**Всероссийский Конкурс исследовательских проектов,**

**выполненных школьниками и студентами при научном консультировании**

**ученых Международной ассоциации строительных вузов**

**ФГБОУ ВПО «Волгоградский государственный архитектурно-строительный университет»**

**Секция:**

**Номинация 11 классы**

**Тема проекта: Применение нейронных сетей для вычисления количества слагаемых в экспериментальных функциях DHO**

**Поздняков Илья Андреевич**

[**pozd74@mail.ru**](mailto:pozd74@mail.ru)

**ученик 11 класса «Б» МОУ лицея №5 имени Ю. А. Гагарина**

**г. Волгоград**

**Научный консультант и руководитель: заведующий кафедрой «Физика» ВолгГТУ**

**Завьялов Дмитрий Викторович**

Оглавление:

Введение……………………………………..……………………………3

Нейронные сети……………………………..……………………………4

Немного о Wolfram Mathematica……………………..………………..11

Экспериментальная часть……………………………...……………….13

### Заключение……………………………………………………………....15

### Список литературы ……………………………………………………..16

**Введение.**

Довольно **актуальной** задачей при исследовании процессов распространения волн в жидкостях и газах является задача подбора некоторой экспериментально полученной функции частоты с помощью серии пиков определенной формы (обычно это так называемые Damped Oscillator Functions).

В математическом виде задача ставится следующим образом: есть набор экспериментальных точек {*w*i, *s*i} отражающих зависимость какой либо физической величины от частоты и нужно подобрать выражение вида

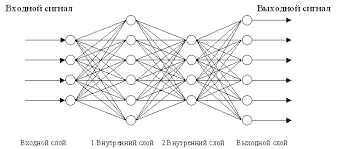
(1)

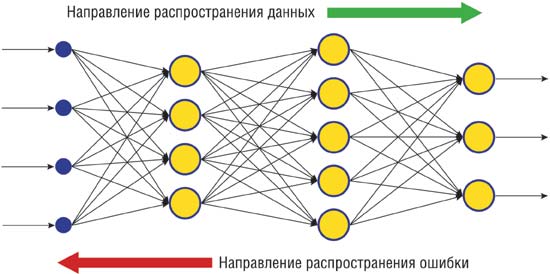
где *Ai* – амплитуда *i*-го пика, *Гi* – средняя ширина *i*-го пика и – значение максимума *i*-го пика для представления экспериментальной функции. Все вышеперечисленные параметры и, кроме того, количество пиков *n* должны подбираться таким образом, чтобы удовлетворять некоему разумному критерию согласия. Обычно при заданном *n* производится подбор параметров *A*, *Г,* всех пиков каким либо численным методом, минимизируя сумму квадратов отклонений экспериментальных значений от теоретических, посчитанных по формуле (1). Подобную процедуру делают для нескольких значений *n* и выбирают ту оптимизацию, которая максимизирует, например, критерий *R*2. Весь процесс является численно очень сложным и занимает много компьютерного времени. Быстрое определение хотя бы параметра *n* позволит существенно сократить время расчетов. Решением этой **задачи** может быть использование нейронной сети, способной подобрать количество слагаемых за приемлемые сроки. Таким образом, **целью** данной работы является ответ на вопрос “Возможно ли с помощью нейронной сети решить выше поставленную задачу”.

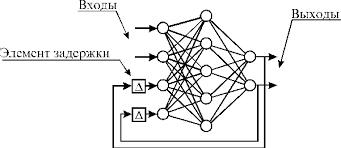
**Нейронные сети.**

**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.

  
  
**Какие бывают нейронные сети?**

* Сети прямого распространения (СПР). В таких сетях информация передается лишь в одну сторону. Все связи направлены строго от входных нейронов к выходным.
* Рекуррентные сети. В таких сетях между слоями существуют связи как в направлении от входного слоя к выходному, так и в обратном.



**Для чего нужны нейронные сети?**

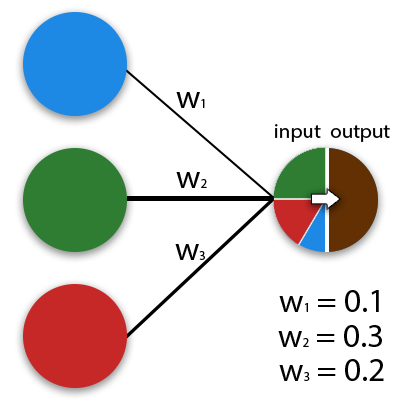
Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг. Самыми распространенными применениями нейронных сетей является:  
**Классификация** — распределение данных по параметрам. Например, на вход дается набор людей и 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.

Теперь, чтобы понять, как же работают нейронные сети, давайте взглянем на ее составляющие и их параметры.

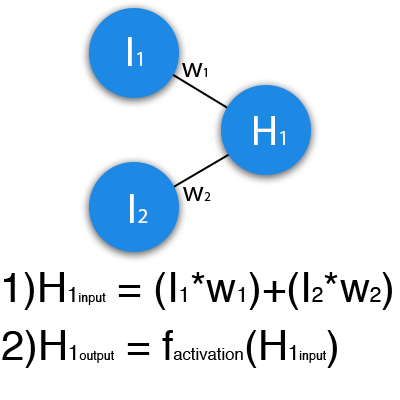
**Что такое нейрон?**

  
Нейрон — это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше. Они делятся на три основных типа: входной (синий), скрытый (красный) и выходной (зеленый). Также есть нейрон смещения и контекстный нейрон. 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=output. В остальных, в поле input попадает суммарная информация всех нейронов с предыдущего слоя, после чего, она нормализуется, с помощью функции активации (пока что просто представим ее f(x)) и попадает в поле output.  
**Важно помнить**, что нейроны оперируют числами в диапазоне [0,1] или [-1,1]. А как же, вы спросите, тогда обрабатывать числа, которые выходят из данного диапазона? На данном этапе, самый простой ответ — это разделить 1 на это число. Этот процесс называется нормализацией, и он очень часто используется в нейронных сетях.

**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. Допустим, есть 3 нейрона, которые передают информацию следующему. Тогда у нас есть 3 веса, соответствующие каждому из этих нейронов. У того нейрона, у которого вес будет больше, та информация и будет доминирующей в следующем нейроне (пример — смешение цветов). 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.  
**Важно помнить**, что во время инициализации нейронной сети, веса расставляются в случайном порядке.

**Как работает нейронная сеть?**

****

В данном примере изображена часть нейронной сети, где буквами I обозначены входные нейроны, буквой H — скрытый нейрон, а буквой w — веса. Из формулы видно, что входная информация — это сумма всех входных данных, умноженных на соответствующие им веса. Тогда дадим на вход 1 и 0. Пусть w1=0.4 и w2 = 0.7 Входные данные нейрона Н1 будут следующими: 1\*0.4+0\*0.7=0.4. Теперь когда у нас есть входные данные, мы можем получить выходные данные, подставив входное значение в функцию активации (подробнее о ней далее). Теперь, когда у нас есть выходные данные, мы передаем их дальше. И так, мы повторяем для всех слоев, пока не дойдем до выходного нейрона. Запустив такую сеть в первый раз мы увидим, что ответ далек от правильно, потому что сеть не натренирована. Чтобы улучшить результаты мы будем ее тренировать. Но прежде чем узнать как это делать, давайте введем несколько терминов и свойств нейронной сети.

**Функция активации**

**Функция активации** — это способ нормализации входных данных (мы уже говорили об этом ранее). 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.

**Линейная функция**

Эта функция почти никогда не используется, за исключением случаев, когда нужно протестировать нейронную сеть или передать значение без преобразований.

**Сигмоид**

Это самая распространенная функция активации, ее диапазон значений [0,1]. Именно на ней показано большинство примеров в сети, также ее иногда называют логистической функцией. Соответственно, если в вашем случае присутствуют отрицательные значения (например, акции могут идти не только вверх, но и вниз), то вам понадобиться функция которая захватывает и отрицательные значения.

**Гиперболический тангенс**

Имеет смысл использовать гиперболический тангенс, только тогда, когда ваши значения могут быть и отрицательными, и положительными, так как диапазон функции [-1,1]. Использовать эту функцию только с положительными значениями нецелесообразно так как это значительно ухудшит результаты вашей нейросети.

**Тренировочный сет**

Тренировочный сет — это последовательность данных, которыми оперирует нейронная сеть. В нашем случае исключающего или (xor) у нас всего 4 разных исхода то есть у нас будет 4 тренировочных сета: 0xor0=0, 0xor1=1, 1xor0=1,1xor1=0.

**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, в нашем случае, 4 сетов или 4 итераций.

**Важно** не путать итерацию с эпохой и понимать последовательность их инкремента. Сначала n  
раз увеличивается итерация, а потом уже эпоха и никак не наоборот. Другими словами, нельзя сначала тренировать нейросеть только на одном сете, потом на другом и т.д. Нужно тренировать каждый сет один раз за эпоху. Так, вы сможете избежать ошибок в вычислениях.

**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.]

**Wolfram Mathematica**

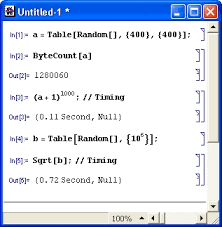
Для создания тренировочного набора и нейронной сети была использована программа “**Wolfram Mathematica**”.

**Wolfram Mathematica** — это программное обеспечение, не только для математических вычислений, это гораздо больше: от моделирования и симуляции, визуализации, документации, до создания веб-сайтов. *Mathematica* обладает возможностью осуществлять вызовы функций и принимать вызовы с C, .NET, Java и других языков, генерировать C код, компилировать автономные библиотеки и исполняемые файлы.

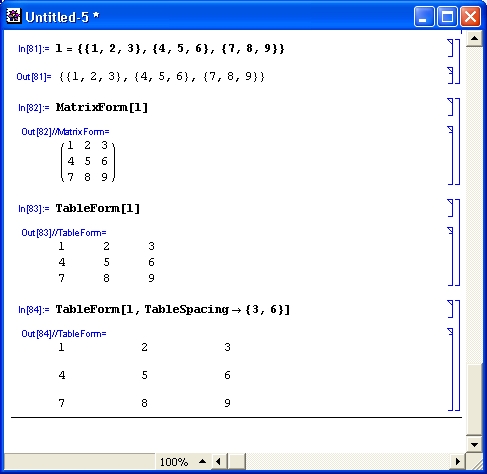
Mathematica была создана Стефаном Вольфрамом (Stephen Wolfram) и разрабатывается Wolfram Research. Первая версия была выпущена 23 июня 1988 года.

Mathematica предоставляет пользователю огромный инструментарий:

* символьные и численные вычисления,



* библиотеки математических и статистических функций, теории групп и теории чисел



* работа с графикой, в том числе инструменты визуализации и анимации, параллельных вычислений, подключения внешних DLL и т.д.



**Практическая часть.**

Основной идеей применения нейронной сети в поставленной задаче подбора количества DHO-пиков является тренировка нейронной сети на так называемом тренировочном множестве. У нас тренировочным множеством является множество функций, построенных по формуле (1) с заранее известными количеством слагаемых *n* и параметрами *A*, *Г,* , выбранными случайным образом. При этом вопросы реализации нейронной сети и алгоритма обучения методом обратного распространения ошибки берет на себя пакет Wolfram Mathematica.

Ниже представлена написанная нами на высокоуровневом языке Wolfram Language программа для тренировки нейронной сети (в качестве сети была выбрана простейшая сеть прямого распространения).

(\*Функция конструирующая формулу из dhocount слагаемых-пиков\*)

**DHOConstruct[dhocount\_] := Module[{vars, csum},**

**vars =**

**Table[{ToExpression["E" <> ToString[i]],**

**ToExpression["A" <> ToString[i]],**

**ToExpression["G" <> ToString[i]]}, {i, dhocount}];**

**csum =**

**Total[Table[**

**Ex^2\*vars[[i]][[1]]^2\*vars[[i]][[2]]\***

**vars[[i]][[**

**3]]/((Ex^2 - vars[[i]][[1]]^2)^2 + (Ex\***

**vars[[i]][[3]])^2), {i, dhocount}]];**

**Return[{csum, Flatten[vars]}];**

**];**

(\*Функция конструирующая список из traincount массивов для тренировки \

сети. Каждый массив - это значения, построенные по формуле из \

dhocount слагаемых-пиков на промежутке от low до high по оси абсцисс. \

Размер каждого массива - pointcount точек. Все параметры DHO-функций \

выбираются случайным образом.\*)

**TrainList[dhocount\_, traincount\_, low\_, high\_, pointcount\_] :=**

**Module[{fun, vars, list, fun1, rules, values},**

**{fun, vars} = DHOConstruct[dhocount];**

**list = {};**

**values = ConstantArray[0, 3\*dhocount];**

**Do[**

**values = RandomReal[{0, 1}, 3\*dhocount];**

**values[[1 ;; All ;; 3]] = RandomReal[{low, high}, dhocount];**

**rules = MapThread[#1 -> #2 &, {vars, values}];**

**fun1 = fun /. rules;**

**AppendTo[list,**

**Map[fun1 /. Ex -> # &, FindDivisions[{low, high}, pointcount]]];**

**, {i, traincount}**

**];**

**Return[list];**

**];**

(\*Функция конструирующая список из traincount массивов для тренировки \

сети. Каждый массив - это значения, построенные по формуле из \

[1,maxdhocount] слагаемых-пиков на промежутке от low до high по оси \

абсцисс. Размер каждого массива - pointcount точек.Все параметры \

DHO-функций выбираются случайным образом.\*)

**ConstructTrainSet[maxdhocount\_, traincount\_, low\_, high\_,**

**pointcount\_] := Module[{trainlist, list},**

**trainlist = {};**

**Do[**

**list = TrainList[dc, traincount, low, high, pointcount];**

**AppendTo[trainlist, Map[# -> dc &, list]];**

**, {dc, maxdhocount}];**

**Return[Flatten[trainlist, 1]]**

**]**

Обучение сети запускается с помощью команды

(2)

Результатом обучения является функция-классификатор, стоящая в левой части выражения (2).

Для проверки качества обучения (и вообще возможности решения поставленной задачи с помощью нейронной сети) нужно применить функцию-классификатор к какой-либо функции, построенной по формуле (1) со случайными параметрами и посмотреть насколько точно она оценивает количество слагаемых в неизвестной ей функции. Функция *d* возвращает вероятности того, что в посланной в качестве аргумента функции было то или иное количество пиков. Типичные результаты нескольких подобных вызовов представлены ниже в виде перечисления пар <количество пиков> -> <вероятность того, что в аргументе это количество пиков>.

В качестве проверочных функций сконструирован набор функций с тремя пиками:

Из представленных результатов следует, что чаще всего самую большую вероятность функция-классификатор приписывает именно трем пикам. Но иногда три пика стоят на втором месте. Это факт отражает неизбежные ошибки нейронной сети в распознавании. Самым прямым способом уменьшения их количества является увеличение тренировочной выборки. Например, в случае нашей задачи тренировочная выборка в 1000 функций давала примерно 70% ошибку в распознавании (хотя правильное количество пиков практически всегда стояло на втором месте по вероятности при проверке). Увеличение базы до 20000 функций дало существенное уменьшение количества ошибок, до 30%. Дальнейшее увеличение тренировочной базы было уже не так эффективно в смысле уменьшения количества ошибок, которое упало примерно до 20% при размере выборки 100000 функций и далее не падало. Это говорит о пределе точности выбранной топологии сети на данном типе задач. Для получения большей точности задачах классификации наука о данных в последнее время рекомендует так называемые Convolution Network.

**Заключение.**

В результате проведенных численных экспериментов стало ясно, что нейронная сеть прямого распространения в целом успешно справляется с задачей определения количества DHO-пиков в экспериментальных данных, но лучшим вариантом все же будет использование более современных сетей типа Convolution Network лучше приспособленных для задач классификации. Это и будет темой дальнейшего исследования.

**Список литературы.**

# Steven Miller - Mind: How to Build a Neural Network

* Matt Mazur - A Step by Step Back propagation Example
* Neural Networks for Java – Burc Derya Gunes