ENTROPY EXERCISE

A monopolist has demand curve $q_t = c_t - bp_t$ and constant costs γ per unit of product. Each period, there is a new realization of c_t, which is always either 10 or 15, with probability .5, i.i.d. across t. With no information cost, the monopolist would of course set $p_t = \frac{1}{2}(c_t/b + \gamma)$. But suppose there is a cost of processing the information in the sequence of c_t random variables, so that the monopolist's profits are

$$(p_t - \gamma)(c_t - bp_t) - \theta\big(H(C) - H(C \mid P)\big)\,,$$

where $H(C)$ is the unconditional entropy of c_t and $H(C \mid P)$ is the expected entropy of the distribution of $C \mid p_t$. The expression $H(C) - H(C \mid P)$ is the **mutual information** between the monopolist's choice p_t and the unobserved state c_t. The monopolist can be thought of as choosing a joint distribution for p_t and c_t. Obviously with $\theta = 0$ the joint distribution will be degenerate, with $p_t \equiv \frac{1}{2}(c_t/b + \gamma)$, but more generally it will not be.

Suppose $\gamma = 2, b = 0.8$.

(1) Find the optimal joint distribution of p_t and c_t as a function of θ. You can assume (though it can be proved) that the distribution will have just four p_t, c_t pairs as points of support.

(2) Show how expected profits vary as a function of θ. The rate of information flow between c_t and p_t per period is their mutual information, $H(C) - H(C \mid A)$. Show how that varies as a function of θ.

(3) If the monopolist, instead of having a fixed cost per bit of information flow has a fixed channel capacity (i.e. an upper bound on the information flow rate), how does the solution vary with capacity? Note that the fixed-capacity and fixed-unit-cost solutions are dual, so you don't need to re-solve the model to answer this. The interesting thing is whether there are capacities at which the shadow price of information is zero and/or costs of information at which no information is processed.

Notes: This will require numerical optimization over the prices chosen and the probabilities — four free parameters. In computing entropy, terms of the form $x \log x$ occur. While $x \log(x) \to 0$ as $x \to 0$, asking the computer to find $x \log x$ with $x = 0$ causes problems, so you need to handle that case.

In numerical optimization, a gradient-based method will try values for probabilities ouside of $(0, 1)$. A program passed to the optimizer to evaluate the firm's objective here will have to check for probabilities outside $(0, 1)$ and return a very small number (if it is maximizing, otherwise very large).

You can use any software you like to do this. A program that evaluates the objective function, written in R, is in the directory with the exercise. However, in order to get you to understand the code if you use it, it is hard-coded with parameter values for c_t, b and γ that you will have to change to solve the problem as posed.

Date: October 22, 2020.

The R package **optimize.1** is also int the exercise directory. You can install it as an R source package (**install.packages**("DirectoryWhereItIs/optimize.1", repos=NULL)), then load it with **library**("optimize.1"). It contains the gradient-based optimizer csminwelNew(), which will work on this problem, though again you are welcome to use other software.

Answer:

Giorgios was right, the probability of the action matching the true state is the same for both states at every value for the cost of information θ, and also that this is because of the quadratic objective. If the demand curve were constant-elasticity, for example, it would no longer be true.

Here is the table of the results I found.

	Low price	High price	pLL	pHH	Obj	eU	info	theta
1	8.8126	8.8124	0.5000	0.5000	-37.1281	37.1281	0.0000	4.0000
2	8.2036	9.4214	0.6948	0.6948	-37.1362	37.4246	0.0780	3.7000
3	7.8899	9.7351	0.7952	0.7952	-37.1760	37.8091	0.1862	3.4000
4	7.7243	9.9095	0.8540	0.8525	-37.2469	38.1029	0.2761	3.1000
5	7.5438	10.0812	0.9060	0.9060	-37.3478	38.4158	0.3814	2.8000
6	7.4379	10.1871	0.9399	0.9399	-37.4752	38.6397	0.4658	2.5000
7	7.3614	10.2636	0.9644	0.9644	-37.6262	38.8127	0.5393	2.2000
8	7.3086	10.3164	0.9813	0.9813	-37.7975	38.9375	0.6000	1.9000
9	7.2754	10.3496	0.9919	0.9919	-37.9848	39.0182	0.6459	1.6000
10	7.2579	10.3671	0.9975	0.9975	-38.1834	39.0616	0.6755	1.3000

Here pLL is the probability of the low price conditional on low demand and pHH is the probability of the high price given high demand. eU is expected utility, info is mutual information between p and c, and theta is θ.

The problems those of you who used csminwelNew() had may have arisen from picking the wrong scale for H0, the initial guess of the inverse Hessian. Usually this has to be quite small. I used **diag**(4) *1e-4. If it's too big, the algorithm at first takes very large steps that may repeatedly send it outside the bounds imposed by probabilities between zero and one.

It's also true that your values for θ have to be in a certain range to get sensible results. If information costs go much below the 1.3 shown in the table, the solution is almost on the boundary of its domain: probabilities go to one and zero. Gradient based optimizers don't work well at such boundaries if the boundaries are enforced (as in my sample code) by jumping discontinuously to a bad function value when the choice variable goes out of bounds.

So long as $\theta > 0$, the solution is never actually at the boundary, even though for practical purposes it may be. The entropy of a two-point distribution is $-p\log p - (1-p)\log(1-p)$. Its derivative with respect to p is $\log\big((1-p)/p\big)$. This goes to $-\infty$ as $p \to 1$ and to $+\infty$ as $p \to 0$. So if we actually have perfect information with $p = 1$, we reduce our information costs at an initially infinite rate as we reduce p below one. Similarly we reduce our costs at an initially infinite rate as we raise p above zero. So with positive θ, it's never optimal to completely resolve uncertainty. But as in this example, solutions in which *almost* no uncertainty remains yet $\theta > 0$ are not uncommon.

Giorgios was also right that the problem can be simplified by recognizing that, given the probabilities of the two possible values of c, the optimal value of p is determined and can be found analytically. This greatly reduces the burden on the computer, which would be important in a

higher-dimensional variant of this problem. It might also make the solution numerically less sensitive, though the equations might still misbehave at the boundaries. The code I provided, and that was used for the table above, lets the computer do the algebra.