Лабораторная работа №4. Логистическая регрессия.

В данной работе вам предстоит реализовать логистическую регрессию. Для выполнения работы запустите редактор (Visual Studio Code, PyCharm или другие) и откройте файл lab04.py.

Вам необходимо использовать логистическую регрессию для построения модели предсказания поступления абитуриента в университет по данным оценок двух его экзаменов.

Файл ex2data1.txt содержит обучающий набор данных из трех столбцов:

– оценка абитуриента по первому экзамену (100 бальная система);

– оценка абитуриента по второму экзамену (100 бальная система);

– результат поступления (0 – не поступил, 1 – поступил).

Файл-заготовку lab04.py необходимо дополнить операторами и вычислениями до работающей программы. Когда вы правильно реализуете одно из заданий работы, можете переходить к следующему. Большинство последующих шагов работает на основе функций, сделанных на предыдущих шагах. Поэтому не стоит пропускать шаги и переходить к следующему, если в программе допущена ошибка, и она не работает как следует.

1. Загрузка и отображение данных

Первым делом в программе при помощи функции loadtxt загружаются данные из файла ex2data1.txt. Далее в переменную x помещаются первые два столбца из файла (оценки по первому и второму предмету), а в y – результат поступления.

Далее данные отображаются на графике в виде множества точек разного цвета.

Вам необходимо реализовать вычисление логистической функции. Найдите в программе функцию sigmoid (строка:

```
def sigmoid(z):
```

Далее, после комментария «добавьте свой код» дополните программу вычислением функции сигмоиды и занесением в переменную g, которая возвращается из функции. Теоретический материал для этого шага описан в разделе 3.2 учебного пособия и в лекции №4.

После того, как функция готова, запустите программу. В функции main вставлены вызовы sigmoid с определенными значениями вывод ожидаемых значений. Если эти значения совпадают, можно переходить к следующему шагу.

2. Функция стоимости и градиент

На этом шаге вам необходимо запрограммировать вычисление функции стоимости функции и вектора градиента для логистической регрессии. Теоретический материал для этого шага описан в разделах 3.4-3.5 пособия и в лекции №4.

Найдите в программе функцию cost_function. На вход она принимает вектор параметров *theta* (в виде одномерного массива размером n+1), матрицу X размером $m\times(n+1)$ с дополнительным столбцом единиц впереди), вектор y (в виде одномерного массива размером m), а возвращает значение функции стоимости $J(\theta)$. Дополните функцию необходимыми операторами для вычисления $J(\theta)$.

Далее найдите в программе функцию gradient_function. На вход она принимает вектор параметров *theta*, матрицу X и вектор y (все также, как и в cost_function), а возвращает вектор значений частных производных (градиент) в точке *theta* длиной n+1. Дополните функцию необходимыми операторами для вычисления градиента.

Когда функции готовы, запустите программу. В командном окне вы увидите

результаты вычисления функции стоимости и градиента для нулевого *theta*, а также ожидаемые значения, которые должны быть получены, если функции написаны правильно. Если ваши значения совпадают с ожидаемыми, переходите к следующему шагу.

3. Обучение логистической регрессии

На этом шаге вам не нужно писать собственных функций. Задача состоит в том, чтобы разобраться, как работать с библиотечными функциями для обучения моделей.

В прошлой лабораторной работе вам предлагалось самим реализовать алгоритм обучения линейной регрессии методом градиентного спуска. В данной работе мы воспользуемся библиотечной функцией оптимизации, входящей в библиотеку SciPy: minimize. Функция реализует мощные методы оптимизации и выбирает подходящий, анализируя входные параметры, которые ей передаются. В данном случае наиболее вероятно, что будет использоваться метод Бройдена – Флетчера – Гольдфарба – Шанно (BFGS). Параметрами функции можно переключать методы оптимизации, но мы не будем этого делать.

Подробно о всех возможностях функции можно посмотреть в документации (https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html).

Программными параметрами, которые мы передадим minimize, являются:

fun – функция, которую необходимо оптимизировать (функция стоимости);

jac – функция вычисления градиента в текущей точке (gradient_function);

x0 – начальные значения *theta* (нулевые);

args – дополнительные аргументы для fun, кроме *theta* (у нас передается *X* и *y*).

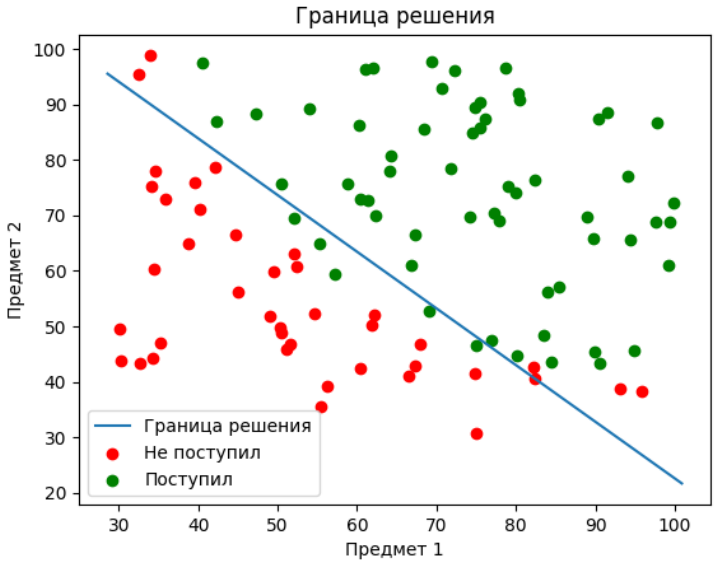
После выполнения вызова этой функции, она найдет минимальное значение функции стоимости и вернет свой результат работы в переменную min_res, в которой:

success – свидетельствует об успешном завершении минимизации (булевый);

message – текст сообщения об ошибке, если выполнение было неуспешным;

x – значение параметров, при которых найден минимум функции (наш искомый оптимальный *theta*).

Далее, для наглядности в программу добавлен код построения границы решения. Если все функции были написаны правильно, то вы должны получить график, похожий на следующий:



Можно переходить к следующему шагу.

4. Предсказание значений

На данном шаге мы воспользуемся обученной на предыдущем шаге гипотезой для предсказания шанса поступления абитуриента с известными оценками по экзаменам (но отсутствующего в обучающей выборке), а также оценим точность модели с использованием обучающей выборки.

Гипотеза логистической регрессии (см. раздел 3.2 учебного пособия и лекцию №4) в нашей задаче имеет вид:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

где $g(z)$ – функция сигмоиды.

Вы уже написали функцию сигмоиды, поэтому нам нужно передать в нее значение линейного многочлена: $\theta_0 + \theta_1 x_1 + \theta_2 x_2$, или в матричной форме: $X\theta$.

Результат гипотезы логистической регрессии принимает значения от 0 до 1 и может быть интерпретирован, как вероятность соотнесения входного X с классом y=1, то есть в нашем случае, если подать на вход абитуриента с оценками по экзаменами 45 и 85, мы получим в результате вероятность его поступления в университет. В NumPy это будет выглядеть так:

```
prob = sigmoid(np.array([1, 45, 85]).dot(theta))
```

(Опционально: Добавьте в программу еще несколько абитуриентов и оцените вероятности их поступления.)

Далее, используя обучающую выборку, оценим общую точность полученной в работе логистической модели, насколько правильно она работает.

При решении, к какому классу относится *x*, руководствуются правилом: если $h_\theta(x) \geq 0.5$, то *x* относится к классу 1, иначе к классу 0.

Вам необходимо дописать функцию predict, которая будет использовать полученные параметры логистической модели (*theta*), входные *X*, вычислять значение гипотезы и возвращать значения 0 или 1 (то есть не вероятность поступления, а класс абитуриента). Эта функция должна допускать на вход матричные параметры, то есть ей можно подать всю матрицу обучающей выборки *X* и вектор *theta*.

В программе уже заложен вызов этой функции и сравнения полученных ею результатов с реальными значениями *y*. Вычисляется среднее число совпадений, выводится в командном окне и сравнивается с ожидаемым результатом. Если функция написана правильно, то результаты должны быть близки.

На этом выполнение лабораторной работы №4 завершается.