Нейронные сети

Основной структурной и функциональной частью нейронной сети является формальный нейрон (formal neuron), представленный на рис. 1, где $x_0, x_1, ..., x_n$ – компоненты вектора входных сигналов, $w_0, w_1, ..., w_n$ – значения весов входных сигналов нейрона, а y – выходной сигнал нейрона.

Формальный нейрон состоит из элементов 3 типов: умножителей (синапсов), сумматора и преобразователя. Синапс характеризует силу (вес) связи между двумя нейронами. Сумматор выполняет сложение входных сигналов, предварительно помноженных на соответствующие веса.

Преобразователь реализует функцию одного аргумента – выхода сумматора. Эта функция называется функцией активации или передаточной функцией нейрона. Исходя из данного описания, математическая модель нейрона может быть представлена следующим образом:

$$y = f(S),$$

$$S = \sum_{i=1}^{n} w_i x_i + b$$

где b- смещение, его можно представить как $w_0 \cdot 1$.

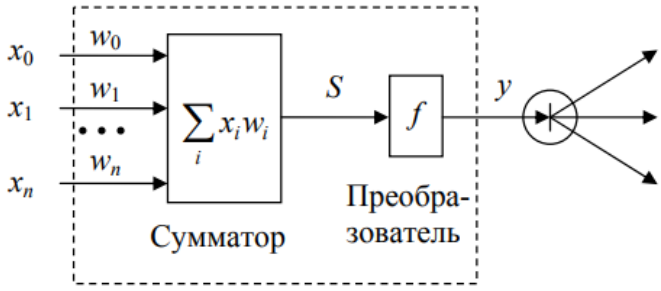


Рисунок 1. Формальный нейрон

Примеры некоторых активационных функций представлены в табл. 1 и на рис. 2.

Таблица 1. Активационные функции

Название	Формула	Область значений
Пороговая	$f(S)=\begin{cases}0, S<\Theta\\1, S\geq\Theta\end{cases}$	(0, 1)
Линейная	$f(S)=aS$	$(-\infty,+\infty)$
Лог-сигмоидная	$f(S)=\frac{1}{1+e^{-aS}}$	(0, 1)
Гиперболический тангенс	$f(S)=\frac{e^{aS}-e^{-aS}}{e^{aS}+e^{-aS}}$	(–1, 1)

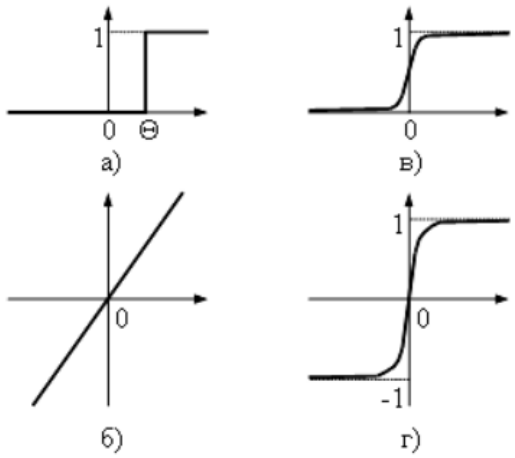


Рисунок 2. Примеры активационных функций: а) пороговая; б) линейная; в) лог-сигмоидная; г) гиперболический тангенс

НЕЙРОННЫЕ СЕТИ

Формальные нейроны можно объединять таким образом, что выходные сигналы одних нейронов являются входными для других. Полученное множество связанных между собой нейронов называют искусственными нейронными сетями (artificial neural networks, ANN, ИНС) или, коротко, нейронными сетями.

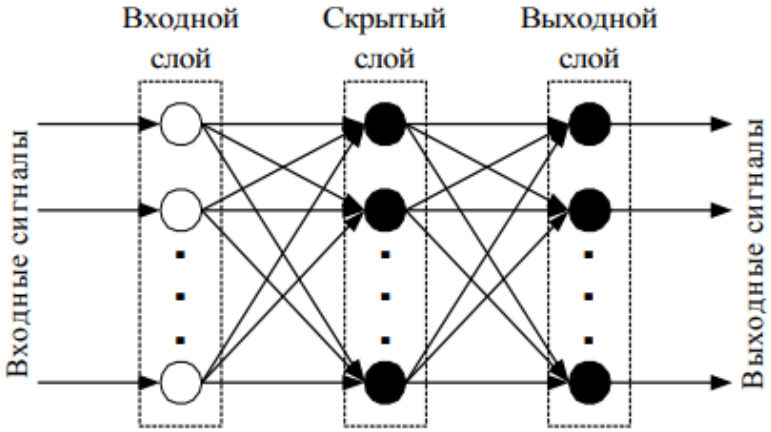


Рисунок 3. Пример многослойной нейронной сети прямого распространения

Различают следующие три общих типа нейронов, в зависимости от их положения в нейронной сети:

- входные нейроны (input nodes), на которые подаются входные для всей сети сигналы. Такие нейроны имеют, как правило, один вход с единичным весом, смещение отсутствует, а значение выхода нейрона равно входному сигналу;
- выходные нейроны (output nodes), выходные значения которых представляют результирующие выходные сигналы нейронной сети;
- скрытые нейроны (hidden nodes), не имеющие прямых связей с входными сигналами, при этом значения выходных сигналов скрытых нейронов не являются выходными сигналами ИНС.

Отметим, что структуру ИНС можно рассматривать как ориентированный граф, в котором узлы соответствуют нейронам, а ребра – межнейронным связям. По структуре межнейронных связей различают два класса ИНС:

1. ИНС прямого распространения (feed-forward ANNs), в которых сигнал распространяется только от входных нейронов к выходным. Орграф, соответствующий таким ИНС, не имеет циклов и петель.

2. Рекуррентные ИНС (recurrent ANNs) – ИНС с обратными связями. В таких ИНС сигналы могут передаваться между любыми нейронами, вне зависимости от их расположения в ИНС. Орграф, соответствующий структуре рекуррентных ИНС, может иметь петли и циклы

Среди различных структур ИНС наиболее известны многослойные ИНС (multi-layered ANNs) (рис. 3). Рассмотрим такие ИНС более подробно.

В многослойных сетях нейроны объединяются в слои таким образом, что нейроны одного слоя имеют одинаковые входные сигналы. Число нейронов в слое может быть произвольным и зависит в большей степени от решаемой задачи, чем от количества нейронов в других слоях. Внешние (входные) сигналы подаются на нейроны входного слоя (его часто нумеруют как нулевой), а выходами сети являются выходные сигналы нейронов последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети может быть один или несколько скрытых слоев. Выходные сигналы нейронов слоя (q) являются входными сигналами следующего слоя (q+1). Одной из основных характеристик многослойных нейронных сетей является число слоев. В дальнейшем для описания структуры многослойных ИНС будем использовать количество присутствующих в ней скрытых слоев.

По структуре многослойные ИНС могут представлять как ИНС прямого распространения (рис. 3), так и рекуррентные.

ОБУЧЕНИЕ ИНС

Для ИНС прямого распространения без скрытых слоев с n_1 входами и n_0 выходами зависимость выходных сигналов ИНС от входных можно представить следующим образом (будем считать, что одномерный вектор представлен как вектор-столбец):

$$Y = G(X) = F(W^T X),$$

где $X = \{x_i, i = \overline{1, n_1}\}$ и $Y = \{y_j, j = \overline{1, n_0}\}$ – векторы входных и выходных сигналов соответственно, $W = \{w_{ij}, i = \overline{1, n_1}, j = \overline{1, n_0}\}$ – матрица весов межнейронных связей, в которой элемент w_{ij} соответствует весу связи от i-го входного нейрона к j-му выходному, T – операция транспонирования; F(·) – вектор-функция выходного сигнала слоя нейронов, вид которой зависит от выбранной активационной функции (см. таблицу 1).

При добавлении скрытого слоя с n_H нейронами в структуру ИНС значение выражения для выходных сигналов ИНС изменится:

$$Y = G(X) = F(W^{(1)^T} Y^{(1)}) = F(W^{(1)^T} F(W^{(0)^T} Y^{(0)})),$$

где $W^{(0)} = \{w_{ij}^{(0)}, i = \overline{1, n_1}, j = \overline{1, n_H}\}$ и $W^{(1)} = \{w_{ij}^{(1)}, i = \overline{1, n_H}, j = \overline{1, n_0}\}$ – матрицы весов межнейронных связей для скрытого и выходного слоев 9 соответственно.

Добавление дополнительных скрытых слоев соответствующим образом изменит вид функции выходного сигнала ИНС. Однако для выбранной структуры ИНС, при условии постоянного вектора входных сигналов X и фиксированной функции активации нейронов, значение выходного сигнала ИНС зависит только от значений весов связей.

Для решения практических задач важным является поиск такого набора значений весов межнейронных связей, при котором выходные сигналы ИНС изменяются в определенной зависимости от предъявляемого вектора входных сигналов. Процесс подстройки весов межнейронных связей называется **обучением нейронной сети**.

Обучение с учителем (supervised learning) подразумевает использование заранее сформированного множества обучающих примеров. Каждый пример содержит вектор входных сигналов и соответствующий вектор эталонных выходных сигналов, которые зависят от поставленной задачи. Данное множество называют обучающей выборкой или обучающим множеством. Обучение нейронной сети направлено на такое изменение весов связей ИНС, при котором значение выходных сигналов ИНС как можно меньше отличаются от требуемых значений выходных сигналов для данного вектора входных сигналов

В дальнейшем будем рассматривать обучение с учителем, которое можно описать следующей последовательностью действий:

1. Подготовка обучающей выборки.
2. Выбор структуры ИНС.
3. Настройка весов связей (обучение ИНС).

В процессе обучения на вход нейронной сети предъявляются данные из обучающей выборки и, в соответствии со значениями выходных сигналов, определяющих ошибку функционирования сети, производится коррекция весов связей.

В результате обучения должна быть получена нейронная сеть, которая без перенастройки весов связей формирует с требуемой погрешностью выходные сигналы Y при подаче на вход сети любого набора входных сигналов из обучающего множества.

Качество обученной нейронной сети проверяется с использованием данных, не участвовавших в процессе обучения.

Опишем каждый этап более подробно.

Обучающая выборка состоит из множества пар векторов $(X_i, Y_i), i = \overline{1, K}$, где K – количество примеров в обучающей выборке, $X_i = \{x_{ij}, j = \overline{1, n_1}\}$ – вектор входных сигналов; $Y_i = \{y_{ij}, j = \overline{1, n_0}\}$ – вектор соответствующих X_i выходных сигналов.

После того как определены обучающие данные и структура сети, производится настройка весов связей. Веса инициализируются случайными значениями, как правило, из диапазона [– 0,01; 0,01] – [– 0,1; 0,1], для того, чтобы впоследствии избежать возможного насыщения функций активации нейронов и повышенной чувствительности выхода ИНС к несущественным (в пределах погрешности) изменениям сигналов.

В процессе обучения происходит коррекция (подстройка) значений весов связей. При этом пары векторов $(X_i, Y_i), i = \overline{1, K},$ из обучающего множества могут предъявляться многократно.

Один «прогон» всех наборов данных из обучающей выборки вместе с коррекцией весов составляет одну эпоху обучения. Типичная длительность обучения может составлять от десятков до нескольких десятков тысяч эпох в зависимости от поставленной задачи, структуры ИНС, качества самих данных и выбранного алгоритма подстройки весов связей.

Рассмотрим распространенный алгоритм обратного распространения ошибки и его модификацию, использующую инерционность.

АЛГОРИТМ ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

В настоящее время существует множество алгоритмов обучения. Наиболее известный из них – **алгоритм обратного распространения ошибки.** Данный алгоритм используется для минимизации отклонения реальных значений выходных сигналов нейронной сети от требуемых.

В качестве функции ошибки ИНС будем рассматривать следующую величину:

$$E(w) = \frac{1}{2}\sum_i E_i = \frac{1}{2}\sum_{i,k}\left(f_{i,k} - y_{i,k}\right)^2, \quad (1)$$

где $f_{i,k}$ – значение выходного сигнала *k*-го выходного нейрона сети при подаче на её входы *i*-го набора обучающих данных, $y_{i,k}$ – требуемое значение выходного сигнала *k*-го выходного нейрона для *i*-го набора данных для обучения. Суммирование ведется по всем нейронам выходного слоя и по всем наборам данных из обучающей выборки. Обучение ИНС направлено на минимизацию функции E(w).

Минимизация **методом градиентного спуска** обеспечивает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(q)} = \eta \frac{\partial E}{\partial w_{ij}}, \quad (2)$$

где $\Delta w_{ij}^{(q)}$ – величина изменения веса связи, соединяющей *i*-й нейрон (*q*-1) слоя с *j*-м нейроном слоя *q*, η – коэффициент скорости обучения, $0 < \eta < 1$.

Таким образом, вес связи изменяется пропорционально её вкладу в значение ошибки нейрона, для которого эта связь является входной, т.к. частная производная по весу $\frac{\partial E}{\partial w_{ij}}$ показывает зависимость скорости изменения функции ошибки E от изменения этого веса.

Опустим преобразования формулы (2) и представим сразу конечный результат (3). Изменение веса связи определяется следующим образом:

$$\Delta w_{ij}^{(q)} = -\eta \delta_j x_i, \quad (3)$$

где δ_j – значение ошибки j-го нейрона в слое q, x_i – значение i-го входного сигнала для j-го нейрона слоя q. Данная формула применима и для настройки смещений нейронов, только вместо x_i необходимо подставить «1».

Значение ошибки нейрона определяется в зависимости от его положения в сети.

Для нейронов выходного слоя

$$\delta_i = \left(f_{ik}(S)\right)'\left(f_{ik} - y_{ik}\right) \tag{4}$$

где y_{ik} – требуемое, а f_{ik} – фактическое значение выходного сигнала k-го нейрона для i-го набора данных из обучающей выборки, $\left(f_{ik}(S)\right)'$ – значение производной активационной функции k-го нейрона для i-го набора обучающих данных.

Если нейрон принадлежит одному из скрытых слоев, то

$$\delta_i^{(q)} = \left(f_i^{(q)}(S)\right)'\sum_j w_{ij}\delta_j^{(q+1)}, \tag{5}$$

где $\delta_i^{(q)}$ – ошибка j-го нейрона в слое q, $\delta_j^{(q+1)}$ – ошибка j-го нейрона в (q+1) слое, w_{ij} – вес связи, соединяющей эти нейроны, $\left(f_i^{(q)}(S)\right)'$ – значение производной активационной функции i-го нейрона слоя q.

Таким образом, значение ошибки нейрона пропорционально его «влиянию» на величины ошибок нейронов следующего слоя, а также скорости изменения его выходного сигнала для k-го набора обучающих данных.

Рассмотрим ИНС с нейронами с лог-сигмоидными функциями активации:

$$f(S) = \frac{1}{1+e^{-aS}}, \tag{6}$$

где a – константа, S – взвешенная сумма входных сигналов нейрона, тогда

$$f'(S) = \left(\frac{1}{1+e^{-aS}}\right)' = a\frac{1}{1+e^{-aS}}\cdot\frac{e^{-aS}}{1+e^{-aS}} = af(S)\left(1-f(S)\right) \tag{7}$$

и формулы (4), (5) соответственно примут вид

$$\delta_i = af_{ik}(1 - f_{ik})(f_{ik} - y_{ik}), \quad (8)$$

$$\delta_i^{(q)} = af_i(1 - f_i)\sum_j w_{ij}\delta_j^{(q+1)} \quad .(9)$$

Для реализации алгоритма обратного распространения ошибки может быть использована следующая последовательность действий:

1. Предъявление очередного набора из обучающей выборки на вход нейронной сети.
2. Вычисление выходного сигнала сети.
3. Определение величин ошибок нейронов выходного слоя по формуле (4) или (8).
4. Определение величин ошибок нейронов скрытых слоев по формулам (5) или (9).
5. Однократная коррекция весов связей.
6. Если в обучающей выборке есть неиспользованные в данной эпохе наборы данных, то переход на шаг 1.
7. Подсчет ошибки сети по формуле (1). Если ошибка меньше заданной, то конец обучения, иначе, начало новой эпохи обучения и переход на шаг 1.

Отметим, что алгоритм обратного распространения ошибки применим только для нейронных сетей, содержащих нейроны с дифференцируемой функцией активации. Т.е. рассмотренный алгоритм не подходит для настройки сетей, построенных на нейронах с пороговыми функциями активации. Однако во многих случаях в нейронных сетях используются нейроны с сигмоидальными функциями активации (лог-сигмоидные функции и гиперболический тангенс), что дает возможность применять для настройки и обучения таких сетей градиентные алгоритмы.

ПРИМЕР РАБОТЫ И ОБУЧЕНИЯ НЕЙРОННОЙ СЕТИ

Рассмотрим работу и обучение нейронной сети на примере.

Дана сеть с 2 входами, двумя скрытыми слоями, в каждом из которых по 2 нейрона, и одним выходом. Функции активации нейронов – лог-сигмоидные (6), a = 1. Скорость обучения η примем равной 0,8. Необходимо реализовать с помощью ИНС логическую операцию «ИЛИ». Таблица

истинности представлена ниже, где x_1 и x_2 – входные переменные, а y – значение результата.

Таблица 2

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Предположим, что на некотором этапе обучения получена сеть, соответствующая изображенной на рис. 4. Допустим также, что на некотором шаге веса сети равны значениям из таблицы 3. Смещения нейронов обозначены как веса с индексом «–».

Таблица 3

Номер слоя	**Индекс нейрона**	**Индекс веса**	**Вес**
1	0	-	0,02
1	0	0	0,12
1	0	1	0,35
1	1	-	-0,015
1	1	0	-0,5
1	1	1	0,24
2	0	-	-0,084
2	0	0	-0,33
2	0	1	0,27
2	1	-	0,037
2	1	0	-0,08
2	1	1	0,79
3	0	-	0,04
3	0	0	0,062
3	0	1	0,64

Роль обучающей выборки в рассматриваемом случае играют данные из табл. 2, в которой каждая строка соответствует одному набору данных.

Пусть на вход нейронной сети поданы входные сигналы из 2-го обучающего набора (x_1 = 0, x_2 = 1). Тогда, в соответствии с таблицей истинности (табл. 2), выход сети должен равняться 1. Действительные значения выходов нейронов приведены в таблице 4. На рисунке 4 поясняется вычисление выходных сигналов ИНС.

Таблица 4. Значения выходных сигналов нейронов

Номер слоя	Индекс нейрона	Значение выхода
1	0	0,591
1	1	0,556
2	0	0,468
2	1	0,606
3	0	0,612

Требуемое значение выхода сети «1», следовательно, ошибка сети на данном этапе равна:

$$E_2 = 0{,}5 * (1 - 0{,}612)^2 = 0{,}075.$$

Ошибка выходного нейрона, согласно (8):

$$\delta_0^{(3)} = 0{,}612 * (1 - 0{,}612) * (0{,}612 - 1) = -0{,}092.$$

Ошибки нейронов 2 скрытого слоя по формуле (9):

$$\delta_0^{(2)} = 0{,}468 * (1 - 0{,}468) * (-0{,}092) * 0{,}062 = -0{,}001,$$
$$\delta_1^{(2)} = 0{,}606 * (1 - 0{,}606) * (-0{,}092) * 0{,}64 = -0{,}014.$$

Ошибки нейронов 1 скрытого слоя по формуле (9)

$$\delta_0^{(1)} = 0{,}591 * (1 - 0{,}591) * ((-0{,}001) * (-0{,}33) + (-0{,}014) * (-0{,}08)) = 0{,}0002,$$
$$\delta_1^{(1)} = 0{,}556 * (1 - 0{,}556) * ((-0{,}001) * 0{,}27 + (-0{,}014) * 0{,}79) = -0{,}003.$$

Вес первой связи выходного нейрона изменится, согласно (3), на следующую величину:

$$\Delta w_{00}^{(3)} = -0{,}8\ \delta_0^{(3)}\ f_0^{(2)} = -0{,}8 * (-0{,}092) * 0{,}468 = 0{,}034.$$

Таким образом, новое значение веса будет равно:

$$w_{00}^{(3)}(t+1) = w_{00}^{(3)}(t) + \Delta w_{00}^{(3)} = 0{,}062 + 0{,}034 = 0{,}096.$$

Аналогично производится коррекция остальных весов связей ИНС:

$$w_{10}^{(3)}(t+1) = 0{,}64 - 0{,}8 * (-0{,}092) * 0{,}606 = 0{,}685;$$

$$w_{-0}^{(3)}(t+1) = 0{,}04 - 0{,}8 * (-0{,}092) * 1 = 0{,}114;$$

$$w_{00}^{(2)}(t+1) = -0{,}33 - 0{,}8 * (-0{,}001) * 0{,}591 = -0{,}329;$$

$$w_{10}^{(2)}(t+1) = 0{,}27 - 0{,}8 * (-0{,}001) * 0{,}556 = 0{,}27;$$

$$w_{-0}^{(2)}(t+1) = -0{,}084 - 0{,}8 * (-0{,}001) * 1 = -0{,}083;$$

$$w_{01}^{(2)}(t+1) = -0{,}08 - 0{,}8 * (-0{,}001) * 0{,}591 = -0{,}073;$$

$$w_{11}^{(2)}(t+1) = 0{,}79 - 0{,}8 * (-0{,}014) * 0{,}556 = 0{,}796;$$

$$w_{-1}^{(2)}(t+1) = 0{,}037 - 0{,}8\ (-0{,}014) * 1 = 0{,}048;$$

$$w_{00}^{(1)}(t+1) = 0{,}12 - 0{,}8 * 0{,}0002 * 0 = 0{,}12;$$

$$w_{10}^{(1)}(t+1) = 0{,}35 - 0{,}8 * 0{,}0002 * 1 = 0{,}35;$$

$$w_{-0}^{(1)}(t+1) = 0{,}02 - 0{,}8 * 0{,}0002 * 1 = 0{,}02;$$

$$w_{01}^{(1)}(t+1) = -0{,}5 - 0{,}8 * (-0{,}003) * 0 = -0{,}5;$$

$$w_{11}^{(1)}(t+1) = 0{,}24 - 0{,}8 * (-0{,}003) * 1 = 0{,}242;$$

$$w_{-1}^{(1)}(t+1) = -0{,}015 - 0{,}8 * (-0{,}003) * 1 = -0{,}013.$$

В случаях, когда вместо индекса начального нейрона связи стоит символ «–», производится настройка смещения нейрона. Т.е. запись $w_{-1}^{(2)}(t+1)$ означает новое значение смещения у нейрона с индексом 1 во втором скрытом слое. Использованные для расчета значения ошибок выходов нейронов показаны на рис. 5.

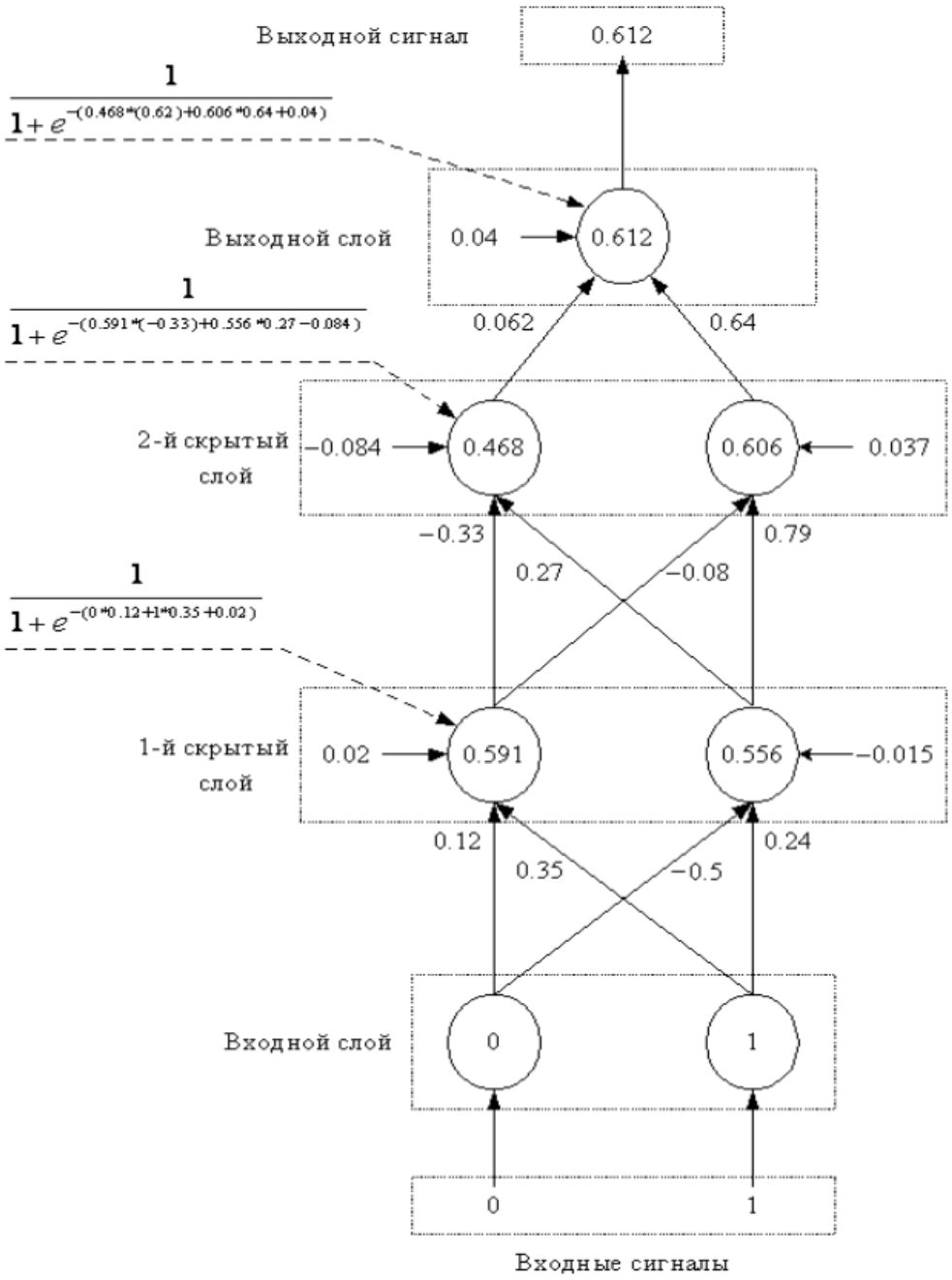


Рисунок 4. Прямой проход для вычисления выхода ИНС

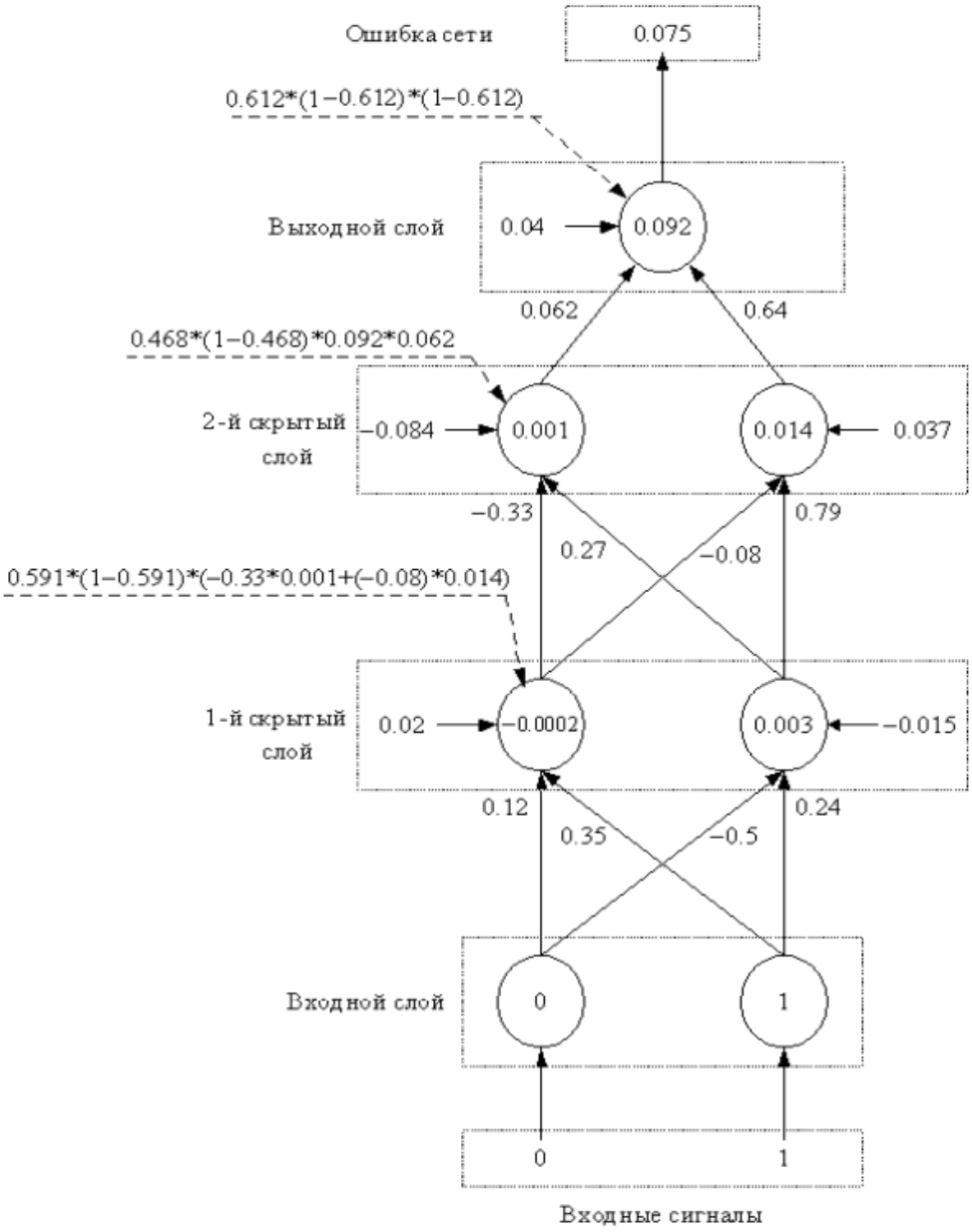


Рисунок 5. Обратный проход для вычисления ошибок нейронов

Значения выходных сигналов ИНС при повторном подсчете выходов нейронов сети при использованном для коррекции весов наборе обучающих данных представлены в табл. 5 и на рис. 6.

Таблица 5. Значения выходных сигналов нейронов после корректировки весов связей ИНС

Номер слоя	Индекс нейрона	Значение выхода
1	0	0,591
1	1	0,557
2	0	0,468
2	1	0,610
3	0	0,640

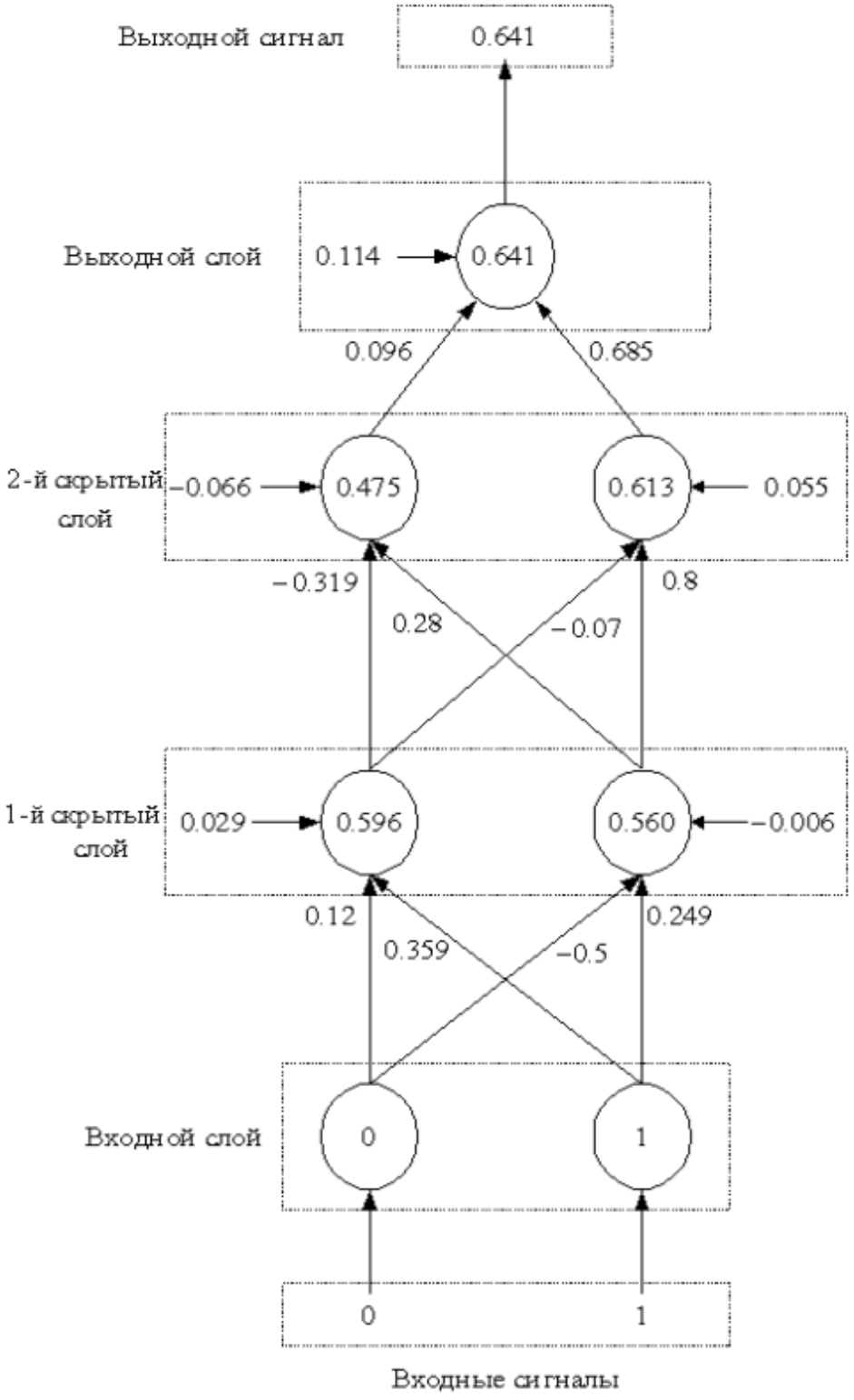


Рисунок 6. Прямой проход после коррекции весов

Новое значение ошибки ИНС для 2-го набора обучающих данных равно:

$$E_2 = 0{,}5\ (1 - 0{,}641)^2 = 0{,}065.$$

Напомним, что значение ошибки E_2 до коррекции весов равнялось 0,075. Таким образом, в результате одного шага обучения значение ошибки уменьшилось на 0,01. Для дальнейшего обучения ИНС на вход подается следующий вектор данных из обучающего множества и производится коррекция весов связей с учетом расхождения выходного сигнала ИНС и вектора требуемых выходных сигналов, соответствующего входному. Обучение ИНС продолжается до тех пор, пока ошибка выхода ИНС, вычисленная для всех наборов данных из обучающей выборки, по значению не будет меньше требуемой.

Задание:

сделать еще одну итерацию прямого прохода, коррекции весов и прямого прохода после коррекции весов со значениями переменных входной выборки x_1=1, x_2=1 и y=1. В качестве весов можно взять веса, полученные на рисунке 6.

Реализация нейронных сетей на R

Обучим многослойный персептрон для примера из книги Ланц «». Речь идет об исследовании влияния на прочность бетона (при сжатии) восьми характеристик, описывающих компоненты, используемые в составе бетонной смеси, таким как: количество (в килограммах на кубический метр) цемента, воды, разных добавок, крупных и мелких заполнителей типа щебня и песка, используемых в готовом продукте, а также время схватывания (в днях).

Необходимо загрузить таблицу с данными concrete.csv из репозитория http://archive.ics.uci.edu/ml

С помощью команды

str(concrete)

можно изучить структуру таблицы.

Необходимо изменить масштаб данных с помощью их нормализации или стандартизации. Если распределение данных соответствует нормальному распределению колоколообразной кривой, то имеет смысл использовать стандартизацию с помощью встроенной функции scale(). Если же распределение данных близко к равномерному или сильно отличается от нормального, то более подходящей может быть нормализация к диапазону от 0 до 1 с помощью формулы

$$x_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}.$$

В данном случае будем использовать последний вариант.

```
normalize<-function (x) {
  return((x-min(x))/(max(x)-min(x)))
}
```

После того как будет выполнен этот код, можно применить функцию normalize() ко всем столбцам выбранного фрейма данных с помощью функции lapply():

```
concrete_norm<-as.data.frame(lapply(concrete, normalize))
```

Разделим данные на обучающую выборку, включающую в себя 75 % всех примеров, и тестовую, состоящую из 25 %. Используемый CSV-файл отсортирован в случайном порядке, поэтому нам остается лишь разделить его на две части:

```
#делим набор на обучающую и контрольную выборки
concrete_train<-concrete_norm[1:773,]
concrete_test<-concrete_norm[774:1030,]
```

Мы будем использовать тренировочный набор данных для построения нейронной сети и тестовый набор данных для оценки того, насколько хорошо модель обобщается для будущих результатов.

Построим многослойную нейронную сеть прямого распространения. Для этого воспользуемся пакетом neuralnet.

```
model <- neuralnet(Y~ predictors, data = mydata, hidden = 1,
act.fct = "logistic")
```

Параметр	Назначение
Y	Отклик
predictors	R-формула, определяющая признаки из фрейма данных mydata, которые будут использоваться при прогнозировании
data	фрейм данных, которому принадлежат Y и predictors
hidden	количество нейронов в скрытом слое (по умолчанию 1) (для описания нескольких скрытых слоев используется вектор целых чисел, например c(2, 2));
act.fct	функция активации: «logistic» или «tanh» (также может быть использована любая другая дифференцируемая функция).

Другие параметры функции

- threshold ~ числовое значение, определяющее порог для частных производных функции ошибки как критерий остановки.
- stepmax ~ максимальные шаги для обучения нейронной сети. Достижение этого максимума приводит к остановке процесса обучения нейронной сети.
- rep ~ количество повторений для обучения нейронной сети
- startweights ~ вектор, содержащий начальные значения для весов. Весы не будут случайным образом инициализированы
- learningrate.limit ~ вектор или список, содержащий самый низкий и самый высокий предел для скорости обучения. Используется только для RPROP и GRPROP
- learningrate.factor ~ вектор или список, содержащий коэффициенты умножения для верхней и нижней скорости обучения. Используется только для RPROP и GRPROP

- learningrate ~ числовое значение, определяющее скорость обучения, используемую традиционным обратным распространением. Используется только для традиционного backpropagation
- lifesign ~ строка, определяющая, насколько функция будет печататься во время вычисления нейронной сети. ‘none’, ‘minimum’ или ‘full’
- lifesign.stepr ~ целое число, определяющее шаг, чтобы напечатать минимальный порог в режиме полной жизни
- algorithm ~ строка, содержащая тип алгоритма для вычисления нейронной сети. Возможны следующие типы: backprop, rprop +, rprop-, sag или slar. «backprop» относится к backpropagation, «rprop +» и «rprop-» относятся к отказоустойчивой backpropagation с возвратом backtracking и без него, тогда как «sag» и «slr» вызывают использование модифицированного глобально конвергентного алгоритма (grprop).
- err.fct ~ дифференцируемой функции, которая используется для вычисления ошибки. В качестве альтернативы можно использовать строки sse ‘и’ ce ’, которые обозначают сумму квадратов ошибок и кросс-энтропии
- act.fct ~ дифференцируемой функцией, которая используется для сглаживания результата поперечного произведения ковариата или нейронов и весов. Кроме того, для логистической функции и касательной гиперболичности возможны строки, «логистика» и «тань»
- linear.output ~ логичной. Если act.fct не следует применять к выходным нейронам, установите линейный выход в TRUE, иначе FALSE
- exclude ~ вектор или матрицу, определяющую весовые коэффициенты, которые исключаются из расчета. Если задано как вектор, то точное положение весов должно быть известно. Матрица с n-строками и тремя столбцами исключает n весов, где первый столбец обозначает слой, второй столбец для входного нейрона и третий столбец для выходного нейрона веса
- constant.weights ~ вектор, определяющий значения весов, которые исключаются из учебного процесса и рассматриваются как исправление
- likelihood ~ логичной. Если функция ошибки равна отрицательной функции логарифмического правдоподобия, будут вычислены информационные критерии AIC и BIC. Кроме того, использование trust.interval имеет смысл

Функция возвращает объект нейронной сети, который можно использовать для прогнозирования.

Для нашего примера

```
model<-neuralnet(data=concrete_train, strength~.)
plot(model)
```

Функция plot() позволяет визуализировать обученную нейронную сеть (рисунок 7).

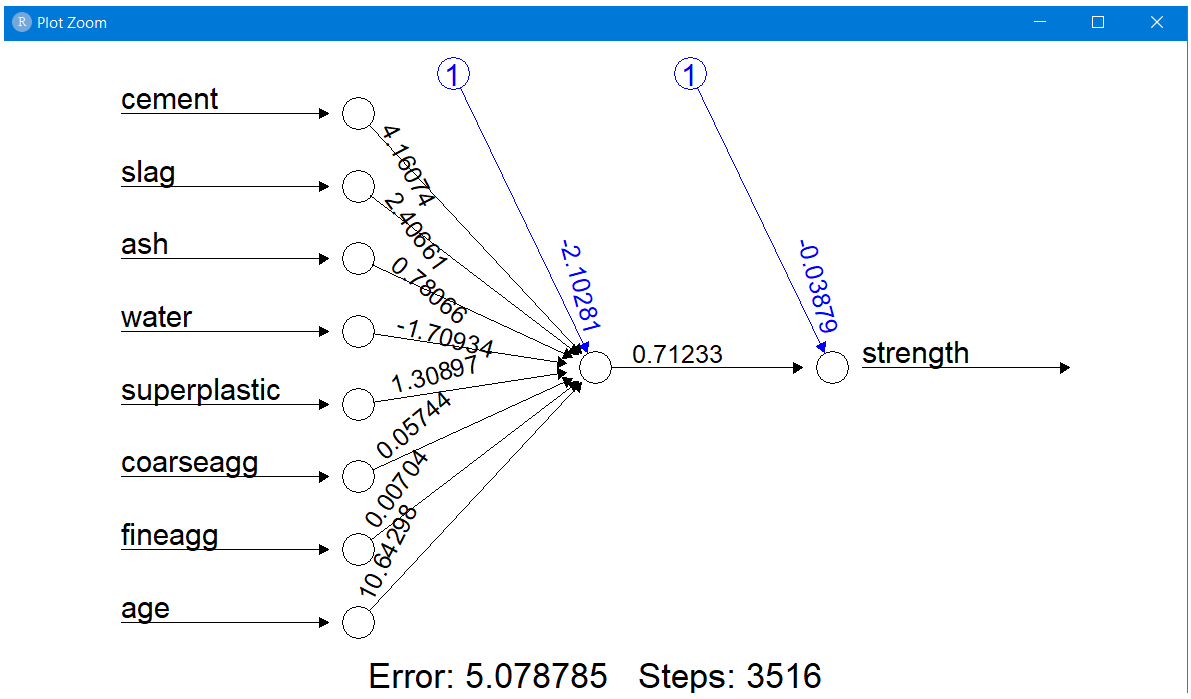


Рисунок 7

В этой простой модели есть по одному входному узлу для каждого из восьми признаков, затем идет один скрытый и один выходной узел, который и дает прогноз прочности бетона. На диаграмме также изображены веса для каждой связи и величина смещения, обозначенная для узлов, помеченных цифрой 1. В нижней части рисунка отображаются количество шагов обучения и величина ошибки (1).

С помощью команды model$result.matrix можно вывести подобранные веса нейронной сети и ошибки:

```
                                    [,1]
error                       5.082550e+00
reached.threshold           9.219327e-03
steps                       1.560000e+03
Intercept.to.1layhid1       1.745719e+00
cement.to.1layhid1         -4.029551e+00
slag.to.1layhid1           -2.275824e+00
ash.to.1layhid1            -6.969016e-01
water.to.1layhid1           1.931298e+00
superplastic.to.1layhid1   -1.260822e+00
coarseagg.to.1layhid1       7.383848e-02
fineagg.to.1layhid1         1.783054e-01
age.to.1layhid1            -1.064142e+01
Intercept.to.strength       6.738455e-01
1layhid1.to.strength       -7.093354e-01
```

Рисунок 8

Прогнозирование осуществляется с помощью функции compute():

p <- compute(model, test)

model — модель, обученная с помощью функции neuralnet();

test — фрейм данных, содержащий контрольную выборку с теми же признаками, что и в обучающей выборке.

Функция возвращает список, состоящий из двух компонентов:

$neurons, где хранятся нейроны для каждого слоя сети,

и $net.result, где хранятся значения, спрогнозированные с помощью данной модели.

С помощью кода

```
p<-compute(model, concrete_test[1:8])
p$net.result
```

найдем расчетные значения переменной strength и поместим их рядом с фактическими значениями переменной strength (нормализованными)

```
tabl<-data.frame(act_strength=concrete_test$strength,
                 predict_strength=p$net.result)
```

Фрагмент таблицы приведен на рисунке 9.

	act_strength	predict_strength
774	0.34645571	0.32802006
775	0.52410614	0.46506995
776	0.27619285	0.23797239
777	0.86321166	0.67292788
778	0.42319671	0.45885738
779	0.45010589	0.47235377
780	0.41771521	0.47743110
781	0.47115984	0.58483928
782	0.36240189	0.31711189

Измерим корреляцию между прогнозируемым и истинным значением прочности бетона. Если прогнозируемые и фактические значения будут сильно коррелировать, то, вероятно, модель окажется полезной для

определения прочности бетона.

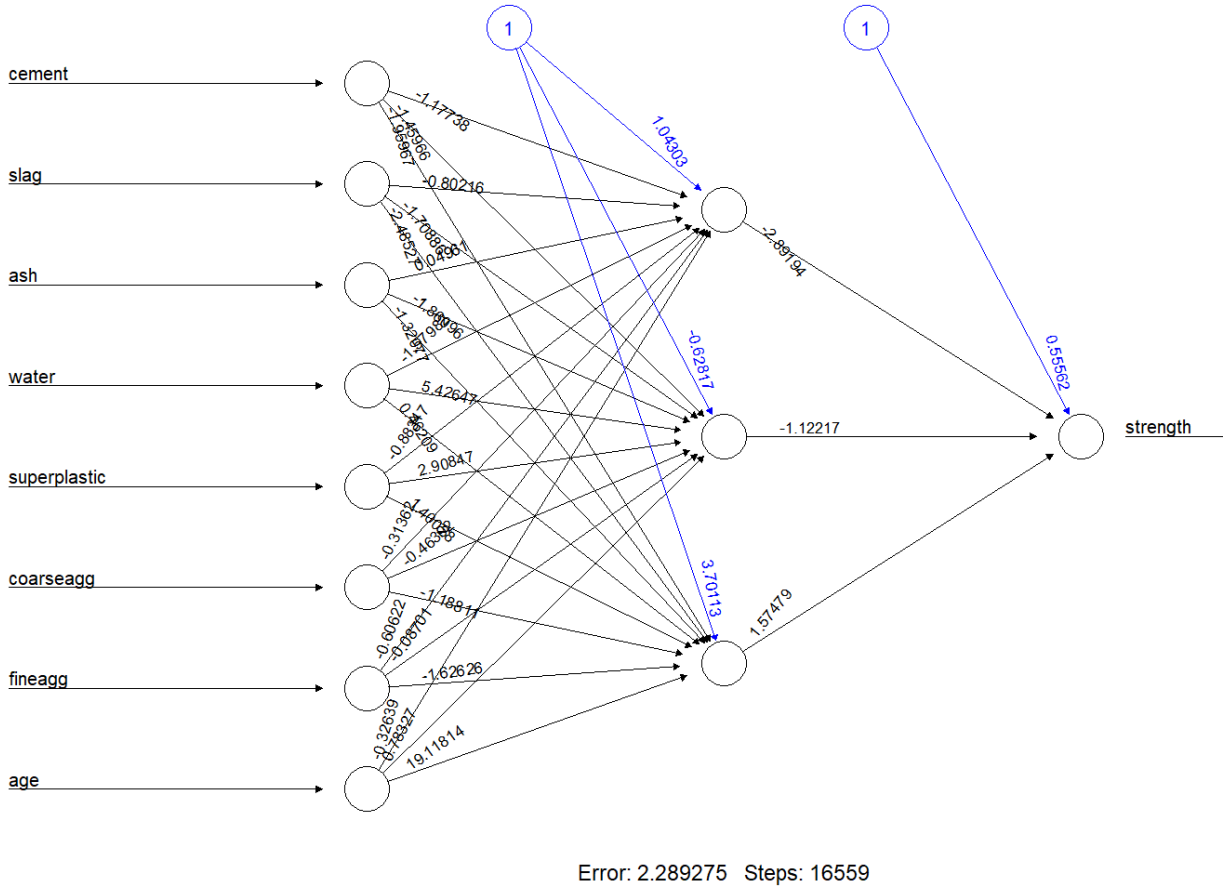
```
cor(concrete_test$strength,p$net.result)
```

Коэффициент корреляции равен примерно 0,806.

Это означает, что модель достаточно хорошо работает даже с единственным скрытым узлом. Учитывая, что мы использовали только один скрытый узел, вполне вероятно, что можно улучшить эффективность модели.

Увеличим количество узлов в скрытом слое до трех. Для этого в функции neuralnet() параметр hidden сделаем равным 3 (если бы нам потребовалось сделать два скрытых слоя, в первом слое 2 нейрона, во втором 3, нужно было бы указать hidden=c(2, 3))

```
model1<-neuralnet(data=concrete_train, strength~., hidden = 3)
plot(model1)
```



Новые веса

```
> model1$weights
[[1]]
[[1]][[1]]
             [,1]        [,2]       [,3]
 [1,]  1.04303206 -0.62816734  3.7011273
 [2,] -1.17737561 -1.45966055 -1.9596683
 [3,] -0.80216090 -1.70885528 -2.4852661
 [4,]  0.04961105 -1.86095600 -1.3207685
 [5,] -1.57981760  5.42647237  0.4620876
 [6,] -0.88347097  2.90847018  1.4002813
 [7,] -0.31362253 -0.46391791 -1.1881122
 [8,] -0.60622422 -0.08700768 -1.6262567
 [9,] -0.32638996  0.78327240 19.1181413

[[1]][[2]]
           [,1]
[1,]  0.5556246
[2,] -2.8919402
[3,] -1.1221727
[4,]  1.5747869
```

Вычислим коэффициент корреляции между фактическими значениями прочности бетона в контрольной выборке и рассчитанными

```
p<-compute(model1, concrete_test[1:8])
cor(concrete_test$strength,p$net.result)
```

Коэффициент корреляции равен примерно 0,918, что превышает предыдущее значение.

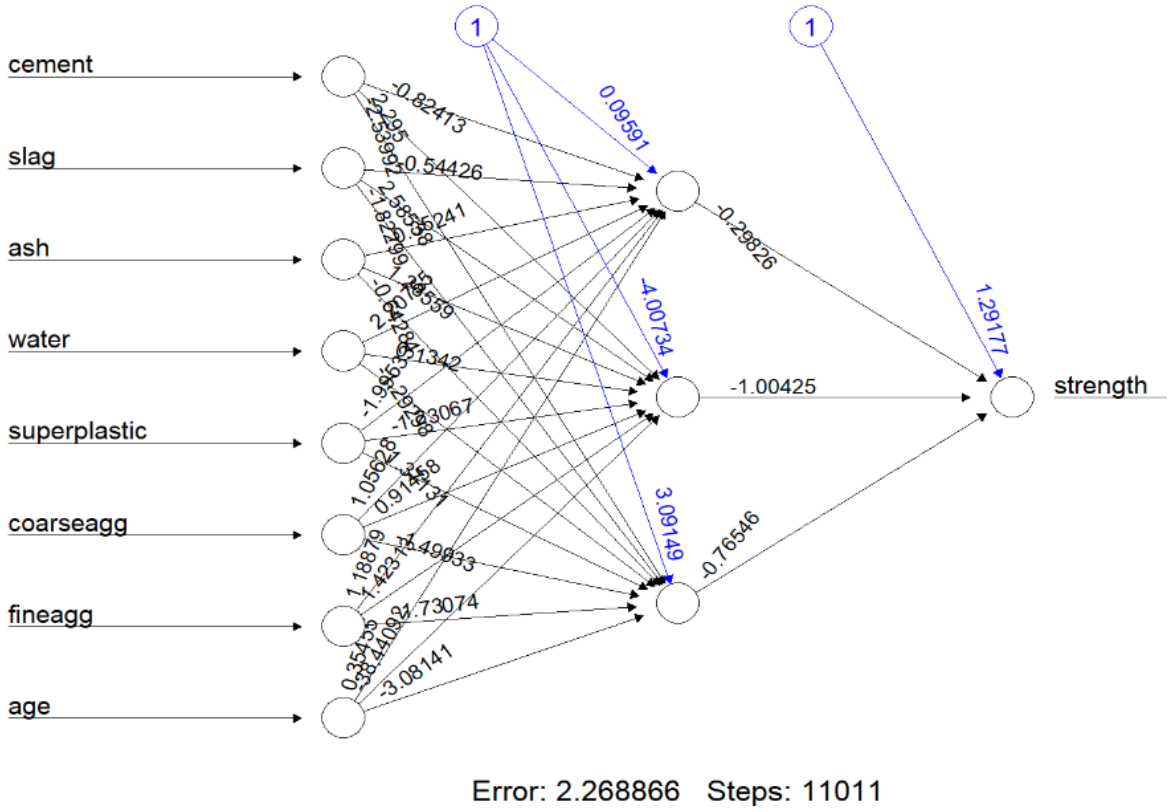
Рассмотрим еще случай, когда требуется в качестве функции активации использовать свою функцию. Например, $f(x)=\log(1+e^x)$, напишем для нее код

```
smReLu<-function(x){
log(1+exp(x))
}
```

Такую функцию активации можно предоставить на вход neuralnet() с помощью параметра act.fct

```
40 model2<-neuralnet(data=concrete_train, strength~., hidden = 3,
41                    act.fct=smReLu)
42 plot(model2)
43 model2$weights
44 p<-compute(model2, concrete_test[1:8])
45 cor(concrete_test$strength,p$net.result)
```

В результате получим сеть



Значение коэффициента корреляции равно 0,923, количество итераций уменьшилось.

Во всех случаях мы строили модель на нормализованных данных, поэтому расчетные значения прочности бетона получились тоже нормализованными. Приведем их к исходному виду, для этого напишите функцию для денормализации.