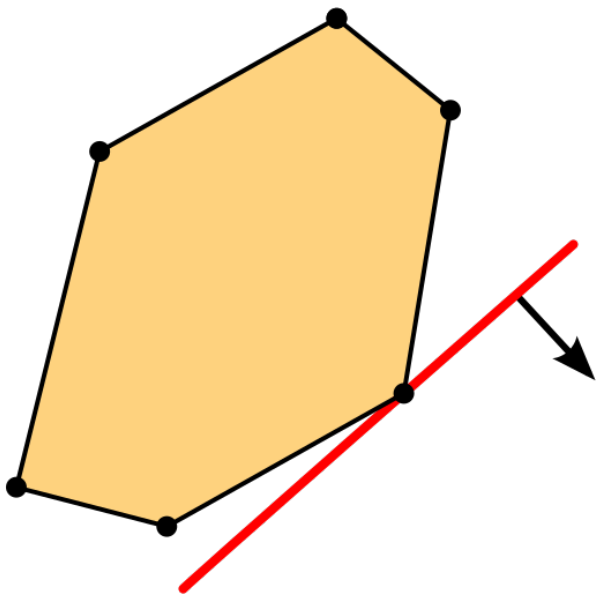
Optimization Notes

November 26, 2018

1 Linear Programming

Assumptions of Linear Programming

· Proportionality
· Additivity
· Divisibility
· Deterministic



Feed mix problem

Feed Mix Problem

An agricultural mill manufactures feed for chickens. This is done by mixing several ingredients, such as corn, limestone, or alfalfa. The mixing is to be done in such a way that the feed meets certain levels for different types of nutrients, such as protein, calcium, carbohydrates, and vitamins. To be more specific, suppose that n ingredients $j = 1,..., n$ and m nutrients $i = 1,..., m$ are considered. Let the unit cost of ingredient j be c_j and let the amount of ingredient j to be used be x_j. The cost is therefore $\sum_{j=1}^{n} c_j x_j$. If the amount of the final product needed is b, then we must have $\sum_{j=1}^{n} x_j = b$. Further suppose that a_{ij} is the amount of nutrient i present in a unit of ingredient j, and that the acceptable lower and upper limits of nutrient i in a unit of the chicken feed are ℓ_i' and u_i', respectively. Therefore, we

---

must have the constraints $\ell_i' b \leq \sum_{j=1}^{n} a_{ij} x_j \leq u_i' b$ for $i = 1,..., m$. Finally, because of shortages, suppose that the mill cannot acquire more than u_j units of ingredient j. The problem of mixing the ingredients such that the cost is minimized and the restrictions are met can be formulated as follows:

$$\text{Minimize}$$
$$\sum_{j=1}^{n} c_j x_j$$
$$\text{subject to}$$
$$\sum_{j=1}^{n} x_j = b$$
$$bl_i^{'} \leq \sum_{j=1}^{n} a_{ij} x_j \leq b u_i^{'}, \ i = 1, ..., m$$
$$0 \leq x_j \leq u_j, \ j = 1, ..., n$$

Numerical example

```
# An agricultural mill manufactures feed for chickens.
# This is done by mixing
# several ingredients, such as corn, limestone, or
# alfalfa. The mixing is to be done in
# such a way that the feed meets certain levels for
# different types of nutrients, such
# as protein, calcium, carbohydrates, and vitamins.
# Minimize cost.

using JuMP
using Clp

m=Model(solver = ClpSolver())

@variable(m, 0<=x<=7)
@variable(m, 0<=y<=4)

@objective(m, Min, 0.3*x + 0.6*y)

@constraint(m, 0.1<=0.1*x+0.3*y<=1.3)
@constraint(m, 0.2<=0.2*x+0.7*y<=1.4)
@constraint(m, 0.3<=0.05*x+0.17*y<=1.5)

status = solve(m)

println("Objective value: ", getobjectivevalue(m))
println("x = ", getvalue(x))
println("y = ", getvalue(y))
```

Numerical Output

```
Objective value: 1.0588235294117645
x = 0.0
y = 1.764705882352941
```

Small-scale farmer example

2. **Solve graphically a LP:** A small-scale farmer has 100 acres of land. He can use it as pasture for cows, at a profit of £10 per acre per year, or as arable land for growing crops, at a profit of £15 per acre per year; or he can leave it fallow at zero profit. Pasture requires 30 hours work per acre per year, arable needs 60 hours work per acre per year. The farmer wants to maximize profits, and is willing to work 4200 hours per year.

We will solve this problem numerically instead.

Numerical Example

```
using JuMP
using Clp
m=Model(solver=ClpSolver())
@variable(m,x[1:3]>=0)
@constraint(m,sum(x[i] for i=1:3)==100)
@constraint(m,30*x[1]+60*x[2]<=4200)
@objective(m,Max,10*x[1]+15*x[2])
status = solve(m)
println("Objective value: ", getobjectivevalue(m))
println("x[1]: ", getvalue(x[1]))
println("x[2]: ", getvalue(x[2]))
println("x[3]: ", getvalue(x[3]))
```

Numerical Output

```
Objective value: 1200.0
x[1]: 60.0
x[2]: 40.0
x[3]: 0.0
```

(b) How much extra profit will the farmer get, if he acquires an extra acre of land, the rest of the data being the same?

(c) What if he still has 100 acres, but the profit from pasture changes to £7 per acre per year?

The way to modify the JuMP program is trivial so we only show the output. For (b):

```
Objective value: 1205.0
x[1]: 62.0
x[2]: 39.0
x[3]: 0.0
```

For (c):

```
Objective value: 1050.0
x[1]: 0.0
```

```
x[2]: 70.0
x[3]: 30.0
```

Production scheduling: An optimal control problem

Production Scheduling: An Optimal Control Problem

A company wishes to determine the production rate over the planning horizon of the next T weeks such that the known demand is satisfied and the total production and inventory cost is minimized. Let the known demand rate at time t be $g(t)$, and similarly, denote the production rate and inventory at time t by $x(t)$ and $y(t)$, respectively. Furthermore, suppose that the initial inventory at time 0 is y_0 and that the desired inventory at the end of the planning horizon is y_T. Suppose that the inventory cost is proportional to the units in storage, so that the inventory cost is given by $c_1 \int_0^T y(t)\,dt$ where $c_1 > 0$ is known. Also, suppose that the production cost is proportional to the rate of production, and is therefore given by $c_2 \int_0^T x(t)\,dt$. Then the total cost is $\int_0^T [c_1 y(t) + c_2 x(t)]\,dt$. Also note that the inventory at any time is given according to the relationship

$$y(t) = y_0 + \int_0^t [x(\tau) - g(\tau)]\,d\tau, \text{ for } t \in [0,T].$$

Suppose that no backlogs are allowed; that is, all demand must be satisfied. Furthermore, suppose that the present manufacturing capacity restricts the production rate so that it does not exceed b_1 at any time. Also, assume that the available storage restricts the maximum inventory to be less than or equal to b_2. Hence, the production scheduling problem can be stated as follows:

Minimize

$$\int_0^T [c_1 y(t) + c_2 x(t)]dt$$

subject to

$$y(t) = y_0 + \int_0^t [x(\tau) - g(\tau)]dt$$

$$y(T) = y_T$$

$$0 \le x(t) \le b_1,\ t \in [0,T]$$

$$0 \le y(t) \le b_2,\ t \in [0,T]$$

Linear program approximation

Minimize

$$\sum_{j=1}^{n} (c_1 \Delta T) y_j + \sum_{j=1}^{n} (c_2 \Delta T) x_j$$

subject to

$$y_j = y_{j-1} + (x_j - g_j)\Delta T,\ j = 1, ..., n$$

$$y_n = y_T$$

$$0 \le x_j \le b_1,\ j = 1, ..., n$$

$$0 \le y_j \le b_2,\ j = 1, ..., n$$

where $n\Delta T = T$

Numerical example

```
# A company wishes to determine the production rate over the planning
# horizon of the next T weeks such that the known demand is satisfied
# and the total production and inventory cost is minimized.
```


```
# A company wishes to determine the production rate over the planning
# horizon of the next T weeks such that the known demand is satisfied
# and the total production and inventory cost is minimized.
```


```
using JuMP
using Clp


m=Model(solver = ClpSolver())

T = 4 # 4 weeks horizon
dT = 0.1

@expression(m, g[t=1:dT:T], -t+1)

@variable(m, 0 <= x[1:dT:T] <= 0.9)
@variable(m, 0 <= y[1:dT:T] <= 0.9, start=0.0)

@objective(m, Min, 2*dT*sum(x[round(i)] for i=1:dT:T) + 3*dT*sum(y[round(i)]
                                                                for i=1:dT:T))

@constraint(m, [t in 1:dT:(T-dT)], y[round(t+dT)] == y[t] +
                                        (x[round(t+dT)] - g[round(t+dT)]))

status = solve(m)

println("Objective value: ", getobjectivevalue(m))
```

The above is infeasible. I added it more as pseudo-code.

Manufacturing remedies example

3. **Optional: Formulate as LP, do not solve.** A Manufacturer of herbal medicine distills three essential medicine Components out of three different plant Extracts. The three Components are combined to form two different mixtures, called Remedy 1 and Remedy 2. The Manufacturer has to decide on the amounts of Extracts to buy and Remedies to produce, in order to maximize his gains. Formulate the task as a LP, given that:

   - To produce one gram of Remedy 1, one needs .25g., .35g. and .15g. of Components 1,2, and 3, respectively. The rest is alcohol, which is available free and unlimited (what a world would that be!). Remedy 1 sells for £10 per gram and the manufacturer cannot sell more than 100 grams of it.
   - To produce one gram of Remedy 2, one needs .20g., .10g. and .30g. of Components 1,2, and 3, respectively. The rest is alcohol. Remedy 2 sells for £13 per gram and the manufacturer cannot sell more than 130 grams.
   - One gram of Extract 1 contains .20g., .15g., and .25g. of Components 1,2, and 3, respectively, costs £3, and is available in unlimited quantities.
   - One gram of Extract 2 contains .30g., .30g., and 0g. of Components 1,2, and 3, respectively, costs £4, and no more than 1000 grams are available on the market.
   - One gram of Extract 3 contains .10g., .15g., and .45g. of Components 1,2, and 3, respectively, costs £5, and no more than 500 grams are available.
   - Distilling the Components out of Extracts and mixing the Remedies does not involve any additional costs.

But we can formulate it as a LP and solve it numerically!

Maximize

$$10r_1 + 13r_2 - 3e_1 - 4e_2 - 5e_3$$

subject to

$$0.25r_1 + 0.2r_2 - 0.2e_1 - 0.3e_2 - 0.1e_3 \leq 0$$

$$0.35r_1 + 0.1r_1 - 0.15e_1 - 0.3e_2 - 0.15e_3 \leq 0$$

$$0.15r_1 + 0.3r_2 - 0.25e_1 - 0.45e_3 \leq 0$$

$$0 \leq r_1 \leq 100, 0 \leq r_2 \leq 130, e_1 \geq 0, 0 \leq e_2 \leq 1000, 0 \leq e_3 \leq 500$$

where r_i are the remedies, e_j are the extracts and the constraints reflect the fact that sufficient extracts are needed to produce the remedies.

Numerical Example

```
using JuMP
using Clp

m=Model(solver=ClpSolver())

@variable(m, r[1:2]>=0)
@variable(m, e[1:3]>=0)
```

```
@constraint(m, 0.25*r[1]+0.2*r[2]-0.2*e[1]-0.3*e[2]-0.1*e[3]<=0)
@constraint(m, 0.35*r[1]+0.1*r[2]-0.15*e[1]-0.3*e[3]-0.15*e[3]<=0)
@constraint(m, 0.15*r[1]+0.3*r[2]-0.25*e[1]-0.45*e[3]<=0)
@constraint(m, r[1]<=100); @constraint(m, r[2]<=130);
@constraint(m, e[2]<=1000); @constraint(m, e[3]<=500);

@objective(m, Max, 10*r[1]+13*r[2]-3*e[1]-4*e[2]-5*e[3])

status = solve(m)

println("Objective value: ", getobjectivevalue(m))
println("Remedy 1: ", round(getvalue(r[1])))
println("Remedy 2: ", getvalue(r[2]))
println("Extract 1: ", round(getvalue(e[1])))
println("Extract 2: ", getvalue(e[2]))
println("Extract 3: ", round(getvalue(e[3])))
```

Numerical Output

```
Objective value: 1834.0
Remedy 1: 100.0
Remedy 2: 130.0
Extract 1: 242.0
Extract 2: 0.0
Extract 3: 26.0
```

Investment plan example

[1.1] Fred has $5000 to invest over the next five years. At the beginning of each year he can invest money in one– or two–year time deposits. The bank pays 4 percent interest on one–year time deposits and 9 percent (total) on two–year time deposits. In addition, West World Limited will offer three–year certificates starting at the beginning of the second year. These certificates will return 15 percent (total). If Fred reinvests his money that is available every year, formulate a linear program to show him how to maximize his total cash on hand at the end of the fifth year.

Minimize

$$\sum_{i=1}^{3} x(i) \cdot \text{interest}(i) \sum_{t=1}^{\text{floor}(5/\text{duration}(i))} (1+\text{interest}(i))^{t-1}$$

subject to

$$0 \leq \sum_{i=1}^{3} x(i) \leq 5000$$

where $i \in \{\text{1-year, 2-year, 3-year}\},\ t = 1, ..., 5$

Numerical Example

```
# Fred has $5000 to invest over the next five years.
# At the beginning of each
# year he can invest money in one- or two-year time deposits.
# The bank pays 4
# percent interest on one-year time deposits
# and 9 percent (total) on two-year
# time deposits.
# In addition, West World Limited will offer three-year
# certificates starting at the beginning of the second year.
# These certificates will return 15 percent (total).
# If Fred reinvests his money that is available
# every year,
# formulate a linear program to show him
# how to maximize his total cash on hand
# at the end of the fifth year.

using JuMP
using Clp

m=Model(solver=ClpSolver())

portfolios = ["1-year","2-year","3-year"]
```

```
t = [1,2,3,4,5]
interest = [0.04, 0.09, 0.15]
duration = [1,2,3]
numportf = length(portfolios)
numt = length(t)


@variable(m, x[1:numportf]>=0)

@constraint(m, sum(x[i] for i=1:numportf)<=5000)

@objective(m, Max, sum(x[i]*interest[i]*sum((1+interest[i])^(t-1)
                    for t=1:Int(floor(5/duration[i]))) for i=1:numportf))

status = solve(m)

println("Objective value: ", getobjectivevalue(m))
println("x[1]: ", getvalue(x[1]))
println("x[2]: ", getvalue(x[2]))
println("x[3]: ", getvalue(x[3]))
```

Numerical Output

```
Objective value: 1083.264512
x[1]: 5000.0
x[2]: 0.0
x[3]: 0.0
```

Blending as a linear program

[1.2] A manufacturer of plastics is planning to blend a new product from four chemical compounds. These compounds are mainly composed of three elements: A, B, and C. The composition and unit cost of these chemicals are shown in the following table:

CHEMICAL COMPOUND	1	2	3	4
Percentage A	35	15	35	25
Percentage B	20	65	35	40
Percentage C	40	15	25	30
Cost/kilogram	20	30	20	15

The new product consists of 25 percent element A, at least 35 percent element B, and at least 20 percent element C. Owing to side effects of compounds 1 and 2, they must not exceed 25 percent and 30 percent, respectively, of the content of the new product. Formulate the problem of finding the least costly way of blending as a linear program.

Minimize

$$\sum_i c(i)x(i)$$

subject to

$$\sum_i p(1,i)x(i) = 0.25$$

$$\sum_i p(2,i)x(i) \geq 0.35$$

$$\sum_i p(3,i)x(i) \geq 0.2$$

$$x(1) \leq 0.25$$

$$x(2) \leq 0.3$$

where $i \in \{1,2,3,4\},\ j \in \{A,B,C\},\ c(i)$: cost,

and $p(j,i)$ is the percentage of element j in compound i

Numerical Example

```
# A manufacturer of plastics is planning to blend a new product from four
# chemical compounds. Formulate the problem of finding the least costly way of
# blending as a linear program.

using JuMP
using Clp

m = Model(solver=ClpSolver())
```

```
p = zeros((3,4))

p[1,1] = 0.35; p[1,2] = 0.15; p[1,3] = 0.35; p[1,4] = 0.25;
p[2,1] = 0.20; p[2,2] = 0.65; p[2,3] = 0.35; p[2,4] = 0.40;
p[3,1] = 0.40; p[3,2] = 0.15; p[3,3] = 0.25; p[3,4] = 0.30;

c = [20,30,20,15]

@variable(m, x[1:4]>=0)

@constraint(m, x[1]<=0.25)
@constraint(m, x[2]<=0.30)

@constraint(m, sum(p[1,i]*x[i] for i in 1:4)==0.25)
@constraint(m, sum(p[2,i]*x[i] for i in 1:4) >= 0.35)
@constraint(m, sum(p[3,i]*x[i] for i in 1:4) >= 0.20)

@objective(m, Min, sum(c[i]*x[i] for i in 1:4))

status = solve(m)

println("Objective value: ", getobjectivevalue(m))
println("Compound 1: ", getvalue(x[1]))
println("Compound 2: ", getvalue(x[2]))
println("Compound 3: ", getvalue(x[3]))
println("Compound 4: ", getvalue(x[4]))
```

Numerical Output

```
Objective value: 14.761904761904763
Compound 1: 0.0
Compound 2: 0.0
Compound 3: 0.23809523809523822
Compound 4: 0.6666666666666665
```

Student quiz example

4. **Optional: Formulate the following optimal assignment task as a LP.** A group of four Universities in the South-West of England has to form a student team for a quiz show. The conditions of the show are as follows. One team member is to be selected from each University. The team member must be enrolled in one of the following single honours Degree Programmes: Mathematics, Physics, Chemistry, or Biology. No two team members are allowed to come from Degree Programmes in the same subject. Each Degree Programme in each University presents one candidate, and on the basis of a series of tests, each candidate is assigned an Erudition coefficient e_{ij}, where i marks the University and j marks the Degree Programme that the candidate comes from. The objective now is to maximise the overall strength of the team, i.e., the sum of the Erudition coefficients of chosen candidates.

   HINT: Introduce the unknowns $x_{ij} = \begin{cases} 1 & \text{if the corresponding candidate is chosen,} \\ 0 & \text{otherwise.} \end{cases}$

Maximize

$$\sum_{i,j} e_{ij} x_{ij}$$

subject to

$$\sum_{i} x_{ij} = 1$$

$$\sum_{j} x_{ij} = 1$$

where $i, j = 1, ..., 4$

Let's also solve the problem numerically for specific values to find out that the variables x_{ij} turn out to be integer without requiring them to be.

Numerical Example

```
using JuMP
using Clp

e = [.5 .3 .64 .12; .43 .66 .23 .33; .12 .77 .91 .12; .98 .33 .55 .34]

m = Model(solver=ClpSolver())

@variable(m, x[1:4,1:4]>=0)

@constraint(m, sum(x[i,1:4] for i=1:4).==1)
@constraint(m, sum(x[1:4,j] for j=1:4).==1)

@objective(m, Max, sum(e[i,j]*x[i,j] for i=1:4 for j=1:4))
```

```
status = solve(m)

println("Objective: ", getobjectivevalue(m))
println("Variables: ", getvalue(x))
```

Numerical Output

```
Objective: 2.7199999999999998
Variables: [0.0 0.0 1.0 0.0; 0.0 0.0 0.0 1.0; 0.0 1.0 0.0 0.0; 1.0 0.0 0.0 0.0]
```

Indeed, the variables are binary.

Toy company example

1. **Formulate as a LP, but do not try to solve:** A toy factory makes three types of wooden dolls: Grumpy, Sleepy, and Bashful. The manufacturing process uses three different machines A,B,C. Each Grumpy is processed for one hour each on machines A and C and three hours on B. Sleepy takes two hours on machine A and four hours on C. Bashful takes an hour on machine A and two hours on machine B. Because of maintenance schedules, machine A is available for 40 hours a week, B for 50 and C for 45. The profit per item is £4, £2, £5 respectively for the three types. How many of each type should the factory make per week in order to maximize the total profit?

$$
\begin{aligned}
&\text{Maximize} \\
&4x_1 + 2x_2 + 5x_3 \\
&\text{subject to} \\
&x_1 + 2x_2 + x_3 \leq 40 \\
&3x_1 + 2x_3 \leq 50 \\
&x_1 + 4x_2 \leq 45 \\
&x_1, x_2, x_3 \geq 0
\end{aligned}
$$

Numerical Example

```
using JuMP
using Clp
A = [1 2 3;3 0 2;1 4 0]
b = [40,50,45]
m=Model(solver = ClpSolver())
@variable(m,x[1:3]>=0)
@constraint(m,A*x.<=b)
@objective(m, Max, 4*x[1]+2*x[2]+5*x[3])
status = solve(m)
println("Objective value: ", getobjectivevalue(m))
println("x[1]: ", getvalue(x[1]))
println("x[2]: ", getvalue(x[2]))
println("x[3]: ", getvalue(x[3]))
```

Numerical Output

```
Objective value: 90.0
x[1]: 9.999999999999998
x[2]: 0.0
x[3]: 10.0
```

Suppose now there is an extra condition that exactly 360 toys are to be manufactured each week. Use this to reduce the number of variables, so the problem can be solved graphically. Do the graphs and show that such a problem is unfeasible: the feasible set is empty.

The reduced problem is

$$
\begin{aligned}
&\text{Maximize} \\
&1800 - x_1 - 3x_2 \\
&\text{subject to} \\
&x_2 \leq -320 \\
&x_1 - 2x_2 \leq -670 \\
&x_1 + 4x_2 \leq 45 \\
&x_1, x_2 \geq 0
\end{aligned}
$$

This does not need to be solved graphically, it can be solved algebraically. The first constraint gives that $x_2 \leq -320$ but also we have that $x_2 \geq 0$. So the feasible set is empty (since it is the product of the empty set with some other set for x_1).

Duality

We will show how to derive the dual from a primal through an example.

Primal

$$\begin{aligned}
&\text{Maximize}\\
&z = 5x_1 + 4x_2\\
&\text{subject to}\\
&x_1 \leq 4\\
&x_1 + 2x_2 \leq 10\\
&3x_1 + 2x_2 \leq 16\\
&x_1, x_2 \geq 0
\end{aligned}$$

Any feasible solution, say $x_1 = 4, x_2 = 2$, can be used to derive a lower bound: $z^* \geq 28$.
We now want to derive the best upper bound.
We multiply the constraints by scalars $y_1, y_2, y_3 \geq 0$ and add them together:

$$(y_1 + y_2 + 3y_3)x_1 + (2y_2 + 2y_3)x_2 \leq 4y_1 + 10y_2 + 16y_3$$

To derive an upper bound, we impose that

$$\begin{aligned}
&y_1 + y_2 + 3y_3 \geq 5,\\
&2y_2 + 2y_3 \geq 4
\end{aligned}$$

i.e. the coefficients of x_i's in the constraints dominate the corresponding coefficients in the objective function.
Then, to find the best upper bound, we form the dual problem.

Dual

$$\begin{aligned}
&\text{Minimize}\\
&w = 4y_1 + 10y_2 + 16y_3\\
&\text{subject to}\\
&y_1 + y_2 + 3y_3 \geq 5\\
&2y_2 + 2y_3 \geq 4\\
&y_1, y_2, y_3 \geq 0
\end{aligned}$$

Strong duality theorem

If z^* (the optimal objective value of the primal) is finite then so is w^* (the optimal objective value of the dual) and $z^* = w^*$.

Complementary slackness
Consider the primal and dual problems in canonical form.

Maximize	Minimize
$z = c^T x$	$w = b^T y$
subject to	subject to
$Ax \leq b$	$A^T y \geq c$
$x \geq 0$	$y \geq 0$

Complementary slackness theorem
If x is feasible in (P) and y is feasible in (D) then x is optimal in (P) **and** y is optimal in (D) iff $y^T(b - Ax) = 0$ **and** $x^T(A^T y - c) = 0$.

Proof
By strong duality we know that x is optimal in (P) and y is optimal in (D) iff $c^T x = b^T y$.
Moreover,

$$A^T y \geq c \Leftrightarrow y^T A \geq c^T \text{ and } x \geq 0$$
$$\Rightarrow c^T \leq y^T A \Rightarrow c^T x \leq y^T Ax$$

Also,

$$Ax \leq b \text{ and } y \geq 0 \Rightarrow y^T Ax \leq y^T b$$

So,

$$\Rightarrow c^T x \leq y^T Ax \leq y^T b = b^T y$$

Therefore,

$$c^T x = b^T y \Leftrightarrow c^T x = y^T Ax \text{ and } y^T Ax = y^T b$$
$$\Rightarrow y^T(b - Ax) = 0 \text{ and } (c^T - y^T A)x = 0 \Leftrightarrow x^T(A^T y - c) = 0$$

QED.

Farmer Giles example

Let's consider a simple example to illustrate the primal/dual problem formulations.

3. Farmer Giles has 100 acres of land, any part of which he can use to grow potatoes, or grow wheat, or raise free-range pigs. His annual profit per acre on potatoes is £300, and on wheat is £500. 50 pigs can be raised each year on each acre, and the profit on sale of one pig to a supermarket chain is £14. Growing potatoes requires 25 hours of labour per acre per year; growing wheat requires 50 hours of labour per acre per year, and raising pigs requires 75 hours of labour per acre per year. Giles' farm labourers are contracted for a maximum (all labourers, all activities) of 7000 hours of labour per year.

   Giles asks for an advise about what he should do next year, so as to maximize his profit.

   (a) Set up the linear programme Giles have to solve. Write down the dual problem.

   (b) Giles hires a consultant who tells him that he can either allocate 20 acres for wheat and 80 for pigs, and grow no potatoes, or allocate 10 acres for potatoes, 90 for pigs, and skip the wheat. Use the complimentary slackness theory to show that both alternative solutions are indeed optimal.

   In the following two parts assume that α and β are small positive numbers.

   (c) How much will Giles' profit increase if the total number of hours of labour that his labourers are contracted for next year increases by α hours?

   (d) What should Giles do, and by how much will his profit then increase or decrease if the supermarket chain raises or lowers the price it offers him for his pigs, in such a way that his profit per pig next year (I) increases or (II) decreases, by £β?

Primal

$$
\begin{aligned}
&\text{Maximize}\\
&300x_1 + 500x_2 + 700x_3\\
&\text{subject to}\\
&x_1 + x_2 + x_3 \leq 100\\
&25x_1 + 50x_2 + 75x_3 \leq 7000\\
&x_1, x_2, x_3 \geq 0
\end{aligned}
$$

Dual

$$
\begin{aligned}
&\text{Minimize}\\
&100y_1 + 7000y_2\\
&\text{subject to}\\
&y_1 + 25y_2 \geq 300\\
&y_1 + 50y_2 \geq 500\\
&y_1 + 75y_2 \geq 700\\
&y_1, y_2 \geq 0
\end{aligned}
$$

Using complementary slackness, we see that both alternative solutions satisfy the constraints as equalities and so are optimal.
If $\alpha = \beta = 0$, then

```
Objective value:66000.0
```

If $\alpha = 0.1$, then

```
Objective value:66000.8
```

If $\beta = 0.1$, then

```
Objective value:66009.0
x[1]: 10.0
x[2]: 0.0
x[3]: 90.0
```

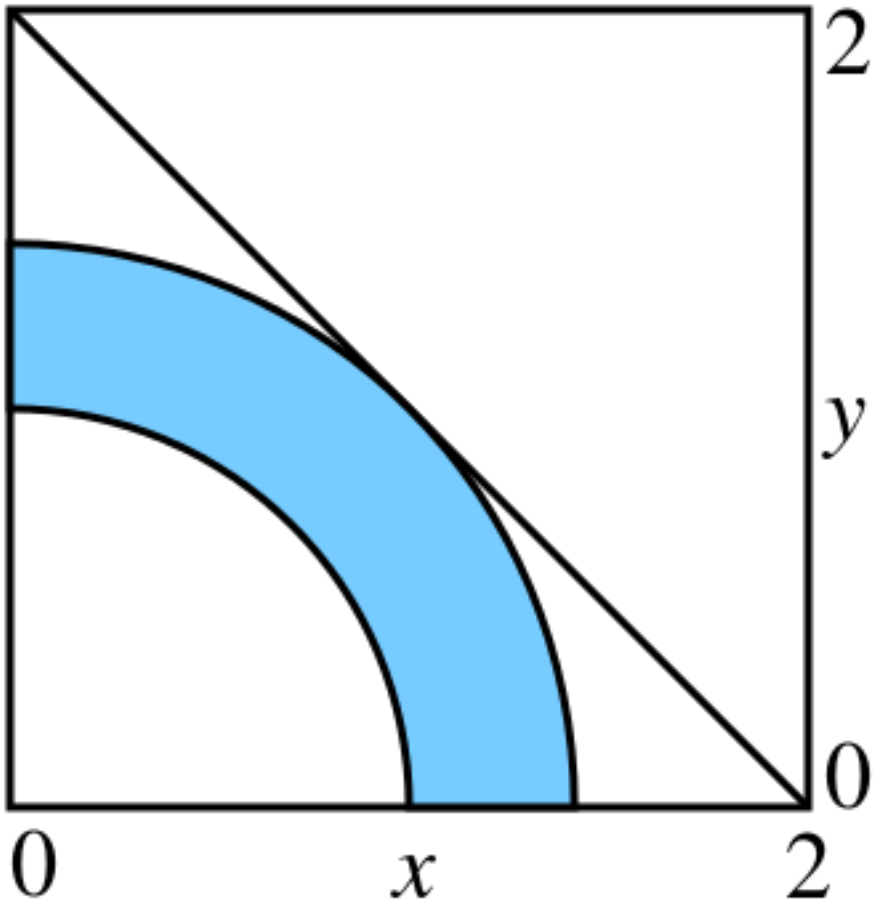
If $\beta = -0.1$, then

```
Objective value:65992.0
x[1]: 0.0
x[2]: 20.0
x[3]: 80.0
```

So we see that a small change in the profit for pigs changes which alternative the farmer should choose.

2 Nonlinear Programming

When one or all of the functions in an optimization problem are nonlinear, we have a problem of nonlinear programming.



As can be seen in the figure, the feasible region may not be a polyhedron. Still, the tangency of the feasible region with the contours of the objective function gives the optimizer.

[1.1] Consider the following nonlinear programming problem:

$$\begin{aligned} \text{Minimize} \quad & (x_1 - 4)^2 + (x_2 - 2)^2 \\ \text{subject to} \quad & 4x_1^2 + 9x_2^2 \le 36 \\ & x_1^2 + 4x_2 = 4 \end{aligned}$$

$$\mathbf{x} = (x_1, x_2) \in X \equiv \{\mathbf{x} : 2x_1 \ge -3\}.$$

Numerical Example

```
# Let's solve this problem numerically

using JuMP
using Ipopt

m = Model(solver=IpoptSolver())

@variable(m, x[1:2])
@constraint(m, 4*x[1]^2 + 9*x[2]^2 <= 36 )
@constraint(m, x[1]^2 + 4*x[2] == 4)
@constraint(m, 2*x[1] >= -3)
@objective(m, Min, (x[1]-4)^2 + (x[2]-2)^2)

status = solve(m)

println("Objective value: ", getobjectivevalue(m))
println("x[1] = ", getvalue(x[1]))
println("x[2] = ", getvalue(x[2]))
```

Numerical Output

```
Objective value: 7.999999999999975
x[1] = 1.9999999997851747
x[2] = 2.1483161958210293e-10
```