COMP3221 Assignment 2 - Federated Learning - 510460295

This report investigates Federated Learning using Multinomial Logistic Regression as the classification model. The dataset being used is based on handwritten digits from 0 to 9, which the model must learn to correctly identify. Federated Learning is a decentralised machine learning approach that enables multiple clients to collaboratively train a shared model while preserving data privacy.

Multinomial Logistic Regression, is a supervised learning algorithm that models the relationship between a set of features and a discrete target variable with multiple categories. The algorithm works by estimating the probability of an observation belonging to each class using a softmax function. The class with the highest probability is then selected as the predicted class.

I chose Multinomial Logistic Regression as my classification method as it has a fairly simple implementation and I could effectively use information from Tutorial 6 in order to aid me in my code production.

During the development process, one of the challenges encountered was ensuring accurate communication between the client and server, as the transmitted data would sometimes be utf-encoded and other times b64-encoded. To address this issue, I implemented a header function that provided information about the data type, message length, and the sender's port number. Additionally, to obtain the accuracy and loss data, I implemented a second header that would only be sent when the client transmitted a local model containing this information. This header methodology effectively helped me overcome the communication challenges that arose during development.

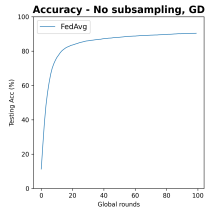
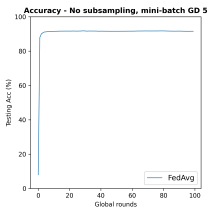
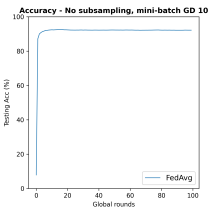
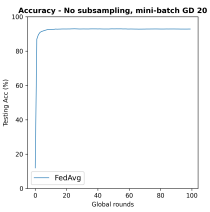
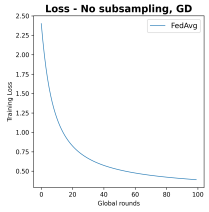
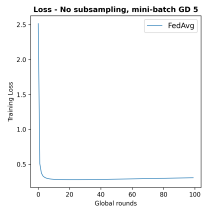
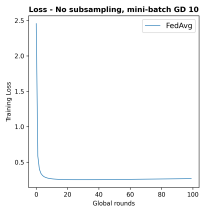
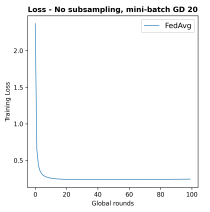
Furthermore, this implementation relied on Multinomial Logistic Regression as the sole learning model with local models being trained on 2 epochs, and utilised the FedAvg algorithm for federated learning. FedAvg is an algorithm that computes the average of the local model weights from all the clients, factoring in the amount of data each client has contributed to the learning process. This weighted average is then used to create a global model, which is shared with all clients for further training. Some limitations of my approach include the inability to handle client disconnections and the absence of alternative learning models or federated learning algorithms for a more comprehensive comparison.

All of the specifications of the task have been met in my implementation, all of the technologies involved in this process were torch (for the machine learning aspect), threading (for creating a listener function for the server), sockets (for handling communication between client and server), signal (for handling ending the processes), matplot (for creating the graphs).

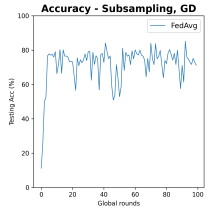
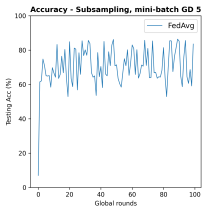
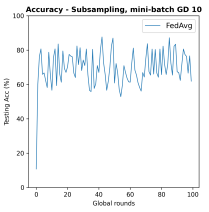
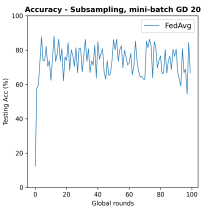
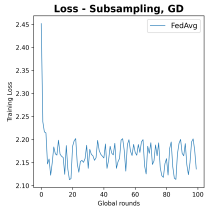
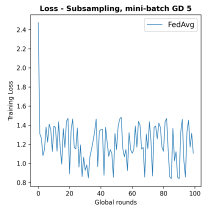
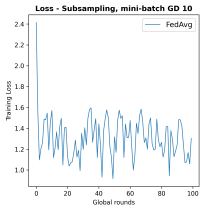
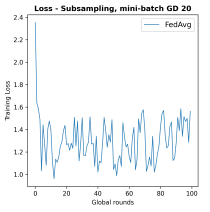
Results

To enhance the visibility of the results, consider increasing the font size, as the images are in PNG format and will maintain their clarity when enlarged.

No Subsampling

	Gradient Descent	Mini Batch Gradient Descent (Batch size: Increasing left to right)		
Accuracy				
Loss				

Subsampling

	Gradient Descent	Mini Batch Gradient Descent (Batch size: Increasing left to right)		
Accuracy				
Loss				

For each mini batch run, the learning rate was set at 0.01, I did run some with a learning rate at 0.001, however the values seemed to be similarly noisy.

Discussion

When comparing the accuracy graphs for gradient descent and mini-batch gradient descent with no subsampling, it is evident that mini-batch GD local training causes the global model to converge in fewer global rounds than gradient descent training. This is true for all batch sizes; however, a batch size of 5 demonstrates the sharpest curvature, indicating that it reaches convergence the fastest. In contrast, gradient descent does not appear to have reached convergence by the end of the 100 iterations, as it still exhibits a slight upward gradient.

An inversely identical trend is observed when examining the loss values for these same parameters, indicating that mini-batch training reaches convergence in significantly fewer iterations compared to standard gradient descent. These results clearly demonstrate that mini-batch gradient descent is more effective in reaching convergence within fewer iterations when compared to standard gradient descent. However, an observation not shown in the graphs is that mini-batch local training takes longer for each iteration and is thus more computationally demanding per iteration.

When comparing no subsampling to subsampling in the FedAvg function, there is a stark contrast in the effectiveness of these techniques. While all graphs without subsampling follow a predictable curvature, those with subsampling appear to be extremely noisy, exhibiting large fluctuations between each global round. An interesting observation is that the loss fluctuates more significantly when clients use mini-batch gradient descent compared to gradient descent. This observation could be studied further with more in-depth parameters, but such analysis is beyond the scope of this discussion.

From these results, it is evident that using subsampling when aggregating the parameters for the FedAvg algorithm is not suitable for this machine learning model, as it produces highly unpredictable results with no clear indication of increasing accuracy with each iteration over time. In contrast, without subsampling, the accuracy clearly trends upward as the loss consistently trends downward, effectively creating a machine learning model that can accurately (>90%) predict the class to which the data belongs.

I believe the reason that sub-sampling was ineffective for this model is because each of the clients is only training on 3/10 classes of the data. When subsampling, only 2 of the models are being chosen, which means there is a maximum of 6/10 classes being represented during the aggregation of the data. This limited knowledge of the classes being aggregated at one time could ‘dilute’ the current behaviour of the model in such a way that it biases the most recently learned classes, causing the fluctuations being seen.

In conclusion, the most effective combination for this machine learning model is using no subsampling for the FedAvg calculation and having the clients use mini-batch gradient descent with a batch size of 5 for training their local models.