Maximum Likelihood Estimation and Gradient Descent

Shreeyesh Menon

1 Introduction

Suppose we observe data $X_1, \ldots, X_n$ that we believe are generated according to a parametric statistical model with parameter θ. The fundamental inferential question is: *what value of θ best explains the observed data?* This brings us to the concept of *likelihood*.

2 Likelihood Function

Let $f(x \mid \theta)$ denote the probability density (or mass) function of the model. Assuming independent observations, the joint density of the data is

$$L(\theta) = f(x_1, \ldots, x_n \mid \theta) = \prod_{i=1}^{n} f(x_i \mid \theta),$$

which we call the *likelihood function*.

2.1 Maximum Likelihood Estimation

Now, given the observations, what would be a good guess for the underlying parameter θ that generated the data? One way to arrive at a guess would be to maximize the likelihood function over all possible θ and select the one at the optimum.

The *maximum likelihood estimator* is defined as

$$\hat{\theta} = \arg\max_{\theta} L(\theta).$$

Because the logarithm is monotone and often simplifies products, we usually maximize the *log-likelihood*:

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log f(x_i \mid \theta).$$

2.2 Example: Gaussian Mean

If $X_i \sim \mathcal{N}(\mu, \sigma^2)$ with σ^2 known, then

$$\ell(\mu) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2.$$

Maximizing this is equivalent to minimizing the sum of squared deviations. Taking derivative and setting it to zero:

$$\frac{\partial \ell}{\partial \mu} = \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \mu) = 0 \quad \Rightarrow \quad \hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

This example shows that closed-form solutions are sometimes available. However, in many realistic models they are not.

3 Estimation Problems are Optimization Problems

MLE reduces statistical estimation to solving a maximization problem:

$$\hat{\theta} = \arg\max_{\theta} \ell(\theta).$$

Equivalently, we can consider minimizing the negative log-likelihood:

$$J(\theta) = -\ell(\theta).$$

This transformation casts estimation as an optimization problem.

3.1 When Closed-Form Solutions Fail

In simple models (Gaussian mean, linear regression), derivatives lead to an explicit solution. But many important models—logistic regression, neural networks, mixture models—yield nonlinear objective functions for which:

- the derivative equations have no closed-form solution,
- the likelihood is non-convex or high-dimensional,
- the Hessian is expensive to compute.

In such settings, iterative numerical optimization becomes essential.

4 Gradient-Based Optimization

The central idea is: if we cannot solve for the optimum analytically, we can *iteratively move toward it*.

4.1 Gradient Descent Algorithm

Consider minimizing $J(\theta)$. The gradient gives the direction of steepest increase; hence the steepest *decrease* is $-\nabla J(\theta)$.

Gradient Descent updates parameters iteratively:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla J(\theta^{(t)}),$$

where $\alpha > 0$ is the *learning rate*.

Interpretation:

- The update moves θ in the direction that most rapidly reduces the objective.
- The learning rate controls the step size: too large leads to divergence, too small leads to slow convergence.
- GD is particularly powerful when $J(\theta)$ is a sum of many terms, such as a log-likelihood over many samples.

4.2 MLE using Gradient Descent

When we minimize negative log-likelihood,

$$J(\theta) = -\sum_{i=1}^{n} \log f(x_i \mid \theta),$$

the gradient is

$$\nabla J(\theta) = -\sum_{i=1}^{n} \nabla \log f(x_i \mid \theta).$$

Thus gradient descent becomes a computational tool to solve MLE problems:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha \sum_{i=1}^{n} \nabla \log f(x_i \mid \theta^{(t)}).$$

4.3 Variants

- **Stochastic Gradient Descent (SGD)**: uses one observation at a time.
- **Mini-batch GD**: compromise between deterministic and stochastic.
- **Momentum methods**: accelerate convergence.
- **Adam, RMSProp, AdaGrad**: adaptive learning-rate algorithms.

Gradient descent is a powerful algorithm that can be used in any problem that allows for the loss function to be written as a differentiable function of the model parameters.