ML_Lab_6_Linear_Regression_with_gradient_descent

November 9, 2021

1 Gradient Descent with Linear Regression

1.0.1 Gradient descent is a name for a generic class of computer algorithms which minimize a function.

1.0.2 These algorithms achieve this end by starting with initial parameter values and iteratively moving towards a set of parameter values that minimize some cost function or metric—that's the descent part.

1.0.3 The movement toward best-fit is achieved by taking the derivative of the variable or variables involved, towards the direction with the lowest (calculus-defined) gradient—that's the gradient part.

[1]:
```
%%html
<img src='image/gdalgo.jpg', width=900, height=600>
```

```
<IPython.core.display.HTML object>
```

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

class GradientDescentLinearRegression:
    def __init__(self, learning_rate=0.01, iterations=200):
        self.learning_rate, self.iterations = learning_rate, iterations

    def fit(self, X, y):
        b0 = 0
        b1 = 0
        n = X.shape[0]
        for _ in range(self.iterations):
            b0_deri = np.sum(b1*X + b0 - y)/n
            b1_deri = np.sum(X*((b1*X + b0) - y))/n
            b0 = b0 - (self.learning_rate*b0_deri)
            b1 = b1 - (self.learning_rate*b1_deri)
            plt.plot(X,(b0+b1*X))
        self.b0, self.b1 = b0, b1
```

```
    def predict(self, X):
        return self.b0 + self.b1*X

    def rmse(self, X):
        rmse = 0
        n = X.shape[0]
        for i in range(n):
            y_pred = self.predict(X[i])
            rmse += (y_pred - y[i])**2
        rmse = np.sqrt(rmse/n)
        return rmse
```

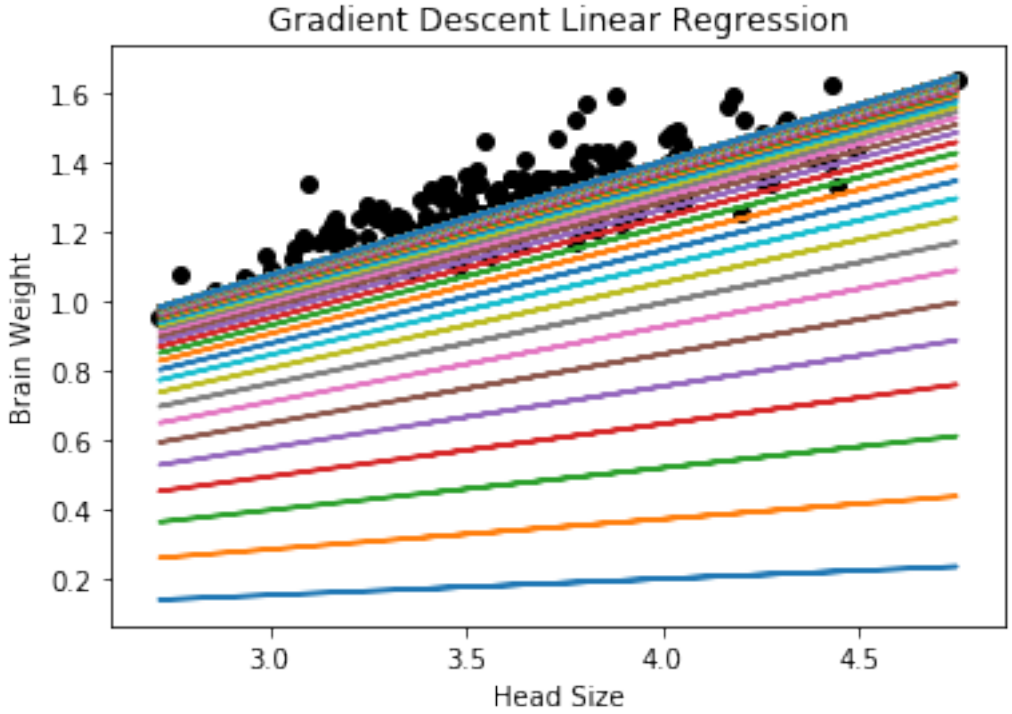
[3]:
```
data = pd.read_csv('image/headbrain.csv')

# Collecting X and y
X = data['Head Size(cm^3)'].values/1000
y = data['Brain Weight(grams)'].values/1000
print(X)
print(y)
```

```
[4.512 3.738 4.261 3.777 4.177 3.585 3.785 3.559 3.613 3.982 3.443 3.993
 3.64  4.208 3.832 3.876 3.497 3.466 3.095 4.424 3.878 4.046 3.804 3.71
 4.747 4.423 4.036 4.022 3.454 4.175 3.787 3.796 4.103 4.161 4.158 3.814
 3.527 3.748 3.334 3.492 3.962 3.505 4.315 3.804 3.863 4.034 4.308 3.165
 3.641 3.644 3.891 3.793 4.27  4.063 4.012 3.458 3.89  4.166 3.935 3.669
 3.866 3.393 4.442 4.253 3.727 3.329 3.415 3.372 4.43  4.381 4.008 3.858
 4.121 4.057 3.824 3.394 3.558 3.362 3.93  3.835 3.83  3.856 3.249 3.577
 3.933 3.85  3.309 3.406 3.506 3.907 4.16  3.318 3.662 3.899 3.7   3.779
 3.473 3.49  3.654 3.478 3.495 3.834 3.876 3.661 3.618 3.648 4.032 3.399
 3.916 4.43  3.695 3.524 3.571 3.594 3.383 3.499 3.589 3.9   4.114 3.937
 3.399 4.2   4.488 3.614 4.051 3.782 3.391 3.124 4.053 3.582 3.666 3.532
 4.046 3.667 2.857 3.436 3.791 3.302 3.104 3.171 3.572 3.53  3.175 3.438
 3.903 3.899 3.401 3.267 3.451 3.09  3.413 3.323 3.68  3.439 3.853 3.156
 3.279 3.707 4.006 3.269 3.071 3.779 3.548 3.292 3.497 3.082 3.248 3.358
 3.803 3.566 3.145 3.503 3.571 3.724 3.615 3.203 3.609 3.561 3.979 3.533
 3.689 3.158 4.005 3.181 3.479 3.642 3.632 3.069 3.394 3.703 3.165 3.354
 3.    3.687 3.556 2.773 3.058 3.344 3.493 3.297 3.36  3.228 3.277 3.851
 3.067 3.692 3.402 3.995 3.318 2.72  2.937 3.58  2.939 2.989 3.586 3.156
 3.246 3.17  3.268 3.389 3.381 2.864 3.74  3.479 3.647 3.716 3.284 4.204
 3.735 3.218 3.685 3.704 3.214 3.394 3.233 3.352 3.391]
[1.53  1.297 1.335 1.282 1.59  1.3   1.4   1.255 1.355 1.375 1.34  1.38
 1.355 1.522 1.208 1.405 1.358 1.292 1.34  1.4   1.357 1.287 1.275 1.27
 1.635 1.505 1.49  1.485 1.31  1.42  1.318 1.432 1.364 1.405 1.432 1.207
 1.375 1.35  1.236 1.25  1.35  1.32  1.525 1.57  1.34  1.422 1.506 1.215
 1.311 1.3   1.224 1.35  1.335 1.39  1.4   1.225 1.31  1.56  1.33  1.222
 1.415 1.175 1.33  1.485 1.47  1.135 1.31  1.154 1.51  1.415 1.468 1.39
 1.38  1.432 1.24  1.195 1.225 1.188 1.252 1.315 1.245 1.43  1.279 1.245
```

```
1.309 1.412 1.12  1.22  1.28  1.44  1.37  1.192 1.23  1.346 1.29  1.165
1.24  1.132 1.242 1.27  1.218 1.43  1.588 1.32  1.29  1.26  1.425 1.226
1.36  1.62  1.31  1.25  1.295 1.29  1.29  1.275 1.25  1.27  1.362 1.3
1.173 1.256 1.44  1.18  1.306 1.35  1.125 1.165 1.312 1.3   1.27  1.335
1.45  1.31  1.027 1.235 1.26  1.165 1.08  1.127 1.27  1.252 1.2   1.29
1.334 1.38  1.14  1.243 1.34  1.168 1.322 1.249 1.321 1.192 1.373 1.17
1.265 1.235 1.302 1.241 1.078 1.52  1.46  1.075 1.28  1.18  1.25  1.19
1.374 1.306 1.202 1.24  1.316 1.28  1.35  1.18  1.21  1.127 1.324 1.21
1.29  1.1   1.28  1.175 1.16  1.205 1.163 1.022 1.243 1.35  1.237 1.204
1.09  1.355 1.25  1.076 1.12  1.22  1.24  1.22  1.095 1.235 1.105 1.405
1.15  1.305 1.22  1.296 1.175 0.955 1.07  1.32  1.06  1.13  1.25  1.225
1.18  1.178 1.142 1.13  1.185 1.012 1.28  1.103 1.408 1.3   1.246 1.38
1.35  1.06  1.35  1.22  1.11  1.215 1.104 1.17  1.12 ]
```

[4]:
```
clf = GradientDescentLinearRegression()
clf.fit(X,y)
plt.scatter(X,y,color='black')
plt.plot(X, clf.predict(X))
plt.xlabel('Head Size')
plt.ylabel('Brain Weight')
plt.title("Gradient Descent Linear Regression")
plt.show()
#compute rmse
rmse = clf.rmse(X)
print("RMSE",rmse)
```



```
RMSE 0.07574196359651633
```

[ ]:

[ ]: