Gradient descent principles

Lluís Garrido – lluis.garrido@ub.edu

September 2016

Abstract

This laboratory is focused on unconstrained optimization of a function $f(x)$ and, in particular, on understanding the gradient descent principles. Why does this mehod work? Is it a good method?

1 A simple quadratic function

We begin by focusing on a simple two-dimensional function. Concretely,

$$f(x) = x_1^2 + x_2^2$$

where , $x \in R^2$, $x = (x_1, x_2)^T$ (vectors are expressed column-wise). This function is easy to minimize from a numerical point of view since it is a convex function (later on Gerard will define you exactly what a convex function is). Convex functions are characterized because they have one unique minimum.

How can a function $f(x)$, not necessarily convex, be minimized? Let us begin with an experiment using the previous simple convex function. Follow the next steps:

1. Take the example source code you have included within this document. Take a look at the code and observe at the plot. The code performs a contour plot of $f(x) = x_1^2 + x_2^2$ and draws the gradient of the function. Observe the direction of the arrows associated to the gradient. The gradient of a function, ∇f, points to the direction at which the function value increases and has the highest slope. That is, we have to follow the gradient direction in order to increase the function value at the highest rate.

   In order to find the minimum of this function we may follow the gradient in the opposite direction. The resulting algorithm is called therefore gradient descent. Given an initial guess x^0, we may try to find the minimum by performing a gradient descent

   $$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

   where k is the iteration. The latter equation defines an iterative algorithm that successively approaches the minimum. The value x^{k+1} is computed from x^k and we would like x^{k+1} to be "nearer" to the minimum than x^k. For that issue the step, $\alpha^k \in R$, is an important value to compute. For the moment we will just assume that the step is constant, $\alpha^k = 0.1$. This is a value that seems to work well for the function to minimize.

The gradient descent is a very simple algorithm and it works for simple functions like the one studied here. We would like to warn that there are many improvements to the latter algorithm that will be seen during the next weeks. In particular, taking the opposite of the gradient is not the best descent direction one may take. There are other directions, such as the one given by the Newton direction, that are much better if the function to minimize has a "weird" shape. In addition, the step α^k also plays an important role since we would like to approach the minimum in a fast way. There are many research works that concentrate on computing in a fast way a good value for α^k. For instance, the Armijo rule will be seen in the lectures.

2. Assume that we follow the simple gradient descent

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

You are requested to implement the previous algorithm with a constant value of $\alpha^k = 0.1$, for instance. Take an initial point x^0 and perform at most 100 iterations (or whatever you prefer). You are proposed to try different starting points x^0. Observe that the algorithm will always converge to the unique minimum the function has. You are proposed to draw the path the gradient descent follows for each of the starting points x^0 you have studied.

You may try other values of α such as $\alpha^k = 1$. In this case you may see that the gradient descent performs "too big steps" and does not perform well at all. This shows you the importance of selecting a good value for α, the step.

2 The exercise of lab 1

Let us continue with the function you studied in lab 1

$$f(x_1, x_2) = x_1^2\,(4 - 2.1\,x_1^2 + \frac{1}{3}x_1^4) + x_1 x_2 + x_2^2\,(-4 + 4x_2^2)$$

You are requested to perform the next experiments

1. Perform a contour plot of the previous function and draw the gradient of the function. Can you see where the minimum are? Recall that you studied this function in the previous lab and that you focused on finding, using brute force, which are the minimum of the latter function. How many minimum did you find? How many minima can you see on the plot? Is everthing in accordance?

This function is more complicated than the previous one: it is not a convex function since it has several local minima. Which minima will be found? Can you intuit it? It will depend, in fact, on the initial value x^0 you may take.

2. Assume that we follow the simple gradient descent with 100 iterations

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

Using the plot found at step 1, try to start at different starting points x^0 using $\alpha^k = 0.1$. Draw the path the minimization algorihm follows and observe to which minimum the algorithm converges. You should see that, for a given starting point x^0, the algorithm should converge to the minimum located in the valley to which x^0 belongs.

3. Let us perform an improvement to the previous algorithm. Indeed, until now we have considered a constant value for α^k. Let us now consider adapting the value of α^k at each iteration k

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

There are many algorithms that try to compute a good value of α^k. We will consider now the backtracking algorithm. It works quite well in many cases. However, in general you are recommended to use other algorithms that implement, for instance, the Armijo rule (the definition of the rule will be given in the lectures). The advantage of the latter algorithms is that they find much better values for α^k than the simple backtracking algorithm.

This is the backtracking algorithm. Assume that x^k and $\nabla f(x^k)$ have been computed. In order to compute x^{k+1} we start with $\alpha^k = 1$ and perform the next iterations

(a) Check if $f(x^k - \alpha^k \nabla f(x^k)) < f(x^k)$. This condition checks if the proposed update reduces the value of $f(x^k)$.

(b) If (a) is satisfied, perform the update $x^{k+1} = x^k - \alpha^k \nabla f(x^k)$. Compute $\nabla f(x^{k+1})$ start again this algorithm (with $\alpha^{k+1} = 1$).

(c) If (a) is not satisfied, update $\alpha^k = \alpha^k/2$, for instance. That is, the step is divided by 2. Go to step (a) and check again.

(d) If α^k reduces below a certain threshold, you may assume that the gradient descent has converged. You have found the minimum!

The previous algorithm shows the principles of the backtracking algorithm: the idea is to compute α^k by progressively reducing its value until you ensure that the value of f is reduced. You may find several improvements to the algorithm, but this algorithm just gives you the basic ideas behind it.

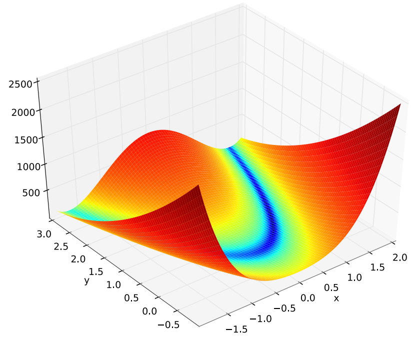

Figure 1: The Rosenbrock function of two variables, $a = 1$ and $b = 100$. Plot obtained from wikipedia, see `https://en.wikipedia.org/wiki/Rosenbrock_function`.

> Implement the previous algorithm. Rather than using 100 iterations, perform the necessary number of iterations k until you find an iteration k at which α^k goes below e.g. 10^{-5}. How does the algorithm perform? For instance, compare the results with the results you have obtained in step 2. How many iterations are needed to find the minimum? 100? Or may be more? There is no “correct” answer to this question. Just experiment a little bit!

3 The problems of the gradient descent

We have seen the basics of the gradient descent

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

Intuitively we follow the direction of the highest descent. Is this the best direction we may take? The answer is clearly NO. The gradient descent works if the variables of the function are well scaled. But if not, the gradient descent may work very badly. Let us see it with an example.

Let us consider the Rosenbrock function, see figure 1. The function is defined as

$$f(x_1, x_2) = (a - x_1)^2 + b\,(x_2 - x_1^2)^2$$

The function has a global minimum at $(x_1, x_2) = (a, a^2)$, where $f(x_1, x_2) = 0$. The global minimum is inside a long, narrow, parabolic shaped valley. The convergence to the global minimum is difficult.

You are asked to perform the next experiments:

1. Plot the contours of the Rosenbrock function for $a = 1$ and $b = 100$. You may also draw the gradient information.

2. Try different starting points x^0 in order to check the robustness of the backtracking descent algorithm you have implemented. Draw the path the gradient descent follows in order see how good your algorithm performs.