This homework is done by Tianwei Mo.

1.

a)

$$\frac{\partial J}{\partial w_1} = a_1 w_2 e^{z_1 z_2} + 2 z_1 a_1 a_2 w_1 w_2 e^{z_1 z_2}$$

$$\frac{\partial J}{\partial w_2} = a_1 w_1 e^{z_1 z_2} + z_1 (a_1 a_2 w_1^2 + 2 a_1 a_3 w_2) e^{z_1 z_2}$$

b) The python function:

```
import numpy as np

def Jeval(w, a):
    a1 = a[0]
    a2 = a[1]
    a3 = a[2]
    w1 = w[0]
    w2 = w[1]
    z1 = a1*w1*w2
    z2 = a2*w1+a3*w2**2
    J = z1*np.exp(z1*z2)

    dw1 = a1*w2*np.exp(z1*z2)+2*z1*a1*a2*w1*w2*np.exp(z1*z2)
    dw2 = a1*w1*np.exp(z1*z2)+z1*(a1*a2*w1**2+2*a1*a3*w2)*np.exp(z1*z2)
    Jgrad = np.array([dw1, dw2])

    return J, Jgrad
```

2.

a)

$$\frac{\partial J}{\partial w_j} = \sum_{i=1}^{N} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_j} = \sum_{i=1}^{N} 2(\log \hat{y}_i - \log y_i) x_{ij}$$

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{N} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial b} = \sum_{i=1}^{N} 2(\log \hat{y}_i - \log y_i)$$

b) The python function:

```python
import numpy as np

def Jeval(w, b, x, y):
    yhat = x*w+b
    J = np.sum((np.log(y)-np.log(yhat))**2)

    Jgradw = np.sum(2*(np.log(y)-np.log(yhat))*x)
    Jgradb = np.sum(2*(np.log(y)-np.log(yhat)))

    return J, Jgradw, Jgradb
```

3.

a) Assume $A = [1\ X]$, matrix with ones on the first column.

$$z_i = w_0 + \sum_{j=1}^{d} w_j x_{ij} = \sum_{j=0}^{d} w_j A_{ij}$$

$$\frac{\partial J}{\partial w_j} = \sum_{i=1}^{n} \frac{\partial J}{\partial z_i}\frac{\partial z_i}{\partial w_j} = \sum_{i=1}^{n}\left(-\frac{2}{z_i^3} + \frac{2y_i}{z_i^2}\right) A_{ij}$$

b) The python function:

```python
import numpy as np

def Jeval(w, x, y):
    i = x.shape[0]
    one = np.ones((i, 1))
    A = np.hstack((one, x))
    z = w*A
    J = np.sum((y-1/z)**2)

    Jgrad = np.sum((-2/z**3+2*y/z**2)*A)

    return J, Jgrad
```

4.

a)

$$\frac{\partial J}{\partial a_j} = \frac{\partial J}{\partial z_i}\frac{\partial z_i}{\partial a_j} = \sum_{i=1}^{N}\left(\frac{z_i + e^{z_i}}{1 + e^{z_i}} - y_i\right) e^{-\frac{(x_i - b_j)^2}{2}}$$

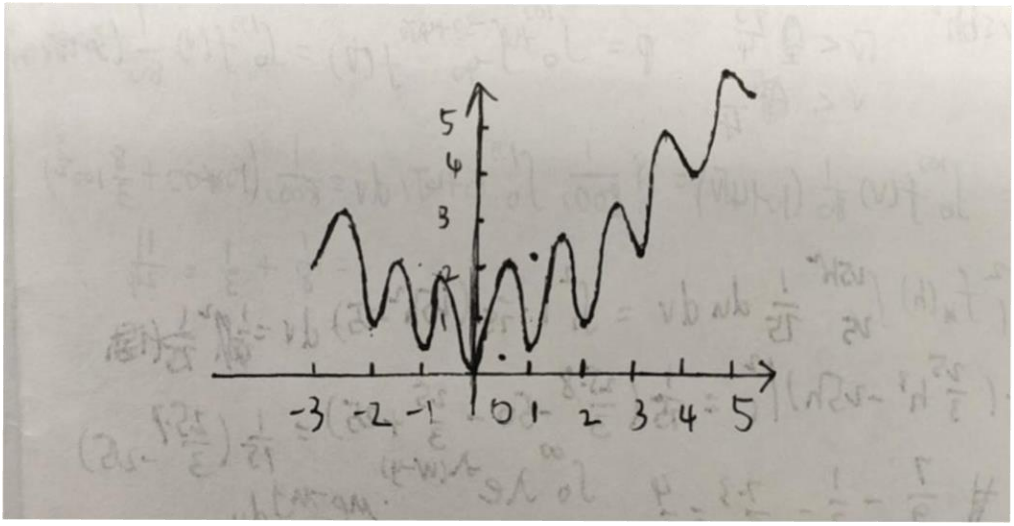
$$\frac{\partial J}{\partial b_j} = \frac{\partial J}{\partial z_i}\frac{\partial z_i}{\partial b_j} = \sum_{i=1}^{N}\left(\frac{z_i + e^{z_i}}{1 + e^{z_i}} - y_i\right) a_j (x_i - b_j) e^{-\frac{(x_i - b_j)^2}{2}}$$

b) The python function:

```python
import numpy as np

def Jeval(a, b, x, y):
    z = np.sum(a*np.exp(-((x-b)**2)/2))
    J = np.sum(np.log(1+np.exp(z))-y*z)

    Jgrada = np.sum(((z+np.exp(z))/(1+np.exp(z))-y)*np.exp(-((x-b)**2)/2))
    Jgradb = np.sum(((z+np.exp(z))/(1+np.exp(z))-y)*a*(x-b)*np.exp(-((x-b)**2)/2))

    return J, Jgrada, Jgradb

```

5.

a) Draft of $f(x)$:



b)

$$\frac{df(x)}{dx} = \frac{1}{2}x + 2\pi \sin 2\pi x$$

$$x^{k+1} = x^k - a_k\left(\frac{1}{2}x^k + 2\pi \sin 2\pi x^k\right)$$

c) The global minima of $f(x)$ is 0.

d) Let initial x to be 1.1. By gradient decent, it would converge to 1. However, 1 is a local minima instead of global minima.

6.

a)

$$\nabla J(\boldsymbol{w}) = [b_1 w_1, b_2 w_2]$$

b)

$$\boldsymbol{w}^* = [0, 0]$$

c)

$$w_1^{k+1} = w_1^k - a\nabla f\left(w_1^k\right) = (1 - ab_1)w_1^k$$

$$w_2^{k+1} = w_2^k - a\nabla f\left(w_2^k\right) = (1 - ab_2)w_2^k$$

$$\rho_1 = 1 - ab_1$$
$$\rho_2 = 1 - ab_2$$

d)

$$w^k \to w^* = w^k \to 0$$
$$(1 - ab_1)w_1^k \to 0, (1 - ab_2)w_2^k \to 0$$

A should be a number such that ab_1 and ab_2 are both close to 1.

e)

$$w_1^{k+1} = \left(1 - \frac{2b_1}{b_1 + b_2}\right)w_1^k = \left(\frac{b_2 - b_1}{b_1 + b_2}\right)w^k = \left(\frac{\kappa - 1}{\kappa + 1}\right)w^k = Cw^k$$

Thus,

$$w^k = C^k w^0$$

7.

a)

$$\nabla_{\mathbf{P}} z_i = x_i^T x_i$$

b) A

$$\nabla_{\mathbf{P}} J(\mathbf{P}) = \nabla_{z_i} J(\mathbf{P}) \nabla_{\mathbf{P}} z_i = \sum_{i=1}^{n} \left(\frac{1}{y_i} - \frac{1}{z_i}\right) x_i^T x_i$$

c) The python function:

```python
def Jeval_with_loop(P, x, y):
    z = x.T@P@x
    n = y.shape[0]

    J = np.zeros(y.shape)
    for i in range(n):
        J[i, :] = np.sum(z[i, :]/y[i, :]-np.log(z[i, :]))

    JgradP = np.zeros(y.shape)
    for i in range(n):
        JgradP = np.sum(1/y[i, :]-1/z[i, :])*(x[i, :].T@x[i, :])

```

d) The python function:

```python
import numpy as np

def Jeval(P, x, y):
    z = x.T@P@x
    J = np.sum(z/y-np.log(z))

    JgradP = np.sum(1/y-1/z)*(x.T@x)

    return J, JgradP
```

8.

a)

$$J_1(w_1) = \min_{w_2} J(w_1, w_2) = J\big(w_1, \widehat{w}_2(w_1)\big)$$

$$\nabla_{w_1} J_1(w_1) = \nabla_{w_1} J\big(w_1, \widehat{w}_2(w_1)\big) = \nabla_{w_1} J(w_1, w_2)|_{w_2=\widehat{w}_2}$$

b)

$$\nabla_{\mathrm{b}} J(a, b) = 2\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{d} b_j e^{-a_j x_i}\right)\sum_{j=1}^{d} e^{-a_j x_i}$$

$$b^{k+1} = b^k - a_k \nabla J_b(a, b)$$

We find b^* when we iterate through above equation.

c)

$$\nabla_{\mathrm{a}} J(a, b) = 2\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{d} b_j e^{-a_j x_i}\right)\sum_{j=1}^{d} b_j x_i e^{-a_j x_i}$$