Problem Set 12

published: 27 July 2022
discussion: 29 July 2022

Data Compression With And Without Deep Probabilistic Models
Prof. Robert Bamler, University of Tuebingen

Course materials available at https://robamler.github.io/teaching/compress22/

Note: This problem set covers topics from the entire semester. The problems are designed to resemble more closely potential exam questions than some of the problems on previous problem sets.

Problem 12.1: Probabilities, Entropies, and Mutual Information

Consider an unbiased coin that, when thrown, comes up either "one" or "zero" with equal probability (and independent of any previous results). Assume that you flip the coin three times and let $X_i \in \{0, 1\}$ for $i \in \{1, 2, 3\}$ be the outcome of the i'th throw. Let $X_{\text{sum}} = X_1 + X_2 + X_3$ be the total number of "one" throws.

In the following, all entropies should be calculated in *bits* (i.e., with base 2), as we always did in this course.

(a) What is the entropy $H_P(X_i)$ for each $i \in \{1, 2, 3\}$? (no calculation required; it suffices if you state the correct result.)

(b) What is the joint entropy $H_P\big((X_1, X_2, X_3)\big)$ of the tuple (X_1, X_2, X_3)? (no calculation required; it suffices if you state the correct result.)

(c) What is the probability $P(X_{\text{sum}} = x_{\text{sum}})$ for each $x_{\text{sum}} \in \{0, 1, 2, 3\}$?

(d) What is the entropy $H_P(X_{\text{sum}})$? Provide your result in the form $H_P(X_1) = a + b \log_2 3$ where a and b are rational numbers. Hint: $\log_2 8 = 3$.

(e) What is the entropy $H_P\big((X_1, X_2, X_3, X_{\text{sum}})\big)$?

(f) What is the conditional probability $P(X_1 = x_1 \,|\, X_{\text{sum}} = 2)$ for each $x_1 \in \{0, 1\}$?

(g) What is the conditional entropy $H_P(X_1 \,|\, X_{\text{sum}} = 2)$? Provide your result again in the form $H_P(X_1) = a + b \log_2 3$ where a and b are rational numbers.

(h) Let $X'_{\text{sum}} := X_1 + X_2$. Which of the following two statements about mutual informations is true?

(i) $I_P(X_1; X_{\text{sum}}) \geq I_P(X_1; X'_{\text{sum}})$; or

(ii) $I_P(X_1; X_{\text{sum}}) \leq I_P(X_1; X'_{\text{sum}})$.

(you may assume that only one of the two statements is true.) *Hint:* no calculation is needed, but write a brief statement (at most one sentence) to justify your answer.

Problem 12.2: Source Coding Theorem & Huffman Coding

Consider the following probability distribution over a random variable X_i from the alphabet $\mathfrak{X} = \{\text{'a'}, \text{'b'}, \text{'c'}, \text{'d'}, \text{'e'}\}$:

$$P(X_i = \text{'a'}) = 0.3; \qquad P(X_i = \text{'c'}) = 0.25; \qquad P(X_i = \text{'e'}) = 0.3;$$
$$P(X_i = \text{'b'}) = 0.1; \qquad P(X_i = \text{'d'}) = 0.05.$$

(a) Draw a Huffman tree for this probability distribution and write down a table for a corresponding Huffman code $C_{\text{Huffman}} : \mathfrak{X} \to \{0,1\}^*$.

(b) Calculate the expected code word length, $L := \mathbb{E}_P\big[|C_{\text{Huffman}}(X_i)|\big]$ where $|\cdot|$ denotes the length of a bit string.

(c) Use your result for the expected code word length L and the source coding theorem to derive a lower and an upper bound for the entropy $H_P(X_i)$. Express your result in the form "a ? $H_P(X_i)$? $a+1$" where a is a rational number and each "?" denotes either "$<$" or "$\leq$".

Assume now that you have a random message $\mathbf{X} \in \mathfrak{X}^k$ consisting of k symbols $X_i \in \mathfrak{X}$, $i \in \{1, \ldots, k\}$ that are all statistically independent and distributed according to the above probability distribution. Assume that you encode these symbols with your Huffman code.

(a) What is the *expected* length $|C_{\text{Huffman}}(\mathbf{X})|$ of the encoded representation of the entire message $\mathbf{X}$?

(b) Let $\sigma^2 := \mathbb{E}_P\big[\big(|C_{\text{Huffman}}(X_i)| - L\big)^2\big]$ be the variance of the code word lengths in your Huffman code. Use the weak law of large number to provide a lower bound for the probability $P(|C_{\text{Huffman}}(\mathbf{X})|/k - L < \beta)$ for arbitrary $\beta \geq 0$.

Problem 12.3: Variational Inference

Consider a so-called "hierarchical" latent variable model with observed data (or "message") $\mathbf{X}$ and with *two* latent variables $\mathbf{Z}$ and $\mathbf{Z}'$. The joint probability distribution is:

$$P(\mathbf{Z}, \mathbf{Z}', \mathbf{X}) = P(\mathbf{Z})\, P(\mathbf{Z}'|\mathbf{Z})\, P(\mathbf{X}|\mathbf{Z}'). \tag{1}$$

Assume that we observe some data $\mathbf{x}$ and we now want approximate the (intractable) posterior distribution $P(\mathbf{Z}, \mathbf{Z}'|\mathbf{X} = \mathbf{x})$ with a variational distribution Q_ϕ (where ϕ are the variational parameters) that factorizes as follows,

$$Q_\phi(\mathbf{Z}, \mathbf{Z}' \,|\, \mathbf{X}{=}\mathbf{x}) = Q_\phi(\mathbf{Z} \,|\, \mathbf{X}{=}\mathbf{x})\, Q_\phi(\mathbf{Z}' \,|\, \mathbf{Z}, \mathbf{X}{=}\mathbf{x}). \tag{2}$$

To find the optimal variational parameters ϕ^*, we maximize the evidence lower bound (ELBO), which is defined analogous to our usual definition,

$$\mathrm{ELBO}(\phi) := \mathbb{E}_{Q_\phi(\mathbf{Z},\mathbf{Z}'|\mathbf{X}=\mathbf{x})}\left[\log \frac{P(\mathbf{Z},\mathbf{Z}',\mathbf{X}=\mathbf{x})}{Q_\phi(\mathbf{Z},\mathbf{Z}' \mid \mathbf{X}=\mathbf{x})}\right]. \tag{3}$$

Express the ELBO in the following form:

$$\mathrm{ELBO}(\phi) = -D_{\mathrm{KL}}\big(Q_\phi(\mathbf{Z} \mid \mathbf{X}=\mathbf{x}) \,\big\|\, P(\mathbf{Z})\big) - \mathbb{E}_?\big[D_{\mathrm{KL}}\big(? \,\big\|\, ?\big)\big] + \mathbb{E}_{Q_\phi(\mathbf{Z}'|\mathbf{X}=\mathbf{x})}\big[P(\mathbf{X}=\mathbf{x} \mid \mathbf{Z}')\big] \tag{4}$$

and fill in the three blanks marked with "?". Here, D_{KL} is the Kullback-Leibler divergence.

Problem 12.4: Bits-Back Coding And Asymmetric Numeral Systems (ANS)

Consider a latent variable model with latent variables $\mathbf{Z}$, observed data (or "message") $\mathbf{X}$, and joint probability distribution $P(\mathbf{Z},\mathbf{X}) = P(\mathbf{Z})\,P(\mathbf{X}|\mathbf{Z})$.

(a) Assume you want to *encode* a message $\mathbf{x}$ using this latent variable model and the bits-back trick. This means that you have to follow three steps, where each step can be phrased in the following form:

$$\left\{\begin{array}{c}\text{encode or}\\ \text{decode}\end{array}\right\}\left\{\begin{array}{c}\mathbf{x}\text{ or}\\ \mathbf{z}\end{array}\right\}\text{ with entropy model }\left\{\begin{array}{c}P(\mathbf{Z})\text{ or}\\ P(\mathbf{X} \mid \mathbf{Z}=\mathbf{z})\text{ or}\\ P(\mathbf{Z} \mid \mathbf{X}=\mathbf{x}).\end{array}\right\} \tag{5}$$

Write down the three steps for *encoding* $\mathbf{x}$ in the correct order, phrasing each step in the form of Eq. 5.

(b) Now formulate the three steps of the corresponding bits-back *decoder*, again phrasing each step in the form of Eq. 5.

(c) As you've learned in the lecture, the Asymmetric Numeral Systems (ANS) algorithm can be understood as bits-back coding for each symbol X_i. Listing 1 shows the code for a simple (albeit slow) ANS coder. The code listing also contains a usage example just in case it is not clear what the coder implementation does. Consider the methods `push` and `pop` for encoding and decoding a symbol, respectively. For each of the three steps of the bits-back algorithm that you identified in parts (a) and (b) above, identify the (possibly empty) set of lines in the code listing that implements that step.

Problem 12.5: Lossy Compression

Note: This is the only problem on the current problem set that would not be suitable for an exam question. This is done deliberately so that I can save a more self-contained problem for the exam.

In contrast to lossless compression, lossy compression can also be used for *continuous* data. Interestingly, the rate/distortion-theorem holds for continuos data in the same way as it does for discrete data, i.e., the optimal expected amortized bit rate for a given lossy compression is given by the mutual information $I_P(X;\hat{X})$ between the original message X and the reconstruction $\hat{X}$.

Consider a data source that generates scalar continuous messages $X \in \mathbb{R}$ with distribution $P(X) = \mathcal{N}(X; 0, \sigma^2)$, i.e., normal distributed with zero mean and standard deviation σ^2. After encoding and decoding, the reconstructed symbol $\hat{X} \in \mathbb{R}$ acquires some additional Gaussian noise with variance γ^2, i.e., $P(\hat{X}|X) = \mathcal{N}(\hat{X}; X, \gamma^2)$, i.e., given X, the reconstruction $\hat{X}$ is normal distributed with mean X and variance γ^2. Thus, the marginal distribution of the reconstruction, $P(\hat{X}) = \mathbb{E}_{P(X)}[P(\hat{X}|X)]$ is the convolution of two normal distributions. It is well-known that this convolution results again in a Gaussian, where the variances add up, i.e., $P(\hat{X}) = N(\hat{X}; 0, \sigma^2 + \gamma^2)$ (this is known as "Gaussian error propagation").

(a) The mutual information for continuous variables is defined as follows,

$$I_P(X;\hat{X}) = \mathbb{E}_P\left[\log \frac{p(X,\hat{X})}{p(X)\,p(\hat{X})}\right] \tag{6}$$

where lower case "p" denotes the probability density function. Convince yourself that, analogous to the case of discrete random variables, one can equivalently express $I_P(X;\hat{X})$ as follows,

$$I_P(X;\hat{X}) = h_P(\hat{X}) - h_P(\hat{X}|X) \tag{7}$$

where lower case h_P denotes the *differential* entropy $h_p(X) := -\int \log_2 p(X)\, dX$.

Note: while a differential entropy can be negative, one can show that the difference of differential entropies on the right-hand side of Eq. 7 is always positive.

(b) Look up the differential entropy of a normal distribution on Wikipedia (it is simply called "entropy" there) and calculate the mutual information (and thus the optimal expected amortized bit rate) $I_P(X;\hat{X})$ using Eq. 7. Express your result as a function of the *signal to noise ration* σ^2/γ^2.

```python
class SimpleAnsCoder:
    def __init__(self, precision, compressed=0):
        self.n = 2**precision          # ("**" denotes exponentiation)
        self.compressed = compressed

    def push(self, symbol, m):         # Encodes one symbol.
        z = self.compressed % m[symbol]
        self.compressed //= m[symbol] # ("//" denotes integer division)
        self.compressed = self.compressed * self.n + z

    def pop(self, m):                  # Decodes one symbol.
        z = self.compressed % self.n
        self.compressed //= self.n     # ("//" denotes integer division)
        for symbol, m_symbol in enumerate(m):
            if z >= m_symbol:
                z -= m_symbol
            else:
                break
        self.compressed = self.compressed * m_symbol + z
        return symbol

    def get_compressed(self):
        return self.compressed

# USAGE EXAMPLE:

# Define an approximate entropy model Q with 4 bits of precision and
# Q_i(X_i=0) = 7/2^4, Q_i(X_i=1) = 3/2^4, and Q_i(X_i=2) = 6/2^4.
precision = 4
m = [7, 3, 6]

# Encode an example message (in reversed order):
example_message = [2, 0, 2, 1, 0]
encoder = SimpleAnsCoder(precision)
for symbol in reversed(example_message):
    encoder.push(symbol, m) # We could use a different m for each symbol.
compressed = encoder.get_compressed()
print(f'Compressed bit string: {compressed:b}')

# Decode the example message:
decoder = SimpleAnsCoder(precision, compressed)
reconstructed = [decoder.pop(m) for _ in range(5)]
assert reconstructed == example_message # Verify correctness.
```

Listing 1: A simple (but slow) ANS coder.