Real-Time, Robust, and Reliable (R^3) Machine Learning Over Wireless Networks: Collaborative Feature Extraction

RUTGERS
WINLAB | Wireless Information Network Laboratory

Ayaan Qayyum (GR), Hasan Ali (UG), Nihal Abdul Muneer (GR), Joshua Menezes (GR), Madhav Subramaniyam (UG), Anand Sarwate, Waheed Bajwa

Abstract

This project investigates how streaming Principal Component Analysis (PCA) can be distributed across nodes to perform real-time dimensionality reduction efficiently with minimal communication, and explores trade-offs relevant to practical deployment.

Experiment Overview

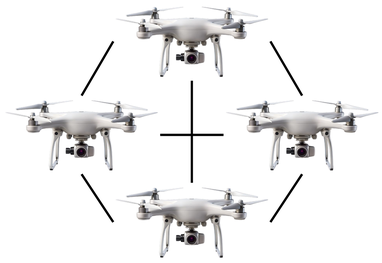
Evaluate distributed, streaming PCA algorithms. Explore system-level considerations, including performance, scalability, robustness, and accuracy in a testbed environment, while designing for real-world conditions.

Motivation

In dynamic distributed systems, such as a fleet of drones collecting sensor data, streaming PCA becomes important to run machine learning (ML) tasks with fewer dimensions.

Each drone collects its own data, and independently performs PCA, sending updates to surrounding drones which use those updates in subsequent iterations.

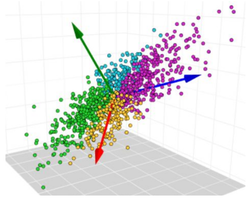
Drone Example



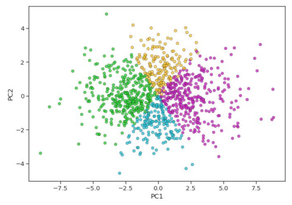
Background

Principal Component Analysis (PCA)

- A dimensionality reduction & compression technique
- Identifies the top informative features from the data



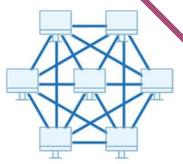
Original 3-Dimensional Data



Dimensionality reduced to 2-D after PCA process

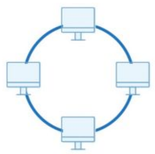
Network Topologies

In these distributed algorithms, fully connected mesh consists of multiple identical nodes that individually calculate their vector updates and communicates them with all other nodes.



Fully Connected Mesh

In a partially connected mesh network, nodes can share their vector updates with only the other nodes that are directly connected to it.



Partially Connected Mesh

Krasulina's Method: Stochastic, Iterative, streaming PCA algorithm that operates on one data sample at a time to find the top PC. Runtime in streaming setting is O(d/ε), where ε is the desired error.
Distributed Streaming Krasulina (DiSK): Same as Krasulina's method, but performs k-PCA to find top k eigenvectors

Methodology

Implemented in Fully Connected Mesh Topology:

D-Krasulina: Updates the top principal component estimate using a single data point at a time.
DM-Krasulina: Same as D-Krasulina, but uses a mini-batch of data samples at each step.
DiSK: Updates the k top principal components estimate using a mini-batch of data samples at each step.

Implemented in Partially Connected Mesh Topology:

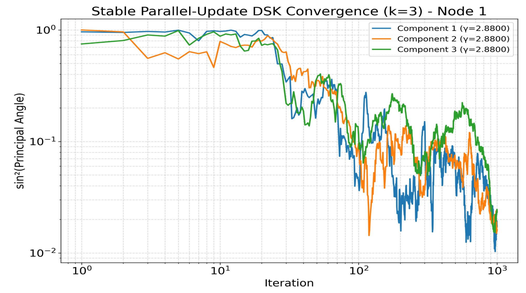
C-DIEGO: Nodes compute local top-1 PCA updates and average with neighbors to reach consensus, with no central server.

Experimentation

Synchronous	Asynchronous
All nodes update and communicate in coordinated steps.	Nodes update and communicate independently without waiting for others

Network Parameter Exploration: Conducted experiments by varying the network topology, communication frequency, and latency to assess the impact on convergence.

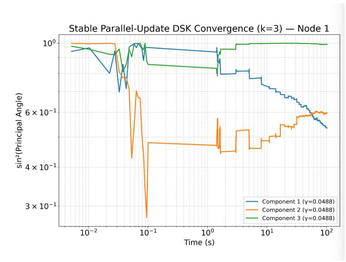
Results

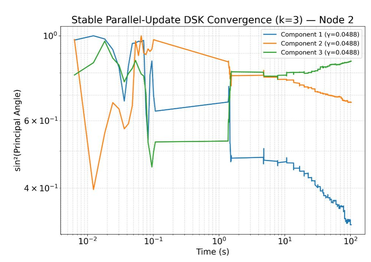


This graph depicts the convergence of the top k Principal Components for the DiSK algorithm with 4 nodes on ORBIT using synthetic data with 100 dimensions.

The algorithm is very sensitive to the number of dimensions and the learning rate (step size), which falls as 1/t, where t is the time iteration.

As the number of dimensions is scaled up, error fluctuates frequently, even as the learning rate falls as an inverse function of time. If the underlying dataset has less low-level isotropic noise, the variance captured by the top *k* eigenectors is smaller, leading to easier convergence.





By slowing down the network, we find an algorithmic breakdown of the asynchronous system. If more than 14 iterations passing before 1 update is sent in the network, consensus averaging cannot lead to an estimate.

Conclusion & Special Thanks

We have confirmed the theoretical convergence rates of D & DM-Krasulina, DiSK and C-DIEGO through empirical analysis. We broke algorithmic constraints to match real-world constraints. We plan to find the best topological configurations for the DiSK algorithm.

We thank Professors Waheed Bajwa and Anand Sarwate, and Muhammad Zulqarnain and for their undying support.

WINLAB

Real-time, Robust, and Reliable (R3) Machine Learning Over Wireless Networks: Learning to Help (L2H)

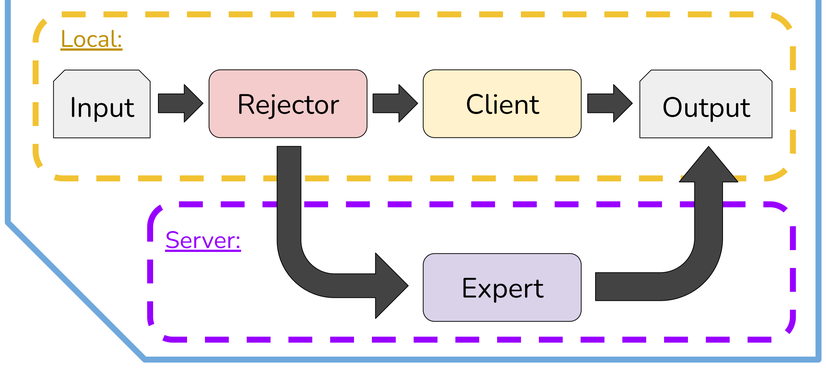
RUTGERS
WINLAB | Wireless Information Network Laboratory

Nihal Abdul Muneer (GR), Joshua Menezes (GR), Ayaan Qayyum (GR), Hasan Ali (UG), Madhav Subramaniyam (UG)

Introduction

Goal:
Implement the L2H system on a testbed environment and analyze its performance. Experiment with different parameters and compare results.

Purpose:
For a legacy device, it may have less accurate results from the machine learning model as it becomes more outdated. The user can use a model hosted on a server, but then they need to pay to use it in either money, time, or both. L2H aims to minimize this loss in time and money while not requiring the user to completely overhaul their system. It does this by introducing a rejector model which determines what can be handled locally and what needs to be sent to the server not fully relying on either one.



Models

Artificial Neural Network (ANN):
Used for Rejector model and Client models. A simple neural network chosen for its simplicity and low compute time.

Convolutional Neural Network (CNN):
Used for Rejector, Client, and Expert models. Generally used for expert due to its great performance with images compared to an ANN for both the client and the rejector.

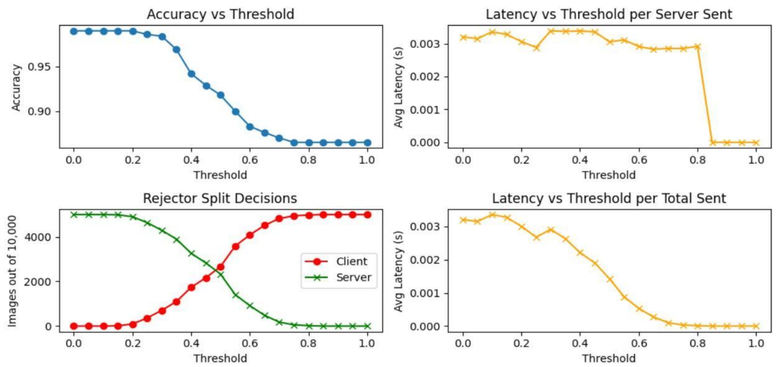
Vision Transformer (ViT):
Used for Client and Expert model. Larger more advanced model used when attempted to scale the problem with a real world scenario, where the models used are larger and the difference between the Client, Rejector, and Expert is significant.

Variational Autoencoder (VAE):
Attempted to train on server side as an attempt to adapt current L2H implementation so expert converts image for client to one it can understand better as opposed to predicting class itself

Multi-Armed Bandit (MAB) and Reinforcement Learning (Markov Decision Process):
Pioneered in an attempt to explore the ability to convert the current L2H implementation to an online setting, constantly learning from its environment and either adjusting the model or the threshold used for the L2H rejector.

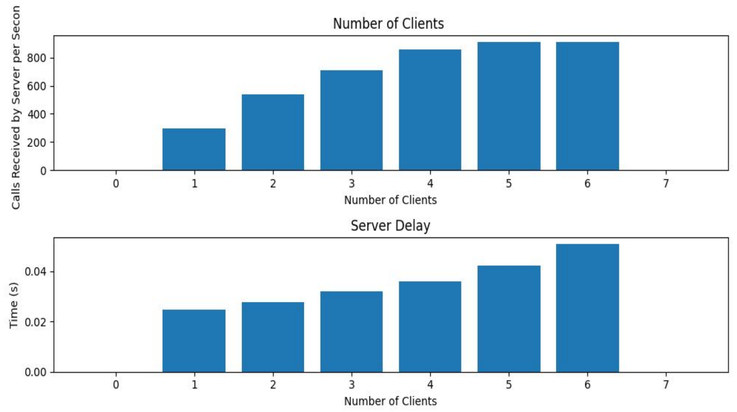
Results

One Client One Expert:



This experiment was run on MNIST with an ANN client and rejector model and a CNN Expert model. The rejector chooses the client more when threshold is 1 and chooses the expert more when threshold is 0. As you choose the expert more, the accuracy is higher which is good but the latency is also higher which is bad. As you transition towards using the client model, it shifts to having better latency and worse accuracy.

Many Clients One Expert:



This experiment was run on MNIST with an ANN client and rejector model and a CNN Expert model. The code was adjusted to handle a greater number of clients connected to a single expert. Each client utilizes 8 ports sending 8 times the requests. As you increase the number of clients the calls per second increases but caps out at a maximum value. As you add more clients to the expert, the latency increases but non linearly. This implies that an increase in clients causes a much larger latency issue the more client there are.

Datasets

MNIST:
- 28x28 grayscale images of handwritten digits (10 classes)
- 60,000 training images and 10,000 testing images

TINY-IMAGE-NET:
- 64x64 color images of 200 classes (Subset of imagenet)
- 100,000 training images and 10,000 testing images

CIFAR-100:
- 32x32 color images of 100 classes (grouped in 20)
- 50,000 training images and 10,000 testing images

CIFAR-10:
- 32x32 color images of 10 classes
- 50,000 training images and 10,000 testing images

Conclusion & Future Work

Conclusion:
The L2H system works in a real world testing set up and not just theoretically. The system was able to effectively trade accuracy for latency by choosing between a client and expert model. Additionally, through testing with many clients to a single expert, we were able to discover new possibilities for how to improve the system noticing that there is an exponential increase in latency when you linearly increase the number of clients.

Future Work::
Exploring how to mitigate the dramatic latency increase as you add more clients would be a substantial research question for determining the effectiveness of this system in a real world environment. Additionally, testing how having different clients of different levels of effectiveness affects the experts ability to help would be useful in considering a real world system of many generations of devices. Lastly, scaling the L2H up would likely give better insight into how the system could perform in a real world scenario.

Special Thanks

We thank Professors Waheed Bajwa and Anand Sarwate, and Yu Wu, Xiaotian Zhou, and Nitya Sathyavageeswaran for their support and insight in completing this project.

WINLAB