

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего
образования
«Северо-Кавказский федеральный университет»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению практических работ
по дисциплине «**Системы искусственного интеллекта**»
для студентов направления
13.03.02 «Электроэнергетика и электротехника»
Направленность (профиль): «Цифровые технологии в
электроэнергетике»

Ставрополь 2026

Содержание

Практическая работа 1. Разведочный анализ данных	3
Практическая работа 2. Классификация с помощью деревьев решений	34
Практическая работа 3. Поиск ассоциативных правил	44
Практическая работа 4. Классификация с помощью персептрона	54
Практическая работа 5. Прогнозирование временных рядов с помощью НС	74
Практическая работа 6. Классификация с помощью сетей Кохонена	83
Практическая работа 7. Генетические алгоритмы	96
Практическая работа 8. Построение функции принадлежности нечеткого множества	102
Практическая работа 9. Логическое программирование. Логика предикатов	111

Практическая работа 1. Разведочный анализ данных

Теоретические сведения

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); **8** 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 gradient analysis), 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;

$\bar{x}$ – 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: $-13.333+6.667+6.667 \approx 0$, 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

$\bar{x}$ – 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;

$\bar{x}$ – 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

$$medianAD = \text{медиана}(|x_1 - m|, |x_2 - m|, \dots, |x_n - m|),$$

где m – медиана;

$x_i - i$ -е значение признака

n – число значений признака.

Для рассматриваемого набора данных $medianAD$ вычисляется как $medianAD = \text{медиана}(0, 0, 20) = 0$

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百分иль – такое число z_p , что значения P -й части набора меньше или равны z_p . Например, 25-й百分иль – такое число z_p , что 25% значений признака меньше или равны z_p .

Медиана соответствует 50-й процентилю.

百分иль аналогичен квантилю, при этом квантили индексируются долями (так, например, квантиль 0,25 – это то же самое, что и 25-й百分иль). 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

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 процентиям, 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)

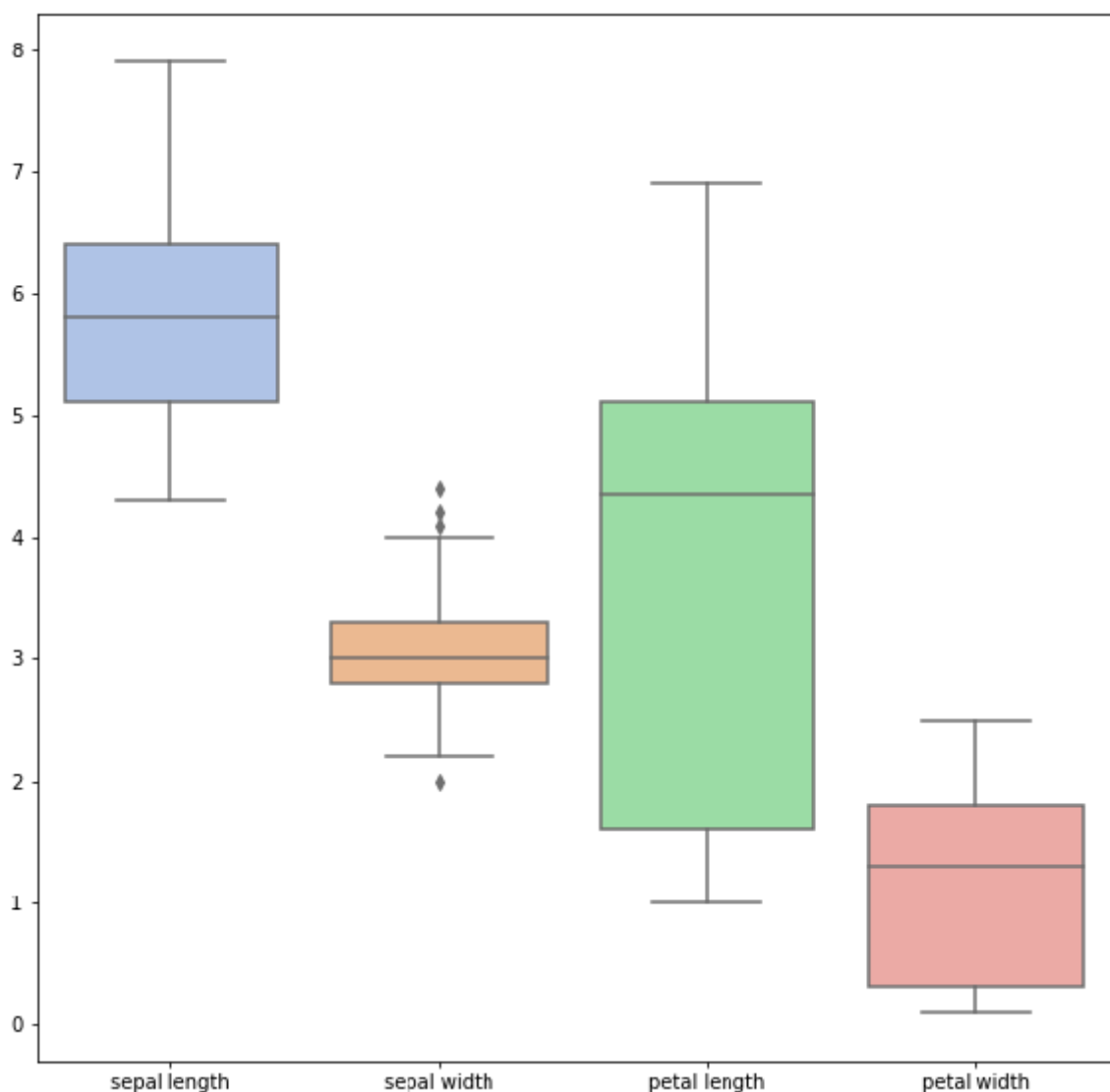


Рисунок 1. Визуализация распределения данных на примере набора «Ирисы Фишера» (The Iris Dataset) с применением коробчатых диаграмм

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.

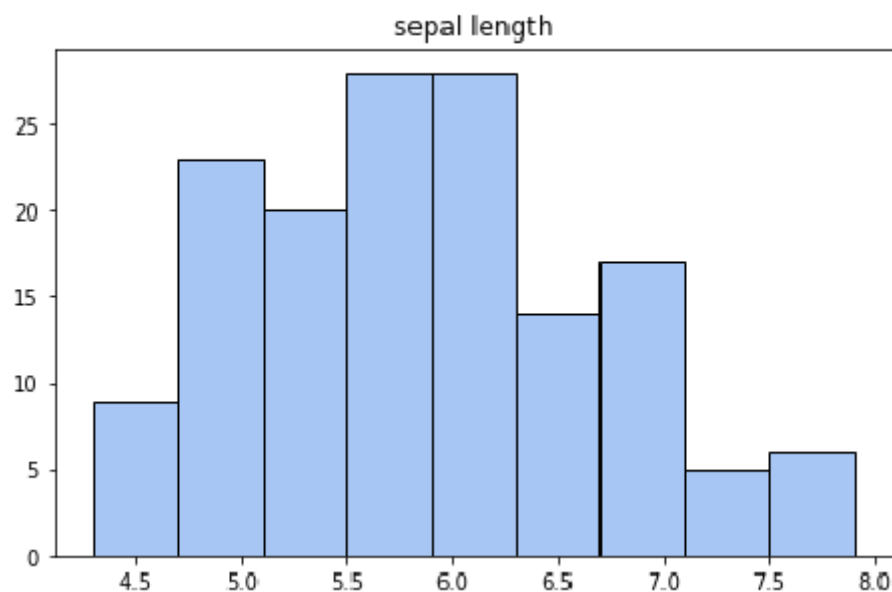


Рисунок 2. Гистограмма распределения для признака sepal length набора данных «Ирисы Фишера» (The Iris Dataset)

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 признака 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 – значения признаков X и Y соответственно

n – число значений признака;

$\bar{x}, \bar{y}$ – 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.

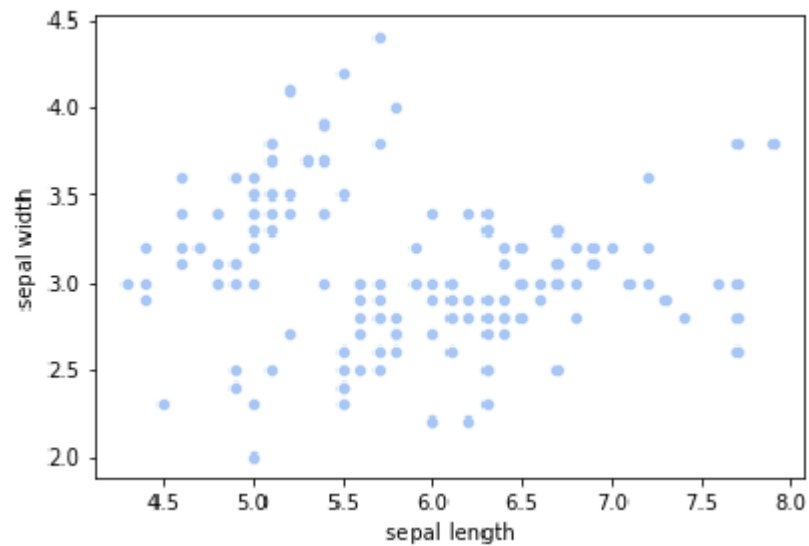


Рисунок 3. Диаграмма рассеяния для пары признаков sepal length и sepal width набора данных «Ирисы Фишера» (The Iris Dataset)

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.

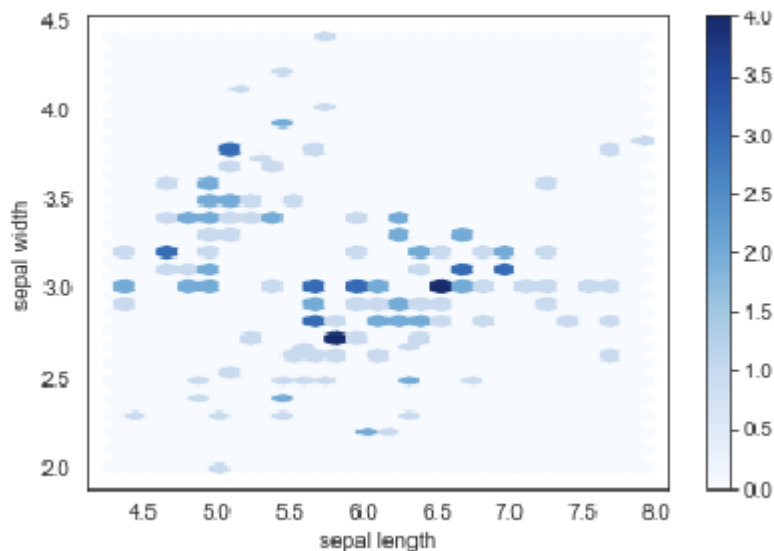


Рисунок 4. Визуализация связи для пары признаков `sepal length` и `sepal width` набора данных «Ирисы Фишера» (The Iris Dataset) с использованием графика с шестиугольной сеткой

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 $avg = \text{sum}(a * \text{weights}) / \text{sum}(\text{weights})$, при этом $\text{sum}(\text{weights})$ не должна быть равна нулю;

returned (тип: `bool`) – параметр, который принимает 2 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 принимает 2 значения: `True` или `False`; если параметр `keepdims` имеет значение `True`, то размерность измерений, по которым 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` или `double`) – 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 $n - \text{ddof}$, 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-следовательность 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-перцентиль вдоль выбранного измерения массива.

Входные параметры:

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

q (тип: array_like типа float) – перцентиль или последовательность перцентилей, 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 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;

Method (тип: str) – метод, используемый для оценки перцентиль: «*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 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;

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 процентилем, а другие измерения – это измерения, которые остались после выполнения действий над массивом *data*; если входные данные содержат целые числа или числа с плавающей запятой меньше, 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 типа float) – 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_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 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;

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

Выходной параметр:

ptr (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 2 значения: 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 2 значения: 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`) – массив ребер 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)
```

Функция 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 $x/\text{sum}(x)$. 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 значения 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 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».

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

Контрольные вопросы

1. Для чего выполняется разведочный анализ данных?
2. Какие инструменты применяются в разведочном анализе?
3. Для чего используется корреляция признаков?
4. Что показывает диаграмма рассеяния?

Практическая работа 2. Классификация с помощью деревьев решений

Теоретические сведения

Наряду с нейронными сетями деревья решений (ДР) являются популярной технологией ИАД, широко используемой для решения задач численного предсказания и классификации.

В основе работы ДР лежит процесс рекурсивного разбиения исходного множества наблюдений или объектов на подмножества, ассоциированные с предварительно заданными классами. Разбиение производится с помощью решающих правил, в которых осуществляется проверка значений атрибутов по заданному условию. Рекурсивными называются алгоритмы, которые на каждом последующем шаге используют результаты, полученные на предыдущем.

Обучающая выборка для ДР формируется по тем же принципам, что и для НС. Но поскольку ДР ориентированы больше на задачу классификации, когда целевая переменная является категориальной, то и в обучающей выборке выходная переменная представляет собой метку класса. ДР состоят из объектов 2-х типов – узлов и листьев. В узлах располагаются правила, а в листьях – результат, т.е. наблюдения исходного набора данных, имеющих одинаковую переменную класса. Таким образом, каждый лист в дереве связан с определенным классом.

Также, как и нейронные сети, деревья решений реализуются специализированными программными средствами или входят в состав аналитических пакетов. К последним относятся IBM Data Modeler, SAS Enterprise Miner, Rapid Miner, Deductor и др. Специализированными инструментами, разработанными специально для моделирования на основе ДР, являются SPSS Answer Tree, See5, C4.5 и др.

Пример выполнения задания

Рассмотрим основные этапы построения ДР на примере обучающего набора данных, отражающего зависимость наличия или типа отказа электрооборудования от пусковых условий. Для построения используем пакет Deductor Studio Academic.

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 двигателя 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 $f = 10$ 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.
- метка «сбой машины», которая указывает, является ли машина неисправной в этой конкретной точке данных для любого из следующих режимов отказа.
- цель: отказ или нет
- тип отказа: тип отказа

На платформе Deductor для импорта данных необходимо выполнить следующие шаги:

- 1) Загрузка файла, определение правил формирования столбцов данных;
- 2) Определение параметров импорта (рисунок 1);

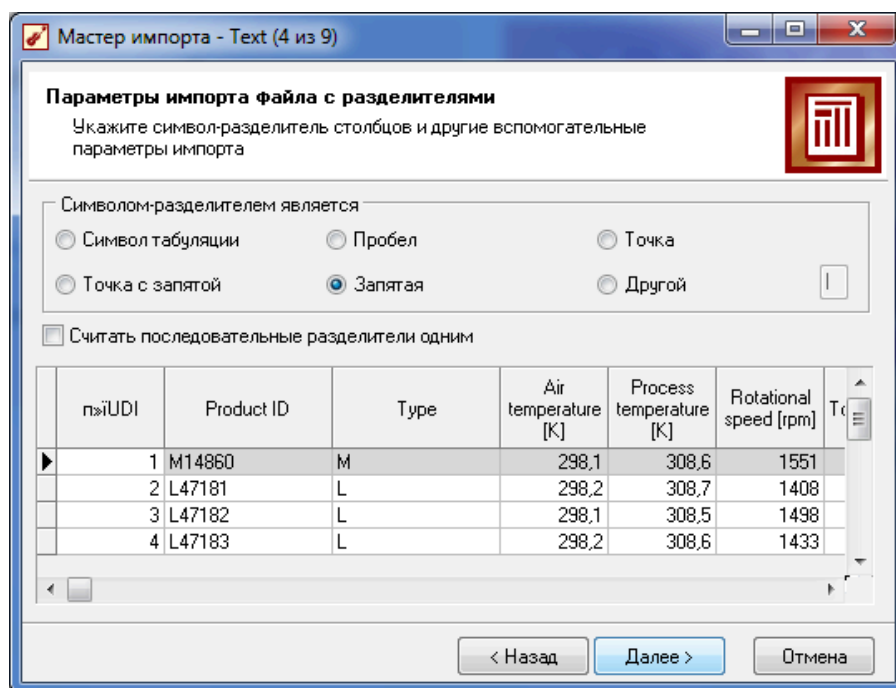


Рисунок 1 – Определение параметров импорта

- 3) Определение типов данных для столбцов, а также определение их назначения в модели (рисунок 2). Для решения поставленной задачи столбцы «Target» и «Failure Type» («Отказ» и «Тип отказа») имеют назначение «Выходное», остальные – «Входное».

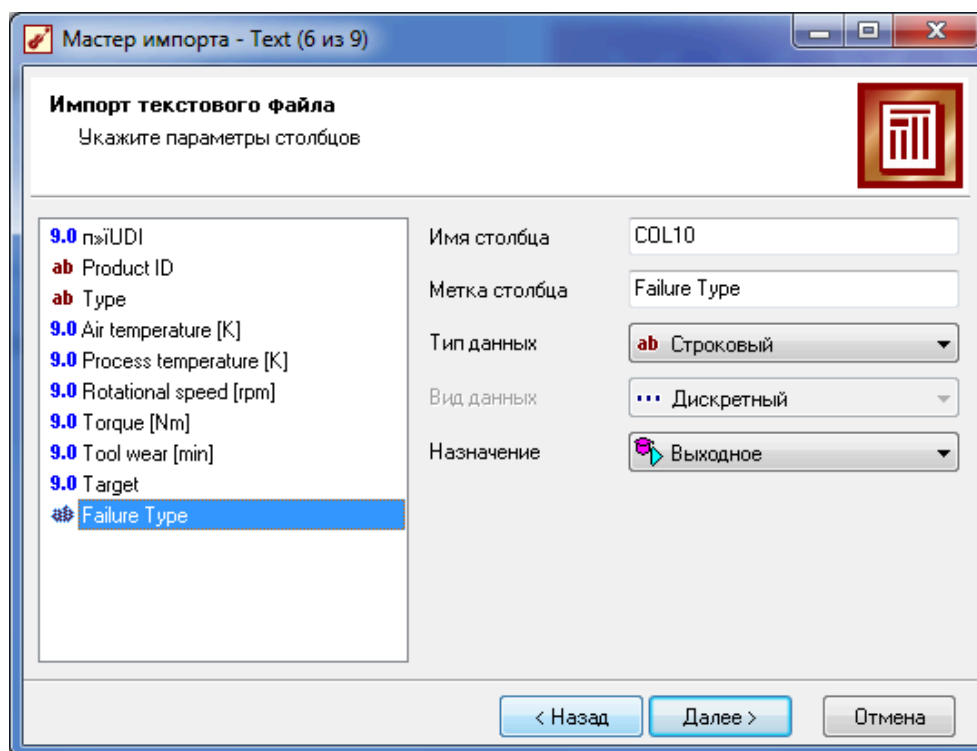


Рисунок 2 – Определение типов данных

Результат импорта данных в пакет Deductor показан на рисунке 3

pшUD1	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
1	M14860	M	298.1	398.6	1993	42.8	0	0	No Failure
2	L47181	L	298.2	398.7	1408	46.3	3	0	No Failure
3	L47182	L	298.1	398.5	1488	49.4	5	0	No Failure
4	L47183	L	298.2	398.6	1433	39.5	7	0	No Failure
5	L47184	L	298.2	398.7	1408	40	9	0	No Failure
6	M14865	M	298.1	398.6	1425	41.9	11	0	No Failure
7	L47186	L	298.1	398.6	1598	42.4	14	0	No Failure
8	L47187	L	298.1	398.6	1527	40.2	16	0	No Failure
9	M14868	M	298.3	398.7	1607	28.6	19	0	No Failure
10	M14869	M	298.5	399	1741	28	21	0	No Failure
11	H29424	H	298.4	398.9	1782	23.9	24	0	No Failure
12	H29425	H	298.6	399.1	1423	44.3	29	0	No Failure
13	M14872	M	298.6	399.1	1329	51.1	34	0	No Failure
14	M14873	M	298.6	399.2	1742	30	37	0	No Failure
15	L47194	L	298.6	399.2	2005	19.6	40	0	No Failure
16	L47195	L	298.6	399.2	1542	48.4	42	0	No Failure
17	M14876	M	298.6	399.2	1311	46.6	44	0	No Failure
18	M14877	M	298.7	399.2	1410	65.6	47	0	No Failure
19	H29432	H	298.8	399.2	1306	54.5	50	0	No Failure
20	M14879	M	298.9	399.3	1632	32.5	55	0	No Failure
21	H29434	H	298.9	399.3	1375	42.7	56	0	No Failure
22	L47201	L	298.8	399.3	1490	44.8	63	0	No Failure
23	M14882	M	298.9	399.3	1581	30.7	65	0	No Failure
24	L47203	L	299	399.4	1798	25.7	68	0	No Failure
25	M14884	M	299	399.4	1581	37.3	70	0	No Failure
26	L47205	L	299	399.5	1861	23.3	73	0	No Failure
27	L47206	L	299.1	399.5	1932	39	75	0	No Failure
28	H29441	H	299.1	399.4	1811	24.6	77	0	No Failure
29	L47208	L	299.1	399.4	1439	44.2	82	0	No Failure
30	L47209	L	299	399.4	1693	30.1	84	0	No Failure
31	M14890	M	299.1	399.5	1339	48.2	86	0	No Failure
32	L47211	L	299	399.4	1798	25.5	89	0	No Failure
33	L47212	L	299	399.4	1419	48.3	91	0	No Failure
34	L47213	L	298.9	399.3	1685	32.5	93	0	No Failure
35	M14894	M	299.0	399.1	1959	34.7	95	0	No Failure
36	M14895	M	298.8	399.2	1492	48.6	98	0	No Failure

Рисунок 3 – Результат импорта данных

Чтобы приступить к работе с ДР, нужно открыть Мастер обработки и в списке доступных обработчиков в разделе Data Mining выбрать пункт «Дерево решений».

На 2-м шаге нужно выбрать назначение столбцов, в качестве выходного поля следует выбрать не «Отказ», или «Тип отказа».

На 3-м шаге производится разбиение исходного набора данных на обучающее и тестовое множество.

На 4-м шаге требуется настроить параметры упрощения ДР – ранней остановки или отсечения ветвей. В параметрах ранней остановки требуется указать минимальное количество примеров в узле, при котором формирование узлов-потомков будет запрещено. В параметрах отсечения ветвей требуется указать достоверность (уровень доверия), которая должна быть достигнута путём отсечения ветвей. Чтобы каждый алгоритм был задействован, должен быть установлен соответствующий флажок.

На 5-м шаге задаётся режим обучения ДР – автоматический или интерактивный. При автоматическом построении алгоритм сам выбирает атрибут ветвления на каждом шаге. В интерактивном режиме аналитик имеет возможность выбирать, какой атрибут следует использовать для разбиения. Интерактивный режим особенно полезен, если на некотором уровне разбиения встречаются два атрибута, дающих одинаковый прирост информации, т.е. равнозначные с точки зрения алгоритма. Но для аналитика они могут представлять различный интерес, поэтому он может принять решение о включении в дерево наиболее интересующего его атрибута. Кроме этого, с использованием интерактивного режима аналитик может строить альтернативные деревья.

Если был выбран автоматический режим, то на 5-м шаге откроется окно управления процессом обучения (рис. 4).

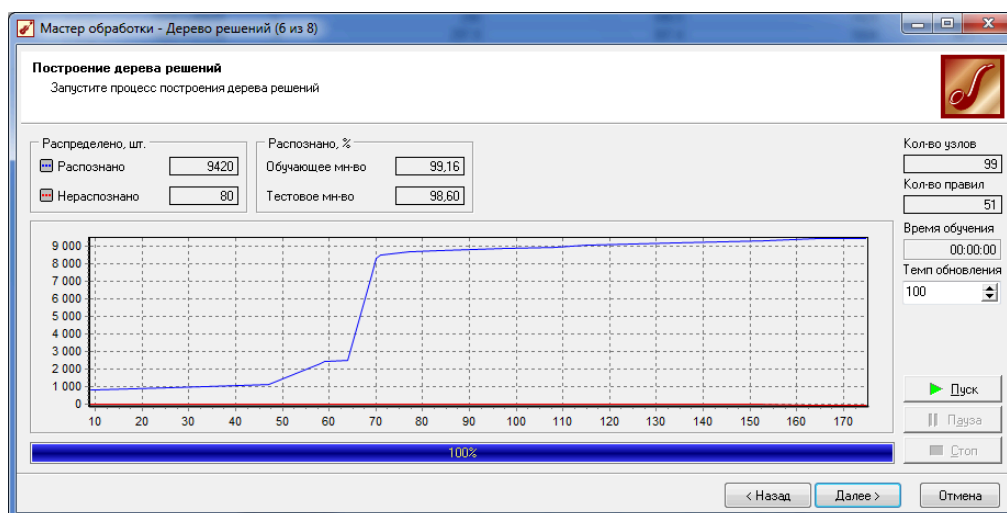


Рисунок 4 – Процесс обучения

Окна управления процессом обучения ДР содержит текущие параметры процесса обучения в числовом и графическом виде, кнопки для запуска и остановки обучения.

Для ДР аналогом ошибки является число распознанных (метка класса, присвоенная ДР соответствует фактическому классу) и не распознанных (метка класса, присвоенного деревом, не соответствует фактической) примеров на ОМ и ТМ. Также отображается текущее число узлов и правил в дереве.

Процесс обучения ДР можно разделить на два этапа. На первом формируются наиболее значимые правила, которые строят узлы и листья с большим числом примеров. Обычно число таких правил невелико (не более 10 – 12), но они распознают до 90 % примеров. На графике первый этап отображается интенсивным возрастанием числа распознанных примеров с возрастанием числа узлов и правил.

На втором этапе алгоритм обучения пытается сформировать правила для нетипичных примеров, которые, скорее всего, содержат аномальные значения атрибутов. На этом этапе алгоритм начинает создавать большое количество узлов, в каждый из которых попадает всего несколько примеров, при этом число распознанных примеров практически не увеличивается. Обычно второй этап обучения не приводит к улучшению классифицирующей

способность дерева, и процесс обучения следует завершить с помощью кнопки «Стоп».

На 6-м шаге пользователю предлагается возможность выбрать способы представления результатов обучения. Наиболее информативными визуализаторами являются «Дерево решений», «Правила», «Значимость атрибутов», «Что-если» и «Обучающий набор».

Визуализатор «Дерево решений» (рис. 5) отображает ДР в виде иерархической структуры, в которой узлы представлены в виде прямоугольников, цветовая заливка которых отражает соотношение классов в узле. Если узел окрашен одним цветом, он является «чистым», т.е. содержит примеры единственного класса и объявляется листом.

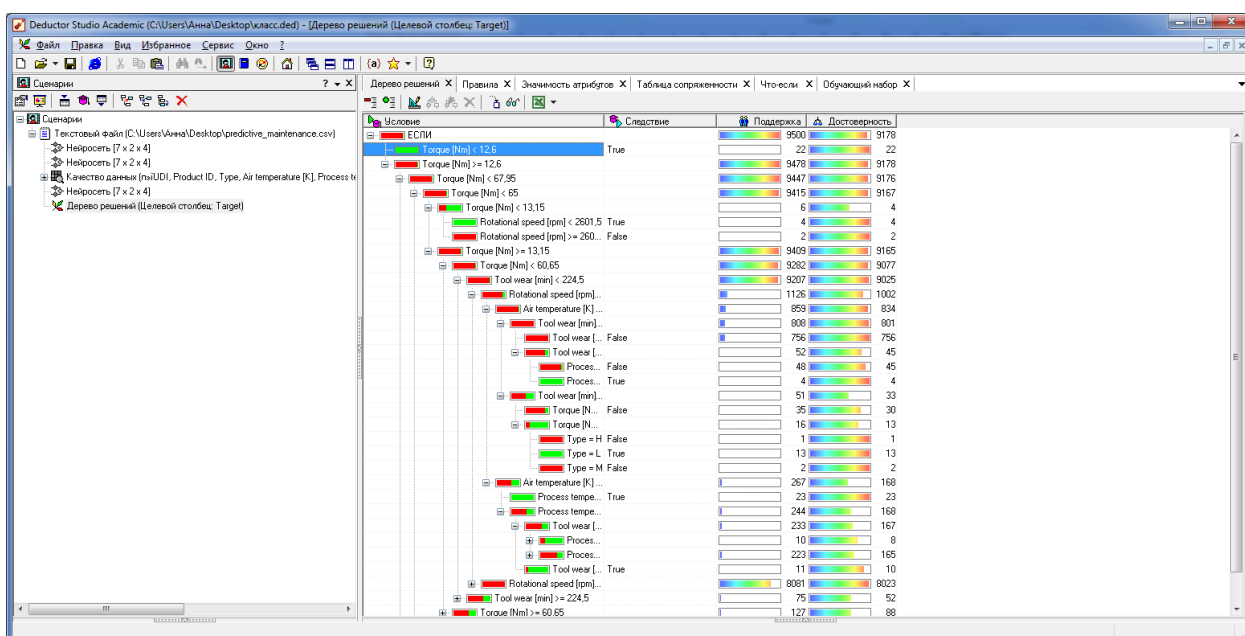
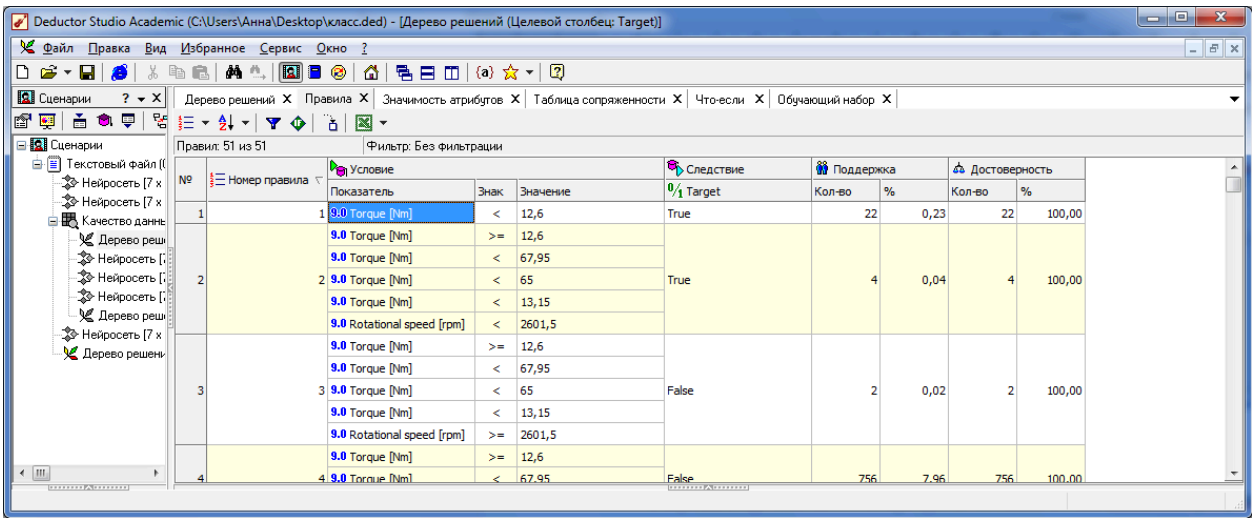


Рисунок 5 – Дерево решений

Справа от каждого узла указывается правило, с помощью которого он был сформирован, а также метка класса (следствие) и значения поддержки и достоверности.

Визуализатор «Правила» отображает решающие правила, извлеченные из ДР (рис. 6). В столбце «Условие» указываются атрибуты (показатели), знаки отношений и числовые значения правила, столбец «Следствие» содержит метку класса. Например, легко увидеть, что правило No 1 «Если

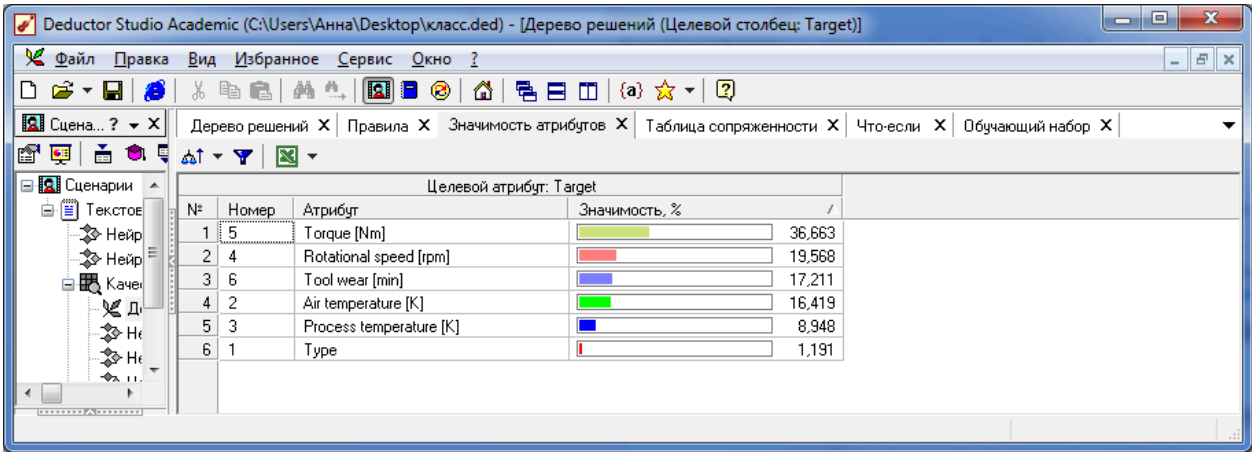
крутящий момент меньше 12,6, то класс «Отказ=Истина» позволяет правильно классифицировать все 22 примера данного класса (100 %). Это правило является наиболее значимым.



№	Номер правила	Условие	Знак	Значение	Следствие	Поддержка	Достоверность
		Показатель			% Target	Кол-во	%
1	1	9.0 Torque [Nm]	<	12,6	True	22	0,23
2	2	9.0 Torque [Nm]	>=	12,6	True	4	0,04
		9.0 Torque [Nm]	<	67,95			
		9.0 Torque [Nm]	<	65			
		9.0 Torque [Nm]	<	13,15			
		9.0 Rotational speed [rpm]	<	2601,5			
3	3	9.0 Torque [Nm]	>=	12,6	False	2	0,02
		9.0 Torque [Nm]	<	67,95			
		9.0 Torque [Nm]	<	65			
		9.0 Torque [Nm]	<	13,15			
		9.0 Rotational speed [rpm]	>=	2601,5			
4	4	9.0 Torque [Nm]	>=	12,6	False	756	7,96
		9.0 Torque [Nm]	<	67,95			

Рисунок 6 - Правила

Визуализатор «Значимость атрибутов» показывает для каждого атрибута процент примеров, распознанных с его помощью (рис. 7).



№	Номер	Атрибут	Значимость, %
1	5	Torque [Nm]	36,663
2	4	Rotational speed [rpm]	19,568
3	6	Tool wear [min]	17,211
4	2	Air temperature [K]	16,419
5	3	Process temperature [K]	8,948
6	1	Type	1,191

Рисунок 7 - Значимость атрибутов

Например, атрибутом с самой высокой значимостью является крутящий момент, с помощью которой было распознано 37 % примеров. Это означает, что данный фактор является наиболее важным с точки зрения отказов.

Визуализатор «Таблица сопряженности» компактно представляет число распознанных и не распознанных примеров (рис. 8).

Фактически	Классифицировано		Итого
	False	True	
False	9655	6	9661
True	81	258	339
Итого	9736	264	10000

Рисунок 8 – Таблица сопряженности

Из таблицы видно, что дерево при обучении допустило достаточно мало ошибок.

Визуализатор «Обучающий набор» (рис. 9) включает столбцы входных и выходных атрибутов исходного набора данных, а также столбец выходов OUT.

Failure Type	Target_OUT	Target Номер правила	Target Поддержка, %	Target Достоверность, %
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	25	0,115789473684211	90,9090909090909
Heat Dissipation Failure	✓	44	0,178947368421053	100
Heat Dissipation Failure	✓	25	0,115789473684211	90,9090909090909
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Power Failure	✓	1	0,231578947368421	100
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963
Heat Dissipation Failure	✓	21	0,284210526315789	96,2962962962963

Рисунок 9 - Визуализатор «Обучающий набор»

Данный тип представления результатов позволяет визуально определить, на каких именно примерах ДР допустило ошибки, и попытаться сформулировать гипотезы о причинах появления ошибки.

Задание для самостоятельного выполнения

1. Постройте дерево решений, определяющее наличие отказа при заданном наборе стартовых условий.
2. Постройте дерево решений, определяющее тип отказа при заданном наборе стартовых условий.

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

Контрольные вопросы

1. Как строится дерево решений?
2. Как обучается дерево решений?
3. Что является признаком окончания разбиения при построении дерева решения?

Практическая работа 3. 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 англ. 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 А и В, происходящими одновременно, записывается в виде:

$$A \rightarrow B,$$

что читается «Если А, то В», где событие А называют условием (antecedent), а событие В - 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 D_i 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, отнесенное к общему числу транзакций.

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

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

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

Достоверностью АП $A \rightarrow B$ является:

$$C(A \rightarrow B) = P(A | B) = \frac{P(A \cap B)}{P(A)}$$

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 интересоваться, 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 наборов (frequent itemset). В основе данного метода лежит алгоритм Apriori, который стал родоначальником целого семейства методов и алгоритмов, использующих концепцию частых предметных наборов (AprioriGen, FP-Growth, 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 А встречался в 70 транзакциях из 100, а товар В — 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 $L = 1$ 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 «с», 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.

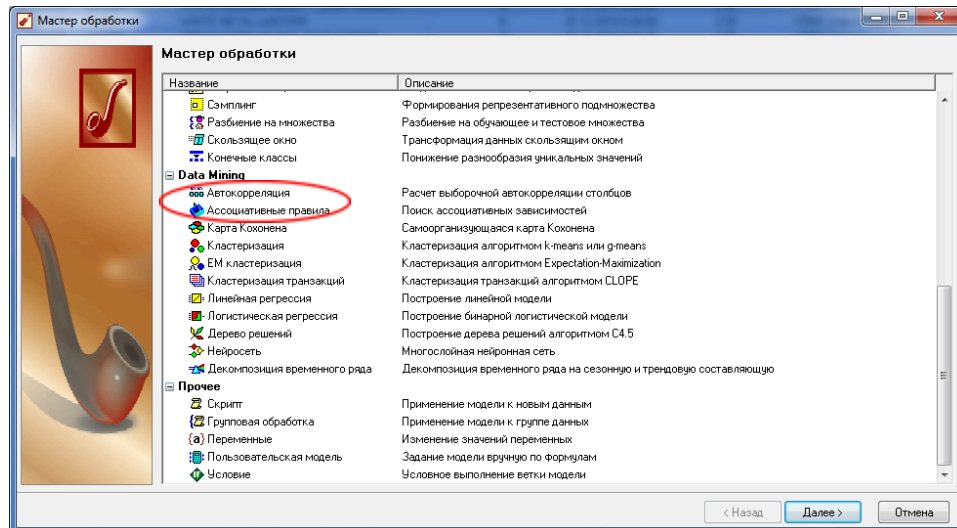
UnitPrice: цена за единицу. Числовой. Цена продукта за единицу в фунтах стерлингов.

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

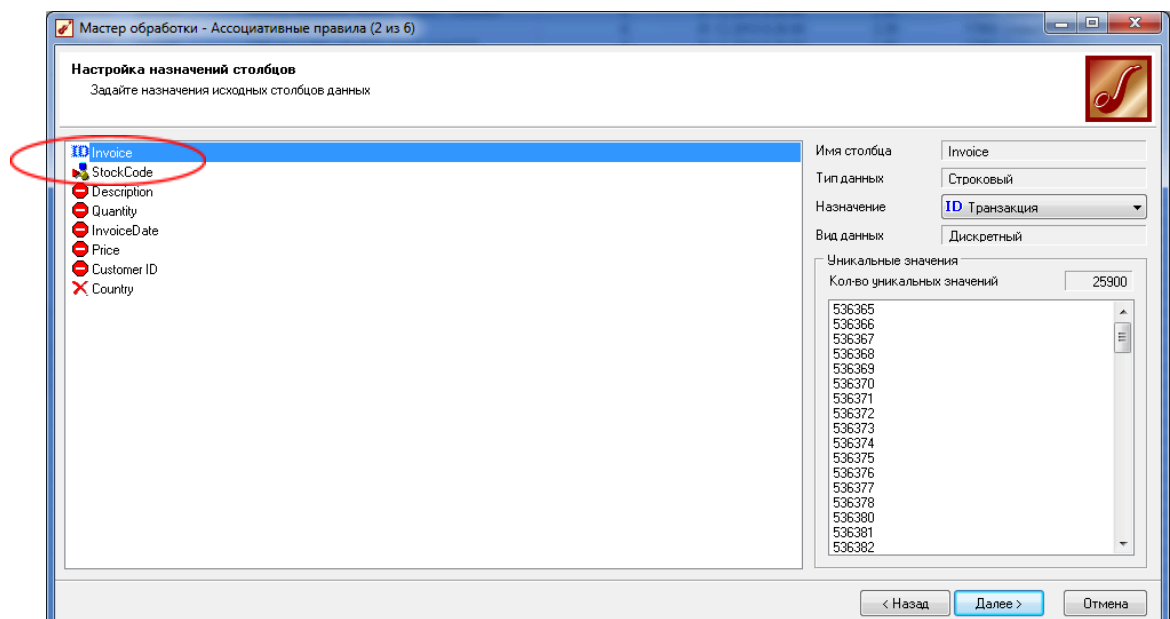
Страна: Название страны. Номинал. Название страны, в которой проживает клиент.

Для поиска ассоциативных правил воспользуемся платформой Deductor.

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

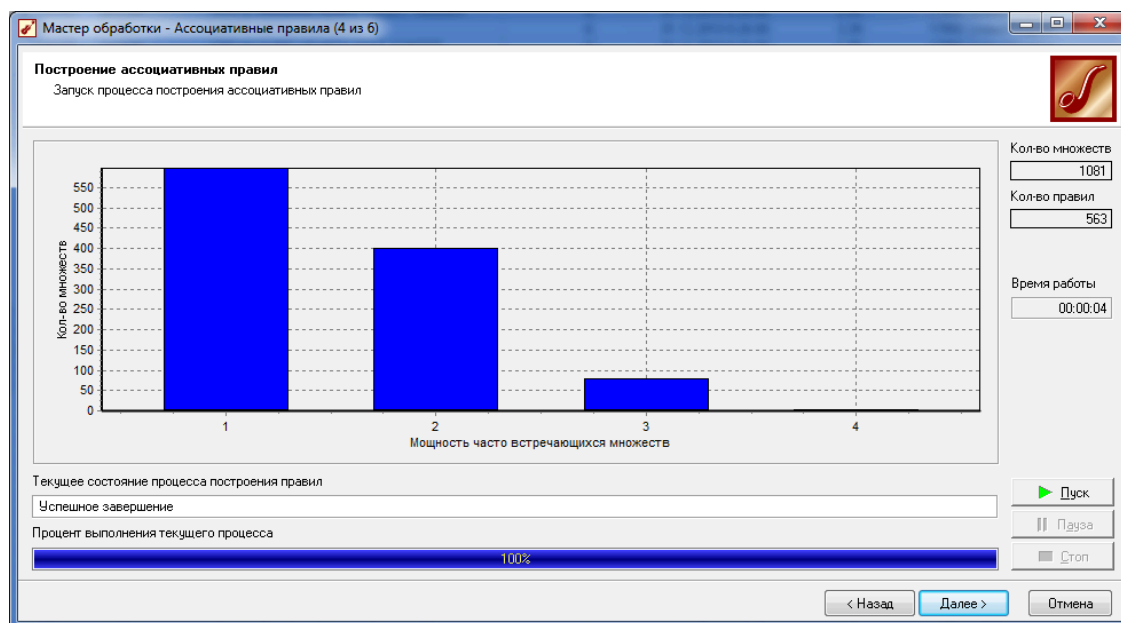


Набор обучающих данных для построения ассоциативной модели имеет специфический формат и должен содержать всего два поля: идентификатор транзакции и предмет. Для анализа ассоциаций в данном наборе данных подходящими являются данные в полях Invoice и StockCode. Поэтому на втором шаге анализа выбираем именно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



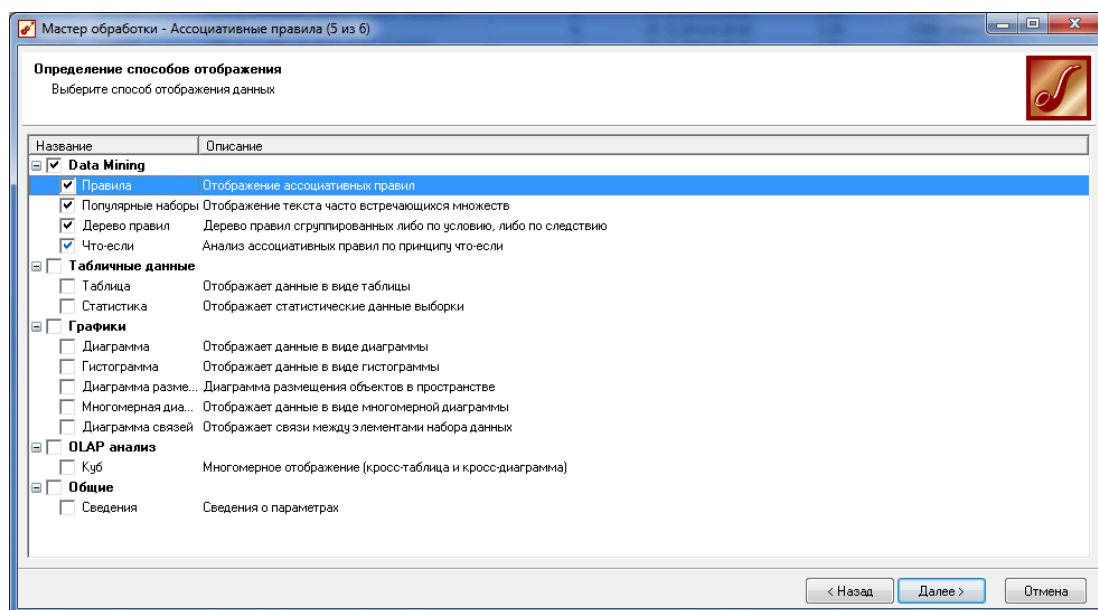
На 3-м шаге нужно выбрать верхнюю и нижнюю границы поддержки и достоверности для нахождения частых предметный наборов, а также для обнаружения на их основе АП.

«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.

Deductor Studio Academic (C:\Users\Анна\Desktop\2.ded) - [Ассоциативные правила [TID="Invoice"; AID="StockCode"]]

Правил: 563 из 563 Фильтр: Без фильтрации

№	Номер правила	Условие	Следствие	Поддержка Кол-во	%	Достоверность	Лифт
1	169	22386	850998	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	850998	733	2,83	61,03	7,404
5	173	22411	850998	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		Достоверность	Лифт
				Кол-во	%		
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.

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

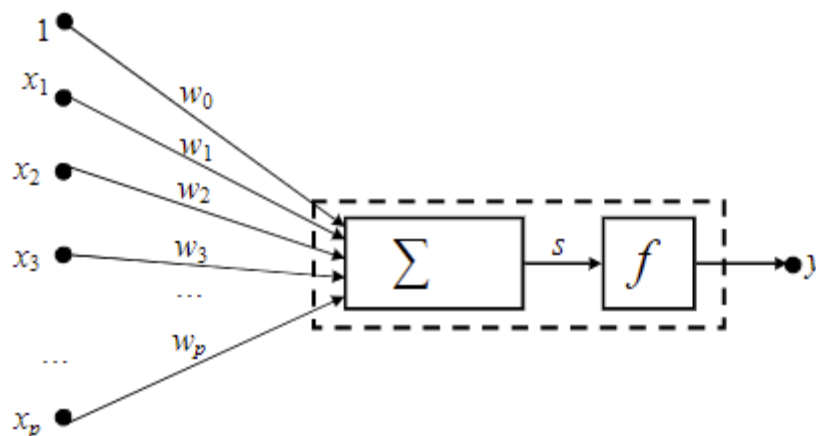
Контрольные вопросы

1. Что лежит в основе ассоциативного анализа?
2. Что описывают ассоциативные правила?
3. Какие методы используются при поиске АП?
4. Какие бывают показатели значимости АП?

Практическая работа 4. Классификация с помощью персептрона

Теоретические сведения

Искусственный нейрон составляет основу искусственной нейронной сети. Функционирование нейрона определяется его текущим состоянием, подобно тому, как это происходит в нервной системе, где нервные клетки могут находиться в возбужденном или заторможенном состоянии. У каждого нейрона есть группа синапсов (входных связей), соединенных с выходами других нейронов, ядро нейрона составляет ячейка нелинейного преобразователя, а на выходе – аксон, с которого сигнал (возбуждения или торможения) поступает на входные синапсы следующих нейронов. Общая структура нейрона приведен на рис. 1



Функционирование синапсов характеризуется величиной синаптической связи или весом синапса w_i . Синапс вычисляет произведение входного сигнала x_i на синаптический вес и передает получившееся произведение на выход. Далее сигналы попадают на вход сумматора (Σ). Соответственно, текущее состояние нейрона определяется, как взвешенная сумма его входов:

$$s = w_0 + \sum_{j=1}^p x_j w_j.$$

Следующий элемент нейрона – нелинейный преобразователь (f). Соответственно, выход нейрона есть функция его состояния $y=f(s)$. Математическое описание модели искусственного нейрона может быть записано в виде:

$$y = f(w_0 + \sum_{j=1}^P x_j w_j).$$

Функция нелинейного преобразователя называется функцией активации нейрона или его передаточной функцией. Функция активации может иметь различный вид. Вид функции во многом определяет способность ИНС решать прогностические и классификационные задачи при малом числе элементов. Кроме того, существуют ограничения в применимости некоторых функций активации в ИНС к наиболее эффективным методам обучения на базе алгоритмов обратного распространения ошибки.

Примеры функции нелинейного преобразователя

Название функции	Формула	Область значений	График
Пороговая	$f(s) = \begin{cases} 0, s < 0, \\ 1, s \geq 0 \end{cases}$	$[0, +1]$	
Сигнатурная (знаковая)	$f(s) = \begin{cases} 0, s < 0, \\ 1, s \geq 0 \end{cases}$	$[-1, +1]$	
Сигмоидальная (логистическая)	$f(s) = \frac{1}{1 + e^{-sx}}$	$(0, +1)$	
Полулинейная	$f(s) = \begin{cases} s, s > 0, \\ 0, s \leq 0 \end{cases}$	$[0, +\infty)$	
Линейная	$f(s) = s$	$(-\infty; +\infty)$	

Наиболее распространенной является сигмоидальная функция, функция S-образного вида

$$f(s) = \frac{1}{1 + e^{-sx}}.$$

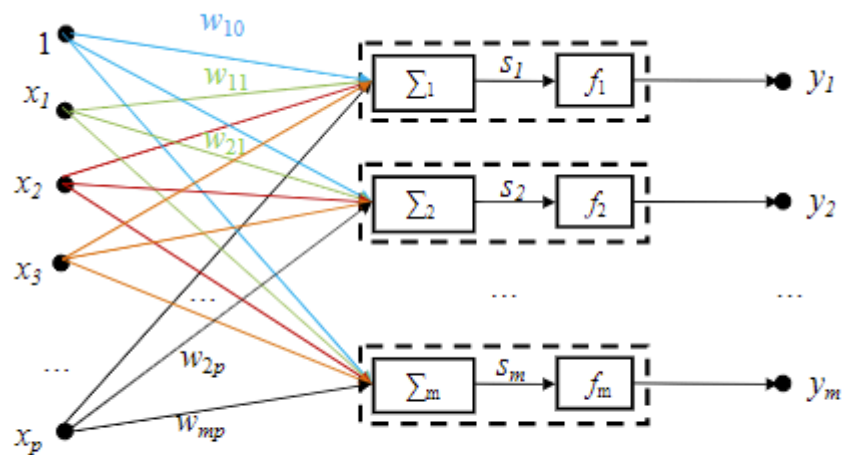
При уменьшении S сигмоида становится более полой, в пределе при $S = 0$ вырождаясь в горизонтальную линию на уровне 0,5, при увеличении S сигмоида приближается по внешнему виду к функции единичного скачка с порогом в точке $s = 0$. Значение выходного сигнала нейрона лежит в диапазоне $(0, +1)$.

Многослойные нейронные сети прямого распространения состоят из входного, выходного и расположенных между ними одного или не-скольких скрытых слоев нейронов. На примере двухслойной нейронной сети рассмотрим принцип ее работы. На p входов поступают сигналы, преобразующие в синапсах и поступающие на сумматоры на m нейронов, образующие скрытый первый слой и выдающие m сигналов.

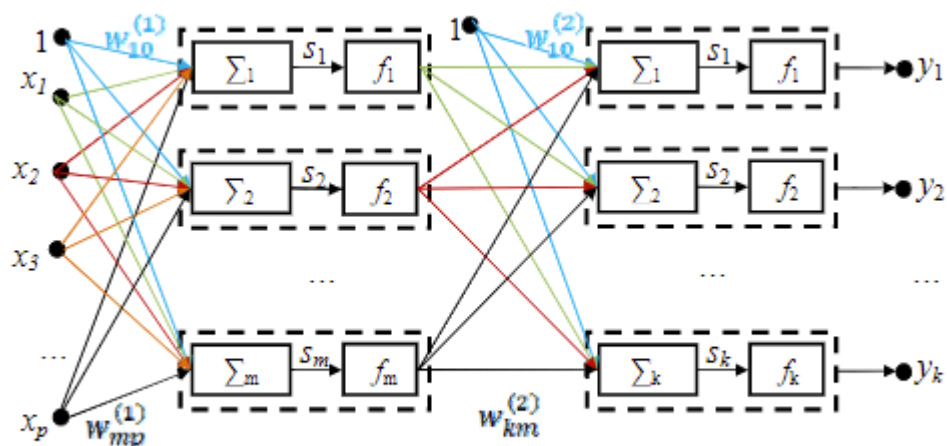
Затем полученные m сигналы, преобразующие в синапсах и поступающие на сумматоры на k нейронов, образующие скрытый второй слой и выдающие k сигналов:

$$y_i = f\left(w_{i0}^{(2)} + \sum_{j=1}^m f(s_j) w_{ij}^{(2)}\right), \text{ где } i = 1, \dots, k,$$

$$s_j = w_{j0}^{(1)} + \sum_{i=1}^p x_i w_{ji}^{(1)}, \quad j = 1, \dots, m.$$



Однослойная нейронная сеть



Двухслойная нейронная сеть

Классификации однослойным персептроном на 2 класса

Однослойный персептрон - ИНС, состоящая из одного слоя нейронов, каждый из которых имеет пороговую функцию активации.

Персептрон представляет собой простейшую нейронная сеть, параметры которой могут быть настроены на решение задачи классификации входных векторов.

Персептроны строятся на базе нейронов с пороговой функцией активации с жесткими ограничениями.

При выполнении такта функционирования персептрона каждое значение элемента входного вектора умножается на соответствующий синаптический вес, затем эти значения суммируются результат поступает на вход функции активации. Если значение входа функции активации $n \geq 0$, то нелинейный преобразователь персептрона возвращает 1, если $n < 0$, то 0.

Использование пороговой функции активации позволяет персептрону классифицировать входные вектора, разделяя пространство входов на две области. Пространство входов делится на две области разделяющей гиперплоскостью (в случае двумерного пространства входов это линия, в случае трехмерного – плоскость). Эта гиперплоскость ортогональна вектору весов w и смещена на величину b . Точки, соответствующие входным векторам выше разделяющей гиперплоскости (например, линии),

соответствуют положительному потенциалу нейрона, и, следовательно, выход персептрона для этих векторов будет равен 1. Те, которые расположены ниже соответствуют выходу персептрона, равному 0.

Изменение значений обучаемых параметров – синаптических весов и смещений – граница разделяющей гиперплоскости изменяет свое положение.

Персептрон без смещения всегда формирует разделяющую гиперплоскость, проходящую через начало координат; добавление смещения позволяет сформировать гиперплоскость, которая не проходит через начало координат.

Работа персептрона заключается в классификации (отнесении) входных n -мерных сигналов к одному из некоторого числа классов. Как видно из предыдущих абзацев, с математической точки зрения решение этой задачи описывается путем разбиения n -мерного гиперпространства $(n-1)$ -мерными гиперплоскостями. Двумерное пространство (плоскость) делится линиями, трехмерное пространство – двумерной плоскостью и т. д. В случае однослойного персептрона это можно записать так:

$$\sum_{i=1}^n x_i w_{ik} = T_k, k = 1 \dots m$$

Каждая полученная при таком делении область является областью определения одного из классов выбранной классификации. Число таких классов для одной персептронной ИНС не превышает 2^m , где m – число выходов сети.

Пример выполнения задания

Обучить однослойную ИНС из одного нейрона с пороговой функцией активации – классификации двумерных входных векторов на два класса:

```
np3.py - G:\курсы анализ данных\np3.py (3.11.0)
File Edit Format Run Options Window Help

import matplotlib.pyplot as plt
import numpy as np

def predict(row, weights, delta):
    activation = delta
    for i in range(len(row) - 1):
        activation += weights[i] * row[i]
    return 1.0 if activation >= 0.0 else 0.0

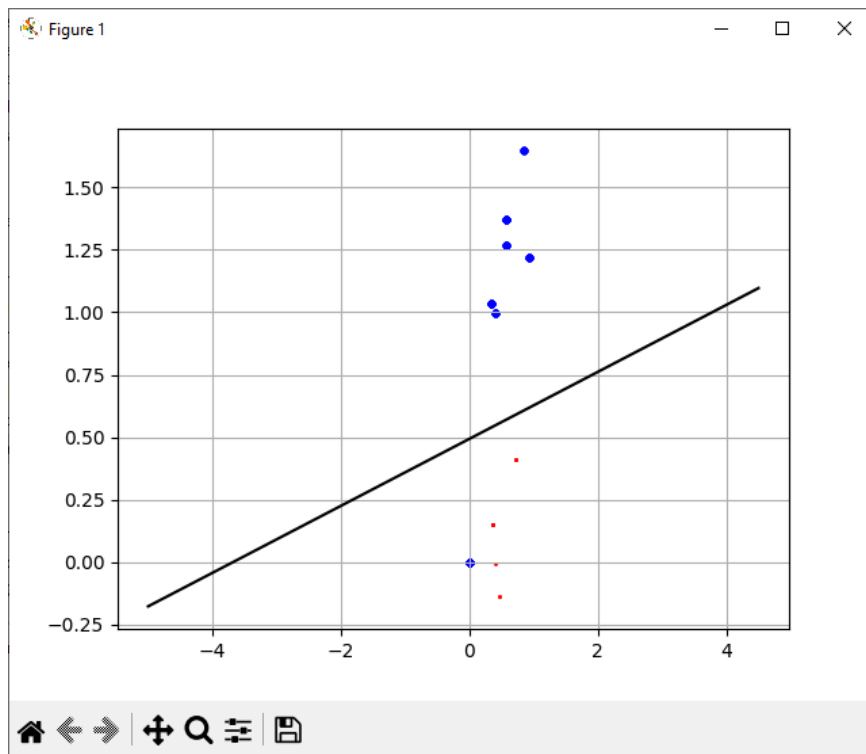
def train_we(train, lr, n_epoch):
    weights = [0.0 for _ in range(len(train[0]) - 1)]
    for epoch in range(n_epoch):
        delta = 0.0
        sum_error = 0.0
        for row in train:
            error = row[-1] - predict(row, weights, delta)
            sum_error += error ** 2
            delta += lr * error
            for i in range(len(row) - 1):
                weights[i] += lr * error * row[i]
            print('>epoch=%d, lrate = %.3f, error = %.3f' % (epoch, lr, sum_error))
    return weights, delta

def line(x1, weights):
    return ((weights[1] * x1) / -weights[2]) - (weights[0] / weights[2])

N = 10
dataset = [[0] * 4] * 10
x1 = np.random.random(N)
x2 = x1 + [np.random.randint(10) / 10 for _ in range(N)]
x3 = np.random.random(N)
x4 = x3 - [np.random.randint(10) / 10 for _ in range(N)] - 0.1
for i in range(10):
    if i <= 5:
        dataset[i] = [1, x1[i], x2[i], 0]
    else:
        dataset[i] = [1, x3[i], x4[i], 1]
    l_rate = 0.1
    n_epoch = 1000
    weights, delta = train_we(dataset, l_rate, n_epoch)
    print(weights)
    for row in dataset:
        predictit = predict(row, weights, delta)
        print('Выбрано: %d, предсказано: %d' % (row[-1], predictit))
    for row in dataset:
        plt.scatter(row[1], row[2], s=1, c='red') if row[-1] == 1 else plt.scatter(row[1], row[2], s=10, c='blue')

x1_range = np.arange(-5, 5, 0.5)
plt.plot(x1_range, line(x1_range, weights), color='black')
plt.grid(True)
plt.show()
```

Ln: 1 Col: 0



Задание для выполнения

1. Построить нейронную сеть, которая производит классификацию на 2 класса. Произвести начальную инициализацию весов нейронной сети;
2. Построить обучающую выборку, которая позволяет правильно классифицировать заданную проверочную выборку. Координаты точек обучающей выборки должны быть подобраны геометрически. Для этого первоначально нанести точки проверочной выборки на график, ориентировочно построить вокруг каждой из этих точек несколько точек обучающей выборки, затем записать координаты этих точек в массив обучающей выборки. На каждую из точек проверочной выборки должно приходиться не менее 3 точек обучающей выборки;
3. Показать обучающую выборку на графике;
4. Произвести обучение нейронной сети на составленной обучающей выборке. В процессе обучения показывать изменение линий разбиения на классы в нейронной сети;
5. Используя обученную нейронную сеть, произвести классификацию индивидуального для каждого варианта проверочного множества. Поскольку

номер класса для каждой точки может меняться при каждом новом обучении сети, номера классов в таблице выведены условно. Точки с разными номерами классов в таблице при классификации должны иметь разные номера классов, точки с одинаковыми – одинаковые. Привести график классификации точек. Вывести результаты работы сети на график. Для графика возможно применение различных цветов.

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

Контрольные вопросы

1. Опишите структуру искусственного нейрона
2. Перечислите активационные функции
3. Из чего состоят многослойные нейронные сети прямого распространения?
4. Как выполняется классификации однослойным персептроном?

Практическая работа 4. Аппроксимация функции при помощи искусственных нейронных сетей

Теоретические сведения

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;

β_i – весовые коэффициенты, задающие относительную важность i -й точки таким образом, что рост β_i 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 объема 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.

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

где y_i^o – 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

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

Во втором варианте ошибка пересчитывается после каждого примера:

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

Установим

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

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

Тогда

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

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

где y_i – функция активации

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

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

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

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

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

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

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_j^n для нейронов выходного слоя по формуле для внешнего нейрона.

Шаг 4. Рассчитываются значения δ_j^n для всех внутренних нейронов по формуле.

Шаг 5. Вычисляются приращения весовых коэффициентов ΔW_{ij}^n .

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_{ij}^n 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.

Пример выполнения задания

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 + 2.718281828459045235360287471352662497757 ** -a)
        error = y[i] - b
        errorreturn = errorreturn + abs(error)
        delta = (error) * 1 / (1 +
2.718281828459045235360287471352662497757 ** -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 + 2.718281828459045235360287471352662497757 ** -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

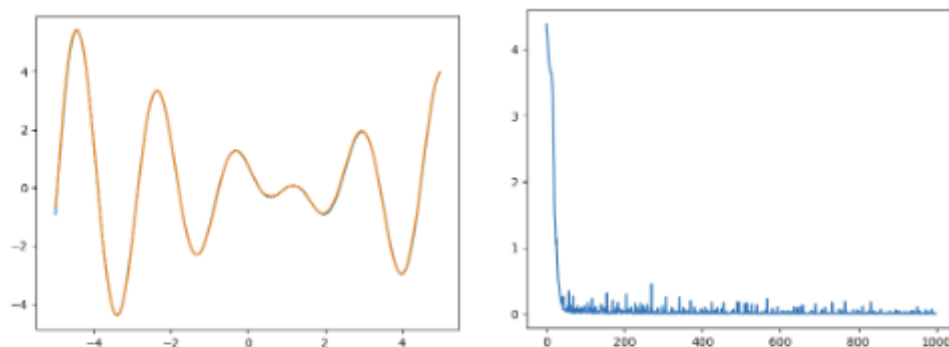


Рис. 3.1. Результат аппроксимации при функции активации ReLu со средним квадратом ошибок и график функции потерь (средний квадрат ошибок)

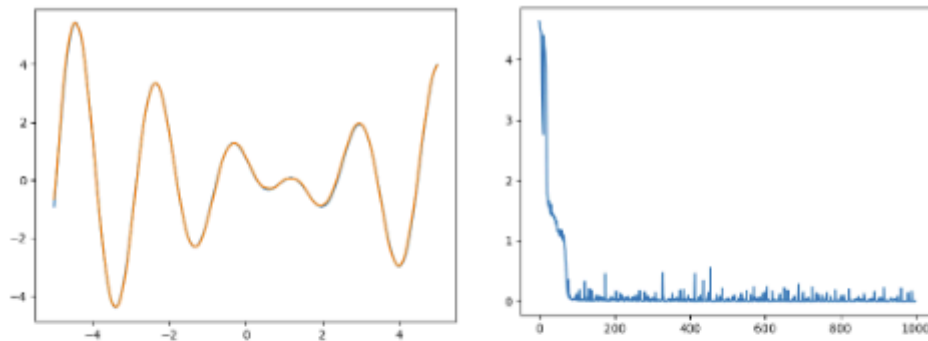


Рис. 3.2. Результат аппроксимации функции, при функции активации гиперболического тангенса и график функции потерь

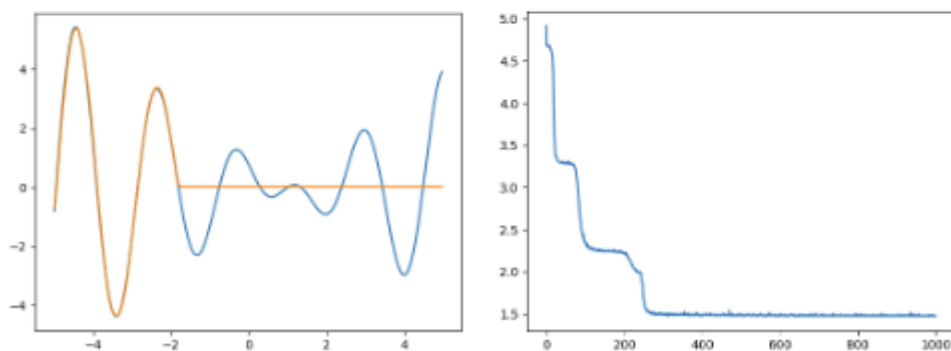


Рис. 3.2. Результат аппроксимации функции, при сигмоидальной функции активации и график функции потерь

Задание для выполнения

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	$\sin(x)/x^2$	$x \in [3.1, 20]$	3
4	$\sin(2x)/x^2$	$x \in [-20, -3.1]$	4
5	$\cos(2x)/x^2$	$x \in [-20, -2.3]$	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

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

Контрольные вопросы

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?

Практическая работа 5. Прогнозирование временных рядов с помощью НС

Теоретические сведения

Временной ряд (ВР) – это последовательность значений, описывающих протекающий во времени процесс, измеренных в последовательные моменты времени, обычно через равные временные интервалы, то есть ряд наблюдений $x(t_1), x(t_2), \dots, x(t_N)$ анализируемой величины $x(t)$, произведенных в последовательные моменты времени $t_1, t_2, \dots, t_N$.

Данные типа временных рядов широко распространены в самых разных областях человеческой деятельности. В экономике это ежедневные цены на акции, курсы валют, еженедельные и месячные объемы продаж, годовые объемы производства и т. п.

Цели анализа временных рядов следующие:

- 1) краткое описание характерных особенностей ряда;
- 2) подбор статистической модели, описывающей ВР;
- 3) предсказание будущих значений на основе прошлых наблюдений;
- 4) управление процессом, порождающим ВР.

Сформулируем постановку задачи одношагового прогнозирования:

Имеется временной ряд Y , заданный значениями измерений величины, в предшествующие моменты времени

$$Y_{t-k}, Y_{t-k+1}, \dots, Y_t$$

где t -текущий момент времени, k -глубина исторической выборки а также временным шагом (или «лагом» от английского lag – запаздывание).

Необходимо спрогнозировать величину измеряемого параметра в следующий момент времени, на основе анализа прошлых значений, а также истории изменения других факторов, влияющих на динамику прогнозируемой величины.

Существует два различающихся подхода к прогнозированию. При первом важным является минимальная величина ошибки прогнозирования.

При втором (который, в частности, применяется при прогнозе биржевых курсов для управления торгами) важно правильное определение характера дальнейшего изменения значения прогнозируемой величины – рост или уменьшение.

При решении задачи прогнозирования временного ряда необходимо найти значение (или несколько величин, вектор) величины прогнозируемого параметра Y в будущий момент времени на основе известных предыдущих значений

$$Y_{t+1} = g(Y_{t-k} \dots Y_t, V_{1,t-k} \dots V_{1,t}, V_{n,t-k} \dots V_{n,t}, C_{1,t-k} \dots C_{1,t}, C_{n,t-k} \dots C_{n,t}).$$

(1)

Формула (1) описывает основное соотношение, определяющее прогноз временного ряда.

Определение структуры нейронной сети для прогноза ВР

Определение структуры нейронной сети предполагает задание ее архитектуры (число слоев, число нейронов на каждом слое), а также функции активации нейронов.

При использовании многослойного персептрона для решения задач аппроксимации функций (в том числе и задач прогнозирования) нет необходимости в числе скрытых слоев больше одного. Таким образом, предлагается использовать нейронную сеть, имеющую в своем составе входной, один скрытый и выходной слои.

Число нейронов первого (то есть входного) слоя $n^{(1)}$ зависит от количества входных факторов и глубины ретроспективной выборки по каждому фактору и определяется по формуле:

$$n^{(1)} = \sum_p P \cdot k_p \quad (2)$$

где P – число входных факторов;

k_p – глубина ретроспективной выборки по p -ому фактору.

Таким образом, число нейронов входного слоя равно числу элементов входных данных.

Выходной слой содержит один нейрон, выход которого представляет собой полученное в результате работы нейронной сети значение прогноза.

При необходимости получения одновременно нескольких выходных значений (например, котировок продажи и покупки ценной бумаги) предпочтительнее не увеличивать число выходных нейронов, а обучать несколько отдельных нейронных сетей, каждая из которых имеет один выход.

В пользу предлагаемого подхода свидетельствуют результаты исследований. Так, в частности, в результате экспериментов было установлено, что точность прогноза нейросети с двумя выходами хуже, чем двух нейросетей с единственным выходом при обучении на одних и тех же данных.

Число нейронов второго (скрытого) уровня первоначально предлагается принимать равным полусумме числа нейронов входного и выходного слоев

$$n^{(2)} = \frac{\sum P \cdot k_p + 1}{2}. \quad (3)$$

В процессе определения и настройки параметров нейронной сети число скрытых нейронов может варьироваться с целью увеличения точности прогноза. При этом порядок их количества можно приблизительно оценить с помощью теоремы Колмогорова и следствия из нее. В соответствии с этими теоретическими результатами непрерывную функцию можно аппроксимировать с любой точностью при помощи трехслойной нейронной сети, которая имеет N входных, $2N + 1$ скрытых и один выходной нейрон.

Таким образом, число $2N + 1$ теоретически можно считать верхней границей для числа нейронов на скрытых уровнях. Однако данная оценка справедлива для наилучшей сети, соответствующей глобальному минимуму адаптивного рельефа ошибки в пространстве весов. На практике достичь

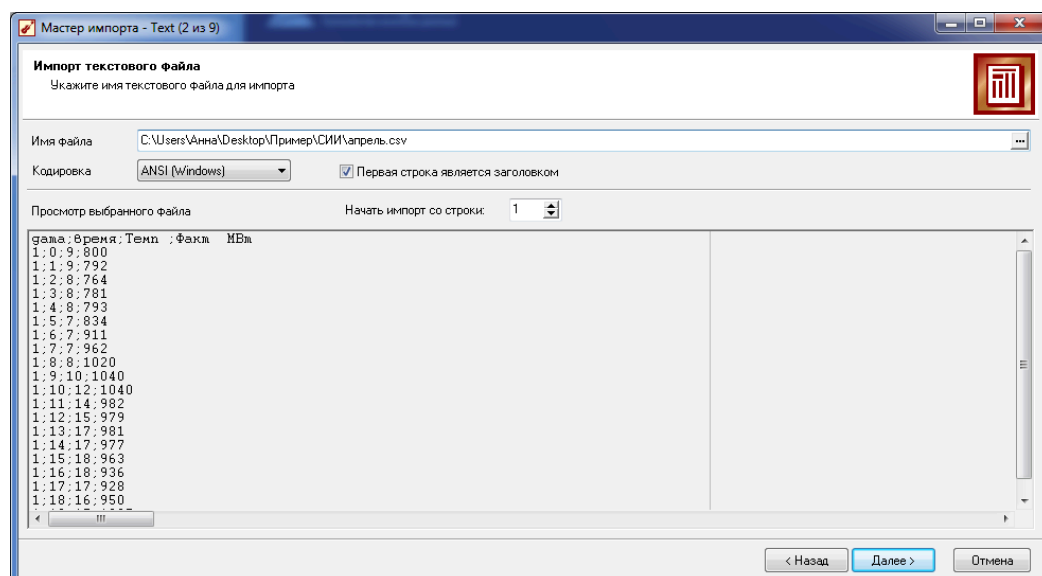
глобального минимума обычно не удастся. Кроме того, имея $2N + 1$ скрытых нейронов, сеть способна в точности запомнить обучающие примеры.

В то же время в реальности от нее требуется их обобщить, выявив скрытые закономерности. Таким образом, оценка Колмогорова, по сути, сделана для другой задачи и в данном случае служит лишь ориентиром. В ходе экспериментов эмпирическим путем было установлено, что число скрытых нейронов может составлять от $N/2$ до $3N$.

В качестве функции активации нейронов предлагается использовать биполярную сигмоиду. Сигмоидная функция традиционно используется в сетях типа «многослойный персептрон» так как она является гладкой, дифференцируемой, а ее производная выражается через саму функцию, что важно для обучения сети. В тоже время рациональная сигмоида с областью определения $[0;1]$ имеет недостаток, связанный с ее несимметричностью относительно оси абсцисс, который уменьшает скорость обучения.

Пример выполнения задания

В пакете Deductor Studio с помощью кнопки «Мастер импорта» необходимо загрузить данные из заданного файла



На шаге 4 выбрать правильный разделитель строк, чтобы в окне предварительного просмотра данные отобразились в виде таблицы

Мастер импорта - Text (4 из 9)

Параметры импорта файла с разделителями
Укажите символ-разделитель столбцов и другие вспомогательные параметры импорта

Символом-разделителем является

☐ Символ табуляции
 ☐ Пробел
 ☐ Точка
 ☒ Точка с запятой
 ☐ Запятая
 ☐ Другой

☐ Считать последовательные разделители одним

	дата	время	Темп	Факт МВт
▶	1	0	9	800
	1	1	9	792
	1	2	8	764
	1	3	8	781
	1	4	8	793
	1	5	7	834
	1	6	7	911
	1	7	7	962
	1	8	8	1020
	1	9	10	1040

< Назад Далее > Отмена

На следующем шаге задать параметры столбцов: дата и время – входное, Факт - выходное

Мастер импорта - Text (6 из 9)

Импорт текстового файла
Укажите параметры столбцов

Имя столбца	COL4
9.0 дата	
9.0 время	
9.0 Темп	
9.0 Факт МВт	

Метка столбца: Факт МВт

Тип данных: 9.0 Вещественный

Вид данных: — Непрерывный

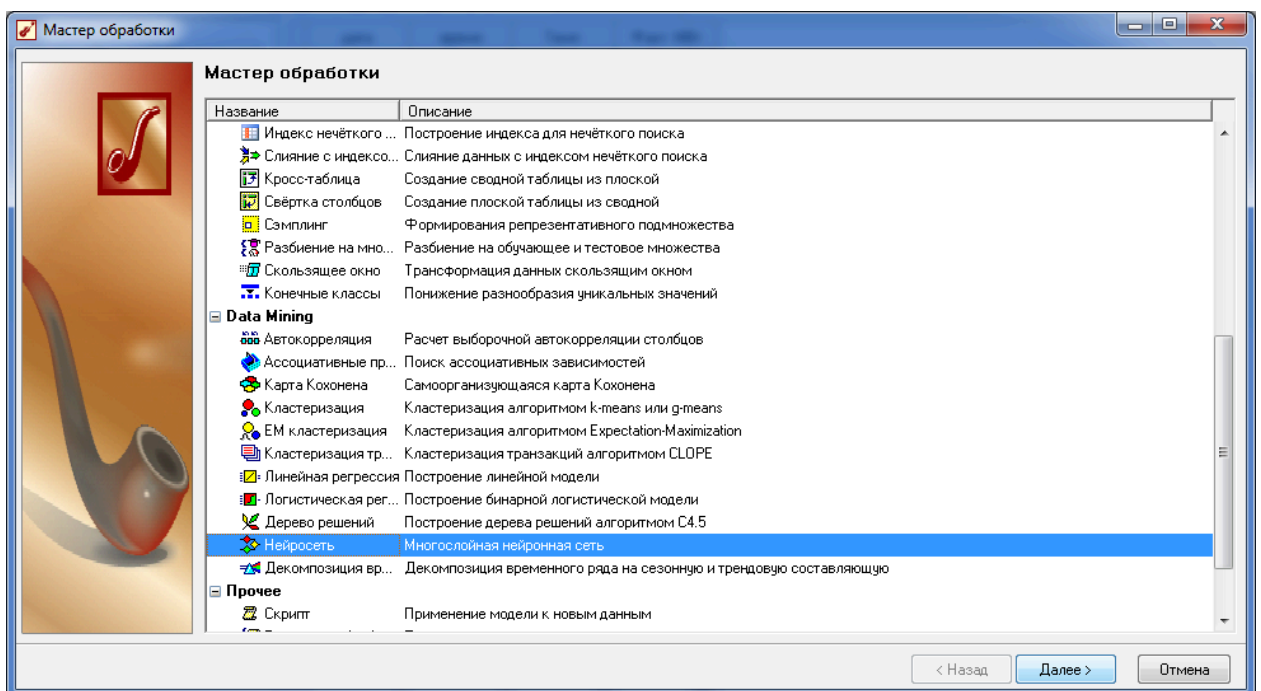
Назначение: Выходное

< Назад Далее > Отмена

После окончания загрузки данные будут отображены в виде таблицы:

дата	время	Темп	Факт MBT
1	0	9	800
1	1	9	792
1	2	8	764
1	3	8	781
1	4	8	793
1	5	7	834
1	6	7	911
1	7	7	962
1	8	8	1020
1	9	10	1040
1	10	12	1040
1	11	14	982
1	12	15	979
1	13	17	981
1	14	17	977
1	15	18	963
1	16	18	936
1	17	17	928
1	18	16	950
1	19	15	1087
1	20	14	1136
1	21	13	1063
1	22	13	965
1	23	12	962
2	0	12	806
2	1	12	774
2	2	11	767
2	3	11	759
2	4	11	769
2	5	11	845
2	6	10	919
2	7	10	945
2	8	11	1020
2	9	13	1024
2	10	14	1022
2	11	15	978

Далее вызываем Мастер обработки – нейросеть:



Задаем входные и выходные параметры:

Мастер обработки - Нейросеть (2 из 9)

Настройка назначений столбцов

Задайте назначения исходных столбцов данных

- дата
- время
- Темп
- Факт MBt

Имя столбца: COL3

Тип данных: Вещественный

Назначение: ☒ Неиспользуемое

Вид данных: Непрерывный

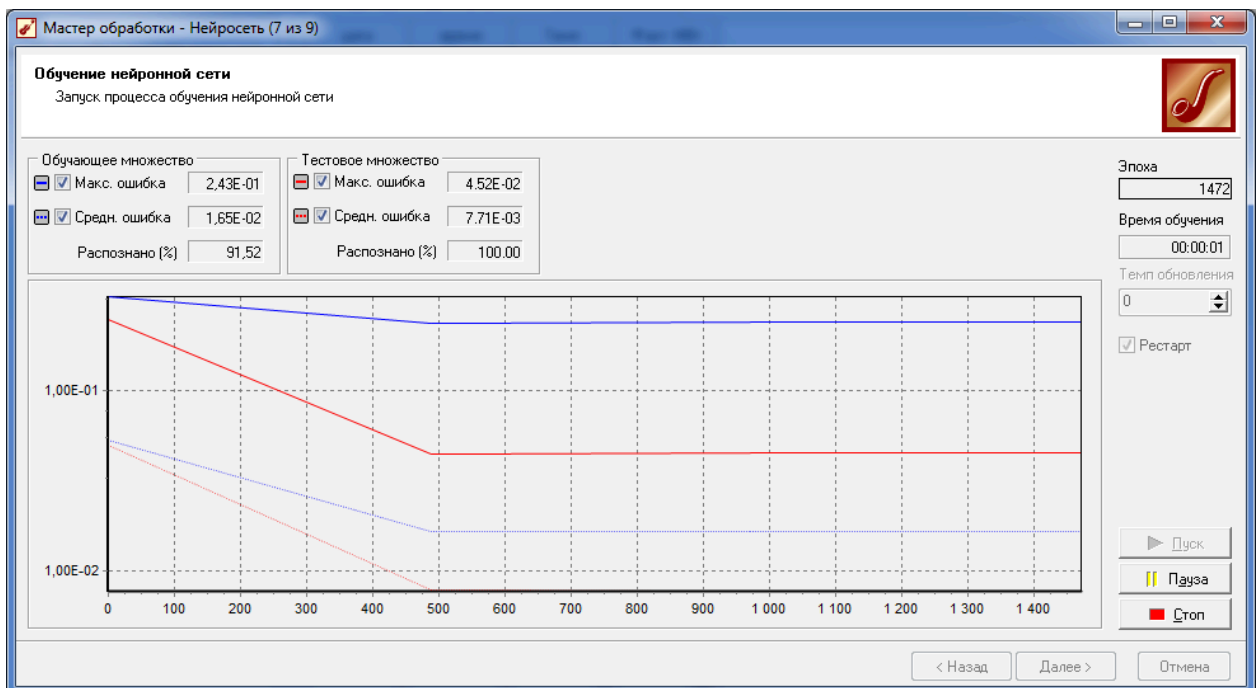
Статистика

Минимум	-3
Максимум	22
Среднее	8,95138888888889
Стандартное откл.	5,31841456960903

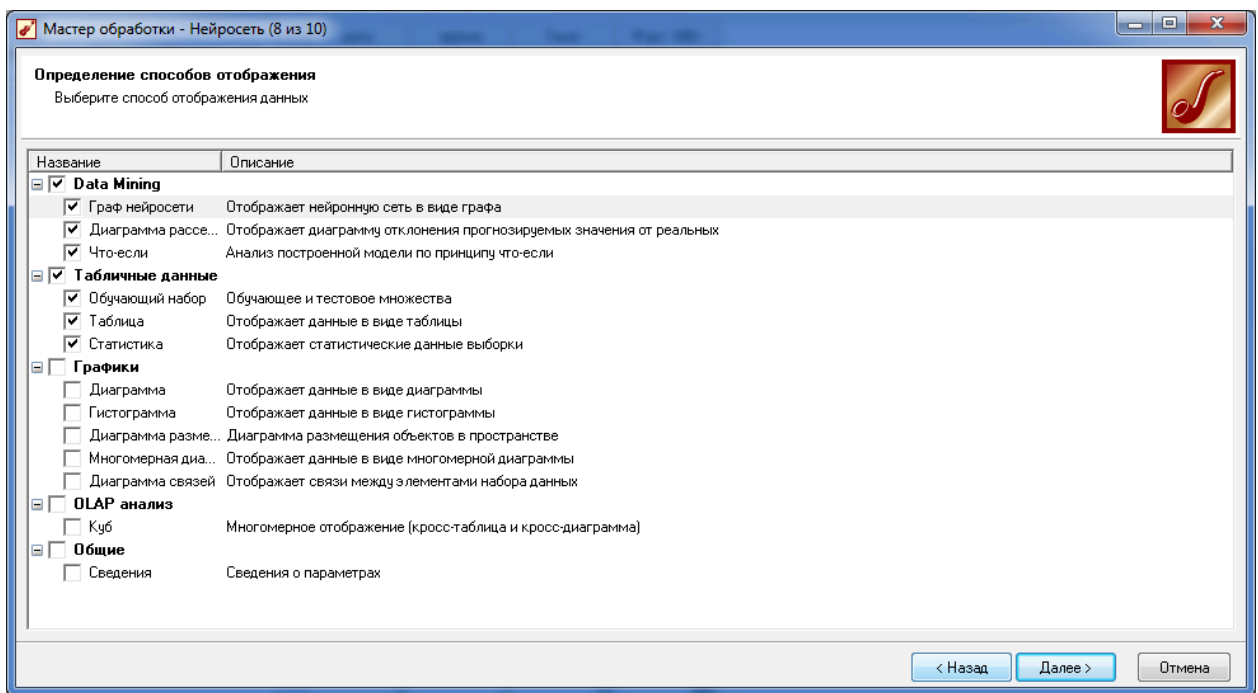
Настройка нормализации...

< Назад Далее > Отмена

Остальные параметры оставляем по умолчанию, запускаем сеть на обучение



Для визуализации данных отмечает следующие виды:



Задание для выполнения

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.

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения

задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

Контрольные вопросы

1. Для чего выполняется анализ временных рядов?
2. В чем особенности прогнозирования временного ряда?
3. Как определяется структура НС для прогнозирования ВР?

Практическая работа 6. Классификация с помощью сетей Кохонена

Теоретические сведения

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.

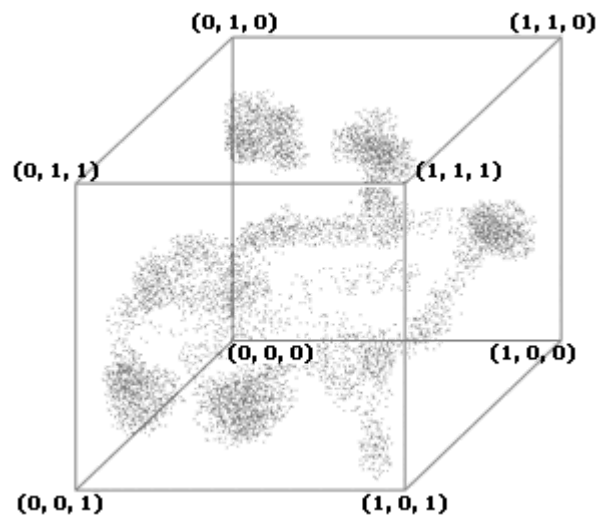
С точки зрения извлечения новых знаний кластеризация представляет больший интерес, чем классификация. Действительно, при классификации любой объект, предъявляемый модели, обязательно будет отнесен к одному из заранее определённых классов, даже если в действительности он ни к одному из классов не относится. 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 (проекция Саммона).

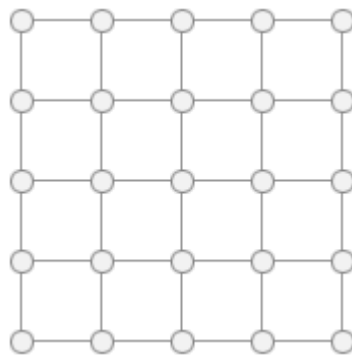
Рассмотрим, как решаются такие задачи и как карты Кохонена находят закономерности в исходных данных. Для упрощения рассмотрения будем считать, что объекты имеют 3 признака (на самом деле их может быть любое количество).

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.

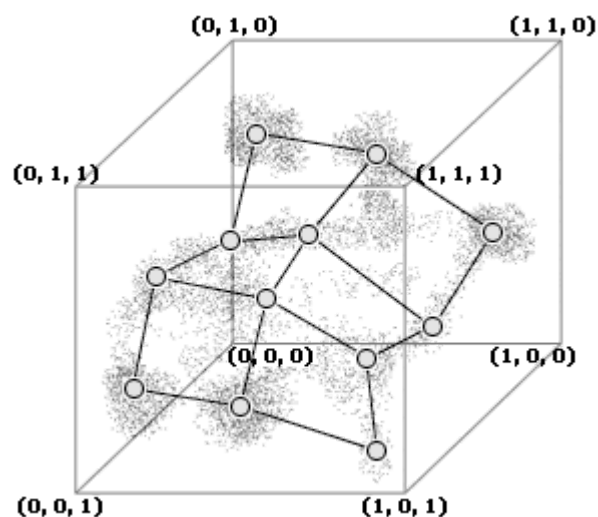


Взглянув на этот рисунок, мы можем увидеть, как расположены объекты в пространстве, причем легко заметить участки, где объекты группируются, т.е. у них схожи параметры, значит, и сами эти объекты, скорее всего, принадлежат одной группе. Но так легко можно поступить

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 получить 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 задания

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.

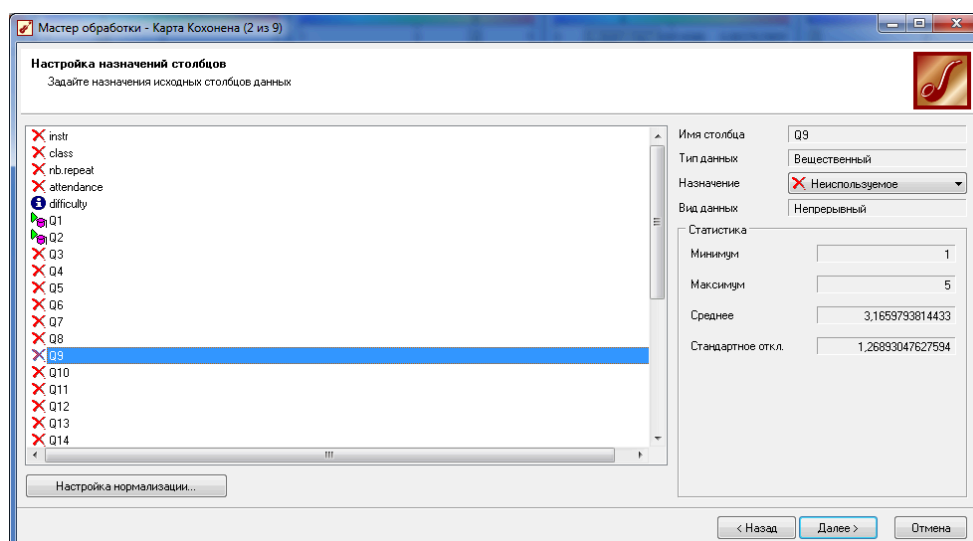
Все Q1-Q28 относятся к типу Лайкерта, что означает, что значения взяты из набора $1,2,3,4,5$.

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.

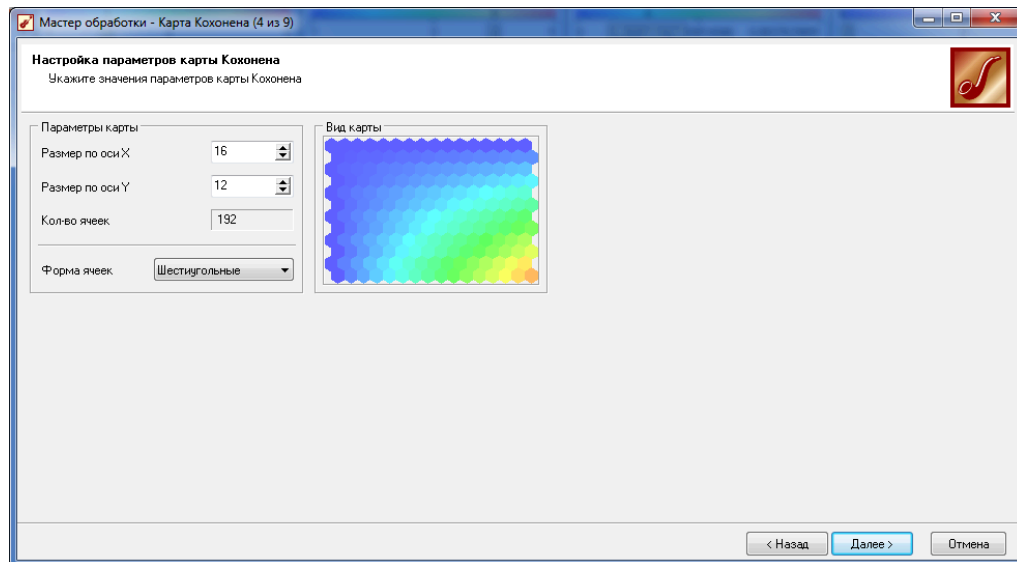
Таким образом, все столбцы, которые мы хотим использовать для построения карты, следует выбрать в качестве входных. Выберем для примера значения Q1 и Q2.

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)



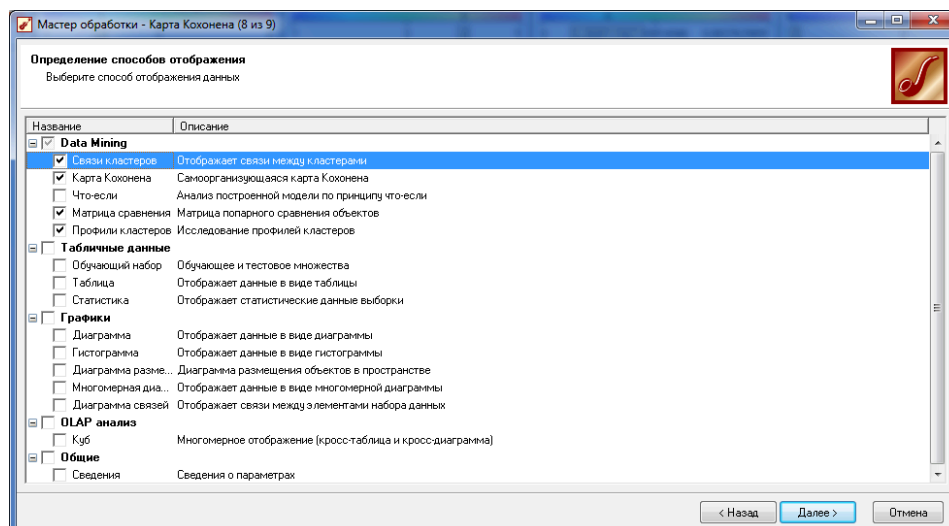
Размер карты, наилучшим образом отображающий структуру кластеров, заранее неизвестен и вообще субъективен. Тем не менее, можно рекомендовать начальное число ячеек в 2-3 раза больше числа обучающих примеров. Что касается формы ячеек, то, как отмечено выше, шестиугольная форма более корректно отображает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

На 6-м шаге задаются параметры обучения карты. В большинстве случаев их оптимальные значения заранее неизвестны и требуют экспериментального подбора. Способ начальной инициализации карты определяет, каким образом 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.

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 по признаку, 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.

Задание для 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.

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

Контрольные вопросы

1. Чем отличаются задачи классификации и кластеризации?
2. Какой вид обучения используется в Картах Кохонена?
3. Какой принцип используется в Картах Кохонена?

Практическая работа 7. Генетические алгоритмы

Теоретические сведения

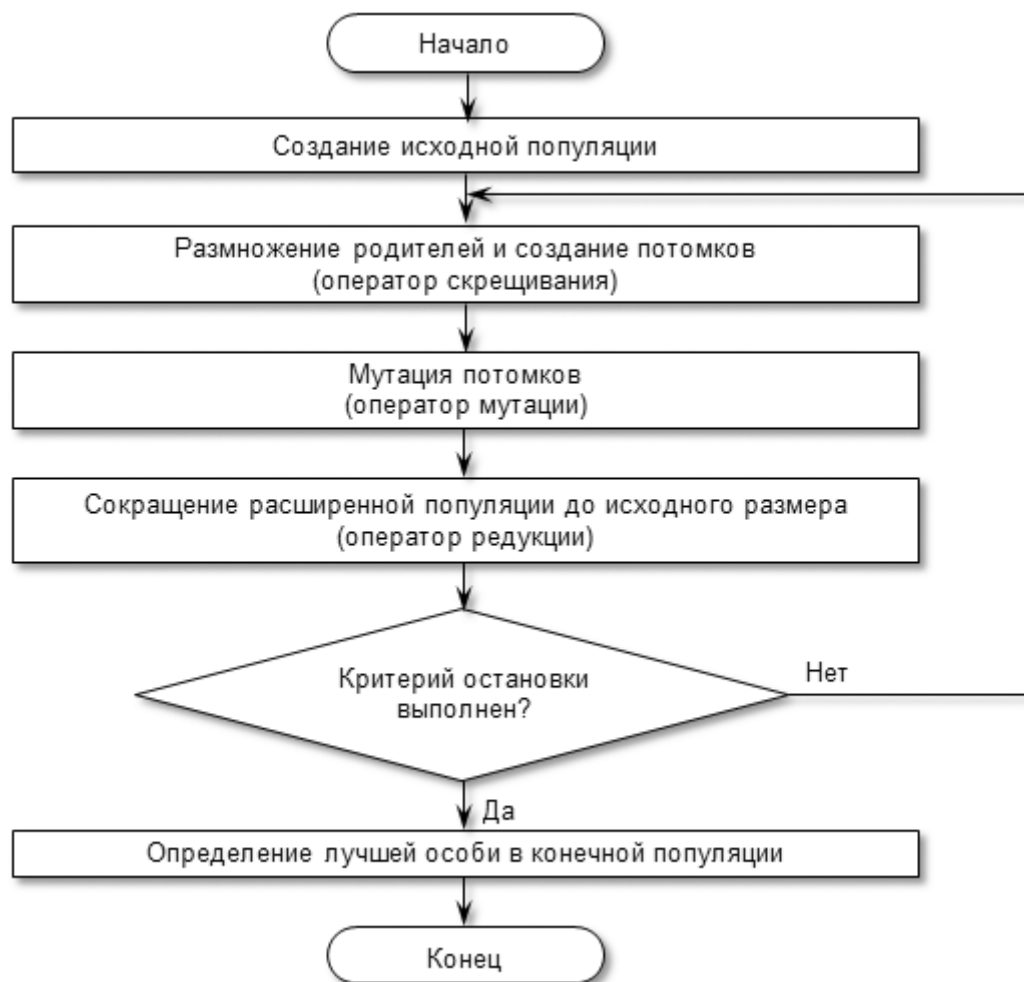
Генетический алгоритм — это алгоритм, использующий принципы естественного отбора (биология) для нахождения решения задачи. Является разновидностью эволюционных вычислений.

Отличительной особенностью генетического алгоритма является акцент на использование оператора «кроссовера», который производит операцию, роль которой аналогична роли скрещивания в живой природе.

Задача кодируется таким образом, чтобы ее решение могло быть представлено в виде вектора («хромосома»). Случайным образом создается некоторое количество начальных векторов («начальная популяция»). Они оцениваются с использованием «функции приспособленности», в результате чего каждому вектору присваивается определенное значение («приспособленность»), которое определяет вероятность выживания организма, представленного данным вектором.

После этого с использованием полученных значений приспособленности выбираются векторы (селекция), допущенные к «скрещиванию». К этим векторам применяются «генетические операторы» (в большинстве случаев «кроссовер» и «мутация»), создавая, таким образом, следующее «поколение». Особи следующего поколения также оцениваются, затем производится селекция, применяются генетические операторы и т. д. Так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (поколений), пока не будет выполнен критерий останова алгоритма.

Общая схема работы алгоритма представлена на следующем рисунке.



Критерием остановки работы генетического алгоритма может быть одно из следующих событий

- сформировано заданное число поколений;
- исчерпано время, отведенное на эволюцию;
- популяция достигла заданного качества (значение критерия одной (нескольких, всех) особей превысило заданный порог);
- достигнут некоторый уровень сходимости (особи в популяции стали настолько подобными, что дальнейшее их улучшение происходит чрезвычайно медленно);
- и т.д.

Пример выполнения задания

Целевая функция задана выражением $f(x) = 4(x-10)^2 + 4$. Требуется найти максимальное и минимальное значение целевой функции в интервале $x \in [0, 30]$.

Вариант решения задачи (особь) можно представить в виде битовой строки, которая будет соответствовать целочисленному значению x в заданном интервале. Таким образом, ген – это отдельный бит строки, хромосома – последовательность из 4-5 битов, генотип состоит из одной хромосомы (генотип $\leftrightarrow$ хромосома)

Пусть размер популяции будет 4 особи, число скрещиваний – 2, число мутаций – 1 потомок на поколение. Процесс мутации заключается в инверсии одного из битов строки, выбираемого случайно.

1. Случайным образом генерируются особи исходной популяции.

особь	значение		Значение функции
	десятичное	двоичное	
1	3	0011	200
2	6	0110	68
3	15	1111	104
4	20	10100	404

2. Первая итерация – оператор скрещивания. Для скрещивания случайным образом выбраны две пары особей (1, 2) и (4, 4). В каждой паре разбиение на подстроки происходит независимо от другой пары и случайным образом. Потомки получаются в результате объединения подстроки одного родителя с подстрокой другого.

потомок	скрещивание		результат скрещивания	значение скрещивания	Значение функции
5	1 и 2	00 11	00 10	2	260
		01 10			
		11 11			
6	3 и 4		11 00	12	20

		101 00			
--	--	--------	--	--	--

3. Первая итерация – оператор мутации. Для мутации случайным образом выбран потомок 6, а в нем для инверсии случайно выбран 4 бит.

потомок	мутирующий потомок		результат мутации		Значение функции
7	6	1100	1101	13	40

4. Первая итерация – оператор редукции. Отбор лучших особей из родителей и потомков выполняется по минимальным значения целевой функции с учетом требуемого размера популяции.

Отобранные особи	десятичное	двоичное	Значение функции
2	6	110	68
3	15	1111	104
6	12	1100	20
7	13	1101	40

5. Вторая итерация – оператор скрещивания

потомок	скрещивание		результат скрещивания	значение	Значение функции
8	2 и 6	01 10	11 10	14	68
		11 00			
9	6 и 7	11 00	11 01	13	40
		11 01			

6. Вторая итерация – оператор мутации

потомок	мутирующий потомок		результат мутации		Значение функции
10	9	1101	1001	9	8

7. Вторая итерация – оператор редукции

отобранные особи	десятичное	двоичное	Значение функции
6	12	1100	20

7	13	1101	40
9	13	1101	40
10	9	1001	8

8. Третья итерация – оператор скрещивания

потомок	скрещивание		результат скрещивания	значение	Значение функции
11	6 и 10	11 00	10 00	8	20
		10 01			
12	7 и 10	11 01	10 01	9	9
		11 01			

9. Третья итерация – оператор мутации

потомок	мутирующий потомок		результат мутации		Значение функции
13	11	1000	1010	10	4

10. Третья итерация – оператор редукции

отобраны особи	десятичное	двоичное	Значение функции
6	12	1100	20
11	8	1000	20
12	9	1001	9
13	10	1010	4

Задание для выполнения:

Найти минимум функции

- 1) $Y = 3 \cdot \sin(x/2 - 5)$ на интервале $[0; 10]$
- 2) $Y = 2 \cdot (x/2 - 5)^2$ на интервале $[0; 20]$
- 3) $Y = \lg(x/4 - 5)^2$ на интервале $[0; 15]$
- 4) $Y = (4x - 8)^4$ на интервале $[0; 20]$

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

Контрольные вопросы

1. В чем особенность генетических алгоритмов?
2. Перечислите шаги генетического алгоритма
3. Что представляет собой операция кроссовера?

Практическая работа 8. Построение функции принадлежности нечеткого множества

Теоретические сведения

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.

Существуют множества типовых форм для задания функций принадлежности. Наиболее распространенные 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.

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

При $b-a=c-b$ имеет случай симметричной треугольной функции

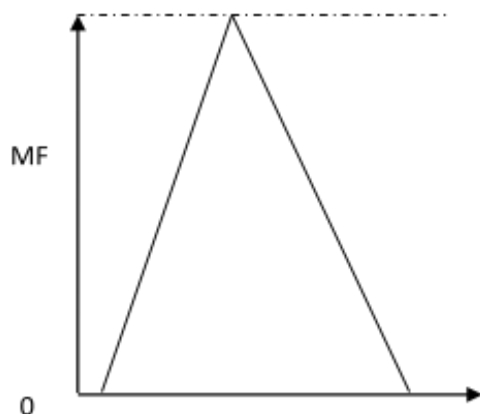
принадлежности, может быть однозначно задана двумя параметрами из трех.

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

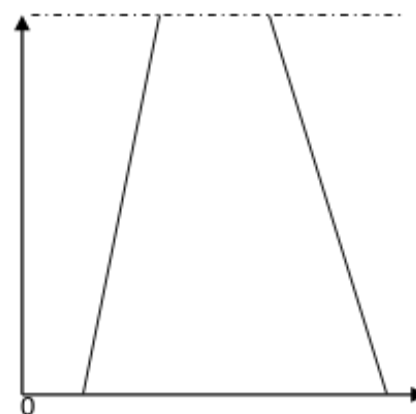
Трапецеидальная функция принадлежности.

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

1) Треугольная



2) Трапецеидальная



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

Пример выполнения задания

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; $b_{j,i}^k$ - 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.: $b_{j,i}^k \in \{0,1\}$, 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 ($j \in \overline{1,m}$) рассчитываются следующим образом:

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

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 принадлежности 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

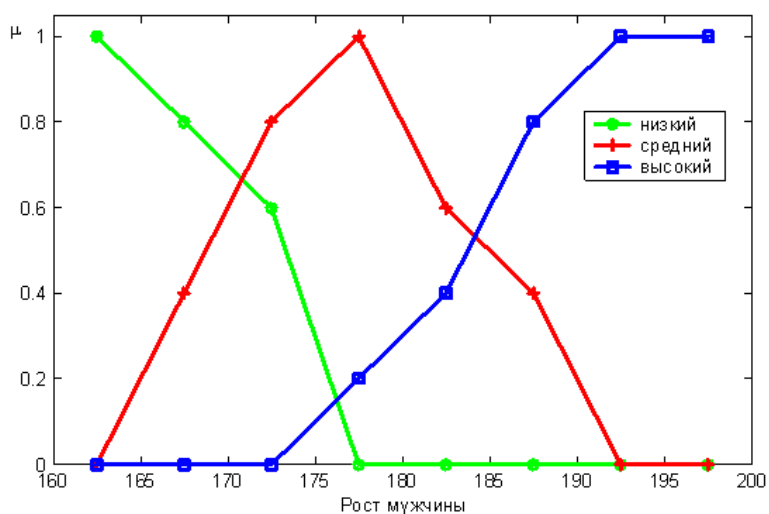


Рисунок 1. Функции принадлежности нечетких множеств

Пример 2. Построить 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:

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

Собственный вектор находится из следующей системы уравнений:

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

где $\lambda_{\max}$ - максимальное собственное значение матрицы A.

Решение:

Приведенным экспертным высказываниям соответствует такая матрица парных сравнений:

	170	175	180	185	190	195	Собственные значения этой матрицы парных сравнений равны:	
170	1	1/2	1/4	1/6	1/8	1/9		
175	2	1	1/3	1/5	1/7	1/8		
180	4	3	1	1/4	1/4	1/5		
185	6	5	4	1	1/3	1/3		
190	8	7	4	3	1	1		
195	9	8	5	3	1	1		
	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	3,2031009 5	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

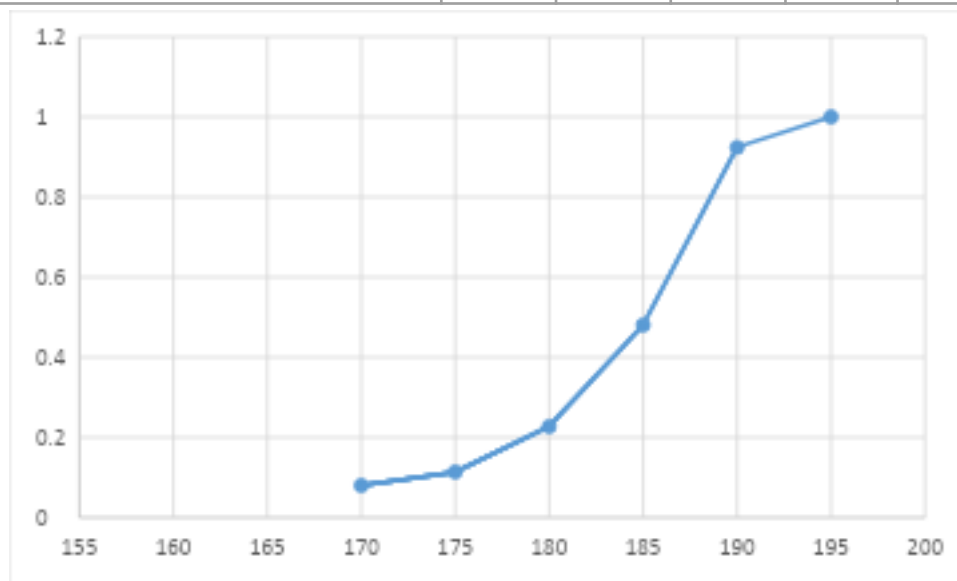


Рисунок 2. Функции принадлежности нечеткого множества "высокий мужчина"

Задание для выполнения

1. Пусть $U = \{0, 1, 2, \dots, 120\}$ – 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.
2. Решить задачу 1 с помощью метода статистической обработки экспертной информации, в качестве экспертов использовать своих сокурсников.
3. «Пользователь установил приложение на смартфон. Через некоторое время ему предлагается оценить приложение. Универсум оценок

дискретный $M = \{1, 2, 3, 4, 5\}$ где 1 – неудовлетворительное приложение, 2 – имеются существенные проблемы и т.д., 5 – отличное приложение. Пусть пользователь имеет возможность посоветовать приложение своим друзьям: универсум количества друзей $NF = \{1, 2, 3, 4, 5, 6\}$. Предложить вариант нечеткого отношения R между оценкой приложения и количеством друзей, которым оно будет порекомендовано.

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

Контрольные вопросы

1. В чем отличие нечеткого множества от четкого?
2. Что показывает функция принадлежности нечеткому множеству?
3. Назовите виды функций принадлежности нечеткому множеству
4. Перечислите характеристики нечеткого множества

Практическая работа 9. Логическое программирование. Логика предикатов

Теоретические сведения

Предикат $P(x_1, x_2, \dots, x_n)$ — это логическая функция n переменных, которая в зависимости от значений переменных $x_1, x_2, \dots, x_n$ принимает два значения: истина (Т) или ложь (F). Обычно такого рода предикат называют n -местным.

Предикат $P(a_1, a_2, \dots, a_n)$, имеющий истинное значение и полученный при подстановке или конкретизации $x_1 = a_1, x_2 = a_2, \dots, x_n = a_n$, где a_i — объекты или константы, называется фактом. Приведем примеры предикатов с различным числом аргументов.

1. Предикат нульместный или простое высказывание:

А — выключатель включен; В — вилка исправна.

2. Предикат одноместный:

положительный(x) — число $x > 0$;

большой(y) — животное y большое.

Предикаты становятся фактами при подстановке $x = 8$ и $y = \text{медведь}$, т. е.

положительный(8) = Т;

большой(медведь) = Т.

3. Предикат двухместный, выражающий отношение:

больше(x_1, y_1) — число x_1 больше, чем y_1 ;

любит(x_2, y_2) — животное x_2 любит пищу y_2 .

Первый предикат приобретает значение «истина» и является фактом при $x_1 = 2, y_1 = 1$. Второй предикат может стать фактом при такой подстановке: $x_2 = \text{овца}, y_2 = \text{трава}$.

Полученные факты имеют вид:

больше(2, 1);

любит(овца, трава).

Ложными предикаты больше (x_1, y_1) , любит (x_2, y_2) будут соответственно при $x_1 = 2, y_1 = 3$ и $x_2 = \text{заяц}, y_2 = \text{мясо}$.

4. Предикат трехместный:

сумма (x, y, z) — z является суммой x и y .

Очевидно, что при $x = 2, y = 3, z = 5$ значение этого предиката истинное, получаем факт сумма $(2, 3, 5)$, а при $x = 5, y = 7, z = 3$ — ложное.

Для предикатов справедливы все ранее рассмотренные логические операции и законы логики.

Для работы с предикатами в 1972 г. шотландским ученым Р. Ковальским был предложен язык логического программирования (Пролог). Логическое программирование основано на том, что компьютер должен решать задачу в свойственной человеку манере, и предполагает, что информация о задаче формулируется в виде предикатов-фактов и логических выражений, называемых правилами.

Правила используются в тех случаях, когда необходимо показать, что некоторый факт зависит от других фактов (условий). Правила обладают большей общностью, чем факты. Это объясняется тем, что в правилах обычно содержатся переменные. Важно помнить, что переменная может обозначать не один, а несколько различных объектов. Область действия переменной — одно правило. Кроме того, переменная обозначает один и тот же объект по всему правилу. Вот почему в процессе логического вывода все вхождения одной переменной в правило заменяются одним и тем же значением

Благодаря правилам и переменным система логического вывода позволяет выводить такие знания, которые в явном виде в базе знаний отсутствуют.

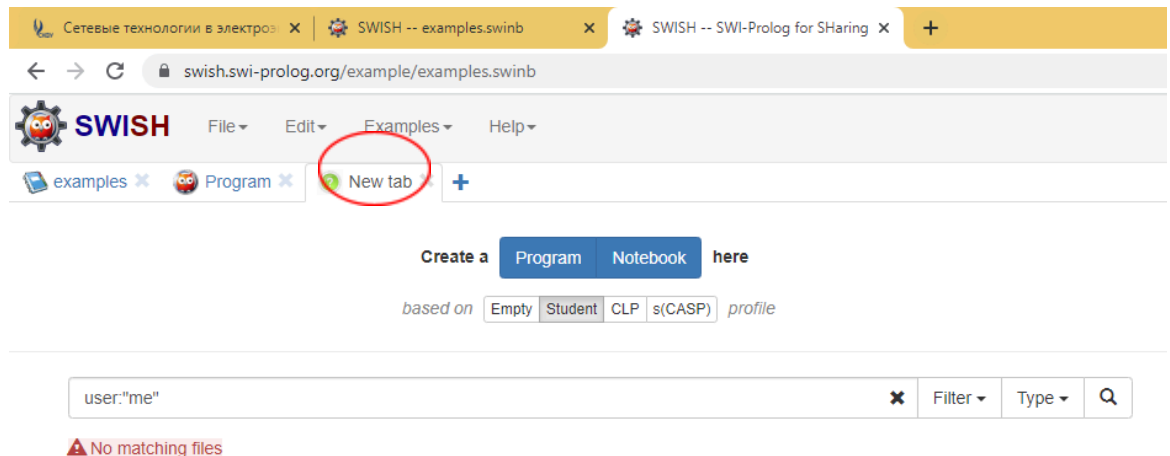
Совокупность фактов и правил образует базу знаний (БЗ). БЗ может быть использована для решения задачи поиска знаний с привлечением целевого утверждения — предиката цели. БЗ вместе с предикатом цели называется логической программой. БЗ в системе логического

программирования делится на две части: описание фактов и описание логических правил.

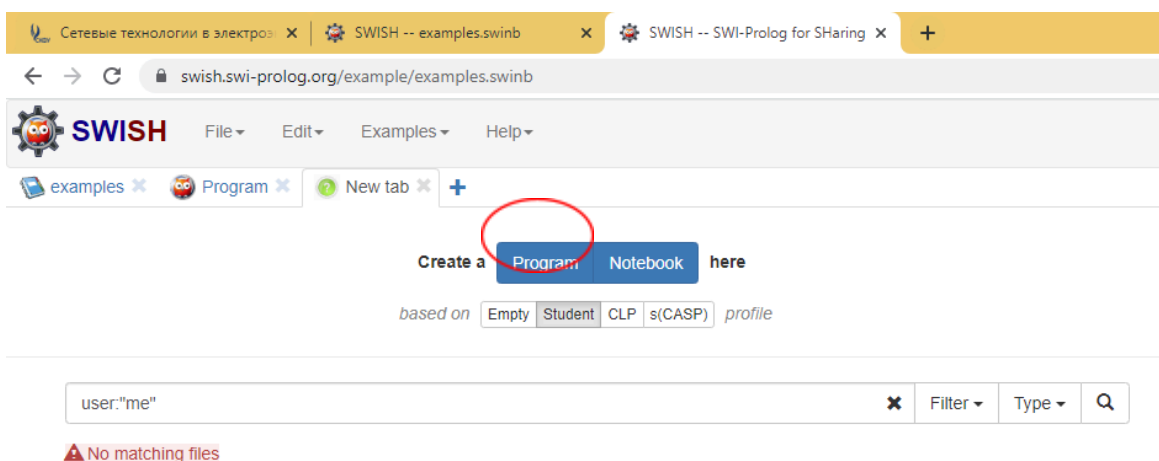
Пример выполнения задания

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

1. Откройте новую страницу:



2. Выберите создание программы

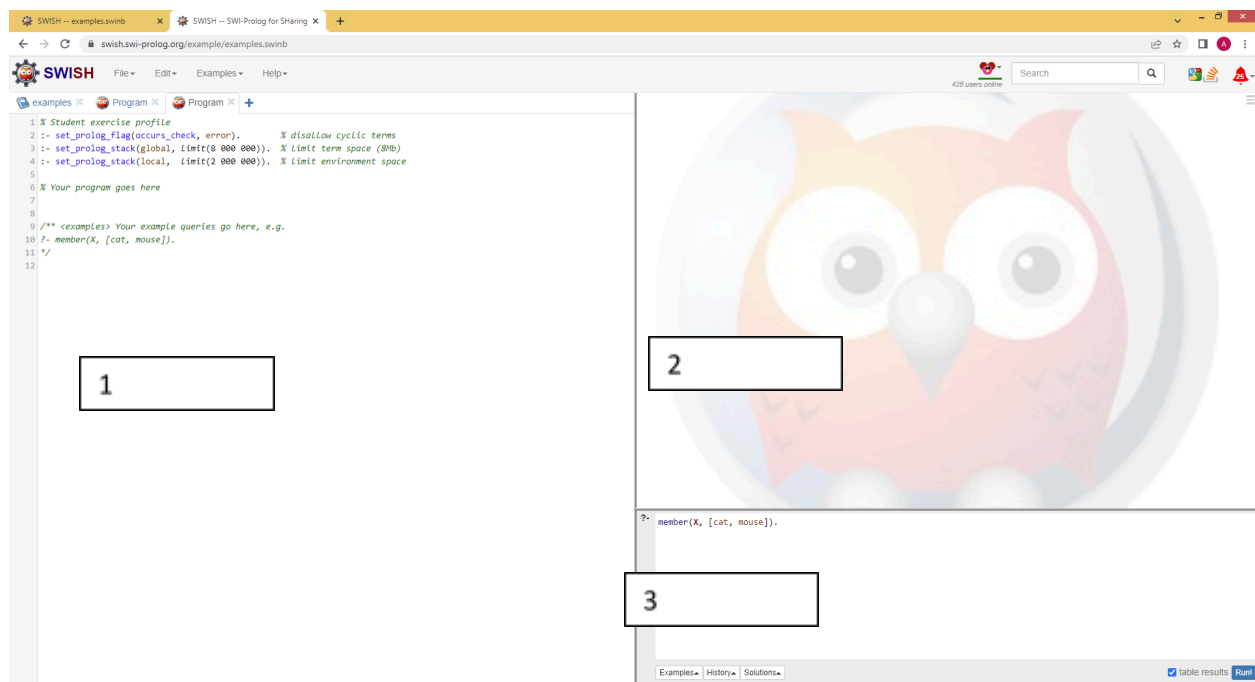


3. В появившемся окне:

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 x SWISH -- SWI-Prolog for SHaring x

swish.swi-prolog.org/example/examples.swinb

SWISH File Edit Examples Help 440 users online Search

examples Program Program +

```
1 % Student exercise profile
2 :- set_prolog_flag(occurs_check, error). % disa
3 :- set_prolog_stack(global, limit(8 000 000)). % limi
4 :- set_prolog_stack(local, limit(2 000 000)). % limi
5
6
7 city('Moscow', 'Russia', 'Europe').
8 city('Tokyo', 'Japan', 'Asia').
9 city('Paris', 'France', 'Europe').
10
11 /*создаем правила */
12 chast_sveta(X, Y) :-
13     city(X,_,Y).
14
15 strana(X,Y):-
16     city(X,Y,_).
17
```

chast_sveta(X, 'Europe')

X	
'Moscow'	1
'Paris'	2

?- chast_sveta(X, 'Europe')

Examples History Solutions table results Run!

SWISH -- examples.swinb x SWISH -- SWI-Prolog for SHaring x

swish.swi-prolog.org/example/examples.swinb

SWISH File Edit Examples Help 446 users online Search

examples Program Program +

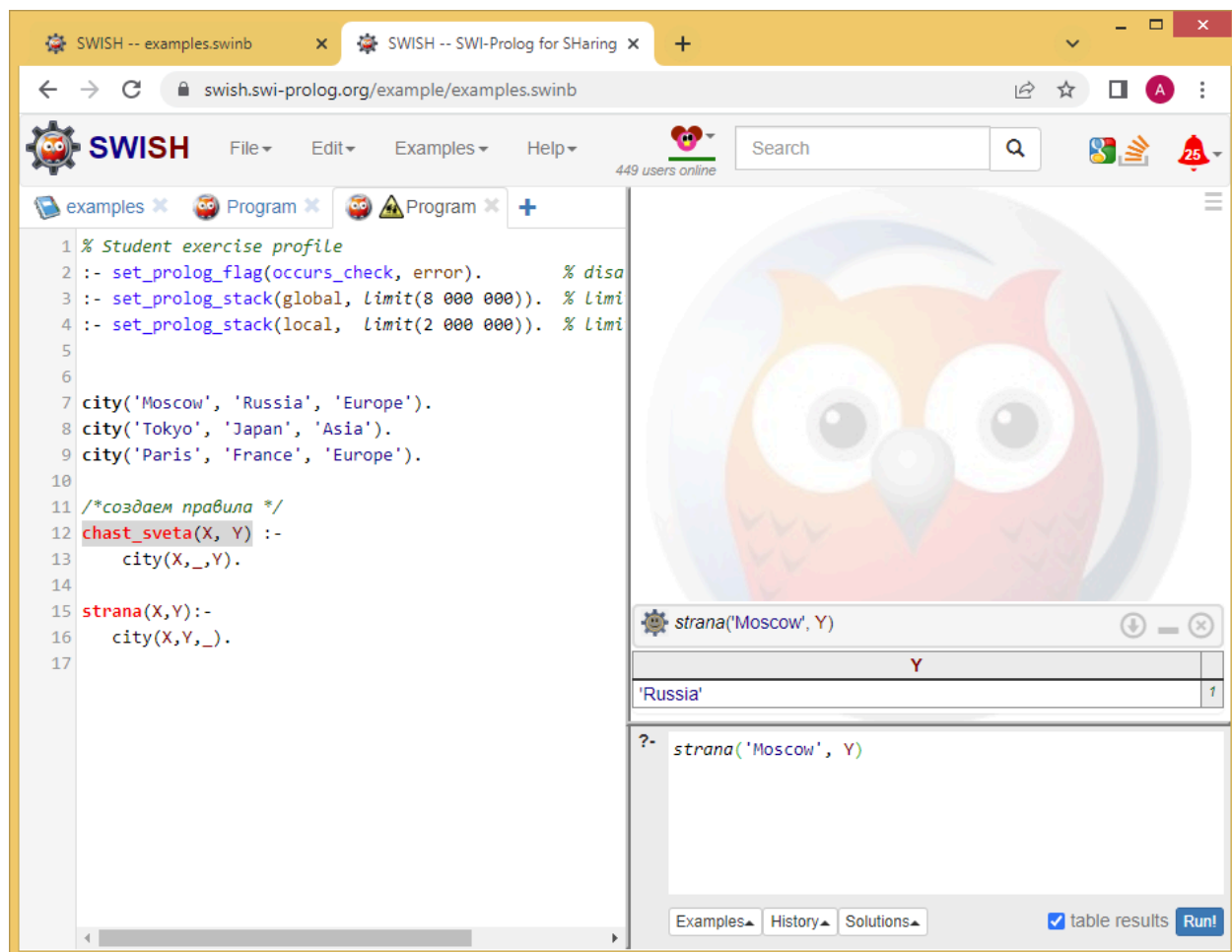
```
1 % Student exercise profile
2 :- set_prolog_flag(occurs_check, error). % disa
3 :- set_prolog_stack(global, limit(8 000 000)). % limi
4 :- set_prolog_stack(local, limit(2 000 000)). % limi
5
6
7 city('Moscow', 'Russia', 'Europe').
8 city('Tokyo', 'Japan', 'Asia').
9 city('Paris', 'France', 'Europe').
10
11 /*создаем правила */
12 chast_sveta(X, Y) :-
13     city(X,_,Y).
14
15 strana(X,Y):-
16     city(X,Y,_).
17
```

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) :-

$X1 = \text{петя, соответствие}(X1, Y1),$
 $X2 = \text{коля, соответствие}(X2, Y2),$
 $X3 = \text{алеша, соответствие}(X3, Y3),$
 $Y1 \setminus = Y2, Y2 \setminus = Y3, Y1 \setminus = Y3.$

Проверяем решение

?-решение($X1, Y1, X2, Y2, X3, Y3$).

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 для выполнения

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.

Защита работы

Перед выполнением практической работы каждый студент получает задание. Защита практической работы происходит только после выполнения задания и оформления отчета по работе. При защите практической работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное задание.

Требования к отчету

Отчет должен содержать иллюстрации с подтверждением выполнения задания, пояснения к иллюстрациям, ответы на контрольные вопросы.

Контрольные вопросы

1. Что представляет собой предикат?
2. Что представляет собой правило?
3. Что представляет собой база знаний?

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное образовательное учреждение
высшего образования**
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

**по организации самостоятельной работы студента
по дисциплине**
«Системы искусственного интеллекта»

Направление подготовки 13.03.02 «Электроэнергетика и
электротехника»
Направленность (профиль):
«Цифровые технологии в электроэнергетике»
Форма обучения: очная

Ставрополь, 2026

СОДЕРЖАНИЕ

Введение	123
1 Общая характеристика самостоятельной работы студента при изучении дисциплины	124
2 План-график выполнения самостоятельной работы	127
3 Методические рекомендации по изучению теоретического материала	128
4 Методические указания по подготовке к практическим работам	136
Список рекомендуемой литературы	137

Введение

Самостоятельная работа является внеаудиторной и предназначена для самостоятельного изучения определенных тем дисциплины по материалам, рекомендованным преподавателем, а также в порядке подготовки к практическим занятиям.

Цель дисциплины «Системы искусственного интеллекта» – формирование согласно ООП ВО набора компетенций будущего бакалавра по направлению 13.03.02 «Электроэнергетика и электротехника».

Для достижения этой цели в процессе изучения дисциплины решаются следующие задачи: ознакомление с основными понятиями и терминами изучаемой дисциплины, которыми будущий специалист будет оперировать в своей практической деятельности.

1 Общая характеристика самостоятельной работы студента при изучении дисциплины

Основная задача высшего образования заключается в формировании творческой личности специалиста, способного к саморазвитию, самообразованию, инновационной деятельности. Решение этой задачи вряд ли возможно только путем передачи знаний в готовом виде от преподавателя к студенту. Для решения этой задачи в учебные планы всех специальностей включена самостоятельная работа, составляющая 50% учебного времени студента.

В широком смысле под самостоятельной работой студентов следует понимать совокупность всей самостоятельной деятельности студентов, как в учебной аудитории, так и вне ее, в контакте с преподавателем и в его отсутствие.

В учебном процессе выделяют два вида самостоятельной работы:

- аудиторная;
- внеаудиторная.

Аудиторная самостоятельная работа по дисциплине выполняется на учебных занятиях под непосредственным руководством преподавателя и по его заданию.

Внеаудиторная самостоятельная работа выполняется студентом по заданию преподавателя, но без его непосредственного участия.

Содержание внеаудиторной самостоятельной работы определяется в соответствии с рекомендуемыми видами заданий согласно примерной и рабочей программ учебной дисциплины. Видами заданий для внеаудиторной самостоятельной работы по дисциплине «Системы искусственного интеллекта» являются:

– для овладения знаниями: чтение текста (учебника, первоисточника, дополнительной литературы), составление плана текста, графическое изображение структуры текста, конспектирование текста, выписки из текста, работа со словарями и справочниками, ознакомление с нормативными

документами, учебно-исследовательская работа, использование аудио- и видеозаписей, компьютерной техники и Интернета и др.;

– для закрепления и систематизации знаний: обработка текста, повторная работа над учебным материалом (учебника, первоисточника, дополнительной литературы, аудио и видеозаписей, составление плана, составление таблиц для систематизации учебного материала, ответ на контрольные вопросы, заполнение рабочей тетради, аналитическая обработка текста (аннотирование, рецензирование, реферирование, конспект-анализ и др.);

– для закрепления навыков – подготовка к лабораторным работам, оформление отчетов по лабораторным работам, выполнение дополнительных заданий, выданных преподавателем.

Самостоятельная работа может осуществляться индивидуально или группами студентов в зависимости от цели, объема, конкретной тематики самостоятельной работы, уровня сложности, уровня умений студентов. Контроль результатов внеаудиторной самостоятельной работы студентов может осуществляться в пределах времени, отведенного на обязательные учебные занятия по дисциплине и внеаудиторную самостоятельную работу студентов по дисциплине, может проходить в письменной, устной или смешанной форме.

Следует с первых дней обучения в вузе приучать себя к целесообразному распределению занятий по месяцам и неделям семестра. При этом очень важна равномерная работа как основное условие успешных занятий в вузе.

Высшей ступенью обучения является приобретение навыков самостоятельной творческой работы, а творческому применению знаний нужно учиться параллельно с их приобретением.

Задачами самостоятельной работы студентов по дисциплине «Системы искусственного интеллекта» являются:

- формирования готовности студентов к поиску, обработке и применению информации для решения профессиональных задач;
- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- развитие познавательных способностей и активности студентов, творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- выработка навыков эффективной самостоятельной профессиональной деятельности.

2 План-график выполнения самостоятельной работы

Общий объём самостоятельной работы в рамках изучения дисциплины «Системы искусственного интеллекта» составляет 72 академических часа.

В структуре самостоятельной работы предусмотрено самостоятельное изучение теоретического материала, подготовка к практическим занятиям.

3 Методические рекомендации по изучению теоретического материала

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. 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. Рекомендуются 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.

1. План - наиболее краткая формой записи. Это перечень вопросов, рассматриваемых в книге или статье. План обычно раскрывает структуру произведения, логику автора, способствует лучшей ориентации в содержании.

Так, составленным планом можно воспользоваться, чтобы вспомнить прочитанное или быстро отыскать в книге нужное место. Представление об основных пунктах плана дает оглавление книги, поэтому во многих случаях наименования глав и разделов можно использовать в качестве пунктов. Составление плана приучает логически мыслить, вырабатывать умение сжато и последовательно излагать суть вопроса в письменной и устной форме.

Существует два способа составления плана: работа над ним по ходу чтения и составление плана после ознакомления с произведением. При этом план получается более последовательным и стройным.

Трудность составления плана состоит в том, что надо выяснить для себя, прежде всего, построение изучаемого текста, ход мыслей автора и лишь затем изложить содержание работы кратко и ясно. При всей своей краткости план дает представление о содержании прочитанного. Следует учесть, что форма плана не исключает цитирования отдельных мет и обобщений. Различают простой и развернутый план. В отличие от простого плана развернутый план не только содержит перечисление вопросов, но и раскрывает основные идей произведения, может включать выдержки из него, схемы, таблицы. Планом, особенно развернутым, необходимо пользоваться при написании выступления или статьи.

В целом развернутый план дает гораздо большее представление о произведении, его основных идеях, задачах, которые в нем решаются. Он может включать положения, замечания, собственные мысли студента.

Важно знать, что составление планов помогает вырабатывать способность к отвлеченному, абстрактному мышлению, но наибольшую

пользу составление плана даст подготовленным лицам, которые бывают достаточно лишь взглянуть на перечень основных вопросов, чтобы воспроизвести содержание прочитанного.

2. Тезисы – более сложная и совершенная форма записи, чем составление плана.

Это сжатое изложение основных мыслей прочитанного произведения или подготовляемого вступления. Особенностью тезисов является их утвердительный характер.

В них сосредотачивается самое главное, только выводы и обобщения, в них меньше доказательств, иллюстрации и пояснений. Тезисы не должны повторять дословно текст, но в ряде мест могут быть близки к нему, воспроизводя некоторые характерные выражения автора, важные для понимания хода его мыслей. Составление тезисов помогает глубже понять основные идеи произведения, выделить главное в нем; приучают сжато, точно и четко сформулировать свои мысли, повышает культуру речи и письма. При составлении тезисов учитывают следующее. Прежде всего, если произведение небольшое, необходимо внимательно изучать его в целом, если большое – изучать по главам и разделам. Затем, когда будут ясны основные идеи, кратко и последовательно излагать их в виде пунктов.

Различают простые и сложные, развернутые тезисы. Если записывают только утверждение чего – либо, такой тезис называют простым, а сложным тезисом будет выражение главной мысли, содержащее, кроме утверждения, еще и краткое ее доказательство.

Часто тезисы формулируются самим автором как выводы и обобщения в заключении книги или разделах книги. Нередко тезисы выделяются в тексте другим шрифтом.

Рекомендуется делать тезисные записи своими словами, причем можно записывать один абзац за другим, учитывая смысловую связь между ними. Но в большинстве случаев следует составлять сводный тезис, сложный по

форме. При этом объединяется несколько утверждений, тесно связанных между собой.

Тезисы по содержанию очень близки к конспекту, но конспект носит более описательный характер, и его положения не столь категоричны, как в тезисах. Кроме того, конспект представляет собой более полную форму записи.

Следует отметить, что различие между формами записей условно, но в любой форме запись – важнейшая часть самостоятельной работы с книгой.

3. Выписки. Это записи текста из книги: теоретических положений, статистических данных, имеющих по мнению читателя важное значение.

Достоинство выписок состоит в точности воспроизведения текста книги, удобстве пользования записями при последующей работе, в накоплении обобщений и фактического материала. Выписки полезны для повторения, освежения в памяти прочитанного, для быстрой мобилизации своих знаний, когда необходимо в короткий срок вспомнить материал. Выписки выделяют из текста самое главное и тем самым помогают глубже понять его. Без них трудно обойтись при подготовке доклада, реферата, выступления. Выписки следует рассматривать как составную часть тезисов и конспектов.

Выписывать текст можно и по ходу чтения и после его завершения. В последнем случае надо замечать места, которые потом будут выписаны. Необходимо каждую выписку снабжать ссылкой на источник с указанием соответствующей страницы. Это нужно, чтобы в последствии можно было быстро найти в книге соответствующее место. Целесообразно выписывать из текста только такие места, в которых содержится самое главное, суть вопроса. Выписки должны быть ориентированы на изучение произведения в целом, а не отдельных мест, поскольку положения, вырванные из общего контакта, понимаются нередко совсем не так, как этого хотел автор. Иначе говоря, отдельно взятые, лишённые пояснений выдержки могут быть не поняты или поняты неправильно.

Выписки бывают дословные (цитаты) и «свободные», когда мысли автора излагаются своими словами. Следует учесть, что большие отрывки, которые трудно цитировать, целесообразнее в краткой форме переложить своими словами, но «яркие» и важные места лучше выписывать дословно. Каждую цитату следует заключать в кавычки. Если ее берут из середины предложения, то после вводных кавычек ставят три точки. Ставят их и в конце цитаты, если из предложения опущены последние слова.

Следует знать, что какого-либо единого метода выписок, годного для всех случаев, не существует, поскольку у каждого человека свои особенности мышления и восприятия, свой подход к теме. Все это влияет на содержание и характер выписок.

Выписки рекомендуется хранить в картотеке, конвертах или папках, на которых следует обозначить общую тему.

4. Аннотация – еще одна форма записи, являющаяся кратким обобщением содержания книги. Ею удобно пользоваться, если имеется намерение вернуться к изучаемому произведению. Аннотация может быть необходима и для того, чтобы не забыть о нем.

Для составления аннотации надо сначала полностью прочитать и глубоко продумать произведение. При всей своей краткости аннотация может содержать отдельные фрагменты авторского текста, а не только оценку книги или статьи.

5. Резюме очень близко к аннотации. Это запись, являющаяся краткой оценкой прочитанного материала. Различие между ними состоит в том, что аннотация сжато характеризует произведение в целом, а резюме концентрирует внимание на его выводах, главных итогах.

6. Конспект – наиболее совершенная и наиболее сложная форма записи. Слово «конспект» происходит от латинского «conspectus», что означает «обзор, изложение». В правильно составленном конспекте обычно выделено самое основное в изучаемом тексте, сосредоточено внимание на наиболее

существенном, в кратких и четких формулировках обобщены важные теоретические положения.

Конспект представляет собой относительно подробное, последовательное изложение содержания прочитанного. На первых порах целесообразно в записях ближе держаться тексту, прибегая зачастую к прямому цитированию автора. В дальнейшем, по мере выработки навыков конспектирования, записи будут носить более свободный и сжатый характер.

Конспект книги обычно ведется в тетради. В самом начале конспекта указывается фамилия автора, полное название произведения, издательство, год и место издания. При цитировании обязательная ссылка на страницу книги. Если цитата взята из собрания сочинений, то необходимо указать соответствующий том. Следует помнить, что четкая ссылка на источник – непереносимое правило конспектирования. Если конспектируется статья, то указывается, где и когда она была напечатана.

Конспект подразделяется на части в соответствии с заранее продуманным планом. Пункты плана записываются в тексте или на полях конспекта. Писать его рекомендуется четко и разборчиво, так как небрежная запись с течением времени становится малопонятной для ее автора. Существует правило: конспект, составленный для себя, должен быть по возможности написан так, чтобы его легко прочитал и кто-либо другой.

Формы конспекта могут быть разными и зависят от его целевого назначения (изучение материала в целом или под определенным углом зрения, подготовка к докладу, выступлению на занятии и т.д.), а также от характера произведения (монография, статья, документ и т.п.). Если речь идет просто об изложении содержания работы, текст конспекта может быть сплошным, с выделением особо важных положений подчеркиванием или различными значками.

В случае, когда не ограничиваются переложением содержания, а фиксируют в конспекте и свои собственные суждения по данному вопросу или дополняют конспект соответствующими материалами их других

источников, следует отводить место для такого рода записей. Рекомендуется разделить страницы тетради пополам по вертикали и в левой части вести конспект произведения, а в правой свои дополнительные записи, совмещая их по содержанию.

Конспектирование в большей мере, чем другие виды записей, помогает вырабатывать навыки правильного изложения в письменной форме важные теоретических и практических вопросов, умение четко их формулировать и ясно излагать своими словами.

Таким образом, составление конспекта требует вдумчивой работы, затраты времени и труда. Зато во время конспектирования приобретаются знания, создается фонд записей.

Конспект может быть текстуальным или тематическим. В текстуальном конспекте сохраняется логика и структура изучаемого произведения, а запись ведется в соответствии с расположением материала в книге. За основу тематического конспекта берется не план произведения, а содержание какой-либо темы или проблемы.

Текстуальный конспект желательно начинать после того, как вся книга прочитана и продумана, но это, к сожалению, не всегда возможно. В первую очередь необходимо составить план произведения письменно или мысленно, поскольку в соответствии с этим планом строится дальнейшая работа. Конспект включает в себя тезисы, которые составляют его основу. Но, в отличие от тезисов, конспект содержит краткую запись не только выводов, но и доказательств, вплоть до фактического материала. Иначе говоря, конспект – это расширенные тезисы, дополненные рассуждениями и доказательствами, мыслями и соображениями составителя записи.

Как правило, конспект включает в себя и выписки, но в него могут войти отдельные места, цитируемые дословно, а также факты, примеры, цифры, таблицы и схемы, взятые из книги. Следует помнить, что работа над конспектом только тогда будет творческой, когда она не ограничена текстом

изучаемого произведения. Нужно дополнять конспект данными из другими источников.

В конспекте необходимо выделять отдельные места текста в зависимости от их значимости. Можно пользоваться различными способами: подчеркиваниями, вопросительными и восклицательными знаками, репликами, краткими оценками, писать на полях своих конспектов слова: «важно», «очень важно», «верно», «характерно».

В конспект могут помещаться диаграммы, схемы, таблицы, которые придадут ему наглядность.

Составлению тематического конспекта предшествует тщательное изучение всей литературы, подобранной для раскрытия данной темы. Бывает, что какая-либо тема рассматривается в нескольких главах или в разных местах книги. А в конспекте весь материал, относящийся к теме, будет сосредоточен в одном месте. В плане конспекта рекомендуется делать пометки, к каким источникам (вплоть до страницы) придется обратиться для раскрытия вопросов. Тематический конспект составляется обычно для того, чтобы глубже изучить определенный вопрос, подготовиться к докладу, лекции или выступлению на семинарском занятии. Такой конспект по содержанию приближается к реферату, докладу по избранной теме, особенно если включает и собственный вклад в изучение проблемы.

4 Методические указания по подготовке к практическим работам

В рамках самостоятельной работы предусмотрена подготовка студента к практическим занятиям. Практические работы позволяют интегрировать теоретические знания и формировать практические умения и навыки студентов в процессе учебной деятельности.

Цели практических занятий по дисциплине «Системы искусственного интеллекта»:

1) закрепление теоретического материала путем систематического контроля за самостоятельной работой студентов;

2) формирование умений использования теоретических знаний в процессе выполнения практических работ;

3) развитие аналитического мышления путем обобщения результатов практических работ;

4) формирование навыков оформления результатов практических работ в виде таблиц, графиков, выводов.

На практических занятиях осуществляются следующие формы работ со студентами: индивидуальная (оценка знаний, выполненных тестовых заданий, проверка рабочих тетрадей); групповая (выполнение заданий малыми группами по 2-4 человека); фронтальная (подведение итогов выполнения лабораторных работ).

Список рекомендуемой литературы

Основная литература:

1. Остроух, А. В. Системы искусственного интеллекта : монография / А. В. Остроух, Н. Е. Суркова. — 4-е изд., стер. — Санкт-Петербург : Лань, 2024. — 228 с. — ISBN 978-5-507-47478-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/379988>
2. Макшанов, А. В. Системы поддержки принятия решений : учебное пособие для вузов / А. В. Макшанов, А. Е. Журавлев, Л. Н. Тындыкарь. — 2-е изд., стер. — Санкт-Петербург : Лань, 2021. — 108 с. — ISBN 978-5-8114-8489-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/176903>
3. Москвитин, А. А. Данные, информация, знания: методология, теория, технологии : монография / А. А. Москвитин. — Санкт-Петербург : Лань, 2022. — 236 с. — ISBN 978-5-8114-3232-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/206267>
4. Романов, П. С. Системы искусственного интеллекта. Моделирование нейрон-ных сетей в системе MATLAB. Лабораторный практикум / П. С. Романов, И. П. Романо-ва. — 2-е изд., стер. — Санкт-Петербург : Лань, 2022. — 140 с. — ISBN 978-5-8114-9991-5. — Текст : электронный // Лань : электронно-библиотечная система.

Дополнительная литература:

1. Павлова, А. И. Искусственные нейронные сети : учебное пособие / А. И. Павло-ва. — Москва : Ай Пи Ар Медиа, 2021. — 190 с. — ISBN 978-5-4497-1165-6. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]
2. Сысоев, Д. В. Введение в теорию искусственного интеллекта : учебное пособие / Д. В. Сысоев, О. В. Курипта, Д. К. Проскурин. — Москва : Ай Пи Ар Медиа, 2021. — 170 с. — ISBN 978-5-4497-1092-5. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт].

3. Советов, Б. Я. Информационные технологии: теоретические основы : учебное пособие / Б. Я. Советов, В. В. Цехановский. — 2-е изд., стер. — Санкт-Петербург : Лань, 2022. — 444 с. — ISBN 978-5-8114-1912-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/209876>

4. Сотник, С. Л. Проектирование систем искусственного интеллекта : учебное пособие / С. Л. Сотник. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 228 с. — ISBN 978-5-4497-0868-7. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]

Учебно-методическая литература:

1. Методические указания к практическим работам по дисциплине «Системы искусственного интеллекта», СКФУ, 2025.

Интернет-ресурсы:

1. <http://catalog.ncfu.ru> – электронные каталоги Ассоциации электронных библиотек учебных заведений и организаций СКФО.

2. <http://www.biblioclub.ru> – электронно-библиотечная система «Университетская библиотека онлайн».

3. <http://www.iprbookshop.ru/> – ЭБС «IPRbooks».