критерий Ханнана-КуинаФедеральное государственное бюджетное
образовательное учреждение высшего образования
«КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»

А. В. Снытников

Е. Ю. Заболотнова

ВЫСОКОУРОВНЕВЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие по выполнению лабораторных работ
для студентов бакалавриата по направлениям подготовки
09.03.01 Информатика и вычислительная техника,
09.03.03 Прикладная информатика

Калининград
Издательство ФГБОУ ВО» КГТУ
2024

УДК 004.4(075)

Рецензент
кандидат экономических наук, зав. кафедрой прикладной информатики ФГБОУ ВО «Калининградский государственный технический университет» М. В. Соловей

Снытников, А. В.
Высокоуровневые технологии программирования: учебно-методическое пособие по выполнению лабораторных работ для студентов направлений подготовки 09.03.01 Информатика и вычислительная техника, 09.03.03 Прикладная информатика / А. В. Снытников, Е. Ю. Заболотнова. – Калининград: Изд-во ФГБОУ ВО «КГТУ», 2024. – 56 с.

Данное учебно-методическое пособие содержит лабораторные работы по дисциплине: задания, методические указания по выполнению работы, структуру отчета и требования к его оформлению, приведены контрольные вопросы и порядок зашиты лабораторной работы.
Табл. 1, рис. 12

Учебно-методическое пособие рассмотрено и одобрено в качестве локального электронного методического материала кафедрой прикладной информатики института цифровых технологий ФГБОУ ВО «Калининградский государственный технический университет» 19 сентября 2023 г., протокол № 3

Учебно-методическое пособие рекомендовано к использованию в учебном процессе в качестве локального электронного методического материала методической комиссией института цифровых технологий 30 сентября 2024 г., протокол № 6

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Данное учебно-методическое пособие содержит описание лабораторных работ по дисциплине «Высокоуровневые технологии программирования»: задания, методические указания по выполнению работы, структуру отчета и требования к его оформлению. Приведены контрольные вопросы и порядок зашиты лабораторных работ.

Целью лабораторного практикума по дисциплине является развитие у студентов практических навыков разработки программного обеспечения, что позволит закрепить, расширить и углубить знания, полученные в теоретическом курсе. Отчеты по работам оформляются в электронном виде. При выполнении работы в Colab-ноутбуке чередуйте блоки текста и кода. В текстовых блоках указывайте номер и содержание задания, а в следующем за ним блоке кода приведите его программную реализацию. В начале файла в текстовом блоке укажите группу и фамилию студента, а также информацию о номере и теме лабораторной работы. Защита работы проходит в виде беседы с преподавателем по ходу выполнения работы и контрольным вопросам.

В данном учебно-методическом пособии приведены лабораторные работы для студентов направления подготовки 09.03.01 Информатика и вычислительная техника и 09.03.03 Прикладная информатика.

ЛАБОРАТОРНАЯ РАБОТА № 1. РАБОТА С ДАННЫМИ С ПОМОЩЬЮ БИБЛИОТЕКИ PANDAS

Цель работы

Знакомство с библиотекой Pandas. Создание и редактирование датафрейма. Основные операции с датафреймами. Методы библиотеки Pandas для работы с датафреймами.

Теоретическая часть

Pandas – это библиотека Python для обработки и анализа структурированных данных, её название происходит от «panel data» («панельные данные»). Панельными данными называют информацию, полученную в результате исследований и структурированную в виде таблиц. Для работы с такими массивами данных и создан Pandas.

С помощью Pandas специалисты могут группировать и визуализировать данные, создавать сводные таблицы и делать выборку по определенным признакам.

Pandas построен на основе двух основных пакетов Python: NumPy и Matplotlib. Numpy предоставляет объекты типа «многомерный массив» для простой обработки данных, а Matplotlib обладает мощными возможностями визуализации данных, которыми Pandas пользуется.

Pandas DataFrame – это двумерный массив, похожий на таблицу/лист Excel (кстати, данные из Excel можно читать с помощью команды pandas.read_excel('file.xls')). В нем можно проводить такие же манипуляции с данными: объединять в группы, сортировать по определенному признаку, производить вычисления. Как любая таблица, датафрейм состоит из столбцов и строк, причем столбцами будут объекты – Series.

Структура DataFrame отлично подходит для представления реальных данных: строки соответствуют признаковым описаниям отдельных объектов, а столбцы соответствуют признакам. Объект DataFrame имеет два индекса: по строкам и по столбцам. Если индекс по строкам явно не задан (например,

колонка по которой нужно их строить), то pandas задаёт целочисленный индекс RangeIndex от 0 до N-1, где N это количество строк в таблице.

Для подключения библиотеки используем инструкцию *import pandas as pd*

Создание датафреймов

Способ 1. Из словаря.

dict_dogs ={‘name’:[‘Bella’, ‘Chalie’, ‘Lucy’,’Cooper’,’Max’ ,’Stella’ ,’Bernie’],

’color’:[‘Brown’,’Black’,’Brown’,’Gray’,’Black’,’Tan’,’While’],

’breed’:[‘Labrador’,’Poodle’,’Chow Chow’, ’Schnauzer’, ’Labrador’,’Chihuahua’, ’St.Bernard’],

’height_cm’:[56,43,46,49,59,18,77],

’weight_kd’:[24,24,24,17,29,2,74],

’date_of_birth’:[‘2013-07-01’,’2016-09-16’,’2014-08-25’,’2011-12-11’,’2017-01-20’,’2-15-04-20’,’2018-02-27’]}

df=pd.DataFrame(dict_dogs**)**

Способ 2. Из списка.

list_dogs=[[‘Bella’, ‘Chalie’, ‘Lucy’,’Cooper’,’Max’ ,’Stella’ ,’Bernie’],

[‘Brown’,’Black’,’Brown’,’Gray’,’Black’,’Tan’,’While’],

[‘Labrador’,’Poodle’,’Chow Chow’, ’Schnauzer’, ’Labrador’,’Chihuahua’ ’St.Bernard’], [56,43,46,49,59,18,77], [24,24,24,17,29,2,74],

[2013-07-01’,’2016-09-16’,’2014-08-25’,’2011-12-11’,’2017-01-20’,’2-15-04-20’,’2018-02-27’]]

df=pd.DataFrame(list_dogs)

pd.DataFrame().T

При этом каждый список будет представлять из себя строку, чтобы представить списки в виде столбцов транспонируем полученный датафрейм.

Способ 3. Из файла.

Если данные сохранены в файле, то создание датафрейма на его основе:

df=pd.read_csv(‘D:/Primer/dogs.csv’)

Атрибуты датафрейма:

Атрибут shape выводит в виде кортежа пару чисел – число строк и число столбцов – *db.shape;*

df.values – атрибут values содержит значения датафрейма в виде двумерного списка (по строкам);

df. columns – атрибут columns содержит имена столбцов;

df.index – атрибут index содержит номера строк или имена строк.

Методы для работы с дататфреймами:

df.head() – выдает пять первых столбцов таблицы;

df.head(15) – выдает заданное количество (в примере -15) первых строк таблицы;

df.tail() – выдает заданное количество последних строк таблицы;

df.info() – чтобы посмотреть общую информацию по датафрейму и всем признакам, воспользуемся методом info;

df.describe() – метод describe показывает основные статистические характеристики данных по каждому числовому признаку: число непропущенных значений, среднее, стандартное отклонение, диапазон, медиану, 0.25 и 0.75 квартили;

df.describe(include=’all’) – выдает информацию не только по числовым столбцам, а по всем;

df['название столбца'].head(или df.название_столбца.head() – получение данных из столбца с указанным названием, первые 5 строк.

С помощью методов loc и iloc – можно из начального датафрейма зафиксировать определённые интервал строк и интересующих столбцов и работать/смотреть только их.

df.loc[1:5, ['name']] или df.name.loc[1:5] – выводим строки с номерами из указанного интервала из столбца с заданным названием. Имена столбцов можно задавать названиями столбцов или их номерами.

Посмотрим на наш датафрейм, проверив данные на соответствие какому-то условию:

df[(df['female'] == 1) & (df['black'] == 1)].head(10)

Методы *loc* и *iloc* позволяют выделять из исходной таблицы строки и столбцы. loc выбирает строки и столбцы с определенными метками, а iloc выбирает строки и столбцы в определенных целочисленных позициях. Данные методы могут работать как срезы, при этом указывается начальное и конечное значение интервала и шаг при необходимости.

Метод *sort_values* применяется для сортировки датафрейма и выглядит следующим образом с указанием всех параметров:

DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last', ignore_index=False, key=None).

by – определяет список столбцов для сортировки;

ascending – задает порядок сортировки;

sort_values по умолчанию сортирует по возрастанию.

Чтобы задать порядок «по убыванию», следует указать ascending=False.

Методы для обработки числовых данных.

Одной из наиболее распространенных характеристик сводных статистических числовых данных является среднее значение. Вы можете вычислить среднее значение столбца, выбрав столбец и вызвав функцию mean(). Существует множество других сводных статистических данных, которые вы можете вычислить по столбцам, например, медиана и мода, минимум (min()) и максимум (max()), а также дисперсия и стандартное отклонение.

Задание для выполнения:

1. Создайте любым из приведенных выше способов датафрейм о собаках.

	name	breed	color	height_cm	weight_kg	date_of_birth
0	Bella	Labrador	Brown	56	24	2013-07-01
1	Charlie	Poodle	Black	43	24	2016-09-16
2	Lucy	Chow Chow	Brown	46	24	2014-08-25
3	Cooper	Schnauzer	Gray	49	17	2011-12-11
4	Max	Labrador	Black	59	29	2017-01-20
5	Stella	Chihuahua	Tan	18	2	2015-04-20
6	Bernie	St. Bernard	White	77	74	2018-02-27

2. Выведите на экран первые три строки (head) и последние три строки таблицы (tail).

3. С помощью метода info отобразите имена столбцов, содержащиеся в них типы данных и наличие пропущенных значений.

4. С помощью метода описания датафрейма (describe) вычислите сводную статистику для числовых столбцов.

5. Используя соответствующие атрибуты датафрейма, выведите его в виде матрицы (таблицы), выведите количество строк и столбцов, а также названия столбцов таблицы о собаках.

6. Для столбцов с числовыми значениями определите максимальное и минимальное значение.

7. Используя нижеприведенную функцию pct30, которая вычисляет тридцатый процентиль столбца, найти тридцатый процентиль веса собак.

```
def pct30(column):
    return column.quantile(0.3)
```

8. Создайте новую таблицу df1, куда занесите только собак коричневого окраса.

9. Создайте новую таблицу df2, куда занесите только собак с именами Белла и Стелла.

10. Отсортируйте данные в исходной таблице в порядке убывания по столбцу дата рождения.

Индивидуальное задание

Не предусмотрено.

Методические указания по выполнению работы

Сначала познакомьтесь с теоретической частью лабораторной работы и материалами лекции по данной теме. Ниже приведены ссылки на сайты с материалами по данной тематике. Выполните работу в черновом варианте. При этом рекомендуется для каждой части задания использовать отдельный блок кода, а затем оформите отчет, добавив комментарии или текстовые блоки.

Список источников

1. Аналитикам: большая шпаргалка по Pandas [Электронный ресурс]. – Режим доступа: https://smysl.io/blog/pandas/ : (дата обращения: 05.01.2023).

2. Структуры данных в pandas [Электронный ресурс]. – Режим доступа: https://pythonru.com/biblioteki/struktury-dannyh-v-pandas (дата обращения: 05.01.2023).

Контрольные вопросы:

1. Что такое квантиль?

2. Что такое процентиль?

3. Как устроен CSV-файл?

4. Как обеспечить доступ к элементу данных датафрейма df в колонке с именем AAA и в строке с номером 8?

5. Как заменить содержание всей колонки?

6. Как вывести список названий все колонок датафрейма?

7. Как изменить разделитель целой и дробной части (запятую на точку, или наоборот)?

8. Как избавится от нечисловых значений в датафрейме?

9. Можно ли открыть датафрейм в Microsoft Excel?

ЛАБОРАТОРНАЯ РАБОТА № 2. ЛИНЕЙНАЯ РЕГРЕССИЯ

Цель работы:

Познакомиться с линейной регрессией с двумя переменными.

Теоретическая часть

Линейная регрессия – выраженная в виде прямой зависимость среднего значения какой-либо величины от некоторой другой величины. В отличие от функциональной зависимости $y = f(x)$, когда каждому значению независимой переменной x соответствует одно определённое значение величины y, при линейной регрессии одному и тому же значению x могут соответствовать в зависимости от случая различные значения величины y.

	Признак 1	Признак 2	Признак 3	Признак 4	Признак 5	
	Площадь	Удаленность от центра	Год постройки	Лет после ремонта	Тип постройки	Стоимость квартиры > 100k $
1	25	3	2005	1	1	65
2	55	10	1987	5	2	120
3	50	12	1990	6	5	125
	k_1	k_2	k_3	k_4	k_5	y

Линейная регрессия представляет ответ (значение целевой переменной) как взвешенную сумму значений признаков:

$$\hat{y} = k_0 + \sum x_i k_i = k_0 + k_1 x_1 + k_2 x_2 + k_3 x_3 + k_4 x_4 + k_5 x_5$$

x_i — значение i-го признака элемента

Рисунок 2.1 – Пример использования линейной регрессии для оценки стоимости квартиры

Если в результате наблюдения установлено, что при каждом определённом значении x существует сколько-то (n) значений переменной y, то зависимость средних арифметических значений y от x и является регрессией в статистическом понимании. Обычно парная регрессия используется в том случае, когда из всего круга факторов, влияющих на результат, можно выделить один фактор, оказывающий наиболее сильное влияние. Он и используется в качестве объясняющей переменной.

Математическое определение:

Пусть, есть два ряда данных:

$X = x_1, x_2, \ldots, x_n$

$Y = y_1, y_2, \ldots, y_n$

где n – число наблюдений в парной линейной регрессии связь между переменными определяется следующим образом:

$y = \hat{y}(x) + \varepsilon = a + b \cdot x + \varepsilon;$

y – зависимая (объясняемая) переменная;

x – независимая переменная;

$\hat{y}(x)$ – зависимая переменная, рассчитанная с помощью уравнения регрессии;

также $\hat{y}(x)$ – теоретическое значение (в данном случае оно рассчитывается по линейному уравнению регрессии);

a, b – константы, параметры уравнения линейной регрессии;

ε – случайная компонента, или возмущение.

Уравнение простой регрессии характеризует связь между двумя переменными, которая проявляется как некоторая закономерность лишь в среднем или в целом по совокупности наблюдаемых данных.

Основная идея состоит в том, чтобы получить линию, которая наилучшим образом соответствует данным. Линия наилучшего соответствия – это та, для которой общая ошибка прогноза (все точки данных) как можно меньше. Ошибка – это расстояние между точкой и линией регрессии.

Рисунок 2.2 – Геометрический смысл линейной регрессии на примере зависимости двух признаков

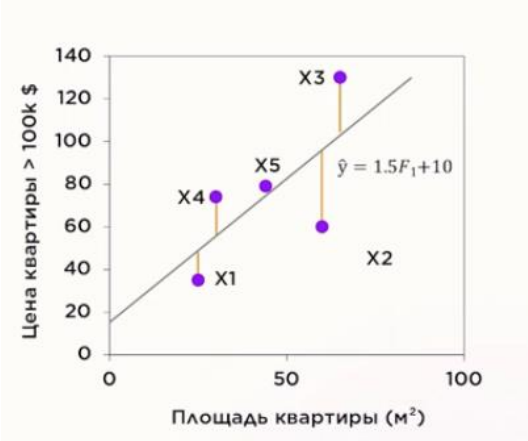


Рисунок 2 3 – Графическое представление ошибок при заданных значениях коэффициентов

Используя обучающие данные, получим линию регрессии, которая даст минимальную ошибку. Затем это линейное уравнение используется для любых новых данных. То есть, если мы даем количество часов, отработанных студентом в качестве входных данных, наша модель должна предсказать его оценку с минимальной ошибкой.

$$Y(pred) = b0 + b1*x.$$

Значения b0 и b1 необходимо выбирать так, чтобы они минимизировали ошибку. Если в качестве метрики для оценки модели используется сумма квадратов ошибок, то цель состоит в том, чтобы получить линию, которая наилучшим образом уменьшает ошибку:

$$\eta = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

$$b_0 = \overline{y} - b_1 \overline{x}$$

$$b_1 = \frac{\sum_{i=1}^{N} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{N} (x_i - \overline{x})^2}$$

Самым простым способом определения вида связи, является построение поля корреляции. Каждую пару наблюдений (xi, yi) можно отобразить в качестве точки на плоскости XY. В этом случае наилучшей функцией считается та, чей график проходит через наибольшее количество точек или как можно ближе к ним. В каждом из наблюдений величину случайной компоненты можно определить, как разность между фактическим значением результата и рассчитанным по уравнению регрессии (см. рисунок 2.3)

$$\varepsilon i = yi - \hat{}yi.$$

Выполнение аппроксимации кривой в Python

Подгонку кривой можно выполнить для набора данных с помощью Python. Python предоставляет библиотеку с открытым исходным кодом, известную как пакет SciPy. Этот пакет включает функцию, известную как функция curve_fit(), которая используется для подгонки кривой с помощью нелинейных наименьших квадратов. Функция curve_fit() принимает те же входные и выходные данные, что и параметры, в дополнение к имени целевой функции, которую нужно использовать. Целевая функция должна включать примеры входных данных и небольшое количество параметров. Оставшиеся параметры станут коэффициентами или весовыми константами, которые

оптимизируют процесс нелинейной оптимизации методом наименьших квадратов. Подобрать данные с помощью функции curve_fit() довольно просто, она предоставляет функцию сопоставления, данные x и y соответственно. Метод curve_fit() возвращает оптимальные аргументы и рассчитанные значения ковариации в качестве выходных данных.

Функция scipy.optimize.curve_fit используется для подгонки кривой к данным, минимизируя сумму квадратов отклонений между данными и моделью. Общий вид функции выглядит следующим образом:

*scipy.optimize.curve_fit(f, xdata, ydata, p0=None, sigma=None, absolute_sigma=False, check_finite=True, bounds=(-inf, inf), method=None, jac=None, **kwargs)*

Параметры функции:

f: Функция модели, которая должна быть подогнана к данным. Она должна принимать первым аргументом массив независимых переменных и возвращать массив зависимых переменных.

xdata: Массив независимых переменных.

ydata: Массив зависимых переменных (наблюдаемых данных).

p0: Начальные значения параметров модели. Если не указано, начальные значения будут определены автоматически.

sigma: Массив ошибок (стандартных отклонений) для ydata. Если указано, то минимизация будет выполняться с учетом весов.

absolute_sigma: Логическое значение, указывающее, являются ли значения sigma абсолютными. Если False, то sigma интерпретируется как относительные ошибки.

check_finite: Логическое значение, указывающее, следует ли проверять, что все значения в xdata и ydata конечны.

bounds: Границы для параметров модели. Должен быть кортеж из двух массивов, указывающих нижние и верхние границы для каждого параметра.

method: Метод оптимизации. По умолчанию используется метод Левенберга-Марквардта.

jac: Якобиан функции модели. Если не указано, якобиан будет вычислен численно.

**kwargs: Дополнительные аргументы, передаваемые в функцию оптимизации.

Функция возвращает два значения:

popt: Оптимальные значения параметров модели.

pcov: Ковариационная матрица оценок параметров.

Задание для выполнения

У нас есть набор данных, который содержит информацию о взаимосвязи между некоторыми числовыми последовательностями. Это будут наши тренировочные данные. Цель работы состоит в том, чтобы разработать модель, которая может предсказывать значения по заданному набору исходных данных. Выполните пример для реализации линейной регрессии всех файлов к заданию.

Пример программы на Python для реализации линейной регрессии:

```
import numpy as np
import matplotlib.pyplot as plt  # To visualize
import pandas as pd  # To read data
from sklearn.linear_model import LinearRegression
data = pd.read_csv('data.csv')  # load data set
X = data.iloc[:, 0].values.reshape(-1, 1)  # values converts it into a numpy array
Y = data.iloc[:, 1].values.reshape(-1, 1)  # -1 means that calculate the dimension
linear_regressor = LinearRegression()  # create object for the class
linear_regressor.fit(X, Y)  # perform linear regression
Y_pred = linear_regressor.predict(X)  # make predictions
```

Затем нам нужно описать функцию отображения, чтобы сопоставить независимую переменную и зависимую. Функция отображения должна быть реализована как функция Python, которая принимает входные данные и аргументы. Это может быть прямая линия, и в этом случае она будет выглядеть следующим образом:

```
# objective function
def objective(x, a, b, c):
    return a * x + b
```

Затем мы можем вызвать функцию curve_fit(), чтобы подобрать прямую линию к набору данных, используя нашу определенную функцию. Функция curve_fit() возвращает оптимальные значения для функции отображения, например, значения коэффициентов. Он также возвращает ковариационную матрицу для оценочных параметров, но пока мы можем ее игнорировать.

```
# fit curve
popt, _ = curve_fit(objective, x_values, y_values)
```

Если на графике все точки (xi, yi) совпадают с линией регрессии, тогда между результативным признаком Y и фактором X существует строгая функциональная связь и выполняется следующее равенство:

$\varepsilon i = 0$ для каждого $i = 1, 2, \ldots, n$.

Хорошо подобранная модель линейной регрессии обладает рядом характеристик, которые указывают на её адекватность и надежность. Вот основные из них:

1. Высокий коэффициент детерминации (R^2): R^2 показывает, какую долю дисперсии зависимой переменной объясняет модель.

Интерпретация R^2

- $R^2=0$: Модель не объясняет никакой части дисперсии зависимой переменной;
- $R^2=1$: Модель объясняет всю дисперсию зависимой переменной;
- $0<R^2<1$: Модель объясняет часть дисперсии зависимой переменной.

Значение R^2 близкое к 1 указывает на то, что модель хорошо объясняет данные. Коэффициент детерминации является важным показателем качества модели регрессии, но он не всегда является единственным критерием. Важно также учитывать другие метрики и диагностические тесты для полной оценки модели.

2. Низкие значения ошибок (MSE, RMSE, MAE): Среднеквадратичная ошибка (MSE), корень из среднеквадратичной ошибки (RMSE) и средняя абсолютная ошибка (MAE) должны быть минимальными. Это указывает на то, что предсказания модели близки к наблюдаемым данным.

Индивидуальное задание

Не предусмотрено.

Методические указания по выполнению работы

1. Загрузите файл с исходными данными
2. Выберите числовые поля для анализа, определите целевой признак
3. Очистите таблицу от пропусков данных (функции dropna() или fillna())
4. Выполните предложенную программу и проанализируйте результат.

Список источников

1. Подгонка кривой в Python с помощью библиотеки SciPy [Электронный ресурс]. – Режим доступа https://pythonpip.ru/examples/podgonka-krivoy-v-python-s-pomoschyu-biblioteki-scipy : (дата обращения: 05.01.2023).

2. Линейная регрессия на Python: объясняем на пальцах [Электронный ресурс]. – Режим доступа: https://proglib.io/p/linear-regression/ (дата обращения: 05.01.2023).

3. Полное руководство по линейной регрессии в Python [Электронный ресурс]. – Режим доступа: https://www.codecamp.ru/blog/linear-regression-python// (дата обращения: 05.01.2023).

Контрольные вопросы:

1. Назовите необходимое и достаточное условие применимости линейной регрессии.
2. Что входит в понятие построения линейной регрессии (что выполняет функция fit)?
3. Как вывести коэффициент и свободный член линейной регрессии для построенной модели?
4. В каком интервале значений независимой переменной применимы предсказания, полученные на основе линейной регрессии?
5. Как вычисляется ошибка аппроксимации?
6. Каковы характеристики хорошо подобранной модели?

ЛАБОРАТОРНАЯ РАБОТА № 3. МНОЖЕСТВЕННАЯ ЛИНЕЙНАЯ РЕГРЕССИЯ

Теоретическая часть

Множественная регрессия похожа на линейную регрессию, но с более чем одним независимым значением, что означает, что мы пытаемся предсказать значение на основе двух или более переменных. Взгляните на набор данных ниже, он содержит некоторую информацию об автомобилях (таблица 3.1).

Таблица 3.1 Исходные данные для множественной регрессии, сведения об автомобилях

Car	Model	Volume	Weigl	CO_2
Toyota	Aygo	1000	790	99
Mitsubishi	Space Star	1200	1160	95
Skoda	Citigo	1000	929	95
Fiat	500	900	865	90
Mini	Cooper	1500	1140	105
VW	Up!	1000	929	105
Skoda	Fabia	1400	1109	90
Mercedes	A-Class	1500	1365	92
Ford	Fiesta	1500	1112	98
Audi	A1	1600	1150	99
Hyundai	I20	1100	980	99
Suzuki	Swift	1300	990	101
Ford	Fiesta	1000	1112	99
Honda	Civic	1600	1252	94

Hundai	I30	1600	1326	97
Opel	Astra	1600	1330	97
BMW	1	1600	1365	99
Mazda	3	2200	1280	104
Skoda	Rapid	1600	1119	104
Ford	Focus	2000	1328	105
Ford	Mondeo	1600	1584	94
Opel	Insignia	2000	1428	99
Mercedes	C-Class	2100	1365	99
Skoda	Octavia	1600	1415	99
Volvo	S60	2000	1415	99
Mercedes	CLA	1500	1465	102
Audi	A4	2000	1490	104
Audi	A6	2000	1725	114
Volvo	V70	1600	1523	109
BMW	5	2000	1705	114
Mercedes	E-Class	2100	1605	115
Volvo	XC70	2000	1746	117
Ford	B-Max	1600	1235	104
BMW	2	1600	1390	108
Opel	Zafira	1600	1405	109
Mercedes	SLK	2500	1395	120

Мы можем предсказать выброс CO_2 автомобиля на основе размера двигателя, но с помощью множественной регрессии мы можем добавить больше переменных, таких как вес автомобиля, чтобы сделать прогноз более точным.

В Python у нас есть модули, которые сделают работу за нас. Начните с импорта модуля Pandas.

```
import pandas
```

Подключите файл, подготовленный для тестирования:

```
df = pandas.read_csv("cars.csv")
```

Затем составьте список независимых значений и определите, что будем считать за переменную x, поместите зависимые значения в переменную с именем y.

```
X = df[['Weight', 'Volume']]
y = df['CO2']
```

Совет: список независимых значений принято называть прописной буквой X, а список зависимых значений строчной буквой y.

Мы будем использовать некоторые методы из модуля sklearn, поэтому нам также придется импортировать этот модуль:

```
from sklearn import linear_model
```

Из модуля sklearn мы будем использовать метод LinearRegression() для создания объекта линейной регрессии.

Этот объект имеет метод fit(), который принимает независимые и зависимые значения в качестве параметров и заполняет объект регрессии данными, описывающими связь:

```
regr = linear_model.LinearRegression()
regr.fit(X, y)
#predict the CO2 emission of a car where the weight is 2300kg, and
the volume is 1300cm³:
predictedCO2 = regr.predict([[2300, 1300]])
```

Задание для выполнения

Выполните приведенные ниже примеры:

```
import pandas
from sklearn import linear_model

df = pandas.read_csv("cars.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

regr = linear_model.LinearRegression()
regr.fit(X, y)

#predict the CO2 emission of a car where the weight is 2300kg, and the
volume is 1300cm³:
predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)
```

Result:

```
[107.2087328]
```

Рисунок 3. 1 – Пример выполнения программы 1

Мы предсказали, что автомобиль с 1,3-литровым двигателем и весом 2300 кг будет выделять примерно 107 граммов CO2 на каждый километр, который он проезжает.

Коэффициент

Коэффициент – это фактор, который описывает связь с неизвестной переменной.

Пример: если x – переменная, то 2x – это x два раза, x – неизвестная переменная, а число 2 – коэффициент.

В этом случае мы можем запросить значение коэффициента веса против CO2 и для объема против CO2. Ответ(ы), который мы получаем, говорит нам, что произойдет, если мы увеличим или уменьшим одно из независимых значений.

```
import pandas
from sklearn import linear_model
```

```
df = pandas.read_csv("cars.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

regr = linear_model.LinearRegression()
regr.fit(X, y)

print(regr.coef_)
```

Result:

```
[0.00755095 0.00780526]
```

Рисунок 3.2 – Пример выполнения программы 2

Множественная регрессия – это статистический метод, используемый для моделирования зависимости одной переменной (зависимой переменной) от нескольких других переменных (независимых переменных). Коэффициенты регрессии показывают, как изменение каждой независимой переменной влияет на зависимую переменную. Значимость коэффициентов оценивается с помощью p-значений (p-values). Если p-значение для коэффициента меньше уровня значимости (обычно 0.05), то коэффициент считается статистически значимым.

Для оценивания качества модели множественной регрессии используются различные методы и коэффициенты. Вот основные из них:

1. Коэффициент детерминации (R^2).

Коэффициент детерминации показывает, какую долю дисперсии зависимой переменной объясняет модель.

2. Скорректированный коэффициент детерминации (R_{adj}^2).

Скорректированный коэффициент детерминации учитывает количество независимых переменных и количество наблюдений, что делает его более надежным при сравнении моделей с разным числом переменных.

3. Среднеквадратичная ошибка (MSE).

Среднеквадратичная ошибка измеряет среднюю квадратичную разницу между наблюдаемыми и предсказанными значениями.

4. Корень из среднеквадратичной ошибки (RMSE).

RMSE – это корень из MSE, что делает его более интерпретируемым в тех же единицах, что и зависимая переменная.

5. Средняя абсолютная ошибка (MAE).

MAE измеряет среднюю абсолютную разницу между наблюдаемыми и предсказанными значениями.

Индивидуальное задание

Не предусмотрено.

Методические указания по выполнению работы

1. Загрузите файл с исходными данными.

2. Выберите числовые поля для анализа, определите целевой признак.

3. Выполните предложенные программы, выведите значение признака и коэффициенты регрессии. Проанализируйте результат.

Список источников

1. Множественная регрессия в Python [Электронный ресурс]. – Режим доступа: https://www.delftstack.com/ru/howto/python/perform-multiple-linear-regression-python/: (дата обращения: 15.01.2023).

2. Полное руководство по линейной регрессии в Python [Электронный ресурс]. – Режим доступа: https://www.codecamp.ru/blog/linear-regression-python// (дата обращения: 15.01.2023).

3. Линейная регрессия и основные библиотеки Python для анализа данных и научных вычислений [Электронный ресурс]. – Режим доступа: https://notebook.community/agushman/coursera/src/cours_2/week_1/peer_review_linreg_height_weight /: (дата обращения: 15.01.2023).

Контрольные вопросы:

1. Каковы условия применимости множественной регрессии?
2. Как измерить ошибки предсказаний, выполненных с помощью множественной регрессии?
3. Как вывести коэффициенты и свободный член множественной регрессии?
4. Могут ли для множественной регрессии использоваться нечисловые переменные?
5. Как найти наиболее значимую из независимых переменных (переменную, которая оказывает наибольшее влияние на результат)?

ЛАБОРАТОРНАЯ РАБОТА № 4. ПРОГНОЗИРОВАНИЕ ВО ВРЕМЕННЫХ РЯДАХ. МОДЕЛЬ ARIMA

Цель:

Построение регрессионной модели для прогнозирования во временных рядах.

Теоретическая часть

Временные ряды (Time series) дают возможность прогнозировать будущие значения. На основании предыдущих значений, временные ряды можно использовать для прогнозирования тенденций в экономике, погоде, планировании пропускной способности и т. д. Прогнозирование временных рядов обычно используется во многих производственных компаниях, поскольку оно управляет первичным бизнес-планированием, закупками и производственной деятельностью. Любые ошибки прогнозов будут проявляться по всей цепочке поставок или любой бизнес-структуре в этом отношении. Таким образом, это важно для получения точных прогнозов, позволяющих сэкономить на затратах, и критически важно для успеха.

Модель ARIMA (AutoRegressive Integrated Moving Average) является одной из наиболее популярных и широко используемых моделей для временных рядов. Она сочетает в себе три компонента: авторегрессию (AR), интегрированность (I) и скользящее среднее (MA). Назначение и сферы применения модели ARIMA включают:

Назначение модели ARIMA

1. Прогнозирование временных рядов:

- Основное назначение модели ARIMA – это прогнозирование будущих значений временного ряда на основе его прошлых значений.

2. Моделирование стационарных и нестационарных данных:

- ARIMA может работать как с стационарными, так и с нестационарными данными. Нестационарные данные могут быть преобразованы в стационарные с помощью дифференцирования (компонент I).

3. Учет сезонности:

- Хотя базовая модель ARIMA не учитывает сезонность, её расширение – модель SARIMA (Seasonal ARIMA) – позволяет моделировать сезонные компоненты.

4. Анализ и понимание структуры данных:

- Модель ARIMA помогает выявить и понять структуру временного ряда, включая автокорреляцию, тренды и сезонность.

Ниже приведен пример использования модели ARIMA в Python. Этот пример демонстрирует, как использовать модель ARIMA для прогнозирования временного ряда в Python. В данном случае используется модель ARIMA (1, 1, 1), что означает один авторегрессионный компонент, одно дифференцирование и один компонент скользящего среднего.

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
# Пример данных
data = pd.Series([112, 118, 132, 129, 121, 135, 148, 148, 136, 119, 104, 118, 115, 126, 141, 135, 125, 149, 170, 170, 158, 133, 114, 140, 145, 150, 178, 163, 172, 178, 199, 199, 184, 162, 146, 166, 171, 180, 193, 181, 183, 218, 230, 242, 209, 191, 172, 194, 192, 186, 217, 231, 248, 209, 198, 195, 229, 243, 264, 217, 202, 200, 238, 260, 273, 224, 206, 204, 240, 267, 287, 232, 210, 206, 248, 273, 291, 235, 211, 211, 256, 280, 296, 236, 214, 218, 261, 281, 300])
# Построение модели ARIMA
model = ARIMA(data, order=(1, 1, 1))
model_fit = model.fit()
# Прогнозирование
forecast = model_fit.forecast(steps=12)
# Визуализация
plt.figure(figsize=(10, 6))
```

```
plt.plot(data, label='Оригинальные данные')
plt.plot(forecast, label='Прогноз', color='red')
plt.legend()
plt.show()
print("Прогнозируемые значения:", forecast)
```

Рассмотрим более подробно порядок действий:

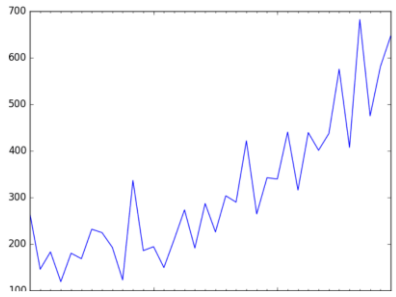
Сначала подключаем нужные библиотеки и загружаем датафрейм с исходными данными. Этот набор данных описывает ежемесячное количество продаж шампуня за трехлетний период.

```
from pandas import read_csv
from pandas import datetime
from matplotlib import pyplot
def parser(x):
return datetime.strptime('190'+x, '%Y-%m')
series = read_csv('shampoo.csv', header=0, parse_dates=[0],
index_col=0,
squeeze=True,
date_parser=parser)
print(series.head())
series.plot()
pyplot.show()
```

Выведем первые 5 строк из загруженного датафрейма:

```
Month
1901-01-01 266.0
1901-02-01 145.9
1901-03-01 183.1
1901-04-01 119.3
1901-05-01 180.3
Name: Sales, dtype: float64
```

Построим график зависимости:

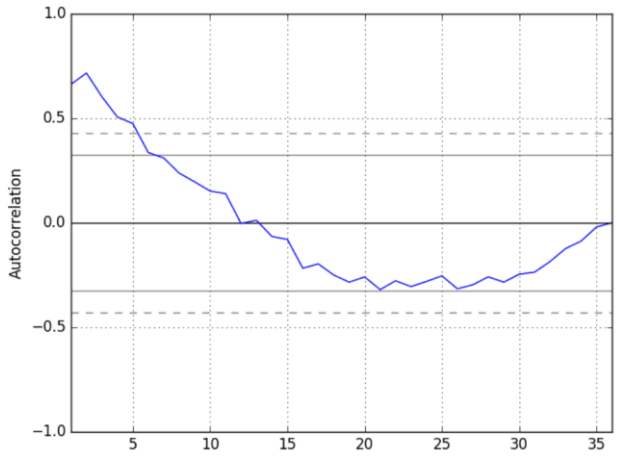


Мы видим, что набор данных по продажам шампуня имеет четкую линию тренда. Это говорит о том, что временной ряд не является стационарным и потребуется его продифференцировать, чтобы сделать его стационарным.

График автокоррекции

```
from pandas import read_csv
from pandas import datetime
from matplotlib import pyplot
from pandas.plotting import autocorrelation_plot
def parser(x):
return datetime.strptime('190'+x, '%Y-%m')
series = read_csv('shampoo.csv', header=0, parse_dates=[0], index_col=0,
squeeze=autocorrelation_plot(series)
pyplot.show()
```

График анализа автокорреляции:



Запустив пример, мы видим, что существует положительная корреляция для значений запаздывания (лага) 10–12, которая, возможно, существенна для значений лага до 5.

Хорошей отправной точкой для параметра AR модели ARIMA может быть 5.

Создание модели АРИМА

Определите модель, вызвав ARIMA() и передав параметры p, d и q.

Модель создается на обучающих данных путем вызова функции fit().

Предсказания можно сделать, вызвав функцию predict() и указав индекс времени или времени для предсказания. Мы обучим модель ARIMA по всему набору данных о продажах шампуней и рассмотрим остаточные ошибки.

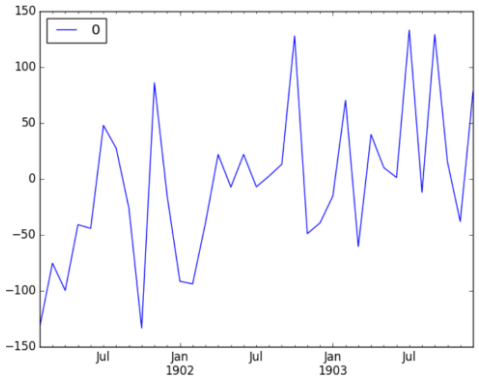
```
 # fit an ARIMA model and plot residual errors
from pandas import datetime
from pandas import read_csv
from pandas import DataFrame
from statsmodels.tsa.arima.model import ARIMA
from matplotlib import pyplot
# load dataset
def parser(x):
return datetime.strptime('190'+x, '%Y-%m')
series = read_csv('shampoo-sales.csv', header=0, index_col=0,
parse_dates=True, # fit model
model = ARIMA(series, order=(5,1,0))
model_fit = model.fit()
```

Вывод информации о точности модели и графиков ошибок

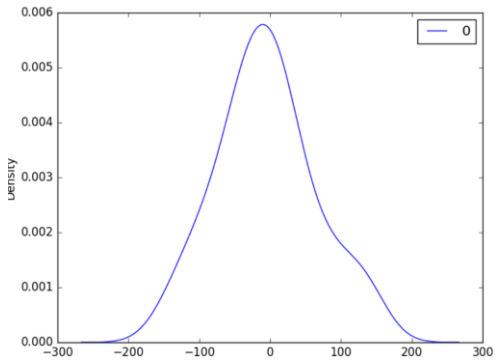
```
# fit an ARIMA model and plot residual errors
from pandas import datetime
from pandas import read_csv
```

```
from pandas import DataFrame
from statsmodels.tsa.arima.model import ARIMA
from matplotlib import pyplot
# load dataset
def parser(x):
return datetime.strptime('190'+x, '%Y-%m')
series = read_csv('shampoo-sales.csv', header=0, index_col=0,
parse_dates=True, # fit model
model = ARIMA(series, order=(5,1,0))
model_fit = model.fit()
# summary of fit model
print(model_fit.summary())
# line plot of residuals
residuals = DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
# density plot of residuals
residuals.plot(kind='kde')
pyplot.show()
```

Остаточные ошибки



Мы получаем линейный график остаточных ошибок, что означает, что все еще может быть некоторая информация о тренде, не воспроизводимая моделью.



Затем мы получаем график плотности остаточных значений ошибок, предполагая, что ошибки являются гауссовыми, но могут не центрироваться в нуле.

Параметры, характеризующие результат работы модели

```
==============================================================================
Dep. Variable:                  Sales   No. Observations:                   36
Model:                 ARIMA(5, 1, 0)   Log Likelihood                -198.485
Date:                Sat, 14 May 2022   AIC                            408.969
Time:                        16:05:03   BIC                            418.301
Sample:                    01-31-1901   HQIC                           412.191
                         - 12-31-1903
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.9014      0.247     -3.647      0.000      -1.386      -0.417
ar.L2         -0.2284      0.268     -0.851      0.395      -0.754       0.298
ar.L3          0.0747      0.291      0.256      0.798      -0.497       0.646
ar.L4          0.2519      0.340      0.742      0.458      -0.414       0.918
ar.L5          0.3344      0.210      1.593      0.111      -0.077       0.746
sigma2      4728.9607   1316.021      3.593      0.000    2149.607    7308.314
===================================================================================
Ljung-Box (L1) (Q):                   0.61   Jarque-Bera (JB):                 0.96
Prob(Q):                              0.44   Prob(JB):                         0.62
Heteroskedasticity (H):               1.07   Skew:                             0.28
```

Автокорреляция – статистическая взаимосвязь между последовательностями величин одного ряда, взятыми со сдвигом, например, для случайного процесса – со сдвигом по времени. Данное понятие широко используется в эконометрике. Наличие автокорреляции случайных ошибок регрессионной модели приводит к ухудшению качества МНК-оценок параметров регрессии, а также к завышению тестовых статистик, по которым

проверяется качество модели (то есть создается искусственное улучшение качества модели относительно её действительного уровня точности).

Поэтому тестирование автокорреляции случайных ошибок является необходимой процедурой построения регрессионной модели.

Задание для выполнения

1. Скачайте датафрейм с сайта или используйте датафреймы, представленные в ЭИОС:
https://raw.githubusercontent.com/selva86/datasets/master/wwwusage.csv

2. Постройте график основной величины и график автокорреляции.

3. Постройте модель ARIMA с параметрами (1,1,2).

4. Нарисуйте график колонки value и график автокорреляции.

5. Постройте модель ARIMA с параметрами 2,1,2.

Индивидуальное задание

Не предусмотрено

Методические рекомендации по выполнению работы

Используйте примеры программного кода из теоретической части**.**

Список источников

1. Прогнозирование временных рядов с помощью Arima в Python 3 [Электронный ресурс]. – Режим доступа: https://www.8host.com/blog/prognozirovanie-vremennyx-ryadov-s-pomoshhyu-arima-v-python-3/: (дата обращения: 25.03.2023).

2. Модель ARIMA в Python для прогнозирования временных рядов [Электронный ресурс]. – Режим доступа: https://pythonpip.ru/examples/model-arima-v-python: (дата обращения: 25.03.2023).

3. Анализ временных рядов с помощью python [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/207160/: (дата обращения: 25.03.2023).

Контрольные вопросы

1. Расшифруйте название модели – ARIMA.

2. Объясните смысл трех основных параметров модели – model = ARIMA(series, order=(5,1,0)).

3. В чем значение графика автокорреляции?

4. В чем значение графика плотности остаточных ошибок?

5. Что означают величины AIC, BIC и HQIC?

ЛАБОРАТОРНАЯ РАБОТА № 5. ОСНОВЫ РАБОТЫ С БИБЛИОТЕКОЙ PYTORCH

Теоретическая часть

Библиотека PyTorch является универсальным инструментом машинного обучения. Она популярна при проектировании новых архитектур нейронных сетей.

В библиотеке есть четыре ключевых составляющих:

- Развитый инструментарий для работы с тензорами. Он похож на NumPy, но даёт дополнительные возможности по контролю выделяемой памяти, что важно при работе с большими моделями и данными.
- Простое построение динамического вычислительного графа, позволяющего получать градиенты целевых функций от параметров модели.
- Большой набор готовых слоёв для построения нейронных сетей произвольной архитектуры.
- Возможность перенаправлять вычисления на графические процессоры GPU.

Создание тензоров

```
# Import torch
_____

# создайте случайный тензор размера 3 на 3
your_first_tensor = torch.____(3, 3)

# вычислите размерность тензора
tensor_size = your_first_tensor.____

# напечатайте компоненты тензорав и его размерность
print(____)
print(____)
```

Умножение матриц

```
# создайте матрицу размером 3 на 3
tensor_of_ones = torch.____(3, 3)

# создайте единичную матрицу размером 3 на 3
identity_tensor = torch.____(3)

# выполните матричное умножение матрицы tensor_of_ones на матрицу identity_tensor
matrices_multiplied = torch.____(tensor_of_ones, ____)
print(matrices_multiplied)

# выполните поэлементное умножение матрицы tensor_of_ones на матрицу identity_tensor
element_multiplication = ____
print(element_multiplication)
```

В последние годы о нейронных сетях (НС) было сказано и написано много – от концепции персептрона до сложной многослойной архитектуры нейронов. Лабораторная работа представляет собой попытку объяснить два фундаментальных алгоритма (прямого распространения и обратного распространения) которые обеспечивают работу нейронной сети.

Эти алгоритмы демонстрируются в простейшей форме с использованием программы Microsoft Excel.

Задание для выполнения

Описание примера:

Рассматриваемый пример действительно базовый и далек от реальности. Цель заключается в том, чтобы сделать его простым и интуитивно понятным, чтобы понять рабочую логику, а не сосредотачиваться на сложной математике, стоящей за ней. Для начала рассмотрим только один входной вектор:

V = [X1 = 1,X2 = 0, X3 = 1,X4 = 0]

с одним скрытым слоем, состоящим из 3 нейронов и выходного слоя.

Целевая функция равна 1.

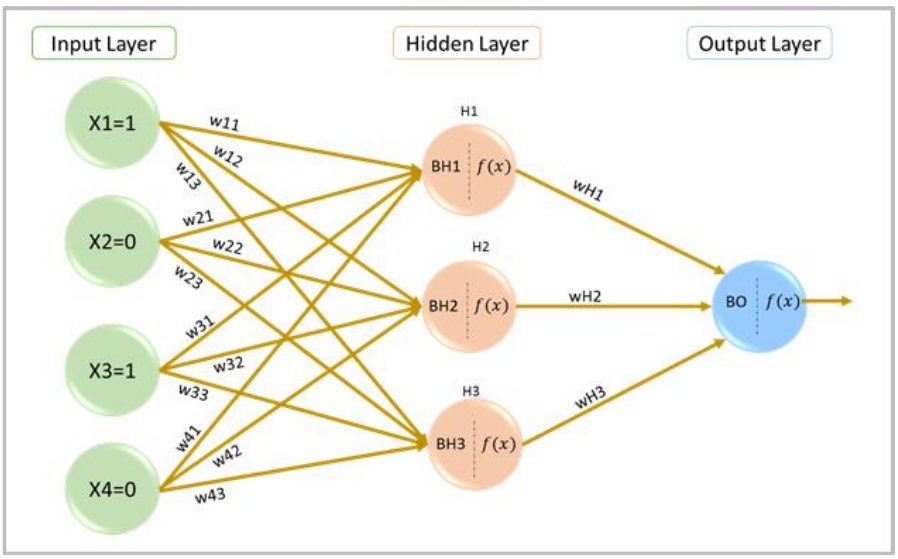


Рисунок 5.1 – Схема нейронной сети

Часть 1. Задание нейронной сети

Ввод и вывод. В качестве примера, скажем, мы ожидаем, что алгоритм выдаст результат «1» для указанного ненулевого значения для «X1» и «X3» (скажем, 1) и ноль для «X2» и «X4».

Итак, рассматриваемый здесь входной вектор равен [1,0,1,0].

Input Vector				*Target Output*
X1	*X2*	*X3*	*X4*	
1	0	1	0	1

Рисунок 5.2 – Исходная таблица

Прямое распространение информации. Шаг 1.

Инициализируйте сетевые параметры

Первый шаг – инициализировать веса и смещения с помощью функции rand() в MS Excel. (P.S: выделенные ячейки во всех таблицах ниже представляют производные значения на основе формул:

=SUMPRODUCT(1,0,1,0,0.49,0.35,0.44,0.80)

=SUMPRODUCT(1,0,1,0,0.72,0.90,0.58,0.92)

=SUMPRODUCT(1,0,1,0,0.38,0.43,0.39,0.21)

Input Vector				Target Output
X1	X2	X3	X4	
1	0	1	0	1

Рисунок 5.3 – Результат выполнения 1 шага

Веса и смещения

Weights: Input to Hidden Layer					
w11	w12	w13	0.49	0.72	0.38
w21	w22	w23	0.35	0.90	0.43
w31	w32	w33	0.44	0.58	0.39
w41	w42	w43	0.80	0.92	0.21

Bias at Hidden Layer	
BH1	0.40
BH2	0.00
BH3	0.22

Рисунок 5.4 – Таблица значений весовых коэффициентов

Прямое распространение информации. Шаг 2.

Рассчитайте чистый ввод в скрытых узлах слоя Чистый вход – умноженный на вес, а затем увеличенный на значение смещения. При использовании матричного умножение входного вектора [1x4] и весов [4x3], результирующая матрица имеет размерность [1X3].

Чтобы это работало в Excel, используйте =СУММПРОИЗВ(), чтобы получить результирующую матрицу [1X3], как показано ниже.

=СУММПРОИЗВ(1,0,1,0,0.49,0.35,0.44,0.80)

=СУММПРОИЗВ(1,0,1,0,0.72,0.90,0.58,0.92)

=СУММПРОИЗВ(1,0,1,0,0.38,0.43,0.39,0.21)

Прямое распространение информации. Шаг 3

Теперь добавьте смещения к этим входным значениям, умноженным на вес: используемые формулы:

=СУМ(0.92,0.40)

=СУМ(1.30,0.00)

=СУМ(0.76,0.99)

Input times weight at Hidden Layer			+	Biases			=	Net Input		
0.92	1.30	0.76		0.40	0.00	0.22		1.33	1.30	0.99

Рисунок 5.5 – Результата выполнения 3 шага

Прямое распространение информации. Шаг 4.

Пропустите входные значения (X) через функцию активации (сигмоида). Давайте передадим выходные данные «Шага 2» [1.33,1.30,0.99] в качестве входных данных для функции активации на каждом нейроне скрытого слоя как [f(1.33), f(1.30), f(0.99)].

В Excel это можно сделать, определив новую функцию f(x) =

1/(1+exp (-x)).

=1/(1+exp(-1.33))

=1/(1+exp(-1.30))

=1/(1+exp(-0.99))

Net Input	f(Net Input)		Output (Hidden Layers)
1.33	f(1.33)	=	0.79
1.30	f(1.30)		0.79
0.99	f(0.99)		0.73

Рисунок 5. 6 – Результата выполнения 4 шага

Прямое распространение информации. Шаг 5

Рассчитайте чистый вход в выходном узле. Теперь выходные данные «Шага 3» [0,79,0,79,0,73] будут действовать как входные данные для выходного узла. Давайте повторим «Шаг 2» с входным вектором [0,79,0,79,0,73], вектором весов [0,71,0,16,0,57] и выходным смещением [0,83].

=СУММПРОИЗВ (0.79,0.79,0.73, 0.71,0.16,0.57) + 0.83 после

упрощения = 1.93

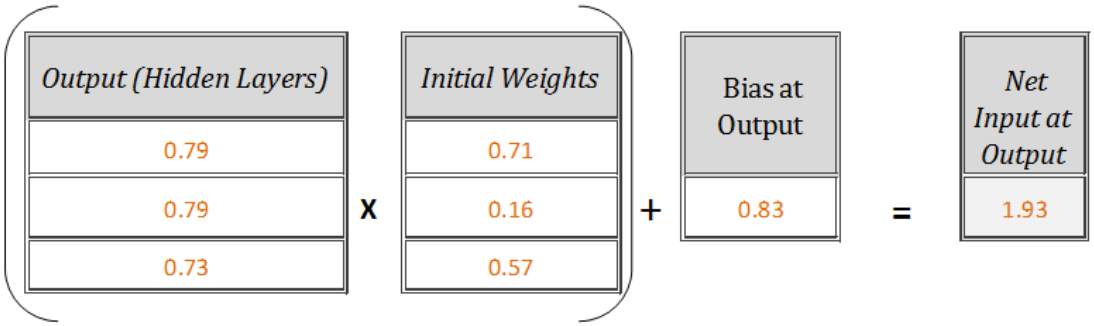


Рисунок 5.7 – Результата выполнения 5 шага

Прямое распространение информации. Шаг 6

Получите окончательный результат прямого распространения информации в нейронной сети. Передайте результат, полученный на «Шаге 4» [1.93], в функцию активации как f(1.93), которую снова можно вычислить, используя f(x)=1/(1+exp(-x)), в результате чего окончательный вывод нейронной сети.

=1/(1+exp(-1,93))

Итоговое значение

0.87

Часть 2. Пример использования библиотеки PyTorch

```
import torch
torch.rand(3,4)
x = torch.Tensor([1,0,1,0])
w = torch.Tensor([[0.49,0.35,0.44,0.80],[0.72,0.90,0.58,0.92],[0.38,0.43,0.39,0.21]])
input_for_hidden = torch.matmul(w,x)
print(input_for_hidden)
bias_at_hidden_layer = torch.Tensor([0.4,0.0,0.22])
inactivated_at_hidden_layer = input_for_hidden+bias_at_hidden_layer
output_hidden_layer = torch.sigmoid(inactivated_at_hidden_layer)
print(output_hidden_layer)
wO = torch.Tensor([0.71,0.16,0.57])
input_for_out_layer = torch.matmul(wO,output_hidden_layer)
bias_output_layer = 0.83
inactivated_output = input_for_out_layer+bias_output_layer
output_for_out_layer = torch.sigmoid(inactivated_output)
```

```
print(output_for_out_layer)
loss = torch.abs(output_for_out_layer - 1.0)
print(loss)
optimizer = torch.optim.SGD([w, BH, WH, BO], lr = 0.1)
for n in range(100):
 optimizer.zero_grad()
 input_for_hidden = torch.matmul(w,x)
 inactivated_at_hidden_layer = input_for_hidden+bias_at_hidden_layer
 output_hidden_layer = torch.sigmoid(inactivated_at_hidden_layer)
```

Результат

```
tensor([0.9300, 1.3000, 0.7700])
tensor([0.7908, 0.7858, 0.7291])
tensor(0.8736)
tensor(0.1264)
```

Методические указания для выполнения задания

1. Сначала выполните работу в электронных таблицах Excel с указанными числами (см. пример выше).

2. Затем повторите действия со случайными числами.

3. Потренируйтесь работать с тензорами (см. пример в теоретической части).

4. Реализуйте вариант программы с помощью библиотеки PyTorch.

5. Сформируйте функцию потерь и проведите обучение нейронной сети, т. е. подберите значений весов и смещений, минимизирующих функцию потерь.

Список источников

1. Туториал по PyTorch: от установки до готовой нейронной сети [Электронный ресурс]. – Режим доступа: https://neurohive.io/ru/tutorial/glubokoe-obuchenie-s-pytorch//: (дата обращения: 15.01.2023).

2. PyTorch – ваш новый фреймворк глубокого обучения [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/334380//: (дата обращения: 15.01.2023).

3. PyTorch – Простой справочник [Электронный ресурс]. – Режим доступа: Для начинающих https://pythobyte.com/pytorch-97342f47//: (дата обращения: 15.01.2023).

Контрольные вопросы

1. Что такое тензор PyTorch?
2. Зачем нужен атрибут requires_grad?
3. Какую роль играет функция потерь?
4. Что такое optimizer?
5. Зачем выполняется вызов zero_grad() ?

ЛАБОРАТОРНАЯ РАБОТА № 6. НЕЙРОННАЯ СЕТЬ ДЛЯ АППРОКСИМАЦИИ ФУНКЦИИ СИНУС

Цель работы:

Построение нейронной сети для построения задачи регрессии.

Теоретическая часть.

Порядок действий при построении нейронной сети для аппроксимации функции следующий:

1. Подготовка данных:

• Создайте набор данных, содержащий пары значений (x,sin(x)). Например, можно сгенерировать данные для x в диапазоне от 0 до 2π.

2. Разделение данных:

• Разделите данные на обучающую и тестовую выборки. Обычно используется соотношение 80/20 или 70/30.

3. Создание модели нейронной сети:

• Используйте библиотеку для машинного обучения, такую как TensorFlow или PyTorch, для создания модели нейронной сети.

• Определите архитектуру нейронной сети. Например, можно использовать простую полносвязную нейронную сеть с несколькими скрытыми слоями.

4. Компиляция модели:

• Выберите функцию потерь (например, среднеквадратичная ошибка) и оптимизатор (например, Adam).

5. Обучение модели:

• Обучите модель на обучающей выборке, используя метод обратного распространения ошибки.

6. Оценка модели:

• Оцените производительность модели на тестовой выборке, чтобы убедиться, что она хорошо аппроксимирует функцию синус.

7. Визуализация результатов:

• Визуализируйте результаты, чтобы увидеть, насколько хорошо модель аппроксимирует функцию синус.

Задание для выполнения:

1. Запустить представленный код (предсказание значения функции Sin()), вывести все графики.

2. Заменить функцию: синус на косинус и/или экспоненту, вывести все графики.

3. Заменить оптимизатор SGD на Adam, нарисовать график убывания функции потерь отдельно для каждого оптимизатора.

4. Варьировать параметры: количество нейронов скрытого слоя, скорость обучения и количество эпох (только для синуса), и составить таблицу, в которой отметить точность аппроксимации для каждого сочетания параметров.

```
import matplotlib.pyplot as plt
import matplotlib
import torch
```

После подключения библиотек формируется обучающая выборка:

```
x_train = torch.rand(100)
x_train = x_train*20 - 10.0

y_train = torch.sin(x_train)

plt.plot(x_train.numpy(),y_train.numpy(),'o')
plt.show()
```

Формирование шумовой добавки:

```
noise = torch.randn(y_train.shape)/5.0
plt.plot(x_train.numpy(),noise.numpy(),'o')
plt.title('Gaussian noise')
plt.show()
```

Добавка шума к обучающей выборке.
В итоге получается зашумленный синус. Это похоже на сигнал в реальной жизни. Вопрос в том, сможет ли нейронная сеть его распознать.

```
y_train = y_train+noise
plt.plot(x_train.numpy(),y_train.numpy(),'o')
```

```
plt.title('noisy sine')
plt.show()
```

Изменение размерности тензоров обучающей выборки: превращение в столбцы:

```
x_train.unsqueeze_(1)
y_train.unsqueeze_(1)
```

Создание проверочной (валидационной) выборки:

```
x_validation = torch.linspace(-10,10,100)
y_validation = torch.sin(x_validation)
plt.figure(4)
plt.plot(x_validation.numpy(),y_validation.numpy(),'o')
plt.title('sin(x)')
plt.xlabel('x_validation')
plt.ylabel('y_validation')

x_validation.unsqueeze_(1)
y_validation.unsqueeze_(1)
```

Создание нейронной сети:

```
class SineNet(torch.nn.Module):
    def __init__(self,n_hidden_neurons):
        super(SineNet,self).__init__()
        self.fc1 = torch.nn.Linear(1,n_hidden_neurons)
        self.act1 = torch.nn.Sigmoid()
        self.fc2 = torch.nn.Linear(n_hidden_neurons,1)

    def forward(self,x):
        x = self.fc1(x)
        x = self.act1(x)
        x = self.fc2(x)
        return x

sn = SineNet(50)

print('sn created')
```

Предсказание:

```
def predict(net,x,y,title):
```

```
    y_pred =net.forward(x)

    plt.plot(x.numpy(),y.numpy(),'o',label='Truth')
    plt.plot(x.numpy(),y_pred.data.numpy(),'o',
             c='r',label='Prediction')

    plt.title(title)
    plt.show()
plt.figure(5)
predict(sn,x_validation,y_validation,'1st prediction attempt ')
```

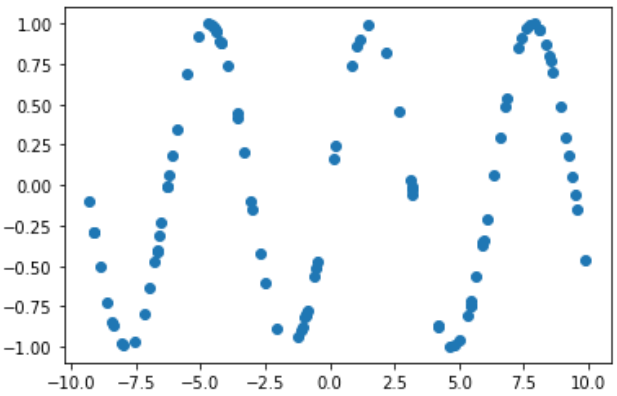
Оптимизатор и функция потерь:

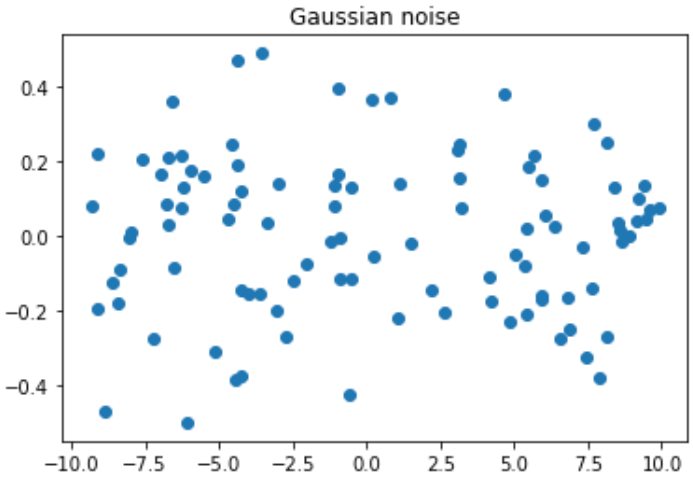
```
optimizer = torch.optim.Adam(sn.parameters(),lr=0.01)

def loss(pred,target):
    squares = (pred - target)**2
    return squares.mean()
```

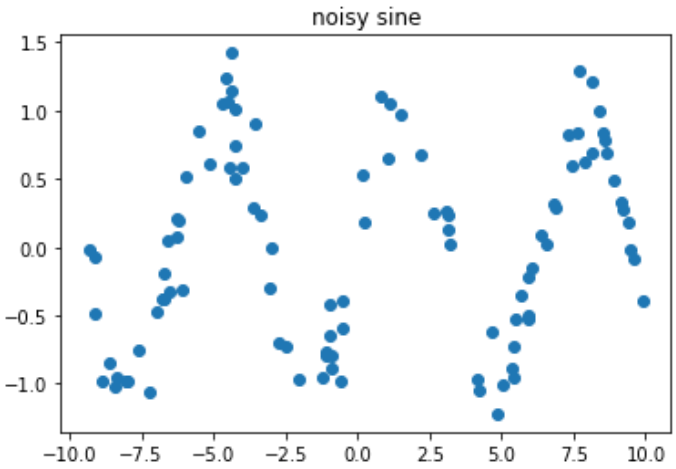
Цикл обучения нейронной сети:

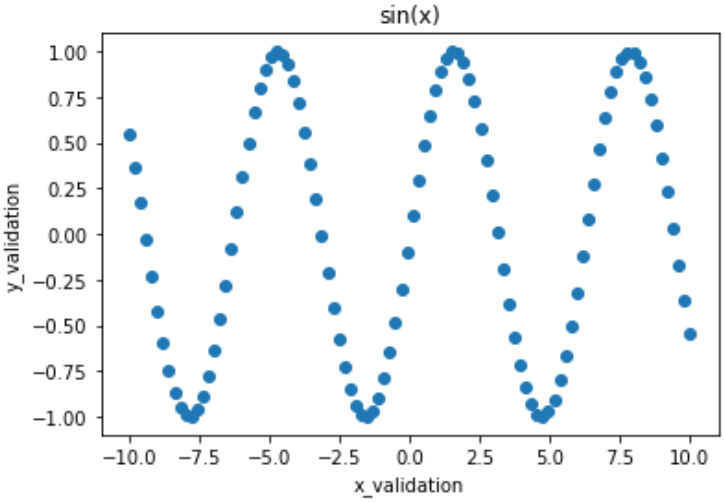
```
for epoch_index in range(500):
   # print('Epoch==================== ',epoch_index)
    optimizer.zero_grad()
    y_pred = sn.forward(x_train)
    loss_val = loss(y_pred,y_train)
    loss_val.backward()
    optimizer.step()
    #print(epoch_index)

print('FINAL LOSS VALUE ',loss_val)

plt.figure(6)
predict(sn,x_validation,y_validation,'prediction after training')
plt.show()
```

В результате получаются следующие графики:


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Gaussian noise
0.4
0.2
0.0
−0.2
−0.4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


noisy sine
1.5
1.0
0.5
0.0
−0.5
−1.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0



Методические указания к выполнению задания:

1. Выполните задание, используя программный код из теоретической части и оформите отчет.

Список источников

1. Аппроксимируем функцию с помощью нейросети [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/428281//: (дата обращения: 12.11.2024).

2. Нейросеть (регрессия) [Электронный ресурс]. – Режим доступа: https://help.loginom.ru/userguide/processors/datamining/neural-network-regression.html//: (дата обращения: 12.11.2024).

Контрольные вопросы:

1. Чем отличаются тренировочный (обучающий) и тестовый (проверочный) наборы данных?

2. Что такое оптимизатор?

3. Что такое эпоха обучения нейронной сети?

4. Для чего вызывается функция zero_grad() ?

5. Где задается и на что влияет скорость обучения?

ПРИЛОЖЕНИЕ

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое квантиль?

Ответ: Квантиль – значение, которое заданная случайная величина не превышает с фиксированной вероятностью.

2. Что такое процентиль?

Ответ: Процентиль – это значение, которое заданная случайная величина не превышает с фиксированной вероятностью, заданной в процентах.

3. Как устроен CSV-файл?

Ответ: Это текстовый файл, в котором содержится информация.

Каждая строка – это отдельная строка таблицы, а столбцы отделены один от другого специальными символами – разделителями (например, запятой). Разделителем может быть не только запятая, но и другие символы (пробел, точка с запятой, табуляция, другое).

4. Как обеспечить доступ к элементу данных датафрейма df в колонке с именем AAA и в строке с номером 8?

Ответ: df['AAA'][8].

5. Как заменить содержание всей колонки?

Ответ: Для колонки с названием ’AAA’ и массива библиотеки Numpy с названием arr (количество элементов массива равно количеству строк в колонке): df['AAA'] = arr.

6. Как вывести список названий всех колонок датафрейма?

Ответ: df.columns.

7. Как изменить разделитель целой и дробной части (запятую на точку, или наоборот)?

Ответ: Если данные содержаться в числовой (вещественной) колонке с названием ’a’, то замена запятой на точку выполняется следующим образом:

df['a'] = df['a'].str.replace(',', '.').astype(float).

8. Как избавится от нечисловых значений (NaN) в датафрейме?

Ответ: NaN (сокращение от Not a Number – «не число») – нечисловое значение, появляющееся в прочитанных из файла датафреймах вследствие ошибок. Любые арифметические операции с использованием NaN в качестве операнда в результате также дадут NaN. Замена NaN на 0 во всем датафрейме выполняется с помощью следующей операции: df.fillna(0).

9. Можно ли открыть датафрейм в Microsoft Excel?

Ответ: Да, можно, CSV-файл может быть открыт с помощью Microsoft Excel.

С другой стороны, XLSX-файл может быть прочитан средствами Pandas:

import pandas as pd

pd.read_excel ('tmp.xlsx', index_col=0).

10. Назовите необходимое и достаточное условие применимости парной линейной регрессии.

Ответ: Наличие корреляции между зависимой и независимой переменными (коэффициент корреляции Пирсона по модулю больше 0.5).

Коэффициент корреляции Пирсона вычисляется следующим образом: df.corr()

11. Что входит в понятие построения линейной регрессии (что выполняет функция fit)?

Ответ: Функция fit() выполняет вычисление коэффициента и свободного члена линейной регрессии.

12. Как вывести коэффициент и свободный член линейной регрессии для построенной модели?

Ответ:

linr = LinearRegression()

linr.fit(X,Y)

print(linr.coef_)

print(linr.intercept_).

13. В каком интервале значений независимой переменной применимы предсказания, полученные на основе линейной регрессии?

Ответ: В интервале между максимальным и минимальным значениями независимой переменной.

14. Как вычисляется ошибка аппроксимации?

Ответ: С помощью средней ошибки по модулю с процентах (MAPE, Mean Absolute Percent Error):

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i - y_i|}{|y_i|} \times 100\%$$

здесь

• N – количество наблюдений (точек на графике);

• yi – реальные значения независимой переменной;

• ˆyi – значения независимой переменной, предсказанные моделью.

15. Каковы характеристики хорошо подобранной модели?

Ответ: Модель считается подобранной достаточно хорошо, если средняя ошибка аппроксимации не превышает 8–10 %.

16. Как найти наиболее значимую из независимых переменных (переменную, которая оказывает наибольшее влияние на результат)?

Ответ: Для датафрейма boston_df и построенной на его основе модели линейной множественной регрессии linear_regression_model следующий код выдаст список независимых переменных, отсортированных по модулю коэффициентов линейной регрессии:

var_df = sorted(list(zip(boston_df.columns, linear_regression_model.coef_)),

key=lambda x: abs(x[1])).

17. Расшифруйте название модели – ARIMA.

Ответ: Означает AutoRegressive Integrated Moving Average – авторегрессионная интегрированная модель скользящего среднего.

18. Объясните смысл трех основных параметров модели –

model = ARIMA(series, order=(p,d,q)).

Ответ: (a) p – параметр автокорреляции. Он означает количество значений временного ряда, соответствующих моментам времени в прошлом,

используемых для предсказания значения временного ряда, соответствующего текущему моменту времени.

(b) параметр d показывает, сколько раз рассматриваемый временной ряд будет продифференцирован для достижения стационарности.

(c) q – размер окна скользящего среднего.

19. В чем значение графика автокорреляции?

Ответ: График автокорреляции показывает зависимость коэффициента корреляции для различных участков временного ряда от разницы по времени между этими участками. На основе графика автокорреляции определяется параметр p.

20. В чем значение графика плотности остаточных ошибок?

Ответ: График остаточных ошибок показывает качество построенной модели. Для правильно построенной модели максимум графика расположен в нуле, сам график симметричен и близок к гауссовой кривой.

21. Что означают величины AIC, BIC и HQIC?

Ответ:

• AIC – Информационный критерий Акаике. Это метрика, которая помогает оценить качество модели. Входными параметрами для него являются значения функции максимального правдоподобия, а также общее количество параметров. Чем ниже это значение, тем лучше работает модель.

• BIC – Байесовский информационный критерий. Он очень похож на AIC, но также учитывает количество строк в наборе данных. Опять же, чем ниже BIC, тем лучше работает модель. Значение BIC сильнее уменьшается для моделей со сложными параметрами по сравнению с AIC.

• HQIC – Информационный критерий Ханнана-Куина. В статистике и анализе данных информационный критерий Ханнана-Куина используется для сравнения моделей с разным числом параметров, когда требуется выбрать лучший набор независимых переменных. Лучшая модель имеет наименьшее значение критерия.

22. Что такое тензор PyTorch?

Ответ: Массив произвольной размерности и неограниченного размера, состоящий из вещественных (одинарной и двойной точности) компонент, реализуемый средствами библиотеки PyTorch.

23. Зачем нужен атрибут requires_grad?

Ответ: Этот атрибут тензора PyTorch разрешает вычислением производных по данному тензору.

24. Какую роль играет функция потерь?

Ответ: Функция потерь показывает ошибку аппроксимации некоторой величины с помощью нейронной сети. Производная функции потерь используется для вычисления новых значений весов и смещений нейронной сети.

25. Чем отличаются тренировочный (обучающий) и тестовый (проверочный) наборы данных?

Ответ: Тренировочный и тестовый наборы данных должны иметь одинаковые свойства с точки зрения математической статистики, при этом обучающий набор данных используется только для обучения нейронной сети, а тестовый только для проверки качества обучения.

26. Что такое оптимизатор?

Ответ: Оптимизатор (optimizer) – объект, изменяющий параметры (веса и смещения) нейронной сети в соответствии с вычисленными значениями производных для минимизации функции потерь нейронной сети.

27. Что такое эпоха обучения нейронной сети?

Ответ: Эпоха – одна итерация в процессе обучения, включающая предъявление всех примеров из обучающего множества, перерасчет весов и смещений нейронной сети и, возможно, проверку качества обучения на контрольном множестве. Процесс обучения осуществляется на обучающей выборке.

28. Для чего вызывается функция zero_grad() ?

Ответ: Для обнуления градиентов, вычисленных в ходе предыдущей эпохи обучения нейронной сети.

29. Где задается и на что влияет скорость обучения?

Ответ: Скорость обучения задается в параметрах оптимизатора (параметр lr)

optimizer = torch.optim.Adam(sn.parameters(),lr=0.01).

Скорость обучения влияет на то, насколько быстро изменяются веса и смещения нейронной сети под влиянием оптимизатора. Слишком маленькие значения скорости обучения приведут к большому времени обучения нейронной сети. Слишком большие значения скорости обучения приведут к плохой точности аппроксимации.

Локальный электронный методический материал

Алексей Владимирович Снытников

Елена Юрьевна Заболотнова

ВЫСОКОУРОВНЕВЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Редактор С. Кондрашова
Корректор Т. Звада

Уч.-изд. 2,9. Печ. л. 3,5.

Издательство федерального государственного бюджетного
образовательного учреждения высшего образования
«Калининградский государственный технический университет».
236022, Калининград, Советский проспект, 1