CS 760: Machine Learning Fall 2020

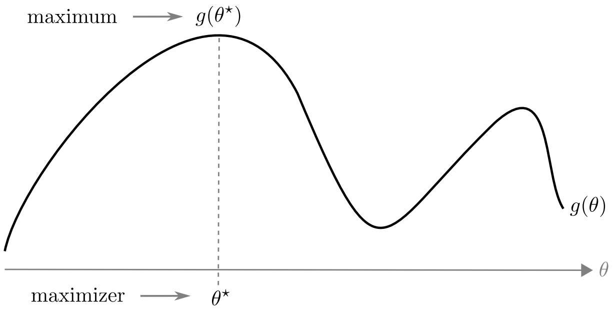
Topic 4: Review of Optimization 101

Instructor: Daniel L. Pimentel-Alarcón

GO GREEN. AVOID PRINTING, OR PRINT 2-SIDED OR MULTIPAGE.

4.1 Introduction

Most machine learning problems can be posed as finding the *maximizer* of a function $g(\theta)$, that is, the value $\theta^\star$ such that $g(\theta^\star) \geq g(\theta)$ for every θ in the domain of g:



Example 4.1. Suppose θ denotes the moment of your life when you stop studying, and start working, e.g., after high school, after college, after a masters, after a Ph.D, after a postdoc, or somewhere in between. Let g be the amount of money that you will earn throughout your life as a function of θ. The more you study, the higher pay you'll earn when you start working; on the other hand, the sooner you start working, the more experience you'll gain, the sooner you can get a promotion and a raise. You want to find the sweet spot (maximizer) $\theta^\star$ that produces the maximum pay $g(\theta^\star)$.

4.2 Optimizing Simple Concave Functions

If g is concave and *simple* enough, $\theta^\star$ can be determined using our elemental calculus recipe:

1. Take derivative of $g(\theta)$
2. Set derivative to zero, and solve for the maximizer.

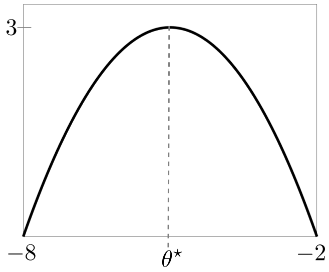
Example 4.2. Consider $g(\theta) = 3 - (\theta + 5)^2$. We can follow our recipe to find its maximizer:

1. The derivative of g is given by $\nabla g(\theta) = -2(\theta + 5)$.

2. Setting the derivative to zero and solving for θ we obtain:

$$\begin{aligned} -2(\theta+5) &= 0 \\ \theta &= -5. \end{aligned}$$

Since g is concave (can you show this?), we conclude that its maximizer is $\theta^\star = -5$, as depicted below:



4.3 Matrix Derivatives

In general, g will not always be a function as simple as in Example 4.2. In fact, in most machine learning problems, g will be a complex multivariate function in matrix form, for example:

$$g(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^\mathsf{T}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}),$$

where $\mathbf{y}$ and $\boldsymbol{\theta}$ are vectors, and $\mathbf{X}$ is a matrix. If we want to optimize $g(\boldsymbol{\theta})$, we need to take the derivative with respect to a vector, or more generally, with respect to a matrix.

To learn more about how to take derivatives w.r.t. vectors and matrices I recommend taking a look at *Old and new matrix algebra useful for statistics* by Thomas P. Minka, which shows how to take derivatives of some matrix functions, such as:

$$\begin{aligned} g(\boldsymbol{\theta}) &= \boldsymbol{\theta}^\mathsf{T}\mathbf{A} & \Rightarrow && g'(\boldsymbol{\theta}) &= \mathbf{A}, \\ g(\boldsymbol{\theta}) &= \boldsymbol{\theta}^\mathsf{T}\mathbf{A}\boldsymbol{\theta} & \Rightarrow && g'(\boldsymbol{\theta}) &= 2\mathbf{A}\boldsymbol{\theta}. \end{aligned}$$

4.4 Gradient Ascent

Some functions, however, are too complex to solve for θ in step 2. For example, consider the following function that describes the likelihood of a Bernoulli(θ) random variable:

$$g(\boldsymbol{\theta}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{y_i} \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i}}}\right) + (1-\mathrm{y_i})\log\left(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i}}}\right).$$

Its gradient is given by:

$$g'(\boldsymbol{\theta}) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(\mathrm{y_i} - \frac{1}{1+e^{-\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i}}}\right)\mathbf{x}_\mathrm{i}.$$

If we set this to zero, can you solve for $\boldsymbol{\theta}$?

For cases where our calculus 101 recipe does not work, we use *optimization*, which is the field of mathematics that deals with finding maximums (and minimums). In particular, we will use one of the most elemental tools of optimization: gradient ascent (resp. descent).

The setting is is this: you have a function $g(\theta)$. You want to find its maximum. You cannot solve for it directly using the derivative trick, so what can you do? You can *test* the value of g for different values of θ. For example, you can test $g(0)$, then maybe $g(1)$, then maybe $g(-1)$, then maybe $g(1.5)$, and so on, until you find the maximizer. Of course, depending on the domain of g, there could be infinitely many options, so testing them all would be infeasible.

As the name suggests, the main idea of gradient ascent is to test some initial value θ_0 (for example 0), and iteratively use the gradient (another name for derivative) to determine which value of θ to test next, such that the each new value $\theta_{\text{t}+1}$ produces a higher value for g, until we find the maximum. The main intuition is that the gradient $\nabla g(\theta)$ tells us the slope of g at θ. If this slope is positive, then we know that g is increasing, and we should try a larger value of θ, say $\theta_{\text{t}+1} = \theta_{\text{t}} + \eta$, where η is often referred to as *step-size*. If the slope is negative, then we know that g is decreasing, and we should try a smaller value of θ, say $\theta_{\text{t}+1} = \theta_{\text{t}} - \eta$ (see Figure 4.1 to build some intuition).

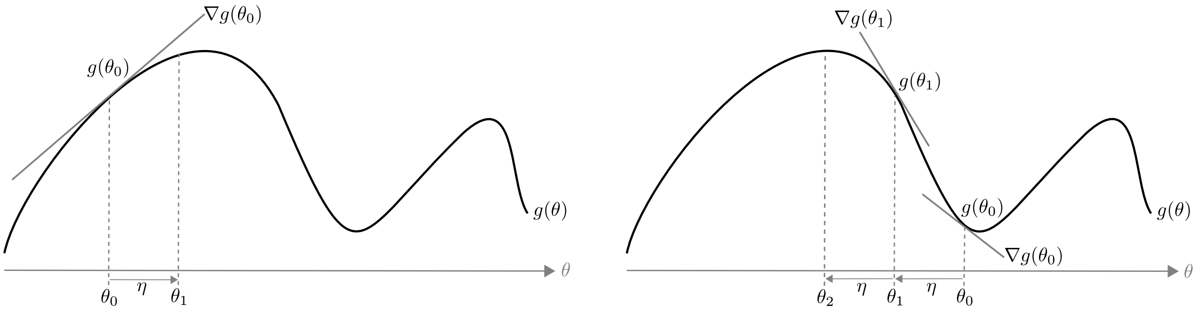


Figure 4.1: Start at some point θ_0. If the gradient is positive (left figure), try a larger value of θ, say $\theta_1 = \theta_0 + \eta$. If the gradient is negative (right figure), try a smaller value of θ, say $\theta_1 = \theta_0 + \eta$. Repeat this until convergence.

The same insight extends to multivariable functions. If g is a function of a vector $\boldsymbol{\theta} \in \mathbb{R}^{\text{D}}$, then $\nabla g(\boldsymbol{\theta}) \in \mathbb{R}^{\text{D}}$ gives the slope of g in each of the D coordinates of $\boldsymbol{\theta}$. Based on this insight, gradient ascent can be summarized as follows:

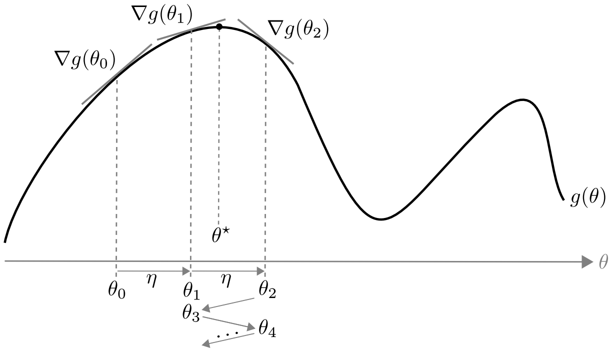
```
Algorithm 1: Gradient Ascent
Input: Function $g$, step-size parameter $\eta > 0$.
Initialize $\boldsymbol{\theta}_0$. For example, $\boldsymbol{\theta}_0 = \mathbf{0}$.
Repeat until convergence: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \nabla g(\boldsymbol{\theta}_t)$.
Output: $\boldsymbol{\theta}^\star = \boldsymbol{\theta}_t$.
```

4.4.1 Step-size η

The keen reader will be wondering, what if we move too far? In our example of Figure 4.1, we could run into an infinite loop, where

$$\begin{aligned} \theta_1 &= \theta_3 = \theta_5 = \theta_7 = \cdots \\ \theta_2 &= \theta_4 = \theta_6 = \theta_8 = \cdots, \end{aligned}$$

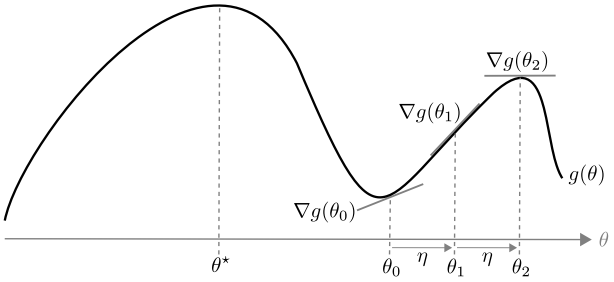
without ever achieving $\theta^\star$, as depicted below:



How would you solve this?

4.4.2 Initialization

The keen reader will also be wondering: what if we start at the wrong place, as depicted below:



In cases like these we could run into a so-called local maximum, that is, a point that is larger than all other points in its vicinity, but not necessarily the maximum over the whole domain of g. In the figure above, θ_2 is a local maximizer.

How would you solve this?

4.4.3 Minimization

How would things change if you wanted to minimize, rather than maximize?