ML for Earth System Science

Assignment 1

Utkarsh Dwivedi

21CL60R06

A1. Spatial Process Consider a square region of size 20x20 and 100 time-steps.

1) At t=1, select 10 random locations (likely to be different for all students). Put down observed values X(s,t) according to the formula X(s,t) = s1+s2+random noise, where s1, s2 are the horizontal and vertical positions of the location s in the square.

2) Using this initial data, use a Gaussian Process to generate data for the whole region (fitting the initial data) and for all the time-points.

3) Once the data has been generated, pick up a random set of 20 locations. Consider that you know the values at these locations for all time-points. Use Kriging to estimate the values at other locations. Compare the estimated values with generated values.

1) Randomly selected 10 locations and their values are listed below.

S_2/S_1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1									10						16					2
2						8													21	22
3		5																		
4								12										22		
5														19						
6											17									
7			21				14													
8																				
9																25				
10	11								19											
11									20			23							30	
12					17															
13																				
14							21					26								
15				19																
16																32			35	
17	18						24													
18																		37		
19			22																	
20										30										

X	S1	S2	Random Noise	S1+S2=Random Noise
19,2	19	2	0.967	21.967
4,15	4	15	0.547	19.547
8,10	8	10	0.973	18.973
20,2	2	20	0.714	22.714
14,5	14	5	0.698	19.698
7,7	7	7	0.216	14.216

18,18	18	18	0.976	36.976
8,4	8	4	0.253	12.253
6,2	6	2	0.006	8.006
10,11	10	11	0.434	21.434

```
# Data Generation
import numpy as np
x_train=[19,4,8,2,14,7,18,8,6,10]
y_train=[2,15,10,20,5,7,18,4,2,11]
rng = np.random.RandomState(4)
x_train=(x_train+rng.uniform(0, 1, 10)).reshape(-1,1)
y_train=(y_train+rng.uniform(0, 1, 10))
x_train
```

```
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF

kernel = 1 * RBF(length_scale=1.0, length_scale_bounds=(1e-2,1e+2 ))
gaussian_process = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=9)
gaussian_process.fit(x_train, y_train)
gaussian_process.kernel_
```

```
12.2**2 * RBF(length_scale=0.973)
```

```
# Data generation for 20 X 20
x=[]
for i in range(20):
  for j in range(20):
    x.append(i+j+2)
x1=np.array(x).reshape(-1,1)
x1
```

2) On fitting the above data and fitting the gaussian process on Radial Basis Function (RBF) kernel, I got the following parameters of the kernel. Using the trained gaussian model, spatial data for 100 time-steps, 20 x 20 data was generated.

```
[ ]  # Data generation for 20 X 20
     x=[]
     for i in range(20):
       for j in range(20):
         x.append(i+j+2)
     x1=np.array(x).reshape(-1,1)
     x1
```

```
     # predicting values for spatial locations for 100 time steps
     y_space=gaussian_process.sample_y(x1, 100)
     y_space
```

The length scale was taken as 1.0 and the length scale bounds was taken as 0.01 to 100.

12.2**2 * RBF(length_scale=0.973)

3) Selecting 20 random locations.

S1	S2	X=S1+S2+Random Noise
9	1	10.206
15	1	16.109
20	1	21.856
2	3	5.096
18	4	22.221
11	6	17.156
3	7	21.987
16	9	25.256
1	10	11.145
12	11	23.265
19	11	30.965
5	12	17.874
7	14	21.265
12	14	26.456
16	16	32.698
19	16	35.254
1	17	18.123
7	17	24.354
3	19	22.497
10	20	30.265

The above data has been splitted using a train-test-split and fitted via three models, viz. Support Vector Regressor (SVR), Random Forest Regressor (RVR) and Linear Regressor Model (LRM) and then the trained model (SVR) is used to generate values for the various time points.

```
# 3D creation 20 x 20 x 100
y_space_3d=[]
for i in range(100):
  arr=y_space[:,i]
  y_space_3d=np.hstack((y_space_3d,arr))
y_space_3d=y_space_3d.reshape(20,20,100)
y_space_3d.shape
```

```
(20, 20, 100)
```

```
# Krigging
!pip install pykrige
```

```
import sys

from sklearn.svm import SVR
Loading... rn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

from pykrige.rk import RegressionKriging

svr_model = SVR(C=0.1, gamma="auto")
rf_model = RandomForestRegressor(n_estimators=100)
lr_model = LinearRegression(normalize=True, copy_X=True, fit_intercept=False)

models = [svr_model, rf_model, lr_model]

# take the first 5000 as Kriging is memory intensive
p = y_space_3d[:5000, :-2]
x = y_space_3d[:5000, -2:]
target = y_space_3d[:5000]
```

```
p_train, p_test, x_train, x_test, target_train, target_test = train_test_split(
    p, x, target, test_size=0.3, random_state=42
)

for m in models:
    print("=" * 40)
    print("regression model:", m.__class__.__name__)
    m_rk = RegressionKriging(regression_model=m, n_closest_points=10)
    m_rk.fit(p_train, x_train, target_train)
    print("Regression Score: ", m_rk.regression_model.score(p_test, target_test))
    print("RK score: ", m_rk.score(p_test, x_test, target_test))
```

```
========================================
regression model: SVR
Finished learning regression model
Finished kriging residuals
Regression Score:  -0.03405385545698292
RK score:  0.6619557666501965
========================================
regression model: RandomForestRegressor
Finished learning regression model
Finished kriging residuals
Regression Score:  0.7001356014102058
RK score:  0.7430821496025737
```

```
regression model: LinearRegression
Finished learning regression model
Regression Score:  0.5277968398381674
RK score:  0.6049089336167248
```

A2. Spatio-temporal process Repeat the initialization of step 1 of A1, at 5 random time-points and 5 spatial locations (25 points overall), setting X(s,t) = s1+s2+0.5*t + random noise. Repeat step 2 using

spatio-temporally separable Gaussian Processes (i.e. decompose as a product of two GPs – one spatial and one temporal). Explain how the generated dataset is different from the dataset obtained in Step 2.

S1	S2	T	NOISE	X
4	1	97	0.583025	54.08302
17	12	27	0.741657	43.24166
4	2	37	0.513672	25.01367
5	17	81	0.407779	62.90778
3	9	95	0.892278	60.39228
13	7	22	0.078694	31.07869
13	20	36	0.269916	51.26992
2	2	44	0.21451	26.21451
1	4	25	0.670796	18.1708
16	11	15	0.653553	35.15355
16	9	88	0.01093	69.01093
12	4	66	0.366307	49.36631
8	1	59	0.646959	39.14696
7	1	84	0.295364	50.29536
10	12	13	0.214462	28.71446
4	13	6	0.900563	20.90056
7	8	80	0.827718	55.82772
18	5	32	0.467993	39.46799
1	18	2	0.139591	20.13959
15	4	13	0.058188	25.55819
14	5	1	0.662821	20.16282
1	14	87	0.918798	59.4188
2	11	23	0.715437	25.21544
15	1	4	0.766384	18.76638
10	11	93	0.467417	67.96742

The generated dataset in this is different from the previous state as in the previous step, we knew values for a particular dimension at every time point. Whereas, in this we have data for random time-steps and locations.