# Лабораторная робота №3

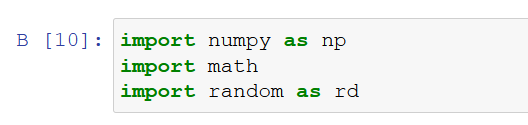
**Тема:** Обучение многослойного персептрона. Предсказание временных рядов с помощью нейросети.

**Цель:** изучить основы нейросетевого моделирования, написать программу обучения многослойного персептрона, приобрести практические навыки использования аппарата нейросети в задачах предсказания временных рядов.

**Ход работы:**

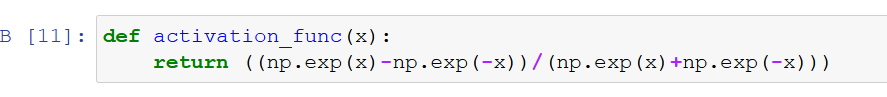
Первым делом перед написанием нейронной сети нужно определиться с логикой написания самой программы. На мой взгляд наиболее доступным и простым является следующий алгоритм действий:

1. Первым делом нужно определиться с количеством скрытых слоёв в нейронной сети (благо в задании указано, что их два). Далее, следует определиться с количеством данных на входе и выходе (для данного случая были взяты 3 числа на входе и одно на выходе). И последнее, о чём нужно подумать заранее: обучающая выборка (можно взять как на предложенном сайте, так и придумать самостоятельно, в последнем случае удачи, шучу).
2. Теперь перейдём к самому построению нейронной сети (параллельно поясняем написанный код).
3. Первым делом импортируем нужные библиотеки языка Python (вопреки ожиданиям их немного и совсем не связаны с нейронными сетями, т.е. изначально можно использовать готовую библиотеку TanserFlow, но мы пойдём другим путём). Первая библиотека numpy отвечает за математическую составляющую, позволяет проворачивать более сложные вещи нежели библиотека math, но нам понадобятся обе. Последняя – random, отвечает за генерацию псевдослучайных чисел, нужен только в самом начале при инициализации весовых коэффициентов.

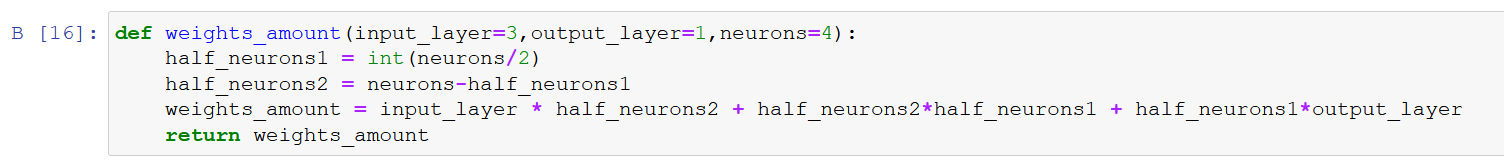


1. Так-с, теперь очередь за некоторыми статическими данными (количество скрытых слоёв, входных и выходных данных), которые пока будут фиксированными, среди них и функция активации.

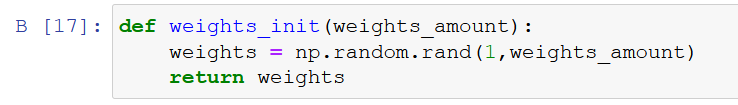




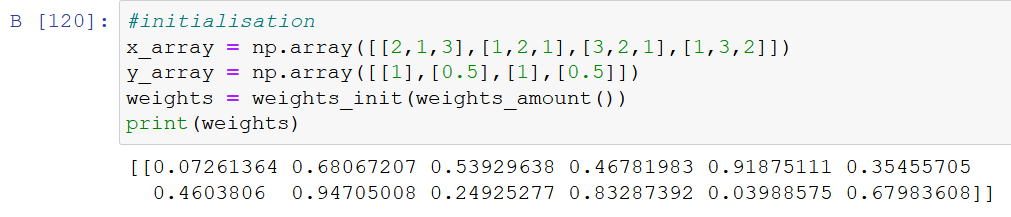
1. Следующим по очереди функция для расчёта количества весовых коэффициентов. Исходим из количества входных и выходных значений и количества нейронов. Логика такая: берём количество значений на входе и умножаем на количество нейронов в первом скрытом слое, потом прибавляем произведение количество нейронов на первом и втором скрытых слоях и плюсуем произведение количества нейронов второго скрытого слоя и количество значений на выходе. Количество нейронов в каждом из слоёв вычисляем как ½ количества нейронов всего (если число нечётное, то опираемся на то, что в первом слое на 1 нейрон больше, чем во втором).



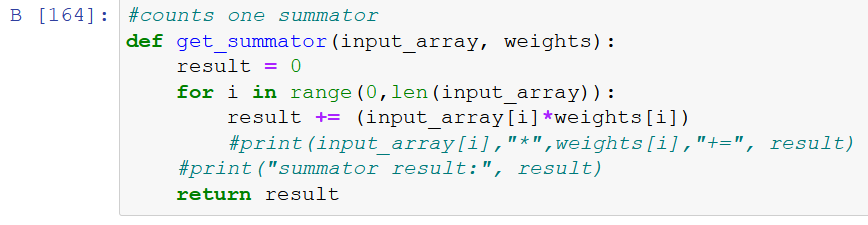
1. После прописываем инициализацию весовых коэффициентов псевдослучайными числами.



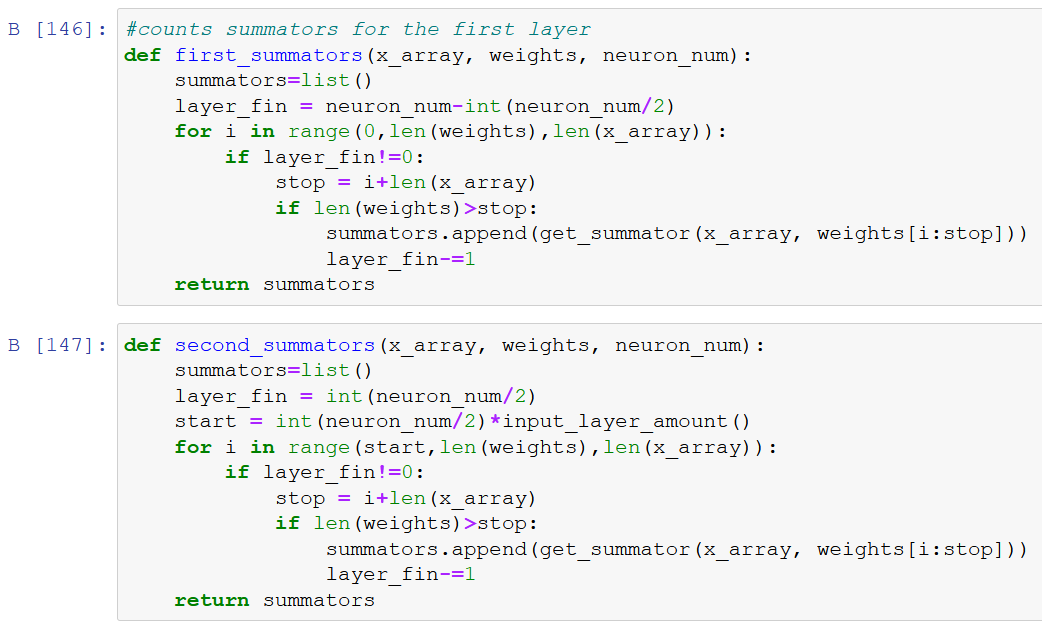
1. Теперь проверим, что уже написано (возьмём несколько выборок для демонстрации), x\_array-входные данные, y\_array-выходные данные, weights - весовые коэффициенты.



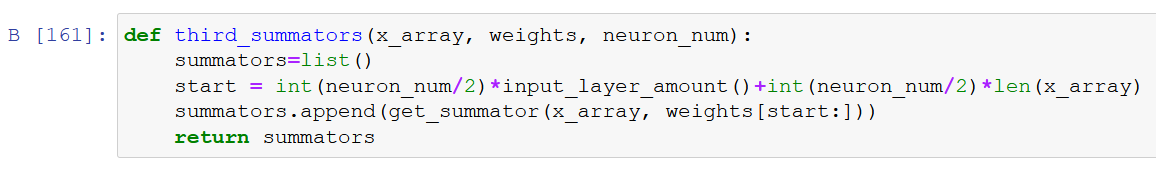
1. Далее, по алгоритму нам нужна функция, отвечающая за расчёт сумматора по каждому коэффициенту.



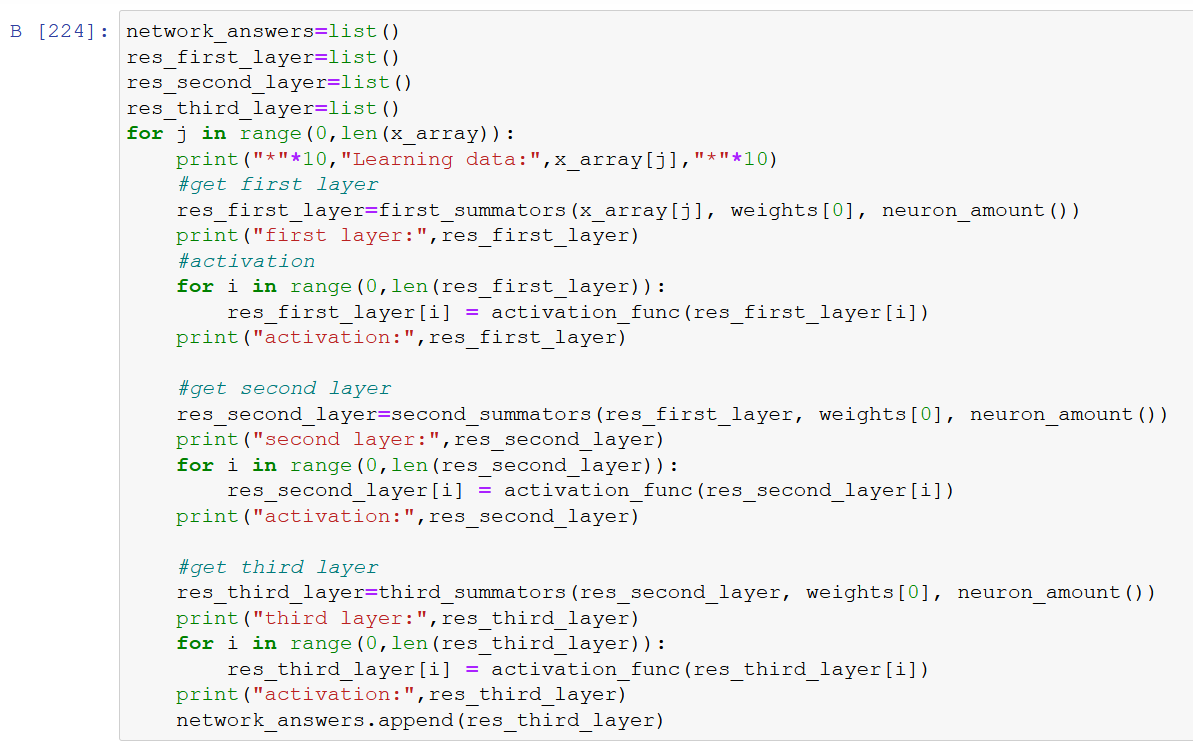
1. Далее, две отдельные функции для расчёта 1-го скрытого слоя и 2-го скрытого слоя, в предыдущую функцию мы посылаем входные данные/результат на предыдущем слое и форматированный массив с весовыми коэффициентами (ничего такого, чисто для удобства расчёта).



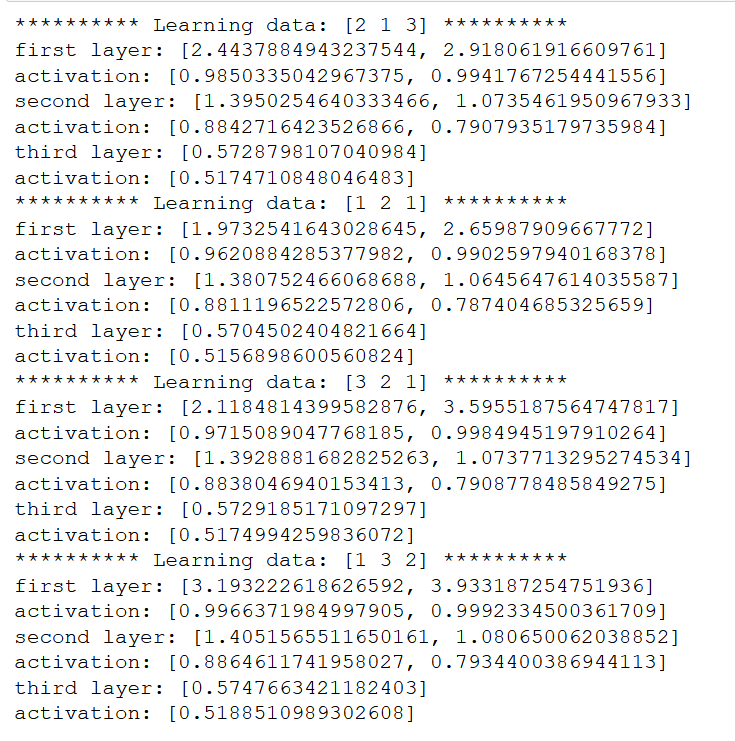
1. И последний слой, формула расчёта та же)



1. Проверка, что у нас тут получилось. Ну, почти, в начале нужно «собрать» всё это безобразие)



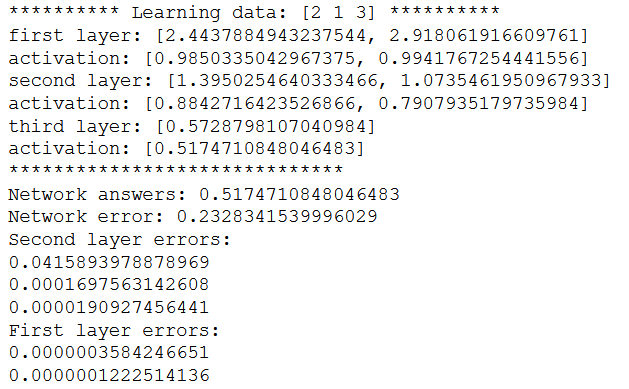
1. А теперь результат:

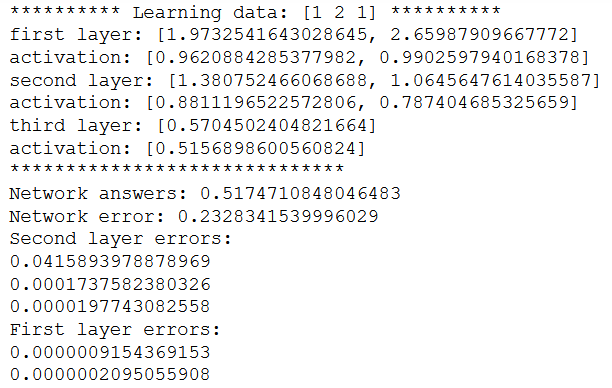


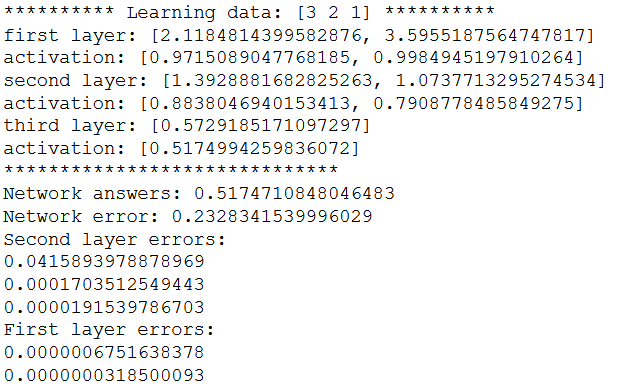
1. Учитывая, что изначальные данные на выходе должны быть вот такие

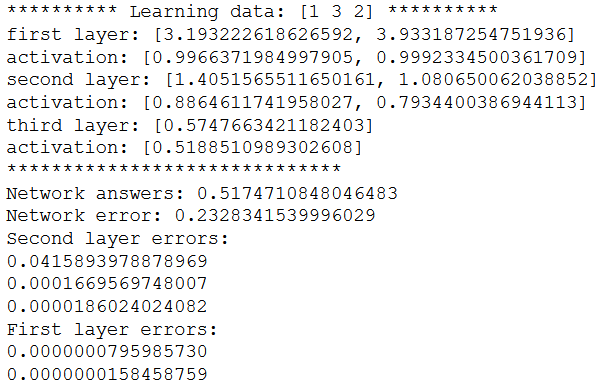


1. Ошибки для каждого слоя

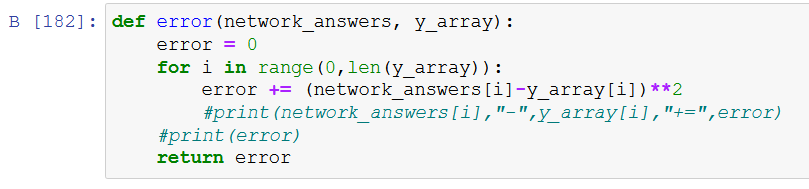


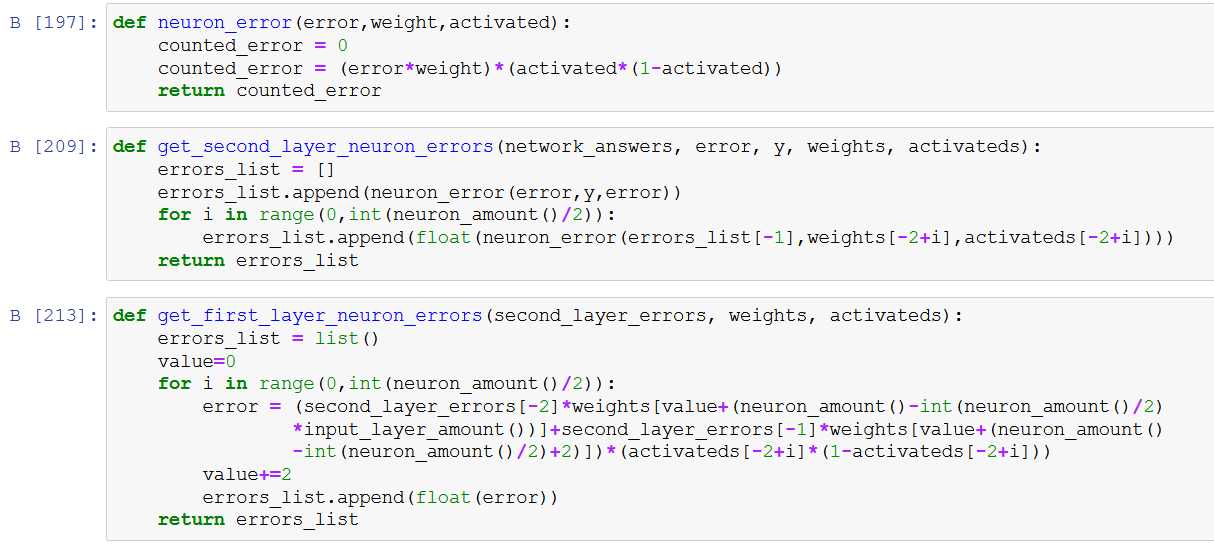






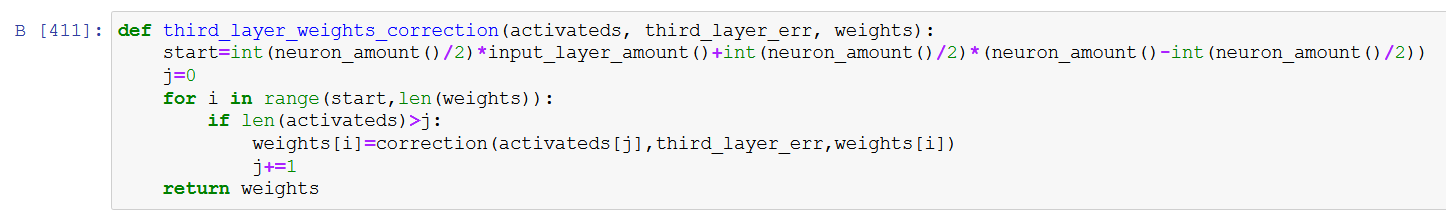
1. Функции, что высчитывают эти ошибки. Первая – общая ошибка на выходе, далее – ошибки на 2-м и 1-м слоях.





1. И последнее, это коррекция весовых коэффициентов.

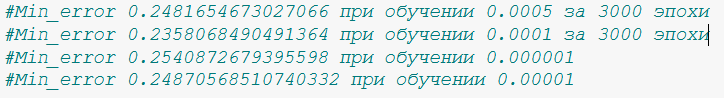




1. Результаты после коррекции

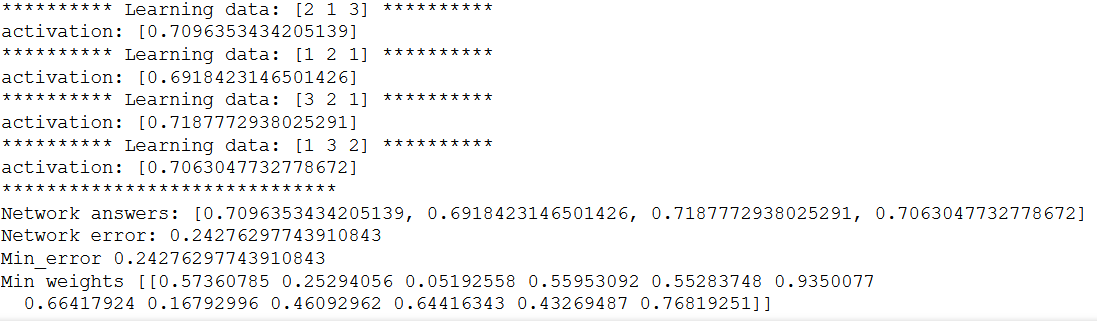
Самым оптимальным принимаем количество эпох обучения в 3000, как практика показала, даже такого, казалось бы, небольшого количества может быть достаточно, стоит признать, что во время прохода по эпохам было выявлена возможность переобучения (если можно так выразиться), т.е. сама ошибка сети с определённого момента начинает резко возрастать, а когда значение ответа сети достигает единицы, то все значения активации, так и ошибок являются NaN, возникает ошибка переполнения. Поэтому обучение нейронной сети приостанавливается, когда ошибка сети начинает возрастать.

Стоит заметить, что ошибка сети так же может быть минимизирована благодаря коэффициенту скорости обучения, было протестировано несколько вариантов и выбран самый оптимальный:



Таки образом, можно заключить, что коэффициент обучения равный 0.0001 за 3 тысячи эпохи обучения показал наилучший результат.

Из-за того, что весовые коэффициенты инициализируются случайным образом, то количество эпох может меняться. Например, 6000 эпох, последняя эпоха.



Подводя итоги, можно сказать, что обучение нейронной сети процесс очень долгий и длинный.