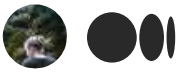

João Pedro Campos

Introdução à Programação Linear com PuLP

João Pedro Campos Just now · 6 min read

PuLP é um *framework* para modelagem e resolução de problemas de programação linear e programação inteira. Uma de suas vantagens é poder gerar arquivos para diversos tipos de *solver* disponíveis e populares, tais como IBM CPLEX, Gurobi, Mosek, etc.

Ok, mas vamos por partes. O que é programação linear?

Uma breve história da Programação Linear

Em matemática, um problema de programação linear é um problema de otimização onde tanto a função objetivo — que deseja maximizar ou minimizar uma função— quanto as restrições do problema são todas lineares, isto é, possuem grau 1.

Tudo começou na época da Segunda Guerra Mundial, em que países como Estados Unidos passaram a investir pesadamente em pesquisa e engenharia. Isso porque, no período da guerra, problemas de grande escala operacional começaram a surgir. Na época, era comum que físicos, engenheiros, dentre diversos outros pesquisadores, tirassem licença para servir as Forças Armadas americanas. Em uma dessas ocasiões, um jovem chamado George Dantzig integrou o cargo de chefe em uma subdivisão de combate, no Centro de Controle Estatístico da Força Aérea americana.

O trabalho de Dantzig não era lá muito simples. Ele deveria coletar e analisar dados de combates aéreos, bem como gerir a cadeia de suprimentos de sua divisão. Se hoje em dia esse tipo de problema já parece complicado, imagine nos anos 40, em que computadores com *python* instalado eram uma realidade muito, muito distante! (rsrs)

Os problemas, então, eram resolvidos manualmente, com réguas de cálculo e diversos aparatos para buscar a simplificação das contas matemáticas. Os problemas de otimização de recursos geraram a necessidade de criação de um modelo que conseguisse generalizar todos esses problemas num único método de solução, que fosse simples e elegante.

George Dantzig

Foi então que, poucos anos depois, em 1947, Dantzig gerou o primeiro modelo do método Simplex de otimização para problemas de programação linear, esse que foi fundamental para a criação de *frameworks* atuais, como o PuLP. E, apesar de seu relacionamento com o período de guerra, o método Simplex serve para resolver uma grande variedade de problemas, alguns muito atuais, como veremos a seguir.

Ok, agora que sabemos de todo esse contexto histórico, vamos resolver um problema de programação linear na prática!

O Problema da Wyndor Glass Co.

A empresa Wyndor Glass Co. deseja produzir dois novos produtos, e dispõe de 3 fábricas para produzi-los. Cada produto requer certa quantidade de horas em determinadas fábricas para serem concluídos, e geram diferentes lucros, como a seguir:

- Produto 1: necessita1 hora de produção na Fábrica 1 e 3 horas na Fábrica 3. Gera $ 3,000 de lucro por lote.
- Produto 2: necessita 2 horas de produção na fábrica 2 e 2 horas na Fábrica 3. Gera $ 5,000 de lucro por lote.

Contudo, as fábricas não estão sempre disponíveis para esses novos produtos, e o tempo disponível de cada uma, semanalmente, é o a seguir:

- Fábrica 1: 4 horas
- Fábrica 2: 12 horas
- Fábrica 3: 18 horas

Deseja-se, é claro, maximizar o lucro da produção desses dois novos produtos. Para isso, devemos encontrar a quantidade ótima de produtos a serem fabricados pelas fábricas 1, 2 e 3. Assumindo x1 como a quantidade fabricada do produto 1, e x2 como a quantidade fabricada do produto 2, podemos modelar a situação como um problema de programação linear:

Maximizar $Z = 3000*x1 + 5000*x2$

sujeito às restrições:

$x1 \leq 4,$

$2*x2 \leq 12,$

$3*x1 + 2*x2 \leq 18$

e

$x1 \geq 0, x2 \geq 0.$

Modelagem do problema Wyndor Glass Co.

A primeira linha é chamada de função-objetivo do problema, e resulta numa certa quantidade de dólares, gerados semanalmente. O restante das linhas é formado pelo que chamamos de restrições do problema. Por exemplo: como o produto 1 necessita de 1 hora na fábrica 1 para ser produzido, então escrevemos que a quantidade de produtos fabricados semanalmente tem de ser menor que as 4 horas disponíveis na fábrica 1. Ou seja: $1*x1 \leq 4$. E o mesmo se faz para as outras fábricas. E tem mais! Já que x1 e x2 representam quantidades de produtos reais, eles não podem ser negativos! Por isso, as últimas restrições são as chamas restrições de não-negatividade.

Uma das formas de se resolver o problema é utilizar, manualmente, o algoritmo Simplex, criado por Dantzig. Contudo, esse não é o intuito desse artigo, visto que, na

vida real, matemáticos e engenheiros utilizam *softwares* para resolver esse tipo de coisa. Portanto, vamos implementar esse problema, agora modelado, no PuLP!

Implementação no PuLP

Depois de instalar o PuLP (para fazer isso com o pip, basta utilizar o comando "pip install pulp", em sua máquina) já podemos começar a escrever o nosso código. Precisamos, é claro, importar a biblioteca para utilizá-la. Aqui, chamaremos o PuLP pelo codinome plp, para facilitar a escrita.

```
import pulp as plp
```

Agora, devemos informar ao PuLP quais são os parâmetros do problema de programação linear que queremos resolver. Nesse caso, desejamos resolver um problema de maximização. Assim, escrevemos:

```
# Definindo o problema

max_z = plp.LpProblem("Wyndor_Glass_Co._Problem", plp.LpMaximize)
```

A classe .LpProblem recebe dois parâmetros: o nome do problema, e seu tipo: se é de minimização (*default* para o PuLP) ou de maximização. Atribuímos o problema à uma variáveis — do Python — chamada max_z, cujo nome é arbitrário.

Em seguida, vamos definir as variáveis do problema. Como foi mostrado anteriormente, x1 foi definido como a quantidade do produto 1 a ser fabricada semanalmente pelas fábricas, e analogamente para o produto 2.

```
# Definindo as variáveis do problema

x1 = plp.LpVariable('x1', lowBound=0, cat='Integer') # Não negativa
x2 = plp.LpVariable('x2', lowBound=0, cat='Integer') # Não negativa
```

A classe .LpVariable do PuLP serve para definirmos as variáveis. Aqui, inserimos três parâmetros importantes: o nome da variável, seu respectivo intervalo de domínio e a categoria da variável. O primeiro parâmetro é fácil, basta inserir uma *string* para o nome. Como no nosso problema as variáveis não podem assumir valores negativos, o menor valor possível para elas é zero. Por isso, no segundo parâmetro, escrevemos lowBound=0. Por fim, no nosso problema, não podemos produzir valores quebrados

de produtos (por exemplo, fabricar 0,33 produtos 1). Portanto, escrevemos que a categoria das variáveis é inteira: cat='Integer'.

Finalmente, podemos colocar os dados do problema:

```
# Escrevendo a função-objetivo

max_z += 3000*x1+5000*x2
```

```
# Escrevendo as restrições

max_z += x1 <= 4
max_z += 2*x2 <= 12
max_z += 3*x1 + 2*x2 <= 18
```

Note que não foi necessário inserir as restrições de não-negatividade, pois na definição das variáveis, já foi definido que elas não podem ser menores que zero.

Pronto! Já implementamos o problema no PuLP. Foi bem de boas, né? Agora, vamos resolver o modelo! Para isso, utilizamos o comando:

```
# Resolvendo

max_z.solve()
```

```
1
```

O *output* 1 retornado indica que foi encontrada uma solução ótima. E, para descobrirmos os valores de x1 e x2, escrevemos:

```
# Resultado

print(x1.varValue)
print(x2.varValue)
```

```
2.0
6.0
```

Descobrimos então que a empresa Wyndor Glass Co. deve produzir 2 produtos 1 e 6 produtos 2 durante a semana para maximizar o seu lucro! Este que pode ser calculado inserindo os valores das variáveis na função-objetivo. Ou, então, chamamos o seguinte comando:

```
# Valor ótimo

plp.value(max_z.objective)
```

```
36000.0
```

Ou seja, o lucro semanal da Wyndor Glass Co. com a venda dos novos produtos 1 e 2 será de $ 36,000.00.

Linear Programming Data Science

About Help Legal

Get the Medium app