

1.1 Visualizing the data

Before starting to implement any learning algorithm, it is always good to visualize the data if possible. In the first part of `ex2.m`, the code will load the data and display it on a 2-dimensional plot by calling the function `plotData`.

You will now complete the code in `plotData` so that it displays a figure like Figure 1, where the axes are the two exam scores, and the positive and negative examples are shown with different markers.

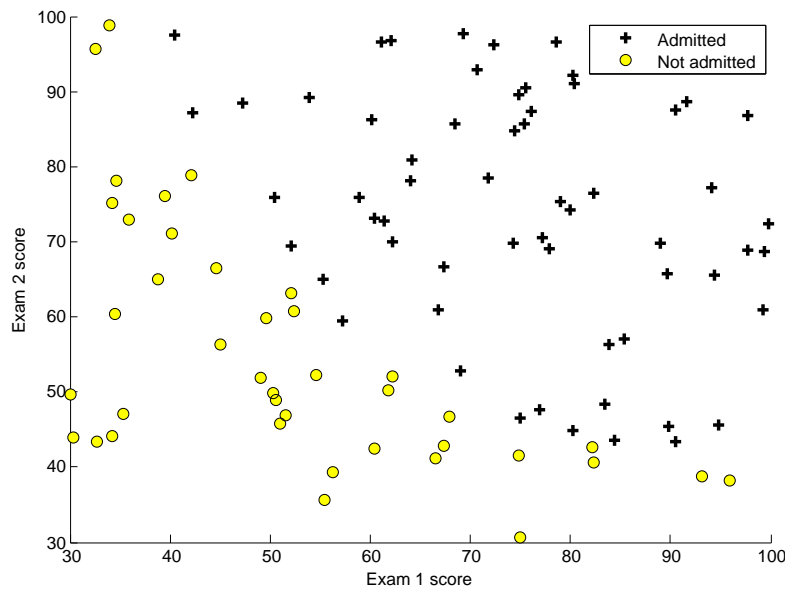


Figure 1: Scatter plot of training data

To help you get more familiar with plotting, we have left `plotData.m` empty so you can try to implement it yourself. However, this is an *optional (ungraded) exercise*. We also provide our implementation below so you can copy it or refer to it. If you choose to copy our example, make sure you learn what each of its commands is doing by consulting the Octave/MATLAB documentation.

```
% Find Indices of Positive and Negative Examples
pos = find(y==1); neg = find(y == 0);

% Plot Examples
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 Implementation

1.2.1 Warmup exercise: sigmoid function

Before you start with the actual cost function, recall that the logistic regression hypothesis is defined as:

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

where function g is the sigmoid function. The sigmoid function is defined as:

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

Your first step is to implement this function in `sigmoid.m` so it can be called by the rest of your program. When you are finished, try testing a few values by calling `sigmoid(x)` at the Octave/MATLAB command line. For large positive values of x , the sigmoid should be close to 1, while for large negative values, the sigmoid should be close to 0. Evaluating `sigmoid(0)` should give you exactly 0.5. Your code should also work with vectors and matrices. **For a matrix, your function should perform the sigmoid function on every element.**

You can submit your solution for grading by typing `submit` at the Octave/MATLAB command line. The submission script will prompt you for your login e-mail and submission token and ask you which files you want to submit. You can obtain a submission token from the web page for the assignment.

You should now submit your solutions.

1.2.2 Cost function and gradient

Now you will implement the cost function and gradient for logistic regression. Complete the code in `costFunction.m` to return the cost and gradient.

Recall that the cost function in logistic regression is

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

and the gradient of the cost is a vector of the same length as θ where the j^{th} element (for $j = 0, 1, \dots, n$) is defined as follows: