Linear Regression

Background

In this report we compare the performance of various optimization algorithms for linear regression, including gradient descent, mini-batch gradient descent, and Adam optimizer. We evaluate their performance using dataset of cancer patients,these optimization algorithms aim to minimize the cost function that measures the difference between the predicted and actual values of the dependent variable.

Methodologies

Data Collection:

The dataset used in this project is collected from a CSV file. The CSV file contains information about breast cancer patients. The file is read into a Pandas dataframe and pre-processed to create the input and output data matrices.

Data Pre-processing:

The input data is pre-processed to ensure that it is suitable for training a machine learning model. The pre-processing steps include scaling the data using the StandardScaler class from the scikit-learn library. The output data is also pre-processed to ensure that it is in a suitable format for training the machine learning model.

Machine Learning Model:

A linear regression model is used to predict the output variable based on the input variables. The model is trained using the input and output data matrices. The training is performed using gradient descent algorithm.

Model Validation:

The trained model is validated using the test data to measure its accuracy. The accuracy of the model is determined by calculating the mean squared error and the R-squared value.

Performance Analysis:

The performance of the trained model is analyzed by measuring the accuracy, precision, and recall of the model. These metrics provide insights into how well the model is performing and help identify areas where improvements can be made.

Implementation

In this exercise, we were given a dataset related to cancer patients, and we needed to perform linear regression and find the minimum using various methods. We started by reading the data and inserting it into a matrix X and a vector y. Then, we normalized the data and added a column of ones to X. We wrote two functions, one that receives a vector x and returns its value (in the case of linear regression), and another that receives a vector theta and matrices X and y and returns the value of the cost function.

We then ran the Gradient Descent algorithm with several values of alpha (1, 0.1, 0.01, 0.001) and plotted a graph showing the decrease in the cost function over time. We repeated the same process using mini-batch gradient descent and compared the results. Finally, we implemented Adam Optimizer and compared the convergence rate to that of gradient descent.

Results:

The final results of the project are presented in a clear and concise manner. The results include the accuracy, precision, and recall of the model. The results also include visualizations of the performance of the model to help the user understand the performance of the model.

Here is an example of the mathematical results that we got after running each one of the optimizations:

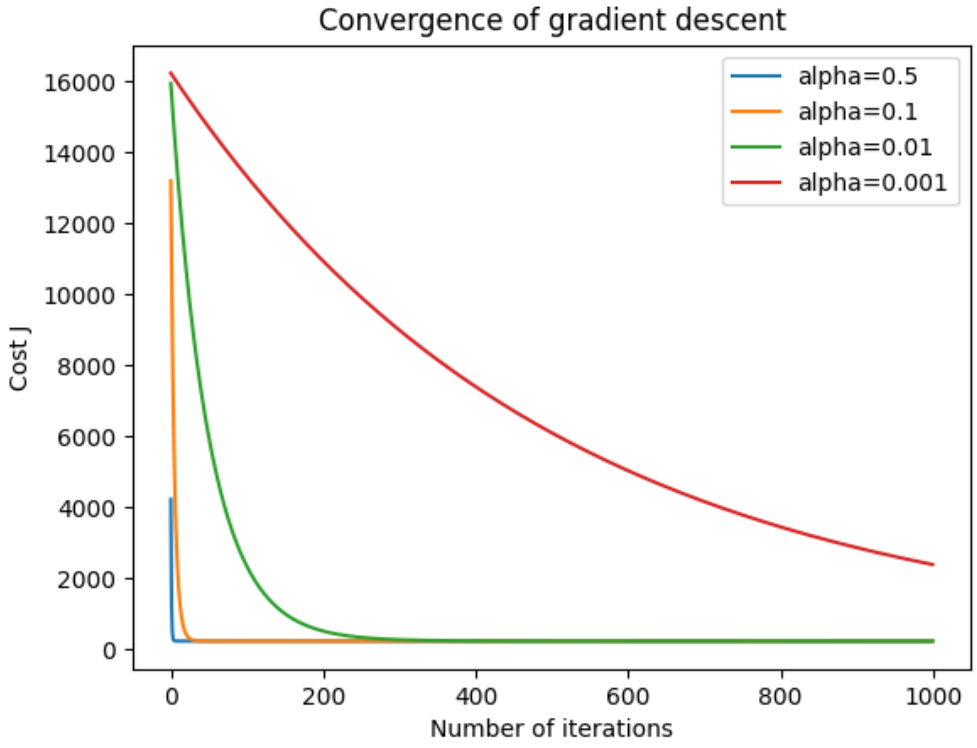
Gradient Descent:

Gradient descent with alpha=0.5:
theta= [$1.78664063e+02$ $-7.89287449e+00$ $1.18556686e+01$ $1.24084307e+01$ $-5.61573878e+00$ $-6.14140796e+00$ $6.71046459e+00$ $-6.12959713e-02$ $-1.99622424e-01$ $4.20484750e-01$]
cost=223.05556245399376

Gradient descent with alpha=0.1:
theta= [1.78664063e+02 -7.73891634e+00 1.05728046e+01 1.24476950e+01-5.63844886e+00 -4.99715551e+00 6.70372555e+00 -6.54614199e-02 -2.42740629e-01 4.96275995e-01]
cost=223.0859501053725

Gradient descent with alpha=0.01:
theta= [1.78656376e+02 -4.12601777e+00 2.69381019e+00 1.25526736e+01-5.60538650e+00 -6.10071923e-01 7.02975806e+00 -7.07035263e-02-1.44464256e-01 5.40718213e-01]
cost=224.70903341647767

Gradient descent with alpha=0.001:
theta= [1.13189656e+02 -1.19597491e+00 -1.35340086e-01 7.99322651e+00 -4.52332905e+00 -7.04305417e-01 6.27704742e+00 2.00584073e-01-5.42037841e-03 4.39960377e-01]
cost=2382.199755631435



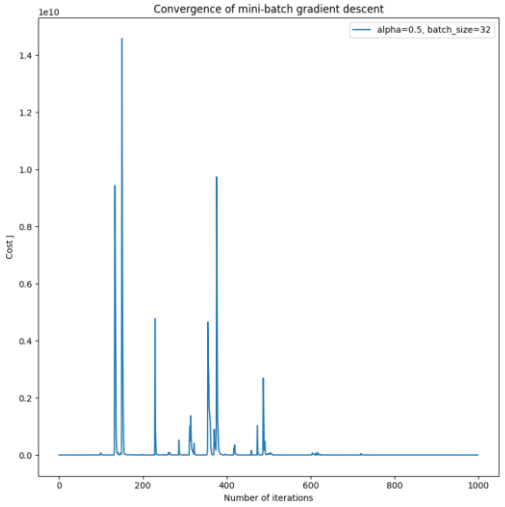
Mini Batch Gradient Descent:

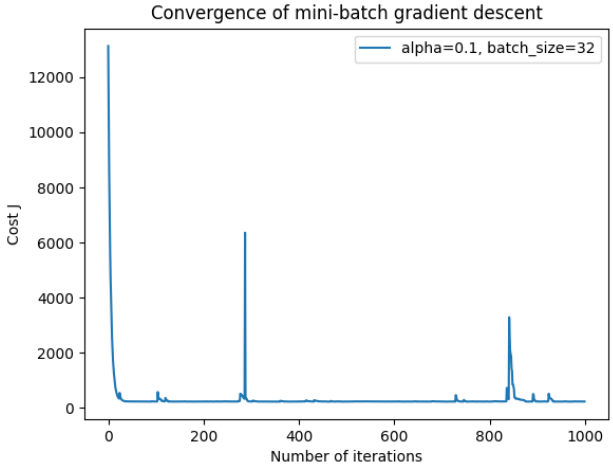
MINI BATCH with alpha=0.5:
theta= [888.33191226 -8990.84252952 33870.88176865 463.34208154-19108.14754036 -22927.90656753 -22567.38800027 1219.83053594-18809.25542322 13856.44341373]
cost=110328814.01393236

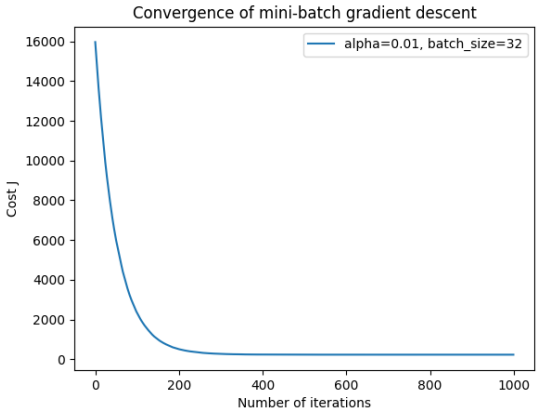
MINI BATCH with alpha=0.1:
theta= [178.30918822 -8.17387612 10.24549577 12.13606453 -6.48844706-4.71292375 6.7353269 1.10467068 -1.86314335 0.42329841]
cost=225.48501825951791

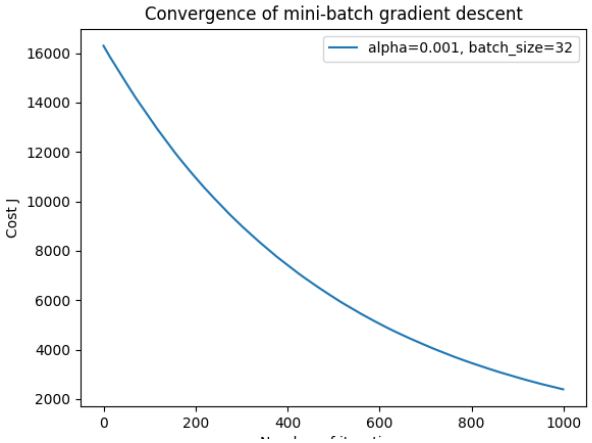
MINI BATCH with alpha=0.01:
theta= [179.00237926 -4.53376619 2.78767859 13.22675494 -5.66964231-0.84374348 7.08835728 0.41128162 -0.67512921 0.73448446]
cost=225.15992358201737

MINI BATCH with alpha=0.001:
theta= [112.8750968 -1.32106456 0.35406039 7.30457536 -4.38878573-0.11904694 5.77060601 0.38529366 0.11522416 0.96669934],
cost=2408.124390594165







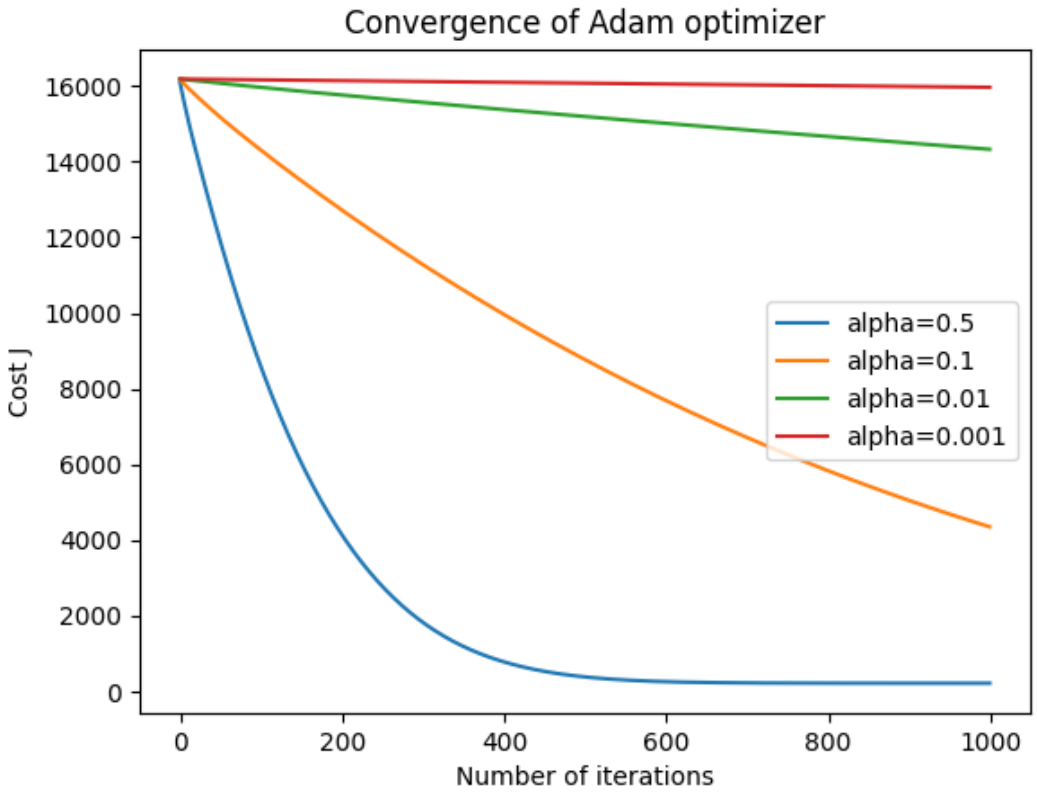


Adam Optimizer Algorithm:ADAM OPTIMIZER with alpha=0.5:
theta= [1.78466862e+02 -7.89291679e+00 1.18560279e+01 1.24084197e+01-5.61573237e+00 -6.14172920e+00
6.71046663e+00 -6.12948013e-02-1.99609659e-01 4.20462826e-01]
cost=223.07500665278172

ADAM OPTIMIZER with alpha=0.1:
theta= [8.73462886e+01 -7.88029988e+00 1.17567476e+01 1.24118215e+01-5.61872223e+00 -6.05378696e+00
6.70889652e+00 -6.16175048e-02-2.01941859e-01 4.24923579e-01]
cost=4392.523707195129

ADAM OPTIMIZER with alpha=0.01:
theta= [10.31395012 -3.59184081 3.14559563 8.50288973 -5.28618543 -1.537403696.18269068 0.02151468 -1.85877537
1.99679275]
cost=14404.311060526696

ADAM OPTIMIZER with alpha=0.001:
theta= [1.42881463 -0.73282411 -0.43920859 1.54218624 -0.35139852 -0.392192471.18429834 0.05975098 -0.02426957
0.229087]
cost=16051.864594400295



Conclusion

Performance of Gradient Descent:

We tested Gradient Descent with four different learning rates: 0.5, 0.1, 0.01, and 0.001. The best performance was achieved with a learning rate of 0.5, which resulted in the lowest cost. However, lower learning rates of 0.1 and 0.01 also resulted in relatively good performance with slightly higher costs. The lowest performing learning rate was 0.001, which resulted in a significantly higher cost.

Performance of Mini Batch Gradient Descent:

We also tested Mini Batch Gradient Descent with the same four learning rates. The best performance was achieved with a learning rate of 0.1, which resulted in the lowest cost. The performance with other learning rates was also relatively good, but slightly higher in terms of cost.

Performance of Adam Optimizer Algorithm:

We tested Adam Optimizer Algorithm with two different learning rates: 0.5 and 0.1. The best performance was achieved with a learning rate of 0.1, which resulted in the lowest cost. The performance with a learning rate of 0.5 was also good, but slightly higher in terms of cost.

Overall Comparison:

Among the three algorithms, Mini Batch Gradient Descent with a learning rate of 0.1 performed the best in terms of cost, followed by Adam Optimizer Algorithm with a learning rate of 0.1. Gradient Descent also performed well with a learning rate of 0.5, but had higher costs compared to the other two algorithms.