Assignment-1

Pawan Kumar

January 21, 2024

Abstract

This is first assignment. In this you need to play with basic generative modeling concepts. Deadline: 28th Jan. Deadline can be relaxed, but it is recommended to do this before Quiz-1!

1 Problem

[5 Marks] We have seen in class that maximum likelihood estimates can be used to learn a parametric distribution. We first try a similar problem to get a practice on MLE estimate for generative modeling.

Consider a bag that is known to have only two balls green and blue. However, the number of the balls or their distribution is not known. We would like to know the distribution experimentally, by taking some samples from the bag.

Assume that we took six samples from the bag, and we obtained the balls: G,G,B,G,B,G. Fit a parametric distribution for these samples. After learning the parameters, use python to generate 3 samples from the learned distribution. You may use numpy `random.binomial(n, p, size=None)` with $n = 1$, the parameter p that you just learned to sample. See the source link https://numpy.org/doc/stable/reference/random/generated/numpy.random.binomial.html.

2 Problem

[5 Marks] In this problem, we would like to use a inverse CDF approach. We recall from the class that a generative modeling problem is defined as learning a function g such that $g : Z \to X$. That is we can sample from a known distribution $z \sim Z$, then $g(z)$ generates samples from $X \sim p$, where p is the required distribution. A basic approach we studied involves two steps:

1. sample from known distribution, in this case sample from interval $[0, 1]$
2. use inverse CDF $g = F^{-1}$, where F is the CDF function. Note that it may not be invertible, in that case, we consider inverse map.

This approach requires CDF to be invertible, although, it may not be (if one of the samples never shows up while samples, CDF will be flat corresponding to those) strictly invertible, you may output any one of multiple inverse values in those cases. That is if there exists an y such that $F^{-1}(y) = A$, where A is a set, then out any one entry of A picked uniformly as a sample!

To practice this approach, we again roll a 6 sided die 10 times and we obtain say $\{1, 3, 2, 4, 2, 3, 5, 6, 3, 2\}$. Compute PMF values for each of the possible outcomes of a die roll $\{1, 2, 3, 4, 5, 6\}$. Then draw PMF graph and CDF graph. Is the CDF graph strictly increasing? If not, is it invertible? What if CDF was strictly increasing, could it be necessarily invertible?

Using inverse CDF method, draw a sample from $[0, 1]$ on $Y-$axis, then with inverse CDF, generate a sample.

Generalize this hand calculations approach by writing a python code. Your code should consider n sided die. Although, in practice, there may not exist a die for any arbitrary n. Your code will take k samples, usually keep $k >> n$. Usually $k = 2n$ would be good number of samples. Plot the histogram of PMF values using any inbuilt Python command, for example

$$\texttt{numpy.histogram}(\texttt{a}, \texttt{bins} = \texttt{n}, \texttt{range} = \texttt{None}, \texttt{density} = \texttt{None}, \texttt{weights} = \texttt{None})$$

Here `bins=n`. Number of samples may get determined by the length of input vector a, where a is vector of samples.

To learn about how to plot CDF, you may want to refer to https://matplotlib.org/stable/gallery/statistics/histogram_cumulative.html. In inverse CDF method, you code will sample say u from uniform distribution in the range $[0, 1]$. For this you may see https://numpy.org/doc/stable/reference/random/generated/numpy.random.uniform.html. Now given that you have bin array of possible outcomes, say, $b = \{x_1, x_2, \ldots, x_k\}$ and you have CDF values $F = \{f_1, f_2, \ldots f_k\}$, locate where u in F array, it may lie in between $\{f_i, f_{i+1}\}, i < k$, set generated sample

$$x = \arg\min \{u - f_i(x_i), f_{i+1}(x_{i+1}) - u\},$$

where x could be either x_i or x_{i+1} for which above minimum is achieved. In case it is exactly in between, that is, if both $u - f_i(x_i)$ and $f_{i+1}(x_{i+1}) - u$ are same, then pick any one of f_i or f_{i+1} with a fair coin toss!

3 Generate intelligent people!

[10 Marks] From some sources it was found that Chimpanzee and humans differ in only few differences in pattern. However, it is still very hard to know the exact pattern. So, we want to create human like genes, without even knowing their distribution. Let us assume for simplicity that two genes denoted by 1 and 0 and their pattern makes the difference and how they are distributed. We take a random sample of genes from humans. For simplicity, let us assume that it is of length 10 only. We have the following 12 samples.

#	samples
s_1	[1,0,1,0,0,1,1,0,0,1]
s_2	[0,0,1,1,0,1,0,1,0,0]
s_3	[1,1,0,1,0,0,0,0,0,0]
s_4	[0,0,0,1,0,1,0,0,1,1]
s_5	[0,0,0,0,0,0,1,0,1,1]
s_6	[1,1,1,0,0,1,0,1,0,0]
s_7	[1,0,1,0,1,0,1,1,1,1]
s_8	[0,0,0,0,1,1,0,1,0,0]
s_9	[0,0,0,1,0,0,1,0,1,1]
s_{10}	[1,1,0,0,1,1,0,1,1,1]
s_{11}	[0,1,0,0,1,0,1,1,1,1]
s_{12}	[0,1,1,1,1,1,1,0,1,0]

One of the common string that we see is the occurrence of the strings "1, 0, 1" and "1,1" seems to be the defining pattern and perhaps few others. Our goal is to generate similar genes. Of course, during generative process, some may come out to be defective leading to disability in extreme cases, hopefully, such cases are rare, and our model can generate mostly good genes. Each of the 10 tuples in the sample could be considered as random variables, where the ith one is denoted by x_i. Answer the following:

1. If we assume Markov property, then from chain rule, the joint probability is given by:

   $$p(x_1, x_2, \ldots, x_{10}) = p_{\theta_1}(x_1)p_{\theta_2}(x_2|x_1)\cdots p_{\theta_n}(x_n|x_{n-1}),$$

   where θ_i are parameters of the distribution p corresponding to random variable X_i.

2. To estimate $p_{\theta_1}(x_1)$, we need to look into only the first entry of the samples s_1 until s_{12}. In fact, this is exactly similar to the first problem with 12 samples $\{1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0\}$ and two balls denoted by 0 and 1!

3. We now look at second distribution in chain rule. To estimate $p_{\theta_2}(x_2|x_1)$, we need to do "branching" (remember choice tree?), and try to learn a probability distribution for each branch. This

essentially means that θ_2 will have two components, one for each branch, say θ_2^1, θ_2^2. Let us assume that left branch corresponds to x_1 being 1 and the right branch corresponds to x_1 being 0. For left branch, we need samples that have $x_1 = 1$, which we find are $\{s_1, s_3, s_6, s_7, s_{10}\}$. Use the second element of these as samples to learn $p_{\theta_2^1}(x_2|1)$. Estimating this is similar to first problem: that is, use the samples $\{0, 1, 1, 0, 1\}$ to fit a Bernoulli (Is Bernoulli a bad choice?) for these 5 samples, and two outcomes (0 or 1 like balls in first problem). Similarly, learn $p_{\theta_2^2}(x_2|0)$ for right branch.

4. The process is same as above to calculate probability distribution for $p_{\theta_i}(x_i|x_{i-1})$. Note that here again θ_i has two components! Note that the conditional distribution can be also taken to be Bernoulli distribution.

5. Having learnt all the distributions, write a python code that does the following:

   (a) samples first entry $t_1 \sim p_{\theta_1}(x_1)$, that is sample t_1 from distribution $p_{\theta_1}(x_1)$.

   (b) if the generated sample $t_1 = 1$, then it generates sample t_2 from distribution $p_{\theta_2^1}(x_2)$, otherwise, it samples from distribution $p_{\theta_2^2}(x_2)$. Note that as mentioned in class two parameters are required to describe $p(x_2|x_1)$.

   (c) The above process is repeated until the 10th sample.

   (d) Report $[t_1, t_2, \ldots, t_{10}]$ as generated sample. We found in our sample that string "1,0,1" seems to be a common pattern, did the generated sample have this string? If not, then try generating few more, say 10 samples and see the pattern.

   Assuming that those who have "1,0,1" and "1,1" strings are intelligent humans, how many out of 10 are found intelligent. Is our chain rule method good enough to capture these? If not then what is wrong? If it does not capture these strings, then will conditioning on two previous samples help? That is,

$$p(x_1, x_2, \ldots, x_{10}) = p_{\theta_1}(x_1)p_{\theta_2}(x_2|x_1)p_{\theta_2}(x_3|x_1, x_2) \cdots p_{\theta_n}(x_n|x_{n-1}, x_{n-2}).$$

   Modify your code for this. Is this better to recognize the string "1,0,1"? Justify with generated samples. In general to identify a string of length ℓ, how many previous samples we may prefer to condition our distribution on? Does ordering of samples play any role? Suggest how this could be extended to more complicated Grey scale images. You may take one more week for this part. Please connect with me (after taking appointment) on `chatMGM` on course.iiit.ac.in to discuss if you have doubts.

   Further reading: https://arxiv.org/abs/1601.06759

   Happy Generating!