Typicality

Information Theory (APMA 1710), Fall 2011

In these exercises, you will visualize and empirically verify the results proved in the theorems on typical sequences and jointly typical sequences. We will investigate the particular case of Bernoulli sequences.

(Suggestion: For this assignment, Matlab is probably the easiest language to use.)

(1) Typical sequences

Suppose $X_1, \ldots, X_n \sim$Bernoulli(α) are i.i.d. with $\alpha \in [0,1]$ (i.e. $X_i = 1$ with probability α and $X_i = 0$ with probability $1-\alpha$.) Recall that the (n, ε) typical set is

$$A_\varepsilon^{(n)} = \left\{ x_{1:n} \in \{0,1\}^n : \left| H(X) - \frac{1}{n} \log_2 \frac{1}{p(x_{1:n})} \right| \leq \varepsilon \right\},$$

where $X \sim$Bernoulli(α). Recall that the theorem on typical sequences says that for any $\varepsilon > 0$,

$$\mathbb{P}(X_{1:n} \in A_\varepsilon^{(n)}) \to 1$$

as $n \to \infty$, and

$$|A_\varepsilon^{(n)}| \approx 2^{nH(X)}$$

when n is large.

(a) (Sampling)

Write code to sample m such i.i.d. Bernoulli(α) sequences:

$$x_1^{(1)}, \ldots, x_n^{(1)},$$

$$x_1^{(2)}, \ldots, x_n^{(2)},$$

$$\ldots$$

$$x_1^{(m)}, \ldots, x_n^{(m)},$$

compute $k^{(j)} = \sum_{i=1}^n x_i^{(j)}$ for $j = 1, \ldots, m$, and plot a histogram of the values $k^{(1)}, \ldots, k^{(m)}$. (Suggestion: In Matlab, the `hist` command creates a histogram for you - for example, `hist(k_values,25)` works well.)

(b) (Visualize the typical set)

Since $p(x_{1:n})$ depends only on $k = \sum_i x_i$ (and not on the particular values of $x_1, \ldots, x_n$), it is possible to determine whether a sequence is typical based on its value of k. Reformulate the condition for typicality

$$\left| H(X) - \frac{1}{n} \log_2 \frac{1}{p(x_{1:n})} \right| \leq \varepsilon$$

in terms of a pair of inequalities for k. (In other words, determine the range of typical k values.) Modify your code to draw these lower and upper thresholds on each histogram plot. (Suggestion: If you're using Matlab, the command `line([k,k],[0,max_count],'Color','r','LineWidth',3)` works well.)

Run your code for $n = 100, 500, 1000, 5000$ using the following parameter values: $\alpha = 0.3$, $\varepsilon = 0.02$, $m = 10000$. Explain what you see. Does it make sense? Why? (Include your histogram plots when you submit your homework.)

(c) (Estimate the probability of the typical set)

We can use our samples from (a) to estimate the probability of the typical set, by computing the fraction of samples that were in the typical set. Since the probabilities $p(x_{1:n})$ involved will be very small, it may be difficult to represent them on the computer, so you will want to check whether a sequence is typical by checking whether its value of k is typical (rather than comparing $\frac{1}{n} \log_2 \frac{1}{p(x_{1:n})}$ to $H(X)$). Write code to compute the following estimate of $\mathbb{P}(X_{1:n} \in A_\varepsilon^{(n)})$:

$$\hat{p} = \frac{1}{m} \sum_{j=1}^{m} I(x_{1:n}^{(j)} \in A_\varepsilon^{(n)}),$$

where I denotes the indicator function (i.e. the jth term is 1 if the jth sequence is typical, and 0 otherwise.) (By the way, this is called a "Monte Carlo" estimate.)

Compute this estimate using the same values of n as in (b). (Use the same values for the other parameters as well.) Do your empirical results agree with the theoretical predictions?

(d) (Estimate the size of the typical set)

We can also use our samples to estimate the size of the typical set. To do this, we will use a powerful technique from statistics called "importance sampling". Importance sampling can be used to derive the following estimate of $|A_\varepsilon^{(n)}|$:

$$\hat{r} = \frac{1}{m} \sum_{j=1}^{m} \frac{1}{p(x_{1:n}^{(j)})} I(x_{1:n}^{(j)} \in A_\varepsilon^{(n)}).$$

However, there is a numerical issue: since the probabilities will be very small, the terms in the sum are going to be too big to represent on the computer. A classic "trick of the trade" to work around this problem is called the "log-sum-exp" trick, and it goes as follows.

The standard thing to do to represent very small (or very large) numbers $b_1, b_2, \ldots, b_m$ is to work with the logs of the numbers instead, $\beta_j = \ln b_j$. However, the problem we face above is that we

need to add up a bunch of very large numbers, and there is no obvious way to do this in a "log" representation (without exponentiating the numbers and adding them, then taking the logs again). (In other words, we want to compute $\sigma = \ln \sum b_j = \ln \sum e^{\beta_j}$.) But of course, if we exponentiate the β_j's first then we will have gained nothing by using our log representation. The trick is to do the following: let $c = \max \beta_j$, and compute

$$\sigma = c + \ln \sum e^{\beta_j - c}.$$

This shifts the largest numbers into a reasonable range, does the operation, and then shifts them back. It is possible that some of the very small b_j's will be annihilated by the operation $e^{\beta_j - c}$, but these would have contributed a negligible amount to the result anyway. I have implemented this "log-sum-exp" trick in the form of a Matlab function `logsum.m`, which is included along with these instructions.

So, to get the log of the sum in the importance sampling estimate $\hat{r}$, compute the values

$$\beta_j = \ln \left(\frac{1}{p(x_{1:n}^{(j)})} \right) = -k^{(j)} \ln \alpha - (n - k^{(j)}) \ln(1 - \alpha),$$

take the subset of these for which $k^{(j)}$ is typical, and pass them to the `logsum.m` function. For instance, in Matlab, all of this can be implemented as:

```
log_p = k*log(a) + (n-k)*log(1-a);
log_size = log(1/m)+logsum(-log_p(typical));
```

Now, the typicality theorem says that $|A_\varepsilon^{(n)}| \approx 2^{nH(X)}$ when n is large. To compare our estimate with the theoretical value, we should compare $\frac{1}{n} \log_2 \hat{r}$ with $H(X)$. (For instance, continuing in Matlab: `(1/n)*log_size/log(2)`.) Implement this comparison, and run it on the same values of n as above. How does it compare?

(Extra Credit) Jointly typical sequences

Making an empirical verification of the jointly typical theorem is more challenging, since n needs to be quite large, and it would appear that m needs to be extremely large. Nonetheless, with some more tricks from statistics (statistrics?), it is possible to get around these difficulties. The attached file `joint_typicality.m` implements this. For extra credit: figure out what this script is doing, why it makes sense, and check whether the results are consistent with the theorem.