МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Проект

Моделирование искусственных нейронных сетей

Выполнил студент
группы № М3303
Слобода Полина Олеговна

САНКТ-ПЕТЕРБУРГ
2019

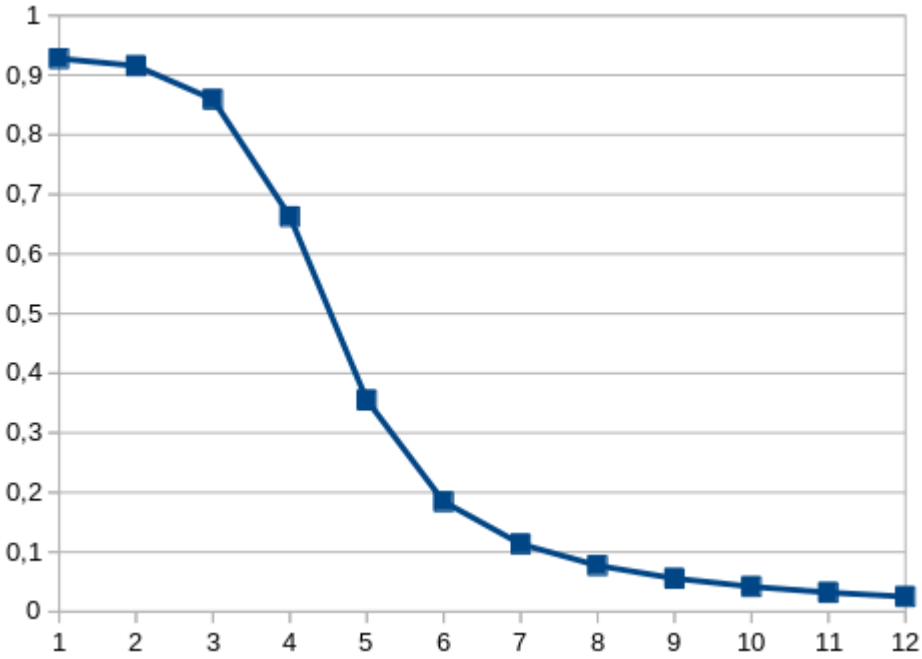
Содержание

Моделирование аналитически заданной функции

Я решала задачи моделирования искусственных нейронных сетей для решения двух задач. Первая из них — аналитически заданная функция:

$$x \in [-10, 10]$$
$$f_1(x) = \ln(1 + e^x)$$

Я решила смоделировать сеть с одной входной вершиной, котрая будет нормировать входное значение и передавать дальше функцию активации, двумя скрытыми слоями по 10 вершин и одним выходным (слои собирают данные от всех выходов предыдущего слоя, домножают их на соответствующие веса и суммируют, передавая дальше значение функции активации от полученного результата). В качестве функции активации мною была взята сигмоида.
Я передавала сети значения, и если сеть выдавала на них неподходящий результат (сравнивалась выходная степень возбуждения и отнормированное «нужное» значение), запускалась функция обучения (обучения с учителем). Я использовала алгоритм обратного распространения ошибки.



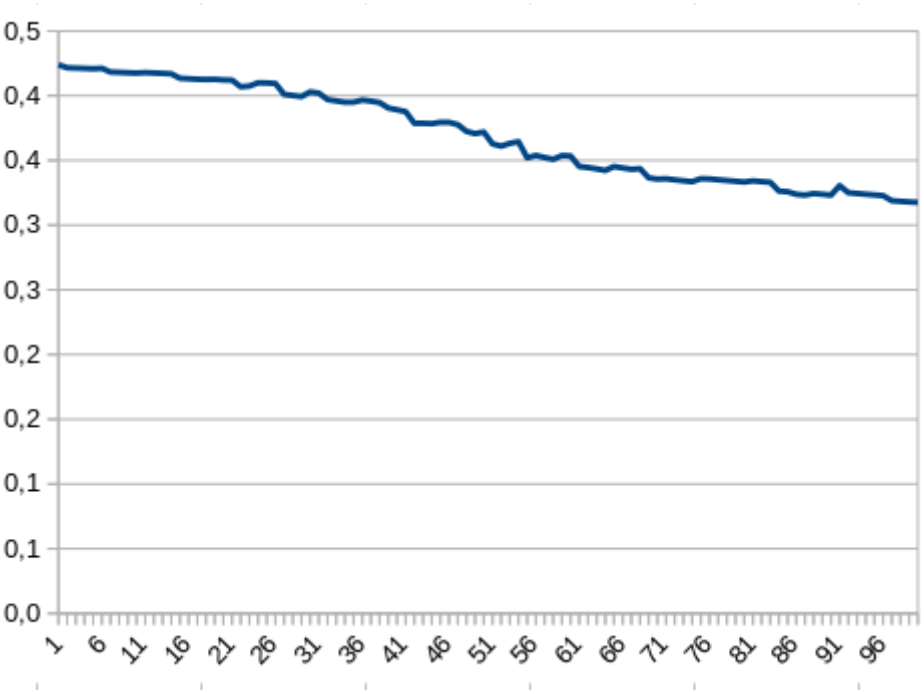
Это я скормила сети по очереди 12 нулей, на графике видно отклонение выходного значения возбужденности сети от идеального значения возбужденности (полученного, опять же, нормировкой моделируемой функции). Видно, что по мере продвижения сеть все больше и больше «убеждается», что нужно возвращать.

После этого я попробовала запустить сеть на различных значениях (случайных) и посмотреть, насколько она хорошо учится в таком случае. Ответ — не очень. Эффективность порядка 10 процентов (сравнение возбужденностей производила с точностью до 0.1, обучение и тестирование по 10 000 значений). Результат неутешительный от слова совсем. Попытки увеличить количество вершин в скрытых слоях мало помогают, результаты примерно такие же. Вообще если учесть, что с точностью до 0.1 сравниваются возбужденности (от 0 до 1), результаты очень похожи на простое гадание.

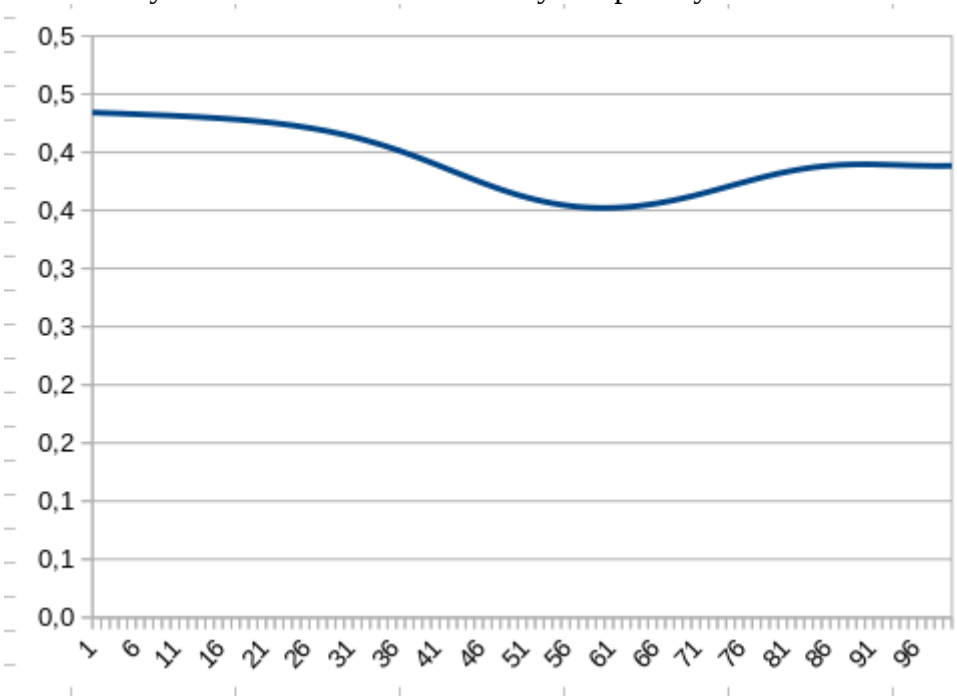
При попытках обучить сеть на равномерно возрастающем входном значениии я заметила, что сеть улавливает, какой примерно ответ от нее ждут и в процессе обучения часто угадывает. Но если неожиданно подсунуть входной параметр, выбивающийся из общей массы, она очень сильно ошибется.

Это наводит на мысли о переобучении. Все это время коэффициент n скорости обучения был равен единице. Уменьшаю до 0.5, обучаю с 20 вершинами в скрытом слое, все еще по 10 000 примеров для обучения и тестирования, все еще сравниваю с точностью до 0.1 степени возбуждения. 11%. Уменьшаю до 0.2 -> 12%. Пробуем 0.1 скорость обучения и увеличим количество данных для обучения до 20 000. И снова ничего не меняется. При любом раскладе получаю примерно 10% точности.

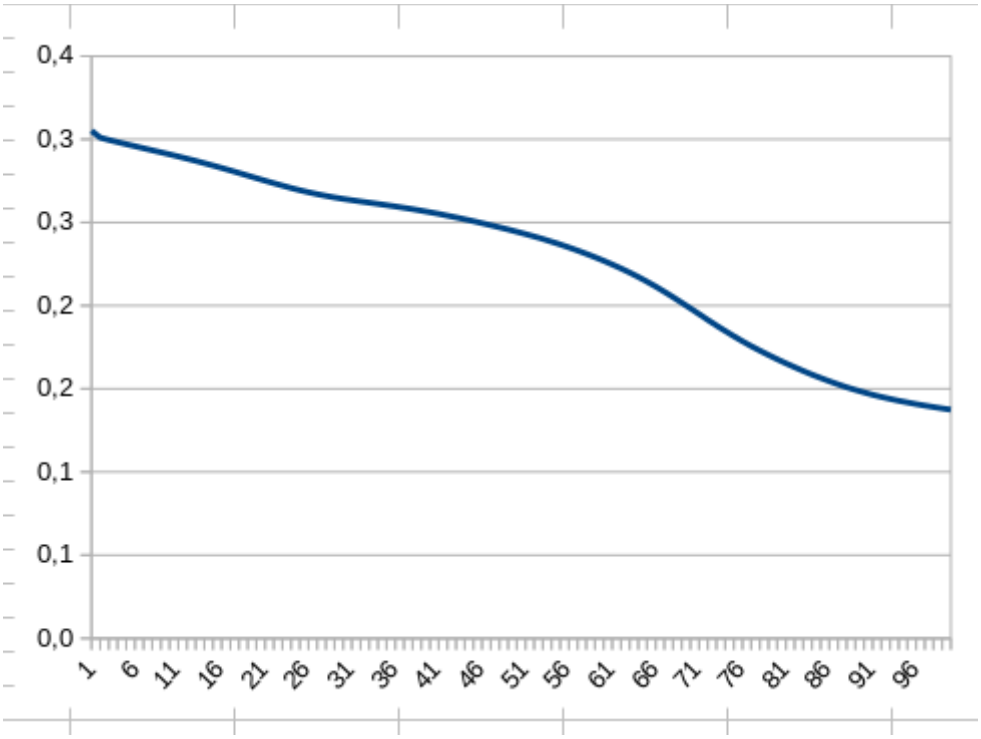
После стольких неудач и исследований я пришла к выводу, что что-то я глобально делаю не так. Во-первых, чтобы приблизить мою функцию, мне будет достаточно одного скрытого слоя — это существенное упрощение кода. Это в прнципе понятно интуитивно, функции же можно разложить в ряд. Это подтверждает и теорема, гласящая, что нейросети достаточно 1 скрытого слоя, чтобы аппроксимировать любую непрерывную функцию многоих переменных с любой точностью.
А самое главное — что сеть не может с первого раза научиться и что одни и че же данные ей необходимо давать несколько раз. Пожалуй, мой предыдущий метод обучения мог бы сработать, если бы я поставила нереально большое количество генераций.
Проверка моей догадки про обучение по эпохам- поставила 100 эпох, данные — равномерно проставленные от -10 до 10 точки с интервалом в 0.1. Также оценивала степени возбужденности с точностью до десятой, на графике видны средние квадраты отклонения реально полученных значений от ожидаемых значений для каждой эпохи.



При этом, кстати, не стоит забывать про коэффициент скорости обучения — при коэффициенте 1 я получила в какой-то момент такую картинку:

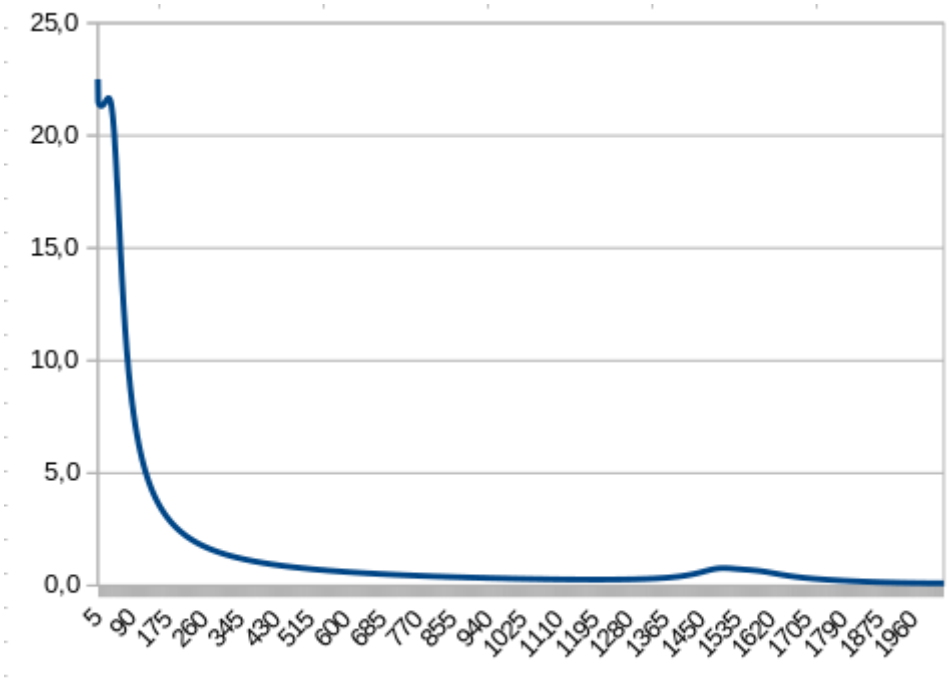


После некоторого момента обучение пошло в обратную сторону, что странно.



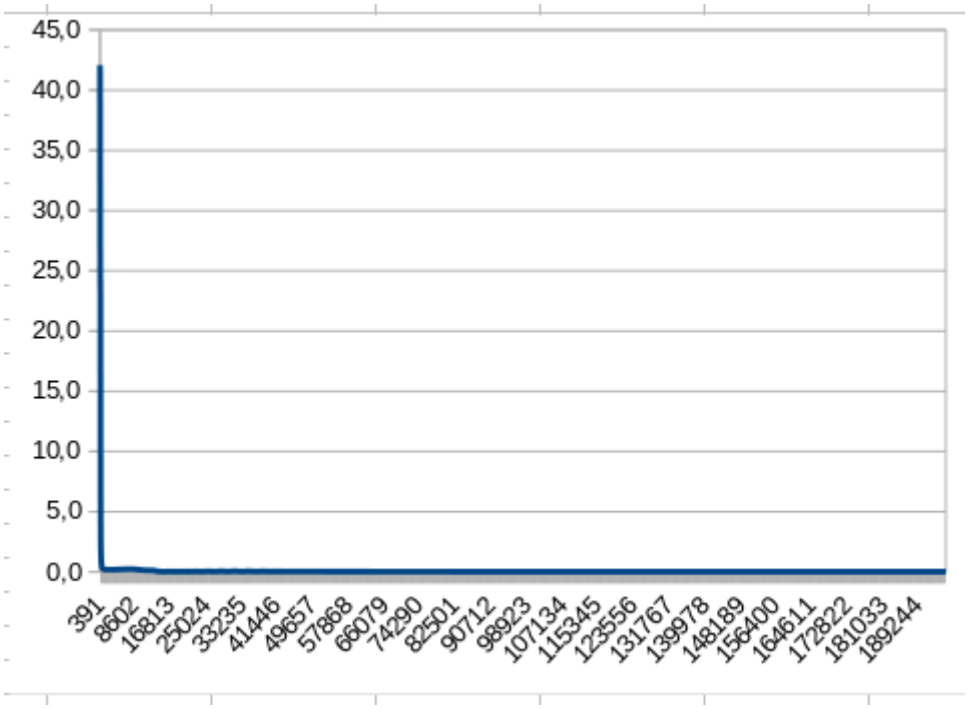
Тут я поправила скорость обучения на более мелкую. На самом деле есть подозрение, что это в общем случае может не помочь, и что в самой сети есть проблема.

Хорошо, тогда сделаем более красиво — перейдем на графике из возбужденностей в абсолютные единицы, ужесточим критерий успешности предсказания — разница в возбужденностях должна быть одной сотой, а эпох поставим столько, сколько потребуется, пока среднее значение квадрата разниц не станет, к примеру, меньше 0.1. Также есть вход смещения — входной нейрон, который всегда получает единицу, и такой нейрон можно попробовать добавить.

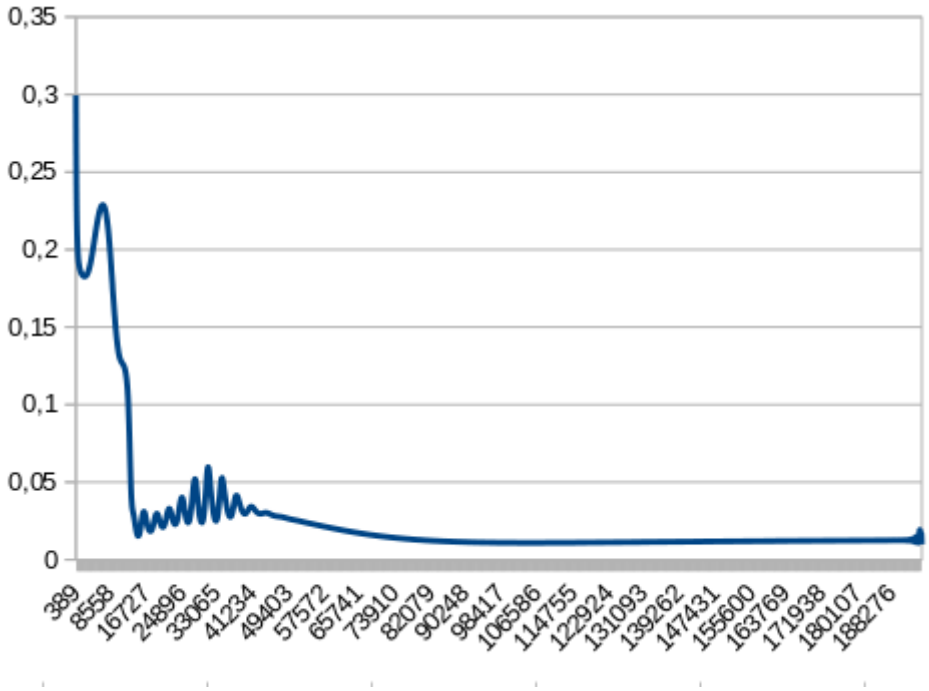


И это помогает! По горизонтальной оси можно видеть эпохи, по вертикальной — средний квадрат отклонения предсказаний сети на эпохе от ожидаемых результатов в абсолютных единицах. Видно, что проблема с тем, что обучение застревало, исчезла.

Корректируем параметры для еще более хорошего результата. Ставим скорость обучения 0.5, учимся до тех пор, пока средний квадрат отклонения не будет меньше 0.01.
Вот полученный график. Видно, что для такой точности потребовалось примерно 200 тысяч эпох, при этом почти все время сеть оттачивала коэффициенты (единица достигнута почти сразу).



Вот график на части эпох, где значение перешло порог в 0.3:



Видно, что ошибка уменьшается, колебания вызваны тем, что подобранное значение скорости иногда большевато, а квадрат это усугубляет визуально. В идеале можно управлять скоростью динамически.

После этого я запустила тестирование сети, для этого было сгенерировано 10 тысяч случайных значений, на них проверялось, как хорошо считает сеть. Параметр проверки — возбужденность отличалась не более, чем на сотую (это примерно как отличие абсолютных данных на десятую — как и в тренировочном сете). Результат — около 78.8% правильных ответов.

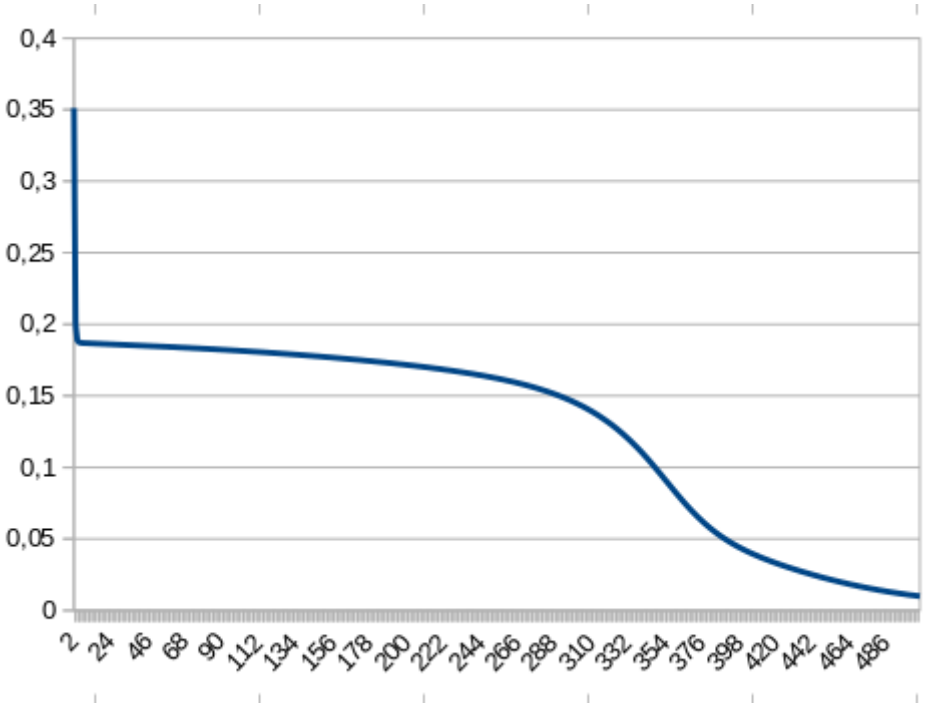
Моделирование булевой функции

Задана функция

$$f_2(x, y, z) = (x \sim y) \downarrow \bar{x}z$$

Ее я буду моделировать примерно такой же сетью. Во входных вершинах появятся еще две, а так — ничего нового. По обучению и проверке — будет всего 8 вариантов входных данных, поэтому все они будут во входном файле, специальные функции не нужны. Проверять на генерированных данных также бесполезно, проверка будет по тем же данным. Если выход сети больше 0.5, считается, что ответ 1, если меньше — ноль.

Сеть обучена с параметрами скорость 0.5, значение квадрата разности для остановки — 0.01, получена слудующая картина по динамике среднего квадрата отклонения:



Гораздо быстрее и проще, чем в предыдущей задаче, и неудивительно — данных гораздо меньше.

Сеть запущена на всех 8 вариантах — результат, конечно, деальный. Все комбинации правильные, точность 100%.