1 Gradient

1.1 What is Gradient

Gradient is generalization of derivative for functions on $\mathbb{R}^n$. While the derivative describes the slope of the function in 1-dimensional case, gradient indicates both the slopes (along different axes) and the direction of the steepest ascent in case of n-dimensional functions. Below, we only look at the scalar functions $\mathbb{R}^n \to \mathbb{R}$, i.e. the function that compute into a single number for n-dimensional input.

Perhaps the easiest way to understand this is to think about a hilly landscape: the gradient value in a given direction tells us how fast the terrain is going up in that direction (or pehaps going down instead, if it is negative), while the direction of the gradient vector tells which way the climb is the steepest.

In a way, gradient is just a habit to stack the set of derivatives in a compact form: in n-dimensional space, we have n partial derivatives $\frac{\partial}{\partial x_1} f(\boldsymbol{x})$, $\frac{\partial}{\partial x_2} f(\boldsymbol{x})$, $\dots \frac{\partial}{\partial x_n} f(\boldsymbol{x})$. For instance, for a function g

$$g(\boldsymbol{x}) = x_1 \cdot \log x_2, \tag{1}$$

the partial derivatives are $\frac{\partial}{\partial x_1} g(\boldsymbol{x}) = \log x_2$ and $\frac{\partial}{\partial x_2} g(\boldsymbol{x}) = x_1 / x_2$.

In a way, gradient, commonly denoted by $\nabla g(\boldsymbol{x})$ or sometimes $\partial f(\boldsymbol{x}) / \partial \boldsymbol{x}$, is just a compact way to write this in vector form:

$$\nabla g(\boldsymbol{x}) = \begin{pmatrix} \log x_2 \\ x_1/x_2 \end{pmatrix}. \tag{2}$$

or in general n-dimensional case

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial}{\partial x_1} f(\boldsymbol{x}) \\ \frac{\partial}{\partial x_2} f(\boldsymbol{x}) \\ \dots \\ \frac{\partial}{\partial x_n} f(\boldsymbol{x}) \end{pmatrix}. \tag{3}$$

As such, gradient is just a vector-valued function $\mathbb{R}^n \to \mathbb{R}^n$, just as the ordinary derivative is a function $\mathbb{R} \to \mathbb{R}$. However, we can calculate this function for any particular value of $\boldsymbol{x}$, and the result will be a vector (not function), exactly as the ordinary derivative, calculated at a particular x value is a number, not function. For instance, using the example function $g(\boldsymbol{x})$ above, let's calculate it at $\boldsymbol{x} = (1, 2)'$:

$$\nabla g(\boldsymbol{x})|_{\boldsymbol{x}=(1,2)'} = \begin{pmatrix} \log x_2 \\ x_1/x_2 \end{pmatrix}\Bigg|_{\boldsymbol{x}=(1,2)'} = \begin{pmatrix} \log 2 \\ 1/2 \end{pmatrix}. \tag{4}$$

This value is often written in shorter but imprecise way as

$$\nabla g \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} \log 2 \\ 1/2 \end{pmatrix}. \tag{5}$$

Note that this is not correct way of writing gradient as $g \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ is a constant ($\log 2$), and gradient of a constant is 0. However, it is a widely used shortcut in

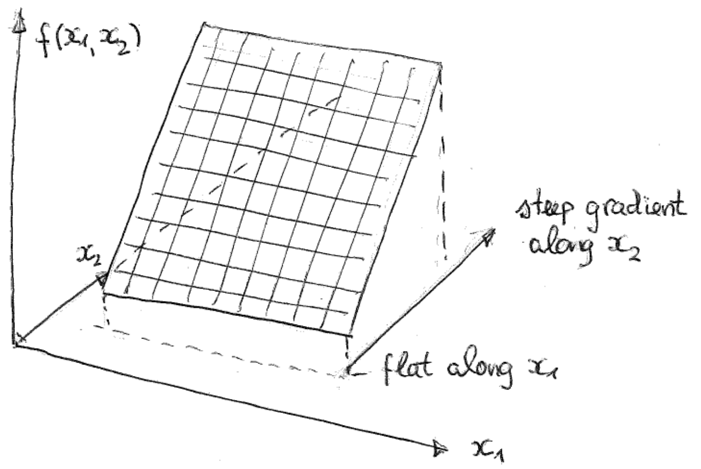

Figure 1: Function increasing along x_2 while constant along x_1.

the literature. It is important to understand when we talk about gradient of a function with respect of it's argument, calculated at a fixed argument value, and when we talk about a function, computed at the same fixed argument values. In the shortcut we implicitly understand that $\begin{pmatrix}1\\2\end{pmatrix}$ is the argument, gradient is computed with respect of the argument, and afterwards evaluated at this value of the argument.

1.2 Properties

However, it appears gradient is much more than a handy way to write a number of derivatives in a compact way. This shortcut naturally generalizes a number of properties of 1-dimensional derivatives.

1.2.1 Gradient and Direction of Steepest Ascent

As components of the gradient are just ordinary partial derivatives, each component indicates the slope of the function along that axis: how much will the function value grow if we move along the axis, while keeping the location on the other axis constant. If one component is large and another is small, we have a steep hill in the first direction while it is pretty flat along the other axis.

Let's look at linear functions–simple even surfaces with no curvature whatsoever. The situation looks something like on Figure 1. If the surface is rising rapidly along x_2 while staying constant along x_1, the direction of fastest climb is just along x_2. Analogously, if the function grows along x_1 while staying flat along x_2, we have to move toward x_1. Such a situation is depicted on Figure 2 although here the first gradient component $\partial f(\boldsymbol{x})/\partial x_1 < 0$ and hence we have to move to smaller values of x_1 instead. Obviously, if none of the gradient components are zero, we have to move somewhere in-between of these two directions. This is shown on Figure 3. It is also intuitive, that the "somewhere in-between" should be closer to the steeper gradient than to the smaller gradient component.

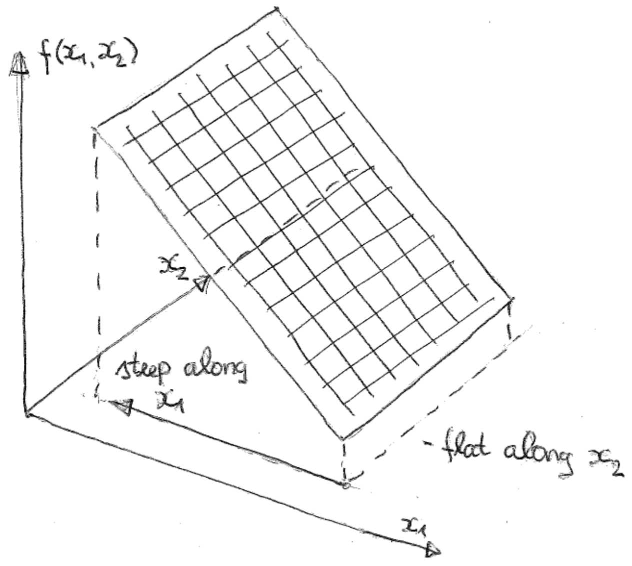

Figure 2: Function increasing along $-x_1$ while constant along x_2.

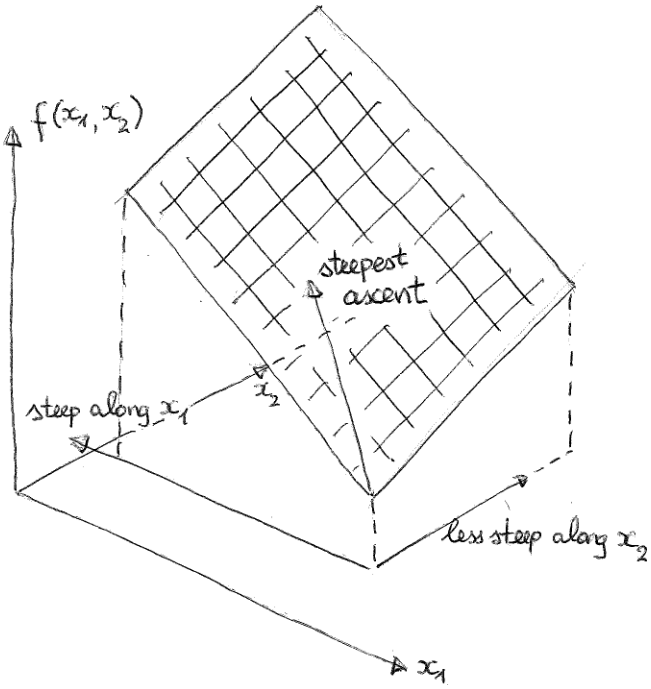

Figure 3: Function increasing both $-x_1$ and x_2. The direction of steepest climb is somewhere inbetween of these two gradient components.

It is easy to show that the exact direction of steepest climb is the same as the direction of the gradient vector. Let's choose a point (x_1, x_2) where the function's value is $f(x_1, x_2)$. Now move away from this point by $(\Delta x_1, \Delta x_2)$.

This causes the function to grow by

$$\Delta f \equiv f(x_1 + \Delta x_1, x_2 + \Delta x_2) - f(x_1, x_2) = g_1 \cdot \Delta x_1 + g_2 \cdot \Delta x_2 \qquad (6)$$

where g_1 and g_2 are the corresponding gradient components, calculated at (x_1, x_2). However, for not to go too much wild, we'll change the coordinates in this way that the total move will be of length one. Hence $\Delta x_1^2 + \Delta x_2^2 = 1$, or alternatively $\Delta x_2 = \sqrt{1 - \Delta x_1^2}$, and the function change can be written as

$$\Delta f = g_1 \cdot \Delta x_1 + g_2 \cdot \Delta x_2 \quad \text{or} \quad \Delta f = g_1 \cdot \Delta x_1 + g_2 \cdot \sqrt{1 - \Delta x_1^2}. \qquad (7)$$

Which Δx_1 results in the largest value of Δf? The optimality condition gives us

$$\frac{\partial \Delta f}{\partial \Delta x_1} = g_1 + g_2 \frac{-2\Delta x_1}{2\sqrt{1 - \Delta x_1^2}} = g_1 - g_2 \frac{\Delta x_1}{\Delta x_2} = 0 \qquad (8)$$

where we used the fact that $\sqrt{1 - \Delta x_1^2} = \Delta x_2$. The solution is

$$\frac{\Delta x_1}{\Delta x_2} = \frac{g_1}{g_2}. \qquad (9)$$

In other words, this means that the direction vector $(\Delta_{x1}, \Delta x_2)$ must be parallel to the gradient vector (g_1, g_2).

1.3 Gradient Ascent

Gradient Ascent (GA) and it's mirror image Gradient Descent, is a popular method to find maxima (and minima) of functions, widely used in various machine learning applications. It also serves as a basis for many more complex methods, such as Newton-Raphson. While one can easily find the optimum of simple functions, such as quadratic function, the optima of more complex ones cannot be found analytically. This includes most of the objective functions we encounter in machine learning practice, with linear regression being the only exception. So we have to rely on numerical computations, usually referred as *non-linear optimization*, to find the solution. GA is one of the most popular such methods.

The idea with GA is the following (see Figure 4).

1. Consider a 2D case. Start with an initial guess $\boldsymbol{x}^0$ of the location of the maximum. Note: I denote by the superscript 0 the initial vector, while it's components are denoted by subscripts: $\boldsymbol{x}^0 = (x_1^0, x_2^0)$.

2. Compute the gradient $\nabla f(\boldsymbol{x}^{\mathbf{0}})$.

3. Now take a step at the direction of the gradient. This leads to a new location $\boldsymbol{x}^1 = \boldsymbol{x}^0 + R \cdot \nabla f(\boldsymbol{x}^{\mathbf{0}})$. Scalar R, *learning rate*, determines the length of the step. As the gradient is pointing uphill, the function value at $\boldsymbol{x}^1$ is larger than at $\boldsymbol{x}^0$.

4. Now repeat the process choosing $\boldsymbol{x}^1$ as the starting point. Repeat until gradient is close to zero and the function value does not improve any more.

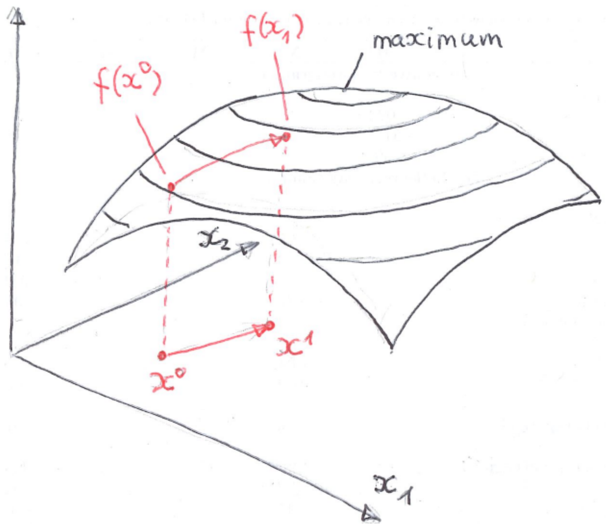

Figure 4: One step of Gradient Ascent. We start from an initial guess $\boldsymbol{x}^0$ and take a step along the gradient. This takes us uphill to $\boldsymbol{x}^1$. The function $f(\boldsymbol{x})$ is depicted by (rather circular) *level sets*.

We can further illustrate it with the function $g(\boldsymbol{x}) = x_1 \cdot \log x_2$ we used above. Remember it's gradient is $g(\boldsymbol{x}) = (\log x_2, x_1/x_2)'$.

1. Pick a starting point. Let's choose $\boldsymbol{x}^0 = (1,1)'$. The function value at that point is $f(\boldsymbol{x}^0) = 0$.

2. The gradient at this point is obviously $g(\boldsymbol{x})|_{\boldsymbol{x}=(1,1)'} = (\log 1, 1/1)' = (0,1)'$.

3. Now take a step from $\boldsymbol{x}^0$ along the gradient. But first we have to choose the learning rate R. Let's pick $R = 0.5$. Now we have

$$\boldsymbol{x}^1 = \boldsymbol{x}^0 + R \cdot \nabla f(\boldsymbol{x}^1) = (1,1)' + 0.5 \cdot (0,1)' = (1, 1.5)' \qquad (10)$$

   The new function value is $f((1,1.5)') = 1 \cdot \log 1.5 = 0.405$. Indeed, we moved uphill.

4. Now we ought to repeat the process further until we have reached the maximum – although note that this particular function does not posess a maximum.

This particular example is illustrated on Figure 5 by contour plot. Note that at $\boldsymbol{x}^0$, the gradient points straight up as the function is constant along x_1 at that point, and hence we move along the direction of steepest ascent at that point. However, as R is relatively large, the steepest ascent direction at $\boldsymbol{x}^1$ is slightly different.

Now we describe the algorithm in more detail for n dimensional case.

1. Pick an initial value of the parameter $\boldsymbol{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)'$. It is always good idea to choose parameters as close to the actual maximum as you can, but often you have little guidance about how to choose good starting values.

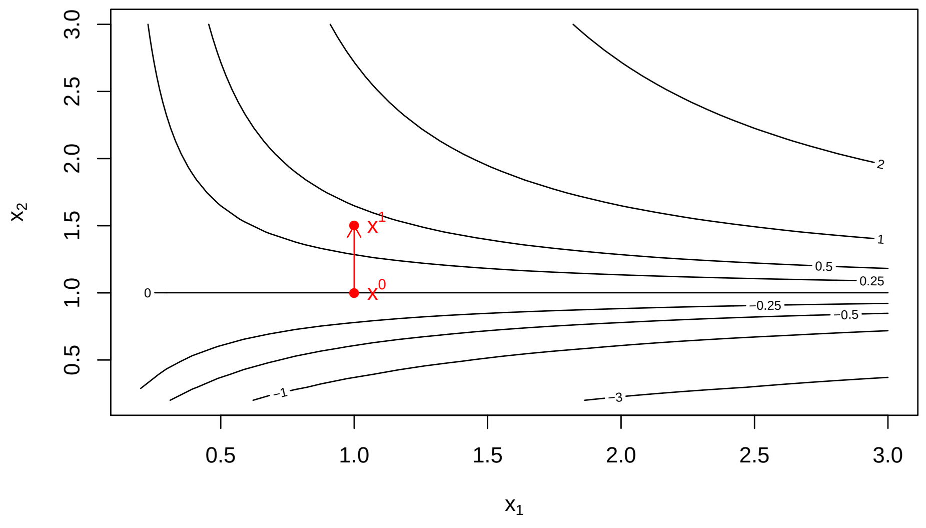

Figure 5: One Gradient Ascent step for function $g(\boldsymbol{x}) = x_1 \cdot \log x_2$

2. Compute the gradient of your objective function $\nabla f(\boldsymbol{x})$ at $\boldsymbol{x}^0$.

3. Take a step from $\boldsymbol{x}^0$ in the gradient direction. The step should be neither too long (you may overshoot) nor too short (it takes too long time to find the maximum). So we employ the learning rate R and take the step of length $R \cdot \nabla f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^0}$. This will land you into a new place we call $\boldsymbol{x}^1$:

$$\boldsymbol{x}^1 = x^0 + R \cdot \nabla f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^0} \tag{11}$$

   This is our new best bet for the location of maximum. It is probably not too good a place, but unless your code is wrong (or the function constant at $\boldsymbol{x}^0$, it is a better place than $\boldsymbol{x}^0$.

   Choice of a good R value is important. This is a parameter of your model, although not one that is directly related to the process you are modeling. Such parameters are typically called *hyperparameters*. Unlike many other hyperparameters, such as choice of k for k-nearest neigbhors, this one will quite likely not affect your final results, just the speed of convergence (or it may determine if your model will converge in the first place).

4. Are we in the correct place? There are several ways to check this:

   (a) Gradient is very small. Say, $|\nabla f(\boldsymbol{x}^1)| < \epsilon^g$, where $|\cdot|$ mean length of the vector, and ϵ^g is a small number, say 10^{-6}. Small gradient indicates that the function is flat at that point, and differentiable functions are flat at maximum.

   (b) Function value does not grow much any more. Say, $|f(\boldsymbol{x}^1) - f(\boldsymbol{x}^0)| < \epsilon^f$ where ϵ^f is another small number.

   (c) One may suggest more conditions, for instance about relative size of gradient.

    Typically we only need one of the criteria above: if gradient is close to zero, the function stops growing, and vice versa.

    These conditions are called *stopping criteria*. In practice, you always need an additional, bail-out criterion: stop if the process has been repeated too many times already. This is because we too often choose too small learning rate, run into numerical problems, or have coding errors.

5. If we are in correct place, stop here. If not, set $\boldsymbol{x}^0 \leftarrow \boldsymbol{x}^1$ and repeat from step 2.

Finally, a note about Gradient Ascent and Gradient Descent. Depending on the task, you may need to go downhill instead of uphill, for instance when finding the place of minimal loss. There are two trivial ways to turn GA into GD:

1. flip the sign of your objective function: instead of minimizing $f(\boldsymbol{x})$, maximize $-f(\boldsymbol{x})$.

2. change the algorithm in a way to take a step into the opposite direction of the gradient. If you don't want to change your code, just use negative learning rate R.

1.4 Resources

- Khan Academy's "Why gradient is the direction of steepest ascent" https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/gradient-and-directional-derivatives/v/why-the-gradient-is-the-direction-of-steepest