Projeto 3 de Cálculo 2 Honors
Newton

Prof. Lucas Pedroso

2º semestre de 2024

Última atualização: 15/11 (confira no final do arquivo a lista com as modificações desde a publicação)

Este arquivo trata do achievement do método de Newton do Projeto 3.

1 Motivação teórica

No método do gradiente, usamos a direção do (menos) gradiente para a busca, ou seja, $d^k = -\nabla f(x^k)$, e andamos uma fração α_k dessa direção para obtermos o próximo ponto através da expressão $x^{k+1} = x^k + \alpha_k d^k$.

A ideia do método de Newton é usar uma direção de busca melhor que a do gradiente. Para tanto, aproximamos em torno de x^k a função a ser minimizada por Taylor de ordem 2 em torno de x^k :

$$f(x) \approx q(x) = \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k)(x - x^k) + \nabla f(x^k)^T (x - x^k) + f(x^k).$$

Como a quadrática q aproxima a função f, esperamos que o minimizador de q (que é fácil calcular) esteja próximo do minimizador da f (que desejamos). Derivando q e iqualando a 0, obtemos

$$\nabla q(x) = 0 \Rightarrow \nabla^2 f(x^k)(x - x^k) + \nabla f(x^k) = 0 \Rightarrow \nabla^2 f(x^k)(x - x^k) = -\nabla f(x^k)$$

Note que como x^k é fixo, chamando $d^k = x - x^k$, a expressão $\nabla^2 f(x^k) d^k = -\nabla f(x^k)$ é um sistema linear. Resolvendo esse sistema, é encontrada a direção de Newton.

Os dados de entrada e saída do algoritmo acima são os mesmos que os do algoritmo do gradiente. Essencialmente, em suas versões mais simples, a única diferença entre os métodos do gradiente e de Newton é o cálculo da direção: para o método do gradiente temos $d^K = -\nabla f(x^k)$ e para o de Newton $d^k = -(\nabla^2 f(x^k))^{-1} \nabla f(x^k)$.

Algorithm 1 Método de Newton

Data: $f : \mathbb{R}^n \to \mathbb{R}$, $x^0 \in \mathbb{R}^n$, $\alpha, \varepsilon > 0$, $K \in \mathbb{N}$.
Initialization: $x \leftarrow x^0$ e $k \leftarrow 0$.
  while $\|\nabla f(x)\| > \varepsilon$ e $k < K$ **do**
    $k \leftarrow k + 1$.
    Solve $\nabla^2 f(x) d = -\nabla f(x)$
    $x \leftarrow x + \alpha d$.
  end while
  return x, k

2 Implementação

A implementação será bem semelhante à do método do gradiente.

2.1 Primeira linha da função

```
def newton(f,x0,grad,hess,eps = 1e-5,alpha = 0.1,itmax = 10000,fd =
           False,h = 1e-7,plot = False,search = False,):
```

Os dados de entrada já foram explicados no método do gradiente, exceto por

- `hess`: função que calcula a Hessiana da função no ponto. Será explicada abaixo;

A saída também será igual à do método do gradiente.

Sobre a função `hess`: deve ter como entrada um ponto `x`, que é um numpy array com n componentes, e como saída a matriz Hessiana como numpy array $n \times n$. Por exemplo, se tivermos

```
def f(x):
   return x[1]**2*np.sin(x[0])
```

então teremos

```
def grad(x):
   return np.array([x[1]**2*np.cos(x[0]),
                  2*x[1]*np.sin(x[0]))])
```

e

```
def hess(x):
   return np.array([[-x[1]**2*np.sin(x[0]),2*x[1]*np.cos(x[0])]
                  [-2*x[1]*np.cos(x[0])),2*np.sin(x[0]))]])
```

3 Exemplo

Considere

```
def f(x):
   return x[0]**4-2*x[0]**2+x[0]-x[0]*x[1]+x[1]**2
```

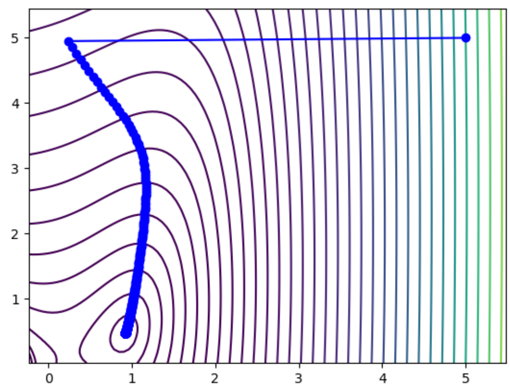
```
def grad(x):
   return np.array([4*x[0]**3-4*x[0]+1-x[1],-x[0]+2*x[1]])
def hess(x):
   return np.array([[12*x[0]**2-4,-1],[-1,2]])
```

Ao rodarmos

```
x,k = gd(f,np.array([5,5]),grad,alpha=1e-2,eps = 1e-6,plot=True)
print(f"x = x")
print(f"k = k")
```

obtemos a saída

```
x = [0.92442515 0.46221306]
k = 879
```

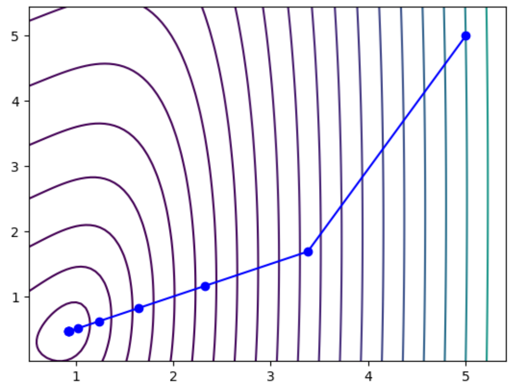


Já com o método de Newton, ao rodarmos

```
x,k = newton(f,np.array([5,5]),grad,hess,alpha=1e0,eps = 1e-6,plot=True)
print(f"x = x")
print(f"k = k")
```

obtemos

```
x = [0.92442502 0.46221251]
k = 9
```



ou seja, a convergência se dá em um número muito menor de iterações.

4 Detalhes e comentários

- Ao se fazer Newton sem busca linear, em geral se usa o passo constante $\alpha = 1$;
- A convergência de Newton é quadrática, enquanto que a do método do gradiente é linear. Isso em resumo diz que Newton converge bem mais rápido em geral (mas não sempre).
- A direção de Newton pode ser de subida! Por isso ter salvaguardas (que é um achievement do projeto) pode ser crucial dependendo do problema;
- As salvaguardas também são importantes caso a matriz Hessiana seja singular ou quase singular, pois nesse caso o sistema $\nabla^2 f(x^k)d^k = -\nabla f(x^k)$ pode não ter solução ou ter infinitas;
- Assim como o método do gradiente, Newton em geral funciona melhor usando busca linear;
- A direção de Newton pode ser escrita como $d^k = -(\nabla^2 f(x^k))^{-1}\nabla f(x^k)$. Porém calcular inversa é muito caro computacionalmente, por isso preferimos sempre resolver o sistema linear $\nabla^2 f(x^k)d^k = -\nabla f(x^k)$ em vez de calcular a inversa $(\nabla^2 f(x^k))^{-1}$ e multiplicar por $-\nabla f(x^k)$.