ORIE 4742: Information Theory and Bayesian ML
Homework 2
Spring 2021
Sid Banerjee (sbanerjee@cornell.edu)

Problem 1: Information Theory helps with Sports Betting

In class we have mainly looked at compression and communication (and soon, learning) as examples of applications of information theory. In this question, we will see another surprising application to betting (and more generally, portfolio optimization). Given that competitive cycling races are just restarting across the world, we will formulate this in the context of cycling races; hopefully this will also help you get ready for filling your March madness brackets!

Consider a competition with \( m \) riders, labeled \([m] = \{1, 2, \ldots, m\}\), where the \( i^{th} \) rider has a probability \( p_i \) of winning the competition. For convenience, we assume that \( p_1 > p_2 > \ldots > p_m \).

Part (a)

A betting market often ranks riders in terms of odds, where an odds of \( o_i \)-for-1 for cyclist \( i \) means that if you bet 1 dollar on rider \( i \), then you win \( o_i \) dollars if the rider wins (and nothing if the rider loses). How should the bookmaker set the odds for rider \( i \) in order to make sure that the expected reward from betting on \( i \) is 0, if it believed that the winning probabilities were \([p_i]\)? Such odds are called fair odds: prove that any fair odds obey \( \sum_{i=1}^{m} 1/o_i = 1 \).

Part (b)

If you had a different belief, in that you thought the winning probabilities were \([q_i]\), and wanted to bet 1 dollar, then which rider should you bet on?

Part (b)

Now suppose you are watching the Tour de France, and invest 1 dollar in betting on the winners of different stages. Every time you win, you can then reinvest some or all of your winnings; however, once you run out of money, you stop. In such a setting, a more important quantity to consider is not your winnings in one bet, but rather, your rate of returns over multiple rounds.

Suppose at the start of the \( t^{th} \) stage, you have a capital of \( S_t \) dollars, and suppose the winner of that stage is rider \( i \) with probability \( p_i \), and moreover, the market offers odds of \( o_i \). You decide to split your \( S_t \) dollars into bets of \( b_i S_t \) on each rider \( i \) (with \( \sum_i b_i = 1 \)). Express your expected wealth \( S_{t+1} \) at the end of the stage, and also, the expected doubling rate \( W_t(b, o, p) = \log_2(S_{t+1}/S_t) \).
Note: $W_t(b, o, p)$ is called the doubling rate because if you managed to keep getting a rate of $W$ over many stages, then your wealth grows roughly at a rate of $2^{tW}$ (by the law of large numbers...).

Part (c)

Now we want to try and find the betting strategy that maximizes the expected doubling rate. For this, argue that under any fair odds $o_i$ (with $\sum_i 1/o_i = 1$), winning probabilities $p_i$ and betting fractions $b_i$, we have:

$$W(b, o, p) = D(p||1/o) - D(p||b)$$

Using this, find the optimal betting strategy for maximizing the doubling rate.

Note: This is the so-called Kelly betting strategy, also called proportional betting; do you see why?

Part (d) (Optional)

To understand how this works in non-asymptotic settings, simulate the Kelly betting strategy, as well as any other strategy you want to compare with (in particular, greedy strategies where you bet on the most likely outcome), for a setting with 10 riders (with any distribution of your choice) and 21 rounds (as in the Tour de France). Can you beat the Kelly strategy?

The Kelly strategy is something that is actually used in practice in betting, and also, in stock markets, with a lot of success! If you are interested in knowing more, there are a lot of discussions available about this idea - for example, see this blog post (and links therein).

Problem 2: As random as can be: maximum entropy distributions

Once you accept the idea that entropy is a measure of randomness, a natural question is which distributions have the most entropy under different constraints. Indeed, when confronted by a probability distribution $P(x)$ about which only a few facts are known, the maximum entropy principle (or MaxEnt) suggests that one should always choose a distribution that maximizes entropy while satisfying these constraints.

This question requires you to optimize non-linear functions with constraints; for this, you need to remember how to use Lagrange multipliers. Before proceeding, please revise these (for example, go through the Khan academy lectures on constrained optimization.

Part (a)

Consider a distribution $p(x)$ over a finite set $\mathcal{X} = \{1, 2, \ldots, m\}$. Write down a constrained optimization problem for computing the maximum entropy distribution $p$, and solve it using Lagrange multipliers.

Note: We have already done this in class using Jensen’s inequality - however, doing it this way will be useful for the next part.
Part (b)
Consider next a distribution \( p(x) \) over the natural numbers \( \mathcal{X} = \{1, 2, \ldots\} \), and now suppose you are also told that \( E[X] = \mu \) (for some \( \mu \geq 0 \)). Write down a constrained optimization problem for computing the maximum entropy distribution \( p \), and solve it using Lagrange multipliers.

Part (c)
In the above setting (\( p(x) \) over finite set \( \mathcal{X} = \{1, 2, \ldots, m\} \)), suppose you are given a set of constraints of the form \( E[g_i(X)] = g_i \) for a collection of functions \( i \in \{1, 2, \ldots, k\} \). Argue that the maximum entropy distribution \( p \) in any such case is given by a distribution belonging to a so-called Exponential family, in particular, of the form \( p(x) = \exp \sum_{i=1}^{k} \theta_i g_i(x)/Z \), where \( Z \) is a normalizing constant, and \( \theta = \{\theta_i\}_{i=1}^{k} \) are parameters given by a series of implicit equations.

Note: Although we are asking this for a finite set, the idea works for infinite sets (as in part (b)) and also continuous sets. Indeed, almost all distributions you know correspond to MaxEnt distributions under some appropriate constraint: see Wikipedia article on MaxEnt

Part (d)
A common approach in binary classification problems is to fit the data using logistic regression, i.e.

\[
p(1|x) = \frac{e^{(\beta_0 + \sum_{i=1}^{k} \beta_i r_i(x))}}{1 + e^{(\beta_0 + \sum_{i=1}^{k} \beta_i r_i(x))}}
\]

where \( y \) is a binary random variable that indicates the class, and \( r_1(x), r_2(x), \ldots, r_k(x) \) are the features associated with each data point \( x \in \mathcal{X} \). Suppose \( X = x \) with probability \( q(x) \) for \( x \in \mathcal{X} \). Show that this model arises as the solution to the following maximum conditional entropy problem:

\[
\begin{align*}
\max_{p} & \quad H(Y|X) \\
\text{s.t.} & \quad p(y|x) > 0 \quad \forall y \in \{0, 1\}, \forall x \in \mathcal{X} \\
& \quad p(0|x) + p(1|x) = 1 \quad \forall x \in \mathcal{X} \\
& \quad \sum_{x \in \mathcal{X}} q(x)p(1|x)r_i(x) = \alpha_i \quad \forall i \in \{1, 2, \ldots, k\}
\end{align*}
\]

for some appropriate choice of \( \alpha_i \).

Problem 3: Source Coding and Random Number Generation
In this question, we will see some non-compression applications of compression algorithms.
Part (a)
Suppose we are given a random number generator that generates $X_i \sim \text{Bernoulli}(1/2)$. Describe how you can it to generate:
(i) $Y \sim \{a, b, c, d, e, f\}$ with prob $\{0.5, 0.25, 0.125, 0.125\}$
(ii) $Z \sim \{a, b\}$ with prob $\{1/3, 2/3\}$
(iii) suppose you are instead given a random number generator which returns $W_i \sim \text{Bernoulli}(p)$ for some unknown $p$. How can you use this to generate $X_i \sim \text{Bernoulli}(1/2)$?

Part (b)
Suppose we want to generate 1000 Bernoulli random variables with $p = 0.01$. Do this (in Python) using the following methods:
(a) To generate each bit, first generate 7 fair random bits (i.e., Bernoulli(1/2) samples) using `np.random.randint(2, size=7)`, and then use these to generate the desired bit by thresholding. *(Note that $2^7 > 100$)*
(b) Next, generate the entire sequence in one go via Arithmetic coding using the correct model, fed with a sequence of standard random bits (i.e., using `np.random.randint(2, size=1)`).
Compare how many bits you need using each method with the entropy of the source (for the second method, you should repeat the process to estimate the average number of bits).

Problem 4: Arithmetic Coding, Dictionary Coding, and Probabilistic Modeling
In this question, we will try and look at some examples of stream codes, to understand better what they do. Before (while...) doing this question, you should go over Chapter 6 in Mackay (if you have not already).

Part (a)
Suppose you want to store the number of days it takes for Sid to upload a homework. Each dataset consists of a string of 0s (depicting a day without the homework being uploaded), followed by a single 1 when the homework is actually uploaded. Moreover, suppose you also believe that the number of days till the homework is uploaded is Poisson(1/10).

Explain how this data is encoded using an arithmetic code (for a sequence of the form 0…01 of length $k \geq 1$, you should give both the upper and lower bounds of the interval, as well as the final output), and compute the expected length of the compressed data.

Part (b)
In the above example, outline how a length $k$ sequence (of the for 0…01) would be stored under LZW and again compute the expected length.
Part (c)

In many settings, you often need to build a probabilistic model simultaneously while computing an arithmetic code for some data. For this part, generate a sequence of length 100 from a Bernoulli($p$) source (for any $p$ of your choosing) – we will try to compress this without knowing the value of $p$. To do so, maintain two counters that count the number of 0s and 1s upto the $t^{th}$ bit (i.e., after observing the first $t$ bits $X_1, X_2, \ldots, X_t$, your counters $N_0[t]$ and $N_1[t]$ should store the number of 0s and 1s seen so far). Now compress the data using arithmetic coding assuming that:

1. **(Laplace model)** The probability at step $t$ is given by $P[X_{t+1} = 1] = \frac{N_1[t]+1}{t+2}$, $P[X_{t+1} = 0] = \frac{N_0[t]+1}{t+2}$.

2. **(Dirichlet model)** The probability at step $t$ is given by $P[X_{t+1} = 1] = \frac{N_1[t]+\alpha}{t+2\alpha}$, $P[X_{t+1} = 0] = \frac{N_0[t]+\alpha}{t+2\alpha}$, for a different values of $\alpha \in [0, 1]$ (you can try and optimize over $\alpha$ if you want; note that $\alpha = 1$ is the Laplace model).

Part (d) (Optional)

Use the Dirichlet model from above + arithmetic coding to compress your copy of War and Peace from last time (or any other text file of your choice…).

**Note 1:** The Dirichlet model for an alphabet of size $k$ is almost the same, but now with $P[X_{t+1} = x] = \frac{N_x[t]+\alpha}{t+k\alpha}$. You do need to know the number of unique characters (and this should include spaces, line-breaks and an end-of-book character).

**Note 2:** You should store the book as a sequence of bits; also try and write a decoder for the compressed file (this is easiest to test on a smaller string first).