Sample solutions to Homework 4, Information-Theoretic Modeling (Fall 2014)

Jussi Määttä

October 2, 2014

Question 1

[First, note that we use the symbol ! as an end-of-message symbol. When we see it, we know that the message has ended. This is not mentioned in the lecture slides.]

(a)

The cumulative function $F(x) = \sum_{y \leq x} p(y)$ takes the values $F(a) = 0.05$, $F(b) = 0.55$, $F(c) = 0.9$ and $F(!) = 1$.

For simplicity, denote $F_l(\cdot) = (0, 0.05, 0.55, 0.9)$ and $F_r(\cdot) = (0.05, 0.55, 0.9, 1)$. For instance, $F_l(c) = 0.55$ and $F_r(c) = 0.9$.

In general, if we have the interval $[a, b) \subset [0, 1]$ and we see a symbol x, the new interval will be

$$\Big[a + (b-a)F_l(x),\, a + (b-a)F_r(x)\Big)$$

which is a subinterval of $[a, b)$. This is nothing different from what is shown e.g. in the lecture slides; it's just the idea of recursive partitioning of intervals put into a formula.

Initially, for the "empty message", we have the interval $[0, 1)$. The interval after the first symbol c is $[F_l(c), F_r(c)) = [0.55, 0.9)$.

After the second symbol a, the interval becomes

$$\begin{aligned} I(ca) &= [0.55 + (0.9 - 0.55)F_l(a), 0.55 + (0.9 - 0.55)F_r(a)) \\ &= [0.55, 0.5675). \end{aligned}$$

After the third symbol b, the interval becomes

$$\begin{aligned} I(cab) &= [0.55 + (0.5675 - 0.55)F_l(b), 0.55 + (0.5675 - 0.55)F_r(b)) \\ &= [0.550875, 0.559625). \end{aligned}$$

Finally, we get

$$I(cab!) = [0.55875, 0.559625).$$

(b)

(i) The shortest codeword within the interval is 0.559. It is the only codeword within $I(cab!)$ that has less than four decimals.

(ii) We cannot use $C = 0.559$, because $0.5597 \notin I(cab!)$. But $C = 0.5595$ suffices. For any number D that is a continuation of C, we have

$$0.5595 = C \leq D \leq 0.5595999\ldots = 0.5596.$$

In fact, any $C \in \{0.5590, 0.5591, \ldots, 0.5595\}$ will do.

(c)

The probabilities have changed, so let us first find the new interval. We have $F_l(\cdot) = (0, 2/32, 18/32, 29/32)$ and $F_r(\cdot) = (2/32, 18/32, 29/32, 1)$. Hence,

$$\begin{aligned} I(c) &= [18/32, 29/32), \\ I(ca) &= [18/32, 299/512), \\ I(cab) &= [4619/8192, 4707/8192), \\ I(cab!) &= [18795/32768, 4707/8192). \end{aligned}$$

In binary, we have

$$I(cab!) = [0.100100101101011, 0.1001001100011).$$

Let's take a closer look at these numbers:

(0.)	1	0	0	1	0	0	1	0	1	1	0	1	0	1	1
(0.)	1	0	0	1	0	0	1	1	0	0	0	1	1		

The shortest codeword within $I(cab!)$ is 0.10010011. The (unique) shortest codeword whose all continuations are in $I(cab!)$ is 0.10010010111.

We got a prefix codeword of 11 bits. This agrees with the result mentioned in the lecture slides that

$$\left\lceil \log_2 \frac{1}{p(c)\,p(a)\,p(b)\,p(!)} \right\rceil + 1 = 11$$

bits is sufficient.

What would happen if we instead considered a Shannon symbol code? The codelength for $cab!$ would be

$$\left\lceil \underbrace{\log_2 \frac{1}{p(c)}}_{\approx 1.54} \right\rceil + \left\lceil \underbrace{\log_2 \frac{1}{p(a)}}_{=4} \right\rceil + \left\lceil \underbrace{\log_2 \frac{1}{p(b)}}_{=1} \right\rceil + \left\lceil \underbrace{\log_2 \frac{1}{p(!)}}_{\approx 3.42} \right\rceil = 11.$$

The Huffman code for single-letter symbols would give a code-length of 9 bits. So arithmetic coding does not *always* beat other codes (you should not be surprised by this).

Note 1. Consider the message $ccc!$. It seems reasonable to expect that arithmetic coding will fare better with this message: $\log_2(1/p(c))$ and $\log_2(1/p(!))$ are not integers and hence arithmetic coding should benefit. And indeed, the Shannon code gives code-length 10, Huffman gives 9 and arithmetic coding can encode the message into (0.)110111000, i.e. 9 bits. To better see the advantage of arithmetic coding, we would have to consider longer messages.

Note 2. If you got different answers and used a computer, you may have had problems with the accuracy of floating-point numbers. It's safest to do this with pen and paper, or use e.g. wxMaxima[1]. Of course, proper algorithmic implementations of arithmetic coding work around these issues; see for instance the paper by Witten, Neal & Cleary (link at the course webpage) for a C implementation.

[1] `https://andrejv.github.io/wxmaxima/`

Question 2

For $\theta \in \{0.25, 0.75\}$, the probability of the data is $0.25^2\, 0.75^2 = 9/256 \approx 0.035$. For $\theta = 0.5$, the probability is $0.5^4 = 1/16 \approx 0.063$.

Hence, with this quantization, using the maximum likelihood parameter gives the total code-length

$$\ell(0.5) + \log_2 \frac{1}{1/16} = 1 + 4 = 5.$$

(Had we used $\theta = 0.25$ or $\theta = 0.75$, the code-length would be $2 + \log_2(256/9) \approx 6.83$.)

Question 3

Consider first the more general setting where the parameter θ is distributed according to the beta distribution, $\theta \sim \text{Beta}(\alpha, \beta)$, $\alpha > 0$, $\beta > 0$. The so-called beta-binomial distribution corresponds to a generalization of the binomial distribution where before each trial, the success probability p is drawn from the beta distribution. The probability mass function of the beta-binomial distribution can be written in the form[2]

$$f(k) = \binom{n}{k} \frac{B(k + \alpha, n - k + \beta)}{B(\alpha, \beta)}$$

where n is the number of trials, k is the number of successes and $B(x, y)$ is the beta function[3].

In our case, the parameter θ has the uniform distribution, or equivalently

[2] `http://mathworld.wolfram.com/BetaBinomialDistribution.html`

[3] If x and y are positive integers, then we may simplify

$$B(x, y) = \frac{(x-1)!\,(y-1)!}{(x+y-1)!}.$$

$\theta \sim \text{Beta}(1,1)$, so we have

$$\begin{aligned} f(k) &= \binom{n}{k} \frac{B(k+1, n-k+1)}{B(1,1)} \\ &= \binom{n}{k} \frac{k!\,(n-k)!}{(n+1)!} \\ &= \frac{n!}{k!\,(n-k)!} \frac{k!\,(n-k)!}{(n+1)!} \\ &= \frac{n!}{(n+1)!} \\ &= \frac{1}{n+1} \end{aligned}$$

which we have to divide by $\binom{n}{k}$ because we are using an n-fold Bernoulli distribution instead of a binomial distribution. So for the sequence 0011, we have $n = 4$ and $k = 2$ and hence its probability is

$$p^w(0011) = \frac{1/(n+1)}{\binom{n}{k}} = \frac{1/5}{6} = \frac{1}{30}.$$

Hence, the mixture code-length is $\log_2(1/p^w(0011)) = \log_2 30 \approx 4.91$.

Note 1. This particular case can also be solved directly by integration,

$$\begin{aligned} \int_0^1 \theta^2 (1-\theta)^2 \,\mathrm{d}\theta &= \int_0^1 \left(\theta^4 - 2\theta^3 + \theta^2\right) \mathrm{d}\theta \\ &= \left[\frac{\theta^5}{5} - \frac{\theta^4}{2} + \frac{\theta^3}{3}\right]_0^1 \\ &= \frac{1}{5} - \frac{1}{2} + \frac{1}{3} = \frac{1}{30}, \end{aligned}$$

and then taking the logarithm.

Note 2. *(Optional, if you're interested:)* It was pointed out during the exercise session that there is yet another way to solve this, by doing "updates" on the beta distribution. The distribution $\text{Beta}(1,1)$ can be interpreted so that before seeing any data, we assume that we've already seen one 0 and one 1. Then we proceed as follows:

Bits seen	# of zeros	# of ones	Pr[0]	Pr[1]
	1	1	1/2	1/2
0	2	1	2/3	1/3
00	3	1	3/4	1/4
001	3	2	3/5	2/5

So the probability of the sequence 0011 is

$$\frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{4} \cdot \frac{2}{5} = \frac{1}{30}.$$

However, this approach may not be very illustrative of the point of mixture codes, so I will not go into details here.

Question 4

(a)

The number of 1's in a binary sequence of four bits is one of $0 \leq k \leq 4$. What is the maximum likelihood parameter $\hat{\theta}$?

For $k = 0$, the probability is $\theta^0 (1-\theta)^4 = (1-\theta)^4$, which is maximized when $\theta = 0$. Similarly, for $k = 4$ the maximum likelihood parameter is $\theta = 1$. For $k = 1, 2, 3$, the maximum likelihood parameter can be found taking the logarithm and differentiating:

$$\begin{aligned} \log p_\theta(x_1, \ldots, x_4) &= \log \theta^k (1-\theta)^{4-k} \\ &= k \log \theta + (4-k) \log(1-\theta), \\ \frac{\mathrm{d}}{\mathrm{d}\theta} \log p_\theta(x_1, \ldots, x_4) &= \frac{k}{\theta} - \frac{4-k}{1-\theta}. \end{aligned}$$

The derivative equals zero when $\theta = k/4$. (I leave it to the readers to convince themselves that this is the global maximum.)

Combining the above, we see that the maximum likelihood parameter for sequences with k 1's is $\hat{\theta}_{(k)} = k/4$.

Now, for a given $0 \leq k \leq 4$, there are $\binom{n}{k}$ four-bit sequences that have k 1's.

Hence, we may compute the NML normalizing constant as follows:

$$\begin{aligned} C &= \sum_{D\in\{0,1\}^4} p_{\hat{\theta}}(D) \\ &= \sum_{k=0}^{4} \binom{4}{k} \hat{\theta}_{(k)}^{k} \, (1-\hat{\theta}_{(k)})^{4-k} \\ &= \sum_{k=0}^{4} \binom{4}{k} \left(\frac{k}{4}\right)^{k} \left(\frac{4-k}{4}\right)^{4-k} \\ &\approx 3.2188. \end{aligned}$$

(b)

By the above, the NML code-length for $D = 0011$ is

$$\begin{aligned} \log_2 \frac{1}{p_{\mathrm{NML}}(0011)} &= -\log_2 \frac{p_{\hat{\theta}}(0011)}{C} \\ &= -\log_2 \frac{0.5^4}{C} \\ &= 4 + \log_2 C \\ &\approx 5.6865. \end{aligned}$$

Note 1. As mentioned in the problem statement, another option for calculating C would be to simply go through all possible sequences $(0000, 0001, 0010, \ldots, 1111)$, compute their maximum likelihood probabilities and sum them up. Then you have a sum of 2^n terms instead of a sum of $n+1$ terms.

Note 2. As was pointed out by someone at the exercise session, the sum from $k = 0$ to 4 can be made computationally easier by realizing that the terms for $k = 0$ and $k = 4$ agree, as do the terms for $k = 1$ and $k = 3$.

Note 3. For the sequence 0011, we got the following code-lengths: 5.00 (two-part code), 4.91 (mixture) and 5.69 (NML). What about the sequence 0000? The code-lengths will be approximately 3.66 (two-part code), 2.32 (mixture) and 1.69 (NML).

Note 4. This question was asked at the exercise session. Consider the code-lengths for 0011 for the two-part code and the mixture code. They both round up to 5 bits. So is the mixture code really any better than the two-part

code? The answer is that indeed it is: Consider, for instance, the problem of *model selection*: given some data, we want to pick a model (class) that seems to describe the data well. Our problem is then not compression but simply comparing the models classes, and hence there is no need to round up—and we will end up preferring the mixture model over the two-part code.

Note 5. *(Optional, if you're interested:)* Computing the value of C has been an interesting research problem. There's a linear-time algorithm for computing its value for multinomials (of which the Bernoulli is a special case) that was developed in our department. Take a look at the paper by Kontkanen and Myllymäki at `http://dx.doi.org/10.1016/j.ipl.2007.04.003` (should be accessible from the university network).