Упражнение 2: Логистическая регрессия

Введение

В этом упражнении примените метод логистической регрессии к двум различным наборам данных: система сигнализации об исправности двигателя автомобиля в процессе эксплуатации и автомат по отбраковке микрочипов на производстве.

Описание файлов

ex2.m – поможет Вам при выполнении работы, не прибегая к проверке результатов в интернете
ex2_reg.m – скрипт Octave для остальных частей задания
ex2data1.txt – набор данных для первой части упражнения
ex2data2.txt - набор данных для второй части упражнения
submit.m – программа для автоматической проверки результатов вычислений
mapFeature.m – функция генерации новых полиномиальных свойств
plotDecisionBounday.m – построение границы разделения областей разнородных данных с привлечением классификатора
[*] *plotData.m* –2D построение данных для положительных и отрицательных значений
[*] *sigmoid.m* –Сигмоид-функция
[*] *costFunction.m* – Функция стоимости для логистической регрессии
[*] *predict.m* –Функция метода логистической регрессии
[*] *costFunctionReg.m* – регуляризованная функция стоимости метода логистической регрессии

* необходимо выполнить самостоятельно
В этом упражнении используются файлы *ex2.m* и *ex2_reg.m*. В них подготовлен сбор необходимых исходных данных, а также обращения к функциям, которые Вы напишите. Нет необходимости их изменения файлов *ex2.m* и *ex2_reg.m*.

1 Логистическая регрессия

Во время эксплуатации двигателя на автомобиле с использованием логистической регрессии можно оценить, исправен он или нет, при отклонении от нормального режима работы.
Можно сигнализировать о неисправности двигателя на основании всего двух величин – шума (или вибрации), неравномерности вращения ($(w_{max}-w_{min})/w_{cp}$). Вы располагаете предыдущими данными, которые здесь можно использовать. Для каждого двигателя этой серии определены неравномерность вращения и шум, по которым даны заключения о техническом состоянии (исправен или нет).
Ваша задача на основании двух оценок сформулировать модель классификации, в которой оценивается вероятность поломки двигателя в автомобиле во время движения. Для этого воспользуйтесь файлом-заготовкой *ex2.m*.

1.1 Построение данных

До начала выполнения заданий желательно представить данные в графическом виде. В первой части *ex2.m* для этого будет вызвана функция построения двухмерных графиков *plotData*.
Вам необходимо завершить программу *plotData.m* так, чтобы её результатом служил график

подобный рис.1, где оси соответствуют двум оценкам, а положительные или отрицательные результаты – маркерам разных цветов.

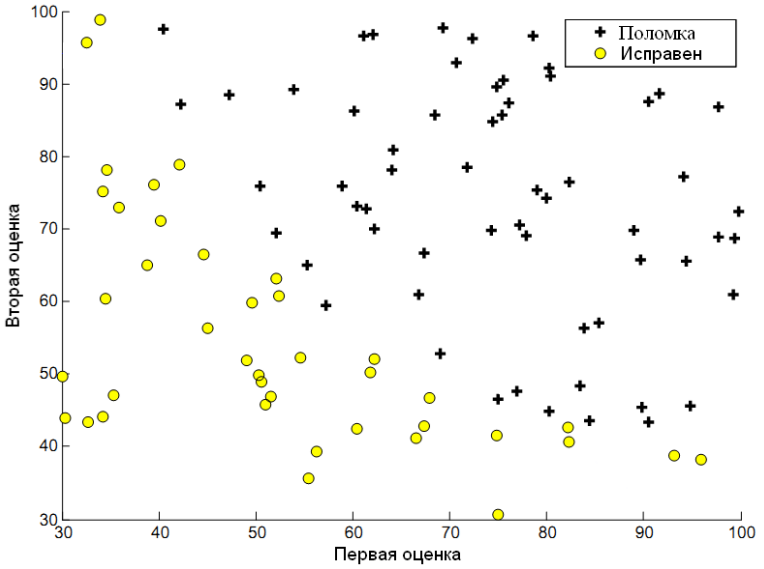

Рис.1: Точечный график исследуемых данных.

Для того, чтобы ознакомится с построением графиков, файл *plotData.m* оставлен пустым, так что у Вас есть возможность выполнить построение самостоятельно. Это задание останется без оценки. Кроме того, Вы можете воспользоваться уже решенной задачей приведенной ниже. В этом случае обязательно ознакомьтесь с каждой командой из документации Octave.

```
% Опрделение положительных или отрицательных индексов (Indices)
pos = find(y==1); neg = find(y == 0);

% Построение точек
plot(X(pos, 1), X(pos, 2), 'k+','LineWidth', 2, ...
'MarkerSize', 7);
plot(X(neg, 1), X(neg, 2), 'ko', 'MarkerFaceColor', 'y', ...
'MarkerSize', 7);
```

1.2 Выполнение

1.2.1 Сигмоид-функция[1]

До того, как приступать к написанию функции стоимости, определим гипотезу логистической регрессии:

$$h_\theta(x) = g(\theta^T x),$$

[1] Сигмоид-функция -это гладкая монотонная нелинейная возрастающая S-образная функция, которая часто применяется для "сглаживания" значений некоторой величины.

где функция *g* сигмоид-функция, которая определяется как:

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

Сначала следует описать эту функцию в файле *sigmoid.m* так, чтобы к ней могла обращаться остальная часть программы. Для проверки попробуйте вычислить несколько значений сигмоид-функции, написав в командной строке *sigmoid(x)*. Для больших положительных значений x, сигмоид должен быть близок к 1, при больших отрицательных – к 0. Значение *sigmoid(0)* должно быть равно 0,5. Программа должна работать также с векторами и матрицами. **Для работы с матрицами сигмоид-функция должна обрабатывать каждый элемент матрицы по отдельности.**
Для отправки на проверку, как и раньше, наберите в командной строке Octave команду *submit*.

Теперь можно отправить это пробное упражнение на проверку.

1.2.2 Функция стоимости и её градиент

Теперь можно завершить код функции в файле *costFunction.m* для вычисления стоимости и её градиента для метода логистической регрессии.
Функция стоимости для метода логистической регрессии определяется как:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right],$$

и её градиент это вектор θ, где j – ый элемент (для $j = 0, 1, \ldots, n$) определяется как:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Эта функция выглядит схоже с одноимённой функцией метода линейной регрессии, кроме функции гипотезы $h_\theta(x)$, которая определяется по-другому.
После этого программа *ex2.m* вызовет функцию *costFunction* с начальными параметрами θ. Значение функции стоимости должно быть примерно равным 0.693.

Теперь можно отправить функцию стоимости и её градиент для метода логистичекой регрессии на проверку в два этапа: один для функции стоимости, другой для её градиента.

1.2.3 Определение параметров θ с использованием функции *fminunc*.

В предыдущих заданиях Вы определяли оптимальные параметры метода линейной регрессии с использованием градиентного спуска: вычисляли функцию стоимости и затем её градиент. На этот раз, вместо вычисления градиента используйте встроенную в Octave функцию *fminunc*. Эта функция служит для нахождения минимума неограниченной[2] (unconstrained) функции. Для логистической регрессии Вам необходимо оптимизировать функцию стоимости $J(\theta)$ относительно параметров θ.

[2] Ограничения в оптимизации часто относится к ограничениям в параметрах, как, например, налагаемые ограничения на возможные значения параметров θ, которые они могут принимать (т.е., $\theta \leq 1$). Логистическая регрессия не содержит подобных ограничений, т.к. θ могут принимать любые вещественные значения.

А именно, используйте функцию *fminunc*, для нахождения наилучших параметров θ для функции стоимости метода логистической регрессии относительно Ваших данных (значений X и y). Функции *fminunc* будет иметь следующие входные данные:

- Начальные параметры для оптимизации
- Функцию, которая при подаче исследуемых данных (X , y) и конкретных θ, вычисляет логистическую регрессию и её градиент.

Программа *ex2.m* вызывает *fminunc* с нужными аргументами.

```
%        Настройка вызова функции fminunc
options = optimset('GradObj', 'on', 'MaxIter', 400);

%        Запуск fminunc для вычисления оптимальных theta
%        Эта функция вернёт величины theta и стоимости.
[theta, cost] = fminunc(@(t)(costFunction(t, X, y)), initial theta, options);
```

В приведённом листинге сначала определяются опции для использования *fminunc*. В частности включена опция *GradObj*, позволяющая функции *fminunc* возвращать оба значения: стоимость и градиент. Количество допустимых итерации установлено в 400 опцией *MaxIter*, после достижения которых функция завершается.

Функция для оптимизации обозначена строкой *@(t)(costFunction(t, X, y))*, что создаёт функцию с аргументом *t*, которая вызывает написанную Вами функцию *costFunction*.

В случае правильного выполнения функции *costFunction*, функция *fminunc* сойдётся с надлежащими параметрами оптимизации и возвратит окончательные значения стоимости и θ. Заметим, что использование *fminunc* не вызывает необходимости писать какие-либо циклы или устанавливать скорость обучения как в методе градиентного спуска. Всё это проделывается функцией *fminunc* самостоятельно – необходимо только указать функцию для отыскания стоимости и её градиента.

По завершении *fminunc*, программа *ex2.m* вызовет функцию *costFunction*, используя найденные оптимальные параметры θ. Стоимость должна быть около значения 0.203.

Окончательные значения θ будут использованы для построения границы на исследуемых данных сходно с рис.2. Посмотреть, как именно происходит построение с использованием параметров θ можно в *plotDecisionBoundary.m*.

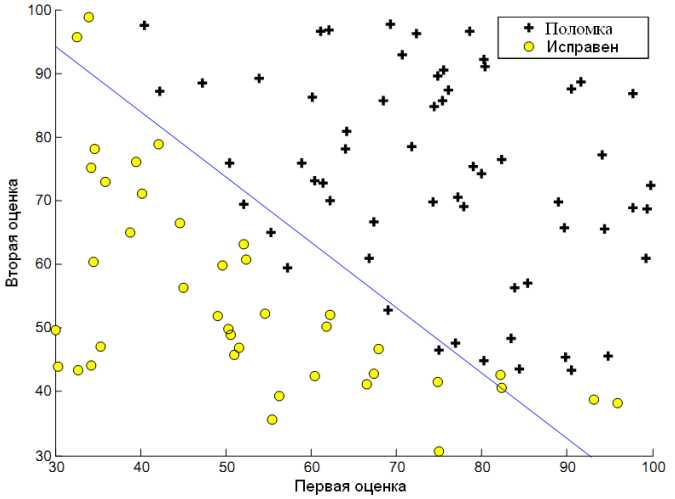

Рис.2: Исследуемые данные с обозначенной границей

1.2.4 Оценка логистической регрессии

После определения оптимальных параметров θ, можно оценить, исправен конкретный двигатель или нет. Для двигателя с условной шумностью 45% и неравномерностью вращения 85% следует ожидать результата с вероятностью поломки в 0.776.
Другой способ оценить качество полученных параметров, это оценить насколько верно они отображены. В этой части необходимо заполнить *predict.m*. Эта функция определит “1” или “0” для используемого набора данных и оптимального вектора θ.
После выполнения программы *predict.m*, скрипт *ex2.m* покажет точность вычислений обученной модели и найденных параметров θ, вычислив процент примеров, определённых правильно.

Теперь Вы можете отправить результаты вычислений для логистической регрессии.

2 Регуляризация логистической регрессии

В этой части упражнения необходимо применить регуляризованную логистическую регрессию для предсказания проходят ли микрочипы проверку качества или нет. Во время такой проверки каждый чип проходит несколько тестов для выявления каких-либо неполадок.
Вы консультант по вопросам качества крупного производства и располагаете данными результатов двух типов тестов для некоторых микрочипов, из которых необходимо определить, могут ли быть пропущены данные микросхемы или отбракованы. Для того, чтобы принять такое решение воспользуйтесь данными двух тестов и постройте модель логистической регрессии. Шаблон *ex2_reg.m* содействует в выполнении этой части.

2.1 Построение данных

Как и в предыдущих частях этого упражнения *plotData* используется для построения графика подобно рис.3, с двумя оценками по обеим осям и положительными (работоспособность) или отрицательными (брак) значениями оценочной функции.

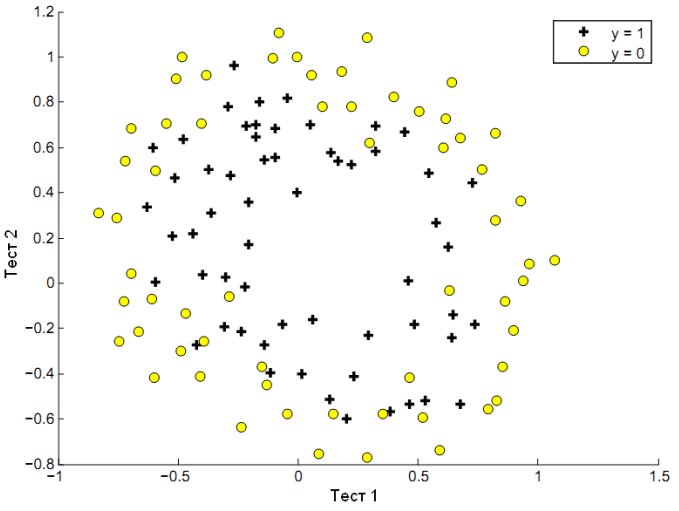

Рис.3: Построение исследуемых данных

На рис.3 показано, что исследуемые данные не могут быть разделены на положительные и

отрицательные ответы прямой линией. Поэтому буквальное применение метода логистической регрессии не подходит в данном примере, поскольку он подразумевает прямолинейную границу раздела двух областей.

2.2 Расширение свойств

Один из способов лучше оценить данные это создать больше свойств в каждой точке. Функция *mapFeature.m* позволяет расширить исследуемые свойства, увеличив степень полинома *(x_1, x_2)* до 6 степени.

$$\text{mapFeature}(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ \vdots \\ x_1 x_2^5 \\ x_2^6 \end{bmatrix}$$

В результате увеличения степени вектор, состоящий из двух свойств (двух оценок) был трансформирован в 28-размерный вектор. Классификатор логистической регрессии подобрал коэффициенты на этом высокоразмерном векторе свойств, имеющий более сложную нелинейную границу раздела между областями разнородных данных. В чём можно убедиться, построив её двухмерный график.

Увеличение количества свойств позволяет описать более выразительный классификатор, однако он склонен к резким изменениям границы раздела (*overfitting* или переобучение[3]). В следующих заданиях вы выполните регуляризованную логистическую регрессию для лучшей классификации с использованием оценочной функции. Вы обнаружите, как регуляризация помогает бороться с проблемой резкого изменения границы раздела (переобучения).

2.3 Функция стоимости и её градиент

Теперь завершим программу вычисления функции стоимости и её градиента для регуляризованной логистической регрессии. Для этого следует использовать файл *costFunctionReg.m*.

Напомним, что регуляризованная функция стоимости для логистической регрессии имеет вид:

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

Заметим, что не следует регуляризовать параметр θ_0; т.е. последнее суммирование в в этом

[3] В машинном обучении, переобучение (или *оверфиттинг*, *переподгонка*) — это явление сопровождающееся избытком шума, нежели отображением реальных процессов. При построении алгоритма обучения получается такой алгоритм, который слишком хорошо работает на примерах, участвовавших в обучении (т.е. на примерах из обучающей выборки), но достаточно плохо работает на примерах, не участвовавших в обучении (т.е. на примерах из тестовой выборки).

выражении выполняется от $j = 1$ до n, а не от $j = 0$ to n. Градиент функции стоимости это вектор, где его j-ый элемент определяется так:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \qquad \text{for } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\left(\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \lambda\theta_j\right) \qquad \text{for } j \geq 1$$

Завершив это, файл *ex2_reg.m* вызовет написанную Вами функцию *costFunctionReg* используя начальные нулевые значения θ. Ваш ответ должен быть примерно равен 0.693.

Теперь вы можете отправить функцию стоимости и её градиент регуляризованной логистической регрессии на проверку в два этапа: отдельно для функции стоимости и её градиента.

2.3.1 Подобор параметров используя функцию *fminunc*

Как и в предыдущих частях, для нахождения оптимальных параметров θ во второй части задания *ex2_reg.m* при правильном вычислении функции стоимости и её градиента регуляризованной логистической регрессии (*costFunctionReg.m*) используйте функцию *fminunc*.

2.4 Построение нелинейной границы разделения областей разнородных данных

После нахождения оптимальных параметров θ программа *ex2_reg.m* для визуализации обученной классификатором модели вызовет функцию *plotDecisionBoundary.m*, которая построит нелинейную границу разделения областей разнородных данных - отделит положительные и отрицательные примеры (см. рис.4).

2.5 Дополнительны задания (без оценки)

Требуется использовать несколько регуляризационных параметров λ для исследуемого набора данных с тем, чтобы выяснить влияние регуляризации на предотвращение быстрых изменений границы раздела (переобучение).

Обратите внимание на изменение границы раздела с изменением λ. Для малой λ классификатор верно определит принадлежность всех точек к своим типам, но граница раздела данных получится слишком извилистой (см. рис.5). Это не совсем правильная граница раздела, например, поучается, что точка $x = (-0.25, 1.5)$ принадлежит типу «положительно» (без брака: $y = 1$), что скорее неверно при известных предварительных данных.

С большим λ, график получится с упрощённой границей раздела, что также довольно хорошо разделяет положительные и отрицательные результаты. Однако если λ слишком велико, хороших предсказаний не получится, и граница не будет соответствовать действительности, следовательно, граница раздела излишне сглаживается (рис.6).

Нет необходимости отправлять данную часть задания на проверку, т.к. она не оценивается.

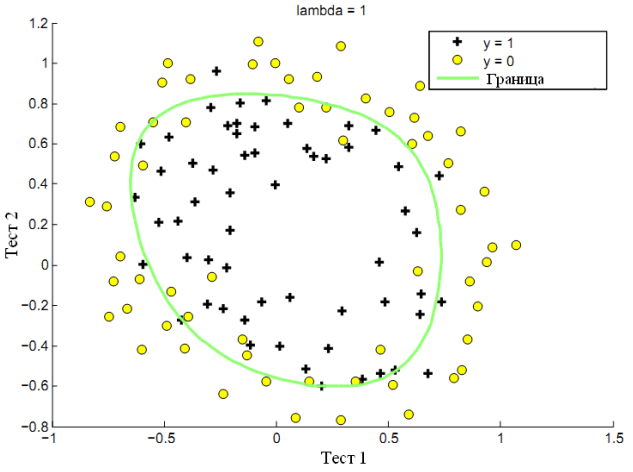

Рис.4: Пример надлежащей границы разделения разнородных данных

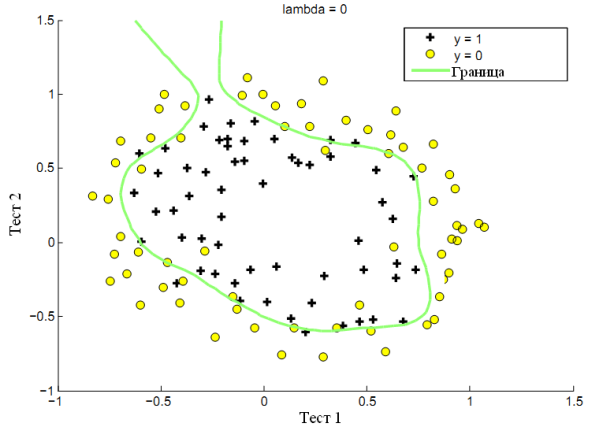

Рис.5: Без регуляризации (*Overfitting*) ($\lambda = 0$)

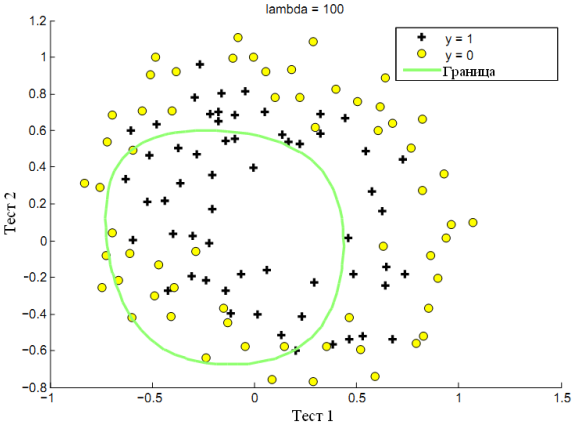

Рис.6: Пример слишком большой регуляризации (*Underfitting*[4]) ($\lambda = 100$)

Отправка результатов и проверка

После выполнения всех заданий не забудьте отправить их на проверку командой «*submit*». Ознакомьтесь с градацией оценок упражнений:

Часть	**Файл**	**Баллов**
Sigmoid Function	*sigmoid.m*	5
Compute cost for logistic regression	*costFunction.m*	30
Gradient for logistic regression	*costFunction.m*	30
Predict Function	*predict.m*	5
Compute cost for regularized LR	*costFunctionReg.m*	15
Gradient for regularized LR	*costFunctionReg.m*	15
Всего баллов		100

[4] *Underfitting*- противопроложность *overfitting*