5.3 선형계획법 문제와 이차계획법 문제

5.1절과 5.2절에서는 일반적인 최적화 문제를 다루었다. 하지만 데이터 분석에서는 목적함수나 제한조건이 특정한 수식은 최적화 문제가 많이 등장한다. 이 절에서는 그 중 선형 계획법과 이차계획법을 소개한다.

선형계획법 문제

방정식이나 부등식 제한 조건을 가지는 선형 모형(linear model)의 값을 최소화하는 문제를 **선형계획법(Linear Programming)** 문제라고 한다. LP 문제라고도 한다.

선형계획법 문제의 목적함수는

$$\arg\min_x c^T x \tag{5.3.1}$$

이고 선형 연립방정식으로 된 등식 제한조건

$$Ax = b \tag{5.3.2}$$

과 변수값이 모두 음수가 아니어야하는 부등식 제한조건

$$x \geq 0 \tag{5.3.3}$$

를 동시에 가진다.

선형계획법 문제는 여러가지 형태가 존재하는데 위와 같은 형태를 선형계획법 문제의 기본형(standard form)이라고 한다. 마지막 부등식 제한 조건은 벡터 x의 모든 원소가 양수거나 0이 되어야 한다는 것을 의미한다. 표준형을 확장한 정규형(canonical form) 선형계획법 문제는 부등식 조건을 허용한다.

$$\arg\min_x c^T x \tag{5.3.1}$$

$$Ax \leq b \tag{5.3.5}$$

$$x \geq 0 \tag{5.3.3}$$

예제

어떤 공장에서 두가지 제품을 생산해야 한다고 하자.

- 제품 A와 제품 B 각각 100개 이상 생산해야 한다.
- 시간은 500시간 밖에 없다.
- 제품 A는 생산하는데 1시간이 걸리고 제품 B는 2시간이 걸린다.
- 특정 부품이 9800개밖에 없다.
- 제품 A는 생산하는데 특정 부품을 4개 필요로 하고 제품 B는 생산하는데 특정 부품을 5개 필요로 한다.
- 제품 A의 이익은 하나당 3만원이고 제품 B의 이익은 하나당 5만원이다.

제품 A와 제품 B의 생산량을 각각 x_1, x_2라고 하면 최소화하려는 목적함수는

$$-3x_1 - 5x_2 \tag{5.3.7}$$

이고 제한 조건은 다음과 같다.

$$\begin{array}{rrrr} -x_1 & & \leq & -100 \\ & -x_2 & \leq & -100 \\ x_1 + & 2x_2 & \leq & 500 \\ 4x_1 + & 5x_2 & \leq & 9800 \end{array} \tag{5.3.8}$$

$$x_1 \geq 0, \;\; x_2 \geq 0 \tag{5.3.9}$$

이를 정규형 선형계획법 문제로 표현하면 다음과 같다.

$$\min_x \begin{bmatrix} -3 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{5.3.10}$$

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 2 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} -100 \\ -100 \\ 500 \\ 9800 \end{bmatrix} \tag{5.3.11}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.3.12}$$

사이파이를 이용한 선형계획법 문제 계산

scipy.optimize 패키지의 `linprog()` 명령을 사용하면 선형계획법 문제를 풀 수 있다. 사용법은 다음과 같다.

```
linprog(c, A, b)
```

- `c` : 목적함수의 계수 벡터
- `A` : 등식 제한조건의 계수 행렬
- `b` : 등식 제한조건의 상수 벡터

예제

다음 코드는 위 예제 선형계획법 문제를 사이파이로 계산하는 코드다.

In [1]:

```
import scipy.optimize

A = np.array([[-1, 0], [0, -1], [1, 2], [4, 5]])
b = np.array([-100, -100, 500, 9800])
c = np.array([-3, -5])

result = sp.optimize.linprog(c, A, b)
result
```

Out[1]:

```
     con: array([], dtype=float64)
     fun: -1400.0
 message: 'Optimization terminated successfully.'
     nit: 3
   slack: array([ 200.,    0.,    0., 8100.])
  status: 0
 success: True
       x: array([300., 100.])
```

제품 A를 300개, 제품 B를 100개 생산할 때 이익이 1400으로 최대가 됨을 알 수 있다.

CVXPY를 이용한 선형계획법 문제 계산

CVXPY 또는 PuLP와 같은 파이썬 패키지를 사용하면 선형계획법 문제의 계수 행렬 A, b, c를 직접 숫자로 정의하지 않고 심볼로 정의하여 더 직관적인 파이썬 코드를 만들 수 있다. 다음 코드는 위에서 풀었던 예제를 CVXPY로 다시 계산한 것이다. 다만 이 방법은 변수나 조건의 수가 아주 많을 경우에는 심볼릭 연산으로 인해 속도가 느려질 수 있다.

CVXPY는 conda 패키지 매니저로 설치할 수 있다.

```
conda install cvxpy
```

In [2]:

```
import cvxpy as cp

# 변수의 정의
a = cp.Variable()  # A의 생산량
b = cp.Variable()  # B의 생산량

# 조건의 정의
constraints = [
    a >= 100,  # A를 100개 이상 생산해야 한다.
    b >= 100,  # B를 100개 이상 생산해야 한다.
    a + 2 * b <= 500, # 500시간 내에 생산해야 한다.
    4 * a + 5 * b <= 9800,  # 부품이 9800개 밖에 없다.
]

# 문제의 정의
obj = cp.Maximize(3 * a + 5 * b)
prob = cp.Problem(obj, constraints)

# 계산
prob.solve()

# 결과
print("상태:", prob.status)
print("최적값:", a.value, b.value)
```

```
상태: optimal
최적값: 299.99999999999983 100.00000000000001
```

이차계획법 문제

방정식이나 부등식 제한 조건을 가지는 일반화된 이차형식(quadratic form)의 값을 최소화하는 문제를 **이차계획법(Quadratic Programming)** 문제라고 한다. QP 문제라고도 한다.

이차계획법 문제의 목적함수는

$$\frac{1}{2}x^TQx + c^Tx \tag{5.3.13}$$

이고 등식 제한조건과 부호 제한조건은 선형계획법 문제와 같다.

$$Ax = b \tag{5.3.14}$$

$$x \geq 0 \tag{5.3.15}$$

잔차 제곱합을 최소화하는 예측 모형에 추가적인 제한조건이 있으면 이차계획법 문제가 된다.

예제

앞 절에서 풀었던 등식 제한조건이 있는 최적화 문제도 사실은 이차계획법 문제다.

$$\arg\min_x x_1^2 + x_2^2 \tag{5.3.16}$$

$$x_1 + x_2 - 1 = 0 \tag{5.3.17}$$

이 문제를 QP 형식으로 바꾸면 다음과 같다.

$$\arg\min_x \frac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{5.3.18}$$

$$\begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1 \tag{5.3.19}$$

CvxOpt를 이용한 이차계획법 문제 계산

CvxOpt라는 패키지를 사용하면 이차계획법 문제를 풀 수 있다. CvxOpt를 쓸 때는 NumPy의 `ndarray` 배열을 CvxOpt 전용의 `matrix` 자료형으로 바꿔야 한다. 또 정수 자료형을 사용하지 못하므로 항상 부동소수점 실수가 되도록 명시해야 한다.

CvxOpt도 conda 패키지 매니저로 설치할 수 있다.

```
conda install cvxopt
```

In [3]:

```
from cvxopt import matrix, solvers

Q = matrix(np.diag([2.0, 2.0]))
c = matrix(np.array([0.0, 0.0]))
A = matrix(np.array([[1.0, 1.0]]))
b = matrix(np.array([[1.0]]))

sol = solvers.qp(Q, c, A=A, b=b)
np.array(sol['x'])
```

Out[3]:

```
array([[0.5],
       [0.5]])
```

연습 문제 5.3.1

다음 문제가 QP 문제임을 보이고 $N=3$인 경우에 대해 QP 문제의 Q, c, A, b를 각각 구하라(문제에서 x는 벡터이고 y는 실수다).

$$\arg\min_{a_i} \left(\sum_{i=1}^N a_i - \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N a_i a_j y_i y_j x_i^T x_j \right) \tag{5.3.20}$$

$$\sum_{i=1}^N a_i y_i = 0 \tag{5.3.21}$$

$$a_i \geq 0 \tag{5.3.22}$$