Untitled2

March 20, 2025

[14]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

#
class linear_regression():
    def fitness(self, Date_X_input, Date_Y, learning_rate=0.5, lamda=0.03):
        sample_num, property_num = Date_X_input.shape#
        Date_X = np.c_[Date_X_input, np.ones(sample_num)]
        self.theta = np.zeros([property_num + 1, 1])        #     theta
        Max_count = int(1e8)  #
        last_better = 0  #
        last_Jerr = int(1e8)  #
        threshold_value = 1e-8  #
        threshold_count = 10  #
        for step in range(0, Max_count):
            predict = Date_X.dot(self.theta)            #
            J_theta = sum((predict - Date_Y) ** 2) / (2 * sample_num)
 ↪ #
            self.theta -= learning_rate * (lamda * self.theta + (Date_X.T.
 ↪dot(predict - Date_Y)) / sample_num)            #   theta
            if J_theta < last_Jerr - threshold_value:          #
                last_Jerr = J_theta
                last_better = step
            elif step - last_better > threshold_count:
                break
            if step % 50 == 0:#
                print("step %s: %.6f" % (step, J_theta))
    def predicted(self, X_input):
        sample_num = X_input.shape[0]
        X = np.c_[X_input, np.ones(sample_num, )]
        predict = X.dot(self.theta)
        return predict
def property_label(pd_data):#          X Y
    row_num = pd_data.shape[0]
    column_num = len(pd_data.iloc[0, 0].split())#
```

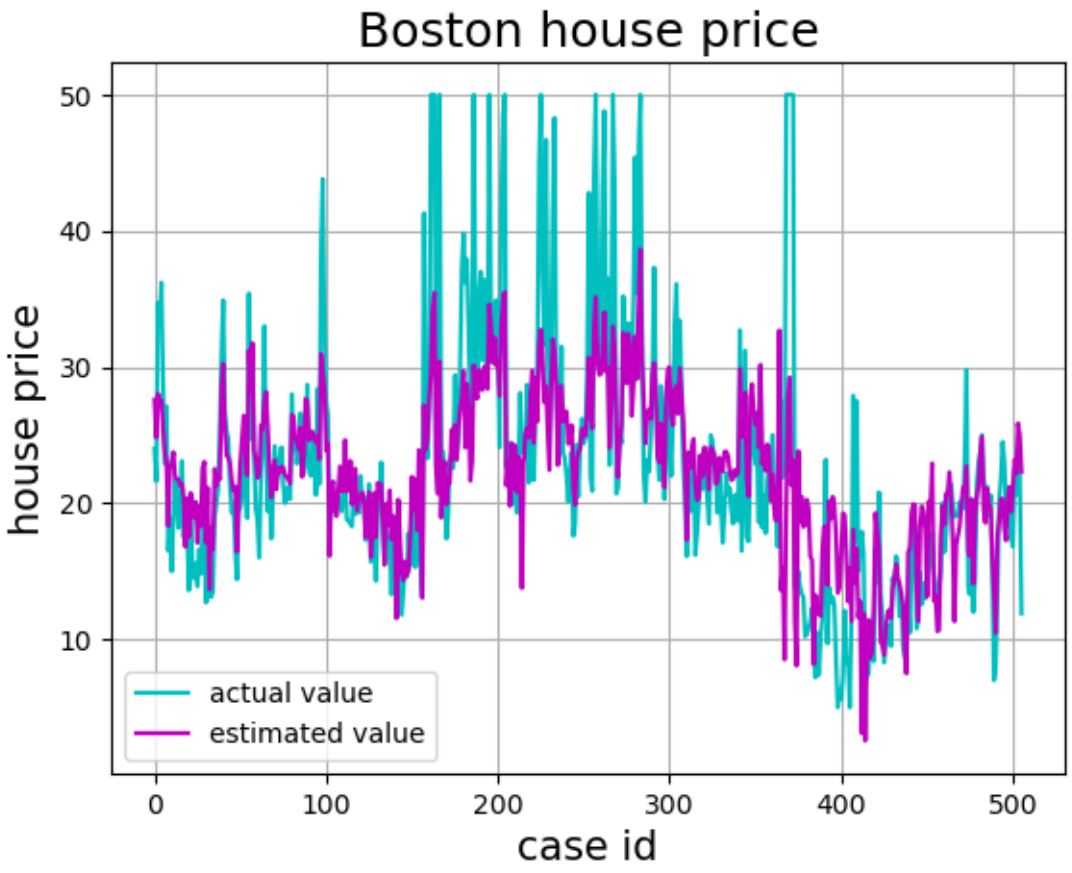
```
    X = np.empty([row_num, column_num - 1])
    Y = np.empty([row_num, 1])
    for i in range(0, row_num):
        row_array = pd_data.iloc[i, 0].split()
        X[i] = np.array(row_array[0:-1])
        Y[i] = np.array(row_array[-1])
    return X, Y
def  standardization (X_input):#
    Maxx = X_input.max(axis=0)
    Minx = X_input.min(axis=0)
    X = (X_input - Minx) / (Maxx - Minx)
    return X, Maxx, Minx
if __name__ == "__main__":
    data = pd.read_csv("housing-data.csv", header=None)
    Date_X, Date_Y = property_label(data)    #    X Y
    Standard_DateX, Maxx, Minx =  standardization (Date_X)    # X
    model = linear_regression()
    model.fitness(Standard_DateX, Date_Y)
    Date_predict = model.predicted(Standard_DateX)
    Date_predict_error = sum((Date_predict - Date_Y) ** 2) / (2 *␣
 ↪Standard_DateX.shape[0])
    print("Test error is %d" % (Date_predict_error))
    print(model.theta)
    t = np.arange(len(Date_predict))
    plt.figure(facecolor='w')
    plt.plot(t, Date_Y, 'c-', lw=1.6, label=u'actual value')
    plt.plot(t, Date_predict, 'm-', lw=1.6, label=u'estimated value')
    plt.legend(loc='best')
    plt.title(u'Boston house price', fontsize=18)
    plt.xlabel(u'case id', fontsize=15)
    plt.ylabel(u'house price', fontsize=15)
    plt.grid()
    plt.show()
```

```
C:\Users\22839\AppData\Local\Temp\ipykernel_18696\581821051.py:27:
DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is
deprecated, and will error in future. Ensure you extract a single element from
your array before performing this operation. (Deprecated NumPy 1.25.)
  print("step %s: %.6f" % (step, J_theta))
```

```
step 0: 296.073458
step 50: 18.249778
step 100: 16.700905
step 150: 16.393434
step 200: 16.318375
step 250: 16.297807
step 300: 16.291614
step 350: 16.289590
```

```
step 400: 16.288881
step 450: 16.288619
step 500: 16.288519
step 550: 16.288479
step 600: 16.288463
step 650: 16.288457
step 700: 16.288454
step 750: 16.288453
step 800: 16.288453
step 850: 16.288452
step 900: 16.288452
Test error is 16
[[-1.87596883]
 [ 3.7838411 ]
 [-1.09119481]
 [ 3.27492537]
 [-0.8944313 ]
 [15.3939236 ]
 [ 1.70332823]
 [-0.26609157]
 [ 1.02396913]
 [-2.15272749]
 [-3.63278677]
 [ 8.04945354]
 [-9.54497195]
 [11.65607057]]

C:\Users\22839\AppData\Local\Temp\ipykernel_18696\581821051.py:56:
DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is
deprecated, and will error in future. Ensure you extract a single element from
your array before performing this operation. (Deprecated NumPy 1.25.)
  print("Test error is %d" % (Date_predict_error))
```


Boston house price
house price
case id
actual value
estimated value
50
40
30
20
10
0
100
200
300
400
500

[ ]: