МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ
Институт системной и программной инженерии
и информационных технологий (Институт СПИНТех)

Лабораторный практикум по курсу "Интеллектуальные системы" (09/22– 01/23)

Лабораторная работа 2

Предсказание вероятности возникновения события
по значениям множества признаков (логистическая регрессия) [1].

На этом занятии компьютерного практикума вы изучите *логистическую регрессию* и примените метод логистической регрессии к обработке двух различных наборов данных, а именно, данных, характеризующих систему сигнализации об исправности двигателя автомобиля в процессе эксплуатации, и данных, описывающих автомат по отбраковке микрочипов на производстве. Прежде чем приступить собственно к программированию, настоятельно рекомендуется ознакомиться с материалом лекций, а также с дополнительными материалами, имеющими отношение к задаче логистической регрессии и классификации.

Файлы, включенные в задание:

ex2.m - скрипт, реализующий пошаговое выполнение первой части задания по разделу «Логистическая регрессия»;
ex2_reg.m – скрипт, реализующий пошаговое выполнение второй части задания;
ex2data1.txt – набор данных для первой части упражнения;
ex2data2.txt – набор данных для второй части упражнения;
mapFeature.m – функция генерации новых полиномиальных свойств;
plotDecisionBounday.m - построение границы разделения областей разнородных данных с привлечением классификатора;
plotData.m (*) – 2D отображение данных для положительных и отрицательных значений;
sigmoid.m (*) – сигмоидная функция;
costFunction.m (*) – функция стоимости для логистической регрессии;
predict.m (*) – функция метода логистической регрессии;
costFunctionReg.m (*) – регуляризованная функция стоимости метода логистической регрессии.

Функции, отмеченные знаком *, следует написать самостоятельно.

В этой Лабораторной работе следует использовать скрипты *ex2.m* и *ex2_reg.m*, не изменяя их. В скриптах (коротких программах) подготовлены обращения к исходным данным. Далее

[1] Материал лабораторной работы составлен на основании аналогичного задания по курсу «Машинное обучение» на портале онлайн обучения Coursera.org (профессор Эндрю Ын, Стэнфордский университет - https://ru.wikipedia.org/wiki/Ын,_Эндрю)

производится вызов функций, написанных Вами, и отображаются результаты вычислений. Необходимо дописать функции в файлах по инструкциям упражнения.

1 Логистическая регрессия

Во время эксплуатации двигателя 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) и неравномерности вращения ($(w_{max}-w_{min}) / w_{cp}$). 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*.

Указание: Вы располагаете также написанными ранее (Лабораторная работа 1) программами, которые здесь можно использовать.

1.1 Построение данных

До начала выполнения заданий желательно представить данные в графическом виде. В первой части *ex2.m*, как и в Лабораторной работе 1, посвященной линейной регрессии, для этого будет вызвана функция построения двухмерных графиков *plotData*. Вам необходимо завершить программу *plotData.m* так, чтобы её результатом служил график, подобный приведенному на Рис. 1, где оси соответствуют двум оценкам, а положительные или отрицательные результаты – маркерам разных цветов.

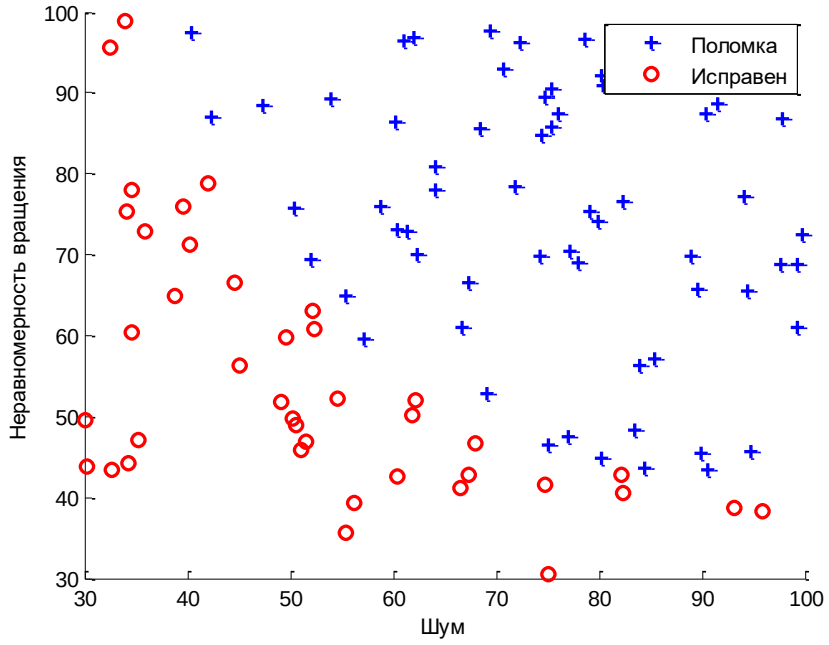

Рис. 1. Точечный график обучающих данных

Для того чтобы ознакомиться с построением графиков, файл *plotData.m* оставлен пустым, так что у Вас есть возможность выполнить построение самостоятельно. Кроме того, Вы можете воспользоваться уже ранее использовавшейся в Лабораторной работе 1 функцией.

Указание:

```
% Определение 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 Сигмоидная функция

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

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

где сигмоидная функция g , 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 сигмоидной функции, написав в командной строке `sigmoid(x)`. Для больших положительных значений x, значение сигмоидной функции должно быть близко к 1, а для высоких отрицательных – к 0. Значение функции *sigmoid.m* в нуле должно быть равно 0,5. Программа должна работать также с векторами и матрицами. Для работы с матрицами сигмоидная функция должна обрабатывать каждый элемент матрицы по отдельности.

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 относительно вектора θ, где j – й элемент (для j = 0, 1, . . . , 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 не отличается по виду от одноимённой функции в методе линейной регрессии, при этом функция, характеризующая логистическую гипотезу $h_\theta(x)$, определяется по-другому.

После этого программа *ex2.m* вызовет функцию *costFunction* с начальными параметрами θ. Значение функции стоимости должно быть примерно равным 0.693.

1.2.3 Определение параметров θ с использованием функции fminunc (fmincg)

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 MATLAB функцию *fminunc (fmincg)*. Эта функция служит для нахождения минимума неограниченной (*unconstrained*) функции.

Указание: 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.

Для логистической регрессии Вам необходимо оптимизировать функцию стоимости $J(\theta)$ относительно параметров θ. А именно, используйте функцию *fminunc (fmincg)*, для нахождения наилучших параметров θ для функции стоимости метода логистической регрессии относительно Ваших данных (значений X и y).

Функция *fminunc (fmincg)* 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 (fmincg)* с нужными аргументами.

```
% Настройка вызова функции fminunc (fmincg)
options = optimset('GradObj', 'on', 'MaxIter', 400);
% Запуск fminunc (fmincg)для вычисления оптимальных theta
% Эта функция вернёт величины theta и стоимости
[theta, cost] = fminunc(@(t)(costFunction(t, X, y)), initial theta,
options);
% Или
[theta, cost] = fmincg(@(t)(costFunction(t, X, y)), initial theta,
options);
```

В приведённом листинге сначала определяются опции для выполнения *fminunc (fmincg)*. В частности, включена опция *GradObj*, позволяющая функции *fminunc (fmincg)* возвращать оба значения: стоимость и градиент. Количество допустимого числа шагов итерационного процесса установлено равным *400* опцией *MaxIter*, после выполнения которых итерационный процесс завершается. Оптимизируемая функция обозначена строкой *@(t)(costFunction(t, X, y))*, что указывает на функцию с аргументом *t*, обращающуюся к написанной Вами функции *costFunction*. В случае правильного выполнения функции *costFunction*, итерационный процесс с надлежащими параметрами оптимизации сойдётся и возвратит окончательные значения стоимости и θ.

Заметим, что использование *fminunc (fmincg)* не вызывает необходимости писать какие-либо циклы или устанавливать скорость обучения как в методе градиентного спуска. Всё это проделывается функцией *fminunc (fmincg)* самостоятельно – необходимо только определить указатель на функцию для отыскания стоимости и её градиента. После завершения работы *fminunc (fmincg)*, программа *ex2.m* вызовет функцию *costFunction*, используя найденные оптимальные параметры θ. Стоимость должна быть равной ~ 0.203.

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

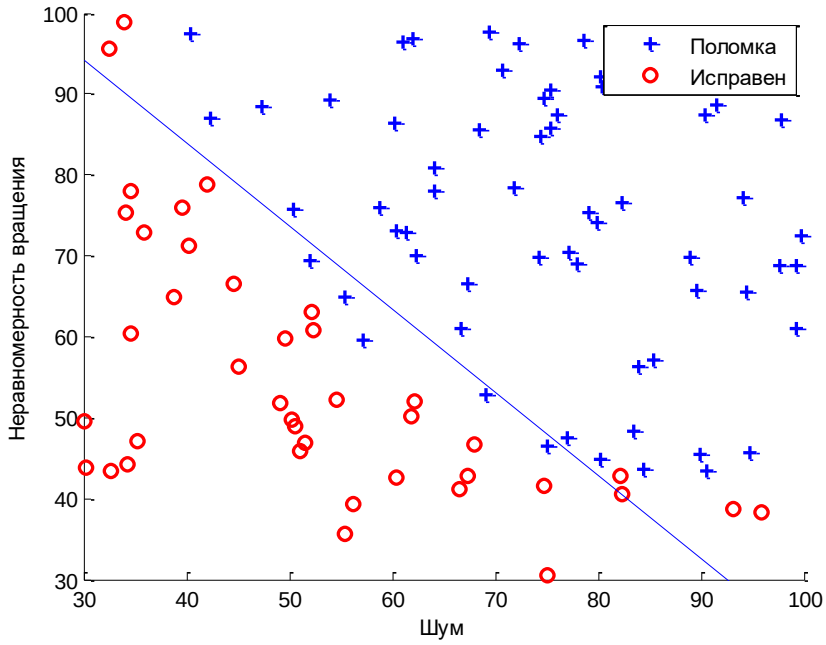

Рис. 2. Исследуемые данные с обозначенной границей между классами

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 запрограммировать функцию *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.

2 Регуляризация логистической регрессии

В этой части Лабораторной работы 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 Отображение 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.

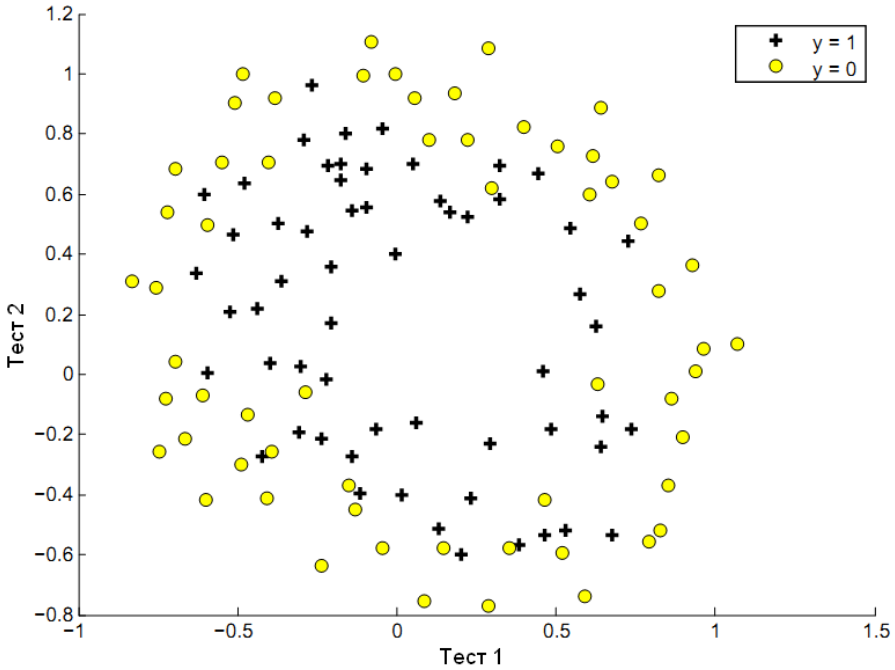

Рис. 3. Отображение исследуемых данных

На Рис. 3 показано, что исследуемые данные не являются *линейно сепарабельными* и, следовательно, не могут быть разделены на положительный и отрицательный классы прямой линией. Поэтому буквальное применение метода простой 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 (x1, x2) 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.

Указание: Установление размерности вектора является важным элементом для понимания задачи логистической регрессии и классификации в целом. Поясните полученную размерность.

Классификатор логистической регрессии подобрал коэффициенты на этом высокоразмерном векторе свойств, имеющем более сложную нелинейную границу раздела между областями разнородных данных. В этом можно убедиться, построив соответствующий двухмерный график. Увеличение количества свойств позволяет сконструировать более чувствительный классификатор, однако он склонен к резким изменениям границы раздела (*overfitting* или переобучение).

Указание: В машинном обучении в целом, и в теории нейронных сетей в частности, 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).

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 ее градиент

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

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 производить регуляризацию параметра θ_0, т.е. последнее суммирование в этом выражении выполняется от $j = 1 \ldots n$, а не от $j = 0 \ldots n$. Градиент функции стоимости - это вектор, каждый элемент j-й элемент которого определяется следующим образом:

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

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

Первое выражение соответствует случаю $j=0$, в то время как второе - $j \geq 1$.

С помощью файла *ex2_reg.m* следует вызвать написанную Вами функцию *costFunctionReg*, используя начальные нулевые значения θ. Ваш ответ должен быть примерно равен 0.693.

2.3.1 Подбор параметров θ с использованием функции fminunc (fmincg)

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 (fmincg)*.

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).

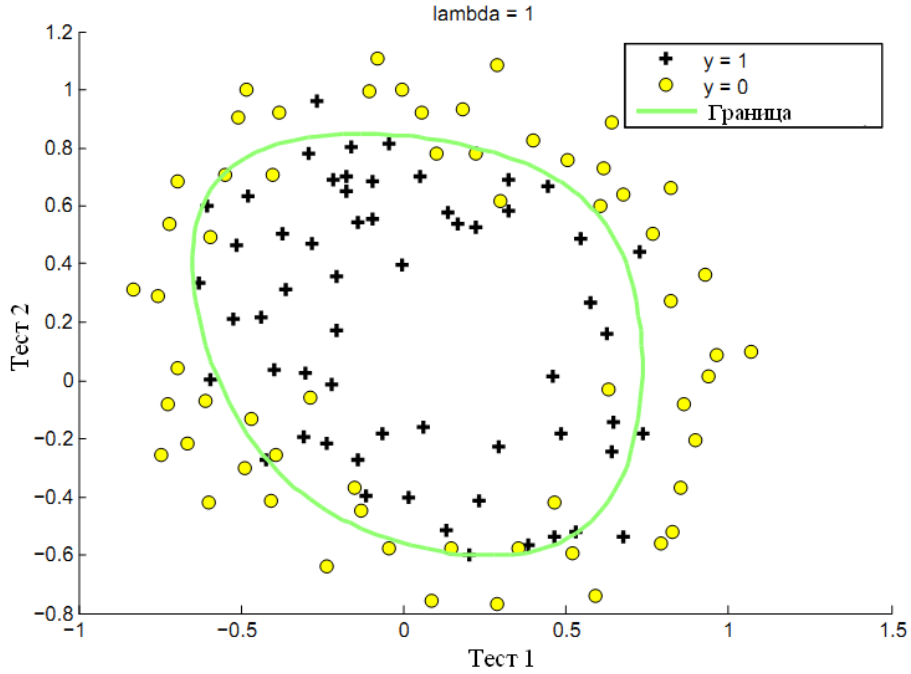

Рис. 4. Пример надлежащей границы разделения разнородных данных

2.5 Влияние регуляризации на предотвращение переобучения

Требуется использовать несколько регуляризирующих 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, получается, что точка x = (−0.25, 1.5) принадлежит типу «положительный» (без брака: y = 1), что скорее неверно при известных предварительных данных.

При высоком значении 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).

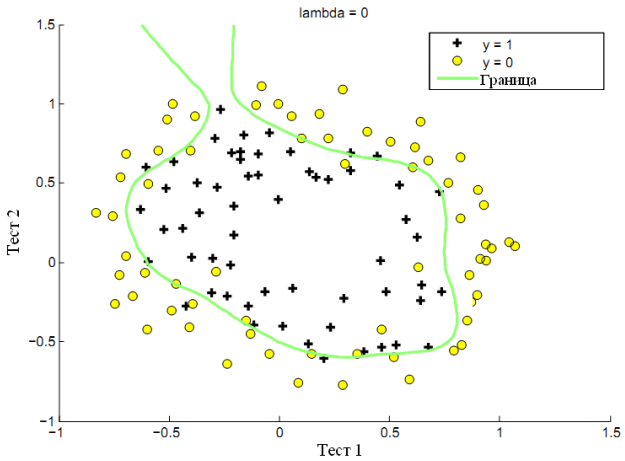

Рис. 5. Без регуляризации (*Overfitting*) ($\lambda = 0$)

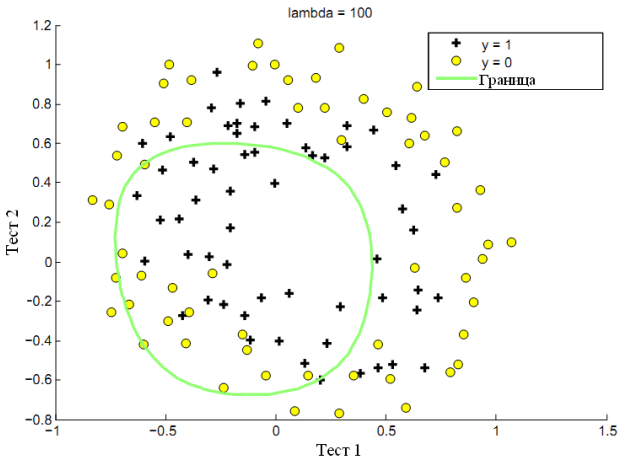

Рис. 6. Пример слишком высокой степени регуляризации (*Underfitting*) ($\lambda = 100$)