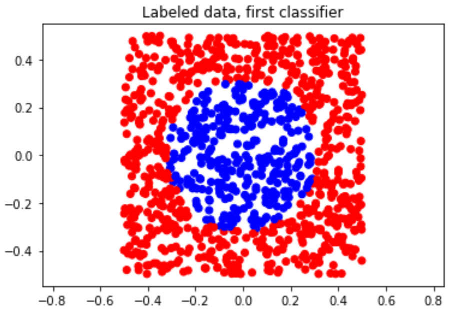
2a)

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))
```

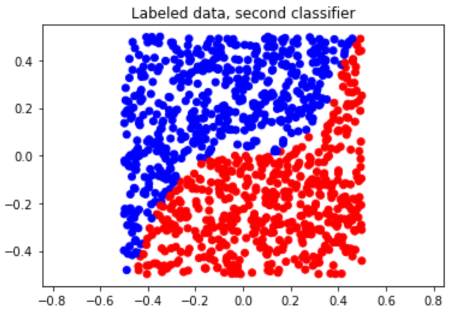
In [2]:

```
# 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()
```



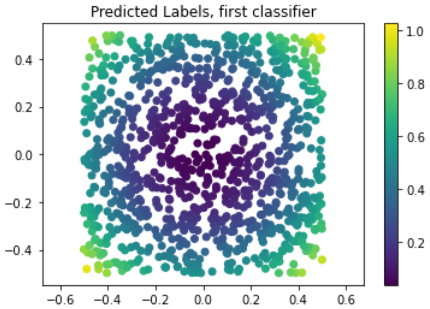
In [4]:

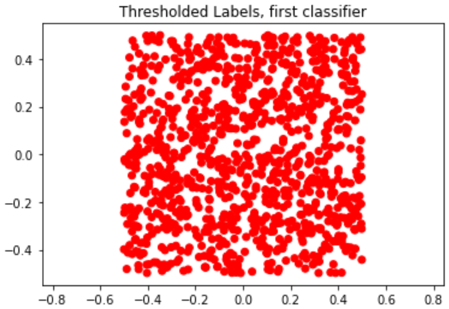
```
# Train Classifiers

sigma = 5
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))

alpha1 = np.linalg.inv(K+lam*np.identity(n))@Y1
alpha2 = np.linalg.inv(K+lam*np.identity(n))@Y2
```

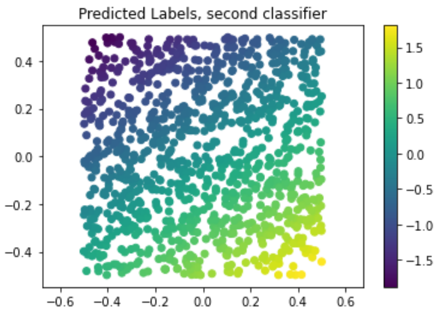
In [5]:

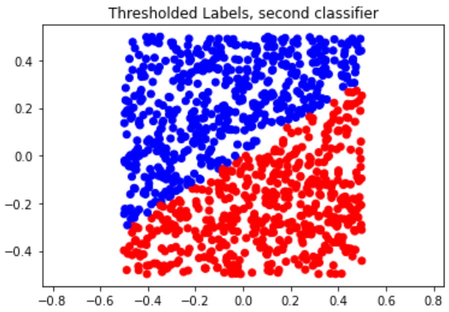
```
# Predict labels

Yhat = K@np.hstack((alpha1, alpha2))
Yhat_thresh=np.sign(Yhat)
```

In [6]:
```
# Display results

plt.scatter(X[:,0], X[:,1], c=Yhat[:,0])
plt.colorbar()
plt.title('Predicted Labels, first classifier')
plt.axis('equal')
plt.show()

plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Yhat_thresh[:,0]])
plt.axis('equal')
plt.title('Thresholded Labels, first classifier')
plt.show()

plt.scatter(X[:,0], X[:,1], c=Yhat[:,1])
plt.title('Predicted Labels, second classifier')
plt.colorbar()
plt.axis('equal')
plt.show()

plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Yhat_thresh[:,1]])
plt.axis('equal')
plt.title('Thresholded Labels, second classifier')
plt.show()
```









In [7]:
```
err_c1 = np.sum(np.abs(Yhat_thresh[:,0]-Y[:,0]))
print('Errors, first classifier:', err_c1)

err_c2 = np.sum(np.abs(Yhat_thresh[:,1]-Y[:,1]))
print('Errors, second classifier:', err_c2)
```

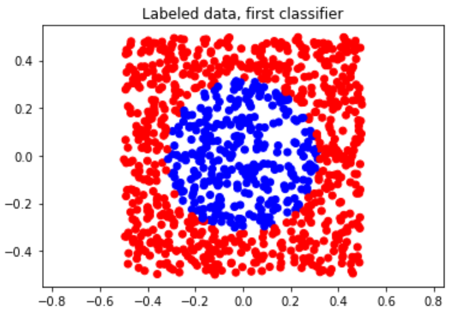
```
Errors, first classifier: 632.0
Errors, second classifier: 142.0
```

In [ ]:

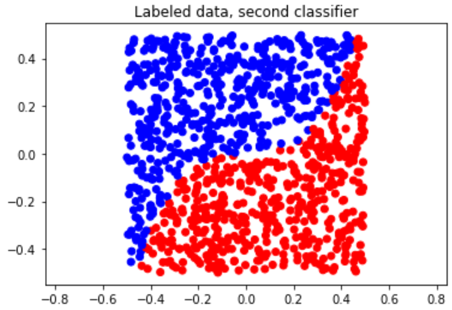
2b)

In [8]:
```
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 [9]:
```
# 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()
```



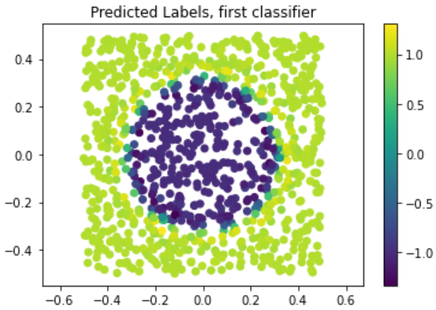
In [10]:
```
# Plot training data for second classification problem
plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Y2[:,0]])
plt.title('Labeled data, second classifier')
plt.axis('equal')
plt.show()
```

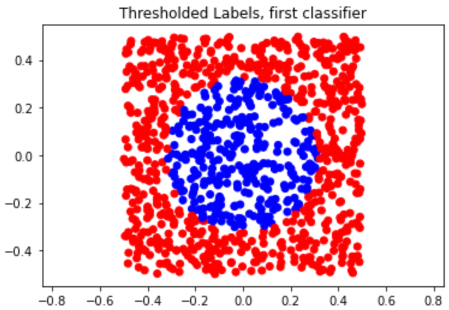


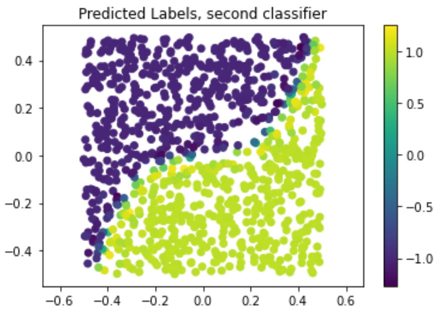
In [11]:
```
# Train Classifiers

sigma = 0.05
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))

alpha1 = np.linalg.inv(K+lam*np.identity(n))@Y1
alpha2 = np.linalg.inv(K+lam*np.identity(n))@Y2
```

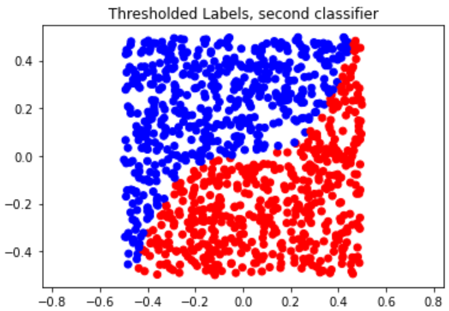
In [12]:
```
# Predict labels

Yhat = K@np.hstack((alpha1, alpha2))
Yhat_thresh=np.sign(Yhat)
```

In [13]:
```python
# Display results

plt.scatter(X[:,0], X[:,1], c=Yhat[:,0])
plt.colorbar()
plt.title('Predicted Labels, first classifier')
plt.axis('equal')
plt.show()

plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Yhat_thresh[:,0]])
plt.axis('equal')
plt.title('Thresholded Labels, first classifier')
plt.show()

plt.scatter(X[:,0], X[:,1], c=Yhat[:,1])
plt.title('Predicted Labels, second classifier')
plt.colorbar()
plt.axis('equal')
plt.show()

plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Yhat_thresh[:,1]])
plt.axis('equal')
plt.title('Thresholded Labels, second classifier')
plt.show()
```









In [14]:
```python
err_c1 = np.sum(np.abs(Yhat_thresh[:,0]-Y[:,0]))
print('Errors, first classifier:', err_c1)

err_c2 = np.sum(np.abs(Yhat_thresh[:,1]-Y[:,1]))
print('Errors, second classifier:', err_c2)
```

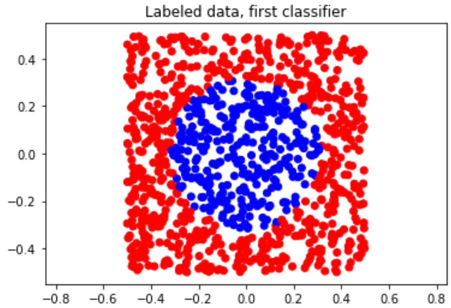
```
Errors, first classifier: 0.0
Errors, second classifier: 0.0
```

In [ ]:

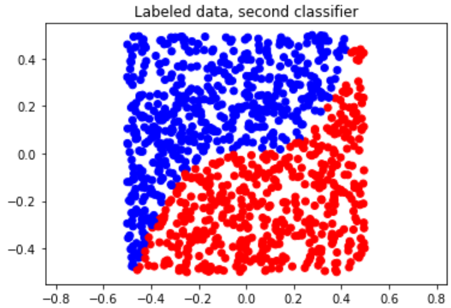
2c)

In [15]:
```
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 [16]:
```
# 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()
```



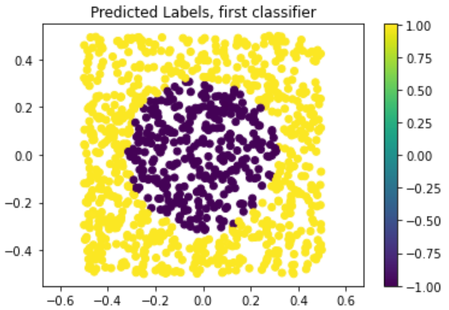
In [17]:
```
# Plot training data for second classification problem
plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Y2[:,0]])
plt.title('Labeled data, second classifier')
plt.axis('equal')
plt.show()
```

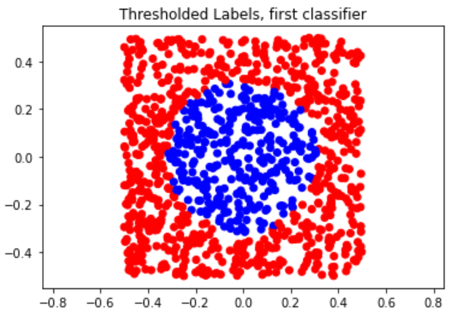


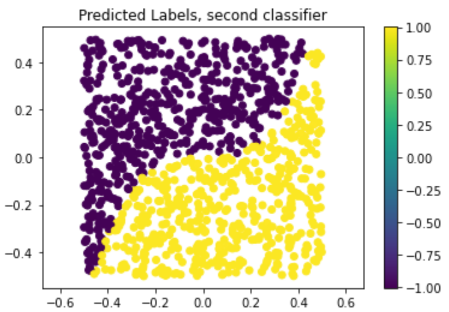
In [18]:
```
# Train Classifiers

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))

alpha1 = np.linalg.inv(K+lam*np.identity(n))@Y1
alpha2 = np.linalg.inv(K+lam*np.identity(n))@Y2
```

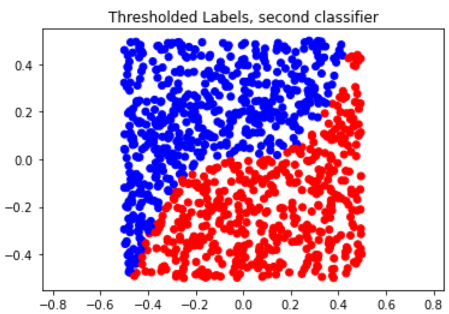
In [19]:
```
# Predict labels

Yhat = K@np.hstack((alpha1, alpha2))
Yhat_thresh=np.sign(Yhat)
```

In [20]:
```python
# Display results

plt.scatter(X[:,0], X[:,1], c=Yhat[:,0])
plt.colorbar()
plt.title('Predicted Labels, first classifier')
plt.axis('equal')
plt.show()

plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Yhat_thresh[:,0]])
plt.axis('equal')
plt.title('Thresholded Labels, first classifier')
plt.show()

plt.scatter(X[:,0], X[:,1], c=Yhat[:,1])
plt.title('Predicted Labels, second classifier')
plt.colorbar()
plt.axis('equal')
plt.show()

plt.scatter(X[:,0], X[:,1], color=['b' if i==-1 else 'r' for i in Yhat_thresh[:,1]])
plt.axis('equal')
plt.title('Thresholded Labels, second classifier')
plt.show()
```









In [21]:
```python
err_c1 = np.sum(np.abs(Yhat_thresh[:,0]-Y[:,0]))
print('Errors, first classifier:', err_c1)

err_c2 = np.sum(np.abs(Yhat_thresh[:,1]-Y[:,1]))
print('Errors, second classifier:', err_c2)
```

```
Errors, first classifier: 0.0
Errors, second classifier: 0.0
```

We see that the sigma parameter when decreased gives much better decision boundaries. Boundaries that classify lower and lower amounts of points incorrectly.

There is a downside to this. You can overfit the data if the sigma is too small as your classifier will try to classofy every little detail as part of the function, even noise. This is undesirable.

```
In [ ]:
```