Cost Function Derivation and Gradient Descent Algorithm

Cost Function Derivation

So far this summer, I have been deep-diving into Machine Learning—not just using the libraries for business solutions, but actually understanding the math behind the algorithms. My journey has been a bit random, and I haven't always been consistent, but every now and then I stumble upon something that makes me stop and think.

One of those questions: Why do we square errors in the cost function

$$\frac{1}{2m}\sum(y' - y)^2?$$

And why do we divide by $2m$?

Was this formula derived mathematically or developed out of intuition? It's a little of both. Here's how a mathematician may have thought through it:

The first question to answer is: *How can we measure how far a model's prediction is from the true value?* A natural approach is to consider the difference $y' - y$ as a measure of the extent to which a prediction diverges from reality. Summing these errors directly would cancel out positive and negative errors, leading to misleading conclusions.

One possible solution is to take the modulus $|y' - y|$ of differences, ensuring all errors contribute positively. However, since absolute values are non-differentiable at zero, optimization becomes complex.

Squaring the errors, $(y' - y)^2$, solves this issue. It makes all errors positive, provides a smooth and differentiable curve everywhere, and leads to a unique minimum. Higher powers, like cubing, reintroduce the issue of negative values, while higher even powers increase complexity unnecessarily. Thus, squaring is a natural choice.

To assess the model's performance on the entire dataset, we compute the average of the squared errors:

$$\frac{1}{m}\sum(y' - y)^2.$$

Finally, the factor $\frac{1}{2}$ simplifies derivative calculations. When taking derivatives of the squared error, a factor of 2 emerges. Including $\frac{1}{2}$ cancels this factor, streamlining computations.

Gradient Descent Algorithm

Having explored error calculation, our next step is to minimize these errors to improve prediction accuracy.

In calculus, finding the minimum of a function involves using derivatives. When the derivative isn't straightforward, numerical methods, like the Gradient Descent Algorithm, are employed. Here's how Gradient Descent works for a simple single-variable function $f(x)$:

$$x_n = x_{n-1} - \alpha f'(x_{n-1}),$$

where:

- x_n is the updated value of x,
- α is the learning rate,
- $f'(x_{n-1})$ is the derivative of the function at x_{n-1}.

This formula ensures the parameter moves in the direction of steepest descent. Larger slopes result in bigger steps, while smaller slopes slow down progress, ensuring convergence near the minimum.

The learning rate α controls step size:

- Large α: faster but may overshoot the minimum.
- Small α: slower but more precise.

Multivariable Gradient Descent

In machine learning, cost functions often depend on multiple parameters, leading to multivariable gradient descent. The cost function $J(\theta_0, \theta_1, \ldots, \theta_n)$ is minimized iteratively by updating each parameter:

$$\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \ldots, \theta_n),$$

where $\frac{\partial}{\partial \theta_i} J$ is the partial derivative of J with respect to θ_i.

Example: Linear Regression

Consider linear regression with parameters m (slope) and b (y-intercept). The hypothesis is:

$$h_\theta(x) = mx + b.$$

The cost function is:

$$J(m, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2.$$

The gradient descent updates for m and b are:

$$m = m - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)},$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right).$$

Conclusion

Gradient descent is a fundamental optimization tool in machine learning, capable of handling multiple parameters. Its success depends on an appropriate learning rate and efficient preprocessing, such as feature scaling. With these in place, gradient descent enables powerful and practical optimization.