GPU Accelerated Machine Learning for Bond Price Prediction

Venkat Bala Rafael Nicolas Fermin Cota

R N F C

Motivation

Primary Goals

- Demonstrate potential benefits of using GPUs over CPUs for machine learning
- Exploit inherent parallelism to improve model performance
- Real world application using a bond trade dataset

Highlights

Ensemble

- **Bagging**: Train independent regressors on equal sized bags of samples
- Generally, performance is superior to any single individual regressor
- **Scalable**: Each individual model can be trained independently and in parallel

Hardware Specifications

- CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
- GPU: GeForce GTX 1080 Ti
- RAM : 1 TB (DDR4 2400 MHZ)

Bond Trade Dataset

Feature Set

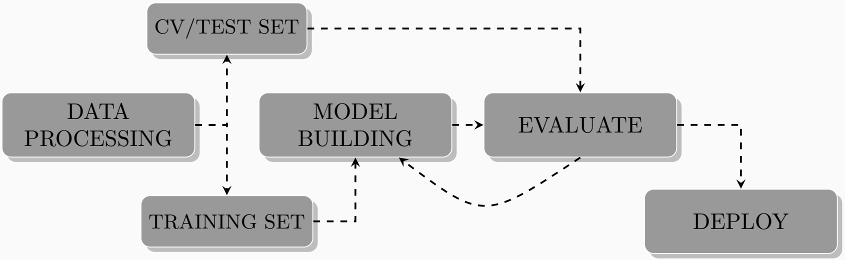
- 100+ features per trade
  - Trade Size/Historical Features
  - Coupon Rate/Time to Maturity
  - Bond Rating
  - Trade Type: Buy/Sell
  - Reporting Delays
  - Current Yield/Yield To Maturity

Response

- Trade Price

Modeling Approach

The Machine Learning Pipeline



Accelerate each stage in the pipeline for maximum performance

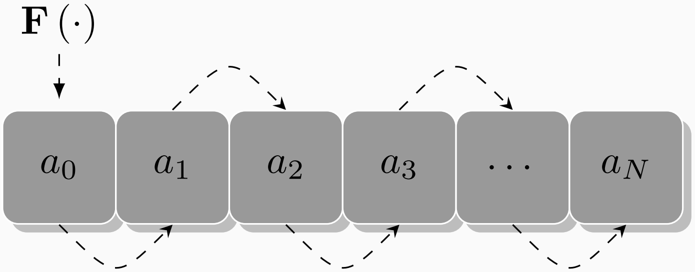
Data Preprocessing

Exposing Data Parallelism

- Important stage in the pipeline (**Garbage In** $\rightarrow$ **Garbage out**)
- Many models rely on input data being on the same scale
- Standardization, log transformations, imputations, polynomial/non-linear feature generation, etc.
- Most cases, no data dependence so each operation can be executed independently
- Significant speedups can be obtained using GPUs, given sufficient data/computation

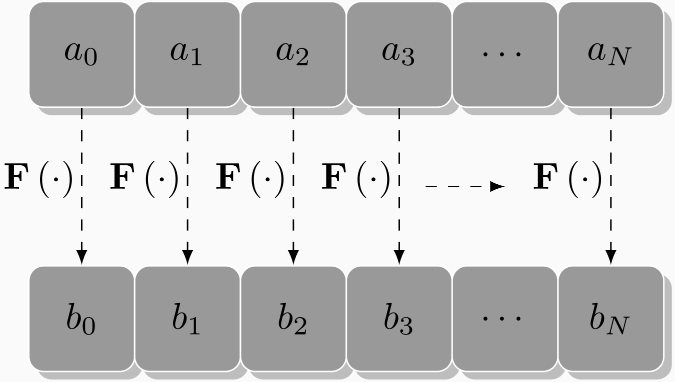
Data Preprocessing: Sequential Approach

Apply function $\mathsf{F}(\cdot)$ sequentially to each element in a feature column



Data Preprocessing: Parallel Approach

Apply function $\mathsf{F}(\cdot)$ in parallel to each element in a feature column

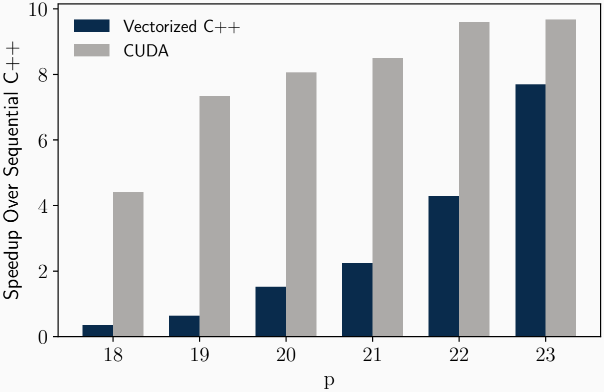


Implementation Basics

- Task is **embarrassingly parallel**
- Improve CPU code performance
    - Auto vectorizations + compiler optimizations
    - Using performance libraries (Intel MKL)
    - Adopting Threaded (OpenMP)/Distributed computing (MPI) approaches
- Great application case for GPUs
    - Offload computations onto the GPU via CUDA kernels
    - Launch as many threads as there are data elements
    - Launch several kernels concurrently using CUDA streams

Toy Example: Speedup Over Sequential C++

- Log transformation of an array of floats
- $N = 2^p$, Number of elements, $p = \log_2(N)$



Bond Dataset Preprocessing

Applied Transformations

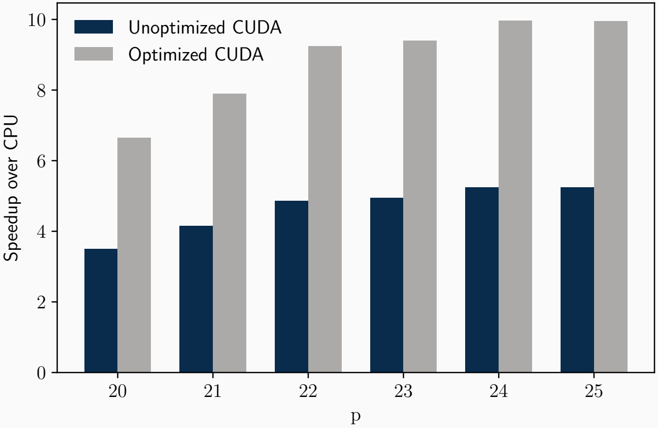
- Log transformation of highly skewed features (Trade Size, Time to Maturity)
- Standardization (Trade Price & historical prices)
- Missing value imputation
- Winsorizing features to handle outliers
- Feature generation (Price differences, Yield measurements)

Implementation Details

- CPU: C++ implementation using Intel MKL/Armadillo
- GPU: CUDA

GPU Speedup over CPU implementation

- Nearly 10x speedup obtained after CUDA optimizations



CUDA Optimizations

Standard Tricks

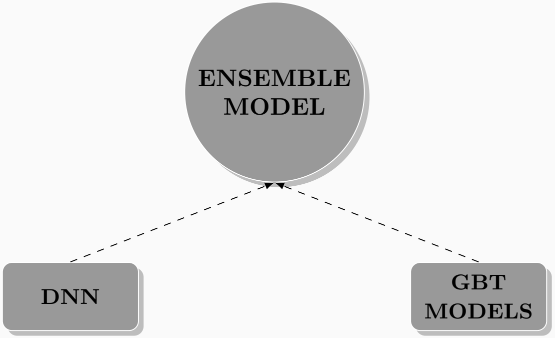
- Concurrent kernel executions of kernels using CUDA streams to maximizing GPU utilization
- Use of optimized libraries such as cuBLAS/Thrust
- Coalesced memory access
- Maximizing memory bandwidth for low arithmetic intensive operations
- Caching using GPU shared memory

Model Building

Ensemble Model

Model Choices

- **GBT**: XGBoost, **DNN**: Tensorflow/Keras



Hyperparameter Tuning: Hyperopt

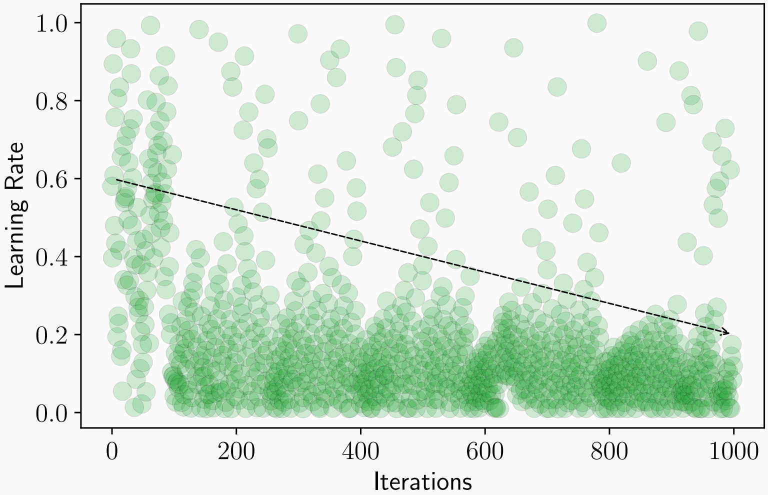
GBT: XGBoost

- Learning Rate
- Max depth
- Minimum child weight
- Subsample, Colsample-bytree
- Regularization parameters

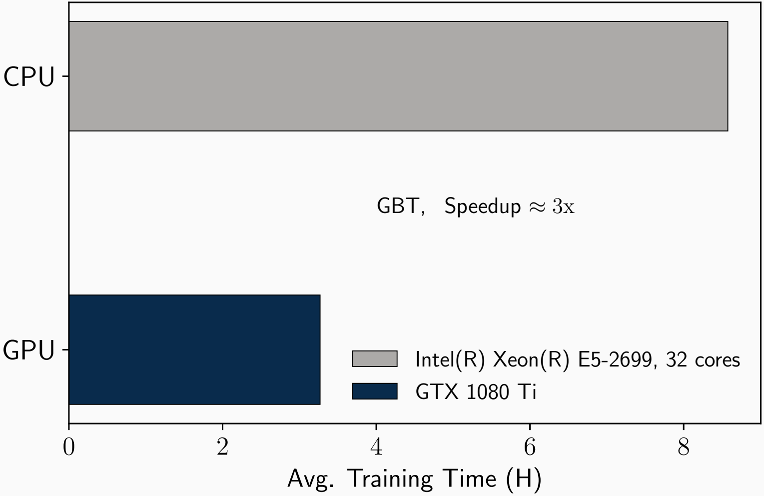
DNN: MLPs

- Learning Rate/Decay Rate
- Batch Size
- Epochs
- Hidden layers/Layer width
- Activations/Dropouts

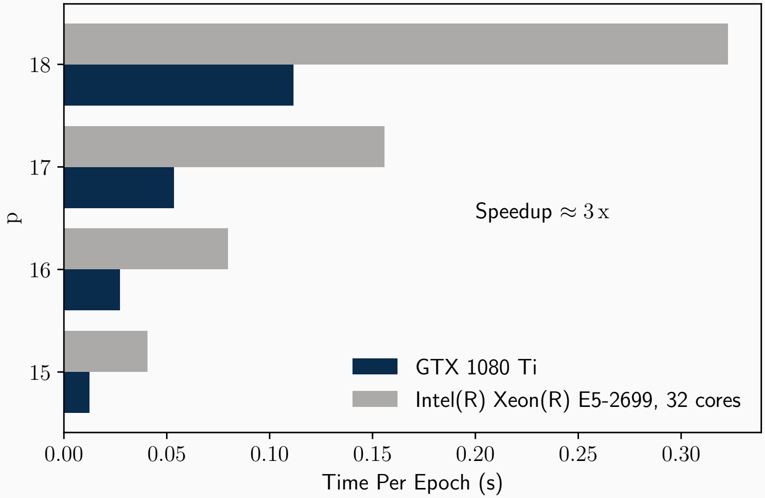
Hyperparameters Tuning: Hyperopt



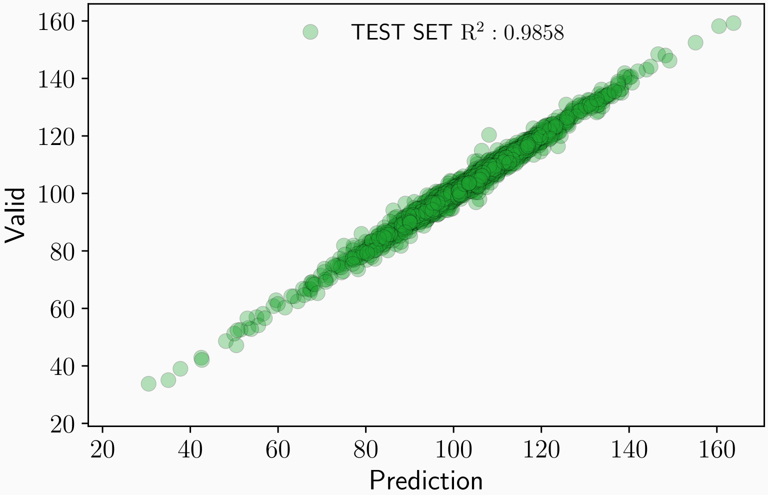
XGBoost: Training & Hyperparameter Optimization Time



TensorFlow/Keras Time Per Epoch



Model Test Set Performance



Summary

Summary

Final Remarks

- Leveraging the GPU computation power $\rightarrow$ dramatic speedups
- Maximum performance when GPUs incorporated into every stage of the pipeline
- Ensembles: Bagging/Boosting to improve model accuracy/throughput
- Shorter training times allows more experimentation
- Extensive support available
- **Deploy this pipeline now in our in-house DGX-1**

Questions?