In [ ]:
```
import numpy as np
import copy
import matplotlib.pyplot as plt
import math
import itertools
from mpl_toolkits.mplot3d import Axes3D
from mlrefined_libraries import math_optimization_library as optlib
from sklearn.linear_model import LinearRegression
static_plotter = optlib.static_plotter.Visualizer();
```

4.1

proof:

a. ∵ matrix C is a Hessian Matrix ∴ C is a symmetry matrix

∴ $C = Q\Lambda Q^T$ ∵ $\lambda_i \geq 0$ ∴ $C = Q\Lambda^{\frac{1}{2}}(\Lambda^{\frac{1}{2}})^T Q^T$

let $R = (Q\Lambda^{\frac{1}{2}})^T$ ∴ $C = R^T R$ ∴ R's row is Linear independence.

∴ $z^T C z = z^T R^T R z = (\|Rz\|)^2 \geq 0$

b. ∵ C is a symmetry matrix ∴ C can be apply to element diagonalization, so $C = Q\Lambda Q^T$ C is a positive definite matrix

∴ $z^T Q\Lambda Q^T z > 0$ let $y = Q^T z = [y_1 \dots y_n]^T$

∴ $y\Lambda y^T \geq 0 \Rightarrow \sum_{i=1}^{n} \lambda_i y_i^2 \geq 0$

C. one-order: $\frac{\partial g(w)}{\partial w} = b + Cw + C^T w$ $\frac{\partial^2 g(w)}{\partial w^2} = C + C = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$

d: $\begin{bmatrix} \lambda-1+k & -1 \\ -1 & \lambda-1+k \end{bmatrix} = 0$ $(\lambda-1+k)^2 = 1$ $\lambda_1 = 2-k$

$\lambda_2 = -k$

∴ $k_{min} = -1$

5.2

In [ ]:
```
def mode(x,w):
    y_hat = np .dot(x, w)
    return y_hat
def least_squares (y,x,w):
    cost = np. sum (( mode(x, w) - y) ** 2)
    return cost / (2*float(y. size))
def linear_gradient_descent(x,y,w,alpha=0.1,max_its=100):
    gradient = lambda w : (1 / float(y.size)) * (x.T.dot(x.dot(w) - y))
    weight_history = [w]
    cost_history = [least_squares(y,x,w)]
    for _ in range(max_its):
        grad_eval = gradient(w)
        w = w - alpha*grad_eval
        weight_history.append(w)
        cost_history.append(least_squares(y,x,w))
    return weight_history,cost_history

data_path="data/"
file_name="kleibers_law_data.csv"
csv_name=data_path+file_name
data=np.loadtxt(csv_name,delimiter=',')
x=data[:-1,:]
y=data[-1:,:]
#print(np.log(1370))
x=[np.log(xi) for xi in x]
y=[np.log(yi) for yi in y]
x = np.reshape(x,(-1,1))
y=np.reshape(y,(-1,1))
model=LinearRegression()
model.fit(x,y)
w0=model.intercept_
w1=model.coef_
y=lambda x:np.log(w0+w1*x)
print(w0)
print(w1)
print(y(10))
```

```
[6.81473477]
[[0.6528121]]
[[2.59098109]]
```

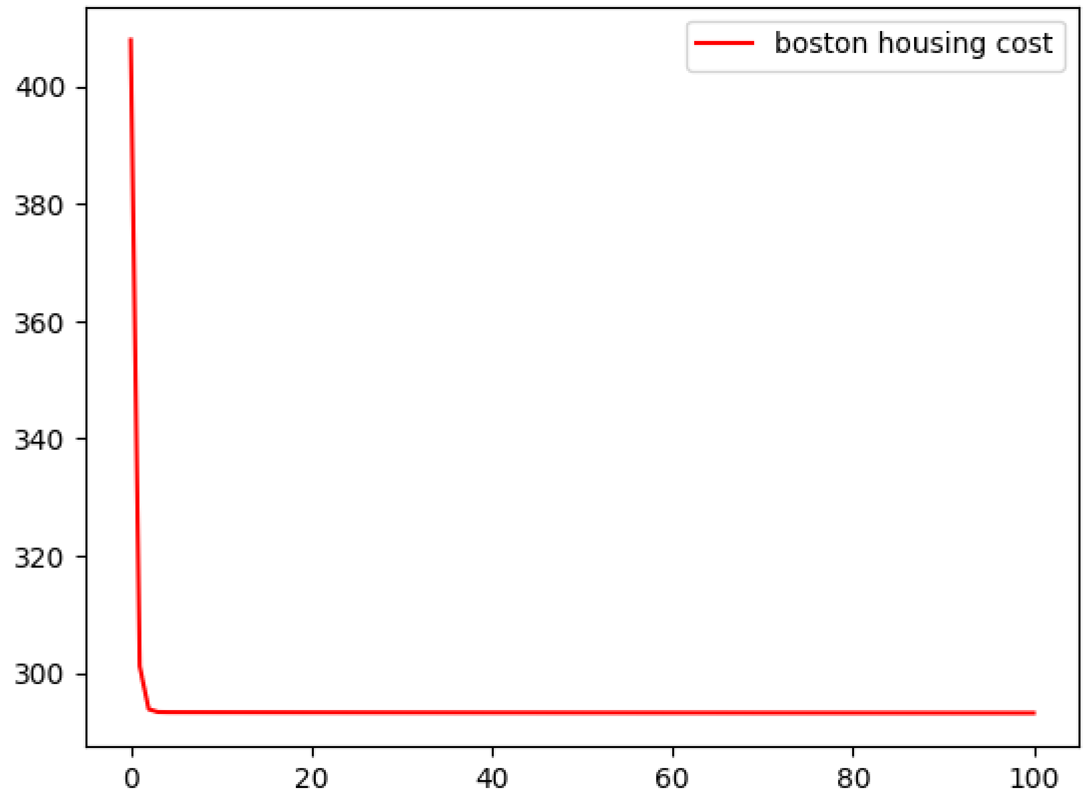
c. $y = x^{0.6528121} + e^{6.81473477}$

d. 2.59098109

In [ ]:
```
#5.9
def normalization(data):
    range = np.max(data) - np.min(data)
    return (data - np.min(data)) / range
def standardization(data):
    mu = np.mean(data, axis=0)
    sigma = np.std(data, axis=0)
    return (data - mu) / sigma
file_name="boston_housing.csv"
csv_name=data_path+file_name
data=np.loadtxt(csv_name,delimiter=',')
x = data[:-1,:]
y = data[-1:,:]
print(np.max(x))
x=np.reshape(x,(506,13))
x=standardization(x)
y=np.reshape(y,(506,1))
print(np.shape(x))
print(np.shape(y))
w=np.ones((13,1))
_,cost_mse_bos=linear_gradient_descent(x,y,w)
file_name="auto_data.csv"
csv_name=data_path+file_name
data=np.loadtxt(csv_name,delimiter=',')
xx = data[:-1,:]
yy = data[-1:,:]
xx=np.reshape(xx,(398,7))
print(np.max(xx))
xx=normalization(xx)
yy=np.reshape(yy,(398,1))
print(np.shape(xx))
print(np.shape(yy))
w=np.ones((7,1))
_,cost_mse_mobile=linear_gradient_descent(xx,yy,w)
w=np.linspace(0,100,101)

plt.plot(w,cost_mse_bos,color='r',label="boston housing cost")
plt.legend()
plt.show()
```

```
711.0
(506, 13)
(506, 1)
nan
(398, 7)
(398, 1)
```



In [ ]:

```python
#6.5
def mod(x,w):
    a=w[0]+np.dot(x.T,w[1:])
def sigmoid(t):
    return 1/(1+np.exp(-t))
def cross_entropy(w):
    a=sigmoid(mod(x,w))
    ind=np.argwhere(y==0)[:,1]
    cost=-np.sum(np.log(1-a[:,ind]))
    ind=np.argwhere(y==1)[:,1]
    cost-=np.sum(np.log(a[:,ind]))
    return cost/y.size
```

6 – 5

$$g(w) = -\frac{1}{P}\sum_{p=1}^{P} y_p \log(\sigma(x_p^T w)) + (1-y_p)\log(1-\sigma(x_p^T w))$$

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\cdot(1-\sigma(x))$$

$$\frac{\partial g(w)}{\partial w} = -\frac{1}{P}\sum_{p=1}^{P} \frac{y_p x_p}{\sigma(x_p^T w)} \cdot \sigma(x_p^T w)\cdot(1-\sigma(x_p^T w)$$

$$= -\frac{1}{P}\sum_{p=1}^{P} (y_p - \sigma(x_p^T w))\, x_p$$

$$\frac{\partial^2 g(w)}{\partial w^2} = \frac{1}{P}\sum_{p=1}^{P} \sigma(x_p^T w)(1-\sigma(x_p^T w))\, x_p x_p^T$$

6 − 11

$$g(w) = \frac{1}{P}\sum_{p=1}^{P} \log(1+e^{-y_p x_p^T w}) = -\frac{1}{P}\left[\sum_{i=1}^{P}\sum_{j=1}^{k} 1\{y^{(i)}=j\} \log \frac{e^{w_j^T x_i}}{\sum_{l=1}^{k} e^{w_l^T x_i}}\right]$$

k is the number of classes

let $a_j = \dfrac{e^{w_j^T x}}{\sum_{l=1}^{k} e^{w_l^T x}}$

when $n \neq j$ $\quad \nabla_{w_n} a_j = \dfrac{-e^{w_j^T x} e^{w_n^T x}}{\left(\sum_{l=1}^{k} e^{w_l^T x}\right)^2} = -a_j a_n x$

$n = j$ $\quad \nabla_{w_n} a_j = a_j (1-a_j) x$

$\therefore \nabla_{w_n} g(w) = -\dfrac{1}{P}\sum_{i=1}^{P}\left[x_i\left(1\{y^{(i)}=n\} - a_n\right)\right]$

$\therefore \nabla^2 g(w) = \dfrac{1}{P}\sum_{i=1}^{P} a_n(1-a_n) x_i x_i^T$

$\therefore$ it is a positive semidefinite matrix

$\therefore$ it is a convex function

19 / 19