Written Assignment 2

Due: Friday 02/16/2024 @ 11:59pm EST

Disclaimer

I encourage you to work together, I am a firm believer that we are at our best (and learn better) when we communicate with our peers. Perspective is incredibly important when it comes to solving problems, and sometimes it takes talking to other humans (or rubber ducks in the case of programmers) to gain a perspective we normally would not be able to achieve on our own. The only thing I ask is that you report who you work with: this is **not** to punish anyone, but instead will help me figure out what topics I need to spend extra time on/who to help. When you turn in your solution (please use some form of typesetting: do **NOT** turn in handwritten solutions), please note who you worked with.

Remember that if you have a partner, you and your partner should submit only **one** submission on gradescope.

Question 1: Gradient Descent (25 points)

Given two vectors $\hat{\vec{y}}^{(t)} \in \mathbb{R}^n$ and $\vec{y} \in \mathbb{R}^n$, we can measure the distance between them using the following objective function:

$$L\left(\hat{\vec{y}}^{(t)}, \vec{y}\right) = \frac{1}{2}\sum_{i=1}^{n}\left(\hat{y}_i^{(t)} - y_i\right)^2$$

Assume that $\hat{\vec{y}}^{(t)}$ is the output of an Agent at time t and that we wish to improve using gradient descent by minimizing the distance between $\hat{\vec{y}}^{(t)}$ and $\vec{y}$. Calculate the gradient $\nabla_{\hat{\vec{y}}^{(t)}} L$ that we would need to implement in our code to execute the gradient descent algorithm.

The gradient of L with respect to $\hat{\vec{y}}^{(t)}$ is a vector of the partial derivatives of L with respect to each $\hat{y}_i^{(t)}$. For each $\hat{y}_i^{(t)}$, the partial derivative is:

$$\frac{\partial L}{\partial \hat{y}_i^{(t)}} = \frac{\partial}{\partial \hat{y}_i^{(t)}}\left(\frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i^{(t)} - y_i)^2\right)$$

Since the summation is over all i, and the derivative is with respect to $\hat{y}_i^{(t)}$, only the term in the sum that involves $\hat{y}_i^{(t)}$ will contribute to the derivative. Thus, for each i, we simplify the expression above and get:

$$\frac{\partial L}{\partial \hat{y}_i^{(t)}} = \frac{1}{2}\cdot 2 \cdot (\hat{y}_i^{(t)} - y_i) = \hat{y}_i^{(t)} - y_i$$

$\nabla_{\hat{\vec{y}}^{(t)}} L$ is the vector of these partial derivatives for all i from 1 to n:

$$\nabla_{\hat{\vec{y}}^{(t)}} L = \begin{bmatrix} \hat{y}_1^{(t)} - y_1 \\ \hat{y}_2^{(t)} - y_2 \\ \vdots \\ \hat{y}_n^{(t)} - y_n \end{bmatrix}$$

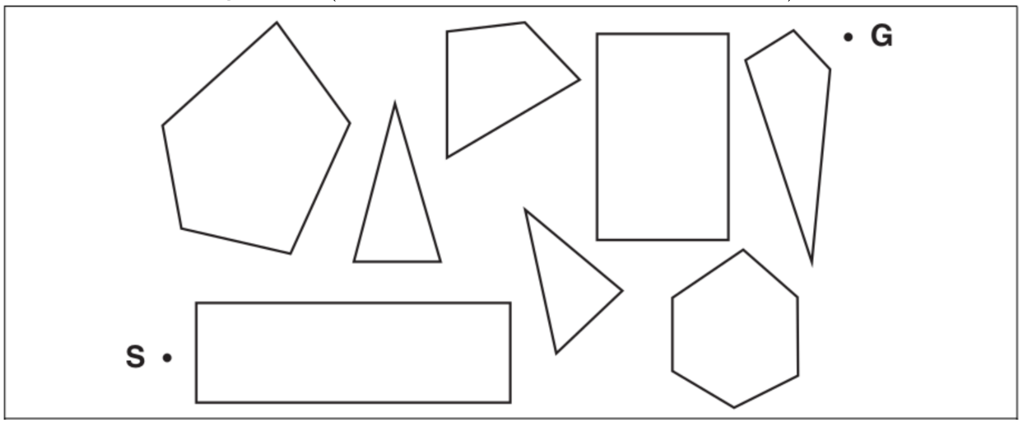
This gradient points in the direction of the steepest increase of the loss function L. In gradient descent, at each iteration t, we update $\hat{\vec{y}}^{(t)}$ in the opposite direction of the gradient to minimize L:

$$\hat{\vec{y}}^{(t+1)} = \hat{\vec{y}}^{(t)} - \alpha \nabla_{\hat{\vec{y}}^{(t)}} L$$

where α is the learning rate.

Question 2: Hill Climbing Optimality (25 points)

Consider the following world (shown from an arial top-down view):



a) Consider discretizing this world (i.e. laying a grid down on top of the world so we see a discrete coordinate system). Assume that the discretization is a fine as required (but not continuous) to preserve the geometry of the shapes present in the world. A *convex* shape is a shape where if we were to draw a line between any two points of the shape, the line would entirely be contained *within* the shape. Given a world that contains convex *obstacles*, is it possible for a vanilla hill climbing agent to get stuck?
In a world that contains convex obstacles, a vanilla hill climbing agent can still get stuck. Due to the limitation of the vanilla hill climbing algorithm, they are greedy and can get stuck in local maxima or at saddle points: specifically, they can stuck in places where the current solution isn't the final optimal one to the problem, but none of the neighbors of the current solution are better than the current one. If the agent is seeking a path from S to G, and the direct path is blocked by a convex obstacle, the agent will follow the obstacle boundary only if it leads to a current highest value of the objective function at each step. If the agent finds a local maximum along the boundary of an obstacle before reaching G, it may get stuck since a vanilla hill climbing algorithm does not backtrack or turn to explore alternative paths if it gets stuck in local maxima or at saddle points.

b) What if we could prove that the **objective surface** was convex. Would it be possible for a vanilla hill climbing agent to get stuck?
If the objective surface is convex, it means there are no local maxima other than the global maximum. The gradient at each point always points in the direction of the global maximum. From any starting point, any step taken in the direction indicated by the gradient will lead to an increase in the objective function's value, ultimately leading the agent towards the global maximum. Therefore, a vanilla hill climbing agent cannot get stuck.

Extra Credit: Gradient Descent and Lipshitz Continuity (50 points)

Gradient Descent is the continuous version of hill climbing where we use the gradient of the objective function to measure which direction leads uphill (we can then decide whether or not to follow the gradient uphill or downhill). Let us consider going downhill (i.e. we are choosing to *minimize*). The trouble is that gradient descent is a local search algorithm, meaning that it is not guaranteed to:

- Converge at all. There may not be any local minima to find.
- Arrive at the global minima if one exists.

A function f is *Lipschitz Continuous* with constant $k > 0$ if $\forall \vec{x}, \vec{y} \quad ||f(\vec{x}) - f(\vec{y})||_2 \leq k||\vec{x} - \vec{y}||_2$. Think of it this way: for every pair of points, f is bounded on how fast it can change by k. Here is a good gif that visualizes lipschitz continuity.

If we know that our objective function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, *and* we know that the gradient of f (i.e. ∇f) is Lipschitz continuous with (minimum) constant $c > 0$, prove that **not only** is gradient descent **guaranteed** to converge (for specific values of the step size η), but that it **will** converge if $\eta \leq \frac{2}{c}$.

Hint: you may find it useful to use a 2nd degree taylor polynomial of f centered around $f(\vec{x})$, which can be expressed as:

$$f(\vec{y}) \leq f(\vec{x}) + \nabla f(\vec{x})^T(\vec{y} - \vec{x}) + \frac{1}{2}(\vec{y} - \vec{x})^T \nabla^2 f(\vec{x})(\vec{y} - \vec{x})$$