Algorithms in the Real World (15-853), Spring 2018
Assignment #1

Instructions

Complete all problems (except the two exercises at the end).

Please typeset your solutions. Each problem should start on a newpage. Your solution to each problem should fit within a single page.

We will collect homework solutions electronically through Gradescope. The course code for 15-853 is **M5V4N8**.

We are using Piazza this semester. Participating by asking good questions and providing helpful answers is strongly encouraged!

You are not permitted to look at solutions of previous year assignments. You can work together in groups, but all solutions must be written up individually. If you get information from sources other than the course notes and slides, please cite the information, even if from Wikipedia or a textbook.

Part 1 (due 5pm Friday, January 26)

Please provide a simple illustration of your solution, more than just an answer.

Problem 1: Conditional Probabilities (10pt)
Consider the following conditional probabilities for a two state Markov Chain. What factor would one save by coding based on the conditional entropy instead of the unconditional entropy?

$$p(w|w) = .95 \quad p(b|w) = .05$$
$$p(w|b) = .15 \quad p(b|b) = .85$$

Problem 2: Arithmetic Codes (15pt)
Given the following probability model:

Letter	$p(a_i)$	$f(a_i)$
a	.1	0
b	.2	.1
c	.7	.3

Decode the 4 letter message given by `00011011010`, assuming it was coded using arithmetic coding. Why is this message longer than if we simply had used a fixed-length code of 2 bits per letter, even though the entropy of the set $\{.1, .2, .7\}$ is just a little more than 1 bit per letter? Note that once you figure out how to do the decoding, it should not take more than a few minutes with a calculator or scripting language.

Problem 3: Decoding Prefix Codes (20pt)
Being able to quickly decode prefix codes is extremely important in many applications.

Assume you have a machine with word length w (e.g. 32 bits). Suppose you are given some prefix code for a message set whose longest codeword is no more than $w/2$ bits. Assume that a sequence of codes is stored in memory broken into words (i.e. the first w bits are in the first word, etc.).

The naive way to decode is to use a binary tree and take constant time per bit by traversing the tree. Describe, in no more than half a page, how to decode each codeword in constant time (independently of w).

Hint: you can use $O(2^{w/2})$ preprocessing time, and the same amount of memory.

Problem 4: Bounds on Prefix Codes (20pt)

(a) Prove the first part of the Kraft-McMillan inequality for prefix codes. That is, show that for any prefix code C,

$$\sum_{(s,w)\in C} 2^{-l(w)} \leq 1.$$

(b) If a prefix code has $n = 2^k$ codewords and at least one codeword is shorter than k bits, prove that there must be at least two codewords longer than k bits.

Part 2 (due 5pm Friday, February 2)

Problem 5: PPM (20pt)
In this exercise you will implement your own version of the *model-part* of PPM-C algorithm (see the required reading for compression). You may use any programming or scripting language of your choice. Attach your code to the writeup. The performance of your code does not matter, but be careful of the correctness. *Your code must implement the optimization described in the last paragraph on page 31 of "Introduction to Data Compression".*

The website http://realworld.herokuapp.com provides a test input/output file for developing your code, as well as the input to use for parts (b)-(d) below. That input is based on your Andrew ID, so make sure that you've entered it correctly.

(a) Implement the PPM-C algorithm, taking the value of k as a parameter. Your code does not need to implement an encoder. Instead, it should output a log of each message (including the escape character <$>) and its probability Pr(char | context). It should also give the total sum of bits of the encoded messages (using the theoretic, non-rounded number of bits). Make sure that your code operates on bytes, rather than letters, so that there are 256 possible outcomes. Here is an example of possible output:

```
b, 0.15
a, 0.2
<$>, 0.4
u, 0.18
...
y, 0.45
Total bits:  93232.3
```

(b) Provide the theoretical number of bits encoded for your input (information content) using $k = 1, 2, 3, 4, 5$.

(c) For $k = 3$, provide the first 10 and last 10 lines of your message log.

(d) Using $k = 3$, does the total number of bits change if you reverse the order of characters in the input? Explain, in no more than two sentences.

Exercises

Solutions to these two problems are not to be handed in. They are meant to help you (a) brush up on your basic probability skills, and (b) tell you about a new theorem called Hoeffding's bound. Among other things, you may find it useful to recall basic properties of expectation and variance, and Markov and Chebychev's inequality.

Problem 6: A Quick Tester (5pt)
Your company is building rain detectors, but they are not entirely reliable. If it is not raining, your rain detector always says `No`; however, if it is raining it says `Yes` with probability $p > 0$ and `No` with probability $1 - p$. The errors are random, and if you have any collection of k such detectors, *they all behave independently of each other*.

1. Suppose it is raining, and I have two detectors. What is the probability both of them say `No`?
2. Suppose we take $k = \lceil \frac{\log_2 \delta}{\log_2(1-p)} \rceil$ independent detectors, and output the OR of these answers (i.e., output `Yes` if any of them answer `Yes`, and `No` if all answer `No`). Show that the chance of this output being incorrect is at most δ.

Problem 7: Simple Samplers (10pt)
Suppose X is a random variable which takes on values in the interval $[0, 1]$. Let $\mathbb{E}[X] = c$. Initially, you don't know anything about c, or about the probability distribution of X. However, you are given a black-box that every time you query it, it gives you an independent random sample drawn according to X. You want to estimate c.

> As an example, imagine that you are given a coin with unknown bias (i.e., probability of heads) c. Then every time you flip it you get 1 (i.e., "heads") with probability c, and 0 (i.e., "tails") with probability $1 - c$ (and this is independent over different flips). If X denotes the outcome of a coin toss, then $\mathbb{E}[X] = c$. You want to estimate the coin bias c.

A natural scheme is: sample from the black-box N times—call these samples $X_1, X_2, \ldots, X_N$—and return the *empirical mean* $\hat{c} := \frac{1}{N} \sum_{i=1}^{N} X_i$. We ask: how big does the number of samples N have to be so that

$$\Pr[|\hat{c} - c| \leq \varepsilon] \geq 1 - \delta. \tag{1}$$

I.e., you want to be within error ε with confidence $1 - \delta$.

1. Use Chebyshev's inequality to show that $N = O(\frac{1}{\varepsilon^2 \delta})$ samples suffice to ensure (1). Hence, to get $\delta = 1/n^k$ for some values n, k, we would take $O(n^k/\varepsilon^2)$ samples.

   > Hint: Given two *independent* random variables Y_1 and Y_2, suppose $Y = Y_1 + Y_2$. What is the mean of Y in terms of the mean of Y_1 and Y_2? How about the variances? What about when $Y = Y_1 + Y_2 + Y_3$ for independent r.v.s Y_i? Which of these statements would hold if they are not independent?

2. A famous bound called Hoeffding's bound about sums of independent bounded random variables says the following:

   Theorem 1 *Suppose $Y_1, Y_2, \ldots, Y_T$ are independent $[0, M]$-bounded random variables, and define $Y := \sum_{t=1}^{T} Y_t$ be their sum. Let $\mu = \mathbb{E}[Y]$. Then*

$$\Pr[Y \geq \mu + \lambda] \leq \exp\left\{-\frac{\lambda^2}{M(2\mu + \lambda)}\right\}$$
$$\Pr[Y \leq \mu - \lambda] \leq \exp\left\{-\frac{\lambda^2}{2M\mu}\right\}$$

(You don't have to prove this bound, but like Chebychev's bound it follows from Markov's inequality via a simple transformation.)

Use Hoeffding's bound to show that $N = O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ samples are sufficient to ensure (1). Hence, to get an error probability $\delta = 1/n^k$ we would take $O(\frac{k \log n}{\varepsilon^2})$ samples.