In [236]:
```python
from PIL import Image
```

In [240]:
```python
img = Image.open("IMG_3340.JPG")
img = img.transpose(Image.ROTATE_270)
img = img.resize((600,700),Image.ANTIALIAS)
img
```

Out[240]:

① 1.1) B

1.2) C, D

② 2.1)

1) $p(y=-1|x) + p(y=1|x) = 1$

$$p(y=-1|x) = 1 - p(y=1|x)$$

$$= 1 - \frac{1}{1+e^{-(-w^Tx+b)}}$$

$$= \frac{1+e^{-(-w^Tx+b)}}{1+e^{-(w^Tx+b)}} - \frac{1}{1+e^{-(w^Tx+b)}}$$

$$= \frac{e^{-(-w^Tx+b)}}{1+e^{-(w^Tx+b)}}$$

$$= \frac{1}{1+e^{(w^Tx+b)}}$$

2) $p(y=-1|x) = \frac{1}{1+e^{(w^Tx+b)}} = \frac{1}{1+e^{-y(w^Tx+b)}}$

$$p(y=1|x) = \frac{1}{1+e^{-(w^Tx+b)}} = \frac{1}{1+e^{-y(w^Tx+b)}}$$

$$\therefore p(y|x) = \frac{1}{1+e^{-y(w^Tx+b)}}$$

3) $f(x) = \begin{cases} 1 & \text{if } \frac{1}{1+e^{-(w^Tx+b)}} \geq 0.5 \\ -1 & \text{otherwise} \end{cases}$

$$0.5 \leq \frac{1}{1+e^{-(w^Tx+b)}}$$

$$0.5(1+e^{-(w^Tx+b)}) \leq 1$$

$$1+e^{-(w^Tx+b)} \leq 2$$

$$e^{-(w^Tx+b)} \leq 1$$

Decision Boundary

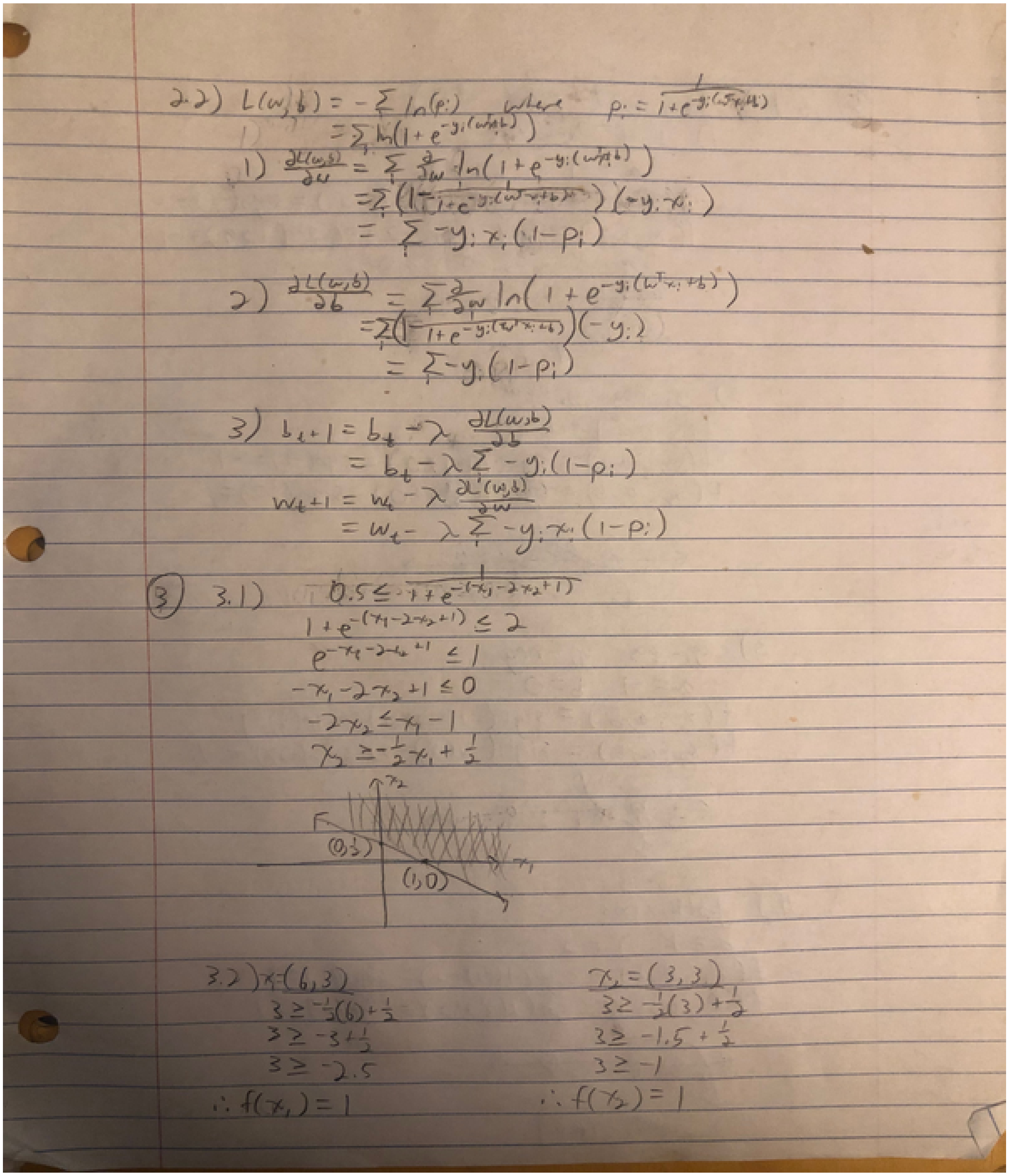
$$-w^Tx - b \leq \ln(1) = 0 \Rightarrow w^Tx + b = 0$$

$$w^Tx \leq b$$

$$x \geq -b(w^T)^{-1} \Rightarrow y = 1, \quad x < -b(w^T)^{-1} \Rightarrow y = -1$$

In [241]:
```
img = Image.open("IMG_3341.JPG")
img = img.transpose(Image.ROTATE_270)
img = img.resize((600,700),Image.ANTIALIAS)
img
```

Out[241]:



2.2) $L(w,b) = -\sum \ln(p_i)$ where $p_i = \frac{1}{1+e^{-y_i(w^T x_i + b)}}$

$= \sum_i \ln(1 + e^{-y_i(w^T x_i + b)})$

1) $\frac{\partial L(w,b)}{\partial w} = \sum_i \frac{\partial}{\partial w} \ln(1 + e^{-y_i(w^T x_i + b)})$

$= \sum_i \left(1 - \frac{1}{1+e^{-y_i(w^T x_i + b)}}\right)(-y_i x_i)$

$= \sum_i -y_i x_i (1-p_i)$

2) $\frac{\partial L(w,b)}{\partial b} = \sum_i \frac{\partial}{\partial w} \ln(1 + e^{-y_i(w^T x_i + b)})$

$= \sum_i \left(1 - \frac{1}{1+e^{-y_i(w^T x_i + b)}}\right)(-y_i)$

$= \sum_i -y_i (1-p_i)$

3) $b_{t+1} = b_t - \lambda \frac{\partial L(w,b)}{\partial b}$

$= b_t - \lambda \sum_i -y_i (1-p_i)$

$w_{t+1} = w_t - \lambda \frac{\partial L(w,b)}{\partial w}$

$= w_t - \lambda \sum_i -y_i x_i (1-p_i)$

3 3.1) $0.5 \leq \frac{1}{1+e^{-(x_1-2x_2+1)}}$

$1 + e^{-(x_1-2x_2+1)} \leq 2$

$e^{-x_1-2x_2+1} \leq 1$

$-x_1 - 2x_2 + 1 \leq 0$

$-2x_2 \leq x_1 - 1$

$x_2 \geq -\frac{1}{2}x_1 + \frac{1}{2}$

3.2) $x_1 = (6,3)$

$3 \geq -\frac{1}{2}(6) + \frac{1}{2}$

$3 \geq -3 + \frac{1}{2}$

$3 \geq -2.5$

$\therefore f(x_1) = 1$

$x_2 = (3,3)$

$3 \geq -\frac{1}{2}(3) + \frac{1}{2}$

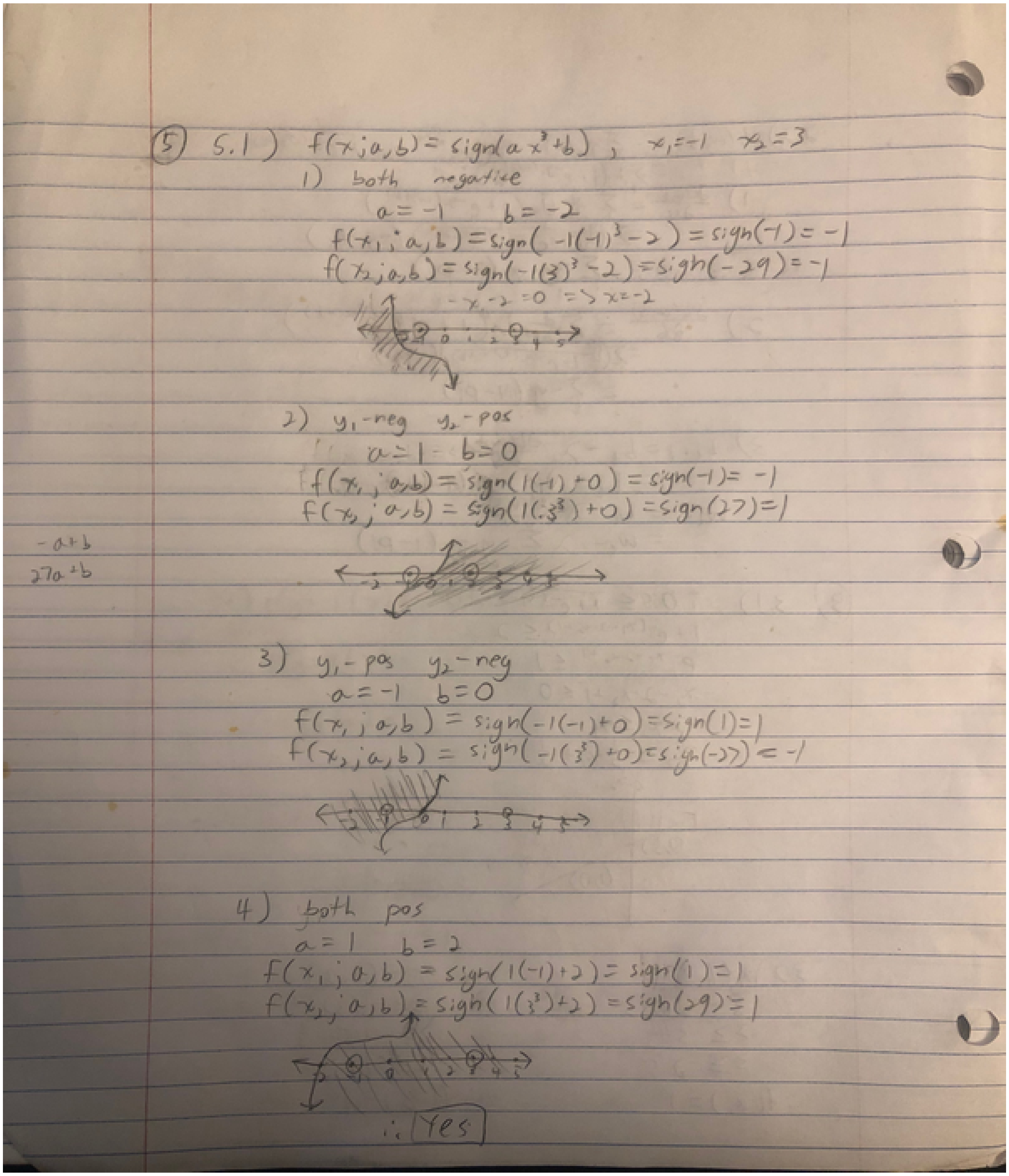
$3 \geq -1.5 + \frac{1}{2}$

$3 \geq -1$

$\therefore f(x_2) = 1$

In [242]:
```
img = Image.open("IMG_3342.JPG")
img = img.transpose(Image.ROTATE_270)
img = img.resize((600,700),Image.ANTIALIAS)
img
```

Out[242]:



(5) 5.1) $f(x;a,b) = \text{sign}(a x^3 + b)$, $x_1 = -1$ $x_2 = 3$

1) both negative

$a = -1$ $b = -2$

$f(x_1; a,b) = \text{sign}(-1(-1)^3 - 2) = \text{sign}(-1) = -1$

$f(x_2; a,b) = \text{sign}(-1(3)^3 - 2) = \text{sign}(-29) = -1$

$-x - 2 = 0 \Rightarrow x = -2$

2) y_1 - neg y_2 - pos

$a = 1$ $b = 0$

$f(x_1; a,b) = \text{sign}(1(-1) + 0) = \text{sign}(-1) = -1$

$f(x_2; a,b) = \text{sign}(1(3^3) + 0) = \text{sign}(27) = 1$

$-a+b$

$27a+b$

3) y_1 - pos y_2 - neg

$a = -1$ $b = 0$

$f(x_1; a,b) = \text{sign}(-1(-1) + 0) = \text{sign}(1) = 1$

$f(x_2; a,b) = \text{sign}(-1(3^3) + 0) = \text{sign}(-27) = -1$

4) both pos

$a = 1$ $b = 2$

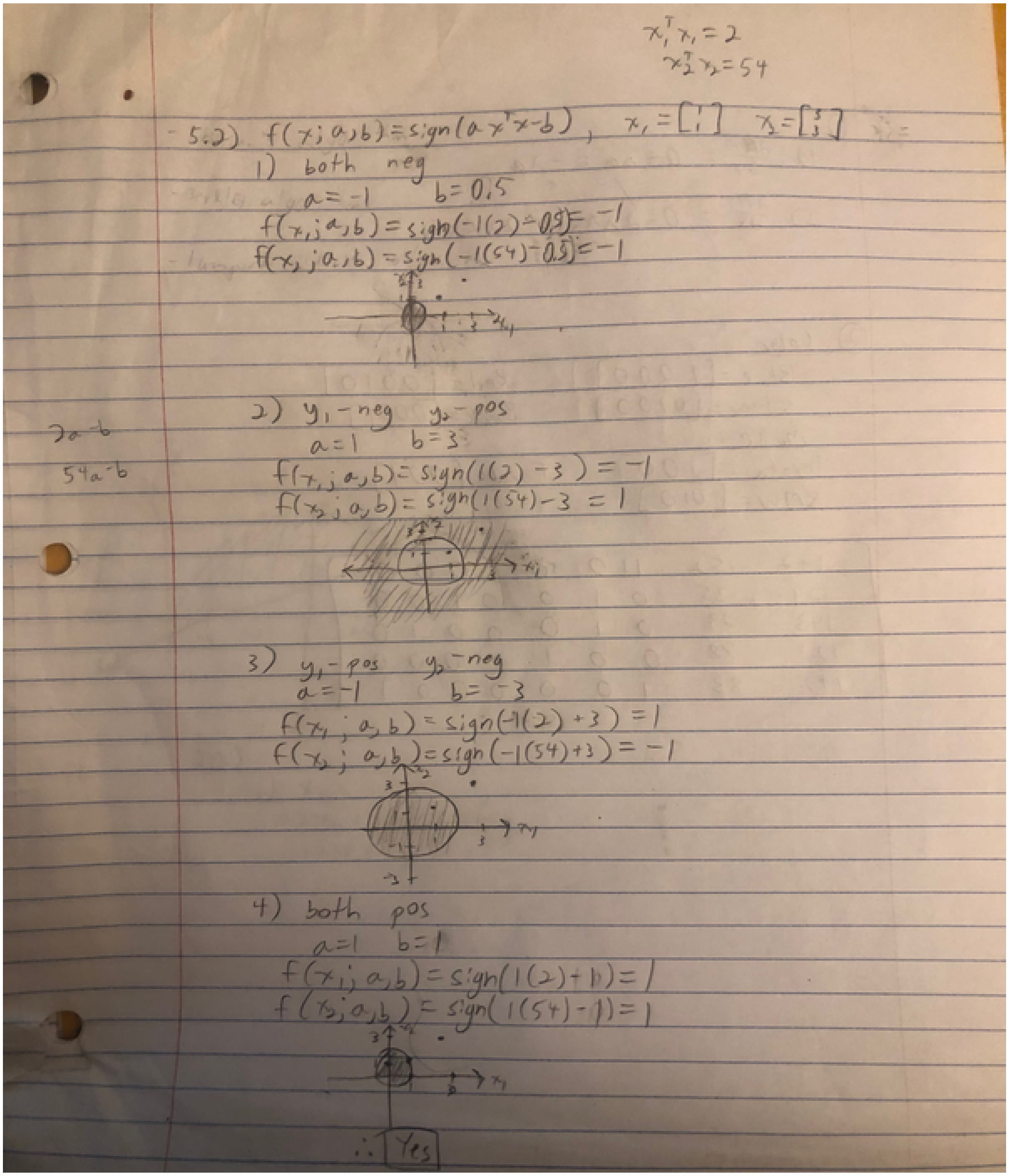
$f(x_1; a,b) = \text{sign}(1(-1) + 2) = \text{sign}(1) = 1$

$f(x_2; a,b) = \text{sign}(1(3^3) + 2) = \text{sign}(29) = 1$

$\therefore \boxed{\text{Yes}}$

In [243]:
```
img = Image.open("IMG_3344.JPG")
img = img.transpose(Image.ROTATE_270)
img = img.resize((600,700),Image.ANTIALIAS)
img
```

Out[243]:



$x_1^T x_1 = 2$

$x_2^T x_2 = 54$

5.2) $f(x; a, b) = \text{sign}(a x^T x - b)$, $x_1 = \begin{bmatrix}1\\1\end{bmatrix}$ $x_2 = \begin{bmatrix}5\\3\end{bmatrix}$

1) both neg

$a = -1$ $b = 0.5$

$f(x_1; a, b) = \text{sign}(-1(2) - 0.5) = -1$

$f(x_2; a, b) = \text{sign}(-1(54) - 0.5) = -1$

2) y_1 - neg y_2 - pos

$2a - b$

$54a - b$

$a = 1$ $b = 3$

$f(x_1; a, b) = \text{sign}(1(2) - 3) = -1$

$f(x_2; a, b) = \text{sign}(1(54) - 3) = 1$

3) y_1 - pos y_2 - neg

$a = -1$ $b = -3$

$f(x_1; a, b) = \text{sign}(-1(2) + 3) = 1$

$f(x_2; a, b) = \text{sign}(-1(54) + 3) = -1$

4) both pos

$a = 1$ $b = 1$

$f(x_1; a, b) = \text{sign}(1(2) + 1) = 1$

$f(x_2; a, b) = \text{sign}(1(54) - 1) = 1$

$\therefore$ $\boxed{\text{Yes}}$

Logistic Regression

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import math
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
from sklearn.utils import shuffle
import scipy.io as sio
plt.rcParams['figure.figsize'] = 8,8
```

Original Data

In [2]:
```
X_and_Y_train = np.load('./logistic-train.npy')
X_train = X_and_Y_train[:, :2]    # Shape: (70,2)
Y_train = X_and_Y_train[:, 2]    # Shape: (70,)
Y_train = 2 * Y_train - 1        # Convert (0, 1) to (-1, 1)
#print(X_and_Y_train.shape, X_train.shape, Y_train.shape)
```

In [3]:
```
X_and_Y_test = np.load('./logistic-test.npy')
X_test = X_and_Y_test[:, :2]   # Shape: (70,2)
Y_test = X_and_Y_test[:, 2]    # Shape: (70,)
Y_test = 2 * Y_test - 1        # Convert (0, 1) to (-1, 1)
#print(X_and_Y_test.shape, X_test.shape, Y_test.shape)
```

In [4]:
```
mpl.style.use('seaborn')

fig = plt.figure()
plt.scatter(X_train[Y_train==-1, 0], X_train[Y_train==-1, 1], marker='x', color=
plt.scatter(X_train[Y_train==1, 0], X_train[Y_train==1, 1], marker='o', color='r
plt.xlabel('$x_0$')
plt.ylabel('$x_1$')
plt.legend(loc='upper right', fontsize=10)
plt.title('Training data')
plt.show()
#fig.savefig('scatter_1.png', format='png', dpi=400)
```

Gradient Descent

In [5]:
```
def sigmoid(z):
    return 1.0/(1.0+np.exp(-z))
```

In [226]:
```
# gradient of loss function L(w, b)
def L_prime_w_b(X, Y, w, b):
    ########### YOUR CODE HERE ###########
    # This function returns the tuple(gradient for w, gradient for b)
    #####################################
    #print(w.shape)
    grad_w = np.zeros(X[1].shape)
    grad_b = 0
    for i in range(len(Y)):
        x = X[i]
        z = Y[i] * (w.T.dot(X[i]) + b)
        h = sigmoid(z)
        grad_w = grad_w - Y[i] * X[i] * (1 - h)
        grad_b = grad_b - Y[i] * (1 - h)
    #print(grad_w.reshape((-1, 1)).shape)
    return (grad_w.reshape((-1, 1)), grad_b)
```

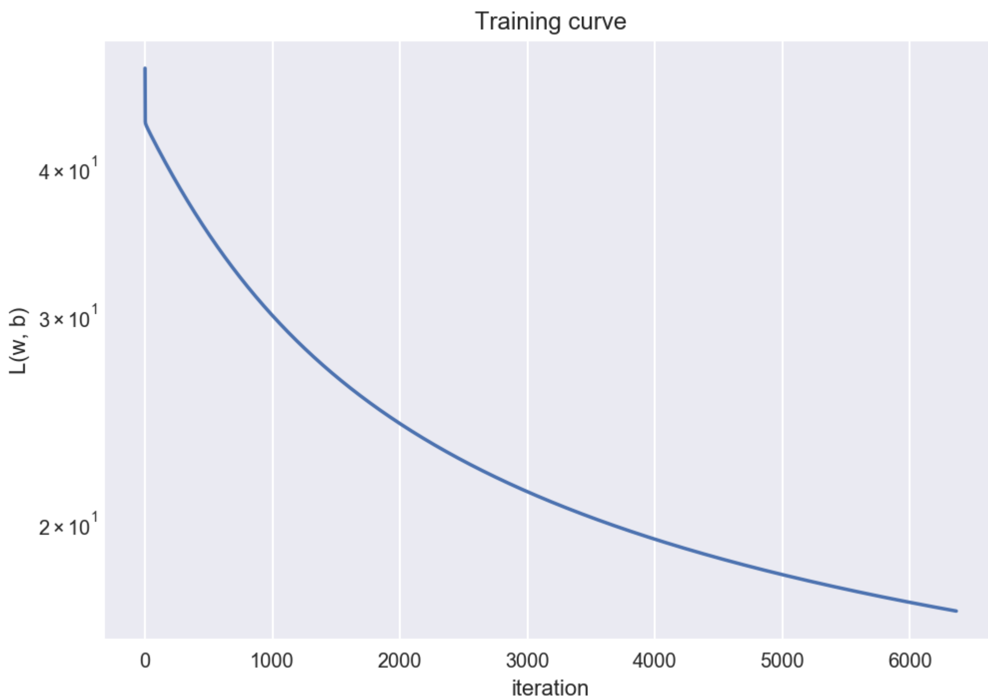
In [227]:
```
def L_w_b(X, Y, w, b):
    ########### YOUR CODE HERE ###########
    L = 0
    for i in range(len(Y)):
        x = X[i]
        z = Y[i] * (w.T.dot(x) + b)
        h = sigmoid(z)
        L = L - np.log(h)
    return L
```

In [228]:
```
learning_rate = 0.001
n_iter = 10000
w = np.zeros((X_train.shape[1], 1))
b = 0

# We will keep track of training loss over iterations
iterations = [0]
L_w_b_list = [L_w_b(X_train, Y_train, w, b)]
for i in range(n_iter):
    gradient_w, gradient_b = L_prime_w_b(X_train, Y_train, w, b)
    w_new = w - learning_rate * gradient_w
    b_new = b - learning_rate * gradient_b
    iterations.append(i+1)
    L_w_b_list.append(L_w_b(X_train, Y_train, w_new, b_new))

    if np.linalg.norm(w_new - w, ord = 1) + abs(b_new - b) < 0.001:
        print("gradient descent has converged after " + str(i) + " iterations")
        break
    w = w_new
    b = b_new

print ("w vector: \n" + str(w))
print ("b: \n" + str(b))
```

```
gradient descent has converged after 6365 iterations
w vector:
[[0.97466299]
 [0.88907048]]
b:
[-11.48322099]
```

Training curve

In [229]:
```
plt.title('Training curve')
plt.xlabel('iteration')
plt.ylabel('L(w, b)')
plt.semilogy(iterations, np.array(L_w_b_list).reshape(-1, 1))
plt.show()
```



Results on Training data

In [230]:
```
prediction = sigmoid(np.dot(X_train, w) + b) >= 0.5
prediction = 2 * prediction - 1                         # Convert (0, 1) to (-1, 1)

testing_accuracy = np.sum(prediction == Y_train.reshape(-1, 1))*1.0/X_train.shap

x = np.arange(np.min(X_train[:,0])-1,np.max(X_train[:,0])+1,1.0)
y = (-b[0]-w[0][0]*x)/w[1][0]
plt.scatter(X_train[Y_train==-1, 0], X_train[Y_train==-1, 1], marker='x', color=
plt.scatter(X_train[Y_train==1, 0], X_train[Y_train==1, 1], marker='o', color='r


plt.xlabel('$x_0$')
plt.ylabel('$x_1$')
plt.plot(x,y, 'dodgerblue', label='decision boundary')
plt.title('Training data and decision boundary')

plt.legend(loc='upper right', fontsize=10)
```

Out[230]: `<matplotlib.legend.Legend at 0x21e8e3e28d0>`

Results on Testing data

In [231]:
```
prediction = sigmoid(np.dot(X_test, w) + b) >= 0.5
prediction = 2 * prediction - 1                          # Convert (0, 1) to (-1, 1)
testing_accuracy = np.sum(prediction == Y_test.reshape(-1, 1))*1.0/X_test.shape[
print ("testing accuracy: " + str(testing_accuracy))

x = np.arange(np.min(X_train[:,0])-1,np.max(X_train[:,0])+1,1.0)
y = (-b[0]-w[0][0]*x)/w[1][0]
plt.scatter(X_test[Y_test==-1, 0], X_test[Y_test==-1, 1], marker='x', color='b',
plt.scatter(X_test[Y_test==1, 0], X_test[Y_test==1, 1], marker='o', color='r', a


plt.xlabel('$x_0$')
plt.ylabel('$x_1$')
plt.plot(x,y, 'dodgerblue', label='decision boundary')
plt.title('Testing data and decision boundary')

plt.legend(loc='upper right', fontsize=10)
```

```
testing accuracy: 0.8333333333333334
```

Out[231]: `<matplotlib.legend.Legend at 0x21e8e54f518>`

In [ ]: