Assignment 2

CS215

Abhineet Majety
23B0923

Mohana Evuri
23B1017

Saksham Jain
23B1074

Fall 2024

Contents

1 Mathemagic 2

2 Normal Sampling 7

3 Fitting Data 16

4 Quality in Inequalities 26

5 A Pretty "Normal" Mixture 32

PROBLEM 1

Mathemagic

We explore some connections between special random variables via the notion of probability-generating functions.

Definition 1 (PGF, MGF). *Let X be a random variable taking on only non-negative integer values. Suppose that X is distributed according to probability mass function P. We define the probability-generating function of the distribution $P[X]$ of random variable X by*

$$G(z) := \mathbb{E}[z^X] = \sum_{n=0}^{\infty} P[X = n]z^n.$$

The moment-generating function *of the same distribution is defined by $M(t) := G(e^t)$ for t such that the series for G converges.*

§ Task A [1]

Problem Statement

Derive the PGF when X is a Bernoulli random variable with parameter p, that is, $X \sim \text{Ber}(p)$. Call this PGF G_{Ber}.

For Bernoulli random variable $X \sim \text{Ber}(p)$, the probability mass function is

$$P[X = x] = \begin{cases} 1 - p & x = 0 \\ p & x = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

Hence, the PGF for Bernoulli random variable is

$$\begin{aligned} G_{\text{Ber}}(z) &= E[z^X] \\ &= (1 - p)z^0 + pz^1 \\ &= 1 - p + pz. \end{aligned} \tag{1.2}$$

§ Task B [2]

Problem Statement

Let G_{Bin} be the PGF when $X \sim \text{Bin}(n, p)$. Show that $G_{\text{Bin}}(z) = G_{\text{Ber}}(z)^n$.

For Binomial random variable $X \sim \text{Bin}(n, p)$, the probability mass function is

$$P[X = x] = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & x = 0, 1, 2, \dots, n \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

Hence, the PGF for Binomial random variable is

$$\begin{aligned} G_{\text{Bin}}(z) &= E[z^X] \\ &= \sum_{x=0}^{n} \binom{n}{x} p^x (1-p)^{n-x} z^x \\ &= \sum_{x=0}^{n} \binom{n}{x} (pz)^x (1-p)^{n-x} \\ &= (pz + 1 - p)^n. \end{aligned} \tag{1.4}$$

Hence,

$$G_{\text{Bin}}(z) = G_{\text{Ber}}(z)^n \tag{1.5}$$

§ Task C ($\star$) [4]

Problem Statement

Suppose that $X_1, X_2, \cdots, X_k$ are independent non-negative-integer valued random variables, each distributed with the same probability mass function P. Let G be their common PGF. Consider random variable $X = X_1 + X_2 + \cdots + X_k$ defined on the cartesian product of the sample spaces underlying the random variables X_i. Let the PGF corresponding to X be G_Σ. Show that $G_\Sigma(z) = G(z)^k$.

Consider random variable $X = X_1 + X_2 + \cdots X_k$, where $X_1, X_2, \dots, X_k$ are independent non-negative integer-valued random variables with the same probability mass function P and probability generating function G.

We know that for two independent random variables X and Y, $E[XY] = E[X] \cdot E[Y]$. Also, z^X and z^Y are independent random variables. Using these results, the probability generating function for X is,

$$\begin{aligned} G_\Sigma(z) &= E[z^X] \\ &= E[z^{X_1 + X_2 + \cdots + X_k}] \\ &= E[z^{X_1} z^{X_2} \dots z^{X_k}] \\ &= E[z^{X_1}] \cdot E[z^{X_2}] \cdots E[z^{X_k}] \\ &= G(z)^k. \end{aligned} \tag{1.6}$$

§ Task D [1]

Problem Statement

Consider now the Geometric distribution. Let $X \sim \text{Geo}(p)$. Derive its PGF.

For the Geometric random variable $X \sim \text{Geo}(p)$, the probability mass function is

$$P[X = n] = (1-p)^{n-1} \cdot p \tag{1.7}$$

where $n = 1, 2, \dots$

Hence, the probability-generating function is

$$\begin{aligned} G_{\text{Geo}}(z) &= E[z^X] \\ &= \sum_{n=1}^{\infty} (1-p)^{n-1} p z^n \\ &= pz \cdot \sum_{n=0}^{\infty} ((1-p)z)^n \\ &= \frac{pz}{1-(1-p)z} \\ &= \frac{pz}{pz+1-z}. \end{aligned} \tag{1.8}$$

$|(1-p)z| < 1$ is needed for the geometric series obtained to converge.

§ Task E [3]

Problem Statement

Consider $X \sim \text{Bin}(n, p)$ and $Y \sim \text{NegBin}(n, p)$. Let their PGFs be $G_X^{(n,p)}(z)$ and $G_Y^{(n,p)}(z)$ respectively. Show using previous tasks or otherwise that for every $0 < p < 1$, we have

$$G_Y^{(n,p)}(z) = \left(G_X^{(n,p^{-1})}(z^{-1}) \right)^{-1}.$$

Consider $X \sim \text{Bin}(n.p)$ and $Y \sim \text{NegBin}(n, p)$. Then Y can be expressed as a sum of n geometric random variables $Y_1, Y_2, \ldots, Y_n \sim \text{Geo}(p)$ such that $Y = Y_1 + \cdots + Y_n$. Using the results 1.4, 1.6 and 1.8, the PGF of Y is

$$\begin{aligned} G_Y^{(n,p)}(z) &= \left(\frac{pz}{pz+1-z} \right)^n \\ &= \left(\frac{1}{1+\frac{1}{pz}-\frac{1}{p}} \right)^n \\ &= \left(\left(\frac{1}{pz} + 1 - \frac{1}{p} \right)^n \right)^{-1} \\ &= \left(G_X^{(n,p^{-1})}(z^{-1}) \right)^{-1} \end{aligned} \tag{1.9}$$

for every $0 < p < 1$.
As the previous task shows, the series converges for $|(1-p)z| < 1$.

That shows that the negative binomial distribution is not only morally an "inverse" of the binomial distribution (in that it models the number of trials while fixing the number of successes, while the binomial fixes the number of trials and models the number of successes) but negates the binomial distribution in every way possible! There is more; see Task F below.

§ Task F [2]

Problem Statement

We shall derive the negative binomial theorem using the result from Task E. First, we generalize the binomial coefficient: let $\alpha \in \mathbb{R}$ and $k \in \mathbb{N}$. Then

$$\binom{\alpha}{k} := \frac{\alpha(\alpha-1)\cdots(\alpha-k+1)}{k!}$$

Theorem 2 (Binomial theorem, negative exponent). *Let $n \in \mathbb{N}$ and $|x| < 1$. Then*

$$(1-x)^{-n} = \sum_{r=0}^{\infty} \binom{n+r-1}{r} (-x)^r = \sum_{r=0}^{\infty} \binom{-n}{r} x^r.$$

It is almost (the summation not being finite is the only difference) as if one could put a negative number for the exponent in the usual binomial theorem.

Consider $X \sim \text{Bin}(n.p)$ and $Y \sim \text{NegBin}(n,p)$. Using the definition of probability generating function, we have

$$\begin{aligned} G_Y^{(n,p)}(z) &= E[z^Y] \\ &= \sum_{r=0}^{\infty} P[Y=r] z^r \\ &= \sum_{r=n}^{\infty} \binom{r-1}{r-n} p^n (1-p)^{r-n} z^r \end{aligned} \tag{1.10}$$

since $P[Y=r] = 0$ for $0 \le r < n$. Substituting $r-n$ for r, and using result 1.9, we get

$$\left(G_X^{(n,p^{-1})}(z^{-1})\right)^{-1} = \sum_{r=0}^{\infty} \binom{r+n-1}{r} p^n (1-p)^r z^{r+n}. \tag{1.11}$$

Using result 1.4, we get

$$\begin{aligned} \left(\frac{1}{pz} + 1 - \frac{1}{p}\right)^{-n} &= p^n z^n \sum_{r=0}^{\infty} \binom{r+n-1}{r} (1-p)^r z^r \\ \implies (1+pz-z)^{-n} &= \sum_{r=0}^{\infty} \binom{r+n-1}{r} ((1-p)z)^r \\ \implies (1+z(p-1))^{-n} &= \sum_{r=0}^{\infty} (-1)^r \binom{r+n-1}{r} (z(p-1))^r. \end{aligned} \tag{1.12}$$

Substituting $x = z(p-1)$, we get

$$(1+x)^{-n} = \sum_{r=0}^{\infty} (-1)^r \binom{r+n-1}{r} x^r. \tag{1.13}$$

This is the negative binomial theorem. The series on the right converges for $|(1-p)z| < 1$, i.e., $|x| < 1$.

§ Task G [2]

Problem Statement

An easy consequence of all the hard work. Suppose the PGF of random variable X is $G(z)$. Show that the expectation of X is simply the derivative of G at 1, i.e., $\mathbb{E}[X] = G'(1)$. Using this and the PGFs constructed previously, derive the means of the Bernoulli, Binomial, Geometric and Negative binomial distributions are a function of their parameters.

The PGF of random variable X is defined as $G(z) = \sum_{n=0}^{\infty} P[X = n]z^n$. Hence

$$G'(z) = \sum_{n=0}^{\infty} P[X = n]nz^{n-1}. \tag{1.14}$$

Hence,

$$G'(1) = \sum_{n=0}^{\infty} P[X = n] \cdot n = E[X]. \tag{1.15}$$

Finding means of various distributions

Bernoulli: For random variable $X \sim \text{Ber}(p)$, using the result 1.2,

$$\begin{aligned} G(z) &= 1 - p + pz \\ \implies G'(z) &= p \\ \implies G'(1) &= p \\ \implies E[X] &= p. \end{aligned} \tag{1.16}$$

Binomial: For $X \sim \text{Bin}(n, p)$, using the result 1.4,

$$\begin{aligned} G(z) &= (pz + 1 - p)^n \\ \implies G'(z) &= n(pz + 1 - p)^{n-1}p \\ \implies G'(1) &= np \\ \implies E[X] &= np. \end{aligned} \tag{1.17}$$

Geometric: For $x \sim \text{Geo}(p)$, using the result 1.8,

$$\begin{aligned} G(z) &= \frac{pz}{pz + 1 - z} \\ \implies G'(z) &= \frac{p(pz + 1 - z) - pz(p - 1)}{(pz + 1 - z)^2} \\ \implies G'(1) &= \frac{p}{(p)^2} \\ \implies E[X] &= \frac{1}{p}. \end{aligned} \tag{1.18}$$

Negative Binomial: For $X \sim \text{NegBin}(n, p)$, using the result 1.9,

$$\begin{aligned} G(z) &= \left(\frac{pz}{pz + 1 - z}\right)^n \\ \implies G'(z) &= n\left(\frac{pz}{pz + 1 - z}\right)^{n-1} \frac{p(pz + 1 - z) - pz(p - 1)}{(pz + 1 - z)^2} \\ \implies G'(1) &= n(1)^{n-1}\frac{p}{p^2} \\ \implies E[X] &= \frac{n}{p}. \end{aligned} \tag{1.19}$$

PROBLEM 2

Normal Sampling

It's nice to know that standard normal random variables exist and are well behaved, but can we sample from a standard normal distribution?

Let's be clear on what we want to do. We would like to find an algorithm $\mathcal{A}$ that takes some uniformly random numbers in $[0,1]$, and generates an output $x \in \mathbb{R}$. The algorithm should use randomness of the numbers it receives in such a way that the output x of the algorithm is distributed standard normally. Denote by $f_{\mathcal{A}}(x)$ the PDF of the algorithm's output - a random variable - X. We wish that for every $x \in \mathbb{R}$,

$$f_{\mathcal{A}}(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

We will show and then use a general theorem to find such an algorithm. A note on notation: we shall use f_X to denote the PDF of the random variable X and F_X to denote the CDF.

§ Task A [2]

Problem Statement

Prove the theorem below.

Theorem 3. *Let X be a continuous real-valued random variable with CDF $F_X : \mathbb{R} \to [0,1]$. Assume that F_X is invertible. Then the random variable $Y := F_X(X) \in [0,1]$ is uniformly distributed in $[0,1]$.*

We are given the random variable Y, where $Y := F_X(X)$, where X is a continuous random variable with its CDF as $F_X : \mathbb{R} \to [0,1]$.

We proceed by finding the CDF F_Y of Y.

$$\begin{aligned} F_Y(Y) &= \mathrm{P}(Y \le y) \\ &= \mathrm{P}(F_X(X) \le y). \end{aligned} \tag{2.1}$$

As we know, F_X is a continuous, monotonically increasing, and invertible function; we get

$$F_X(X) \le y \implies X \le F_X^{-1}(y) \tag{2.2}$$

From 2.1 and 2.2, we have

$$F_Y(Y) = \mathrm{P}(X \le F_X^{-1}(y)) \tag{2.3}$$

Now, by the definition of F_X, we can rewrite the RHS of 2.3 as

$$\begin{aligned} F_Y(Y) &= P(X \le F_X^{-1}(y)) \\ &= F_X(F_X^{-1}(y)) = y \\ &= y \\ \implies F_Y(Y) &= y. \end{aligned} \tag{2.4}$$

Since $F_Y(y) = y$, the CDF of Y is that of a uniform random variable on $[0,1]$. Therefore, Y is uniformly distributed on $[0,1]$.

§ Task B [2]

Problem Statement

Suppose we are given a random variable Y, uniform over $[0,1]$. Explain how an algorithm $\mathcal{A}$ may be constructed taking as input a sample y according to the distribution of Y (so the algorithm is given just one uniformly random number in $[0,1]$), such that for every $u \in \mathbb{R}$, we have

$$F_{\mathcal{A}}(u) = F_X(u).$$

In other words, the output of $\mathcal{A}$ has the same cumulative distribution function as X, and upon taking the derivative, it has the same PDF.

Note: We are assuming that F_X is invertible (and obviously monotonic).

To construct an algorithm $\mathcal{A}$ that transforms a uniformly random variable Y over $[0,1]$ into a random variable with the same cumulative distribution function (CDF) as X, we can use the **Inverse Transform Sampling** method. (The most common method used for this purpose; this is the one which I learnt in my SoC.)

The Algorithm $\mathcal{A}$:

1. **Compute the Inverse CDF:** Let $F_X(u)$ be the CDF of the target random variable X. Find the inverse of this CDF, denoted as $F_X^{-1}(u)$.
2. **Generate a Uniform Random Variable:** The input to the algorithm $\mathcal{A}$ is a sample y drawn from the uniform distribution over $[0,1]$.
3. **Transform Using the Inverse CDF:** Set the output of the algorithm as

$$\begin{aligned} \mathcal{A}(Y) &= F_X^{-1}(Y) \\ F_{\mathcal{A}}(u) &= P(\mathcal{A}(Y) \le u) \end{aligned} \tag{2.5}$$

Verifying that $\mathcal{A}$ works:

We need to show that $F_{\mathcal{A}}(u) = F_X(u)$ for all $u \in \mathbb{R}$. Using 2.5 and the note:

$$\begin{aligned} F_{\mathcal{A}}(u) &= P(\mathcal{A}(Y) \le u) \\ &= P(F_X^{-1}(Y) \le u) \\ &= P(Y \le F_X(u)) \\ &= F_Y(F_X(u)) = F_X(u) \end{aligned}$$

where F_Y is the CDF of the uniform distribution, and since Y is uniformly distributed over $[0,1]$, $F_Y(y) = y$.

§ Task C [8]

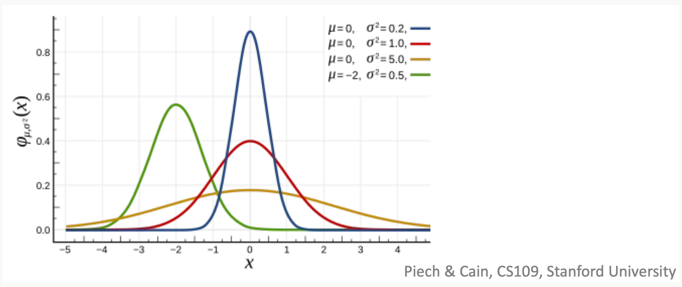
Problem Statement

In this task, you will use the `numpy.random` module to generate random numbers uniformly in $[0, 1]$ and the `norm` class (gives you access to F_X and F_X^{-1} for a Gaussian random variable X) from `scipy.stats` to sample from a Gaussian. In particular,

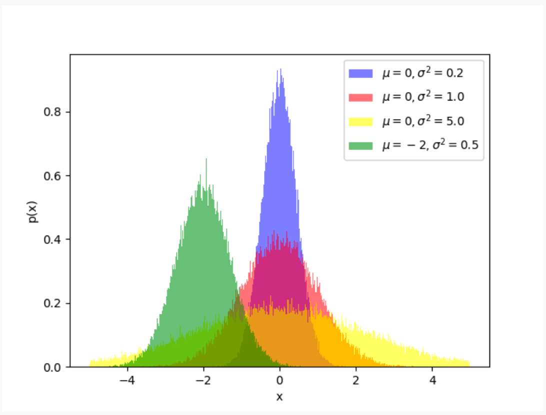
– Write a Python function `sample(loc, scale)` that samples from the Gaussian with mean at $x =$ `loc` and standard deviation `scale`.

– Generate $N = 1e5$ independent samples using the function above for the four-parameter choices $(\mu, \sigma^2) = (0, 0.2), (0, 1.0), (0, 5.0), (-2, 0.5)$.

– Plot the samples for each parameter choice using `matplotlib.pyplot`. You should roughly reproduce the shape of the plot of the four Gaussians from the lecture slides (i.e., figure 2.1a). The figure 2.1b is a plot of the samples; it mimics the top plot of the different PDFs, confirming that we have indeed sampled from the different Gaussian PDFs. Save the plot in `2c.png`.



(a) Gaussian



(b) The figure to be reproduced

Figure 2.1: The Bell curve, from uniformly random numbers.

You may not use the builtin `numpy.random.normal` or `random.randn` functions that directly sample from Gaussians for this task.

For the plot of the samples, you may not use kernel-density estimation (`kde`) tools; `matplotlib.pyplot` by itself is sufficient.

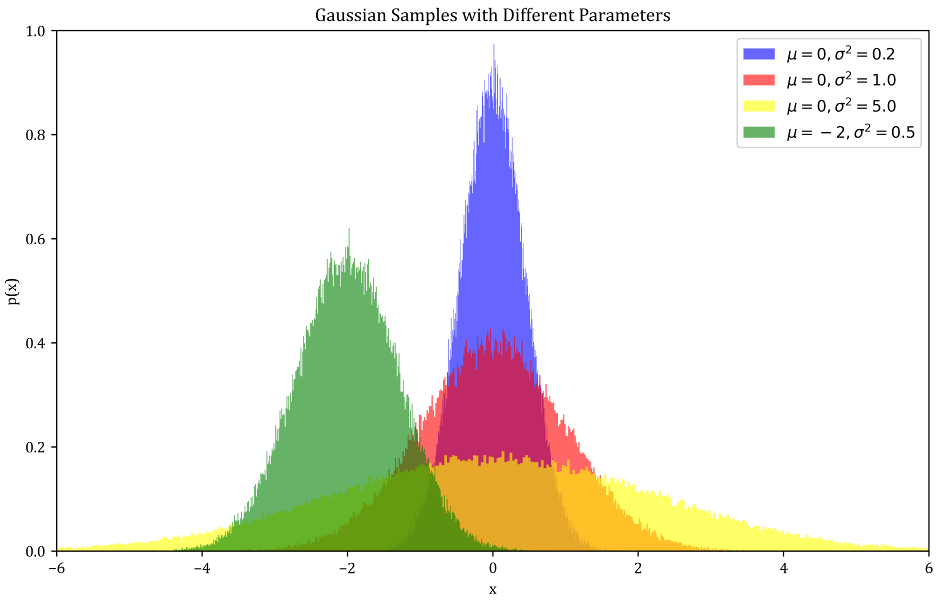
(You can use these if you wish for the next problem, however). Please write all your code for this task in one file called `2c.py`.

For the the function `sample`, we use the result 2.5 derived in Task B. We use the `norm` class from `scipy.stats` library. The strategy is to use the PPF (Inverse CDF) function of the `norm` class and pass the uniform randoms generated from `np.random.random` to it, to generate the **Gaussian RV**. The code for the function is at code:2.1.

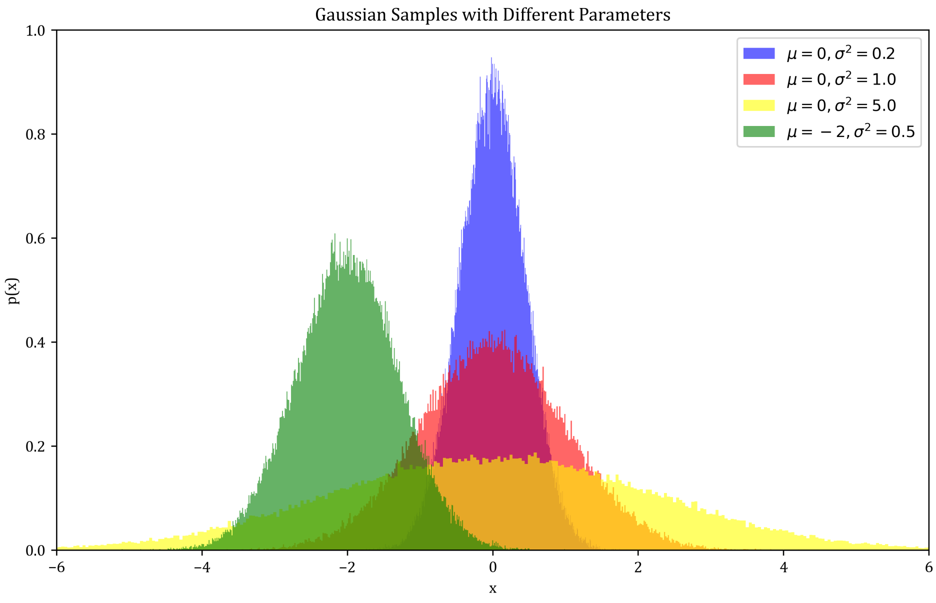
```python
import numpy as np
from scipy.stats import norm


def sample(loc: float, scale: float, size: int = 100000) -> np.ndarray:
    """
    Samples from a Gaussian using `scipy.stats.norm` and `np.random`
    """

    uniform_randoms = np.random.random(size)

    # PPF - Percent Point Function - Inverse of the CDF
    gaussian_randoms = norm.ppf(uniform_randoms, loc=loc, scale=scale)
    return np.array(gaussian_randoms)
```

Code 2.1: The function `sample`

Some of the reproduced images are 2.2a and 2.2b:



(a)



(b)

Figure 2.2: The reproduced images.

Code and Image locations

To generate the image, no special instructions required, just running the code `code/2c.py` will generate the image.

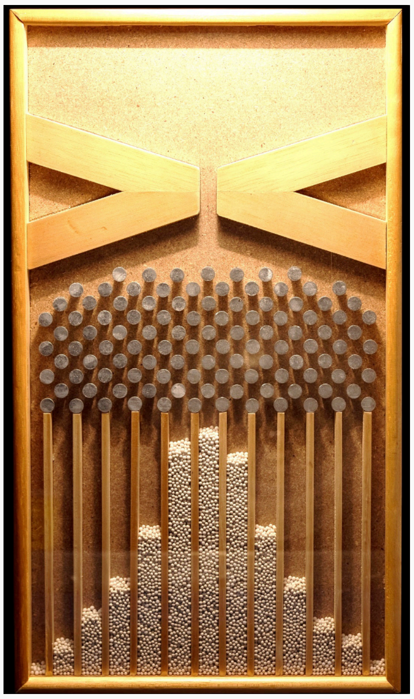
The files:

- `code/2c.py`
- `images/2c.png`

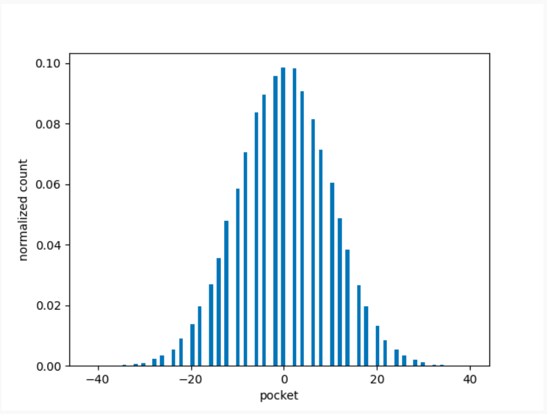
§ Task D (⋆) [8]

Problem Statement

Now we consider another way to sample normal distributions - approximately - using a Galton Board (see figure 2.3a for a picture). Imagine a ball that starts at the top of a Galton board. As it hits the first peg, it moves to the left of the peg with probability 1/2 and to the right of the peg with probability 1/2. After falling vertically downwards a little, it hits another peg. Again, it moves to the left or right of the new peg with equal probability. Suppose it makes h collisions with pegs before reaching the bottom of the Galton board. We call h the depth of the board. In the end, the ball falls into one of the wood-piece-separated pockets. There are $h + 1$ pockets at the bottom of the board, and each ball falls into exactly one of these as per the random directions it chose at each collision.



(a) A Galton board with $h = 10$ (ignore the leftmost and rightmost pockets).



(b) A simulation with $N = 10^5$ and $h = 100$.

Figure 2.3: Question 2, Task D

Consider simulating the motion of a large number N of balls along the Galton board, and record the fraction of balls that finally end up in each of the $h + 1$ pockets.

The simulation works by simulating the motion of each ball from the top to the bottom of the Galton board. Initially, the ball is at $x = 0$. In the first step, the ball moves left or right with equal probability - so its current position x is incremented or decremented with equal probability. The second step is identical - with a different starting position x. Suppose in the first step x was incremented, so $x = 1$ now. Then, in the second step, x is decremented (back to 0) or incremented (to 2) with equal probability. Perhaps it was incremented again. The third step decrements it to 1 or increments it to 3 with equal probability. Maybe it was decremented to 1. A fourth step takes it to 0 or 2 with equal probability. And so on. h steps ensue till the final value of its position x is the pocket the ball will fall into.

The choice of moving left or right can be simulated by simulating from {0, 1} uniformly randomly (find a function in `numpy.random` to do this). h uniform samples from {0, 1} thus allow us to simulate one ball.

The process can be repeated N times, with the final pockets of each simulated ball being recorded to yield a simulation of the Galton board.

Your task is to carry out this simulation for $N = 1000$ for the three values $h = 10, 50, 100$ of the depth of the board. For each value of h, plot the counts of balls in each pocket obtained as a histogram, with one bin for each pocket. The code is to be written in file `2d.py`, with the three histograms saved to files `2d1.png`, `2d2.png` and `2d3.png` (see figure 2.3b for a reference plot). What do you notice about the shape of the tops of the histogram?

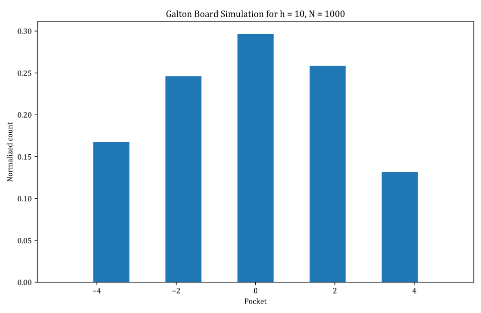
To simulate the Galton board, we can write a code as in code:2.2.

```python
import numpy as np


def simulate_galton_board(depth: int, number_of_balls: int) -> np.ndarray:
    """
    Simulates the Galton board with depth h and N balls.
    """

    # Random movement sequence, less than 0.5 = left, greater than 0.5 = right.
    random_movement = np.random.uniform(0, 1, (number_of_balls, depth))
    movement_array = np.where(random_movement < 0.5, -1, 1)

    final_positions_galton_board = np.sum(movement_array, axis=1)
    return final_positions_galton_board
```

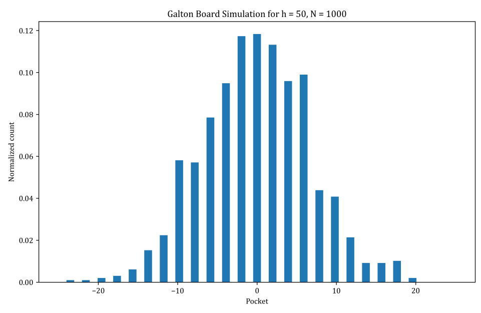
Code 2.2: The code to stimulate a Galton board

Here, we generate a random sequence of L and R for every ball, and calculate the final position for all the balls and output this array.

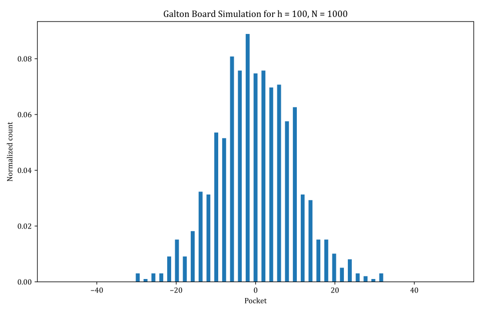
The simulations for N = 1000 and h = {10, 50, 100} give the images 2.4a, 2.4b and 2.4c:



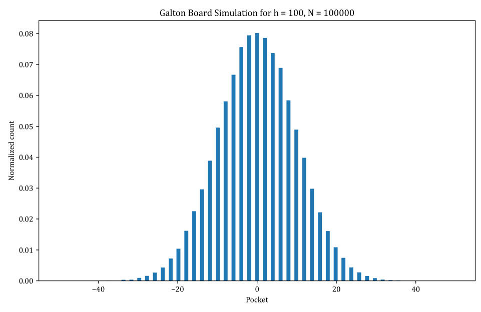
(a) $N = 1000, h = 10$



(b) $N = 1000, h = 50$



(c) $N = 1000, h = 100$



(d) $N = 10^5, h = 100$

Figure 2.4: Simulations for various h (Depth) and N (Dumber of balls)

Code and Image locations

To generate the images, no special instructions required, just running the code `code/2d.py` will generate the images.

The files:

- `code/2d.py`
- `images/2d1.png`
- `images/2d2.png`
- `images/2d3.png`

§ Task E (B) [5]

Problem Statement

The goal is to theoretically show that the number of balls in each pocket is normally distributed. Suppose that $h = 2k$ is even. Consider a Galton board of depth h. Let random variable $X \in \{-h, -h+2, \dots, 0, 2, \dots, h-2, h\}$ describe the pocket in which a ball finally lands (notice that X can only be even). The probability mass distribution $P_h[\cdot]$ of X is determined by the randomness of the outcome of each collision. There are two sub-tasks here.

- Compute (in closed form)the value $P_h[X = 2i]$ for each $i \in \{-k, -k+1, \dots, k-1, k\}$.
- Show that for large enough h and even $i \ll \sqrt{h}$,

$$P_h[X = i] \approx \frac{1}{\sqrt{\pi k}} e^{-\frac{i^2}{k}} = \mathcal{N}\left(\mu = 0, \sigma^2 = \frac{k}{2}\right)(i).$$

You may need to use Stirling's approximation for the factorial.

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

to solve this sub-task.

The PMF in closed format

As $X = 2i$ can be achieved by arranging a particular number of Ls and Rs (which can be calculated) in a random order and as the sample space is the total number of ways to get a string of Ls and Rs (From the way how a Galton board's distribution is generated in the code 2.2).

Now, to calculate the numbers of Ls and Rs: The sum of all the Ls(1) and Rs(-1) is $2i$, which means out of the h characters, we have $2i$ Ls and the rest are an equal amount of Ls and Rs.

$$\text{Number of Ls} = 2i + \frac{h-2i}{2} = \frac{h}{2} + i$$
$$\text{Number of Rs} = \frac{h-2i}{2} = \frac{h}{2} - i$$

$P_h[X = 2i]$ can be rewritten as:

$$P_h[X = 2i] = \frac{\text{Number of ways to arrange } \frac{h}{2} + i \text{ Ls and } \frac{h}{2} - i \text{ Rs}}{\text{The total number of ways to get a string of Ls and Rs}} \tag{2.6}$$

Using the principles of counting on 2.6:

$$P_h[X = 2i] = \frac{h!}{\left(\frac{h}{2} + i\right)! \cdot \left(\frac{h}{2} - i\right)! \cdot 2^h} \tag{2.7}$$

Equation 2.7 is the closed form of $P_h[X = 2i]$.

Approximation for a large h

I am writing the expression for $P_h[X = 2i]$ as $P_h[X = i]$ when i is odd doesn't make sense.

For large h, we use the Stirling's approximation on the factorials from the equation 2.7 to get

$$P_h[X = 2i] \approx \frac{\sqrt{2\pi h}\left(\frac{h}{e}\right)^h}{\sqrt{2\pi\left(\frac{h+2i}{2}\right)}\left(\frac{h+2i}{2e}\right)^{\frac{h+2i}{2}} \cdot \sqrt{2\pi\left(\frac{h-2i}{2}\right)}\left(\frac{h-2i}{2e}\right)^{\frac{h-2i}{2}} \cdot 2^h} \tag{2.8}$$

Simplifying (rearranging powers, cancelling terms, no approximations) 2.8 gives

$$P_h[X = 2i] \approx \frac{\sqrt{2}}{\sqrt{\pi h}\left(1 - \frac{4i^2}{h^2}\right)^{\frac{h+1}{2}} \cdot \left(\frac{h+2i}{h-2i}\right)^i} \tag{2.9}$$

Now approximating:

We begin by simplifying the term:

$$\left(1 - \frac{4i^2}{h^2}\right)^{\frac{h+1}{2}}$$

For large h, we assume that $\frac{i^2}{h^2}$ is small. We can then use the binomial expansion (or a Taylor series expansion) for small x, which states:

$$(1 - x)^n \approx e^{-nx} \text{ for small } x$$

Here, $x = \frac{4i^2}{h^2}$ and $n = \frac{h+1}{2}$, so we approximate:

$$\left(1 - \frac{4i^2}{h^2}\right)^{\frac{h+1}{2}} \approx \exp\left(-\frac{h+1}{2} \cdot \frac{4i^2}{h^2}\right)$$

For large h, the factor $\frac{h+1}{2}$ is approximately $\frac{h}{2}$, so this simplifies further to:

$$\exp\left(-\frac{h}{2} \cdot \frac{4i^2}{h^2}\right) = e^{-\frac{2i^2}{h}}$$

$$\implies \left(1 - \frac{4i^2}{h^2}\right)^{\frac{h+1}{2}} \approx e^{-\frac{2i^2}{h}} \tag{2.10}$$

Next, we simplify the term:

$$\left(\frac{h+2i}{h-2i}\right)^i$$

For large h, we can use the logarithmic approximation. First, express the ratio as:

$$\frac{h+2i}{h-2i} = 1 + \frac{4i}{h-2i}$$

For large h, $h - 2i \approx h$, so we approximate the ratio as:

$$\frac{h+2i}{h-2i} \approx 1 + \frac{4i}{h}$$

Now, we apply the logarithmic expansion for small x, which states:

$$\log(1 + x) \approx x \text{ for small } x$$

Thus:

$$\log\left(\frac{h+2i}{h-2i}\right) \approx \frac{4i}{h}$$

$$\left(\frac{h+2i}{h-2i}\right)^i \approx \exp\left(i \cdot \frac{4i}{h}\right) = e^{\frac{4i^2}{h}}$$

$$\implies \left(\frac{h+2i}{h-2i}\right)^i \approx e^{\frac{4i^2}{h}} \tag{2.11}$$

Using 2.10 and 2.11 in 2.9, we get

$$\begin{aligned}
P_h[X=2i] &\approx \frac{\sqrt{2}}{\sqrt{\pi h}} \cdot \frac{1}{e^{-\frac{2i^2}{h}} \cdot e^{\frac{4i^2}{h}}} \\
\implies P_h[X=2i] &\approx \frac{1}{\sqrt{\pi\left(\frac{h}{2}\right)}} \cdot e^{\frac{-i^2}{\left(\frac{h}{2}\right)}} \\
\implies P_h[X=2i] &\approx \frac{1}{\sqrt{\pi k}} \cdot e^{\frac{-i^2}{k}}
\end{aligned} \tag{2.12}$$

Comparing this result from 2.12 with the Gaussian form

$$\frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{\frac{-i^2}{2\sigma^2}},$$

We get $2\sigma^2 = k$.

So, $P_h[X=2i]$ is a Gaussian in i, with its variance $\sigma^2 = \frac{k}{2}$, as represented in 2.13.

$$\begin{gathered}
P_h[X=2i] \approx \mathcal{N}\left(\mu=0, \sigma^2=\frac{k}{2}\right)(i) \\
\text{(or)} \\
P_h[X=2i] \approx \mathcal{N}\left(\mu=0, \sigma^2=\frac{h}{4}\right)(i) \\
\text{for } i \ll \sqrt{h}.
\end{gathered} \tag{2.13}$$

PROBLEM 3

Fitting Data

In this problem, we are given a dataset from some distribution that we don't know offhand, and we want to find that distribution. Why? We can then predict future outputs sampled from the same distribution if we get close enough to the correct answer.

Suppose we have a dataset consisting of n samples of $S = \{x_1, \dots, x_n\}$, with each $x_i \in \mathbb{R}^d$. You are given a dataset which we will call $\mathcal{D}$ in the file `3.data`. We assume that there is an underlying probability distribution function P for a random variable X such that x_i was sampled independently from $P[X]$: the i^{th} sample is a random variable X_i with the same distribution as X, i.e., $P[X_i = x_i] = P[X = x_i]$. Notice that a dataset may then be treated as the collection of outcomes of a number n of experiments, each consisting of sampling X from its underlying distribution P.

Guessing or estimating the distribution P will tell us roughly what $X_{n+1}, X_{n+2}, \dots$ might turn out to be. This is useful, so trying to estimate an unknown distribution from a sample will be our goal for this question.

Code for all sub-parts should be written in `3.py`. If you wish, you may display plots with the code using an `.ipynb` file. Please name it `3.ipynb` in this case.

§ Task A [2]

Problem Statement

The i^{th} moment of a random variable is defined by $\mu_i = \mathbb{E}[X^i]$. When given a sample of n data points, we define the sample i^{th} moment as the moment of the random variable that is each datapoint with equal probability $1/n$. In particular, the sample i^{th} moment is

$$\hat{\mu}_i = \frac{x_1^i + \cdots + x_n^i}{n}$$

Compute the first two $(i = 1, 2)$ moments of $\mathcal{D}$. You may use `numpy` arrays and operations on them, but no loops are allowed to compute the moments (yes, it can be done with `numpy`).

The first two moments of $\mathcal{D}$ were calculated using `numpy.square` and `numpy.sum` functions.

```python
import numpy as np

f = open("3.data", "r")
D = np.loadtxt(f)

# First moment of D
m1 = np.sum(D) / D.size
m2 = np.sum(np.square(D)) / D.size
print(m1, m2, sep='\n')
```

Code 3.1: The first two moments of $\mathcal{D}$

Their values computed from the code:3.1 are

$$\hat{\mu}_1 = 6.4961456183248l7$$

and

$$\hat{\mu}_2 = 46.554361807879815$$

The code can be found in the **Task A** section in `code/3.ipynb`.

§ Task B [2]

Problem Statement

Let's first try to guess the distribution from our dataset graphically.

Recall that a dataset is simply the collection of outcomes of many independent identical experiments and that the probability of an element x models the fraction of experiments with outcome x.

Compute a histogram of the dataset and plot it. From the histogram, graphically guess the normal distribution parameter μ. Please attach any code used in file `3.py`, saving the histogram computed in file `3a.png`.

A histogram is typically used to "see" what kind of distribution the sample could have come from.

The histogram was plotted using `matplotlib.pyplot` module. The histogram can be found in `images/3b.png`. From the histogram, the mode of the distribution is around 6.

My Guess of μ: ~ 7.

The histogram plot of the data in `3.data` is at 3.1.

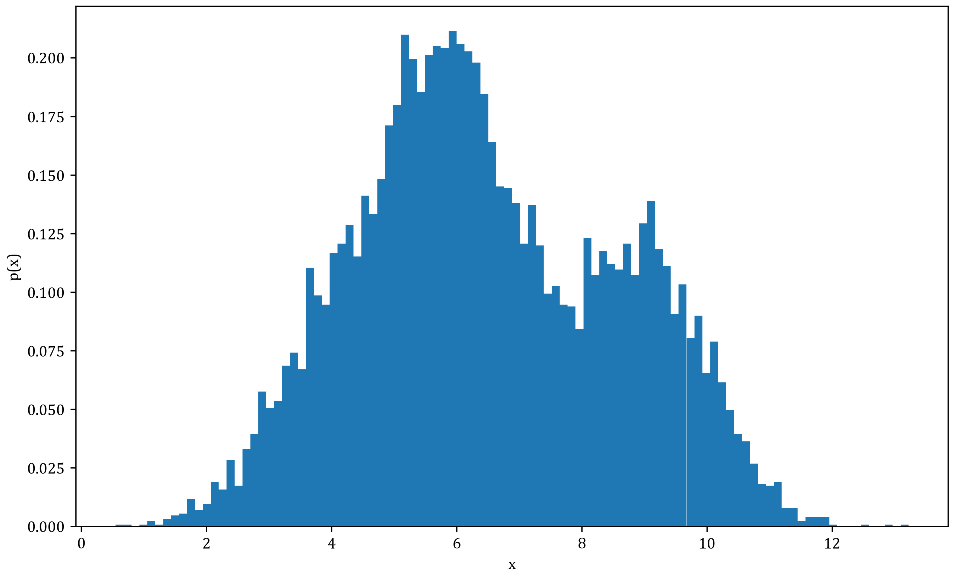


Figure 3.1: Histogram of the dataset

The code can be found in the **Task B** section in `code/3.ipynb`.

§ Task C ($\star$) [1+2+2]

Problem Statement

We are still nowhere near finding a distribution that fits our data well. Let's guess. It looks reasonably like a binomial distribution centered near about 6. Let us find the binomial distribution closest to our data.

One way to define closeness is to ask for the first few moments to be equal (it is a theorem in statistics that if every moment of two distributions is identical, then the two distributions must be identical as well - we are roughly approximating this theorem here).

In this task, we will find the best-fit binomial distribution to $\mathcal{D}$ by asking for the first two moments $\hat{\mu}_1$ and $\hat{\mu}_2$ to be equal to the corresponding two moments of $\text{Bin}(n, p)$, for some suitable choice of n and p.

(You need to estimate the mode, not the mean.)

1. First, compute an expression for the first two moments μ_1^{Bin}, μ_2^{Bin} of the distribution $\text{Bin}(n, p)$ as a function of n and p.
2. Then, use the `fsolve` function from `scipy.optimize` to compute a solution (n, p) to $\hat{\mu}_i = \mu_i^{Bin}, i = 1, 2$. Round n to either $\lfloor n \rfloor$ or $\lceil n \rceil$ based on which one satisfies the equalities better (they will not be exactly satisfied). Say the found parameters are (n^*, p^*).
3. Finally, using `numpy.linspace` and `scipy.stats.binom.pmf`, plot the binomial distribution $\text{Bin}(n^*, p^*)$ on top of the histogram of $\mathcal{D}$. It would help to get something like the figure in figure 3.2.

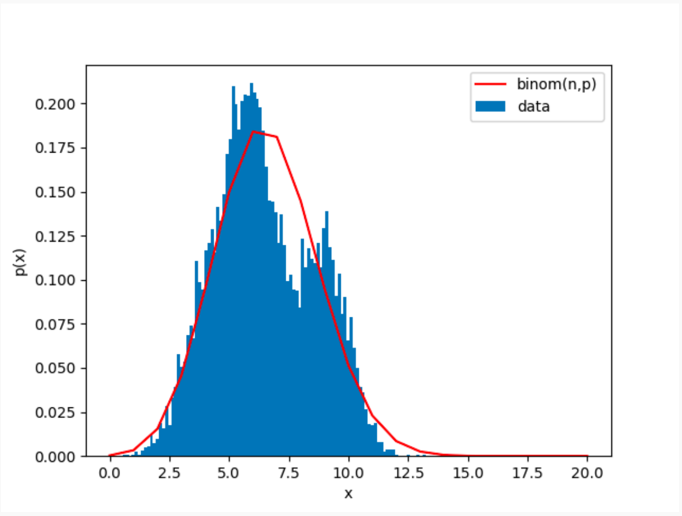


Figure 3.2: The best binomial distribution approximation to the true distribution

We will use the moment generating function $M(t) = \sum_{n=0}^{\infty} P[X = n]e^{nt}$ to compute the expressions for the first two moments of the distribution $\text{Bin}(n, p)$. For the binomial distribution, the PGF, as seen in 1.4 of Question 1, is

$$G(z) = (pz + 1 - p)^n.$$

Since $M(t) = G(e^t)$ by definition, we have

$$M(t) = (pe^t + 1 - p)^n.$$

Now, the two moments can be found using the relations $\mu_1^{\text{Bin}} = M'(0)$ and $\mu_2^{\text{Bin}} = M''(0)$.

$$\begin{aligned} M'(t) &= n(pe^t + 1 - p)^{n-1} \cdot pe^t \\ \implies M'(0) &= np \\ \implies \mu_1^{\text{Bin}} &= np. \end{aligned}$$

and for $n > 1$,

$$
\begin{aligned}
M''(t) &= npe^t[(n-1)p(pe^t+1-p)^{n-2} + (pe^t+1-p)^{n-1}] \\
\implies M''(0) &= np((n-1)p+1) \\
\implies \mu_2^{\text{Bin}} &= np((n-1)p+1) \\
\implies \mu_2^{\text{Bin}} &= np + n(n-1)p^2
\end{aligned}
$$

The parameters for the best-fit binomial distribution were calculated using the `scipy.optimize.fsolve` function. The parameters (n^*, p^*) are:

$$(n^*, p^*) = (20, 0.32968652963757006).$$

The graph plotted can be found in `images/3c.png`.
It was plotted using `scipy.stats.binom.pmf` function, `np.linspace` function and `matplotlib.pyplot`.
The plotted graph is at 3.3.

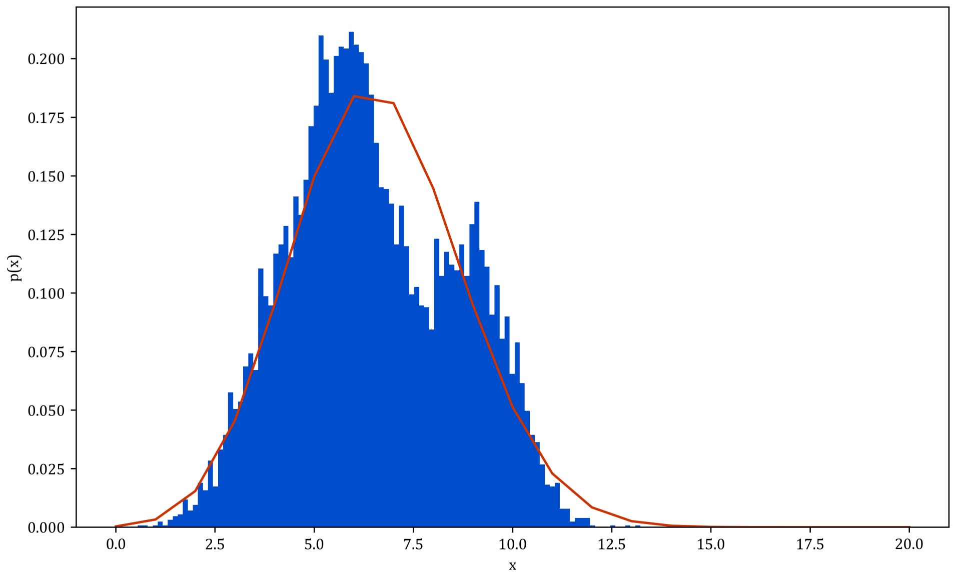


Figure 3.3: Binomial fit

The code can be found in the **Task C** section in `code/3.ipynb`.

§ Task D [3+2+2]

Problem Statement

Well, that was not too bad an approximation. Let us try another distribution, which is continuous this time. The gamma distribution is a two-parameter family of continuous probability distributions. Two parameters parameterize it: shape parameter k and scale parameter θ/ Its probability density function is given by

$$f(x;k,\theta) = \frac{1}{\theta^k\Gamma(k)}x^{k-1}e^{-\frac{x}{\theta}}$$

where $\Gamma(k) = \int_0^\infty t^{k-1}e^{-t}dt$ is the Gamma function. We now wish to find the best Gamma-distribution approximation to the true distribution of $\mathcal{D}$.

Your task and approach are essentially the same as in Task C. Restated for clarity, you must do the following:

1. First, compute an expression for the first two moments $\mu_1^\Gamma, \mu_2^\Gamma$ of the distribution $\Gamma(k,\theta)$ as a function of k and θ.
2. Then, use the `fsolve` function from `scipy.optimize` to compute a solution (k,θ) to $\hat{\mu}_i = \mu_i^\Gamma, i = 1,2$. No rounding is required since k, θ may be real. The found parameters are (k^*,θ^*).
3. Finally, using `numpy.linspace` and `scipy.stats.gamma.pdf` (the pdf takes three parameters: find out which ones are k and θ and which one is 0), plot the binomial distribution $\text{Bin}(k^*,\theta^*)$ on top of the histogram of $\mathcal{D}$. It would help to get something like the figure in the figure 3.4.

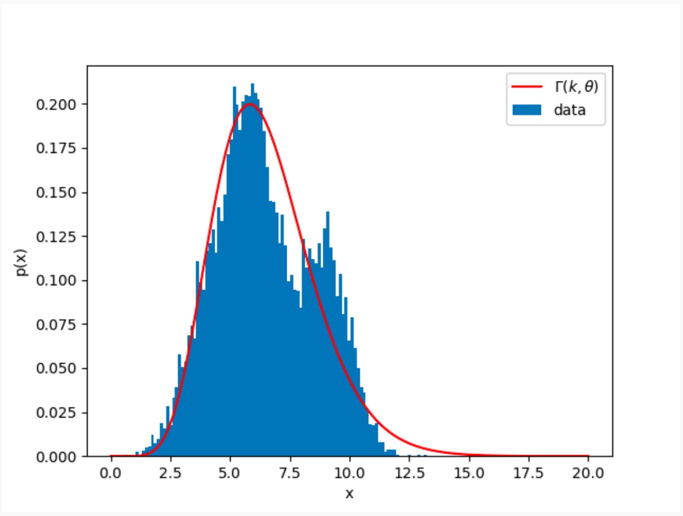


Figure 3.4: The best Γ distribution approximation to the true distribution

We will first find the moment-generating function for the gamma distribution. Let $x \sim \Gamma(k,\theta)$, then

$$\begin{aligned} M(t) &= E[e^{tx}] \\ &= \int_{-\infty}^{\infty} e^{tx}f(x;k,\theta)dx \\ &= \int_{-\infty}^{\infty} e^{tx}\frac{1}{\theta^k\Gamma(k)}x^{k-1}e^{\frac{-x}{\theta}}dx. \end{aligned}$$

The first moment μ_1^Γ can be evaluated as $M'(0)$. Now

$$M'(t) = \int_{-\infty}^{\infty} xe^{tx}\frac{1}{\theta^k\Gamma(k)}x^{k-1}e^{\frac{-x}{\theta}}dx.$$

Substituting $\frac{x}{\theta}$ for x,

$$\begin{aligned}
M'(t) &= \int_{-\infty}^{\infty} \frac{\theta}{\Gamma(k)} x^k e^{(t\theta-1)x} dx \\
\implies M'(0) &= \int_{-\infty}^{\infty} \frac{\theta}{\Gamma(k)} x^k e^{-x} dx \\
&= \frac{\theta}{\Gamma(k)} \int_{-\infty}^{\infty} k \cdot x^{k-1} e^{-x} dx \text{ [using chain rule]} \\
&= k\theta \\
\implies \mu_1^{\Gamma} &= k\theta.
\end{aligned}$$

Similarly, we will calculate the second moment μ_2^{Γ}.

$$\begin{aligned}
M''(t) &= \int_{-\infty}^{\infty} \frac{\theta}{\Gamma(k)} x^k \cdot \theta x e^{(t\theta-1)x} dx \\
\implies M''(0) &= \int_{-\infty}^{\infty} \frac{\theta^2}{\Gamma(k)} x^{k+1} e^{-x} dx \\
&= \frac{\theta^2}{\Gamma(k)} \int_{-\infty}^{\infty} (k+1) \cdot x^{k-1} e^{-x} dx \\
&= k(k+1)\theta^2 \\
\implies \mu_2^{\Gamma} &= k(k+1)\theta^2.
\end{aligned}$$

The parameters for the best-fit gamma distribution were found to be

$$(k^*, \theta^*) = (9.691205541218757, 0.6703134703624737).$$

The equations were solved using `scipy.optimize.fsolve`.
The graph was plotted using `scipy.stats.gamma.pdf` function. The plot can be found in `images/3d.png`.
The plotted graph is at 3.5.

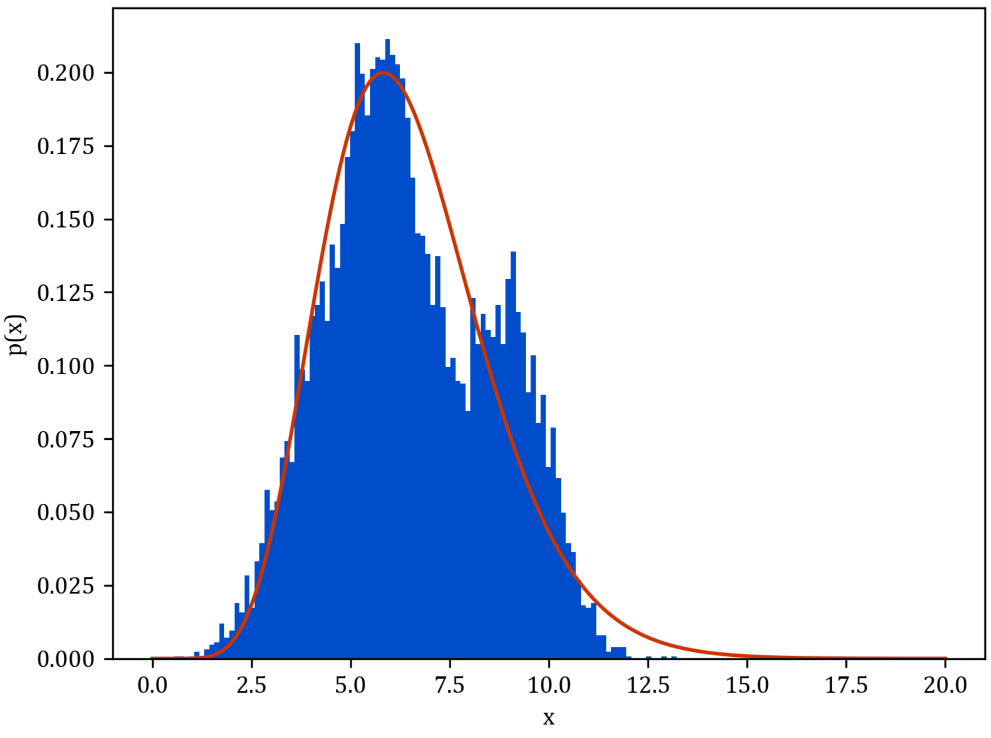


Figure 3.5: Gamma fit

The code can be found in the **Task D** section in `code/3.ipynb`.

§ Task E (⋆) [3+3]

Problem Statement

It looks like both of the distributions did a good job, but which one did a better job?

A straightforward way to find out is to compute what is called the likelihood of the dataset. This is simply the probability that the dataset would $\mathcal{D}$, supposing that the actual distribution was our best-fit distribution.

Definition 4 (Likelihood). *Given a dataset S and a choice of parameter $\lambda = \lambda_0$ for a family of distributions $P[\lambda]$. Parameterized by λ, define the likelihood of λ_0 by*

$$\mathcal{L}(\lambda_0|S) := P_{\lambda_0}[S] = \prod_{i=1}^{n} P_{\lambda_0}[X_i].$$

Here, $P_{\lambda_0}[x]$ is the PDF of the distribution whose parameter is λ_0. In our case, $P_\lambda[x]$ is $\text{Bin}(\lambda = (n,p))(x)$ or $\Gamma(\lambda = (n,p))(x)$.

Since each probability is a small number for large datasets, the likelihood can underflow. Thus, the average log-likelihood is typically calculated:

$$\ell(\theta|S) := \frac{\log \mathcal{L}(\theta|S)}{n},$$

where n is the size of the dataset. Calculate (in code, no for loops allowed) the average log-likelihood for both best-fit distributions. Since $\text{Bin}(n,p)(x)$ is nonzero only at integer values of x. Before computing the likelihood, you must round each data point to the nearest integer. No such thing is required for the Gamma distribution.

A larger likelihood is typically attributed to a better fit. Which distribution was a better fit?

The average log-likelihood for the best-fit binomial distribution is

$$\ell((n,p)|\mathcal{D}) = -2.15706811543467 85 \tag{3.1}$$

and the average log-likelihood for the best-fit gamma distribution is

$$\ell((k,\theta)|\mathcal{D}) = -2.1608217722066647. \tag{3.2}$$

The binomial distribution is a better fit.

The computation was done using `scipy.stats` module, and the `numpy.log` `numpy.sum` functions.
The code used for calculating this is at code:3.2.

```python
# Average log-likelihood of binomial distribution
rnd = np.round(D)
probs = scipy.stats.binom.pmf(rnd, n, p)
L_bin = np.sum(np.log(probs)) / D.size
print("Avg. log-likelihood of binomial distribution:", L_bin)

# Average log-likelihood of gamma distribution
probs = scipy.stats.gamma.pdf(D, k, 0, theta)
L_gamma = np.sum(np.log(probs)) / D.size
print("Avg log-likelihood of gamma  distribution:", L_gamma)

# The binomial distribution was a better fit
```

Code 3.2: Average log-likelihood

The code can be found in the **Task E** section in `code/3.ipynb`.

§ Task F [2+2+2+2]

Problem Statement

Notice the two peaks in the distribution? This immediately tells us that the distribution could not have been from a Binomial or Gamma function since those have a unique mode.

A typical distribution with two peaks is the Gaussian Mixture Model, which is the subject of Question 5. Please read Task A of Question 5 before moving ahead.

We will assume now that our distribution is composed of a two-component Gaussian mixture, each component having variance 1. That is, the distribution modeling of our data is assumed to have the pdf

$$P[x] = \frac{1}{\sqrt{2\pi}}\left(p_1 \exp\left(-\frac{(x-\mu_1)^2}{2}\right) + p_2 \exp\left(-\frac{(x-\mu_2)^2}{2}\right)\right)$$

We have four parameters to find, and so we need four moments. To make things easy, here are the first four moments of this distribution (here $\sigma_1 = \sigma_2 = 1$):

$$\begin{aligned}
\mu_1^{\text{gmm}} &= p_1\mu_1 + p_2\mu_2.\\
\mu_2^{\text{gmm}} &= p_1(\sigma_1^2 + \mu_1^2) + p_2(\sigma_2^2 + \mu_2^2).\\
\mu_3^{\text{gmm}} &= p_1(\mu_1^3 + 3\mu_1\sigma_1^2) + p_2(\mu_2^3 + 3\mu_2\sigma_2^2).\\
\mu_4^{\text{gmm}} &= p_1(\mu_1^4 + 6\mu_1^2\sigma_1^2 + 3\sigma_1^4) + p_2(\mu_2^4 + 6\mu_2^2\sigma_2^2 + 3\sigma_2^4).
\end{aligned}$$

As in Tasks C and D, compute the following:

1. First, compute $\hat{\mu}_i$ for $i = 3, 4$.
2. Then, use the `fsolve` function from `scipy.optimize` to compute a solution (μ_1, p_1, μ_2, p_2) to $\hat{\mu}_i = \mu_i^{\text{gmm}}, i = 1, 2, 3, 4$. No rounding is required. Say the found parameters are $(\mu_1^*, p_1^*, \mu_2^*, p_2^*)$.
3. Finally, using `numpy.linspace` and `scipy.stats.norm.pdf`, plot the GMM distribution obtained on top of the histogram of $\mathcal{D}$. You should get something like the figure in figure 3.6.

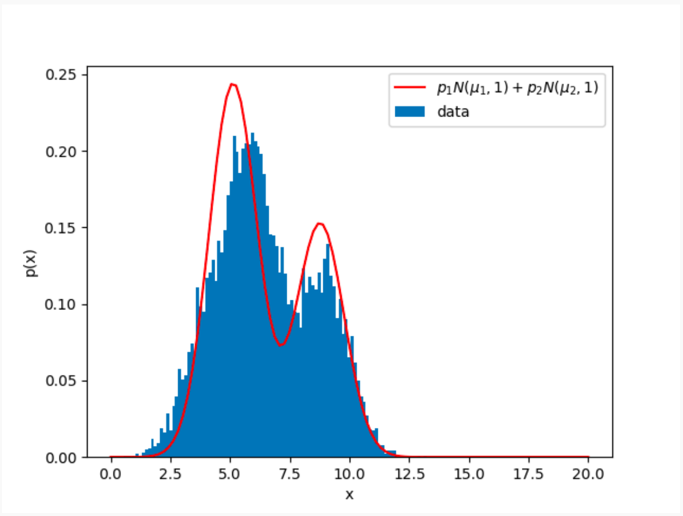


Figure 3.6: The best two-component unit-variance GMM approximation to the true distribution

To finish, compute the average negative log-likelihood of the obtained GMM distribution. Is it better an approximation than the previous two?

The 3rd and 4th moments of $\mathcal{D}$, $\hat{\mu}_3$ and $\hat{\mu}_4$ are

$$\hat{\mu}_3 = 360.56586952543273$$

and

$$\hat{\mu}_4 = 2968.068491427333.$$

They can be calculated in a way similar to the one mentioned in **Task A**.

We have used a two-component Gaussian mixture, each component having variance 1. The distribution is assumed to have the pdf:

$$P[x] = \frac{1}{\sqrt{2\pi}} \left(p_1 \exp\left(-\frac{(x-\mu_1)^2}{2}\right) + p_2 \exp\left(-\frac{(x-\mu_2)^2}{2}\right)\right).$$

The average negative log-likelihood of the best-fit GMM distribution is

$$-\ell((p_1, p_2, \mu_1, \mu_2)|\mathcal{D}) = 2.1830387449113133.$$

This turns out to be a worse approximation than the other two. Among the three distributions seen till now, the binomial distribution is the best fit.

The plot can be found in `3f.png` and at 3.7.

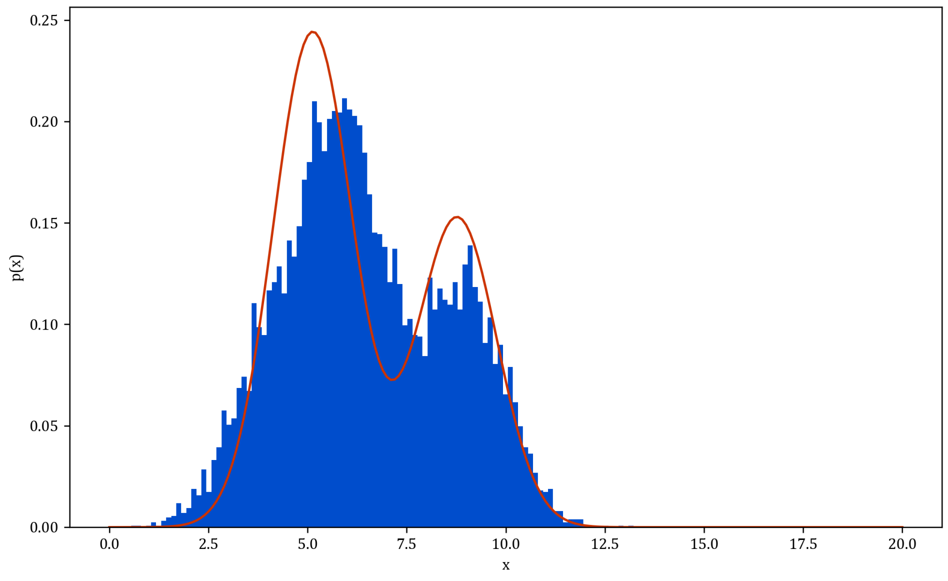


Figure 3.7

This task shows the power of a more versatile distribution like the GMM; indeed, clustering is typically done this way, using a GMM, the only difference being that an equation solver like `fsolve` is replaced by a heuristic called *expectation maximization*, since solvers are slow for many-component GMMs.

The code can be found in the **Task F** section in `code/3.ipynb`.

§ Teaser

A different choice of distribution, with six parameters, gave the fit in the figure 3.8. An important part of machine learning is the parameter estimation to fit data. Feedforward neural networks typically do just that. As we have seen, more parameters can get you closer to the "right distribution". It is not surprising that GPT has a few billion parameters that it learns. It is not just roses, though. A large number of parameters runs the risk of overfitting. Data Analysis and ML study better ways to learn parameter values, better families of distributions, how to detect and avoid overfitting, and more.

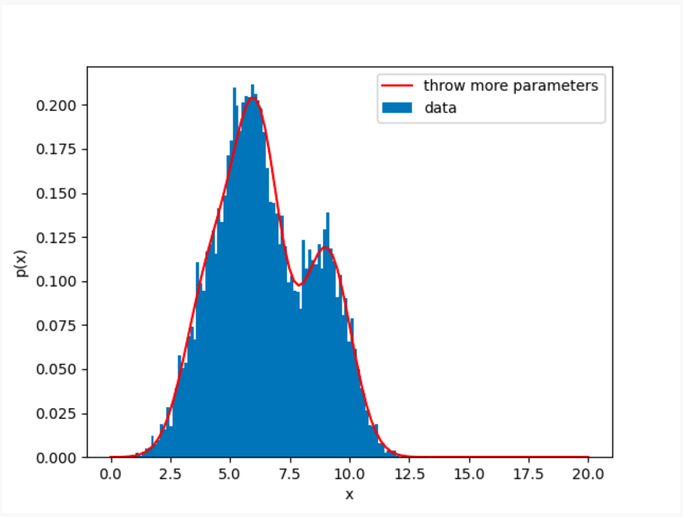


Figure 3.8: Two more parameters bring the total to 6 parameters estimated using the first six moments

Ungraded Bonus.

Can you guess what the true distribution is? Winners can claim a coffee treat at Cafe92 from the authors.

The distribution which I got, to the closest of that curve is a 3 GMM.
I added what I got in the file `code/3.ipynb`.

PROBLEM 4

Quality in Inequalities

Let us dive deeper into the inequalities we have studied in class (and a new one):

Definition 5 (Markov's Inequality). *Let X be any non-negative random variable and $a > 0$,*

$$\mathrm{P}[X \geq a] \leq \frac{\mathbb{E}[X]}{a}.$$

§ Task A [1+2]

Problem Statement

Give an intuitive "proof" for this inequality and a reason why it could be correct (you can try playing around with different X's and a's).

Now, prove this inequality rigorously using continuous random variables. Try to reason about the definition of expectation and how you can manipulate it to serve your purpose.

The variable X is a non-negative random variable. Hence, there is a lower bound on the value that X can take. Intuitively, for a given expected value $\mathbb{E}[X]$, there must also be a upper bound on the probability of X taking a large value.

Let us look at the rigorous proof. Consider a continuous random variable x with pdf $f(x)$, and let $a > 0$. The expected value is defined as

$$\begin{aligned}\mathbb{E}[x] &= \int_0^\infty xf(x)dx \\ &= \int_0^a xf(x)dx + \int_a^\infty xf(x)dx \\ &\geq \int_0^a 0 \cdot f(x)dx + \int_a^\infty a \cdot f(x)dx \\ &\geq 0 + a\int_a^\infty f(x)dx.\end{aligned}$$

Now, $P[x \geq a] = \int_a^\infty f(x)dx$. Hence,

$$\begin{aligned}\mathbb{E}[x] &\geq a \cdot P[x \geq a] \\ \implies P[x \geq a] &\leq \frac{\mathbb{E}[x]}{a}.\end{aligned}$$

Hence, Markov's inequality holds.

§ Task B [4]

Problem Statement

Now that we have established this inequality, let us move on to linking it to what we have already studied in class is the Chebyshev-Cantelli inequality.

Use Markov's inequality to prove the following version of the Chebyshev Cantelli inequality for a random variable X with mean μ and variance σ^2: for every $\tau > 0$, we have

$$\mathrm{P}[(x-\mu) \geq \tau] \leq \frac{\sigma^2}{\sigma^2+\tau^2}.$$

Let X be any random variable with mean μ and variance σ^2. Consider a new random variable $Y = X - \mu$. Then $\mathbb{E}[Y] = 0$ and $Var[Y] = \sigma^2$. Let $\tau > 0$ be a number. Let t be a number such that $t + \tau > 0$. Then

$$\begin{aligned} \Pr[Y \geq \tau] &= \Pr[Y + t \geq \tau + t] \\ &= \Pr\left[\left(\frac{Y+t}{\tau+t}\right) \geq 1\right] \\ &\leq \Pr\left[\left(\frac{Y+t}{\tau+t}\right)^2 \geq 1\right] \end{aligned}$$

Using Markov's inequality on the non-negative random variable $\frac{Y+t}{\tau+t}$, we get

$$\begin{aligned} \Pr[Y \geq \tau] &\leq \mathbb{E}\left[\left(\frac{Y+t}{\tau+t}\right)^2\right] \\ &\leq \frac{\mathbb{E}[(Y+t)^2]}{(\tau+t)^2} \\ &\leq \frac{\mathbb{E}[Y^2]+0+t^2}{(\tau+t)^2} \\ &\leq \frac{0+\sigma^2+t^2}{(\tau+t)^2}. \end{aligned}$$

Hence, the inequality

$$\Pr[Y \geq \tau] \leq \frac{\sigma^2+t^2}{(\tau+t)^2}$$

should hold for any t such that $t + \tau > 0$. It can be shown that the RHS is minimized for $t = \frac{\sigma^2}{\tau}$. Also, $\tau + t \geq 0$ in this case. Hence

$$\begin{aligned} \Pr[X-\mu > \tau] &\leq \frac{\sigma^2 + \frac{\sigma^4}{\tau^2}}{\left(\tau + \frac{\sigma^2}{\tau}\right)^2} \\ &\leq \sigma^2 \frac{\tau^2+\sigma^2}{(\tau^2+\sigma^2)^2} \\ &\leq \frac{\sigma^2}{\sigma^2+\tau^2}. \end{aligned}$$

Hence, Chebyshev-Cantelli inequality holds.

§ Task C [3]

Problem Statement

Yay, our inequalities are successfully linked! Now we can move on to proving a strong bound through these inequalities... start by showing that for a random variable X where $M_X(t)$ represents the MGF (see Question 1) for X, the following hold:

$$P[X \geq x] \leq e^{-tx} M_X(t) \; \forall t > 0.$$

$$P[X \leq x] \leq e^{-tx} M_X(t) \; \forall t < 0.$$

The MGF is defined by $M_X(t) = \mathbb{E}[e^{tX}]$. Now, for $t > 0$,

$$\begin{aligned} \Pr[X \geq x] &= \Pr[tX \geq tx] \\ &= \Pr[e^{tX} \geq e^{tx}]. \end{aligned}$$

Now, using Markov's inequality on the random variable e^{tX}, we get

$$\Pr[e^{tX} \geq e^{tx}] \leq \mathbb{E}[e^{tX}] \cdot e^{-tx}.$$

Hence, using the definition of MGF,

$$\Pr[X \geq x] \leq e^{-tx} M_X(t) \; \forall t > 0. \tag{4.1}$$

Similarly, for $t < 0$,

$$\begin{aligned} \Pr[X \leq x] &= \Pr[tX \geq tx] \\ &= \Pr[e^{tX} \geq e^{tx}]. \end{aligned}$$

Using Markov's inequality and the definition of MGF, we get

$$\Pr[e \leq x] \leq \mathbb{E}[e^{tX}] e^{-tx}.$$

Hence,

$$\Pr[X \leq x] \leq e^{-tx} M_X(t) \; \forall t < 0. \tag{4.2}$$

The required inequalities are 4.1 and 4.2.

§ Task D ($\star$) [1+4+1]

Problem Statement

Now take n **independent** Bernoulli random variables $X_1, X_2, \ldots, X_n$ where $\mathbb{E}[X_i] = p_i$. Since each X_i has the same distribution and is independent of all other X_j's, we call the collection of random variables $X_1, \ldots, X_n$ a collection of *independent* and *identically distributed* (i.i.d) random variables.

Let us define a new random variable Y as the sum of these random variables that is, $Y = \sum_{i=1}^{n} X_i$.

1. What is the expectation of Y?
2. Show that

$$P[Y \geq (1+\delta)\mu] \leq \frac{e^{\mu(e^t-1)}}{e^{(1+\delta)t\mu}}$$

3. Show how to improve this bound further by choosing an appropriate value of t.

Take n independent Bernoulli random variables $X_1, X_2, \ldots, X_n$ where $\mathbb{E}[X_i] = p_i$. Since the random variables are given to be identically distributed, let $p = p_i$. Define random variable $Y = \sum_{i=1}^{n} X_i$.

1

The expected value of Y is

$$\begin{aligned}\mathbb{E}[Y] &= \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] \\ &= \sum_{i=}^{n} \mathbb{E}[X_i] \\ &= \sum_{i=1}^{n} p_i \\ &= np.\end{aligned}$$

Hence, $\mu = \mathbb{E}[Y] = np$.

2

Since X_i are independent, the MGF of Y is given by

$$\begin{aligned}M_Y(t) &= \mathbb{E}[e^{tY}] \\ &= \mathbb{E}[e^{t(X_1+\cdots+X_n)}] \\ &= \left(\mathbb{E}[e^{tX}]\right)^n \\ &= (M_X(t))^n\end{aligned}$$

where X is a Bernoulli random variable. Using 1.2 with $z = e^t$, we get

$$\begin{aligned}M_X(t) &= 1 - p + pe^t \\ &= 1 + p(e^t - 1) \\ &\leq e^{p(e^t-1)}\end{aligned}$$

since $e^x - x - 1 \geq 0 \; \forall x \in \mathbb{R}$. Using this result,

$$\begin{aligned}M_Y(t) &\leq \left(e^{p(e^t-1)}\right)^n \\ &\leq e^{np(e^t-1)} \\ &\leq e^{\mu(e^t-1)}.\end{aligned}$$

Now, we will use the inequality 4.1 proven in Task C. For $t > 0$ and any δ,

$$\begin{aligned}\Pr[Y \geq (1+\delta)\mu] &\leq \frac{M_Y(t)}{e^{(1+\delta)t\mu}} \\ &\leq \frac{e^{\mu(e^t-1)}}{e^{(1+\delta)t\mu}}.\end{aligned}$$

Hence, the result is proved.

3

Using calculus, it can be shown that the expression

$$\frac{e^{\mu(e^t-1)}}{e^{(1+\delta)t\mu}}$$

attains its minimum value when $e^t = 1+\delta$, i.e.,

$$\begin{aligned}\Pr[Y \geq (1+\delta)\mu] &\leq \frac{e^{\mu((1+\delta)-1)}}{e^{(1+\delta)\ln(1+\delta)\mu}} \\ &\leq \frac{e^{\mu\delta}}{e^{(1+\delta)\mu\ln(1+\delta)}}.\end{aligned}$$

Hence, a better bound for $\Pr[Y \geq (1+\delta)\mu]$ is

$$\Pr[Y \geq (1+\delta)\mu] \leq e^{\mu[\delta-(1+\delta)\ln(1+\delta)]}, \tag{4.3}$$

where $\delta > 0$. This constraint on δ is needed to ensure that $t > 0$. For $\delta < 0$, we will try to use the other bound 4.2 derived in Task C. For some $t > 0$,

$$\begin{aligned}\Pr[Y \leq \mu(1+\delta)] &= \Pr[-Y \geq -\mu(1+\delta)] \\ &\leq e^{t\mu(1+\delta)}M_{(-Y)}(t).\end{aligned}$$

Now,

$$\begin{aligned}M_{(-Y)}(t) &= \mathbb{E}[e^{-Yt}] \\ &= \left(\mathbb{E}[e^{-Xt}]\right)^n \\ &= (1-p+pe^{-t})^n \\ &\leq e^{np(e^{-t}-1)} \\ &\leq e^{\mu(e^{-t}-1)}.\end{aligned}$$

Hence,

$$\Pr[Y \leq \mu(1+\delta)] \leq e^{\mu[t(1+\delta)+e^{-t}-1]}.$$

Using calculus, we can see that the RHS has its minima when $t = -\ln(1+\delta) > 0$, where $\delta > -1$. The inequality must hold for this t too. Hence,

$$\begin{aligned}&\Pr[Y \leq \mu(1+\delta)] \leq e^{\mu[\delta-(1+\delta)\ln(1+\delta)]} \text{ for } -1 < \delta < 0 \\ \implies &\Pr[Y \leq \mu(1-\delta)] \leq e^{\mu[-\delta-(1-\delta)\ln(1-\delta)]} \text{ for } 0 < \delta < 1.\end{aligned} \tag{4.4}$$

Now, for $0 \leq \delta \leq 1$, we can see that

$$-\delta-(1-\delta)\ln(1-\delta) < \delta-(1+\delta)\ln(1+\delta).$$

Hence, using 4.3 and 4.4, for $0 < \delta < 1$,

$$\Pr[|Y-\mu| \geq \delta\mu] \leq e^{\mu[-\delta-(1-\delta)\ln(1-\delta)]}. \tag{4.5}$$

Finally, equations 4.3, 4.4 and 4.5 provide bounds on Y.

The resulting best bound for $P[Y \geq (1+\delta)\mu]$ is called a Chernoff bound and is an example of a *concentration* theorem - it can be shown that most of Y's probability density is concentrated about μ.

Chernoff bounds are related to the very useful *Central Limit Theorem* and also play critical roles in the analysis of randomized algorithms and the theory of machine learning. They are thus considered a cornerstone of probability theory. It is understood that everyone studying probability must have seen a Chernoff bound - now you know!

§ Task E [4]

Problem Statement

Another important theorem, especially important for estimation using samples: *the weak law of large numbers* (WLLN). We shall try to prove it in this task using the Chernoff bound.

Theorem 6. *Let $X_1, \ldots, X_n$ be i.i.d. Bernoulli random variables with each having mean μ. We define $A_n = \frac{\sum_{i=1}^{n} X_i}{n}$. Then for all $\epsilon > 0$, we have*

$$\lim_{n \longrightarrow \infty} \mathrm{P}[|A_n - \mu| > \epsilon] = 0$$

Essentially, the average of the variables has to be roughly constant at the value μ - it takes on any other value with a probability approaching 0. Intuitively, if you keep sampling from the identical distributions and add up all of them, deviations left of the mean are canceled by deviations right of the mean. The net result is that the sum is always roughly the same - $n\mu$, from which it follows that the mean is always roughly μ.

Prove WLLN using the Chernoff bound from Task D. If you did not solve Task D, you may provide proof using just the Chebyshev inequality. However, if you did solve it, you should use the Chernoff bound obtained from Task D to prove WLLN.

Continuing from the previous task, define $A_n = \frac{Y}{n}$. Let μ be the expected value of A_n. Then, the expected value of Y is $n\mu$. Using the bound 4.5 for Bernoulli random variable, for $0 < \delta < 1$,

$$\Pr[|nA_n - n\mu| \geq n\delta\mu] \leq e^{n\mu[-\delta-(1-\delta)\ln(1-\delta)]}$$
$$\implies \Pr[|A_n - \mu| \geq \delta\mu] \leq e^{n\mu[-\delta-(1-\delta)\ln(1-\delta)]}.$$

Now, for $0 < \delta < 1$,

$$-\delta - (1-\delta)\ln(1-\delta) < 0.$$

Hence,

$$\lim_{n\to\infty} \Pr[|A_n - \mu| \geq \delta\mu] \leq \lim_{n\to\infty} e^{n\mu[-\delta-(1-\delta)\ln(1-\delta)]}$$
$$= 0.$$

For $\delta > 1$ and $0 < \delta_1 < 1$, clearly

$$\lim_{n\to\infty} \Pr[|A_n - \mu| \geq \delta\mu] \leq \lim_{n\to\infty} \Pr[|A_n - \mu| \geq \delta_1\mu]$$
$$= 0.$$

Hence, $\forall \epsilon > 0$,

$$\lim_{n\to\infty} \Pr[|A_n - \mu| > \epsilon] = 0.$$

This proves the weak law of large numbers for Bernoulli random variable.

PROBLEM 5

A Pretty "Normal" Mixture

We have been looking at Gaussian (normal) random variables and their manipulation. Now, we shall take many such Gaussians and mix them!

Definition 7 (GMM). *A Gaussian Mixture Model (GMM) is a random variable defined in terms of K Gaussian random variables and follows the PDF*

$$\mathrm{P}[X = x] = \sum_{i=1}^{K} p_i P[X_i = x],$$

where each $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ *is a Gaussian random variable with mean* μ_i *and variance* σ_i^2 *for all* $i \in \{1, 2, \cdots, K\}$. *Moreover, each* $p_i \geq 0$ *and* $\sum_{i=1}^{K} p_i = 1$.

§ Task A [2]

Problem Statement

To sample from a GMM's distribution, we use the following algorithm:

1. First, one of the Gaussian variables X_i is randomly chosen (or effectively, an index i is chosen) according to the PMF $\{p_1, p_2, \ldots, p_k\}$. That is, i or X_i is chosen in this step with probability p_i.
2. Next, we sample a value from the chosen Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$ and this is the final value sampled from the GMM.

Suppose the output of the algorithm is a random variable $\mathcal{A}$ with PDF $f_{\mathcal{A}}$ and the PDF of the GMM variable X is f_X. Show that for every $u \in \mathbb{R}$, $f_{\mathcal{A}}(u) = f_X(u)$, that is, indeed, this algorithm samples from the GMM variable's distribution.

Let X_i be the event of choosing X_i as the Gaussian in the first step. The PDF of the algorithm's output can be represented using the Total probability theorem as

$$\begin{aligned}\Pr[X = x] &= \sum_{i=1}^{K} \Pr\left[(X = x) | X_i\right] \cdot \Pr[X_i] \\ &= \sum_{i=1}^{K} p_i \Pr[X_i = x],\end{aligned}$$

since $\Pr[X_i] = p_i$. Hence, if f_{X_i} is the PDF of X_i, then

$$\begin{aligned} f_{\mathcal{A}} &= \sum_{i=1}^{K} p_i f_{X_i} \\ &= f_X. \end{aligned}$$

§ Task B [1+2+2]

Problem Statement

Let X be a GMM sampled by the method described above, where each Gaussian $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ is chosen with a probability $p_i \geq 0$. Then compute

1. $\mathbb{E}[X]$.
2. $\text{var}[X]$.
3. The MGF $M_X(t)$ of X.

1. Expected Value $\mathbb{E}[X]$

To find the expected value of X, where X is a Gaussian Mixture Model, we use the law of total expectation:

$$\begin{aligned} \mathbb{E}[X] &= \sum_{i=1}^{K} \mathbb{E}\left[X|X_i\right] \Pr[X_i] \\ &= \sum_{i=1}^{K} \mathbb{E}[X_i] \cdot p_i. \end{aligned}$$

Since $\mathbb{E}_{X_i} = \mu_i$, the expected value of X is

$$\mathbb{E}[\mathbb{X}] = \sum_{i=1}^{K} p_i \mu_i.$$

2. Variance Var$[X]$

We know that the variance of any random variable Y is related to the expected value by

$$\text{Var}[Y] = \mathbb{E}[Y^2] - (\mathbb{E}[Y])^2.$$

To find variance of X, where X is GMM, we again use the law of total expectation, but coupled with the above relation:

$$\begin{aligned} \text{Var}[X] &= \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \\ &= \sum_{i=1}^{K} \mathbb{E}[X_i^2] \Pr[X_i] - \left(\sum_{i=1}^{K} p_i \mu_i\right)^2 \\ &= \sum_{i=1}^{K} p_i \left((\mathbb{E}[X_i])^2\right) - \left(\sum_{i=1}^{K} p_i \mu_i\right)^2 \\ &= \sum_{i=1}^{K} p_i \left(\text{Var}[X_i] + (\mathbb{E}[X_i])^2\right) - \left(\sum_{i=1}^{K} p_i \mu_i\right)^2. \end{aligned}$$

Since $\text{Var}[X_i] = \sigma_i^2$, the variance of X is:

$$\text{Var}[X] = \sum_{i=1}^{K} p_i(\sigma_i^2 + \mu_i^2) - \left(\sum_{i=1}^{K} p_i \mu_i\right)^2.$$

3. Moment-Generating Function (MGF) $M_X(t)$

The moment-generating function (MGF) of X is

$$M_X(t) = \mathbb{E}[e^{tX}].$$

Using the law of total expectation,

$$M_X(t) = \sum_{i=1}^{K} p_i \cdot \mathbb{E}[e^{tX_i}].$$

For a Gaussian random variable $X_i = \mathcal{N}(\mu_i, \sigma_i^2)$, the MGF is:

$$M_{X_i}(t) = \exp\left(t\mu_i + \frac{1}{2}t^2\sigma_i^2\right)$$

Therefore, the MGF of X is

$$M_X(t) = \sum_{i=1}^{K} p_i \exp\left(t\mu_i + \frac{1}{2}t^2\sigma_i^2\right).$$

§ Task C [1+1+2+2+1+1]

Problem Statement

We may be inclined to think "Isn't this just a weighted sum of Gaussians?" Let us now prove (or disprove) this property. Let us take a random variable Z to be a weighted sum of k **independent** Gaussian random variables,

$$Z = \sum_{i=1}^{k} p_i X_i,$$

where $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. For our new random variable Z find the same expressions as in Task B:

1. $\mathbb{E}[Z]$.
2. $\text{var}[Z]$.
3. The PDF $f_Z(u)$ of Z
4. The MGF $M_Z(t)$ of Z.
5. What can you conclude? Do X and Z have the same properties?
6. What distribution does Z seem to follow?

We are given a random variable Z which is a weighted sum of k independent Gaussian random variables:

$$Z = \sum_{i=1}^{K} p_i X_i$$

where $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$.

1. Expected Value of Z

The expected value $\mathbb{E}[Z]$ is

$$\mathbb{E}[Z] = \mathbb{E}\left[\sum_{i=1}^{K} p_i X_i\right]$$

Using the linearity of expectation,

$$\mathbb{E}[Z] = \sum_{i=1}^{K} p_i \mathbb{E}[X_i]$$

Since X_i are Gaussian, we have $\mathbb{E}[X_i] = \mu_i$. Hence,

$$\mathbb{E}[Z] = \sum_{i=1}^{K} p_i \mu_i. \tag{5.1}$$

2. Variance of Z

The variance Var$[Z]$is

$$\text{Var}[Z] = \text{Var}\left(\sum_{i=1}^{K} p_i X_i\right)$$

Since X_i are independent:

$$\begin{aligned}\text{Var}[Z] &= \sum_{i=1}^{K} \text{Var}[p_i X_i] \\ &= \sum_{i=1}^{K} p_i^2 \text{Var}[X_i].\end{aligned}$$

Now $\text{Var}[X_i] = \sigma_i^2$. Hence,

$$\text{Var}[Z] = \sum_{i=1}^{K} p_i^2 \sigma_i^2. \tag{5.2}$$

3. PDF of Z

Using induction, we will show that Z has a Gaussian distribution.

Claim: For independent Gaussian random variables $X_1, \ldots, X_K$, the random variable $\sum_{i=1}^{K} p_i X_i$ is also a Gaussian random variable.

Proof: We will prove the result using induction. The **base case** is $K = 1$, when the claim holds by our assumptions. Now, we will try to show that: If $Y = \sum_{i=1}^{r-1} p_i X_i$ is a Gaussian, then $\sum_{i=1}^{r} p_i X_i$ is also a Gaussian.

Let the pdf of Y be f_Y, where

$$f_Y = \frac{c_Y}{\sqrt{2\pi}\sigma_Y} \exp\left(\frac{-(Y-\mu_Y)^2}{2\sigma_Y^2}\right).$$

Then the pdf of $Y' = Y + p_r X_r$ is

$$\begin{aligned}f_{Y'} &= \int_{-\infty}^{\infty} f_Y(Y'-x) \cdot p_r \cdot f_{X_r}(x) dx \\ &= \int_{-\infty}^{\infty} \frac{c_Y}{\sqrt{2\pi}\sigma_Y} \exp\left(\frac{-(Y'-x-\mu_Y)^2}{2\sigma_Y^2}\right) \cdot \frac{p_r}{\sqrt{2\pi}\sigma_r} \exp\left(\frac{-(x-\mu_r)^2}{2\sigma_r^2}\right) dx \\ &= \frac{c_Y p_r}{2\pi\sigma_Y\sigma_r} \int_{-\infty}^{\infty} \exp\left(\frac{(x-\mu_r)^2}{2\sigma_r^2} - \frac{(Y'-x-\mu_Y)^2}{2\sigma_Y^2}\right) dx.\end{aligned}$$

Simplifying this, and using the Gaussian integral, we get

$$f_{Y'} = \frac{c_Y p_r}{\sqrt{2\pi(\sigma_X^2+\sigma_Y^2)}} \exp\left(-\frac{(Y'-(\mu_X+\mu_Y)^2)}{2(\sigma_X^2+\sigma_Y^2)}\right).$$

which is a Gaussian. Hence, by induction, the pdf for Z will be a gaussian. Let the pdf of Z be

$$f_Z(u) = \frac{c}{\sqrt{2\pi}\sigma_Z} \exp\left(-\frac{(u-\mu_Z)^2}{2\sigma_Z^2}\right).$$

Since Z is a Gaussian random variable, $c = 1$, $\sigma_Z^2 = \text{Var}[Z]$ and $\mu_Z = \mathbb{E}[Z]$. Substituting these values from 5.1 and 5.2, the PDF of Z is

$$f_Z(u) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right) \tag{5.3}$$

where $\mu = \sum_{i=1}^K p_i\mu_i$ and $\sigma = \sqrt{\sum_{i=1}^K p_i^2\mu_i^2}$.

4. MGF of Z

The Moment Generating Function (MGF) of a Gaussian random variable X_i with mean μ_i and variance σ^2 is:

$$M_{X_i}(t) = \exp\left(\mu_i t + \frac{1}{2}\sigma_i^2 t^2\right)$$

For $Z = \sum_{i=1}^K p_i X_i$, the MGF is

$$\begin{aligned} M_Z(t) &= \mathbb{E}\left[\exp(tZ)\right] \\ &= \mathbb{E}\left[\exp\left(t\sum_{i=1}^K p_i X_i\right)\right]. \end{aligned}$$

Since X_i are independent,

$$\begin{aligned} M_Z(t) &= \prod_{i=1}^K \mathbb{E}\left[\exp\left(tp_iX_i\right)\right] \\ &= \prod_{i=1}^K \exp\left(p_i\mu_i t + \frac{1}{2}(p_i^2\sigma_i^2)t^2\right) \\ &= \exp\left(\sum_{i=1}^K p_i\mu_i t + \frac{1}{2}\sum_{i=1}^K p_i^2\sigma_i^2 t^2\right). \end{aligned}$$

Hence, the MGF of Z is

$$M_Z(t) = \exp\left(t\sum_{i=1}^K p_i\mu_i + \frac{1}{2}t^2\sum_{i=1}^K p_i^2\sigma_i^2\right).$$

5. Conclusion

We see that X and Z differ in their variance and MGF, and are hence not the same random variable. X is not a Gaussian in general, while Z always has a Gaussian distribution as seen from its PDF.

6. Distribution of Z

As seen in 5.3, Z follows a Gaussian distribution. Specifically, Z is distributed as:

$$Z \sim \mathcal{N}\left(\sum_{i=1}^K p_i\mu_i, \sum_{i=1}^K p_i^2\sigma_i^2\right)$$

§ Task D (B) [3]

Problem Statement

Theorem 8. *For a random variable X, if it is*

1. *either finite and discrete,*
2. *or if it is continuous and its MGF $\phi_X(t)$ is known for some (non-infinitesimal) interval,*

then its MGF and PDF ***uniquely*** *determine each other.*

Prove the above theorem for the finite discrete case.
What can you now conclude about X and Z? Also, explain logically why this may be the case.

We will show that for two random variables X and Y:

– If their PDFs are the same, they have the same MGF.

– If their MGFs are the same, they have the same PDF.

Discrete and Finite random variables

Let the PDFs of X and Y be f_X and f_Y respectively. Let the MGFs be M_X and M_Y respectively.

Same PDF implies same MGF: Let the common support of X and Y be $\{a_1, a_2, \ldots, a_k\}$. Then,

$$\begin{aligned} M_X(t) &= \mathbb{E}[e^{tx}] \\ &= \sum_{i=1}^{k} \Pr[X = a_i]e^{ta_i} \\ &= \sum_{i=1}^{k} \Pr[Y = a_i]e^{ta_i} \\ &= M_Y(t). \end{aligned}$$

Hence, the MGFs are identical.

Same MGF implies same PDF: Let the support of X be $\{a_1, \ldots, a_k\} \cup \{c_1, \ldots, c_l\}$, and the support of Y be $\{b_1, \ldots, b_m\} \cup \{c_1, \ldots, c_l\}$, where $a_i \neq b_j$. Then

$$M_X(t) = \sum_{i=1}^{k} \Pr[X = a_i]e^{ta_i} + \sum_{i=1}^{l} \Pr[X = c_i]e^{tc_i}$$

and

$$M_Y(t) = \sum_{i=1}^{m} \Pr[Y = b_i]e^{tb_i} + \sum_{i=1}^{l} \Pr[X = c_i]e^{tc_i}.$$

Since $M_X(t) = M_Y(t)$,

$$\sum_{i=1}^{k} \Pr[X = a_i]e^{ta_i} + \sum_{i=1}^{l} (\Pr[X = c_i] - \Pr[Y = c_i])e^{tc_i} - \sum_{i=1}^{m} \Pr[Y = b_i]e^{tb_i} = 0.$$

This holds for infinitely many values of t. Now, the above equation is a system of linear equations in the variables $e^{ta_i}, e^{tb_i}, e^{tc_i}$. Since there are more than $(k + l + m)$ roots of the system, the coefficients must all be 0. Hence,

$$\begin{aligned} &\Pr[X = a_i] = 0 \\ &\Pr[X = c_i] = \Pr[Y = c_i] \forall i \in \{1, \ldots, l\} \\ &\Pr[Y = b_i] = 0. \end{aligned}$$

Hence, X and Y have the same PDFs.