МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению практических работ
по дисциплине «Машинное обучение и анализ данных для решения задач электроэнергетики»
для студентов направления подготовки /специальности
13.04.02 «Электроэнергетика и электротехника»,

Магистерская программа «Интеллектуальные системы электроснабжения "

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

Практическая работа №1

Математическое моделирование нейрона

Математический нейрон был предложен американскими учеными Уорреном Мак-Каллоком и Вальтером Питтсем в 1943г.

Математический нейрон – это математическая модель биологического нейрона мозга. Его изображают в виде кружочка со стрелками, обозначающими входы и выход. На рис.1 математический нейрон имеет J входов и один выход.

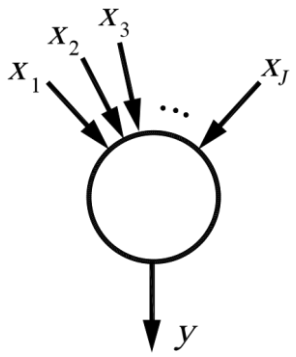


Рис. 1. Математический нейрон Мак-Каллока – Питтса

Через входы математический нейрон принимает входные сигналы x_j, которые суммирует, умножая каждый входной сигнал на некоторый весовой коэффициент w_j:

$$S = \sum_{j=1}^{J} w_j x_j . \qquad (1)$$

Затем математический нейрон формирует свой выходной сигнал согласно правилу:

$$y = \begin{cases} 1, & \text{если } S \geq \theta \\ 0, & \text{если } S < \theta \end{cases}, \qquad (2)$$

в котором величину θ называют порогом чувствительности нейрона. Таким образом, математический нейрон может существовать в двух состояниях. Если взвешенная сумма входных сигналов S меньше порога θ, то его выходной сигнал y равен нулю. В этом случае говорят, что нейрон не возбужден. Если же входные сигналы достаточно интенсивны и их взвешенная сумма достигает

порога чувствительности θ, то нейрон переходит в возбужденное состояние, и на его выходе, согласно формуле (2), образуется сигнал $y=1$.

Весовые коэффициенты w_j имитируют электропроводность нервных волокон – силу синаптических связей между нейронами. Чем эти силы выше, тем больше вероятность перехода нейрона в возбужденное состояние. С другой стороны, вероятность перехода нейрона в возбужденное состояния повышается при уменьшении порога чувствительности θ .

Логическая функция (2) называется активационной функцией нейрона. Ее графическое изображение имеет вид, представленный на рис.2. За этот вид ее иногда называют «функцией-ступенькой».

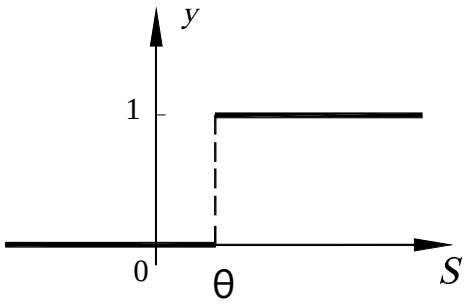


Рис. 2. Активационная функция нейрона: «ступенька»

С помощью математического нейрона можно моделировать различные логические функции, например, функцию логического умножения «И» («AND»), функцию логического сложения «ИЛИ» («OR») и функцию логического отрицания «НЕТ» («NOT»). Таблицы истинности этих логических функций приведены на рис.3.

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

«И»

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

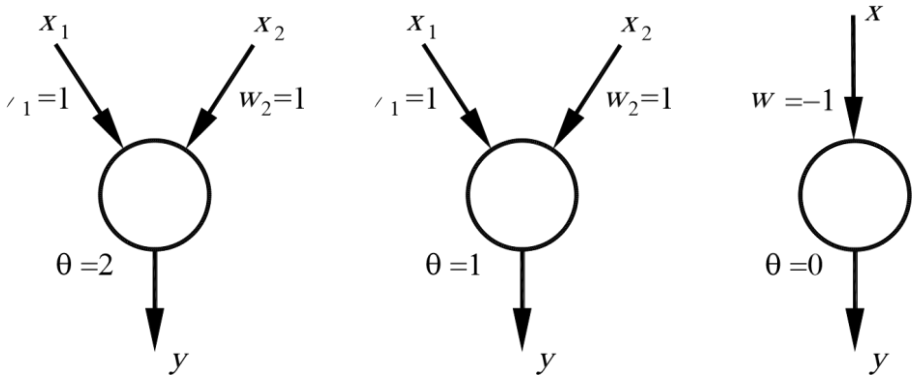
«ИЛИ»

x	y
0	1
1	0

«НЕТ»

Рис. 3. Таблицы истинности логических функций

С помощью этих таблиц и формул (1)-(2) нетрудно убедиться, что математический нейрон, имеющий два входа с единичными силами синаптических связей $w_1 = w_2 = 1$, моделирует функцию логического умножения «И» при $\theta = 2$, Этот же нейрон моделирует функцию логического сложения «ИЛИ» при задании $\theta = 1$. Математический нейрон с одним входом моделирует функцию «НЕТ» при задании $w = -1$ и $\theta = 0$.



"И" "ИЛИ" "НЕТ"

Рис.4. Математические нейроны, моделирующие логические функции

Однако существуют логические функции, которые невозможно моделировать с помощью математического нейрона Мак-Каллока – Питтса. Такой логической функцией является «Исключающее ИЛИ», таблица истинности которой приведена на рис. 5.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Рис.5. Таблица истинности функции «Исключающее ИЛИ»

Задачи, которые подобно проблеме «Исключающего ИЛИ» с помощью однослойного персептрона решены быть не могут, называют *линейно неразделимыми задачами*. В свое время ученые потратили немало сил и средств, пытаясь решить такие задачи, ошибочно полагая, что причина их неудач состоит в недостаточной мощности существующих компьютеров и в недостаточном количестве совершенных попыток.

По-видимому, нечто подобное может случиться и с Вами при выполнении лабораторной работы №1. Подобрав значения синаптических весов w_1, w_2 и порога θ, Вы успешно справляетесь с моделированием логических

функций «И» и «ИЛИ», тогда как попытки моделирования функции «Исключающее ИЛИ» к успеху не приводят.

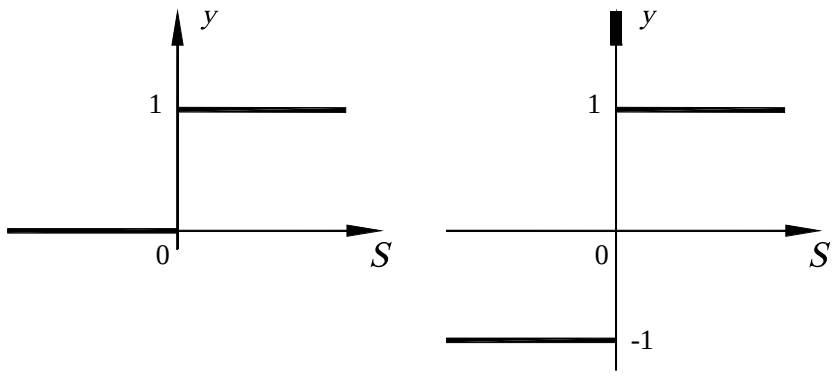
В заключение отметим, что в современной литературе иногда вместо понятия порога чувствительности нейрона θ используют термин *нейронное смещения* b, которое отличается от порога θ только знаком: $b = -\theta$. Если величину b добавить к сумме (1):

$$S = \sum_{j=1}^{J} w_j x_j + b, \qquad (3)$$

то пороговая активационная функция нейрона примет вид:

$$y = \begin{cases} 1, & \text{если } S \geq 0; \\ 0, & \text{если } S < 0, \end{cases} \qquad (4)$$

Графическое представление этой активационной функции приведено на рис. 6, *а*.



а *б*

Рис. 6. Пороговые активационные функции нейрона, заданные формулами: а – (4); б – (5)

Еще более симметричный вид, представленный на рис. 6, *б*, активационная функция нейрона приобретает при использовании формулы: y

$= \begin{cases} 1, \text{если } S \geq 0; \end{cases}$ (5)

$$\square -1, \text{ если } S < 0 ,$$

В формуле (3) нейронное смещение b можно рассматривать как вес w_0 некоторого дополнительного входного сигнала x_0, величина которого всегда равна единице:

$$S = \sum_{j=1}^{J} w_j x_j + w_0 x_0 = \sum_{j=0}^{J} w_j x_j \qquad (6)$$

Нейрон с дополнительным входом x_0 изображен на рис. 7.

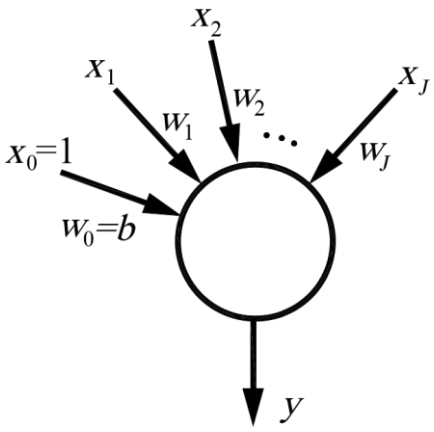


Рис. 7. Нейронное смещение b интерпретируется как вес дополнительного входа w_0, сигнал которого x_0 всегда равен 1

Задание: В приложенном симуляторе подобрать параметры нейронов для предложенных функций.

Содержание отчета:
- Скриншоты выполненных заданий
- Выводы

Практическая работа № 2

ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ АППРОКСИМАЦИИ НЕИЗВЕСТНЫХ ФУНКЦИЙ.

Цель работы: Изучить особенности применения нейронных сетей для моделирования систем с неизвестной структурой.

Введение

При постановке и решении задач моделирования пользуются понятием “черного ящика”.

Черным ящиком называют систему, внутреннее содержание которой неизвестно наблюдателю, а наблюдению доступны только вход $\overline{X}$ и выход $\overline{Y}$ системы.

Рисунок В.1 – Условное изображение черного ящика

Целью исследования черного ящика является построение математической модели этого объекта, т.е. определение зависимости, связывающей вход и выход системы.

Если осуществить достаточно длительный эксперимент, в котором на вход черного ящика подаются различные входные воздействия $\overline{X}$ и наблюдаются соответствующие им реакции $\overline{Y}$ системы, то после анализа результатов эксперимента можно, несмотря на незнание внутренней структуры системы, составить правильное представление о ее поведении. Это дает возможность сделать достаточно достоверное предсказание поведения системы в непроверенных условиях.

Компоненты вектора $\overline{X}$ называют *факторами или предикторами* (т.е. предсказателями), реакцию системы $\overline{Y}$ - *откликом*. Ограничимся случаем, когда выходной параметр Y – скалярная величина. Неизвестную функцию

$y = \varphi(x_1, x_2, \cdots, x_n)$, которая, как мы предполагаем, связывает $\overline{X}$ и Y, назовем *функцией отклика*, а ее график – *поверхностью отклика*.

Итак, описание функционирования черного ящика сводится к модели, устанавливающей соответствие между входами и выходом системы или, что то же, между *факторами* и *функцией отклика*.

Построение модели системы по результатам описанного эксперимента называют *идентификацией* черного ящика.

Пример: моделирование системы с одним входом и одним выходом. Это знакомая задача об аппроксимации экспериментальной зависимости в случае, когда даны n значений $(x_i, y_i), i = 1,2,...,n$, а вид сглаживающей функции по условию неизвестен.

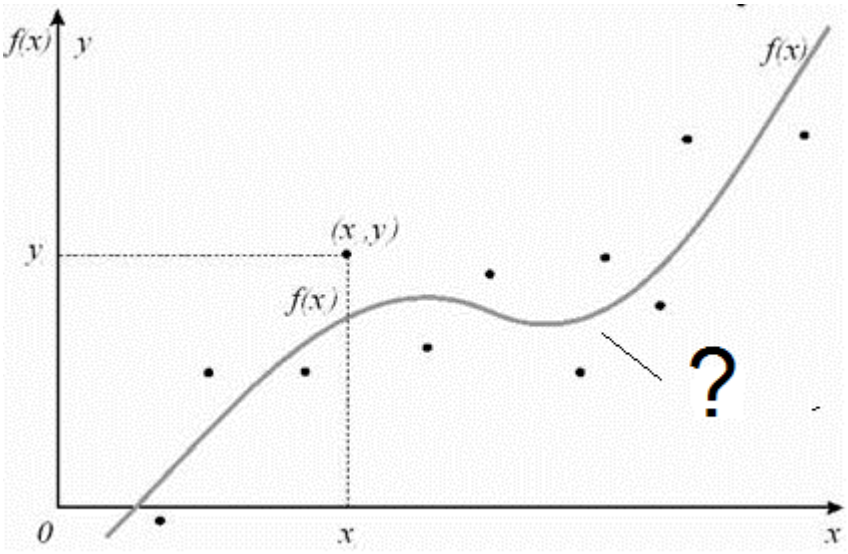


Рисунок В.2 – Аппроксимация неизвестной функции

По данным эксперимента (не привлекая дополнительных данных) нельзя сделать общие выводы о внутренней структуре системы, так как одинаковым поведением могут обладать различные по своей структуре системы. Например, одно и то же соотношение между входами и выходами может согласовываться с несколькими математическими выражениями.

Пусть, например, мы последовательно подаем на вход системы целые числа 1,2,3,4,5,6 и получаем на выходе числа 2,4,6,8,10,12. Разумно предположить, что для любого входного значения n на выходе появляется число 2n и следующими на выходе будут числа 14, 16., 18,..., 2n. Однако результаты опыта столь же хорошо согласуются с формулой
2n+(n-1)(n-2)(n-3)(n-4)(n-5)(n-6).
Основываясь на ней, можно предсказать, что следующими числами в выходной последовательности будут 734, 5056 и т.д. - результат, резко отличающийся от предыдущего.
В этом и заключается проблема идентификации черного ящика – основная проблема моделирования.

В настоящей лабораторной работе рассматривается способ идентификации черного ящика, который не требует никаких предположений о внутреннем содержании системы. Он основан на использовании искусственных нейронных сетей и позволяет получить функциональный аналог черного ящика, практическое применение которого не требует знания его структуры.

1.1 Краткие теоретические сведения о нейронных сетях

Нейронная сеть представляет собой совокупность формальных нейронов, определенным образом связанных между собой. Это сильно упрощенная модель коры головного мозга человека.

Формальный нейрон (далее - нейрон) моделирует некоторые функции своего прототипа - биологического нейрона. Он имеет некоторое количество входов (синапсов), на которые поступают входные сигналы x_i , и один выход (аксон), с которого снимается выходной сигнал y. Каждый синапс имеет вес w_i, на который умножается входной сигнал x_i.

Структура нейрона представлена на рисунке 1.1.

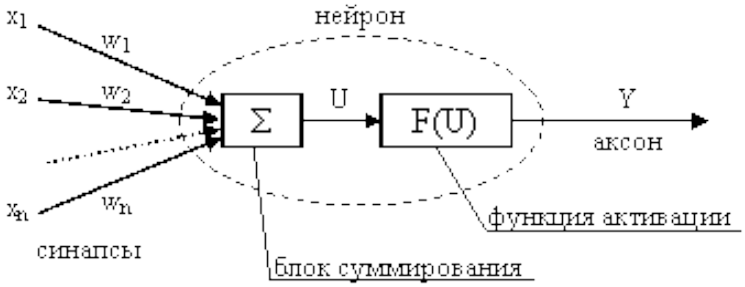


Рисунок 1.1 – Структура нейрона

Внутри нейрона можно выделить блок суммирования, определяющий взвешенную сумму всех входных сигналов

$$U = \sum_{i=1}^{n} w_i \cdot x_i \qquad (1.1)$$

и блок функции активации $Y = F(U)$.

Формальный нейрон функционирует за два такта:

1) суммирование входных сигналов;
2) вычисление Y по функции активации.

Функция активации должна удовлетворять двум условиям:

1) $|F(U)| < 1$ при любом U,
2) функция должна быть монотонной неубывающей.

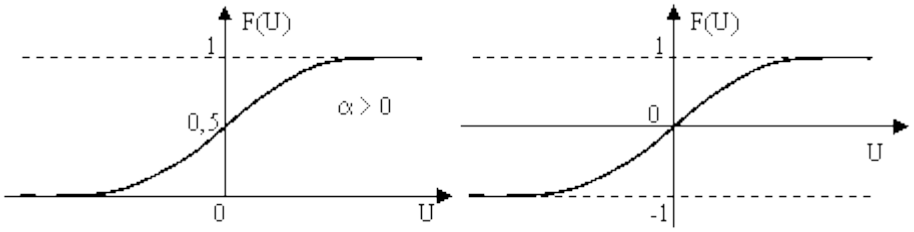
Наиболее часто в качестве функций активации используются следующие функции:

1) ступенчатая функция

$$F(U) = \begin{cases} 1, & если\ U \geq \alpha \\ 0, & если\ U < \alpha \end{cases}; \qquad (1.2)$$

2) сигмоидная функция (рисунок 1.2,а)

$$F(U) = \frac{1}{1 + e^{-\alpha U}} . \qquad (1.3)$$



а) б)

Рисунок 1.2 – Функции активации

3) гиперболический тангенс (рисунок 2,б)

$$F(U) = th(U) = \frac{e^{U} - e^{-U}}{e^{U} + e^{-U}} \tag{1.4}$$

4) гладкие сжимающие функции

$$F(U) = \frac{U + Q}{|U + Q| + \alpha}, \tag{1.5}$$

где Q – порог (смещение),

a - параметр, определяющий крутизну статической характеристики нейрона.

Нейроны образуют нейронные сети путем соединения синапсов с аксонами. Наиболее распространенными и хорошо изученными являются трехслойные НС, состоящие из трех слоев нейронов: входного, скрытого и выходного (рисунок 1.3). Это пример НС с 3 входными, 4 скрытыми и 2 выходными нейронами. Такая НС для краткости обозначается как (3-4-2).

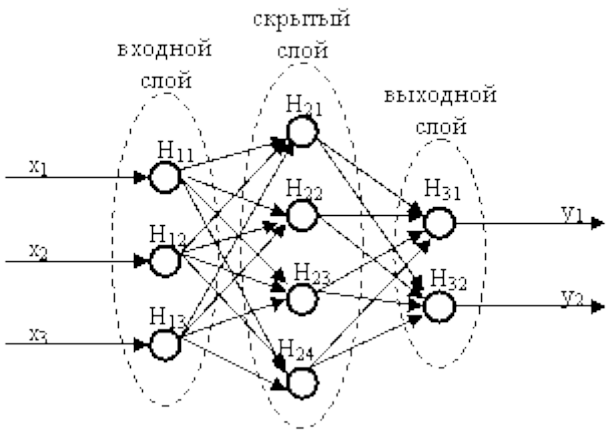


Рисунок 1.3 – Нейронная сеть вида (3-4-2)

Нейроны входного слоя имеют только по одному синапсу. Количество нейронов входного слоя соответствует количеству входных переменных сети *X*. Нейроны этого слоя нужны только для распределения входных сигналов по нейронам скрытого слоя, суммирования и вычисления функции активации в них не происходит. Количество нейронов в скрытом слое может быть различным и часто подбирается экспериментально. Недостаточное или избыточное количество нейронов в скрытом слое приводит к ухудшению точности аппроксимации. Кроме того, избыточное количество усложняет сеть и уменьшает быстродействие. Нейроны выходного слоя формируют выходные сигналы, их количество соответствует количеству выходов *Y*.

Данные сети относятся к сетям прямого распространения, поскольку в них входные сигналы последовательно проходят через все нейроны и после преобразований напрямую подаются на выходы. Выходной сигнал y_{ij} каждого *j*-го нейрона в *i*-м слое определяется как

$$y_{ij} = F\left(\sum_{k=1}^{n(i-1)} w_{ij}^{k} \cdot y_{i-1,k}\right), \qquad (1.6)$$

где $n(i)$ – число нейронов в *i*-м слое.

Наиболее популярный класс многослойных сетей прямого распространения образуют многослойные персептроны (MLP), в которых каждый вычислительный элемент использует пороговую или сигмоидальную функцию активации. Многослойный персептрон может формировать сколь угодно сложные границы принятия решения и реализовывать произвольные булевы функции.

Нейронные сети способны аппроксимировать любые функции

$$Y = F(X), \qquad (1.7)$$

где *Y* – вектор выходных переменных, *X* – вектор входных. Их использование позволяет решить задачу о «черном ящике».

Процесс аппроксимации заключается в подборе весовых коэффициентов w_{ij} и называется обучением нейронной сети (НС. То есть НС может функционировать в двух режимах:

1. Режим обучения, когда происходит корректировка весов таким образом, чтобы выходные сигналы наиболее точно соответствовали желаемым.
2. Режим эксплуатации, когда на вход подаются сигналы, а на выходе снимаются результаты вычислений.

От качества обучения НС зависит точность ее работы в режиме эксплуатации.

Схема процесса обучения представлена на рисунке 1.4, где обозначены: $Y_{жел}$ – желаемые значения выходных сигналов, E – ошибка обучения ($E = Y_{жел} – Y$), K – корректирующие воздействия (обычно изменения весов Δw_{ij}).

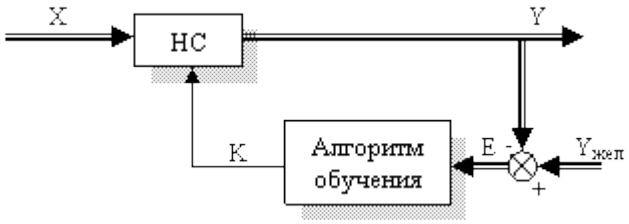


Рисунок 1.4 – Процесс обучения НС

Для обучения НС составляется обучающая выборка входных сигналов и соответствующих им выходных. Выборка может быть разделена на две части: рабочую выборку (на основе которой производится собственно обучение) и тестирующую выборку (для проверки качества обучения).

Далее определяется структура НС. Для трехслойной НС количества входных и выходных нейронов определяются по количествам входных и выходных переменных.

Весам синапсов необученной НС изначально присваиваются произвольные значения. Далее на вход НС подается первый вектор X из рабочей выборки, определяется вектор Y и ошибка обучения E. Исходя из значений вектора E корректируются веса синапсов. Затем подается следующий

вектор X из выборки и т.д. Циклы обучения повторяются многократно, пока качество обучения не станет удовлетворительным (это можно проверить по тестирующей выборке).

Существует несколько методов обучения, которые можно классифицировать по способам использования учителя:

1. Обучение с учителем (коррекция весов производится исходя из сравнения текущего и желаемого выходных векторов);
2. Обучение с последовательным подкреплением знаний (сети не даются желаемые значения выходов, а ставится оценка «хорошо» или «плохо»);
3. Обучение без учителя (сеть сама вырабатывает правила обучения путем выделения особенностей из набора входных данных).

В рассматриваемом случае используется обучение с учителем (обучение на примерах).

Для обучения обычно используется НС с функциями активации сигмоидного типа. Целью обучения по правилу обратного распространения является минимизация ошибки обучения, которая определяется как

$$E = \frac{1}{2}\sum_{i=1}^{N_m}\left(Y_i - Y_{жел.i}\right)^2 . \qquad (1.9)$$

Для уменьшения ошибки веса изменяются по правилу

$$w_{ij} = w_{ij} - \nu\frac{\partial E}{\partial w_{ij}} \qquad (1.10)$$

где n - константа, характеризующая скорость обучения. Данная формула описывает процесс градиентного спуска в пространстве весов.

Алгоритм обратного распространения ошибки состоит из следующих шагов.

Шаг 1. На вход НС подается вектор X из обучающей выборки и вычисляются выходы всех нейронов Y_{ij}.

Шаг 2. Определяется величина градиента ошибки *EI* для каждого нейрона выходного слоя:

$$EI_{Nj} = (Y_j - Y_{эксп.j}) \cdot Y_j \cdot (1 - Y_j), \quad (1.11)$$

где Y_j – выход *j*-го нейрона выходного слоя.

Шаг 3. Двигаясь от последнего слоя к первому, определяются градиенты *EIij* для каждого *j*-го нейрона каждого *i*-го слоя:

$$EI_{ij} = Y_{ij} \cdot (1 - Y_{ij}) \cdot \sum_{k=1}^{n(i+1)} EI_{i+1,k} \cdot w_{i+1,k}^{j}, \quad (1.12)$$

где *k* – номер синапса, соединяющего нейрон H_{ij} с нейроном $H_{i+1,k}$ следующего слоя.

Шаг 4. Коррекция весов синапсов:

$$w_{ij}^{k} = w_{ij}^{k} - \nu \cdot EI_{ij} \cdot Y_{i-1,k}. \quad (1.13)$$

Коррекция весов для входного слоя не производится.

Шаг 5. Если обучающая выборка не закончилась, то шаги 1 – 5 повторяются.

Шаг 6. Определяется величина ошибки *E*. Если она недостаточно мала, то шаги 1 – 6 повторяются.

Из описанного алгоритма видно, что процесс обучения НС включает два вложенных цикла обучения: внутренний цикл (шаги 1 – 5) повторяется соответственно количеству примеров из обучающей выборки, внешний (шаги 1 – 6) – до тех пор, пока не будет достигнуто удовлетворительное (с точки зрения ошибки Е) качество обучения.

После успешного обучения НС может быть протестирована на тестовой выборке. Если ошибка обучения на каждом примере из тестовой выборки удовлетворительна, то НС можно считать обученной и приступать к ее эксплуатации.

Пример одного цикла обучения НС.

В качестве примера можно взять обучающую выборку:

x_1	x_2	y
1	0	2
2	1	6
4	2	16

Здесь x_1 и x_2 – входные параметры НС, y – желаемый выходной параметр.

Поскольку у аппроксимируемой функции два входных параметра и один выходной, то выбирается НС с двумя нейронами во входном слое и одним в выходном. Количество нейронов скрытого слоя примем равным двум, то есть формируется сеть вида 2-2-1 (см. рисунок 1.5).

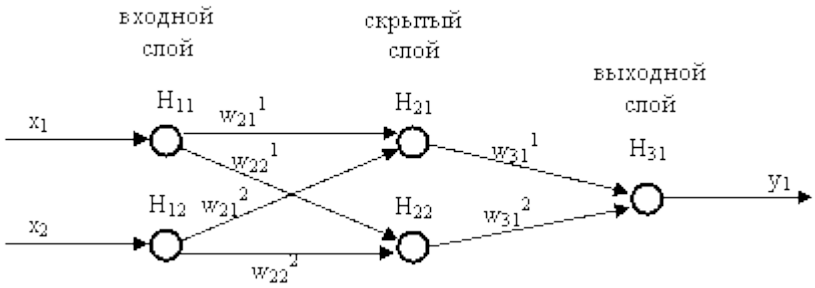


Рисунок 1.5 – Пример НС

В качестве функции активации выбирается сигмоидная функция с коэффициентом $a = 1$. Начальные значения весов синаптических связей принимаются равными 0,5: $w_{21}^1 = w_{22}^1 = w_{21}^2 = w_{22}^2 = w_{31}^1 = w_{31}^2 = 0{,}5$.

Поскольку исходные значения x_1, x_2 и y не лежат в пределах [0, 1], их необходимо нормировать, поделив, например, x_1 на 4, x_2 на 2, а y на 16. В результате получена нормированная выборка:

x_1	x_2	y
0,25	0	0,125
0,5	0,5	0,375
1	1	1

Скорость обучения принимается равной $n = 0{,}2$.

После подготовки можно приступать к обучению.

Шаг 1. На входы НС подается первый вектор входных параметров: $x_1 = 0,25$ и $x_2 = 0$. При этом $y_{жел} = 0,125$.

Выходы нейронов входного слоя: $Y_{11} = 0,25$, $Y_{12} = 0$.

Для скрытого слоя:

$U_{21} = w_{21}^1 * Y_{11} + w_{21}^2 * Y_{12} = 0,5*0,25 + 0,5*0 = 0,125;$

$U_{22} = w_{22}^1 * Y_{11} + w_{22}^2 * Y_{12} = 0,5*0,25 + 0,5*0 = 0,125;$

$Y_{21} = 1 / (1 + exp(-a*U_{21})) = 1 / (1 + exp(-0,125)) = 0,5312;$

$Y_{22} = 1 / (1 + exp(-a*U_{22})) = 1 / (1 + exp(-0,125)) = 0,5312.$

Для выходного слоя:

$U_{31} = w_{31}^1 * Y_{21} + w_{31}^2 * Y_{22} = 0,5*0б5312 + 0,5*0б5312 = 0,5312;$

$Y_{31} = 1 / (1 + exp(-a*U_{31})) = 1 / (1 + exp(-0,5312)) = 0,6298.$

Шаг 2. Величина градиента для выходного нейрона

$EI_{31} = (Y_{31} – y_{жел})*Y_{31}*(1 – Y_{31}) = (0,6298 – 0,125)*0,6298*(1 - 0,6298) = 0,1177.$

Шаг 3. Величины градиентов для скрытого слоя:

$EI_{21} = Y_{21}*(1 – Y_{21})*[EI_{31}*w_{31}^1] = 0,5312*(1 – 0,5312)*0,1177*0,5 = 0,01466,$

$EI_{22} = Y_{22}*(1 – Y_{22})*[EI_{31}*w_{31}^2] = 0,5312*(1 – 0,5312)*0,1177*0,5 = 0,01466.$

Шаг 4. Коррекция весов синапсов:

$w_{21}^1 = w_{21}^1 - n*Y_{11}*EI_{21} = 0,5 – 0,2*0,25*0,01466 = 0,4993,$

$w_{22}^1 = w_{22}^1 - n*Y_{11}*EI_{22} = 0,5 – 0,2*0,25*0,01466 = 0,4993,$

$w_{21}^2 = w_{21}^2 - n*Y_{12}*EI_{21} = 0,5 – 0,2*0*0,01466 = 0,5,$

$w_{22}^2 = w_{22}^2 - n*Y_{12}*EI_{22} = 0,5 – 0,2*0*0,01466 = 0,5,$

$w_{31}^1 = w_{31}^1 - n*Y_{21}*EI_{31} = 0,5 – 0,2*0,5312*0,1177 = 0,4875,$

$w_{31}^2 = w_{31}^2 - n*Y_{22}*EI_{31} = 0,5 – 0,2*0,5312*0,1177 = 0,4875.$

Если при полученных весах на вход НС подать тот же вектор входных параметров, то на выходе будет $y = 0,6267$, что уже ближе к желаемому $y_{жел} = 0,125$. То есть данный цикл обучения приблизил ответ НС к желаемому на величину $\Delta y = 0,6298 – 0,6267 = 0,0031$.

Поскольку обучающая выборка не закончилась, то шаги 1 – 4 повторяются аналогично для следующего вектора входных параметров

1.3 Варианты заданий

Лабораторная работа выполняется с использованием программного нейроимитатора *Neural Network Wizard* **(***NNW***)** в порядке, описанном ниже.

Необходимо:

1) Получить задание согласно варианту (см. табл.).

1	$Y=2x1+x2$
2	$Y=x1^2+x2$
3	$Y=x1+x2^2$
4	$Y=\sqrt{x1}+x2$
5	$Y=\sqrt{x2}+x1$
6	$Y=2x1^2+3x2$
7	$Y=2x1+2x2^2$
8	$Y=3x1+6x2$
9	$Y=6x1+2x2^2$
10	$Y=3x1^2+3x2$

ПРИМЕЧАНИЕ. График функции должен находиться в верхней полуплоскости (особенность нейроимитатора *NNW*).

1) Подготовить обучающую выборку средствами приложения Microsoft Excel и оформить ее в виде файла *.csv (не забудьте заменить десятичные запятые на точки). Для первого и второго столбцов выборки сформировать случайные числа (от 20 наборов). *Примечание: Чтобы создать набор случайных чисел, воспользуйтесь функцией: =СЛЧИС()*100.*

2) В третьем столбце рассчитайте значение заданной функции. *Сохраните файл, выбрав расширение .txt.* Можно создать обучающую выборку "вручную" в виде текстового файла, располагая точки вдоль оси абсцисс более или менее равномерно.

3) Провести обучение нескольких нейронных сетей с помощью нейроимитатора *Neural Network Wizard* с количеством нейронов в скрытом слое, равном 1, 3, 5, 7 и функцией активации «сигмоидная».

4) Проверить качество каждой обученной сети, для чего рассчитать значения выходной переменной по заданной функции и по нейронной сети, затем на графике показать близость каждой обученной модели к исходной.

5) В каждом случае результат представить в виде пары графиков (заданная функция, созданная сетью функция), построенных в одной системе координат.

1.4. Порядок работы с Neural Network Wizard

Программный пакет *Neural Network Wizard* может быть использован без предварительной инсталляции. Путь к программе: *Neural Network Wizard\Bin\Wizard.exe.*

Прежде чем запускать программу, необходимо в любом текстовом редакторе подготовить текстовый файл с обучающей выборкой. Пример такого файла для функции res = s1 + s2 приведен ниже.

s1	s2	res
0	0	0
1	1	2
2	2	4
3	3	6
4	4	8
5	5	10
6	6	12

В первой строке файла указываются имена входных/выходных переменных: *s1* и s2 – имена входных переменных, *res* – имя выходной переменной. Далее идут их значения в колонках. Файл сохраняется как текстовый с расширением TXT.

После запуска программы в первом окне задается имя файла. Кнопка «Далее» позволяет перейти к следующему окну (рисунок 1.6).

В новом окне перечислен список доступных полей, взятый из первой строки указанного файла. Для каждого из полей необходимо указать, чем является данная переменная.

Если данная переменная входная, то в группе «Использовать поле как» выбирается вариант «Входное», если выходная – «Целевое». Кроме того, для каждой переменной можно указать вид нормализации (приведения к диапазону [0,1]) с соответствующими параметрами. Кнопка «Далее» переводит к следующему окну.

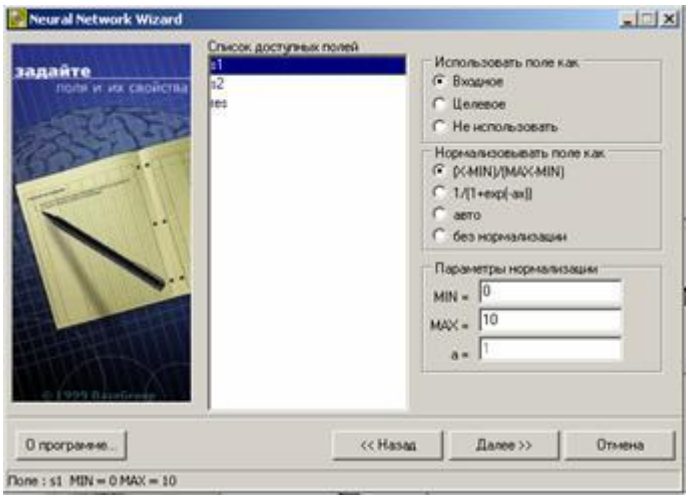


Рисунок 1.6

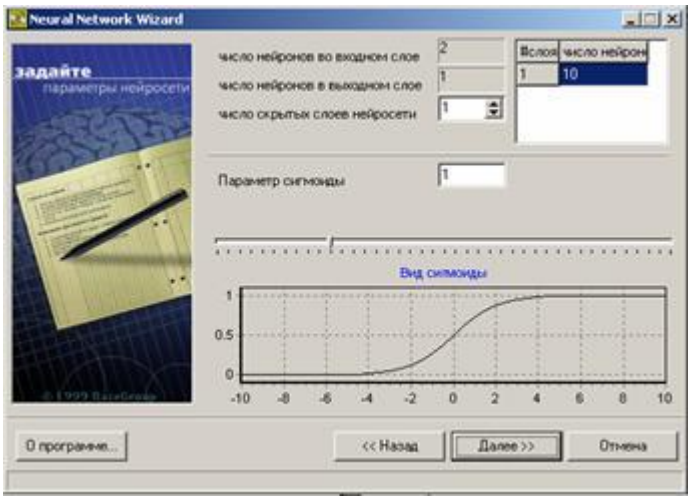


Рисунок 1.7

В данном окне (рисунок 1.7) определяются структура и параметры НС: количество скрытых слоев, количество нейронов в каждом слое, а также вид сигмоидной функции.

В следующем окне (рисунок 1.8) задаются параметры обучения и критерии остановки обучения, если она требуется. Кнопка «Далее» показывает краткий предварительный отчет.

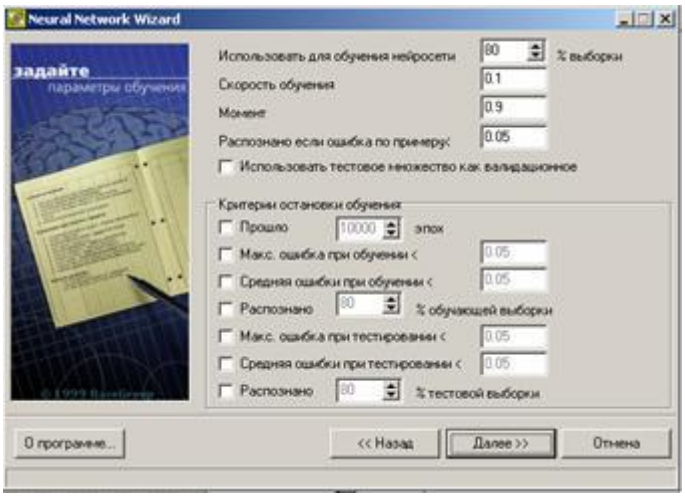


Рисунок 1.8

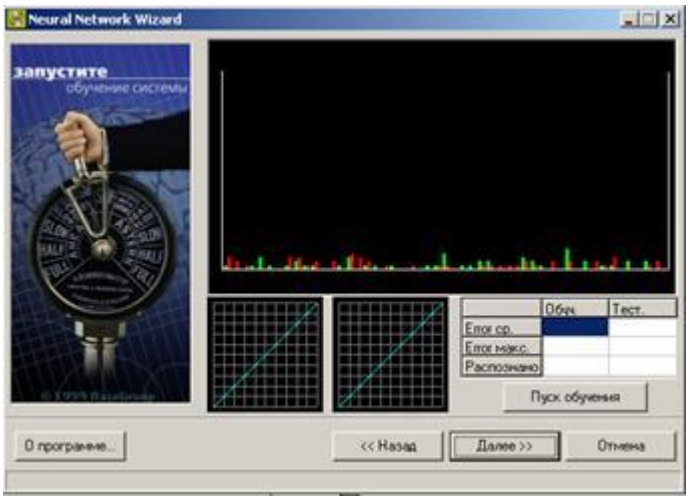


Рисунок 1.9

В следующем окне визуализирован процесс обучения (рисунок 1. 9). Чтобы запустить обучение, нажимается кнопка «Пуск обучения». На верхней диаграмме показано распределение ошибки обучения: по горизонтали значение ошибки (чем правее столбец, тем больше ошибка), по вертикали количество примеров из выборки с данной ошибкой.

Зеленые столбцы – ошибка на рабочей обучающей выборке, красные – на тестовой. В процессе обучения столбцы должны стремиться в левую часть диаграммы.

Ниже диаграммы отображается распределение примеров на рабочей и тестовой выборках. Каждый пример изображен здесь точкой. Чем ближе точка к диагонали, тем точнее НС предсказала ее значение.

Остановка обучения происходит либо по ранее указанному критерию, либо с помощью той же кнопки «Пуск обучения».

В следующем окне (рисунок 1.10) предоставляется возможность оценить точность работы НС в эксплуатационном режиме. Для этого в левом поле указывается произвольное значение входного сигнала. После нажатия на кнопку «Расчет» в правом поле появляется рассчитанное сетью значение выходного. Кнопка «Сохранить» позволяет записать параметры обученной сети в виде файла (по умолчанию расширение файла NNW).

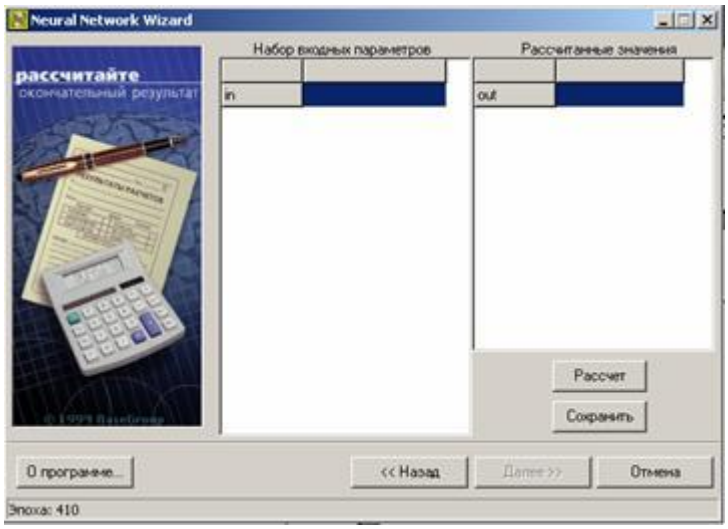


Рисунок 1.10

В данном файле кроме прочих параметров указаны:

Epoch – количество эпох (циклов) обучения,

Layer_* - количество нейронов в соответствующем слое (нейроны нумеруются от 0 до N-1),

$W\_i\_j\_k$ – веса синапсов (i = номер слоя - 2, j – номер нейрона, k – номер синапса данного нейрона).

1.5 Содержание отчета

Результаты оформить в виде отчета. В отчете представить:

1. Цель работы.
2. Задание.
3. Обучающую выборку.
4. Параметры каждой обученной сети: количества нейронов во входном, скрытом и выходном слоях и т.д.,
5. Графики, построенные для заданной функции и обученных моделей .
6. Выводы по результатам работы..

1.6 Контрольные вопросы

1. В чем заключается проблема идентификации черного ящика?
2. Что такое нейронная сеть и каковы ее основные свойства?
3. Какова структура нейрона?
4. Какие функции активации могут быть использованы в нейронных сетях?
5. Какие требования предъявляются к функциям активации?
6. Какие функции выполняет входной слой в многослойной сети?
7. Можно ли обучить нейронную сеть без скрытого слоя?
8. В чем заключается обучение нейронных сетей?
9. Чем отличается обучение с учителем от обучения без учителя?
10. Почему входные и выходные сигналы нейронной сети должны быть нормированы, т.е. приведены к диапазону [0,1]?

Список литературы

1. Роберт Хехт-Нильсен. Нейрокомпьютинг: история, состояние, перспективы // Открытые системы. -1998. -№ 4-5. -С. 23 - 28.
2. Розенблатт Ф. Принципы нейродинамики. Персептроны и теория механизмов мозга. -М.: Мир, 1965.

3. Гордиенко Е.К., Лукьяница А.А. Искусственные нейронные сети. I Основные определения и модели// Изв. РАН. Техническая кибернетика. - 1994. -№ 5. -С. 79 - 92.
4. Короткий С.Г. Нейронные сети: алгоритм обратного распространения. - BYTE/Россия. -2000. -№ 5. -С. 26-29.
5. Интеллектуальные системы управления с использованием нейронных сетей: учеб. пособие. / В.И. Васильев, Б.Г. Ильясов, С.С. Валеев и др.; Уфимск. гос. авиац. техн. ун-т. Уфа, 1997. -92 с.
6. Куликов Г.Г., Брейкин Т.В., Арьков В.Ю. Интеллектуальные информационные системы: учеб. пособие / Уфимск. гос. авиац. техн. ун-т. -Уфа, 1999. -129 с.
7. Интеллектуальные системы обработки информации на основе нейросетевых технологий: учеб. пособие. / Ю.И. Зозуля, Уфимск. гос. авиац. техн. ун-т. – Уфа. - 2000. 138 с.

Практическая работа № 3-4. Исследование нейронных сетей в пакете STATISTICA.

Порядок исследования нейронных сетей в пакете STATISTICA.

Шаг 1. Вы начинаете со стартовой панели (посмотрите на рис. 1).

В данной панели вы можете выбрать различные виды анализа, которые Вам необходимо выполнить: регрессию, классификацию, прогнозирование временных рядов (с непрерывной и категориальной зависимой переменной), кластерный анализ.

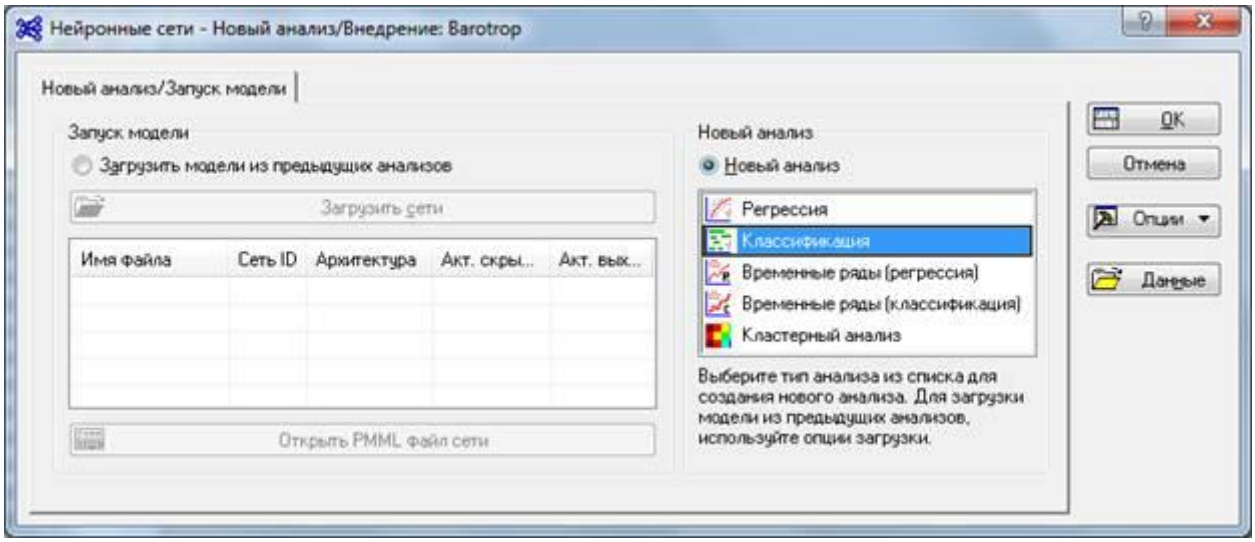


Рис. 1. Стартовая панель STATISTICA Automated Neural Networks (SANN)

Выберите, например, *Временные ряды (регрессия)*, если вы хотите построить прогноз, или *Классификация*, если решается задача классификации.

Нажав кнопку *ОК*, перейдем к диалоговому окну выбора данных.

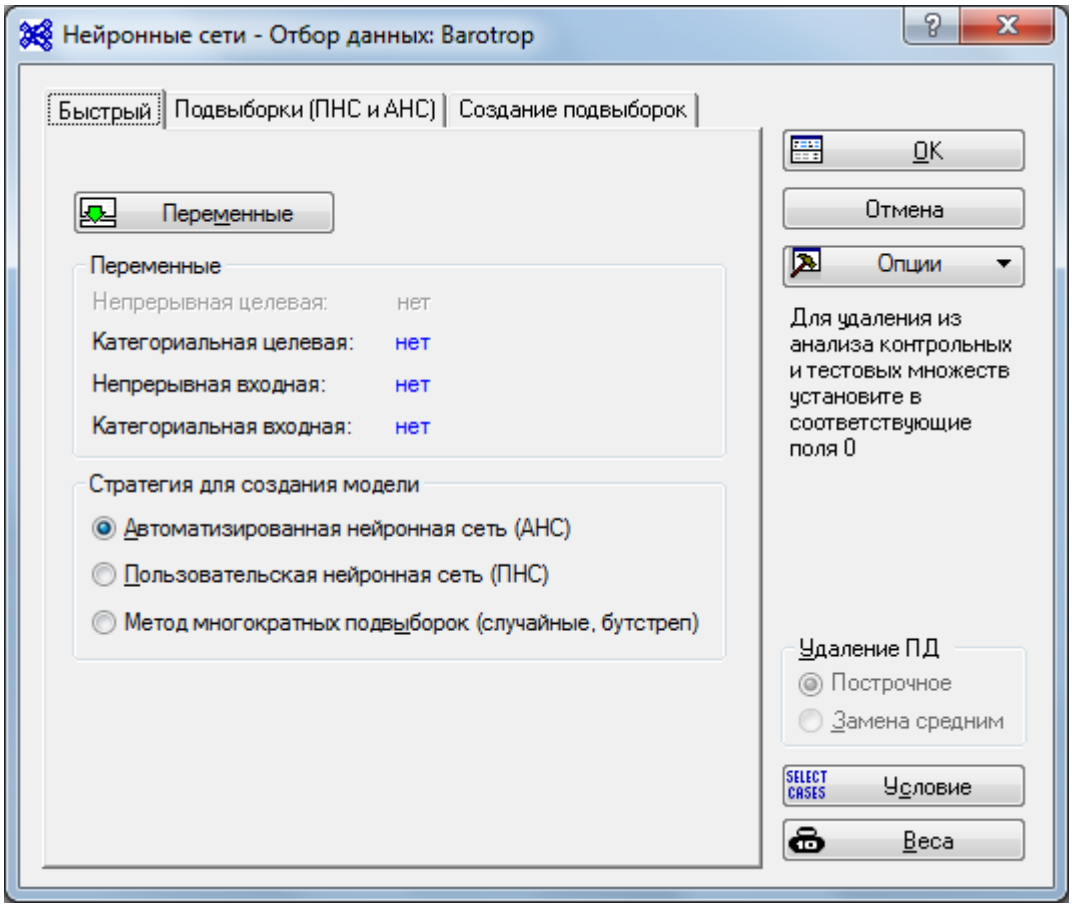


Рис. 2. Диалоговое окно Нейронные сети – Отбор данных - вкладка Быстрый

Шаг 2. На вкладке *Быстрый* следует выбрать необходимые переменные для анализа. Переменные могут быть непрерывными и категориальными, зависимыми и независимыми; кроме того, наблюдения могут принадлежать разным выборкам.

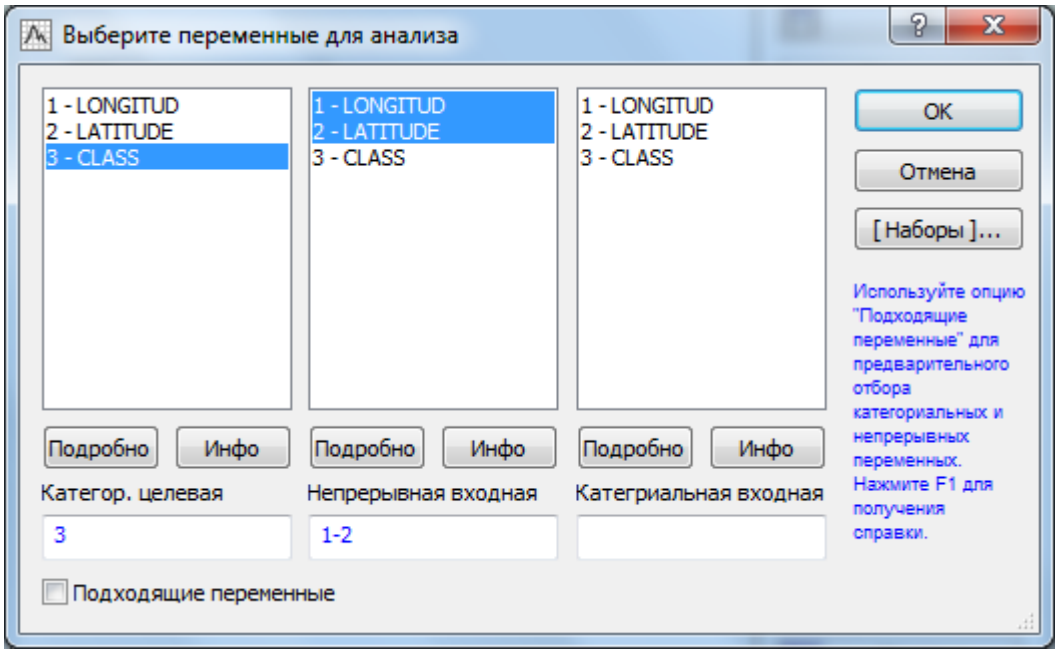


Рис. 3. Окно выбора переменных

Далее следует выбрать стратегию построения нейронной сети.

Для начинающих пользователей рекомендуется выбирать стратегию *Автоматизированная нейронная сеть (АНС)*. Опытный пользователь может с легкостью использовать любую доступную стратегию: *Автоматизированная нейронная сеть (АНС), Пользовательская нейронная сеть (ПНС)* и *Метод многократных подвыборок*. Мы выберем *Автоматизированная нейронная сеть (АНС)*.

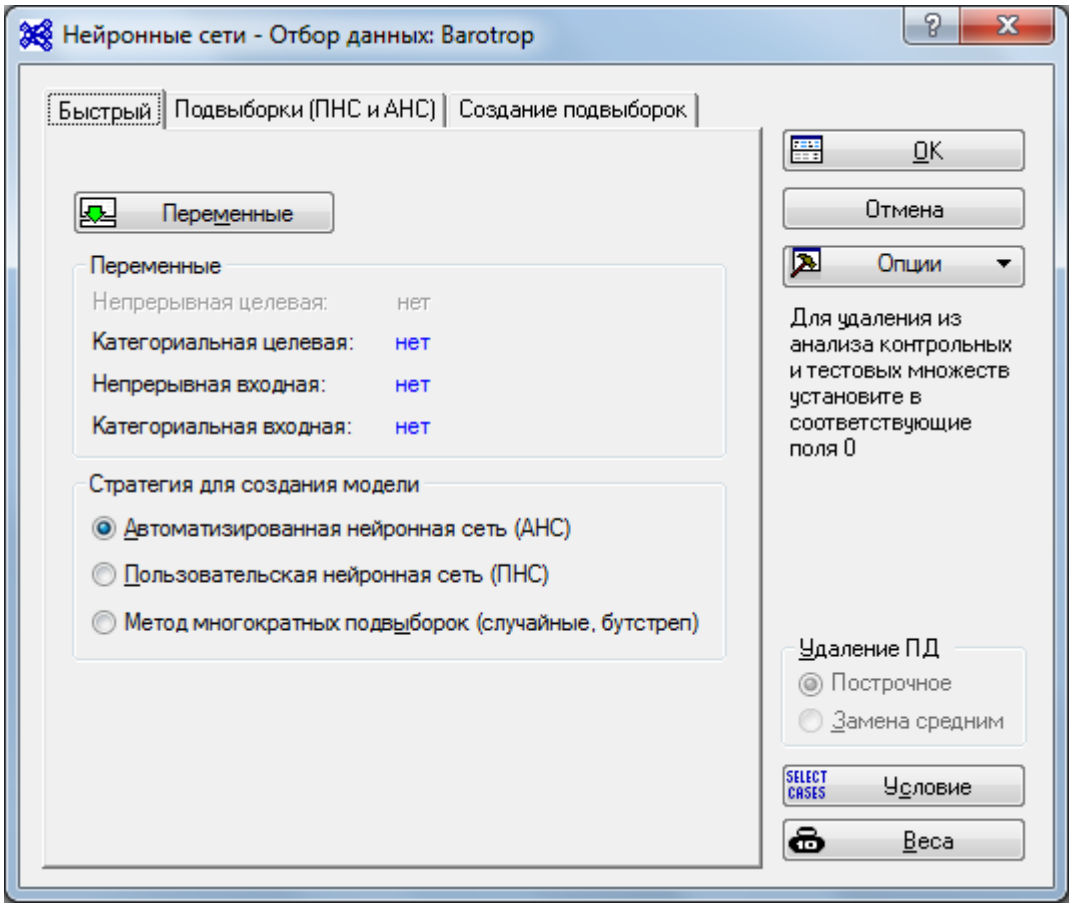


Рис. 4. Диалоговое окно Нейронные сети – Отбор данных - вкладка Быстрый

На вкладке *Подвыборки (ПНС и АНС)* следует задать желаемое разбиение данных на подвыборки: обучающую, контрольную и тестовую. Разбиение можно задавать случайным образом, а можно фиксировать с помощью дополнительной переменной кодов.

В данном случае будем использовать случайное разбиение.

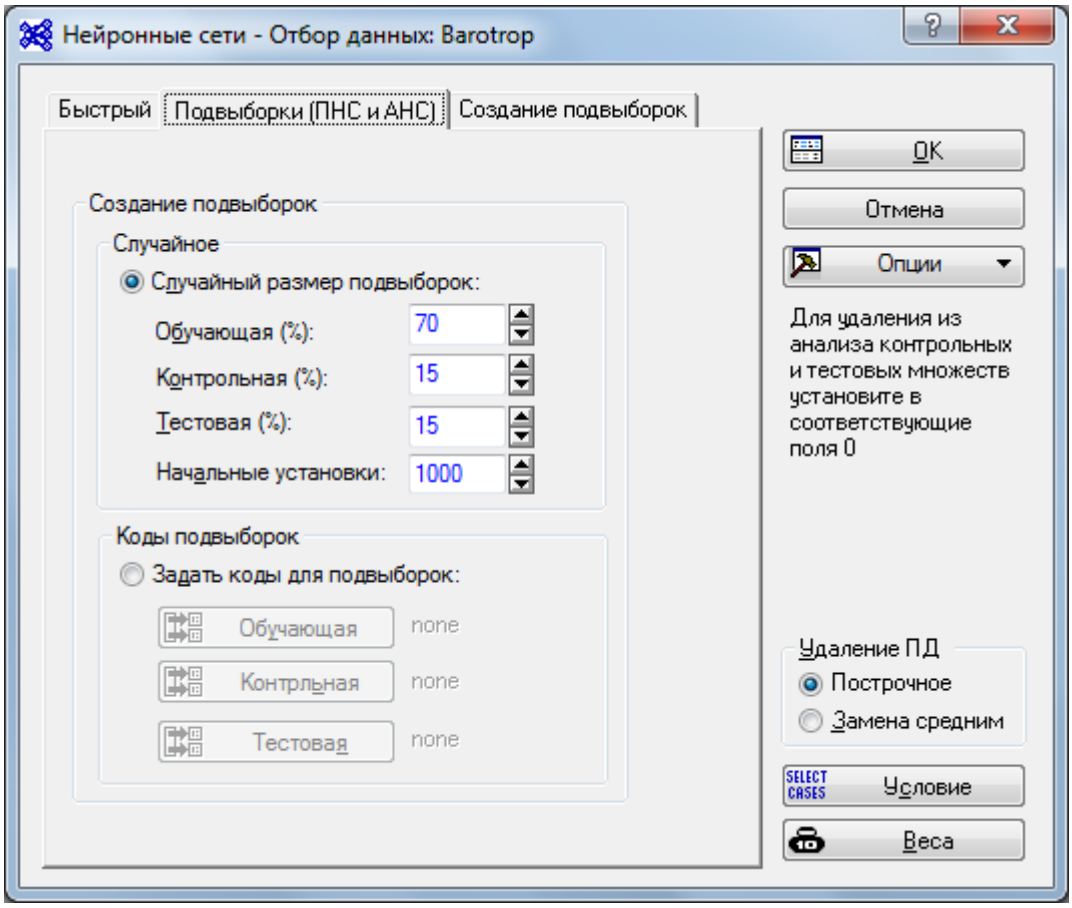


Рис. 5. Диалоговое окно Нейронные сети – Отбор данных - вкладка Подвыборки (АНС и ПНС)

Вкладка *Подвыборки (ПНС и АНС)* предназначена для первых двух стратегий: *Автоматизированная нейронная сеть (АНС)* и *Пользовательская нейронная сеть (ПНС)*; а вкладке *Создание подвыборок* используется для последней стратегии: *Метод многократных подвыборок.*

Нажимаем *ОК* и переходим к шагу задания параметров архитектуры.

Шаг 3. На вкладке *Быстрый* диалогового окна *Автоматизированные нейронные сети* необходимо указать тип сети, количество скрытых нейронов, количество обучаемых и сохраняемых сетей, а также тип используемых функций ошибок.

Программа предлагает следующие типы сетей: многослойные персептроны и сети радиальных базисных функций.

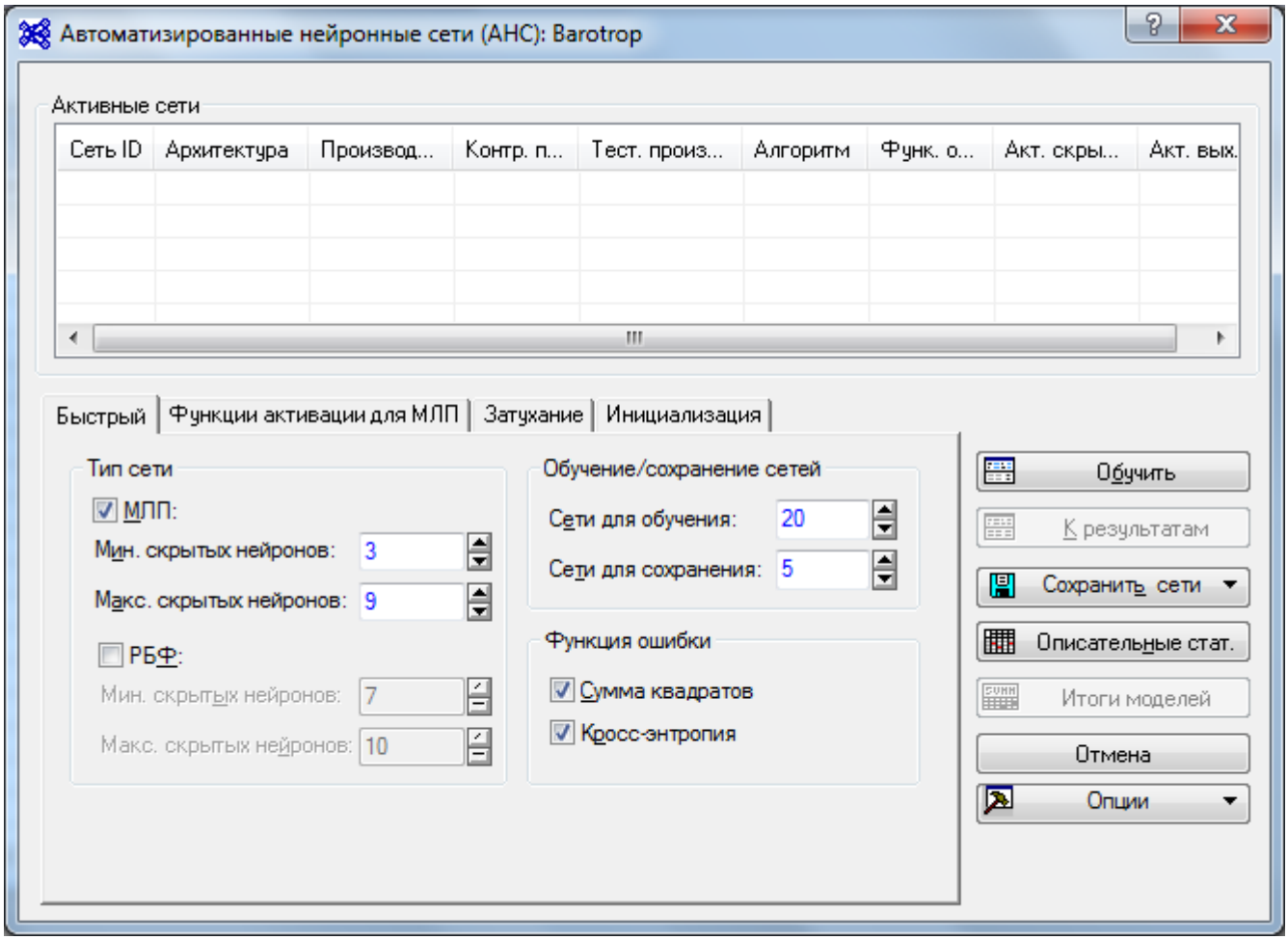


Рис. 6. Диалоговое окно Автоматизированные нейронные сети - вкладка Быстрый

Далее на вкладке *Функции активации для МЛП* задаются типы функций активаций, которые мы хотим использовать в различных нейросетевых моделях на разных слоях многослойных персептронов.

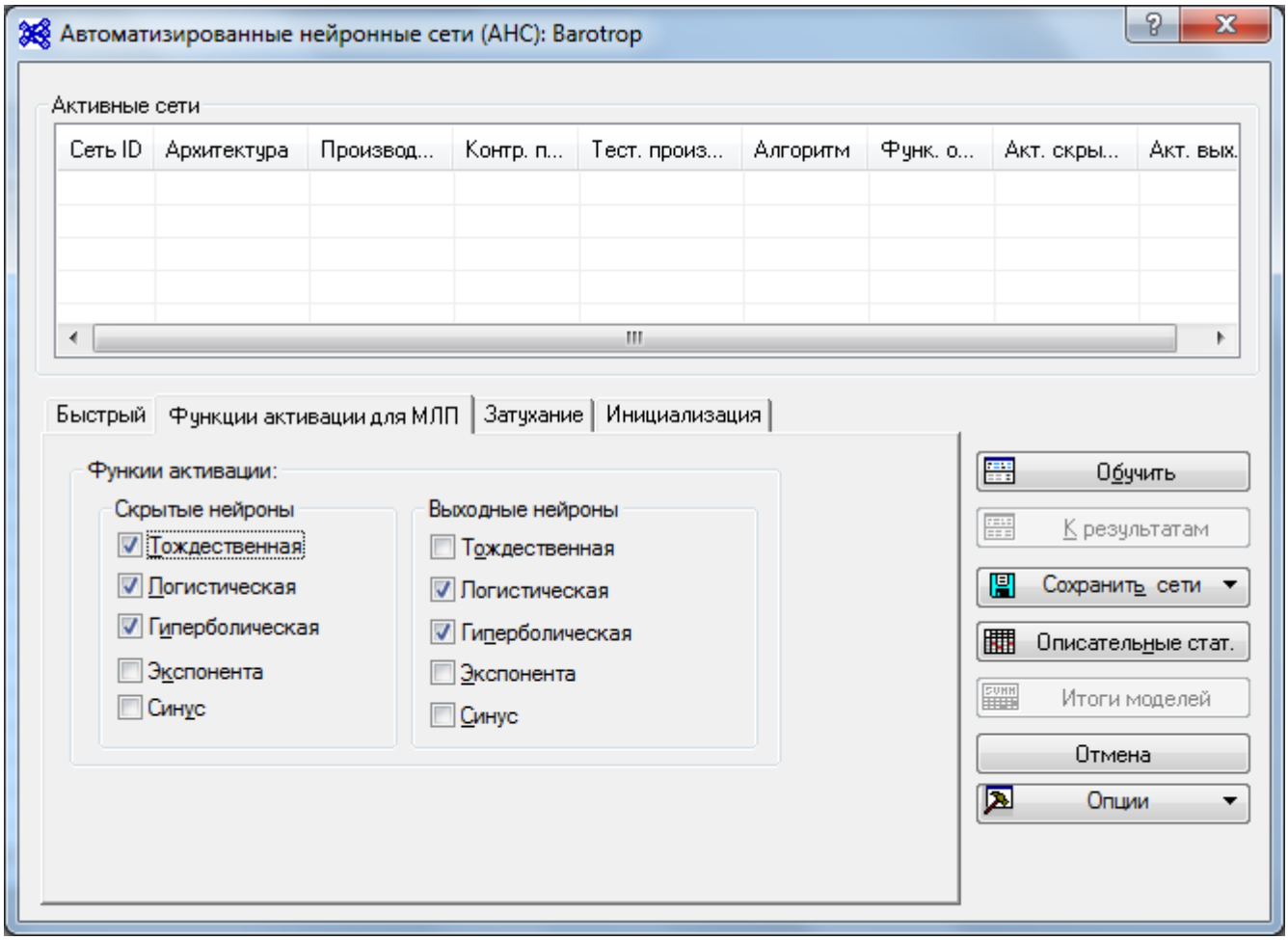


Рис. 7. Диалоговое окно Автоматизированные нейронные сети - вкладка Функции активации для МЛП

На вкладке *Затухание* можно включить опцию регуляризации весов, которая будет регулировать сложность обучаемых сетей. Это полезно, когда задача имеет большое число входных переменных, а также задано большое число нейронов на скрытом слое.

Но в нашем случае мы это использовать не будем.

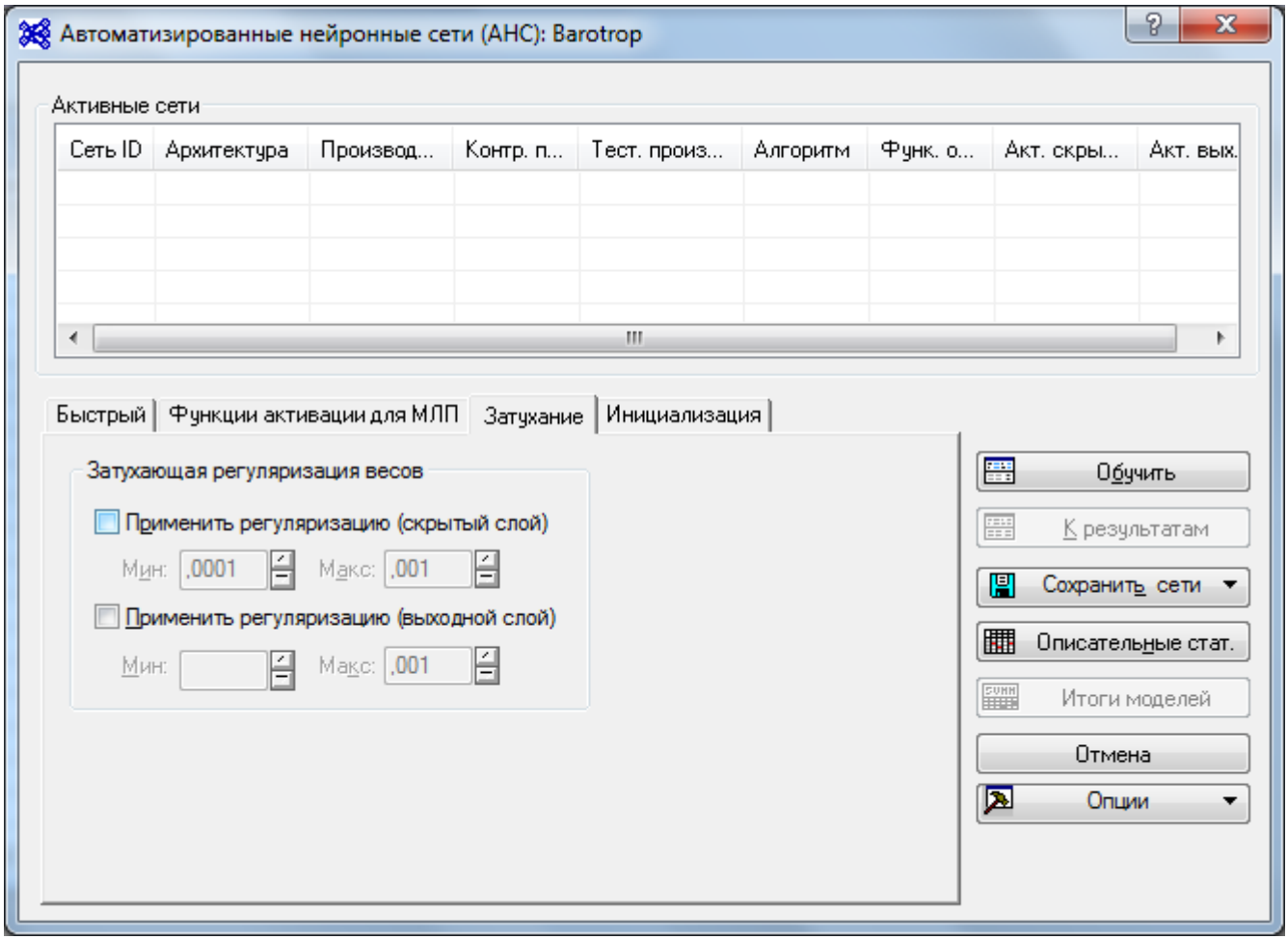


Рис. 8. Диалоговое окно Автоматизированные нейронные сети - вкладка Затухание

Теперь можно перейти к шагу обучения нейронных сетей.

Шаг 4. Запустите процедуру обучения нейронных сетей, нажав кнопку *ОК*. В диалоговом окне, приведенном на рис. 9, отображается некоторая информация о текущей обучаемой нейронной сети. Мы можем анализировать архитектуру сети, смотреть за ходом итераций алгоритма и фиксировать ошибки моделей. Для регрессии используется среднеквадратичная ошибка, для классификации используется процент правильной классификации наблюдений (как в нашем случае).

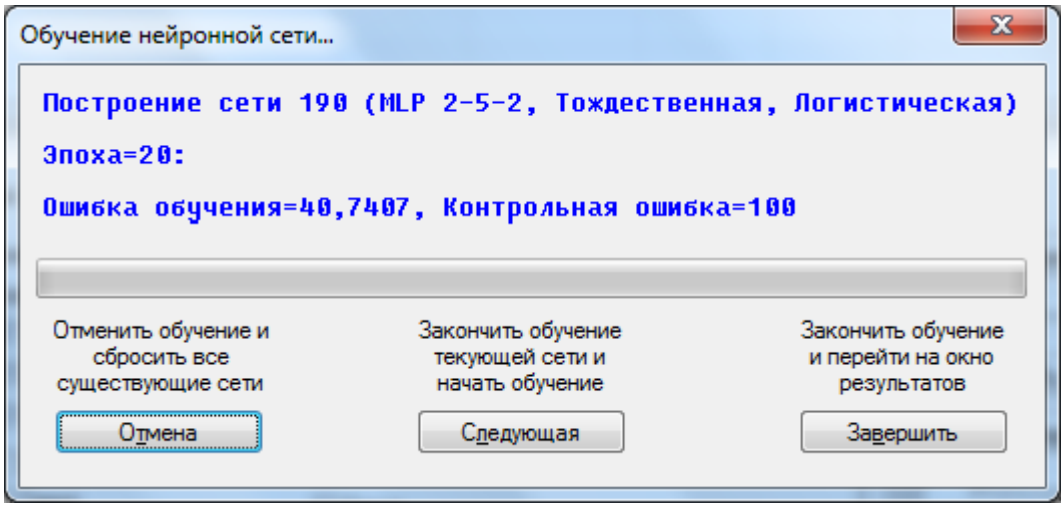


Рис. 9. Диалоговое окно Обучение нейронной сети

Программа автоматически переходит к следующему шагу.

Шаг 5. Анализ результатов. В окне результатов вы можете проанализировать полученные решения. Программа отберет лучшие сети и покажет качество решения.

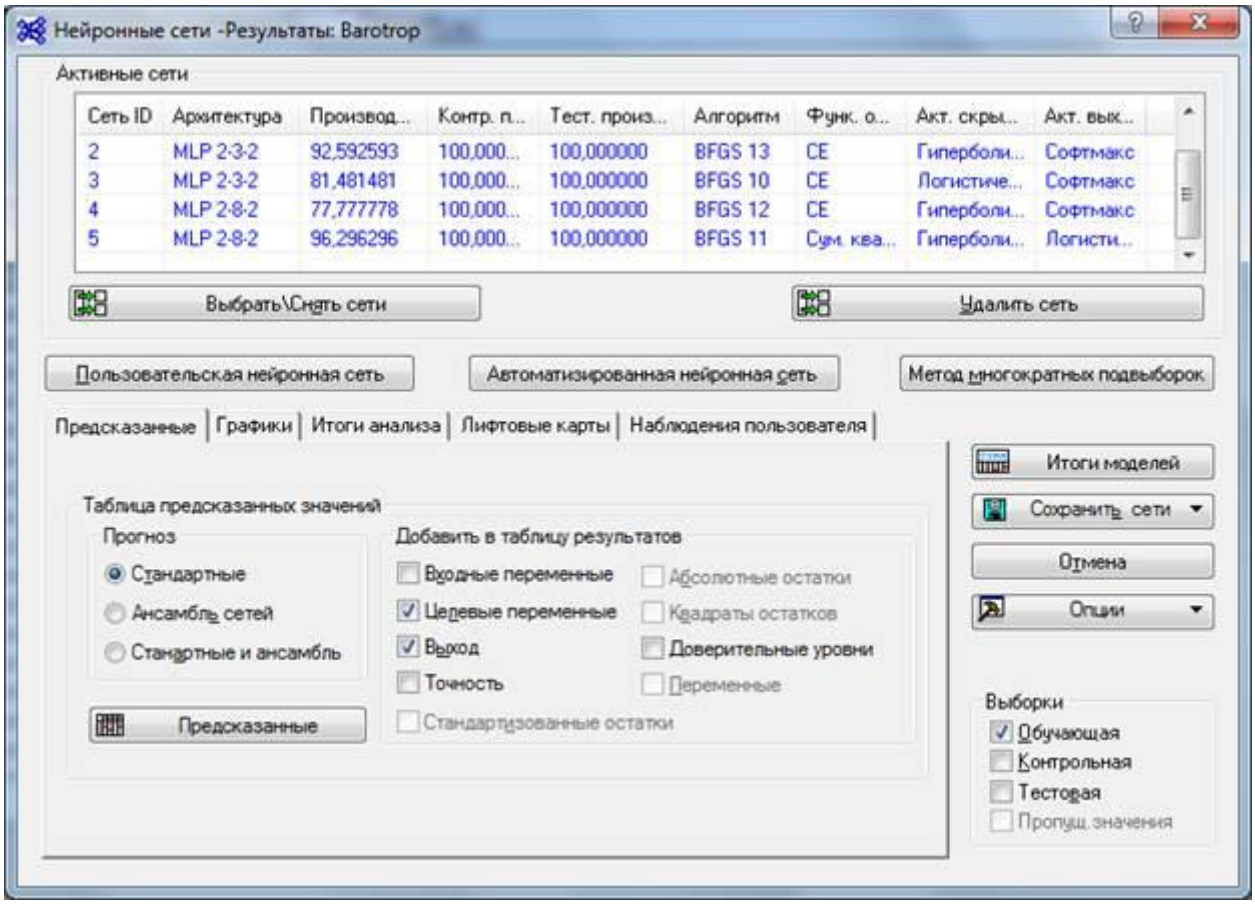


Рис. 10. Диалоговое окно Нейронные сети - Результаты - вкладка Предсказанные

Можно выбрать определенную сеть, лучшую на наш взгляд, с помощью кнопки *Выбрать/Снять сети*.

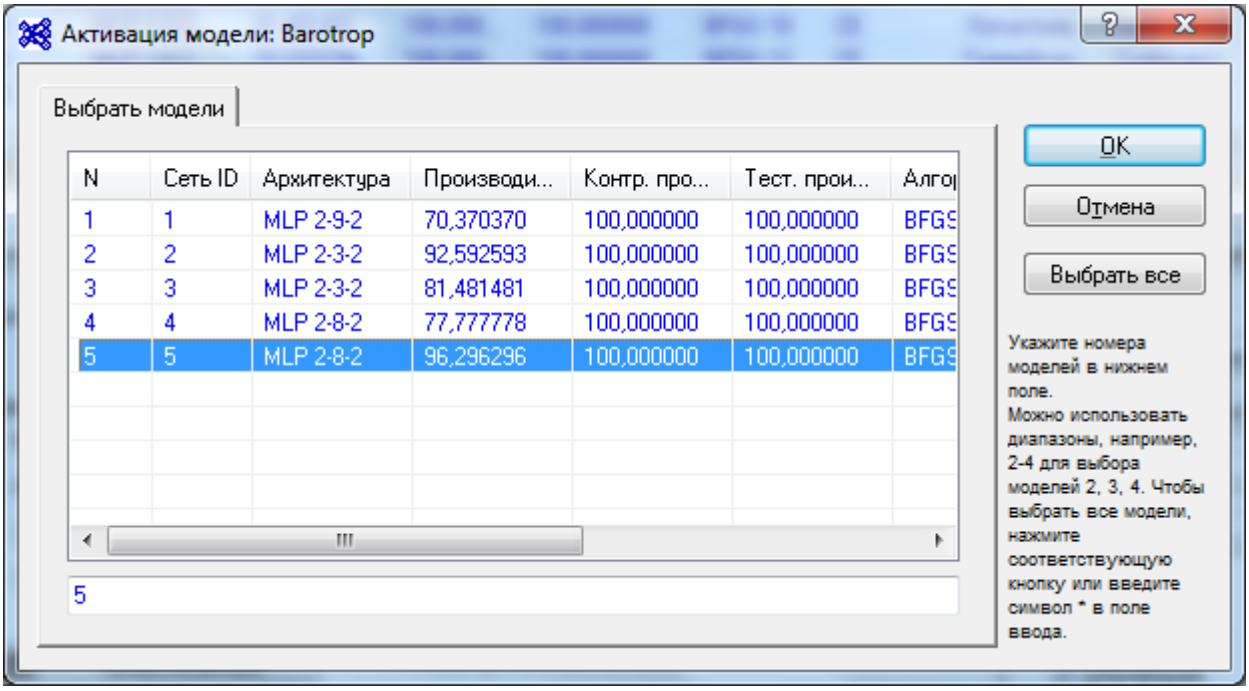


Рис. 11. Диалоговое окно Активация модели

Далее можно вывести всевозможные результаты и проанализировать их для выбранной сети.

Например, одним из способов проверки является сравнение наблюдаемых значений и предсказанных результатов. Сравнение наблюдаемых и предсказанных значений для выбранной сети, например, для обучающей и тестовой выборок.

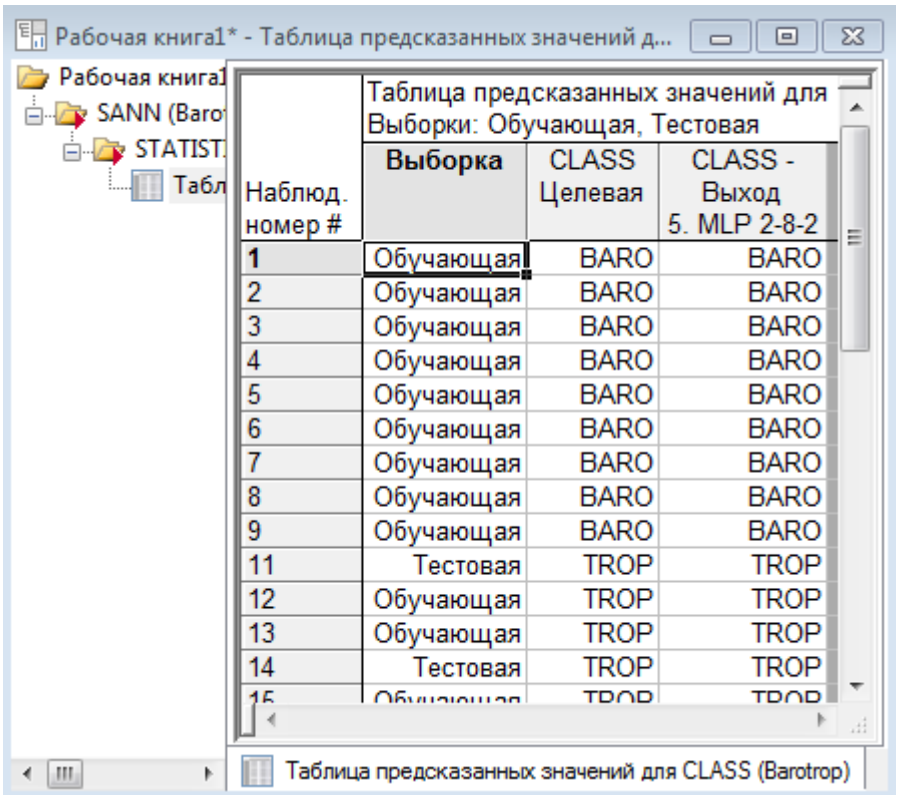


Рис. 12. Таблица наблюдаемых и предсказанных значений

Или посмотреть матрицу ошибок классификации на тестовой выборке:

CLASS (Итоги классификации) (Barotrop)
Выборки: Тестовая

		CLASS-BARO	CLASS-TROP	CLASS-Все
5.MLP 2-8-2	Все	1,0000	4,0000	5,0000
	Правильно	1,0000	4,0000	5,0000
	Неправильно	0,0000	0,0000	0,0000
	Правильно (%)	100,0000	100,0000	100,0000
	Неправильно (%)	0,0000	0,0000	0,0000

Рис. 13. Матрица классификаций

Шаг 6. Сохраните лучшие сети с целью дальнейшего использования, например, для автоматического построения прогнозов.

Для дальнейшего запуска, сети сохраняются в формате PMML.

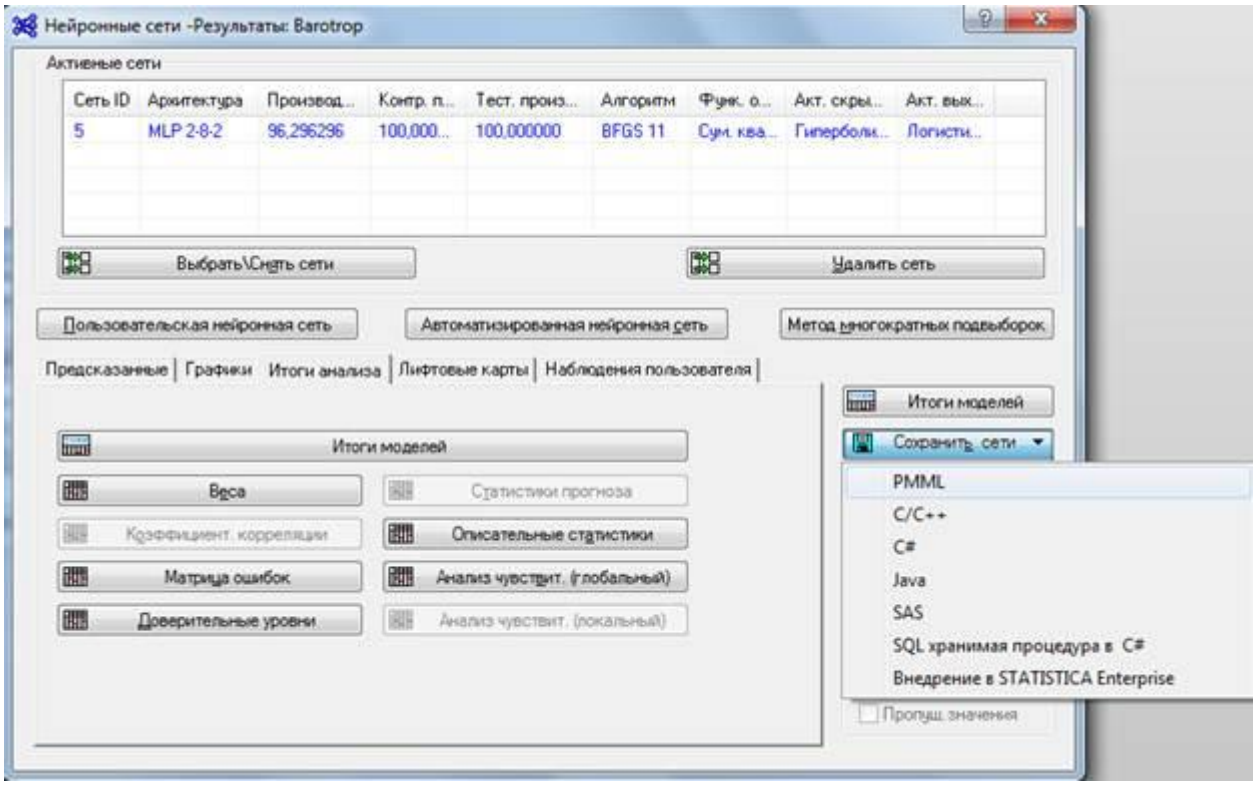


Рис. 14. Диалоговое окно Нейронные сети – Результаты – Сохранение сетей

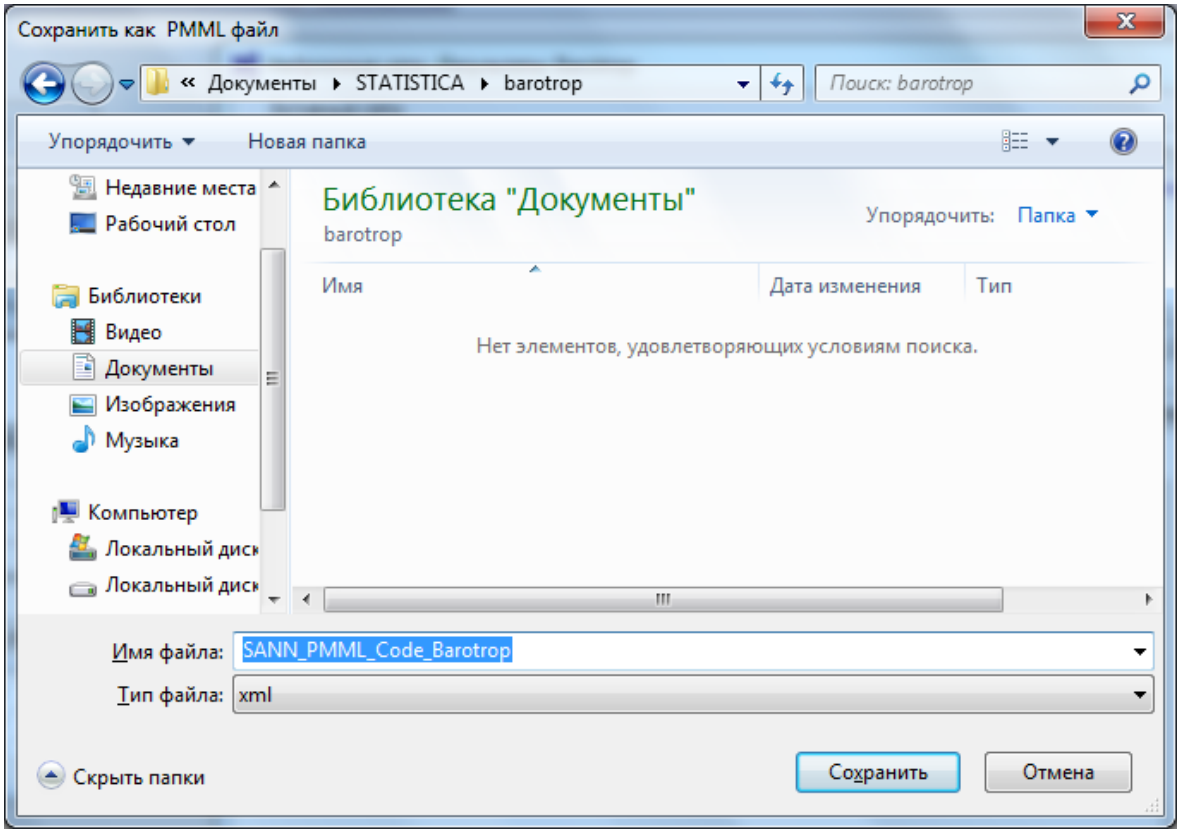


Рис. 15. Стандартное окно сохранения файла сети

Шаг 7. Запуск сохраненных моделей на новых данных. Итак загружаем новые данные, но чтобы переменные совпадали с переменными в моделях. Чтобы запустить модель на новых данных, можно на стартовой панели (рис. 1) выбрать опцию *Загрузить модели из предыдущих анализов* и нажить кнопу *Загрузить сети.*

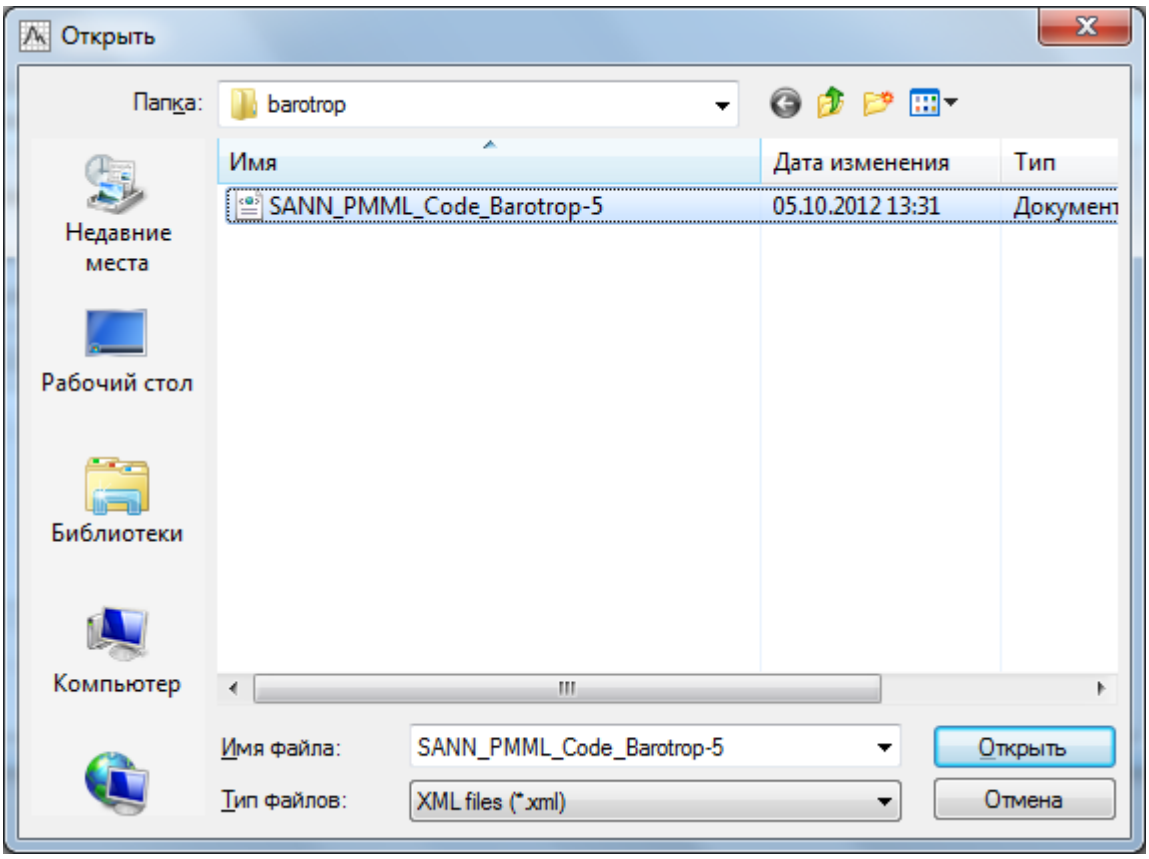


Рис. 16. Стандартное окно выбора файла сети

Получаем:

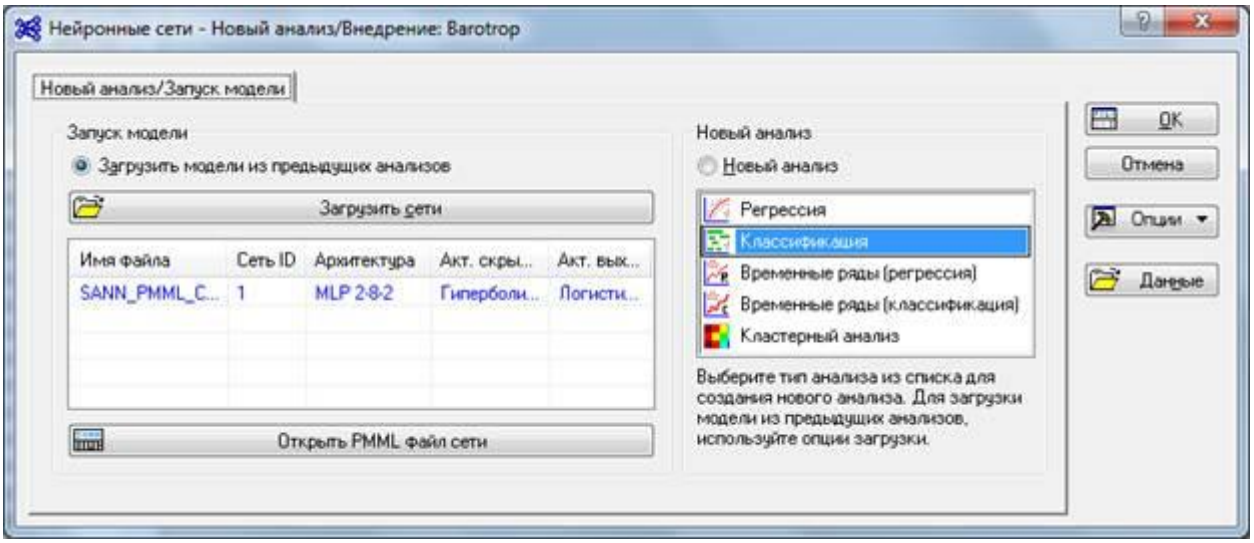


Рис. 17. Стартовая панель STATISTICA Automated Neural Networks (SANN)

После выбора необходимого файла, все настройки автоматически определяются, поэтому можно сразу переходить к окну результатов (нажимая два раза кнопку *OK*) и анализировать полученные результаты.

Задание

Пользуясь файлом апрель.xls постройте и обучите нейронную сеть предсказанию потребления электроэнергии.

1. Потребление представляет собой временной ряд, предсказание – регрессию.
2. Для прогноза не важен день недели, важно – является ли день выходным. Очистите выборку, удалив из нее все выходные дни. Сделать это можно проанализировав график посуточного потребления.
3. Обучите нейронную сеть, не учитывая фактор температуры. Проверьте результаты предсказания.
4. Обучите нейронную сеть, учитывая фактор температуры. Проверьте результаты предсказания.
5. Сравните результаты.

Практическая работа № 5
Задача классификации

1. Решение задачи классификации с помощью дерева решений.

Для построения дерева решений можно воспользоваться пакетом «Statistica» или платформой «Deductor»

Задание: Построить дерево решений для задачи определения исхода матча футбольной командой. Исходные данные:

Соперник	Играем	Лидеры	Дождь	Победа
Выше	Дома	На месте	Да	Нет
Выше	Дома	На месте	Нет	Да
Выше	Дома	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Нет	Да
Ниже	В гостях	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Да	Да
Выше	В гостях	На месте	Да	Нет
Ниже	В гостях	На месте	Нет	???

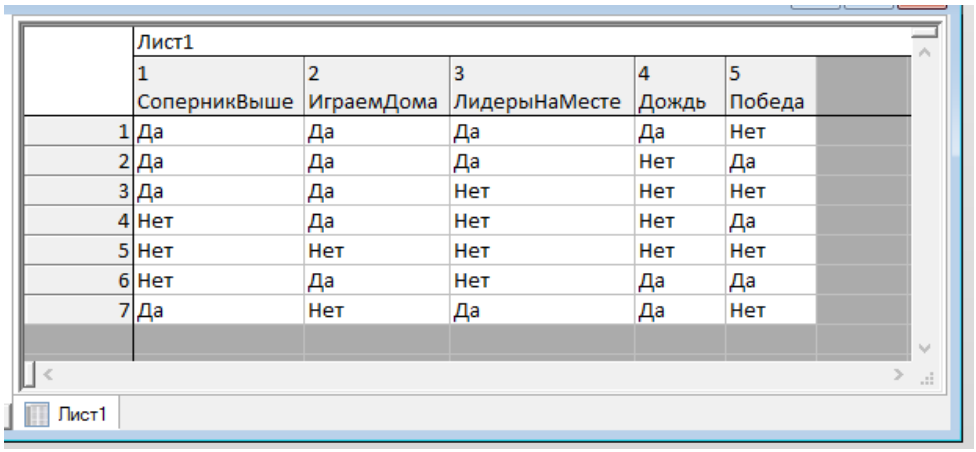
На первом этапе необходимо подготовить данные.

Для решения задачи в пакете «Statistica» данные можно представить в формате Excel.

Для решения задачи на платформе «Deductor» данные можно представить в формате txt с разделителем «табуляция». Получить такой файл можно сохранив файл Ecxel в соответствующем формате.

Решение задачи в пакете «Statistica»:

1) Ввод данных

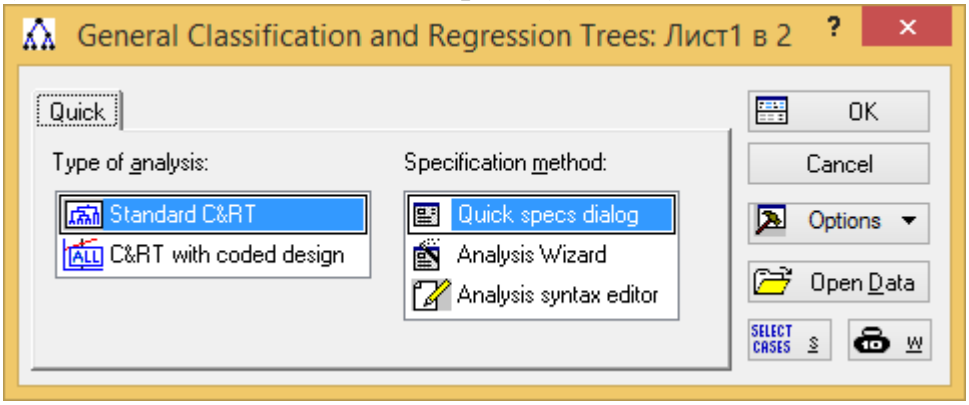


	Лист1				
	1 СоперникВыше	2 ИграемДома	3 ЛидерыНаМесте	4 Дождь	5 Победа
1	Да	Да	Да	Да	Нет
2	Да	Да	Да	Нет	Да
3	Да	Да	Нет	Нет	Нет
4	Нет	Да	Нет	Нет	Да
5	Нет	Нет	Нет	Нет	Нет
6	Нет	Да	Нет	Да	Да
7	Да	Нет	Да	Да	Нет

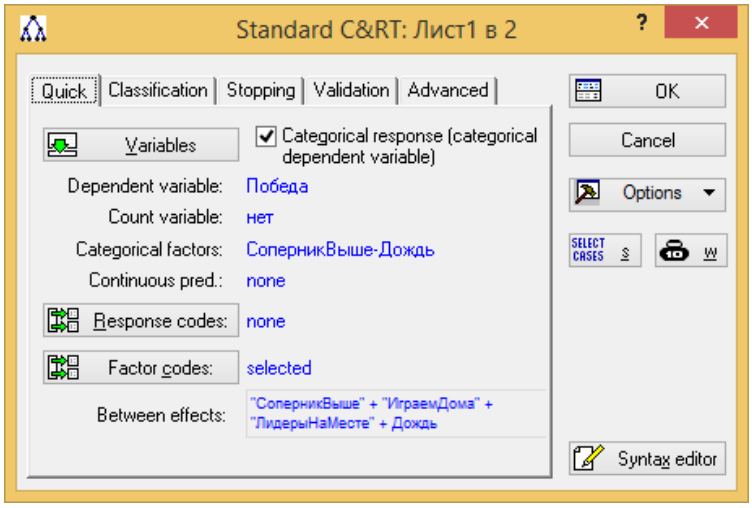
Лист1

2) Определение вида дерева

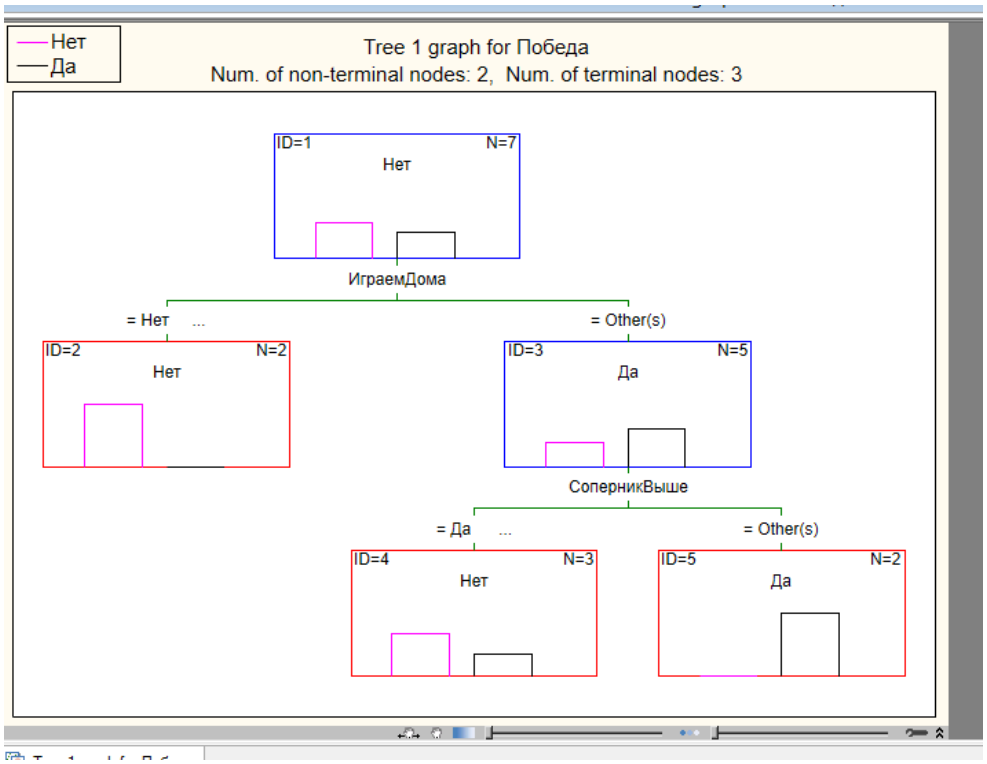
(Добыча данных – Общие деревья)



3) Определение переменных (Определяемая переменная – Победа, остальные – категориальные)

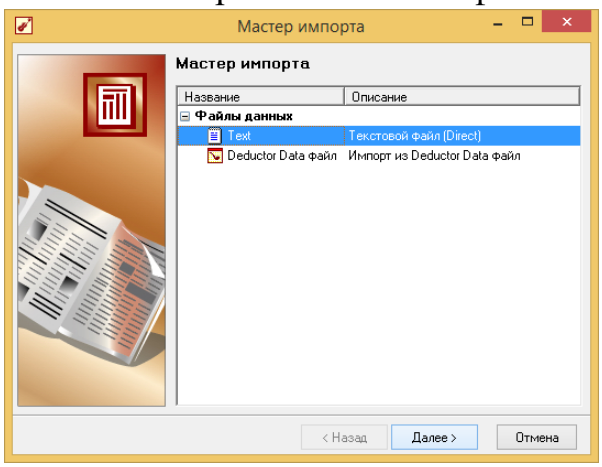


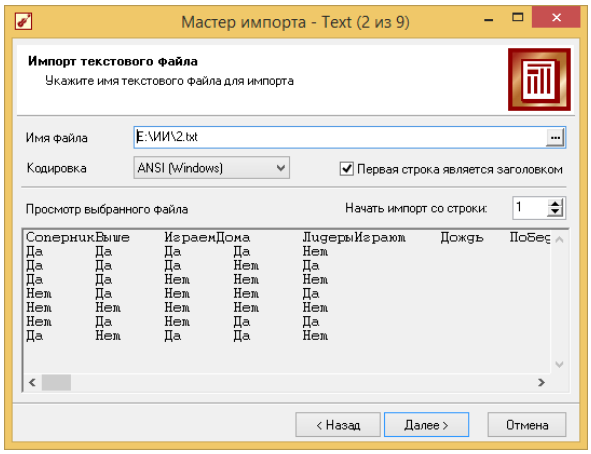
4) Вид решения:



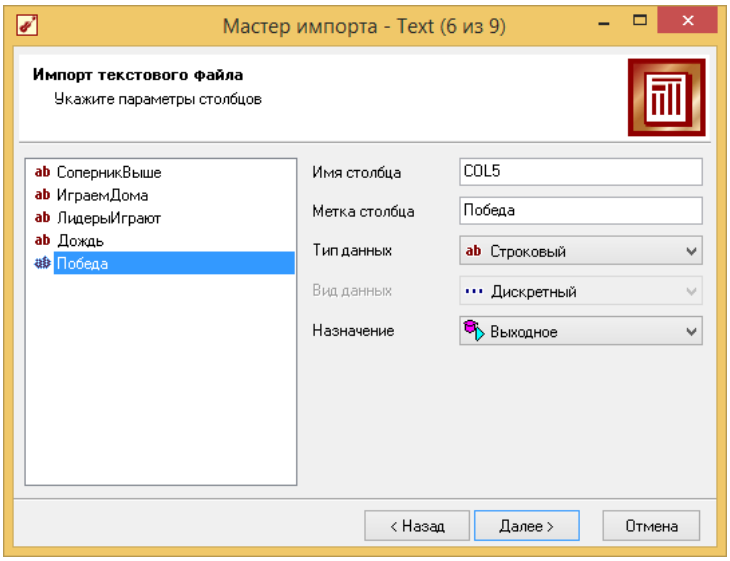
Решение задачи на платформе «Deductor»

1. Открыть мастер импорта  на панели инструментов
2. Выбрать текстовый файл

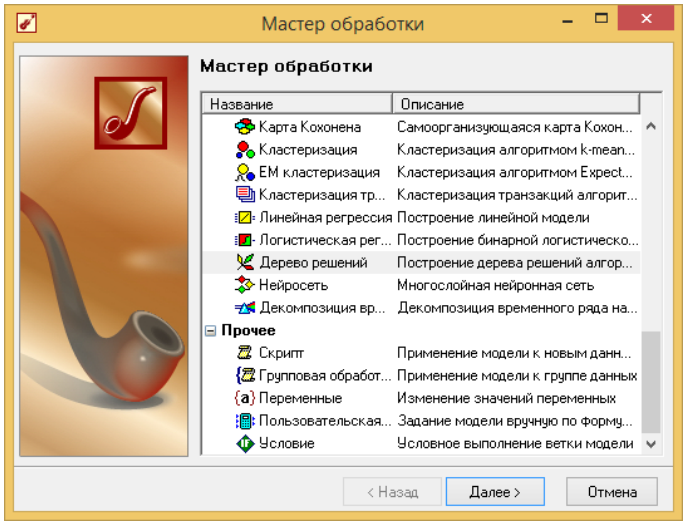




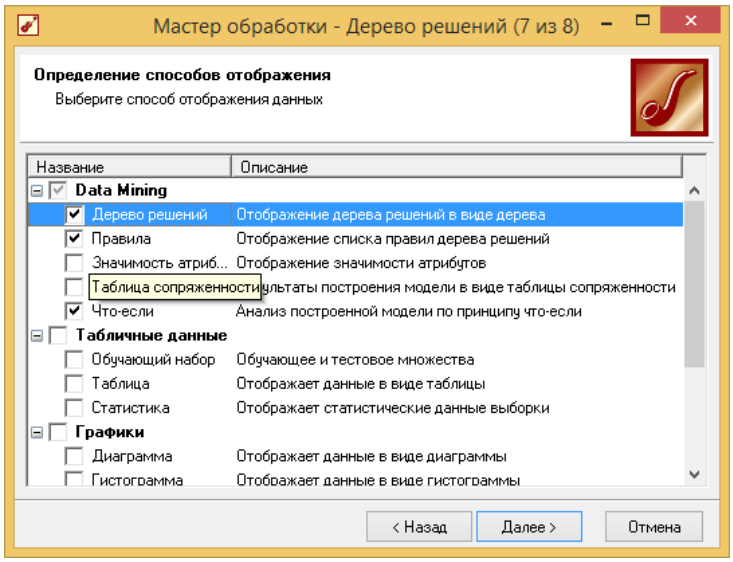
3. Задать назначения столбцов: Победа – выходное, остальные – входные



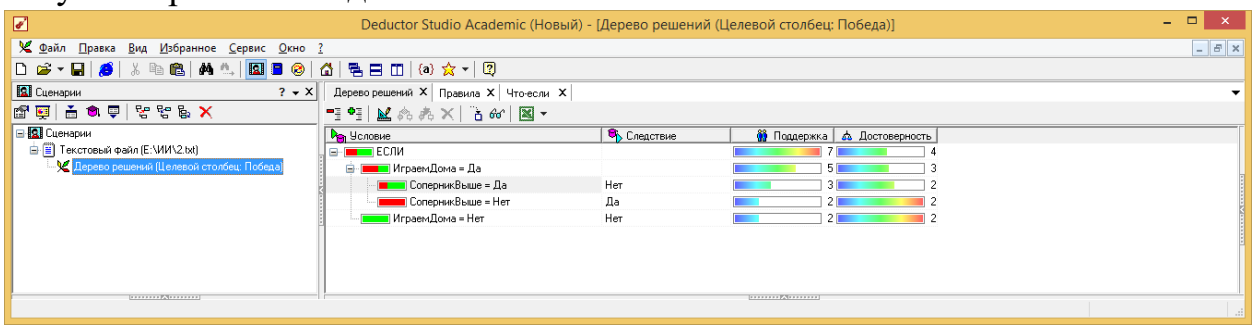
4. После успешного импорта открыть мастер обработки  и выбрать дерево решений:



5. Параметры построения оставить по умолчанию, в параметрах отображения указать:



6. Результат решения задачи:



С помощью анализа «Что-если» можно предсказать исход матча.

2 Классификация с помощью нейронных сетей

Рассмотрим классификацию на классическом примере *ирисов Фишера* (150 цветов трех сортов; каждый экземпляр описывается четырьмя параметрами: длина и ширина чашелистика, длина и ширина лепестка).

1. При обращении к Deductor Academic появляется окно (рис.1).

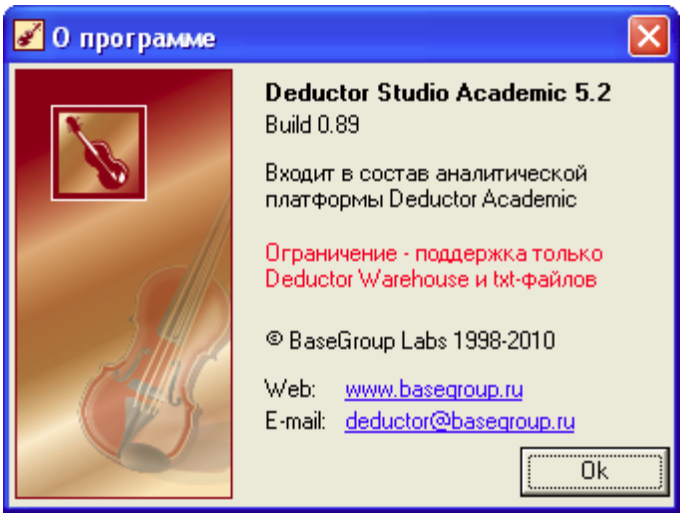


Рис.1 Исходное окно

После нажатия ОК появляется первое окно, предоставляющее очень мало возможностей. С помощью мастера импорта  можно загрузить в Deductor текстовый файл с исходными данными. В приложении приведены исходные данные, которые нужно перевести в текстовый файл и заменить обозначения сорта цветов на 1, 2, 3.

2.**Мастер импорта** в этой задаче содержит 10 шагов. На 2-ом шаге указывается имя файла с данными (рис.2).

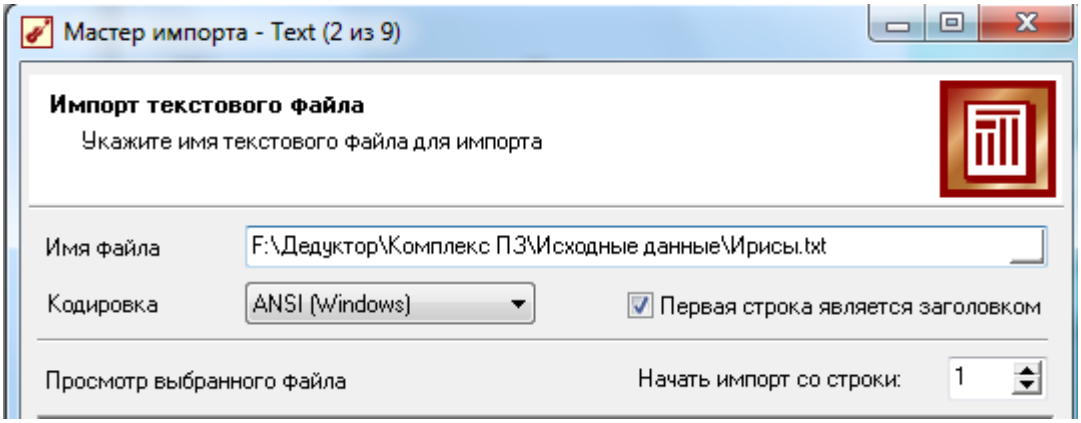


Рис.2 Ввод файла

На 3-ем и 4-ом шагах обычно ничего не меняем в установках. На следующем 6-ом шаге (5 шаг в программе пропущен) надо указать параметры каждого столбца данных, хотя здесь это не критично (рис.3).

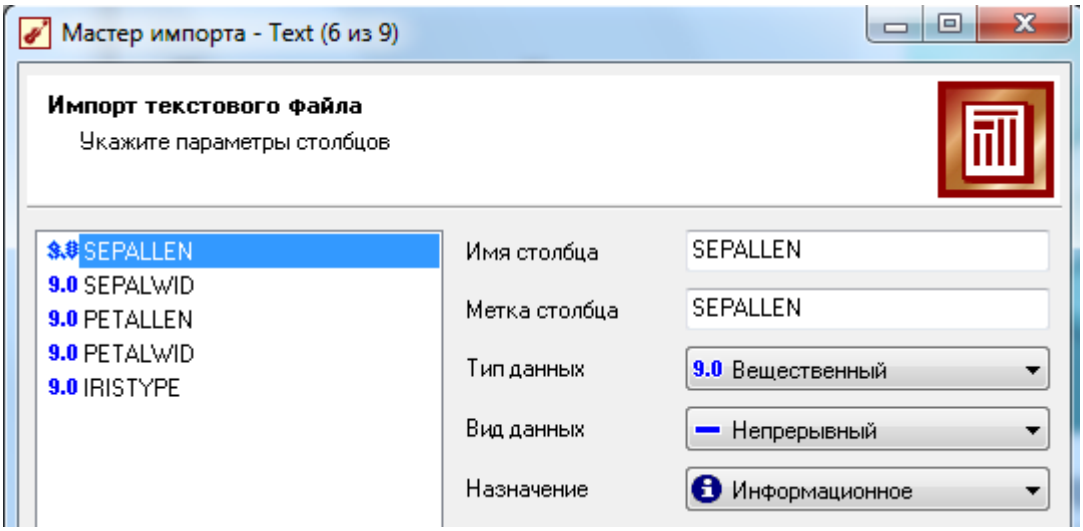


Рис.3 Указание параметров столбцов

Для последнего столбца – тип цветка – надо указать Дискретный вид данных. На 7-ом шаге нажать кнопку «Пуск». На 8-ом - указать вид, в котором отображаются исходные данные. Обычно таблицы данных достаточно. Можно отметить еще диаграмму размещения, тогда на 9-ом шаге отметить обозначения осей. После нажатия «Готово» на 10-ом шаге появляется поле результата работы мастера импорта (рис.4, 5).

Таблица | Диаграмма размещения

1 / 150

SEPALLEN	SEPALWID	PETALLEN	PETALWID	IRISTYPE
5	3,3	1,4	0,2	1
6,4	2,8	5,6	2,2	2
6,5	2,8	4,6	1,5	3
6,7	3,1	5,6	2,4	2
6,3	2,8	5,1	1,5	2
4,6	3,4	1,4	0,3	1
6,9	3,1	5,1	2,3	2
6,2	2,2	4,5	1,5	3
5,9	3,2	4,8	1,8	3
4,6	3,6	1	0,2	1
6,1	3	4,6	1,4	3

Рис.4 Таблица данных

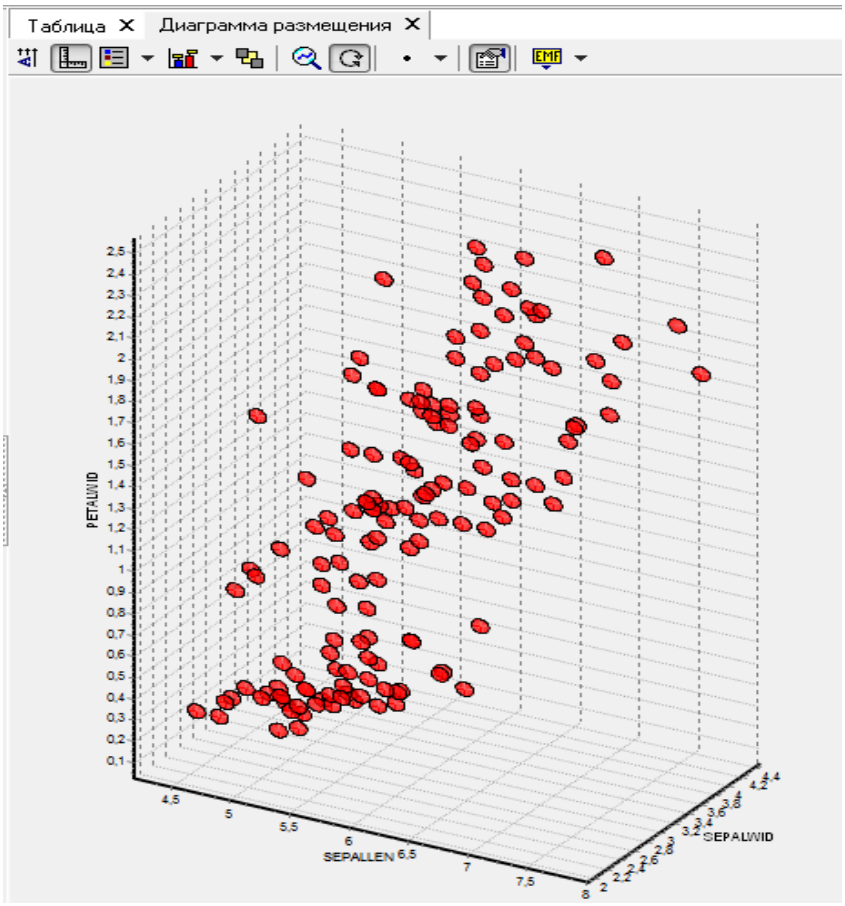


Рис.5 Диаграмма размещения 150 цветков

3.Мастер обработки вводится нажатием на выделенный блок

. Здесь мастер обработки имеет 9 шагов. Выделяется опция «Нейросеть» в окне Data Mining (рис.6).

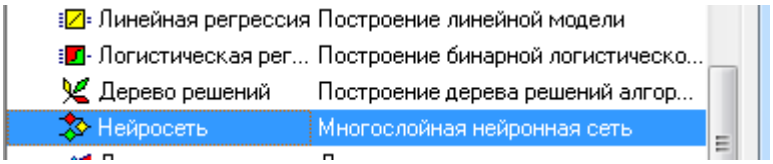


Рис.6 Выделение нейросети

На 2-ом шаге (важный шаг, поэтому нужно выполнить его внимательно) вначале настраиваются столбцы (рис.7).

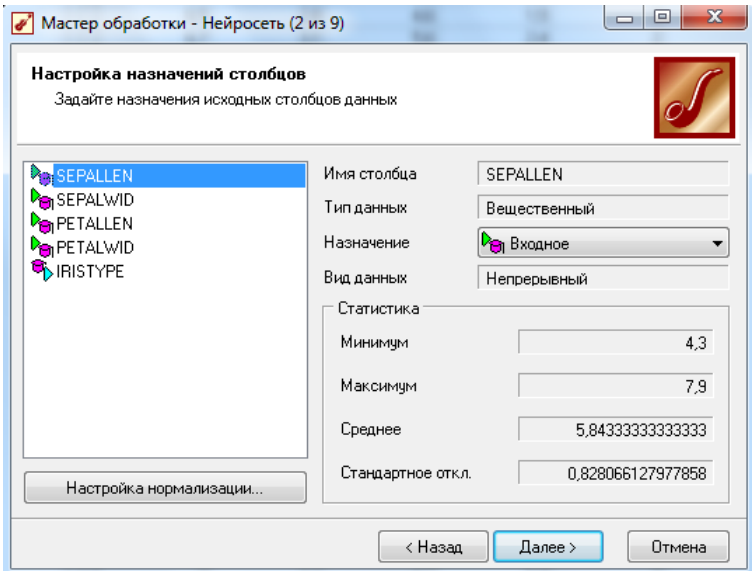


Рис.7 Настройка столбцов

Все входные переменные настраиваются одинаково. Выходной - иначе (рис.8).

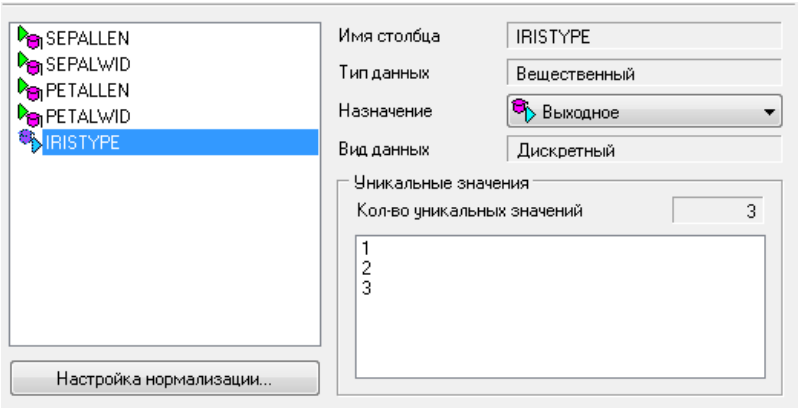


Рис.8 Настройка выходного параметра

Далее на этом же шаге нажать «Настройка нормализации» и выделить выходной параметр, в котором установить опцию «Уникальные значения» (рис.9).

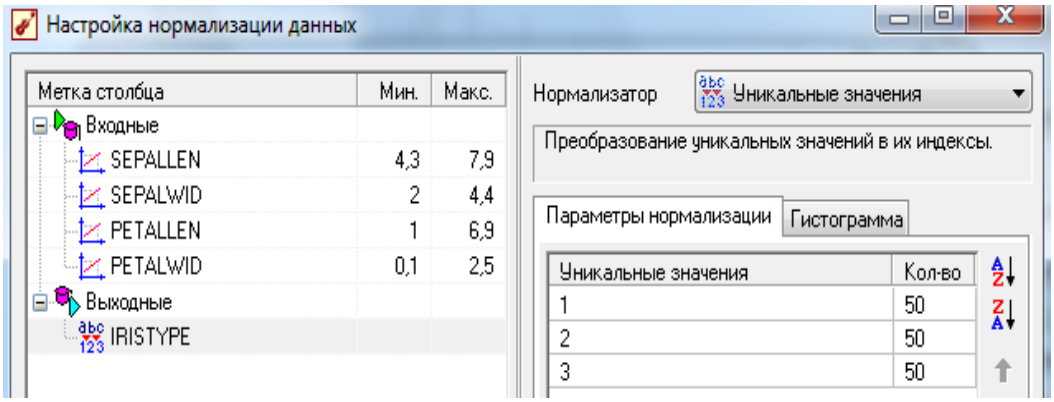


Рис.9 Настройка нормализации для выходного параметра

На 3-ем шаге разбить все данные на обучающее и тестовое множества в отношении 70% и 30% (105 и 45 наблюдений, соответственно, рис.10).

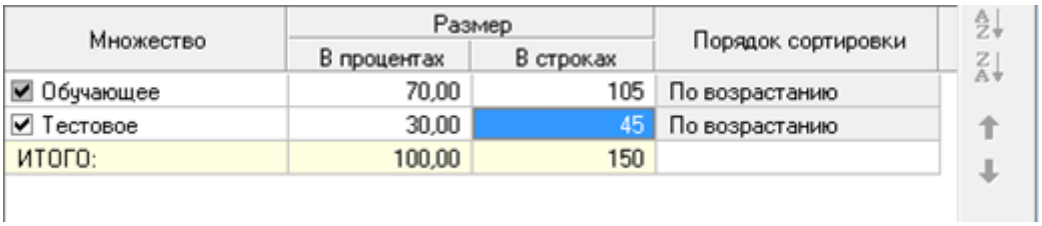


Рис.10 Обучающее и тестовое множества

На 4-ом шаге появляется структура нейронной сети, в которой увеличим до 4 число нейронов в скрытом слое (рис.11).

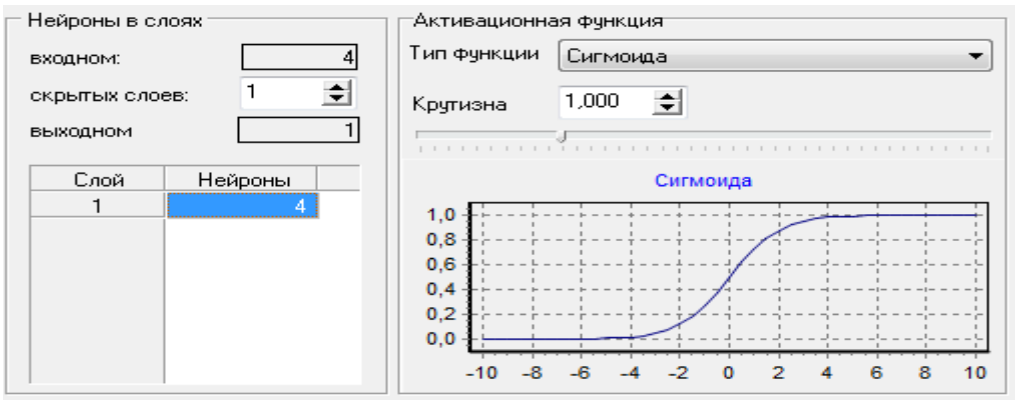


Рис.11 Структура нейронной сети

На 5 –ом шаге выбираем метод обучения НС - Back Prop., на 6-ом - все параметры принимаем по умолчанию, на 7- ом - обучаем НС, нажав «Пуск» (рис.12).

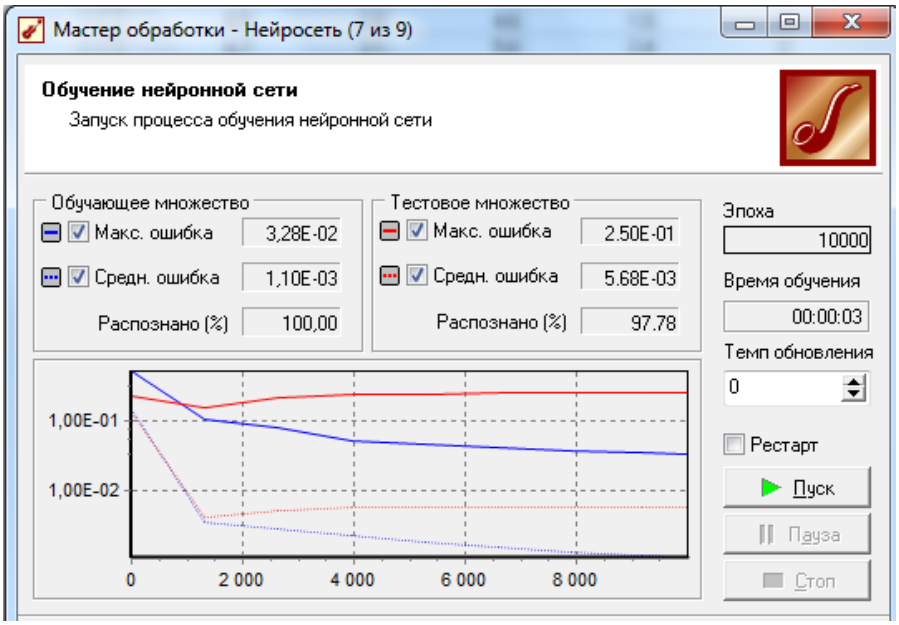


Рис.12 Обучение НС

На 8-ом - указываем те характеристики, которые желательно получить и анализируем итоги (рис.13 -15).

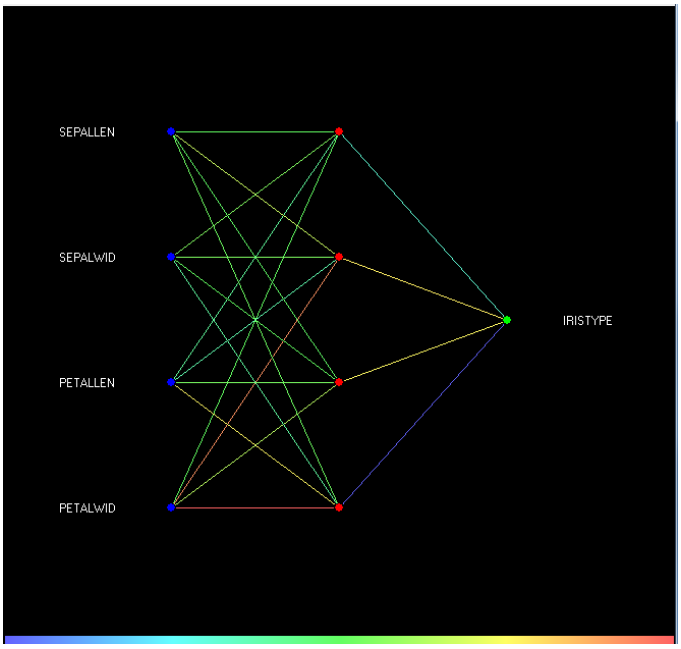


Рис.13 Нейросеть

Граф нейросети | Таблица сопряженности | Что-ес.

IRISTYPE

	Классифицировано			
Фактически	1	2	3	Итого
1	50			50
2		43	7	50
3		1	49	50
Итого	50	44	56	150

Рис.14 Таблица сопряженности

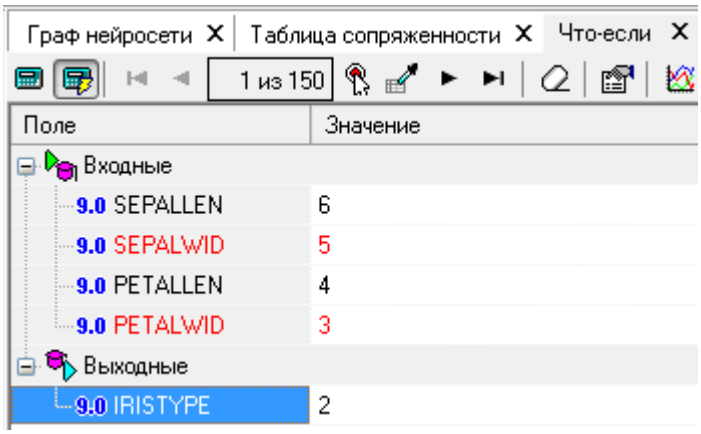


Рис.15 Проверка новых данных

Как видно из рис.13-15, сеть имеет структуру 4-4-1 (как и планировалось); цветы одного типа Setoza - распознаются безошибочно, а другие сорта - с ошибками. С помощью механизма «Что , если» можно определить сорт нового цветка: на рис.15 при вводе новых значений параметров нейросеть определяет тип цветка.

Задание: решить задачу классификации ирисов Фишера

Приложение

SEPALLEN	SEPALWID	PETALLEN	PETALWID	IRISTYPE
5	3,3	1,4	0,2	SETOSA
6,4	2,8	5,6	2,2	VIRGINIC
6,5	2,8	4,6	1,5	VERSICOL
6,7	3,1	5,6	2,4	VIRGINIC
6,3	2,8	5,1	1,5	VIRGINIC
4,6	3,4	1,4	0,3	SETOSA
6,9	3,1	5,1	2,3	VIRGINIC
6,2	2,2	4,5	1,5	VERSICOL
5,9	3,2	4,8	1,8	VERSICOL
4,6	3,6	1	0,2	SETOSA
6,1	3	4,6	1,4	VERSICOL
6	2,7	5,1	1,6	VERSICOL
6,5	3	5,2	2	VIRGINIC
5,6	2,5	3,9	1,1	VERSICOL

6,5	3	5,5	1,8	VIRGINIC
5,8	2,7	5,1	1,9	VIRGINIC
6,8	3,2	5,9	2,3	VIRGINIC
5,1	3,3	1,7	0,5	SETOSA
5,7	2,8	4,5	1,3	VERSICOL
6,2	3,4	5,4	2,3	VIRGINIC
7,7	3,8	6,7	2,2	VIRGINIC
6,3	3,3	4,7	1,6	VERSICOL
6,7	3,3	5,7	2,5	VIRGINIC
7,6	3	6,6	2,1	VIRGINIC
4,9	2,5	4,5	1,7	VIRGINIC
5,5	3,5	1,3	0,2	SETOSA
6,7	3	5,2	2,3	VIRGINIC
7	3,2	4,7	1,4	VERSICOL
6,4	3,2	4,5	1,5	VERSICOL
6,1	2,8	4	1,3	VERSICOL
4,8	3,1	1,6	0,2	SETOSA
5,9	3	5,1	1,8	VIRGINIC
5,5	2,4	3,8	1,1	VERSICOL
6,3	2,5	5	1,9	VIRGINIC
6,4	3,2	5,3	2,3	VIRGINIC
5,2	3,4	1,4	0,2	SETOSA
4,9	3,6	1,4	0,1	SETOSA
5,4	3	4,5	1,5	VERSICOL
7,9	3,8	6,4	2	VIRGINIC
4,4	3,2	1,3	0,2	SETOSA
6,7	3,3	5,7	2,1	VIRGINIC
5	3,5	1,6	0,6	SETOSA
5,8	2,6	4	1,2	VERSICOL
4,4	3	1,3	0,2	SETOSA
7,7	2,8	6,7	2	VIRGINIC
6,3	2,7	4,9	1,8	VIRGINIC
4,7	3,2	1,6	0,2	SETOSA
5,5	2,6	4,4	1,2	VERSICOL
7,2	3,2	6	1,8	VIRGINIC
4,8	3	1,4	0,3	SETOSA
5,1	3,8	1,6	0,2	SETOSA
6,1	3	4,9	1,8	VIRGINIC
4,8	3,4	1,9	0,2	SETOSA
5	3	1,6	0,2	SETOSA
5	3,2	1,2	0,2	SETOSA

6,1	2,6	5,6	1,4	VIRGINIC
6,4	2,8	5,6	2,1	VIRGINIC
4,3	3	1,1	0,1	SETOSA
5,8	4	1,2	0,2	SETOSA
5,1	3,8	1,9	0,4	SETOSA
6,7	3,1	4,4	1,4	VERSICOL
6,2	2,8	4,8	1,8	VIRGINIC
4,9	3	1,4	0,2	SETOSA
5,1	3,5	1,4	0,2	SETOSA
5,6	3	4,5	1,5	VERSICOL
5,8	2,7	4,1	1	VERSICOL
5	3,4	1,6	0,4	SETOSA
4,6	3,2	1,4	0,2	SETOSA
6	2,9	4,5	1,5	VERSICOL
5,7	2,6	3,5	1	VERSICOL
5,7	4,4	1,5	0,4	SETOSA
5	3,6	1,4	0,2	SETOSA
7,7	3	6,1	2,3	VIRGINIC
6,3	3,4	5,6	2,4	VIRGINIC
5,8	2,7	5,1	1,9	VIRGINIC
5,7	2,9	4,2	1,3	VERSICOL
7,2	3	5,8	1,6	VIRGINIC
5,4	3,4	1,5	0,4	SETOSA
5,2	4,1	1,5	0,1	SETOSA
7,1	3	5,9	2,1	VIRGINIC
6,4	3,1	5,5	1,8	VIRGINIC
6	3	4,8	1,8	VIRGINIC
6,3	2,9	5,6	1,8	VIRGINIC
4,9	2,4	3,3	1	VERSICOL
5,6	2,7	4,2	1,3	VERSICOL
5,7	3	4,2	1,2	VERSICOL
5,5	4,2	1,4	0,2	SETOSA
4,9	3,1	1,5	0,2	SETOSA
7,7	2,6	6,9	2,3	VIRGINIC
6	2,2	5	1,5	VIRGINIC
5,4	3,9	1,7	0,4	SETOSA
6,6	2,9	4,6	1,3	VERSICOL
5,2	2,7	3,9	1,4	VERSICOL
6	3,4	4,5	1,6	VERSICOL
5	3,4	1,5	0,2	SETOSA
4,4	2,9	1,4	0,2	SETOSA

5	2	3,5	1	VERSICOL
5,5	2,4	3,7	1	VERSICOL
5,8	2,7	3,9	1,2	VERSICOL
4,7	3,2	1,3	0,2	SETOSA
4,6	3,1	1,5	0,2	SETOSA
6,9	3,2	5,7	2,3	VIRGINIC
6,2	2,9	4,3	1,3	VERSICOL
7,4	2,8	6,1	1,9	VIRGINIC
5,9	3	4,2	1,5	VERSICOL
5,1	3,4	1,5	0,2	SETOSA
5	3,5	1,3	0,3	SETOSA
5,6	2,8	4,9	2	VIRGINIC
6	2,2	4	1	VERSICOL
7,3	2,9	6,3	1,8	VIRGINIC
6,7	2,5	5,8	1,8	VIRGINIC
4,9	3,1	1,5	0,1	SETOSA
6,7	3,1	4,7	1,5	VERSICOL
6,3	2,3	4,4	1,3	VERSICOL
5,4	3,7	1,5	0,2	SETOSA
5,6	3	4,1	1,3	VERSICOL
6,3	2,5	4,9	1,5	VERSICOL
6,1	2,8	4,7	1,2	VERSICOL
6,4	2,9	4,3	1,3	VERSICOL
5,1	2,5	3	1,1	VERSICOL
5,7	2,8	4,1	1,3	VERSICOL
6,5	3	5,8	2,2	VIRGINIC
6,9	3,1	5,4	2,1	VIRGINIC
5,4	3,9	1,3	0,4	SETOSA
5,1	3,5	1,4	0,3	SETOSA
7,2	3,6	6,1	2,5	VIRGINIC
6,5	3,2	5,1	2	VIRGINIC
6,1	2,9	4,7	1,4	VERSICOL
5,6	2,9	3,6	1,3	VERSICOL
6,9	3,1	4,9	1,5	VERSICOL
6,4	2,7	5,3	1,9	VIRGINIC
6,8	3	5,5	2,1	VIRGINIC
5,5	2,5	4	1,3	VERSICOL
4,8	3,4	1,6	0,2	SETOSA
4,8	3	1,4	0,1	SETOSA
4,5	2,3	1,3	0,3	SETOSA
5,7	2,5	5	2	VIRGINIC

5,7	3,8	1,7	0,3	SETOSA
5,1	3,8	1,5	0,3	SETOSA
5,5	2,3	4	1,3	VERSICOL
6,6	3	4,4	1,4	VERSICOL
6,8	2,8	4,8	1,4	VERSICOL
5,4	3,4	1,7	0,2	SETOSA
5,1	3,7	1,5	0,4	SETOSA
5,2	3,5	1,5	0,2	SETOSA
5,8	2,8	5,1	2,4	VIRGINIC
6,7	3	5	1,7	VERSICOL
6,3	3,3	6	2,5	VIRGINIC
5,3	3,7	1,5	0,2	SETOSA
5	2,3	3,3	1	VERSICOL

Задание: Выполнить классификацию отказов профилактического обслуживания машин

Набор данных состоит из 10 000 точек данных, хранящихся в виде строк с 14 функциями в столбцах.

- UID: уникальный идентификатор в диапазоне от 1 до 10000
- productID: состоит из букв L, M или H для низкого (50 % всех продуктов), среднего (30 %) и высокого (20 %) вариантов качества продукта и серийного номера для конкретного варианта.
- температура воздуха [K]: генерируется с использованием процесса случайного блуждания, позже нормализованного до стандартного отклонения 2 K около 300 K.
- температура процесса [K]: генерируется с использованием процесса случайного блуждания, нормализованного до стандартного отклонения 1 K, добавленного к температуре воздуха плюс 10 K.
- скорость вращения [об/мин]: рассчитывается исходя из мощности 2860 Вт с наложением нормально распределенного шума
- крутящий момент [Нм]: значения крутящего момента обычно распределяются около 40 Нм с Ïf = 10 Нм и без отрицательных значений.

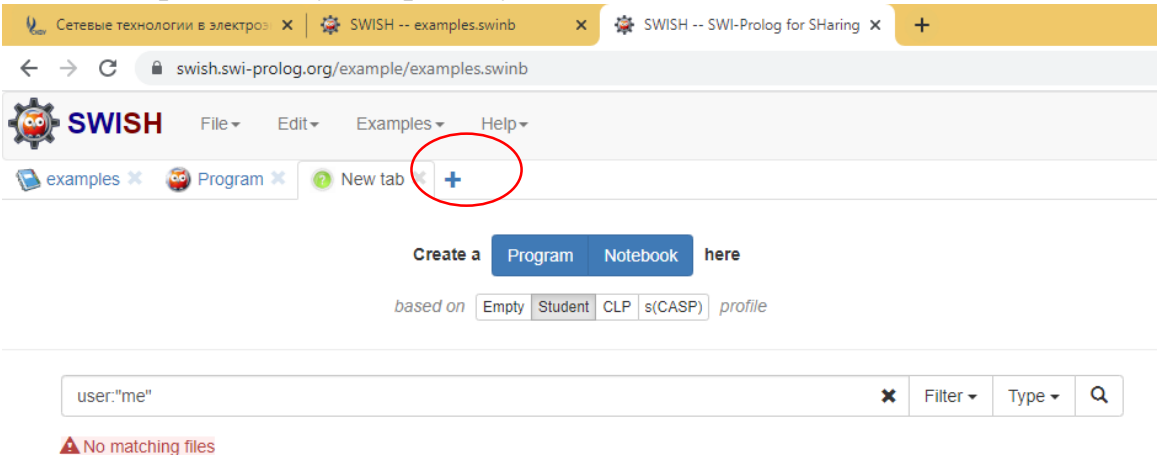
- износ инструмента [мин]: Варианты качества H/M/L добавляют 5/3/2 минут износа инструмента к используемому в процессе инструменту.
- И метка «сбой машины», которая указывает, является ли машина неисправной в этой конкретной точке данных для любого из следующих режимов отказа.
- Цель: провал или нет
- Тип отказа : Тип отказа

Практическая работа № 6-7. Логическое программирование. Логика предикатов.

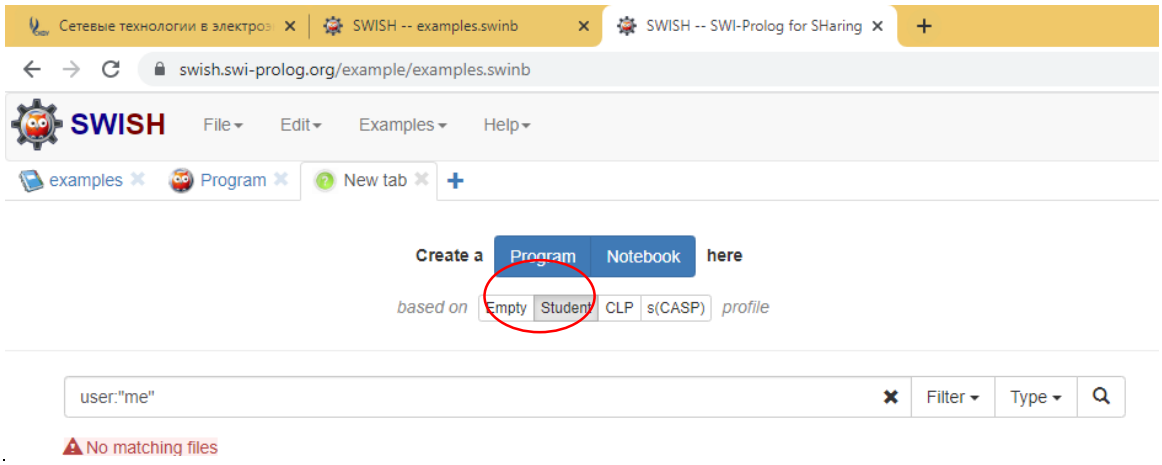
Методические указания по выполнению работы

Работа выполняется в онлайн-сервисе SWI-Prolog https://swish.swi-prolog.org/example/examples.swinb

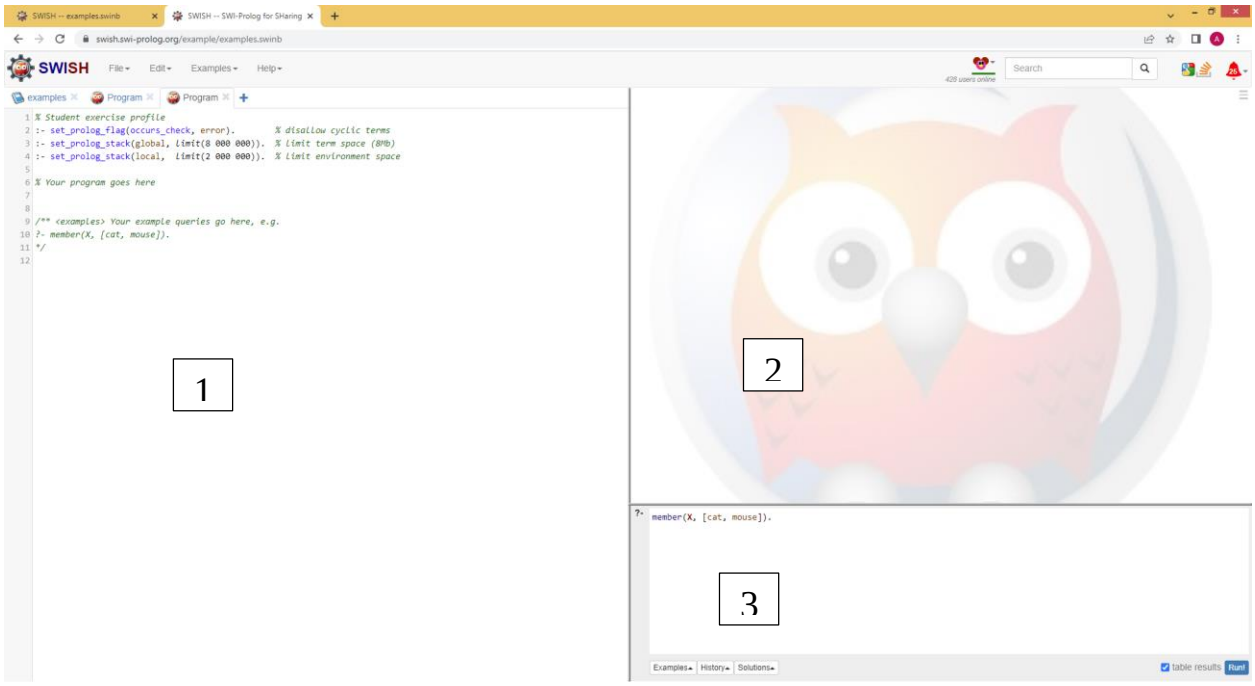
1. Откройте новую страницу:



2. Выберите создание программы



3. В появившемся окне:
   1 – редактор текста программы
   2 – поле вывода результата
   3 – поле вывода запроса



Задания

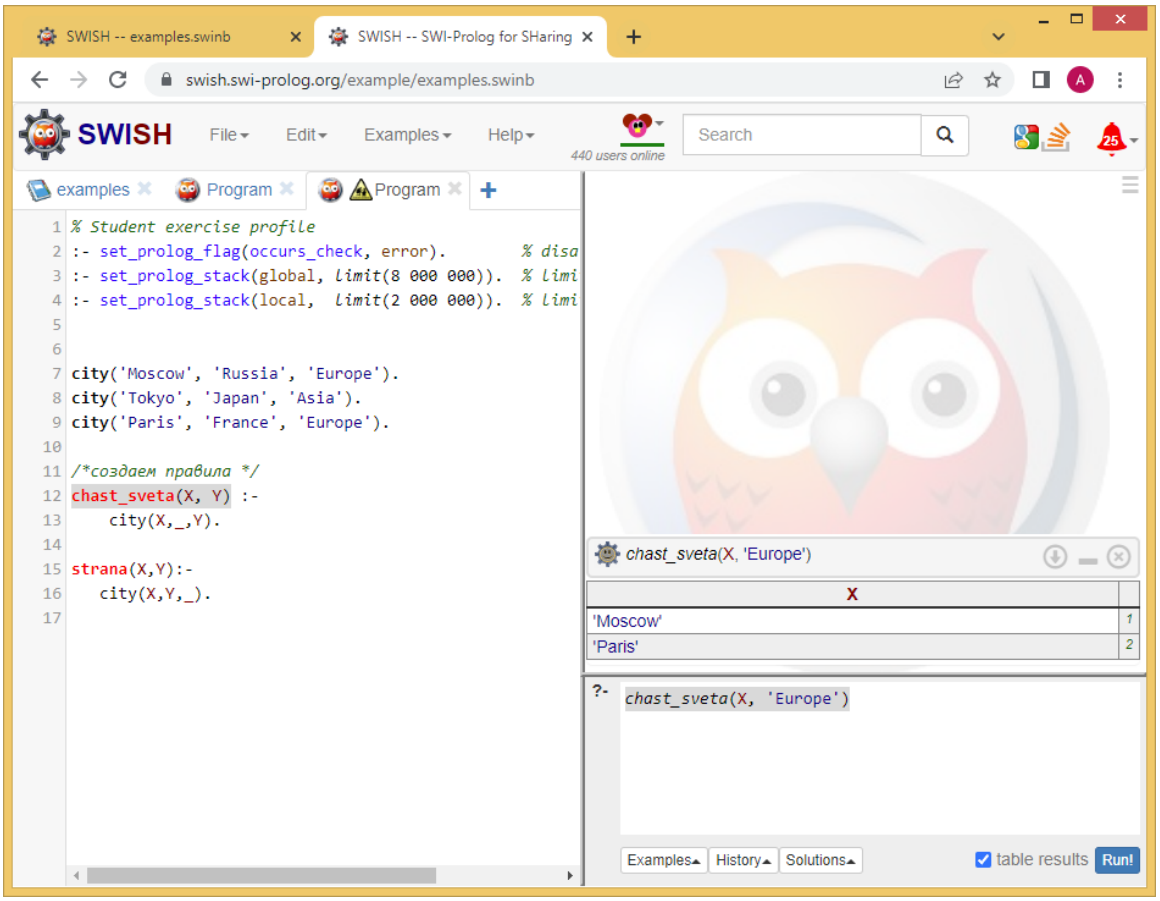
1. Создайте программу, определяющую страну или часть света, к которой принадлежит заданный город.

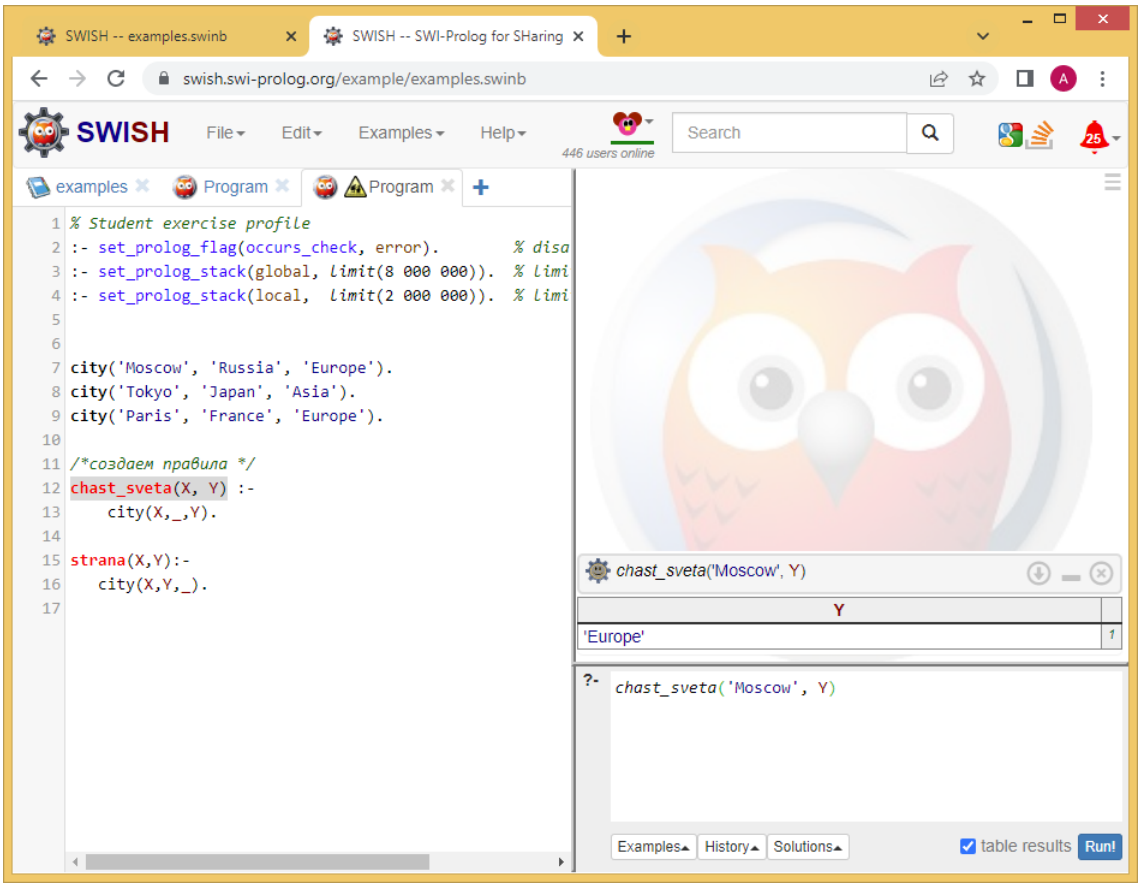
Решение:

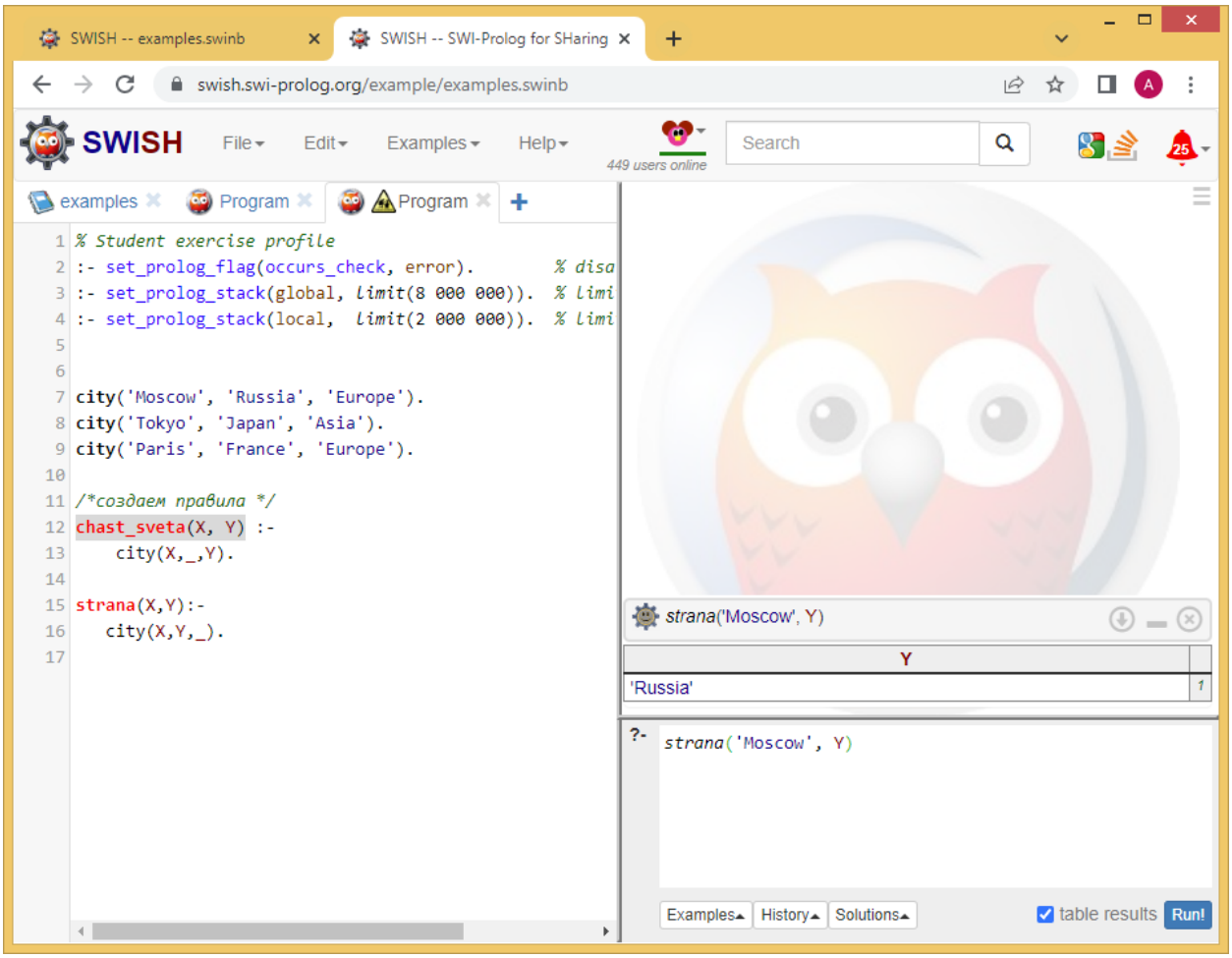
```
/*Создаем базу знаний (фактов)*/
city('Moscow', 'Russia', 'Europe').
city('Tokyo', 'Japan', 'Asia').
city('Paris', 'France', 'Europe').

/*создаем правила*/
chast_sveta(X, Y) :-
    city(X,_,Y).
strana(X,Y):-
   city(X,Y,_).
```

Результаты решения:


SWISH -- examples.swinb
SWISH -- SWI-Prolog for SHaring
swish.swi-prolog.org/example/examples.swinb
SWISH
File
Edit
Examples
Help
440 users online
Search
examples
Program
Program
% Student exercise profile
:- set_prolog_flag(occurs_check, error). % disa
:- set_prolog_stack(global, limit(8 000 000)). % limi
:- set_prolog_stack(local, limit(2 000 000)). % limi
city('Moscow', 'Russia', 'Europe').
city('Tokyo', 'Japan', 'Asia').
city('Paris', 'France', 'Europe').
/*создаем правила */
chast_sveta(X, Y) :-
city(X,_,Y).
strana(X,Y):-
city(X,Y,_).
chast_sveta(X, 'Europe')
X
'Moscow'
1
'Paris'
2
?-
chast_sveta(X, 'Europe')
Examples
History
Solutions
table results
Run!


SWISH -- examples.swinb
SWISH -- SWI-Prolog for SHaring
swish.swi-prolog.org/example/examples.swinb
SWISH
File
Edit
Examples
Help
446 users online
Search
examples
Program
Program
% Student exercise profile
:- set_prolog_flag(occurs_check, error). % disa
:- set_prolog_stack(global, limit(8 000 000)). % limi
:- set_prolog_stack(local, limit(2 000 000)). % limi
city('Moscow', 'Russia', 'Europe').
city('Tokyo', 'Japan', 'Asia').
city('Paris', 'France', 'Europe').
/*создаем правила */
chast_sveta(X, Y) :-
city(X,_,Y).
strana(X,Y):-
city(X,Y,_).
chast_sveta('Moscow', Y)
Y
'Europe'
1
?-
chast_sveta('Moscow', Y)
Examples
History
Solutions
table results
Run!



2. Дополните базу знаний фактами
3. Дополните базу знаний и правил таким образом, чтобы можно было определять полушарие, в котором находится город. Необходимо предусмотреть возможность вывода всех стран заданного полушария, всех стран заданной части света.

4. Создайте программу, позволяющую отследить родственные связи между заданными людьми.

   Решение:

```
/*Создаем базу знаний (фактов)*/
mother('Ann', 'Kate').
mother('Kate', 'Nancy').
mother('Kate', 'Joe').
mother('Samanta', 'Nick').
mother('Lea', 'John').
```

```
father('Jim', 'Kate').
father('Luce', 'Nancy').
father('Luce', 'Joe').
father('Luce', 'Nick').
father('Colin', 'John').

/*создаем правила*/
children(X,Y):-
   mother(X,Y); father(X,Y).

grandmother(X,Y):-
   mother(X,Z), mother(Z,Y).

grandsons(X,Y):-
   (mother(Y,Z), mother(Z,X));
   (father(Y,Z), father(Z,X));
   (mother(Y,Z), father(Z,X));
   (father(Y,Z), mother(Z,X)).
```

Проверяем запросы:

```
?-grandsons('Nancy', X)
?-children(X, 'Joe')
?-children('Kate', Y)
```

5. Решите логическую задачу: В автомобильных гонках три первых места заняли Алеша, Петя и Коля. Какое место занял каждый из них, если Петя занял не второе и не третье место, а Коля - не третье?

Решение

```
/* База имен */
имя(алеша).
имя(петя).
имя(коля).

/* База призовых мест */
```

```
место(первое).
место(второе).
место(третье).

/* Устанавливаем взаимно-однозначное соответствие
между базами данных, X - элемент из базы данных имен,
Y - элемент из базы данных занятых мест */

/* Петя занял не второе и не третье место */
соответствие(X, Y) :- имя(X), X=петя,
 место(Y), not(Y=второе), not(Y=третье).

/* Коля занял не третье место */
соответствие(X, Y) :- имя(X), X=коля,
 место(Y), not(Y=третье).

соответствие(X, Y) :- имя(X), X=алеша, место(Y).

/* У всех ребят разные места */
решение(X1,Y1,X2,Y2,X3,Y3) :-
 X1=петя, соответствие(X1,Y1),
 X2=коля, соответствие(X2,Y2),
 X3=алеша, соответствие(X3,Y3),
 Y1\=Y2, Y2\=Y3, Y1\=Y3.
```

Проверяем решение

```
?-решение(X1,Y1,X2,Y2,X3,Y3).
```

6. Решите логическую задачу

Витя, Юра и Миша сидели на скамейке. В каком порядке они сидели, если известно, что Миша сидел слева от Юры, а Витя слева от Миши.

Витя, Юра и Миша сидели на скамейке. В каком порядке они сидели, если известно, что Миша сидел слева от Юры, а Витя слева от Миши.

В условии задачи перечисляются объекты одного типа, связанные между собой. Заполним базу данных:

```
/* Миша сидел слева от Юры */
слева(юра, миша).

/* Витя сидел слева от Миши */
слева(миша, витя).
```

Правило для установления следования объектов друг за другом будет таким:

```
/* Объекты X, Y и Z образуют ряд,
если X слева от Y и Y слева от Z */

ряд(X, Y, Z) :- слева(Y, X), слева(Z, Y).
```

Количество аргументов в голове данного правила равно количеству объектов в задаче. Запрос

```
?- ряд(X, Y, Z).
```

даст нам решение задачи.

Задание самостоятельное

Решите следующие логические задачи с помощью интерпретатора Пролога.

1. Трое ребят вышли гулять с собакой, кошкой и хомячком. Известно, что Петя не любит кошек и живет в одном подъезде с хозяйкой хомячка. Лена дружит с Таней, гуляющей с кошкой. Определить, с каким животным гулял каждый из детей.
2. Витя, Юра и Миша сидели на скамейке. В каком порядке они сидели, если известно, что Юра сидел слева от Миши и справа от Вити.

Практическая работа № 8-9. Нечеткая логика. Математические основы.

Пусть E – универсальное множество, X-элемент E, а R – некоторое свойство. Обычное (четкое) подмножество A универсального множества E, элементы которого удовлетворяют свойству R, определяется как множество упорядоченных пар $A = \{\mu_A(x)/x\}$, где $\mu_A(x)$ – характеристическая функция принадлежности (или просто функция принадлежности), принимающая значения в некотором вполне упорядоченном множестве M (например, M=**[**0,1**]**). Функция принадлежности указывает степень (или уровень) принадлежности элемента x подмножеству A. Множество M называют множеством принадлежностей. Если M=**{**0,1**}**, то нечеткое подмножество **A** может рассматриваться как обычное или четкое множество.

Нечеткое множество отличается от обычного тем, что для элементов X из подмножества E нет однозначного ответа (да - нет) относительно свойства R.

В связи с этим нечеткое подмножество A универсального множества E определяется как множество упорядоченных пар.

Функция принадлежности $\mu_A(x)$ указывает степень (уровень) принадлежности элемента X подмножеству A.

Пример:

Пусть $E= \{X_1,X_2,X_3,X_4,X_5\}$, M=[0; 1]

A – нечеткое множество, для которого заданы функции принадлежности.

$\mu_A(X_1) = 0,3$; $\mu_A(X_2) = 0$; $\mu_A(X_3) = 1$; $\mu_A(X_4) = 0,5$; $\mu_A(X_5) = 0,9$.

Тогда

$A = \{0,3/x_1; 0/x_2; 1/x_3; 0,5/x_4; 0,9/x_5\}$ или

$A = 0,3/x_1 + 0/x_2; 1/x_3; 0,5/x_4; 0,9/x_5$

Основные характеристики нечетких множеств:

Величина sup $\mu_A(x)$ называется высотой нечеткого множества A.

Нечеткое множество A – нормально, если его высота равно 1 , т.е. верхняя граница его функции принадлежности равна 1.

При sup $\mu_A(x) < 1$ – нечеткое множество субнормальное.

Если функция принадлежности = 0, то нечеткое множество пусто.

Если функция принадлежности равна 1, $\mu_A(x) = 1$ только для одного элемента X, то нечеткое множество унимодально.

Если подмножество содержит функции принадлежности > 0, то оно называется A – носителем нечеткого подмножества.

Если функция принадлежности равна 0, $0 = \mu_A$, то элементы X, для которого выполняется это условие называется точками перехода.

Нечеткая и лингвистические переменные. Формы представления функции принадлежности и нечеткий вывод

Для нечетких множителей вводите понятие нечетко лингвистических переменных. Нечеткая переменная описывается набором (α, χ, A), где α - название переменной, χ - универсальное множество (область α), A-нечеткие множества на x, описывающие ограничение ($M_A(x)$ на значение нечеткой переменной α).

Значение нечетной лингвистической переменной могут быть нечеткие переменные, описывается набором (B, T, X, G, P) B- лингвистическое, T-множества ее значений, представляющих собой наименование нечетких переменных- X, областью определения каждой из которых является множество - X. Множество T называется базовым лингвистической переменной. G - синтаксическая процедура, позволяющая оперировать элементами - T (например генерировать новые значения), M - синтаксическая процедура, позволяющая превратить каждое новое значение лингвистической переменной, образующая процедурой G в нечеткого переменного, т.е. сформировать соответствующие нечеткие множество.

Существуют множества типовых форм для задания функций принадлежности. Наиболее распространенное получили Треугольная, Трапецеидальная и Гауссова функция принадлежности.

М функция принадлежности определяется тройкой чисел (a,b,c) и ее значения в точке x вычисляется согласно выражению.

Треугольная функция принадлежности.

$$MF(x) = \begin{cases} 1 - \dfrac{b - x}{b - a}, a \le x \le b \\ 1 - \dfrac{x - b}{c - b}, b \le x \le c \\ 0, \text{в остальных случаях} x \end{cases}$$

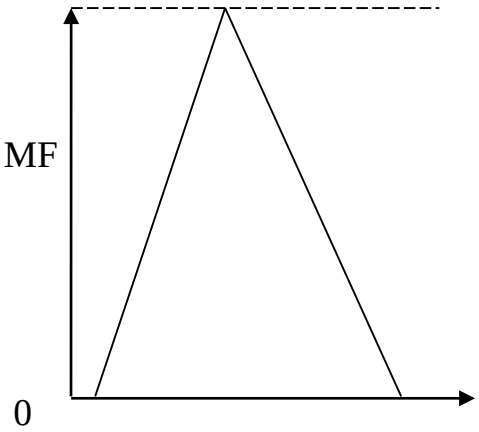
При b-a=c-b имеет случай симметричной треугольной функции принадлежности, может быть однозначно задана двумя параметрами из трех.

Для задания трапецеидальная функции принадлежности наибольшая четверка чисел (a,b,c,d).

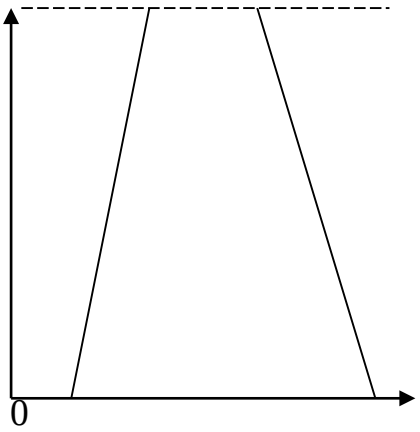
Трапецеидальная функция принадлежности.

$$MF(x) = \begin{cases} 1 - \dfrac{b - x}{b - a}, a \le x \le b \\ 1, b \le x \le c \\ 1 - \dfrac{x - c}{d - c}, c \le x \le d \\ 0, \text{в остальных случаях} \end{cases}$$

1) Треугольная



2) Трапецеидальная



Совокупность функции принадлежности для каждого термина базового множества T обычно изображаются вместе на одном графике.

Решение типовых задач

Пример 1. Построить функции принадлежности термов "низкий", "средний", "высокий", используемых для лингвистической оценки переменной "рост мужчины". Результаты опроса пяти экспертов приведены в табл.1.

Рассматриваются два метода построения функций принадлежности. Первый метод основан на статистической обработке мнений группы экспертов. Второй метод базируется на парных сравнениях, выполняемых одним экспертом.

Метод статистической обработки экспертной информации

Каждый эксперт заполняет опросник, в котором указывает свое мнение о наличии у элементов u_i, i =1,n свойств нечеткого множества l_j (j =1,m). Опросник имеет следующий вид:

	u_1	u_2	…	u_n
l_1				
l_2				
…				
l_m				

Введем следующие обозначения: K - количество экспертов; $b^k_{j,i}$ - мнение k-го эксперта о наличии у элемента u_i свойств нечеткого множества l_j, k=1,..K , j=1,..m и i =1,..n. Будем считать, что экспертные оценки бинарные, т.е.: $b^k_{j,i}$ ={0,1}, где 1 (0) указывает на наличие (отсутствие) у элемента u_i свойств нечеткого множества l_j. По результатам опроса экспертов, степени принадлежности нечеткому множеству l_j (j 1,m) рассчитываются следующим образом:

$$\mu_{l_j}(u_i) = \frac{1}{K} \sum_{k=\overline{1,K}} b^k_{j,i}\ ,\ i = \overline{1,n}\ .$$

Таблица 1. - Результаты опроса экспертов

k	термы	[160, 165)	[165, 170)	[170, 175)	[175, 180)	[180, 185)	[185, 190)	[190, 195)	[195, 200)

Эксперт 1	низкий	1	1	1	0	0	0	0	0
	средний	0	0	1	1	1	0	0	0
	высокий	0	0	0	0	0	1	1	1
Эксперт 2	низкий	1	1	1	0	0	0	0	0
	средний	0	0	1	1	0	0	0	0
	высокий	0	0	0	0	1	1	1	1
Эксперт 3	низкий	1	0	0	0	0	0	0	0
	средний	0	1	1	1	1	1	0	0
	высокий	0	0	0	0	0	1	1	1
Эксперт 4	низкий	1	1	1	0	0	0	0	0
	средний	0	0	0	1	1	1	0	0
	высокий	0	0	0	0	0	0	1	1
Эксперт 5	низкий	1	1	0	0	0	0	0	0
	средний	0	1	1	1	0	0	0	0
	высокий	0	0	0	1	1	1	1	1

Результаты обработки экспертных мнений представлены в таблице 2. Числа над линией - это количество голосов, отданных экспертами за принадлежность нечеткому множеству соответствующего элемента универсального множества. Числа под линией - степени принадлежности, рассчитанные по формуле (1). Графики функций принадлежностей показаны на рис. 1.

Таблица 2 - Результаты обработки мнений экспертов

термы	**[160, 165)**	**[165, 170)**	**[170, 175)**	**[175, 180)**	**[180, 185)**	**[185, 190)**	**[190, 195)**	**[195, 200)**
низкий	5	4	3	0	0	0	0	0
	1	0.8	0.6	0	0	0	0	0
средний	0	2	4	5	3	2	0	0
	0	0.4	0.8	1	0.6	0.4	0	0
высокий	0	0	0	1	2	4	5	5
	0	0	0	0.2	0.4	0.8	1	1

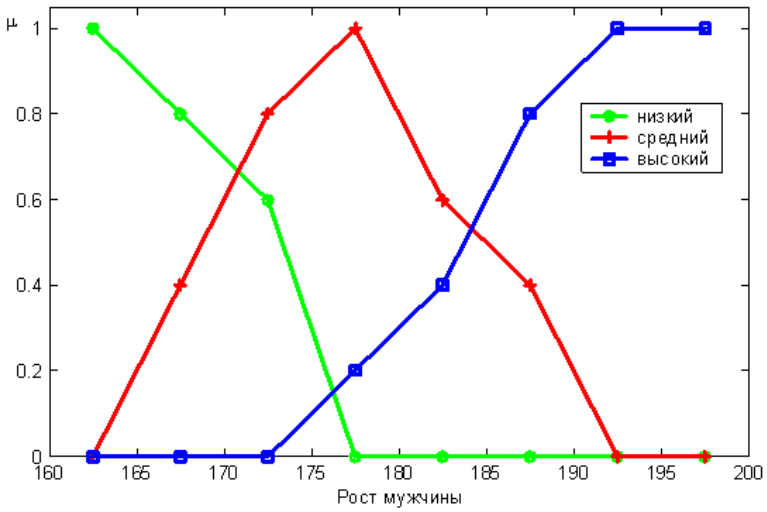


Рисунок 1. Функции принадлежности нечетких множеств

Пример 2. Построить функции принадлежности на основе парных сравнений термов "низкий", "средний", "высокий", используемых для лингвистической оценки переменной "рост мужчины".

Пример 3. Построить функцию принадлежности нечеткого множества "высокий мужчина" на универсальном множестве {170, 175, 180, 185, 190, 195}, если известны такие экспертные парные сравнения:

- *абсолютное преимущество 195 над 170;*
- *явное преимущество 195 над 175;*

• *существенное преимущество 195 над 180;*

• *слабое преимущество 195 над 185;*

• *отсутствует преимущество 195 над 190.*

Справочный материал для решения задачи

Исходной информацией для построения функций принадлежности являются экспертные парные сравнения. Для каждой пары элементов универсального множества эксперт оценивает преимущество одного элемента над другим по отношению к свойству нечеткого множества. Парные сравнения удобно представлять следующей матрицей:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

где a_{ij} - уровень преимущества элемента u_i над u_j ($i, j = 1,n$), определяемый по девятибальной шкале Саати:

1 - если *отсутствует преимущество* элемента u_i над элементом u_j;

3 - если имеется *слабое преимущество* u_i над u_j;

5 - если имеется *существенное преимущество* u_i над u_j;

7 - если имеется *явное преимущество* u_i над u_j;

9 - если имеется *абсолютное преимущество* u_i над u_j;

2,4,6,8 - *промежуточные* сравнительные оценки.

Матрица парных сравнений является диагональной и обратно симметричной

Степени принадлежности принимаются равными соответствующим координатам собственного вектора $W = (w_1, w_2, \dots, w_n)$ матрицы парных сравнений:

$$\mu(u_i) = w_i, \; i = \overline{1,n}. \tag{2}$$

Собственный вектор находится из следующей системы уравнений:

$$\begin{cases} A \cdot W = \lambda_{max} \cdot W \\ w_1 + w_2 + \ldots + w_n = 1 \end{cases}, \quad (3)$$

где λ_{max} - максимальное собственное значение матрицы А.

Решение:

Приведенным экспертным высказываниям соответствует такая матрица парных сравнений:

$$A = \begin{array}{c} \\ 170 \\ 175 \\ 180 \\ 185 \\ 190 \\ 195 \end{array} \begin{array}{c} \begin{array}{cccccc} 170 & 175 & 180 & 185 & 190 & 195 \end{array} \\ \begin{bmatrix} 1 & 1/2 & 1/4 & 1/6 & 1/8 & 1/9 \\ 2 & 1 & 1/3 & 1/5 & 1/7 & 1/8 \\ 4 & 3 & 1 & 1/4 & 1/4 & 1/5 \\ 6 & 5 & 4 & 1 & 1/3 & 1/3 \\ 8 & 7 & 4 & 3 & 1 & 1 \\ 9 & 8 & 5 & 3 & 1 & 1 \end{bmatrix} \end{array}$$

Собственные значения этой матрицы парных сравнений равны:

	170	175	180	185	190	195	**Wi**	**Wi норм**
170	1	=1/2	=1/4	=1/6	=1/8	=1/9	**=СТЕПЕНЬ(B2*C2*D2*E2*F2*G2; 1/6)**	**=H2/$H$7**
175	2	1	=1/3	=1/5	=1/7	=1/8	**=СТЕПЕНЬ(B3*C3*D3*E3*F3*G3; 1/6)**	**=H3/$H$7**
180	4	3	1	=1/4	=1/4	=1/5	**=СТЕПЕНЬ(B4*C4*D4*E4*F4*G4; 1/6)**	**=H4/$H$7**
185	6	5	4	1	=1/3	=1/3	**=СТЕПЕНЬ(B5*C5*D5*E5*F5*G5; 1/6)**	**=H5/$H$7**
190	8	7	4	3	1	1	**=СТЕПЕНЬ(B6*C6*D6*E6*F6*G6; 1/6)**	**=H6/$H$7**
195	9	8	5	3	1	1	**=СТЕПЕНЬ(B7*C7*D7*E7*F7*G7; 1/6)**	**=H7/$H$7**

	170	175	180	185	190	195	**Wi**	**Wi норм**
170	1	0,5	0,25	0,166667	0,125	0,111111	**0,25718031**	**0,080291**
175	2	1	0,333333	0,2	0,142857	0,125	**0,36541956**	**0,114083**
180	4	3	1	0,25	0,25	0,2	**0,72892337**	**0,227568**
185	6	5	4	1	0,333333	0,333333	**1,53989032**	**0,48075**
190	8	7	4	3	1	1	**2,95956725**	**0,923969**
195	9	8	5	3	1	1	**3,20310095**	**1**

Таблица 3 - Функции принадлежности нечеткого множества "высокий мужчина"

u_i	170	175	180	185	190	195
$\mu_{\text{высокий мужчина}}(u_i)$	0.08	0.11	0.22	0.48	0.92	1.0000

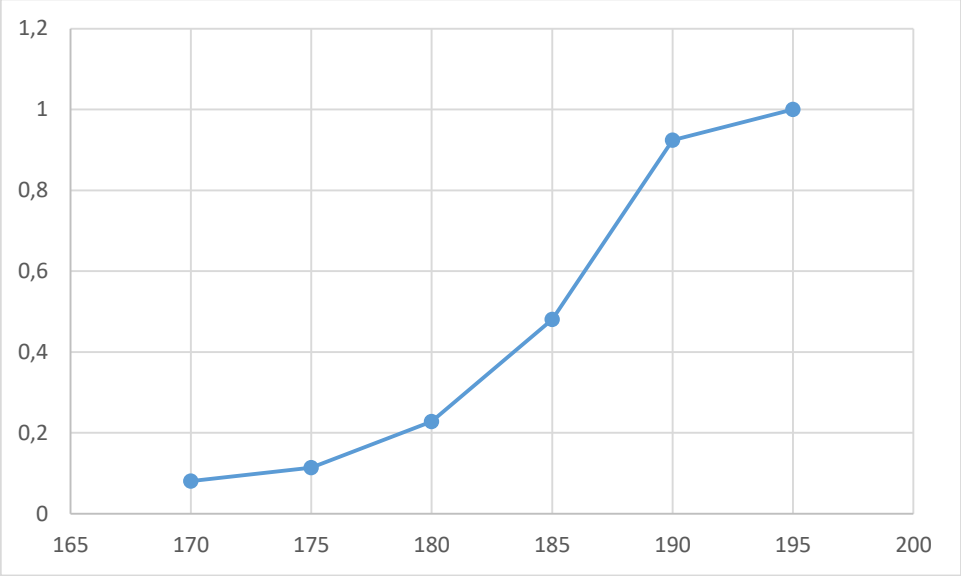


Рисунок 2. Функции принадлежности нечеткого множества "высокий мужчина"

Задание для самостоятельного выполнения

1. Пусть ***U***={понедельник, вторник, среда, четверг, пятница, суббота, воскресенье}. Выступая в роли эксперта, запишите в форме A = {$\mu_A(x)/x$}, следующие нечеткие множества: **A** – начало недели, ***B*** – середина недели, ***C*** – конец недели, ***D*** – не начало, но и не конец недели. Есть ли среди определенных вами функций принадлежности унимодальные?

2. Пусть ***U***={0,1,2,…120} – возможный возраст человека. Выступая в роли эксперта, постройте графики функций принадлежности следующих нечетких множеств с помощью метода парных сравнений: ***A*** - молодой, ***B*** – старый, ***C*** – очень молодой, ***D*** – не старый. Запишите эти множества в стандартной форме.

3. Решить задачу 2 с помощью метода статистической обработки экспертной информации, в качестве экспертов использовать своих одногруппников.

4. Пусть ***U*** – множество дисциплин, изучаемых в текущем семестре. Присвойте номер каждой дисциплине и, выступая в роли эксперта, запишите нечеткие множества:

A – мне нравится эта дисциплина

B – я не понимаю эту дисциплину

C – мне не нравится эта дисциплина

5. Пусть U – цены автомобилей, 4<=*u*<=5000 (усл.ед.). Выступая в роли эксперта, постройте графики функций принадлежности следующих нечетких множеств:

A – цены автомобилей для среднего класса

B – цены автомобилей для богатых людей

C - цены автомобилей для небогатых людей

Для каждой кривой найдите подходящую формулу и запишите функции принадлежности аналитически.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению лабораторных работ
по дисциплине «Машинное обучение и анализ данных для решения задач электроэнергетики»
для студентов направления подготовки /специальности
13.04.02 «Электроэнергетика и электротехника»,

Магистерская программа «Интеллектуальные системы электроснабжения "

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

Оглавление

Лабораторная работа №1. Разведочный анализ данных

Разведочный анализ данных (исследовательский анализ данных, exploratory data analysis, EDA) – подход к анализу данных для обобщения их основных характеристик, зачастую – с использованием статистической графики и других методов визуализации данных.

Разведочный анализ данных подразумевает выполнение анализа основных свойств данных, выявление общих закономерностей, основных структур, наиболее важных переменных, распределений, отклонений и аномалий в данных.

Разведочный анализ данных предполагает, что будут осуществляться:

- систематический сбор данных, в ходе которого выполнять проверка гипотез и оценка результатов;

- очистка данных, в ходе которой осуществляется проверка правильности и достоверности данных, в частности, выявление ошибок в данных или отсутствие данных, а также исправление, удаление ошибок и т.п.;

- предварительная обработка данных, в ходе которой осуществляется преобразование необработанных данных в некоторый формат с применением инструментов масштабирования, извлечения, выбора признаков и т.п. с последующим получением итогового набора данных, который может быть использован для решения различных прикладных задач;

- визуализация данных, в ходе которой реализуется графическое представление информации и данных с использованием статистических информационных графиков, а также других инструментов, обеспечивающих четкую и эффективную передачу информации.

Типичными графическими инструментами, используемыми в разведочном анализе, являются:

диаграмма размаха (box plot, box-and-whiskers diagram, «ящик с усами»,«ящичковая диаграмма»), разработанная Джоном Тьюки в 1970-х годах;

гистограмма (histogram, bar chart);

многовариантная диаграмма (multi-vari chart);

диаграмма Парето (Pareto chart);

диаграмма рассеяния в 2D/3D (scatter plot);

стебле-листовой график (stem-and-leaf plot);

параллельные координаты (parallel coordinates);

отношение шансов (odds ratio);

целенаправленное преследование проекций (targeted projection pursuit);

тепловая карта (heat map);

столбчатая диаграмма (bar chart);

график горизонта (horizon graph).

Типичными инструментами снижения размерности, используемыми в разведочном анализе, являются:

многомерное шкалирование (multidimensional scaling), представляющее собой метод анализа и визуализации данных с помощью расположения точек, соответствующих изучаемым (шкалируемым) объектам, в пространстве меньшей размерности, чем пространство признаков объектов;

методы и алгоритмы линейного снижения размерности (lineardimensionality reduction);

методы и алгоритмы нелинейного снижения размерности (nonlineardimensionality reduction);

иконография корреляций (iconography of correlations), заключающаяся в замене корреляционной матрицы диаграммой, где «замечательные»(remarkable) корреляции представлены сплошной линией, в случае положительной корреляции и пунктирной линией, в случае отрицательной корреляции.

Эффективными количественными инструментами, применяемыми в разведочном анализе данных, являются:

медианная полировка (median polish), предложенная Джоном Тьюки и позволяющая итеративно оценить влияние строк и столбцов на данные с применением медиан, вычисленных на основе строк и столбцов набора данных;

тримин-мера (trimean), предложенная Джоном Тьюки и позволяющая оценить расположение распределения вероятностей на основе средневзвешенного для медианы распределения и его двух квартилей;

ординационный или градиентный анализ (ordination or gradientanalysis), дополняющий кластеризацию данных и позволяющий упорядочивать многомерные объекты, которые характеризуются значениями нескольких переменных таким образом, что похожие объекты находятся рядом друг с другом, а непохожие объекты находятся дальше друг от друга.

В настоящее время многие методы и алгоритмы разведочного анализа данных адаптированы для интеллектуального анализа данных [1].

Метрики центральной тенденции для количественных признаков

Самый простой подход к обобщению количественного признака предполагает вычисление среднего, т.е. среднего арифметического значения на основе всех известных значений анализируемого признака. Хотя среднее вычисляется очень просто и его удобно использовать, оно не всегда является лучшей мерой центральной тенденции. В связи с этим на практике используют несколько альтернативных метрик, характеризующих центральную тенденцию анализируемого признака, в частности, среднее, среднее усеченное, среднее взвешенное, медиану, медиану взвешенную.

Среднее (mean, среднее арифметическое) – сумма всех значений признака, деленная на число этих значений.

Среднее усеченное (trimmed mean) является разновидностью среднего. Среднее усеченное вычисляется посредством отбрасывания фиксированного

числа значений с каждого конца сортированных последовательности и последующего взятия среднего арифметического оставшихся значений. Среднее усеченное позволяет устранить влияние предельных значений. Во многих случаях среднее усеченное более предпочтительно, чем обычное среднее.

Среднее взвешенное (weighted mean, среднее арифметическое взвешенное)вычисляется как

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i \cdot x_i}{\sum_{i=1}^{n} w_i},$$

где x_i –i-е значение признака;

w_i – вес i-го значения признака;

x –среднее;

n – число значений признака.

Среднее взвешенное целесообразно применять, например, если разные значения признака имеют разную точность (тогда менее точным значениям следует придать меньший вес) или значения признака не одинаково представляют разные оцениваемые (измеряемые) группы (тогда более высокий вес следует придать значениям признака из тех групп, которые были представлены недостаточно). Однако при применении среднего взвешенного возникает проблема адекватности используемых весов, т.к. процедура назначения весов имеет субъективный характер.

Медиана (median) – значение признака, расположенное в сортированном списке значений признаков ровно посередине. Если число значений признаков четное, средним значением является то, которое не находится в наборе данных фактически, а является средним арифметическим двух значений, которые делят сортированные значения признаков на верхнюю и нижнюю половины.

Медиана взвешенная (weighted median) – значение признака, для которого суммы весов для нижней и верхней половин сортированного списка значений признака равны. Медиана и медиана взвешенная устойчивы к выбросам. Медиану называют робастной (robust) метрикой центральной тенденции, т.к. она не находится под влиянием выбросов, которые могут исказить результаты. Медиана не чувствительна к выбросам, т.е. устойчива к ним.

Выброс (outlier) – любое значение признака, которое сильно удалено (сильно отличается) от других значений признака. Выброс не делает значение признака недопустимым или ошибочным. Тем не менее, выбросы часто являются результатом ошибок в данных, таких как смешивание данных с разными единицами измерения (например, килограммы и граммы) или плохие показания датчика. Если выбросы являются результатом ошибочных данных, среднее значение будет плохой метрикой центральной тенденции, в то время как медиана будет по-прежнему допустимой. Следует отметить, что выбросы всегда должны быть идентифицированы и исследованы.

Метрики вариабельности для количественных признаков

Метрики вариабельности признака могут быть основаны на отклонениях (ошибках, остатках, deviations) между метрикой центральной тенденции и наблюдаемыми значениями признака.

Пусть имеется набор значений признака, характеризующего частоту сердечных сокращений (число ударов в минуту, пульс):

{60, 80, 80}.

Среднее для этого набора равно *73.333*, а отклонения от среднего имеют вид: *−13.333, 6.667, 6.667.*

Эти отклонения показывают, насколько значения признака разбросаны вокруг центральной тенденции. Очевидно, что следует каким-либо образом оценить типичное значение этих отклонений. Среднее (среднее

арифметическое) отклонений может оказаться равным нулю (или очень близким к нулю), если сумма отклонений равна нулю (близка к нулю), например, как в рассматриваемом примере: −13.333+6.667+6.667 ≈0, хотя по факту отклонения ненулевые.

Как вариант, в качестве метрики вариабельности можно использовать среднее абсолютное отклонение от среднего (meanAD) , вычисляемое как:

$$meanAD = \frac{\sum_{i=1}^{n} |x_i - \bar{x}|}{n},$$

где x_i –i -е значение признака

x − среднее

n – число значений признака.

Для рассматриваемого набора данных вида {60, 80, 80} meanAD вычисляется как

meanAD=(13,333+6,667+6,667)/3=8.889

Наиболее часто в качестве метрик вариабельности используют дисперсию и стандартное отклонение, основанные на квадратических отклонениях.

Дисперсия (variance) – сумма квадратических отклонений от среднего, деленная на n-1, где n – число значений признака:

$$variance = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1},$$

где x_i –i -е значение признака;

x − среднее;

n – число значений признака.

По данной формуле вычисляется несмещённая оценки дисперсии, т.к. в знаменателе стоит n-1 . Если заменить n-1 на n , то оценка будет смещенной.

Дисперсия представляет собой среднее квадратических отклонений.

Стандартное отклонение (среднеквадратическое отклонение, standard deviation, STD, среднеквадратическое отклонение) – квадратный корень из дисперсии:

$$\sigma = \sqrt{variance}$$

Стандартное отклонение интерпретируется проще, чем дисперсия, т.к. оно находится на той же шкале измерения, что и значения признака.

При этом дисперсия и стандартное отклонение чувствительны к выбросам сильнее, чем среднее абсолютное отклонение, т.к. они основаны на квадратических отклонениях.

Робастной метрикой вариабельности является медианное абсолютное отклонение от медианы

$$medianAD = медиана\left(|x_1 - m|, |x_2 - m|, \ldots, |x_n - m|\right),$$

где m – медиана;

x_i –i -е значение признака

n – число значений признака.

Для рассматриваемого набора данных *medianAD* вычисляется как *medianAD = медиана (0,0,20)=0*

Как и в случае с медианой, medianAD не находится под влиянием выбросов. Следует отметить, что можно вычислить усеченное стандартное отклонение по аналогии со средним усеченным. Усеченное стандартное отклонение можно считать робастной метрикой вариабельности, т.к. оно позволяет предотвратить влияние выбросов.

Дисперсия, стандартное отклонение, среднее абсолютное отклонение и медианное абсолютное отклонение не являются эквивалентными метриками, даже если признак имеет нормальный закон распределения. Стандартное отклонение всегда больше среднего абсолютного отклонения, которое всегда больше медианного абсолютного отклонения.

Метрики вариабельности на основе процентилей

Метрики вариабельности признака могут быть основаны на процентилях. В этом случае рассматривают разброс (спред, spread) сортированных данных (значений признака). Порядковые статистики – статистические показатели на основе сортированных (ранжированных) данных.

Самая элементарная метрика, оценивающая разброс, – размах, т.е. разница между самым большим и самым малым значениями признака.

Размах очень чувствителен к выбросам и не очень полезен в качестве метрики вариабельности данных, хотя минимальные и максимальные значения признака полезно знать, т.к. они помогают идентифицировать выбросы.

Чтобы снизить чувствительность к выбросам, целесообразно рассмотреть размах данных (размах значений признака) после отбрасывания части значений с каждого конца сортированного набора значений признака. Метрики такого типа формально основаны на разницах между процентилями.

В наборе значений признака P-й процентиль – такое число z_P , что значения P-й части набора меньше или равны z_P .Например, 25-й процентиль – такое число z_P , что 25% значений признака меньше или равны z_P.

Медиана соответствует 50-й процентили.

Процентиль аналогичен квантилю, при этом квантили индексируются долями (так, например, квантиль 0,25 – это то же самое, что и 25-й процентиль). Обычно в качестве еще одной метрики вариабельности используют разницу между 25-м и 75-м процентилями. Эта метрика называется межквартильным размахом (interquartile range, IQR).

Пусть имеется набор значений признака, характеризующего частоту сердечных сокращений (число ударов в минуту, пульс):

{60, 55, 80, 75, 95, 90, 80, 70}.

Сначала числа в этом наборе необходимо отсортировать по возрастанию: *55, 60, 70, 75, 80, 80, 90, 95.*

25-й процентиль для рассматриваемого набора равен z_{25} =65, 75-й процентиль для рассматриваемого набора равен z_{75}=90, поэтому межквартильный размах будет 90 − 65 = 25.

Для больших наборов данных (с большим числом значений признака) расчет точных процентилей может быть вычислительно затратным ввиду необходимости выполнения процедуры сортировки всех значений признака. В этом случае могут быть использованы специальные алгоритмы, которые позволяют получить очень быстро приблизительный процентиль, гарантируя обеспечение требуемой точности.

Необходимо отметить следующее: если имеется четное число значений признаков (т.е. n – чётное), то определение процентиля не является однозначным. Можно взять любое значение между порядковыми статистиками

4. Анализ распределения данных для количественных признаков

Для выполнения исследования характера распределения значений признака в целом могут быть использованы такие инструменты, как коробчатая диаграмма (boxplot), частотная таблица (frequency table), гистограмма (histogram), график плотности (density plot).

Коробчатую диаграмму («ящик с усами», box-and-whiskers diagram) удобно использовать в качестве быстрого способа визуализации распределения данных.

Частотная таблица представляет собой сводку количеств числовых значений, которые разбиты на серию частотных интервалов (корзин, бинов).

Гистограмма является графическим изображением частотной таблицы, где частотные интервалы откладываются на оси x, а количества (или доли) – на оси y.

График плотности представляет собой сглаженную версию гистограммы (зачастую – на основе ядерной оценки плотности).

Коробчатые диаграммы основаны на процентилях и обеспечивают быстрый способ визуализации распределения данных (рисунок 1).

При вертикальном изображении коробчатой диаграммы верх и низ коробки (границы ящика) соответствуют 75-му и 25-му процентилям, т.е. первому и третьему квартилю. Кроме того, посередине коробки коробчатой диаграммы горизонтальной линией изображается медиана (50-й процентиль). Линии, называемые усами и выходящие из верха и низа коробки, говорят о размахе основной части данных (значений признака). Концы усов представляют собой края статистически значимой выборки (без выбросов)

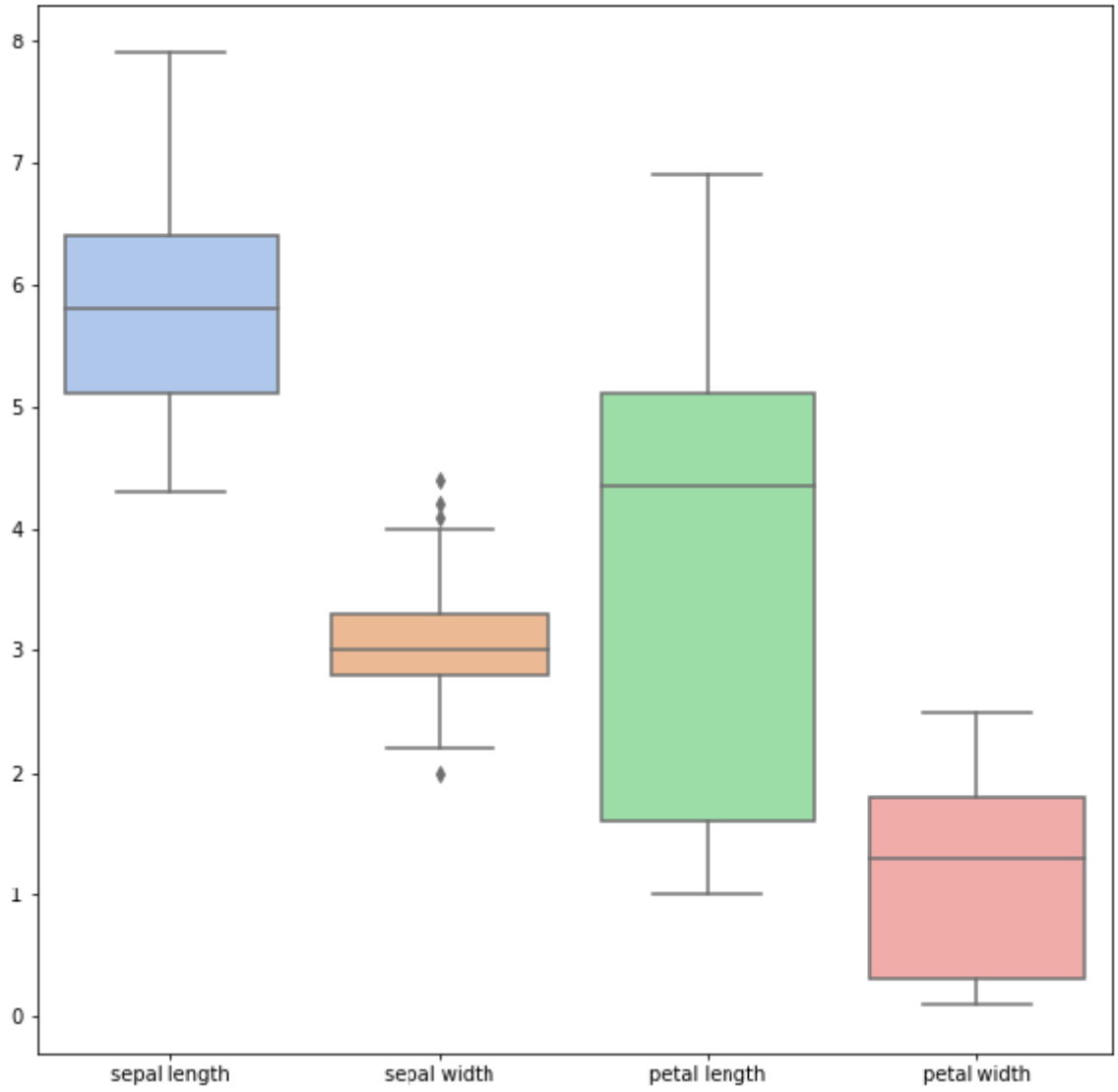


Рисунок 1. Визуализация распределения данных на примере набора «Ирисы Фишера» (The Iris Dataset) с применением коробчатых диаграмм

Частотная таблица делит диапазон значений признака на равноотстоящие интервалы и показывает, сколько значений признака попадает в каждый интервал. При этом необходимо учитывать пустые интервалы: обнаружение отсутствия значений признака в некоторых интервалах является полезной информацией.

Полезно экспериментировать с разными размерами интервалов. Если размеры интервалов будут очень большими, то полученные результаты окажутся слишком общими, а важные особенности распределения могут быть утеряны (сглажены). Если размеры интервалов будут очень маленькими, то полученные результаты будут очень детальными, и будет невозможно наблюдать общую картину.

Гистограмма (histogram) реализует визуализацию частотной таблицы таким образом, что частотные интервалы откладываются на оси x, а число значений признака – на оси y (рис. 2). Гистограмма представляет собой диаграмму, которая обычно является столбиковой. Высота каждого столбика гистограммы отражает число значений признака для интервала, на который он опирается.

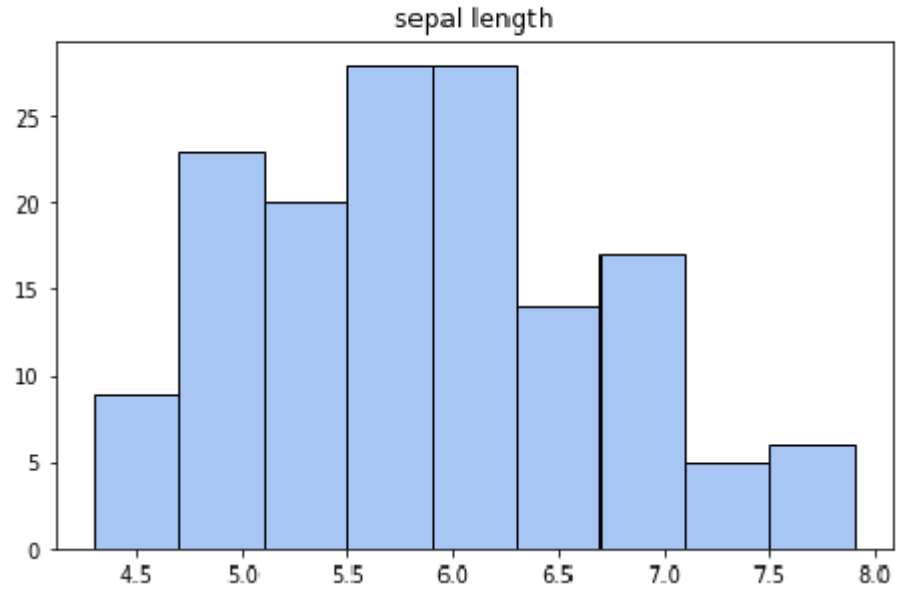


Рисунок 2. Гистограмма распределения для признака sepallengthнабора данных «Ирисы Фишера» (The Iris Dataset)

На гистограмме частот отображают число значений признака, попавших в каждый интервал. Для получения гистограммы для оценок вероятностей необходимо разделить число значений признака в каждом интервале на общее число значений признака. В этом случае сумма высот всех столбцов гистограммы будет равна 1 (как площадь под кривой закон распределения в теории вероятностей).

По результатам анализа формы гистограммы может предположить, какому статистическому закону распределения подчиняется анализируемый признак. Так, если все столбцы гистограммы имеют примерно одинаковую высоту, то можно предположить соответствие данных равномерному закону распределения, если все столбцы гистограммы образуют симметричный «холм», то предположить соответствие данных нормальному закону распределения.

Корреляция признаков

Существенный интерес в разведочном анализе данных представляет изучение корреляции между признаками, в том числе – корреляция признаков, описывающих объекты, с целевым признаком.

Признаки X и Y коррелируют положительно, если большим значениям признака X соответствуют большие значения признака Y, а малым значениям признака X соответствуют малые значения признака Y.

Признаки X и Y коррелируют отрицательно, если большим значениям признака X соответствуют малые значения признака Y, а малым значениям при-знака X соответствуют большие значения признака Y.

Для оценки корреляции между двумя признаками, находящимися на одинаковой шкале измерения, используют коэффициент корреляции (correlation coefficient) Пирсона:

$$r = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{(n-1) \cdot \sigma_x \cdot \sigma_y},$$

где x_i, y_i –i -е значения признаков X и Y соответственно

n – число значений признака;

x ,y – средние для признаков X и Y соответственно;

σ_x, σ_y – стандартные отклонения для признаков X и Y соответственно.

Коэффициент корреляции всегда находится между числом «−1», соответствующим идеальной отрицательной корреляции, и числом «+1», соответствующим идеальной положительной корреляции.

Значение коэффициента корреляции, равное числу «0», свидетельствует об отсутствии корреляции. Коэффициент корреляции, как среднее и стандартное отклонение, чувствителен к выбросам данных.

Следует отметить, что признаки могут иметь нелинейную связь. В этом случае коэффициент корреляции может оказаться бесполезным.

На основе значений коэффициента корреляции для различных пар признаков может быть создана корреляционная матрица (correlation matrix). В этой матрице строки и столбцы соответствуют признакам в наборе данных, а значения на пересечении строк и столбцов представляют собой значения коэффициента корреляции для соответствующей пары признаков.

Можно отметить, что на диагонали корреляционной матрицы стоят единицы (т.к. корреляция любого признака с самим собой равна 1), при этом матрица симметрична относительно главной диагонали.

Диаграмма рассеяния (scatter plot) является стандартным инструментом визуализации связи между двумя количественными признаками (рис. 3). На такой диаграмме ось x представляет один признак, ось y – другой.

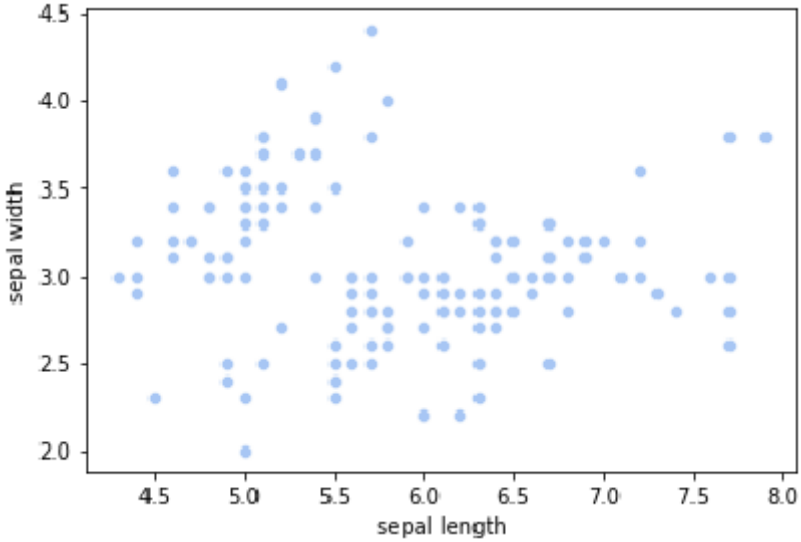


Рисунок 3. Диаграмма рассеяния для пары признаков sepallength и sepalwidth набора данных «Ирисы Фишера» (The Iris Dataset)

Диаграммы рассеяния удобно использовать, если в наборе данных имеется небольшое число объектов. Для наборов данных с сотнями тысяч и миллионов объектов диаграмма рассеяния будет очень плотной, в этом случае удобнее использовать график с шестиугольной сеткой (hexagonal binning) для отображения связи между двумя значениями признаков (рис. 4). При работе с шестиугольной сеткой данные группируют в шестиугольные сегменты, показывая цветом число записей в конкретном сегменте.

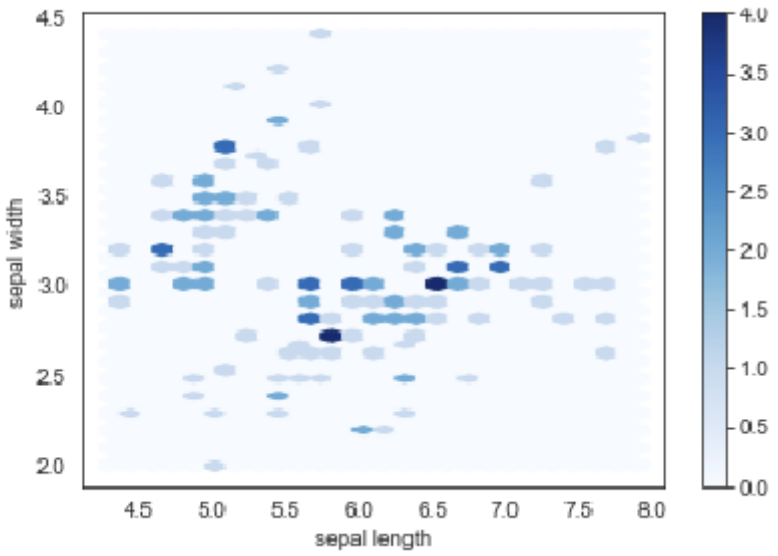


Рисунок 4. Визуализация связи для пары признаков sepallength и sepalwidth набора данных «Ирисы Фишера» (The Iris Dataset)с использованием графика с шестиугольной сеткой

Разведочный анализ с использованием библиотеки NumPy

Модуль Statistics из программной библиотеки NumPy содержит функции, позволяющие вычислить порядковые статистики, метрики среднего положения и вариативности, оценить корреляцию, выполнить расчеты для построения гистограмм.

median(data[, axis,out,overwrite_input,keepdims])

Возвращает медиану вдоль выбранного измерения массива.

Входные параметры:

data (тип: array_like) – массив или объект, который можно преобразовать в массив;

axis (тип: int, последовательность из int или None) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; последовательность измерений поддерживается с версии 1.9.0;out (тип: ndarray) – альтернативный выходной массив, в который следует поместить

результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости;

overwrite_input (тип: bool) – параметр, который принимает 2 значения: True или False; если overwrite_input имеет значение True, то разрешено использование памяти массива data для вычислений: массив будет изменен вызовом функции, что позволит сэкономить память, когда не нужно сохранять содержимое входного массива; по умолчанию overwrite_input имеет значение False;

keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве.

Выходной параметр:

median (тип: ndarray) – массив, содержащий результат; если входные данные содержат целые числа или числа с плавающей запятой меньшие, чем float64, то тип выходных данных – np.float64, в противном случае тип выходных данных совпадает с типом входных данных; если параметр out специфицирован, то возвращается ссылка на выходной массив.

average (data[, axis,weights,returned,keepdims])Возвращает взвешенное среднее вдоль выбранной оси массива.

Входные параметры:

data (тип: array_like) – массив или объект, который можно преобразовать в массив;

axis (тип: int, последовательность из int или None) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву

data; последовательность измерений поддерживается с версии 1.9.0;weights (тип: array_like) – массив весов, связанных со значениями в массиве data: каждое значение в массиве data вносит свой вклад в среднее значение в соответствии с соответствующим весом; массив весов может быть одно-мерным (в этом случае его длина должна быть равна размеру массива data по заданному измерению axis) или иметь ту же форму, что и массив data; если *weights=None*, то предполагается, что все данные в массиве data имеют вес, равный единице; взвешенное среднее вычисляется как *avg=sum(a*weights)/sum(weights),* при этом *sum(weights)* не должна быть равна нулю;

returned (тип: bool) – параметр, который принимает2 значения: True или False; если параметр returned имеет значение True, функция возвращается кортеж *(average, sum_of_weights)*, в противном случае возвращается только среднее значение; по умолчанию параметр returned имеет значение False. Если *weights=None*, величина *sum_of_weights* эквивалентна числу элементов, по которым берется среднее значение;

keepdims (тип: bool) – параметр, который принимает2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по кото-рым выполняется вычисление функции, полагается равной1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве.

Выходной параметр:

retval, [sum_of_weights] (тип: array_type илидouble) – среднее значение по выбранной оси. Если returned равно True, функция возвращает кортеж со средним значением в качестве первого элемента и суммой весов в качестве второго элемента; параметр sum_of_weights имеет тот же тип, что и retval.

mean (data[, axis,dtype, out,keepdims, where]) Возвращает среднее арифметическое вдоль выбранной оси массива.

Входные параметры:

data (тип: array_like) – массив или объект, который можно преобразовать в массив;

axis (тип: int, последовательность из int или None) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; последовательность измерений поддерживается с версии 1.9.0;

dtype (тип: data-type) – тип результата при вычислении среднего значения: для целочисленных входных данных используется тип float64;для входных данных с плавающей запятой используется тот же тип, что и тип у массива data;

out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости;

keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве;

where (тип: array_like,при этом тип элементов – bool) – массив, показывающий, какие элементы массива data следует использовать при вычислении среднего.

Выходной параметр:

m (тип: ndarray) – массив, содержащий средние значения, если out=None; в противном случае – ссылка на выходной массив.

std (data[, axis,dtype, out, ddof,keepdims, where]) Возвращает стандартное отклонение вдоль выбранной оси массива.

Входные параметры:

Data (тип: array_like) – массив или объект, который можно преобразовать в массив;

axis (тип: int, последовательность из int или None) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; последовательность измерений поддерживается с версии 1.9.0;

dtype (тип: data-type) – тип результата при вычислении среднего значения: для целочисленных входных данных используется тип float64; для входных данных с плавающей запятой используется тот же тип, что и тип у массива data;

out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости;

ddof (тип: int) – число степеней свободы; в расчетах используется делитель n– ddof, где n – число элементов; по умолчанию параметр ddof равен нулю;

keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве;

where (тип: array_like, при этом тип элементов – bool) – массив, показывающий, какие элементы массива data следует использовать при вычислении стандартного отклонения.

Выходной параметр:

standard_deviation (тип: ndarray) – массив, содержащий стандартные отклонения, если out=None; в противном случае – ссылка на выходной массив.

var (data[, axis,dtype, out, ddof,keepdims, where]) Возвращает дисперсию вдоль выбранной оси массива.

Входные параметры:

data (тип: array_like) – массив или объект, который можно преобразовать в массив;

axis (тип: int, последовательность из int или None) – измерение (или измерения), по которому (по которым) рассчитывается статистика; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data; по-ледовательность измерений поддерживается с версии 1.9.0;

dtype (тип: data-type) – тип результата при вычислении среднего значения: для целочисленных входных данных используется тип float64; для входных данных с плавающей запятой используется тот же тип, что и тип у массива data;

out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости;

ddof (тип: int) – число степеней свободы; в расчетах используется делитель n– ddof, где n – число элементов; по умолчанию параметр ddof равен нулю;

keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве;

where (тип: array_like, при этом тип элементов – bool) – массив, показывающий, какие элементы массива data следует использовать при вычислении дисперсии.

Выходной параметр:

variance (тип: ndarray) – массив, содержащий значения дисперсии, если out=None; в противном случае –ссылка на выходной массив.

percentile (data,q[, axis, out,overwrite_input,method, keepdims, *, interpolation]) Возвращает q-процентиль вдоль выбранного измерения массива.

Входные параметры:

data (тип: array_like) – массив или объект, который можно преобразовать в массив;

q (тип: array_like типafloat) – процентиль или последовательность процентилей, которые должны быть в диапазоне от 0 до 100 включительно;

axis (тип: int, кортеж из int или None) – измерение, по которому ищутся пики; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data;

out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив дол-жен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости;

overwrite_input (тип: bool) – параметр, который принимает 2 значения: True или False; если overwrite_input имеет значение True, то разрешено использование памяти массива data для вычислений: массив будет изменен вызовом функции, что позволит сэкономить память, когда не нужно сохранять содержимое входного массива; по умолчанию overwrite_input имеет значение False;

Method (тип:str) – метод, используемый для оценки процентиля: *«inverted_cdf», «averaged_inverted_cdf», «closest_observation», «interpolated_inverted_cdf», «hazen», «weibull», «linear»* (по умолчанию), *«median_unbiased», «normal_unbiased»,* кроме того, имеются 3 вариации метода *«linear»: «lower», «higher», «midpoint» и «nearest»;*

keepdims (тип: bool) – параметр, который принимает 2значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве;

interpolation (тип:str) – устаревшее имя для параметра method.

Выходной параметр:

percentile (тип: scalar или ndarray) – массив, содержащий результат; если q – одиночный процентиль и axis=None, то результат – скаляр; если дано несколько процентилей, то первое измерение результата соответствует процентилям, а другие измерения – это измерения, которые остались после выполнения действий над массивом data; если входные данные содержат целые числа или числа с плавающей запятой меньшие, чем float64, то тип выходных данных – np.float64,в противном случае тип выходных данных совпадает с типом входных данных; если параметр out специфицирован, то возвращается ссылка на выходной массив.

quantile (data,q [, axis, out,overwrite_input, method, keepdims, *, interpolation]) Возвращает q-квантиль вдоль выбранного измерения массива.

Входные параметры:

data (тип: array_like) – массив или объект, который можно преобразовать в массив;

q (тип: array_like типаfloat) – квантиль или последовательность квантилей, которые должны быть в диапазоне от 0 до 1 включительно;

axis (тип: int, кортеж из int или None) – измерение, по которому ищутся пики; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data;

out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину

буфера, что и ожидаемый результат, но тип результата может быть преобразован при необходимости;

overwrite_input (тип: bool) – параметр, который принимает 2 значения: True или False; если overwrite_inputимеет значение True, то разрешено использование памяти массива data для вычислений: массив будет изменен вызовом функции, что позволит сэкономить память, когда не нужно сохранять содержимое входного массива; по умолчанию overwrite_input имеет значение False;

method (тип:str) – метод, используемый для оценки процентиля: *«inverted_cdf», «averaged_inverted_cdf», «closest_observation», «interpolated_inverted_cdf», «hazen», «weibull», «linear»* (по умолчанию), *«median_unbiased», «normal_unbiased»*, кроме того, имеются 3 вариации метода *«linear»: «lower», «higher», «midpoint» и «nearest»;*

keepdims (тип: bool) – параметр, который принимает2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве;

interpolation (тип:str) – устаревшее имя для пара-метра method.

Выходной параметр:

quantile (тип: scalar или ndarray) – массив, содержащий результат; если q – одиночный квантиль и *axis=None*, то результат – скаляр; если дано несколько квантилей, то первое измерение результата соответствует квантилям, а другие измерения – это измерения, которые остались после выполнения действий над массивом data; если входные данные содержат целые числа или числа с плавающей запятой меньшие, чем float64, то тип выходных данных – np.float64, в противном случае тип выходных данных совпадает с типом входных данных; если параметр out специфицирован, то возвращается ссылка на выходной массив.

ptp (a[,axis,out,keepdims]) Возвращает диапазон значений (максимум – минимум) вдоль выбранного измерения массива.

Входные параметры:

data (тип: array_like) – массив или объект, который можно преобразовать в массив;

axis (тип: int, кортеж из int или None) – измерение, по которому ищутся пики; по умолчанию (если axis отсутствует) вычисления проводятся по всему массиву data;

out (тип: ndarray) – альтернативный выходной массив, в который следует поместить результат; этот массив должен иметь ту же форму и длину буфера, что и ожидаемый результат, но тип результата может быть преобразован принеобходимости;

keepdims (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр keepdims имеет значение True, то размерность измерений, по которым выполняется вычисление функции, полагается равной 1 с целью поддержания того же числа измерений в результирующем массиве, что и в исходном массиве.

Выходной параметр:

ptp (тип: ndarray) – массив, содержащий результат; если параметр out специфицирован, то возвращается ссылка на выходной массив.

corrcoef (x[, y,rowvar,dtype]) Возвращает коэффициенты корреляции Пирсона.

Входные параметры:

x (тип: array_like) – одномерный или двумерный массив, содержащий несколько переменных и наблюдений: каждая строка массива x представляет собой переменную, а каждый столбец – одно наблюдение всех этих переменных;

y (тип: array_like) – дополнительный массив, содержащий несколько переменных и наблюдений: каждая строка массива x представляет собой переменную, а каждый столбец – одно наблюдение всех этих переменных; массив y имеет те же размеры, что и массив x;

rowvar (тип: bool) – параметр, который принимает 2значения: True или False; если rowvar имеет значение True (по умолчанию), то каждая строка представляет собой переменную с наблюдениями в столбцах; в противном случае массив транспонируется: каждый столбец представляет собой переменную, а строки содержат наблюдения;

dtype (тип: data-type) – тип данных результата; по умолчанию возвращаемый тип данных будет иметь точность не менее numpy. float64.

Выходной параметр:

R (тип: ndarray) – массив коэффициентов корреляции переменных.

correlate (a, v[, mode]) Возвращает взаимную корреляцию двух одномерных последовательностей.

Входные параметры:

a, v (тип: array_like) – входные последовательности;

mode ({’valid’, ’same’, ’full’}) – параметр, определяющий тип свертки данных; значение ’full’ используется по умолчанию.

Выходной параметр:

out (тип: ndarray) – массив, характеризующий дискретную взаимную корреляцию массивов a и v.

cov (m[, y,rowvar, bias,ddof, fweights,aweights, dtype]) Возвращает ковариационную матрицу.

Входные параметры:

m (тип: array_like) – одномерный или двумерный массив, содержащий несколько переменных и наблюдений: каждая строка массива x представляет

собой переменную, а каждый столбец – одно наблюдение всех этих переменных;

y (тип: array_like) – дополнительный массив, содержащий несколько переменных и наблюдений: каждая строка массива x представляет собой переменную, а каждый столбец – одно наблюдение всех этих переменных; массив y имеет те же размеры, что и массив x;

rowvar (тип: bool) – параметр, который принимает 2значения: True или False; если rowvar имеет значение True (по умолчанию), то каждая строка представляет собой переменную с наблюдениями в столбцах; в противном случае массив транспонируется: каждый столбец представляет собой переменную, а строки содержат наблюдения;

dtype (тип: data-type) – тип данных результата; по умолчанию возвращаемый тип данных будет иметь точность не менее numpy. float64.

bias (тип: bool) – параметр, который принимает 2 значения: True или False; если *bias=False* (по умолчанию), нормализация осуществляется с (n–1), где n – число приведенных наблюдений (непредвзятая оценка); если *bias=True*, то нормализация осуществляется с n; эти значения можно переопределить с помощью параметр addof в версиях numpy>= 1.5;

ddof (тип: int) – параметр, принимающий по умолчанию значение None; если ddof=1, то функция возвращает несмещенную оценку, даже если указаны как fweights, так и aweights; если ddof=0, то функция возвращает простое среднее;

fweights (тип: array_like, int) – одномерный массив целочисленных частотных весов, определяющий число повторений каждого вектора наблюдения;

aweights (тип: int) – одномерный массив весов векторов наблюдения; эти относительные веса обычно велики для наблюдений, считающихся «важными», и малы для наблюдений, считающихся менее «важными»; если

ddof=0, массив весов можно использовать для присвоения вероятностей векторам наблюдения;

dtype (тип: int) – тип данных результата; по умолчанию возвращаемый тип данных будет иметь точность не меньше, чем numpy. float64.

Выходной параметр:

out (тип: ndarray) – ковариационная матрица переменных.

histogram (a[, bins,range, density, weights]) Возвращает гистограмму.

Входные параметры:

data (тип: array_like) – массив или объект, который можно преобразовать в массив; гистограмма вычисляется по всему массиву;

bins (тип: int или последовательность скаляров или str) – параметр, определяющий бины (интервалы); если bins имеет значение int, то параметр определяет число бинов равной ширины в заданном диапазоне (по умолчанию 10); если bins – последовательность, то параметр определяет монотонно увеличивающийся массив ребер бинов, включая крайнее правое ребро, что допускает не-равномерную ширину бинов; если bins – строка, то параметр определяет метод, используемый для вычисления оптимальной ширины бина;

range (тип: (float, float)) – нижняя и верхняя граница бинов (значения вне диапазона игнорируются); если параметр не специфицирован, то диапазон определяется как (*a.min(), a.max());*

density (тип: bool) – параметр, который принимает 2 значения: True или False; если параметр density равен False, результат будет содержать число значений в каждом бине; если параметр density равен True, результат будет равен значению функции плотности вероятности в бине, нормализованной таким образом, что интеграл по диапазону равен 1;

weights (тип: array_like) – массив весов того же размера, что и массив data; если для параметра density установлено значение True, веса нормализуются, так что интеграл плотности по диапазону остается равным 1;

Выходной параметр:

hist (тип: array) – массив, содержащий значения гистограммы;

bin_edges (тип: array of dtype float) – массивребер бинов.

bincount (x,/[, weights,minlength]) Возвращает число вхождений каждого значения в массиве неотрицательных целых чисел.

x (тип: array_like) – массив или объект, который можно преобразовать в массив;

weights (тип: array_like) – массив весов тех же размеров, что и массив x;

minlength (тип: int) – минимальное число бинов для выходного массива.

out (тип: ndarrayofints) – массив с результатами бинирования массива x; длина массива out равна np.amax(x)+1.

Визуализация с применением библиотеки matplotlib

Функция matplotlib.axes.Axes.bar.

Функция реализует построение столбчатой диаграммы. Прямоугольники располагаются по x с заданным выравниванием. Размеры прямоугольников указываются по высоте и ширине. Вертикальная базовая линия – нижняя (по умолчанию 0). Многие параметры могут принимать либо одно значение, применимое ко всем столбцам, либо последовательность значений, по одному для каждого столбца.

Синтаксис:

*Axes.bar(x, height, width=0.8, bottom=None, *,align='center', data=None, **kwargs)*

Функция matplotlib.axes.Axes.hist.

Функция реализует вычисление и построение гистограммы. Функция использует numpy.histogram для группировки данных по x и подсчета числа

значений в каждом бине, а затем строит распределение в виде BarContainer или Polygon. Параметры бинов, диапазона, плотности и веса передаются в numpy.histogram.

Синтаксис:

*Axes.hist (x, bins=None, range=None, density=False, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None, log=False, color=None, label=None, stacked=False, *, data=None, **kwargs)*

Фукция matplotlib.axes.Axes.boxplot.

Функция реализует построение коробчатой диаграммы (диаграммы «ящик с усами»). Коробка простирается от первого квартиля Q1 до третьего квартиля Q3 данных с линией в медиане. Усы выходят за пределы коробки на 1,5-кратный межквартильный диапазон IQR. Точки-выбросы – это данные, которые находятся за концами усов.

Синтаксис:

*Axes.boxplot(x, notch=None, sym=None, vert=None, whis=None, positions=None, widths=None, patch_artist=None, bootstrap=None, usermedians=None, conf_intervals=None, meanline=None, showmeans=None, showcaps=None, show-box=None, showfliers=None, boxprops=None, labels=None, flierprops=None, medianprops=None, meanprops=None, cap-props=None, whiskerprops=None, manage_ticks=True, autorange=False, zorder=None, capwidths=None, *, data=None)*

Функция matplotlib.axes.Axes.pie.

Функция реализует построение круговой диаграммы массива x. Дробная площадь каждого клина определяется как x/sum(x). Клинья строятся против часовой стрелки, по умолчанию начиная с оси x.

Синтаксис:

*Axes.pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=0, radius=1, counterclock=True, wedgeprops=None, textprops=None, center=(0,0), frame=False, rotatelabels=False, *, normalize=True, data=None)*

Функция matplotlib.pyplot.scatter.

Функция реализует построение диаграммы рассеяния для признаков x и y с разным размером и/или цветом маркера.

Синтаксис:

*matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, *, edgecolors=None, plotnonfinite=False, data=None, **kwargs)*

Пример выполнения задания:

Задание: провести разведочный анализ данных о потреблении электроэнергии за месяц, содержащихся в файле «апрель.csv». Построить гистограмму распределения данных о потреблении.

main.py - G:/курсы анализ данных/main.py (3.11.0)

File Edit Format Run Options Window Help

```
from csv import reader
import numpy as np
import matplotlib.pyplot as plt

with open('апрель.csv', 'r') as csvfile: # Открываем csv в формате чтения
    cr = list(reader(csvfile, delimiter=';')) # делим файл точкой с запятой
    cr = list(map(lambda x: list(map(lambda y: int(y), x)), cr[1:])) # переводим все знавчения в int
a = np.array(cr).transpose()[3]
t = np.array(cr).transpose()[2]
print(np.median(a))
print(np.average(a))
print(np.mean(a))
print(np.std(a))
print(np.var(a))
print(np.percentile(a, 10))
print(np.quantile(a, 0))
print(np.ptp(a))
print(np.corrcoef(a, t))
print(np.correlate(a, t))
print(np.cov(a, t))
print(np.histogram(a, 10))
print(np.bincount(a))
plt.hist(a)
plt.show()
```

Задание для самостоятельного выполнения:

1. Провести разведочный анализ данных о потреблении электроэнергии за месяц, содержащихся в файле «апрель.csv».
2. Построить различные диаграммы по данным данных о потреблении.
3. Определить наличие корреляции между признаками «Температура» и «Потребление», «Время» и «Потребление».

Литература:

1. Демидова, Л. А. Разведочный анализ данных. Python : учебно-методическое пособие / Л. А. Демидова. — Москва : РТУ МИРЭА, 2022 — Часть 1 — 2022. — 107 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/310970

Лабораторная работа 2. Классификация с помощью сетей Кохонена

Важной задачей ИАД является кластеризация. Кластеризация - группировка объектов (точнее, их векторов в пространстве признаков) в обособленные группы, называемые кластерами. При этом в каждый кластер группируются объекты с близкими значениями признаков.

На первый взгляд, задачи классификации и кластеризации похожи: и в той, и в другой имеет место группировка «похожих» объектов. Но на самом деле между этими задачами есть принципиальное отличие: при классификации классы должны быть предварительно определены и описаны, а для кластеризации таких требований нет - при кластеризации алгоритм должен самостоятельно объединить все представленные объекты в кластеры исключительно на основе каких-либо критериев их близости. Кластеризация оказывается особенно полезной, когда отсутствуют априорные сведения о структурных характеристиках набора данных. Ещё одним важным моментом является отсутствие необходимости использовать в кластерных моделях целевую переменную, поэтому для построения кластерной модели может использоваться обучения без учителя.

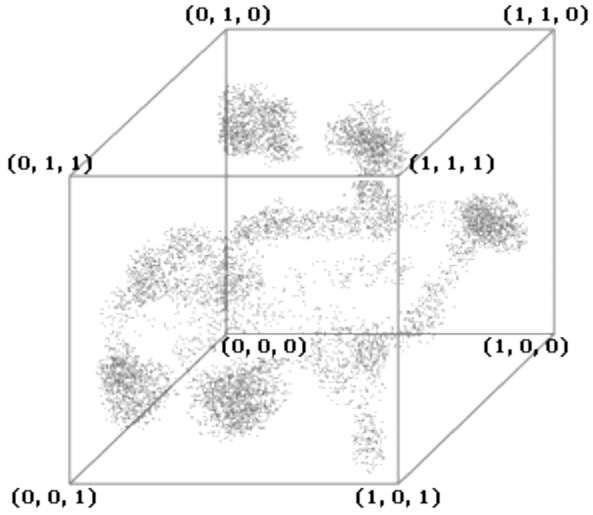
С точки зрения извлечения новых знаний кластеризация представляет больший интерес, чем классификация. Действительно, при классификации любой объект, предъявляемый модели, обязательно будет отнесен к одному из заранее определённых классов, даже если в действительности он ни к одному из классов не относится. В случае кластеризации «нетипичные» объекты образуют новый кластер, что позволяет обнаружить их новые свойства. Иными словами, кластеризация более полезна с точки зрения «обнаружения новизны», чем классификация, поскольку позволяет обнаруживать и описывать объекты с ранее неизвестными, новыми свойствами.

Одним из наиболее популярных методов кластеризации, который позволяет не только формировать многомерную кластерную структуру, но и

эффективно визуализировать ее, являются самоорганизующиеся карты признаков (SOM –self-organizing map), или просто карты Кохонена. В основе функционирования SOM лежит технология обучения без учителя для специального вида НС, называемых сетями Кохонена (СК), с визуализацией кластеров на основе проекции многомерных данных на плоскость с сохранением топологического подобия (проекции Саммона).

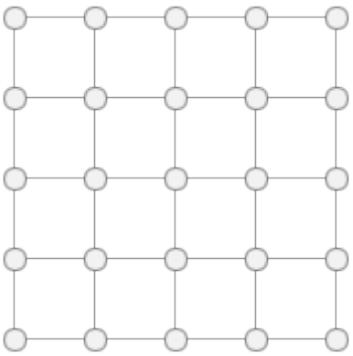
Рассмотрим, как решаются такие задачи и как карты Кохонена находят закономерности в исходных данных. Для упрощения рассмотрения будем считать, что объекты имеют 3 признака (на самом деле их может быть любое количество).

Теперь представим, что все эти три параметра объектов представляют собой их координаты в трехмерном пространстве (в том самом пространстве, которое окружает нас в повседневной жизни). Тогда каждый объект можно представить в виде точки в этом пространстве, что мы и сделаем (чтобы у нас не было проблем с различным масштабом по осям, пронормируем все эти признаки в интервал [0,1] любым подходящим способом), в результате чего все точки попадут в куб единичного размера. Отобразим эти точки.



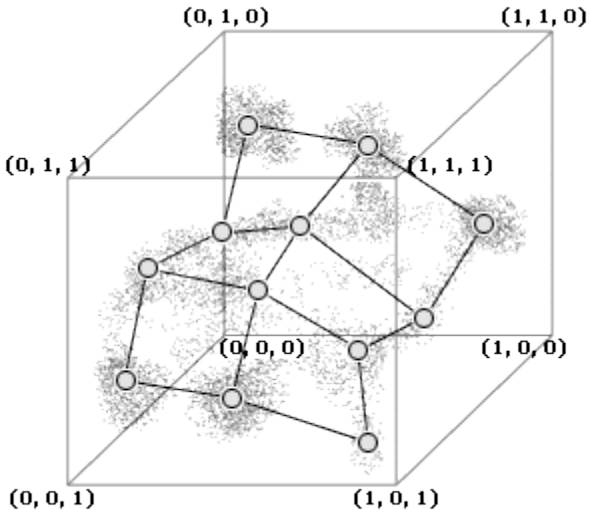
Взглянув на этот рисунок, мы можем увидеть, как расположены объекты в пространстве, причем легко заметить участки, где объекты группируются, т.е. у них схожи параметры, значит, и сами эти объекты, скорее всего, принадлежат одной группе. Но так легко можно поступить только в случае, когда признаков

немного. Значит, нам надо найти способ, которым можно преобразовать данную систему в простую для восприятия, желательно двумерную систему (потому что уже трехмерную картинку невозможно корректно отобразить на плоскости) так, чтобы соседние в искомом пространстве объекты оказались рядом и на полученной картинке. Для этого используем самоорганизующуюся карту Кохонена. В первом приближении ее можно представить в виде сети, изготовленной из резины.



Мы, предварительно "скомкав", бросаем эту сеть в пространство признаков, где у нас уже имеются объекты, и далее поступаем следующим образом: берем один объект (точку в этом пространстве) и находим ближайший к нему узел сети. После этого этот узел подтягивается к объекту (т.к. сетка "резиновая", то вместе с этим узлом так же, но с меньшей силой подтягиваются и соседние узлы). Затем выбирается другой объект (точка), и процедура повторяется. В результате мы получим карту, расположение узлов которой совпадает с расположением основных скоплений объектов в исходном пространстве. Кроме того, полученная карта обладает следующим замечательным свойством – узлы ее расположились таким образом, что объектам, похожим между собой соответствуют соседние узлы карты. Теперь определяем, какие объекты у нас попали в какие узлы карты. Это также определяется ближайшим узлом – объект попадает в тот узел, который находится ближе к нему. В результате всех этих операций объекты со схожими параметрами попадут в один узел или в соседние узлы. Таким образом можно

считать, что мы смогли решить задачу поиска похожих объектов и их группировки.



Но на этом возможности карт Кохонена не заканчиваются. Они позволяют также представить полученную информацию в простой и наглядной форме путем нанесения раскраски. Для этого мы раскрашиваем полученную карту (точнее ее узлы) цветами, соответствующими интересующим нас признакам объектов. Возвращаясь к примеру с классификацией банков, можно раскрасить одним цветом те узлы, куда попал хоть один из банков, у которых была отозвана лицензия. Тогда после нанесения раскраски мы получим зону, которую можно назвать зоной риска, и попадание интересующего нас банка в эту зону говорит о его ненадежности.

Но и это еще не все. Мы можем также получить информацию о зависимостях между параметрами. Нанеся на карту раскраску, соответствующую различным статьям отчетов, можно получит так называемый атлас, хранящий в себе информацию о состоянии рынка. При анализе, сравнивая расположение цветов на раскрасках, порожденных различными параметрами, можно получить полную информацию о финансовом портрете банков – неудачников, процветающих банков и т.д.

При всем этом описанная технология является универсальным методом анализа. С ее помощью можно анализировать различные стратегии

деятельности, производить анализ результатов маркетинговых исследований, проверять кредитоспособность клиентов и т.д.

Пример выполнения работы

Рассмотрим построение кластерной модели на основе карты Кохонена в аналитической платформе Deductor для набора данных Turkiye Student Evaluation, предоставленных студентами Университета Гази в Анкаре (Турция). Данные взяты из открытого источника http://archive.ics.uci.edu/dataset/262/turkiye+student+evaluation. Всего имеется 28 конкретных вопросов курса и 5 дополнительных атрибутов.

instr: идентификатор инструктора; значения взяты из набора {1,2,3}:

class: код курса (дескриптор); значения, взятые из {1-13},

nb.repeat: сколько раз студент проходит этот курс; значения взяты из {0,1,2,3,...}

attendance (посещаемость): Код уровня посещаемости;

difficulty (значения сложности): уровень сложности курса, как он воспринимается студентом; значения взяты из {1,2,3,4,5}

Следующие поля содержат ответы на заданные вопросы:

Q1: Содержание семестрового курса, метод обучения и система оценки были предоставлены в начале.

Q2: Цели и задачи курса были четко сформулированы в начале периода.

Q3: Курс оправдал назначенную на него сумму кредитов.

Q4: Курс преподавался в соответствии с программой, объявленной в первый день занятий.

Q5: Обсуждения в классе, домашние задания, приложения и учеба были удовлетворительными.

Q6: Ресурсы учебников и других курсов были достаточными и актуальными.

Q7: Курс допускал полевые работы, прикладные, лабораторные, дискуссионные и другие исследования.

Q8: Викторины, задания, проекты и экзамены способствовали обучению.

Q9: Мне очень понравилось занятие, и я очень хотел активно участвовать в его лекциях.

Q10: Мои первоначальные ожидания относительно курса оправдались в конце периода или года.

Q11: Курс был актуален и полезен для моего профессионального развития.

Q12: Курс помог мне взглянуть на жизнь и мир по-новому.

Q13: Знания инструктора были актуальными и актуальными.

Q14: Преподаватель пришел подготовленным к занятиям.

Q15: Преподаватель преподавал в соответствии с объявленным планом урока.

Q16: Инструктор был предан курсу и его можно было понять.

Q17: Преподаватель прибыл на занятия вовремя.

Q18: У преподавателя плавная и понятная речь.

Q19: Преподаватель эффективно использовал классные часы.

Q20: Преподаватель объяснил курс и стремился помочь студентам.

Q21: Преподаватель продемонстрировал позитивный подход к ученикам.

Q22: Преподаватель был открыт и уважительно относился к мнениям студентов о курсе.

Q23: Преподаватель поощрял участие в курсе.

Q24: Преподаватель давал соответствующие домашние задания/проекты и помогал/направлял учащихся.

Q25: Преподаватель ответил на вопросы о курсе внутри и за пределами курса.

Q26: Система оценки преподавателя (промежуточные и заключительные вопросы, проекты, задания и т. д.) эффективно оценивала цели курса.

Q27: Преподаватель предлагал решения для экзаменов и обсуждал их со студентами.

Q28: Преподаватель относился ко всем ученикам корректно и объективно.

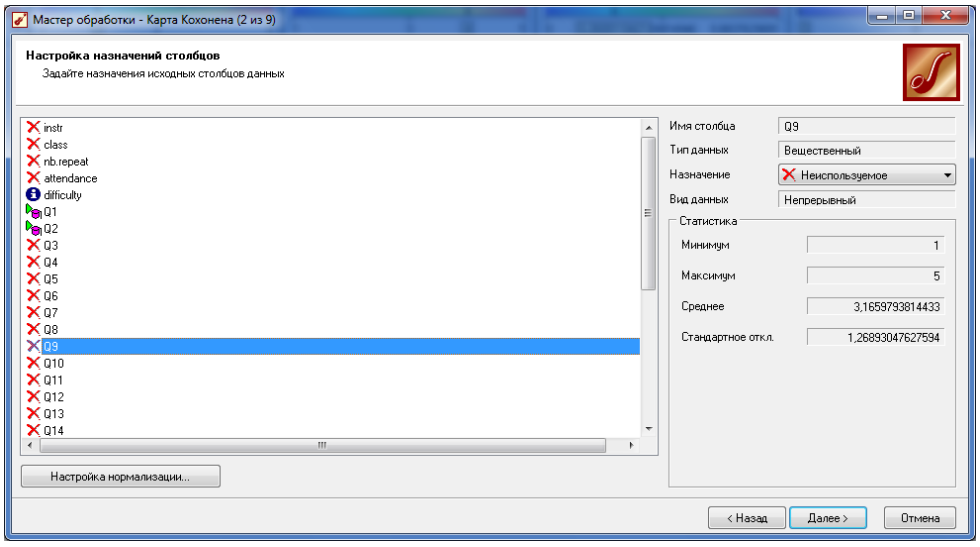
Все Q1-Q28 относятся к типу Лайкерта, что означает, что значения взяты из набора 1,2,3,4,5}.

Чтобы приступить к работе, нужно запустить Мастер обработки и в разделе «Data Mining» выбрать пункт «Карта Кохонена».

На 2-м шаге требуется выбрать назначение столбцов обучающего набора. При построении SOM используется обучение без учителя, поэтому выбирать выходную переменную нет смысла, поскольку модель на основе SOM не имеет выхода как такового, а просто визуализирует распределение значений признаков объектов в проекции многомерной кластерной структуры на плоскость.

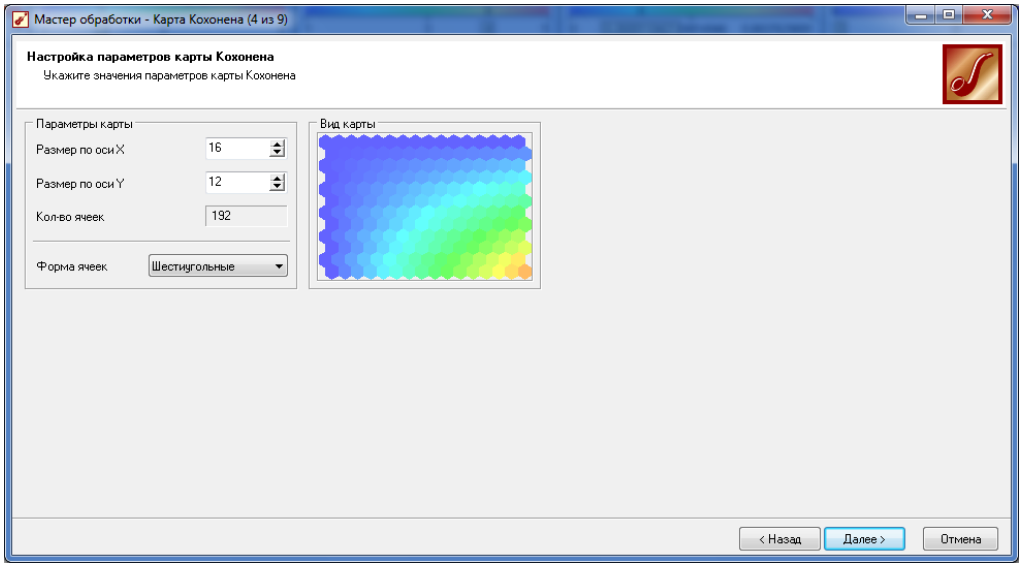
Таким образом, все столбцы, которые мы хотим использовать для построения карты, следует выбрать в качестве входных. Выберем для примера значения Q1 и Q2.

Столбец «Difficulty» отразим как «Информационное». Информационные столбцы в аналитической системе Deductor не используются при построении модели, но отображаются вместе с результатами (в отличие от столбцов с назначением «Неиспользуемые», которые не используются никаким образом).



На 3-м шаге требуется настроить параметры разделения обучающего набора данных на обучающее и тестовое множества.

На 4-м шаге следует настроить параметры карты: число ячеек по вертикали и горизонтали, а также выбрать форму ячеек (шестиугольную или прямоугольную)



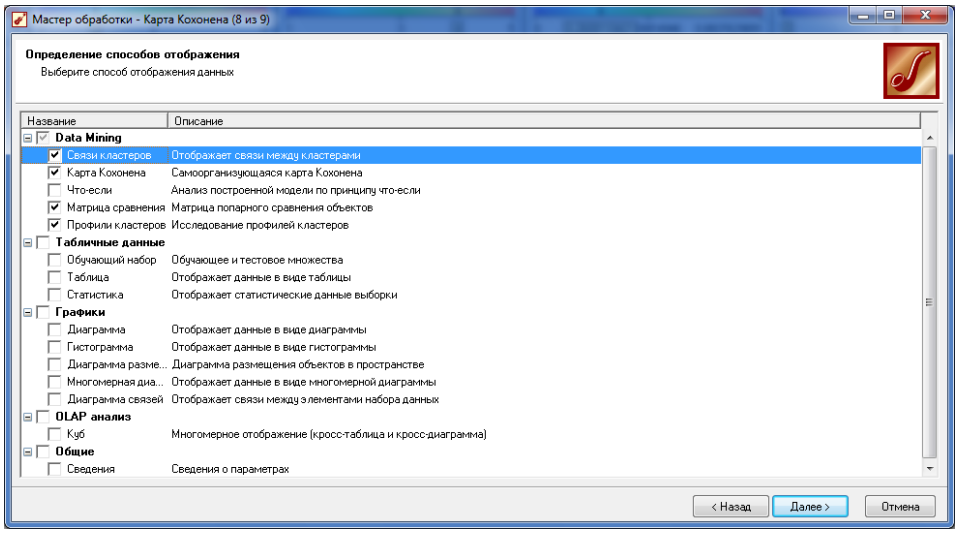
Размер карты, наилучшим образом отображающий структуру кластеров, заранее неизвестен и вообще субъективен. Тем не менее, можно рекомендовать начальное число ячеек в 2-3 раза больше числа обучающих примеров. Что касается формы ячеек, то, как отмечено выше, шестиугольная форма более

корректно отображает расстояния на карте, соответствующие расстояниям в многомерном пространстве.

На 5-м шаге задаются параметры остановки обучения. Величина, называемая здесь ошибкой, представляет собой среднее расстояние между векторами объектов и центрами кластеров, в которые они попали в процессе обучения. Если объект расположен вблизи центра кластера, это повышает уверенность, что объект действительно является членом данного кластера и разделяет свойства остальных его членов. Напротив, если объект оказался на периферии кластера (т.е. больше «похож» на объекты из соседнего кластера), степень уверенности в общ-ости свойств данного объекта и других объектов кластера снижается.

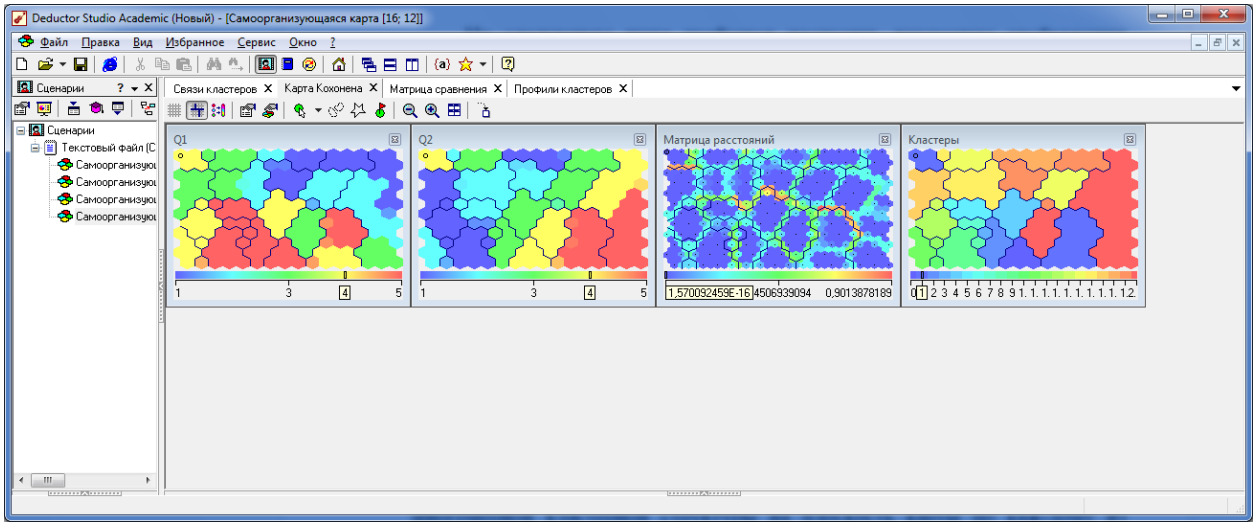
На 6-м шаге задаются параметры обучения карты. В большинстве случаев их оптимальные значения заранее неиз-вестны и требуют экспериментального подбора. Способ начальной инициализации карты определяет, каким об-разом присваиваются начальные веса её нейронам. От того, насколько удачно это будет сделано, зависят результат обучения и число итераций, которые для этого потребуются.

На 8-м шаге следует выбрать способ визуализации карты. Для этого в списке доступных визуализаторов в секции Data Mining установить флажок «Карта Кохонена»



На следующем шаге требуется настроить параметры отображения карты: выбрать отображаемые проекции признаков обучающего набора (в общем случае их может быть много, а интерес для анализа представляют лишь некоторые), специальные визуализаторы (для опытных пользователей), а также свойства визуализации самой карты. Способ раскрашивания ячеек карты позволяет выбрать цветную палитру или оттенки серого. Сглаживание цветов карты, выделение границ ячеек и кластеров, изменение размера ячеек позволяют сделать карту удобнее для визуального восприятия.

После завершения Мастера обработки будут открыты окна с выбранными элементами карты и можно будет приступать к содержательной интерпретации сформированной кластерной структуры



Отображение, показанное на рисунке, представляет собой проекции многомерной кластерной структуры на плоскость карты по каждому из признаков, которые использовались при её обучении. Расстояние между точками карты пропорционально расстоянию между векторами объектов по соответствующей координате. Т.е. расстояние на карте «Q1» отражает разность именно по значению Q1, на карте «Q2» - по содержанию Q2 и т.д. Такой способ отображения позволяет представлять информацию, содержащуюся в сложной для восприятия многомерной структуре, в виде набора простых 2-мерных структур.

Щелкнув мышью по любому шестиугольнику на карте, получаем выделение соответствующих ячеек на других картах.

Цвет на карте отображает значение признака, по которому она построена. Причем цвет относится не к отдельным объектам, а к ячейкам, т.е. цветом заливается вся ячейка. Значение оттенка цветовой гаммы определяется на основе среднего значения признака попавших в ячейку примеров. В нижней части карты представлена полоса с палитрой, на которой отображается значение признака в ячейке, где установлен маркер. Меньшие значения признака отображаются синим цветом, большие – красным.

Рассмотрим несколько случаев.

1. Объекты на карте расположены далеко друг от друга - в ячейках, залитых разным цветом. Это означает, что объекты значительно отличаются, причём признак, отображаемый на карте, хорошо показывает это отличие.

2. Объекты на карте расположены в её различных углах – в ячейках, залитых одинаковым цветом. Тогда объекты значительно отличаются, но признак на карте, плохо показывает это отличие.

3. Объекты на карте расположены в одной области, в ячейках залитых разным цветом. Это означает, что объекты похожи но признак, представленный на карте, плохо отражает это сходство.

4. Объекты на карте расположены в одной области, в ячейках залитых одинаковым цветом. Это означает, что объекты похожи, и признак, представленный на карте, хорошо отражает это сходство. Таким образом, при хорошей кластеризации, похожие объекты расположены близко на карте - в ячейках, залитых примерно одним цветом, а непохожие – на большом расстоянии, в ячейках залитых разными цветами. Иными словами, при хорошей кластеризации, чем дальше друг от друга находятся два объекта на карте, тем сильнее различаются цвета ячеек, в которых они расположены. В идеальном случае внутри каждого кластера должны располагаться ячейки одного цвета (возможны только вариации оттенков), а смена цвета происходит

в узкой полосе на границе кластера. Но на практике такая ситуация встречается достаточно редко.

Сами объекты на картах не отображаются. Чтобы «увидеть» объект на карте, нужно установить маркер в ячейку щелчком мыши и воспользоваться кнопкой - «Показать/скрыть окно данных». В результате под картой откроется окно с набором обучающих данных. В окне для каждого обучающего примера представлены значения признаков, по которым производилась кластеризация, а также информация о его расположении на карте: номер кластера и расстояние до его центра, номер ячейки и расстояние до её центра. Если требуется быстро найти объект на карте, достаточно выделить его строку в окне данных и воспользоваться кнопкой «Поиск ячеек на карте». В результате маркер установится в соответствующую ячейку.

При содержательной интерпретации кластеров требуется просмотреть содержимое отдельных кластеров и отдельных ячеек. Это можно реализовать с помощью фильтрации по кластеру или по ячейке. Для этого следует установить маркер в соответствующую ячейку или кластер и в меню, вызываемом кнопкой - «Изменить способ фильтрации», выбрать пункт «Фильтр по ячейке» или «Фильтр по кластеру».

После этого в окне данных останутся только строки с примерами, попавшими в помеченный маркером кластер или ячейку.

Включив фильтрацию по кластерам, мы обнаружим распределение примеров в них. Можно увидеть, что в кластере N 1 содержится примеры со всеми значениями сложности. Можно предположить, что на субъективное восприятие курса как сложного не влияют оглашение содержания семестрового курса, метода обучения, системы оценки, цели и задач курса в начале периода.

Задания для самостоятельного выполнения:

1. Выполните кластерный анализ для других признаков (не менее 3 в совокупности).

2. Определите взаимное сочетание значений признаков
3. Определите влияние каждого из признаков или их совокупности на субъективное восприятие сложности курса.

Литература:

Орешков, В. И. Интеллектуальный анализ данных : учебное пособие / В. И. Орешков. — Рязань : РГРТУ, 2017. — 160 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/168028

Лабораторная работа 3. Итерационные алгоритмы кластерного анализа

Кластерный анализ – совокупность методов, алгоритмов, подходов и процедур, разработанных для решения проблемы формирования однородных (гомогенных) групп объектов с учетом характеризующих их признаков в произвольной проблемной области.

При выявлении таких групп – кластеров (clusters) – используется неформальное предположение о том, что объекты, относимые к одному кластеру, должны иметь большее сходство между собой, чем с объектами из других кластеров.

Алгоритмы иерархической кластеризации, называемые также алгоритмами таксономии, – это алгоритмы упорядочивания, реализующие создание иерархии (дерева) вложенных кластеров для объектов анализируемого набора данных.

Итерационные алгоритмы кластерного анализа – это алгоритмы упорядочивания, реализующие поиск наилучшего разбиения объектов из анализируемого набора данных на кластеры в ходе некоторого числа итераций в соответствии с тем или иным критерием качества кластеризации.

В таких алгоритмах требуется задать некоторые начальные условия, с учетом которых и выполняется поиск. Многие из этих алгоритмов используют априори заданное число кластеров (которое, вообще говоря, может быть не оптимальным) для инициализации разбиения объектов на кластеры или, например, для инициализации координат центроидов кластеров, а затем в ходе некоторого числа итераций уточняют разбиение объектов на кластеры или координаты центроидов кластеров. Также априори может быть задан некоторый набор параметров, на основе значений которого реализуется работа итерационного алгоритма и определяется число кластеров, а также их состав, соответствующие априори заданному набору параметров.

Известно большое число итерационных алгоритмов кластерного анализа, среди которых, в первую очередь, следует назвать алгоритм k средних (k-means-алгоритм) и алгоритм нечетких c-средних (fuzzy c-means-алгоритм, FCM-алгоритм), отличительными особенностями которых является выделение объектов-эталонов (прототипов), представляющих кластеры.

Для работы с алгоритмами иерархической агломеративной кластеризации следует воспользоваться программной библиотекой, предложенной по ссылке:

https://pypi.org/project/scipy/

Для установки программной библиотеки необходимо выполнить команду:

pip install scipySciPy

SciPy предоставляет программные реализации алгоритмов для задач оптимизации, интегрирования, интерполяции, статистики, поиска решений алгебраических и дифференциальных уравнений и многих других классов задач.

Подробная документация по всем функциям, используемым в программной реализации алгоритма иерархической агломеративной кластеризации, приведена по ссылке:

https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html

Кроме того, можно использовать программную библиотеку scikit-learn, предоставляющую различные инструменты для прогнозного анализа данных, в том числе, инструменты машинного обучения. Эта библиотека доступна по ссылке:

https://scikit-learn.org/stable/

Для установки программной библиотеки необходимо выполнить команду:

pip install -U scikit-learn

Подробная документация по программной реализации k-means-алгоритма приведена по ссылке

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans

Ниже представлена информация по классу KMeans, реализующему k-means-алгоритм.

Синтаксис:

*class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='auto')*

Входные параметры:

n_clusters – число кластеров, которые нужно сформировать, а также число центроидов, которые нужно сгенерировать;

init – метод инициализации: 'k-means++' («умная» инициализация центроидов кластеров для ускорения сходимости: центроиды выбираются таким образом, чтобы получить доказуемую верхнюю границу WCSS (Within Clusters Sum ofSquares) – суммы квадратов внутри кластерных расстояний); 'random' (случайный выбор n_clusters объектов из набора данных для инициализации центроидов кластеров);

n_init – число запусков алгоритма k-средних с разными инициализациями центроидов, при этом итоговым результатом кластеризации будет лучший результат, полученный в ходе n_init последовательных запусков с точки зрения показателя инерции;

max_iter – максимальное число итераций алгоритма k-средних за один запуск;

tol – относительное допустимое отклонение для нормы Фробениуса при оценке различия в координатах центроидов кластеров на двух последовательных итерациях для объявления сходимости;

verbose – параметр, отвечающий за режим детализации;

random_state – параметр, отвечающий за генерацию случайных чисел для инициализации центроидов (None или некоторое целое число, позволяющее обеспечить воспроизводимость результатов эксперимента за счёт детерминированности);

copy_x – параметр, определяющий, будет ли создаваться резервная копия набора данных перед тем, как будут вычисляться расстояния между объектами (при вычислении расстояний целесообразно сначала центрировать данные, если параметр copy_x имеет значение True (по умолчанию), исходные данные не изменяются, если параметр copy_x имеет значение False, исходные данные изменяются и трансформируются обратно до того, как будет выведен результат, но при этом могут появиться небольшие числовые различия из-за выполнения операций вычитания и сложения; при этом есть ряд ситуаций, связанных с типом данных, при которых создается копия, даже если copy_x=False);

algorithm – используемая версия k-means-алгоритма: 'auto' (в настоящий момент – алгоритм 'elkan', который может быть в будущем заменен на другой для обеспечения лучшей эвристики); 'full' (классический алгоритм в EM-стиле);' elkan' (алгоритм, наиболее эффективный для данных с четко определенными кластерами за счет использования неравенства треугольника, но требующий больших объемов памяти из-за необходимости выделения дополнительного массива размером n_samples x n_clusters (n_samples=n, n_clusters=k));

Значения входных параметров класса KMeans, используемые по умолчанию, указаны при описании его синтаксиса. «*», фигурирующая в списке входных параметров, показывает, что параметр, стоящей перед ней можно передавать как по значению, так и по имени, а параметры, стоящие после нее, можно передавать только по имени.

Выходные параметры:

cluster_centers_ – матрица размером n_clusters xn_features (n_clusters=k, n_features=p), содержащая координаты центроидов кластеров;

labels_ – вектор длиной n_samples, где n_samples – число объектов в наборе данных (n_samples=n), содержащий метки кластеров для объектов;

inertia_ – общая сумма квадратов расстояний от объектов до центроида ближайшего к ним кластера, взвешенная по весам объектов, если они предоставлены;

n_iter_ – число итераций;n_features_in_ – число признаков во время обучения;

feature_names_in_ – вектор названий признаков длиной n_features_in_ во время обучения, определяемый, если в наборе данных названия признаков являются строками.

Параметр inertia_, содержащий значение общей суммы квадратов расстояний от объектов до центроида ближайшего к ним кластера, может быть использован при применении метода локтя Elbow при оценке оптимального числакластеров.

Методы

1. fit (X, y=None, sample_weight=None).

Реализует k-means-алгоритм.

Входные параметры:

X – матрица размером n_samples x n_features (n_samples=n, n_features=p), содержащая набор данных, на основе которого происходит обучение;

y – параметр, который не используется, присутствует здесь для согласованности API по соглашению;

sample_weight – массив длиной n_samples (n_samples=n), содержащий веса для каждого объекта набора данных (если None, всем объектам присваивается одинаковый вес).

Значения входных параметров метода fit, используемые по умолчанию, указаны при описании его синтаксиса.

Выходной параметр:

self – обученный кластеризатор.

2. fit_predict(X, y=None, sample_weight=None)

Вычисляет центроиды кластеров и предсказывает метку кластера для каждого объекта из набора данных X.

Входные параметры:

X – матрица размером n_samples x n_features (n_samples=n,n_features=p), содержащая набор данных;

y – параметр, который не используется, присутствует здесь для согласованности API по соглашению;

sample_weight – массив длиной n_samples (n_samples=n), содержащий веса для каждого объекта набора данных (если None, всем объектам присваивается одинаковый вес).

Значения входных параметров метода fit_predict, используемые по умолчанию, указаны при описании его синтаксиса.

Выходной параметр:

labels – вектор размером n_samples, содержащий метки кластеров объектов набора данных.

3. fit_transform(X, y=None, sample_weight=None)

Вычисляет результаты кластеризации и преобразует набор данных X в пространство кластерных расстояний.

Входные параметры:

X – матрица размером n_samples x n_features (n_samples=n,n_features=p), содержащая набор данных;

y – параметр, который не используется, присутствует здесь для согласованности API по соглашению;

sample_weight – массив длиной n_samples (n_samples=n), содержащий веса для каждого объекта набора данных (если None, всем объектам присваивается одинаковый вес).

Значения входных параметров метода fit_transform, используемые по умолчанию, указаны при описании его синтаксиса.

Выходной параметр:

X_new – матрица размером n_samples x n_clusters, содержащий матрицу X, переведенную в новое пространство.

4. get_params(deep=True)

Выводит параметры кластеризатора.

Входной параметр:

deep – параметр, отвечающий за возврат параметров кластеризатора (вывод осуществляется, если deep=True).Значение входного параметра метода get_params, используемое по умолчанию, указано при описании его синтаксиса.

Выходной параметр:

params – словарь, в котором имена параметров сопоставлены с их значениями.

5. predict (X, sample_weight=None)

Предсказывает ближайший кластер, к которому принадлежит каждый объект из набора данных X.

Входные параметры:

X – матрица размером n_samples x n_features (n_samples=n,n_features=p), содержащая набор данных;

sample_weight – массив длиной n_samples (n_samples=n), содержащий веса для каждого объекта набора данных (если None, всем объектам присваивается одинаковый вес).

Значения входных параметров метода predict, используемые по умолчанию, указаны при описании его синтаксиса

Выходной параметр:

labels – вектор размером n_samples, содержащий метки кластеров объектов набора данных.

6. score(X, y=None, sample_weight=None)

Вычисляет значение суммы, взятой со знаком минус, для квадратов расстояний от объектов до центроида ближайшего к ним кластера, взвешенной повесам объектов, если они предоставлены (значение выходного параметра inertia_, взятое со знаком минус).

Входные параметры:

X – матрица размером n_samples x n_features (n_samples=n,n_features=p), содержащая набор данных, на основе которого происходит обучение;

y – параметр, который не используется, присутствует здесь для согласованности API по соглашению;

sample_weight – массив длиной n_samples (n_samples=n), содержащий веса для каждого объекта набора данных (если None, всем объектам присваивается одинаковый вес).

Значения входных параметров метода score, используемые по умолчанию, указаны при описании его синтаксиса.

Выходной параметр:

score – сумма, взятая со знаком минус, для квадратов расстояний от объектов до центроида ближайшего к ним кластера, взвешенной по весам объектов, если они предоставлены (score= - inertia_).

*7. set_params(**params)*

Устанавливает параметры кластеризатора.

Входной параметр:

**params – словарь с параметрами кластеризатора.

Выходной параметр:

self – экземпляр кластеризатора.

8. transform(X)

Преобразует X в пространство кластерного расстояния.

Входной параметр:

X – матрица размером n_samples x n_features (n_samples=n,n_features=p), содержащая набор данных, на основе которого происходит обучение.

Выходной параметр:

X_new – матрица размером n_samples x n_clusters, содержащий матрицу X, переведенную в новое пространство.

Альтернативно можно использовать еще одну программную реализацию k-means-алгоритма – Mini-Batch k-Means-алгоритм.

Подробная документация по программной реализации Mini-Batch k-Means-алгоритм приведена по ссылке:https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html.

Этот алгоритм реализует кластеризацию с использованием выборок (мини-пакетов) из исходного набора данных. Размер таких выборок – batch_size, является одним из входных параметров Mini-Batch k-Means-алгоритма.

Для работы с fuzzy-c-means-алгоритмом можно воспользоваться различными программными библиотеками.

В частности, можно использовать программную библиотеку, предложенную по ссылке:

https://pypi.org/project/scikit-fuzzy/.SciKit-Fuzzy – библиотека, реализующая алгоритмы на основе теории нечетких множеств и нечеткой логики.

Для установки программной библиотеки необходимо выполнить команду:

pip install scikit-fuzzy

Подробная документация по библиотеке приведена по ссылке [17]:https://scikit-fuzzy.readthedocs.io/en/latest/api/skfuzzy.html.Информация по программной реализации fuzzy-c-means-алгоритма приведена по ссылке:

https://github.com/scikit-fuzzy/scikit-fuzzy/blob/master/skfuzzy/cluster/_cmeans.py.

Кроме того, можно использовать программную библиотеку, предложенную по ссылке:

https://pypi.org/project/fuzzy-c-means/.

Для установки программной библиотеки необходимо выполнить команду:

pip install fuzzy-c-means

Информация по программной реализации fuzzy-c-means-алгоритма приведена по ссылке:

https://github.com/omadson/fuzzy-c-means.

В целом принципы работы fuzzy-c-means-алгоритмом аналогичны принципам работы с k-means-алгоритмом.

Для работы с fuzzy-c-means-алгоритмом используется класс FCM.

Входные параметры:

n_clusters – число кластеров, которые нужно сформировать, а также число центроидов, которые нужно сгенерировать (по умолчанию n_clusters=5);

max_iter – максимальное число итераций алгоритма нечетких с-средних за один запуск (по умолчанию max_iter =150);

max_iter – фаззификатор (по умолчанию m =2);

error – относительное допустимое отклонение при оценке различия в нечетких разбиениях объектов на кластеры на двух последовательных итерациях для объявления сходимости;

random_state – параметр, отвечающий за случайный выбор n_clusters объектов из набора данных для инициализации центроидов кластеров;

trained – индикатор, определяющий факт обученности.

Выходные параметры:

centers – массив, содержащий координаты центроидов кластеров;

u – матрица, содержащая результаты нечеткого разбиения объектов на кластеры;

labels – вектор, содержащий метки кластеров объектов.

Метод fit обеспечивает обучение без учителя на основе анализируемого набора данных с использованием алгоритма нечетких с-средних.

Задание для самостоятельного выполнения:

1. Подберите подходящий набор для кластеризации на сайте https://archive.ics.uci.edu/datasets
2. Конвертируйте набор в подходящий для анализа формат
3. Выполните кластерный анализ по алгоритмам k-means и c-means

Литература

Демидова, Л. А. Кластерный анализ. Python : учебное пособие / Л. А. Демидова. — Москва : РТУ МИРЭА, 2022. — 103 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/240092

Лабораторная работа 4. Аппроксимация функции при помощи искусственных нейронных сетей

Аппроксимацией (приближением) функции f(x) называется нахождение такой функции g(x) (аппроксимирующей функции), которая была бы близка заданной. Разработано несколько критериев близости функций f(x) и g(x). Основной задачей аппроксимации является построение аппроксимирующей функции, наиболее близко проходящей около заданных точек или около графика непрерывной функции. Возникновение такой задачи связано с наличием погрешностей в исходных данных. В этом случае точно проводить функцию через все точки исходных данных, как это делается в интерполяции, нецелесообразно.

Аппроксимация позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов (например, таких, характеристики которых легко вычисляются или свойства которых уже известны).

Близость исходной и аппроксимирующей функций определяется числовой мерой – критерием аппроксимации (близости). Наибольшее распространение получил критерий близости в заданных точках, равный сумме квадратов отклонений расчетных значений от «экспериментальных». Такой критерий называется «квадратичным».

$$R = \sum_{i=1}^{n} \beta_i (y_i - g_i)^2$$

где y_i – заданные табличные значения функции в i-й точке;

g_i – значения аппроксимирующей функции в i-й точке;

βi – весовые коэффициенты, за-дающие относительную важность i-й точки таким образом, что рост βi при минимизации критерия R в первую очередь приводит к уменьшению отклонения в i-й точке.

Квадратичный критерий обладает рядом достоинств, в числе которых дифференцируемость и наличие единственного решения задачи аппроксимации при полиномиальных аппроксимирующих функциях.

Выделяют две основные задачи аппроксимации:

1. Получение аппроксимирующей функции, описывающей имеющиеся данные, с погрешностью не выше заданной.

2. Получение аппроксимирующей функции заданной структуры с наилучшей возможной погрешностью.

Чаще всего первая задача сводится ко второй перебором различных аппроксимирующих функций и последующим выбором наилучшей.

Нейросетевое решение задачи аппроксимации

Как было показано выше, аппроксимация может сводиться к нахождению функциональной зависимости по имеющемуся набору точек. В данном случае, функциональная зависимость будет представлена в нейросетевом базисе, т.е. через комбинацию активационных функций нейронных сетей.

Обучение нейронной сети

В теории нейронных сетей существуют две актуальных проблемы, одной из которых является выбор оптимальной структуры нейронной сети, а другой построение эффективного алгоритма обучения нейронной сети.

Оптимизация нейронной сети направлена на уменьшение объёма вычислений при условии сохранения точности решения задачи на требуемом уровне. Параметрами оптимизации в нейронной сети могут быть:

- размерность и структура входного сигнала нейросети;
- синапсы нейронной сети (они упрощаются с помощью удаления из сети или заданием «нужной» или «оптимальной» величины веса синапса);
- количество нейронов каждого слоя сети: нейрон целиком удаляется из сети, с автоматическим удалением тех синапсов нейронов следующего слоя, по которым проходил его выходной сигнал;

- количество слоёв сети.

Вторая проблема заключается в разработке качественных алгоритмов обучения нейросети. Обучение нейронных сетей – это подбор весовых коэффициентов нейронов. Тип и характер обучения определяются, прежде всего, объемом априорной и текущей информации о среде, в которую «погружена» сеть, а также критерием качества (целевой функцией), при котором свободные параметры нейронной сети адаптируются в результате ее непрерывной стимуляции внешним окружением.

Можно выделить два основных вида обучения: обучение с «учителем», и обучение «без учителя».

Алгоритм обратного распространения ошибки

Обучение многослойного персептрона чаще всего происходит с помощью алгоритма обратного распространения ошибки или его модификации.

Обратное распространение ошибки – системный метод обучения ИНС. В основу метода положено обучение ИНС по образцам с помощью примеров обучающей выборки. Для каждого примера из обучающей выборки считается известным требуемое (целевое) значение выхода ИНС, то есть метод обратного распространения ошибки представляет собой обучение с учителем.

Такое обучение можно представить как решение оптимизационной задачи на минимизацию функции ошибки (невязки) E на всей обучающей выборке путем подстройки обучаемых параметров ИНС – значений весов синапсов w, параметров активационных функций.

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(y_i^o - y_i\right)^2$$

где y^o_i – ожидаемое значение выхода на i-м образце выборки,

y_i – рассчитываемое значение искусственной нейронной сетью,

n – число образцов обучающей выборки.

Минимизация функционала ошибки E в методе обратного распространения ошибки осуществляется с помощью одного из градиентных методов оптимизации, например, «наискорейшего спуска» или его модификаций. При этом веса синапсов изменяются в направлении, обратном направлению наибольшего возрастания функции ошибки – в направлении антиградиента E:

$$W(t+1) = W(t) - \eta \frac{\partial E}{\partial W}$$

где $0 < \eta \leq 1$ – «шаг обучения», определяемый пользователем параметр, определяющий скорость обучения, в ряде методов он подбирается итерационно.

Метод обратного распространения ошибки реализуют в двух вариантах. В первом варианте обучаемые параметры пересчитываются после подачи всей обучающей выборки, и ошибка имеет вид

$$E = \frac{1}{2} \sum_{i=1}^{n} \left(y_i^o - y_i \right)^2$$

Во втором варианте ошибка пересчитывается после каждого примера:

$$E_i = \frac{1}{2} \left(y_i^o - y_i \right)^2$$

Установим

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}},$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

Тогда

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_{ij}} \cdot \frac{\partial y_i}{\partial S_j} \cdot \frac{\partial S_j}{\partial w_{ij}},$$

$$y_j = f\left(S_j\right) = \frac{1}{1+e^{-S_j}},$$

где y_i – функция активации

$$f'\left(S_j\right) = \frac{\partial}{\partial S_j}\left(\frac{1}{1+e^{-S_j}}\right) = \frac{1}{\left(1+e^{-S_j}\right)^2}\left(-e^{-s_j}\right) = \frac{1}{1+e^{-S_j}} \cdot \frac{-e^{-S_j}}{1+e^{-S_j}} = y_j\left(1-y_j\right)$$

$$S_j = \sum W_{ij} y_i^{n-1} \frac{\partial S_i}{\partial W_{ij}} = y_i^{(n-1)},$$

$$\frac{\partial E}{\partial y_i} = \sum_k \frac{\partial E}{\partial y_k} \frac{dy_k}{dS_k} \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \frac{dy_k}{dS_k} W_{jk}^{(n+1)}$$

при этом суммирование по k идет среди нейронов n-го слоя.

Введем новое обозначение

$$\delta_j = \frac{\partial E}{\partial y_k} \frac{dy_j}{dS_j};$$

$$\delta_j^{n-1} = \left[\sum_k \delta_k^n W_{jk}^n\right] \frac{dy_j}{dS_j}.$$

Для внутреннего нейрона

$$\delta_j^n = (d_j^n - y_j^n) \frac{dy_j}{dS_j}$$

Для внешнего нейрона

$$\Delta W_{ij}^n = -\eta \delta^{(n)} y_j^{n-1};$$

$$W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij}$$

Обучение по методу обратного распространения ошибки производится в соответствии со следующим алгоритмом:

Шаг 1. Производится инициализация начальных значений обучаемых параметров. Обычно начальные значения составляют малые случайные величины.

Шаг 2. На входы ИНС подается один из примеров обучающей выборки. В режиме обычного функционирования ИНС (сигналы распространяются от входов к выходам) производится расчет значений выходов всех нейронов.

Шаг 3. Рассчитываются значения δ^n_j для нейронов выходного слоя по формуле для внешнего нейрона.

Шаг 4. Рассчитываются значения δ^n_j для всех внутренних нейронов по формуле.

Шаг 5. Вычисляются приращения весовых коэффициентов ΔW^n_{ij} .

Шаг 6. Скорректировать веса синапсов $W_{ij}(t+1)=W_{ij}(t)+\Delta W_{ij}$

Шаг 7. Повторить шаги 2–6 для каждого примера обучающей выборки до тех пор, пока значение функционала ошибки E не станет достаточно маленькой.

Альтернативным условием останова цикла на шаге 7 является невозможность обучить ИНС до приемлемой точности. Это решение принимается исходя из того, что приращения весовых коэффициентов ΔW^n_{ij} станут слишком малы и количество итераций в этой точке больше заданного.

Такая ситуация может возникнуть в следующих двух случаях:

1. Алгоритм оптимизации попал в локальный экстремум функционала ошибки E. Для того, чтобы исключить данную возможность, необходимо стартовать обучение из другой стартовой конфигурации обучаемых параметров.

2. Архитектура ИНС недостаточна для решения задачи. Если многократный запуск обучения сети все-таки не привел к успешному обучению, то, скорее всего, следует усложнить сеть – увеличить количество нейронов или даже количество слоев.

Рассмотрим один из вариантов обучения искусственной нейронной сети

1:

```
import random

def train(X, y, syn0, lens0, lens01, lr):
    errorreturn = 0
    for i in range(0, lens0):
        a = 0
        for j in range(0, lens01):
            a = a + syn0[i][j] * X[j]
        b = 1 / (1 + 2.71828182845904523536028747135266249775 7 ** -a)
        error = y[i] - b
        errorreturn = errorreturn + abs(error)
        delta = (error) * 1 / (1 +
2.71828182845904523536028747135266249775 7 ** -b) * lr
        for z in range(0, lens01):
            syn0[i][z] = syn0[i][z] + X[z] * delta
    return syn0, errorreturn

def predict(syn0, X, lens0, lens01):
    ret = []
    for i in range(0, lens0):
        a = 0
        for j in range(0, lens01):
            a = a + syn0[i][j] * X[j]
        b = 1 / (1 + 2.71828182845904523536028747135266249775 7 ** -a)
        ret.append(b)
    return ret

def trainmass(X, y, syn, lens0, lens01, lr, iter):
    for i in range(iter):
        for j in range(lens01):
            syn, er = train(X[j], y[j], syn, lens0, lens01, lr)
    return syn, er

def create(x, y):
    syn0 = []
    for z in range(0, x):
        h = []
        for i in range(0, y):
            h.append(random.uniform(-0.1, 0.1))
        syn0.append(h)
    return syn0

syn = create(2, 4)
X = [[0.1, 0.2, 0.3, 0.4],[0.2, 0.3, 0.4, 0.5],[0.4, 0.5, 0.6, 0.7],[0.5,0.6,0.7,0.8]]
y = [[0.5, 0.6],[0.6, 0.7],[0.8, 0.9],[0.9, 1]]
g, r = trainmass(X, y, syn, 2, 4, 1, 60000)
h = predict(syn, X[0], 2, 4)
h2 = predict(syn, [0.5, 0.6, 0.7, 0.8], 2, 4)
print(h)
print(h2)
```

Функция train проводит обучение, функция predict обрабатывает входные данные уже обученными синапсами и выдает ответ, функция trainmass – запускает обучение по всем входным данным, функция create создает массив случайных синапсов для обучения. Массив X содержит 4 набора входных данных и массив y содержит 4 набора выходных данных.

После запуска имеем вывод:

```
«[0.4835439912709079, 0.5593653075469635]
[0.8848108739666853, 0.9682956213528969]
```

Первый массив – это ответ на входные данные X[0] – [0.1, 0.2,0.3, 0.4]. Второй массив – это ответ на данные, которые не были в обучающей выборке [0.5, 0.6, 0.7, 0.8].

И методом обратного распространения ошибки находит соответствующие x1, x2, x3, x4. Для второго выходного значения создаются другие y1, y2, y3, y4.

Рассмотрим пример аппроксимации гладкой функции одной переменной:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import random
import numpy as np

for i in range(1000):
    x = (np.random.random(1000)-0.5) * 10
    y = (x-1)*np.cos(3*x-15)

regressor = Sequential()
regressor.add(Dense(units=20, activation='relu', input_dim=1))
regressor.add(Dense(units=20, activation='relu'))
regressor.add(Dense(units=20, activation='relu'))
regressor.add(Dense(units=20, activation='relu'))
regressor.add(Dense(units=20, activation='relu'))
regressor.add(Dense(units=20, activation='relu'))
regressor.add(Dense(units=1, activation='linear'))
regressor.compile(loss='mse', optimizer='adam')
log = regressor.fit(x,y, epochs=1000,steps_per_epoch = 100, verbose = 0)
x = (np.random.random(1000)-0.5)*10
x = np.sort(x)
y = (x-1)*np.cos(3*x-15)
plt.plot(x, y)
plt.plot(x, regressor.predict(x))
#plt.plot(log.history['loss'])
plt.show()
```

На рис. 3.1–3.2 представлены результаты аппроксимации функции с различными функциями активации, а также приведены графики функции потерь

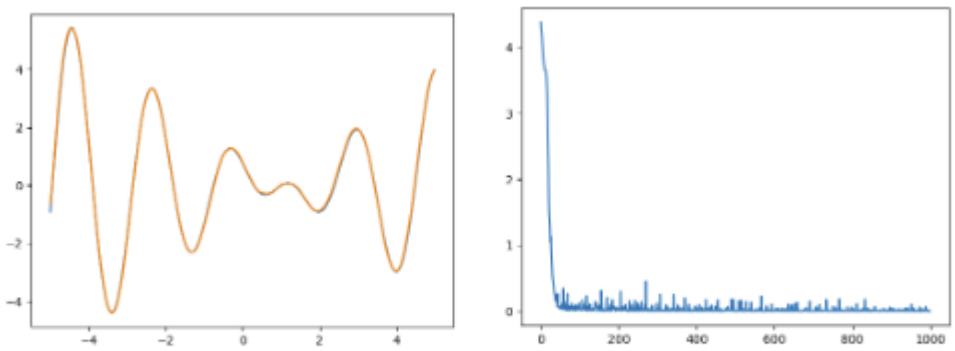


Рис. 3.1. Результат аппроксимации при функции активации ReLu
со средним квадратом ошибок и график функции потерь (средний квадрат ошибок)

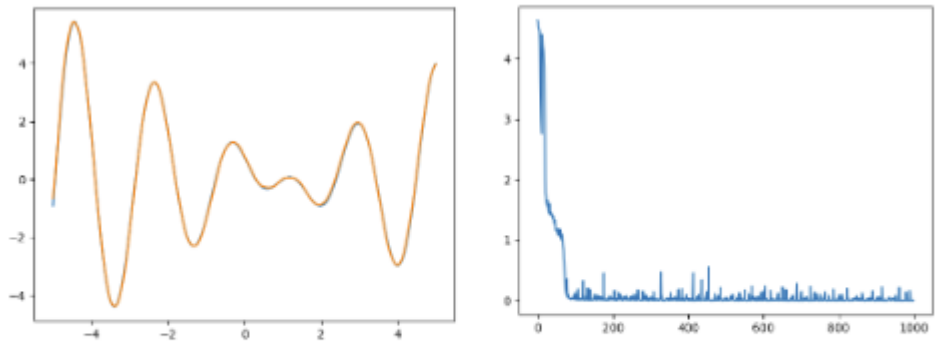


Рис. 3.2. Результат аппроксимации функции, при функции активации гиперболического тангенса и график функции потерь

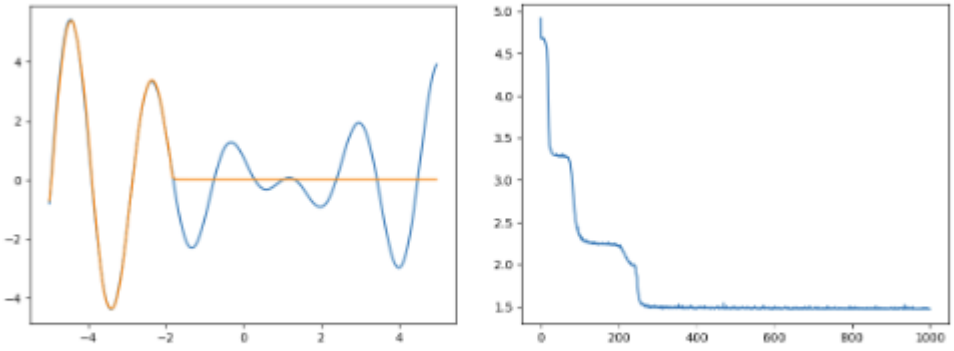


Рис. 3.2. Результат аппроксимации функции, при сигмоидальной функции активации и график функции потерь

Порядок выполнения работы

1. Для заданной функции (табл. 1) построить ее табличные значения (количество значений должно быть достаточным для того, чтобы аппроксимированная функция визуально совпадала с табличными значениями).

2. Использовать нейронную сеть для аппроксимации этих табличных значений и нахождения аппроксимированной функции. Обучение нейронной сети выполнять с помощью функции train.

3. Вывести на график табличные значения (точками) и аппроксимированную функцию (линией).

4. Найти численное значение погрешности результата аппроксимации, вывести на экран.

Таблица 1. Индивидуальные задания

Вариант	Вид функции	Промежуток нахождения решения	Контрольный вопрос
1	$(1.85-x)\cdot\cos(3.5x-0.5)$	$x\in[-10,10]$	1
2	$\cos(\exp(x))/\sin(\ln(x))$	$x\in[2,4]$	2
3	$\sin(x)/x2$	$x\in[3.1,20]$	3
4	$\sin(2x)/x2$	$x\in[-20,-3.1]$	4
5	$\cos(2x)/x2$	$x\in[-20,-2.3]$	5
6	$(x-1)\cos(3x-15)$	$x\in[-10,10]$	6
7	$\ln(x)\cos(3x-15)$	$x\in[1,10]$	1
8	$\cos(3x-15)/\text{abs}(x)=0$	$x\in[-10,-0.3),(0.3,10]$ $x\in[-0.3,0.3]$	2
9	$\cos(3x-15)\cdot x$	$x\in[-9.6,9.1]$	3
10	$\sin(x)/(1+\exp(-x))$	$x\in[0.5,10]$	4
11	$\cos(x)/(1+\exp(-x)$	$x\in[0.5,10]$	5
12	$(\exp(x)-\exp(-x))\cos(x)/(\exp(x)+\exp(-x))$	$x\in[-5,5]$	6
13	$(\exp(-x)-\exp(x))\cos(x)/(\exp(x)+\exp(-x))$	$x\in[-5,5]$	1
14	$\cos(x-0.5)/\text{abs}(x)$	$x\in[-10,0),(0,10]$, min	2
15	$\cos(2x)/\text{abs}(x-2)$	$x\in[-10,2),(2,10]$, max	3

Контрольные вопросы и задания для самоподготовки

1. Дайте определение понятия «аппроксимация».

2. Опишите параметры обучения при использовании для обучения функции train.

3. Какие вы знаете виды прекращения обучения сети при использовании функции обучения train?

4. Охарактеризуйте способ обучения, приведенный в примере.

5. Какие вы знаете способы нахождения погрешности результата?

Литература:

Доррер, М. Г. Моделирование нейронный сетей на языке Python: Лабораторный практикум для студентов бакалавриата по направлениям подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04

«Программная инженерия» всех форм обучения : учебное пособие / М. Г. Доррер, Г. Ш. Шкаберина, А. В. Коробко. — Красноярск : СибГУ им. академика М. Ф. Решетнёва, 2022. — 76 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/330107 (

Лабораторная работа 5. Поиск ассоциативных правил

Одним из важнейших свойств интеллекта человека является ассоциативность мышления. Поэтому задача обнаружения ассоциативных связей между объектами и событиями представляет огромный интерес с точки зрения поиска знаний.

Одним из популярных методов ИАД является аффинитивный анализ (от анг. affinity - близость, сходство). В его основе лежит исследование связи между событиями, которые происходят совместно, а цель — найти правила для количественного описания взаимной связи между двумя или более событиями. Такие правила называются ассоциативными правилами (АП) и широко применяются в различных областях анализа данных.

Типичное АП, устанавливающее связь между двумя событиями A и B, происходящими одновременно, записывается в виде:

$A \rightarrow B$,

что читается «Если A , то B», где событие A называют условием (antecedent), а событие B - следствием (consequent). Следовательно, поиск АП должен производиться на некотором наборе логически связанных событий. Такие наборы называются транзакциями.

Транзакция - некоторое множество событий, происходящих одновременно, например приобретение набора товаров в супермаркете по одному чеку, набор услуг, оказываемых на СТО по одному заказ-наряду, и т.д.

Анализ рыночной корзины — это анализ транзакционных наборов данных для определения комбинаций связанных между собой событий. Иными словами, производится поиск событий, присутствие которых в транзакции влияет на вероятность наличия других событий или их комбинаций. Например, событием является покупка товара или обнаружение неисправности. Зная устойчивые комбинации товаров, покупаемых совместно, можно

оптимизировать ассортимент товаров. Зная типичные наборы неисправностей, можно оптимизировать ремонт и обслуживание техники.

АП количественно описывают связь между наборами предметов, которые соответствуют условию и следствию. Эта связь описывается двумя величинами — поддержкой (support) и достоверностью (confidence). Обозначим транзакционную БД как D, а число транзакций как N. Тогда каждая транзакция Di включает несколько предметов (item), образующих предметный набор (itemset).

Поддержкой АП (в литературе обозначается буквой S - support) называется число транзакций, которые содержат одновременно условие и следствие, отнесённое к общему числу транзакций.

Например, для ассоциации $A \to B$ можно записать:

$$S(A \to B) = P(A \cap B) = \frac{N(AB)}{N}$$

т.е. число транзакций, в которых появляются предметы A и B, отнесенное к общему числу транзакций.

Достоверностью АП $A \to B$ является:

$$C(A \to B) = P(A \mid B) = \frac{P(A \cap B)}{P(A)}$$

т.е. число транзакций, в которых появляются предметы A и B, отнесенное к числу транзакций, содержащих только A.

Если поддержка и достоверность достаточно высоки, можно с большой вероятностью утверждать, что любая будущая транзакция, которая включает условие, будет также содержать и следствие.

«Правила, для которых значения поддержки или достоверности превышают заданный пользователем порог, называются сильными правилами (strong rules). Например, аналитика может интересовать, какие товары, покупаемые вместе в супермаркете, образуют ассоциации с минимальной поддержкой 20 % и минимальной достоверностью 70 %.

Методика построения ассоциативной модели использует специфический вид данных – транзакции как совокупность логически связанных событий, а не признаковое описание, когда признаки упорядочены в типизированные столбцы таблицы.

Кроме этого, ассоциативная модель работает только с категориальными данными (наименованиями предметов, объектов). Тем не менее, в большинстве случаев существует возможность преобразования признакового описания в транзакционное. Для этого надо поставить вопрос: что является событием в решаемой задаче и как события могут быть логически и формально связаны. Например, в задаче о рыночной корзине событием является появление товара в наборе, покупаемом по одному чеку, а для идентификации транзакции можно использовать номер чека.

Методики поиска АП обнаруживают все ассоциации, которые удовлетворяют ограничениям на поддержку и достоверность. Это приводит к необходимости рассматривать громадное количество ассоциаций, что делает невозможным обработку такого количества данных вручную. Число возможных ассоциаций с увеличением числа предметов растет экспоненциально.

На практике при поиске АП используются методы, позволяющие уменьшить количество подмножеств, которое требуется рассмотреть. Одним из наиболее распространенных является метод, основанный на обнаружении так называемых частых (популярных) предметных наоров (frequent itemset). В основе данного метода лежит алгоритм Apriori, который стал родоначальником целого семейства методов и алгоритмов, использующих концепцию частых предметных наборов (AprioriGen, FP-Grouth, иерархические АП и др.).

Под частотой в данном случае понимается количество транзакций, в которых содержится данный набор. В основе работы алгоритма Apriori лежит свойство антимонотонности множеств, в соответствии с которым любое подмножество частого подмножества также является частым. Таким образом,

для поиска АП алгоритм Apriori анализирует не все возможные ассоциации, а только те, в которых есть частые подмножества.

Если условие и следствие независимы, то поддержка правила примерно соответствует произведению поддержек условия и следствия. Это значит, что хотя условие и следствие часто встречаются вместе, не менее часто они встречаются и по отдельности. Например, если товар A встречался в 70 транзакциях из 100, а товар B — в 80 и в 50 транзакциях из 100 они встречаются вместе, то, несмотря на высокую поддержку (SAB = 0,5), это не обязательно правило. Просто эти товары покупаются по отдельности, но в силу их популярности часто встречаются и вместе.

Однако если условие и следствие независимы, то правило вряд ли представляет интерес, даже несмотря на высокие значения поддержки и достоверности. Действительно, если статистика дорожно-транспортных происшествий показывает, что из 100 аварий в 80 участвуют автомобили марки ВАЗ, то, на первый взгляд, это выглядит, как правило, «если авария, то ВАЗ». Но поскольку количество автомобилей ВАЗ составляет, скажем, 80 % от общего числа легковых автомобилей, то данное правило вряд ли можно назвать значимым.

По этой причине при поиске ассоциативных правил используют дополнительные показатели, позволяющие оценить их значимость. Эти показатели могут быть объективными и субъективными. Объективными показателями являются поддержка и достоверность, которые могут применяться независимо от конкретного приложения. Субъективные связаны со специальной информацией, определяемой пользователем в контексте решаемой задачи. Такими субъективными мерами являются лифт (lift) и левередж (от англ. leverage — плечо, рычаг).

Лифт представляет собой отношение частоты появления условия в транзакциях, которые также содержат и следствие, к частоте появления следствия в целом:

$$L(A \rightarrow B) = C(A \rightarrow B) / S(B)$$

Если L>1, то условие чаще появляется в транзакциях, содержащих следствие, чем в остальных. Можно сказать, что лифт является обобщенной мерой связи двух предметных наборов: при L>1 связь положительная, при 1 она отсутствует, а при L<1 — отрицательная.

Пример выполнения задания

Необходимо построить ассоциативную модель для анализа продаж. Для анализа используем набор данных Online Retail II, который содержит все транзакции, произошедшие в британской зарегистрированной онлайн-торговле вне магазинов в период с 12.01.2009 по 12.09.2011. Компания в основном продает уникальные подарочные изделия на все случаи жизни. Многие клиенты компании являются оптовиками.

Набор данных взят из открытого источника http://archive.ics.uci.edu/dataset/502/online+retail+ii

Набор данных содержит столбцы:

Invoice: Шестизначный целый номер, уникально присваиваемый каждой транзакции. Если этот код начинается с буквы «c», это означает отмену.

StockCode: код продукта (товара) - Пятизначный целый номер, уникально присвоенный каждому отдельному продукту.

Description: Название продукта (товара).

Quantity: количество каждого продукта (товара) за транзакцию. Числовой.

InvoiceDate: дата и время выставления счета. Числовой. День и время создания транзакции.

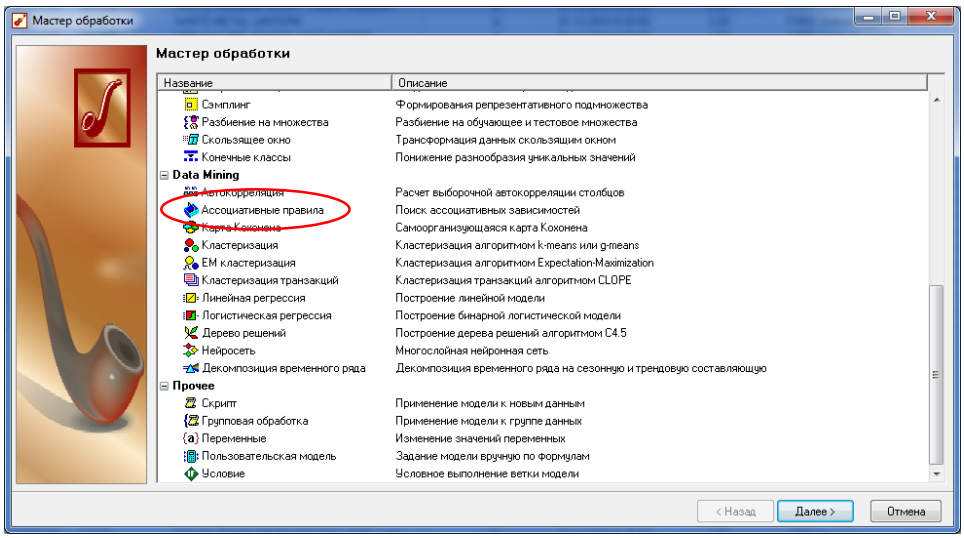
UnitPrice: цена за единицу. Числовой. Цена продукта за единицу в фунтах стерлингов.

CustomerID: номер клиента. Номинал. Пятизначный целый номер, уникально присваиваемый каждому клиенту.

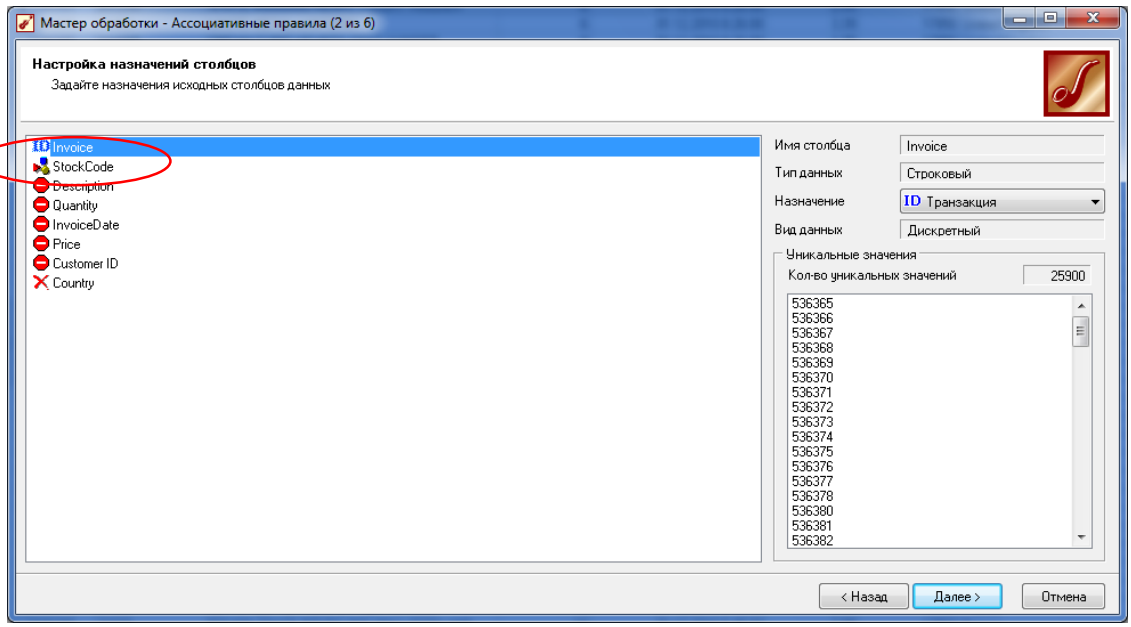
Страна: Название страны. Номинал. Название страны, в которой проживает клиент.

Для поиска ассоциативных правил воспользуемся платформой Deductor.

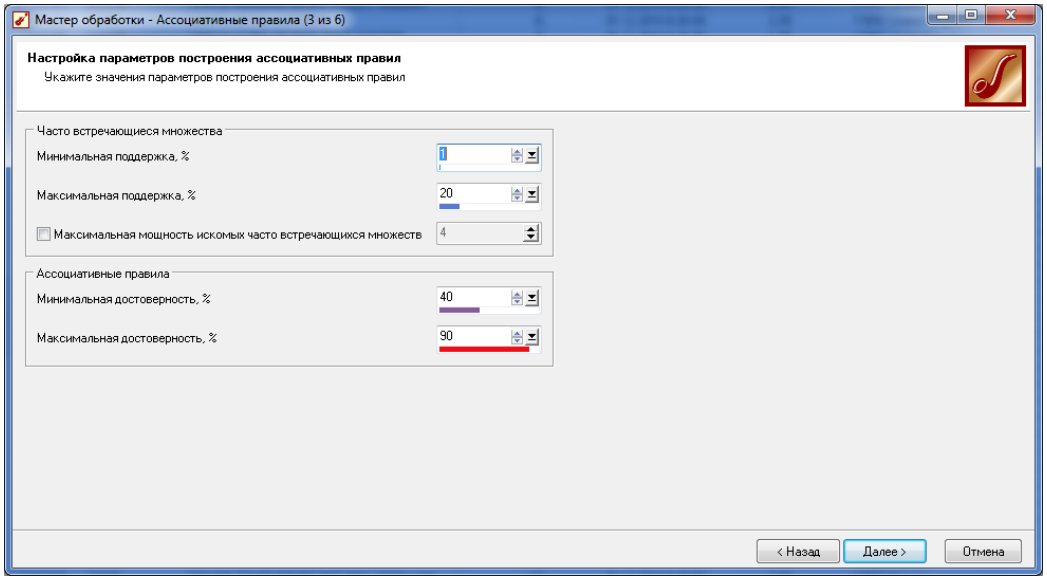
После загрузки данных для анализа выбираем анализ вида «Ассоциативные правила»:



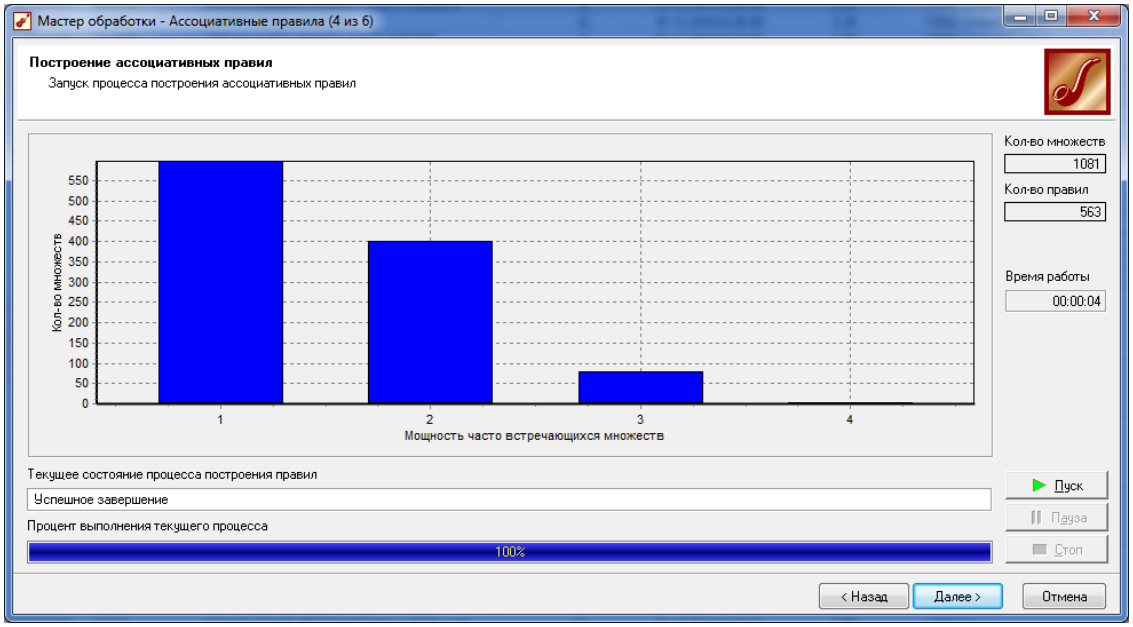
Набор обучающих данных для построения ассоциативной модели имеет специфический формат и должен содержать всего два поля: идентификатор транзакции и предмет.Для анализа ассоциаций в данном наборе данных подходящими являются данные в полях Invoice и StockCode. Поэтому на втором шаге анализа выбираем именно эти столбцы:



На 3-м шаге нужно выбрать верхнюю и нижнюю границы поддержки и достоверности для нахождения частых предметный наборов, а также для обнаружения на их основе АП.

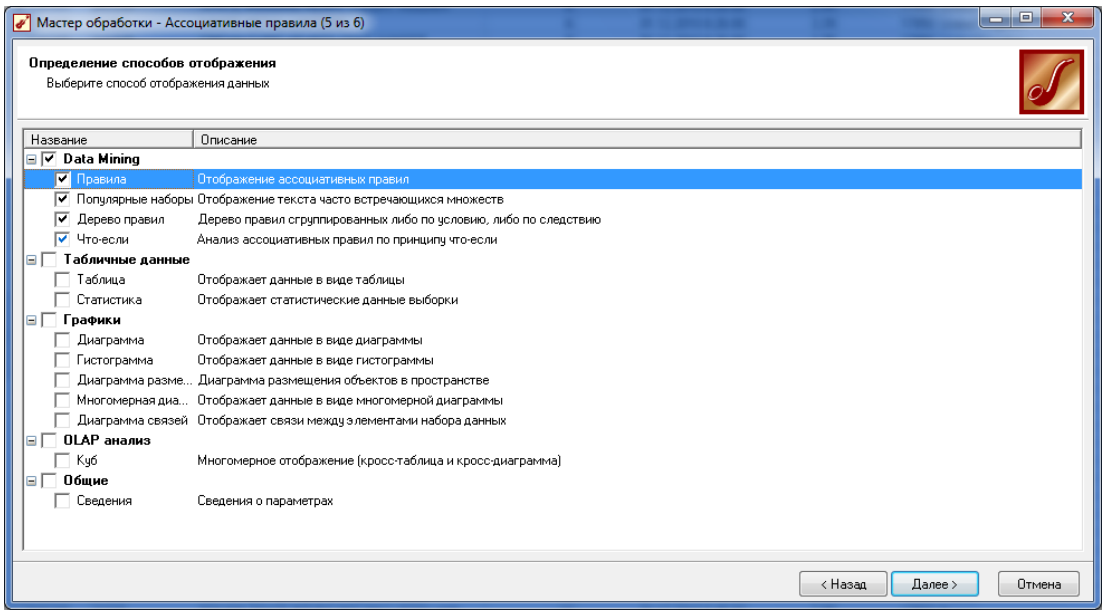


«На следующем, 4-м, шаге производится запуск процесса поиска ассоциативных правил и управление им. На графике отображается диаграмма, позволяющая интерактивно наблюдать количество и мощность предметных наборов, обнаруженных алгоритмом.

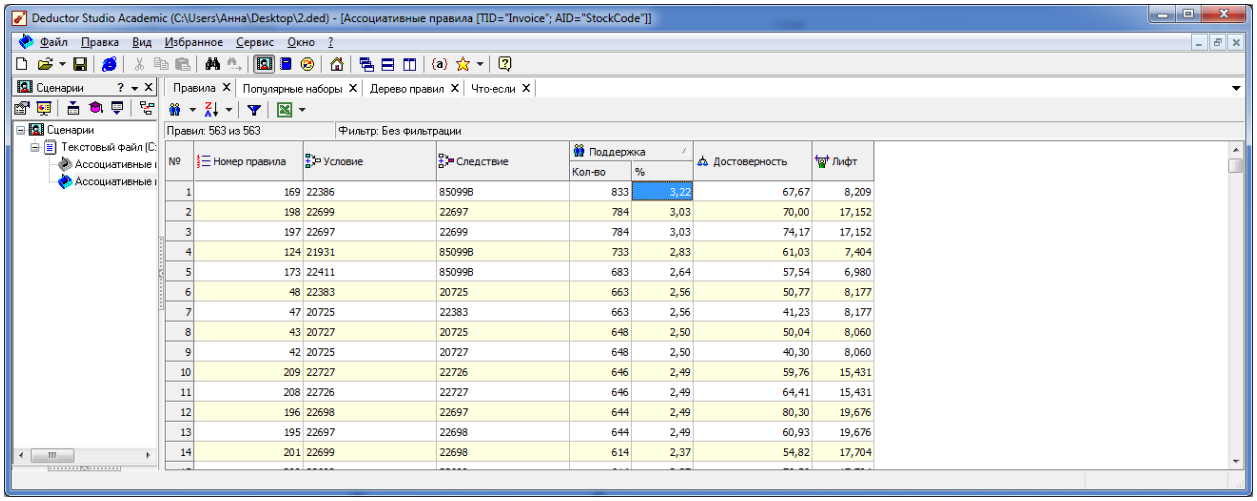


Как видно из рисунка, программа обнаружила более 600 1-предметных наборов, 400 2-предметных, около 80 3-предметных.

На 5-м шаге следует выбрать способ визуализации обнаруженных АП.



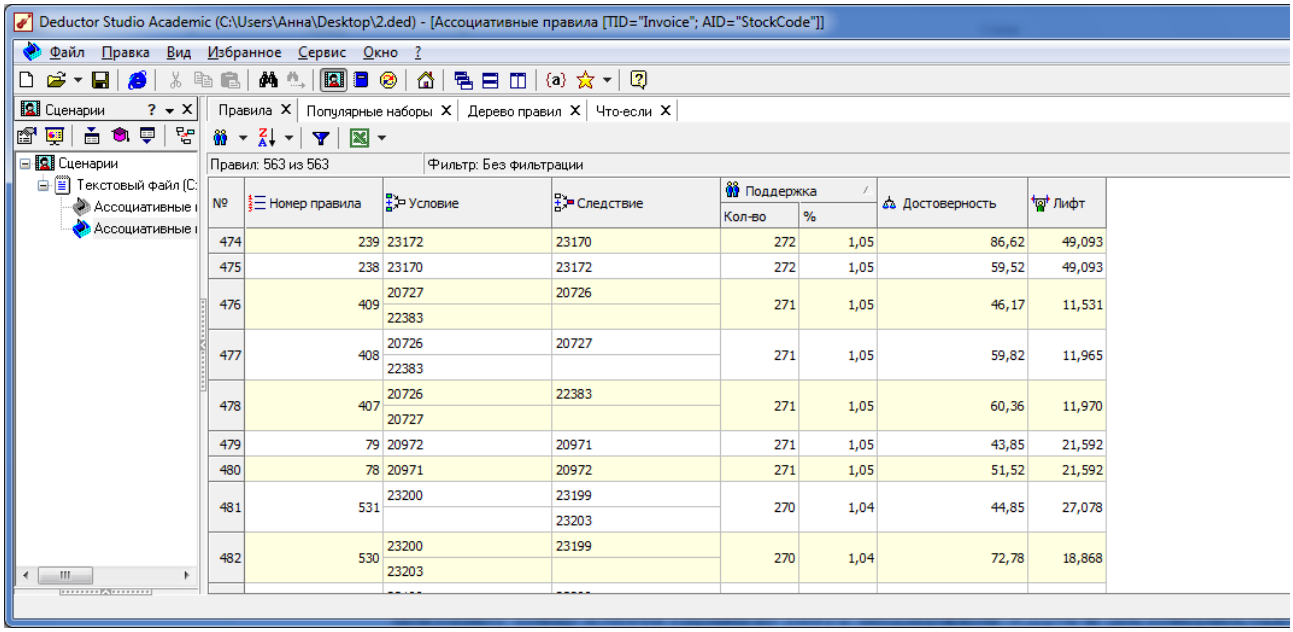
Для интерпретации и понимания результатов наиболее удобным видом представления являются «Правила». При данном способе представления для каждого правила показываются условие и следствие, а также его характеристики, по кото-рым аналитик может судить о значимости правила: поддержку, достоверность и лифт. При этом имеется возможность отображать только те правила, в которых участвуют интересующие нас предметы.



№	Номер правила	Условие	Следствие	Поддержка Кол-во	Поддержка %	Достоверность	Лифт
1	169	22386	85099B	833	3,22	67,67	8,209
2	198	22699	22697	784	3,03	70,00	17,152
3	197	22697	22699	784	3,03	74,17	17,152
4	124	21931	85099B	733	2,83	61,03	7,404
5	173	22411	85099B	683	2,64	57,54	6,980
6	48	22383	20725	663	2,56	50,77	8,177
7	47	20725	22383	663	2,56	41,23	8,177
8	43	20727	20725	648	2,50	50,04	8,060
9	42	20725	20727	648	2,50	40,30	8,060
10	209	22727	22726	646	2,49	59,76	15,431
11	208	22726	22727	646	2,49	64,41	15,431
12	196	22698	22697	644	2,49	80,30	19,676
13	195	22697	22698	644	2,49	60,93	19,676
14	201	22699	22698	614	2,37	54,82	17,704

Например, на рисунке видно, что вместе с товаром 22386 чаще покупают товар 850998 (правило 169) с поддержкой 3,22% и достоверностью 67,67%. В общем случае, чем выше поддержка и достоверность АП, тем выше его значимость.

На следующем рисунке правило 531 следует понимать как «Если куплен товар 23200, то с ним купят товары 23199 и 23203». Правило 409 следует понимать как «Если куплены товары 20727 и 22383, то с ними купят товар 20726».



№	Номер правила	Условие	Следствие	Поддержка Кол-во	Поддержка %	Достоверность	Лифт
474	239	23172	23170	272	1,05	86,62	49,093
475	238	23170	23172	272	1,05	59,52	49,093
476	409	20727 22383	20726	271	1,05	46,17	11,531
477	408	20726 22383	20727	271	1,05	59,82	11,965
478	407	20726 20727	22383	271	1,05	60,36	11,970
479	79	20972	20971	271	1,05	43,85	21,592
480	78	20971	20972	271	1,05	51,52	21,592
481	531	23200	23199 23203	270	1,04	44,85	27,078
482	530	23200 23203	23199	270	1,04	72,78	18,868

Задание для самостоятельного выполнения:

1. Выполнить анализ продаж по набору данных Online Retail II
2. Выполнить анализ продаж по набору данных
   http://archive.ics.uci.edu/dataset/292/wholesale+customers
   - Выполнить преобразование массива к нужному формату;
   - Загрузить массив в Deductor;
   - Определить значимые переменные;
   - Выполнить поиск ассоциативных правил;
   - Определить 5 наиболее значимых правил и представить их в текстовом виде.

Литература:

Орешков, В. И. Интеллектуальный анализ данных : учебное пособие / В. И. Орешков. — Рязань : РГРТУ, 2017. — 160 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/168028

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по организации самостоятельной работы

по дисциплине «Машинное обучение и анализ данных для решения задач электроэнергетики»

для студентов направления подготовки /специальности

13.04.02 «Электроэнергетика и электротехника»,

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

1. СТРУКТУРА САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Общий объём самостоятельной работы составляет 54 астр. час.

Самостоятельная работа распределена следующим образом:

Вид деятельности студентов	Итоговый продукт самостоятельной работы	Средства и технологии оценки	Объем часов (астр)
Подготовка к лекции	Конспект	Собеседование	6.00
Подготовка к практическому занятию	Письменный отчет	Собеседование	18.00
Подготовка к лабораторным занятиям	Письменный отчет	Собеседование	18.00
Самостоятельное изучение литературы	Конспект	Собеседование	12
ИТОГО			54.00

2. МЕТОДИКА САМОСТОЯТЕЛЬНОЙ РАБОТЫ

2.1. Методика самостоятельной работы по подготовке к лекции

При подготовке к лекции в первую очередь необходимо прочитать конспект лекции.

Конспект лекции следует рассматривать как источник информации по конкретной дисциплине или группе дисциплин. Любой источник информации содержит лишь некоторый набор сведений, далеко не исчерпывающий существующие точки зрения, статистические данные, аналитические выкладки, касающиеся прямо или косвенно данной тематики. В силу этого обстоятельства конспекты лекций рекомендуется расширять и обогащать, активно используя дополнительную литературу.

Даже самого лучшего конспекта недостаточно, чтобы безупречно подготовиться к тесту, семинару, зачету, экзамену. Конспект лекций – один (но далеко не единственный) из основных источников информации по конкретному курсу, помимо рекомендованных учебников, учебных и учебно-методических пособий, научных работ, аналитических и статистических сборников и прочего. При этом преподаватель в процессе оценки знаний студента обычно ориентируется именно на прочитанные им лекции, поэтому конспекты следует использовать при подготовке к ответу в обязательном порядке.

Во-первых, тему целесообразно учить в соответствии с планом, отмеченным в конспекте. В учебниках различных авторов в соответствии с их подходом к преподавании дисциплины темы могут излагаться в различном порядке.

Во-вторых, рекомендованная преподавателем литература по соответствующей теме, отмеченная в конспекте, будет нужна для более широкого обзора темы и охвата всех вопросов, предложенных преподавателем.

При этом самостоятельно, без консультации преподавателя, дополнительную литературу подобрать достаточно сложно.

В-третьих, в конспекте содержится уже проработанная информация, не требующая детального подхода к изучению. Стиль изложения материала в различных литературных источниках далеко не всегда бывает доступным.

В-четвертых, содержание конспекта – минимум, который студент обязан знать в обязательном порядке в соответствии с учебным планом. При этом в авторских учебниках и пособиях отдельным разделам может уделяться большее внимание, чем остальным, а ваш лектор может иметь на этот счет собственное мнение.

В-пятых, конспект окажет вам большую услугу, если рассматривать его как маленькую энциклопедию важнейших вопросов, которые могут быть вам заданы преподавателем. Большинство вопросов при итоговой оценке знаний будет задано с учетом того, что в лекциях предлагались ответы на них.

2.2. Методика самостоятельной работы по подготовке к практическим занятиям

Практические занятия по дисциплине предполагают изучение определенной компьютерной технологии на примере конкретного программного продукта. В ходе самостоятельной подготовки к практическим занятиям студенты готовят программы расчетов, обрабатывают результаты и составляют отчеты.

Форма отчётности – наличие отчетов по решенным задачам, правильность которых проверяет преподаватель и делает соответствующие пометки в журнале.

Ряд практических занятий проводится в форме дискуссии, на которых проходит обсуждение конкретных компьютерных технологий. Обсуждения направлены на освоение научных основ, эффективных методов и приемов решения конкретных практических задач, на развитие способностей к творческому использованию получаемых знаний и навыков. Основная цель

заключается в закреплении знаний полученных в ходе прослушивания лекционного материала. Семинар проводится в форме устного опроса студентов по вопросам семинарских занятий, а также в виде решения практических задач или моделирования практической ситуации. В ходе подготовки к семинару студенту следует просмотреть материалы лекции, а затем начать изучение учебной литературы. Следует знать, что освещение того или иного вопроса в литературе часто является личным мнением автора, построенного на анализе различных источников, поэтому следует не ограничиваться одним учебником или монографией, а рассмотреть как можно больше материала по интересуемой теме.

В ходе самостоятельной работы студенту необходимо отслеживать научные статьи в специализированных изданиях.

Студенту рекомендуется следующая схема подготовки к семинарскому занятию:

1. Проработать конспект лекций;

2. Прочитать основную и дополнительную литературу, рекомендованную по изучаемому разделу;

3. Ответить на вопросы плана семинарского занятия;

4. Выполнить домашнее задание;

5. Проработать тестовые задания и задачи;

6. При затруднениях сформулировать вопросы к преподавателю.

При подготовке к семинарским занятиям следует руководствоваться указаниями и рекомендациями преподавателя, использовать основную литературу из представленного им списка. Для наиболее глубокого освоения дисциплины рекомендуется изучать литературу, обозначенную как «дополнительная» в представленном списке. При подготовке доклада на семинарское занятие желательно заранее обсудить с преподавателем перечень используемой литературы, за день до семинарского занятия предупредить о

необходимых для предоставления материала технических средствах, напечатанный текст доклада предоставить преподавателю

2.4. Методика самостоятельной работы по самостоятельному изучению литературы

Большая часть самостоятельной работы студента состоит в **изучении литературы**. Одна из задач студента – научиться самостоятельно работать с книгой, а это требует определенных затрат энергии и времени. Поэтому надо научиться делать эту работу рационально, то есть необходимо учиться читать.

Методы эффективной работы с книгой в целях развития интеллекта можно условно разделить на две группы:

1. Правильная организация процесса чтения
2. Повышение скорости чтения и восприятия.

В комплексе оба метода могут в 2-3 раза сократить время прочтения различных материалов.

При чтении текста мозг формирует «свою трактовку содержания» прочитанного. Происходит перекодирование сообщения на языке собственных мыслей читателя. Мозг выделяет «ядерное», сущностное значение из текста. Эффективность такой перекодировки зависит от осмысления и внимательности чтения.

Дифференциальный алгоритм чтения в соответствии с блоками позволяет реализовать логико-семантический анализ текста: вначале выделить ключевые слова, затем построить смысловые ряды и, наконец, выделив цепь знаний, сформулировать доминанту. Именно так и только так (по О.А. Андрееву) можно увидеть главное, действительно, проникнуть в суть вещей, явлений, излагаемых автором.

Возможны три основных способа чтения.

- Первый способ – артикуляция или проговаривание вслух (или почти вслух) того, что читаешь. Скорость такого чтения невелика.
- Второй способ – чтение про себя, при котором речевой процесс проявлен в форме внутренней речи, то есть без открытой артикуляции. Текст, при этом усваивается более эффективно. Способ в принципе допускает быстрое чтение.
- Третий, наиболее совершенный способ чтения – тоже молча, но в условиях максимального сжатия внутренней речи, при котором она проявляется в виде коротких залпов ключевых слов и смысловых рядов, адекватно отражающих смысл текста.

Итак, артикуляция замедляет процесс чтения и от нее необходимо избавиться. Однако не приведет ли сокращение артикуляции при повышении скорости чтения к снижению качества восприятия и осмысления полученной информации?

Как показали исследования психологов, иногда при чтении слова могут быть заменены наглядными представлениями, пространственными схемами. Целые группы слов – одним словом.

Быстро читающие люди обладают способностью, не проговаривания читаемый текст, сразу улавливать и фиксировать замысел автора, а затем усваивать его на уровне внутренней речи. В этом случае, несмотря на высокую скорость чтения, происходит глубокое понимание и усвоение прочитанного, так как основная идея понятна с самого начала. Задачу научиться такому чтению можно решать в два этапа. Первый предполагает сокращение артикуляции, если она ярко выражена, второй – овладение приемами чтения, при которых текст воспринимается крупными информационными блоками.

Перед углубленным чтением любого текста (статьи, книги, конспекта, лекции перед экзаменом) сначала бегло просмотрите его целиком. При этом постарайтесь выявить основные стержневые идеи, наиболее крупные части и

логику их изложения. Лишь после такого просмотра переходите к более детальному чтению.

Перед чтением статьи или параграфа учебника попробуйте проделать следующее: прочитайте внимательно первый абзац, потом бегло просмотрите первые или последние фразы следующих абзацев (в них обычно содержится основная информация), обратите внимание на курсивы, разрядки, подзаголовочный текста и, наконец, внимательно прочтите один-два последних абзаца; постарайтесь выявить основное направление текста и его построение.

Прочитав в тексте интересную идею, полезно остановить свое внимание на ней, прислушаться к тем мыслям, которые она у вас вызвала, подумать о тех последствиях, которые из нее вытекают, попытаться развивать ее дальше.

Существенно замедляют чтение регрессии – частые возвратные движения глаз, многократное повторное прочитывание материала. Возвратиться к уже прочитанному, но недостаточно хорошо понятому участку лучше всего, когда прочитан законченный смысловой фрагмент текста и сделана хотя бы попытка его осмысления, а не в процессе чтения предложения.

Любой текст не однороден по своей информационной насыщенности. В некоторых предложениях, абзацах сконцентрировано очень много информации, например, формулируются основные положения, ведущие идеи и т.д., а другие служат лишь иллюстрацией, фоном. Таким образом, текст имеет «смысловой рельеф». Чем точнее читатель умеет определить степень важности каждого отрезка теста и приспособить к «смысловому барьеру» способ своего чтения (то есть замедлить и углубить в более важных местах и ускорять в менее важных), тем продуктивнее чтение. Постарайтесь гибко варьировать способ работы с текстом в соответствии с его «смысловым барьером».

Чтобы чтение было эффективным, попробуйте по прочитанному всегда отвечать на 6 вопросов: «*Кто делает? Что делает? Когда? Почему? Где? Как?»*

Самостоятельная работа с книгой может быть успешной, если текст не только прочитан, но и законспектирован. Существует несколько **форм записей**, но любая форма записи не даст нужного результата, если не будет пробуждать мысли того, кто ее ведет, если отсутствует активная работа ума и формирование своих выводов из прочитанного.

Выбор формы записи зависит от индивидуальных особенностей человека, его образованности и опыта. При этом не меньшую роль играет назначение записей, то есть то, какие задачи ставит перед собой человек (для самообразования, для выступления на семинаре, для использования в будущем).

Введение записей мобилизует наряду со зрительной памятью, также и моторную память. Кроме того, у человека, систематически ведущего записи изучаемой литературы, создается свой фонд материалов для быстрого повторения и мобилизации накопленных знаний.

Все записи должны быть убористыми и компактными. Интервалы между строками должны быть достаточными, чтобы вписывать дополнения. Рекомендуется вести записи ручкой, а карандашом или ручкой другого цвета пользоваться для отметок и выделений при последующей работе. Полезно также датировать записи.

Записи могут носить различный характер: план, выписки, тезисы, аннотирование, конспектирование, реферирование.

3 УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Для успешного освоения дисциплины, необходимо выполнить следующие виды самостоятельной работы, используя рекомендуемые источники информации

№ п/п	Виды самостоятельной работы	Рекомендуемые источники информации (№ источника)		
		Основная	Дополнительная	Интернет-ресурсы
1	Выполнение исследовательского проекта по заданной проблематике	2 3		1 2
2	Подготовка к лекции	1 2 3	1 2	1 2
3	Подготовка к практическому занятию		1 2	1 2
4	Подготовка реферата, доклада	1 2 3	1 2	1 2
5	Самостоятельное изучение литературы	1 2 3	1 2	1 2

Основная литература

- Персова, М. Г. Современные компьютерные технологии : Конспект лекций / Персова М. Г. - Новосибирск : Новосибирский государственный технический университет, 2014. - 80 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-7782-2427-8
- Современные компьютерные технологии / Р.Г. Хисматов / Р.Г. Сафин / Д.В. Тунцев / Н.Ф. Тимербаев : учебное пособие Электронный ресурс : Казанский национальный исследовательский технологический университет ; Казань, 2014. - 83 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-7882-1559-4

- Лыкин, А. В. Математическое моделирование электрических систем и их элементов : Учебное пособие / Лыкин А. В. – Новосибирск : Новосибирский государственный технический университет, 2013. – 227 с. – Книга находится в базовой версии ЭБС IPRbooks. – ISBN 978-5-7782-2262-5

Дополнительная литература

- Введение в математическое моделирование : учеб. пособие / В.Н. Ашихмин, М.Б. Гитман, И.Э. Келлер и др.] ; [под ред. П. В. Трусова]. - Москва : Логос : [Университетская книга], 2007. - 439 с. : ил. - (Новая университетская библиотека). - Гриф: Доп. МО. - Библиогр.: с. 431-435 (130 назв.). - Предм. указ.: 436-437. - ISBN 978-5-98704-037-X
- Карпова, И. М. Компьютерные технологии в науке и производстве. Расчет физических полей в электроэнергетике Электронный ресурс : Учебное пособие / И. М. Карпова, В. В. Титков. – Компьютерные технологии в науке и производстве. Расчет физических полей в электроэнергетике,2021-03-25. – Санкт-Петербург : Санкт-Петербургский политехнический университет Петра Великого, 2010. – 212 с. – Книга находится в премиум-версии ЭБС IPR BOOKS. – ISBN 978-5-7422-3026-7
- Титков, В. В. Компьютерные технологии. Comsol Multiphysics в задачах энергетики Электронный ресурс : Учебное пособие / В. В. Титков, Э. И. Янчус. – Компьютерные технологии. Comsol Multiphysics в задачах энергетики,2021-03-25. – Санкт-Петербург : Санкт-Петербургский политехнический университет Петра Великого, 2012. – 184 с. – Книга находится в премиум-версии ЭБС IPR BOOKS. – ISBN 978-5-7422-3684-9

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1 http://catalog.ncfu.ru – электронные каталоги Ассоциации электронных библиотек учебных заведений и организаций СКФО;

2 http://window.edu.ru – Информационная система "Единое окно доступа к образовательным ресурсам".