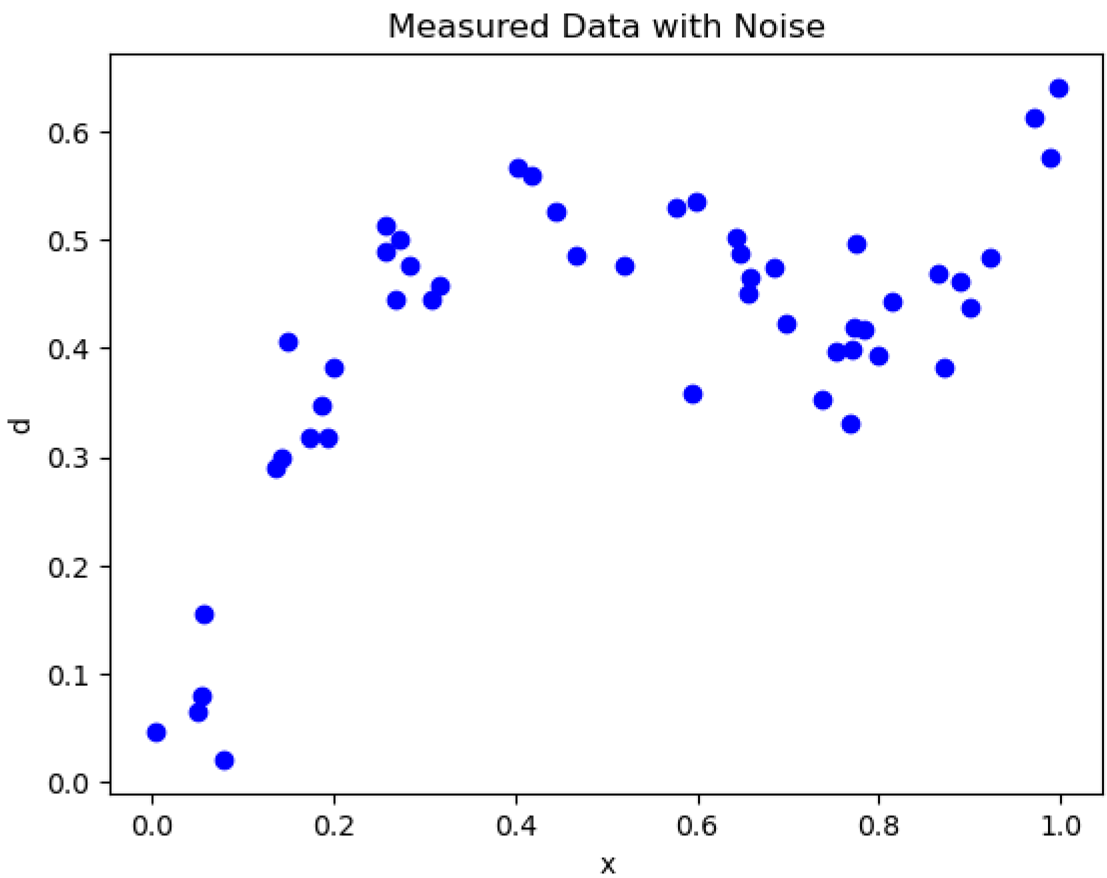
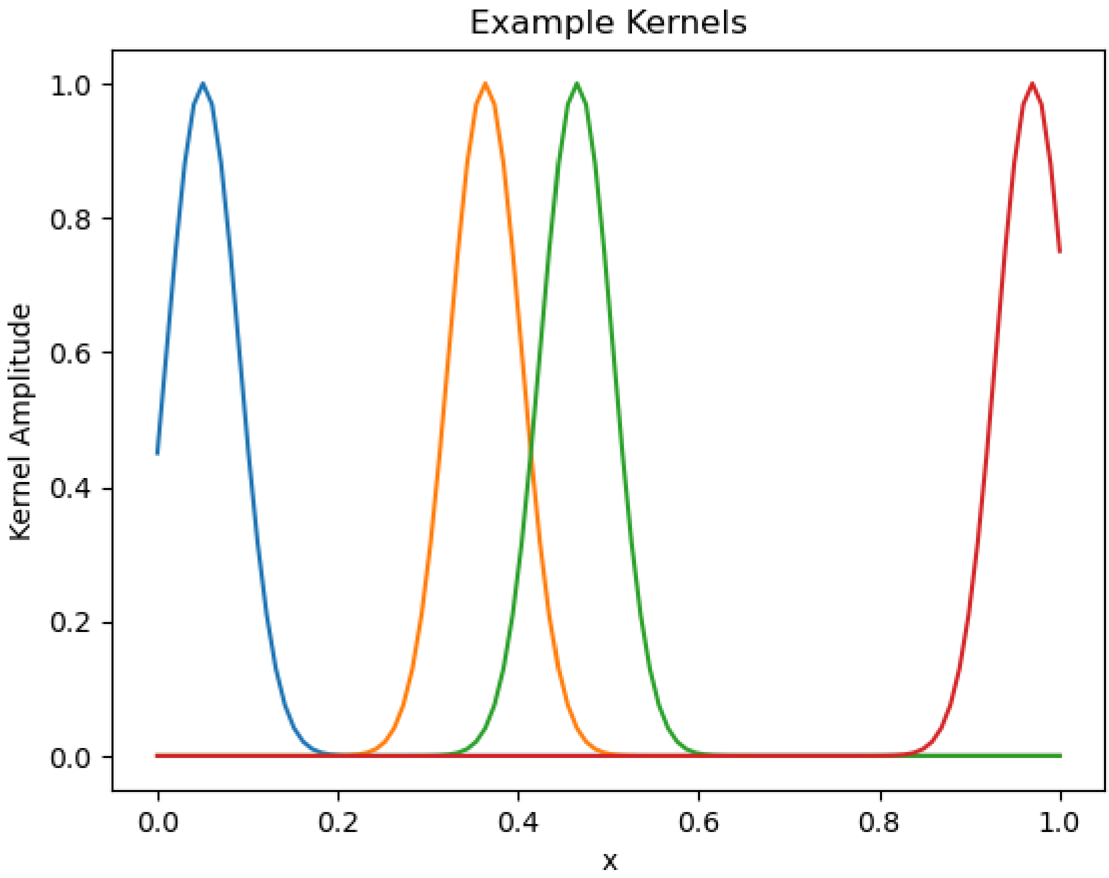
In [14]:
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(1024)  # ensure same noise for each run

# number of training points
n = 50

# sample n random points between 0 and 1
x = np.random.rand(n,1)

# set d = x^2 + .4 sin(1.5 pi x) + noise
d = x*x + 0.4*np.sin(1.5*np.pi*x) +0.04*np.random.randn(n,1)

# plot result
plt.plot(x,d,'bo')
plt.xlabel('x')
plt.ylabel('d')
plt.title('Measured Data with Noise')
plt.show()
```



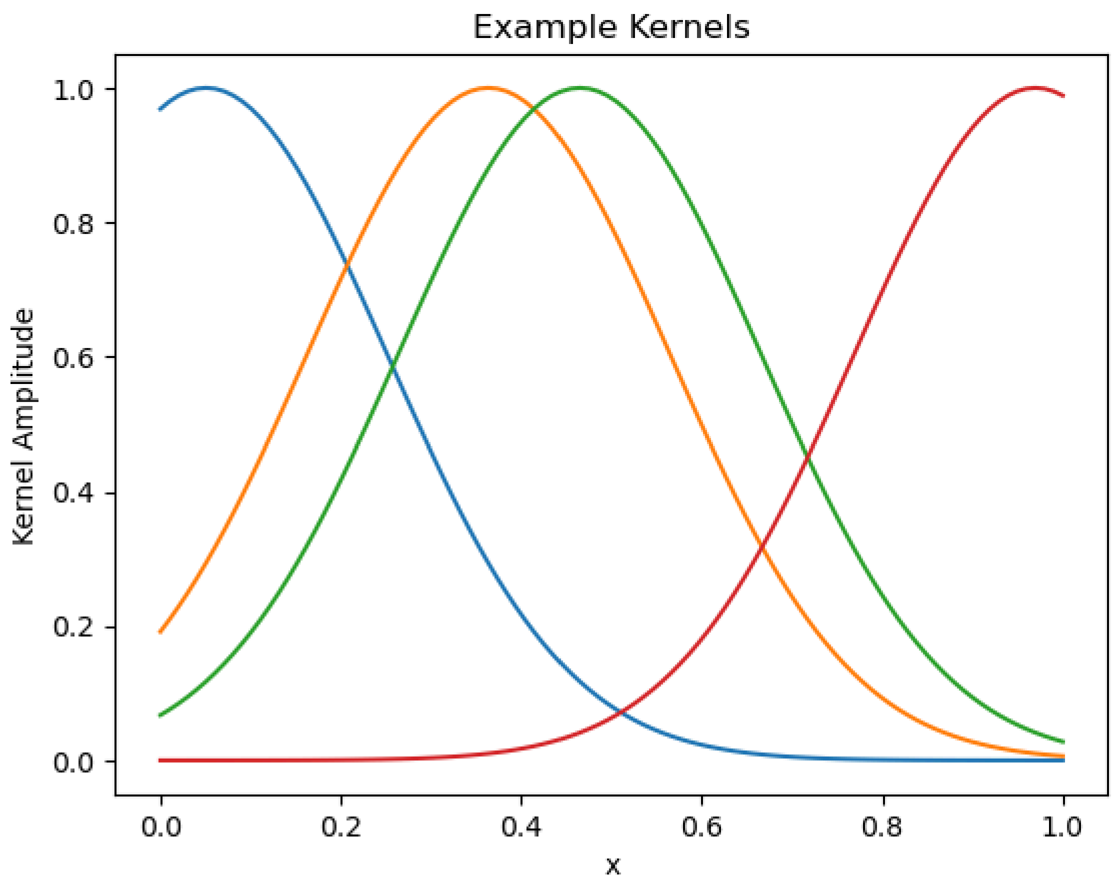
In [15]:
```
# sigma = 0.04 #defines Gaussian kernel width
sigma = [0.04, 0.2, 1, 0.04, 0.2]
p = 100 #number of points on x-axis

# Display examples of the kernels
x_test = np.linspace(0,1.00,p)  # uniformly sample interval [0,1]
j_list = [5, 36, 46, 96]  #list of indices for example kernels

Kdisplay = np.zeros((p,len(j_list)),dtype=float)
```

```
def display_kernels(sigma, p, x_test, j_list, Kdisplay):
    for v in sigma:
        for i in range(p):
            for j in range(len(j_list)):
                Kdisplay[i,j]= np.exp(-(x_test[i]-x_test[j_list[j]])**2/(2*v**2))

        print('Sigma = ',v)
        plt.plot(x_test, Kdisplay)
        plt.title('Example Kernels')
        plt.xlabel('x')
        plt.ylabel('Kernel Amplitude')
        plt.show()

    return Kdisplay

Kdisplay = display_kernels(sigma, p, x_test, j_list, Kdisplay)
```

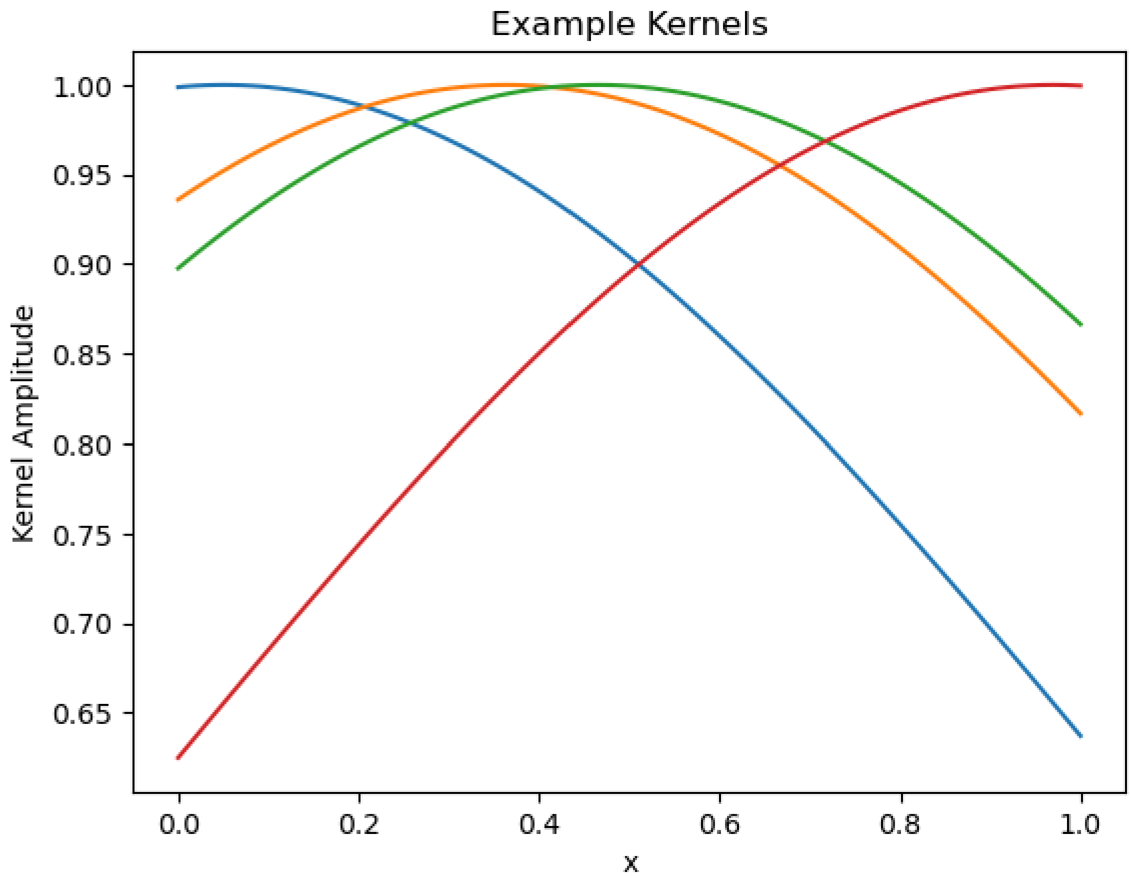
```
Sigma =  0.04
```



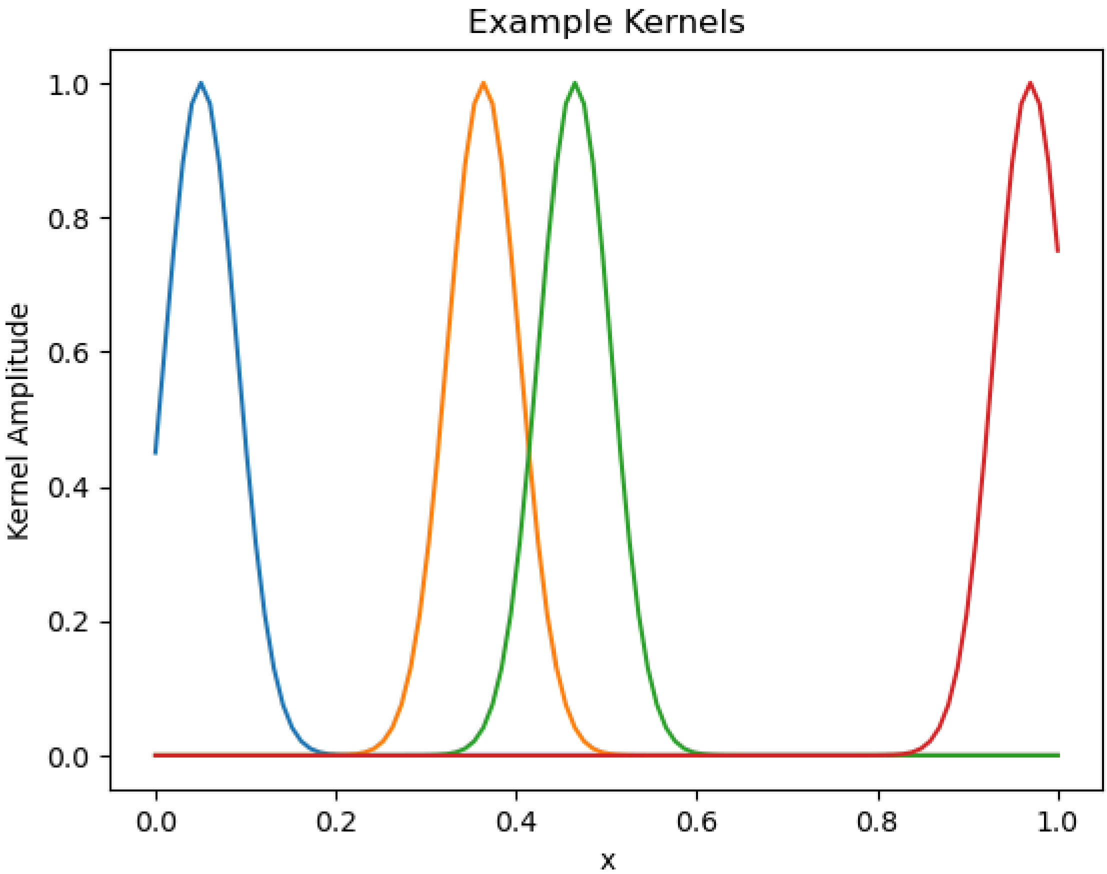

```
Sigma =  0.2
```



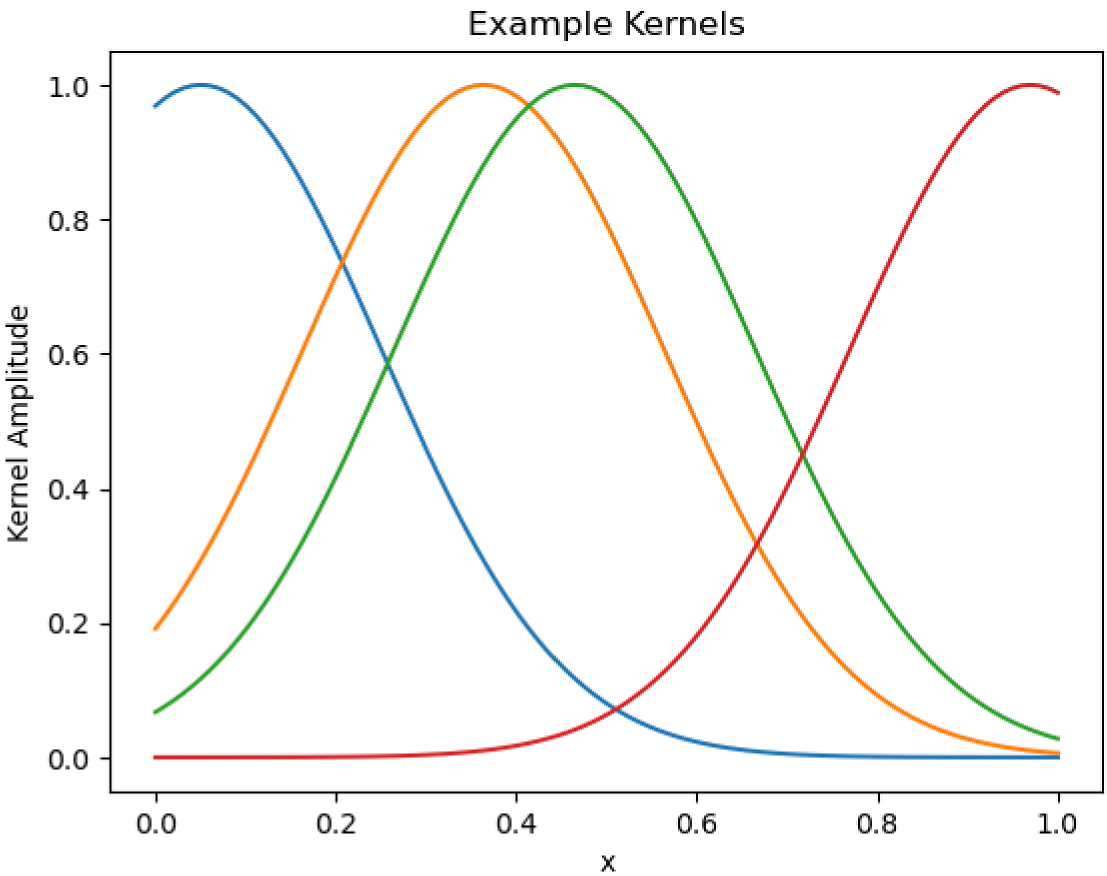
Sigma = 1



Sigma = 0.04




```
Sigma =  0.2
```



1a) Run the regression script with $\sigma = 0.04$ and $\lambda = 0.01$. Figure 1 displays several of the kernels K(x, xi). What is the value xi associated with the kernel having the third peak from the left? What property of the kernel is determined by xi? What property is determined by σ?

xi decides the kernel position (the center of the curve), while sigma decide the width of the curve.

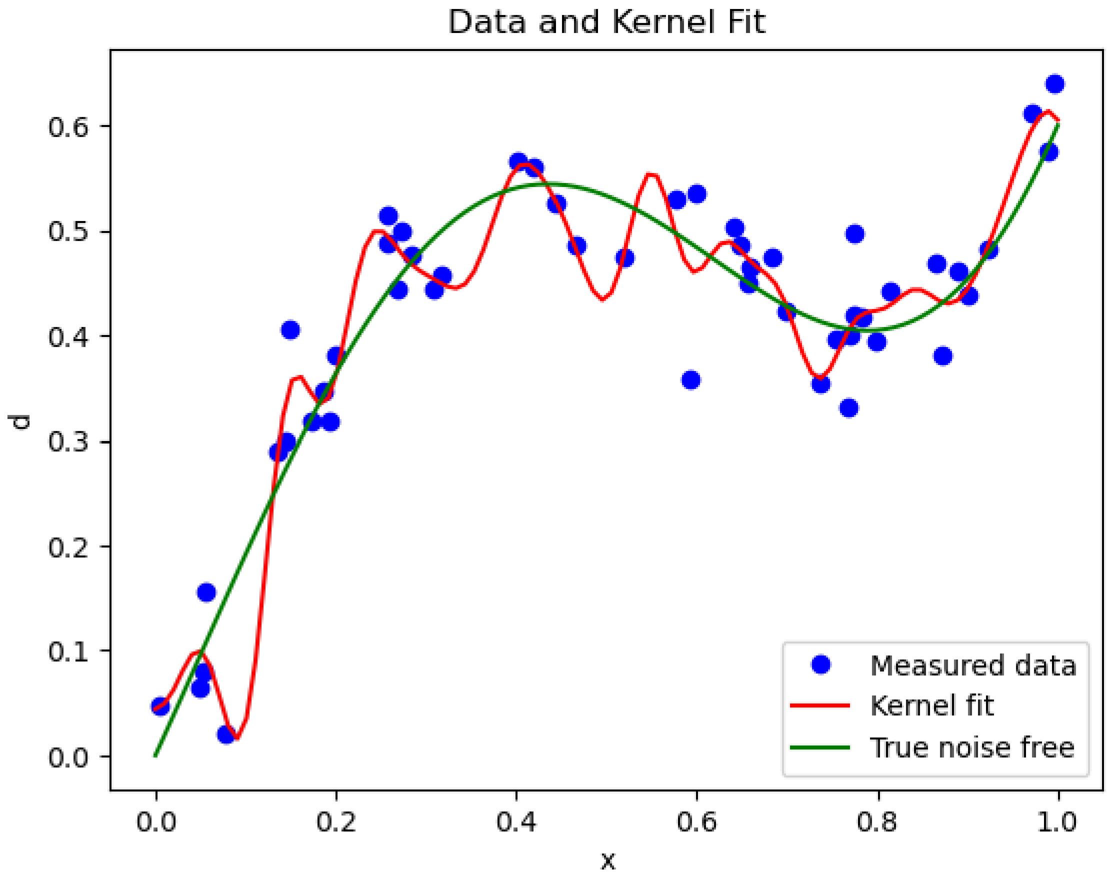
In [16]:
```
# Kernel fitting to data

# lam = 0.01 #ridge regression parameter
lam = [0.01, 0.01, 0.01, 1, 1]
distsq=np.zeros((n,n),dtype=float)

for i in range(0,n):
    for j in range(0,n):
        distsq[i,j]=(x[i]-x[j])**2

K = [0, 0, 0, 0, 0]
for i, v in enumerate(sigma):
    K[i] = np.exp(-distsq/(2*v**2))

alpha = [0, 0, 0, 0, 0]
for i, v in enumerate(lam):
    alpha[i] = np.linalg.inv(K[i]+v*np.identity(n))@d
```

```
C:\Users\ftstc\AppData\Local\Temp\ipykernel_11404\2047022157.py:9: DeprecationWar
ning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will er
ror in future. Ensure you extract a single element from your array before perform
ing this operation. (Deprecated NumPy 1.25.)
  distsq[i,j]=(x[i]-x[j])**2
```

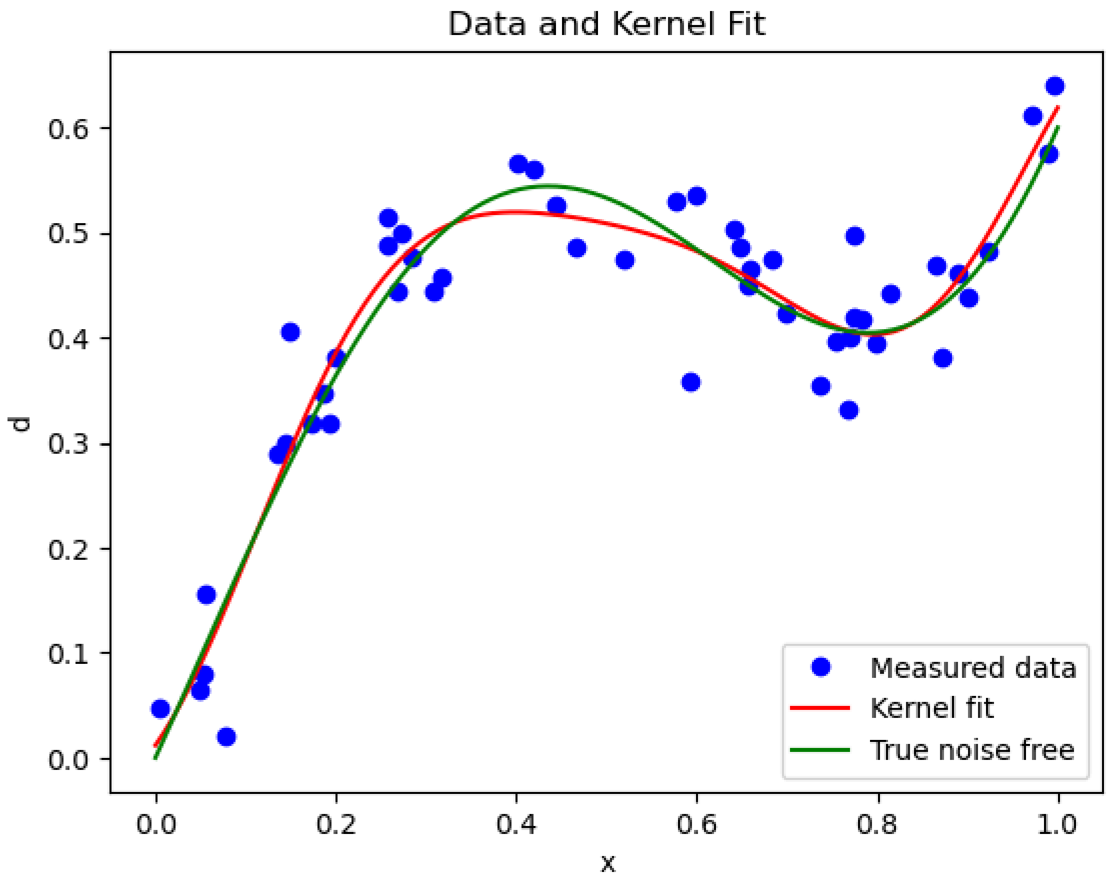
In [17]:
```
# Generate smooth curve corresponding to data fit

distsq_xtest = np.zeros((p,n),dtype=float)
for i in range(0,p):
    for j in range(0,n):
        distsq_xtest[i,j] = (x_test[i]-x[j])**2

dtest = [0, 0, 0, 0, 0]
for i, v in enumerate(sigma):
    dtest[i] = np.exp(-distsq_xtest/(2*v**2))@alpha[i]

    dtrue = x_test*x_test + 0.4*np.sin(1.5*np.pi*x_test)  # noise free data for

    print('Sigma = ',sigma[i])
    print('Lambda = ',lam[i])
    plt.plot(x,d,'bo',label='Measured data')
    plt.plot(x_test,dtest[i],'r',label='Kernel fit')
    plt.plot(x_test,dtrue,'g',label='True noise free')
    plt.title('Data and Kernel Fit')
    plt.legend(loc='lower right')
    plt.xlabel('x')
    plt.ylabel('d')
    plt.show()
```

```
Sigma =  0.04
Lambda =  0.01
```

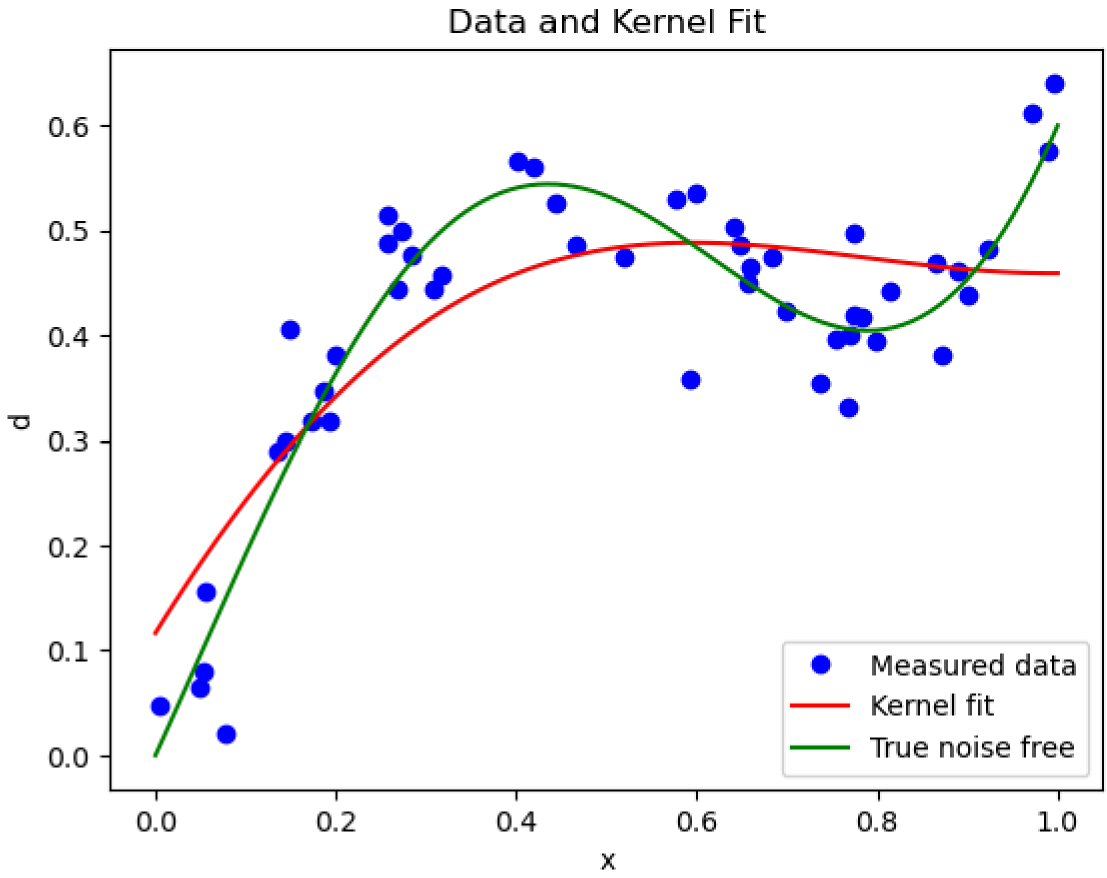
```
C:\Users\ftstc\AppData\Local\Temp\ipykernel_11404\1025918452.py:6: DeprecationWar
ning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will er
ror in future. Ensure you extract a single element from your array before perform
ing this operation. (Deprecated NumPy 1.25.)
  distsq_xtest[i,j] = (x_test[i]-x[j])**2
```



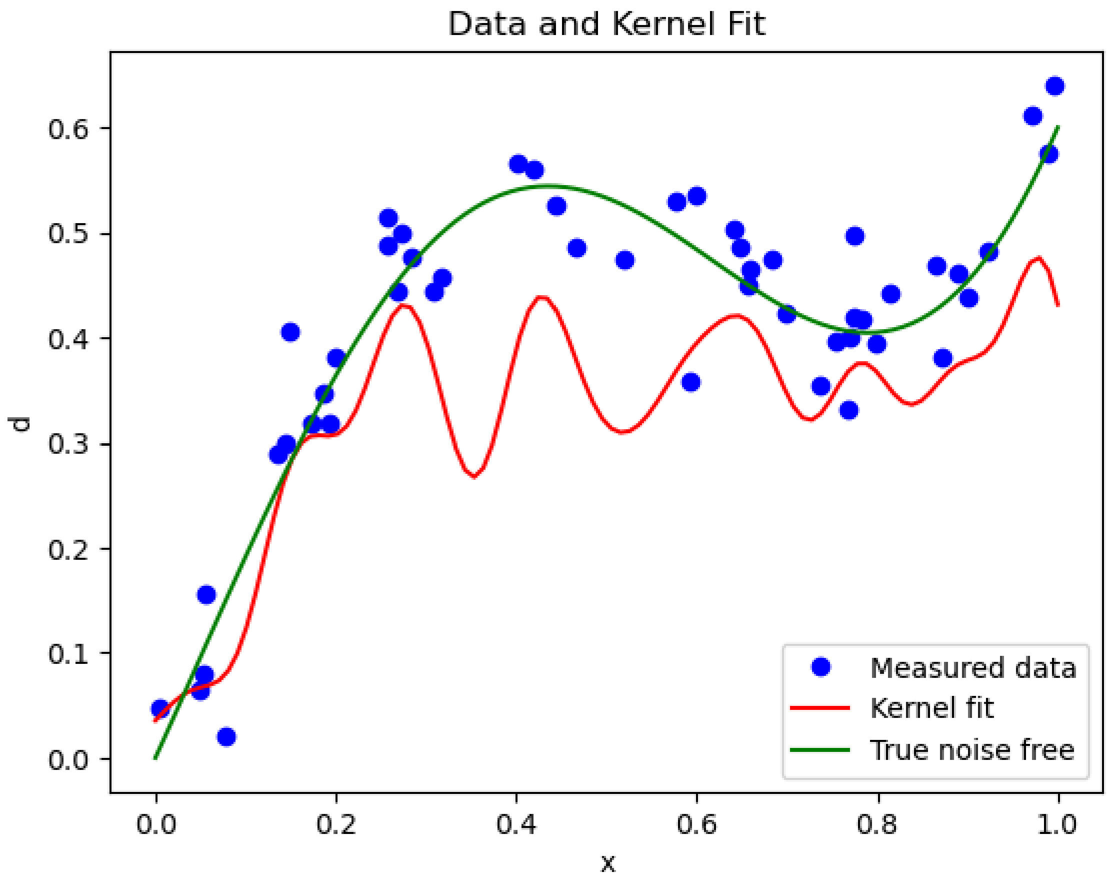
Sigma = 0.2
Lambda = 0.01



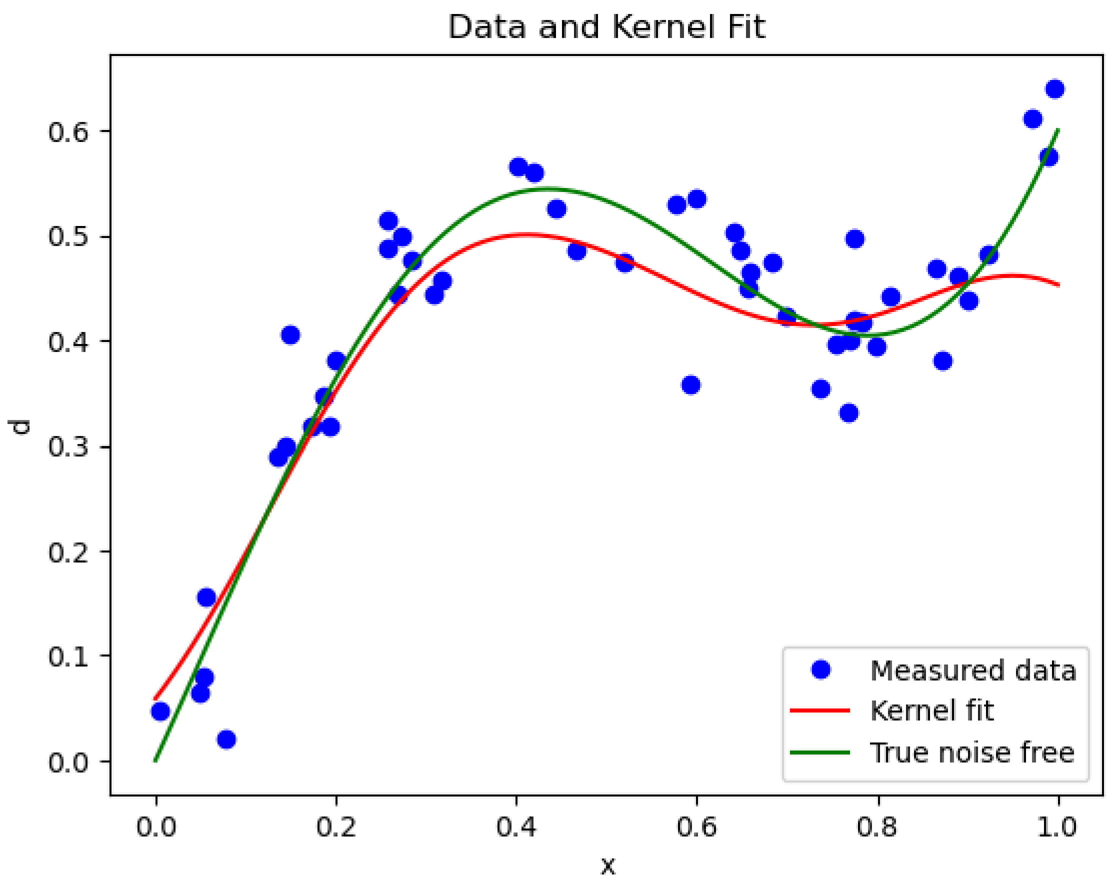
Sigma = 1
Lambda = 0.01




```
Sigma =  0.04
Lambda =  1
```




```
Sigma =  0.2
Lambda =  1
```



1b) Discuss how λ and σ affect the characteristics of the kernel regression to the measured data, and support your conclusions with rationale and plots.

When lambda turns bigger, the kernel fit line are pulled down, showing d declines. When sigma turns bigger, the kernel fit line become smoother.

1c) What principle could you apply to select appropriate values for λ and σ?

Use the combination of lambda and sigma, utilize cross validation, and then calculate the sqaured error.

In [ ]:

```

```

In [ ]:

```

```

3.

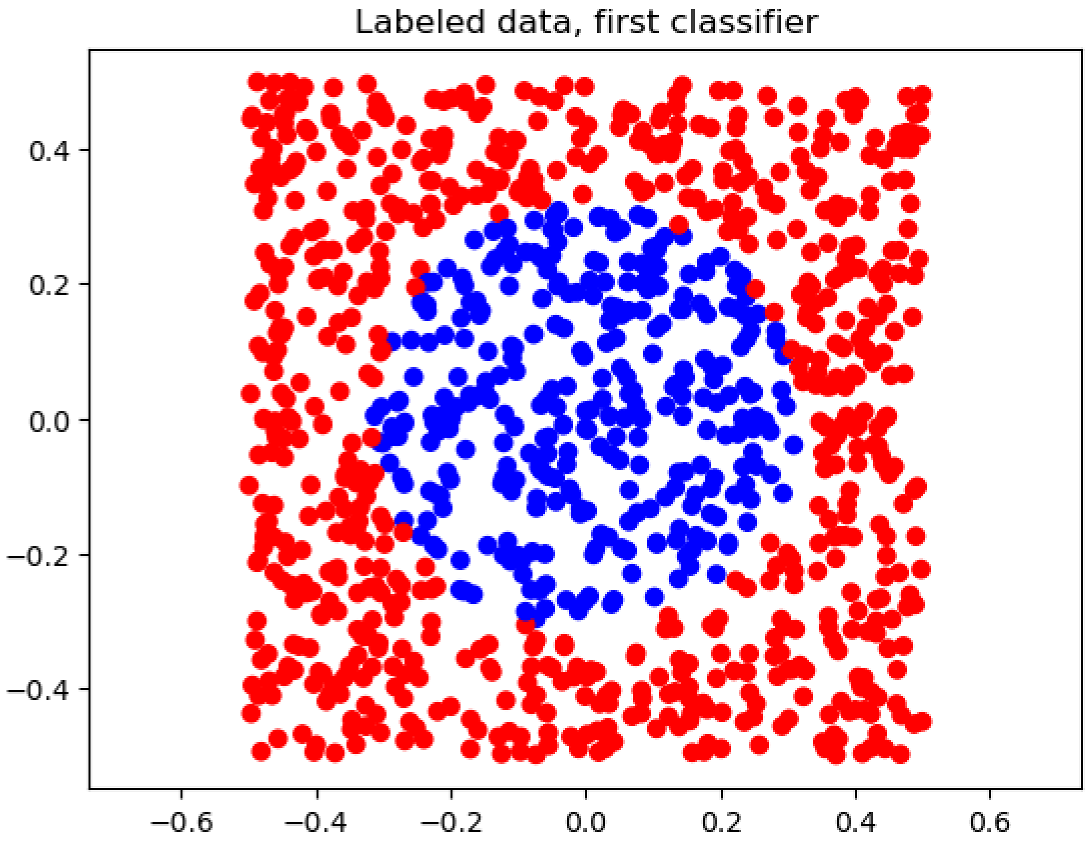
a) $\text{sign}\left(\sum_{j=1}^{N} \alpha_i K(x, x_j) \right)$

b) $\text{sign}\left(\alpha_{100} K(x, x_{100}) + \alpha_{101} K(x, x_{101}) \right)$

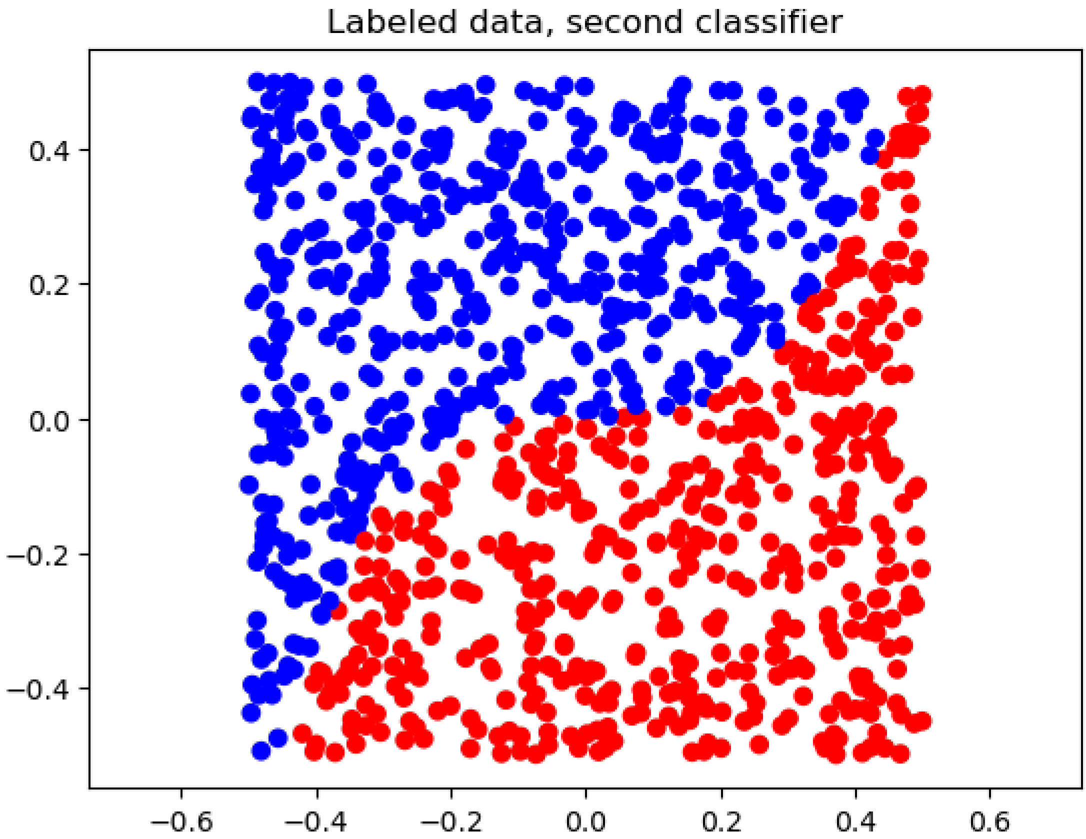
Kernel Classification Example

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

p = int(2) #features
n = int(1000) #examples

## generate training data
X = np.random.rand(n,p)-0.5
Y1 = np.sign(np.sum(X**2,1)-.1).reshape((-1, 1))

Y2 = np.sign(5*X[:,[0]]**3-X[:,[1]])
Y = np.hstack((Y1, Y2))
```

In [2]:
```
# Plot training data for first classification problem
plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Y1[:,0]])
plt.axis('equal')
plt.title('Labeled data, first classifier')
plt.show()
```



In [3]:
```
# Plot training data for second classification problem
plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Y2[:,0]])
plt.title('Labeled data, second classifier')
plt.axis('equal')
plt.show()
```

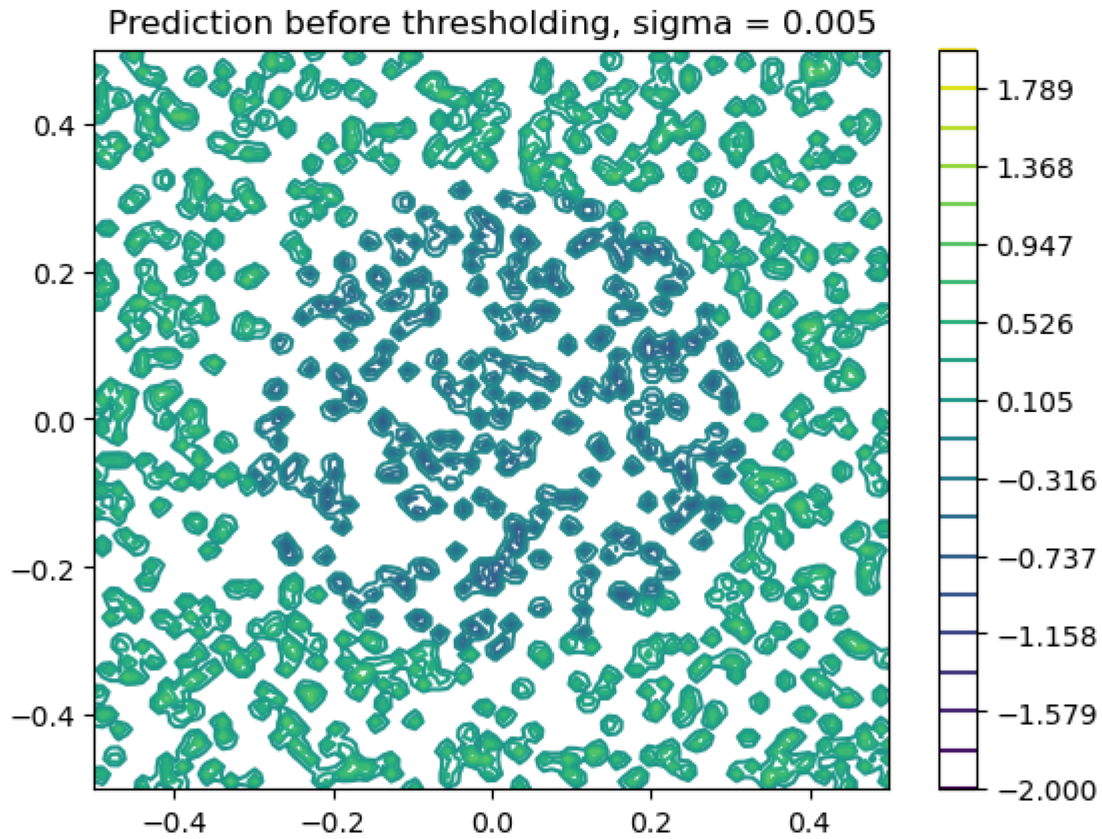


In [10]:

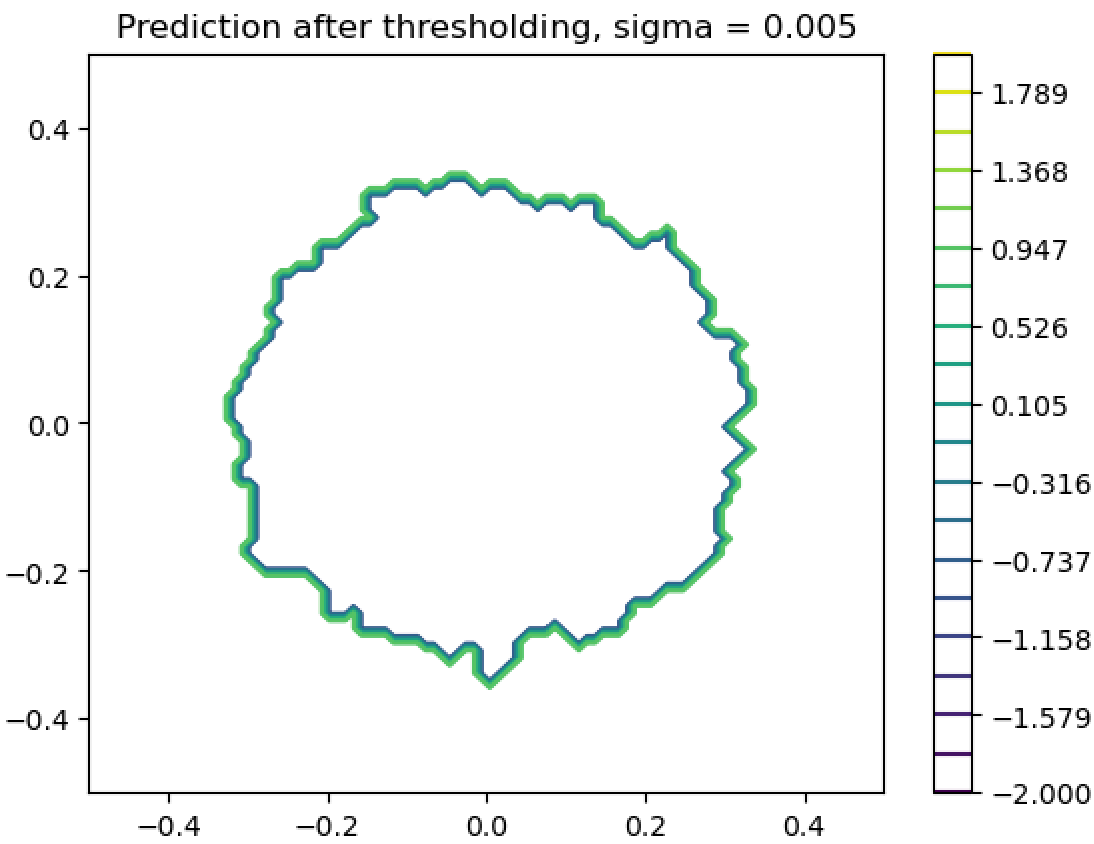
```
# Train Classifier 1
# sigma = 5
# sigma = 0.05
sigma = 0.005

lam = 0.01

distsq=np.zeros((n,n),dtype=float)

for i in range(0,n):
    for j in range(0,n):
        d = np.linalg.norm(X[i,:]-X[j,:])
        distsq[i,j]=d**2

K = np.exp(-distsq/(2*sigma**2))

alpha = np.linalg.inv(K+lam*np.identity(n))@Y1


# Predict labels on a grid of points

X_grid = []
Y_hat_grid = []

g = 100 #number of grid points
Y_hat_grid = np.zeros((g,g))

x1_grid = np.linspace(-.5,.5,g)
x2_grid = np.linspace(-.5,.5,g)

for i,x1 in enumerate(x1_grid):
    for j,x2 in enumerate(x2_grid):
        Y_hat_grid[i,j] = np.exp(-np.linalg.norm(X - np.array([x1,x2]), axis = 1
```

```python
plt.contour(x1_grid, x2_grid, Y_hat_grid, np.linspace(-2,2,20))
plt.colorbar()
plt.title('Prediction before thresholding, sigma = '+ str(sigma))
plt.show()

plt.contour(x1_grid, x2_grid, np.sign(Y_hat_grid), np.linspace(-2,2,20))
plt.colorbar()
plt.title('Prediction after thresholding, sigma = '+ str(sigma))
```

```
C:\Users\ftstc\AppData\Local\Temp\ipykernel_4392\74362037.py:33: DeprecationWarni
ng: Conversion of an array with ndim > 0 to a scalar is deprecated, and will erro
r in future. Ensure you extract a single element from your array before performin
g this operation. (Deprecated NumPy 1.25.)
  Y_hat_grid[i,j] = np.exp(-np.linalg.norm(X - np.array([x1,x2]), axis = 1)**2/(2
*sigma**2))@alpha
```




```
Out[10]:  Text(0.5, 1.0, 'Prediction after thresholding, sigma = 0.005')
```

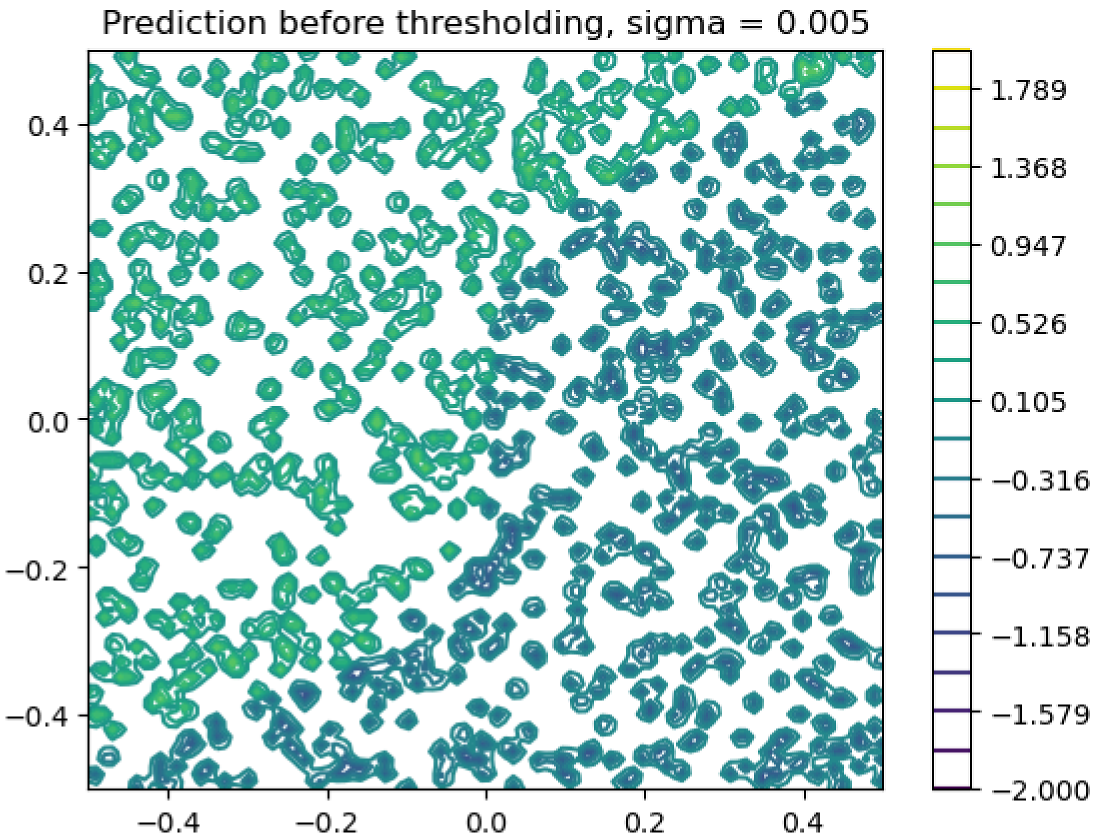


In [11]:

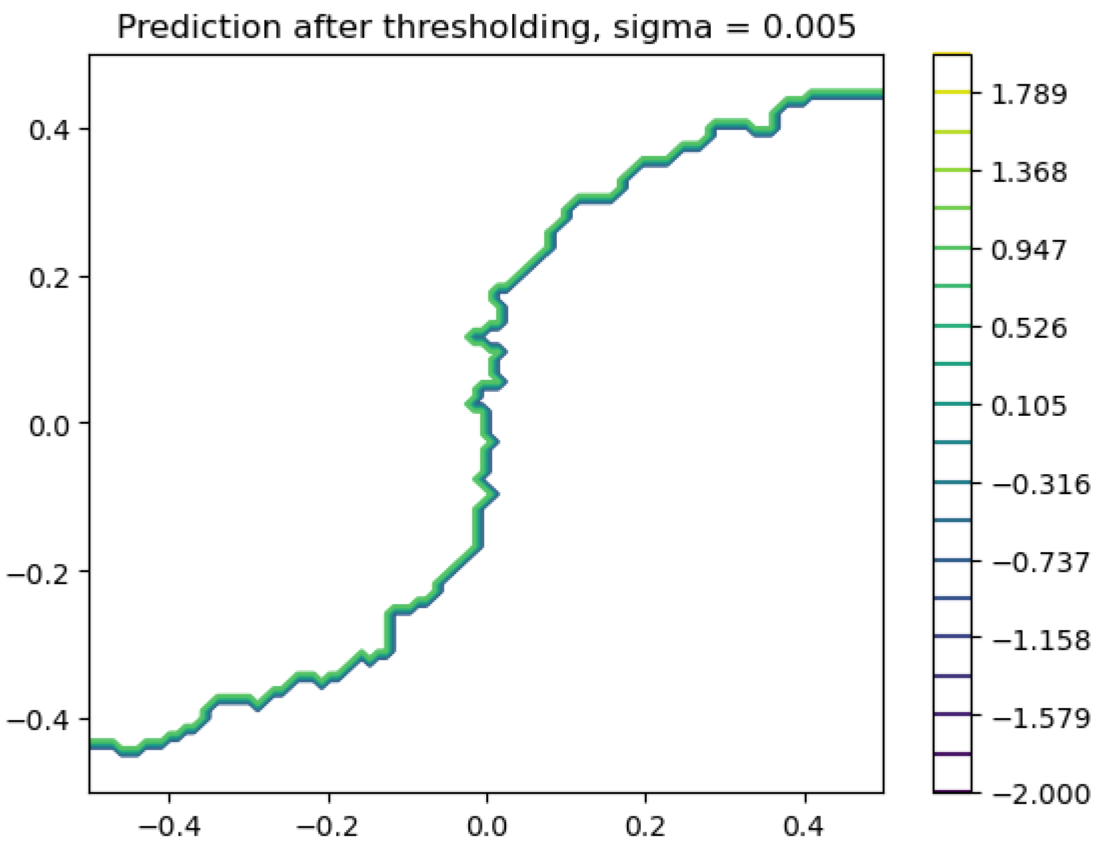
```
# Train Classifier 2
# sigma = 5
# sigma = 0.05
sigma = 0.005

lam = 0.01

distsq=np.zeros((n,n),dtype=float)

for i in range(0,n):
    for j in range(0,n):
        d = np.linalg.norm(X[i,:]-X[j,:])
        distsq[i,j]=d**2

K = np.exp(-distsq/(2*sigma**2))

alpha = np.linalg.inv(K+lam*np.identity(n))@Y2


# Predict labels on a grid of points
X_grid = []
Y_hat_grid = []

g = 100 #number of grid points
Y_hat_grid = np.zeros((g,g))

x1_grid = np.linspace(-.5,.5,g)
x2_grid = np.linspace(-.5,.5,g)

for i,x1 in enumerate(x1_grid):
    for j,x2 in enumerate(x2_grid):
        Y_hat_grid[i,j] = np.exp(-np.linalg.norm(X - np.array([x1,x2]), axis = 1
```

```
plt.contour(x1_grid, x2_grid, Y_hat_grid, np.linspace(-2,2,20))
plt.colorbar()
plt.title('Prediction before thresholding, sigma = '+ str(sigma))
plt.show()

plt.contour(x1_grid, x2_grid, np.sign(Y_hat_grid), np.linspace(-2,2,20))
plt.colorbar()
plt.title('Prediction after thresholding, sigma = '+ str(sigma))
```

```
C:\Users\ftstc\AppData\Local\Temp\ipykernel_4392\1626496689.py:32: DeprecationWar
ning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will er
ror in future. Ensure you extract a single element from your array before perform
ing this operation. (Deprecated NumPy 1.25.)
  Y_hat_grid[i,j] = np.exp(-np.linalg.norm(X - np.array([x1,x2]), axis = 1)**2/(2
*sigma**2))@alpha
```




```
Out[11]:  Text(0.5, 1.0, 'Prediction after thresholding, sigma = 0.005')
```



When sigma turns smaller, the model become overfitting.