CS 335 Homework 1, Fall 2014

Dan Sheldon

Revisions:

- First release on Friday 9/12
- Minor edits and format changes made on Sunday 9/14

Instructions

All parts of the assignment are due on Friday 9/19 at noon.

Complete all exercises and problems. Submit the following files on moodle:

- `problem_3.m`
- `cost_function.m`
- `problem_4.m`
- `report.pdf` (optional: see below)

Also submit any auxiliary files you created that contain functions needed to run your code.

You should submit a report (written or typed) with solutions to the exercises and any non-coding parts of the problems. Please include the plot of your data from Problem 4 in the report. This can be submitted either electronically or by hard copy. Hard copies can be left in the drop box with my name on it outside the CS lounge (Clapp 222B).

Exercises (2 points each)

These are designed to test your understanding of basic concepts. I strongly recommend attempting these first on your own.

Partial derivatives

Consider the following functions of the variables u, v, and w. Assume the variables x, y, $x^{(i)}$ and $y^{(i)}$ are **constants**: they represent numbers that will not change during the execution of a machine learning algorithm (e.g., the training data).

- $f(u, v) = 8uv^2 + 3u + 4v$
- $g(u, v, w) = x \cdot log(u) + yuvw + 10$

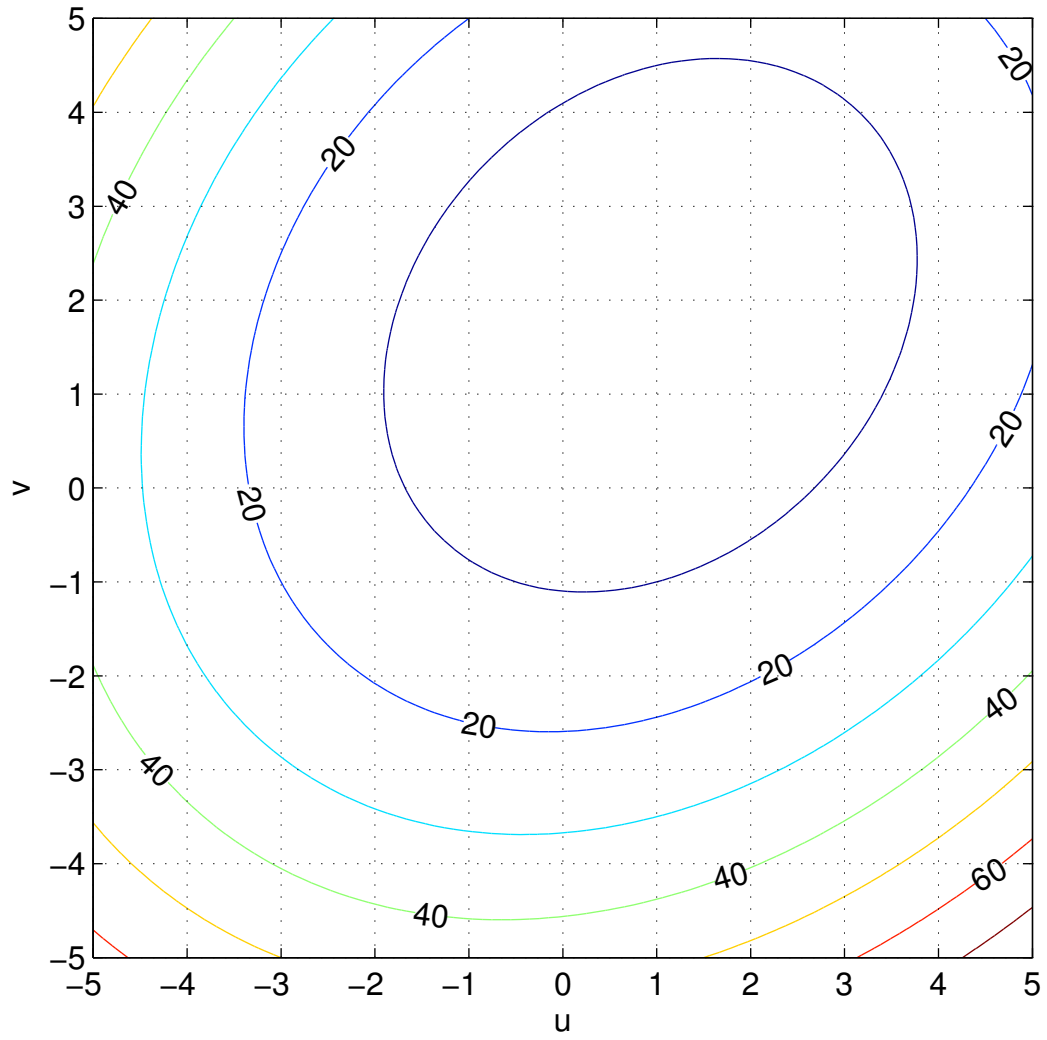
- $h(u,v) = \sum_{i=1}^{m} \frac{1}{2}(x^{(i)}u + y^{(i)}v)^2$

Write the following partial derivatives:

1. $\frac{\partial}{\partial u} f(u,v) =$
2. $\frac{\partial}{\partial v} f(u,v) =$
3. $\frac{\partial}{\partial u} g(u,v,w) =$
4. $\frac{\partial}{\partial v} g(u,v,w) =$
5. $\frac{\partial}{\partial w} g(u,v,w) =$
6. $\frac{\partial}{\partial u} h(u,v) =$
7. $\frac{\partial}{\partial v} h(u,v) =$

Partial derivative intuition

Consider the following contour plot of a function $f(u, v)$:



For each of the following partial derivatives, state whether it is positive, negative, or equal to zero. Briefly explain. These questions can be answered from the contour plot without knowing the formula for the function.

(**Note**: for two numbers a and b we will use the notation $\frac{\partial}{\partial u} f(a, b)$ to mean "the partial derivative of $f(u, v)$ with respect to u at the point where $u = a$ and $v = b$". This notation is succinct but obfuscates the original variable names. A more explicit way to write the same thing is $\frac{\partial}{\partial u} f(u, v)|_{u=a, v=b}$.)

1. $\dfrac{\partial}{\partial u} f(2, -2)$
2. $\dfrac{\partial}{\partial v} f(2, -2)$
3. $\dfrac{\partial}{\partial u} f(-3, -3)$

4. $\dfrac{\partial}{\partial v} f(-3, -3)$
5. To the nearest integer, estimate the values of u and v that minimize $f(u, v)$.

Problems (10 points each)

The first few problems will consider the "slope-only" (i.e., $\theta_0 = 0$) linear regression model from class, which has the following hypothesis and cost function:

$$h_\theta(x) = \theta_1 x, \quad J(\theta_1) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

1. Consider the following small data set:

x	y
1	3
-1	-2
2	4

   Solve for the value of θ_1 that minimizes the cost function by substituting the values from the training set into the cost function, setting the derivative equal to zero, and solving for θ_1. Show your work.

2. Now do the same thing, but don't substitute the values of the training set into the cost function. Instead, leave the $x^{(i)}$ and $y^{(i)}$ variables, take the derivative with respect to θ_1, set it equal to zero, and solve for θ_1. This will give you a general expression for θ_1 in terms of the training data.

   Check your answer by plugging in the training data from the previous problem into your expression for θ_1. You should get the same value for θ_1 that you got in that problem.

3. Extra credit. Do the same thing as in the previous problem, but use the more general hypothesis and cost function:

$$h_\theta(x) = \theta_0 + \theta_1 x, \quad J(\theta_0, \theta_1) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

   Take partial derivatives with respect to both θ_0 and θ_1 and set them to zero, then solve the system of two equations to come up with expressions for θ_0 and θ_1 in terms of the training data.

4. In this problem you will implement gradient descent for linear regression. Open the file `problem_3.m` and read the code. If you try to run the script, you will get an error because it attempts to call the function `cost_function()`, which is not yet defined. To remedy this, create a function in the file `cost_function.m` with the following specification:

```
function [ cost ] = cost_function( x, y, theta0, theta1 )
% COST_FUNCTION Compute the squared error cost function for linear regression
% Inputs:
%   x        vector of m input values from the training set
%   y        vector of m output values from the training set
%   theta0   the intercept parameter (scalar)
%   theta1   the slope parameter (scalar)
% Output:
%   cost     the value of the squared error cost function (scalar)
```

After you properly implement `cost_function()`, you can run the script `problem_3` and a figure containing two plots will appear. The left plot shows the contours of the cost function, and the right plot shows both the data and the current hypothesis, which is not a good fit to the data.

Now, to find a good hypothesis, implement gradient descent in the spot indicated in the code. Here are some guidelines:

- Select a step size
- Run for a fixed number of iterations (say, 200)
- Update `theta0` and `theta1` using the partial derivatives

Record the value of the cost function attained in each iteration of gradient descent so you can examine its progress. After running for the specified number of iterations, plot the cost function vs. iteration in a new figure window. Include the code to create this plot in your submission.

After you have completed the implementation, do some experiments with different numbers of iterations and step sizes to assess convergence of the algorithm. Report the following in your written submission:

- A step size for which the algorithm converges to the minimum in at most 200 iterations
- A step size for which the algorithm converges, but it takes more than 200 iterations
- A step size for which the algorithm does not converge, no matter how many iterations are run

To summarize, **here are the items you should complete in this problem**:

1. Implement the cost function in `cost_function.m`
2. Implement gradient descent in `problem_3.m`
3. Write code in `problem_3.m` to plot the cost function vs. iteration
4. Run the convergence experiments and answer the questions about step sizes in your report

5. In this problem you will find a data set of your own (with at least 5 training examples) that is suitable for linear regression with one input variable. Do the following:

   - Briefly describe what x and y are in your data, and why it may be interesting to predict y for a value of x that is not in the training set.
   - Create a file `problem_4.m` in which you either enter the data directly into MATLAB variables or load it from file, then learn the parameters θ_0 and θ_1 of a linear regression model. (If you can't get gradient descent working, let me know and I will provide an alternate way to solve for θ_0 and θ_1.)
   - Plot your data and the best fitting hypothesis. Include the plot in your written report.
   - Use the learned hypothesis to make a prediction for an input value x that was not in the data set. Briefly discuss the result in the context of the data set you chose. Does the prediction seem useful?