- Assignment 1 Grading: In progress
- Assignment 2 due on Thu!
- Course Project: Introduction due Oct 17th
Bismarck → logistic Regression

Supervised learning, Unified Interface
Shared memory, Model fits in memory

Parameter Server

Large datasets, large models (PB scale)
Shard data, parameters

Consistency model
Fault tolerance
WHAT IS DEEP LEARNING?
Optimization

\[
\min_{w \in \mathbb{R}^d} \sum_{i=1}^{N} f(w, z_i) + P(w)
\]
DEEP LEARNING

ReLU

\[
\text{ReLU} \quad \max \left( \sum x_i w_i, 0 \right)
\]

Non-linearity
MODEL TRAINING

\[ w^{(k+1)} = w^{(k)} - \alpha_k \nabla f(w^{(k)}) \]

Initialize \( w \)

For many iterations:

- Compute Gradient
- Update model

End

Stochastic Gradient Descent

Gradient using backprop
Computes Intensive!
DIST BELIEF SHORTCOMINGS

- Written in C++, hard to experiment
- Writing new types of layers
- New optimization methods
- Execution pattern fixed
  - Push, Pull
  "Flexibility" in how work was partitioned
  and how comm. happened
TensorFlow: Design Principles

- Dataflow graphs of primitive operators
- Deferred execution: Symbolic dataflow graph
- Heterogeneous accelerators
DATAFLOW GRAPH

Unify computation and state management

State

Vertices
- Operations ≈ Computation
- State associated = Mutable

Edges
- Tensors

Unify computation and state management
EXECUTION MODEL

Multiple concurrent executions of overlapping sub-graphs

Vertices have mutable state; shared between executions
DATAFLOW GRAPH ELEMENTS

Edges: Tensors
- N-dimensional arrays, dense representation by default
- Operations take in tensors and return tensors

Vertices: Operations
- Tensor \( \rightarrow \) Tensor functions

Stateful Operations
- Variables
- Queues
PARTIAL EXECUTION

Concurrent overlapping
Subgraph
Queue -> FIFO priority /Shuffle

Dataflow graph
DFS

Parameters
Read params
Apply grads
Fwd
Back
Training

Periodic checkpoint

Dist. FS
EXECUTION MODES

Partial Execution
- Input batches from queue
- Concurrent training steps
- Shared model
- “Horizontal” parallelism?

Distributed Execution
- Operations placed on devices
- Account for colocation
- Manual placement decisions?
- Send-Recv to stitch subgraphs
CONTROL FLOW

- Support for RNNs, LSTMs
- Switch and Merge operators to support conditionals
- Enter, Exit, NextIteration to support while loops

```python
input = ...  # A sequence of tensors
state = 0    # Initial state
w = ...      # Trainable weights

for i in range(len(input)):
    state, out[i] = f(state, w, input[i])
```
EXTENSIONS

Automatic Differentiation
- Given a symbolic expression, generate its gradient
- Also extend to control flow operations

Fault Tolerance
- User-level checkpointing
- Save and Restore operations in graph
- Not necessarily consistent?
SYNC VS ASYNC

(a) Asynchronous replication
(b) Synchronous replication
(c) Synchronous w/ backup worker

Worker 1
Worker 2
Worker 3

Stragglers =

Fastest 2 out of 3
3 workers
≤ 48 out of 50
IMPLEMENTATION

Training libraries
Python client
C API
Distributed master
Kernel implementations
Networking layer

Inference libs
C++ client
Dataflow executor
ReLU Queue

Heterogeneous devices

CPU GPU
DISCUSSION

https://forms.gle/L9oA69DQe2a7yg3CA
How is the dataflow graph used in Tensorflow similar/different from Apache Spark? What are the implications of that?

Tensorflow:
- Vertices: Operations, State
- Edges: Tensors
- Multiple subgraphs
- Parallelism within a step
- Vertex resides in 1 node

Apache Spark:
- Input to RDDs
- Map and Reduce
- Entire cluster checkpointed
- RDDs are partitioned, can do partial recovery
What are some shortcomings of the programming model used in Tensorflow? What could be some ways to improve it?

- TF graph is static
- Placement is low-level, not automatic
Next class: Ray
Assignment 2 due this week!
Course project