Gaussian Process Posterior (Python)

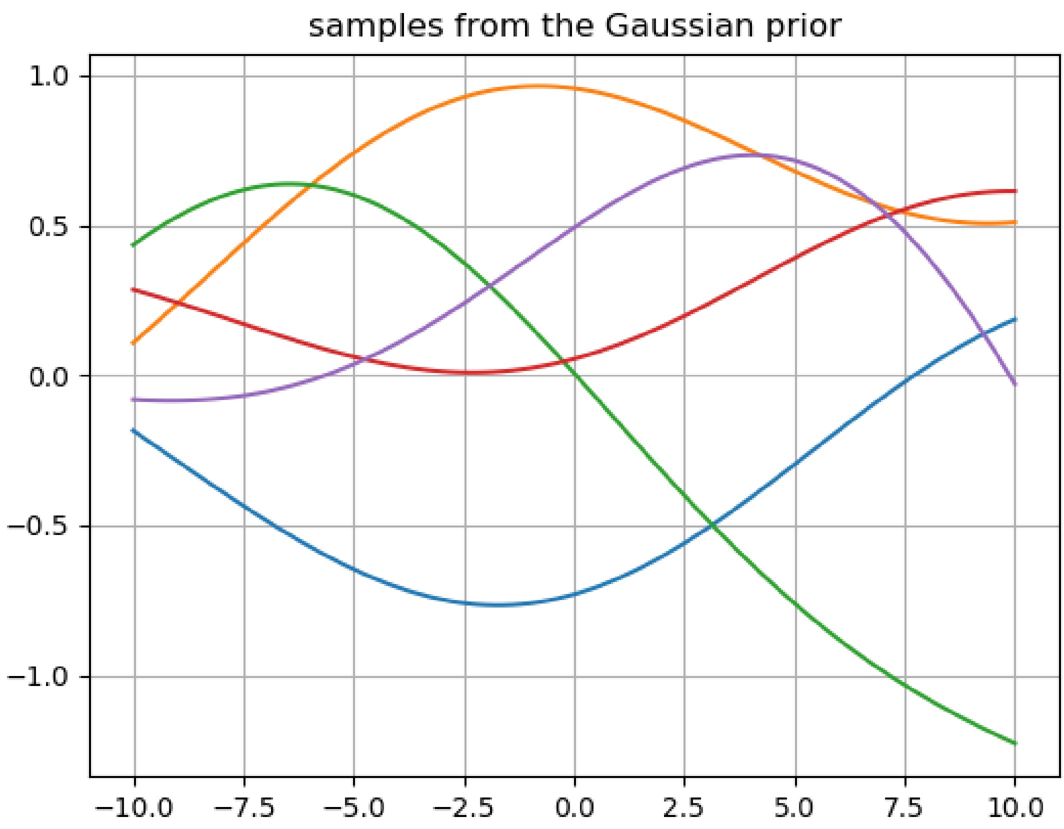
Asked 4 years ago Active 4 years ago Viewed 1k times

4

1

I have created and sampled a jointly Gaussian prior with mean=0 using the code below:

```
import numpy as np
import matplotlib.pyplot as plt
from math import pi
from scipy.spatial.distance import cdist
import scipy.stats as sts

x_prior = np.linspace(-10,10,101)
x_prior = x_prior.reshape(-1,1)
mu = np.zeros(x_prior.shape)

#defining the Kernel for the covariance function

def sec(a,b, length_scale , sigma) :
    K = sigma * np.exp(-1/(2*length_scale) * cdist(a,b)**2)
    return K

#defining the Gaussian Process prior

def GP(a , b, mu , kernel , length_scale, sigma , samples ) :
    f = np.random.multivariate_normal(mu.flatten(), kernel(a ,b , length_scale , sigma
) , samples)
    return f

prior = GP(x_prior ,x_prior, mu , sec , 100, 1 , 5)

plt.figure()
plt.grid()
plt.title('samples from the Gaussian prior')
plt.plot(x_prior , prior.T)
plt.show()
```



Then, when adding in some 'observed' data, I wish to compute the posterior over these points but this is where I become stuck.

Here's my code for introducing new data:

```
x_train = np.array([-10,-8,5,-1,2])
x_train = x_train.reshape(-1,1)
def straight_line(m , x , c):
    y = 5*x + c
    return y
ytrain = straight_line(5 , x_train , 0)
```

It's my understanding that you calculate a conditional distribution over the new data given the prior and new x values associated with the observed data.

Do you then wish to update the multivariate prior to become the posterior by performing some sort of change to the mean values to include the new y values?

I have used the following resources to try and attempt this:

http://katbailey.github.io/post/gaussian-processes-for-dummies/
https://www.robots.ox.ac.uk/~mebden/reports/GPtutorial.pdf

but I'm really trying to understand what happens at each stage, and why, so that when I get a posterior (which I can't do) I know exactly how I got there.

Here's some solutions I've been trying to implement but so far no avail:

```
K_train = sec(x_train , x_train , 1,1)
K_prior = sec(x_prior , x_prior , 1,1)
K_pt =  sec(x_prior , x_train , 1,1)
K_tp = sec(x_train , x_prior ,  1,1)  ## = k_tp transpose
prior = sts.multivariate_normal(mu.flatten(), K_prior)
#mean_test = np.dot(K_p , np.linalg.inv(K_prior))
mean_function = np.dot(np.dot(K_tp ,np.linalg.inv(K_prior).T) , prior )
covariance_function = K_train - np.dot(np.dot(K_tp ,np.linalg.inv(K_prior).T) , K_pt)
```

python process gaussian sampling Edit tags

Share Edit Follow Close Flag

asked Nov 27 '17 at 18:08
user8188120
705 1 9 20

2 Answers

Active Oldest Votes

1

Just for additional follow-up. I have written my code into a Juypiter format here:

https://github.com/SpaceMeerkat/Scariff

with associated read-through material here:

https://spacemeerkat.wordpress.com/

Just in case anyone wanted to work through this kind of material and became stuck like I did.

Share Edit Follow Flag

answered Dec 7 '17 at 17:02
user8188120
705 1 9 20

0

Just an update for anyone who looked at this. I found the solution reading this paper:

https://arxiv.org/pdf/1711.10834.pdf

and the following code:

```
mean_function = np.dot(np.dot(K_pt ,np.linalg.inv(K_train)), ytrain)

covariance_function = K_prior - np.dot(np.dot(K_pt ,np.linalg.inv(K_train)) , K_tp)

f = np.random.multivariate_normal(mean_function[:,0],covariance_function , 100)
```

where f is the posterior joint Gaussian from which you sample from

Share Edit Follow Flag

answered Nov 30 '17 at 11:32

user8188120
705 1 9 20