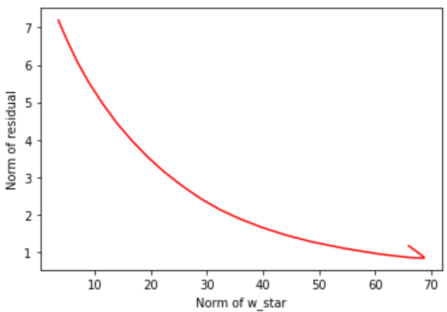
```python
In [1]: def ista_solve_hot( A, d, la_array ):
    # ista_solve_hot: Iterative soft-thresholding for multiple values of
    # lambda with hot start for each case - the converged value for the previous
    # value of lambda is used as an initial condition for the current lambda.
    # this function solves the minimization problem
    # Minimize |Ax-d|_2^2 + lambda*|x|_1 (Lasso regression)
    # using iterative soft-thresholding.
    max_iter = 10**4
    tol = 10**(-3)
    tau = 1/np.linalg.norm(A,2)**2
    n = A.shape[1]
    w = np.zeros((n,1))
    num_lam = len(la_array)
    X = np.zeros((n, num_lam))
    for i, each_lambda in enumerate(la_array):
        for j in range(max_iter):
            z = w - tau*(A.T@(A@w-d))
            w_old = w
            w = np.sign(z) * np.clip(np.abs(z)-tau*each_lambda/2, 0, np.inf)
            X[:, i:i+1] = w
            if np.linalg.norm(w - w_old) < tol:
                break
    return X
```

1a)

```python
In [2]: import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
import pickle

X = loadmat("BreastCancer.mat")['X']
y = loadmat("BreastCancer.mat")['y']

X_100 = X[:100,:]
y_100 = y[:100,:]

lam = np.logspace(-8, np.log10(20), 100)

w_star = ista_solve_hot(X, y, lam)

print((X_100@w_star-y_100).shape)
coord1vals = []
coord2vals = []
for c in range(100):
    temp1 = np.linalg.norm(w_star[:,[c]], ord=1)
    coord1vals.append(temp1)
    temp2 = ((np.sum(((X_100@w_star[:,[c]]-y_100))**2))**0.5)
    #temp2 = (((np.sum((X_100@w_star-y_100)[:,[c]]))**2))**0.5
    coord2vals.append(temp2)

plt.plot(coord1vals, coord2vals, 'r')
plt.xlabel("Norm of w_star")
plt.ylabel("Norm of residual")
```

```
(100, 100)
```

```
Out[2]: Text(0, 0.5, 'Norm of residual')
```

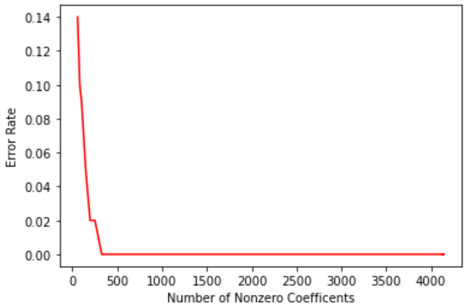


The lower l1 norm represents a lower lambda value. Small values of lambda increase the squared error. Thus we obtain such a graph where for high norm of w* we have a lower norm of residual.

1b)

In [3]:
```python
def sparcity(w_arr):
    nonzeroCount = 0
    for i in w_arr:
        if i>10**-6:
            nonzeroCount += 1
    return nonzeroCount

nonzero_values = []
err = []

for i in range(100):
    nonzero_values.append(sparcity(w_star[:,i]))
    y_hat2 = np.sign(X_100@w_star[:,i:i+1])
    if np.all((y_hat2==0)):
        err.append(1)
        continue
    #err.append(error_rate(np.sign(X_100@w[:,i:i+1], y_train)))
    err.append(np.sum(np.abs(y_hat2-y_100))/2/100)

plt.plot(nonzero_values, err, 'r')
plt.xlabel("Number of Nonzero Coefficents")
plt.ylabel("Error Rate")
```

Out[3]: `Text(0, 0.5, 'Error Rate')`



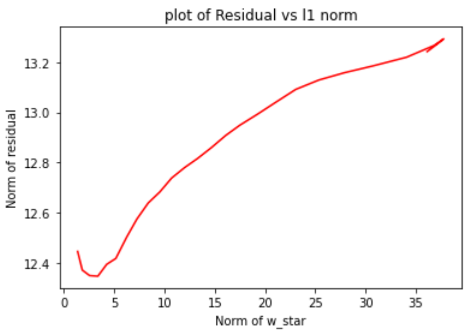
As the number of nonzero coefficients goes up, we get decreasing error rates. Intuitively, this makes perfect sense, as more and more data points are used in the calculation, and few a wasted as zero coefficients. Thus, the error rate tends to 0 at really high nonzero coefficients or low sparsity.

1c)

We repeat the same process with data from rows 101 to beyond.

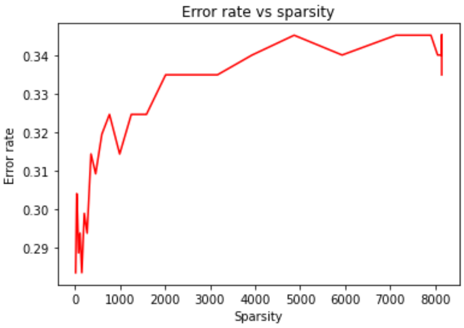
In [4]:
```python
X_c = X[101:]
y_c = y[101:]
lambda_arr = np.logspace(-8,np.log10(20),100)
w_star = ista_solve_hot(X_100,y_100,lambda_arr)
coordx = []
coordy = []
for c in range(len(w_star[0])):
    w_temp = w_star[:, c:c+1]
    coordx.append(np.linalg.norm(w_temp,ord=1))
    coordy.append(np.linalg.norm(X_c@w_temp-y_c))
plt.plot(coordx,coordy, 'r')

plt.xlabel("Norm of w_star")
plt.ylabel("Norm of residual")

plt.title("plot of Residual vs l1 norm")
```

Out[4]: `Text(0.5, 1.0, 'plot of Residual vs l1 norm')`



In [5]:
```
coordx = []
coordy = []
for i in range(len(w_star[0])):
    w = w_star[:, i:i+1]
    d_hat = np.sign(X_c@w)
    error_vec = [0 if m[0]==m[1] else 1 for m in np.hstack((d_hat, y_c))]
    error_rate = sum(error_vec)/len(error_vec)
    sparsity = 0
    for j in w:
        if j!=0:
            sparsity+=1
    coordx.append(sparsity)
    coordy.append(error_rate)
plt.plot(coordx, coordy, 'r')

plt.xlabel("Sparsity")
plt.ylabel("Error rate")
plt.title("Error rate vs sparsity")
```

Out[5]: Text(0.5, 1.0, 'Error rate vs sparsity')



We see a completely different plot for the first plot, whereas in the first case there was an inverse relation between ||w*||1 and ||Aw∗ −d||2 in the second case the relation is directly proportional. The error rate wrt sparsity increases a lot initially in this case whereas it does increase as much initially, and instead falls in (b). Thus we conclude that the training set is not a good fit for the data. More features incorporated makes the error rate go up.

In [ ]:

In [ ]: