ECEN321: Engineering Statistics
Lab 1 - Densities - Submission

Daniel Eisen : 300447549
March 23, 2020

1 Introduction

This exercise was centred on the exploration of three major probability densities, the Normal (Gaussian), Student t, and Chi-squared. It was done through the generating random samples intended to be representative of these distributions and comparing the sample statistics and features to that of the theoretical ideal. As a motive of showing the "real-world" effects of sampling a greater population and the variation from the **true parameters.**

2 Theory

Part 1 Normal

This is a type of continuous distribution for a ℝeal random variable. The general form of its probability density function is described right:

$$p(x;\mu,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

With the distributionsCDF described as
Such that erf describes the "error function".

$$\frac{1}{2}\left[1+\operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

Part 2 Student t

The t-distribution is a continuous probability distribution that arises when estimating the mean of a normally distributed population in situations where the sample size is small and the population standard deviation is unknown. It is described by

$$p(x)=\frac{\Gamma((\nu+1)/2)}{\sqrt{\nu\pi}\,\Gamma(\nu/2)\,(1+x^2/\nu)^{(\nu+1)/2}}$$

And a CDF

$$\frac{1}{2}+x\Gamma\left(\frac{\nu+1}{2}\right)\times\frac{{}_2F_1\left(\frac{1}{2},\frac{\nu+1}{2};\frac{3}{2};-\frac{x^2}{\nu}\right)}{\sqrt{\pi\nu}\,\Gamma\left(\frac{\nu}{2}\right)}$$

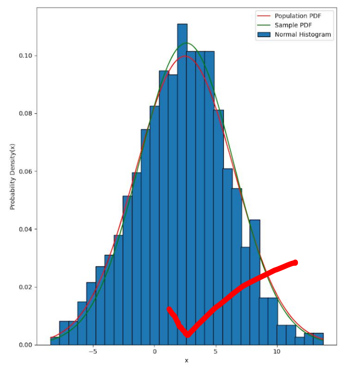
Part 3 Chi Squared

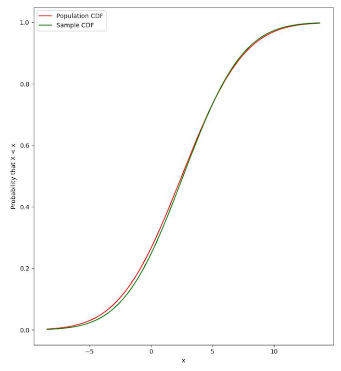
The chi-square distribution with k degrees of freedom is the distribution of a sum of the squares of k independent standard normal random variables.

$$p(x)=\frac{(1/2)^{\nu/2}}{\Gamma(\nu/2)}x^{\nu/2-1}e^{-x/2}$$

3 Results

Part 1 Normal





Sample was set up with a mean of 2.5, and std of 4.

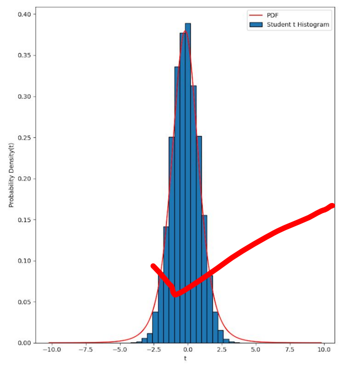
Figure right shows the output of part 1.

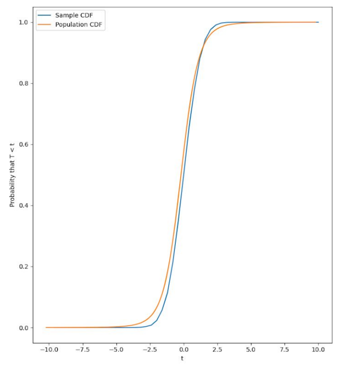
The histogram of the sample, with the computed from sample pdf and theoretical pdf overlayed.

CDF’s are computed directly from the equation from both ideal and sample statistics

There are variations that can be seen in both pdf and cdf in the characteristics of the generated samples and the ideal.

Part 2 Student t





Sample was set up with a mean of 2.5, and std of 4. And used to gen a 2d matrix from which the t sample was generated:

elements →
```
t = (np.mean(np.transpose(sample)[i]) - mean) /
    sample_std * np.sqrt(M)
```

Figure right shows the output of part 2.

The histogram of the sample, with the computed from sample pdf overlayed. The pdf was computed using the `stats.t.pdf()` function from Python's SciPy library.

The theoretical CDF was computed from the `stats.t.cdf()` and the sample CDF was computed using a sorted cumulative summation.

Differences between the sample and population, seen both in the pdf and cdf. With the histogram being more centrally concentrated, the cdf not rolling off and smoothly.

Part 3 Chi Squared

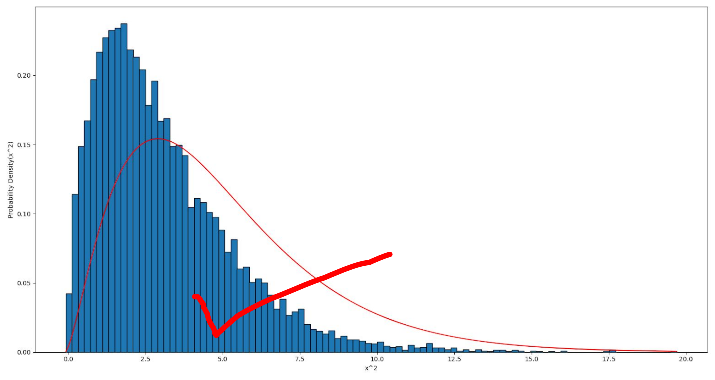


Figure right shows the output of part 3.

Sample generated from the same matrix as part 2, with each element:
```
val = np.var(np.transpose(sample)[i]) * ((M - 1) / var)
```

The pdf was computed using the `stats.chi2.pdf()` function from Python's SciPy library.

Appendices

Part 1 Normal

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import erf

def pdf(mu, sig, x):
   return (1 / (np.sqrt(2 * np.pi * sig ** 2))) * np.exp(-((x - mu) ** 2) / (2 * sig ** 2))

mean = 2.5
var = 16
std = 4

sample = mean + std * (np.random.randn(1000))  # such that the Gaussian Distribution is the population

sample_mean = np.mean(sample)
sample_var = np.var(sample)
sample_std = np.std(sample)

print("Mean: {}\nVar: {}\nStd: {}".format(sample_mean, sample_var, sample_std))

hist, bins = list(np.histogram(sample, 30))
bin_w = bins[1] - bins[0]
hist_scaled = hist / 1000
hist_pd = hist_scaled / bin_w

x = np.linspace(bins[0], bins[-1], 1000) - bin_w / 2

cdf_pop = (1 / 2) * (1 + erf((x - 2.5) / (4 * np.sqrt(2))))
cdf_samp = (1 / 2) * (1 + erf((x - sample_mean) / (sample_std * np.sqrt(2))))

ax1 = plt.subplot(121)
plt.bar(bins[:-1], hist_pd, width=0.81, edgecolor="black")
plt.plot(x, pdf(2.5, 4, x), 'r')
plt.plot(x, pdf(sample_mean, sample_std, x), 'g')
plt.ylabel("Probability Density(x)")
plt.xlabel("x")
plt.legend(["Population PDF", "Sample PDF", "Normal Histogram"])

ax2 = plt.subplot(122)
plt.plot(x, cdf_pop, 'r')
plt.plot(x, cdf_samp, 'g')
plt.ylabel("Probability that X < x")
plt.xlabel("x")
plt.legend(["Population CDF", "Sample CDF"])

plt.show()
```

Part 2 Student t

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

M = 5
N = 10000

var = 16
std = 4
mean = 2.5

sample = mean + std * (np.random.randn(M, N))
sample_std = np.std(sample)

student_t = []

for i in range(N):
   t = (np.mean(np.transpose(sample)[i]) - mean) / sample_std * np.sqrt(M)
   student_t += [t]

hist, bins = np.histogram(student_t, bins=50, range=[-10, 10])
bin_w = bins[1] - bins[0]
hist_norm = hist / N
hist_plt = hist_norm / bin_w

plt.subplot(121)
plt.bar(bins[:-1], hist_plt, width=0.4, edgecolor="black")

x = np.linspace(bins[0], bins[-1], 1000) - bin_w / 2
plt.plot(x, stats.t.pdf(x + bin_w / 2, M), 'r')
plt.ylabel("Probability Density(t)")
plt.xlabel("t")
plt.legend(["PDF", "Student t Histogram"])

plt.subplot(122)
plt.plot(bins[1:], np.cumsum(hist_norm))
plt.plot(x, stats.t.cdf(x + bin_w / 2, M))
plt.ylabel("Probability that T < t")
plt.xlabel("t")
plt.legend(["Sample CDF", "Population CDF"])

plt.show()
```

Part 3 Chi Squared

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

M = 5
N = 10000

var = 16
std = 4
mean = 2.5

sample = mean + std * (np.random.randn(M, N))

chi_squared = []
for i in range(N):
   val = np.var(np.transpose(sample)[i]) * ((M - 1) / var)
   chi_squared += [val]

hist, bins = np.histogram(chi_squared, bins=100)
bin_w = bins[1] - bins[0]
hist_norm = hist / N
hist_plt = hist_norm / bin_w

x = np.linspace(bins[0], bins[-1], 1000) - bin_w / 2

plt.bar(bins[:-1], hist_plt, width=0.2, edgecolor="black")
plt.plot(x, stats.chi2.pdf(x + bin_w / 2, M), 'r')
plt.ylabel("Probability Density(x^2)")
plt.xlabel("x^2")

plt.show()
```