Optimization

In this homework you will explore logistic regression problem, batch stochastic gradient descent and asynchronous batch stochastic gradient descent.

Due Date: Wed, April 27th at 5pm

Background

- Read gradient descent backgrounders
  - http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html
  - http://ss.sysu.edu.cn/~py/.%5CDM%5CLecture6.pdf
  - https://www.metacademy.org/graphs/concepts/stochastic_gradient_descent
- Read the Hogwild! paper, https://papers.nips.cc/paper/4390-hogwild-a-lock-free-approach-to-parallelizing-stochastic-gradient-descent.pdf
- Install the anaconda python installation, python version 3.4+
- Optional -- this blog on python and C, http://chimera.labs.oreilly.com/books/1230000000393/ch15.html#_discussion_248
- Exercise files and data: https://s3.amazonaws.com/cse6240-spr2016/ex2-003.zip

Part 1 -- 100 points

You will port Exercise #2 from Andrew Ng's Coursera course "Machine Learning -003" to python and complete it using python / ipython notebooks. You can look at this blog as a reference, http://aimotion.blogspot.com/2011/11/machine-learning-with-python-logistic.html

1. Download the zip file ex2-003.zip
2. Ignore submit.m and submitWeb.m
3. Implement ex2.m and ex2_reg.m as ipython notebooks
4. Convert the remaining .m files into .py files, refactoring as necessary and implementing the missing pieces of code where indicated. Make sure you follow python programming standards where possible. Document your code appropriately (removing the prompt comments).

5. Hint: you will need to replace the matlab function fmin_unc with an alternative function that exists in python.
6. To submit, zip up the folder with your python code and submit electronically.
7. This portion of the assignment is worth 100 points, to be graded in the same manner as specified in Andrew Ng's handout.

Part 2 -- 100 points

In this part of the exercise you will modify your code from Part 1 to use batch stochastic gradient descent.

1. Implement a function `batch_sgd(X,y,theta,learning_rate,max_iters=400,tolerance=1e-3)` which implements batch stochastic gradient descent for optimization.
    a. The function should return a tuple of arrays: (theta, J) which are the parameters learned at each iteration
    b. max_iters and tolerance should be used for terminating batch_sgd
2. Replace the optimization routine in your file ex2_reg.py (line 96 in ex2_reg.m) with your batch_sgd code. Hint: you may want to standardize your input data for faster convergence.
3. Replace the plotting code in your file ex2_reg.py (line 99+ in ex2_reg.m). Instead of plotting the decision boundary, plot the cost vs. number of iterations for at least 6 values of learning rate. Hint: you might start with LR values of [.01, .03, .05, .1, .3, .5]
4. Submit your new code for batch_sgd and the cost plot.
5. This part of the assignment will be worth 100 points: 80 points for the code (accurate, complete, sylisticially ok) and 20 points for the plot.

Part 3 -- 200 points

In this part of the exercise you will modify your code from Part2 to use parallel batch stochastic gradient descent along the lines of Hogwild!

1. Implement a function `parallel_batch_sgd(X,y,theta,learning_rate,nthreads=1,max_iters=400,tolerance=1e-3)` which extends implements Hogwild!
    a. You may implement parallel workers as processes, for example using multiprocess and joblib for memory mapped files, or as multiple threads.
2. Repeat steps 2, 3 from Part2 using parallel_batch_sgd().
3. Profile the relative speedup of parallel_batch_sgd() for 1,2,3 and 4 threads compared to batch_sgd using the same data and parameters (your best learning

rate, etc.). Does this vary for different learning rates? How sensitive is to standardizing the data?

4. Submit your new code + the new plot + analysis from #3.
5. This part of the assignment will be worth 200 points:
    a. 100 points for parallel_batch_sgd() implementation -- accurate, complete code; stylistically correct
    b. 25 points for the plot and answers
    c. 75 points for performance:
        i. The top 10 fastest homeworks will receive 75 points
        ii. The bottom 10 slowest homeworks will receive 15 points
        iii. Middle finishers will receive 50 points
        iv. Homeworks that are incorrect will receive 0 points.