

# Computer Vision – Programming Project 1

王順興 0210184

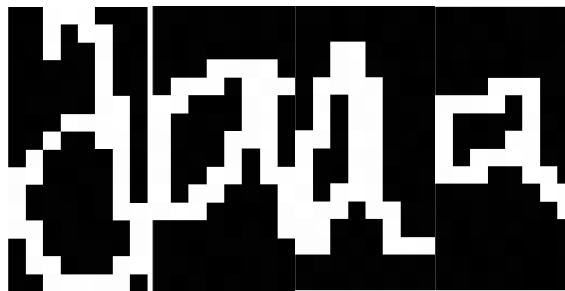
## Introduction

In this project, I compare the results of Gaussian MLE and Gamma MLE, and find the cause of the difference between the two. I am using Matlab to process, experiment, and examine the result.

The test for the two distribution are English alphabet classification.

### i. The Dataset

The dataset I am using can be found [here](http://ai.stanford.edu/~btaskar/ocr/)(<http://ai.stanford.edu/~btaskar/ocr/>). Each sample is a 16x8 binary image (0 and 1) of an English alphabet (letter) taken from a written word (as opposed to test subjects writing individual letters), this makes the recognition more difficult.



Some sample of 'a'

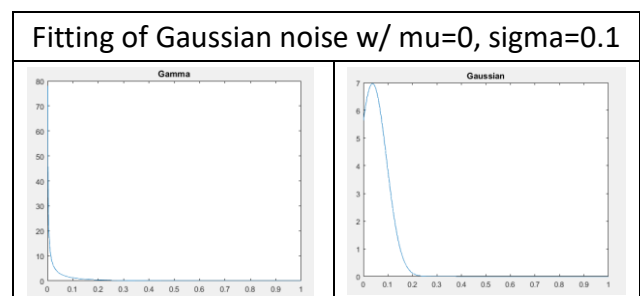
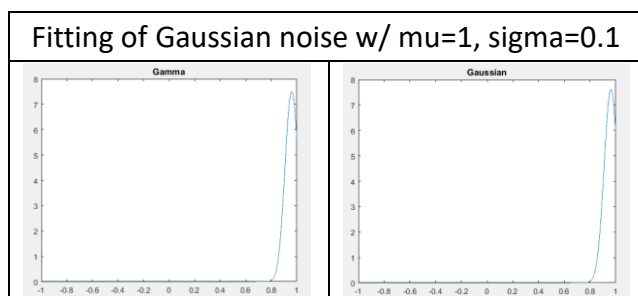
### ii. Motivation

Originally, I was going to implement Gaussian MLE and MoG, then compare the two results. But, when 賈恩宇 told me that he found a paper regarding the usage of Bernoulli distribution in binary image, I wondered if Gamma distribution will be a good distribution to fit for an noise riddled (added by me) image.

And so I'll be comparing the letter classification results of Gaussian MLE and Gamma MLE.

### iii. Why Gamma Distribution?

Because Gamma is more flexible; Gamma can do whatever Gaussian can, and some times better!



This makes me more curious as to why Gamma Distribution aren't used as much as Gaussian.

## Method

### i. Gaussian Maximum Likelihood

Gaussian Maximum Likelihood is pretty straight forward, since its means and covariances have closed form:

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n \quad \text{and} \quad \hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \hat{\mu})^2$$

Compare the posteriors generated by each class to classify a sample:

$$Pr(w = 1|\mathbf{x}) = \frac{Pr(\mathbf{x}|w = 1)Pr(w = 1)}{\sum_{k=0}^1 Pr(\mathbf{x}|w = k)Pr(w = k)}$$

### ii. Gamma Maximum Likelihood

Gamma Distribution, similar to Gaussian, has two parameters:  $k$ ,  $\theta$

$$\frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}}$$

The derivation of these parameters ML fitting can be found in *Estimating a Gamma distribution* by Thomas P. Minka, but his derivation is done in  $a$  and  $b$ , as in the alternate form of Gamma, so these results are taken from Wikipedia.

$\theta$  does have a closed form, though it relies on  $k$ :

$$\hat{\theta} = \frac{1}{kN} \sum_{i=1}^N x_i$$

$k$  does not have a closed form. It can be found by Newton's method:

Let

$$s = \ln\left(\frac{1}{N} \sum_{i=1}^N x_i\right) - \frac{1}{N} \sum_{i=1}^N \ln(x_i)$$

the initial approximation of  $k$  can be found:

$$k \approx \frac{3 - s + \sqrt{(s - 3)^2 + 24s}}{12s}$$

We can update it by Newton's method:

$$k \leftarrow k - \frac{\ln(k) - \psi(k) - s}{\frac{1}{k} - \psi'(k)}$$

From the paper aforementioned,  $k$  converges in about 4 iterations.

What about likelihood function? As the ML parameters are derived from univariate Gamma Distribution, the likelihood function of a sample is simply the multiplication of univariate Gamma of each pixel.

$$\text{Likelihood: } p(\mathbf{x}|\mathbf{w}) = \prod_d^D p(x_d|\mathbf{w})$$

I did search for a multivariate form of Gamma Distribution, none can be found, and the math required to derive one are simply beyond me. But apart from the issue of having to multiply multiple terms, the accuracy is not being affected by the lack of a multivariate form; as in the Multivariate Gaussian, we still only use diagonal terms of covariance matrix.

### iii. Technical Details

#### When calculating the Gaussian Likelihood function:

The underflow/overflow issue when calculating Likelihood function is mitigated by using smaller data dimension and normalizing the range from 0 ~ 255 to 0 ~ 1.

The determinant of covariance matrix(sigma) will become 0 when one pixel has the same value across all dataset. This can be solved by adding some noise to each image.

#### When calculating $k$ in Gamma Maximum Likelihood:

Notice the form of  $s$

$$s = \ln\left(\frac{1}{N} \sum_{i=1}^N x_i\right) - \frac{1}{N} \sum_{i=1}^N \ln(x_i)$$

The nature log of each pixel is calculate. What is  $\ln(0)$ ? God knows. So add a tiny value (I chose 0.001) to each pixel that equals to 0.

#### The size of each class in the dataset is different:

This is very trivial, but anyway this is the size of each class in my dataset:

[4034 1284 2114 1442 4955 921 2472 861 4913 189 909 3140 1602 5024 3897 1377 341 2673 1394 2136 2562 664 520 413 1221 1094]

Of all 52k samples: 4k are 'a', 1.2k are 'b'... The largest set has 5k sample while some only have few hundreds. So when storing these data, use cell instead of matrix.

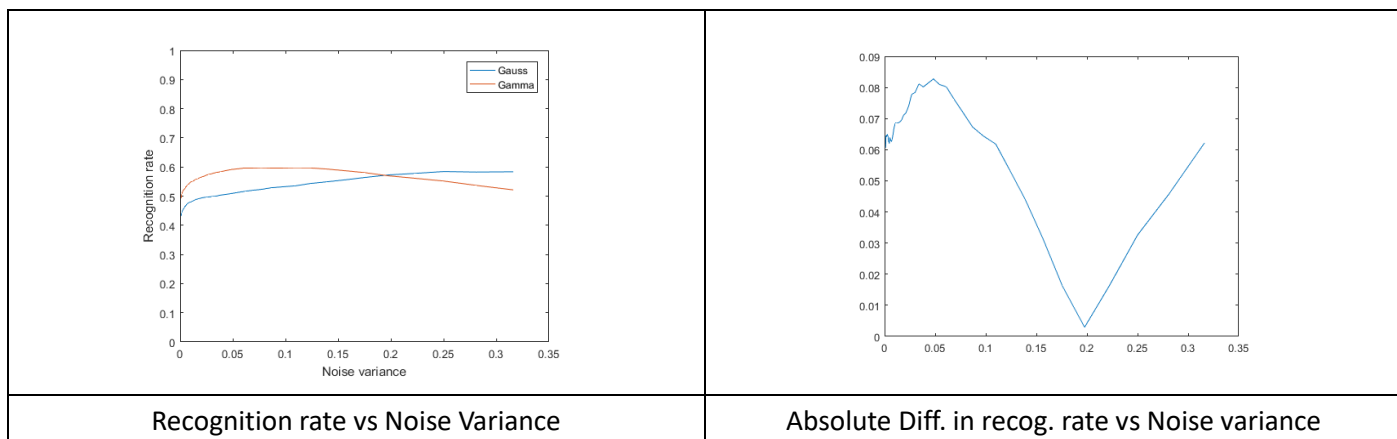
## Experiment

The goal is to compare the accuracy of Gaussian and Gamma MLE, and try to find out what causes the difference.

In this comparison, half of the data from each class are used in training; the min(remaining, 200) are used in testing.

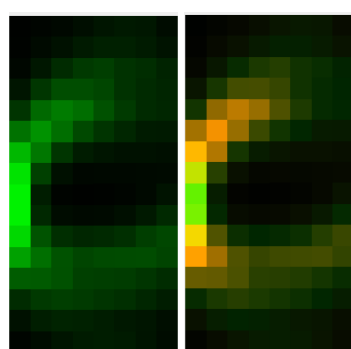
For each test, Gaussian Noise is applied to every image. The noise has a mean of 0 and variance from 0.001 to 0.3. Each pixel is limited between 0 and 1 after the noise is applied.

Both estimator assumes that each pixel are independent.



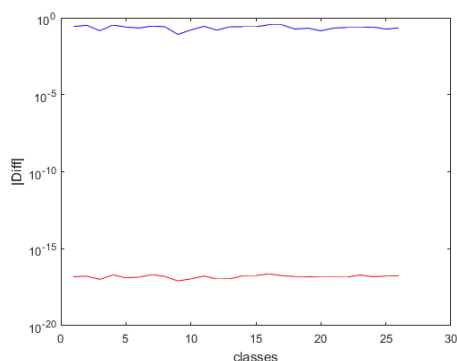
The first thing I compare is the Recognition rate. When there is little noise, Gamma distribution has a higher recognition rate. But as the noise increases, Gaussian becomes a better estimator.

The result genuinely suprised me, so I try to find the cause of the difference by looking at a sample.



Mean Diff.

Var. Diff.



Red: Diff. Mean

Blue: Diff Var.

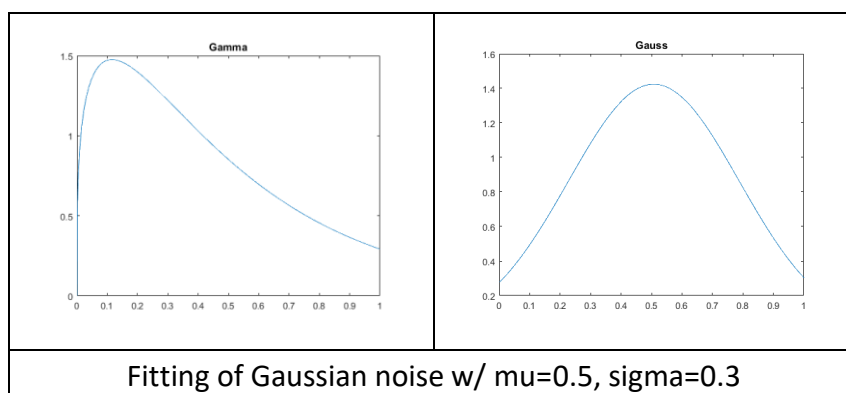
The Diff. Mean is low whatsoever.

The Diff. Var. is the main factor of difference.

↑ Training sample of letter 'c'. The green part is the average Mean/Var of the two estimation. The orange part shows the difference in Mean/Var.

There is no difference in mean, and the difference in variance occurs on pixels with higher variance (where the average mean aren't so significant). So the difference between Gamma and Gaussian might be caused by the way they model high variance.

In the **Why Gamma Distribution?** part of this report, I stated that Gamma can do whatever Gaussian can do, and showed two examples. But now we see that Gamma did a poor job when the noise increases, so I did more test:



There're more tests that can't fit in here. But just look at one example with a high variance(noise), we can tell that Gamma has a different fitting strategy. And since the noise that I applied to the data is a Gaussian noise, Gamma has a trouble fitting them when the variance is large, which causes the accuracy to drop after a certain point. Maybe this is why Gamma is not used in real world. There is only a few usage of Gamma distribution in Image Processing. Gaussian noise is everywhere, a distribution that can't properly fit it is therefore useless in most case. (4 page too little.)