IDVE_EXAM_Q3

December 10, 2021

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[2]:
```
import numpy as np
import pandas as pd
import math
from scipy import stats
import matplotlib.pyplot as plt
```

[3]:
```
!pip install minisom
```

```
Requirement already satisfied: minisom in /usr/local/lib/python3.7/dist-packages
(2.2.9)
```

[4]:
```
mu = 0
std = 1
snd = stats.norm(mu, std)
```

[5]:
```
s = np.linspace(0, 2*np.pi, num=1000)
x1 = np.cos(s) + 0.1*stats.norm(mu, std).pdf(s)
x2 = np.sin(s) + 0.1*stats.norm(mu, std).pdf(s)
x3 = s + 0.1*stats.norm(mu, std).pdf(s)
```

[6]:
```
x1.shape, x2.shape, x3.shape
```

[6]:
```
((1000,), (1000,), (1000,))
```

[7]:
```
f = np.vstack((x1,x2,x3))
```

[8]:
```
f
```

[8]:
```
array([[ 1.03989423e+00,  1.03987366e+00,  1.03981196e+00, ...,
         9.99920886e-01,  9.99980221e-01,  1.00000000e+00],
       [ 3.98942280e-02,  4.61828723e-02,  5.24696898e-02, ...,
        -1.25786177e-02, -6.28943321e-03,  1.06728304e-10],
       [ 3.98942280e-02,  4.61829138e-02,  5.24700215e-02, ...,
         6.27060636e+00,  6.27689583e+00,  6.28318531e+00]])
```

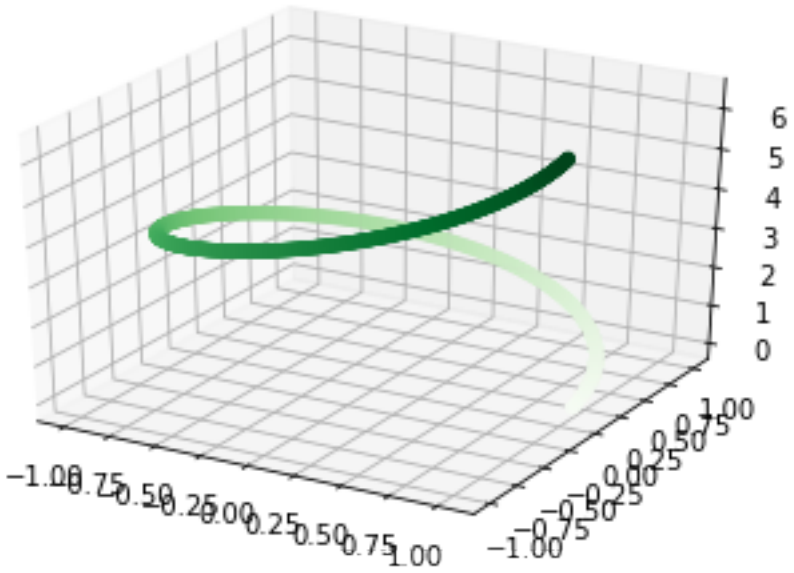
[9]:
```
z = np.array([x1,x2,x3])
```

0.0.1 3.1 3D Scatterplot

[10]:
```
ax = plt.axes(projection='3d')

# Data for a three-dimensional line

ax.scatter3D(f[0,:], f[1,:], f[2,:], c=f[2,:], cmap='Greens');
```



[11]:
```
import plotly.express as px

fig = px.scatter_3d(f, x=f[0,:], y=f[1,:], z=f[2,:],
              color=f[2,:])
fig.show()
```

[12]:
```
from minisom import MiniSom
```

[13]:
```
finaldf = []
for j in range(0,1000):
  finaldf.append([x1[j],x2[j],x3[j]])
finaldf = np.array(finaldf)
```

[14]:
```
finaldf.shape
```

[14]:
```
(1000, 3)
```

[15]:
```
finaldf = finaldf.reshape(3, 1000)
finaldf.shape
```

[15]:
```
(3, 1000)
```

0.0.2 3.2 Fit SOM to the Data

[16]:
```
def runSOM(sig):
  som = MiniSom(12, 12, 1000, sigma=sig, learning_rate=0.5) # initialization of 6x6 SOM
  som.train(finaldf, 100, verbose=True) # trains the SOM with 100 iterations
  return som.get_weights()
```

[17]:
```
def get_xyz(som_weights):
    x,y,z = som_weights[:,:,0],som_weights[:,:,1],som_weights[:,:,2]
    return x,y,z
```

[18]:
```
def plot_scatter3d(x,y,z,c='r'):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(x, y, z, c=c, marker='o')

    ax.set_xlabel('X Label')
    ax.set_ylabel('Y Label')
    ax.set_zlabel('Z Label')
    fig.show()
```

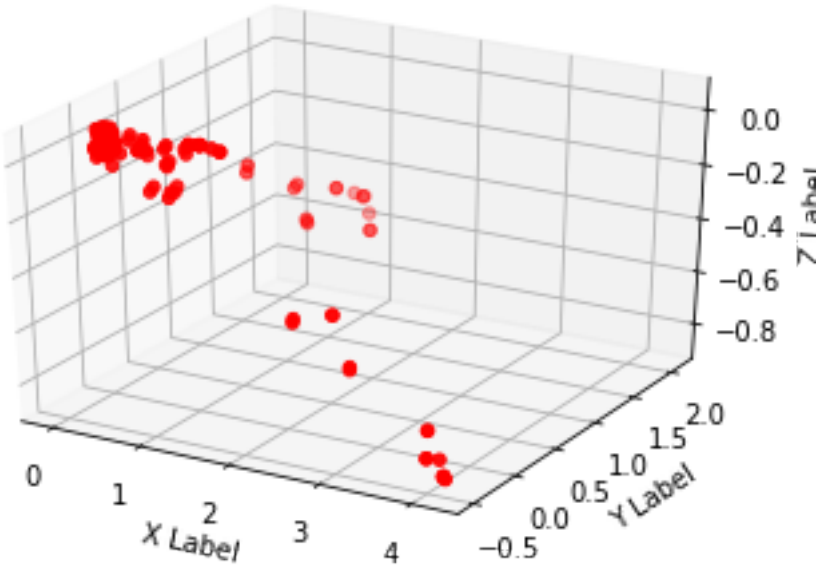
sigma = 1.5

[19]:
```
somWeights = runSOM(1.5)
```

```
[ 100 / 100 ] 100% - 0:00:00 left
quantization error: 0.008178773209443636
```

[20]:
```
x,y,z = get_xyz(somWeights)
```

[21]:
```
plot_scatter3d(x,y,z)
```



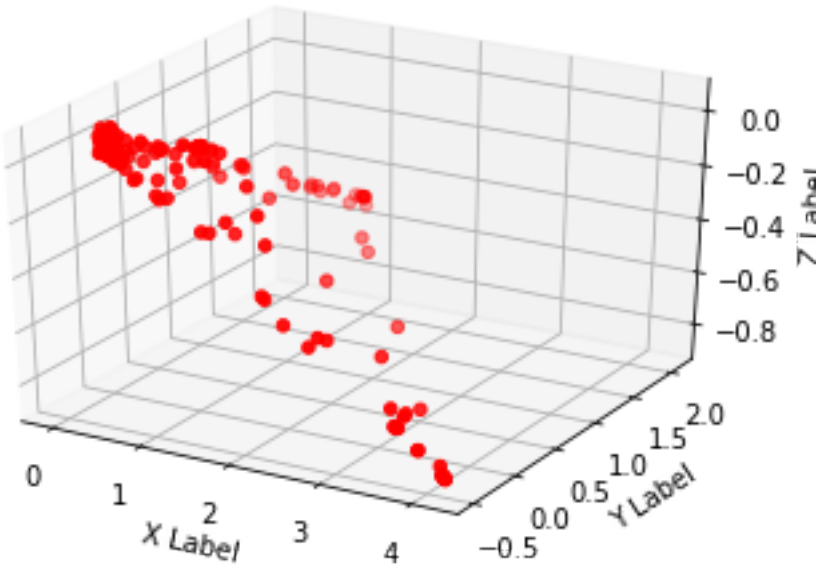
sigma = 2

[22]:
```
somWeights = runSOM(2)
```

```
[ 100 / 100 ] 100% - 0:00:00 left
quantization error: 0.0040476311439370735
```

[23]:
```
x,y,z = get_xyz(somWeights)
```

[24]:
```
plot_scatter3d(x,y,z)
```



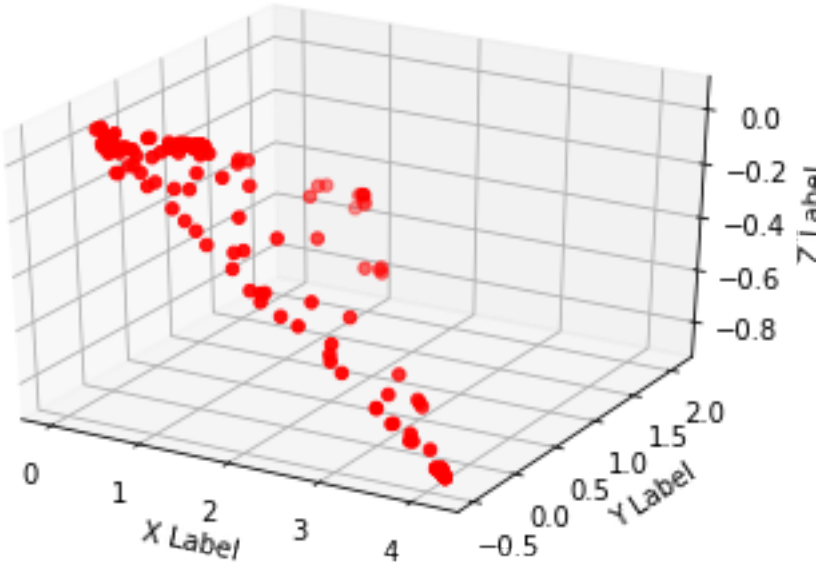
sigma = 3

[25]:
```
somWeights = runSOM(3)
```

```
[ 100 / 100 ] 100% - 0:00:00 left
quantization error: 0.00299271453602061
```

[26]:
```
x,y,z = get_xyz(somWeights)
```

[27]:
```
plot_scatter3d(x,y,z)
```



sigma = 4

[28]:
```
somWeights = runSOM(4)
```

```
[ 100 / 100 ] 100% - 0:00:00 left
quantization error: 0.003385865695732167
```

[29]:
```
x,y,z = get_xyz(somWeights)
```

[30]:
```
plot_scatter3d(x,y,z)
```