LinearCostFunction

April 14, 2023

[1]:
```
import numpy as np
%matplotlib widget
import matplotlib.pyplot as plt
from lab_utils_uni import plt_intuition, plt_stationary, plt_update_onclick,␣
 ↪soup_bowl
plt.style.use('./deeplearning.mplstyle')
```

[2]:
```
x_train = np.array([1.0, 2.0])           #(size in 1000 square feet)
y_train = np.array([300.0, 500.0])           #(price in 1000s of dollars)
```

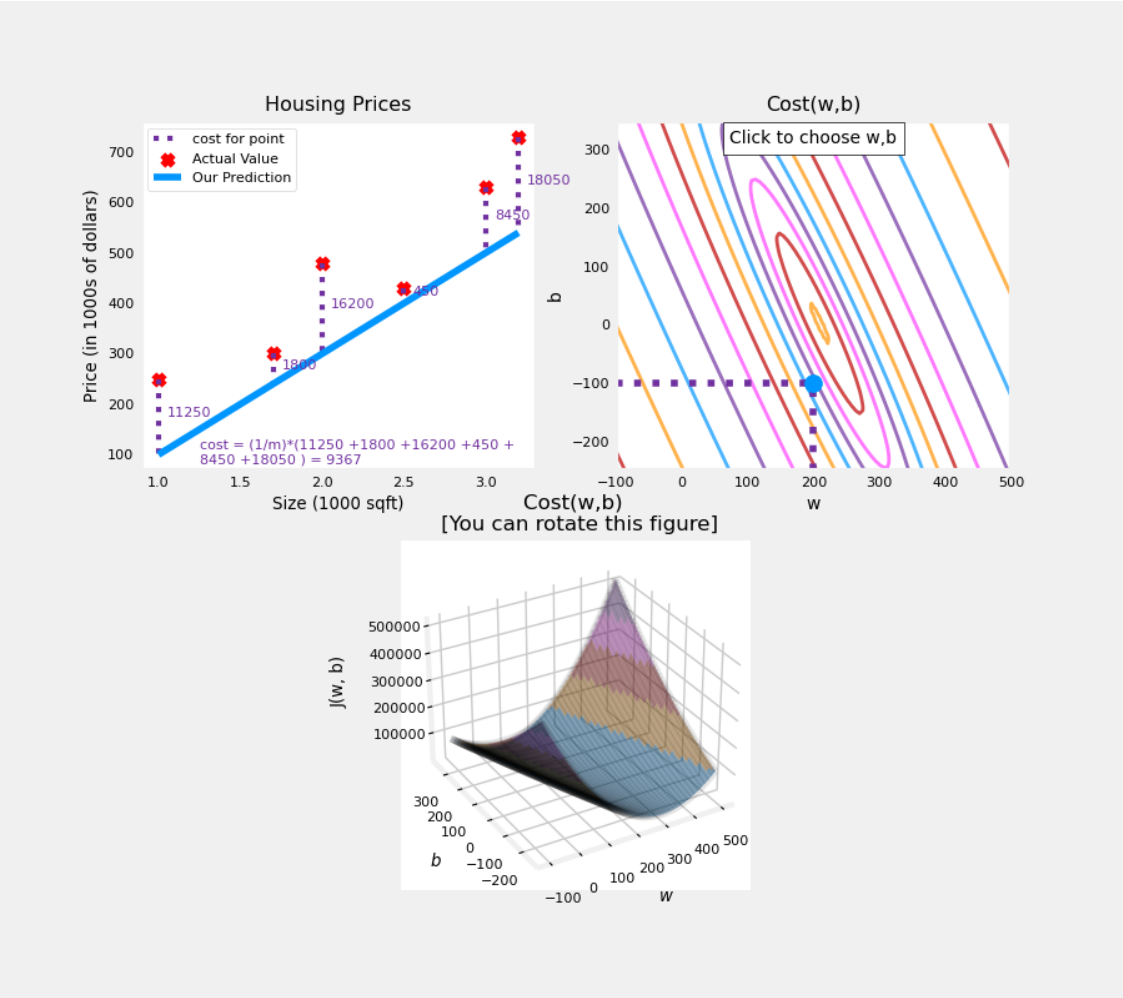
[3]:
```
def compute_cost(x, y, w, b):
    """
    Computes the cost function for linear regression.

    Args:
      x (ndarray (m,)): Data, m examples
      y (ndarray (m,)): target values
      w,b (scalar)    : model parameters

    Returns
        total_cost (float): The cost of using w,b as the parameters for linear␣
 ↪regression
               to fit the data points in x and y
    """
    # number of training examples
    m = x.shape[0]

    cost_sum = 0
    for i in range(m):
        f_wb = w * x[i] + b
        cost = (f_wb - y[i]) ** 2
        cost_sum = cost_sum + cost
    total_cost = (1 / (2 * m)) * cost_sum

    return total_cost
```

[4]:
```
plt_intuition(x_train,y_train)
```

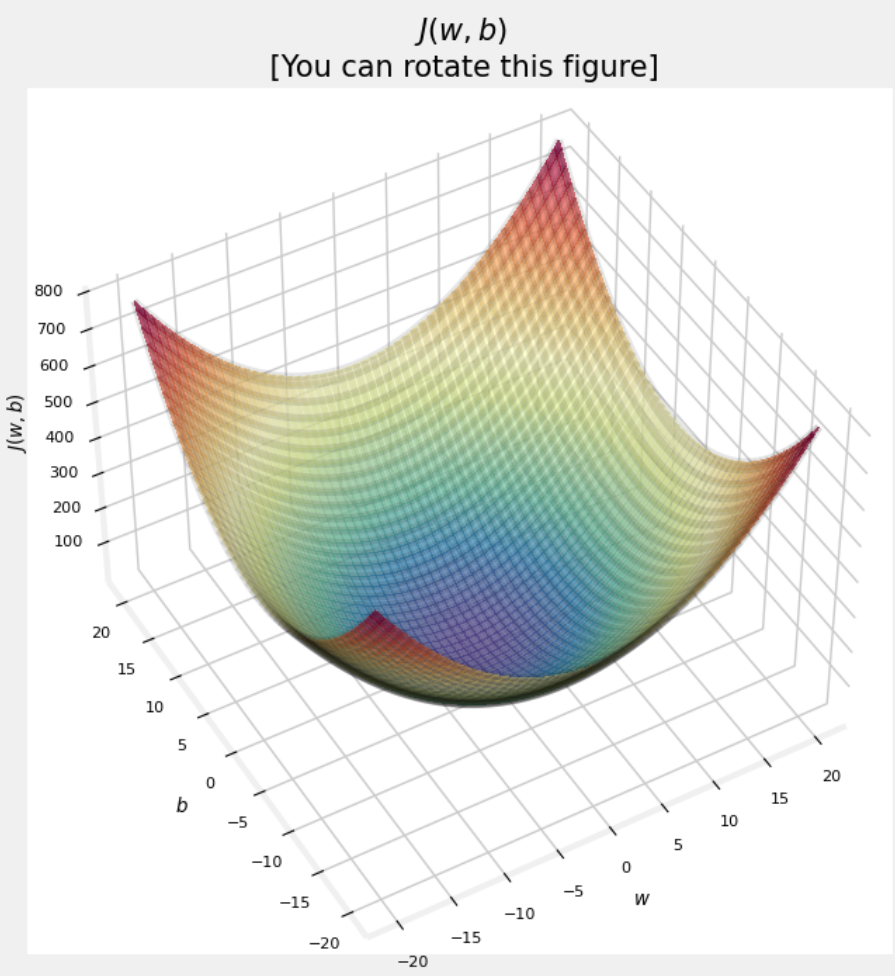
```
interactive(children=(IntSlider(value=150, description='w', max=400, step=10),␣
↪Output()), _dom_classes=('widge…
```

[5]:
```
x_train = np.array([1.0, 1.7, 2.0, 2.5, 3.0, 3.2])
y_train = np.array([250, 300, 480,  430,   630, 730,])
```

[6]:
```
plt.close('all')
fig, ax, dyn_items = plt_stationary(x_train, y_train)
updater = plt_update_onclick(fig, ax, x_train, y_train, dyn_items)
```



[7]:
```
soup_bowl()
```


$J(w, b)$
[You can rotate this figure]
800
700
600
500
400
300
200
100
$J(w, b)$
20
15
10
5
0
−5
−10
−15
−20
b
20
15
10
5
0
−5
−10
−15
−20
w

[ ]: