Санкт-Петербургский политехнический университет
Институт Прикладной математики и механики
Кафедра теоретической механики

Курсовой проект на тему:

«Разработка и исследование музыкальной нейронной сети»

Выполнили: студенты гр. 23604/1 Нарядчиков Александр Владимирович, Лебедев Станислав Филиппович
Проверил: Панченко Артём Юрьевич

Введение

Как видно из названия, наша нейронная сеть создает музыку. Главной задачей мы ставили не столько создать шедевры мировой музыкальной индустрии, сколько хотелось разобраться с двумя необычными сферами программирования: нейронные сети и работа с файлами-контейнерами формата WAV.

Что такое нейронная сеть?

Нейронная сеть — это последовательность нейронов, соединенных между собой синапсами. Структура нейронной сети пришла в мир программирования прямиком из биологии. Благодаря такой структуре, машина обретает способность анализировать и даже запоминать различную информацию. Нейронные сети также способны не только анализировать входящую информацию, но и воспроизводить ее из своей памяти. Другими словами, нейросеть это машинная интерпретация мозга человека, в котором находятся миллионы нейронов передающих информацию в виде электрических импульсов.

Для чего нужны нейронные сети?

Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг. Самыми распространенными применениями нейронных сетей является:

Классификация — распределение данных по параметрам. Например, на вход дается набор людей и нужно решить, кому из них давать кредит, а кому нет. Эту работу может сделать нейронная сеть, анализируя такую информацию как: возраст, платежеспособность, кредитная история и тд.

Предсказание — возможность предсказывать следующий шаг. Например, рост или падение акций, основываясь на ситуации на фондовом рынке.

Распознавание — в настоящее время, самое широкое применение нейронных сетей. Используется в Google, когда вы ищете фото или в камерах телефонов, когда оно определяет положение вашего лица и выделяет его и многое другое.

Что такое нейрон?

Нейрон — это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше. Они делятся на три основных типа: входной, скрытый и выходной. В том случае, когда нейросеть состоит из большого количества нейронов, вводят термин слоя. Соответственно, есть входной слой, который получает информацию, n скрытых слоев (обычно их не больше 3), которые ее обрабатывают и выходной слой, который выводит результат. У каждого из нейронов есть 2 основных параметра: входные данные (input data) и выходные данные (output data). В случае входного нейрона: input равна output. В остальных, в поле input попадает суммарная информация всех нейронов с предыдущего слоя, после чего, она нормализуется, с помощью функции активации и попадает в поле output. Нейроны оперируют числами в диапазоне [0,1] или [-1,1].

Что такое синапс?

Синапс это связь между двумя нейронами. У синапсов есть 1 параметр — вес. Благодаря ему, входная информация изменяется, когда передается от одного нейрона к другому. На самом деле, совокупность весов нейронной сети или матрица весов — это своеобразный мозг всей системы. Именно благодаря этим весам, входная информация обрабатывается и превращается в результат. Во время инициализации нейронной сети, веса расставляются в случайном порядке.

Функция активации

Функция активации — это способ нормализации входных данных. То есть, если на входе у вас будет большое число, пропустив его через функцию активации, вы получите выход в нужном вам диапазоне. Функций активации достаточно много поэтому мы рассмотрим самые основные: Линейная, Сигмоид (Логистическая) и Гиперболический тангенс. Главные их отличия — это диапазон значений.

Тренировочный сет

Тренировочный сет — это последовательность данных, которыми оперирует нейронная сеть.

Итерация

Это своеобразный счетчик, который увеличивается каждый раз, когда нейронная сеть проходит один тренировочный сет. Другими словами, это общее количество тренировочных сетов пройденных нейронной сетью.

Эпоха

При инициализации нейронной сети эта величина устанавливается в 0 и имеет потолок, задаваемый вручную. Чем больше эпоха, тем лучше натренирована сеть и соответственно, ее результат. Эпоха увеличивается каждый раз, когда мы проходим весь набор тренировочных сетов.

Ошибка

Ошибка — это процентная величина, отражающая расхождение между ожидаемым и полученным ответами. Ошибка формируется каждую эпоху и должна идти на спад. Если этого не происходит, значит, вы что-то делаете не так. Ошибку можно вычислить разными путями, но мы рассмотрим лишь три основных способа: Mean Squared Error (далее MSE), Root MSE и Arctan. Здесь нет какого-либо ограничения на

использование, как в функции активации, и вы вольны выбрать любой метод, который будет приносить вам наилучший результат. Стоит лишь учитывать, что каждый метод считает ошибки по разному. У Arctan, ошибка, почти всегда, будет больше, так как он работает по принципу: чем больше разница, тем больше ошибка. У Root MSE будет наименьшая ошибка, поэтому, чаще всего, используют MSE, которая сохраняет баланс в вычислении ошибки.

MSE

$$\frac{(i_1-a_1)^2+(i_2-a_2)^2+\ldots+(i_n-a_n)^2}{n}$$

Root MSE

$$\sqrt{\frac{(i_1-a_1)^2+(i_2-a_2)^2+\ldots+(i_n-a_n)^2}{n}}$$

Arctan

$$\frac{\arctan^2(i_1-a_1)+\ldots+\arctan^2(i_n-a_n)}{n}$$

Принцип подсчета ошибки во всех случаях одинаков. За каждый сет, мы считаем ошибку, отняв от идеального ответа, полученный. Далее, либо возводим в квадрат, либо вычисляем квадратный тангенс из этой разности, после чего полученное число делим на количество сетов.

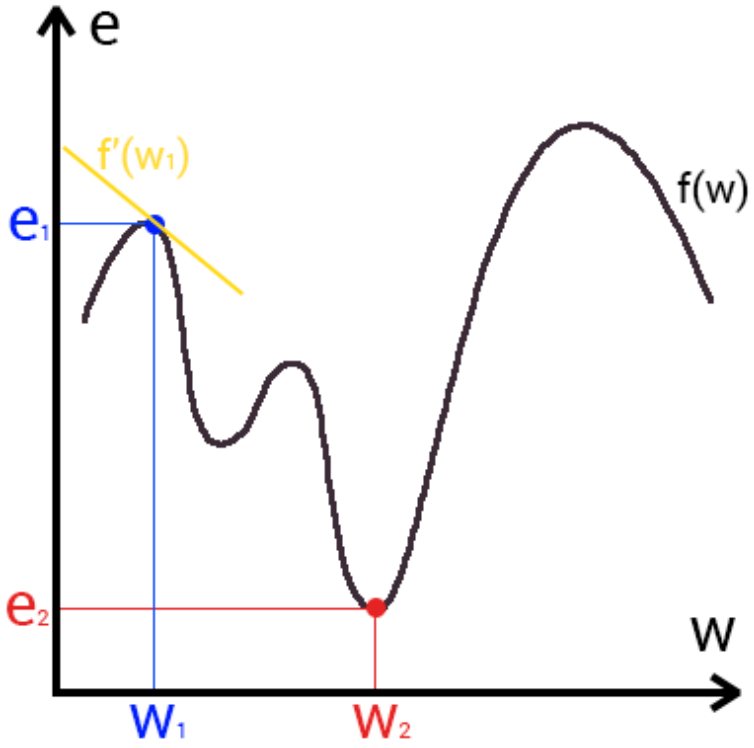
Как сделать чтобы НС давала правильные ответы?

Существует несколько методов обучения НС и я выделю 3, на мой взгляд, самых интересных:

- Метод обратного распространения (Backpropagation)
- Метод упругого распространения (Resilient propagation или Rprop)
- Генетический Алгоритм (Genetic Algorithm)

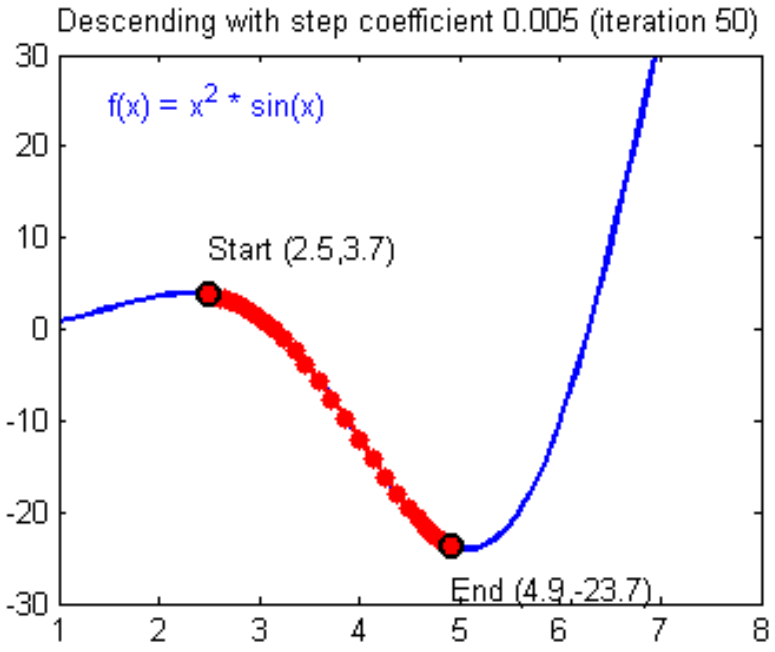
Что такое градиентный спуск?

Это способ нахождения локального минимума или максимума функции с помощью движения вдоль градиента. Если вы поймете суть градиентного спуска, то у вас не должно возникнуть никаких вопросов во время использования метода обратного распространения. Для начала, давайте разберемся, что такое градиент и где он присутствует в нашей НС. Давайте построим график, где по оси x будут значения веса нейрона(w) а по оси y — ошибка соответствующая этому весу(e).

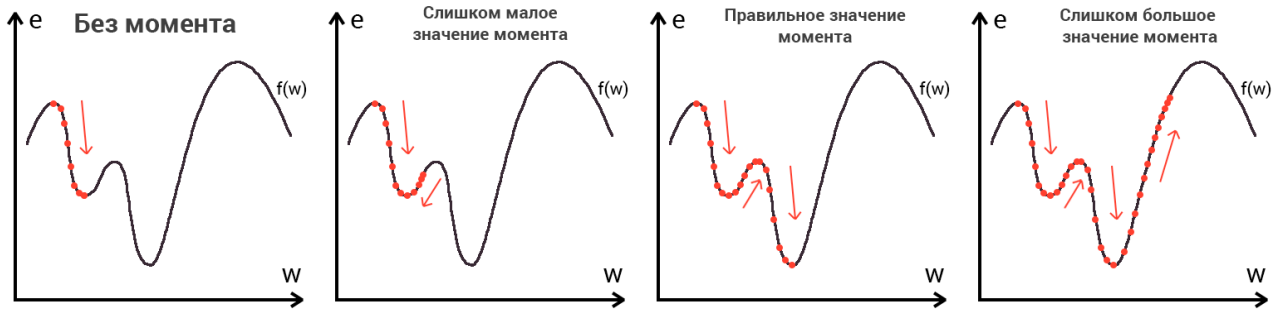


Посмотрев на этот график, мы поймем, что график функция f(w) является зависимостью ошибки от выбранного веса. На этом графике нас интересует глобальный минимум — точка (w2,e2) или, иными словами, то место где график подходит ближе всего к оси x. Эта точка будет означать, что выбрав вес w2 мы получим самую маленькую ошибку — e2 и как следствие, самый лучший результат из всех возможных. Найти же эту точку нам поможет метод градиентного спуска (желтым на графике обозначен градиент). Соответственно у каждого веса в нейросети будет свой график и градиент и у каждого надо найти глобальный минимум.

Так что же такое, этот градиент? Градиент — это вектор который определяет крутизну склона и указывает его направление относительно какой либо из точек на поверхности или графике. Чтобы найти градиент нужно взять производную от графика по данной точке (как это и показано на графике). Двигаясь по направлению этого градиента мы будем плавно скатываться в низину. Теперь представим что ошибка — это лыжник, а график функции — гора. Соответственно, если ошибка равна 100%, то лыжник находиться на самой вершине горы и если ошибка 0% то в низине. Как все лыжники, ошибка стремится как можно быстрее спуститься вниз и уменьшить свое значение. В конечном случае у нас должен получиться следующий результат:

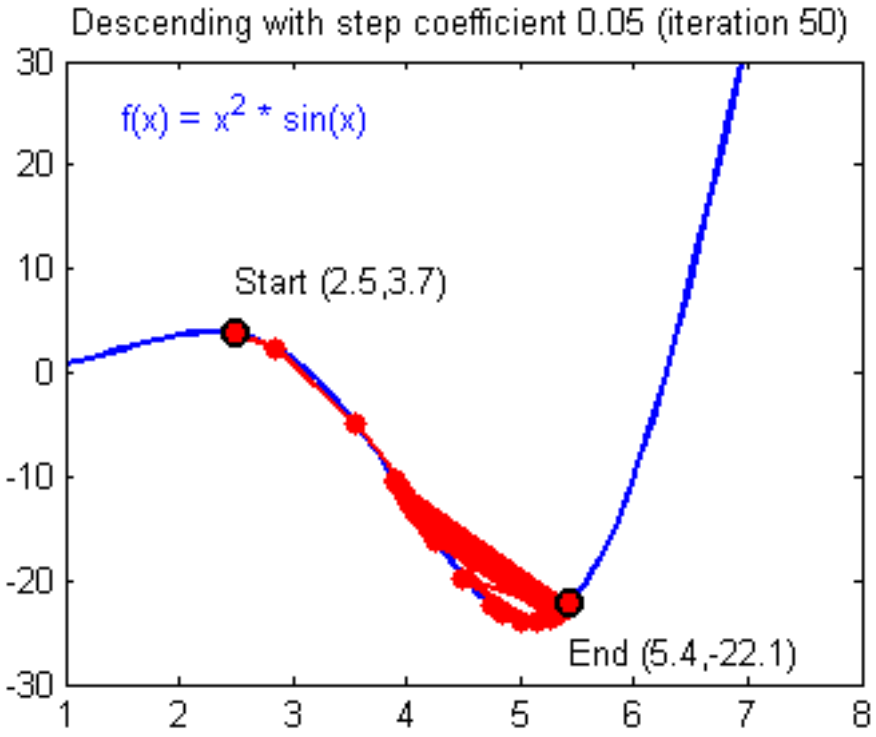


Представьте что лыжника забрасывают, с помощью вертолета, на гору. На сколько высоко или низко зависит от случая (аналогично тому, как в нейронной сети при инициализации веса расставляются в случайном порядке). Допустим ошибка равна 90% и это наша точка отсчета. Теперь лыжнику нужно спуститься вниз, с помощью градиента. На пути вниз, в каждой точке мы будем вычислять градиент, что будет показывать нам направление спуска и при изменении наклона, корректировать его. Если склон будет прямым, то после n-ого количества таких действий мы доберемся до низины. Но в большинстве случаев склон (график функции) будет волнистый и наш лыжник столкнется с очень серьезной проблемой — локальный минимум. Попадание в локальный минимум чревато тем, что наш лыжник навсегда останется в этой низине и никогда не скатиться с горы, следовательно мы никогда не сможем получить правильный ответ. Но можно избежать этого, снарядив лыжника моментом (momentum). Вот его краткая иллюстрация:



Момент придаст лыжнику необходимое ускорение чтобы преодолеть холм, удерживающий нас в локальном минимуме, однако здесь есть одно НО. Представим что мы установили определенное значение параметру момент и без труда смогли преодолеть все локальные минимумы, и добраться до глобального минимума. Так как мы не можем просто отключить реактивный ранец, то мы можем проскочить глобальный минимум, если рядом с ним есть еще низины. В конечном случае это не так важно, так как рано или поздно мы все равно вернемся обратно в глобальный минимум, но стоит помнить, что чем больше момент, тем больше будет размах с которым лыжник будет кататься по низинам. Вместе с моментом в методе обратного распространения также используется такой параметр как скорость обучения (learning rate). Скорость обучения, также как и момент, является гиперпараметром — величина которая

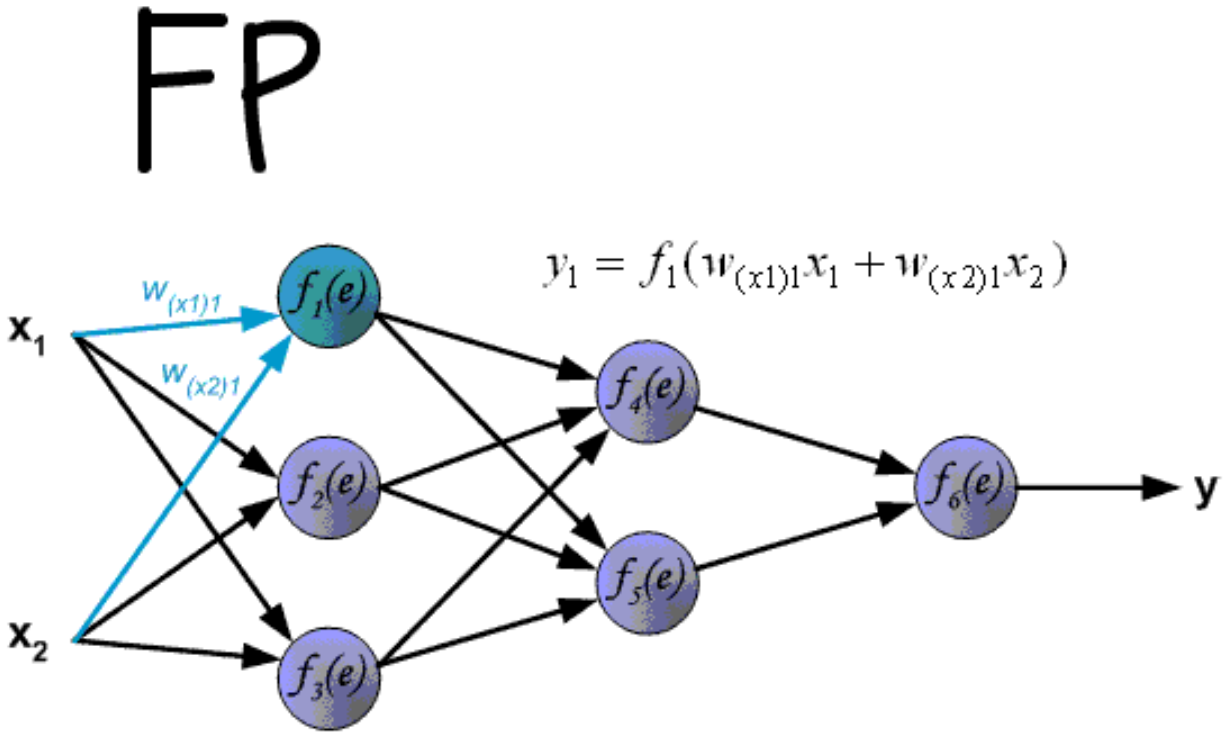
подбирается путем проб и ошибок. Скорость обучения можно напрямую связать со скоростью лыжника и можно с уверенностью сказать — тише едешь дальше будешь. Однако здесь тоже есть определенные аспекты, так как если мы совсем не дадим лыжнику скорости то он вообще никуда не поедет, а если дадим маленькую скорость то время пути может растянуться на очень и очень большой период времени. Что же тогда произойдет если мы дадим слишком большую скорость?



Как видите, ничего хорошего. Лыжник начнет скатываться по неправильному пути и возможно даже в другом направлении, что как вы понимаете только отдалит нас от нахождения правильного ответа. Поэтому во всех этих параметрах нужно находить золотую середину чтобы избежать не сходимости НС.

Что такое Метод Обратного Распространения (МОР)?

Визуализация МОРа:



Последовательно передается информацию от входных нейронов к выходным. После чего вычисляется ошибку и основываясь на ней делается обратную передачу, которая заключается в том, чтобы последовательно менять веса нейронной сети, начиная с весов выходного нейрона. Значение весов будут меняться в ту сторону, которая даст нам наилучший результат.

Что еще нужно знать о процессе обучения?

Нейросеть можно обучать с учителем и без (supervised, unsupervised learning).

Обучение с учителем — это тип тренировок присущий таким проблемам как регрессия и классификация (им мы и воспользовались в примере приведенном выше). Иными словами здесь вы выступаете в роли учителя а НС в роли ученика. Вы предоставляете входные данные и желаемый результат, то есть ученик посмотрев на входные данные поймет, что нужно стремиться к тому результату который вы ему предоставили.

Обучение без учителя — этот тип обучения встречается не так часто. Здесь нет учителя, поэтому сеть не получает желаемый результат или же их количество очень мало. В основном такой вид тренировок присущ НС у которых задача состоит в группировке данных по определенным параметрам. Допустим вы подаете на вход 10000 статей на хабре и после анализа всех этих статей НС сможет распределить их по категориям основываясь, например, на часто встречающихся словах. Статьи в которых упоминаются языки программирования, к программированию, а где такие слова как Photoshop, к дизайну.

Существует еще такой интересный метод, как **обучение с подкреплением** (reinforcement learning). Этот метод заслуживает отдельной статьи, но я попытаюсь вкратце описать его суть. Такой способ применим тогда, когда мы можем основываясь на результатах полученных от НС, дать ей оценку. Например мы хотим научить НС играть в PAC-MAN, тогда каждый раз когда НС будет набирать много очков мы будем ее поощрять. Иными словами мы предоставляем НС право найти любой способ достижения цели, до тех пор пока он будет давать хороший результат. Таким способом, сеть начнет понимать чего от нее хотят добиться и пытается найти наилучший способ достижения этой цели без постоянного предоставления данных “учителем”.

WAV

Итак, рассмотрим самый обычный WAV файл (Windows PCM). Он представляет собой две, четко делящиеся, области. Одна из них – заголовок файла, другая – область данных. В заголовке файла хранится информация о: размере файла, количестве каналов, частоте дискретизации, количестве бит в сэмпле (эту величину еще называют глубиной звучания). Но для большего понимания смысла величин в заголовке следует еще рассказать об области данных и оцифровке звука. Звук состоит из колебаний, которые при оцифровке приобретают ступенчатый вид. Этот вид обусловлен тем, что компьютер может воспроизводить в любой короткий промежуток времени звук определенной амплитуды (громкости) и этот короткий момент далеко не бесконечно короткий. Продолжительность этого промежутка и определяет частота дискретизации. Например, у нас файл с частотой дискретизации 44.1 kHz, это значит, что тот короткий промежуток времени равен 1/44100 секунды (следует из размерности величины Гц = 1/с). Современные звуковые карты поддерживают частоту дискретизации до 192 kHz. Так, со временем разобрались. Теперь, что касается амплитуды (громкости звука в коротком промежутке времени). От нее, я бы сказал, зависит точность звука. Амплитуда выражается числом, занимаемым в памяти (файле) 8, 16, 24, 32 бит (теоретически можно и больше). Как известно, 8 бит = 1 байту, следовательно, какая-то одна амплитуда в какой-то короткий промежуток времени в памяти (файле) может занимать 1, 2, 3, 4 байта соответственно. Таким образом, чем больше число занимает места в памяти (файле), тем больше диапазон значений для этого числа, а значит и для амплитуды.
1 байт – 0..255 2 байта – 0..65 535 3 байта – 0..16 777 216 4 байта – 0..4 294 967 296. В моно варианте значения амплитуды расположены последовательно. В стерео же, например, сначала идет значение амплитуды для левого канала, затем для правого, затем снова для левого и так далее. Совокупность амплитуды и короткого промежутка времени носит название сэмпл. Теперь таблица, наглядно показывающая структуру WAV файла.

Местоположение	Поле	Описание
0..3 (4 байта)	`chunkId`	Содержит символы “RIFF” в ASCII кодировке (`0x52494646` в big-endian представлении). Является началом RIFF-цепочки.
4..7 (4 байта)	`chunkSize`	Это оставшийся размер цепочки, начиная с этой позиции. Иначе говоря, это размер файла – 8, то есть, исключены поля chunkId и chunkSize.
8..11 (4 байта)	`format`	Содержит символы “WAVE” (`0x57415645` в big-endian представлении)
12..15 (4 байта)	`subchunk1Id`	Содержит символы “fmt “ (`0x666d7420` в big-endian представлении)
16..19 (4 байта)	`subchunk1Size`	16 для формата **PCM**. Это оставшийся размер подцепочки, начиная с этой позиции.
20..21 (2 байта)	`audioFormat`	Аудио формат, полный список можно получить здесь. Для **PCM** = 1 (то есть, Линейное квантование). Значения, отличающиеся от 1, обозначают некоторый формат сжатия.
22..23 (2 байта)	`numChannels`	Количество каналов. Моно = 1, Стерео = 2 и т.д.
24..27 (4 байта)	`sampleRate`	Частота дискретизации. 8000 Гц, 44100 Гц и т.д.
28..31 (4 байта)	`byteRate`	Количество байт, переданных за секунду воспроизведения.
32..33 (2 байта)	`blockAlign`	Количество байт для одного сэмпла, включая все каналы.
34..35 (2 байта)	`bitsPerSample`	Количество бит в сэмпле. Так называемая “глубина” или точность звучания. 8 бит, 16 бит и т.д.
36..39 (4 байта)	`subchunk2Id`	Содержит символы “data” (`0x64617461` в big-endian представлении)
40..43 (4 байта)	`subchunk2Size`	Количество байт в области данных.
44..	`data`	Непосредственно WAV-данные.

Итоги

Создана нейоронная сеть с возможностью замены метода тренировки и удобным классом загрузки музыки формата WAV.

Использованные статьи

1. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
2. http://stevenmiller888.github.io/mind-how-to-build-a-neural-network/