**Нейронные сети**

Добрый день, меня зовут Настя, и я хочу рассказать вам о нейронных сетях и немного о моей исследовательской работе в этой области.

Искусственные нейросети были созданы как подобие биологических.

Биологическая нейронная сеть— совокупность связанных [нейронов](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD), выполняющих специфические функции. Нейроны переносят электрический сигнал от дендритов к терминалам аксона: отросткам нейрона. Дальше сигнал переходит к другому нейрону и все повторяется. Именно таким образом ваше тело чувствует свет, звуки, прикосновения, тепло и так далее. Сигналы принимаются из внешней среды специальными сенсорными нейронами. Затем эти сигналы передаются через нервную систему в мозг, который почти полностью состоит из нейронов.

Биологические нейросети способны решать сложнейшие задачи несмотря на то, что они работают медленнее и имеют меньше памяти, чем современные компьютеры, также они не позволяют зашумленным сигналам серьезно повлиять на результаты в сравнении с традиционными компьютерами. Поэтому попытки создать совершенную нейросеть продолжаются.

ИНС представляет собой [систему](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) соединённых и взаимодействующих между собой простых процессоров ([искусственных нейронов](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD)), передающих сигналы как и БНС.

Искусственный нейрон получает на вход какое-то значение с весом, позже веса суммируются в сумматоре (Результатом сумматора: сумма входных сигналов, умноженных на их веса.) и проходят через функцию активации, которая определяет, пройдет значение на выход или нет. По сути, биологический нейрон работает схоже. Например, условно у вас в мозгу существует нейрон, к нему на вход поступают сигналы о ваших потребностях: голоде, сне, жажде, общении и т.д. Допустим, сейчас вам хочется есть больше, чем спать. Таким образом, вес входного значения голода был бы больше, чем вес сна, и вы бы подумали сначала о еде, и потом только об отдыхе. Тем не менее, вы учитываете, что спать вам тоже хочется, взвешивая сигналы, как на сумматоре, и как бы проводя сигнал через функцию активации, вы понимаете, что сейчас пойдете на кухню.

Нейронные сети не [программируются](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), они [обучаются](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными [алгоритмами](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC). Технически обучение заключается в нахождении коэффициентов связей между нейронами (т.е. весов). В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять [обобщение](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5).

С целью обучить сеть создаются обучающие выборки - набор входных сигналов, по которым происходит обучение сети. Существует два подхода к классическому обучению, дающее разные результаты: обучение с учителем и самостоятельное.

**Обучение с учителем** (нахождение верного ответа из предложенной выборки)— вид обучения сети, при котором ее веса меняются так, чтобы ответы сети минимально отличались от уже готовых правильных ответов.

**Обучение без учителя** (классификация)— вид обучения сети, при котором сеть самостоятельно классифицирует входные сигналы. Правильных сигналов нет.

Теперь немного об областях применения нейросетей:

Во-первых, нейросети применяются в распознавании образов, лиц, обработки фото и видео и классификации полученных данных. Конкретные примеры - это Apple Face ID (используемый для аутентификации пользователя), Google Self-Driving Car (который может передвигаться без участия [человека](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C)), и т.д.

Во-вторых, нейросети применяется в сфере безопасности, анализ данных позволяет следить за определенными закономерностями в хранении информации в облачных базах данных. Таким образом, компания Deep Instinct разработала программу, способную с высокой точностью определять зараженные файлы.

В-третьих, нейросети используются в предсказании выходных данных на основе входных. Это часто применяется в экономике и финансах, так, например, алгоритм Yandex Data Factory способен предсказывать влияние промоакций на объем продаж товаров. Прогнозирование также успешно нашло применение в сфере развлечений и игр. Итак, одна программа в рамках чемпионата The International 2017 победила нескольких профессиональных игроков в Dota 2 с высоким уровнем игры.

Вот ещё немного примеров: приложение Cardiogram работает на Apple Watch и собирает статистику сердечных сокращений для выявления аритмии, а MedyMatch позволяет быстро принимать решения при инсульте, когда каждая минута на счету, система [нейронного машинного перевода](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4) (NMT), разработанная компанией Google, которая использует [искусственную нейронную сеть](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) для повышения беглости и точности перевода.

У нейросетей есть много вариантов использования, поэтому они становятся всё популярнее в последнее время. И меня это заинтересовало. Я написала небольшую программу. Сначала программа получит на вход картинки всех цифр, и картинки искаженной цифры 9, и после перевода картинок в сигналы и обработки их нейросетью, мы посмотрим, как нейросеть научилась распознавать цифру 9. Те её задача видеть 9 даже в искаженной 9.

Для начала переведем цифры в сигналы, это можно сделать с помощью создания картинок цифр на клеточном поле 3 на 5. Закрашенные клетки – 1, не закрашенные – 0.

Для начала я импортировала модуль для работы со случайными числами.

1. **import** random

Создаем обучающую выборку (напомню вам, что это набор сигналов, по которым и производится обучение)

1. # Цифры (Обучающая выборка)
2. num0 = list('111101101101111')
3. num1 = list('001001001001001')
4. num2 = list('111001111100111')
5. num3 = list('111001111001111')
6. num4 = list('101101111001001')
7. num5 = list('111100111001111')
8. num6 = list('111100111101111')
9. num7 = list('111001001001001')
10. num8 = list('111101111101111')
11. num9 = list('111101111001111')

Функция list(\*) позволяет создать список (массив), состоящий из отдельных символов, на которые разбивается длинная строка.

Далее, для простоты я добавила все эти 10 цифр в список (для быстрого доступа к ним).

1. # Список всех вышеуказанных цифр
2. nums = [num0, num1, num2, num3, num4, num5, num6, num7, num8, num9]

Также записала 6 видов искаженной девятки в строковом формате.

1. # Виды цифры 9 (Тестовая выборка)
2. Num92 = list('011101111001111')
3. Num93 = list('110101111001111')
4. Num94 = list('110101111001110')
5. Num95 = list('111101111001110')
6. Num96 = list('111111111001111')

Теперь нужно создать список весов. Пусть все веса вначале будут равны 0(потому что очень сложно подобрать значения весов вручную). Так как есть 15 входов, то потребуется 15 связей.

1. # Инициализация весов сети
2. weights = [0 **for** i **in** range(15)]

Здесь я использовала генератор списков. Вместо этой записи можно использовать обычный цикл, повторяющийся 15 раз и присваивающий всем элементам нашего списка весов 0:

1. # Инициализация весов сети
2. weights = []
3. **for** i **in** range(15):
4. weights.append(0)

ТК я использовала пороговую функцию активации, то было необходимо установить какое-то число (порог). Если взвешенная сумма будет больше или равна ему, то сеть выдаст 1 (что означает, что она считает предоставленную цифру 9). Порог можно выбрать любой. Например 7.

1. # Порог функции активации
2. bias = 7

Далее я создала 3 вспомогательные функции, первая определяет является ли полученное на вход число 9, если оно им является и нейросеть сказала да, то ничего не меняем, если же она сказала нет, то мы увеличиваем веса, связанные с возбужденным входом (возбужденным входом считается вход, который получил черный пиксель, те 1). Потому что важность сигнала определяют именно веса. И увеличить эту важность в данном случае нам поможет 3 функция. А если полученное число не 9, но сеть распознала его как 9, то мы соответственно уменьшаем веса при помощи функции 2.

Здесь я создала простейшую функцию, которая считает взвешенную сумму и сравнивает ее с порогом. Фактически эта функция представляет собой единичный шаг работы нашей нейронной сети.

1. # Является ли данное число 9
2. **def** proceed(number):
3. # Рассчитываем взвешенную сумму
4. net = 0
5. **for** i **in** range(15):
6. net += int(number[i])\*weights[i]
7. # Превышен ли порог? (Да - сеть думает, что это 9. Нет - сеть думает, что это другая цифра)
8. **return** net >= bias

Результатом работы выражения **return** net >= bias этой функции может быть True (Правда), что означает 1 или False (Ложь), что означает 0.

Также есть еще две вспомогательные функции.

Первая функция вызывается, когда сеть считает за 9 неверную цифру (выход равен 1 при демонстрации не 9) и уменьшает на единицу все веса, связанные с возбужденным входами. Вход считатеся возбужденным, если он получил черный пиксель (то есть 1).

1. # Уменьшение значений весов, если сеть ошиблась и выдала 1
2. **def** decrease(number):
3. **for** i **in** range(15):
4. # Возбужденный ли вход
5. **if** int(number[i]) == 1:
6. # Уменьшаем связанный с ним вес на единицу
7. weights[i] -= 1

Я использовала функцию int(number[i]) для преобразования символа  '1' в цифру 1. Если этого не делать, то возникла бы ошибка **if** int(number[i]) == 1:, так как Python не умеет сравнивать символы (текст) с цифрами.

Вторая функция вызывается, если сеть не смогла распознать цифру 9 (выход равен 0 при демонстрации 9) и увеличивает на единицу все веса, связанные с возбужденными входами.

1. # Увеличение значений весов, если сеть ошиблась и выдала 0
2. **def** increase(number):
3. **for** i **in** range(15):
4. # Возбужденный ли вход
5. **if** int(number[i]) == 1:
6. # Увеличиваем связанный с ним вес на единицу
7. weights[i] += 1

Теперь посмотрим на обучение нейронной сети. Оно будет проходить в цикле с достаточно большим количеством повторений:

1. # Тренировка сети
2. **for** i **in** range(100000):
3. # Генерируем случайное число от 0 до 9
4. option = random.randint(0, 9)
5. # Если получилось НЕ число 9
6. **if** option != 9:
7. # Если сеть выдала True/Да/1, то уменьшаем вес
8. **if** proceed(nums[option]):
9. decrease(nums[option])
10. # Если получилось число 9
11. **else**:
12. # Если сеть выдала False/Нет/0, то показываем, что эта цифра - то, что нам нужно
13. **if** **not** proceed(num9):
14. increase(num9)

Далее выводим результаты обучения. Сначала выведем результирующие значения весов. После этого еще раз прогоним сеть по всем цифрам от 0 до 9, но уже без обучения. Просто чтобы удостоверится, что она обучилась. А в конце опробуем сеть на нашей тестовой выборке(на искаженных 9).

1. # Вывод значений весов
2. **print**(weights)
3. # Прогон по обучающей выборке
4. **print**("0 это 9? ", proceed(num0))
5. **print**("1 это 9? ", proceed(num1))
6. **print**("2 это 9? ", proceed(num2))
7. **print**("3 это 9? ", proceed(num3))
8. **print**("4 это 9? ", proceed(num4))
9. **print**("6 это 9? ", proceed(num6))
10. **print**("7 это 9? ", proceed(num7))
11. **print**("8 это 9? ", proceed(num8))
12. **print**("9 это 9? ", proceed(num9), '\n')
13. # Прогон по тестовой выборке
14. **print**("это 9? ", proceed(num9))
15. **print**("это 9? ", proceed(num92))
16. **print**("это 9? ", proceed(num93))
17. **print**("это 9? ", proceed(num94))
18. **print**("это 9? ", proceed(num95))
19. **print**("это 9? ", proceed(num96))

При каждом запуске программы сеть тренируется по разному, тк ей на вход поступают рандомные 10.000 чисел, поэтому и результаты тоже различны.

Первый результат работы моей сети оказался далеко не идеальным, но удовлетворительным, нейросеть распознала все цифры правильно, даже искаженные 9.

Теперь давайте вспомним, как мы представляли цифры: мы создавали картинку на клеточном поле и потом переводили её в строковый формат, закрашенные клетки становились 1, незакрашенные - 0. Так вот, нейросеть в конце показала , что она считает за 9. И я перевела эту строку в картинку, но, как вы видите, здесь есть цифры больше 1 или меньше 0. Таким образом она показывала свою уверенность в том, что это пиксель 9. Если число отрицательное, то этот пиксель точно не пиксель 9, если оно больше 1, то это место нейросеть считала частью 9. Поэтому переводя строку в картинку, красным цветом я показала отрицательные числа, а серым и черным положительные и 0. Здесь также мой второй и третий результаты, я наложила их друг на друга, и, если присмотреться, то это вполне похоже на 9.

**Правила Хебба** (*Hebb’s rule, Hebbian learning rule*) —два правила, составляющие алгоритм обучения персептронов для решения простейших задач классификации, когда входы могут быть равны только 0 или 1:

**1 Правило**. Если сигнал персептрона неверен и равен 0, то необходимо увеличить веса тех входов, на которые была подана единица.

**2 Правило**. Если сигнал персептрона неверен и равен 1, то необходимо уменьшить веса тех входов, на которые была подана единица.

Правила Хебба часто встречается в литературе, как **обучение с коррекцией ошибки**.