COMPSCI 371D Homework 5

Jihyeon Je (jj271), Tim Ho (th265), Rahul Prakash (rp221)

Problem 0 (3 points)

Part 1: The Logistic-Regression Classifier in One Dimension

In [86]:
```
from urllib.request import urlretrieve
from os import path as osp


def retrieve(file_name, semester='fall21', course='371d', homework=5):
    if osp.exists(file_name):
        print('Using previously downloaded file {}'.format(file_name))
    else:
        fmt = 'https://www2.cs.duke.edu/courses/{}/compsci{}/homework/{}
/{}'
        url = fmt.format(semester, course, homework, file_name)
        urlretrieve(url, file_name)
        print('Downloaded file {}'.format(file_name))
```

In [87]:
```
import pickle


file_name = 'data1d.pkl'
retrieve(file_name, homework=5)
with open(file_name, 'rb') as file:
    t = pickle.load(file)
tx, ty = t['x'], t['y']
```

```
Using previously downloaded file data1d.pkl
```

Problem 1.1 (Exam Style)

We were given $f(a) = \frac{1}{1+e^-a}$ and $l(y,p) = -ylogp - (1-y)log(1-p)$

Plugging in f(a) for p, we get $l(y, f(a)) = -ylog\frac{1}{1+e^{-a}} - (1-y)log(1 - \frac{1}{1+e^{-a}})$

$l(y, f(a)) = ylog(1+e^{-a}) - (1-y)(log(e^{-a}) - log(1+e^{-a}))$

$= ylog(1+e^{-a}) - log(e^{-a}) + ylog(e^{-a}) + log(1+e^{-a}) - ylog(1+e^{-a})$

$= -log(e^{-a}) + ylog(e^{-a}) + log(1+e^{-a}) = (y-1)log(e^{-a}) + log(1+e^{-a}) = (1-y)a$
$+ log(1+e^{-a})$

Problem 1.2 (Exam Style)

$l(y, f(a)) = (1-y)a + log(1+e^{-a})$

$\frac{dl}{da} = -y - \frac{e^{-a}}{1+e^{-a}}$

$\frac{d^2l}{da^2} = \frac{e^{-a}}{1+e^{-a}} - e^{-2a}(1+e^{-a})^{-2} = \frac{e^{-a}}{(1+e^{-a})^2}$

Because $e^{-a} > 0$ and $(1+e^{-a})^2 > 0$, therefore $\frac{e^{-a}}{(1+e^{-a})^2} > 0$, so we know that the function $l(y, f(a))$ is globally stricly convex function of a.

Problem 1.3 (Exam Style)

$\frac{d^2l}{db^2} = \frac{d}{db}(\frac{dl}{da}\frac{da}{db}) = \frac{e^{-b-wx}}{(1+e^{-b-wx})^2} > 0$

$\frac{d^2l}{dw^2} = \frac{d}{dw}(\frac{dl}{da}\frac{da}{dw}) = \frac{x^2e^{b+wx}}{(1+e^{b+wx})^2} \geq 0$

$L_T(b, w) = \frac{1}{N}\sum_{n=1}^{N} l(y_n, s(x_n))$

$\frac{d^2L_T(b,w)}{db^2} = \frac{1}{N}[\frac{d^2}{db^2}l(y_1, s(x_1)) + \frac{d^2}{db^2}l(y_2, s(x_2)) + \ldots + \frac{d^2}{db^2}l(y_N, s(x_N))] > 0$

$\frac{d^2L_T(b,w)}{dw^2} = \frac{1}{N}[\frac{d^2}{dw^2}l(y_1, s(x_1)) + \frac{d^2}{dw^2}l(y_2, s(x_2)) + \ldots + \frac{d^2}{dw^2}l(y_N, s(x_N))] \geq 0$

Therefore, we can conclude that the risk L_T is a convex function of its argument b and w.

Problem 1.4 (Exam Style)

$L_T(b,w) = \frac{1}{N}\sum_{n=1}^{N} l(y_n, s(x_n)) = \frac{1}{N}\sum_{n=1}^{N}(1-y)a + log(1+e^{-a})$
$= \frac{1}{N}\sum_{n=1}^{N}(1-y)(b+wx_n) + log(1+e^{-b-wx_n}) = \frac{1}{N}\sum_{n=1}^{N}(1-y)(b) + log(1+e^{-b})$

because $w = 0$

we can remove the summation term as $y = 0$ for F samples and $y = 1$ for $N - F$ samples

$L_T(b,0) = \frac{1}{N}[F(b + log(1+e^{-b})) + (N-F)log(1+e^{-b})]$

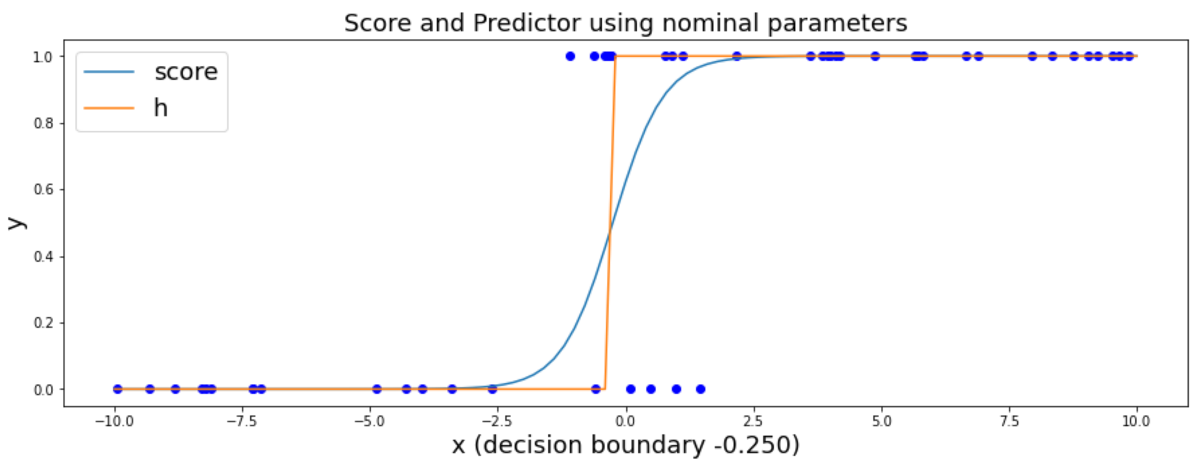
$L_T(0,0) = \frac{1}{N}[F(log(2)) + (N-F)log(2)] = \frac{1}{N}[Nlog(2)] = log(2)$

Problem 1.5

In [88]:
```
import numpy as np

def logistic(a):
    f = 1/(1+np.exp(-a))
    return f

def affine(x,v):
    b,w= v
    return b + (w*x)

def score(x,v):
    s = logistic(affine(x,v))
    return s

def h(x,v):
    hvals = (score(x,v)>0.5)*1
    return hvals
```

In [89]:
```
import matplotlib.pyplot as plt

def plot_score(x,y,v, loss_name, x_bound, n_points, type_size):
    x_pts = np.linspace(-x_bound,x_bound,num = n_points,endpoint=True)
    plt.figure(figsize=(15,5))
    plt.title(f"Score and Predictor using {loss_name}",fontsize= type_si
ze)
    plt.plot(x,y,'bo')
    plt.plot(x_pts,score(x_pts,v), label='score')
    plt.plot(x_pts,h(x_pts,v), label='h')
    idx = np.where(h(x_pts,v)==1)[0][0]
    hbound = (-1)*v[0]/v[1]
    plt.legend(fontsize= type_size)
    plt.xlabel(f"x (decision boundary {hbound:.3f})", fontsize= type_siz
e)
    plt.ylabel("y", fontsize= type_size)
```

In [90]:
```
v_hat_test = np.array((0.5, 2.))
plot_score(tx, ty, v_hat_test, 'nominal parameters',
    x_bound=10., n_points=101, type_size=18)
```



Problem 1.6

In [91]:
```
def cross_entropy(y,p):
    if (y==1):
        return -1*np.log(p)
    else:
        return -1*np.log(1-p)

def sample_loss(x, y, v, loss):
    s = score(x,v)
    sam_l = loss(y,s)
    return sam_l

def risk(x,y,v,loss=cross_entropy):
    totloss = 0
    for i in range(0,len(x)):
        totloss+= sample_loss(x[i], y[i], v, loss)
    avg = totloss / len(x)
    return avg
```

In [92]:
```
from scipy.optimize import minimize

def risk2(v,x,y,loss=cross_entropy):
    totloss = 0
    for i in range(0,len(x)):
        totloss+= sample_loss(x[i], y[i], v, loss)
    avg = totloss / len(x)
    return avg

def train(x,y, loss= cross_entropy):
    ret = minimize(risk2, [0,0], args=(x, y,loss), method='CG')
    return ret.x
```

In [93]:
```
v_hat_ce = train(tx, ty)
print(v_hat_ce)
```

```
[0.47471946 0.7714649 ]
```

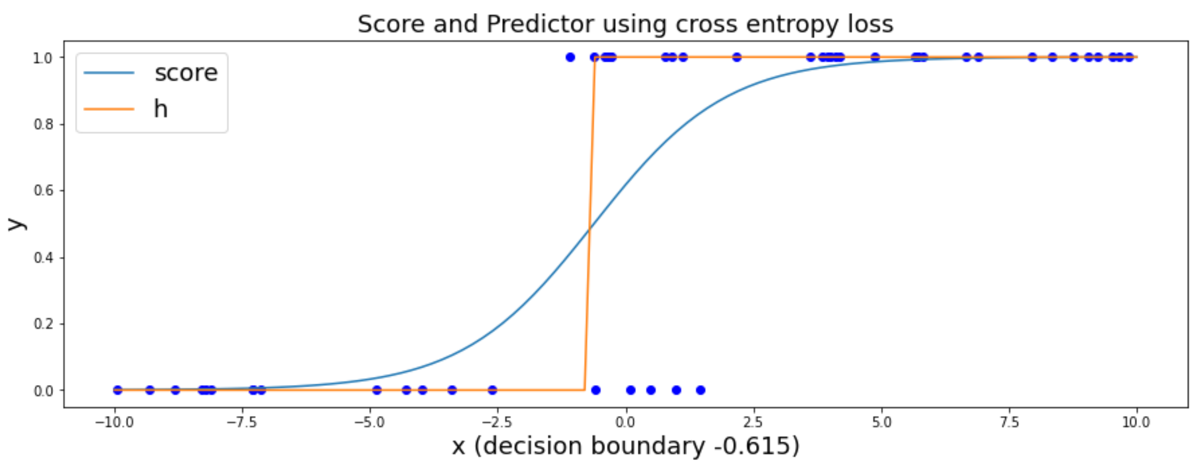
Problem 1.7

In [94]:
```
ce_name = 'cross entropy loss'
x_bound, n_points = 10., 101
type_size = 18
```

In [95]:
```
def plot_losses(x, y, v, loss_fct, loss_name, type_size):

    preds = h(x,v)
    tp_x = []
    tp_y = []
    tn_x = []
    tn_y = []
    fp_x = []
    fp_y = []
    fn_x = []
    fn_y = []

    for i in range(0,len(preds)):
        if ((y[i]==1)&(preds[i]==1)):
            tp_x.append(x[i])
            tp_y.append(sample_loss(x[i],y[i],v, loss = loss_fct))
        if ((y[i]==0)&(preds[i]==0)):
            tn_x.append(x[i])
            tn_y.append(sample_loss(x[i],y[i],v, loss = loss_fct))
        if ((y[i]==1)&(preds[i]==0)):
            fn_x.append(x[i])
            fn_y.append(sample_loss(x[i],y[i],v, loss = loss_fct))
        if ((y[i]==0)&(preds[i]==1)):
            fp_x.append(x[i])
            fp_y.append(sample_loss(x[i],y[i],v, loss = loss_fct))

    hbound = (-1)*v[0]/v[1]
    rsk =  risk(x,y,v,loss=loss_fct)

    print(len(tp_x)+len(tn_x)+len(fn_x)+len(fp_x))
    print(len(x))
    plt.figure(figsize=(15,5))
    plt.title(f"Training Point Losses. Decision Boundary x= {hbound:.3f}
. Risk = {rsk:.3f}",fontsize= type_size)

    plt.plot(tp_x,tp_y,'bo', label = 'TP')
    plt.plot(tn_x,tn_y,'ro', label = 'TN')
    plt.plot(fp_x,fp_y,'o',  color='orange',label = 'FP')
    plt.plot(fn_x,fn_y,'go', label = 'FN')
    plt.axvline(x = hbound, color = 'black')
    plt.xlabel('x',fontsize= type_size)
    plt.ylabel(f'{loss_name}',fontsize= type_size)

    plt.legend(fontsize= type_size)
```

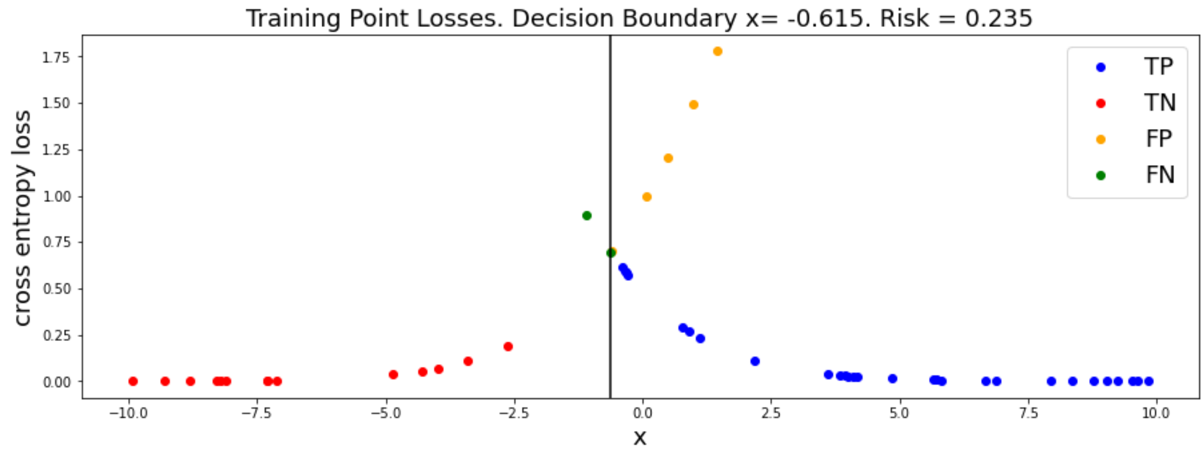
In [96]:
```
import matplotlib.cm as cm

def plot_contours(x, y, v, loss_fct, loss_name, n_points, type_size, fig
_size=(15, 12)):

    b_hat,w_hat = v
    b = np.linspace(b_hat-3., b_hat+3., n_points, endpoint=True)
    w = np.linspace(-0.1, w_hat+5., n_points, endpoint=True)
    box = (b[0], b[-1], w[0], w[-1])
    b_grid, w_grid = np.meshgrid(b, w)
    zgrid = risk(x,y,[b_grid,w_grid],loss_fct)

    fig = plt.figure(figsize=fig_size, tight_layout=True)
    img = plt.imshow(zgrid, interpolation='bilinear',
               origin='lower', extent=box, cmap=cm.hot)
    plt.contour(b_grid, w_grid, zgrid, 50, colors='w', linewidths=1)

    plt.axis('scaled')
    plt.xticks(fontsize=type_size)
    plt.yticks(fontsize=type_size)
    plt.title(f"Risk Using {loss_name}",fontsize=type_size)
    plt.xlabel('b', fontsize=type_size)
    plt.ylabel('w', fontsize=type_size)
    bar = fig.colorbar(img, shrink=1)

    bar.ax.tick_params(labelsize=type_size)



    plt.show()
```

In [97]:
```
plot_score(tx, ty, v_hat_ce, ce_name, x_bound, n_points, type_size)
```

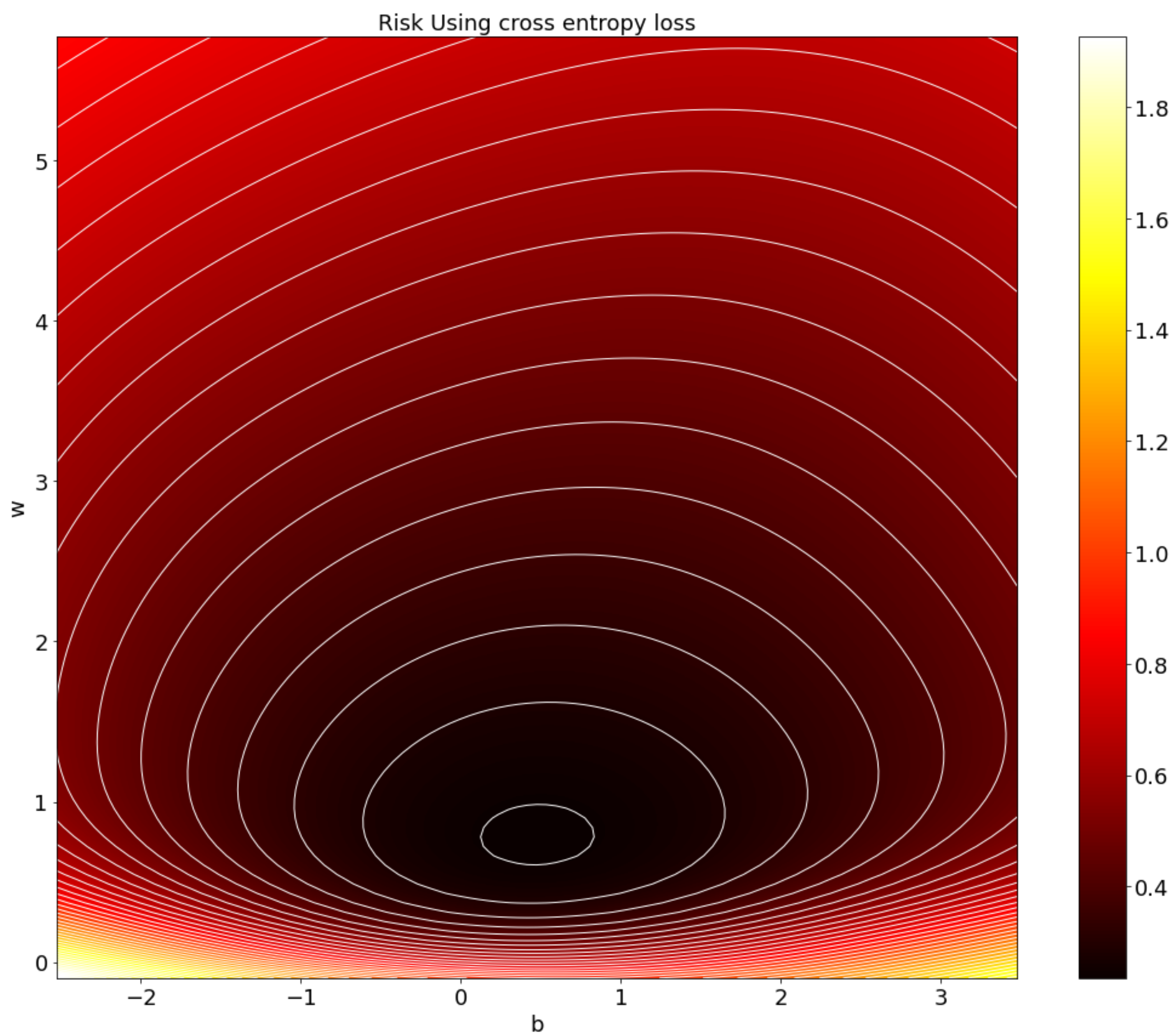


In [98]:
```
plot_losses(tx, ty, v_hat_ce, cross_entropy, ce_name, type_size)
```

```
50
50
```



In [99]:
```
plot_contours(tx, ty, v_hat_ce, cross_entropy, ce_name, n_points, type_size)
```

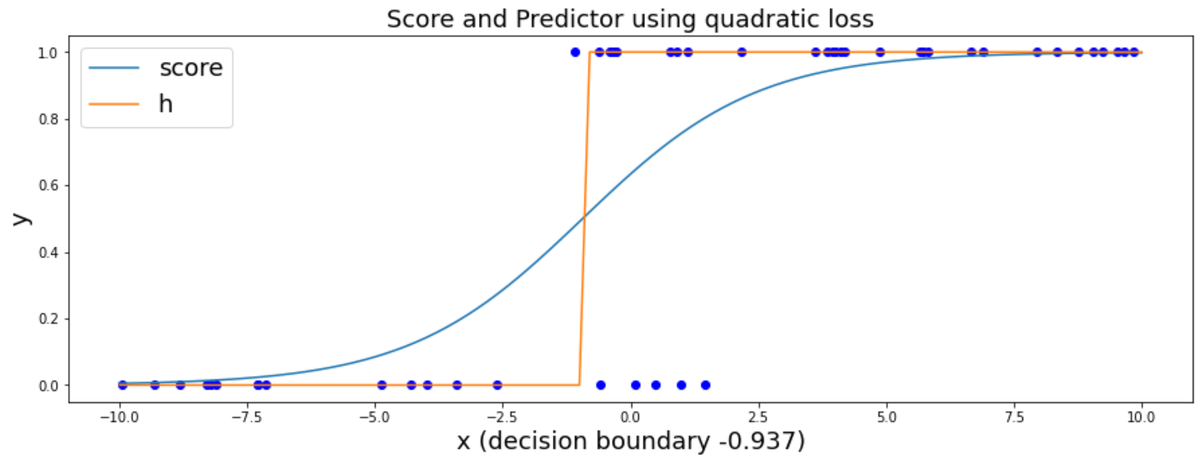


Problem 1.8

In [100]:
```
def quadratic(y,p):
    loss = (y-p)**2
    return loss
```

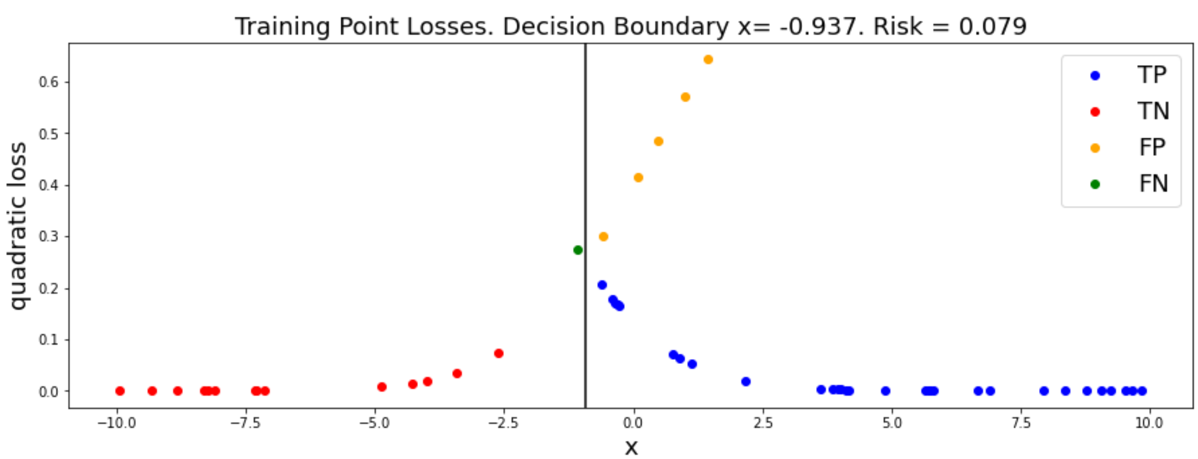
In [101]:
```
v_hat_q = train(tx, ty, loss=quadratic)
q_name = 'quadratic loss'
```

In [102]:
```
plot_score(tx, ty, v_hat_q, q_name, x_bound, n_points, type_size)
```

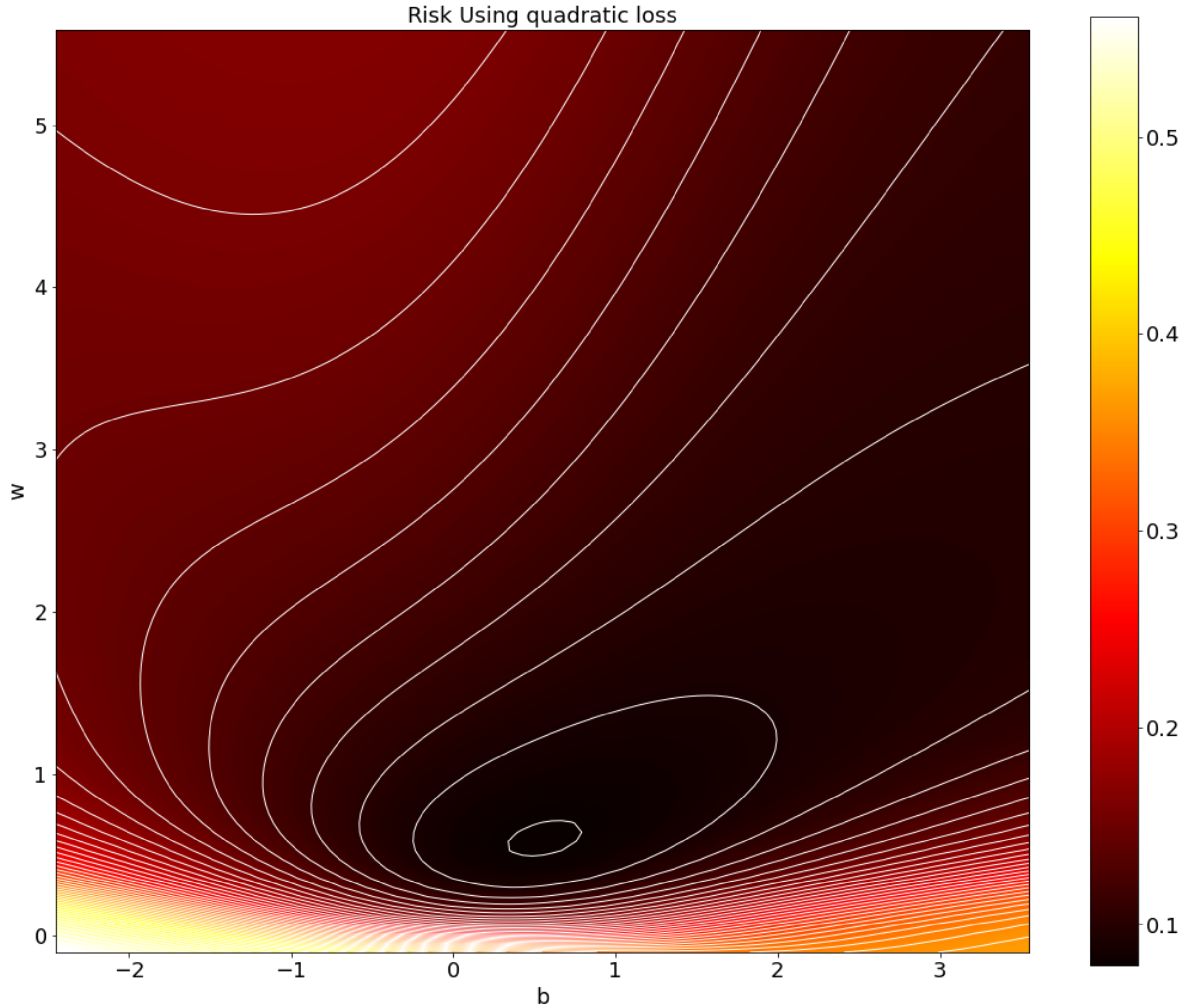


In [103]:
```
plot_losses(tx, ty, v_hat_q, quadratic, q_name, type_size)
```

```
50
50
```



In [104]:
```
plot_contours(tx, ty, v_hat_q, quadratic, q_name, n_points, type_size)
```



Problem 1.9 (Exam Style)

From the plot above, we can conclude that the risk with the quadratic loss is not convex everywhere. In the plot near the top left corner, convexity is not met as the curvatures of the countour lines start to deviate and change in convexivity. We see that convexivity is violated especially w>4 and b< -1 regions.

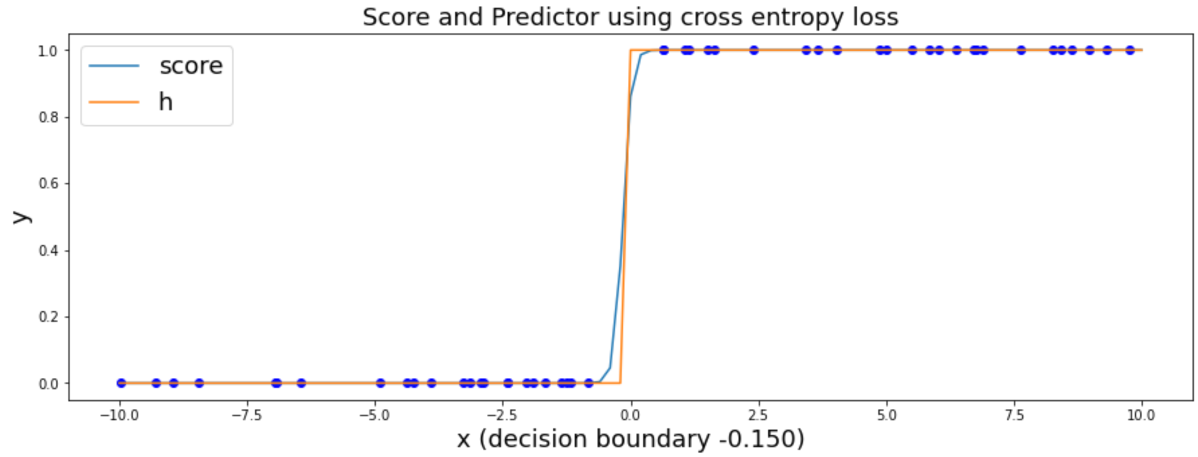
Problem 1.10

In [105]:
```
file_name_sep = 'data1d_sep.pkl'
retrieve(file_name_sep)
with open(file_name_sep, 'rb') as file:
    t_sep = pickle.load(file)
tx_sep, ty_sep = t_sep['x'], t_sep['y']
```

```
Using previously downloaded file data1d_sep.pkl
```

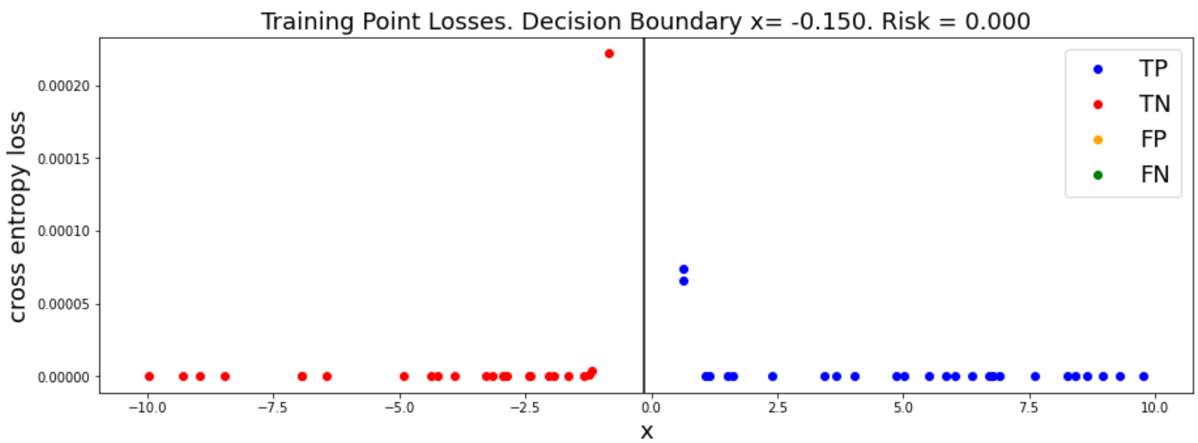
In [106]:
```
v_hat_ce_sep = train(tx_sep, ty_sep, loss=cross_entropy)
```

In [107]:
```
plot_score(tx_sep, ty_sep, v_hat_ce_sep, ce_name, x_bound, n_points, typ
e_size)
```

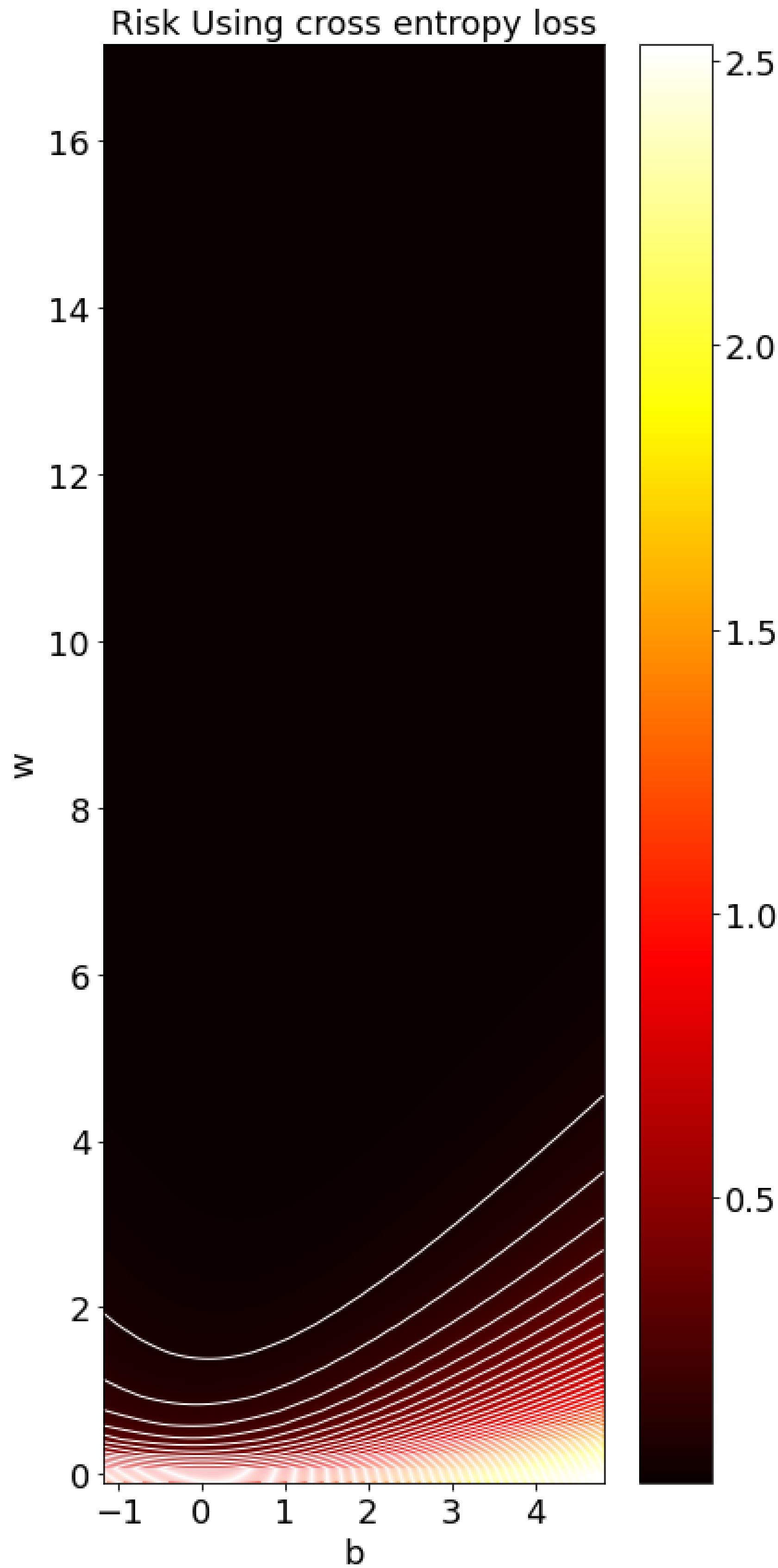


In [108]:
```
plot_losses(tx_sep, ty_sep, v_hat_ce_sep, cross_entropy, ce_name, type_s
ize)
```

```
50
50
```



In [109]:
```
import matplotlib.cm as cm

def plot_contours(x, y, v, loss_fct, loss_name, n_points, type_size, fig
_size=(6, 12)):

    b_hat,w_hat = v
    b = np.linspace(b_hat-3., b_hat+3., n_points, endpoint=True)
    w = np.linspace(-0.1, w_hat+5., n_points, endpoint=True)
    box = (b[0], b[-1], w[0], w[-1])
    b_grid, w_grid = np.meshgrid(b, w)
    zgrid = risk(x,y,[b_grid,w_grid],loss_fct)

    fig = plt.figure(figsize=fig_size, tight_layout=True)
    img = plt.imshow(zgrid, interpolation='bilinear',
               origin='lower', extent=box, cmap=cm.hot)
    plt.contour(b_grid, w_grid, zgrid, 50, colors='w', linewidths=1)

    plt.axis('scaled')
    plt.xticks(fontsize=type_size)
    plt.yticks(fontsize=type_size)
    plt.title(f"Risk Using {loss_name}",fontsize=type_size)
    plt.xlabel('b', fontsize=type_size)
    plt.ylabel('w', fontsize=type_size)
    bar = fig.colorbar(img, shrink=1)

    bar.ax.tick_params(labelsize=type_size)



    plt.show()
plot_contours(tx_sep, ty_sep, v_hat_ce_sep, cross_entropy, ce_name, n_po
ints, type_size)
```


Risk Using cross entropy loss
w
16
14
12
10
8
6
4
2
0
−1
0
1
2
3
4
b
2.5
2.0
1.5
1.0
0.5

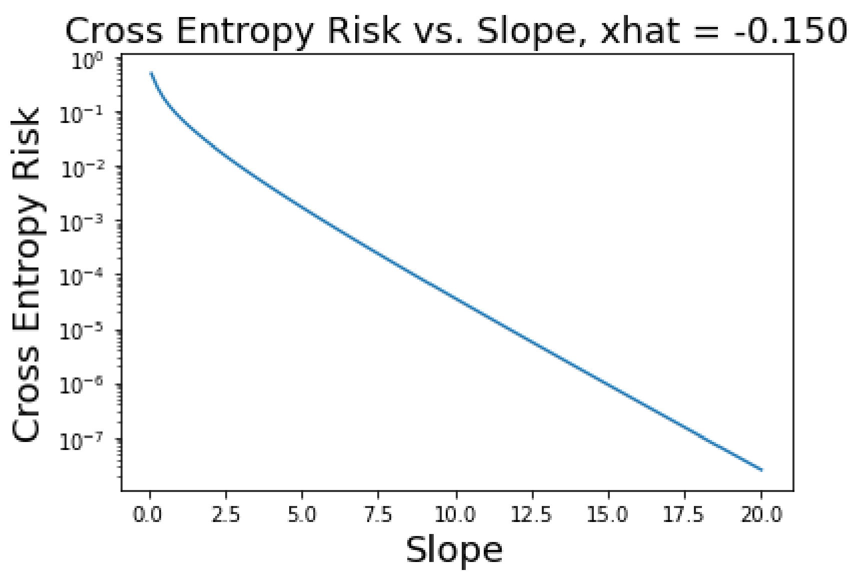
Problem 1.11

In [110]:
```
v_hat_ce_sep = train(tx_sep, ty_sep, loss=cross_entropy)

xhat =   (-1)*v_hat_ce_sep[0]/v_hat_ce_sep[1]
print(xhat)
```

```
-0.14989424847706337
```

In [111]:
```
slopes = np.linspace(0.1,20,n_points, endpoint=True)
b = -xhat*slopes
rsk = []
for i in range(0,len(slopes)):
    rsk.append(risk(tx_sep,ty_sep, [b[i],slopes[i]],loss=cross_entropy))
plt.semilogy(slopes,rsk)
plt.title(f'Cross Entropy Risk vs. Slope, xhat = {xhat:.3f}', fontsize=1
8)
plt.xlabel('Slope', fontsize=18)
plt.ylabel('Cross Entropy Risk', fontsize=18)
```

Out[111]:
```
Text(0, 0.5, 'Cross Entropy Risk')
```



From the plot above, we see that as you increase the slope, cross entropy loss decreases sharply.

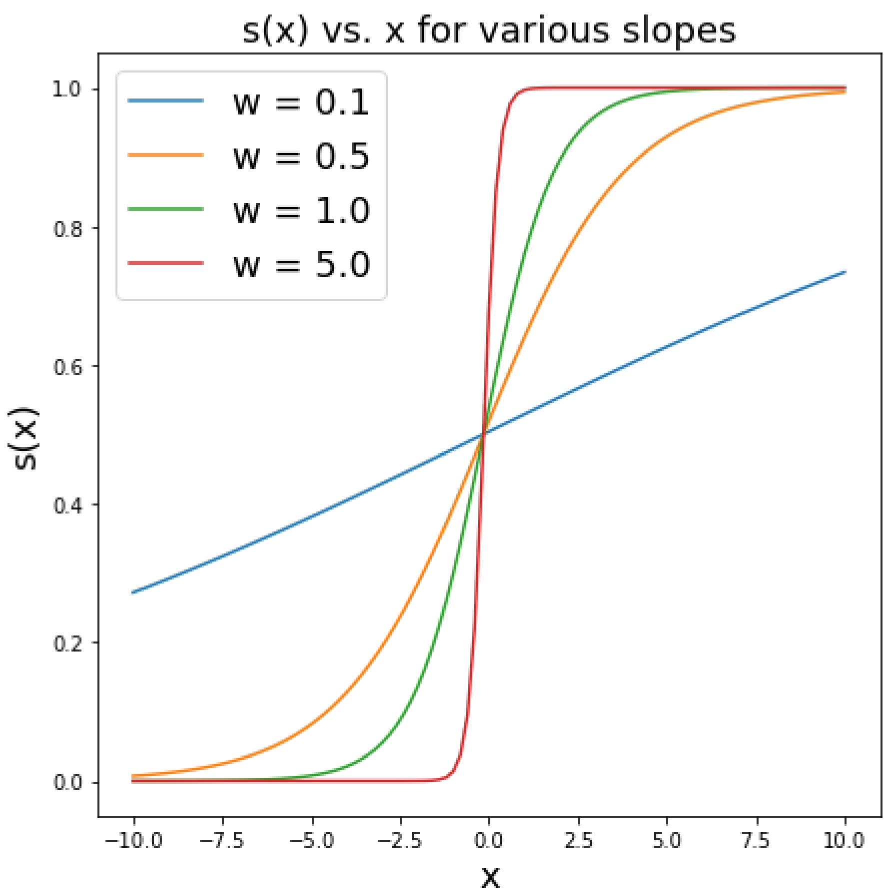
Problem 1.12 (Exam Style)

From the trend suggested by the plot in the previous problem, the algorithm likely stopped once the loss went below a certain threshold due to preset configurations in scipy.optimize.minimize. We see that as we increase the slope, cross entropy decreases sharply. As we keep increasing the slope, the minimization algorithm probably stops optimization once the magnitude change in the risk becomes insignificantly small.

Problem 1.13

In [112]:
```
x = np.linspace(-10,10,n_points)
w = np.array([0.1,0.5,1.,5.])
b = -xhat*w
scores = []
plt.figure(figsize =(7,7))
for i in range(0,4):
    s = score(x,[b[i],w[i]])
#     plt.subplot(2,2,i+1)
    plt.plot(x,s,label=f"w = {w[i]}")
    plt.ylabel("s(x)", fontsize=18)
    plt.xlabel("x", fontsize=18)
plt.legend(fontsize=18)
plt.title(f"s(x) vs. x for various slopes", fontsize=18)
```

Out[112]:
```
Text(0.5, 1.0, 's(x) vs. x for various slopes')
```



As the slope is increased, the score function starts as a linear function and transitions to narrower logistic functions.

Problem 1.14 (Exam Style)

$$b = -\hat{\chi} * w$$
$$\ell(y, p) = -y \log p - (1 - y) \log(1 - p)$$
$$\ell_s(x, y) = \ell(y, s(x))$$
$$\ell_s(x, False) = \ell(0, s(x)) = (-1) \log(1 - s(x)) = -\log(1 - \frac{1}{1 + e^{-a}})$$

$$ = -\log(1-\frac{1}{1+e^{-b -w*x}}) = \log(1-\frac{1}{1+e^{w(\hat{\chi}-\chi)}})$$

$$\ell_s(x, True) = \ell(1, s(x)) = (-1) \log(s(x)) = (-1) \log(\frac{1}{1 + e^{-a}}) = -\log(\frac{1}{1 + e^{w(\hat{\chi}-\chi)}})$$

when $y = True, \hat{\chi} - \chi < 0$ and when $y = False, \hat{\chi} - \chi > 0$

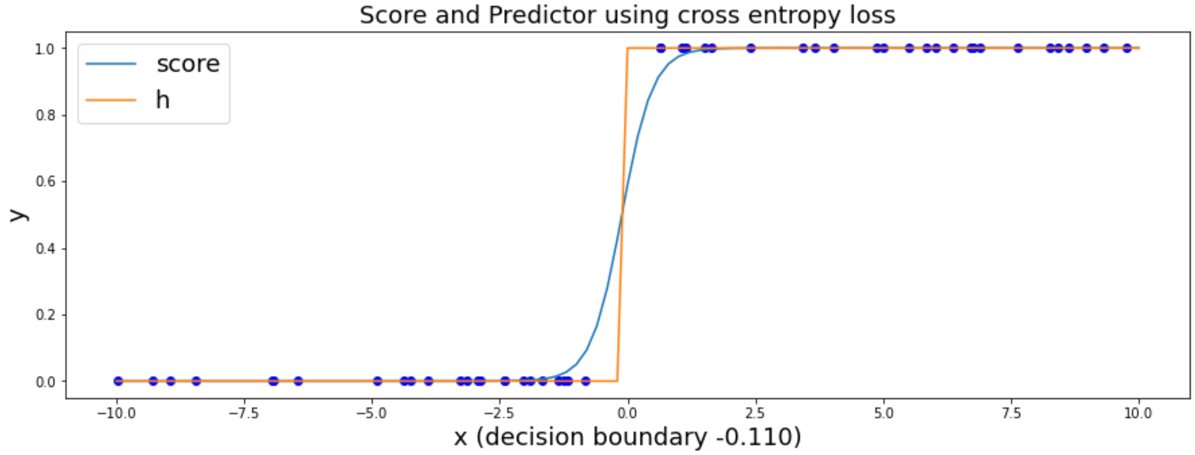
$$\lim_{w\to\infty} l(0, s(x)) = -\log(1 - \frac{1}{1 + e^{\infty}}) = 0$$
$$\lim_{w\to\infty} l(1, s(x)) = -\log(\frac{1}{1 + e^{-\infty}}) = 0$$

Problem 1.15

In [113]:
```
def emprisk(v,x,y,mu,loss=cross_entropy):
    totloss = 0
    for i in range(0,len(x)):
        totloss+= sample_loss(x[i], y[i], v, loss)
    avg = totloss / len(x) + mu * (v[0]**2 + v[1]**2)
    return avg


def train_reg(x,y, loss= cross_entropy, mu = 1.e-3):
    ret = minimize(emprisk, [0,0], args=(x, y,mu,loss), method='CG')
    return ret.x
```

In [114]:
```
v_hat_reg = train_reg(tx_sep,ty_sep)
print(f"vhat_reg={v_hat_reg}")
plot_score(tx_sep, ty_sep, v_hat_reg, ce_name, x_bound, n_points, type_s
ize)
```

```
vhat_reg=[0.3635696  3.29559968]
```



From the plots generated we see that the score function is wider with a larger decision boundary with regularization (since it is penalizing larger w and b).