hw3

April 24, 2020

1 HW 3: Machine Learning

Juan Vila

1.1 Question 1

1.1.1 Part A

For proof we are going to calculate the expected value of the Beta distribution

$$E[p(\theta;\alpha)] = \int_0^1 \theta \frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\theta^{\alpha-1}(1-\theta)^{\alpha-1}d\theta$$
$$E[p(\theta;\alpha)] = \frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\int_0^1 \theta\theta^{\alpha-1}(1-\theta)^{\alpha-1}d\theta$$

Solving by the integral we get that $B(\alpha+1,\alpha) = \int_0^1 \theta\theta^{\alpha-1}(1-\theta)^{\alpha-1}d\theta$ and knowing that $B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$, we can convert the previous result into:

$$E[p(\theta;\alpha)] = \frac{1}{B(\alpha,\alpha)}B(\alpha+1,\alpha)$$
$$E[p(\theta;\alpha)] = \frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\frac{\Gamma(\alpha+1)\Gamma(\alpha)}{\Gamma(2\alpha+1)}$$
$$E[p(\theta;\alpha)] = \frac{\Gamma(2\alpha)}{\Gamma(\alpha)}\frac{\Gamma(\alpha+1)}{\Gamma(2\alpha+1)}$$

Using the property that $\Gamma(\alpha) = \Gamma(\alpha-1)(\alpha-1)$

$$E[p(\theta;\alpha)] = \frac{\Gamma(2\alpha)}{\Gamma(\alpha)}\frac{\Gamma(\alpha)\alpha}{\Gamma(2\alpha)2\alpha}$$

$$E[p(\theta;\alpha)] = \frac{1}{2}$$

We can see that for any value of alpha the beta distribution reflects the prior.

1.1.2 Part B

For resolving this part we need to show that if we multiply the beta density by the Bernoulli likelihood we obtain a beta density.

$$p(\theta|x,\alpha) \propto \left(\theta^{S(x)}(1-\theta)^{N-S(x)}\right)\frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\theta^{\alpha-1}(1-\theta)^{\alpha-1}$$

Re- arraging terms

$$p(\theta|x,\alpha) \propto \frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\left(\theta^{S(x)+\alpha-1}(1-\theta)^{N-S(x)+\alpha-1}\right)$$

$$p(\theta|x,\alpha) \propto B\left(S(x)+\alpha-1, N-S(x)+\alpha-1\right)$$

This implies that beta distribution is conjugate prior of the bernulli distribution

1.1.3 Part C

MAP

$$L(\theta|x,\alpha) = \left(\theta^{S(x)}(1-\theta)^{N-S(x)}\right)\frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\theta^{\alpha-1}(1-\theta)^{\alpha-1}$$

$$L(\theta|x,\alpha) = \frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\left(\theta^{S(x)+\alpha-1}(1-\theta)^{N-S(x)+\alpha-1}\right)$$

$$ln(L(\theta|x,\alpha)) = ln\left(\frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\right) + ln(\theta)(S(x)+\alpha-1) + ln(1-\theta)(N-S(x)+\alpha-1)$$

Now we can take the derivate over θ and set equal to zero and solve for θ

$$\frac{\partial L(\theta|x,\alpha)}{\partial\theta} = \frac{1}{\theta}(S(x)+\alpha-1) - \frac{1}{1-\theta}(N-S(x)+\alpha-1)$$

$$\hat{\theta} = \frac{S(x)+\alpha-1}{N+2\alpha-2}$$

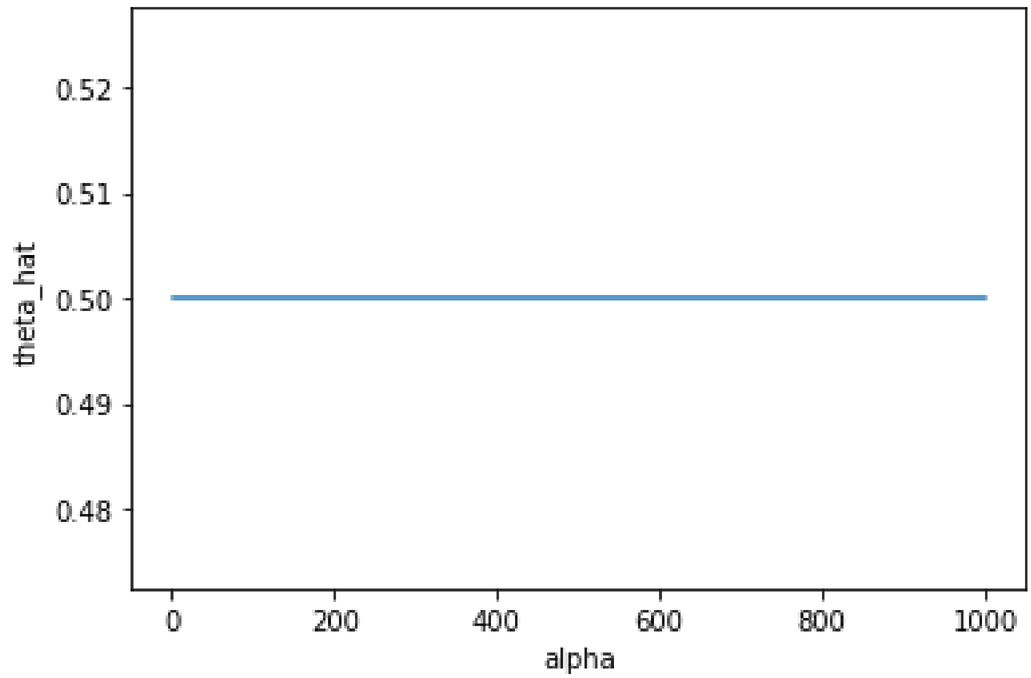
For the effect of α we can made a simulation asuming that S/N=1/2, then we can see that for any value of α there is no effect on the estimator.The same could be made if we take the limit of α as goes to infinity and will see that $\hat{\theta}$ converge to 1/2. For doing this we need to divide de numerator and denominator by $\frac{1}{\alpha}$ and take the limit.

[419]:
```
import numpy as np
import matplotlib.pyplot as plt
alpha=np.arange(1,1001)
N=1000
S=500
```

[420]:
```
theta_hat =[]
for i in alpha:
    num = S+i-1
    dem = N + 2*i -2
```

```
    z=num/dem
    theta_hat.append(z)

plt.plot( alpha,theta_hat)
plt.xlabel('alpha')
plt.ylabel('theta_hat')
```

[420]: Text(0, 0.5, 'theta_hat')



For know the effect of N in the estimator we could do the following assumption, we know that S is the sumation of X's. As X can take values of 0 or 1, we can assume that $S = \theta N$ where θ is the true value of θ.

$$\hat{\theta} = \frac{\theta N + \alpha - 1}{N + 2\alpha - 2}$$

Now if we divide the numerator and denominator by $\frac{1}{N}$, will get:

$$\hat{\theta} = \frac{\theta + \frac{\alpha}{N} - \frac{1}{N}}{1 + 2\frac{\alpha}{N} - \frac{2}{N}}$$

Now if we take the limit of this expression as N goes to infinity we have that all expresions divided by N converge to values very close to zero, showing that:

$$\hat{\theta} = \theta$$

MLE

$$L(\theta|x,\alpha) = \left(\theta^{S(x)}(1-\theta)^{N-S(x)}\right)$$

$$ln(L(\theta|x,\alpha)) = ln(\theta)(S(x)) + ln(1-\theta)(N-S(x))$$

Now we can take the derivate over θ and set equal to zero and solve for θ

$$\frac{\partial L(\theta|x,\alpha)}{\partial\theta} = \frac{1}{\theta}(S(x)) - \frac{1}{1-\theta}(N-S(x))$$

$$\hat{\theta}_{mle} = \frac{S(x)}{N}$$

Now if we divide the numerator and denominator by $\frac{1}{N}$ and we can assume that $S = \theta N$ where θ is the true value of θ, will get:

$$\hat{\theta}_{mle} = \frac{\frac{\theta N}{N}}{\frac{N}{N}}$$

Now if we take the limit of this expression as N goes to infinity we have that all expresions divided by N converge to values very close to zero, showing that:

$$\hat{\theta}_{mle} = \theta$$

Posterior Distribution From the previous part we know the that the prior by the likehood function is:

$$p(\theta|x,\alpha) \propto p(x|\theta)p(\theta)$$

$$p(\theta|x,\alpha) \propto \frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\left(\theta^{S(x)+\alpha-1}(1-\theta)^{N-S(x)+\alpha-1}\right)$$

$$p(\theta|x,\alpha) \propto \frac{\left(\theta^{S(x)+\alpha-1}(1-\theta)^{N-S(x)+\alpha-1}\right)}{B(\alpha,\alpha)}$$

For getting the marginal we have to solve:

$$p(x) = \int_0^1 \frac{\left(\theta^{S(x)+\alpha-1}(1-\theta)^{N-S(x)+\alpha-1}\right)}{B(\alpha,\alpha)}d\theta$$

$$p(x) = \frac{\left(\theta^{S(x)+\alpha}(1-\theta)^{N-S(x)+\alpha}\right)}{B(\alpha,\alpha)}$$

Then we can build the posterior distribution as

$$p(\theta|x,\alpha) \propto \frac{\frac{\left(\theta^{S(x)+\alpha-1}(1-\theta)^{N-S(x)+\alpha-1}\right)}{B(\alpha,\alpha)}}{\frac{\left(\theta^{S(x)+\alpha}(1-\theta)^{N-S(x)+\alpha}\right)}{B(\alpha,\alpha)}}$$

$$p(\theta|x,\alpha) \propto \frac{\left(\theta^{S(x)+\alpha-1}(1-\theta)^{N-S(x)+\alpha-1}\right)}{\left(\theta^{S(x)+\alpha}(1-\theta)^{N-S(x)+\alpha}\right)}$$

$$p(\theta|x,\alpha) \propto \frac{\left(\theta^{S(x)+\alpha-1}(1-\theta)^{N-S(x)+\alpha-1}\right)}{B\left(S(x)+\alpha, N-S(x)+\alpha\right)}$$

Which is a :

$$p(\theta|x,\alpha) \propto B\left(S(x)+\alpha, N-S(x)+\alpha\right)$$

Now the we know that the expected value of any Beta Dist. is $E[B(y,x)] = \frac{y}{y+x}$, then the expected value of the posterior is:

$$E[B(S(x)+\alpha, N-S(x)+\alpha)] = \frac{S(x)+\alpha}{S(x)+\alpha+N-S(x)+\alpha}$$

$$E[B(S(x)+\alpha, N-S(x)+\alpha)] = \frac{S(x)+\alpha}{N+2\alpha}$$

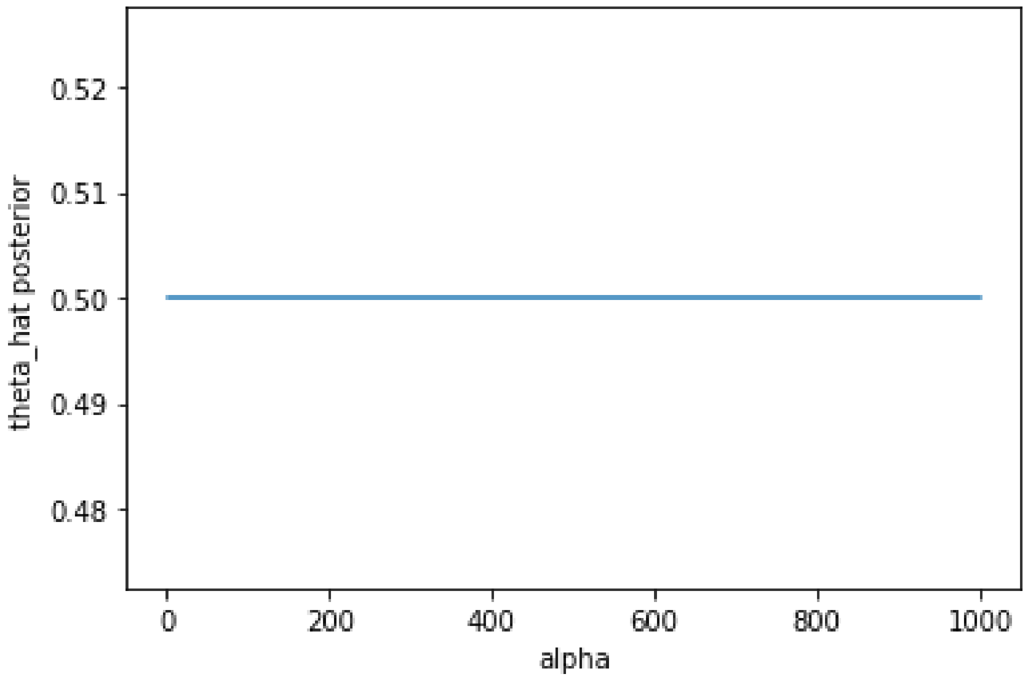
Now if we divide the numerator and denominator by $\frac{1}{N}$ and we can assume that $S = \theta N$ where θ is the true value of θ, will get:

$$E[B(S(x)+\alpha, N-S(x)+\alpha)] = \frac{\theta+\frac{\alpha}{N}}{1+\frac{2\alpha}{N}}$$

Now if we take the limit as N goes to infinity we have that $plim(E[B(S(x)+\alpha, N-S(x)+\alpha)]) = \theta$ On the other hand we can see the effect of alpha is the same as in the MAP

[421]:
```python
theta_hat_post =[]
for i in alpha:
    num = S+i
    dem = N +2*i
    theta_hat_post.append(num/dem)

plt.plot( alpha,theta_hat_post)
plt.xlabel('alpha')
plt.ylabel('theta_hat posterior')
```

[421]: Text(0, 0.5, 'theta_hat posterior')



All this discussion tell us that as N is big all the estimtors converge with the MLE, on the other hand as N is small they believe in the prior.

1.2 Question 2

1.2.1 Part A

$$y_i|x_i, \theta \ Poisson(x_i^T\theta)$$

$$Pr(y_i|\lambda) = \frac{\lambda^{i_i}e^{-\lambda}}{y_i!}$$

The single-parameter exponential family is expressed as:

$$p(x|\eta) = h(x)e^{\eta^T T(x) - A(\eta)}$$

where η is the natural parameter, T(x) is the sufficient statistics, $A(\theta)$: log partition function, and have mean μ.

For obtain a similar form we can convert the Poisson distribution using the exp(log()) transformation, then we get:

$$p(x|\eta) = \frac{1}{x!}e^{xlog(\lambda) - \lambda}$$

Then we can see that $\eta = log\lambda$, $T(x) = x$, $A(\eta) = e^\eta$, and $\mu = e^\eta$. Then we can say that $\eta = \theta^T x$ and probability of observing y_i given x_i and θ is :

$$P(y_i|x_i, \theta) = \frac{1}{y_i!} e^{y_i \theta^T x - e^{\theta^T x}}$$

Then we can convert this into the likelihood function and get:

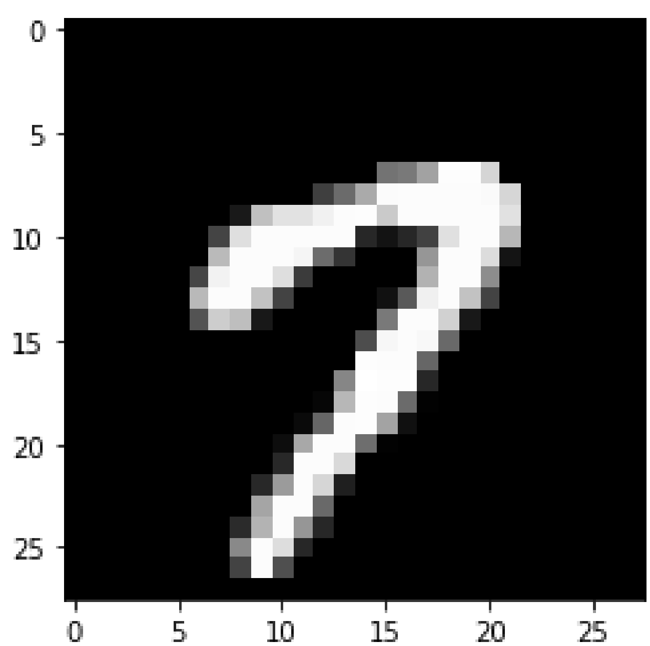
$$L(y_i|x_i, \theta) = \prod_{i=1}^{n} \frac{1}{y_i!} e^{y_i \theta^T x - e^{\theta^T x}}$$

[422]:
```
## Load Data
from scipy.io import loadmat
mu_base = loadmat('PS2 data files/mean.mat')
sevens_base = loadmat('PS2 data files/sevens.mat')
mu=mu_base['mu']
d=sevens_base['d']
```

[423]:
```
def print_image(data):
    data = data.reshape (28 ,28 , -1,order='F')
    print(data.shape)
    return plt.imshow ( data[: ,: ,0] , cmap = 'gray')
print_image(d)
```

```
(28, 28, 1000)
```

[423]: <matplotlib.image.AxesImage at 0x1c3c970110>



1.2.2 Creating the reduced dimension images of X

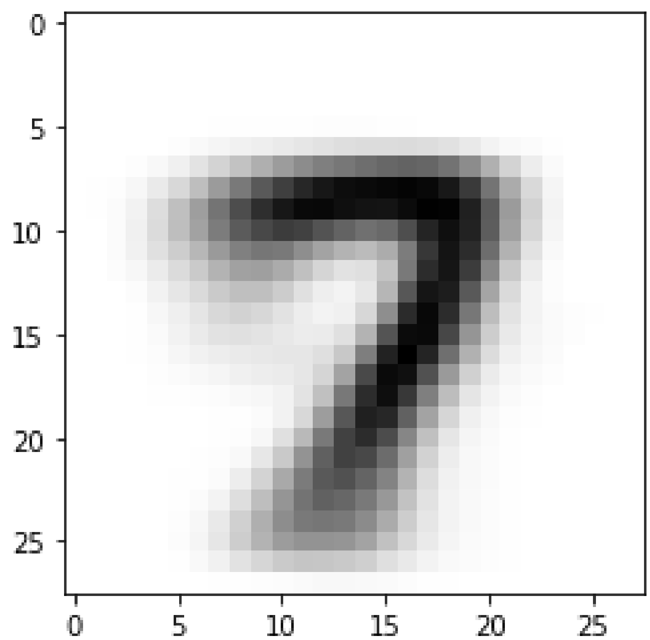
[424]:
```
import numpy as np
# obtain svd
U, S, V = np.linalg.svd(d)

# inspect shapes of the matrices
print(U.shape, S.shape, V.shape)

low_rank_25 = U[:, :25] @ np.diag(S[:25])
low_rank_50 = U[:, :50] @ np.diag(S[:50])
```

```
(784, 784) (784,) (1000, 1000)
```

[425]:
```
print_image(low_rank_25)
```

```
(28, 28, 25)
```

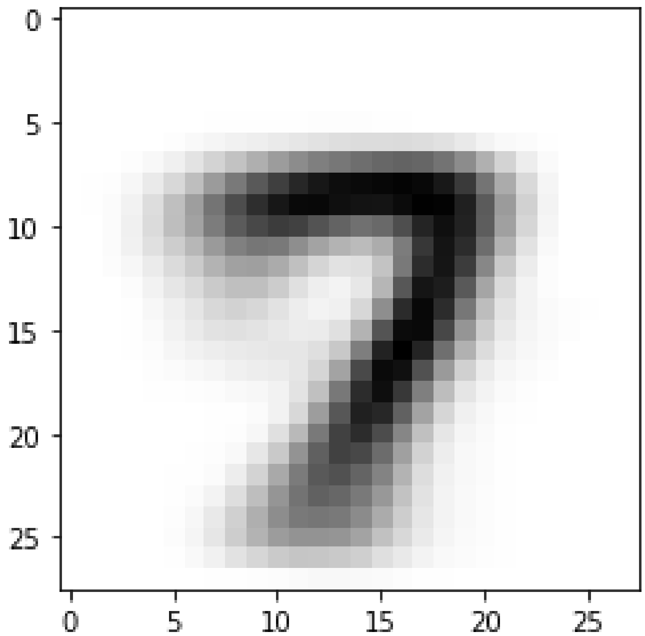
[425]: <matplotlib.image.AxesImage at 0x1c3ca499d0>



[426]:
```
print_image(low_rank_50)
```

```
(28, 28, 50)
```

[426]: <matplotlib.image.AxesImage at 0x1c38b67990>



[427]:
```
U[:, :25].shape
#np.diag(S[:25]).shape
#V[:25, :].shape
```

[427]: (784, 25)

1.2.3 Creating MU and Poisson(μ)

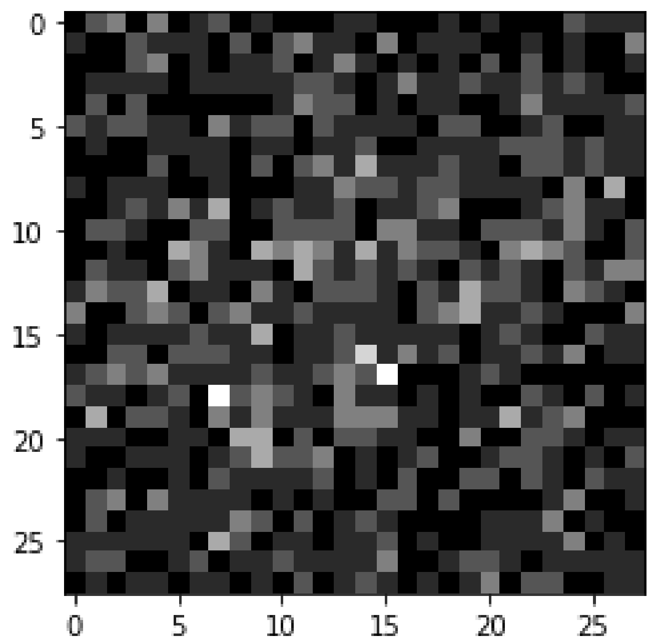
[481]:
```
mu_r = mu.reshape (784 ,1 , -1)
```

[429]:
```
import pandas as pd
df = pd.DataFrame(data=mu_r[::,0], columns=['mu'])
#r = poisson.rvs(mu, size=1000)

modDfObj = df.apply(poisson.rvs)
```

Checking the result with input.

[430]:
```
print_image(modDfObj['mu'].values)
y = modDfObj['mu'].values
```

```
(28, 28, 1)
```



[412]:

```
import scipy
import statsmodels.api as sm

glm_25 = sm.GLM(y, low_rank_25, family=sm.families.Poisson())
res_25 = glm_25.fit()
print(res_25.summary())
```

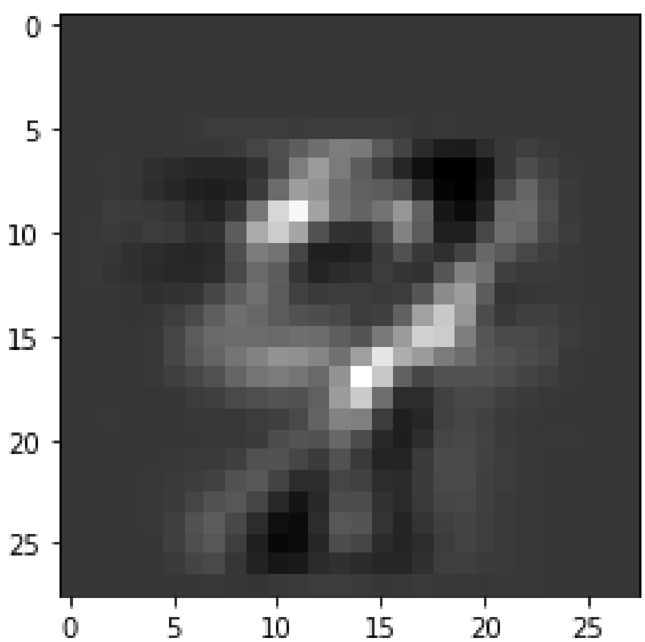
```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  784
Model:                            GLM   Df Residuals:                      759
Model Family:                 Poisson   Df Model:                           24
Link Function:                    log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -1046.2
Date:                Fri, 24 Apr 2020   Deviance:                       885.48
Time:                        11:56:38   Pearson chi2:                     755.
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1         -6.458e-05   1.73e-05     -3.737      0.000   -9.84e-05   -3.07e-05
x2          -5.06e-06   4.68e-05     -0.108      0.914   -9.68e-05    8.67e-05
x3         -6.462e-06   5.53e-05     -0.117      0.907      -0.000       0.000
x4             0.0002   7.41e-05      2.098      0.036    1.02e-05       0.000
x5          7.515e-05   7.55e-05      0.995      0.320   -7.28e-05       0.000
```

```
x6            7.23e-05   9.07e-05      0.797      0.425     -0.000      0.000
x7             -0.0002   9.01e-05     -1.767      0.077     -0.000   1.74e-05
x8             -0.0001      0.000     -1.167      0.243     -0.000   8.79e-05
x9              0.0002      0.000      1.502      0.133   -6.1e-05      0.000
x10            -0.0002      0.000     -1.883      0.060     -0.001   1.03e-05
x11            -0.0004      0.000     -3.160      0.002     -0.001     -0.000
x12          6.609e-05      0.000      0.507      0.612     -0.000      0.000
x13            -0.0006      0.000     -4.213      0.000     -0.001     -0.000
x14            -0.0002      0.000     -1.423      0.155     -0.000   7.41e-05
x15             0.0004      0.000      2.646      0.008      0.000      0.001
x16             0.0001      0.000      0.839      0.401     -0.000      0.000
x17             0.0004      0.000      2.444      0.015   8.32e-05      0.001
x18             0.0004      0.000      2.260      0.024   5.66e-05      0.001
x19            -0.0004      0.000     -2.171      0.030     -0.001  -4.01e-05
x20             0.0001      0.000      0.722      0.471     -0.000      0.000
x21             0.0001      0.000      0.708      0.479     -0.000      0.000
x22             0.0003      0.000      1.310      0.190     -0.000      0.001
x23             0.0003      0.000      1.610      0.107  -7.37e-05      0.001
x24             0.0002      0.000      0.919      0.358     -0.000      0.001
x25         -6.966e-05      0.000     -0.321      0.748     -0.000      0.000
==============================================================================
```

[431]:
```
pred_25 = res_25.predict()
print_image(pred_25)
```

```
(28, 28, 1)
```

[431]: <matplotlib.image.AxesImage at 0x1c2f755d50>



[432]:
```python
glm_50 = sm.GLM(y, low_rank_50, family=sm.families.Poisson())
res_50 = glm_50.fit()
print(res_50.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  784
Model:                            GLM   Df Residuals:                      734
Model Family:                 Poisson   Df Model:                           49
Link Function:                    log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -1020.1
Date:                Fri, 24 Apr 2020   Deviance:                       774.72
Time:                        19:41:33   Pearson chi2:                     652.
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1         -5.889e-05   1.74e-05     -3.389      0.001    -9.3e-05   -2.48e-05
x2         -7.124e-05   4.62e-05     -1.543      0.123      -0.000    1.92e-05
x3             0.0001   5.56e-05      2.174      0.030    1.19e-05       0.000
x4             0.0002   7.89e-05      1.953      0.051   -5.79e-07       0.000
x5             0.0002    7.7e-05      3.190      0.001    9.47e-05       0.000
x6          3.244e-05   9.25e-05      0.351      0.726      -0.000       0.000
x7         -7.983e-05   9.38e-05     -0.851      0.395      -0.000       0.000
x8            -0.0002      0.000     -2.023      0.043      -0.000   -7.33e-06
x9          6.378e-05      0.000      0.520      0.603      -0.000       0.000
x10         6.137e-05      0.000      0.477      0.633      -0.000       0.000
x11         -1.98e-05      0.000     -0.145      0.885      -0.000       0.000
x12          4.29e-05      0.000      0.302      0.763      -0.000       0.000
x13        -3.685e-05      0.000     -0.259      0.796      -0.000       0.000
x14         8.753e-05      0.000      0.612      0.540      -0.000       0.000
x15            0.0004      0.000      2.612      0.009       0.000       0.001
x16         1.184e-05      0.000      0.068      0.946      -0.000       0.000
x17        -2.942e-05      0.000     -0.174      0.862      -0.000       0.000
x18         1.507e-05      0.000      0.082      0.934      -0.000       0.000
x19           -0.0004      0.000     -1.848      0.065      -0.001    2.19e-05
x20           -0.0001      0.000     -0.764      0.445      -0.000       0.000
x21            0.0003      0.000      1.803      0.071   -2.93e-05       0.001
x22           -0.0004      0.000     -1.917      0.055      -0.001    9.44e-06
x23            0.0004      0.000      1.833      0.067   -2.67e-05       0.001
x24           -0.0004      0.000     -1.582      0.114      -0.001    8.57e-05
x25        -1.953e-05      0.000     -0.088      0.930      -0.000       0.000
x26            0.0002      0.000      0.857      0.391      -0.000       0.001
x27           -0.0002      0.000     -0.837      0.403      -0.001       0.000
```

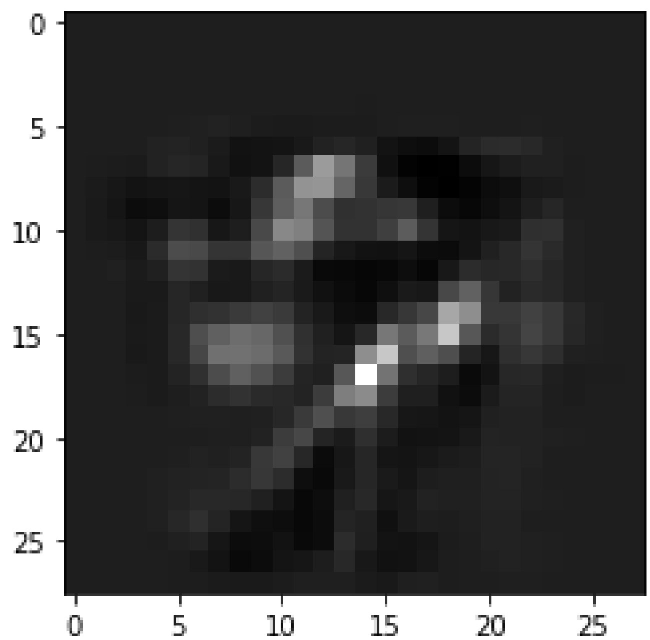
x28	0.0005	0.000	1.979	0.048	4.67e-06	0.001
x29	-0.0003	0.000	-1.239	0.215	-0.001	0.000
x30	0.0005	0.000	2.002	0.045	1.1e-05	0.001
x31	-0.0002	0.000	-0.592	0.554	-0.001	0.000
x32	0.0003	0.000	1.224	0.221	-0.000	0.001
x33	0.0004	0.000	1.543	0.123	-0.000	0.001
x34	0.0005	0.000	1.696	0.090	-7.79e-05	0.001
x35	0.0002	0.000	0.672	0.501	-0.000	0.001
x36	0.0002	0.000	0.622	0.534	-0.000	0.001
x37	0.0003	0.000	1.042	0.297	-0.000	0.001
x38	-0.0002	0.000	-0.678	0.498	-0.001	0.000
x39	0.0002	0.000	0.517	0.605	-0.000	0.001
x40	-0.0003	0.000	-0.854	0.393	-0.001	0.000
x41	-8.487e-05	0.000	-0.262	0.794	-0.001	0.001
x42	0.0008	0.000	2.398	0.016	0.000	0.001
x43	0.0003	0.000	0.863	0.388	-0.000	0.001
x44	0.0002	0.000	0.591	0.554	-0.000	0.001
x45	0.0002	0.000	0.718	0.472	-0.000	0.001
x46	-0.0001	0.000	-0.361	0.718	-0.001	0.001
x47	0.0004	0.000	1.237	0.216	-0.000	0.001
x48	-1.088e-06	0.000	-0.003	0.998	-0.001	0.001
x49	0.0002	0.000	0.419	0.675	-0.001	0.001
x50	0.0008	0.000	2.199	0.028	9.05e-05	0.002

```
==============================================================================
```

[170]:
```python
pred_50 = res_50.predict()
print_image(pred_50)
```

```
(28, 28, 1)
```

[170]:
```
<matplotlib.image.AxesImage at 0x1c355c2dd0>
```



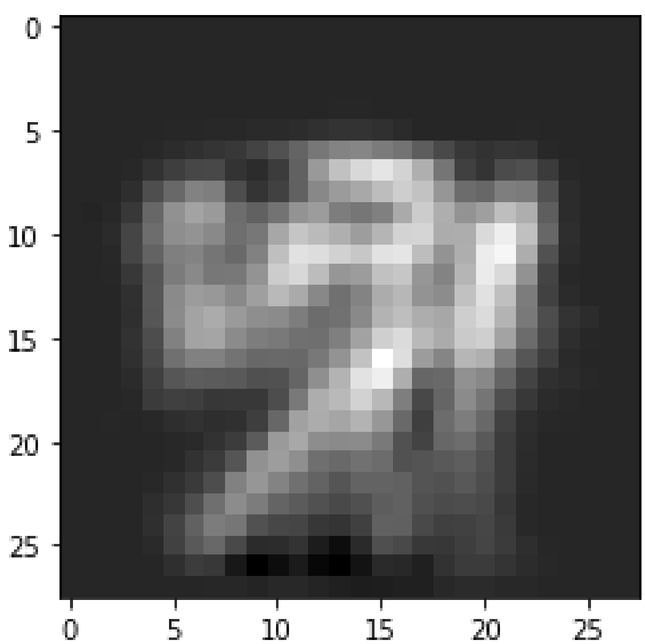
1.2.4 Part C

[433]:
```
glm_25_c = sm.GLM(y, low_rank_25,)
res_25_c = glm_25_c.fit()
print(res_25_c .summary())
pred_25_c = res_25_c.predict()
print_image(pred_25_c)
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  784
Model:                            GLM   Df Residuals:                      759
Model Family:                Gaussian   Df Model:                           24
Link Function:               identity   Scale:                          1.6414
Method:                          IRLS   Log-Likelihood:                -1294.0
Date:                Fri, 24 Apr 2020   Deviance:                       1245.8
Time:                        19:41:37   Pearson chi2:                 1.25e+03
No. Iterations:                     3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.0004   2.42e-05    -15.401      0.000      -0.000      -0.000
x2          5.175e-05   6.43e-05      0.805      0.421   -7.42e-05       0.000
x3             0.0002   7.76e-05      3.049      0.002    8.45e-05       0.000
```

```
x4             0.0008      0.000      7.344      0.000       0.001       0.001
x5             0.0004      0.000      3.671      0.000       0.000       0.001
x6             0.0005      0.000      4.099      0.000       0.000       0.001
x7            -0.0004      0.000     -2.813      0.005      -0.001      -0.000
x8            -0.0002      0.000     -1.365      0.172      -0.001    9.18e-05
x9            -0.0001      0.000     -0.779      0.436      -0.000       0.000
x10         4.691e-05      0.000      0.259      0.796      -0.000       0.000
x11        -8.169e-05      0.000     -0.433      0.665      -0.000       0.000
x12        -4.356e-06      0.000     -0.023      0.982      -0.000       0.000
x13           -0.0003      0.000     -1.582      0.114      -0.001    7.61e-05
x14            0.0006      0.000      2.783      0.005       0.000       0.001
x15            0.0007      0.000      3.128      0.002       0.000       0.001
x16        -1.605e-05      0.000     -0.067      0.946      -0.000       0.000
x17         6.068e-05      0.000      0.246      0.806      -0.000       0.001
x18           -0.0005      0.000     -1.783      0.075      -0.001    4.52e-05
x19           -0.0003      0.000     -1.316      0.188      -0.001       0.000
x20         7.797e-05      0.000      0.290      0.772      -0.000       0.001
x21            0.0003      0.000      1.028      0.304      -0.000       0.001
x22           -0.0006      0.000     -1.993      0.046      -0.001    -9.4e-06
x23            0.0003      0.000      0.984      0.325      -0.000       0.001
x24           -0.0003      0.000     -1.041      0.298      -0.001       0.000
x25           -0.0005      0.000     -1.750      0.080      -0.001    6.55e-05
==============================================================================
(28, 28, 1)
```

[433]: <matplotlib.image.AxesImage at 0x1c38d99650>



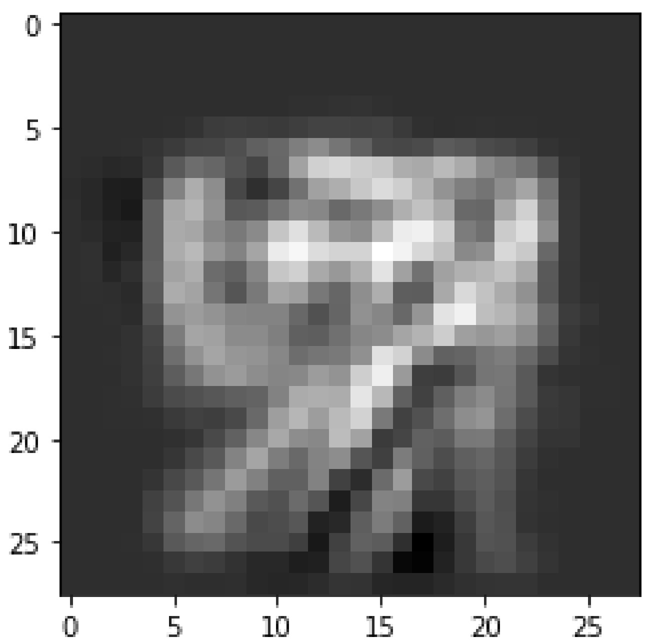
[434]:
```python
glm_50_c = sm.GLM(y, low_rank_50)
res_50_c = glm_50_c.fit()
print(res_50_c .summary())
pred_50_c = res_50_c.predict()
print_image(pred_50_c)
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  784
Model:                            GLM   Df Residuals:                      734
Model Family:                Gaussian   Df Model:                           49
Link Function:               identity   Scale:                          1.5981
Method:                          IRLS   Log-Likelihood:                -1270.4
Date:                Fri, 24 Apr 2020   Deviance:                       1173.0
Time:                        19:41:41   Pearson chi2:                 1.17e+03
No. Iterations:                     3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.0004   2.39e-05    -15.608      0.000      -0.000      -0.000
x2          5.175e-05   6.34e-05      0.816      0.414   -7.25e-05       0.000
x3             0.0002   7.66e-05      3.090      0.002    8.66e-05       0.000
x4             0.0008      0.000      7.443      0.000       0.001       0.001
x5             0.0004      0.000      3.720      0.000       0.000       0.001
x6             0.0005      0.000      4.154      0.000       0.000       0.001
x7            -0.0004      0.000     -2.851      0.004      -0.001      -0.000
x8            -0.0002      0.000     -1.384      0.166      -0.001    8.78e-05
x9            -0.0001      0.000     -0.789      0.430      -0.000       0.000
x10         4.691e-05      0.000      0.262      0.793      -0.000       0.000
x11        -8.169e-05      0.000     -0.439      0.661      -0.000       0.000
x12        -4.356e-06      0.000     -0.023      0.982      -0.000       0.000
x13           -0.0003      0.000     -1.603      0.109      -0.001    7.09e-05
x14            0.0006      0.000      2.820      0.005       0.000       0.001
x15            0.0007      0.000      3.170      0.002       0.000       0.001
x16        -1.605e-05      0.000     -0.068      0.946      -0.000       0.000
x17         6.068e-05      0.000      0.249      0.803      -0.000       0.001
x18           -0.0005      0.000     -1.807      0.071      -0.001    3.86e-05
x19           -0.0003      0.000     -1.334      0.182      -0.001       0.000
x20         7.797e-05      0.000      0.294      0.769      -0.000       0.001
x21            0.0003      0.000      1.042      0.297      -0.000       0.001
x22           -0.0006      0.000     -2.019      0.043      -0.001   -1.69e-05
x23            0.0003      0.000      0.997      0.319      -0.000       0.001
x24           -0.0003      0.000     -1.055      0.292      -0.001       0.000
x25           -0.0005      0.000     -1.774      0.076      -0.001    5.73e-05
x26            0.0001      0.000      0.404      0.686      -0.000       0.001
x27        -9.045e-05      0.000     -0.271      0.787      -0.001       0.001
```

x28	0.0006	0.000	1.869	0.062	-3.11e-05	0.001
x29	-0.0003	0.000	-0.872	0.383	-0.001	0.000
x30	0.0005	0.000	1.387	0.166	-0.000	0.001
x31	-0.0009	0.000	-2.401	0.016	-0.002	-0.000
x32	0.0006	0.000	1.671	0.095	-0.000	0.001
x33	0.0008	0.000	1.971	0.049	4.23e-06	0.002
x34	0.0005	0.000	1.268	0.205	-0.000	0.001
x35	0.0008	0.000	1.913	0.056	-1.92e-05	0.002
x36	0.0002	0.000	0.506	0.613	-0.001	0.001
x37	3.228e-05	0.000	0.077	0.939	-0.001	0.001
x38	-0.0004	0.000	-0.827	0.408	-0.001	0.000
x39	-3.654e-05	0.000	-0.085	0.933	-0.001	0.001
x40	-0.0005	0.000	-1.117	0.264	-0.001	0.000
x41	-0.0003	0.000	-0.589	0.556	-0.001	0.001
x42	0.0009	0.000	1.904	0.057	-2.57e-05	0.002
x43	0.0005	0.000	1.079	0.281	-0.000	0.001
x44	0.0005	0.000	0.982	0.326	-0.000	0.001
x45	0.0007	0.000	1.515	0.130	-0.000	0.002
x46	-0.0013	0.000	-2.668	0.008	-0.002	-0.000
x47	0.0005	0.001	0.942	0.346	-0.001	0.001
x48	0.0001	0.001	0.220	0.826	-0.001	0.001
x49	-0.0002	0.001	-0.344	0.731	-0.001	0.001
x50	0.0009	0.001	1.633	0.102	-0.000	0.002

```
==============================================================================
(28, 28, 1)
```

[434]: <matplotlib.image.AxesImage at 0x1c2f6ec110>

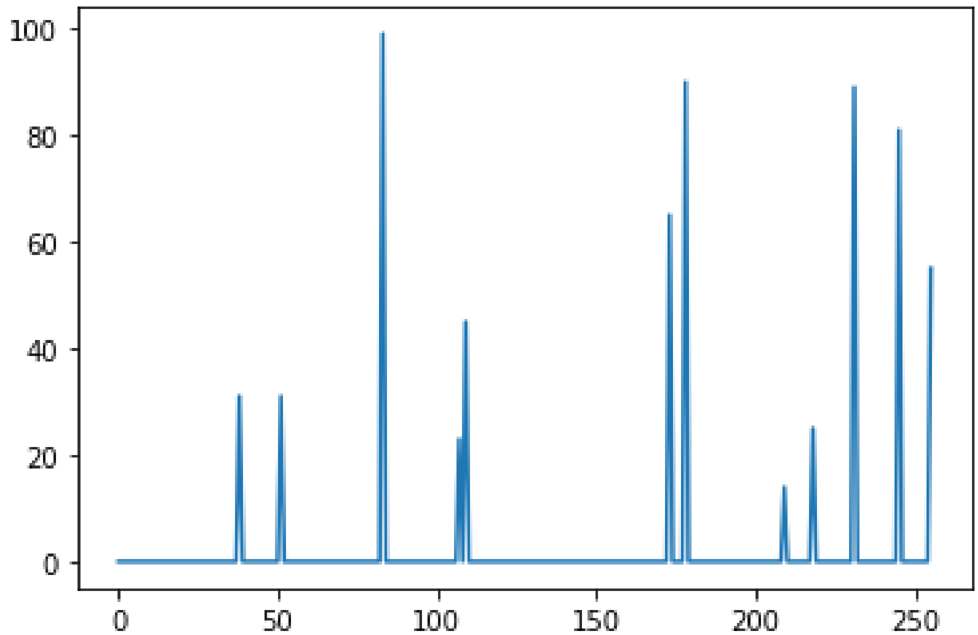


1.3 Question 3

1.3.1 Part A

[475]:
```
random.seed(10)
import random
theta=np.empty([256, ])
for x in range(256):
    p = random.randint(1,101)
    if p>=97:
        theta[x]=random.randint(1,101)
    else:
        theta[x]=0
plt.plot(theta)
```

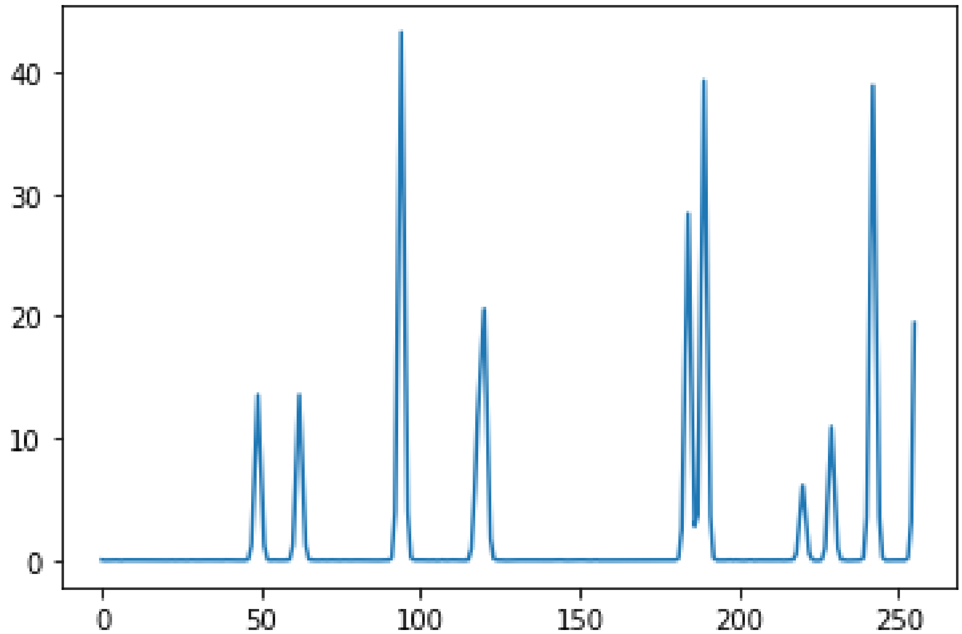
[475]: [<matplotlib.lines.Line2D at 0x1c3d915f50>]



[460]:
```
blur_base = loadmat('PS2 data files/lidar_blur.mat')
blur = blur_base['X']
noise = np.random.normal(0, 1/100, 256)
y_3=blur @theta  + noise
```

[461]:
```
plt.plot(y_3)
```

[461]: [<matplotlib.lines.Line2D at 0x1c39559b10>]



1.3.2 Part B

We know that in case a guassian distribution of MLE model with a lineal model we are going to get the solution

$$\hat{\theta}_{mle} = (X^T X)^{-1} XY$$

However if we try to invert the matrix we are going to notice that this matrix is not invertible do the multicollinearity of the columns. For checking this we can compare the rank with shape. We can see that there are 11 columns that are colineal, which meeans that we cannot invert the matrix.

[457]:
```
rank_blur = np.linalg.matrix_rank(blur)
shape_blur = blur.shape[1]
print('rank blur:'+ ' '+str(rank_blur), 'ncol blur:'+ ' '+str(shape_blur))
```

```
rank blur: 245 ncol blur: 256
```

[462]:
```
theta_mle = np.linalg.inv(np.transpose(blur) @ blur)@blur@y_3
error_sq_mle = ((theta - theta_mle)**2).sum()
```

1.3.3 Part C

[439]:
```
def bayes_reg(X,sigma_e,sigma_theta,y,mu_theta):
    sigma_e_inv = np.linalg.inv(sigma_e*np.identity(X.shape[1]))
    sigma_theta_inv = np.linalg.inv(sigma_theta*np.identity(X.shape[1]))
    p1 = np.linalg.inv( np.transpose(X) @ sigma_e_inv @ X + sigma_theta_inv)
    p3 = y-X@np.full((X.shape[1]), mu_theta)
    p2 = np.transpose(X) @ sigma_e_inv
    theta_bayes = np.full((X.shape[1]), mu_theta) + p1 @ p2 @ p3
    return theta_bayes

def predict_bayes(X,sigma_e,sigma_theta,y,mu_theta):
    theta_hat = bayes_reg(X,sigma_e,sigma_theta,y,mu_theta)
    y_hat = X @ theta_hat
    return y_hat
```

[465]:
```
sigma_theta_list = [1/100,1/100,1/10,1,2,4,5,10]
mle_list = [error_sq_mle]*len(sigma_theta_list)
sim_init = np.zeros(len(sigma_theta_list))
for i,val in enumerate(sigma_theta_list):
    y_hat_c = bayes_reg(blur,(1/100)**2,(val)**2,y_3,0)
    sim_init[i]=((theta-y_hat_c)**2).sum()
```

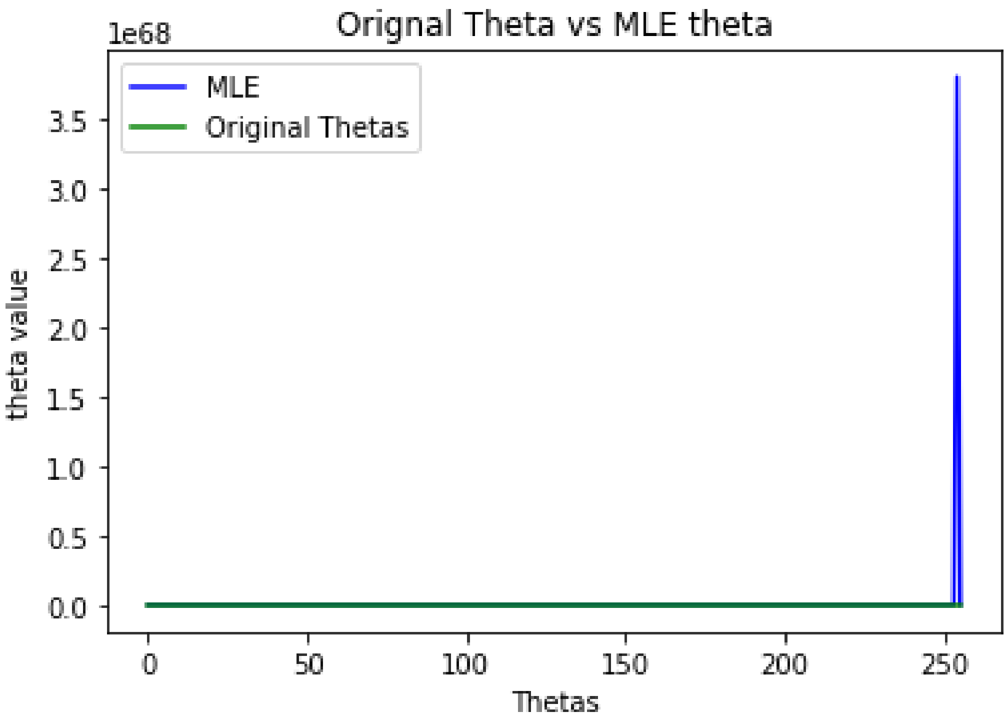
The squared error of difference between the θ of MLE withe original is a very big number, so is very bad estimator. This is show when we compare the next graph were basically the scale is so big that do not deblur correctly.

[467]:
```
mle_list[1]
```

[467]:
```
1.444995799739825e+137
```

[476]:
```
plt.plot( theta_mle, 'b-', label='MLE')
plt.plot(theta, 'g-', label='Original Thetas')
plt.legend(loc='upper left')
plt.xlabel('Thetas')
plt.ylabel('theta value')
plt.title('Orignal Theta vs MLE theta')
```

[476]:
```
Text(0.5, 1.0, 'Orignal Theta vs MLE theta')
```



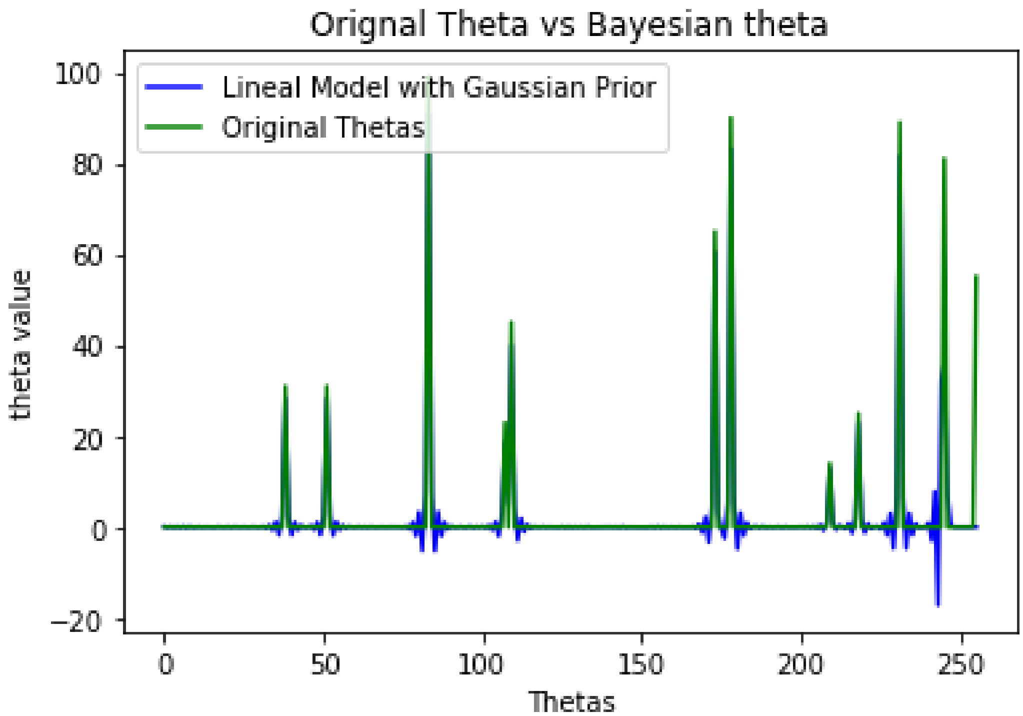
On the other hand if we use the lineal model with the gaussian prior for a gamma of 1 we got a very good result, an squared error if 7388 which is much lower than the MLE alternative.

[473]:
```
### TESTS of squared error #####
test = bayes_reg(blur,1/1000,1,y_3,0)
((theta-test)**2).sum()
```

[473]: 7388.025928316927

[480]:
```
plt.plot( test, 'b-', label='Lineal Model with Gaussian Prior')
plt.plot(theta, 'g-', label='Original Thetas')
plt.legend(loc='upper left')
plt.xlabel('Thetas')
plt.ylabel('theta value')
plt.title('Orignal Theta vs Bayesian theta')
```

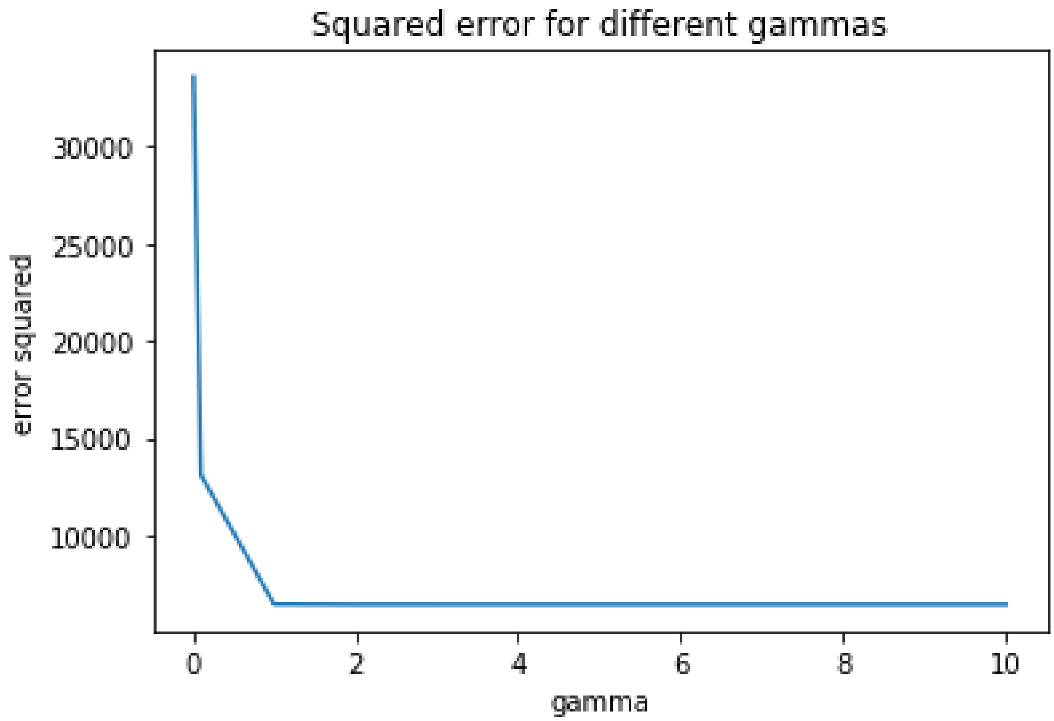
[480]: Text(0.5, 1.0, 'Orignal Theta vs Bayesian theta')



On the other hand, if we graph simulation for different gamma's we can see that the error diminish as gamma increase converge aroung a value of 6400 and gamma of 2.

[472]:
```
plt.plot(sigma_theta_list,sim_init)
plt.xlabel('gamma')
plt.ylabel('error squared')
plt.title('Squared error for different gammas')
```

[472]: Text(0.5, 1.0, 'Squared error for different gammas')


Squared error for different gammas
error squared
30000
25000
20000
15000
10000
0
2
4
6
8
10
gamma