EC ENGR 236-A (Linear Programming) Project Report

Introduction

In this project, a sparse linear regressor was designed, whose primal function is listed in (1). For every task except Task 2, an LP program was used to generate the optimal results.

$$\min_{\theta,b} \frac{1}{N} \|Y - (X\theta + b\mathbf{1})\|_1 + \alpha\|\theta\|_1 \qquad (1)$$

Task 1-1

Equation (1) is not linear because of the norm-1 operator being applied on the function; this can be removed by using slack variables to linearise the program. The reformulated linear program for (1) is written below and this reformulated LP can be used for solving linear programs in libraries such as CVXPY)

$$\begin{gathered}\min_{t_1,t} 1^t t + \alpha(1^t t_1) \\ subject\ to: -t \leqslant \frac{Y-(X\theta+b\mathbb{1})}{N} \leqslant t \\ -t_1 \leqslant \theta \leqslant t_1 \\ t_1, t \in \mathbb{R}^n\end{gathered} \qquad (2)$$

Task 1-2

The implementation was carried out using CVXPY. In this task, the regressor is trained with different values of α, that affect the amount of L1 regularisation being added to the regression, which helps prevent the model from overfitting on the training data. The larger the α, the sparser the solutions. In this experiment, α was varied logarithmically from 10^{-5} to 5, the results for the regressor on the synthetic and online news popularity dataset are shown in Fig. 1 and Fig.2 respectively. By looking at the graphs there is a point following which the regressor's performance starts to degrade. For the synthetic dataset this point is at 10^{-2} and for the online news popularity dataset this is at 0.5. Interestingly after the breaking point, an increase in α does not translate to a change in the regressors performance which becomes near constant after the breaking point, indicating that post a certain sparsity, the regressor does not learn anything more from the data or in other words, the LP is more involved in finding a sparse solution than finding a solution to the regression problem.

Task 1-3

To identify which features to retain, an ILP(Integer Linear Program) was designed to find the optimal indices, which is generalised solution. The regressor is first trained and the θ are then put through an ILP to find the optimal features to be removed as shown in (3). A mask M is used which can either have 0 or 1, where if it holds the former that feature is irrelevant and 1 means it needs to be retained. The constraints force only the diagonal to have non-zero values. The intuition is that by minimising the norm-1 distance between the θ_{original} and θ_{masked}, the optimal solution that arrives from θ_{masked} would not be at the optimal minima θ_{original} is at, but is as close as possible to the original regressor. Other algorithms involving pearson correlation, mutual information theory were considered but these algorithms are sensitive to noise, outliers and assume a gaussian distribution which may not be the case with every dataset, hence a generalised ILP approach was taken. Secondly, another approach was considered using X instead of θ but the computational complexity of the former grew exponentially while the latter stayed almost constant.

$$\begin{gathered}\min_{M} \|\theta - M\theta\|_1 \\ subject\ to: \sum_{i=1}^{n} M_{ii} = k \\ \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij} = k \\ M_{ij} \in \{0,1\} \\ 0 \leqslant k \leqslant n\ , k \in \mathbb{Z}\end{gathered} \qquad (3)$$

The results for the regressor on the synthetic and online news popularity dataset are shown in Fig. 3 and Fig.4 respectively. As seen from the graph, even when fewer features are used the regressor's performance change is negligible until the break down point which is 20% for the synthetic dataset and 30% for the online news popularity dataset. What this highlights is that many of the features are redundant and do not actually contribute to the regressor’s performance in fact at the breakdown point, the performance of the regressor gets even better which highlights that the other features might just be confusing the regressor’s decision making capabilities.

Task 1-4

Similar to Task 1-3, an ILP was used to find the optimal number of samples to take as shown in (4), where if M_{ij} =0, means the i^{th} sample is removed from the optimal solution. Originally the sample reduction ILP was supposed to use *X*, but due to computational constraints, *Y* was used instead which helped improve the run-time since *Y* is a column vector, the graph developed is much smaller compared to the one developed for *X*. Moreover, since the sample reduction would lead to a removal in both *X* and *Y*. Finding the optimal samples to remove in *Y* is equivalent to finding the optimal samples to remove in *X*.

$$\begin{gathered}\min_{M} \|Y - MY\|_1 \\ subject\ to: \sum_{i=1}^{n} M_{ii} = k \\ \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij} = k \\ M_{ij} \in \{0,1\} \\ 0 \leqslant k \leqslant n\,, k \in \mathbb{Z}\end{gathered} \tag{4}$$

The results for the regressor on the synthetic and online news popularity dataset are shown in Fig. 5 and Fig.6 respectively. As seen in the graph, unlike feature reduction, the regressor is more sensitive to fewer samples and the regressor breaks down much earlier. For the synthetic dataset, the regressor’s performance is nearly constant until 40% post which the performance degrades and for the online news popularity dataset this happens at the 80% level. What this highlight is that not all samples are required for training the regressor, but training on too few samples causes overfitting which is the reason why at lower sample values, the training error goes to zero but the test error increases.

Task 1-5

To jointly optimise in both the sample and feature space, both ILP’s from previous tasks were used, although independently since the multiplication of two masks M_1M_2 is not a linear program. Instead the training dataset was first reduced in the feature space before being reduced in the sample space, since both spaces are orthogonal to each other and a reduction in one does not change the properties of the other, or in simpler words, by reducing the number of samples, the number of features don’t change and vice versa. The problem however was trying to find the optimal feature and sample downsampling given a communication cost. From Task 1-3 and 1-4, the regressor is more sensitive to samples, than features, but fixing the features is still not a solution since there are certain communication constraints that can be never be satisfied such as for a feature set of 20%, a communication constraint higher than 50% can never be achieved even after setting the sample set to 100%.

Therefore, an iterative algorithm was used which is akin to an LP that found a feasible sample and feature cost that allows for an optimal split. The results for the regressor on the synthetic and online news popularity dataset are shown in Fig.7 and Fig.8 and the algorithm for iteratively finding an optimal split is shown in Fig.9 respectively. By looking at the graphs, the regressor’s performance is constant and is robust even when trained on fewer samples and features, however after the breakdown point, the model starts to overfit causing the testing error to go up and the training error to go down which is further exacerbated by the reduction in features. The breakdown point for the synthetic dataset is 10% which translates to 20% features and 50% samples and the regressor is robust for both values as seen in Fig.3 and Fig.5. For the online news popularity dataset, the breakdown point is 30% which translates to 30% features and 100% samples. Other splits can be taken such as 40% and 75% , but Fig.6 highlights how sensitive the regressor is to sample reduction, the highest possible sample values were taken, thus preserving the essence of the dataset without a degradation in performance. This emphasis the orthogonality of the feature and sample space since an independent reduction in both still gave a robust regressor. Furthermore, datasets can be compressed sample and feature wise without a loss of generality.

Task 2

.

For this task, since the regressor is being trained in an online manner, the stochastic gradient descent(SGD) formula is used which is depicted in (5), η,γ are hyperparameters wherein the former called the learning rate sets the rate of convergence and the latter called momentum helps set the stabilise training by adding a portion of previous gradients during the update. Additionally, to boost the performance of, the SGD optimiser is trained on each data point 5 times. Too few, and the optimiser doesn't learn properly and too many causes it to overfit on the dataset. Moreover within the 5 iterations, with each iteration the learning rate decreases allowing for more nuanced updates to the weights.

$$\begin{gathered} \theta_n = \theta_{n-1} - \eta(sgn(\langle x_n, \theta_{n-1} \rangle - y_n) + sgn(\alpha\theta_{n-1}) + \gamma\theta_{n-2}) \\ b_n = b_{n-1} - \eta(sgn(\langle x_n, \theta_{n-1} \rangle - y_n) + \gamma b_{n-2})) \\ \eta, \gamma \in \mathbb{R} \end{gathered} \tag{5}$$

Furthermore in certain cases, taking the Polyak-Ruppert Average of *θ* has given better results and this is show by (6). Besides, the two additions to the optimiser side, other functionality were added to reflect that the regressor is

$$\begin{gathered} \bar{\theta} = \frac{1}{N} \sum_{i=1}^{N} \theta_i \\ \bar{\theta}, \theta \in \mathbb{R}^n \end{gathered} \tag{6}$$

now being trained in an online fashion. Similar to Task 1-5, an iterative approach was used to find the optimal feature and sample split to satisfy various communication constraints. Since the sensor and central node do not have *a priori* information about the samples or features, and making decisions on the 1st sample may not be optimal. The regressor is 'warm started' by being trained on 3 samples before feature reduction is done wherein an ILP uses θ to find the features to remove, this is reflected in the communication cost, by incrementing it by a value associated to one sample's worth. An example of this is 1 sample with 100% features is equal to 2 samples at 50% features.

More importantly a *pass-logic* is used to determine whether a sample should be sent to the central node or not. The central node sends the model parameters on the downlink which the sensor uses by evaluating the normalised error, if it is more than 500%, the sample is not sent to the central node. Having a smaller margin, does increase the quality of samples, but there are fewer samples to send to the regressor and having a large margin leads to adversarial samples being sent to the regressor which can corrupt the performance. There is a trade-off between quality samples and training dataset size.

The results for the regressor on the synthetic dataset and online news popularity are shown in Fig.10 and Fig.11 respectively. The algorithm block diagram is shown in Fig.12. By inspecting the results it, especially for the synthetic dataset, the online regressor is worse than the offline regressor and as the communication constraints are made smaller, the model degrades quickly and starts overfitting on the data, the breakdown point is 90%. Training the online regressor however was computationally cheaper. For the online news popularity dataset, there is a negligible difference between the online and offline regressor. The online regressor used the Polyak-Ruppert averaging to boost the performance of the regressor, the same algorithm however did not boost the performance of the regressor trained on the synthetic dataset. Furthermore even as the communication constraints are decreased, the performance of the online regressor stays constant for the regressor trained on the online news popularity dataset, the breakdown point is 10% which is a split of 30% features and 33.3% samples.

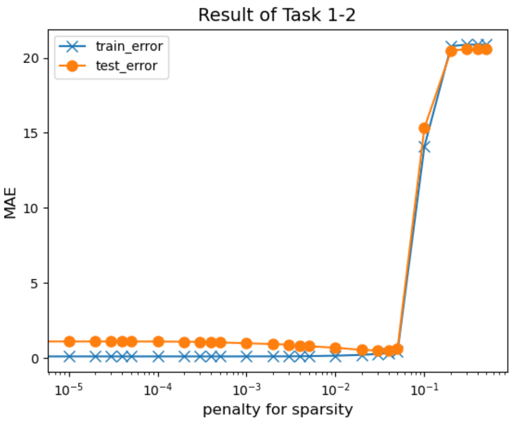
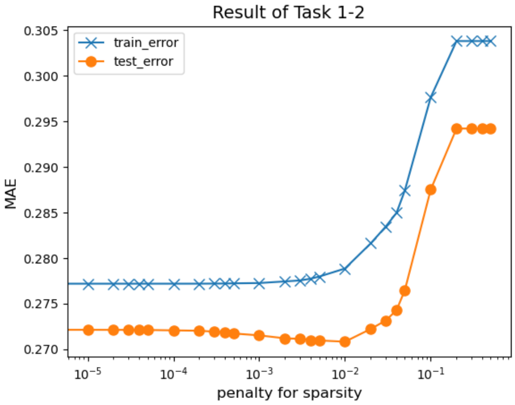
The poor performance of the regressor trained on the synthetic dataset could arise from adversarial examples present during the warm-start of the regressor and or using a wider margin than needed to filter samples, or the sensitivity of SGD to noise is another thing to consider as well.

Conclusion

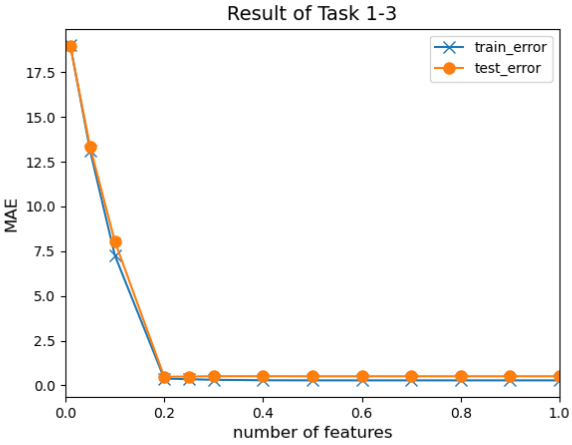
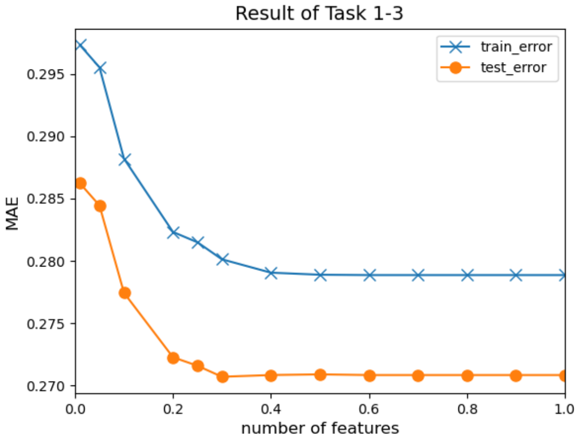
From the various tasks it can be seen that the regressor is more sensitive to sample reduction than feature reduction, furthermore joint optimisation is possible since both spaces are independent of each other allowing for downsampling independently. An optimal α value, helps prevent the regressor from overfitting, but a poorly chosen one can degrade the regressor and for the synthetic dataset although the online regressor trained faster and consumed resources, it came at the expense of a poorer model with a higher error. For the online news popularity dataset, there was no tangible difference between the online regressor and the offline regressor highlighting the lack of adversarial examples in the dataset unlike the former and Polyak-Ruppert Averaging did help boost the performance of the regressor. The trade-off present in all tasks show that there is a trade-off between computational resources and performance.

Appendix

Task 1-2

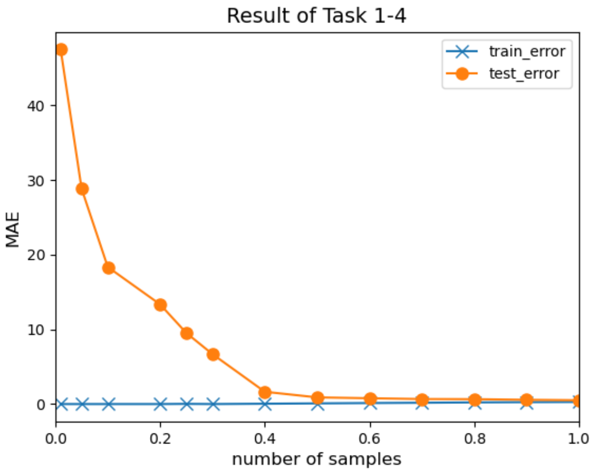
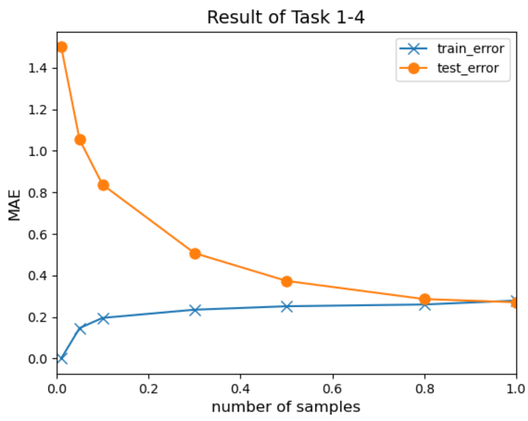
Results for the Synthetic Dataset	Results for the Online News Popularity Dataset
Fig.1	Fig.2
	

Task 1-3

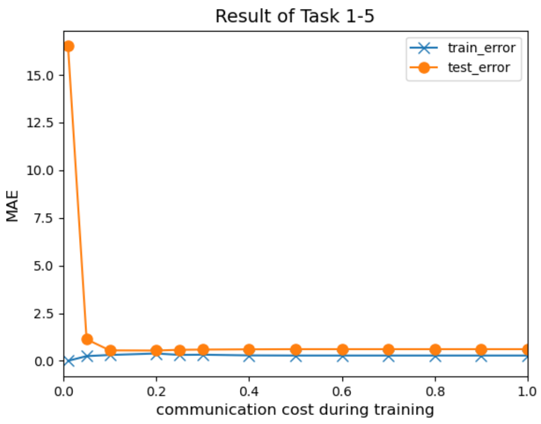
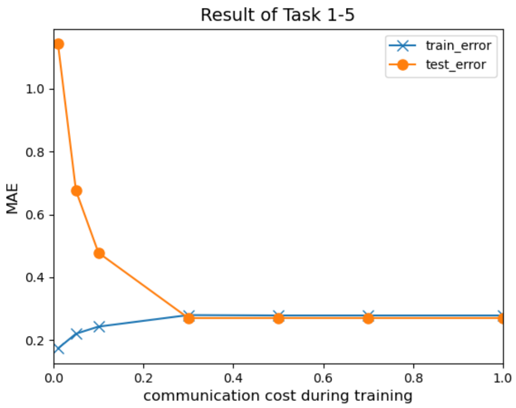
Results for the Synthetic Dataset	Results for the Online News Popularity Dataset
Fig.3	Fig.4
	

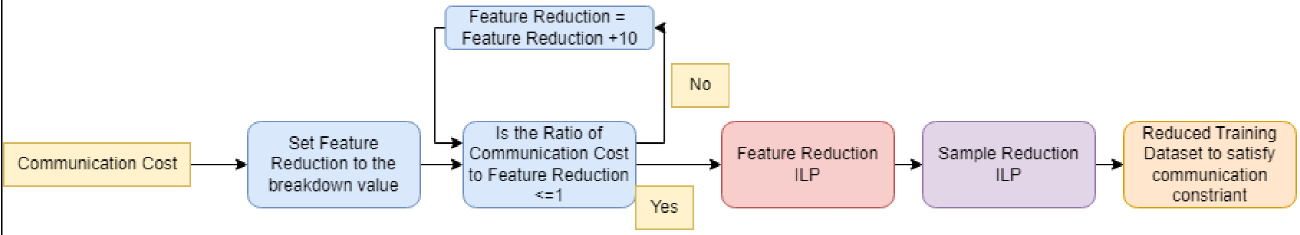
Task 1-4

Results for the Synthetic Dataset	Results for the Online News Popularity Dataset
Fig.5	Fig.6

Task 1-5

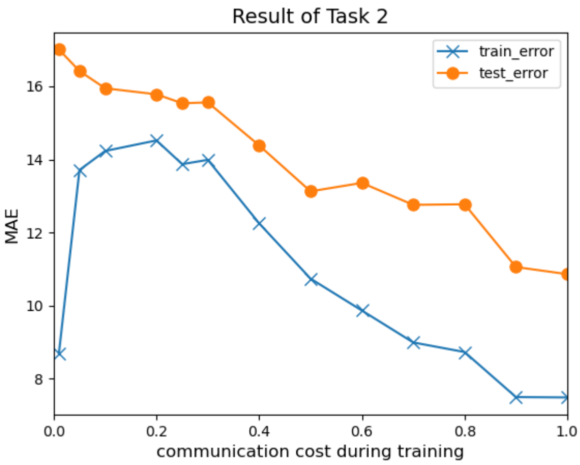
Results for the Synthetic Dataset	Results for the Online News Popularity Dataset
Fig.7	Fig.8
	

Task 1-5 Algorithmic Block Diagram
Fig.9
 Feature Reduction = Feature Reduction +10 No Communication Cost Set Feature Reduction to the breakdown value Is the Ratio of Communication Cost to Feature Reduction <=1 Yes Feature Reduction ILP Sample Reduction ILP Reduced Training Dataset to satisfy communication constriant

Task 2

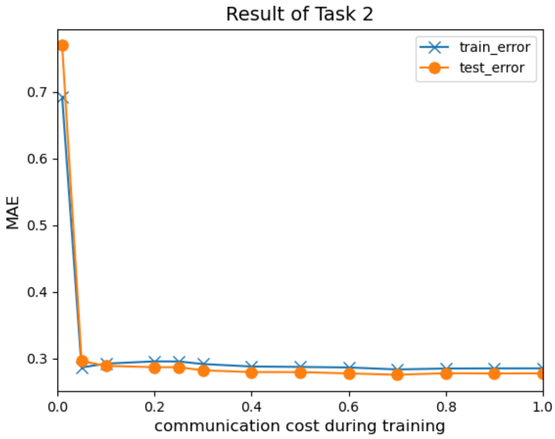
Results for the Synthetic Dataset

Fig.10



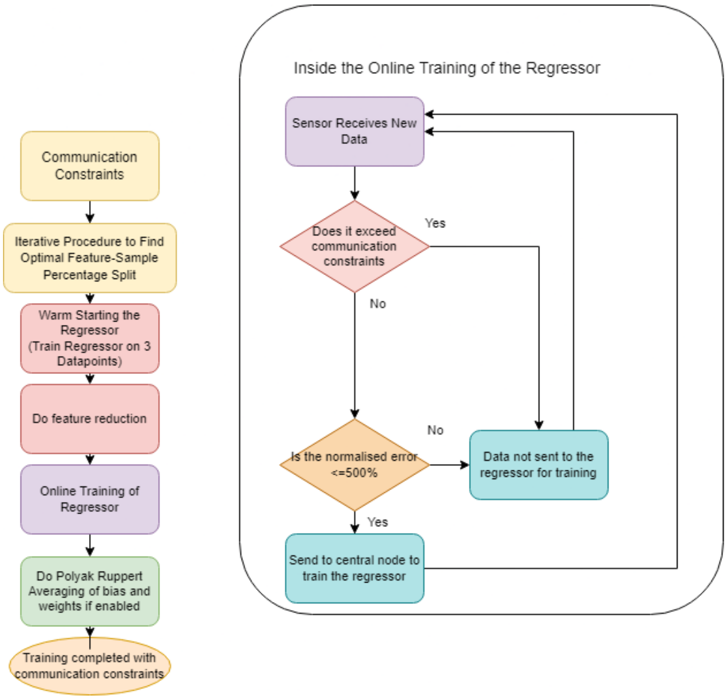
Results for the Online News Popularity Dataset

Fig.11



Task 2 Algorithmic Block Diagram

Fig.12


Communication Constraints
Iterative Procedure to Find Optimal Feature-Sample Percentage Split
Warm Starting the Regressor (Train Regressor on 3 Datapoints)
Do feature reduction
Online Training of Regressor
Do Polyak Ruppert Averaging of bias and weights if enabled
Training completed with communication constraints
Inside the Online Training of the Regressor
Sensor Receives New Data
Does it exceed communication constraints
Yes
No
Is the normalised error <=500%
No
Data not sent to the regressor for training
Yes
Send to central node to train the regressor