Deep Learning: Homework #1

Due on October 20, 2019 at 12:00am

Professor Emad Fatemizadeh

Amirhossein Yousefi

97206984

Problem 1

Part a, b, c, d, e :
In this part we get our hands dirty getting familiar with ***numpy*** as useful package in python in order to work with matrix and arrays.In this part we built gaussian distribution and fit on it a gaussian curve and also we used two method to get sample from gaussian distribution as you can see in source code which are *sample* and *choice*.Figure8 is our results.

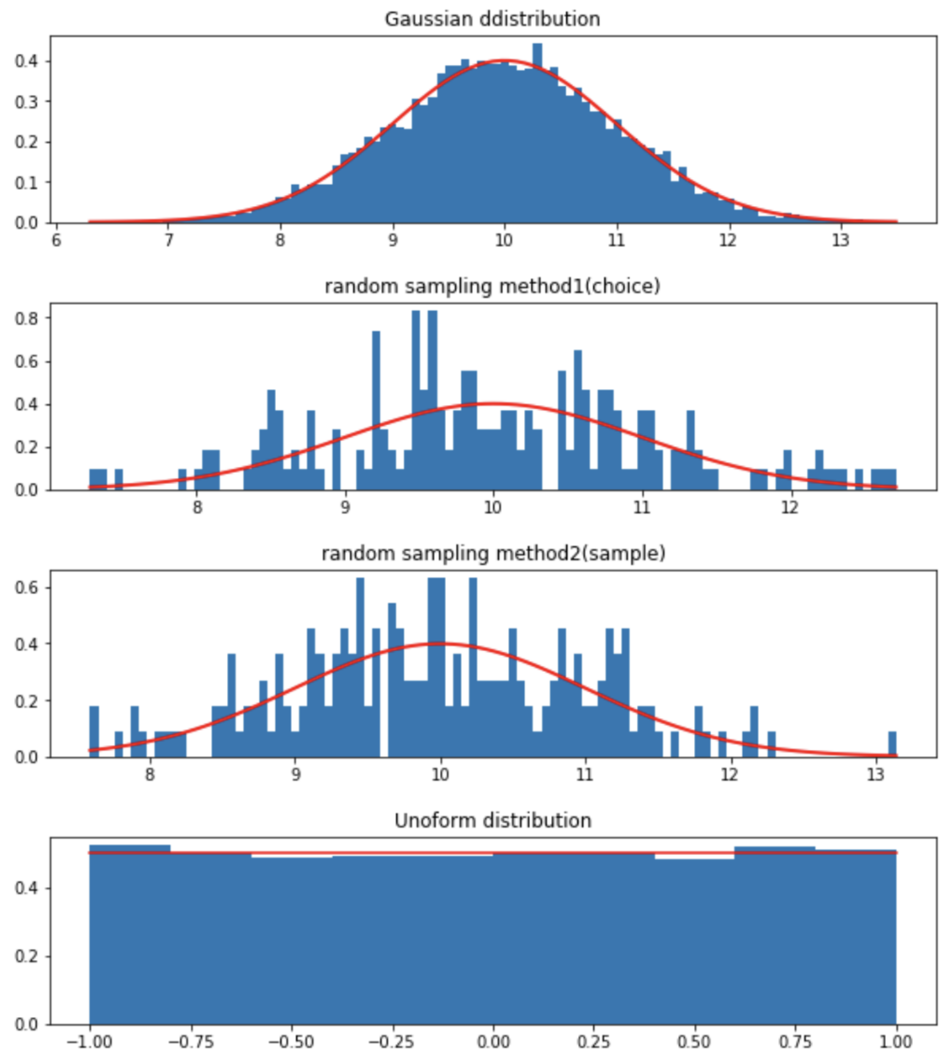
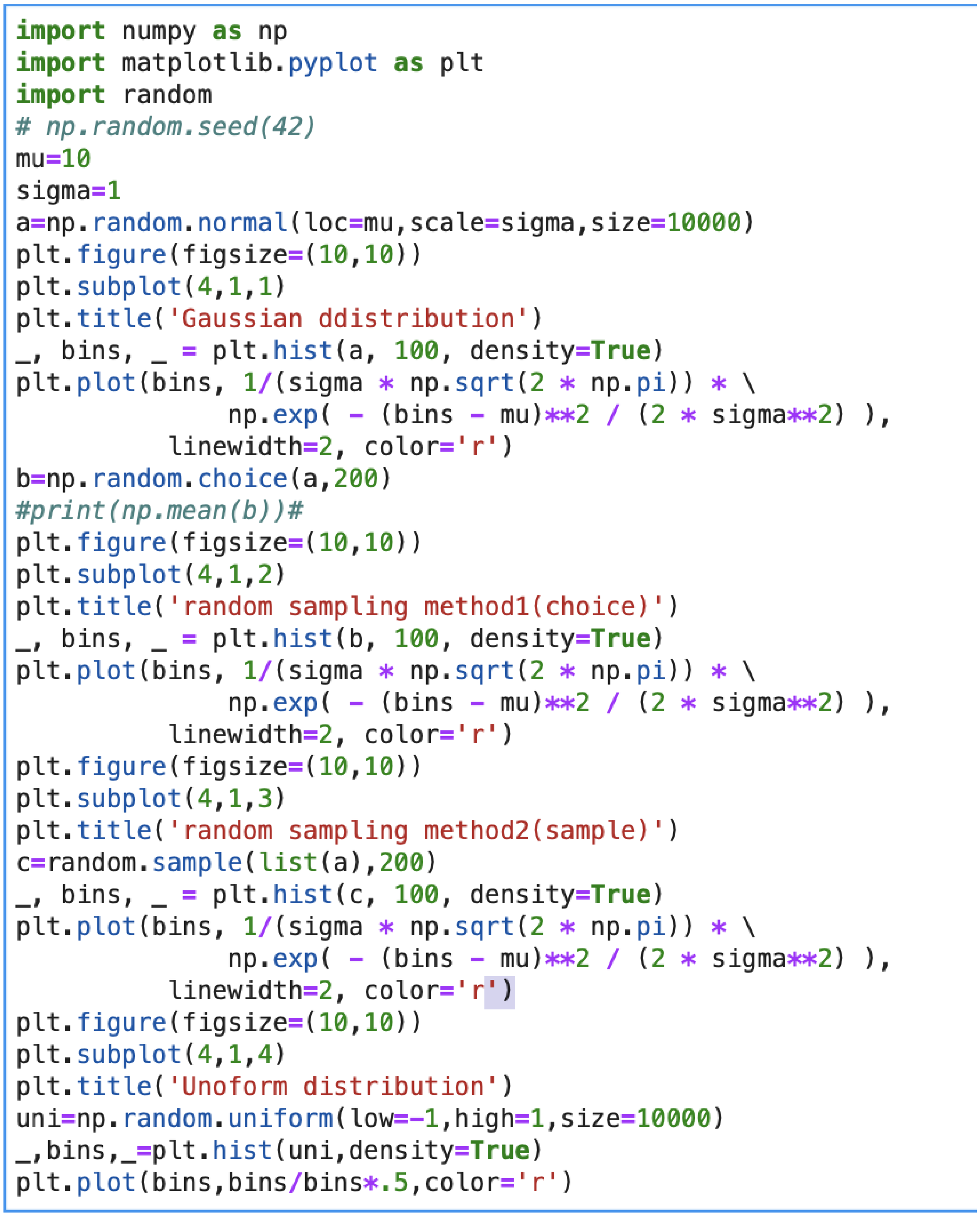


Figure 1: Gaussian Distribution and Uniform distribution and sample from Gaussian distribution.

After that we can see the source code in Figure2.

Problem 1 continued on next page. . .



```python
import numpy as np
import matplotlib.pyplot as plt
import random
# np.random.seed(42)
mu=10
sigma=1
a=np.random.normal(loc=mu,scale=sigma,size=10000)
plt.figure(figsize=(10,10))
plt.subplot(4,1,1)
plt.title('Gaussian ddistribution')
_, bins, _ = plt.hist(a, 100, density=True)
plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) * \
              np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
          linewidth=2, color='r')
b=np.random.choice(a,200)
#print(np.mean(b))#
plt.figure(figsize=(10,10))
plt.subplot(4,1,2)
plt.title('random sampling method1(choice)')
_, bins, _ = plt.hist(b, 100, density=True)
plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) * \
              np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
          linewidth=2, color='r')
plt.figure(figsize=(10,10))
plt.subplot(4,1,3)
plt.title('random sampling method2(sample)')
c=random.sample(list(a),200)
_, bins, _ = plt.hist(c, 100, density=True)
plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) * \
              np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
          linewidth=2, color='r')
plt.figure(figsize=(10,10))
plt.subplot(4,1,4)
plt.title('Unoform distribution')
uni=np.random.uniform(low=-1,high=1,size=10000)
_,bins,_=plt.hist(uni,density=True)
plt.plot(bins,bins/bins*.5,color='r')
```

Figure 2: source code for part *one*

Problem 1 continued on next page. . .

Justification for Part d:
We can see from figure1 that probability on mean is increased from 0.4 to 0.6 and also as *law of large number* states that mean of these sample converges to true mean as number of samples increase because it variance goes to zero by increasing numbers of samples. In other words $\lim_{n\to\infty} \frac{\sum_{i=1}^{n} x_i}{n} \xrightarrow{\text{in probability}} \mu$ and also we can infer that $\lim_{n\to\infty} Var(\frac{\sum_{i=1}^{n} x_i}{n}) \to 0$.
Part f:
The distribution for product of uniform distribution and gaussian distribution is more like a uniform distribution but is not exactly uniform .It is obvious that range of distribution has significantly changed to [-10,10] .After this range, distribution crucialy descended .

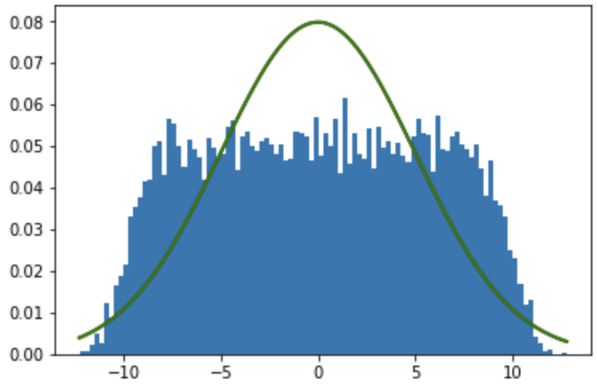


Figure 3: product of gaussian and uniform distribution

Also source code for this part is in figure4

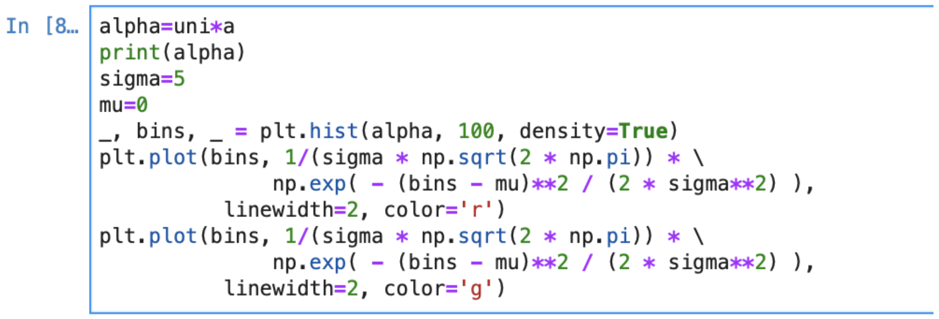

```
In [8…  alpha=uni*a
        print(alpha)
        sigma=5
        mu=0
        _, bins, _ = plt.hist(alpha, 100, density=True)
        plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) * \
                      np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
                linewidth=2, color='r')
        plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) * \
                      np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
                linewidth=2, color='g')
```

Figure 4: source code

Problem 2

Part a and b:
Here is matrix that we plotted in figure5 form img.csv .

Problem 2 continued on next page. . .

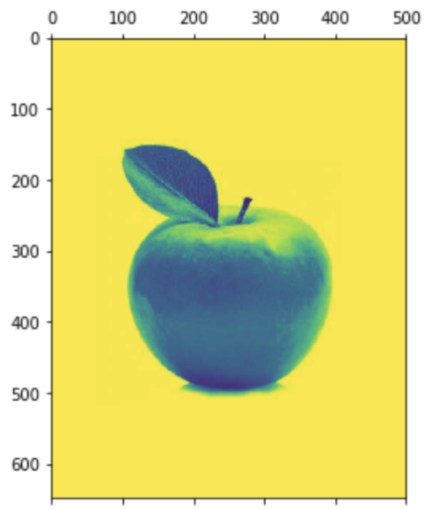
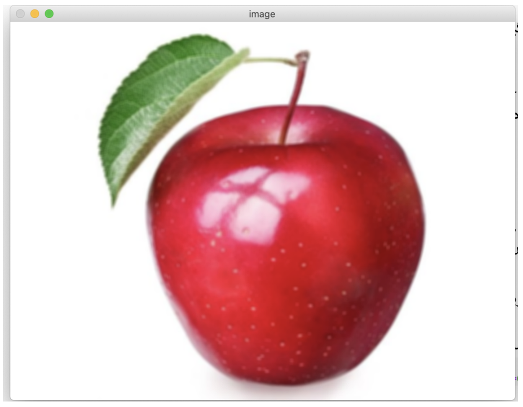


Figure 5: plotted matrix from img.csv

Part f:
Here in we filtered original image by mean filter and output is less brighter than original one..



(a) filtered image

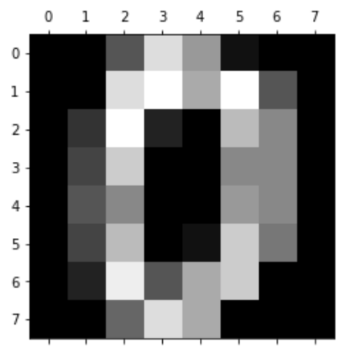
(b) original image

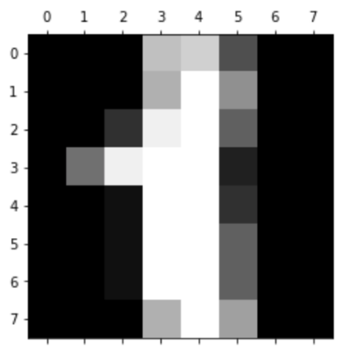
Figure 6: original image and its filtered one

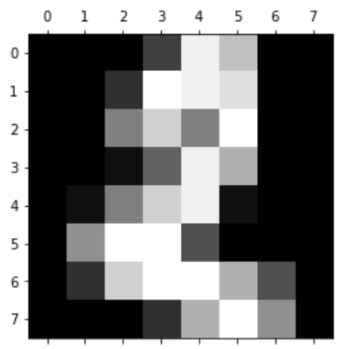
Problem 3

Part b:
Here is examples of digits we plotted in figure7.

Problem 3 continued on next page. . .


(a) Zero


(b) One


(c) Two

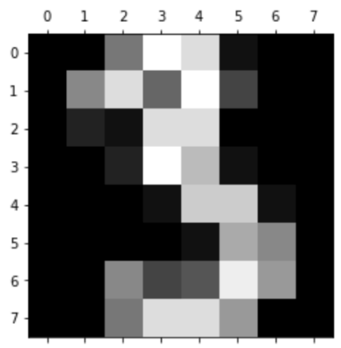

(d) Three

Figure 7: Some digits

Part c:
As we can see below, up to degree of polynomial=3or4 , accuracy of SVM increased as degree of polynomial increased but after that, increasing complexity is not useful because our complexity has to match with size of dataset and as our dataset is scarce so we get into the trouble of overfitting.

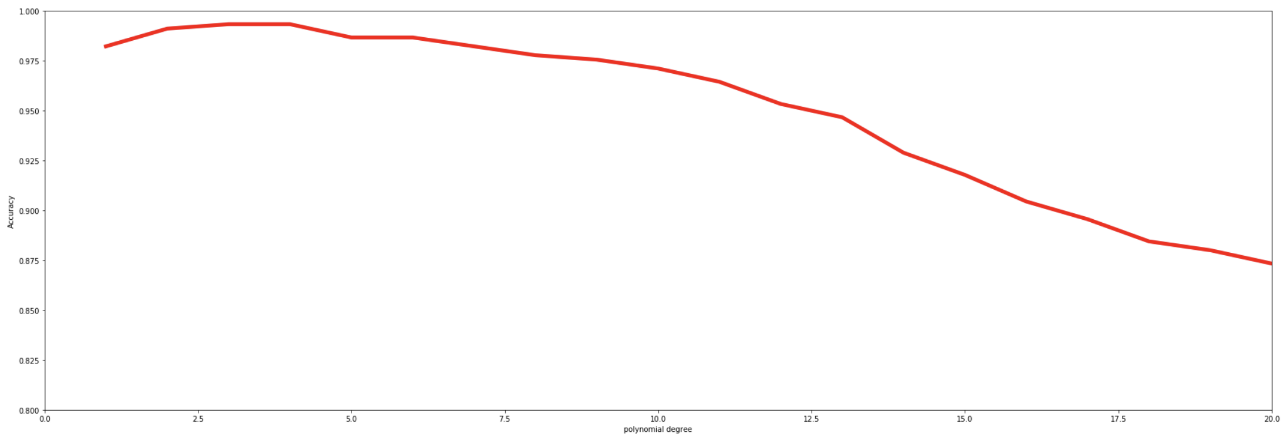
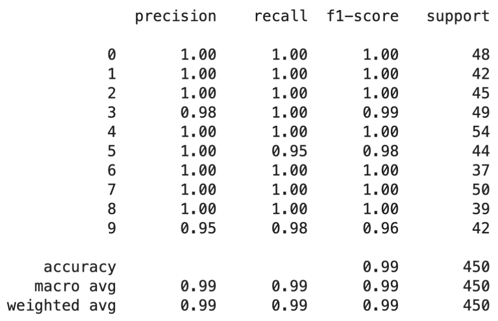


Figure 8: Accuracy of SVM as function of polynomial degree

Part d:
Here we plotted confusion matrix and also computed Sensitivity, Specificity, Precision, Accuracy and Dice

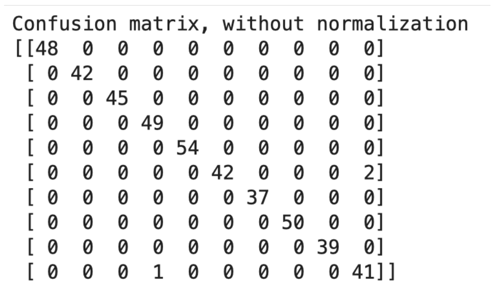
Problem 3 continued on next page. . .

indices and we also bring them below.For number one digit all of these parameters are one.Recall below is sensitivity and F1 score is Dice.We should mention that specificity or true negative for number one digit is also one.Concretely we can calculate these mentioned measure for other digits by below figures which for illustration we plotted two of them colorful.For polynomial degree of 3 and 4 we have accuracy of about 99% for SVM classifier.Random state is 12 for better accuracy.



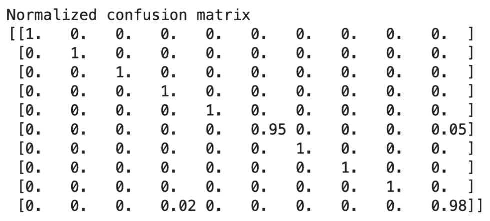
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        48
           1       1.00      1.00      1.00        42
           2       1.00      1.00      1.00        45
           3       0.98      1.00      0.99        49
           4       1.00      1.00      1.00        54
           5       1.00      0.95      0.98        44
           6       1.00      1.00      1.00        37
           7       1.00      1.00      1.00        50
           8       1.00      1.00      1.00        39
           9       0.95      0.98      0.96        42

    accuracy                           0.99       450
   macro avg       0.99      0.99      0.99       450
weighted avg       0.99      0.99      0.99       450
```

(a) Statistical parameters for evaluation



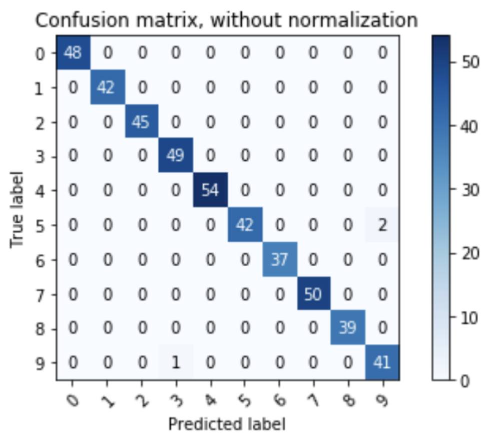
```
Confusion matrix, without normalization
[[48  0  0  0  0  0  0  0  0  0]
 [ 0 42  0  0  0  0  0  0  0  0]
 [ 0  0 45  0  0  0  0  0  0  0]
 [ 0  0  0 49  0  0  0  0  0  0]
 [ 0  0  0  0 54  0  0  0  0  0]
 [ 0  0  0  0  0 42  0  0  0  2]
 [ 0  0  0  0  0  0 37  0  0  0]
 [ 0  0  0  0  0  0  0 50  0  0]
 [ 0  0  0  0  0  0  0  0 39  0]
 [ 0  0  0  1  0  0  0  0  0 41]]
```

(b) Confusion matrix

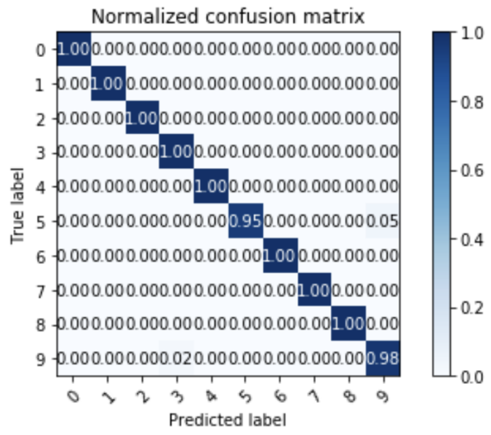


```
Normalized confusion matrix
[[1.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   1.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   1.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   1.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   1.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.95 0.   0.   0.   0.05]
 [0.   0.   0.   0.   0.   0.   1.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   1.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   1.   0.  ]
 [0.   0.   0.   0.02 0.   0.   0.   0.   0.   0.98]]
```

(c) Normalized confusion matrix



(d) Confusion matrix



(e) Normalized confusion matrix

Figure 9: Our Result which is for polynomial degree of **Three**