regression

June 12, 2023

1 Supervised Learning- Linear Regression

[ ]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

1.1 Overview

Linear regression is a machine learning, classification algorithm that is used to predict label for continuous data. The algortihm consists of sketching a line of best fit in the data, and predictions are made on its basis. This line is sketched by the equation

$$y = mx + c$$

where, for a point, y is the y-coordinate of the point, x is the x-coordinate, m is the slope of the line at that point, and c is the y-intercept of the line. Since, y is the dependent variable, and x is the independent variable, our goal effectively becomes to calculate the value of m and c. The cost function for our purposes is

$$f(y) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - (mx_i + c))^2$$

The derivative of this function with respect to m gives us

$$\frac{dy}{dm} = \frac{-2}{n}\sum_{i=1}^{n}(x_i(y_i - (mx_i + c)))$$

The derivative of this function with respect to c gives us

$$\frac{dy}{dc} = \frac{2}{n}\sum_{i=1}^{n}(y_i(mx_i + c))$$

This combined give us the descent gradient, which is used to see when our cost function has reached the lowest value. But for this to happen, we need a number of iterations, and a learning rate, which will be 1000 and 0.001 respectively.

[ ]:

```
class LinearRegression:

    def __init__(self, lr = 0.001, n_iters=1000):
        self.lr = lr
        self.n_iters = n_iters
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0

        for _ in range(self.n_iters):
            y_pred = np.dot(X, self.weights) + self.bias

            dw = (-2/n_samples) * np.dot(X.T, (y_pred-y))
            db = (2/n_samples) * np.sum(y_pred-y)
            try:
                cost = (1/n_samples) * sum([val**2 for val in (y - y_pred)])
            except OverflowError:
                pass

            self.weights = self.weights - self.lr * dw
            self.bias = self.bias - self.lr * db
            print(f"m = {np.sum(self.weights)}\nc = {self.bias}\ncost = {cost}")

    def predict(self, X):
        y_pred = np.dot(X, self.weights) + self.bias
        self.predictions = y_pred
        return y_pred

    def error(self, y_test):
        """Returns the Mean Squared Error between the label and the␣
 ↪predictions"""
        # return np.mean((y_test - self.predictions)**2)
        total = 0
        for i, pred in enumerate(self.predictions):
            total += ((y_test[i] - pred)**2)
        total /= len(self.predictions)
        return total
```

1.2 Medical Price Exercise

[ ]:

```
df = pd.read_csv('./Medical Price Dataset.csv')
print(df.columns)
print(df.describe())
df.head()
```

```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'],
dtype='object')
               age          bmi     children       charges
count  1338.000000  1338.000000  1338.000000   1338.000000
```

```
mean     39.207025    30.663397     1.094918  13270.422265
std      14.049960     6.098187     1.205493  12110.011237
min      18.000000    15.960000     0.000000   1121.873900
25%      27.000000    26.296250     0.000000   4740.287150
50%      39.000000    30.400000     1.000000   9382.033000
75%      51.000000    34.693750     2.000000  16639.912515
max      64.000000    53.130000     5.000000  63770.428010
```

```
[ ]:    age     sex     bmi  children smoker     region      charges
     0   19  female  27.900         0    yes  southwest  16884.92400
     1   18    male  33.770         1     no  southeast   1725.55230
     2   28    male  33.000         3     no  southeast   4449.46200
     3   33    male  22.705         0     no  northwest  21984.47061
     4   32    male  28.880         0     no  northwest   3866.85520
```

1.2.1 Dropping Categorical Columns

```
[ ]: df.drop(['sex', 'smoker', 'region'], axis=1, inplace=True)
```

```
[ ]: df
```

```
[ ]:       age     bmi  children      charges
     0      19  27.900         0  16884.92400
     1      18  33.770         1   1725.55230
     2      28  33.000         3   4449.46200
     3      33  22.705         0  21984.47061
     4      32  28.880         0   3866.85520
     …   …     …       …          …
     1333   50  30.970         3  10600.54830
     1334   18  31.920         0   2205.98080
     1335   18  36.850         0   1629.83350
     1336   21  25.800         0   2007.94500
     1337   61  29.070         0  29141.36030

     [1338 rows x 4 columns]
```

1.2.2 Training and Testing

```
[ ]: y = df['charges']
     X = df.drop(['charges'], axis=1)
```

Splitting into train and test split

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.05,␣
      ↪random_state = 0)
```

```
[ ]: lin_reg = LinearRegression(lr=0.001, n_iters=50)
     lin_reg.fit(X_train, y_train)
lin_reg.predict(X_test);
```

```
m = -1997.6875126271243
c = 26.320657074412278
cost = 317759920.1907936
m = -14461.230176193372
c = 192.5127291953005
cost = 7484885535.309011
m = -92257.89708794827
c = 1231.835824372884
cost = 286781723381.0295
m = -577898.1759061367
c = 7721.681040359379
cost = 11170748070921.045
m = -3609513.7624146147
c = 48236.54227704957
cost = 435309903524658.94
m = -22534454.324109565
c = 301153.5297812838
cost = 1.6963661564805788e+16
m = -140673940.99747428
c = 1879996.9728749662
cost = 6.610598387979186e+17
m = -878163136.922162
c = 11735975.103247227
cost = 2.5760954494254445e+19
m = -5481961031.43693
c = 73262208.88407895
cost = 1.0038830641788327e+21
m = -34221304211.209347
c = 457341526.8020108
cost = 3.912049170284602e+22
m = -213627495706.95203
c = 2854967919.1993256
cost = 1.5244931662678185e+24
m = -1333575900493.967
c = 17822220896.03837
cost = 5.940823626785372e+25
m = -8324886618034.408
c = 111255735961.15067
cost = 2.3150897718337883e+27
m = -51968348531421.25
c = 694517190205.6848
cost = 9.021713129951247e+28
m = -324413937740754.6
c = 4335543900693.5625
cost = 3.515686898597654e+30
m = -2025163507674784.0
```

```
c = 27064759778080.773
cost = 1.3700340712383038e+32
m = -1.264214250896934e+16
c = 168952555578301.34
cost = 5.338909324099663e+33
m = -7.89189448710608e+16
c = 1054691276423221.8
cost = 2.0805287524853352e+35
m = -4.926538247091136e+17
c = 6583941182515396.0
cost = 8.107648261376211e+36
m = -3.0754059294261683e+18
c = 4.110044566010741e+16
cost = 3.159483388618058e+38
m = -1.9198311585897378e+19
c = 2.565707357692498e+17
cost = 1.23122451309438e+40
m = -1.1984602234865721e+20
c = 1.6016503323969595e+18
cost = 4.797979970730433e+41
m = -7.481423045214878e+20
c = 9.998349108584278e+18
cost = 1.8697330628736235e+43
m = -4.670300247315581e+21
c = 6.241498713862333e+19
cost = 7.286194914795733e+44
m = -2.9154486076050426e+22
c = 3.896273852020077e+20
cost = 2.8393698218505156e+46
m = -1.8199773319652754e+23
c = 2.432260363399349e+21
cost = 1.1064789069620272e+48
m = -1.1361261797677225e+24
c = 1.5183456553743931e+22
cost = 4.3118566737247456e+49
m = -7.092300951681455e+24
c = 9.478317222471544e+22
cost = 1.6802948395818518e+51
m = -4.427389640779642e+25
c = 5.916867285904563e+23
cost = 6.547969845867949e+52
m = -2.763810950046567e+26
c = 3.6936217323478295e+24
cost = 2.5516896256769846e+54
m = -1.7253170801231262e+27
c = 2.305754184173237e+25
cost = 9.943723167717928e+55
m = -1.077034240317499e+28
```

```
c = 1.4393738024854496e+26
cost = 3.8749865752178704e+57
m = -6.723417788998583e+28
c = 8.985333117911248e+26
cost = 1.5100501798829468e+59
m = -4.197113246102259e+29
c = 5.609120514797533e+27
cost = 5.8845405048566454e+60
m = -2.620060236243451e+30
c = 3.5015099091658735e+28
cost = 2.2931567052945655e+62
m = -1.6355802760192744e+31
c = 2.1858278159012535e+29
cost = 8.936241785909019e+63
m = -1.0210157775375358e+32
c = 1.3645094158553494e+30
cost = 3.4823794236063323e+65
m = -6.3737208944447676e+32
c = 8.517989991769872e+30
cost = 1.357054424050943e+67
m = -3.9788139354965363e+33
c = 5.317380199564949e+31
cost = 5.288328713845579e+68
m = -2.483786252250149e+34
c = 3.3193901629427104e+32
cost = 2.0608179075236344e+70
m = -1.5505108423967949e+35
c = 2.0721390309352563e+33
cost = 8.030836730801069e+71
m = -9.679109344502072e+35
c = 1.293538858872436e+34
cost = 3.1295505712235714e+73
m = -6.042212356155332e+36
c = 8.074954211242235e+34
cost = 1.2195599420272169e+75
m = -3.771868759558301e+37
c = 5.040813816022276e+35
cost = 4.7525241032162665e+76
m = -2.3546001200766342e+38
c = 3.146742788017876e+36
cost = 1.8520192877200553e+78
m = -1.4698660210314793e+39
c = 1.9643634014946032e+37
cost = 7.217165799887e+79
m = -9.175681685230679e+39
c = 1.2262596065444698e+38
cost = 2.8124697473956194e+81
m = -5.72794616543317e+40
```

```
c = 7.654961508132784e+38
cost = 1.0959961707044952e+83
m = -3.575687169587739e+41
c = 4.7786321410456997e+39
cost = 4.271006318596842e+84
m = -2.2321331879674685e+42
c = 2.983075109544875e+40
cost = 1.6643757944673006e+86
```

```
[ ]: lin_reg.predictions
```

```
[ ]: array([-9.40310049e+43, -8.69246187e+43, -9.86680679e+43, -1.13027348e+44,
           -8.13965987e+43, -6.29696634e+43, -4.73158343e+43, -9.86003942e+43,
           -7.63761730e+43, -6.83089708e+43, -6.08522863e+43, -8.78178725e+43,
           -8.23534393e+43, -6.44414309e+43, -6.07519905e+43, -8.96605593e+43,
           -9.86438119e+43, -6.32407935e+43, -7.54570961e+43, -5.39255976e+43,
           -8.77548434e+43, -1.02506667e+44, -9.46279544e+43, -8.54551076e+43,
           -5.93365339e+43, -7.53593403e+43, -5.15601505e+43, -8.15118922e+43,
           -6.32427154e+43, -8.54741414e+43, -7.74909239e+43, -1.13102226e+44,
           -1.13379476e+44, -1.04421317e+44, -4.95079071e+43, -6.40194431e+43,
           -9.39008597e+43, -6.94214048e+43, -8.12691963e+43, -5.63652338e+43,
           -6.40844143e+43, -6.27623632e+43, -7.39990593e+43, -1.07964250e+44,
           -5.71443497e+43, -5.71187900e+43, -8.93526894e+43, -7.77424426e+43,
           -5.84685861e+43, -9.45601692e+43, -6.16245118e+43, -5.01872943e+43,
           -8.69129012e+43, -9.92207989e+43, -1.12521395e+44, -6.84883338e+43,
           -6.59415210e+43, -8.79295618e+43, -7.83983035e+43, -9.86765965e+43,
           -5.08073038e+43, -1.07390401e+44, -1.06942702e+44, -9.34901434e+43,
           -9.41249170e+43, -1.01440931e+44, -5.22533597e+43])
```

```
[ ]: y_test
```

```
[ ]: 578      9724.53000
     610      8547.69130
     569     45702.02235
     1034    12950.07120
     198      9644.25250
                 …
     435     13919.82290
     1144     9630.39700
     390     10736.87075
     483      9880.06800
     503     32548.34050
     Name: charges, Length: 67, dtype: float64
```

```
[ ]:
```