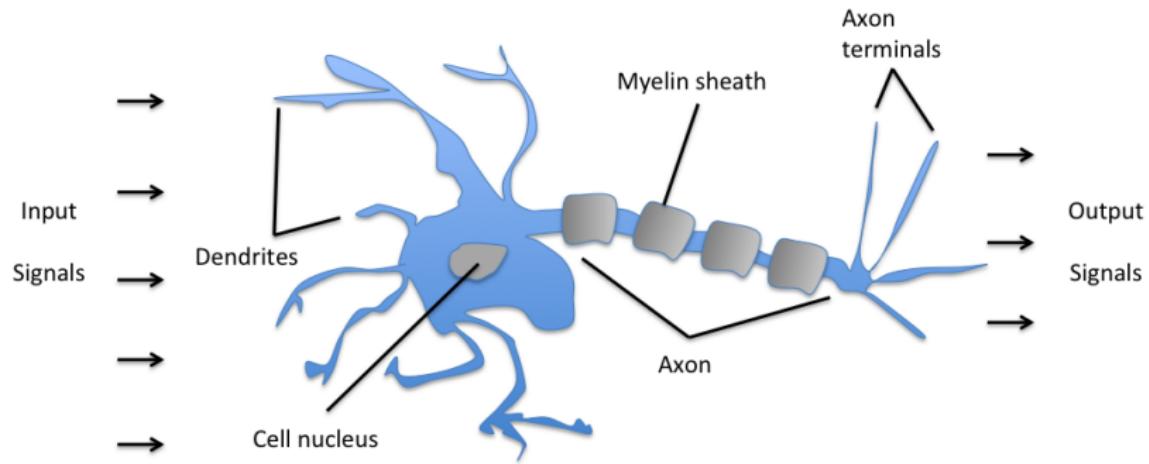
Rows = observations

Player	Minutes	Points	Rebounds	Assists
A	41	20	6	5
B	30	29	7	6
C	22	7	7	2
D	26	3	3	9
E	20	19	8	0
F	9	6	14	14
G	14	22	8	3
I	22	36	0	9
J	34	8	1	3

Нейросети хороши в автоматическом выделении признаков, а в табличных данных они уже выделены

Краткая история глубокого обучения

Краткая история глубокого обучения

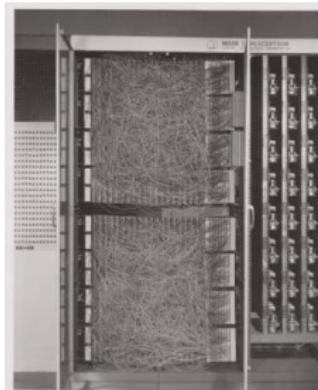
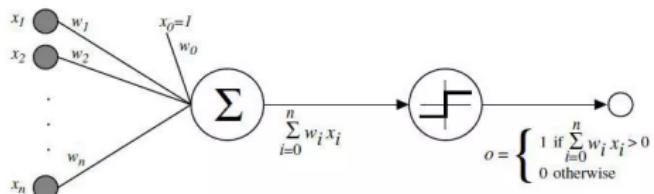


Schematic of a biological neuron.

Краткая история глубокого обучения

1943: McCulloch, Pitts - Математическая модель нейрона (без обучения)

1957: Rosenblatt - Перцептрон (обучение одного искусственного нейрона)



Бинарная классификация, распознавание цифр

Краткая история глубокого обучения

1956: Дартмутский семинар

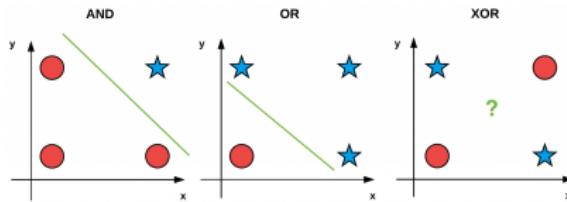
Мы предлагаем исследование искусственного интеллекта сроком в 2 месяца с участием 10 человек летом 1956 года в Дартмутском колледже, Гановер, Нью-Гэмпшир. Исследование основано на предположении, что всякий аспект обучения или любое другое свойство интеллекта может в принципе быть столь точно описано, что машина сможет его симулировать. Мы попытаемся понять, как обучить машины использовать естественные языки, формировать абстракции и концепции, решать задачи, сейчас подвластные только людям, и улучшать самих себя. Мы считаем, что существенное продвижение в одной или более из этих проблем вполне возможно, если специально подобранный группой учёных будет работать над этим в течение лета.

Краткая история глубокого обучения

1969: Minsky, Papert - книга "Перцептроны".

Анализ ограничений перцептрана:

- Перцептрон задает линейную разделяющую поверхность
- Сложные функции, такие как XOR, нельзя моделировать



- Нужны многослойные перцептроны
- В то время их обучать не умели

1970-е: AI Winter

Краткая история глубокого обучения

1986: Rumelhart, Hinton, Williams - Backpropagation,
алгоритм обратного распространения ошибки для обучения
нейросетей

Вообще говоря, открывался и переоткрывался несколько раз
до этого, в частности Seppo Linnainmaa (1970), Paul Werbos
(1982)

- 1980: Fukushima - Неокогнитрон, первая сверточная
нейросеть
- 1989: LeCun - CNN для распознавания рукописных
цифр (почтовые индексы)
- 1997: Hochreiter, Schmidhuber - LSTM

Краткая история глубокого обучения

Середина 1990-ых - 2000-ые - вторая "зима":

- Обучение требовало большого количества времени
- Хорошо обучить большие модели тогда не могли
- Классический ML (SVM, Random Forest) работал лучше

Небольшое количество энтузиастов продолжали заниматься нейросетями



Краткая история глубокого обучения

История из первых рук



Краткая история глубокого обучения

- 2006: Термин Deep Learning
- 2006: Hinton, Deep Belief Networks - успешное обучение глубоких моделей за счет послойного предобучения без учителя
- 2009: Успех в распознавании речи

Краткая история глубокого обучения

2010-е: Революция глубокого обучения

- 2012: AlexNet, победа на ImageNet со сверточными сетями
- 2013: Deep RL, Atari games
- 2014: GAN
- 2015: ResNet
- 2016: AlphaGo
- 2017: Transformers (Attention is all you need)
- 2018: BERT, NLP ImageNet moment
- 2020: GPT-3
- 2022: ChatGPT

ML Recap

Суть машинного обучения

По сути все, что мы делаем в машинном обучении - это аппроксимация и оптимизация функций:

- есть какая зависимость в реальных данных, очень сложная функция
- мы пытаемся подобрать модель (функцию), которая хорошо аппроксимирует реальную зависимость
- для этого выбираем какой-то класс моделей, обычно параметрический
- подгоняем параметры так, чтобы модель хорошо описывала данные
- то есть оптимизируем какую-то функцию ошибки или функцию качества

Нейронные сети - это очень мощный и гибкий класс моделей.

Обучение с учителем

Обучающий датасет

$$D = \{x_i, y_i\}_{i=1}^N$$

Модель - семейство функций

$$\hat{y} = f(x, \theta)$$

Функция потерь (Loss function)

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{y}_i) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i, \theta))$$

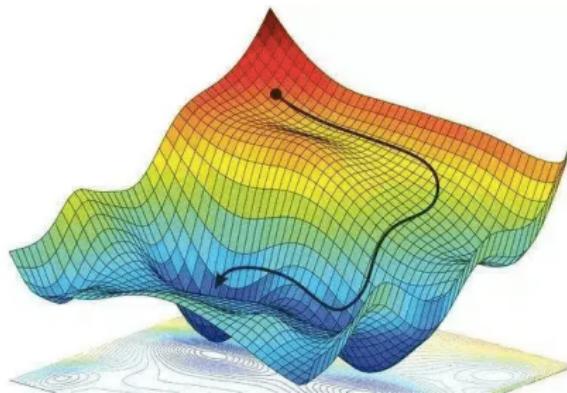
Нужно настроить параметры

$$\theta = \operatorname{argmin}_{\theta} L(\theta)$$

Градиентный спуск

Как оптимизировать функцию потерь?

Обычно градиентный спуск.



$$\theta' = \theta - \alpha \frac{\partial L}{\partial \theta}$$

Обобщающая способность

- Важно, чтобы модель хорошо обобщалась на новые данные (generalization).
- Разбиваем на train/test, обучаем на train и проверяем качество на test.

Модель может быть:

- слишком простой - underfitting
- слишком сложной - overfitting

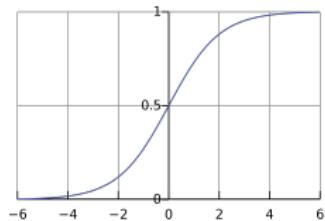
Регуляризация - методы для борьбы с overfitting.

Логистическая регрессия

Бинарная классификация: метки $y_i \in \{0, 1\}$

Как сделать так, чтобы
предсказания были от 0 до 1
(вероятности)?

$$P(y = 1|x, \theta) = \sigma(\theta^T x) = \frac{1}{1 + \exp(-\theta^T x)}$$



Функция потерь:

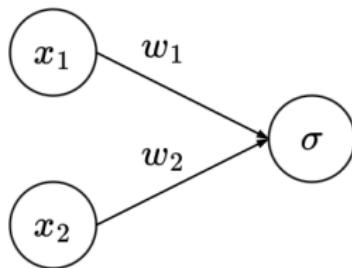
$$L(\theta) = - \sum_i (y_i \log P(y = 1|x, \theta) + (1 - y_i) \log(1 - P(y = 1|x, \theta)))$$

Получаем линейную разделяющую границу

Нейронные сети

От логистической регрессии..

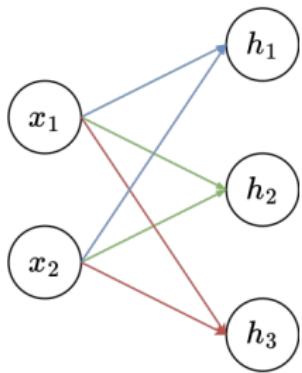
$$y = \sigma(w_0 + w_1x_1 + w_2x_2)$$



Логистическая регрессия = 1 нейрон

Вход -> линейная операция -> нелинейность -> выход

..к простейшей нейросети

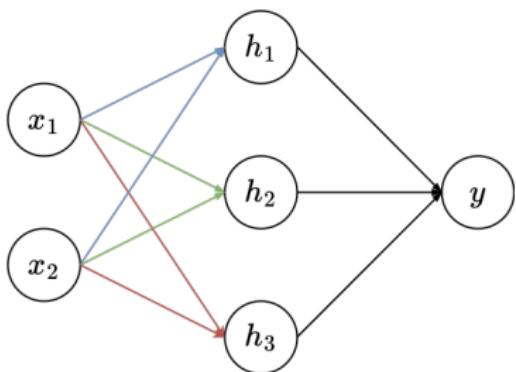


$$h_1 = \sigma(b_1 + w_{11}x_1 + w_{12}x_2)$$

$$h_2 = \sigma(b_2 + w_{21}x_1 + w_{22}x_2)$$

$$h_3 = \sigma(b_3 + w_{31}x_1 + w_{32}x_2)$$

..к простейшей нейросети



$$h_1 = \sigma(b_1 + w_{11}x_1 + w_{12}x_2)$$

$$h_2 = \sigma(b_2 + w_{21}x_1 + w_{22}x_2)$$

$$h_3 = \sigma(b_3 + w_{31}x_1 + w_{32}x_2)$$

$$y = \sigma(b' + w'_{11}h_1 + w'_{12}h_2 + w'_{13}h_3)$$

В матричном виде

$$h_1 = \sigma(b_1 + w_{11}x_1 + w_{12}x_2)$$

$$h_2 = \sigma(b_2 + w_{21}x_1 + w_{22}x_2)$$

$$h_3 = \sigma(b_3 + w_{31}x_1 + w_{32}x_2)$$

$$h = \sigma(Wx + b)$$

$$y = \sigma(W'h + b')$$

$$y = \sigma(b' + w'_{11}h_1 + w'_{12}h_2 + w'_{13}h_3)$$

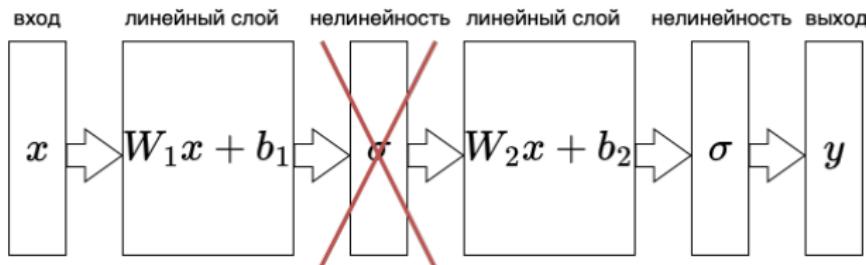


$$y = \sigma(W_2\sigma(W_1x + b_1) + b_2)$$

Нелинейность нужна!

Что если уберем нелинейность?

Суперпозиция линейных функций - снова линейная функция.



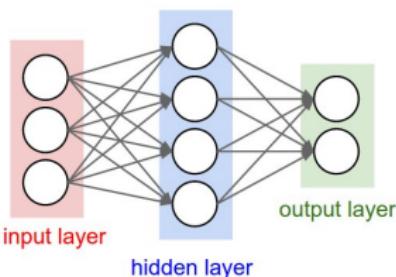
$$y = \sigma(W_2(W_1x + b_1) + b_2) = \sigma(W_2W_1x + W_2b_1 + b_2) = \sigma(W'x + b')$$

Но! Не все нелинейности одинаково полезны!

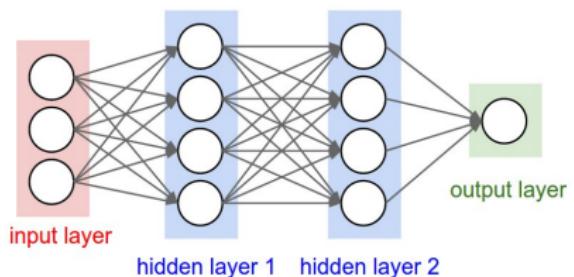
Multilayer Perceptron (MLP)

Многослойный перцептрон

$$y = \sigma(W_2\sigma(W_1x + b_1) + b_2)$$



$$y = \sigma(W_3\sigma(W_2\sigma(W_1x + b_1) + b_2) + b_3)$$



W - веса (weights), b - смещение (bias), σ - функция активации

Организация в слои - можно представить в виде матричных операций и легко параллелизовать

Чем DL отличается от классического ML?

Классическое машинное обучение:

- feature engineering, генерация признаков (MFCC для распознавания речи, SIFT для обработки изображений,..)
- + модель

Глубокое обучение:

- автоматическое выучивание сложных признаков, representation learning
- end-to-end training

Причины успеха DL

- Вычислительные мощности (GPUs, TPUs)
- Огромное количество данных
- Улучшение алгоритмов и понимания того, как обучать глубокие сети
- Open source инструменты и модели
- Интерес бизнеса, больших корпораций

Причины успеха DL

Geoffrey Hinton:

- Our labeled datasets were thousands of times too small
- Our computers were millions of times too slow
- We initialized the weights in a stupid way
- We used the wrong type of non-linearity

Важность данных

Year	Breakthroughs in AI	Datasets (First Available)	Algorithms (First Proposed)
1994	Human-level spontaneous speech recognition	Spoken Wall Street Journal articles and other texts (1991)	Hidden Markov Model (1984)
1997	IBM Deep Blue defeated Garry Kasparov	700,000 Grandmaster chess games, aka "The Extended Book" (1991)	Negascout planning algorithm (1983)
2005	Google's Arabic- and Chinese-to-English translation	1.8 trillion tokens from Google Web and News pages (collected in 2005)	Statistical machine translation algorithm (1988)
2011	IBM Watson became the world Jeopardy! champion	8.6 million documents from Wikipedia, Wiktionary, Wikiquote, and Project Gutenberg (updated in 2010)	Mixture-of-Experts algorithm (1991)
2014	Google's GoogLeNet object classification at near-human performance	ImageNet corpus of 1.5 million labeled images and 1,000 object categories (2010)	Convolution neural network algorithm (1989)
2015	Google's Deepmind achieved human parity in playing 29 Atari games by learning general control from video	Arcade Learning Environment dataset of over 50 Atari games (2013)	Q-learning algorithm (1992)
Average No. of Years to Breakthrough:		3 years	18 years

Важнее собирать данные, а не улучшать алгоритмы

В следующий раз

- Как обучать нейронные сети
- Backpropagation, алгоритм обратного распространения ошибки
- Основные функции активации