From JPEGs to Quantum Field Theory

Solution set for the Santa Fe Institute Renormalization MOOC

http://renorm.complexityexplorer.org

Problem set at http://santafe.edu/~simon/MOOC_problems.pdf

Simon DeDeo
Assistant Professor, Carnegie Mellon University; External faculty, Santa Fe Institute.
simon@santafe.edu; http://santafe.edu/~simon

1 Unit One: An Introduction to Coarse-Graining

Q1: what is the entropy of the distribution?

Answer: 4.18 bits (using Shannon's formula). You can do this by hand if you're really dedicated, or it's a few lines of Python:

```
>>> import math
>>> set=[["a", 0.0777093], ["b", 0.01694125], ["c", 0.02509587], ["d", 0.0415729],
   ["e", 0.1293273], ["f", 0.02237207], ["g", 0.01869932], ["h", 0.063514],
   ["i", 0.0705279], ["j", 0.00162383], ["k", 0.00598080], ["l", 0.04025481],
   ["m", 0.0275251], ["n", 0.0702613], ["o", 0.0746462], ["p", 0.01533419],
   ["q", 0.001168940], ["r", 0.0602125], ["s", 0.0617264], ["t", 0.0869565],
   ["u", 0.02793712], ["v", 0.01067520], ["w", 0.0229406], ["x", 0.001564180],
   ["y", 0.02368643], ["z", 0.001745021]]
>>> sum([-i[1]*math.log(i[1], 2) for i in set])
4.181206228
```

Where `math.log(x, 2)` is the way to get Python to compute the base-two logarithm of x.

If these manipulations feel unfamiliar to you, and you're keen to see the deeper connections between renormalization, coarse-graining, and information theory, take a look at the guide "Information Theory for Intelligent People", http://santafe.edu/~simon/it.pdf; or, even better, sign up for Seth Lloyd's Information Theory course.

Q2: Coarse-grain this distribution to "vowel" vs. "not-vowel". What is the entropy of this coarse-grained distribution? For simplicity, assume the vowels are just the letters a, e, i, o and u–neglect letters that can sometimes function as a vowel, like "y", or cases where a string of letters can indicate a vowel sound, as in "loose".

Answer: I find the probability of a, e, i, o and u is roughly 0.38; the entropy of the distribution $[0.38, 1-0.38]$ is 0.96 bits (or so). This coarse-graining has discarded 3.22 bits of information.

Q3: What are some of the rules that might apply to this coarse-grained timeseries?

Answer: Many possible answers! For example, you'd expect one of the rules is that you never get two DETERMINERs in a row (i.e., you never see something like "the the"). A more probabilistic law, or rule, is that you expect VERB and ADVERB symbols to be close to each other,

as well as NOUN and ADJECTIVE. You might expect that "DETERMINER ADJECTIVE" and "DETERMINER NOUN" are more common than "DETERMINER ADVERB". And so on.

Q4: the final question suggested an open problem—to grab some real-world data from online, and play with different kinds of coarse-grainings. I've taken the suggestion to look at Chess; see `http://santafe.edu/~simon/chess.py` for Python code that grabs a set of historical games by Gary Kasparov in PGN format, and coarse-grains them into timeseries by point-values within each quadrant of the 8×8 board. I find, for example, that coarse-graining by piece value leads to a per-symbol entropy of 6.37 bits, while preserving coarse-grained spatial information required 13.19 bits, about twice as much.

If you want to progress even further, you might think about how to directly recover rules referring to the dynamics of these coarse-grained representations. For example, you might consider trying to directly infer the Markov chain that best describes a chess game. The Markov Chain fitting code `SFIHMM` is available at `http://bit.ly/sfihmm`, and with a little work, and perhaps further coarse-graining, you can try to model a Kasparov game using a system very similar to that we'll describe in the next section. Drop me a line if you do!

2 Renormalizing Markov Chains

Q1: Prove the existence of two-state slippy counters that do not have a corresponding two-state microtheory.

Answer: Let's try to find a two-state microtheory! An arbitrary two-state Markov chain can be written as

$$\begin{pmatrix} P(1|1) & P(1|2) \\ P(2|1) & P(2|2) \end{pmatrix} = \begin{pmatrix} p & q \\ 1-p & 1-q \end{pmatrix}, \tag{1}$$

and we just need to find p and q such that if you take two steps into the future, *i.e.*, if you compute

$$\begin{pmatrix} p & q \\ 1-p & 1-q \end{pmatrix}^2 = \begin{pmatrix} p^2+(1-p)q & pq+(1-q)q \\ p(1-p)+(1-p)(1-q) & (1-q)^2+(1-p)q \end{pmatrix}, \tag{2}$$

this is equal to the slippy counter,

$$\begin{pmatrix} \epsilon & 1-\epsilon \\ 1-\epsilon & \epsilon \end{pmatrix}. \tag{3}$$

However, if we attempt to solve the simultaneous equations involved, say

$$\begin{aligned} p^2+(1-p)q &= \epsilon \\ pq+(1-q)q &= 1-\epsilon, \end{aligned}$$

we find that it requires (for example) that

$$p = \frac{1}{2}\left(1+\sqrt{2\epsilon-1}\right), \tag{4}$$

which can't be satisfied (for real-valued probabilities!) when ϵ is less than $1/2$. No microtheory, in other words, can reproduce a "sufficiently reliable" slippy counter at twice the frequency. By the way, this is something many people don't know—sometimes it's thought of as "taking the square root of a transition matrix".

Occasionally, some applied mathematicians (or, more often, microeconomists working with Markov chains) rediscover this strange fact that you can always square a transition matrix to get another one, but you can't necessarily find a transition matrix that you can square to get the one you have.

Q2: Experiment with the three-state case. Take a Markov Chain over three states, represent it as a matrix, and use a system like Mathematica or Python to find the fixed points by repeated multiplication of the matrix on itself. Confirm the result using theory: find the eigenvector corresponding to the eigenvalue equal to unity, and show that the columns of the fixed point Markov chain are equal to the components of the first eigenvector.

Answer: I've written this up as a Mathematica notebook, to give you a little experience setting up these kinds of problems: `http://santafe.edu/~simon/markov.nb`

Q3: Consider the two rock-paper-scissors playing robots...

Answer: I've written this up as a Python file, to give you a little experience setting up these kinds of problems: `http://santafe.edu/~simon/robot.py`

3 Renormalizing Cellular Automata

Q1: Demonstrate the renormalization of Rule 105 to Rule 150.

Answer: I've written this up as a Python file, to give you a little experience setting up these kinds of problems: `http://santafe.edu/~simon/ca.py`. In addition to the "edge-detector" projection, I find five other coarse-grainings that work.

4 Renormalizing the Creature (Krohn-Rhodes Theorem)

Q1: Find a coarse-graining of Z_6 with two states. Hint: you might find it helpful to draw the transitions in a 2×3 grid.

Answer: Map all the odd-numbered states to a coarse-grained state "A"; all the even-numbered states to coarse-grained state "B". The operator `C` now takes you from A to B, and from B to A. (In words: adding one takes odd to even; even to odd.) If you drew the 2×3 grid, the first column is state A; the second column is state B. Notice how all the arrows cross the boundary of either coarse-grained state.

Q2: Find a coarse-graining of Z_{36} with 3 states.

Answer: Map the states that multiples of three into A, multiples of three, plus one into B, and multiples of three, plus two, into C. (More formally, map the state i into i mod 3.

Q3: Explain why Z_{13} doesn't have a (deterministic) coarse-graining.

Answer: It's prime! Prime numbered groups are so-called "simple" groups, and are some of the basic atoms of the Krohn-Rhodes decomposition.

5 Renormalizing the Thermal Plasma

Q1: Plot the effective electric charge at some fixed distance r, as a function of temperature, and plasma density. How do these effects compete?

Answer: The field generated by a test particle with ("unrenormalized") charge q in a thermal plasma is

$$\phi(r) = \frac{q}{4\pi\epsilon_0 r} e^{-\sqrt{2}r/\lambda_D}, \tag{5}$$

where λ_D, just a way to name a collection of terms that pop out when you do the calculation to derive this form, is the "Debye length", and equal to

$$\lambda_D = \sqrt{\frac{\epsilon_0 T}{n_0 e^2}}. \tag{6}$$

You can think of λ_D as the length scale at which the charge starts to disappear. When you look at a plasma on those coarse-grained scales it appears as if charges start to lose the powers they had at shorter distances—a loss of power over and above what you expect from the standard $1/r$ scaling.

Three things pop out when you look at λ_D. First, it scales as the square root of temperature. First, the hotter the plasma, the longer the Debye length—meaning, the screening of charge, caused by the rearrangement of the other charges in the plasma, is less effective. You can understand this by going back to the original Boltzmann distribution for (let's say) the electrons,

$$n_e(\vec{x}) = n_0 e^{e\phi(x)/kT}. \tag{7}$$

As T gets larger and larger, the local density of electrons becomes less and less sensitive to the local potential field. Electrons don't rearrange themselves as much in response to penalties coming from electrostatics.

Second, the higher the density, the shorter the Debye length. In other words, higher density plasmas are better at screening charges. One way to understand this is to notice that the thermal effects multiply the density in Eq. 7—if you double the density, the same field and temperature combination will move twice as much charge. Each charge

6 Rate-Distortion Theory: Keeping the Things that Matter

Q1: Write down a distortion matrix that leads you to prefer the two-state model. Write down a distortion matrix that leads you to prefer the original three-state coarse-graining. In this case, don't worry about the mutual information term; just find a distortion matrix that makes one preferable to the other as a representation of the system.

Answer: Recall that the two-state model distinguishes even from odd, while the three-state model distinguishes numbers "modulo three" (i.e., whether they are divisible by three, and if not, the remainder. If we write the "true" states as the rows, then here's an example of a distortion matrix

that basically says "try to get the right answer, but if you can't, it's much worse to confuse even and odd":

Truth	Belief					
	1	2	3	4	5	6
1	0	+10	+2	+10	+2	+10
2	+10	0	+10	+2	+10	+2
3	+2	+10	0	+10	+2	+10
4	+10	+2	+10	0	+10	+2
5	+2	+10	+2	+10	0	+10
6	+10	+2	+10	+2	+10	0

(8)

If someone modeling the creature used the odd-even counter, then he would never get more than a +2 penalty; indeed, if the creature tended to occupy each of its six internal states equally, and the "operator" used his coarse-grained model to figure out even vs odd and then guessed one of the three possibilities randomly, than then 1/3rd of the time, he would get it right, for an average penalty of 4/3.

What kind of distortion function fits well with the three-state counter? Let's imagine a world where it's the state of the creature "mod three" that matters most.

Truth	Belief					
	1	2	3	4	5	6
1	0	+10	+10	+2	+10	+10
2	+10	0	+10	+10	+2	+10
3	+10	+10	0	+10	+10	+2
4	+2	+10	+10	0	+10	+10
5	+10	+2	+10	+10	0	+10
6	+10	+10	+2	+10	+10	0

(9)

If someone modeling the creature used the three-state counter, then he would never do worse that +2, and half the time he'd get the right answer, for an average penalty of +1. Meanwhile if someone used the odd-even counter, and guessed randomly, then when the state was 1 (for example), he'd guess 3 or 5 two-thirds of the time, leading to an average penalty of 20/3.

The big missing piece here is, of course, the costs of tracking the creature with larger and more complex models.