In [1]:
```
%matplotlib inline
%config InlineBackend.figure_format='retina'

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from matplotlib import animation, rc, rcParams
rc('font',**{'family':'serif','serif':['DejaVu Sans']})
rc('text', usetex=True)
rcParams['text.latex.preamble']=[r"\usepackage{amsmath}"]
```

In [2]:
```
def plot_all(X,y,w):
    # custom colors for plotting points
    red = [1,0,0.4]
    blue = [0,0.4,1]

    # scatter plot points
    fig = plt.figure(figsize = (4,4))
    ind = np.argwhere(y==1)
    ind = [s[0] for s in ind]
    plt.scatter(X[1,ind],X[2,ind],color = red,edgecolor = 'k',s = 25)
    ind = np.argwhere(y==-1)
    ind = [s[0] for s in ind]
    plt.scatter(X[1,ind],X[2,ind],color = blue,edgecolor = 'k',s = 25)
    plt.grid('off')

    # plot separator
    s = np.linspace(0,1,100)
    plt.plot(s,(-w[0]-w[1]*s)/w[2],color = 'k',linewidth = 2)

    # clean up plot
    plt.xlim([-0.1,1.1])
    plt.ylim([-0.1,1.1])
    fig.savefig('4_03.png', bbox_inches='tight', dpi=500)
    plt.show()

def load_data(csvname):
    data = np.asarray(pd.read_csv(csvname, header=None))
    X = data[:,0:2]
    y = data[:,2]
    y.shape = (len(y),1)
    # pad data with ones for more compact gradient computation
    o = np.ones((np.shape(X)[0],1))
    X = np.concatenate((o,X),axis = 1)
    X = X.T
    return X, y

def sigmoid(t):
    """ That young sigmoid function. """
    def tiny_exp(u):
        s = np.argwhere(u > 100)
        t = np.argwhere(u < -100)
        u[s] = 0
        u[t] = 0
        u = np.exp(u)
        u[t] = 1
        return u
    return 1/(1 + tiny_exp(-t))

def softmax_grad(X,y):
    """Gradient Descent Function for Softmax Cost"""
    w = np.random.randn(3, 1)
    alpha = 10**-2
    max_its = 2000
    for k in range(max_its):

        # gradient calculation
        t = -y * np.matmul(X.T, w)
        r = sigmoid(t) * y
        grad = np.matmul(-1.0 * X, r)
        w = w - (alpha * grad)

    return w
```

In [3]:
```
# load in data
X, y = load_data('imbalanced_2class.csv')

# run gradient descent
w = softmax_grad(X,y)
# plot points and separator
plot_all(X, y, w)
```