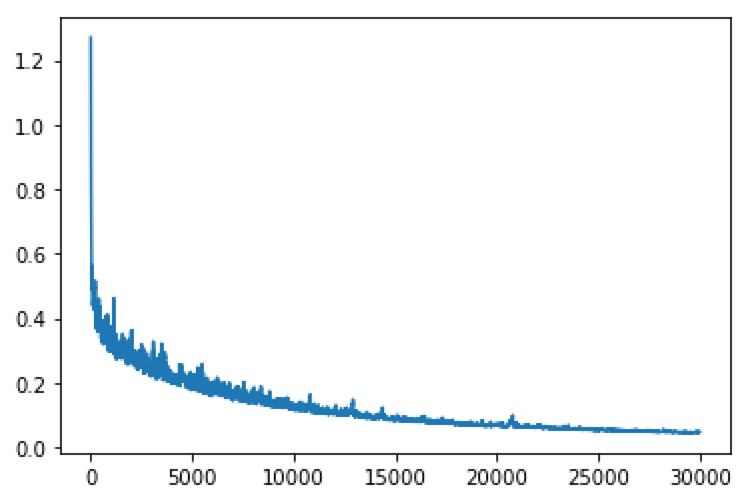
Problem 1

In [9]:

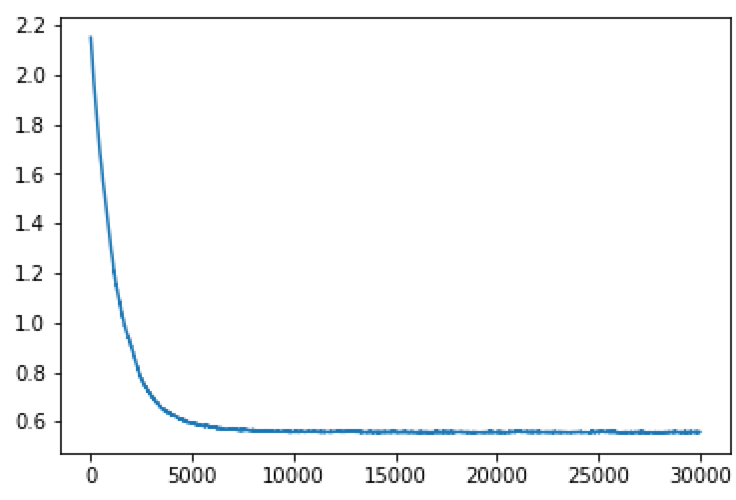
```
import numpy as np
import matplotlib.pyplot as plt


N , p = 30 , 20
np . random . seed (0)
X = np . random . randn (N , p )
Y = 2* np . random . randint (2 , size = N ) - 1


theta = np.random.rand(p)


def f(theta):
    sum = 0
    for i in range(N):
        sum += np.log(1+np.exp(-Y[i]*X[i].T@theta))
    return sum/N


epoch =  1000
lr = 0.1
result = []

for _ in range(epoch):
    for rep in range(N):
        i = np.random.randint(30)
        Xi = X[i]
        Yi = Y[i]

        gradient = -Yi * Xi/(1+np.exp(Yi * (Xi.T @ theta)))

        theta = theta - gradient * lr

        result.append(f(theta))

x = np.arange(epoch*N)
plt.plot(x, result)
plt.show()
```



Problem 2

In [12]:

```
import numpy as np
import matplotlib.pyplot as plt


N , p = 30 , 20
np . random . seed (0)
X = np . random . randn (N , p )
Y = 2* np . random . randint (2 , size = N ) - 1


theta = np.random.rand(p)
l = 0.1

def f(theta):
    sum = 0
    for i in range(N):
        sum += max(0, 1-Y[i]*X[i].T@theta) + l*np.linalg.norm(theta)**2
    return sum/N


epoch =  1000
lr = 0.001
result = []
encounter = False

for _ in range(epoch):
    for rep in range(N):
        i = np.random.randint(30)
        Xi = X[i]
        Yi = Y[i]

        encounter = encounter or (Yi*Xi.T@theta ==1)

        gradient = 2*l*theta - Yi*Xi*(Yi*Xi.T@theta < 1)

        theta = theta - gradient * lr

        result.append(f(theta))

x = np.arange(epoch*N)
plt.plot(x, result)
plt.show()

print("Does it encounter a point of non-differentiability? : ", encounter)
```

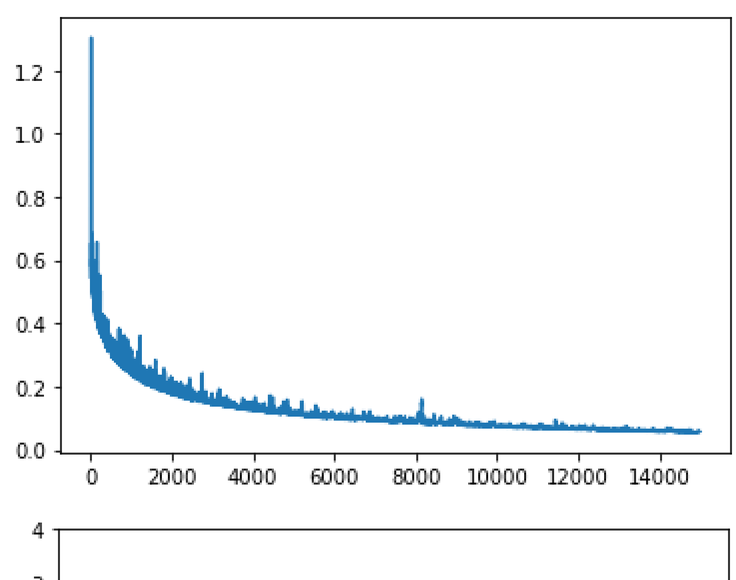


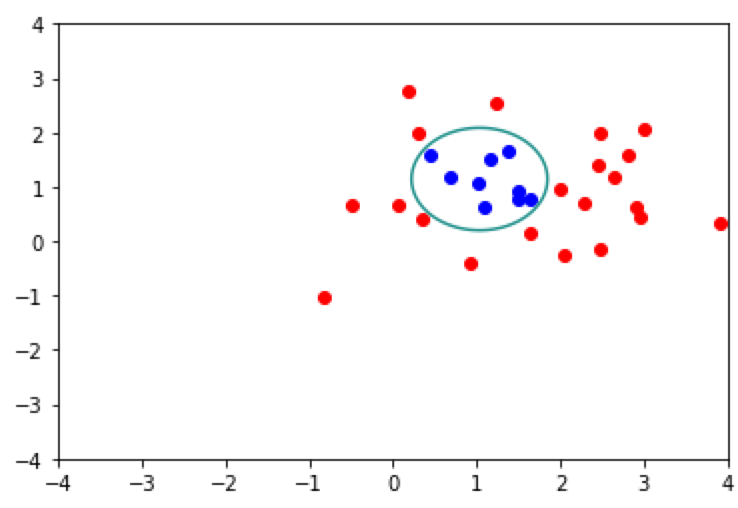

```
Does it encounter a point of non-differentiability? :  False
```

Problem 3

In [32]:

```
import numpy as np
import matplotlib.pyplot as plt

N =30
np . random . seed (0)
X = np . random . randn (2 , N )
y = np . sign ( X [0 ,:]**2+ X [1 ,:]**2 -0.7)
theta = 0.5
c , s = np . cos ( theta ) , np . sin ( theta )
X = np . array ([[ c , -s ] , [s , c ]]) @X
X = X + np . array ([[1] ,[1]])



xcoord_red =[]
ycoord_red =[]
xcoord_blue = []
ycoord_blue = []
for i in range(N):
    if(y[i]==1):
        xcoord_red.append(X[0][i])
        ycoord_red.append(X[1][i])
    else:
        xcoord_blue.append(X[0][i])
        ycoord_blue.append(X[1][i])




w = np.random.rand(5)

phiX = []
for i in range(N):
    phiX.append([1, X[0][i], X[0][i]**2, X[1][i], X[1][i]**2 ])
phiX = np.array(phiX)


def f(w):
    sum = 0
    for i in range(N):
        sum += np.log(1+np.exp(-y[i]*phiX[i].T@w))
    return sum/N



epoch =  500
lr = 0.1
result = []

for _ in range(epoch):
    for rep in range(N):
        i = np.random.randint(N)
        Xi = phiX[i]
        Yi = y[i]

        gradient = -Yi * Xi/(1+np.exp(Yi * (Xi.T @ w)))

        w = w - gradient * lr


        result.append(f(w))

x = np.arange(epoch*N)
plt.plot(x, result)
plt.show()



xx = np . linspace ( -4 , 4 , 1024)
yy = np . linspace ( -4 , 4 , 1024)
xx , yy = np . meshgrid ( xx , yy )
Z = w [0] + ( w [1] * xx + w [2] * xx **2) + ( w [3] * yy + w [4] * yy **2)
plt . contour ( xx , yy , Z , 0)
plt.scatter(xcoord_red, ycoord_red, color='red')
plt.scatter(xcoord_blue, ycoord_blue, color='blue')
plt.show()

linearly_separable = True
for i in range(N):
    if(phiX[i]@w * y[i] <0 ):
        linearly_separable = False
print("linearly separable with the help of kernel methods : ", linearly_separable)
```






```
linearly separable with the help of kernel methods :  True
```

In [ ]:

#4. $\phi''(x) \geq 0$ then $\phi(x)$ is convex.

pf) $\forall x_1, x_2 \in C, \quad x_1 \leq x_2, \quad \eta \in (0,1)$

$$\frac{\phi(\eta x_1 + (1-\eta)x_2) - \phi(x_1)}{(1-\eta)(x_2 - x_1)} = \phi'(x_3) \quad \exists x_3 \in (x_1, \eta x_1 + (1-\eta)x_2) \text{ by Mean Value Theorem}$$

$$\frac{\phi(x_2) - \phi(\eta x_1 + (1-\eta)x_2)}{\eta(x_2 - x_1)} = \phi'(x_4) \quad \exists x_4 \in (\eta x_1 + (1-\eta)x_2, x_2) \text{ by M.V.T}$$

$$\frac{\phi'(x_4) - \phi'(x_3)}{x_4 - x_3} = \frac{1}{(x_4 - x_3)}\frac{1}{(x_2 - x_1)}\left(\frac{\phi(x_2) - \phi(\eta x_1 + (1-\eta)x_2)}{\eta} - \frac{\phi(\eta x_1 + (1-\eta)x_2) - \phi(x_1)}{(1-\eta)}\right) = \phi''(c) \geq 0$$

$\exists c \in (x_3, x_4)$ by M.V.T

$$\therefore (1-\eta)(\phi(x_2) - \phi(\eta x_1 + (1-\eta)x_2)) \geq \eta(\phi(\eta x_1 + (1-\eta)x_2) - \phi(x_1))$$

$$\Leftrightarrow \eta\phi(x_1) + (1-\eta)\phi(x_2) \geq \phi(\eta x_1 + (1-\eta)x_2) \quad \blacksquare$$

이제, $D_{KL}(p\|q) \geq 0$ 임을 보이자.

pf) $f(x) = -\ln(x) \Rightarrow f'(x) = -\frac{1}{x} \Rightarrow f''(x) = \frac{1}{x^2} \geq 0 \quad (x > 0)$

$\therefore$ since $f''(x) > 0$, $f(x)$ is convex. Therefore, by Jensen's inequality $(P(I=i) = p_i)$

$$D_{KL}(p\|q) = \mathbb{E}_I\left[\log\left(\frac{p_I}{q_I}\right)\right] = \mathbb{E}_I\left[-\log\left(\frac{q_I}{p_I}\right)\right] \geq -\log\left(\mathbb{E}_I\left[\frac{q_I}{p_I}\right]\right)$$

$$= -\log\left(\sum_{i=1}^{n} p_i \frac{q_i}{p_i}\right) = -\log(1) = 0.$$

$$D_{KL}(p\|q) \geq 0 \quad \blacksquare$$

#5 $\phi''(x) > 0$ then $\phi(x)$ is strictly convex:

pf) $x_1 < x_2$, $\eta \in (0,1)$

$$\frac{\phi(\eta x_1 + (1-\eta)x_2) - \phi(x_1)}{(1-\eta)(x_2 - x_1)} = \phi'(x_3) \quad \exists x_3 \in (x_1, \eta x_1 + (1-\eta)x_2) \text{ by M.V.T}$$

$$\frac{\phi(x_2) - \phi(\eta x_1 + (1-\eta)x_2)}{\eta(x_2 - x_1)} = \phi'(x_4) \quad \exists x_4 \in (\eta x_1 + (1-\eta)x_2, x_2) \text{ by M.V.T}$$

$$\frac{\phi'(x_4) - \phi'(x_3)}{x_4 - x_3} = \phi''(c) > 0 \quad \exists c \in (x_3, x_4) \text{ by M.V.T}$$

$\Leftrightarrow$ $(1-\eta)(\phi(x_2) - \phi(\eta x_1 + (1-\eta)x_2)) > \eta(\phi(\eta x_1 + (1-\eta)x_2) - \phi(x_1))$

$\Leftrightarrow$ $\eta\phi(x_1) + (1-\eta)\phi(x_2) > \phi(\eta x_1 + (1-\eta)x_2)$: strictly convex! ▭

이제, $D_{KL}(p\|q) > 0$ $(p \neq q)$ 임을 보이자.

$f(x) = -\ln(x) \Rightarrow f''(x) = \frac{1}{x^2} > 0$ $(x > 0)$ 이므로 f는 strictly convex. $\therefore$ By Jensen,

$\phi(E[X]) < E[\phi(X)]$ holds for non constant r.v X.

Consider random variable $\frac{p_I}{q_I}$ where $P(I=i) = p_i$. since $p \neq q$, $\frac{p_I}{q_I}$ is a non constant r.v

$$\therefore D_{KL}(p\|q) = E_I\left[-\log\frac{q_I}{p_I}\right] > -\log\left(E_I\left[\frac{q_I}{p_I}\right]\right) = -\log\left(\sum_{i=1}^{n} p_i \cdot \frac{q_i}{p_i}\right) = -\log 1$$

$$= 0$$

$\therefore D_{KL}(p\|q) > 0$

▭

#6 $f_\theta(x) = \mathbf{u}^T \sigma(\mathbf{a}x+\mathbf{b}) = \sum_{i=1}^{p} u_j \sigma(a_j x + b_j)$

$$\frac{\partial f_\theta(x)}{\partial u_i} = \sigma(a_j x + b_j)$$ 이므로, $\nabla_u f_\theta(x) = (\sigma(a_1 x + b_1), \cdots, \sigma(a_p x + b_p)) = \underline{\sigma(\mathbf{a}x+\mathbf{b})}$

$$\frac{\partial f_\theta(x)}{\partial b_i} = u_j \sigma'(a_j x + b_j)$$ 이므로, $\nabla_b f_\theta(x) = (u_1 \sigma'(a_1 x + b_1), \cdots, u_p \sigma'(a_p x + b_p))$

$$= \sigma'(\mathbf{a}x+\mathbf{b}) \odot \mathbf{u} = \begin{pmatrix} a_1 x + b_1 & & 0 \\ & a_2 x + b_2 & \\ 0 & & \ddots \\ & & & a_p x + b_p \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_p \end{pmatrix} = \mathrm{diag}(\sigma'(\mathbf{a}x+\mathbf{b}))\mathbf{u}$$

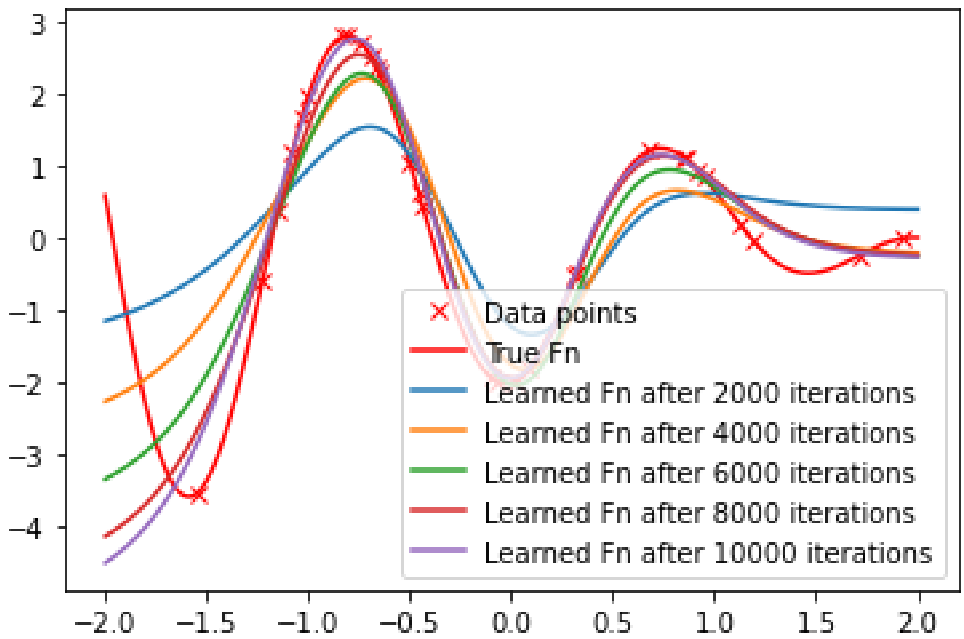
$$\frac{\partial f_\theta(x)}{\partial a_i} = u_j \sigma'(a_j x + b_j) x$$ 이므로, $\nabla_a f_\theta(x) = x(u_1 \sigma'(a_1 x + b_1), \cdots, u_p \sigma'(a_p x + b_p))$

$$= (\sigma'(\mathbf{a}x+\mathbf{b}) \odot \mathbf{u})x = x \begin{pmatrix} a_1 x + b_1 & & 0 \\ & a_2 x + b_2 & \\ 0 & & \ddots \\ & & & a_p x + b_p \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \end{pmatrix} = (\mathrm{diag}(\sigma'(\mathbf{a}x+\mathbf{b}))\mathbf{u})x$$

Problem 7

twolayerSGD (1).py

```python
import numpy as np
import matplotlib.pyplot as plt

def f_true(x) :
    return (x-2)*np.cos(x*4)

def sigmoid(x) :
    return 1 / (1 + np.exp(-x))

def sigmoid_prime(x) :
    return sigmoid(x) * (1 - sigmoid(x))

K = 10000
alpha = 0.007
N, p = 30, 50
np.random.seed(0)
a0 = np.random.normal(loc = 0.0, scale = 4.0, size = p)
b0 = np.random.normal(loc = 0.0, scale = 4.0, size = p)
u0 = np.random.normal(loc = 0, scale = 0.05, size = p)
theta = np.concatenate((a0,b0,u0))


X = np.random.normal(loc = 0.0, scale = 1.0, size = N)
Y = f_true(X)

def f_th(theta, x) :
    return np.sum(theta[2*p : 3*p] * sigmoid(theta[0 : p] * np.reshape(x,(-1,1)) + theta[p : 2*p]), axis=1)

def diff_f_th(theta, x) :
    a = theta[0:p]
    b = theta[p:2*p]
    u = theta[2*p:]

    fa = x * (sigmoid_prime(a*x + b)*u)
    fb = (sigmoid_prime(a*x + b)*u)
    fu = sigmoid(a*x+b)

    return np.concatenate((fa,fb,fu))

xx = np.linspace(-2,2,1024)
plt.plot(X,f_true(X),'rx',label='Data points')
plt.plot(xx,f_true(xx),'r',label='True Fn')

for k in range(K) :

    i = np.random.randint(N)

    gradient = (f_th(theta, X[i]) - Y[i]) * diff_f_th(theta,X[i])

    theta = theta - alpha * gradient


    if (k+1)%2000 == 0 :
        plt.plot(xx,f_th(theta, xx),label=f'Learned Fn after {k+1} iterations')

plt.legend()
plt.show()
```



위와 같은 결과를 얻을 수 있다.