Gradient descent principles

Ignasi Cos – ignasi.cos@ub.edu

October 2023

Abstract

This laboratory focuses on unconstrained optimization of a function $f(x)$ and, in particular, on the gradient descent and Newton algorithms.

Contents

1 Gradient descent methods

This part focuses on understanding the principles of a gradient descent algorithm. We expect to be able to answer questions such as: why does this method work? Why/in which cases is this a good method?

1.1 A simple quadratic function

We begin with a simple two-dimensional function, namely

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

where , $\mathbf{x} \in R^2$, $\mathbf{x} = (x_1, x_2)^T$ (vectors are expressed as columns). It is easy to minimize this function numerically because it is a convex function. Convex functions have a single minimum.

How to minimize a function $f(\mathbf{x})$ numerically? Let us begin with the previous function. Take the example source code you have included within this document and run it. 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 towards the direction in which the function increases with the highest possible slope. In other words, we follow the gradient direction in order to increase the value of the function at its highest rate.

By contrast, if we wish to find the minimum, it is convenient to follow the gradient in the opposite direction. The resulting algorithm is therefore called 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 simple algorithm and it works well for simple functions like the one previously introduced. A constant step also is sometimes used with complex functions in the absence of a better solution. In this practical we will see some procedures that can be used to minimize a function. Indeed,

1. The step α^k plays an important role (from a mathematical point of view) since we would like to approach the minimum fast. There are many research works that focus on computing a good value for α^k per iteration. We will see the basic principles of these algorithms in section 1.2.

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

We begin some experiments with gradient descent using a constant step

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

Perform the next experiments

1. Please, implement the previous algorithm with a constant $\alpha^k = 0.1$ parameter. Take an initial point $\mathbf{x}^0$ and perform at most 100 iterations. Try different starting points $\mathbf{x}^0$. Observe that the algorithm always converges (up to a certain precision) to the only minimum this function has. Observe that the more the iterations, the closer to the minimum. Please, 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 the gradient descent performs poorly due to the too bigger steps. This shows you the importance of selecting an appropriate value for the step α,. 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 A function with multiple minima

Consider now the following function:

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

Plot a contour plot of the previous function within the range $x_1 \in [-2, 2]$ and $x_2 \in [-1, 1]$. This function is fundamentally more complex than the previous because it is not a convex function, which 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 $\alpha^k = 0.1$ (within in range $x_1 \in [-2, 2]$ and $x_2 \in [-1, 1]$). 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 usually converges 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 implements, 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 value of α^k reduces the value of $f(\mathbf{x}^k)$.

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.

(b) 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.

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

There are several convergence criteria that may be used to check the algorithm has found the minimum. For instance, 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 (or if α^k reduces below a certain threshold), you may conclude that the gradient descent has converged. You have found the minimum (up to a certain precision)!

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, although this is not the purpose of the practical.

Implement the previous algorithm. Rather than using 100 iterations, perform the necessary number of iterations k until you find an optimal iteration k for which the convergence criterion is satisfied, e.g. $\left\| \nabla f(\mathbf{x}^{k+1}) \right\| < 10^{-5}$ (or $\left| f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \right| < 10^{-3}$). How does the algorithm perform as compared with the constant step results? How many iterations are needed to reach the minimum? 100? Or may be less? There is no single right answer to this question. Just experiment a little bit!

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`.

1.3 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 no. Gradient descent works well if the variables of the function are well scaled. Otherwise, it may perform poorly. See the example next.

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 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, very 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:

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. $\left\|\nabla f(\mathbf{x}^{k+1})\right\| < 10^{-5}$ (or $\left|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right| < 10^{-3}$). 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? Recall that we know the minimum which is $(x_1^*, x_2^*) = (1, 1)$.

   If the stopping criterion is not able to find the minimum, modify the criterion (by putting lower thresholds, for instance) to see if it is able to find the minimum. Just test if you are able to find 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 reach the minimum so that $\left\|\nabla f(\mathbf{x}^{k+1})\right\| < 10^{-5}$ (or $\left|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right| < 10^{-3}$). 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 $\mathbf{d}^k$ is obtained from the second order Taylor expansion of $f(\mathbf{x}^k)$,which is
$$f(\mathbf{x}^k + \mathbf{d}) = f(\mathbf{x}^k) + \mathbf{d}^T \nabla f(\mathbf{x}^k) + \frac{1}{2}\mathbf{d}^T \nabla^2 f(\mathbf{x}^k)\mathbf{d}$$
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 the convergence criterion is satisfied, $\left\|\nabla f(\mathbf{x}^{k+1})\right\| < 10^{-5}$ (or $\left|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right| < 10^{-3}$), as you have done before. 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 also be interesting to plot the path that each of the method follows.

2.2 A function with multiple minima

We are now going to focus on the function studied in Section 1.2

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

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 $\mathbf{x}^0$, "far away" from a minimum (but e.g. within the range $x_1 \in [-2, 2]$ and $x_2 \in [-1, 1]$), 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 (and 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 classical 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 classical 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 related to the positive eigenvalues. The descent that is not as good as the (ideal) Newton direction but it is better than the simple gradient descent. The algorithm is called "Truncated Newton Method". We do not implement it here due to lack of time.

Just a few words to summarize. You should observe that the Newton method requires, in general, a lesser amount of iterations than the gradient descent to reach 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 L-BFGS. 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 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 (which may use the Adam algorithm that try follow a good descent path). We'll see this algorithm in another practical.

2.3 The Rosenbrock function

Let us consider again the Rosenbrock function, see section 1.3. You should have seen that you required many iterations of the gradient descent to reach the minimum (if you have been able to arrive to it). 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, and the initial starting points $\mathbf{x}^0$.

2. Try now to use the combined Newton-gradient descent 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 requested to deliver a report (PDF, python notebook) in pairs. Comment each of the steps followed, as well as the results and plots obtained. 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 a commented version of the Python notebook.