**Gradient Descent  
Assignment 10**

**CS 195W**

**Overview**

This assignment focuses on implementing gradient descent and then using it to solve 3 different scenarios:

1. Finding the global minimum of known convex function with a given start location and experimenting with different step sizes.
2. Estimating the minimum of an unknown “black box” function (only the .class file is provided) over a given domain by running many trials with a random start location.
3. Estimating the minimum of an unknown “black box” function (only the .class file is provided) over a given domain while using a parallel implementation to search over different sub-domains.

Fortunately, we have provided you with most of the code you will need already here. The only modifications you should need to make are:

1) You will need to implement gradient descent in GradientDescent.java

2) For the 3 scenarios, you need to change some parameters in the driver code in Question1Main.java, Question2Main.java, and Question3Main.java.

**Deliverables**

Please provide a written report fully answering each of the questions and your source code with all modifications included. Submit a single archive file containing all deliverables.

**Implementing Gradient Descent**

The first thing you will need to do is implement the gradient descent algorithm inside the GradientDescent.java class of the provided code. Your code will go inside the while loop of the GradientDescent::start() function. Refer to the “A computational example” pseudo code on gradient descent Wikipedia article for guidance. The gradient (“f\_prime”) can be calculated by using our provided Function class. None of the code for Questions 1, 2, and 3 will work until you have a working gradient descent implementation. Question 1 should be used to test your implementation since the objective function is known.

**Question 1**

Find the global minimum using gradient descent for a known convex function with a given starting location. Experiment with step sizes and discuss the effects of using different step sizes.

Your objective function is:

**y = x^5 + 27x^4 + 37x^3 - 339x^2 - 326x + 600**

which contains more than one min points. Your task is to try out different step-sizes and starting points and eventually converge to the global minimal point.

**Question 2**

Find the minimum within a given domain for an unknown “black box” function. This function is provided as the BlackBoxFunction.class file. The given domain is specified in the constants in the Question2Main.java file. Use the following instructions:

1. Open the Question2Main.java file.
2. Experiment with adjusting the step size value, maximum number of steps, and number of iterations with random starting location.

Hint: If the step size is too large, the gradient descent will be unstable. If the max number of steps is too low, the gradient descent might not travel far enough to find a local minimum. Finally, if the number of total iterations (gradient descent trials with random starting location) is too low, the best minimum you find may not actually be the best global minimum for this domain.

1. Once you are confident that you have the correct global minimum value for this domain, report the coordinates of the global minimum (x, f(x)) and the parameters you used.

**Question 3**

This question is similar to question 2, except that you are adjusted how the work is distributed for a parallel framework. Once again, you are finding the minimum within a given domain for a different unknown “black box” function. This function is provided as the BlackBoxFunction2.class file. The given domain is specified in the constants in the Question3Main.java file. Use the following instructions:

1. Open the Question3Main.java file.
2. Experiment with adjusting the number of worker threads and the number of iterations (gradient descent trials with random starting location) for each worker. Each worker is running gradient descent trials on equally sized sub-domains of the total domain. The size of the sub-domains is inversely proportional to the number of workers. The best minimum from each worker are then compared to find the best global minimum over the entire domain.
3. Once you are confident that you have the correct global minimum value for this domain, report the coordinates of the global minimum (x, f(x)) and the parameters you used.

For questions feel free to ask at ***patrick\_malden@brown.edu, mmichail@cs.brown.edu; xiaofeng\_tao@brown.edu; lixing\_lian@brown.edu; cheng\_ren@brown.edu; quan\_fang@brown.edu***