Coursework 5-Shizhi Chen-10307389

Evidence for Global Warming?

We're going to investigate the question: Is the summer minimum of sea-ice cover getting deeper?

1. Fit GP regression models for each decade separately.

2. From each model (that is, for each decade, separately), generate a large number of random annual traces and estimate the minimum of the sea ice cover over the year. Accumulate lots of these minima and form two distributions—one per decade—from them.

3. Decide whether it's plausible that the minima generated from the 2004 model are compatible with those from the 1990 one and, if you think they're different, justify your claim by measuring how different they are.

Python Code and Output:

Prepare the training data for each decade.

```python
# Import some useful packages
import math # for floor()
import numpy as np
import pandas as pd
import datetime as dt
import scipy.stats as sp
import matplotlib.pyplot as plt
```

```python
# Get the data into a Pandas dataframe
csv_file = "All_GP_training.csv"
seaIce_df = pd.read_csv( csv_file )
```

```python
# Separate two decades of dataset
boolIdx = np.any([(seaIce_df['Decade'] == 1990)], axis=0)
boo2Idx = np.any([(seaIce_df['Decade'] == 2004)], axis=0)
seaIce1990_df = seaIce_df.loc[boolIdx,:]
seaIce2004_df = seaIce_df.loc[boo2Idx,:]
```

Work out the ordinal dates corresponding to the rows in the data frame

```
# Convert the days and months into ordinal dates
# by looping over the rows of the data frame
n_rows, n_cols = seaIce1990_df.shape
day_in_year1990 = np.zeros( n_rows )

year = 1990 # any non-leapyear will do
lastDayOfPrevYear = dt.date( year-1, 12, 31 )

row_num = 0
for index, row in seaIce1990_df.iterrows():
    row_date = dt.date( year, int(row['Month']), int(row['Day']) )
    day_in_year1990[row_num] = row_date.toordinal() - lastDayOfPrevYear.toordinal()
    row_num += 1

# The indexing of Pandas data frames is a bit fussy
seaIce1990_df.loc[:,'Ordinal.Date'] = day_in_year1990
seaIce1990_df.head()
```

	Day	Month	Decade	Mean.Extent	Var.Extent	Ordinal.Date
0	1	1	1990	13.966625	0.058241	1.0
1	15	1	1990	14.597125	0.036833	15.0
2	1	2	1990	15.198375	0.066743	32.0
3	15	2	1990	15.468875	0.024467	46.0
4	1	3	1990	15.613250	0.061991	60.0

```
n_rows, n_cols = seaIce2004_df.shape
day_in_year2004 = np.zeros( n_rows )

year = 2004 # any non-leapyear will do
lastDayOfPrevYear = dt.date( year-1, 12, 31 )

row_num = 0
for index, row in seaIce2004_df.iterrows():
    row_date = dt.date( year, int(row['Month']), int(row['Day']) )
    day_in_year2004[row_num] = row_date.toordinal() - lastDayOfPrevYear.toordinal()
    row_num += 1

# The indexing of Pandas data frames is a bit fussy
seaIce2004_df.loc[:,'Ordinal.Date'] = day_in_year2004
seaIce2004_df.head()
```

	Day	Month	Decade	Mean.Extent	Var.Extent	Ordinal.Date
24	1	1	2004	13.097000	0.014613	1.0
25	15	1	2004	13.683429	0.054808	15.0
26	1	2	2004	14.220000	0.028377	32.0
27	15	2	2004	14.613714	0.032851	46.0
28	1	3	2004	14.812571	0.087765	61.0

Fit GP regression model for 1990 and 2004

```
# Import the things we'll need
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF
```

```
initial_ell_1990 = (np.amax(seaIce1990_df['Ordinal.Date'])
                    - np.amin(seaIce1990_df['Ordinal.Date'])) / 10
initial_scale = np.var( seaIce1990_df['Mean.Extent'] ) # var of the training data
rbf_kernel_1990 = initial_scale * RBF(length_scale=initial_ell_1990,
    length_scale_bounds=(initial_ell_1990/20, 5 * initial_ell_1990))
gp1990 = GaussianProcessRegressor( kernel=rbf_kernel_1990,
                                  alpha=seaIce1990_df['Var.Extent'] )

# Fit things
x = seaIce1990_df['Ordinal.Date'][:, np.newaxis] # makes x into a column vector
y = seaIce1990_df['Mean.Extent']
gp1990.fit( x, y )
```

```
initial_ell_2004 = (np.amax(seaIce2004_df['Ordinal.Date'])
                    - np.amin(seaIce2004_df['Ordinal.Date'])) / 10
initial_scale = np.var( seaIce2004_df['Mean.Extent'] ) # var of the training data
rbf_kernel_2004 = initial_scale * RBF(length_scale=initial_ell_2004,
    length_scale_bounds=(initial_ell_2004/20, 5 * initial_ell_2004))
gp2004 = GaussianProcessRegressor( kernel=rbf_kernel_2004,
                                  alpha=seaIce2004_df['Var.Extent'] )
# Fit things
x = seaIce2004_df['Ordinal.Date'][:, np.newaxis] # makes x into a column vector
y = seaIce2004_df['Mean.Extent']
gp2004.fit( x, y )
```

```
# Get the fitted A and ell from the posterior
fitted_kernel_1990 = gp1990.kernel_
fitted_params_1990 = fitted_kernel_1990.get_params()
A_1990 = math.sqrt( fitted_params_1990["k1__constant_value"] )
ell_1990 = fitted_params_1990["k2__length_scale"]
```

```
fitted_kernel_2004 = gp2004.kernel_
fitted_params_2004 = fitted_kernel_2004.get_params()
A_2004 = math.sqrt( fitted_params_2004["k1__constant_value"] )
ell_2004 = fitted_params_2004["k2__length_scale"]
```

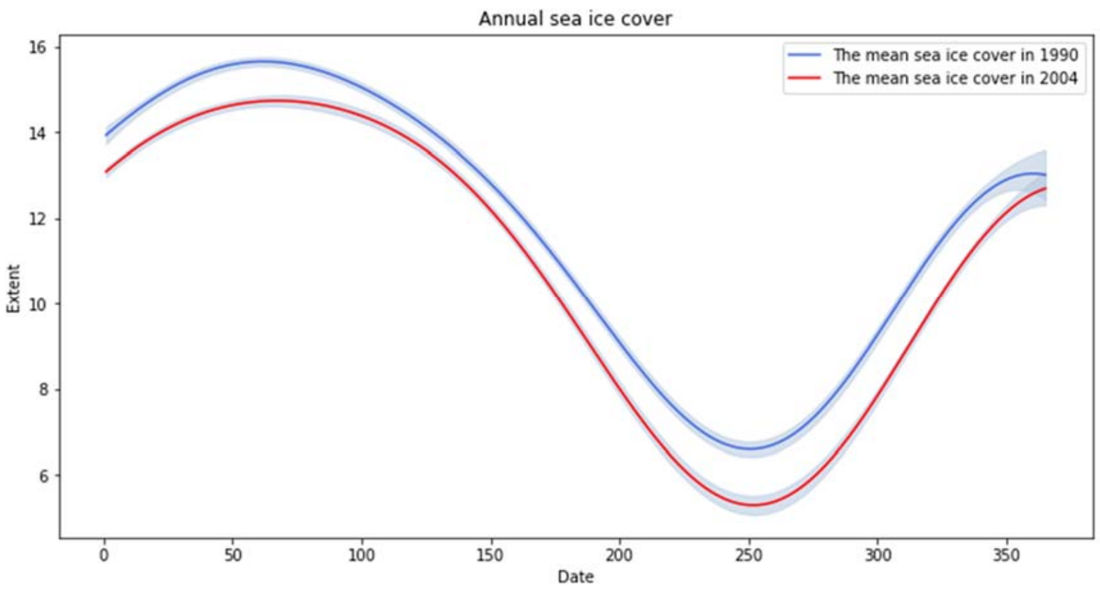
```
# Make predictions at positions listed in a vector xVals

ordinal_dates = np.linspace( 1, 365, 365 ) # Build the N-by-1 array
date_col_vec = ordinal_dates[:, np.newaxis]
y_mean1, y_std1 = gp1990.predict(date_col_vec, return_std=True)
y_mean2, y_std2 = gp2004.predict(date_col_vec, return_std=True)

# Plot the mean with a +/- one std_dev shaded band

plt.figure(figsize=(16,6))
plt.title("Annual sea ice cover")
plt.xlabel("Date")
plt.ylabel("Extent")


plt.plot(ordinal_dates, y_mean1, color="royalblue",
         label="The mean sea ice cover in 1990")
plt.plot(ordinal_dates, y_mean2, color="red",
         label="The mean sea ice cover in 2004")
plt.legend()
plt.fill_between(ordinal_dates, y_mean1 - y_std1, y_mean1 + y_std1,
                 alpha=0.5, color="lightsteelblue" )
plt.fill_between(ordinal_dates, y_mean2 - y_std2, y_mean2 + y_std2,
                 alpha=0.5, color="lightsteelblue" )
plt.show()
```

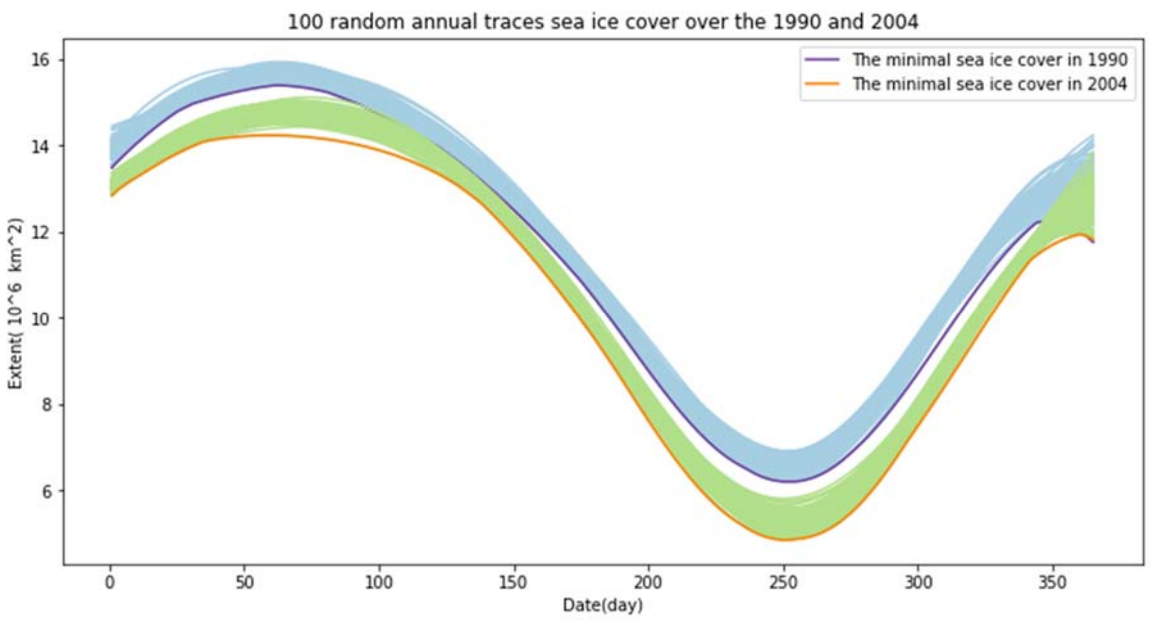


Generate a large number of random annual traces and estimate the minimum of the sea ice cover over the year.

```
# Construct and plot random functions drawn from the posterior
nSamples = 100
y_samples1990 = gp1990.sample_y( date_col_vec, nSamples )
y_samples2004 = gp2004.sample_y( date_col_vec, nSamples )

plt.figure(figsize=(12,6))
pairCmap = plt.get_cmap( name='Paired' )
pairedColors = pairCmap.colors
plt.title("100 random annual traces sea ice cover over the 1990 and 2004")
plt.xlabel("Date(day)")
plt.ylabel("Extent( 10^6  km^2)")

plt.plot( date_col_vec, y_samples1990,color=pairedColors[0] )
plt.plot( date_col_vec, np.min(y_samples1990,axis=1),color=pairedColors[9],
         label="The minimal sea ice cover in 1990" )
plt.plot( date_col_vec, y_samples2004,color=pairedColors[2] )
plt.plot( date_col_vec, np.min(y_samples2004,axis=1),color=pairedColors[7],
         label="The minimal sea ice cover in 2004" )
plt.legend()
plt.show()
```



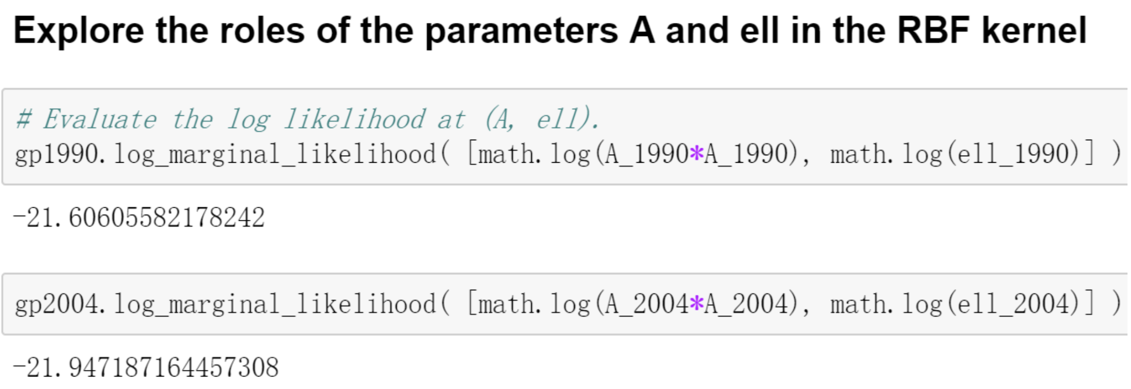
Discussion whether you think the minima are getting deeper

```
minseaIce_1990=np.min(y_samples1990)
minseaIce_2004=np.min(y_samples2004)
meanseaIce_1990=np.mean(y_samples1990)
meanseaIce_2004=np.mean(y_samples2004)


print("The minimum sea ice cover in 1990: ", minseaIce_1990)
print("The minimum sea ice cover in 2004: ", minseaIce_2004)
print("The average sea ice cover in 1990: ", meanseaIce_1990)
print("The average sea ice cover in 2004: ", meanseaIce_2004)
declinePer_min=(minseaIce_1990-minseaIce_2004)/minseaIce_1990
print(declinePer_min)
declinePer_mean=(meanseaIce_1990-meanseaIce_2004)/meanseaIce_1990
print(declinePer_mean)
```

```
The minimum sea ice cover in 1990:  6.196819478648556
The minimum sea ice cover in 2004:  4.847020601628741
The average sea ice cover in 1990:  11.751600899803828
The average sea ice cover in 2004:  10.77461779656006
0.21782123582437937
0.083136171111808
```

As can be seen from the above image, the 2004 sea ice cover was significantly lower than in 1990, especially in the summer. From the data, we can see that the average annual sea ice cover for 1990 was 11.75, and the figure for 2004 was 10.77, a decrease of 8.31%. The situation that the Sea-ice cover became lower became more obvious in the summer. The lowest sea ice cover in the summer of 1990 was about 6.20, this figure was only 4.85 in 2004, this means that the sea ice cover was down by 21.78% during these years.



Explore the roles of the parameters A and ell in the RBF kernel

```
# Evaluate the log likelihood at (A, ell).
gp1990.log_marginal_likelihood( [math.log(A_1990*A_1990), math.log(ell_1990)] )
```

```
-21.60605582178242
```

```
gp2004.log_marginal_likelihood( [math.log(A_2004*A_2004), math.log(ell_2004)] )
```

```
-21.947187164457308
```

The optimizing (Maximum absolute value) LML values of two decades gaussian process are shown above.

Here we use sklearn.gaussian_process fits A and ell automatically, so we got the optimal A and ell in our Gaussian process. While the hyperparameters chosen by optimizing LML have a considerable larger LML, they perform slightly worse according to the log-loss on training data. Therefore, when we obtain the largest log marginal likelihood the A and ell for our model is the optimality.