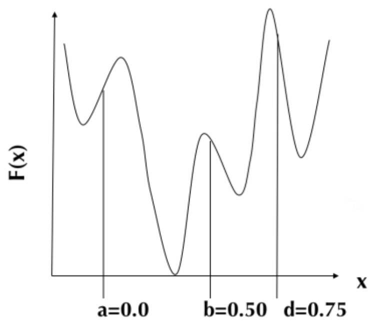
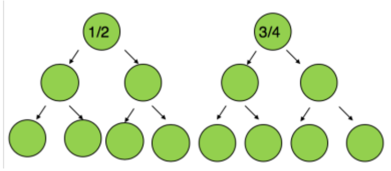
C277B: Machine Learning Algorithms

Homework assignment #1: Local Optimization Methods

Assigned Jan. 23 and Due Feb. 7



1. Bisection vs. Golden Section. In class we used the simple bisection method to take the first step in isolating at least one minimum for the function shown. This first step in placement of *d* reduced the original interval [*a*,*b*,*c*] =1.0 to [*a*,*b*,*d*] = 0.75. But in general, the average size interval <L> after Step 1 is determined by the equal probability of placing point *d* in either sub-interval, such that $<L_1>$=P(left-interval) x ½ + P(right-interval) x ¾ = 0.625 (since you can't *a priori* know the best half)



(a) For step 2, place point *e* at the bisector of larger interval [a,b]. Why is this better than [b,d]?

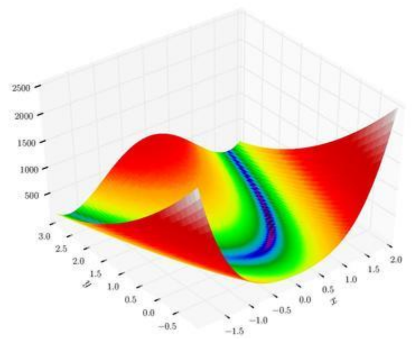
(b) What is the new interval and how much is the search space reduced?

(c) For step 3, reduce the size of the interval from step 2 by placing point *f* at the bisection of your choice.

(d) fill in tree for all possible size intervals for steps 2 and 3. Write your answers in ratios to the interval size of the previous step.

(e) What is average size of interval at steps 2 and 3?

(f) How much does Golden Section improve over Bisection at each step? Please use a chart of steps v.s. different methods to show their difference.



2. Local optimization using 1st and quasi-2nd order methods. You will solve the following optimization problem using a python code you develop for the steepest descents method! For the function

$$f(x,y) = x^4 - x^2 + y^2 + 2xy - 2$$

there are three stationary points found over the range x = [-2,2] and y = [-2,2].

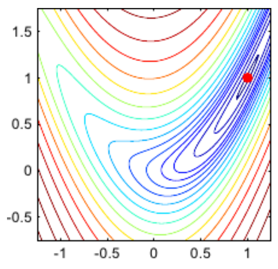
(a) Starting from point $(1.5,1.5)$, and with stepsize =0.1, determine new (x,y) position using one step of the steepest descent algorithm (check against the debugging output). Is it a good optimization step? Depending on this outcome, how will you change the stepsize in the next step?

(b) Implement the steepest decent using the provided template. Continue executing steepest descents. How many steps does it take to converge to the local minimum to tolerance = 1 x 10^{-5} of the gradient (check against the debugging output and compare code timings)?

Note: You don't need to use line search, just take one step in the search direction, and use the following stepsize update: $\lambda = \begin{cases} 1.2\,\lambda \; for\ a\ good\ step \\ 0.5\,\lambda \; for\ a\ bad\ step \end{cases}$

(c) Compare your steepest descent code against conjugate gradients (CG), and BFGS to determine the local minimum starting from $(1.5,1.5)$. In terms of number of steps, are conjugate gradients and/or BFGS more efficient than steepest descents?

Note: See SciPy documentation on how to use CG and BFGS, examples available at the end of webpage: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html



3. Local optimization and machine learning using Stochastic Gradient Descent (SGD). The Rosenbrock Banana Function looks innocuous enough

$$f(x, y) = (1 - x)^2 + 10(y - x^2)^2$$

with only one (global) minimum at $(x, y) = (1.0,1.0)$!

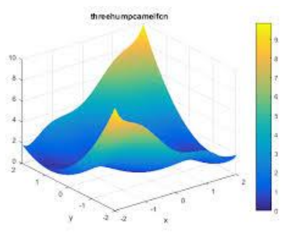
(a) Starting at $x = -0.5$ and $y = 1.5$, and using your code for steepest descents with stepsize =0.1, how many steps to converge to the minimum? Use a tolerance = 1 x 10^{-5}

(b) By adding a small amount of stochastic noise to the gradient at every step (In your code add a random vector that is the same norm as the gradient at that step), which is equivalent to a small batch derivative of any loss function in deep learning, implement your own stochastic gradient descent code by modifying on your steepest descent code, and run the SGD algorithm. (Check against debugging outputs.)

(c) evaluate how much better or worse is the SGD convergence against the CG or BFGS method to find the global minimum, in terms of number of steps. Converge function/gradient to tolerance $=1 \times 10^{-5}$

(d) Can you draw a firm conclusion on the outcome with just one run of each method? If not, explain why

(e) Run all of the algorithms multiple times starting at different (x,y) positions to understand the average performance of each. Explain the relative performance of the non-stochastic and stochastic methods on the Rosenbrock Banana Function.



4. Stochastic Gradient Descent with Momentum (SGDM). The Rosenbrock Banana function with one minimum is not the best way to illustrate the power of the SGD or SGDM method. Hence we next investigate the Three-Hump Camel function

$$f(x, y) = 2x^2 - 1.05x^4 + x^6/6 + xy + y^2$$
$$x \in [-2,2], y \in [-2,2]$$

which is a convex function with three minima. This defines our first "multiple minima" problem where there is a global solution as well as two less optimal solutions.

(a) Utilize SGD to find the *global* minimum, and compare it to CG or BFGS as you did in (2e). Starting from [-1.5,-1.5], converge function and gradient to tolerance = 1×10^{-5} with stepsize =0.1. On average, did you get a better result in finding the global minimum with SGD in terms of fewer steps on average?

(b) Implement the SGDM algorithm with momentum $\gamma = 0.9$. Now use SGD with Momentum to find the global minimum. Again start from [-1.5,-1.5] with stepsize = 0.1 and converge function and gradient to tolerance $=1 \times 10^{-5}$. On average, did you get a better result using SGDM compared to SGD, CG or BFGS in finding the global minimum in terms of fewer steps?

Note

To access your helper code and debugging output, you need to go to bcourse's Files tab, a file named **HW1 MSSE reference.ipynb** is there for you!

Homework format and submission note

- Complete your homework in a jupyter notebook
- Kernel->Restart & Run All to execute the notebook from a blank slate
- Double-check text, math, code, outputs, figures. Re-run if needed
- File->Download as->HTML (.html) to make homework1.html (This keeps the format correctly, don’t directly export pdf because it can mess up your work!)
- Open homework1.html in your web browser
- File->Print (Save as PDF) in your browser to make homework1.pdf
- Submit homework1.pdf to Gradescope