1 24th of October 2018 — A. Frangioni

1.1 Gradient method for non-quadratic functions

We want to move from quadratic functions to wider families of functions.

Do you recall?

The step size α in the quadratic case is defined as follows: $\alpha = \frac{\|d\|^2}{d^T Q d}$.

We would like to find a more general form for the step size, which doesn't depend on the fact that the function is quadratic.

The algorithm for finding local minima of non-quadratic functions has the same structure of the one used for quadratic ones, i.e. first compute the direction of the step and then compute its size.

We will see that, differently from the quadratic case (where the gradient was $\nabla f(x) = Qx + q$) computing the gradient in this more general case isn't easy at all.

We may recall from last lecture that the proof of the orthogonality of the gradient doesn't depend on the quadratic nature of our functions, so it works in this case too.

1.1.1 How fast does it converge?

Given Algorithm 1.1 for finding minima of quadratic functions (that differs from the one provided for quadratic ones for the step size) we would like to understand how fast the convergence is.

Algorithm 1.1 Pseudocode for non-quadratic functions local minimum detection.

```
1: procedure SDQ($(f, x, \varepsilon)$)
2:     while ($\|\nabla f(x)\| > \varepsilon$) do
3:         $d \leftarrow -\nabla f(x)$;
4:         $\alpha \leftarrow \text{argmin}\{f(x + \alpha d)\}$;
5:         $x \leftarrow x + \alpha d$;
6:     end while
7: end procedure
```

Theorem 1.1. *Let f be a function in C^2 and let x_* be a local minimum for f s.t. $\nabla^2 f(x_*) \succ 0$ (which means that the Hessian matrix is strictly positive definite):*

$$\{ x^i \} \to x_* \Longrightarrow \{ f(x^i) \} \to f(x_*)$$

linearly, with same R as the quadratic case, depending on λ_1 and λ_n of $\nabla^2 f(x_*)$.

This theorem means that if the function is differentiable and the Hessian is strictly positive definite then when getting closer and closer to the minimum, the function is more and more similar to a quadratic function.

This similarity is a good new, since we can use the same methods of the quadratic case, but, as we recall from the previous lecture, we must pay attention to **conditioning**.

At this point we need to work on finding the local minimum of the one dimensional function φ^i, s.t.:

$$\varphi^i(\alpha) = f(x^i + \alpha d^i)$$

where $d^i = -\nabla f(x^i)$.

Let's omit the i, since we are concentrating on a single iteration.

We are interested in finding a α^* such that $\varphi'(\alpha^*) = 0$.

Example 1.1. *Let's suppose we are in $\mathbb{R}^2$ and $f(x,y) = x^2e^y$. We can differentiate F and $\nabla f(x,y) = (2xe^y, x^2e^y)$.*

At the i-th iteration $(x,y) = (1,0)$, so $\nabla f(1,0) = (2,1)$. Now $x(\alpha) = (1,0) - \alpha(2,1) = (1-2\alpha, 0-\alpha)$.

At this point we obtain $\varphi(\alpha) = f(x(\alpha)) = (1-2\alpha)^2e^{-\alpha}$.

Chi è $x(\alpha)$?

It's hard to find the roots of this function. The points we can find are not $\varphi'(\alpha) = 0$, but instead $|\varphi'(\alpha) \leq \varepsilon'|$, where the meaning of ε is bounding the directional derivative to be small.

Fact 1.2. *Let $\varphi : \mathbb{R} \to \mathbb{R}$, such that $\varphi(\alpha) = f(x^i + \alpha d^i)$, $\varphi'(\alpha) =< \nabla f(x^i + \alpha d^i), d >$.*

Proof.
TODO: using the **chain rule**: $f : \mathbb{R}^m \to \mathbb{R}^k$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^k$ such that $h(x) = f(g(x)) \Rightarrow$ $\mathbf{Jh(x)} = \mathbf{Jf(g(x))} \cdot \mathbf{Jg(x)}$

Obs: $Jf \in \mathbb{R}^{k\times m}$, $Jg \in \mathbb{R}^{m\times n}$ and $Jh \in \mathbb{R}^{k\times m} \cdot \mathbb{R}^{m\times n} = \mathbb{R}^{k\times n}$. □

Fact 1.3. *We claim that $\varepsilon' = \varepsilon$.*

Proof.
We want to find the relationship between the two parameters ε and ε'. Assuming that we've got a black box that finds α, given ε', we are interested in computing ε'.

Key idea: Normalization of the direction.

We may normalize the direction of movement d^i without perturbing the behaviour of the algorithm: $d^i = -\frac{\nabla f(x^i)}{\|\nabla f(x^i)\|}$. Note that we're not worried of dividing by the norm of the gradient, since if it gets 0 we have already stopped the procedure.

In this new context $\|d^i\| = 1$ and

$$\begin{aligned}
\varphi'(0) &= \frac{\partial f}{\partial d}(x) \\
\text{(From Proposition 1.2)} &=< \nabla f(x), d > \\
&=< \nabla f(x), \frac{-\nabla f(x)}{\|\nabla f(x)\|} > \\
&= -\frac{< \nabla f(x), \nabla f(x) >}{\|\nabla f(x)\|} \\
&= -\frac{\|\nabla f(x)\|^2}{\|\nabla f(x)\|} \\
&= -\left\|\nabla f(x^i)\right\|
\end{aligned} \tag{1.1}$$

$|\varphi'(\alpha^i)| = |< \nabla f(x^{i+1}), d^i >| = \left|< \nabla f(x^{i+1}), -\frac{\nabla f(x^i)}{\|\nabla f(x^i)\|} >\right|$

Hence, if $\{x^i\} \to x$:

$$\lim_{i\to\infty}\left|< \frac{\nabla f(x^i)}{\|\nabla f(x^i)\|}, \nabla f(x^{i+1}) >\right| =< \frac{\nabla f(x)}{\|\nabla f(x)\|}, \nabla f(x) >= \|\nabla f(x)\| \leq \varepsilon \tag{1.2}$$

Since $\|\nabla f(x^i)\| > \varepsilon$ we have the thesis. □

Per each phase the new epsilon is obtained $\varepsilon \leftarrow \varepsilon \|\nabla f(x^i)\|$.

If we can prove that the algorithm is converging we know when to stop.

This convergence isn't the perfect mathematical convergence, since $\varepsilon \neq 0$, because the line search will never terminate.

1.1.2 Exact line search, first orderd approach

We want to find the minimum points of φ, which corresponds to points where the first order derivative is zero and it goes from negative to positive. Since we are talking about numerical algorithms we are going to stop a little before the minimum is reached.

Key idea: We would like to reduce the range in which performing the search, at each step.

How can we be sure that in a given range there is a point where the derivative is 0? Rolle's theorem, as shown in Figure 1.1.

Since the gradient is continuous the directional derivative is continue, so φ is continuous (the scalar product is continuous).

Actually, we only need to find where the derivative is positive, because the 0 of the derivative is between the previous value and this point.

The algorithm is the following:

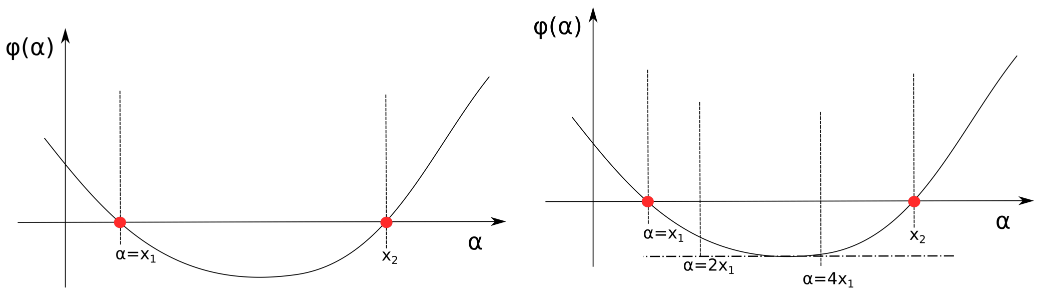

FIGURE 1.1: First, we restrict from $\mathbb{R}$ to $[x_1, x_2]$, then double α until the derivative is greater than 0.

ALGORITHM 1.2 First algorithm for exact line search

```
1: $\bar{\alpha} \leftarrow x_1$; # or whatever value $> 0$
2: **while** $(\varphi'(\bar{\alpha}) < 0)$ **do**
3:     $\bar{\alpha} \leftarrow 2\bar{\alpha}$; #or whatever factor $> 1$
4: **end while**
```

We'll stop when $\alpha < -10^{308}$, which is the smallest value for a double, since we will get a numerical error and stop.

Works if φ **coercive**: $\lim_{\alpha \to \infty} \varphi(\alpha) = \infty$ (e.g. f strongly convex).

Exercise 1.1. *Build an example where $\bar{\alpha}$ exists but it is not found by this algorithm.*

Solution: The function changes its derivative in a range between α and 2α.

An alternative to the algorithm presented above may be the bisection algorithm, which follows.

ALGORITHM 1.3 Bisection algorithm

```
1: **procedure** LSBM$((\varphi', \bar{\alpha}, \varepsilon))$
2:     $\alpha_- \leftarrow 0$;
3:     $\alpha \leftarrow \alpha_+$;
4:     $\alpha_+ \leftarrow \bar{\alpha}$;
5:     **while** $(|\varphi'(\alpha)| > \varepsilon)$ **do**
6:         $\alpha \leftarrow (\alpha_+ + \alpha_-)2$;
7:         **if** $(\varphi'(\alpha) < 0)$ **then**
8:             $\alpha_- \leftarrow \alpha$;
9:         **else**
10:            $\alpha_+ \leftarrow \alpha$;
11:        **end if**
12:    **end while**
13: **end procedure**
```

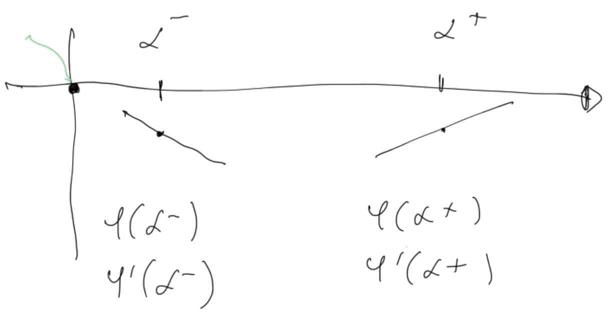

Figure 1.2: The information we have about function φ

We would like to improve this algorithm too, finding a better point in the middle than the middle point.

We may use the information we have about the function, since we know $\varphi(\alpha^-)$, $\varphi'(\alpha^-)$, $\varphi(\alpha^-)$ and $\varphi'(\alpha^-)$.

At this point we can write a model ($m(\alpha) = a\alpha^2 + b\alpha + c$) and specialize it with the information we have, via computing a linear system:

$$\begin{cases} a(\alpha^-)^2 + b(\alpha^-) + c = \varphi(\alpha) \\ a(\alpha^-)^2 + b(\alpha^-) + c = \varphi(\alpha) \\ 2a\alpha^- + b = \varphi'(\alpha^-) \\ 2a\alpha^+ + b = \varphi'(\alpha^+) \end{cases}$$

Then we look for the stationary point of this function and that's the ball where I'm likely to find the root of the derivative.

We can say something more, since the following fact holds:

Fact 1.4. *Let $\varphi \in C^3$, then quadratic interpolation has convergence of order $1 < p < 2$ (superlinear).*

In Figure 1.3 we can observe a situation in which the hypothesis of Proposition 1.4 aren't satisfied.

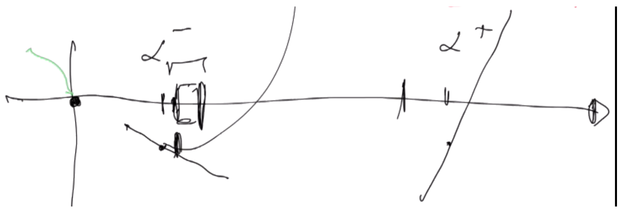

Figure 1.3: If the function isn't C^3 and one derivative is very big, then the range doesn't shrink much.

We would like to modify the formula to have at least linear convergence.

We can ensure to move not to close to one of the extremes, for example more than 10%.

Another idea is, since we have four equations, to build a cubic function, although the operation is very long. In this case the convergence get quadratic, which is better than linear.