

CSE 5819 Assignment #2

by: Aayushi Verma (uef24001)

This assignment is organized as follows:

1. ChatGPT - pg 1–6
2. Problems - pg 5–10

1 Part 1: ChatGPT Self-Learning (20pts)

You can start from the following prompts, but you need to create subsequent questions that attempt to understand the basic concepts of machine learning.

1.1 What is the definition of convex optimization?

Convex optimization is a subfield of optimization that deals with problems where the objective function is convex, and the feasible set is a convex set. Formally, a convex optimization problem can be expressed as:

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } x \in \mathcal{C}, \end{aligned}$$

where:

- $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a convex function.
- $\mathcal{C} \subseteq \mathbb{R}^n$ is a convex set.

A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if, for any two points $x_1, x_2 \in \mathbb{R}^n$ and any $\theta \in [0, 1]$, the following inequality holds:

$$f(\theta x_1 + (1 - \theta)x_2) \leq \theta f(x_1) + (1 - \theta)f(x_2).$$

A set \mathcal{C} is convex if, for any two points $x_1, x_2 \in \mathcal{C}$ and any $\theta \in [0, 1]$, the point $\theta x_1 + (1 - \theta)x_2$ is also in \mathcal{C} .

1.2 How to solve an unconstrained convex optimization problem?

To solve an unconstrained convex optimization problem, you need to minimize a convex objective function $f(x)$ over the variable $x \in \mathbb{R}^n$. The problem can be formulated as:

$$\text{minimize } f(x).$$

Here are common methods to solve this problem:

- **Gradient Descent:** This iterative method updates the solution x by moving in the direction of the negative gradient of the objective function. The update rule is:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k),$$

where α_k is the step size (or learning rate) and $\nabla f(x_k)$ is the gradient of f at x_k .

- **Newton's Method:** This method uses second-order information by incorporating the Hessian matrix $\nabla^2 f(x)$. The update rule is:

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k).$$

- **Conjugate Gradient Method:** This is suitable for large-scale problems where the Hessian is not explicitly available. It iteratively solves a sequence of linear systems and is effective for quadratic objective functions.
- **Quasi-Newton Methods:** These methods, such as BFGS, approximate the inverse Hessian matrix and update it iteratively, combining benefits of both gradient descent and Newton's method.
- **Subgradient Methods:** For nonsmooth convex functions, subgradient methods can be used, where the update rule is:

$$x_{k+1} = x_k - \alpha_k g_k,$$

where g_k is a subgradient of f at x_k .

1.3 How to solve a constrained convex optimization problem?

To solve a constrained convex optimization problem, you need to minimize a convex objective function $f(x)$ subject to convex constraints. The problem can be formulated as:

$$\begin{aligned} & \text{minimize } f(x) \\ & \text{subject to } x \in \mathcal{C}, \end{aligned}$$

where \mathcal{C} is a convex set. Common methods to solve this type of problem include:

- **Projected Gradient Descent:** This method involves iteratively applying gradient descent followed by projection onto the feasible set \mathcal{C} . The update rule is:

$$x_{k+1} = P_{\mathcal{C}}(x_k - \alpha_k \nabla f(x_k)),$$

where $P_{\mathcal{C}}$ denotes the projection operator onto the set \mathcal{C} .

- **Lagrange Duality:** Introduce Lagrange multipliers for the constraints and solve the Lagrangian dual problem. The Lagrangian is:

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^T (h(x) - b),$$

where $h(x)$ represents the constraint functions and λ are the Lagrange multipliers. Solve the dual problem to find the optimal multipliers and use them to obtain the solution to the primal problem.

- **Interior-Point Methods:** These methods solve constrained problems by transforming them into a series of unconstrained problems. They iteratively update the solution while maintaining feasibility with respect to the constraints.
- **Penalty Methods:** Convert the constrained problem into an unconstrained problem by adding a penalty term for constraint violations. The modified objective function is:

$$f(x) + \rho \sum_i \max(0, g_i(x))^2,$$

where $g_i(x) \leq 0$ are the inequality constraints and ρ is a penalty parameter.

- **Augmented Lagrangian Methods:** Combine ideas from Lagrange multipliers and penalty methods. The augmented Lagrangian function is:

$$\mathcal{L}_{\rho}(x, \lambda) = f(x) + \lambda^T (h(x) - b) + \frac{\rho}{2} \|h(x) - b\|^2,$$

where ρ is a penalty parameter and λ are the multipliers.

1.4 Would gradient descent converge if it is used to solve a non-convex optimization problem?

Gradient descent can be used to solve non-convex optimization problems, but its convergence properties are different from those in the convex case. Here are key points to consider:

- **Local Minima:** In non-convex optimization, gradient descent may converge to a local minimum rather than the global minimum. The final solution depends on the initial starting point and the nature of the objective function.
- **Saddle Points:** Gradient descent can also get stuck at saddle points, where the gradient is zero but the point is neither a local minimum nor a maximum.
- **Convergence Guarantees:** There are no general guarantees of convergence to a global minimum in non-convex problems. Convergence is usually to a local minimum or a stationary point, depending on the problem and algorithm specifics.
- **Choice of Step Size:** The choice of step size (learning rate) is crucial. A step size that is too large can lead to divergence or oscillations, while a step size that is too small can result in slow convergence.

- **Stochastic Gradient Descent (SGD):** In practice, techniques like stochastic gradient descent (SGD) and its variants (e.g., Adam, RMSprop) are often used. These methods can help escape local minima and saddle points by adding noise to the gradient updates.
- **Initialization and Restarts:** Multiple restarts with different initial points can increase the likelihood of finding a better local minimum. Proper initialization can also help avoid poor local minima.
- **Convexity Assumptions:** For guaranteed convergence to a global minimum, the objective function should be convex. In the absence of convexity, additional techniques and heuristics are required to handle the complexities of non-convex landscapes.

1.5 What loss functions will form convex functions?

A loss function $L(y, \hat{y})$ is convex in its parameters θ if for all possible parameter values θ_1 and θ_2 and for any $\lambda \in [0, 1]$, the following inequality holds:

$$L(\lambda\theta_1 + (1 - \lambda)\theta_2) \leq \lambda L(\theta_1) + (1 - \lambda)L(\theta_2).$$

Here are some commonly used convex loss functions:

- **Squared Error Loss (Mean Squared Error, MSE):**

$$L(y, \hat{y}) = (y - \hat{y})^2,$$

where y is the true value and \hat{y} is the predicted value. This loss function is convex with respect to \hat{y} .

- **Absolute Error Loss (Mean Absolute Error, MAE):**

$$L(y, \hat{y}) = |y - \hat{y}|,$$

where y is the true value and \hat{y} is the predicted value. This loss function is convex with respect to \hat{y} , but not differentiable at $\hat{y} = y$.

- **Logarithmic Loss (Log Loss or Binary Cross-Entropy):**

$$L(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})],$$

where $y \in \{0, 1\}$ is the binary label and \hat{y} is the predicted probability. This loss function is convex with respect to \hat{y} in the range $(0, 1)$.

- **Hinge Loss:**

$$L(y, \hat{y}) = \max(0, 1 - y\hat{y}),$$

where $y \in \{-1, 1\}$ is the true label and \hat{y} is the predicted value. This loss function is convex with respect to \hat{y} .

- **Kullback-Leibler Divergence (KL Divergence):**

$$L(P, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)},$$

where P and Q are probability distributions. This loss function is convex in Q for a fixed P .

- **Huber Loss:**

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |y - \hat{y}| \leq \delta, \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2 & \text{if } |y - \hat{y}| > \delta, \end{cases}$$

where δ is a threshold parameter. The Huber loss is convex with respect to \hat{y} and is used to be less sensitive to outliers compared to squared error loss.

2 Problems

[Convex Optimization, Mean/Medium/Mode, Preparation for Learning Loss Functions]

Take the training data set $D = \{(x_i, y_i): 1 \leq i \leq n\}$. Now consider the following hypothesis class (i.e., the set of constant functions)

$$H = \{h_z(x) = z, \forall x \in R\}$$

We will choose a model from this hypothesis class in our machine learning task.

2.1 Question One

1. [20 pts] Let us use L_2 vector norm to measure the discrepancy between the observed y and the model output $h_z(x)$. In other words, the squared loss is defined as

$$L_{sq}(h) = \frac{1}{n} \sum_{i=1}^n (h(x_i) - y_i)^2$$

Prove that if we set the constant z equal to the mean of y

$$m = \frac{1}{n} \sum_{i=1}^n y_i$$

this constant function $h_m(\cdot)$ minimizes the squared loss, and in other words,

$$h_m = \arg \min_{h \in H} L_{sq}(h)$$

2.2 Question Two

2. [20 pts] Let us use the L_1 vector norm to measure the discrepancy between the observed y and the model output $h_z(x)$. In other words, the absolute loss is defined as

$$L_{abs}(h) = \frac{1}{n} \sum_{i=1}^n |h(x_i) - y_i|$$

Prove that if we set the constant z equal to the median of y

$$m = \text{median}(\{y_i, \quad 1 \leq i \leq n\})$$

this constant function $h_m(\cdot)$ minimizes the absolute loss, and in other words,

$$h_m = \arg \min_{h \in H} L_{abs}(h)$$

2.3 Question Three

3. [20 pts] Let us use the L_0 vector norm to measure the discrepancy between the observed y and the model output $h_z(x)$. In other words, the binary loss is defined as

$$L_{bin}(h) = \frac{1}{n} \sum_{i=1}^n 1\{h(x_i) \neq y_i\}$$

where $1\{h(x_i) \neq y_i\}$ returns 1 if $h(x_i) \neq y_i$; or otherwise return 0. In other words, this loss counts how many times the model output $h(x_i)$ is not equal to the observed y_i . Then normalize the sum by n .

Prove that if we set the constant z equal to the mode of y

$$m = \text{mode}(\{y_i, \quad 1 \leq i \leq n\})$$

where the mode is the most common number in the observed y 's,

this constant function $h_m(\cdot)$ minimizes the binary loss, and in other words,

$$h_m = \arg \min_{h \in H} L_{bin}(h)$$