MATLAB Companion Script for *Machine Learning* ex2 (Optional)

Introduction

Coursera's *Machine Learning* was designed to provide you with a greater understanding of machine learning algorithms- what they are, how they work, and where to apply them. You are also shown techniques to improve their performance and to address common issues. As is mentioned in the course, there are many tools available that allow you to use machine learning algorithms *without* having to implement them yourself. This Live Script was created by MathWorks to help *Machine Learning* students explore the data analysis and machine learning tools available in MATLAB.

FAQ

Who is this intended for?

- This script is intended for students using MATLAB Online who have completed ex2 and want to learn more about the corresponding machine learning tools in MATLAB.

How do I use this script?

- In the sections that follow, read the information provided about the data analysis and machine learning tools in MATLAB, then run the code in each section and examine the results. You may also be presented with instructions for using a MATLAB machine learning app. This script should be located in the ex2 folder which should be set as your Current Folder in MATLAB Online.

Can I use the tools in this companion script to complete the programming exercises?

- No. Most algorithm steps implemented in the programming exercises are handled automatically by MATLAB machine learning functions. Additionally, the results will be similar, but not identical, to those in the programming exercises due to differences in implementation, parameter settings, and randomization.

Where can I obtain help with this script or report issues?

- As this script is not part of the original course materials, please direct any questions, comments, or issues to the *MATLAB Help* discussion forum.

Logistic Regression

In this Live Script, logistic regression models are implemented using the `fitglm` and `fitclinear` functions from the Statistics and Machine Learning Toolbox. A quick tutorial is also included on the *Classification Learner App*, which provides a graphical interface for creating classification models.

Files needed for this script

- ex2data1.txt - Training set for logistic regression with one variable
- ex2data2.txt - Training set for logistic regression with polynomial features

Table of Contents

MATLAB Companion Script for Machine Learning ex2 (Optional)....1
Introduction....1
FAQ....1
Logistic Regression....1
Files needed for this script....1
Logistic Regression with Two Variables....2
Load the data into a table and preview the data....2
Train the model using fitglm....6
Predict the training accuracy and probability of admission....7
Visualize the decision boundary....7
Using the Classification Learner App....8
Load the data....8
Open the app and select the variables....8
Select and train a classifier model....9
Evaluate the model....9
Export the model....9
Logistic Regression with Polynomial Features....10
Load the data....10
Fit a logistic regression model with polynomial features and interaction terms....13
Predict the test results and training accuracy....14
Visualize the model and decision boundary....15
Logistic Regression with Regularization....16
Load the data....16
Fit the model....16
Predict classes using the regularized model and plot the decision boundary....17
Local Functions:....18
plotMdlData....18

Logistic Regression with Two Variables

This section covers the MATLAB implementation of logistic regression in two dimensions corresponding to the first part of ex2. Recall that the file ex2data1.txt contains scores for two exams in addition to a binary variable which denotes whether students were admitted to a university. In this section we obtain a logistic regression model to predict admission probability using the `fitglm` function.

Load the data into a `table` and preview the data

Run the code below to load the data into a `table`. The first two columns (variables) will contain the exam scores and the third column the admission labels which we convert to `logical` values. We also compute some summary statistics on the three variables. After the table is displayed used the sort and filter controls (see the ex1 companion script for more information on `table` variables) to view only the scores of students that were admitted (`Admitted` = 'true') or denied. Are the results what you would expect?

```
clear;
data = readtable('ex2data1.txt');
data.Properties.VariableNames = {'Exam1','Exam2','Admitted'};
data.Admitted = logical(data.Admitted)
```

```
data = 100×3 table
```

	Exam1	Exam2	Admitted
1	34.6237	78.0247	0
2	30.2867	43.8950	0
3	35.8474	72.9022	0
4	60.1826	86.3086	1
5	79.0327	75.3444	1
6	45.0833	56.3164	0
7	61.1067	96.5114	1
8	75.0247	46.5540	1
9	76.0988	87.4206	1
10	84.4328	43.5334	1
11	95.8616	38.2253	0
12	75.0137	30.6033	0
13	82.3071	76.4820	1
14	69.3646	97.7187	1
15	39.5383	76.0368	0
16	53.9711	89.2074	1
17	69.0701	52.7405	1
18	67.9469	46.6786	0
19	70.6615	92.9271	1
20	76.9788	47.5760	1
21	67.3720	42.8384	0
22	89.6768	65.7994	1
23	50.5348	48.8558	0
24	34.2121	44.2095	0
25	77.9241	68.9724	1
26	62.2710	69.9545	1
27	80.1902	44.8216	1
28	93.1144	38.8007	0
29	61.8302	50.2561	0
30	38.7858	64.9957	0
31	61.3793	72.8079	1
32	85.4045	57.0520	1
33	52.1080	63.1276	0

	Exam1	Exam2	Admitted
34	52.0454	69.4329	1
35	40.2369	71.1677	0
36	54.6351	52.2139	0
37	33.9155	98.8694	0
38	64.1770	80.9081	1
39	74.7893	41.5734	0
40	34.1836	75.2377	0
41	83.9024	56.3080	1
42	51.5477	46.8563	0
43	94.4434	65.5689	1
44	82.3688	40.6183	0
45	51.0478	45.8227	0
46	62.2227	52.0610	0
47	77.1930	70.4582	1
48	97.7716	86.7278	1
49	62.0731	96.7688	1
50	91.5650	88.6963	1
51	79.9448	74.1631	1
52	99.2725	60.9990	1
53	90.5467	43.3906	1
54	34.5245	60.3963	0
55	50.2865	49.8045	0
56	49.5867	59.8090	0
57	97.6456	68.8616	1
58	32.5772	95.5985	0
59	74.2487	69.8246	1
60	71.7965	78.4536	1
61	75.3956	85.7599	1
62	35.2861	47.0205	0
63	56.2538	39.2615	0
64	30.0588	49.5930	0
65	44.6683	66.4501	0
66	66.5609	41.0921	0
67	40.4576	97.5352	1

	Exam1	Exam2	Admitted
68	49.0726	51.8832	0
69	80.2796	92.1161	1
70	66.7467	60.9914	1
71	32.7228	43.3072	0
72	64.0393	78.0317	1
73	72.3465	96.2276	1
74	60.4579	73.0950	1
75	58.8410	75.8584	1
76	99.8279	72.3693	1
77	47.2643	88.4759	1
78	50.4582	75.8099	1
79	60.4556	42.5084	0
80	82.2267	42.7199	0
81	88.9139	69.8038	1
82	94.8345	45.6943	1
83	67.3193	66.5894	1
84	57.2387	59.5143	1
85	80.3668	90.9601	1
86	68.4685	85.5943	1
87	42.0755	78.8448	0
88	75.4777	90.4245	1
89	78.6354	96.6474	1
90	52.3480	60.7695	0
91	94.0943	77.1591	1
92	90.4486	87.5088	1
93	55.4822	35.5707	0
94	74.4927	84.8451	1
95	89.8458	45.3583	1
96	83.4892	48.3803	1
97	42.2617	87.1039	1
98	99.3150	68.7754	1
99	55.3400	64.9319	1
100	74.7759	89.5298	1

```
summary(data)
```

```
Variables:

    Exam1: 100×1 double

        Values:

            Min        30.059
            Median     67.033
            Max        99.828

    Exam2: 100×1 double

        Values:

            Min        30.603
            Median     67.682
            Max        98.869

    Admitted: 100×1 logical

        Values:

            True        60
            False       40
```

Train the model using `fitglm`

Logistic regression models fall under a larger class of linear models referred to as *generalized linear models* in MATLAB. To train a generalized linear model, we use the `fitglm` function. Run the code in this section to train a logistic regression model on the exam data. The result is a `GeneralizedLinearModel` variable which contains all of the information about the model. Note that to obtain a *logistic* regression model from `fitglm`, we set the `Distribution` parameter to `binomial` as in the code below:

```
logMdl = fitglm(data,'Distribution','binomial')
```

```
logMdl = 
Generalized linear regression model:
    logit(Admitted) ~ 1 + Exam1 + Exam2
    Distribution = Binomial

Estimated Coefficients:
                   Estimate       SE        tStat       pValue  
                   ________    ________    _______    __________

    (Intercept)    -25.161       5.7986    -4.3392    1.4297e-05
    Exam1          0.20623     0.048001     4.2964    1.7357e-05
    Exam2          0.20147     0.048625     4.1434    3.4224e-05


100 observations, 97 error degrees of freedom
Dispersion: 1
Chi^2-statistic vs. constant model: 93.9, p-value = 4.07e-21
```

Note the form of the model displayed in the output above. This is short-hand for

$$\text{logit}(Admitted) = 1 * \theta_0 + Exam1 * \theta_1 + Exam2 * \theta_2$$

Since $\text{logit}(x)$ is the *inverse function* of $\text{sigmoid}(x)$, this model is equivalent to logistic regression model form for the probability of admission used in ex2:

$$Admitted = h_\theta(x) = \text{sigmoid}(\theta^T x),$$

where x includes the two exam scores and a bias term. As with the linear regression models trained in the ex1 companion script by `fitlm`, a bias term is added automatically by `fitglm`.

Predict the training accuracy and probability of admission

Recall that a prediction of admission corresponds to a predicted probability > 0.5. Run the code below to extract the θ values from the trained model, predict the probability of admission, and compute the training accuracy. Compare with your results from ex2.

```
theta = logMdl.Coefficients.Estimate
```

```
theta = 3×1
  -25.1613
    0.2062
    0.2015
```

```
% Predict the probability for a student with scores of 45 and 85
prob = predict(logMdl,[45 85]);
fprintf('For a student with scores 45 and 85, we predict an admission probability of %f
```

```
For a student with scores 45 and 85, we predict an admission probability of 0.776291
```

```
% Compute the training accuracy
Admitted = predict(logMdl,data) > 0.5;
fprintf('Train Accuracy: %f\n', mean(double(Admitted == data.Admitted)) * 100);
```

```
Train Accuracy: 89.000000
```

Visualize the decision boundary

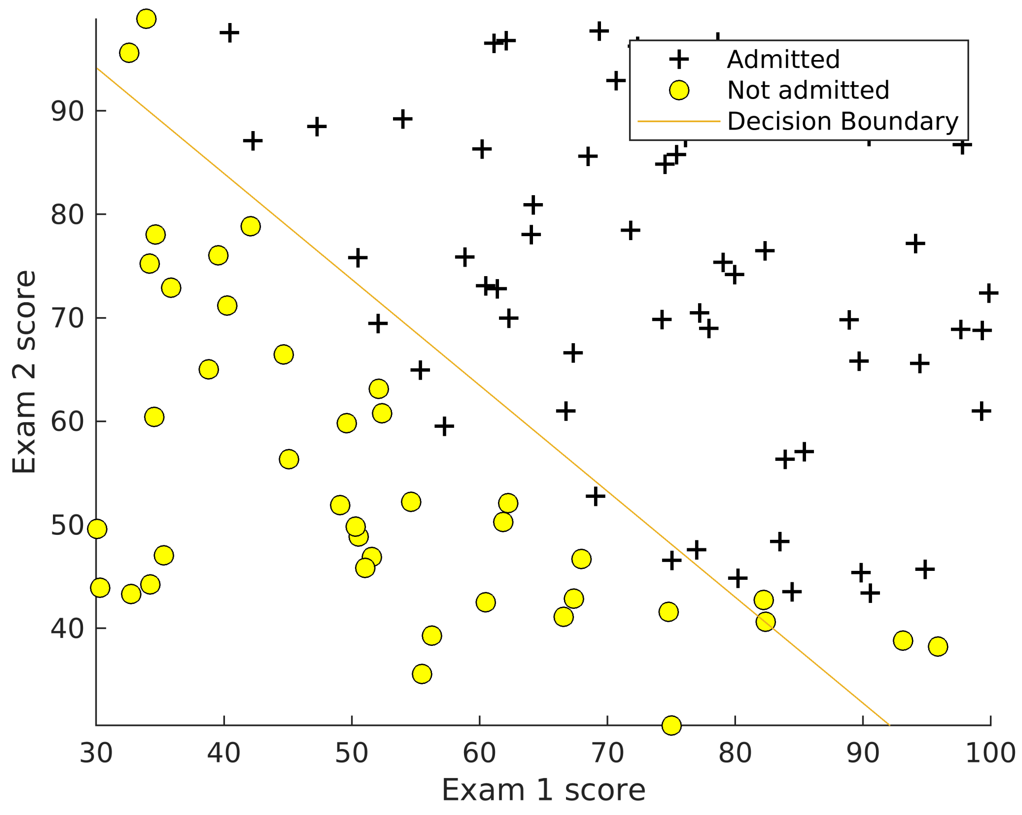
Run the code below to create a grid of exam scores and recreate the decision boundary plot from ex2. A local function, `plotMdlData`, has been included at the end of this script to create the plot.

```
figure; hold on;
% Plot the positive and negative examples
plotMdlData(data);

% Plot the decision boundary
xvals = [min(data.Exam1), max(data.Exam1)];
yvals = -(theta(1)+theta(2)*xvals)/theta(3);
plot(xvals,yvals); hold off;
ylim([min(data.Exam2),max(data.Exam2)]);

% Labels and Legend
xlabel('Exam 1 score')
ylabel('Exam 2 score')
```

```
legend('Admitted','Not admitted','Decision Boundary')
hold off;
```



Note: If you have difficulty reading the instructions below while the app is open in MATLAB Online, export this script to a pdf file which you can then use to display the instructions in a separate browser tab or window. To export this script, click on the 'Save' button in the 'Live Editor' tab above, then select 'Export to PDF'.

Using the Classification Learner App

In this section we provide the steps to reproduce the results of the previous section using the *Classification Learner App*. This app offers a graphical interface for building, training, and evaluating classification models.

Load the data

Run the code below to clear the workspace and reload the housing data. Then follow the instructions in the next few sections to create and train a logistic regression classifier using the app.

```
clear;
data = readtable('ex2data1.txt');
data.Properties.VariableNames = {'Exam1','Exam2','Admitted'};
```

Open the app and select the variables

1. In the MATLAB Apps tab, select the **Classification Learner** app from the Machine Learning section (you may need to expand the menu of available apps).
2. Select '**New Session -> From Workspace**' to start a new interactive session.

3. Under '**Data Set Variable'**, select '**data**' (if not already selected).
4. Under '**Response**' select '**From data set variable**' and '**Admitted**' (if not already selected).
5. Under '**Predictors**' select '**Exam1**' and '**Exam2**' (if already selected).
6. Under '**Validation**' select '**No Validation**'
7. Click the '**Start Session**' button.

Select and train a classifier model

There are many available classification models to choose from. In the model list the default model is 'Fine Tree'. To reproduce the logistic regression model obtained in the previous section:

1. Expand the model list and select '**Logistic Regression**' from the '**Logistic Regression Classifiers**' list.
2. Select '**Train**' to train the model.

Evaluate the model

After training there are several options available for evaluating the model's performance:

- The results, including training accuracy, prediction speed and training time for the selected model is shown in the '**Current Model**' pane.
- The model predictions including the predicted class and misclassified data points are visualized in the '**Scatter Plot**'. You can view the data points vs. the different predictor variables by selecting the desired variables for each axis from the '**Predictors**' list. (Exam1 and Exam2 are selected by default as there are only two predictors).
- The '**Confusion Matrix**', '**ROC Curve**', and '**Parallel Coordinates**' plots provide additional means of evaluating the model.

Export the model

Export and extract the trained model to the MATLAB workspace by following the steps below:

1. Select '**Export Model -> Export Model**'.
2. Select the default output variable name ('trainedModel').
3. Extract the linear model from the output variable by running the command below:

```
logMdl = trainedModel.GeneralizedLinearModel
```

```
logMdl = 
Generalized linear regression model:
    logit(zeroOneResponse) ~ 1 + Exam1 + Exam2
    Distribution = Binomial

Estimated Coefficients:
                   Estimate       SE        tStat       pValue  
                   ________    ________    _______    __________

    (Intercept)    -25.161       5.7986    -4.3392    1.4297e-05
    Exam1          0.20623     0.048001     4.2964    1.7357e-05
    Exam2          0.20147     0.048625     4.1434    3.4224e-05
```

```
100 observations, 97 error degrees of freedom
Dispersion: 1
Chi^2-statistic vs. constant model: 93.9, p-value = 4.07e-21
```

The `logMdl` variable now contains the logistic regression model which can be used in the same manner as the model previously created by `fitglm`.

Logistic Regression with Polynomial Features

In the second part of ex2, you implemented a regularized logistic regression classifier model to predict whether microchips pass quality assurance based on the scores from two tests. In this section we will obtain a corresponding model using `fitglm`.

Load the data

Run the code below to load the test data and results into a table with test score variables `Test1`, `Test2`, and the binary variable `Pass`.

```
clear;
data = readtable('ex2data2.txt');
data.Properties.VariableNames = {'Test1','Test2','Pass'};
data.Pass = logical(data.Pass)
```

```
data = 118×3 table
```

	Test1	Test2	Pass
1	0.0513	0.6996	1
2	-0.0927	0.6849	1
3	-0.2137	0.6923	1
4	-0.3750	0.5022	1
5	-0.5132	0.4656	1
6	-0.5248	0.2098	1
7	-0.3980	0.0344	1
8	-0.3059	-0.1923	1
9	0.0167	-0.4042	1
10	0.1319	-0.5139	1
11	0.3854	-0.5651	1
12	0.5294	-0.5212	1
13	0.6388	-0.2434	1
14	0.7368	-0.1849	1
15	0.5467	0.4876	1
16	0.3220	0.5826	1
17	0.1665	0.5387	1

	Test1	Test2	Pass
18	-0.0467	0.8165	1
19	-0.1734	0.6996	1
20	-0.4787	0.6338	1
21	-0.6054	0.5972	1
22	-0.6285	0.3341	1
23	-0.5939	0.0051	1
24	-0.4211	-0.2727	1
25	-0.1158	-0.3969	1
26	0.2010	-0.6016	1
27	0.4660	-0.5358	1
28	0.6734	-0.5358	1
29	-0.1388	0.5461	1
30	-0.2944	0.7800	1
31	-0.2656	0.9627	1
32	-0.1619	0.8019	1
33	-0.1734	0.6484	1
34	-0.2828	0.4729	1
35	-0.3635	0.3121	1
36	-0.3001	0.0270	1
37	-0.2367	-0.2142	1
38	-0.0639	-0.1849	1
39	0.0628	-0.1630	1
40	0.2298	-0.4116	1
41	0.2932	-0.2288	1
42	0.4833	-0.1849	1
43	0.6446	-0.1411	1
44	0.4602	0.0124	1
45	0.6273	0.1586	1
46	0.5755	0.2683	1
47	0.7252	0.4437	1
48	0.2241	0.5241	1
49	0.4430	0.6703	1
50	0.3220	0.6923	1
51	0.1377	0.5753	1

	Test1	Test2	Pass
52	-0.0063	0.3998	1
53	-0.0927	0.5534	1
54	-0.2079	0.3560	1
55	-0.2079	0.1732	1
56	-0.4384	0.2171	1
57	-0.2195	-0.0168	1
58	-0.1388	-0.2727	1
59	0.1838	0.9335	0
60	0.2241	0.7800	0
61	0.2990	0.6191	0
62	0.5063	0.7580	0
63	0.6158	0.7288	0
64	0.6043	0.5972	0
65	0.7655	0.5022	0
66	0.9268	0.3633	0
67	0.8232	0.2756	0
68	0.9614	0.0855	0
69	0.9384	0.0124	0
70	0.8635	-0.0826	0
71	0.8980	-0.2069	0
72	0.8520	-0.3677	0
73	0.8289	-0.5212	0
74	0.7944	-0.5577	0
75	0.5927	-0.7405	0
76	0.5179	-0.5943	0
77	0.4660	-0.4189	0
78	0.3508	-0.5797	0
79	0.2874	-0.7697	0
80	0.0858	-0.7551	0
81	0.1492	-0.5797	0
82	-0.1331	-0.4481	0
83	-0.4096	-0.4116	0
84	-0.3923	-0.2580	0
85	-0.7437	-0.2580	0

	Test1	Test2	Pass
86	-0.6976	0.0417	0
87	-0.7552	0.2902	0
88	-0.6976	0.6849	0
89	-0.4038	0.7069	0
90	-0.3808	0.9189	0
91	-0.5075	0.9042	0
92	-0.5478	0.7069	0
93	0.1031	0.7800	0
94	0.0570	0.9189	0
95	-0.1043	0.9920	0
96	-0.0812	1.1089	0
97	0.2874	1.0870	0
98	0.3969	0.8238	0
99	0.6388	0.8896	0
100	0.8232	0.6630	0

⋮

```
summary(data)
```

```
Variables:

    Test1: 118×1 double

        Values:

            Min          -0.83007
            Median    -0.0063364
            Max            1.0709

    Test2: 118×1 double

        Values:

            Min       -0.76974
            Median     0.21346
            Max         1.1089

    Pass: 118×1 logical

        Values:

            True        58
            False       60
```

Fit a logistic regression model with polynomial features and interaction terms

Recall that the different pass/fail outcomes could not be well separated by a logistic regression classifier using only the existing features. Instead, polynomial features up to the sixth power including interaction terms were created to capture the increased complexity of the data. Below we use the `fitglm` function to fit a logistic regression model including these polynomial features and interaction terms. Unlike your implementation in ex2, we *will not explicitly create these terms*. Instead we'll including an additional *model specification* parameter in the call to `fitglm`- see the documentation for more information about model specifications.

Run the code below to fit a logistic regression model using with polynomial features and note the form of the model returned.

```
logMdl = fitglm(data,'poly66','Distribution','binomial')
```

```
logMdl = 
Generalized linear regression model:
    logit(Pass) ~ 1 + Test1^2 + Test1*Test2 + Test2^2 + Test1^3 + (Test1^2):Test2 + Test1:(Test2^2) + Test
    Distribution = Binomial

Estimated Coefficients:
                       Estimate      SE        tStat       pValue  
                       ________    ______    _______    _________

    (Intercept)         38.231     16.055     2.3812     0.017255
    Test1               55.596     24.699     2.2509      0.02439
    Test2               98.147     47.061     2.0855     0.037021
    Test1^2            -369.43     152.54    -2.4219     0.015439
    Test1:Test2        -177.12     86.391    -2.0502     0.040343
    Test2^2            -194.26     95.562    -2.0328     0.042071
    Test1^3            -366.01     160.52    -2.2802     0.022596
    Test1^2:Test2      -842.21     370.41    -2.2737     0.022983
    Test1:Test2^2      -719.45     322.03    -2.2341     0.025474
    Test2^3            -511.89      261.6    -1.9568     0.050373
    Test1^4             1182.7     467.21     2.5314      0.01136
    Test1^3:Test2       1279.3     537.03     2.3823     0.017206
    Test1^2:Test2^2     1907.9     778.89     2.4495     0.014305
    Test1:Test2^3       914.32     401.54      2.277     0.022784
    Test2^4             514.28     256.18     2.0075     0.044698
    Test1^5             573.22     255.77     2.2411     0.025018
    Test1^4:Test2       1629.8     706.28     2.3075     0.021025
    Test1^3:Test2^2     2553.6     1104.9     2.3111     0.020828
    Test1^2:Test2^3     2919.1       1360     2.1464     0.031837
    Test1:Test2^4       1780.6     896.36     1.9865     0.046979
    Test2^5             785.32     429.76     1.8273     0.067648
    Test1^6            -1257.9     479.66    -2.6225    0.0087278
    Test1^5:Test2        -2260     900.07    -2.5109     0.012042
    Test1^4:Test2^2    -4142.8     1656.9    -2.5003      0.01241
    Test1^3:Test2^3    -4290.6     1789.1    -2.3982     0.016474
    Test1^2:Test2^4    -4229.7     1740.7    -2.4298     0.015105
    Test1:Test2^5      -2055.5     927.22    -2.2169      0.02663
    Test2^6            -750.38     375.09    -2.0005     0.045445


118 observations, 90 error degrees of freedom
Dispersion: 1
Chi^2-statistic vs. constant model: 112, p-value = 2.76e-12
```

Predict the test results and training accuracy

Next we use the `predict` function compute the probability of passing for the training examples to obtain the training accuracy. Again it is assumed that a probability > 0.5 corresponds to passing. As with the model training in the previous section, there is no need to map the original features to their polynomial counterparts for prediction as we can pass the original data directly to `predict`.

```
% Compute accuracy on our training set
Pass = predict(logMdl,data) > 0.5;
fprintf('Train Accuracy: %f\n', mean(Pass == data.Pass) * 100);
```

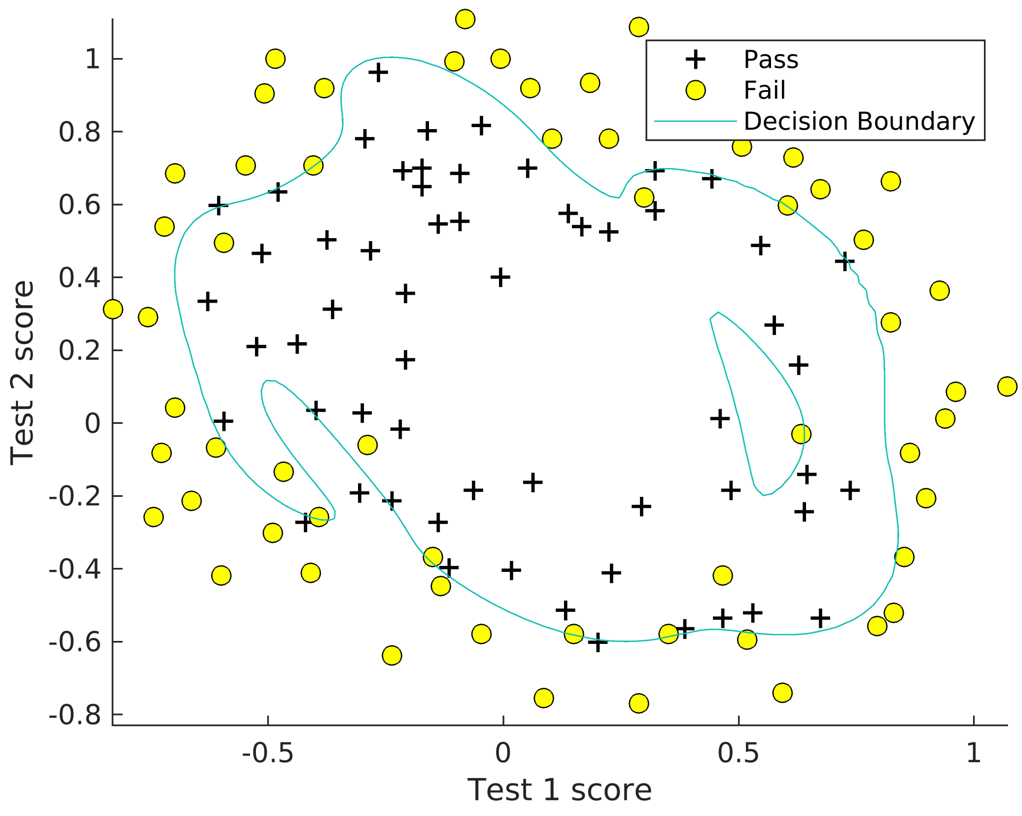
```
Train Accuracy: 88.983051
```

Visualize the model and decision boundary

Run the code in this section to plot the decision boundary and compare with your result from ex2 for $\lambda = 0$. The code below creates a grid of test scores, predicts the probability of passing for each pair, the uses `contour` to estimate the location of the decision boundary where $probability = 0.5$.

```
figure; hold on;
% Plot the positive and negative examples
plotMdlData(data);

% Plot the decision boundary
xvals = linspace(min(data.Test1), max(data.Test1));
yvals = linspace(min(data.Test2), max(data.Test2));
[X, Y] = meshgrid(xvals,yvals);
p = predict(logMdl,[X(:),Y(:)]);
contour(X,Y,reshape(p,size(X)),[0.5,0.5]); hold off;
% Labels and legend
xlabel('Test 1 score')
ylabel('Test 2 score')
legend('Pass', 'Fail','Decision Boundary')
```



Logistic Regression with Regularization

In this section, we use the `fitclinear` function train a logistic regression model *including regularization*. As the `fitclinear` function is generally used with *high dimensional data* (i.e. with lots of variables- like our polynomial feature model) where storing data in `table` variables makes less sense, it takes training data in form of numeric matrices instead.

Load the data

Run the code below to load the data into the feature matrix X and response vector y. We then also create the polynomial feature matrix, `Xpoly`.

```
clear;
X = load('ex2data2.txt');
y = X(:,3);
X(:,3) = [];
% Create the polynomial feature matrix up to power 6
powers = [nchoosek(0:6,2); fliplr(nchoosek(0:6,2));1 1;2 2;3 3]';
powers(:,sum(powers)>6) = [];
Xpoly = (X(:,1).^powers(1,:)).*(X(:,2).^powers(2,:));
```

Fit the model

Next, we train the model using `fitclinear` with the regularization type set to `ridge` (this is the type of regularization used in ex2) and the strength given by `lambda`. The result is a `ClassificationLinear` model variable which contains all of the information about the model. The model coefficients are found in the `Bias` and `Beta` properties of the model variable. Use the control select a value of λ, then examine the effect on the training accuracy and decision boundary from the results in the next two sections:

```
% Choose lambda and train the model
lambda = 0.001;
logMdl = fitclinear(Xpoly,y,'Lambda',lambda,'Learner','logistic','Regularization','ridg
```

```
logMdl = 
  ClassificationLinear
      ResponseName: 'Y'
        ClassNames: [0 1]
    ScoreTransform: 'logit'
              Beta: [27×1 double]
              Bias: 2.6494
            Lambda: 1.0000e-03
           Learner: 'logistic'


  Properties, Methods
```

```
logMdl.Bias
```

```
ans = 2.6494
```

```
logMdl.Beta
```

```
ans = 27×1
    2.8208
   -3.9911
   -0.4335
   -2.6491
   -0.1729
   -1.1961
   -0.5235
   -1.1186
   -1.1088
   -0.8549
      ⋮
```

Predict classes using the regularized model and plot the decision boundary

The `predict` function is used below to classify data using the `ClassificationLinear` model variable in the same manner as with `GeneralizedLinearModel` variables. However, the `predict` function will return a *class label* instead of probability score. If needed, we obtain the probaibility scores by requesting a second output from `predict`. See the code below used to plot the decision boundary. When you are done, try re-training the model using a different value of `lambda` and examine the effects. Which model do you think will generalize the best?

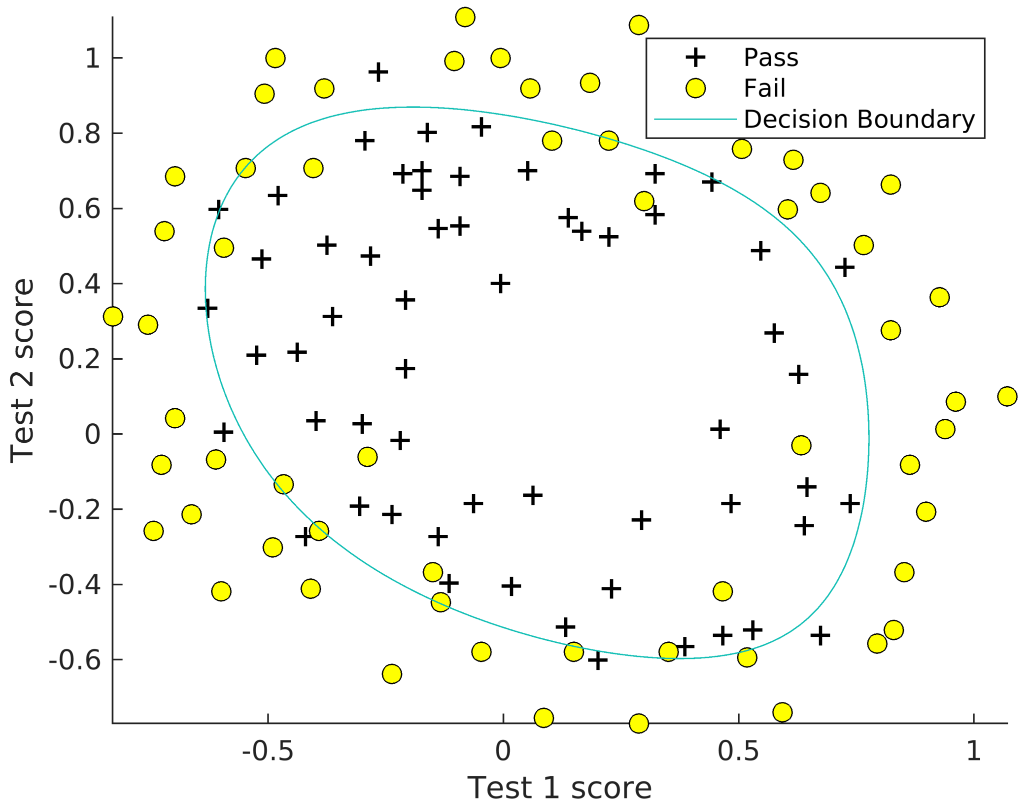
```
% Obtain the class labels and compute the training accuracy
Pass = predict(logMdl,Xpoly);
fprintf('Train Accuracy: %f\n', mean(Pass == y) * 100);
```

```
Train Accuracy: 83.050847
```

```
% Plot the positve and negative examples
```

```
figure; hold on;
plotMdlData(array2table([X y],'VariableNames',{'Test1','Test2','Pass'}));

% Plot the decision boundary
xvals = linspace(min(X(:,1)), max(X(:,1)));
yvals = linspace(min(X(:,2)), max(X(:,2)));
[Xgrid, Ygrid] = meshgrid(xvals,yvals);
Xpolygrid = (Xgrid(:).^powers(1,:)).*(Ygrid(:).^powers(2,:));
[~,Score] = predict(logMdl,Xpolygrid); % Obtain the probability scores
contour(Xgrid,Ygrid,reshape(Score(:,2),size(Xgrid)),[0.5,0.5]); hold off;
% Labels and legend
xlabel('Test 1 score')
ylabel('Test 2 score')
legend('Pass', 'Fail','Decision Boundary')
```



Local Functions:

plotMdlData

`plotMdlData` is used to plot the positive and negative examples for the data sets for better comparison with the plots in ex2.

```
function [] = plotMdlData(data)
% Reproduce the plots from ex2 with positive and negative results for an input table
% Extract variable names from 3 column table
varNames = data.Properties.VariableNames;
```

```
% Plot the data with + for true and 0 for false examples
inds = data.(varNames{3}) == 1;
plot(data.(varNames{1})(inds), data.(varNames{2})(inds), 'k+','LineWidth', 2, 'MarkerSi
inds = data.(varNames{3}) == 0;
plot(data.(varNames{1})(inds), data.(varNames{2})(inds), 'ko', 'MarkerFaceColor', 'y','
end
```