COMP 551 Project Report

Eren Ozturk - Yanze Liu - Siqi Zang

Abstract

In this project, we implemented linear and logistic regression on two separate datasets with the aim of understanding their working principles, as well as evaluating and comparing their performance. We experimented with different optimization techniques and model parameters for both methods and measured their impact on performance metrics such as runtime and accuracy. We identified potential pitfalls in parameter selection and attempted to choose the best parameters. Furthermore, we delved into the underlying meaning of the data and attempted to gain a deeper understanding of the relationships and patterns present. Overall, this project allowed us to gain insight into the principles of linear and logistic regression, which are among the most used machine learning techniques in academia and industry.

Introduction

In this project, two datasets were examined. The first dataset is about energy efficiency. It contains information about 8 different parameters of simulated buildings and the results about the energy required to condition the air inside. We used linear regression analysis with analytic solutions, gradient descent and stochastic gradient descent. We also implemented optimizations such as batching, adding momentum to the learning rate and adaptive gradient. Using these techniques, we obtained a model to predict the air conditioning efficiency of buildings with given parameters.

The second dataset contains discrete grades of various risk factors about companies as well as the true outcome of the companies - whether they are bankrupt or not. For this dataset, we used logistic regression to develop a binary classification model to predict if a company is going to be bankrupt given certain levels of risk factors. This binary classification problem had room for optimization for speed, as the data was more clearly separated than the first one.

For both models, we identified some pitfalls concerning convergence, runtime and numerical accuracy. There were also some challenges with the underlying data. We describe these issues, and characterize different optimization techniques and their relative performances.

Datasets

The first dataset is characterized by two results: heating and cooling load. These are the two output features that will be used in the analysis. The input parameters that were given to the simulation are as follows: relative compactness, surface area, wall area, roof area, height, orientation, glazing area and glazing area distribution.

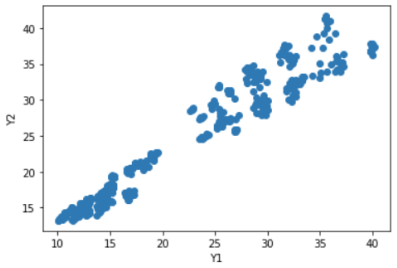
Our first analysis step was to clean the data by dropping any null values and checking for duplicates. After sanitizing the data, we removed outliers to improve the results of the linear regression analysis. We then ran some preliminary analysis to identify potential issues:

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
Mean	0.741	690.71	309.747	190.485	4.807	3.499	0.229	2.81	19.542	22.026
Std	0.0926	81.938	34.856	39.476	1.694	1.118	0.133	1.558	8.9	8.43
Min	0.62	563.5	245	122.5	3.5	2	0	0	6.01	10.9
Max	0.9	808.5	367.5	220.5	7	5	0.4	5	42.11	43.33

In the 613 data points left, we noticed that the input features had widely varying units and ranges. This would cause the features with large values to have an outsized effect on the weights, corrupting our analysis. To mitigate this, we used z-score normalization to normalize the data around a 0 mean and 1 standard deviation:

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
Min	-1.306	-1.553	-1.858	-1.722	-0.772	-1.341	-1.723	-1.804	-1.52	-1.319
Max	1.717	1.437	1.657	0.76	1.294	1.342	1.28	1.405	2.536	2.525

Since heating and cooling are the same process with reverse direction, we guessed that there is a correlation between the energy required for the two. In fact, there is a significant correlation between heating and cooling load:



We considered the effect of this on our analysis process, and decided that this was not a problem because the analysis can be run independently on the two variables.

One ethical concern about this analysis is the potential use of this data. Some parameters penalize rural housing more than urban housing since urban living spaces tend to be less compact. If this model is used to determine an energy inefficiency tax, this could be seen as disadvantageous to rural communities.

The preprocessing for the second dataset was more straightforward. The bankruptcy and solvency outcomes were mapped to ‘1’ and ‘0’. Accordingly, the risk factors were mapped ‘-1’ for positive, ‘0’ for average and ‘1’ for negative outlooks. This way, the weights had the correct sign - higher positive weights were associated with higher log-odds for bankruptcy.

An ethical concern about this data is the fact that it maps qualitative (i.e. subjective) measurements to quantitative measurements. Mappings like this can be used to pretend like subjective measurements are objective ones, by pointing out that they were obtained with machine learning. However, if the underlying data is biased to begin with, the results will also carry the same bias.

Results

1. and 2. Training with 80% of samples (and weights)

a) Linear Regression on Dataset 1

	Training MSE (One test)	Testing MSE (One test)	Training MSE (On anverage)	Testing MSE (On anverage)
Y1	0.099	0.062	0.117	0.114
Y2	0.114	0.091	0.136	0.133

```
The weight and intercept that we have: (The first line is the intercept)
[[ 0.00905824  0.00352972]
 [-1.25190065 -1.23176984]
 [-1.18431448 -1.19463731]
 [ 0.18764808  0.16187544]
 [-0.90375974 -0.93914515]
 [ 0.0644764   0.0123322 ]
 [ 0.00134315  0.01514016]
 [ 0.24561527  0.17678468]
 [ 0.0430158   0.0116564 ]]


The MSE for the train set:
The error for Y1 is: 0.09956333054466665
The error for Y2 is: 0.11470031147982435


The MSE for the test set:
The error for Y1 is: 0.06259754653498671
The error for Y2 is: 0.091188124557499
```

```
The weight and intercept that we have: (The first line is the intercept)
The Training is finished. MSE=0.09234717527095736 trained 100000 times
The Training is finished. MSE=0.10812424403431181 trained 100000 times
[[ 2.67512656e-03 -7.89442408e-04]
 [-1.25738818e+00 -1.23664001e+00]
 [-9.47100401e-01 -9.98824293e-01]
 [ 8.74267675e-02  7.65497533e-02]
 [-1.02067923e+00 -1.06939979e+00]
 [ 1.89564004e-01  7.58887338e-02]
 [ 9.11186761e-03  3.53025059e-02]
 [ 2.41214067e-01  1.80484779e-01]
 [ 4.39408619e-02  1.59065900e-02]]


The MSE for the train set:
The error for Y1 is: 0.09234717527095736
The error for Y2 is: 0.10812424403431181


The MSE for the test set:
The error for Y1 is: 0.09178954808769652
The error for Y2 is: 0.11943609944827872
```

(Left is the result of Closed form linear regression, Right is linear regression with GD)

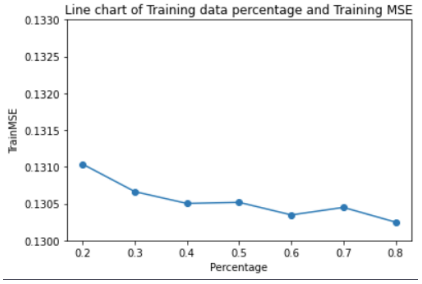
b) Logistic Regression on Dataset 2

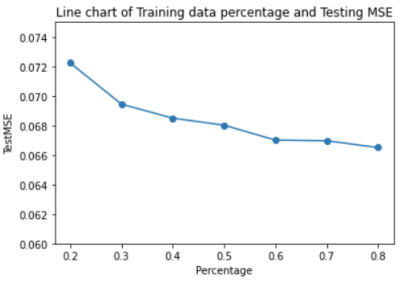
```
Results
_____________________________________________________________________________
Weights:
[0.49238342 0.28612975 1.92275838 1.96136404 3.69902732 0.1433452 ]

Bias term:
-2.1724791968811257


Performance:
_____________________________________________________________________________
Classification accuracy: 100.0%
Training done in 2484.67ms
Classification done in 0.08ms
```

Please see the codes for more high resolution images of results. For Dataset 2, we can see that the 'competitiveness' field has the highest impact for companies in their success.

2. Effects of training / test data ratio

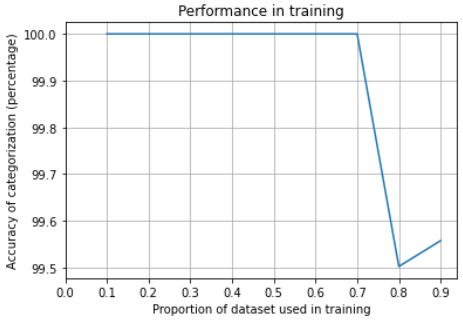
a) Linear Regression

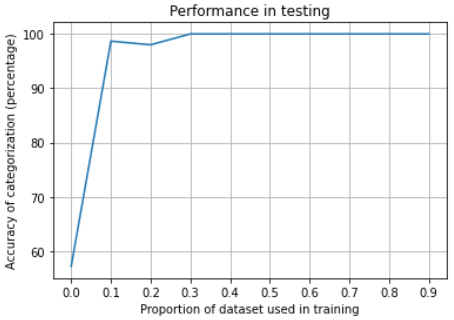




We see that there is a correlation between MSE performance of the linear regression and the amount of data that is used to train the model.

b) Logistic Regression





We see that, when there is no training, the binary classifier is only slightly better than

50% which is just random guessing. The performance is maximized when at least 30% of the dataset is used in training. For the accuracy in training, it is possible that the performance in 80 and 90% are rounding errors (99.55556% accuracy)

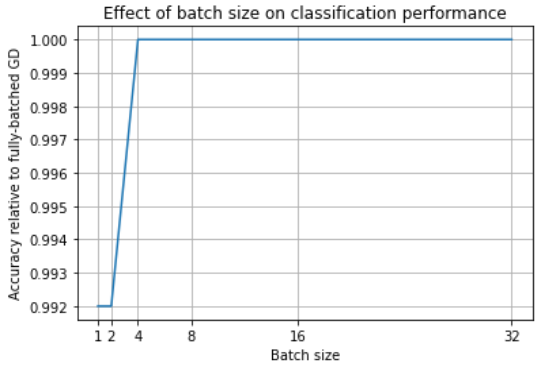
4. Effects of minibatch size

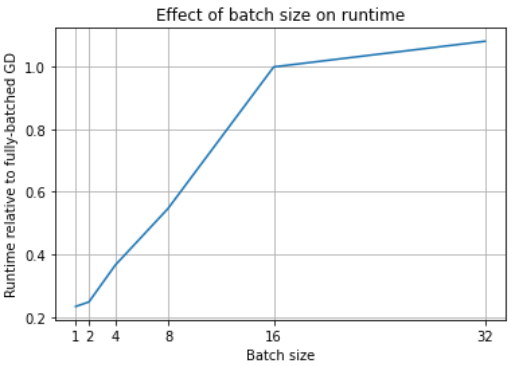
c) Linear Regression

Batch Size	Convergence Speed	Accuracy (MSE)
1, 2, 4	Unable to converge in max. time	~0.5 at most
8	Unable to converge sometimes, very unstable	Sometimes good
16, 32	Unstable, ~60-75 updates	Mostly good
64, 128	Stable, ~40-50 updates	Good
Fully-batched	Slightly more than 128-batched	Good

From this experiment, we concluded that 128-batching was the ideal option for this problem.

d) Logistic Regression





We can see that a batch size of 4 already reaches the same accuracy as a fully-batched model at less than 40% of its runtime. A batch size of 4 is ideal for this relatively simple binary classification problem.

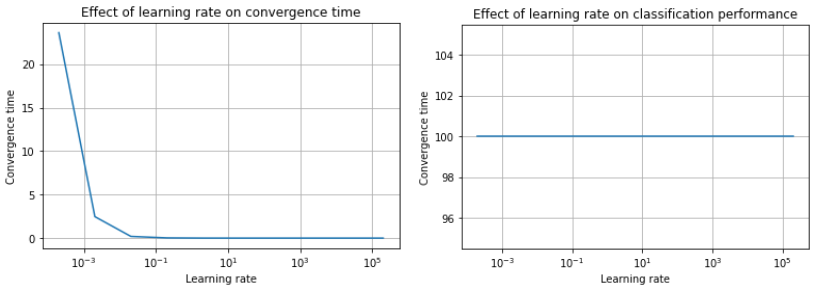
5. Effects of learning rate

a) Linear Regression

Learning Rate	Convergence?	Accuracy
0.6	No	N/A
0.5	Yes, very quickly	Good
0.01	Yes, slowly	Better than 0.5 after many iterations

(Please see the program for more precise numbers. The report page limit does not allow us to include all the data.)

b) Logistic Regression



For logistic regression, there does not seem to be a learning rate which stops convergence. However, there are some overflows after a certain rate, and there is no advantage beyond a rate of 0.1. There is no effect on classification performance, likely because all the rates converge, provided that they do not cause an overflow.

6. Analytical solution vs. Gradient Descent

Theoretically, gradient descent is faster than linear regression for large datasets. In the test case, the linear regression runs much faster. Since the analytical solution is a single iteration of matrix multiplications, machines can take advantage of SIMD instructions, which could possibly be faster than gradient descent. Another possible cause is the fact that our dataset is relatively small, so the advantage of adaptive gradient may not be obvious.

7. Adaptive Gradient Method

A learning rate of 0.6 could not converge in Part 5. One optimization we made was to use the **adaptive gradient method**, together with some momentum:

```
def adagrad(self, g):   #adaptive gradient
    self.sigma = self.sigma + np.square(g)
    v = self.rate / (pow(self.sigma, 0.5) + self.ep) * g - self.momentum * self.lastV
    self.update(v)
    self.lastV = v
```

```
The weight and intercept that we have: (The first line is the intercept)
The Training is finished. MSE=0.08378697821485745 trained 100000 times
The Training is finished. MSE=0.10430964511803553 trained 99892 times
[[-0.00937362 -0.00587656]
 [-1.01601044 -1.13655294]
 [-0.84569454 -1.00808021]
 [ 0.12115163  0.10308778]
 [-0.85804329 -0.959455  ]
 [ 0.21070412  0.09237515]
 [-0.00269948  0.01381247]
 [ 0.23865714  0.16907907]
 [ 0.03280542  0.01133678]]


The MSE for the train set:
The error for Y1 is: 0.08378697821485745
The error for Y2 is: 0.10430964511803553


The MSE for the test set:
The error for Y1 is: 0.12564049768127877
The error for Y2 is: 0.13357783176937665
```

With this method, we are able to get a precise result with a learning rate of 0.6.
Please see the program for more details, as well as another potential method with which we tried to improve efficiency.

Statement of Contributions

Yanze Lyu implemented the linear regression, included the results and explanations about them in the report. He also implemented the extra features of Part 7. Siqi Zhang implemented the data acquisition for logistic regression (from URL). Eren Ozturk implemented the rest of the logistic regression algorithm and added the results and explanations to the report, as well as writing the “Abstract” and “Introduction” sections and editing the writing.