

Topic 4: Review of Optimization 101

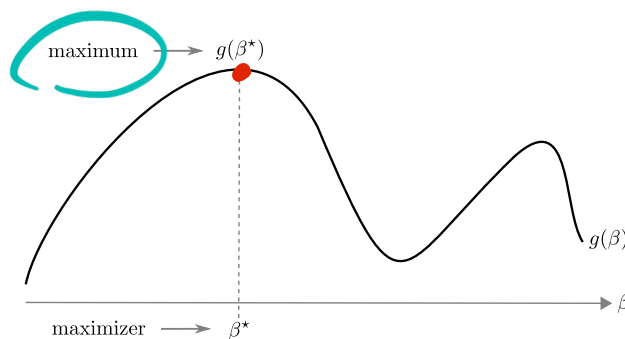
INSTRUCTOR: DANIEL L. PIMENTEL-ALARCÓN

© COPYRIGHT 2020

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 $g(\beta)$, that is, the value β^* such that $g(\beta^*) \geq g(\beta)$ for every β in the domain of g :



Example 4.1. Suppose β 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 β . 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) β^* that produces the maximum pay $g(\beta^*)$.

4.2 Optimizing Simple Concave Functions

凹函数

If g is concave and *simple* enough, β^* can be determined using our elemental calculus recipe:

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

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

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

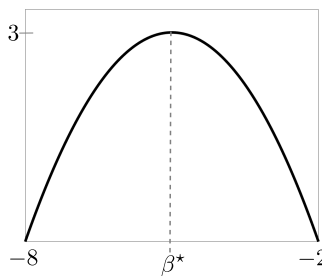
导数

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

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

令导数=0

Since g is concave (can you show this?), we conclude that its maximizer is $\beta^* = -5$, 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:

$$g(\beta) = (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta),$$

where \mathbf{y} and β are vectors, and \mathbf{X} is a matrix. If we want to optimize $g(\beta)$, 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:

$$\begin{aligned} g(\beta) &= \beta^\top \mathbf{A} & \Rightarrow & g'(\beta) = \mathbf{A}, \\ g(\beta) &= \beta^\top \mathbf{A} \beta & \Rightarrow & g'(\beta) = 2\mathbf{A}\beta. \end{aligned}$$

4.4 Gradient Ascent

梯度下降

复杂 function 难以计算

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

$$g(\beta) = \sum_{i=1}^n y_i \log \left(\frac{1}{1 + e^{-\beta^\top \mathbf{x}_i}} \right) + (1 - y_i) \log \left(1 - \frac{1}{1 + e^{-\beta^\top \mathbf{x}_i}} \right).$$

Its gradient is given by:

$$g'(\beta) = \sum_{i=1}^N \left(y_i - \frac{1}{1 + e^{-\beta^\top \mathbf{x}_i}} \right) \mathbf{x}_i.$$

If we set this to zero, can you solve for β ?

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 this: you have a function $g(\beta)$. 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 β . 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 β_0 (for example 0), and iteratively use the gradient (another name for derivative) to determine which value of β to test next, such that the each new value β_{t+1} produces a higher value for g , until we find the maximum. The main intuition is that the gradient $\nabla g(\beta)$ tells us the slope of g at β . If this slope is positive, then we know that g is increasing, and we should try a larger value of β , say $\beta_{t+1} = \beta_t + \eta$, 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 β , say $\beta_{t+1} = \beta_t - \eta$ (see Figure 4.1 to build some intuition).

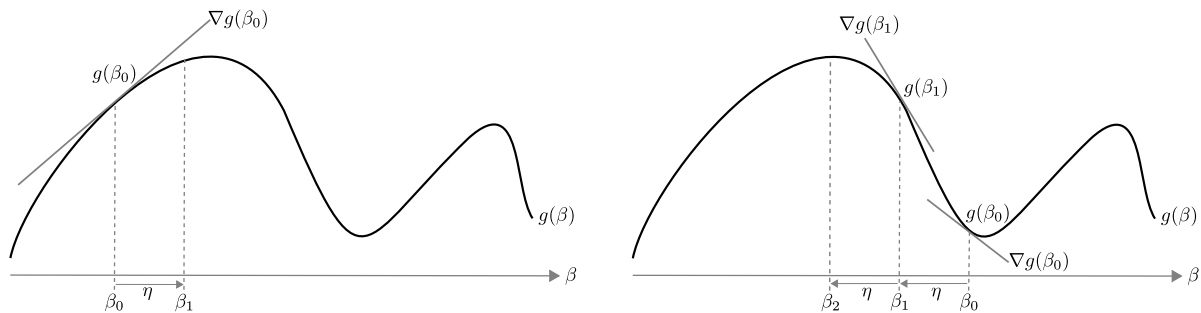


Figure 4.1: Start at some point β_0 . If the gradient is positive (left figure), try a larger value of β , say $\beta_1 = \beta_0 + \eta$. If the gradient is negative (right figure), try a smaller value of β , say $\beta_1 = \beta_0 - \eta$. Repeat this until convergence.

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

Algorithm 1: Gradient Ascent

Input: Function g , step-size parameter $\eta > 0$.

Initialize β_0 . For example, $\beta_0 = \mathbf{0}$.

Repeat until convergence: $\beta_{t+1} = \beta_t + \eta \nabla g(\beta_t)$.

Output: $\beta^* = \beta_t$.

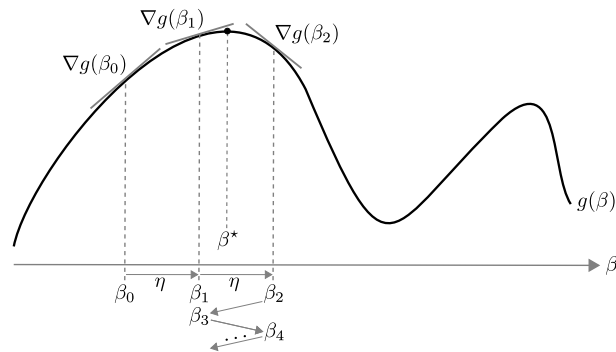


4.4.1 Step-size η

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

$$\begin{aligned}\beta_1 &= \beta_3 = \beta_5 = \beta_7 = \dots \\ \beta_2 &= \beta_4 = \beta_6 = \beta_8 = \dots,\end{aligned}$$

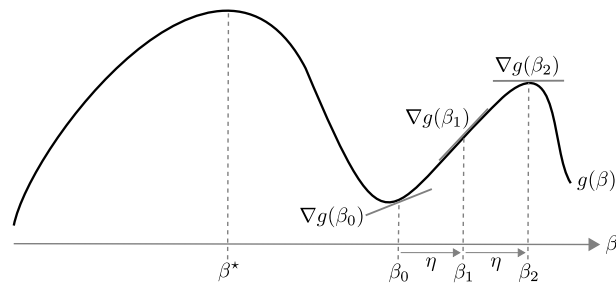
without ever achieving β^* , as depicted below:



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, β_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?