Math 486, Spring 2019. Project

due: Phase I (pick topic, collect data): March 29

Phase II (preliminary output): April 18

Phase III (presentation and report): April 30, report due by May 6

- Based on real data (self-collected or otherwise) or maybe simulated data
- Applying course concepts
- Submit a report (3-4 pages) and make a short in-class presentation

---

Possible topics:

1. Markov Entropy using a text.
   Variation: Markov Entropy using genome data.
2. Hidden Markov Models: estimation
3. Poisson process (e.g. based on earthquake data)
   : homogeneous?
   : if not, estimate rate (e.g. using kernel smoothers)
4. Testing random number generators /chaotic maps using Poisson distribution.
   Variation: Spatial Poisson: real data.
5. Queueing systems: simulation or observation (e.g. checkout lanes)

---

1. Markov Chains in text files.

(Requires programming skills)
Taking a sequence of characters (as, for example, natural language texts, or computer files) we might naturally ask a question: what amount of information is contained in these texts?

Taking the semantic issues aside, let's concentrate on the simplest possible structure:

a) When the characters are independent
b) When the characters in the text follow a Markov Chain.

(a) Entropy for independent characters.

One important measure of information is entropy. For independent, identically distributed (i.i.d) characters, let $P(X_i = k) = p_k$. Then entropy is defined as

$$H_0 = \sum_k -p_k \log_2(p_k)$$

We will call this *zero-order entropy* or simply "entropy".
Entropy can be defined in computer science terms as number of bits needed to represent a text made of these characters. For example, if there are 8 equally likely characters, then $H = 3$. That is, one can represent each character as a sequence of bits (binary digits), 000, 001, ... through 111 (8 combinations).

If, on the other hand, char A appears 50% of the time, and B and C each appear 25% of the time, then $H = 1 * 0.5 + 2 * 2 * 0.25 = 1.5$, meaning that only 1.5 bits per char are needed. How is that possible?

Let us represent A with 1, B with 01 and C with 00. Then any "words" formed from these can be decoded from a sequence of only 0's and 1's, without breaks! (For example, try to decode 101100101.)[1]

You can see that expected number of bits needed to encode such "text" is exactly 1.5. If all the chars were equally likely[2], we would obtain $H = \log_2 3 = 1.585$. We can save space (although not much). This is a principle on which file-compression algorithms are based.

Conversely, if you take a file consisting of well-compressed material (e.g. zipped or jpeg files), then the characters in it will appear fairly random.

Project, part I:
Compute entropy for English-language texts. Spaces, punctuation marks etc will count too! You might also try jpeg files, or other languages, or ... (?)
Then find the degree of compression for your file using `pkzip`, for example. The latter will likely achieve much better compression ratio than zero-order entropy!

---

[1] In general, these are called Huffman codes, and they are obtained by "harvesting" the leaves of a binary tree. Some other codes, e.g. Morse code, take advantage of the unequal frequencies of chars, but Morse code uses breaks between letters, taking up valuable space.

[2] Also, remember the second law of thermodynamics, stating that the system tends to the maximum entropy state. Maximum entropy is achieved when all the chars are equally likely.

(b) Markov-chain based entropy

Let us now treat our text as a Markov chain in which each character is predicted based on the previous char. Based on a sample (long enough) text, we can estimate the transition probabilities from char x to char y as

$$p(y|x) = \frac{\#\{x \text{ is followed by } y\}}{\#\{x \text{ appears}\}}$$

Then define our *first-order entropy* as

$$H_1 = \sum_x \sum_y -\log_2[p(y|x)]\ p(y|x)\, p_x$$

(*Exercise:* when characters appear independently, that is, $p(y|x)$ does not depend on x, then $H_1 = H_0$.)
The actual coding can be performed as follows: depending on your current char, look at the row of the transition matrix and code the next character using this row as your probability distribution[3]. Thus, the interpretation of the code depends on what your current char is!

This is likely to work better, since it captures more of the natural text structure (no longer a sequence of random chars). On the other hand, you will have to keep a longer code table[4].

Project, part II:
Compute the first-order entropy for your text. For English-language texts, it would be interesting to present (in part) the transition table that you have estimated.

(b*) We can extend this idea, requiring a char to depend on k of its predecessors. We can model this as a Markov chain with states being k-tuples of chars. This will, of course, greatly increase the number of states, but, on the other hand, capture more and more of the text structure!

Extra Credit: Compute second- and third- order entropy (if possible) of texts. Compare with commercial compression products and graph H as a function of order.

[3] You will not have to find the actual substitution table.

[4] Another problem with this coding is that if one bit is corrupted then the rest of the text will appear as gibberish. But that's another issue, that of error-correcting codes.

3. Poisson process - real data

These can be anything. Some examples:

- arrivals at a web site
- cars on a freeway
- customers in a store
- Spatial processes

There are two issues:

a) is it a Poisson process at all, that is, are increments independent? To test this, one can use chi-square test for homegeneity/independence:
Split the time domain into intervals A_1, A_2, ..., A_N. Set up a two-way table with rows corresponding to the number of events $N(A_t)$ in A_t, $t = 1, ..., N-1$, and columns corresponding to the number of events in A_{t+1}. In the cell (i, j) of the table put counts of how many times pairs $N(A_t) = i$, $N(A_{t+1}) = j$ were observed.

b) is the rate of arrivals constant? You can test for it using inter-arrival times (they are supposed to be Exponential) and/or estimate the rate in case it's variable. (Testing whether a distribution is Exponential can be done using Kolmogorov-Smirnov test, talk to the instructor for references.)

4. Testing for randomness

Uses chi-square test for goodness of fit to Poisson distribution. Split the area into equal-sized sets and count the frequencies of occurrence. For complete randomness, the counts should resemble Poisson distribution.