Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления   
Кафедра информационных технологий автоматизированных систем

Отчет по лабораторной работе № 7

по дисциплине «Современные системы компьютерного зрения»

**Разработка программы для решения задач классификации с использованием алгоритмов машинного обучения**

Проверил: Выполнил:

Навроцкий А.А. Студент гр.925901

Альромхин Джорж

Минск 2020

**Цель работы**:

Разработка программы для решения задач классификации с использованием алгоритмов машинного обучения

**Листинг кода: (Линейная регрессия)**

Набор данных **ex1data1.txt** представляет собой текстовый файл, содержащий информацию о населении городов (первое число в строке) и прибыли ресторана, достигнутой в этом городе (второе число в строке). Отрицательное значение прибыли означает, что в данном городе ресторан терпит убытки.

Сначала загружаем данные, затем строим график зависимости прибыли ресторана от численности населения города, в котором он расположен. реализуем функцию потерь J (θ) для набора данных ex1data1.формат txt.

Затем реализуем функцию градиентного спуска для выбора параметров модели. Постройте полученную модель (функцию) вместе с графом и построить трехмерный график зависимости функции потерь от параметров модели (θ0 и θ1) как поверхности, так и контуров (контурный график).

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from mpl\_toolkits.mplot3d import Axes3D  from matplotlib import cm  data\_row1 = np.genfromtxt('ex1data1.txt', delimiter=',')  data1 = pd.DataFrame(data\_row1, columns=list(['Population', 'Profit']))  data1.head()  plt.figure()  data1.plot(x='Population', y='Profit', style=['b.'])  plt.grid(True)  plt.show()  def compute\_cost(X, y, theta):  h = [np.matmul(x, theta.T).sum() for x in X]  return np.power(h - y, 2).sum() / (2 \* m)  m = data\_row1.shape[0] # Size of training set  n = data\_row1.shape[1] # Size of feature vector  X1 = data1[['Population']]  X1.insert(0, 'theta\_0', 1)  y1 = data1['Profit']  # theta the coefficients of the linear equation  theta = np.zeros((1, n))  X1.head()  cost = compute\_cost(X1.to\_numpy(), y1.to\_numpy(), theta)  print('Cost =', cost)  def gradient\_descent(X, y, theta, alpha, number\_iterations):  m = y.shape[0]  n = X.shape[1]  j\_history = []  for i in range(0, number\_iterations):  deltas = np.zeros(n)  for j in range(0, n):  xj = X[:, j]  h = [np.matmul(x, theta.T)[0] for x in X]  deltas[j] = ((h - y) \* xj).sum() \* alpha / m  theta[0] -= deltas  j\_history.append(compute\_cost(X, y, theta))  return theta, j\_history  iterations = 1500  alpha = 0.01  (theta, j\_history) = gradient\_descent(X1.to\_numpy(), y1.to\_numpy(), theta, alpha, iterations)  print('gradient descent thera: ', theta)  print('for population = 35,000, profit prediction: ', (np.matmul([1, 3.5], theta.T).sum() \* 10000)) #1st predict  print('for population = 70,000, profit prediction', (np.matmul([1, 7], theta.T).sum() \* 10000)) #2nd predict  h = [np.matmul(x, theta.T).sum() for x in X1.to\_numpy()]  data1\_plot = data1.join(pd.DataFrame({'Linear regression': h}))  plt.figure()  ax = data1\_plot.plot(x='Population', y='Profit', style=['b.'])  data1\_plot.plot(x='Population', y='Linear regression', ax=ax)  plt.grid(True)  plt.show()  theta0\_vals = np.linspace(-10, 10, num=100)  theta1\_vals = np.linspace(-1, 4, num=100)  # initialize J values to a matrix of 0  J\_vals = np.zeros((theta0\_vals.size, theta1\_vals.size))  # Fill J values  for i in range(0, theta0\_vals.size):  for j in range(0, theta1\_vals.size):  t = np.array([[theta0\_vals[i], theta1\_vals[j]]])  J\_vals[i, j] = compute\_cost(X1.to\_numpy(), y1.to\_numpy(), t)    J\_vals = J\_vals.T  fig = plt.figure()  ax = fig.add\_subplot(111, projection='3d')  surf=ax.plot\_surface(theta0\_vals,theta1\_vals,J\_vals,cmap="coolwarm")  fig.colorbar(surf, shrink=0.5, aspect=5)  ax.set\_xlabel(r'$\theta\_0$')  ax.set\_ylabel(r'$\theta\_1$')  ax.set\_zlabel(r'J($\theta$)')  ax.view\_init(30,120)#rotate for better angle  plt.show()  plt.figure()  ax = plt.contour(theta0\_vals, theta1\_vals, J\_vals, np.logspace(-2, 3, 20))  #plt.clabel(ax, inline=0, fontsize=2)  plt.xlabel(r'$\theta\_0$')  plt.ylabel(r'$\theta\_1$')  plt.plot(theta[0, 0], theta[0, 1], 'rx', linewidth=1, markersize=15)  plt.show() |

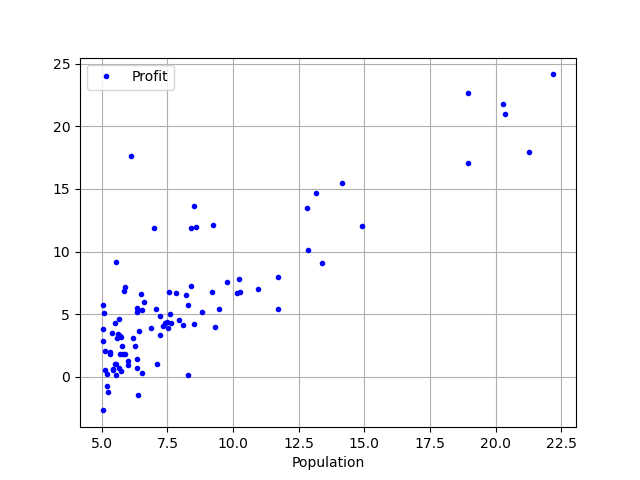


Рисунок 1.1 – график зависимости

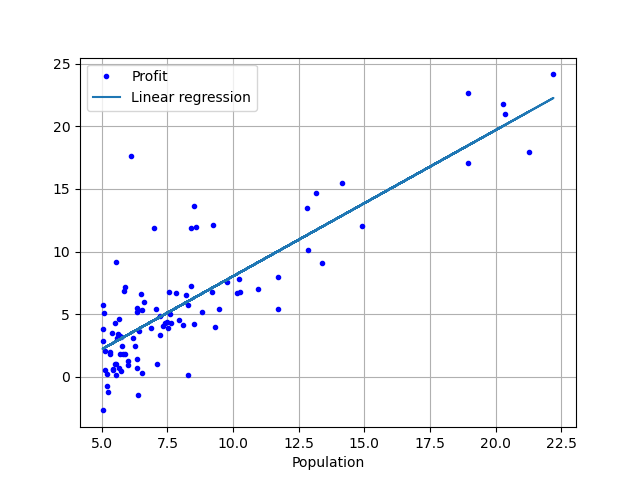


Рисунок 1.2 – график градиентного спуска

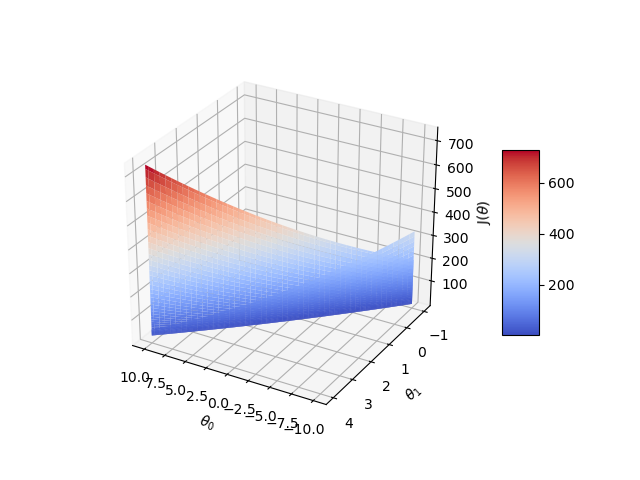


Рисунок 1.3 – трехмерный график зависимости функции потерь

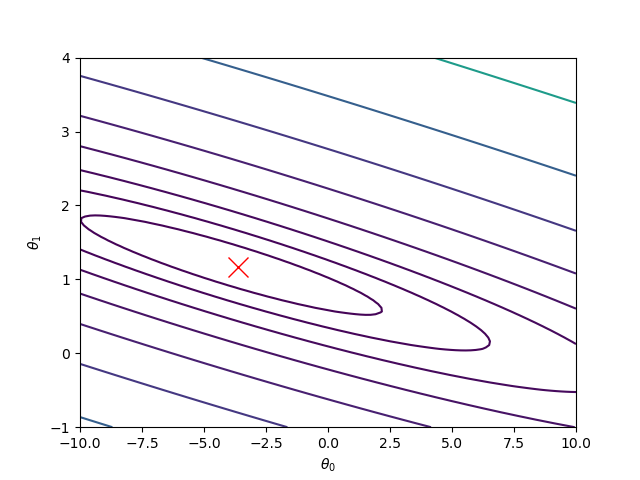


Рисунок 1.4 – контурный график зависимости функции потерь

**ЗАКЛЮЧЕНИЕ**

В лабараторной работе были рассмотрены случаи линейной регресии с помощью методов градиентного спуска.

В работе использовался язык программирования Python, интерактиваня среда разработки Jupyter а также библиотеки numpy, pandas и matplotlib