Parallel Implementation of Machine Learning Algorithms

Team Members
Saurabh Parekh (170100016)
Nabajyoti Majumdar (170100081)
Sai Vamseedhar Bojja (170070048)

Linear Regression

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data.

One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.

For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.

Logistic Regression

Logistic regression is estimating the parameters of a logistic model (a form of binary regression).

Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1".

Logistic regression may be used to predict the risk of developing a given disease.

Pseudo code: Gradient Descent algorithm

Algorithm: Gradient Descent

Input: $Y, \Theta, \mathbf{X}, \alpha$, tolerance, max iterations
Output: $\mathbf{\Theta}$

```
for i = 0; i < max iterations; i++ do
    current cost = Cost(Y, X, Θ)
    if current cost < tolerance then
        break
    else
        gradient = Gradient(Y, X, Θ)
        θ_j ← θ_j − α· gradient
```

Source - https://miro.medium.com/max/2752/1*oJKalifbWzwuo3fRjWJjTg.png

Objective function: Linear regression

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Computation of gradient descent becomes as follows,

Repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{ij}$$, where x_{ij} is the j^{th} feature of i^{th} observation

(for j=0 to j=n) → (Update all θ_j s simultaneously before moving to next iteration.)

}

Source - https://www.crayondata.com/wp-content/uploads/2017/09/image17.png

Objective function: Logistic Regression

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x), y)$$

$$\text{Cost}(h_\theta(\mathbf{x}), y) = -y log(h_\theta(\mathbf{x})) - (1-y) log(1 - h_\theta(\mathbf{x}))$$

Gradient descent for logistic regression:

while not converged {

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \text{ for } j = 0, 1, \ldots, n$$

}

Source - https://i.stack.imgur.com/zgdnk.png

Results

Number of threads(n)	Time taken for Linear Regression (seconds)		Time taken for Logistic Regression (seconds)	
	OpenMP	MPI	OpenMP	MPI
2	1.221	1.447	1.957	2.6713
4	0.6783	0.835	1.223	1.6554
6	0.655	0.984	1.492	1.657
8	0.899	0.878	1.328	1.789

Here, N = number of data points = 1,00,000
Time taken is averaged over 5 runs

Results (continued)

Number of data points (N)	Time taken for Linear Regression (seconds)			Time taken for Logistic Regression (seconds)		
	Serial	OpenMP	MPI	Serial	OpenMP	MPI
100	0.0179	0.0112	0.0438	0.028	0.015	0.018
1000	0.088	0.0496	0.058	0.1656	0.092	0.102
10000	0.2608	0.1534	0.237	0.4675	0.25	0.304
50000	1.115	0.6198	0.894	1.701	0.972	1.128
100000	2.173	1.221	1.447	3.718	1.957	2.6713
500000	12.29	6.4728	8.3548	21.48	11.02	15.477

Here, n = number of threads = 2 for OpenMP and MPI codes
Time taken is average over 10 runs

Conclusions

- OpenMP is slightly faster than MPI as the average times taken by OpenMP are less than MPI
- Linear Regression takes less time than logistic regression because logistic regression additionally requires computing the logistic function values.
- Also linear regression converges closer to the actual parameter values than logistic regression because the data is linearly distributed.
- Average time taken increases as we increase the number of data points since it requires more time to compute the error.
- Number of iterations required to converge increases slightly as the number of data points increases.

Work distribution

All of us contributed equally in all of the tasks, helping and discussing with each other during meets. Following tasks were done:

- Serial implementation of linear and logistic regression
- Parallel implementation using OpenMP for both
- Parallel implementation using MPI for both
- Interpreting the results and drawing conclusions
- Preparing the report and making the presentation