Документацию по каждой функции можно вызвать командой help (название функции).

Т31.

1. **Методы частных производных для поиска локальных экстремумов:**

   def inputChastProizv() –Функция для ввода данных функции, для которой будут находиться локальные экстремумы.

   Параметры:
   params: str
   Строка на вход со списком переменных через пробел
   F: str
   Функция в виде строки
   L: str
   Обозначение для наличия ограничений функции('1' - есть, '0' - нет)
   lim1, lim2: str
   Ограничения функции

   def chastproizv(dictionary) - Функция для поиска экстремумов функции с пом. метода частных производных.

   Параметры:
   dictionary: dict
   На вход подаётся return функции inputChastProizv() - словарь со всеми переменными и функциями, которые ранее были введены.
   def graph(d) - Функция для построения трёхмерного графика.
   Параметры:
   d: dict
   Словарь с функцией и её переменными, координатами точек экстремумов и значением функции в них, типов экстремумов.

2. **Метод множителей Лагранжа.**

   def inputForLagrange() - Функция для ввода данных функции, для которой будут находиться локальные экстремумы.

   Параметры:
   params: str
   Строка на вход со списком переменных через пробел

F: str

Функция в виде строки

L: str

Обозначение для наличия ограничений функции('1' - есть, '0' - нет)

lim1, lim2: str

Ограничения функции

Z: str

Ограничивающая функция

def Lagrange(dictionary) - Функция для поиска экстремумов функции с пом. метода Лагранжа.

Параметры:

dictionary: dict

На вход подаётся return функции inputForLagrange() - словарь со всеми переменными и функциями, которые ранее были введены.

def graph(d) - Функция для построения трёхмерного графика.

Параметры:

d: dict

Словарь с функцией и её переменными, координатами точек экстремумов и значением функции в них, типов экстремумов.

Т32 – Методы одномерной оптимизации

1. Поиск экстремума функции одной переменной методом золотого сечения.

def inputzolotoe() - Функция для ввода данных, с помощью которого будет осуществляться метод золотого сечения

Параметры:

param: str Строка на вход с одной переменной, которая может быть задана любой русской или английской буквой

f: str Функция в аналитическом виде в формате строки

a: str Левая граница оптимизации в формате строки

b: str Правая граница оптимизации в формате строки

def zolotoe(data) - Функция, которая осуществляет алгоритм метода золотого сечения, а также построения графика зависимости значения функции от номера итерации.

Параметры:

data: dictionary Словарь, со всеми переменными, которые были заданы в функции ввода

f: str Функция в аналитическом виде в формате строки

a: float Левая граница оптимизации

b: float Правая граница оптимизации

e: float Точность алгоритма

alpha: float Константа, используемая для расчетов

n: int Количество итераций алгоритма

k: list Список со значением функции на каждой итерации

x1, x2, y1, y2: float Переменные, используемые для осуществления метода золотого сечения.

2. Поиск экстремума функции одной переменной методом парабол.

def inputParabol() - Функция для ввода данных, с помощью которого будет осуществляться метод парабол.

Параметры:

params: str Строка на вход с одной переменной, которая может быть задана любой русской или английской буквой

F: str Функция в аналитическом виде в формате строки

lim1: str Границы оптимизации через пробел в формате строки

def parabola(dictionary, eps=0.00001, maxk=500) - Функция, которая осуществляет алгоритм метода парабол.

Параметры:

eps: float точность оптимизации

maxk: int максимальное количество шагов алгоритма

dictionary: dict Словарь, со всеми переменными, которые были заданы в функции ввода

f: str Функция в аналитическом виде в формате строки

lim1: list Границы области оптимизации

3. **Алгоритм неточной одномерной минимизации (Алгоритм Бройдена — Флетчера — Гольдфарба — Шанно).**
   def f() - Функции для перевода заданной функции из аналитического вида в нужный для дальнейших вычислений.

   Параметры:

   x: str

def f1() - Функции для перевода градиента заданной функции из аналитического вида в нужный для дальнейших вычислений.

Параметры:

x: str - Переменная x.

def func() - Функция для ввода аналитической функции, для которой будут находиться экстремум.

Параметры:

Var1: str Строка на вход с переменной x

Var2: str Строка на вход с переменной y

f: str Строка на вход с функцией f в аналитическом виде.

def BFGS() - Функция для нахождения координаты точки экстремума и значения ф-ции в этой точке.

Параметры:

func: Функция.

grad: np.array Градиенты функции.

x0: list Список с координатами точки.

v1: float Параметр для 1 условия Вольфа.

v2: float Параметр для 2 условия Вольфа.

xmax : int Максимально возможное значение аргумента ф-ции.

interv: float Порог выхода по длине интервала поиска

maxiter: int Максимальное количество итераций.

p1: bool - Флаг «вывод промежуточных результатов»

p2: bool Флаг «запись промежуточных результатов в датасет»

ТЗ3 – Градиентные методы.

1. **Метод градиентного спуска с постоянным шагом и с дроблением шага.**

def func() - Функции для перевода заданной функции из аналитического вида в нужный для дальнейших вычислений.

Параметры:

variables: list Используемые переменные.

f: str Функция в аналитическом виде

def gradFunc() - Функция для нахождения градиента заданной функции.

Параметры:

var1, var2: str Используемые переменные.

f: str Функция в аналитическом виде

def f(x) - Функция, возвращающая введенную функцию для дальнейших вычислений.

def g(x) - Функция, возвращающая массив найденного градиента функции.

def gradient_const_step() - Функция для нахождения координаты точки минимума и значения функции в этой точке методом градиентного спуска с постоянным шагом.

Параметры:

func: str Функция в нужном виде.

grad: np.array Градиент функции.

nvar: int Количество используемых переменных.

h: float Шаг для градиентного спуска.

maxk: int Максимальное количество итераций для нахождения ответа.

eps : float Заданная точность алгоритма.

def gradient_changing_step() - Функция для нахождения координаты точки минимума и значения функции в этой точке методом градиентного спуска с дроблением шагом.

Параметры:

func: str Функция в нужном виде.

grad: np.array Градиент функции.

nvar: int Количество используемых переменных.

sigm: str Параметр дробления шага.

h: float Шаг для градиентного спуска.

maxk: int Максимальное количество итераций для нахождения ответа.

eps: float Заданная точность алгоритма.

2. Метод наискорейшего спуска.

def inputgrad()- Функция для ввода данных, с помощью которого будет осуществляться алгоритм Ньютон-сопряженного градиента

Параметры:

params: str Строка на вход с переменными, которые могуи быть заданы любой русской или английской буквой
F: str Функция в аналитическом виде в формате строки

def grad(dictionary)- Функция, которая осуществляет алгоритм алгоритм Ньютон-сопряженного градиента

Параметры:

dictionary: dict Словарь, со всеми переменными, которые были заданы в функции ввода
eps: float критерий остановки

func: str Функция в аналитическом виде в формате строки

3. **Алгоритм Ньютона-сопряженного градиента**

def inputSopr(): - Функция для ввода данных, с помощью которого будет осуществляться алгоритм Ньютон-сопряженного градиента

Параметры:

params: str Строка на вход с переменными, которые могуи быть заданы любой русской или английской буквой

F: str Функция в аналитическом виде в формате строки

def sopr(dictionary) - Функция, которая осуществляет алгоритм алгоритм Ньютон-сопряженного градиента

Параметры:

dictionary: dict Словарь, со всеми переменными, которые были заданы в функции ввода

eps: float критерий остановки

func: str Функция в аналитическом виде в формате строки

Т34 – Регрессия.

def input4()- функция для определения типа регрессии

Параметры:

r: int '1-линейная,2-полином,3-экспонента'
a: int 'количество столбцов в первом наборе данных'
b: int 'количество столбцов во втором наборе данных'

def regression(data)- функция для генерации данных и построения линейной или полиномиальной регрессии. Функция также строит полученный график регресии.

Параметры:
data: словарь с определением типа регрессии.

Т35 – Метод внутренней точки.

1. Решение задачи оптимизации для функции с ограничениями типа НЕравенства методом логарифмических барьеров.

def logBarMethod(func: str, restrictions: list, start_point: tuple = tuple(), accuracy:float = 10(-6), max_steps: int=500)** - Решение задачи оптимизации для функции методом логарифмических барьеров.

Параметры:

func : str Функция оптимизации.

equality : list Список заданных линейных ограничений.

x0 : tuple Начальная точка.

tol : int, default=5 Кол-во цифр после запятой после округления.

2. Решение задачи оптимизации для функции с ограничениями типа равенства методом Ньютона

def Newton(func: str, equality: list, x0: tuple, tol=5):

Параметры:

func : str Функция оптимизации.

equality : list Список заданных линейных ограничений.

x0 : tuple Начальная точка.

tol : int, default=5 Кол-во цифр после запятой после округления.

3. Решение задачи оптимизации для функции с ограничениями типа НЕравенства прямо-двойственным методом внутренней точки.

def inequality(func, x0, us, a, b, tol=10-5):**

Параметры:

func : str Функция для оптимизации.

x0 : list Начальная точка.

us : list Список заданных линейных ограничений. Варианты ввода: '4*x - 4 = 0' или '4*x - 4 < 0'

a : list Список коэффициентов для переменных в ограничении типа равенства.

b : list Список свободных членов для ограничения типа равенства.

tol : float, default=10**-5 Критерий остановы.

ТЗ6 – Классификация.

def logistic() - Функция, реализующая модель классификации на два класса основе логистической регрессии.

Параметры:

x_, y_: numpy array - Массивы обучающей переменной и предсказываемой

x_train, x_test, y_train, y_test: numpy array Массивы обучающей переменной и предсказываемой, разделенные на обучающую и тестовую выборки

y_pred: numpy array Массив предсказанных значений

def logistic_rbf() - Функция, реализующая модель классификации на два класса основе логистической регрессии с радиальной базисной функцией.

Параметры:

x_, y_: numpy array - Массивы обучающей переменной и предсказываемой

x_train, x_test, y_train, y_test: numpy array Массивы обучающей переменной и предсказываемой, разделенные на обучающую и тестовую выборки

y_pred: numpy array Массив предсказанных значений

def lin_with_l1() - Модель классификации на основе логистической регрессии с регуляризацией L1.

Параметры:

x_, y_: numpy array - Массивы обучающей переменной и предсказываемой

x_train, x_test, y_train, y_test: numpy array Массивы обучающей переменной и предсказываемой, разделенные на обучающую и тестовую выборки

y_pred: numpy array Массив предсказанных значений

def vvod() - Функция для ввода данных, с помощью которых будет проводиться классификация

Параметры:

X: numpy.ndarray Массив значений переменной x

y: numpy.ndarray - Массив значений переменной y

X_train: numpy.ndarray - Массив обучающей выборки переменной x

X_test: numpy.ndarray Массив тестовой выборки переменной x

y_train: numpy.ndarray Массив обучающей выборки переменной y

y_test: numpy.ndarray Массив тестовой выборки переменной y

def SVM(dictionary) - Функция, реализующая модель классификации на два класса на основе метода опорных векторов.

Параметры:

dictionary: dictionary Словарь со всеми переменными, которые ранее были введены в функции vvod

Т37 – Методы отсекающих плоскостей.

1. Функция, решающая задачу целочисленного линейного программирования методом отсекающих плоскостей (Метод Гомори) и Функцию, решающую задачу целочисленного линейного программирования методом полного перебора (Метод ветвей и границ).

   def main () - Функции для считывания введенной функции и ограничений и решающая задачу методом Гомори.
   Параметры:
   c: numpy.array Коэффициенты заданной функции.
   a: numpy.array Коэффициенты заданных ограничений
   b: numpy.array Коэффициенты свободных членов ограничений

   class SimplexMethod - Класс, реализующий симплекс-метод и дальнейшее решение по методу Гомори и методу ветвей и границ.
   def __init__() - Функция для инициализации всех переменных.

   def init_table() - Функция для инициализации таблицы.

   Параметры:

   table: numpy.array Таблица с коэффициентами.

   def get_negative_b_row():

   Параметры

   def get_negative_b_column ():

   Функция получения столбца с максимальным по модулю элементом в строке.

   Параметры:

   column: numpy.array Столбец с коэффициентами.

   def remove_negative_b () - Функция удаления отрицательных свободных коэффициентов.

   Параметры:

   column: numpy.array Столбец с коэффициентами.

   row: numpy.array Строка с коэффициентами.

   def gauss () - Функция выполняющая шаг метода Гаусса.

   Параметры:

table: numpy.array Таблица с коэффициентами.

def calculate_f() - Функция, считающая значения заданной функции.

Параметры:

table: numpy.array - Таблица с коэффициентами.

def get_relations () - Функция, делающая расчет симплекс отношений для столбца column.

Параметры:

q: list Список симплекс-отношений.

def solve () - Функция, решающая симплекс-таблицу и выводящая последовательность итераций.

Параметры:

iteration: int Номер итерации.

def part() - Функция, возвращающая дробную часть числа.

def make_gamori () - Функция, реализующая метод Гомори.

Параметры:

r_main: list

r_basis: list

task1_a: np.array()

task1_b: np.array

def solve_integer () - Функция, получающая целочисленные решения.

def print_table () - Функция, выводящая симплекс-таблицы на каждой итерации.

Т38 – Методы стохастической оптимизации.

1. **Функция, решающая задачу поиска экстремума функции методом стохастического градиентного спуска (SGD).**

def SGD_grad(X, Y) - Функция для нахождения градиента функции

Параметры:

X: numpy.ndarray Массив значений переменной x

Y: numpy.ndarray Массив значений переменной y

def stochastic_gradient_descent(max_epochs, xy_start, obj_func, grad_func) - Функция для нахождения стохастического градиента функции

Параметры:

max_epochs: int Максимальное количество итераций для запуска

xy_start: np.array Начальная точка, с которой начинается градиентный спуск

obj_func: function Ссылка на функцию, которая вычисляет целевую функцию

grad_func: function Ссылка на функцию, вычисляющую градиент функции

2. **Функция, реализующая модель классификации на два класса методом опорных векторов SVM с применением алгоритма градиентного спуска для минимизации функции ошибок (PEGASOS algorithm).**

def SVM(X_train : list, y_train : list, x_test : list, y_test: list, regularization = None , graph = False) - PEGASOS algorithm

Returns an array of predicted classes, an array of regression coefficients and a classification graph.

Параметры:

X_train : list Train data

y_train : list Classes for train data

x_test : tuple Test data

regularization : str Type of regularization.

graph : bool, default=False A parameter that determines whether a graph is needed