Optimization for Big Data - Gradient Descent

Summary

The goal of this second session is to use gradient descent methods with Python on practical problems of optimization. You are invited to modify your parameters in the code and try to understand what happens with the output of the software.

You will be asked to produce a report (one for two students at most) that contains practical and theoretical insights on the optimization methods used.

1 One dimensional examples

We begin illustrating the gradient descent method with 1-dimensional examples.

1.1 First example

Consider $f_1 : x \longrightarrow x^2 - \frac{1}{4}x^4$ and the sequence of points

$$x_{k+1} = x_k - \alpha \nabla f(x_k).$$

1. What are the possible limits of $(x_k)_{k \geq 1}$?
2. Use several values of the step-size and of the initialization point to study the behaviour of the method. Explain.
3. In the situation where the sequence converges, what is the rate of convergence ? Illustrate this result with some graphical insights.

1.2 Second example

We now consider $f_2 : x \longrightarrow \frac{x^2}{1+x^2}$.

1. Is f_2 convex ? Strongly convex ? C^1_L ?
2. If we initialize x_0 in a good interval, is it possible to obtain a stable interval (that may depend on parameters of the step-size) where f_2 enjoys nice properties ?
3. In such a case, what is the convergence rate of the gradient descent scheme ? Illustrate with numerical experiments.

2 Two dimensional examples

2.1 Baseline code

```
def methodNormeGrad(f,X0,alpha,tol,N=200):
    X=X0
    g=gradientApprox(f,X0)
        [...]
        return X,lnormeg
```

Fill the [...] to produce a function that computes the approximate gradient of any function f. Use it for a generic gradient descent scheme :

```
def methodePasFixe(f,X0,alpha,tol,N=200):
    lX=[X0]
    X=X0
    g=gradientApprox(f,X0)
        [...]
    if n==N:
        print("Max number of iterations",N)
    return lX
```

Fill the [...]

2.2 First example

We consider $g_1 : (x, y) \longmapsto (x + y - 1)^2 + (x - 2y)^2 + (2x - 4y + 3)^2$.

1. Is g_1 convex, strongly convex, C^1_L ? Why ?
2. Explain a good set-up for the step-size choice of the gradient descent method.
3. Find the minimizer of g_1 theoretically.

4. Illustrate the behaviour of the gradient descent method.

2.3 Second example

We consider $g_2 : (x, y) \longmapsto 1 - \frac{1}{1+ax^2+y^2}$ where $a > 0$ is a parameter that should be discussed later on.

1. Is g_2 C^1_L ? For which value of L ?
2. How should be chosen the step-size α ? Explain why $\alpha = \frac{1}{1+a}$ leads to optimal results ?
3. Illustrate the possible convergence rates numerically.
4. Instead of computing and using the exact gradient of g_2, use the difference values method to approximate the partial derivatives of g_2 and then plug these estimates in your gradient descent algorithm.
5. With the following code, you can draw the level sets of g_2

```
aX0=linspace(-3,3)
aX1=linspace(-3,3)
Z=array([[fa(array([x0,x1]))
                for x0 in aX0] for x1 in aX1])
contour(aX0,aX1,Z,12)
axis('scaled')
show()
```

6. Have a look at the trajectories of gradient descent :

```
a=2
l=methodePasFixe(fa,[1,1],.2,1e-5)
lx0=[X[0] for X in l]
lx1=[X[1] for X in l]
aX0=linspace(-2,2)
aX1=linspace(-2,2)
Z=array([[fa(array([x0,x1])) for x0 in aX0] for x1 in aX1])
contour(aX0,aX1,Z,12)
plot(lx0,lx1,"-ro")
axis('scaled')
show()
```

3 Looking for a perfect billard trajectory

We consider a parametrization $\gamma : t \longrightarrow \mathbb{R}^2$ of a convex two-dimensional area. For the sake of simplicity, we will consider below

$$\gamma(t) = (a\cos t, b\sin t),$$

where a and b are positive.

1. For a given curve γ, we consider a sequence of n times $(t_0, \dots, t_{n-1})$ and the function

$$L(t_0, \dots, t_{n-1}) := \|\gamma(t_0) - \gamma(t_{n-1})\| + \sum_{i=1}^{n-1} \|\gamma(t_i) - \gamma(t_{i-1}))\|$$

   Explain the geometrical meaning of L.
2. Explain why L possesses a global maximum.

   We admit the mathematical fact that a maximizer of L produces a perfect billard trajectory.
3. We introduce the following function

```
epsilon=1e-8
def gradientApprox(f,x):
    fx=f(x)
    gra=zeros(size(x))
    for i in range(size(x)):
        veps=zeros(size(x))
        veps[i]+=epsilon
        gra[i]=(f(x+veps)-fx)/epsilon
    return gra
```

   Explain the previous function.
4. Define a function that parametrizes γ.

```
def gamma(t):
    return array([A*cos(t),B*sin(t)])
```

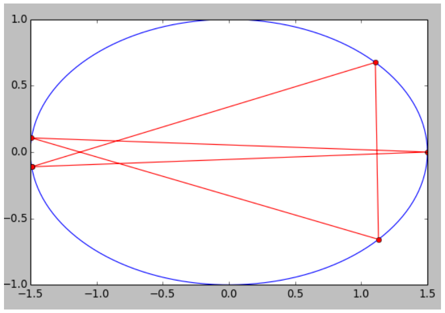
   For a sequence of n points stored in a vector vt, define a function that computes the opposite of L, defined as $moinsL$.

5. Use this function moinsL to optimize L with a gradient descent iteration scheme ran until the norm of the gradient becomes below a tolerance value tol.

```
def methodNormeGrad(f,X0,alpha,tol,N=200):
    X=X0
    g=gradientApprox(f,X0)
        [...]
        return X,lnormeg
```

6. Find some billard trajectories for several values of n !

```
vt,listeNormes=methodNormeGrad(moinsL,
                randn(5),.2,1e-6)
```



7. What is the rate of convergence of the method ? Illustrate it numerically.

4 Linear model - DIY !

1. Give a brief remainder on the statistical model of the linear model.
2. What is the function to be minimized ? We expect to use a gradient descent method (instead of a direct inversion).
3. Sample synthetic entries to produce a design matrix X and a set of observations Y.
4. Solve the linear model with the help of the gradient descent approach. Compare with the explicit formula.
5. Find a real database and solve the linear model issue with the gradient descent approach (instead of using the standard explicit formula).