GP - Rasmussen & Williams - Ch. 2: Regression

Outline

GP prior

$$k(x,y) = \exp(-\tfrac{1}{2}|x-y|^2) \quad (1)$$

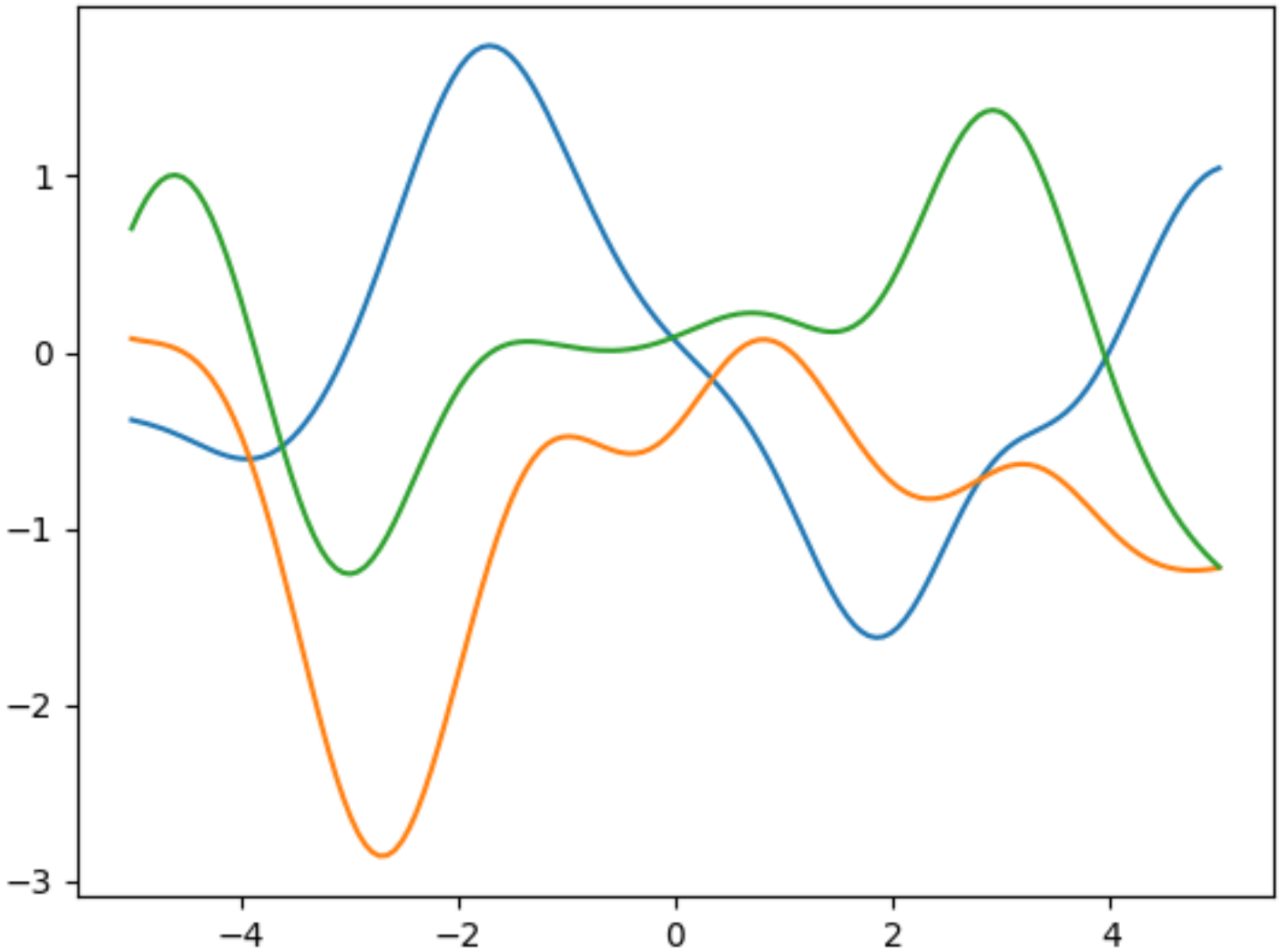
$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, K(\mathbf{x},\mathbf{x})) \quad (2)$$

Sampling from prior: Python code

```python
from numpy import sum, eye, exp #, zeros
from numpy.linalg import cholesky
from numpy.random import normal #, multivariate_normal

def rbf(length_scale):
    def k(x,y):
        if len(x.shape)==1:
            d = 1
        else:
            d  = x.shape[1]
        lx = x.shape[0]
        ly = y.shape[0]
        dists =  sum(((x.T.reshape([d,lx,1]) -  y.T.reshape([d,1,ly]))/length_scale)**2,0)
        return exp(-.5 * dists)
    return k

def genSamplesSimple(x, k):
    n = x.shape[0]
    L = cholesky(k(x,x)+eye(n)*1e-8)
    return L.dot(normal(size=n))
# Same as:
#    return multivariate_normal(zeros(n), k(x,x) + eye(n)*1e-8)
```

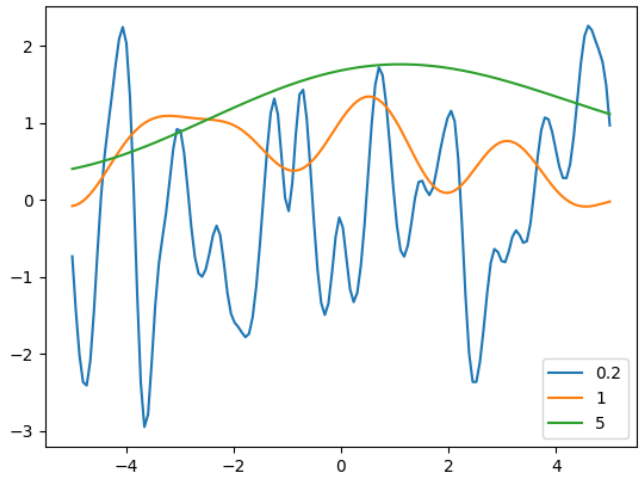
```python
from matplotlib.pyplot import figure, plot, savefig, close, legend
from numpy import linspace

figure()
x = linspace(-5,5,150)
k = rbf(1)
for i in range(3): plot(x, genSamplesSimple(x,k));
```

Random functions in 1D



Different length scales

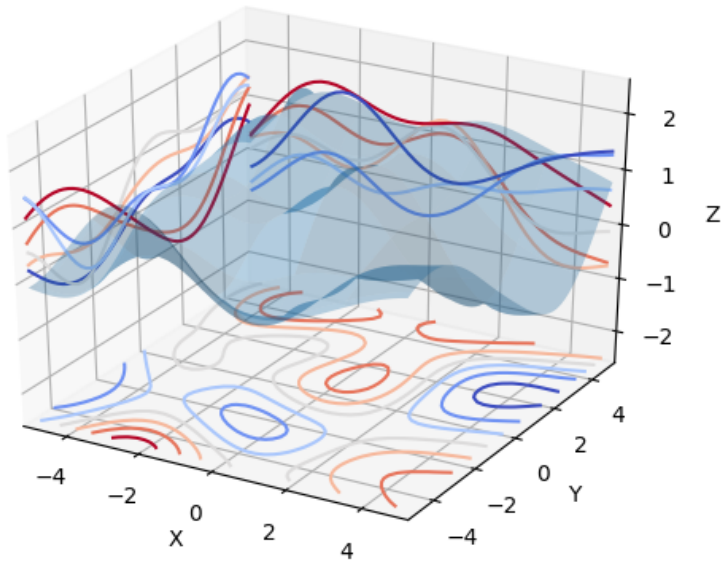
```
scales = [0.2, 1, 5]
for i in scales:
  plot(x, genSamplesSimple(x,rbf(i)))
legend(scales)
```



Two dimensions

```
from numpy import meshgrid, concatenate

x = linspace(5, -5, 50)
xx, yy = meshgrid(x, x)
xy = concatenate([xx.reshape([1, -1]),
                  yy.reshape([1, -1])]).T
z = genSamplesSimple(xy, rbf(2)).reshape([50, 50]
```

```
fig = figure()
ax = fig.gca(projection='3d')
ax.plot_surface(xx, yy, z, rstride=8,
                cstride=8, alpha=0.3)
cset = ax.contour(xx, yy, z, zdir='z',
                  offset=-2.5, cmap=cm.coolwarm)
cset = ax.contour(xx, yy, z, zdir='x',
                  offset=-5, cmap=cm.coolwarm)
cset = ax.contour(xx, yy, z, zdir='y',
                  offset=5, cmap=cm.coolwarm)

ax.set_xlabel('X')
ax.set_xlim(-5, 5)
ax.set_ylabel('Y')
ax.set_ylim(-5, 5)
ax.set_zlabel('Z')
ax.set_zlim(-2.5, 2.5)
```



Computing posterior and sampling: Python code

```python
from numpy import pi, eye, log, diag
from numpy.random import normal
from numpy.linalg import cholesky, solve #, inv
# solve(A,b) equals inv(A)*v, but it is more robust

def compPosterior(y, x, k, X, snoise):
    n = x.shape[0]
    K = k(x,X)
    L = cholesky(k(x, x) + eye(n)*(snoise + 1e-8))
    alpha = solve(L.T,solve(L,y))
    f_mean = K.T.dot(alpha)
    v = solve(L,K)
    V = k(X,X) - v.T.dot(v)
    log_p = -.5*y.T.dot(alpha) - sum(log(diag(L))) - .5*n*log(2*pi)
    return f_mean, V, log_p

def genSamples(x, m, K):
    n = x.shape[0]
    L = cholesky(K+eye(n)*1e-8)
    return m + L.dot(normal(size=n))
```

Fitting some data

```
from numpy import array, sqrt

X = linspace(-4, 4, 150)
k = rbf(1)
x = array([-2, 0, 0.1,   1, 3])
y = array([ 1, 0,   1, 0.6, 1])

f_m, V = compPosterior(y, x, rbf(1), X, 0.01)
s = sqrt(diag(V))

plot(X, f_m, 'r')
plot(X, f_m+2*s, 'r--')
plot(X, f_m-2*s, 'r--')
for i in range(5):
  plot(X, genSamples(X, f_m, V))
plot(x,y,'ob')
```