Gradient descent principles

Lluís Garrido – lluis.garrido@ub.edu

September 2018

Abstract

This laboratory is focused on unconstrained optimization of a function $f(x)$ and, in particular, on gradient descent and Newton principles.

Contents

1 Gradient descent methods

This part is focused on understanding the gradient descent principles. We expect to answer questions such as: why does this method work? Is it a good method?

1.1 A simple quadratic function

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

$$f(\mathbf{x}) = x_1^2 + x_2^2$$

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

How can a function $f(\mathbf{x})$ be minimized using numerical techniques? Let us begin with the previous function. 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 $f(\mathbf{x}) = x_1^2 + x_2^2$ 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 $\mathbf{x}^0$, we may try to find the minimum by performing a gradient descent

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

where k is the iteration. The latter equation defines an iterative algorithm that successively approaches the minimum. The value $\mathbf{x}^{k+1}$ is computed from $\mathbf{x}^k$ and we would like $\mathbf{x}^{k+1}$ to be "nearer" to the minimum than $\mathbf{x}^k$. The value α^k is the step. For the moment we will just assume that the step is constant.

The gradient descent with a constant step 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 in this lab.

1. For instance, the step α^k 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 a good value for α^k at each iteration. We will see the basic principles of these algorithms in this lab.

2. 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 may work much better. We will see the latter issue in section 2.

Let us now assume that we follow the gradient descent

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

Perform the next experiments

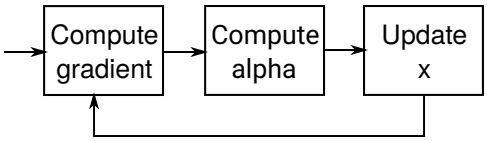

Figure 1: The double loop procedure to perform the gradient descent. The outer loop updates the values of $\mathbf{x}^k$, whereas the inner loop computes the values of α^k. For simplicity, only the outer loop is shown in this figure.

1. You are requested to implement the previous algorithm with a constant value of $\alpha^k = 0.1$, for instance (this is a value that seems to work well for the function to minimize). Take an initial point $\mathbf{x}^0$ and perform at most 100 iterations (or whatever you prefer). You are proposed to try different starting points $\mathbf{x}^0$. 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 $\mathbf{x}^0$ you have studied.

2. You may try other values of α such as $\alpha^k = 1$ and $\alpha^k = 2$. 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. There are many research works that have focused on this issue, but it is not the purpose of this lab to see how they work. Rather, we just will see some of their principles.

1.2 The exercise of lab 1

Let us continue with the function you studied in lab 1

$$f(x_1, x_2) = x_1^2\,(4 - 2.1\,x_1^2 + \frac{1}{3}x_1^4) + x_1x_2 + x_2^2\,(-4 + 4x_2^2)$$

Perform a contour plot of the previous function. 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.

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 $\mathbf{x}^0$ you may take. Perform the next experiments

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

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

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

2. 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.

There are many algorithms that try to compute a good value of α^k. In general you are recommended to use 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 you are going to implement next.

The backtracking algorithm works quite well in many cases. Assume that $\mathbf{x}^k$ and $\nabla f(\mathbf{x}^k)$ have been computed. In order to compute $\mathbf{x}^{k+1}$ we start (at each iteration k) with $\alpha^k = 1$ and perform the next inner loop iterations, see Figure 1.

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

(b) If (a) is satisfied, perform the update $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \nabla f(\mathbf{x}^k)$. Compute $\nabla f(\mathbf{x}^{k+1})$ 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 or if $\left|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right|$ or $\left\|\nabla f(\mathbf{x}^{k+1})\right\|$ is below a certain threshold, you may conclude 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 is not the purpose of the lab.

Implement the previous algorithm. Rather than using 100 iterations, perform the necessary number of iterations k until you find an iteration k at which the stopping criterion is satisfied, e.g. $\left|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right| < 10^{-3}$ (you may also use a threshold on $\left\|\nabla f(\mathbf{x}^{k+1})\right\|$). How does the algorithm perform? Compare the results with the results you have obtained with a constant step. How many iterations are needed to find the minimum? 100? Or may be less? There is no "correct" answer to this question. Just experiment a little bit!

1.3 The Rosenbrock function

We have seen the basics of the gradient descent

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

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 well 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 2. The function is defined as

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

The function is convex and 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. There are no local minimums.

You are asked to perform the next experiments:

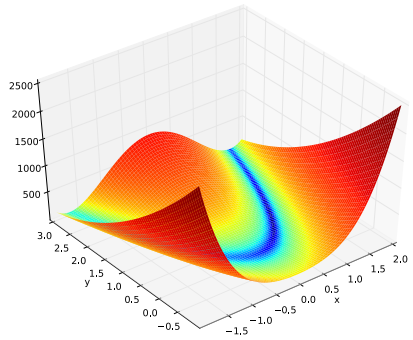

Figure 2: The Rosenbrock function of two variables, $a = 1$ and $b = 100$. Plot obtained from wikipedia, see https://en.wikipedia.org/wiki/Rosenbrock_function.

1. Plot the contours of the Rosenbrock function for $a = 1$ and $b = 100$. The minimum of the function is thus at $(x_1^*, x_2^*) = (1, 1)$. You may also draw the gradient information.

2. Try different starting points $\mathbf{x}^0$ 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. You may stop the algorithm as soon as the stopping criterion is satisfied, e.g. $\left|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right| < 10^{-3}$ (or use the threshold on $\left\|\nabla f(\mathbf{x}^{k+1})\right\|$). Do not limit the iterations to 100. How many iterations are performed until the stopping criterion is satisfied? The fact that the stopping criterion is satisfied does not necessary imply that the minimum has been reached. Once the stopping criterion is satisfied, just check the values of x_1 and x_2 the algorithm has been able to find. Is the algorithm able to reach the minimum?

2 Newton descent method

We have seen that the computation of the step α^k is critical for a good performance of the gradient descent. The question that arises now is: can we use other search directions that may improve the performance of the descent? The answer is yes! What other search directions may we take? One well known search direction is the Newton direction.

2.1 A simple quadratic function

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

$$f(\mathbf{x}) = 100\, x_1^2 + x_2^2$$

where , $\mathbf{x} \in R^2$, $\mathbf{x} = (x_1, x_2)^T$ (vectors are expressed column-wise). Observe that the function is convex and, thus, it has one unique stationary point which corresponds to the minimum. You are proposed to follow the next steps:

1. We are first going to minimize the previous function using the gradient descent algorithm. That is,

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

Given an initial guess x^0, count the number of steps that are needed in order to find the minimum so that $\left|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right| < 10^{-3}$ (or on $\left\|\nabla f(\mathbf{x}^{k+1})\right\|$). The value of α^k can be computed using the backtracking algorithm you implemented in section 1.2.

2. In a general case, an unconstrained minimization algorithm uses the next algorithm

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$$

where $\mathbf{d}^k$ is a descent direction (also called search direction). A well known search direction is the Newton direction. The descent direction $\mathbf{d}^k$ obtained from the second order Taylor expansion of $f(\mathbf{x})$,which is

$$f(\mathbf{x}^k + \mathbf{d}) = f(\mathbf{x}^k) + \mathbf{d}^T \nabla f(\mathbf{x}^k) + \frac{1}{2}\mathbf{p}^T \nabla f(\mathbf{x}^k)\mathbf{p}$$

In other words, we perform a quadratic approximation of $f(\mathbf{x}^k)$. The minimum of the quadratic expression is obtained for

$$\nabla^2 f(\mathbf{x}^k)\, \mathbf{d}^k = -\nabla f(\mathbf{x}^k)$$

The solution to this linear system of equations, $\mathbf{d}^k$,is the so called Newton direction. You are now requested to minimize the previous function using the Newton method. As has been done with the gradient descent, count the number of steps that are needed in order to find the minimum so that $\left|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right| < 10^{-3}$. In order to be fair, be sure to use the same backtracking algorithm and the same initial points as has been done for the gradient descent.

3. Compare the number of iterations that are needed to get to the minimum. It may be interesting to plot the path that each of the method follows.

2.2 The exercise of lab 1

We are now going to focus on the function you studied in lab 1

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

Observe that this function is not convex. You are requested to perform the next experiments

1. Recover the experiments you performed in the previous sections. Indeed, take an initial point $\mathbf{x}^0$, quite far away from a minimum, and compute the number of iterations that are needed to get to the minimum.

2. We are now going to focus on the Newton method. As stated before, the Newton method uses a descent direction which is the solution of

$$\nabla^2 f(x^k)\, \mathbf{d}^k = -\nabla f(\mathbf{x}^k) \tag{1}$$

The vector $\mathbf{d}^k$ is a descent direction only if the Hessian is positive definite. If the Hessian is not positive definite the obtained vector $\mathbf{d}^k$ is not necessarily a descent direction. Thus, another approach has to be devised if the Hessian is not positive definite. There are several methods that try to tackle the previous problem. For instance, a simple method is to use the

gradient descent in case the Hessian is not positive definite. Another method is to perform the descent only for those components $(x_1, x_2)^T$ for which the corresponding eigenvalue is positive (recall that the eigenvalue gives us information about the shape of the function: convex or concave).

In this lab we propose to use the gradient descent in case the Hessian at iteration $\mathbf{x}^k$ is not positive definite. Thus, the algorithm to implement can be summarized as follows:

(a) Assume we are at iteration k. Compute the Hessian matrix. (This may be done analytically).

(b) If the Hessian is positive definite, use the Newton method to perform the descent. Otherwise, use the gradient descent to perform the descent. In both cases use the backtracking algorithm to compute a good value for α^k.

It is interesting to analyze which of both methods, the combined gradient-Newton or the "pure" gradient descent, is used at each iteration k. A simple way to proceed is to plot the path the minimization algorithm follows: use red dots if the algorithm uses a gradient descent, and use green dots if the Newton method is used. You should observe that "near" the minimum the Newton method is mostly used. The function is convex near the minimum and that allows to approach it in a fast way.

3. Compare the Newton method (item 2 in this list) with the pure gradient descent (item 1 in this list). You may test how many iterations are required to arrive to the minimum and compare the path that each of the method follows to arrive to the minimum. .

Observe that in this experiment you perform a "binary" decision: if all the eigenvalues are positive the Newton direction is used, if not, the gradient descent is used. This is a simplification of the techniques that are currently available. Indeed, if only some of the eigenvalues are positive one may compute a descent direction that is not as good as the (ideal) Newton direction but that is better than the gradient descent. The algorithm is called "Truncated Newton Method". We do not implement it here due to lack of time.

Just some words to finish. You should observe that the Newton method requires, in general, less iterations than the gradient descent to arrive to the minimum. But the disadvantage of the Newton method is that it requires solving the linear system of equations (1) and thus it requires higher computational effort if you have to deal with large number of variables. In addition, one needs to know if the Hessian matrix is positive definite in order to be sure that the vector d^k is a descent direction. There are methods that tackle the previous computational issues in a fast way, namely the conjugate gradient descent algorithm. In any case, take into account that the Newton method has great advantages over the gradient descent method. But this does not necessarily mean that the Newton method is always better than the gradient descent to minimize your function. In some cases the gradient descent may be good enough for the problem to be solved. This is in fact what happens with machine learning: the problem to minimize may be so big that even the gradient descent is not able to efficiently deal with the minimization. In such cases a "simplification" of the gradient descent is used, the so called stochastic gradient descent. We'll see them in another lab.

2.3 The Rosenbrock function

Let us consider again the Rosenbrock function, see section 1.3. You should have seen that you require a lot of iterations of the gradient descent in order to arrive to the minimum. This is due to the fact that during iterations the gradient descent continuously jumps from one side to the other side of the valley without taking into account the shape of the valley. How does Newton perform here?

Assume you take $a = 1$ and $b = 100$ for the Rosenbrock function. You are asked to perform the next experiments:

1. Recover the experiments performed in section 1.3, indeed the initial starting points $\mathbf{x}^0$.

2. Try now to use the Newton algorithm and see how many iterations are required to arrive to the minimum. It is interesting to analyze which of both methods (Newton or gradient descent) is used at each iteration. Use two different colors to plot which of both methods is used at each iteration. With the Newton method you should be able to arrive to the minimum.

3 Report

You are asked to deliver an *individual* report (PDF, notebook, or whatever else you prefer). Comment each of the steps you have followed as well as the results and plots you obtain. Do not expect the reader (i.e. me) to interpret the results for you. I would like to see if you are able to understand the results you have obtained.

If you want to include some parts of code, please include it within the report. Do not include it as separate files. You may just deliver the Python notebook if you want.