MLscan

May 25, 2022

[28]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import Sequential
# from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import Model,Sequential
from tensorflow.keras.layers import Flatten,Dense,Dropout
import os
import tensorflow as tf
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

#mirrored_strategy = tf.distribute.MirroredStrategy(devices=["/gpu:0", "/gpu:
↪1"])
## check for a GPU
if not tf.test.gpu_device_name():
    warnings.warn('No GPU found.....')
    sys.exit()
else:
    print('Default GPU device :{}'.format(tf.test.gpu_device_name()))
```

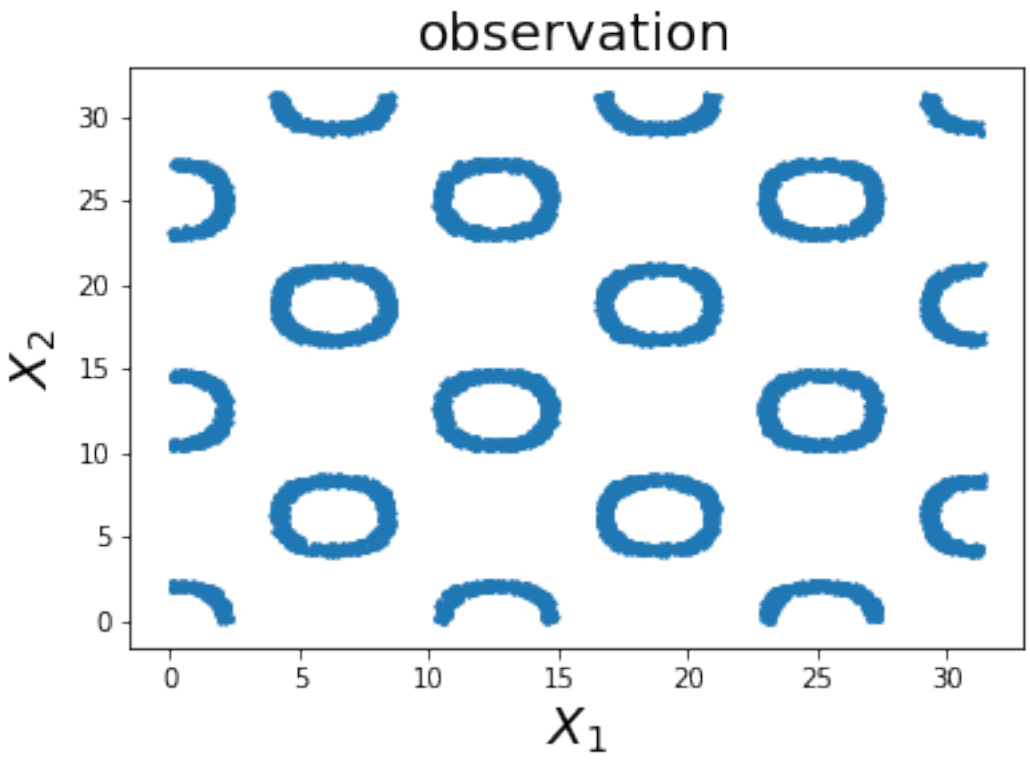
```
Default GPU device :/device:GPU:0
```

[30]:
```
class MLS(Model):
    def __init__(self,inshape,drop_rate):
        super(MLS,self).__init__()
        self.dense1 = Dense(100, input_shape=(inshape,))
        self.dense2 = Dense(100,activation='relu')
        self.dense4 = Dense(1)
        self.dropout= Dropout(drop_rate)
    def call(self,input):
        x = self.dense1(input)
        x = self.dense2(x)
        x = self.dense2(x)
        x = self.dense2(x)
        x = self.dense2(x)
        x = self.dense4(x)
```

```
        return x

model = MLS(2,0.1)
model.compile(optimizer='adam', loss='mse')
```

[31]:
```
def obs(x1,x2):
    F = (2+np.cos(x1/2)*np.cos(x2/2))**5
    return np.array(F)
def generate_init(n):
    x1,x2=[],[]
    for q in range(n):
        x1.append(np.random.uniform(0,10*np.pi))
        x2.append(np.random.uniform(0,10*np.pi))
    return np.array(x1),np.array(x2),np.array([x1,x2]).T


def likelihood(exp_value,std,th):
    ll = np.exp(- (exp_value - th)**2/(2*std**2))
    return ll
```

[41]:
```
x1_input,x2_input,_ = generate_init(100000)
val = obs(x1_input,x2_input)
pred1 = likelihood(100,20,val)

plt.scatter(x1_input[pred1>0.5],x2_input[pred1>0.5],s=2);
plt.xlabel(r'$X_1$',fontsize=20);
plt.ylabel(r'$X_2$',fontsize=20);
plt.title('observation',fontsize=20);
```

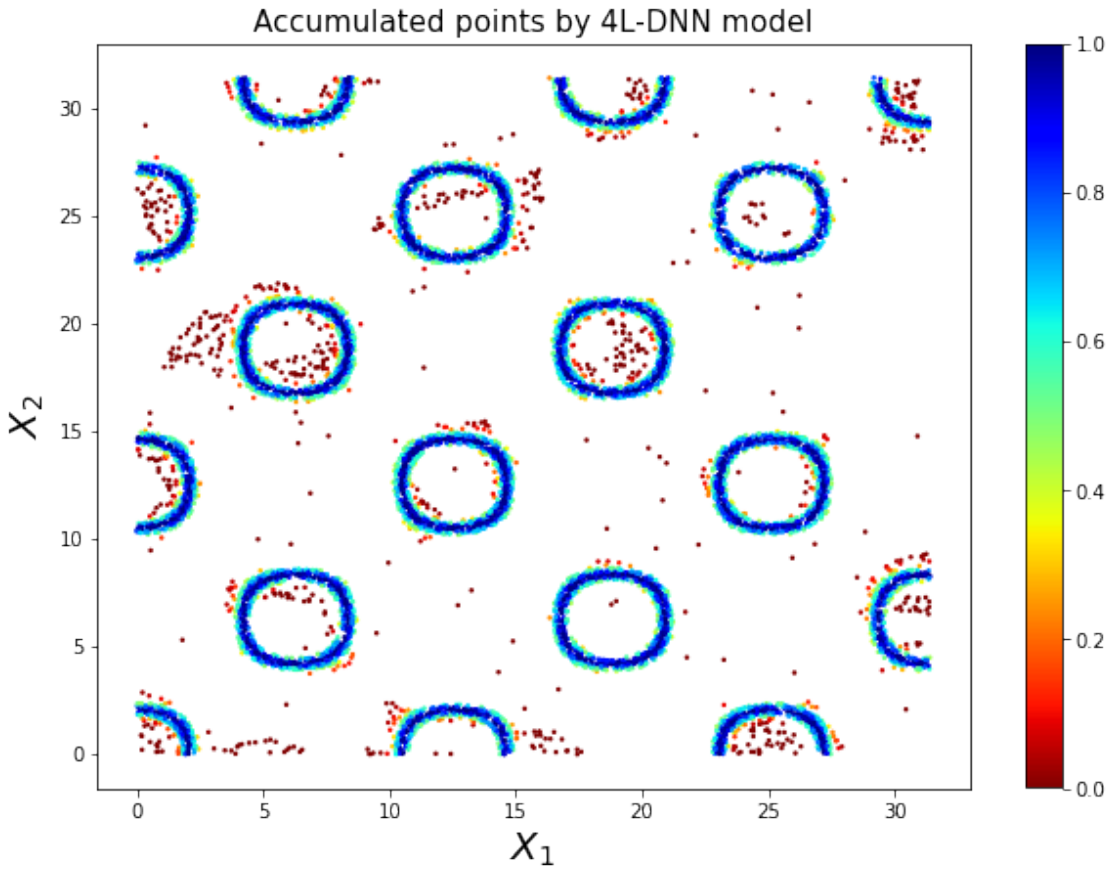


[52]:
```
runs = 10
epoch=1000

x1,x2,X=generate_init(100)
obs1 = obs(x1, x2)
model.fit(X, obs1, epochs=epoch, verbose=0)
for q in range(runs):
    _,_,x = generate_init(10000)
    pred = model.predict(x).flatten()
    pred1 = likelihood(100,20,pred)
    xsel = x[pred1>0.5]
    obs2 = obs(xsel[:, 0], xsel[:, 1])
    X = np.append(X, xsel, axis=0)
    obs1 = np.append(obs1, obs2)
    model.fit(X, obs1,epochs=epoch, batch_size=500,verbose=0)
    print('Run Number {} - Number of collected points= {}'.format(q,len(X)))
```

```
Run Number 1 - Number of collected points= 1370
Run Number 2 - Number of collected points= 2382
Run Number 3 - Number of collected points= 3341
Run Number 4 - Number of collected points= 4326
Run Number 5 - Number of collected points= 5335
Run Number 6 - Number of collected points= 6291
```

```
Run Number 7 - Number of collected points= 7300
Run Number 8 - Number of collected points= 8291
Run Number 9 - Number of collected points= 9270
Run Number 10 - Number of collected points= 10287
```

[60]:
```
plt.figure(figsize=(10,7))
plt.scatter(X[:, 0],X[:, 1],c=likelihood(100,20,obs(X[:, 0],X[:, 1])),
 ↪,s=2,cmap='jet_r')
plt.xlabel(r'$X_1$',fontsize=20);
plt.ylabel(r'$X_2$',fontsize=20);
plt.title('Accumulated points by 4L-DNN model',fontsize=15);
plt.colorbar();
```



[ ]: