NNs. Backpropagation. CNN. RNN. LSTM. GAN.

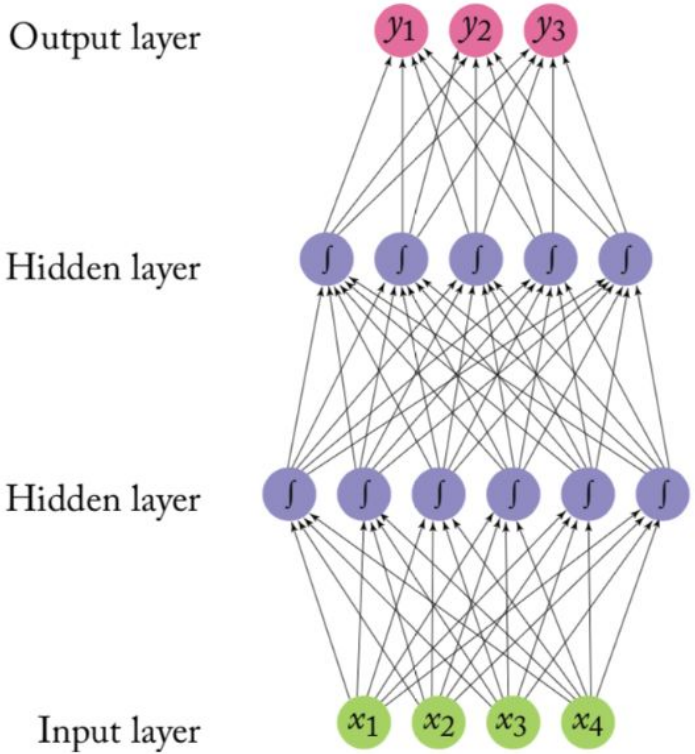
Маша Шеянова, masha.shejanova@gmail.com

План

- как устроена нейросеть
- backpropagation explained
- разные архитектуры
- CNN
- RNN, LSTM
- инструменты NER (по заказам)

Как устроена нейросеть

нейросеть in a nutshell



На входе — вектор признаков.

На каждой стрелочке — какие-то коэффициенты.

На выходе — вектор вероятностей того или иного класса.

“Нейрон” == один кружочек == функция от выдачи предыдущего слоя.

Перцептрон

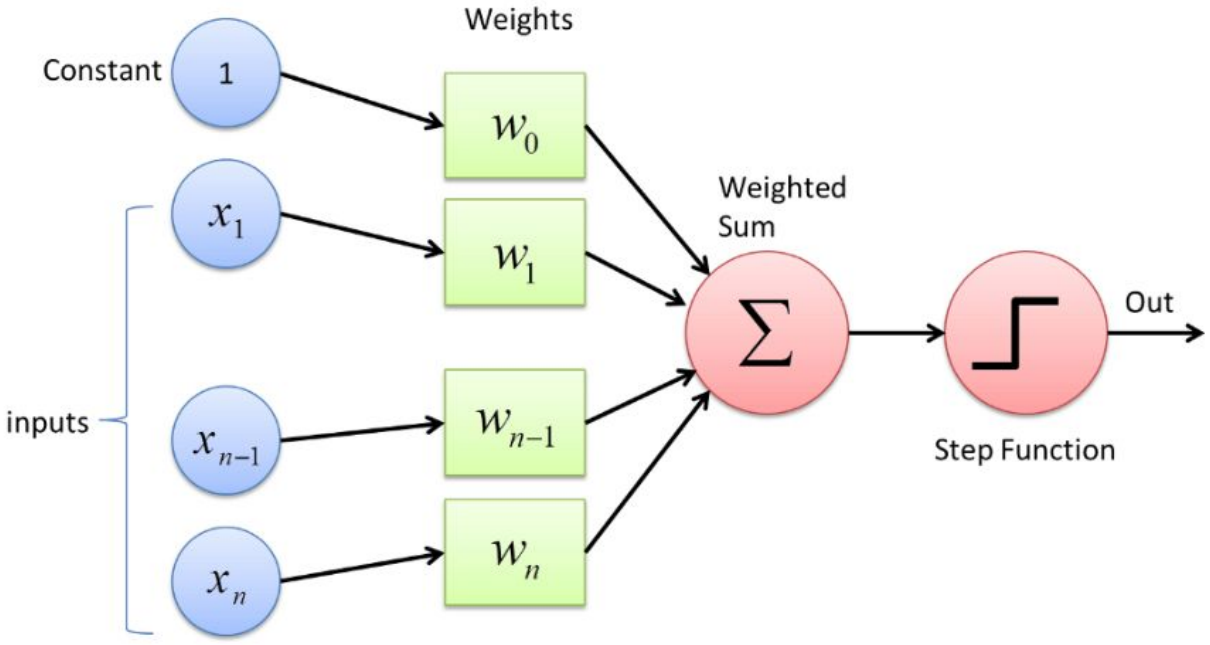
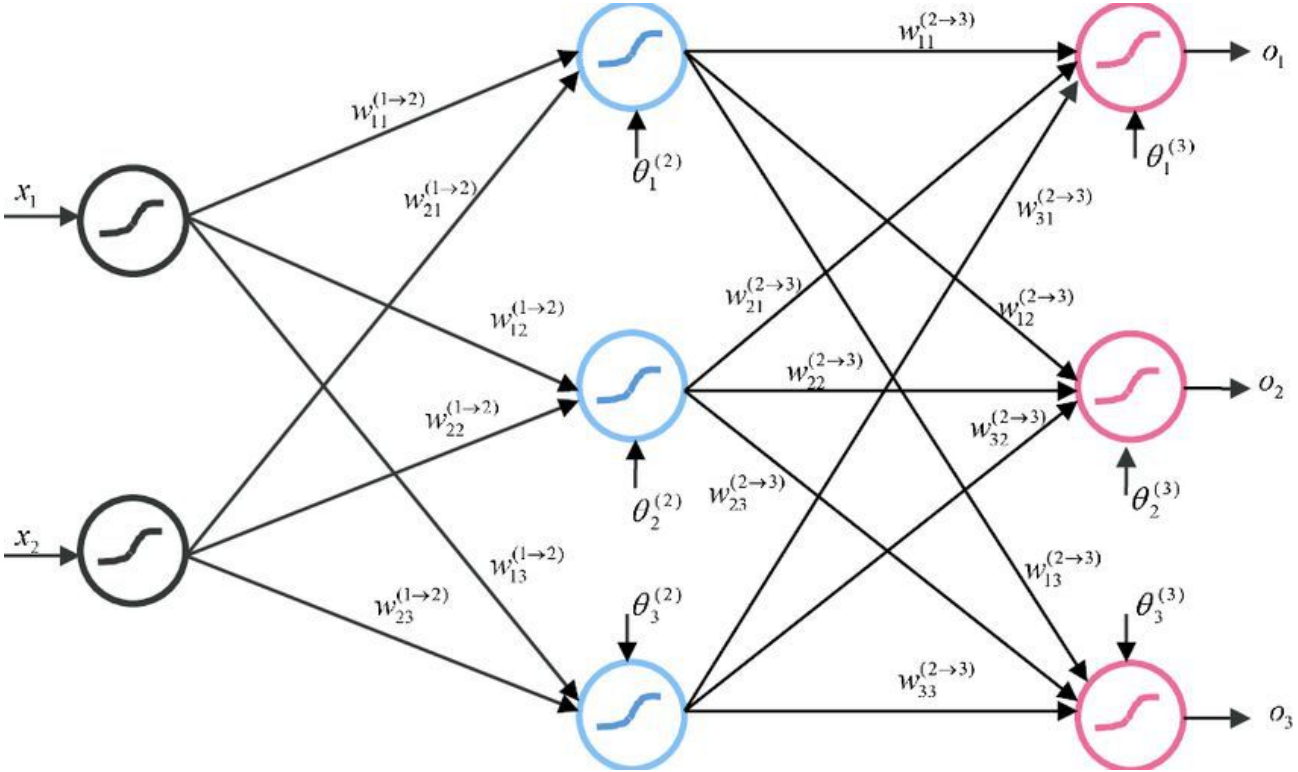


Fig : Perceptron

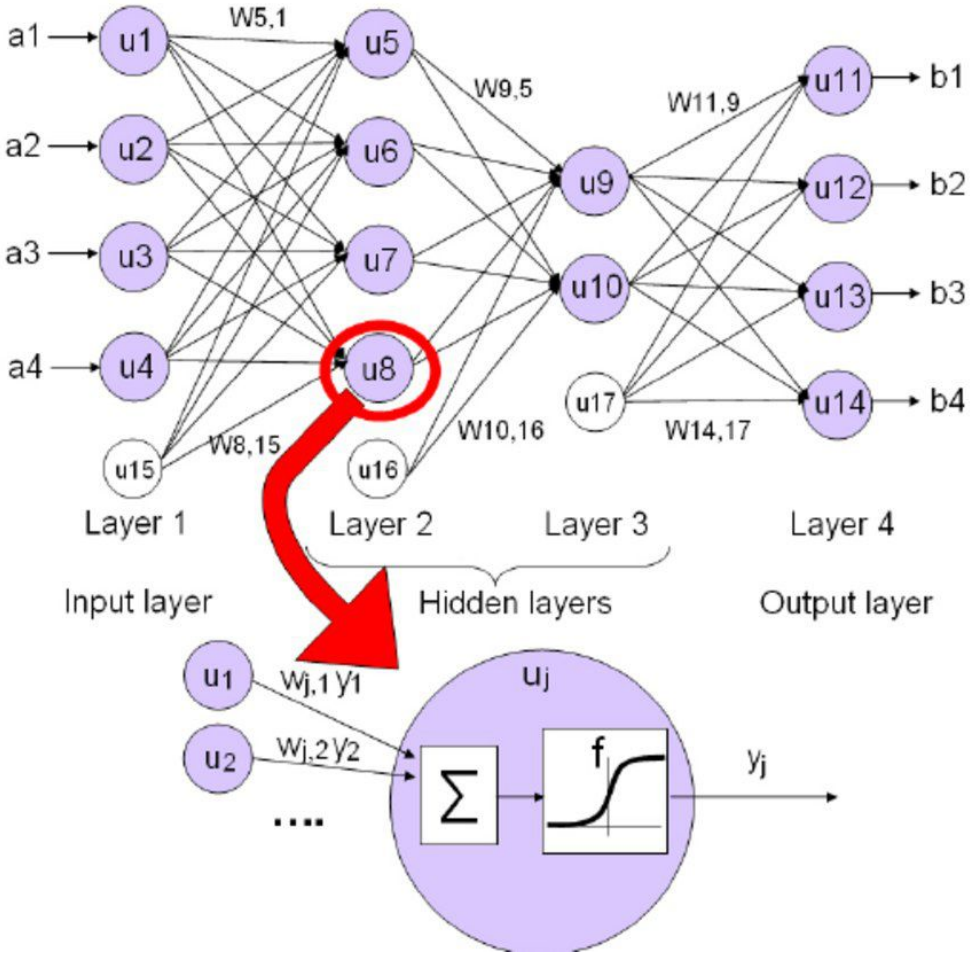
Однослойная нейросеть. С неё всё началось.

Нейросеть как функция



Один нейрон — функция от **вектора** параметров (w_{11}, w_{21}), умноженного на **вектор** объекта x (+ b).
*f(**wx** + b)*

Слой — функция от **матрицы** параметров * x + **вектор** b.
*f(**Wx** + **b**).*



Четырёхслойная нейросеть.

Универсальная теорема аппроксимации: любую функцию можно приблизить нейросетью.

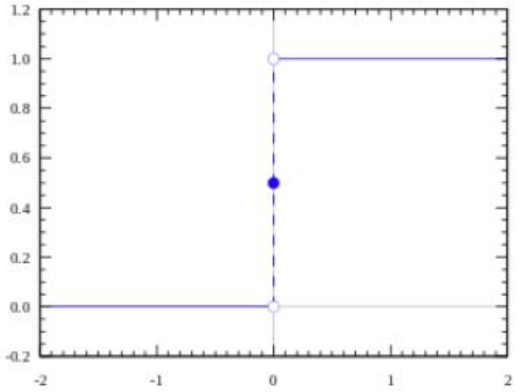
Функции активации

Почему “функция активации”?

… по аналогии с естественными нейросетями.

Справа — “step function”: нейрон активировался (1) или нет (0). (Источник картинки)

Но для artificial NN нужно что-то дифференцируемое.



Почти все картинки этого раздела взяты отсюда.

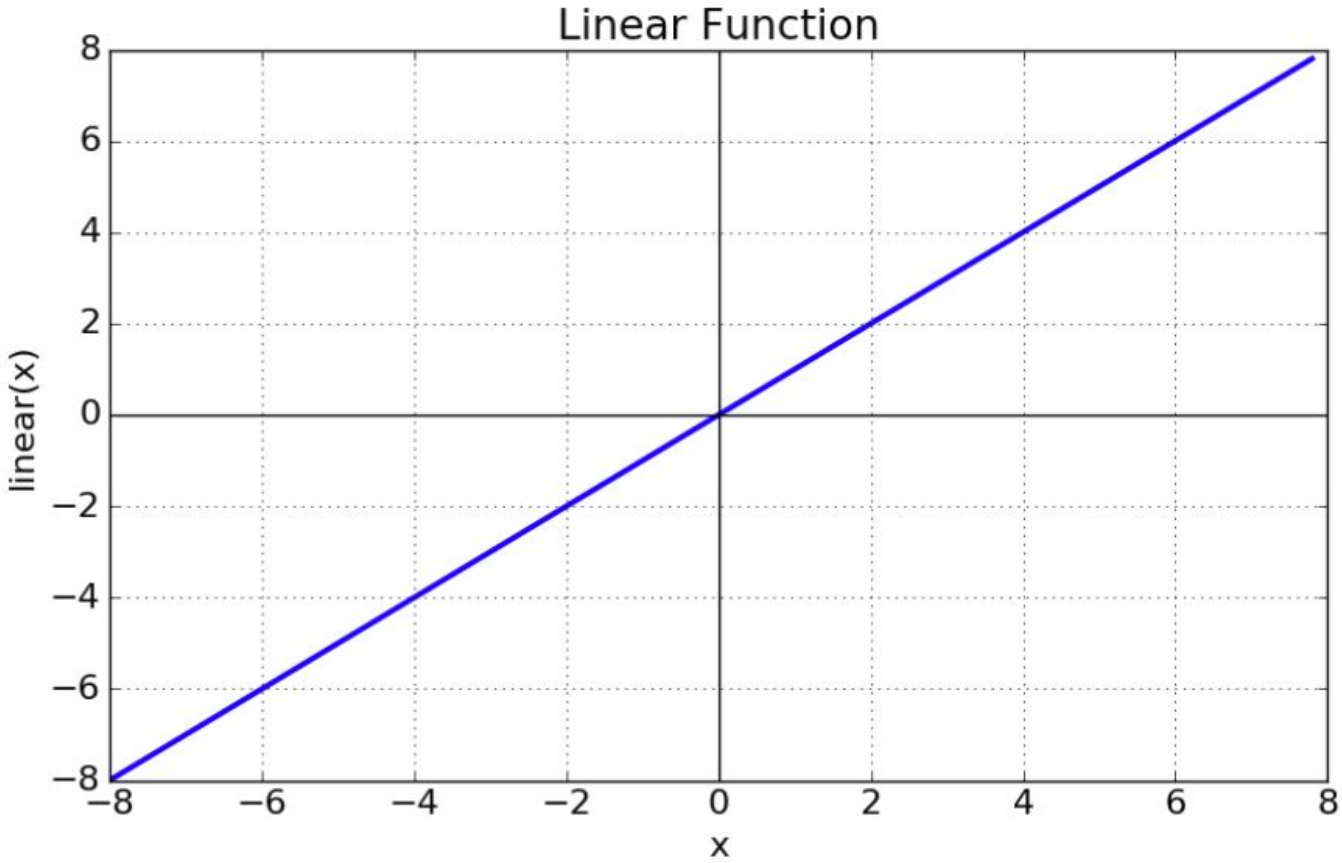
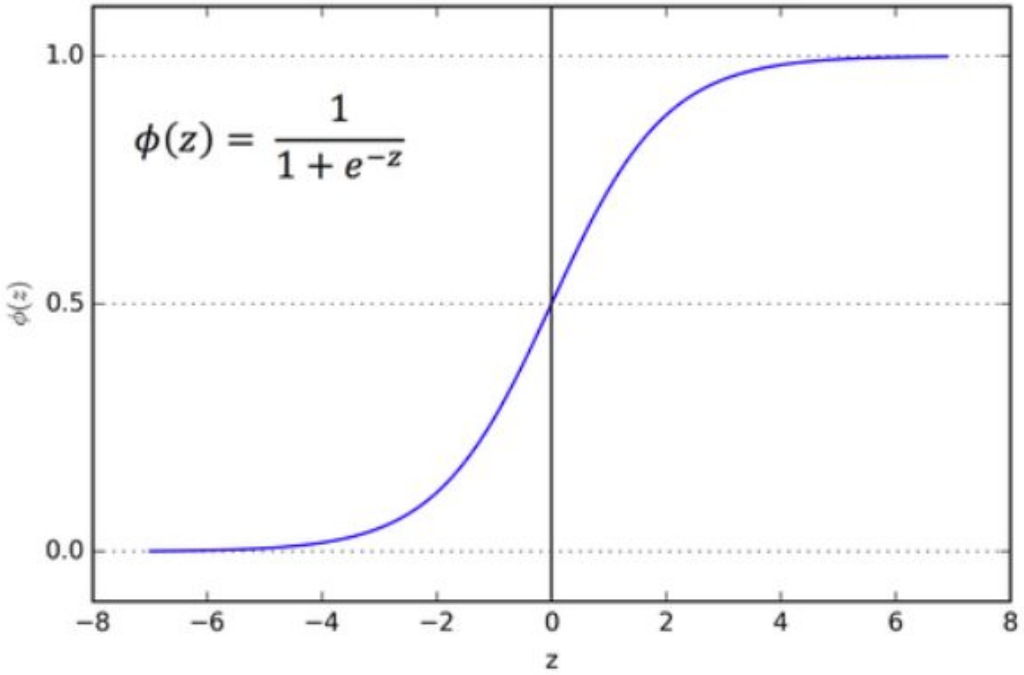


Fig: Linear Activation Function

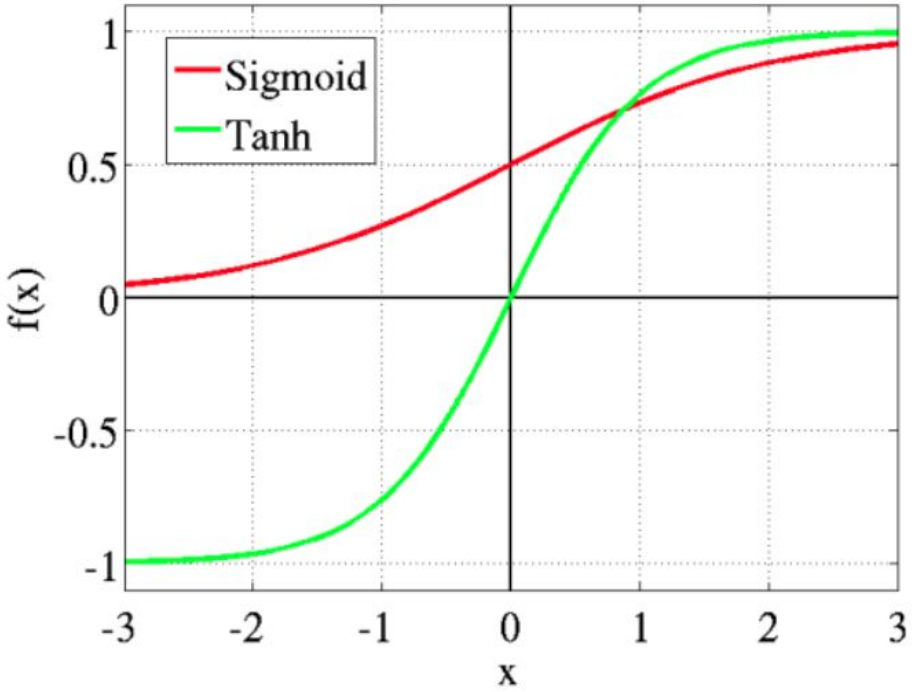
Sigmoid function



Это то же самое, что и логистическая регрессия.

Изменяется от 0 до 1 (и поэтому — хороший выбор для выдачи вероятностей).

Tanh or hyperbolic tangent Activation Function



Похожа на предыдущую, но изменяется от -1 до 1.

ReLU и Leaky ReLU

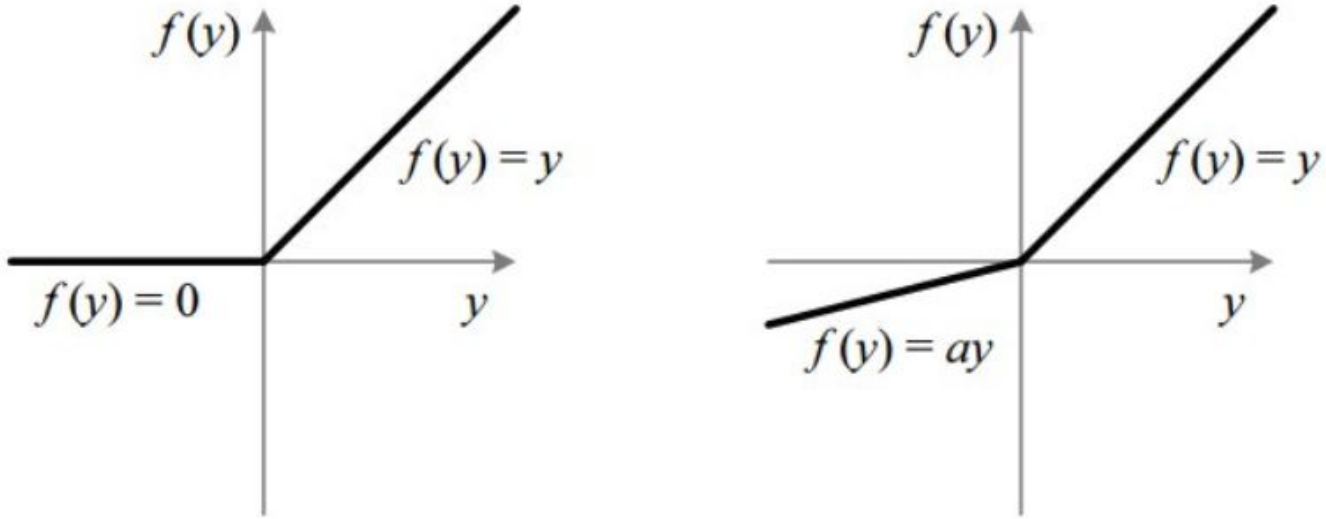


Fig : ReLU v/s Leaky ReLU

Backpropagation

Вспомним градиентный спуск

Производная

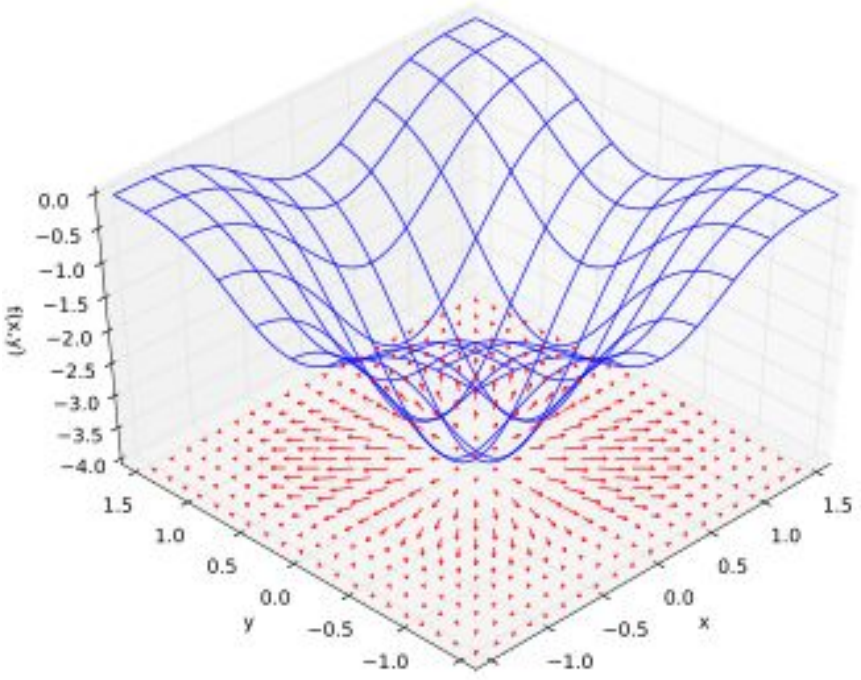
Производная — это мера, насколько быстро растёт функция.

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}$$

У функции от n переменных f(x1, x2, ... x_n) нет одной общей производной — зато есть n частные производные.

$$\frac{\partial f}{\partial x_k}(a_1, \cdots, a_n) = \lim_{\Delta x \to 0} \frac{f(a_1, \ldots, a_k + \Delta x, \ldots, a_n) - f(a_1, \ldots, a_k, \ldots, a_n)}{\Delta x}$$

Что такое градиент



Градиент — это вектор, элементы которого — **значения всех возможных частных производных** в конкретной точке.

Градиент соответствует вектору, указывающему направление **наибольшего роста функции**.

Идея

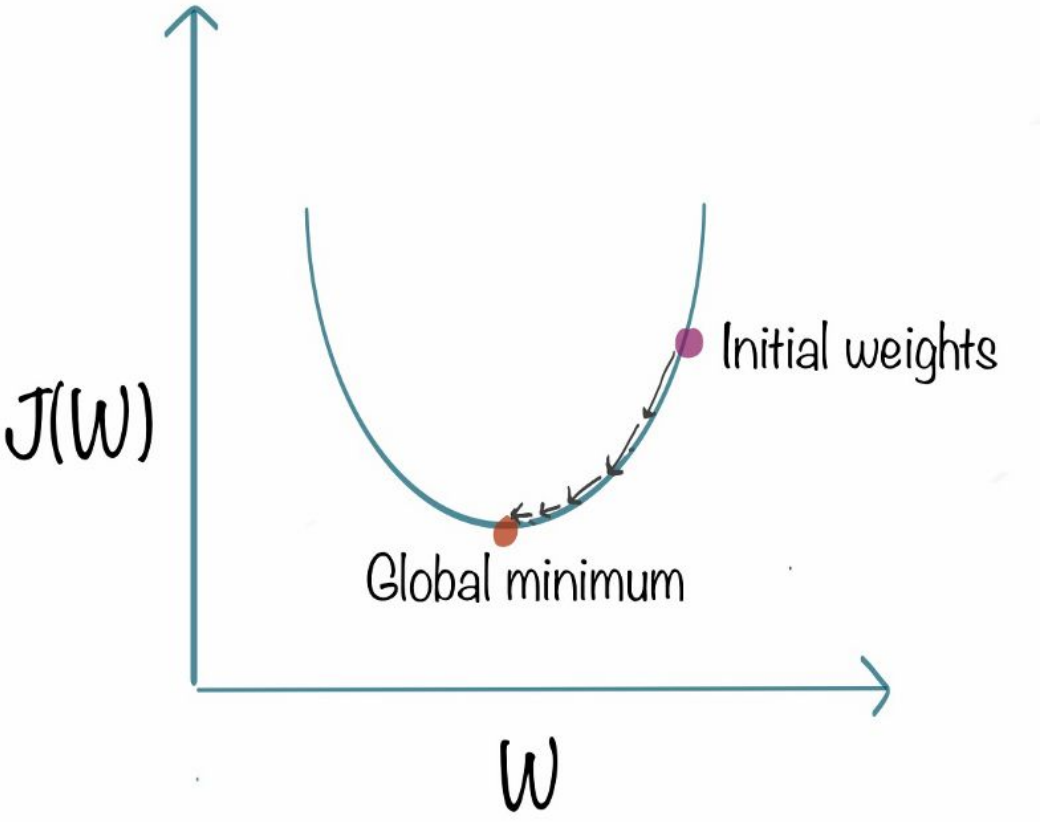
loss function = cost function = error function = функция потерь = J(W)

Её мы хотим минимизировать.

Теперь мы умеем находить, в каком направлении функция растёт быстрее всего. Но нам нужен минимум функции потерь, а не максимум!

Решение очевидно: найдём градиент и пойдём в обратную сторону.

С какой скоростью? Растёт быстро — с большой, медленно — с маленькой.



Источник картинки — очень понятно про то, как оно работает и какое бывает.

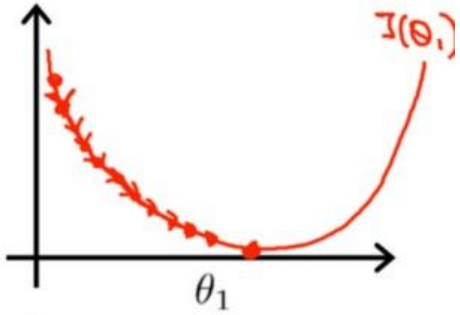
Шаги:

- подобрать случайные коэффициенты
- вычислить градиент функции потерь в этой точке
- обновить коэффициенты
- повторять, пока не сойдётся

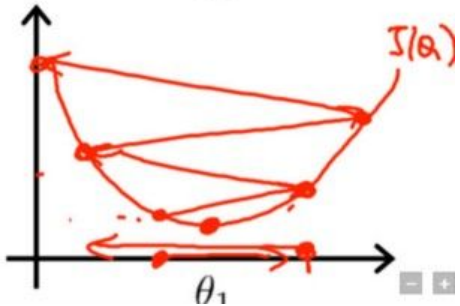
Learning rate

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. (отсюда)

Каким бывает градиентный спуск

- **Batch gradient descent**

  Считает градиент функции потерь с параметрами W сразу для всех обучающих данных. Работает жутко медленно.

- **Stochastic gradient descent (SGD)**

  Рандомно выбирает точку данных каждый раз

- **Mini-batch gradient descent**

  Выбираем кусочек выборки и по нему считаем

Что делать, если всё ещё ничего непонятно

Непонимание градиентного спуска, в принципе, не помешает вам решать типичные задачи готовыми инструментами. Но может помешать улучшать модель и решать проблемы, если что-то пойдет не так.

Если всё ещё ничего непонятно, keep calm and:

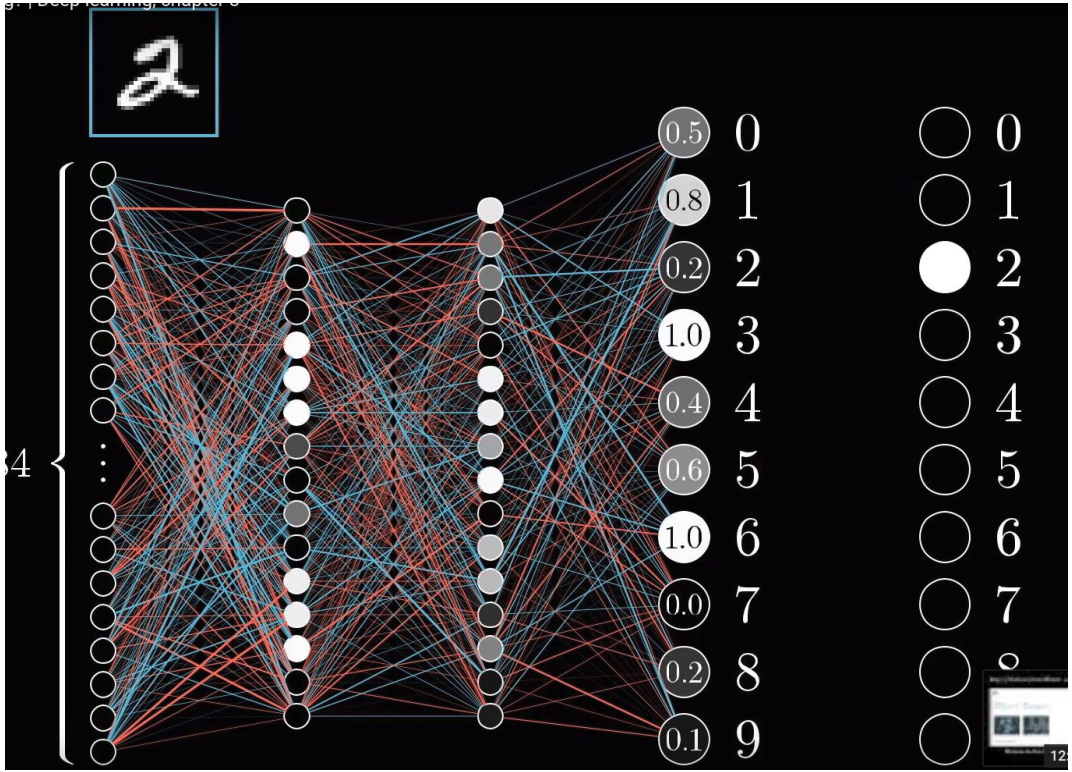
- пройдите небольшой курс по multivariate calculus на khan academy
- посмотрите вот это видео про градиентный спуск
- прочитайте эту и эту статью
- если удастся сформулировать вопросы, feel free to ask

Интуиция за backpropagation

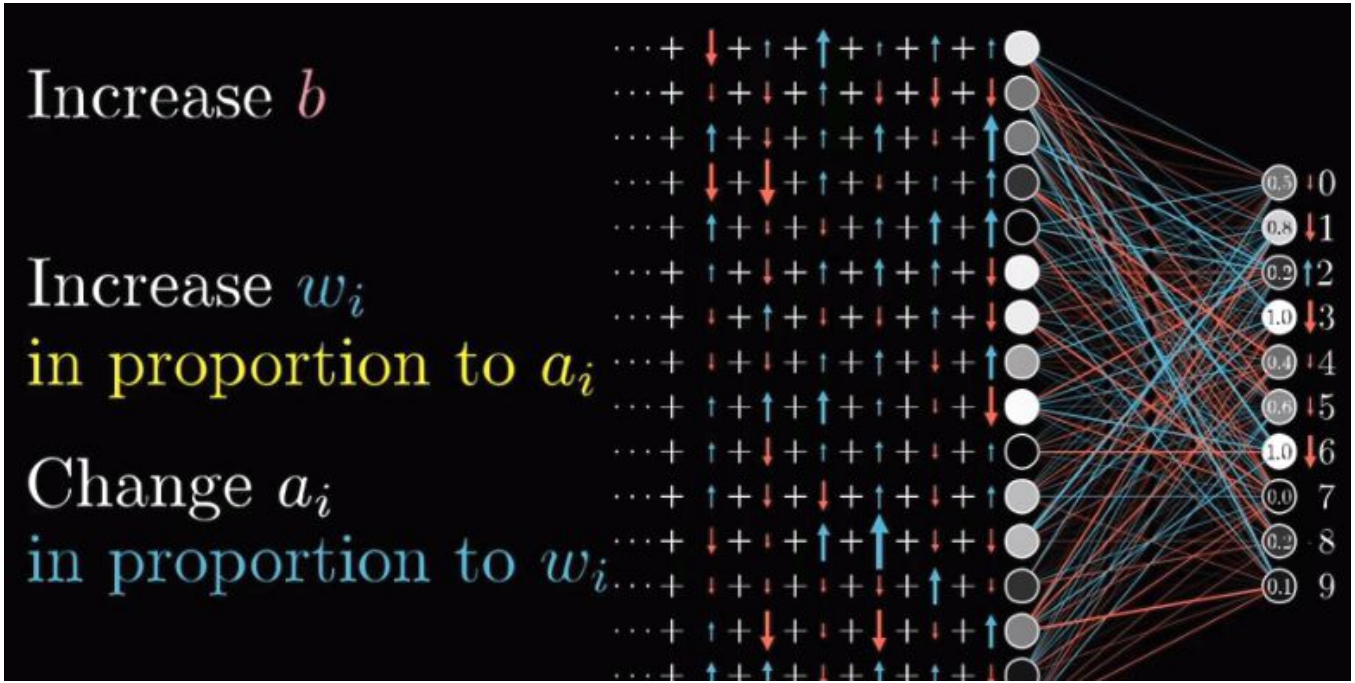
Функция потерь

Она может быть разной, например так:

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2,$$

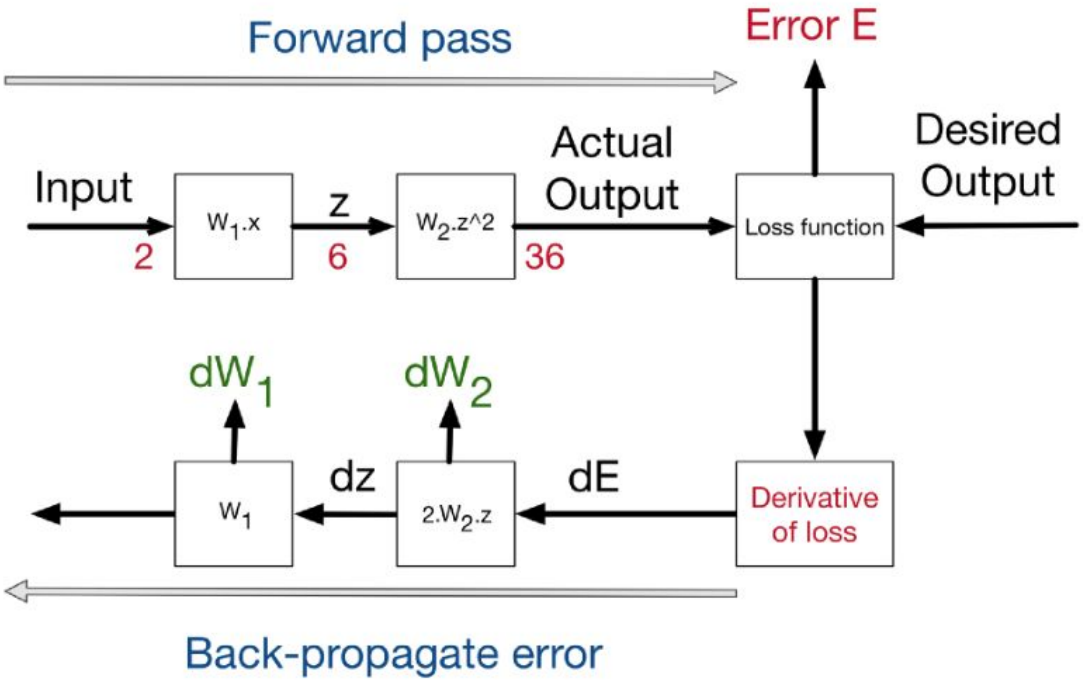


Суммирование ошибки



А дальше оптимизируем функцию потерь градиентным спуском.

Обратное распространение



Теория за backpropagation

Нейросеть — это тоже функция

x — входные данные

h1 = f1(W1 * x + b1)

h2 = f2(W2 * h1 + b2)

y_pred = f3(W3 * h2 + b2)

y_pred = f3(W3 * f2(W2 * f1(W1 * x + b1) + b2) + b2)

C = avg_sum(**y_pred** - **y_true**)^2

Композиция функций

В математике:

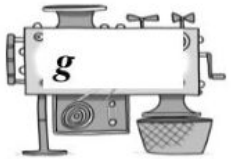
Substituting a function or it's value into **another** function.

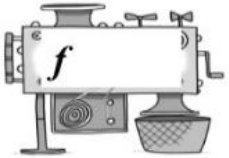
Second

$$f\left(g\left(x\right)\right)$$

First

(inside parentheses always first)





OR

$$f \circ g\left(x\right)$$

В программировании — то же самое!

Chain Rule

Это правило про то, как брать производную от композиции функций.

$$(f \circ g)' = (f' \circ g) \cdot g'.$$

This may equivalently be expressed in terms of the variable. Let $F = f \circ g$, or equivalently, $F(x) = f(g(x))$ for all x. Then one can also write

$$F'(x) = f'(g(x))g'(x).$$

Псевдокод для backprop слоя

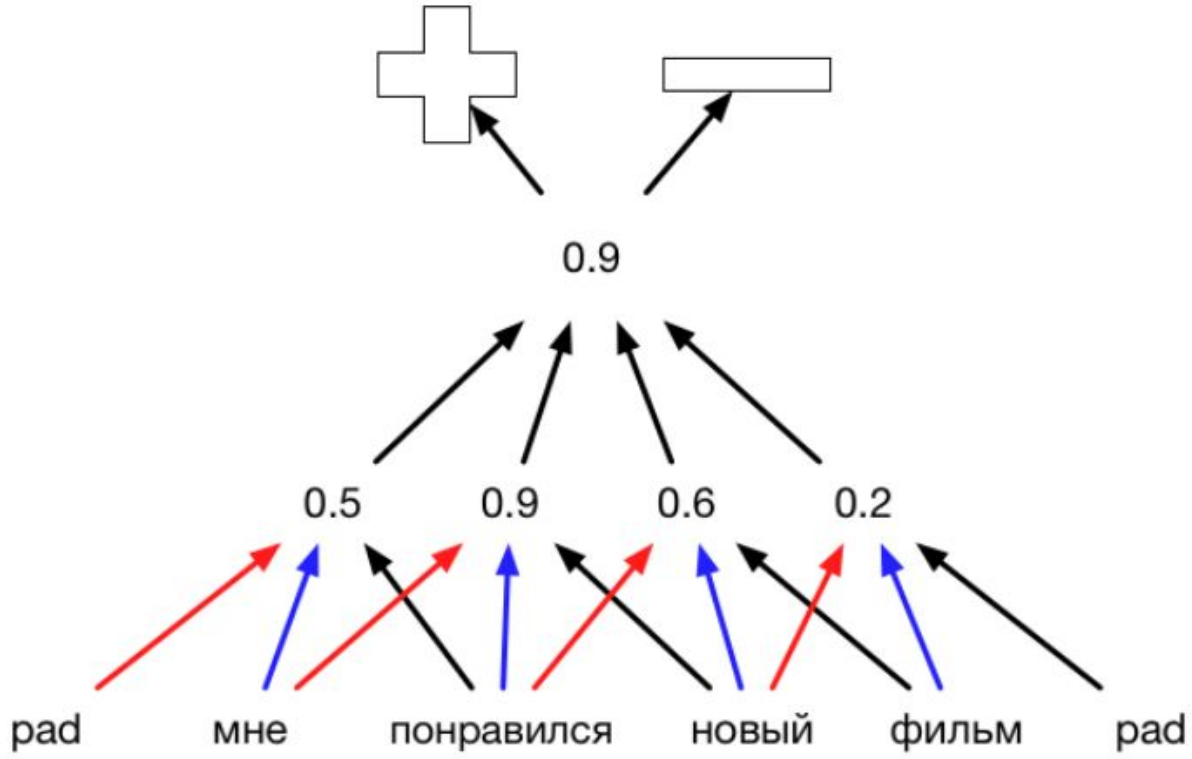
```
def backpropagation(loss):
  for layer in NN:
    layer.layer_loss *= layer.f_by_w(loss)
    loss *= layer.f_by_x(loss)
   return loss
```

А вот здесь есть код.

CNN

Классификатор на основе сверточной сети

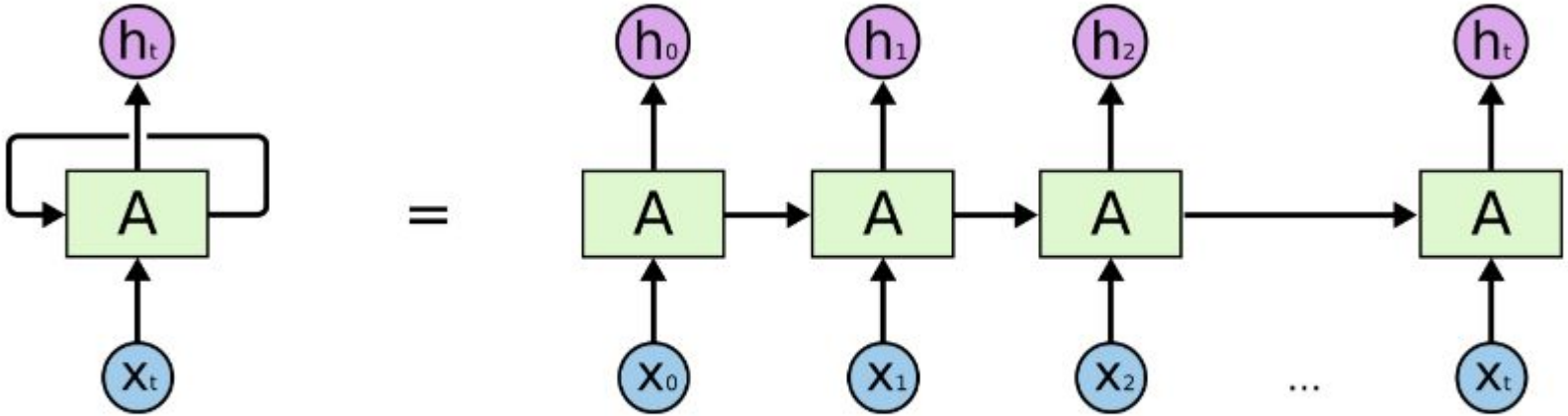
- $y \in [0, 1]$ - истинные значения
- $\hat{y} = c$ - предсказанные значения



RNN

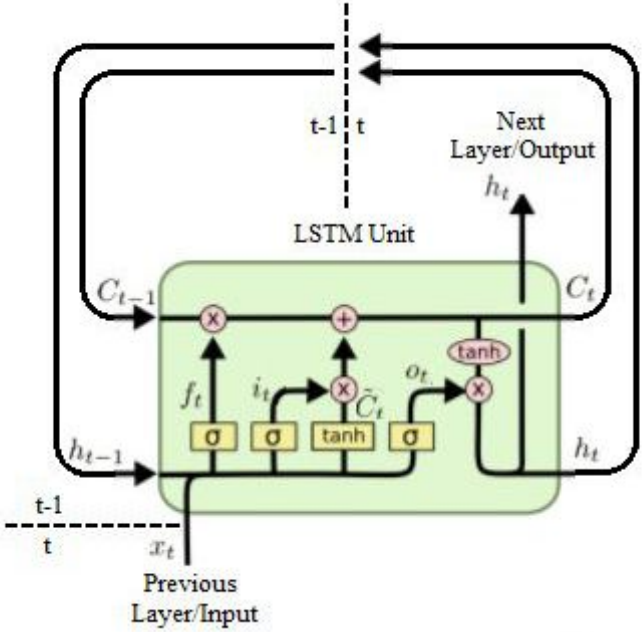
RNNs

В отличие от обычной нейросети, они получают на вход не только данные, но и выход предыдущей клетки RNN.

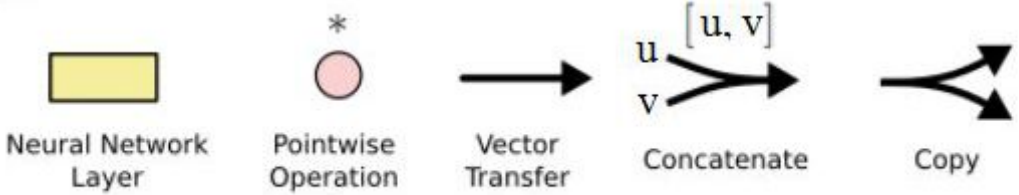


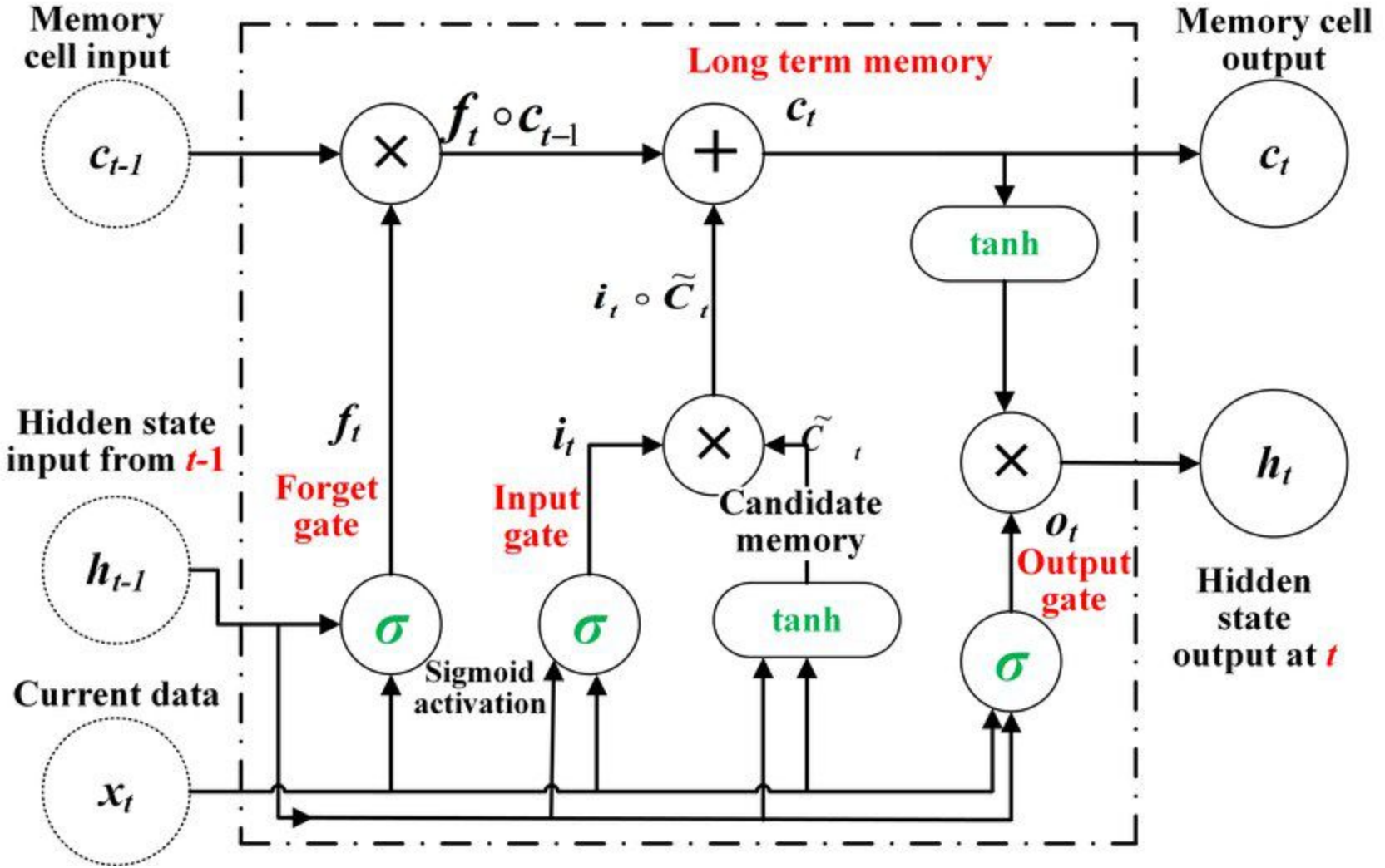
An unrolled recurrent neural network.

LSTM

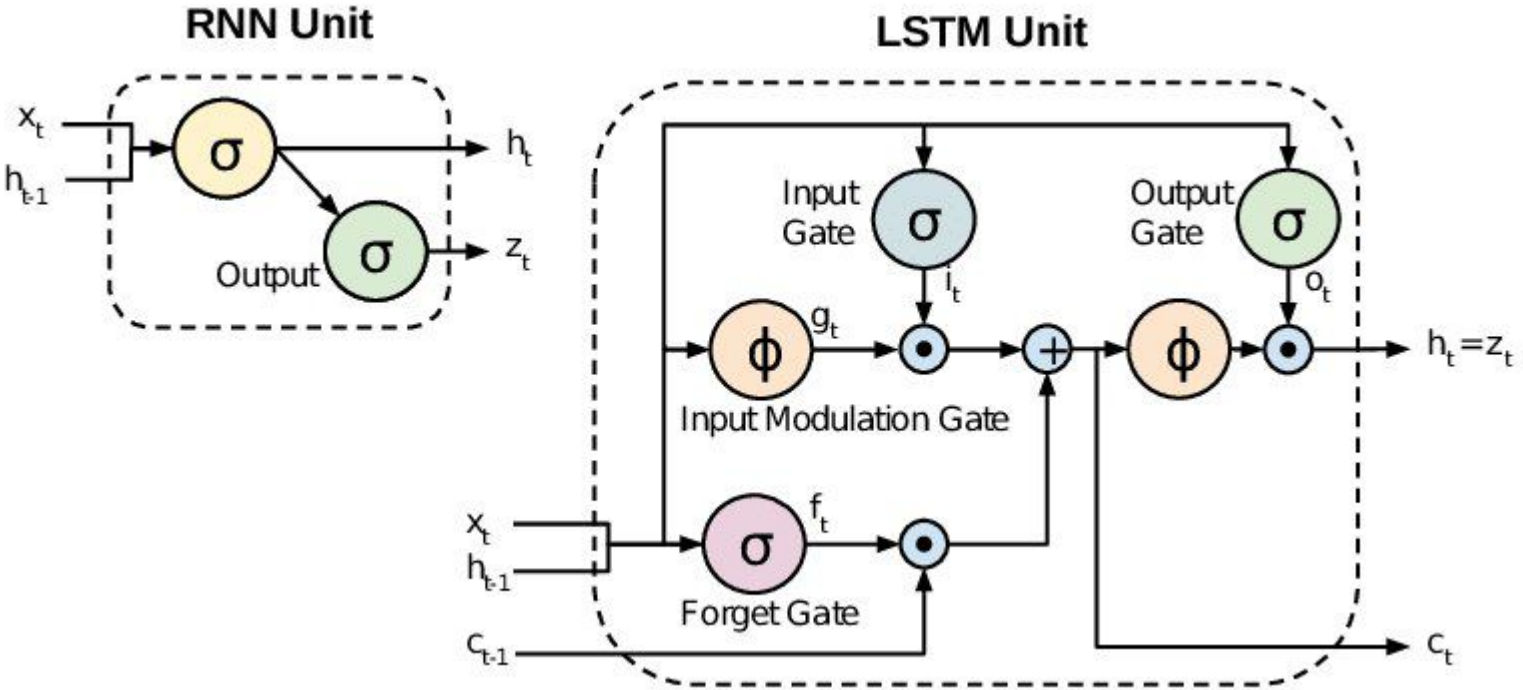


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$




Memory cell input
c_{t-1}
$f_t \circ c_{t-1}$
Long term memory
c_t
Memory cell output
c_t
tanh
$i_t \circ \tilde{C}_t$
Hidden state input from t-1
h_{t-1}
f_t
Forget gate
i_t
Input gate
$\tilde{C}_t$
Candidate memory
o_t
Output gate
h_t
Hidden state output at t
Current data
x_t
σ
Sigmoid activation
σ
tanh
σ

GRU vs LSTM


RNN Unit
x_t
h_{t-1}
σ
h_t
Output
σ
z_t
LSTM Unit
Input Gate
σ
i_t
Output Gate
σ
o_t
g_t
ϕ
Input Modulation Gate
ϕ
$h_t = z_t$
x_t
h_{t-1}
σ
f_t
Forget Gate
c_{t-1}
c_t

Немного “практики”

Библиотеки для нейросетей

- TensorFlow
- Keras (обёртка над TensorFlow)
- PyTorch

Потыкать в интерфейс

keras.js

Туториалы по keras

- от kaggle
- Approaching (Almost) Any NLP Problem on Kaggle (в конце)
- на картинках
- на nlp с длинным объяснением
- про всё, о чем я сегодня рассказывала про NN

Ресурсы

Почитать

- Understanding Activation Functions in Neural Networks
- Activation Functions in Neural Networks
- Neural networks and back-propagation explained in a simple way
- Gentle Dive into Math Behind Convolutional Neural Networks
- Understanding LSTMs (рекуррентные нейронки, lstm)

Посмотреть

Отличная серия видео про нейросети понятным языком:

- But what *is* a Neural Network
- Understanding Gradient Descent
- What backpropagation is really doing?
- Math for backpropagation

Livecoding a NN library (на странноватом новом питоне).

NER: теория, ресурсы

Инструменты и ресурсы

- natasha
- spacy
- NeuroNER
- sequence tagging
- больше ресурсов

Проекты

Как это должно быть

По объёму — что-то в районе 2-го дз, но с **большим** вниманием к данным.

Обязательно:

- анализ, того, что происходит в данных (описание датасета, перекос классов — для задачи классификации, топ tf-idf — для кластеризации)
- лингвистическая предобработка
- какая-то визуализация (можно из пункта 1)
- попробовать разные алгоритмы, гиперпараметры
- анализ результатов

Поощряется: оригинальные идеи, нетривиальные догадки о причинах.

Идеи

- классификация текстов
  - жанр / авторство
  - эмоции
  - “токсичность”
  - ...
- классификация предложений
  - QA (например, простенький QA чатбот на w2v)
  - парафраз
- слов?..
- кластеризация (чего хотите, в общем-то)
- ваша идея

Прошедшие соревнования

- EmoContext, и вообще SemEval 2019
- Kaggle Quora
- toxic messages in Kaggle

Где искать датасеты

- Kaggle datasets
- The Best 25 Datasets for Natural Language Processing
- nlp-datasets
- ещё ML датасеты
- Google datasets (beta)
- и вообще, Google ;)