CSE257 Winter 2025: Assignment 1

Deadline: Jan-26 11:59pm. You will need to make the following two submissions.

1. Submit a pdf file of your (written or typed) answers on Gradescope (if you are enrolled, you should have already been added to the course on Gradescope).

2. Submit your Python code or Jupyter notebook files for problems that have the "Implementation Involved" label, via the Dropbox link: `https://www.dropbox.com/request/XzChzoa7FtsJ7x4e0yR5`. You can import any library that does not already implement the algorithms we ask for. Submit all the code in one zip file. For these questions that have implementation involved, you need to still put whatever is asked in the questions, such as plots and explanations, in the pdf file that you submit to Gradescope. Otherwise the question will be considered unanswered for grading purposes.

3. Again, use of ChatGPT or other language models is strictly forbidden for anything you write in the answer pdf file (even for editing grammar), but they are allowed for the implementation and the code you submit.

Question 1 (3 Points, Implementation Involved)**.** For each of the following functions, use Python to plot 3 different level sets of your choice. Small numerical errors are allowed and it doesn't matter how you find the points (for instance, you can simply create a fine grid within some range and select the points that are close to the level sets). You need to show the plots in the pdf file of your answers. It doesn't matter whether you put all the level sets in the same plot or separate ones, just make them clear.

1. $f_1(x_1, x_2) = x_1^2 + 2x_1x_2 + 2x_2^2$

2. $f_2(x_1, x_2) = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$, i.e., the Gaussian density function over $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, where $\mu = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ is the mean, and $\Sigma = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$ is the covariance matrix and $|\Sigma|$ its determinant. Before plotting, first think about what these level sets should look like, and then it should be very easy to plot.

3. Drop-wave function: $f(x_1, x_2) = -\dfrac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5(x_1^2 + x_2^2) + 2}$.

Btw, do the level sets of the first two functions look similar? If so, explain why; if not, you can ignore this.

Question 2 (4 Points, Implementation Involved)**.** We talked about the three-point method in class as a way to show the limitation of the first-order understanding of nonlinear functions. But it is also a practical algorithm that can be useful when the function is not analytically known or computing gradient is very costly.

The simple version of the algorithm for minimizing a function $f : \mathbb{R}^n \to \mathbb{R}$ can be roughly described as follows. Start from an initial point $x \in \mathrm{dom}(f)$, sample a random vector $p \in \mathbb{R}^n$ with bounded magnitude $\|p\| \leq \varepsilon$ where $\varepsilon \in \mathbb{R}^+$ is a small constant and $\|\cdot\|$ is the Euclidean norm. Evaluate three points $f(x)$, $f(x+p)$, and $f(x-p)$. Update x to the point with the lowest function value out of the three, and iterate until some termination criterion is met.

1. Give an informal justification to this algorithm, and what are the assumptions needed for it to work well.

2. Implement the procedures that you just described to minimize the function $f(x, y) = x^2 + 4y^2$ starting from $(x, y) = (1, 1)$, using constant ε of 0.1 (Hint: you can randomly sample a vector, make it a unit vector of the same direction first, and then multiply by the step size). Plot how the function value changes as the number of iteration increases (i.e., x-axis shows the number of steps taken, y-axis is the function value. Plot 30 iterations). Then, perform gradient descent on the same function, starting from the same initial point, using the same step size $\varepsilon = 0.1$ (you also need to make the gradient a unit vector first, and then multiply with ε). Plot how the function value changes over iterations as well (30 iterations).

3. Now consider the drop-wave function from Question 1.3. Plot how the function value changes over iterations in both three-point methods and gradient descent, both starting from same initial point $(2, 2)$ and using fixed step-size of $\varepsilon = 0.1$.

4. Based on the performance of the two methods on the two functions above, you should see that, saying which method is "better" without specifying the target class of functions does not make much sense. This will be a recurring theme in this course. Now specifically in this context, make this understanding more clear: on what type of functions do you think gradient descent should be better than three-point methods? (define "better" in the first place) and when should it be the opposite?

You can always run multiple rounds of the three-point methods with different random seeds and look at the average behavior.

Question 3 (2 Points)**.** Consider the function $f(x, y) = x^2 + 8y^2$ and the initial point $(1, 1)$. Answer the following:

1. Describe the first two steps of gradient descent from the initial point, using the **optimal** step sizes along the gradient descent directions in each iteration. By optimal, we mean the step size that attains the maximal decrease in the function value along the direction chosen in that iteration.

   For each step, write down the gradient descent direction for that step first, and then explain how you find the optimal step size for that direction.

2. Show that there exists a choice of step sizes such that by still following the gradient descent direction in each iteration, we can converge to the global minimum of $f(x, y)$ in only two iterations from (x_0, y_0). Explain these two steps. That is, give the directions and step sizes that are used in these two iterations.

Question 4 (2 Points)**.** Consider the function $f(x_1, x_2) = x_1^2 - 3x_2^2$:

1. Write down the Hessian of the function.

2. Write down one point x where the Newton direction at x is non-zero and is *not* a descent direction. Explain why.

Question 5 (3 Points, Implementation Involved)**.** Consider $f(x_1, x_2) = x_1^2 - x_1x_2 + 3x_2^2 + 5$ and let the initial point be $x = (2, 2)$. Implement the following procedures for minimizing f. For each case, plot the sequence of points you get in 10 iterations in the x_1-x_2 plane. (Again, show plots in the answer pdf file. If the algorithm converges before 10 iterations then just plot up to convergence.) We write $x^{(k)}$ for the point at the k-th iteration, i.e., the algorithm goes through $x^{(0)}, x^{(1)}, ...$ etc. $\|\cdot\|$ is the Euclidean norm.

1. Select some constant stepsize $\alpha > 0$ such that steepest gradient descent approaches the global minimum x^*, i.e., $\|x^{(k)} - x^*\|$ decreases as k increases, at least in the first 10 iterations that you plot.

2. Select another constant stepsize α', such that the steepest gradient descent using α' as stepsize diverges (i.e. $\|x^{(k)} - x^*\|$ increases as k increases).

3. Perform Newton descent using step size $\alpha = 1$.

Question 6 (3 Points)**.** Derive the Maximum Likelihood Estimator for Gaussian distributions based on necessary conditions for optimality. Let $\mathcal{N}(x|\theta)$ be the Gaussian distribution over $x \in \mathbb{R}^n$, parameterized by $\theta = (\mu, \Sigma)$, where

$\mu \in \mathbb{R}^n$ is the mean and $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix. Let $D = [d_1, ..., d_k]$ be a dataset of k samples, where each sample $d_i \in \mathbb{R}^n$ is an n-dimensional vector. The likelihood function with respect to D is written as

$$L(\theta) = \prod_{i=1}^{k} \mathcal{N}(d_i|\theta).$$

The MLE is then $\theta^* = \arg\max_\theta L(\theta)$, i.e., the parameter choices of μ and Σ that maximize the likelihood function. Intuitively, it should give us a Gaussian distribution that best-explains where the dataset D comes from.

1. Prove that for any positive function $f(x) : \mathbb{R}^n \to \mathbb{R}$ (i.e., $f(x) > 0$ for any x), an input x^* maximizes $f(x)$ if and only if x^* minimizes $-\log(f(x))$, and this is true both locally and globally. (Hint: look up the original definition of local/global min/max.)

2. Calculate the gradient of $-\log(L(\theta))$ with respect to the parameters θ, i.e., μ and Σ. For this specific question, you can use ChatGPT to help you with matrix derivatives.

3. Using the first-order optimality conditions, derive the choice of $\theta^* = (\mu^*, \Sigma^*)$ that maximizes $L(\theta)$. (Note that to be rigorous, we also need to confirm the second-order optimality conditions, but since that calculation over μ, Σ becomes complicated we do not do it here.)

Question 7 (0 Point, no need to submit anything, but I want you to think about it and try)**.** Prove that gradient descent with exact line search (which means it takes the optimal step size in the sense of Question 3.1) should always take orthogonal directions in each iteration. That is, suppose p_k is the gradient descent direction in the k-th iteration of gradient descent for minimizing an arbitrary continuously differentiable and lower-bounded function $f(x) : \mathbb{R}^n \to \mathbb{R}$. After we use a step size α that (at least locally) minimizes $f(x_k + \alpha p_k)$ along the direction of p_k (i.e. $\min_{\alpha \geq 0} f(x_k + \alpha p_k)$), the next direction p_{k+1} in gradient descent always satisfies $p_k p_{k+1} = 0$. You should have seen this when answering Question 3.1, and we are now asking for a proof in the general, for any differentiable function f.