MI Activity 1

Linear Regression

Introduction

- This first exercise will give you practice with linear regression. These exercises have been extensively tested with Matlab, but they should also work in Octave, which has been called a "free version of Matlab."

Data

- Extract the files from the ex2Data.zip file.

```
PhD:~/Workspace$ unzip ex2Data.zip
Archive:  ex2Data.zip
  inflating: ex2x.dat
  inflating: ex2y.dat
```

- The files contain some example measurements of heights for various boys between the ages of two and eights.
- The y-values are the heights measured in meters, and the x-values are the ages of the boys corresponding to the heights.

Data

- Each height and age tuple constitutes one training example in our dataset.

$$x^{(i)},\ y^{(i)}$$

- There are m = 50 training examples, and you will use them to develop a linear regression model.

Supervised learning problem

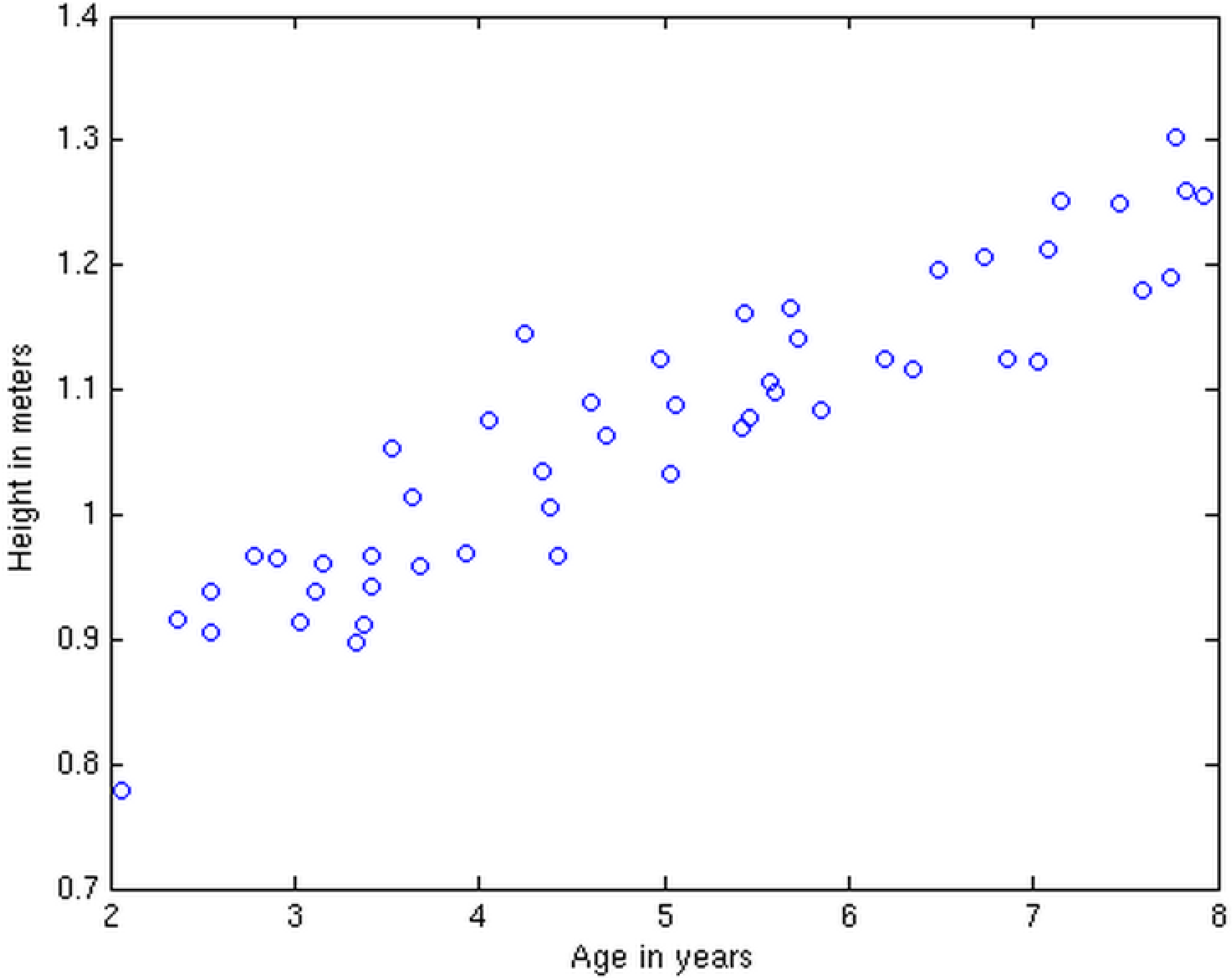
- In this problem, you'll implement linear regression using gradient descent.
- In Octave, you can load the training set using the commands

```
>> y = load('ex2y.dat');
>> whos
Variables in the current scope:

   Attr Name        Size                     Bytes  Class
   ==== ====        ====                     =====  =====
        x          50x1                        400  double
        y          50x1                        400  double

Total is 100 elements using 800 bytes
```

Plot the training set



Convention

- Before starting gradient descent, we need to add the x_0 = 1 intercept term to every example. To do this in Matlab/Octave, the command is

```
m = length(y); % store the number of training examples
x = [ones(m, 1), x]; % Add a column of ones to x
```

Linear regression model

$$h_\theta(x) = \theta^T x = \sum_{i=0}^{n} \theta_i x_i,$$

- Batch gradient descent update rule

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad \text{(for all } j\text{)}$$

Procedure 1

1. Implement gradient descent using a learning rate of $\alpha = 0.07$. Since Matlab/Octave and Octave index vectors starting from 1 rather than 0, you'll probably use `theta(1)` and `theta(2)` in Matlab/Octave to represent θ_0 and θ_1.

Initialize the parameters to $\theta = \vec{0}$ (i.e., $\theta_0 = \theta_1 = 0$), and run one iteration of gradient descent from this initial starting point. Record the value of of θ_0 and θ_1 that you get after this first iteration. (To verify that your implementation is correct, later we'll ask you to check your values of θ_0 and θ_1 against ours.)

Procedure 2

2. Continue running gradient descent for more iterations until θ converges. (this will take a total of about 1500 iterations). After convergence, record the final values of θ_0 and θ_1 that you get.

When you have found θ, plot the straight line fit from your algorithm on the same graph as your training data. The plotting commands will look something like this:

```
hold on % Plot new data without clearing old plot
plot(x(:,2), x*theta, '-') % remember that x is now a matrix with 2 columns
                           % and the second column contains the time info
legend('Training data', 'Linear regression')
```

Note that for most machine learning problems, x is very high dimensional, so we don't be able to plot $h_\theta(x)$. But since in this example we have only one feature, being able to plot this gives a nice sanity-check on our result.

Procedure 3

3. Finally, we'd like to make some predictions using the learned hypothesis. Use your model to predict the height for a two boys of age 3.5 and age 7.

Debugging If you are using Matlab/Octave and seeing many errors at runtime, try inspecting your matrix operations to check that you are multiplying and adding matrices in ways that their dimensions would allow. Remember that Matlab/Octave by default interprets an operation as a matrix operation. In cases where you don't intend to use the matrix definition of an operator but your expression is ambiguous to Matlab/Octave, you will have to use the 'dot' operator to specify your command. Additionally, you can try printing x, y, and theta to make sure their dimensions are correct.

Understanding the cost function J

We'd like to understand better what gradient descent has done, and visualize the relationship between the parameters $\theta \in R^2$ and $J(\theta)$. In this problem, we'll plot $J(\theta)$ as a 3D surface plot. (When applying learning algorithms, we don't usually try to plot $J(\theta)$ since usually $\theta \in R^n$ is very high-dimensional so that we don't have any simple way to plot or visualize $J(\theta)$. But because the example here uses a very low dimensional $\theta \in R^2$, we'll plot $J(\theta)$ to gain more intuition about linear regression.) Recall that the formula for $J(\theta)$ is

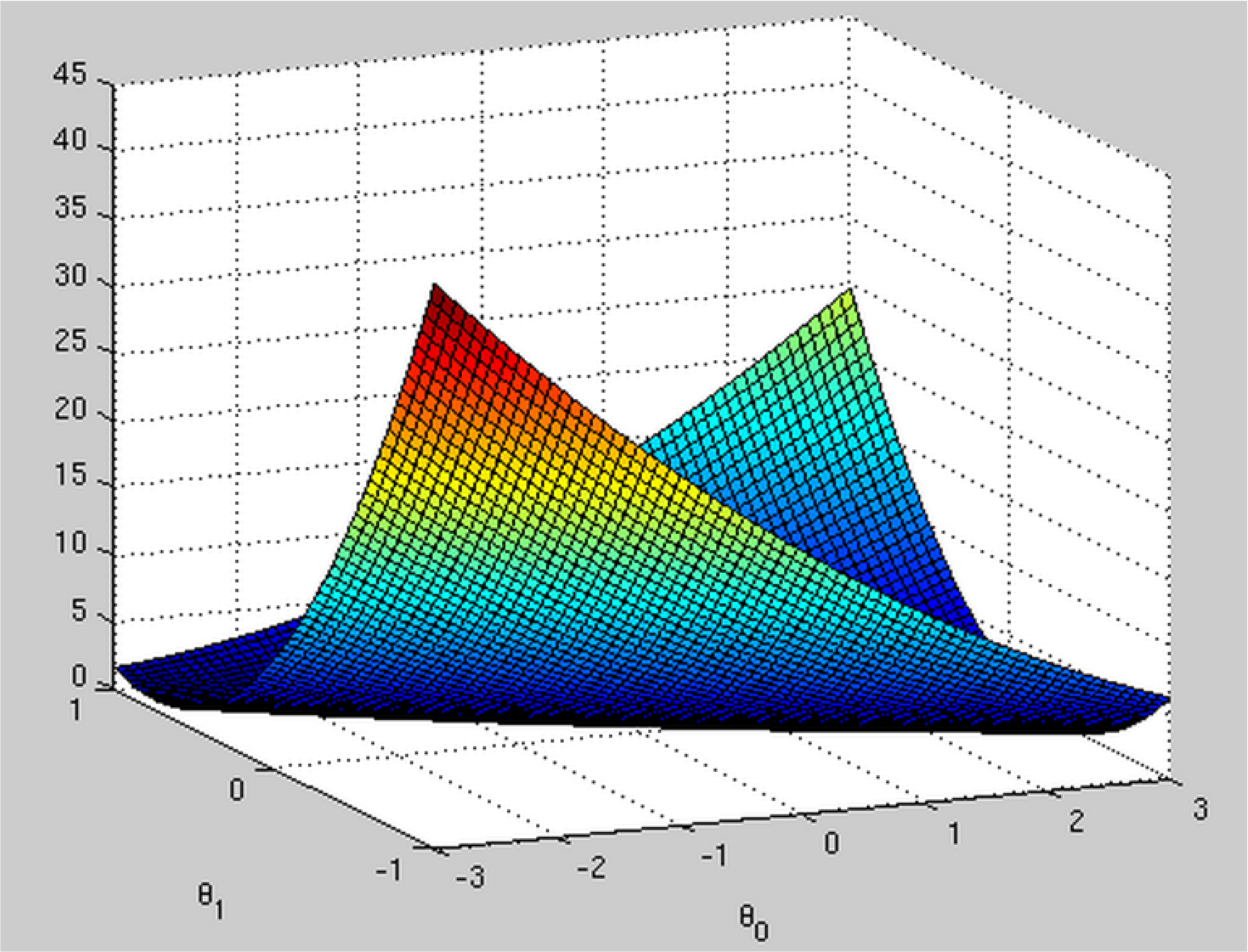
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

To get the best viewing results on your surface plot, use the range of theta values that we suggest in the code skeleton below.

```
J_vals = zeros(100, 100);   % initialize Jvals to 100x100 matrix of 0's
theta0_vals = linspace(-3, 3, 100);
theta1_vals = linspace(-1, 1, 100);
for i = 1:length(theta0_vals)
          for j = 1:length(theta1_vals)
          t = [theta0_vals(i); theta1_vals(j)];
          J_vals(i,j) = %% YOUR CODE HERE %%
    end
end

% Plot the surface plot
% Because of the way meshgrids work in the surf command, we need to
% transpose J_vals before calling surf, or else the axes will be flipped
J_vals = J_vals'
figure;
surf(theta0_vals, theta1_vals, J_vals)
xlabel('\theta_0'); ylabel('\theta_1')
```

Output



Question

- What is the relationship between this 3D surface and the value of

  $$\theta_0$$

  and

  $$\theta_1$$

  that your implementation of gradient descent had found?

Problem

- Provide your own data with at least 50 elements and apply the procedures performed in this experiment.

End

Appendix

Octave Install Ubuntu

- sudo apt-add-repository ppa:octave/stable
- sudo apt-get update
- sudo apt-get install build-dep octave
- sudo apt-get install octave