Федеральное государственное образовательное бюджетное учреждение высшего образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ» (Финансовый университет)
Новороссийский филиал
Кафедра «Информатика, математика и общегуманитарные науки»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

НЕЙРОННЫЕ СЕТИ И ТЕХНОЛОГИИ И ГЛУБОКОЕ ОБУЧЕНИЕ
Направление подготовки: 38.03.05 Бизнес-информатика
Направленность(профиль): ИТ-менеджмент в бизнесе
Форма обучения: очная
Квалификация (степень) выпускника: Бакалавр

Новороссийск 2021

Оглавление

ГЛАВА 1
НЕЙРОНЫ И ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

1.1 Модель искусственного нейрона

Рассмотрим изображение среза нейрона головного мозга млекопитающих. Условно его можно разделить на три области: дендриты, тело нейрона и аксон как показано на рисунке 1.

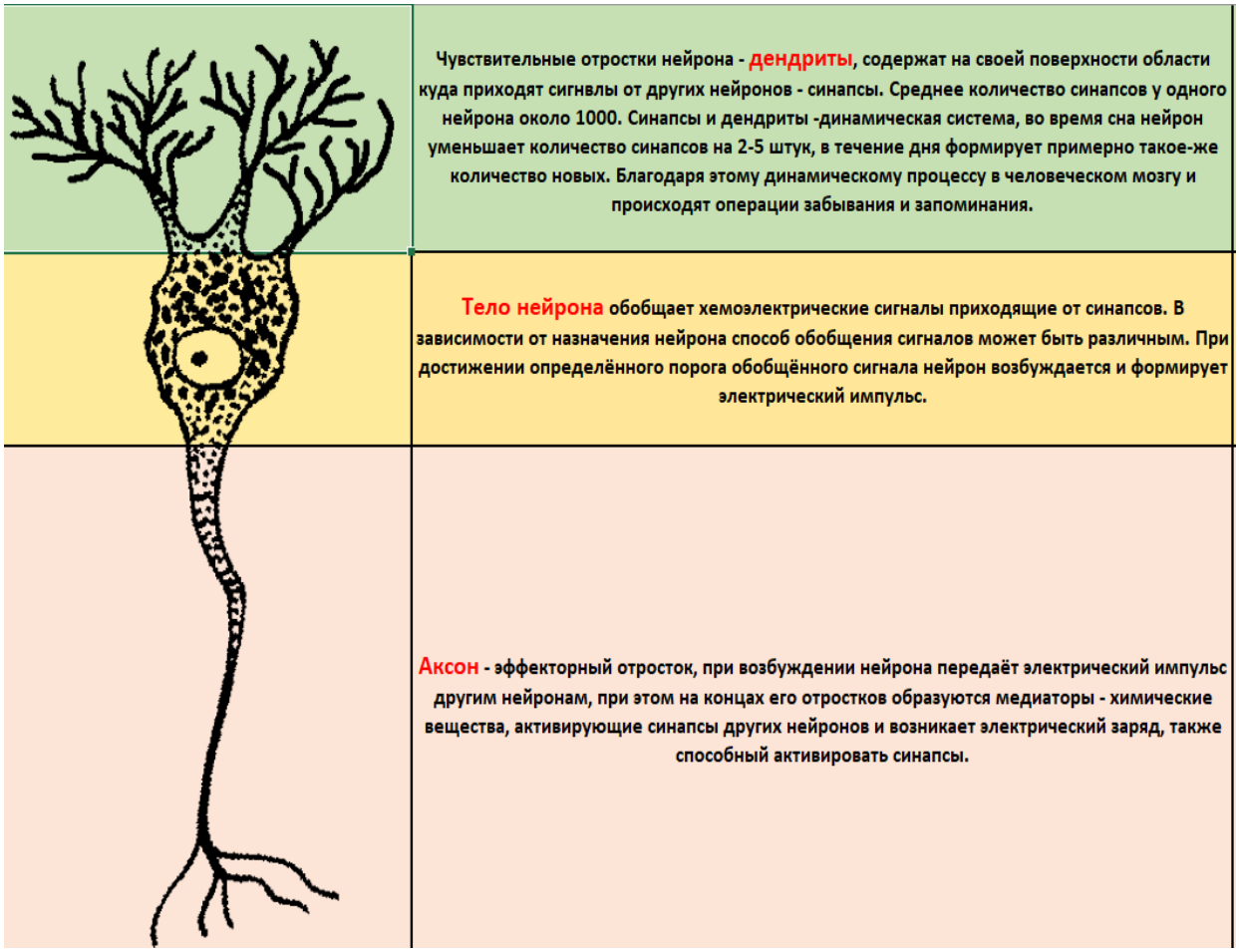


Рисунок 1 – Биологический нейрон

На рисунке 2 представлена формализация нейрона. Приведем описание нейрона с точки зрения биологии и математики. Искусственный нейрон представляет собой максимально упрощённую математическую модель, представленную суперпозицией функций рецепции, агрегации и активации.

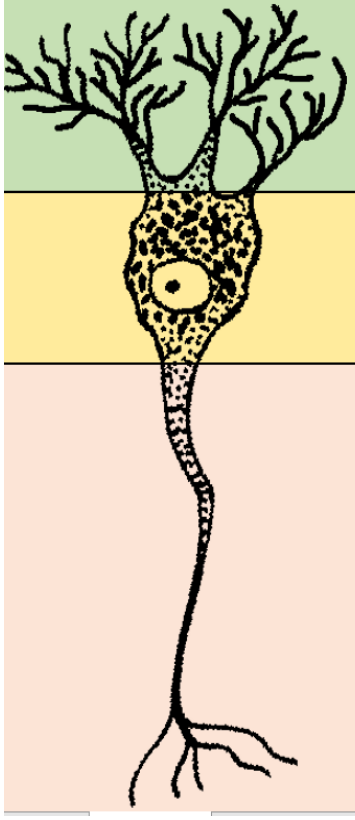
	В биологии	В математике	
	Эта область называется **рецептивным полем**. Снапсы по функциональному назначению можно разделить на четыре вида: **активирующие аддитивные** (химические и электрические) их сигналы взаимно складываются; **активирующие мультипликативные** (электрохимические) их сигналы взаимно умножаются и соответственно **тормозящие аддитивные** и **тормозящие мультипликативные**.	**Рецептивное поле** x_i - входные сигналы ω_i - множители	**Индуцированное локальное поле обозначается:** υ или v $\upsilon = \Sigma(x_i \omega_i)$
	В зависимости от назначения нейрон может обобщать (агрегировать) совокупный входной сигнал различными способами преимущественно умножая, суммируя или наоборот ослабляя входные сигналы. Такую операцию обобщения сигналов называют **агрегацией**.	**Агрегатор - υ** Σ - сумма Π - произведение X - свёртка C - корреляция	
	В зависимости от длины аксона, степени его изоляции, аккумулирующих свойств, химического состава содержимого, количества и типов эффекторов на выходе он способен передавать возбуждение по разному и с различным характером. Такой процесс передачи возбуждения называют **активацией**.	**Активатор - f** L - линейный H - Хевисайда K - Кронекера σ - Ферхюльста PL - линейно-кусочный th - гиперболический тангенс ReLU - линейный выпрямитель ISRU - обратный корень SoftMax - многомерная сигмоида MaxOut - супремум	**Выход сети обозначается:** y $y = f(\upsilon)$

Рисунок 2 – Формализация нейрона

Степень математической примитивизации современных моделей нейронов чрезвычайно высока. Так, реальный нейрон в каждом синапсе содержит порядка 1000 электрических и химических переключателей сигнала (триггеров). При этом у наиболее близкой к реальности модели Ходжкина-Хаксли только в рецептивном поле насчитывается 24 параметра каждого элемента. Таким образом, математическая формула, описывающая работу одного нейрона будет иметь не менее 24 млн. аргументов. Головной мозг среднего взрослого человека содержит не менее 10^{17} триггерных связей, что примерно соответствует объему всех вычислительных мощностей, используемых человечеством на начало 2020 года.

Нейронная сеть - математическая модель совокупности определённых видов нейронов, связанных между собой определённой топологией.

Единого стандарта математических и графических обозначений для нейронных сетей не существует. Традиционно нейрон обозначают кругом с

вписанными в него в обратном порядке функциями активации, агрегации и рецепции. Если используется стандартный нейрон Уидроу, то все обозначения опускают.

Обратите внимание на изображения моделей нейрона (рис.3). Каждый автор изображает один и тот же вид нейронов по своему. Однако, наиболее информативным является первый (самый верхний) рисунок.

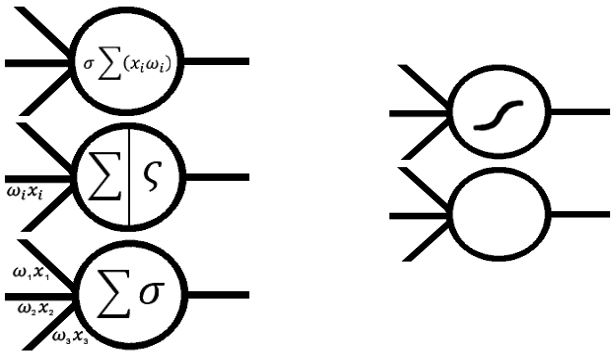


Рисунок 3 – Изображение нейрона Уидроу у различных авторов

Наиболее оптимальным является традиционный формульный (математический) вид записи полной функции нейрона. Также часто используют последовательную запись суперпозиции с разделением функций запятыми. В том случае, когда у нейрона отсутствует обработка одной из функций, вместо неё записывают ноль. Рассмотрим несколько примеров такой записи одного и того же нейрона (рисунок 4).

Традиционная запись	Последовательная запись
$\sigma\sum(x_i\omega_i)$	$x_i\omega_i, \Sigma, \sigma$
$\mathrm{H}\left(\prod(x_i{}^{\omega_i})\right)$	$x_i{}^{\omega_i}, \Pi, \mathrm{H}$
$\mathrm{H}\sum(x_i)$	$0, \Sigma, \mathrm{H}$
Обратите внимание, что синим цветом обозначена рецептивная функция, чёрным - агрегирующая, а фиолетовым активационная.	

Рисунок 4 – Запись нейрона

1.2 Топология нейронных сетей

Топология простых плоских нейронных сетей обычно изображается в форме графа, линии в котором обозначают сигнальные связи, а круглые области - отдельные нейроны. Внутри круглых областей обычно записывают функцию нейрона в определенной форме. Поскольку нейронные сети имеют часто повторяемые однотипные области, их принято разделять на схеме графа, такие области называют слоями. В случае, многомерных слоёв их объединяют в функциональные области.

В нейронной сети принято выделять только входной и выходной слои. Остальные слои или области называют скрытыми или промежуточными, обычно они и несут в себе функционал сети, поэтому их топологии уделяется отдельное внимание.

В качестве обозначающих индексов обычно используют следующий вид записи: i - номер входа, j - номер слоя.

Рассмотрим примеры топологий графов плоских (простых) сетей (рисунок 5).

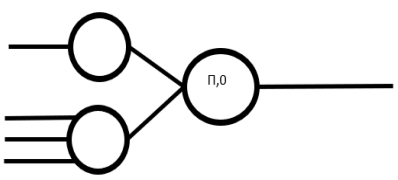
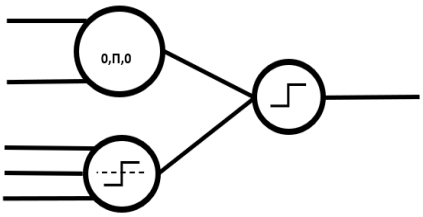
Топология	Формула сети
	$\left(\left(\sigma(x_i\omega_i)\right)\omega_{ij}\right)\cdot\left(\left(\sigma\left(\sum_{i=1}^{i=3}(x_i\omega_i)\right)\right)\omega_{ij}\right)$
Формула сети	**Топология**
$\mathrm{H}\left(\left(\prod_{i=1}^{i=2}x_i\right)\omega_{ij}+\left(\mathrm{K}\left(\sum_{i=1}^{i=3}(x_i\omega_i)\right)\right)\omega_{ij}\right)$	

Рисунок 5 – Примеры топологий сетей

Топология объёмных сетей записывается указанием количества одинаковых нейронов в следующем порядке: ширина входного слоя, высота входного слоя, глубина блока в слоях, ширина выходного слоя, высота выходного слоя (если

топология входов и выходов одинакова - описание ширины и высоты выходов опускают) после чего указывают их тип или тип только нестандартных элементов, если используют нейроны Уидроу, то их тип не указывают (рисунок 6).

Для того, чтобы свободно читать и записывать блочные архитектуры достаточно запомнить несколько простых правил:

- порядок записи блоков всегда один: Ш,В,Г,Ш,В;

- если не обозначен тип активации или агрегации, то это блок классических нейронов Уидроу;

- если не указано количество выходов, то это симметричный блок, то есть выходы повторяют архитектуру входов;

- если перед блоком написано Conv или С, либо после него - Х, то это свёрточный блок, после него всегда указывают тип свёртки;

- архитектура и количество входов следующего блока всегда совпадает с архитектурой выходов предыдущего;

- если не указано иное, то скрытые слои повторяют архитектуру предыдущих слоёв;

- если не указано иное, то внутренняя топология блоков имеет вид "все-со-всеми".

Пример записи объемной топологии	
1. (20,20,3) 2. (20,20,2,10,10)X-FFT 3. (10,10,2,2,2) 4. (2,2,2,1,1)ReLU	1. Блок нейронов Уидроу. Вход: ширина - 20, высота - 20. Глубина: 3. Выход: ширина - 20, высота - 20. 2. Свёрточный блок. Вход: ширина - 20, высота - 20. Глубина: 2. Выход: ширина - 10, высота - 10. Тип свёртки: Быстрое преобразование Фурье. 3. Блок нейронов Уидроу. Вход: ширина - 10, высота - 10. Глубина: 2. Выход: ширина - 2, высота - 2. 4. Блок линейных выпрямителей. Вход: ширина - 2, высота - 2. Глубина: 2. Выход: 1.

Рисунок 6 – Пример записи объемной топологии

Важно. Не следует путать блочную запись архитектуры сети с её с матричным или графовым описанием, используемыми в типовых библиотеках для глубокого обучения нейронных сетей методами обобщённых градиентов второго порядка, таких как, TensorFlow, Theano или Keras. Понятие объёма в

нейронных сетях лишь условность. Поскольку любая мерность входов сети вида X на Y может быть преобразована в 1 на XY, любая архитектура сети является, по сути, плоским графом.

Задание

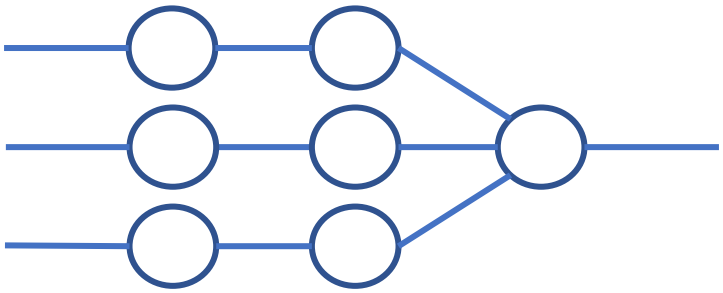
1. Подсчитайте количество рецептивных коэффициентов у сети следующей топологии:

1. (20,20,3)
2. (20,20,2,10,10)X-FFT
3. (10,10,2,2,2)
4. (2,2,2,1,1)ReLU

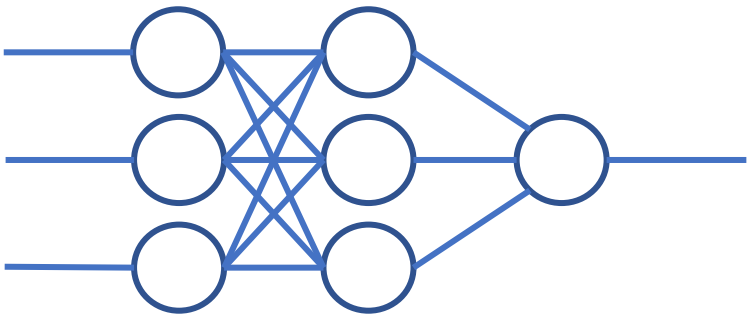
2. Подсчитайте количество рецептивных коэффициентов у сети следующей топологии:

1. (50,20,5)
2. (50,20,2,25,10)
3. (25,10,2,2,2)
4. (2,2,2,1,1)

3. Напишите полную формулу топологии:



4. Напишите блочную запись топологии:



5. Постройте топологический граф по формуле:

$$\sigma\left(\left(\prod_{i=1}^{i=2} x_i\right)\omega_{ij} + \left(\mathrm{H}\left(\sum_{i=1}^{i=2}(x_i\omega_i)\right)\right)\omega_{ij}\right)$$

6. Постройте топологический граф по блочной записи:

(4,1,2,2,1)

(2,1,2,1,1)

7. Постройте топологический граф по блочной записи и подсчитайте количество коэффициентов рецептивного поля:

(5,1,2,3,1)

(3,1,2,2,1)

(2,1,2,1,1)

8. Перед вами образцы цифр. Нарисуйте топологию и напишите формулы предобученной нейронной сети, распознающей любые три цифры кроме единицы на ваш выбор, имеющей четыре двоичных выхода. Нумерация пикселей цифр слева направо, сверху вниз (рисунок 7).

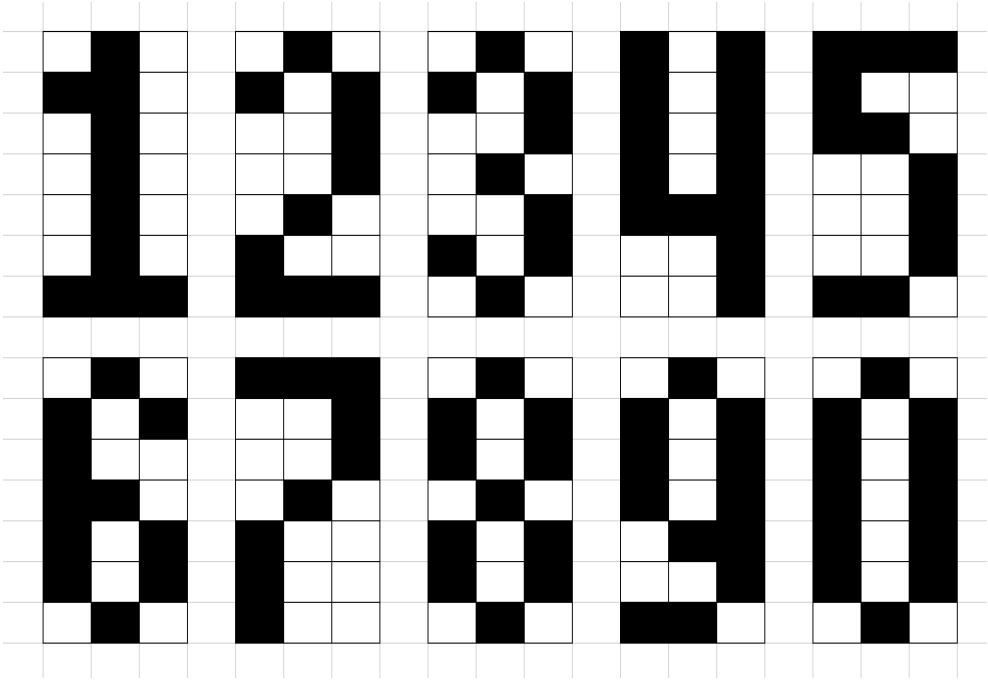
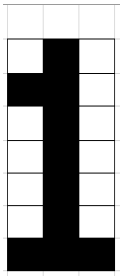


Рисунок 7 – Пиксели цифр

Пример №1 Распознавание цифры 1 (рисунок 8).



	Входы																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	0	1	0	1	1	0	0	1	0	0	1	0	0	1	0	0	1	0	1	1	1
Нули	0		0			0	0		0	0		0	0		0	0		0			
Единицы		1		1	1			1			1			1			1		1	1	1

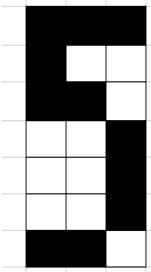
Проверка нулей $H(1-|x_1+x_3+x_6+x_7+x_9+x_{10}+x_{12}+x_{13}+x_{15}+x_{16}+x_{18}|)$

Проверка единиц $x_2x_4x_5x_8x_{11}x_{14}x_{17}x_{19}x_{20}x_{21}$

Полная проверка $x_2x_4x_5x_8x_{11}x_{14}x_{17}x_{19}x_{20}x_{21}\cdot H(1-|x_1+x_3+x_6+x_7+x_9+x_{10}+x_{12}+x_{13}+x_{15}+x_{16}+x_{18}|)$

Рисунок 8 - Распознавание цифры 1

Пример №2 Распознавание цифры 5 (рисунок 9).



	Входы																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	1	1	1	1	0	0	1	1	0	0	0	1	0	0	1	0	0	1	1	1	0
					0	0			0	0	0		0	0		0	0				0
	1	1	1	1			1	1				1			1			1	1	1	

Рисунок – 9 Распознавание цифры 5

Пример №3 Распознавание цифры 15 (рисунок 10).

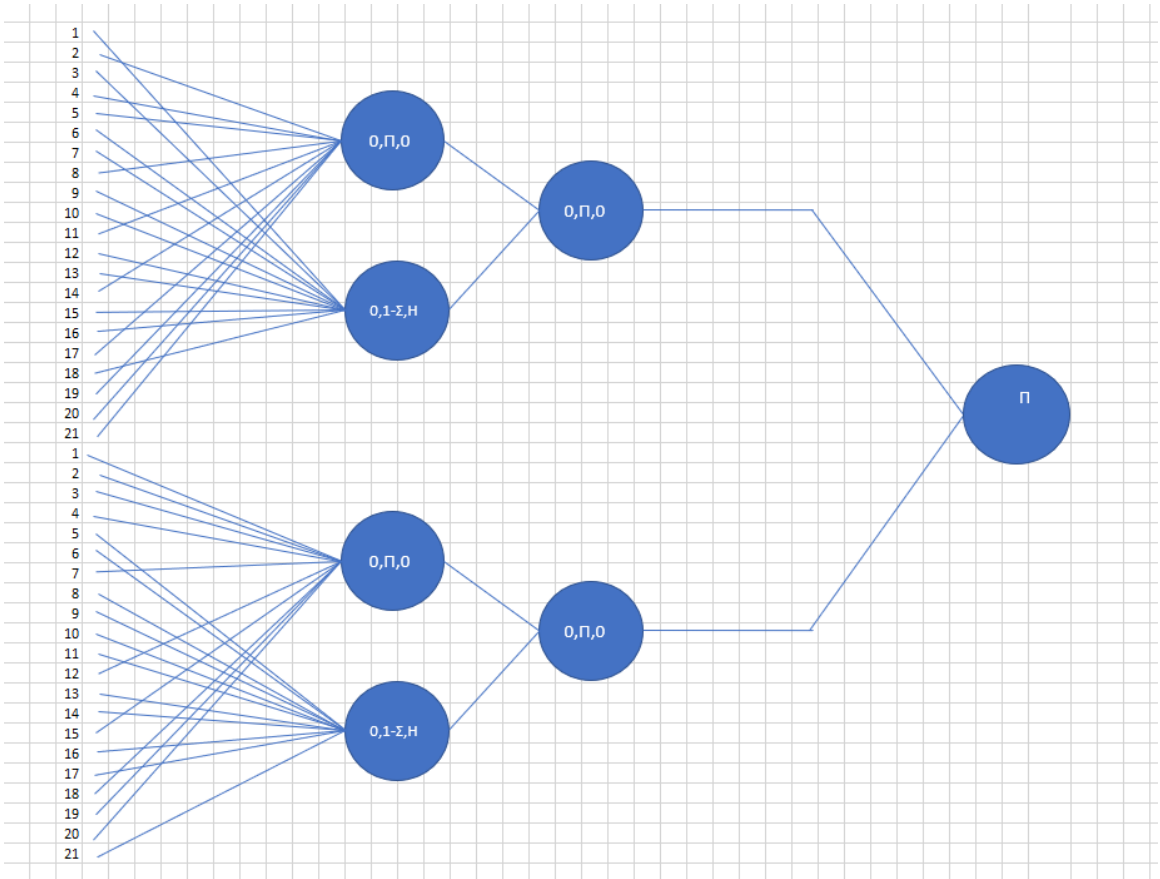


Рисунок 10 – Топология распознавания цифры 15

1.3 Методы обучения нейронной сети

Терминология

Образец - это набор данных, содержащий множество значений для всех входов входов сети (входную матрицу) и желаемое значение выхода сети (эталонный ответ).

Контрольно-обучающая выборка (набор) - совокупность данных содержащая все образцы применяемые для обучения и контроля нейронной сети.

Обучающая выборка - набор образцов применяемых в процессе обучения (оптимизации рецептивных коэффициентов) сети. Обычно составляет 50% от контрольно-обучающей.

Контрольная выборка - набор образцов применяемых при проверке качества обучения (кросс-валидации) нейронной сети. Обычно составляет 50% от контрольно-обучающей.

Цикл обучения (итерация) - процесс коррекции рецептивных коэффициентов нейронной сети по одному образцу. Обычно - один проход обучающего алгоритма по всей сети.

Обучающий пакет (batch) - набор образцов, после прохода которого изменяется какой-либо параметр обучения. Обычно - норма η или функция потерь (метод оценки ошибки сети).

Эпоха обучения - процесс коррекции рецептивных коэффициентов нейронной сети по всему множеству образцов обучающей выборки - по одному циклу обучения на каждый образец.

Обучение нейронной сети - это итеративный (повторяющийся) процесс поиска оптимальных рецептивных коэффициентов общей функции сети, путём многократного применения алгоритма оптимизации.

Классификация методов обучения нейронных сетей (рисунок 11-12).

Рисунок 11 – Классификация по алгоритмическому принципу

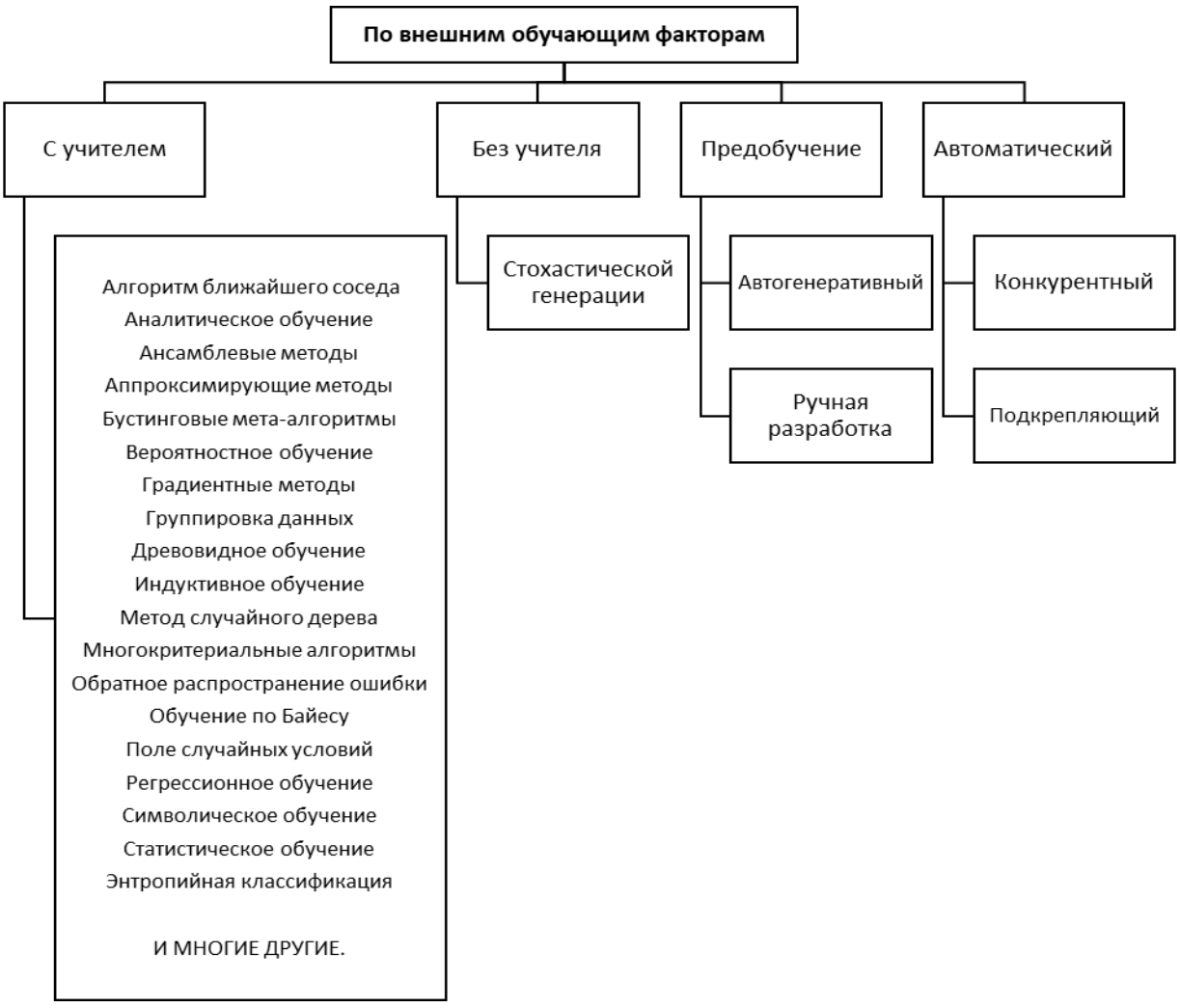


Рисунок 12 – Классификация по внешним обучающим факторам

Стохастические методы обучающей генерации основаны на случайном подборе коэффициентов сети. Метод стохастических множеств рассматривает множества готовых сетей со случайными коэффициентами.

Метод множественной выборки предполагает генерацию большого количества случайных сетей с последующей их оценкой и выбором единственной - лучшей по качеству.

Метод стохастического ансамбля предполагает объединение результатов большого количества случайных сетей посредством обобщающей сети (оркестратора) учитывающей их качество и объединяющей их результаты в соответствии с весами, зависящими от качества этих случайных сетей.

Методы стохастического обучения предполагают изменение коэффициентов готовой сети с целью снижения выходной ошибки до приемлемого уровня.

Метод стохастического обратного смещения заключается в последовательных попытках (от выхода ко входу) изменения коэффициентов сети на случайную величину с немедленной проверкой результата сети. Если ошибка понижается - изменение принимают, иначе - возвращают прежний коэффициент на место и переходят к следующему.

Метод общего стохастического обучения заключается в одновременных попытках изменения всех коэффициентов сети на случайную величину с немедленной проверкой результата сети. Если ошибка понижается - изменение принимают, иначе - возвращают прежние коэффициенты и переходят к следующей попытке.

Градиентные методы обучения заключаются в применении алгоритмов, позволяющих результату сети как бы последовательно двигаться в сторону уменьшения градиента ошибки.

Дельта-правило предполагает пропорциональное изменение коэффициентов линейных нейронов от выхода сети ко входам в сторону уменьшения ошибки сети.

Метод обратного распространения ошибки аналогичен дельта-правилу, однако применим к нелинейным нейронам, необходимое изменение коэффициентов вычисляется для каждого нейрона через производную функции индуцированного поля.

Эластический метод заключается в применении отдельной нормы обучения для каждого нейрона и оценке знака ошибки на выходе нейрона, если знак ошибки не изменяется, коэффициенты уменьшают или увеличивают на норму обучения, после чего саму норму увеличивают вдвое, если знак ошибки изменился, то норму уменьшают вдвое, а направление изменения коэффициентов изменяют на противоположное. Этот метод аналогичен алгебраическому методу половинного приближения.

ГЛАВА 2 МЕТОДЫ ОПТИМИЗАЦИИ НЕЙРОННОЙ СЕТИ

2.1 Дельта-правило

Для упрощения дальнейшего понимания, будем в большинстве случаев рассматривать простейшую нейронную сеть (рисунок 13).

Архитектура блочно: (4,1,2,1,1) L

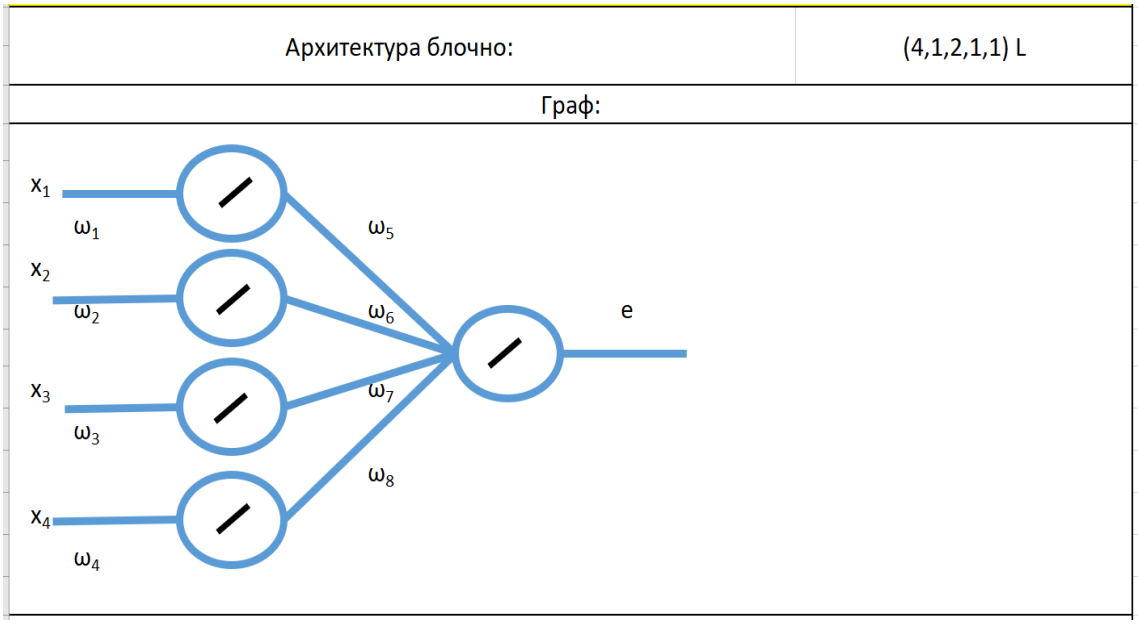


Рисунок 13 – Граф сети

Поскольку активаторы - линейны, то:

$$a = e\omega_1\omega_5; b = e\omega_2\omega_6; c = e\omega_3\omega_7; d = e\omega_4\omega_8; \quad (1)$$

Тогда общая формула сети:

$$f(x) = ax_1 + bx_2 + cx_3 + dx_4 \quad (2)$$

Обучением - называют процесс или алгоритм подбора коэффициентов сети, такой, чтобы сеть удовлетворяла заданным требованиям.

Учителем - называют человека, или систему определяющих заданные требования для обучения сети. (в том числе и меры соответствия этим требованиям).

Ошибкой сети или Функцией потерь - называют меру несоответствия результатов сети, заданным учителем требованиям.

Обратите внимание: Какой бы сложной ни была архитектура сети на линейных активаторах, в конечном итоге она сводится к общей сумме линейных уравнений с ограниченным числом коэффициентов, то есть по сути любая линейная нейронная сеть представляет из себя уравнение множественной (или многомерной, если значения входов относятся к различным мерам) регрессии (или копулу), но не более того.

Рассмотрим модель типовой трёхслойной нейронной сети на нейронах Уидроу топологии ВСЕ СО ВСЕМИ (рисунок 14). Правильными коэффициентами являются 0,1 для всех нейронов, кроме выходного, его коэффициенты 0,1; 0,2; 0,3.

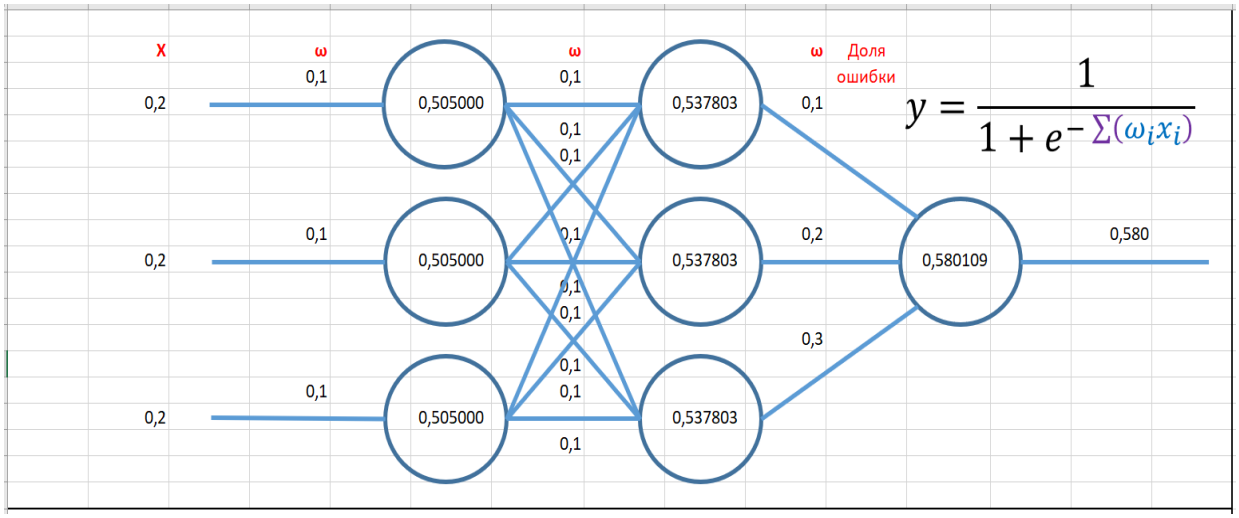


Рисунок 14 – Модель нейронной сети

Обратите внимание на следующие факты:

В топологии ВСЕ СО ВСЕМИ ошибка ОДНОГО нейрона распространяется на ВСЕ нейроны следующего слоя (рисунок 15). Величина привносимой в нейрон ошибки (доля ошибки) изменяется кратно коэффициентам рецептивного поля. Привнесённые ошибки агрегируются (в данном конкретном случае - суммируются).

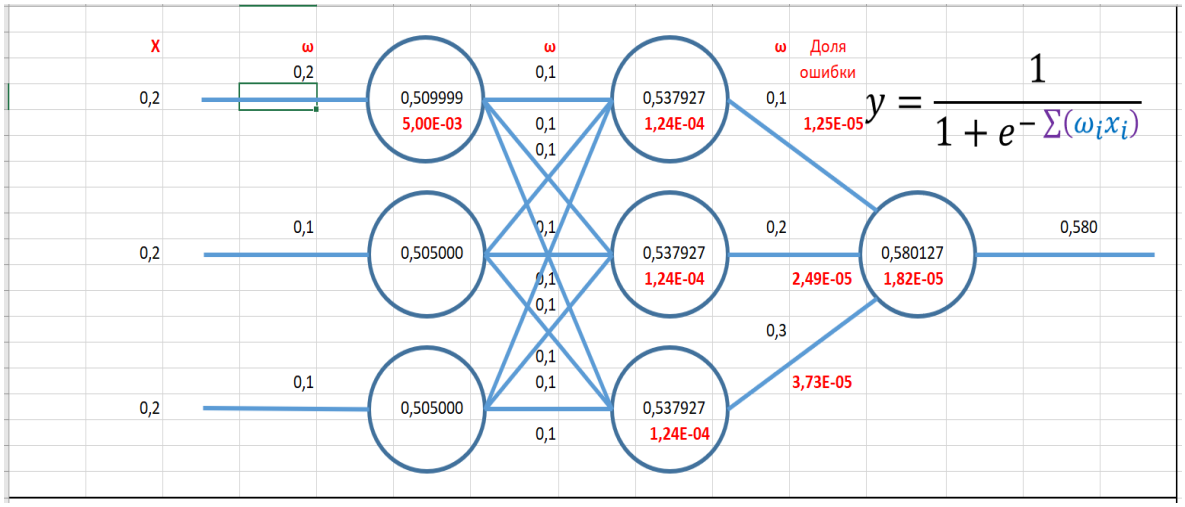


Рисунок 15 – Модель нейронной сети с привнесенной ошибкой

Дельта-правило выражает две основные идеи:

1. Чем большую ошибку привносит нейрон в сеть, тем сильнее нужно корректировать его коэффициенты.

2. Коэффициенты нейронов привносящих ошибку, нужно корректировать пропорционально их величине.

Дельта-правило разработано в докомпьютерную эпоху, поэтому не подходит для гладких функций с немонотонной производной второго порядка. Тем не менее оно прекрасно применимо для сетей типа L с линейными активаторами или линейно-пороговых - типа ReLu. Как правило, его применяют в свёрточных сетях там, где вычисление якобиана и гессиана займёт больше времени, чем дельта-оптимизация.

По сути Дельта-правило - это один из методов решения системы линейных уравнений со множественными аргументами.

Разберём Дельта-правило на примере одного нейрона с одним входом (рисунок 16).

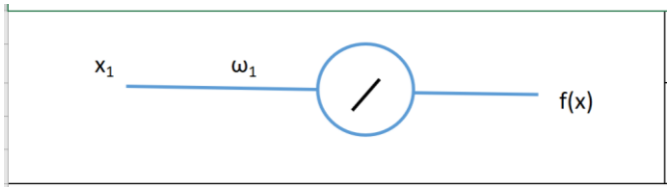


Рисунок 16 – Модель одного нейрона с одним входом

Формула имеет вид: $f(x) = \omega_1 x_1$ (3)

Например, текущее значение ω1 равно 0,5; образцом X1, является значение 0,3, а необходимым результатом обучения, для этого - значение f(x)=0,7.

1. Рассчитаем текущий результат сети: f(x) = 0,5 * 0,3 = 0,15

2. Сравним результат с необходимым значением: ε = 0,7 - 0,15 = 0,55

3. Если ошибка не удовлетворяет заданным требованиям (например нужно ε=0, а у нас ε=0,55 - большая ошибка) - вычисляем необходимую дельту для ω1.

Иначе - завершаем обучение.

$$\Delta\omega_1 = \frac{\varepsilon}{x_1}\eta \quad (4)$$

4. Вычитаем Δω1 из ω1, возвращаемся к пункту 1.

Разберём Дельта-правило на примере одного нейрона с двумя входами (рисунок 17).

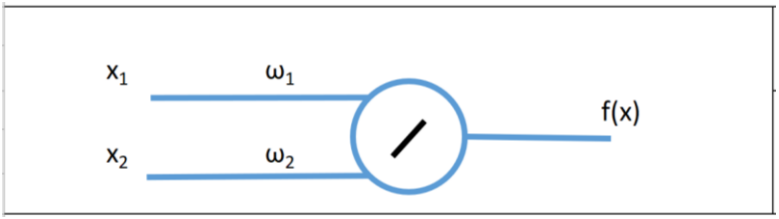


Рисунок 17 – Модель одного нейрона с двумя входами

Формула имеет вид: $f(x) = \omega_1 x_1 + \omega_2 x_2$ (5)

Например, текущие значения ω1 и ω2 равны 0,5 и 0,1; образцом X1 и X2, являются значения 0,3 и 0,9, а необходимым результатом обучения нейрона, для этого образца значение f(x) = 0,7.

1. Рассчитаем текущий результат сети: f(x) = 0,5 * 0,3 + 0,1 * 0,9 = 0,24

2. Сравним результат с необходимым значением: ε = 0,7 - 0,24 = 0,46

3. Если ошибка не удовлетворяет заданным требованиям, например, нужно ε=0, а у нас ε=0,46 (большая ошибка), то вычисляем необходимую дельту для ω1 и ω2.

Иначе завершаем обучение.

$$\Delta\omega_1 = \frac{\omega_1}{\omega_1 + \omega_2} \cdot \frac{\varepsilon}{x_1}\eta;\ \Delta\omega_2 = \frac{\omega_2}{\omega_1 + \omega_2} \cdot \frac{\varepsilon}{x_2}\eta \qquad (6)$$

4. Вычитаем Δω1 из ω1 и Δω2 из ω2, возвращаемся к пункту 1.

Рассмотрим модель типовой трёхслойной нейронной сети на линейных активаторах топологии ВСЕ СО ВСЕМИ (рисунок 18). Правильными коэффициентами являются 0,1 для всех нейронов, кроме выходного, его коэффициенты 0,1; 0,2; 0,3. Обратите внимание, что двигаясь по пути распространения ошибки в обратную сторону от выхода ко входу, можно рассчитать долю ошибки (дельту) предположительно привносимую на выход каждого нейрона, поскольку его входные коэффициенты известны. Изменяя коэффициенты пропорционально их удельным весам на определенную величину (норму обучения) в сторону уменьшения ошибки каждого нейрона можно снизить выходную ошибку сети. Многократное повторение такого процесса приводит сеть в состояние обученности, то есть снижает ошибку сети до приемлемого уровня. Этот метод обучения последовательным градиентным спуском называют Дельта-правилом.

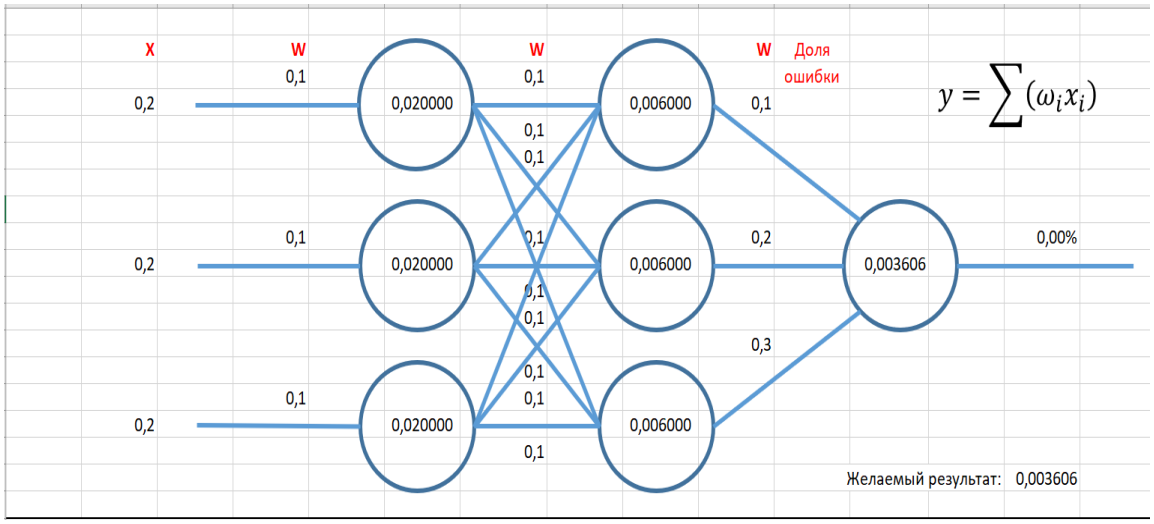


Рисунок 18 – Трехслойная модель нейронной сети

Поскольку дельта-правило является простейшим методом градиентного спуска, возникла необходимость введения такого понятия, как норма обучения (обозначается: η).

Например, в случае единичного отношения линейной dx по dy необходимое ненулевое dx приведёт к ситуации зацикливания, поскольку положительные и отрицательные смещения будут модульно тождественны и не приведут к нахождению оптимума.

Обратив внимание на данную проблему, Лаплас ещё в XVIII веке рекомендовал изменять шаг смещения коэффициентов гладкой функции, при поиске оптимума, деля его пополам, или умножая на два. Эта идея легла в основу эластического алгоритма обучения нейронных сетей. При применении дельта-метода, напротив, используют понятие нормы обучения (коэффициента, меньше единицы, но больше нуля, на который умножают необходимое смещение), обеспечивая тем самым сходимость для симметричных или линейных функций.

Таким образом, при обучении по дельта-правилу:

$$\Delta\omega_{\text{применяемое}} = \eta \cdot \Delta\omega_{\text{вычисленное}} \tag{7}$$

Рассмотрим пример использования дельта-правила для обучения сети (рисунок 19).

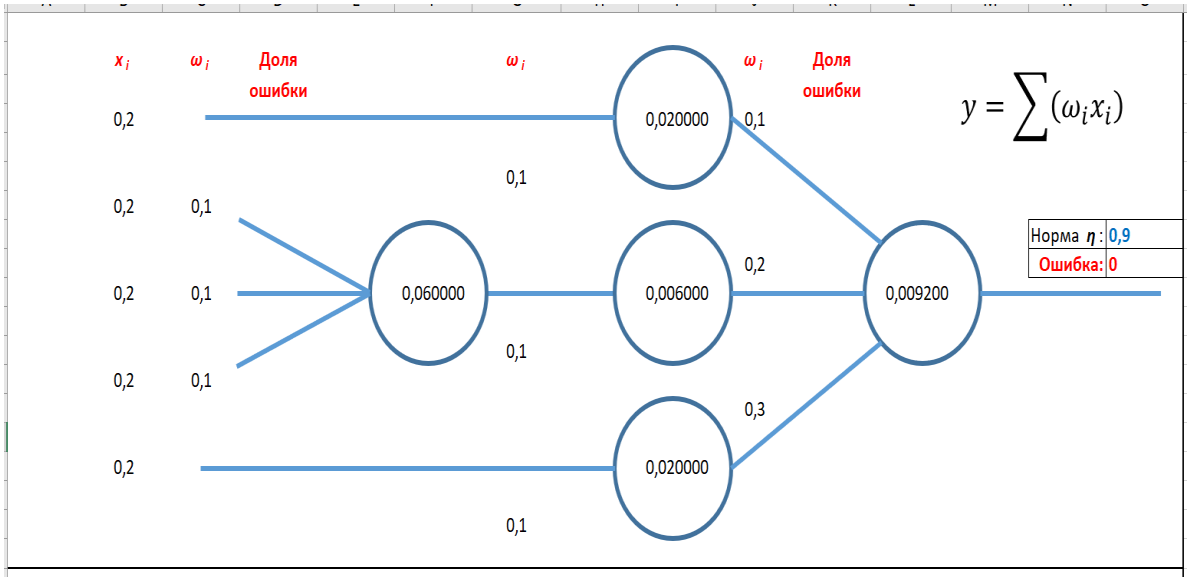


Рисунок 19 – Дельта-правило

В ячейку рядом с выходом сети введем значение ошибки и увидим как в обратном порядке, от выхода сети ко входам, рассчитываются исправленные значения для коэффициентов (рисунок 20).

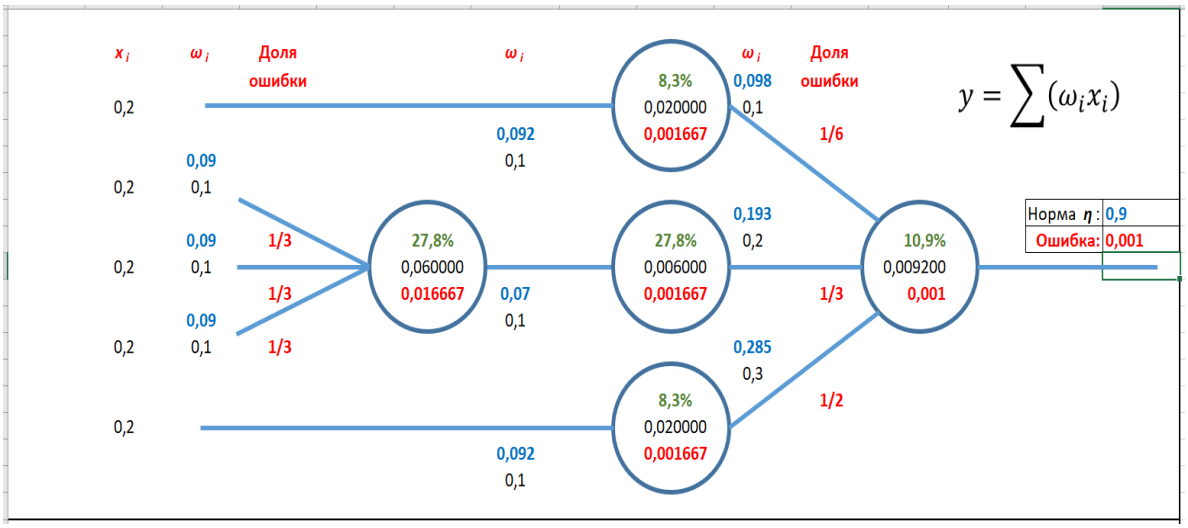


Рисунок 20 – Дельта-правило

Задание

В MS Excel, по заданной формуле постройте графическое изображение топологии нейронной сети и добавьте к нему формулы для расчета коэффициентов по дельта-правилу, аналогично вышеприведённому примеру (рисунок 21-22).

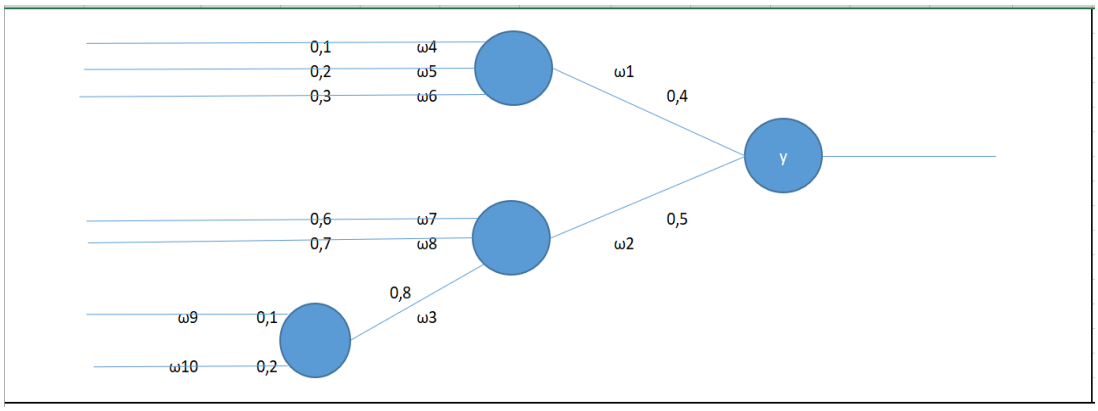
Вариант	Формула	Норма η
Вариант 1	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,99
Вариант 2	$y = 0{,}3(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,95
Вариант 3	$y = 0{,}2(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,9
Вариант 4	$y = 0{,}1(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,85
Вариант 5	$y = 0{,}5(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,8
Вариант 6	$y = 0{,}6(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,75
Вариант 7	$y = 0{,}7(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,7
Вариант 8	$y = 0{,}4(0{,}8x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,65
Вариант 9	$y = 0{,}4(0{,}1x_1 + 0{,}9x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,6
Вариант 10	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}8x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,99
Вариант 11	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}7(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,95
Вариант 12	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}6(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,9
Вариант 13	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}5x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,85
Вариант 14	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}4x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,8
Вариант 15	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}3(0{,}1x_9 + 0{,}2x_{10}))$	0,75
Вариант 16	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}2x_9 + 0{,}2x_{10}))$	0,7
Вариант 17	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}1x_{10}))$	0,65
Вариант 18	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}3x_9 + 0{,}2x_{10}))$	0,6
Вариант 19	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}4(0{,}1x_9 + 0{,}2x_{10}))$	0,7
Вариант 20	$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}5x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$	0,8

Рисунок 21 – Варианты заданий

Пример выполнения первого варианта (рисунок 22).

$$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5\big(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10})\big)$$

Норма $\eta = 0{,}99$



Вариант		Формула			Норма η
Вариант 1		$y = 0{,}4(0{,}1x_1 + 0{,}2x_2 + 0{,}3x_3) + 0{,}5(0{,}6x_6 + 0{,}7x_7 + 0{,}8(0{,}1x_9 + 0{,}2x_{10}))$			0,99
Счёт слоёв ведётся **с конца**.					
Формулы для 3 слоя		ε = 0,05		η= 0,99	
Δω1	-0,0002222222				
Δω2	-0,0002777778				
Формулы для 2 слоя					
Δω3	-0,0001058201				
Δω4	-0,0000370370				
Δω5	-0,0000740741				
Δω6	-0,0001111111				
Δω7	-0,0000793651				
Δω8	-0,0000925926				
Формулы для 1 слоя					
Δω9	-0,0000352734				
Δω10	-0,0000705467				

Рисунок 22 – Пример выполнения задания

2.2 Метод обратного распространения ошибки

Терминология

Функцию потерь (отклонение выхода от желаемого результата - эталона) будем называть ошибкой сети - ε.

Ситуацию, когда в результате цикла обучения, изменяется знак ошибки сети ($\varepsilon \rightarrow \alpha \cdot -\varepsilon$) будем называть промахом обучения.

Набор данных на входах нейронов первого слоя и эталон значения выхода будем называть образцом.

Набор входных коэффициентов всех нейронов сети будем называть весами сети.

Набор входных коэффициентов одного слоя будем называть тензорным набором или просто тензором (хоть это и неправильно).

Целью обучения нейронной сети является подбор весов, такой, при котором суммарная ошибка сети для всего набора предъявляемых образцов будет минимальна.

При этом следует учитывать, что сама ошибка (функция потерь) может быть любой, например квадратичной, кубической или геометрической.

Обучение сети - это многократный процесс предъявления образцов (установка значений входов) и коррекции весов в соответствии с нормой обучения, до достижения желаемого (или теоретически достижимого) оптимума, удовлетворяющего требованиям каждого образца, в достаточной (или максимально возможной) степени.

Один этап предъявления образца и коррекции весов называют циклом обучения.

Дельта-проблема

Изначально, Дельта-правило предполагает применение исключительно в линейных сетях. Имитация даже простейшей естественной реакции нейрона требует применения нелинейных функций, на которых Дельта-правило неприменимо. С целью решения данной проблемы был разработан метод обучения нейронных сетей, получивший название Обратного Распространения

Ошибки, основным требованием которого являются лишь гладкость, дифференцируемость и непрерывность активационной функции нейрона. Благодаря данному методу в нейронных сетях стало возможно использовать множество функций, в том числе и функцию Ферхюльста, нейрон с которой и является, на сегодняшний день, самым распространённым и носит название нейрона Уидроу.

Рассмотрим такой нейрон, активатором которого является непрерывная функция, дифференцируемая на всём промежутке возможных значений индуцированного поля. Заметьте, что речь здесь идёт не о промежутке [-∞;+∞], а как правило, о довольно небольшом, вполне конкретном для каждого вида нейронов промежутке, зависящем от границ возможных значений входов, границ коэффициентов и типа агрегатора.

Границы индуцированного поля определяются, как агрегированные значения минимума и максимума рецептивного поля. Границы минимума и максимума рецептивного поля соответствуют минимальному и максимальному значениям обучающей выборки, умноженным (или иное) на соответствующие значения нижней и верхней границ входных коэффициентов.

$$\upsilon_{\min} = f_1(f_2(\omega_{\min}, x_{\min}),\ n) \tag{8}$$

$$\upsilon_{\max} = f_1(f_2(\omega_{\max}, x_{\max}),\ n) \tag{9}$$

$\upsilon_{\min}$ – левая граница индуцированного поля
$\upsilon_{\max}$ – правая граница индуцированного поля
f_1 – функция агрегатора
f_2 – функция рецептора
n – количество входов
ω, x – коэффициенты и значения входов

Например, для нейрона Уидроу:

$$\upsilon_{\min} = n \cdot (\omega_{\min}, x_{\min}) \tag{10}$$

$$\upsilon_{\max} = n \cdot (\omega_{\max}, x_{\max}) \tag{11}$$

$\upsilon_{\min}$ – левая граница индуцированного поля
$\upsilon_{\max}$ – правая граница индуцированного поля
n – количество входов
ω, x – коэффициенты и значения входов

Для примера рассчитаем границы индуцированного поля нейрона Уидроу с десятью нормированными входами и входными коэффициентами [-1;1].

$$v_{\min} = 10 \cdot (-1 \cdot 0) = 0;\ v_{\max} = 10 \cdot (1 \cdot 1) = 10;\ \ v \in [0; 10] \qquad (12)$$

Для примера рассчитаем границы индуцированного поля нейрона с умножающим агрегатором, десятью тысячами нормированных входов и входными коэффициентами [-1;1].

$$v_{\min} = (-1 \cdot 0)^{10000} = 0;\ v_{\max} = (1 \cdot 1)^{10000} = 1;\ \ v \in [0; 1] \qquad (13)$$

Далее будем считать, что на определённом интервале дифференцируема любая неразрывная функция, существующая на этом интервале. Впрочем, следует отметить, что метод ОРО неприменим для таких функций, имеющих интервалы (но не точки) с нулевой или бесконечной производной.

Для примера (рисунок 23) рассмотрим сигма-функцию на интервале [-6;6].

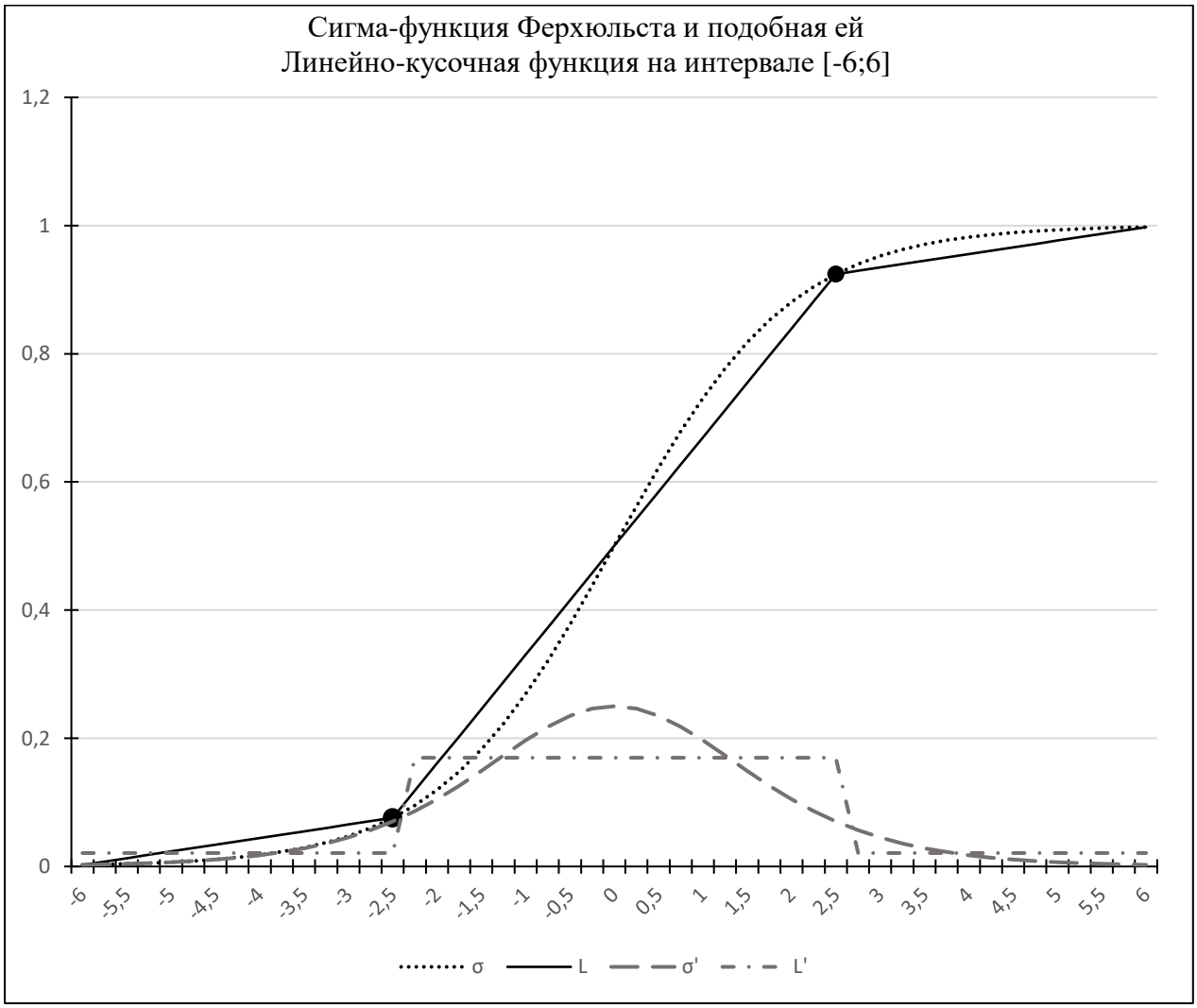


Рисунок 23 – Сигма функция

Обратите внимание, что и сигма-функция, и подобная ей линейно-кусочная - обе дифференцируемы на данном интервале.

Идея алгоритма обучения с обратным распространением ошибки состоит в том, что любую дифференцируемую функцию можно свести к линейно-кусочной с очень маленькими интервалами, при этом дифференциал аргумента относительно функции будет описываться простым линейным уравнением, что позволит применить дельта-правило.

Для вычисления значений индуцированного поля нейронов Уидроу по значению выхода, применяют обратную логистическую функцию (рисунок 24). Обратите внимание, данная функция определена на промежутке от нуля до единицы не включающем граничные значения.

$$\upsilon = -\ln\left(\frac{1}{\sigma} - 1\right);\ \sigma \in]0;1[\tag{14}$$

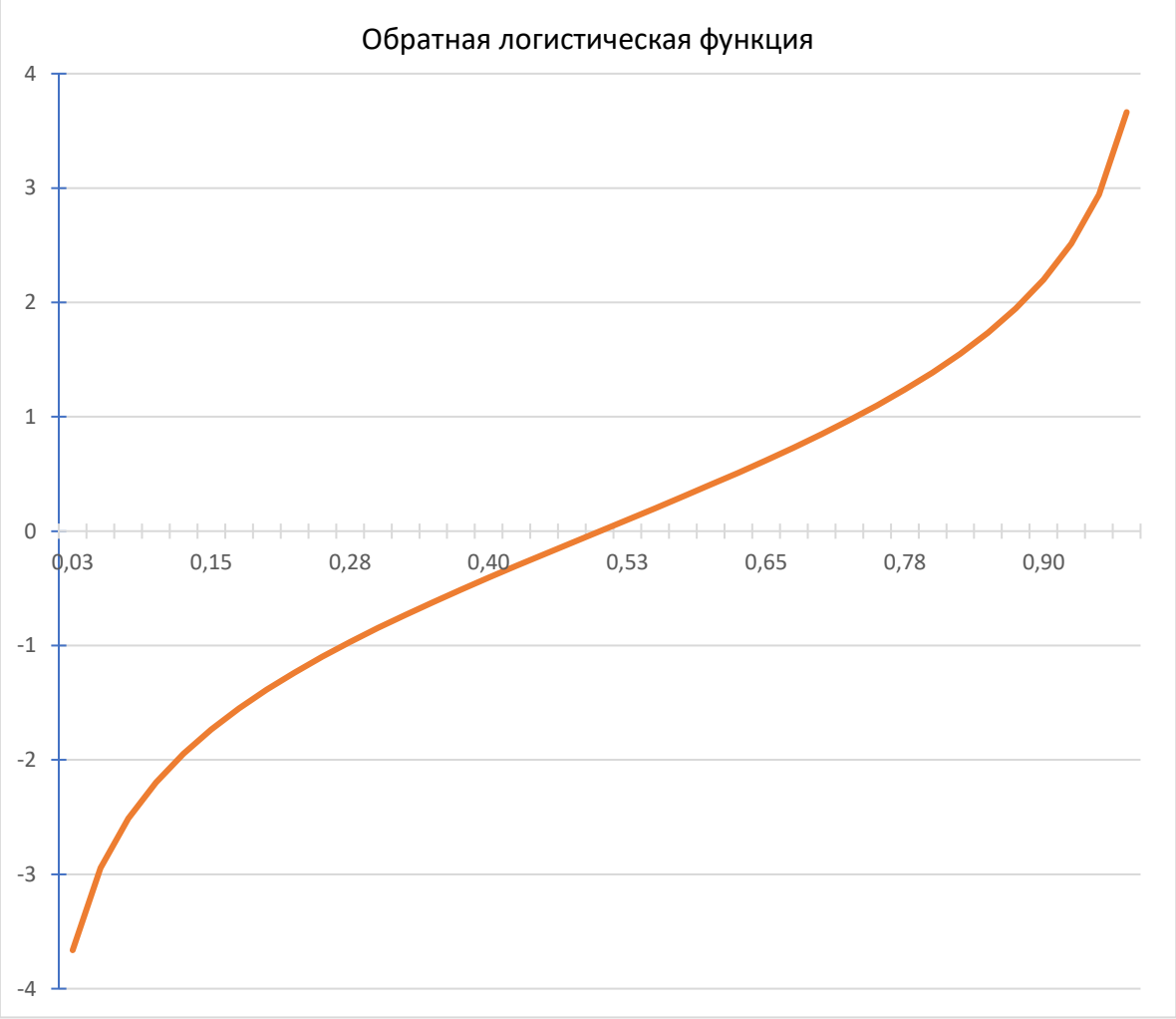


Рисунок 24 – Обратно логистическая функция

Дельта-правило пригодно для тех случаев, когда активатор нейрона линеен и функции агрегатора и рецептора - обратимы, в этом случае, зная значение индуцированного поля υ мы можем пропорционально изменить рецептивные коэффициенты ω_i на норму обучения η в нужную сторону, чтобы уменьшить ошибку выхода сети.

Рассмотрим принцип ОРО на примере фрагмента сигма-функции выходного нейрона (рисунок 25).

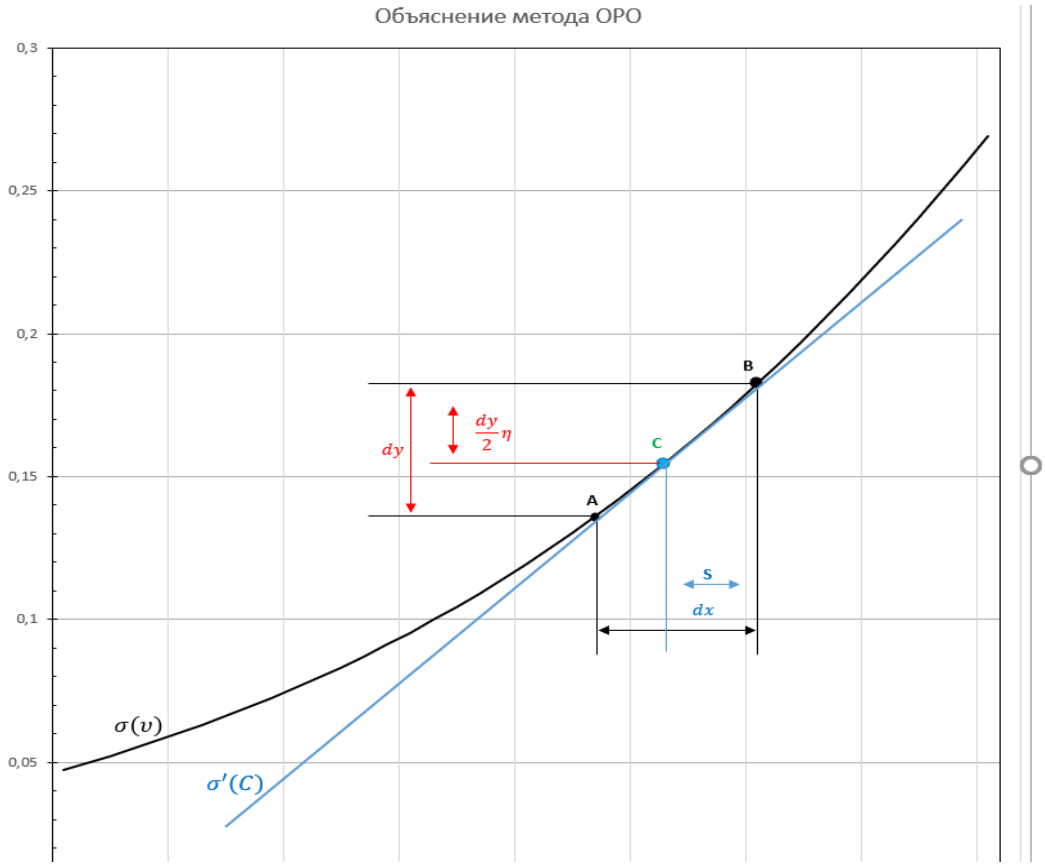


Рисунок 25 – Метод ОРО

Здесь: ось *OX* соответствует значению индуцированного поля *υ*, ось *OY* значению сигма-функции выхода нейрона, сама функция отражена на графике, как *σ*, точка *A* соответствует искомому (безошибочному) значению выхода сети, точка *B* - текущему (ошибочному) значению выхода. Тогда величина разницы значений *σB-σA* обозначенная *dy* будет соответствовать выходной ошибке нейрона, а величина разницы *υB-υA* - необходимой для уменьшения ошибки величине сдвига индуцированного поля υ, обозначаемой *dx*. Проблема заключается в том, что сигма-функция не линейна, а отношение *dy/dx*

изменяется в диапазоне (0;0,25) и даже зная значение ошибки *dy*, невозможно точно вычислить необходимое значение *dx*.

Идея метода ОРО заключается в том, что производная в серединной точке С, соответствующей *σA+dy/2* имеет значение, достаточно близкое к *dy/dx* необходимому для вычисления требуемого смещения *S* по индуцированному полю *υ*. В некоторых случаях *dy/2* сразу умножают на норму обучения *η*, чтобы избежать "промаха" значения выхода мимо точки верного решения, а в некоторых, норму η применяют на уже вычисленное значение смещения рецептивного поля *S*.

Разберём на примере:

$$\sigma B = 0{,}7; \quad \sigma A = 0{,}5; \quad \eta = 0{,}9; \quad \sigma \equiv y; \quad \upsilon \equiv x \tag{15}$$

$$\varepsilon = dy = 0{,}7 - 0{,}5 = 0{,}2$$

$$yC = yA + \frac{dy}{2} = 0{,}6$$

$$x(y) = -\ln\left(\frac{1}{y} - 1\right) = -\ln\left(\frac{1}{0{,}6} - 1\right) \approx 0{,}405$$

$$C'(x) = y(x) \cdot \big(1 - y(x)\big) = y(0{,}405) \cdot \big(1 - y(0{,}405)\big) \approx 0{,}24$$

$$\Delta\omega = dx \approx dy \cdot C'(x) \approx 0{,}2 \cdot 0{,}24 \approx 0{,}048$$

$$S = dx \cdot \eta = 0{,}048 \cdot 0{,}9 = 0{,}0432$$

Далее, зная значение *S*, пропорционально изменяем входные коэффициенты ω_i, используя дельта-правило.

Обратите внимание, что в данном методе присутствуют два "вычислительно тяжёлых" элемента: вычисление половинного интервала *dy* и вычисление обратной сигма-функции. При большей степени огрубления вычислений можно отказаться от них, принимая за отношение *dy/dx* значение производной сигма-функции в точке ошибки (точка *B*).

Для сигма-функции или гиперболического тангенса вычисление значения *x(y)* - не обязательно, поскольку производная в точке *xC* может быть рассчитана через значение *y(x)*, а оно уже известно. Однако для таких активаторов, как *SoftSign, ISRU, arctg* и других, без вычисления *x(y)* не обойтись.

Ускоренный метод обратного распространения ошибки

Ускоренный метод обратного распространения ошибки на сегодняшний день используется при обучении нейронных сетей с активатором сигма-функцией или гиперболическим тангенсом, поскольку менее затратен (рисунок 26).

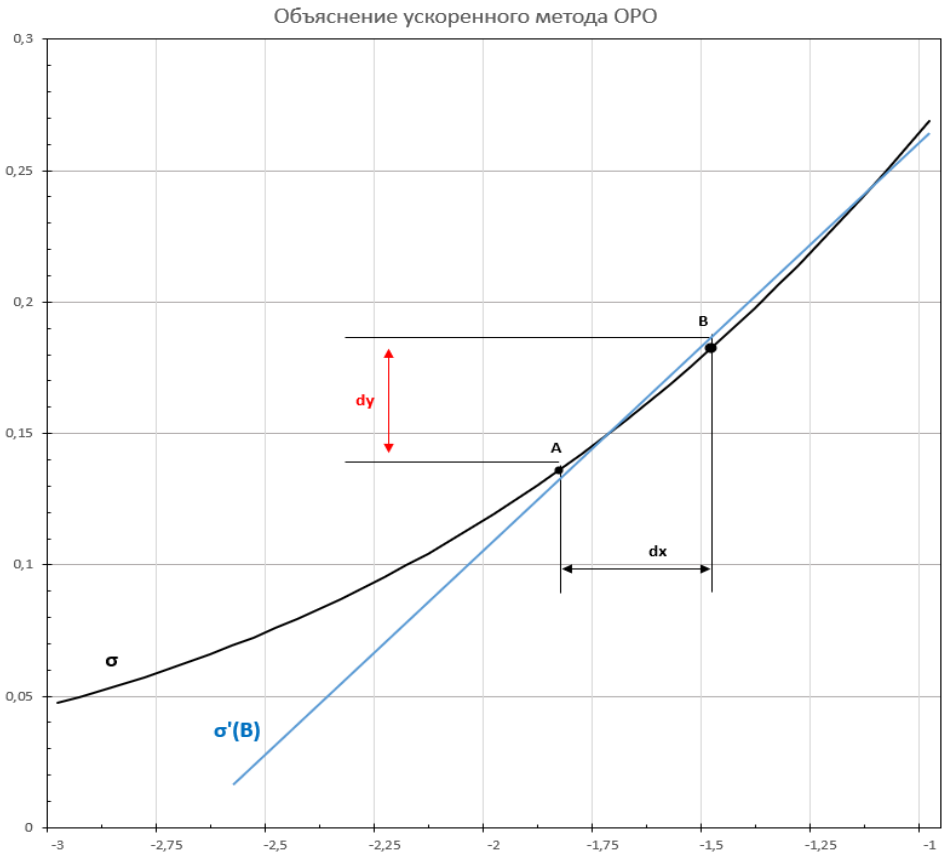


Рисунок 26 – Ускоренный метод обратного распространения ошибки

$$\sigma B = 0{,}7;\ \sigma A = 0{,}5; \upsilon = 0{,}85;\ \eta = 0{,}4 \tag{16}$$

$$\sigma \equiv y;\ \upsilon \equiv x$$

$$dy = 0{,}7 - 0{,}5 = 0{,}2$$

$$xB - \text{нам известно, это значение индуцированного поля } \boldsymbol{\upsilon}$$

$$C'(xB) = y(xB) \cdot \big(1 - y(xB)\big) = y(0{,}85) \cdot \big(1 - y(0{,}85)\big) \approx 0{,}13$$

$$\varepsilon = dx \approx dy \cdot C'(x) \approx 0{,}2 \cdot 0{,}13 \approx 0{,}026$$

$$\Delta\omega = S = dx \cdot \eta = 0{,}026 \cdot 0{,}4 = 0{,}0104$$

Далее, зная значение S пропорционально изменяем входные коэффициенты ω_i используя дельта-правило.

Обратите внимание: норму обучения при этом уменьшают вдвое, чтобы не было "промаха". Кажется, что такая сеть будет обучаться дольше, чем предыдущая, однако, выигрыш от исключения двух операций деления и одной операции взятия логарифма оказывается больше, чем эффект от снижения нормы обучения. Фактически, обучение по ускоренному методу обратного распространения ошибки примерно в четыре раза быстрее классического метода.

Задание

1. Для сети заданной топологии обучаемой по эластическому алгоритму рассчитайте в байтах необходимый объём оперативной памяти, если на хранение одного входного значения x уходит 2 байта, на коэффициент ω каждого входа и норму обучения η каждого нейрона уходит по 4 байта, на значение индуцированного поля υ и значение выхода нейрона y - по 8 байт.

В эластическом алгоритме коэффициентов нормы обучения η столько же, сколько и нейронов. В методе обратного распространения ошибки один общий коэффициент η используется для всей сети.

Рассчитайте объём необходимый для такой же сети при использовании метода обратного распространения ошибки. Оцените выигрыш объёма ОЗУ в процентах.

Таблица 1 – Варианты заданий

	Топология
Вариант 1	(24,24,2,12,12); (6,6,2,1,1)
Вариант 2	(22,22,2,12,12); (6,6,2,1,1)
Вариант 3	(70,70,2,35,35); (18,18,2,1,1)
Вариант 4	(86,86,2,44,44); (22,22,2,1,1)
Вариант 5	(68,68,2,34,34); (17,17,2,1,1)
Вариант 6	(44,44,2,22,22); (11,11,2,1,1)
Вариант 7	(38,38,2,20,20); (10,10,2,1,1)
Вариант 8	(30,30,2,15,15); (8,8,2,1,1)
Вариант 9	(14,14,2,10,10); (5,5,2,1,1)
Вариант 10	(54,54,2,26,26); (13,13,2,1,1)
Вариант 11	(50,50,2,25,25); (13,13,2,1,1)
Вариант 12	(34,34,2,16,16); (8,8,2,1,1)
Вариант 13	(86,86,2,44,44); (22,22,2,1,1)
Вариант 14	(44,44,2,22,22); (11,11,2,1,1)

Вариант 15	(28,28,2,14,14); (7,7,2,1,1)
Вариант 16	(96,96,2,48,48); (24,24,2,1,1)
Вариант 17	(38,38,2,20,20); (10,10,2,1,1)
Вариант 18	(98,98,2,50,50); (25,25,2,1,1)
Вариант 19	(60,60,2,30,30); (15,15,2,1,1)
Вариант 20	(20,20,2,10,10); (5,5,2,1,1)

Пример решения (25,125,2,12,12); (6,4,2,1,1)

Таблица 2 – Пример решения

1 слой					
3125	Нейронов	Байт	Памяти		Нет в ОРО
3125	X	2	6250		
3125	ω	4	12500		
3125	υ	8	25000		
3125	η	4	12500		12500
3125	Y	8	25000		
2 слой					
144	Нейронов				
450000	ω	4	1800000		
144	υ	8	1152		
144	η	4	576		576
144	Y	8	1152		
3 слой					
24	Нейронов				
3456	ω	4	13824		
24	υ	8	192		
24	η	4	96		96
24	Y	8	192		
4 слой					
1	Нейронов				
24	ω	4	96		
1	υ	8	8		
1	η	4	4		4
1	Y	8	8		
		Эласт:	1 898 550	ОРО:	1 885 375
		Выигрыш ОЗУ		**0,7%**	

2. Нейрон с двумя входами обучаемый по классическому (не ускоренному) методу обратного распространения ошибки имеет определённую ошибку на выходе. Рассчитайте новые значения входных коэффициентов нейрона до третьего знака после запятой.

Таблица 3 – Варианты решения

	Коэффициенты		Индуцированное поле	Ошибка	Выход нейрона	Норма обучения	Активатор
Вариант	**ω_1**	**ω_2**	**υ**	**ε**	**y**	**η**	
Вариант 1	0,140	0,840	0,640	0,390	0,655	0,900	Сигма-функция
Вариант 2	0,580	0,740	0,640	0,325	0,655	0,900	Гиперболический тангенс
Вариант 3	0,430	0,860	0,640	-0,300	0,655	0,800	Сигма-функция
Вариант 4	0,720	0,270	0,380	-0,005	0,594	0,800	Гиперболический тангенс
Вариант 5	0,600	0,260	0,650	0,310	0,657	0,700	Сигма-функция
Вариант 6	0,480	0,970	0,190	0,040	0,547	0,700	Гиперболический тангенс
Вариант 7	0,810	0,020	0,220	-0,370	0,555	0,600	Сигма-функция
Вариант 8	0,720	0,660	0,850	-0,475	0,701	0,600	Гиперболический тангенс
Вариант 9	0,090	0,070	0,270	0,030	0,567	0,500	Сигма-функция
Вариант 10	0,040	0,730	0,730	0,060	0,675	0,500	Гиперболический тангенс
Вариант 11	0,900	0,190	0,960	-0,490	0,723	0,900	Сигма-функция
Вариант 12	0,580	0,890	0,010	-0,425	0,502	0,900	Гиперболический тангенс
Вариант 13	0,930	0,460	0,810	0,020	0,692	0,800	Сигма-функция
Вариант 14	0,520	0,000	0,060	0,055	0,515	0,800	Гиперболический тангенс
Вариант 15	0,990	0,160	0,090	-0,330	0,522	0,700	Сигма-функция
Вариант 16	0,530	0,170	0,210	-0,255	0,552	0,700	Гиперболический тангенс
Вариант 17	0,590	0,540	0,080	0,325	0,520	0,600	Сигма-функция
Вариант 18	0,150	0,120	0,920	0,225	0,715	0,600	Гиперболический тангенс
Вариант 19	0,550	0,520	0,960	-0,020	0,723	0,500	Сигма-функция
Вариант 20	0,330	0,430	0,040	-0,370	0,510	0,500	Гиперболический тангенс

Пример варианта 1 и 2 (рисунок 27).

Пример Варианта 1					
ω	Доля ошибки	C`(υ)	Δω	ω_{N1}	ω_{N2}
0,140	0,143	0,226	0,013	0,127	0,764
0,840	0,857		0,076		

Пример Варианта 2					
ω	Доля ошибки	C`(υ)	Δω	ω_{N1}	ω_{N2}
0,580	0,439	0,681	0,097	0,483	0,616
0,740	0,561		0,124		

Рисунок 27 – Пример решения

2.3 Метод Ньютона

Оптимизация – метод нахождения неизвестного значения аргумента, соответствующего заданному значению функции (оптимуму). Методов оптимизации – много, самые известные из них – это метод последовательного перебора, метод половинного приближения и метод Ньютона. Первые два метода подходят для любых функций, но являются очень медленными.

Метод Ньютона – быстрый, но подходит только для дважды дифференцируемых функций (имеющих две производные). Важным ограничением также является ненулевое значение второй производной.

Идея метода Ньютона:

1. Чем быстрее растёт функция в точке, тем быстрее растёт её частная производная.
2. Чем быстрее растёт производная функции, тем больше значение второй производной.

Чтобы при приближении к оптимуму иметь заданную скорость сходимости, нужно использовать не только линейную характеристику крутизны функции, выражаемую первой производной, но и учитывать изменение этой характеристики, выражаемое второй производной.

При известном значении аргумента в заданной точке, для вычисления необходимого смещения к точке оптимума, в методе Ньютона используется отношение первой и второй производных, что позволяет снизить или увеличить шаг приближения не перескакивая через оптимум, тем самым уменьшая количество итераций поиска оптимального значения аргумента.

На рисунке 28 показан итерационный путь трёх различных алгоритмов оптимизации сложных функций (последовательного перебора, половинного приближения и метода Ньютона), как мы видим – последний из них является самым быстрым.

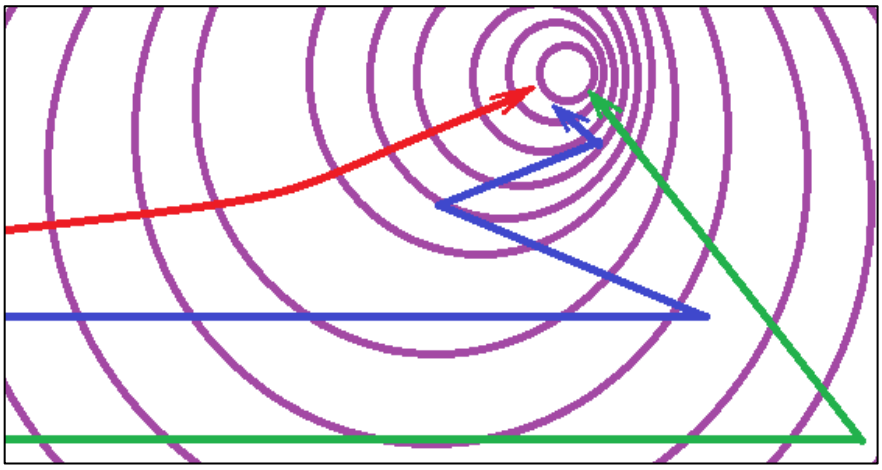


Рисунок – 28 Сравнение методов оптимизации

Опишем метод Ньютона:

Пусть у нас есть какая-либо сложная функция $f(x)$, имеющая известное нам оптимальное значение $f(x)_o$, при этом нам известно значение функции в некоторой текущей точке x_c, равное $f(x)_c$, тогда

$$f(x)_o = f(x)_c \pm \Delta f(x_o \pm x_c) = f(x_c \pm \Delta x) \tag{17}$$

$$f(x_c \pm \Delta x) \approx f(x_c) \pm f'(x_c) \cdot \Delta x \pm \frac{1}{2} f''(x_c) \cdot \Delta x^2 \Rightarrow \Delta x \approx \pm f'(x_c) \cdot \left(f''(x_c)\right)^{-1} \tag{18}$$

Значит, при каждой итерации приближения к оптимуму x_o, значение x_c следует изменять в сторону оптимума на вычисленное Δx. При этом вектор направления вычисляется, как:

$$v = \operatorname{sgn}\left(-f'(x_c) \cdot \left(f''(x_c)\right)^{-1}\right) \tag{19}$$

Применение метода в нейронных сетях.

Алгоритм массированного обратного распространения ошибки (не путать с классическим ОРО) был предложен для обучения глубоких нейронных сетей Йошуа Бенжио, Джеффри Хинтоном и Яном Лекуном в 2007 году.

Основная идея алгоритма заключается в представлении оптимизирующих вычислений нейронной сети в виде, легко поддающемся распараллеливанию за счёт использования матричного представления для хранения и вычисления

параметров сети. Корни идеи лежат в применении Ньютоновского метода для оптимизации множества переменных по обобщённому градиенту функции.

Для лучшего понимания рассмотрим метод ОПГ на примере задачи:

Пусть имеется некоторое трёхмерное пространство с координатами φ, x, y, при этом величина φ, именуемая также **полем** φ, является функцией от переменных x, y.

Пусть $f(x, y) = \varphi$, имеет известный оптимум значения $f(x_o, y_o)$. Для функции также известно значение в текущей точке φ_c с аргументами (x_c, y_c).

Необходимо путём итерационных вычислений найти аргументы φ_o, соответствующие координатам (x_o, y_o).

В данном случае можно рассматривать условный многомерный пространственный вектор, именуемый также **тензором** из точки $f(x_c, y_c)$ в точку $f(x_o, y_o)$, как геометрическую сумму двумерных векторов: $\vec{a}$ из $f(x_c)$ в $f(x_o)$ и $\vec{b}$ из $f(y_c)$ в $f(y_o)$.

Пространственную совокупность этих векторов, формирующую тензор, обращённый в сторону оптимального (или иногда максимального) значения $f(x, y) = \varphi$, и характеризующую совокупную линейную скорость $\Delta\varphi$ по $\Delta x,$ и Δy называют градиентом и обозначают символом **набла: ∇**, в данном случае: $\nabla\varphi$ или $\nabla f(x, y)$. Градиент записывают в векторном виде: $\nabla\varphi = (5; 3)$.

Рассматривая сечения φ соответствующими плоскостями X и Y, имеем набор двумерных функций $f(x)$ и $f(y)$, тогда направления $\vec{a}$ и $\vec{b}$ соответствуют $f'(x)$ и $f'(y)$. Иначе говоря, $\nabla\varphi$ является геометрической суммой производных её аргументов.

Если переходить от общего понятия градиента к частному, то можно сказать, что частный градиент функции нескольких переменных φ в заданной точке является геометрической суммой частных производных функций φ_n, являющихся проекциями родительской функции на соответствующие плоскости (или гиперплоскости, если мерность пространства больше трёх).

В том случае, если φ дифференцируется до второго порядка, для нахождения аргументов оптимума можно применить метод Ньютона, последовательно вычисляя необходимые смещения тензора по градиенту, определяемые, как отношение первой и второй частных производных проекций родительской функции на соответствующие плоскости.

ГЛАВА 3 Практикум в Excel

3.1 Аналитические нейронные сети

По сути любая простая нейронная сеть (да и сложная тоже), математически представляет из себя суперпозицию регрессионных функций, описывающих связанность (корреляцию) между значениями входов и выходов сети.

Нейронные сети можно использовать для анализа корреляций, раскладывая зависимости на элементарные регрессионные уравнения. Такие сети называют аналитическими.

Так, например, сеть с линейными нейронами может быть приведена к конечному виду уравнения множественной линейной регрессии, а сеть с пороговыми активаторами к уравнению машинной логики вида ЕСЛИ-ТО.

Однако для анализа влияния факторов сложного процесса на конечный результат, обычно используют сигма-подобные функции (функцию Ферхюльста, гиперболический тангенс, арктангенс, SoftSign, ISRU и т.д.)

Это связано с тем, что нелинейные функции гораздо лучше помогают описать реальные процессы, по сравнению с линейными или пороговыми.

Например, уравнение Ферхюльста приближённо описывает не менее трёх вариантов традиционно встречающихся в природе зависимостей.

- Экспоненциальной - на промежутке [-6; -1];
- Логарифмической - на промежутке [1; -6];
- Линейной - на промежутке [-1; 1].

Границы промежутков соответствия не могут быть определены точно, и определяются исключительно личными предпочтениями аналитика.

Следует заметить, что любая интерпретация весов аналитической сети неоднозначна. Она прежде всего зависит от личных предпочтений аналитика. Так, например, меньший коэффициент влияния, находящийся в экспоненциальной области, может быть расценен, как более существенный, по сравнению с высоким, находящимся в логарифмической.

Обычно для оценки важности коэффициентов рецептивного поля, используют их близость к центру сигма-функции, которая выражается величиной производной первого порядка.

Отношение же коэффициентов входов единственного нейрона оценивают по их простой пропорции.

Обычно, при анализе используют следующие весовые коэффициенты значимости входов:

Для одного нейрона:

$$P\omega_n = \frac{\omega_n}{\sum_{i=1}^{i=n+m} \omega_i} \tag{20}$$

Пример для нескольких нейронов Уидроу:

$$P\omega_n = \frac{\omega_n \cdot |6 - v_i|}{\sum_{i=1}^{i=n+m} \omega_i \cdot |6 - v_i|} \tag{21}$$

Обычно, аналитические сети строятся с целью определения значимости основных факторов, влияющих на процесс.

Анализ нейронными сетями ничем не отличается от нахождения оптимальных коэффициентов множественной нелинейной аппроксимации.

Сеть изображённая на рисунке 29 - это суперпозиция уравнений линейной регрессии.

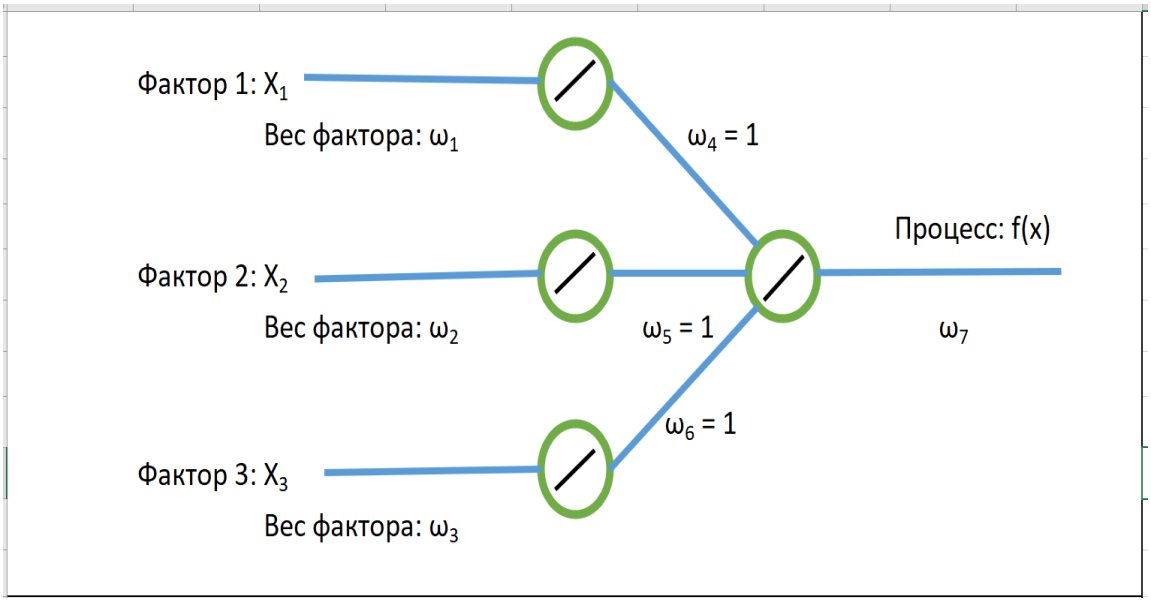


Рисунок 29 – Суперпозиция уравнений линейной регрессии

Формула сети:

$$f(x) = \omega_7 \sum_{i=1}^{i=3} x_i \omega_i \omega_{i+3} \tag{22}$$

Значимость факторов в данном случае будет выражаться, как:

$$P(x_i) = \omega_i \omega_7 \tag{23}$$

Пример № 1

Изучим влияние трёх внешних факторов на доходность российского рубля относительно доллара США.

Для простоты будем использовать нейронную сеть с линейными активаторами, представленную выше.

На первом этапе получим данные (рисунок 30)

1 - Получим данные с сайта ФИНАМ				
Дата	Доллар	Юань	Нефть	Золото
06.11.2019	63,835	9,082	61,79	1492,2
07.11.2019	63,6475	9,1121	62,38	1469,6
08.11.2019	63,7975	9,1235	62,61	1459,8
09.11.2019	63,7975	9,1235	62,61	1459,8

10.11.2019	63,7975	9,1235	62,4	1463,3
11.11.2019	63,7975	9,1264	62,3	1453,2
12.11.2019	64,03	9,1238	62	1462
13.11.2019	64,3825	9,1428	62,63	1466,4
14.11.2019	64,13	9,1404	62,41	1464,5
15.11.2019	63,75	9,1158	63,45	1468,7
16.11.2019	63,75	9,1158	63,45	1468,7
17.11.2019	63,75	9,1158	63,32	1469,6
18.11.2019	63,8925	9,0955	62,29	1472,1

Рисунок 30 – Получение данных

На втором этапе рассчитаем логарифмическую доходность (рисунок 31)

2 - Рассчитаем логарифмическую доходность				
Дата	Доллар	Юань	Нефть	Золото
06.11.2019				
07.11.2019	-0,00128	0,001437	0,004127	-0,00663
08.11.2019	0,001022	0,000543	0,001598	-0,00291
09.11.2019	0	0	0	0
10.11.2019	0	0	-0,00146	0,00104
11.11.2019	0	0,000138	-0,0007	-0,00301
12.11.2019	0,00158	-0,00012	-0,0021	0,002622
13.11.2019	0,002384	0,000903	0,004391	0,001305
14.11.2019	-0,00171	-0,00011	-0,00153	-0,00056
15.11.2019	-0,00258	-0,00117	0,007177	0,001244
16.11.2019	0	0	0	0
17.11.2019	0	0	-0,00089	0,000266
18.11.2019	0,00097	-0,00097	-0,00712	0,000738

Рисунок 31 – Логарифмическая доходность

На третьем этапе проведем обработку данных (рисунок 32).

3 - Удалим не торговые дни				
Дата	Доллар	Юань	Нефть	Золото
07.11.2019	-0,00128	0,001437	0,004127	-0,00663
08.11.2019	0,001022	0,000543	0,001598	-0,00291
12.11.2019	0,00158	-0,00012	-0,0021	0,002622
13.11.2019	0,002384	0,000903	0,004391	0,001305
14.11.2019	-0,00171	-0,00011	-0,00153	-0,00056
15.11.2019	-0,00258	-0,00117	0,007177	0,001244
18.11.2019	0,00097	-0,00097	-0,00712	0,000738

19.11.2019	-0,00061	-0,00065	-0,01044	0,000825
20.11.2019	0,001003	0,001265	0,011488	-5,9E-05
21.11.2019	-0,00167	-0,00222	0,008881	-0,00281
22.11.2019	-0,00019	-0,00067	-0,00157	-0,0011
25.11.2019	0,002058	0,001342	-0,00688	-0,00197
26.11.2019	0,000254	0,001142	0,003112	0,001997

Рисунок 32 – Обработка данных

На третьем этапе создадим простую сеть из четырёх линейных нейронов (рисунок 33-34).

AC5 =СУММКВРАЗН(Z5:Z299;AB5:AB299)

	A	B	C	D	F	G	H	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	Создадим простую сеть из четырёх линейных нейронов.				Нейрон 1			Нейрон 2				Нейрон 3				Нейрон 4											
2					X_1	ω_1	υ_1	X_2	ω_2	υ_2		X_3	ω_3	υ_3		X_4	ω_4	X_5	ω_5	X_6	ω_6	υ_4	ω_7	f(x)		Эталон	Ошибка сети
3																											
4	Доллар	Юань	Нефть	Золото		0,1			0,1				0,1				0,1		0,1		0,1			0,1			
5	-0,0013	0,0014	0,0041	-0,0066	0,0014		0,0001	0,0041		0,0004		-0,0066		-0,0007		0,0001		0,0004		-0,0007		-0,0006		-0,0001		-0,0013	**0,00599**
6	0,0010	0,0005	0,0016	-0,0029	0,0005		0,0001	0,0016		0,0002		-0,0029		-0,0003		0,0001		0,0002		-0,0003		-0,0003		0,0000		0,0010	
7	0,0016	-0,0001	-0,0021	0,0026	-0,0001		0,0000	-0,0021		-0,0002		0,0026		0,0003		0,0000		-0,0002		0,0003		0,0002		0,0000		0,0016	
8	0,0024	0,0009	0,0044	0,0013	0,0009		0,0001	0,0044		0,0004		0,0013		0,0001		0,0001		0,0004		0,0001		0,0002		0,0000		0,0024	
9	-0,0017	-0,0001	-0,0015	-0,0006	-0,0001		0,0000	-0,0015		-0,0002		-0,0006		-0,0001		0,0000		-0,0002		-0,0001		-0,0001		0,0000		-0,0017	
10	-0,0026	-0,0012	0,0072	0,0012	-0,0012		-0,0001	0,0072		0,0007		0,0012		0,0001		-0,0001		0,0007		0,0001		0,0002		0,0000		-0,0026	
11	0,0010	-0,0010	-0,0071	0,0007	-0,0010		-0,0001	-0,0071		-0,0007		0,0007		0,0001		-0,0001		-0,0007		0,0001		0,0000		0,0000		0,0010	
12	-0,0006	-0,0006	-0,0104	0,0008	-0,0006		-0,0001	-0,0104		-0,0010		0,0008		0,0001		-0,0001		-0,0010		0,0001		0,0000		0,0000		-0,0006	
13	0,0010	0,0013	0,0115	-0,0001	0,0013		0,0001	0,0115		0,0011		-0,0001		0,0000		0,0001		0,0011		0,0000		0,0001		0,0000		0,0010	
14	-0,0017	-0,0022	0,0089	-0,0028	-0,0022		-0,0002	0,0089		0,0009		-0,0028		-0,0003		-0,0002		0,0009		-0,0003		-0,0002		0,0000		-0,0017	
15	-0,0002	-0,0007	-0,0016	-0,0011	-0,0007		-0,0001	-0,0016		-0,0002		-0,0011		-0,0001		-0,0001		-0,0002		-0,0001		-0,0001		0,0000		-0,0002	
16	0,0021	0,0013	-0,0069	-0,0020	0,0013		0,0001	-0,0069		-0,0007		-0,0020		-0,0002		0,0001		-0,0007		-0,0002		-0,0003		0,0000		0,0021	
17	0,0003	0,0011	0,0031	0,0020	0,0011		0,0001	0,0031		0,0003		0,0020		0,0002		0,0001		0,0003		0,0002		0,0002		0,0000		0,0003	
18	-0,0003	0,0003	0,0004	-0,0013	0,0003		0,0000	0,0004		0,0000		-0,0013		-0,0001		0,0000		0,0000		-0,0001		-0,0001		0,0000		-0,0003	
19	0,0020	-0,0003	-0,0179	0,0024	-0,0003		0,0000	-0,0179		-0,0018		0,0024		0,0002		0,0000		-0,0018		0,0002		0,0001		0,0000		0,0020	
20	0,0001	-0,0007	-0,0013	0,0001	-0,0007		-0,0001	-0,0013		-0,0001		0,0001		0,0000		-0,0001		-0,0001		0,0000		0,0000		0,0000		0,0001	
21	-0,0010	0,0003	0,0019	0,0041	0,0003		0,0000	0,0019		0,0002		0,0041		0,0004		0,0000		0,0002		0,0004		0,0004		0,0000		-0,0010	
22	-0,0016	-0,0015	0,0131	-0,0004	-0,0015		-0,0002	0,0131		0,0013		-0,0004		0,0000		-0,0002		0,0013		0,0000		0,0001		0,0000		-0,0016	
23	-0,0008	-0,0009	0,0017	0,0000	-0,0009		-0,0001	0,0017		0,0002		0,0000		0,0000		-0,0001		0,0002		0,0000		0,0000		0,0000		-0,0008	
24	-0,0007	0,0000	0,0067	-0,0049	0,0000		0,0000	0,0067		0,0007		-0,0049		-0,0005		0,0000		0,0007		-0,0005		-0,0004		0,0000		-0,0007	
25	0,0001	0,0001	-0,0008	0,0003	0,0001		0,0000	-0,0008		-0,0001		0,0003		0,0000		0,0000		-0,0001		0,0000		0,0000		0,0000		0,0001	
26	-0,0008	-0,0012	0,0009	0,0008	-0,0012		-0,0001	0,0009		0,0001		0,0008		0,0001		-0,0001		0,0001		0,0001		0,0001		0,0000		-0,0008	
27	-0,0003	-0,0001	-0,0016	0,0031	-0,0001		0,0000	-0,0016		-0,0002		0,0031		0,0003		0,0000		-0,0002		0,0003		0,0003		0,0000		-0,0003	
28	-0,0042	-0,0020	0,0036	-0,0028	-0,0020		-0,0002	0,0036		0,0004		-0,0028		-0,0003		-0,0002		0,0004		-0,0003		-0,0003		0,0000		-0,0042	
29	-0,0013	-0,0016	0,0034	0,0032	-0,0016		-0,0002	0,0034		0,0003		0,0032		0,0003		-0,0002		0,0003		0,0003		0,0003		0,0000		-0,0013	
30	-0,0016	0,0006	0,0027	0,0004	0,0006		0,0001	0,0027		0,0003		0,0004		0,0000		0,0001		0,0003		0,0000		0,0001		0,0000		-0,0016	
31	0,0001	-0,0019	0,0033	0,0001	-0,0019		-0,0002	0,0033		0,0003		0,0001		0,0000		-0,0002		0,0003		0,0000		0,0000		0,0000		0,0001	

Рисунок 33 – Нейронная сеть из четырех нейронов

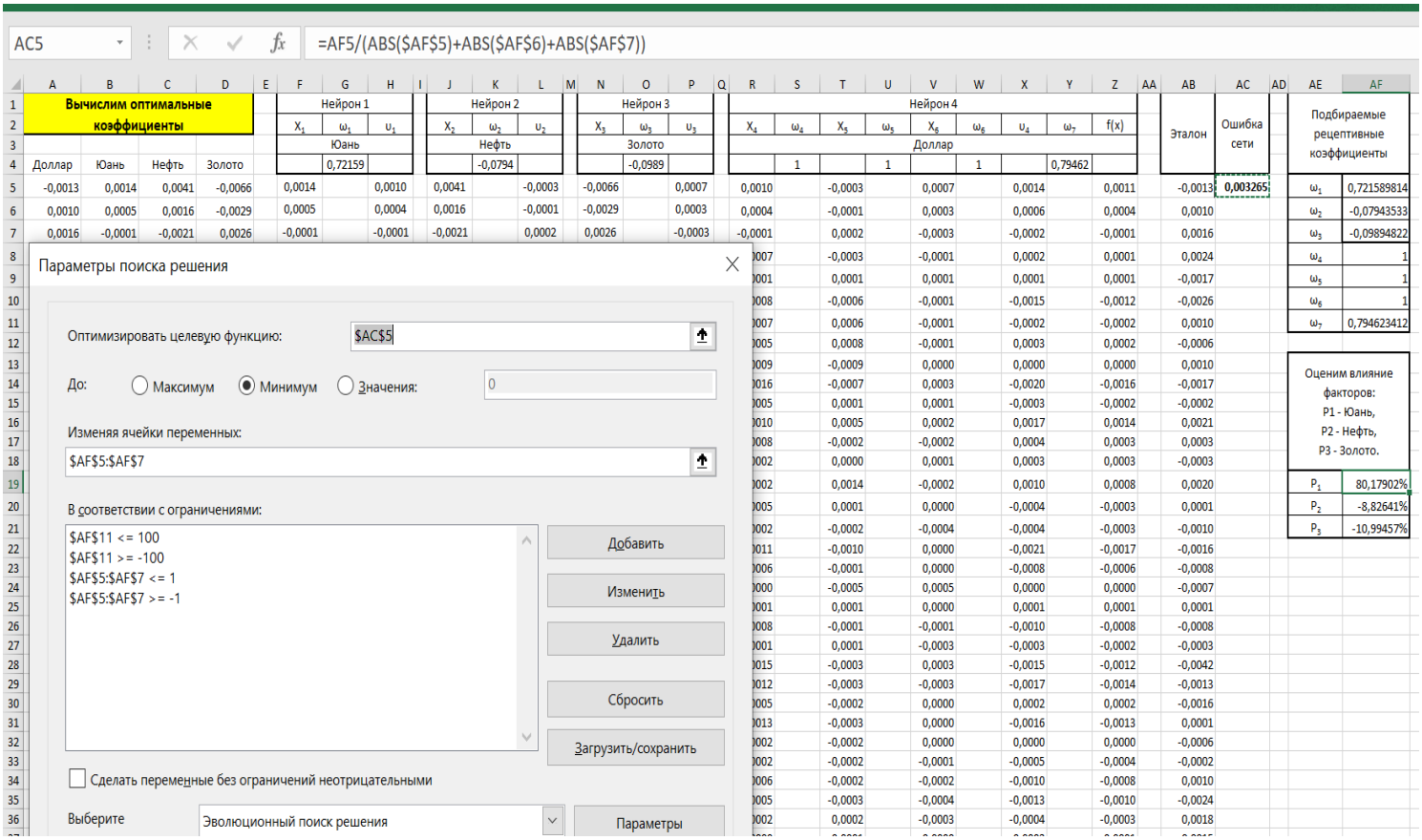


Рисунок 34 – Подбор оптимальных коэффициентов сети

Оценим влияние факторов: P1 - Юань, P2 - Нефть, P3 - Золото.	
P_1	80,17905%
P_2	-8,82630%
P_3	-10,99465%

Рисунок 35 – Результат сети

Задание 1

Выберите произвольные ряды данных, изучите их взаимное влияние при помощи нейронного анализа.

3.2 Введение в прогнозирование на основе нейронных сетей

Рассмотрим реализацию нейронных сетей в Excel.

Один нейрон (рисунок 36):

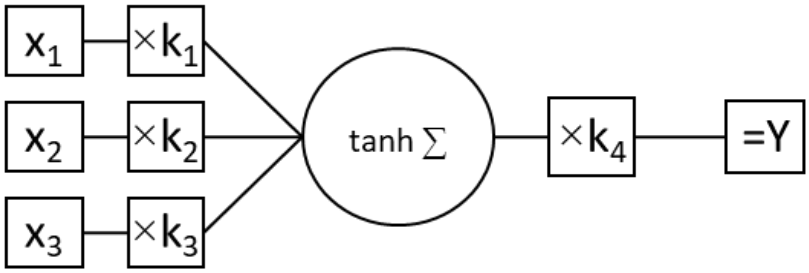


Рисунок 36 – Один нейрон

$$Y = k_4 \tanh \sum_{n=1}^{n=3} k_n x_n\,;\; k_{1-3} \in \left[\frac{-3}{x_{max}}, \frac{3}{x_{max}}\right];\; k_4 \in [\lfloor x_{min} \rfloor, \lceil 6x_{max} \rceil] \qquad (24)$$

EXCEL: **=$K$4*TANH(X1*$K$1+X2*$K$2+X3*$K$3)**

Однослойная сеть (рисунок 37):

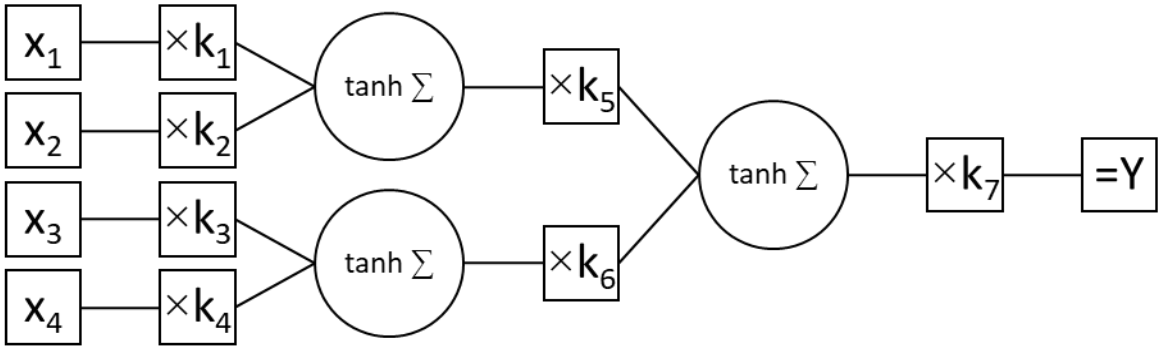


Рисунок 37 – Однослойная сеть

$$Y = k_7 \tanh(k_6 \tanh \sum_{n=3}^{n=4} k_n x_n + k_5 \tanh \sum_{n=1}^{n=2} k_n x_n)\,; \qquad (25)$$

$$k_{1-4} \in \left[\frac{-3}{x_{max}}, \frac{3}{x_{max}}\right];\; k_{5,6} \in [-1,1];\; k_7 \in [\lfloor x_{min} \rfloor, \lceil 6x_{max} \rceil]$$

EXCEL: =$K$7*TANH ($K$5*TANH (X1*$K$1+X2*$K$2) + $K$6*TANH (X3*$K$3+ X4*$K$4))

Многослойная сеть (рисунок 38):

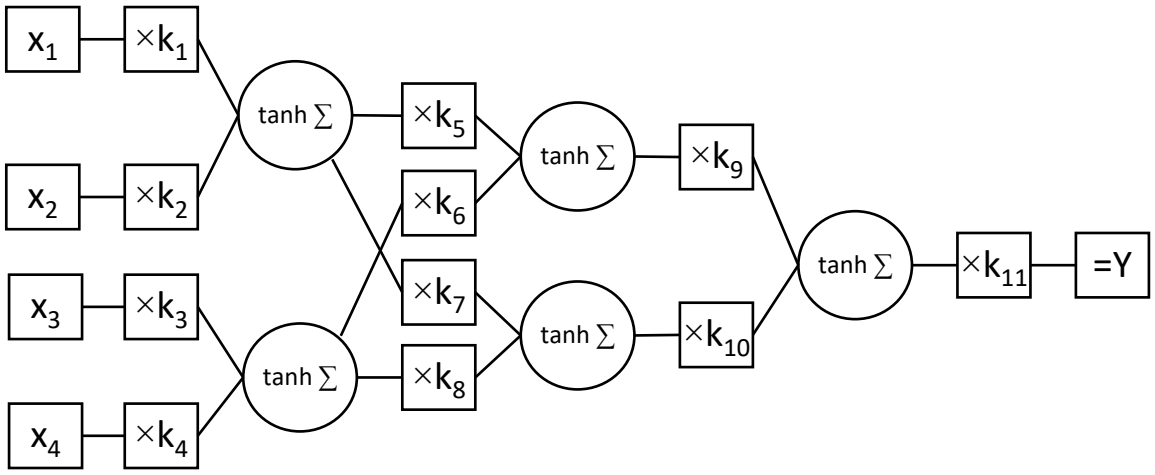


Рисунок 38 – Многослойная сеть

$$Y = k_{11}\tanh\left(k_9\tanh\left(k_5\tanh\sum_{n=1}^{n=2}k_nx_n + k_6\tanh\sum_{n=3}^{n=4}k_nx_n\right)\right.$$
$$\left. + k_{10}\tanh\left(k_7\tanh\sum_{n=1}^{n=2}k_nx_n + k_9\tanh\sum_{n=3}^{n=4}k_nx_n\right)\right)$$

$$k_{1-4} \in \left[\frac{-3}{x_{max}}, \frac{3}{x_{max}}\right];\ k_{5-10} \in [-1,1];\ k_{11} \in [\lfloor x_{min}\rfloor, \lceil 6x_{max}\rceil]$$

EXCEL: =$K$11*TANH ($K$9*TANH ($K$5*TANH (X1*$K$1+X2*$K$2) + $K$6*TANH (X3*$K$3+ X4*$K$4))
+ $K$10*TANH ($K$7*TANH (X1*$K$1+X2*$K$2) + $K$8*TANH (X3*$K$3+ X4*$K$4)))

Ознакомьтесь с примером №1 в EXCEL, самостоятельно рассчитайте прогноз на одном нейроне с 6-ю входами для текущих значений курса Евро.

Пример 1

Рассмотрим построение автокорреляционного нейронного прогноза одним нейроном, для курса доллара США. Предположим, что завтрашний курс доллара зависит от пяти предыдущих значений. Построим временной ряд со сдвигом в 1 день и оценим автокорреляцию (рисунок 39 – 40).

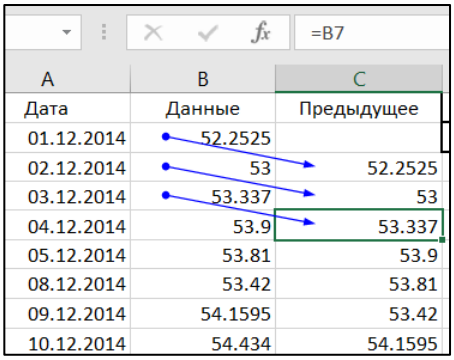


Рисунок 39 – Данные

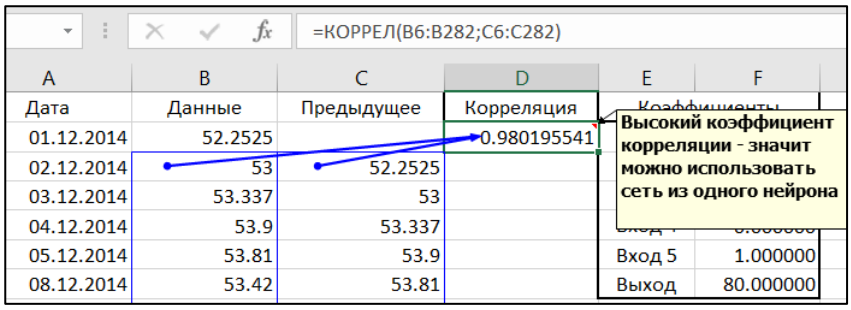


Рисунок 40 – Автокорреляция

Рассчитаем границы коэффициентов нейрона, как $\left[\frac{-3}{x_{max}}, \frac{3}{x_{max}}\right]$ и $[\lfloor x_{min} \rfloor, \lceil 6x_{max} \rceil]$ (рисунок 41).

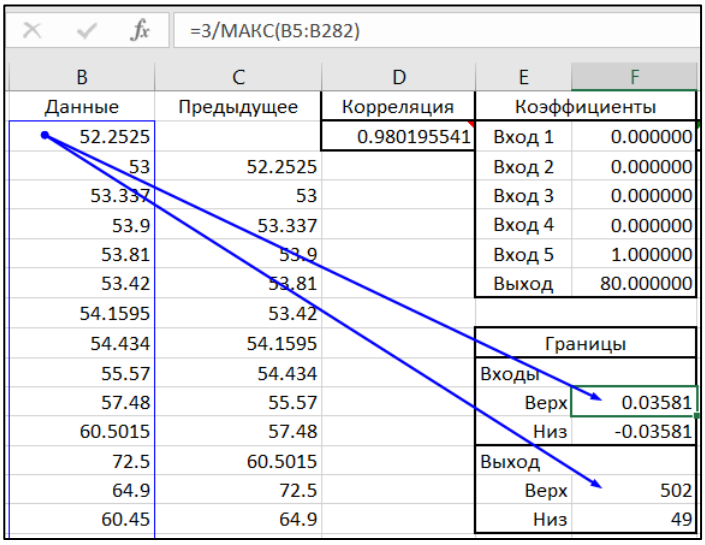


Рисунок 41 – Границы коэффициентов нейрона

Зададим начальные значения коэффициентов и создадим формулу нейронной сети (рисунок 42).

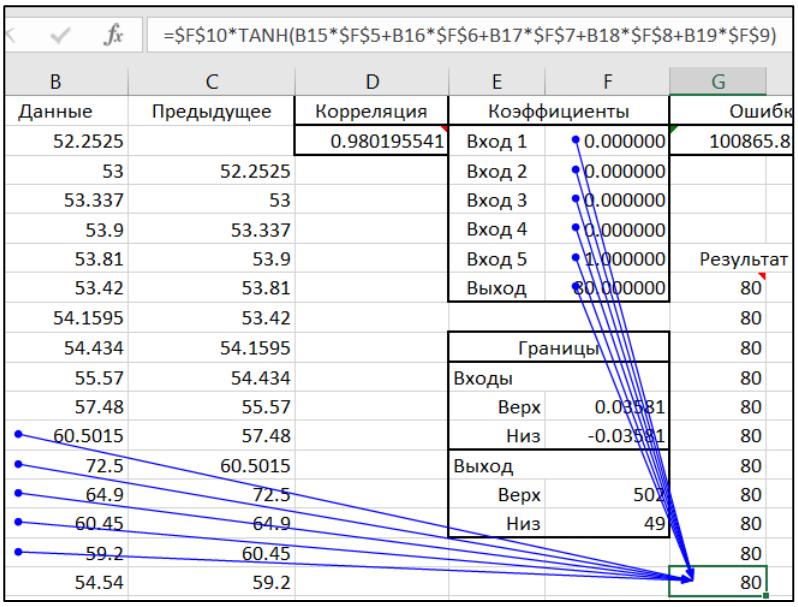


Рисунок 42 – Формула нейронной сети

Рассчитаем ошибку сети как сумму квадратов отклонений (рисунок 43).

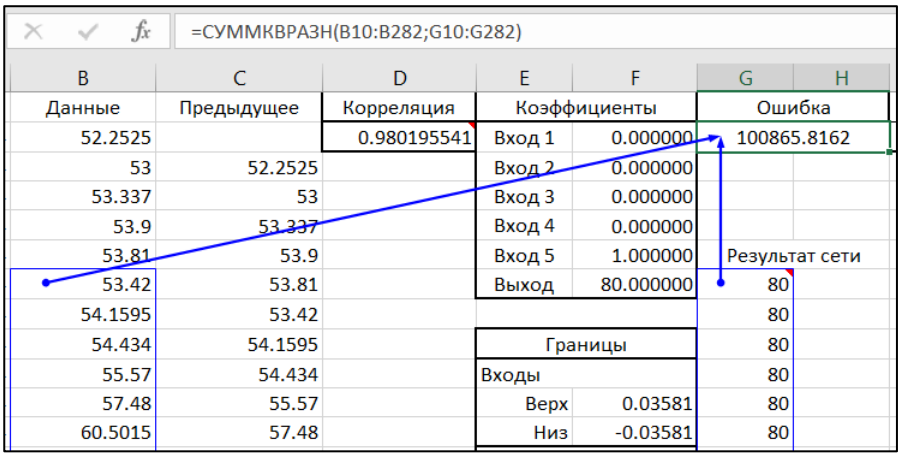


Рисунок 43 – Ошибка сети

Продолжим формулу нейронной сети на одно значение вперёд, сформируем данные прогноза, как результат сети с учётом среднеквадратичного отклонения (рисунок 44).

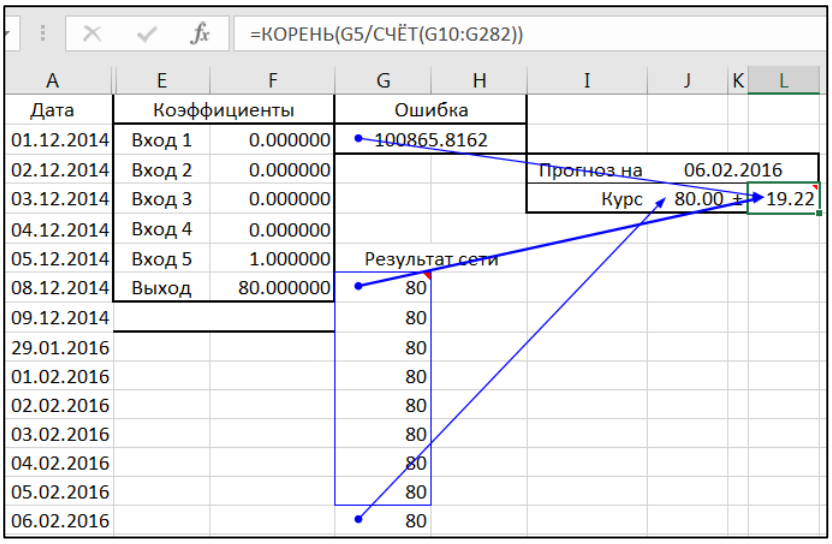


Рисунок 44 – Данные прогноза

Зададим ограничения коэффициентов нейронной сети и проведём их подбор эволюционным алгоритмом, используя как оптимизирующий критерий уменьшение ошибки (рисунок 45).

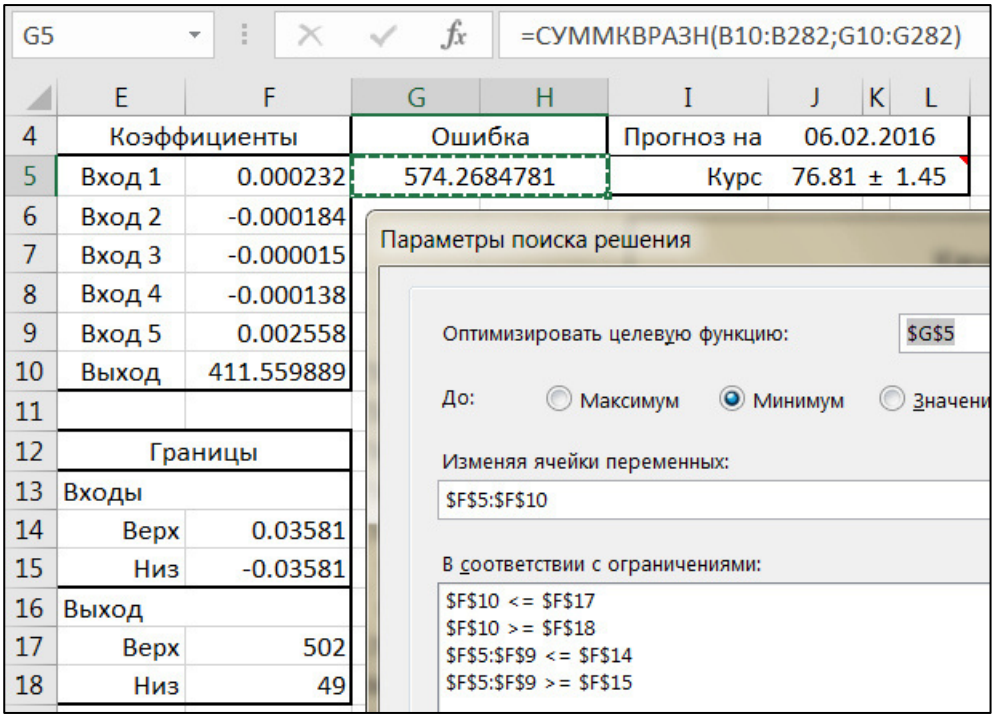


Рисунок 45 – Оптимизация сети

Получим прогноз и оценим его качество, построив график (рисунок 46).

Рисунок 46 – Результат обучения сети

Задание

1. Проведите интуитивный анализ входных и выходных данных. Постройте в подходящем масштабе графики прогнозируемой величины и величин, которые, предположительно влияют на неё. Рассчитайте величины их корреляции. Определите предполагаемые зависимости.
2. Определите подходящую архитектуру нейронной сети.

   В Excel без значительных затрат времени (до 1 часа на первичное обучение сети) возможно использовать следующие виды архитектуры:

- Для явных, зависимостей с высокой (>0.75) корреляцией: 1 нейрон с несколькими входами (до 20 входов) – 21 коэффициент.
- Для неявных, зависимостей с умеренной (>0.5) корреляцией: 1 слой до 10 нейронов с 1 входом + выходной нейрон – 21 коэффициент.

- Для неявных, зависимостей с низкой (<0.5) корреляцией: 2 слоя до 3-х нейронов с 3 входами + выходной нейрон – 22 коэффициента.

3. Задайте формулу сети, формулу ошибки сети и определите обучающую выборку.
4. Рассчитайте значения границ входных и выходных коэффициентов.

   Для сетей с тангенциальной сигма-функцией:

   - границы входных коэффициентов: $\left[\frac{-3}{x_{max}}, \frac{3}{x_{max}}\right]$
   - границы выходных коэффициентов: $[\lfloor x_{min} \rfloor, \lceil 6x_{max} \rceil]$

   Для сетей с рациональной сигма-функцией для $\alpha = 1$:

   - границы входных коэффициентов: $\left[\frac{-6}{x_{max}}, \frac{6}{x_{max}}\right]$
   - границы выходных коэффициентов: $[-12x_{max}, 12x_{max}]$
5. Задайте границы и проведите обучение сети эволюционным алгоритмом.
6. Получите от обученной сети прогноз для текущих данных.

ГЛАВА 4 ТИПОВЫЕ ЗАДАЧИ РЕШАЕМЫЕ НЕЙРОННЫМИ СЕТЯМИ

На рисунке 47 представлены типовые задачи, решаемые нейронными сетями.

Рисунок 47 – Типовые задачи, решаемые нейронными сетями

Прогнозирование – задача по получению оценки будущих значений упорядоченных во времени данных на основе анализа уже имеющихся, а также (при необходимости) тенденции изменения влияющих факторов.

Классификация – задача, в которой имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется выборкой. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

В современных задачах классификации от алгоритмов требуют не просто выдачу ответа, но и некоторой дополнительной информации, например уверенности в ответе. В данном случае, можно потребовать, чтобы алгоритм по описанию объекта выдавал значения из отрезка [0,1]. Полученное значение можно рассматривать, как вероятность того, что объект принадлежит тому или иному классу.

Задача классификации может являться как самостоятельной задачей, так и частью другой задачи машинного обучения. Например, прежде чем прогнозировать продажи товаров имеет смысл разбить их на группы, классифицировать, в зависимости от того насколько хорошо они поддаются прогнозированию (XYZ-группировка) и по объемам продаж (ABC-группировка) и строить прогноз уже в разрезе заданной группы.

Кластеризация – задача разбиения заданной выборки объектов (ситуаций) на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались.

4.1 История и классификация нейронных сетей[1]

Основные этапы в истории исследования и применения искусственных нейронных сетей:

• 1943 — У. Маккалок и У. Питтс формализуют понятие нейронной сети в фундаментальной статье о логическом исчислении идей и нервной активности.

[1] https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C

• 1948 — Н. Винер вместе с соратниками публикует работу о кибернетике. Основной идеей является представление сложных биологических процессов математическими моделями.

• 1949 — Д. Хебб предлагает первый алгоритм обучения.

• В 1958 Ф. Розенблатт изобретает однослойный перцептрон и демонстрирует его способность решать задачи классификации. Перцептрон обрёл популярность — его используют для распознавания образов, прогнозирования погоды и т. д.

• В 1960 году Уидроу совместно со своим студентом Хоффом на основе дельта-правила (формулы Уидроу) разработали Адалин, который сразу начал использоваться для задач предсказания и адаптивного управления. Сейчас Адалин (адаптивный сумматор) является стандартным элементом многих систем обработки сигналов.

• В 1963 году в Институте проблем передачи информации АН СССР. А. П. Петровым проводится подробное исследование задач «трудных» для перцептрона.

• В 1969 году М. Минский публикует формальное доказательство ограниченности перцептрона и показывает, что он неспособен решать некоторые задачи (проблема «чётности» и «один в блоке»), связанные с инвариантностью представлений. Интерес к нейронным сетям резко спадает.

• В 1972 году Т. Кохонен и Дж. Андерсон независимо предлагают новый тип нейронных сетей, способных функционировать в качестве памяти.

• В 1973 году Б. В. Хакимов предлагает нелинейную модель с синапсами на основе сплайнов и внедряет её для решения задач в медицине, геологии, экологии.

• 1974 — Пол Дж. Вербос и А. И. Галушкин одновременно изобретают алгоритм обратного распространения ошибки для обучения многослойных перцептронов

• 1975 — Фукусима представляет когнитрон — самоорганизующуюся сеть, предназначенную для инвариантного распознавания образов, но это

достигается только при помощи запоминания практически всех состояний образа.

• 1982 — после периода забвения, интерес к нейросетям вновь возрастает. Дж. Хопфилд показал, что нейронная сеть с обратными связями может представлять собой систему, минимизирующую энергию (так называемая сеть Хопфилда). Кохоненом представлены модели сети, обучающейся без учителя (нейронная сеть Кохонена), решающей задачи кластеризации, визуализации данных (самоорганизующаяся карта Кохонена) и другие задачи предварительного анализа данных.

• 1986 — Дэвидом И. Румельхартом, Дж. Е. Хинтоном и Рональдом Дж. Вильямсом и одновременно с С. И. Барцевым и В. А. Охониным (Красноярская группа) переоткрыт и существенно развит метод обратного распространения ошибки. Начался взрыв интереса к обучаемым нейронным сетям.

• 2007 Джеффри Хинтоном в университете Торонто созданы алгоритмы глубокого обучения многослойных нейронных сетей. Успех обусловлен тем, что Хинтон при обучении нижних слоев сети использовал ограниченную машину Больцмана (RBM — Restricted Boltzmann Machine).

Классификация нейронных сетей

Одним из решающих вопросов в рамках подготовки ИНС можно назвать выбор архитектуры. Подобный выбор должен быть основан на возможностях и особенностях работы различных систем. Данный вопрос подготовки является наиболее значимым ввиду того, что ошибка, совершенная на этом этапе анализа данных, не может быть исправлена в дальнейшем и приведет к полной переработке системы. Однако для решения этого вопроса нет четких алгоритмов, и оно основывается в основном на опыте применения и знаниях, необходимых для успешного применения той или иной ИНС.

Нейронные сети, можно классифицировать в зависимости от модели связей между нейронами системы. В общем случае такие связи можно разделить на две крупные группы - рекуррентные (рисунок 48 а), где сигнал со

скрытого слоя передается на вход ИНС, и нерекуррентные (рисунок 48 б, в), где такая связь отсутствует. В данный момент для анализа временных рядов наибольшим спросом пользуются подходы создания ИНС, относящихся к рекуррентным сетям, однако и нерекуррентные сети находят свое применение для решения поставленных задач. Исторически, нерекуррентные сети были разработаны раньше рекуррентных, и их развитие сейчас менее интенсивно по сравнению с рекуррентными.

К нереккуррентным сетям относят нейронные сети:

1. Прямого распространения.

В подобных сетях сигнал распространяется только в одном направлении - от входа к выходу. Существуют варианты однослойного и многослойного перцептрона или сети с временной задержкой, где сигнал поступает на следующий уровень сети с определенной задержкой. Подобные архитектуры считаются самыми простыми с точки зрения сложности внутренней архитектуры. Как правило они не имеют скрытых слоев и во время работы, фактически такая сеть умножает вектора входных данных на матрицу весовых функций с добавлением вектора смещения на выходе. В качестве примера можно привести нейронную сеть с временной задержкой, которая является также нейронной сетью прямого распространения и достаточно проста в реализации. Нейросети прямого распространения считаются наиболее простыми, однако могут приводить к ошибкам в полученных данных.

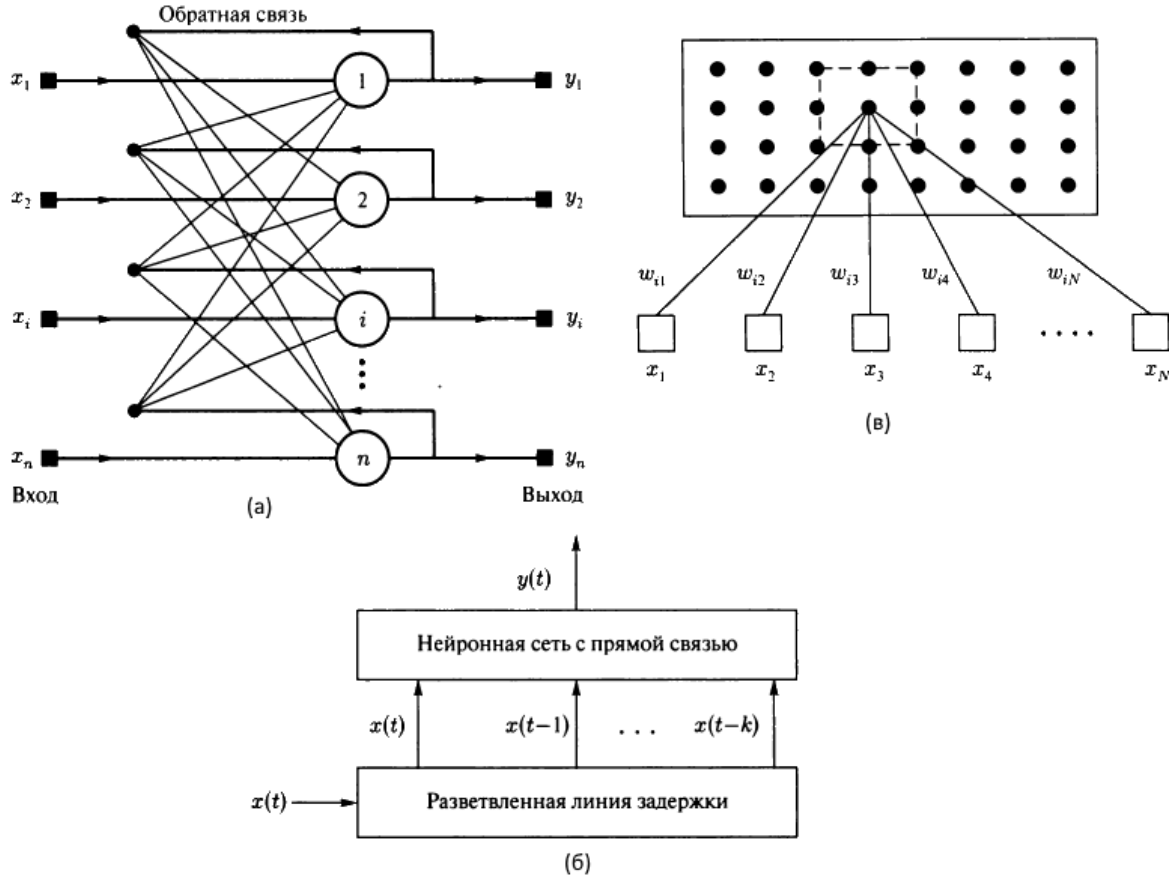


Рисунок 48 – а) Рекуррентная нейронная сеть Хопфилда; б) Нейронная сеть с временной задержкой; в) Самоорганизующуюся Сеть Кохонена

2. С радиально-базисными функциями.

В радиально-базисных сетях функции работы сети похожи на некоторые усложненные варианты сетей с прямой передачей данных, однако, такие сети реализуют иной метод обработки данных. В отличии от нейронных сетей прямого распространения, где преобразование выполняется усилием многих нейронов, в радиально-базисных ИНС преобразование происходит путем адаптации одиночных функций к ожидаемым значениям. Типовая структура радиальной базисной сети включает входной слой с вектором входных сигналов, скрытый слой с нейронами радиального типа и выходной слой из одного или нескольких нейронов, чья функция сводится к взвешенному суммированию сигналов со скрытых слоев.

3. С самоорганизующимися картами.

ИНС с самоорганизующимися картами нашли применение в кластеризованных временных рядах. В таких рядах независимо от алгоритма кластеризации применяется параметр расстояния между точками кластеризованного множества зависимый от входных параметров. Качество прогнозов полученных с помощью подобных ИНС напрямую зависит от того насколько оптимален был такой набор параметров. В типовую сеть с самоорганизующимися картами входит всего два слоя - входной и выходной. В зависимости от детализации этих слоёв изменяется точность обобщающей способности слоя и сложность создания ИНС в целом.

Как видно из особенностей приведенных нерекуррентных сетей, все они способны проводить обработку временных рядов для формирования прогнозов, однако обладают своими, уникальными, показателями. Такая уникальность подобных сетей делает их либо узконаправленными как ИНС с самоорганизующимися картами, которые чаще всего используют для прогнозирования клиентского поведения или поиска аномалий, либо радикально простых моделей для быстрого поиска результата с помощью ИНС линейного распространения.

Конкуренцию всему многообразию класса нереккуррентных сетей составляют рекуррентные сети с различной памятью обладающими своими недостатками и достоинствами, при этом основным достоинством и в тоже время существенным недостатком рекуррентных сетей в любом их виде является способность анализировать события, связанные по времени, то есть эти сети учитывают влияние соседних отсчетов временного ряда. Это позволяет быстрее находить корреляцию данных, изменяющихся во времени и формировать более унифицированное решение для анализа любых временных рядов нежели нерекуррентные ИНС. Однако, в специализированных задачах применение нерекуррентных сетей позволяет получать результат такого же качества при более простых методах обучения сетей.

4. Глубокие нейронные сети

Сети с большим количеством нейронов (сотни тысяч), скрытых слоёв (десятки и сотни) и сложной архитектурой называют глубокими. Обучение таких сетей требует значительного времени и большого количества как эталонных, так и ошибочных образцов решений. Однако, результативность таких сетей весьма значима, что позволяет использовать их в серьёзных научных и коммерческих проектах.

Специализированные алгоритмы глубокого обучения представляют из себя те или иные модификации смешанного метода, ориентированные на параллельное исполнение на большом количестве нейронов или группе нейронных сетей. Для обучения и работы этого типа нейронных сетей используют нейронные процессоры (Neural Processing Unit, NPU) – специализированное оборудование, ориентированное исключительно на задачу математического моделирования нейронных сетей.

Наиболее известные коммерческие нейропроцессоры:

NVIDIA DGX-2 – 8.2×10^4 нейронов, 1.28×10^9 коэффициентов, 2×10^{15} FLOPS, габариты: 52×26×64 см, энергопотребление: 10^3 Вт, цена: $ 400 000. Готовый вычислительный комплекс для обучения сетей.

IBM TrueNorth – Одновременная обработка: 10^6 нейронов, 2.56×10^8 коэффициентов, 4.5×10^{12} FLOPS, габариты: 2×2×1 см, энергопотребление: 0.1 Вт, цена: $ 8 000. Готовые нейронные серверы под заказ.

Nvidia Tesla M4 – Одновременная обработка: 1024 нейронов, 2048 коэффициентов, 2.2×10^{12} FLOPS, габариты: 17×8×5 см, энергопотребление: 75 Вт, цена: $ 1 900. Для настольных компьютеров.

Apple A12 Bionic – Одновременная обработка: 8 нейронов, 16 коэффициентов, 5×10^{12} FLOPS, габариты: 3×3×0.5 см, энергопотребление: 6 Вт, цена: $ 300. Нейронный модуль в чипе IPhone XS.

4.2 Применение нейронных сетей в задачах прогнозирования

В настоящее время прогнозирование временных рядов является ключевым моментом в деятельности многих организаций, так как позволяет предсказать поведение различных факторов в экономических, экологических, социальных и иных системах. Основной целью любого прогнозирования является возможность оценить тенденции в изменениях того или иного фактора. Качество прогноза зависит от наличия предыстории изменяемого фактора, погрешностей измерения рассматриваемой величины, количества одновременно учитываемых параметров временного ряда.

Временным (*или динамическим*) *рядом* называется последовательность статистических данных (наблюдений) какого-либо признака (параметра случайной величины) в разные моменты времени, упорядоченные по возрастанию. Уровнем ряда называется каждое отдельное наблюдение y_t, где t - момент времени, когда было получено наблюдение, $t = 1,2, \dots, n$ (n — число наблюдений). В отличие от простой выборки данных, учитывающей только статистическое разнообразие и характеристики выборки, временной ряд предполагает взаимосвязь измерений со временем[2].

Для успешного прогнозирования временных рядов требуется:

1) тщательное изучение особенностей временного ряда и его основных характеристик;
2) разработка модели, описывающей структуру ряда;
3) определение прогнозных значений исходя из имеющихся наблюдений на основе ретроспективных данных[3].

Выделяют следующие виды динамических моделей:

1) *модели с распределенными лагами,* т. е. такие модели, в которых только независимые (объясняющие) переменные могут являться лаговыми. Модели с лагами имеют следующий вид:

[2] Подкорытова О.А., Соколов М.В. Анализ временных рядов: Учебное пособие. М.: Юрайт, 2020. 267 с
[3] Афанасьев В.Н., Юзбашев М.М. Анализ временных рядов и прогнозирование: Учебник. М.: Финансы и статистика, 2012. 228 с.

$$y_t = \alpha + \beta_0 x_t + \beta_1 x_{t-1} + K + \beta_k x_{t-k} + \varepsilon_t \quad (26)$$

2) *авторегрессионные модели*, т. е. такие модели, в которых зависимые переменные являются лаговыми. Авторегрессионные модели имеют следующий вид:

$$y_t = \alpha + \beta x_t + \gamma y_{t-1} + \varepsilon_t \quad (27)$$

Значения уровней временных рядов y_t складываются из четырёх компонент:

1) *тренд* u_t - отражает влияние долговременных, плавно изменяющихся, систематических факторов, формирует основную тенденцию показателя, описывается плавно меняющимися, гладкими функциями;
2) *сезонная компонента* s_t, связанная с повторяемостью в течение небольшого периода времени экономических процессов. Таким небольшим периодом времени может быть год, месяц, неделя. Описывается периодическими функциями;
3) *циклическая компонента* v_t, связанная с повторяемостью в течении длительных периодов экономических процессов. Примерами этого могут быть экономические, инвестиционные, демографические циклы;

   Первые три компоненты являются *неслучайными* или *закономерными*.
4) *случайная компонента* ε_t, связанная с влиянием случайных и неучтенных факторов.

Различают *аддитивную* модель временного ряда, когда ряд представляется как сумма вышеуказанных компонент, а также *мультипликативную модель* (если ряд представлен как произведение соответствующих компонент).

Аддитивная модель описывается следующим соотношением:

$$y_t = u_t + s_t + v_t + \varepsilon_t \quad (28)$$

Мультипликативная модель описывается следующим соотношением:

$$y_t = u_t s_t v_t \varepsilon_t \quad (29)$$

Модели временных рядов могут быть и смешанными, т.е. включающими

в себя и мультипликативную, и аддитивную составляющие.

Немаловажное значение при изучении временных рядов и описании их случайных составляющих следует придать *стационарным* временным рядам. Под стационарным временным рядом подразумевается:

- в узком смысле - ряд, закон распределения y_t и числовые характеристики которого не зависят от момента времени *t;*
- в широком смысле — ряд, первый и второй момент которого не зависит от времени[4].

Математическое ожидание для стационарного временного ряда вычисляется по следующей формуле:

$$\bar{y} = \frac{\sum_{t=1}^{n} y_t}{n} \qquad (30)$$

Среднее квадратическое отклонение для стационарного временного ряда вычисляется следующим образом:

$$S_t^2 = \frac{\sum_{t=1}^{n} (y_t - \bar{y}_t)^2}{n} \qquad (31)$$

Изучение и анализ временных рядов основывается на реализации следующих основных этапов:

1) временной ряд представляется графически;
2) идентификация и оценка неслучайных составляющих, исключение их из анализируемого ряда;
3) фильтрация данных, заключающаяся в удалении составляющих с низкими и высокими частотами;
4) анализ случайной составляющей, подтверждение адекватности построенной модели;
5) вычисление прогнозных значений на основе сформированной модели.

В современных условиях для успешной реализации любой деятельности необходимо осуществлять качественное и количественное прогнозирование

[4] Афанасьев В.Н., Юзбашев М.М. Анализ временных рядов и прогнозирование: Учебник. М.: Финансы и статистика, 2012. 228 с.

происходящих процессов. Расширяется сфера применения, в связи с чем становится более важной и сложной задача прогнозирования.

Увеличение роли прогнозирования в современном мире породило более ста моделей и методов прогнозирования. По этой причине встаёт задача выбора оптимального варианта прогнозирования исследуемого процесса или системы.

Строгие статистические предположения о характеристиках временных рядов существенно сужают границы применения методов математической статистики, теории случайных процессов, теории распознавания образов и т.п. Большинство действительных процессов нельзя корректно описать, используя традиционные статистические модели, так как реальные процессы не являются линейными и могут иметь или хаотическую, или квазипериодическую, или смешанную основу.

Нелинейность нейронных сетей позволяет устанавливать нелинейные зависимости между будущими и фактическими значениями процессов. Другими важными достоинствами является масштабируемость – параллельная структура искусственных нейронных сетей ускоряет вычисления, что является крайне актуальным в промышленных масштабах, когда необходимо обрабатываться терабайты данных.

Прогнозы становятся важными компонентами процесса принятия управленческих решений. Высокая точность – это главная характеристика любого прогноза.

Качества прогноза определяется в соответствии со следующими главными требованиями: точность, верифицируемость, актуализированность. Точность прогноза зависит от того, насколько сильно приближены прогнозные значения и сама модель прогнозирования к фактическому процессу или объекту прогнозирования. Верифицируемость – это количественная мера достоверности прогноза. Актуализированность связана с возможностью адаптации и корректировки модели в связи с изменениями внешней среды.

Показатели точности модели прогнозирования определяют значение полученной ошибки. Для подтверждения качества и пригодности используемой

модели нужно провести анализ системы показателей, отражающих как адекватность модели, так и ее точность. Точность прогноза определяется величиной ошибки (погрешности) прогноза. Ошибка прогноза Δt - величина, определяющаяся как разность между фактическим и прогнозным значениями показателя. С уменьшением величины ошибки, соответственно, повышается точность прогноза.

Абсолютная ошибка Δt прогноза временного ряда является аналитическим показателем, использование которого даёт возможность количественно рассчитать и оценить значение ошибки прогноза. Абсолютная ошибка Δt вычисляется по следующей формуле:

$$\Delta t = y_t - \hat{y}_t \quad (32)$$

где $\hat{y}_t$ - прогнозное значение показателя;

y_t - фактическое значение.

Относительная ошибка прогноза $\Delta_{\text{отн}}$ вычисляется путем деления абсолютной ошибки прогноза на фактическую величину признака y_t:

$$\Delta_{\text{отн}} = \frac{\Delta t}{y_t} = \frac{y_t - \hat{y}_t}{y_t} * 100\% \quad (33)$$

Средний квадрат ошибки вычисляется по следующей формуле:

$$\Delta t = \frac{(y_t - \hat{y}_t)^2}{y^2} \quad (34)$$

Согласно общей практике, точность модели соответствует основным требованиям, если среднее значение относительной погрешности не больше 5%. Точность модели считается удовлетворительной, когда среднее значение относительной погрешности не больше 15%. И наконец, точность модели считается неудовлетворительной в случае, когда среднее значение относительной погрешности превышает 15%.

Ошибка прогноза идентична по размерности прогнозируемому показателю и напрямую зависит от масштаба измерений уровней временного ряда. Таким образом, точность прогноза увеличивается с уменьшением этого значение. При разработке прогноза для сложной системы, процессы в которой определяются большим количеством показателей, нужно рассчитывать

обобщающий показатель точности. Это получается усреднением модулей абсолютных отклонений.

Средняя абсолютная ошибка прогноза $|\bar{\Delta}|$ вычисляется как простая средняя арифметическая из абсолютных ошибок прогноза и определяется по следующей формуле:

$$|\bar{\Delta}| = \frac{\sum_{t=1}^{n}|\Delta t|}{n} = \frac{\sum_{t=1}^{n}|y_t - \hat{y}_t|}{n} \quad (35)$$

где n - число уровней временного ряда, для которых определялось прогнозное значение.

$$|\bar{\Delta}_{\text{отн}}| = \frac{\sum_{t=1}^{n}|\Delta t|}{n} * 100\ \% = \frac{\sum_{t=1}^{n}|y_t - \hat{y}_t|}{n} * 100\ \% \quad (36)$$

Для оценки точности прогноза применяется средняя квадратическая абсолютная ошибка прогноза, вычисляемая по следующей формуле:

$$\bar{\Delta}_{\text{абс\_квадр}} = \sqrt{\frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{n}} \quad (37)$$

Для оценки точности статистических прогнозов часто используется предложеный Г. Тейлом коэффициент несоответствия (КН), определяемый по следующей формуле:

$$\text{КН} = \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y}_t)^2} \quad (38)$$

При этом $0 \leq \text{КН} \leq 1$, КН = 0, если фактические и прогнозные значения признака абсолютно идентичны.

Оценка качества количественного прогноза предполагает нахождение таких элементов математической статистики и теории вероятностей, как точечная $\bar{y}_i$ и (или) интервальная $\hat{y}_t$ оценка. Точечная оценка представляет собой единичную оценку прогнозного параметра. Интервальная оценка - является числовым интервалом (называется доверительным интервалом), в котором с наибольшей вероятностью лежит прогнозный параметр. Прогнозные значения, полученные в результате проведенных расчетов, должны реализоваться в соответствующее время и с указанной вероятностью. Данные значения должны быть ограничены некой доверительной областью, ширина которой зависит от заданной вероятности. Эта вероятность (P_i) вычисляется путем деления числа

событий, благоприятствующих ее появлению и выполнению прогноза, на общее число событий.

Таким образом, под достоверностью прогноза следует понимать вероятность осуществления прогноза в заданном доверительном интервале 2δ.

Границы доверительного интервала определяются по формуле:

$$\delta = t_a \sigma, \tag{39}$$

где σ - среднеквадратическое отклонение;

t_a - критерий Стьюдента.

Критерий Стьюдента находится исходя из размера выборки и заданной вероятности свершения прогноза. Чем больше вероятность прогноза и чем меньше размер выборки, тем шире должны быть границы доверительного интервала.

Среднеквадратическое отклонение характеризует наименьшую ошибку прогноза и зависит, с одной стороны, от корректности модели, с другой - от стабильности изучаемого показателя в прошлом:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}{n}} \tag{40}$$

где y_i - фактическое значение изучаемой характеристики на участке ретроспекции;

$\bar{y}_i$ - расчетное значение изучаемой характеристики на участке ретроспекции;

n - число наблюдений (размер выборки).

Проверка адекватности модели осуществляется на основе формальных статистических критериев. Обязательным условием такой проверки является наличие надежных и достоверных статистических параметров объекта прогнозирования и самой модели. При отсутствии таких оценок проводится сравнение отдельных свойств изучаемого объекта и модели. При этом первоначально должна проверяться истинность реализуемых функций, затем правильность структуры и, наконец, достоверность достигаемых при этом

значений параметров. На данном этапе осуществляется сопровождающее моделирование, для которого нужно иметь не только саму модель, но и функционирующий оригинал.

Верификация модели – представляет собой анализ функциональной полноты, точности и достоверности модели, осуществляемый на основе всей доступной информации в тех ситуациях, когда проверку адекватности по каким-либо причинам провести нельзя.

Метод верификации очень часто используется в прогнозировании при отсутствии реального объекта, а также при разработке новых функций некоторого объекта прогнозирования.

Как показывает практика, абсолютное совпадение прогноза и реального процесса происходит крайне редко, в связи с чем задача верификации прогнозной модели остается весьма актуальной.

Отличительной особенностью искусственных нейронных сетей (ИНС) применительно к задачам прогнозирования временных рядов является присущая им способность нелинейного моделирования.

Искусственные нейронные сети – это набор простых искусственных нейронов, соединенных в единую систему, взаимосвязанных и взаимодействующих друг с другом.

Несколько отличительных особенностей ИНС делают их ценными и привлекательными в прогнозировании.

Во-первых, ИНС оперируют нелинейными данными. Они способны выполнять нелинейное моделирование без априорных знаний о взаимосвязях между входными и выходными переменными. Таким образом, они являются более общими и гибкими инструментами моделирования для прогнозирования. Непараметрическая модель ИНС может быть предпочтительнее традиционных параметрических статистических моделей в ситуациях, когда входные данные не соответствуют допущениям, требуемым параметрической моделью, или когда в наборе данных видны большие выбросы.

Во-вторых, ИНС являются универсальными функциональными аппроксиматорами. Сеть может аппроксимировать всякую непрерывную функцию с любой требуемой точностью. ИНС имеет более общие и гибкие функциональные формы, чем статистические методы.

В-третьих, ИНС могут обобщать информацию. Поскольку прогнозирование выполняется посредством определения будущего поведения на основе прошлого, оно является идеальной областью использования для нейронных сетей. Эти уникальные особенности делают ИНС полезными для решения многих практических задач прогнозирования[5] .

Нейронные сети отлично подходят для нахождения точных решений в среде, характеризуемой сложной или фрагментарной информацией. В области финансов и экономики значения параметров временных рядов могут более точно моделироваться с помощью методов, отличных от традиционных линейных статистических методов.

Метод прогнозирования на основе нейросетей может быть использован для решения широкого спектра задач.

Основные **преимущества** ИНС по сравнению с другими методами и моделями прогнозирования:

1) ИНС пригодны для реализации задач для неизвестных закономерностей развития процесса и неизвестной взаимосвязи между входными и выходными данными. Классические математические методы и традиционные экспертные системы в данной ситуации оказываются неэффективными;
2) ИНС оказываются эффективными даже при наличии шумов в исходных данных, т.е. нейронные системы весьма устойчивы при работе со значительным количеством малоинформативных, шумовых исходных сигналов, нейронная сеть самостоятельно анализирует данные, фильтрует их

[5] Дулькейт Е.И. Прогнозирование с помощью искусственных нейронных сетей // Прикладная математика и фундаментальная информатика. 2015. № 2. С. 118 – 126

и выбирает только пригодные данные, в связи с чем аналитику нет необходимости проводить отсев данных;

3) ИНС гибко реагируют на изменения окружающей среды, т.е. им присуща возможность адаптироваться. Так, работающие в одной среде нейронные сети возможно переобучить и настроить их на функционирование в условиях среды с небольшим изменением параметров. Если процесс осуществляется в нестационарной среде (т.е. статистика со временем меняется) есть возможность разработать нейронные сети, проходящие переобучение в режиме реального времени. Устойчивость работы ИНС в нестационарных условиях определяется тем, насколько высоки её адаптивные способности, хотя иногда адаптивность не ведет к устойчивости. Пример того может служить высоко адаптивная система с параметрами, значения которых меняются во времени с достаточно высокой скоростью. Такая система так же будет приспосабливаться к посторонним возбуждениям, что снизит её эффективность. Преимущества высокой адаптивности системы могут оказаться весьма полезными при соблюдении двух условий: во-первых, требуется стабильность главных параметров, что позволит не принимать во внимание внешние помехи, во-вторых, требуется гибкость параметров, позволяющая быстро реагировать на изменения внешних факторов;
4) ИНС потенциально могут решать сложные и трудоемкие задачи очень быстро, так как в них функционирует механизм массового параллелизма обработки информации;
5) ИНС при аппаратной реализации оказываются потенциально устойчивыми к неблагоприятным условиям и практически не теряют свою производительность, т. е. они отказоустойчивы. Так, лишь значительные повреждения и дефекты структуры ИНС могут серьёзно повлиять на эффективность её функционирования. Уменьшение качества функционирования ИНС происходит с медленной скоростью.

Несмотря на широкий спектр возможностей искусственные нейронные сети имеют ряд **недостатков** при решении задач:

1) для построения нейронной сети требуется выполнение множества настроек внутренних элементов сети и связей между ними;
2) сложность выбора алгоритма обучения нейронной сети;
3) жесткие требования к обучающей выборке;
4) продолжительные временные затраты на выполнение процедуры обучения сети не позволяют применять нейронные сети в режиме реального времени;
5) сложность выбора архитектуры нейронной сети;
6) большинство известных коммерческих решений для реализации нейронных сетей нельзя назвать широкодоступными.

Использование искусственных нейронных сетей позволяет делать более точные прогнозы, также сеть способна обучаться и обобщать полученные знания, адаптироваться, то есть проявляется высокая устойчивость к помехам. Но при решении задач прогнозирования с использованием нейронных сетей возникают трудности с подготовкой и обработкой входных значений, выбором подходящей архитектуры сети.

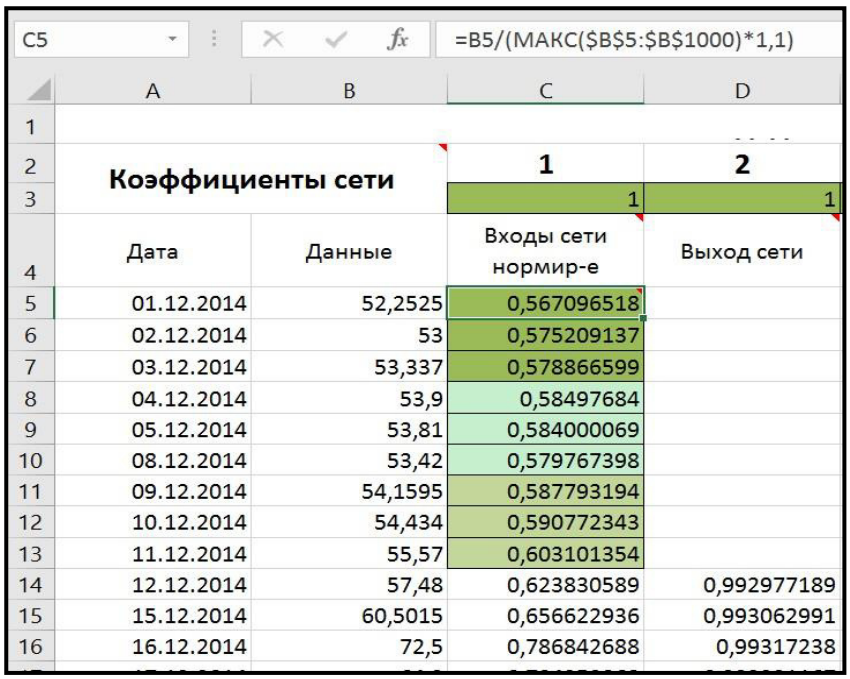
Пример № 1 Прогнозирование временных рядов в Excel

Необходимо построить автокорреляционный прогноз на один период для курса USD/RUB на однослойной нейронной сети.

Решение:

1. На первом этапе экспортируем с сайта finam.ru финансовый временной ряд USD/RUB за 2014-2016гг. Нормируем значения временного ряда (рисунок 49).

C5: =B5/(МАКС($B$5:$B$1000)*1,1)



C5 | =B5/(МАКС($B$5:$B$1000)*1,1)

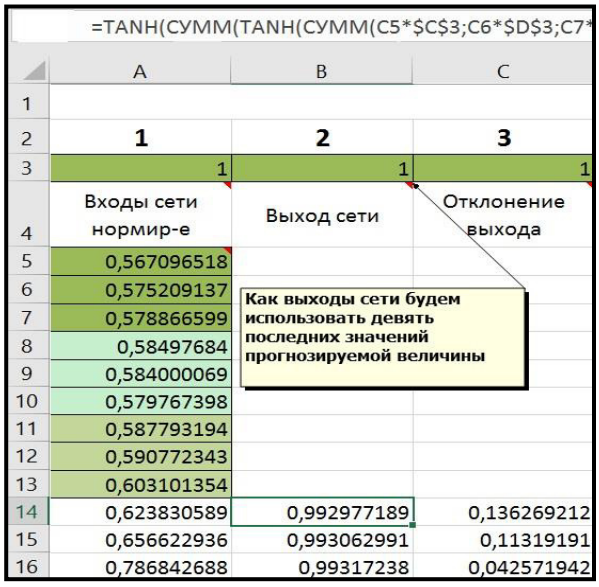
	A	B	C	D
1				
2	Коэффициенты сети		1	2
3			1	1
4	Дата	Данные	Входы сети нормир-е	Выход сети
5	01.12.2014	52,2525	0,567096518	
6	02.12.2014	53	0,575209137	
7	03.12.2014	53,337	0,578866599	
8	04.12.2014	53,9	0,58497684	
9	05.12.2014	53,81	0,584000069	
10	08.12.2014	53,42	0,579767398	
11	09.12.2014	54,1595	0,587793194	
12	10.12.2014	54,434	0,590772343	
13	11.12.2014	55,57	0,603101354	
14	12.12.2014	57,48	0,623830589	0,992977189
15	15.12.2014	60,5015	0,656622936	0,993062991
16	16.12.2014	72,5	0,786842688	0,99317238

Рисунок 49 – Нормирование значений временного ряда

2. Вычисляем формулу выхода сети:

D4:=TANH(СУММ(TANH(СУММ(C5*$C$3;C6*$D$3;C7*$E$3))*$L$3;TANH(СУММ(C8*$F$3;C9*$G$3;C10*$H$3))*$M$3;TANH(СУММ(C11*$I$3;C12*$J$3;C13*

На рисунке 50 представлены выходы сети.



=TANH(СУММ(TANH(СУММ(C5*$C$3;C6*$D$3;C7*

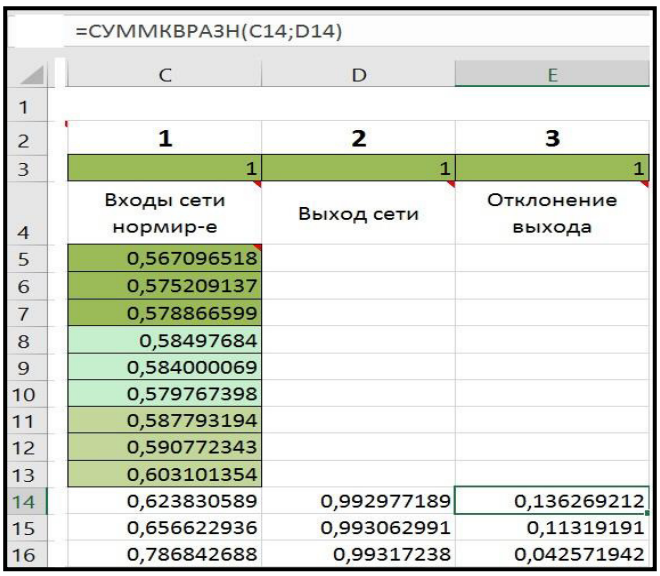
	A	B	C
1			
2	1	2	3
3	1	1	1
4	Входы сети нормир-е	Выход сети	Отклонение выхода
5	0,567096518		
6	0,575209137		
7	0,578866599		
8	0,58497684		
9	0,584000069		
10	0,579767398		
11	0,587793194		
12	0,590772343		
13	0,603101354		
14	0,623830589	0,992977189	0,136269212
15	0,656622936	0,993062991	0,11319191
16	0,786842688	0,99317238	0,042571942

Как выходы сети будем использовать девять последних значений прогнозируемой величины

Рисунок 50 – Выходы нейронной сети

3. Вычисляем отклонение выхода как квадрат разницы выхода нейрона и реального значения (рисунок 51).

E14: =СУММКВРАЗН(C14;D14)

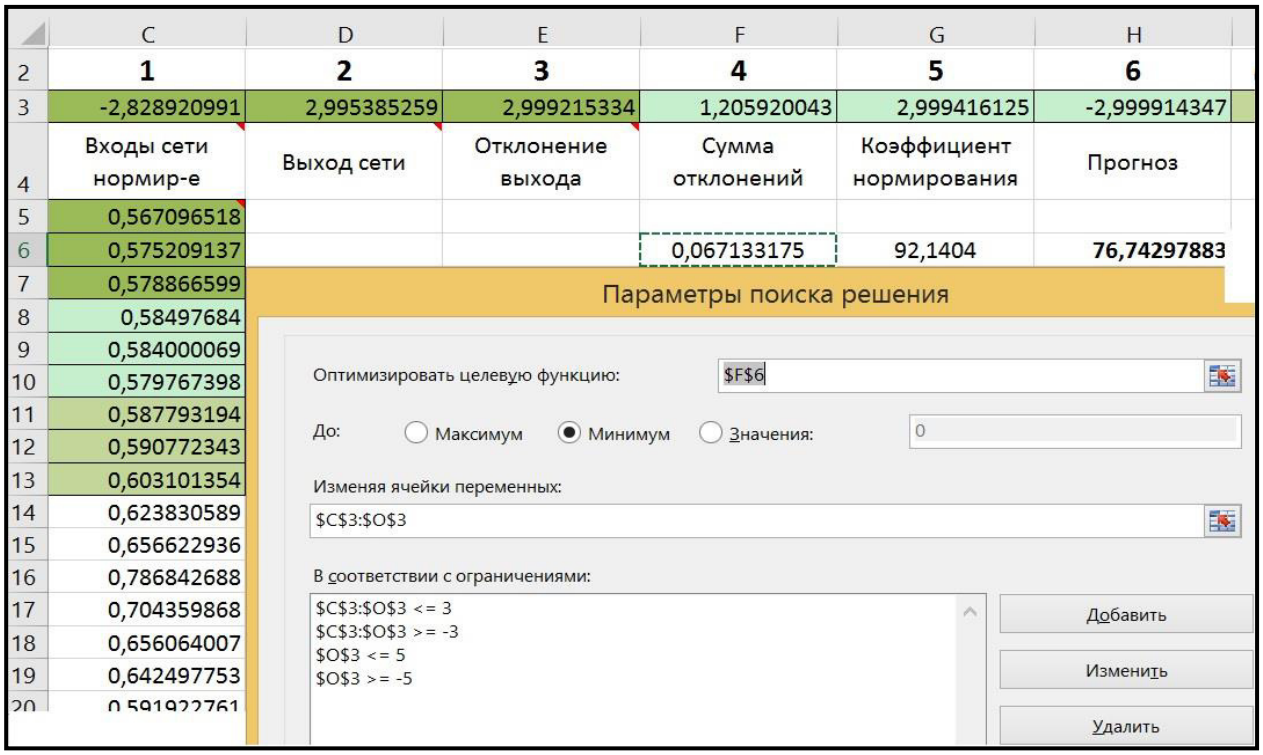


=СУММКВРАЗН(C14;D14)

	C	D	E
1			
2	1	2	3
3	1	1	1
4	Входы сети нормир-е	Выход сети	Отклонение выхода
5	0,567096518		
6	0,575209137		
7	0,578866599		
8	0,58497684		
9	0,584000069		
10	0,579767398		
11	0,587793194		
12	0,590772343		
13	0,603101354		
14	0,623830589	0,992977189	0,136269212
15	0,656622936	0,993062991	0,11319191
16	0,786842688	0,99317238	0,042571942

Рисунок 51 – Расчет отклонения выхода сети

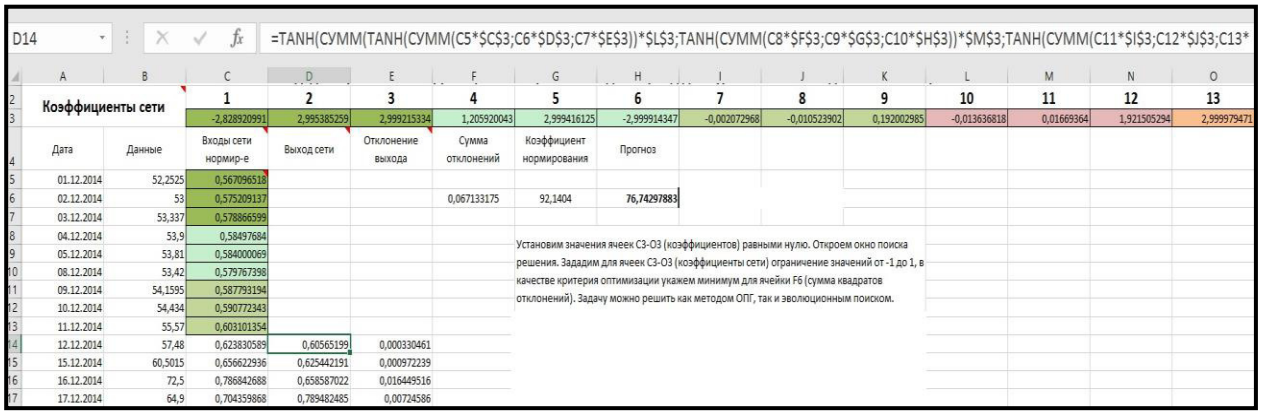
4. Установим значения ячеек C3-O3 (коэффициентов) равными нулю. Откроем окно поиска решения. Зададим для ячеек C3-O3 (коэффициенты сети) ограничение значений от -1 до 1, в качестве критерия оптимизации укажем минимум для ячейки F6 (сумма квадратов отклонений). Задачу можно решить как методом Обобщенного приведенного градиента, так и эволюционным поиском (рисунок 52).



	C	D	E	F	G	H
2	1	2	3	4	5	6
3	-2,828920991	2,995385259	2,999215334	1,205920043	2,999416125	-2,999914347
4	Входы сети нормир-е	Выход сети	Отклонение выхода	Сумма отклонений	Коэффициент нормирования	Прогноз
5	0,567096518					
6	0,575209137			0,067133175	92,1404	76,74297883
7	0,578866599					
8	0,58497684					
9	0,584000069					
10	0,579767398					
11	0,587793194					
12	0,590772343					
13	0,603101354					
14	0,623830589					
15	0,656622936					
16	0,786842688					
17	0,704359868					
18	0,656064007					
19	0,642497753					
20	0,591922761					

Рисунок 52 – Параметры поиска решения для подбора коэффициентов нейронной сети

5. Получаем значение прогноза на основе нейронной сети (рисунок 53).



D14: =TANH(СУММ(TANH(СУММ(C5*$C$3;C6*$D$3;C7*$E$3))*$L$3;TANH(СУММ(C8*$F$3;C9*$G$3;C10*$H$3))*$M$3;TANH(СУММ(C11*$I$3;C12*$J$3;C13*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
2	Коэффициенты сети		1	2	3	4	5	6	7	8	9	10	11	12	13
3			-2,828920991	2,995385259	2,999215334	1,205920043	2,999416125	-2,999914347	-0,002072968	-0,010523902	0,192002985	-0,013636818	0,01669364	1,921505294	2,999979471
4	Дата	Данные	Входы сети нормир-е	Выход сети	Отклонение выхода	Сумма отклонений	Коэффициент нормирования	Прогноз							
5	01.12.2014	52,2525	0,567096518												
6	02.12.2014	53	0,575209137			0,067133175	92,1404	76,74297883							
7	03.12.2014	53,337	0,578866599												
8	04.12.2014	53,9	0,58497684												
9	05.12.2014	53,81	0,584000069												
10	08.12.2014	53,42	0,579767398												
11	09.12.2014	54,1595	0,587793194												
12	10.12.2014	54,434	0,590772343												
13	11.12.2014	55,57	0,603101354												
14	12.12.2014	57,48	0,623830589	0,60565199	0,000330461										
15	15.12.2014	60,5015	0,656622936	0,625442191	0,000972239										
16	16.12.2014	72,5	0,786842688	0,658587022	0,016449516										
17	17.12.2014	64,9	0,704359868	0,789482485	0,00724586										

Установим значения ячеек C3-O3 (коэффициентов) равными нулю. Откроем окно поиска решения. Зададим для ячеек C3-O3 (коэффициенты сети) ограничение значений от -1 до 1, в качестве критерия оптимизации укажем минимум для ячейки F6 (сумма квадратов отклонений). Задачу можно решить как методом ОПГ, так и эволюционным поиском.

Рисунок 53 – Прогноз

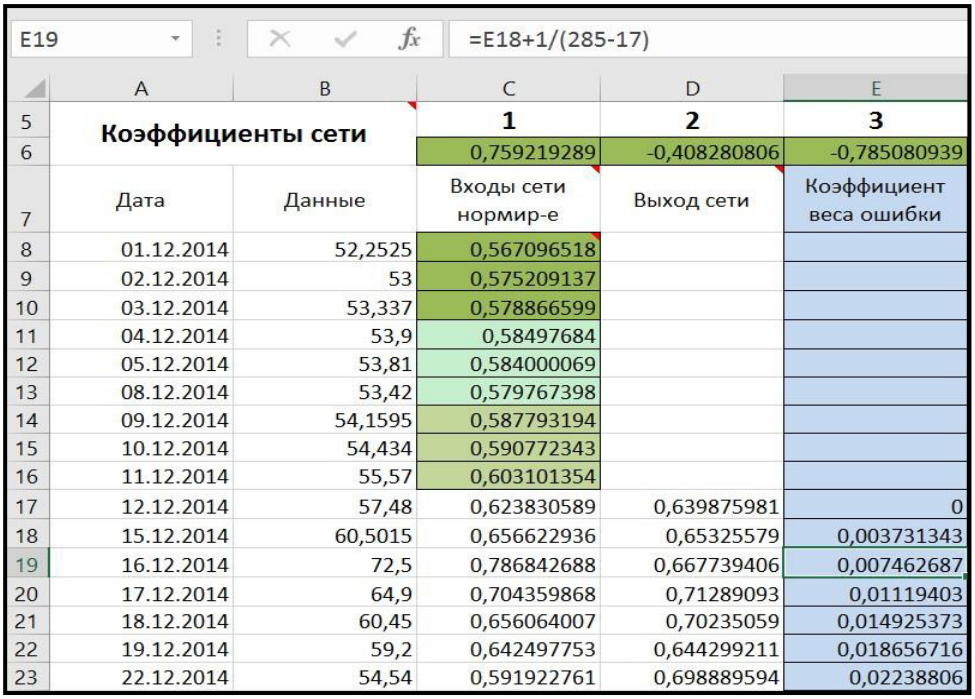
Пример № 2

Возьмём однослойную нейронную сеть, рассмотренную в предыдущем примере. Улучшим качество её прогноза, для текущего периода.

Решение:

1. Введём коэффициент веса ошибки, который будет линейно изменяться от нуля, до единицы (рисунок 54).

E18: =E17+1/(285-17)



E19: =E18+1/(285-17)

	A	B	C	D	E
5	Коэффициенты сети		1	2	3
6			0,759219289	-0,408280806	-0,785080939
7	Дата	Данные	Входы сети нормир-е	Выход сети	Коэффициент веса ошибки
8	01.12.2014	52,2525	0,567096518		
9	02.12.2014	53	0,575209137		
10	03.12.2014	53,337	0,578866599		
11	04.12.2014	53,9	0,58497684		
12	05.12.2014	53,81	0,584000069		
13	08.12.2014	53,42	0,579767398		
14	09.12.2014	54,1595	0,587793194		
15	10.12.2014	54,434	0,590772343		
16	11.12.2014	55,57	0,603101354		
17	12.12.2014	57,48	0,623830589	0,639875981	0
18	15.12.2014	60,5015	0,656622936	0,65325579	0,003731343
19	16.12.2014	72,5	0,786842688	0,667739406	0,007462687
20	17.12.2014	64,9	0,704359868	0,71289093	0,01119403
21	18.12.2014	60,45	0,656064007	0,70235059	0,014925373
22	19.12.2014	59,2	0,642497753	0,644299211	0,018656716
23	22.12.2014	54,54	0,591922761	0,698889594	0,02238806

Рисунок 54 – Коэффициент веса ошибки

2. Умножим отклонение выхода на этот коэффициент. Таким образом оптимизационный критерий отклонения станет гораздо сильнее зависеть от самых последних (текущих) данных и меньше зависеть от старых данных, когда рыночные условия были иными. Проведём поиск решения для коэффициентов сети и получим прогноз на один период вперед (рисунок 55-56).

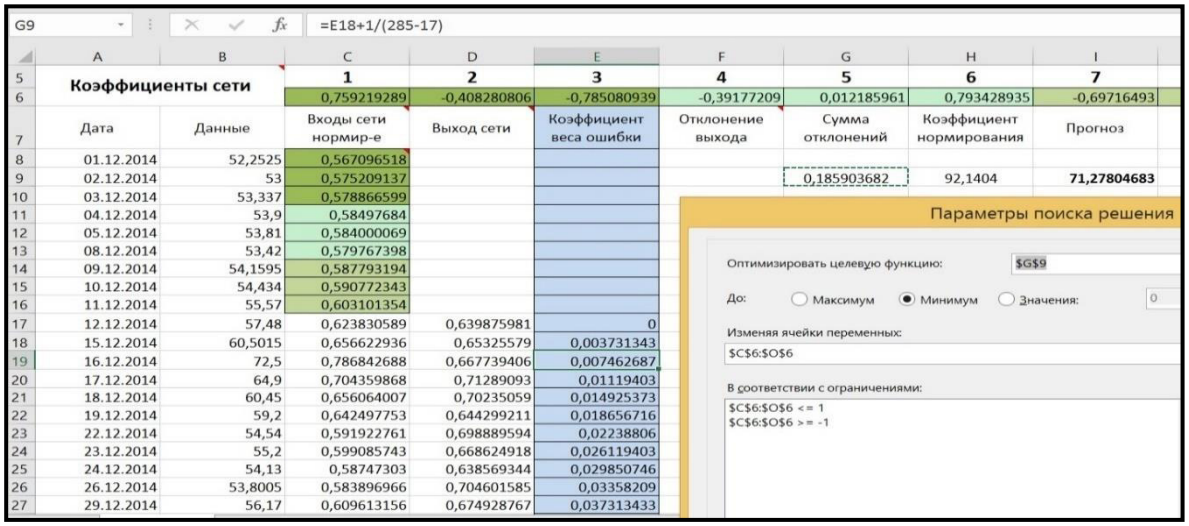


Рисунок 55 – Параметры поиска решения

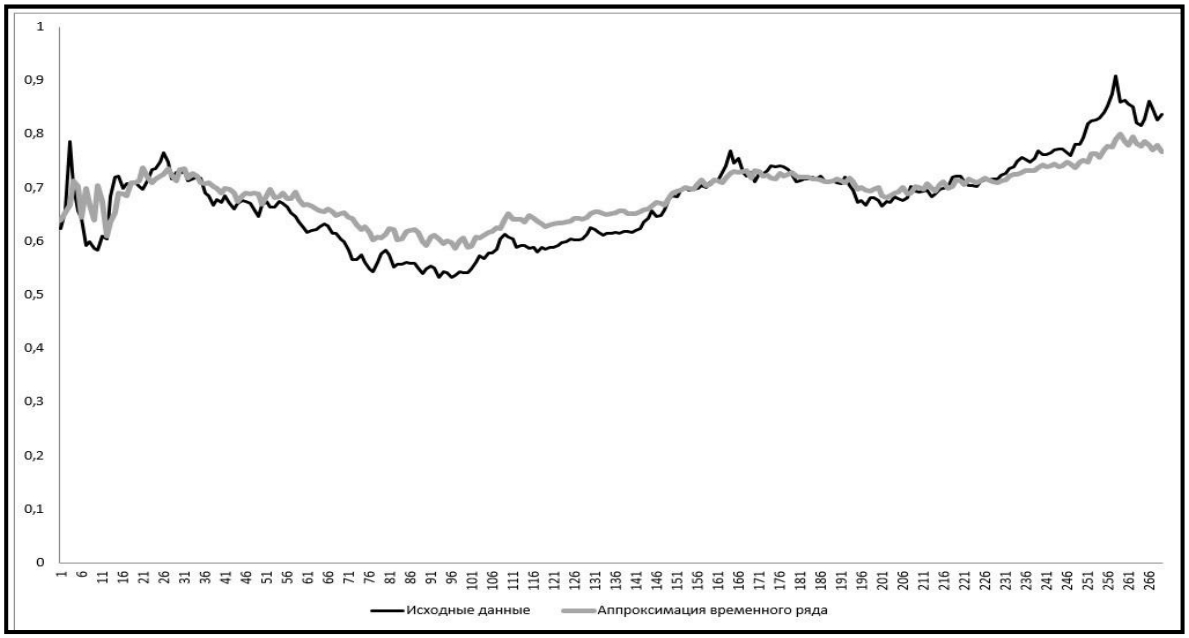


Рисунок 56 – Прогноз временного ряда USD/RUB на один период вперед

Пример № 3

Необходимо построить многофакторный прогноз на основе однослойной нейронной сети.

Решение:

Предположим, что курс доллара зависит не только от собственных тенденций, но и от цен на нефть и золото. Возьмём однослойную нейронную сеть рассмотренную в предыдущем примере. Подадим на входы различных нейронов - различные данные за предшествующие три дня (1-й нейрон: стоимость нефти BRENT; 2-нейрон: стоимость унции золота; 3-й нейрон: курс USD/RUB(TOD). Введём коэффициент веса ошибки (рисунок 57).

H17 =TANH(СУММ(TANH(СУММ(C14*$G$6;C15*$H$6;C16*$I$6))*$P$6;TANH(СУММ(E14*$J$6;E

	A	B	C	D	E	F	G	H	I	J	K
5	Коэффициенты сети						1	2	3	4	5
6							-0,11690481	-0,217226153	-0,980521966	0,925403844	0,149867598
7	Дата	Нефть	Входы сети нефть	Золото	Входы сети золото	Доллар	Входы сети доллар	Выход сети	Коэффициент веса ошибки	Отклонение выхода	Сумма отклонений
8	01.12.2014	53,3	0,70902174	1170	0,86898396	52,2525	0,567096518				
9	02.12.2014	56,27	0,74853008	1181,7	0,8776738	53	0,575209137				0,151155428
10	03.12.2014	54,46	0,7244526	1182,3	0,87811943	53,337	0,578866599				
11	04.12.2014	55,22	0,73456248	1185,8	0,88071895	53,9	0,58497684				
12	05.12.2014	54,7	0,7276452	1188,5	0,8827243	53,81	0,584000069				
13	08.12.2014	55,89	0,74347514	1198,9	0,8904486	53,42	0,579767398				
14	09.12.2014	55,31	0,7357597	1201,9	0,89267677	54,1595	0,587793194				
15	10.12.2014	56,38	0,74999335	1198,3	0,89000297	54,434	0,590772343				
16	11.12.2014	58,86	0,78298348	1193,1	0,88614082	55,57	0,603101354				
17	12.12.2014	56,12	0,74653471	1181,7	0,8776738	57,48	0,623830589	0,618325864	0	0	
18	15.12.2014	55,94	0,74414026	1187,1	0,88168449	60,5015	0,656622936	0,630887201	0,003731343	2,47137E-06	
19	16.12.2014	56,56	0,75238779	1203,3	0,89371658	72,5	0,786842688	0,642824134	0,007462687	0,000154786	
20	17.12.2014	55,1	0,73296619	1202,5	0,8931224	64,9	0,704359868	0,673612372	0,01119403	1,05829E-05	
21	18.12.2014	56,67	0,75385107	1200,9	0,89193405	60,45	0,656064007	0,670427779	0,014925373	3,07937E-06	
22	19.12.2014	56,23	0,74799798	1217,1	0,90396613	59,2	0,642497753	0,666319702	0,018656716	1,05874E-05	
23	22.12.2014	56,47	0,75119057	1213,5	0,90129234	54,54	0,591922761	0,654400737	0,02238806	8,73917E-05	
24	23.12.2014	59,01	0,78497885	1210,7	0,89921272	55,2	0,599085743	0,634888601	0,026119403	3,3481E-05	
25	24.12.2014	59,66	0,79362546	1196,6	0,88874034	54,13	0,58747303	0,625780776	0,029850746	4,38055E-05	
26	26.12.2014	57,29	0,7620986	1196,6	0,88874034	53,8005	0,583896966	0,611714922	0,03358209	2,59871E-05	

Рисунок 57 – Данные для многофакторного прогноза

Выполним поиск решения как для предыдущего прогноза (рисунок 58).

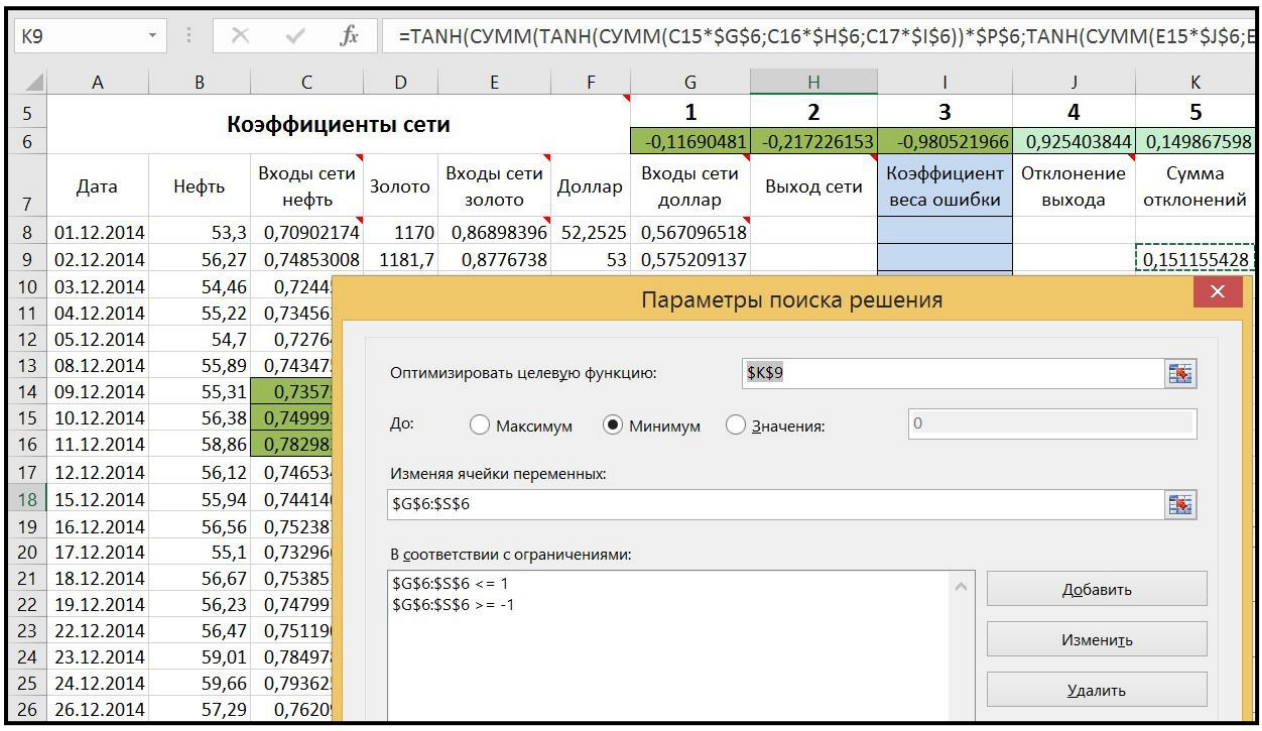


Рисунок 58 – Поиск подбора коэффициентов сети

Получаем прогноз на основе многофакторного прогнозирования (рисунок 59).

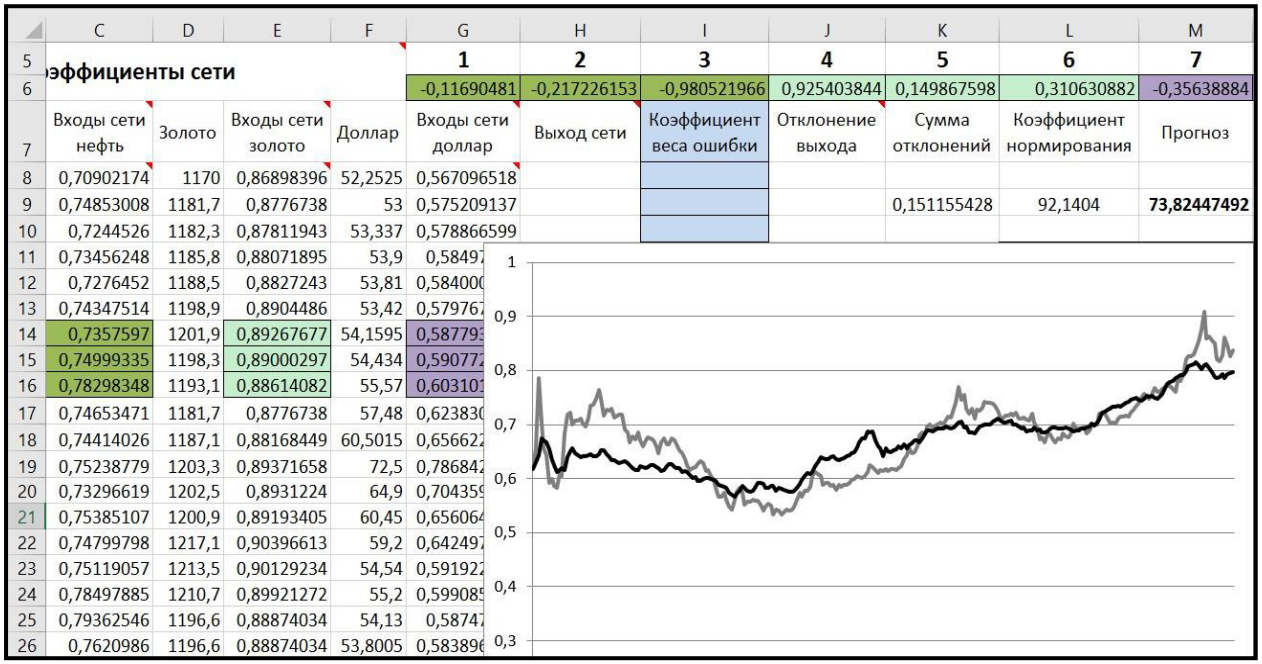


Рисунок 59 – Многофакторный прогноз

Пример № 4

Необходимо построить многослойную нейронную сеть.

Решение:

Возьмём однослойную нейронную сеть, рассмотренную в предыдущем примере. Добавим к ней второй слой. При этом выходы первого слоя будут входами второго, а выходы второго слоя - входами выходного нейрона. Такая сеть обучается горазд медленнее, однако дает более точный прогноз. Выполним поиск решения, как и для предыдущего прогноза (рисунок 60 – 61).

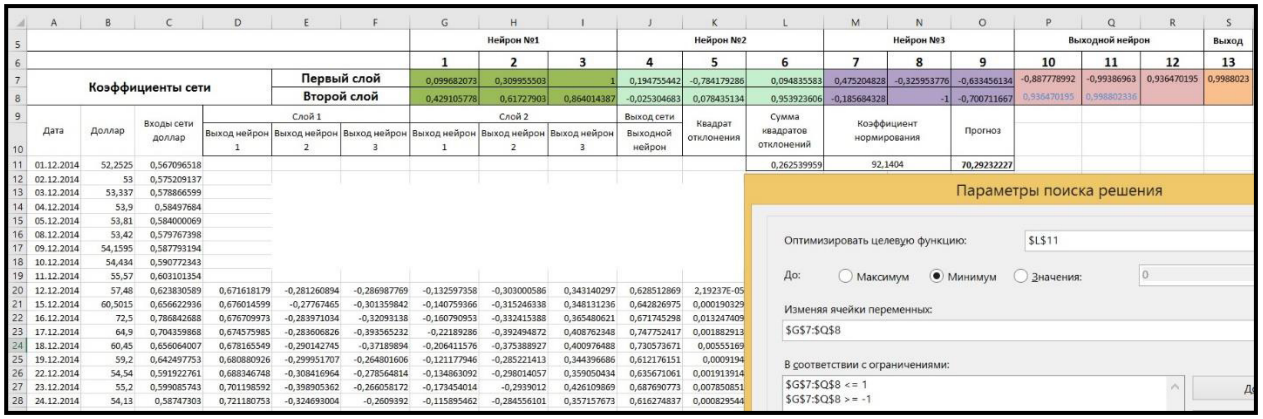


Рисунок 60 – Параметры поиска решения

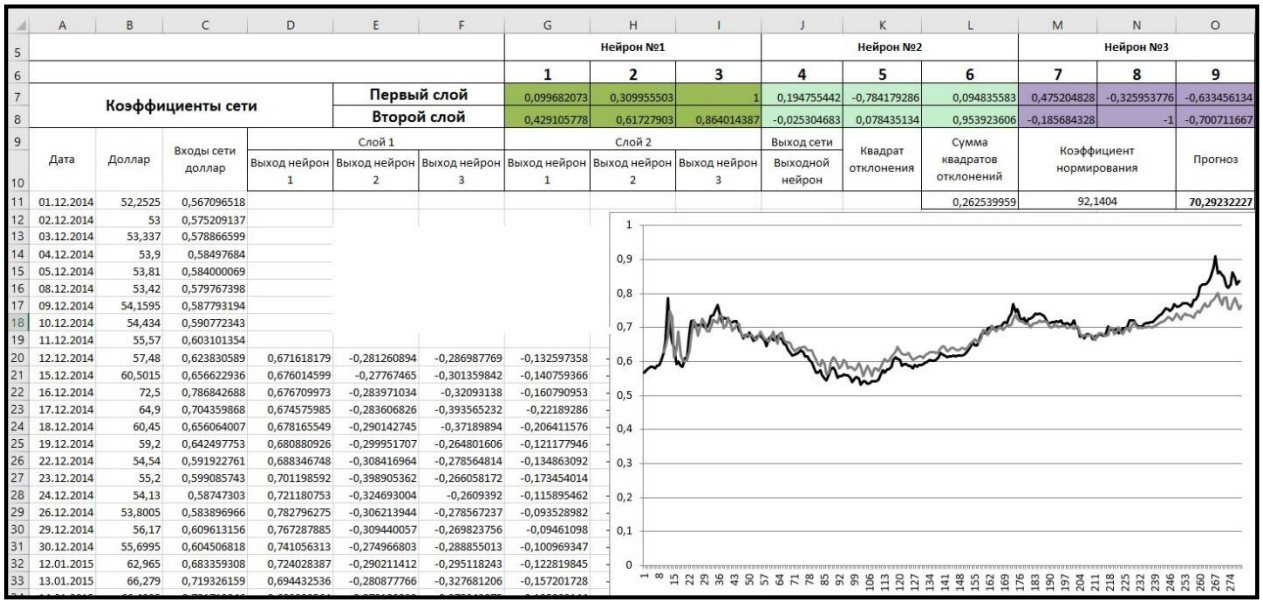


Рисунок 61 – Прогноз

Пример №5

Применение нейронных сетей в ансамблевом прогнозировании

Поэтапно опишем метод создания ансамблевого (совокупного) прогноза согласно формулам (41-43). Совокупный прогноз - это среднее значение

результатов нескольких методов регрессионного прогнозирования, полученное на основе коэффициентов достоверности рассматриваемых методов.

$$F(\tau) = \frac{\sum_{i=1}^{i=n} k_{\mathrm{d}i} f_i(\{\tau,\ x(\tau), \dots\})}{n}, \tag{41}$$

Рассчитаем коэффициент доверия каждого прогноза, как величину, обратную доле вносимой ошибки.

$$k_{\mathrm{d}i} = \frac{1}{RMSE_i} \tag{42}$$

Предложим ансамблевый (совокупный) метод прогнозирования состоящий из трех частных прогнозов. Рассмотрим каждый из трех методов прогнозирования.

1. Линейный метод прогнозирования

$$y = a \cdot x + b \tag{43}$$

2. ETS прогноз. Метод экспоненциального сглаживания.

Методики составления прогнозов, в которых присутствует экспоненциальное сглаживание, являются во многом схожими друг с другом. В них прогнозирование – это взвешенная сумма прошлых наблюдений. При этом в модели применяется экспоненциально уменьшающийся вес для результатов предшествующих наблюдений.

3. Многофакторный метод прогнозирования на основе нейронных сетей

Прогноз нейронной сети с дополнительными входами рассчитывается согласно формуле (44):

$$Y(x_\tau) = k_0 \zeta\, k_1 \zeta\left(\sum_{i=1}^{m} k_i \zeta\left(\sum_{j=i}^{m} k_{ij} \zeta\left(x_{\tau-j}\right)\right)\right) + + k_2 \zeta\left(\sum_{i=1}^{m} k_i \zeta\left(\sum_{j=i}^{m} k_{ij} \zeta\left(v_{\tau-j}\right)\right)\right) \tag{44}$$

где $Y(x_\tau)$ –функция прогнозирования;

$k_{0,1,2,i,j,ij}$ – вычисляемые коэффициенты;

$\zeta(x) = \frac{x}{|x|+\alpha}$ - рациональная сигмоида;

i – номер слоя нейронной сети;

j – номер входа нейронной сети;

m – ширина входа нейронной сети;

τ – временной промежуток первого прогнозируемого значения Y;

x_i – значения основных входов сети;

v_i – значения дополнительных входов.

Данную сеть будем использовать для формирования прогнозов, учитывающих внешние факторы и сторонние исторические данные.

В качестве примера сформируем совокупный прогноз курса рубля к XDR. Для создания совокупного (ансамблевого) прогноза на 30 дней вперёд выполним следующие действия:

1. Получим данные о ценах закрытия активов и рассчитаем их корреляции со стоимостью рубля с задержкой (лагом) в 30 дней.
2. Выберем три наиболее коррелирующих актива, за исключением валют входящих в XDR.
3. Сформируем для рубля простой линейный прогноз на 30 дней, проведём кросс-валидацию по 30-дневному периоду и рассчитаем её остатки.
4. Сформируем для рубля прогноз на 30 дней методом экспоненциального сглаживания (ETS прогноз), проведём кросс-валидацию по 30-дневному периоду и рассчитаем её остатки.
5. Сформируем для рубля прогноз на 30 дней простой нейронной сетью, проведём кросс-валидацию по 30-дневному периоду и рассчитаем её остатки.
6. Оценим стандартное отклонение кросс-валидации для каждого из прогнозов. Рассчитаем коэффициент доверия каждого прогноза, как величину, обратную доле вносимой ошибки.
7. Рассчитаем совокупный прогноз, как сумму произведений значений прогнозов на их коэффициенты доверия. Построим диаграмму прогноза.
8. Рассчитаем доверительные интервалы независимых прогнозов и общий доверительный интервал совокупного прогноза.
9. Рассчитаем коэффициенты доверия соответствующий совокупный прогноз.
10. Рассчитаем и сравним доверительные интервалы всех получившихся прогнозов.
11. Построим график для лучшего прогноза.

Согласно первому этапу, рассчитаем корреляции активов со стоимостью рубля с задержкой (лагом) в 30 дней.

Таблица 4 - Корреляция активов со стоимостью рубля.

Активы	**Корреляция**	**Корреляция по модулю**
Золото	-0,845325766	0,845325766
Amazon	-0,81260061	0,81260061
Билайн	0,799611854	0,799611854
Nvidia акции	-0,799233561	0,799233561
Yandex	-0,757686405	0,757686405
Microsoft	-0,756505163	0,756505163
Газпром акции	0,725986563	0,725986563
Blizzard	-0,718797483	0,718797483
eBay акции	-0,686196227	0,686196227
Alibaba	-0,671257266	0,671257266
Серебро	-0,664151623	0,664151623
Потребсектор	-0,646104957	0,646104957
МосБиржа	-0,64446564	0,64446564
Лукойл акции	0,63682404	0,63682404
Хайтек	-0,629660516	0,629660516
РосТелеком	-0,612671776	0,612671776
Facebook	-0,594712267	0,594712267
Google	-0,499092125	0,499092125
Нефть	0,41165769	0,41165769

Наиболее коррелирующими активами являются: *Золото, Amazon, Билайн.*

На следующем этапе сформируем для рубля однофакторный линейный прогноз на 30 дней. Сформируем для рубля прогноз на 30 дней методом экспоненциального сглаживания (ETS прогноз). Сформируем для рубля прогноз на 30 дней многофакторной нейронной сетью. Приведем пример прогноза на основе нейронной сети (рисунок 62). Данный прогноз является многофакторным и построен на основе формулы (10). То есть построена нейронная сеть с тремя дополнительными входами (факторами), так как на предыдущем этапе было выявлено, что на курс рубля влияют котировки Золота, Amazon и Билайн.

Рисунок 62 – Прогноз курса рубля к XDR

На следующем этапе оценим стандартное отклонение кросс-валидации для каждого из прогнозов. Рассчитаем коэффициент доверия каждого прогноза, как величину, обратную доле вносимой ошибки.

Таблица 5 - Остатки кросс-валидации.

Остатки кросс-валидации		
Линейный прогноз	**ETS прогноз**	**Нейронный прогноз**
0,073264492	-0,03839909	0,003491876
0,087876408	-0,00046781	0,01984938
0,084102209	-0,01000248	0,012780778
0,077525906	-0,01130012	0,004064768
0,078127993	-0,00018638	0,005695219
0,078730079	-0,01128149	0,003571314
0,079332166	-0,00865256	0,004164595
0,088817593	-0,0055173	0,011898193
0,090899767	0,007034984	0,011151271
0,098863493	0,005410426	0,018368741
0,096211856	-0,00055176	0,015966971
0,096813942	-0,02401246	0,016628055
0,097416029	-0,02483565	0,013838003
0,103868211	-0,02083383	0,019193483
0,107831486	-0,01885148	0,023295661
0,105972778	-0,0094012	0,021516002
0,100837689	-0,0198028	0,016333975
0,093572087	-0,0484201	0,008896109
0,094174174	-0,00335799	0,008330294
0,09477626	-0,06151747	0,007086374

0,090708532	-0,03101229	0,002843581
0,081174518	-0,052064	-0,007365482
0,073790353	-0,06571372	-0,014794436
0,07601154	-0,06569578	-0,01179671
0,083093749	-0,05385327	-0,004982243
0,083695835	-0,06288757	-0,006456914
0,084297922	-0,04802539	-0,004543413
0,084832697	-0,04408291	-0,004360899
0,086945561	-0,06711614	-0,002670908
0,087547648	-0,0667488	-0,003536446

Таблица 6 - Расчет коэффициента доверия

RMSE		
Линейный прогноз	**ETS прогноз**	**Нейронный прогноз**
0,090727432	0,038501713	0,012262369
Обратная величина ошибки RMSE		
11,02202478	25,97287028	81,55030989
Коэффициент доверия		
0,092977399	0,21909676	0,687925842

Рассчитаем совокупный прогноз, как сумму произведений значений прогнозов на их коэффициенты доверия. Построим диаграмму прогноза.

Таблица 7 - Данные по частным и совокупному прогнозу

Частные прогнозы			
Дата	**Линейный прогноз**	**ETS прогноз**	**Нейронный прогноз**
02.04.2021	0,847484326	1,006068029	0,93410289
03.04.2021	0,84689581	1,003899018	0,933898218
04.04.2021	0,846307294	0,997159095	0,934381959
05.04.2021	0,845718778	0,985342124	0,933382088
06.04.2021	0,845130262	0,998221535	0,934604175
07.04.2021	0,844541746	1,034475197	0,934002459
08.04.2021	0,84395323	1,045696776	0,933859156
09.04.2021	0,843364714	1,050910013	0,933671675
10.04.2021	0,842776198	1,038506595	0,934239504
11.04.2021	0,842187682	1,046948763	0,935574148
12.04.2021	0,841599166	1,014073502	0,936693783
13.04.2021	0,84101065	1,003153784	0,936196431
14.04.2021	0,840422134	1,002794811	0,937668547
15.04.2021	0,839833618	1,037548929	0,938019926
16.04.2021	0,839245102	1,021609142	0,937071981
17.04.2021	0,838656586	1,02421972	0,93702568
18.04.2021	0,838068069	1,004880828	0,936075229

Совокупный прогноз	
Дата	**Ансамблевый (Совокупный) прогноз**
02.04.2021	0,94181665
03.04.2021	0,941145909
04.04.2021	0,939947273
05.04.2021	0,936615657
06.04.2021	0,940223481
07.04.2021	0,947697886
08.04.2021	0,950003197
09.04.2021	0,950961709
10.04.2021	0,948580065
11.04.2021	0,951293135
12.04.2021	0,944805779
13.04.2021	0,942016444
14.04.2021	0,942895782
15.04.2021	0,950697301
16.04.2021	0,94649811
17.04.2021	0,946983509
18.04.2021	0,942037862

19.04.2021	0,837479553	0,985175951	0,934508994	19.04.2021	0,936588415
20.04.2021	0,836891037	0,982986095	0,934370737	20.04.2021	0,935958795
21.04.2021	0,836302521	0,969899783	0,934366987	21.04.2021	0,933034329
22.04.2021	0,835714005	0,987844482	0,933940673	22.04.2021	0,936617963
23.04.2021	0,835125489	0,983736218	0,932451398	23.04.2021	0,934638626
24.04.2021	0,834536973	1,018154476	0,932070737	24.04.2021	0,94186297
25.04.2021	0,833948457	1,014979647	0,933297521	25.04.2021	0,941956593
26.04.2021	0,833359941	0,966984713	0,932834516	26.04.2021	0,931067826
27.04.2021	0,832771425	0,956414606	0,930997233	27.04.2021	0,927433317
28.04.2021	0,832182909	0,967592753	0,930705203	28.04.2021	0,929626799
29.04.2021	0,831594393	0,928929571	0,930538377	29.04.2021	0,920986338
30.04.2021	0,831005877	0,979753523	0,931006677	30.04.2021	0,932389139
01.05.2021	0,830417361	0,992261188	0,932249335	01.05.2021	0,935929665

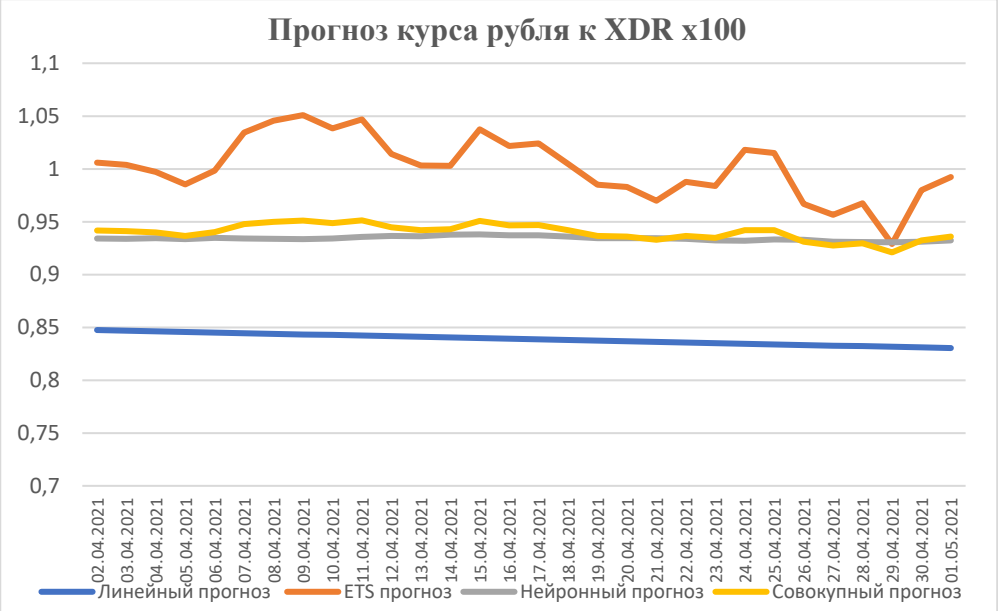


Рисунок 63 - Прогноз курса рубля к XDR

На следующем этапе рассчитаем доверительные интервалы независимых прогнозов и общий доверительный интервал совокупного прогноза.

Опорным значением доверительного интервала служит математическое ожидание, от которого в обе стороны может отклоняться скаляр доверительного интервала. По этой причине доверительным интервалом называют, как скалярную величину, так и границы, образуемые ей, относительно опорной точечной оценки.

$\mu - \varepsilon \leq \mu \leq \mu + \varepsilon -$ интервал с опорной величиной

Для z-распределения доверительный интервал равен

$$\mathbb{P}\left(\overline{X}-z_{1-\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}} \le \mu \le \overline{X}+z_{1-\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}}\right) = 1-\alpha \quad (45)$$

Для t-распределения доверительный интервал равен

$$\frac{s}{\sqrt{n}} \le \mu \le \overline{X}+t_{1-\frac{\alpha}{2},n-1}\frac{s}{\sqrt{n}}\Big) = 1-\alpha \quad (46)$$

При построении прогностического доверительного интервала для временного ряда, в качестве основной принимается гипотеза, что за каждый временной период интервал расширяется в обе стороны на точечную оценку интервала для одного периода, таким образом, границы доверительного интервала соответствуют приведённой формуле $\varepsilon_{\tau+n} = \varepsilon_{\tau}(n+1)$

Таблица 8 - Общий доверительный интервал

Уровень значимости	0,05
Уровень надёжности	0,95
Объём выборки	30
Z-значений	2,045229642
Доверительный интервал	0,012900444

На следующем этапе рассчитаем коэффициенты доверия и соответствующий совокупный прогноз. Рассчитаем и сравним доверительные интервалы всех получившихся прогнозов. Построим график для лучшего прогноза.

Таблица 9 - Доверительный интервал для всех частных прогнозов.

Доверительный интервал				
Тип прогноза				
Линейный прогноз	**ETS прогноз**	**Нейронный прогноз**	**Совокупный классический прогноз**	
Уточнённая дисперсия				
0,010663	0,001482	0,000150	0,000078	
Доверительный интервал				
0,038558	0,014377	0,004579	0,003300	
Накапливаемая за 30 дней ошибка прогноза				
115,67%	43,13%	13,74%	9,90%	

Таким образом, построен совокупный прогноз представленный на рисунке 64.

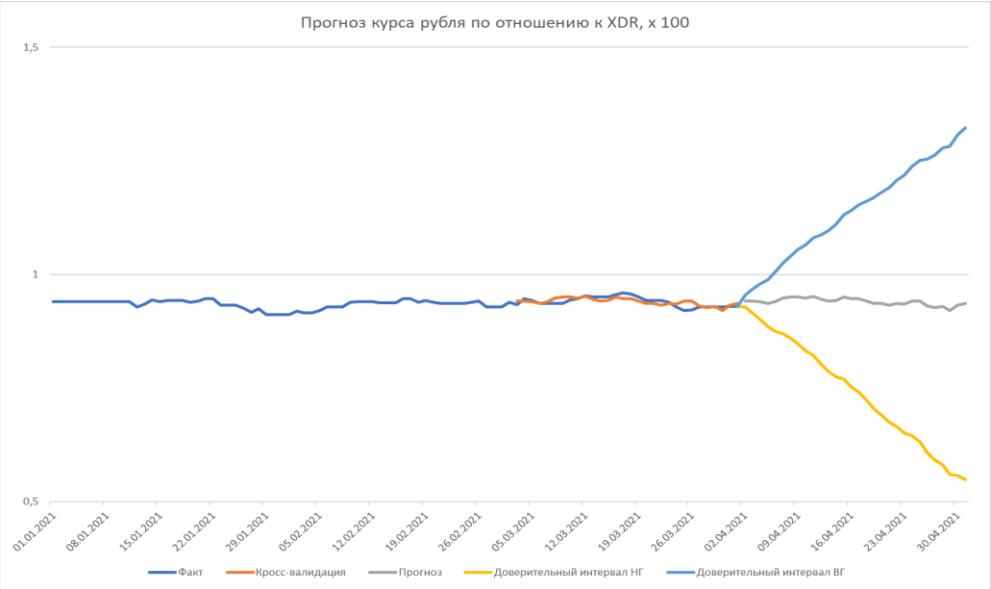


Рисунок 64 - Совокупный прогноз

4.3 Применение нейронных сетей в задачах классификации

Рассмотрим в качестве примера построение аналитической нейронной модели экономической эффективности для реально существующего предприятия.

В качестве исходных данных возьмём предприятие на экономическую эффективность которого предположительно оказывают влияние две группы факторов по два фактора в каждой. При этом предприятие является прибыльным и часть прибыли инвестирует в собственные основные средства и нематериальные, то есть обладает экономической инерцией. Для построения простой не рекуррентной свёрточной сети определим древовидную структуру взаимного влияния элементов эффективности для двухлетней глубины влияния (рисунок 65 - 66).

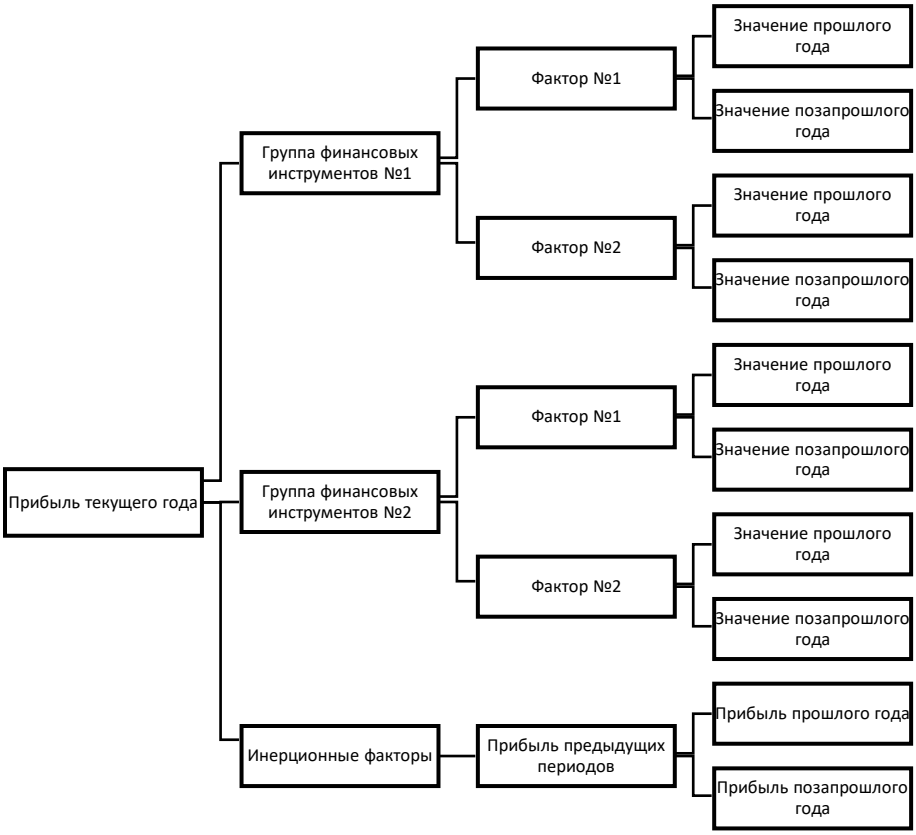


Рисунок 65 - Иерархия системы

Имея данную структуру определим архитектуру нейронной сети (рисунок 66).

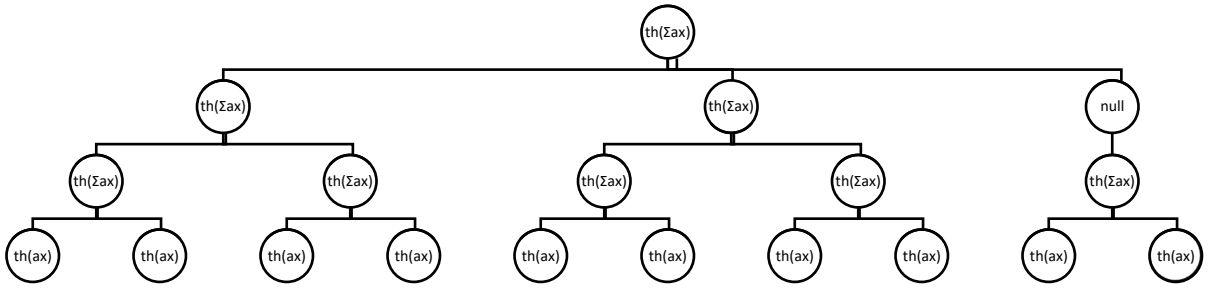


Рисунок 66 - Структура нейронной сети

Далее, имея полную структуру нейронной сети формализуем вычисление функции экономической эффективности, выраженной как прибыль текущего года:

$$E = a\sum_{g=1}^{g} a_g \,\mathrm{th}\sum_{s=1}^{s} a_s \,\mathrm{th}\sum_{n=1}^{n} \mathrm{th}\, a_n x, \text{ где} \quad (47)$$

E - выход нейронной сети (прибыль текущего года);

a – коэффициент при нейроне верхнего уровня;

a_g - коэффициенты при нейронах второго уровня;

a_s – коэффициенты при нейронах третьего уровня;

a_n – коэффициенты при нейронах последующих уровней;

th – функция активации нейронной сети th= $\frac{e^x - e^{-x}}{e^x + e^{-x}}$.

В качестве критерия оптимизации для данной сети выберем обобщённый показатель вида:

$$\text{Совокупная ошибка нейронной сети}(S) = MSE * \sum_{x=x_1}^{x_n} \left| \frac{dy_m}{dx} - \frac{dy_d}{dx} \right|,$$

где $\frac{dy_d}{dx}$ - производная для параметра выхода (d) нейронной сети;

$\frac{dy_m}{dx}$ - производная для параметра входа (m) нейронной сети;

MSE – среднеквадратическая ошибка нейронной сети в момент времени t;

Совокупная ошибка нейронной сети (S) – параметр оптимизации нейронной сети, который необходимо минимизировать.

Если разница производных $\frac{dy_m}{dx} - \frac{dy_d}{dx}$ близка к единице, то совокупная ошибка нейронной сети (S) будет стремиться к значению среднеквадратической ошибки (MSE). Следовательно, чем больше разница между производными входного и выходного параметра нейронной сети, тем меньше должен быть коэффициент влияния входного параметра сети на выходной параметр сети. И наоборот, чем меньше разница между производными входного и выходного параметра нейронной сети, тем больше должен быть коэффициент влияния входного параметра сети на выходной параметр сети.

При оптимизации *Совокупной ошибки нейронной сети* к минимуму, любым из общепринятых методов, получим обученную нейронную сеть, веса коэффициентов в которой будут распределены соответственно степени влияния факторов и их групп на конечную экономическую эффективность. Однако, следует иметь в виду, что малозначащие факторы могут иметь хорошую корреляцию с выходными данными и поэтому искусственно ограничить выходные коэффициенты нейронов a_s и a_g соответственно предполагаемой доле вклада факторов в конечный результат (обычно 3-5-кратным весом).

Разработка нейросетевой модели многофакторного анализа экономической эффективности на примере ГК «Росатом».

Разработаем алгоритм для построения нейросетевой модели оценки эффективности предприятия:

1. Формулировка задачи. Определяем входные и выходные элементы системы для построения модели финансовых перетоков как движение финансов от входных элементов к выходным.
2. Группировка элементов по факторам. Например, финансовые инструменты предприятия, оказывающие влияние на его экономическую эффективность можно разделить на две группы: финансовые инструменты, относящиеся к акционерным формам собственности АО и финансовые инструменты, относящиеся к государственным формам собственности ФГУП.
3. Анализ иерархии факторов, влияющих на экономическую эффективность.
4. Сбор первичной информации из открытых источников.
5. Создание нейронной сети по сгруппированному дереву. Разработка математической модели.
6. Определение оптимальных модельных коэффициентов.
7. Оценка и интерпретация значения коэффициентов нейронной сети.

1. Формулировка задачи.

Крупное предприятие, являющееся монопольной государственной корпорацией (ГК «Росатом») управляет более чем 300 крупными предприятиями, которые либо имеет в полной или долевой собственности как акционерные общества (АО), либо под непосредственным управлением, как ФГУПы(федеральное государственное унитарное предприятие)и ФГБУ (федеральное государственное бюджетное учреждение).

Задача: выяснить степень влияния финансовых инструментов и их групп на общую экономическую эффективность при помощи анализа не рекуррентной нейронной сетью и построить нейросетевую модель формирования прибыли.

Построим диаграмму иерархии влияния финансовых инструментов на прибыль предприятия (рис.67).

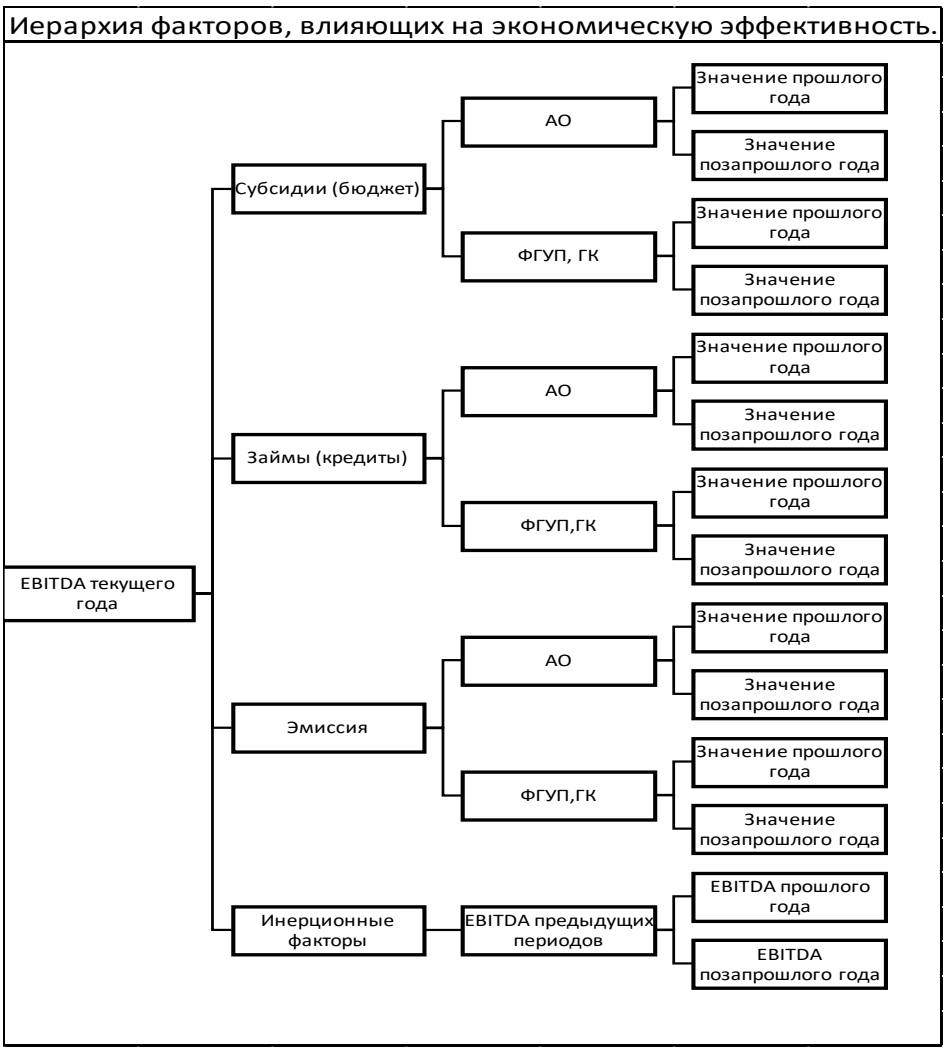
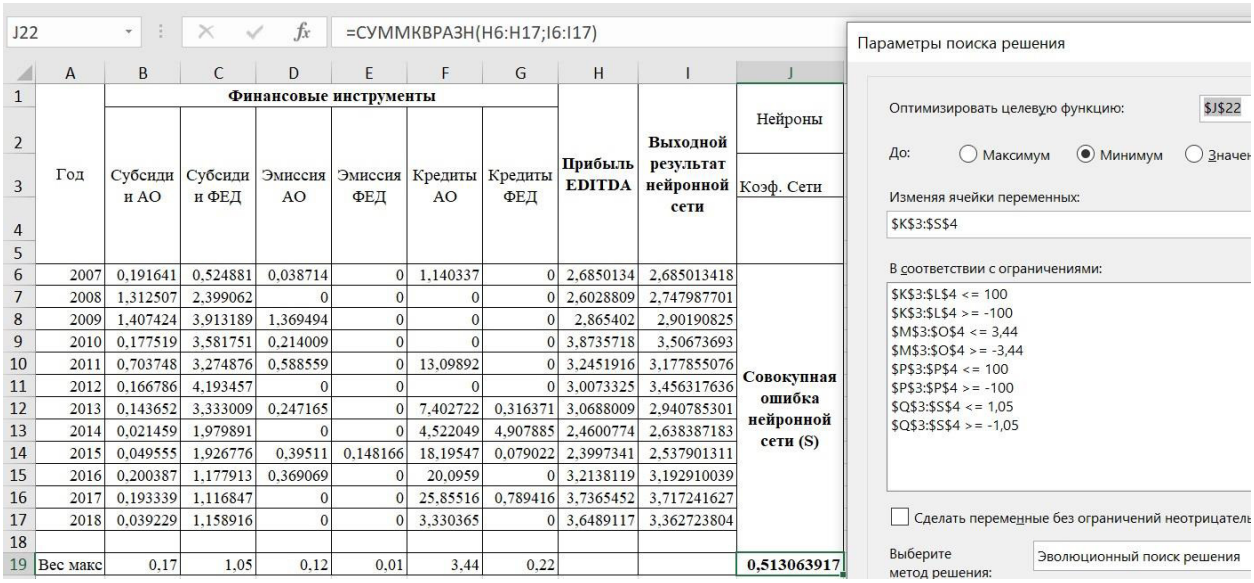


Рисунок 67 - Иерархия факторов

Поскольку групп видов собственности и подгрупп финансовых инструментов четко определено данная задача может быть решена посредством сверточной нейронной сети. При этом коэффициенты свертывающих нейронов будут соответствовать весам эффективности групп форм собственности и подгрупп финансовых инструментов (рисунок 68).



J22 =СУММКВРАЗН(H6:H17;I6:I17)

	A	B	C	D	E	F	G	H	I	J
1		Финансовые инструменты								Нейроны
2	Год	Субсидии АО	Субсидии ФЕД	Эмиссия АО	Эмиссия ФЕД	Кредиты АО	Кредиты ФЕД	Прибыль EDITDA	Выходной результат нейронной сети	
3										Коэф. Сети
4										
5										
6	2007	0,191641	0,524881	0,038714	0	1,140337	0	2,6850134	2,685013418	
7	2008	1,312507	2,399062	0	0	0	0	2,6028809	2,747987701	
8	2009	1,407424	3,913189	1,369494	0	0	0	2,865402	2,90190825	
9	2010	0,177519	3,581751	0,214009	0	0	0	3,8735718	3,50673693	
10	2011	0,703748	3,274876	0,588559	0	13,09892	0	3,2451916	3,177855076	
11	2012	0,166786	4,193457	0	0	0	0	3,0073325	3,456317636	Совокупная ошибка нейронной сети (S)
12	2013	0,143652	3,333009	0,247165	0	7,402722	0,316371	3,0688009	2,940785301	
13	2014	0,021459	1,979891	0	0	4,522049	4,907885	2,4600774	2,638387183	
14	2015	0,049555	1,926776	0,39511	0,148166	18,19547	0,079022	2,3997341	2,537901311	
15	2016	0,200387	1,177913	0,369069	0	20,0959	0	3,2138119	3,192910039	
16	2017	0,193339	1,116847	0	0	25,85516	0,789416	3,7365452	3,717241627	
17	2018	0,039229	1,158916	0	0	3,330365	0	3,6489117	3,362723804	
18										
19	Вес макс	0,17	1,05	0,12	0,01	3,44	0,22			0,513063917

Параметры поиска решения

Оптимизировать целевую функцию: $J$22

До: Максимум Минимум Значен

Изменяя ячейки переменных:

$K$3:$S$4

В соответствии с ограничениями:

$K$3:$L$4 <= 100
$K$3:$L$4 >= -100
$M$3:$O$4 <= 3,44
$M$3:$O$4 >= -3,44
$P$3:$P$4 <= 100
$P$3:$P$4 >= -100
$Q$3:$S$4 <= 1,05
$Q$3:$S$4 >= -1,05

Сделать переменные без ограничений неотрицатель

Выберите метод решения: Эволюционный поиск решения

Рисунок 68 - Подбор коэффициентов нейронной сети при минимизации совокупной ошибки сети (S)

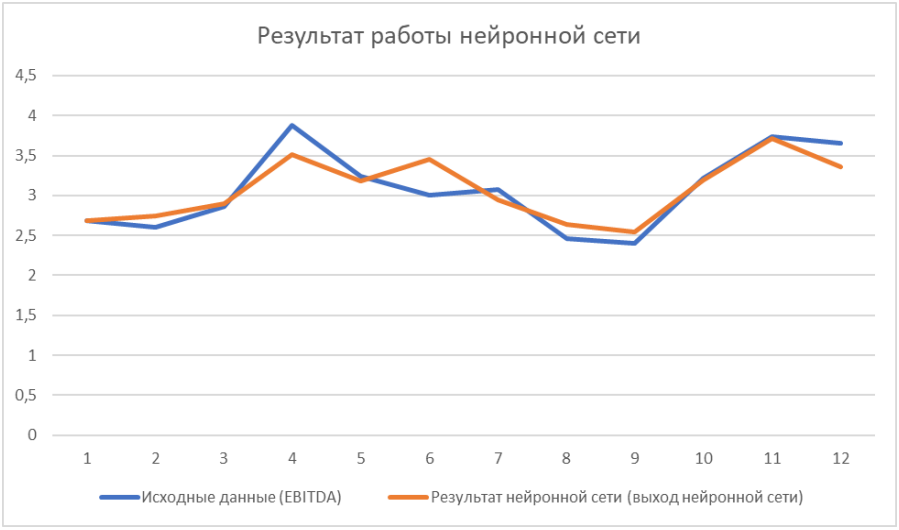


Рисунок 69 - Результат выхода нейронной сети

6. Определение оптимальных модельных коэффициентов.

В таблице 10 представлены результаты нейронной сети. Нейронной сетью осуществляется подбор коэффициентов, определяющих степень влияния финансовых инструментов на прибыль компании.

Таблица 10 - Расчет коэффициентов нейронной сети

Нейроны	Выходной нейрон	Вход АО	Акционерная форма собственности			Вход ФЕД	Федеральная форма собственности			
			Субсидии	Эмиссия	Кредиты		Субсидии	Эмиссия	Кредиты	
Коэф Сети	6,6356	1,3061	8,4051	0,6117	0,0009	0,6312	0,0156	41,1868	1,8231	Поза-прошлый год
	77,6946	55,8185	13,3938	0,0002	0,0007	20,5897	0,0000	0,0023	0,0000	прошлый год

7. Оценка и интерпретация значений коэффициентов нейронной сети.

Для интерпретации коэффициентов нейронной сети, выразим их в процентном отношении. В таблице 11 приводится оценка коэффициентов сети в процентах.

Таблица 11 Интерпретация коэффициентов нейронной сети

Нейроны	Выходной нейрон	Вход АО	Акционерная форма собственности			Вход ФЕД	Федеральная форма собственности			
			Субсидии	Эмиссия	Кредиты		Субсидии	Эмиссия	Кредиты	
КоэфСети	6,64	67%	93%	7%	0%	33%	0,04%	95,73%	4,24%	позапрошлый год (2017)
	77,6946	73%	100%	0,001%	0,005%	27%	0,00%	100,00%	0,00%	прошлый год (2018)

4.4 Распознавание эмоций в тексте на основе нейронных сетей

Одной из основных актуальных задач в развитии интерфейса человек-машина является распознавание человеческих эмоций. Полноценная реализация этого процесса гарантирует обладателю его аппаратной или программной реализации значительные экономические преимущества.

Сложность задачи заключается в том, что человеческие эмоции - это прежде всего продукт гуморальной регуляции организма, имеющей как крайне широкий диапазон значений, так и их комбинаций. Например, концентрация стрессовых гормонов в организме может в течение нескольких секунд изменяться более чем в 20 раз. Несмотря на наличие весьма чёткой формализации модели эмоций, применяемой в психологии, на сегодняшний день не существует автоматизированных систем распознавания человеческих эмоций с эффективностью более 90%.

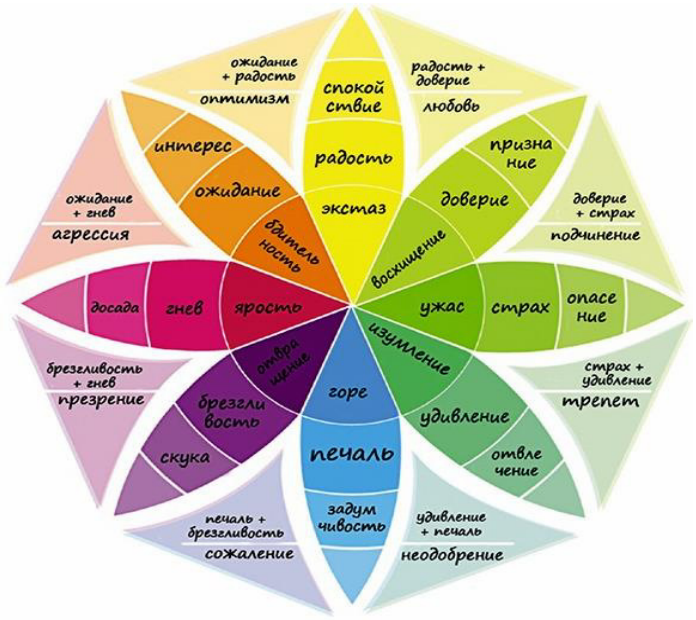


Рисунок 70 – Карта эмоций

Кажущаяся простота распознавания эмоций явно указывает на то, что это простейшая классификационная задача. Однако, несмотря на все усилия и вложенные средства современные голосовые ассистенты или чат-системы обладают лишь самыми зачатками способности восприятия эмоций и подчас совершают грубые, порой смешные ошибки.

Главной проблемой здесь является искусственное огрубление классической психологической эмоциональной модели до состояния простейшего набора базовых эмоций, не допускающего промежуточных или неопределенных состояний.

Типовые классификаторы множеств

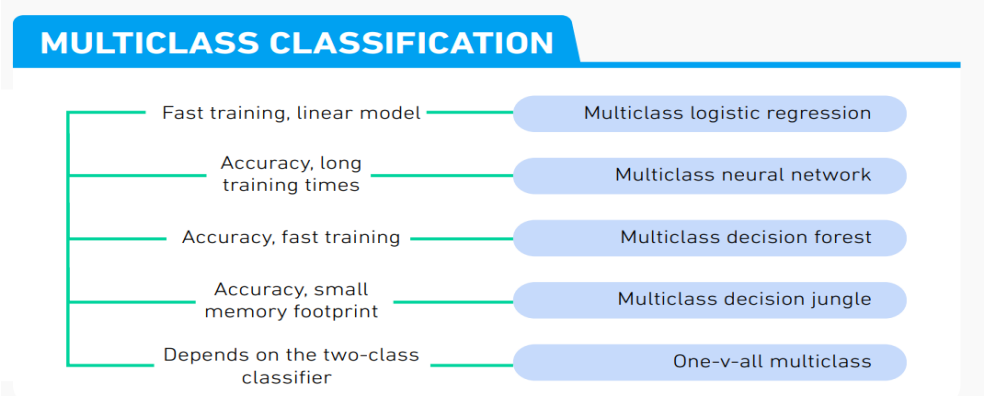


Рисунок 71 - Типовые классификаторы множеств

Mar 22, 2019, Stefan Kojouharov, Becoming Human: Artificial Intelligence Magazine

Типовое решение

Рисунок 72 – Алгоритм распознавания эмоций в тексте

Культурно-языковая специфика

Как нам известно, славянские языки обладают собственным уникальным культурно-эмоциональным контекстом, развившимся в языковой парадигме эмоционального социализма и соответствующим образом, исказившим эмоциональную окраску не только идиоматических и обобщающих выражений, но и отдельных слов и словосочетаний.

Так, многие слова фразы, синекдохи, ставшие традиционными, несмотря на отрицательную смысловую нагрузку имеют положительную эмоциональную коннотацию и наоборот.

Большие данные – большие ошибки

Важным фактором в создании автоматических систем классификации является глубокий анализ исходного пакета данных, используемых для обучения. Казалось бы, чем больше обучающая и контрольная выборки, тем точнее обучится сеть. Однако на практике мы видим совершенно обратный эффект. Чем больше данных мы используем для обучения, тем менее эффективной становится сеть. Это связано прежде всего с низким качеством необработанных обучающих данных. Так, например, корпус русских текстов RuTweetCorp содержит 94% эмоционально неоднозначных выражений, а также фраз на казахском и украинском языках, набранных кириллицей.

Осторожнее со стеммингом

Стемминг – это метод выделения смысловой части слова при помощи отбрасывания окончаний и приставок. Следует учитывать, что в естественном языке окончания и приставки формируют эмоциональную картину слова. Сравните, например:

Хитренькая лисичка – явно положительная эмоциональная коннотация. Эта же фраза после стемминга:

Хитр лис – эмоциональная коннотация отсутствует.

При решении задачи классификации эмоций остатки стемминга следует анализировать в первую очередь.

Синоним не всегда то, чем кажется

Тезауризация (замена слова которому классификатор не обучался на его синоним присутствовавший в обучающей выборке) также как и стемминг кажется весьма привлекательной с точки зрения оптимизации быстродействия и объёма памяти классификатора.

Не следует, однако забывать, что не только омонимы, но и многие другие слова и фразы имеют контекстно-зависимую эмоциональную окраску, совершенно иную, чем подставляемый синоним.

Сравните эмоциональную окраску фраз:

Иду домой, **муку** купил. - Иду домой, **страдание** купил.

Проблема утраты признаков

Типичный приём очистки и оптимизации входных текстовых данных выглядит следующим образом:

Рисунок 73 - Очистка и оптимизация входных текстовых данных

Не следует забывать, что данные операции являются:

- Огрубляющими;
- Необратимыми.

Для анализа эмоциональной составляющей необходимо иметь и оценивать не только обработанные оптимизатором - «быстрые» данные, но и их необработанный «медленный» - исходный вариант для уточнения сомнительного результата.

Чрезмерная оптимизация портит данные, превращая их в трудноразличимые усреднённые величины

Рисунок 74 – Эффект обобщённого облака при чрезмерной оптимизации эмоционально окрашенных фраз

Проблема естественного языка

В естественном русском языке существует большое количество устойчивых просторечных выражений противоположных содержащимся в них словах по смыслу, оксюморонов и синекдох. Следует помнить, что при эмоциональном анализе эти выражения необходимо обрабатывать, как исключения, либо отдельно учитывая их в ленивых классификаторах, либо подвергая тезауризации.

Например:

Ужасно красивое платье → Очень красивое платье.
Работа не волк → Работать лень

Ленивый классификатор

Ленивый классификатор, это по своей сути - простая база данных, оптимизированная по быстродействию и содержащая готовые ответы для наиболее часто повторяющихся входных комбинаций нейронной сети. Его

использование позволяет не только снизить общую вычислительную нагрузку системы, но и значительно увеличить общую скорость классификации.

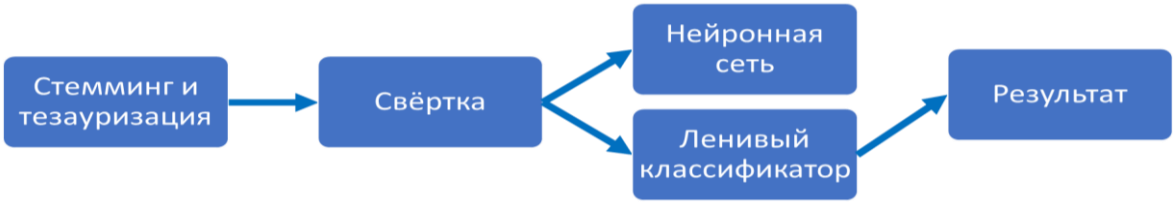


Рисунок 75 - Ленивый классификатор

Сверхленивый классификатор

В системах, позволяющих параллельную обработку вычислительных потоков, оптимальным для применения является сверхленивый классификатор, проверяющий наличие готовых ответов на всех этапах оптимизации входных данных.

Рисунок 76 - Сверхленивый классификатор

Предпочтительная архитектура

Рисунок 77 – Улучшенная архитектура для распознавания эмоций в тексте

SOTA-анализ нейронных классификаторов

Основные критерии (чем выше – тем хуже):

- Процент ошибок классификации;
- Время обучения классификатора;
- Время классификации элемента;
- Объем алгоритмической и динамической памяти;
- Сложность алгоритмов обучения и классификации.

Обеспечение повторяемости:

- Открытость алгоритма.
- Открытость исходного кода классификатора.
- Открытость архитектуры классифицирующей сети.
- Открытость методов обучения и контроля.
- Открытость обучающей и контрольной выборок.

SOTA по распознаванию эмоций в текстах: **https://paperswithcode.com**

Рассмотрим конкретный пример. При рассмотрении работы будем уделять основное внимание:

1. Набору данных.
2. Архитектуре решения.
3. SOTA-оценке

Эмоциональная модель. 1. Счастье. 2. Грусть. 3. Испуг. 4. Гнев

Архитектура сети

Рисунок 78 - Архитектура сети

Согласно интуитивному анализу, такой тип классификатора должен иметь очень высокую скорость, как обучения, так и классификации, но обладать крайне высоким процентом ошибок.

Анализ и предварительная обработка

Для обучения и контроля классификатора использовался корпус маркированных по эмоциональному окрасу коротких текстов **RuTweetCorps.**

При анализе обнаружены фразы на казахском и украинском языках, а также фразы с неявной эмоциональной коннотацией.

После удаления неявных элементов и иноязычных текстов корпус был сокращён с целью увеличения качества выборок.

334 836 фраз → **18 484** фраз

Обучение сети

- **15** эпох по
- **17 610 000** циклов обратного распространения ошибки по
- **712 844** уникальных слов в каждой эпохе

Результаты обучения

Класс	Точность	Усредненная результирующая точность:
0 - Ненависть	0.9781	84.27%
1 - Грусть	0.7682	
2 - Счастье	0.7379	
3 - Страх	0.9647	

Рисунок 79 - Результаты обучения

SOTA – оценка

Language specific algorithm	SOTA average accuracy	Dataset	Paper
M-BERT BaseFiT (Russian)	0.874	RuTweetCorp (full version)	https://github.com/sismetanin/sentiment-analysis-in-russian
Dual-trained Lazy CNN	0.843	RuTweetCorp (clean version)	текущая работа
nb-blinov (Russian)	0.816	ROMIP-2012	https://arxiv.org/ftp/arxiv/papers/1808/1808.07851.pdf
Naive-Bayes + Thesaurus (Russian)	0.697	RuTweetCorp (full version)	https://www.fruct.org/publications/fruct23/files/Lag.pdf

Рисунок 80 - SOTA оценка

Вопреки интуитивным предположениям и несмотря на примитивность архитектурного решения классификатор, основанный на линейных регрессорах показал весьма достойные результаты опередив как наивный Байесовский алгоритм, так и специализированную нейронную сеть Блинова.

Отсюда можно сделать вывод, что при проектировании систем распознавания эмоций в письменной речи, примитивным и достаточно тривиальным решениям уделяется недостаточно внимания несмотря на то, что они в достаточной степени эффективны, а главное являются легко воспроизводимыми на неспециализированной архитектуре такой как вентильные матрицы и ASIC-процессоры.

4.5 Нейронные сети и задача банковского скоринга

Задача банковского скоринга рассматривается как задача бинарной классификации. Для её решения могут использоваться алгоритмы однослойного персептрона и полносвязного многослойного персептрона, для оптимизации весов которых используются различные подходы, такие как стохастический градиентный спуск, стохастический градиентный спуск с учетом импульса, алгоритм Adam, а также методика цикличного изменения

скорости обучения. Рассмотрим данные алгоритмы и методы, а также основные понятия, связанные с задачами классификации.

Задача классификации

Рассмотрим функцию ошибки задачи классификации, метрики качества алгоритма, а также практические моменты связанные с балансировкой классов.

Функция ошибки — функция, значение которой требуется минимизировать для получения оценок параметров w модели, удовлетворяющих заданному требованию. Для задачи классификации требованием, накладываемым на функцию ошибки будет являться максимизация функции

$$p(y \mid X, w) = \prod_i p(y_i \mid x_i, w), \tag{48}$$

называемой функцией правдоподобия.

Обучающую выборку можно рассматривать, как реализацию обобщённой схемы Бернулли: для каждого объекта генерируется случайная величина, которая с вероятностью p (своей для каждого объекта) принимает значение 1 и с вероятностью $(1-p)$ – 0. Обозначим за $a_i = a(x_i, w)$ ответом алгоритма при заданных значениях параметров w на $i-ом$ объекте обучающей выборки. Предположим, что мы как раз и строим нашу модель так, чтобы она генерировала правильные вероятности, но тогда можно записать функцию (49) в следующем виде:

$$p(y \mid X, w) = \prod_i p(y_i \mid x_i, w) = \prod_i a_i^{y_i} (1-a_i)^{1-y_i} \to \max \tag{49}$$

После логарифмирования выражения (50) получаем, что его максимизация эквивалентна минимизации следующей функции:

$$\sum_i (-y_i \ln a_i - (1-y_i)\ln(1-a_i)) \to \min \tag{50}$$

Выражение (50) называют *логистической функцией ошибки* (logloss), и ввиду того, что она удовлетворяет требованию максимизации функции правдоподобия ее можно использовать в качестве функции ошибки в задачах

классификации. График логистической функции ошибки на одном объекте представлен на рисунке 81.

$$-\begin{cases} \log a_i, & y_i = 1, \\ \log(1-a_i), & y_i = 0. \end{cases}$$

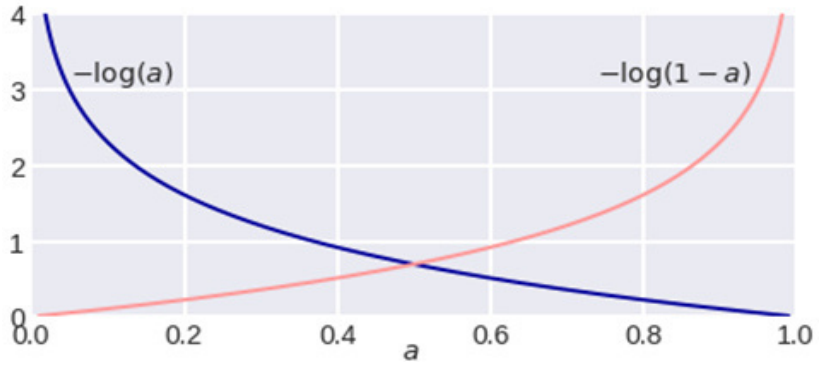


Рисунок 81 - Логистическая функция ошибки на одном объекте

Понятие метрики качества является очень важным в задачах классификации.

Метрика качества – количественный показатель, показывающий насколько хорошо данный алгоритм решает поставленную задачу.

Распространённым функционалом качества является *точность (Accuracy* или *Mean Consequential Error):*

$$Accuracy = \frac{1}{m}\sum_{i=1}^{m} I[a_i = y_i], \qquad (51)$$

т.е. доля объектов, на которых алгоритм выдал правильные ответы.

Для введения других метрик качества определим понятие матрицы ошибки.

Матрица ошибок (confusion matrix) – матрица размера 2×2, *ij*-я позиция которой равна числу объектов *i*-го класса, которым алгоритм присвоил метку *j*-го класса (Рисунок 82).

Рисунок 82 - Матрица ошибок

Объекты, которые алгоритм относит к положительному классу, называются *положительными (Positive)*, те из них, которые на самом деле принадлежат к этому классу – *истинно положительными (True Positive),* остальные – *ложно положительными (False Positive)*. Аналогичная терминология есть для *отрицательного (Negative)* класса. Дальше используем естественные сокращения:

- TP = True Positive,
- TN = True Negative,
- FP = False Positive,
- FN = False Negative.

Через введенные выше обозначения, можно представить формулу (52) через элементы матрицы ошибок:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{52}$$

Введем следующие метрики качества, наиболее часто используемые в задачах классификации.

Полнота (Sensitivity, True Positive Rate, Recall, Hit Rate) отражает, какой процент объектов положительного класса мы правильно классифицировали:

$$TPR = \frac{TP}{TP + FN}. \tag{53}$$

Точность (Precision, Positive Predictive Value) отражает какой процент положительных объектов (т.е. тех, что мы считаем положительными) правильно классифицирован:

$$PPV = \frac{TP}{TP + FP}. \quad (54)$$

F_1-мера (F_1 score) является средним гармоническим точности *(Precision)* и полноты, максимизация этого функционала приводит к одновременной максимизации этих двух критериев:

$$F_1 score = \frac{2}{TPR + PPV} = \frac{2TP}{2TP + FP + FN}. \quad (55)$$

F_2-мера (F_2 score) – среднее арифметическое точности *(Precision)* и полноты:

$$F_2 score = \frac{TPR + PPV}{2}. \quad (56)$$

Из функционалов качества, которые получаются из матрицы несоответствий**,** можно также отметить *специфичность (Specificity)* или *TNR – True Negative Rate:*

$$TNR = \frac{TN}{TN + FP}, \quad (57)$$

т.е. процент правильно классифицированных объектов негативного класса.

Также при определении метрик качества нельзя обойти стороной понятие ROC-кривой.

Пусть алгоритм выдаёт некоторую оценку (может, но не обязательно, вероятность) принадлежности объекта к классу 1. Можно считать, что оценка принадлежит отрезку [0, 1]. Тогда при варьировании порога решающего правила (числа, при превышении которого мы будем относить объект к классу 1), будем получать различный вид матрицы ошибок. Определим FPR (False Positive Rate), как:

$$FPR = 1 - TNR = \frac{FP}{TN + FP}. \quad (58)$$

ROC-кривая (receiver operating characteristic) – график, позволяющий оценить качество бинарной классификации, который отображает соотношение между долей объектов от общего количества носителей признака, верно классифицированных как несущие признак (TPR), и долей объектов от общего

количества объектов, не несущих признака, ошибочно классифицированных как несущие признак (FPR) при варьировании порога решающего правила.[6]

Пример построения ROC-кривой изображен на Рисунок 83.

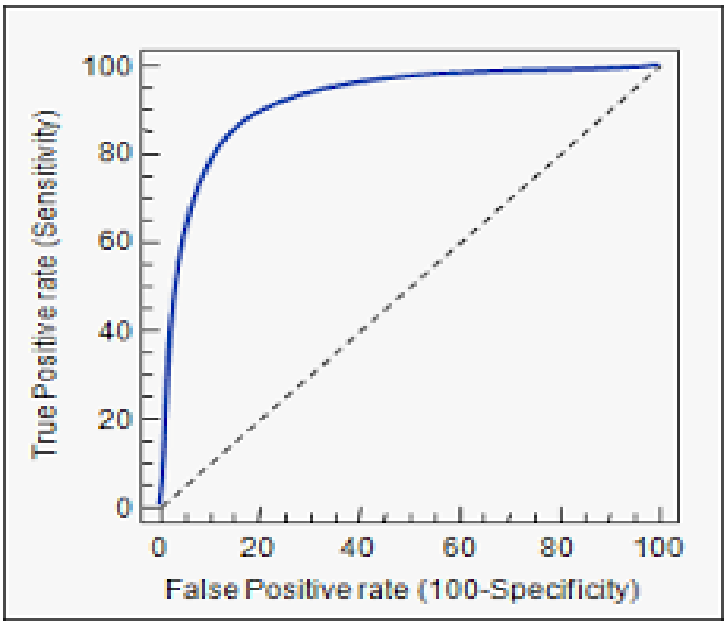


Рисунок 83 - Пример ROC-кривой

Количественную интерпретацию ROC даёт показатель *AUC* (*area under ROC curve*, *площадь под ROC-кривой*)

AUC – площадь, ограниченная ROC-кривой и осью доли ложных положительных классификаций.

Чем выше показатель AUC, тем качественнее классификатор, при этом значение 0,5 демонстрирует непригодность выбранного метода классификации (соответствует случайному гаданию). Значение менее 0,5 говорит, что классификатор действует с точностью до наоборот: если положительные назвать отрицательными и наоборот, классификатор будет работать лучше.

Усовершенствованные алгоритмы обучения нейронной сети

Представим усовершенствованные алгоритмы стохастического градиентного спуска и некоторые другие методики, связанные с варьированием скорости обучения модели.

[6] ROC-кривая [Электронный ресурс]: Википедия. Свободная энциклопе-дия. – URL: https://ru.wikipedia.org/wiki/ROC-кривая

Стохастический градиентный спуск с импульсом

Стохастический градиентный спуск остается популярной стратегией оптимизации, но обучение с его помощью иногда происходит слишком медленно. *Импульсный метод* призван ускорить обучение, особенно в условиях высокой кривизны, небольших, но устойчивых градиентов или зашумленных градиентов. В импульсном алгоритме вычисляется экспоненциально затухающее скользящее среднее прошлых градиентов и продолжается движение в этом направлении.

Формально говоря, в импульсном алгоритме вводится переменная, играющая роль скорости, – это направление и скорость перемещения в пространстве параметров. Скорость устанавливается равной экспоненциально затухающему скользящему среднему градиента со знаком минус. Работа стохастического градиентного спуска с импульсом продемонстрирована на рисунке 84.

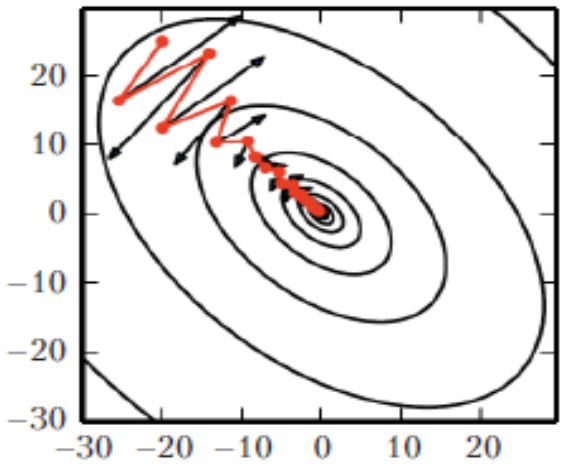


Рисунок 84 - Стохастический градиентный спуск с импульсом

Эллипсы обозначают изолинии функции потерь. Красная линия, пересекающая эллипсы, соответствует траектории, выбираемой в соответствии с правилом обучения с импульсом в процессе минимизации этой функции. Для каждого шага обучения стрелка показывает, какое направление выбрал бы в этот момент метод градиентного спуска. Как видим, плохо обусловленная квадратичная целевая функция выглядит как длинная узкая долина или овраг с крутыми склонами. Импульсный метод правильно перемещается вдоль оврага,

тогда как градиентный спуск впустую тратил бы время на перемещение вперед-назад поперек оврага.

Таким образом, запишем алгоритм стохастического градиентного спуска с импульсом:

1. При подаче на вход одного из образов рассчитать уровень активации.
2. Рассчитать $\delta^{(n)}$ для выходного слоя, а также скорость по формуле:

$$v^{(n)}(t) = \mu v^{(n)}(t-1) - \eta\delta^{(n)}, \quad (59)$$

где $v^{(n)}(t)$ – скорость на предыдущем шаге;

$v^{(n)}(t+1)$ – скорость на новом шаге;

μ – гиперпараметр, отвечающий за размер импульса.

3. Рассчитать значения $\delta^{(n)}$ и $v^{(n)}(t+1)$ для всех остальных слоёв.
4. Скорректировать все веса нейронной сети

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + v^{(n)}(t). \quad (60)$$

5. Если ошибка в сети существенна, перейти на шаг 1, в противном случае завершить обучение.

Чем больше μ относительно η, тем сильнее предшествующие градиенты влияют на выбор текущего направления.

Сходимость и устойчивость данного метода показана в работе[7].

Скорость обучения – один из самых трудных для установки гиперпараметров, поскольку она существенно влияет на качество модели. Функция ошибки зачастую очень чувствительна в некоторых направлениях пространства параметров и нечувствительна в других. Импульсный алгоритм может в какой-то мере сгладить эти проблемы, но ценой введения другого гиперпараметра. Если мы полагаем, что направления чувствительности почти параллельны осям, то, возможно, имеет смысл задавать скорость обучения отдельно для каждого параметра и автоматически адаптировать эти скорости на

[7]Ali Ramezani Kebrya, Ashish Khisti, Ben Liang On the Stability and Conver-gence of Stochastic Gradient Descent with Momentum// CoRR. – 2018. – Pp. 2-28

протяжении всего обучения. Исходя из этих соображений, зародились алгоритмы с адаптивной скоростью обучения.

Adam

Adam (сокращение от adaptive moments, адаптивные моменты) – один из алгоритмов с адаптивной скоростью обучения предложенный в работе Kingma Diederik P, Ba Jimmy Lei[8]. Он является комбинацией импульсного метода и метода масштабируемых градиентов RMSProp[9]. В Adam импульс включен непосредственно в виде оценки первого момента (с экспоненциальными весами) градиента. Adam включает поправку на смещение в оценки как первых моментов (член импульса), так и вторых (нецентрированных) моментов для учета их инициализации в начале координат. Сходимость метода описана в работе Kingma Diederik P, Ba Jimmy Lei [10].

Представим алгоритм метода Adam:

1. Инициализировать начальные параметры, скорость обучения η, первый и второй момент: $s = r = 0$, коэффициенты экспоненциального затухания моментов $\rho_1, \rho_2 \in [0,1)$ и небольшую константу $\gamma = 10^{-8}$ для обеспечения устойчивости.
2. При подаче на вход одного из образов рассчитать уровень активации.
3. Рассчитать $\delta^{(n)}$ для всех слоев и градиент функции ошибок $grad(E(w)) = \delta^{(n)} y^{(n-1)}$:
4. Обновить смещенную оценку первого момента:

$$\vec{s} = \rho_1 s + (1 - \rho_1)\overrightarrow{grad(E(w))}. \quad (61)$$

5. Обновить смещенную оценку второго момента:

$$r = \rho_2 r + (1 - \rho_2)(\overrightarrow{grad(E(w))}, \overrightarrow{grad(E(w))}) \quad (62)$$

6. Скорректировать смещение моментов:

[8] Kingma Diederik P, Ba Jimmy Lei. Adam: A method for stochastic optimiza-tion. arXiv preprint arXiv:1412.6980, 2014
[9] Sebastian Ruder. 2016. An overview of gradient descent optimization algo-rithms. arXiv preprint arXiv:1609.04747
[10] Kingma Diederik P, Ba Jimmy Lei. Adam: A method for stochastic optimiza-tion. arXiv preprint arXiv:1412.6980, 2014

$$\hat{\vec{s}} = \frac{\vec{s}}{1-\rho_1^t}, \tag{63}$$

$$\hat{r} = \frac{\hat{r}}{1-\rho_2^t}, \tag{64}$$

где t – номер шага.

7. Вычислить изменения весов:

$$\Delta w^{(n)}(t) = \frac{\hat{\vec{s}}}{\sqrt{\hat{r}} + \gamma}. \tag{65}$$

8. Пересчитать веса:

$$w^{(n)}(t) = w^{(n)}(t-1) - \eta \Delta w^{(n)}(t) \tag{66}$$

9. В случае если точность удовлетворяет желаемым требованиям закончить алгоритм, иначе вернуться на шаг 2.

Стратегии изменения скорости обучения. Циклическая скорость обучения

Традиционно в практической реализации нейронных сетей принят следующий критерий остановки обучения. После того, как изменение функции ошибки достигает какого-то предела снизить скорость обучения в несколько раз, затем подождать несколько итераций обучения (эпох) и в случае, если изменение ошибки мало, снова снизить скорость обучения и действовать таким образом, пока скорость обучения не достигнет какого-то минимально заданного значения. Но также есть и обратный подход, называемый циклической скоростью обучения[11].

Подход возник из наблюдения, что увеличение скорости обучения дает краткосрочный негативный эффект, но в итоге получается долгосрочный позитивный эффект. Суть заключается в том, что задаются некие границы изменения скорости обучения и на каждой итерации скорость обучения циклически изменяется по заданному профилю внутри этих границ (Рисунок 85).

[11] Leslie N Smith. Cyclical learning rates for training neural networks. arXiv pre-print arXiv:1506.01186v3, 2016

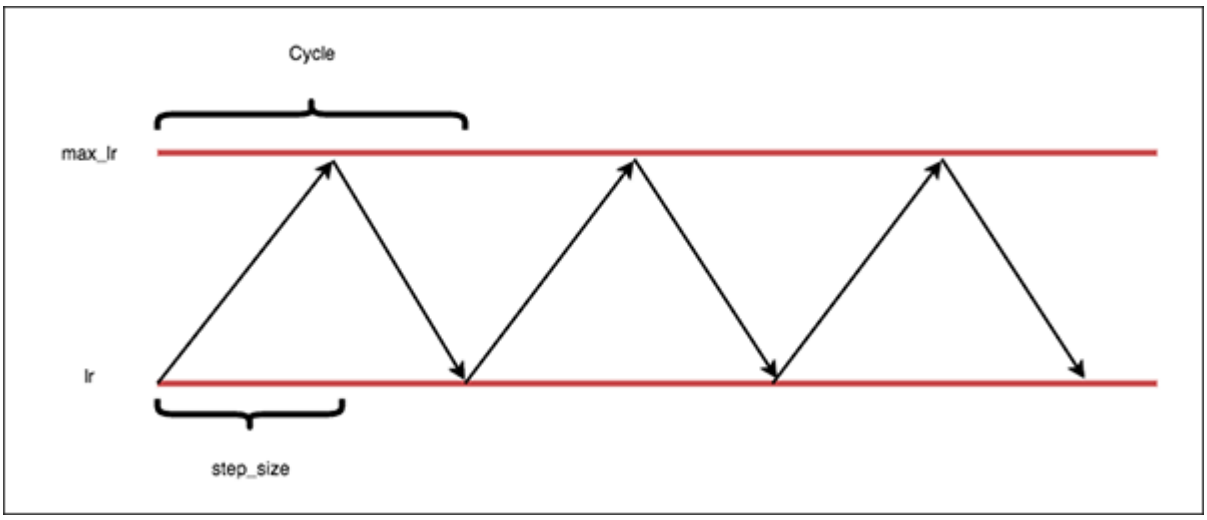


Рисунок 85 - Циклическая скорость обучения

Потом каждые несколько итераций профиль начинает «сжиматься», как показано на Рисунок 86.

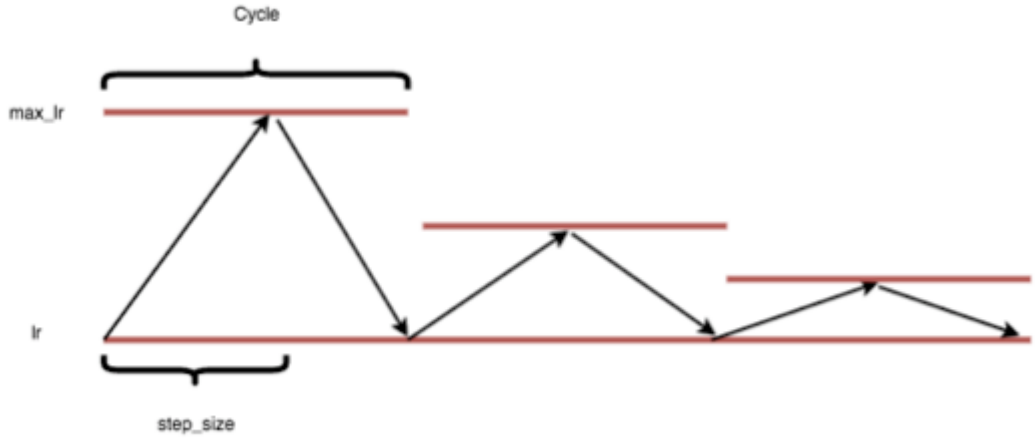


Рисунок 86 - «Сжатие» циклической скорости обучения

Но при выходе на плато вместо того, чтобы снижать скорость обучения, вернем её границы к первоначальным значениям и подождем некоторое число эпох. Если в результате получатся меньшие значения функции ошибки, то продолжим обучения, если большие, то вернемся к весам, которые получились при выходе на плато.

Данный подход позволяет «вылезти» из локального минимума или седловой точки и довольно часто применяется на практике.

Список литературы

1. Афанасьев В.Н., Юзбашев М.М. Анализ временных рядов и прогнозирование: Учебник. М.: Финансы и статистика, 2012. 228 с.
2. Дулькейт Е.И. Прогнозирование с помощью искусственных нейронных сетей // Прикладная математика и фундаментальная информатика. 2015. № 2. С. 118 – 126
3. Нейронная сеть [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C
4. Подкорытова О.А., Соколов М.В. Анализ временных рядов: Учебное пособие. М.: Юрайт, 2020. 267 с
5. Ali Ramezani Kebrya, Ashish Khisti, Ben Liang On the Stability and Convergence of Stochastic Gradient Descent with Momentum// CoRR. – 2018. – Pp. 2-28
6. Kingma Diederik P, Ba Jimmy Lei. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014
7. Leslie N Smith. Cyclical learning rates for training neural networks. arXiv preprint arXiv:1506.01186v3, 2016
8. ROC-кривая [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/ROC-кривая
9. Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747