In [1]:

```
mport numpy as np
rom hw1_utils import load_images, load_labels
mport matplotlib.pyplot as plt
matplotlib inline

rain_X = load_images()
rain_y = load_labels()
```

a) see the code below

In [2]:

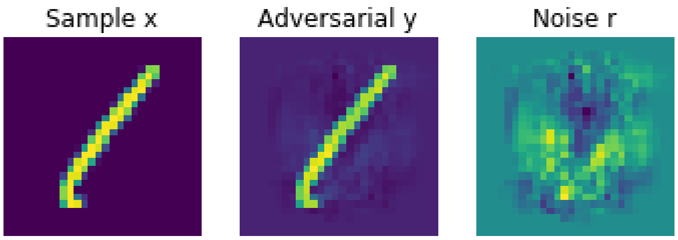
```
 we take out all the Xs that corresponded to a 1 or 6
 = train_X[(train_y==1)|(train_y==6),]
 we set 1 to be class 1 and 6 to be class 0
 = train_y[(train_y==1)|(train_y==6)]==1

 we generate a hyperplane that seperates 1 from 6, recycled from pset 1, it w
rks
x = np.size(X,1)+1
y = np.size(X,0)
0 = np.hstack((X,-np.ones((ny,1))))
0[~y,] = -X0[~y,]
 = np.zeros((nx))
_est = np.squeeze(np.matmul(X0,w))>0
pcnt = 0
hile sum(y_est)<ny:
   lpcnt+=1
   i = 0
   while i<ny:
       y_est[i] = np.matmul(X0[i,],w)>0
       if y_est[i]<=0:
           w+=X0[i,]/np.linalg.norm(X0[i,])
       i+=1
```

b) We pick a random digit and apply the equation from Question 4, except we use the preprocessed x with the -1 added at the end.

In [3]:

```
1 = np.hstack((X,-np.ones((ny,1))))
 = np.squeeze(X1[np.random.randint(ny,size=1),])
 = -np.matmul(x,w)*w/np.linalg.norm(w)**2
 = x+r
lt.subplot(1,3,1)
lt.imshow(np.reshape(x[:-1],[28,28]))
lt.axis('off')
lt.title('Sample x')
lt.subplot(1,3,2)
lt.imshow(np.reshape(y[:-1],[28,28]))
lt.axis('off')
lt.title('Adversarial y')
lt.subplot(1,3,3)
lt.imshow(np.reshape(r[:-1],[28,28]))
lt.axis('off')
lt.title('Noise r')
lt.show()
rint('The norm of r is {:.2f}'.format(np.linalg.norm(r[:-1])))
rint('Compared to the norm of x which is {:.2f}'.format(np.linalg.norm(x[:-1
)))
rint('We check that w*y is indeed {:.2f}'.format(np.matmul(y,w)))
```



```
The norm of r is 208.11
Compared to the norm of x which is 1616.60
We check that w*y is indeed 0.00
```

c) Now we use an optimization package to search for an r that miminized $L1$ norm

In [5]:

```
rom cvxopt import matrix
rom l1 import l1

kipped cause ran out of time
```