ProblemSet 1 – Optimization

George McNinch

2024-01-29

1. An optimization question in auto manufacturing

An automobile manufacturer makes a profit of $1,500 per unit on the sale of a certain car model. It is estimated that for every $100 of rebate, the number of units of this model sold in a given month will increase by 15%.

a. What amount of rebate will maximize the manufacturers profit for the month? Model the question as a single-variable optimization problem.

Let's set some quantities:

- r = amount of rebate to be given, in dollars.
- u_0 = units that will be sold, absent any rebate
- s = rebate benefit; i.e. relative change in sales per $100 of rebate
  0.15 (i.e. "15 %")
- $u = u(r)$ = units that will be sold with rebate r in place
- $p = p(r)$ = profit for the month with rebate r in place

Then the number of units sold in the month with rebate r is given by the expression

$$u = u(r) = u_0 \cdot \left(1 + \frac{0.15}{100} r\right) = u_0 \cdot \left(1 + \frac{s}{100} r\right)$$

and the month's profit is the product

`(profit per automobile)` × `(units sold)`;

i.e.

$$\begin{aligned} p = p(r) &= (1500 - r) \cdot u(r) \\ &= u_0 (1500 - r) \left(1 + \frac{0.15}{100} r\right) \\ &= u_0 (1500 - r) \left(1 + \frac{s}{100} r\right). \end{aligned}$$

Now, p is a quadratic function of r, and the coefficient of r^2 is *negative* (since s is positive). Thus the profit is maximized at the unique critical point of this function, which is found by solving the equation $\dfrac{dp}{dr} = 0$ for r. We must solve the equation:

$$\begin{aligned} 0 = \frac{dp}{dr} &= u_0 \frac{d}{dr} \left(-\frac{s}{100} r^2 + (1500 \frac{s}{100} - 1) r + 1500\right) \\ &= u_0 \left(\frac{-s}{50} r + 15s - 1\right) \end{aligned}$$

Thus we need $r = r^*$ where

$$r^* = \frac{15s - 1}{s/50} = \frac{50 \cdot (15s - 1)}{s}$$

When $s = .15$, find that the maximum profit is attained at $r^* = \$416.\overline{66}$.

As a function of an "unknown" rebate benefit s, the maximum profit is attained when $r^* = 50 \cdot 15 - \dfrac{50}{s}$.

When $s = .15$ the maximum profit is $p(416.66) \approx 1760.4$.

As a function of s, the maximum profit is

$$p\left(50 \cdot 15 - \frac{50}{s}\right) = (5625 \cdot s^2 + 750 \cdot s + 25)/s$$

b. Compute the *sensitivity* of your answer to the 15% assumption. Consider both the amount of rebate and the resulting profit.

Note that

$$\frac{dr^*}{ds} = u_0 \frac{d}{ds}\left(50 \cdot 15 - \frac{50}{s}\right) = u_0 \frac{50}{s^2}.$$

Thus the sensitivity of the refund is

$$S(r^*, s) = \frac{dr^*}{ds} \cdot \frac{s}{r^*} = \left.\frac{50}{s^2}\right|_{s=0.15} \cdot \frac{0.15}{416.66} = \frac{50}{0.15 \cdot 416.66} \approx 0.800$$

(observe that the factor of u_0 in $\dfrac{dr^*}{ds}$ cancels the corresponding reciprocal factor in $\dfrac{s}{r^*}$).

We've seen already that the maximum profit is given by

$$p(s) = (5625 \cdot s^2 + 750 \cdot s + 25)/s$$

so

$$\frac{dp}{ds} = 5625 - \left.\frac{25}{s^2}\right|_{s=0.15} = 4513.89$$

Thus

$$S(p, s) = \frac{dp}{ds} \cdot \frac{s}{p} = 4513.88 \cdot \frac{0.15}{1760.4} \approx .384$$

This means that a 1% change in the "rebate benefit" should cause a 0.8% change in the "optimal rebate" and roughly a 0.4% change in the monthly profits.

c. Suppose that rebates actually generate only a 10% increase in sales per \$100. What is the effect? What if the response is somewhere between 10% and 15% per \$100 of rebate?

$s = .10$ amounts to a -33% change in the "rebate benefit" from $s = .15$. So if instead s were only 10%, you'd expect the optimal rebate to be

$$r^* \approx 416.66 \cdot (1 - .33 \cdot .8) \approx 306.66$$

and the optimized profit to be

$$p \approx 1760.4 \cdot (1 - .33 \cdot .384) \approx 1537.32$$

d. Under what circumstances would an offer of a rebate cause a reduction in profit?

If the "optimal rebate" is ≤ 0, offer of a rebate causes a reduction in profit. Since the optimal rebate is given by the expression

$$r^* = \frac{15s - 1}{s/50} = \frac{50 \cdot (15s - 1)}{s}$$

it is non-positive precisely when $15s - 1 \leq 0$ i.e. when $s \leq \dfrac{1}{15} = 0.06$. So if the rebate benefit is less than 6%, we lose money when a rebate is given.

2. Computing yields with multi-variate optimization

A chemist is synthesizing a compound. In the last step, she must dissolve her reagents in a solution with a particular `pH` level H, for $1.2 \leq H \leq 2.7$, and heated to a temperature T (in degrees Celsius), for $66 \leq T \leq 98$. Her goal is to maximize her percent yield as a percentage of the initial mass of the reagents.

The equation determining the percentage $F(H,T)$ is

$$F(H,T) = -0.038 \cdot T^2 - 0.223 \cdot T \cdot H - 10.982 \cdot H^2 + 7.112 \cdot T + 60.912 \cdot H - 328.898.$$

1. Find the optimal temperature and `pH` level in the allowed range.

To find optimal values let us consider partial derivatives:

$$\frac{\partial F}{\partial H} = -0.223T - 21.964H + 60.912 = 0$$
$$\frac{\partial F}{\partial T} = -0.076T - 0.223H + 7.112 = 0$$

One can solve this using elementary methods to obtain:

$$T = 88.0651, H = 1.87914$$

2. Use `matplotlib` to produce a graph and a contour plot of the percentage of the powder function $F(H,T)$.

(To get a usable copy of your image, you can proceed in a few ways:

- if you produce the graph in colab you can right-click on the image and Save As a file on your file system.
- if you work in Python on your computer, you can save the image via a command like

```
> g.savefig("my graph image.png")
```

3. Blood typing

Human blood is generally classified in the "ABO" system, with four blood types: `A`, `B`, `O`, and `AB`. These four types reflect six gene pairs (genotypes), with blood type `A` corresponding to gene pairs `AA` and `AO`, blood type `B` corresponding to gene pairs `BB` and `BO`, blood type `O` corresponding to gene pair `OO`, and blood type `AB` corresponding to gene pair `AB`. Let p be the proportion of gene `A` in the population, let q be the proportion of gene `B` in the population, and let r be the proportion of gene `O` in the population. Observe that $p + q + r = 1$.

a. The Hardy-Weinberg principle implies that:

> ($\clubsuit$) The quantities p, q, and r remain constant from generation to generation, as do the frequencies of occurrence of the different genotypes `AA`, `AO`, … .

Assuming the validity of ($\clubsuit$), what is the probability that an individual has genotype `AA`? `BB`? `OO`? What is the probability of an individual having two different genes? Express your response using the quantities p, q and r.

Answer: Since the probability of a single gene being `A` is p, the probability that an individual has 2 `A` genes, as required for the `AA` genotype, is just p^2. Similarly, the probability of genotype `BB` is q^2, and the probability of genotype `OO` is r^2. The probability of an individual having two different genes is, simply, the probability that they are of none of genotypes `AA`, `BB`, or `OO`; this is

$$(\clubsuit) \quad (1 - p^2 - q^2 - r^2.$$

Alternatively, notice that the genotypes should be viewed as *ordered pairs* - i.e. `AB` is distinct from `BA`. So the probability of having two different genes is the probability of having genotype given by one of the ordered pairs `AB`, `BA`, `AO`, `OA`, `BO`, or `OB`. Thus that probability is

$$(\diamondsuit) \quad pq + qp + pr + rp + qr + rq = 2pq + 2pr + 2qr$$

Notice that

$$1 = p + q + r \implies 1 = (p + q + r)^2 = p^2 + q^2 + r^2 + 2pq + 2pr + 2pq$$

which confirms that $(\clubsuit) = (\diamondsuit)$ coincide.

b. Still assuming the valiidty of $(\clubsuit)$, find the maximum percentage of the population that can have two different genes. Perform this computation in two different ways:

- directly maximize a function of only two variables
- use the method of Lagrange multipliers.

Answer: This question asks us to consider the maximum possible value of $f(p, q, r) = 1 - p^2 - q^2 - r^2$, subject to the constraint that $p + q + r = 1$. For the first approach, we write $r = 1 - p - q$, to obtain $g(p, q) = 1 - p^2 - q^2 - (1 - p - q)^2$ as a function of two variables that we can maximize directly. The second approach will use the constraint $h(p, q, r) = p + q + r = 1$.

For the first approach, we compute the partial derivates of $g(p, q)$:

$$\frac{\partial g}{\partial p} = -2p - 2(1 - p - q)(-1) = 2 - 4p - 2q$$
$$\frac{\partial g}{\partial q} = -2q - 2(1 - p - q)(-1) = 2 - 2p - 4q$$

Thus

$$\mathbf{0} = \nabla g = (2 - 4p - 2q)\mathbf{i} + (2 - 2p - 4q)\mathbf{j} \implies p = q = \frac{1}{3}.$$

Then also $r = \frac{1}{3}$ as one sees using the formula $p + q + r = 1$.

The problem is defined for $0 \leq p, r, q \leq 1$; since $p + q + r = 1$, the "boundary values" occur when $\begin{bmatrix} p \\ q \\ r \end{bmatrix}$ is equal to **i**, **j**, or **k**.

Evaluating the function g at these boundary values and at $\frac{1}{3} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, one sees that the maximum value is

$$g(1/3, 1/3, 1/3) = 2(1/9 + 1/9 + 1/9) = 2/3$$

.

For the second approach, we note that

$$\nabla f = \frac{\partial f}{\partial p}\mathbf{i} + \frac{\partial f}{\partial q}\mathbf{j} + \frac{\partial f}{\partial r}\mathbf{k} = -2(p\mathbf{i} + q\mathbf{j} + r\mathbf{k})$$

while

$$\nabla h = \mathbf{i} + \mathbf{j} + \mathbf{k}$$

Thus

$$\nabla f = \lambda \nabla h \implies -2p = -2q = -2r = \lambda.$$

Thus, $-2(p + q + r) = -2p - 2q - 2r = 3\lambda$; now using the constraint $p + q + r = 1$ we see that $\lambda = \frac{-2}{3}$ so that $p = q = r = \frac{1}{3}$ determine the optimal values.

c. Explain in words what the Lagrange multiplier represents in the second computation of part (b).

Thinking about it as a "shadow price", this would represent how much the probability of having two different genes would change if the total probability of having any one gene changes. Of course this isn't really possible, because the sum of probabilities is always equal to 1. But it would say that if the probability of having an `A`, `B`, or `O` gene went down (maybe because there was a new gene discovered…), then the likelihood of having two distinct genes would go up!

4. Newton's method and root finding

a. microprocessors

One of the uses of Newton's method is in implementing division on microprocessors, where only addition and multiplication are available as primitive operations. To compute $x = a/b$, first the root of $f(x) = 1/x - b$ is found using Newton's method, then the fraction is computed with one last multiplication by a.

Find the Newton iteration needed to solve $f(x) = 0$ and explain why it is well-suited to this purpose. (**Note**: We are trying to approximate division, so we shouldn't actually use division functions implemented in `python`...)

Given an initial guess x_0 for the reciprocal of b, Newton's method refines the guess by the rule

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

More generally, given if you've already found the nth refined guess x_n, you obtain the next refinement by the rule

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Note that $f'(x) = \dfrac{-1}{x^2}$. Now, let's ofind a simpler expression for x_{n+1}:

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{\dfrac{1}{x_n} - b}{\dfrac{-1}{x_n^2}} = x_n + x_n - bx_n^2 \\ &= 2x_n - bx_n^2 \end{aligned}$$

Thus, computing x_{n+1} from x_n involves a few multiplications and a subtraction operation (but no division!).

Here is some `python` code that implements this strategy:

```
def newtonStep(x,b):
    return 2*x - b*x*x

def approx(b,x0,num):
    x = x0*1.0
    for i in range(num):
        xnew = newtonStep(x,b*1.0)
        x = xnew
    return x
```

b. experiments

Apply Newton's Method to compute $1/b$, where b is: (i) the last 3 digits of your student number; and (ii) the area code of your phone number. For these experiments, report the number of iterations required for the approximation to be consistent to 10 digits.

Let's use this strategy to estimate $1/4242$.

```
from pprint import pprint

results = [ f"{i} - {approx(4242,.0001,i)}"
            for i in range(10)]

pprint(results)
```

```
=>
['0 - 0.0001',
 '1 - 0.00015758',
 '2 - 0.0002098249619512',
 '3 - 0.00023288944872391684',
 '4 - 0.0002357034422709971',
 '5 - 0.000235737854475393',
 '6 - 0.00023573785950023563',
 '7 - 0.00023573785950023574',
 '8 - 0.00023573785950023574',
 '9 - 0.00023573785950023574']
```

so the approximate is consistent to 10 digits after 5 iterations.