Predicting Atom Location Using Machine Learning Algorithms

Adam Martini, Ran Tian, Wes Erickson

University of Oregon

martini@cs.uoregon.edu

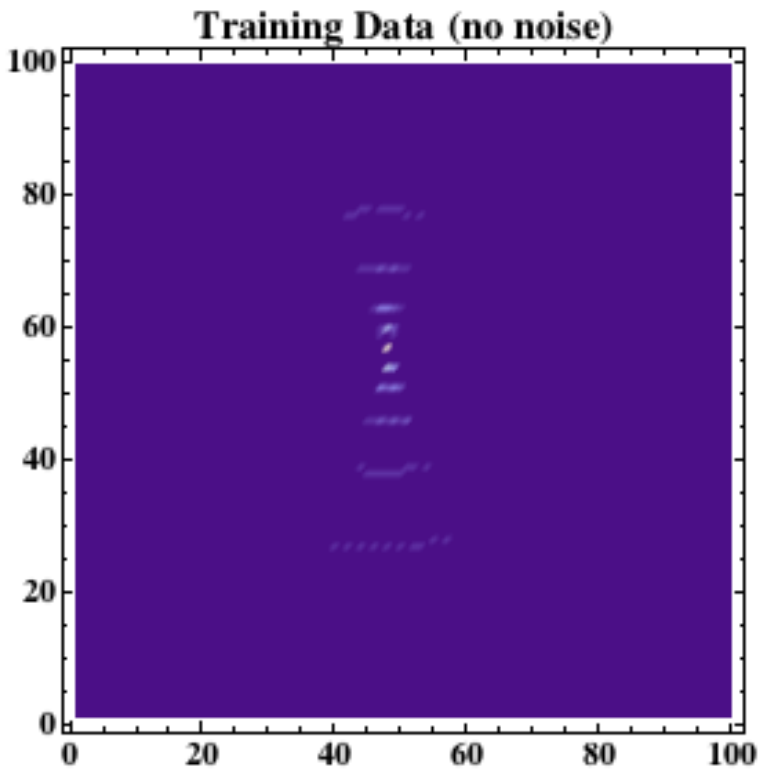
jmty0083@cs.uoregon.edu

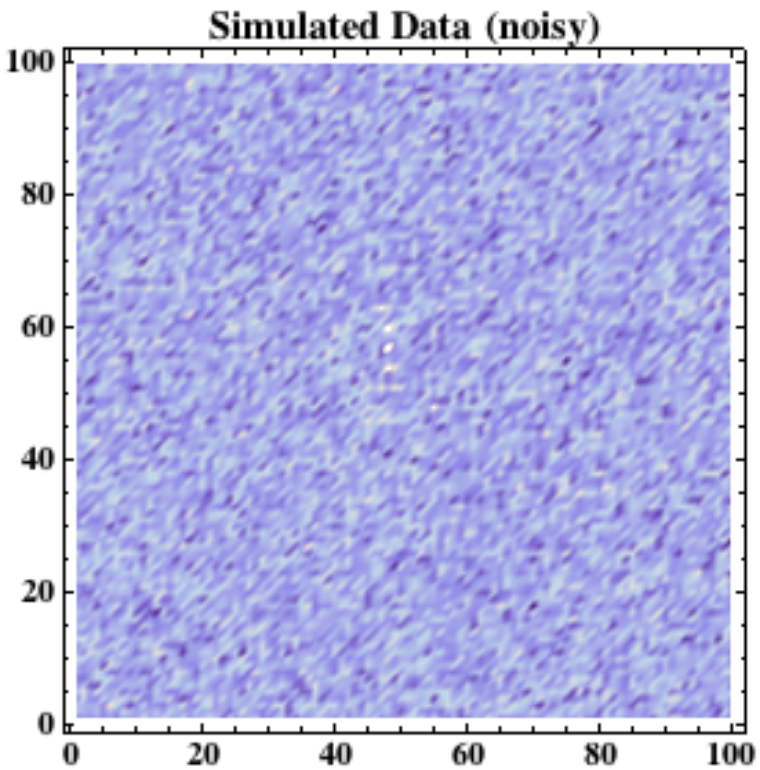
wwe@uoregon.edu

Friday 13th June, 2014

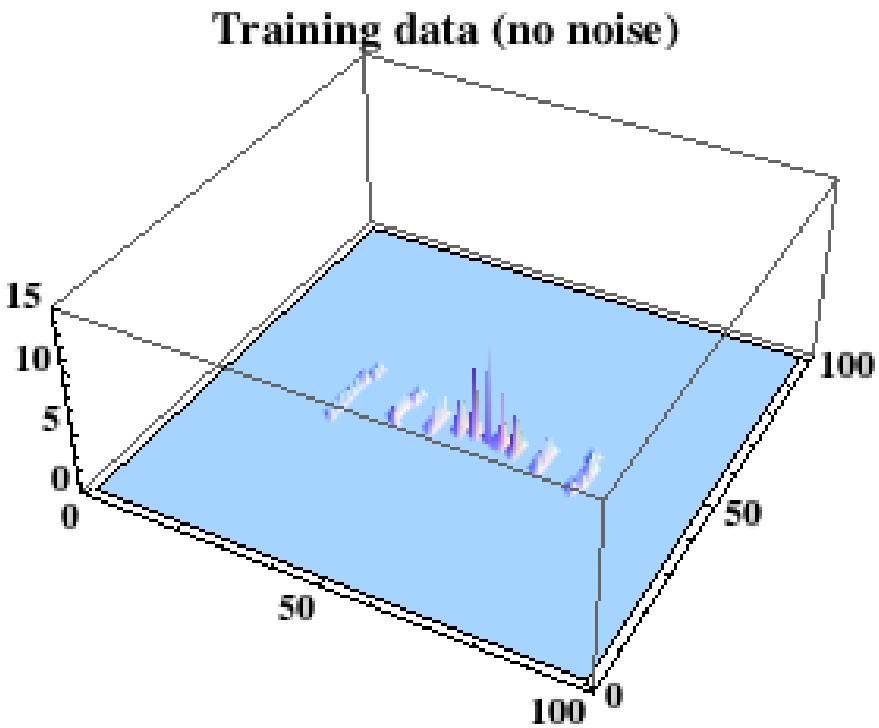
- Why atom location?
  - Camera in the Steck Lab takes pictures of atoms, but we want to know where its location.
- Why machine learning?
  - Automatically predict atom location for a new image based on training examples.
- Why shared memory parallelism?
  - Generate finer resolution images faster.
  - Finer resolution resolution images provide better accuracy.
- Why distributed parallelism?
  - Gradient descent is an *embarrassingly parallel* iterative process.
  - Scalable data $\Rightarrow$ more available parallelism (Gustafson's Law).

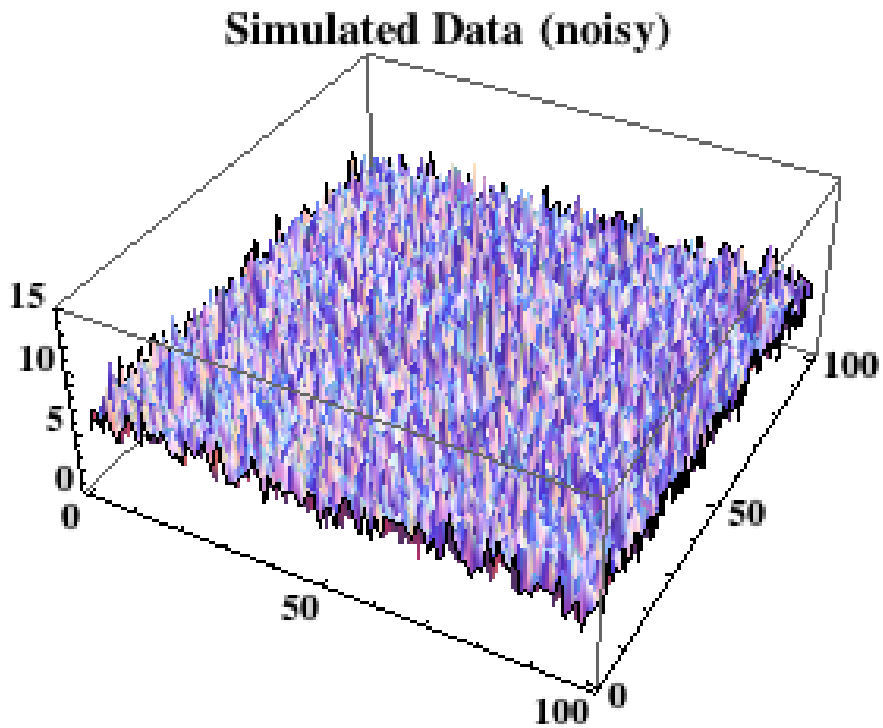
Input Data


Training Data (no noise)
100
80
60
40
20
0
0
20
40
60
80
100


Simulated Data (noisy)
100
80
60
40
20
0
0
20
40
60
80
100

Input Data


Training data (no noise)
15
10
5
0
0
50
100
0
50
100


Simulated Data (noisy)
15
10
5
0
0
50
100
0
50
100

Design and Objective

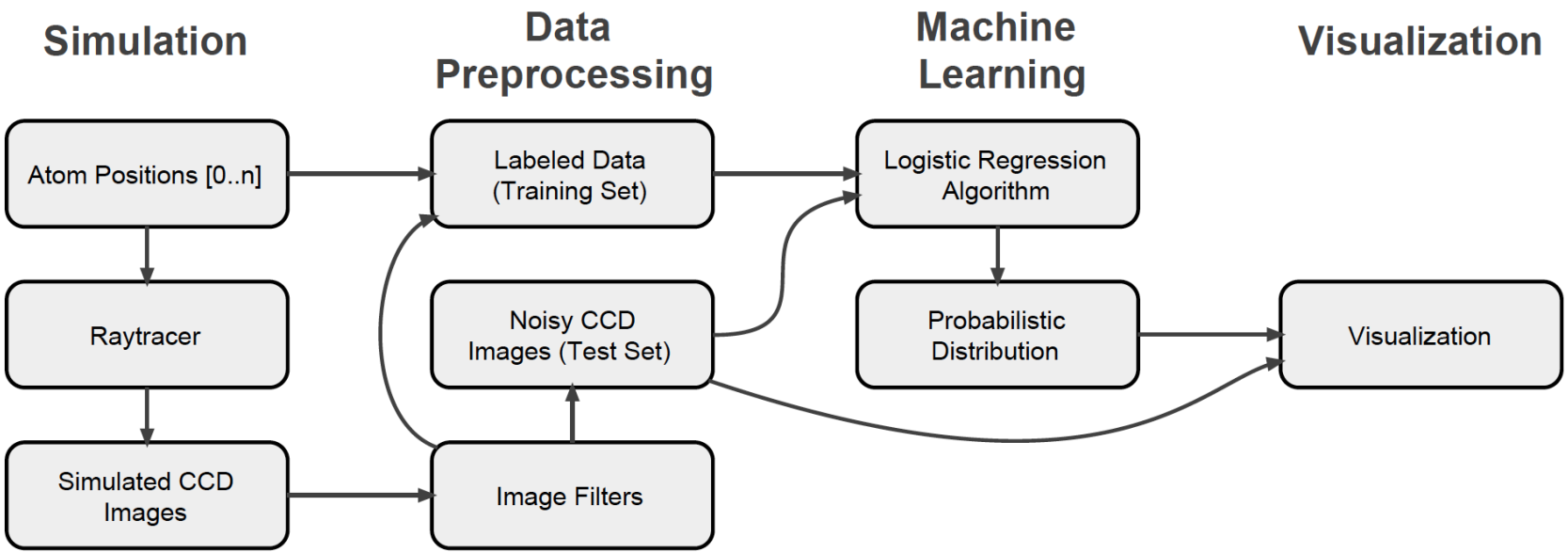


Figure : High Level Data Flow Architecture Diagram

Objective: Create and test a scalable machine learning solution for atom location prediction using CCD images.

Development and Implementation

Our development efforts are divided into sections based on our data flow architecture with some overlap of effort in the data preprocessing step.

4 Development Directions:

- Simulation
- Data Preprocessing
- Machine Learning
- Visualization

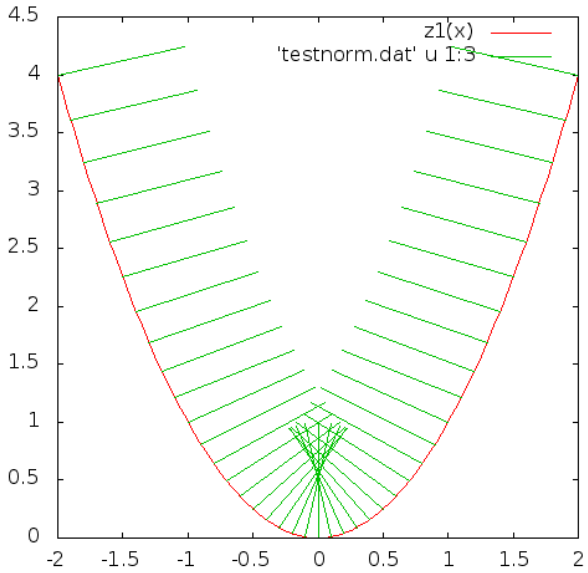
Simulation

Raytracer Development

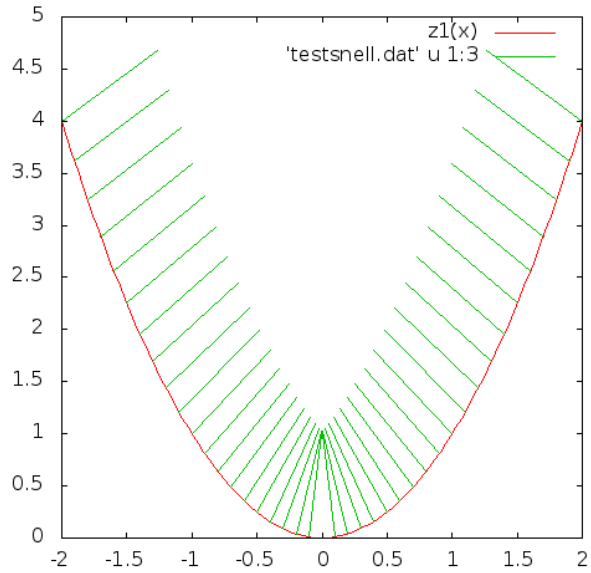
- Basic Raytracer – vector operations, surface definition, refraction
- Imaging System

Figure : Scale model of the atom imaging system.

Simulation

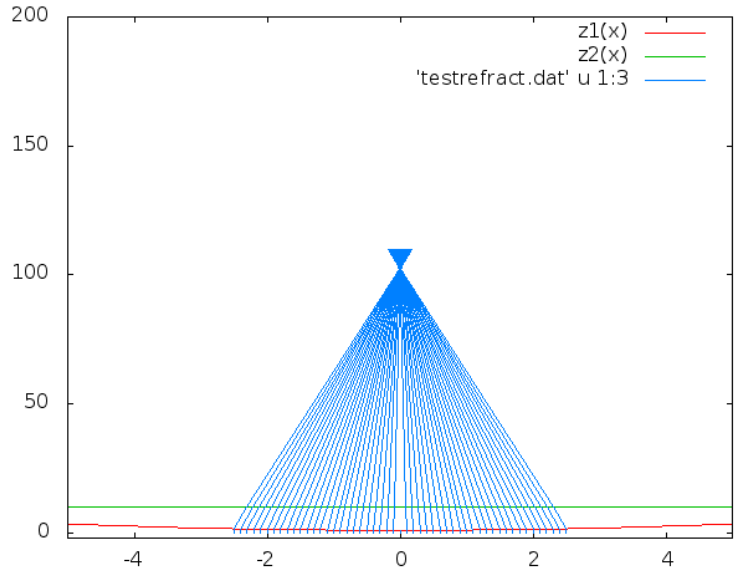


(a) A norm

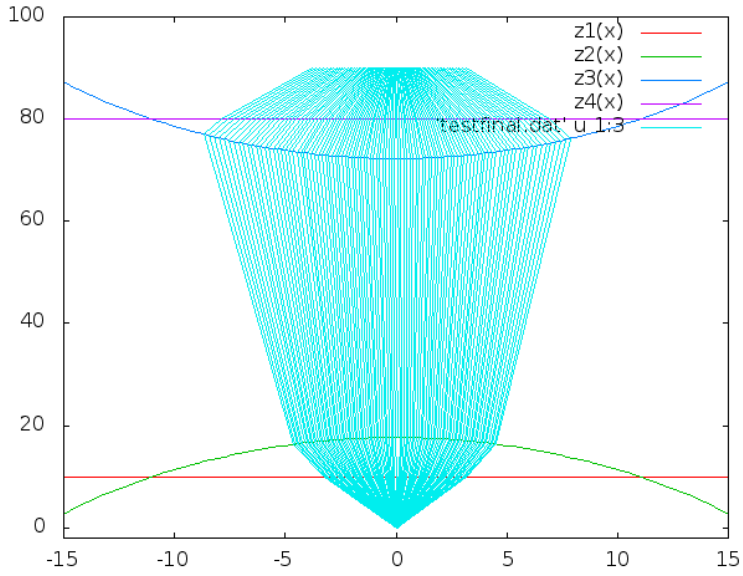


(b) A snell

Simulation



(a) A refract



(b) A lenstest

Data Preprocessing

4 Directions for Data Preprocessing:

- Filtering
- Noise Adding
- Data Partitioning
- Feature Scaling

Machine Learning

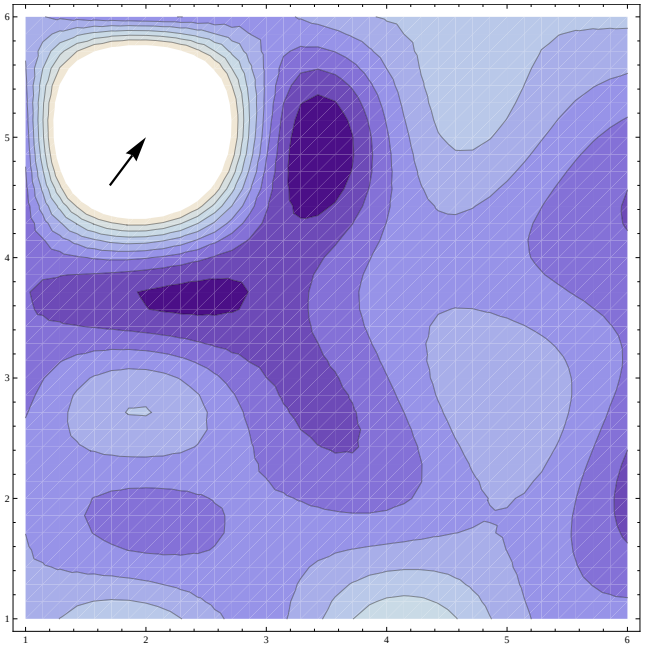
Built distributed logistic regression classifier capable of multi-class classification and mini-batch processing:

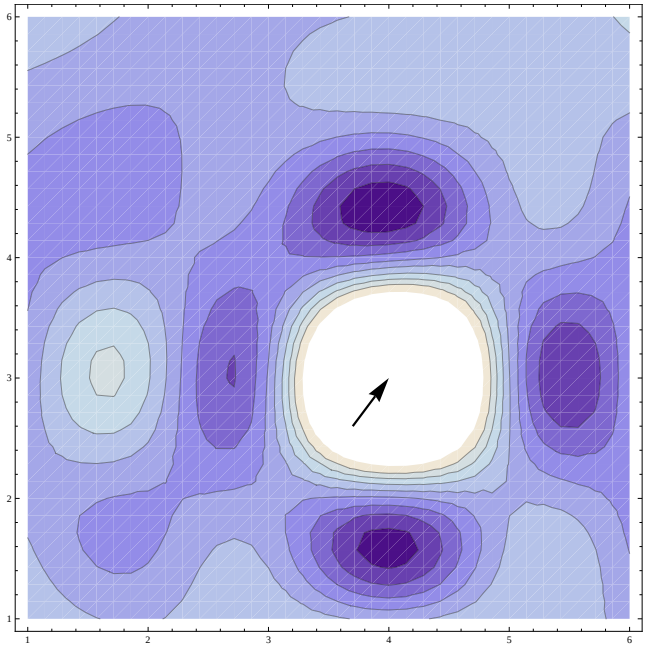
- Data partitioning and feature scaling handled automatically
- Vectorized implementation using Eigen
- MPI with Allreduce and custom reduce fuctions
- Iteratively performs gradient updates, reduces updates summation, normalization occurs on all nodes

Unsuccessful attempt to use distributed SVM implementation to compare with our own classifier.

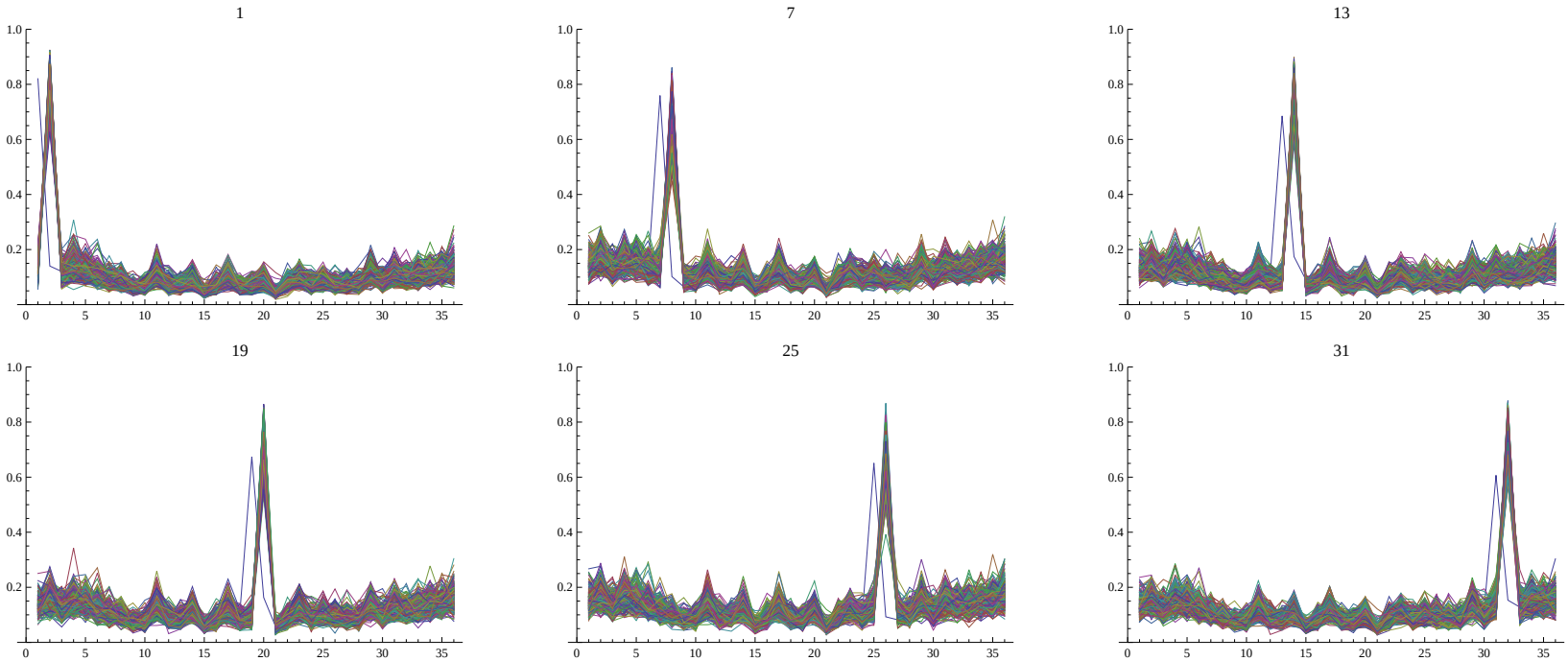
Data Visualization

Used Mathematica to create contour plots to visual images and plots to demonstrate accuracy of the classifier.





Data Visualization



Experiments

Trained and on two data, one small and one large to test scalability.

Prediction Results are 100% accurate after only 10 iterations.

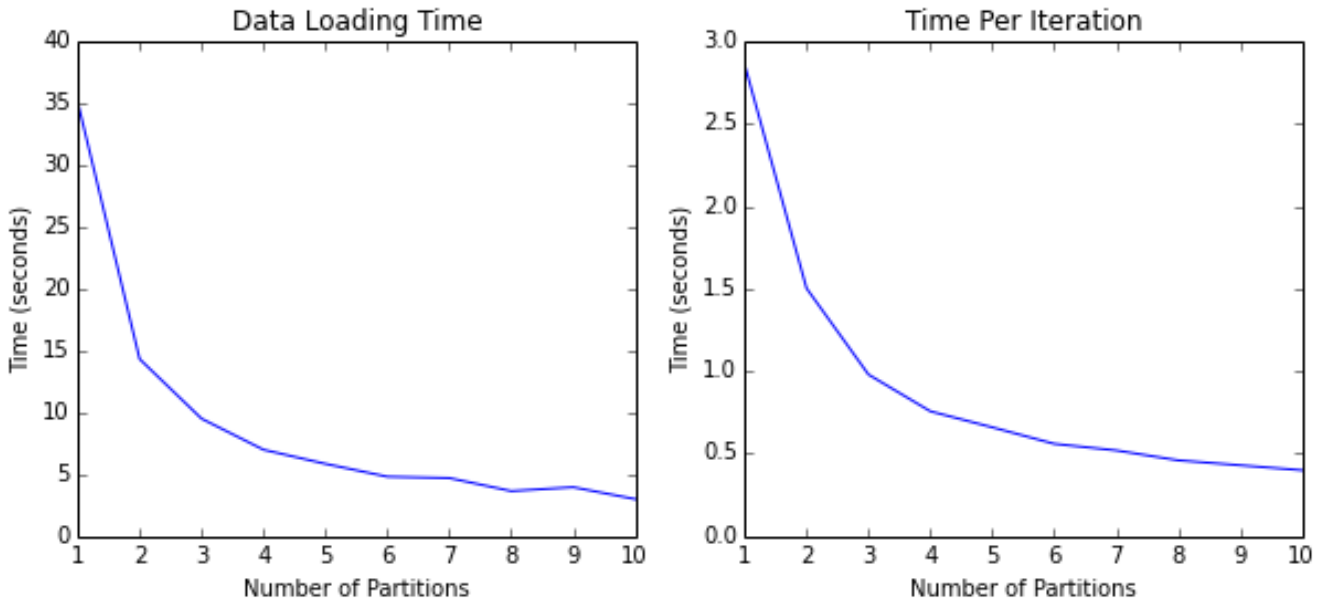


Figure : Performance gains for both data loading and iteration time on the small training set.

Experiments

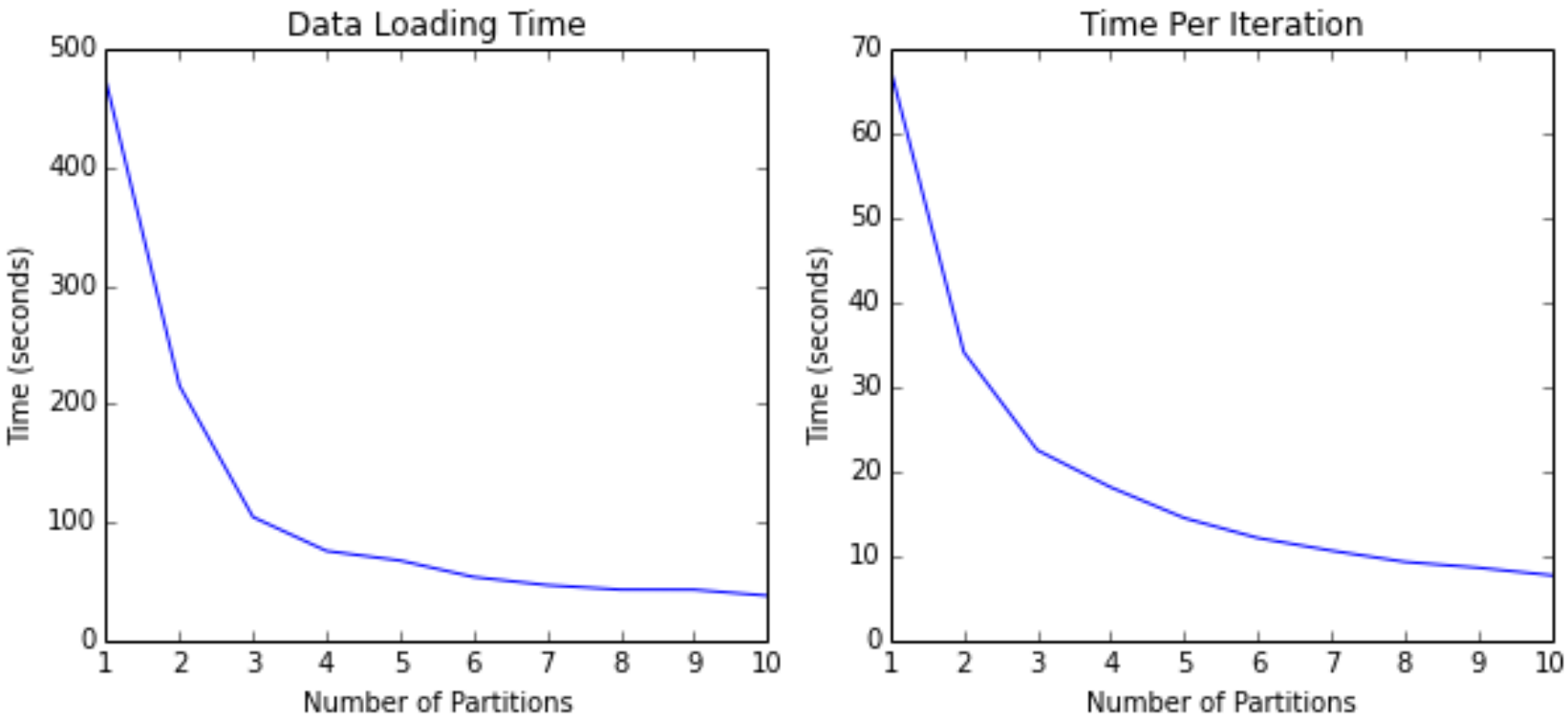


Figure : Performance gains for both data loading and iteration time on the big training set.

Conclusion

- Implemented parallelized raytracer and generated lots of images
- Preprocessed images to create training and testing sets
- Successfully implemented and tested distributed logistic regression classifier using our training and testing dataset
- Visualized out results to prove classifier accuracy and performance

The End