Gradient descent

Posted Oct 3, 2022

By Davide Evangelista

5 min read

Optimization in Machine Learning

Training a Machine Learning model usually ends up to an *optimization problem*. As an example, we already saw that LDA algorithm requires to solve

$$\max_q q^T S_w q \qquad \text{s.t. } q^T S_b q = 1$$

Moreover, it can be shown that PCA is equivalent to find the directions $q_1, \dots, q_K$ with $||q_i|| = 1$ for $i = 1, \dots, K$, over which the spread of the data is maximized. Mathematically, this can be re-written as

$$\max_{q:||q||_2=1} ||q^T X||_2$$

but

$$\max_{q:||q||_2=1} ||q^T X||_2 = \max_{q:||q||_2=1} \frac{||q^T X||_2}{||q||_2} = ||X||_2$$

by the definition of matrix norm. The relationship between PCA and SVD comes from the fact that $||X||_2 = \sigma_1$, where σ_1 is the largest eigenvalue of X.

Introduction to Optimization

From the examples above, it is clear that it is necessary to find a *general* way of solving optimization algorithm. For the sake of generality, from now on we will consider a general *continuous* function $f : \mathbb{R}^n \to \mathbb{R}$. Given a set of *admissible solutions* $\Omega \subseteq \mathbb{R}^n$, an optimization problem is a problem of the form

$$\min_{x \in \Omega} f(x)$$

If $\Omega = \mathbb{R}^n$, we say that the optimization is **unconstrained**. If $\Omega \subset \mathbb{R}^n$, the problem is **constrained**. In this first part, we will always assume $\Omega = \mathbb{R}^n$, i.e. the unconstrained setup.

Convexity

A common assumption in optimization is the *convexity*. By definition, a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2) \qquad \forall x_1, x_2 \in \mathbb{R}^n, \quad \forall t \in [0, 1]$$

The importance of convex functions in optimization is that if $f(x)$ is convex, than every minimum point x^* of $f(x)$ is a *global minimum* and the set of global minima is connected (the n-dimensional extension of the concept of *interval*). On the other side, if a function is non-convex (*NOTE:* the opposite of convex is **not** concave), then there can be multiple distinct minimum points, some of which are *local minimum* while others are *global minimum*. Since in ML applications we want to find global minima, and discriminating between local and global minima of a function is an NP-hard problem, having convexity is a good thing.

First-order sufficient condition

Most of the algorithms to find the minimum points of a given function $f(x)$ are based on the following property:

> ***First-order sufficient condition:*** If $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function and $x^* \in \mathbb{R}^n$ is a minimum point of $f(x)$, then $\nabla f(x^*) = 0$.

Moreover, it holds

> ***First-order necessary condition:*** If $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function and $\nabla f(x^*) = 0$ for $x^* \in \mathbb{R}^n$, then x^* is either a (local) minimum, a (local) maximum or a saddle point of $f(x)$.

Consequently, we want to find a point $x^* \in \mathbb{R}^n$ such that $\nabla f(x^*) = 0$. Those points are called **stationary points** of $f(x)$.

Gradient descent (GD)

The most common algorithm to solve optimization problems is the so-called Gradient Descent (GD). It is an iterative algorithm, i.e. an algorithm that iteratively updates the estimate of the solution, converging to the correct solution after infinite steps, such that, at each iteration, the successive estimate is computed by moving in the direction of maximum decreasing of $f(x)$: $-\nabla f(x)$. Specifically,

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \qquad k = 0, 1, \dots$$

where the initial iterate, $x_0 \in \mathbb{R}^n$, is given as input and the **step-size** (equivalently, **learning rate**) $\alpha_k > 0$ controls the decay rapidity of $f(x)$ for any $k \in \mathbb{N}$.

Choice the initial iterate

The Gradient Descent (GD) algorithm, always require the user to input an initial iterate $x_0 \in \mathbb{R}^n$. Theoretically, since GD has a *global convergence* proprerty, for any x_0 it will always converge to a **stationary point** of $f(x)$, i.e. to a point such that $\nabla f(x) = 0$.

If $f(x)$ is convex, then every stationary point is at (global) minimum of $f(x)$, implying that the choice of x_0 is not really important, and we can always use $x_0 = 0$. On the other side, when $f(x)$ is not convex, we have to choose x_0 such that it is as close as possible to the *right* stationary point, to increase the chances of getting to that. If an estimate of the correct minimum point is not available, we will just consider $x_0 = 0$ to get to a general local minima.

Step Size

Choosing the step size is the hardest component of gradient descent algorithm. Indeed, if α_k is too small, there is a chance we never get to the minimum, getting closer and closer without reaching it. Moreover, we can easily get stuck on local minima when the *objective function* is non convex. On the contrary, if α_k is too large, there is a chance we get stuck, bouncing back and forth around the minima.

Stopping Criteria

The gradient descent is an iterative algorithm, meaning that it iteratively generates new estimates of the minima, starting from x_0. Theoretically, after infinite iterations, we converge to the solution of the optimization problem but, since we cannot run infinite iterations, we have to find a way to tell the algorithm when its time to stop. A convergence condition for an iterative algorithm is called **stopping criteria**.

Remember that gradient descent aim to find stationary point. Consequently, it would make sense to use the norm of the gradient as a stopping criteria. In particular, it is common to check if the norm of the gradient on the actual iterate is below a certain tollerance and, if so, we stop the iterations. In particular

> ***Stopping criteria 1:*** Given a tollerance tol_f, for any iterate x_k, check whether or not $||\nabla f(x_k)|| < tol_f||\nabla f(x_0)||$. If so, stop the iterations.

Unfortunately, this condition alone is not sufficient.

Lab3, teaching

gradient descent optimization minimization

Further Reading

Sep 15, 2022
Introduction to Python for Linear Algebra
Introduction Numerical Linear Algebra (NLA) is the study of how matrix operations can …

Sep 15, 2022
Plotting with matplotlib
Plotting Visualization in Python can be performed by a famous library named…

Sep 16, 2022
Linear Systems with Numpy and Scipy
Introduction In the following we want to study how to use numpy and scipy to solv…

OLDER
Classification with clustering algorithms

NEWER
-