What is a Cost Function?

22.06.2025

* In many ML problems, our goal is to create a model that represents a trend in our data. For example, in a regression task, we might want to fit a line through a scatter plot of data points.

Job of a cost function

* Out of the infinite possible lines we could draw, how does a computer determine which one is the absolute "best fit"? How does it measure the error of a bad line vs. a good one? This is the job of the cost function (aka loss function or objective function).

Definition of a cost function

* A cost function is a formula that measures the "cost" or "error" of a model. It calculates a single number that represents how wrong the model's predictions were, compared to the actual, true values.
* The goal of training a model is to find the set of internal parameters (which define the model's shape and position) that results in the lowest possible cost. In simple terms: smaller cost = better model

Example

* Imagine you're playing darts. Your goal is to hit the bullseye. The cost function is like a rule that says, "The cost of your throw is the distance from your dart to the bullseye." A perfect shot that hits the bullseye has a cost of zero. A dart that lands far away from the bullseye has a high cost. An ML model is like a darts player, constantly trying to adjust its aim (the parameters) to make the cost (the error) as low as possible.

* To train an ML model, we first quantify its error using a cost function (or loss function), which calculates a single score of how wrong its predictions are. The process of finding the model parameters that minimize this score is called Optimization.

Common cost functions

* There are many types of cost functions, each suited for different problems. You don't need to know their formulas yet, but it's good to recognize their names and what they are used for:
  * Means Squared Error (MSE): The most common cost function for regression models, which measures the average of squared distances between predicted and actual numerical values.
  * Binary Cross-Entropy (or Log Loss): The most common cost function for classification problems with two outcomes, which quantifies how well a model's predicted probabilities match the true class labels (0 or 1).

Optimization and Gradient Descent

22.06.2025

What is optimization

* Optimization is the process of systematically and efficiently finding the set of model parameters that minimizes the value of the cost function. While there are several optimization algorithms, the most fundamental and widely used is called Gradient Descent.

A real-world analogy: finding the bottom of the valley

* To understand Gradient Descent, imagine you are a hiker, blindfolded, standing somewhere on the side of a huge, foggy mountain valley.
* The valley represents the cost function. The height at any point in the valley is the "cost" or "error" for a specific set of your parameters.
* Your location on the mountainside represents the current values of your model's parameters (e.g. the current slope m and intercept b of a line).
* Your goal is to get to the absolute lowest point in the valley, the point where the cost is at its minimum.

Gradient Descent

* Since you're blindfolded, you can't see the bottom of the valley directly. What can you do? You can feel the ground right where you are standing.

① Feel the slope: You tap your foot around to determine which direction is the steepest downhill path from your current spot. In mathematical terms, this direction of steepest descent is the negative of the gradient.

* The most common optimization algorithm, Gradient Descent, achieves this by iteratively taking small steps in the steepest "downhill" direction of the cost function.

(2) Take a step: You take one small, deliberate step in that downhill direction.

(3) Repeat: You are now at a new, slightly lower spot in the valley. You repeat the process: feel the slope again, take another small step in the new steepest downhill direction, and so on.

* By repeating this simple process over and over, you will eventually walk your way down the side of the mountain and settle at the very bottom.

The Learning Rate Hyperparameter

22.06.2025

What is the learning rate?

* It is a hyperparameter, that you, the data scientist set before training. It controls the size of steps the optimization algorithm takes at each iteration as it moves toward the minimum of the cost function.
* Finding a good learning rate is a "Goldilocks" problem: it can't be too big or too small; it needs to be "just right". Let's go back to our hiker analogy.

Real-world analogy: Learning rate too large

* A large learning rate (taking giant leaps): Imagine the hiker tries to get to the bottom by taking huge leaps. They might cover ground quickly, but they could easily "overshoot" the lowest point and land on the other side of the valley, possibly even higher up than they started.
* In ML, if the learning rate is too high, the model's cost can bounce around erratically and may never find the best solution (it fails to converge).

Learning rate too small

* A small learning rate (Taking tiny steps): Imagine the hiker takes tiny, shuffling steps. They are almost guaranteed to eventually reach the bottom, but it could take an incredibly long time and thousands of steps.
* In ML, if the learning rate is too low, the model's training process will be very slow. In some complex scenarios, it might even get "stuck" in a small local dip without having the momentum to reach the true, lowest point of the valley.

* The size of these steps is controlled by the learning rate, a crucial hyperparameter that must be set carefully to ensure the model finds the minimum more efficiently without overshooting the solution.

Optimal learning rate

* A good learning rate is the step size that is large enough to make progress efficiently but small enough that it doesn't overshoot the target.
* In ML, a good learning rate allows the model to find the minimum cost reliably and in a reasonable amount of time.