Resultados dos Problemas de Otimização - Método do Gradiente Espelhado

Análise Computacional

16 de setembro de 2025

O **Gradiente Espelhado** (também conhecido como **Mirror Descent**) é um algoritmo de otimização que generaliza o método do gradiente descendente clássico. Ele é particularmente útil para problemas de otimização em espaços não-euclidianos e quando queremos incorporar informações sobre a geometria do problema através de uma função de distância.

1 Conceitos Fundamentais

1.1 Divergência de Bregman

O algoritmo do gradiente espelhado baseia-se na **divergência de Bregman**, que é uma generalização da distância euclidiana. Para uma função convexa ϕ, a divergência de Bregman é definida como:

$$D_\phi(x, y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y \rangle \tag{1}$$

onde $\nabla\phi(y)$ é o gradiente de ϕ no ponto y.

1.2 Função de Distância

No nosso algoritmo, utilizamos a **função euclidiana**:

$$\phi(x) = \frac{1}{2}\|x\|^2 = \frac{1}{2}\sum_{i=1}^{n} x_i^2 \tag{2}$$

$$\nabla\phi(x) = x \tag{3}$$

Para esta função, a divergência de Bregman se reduz à distância euclidiana ao quadrado:

$$D_\phi(x, y) = \frac{1}{2}\|x - y\|^2 \tag{4}$$

2 Algoritmo do Gradiente Espelhado

2.1 Formulação do Problema

Dado um problema de minimização:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{5}$$

onde f é uma função diferenciável.

2.2 Algoritmo

O algoritmo do gradiente espelhado funciona da seguinte forma:

[label=**Passo 2:**]

1. **Inicialização:** Escolha um ponto inicial x_0 e parâmetros $\eta > 0$ (taxa de aprendizado) e tolerância ϵ.

2. **Iteração k:**

   [label=(b)]

   (a) Calcule o gradiente: $g_k = \nabla f(x_k)$

   (b) Resolva o subproblema:

$$x_{k+1} = \arg\min_x \left\{ \langle g_k, x \rangle + \frac{1}{\eta} D_\phi(x, x_k) \right\} \tag{6}$$

   (c) Verifique convergência: se $\|x_{k+1} - x_k\| < \epsilon$, pare.

2.3 Resolução do Subproblema

Para a função euclidiana $\phi(x) = \frac{1}{2}\|x\|^2$, o subproblema se torna:

$$x_{k+1} = \arg\min_x \left\{ \langle g_k, x \rangle + \frac{1}{2\eta}\|x - x_k\|^2 \right\} \tag{7}$$

$$= \arg\min_x \left\{ \langle g_k, x \rangle + \frac{1}{2\eta}\|x\|^2 - \frac{1}{\eta}\langle x_k, x \rangle + \frac{1}{2\eta}\|x_k\|^2 \right\} \tag{8}$$

Tomando o gradiente e igualando a zero:

$$g_k + \frac{1}{\eta}(x_{k+1} - x_k) = 0 \tag{9}$$

Portanto:

$$x_{k+1} = x_k - \eta g_k \tag{10}$$

3 Implementação no Código

3.1 Estrutura Principal

O algoritmo implementado segue a seguinte estrutura:

```
def gradiente_espelhado(f, x0, eta, max_iter, tol, phi, phi_grad):
    x = x0
    fo = f(x0)  # Valor inicial da função

    for k in range(max_iter):
        # 1. Calcular gradiente
        grad_f = calcular_gradiente(f, x)

        # 2. Resolver subproblema
```

```
        x_novo, f_novo = resolver_subproblema(grad_f, x, eta, phi, phi_grad)

        # 3. Verificar convergência
        if ||x_novo - x|| < tol:
            break

        x = x_novo
        fo = f(x)

    return x, fo, k
```

3.2 Cálculo do Gradiente

O gradiente é calculado numericamente usando diferenças finitas centrais:

```
def calcular_gradiente(f, x):
    n = len(x)
    grad = zeros(n)

    for i in range(n):
        h = 1e-6
        x_plus = x.copy()
        x_plus[i] += h
        x_minus = x.copy()
        x_minus[i] -= h

        grad[i] = (f(x_plus) - f(x_minus)) / (2 * h)

    return grad
```

3.3 Resolução do Subproblema

O subproblema é resolvido usando o otimizador BFGS do SciPy:

```
def resolver_subproblema(grad_f, x_atual, eta, phi, phi_grad):
    def funcao_subproblema(x):
        bregman_div = phi(x) - phi(x_atual) - dot(phi_grad(x_atual), x - x_atual)
        return dot(grad_f, x) + (1/eta) * bregman_div

    resultado = minimize(funcao_subproblema, x0=x_atual)
    return resultado.x, resultado.fun
```

4 Vantagens do Gradiente Espelhado

- **Generalização:** Funciona em espaços não-euclidianos
- **Flexibilidade:** Permite incorporar informações geométricas através da função ϕ
- **Convergência:** Garante convergência para funções convexas
- **Estabilidade:** Mais robusto que o gradiente descendente clássico em alguns casos

5 Parâmetros do Algoritmo

No código implementado, os parâmetros utilizados são:

- **Taxa de aprendizado (η):** 0.01 (padrão)
- **Máximo de iterações:** 1000 (padrão)
- **Tolerância:** 10^{-6} (padrão)
- **Precisão do gradiente:** $h = 10^{-6}$ (diferenças finitas)

6 Problemas de Otimização

O algoritmo foi testado em 16 problemas de otimização da coleção Liu-Nocedal:

A tabela 1 apresenta os problemas de otimização não-linear resolvidos usando o método do Gradiente Espelhado (Mirror Descent) e o número de variáveis de cada problema.

Tabela 1: Problemas de otimização e número de variáveis

Problema	**Número de Variáveis**
ROSENBROCK	10
PENALTY	5
TRIGONOMETRIC	10
EXTENDED ROSENBROCK	10
EXTENDED POWELL	12
QOR	50
GOR	50
PSP	50
TRIDIAGONAL	10
ENGGVAL1	10
LINEAR MINIMUM SURFACE	9
SQUARE ROOT 1	16
SQUARE ROOT 2	16
FREUDENTHAL ROTH	10
SPARSE MATRIX SQRT	10
ULTS0	64

7 Resultados

A tabela 2 apresenta os resultados de convergência para cada problema, incluindo o número de iterações necessárias, o valor mínimo da função objetivo encontrado e a precisão da solução (norma do gradiente).

A tabela 3 apresenta as primeiras 5 variáveis da solução encontrada para cada problema. Para problemas com menos de 5 variáveis, apenas as variáveis disponíveis são mostradas.

Tabela 2: Resultados de convergência dos problemas de otimização

Problema	**Iterações**	**Valor Mínimo**	**Precisão** ($||\nabla f(x^*)||$)	**Tempo (s)**
ROSENBROCK	0	0.000000e+00	3.999688e-10	0.000s
PENALTY	332	9.147106e-02	9.737285e-05	0.336s
TRIGONOMETRIC	999	1.709066e-04	6.868246e-03	1.002s
EXTENDED ROSENBROCK	0	0.000000e+00	8.943575e-10	0.000s
EXTENDED POWELL	7	1.430492e+35	1.591747e+27	0.106s
QOR	716	1.175472e+03	2.819951e-04	169.942s
GOR	999	1.381140e+03	1.061535e-01	209.612s
PSP	999	2.029601e+02	1.339710e+01	164.278s
TRIDIAGONAL	713	3.499057e-09	9.660990e-05	0.659s
ENGGVAL1	464	9.177470e+00	9.722345e-05	0.445s
LINEAR MINIMUM SURFACE	999	3.469804e+00	6.846937e-01	15.453s
SQUARE ROOT 1	999	1.075595e-04	7.032637e-03	1.780s
SQUARE ROOT 2	999	4.521564e-04	3.954493e-03	1.716s
FREUDENTHAL ROTH	3	8.915542e+81	0.000000e+00	0.069s
SPARSE MATRIX SQRT	999	1.799093e-04	8.526912e-03	1.451s
ULTS0	3	5.086273e+28	0.000000e+00	1.047s

Tabela 3: Primeiras 5 variáveis das soluções encontradas

Problema	**x1**	**x2**	**x3**	**x4**	**x5**
ROSENBROCK	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
PENALTY	9.815904e-01	9.815905e-01	9.815905e-01	9.815904e-01	9.815906e-01
TRIGONOMETRIC	6.055791e-02	6.273479e-02	6.530700e-02	6.847020e-02	7.277235e-02
EXTENDED ROSENBROCK	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
EXTENDED POWELL	1.764691e+00	-9.190465e+07	1.895674e+08	1.336783e+00	1.764693e+00
QOR	5.923306e-01	-7.111491e-01	6.285711e-02	-2.651213e+00	1.583524e+00
GOR	-1.789059e+00	-3.514595e-01	-3.031726e+00	-1.043714e-01	7.394256e+00
PSP	4.999685e+00	4.943255e+00	5.002957e+00	2.903052e+00	4.995765e+00
TRIDIAGONAL	1.000054e+00	5.000361e-01	2.500240e-01	1.250159e-01	6.251044e-02
ENGGVAL1	9.010301e-01	5.458806e-01	6.512110e-01	6.240699e-01	6.319669e-01
LINEAR MINIMUM SURFACE	3.404830e+00	6.562140e+00	6.361524e+00	6.568676e+00	4.510520e+00
SQUARE ROOT 1	8.316581e-01	-1.267529e-01	-6.284486e-01	-5.949608e-01	-7.664529e-01
SQUARE ROOT 2	7.378219e-01	-2.573644e-01	-6.349570e-02	-6.657820e-01	-7.189708e-01
FREUDENTHAL ROTH	-6.970890e-01	-3.857246e+11	-3.382124e+11	-3.381847e+11	-3.382064e+11
SPARSE MATRIX SQRT	8.316120e-01	-7.320686e-01	4.043655e-01	-2.611206e-01	-1.323311e-01
ULTS0	1.679770e+06	-2.082768e+06	2.461385e+06	1.893953e+06	6.565687e+05