ECEN321 : Estimation
Lab 2 Submission

Daniel Eisen : 300447549

May 25, 2020

1 Introduction

This lab is covers methods of fitting real world dataset to theoretical distributions. This necessitates analysing the dataset and estimating the parameters of the distribution that represents the signal. This lab specifically covers the effect of the number of realisations on the precision and accuracy of the parameter estimation; and a numerical parameter estimation of a real world signals.

2 Theory

Part 1

Estimation base on N realisations:

The estimator $\bar{X}$ of the population mean μ is unbiased and its variance is equal to $\frac{\sigma^2}{N}$. So as we sample from a greater N of realisations the variance in the estimated mean will decrease, but even for low N the distribution in these estimations will still be centred on the true mean.

The estimator of the population $sigma^2$ is s^2. Due to these samples being taken from a normal distribution we can say that $\frac{N-1}{\sigma^2}S^2$ is equal to the Chi-squared distribution with $n-1$ degrees of freedom. So as more realisations are taken, N increases, the distribution in of the estimated variance becomes more normally distributed; just as the Chi-squared with increasing degrees of freedom.

Part 2

The samples in the signal are in reality not independent, but they are more dependant the closer they are. So for a signal of larger length such as this we can treat all samples are independent without such a great impact on the analysis, and in parameter estimation.

For the estimation we take the Likelihood function of the Generalised Normal Distribution, which we use the log of due to break down in computer precision and maximise its output for a range of alpha and beta values, repeating this until convergence.

Generalised Normal Distribution

$$f(x;\alpha,\beta,\mu) = \frac{\beta}{2\alpha\Gamma(1/\beta)}e^{-(|x-\mu|\alpha)^\beta}$$

Likelihood Equation

$$L(\alpha,\beta;x) = \prod_{i=1}^{N}\frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})}e^{-(\frac{|x-\mu|}{\alpha})^\beta}$$

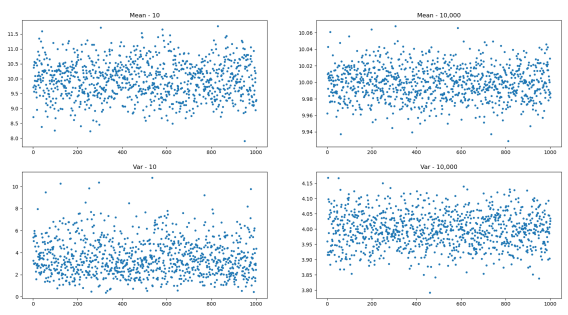
$$-ln(L(\alpha,\beta;x)) = \sum_{i=1}^{N} -ln\left(\frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})}\right) + \sum_{i=1}^{N} -ln(e^{-(\frac{|x-\mu|}{\alpha})^\beta})$$

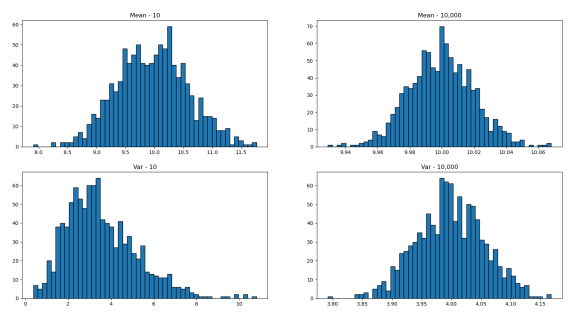
$$= N\left(ln\left(\frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})}\right)\right) - \sum_{i=1}^{N}\left(\frac{|x-\mu|}{\alpha}\right)^\beta$$

$$= N(ln(\beta)) - ln(2\alpha) - ln(\Gamma(\frac{1}{\beta})) - \sum_{i=1}^{N}\left(\frac{|x-\mu|}{\alpha}\right)^\beta$$

3 Results

Part 1

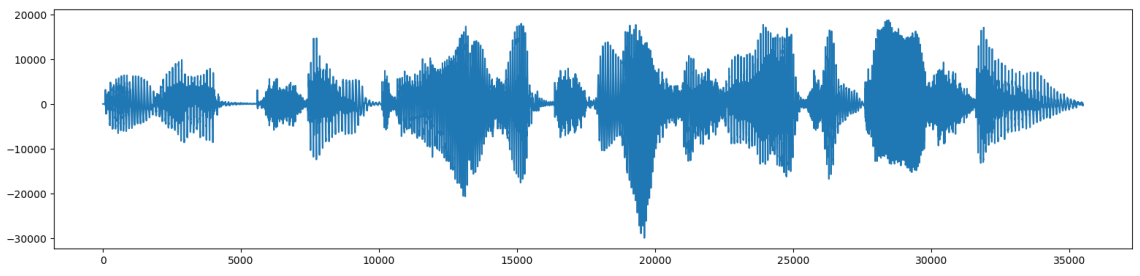




The figures above show the results of estimating the mean and variance from Gaussian RV, at 10 and 10,000 realisations. With greater N of realisations the mean is seen to converge symmetrically around the parameter where the variance is cantered left and tails off, ie right skewed.

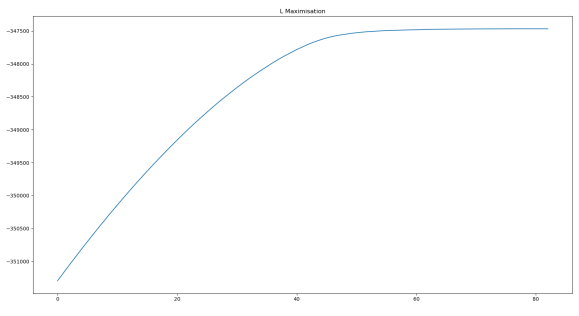
Part 2

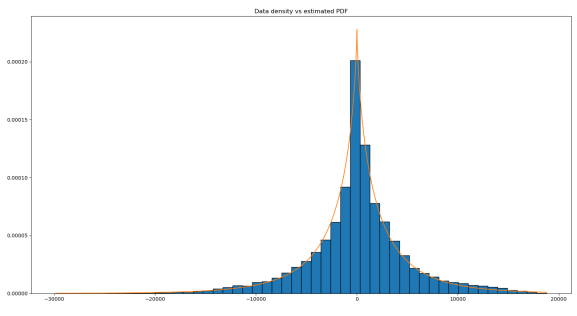
Imported sound data from Speech.pcm



Implementing the theory (outlined above) with python to maximise the likelihood equation and repeating until convergence produce the final values:

$$\alpha = 0.7119 \qquad \beta = 1755.1$$





The figure right show the maximisation of the ln(likelihood) equation as the algorithm progressed and the figure right show the density histogram of the signal overlaid with the computed theoretical pdf using the estimated parameters; showing the success of this estimation method.
Note: The outer loop only iterated 83 times before convergence.

Appendix

Part 1

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

mu = 10
sigma = 2
var = 4

ten = mu + (sigma * (np.random.randn(10, 1000)))
tenThousand = mu + (sigma * (np.random.randn(10000, 1000)))

ten_mean = np.mean(ten, axis=0)
tenThousand_mean = np.mean(tenThousand, axis=0)

ten_var = np.var(ten, axis=0)
tenThousand_var = np.var(tenThousand, axis=0)

index = np.linspace(1, 1000, 1000)
plt.figure(1)

ax1 = plt.subplot(2, 2, 1)
plt.scatter(index, ten_mean, marker=".")
plt.title("Mean - 10")

ax2 = plt.subplot(2, 2, 2)
plt.scatter(index, tenThousand_mean, marker=".")
plt.title("Mean - 10,000")

ax3 = plt.subplot(2, 2, 3)
plt.scatter(index, ten_var, marker=".")
plt.title("Var - 10")

ax4 = plt.subplot(2, 2, 4)
plt.scatter(index, tenThousand_var, marker=".")
plt.title("Var - 10,000")
plt.show()

plt.figure(2)

ax1 = plt.subplot(2, 2, 1)
plt.hist(ten_mean, bins=50, edgecolor="black")
plt.title("Mean - 10")

ax2 = plt.subplot(2, 2, 2)
plt.hist(tenThousand_mean, bins=50, edgecolor="black")
plt.title("Mean - 10,000")

ax3 = plt.subplot(2, 2, 3)
plt.hist(ten_var, bins=50, edgecolor="black")
plt.title("Var - 10")

ax4 = plt.subplot(2, 2, 4)
plt.hist(tenThousand_var, bins=50, edgecolor="black")
plt.title("Var - 10,000")
plt.show()
```

Part 2

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import gammaln, gamma

with open('../321_lab3/Speech.pcm', 'rb') as f:
    buf = f.read()
    data = np.frombuffer(buf, dtype='int16')

data = data.astype("int64")
plt.figure()
plt.plot(data)
N = data.size
sample_var = np.var(data)
mu = 0

alpha = np.sqrt((2 * sample_var))
beta = 2  # init beta


max_L = N * (np.log(beta) - np.log(2 * alpha) - gammaln(1 / beta)) - np.sum(
    np.power((np.abs(data) / alpha), beta))  # first result
opt_alpha = alpha
opt_beta = beta  # init max
prev_opt_beta = opt_beta
prev_opt_alpha = opt_alpha
l = []

while False:
    for beta in np.linspace(0.01, beta+1, 1000):
        beta_r = N * (np.log(beta) - np.log(2 * alpha) - gammaln(1 / beta)) - np.sum(np.power((
                                                    np.abs(data) / alpha), beta))
        if beta_r > max_L:
            max_L = beta_r
            opt_beta = beta
    beta = opt_beta
    for alpha in np.linspace(alpha - 100, alpha + 100, 1000):
        alpha_r = N * (np.log(beta) - np.log(2 * alpha) - gammaln(1 / beta)) - np.sum(np.power((
                                                    np.abs(data) / alpha), beta))
        if alpha_r > max_L:
            max_L = alpha_r
            opt_alpha = alpha
    l += [max_L]
    if prev_opt_alpha == opt_alpha:
        alpha = opt_alpha
        beta = opt_beta
        break
    alpha = opt_alpha
    beta = opt_beta
    prev_opt_alpha = opt_alpha
    print(opt_beta)
    print(opt_alpha)

plt.figure()
plt.title("L Maximisation")
plt.plot(np.array(l))
print(beta)
print(alpha)

plt.figure()
x = np.linspace(np.min(data), np.max(data), data.size)
pdf = (beta / (2 * alpha * gamma(1 / beta))) * np.exp(-1 * np.power(np.abs(x) / alpha, beta))
plt.hist(data, bins=50, edgecolor="black", density=True)
plt.plot(x, pdf)
plt.title("Data density vs estimated PDF")
```