Лабораторная работа №5. Регуляризованная логистическая регрессия.

В данной работе вам предстоит реализовать регуляризованную логистическую регрессию для построения модели предсказания качества изготовления микрочипа по результатам двух технологических тестов (Тест 1 и Тест 2).

Файл ex2data2.txt содержит обучающий набор данных из трех столбцов:

– значение первого теста для микрочипа;

– значение второго теста для микрочипа;

– качество (0 – бракованный чип, 1 – годный).

Откройте файл-заготовку lab05.py и дополните необходимыми операторами и вычислениями до работающей программы.

1. Загрузка и отображение данных

На первом шаге загружаются данные из файла ex2data1.txt. Далее в переменную *x* помещаются первые два столбца из файла (значения по первому и второму тесту), а в *y* – результат отбраковки.

Вам необходимо реализовать функцию plot_data для отображения множества точек обучающей выборки. Допишите ее таким образом, чтобы она рисовала график со значением первого теста чипа по горизонтальной оси, второго теста по вертикальной оси, качество чипа должно отображаться точками различных форм и/или цветов. Пример можно посмотреть в лабораторной работе №4.

2. Регуляризованная логистическая регрессия

На этом шаге мы построим гипотезу для сложной логистической регрессии в форме полинома шестой степени. Такой сложности модели должно быть достаточно, чтобы обучить классификатор для задачи предсказания брака микрочипа.

Исходный вектор характеристик чипа имеет вид:

$$x = [x_1 \quad x_2]$$

где x_1 – значение первого теста чипа, x_2 – значение второго теста чипа.

В результате конструирования модели мы получим следующий вектор:

$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ \dots \\ x_1 x_2^5 \\ x_2^6 \end{bmatrix}$$

В векторе 28 элементов. Полином 6-й степени имеет также 28 элементов в векторе $\theta = [\theta_0 \quad \dots \quad \theta_{27}]$.

Вам не нужно самостоятельно конструировать модель. В файле уже имеется готовая функция map_feature, которая из двух значений x_1, x_2 делает вектор из 28 элементов конструированной усложненной модели. Ознакомьтесь с кодом этой функции и разберитесь с принципом ее работы.

Далее вам необходимо запрограммировать вычисление функции стоимости функции и вектора градиента для регуляризованной логистической регрессии.

Теоретический материал для этого шага описан в разделах 4.5 пособия.

Допишите функцию sigmoid для вычисления логистической функции. В случае регуляризованной логистической регрессии она ничем не отличается от нерегуляризованной, поэтому можно воспользоваться решением из прошлой работы.

Допишите функцию cost_function. На вход она принимает вектор параметров *theta* (в виде одномерного массива размером n=28), матрицу X размером $m \times n$, вектор y (в виде одномерного массива размером m), параметр регуляризации *lamb*, а возвращает значение функции стоимости $J(\theta)$.

Далее допишите функцию gradient_function. На вход она принимает вектор параметров *theta*, матрицу X, вектор y и параметр регуляризации *lamb* (все также, как и в cost_function), а возвращает вектор значений частных производных (градиент) в точке *theta* длиной n.

Когда функции готовы, запустите программу. В командном окне вы увидите результаты вычисления функции стоимости и градиента для двух разных значений *theta*, а также ожидаемые значения, которые должны быть получены, если функции написаны правильно.

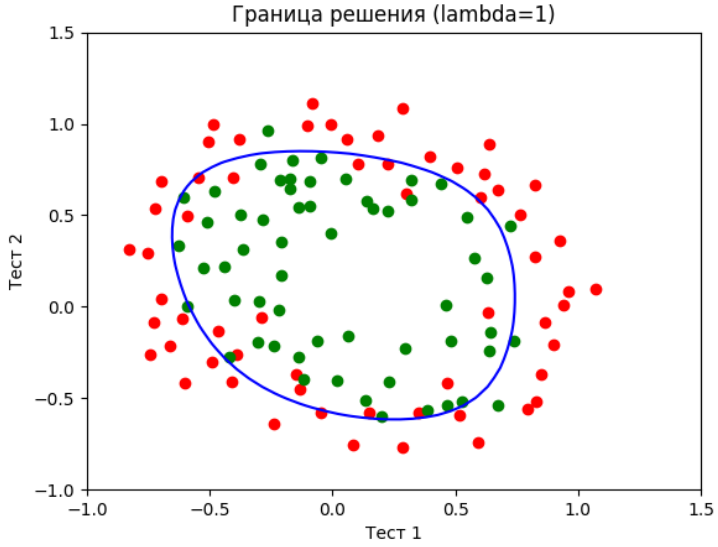
3. Обучение логистической регрессии

На этом шаге вам не нужно писать собственных функций. Для обучения регуляризованной логистической регрессии используется та же функция minimize из библиотеки SciPy, что и в прошлой лабораторной работе.

Отличием этой работы является то, что к дополнительным аргументам минимизируемой функции стоимости добавляется *lamb* (параметр регуляризации).

После минимизации функции стоимости в программе выводятся первые пять (из 28) элементов из оптимального вектора *theta* и приводятся ожидаемые значения.

Далее отрисовывается граница найденного решения. Если все функции были написаны правильно, то вы должны получить график, похожий на следующий:



4. Оценка точности модели

На данном шаге обученная ранее модель используется для предсказания брака чипа по заданным результатам двух технологических тестов, а также оценивается точность модели с использованием обучающей выборки.

Вам необходимо дописать функцию predict для правильного предсказания классов матрицы объектов. На вход функция predict принимает матрицу X размером

m×*n*, где *m* = 118 (объем обучающей выборки), *n* = 28 (количество признаков полинома степени 6 для двух переменных, которые генерируются функцией map_feature); вторым параметром функции predict является *theta* – вектор параметром обученной модели (длиной *n*). Таким образом, для вектора *X* размером *m*×*n* функция должна вернуть вектор у длиной *m* (результат 0 (брак) или 1 (годен) для каждой строки матрицы *X*).

После того, как функция predict готова, запустите программу. В командном окне выводятся результаты классификации для двух микрочипов и сравниваются с ожидаемыми значениями. Если все выполнено правильно, они должны совпадать.

Также программа вычисляет оценку точности классификации на обучающей выборке. Она тоже должна совпадать с ожидаемым значением.

5. Влияние регуляризации

На последнем шаге вам не придется дописывать никаких функций. С использованием уже готовых решений программа строит две модели и выводит графики их границ решений на обучающей выборке.

В первом случае запускается обучение модели с параметром *lambda* = 0 (отсутствие регуляризации). Вы должны получить переобученную модель. Во втором случае запускается обучение модели с параметром *lambda* = 100 (очень сильная регуляризация). Вы должны получить недообученную модель. Причины этого описаны в главе 4 учебного пособия.

Таким образом, в результате выполнения программы вы получите три различных графика границ решения на обучающей выборке. Определите, какой из графиков соответствует какому из случаев: оптимальная модель, переобученная модель, недообученная модель.

На этом выполнение лабораторной работы №5 завершается.