

Neural Networks

Введение

Про дипломную работу

В этой серии постов я хочу разобрать нейросети под микроскопом.

Нейросети - это универсальный солдат, решающий задачи классификации, кластеризации и даже регрессии.

С задачей классификации связан большой раздел машинного обучения, называемый **распознаванием образов**.

Про Deep Learning

Про CV, NLP, распознавание и синтез звуков и прочие задачи распознавания

Хорошо обученная нейросеть могла притворяться любым алгоритмом из этой статьи, а зачастую даже работать точнее. Такая универсальность сделала их дико популярными.

Чем Deep Learning отличается от классического машинного обучения?

Здесь написано: https://www.wikiwand.com/ru/Глубокое_обучение

Подробнее про Feature Learning: https://www.wikiwand.com/en/Feature_learning

Что ты такое?

Машинное обучение 8. Intro to Deep Learning

Ссылка на материалы занятия: https://github.com/girafe-ai/ml-mipt/tree/basic_s20/week0_08_Intro_to_DL Семинар:
<https://www.youtube.com/watch?v=wgWfAMhw5ik>

G https://youtu.be/wgWfAMhw5ik?list=PL4_hYwCyhAvZyW6qS58x4uElZgAkMVUvj



Машинное обучение 9. Optimization and regularization in Deep Learning

Ссылка на материалы занятия: https://github.com/girafe-ai/ml-mipt/tree/basic_s20/week0_09_Optimization_and_Regularization_in_DL Семинар:
<https://youtu.be/kBIMPomCGGA> 1:34 - повторение 7:19 - SGD 11:01 - momentum
G https://youtu.be/XWAvufvk2M?list=PL4_hYwCyhAvZyW6qS58x4uElZgAkMVUvj



Нейросеть - это куча связанных между собой нейронов.

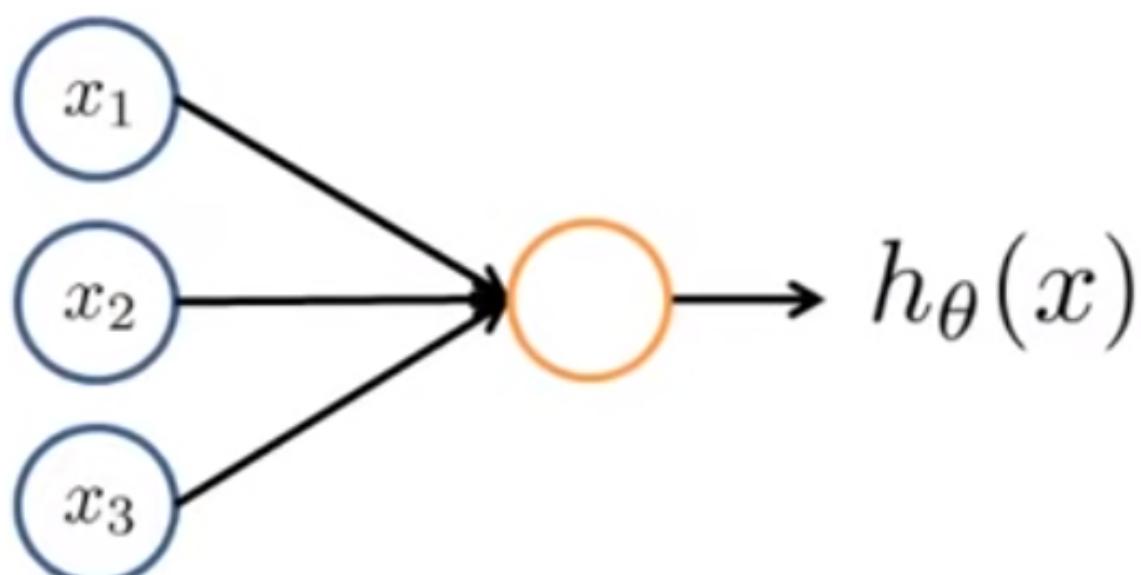
Нейрон - это просто функция, принимающая множество чисел и отдающая одно число на выходе.

Нейроны связаны между собой, причём каждая связь имеет определенный **вес**, который можно понимать как силу связи: число, проходящее через связь, умножается на вес этой связи. Веса - это параметры модели. Их нужно искать, подобрав оптимальные. Для этого мы обучаем нейросеть, как и любую модель.

Вспомним, например, парную линейную регрессию: там мы обучали модель с целью найти два параметра - θ_0 и θ_1 (slope и intercept), по которым можно было нарисовать прямую, наилучшим образом описывающую данные. Здесь с параметрами всё идеально так же. Когда нейросеть обучится, эти веса примут конкретные значения, и тогда модель будет способна предсказывать результат на новых данных, используя всю эту кучу взвешенных связей.

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

(нарисовать веса)



В каждом нейроне определена своя функция активации (что это?)

https://ru.wikipedia.org/wiki/Функция_активации

В качестве своего параметра (x) она принимает сумму всех чисел, переданных от других нейронов (с учётом весов).

Логистическая функция активации - одна из наиболее популярных.

$$h_{\Theta}(x) = \frac{1}{1 + e^{-\Theta^T x}}$$

Пёсики нравятся? Смотрим пример:

(нарисовать такую же картинку, как и первую, но с числами, подписать где связь сильная и слабая, на выходе 1 или 0, в нейроне логистическую регрессию)

В итоге все числа суммируются в нейроне, результат - аргумент функции активации.

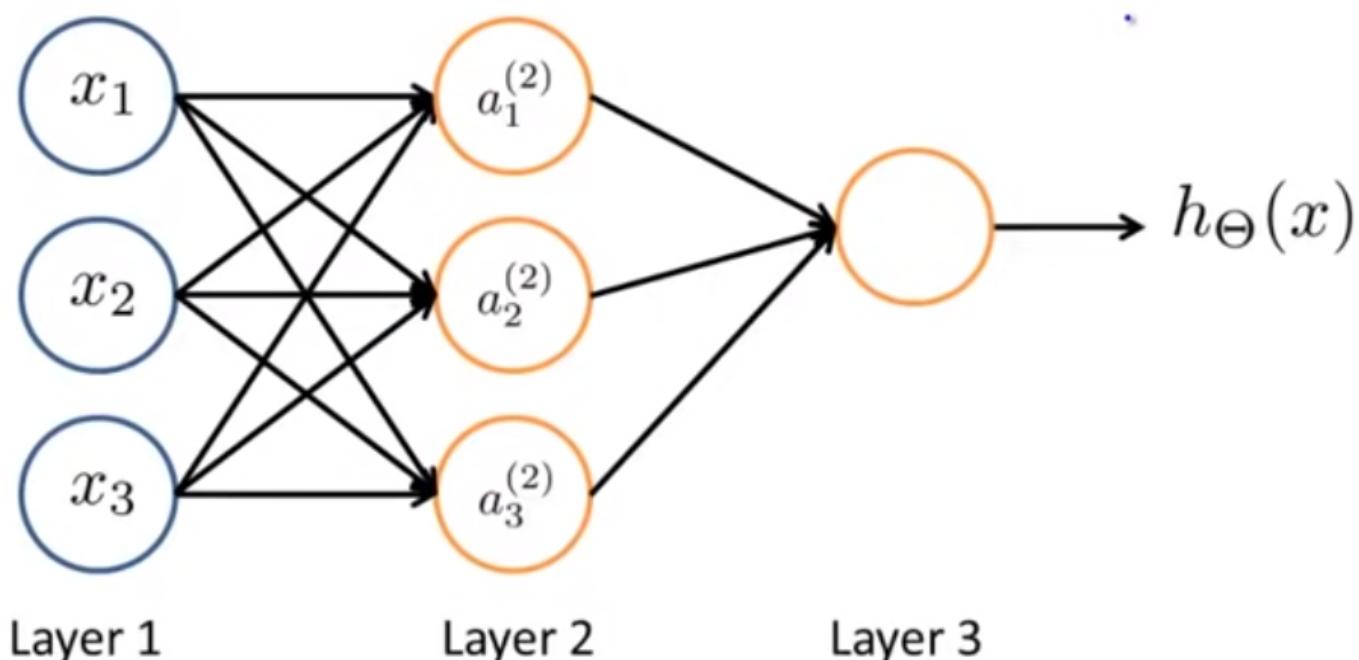
Подставляем в функцию активации и получаем число

(показать, где лежит полученное число на графике сигмоиды)

$g(\dots) \approx 1$, что означает активацию нейрона (обученная на собаках нейросеть отличила собаку от не-собаки, например).

В нейросетях нейроны расположены **слоями**. Каждый нейрон связан только с нейронами из предыдущего и следующего слоёв. На примере выше слоя всего два - входной и выходной. Все слои между крайними называются **скрытыми**.

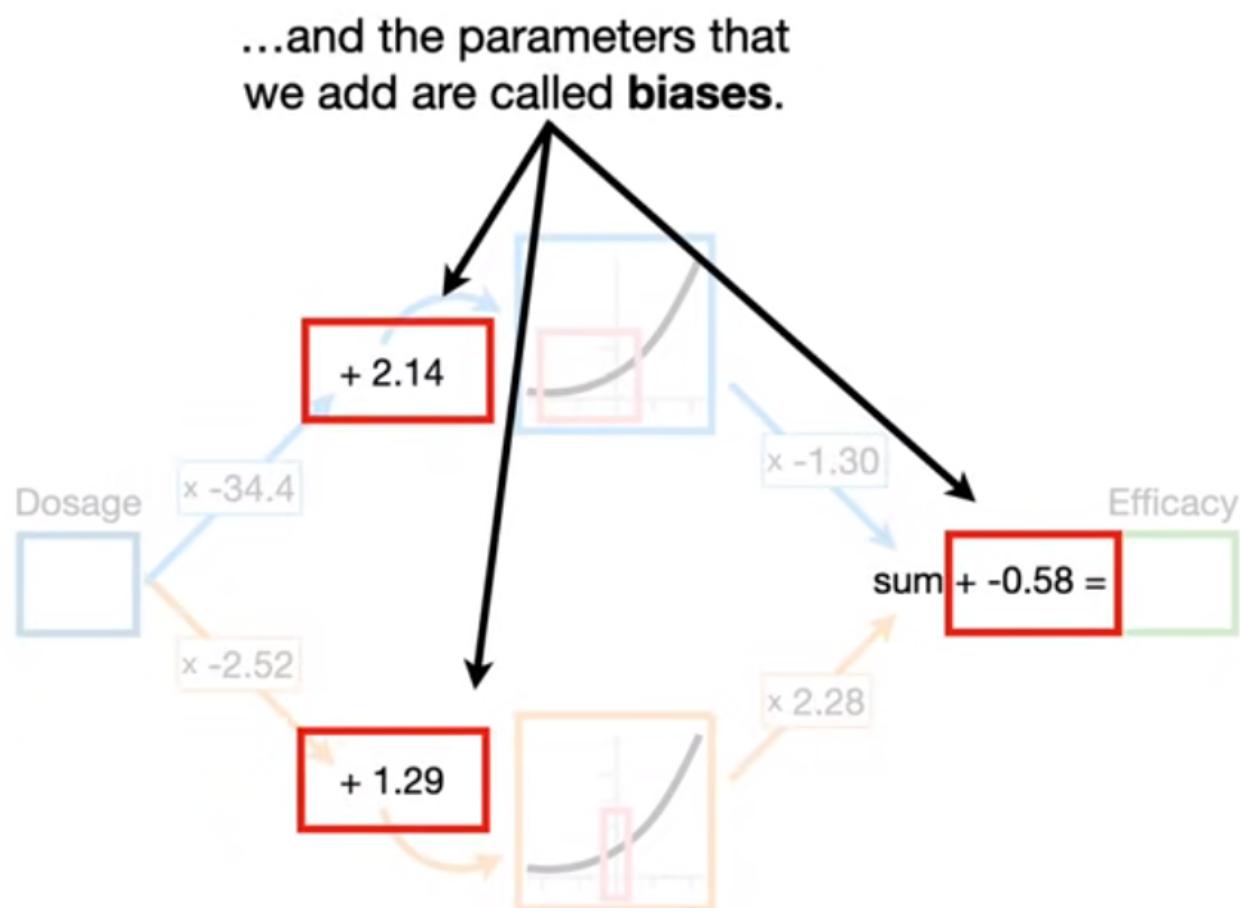
Добавим один скрытый слой к нашей нейросети:



В скобках указан номер слоя.

Также на каждом слое обычно задано смещение (bias). Для схемы нейросети выше - это x_0 и a_0 .

<https://stackoverflow.com/questions/2480650/what-is-the-role-of-the-bias-in-neural-networks>



<https://www.youtube.com/watch?v=aircAruvnKk>

https://ru.wikipedia.org/wiki/Нейронная_сеть

Опять математика

Что происходит в картинке математически:

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

g - это функция активации.

Разумеется, складывать всё это вручную мы не будем, поэтому перейдём к векторной записи.

Для этого аргумент g обозначим как z :

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

А ведь z - это сумма x с весами тетта. Получаем:

$$\begin{aligned} z^{(2)} &= \Theta^{(1)}x \\ a^{(2)} &= g(z^{(2)}) \end{aligned}$$

Вектор x ещё обозначают как $a^{(1)}$.

Не забываем смещение и представляем следующий слой в векторной форме:

Add $a_0^{(2)} = 1.$

$$z^{(3)} = \Theta^{(2)}a^{(2)}$$

$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

Один в один формула логистической регрессии, прошу заметить. Мы просто считаем её для каждого слоя, только теперь вместо обычных признаков у нас значения, пришедшие из

прошлых слоёв (a_1, a_2, a_3).

Этот процесс последовательного учёта весов на каждом слое иногда называют **прямым распространением** (forward propagation). Итераций получаем на единицу меньше количества слоёв нейросети, так как в первом слое ничего считать не нужно.

Таким образом, обучаясь, нейросеть на каждом новом слое создаёт собственные признаки, а веса связей позволяют ей запомнить информацию, повторив (на новых данных) путь по нужным нейронам к выходному слою.

Нейроны и связи можно расставить по-разному. Поэтому придумали различные архитектуры. Описанная выше архитектура называется **перцептроном** - нейросетью прямого распространения, то есть в которой все связи направлены строго от входных нейронов к выходным.

Перцептрон

Виды функции активации

Немного хочется поговорить про то, какие и в каких случаях лучше применять те или иные функции активации.

Отличия

Сигмоида

ReLU

Устраняет отрицательные значения

Гиперболический тангенс

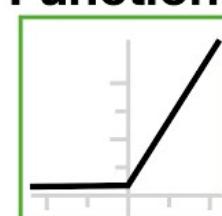
...

Neural Networks Pt. 3: ReLU In Action!!!

The ReLU activation function is one of the most popular activation functions for Deep Learning and Convolutional Neural Networks. However, the function itself...

YouTube <https://youtu.be/68BZ5f7P94E>

The ReLU Activation Function...



$f(x) = \max(0, x)$
...Clearly
Explained!!!

Neural Networks Part 5: ArgMax and SoftMax

When your Neural Network has more than one output, then it is very common to train with SoftMax and, once trained, swap SoftMax out for ArgMax. This video gi...

 <https://youtu.be/KpKog-L9veg>

ArgMax and SoftMax for Neural Networks...



...Clearly Explained!!!

The SoftMax Derivative, Step-by-Step!!!

Here's step-by-step guide that shows you how to take the derivatives of the SoftMax function, as used as a final output layer in a Neural Networks. 

NOTE: Wh...

 <https://youtu.be/M59JEIEPgIg>

The SoftMax Derivative...

$$\frac{d "p_{Setosa}"}{d \text{Raw}_{Setosa}} = "p_{Setosa}" \times (1 - "p_{Setosa}")$$



...Step-by-Step!!!

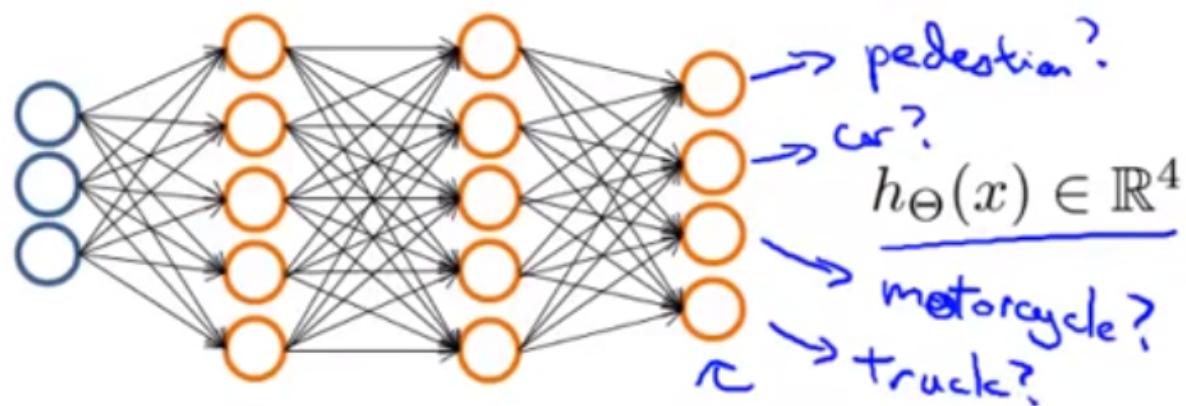
Больше нейронов!!!

Если вы уже додумались нахреначить скрытых слоёв в перцептроне, то поздравляю, вы получили архитектуру, называемую **многослойным перцептроном** (MLP).

Если ещё к выходному слою добавить несколько нейронов, то уже можно будет использовать нейросеть для решения серьёзных задач.

В нашем примере с однослойным перцептроном нейросеть могла лишь отличить собаку от не-собаки. В случае, если на выходном слое несколько нейронов, мы уже можем обучить нейросеть распознавать, например, породы собак. Каждый нейрон выходного слоя будет отвечать за конкретный класс. Какой нейрон на выходе сработал (выдал 1) - к тому классу и будет отнесён объект.

Multiple output units: One-vs-all.



when pedestrian when car when motorcycle

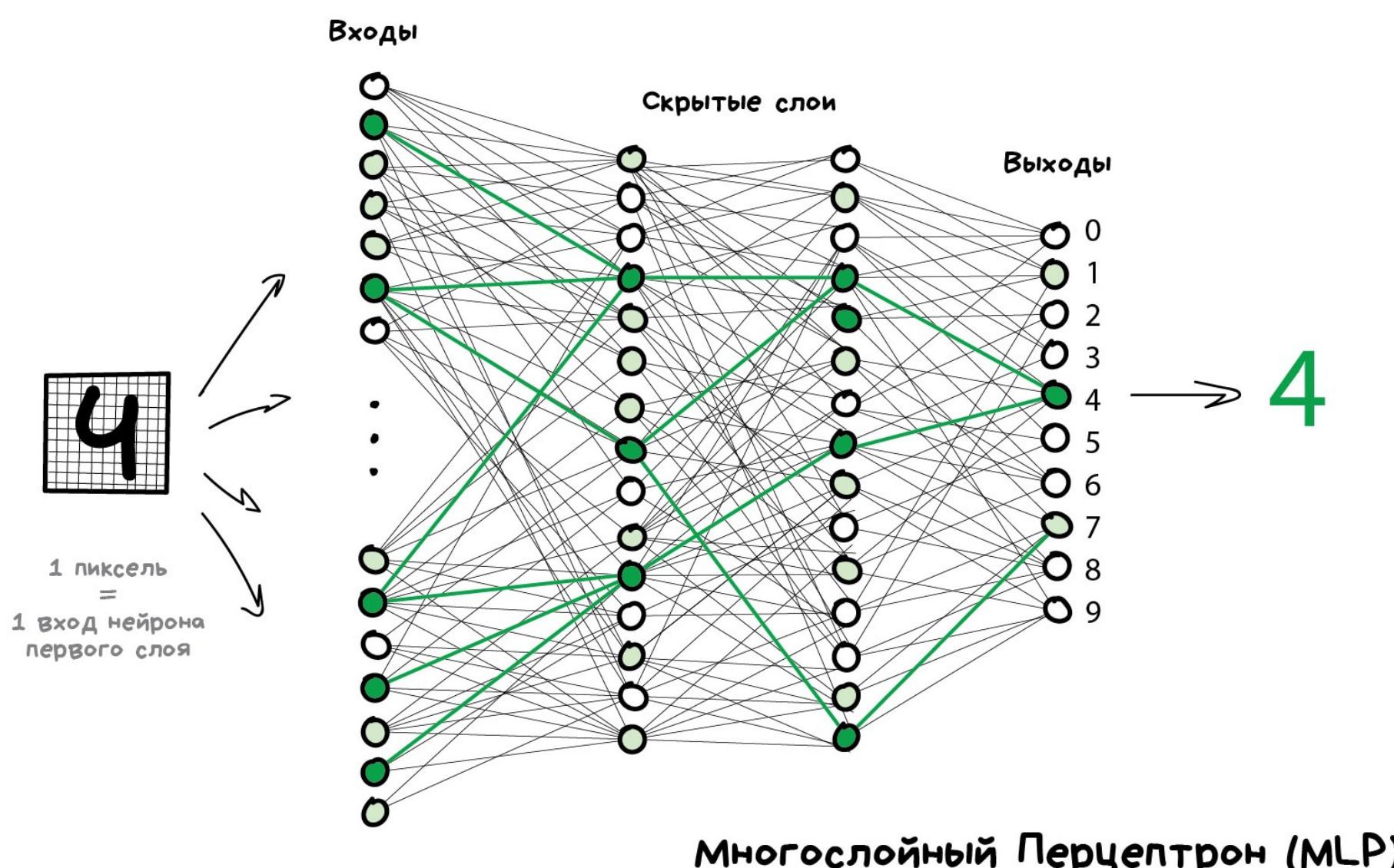
Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

$y^{(i)}$ one of $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$
 pedestrian car motorcycle truck

~~Previously~~
 ~~$y \in \{1, 2, 3, 4\}$~~
 ~~$h_{\Theta}(x^{(i)}) \approx y^{(i)}$~~

Если нафигачить достаточноное количество слоёв и правильно расставить веса в такой сети, получается следующее — подав на вход, скажем, изображение написанной от руки цифры 4, чёрные пиксели активируют связанные с ними нейроны, те активируют следующие слои, и так далее и далее, пока в итоге не загорится самый выход, отвечающий за четвёрку.

Результат достигнут.

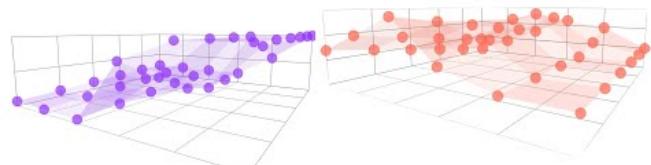


Neural Networks Pt. 4: Multiple Inputs and Outputs

So far, this series has explained how very simple Neural Networks, with only 1 input and 1 output, function. This video shows how these exact same concepts...
g...

 <https://youtu.be/83LYR-1IcjA>

Neural Networks: multiple inputs and outputs...



...Step-by-step!!!

Забавно, но мы слишком тупые для понимания происходящего в скрытых слоях. Слишком этот процесс непредсказуемый, слишком много в нём переменных.

Обучение нейросети

Суть обучения модели - оптимизация параметров (в случае с нейросетями этиими параметрами являются веса). Для обучения нейростей используется **метод обратного распространения ошибки** (backpropagation). Это апгрейд алгоритма градиентного спуска. А правильнее говоря — метод вычисления градиента.

Тут нужно вспомнить, что у нас же есть данные — примеры «входов» и правильных «выходов». Будем показывать нейросети рисунок той же цифры 4 и говорить «подстрой свои веса так, чтобы на твоём выходе при таком входе всегда загоралась четвёрка».

Сначала все веса просто расставлены случайно, мы показываем сети цифру, она выдаёт какой-то случайный ответ (весов-то нет), а мы сравниваем, насколько результат отличается от нужного нам. Затем идём по сети в обратном направлении, от выходов ко входам, и говорим каждому нейрону — так, ты вот тут зачем-то активировался, из-за тебя всё пошло не так, давай ты будешь чуть меньше реагировать на вот эту связь и чуть больше на вон ту, ок?

Через тысяч сто таких циклов «прогнали-проверили-наказали» есть надежда, что веса в сети откорректируются так, как мы хотели. Научно этот подход называется Backpropagation или «Метод обратного распространения ошибки». Забавно то, что чтобы открыть этот метод понадобилось двадцать лет. До него нейросети обучали как могли.

Второй мой любимый видос более подробно объясняет весь процесс, но всё так же просто, на пальцах.

Neural Networks Pt. 2: Backpropagation Main Ideas

Backpropagation is the method we use to optimize parameters in a Neural Network. The ideas behind backpropagation are quite simple, but there are tons of det...

 <https://youtu.be/IN2XmBhILt4>

Backpropagation for Neural Networks...



Backpropagation Details Pt. 1: Optimizing 3 parameters simultaneously.

The main ideas behind Backpropagation are super simple, but there are tons of details when it comes time to implementing it. This video shows how to optimize...

 <https://youtu.be/lyn2zdALii8>

Backpropagation Details...

$$\frac{d \text{SSR}}{d w_3} = \frac{d \text{SSR}}{d \text{Predicted}} \times \frac{d \text{Predicted}}{d w_3}$$
$$\frac{d \text{SSR}}{d w_4} = \frac{d \text{SSR}}{d \text{Predicted}} \times \frac{d \text{Predicted}}{d w_4}$$
$$\frac{d \text{SSR}}{d b_3} = \frac{d \text{SSR}}{d \text{Predicted}} \times \frac{d \text{Predicted}}{d b_3}$$

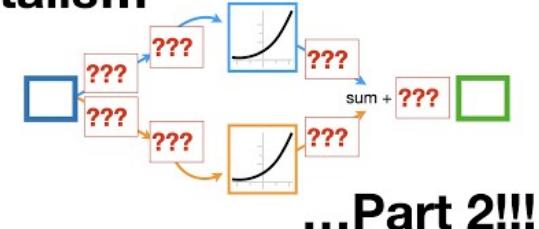
...Part 1!!!

Backpropagation Details Pt. 2: Going bonkers with The Chain Rule

This StatQuest picks up right here Part 1 left off, and this time we're going to go totally bonkers with The Chain Rule and optimize every single parameter i...

 <https://youtu.be/GKZoOHXGcLo>

Backpropagation Details...



Опять математика

Для более формального объяснения метода сначала стоит ввести функцию потерь для нейросети. Она очень похожа на функцию кросс-энтропии, я даже поставлю их рядом. И добавлю регуляризацию:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left(y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right) + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_\theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\theta(x^{(i)}))_k) \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\theta_{ji}^{(l)})^2$$

L - количество слоёв нейросети

s_l - количество нейронов на слое l

В плейлисте курса от Andrew Ng: Lecture 9.1 - 9.8

https://www.youtube.com/watch?v=0twSSFZN9Mc&list=PLLssT5z_DsKh9vYZkQkYNWcItqhIRJLN&index=50

<https://course.fast.ai/>

https://ru.wikipedia.org/wiki/Метод_обратного_распространения_ошибки

<https://www.youtube.com/watch?v=IHZwWFHWa-w>

https://youtu.be/lIg3gGewQ5U?list=PL_h2yd2CGtBHEKwEH5igTZH85wLS-eUzv

https://youtu.be/tleHLnjs5U8?list=PL_h2yd2CGtBHEKwEH5igTZh85wLS-eUzv

Про переобучение и недообучение

Архитектуры и модификации

Помимо перцептрона, на практике используются и другие архитектуры, каждая из которых эффективно показывает себя в определенных задачах. Их очень много, поэтому я расскажу о самых важных.

<https://habr.com/ru/company/oleg-bunin/blog/340184/>

Convolutional Neural Network (CNN)

Конволюшки применяют для распознавания графических образов.

Машинное обучение 11. Convolutional neural networks

Ссылка на материалы занятия: https://github.com/girafe-ai/ml-mipt/tree/basic_s20/week0_11_CNN Продолжение курса:

[https://www.youtube.com/playlist?
G https://youtu.be/6_vFNBwfK28?list=PL4_hYwCyhAvZyW6qS58x4uElZgAk](https://www.youtube.com/playlist?list=PL4_hYwCyhAvZyW6qS58x4uElZgAk)



Neural Networks Part 8: Image Classification with Convolutional Neural Networks

One of the coolest things that Neural Networks can do is classify images, and this is often done with a type of Neural Network called a Convolutional Neural ...

 <https://youtu.be/HGwBXDKFk9I>

Deep Learning Image Classification...



Свёрточные Сети

Проблема с изображениями всегда была в том, что непонятно, как выделять на них признаки. Картинки приходилось размечать руками, объясняя машине, где у котика на фотографии ушки, а где хвост. Пришлось заставить машину самой учиться искать эти признаки, составляя из каких-то базовых линий.

Будем делать так: для начала разделим изображение на блоки 8x8 пикселей и выберем какая линия доминирует в каждом — горизонтальная [-], вертикальная [] или одна из диагональных [/]. Могут и две, и три, так тоже бывает, мы не всегда точно уверены.

На выходе мы получим несколько массивов палочек, которые по сути являются простейшими признаками наличия очертаний объектов на картинке. По сути это тоже картинки, просто из палочек. Значит мы можем вновь выбрать блок 8x8 и посмотреть уже, как эти палочки сочетаются друг с другом. А потом еще и еще.

Такая операция называется свёрткой, откуда и пошло название метода. Свёртку можно представить как слой нейросети, ведь нейрон — абсолютно любая функция.



Свёрточная Нейросеть (CNN)

Когда мы пропускаем через нашу нейросеть кучу фотографий котов, она автоматически расставляет большие веса тем сочетаниям из палочек, которые увидела чаще всего. Причём неважно, это прямая линия спины или сложный геометрический объект типа мордочки — что-то обязательно будет ярко активироваться.

На выходе же мы поставим простой перцептрон, который будет смотреть какие сочетания активировались и говорить кому они больше характерны — кошке или собаке.

Как работают сверточные нейронные сети | #13 нейросети на Python

Что из себя представляют сверточные нейросети, их архитектура. Операции свертки (сверточные слои), операции Pooling: MaxPooling, MinPooling, AveragePooling. Их смысл. Пример архитектуры сверточной нейронной сети.

G <https://www.youtube.com/watch?v=CEUNTRdLhKk>



Как работают сверточные нейронные сети (архитектура)

Зачем нужна свертка размером 1x1?

Recurrent Neural Network (RNN)

Машинное обучение 10. Recurrent Neural Networks and Language Models

G https://www.youtube.com/watch?list=PL4_hYwCyhAvZyW6qS58x4uElZgAkMVUvj&v=TkFBozjO72Y&feature=youtu.be



Машинный перевод, синтез речи

На них решают все задачи, связанные с последовательностями — голосовые, текстовые или музыкальные. Каждое слово или звук — как бы самостоятельная единица, но которая зависит от предыдущих.

Рекуррентные Сети

Достаточно легко обучить сеть произносить отдельные слова или буквы. Берём кучу размеченных на слова аудиофайлов и обучаем по входному слову выдавать нам последовательность сигналов, похожих на его произношение. Сравниваем с оригиналом от диктора и пытаемся максимально приблизиться к идеалу. Для такого подойдёт даже перцептрон.

Вот только с последовательностью опять беда, ведь перцептрон не запоминает что он генерировал ранее. Для него каждый запуск как в первый раз. Появилась идея добавить к каждому нейрону память. Так были придуманы рекуррентные сети, в которых каждый нейрон запоминал все свои предыдущие ответы и при следующем запуске использовал их как дополнительный вход. То есть нейрон мог сказать самому себе в будущем — эй, чувак, следующий звук должен звучать повыше, у нас тут гласная была (очень упрощенный пример).

Была лишь одна проблема — когда каждый нейрон запоминал все прошлые результаты, в сети образовалось такое дикое количество входов, что обучить такое количество связей становилось нереально.

Когда нейросеть не умеет забывать — её нельзя обучить (у людей та же фигня).

Сначала проблему решили в лоб — обрубили каждому нейрону память. Но потом придумали в качестве этой «памяти» использовать специальные ячейки, похожие на память компьютера или регистры процессора. Каждая ячейка позволяла записать в себя циферку, прочитать или сбросить — их называли ячейки долгой и краткосрочной памяти (LSTM).

Когда нейрону было нужно поставить себе напоминалку на будущее — он писал это в ячейку, когда наоборот вся история становилась ненужной (предложение, например, закончилось) — ячейки сбрасывались, оставляя только «долгосрочные» связи, как в классическом перцептроне. Другими словами, сеть обучалась не только устанавливать текущие связи, но и ставить напоминалки.

Autoencoder

Автокодировщики позволяют использовать backprop для обучения без учителя

Автоэнкодеры

Generative Adversarial Network (GAN)

[https://ru.wikipedia.org/wiki/Генеративно-состязательная сеть](https://ru.wikipedia.org/wiki/Генеративно-состязательная_сеть)

Residual Neural Network (ResNet)

<https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035>

Модификация CNN

https://youtu.be/M_ukL_cDvWg

AlexNet

Inception

Gated Recurrent Unit (GRU)

[https://www.youtube.com/playlist?list=PLA0M1Bcd0w8yv0XGiF1wjерjSZVSrYbjh](https://www.youtube.com/playlist?list=PLA0M1Bcd0w8yv0XGiF1wjerjSZVSrYbjh)

http://www.machinelearning.ru/wiki/index.php?title=Категория:Нейронные_сети

Neural Networks Part 6: Cross Entropy

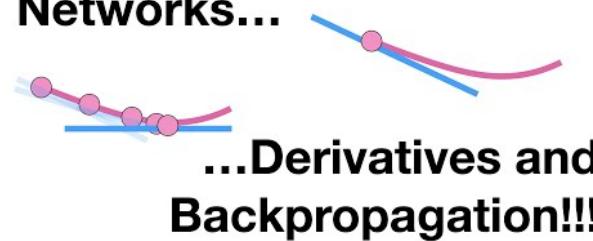
When a Neural Network is used for classification, we usually evaluate how well it fits the data with Cross Entropy. This StatQuest gives you an overview of ...

Neural Networks Part 7: Cross Entropy Derivatives and Backpropagation

Here is a step-by-step guide that shows you how to take the derivative of the Cross Entropy function for Neural Networks and then shows you how to use that d...

 <https://youtu.be/xBEh66V9gZo>

Cross Entropy for Neural Networks...



<https://www.youtube.com/watch?v=L35fFDpwIM4>

Data Science from Scratch - главы 18 и 19, стр. 292, 306

file:///home/lenferdetroud/Documents/Data%20Science%20from%20Scratch%20(2nd%20Edition).r

Kaggle Course: Intro to Deep Learning (не забыть выложить задания на GitHub)

<https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning>

Специализация Deep Learning:

Курс 1 (проходил давно, нужно быстро повторить материал и код заданий на GitHub)

Kypc 2

Kypc 3

Kypc 4

Kypc 5

https://

Не забывать выкладывать задания на GitHub

Разделы Нейросети и Глубокое обучение: https://neerc.limo.ru/wiki/index.php?title=Машинное_обучение#.D0.9A.D0.BB.D0.B0.D1.81.D1.81.D0.B8.D1.84.D0.B8.D0.BA.D0.B0.Г

Мои отчет по практике за 6 семестр

Ко всем архитектурам красивые картинки

Более экзотические архитектуры:

<https://habr.com/ru/company/wunderfund/blog/313696/>

- Глубокие регрессионные нейронные сети.

Artificial neural network

Autoencoder · Cognitive computing · Deep learning · DeepDream · Multilayer perceptron · RNN (LSTM · GRU · ESN · reservoir computing) · Restricted Boltzmann machine · GAN · SOM · Convolutional neural network (U-Net) · Transformer (Vision) · Spiking neural network · Memtransistor · Electrochemical RAM (ECRAM)