DSCI 320: Optimization Methods in Data Science

Homework assignment 4 – due Friday 11/01/2019

Problem 1 (Stochastic gradient descent). Let's go back to finding a function that approximates a whole data set using the least-squares method. In particular, we're going to try and find a linear function $at+b$ that approximates data points $(t_i, y_i)_{i=1}^N$.

(a) To this end, we need a data set. We could go out to the internet for one, but for the purpose of testing algorithms, we can also just generate data. First, randomly choose $N = 1000$ points t_i in the interval $[0, 100]$. For each of these t_i, generate a y_i by drawing from the normal disttribution $N(50 + \frac{1}{2}t_i, 10)$ – in other words, one could think of the exact values to lie on the line $y = \frac{1}{2}t + 50$, but actual measurement points are normally distributed around this line with a standard deviation of 10. Show a plot of all 1000 of these data points.

(b) Next, we need to find the approximating line to this data set. The least squares approach would require us to find $x = (a, b)$ so that

$$f(a, b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (at_i + b))^2$$

is minimized. Use your method of choice to find this optimum $x^* = (a^*, b^*)$ and show a plot of both the original data and the approximating linear function. (Because the objective fuction is quadratic, your "method of choice" should in fact be Newton's method, because then you are done in exactly one step.)

(c) Let us use the Steepest Descent method for this problem and track how fast it converges.[1] To this end, implement a method in which you compute $x_{k+1} = (a_{k+1}, b_{k+1})$ using the iteration

$$x_{k+1} = x_k + \alpha_k p_k$$

where p_k is computed as the steepest descent direction using all data points:

$$p_k = -\nabla\left[\frac{1}{N}\sum_{i=1}^{N}(y_i - (at_i + b))^2\right].$$

Start at $x_0 = (a_0, b_0) = (0, 500)$ (i.e., corresponding to a horizontal with vertical offset equal to 500). For the step length, choose $\alpha_k = \frac{500}{(k+1)\|p_k\|}$.[2]

[1] Because the problem is quadratic, we really wouldn't choose the Gradient/Steepest Descent method for such a problem, but the approximation by a linear function is an easy to use test case from which we can learn.

[2] You might want to think about why I chose it that way.

Since you know x^* from the previous problem, track $\|x^* - x_k\|$ and show it as a function of k in the form of a graph.

(d) Let's also try the Stochastic Gradient Descent method. To this end, implement a method in which you compute $x_{k+1} = (a_{k+1}, b_{k+1})$ using the iteration

$$x_{k+1} = x_k + \alpha_k p_k$$

where p_k is computed as the steepest descent direction using only a subset S_k of all data points:

$$p_k = -\nabla \left[\frac{1}{M} \sum_{i \in S_k} (y_i - (at_i + b))^2 \right].$$

Here, S_k is a randomly chosen subset of $\{1, \ldots, N\}$ in each iteration k with $M = 10$ elements. In other words, p_k only uses 1% of all of the data in each step. Using the same step length strategy as above, generate again the sequence x_k that results from the method. As before, plot $\|x^* - x_k\|$ as a function of k.

(40 points)

Problem 2 (Monte Carlo optimization). In the previous homework, you were asked to find the equilibrium position (x, z) of a mass suspended from the ceiling via three springs. This was equivalent to finding the minimizer of the (energy) function

$$f(x, z) = \sum_{i=1}^{3} \frac{1}{2} D(L_i(x, z) - L_0)^2 + mgz,$$

where $L_i(x, z) = \sqrt{(x - x_i)^2 + (z - z_i)^2}$ is the length of the ith spring or, equivalently, the distance of the point (x, z) from the i spring attachment point. The attachment points, spring constant D, and mass m can be found in the previous assignment.

Implement a Monte-Carlo method to find the minimum. To this end, discuss your choice the constant T in the Monte-Carlo method that controls whether or not you accept a worse sample. Also discuss your choice of the parameter σ that controls how far a trial point is from the current iterate.

Using your sequence of points, generate a plot that shows these points in the $x - z$ plane. Also generate a plot that shows the function values $f(x_k, z_k)$ for each k. Finally, indicate what the best point is that you have found (i.e., the coordinates of that point (x_k, z_k) that has the lowest function value.

(40 points)

Problem 3 (Simulated Annealing). Modify your program from the previous problem so that it solves the same problem, but using the Simulated Annealing method. This is easily done if, instead of a constant parameter T, you choose T_k differently in each iteration so that $T_k \to 0$. Play with different ways of choosing T_k – in general, you want it to be large in the beginning and small towards the end of the sampling process, with a slow decrease inbetween (e.g., $T \propto 1/k$ or $T \propto 1/\sqrt{k}$, or something even slower such as $T \propto 1/k^{0.1}$).
(20 points)

Bonus problem (Genetic algorithms for optimization). If you want to play a bit, use the same problem as before and use a genetic algorithm to solve it. Explain your choices of initial population, mating strategy, and mutation strategy. Illustrate the evolution of your population graphically in ways you think show how the algorithm works. **(10 points)**

Project ideas. It is also time to think about ideas for your final project. The stochastic methods you have investigated in this homework, as well as the mechanical problem with a mass and springs would make for some good projects.

For example, think about the following observation: With just one mass, there are already two minima – one where the mass is suspended from the ceiling, but there is another minimum where the mass is propped up by the springs to a position above the line where the springs are attached. For such a simple system, it is easy to reason about how many minima there are, and what a good starting point for a Newton iteration would be, and so a stochastic algorithm would not be used in practice. But you could easily extend this example to several masses connected to the ceiling and each other by springs. In such cases, the number of local minima often rises very quickly, and stochastic algorithms are in fact routinely used for these kinds of problems. (In fact, this is exactly the situation the field of *molecular dynamics (MD)* finds itself in, if you wanted to take a look around the web.)

If you have comments on the way I teach – in particular suggestions how I can do things better, if I should do more or less examples, powerpoint slides vs whiteboard, etc – or on other things you would like to critique, feel free to hand those in with your homework as well. I want to make this as good a class as possible, and all comments are certainly much appreciated!