Gradient Descent

Data 100: Principles and Techniques of Data Science

Sandrine Dudoit

Department of Statistics and Division of Biostatistics, UC Berkeley

Spring 2019

Outline

Version: 14/03/2019, 18:09

Motivation

- Optimal statistical inference. A very broad class of statistical inference methods can be framed in terms of risk optimization.
- Least squares estimation (LSE) involves minimizing risk for the squared error loss function.
- Maximum likelihood estimation (MLE) involves minimizing risk for the negative log loss function.
- One can obtain closed-form expressions for risk minimizers for the squared error/L_2 and absolute error/L_1 loss functions: Means minimize mean squared error (MSE), while medians minimize mean absolute error (MAE).
- In general, however, there are no closed-form solutions for risk optimization, e.g., Huber loss function.

Motivation

- Instead, one can turn to numerical optimization methods such as gradient descent, simulated annealing, and genetic algorithms.

Optimization

- Suppose we wish to minimize the function $f : \mathbb{R}^J \to \mathbb{R}$, i.e., find
$$\text{argmin}_{\theta \in \mathbb{R}^J} f(\theta).$$
- The function f is referred to as objective function.
- In statistical inference, f typically corresponds to a risk function, i.e., the expected value of a loss function.
- The function f could be the empirical risk for the squared error loss function, i.e., the empirical mean squared error,
$$f(\theta) = R_2(P_n, \theta) = \frac{1}{n} \sum_{i=1}^{n} (X_i - \theta)^2,$$
where $\theta \in \mathbb{R}$.

Optimization

- The function f could be the empirical risk for the Huber loss function

$$f(\theta) = R_H(P_n, \theta) = \frac{1}{n} \sum_{i=1}^{n} L_H(X_i, \theta),$$

where

$$L_H(X, \theta) = \begin{cases} \frac{1}{2}(X - \theta)^2, & |X - \theta| \leq \delta \\ \delta \left(|X - \theta| - \frac{1}{2}\delta \right), & \text{otherwise} \end{cases},$$

$\theta \in \mathbb{R}$, and $\delta \in \mathbb{R}^+$ is a tuning parameter.

Gradient Descent Algorithm

- Gradient descent algorithms are iterative algorithms that seek to iteratively improve the solution to a particular optimization problem.
- That is, given a current estimate $\theta^{(t)}$, the algorithm aims to produce a next estimate $\theta^{(t+1)}$ such that $f(\theta^{(t)}) \geq f(\theta^{(t+1)})$.
- The intuition behind gradient descent algorithms is that the gradient (cf. slope) $\nabla_\theta f(\theta)$ suggests the direction in which to update θ.
  - If the gradient is negative, increase θ.
  - If the gradient is positive, decrease θ.
- Specifically, the gradient descent algorithm is as follows.
  1. Choose a starting value θ^0.

Gradient Descent Algorithm

2. Update θ according to the following iteration

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_\theta f(\theta^{(t)}), \tag{1}$$

   where α is a tuning parameter known as learning rate.
3. Repeat Step 2 until a stopping criterion is met.

- As with any iterative algorithm, important and practical decisions include the choice of starting value and stopping rule.
- Possible starting values can be obtained from loss functions that have closed-form expressions for their risk minimizer, e.g., means. Using multiple starting values is also advisable.
- Likewise, a variety of stopping rules can be used.
  - Stop after a fixed number of iterations.

Gradient Descent Algorithm

- Stop once θ doesn't change between iterations, i.e., $||\theta^{(t+1)} - \theta^{(t)}|| \leq \epsilon$ or $|\theta^{(t+1)} - \theta^{(t)}| \leq \epsilon_1(|\theta^{(t)}| + \epsilon_2)$ when elements of θ are of different magnitues.
- Stop once the objective function doesn't change between iterations, i.e., $|f(\theta^{(t+1)}) - f(\theta^{(t)})| \leq \epsilon$.

- The higher the learning rate α, the more "aggressive" the moves, at the risk of overshooting the minimum. The smaller the learning rate, the more precise the moves, but the more time-consuming the implementation.

Gradient Descent Algorithm

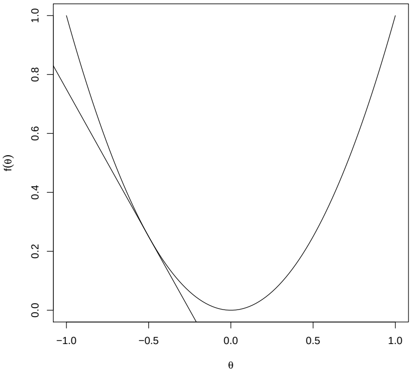

Figure 1: *Gradient descent.* The slope of the tangent line tells us in which direction to update θ in order to decrease the objective function.

Stochastic Gradient Descent Algorithm

- With the above gradient descent algorithm, the gradient is computed for empirical risk based on the entire learning set

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L(X_i, \theta^{(t)}).$$

- Such an approach, known as batch gradient descent, can be computationally inefficient for large datasets.
- An alternative, known as stochastic gradient descent (SGD), is to compute the gradient for a randomly chosen observation X_i, that is, have the updates

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_\theta L(X_i, \theta^{(t)}).$$

Stochastic Gradient Descent Algorithm

- Stochastic gradient descent often takes steps away from the optimum, but makes more aggressive updates and often converges faster than batch gradient descent.
- Mini-batch gradient descent strikes a balance between batch gradient descent and stochastic gradient descent by using a random sample of several observations for each update.

Convexity

- Not all functions are equally easy to optimize.
- The empirical MSE has a unique global minimizer, the mean.

$$\bar{X}_n = \text{argmin}_{\theta \in \mathbb{R}} R_2(P_n, \theta) = \text{argmin}_{\theta \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} (X_i - \theta)^2.$$

- The empirical MAE could have multiple minima, the median, but these are global minima.

$$\tilde{X}_n = \text{argmin}_{\theta \in \mathbb{R}} R_1(P_n, \theta) = \text{argmin}_{\theta \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} |X_i - \theta|.$$

- Although there is no closed-form expression for the Huber risk minimizer, it is unique.

Convexity

- The above loss functions are convex functions of the parameter θ.
- A function f is convex if and only if it satisfies the following inequality

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y). \qquad (2)$$

  That is, all lines connecting two points of the function must reside on or above the function itself.
- For convex functions, any local minimum is also a global minimum.
- Convexity of a loss function allows gradient descent to efficiently find the global risk minimizer.
- While gradient descent will converge to a local minimum for non-convex loss functions, these local minima are not guaranteed to be globally optimal.

Convexity

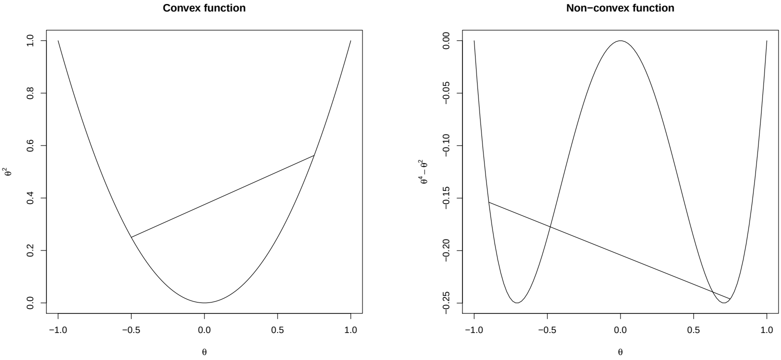

Figure 2: *Convexity.*