Trabalho de Programação Linear

Luiz Fernando Bossa
Profa. Dra. Melissa Weber Mendonça

19 de julho de 2016

1 Motivação

Implementei meus algoritmos em Python por se tratar de código livre e que pode ser rodado em qualquer computador. Usando a biblioteca `numpy`, temos basicamente as mesmas funcionalidades que o tão difundido MATLAB®.

Além disso, usei classes para permitir uma melhor portabilidade no código, que está disponível em

github.com/LFBossa/simplex

O uso dos notebooks de IPython permitiu criar documentos que mesclam explicações da teoria, códigos para rodar os exemplos e inclusão de imagens interativas.

2 Simplex Primal

2.1 Modo de Usar

Dado o problema de minimização na forma padrão

$$\begin{aligned} &\min\ c^T x \\ &\text{s.a. } Ax = b \\ &\qquad x \geq 0 \end{aligned} \tag{P}$$

tudo o que temos que fazer é escrever a matriz A como um array bidimensional, e b e c como arrays unidimensionais. Importamos a classe `SimplexPrimal` do arquivo `metodosimplex.py`. Em seguida, criamos uma instância da classe `SimplexPrimal` e usamos seu método `resolver()`, da seguinte forma.

```
>>> P = SimplexPrimal(A,b,c)
>>> P.resolver()
```

O pivotamento é mostrado na tela, e ao final, se a região viável não for vazia e o problema tiver uma solução ótima limitada, ela é exibida na tela.

2.2 Funcionamento

Quando iniciamos uma instância, já é chamado o método `check()` para verificar se $b \geq 0$. Caso alguma entrada de b seja negativa, ele retorna um `ValueError` explicando que b deve ser maior que zero.

Quando chamamos o método `resolver()`, ele verifica se existe uma base igual a matriz identidade entre as colunas da matriz A. Se sim, ele chama o método `jatembase()`. Se não, ele chama os métodos `fase1()` e, em seguida, `fase2()`.

O método `jatembase()` cria o tableau, e chama o método `run()`.

A `fase1()` verifica quais colunas faltam para termos uma base igual a identidade, adiciona as variáveis artificiais correspondentes, monta o tableau e chama o método `run()`. Após isso, ele verifica se existe alguma variável artificial que continuou na base. Se sim e o valor dessa variável for zero, tenta removê-la (ou apaga a linha correspondente no tableau se isso não for possível). Se alguma variável artificial que está na base tiver valor diferente de zero, ele retorna o erro `SemSolucoesViaveis`.

A `fase2()` recebe o tableau da `fase1()`, elimina as colunas correspondentes as variáveis artificiais, coloca o vetor dos custos no tableau e chama o método `run()`.

O método `run()` é a implementação do pivotamento. Ele decide quem vai entrar na base pegando o maior elemento positivo da última linha do tableau, e chama o método `quem_sai_da_base()`. Este último, por sua vez, usa o teste da razão e a validação lexicográfica para decidir qual variável sairá da base, e se nenhuma variável puder sair da base, ele retorna o erro `ProblemaIlimitado`.

2.3 Exemplos

No arquivo `ExemplosSimplex.ipynb` temos exemplos de problemas com base inicial, sem base inicial, soluções ilimitadas, conjuntos viáveis vazios e redundância.

3 Simplex Dual

3.1 Modo de Usar

Idêntico ao `SimplexPrimal`:

```
>>> P = SimplexDual(A,b,c)
>>> P.resolver()
```

3.2 Funcionamento

A classe `SimplexDual` é filha da classe `SimplexPrimal`, porém com alguns métodos modificados.

O método `check()` verifica se o vetor c é positivo, retornando um erro caso contrário.

Quando chamamos o método `resolver()`, ele assume que as últimas m colunas de A formam uma base, e chama o método `jatembase()`. Este último é exatamente igual ao da classe `SimplexPrimal`.

O método `run()` agora decide quem vai sair da base pegando o menor elemento negativo da última coluna do tableau, e chama o método `quem_entra_na_base()`.

Esse método `quem_entra_na_base()` usa o teste da razão e validação lexicográfica para decidir qual variável entrará na base. Se ninguém puder entrar, ele dá o erro `ProblemaIlimitado` significando que a solução para o problema *dual* é ilimitada (e consequentemente a região viável para o primal é vazia).

3.3 Exemplos

Os exemplos do método simplex dual estão no arquivo `ExemplosSimplex.ipynb`.

4 Pontos Interiores

O algoritmo primal-dual path following está implementado no arquivo `pontosinteriores.py`. Usamos o método de vizinhança central com $\gamma = 10^{-3}$.

4.1 Modo de usar

Para resolver o problema (P), temos que ter um ponto $x > 0$ viável para o primal, um ponto w viável para o dual cuja folga complementar $s = c - A^T w > 0$. Importamos a classe `PrimalDualPF`, iniciamos uma instância da mesma e usamos o método `solve()`.

```
>>> prob = PrimalDualPF(A,b,c,x,w,s)
>>> prob.solve()
```

Fazendo isso ele resolve o problema usando as configurações padrão de tolerância (`TOL=1e-3`), número máximo de passos (`MAXSTEPS=1000`) e σ (`SIGMA=.2`).

4.2 Funcionamento

Quando iniciamos a instância, o método ajusta $\gamma = 10^{-3}$ e calcula $\mu = \langle x, s\rangle / n$.

O método `solve()` é a implementação em si. Ele roda no máximo `MAXSTEPS` passos do método. Enquanto μ for maior do que `TOL`, ele grava os valores de x_1 e x_2 na variável `caminho`, e em seguida chama sucessivamente os métodos `update()` e `step()`.

O método `update()` atualiza a a matriz jacobiana e o lado direito do sistema linear

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S_k & 0 & X_k \end{bmatrix} \begin{bmatrix} p_k^x \\ p_k^w \\ p_k^s \end{bmatrix} = \begin{bmatrix} c - A^T w_k - s_k \\ b - Ax_k \\ -X_k S_k \mathbb{1} + \sigma \mu_k \mathbb{1} \end{bmatrix}$$

O método `step()` resolve o sistema linear acima e faz

$$(x_{k+1}, w_{k+1}, s_{k+1}) = (x_k, w_k, s_k) + \alpha_k (p_k^x, p_k^w, p_k^s)$$

com α_k escolhido usando *backtracking*.

4.3 Exemplos

O exemplo visto em sala de aula está no arquivo `ExemploPontosInteriores1.ipynb`. Com a ajuda do GeoGebra, eu construí outro exemplo e incluí no arquivo `ExemploPontosInteriores2.ipynb`.

5 Considerações Finais

Foi muito proveitoso fazer esse trabalho pois (re)aprendi a usar o Python. Usando a distribuição Anaconda, pude aprender a usar os pacotes científicos `numpy`, `matplotlib`, além de ter descoberto todas as funcionalidades dos notebooks de IPython.

Também foi muito bem-vinda a sugestão da professora de que eu criasse um GitHub para colocar meus códigos. Isso me incentivou a cuidar bem da legibilidade do mesmo, bem como a criação de documentação.