Linear Regression Algorithm with formulas and code

Prediction formula: $\boldsymbol{y = wx + b}$

Here w is the vector of weights: $w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \dots \\ w_n \end{bmatrix}$ and x is the vector of the featues $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{bmatrix}$, b is the bias value. Weights are also slopes, bias is also called "intersection".

Values of w and be must be optimized continuously until the best fit line is obtained. How can we check whether the lines is "best fit" or not? Cost function will be used for this purpose.

Cost function formula: $J(w,b) = \frac{1}{N}\sum_{i=1}^{n}\left(y^i_{actual} - y^i_{pred}\right)^2$ or

$$J(w,b) = \frac{1}{N}\sum_{i=1}^{n}\left(y^i_{actual} - (wx_i + b)\right)^2$$

Here y^i_{actual} and y^i_{pred} are actual and predicted values of i-th sampe, respectively.

We need to minimize the cost function as much as possible. In order to minimize the cost function the values of weights and intercept must be optimizited. For this optimization purpose, the gradient descent algorithm is used. According to the gradient descent algorithm the derivates of the cost function with respect to the slopes and intercepts must be obtained, and that obtained derivatives will be used for optimization purpose.

Derivatives of cost function with respect to weights (w) and intercept (b):

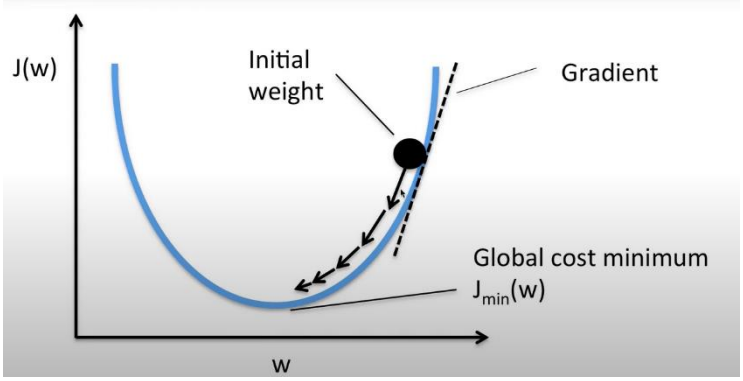
$$J'(w) = \frac{1}{2N} * 2\sum_{i=1}^{N}\left(y^i_{actual} - (wx_i + b)\right)(-x_i)$$

$$J'(b) = \frac{1}{2N} * 2\sum_{i=1}^{N}\left(y^i_{actual} - (wx_i + b)\right)(-1)$$

Derivation results:

$$J'(w) = -\frac{1}{N}\sum_{i=1}^{N} x_i\left(y^i_{actual} - (wx_i + b)\right) \quad Eq\ 1$$

$$J'(w) = -\frac{1}{N}\sum_{i=1}^{N}\left(y^i_{actual} - (wx_i + b)\right) \quad Eq\ 2$$

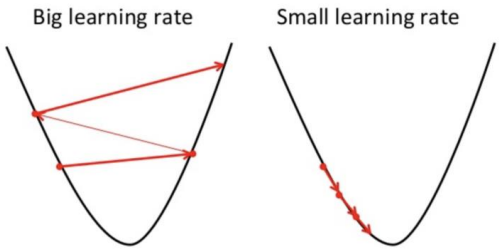


Optimization formulas (Update rules):

$$w = w - \alpha\, J'(w)$$

$$b = b - \alpha\, J'(b)$$

The term α here is called the "learning rate" and defines how fast the weights will reach to the global maxima. If it is too small, the optimized results will be reached late, if it is too big, then the results will change rapidly and global maxima will be obtained too late.



Coding part explanation

LinearRegression class will have 3 functions:

1. Initialization function (_init_): that will initialize the parameters of linear regression: learning_rate (default small value), number_of_iterations(default large value), weights (none) and bias (none).
2. "Fit" function: which will take two parameters: {X_train, y_train}. The main process will occur here. The two parameters weight and bias will be optimized here to fit the best regression line to the given training sets.
3. "Predict" function: that will take only the test set {X_test} and will return the predictions.

Let’s create these functions in Pycharm:

```
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
```

Figure 1. Initalization function

```
def fit(self, X, y):
    # initialization
    n_samples, n_features = X.shape
    self.weights = np.zeros(n_features)  # initial values for the weights
    self.bias = 0

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

        dw = (1 / n_samples) * np.dot(X.T, (y_pred - y))
        db = (1 / n_samples) * np.sum(y_pred - y)

        self.weights = self.weights - self.lr * dw
        self.bias = self.bias - self.lr * db
```

Figure 2. Fitting the best line

Here fit function will take two training sets **{X_train and y_train**}.

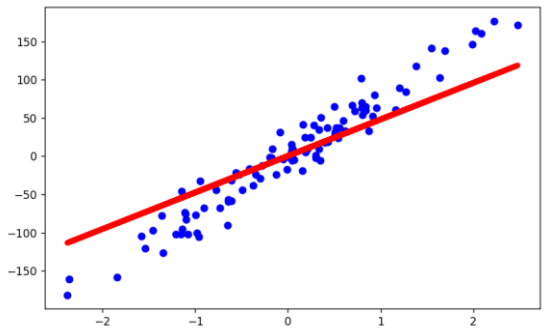
Then zeros are assigned to the weights and bias as the initial values.

After that, the iterations of gradient descent algorithm starts. In each iteration, the new predictions will be obtained. These predictions will be used in the Eq1 and Eq 2. Dw and db are the derivatives of the cost function with respect to weights and bias, respectively.

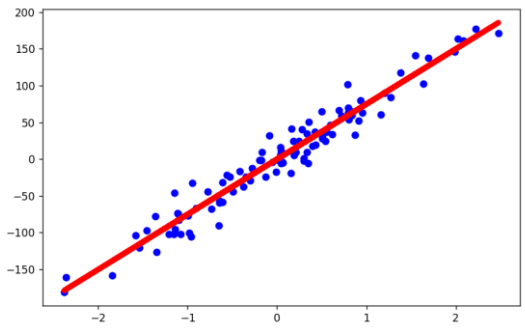
```
def predict(self, X):
    y_pred = np.dot(X, self.weights) + self.bias
    return y_pred
```

Predict function will return the prediction results for test dataset **{X_test}.**

If we test the code we will get the following results for different learning rates:



Predictions for lr = 0.001



Predictions for lr = 0.1

```
MSE if lr = 0.001: 783.8155465
```

```
MSE ifS lr = 0.1: 305.774131
```