9/10 *9. Подсказки и эталонные ответы к заданиям модуля

1 задание

текущий урок

1. Введение

ЗАДАНИЕ 1.1

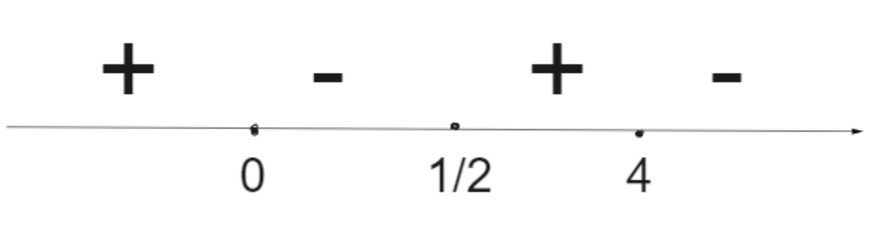
$$f^{\prime}(x)=\left(-x^{4}+6 x^{3}-4 x^{2}+80\right)^{\prime}_{x}=$$

$$=-4 x^{3}+18 x^{2}-8 x=0$$

$$-2 x^{3}+9 x^{2}-4 x=0$$

$$x\left(-2 x^{2}+9 x-4\right)=0$$

$$x=0 \quad x=4 \quad x=\frac{1}{2}$$



$$f(0)=80$$

$$f(4)=144$$

ЗАДАНИЕ 1.4

$$L(x, y, \lambda)=x^{2}+2 y^{2}+\lambda \cdot(x+y-20)$$

$$\left\{\begin{array}{l}L^{\prime}_{x}=2 x+\lambda=0 \\ L^{\prime}_{y}=4 y+\lambda=0 \\ L^{\prime}_{ \lambda}=x+y-20=0\end{arra$$

$$x=-\frac{\lambda}{2} \quad y= - \frac{\lambda}{4}$$

$$\frac{-\lambda}{2}-\frac{\lambda}{4}-20=0$$

$$\frac{-3 \lambda}{4}=20$$

$$x=\frac{40}{3} \quad y=\frac{20}{3}$$

$$x \approx 13 \quad y\approx7$$

$$13^{2}+2 \cdot 7^{2}=267$$

Вернуться в юнит 1 →

2. Градиентный спуск: применение и модификации

ЗАДАНИЕ 2.7

```
from sklearn.linear_model import SGDRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import seaborn as sns
import pandas as pd
import numpy as np

df = sns.load_dataset('diamonds')

df.drop(['depth', 'table', 'x', 'y', 'z'], axis=1, inplace=True)
df = pd.get_dummies(df, drop_first=True)

df['carat'] = np.log(1+df['carat'])
df['price'] = np.log(1+df['price'])

X_cols = [col for col in df.columns if col!='price']
X = df[X_cols]
y = df['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

parameters = {
    "loss": ["squared_error", "epsilon_insensitive"],
    "penalty": ["elasticnet"],
    "alpha": np.logspace(-3, 3, 15),
    "l1_ratio": np.linspace(0, 1, 11),
    "max_iter": np.logspace(0, 3, 10).astype(int),
    "random_state": [42],
    "learning_rate": ["constant"],
    "eta0": np.logspace(-4, -1, 4)
}

sgd = SGDRegressor(random_state=42)
sgd_cv = GridSearchCV(estimator=sgd, param_grid=parameters, n_jobs=-1)
sgd_cv.fit(X_train, y_train)

print(sgd_cv.best_params_)

sgd = SGDRegressor(**sgd_cv.best_params_, random_state = 42)

sgd.fit(X_train, y_train)
sgd.score(X_train, y_train) # r2
ls = sgd.predict(X_test)

round(mean_squared_error(y_test, ls), 3)
```

Вернуться в юнит 2 →

3. Метод Ньютона

ЗАДАНИЕ 3.1

```
def func1(x):
    return 6*x**5-5*x**4-4*x**3+3*x**2

def func2(x):
    return 30*x**4-20*x**3-12*x**2+6*x
init_value = 0.7
iter_count = 0
x_curr = init_value
epsilon = 0.000001
f = func1(x_curr)

while (abs(f) > epsilon):
    f = func1(x_curr)
    f_prime = func2(x_curr)
    x_curr = x_curr - (f)/(f_prime)
    iter_count += 1
    print(x_curr)
print(iter_count)
```

ЗАДАНИЕ 3.6

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} =$$

$$= 12 - \frac{(12)^3 - 72 \times 12 - 220}{3 \times (12)^2 - 72} = 12 - \frac{644}{390} = \frac{919}{90} \approx 10.211$$

$$\cdots$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} \approx 9.727$$

$$x_4 \approx 9.727$$

ЗАДАНИЕ 3.7

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} =$$

$$= x_n - \frac{x^2_n + 9x_n - 5}{2x_n + 9} = \frac{x^2_n + 5}{2x_n + 9}$$

ЗАДАНИЕ 3.9

```
def func(x):
    return 8*x**3-2*x**2-450
def func1(x):
    return 24*x**2 - 4*x
def func2(x):
    return 48*x -4

init_value = 42
iter_count = 0
x_curr = init_value
epsilon = 0.0001
f = func1(x_curr)

while (abs(f) > epsilon):
    f = func1(x_curr)
    f_prime = func2(x_curr)
    x_curr = x_curr - (f)/(f_prime)
    iter_count += 1
    print(x_curr)

print(round(x_curr, 3))
print(round(func(x_curr),3))
```

Вернуться в юнит 3 →

4. Квазиньютоновские методы

ЗАДАНИЕ 4.1

```
def func(x):
    return x[0] ** 4.0 - x[0] * x[1] + x[1] ** 2 + 9 * x[0] - 6 * x[1] \
        + 20

def grad_func(x):
    return np.array([2 * x[0] - x[1] + 9, -x[0] + 2 * x[1] - 6])

x_0 = [-400, -400]
result = minimize(func, x_0, method='BFGS', jac=grad_func)
solution = result['x']
print solution
```

ЗАДАНИЕ 4.4

```
from scipy.optimize import minimize

def func(x):
    return x[0]**2.0 - 3*x[0] + 45

def grad_func(x):
    return 2*x[0]-3

x_0 = 10
result = minimize(func, x_0, method='BFGS', jac=grad_func)
print('Статус оптимизации %s' % result['message'])
print('Количество оценок: %d' % result['nfev'])
solution = result['x']
evaluation = func(solution)
print('Решение: f(%s) = %.5f' % (solution, evaluation))
```

ЗАДАНИЕ 4.5

```
x_0 = 10
result = minimize(func, x_0, method='L-BFGS-B', jac=grad_func)
print('Статус оптимизации %s' % result['message'])
print('Количество оценок: %d' % result['nfev'])
solution = result['x']
evaluation = func(solution)
print('Решение: f(%s) = %.5f' % (solution, evaluation))
```

ЗАДАНИЕ 4.7

```
def func(x):
    return x[0]**4.0 + 6*x[1]**2.0 + 10

def grad_func(x):
    return np.array([4* x[0] ** 3, 12* x[1]])

x_0 = [100.0, 100.0]
result = minimize(func, x_0, method='BFGS', jac=grad_func)
print('Статус оптимизации %s' % result['message'])
print('Количество оценок: %d' % result['nfev'])
solution = result['x']
evaluation = func(solution)
print('Решение: f(%s) = %.5f' % (solution, evaluation))
```

Вернуться в юнит 4 →

5. Линейное программирование

ЗАДАНИЕ 5.5

$$\left\{\begin{aligned} 35 \leq x & \leq 45 \\ 40 \leq y & \leq 55 \\ x+y &=80 \\ \end{aligned}\right.$$

Максимизируем следующую функцию:

$$6 \cdot x+5.5 \cdot y$$

Получаем, что оптимальная точка: $x =40$ и $y = 40$.

$$6\cdot 40 + 5.5 \cdot 40 = 460$$

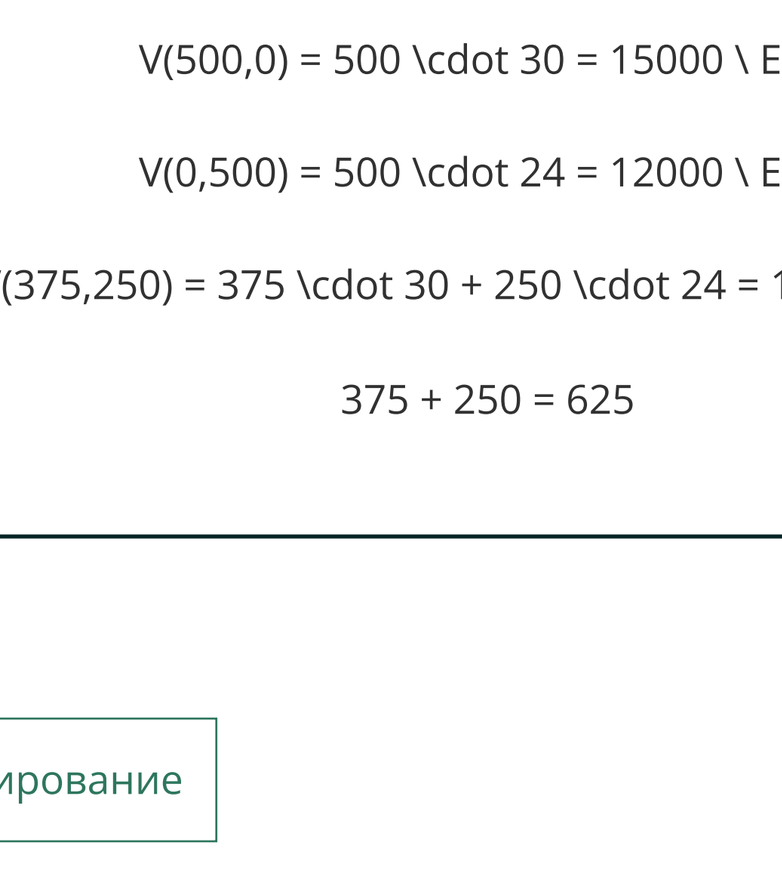
ЗАДАНИЕ 5.6

Обозначим за S количество произведённых свитшотов, а за C — количество произведённых рубашек. Запишем ограничения:

$$\begin{aligned} &S \geq 0 \\ &C \geq 0 \\ &S+1.5 C \leq 750 \\ &2 S+C \leq 1000 \end{aligned}$$

Максимизируем следующую функцию:

$$V(S, C)=30 S+24 C$$



$$V(0,0) = 0 \ EUR$$

$$V(500,0) = 500 \cdot 30 = 15000 \ EUR$$

$$V(0,500) = 500 \cdot 24 = 12000 \ EUR$$

$$V(375,250) = 375 \cdot 30 + 250 \cdot 24 = 17250 \ EUR$$

$$375 + 250 = 625$$

Вернуться в юнит 5 →

6. Практика: линейное программирование

ЗАДАНИЕ 6.1

Подробный разбор задания — в ноутбуке.

ЗАДАНИЕ 6.2

Подробный разбор задания — в ноутбуке.

ЗАДАНИЕ 6.3

Подробный разбор задания — в ноутбуке.

Вернуться в юнит 6 →

‹ Предыдущий урок Следующий урок ›