Report

Pre-Processing:

Given 1338 tuples of data containing three feature attributes and one target attribute. Before the model was trained, the dataset was pre-processed as follows.

1. Shuffling the data:

The dataset was randomly shuffled to ensure the split into train and test data remains random and the model is general.

2. Splitting the data:

The dataset was split into training and testing data in 70:30 ratio in order.

3. Standardising the data:

The feature attributes of the training and testing datasets were standardised separately to prevent leakage of testing data into training data. Finally, the Panda Series were converted into Numpy Arrays for easier and faster computation.

Model:

The models were generated by solving Normal Equations, by Gradient Descent and by Stochastic Gradient Descent algorithms after each random shuffle and split of the dataset. The intercept of the models was close to 13000 as expected, which is the mean of the target attribute. Error Sum of Squares (SSE) was used as the key performance indicator of the of the predictive models. All the three optimization algorithms yielded similar linear regression models.

Hypothesis - $y = w_0 + w_1x_1 + w_2x_2 + w_3x_3$

x_1, x_2, x_3 represent the age, bmi and number of children respectively. w_1, w_2, w_3 are weights associated with x_1, x_2, x_3.

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} \\ 1 & x_{21} & x_{22} & x_{23} \\ . & . & . & . \\ . & . & . & . \\ 1 & x_{m1} & x_{m2} & x_{m3} \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_m \end{bmatrix}$$

where m = size of training data

$$\omega = \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

$$b = X^T.Y$$

Matrix dot product and inversion were performed with Numpy library by first converting Pandas Series to Numpy Array.

Algorithms:

Linear Regression by Solving Normal Equations

The parial derivatives of error function with respect to each weight was equated to zero. The best fit line was derived by solving for these equations for the weights by means of linear algebra.The model with the least error by solving normal eqations had the following weights:
ω_0 **= 13072.157252916671,** ω_1 **= 3173.6711350795026,** ω_2 **= 2172.652465579652 and** ω_3 **= 569.7149647829754**

$$\omega = (X^T.X)^{-1}x^T.Y = (X^T.X)^{-1}.b$$

Linear Regression by Gradient Descent

Gradient Descent algorithm was implemented with SSE as the error function on the hypothesis. **np.random.randn()** function was used to initialize the weights to random values. The learning rate used for the same was 10^{-7}, and the model converged to a error sum of squares of 6.086349e+10 after approximately 140,000 iterations, the weights obtained corresponding to the least training error were: ω_0 **= 13072.142207938474,** ω_1 **= 3173.667317195016,** ω_2 **= 2172.653751292538 and** ω_3 **= 569.7173903721999**

$$SumofSquaresofErrors = E(\omega) = \frac{1}{2} * \sum_{n=0}^{N}(x_n * \omega - y_n)^2 \tag{1}$$

$$\frac{\partial E(\omega)}{\partial \omega} = (X.\omega - Y).X \tag{2}$$

$$Gradient$$

$$\omega = \omega - \eta * \frac{\partial E(\omega)}{\partial \omega}$$

where η is the learning rate

Linear Regression by Stochastic Gradient Descent

SGD made sequential passes over the training data, and during each pass, updated feature weights one example at a time with the aim of approaching the optimal weights that minimize the loss. **np.random.randn()** function was used to initialize the weights to random values. The model converged to a minimum training error of 5.868913e+10 after around 850,000 iterations with the corresponding weights as follows:
ω_0 **= 13071.179444392286,** ω_1 **= 3174.536142389335,** ω_2 **= 2174.1187630838886,** ω_3 **= 570.4156365220362**

$$\frac{\partial E(\omega)}{\partial \omega}_{\omega=\omega_n} = (x_n.\omega - y_n).x_n \tag{3}$$

$$Gradient$$

$$\omega = \omega - \eta * \frac{\partial E(\omega)}{\partial \omega}_{\omega=\omega_n}$$

where η is the learning rate

Mean, Variance and Minimum of Training Error

Normal Equations

Mean of training errors obtained over 20 regression models = 60863489090.22312
Variance of training errors obtained over 20 regression models = 2.3592510523138785e+18
Minimum training errors obtained over 20 regression models = 58689130587.04303

Gradient Descent

Mean of training errors obtained over 20 regression models = 60863489090.34578
Variance of training errors obtained over 20 regression models = 2.359251052323336e+18
Minimum of training errors obtained over 20 regression models = 58689130587.1583

Stochastic Gradient Descent

Mean of training errors obtained over 20 regression models = 60863507490.357376
Variance of training errors obtained over 20 regression models = 2.3592502939019807e+18
Minimum training errors obtained over 20 regression models = 58689132776.446106

Mean, Variance and Mean of Testing Error

Normal Equations

Mean of testing error obtained over 20 regression models = 25548673963.726776
Variance of training error obtained over 20 regression models = 2.2762131308133821e+18
Minimum testing error obtained over 20 regression models = 22987187804.050194

Gradient Descent

Mean of testing error obtained over 20 regression models = 25548672100.185345
Variance of testing error obtained over 20 regression models = 2.2762222241018094e+18
Minimum testing error obtained over 20 regression models = 22987179664.889412

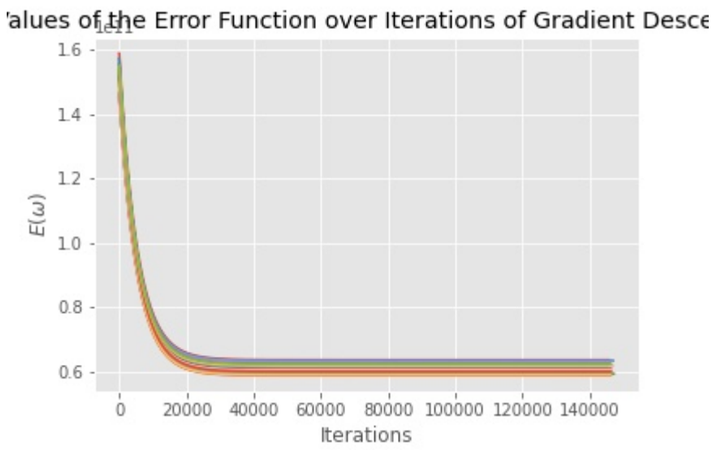
Stochastic Gradient Descent

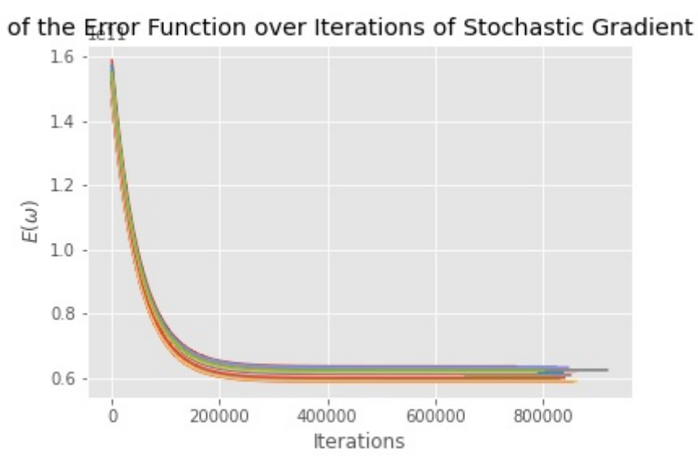
Mean of testing error obtained over 20 regression models = 25548244755.53107
Variance of training error obtained over 20 regression models = 2.279184996206161e+18
Minimum testing error obtained over 20 regression models = 22982193889.584408

The large error values are obtained due to the choice of the model's key performance indicator, namely, sum of squares fo errors.

Plot the Convergence: Error vs Epochs

Plot of $E(\omega)$ against the number of iterations of Gradient Descent





Plot of $E(\omega)$ against the number of iterations of Stochastic Gradient Descent

---

Additional Questions

All three methods yield very close models. Normal Equations and Gradient Descent methods differ only in decimals while Stochastic Gradient Descent differs by a small value. The result is because the error function has a single global minima, and all three algorithms aim to converge to it. Since the minima is unique, the models converge to the same weights. However, the small discrepancy is because the gradient and stochastic gradient descent algorithms are stopped prematurely when the difference in gradient calculated is too small to result in a significant change in the model. The most efficient algorithm to work with would be Stochastic Gradient Descent, which reaches the minima faster than other algorithms when the dataset is huge.

Standardization is a scaling technique where the values are centred around the mean, i.e. mean of the distribution becomes zero with a unit standard deviation. In multivariate regression, standardization brings variables having different units to comparable units, better ensuring that all the weights are updated at similar rates and a more accurate predictive model.

Stochastic gradient descent plots converged to the minima significantly faster than gradient descent algorithm. Approximately, it took SGD 8000 epochs, while it took GD 140,000 epochs to achieve the same. However, the GD algorithm yielded a more accurate model. In real-world data, SGD is a more practical way of implementing linear regression. In general, increasing the number of training iterations tends to minimize the loss function and resulting in a more accurate model. However, after a large number of epochs, the change in error function becomes negligible, and the computational complexity of running more iterations would outweigh the resulting predictive accuracy of the model

If a large learning rate is used in the GD/SGD algorithm, the error value will overshoot the minima of the loss function. It may keep diverging from the minima indefinitely, thereby resulting in an infinite loop.

If the model does not have a bias term, it'll be equal to zero when all the variables are zero. However, the mean of the 'charges' variable is equal to 13000. Therefore such a model would not fit the data well and result in larger error values. The minima would have been larger without a bias term.

The final vector value signifies the linear model that best fits the training data, given the initial hypothesis. Since the feature attributes are standardized, they update at similar rates. Noticeably, ω_1 has the largest value among all the weights, excluding bias value, meaning the model is susceptible to bigger change due to ω_1 given the same difference in all feature attributes. Therefore, 'age' has the greatest influence on the target attribute. Similarly, it can be deduced that 'children' has the least impact on the target attribute.