

## Практическая работа № 2. Логистическая регрессия

В этой работе Вы используете метод логистической регрессии для решения задач бинарной и множественной классификации. Строите границу решений. Наблюдаете явление переобучения и используете регуляризацию для борьбы с ним.

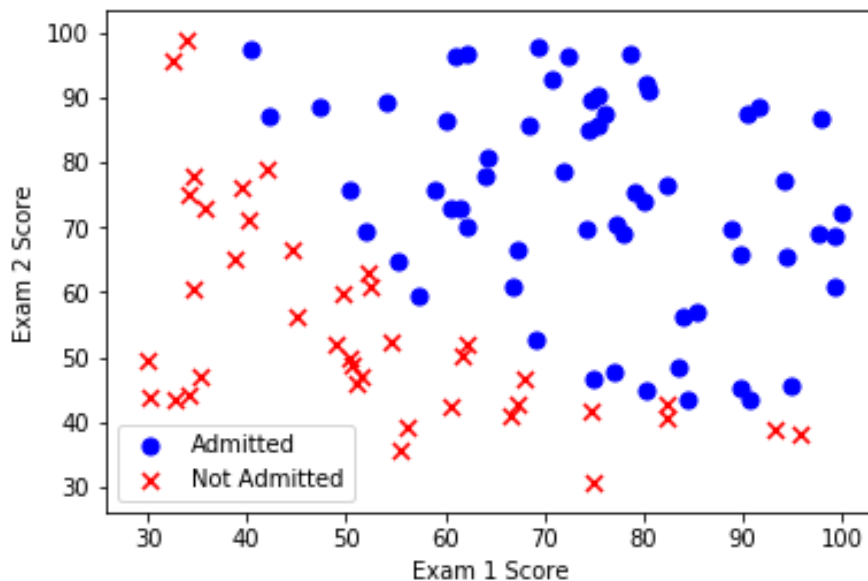
### 1. Бинарная классификация. Случай линейно разделимых классов

В этой работе Вы решаете задачу определения шанса абитуриента поступить в институт по известным результатам его двух экзаменов.

В качестве обучающего набора у Вас есть аналогичные данные прошлых лет абитуриентов. Для решения данной задачи бинарной классификации (не поступит-поступит) Вы будете использовать логистическую регрессию.

Исходные данные - в файле ex2data1.txt

**Загрузите таблицу с данными, отобразите на графике точки с двумя координатами (результаты двух экзаменов) и различного цвета в зависимости от того, поступил или нет данный абитуриент. У Вас должна получиться подобная картинка.**



Далее Вам нужно, как и в прошлой лабораторной работе,

**подготовить данные для дальнейшего обучения.**

Создайте две матрицы  $X$  и  $y$  со значениями входа и выхода элементов обучающей выборки. Не забудьте в  $X$  добавить первый столбец из единиц.

Создайте массив  $\theta$  (сколько в нем элементов?), инициализируйте нулевыми начальными значениями.

Для дальнейшей работы, нам нужна будет сигмоида - функция, возвращающая значение:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

**Напишите свою функцию `sigmoid(z)`.**

Также реализуем функцию `costFunction(theta, X, y)`, возвращающую значение целевой функции. Какая она в нашей задаче?

`costFunction(theta, X, y)` будет использовать написанную Вами ранее функцию `sigmoid(z)`.

**Реализуйте функцию `costFunction(theta, X, y)`. Проверьте корректность работы Ваших функций, посчитав значение целевой функции для данных  $X$ ,  $y$  и нулевых  $\theta$ . Должно получиться 0.693.**

Дальше мы могли бы взять функцию `gradientDescent(X, y, theta, alpha, iters)`, реализующую метод градиентного спуска из прошлой лабораторной работы. Почему?

Мы не будем этого делать, т.к. чистый метод градиентного спуска, реализованный в ней, работает медленно и практически уже не используется. Давайте воспользуемся его модификацией - готовой реализацией метода Ньютона сопряженных градиентов из пакета `Scipy`. Добавим импорт

```
import scipy.optimize as opt
```

Мы будем использовать `fmin_tnc` функцию, которая дает оптимальные значения  $\theta$  при данных  $X$  и  $y$ . Помимо этих входных значений, она также требует значения целевой функции и значения ее производной.

Поэтому добавим еще функцию `gradientFunc(theta, X, y)`: вычисляющую значение производной целевой функции.

**Реализуйте функцию `gradientFunc(theta, X, y)`. Проверьте корректность ее работы, посчитав значение производной целевой функции для данных  $X$ ,  $y$  и нулевых  $\theta$ . Должно получиться (-0.1, -12, -11).**

Далее запустим процесс минимизации целевой функции

```
result = opt.fmin_tnc(func = costFunction,  
                     x0 = theta, fprime = gradientFunc,  
                     args = (X, y))  
theta_optimized = result[0]  
print(theta_optimized)
```

и получим новые значения  $\theta$ . Оценим теперь качество полученного решения.

**Постройте границу решения. Что получилось?**

Для студента, сдавшего экзамены на 45 и 85 баллов, оцените вероятность поступления.

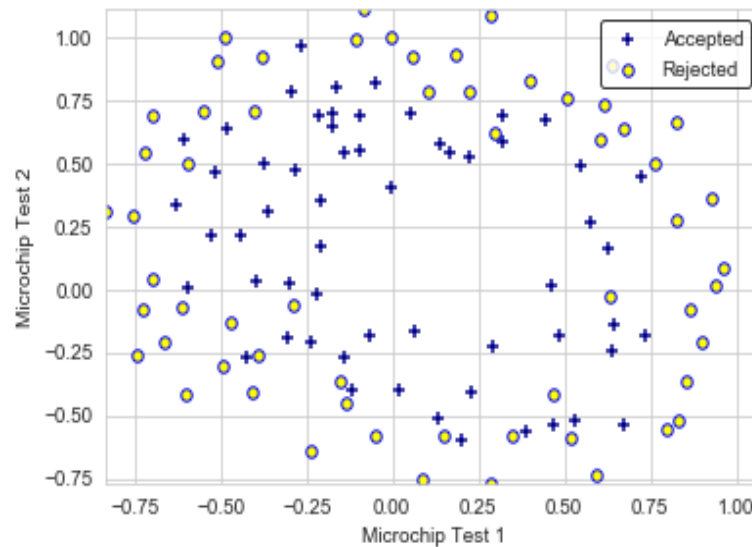
Оцените точность классификатора. Каков процент его правильных ответов?

## 2. Бинарная классификация. Случай не разделимых линейно классов

Рассмотрим далее более сложную задачу, когда классы не являются линейно разделимыми. Предположим, у Вас есть результаты двух тестов микрочипов, по которым Вы должны решить, бракованные они или стандартные, принимаете Вы их или отклоняете.

Исходные данные - в файле ex2data2.txt

**Загрузите и визуализируйте данные. У Вас должна получиться примерно такая картина.**



В прошлой задаче входной вектор для нашей модели имел вид:

$$X = (1 \quad x_1 \quad x_2)^T.$$

Граница решения в этом случае имела вид  $\theta^T X = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$  и являлась прямой. Ясно, что теперь, в отличии от прошлой задачи, границей решения (decision boundary) не может являться прямая линия.

Видно, что микрочипу, прошедшему проверку, соответствуют малые значения ошибок (+), «сидящие» в районе нуля. Бракованному микрочипу (o) – большие ошибки. Границей решения будет что-то вроде окружности с центром в нуле. Перед нами случай линейно не разделимых классов.

В этой задаче мы добавим больше входных признаков, являющихся многочленами до 6 степени от двух данных  $x_1, x_2$ , т.е. будем рассматривать следующий вектор признаков:

$$X = (1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_1 x_2 \quad x_2^2 \quad x_1^3 \quad \dots \quad x_1 x_2^5 \quad x_2^6)^T.$$

**Добавьте входные признаки.**

Можно написать код, который создаст такой вектор, самостоятельно. Можно воспользоваться готовой функцией `PolynomialFeatures` из пакета `sklearn` (<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html>).

Теперь, когда у нас есть такой входной вектор и мы можем получить достаточно сложную границу решений  $\theta^T X = 0$ , которой "под силу" разделить наши два класса, мы можем продолжить.

В этой задаче мы улучшим метод логистической регрессии добавлением регуляризации. Нас смущает огромное количество входных признаков (теперь их стало 28!) и мы всерьез опасаемся переобучения нашей модели. Чтобы избежать этого явления, добавим соответствующий член, отвечающий за регуляризацию, в наш алгоритм.

К целевой функции мы добавляем последнее слагаемое:

$$J(\theta) = - \left[ \frac{1}{m} \sum_{i=1}^m y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log (1 - h(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^m \theta_j^2.$$

Производная этого слагаемого добавится также к производной исходной целевой функции. Будьте внимательны, производная по  $\theta_0$  не меняется!

**Реализуйте функции `costFunctionR(theta, X, y, lam)` и `gradientFuncR (theta, X, y, lam)`, вычисляющие значения целевой функции и ее производной с учетом нового слагаемого. Проверьте корректность ее работы, посчитав значение производной целевой функции для данных X, y, нулевых  $\theta$  и  $\lambda = 1$ . Должно получиться**

**Cost: 0.693**

**Gradient: [8.47457627e-03 1.87880932e-02 7.77711864e-05 5.03446395e-02 1.15013308e-02 3.76648474e-02 1.83559872e-02 7.32393391e-03 8.19244468e-03 2.34764889e-02 3.93486234e-02 2.23923907e-03 1.28600503e-02 3.09593720e-03 3.93028171e-02 1.99707467e-02 4.32983232e-03 3.38643902e-03 5.83822078e-03 4.47629067e-03 3.10079849e-02 3.10312442e-02 1.09740238e-03 6.31570797e-03 4.08503006e-04 7.26504316e-03 1.37646175e-03 3.87936363e-02]**

Далее, как и в прошлой задаче, запускаем процесс минимизации целевой функции.

**Для оценки полученного решения постройте границу решений и посмотрите, насколько хорошо она разделила Ваши два класса.**

Давайте теперь пронаблюдаем явления недообучения и переобучения модели.

Для этого мы повторим уже проделанный для  $\lambda = 1$  процесс для значений  $\lambda = 0$  (регуляризации нет, мы увидим явление переобучения) и для  $\lambda = 100$  (все силы алгоритма брошены на минимизацию значений  $\theta$ , задача разделения классов не решается, мы увидим явление недообучения).

**Постройте границу решений и в этих случаях.**

Можете поэкспериментировать с другими значениями  $\lambda$ . Если  $\lambda = 0$  и регуляризации нет, как еще, кроме повышения значения  $\lambda$ , можно бороться с нежелательным явлением переобучения?

### 3. Использование логистической регрессии для решения задачи множественной классификации – распознавания рукописных цифр от 0 до 9

В этой задаче вам нужно решить задачу классификации рукописных цифр от 0 до 9 с помощью множественной логистической регрессии, а точнее метода *one-vs-all*.

Идея этого метода очень проста. Если у нас есть объекты  $K$  классов, мы строим  $K$  различных бинарных классификаторов, которые объекты определенного класса отделяют от всех остальных. Т.е. первый классификатор отделяет объекты первого класса от всех прочих ("не первого" класса), второй – второго и т.д. Теперь, когда у нас есть  $K$  таких классификаторов, для любого нового объекта мы можем вычислить вероятность его принадлежности к каждому из этих классов и выбрать тот класс, для которого это значение оказалось наибольшим.

В этой работе набор данных содержит объекты 10 классов – это рукописные цифры от 0 до 9. Поэтому Вам предстоит обучать 10 различных бинарных классификаторов.

**Загрузите данные из файла `ex2data3.txt`**

**Первые 400 столбцов – это "цифры"  $X$ , последний столбец – метки классов  $y$ . Отделите их.**

Набор данных содержит 5000 рукописных цифр. Каждая цифра была изначально gray scale картинкой 28x28 пикселей, которую затем "развернули" в строку из 784 элементов со значениями, характеризующими интенсивность данного пикселя.

Вот так выглядят некоторые из цифр нашего набора.



Все цифры теперь хранятся у Вас в массиве  $X$  размером 5000x400, а правильные ответы в  $y$  размером 5000x1.

**Добавьте, как обычно, к массиву X столбец из 1.**

Далее нам нужны все те же самые функции, что и в прошлой лабораторной работе: **sigmoid**, **costFunction**, **gradientFunc**, которые мы можем просто взять оттуда.

Мы также будем использовать **fmin\_tnc** функцию, которая дает оптимальные значения  $\theta$  при данных X и y. Помимо этих входных значений, она требует значения целевой функции и значения ее производной.

**Постройте 10 бинарных классификаторов, решающих задачи "цифра k" - не "цифра k", для  $k = 0, \dots, 9$ . У Вас должно получиться 10 наборов оптимальных значений  $\theta$ , которые разумно хранить в одном массиве (размером 10x401.)**

Осталось найти выходные значения каждого из 10 классификаторов

```
h=sigmoid(np.dot(X,theta.T)) # size (5000,10)
```

и выбрать класс с максимальным значением вероятности принадлежности объекта к нему

```
h_argmax = np.argmax(h, axis=1)
```

Тем самым, мы можем

**предсказать, к какому классу относится наш объект. Найдите долю правильных ответов Вашей модели.**

Итак, мы применили простой алгоритм логистической регрессии и метод one-vs-all к достаточно сложной задаче распознавания рукописных цифр и получили, как видите, очень неплохой результат.