BLUE WATERS

SUSTAINED PETASCALE COMPUTING

9/11/2019

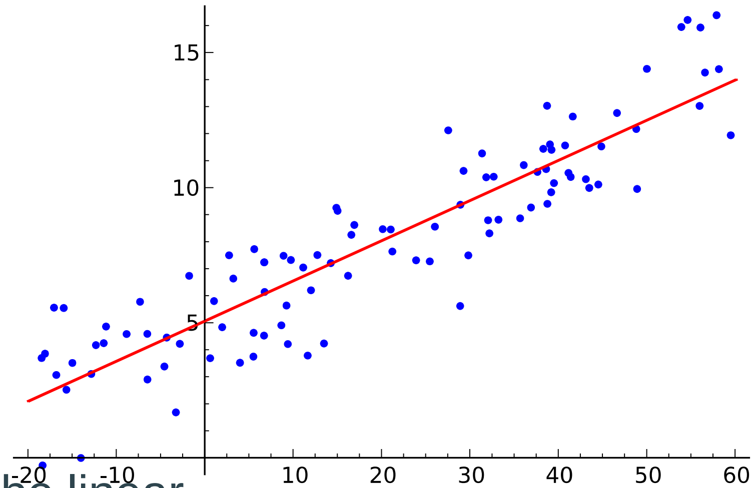
Distributed Training on HPC

Presented By: Aaron D. Saxton, PhD

GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION

Statistics Review

- Simple $y = m \cdot x + b$ regression
  - Least Squares to find m,b
  - With data set $\{(x_i, y_i)\}_{i=1,\ldots,n}$
  - Let the error be
    - $R = \sum_{i=1}^{n}[(y_i - (m \cdot x_i + b)]^2$
  - Minimize R with respect to m and b.
    - Simultaneously Solve
      - $R_m(m, b) = 0$
      - $R_b(m, b) = 0$
    - Linear System
- We will consider more general $y = f(x)$
  - $R_m(m, b) = 0$ and $R_b(m, b) = 0$ may not be linear

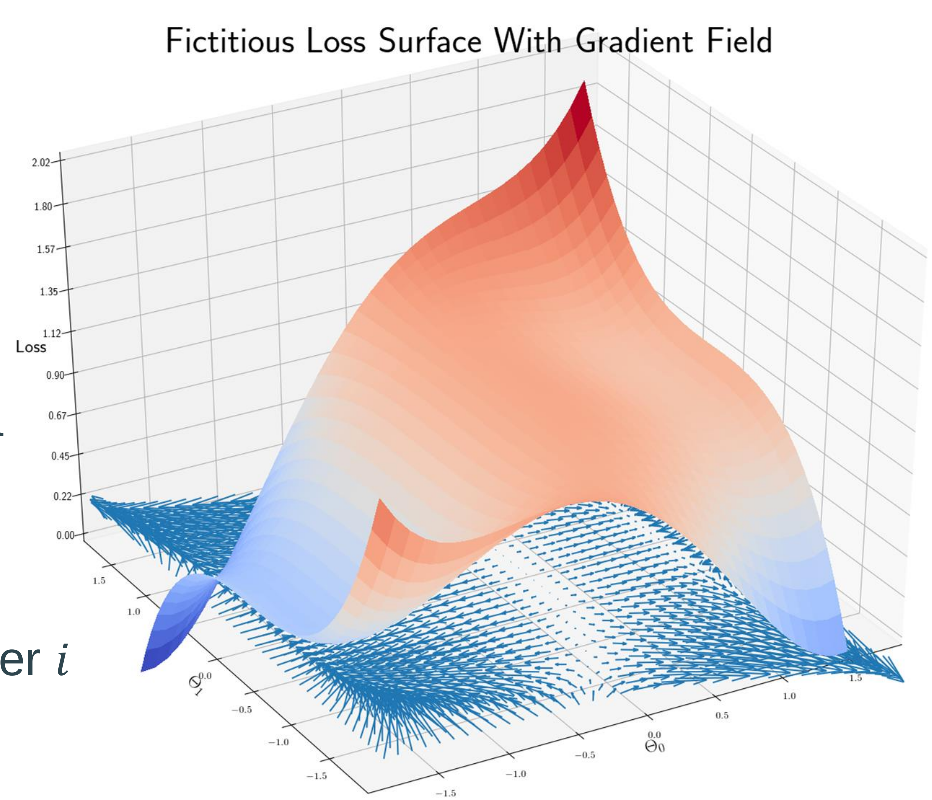


Statistics Review

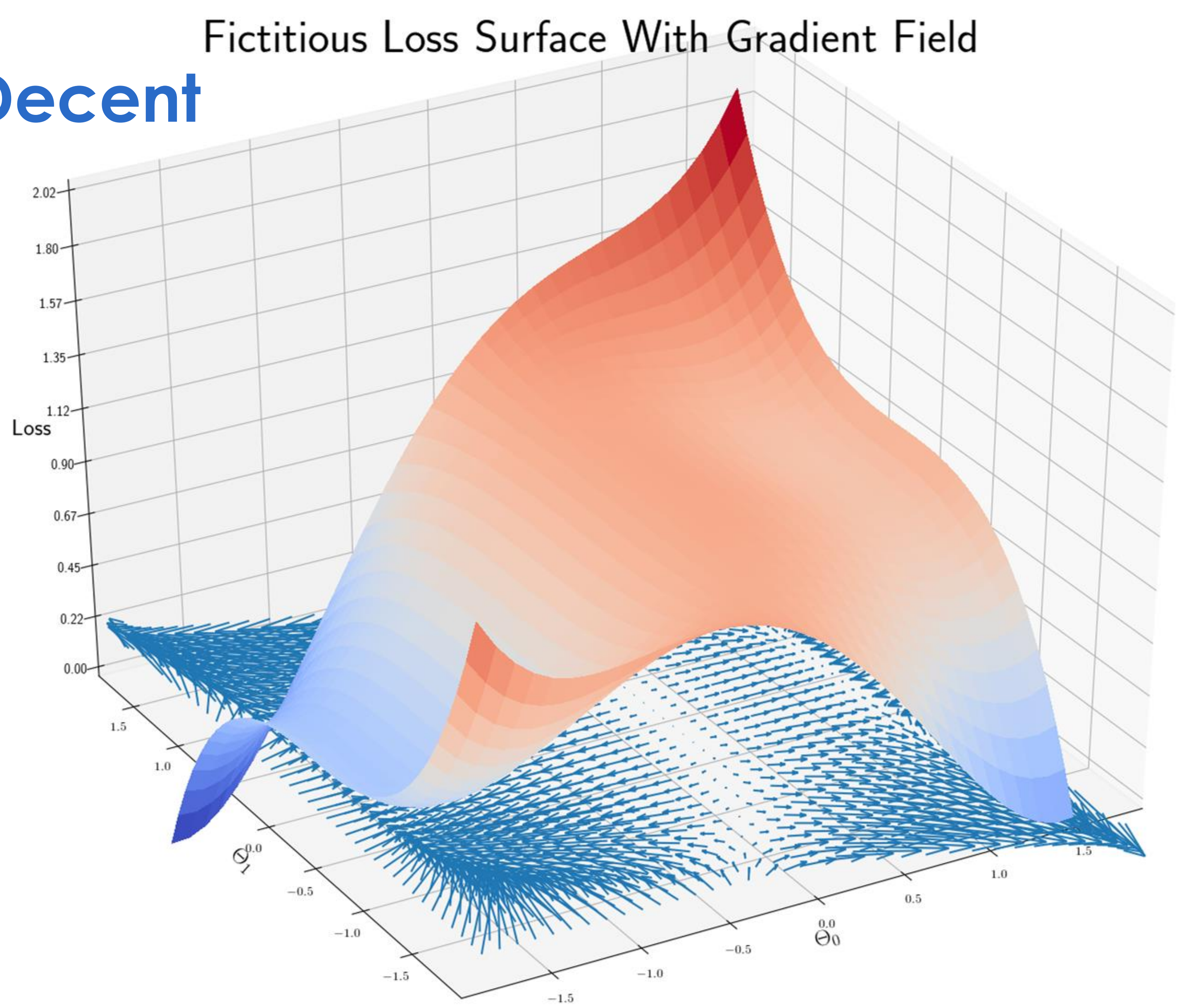
- Regressions with parameterized sets of functions. e.g.
  - $y = ax^2 + bx + c$ (quadratic)
  - $y = \sum a_i \ x^i$ (polynomial)
  - $y = Ne^{rx}$(exponential)
  - $y = \frac{1}{1+e^{-(a+bx)}}$ (logistic)

Gradient Decent

- Searching for minimum
- $\nabla R = \langle R_{\theta_0}, R_{\theta_2}, \ldots, R_{\theta_n} \rangle$
- $R(\vec{\theta}_{t+1}) = R(\vec{\theta}_t + \gamma \nabla R)$
- γ: Learning Rate
- Recall, Loss depends on data
  Expand notation,
    - $R(\vec{\theta}_t; \{(x_i, y_i)\}_n)$
    - Recall R and ∇R is a sum over i
- Want R with ALL DATA ….. ?
    - $(R = \sum_{i=1}^{n} [(y_i - f_{\theta_t}(x_i)]^2)$



Gradient Decent



Stochastic Gradient Decent

- Recall R is a sum over i ($R = \sum_{i=1}^{n}[(y_i - f_{\theta_t}(x_i)]^2$)
- Single training example, (x_i, y_i), Sum over only one training example
- $\nabla R_{(x_i,y_i)} = \langle R_{\theta_0}, R_{\theta_2}, \ldots, R_{\theta_n} \rangle_{(x_i,y_i)}$
- $R_{(x_i,y_i)}(\vec{\theta}_{t+1}) = R_{(x_i,y_i)}(\vec{\theta}_t + \gamma \nabla R_{(x_i,y_i)})$
- γ: Learning Rate
- Choose next (x_{i+1}, y_{i+1}), (Shuffled training set)

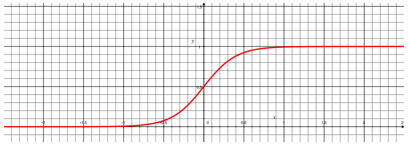
- SGD with mini batches
- Many training example, (x_i, y_i), Sum over many training example
  - Batch Size or Mini Batch Size (This gets ambiguous with distributed training)
- SGD often outperforms traditional GD, want small batches.
  - https://arxiv.org/abs/1609.04836, On Large-Batch Training … Sharp Minima
  - https://arxiv.org/abs/1711.04325, Extremely Large ... in 15 Minutes
- Optimization Methods for Large-Scale Machine Learning
  - https://epubs.siam.org/doi/pdf/10.1137/16M1080173

Neural Networks

- Activation functions

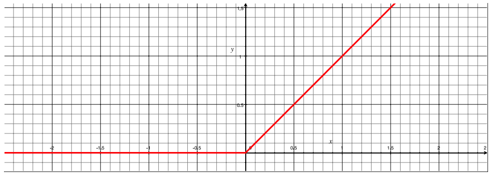
Logistic

$\sigma(x) =$



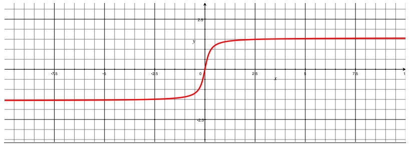
ReLU (Rectified Linear Unit)

$\sigma(x) =$



Arctan

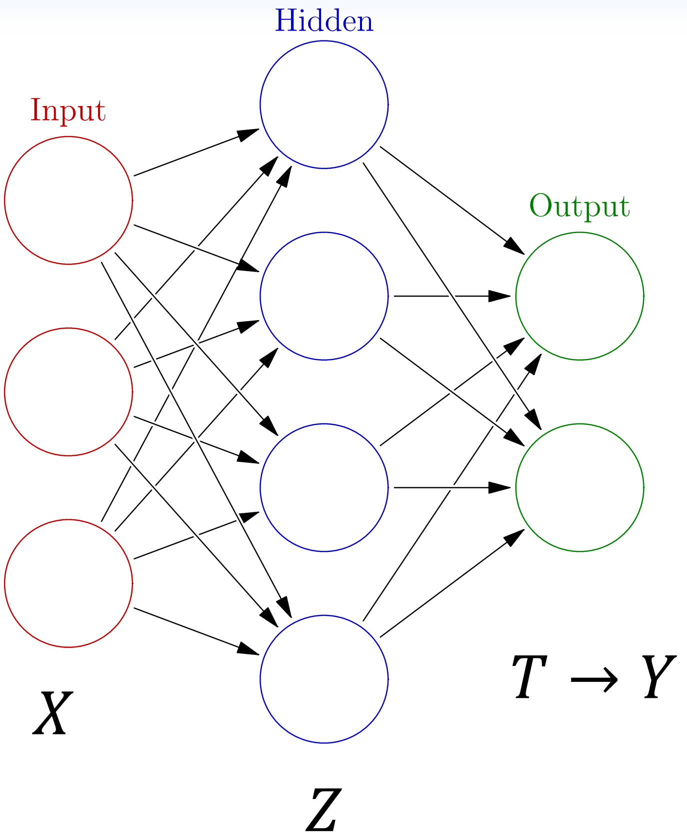
$\sigma(x) =$



- Softmax
  - $g_k(x_1, x_2, \ldots, x_N) = \frac{e^{x_k}}{\sum e^{x_i}}$

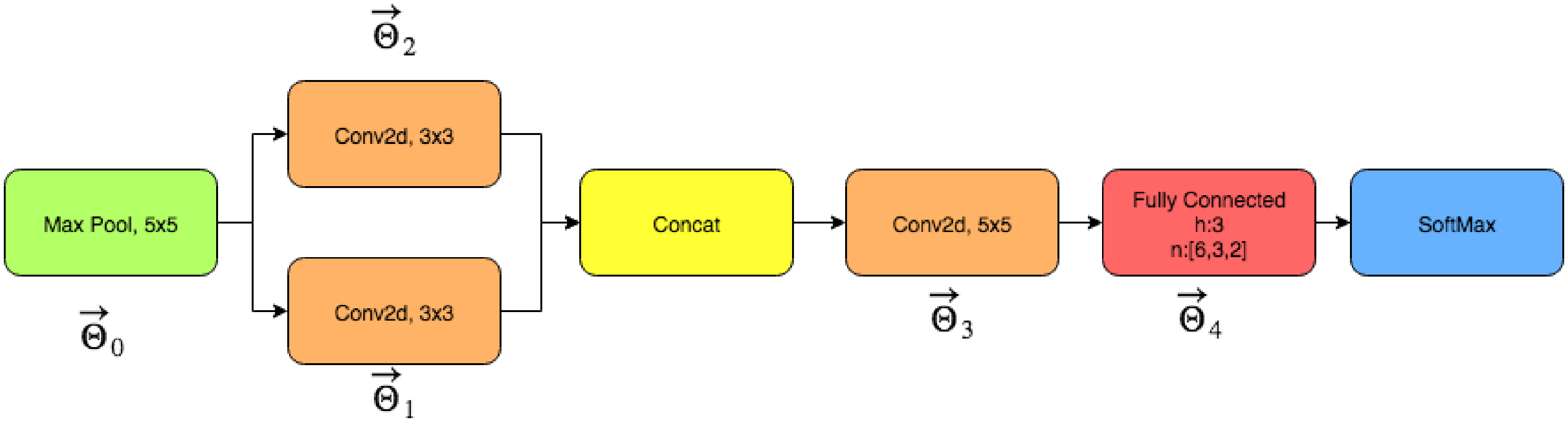
Neural Networks

- Parameterized function
  - $Z_M = \sigma(\alpha_{0m} + \alpha_m X)$
  - $T_K = \beta_{0k} + \beta_k Z$
  - $f_K(X) = g_k(T)$
- Linear Transformations with bias (Affine?) and pointwise evaluation of nonlinear function, σ



- $\beta_{0i}, \beta_i, \alpha_{0m}, \alpha_m$
  - Weights to be optimized

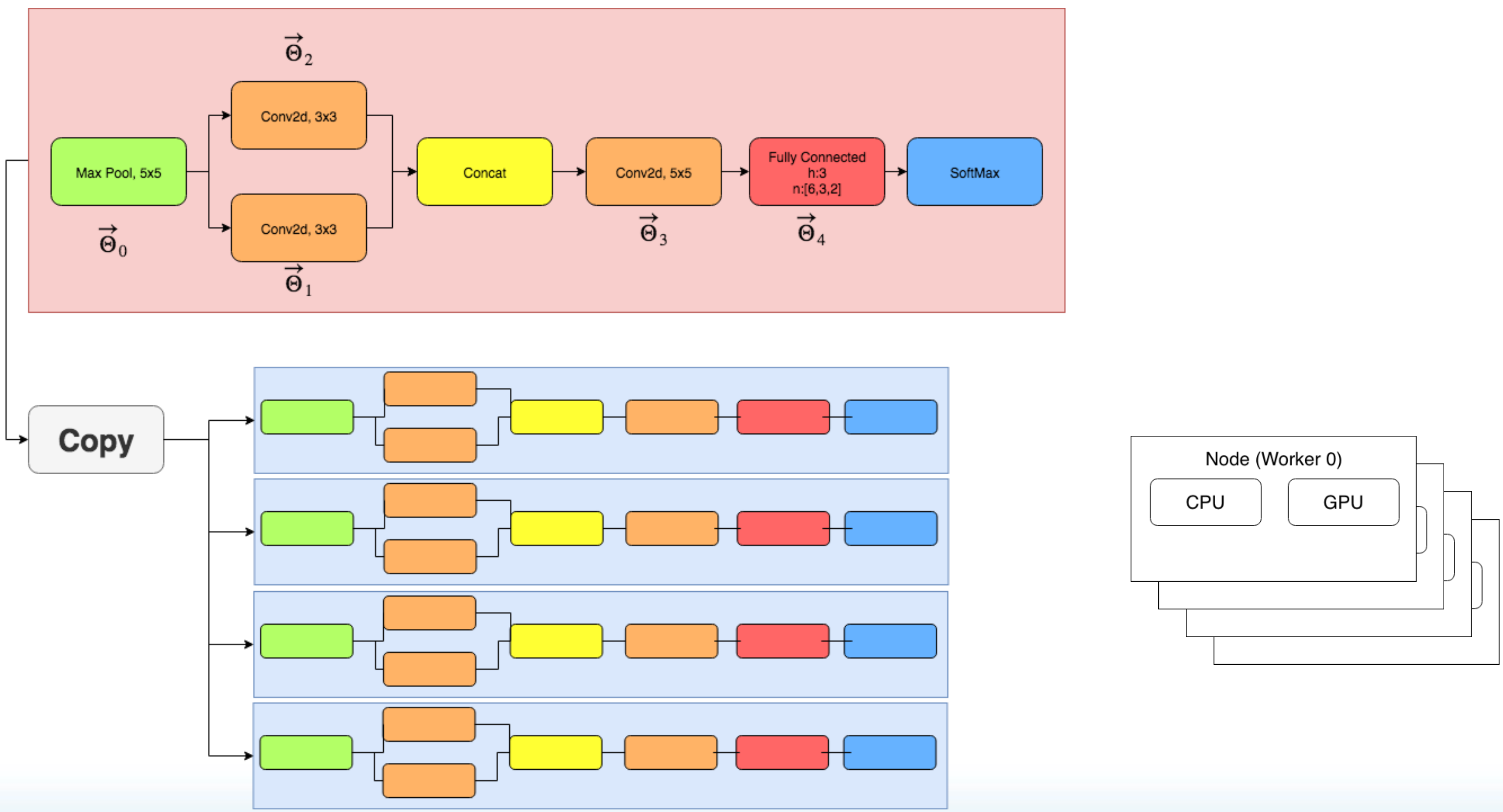
Faux Model Example



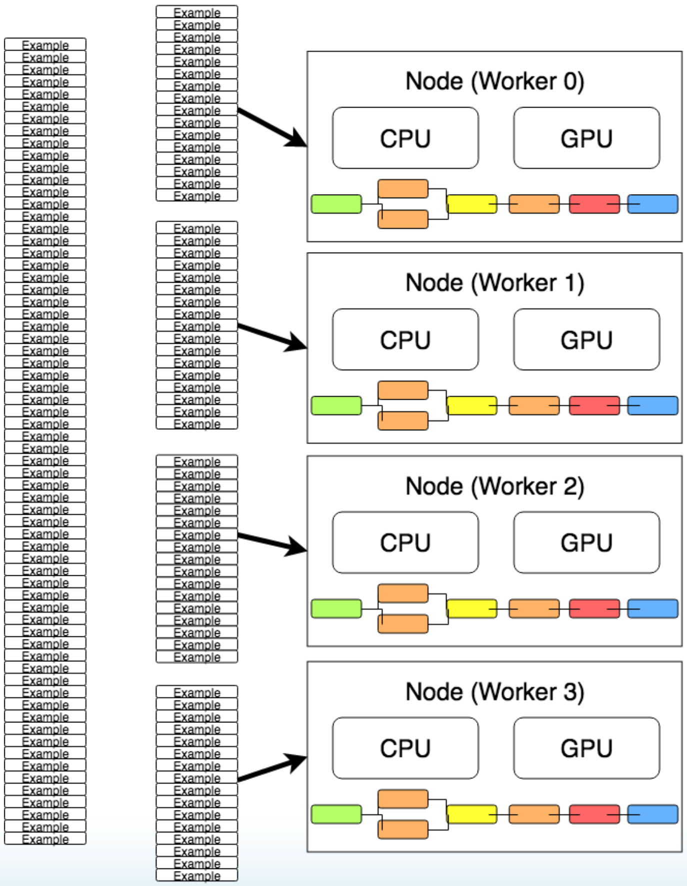
Trainable Weights

$$\left\{ \vec{\Theta}_i : i \in [0, 1, 2, 3, 4] \right\}$$

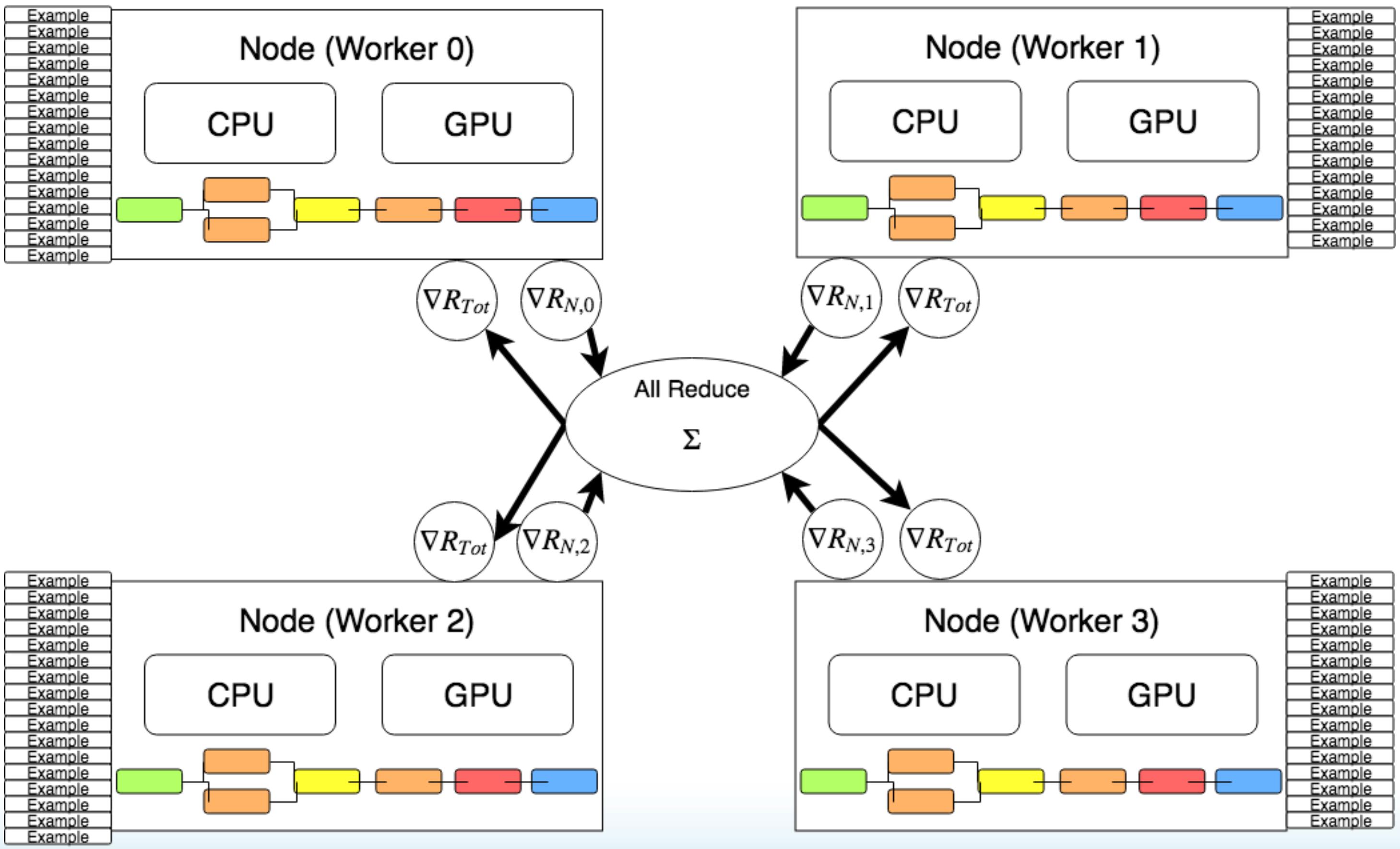
Distributed Training, data distributed



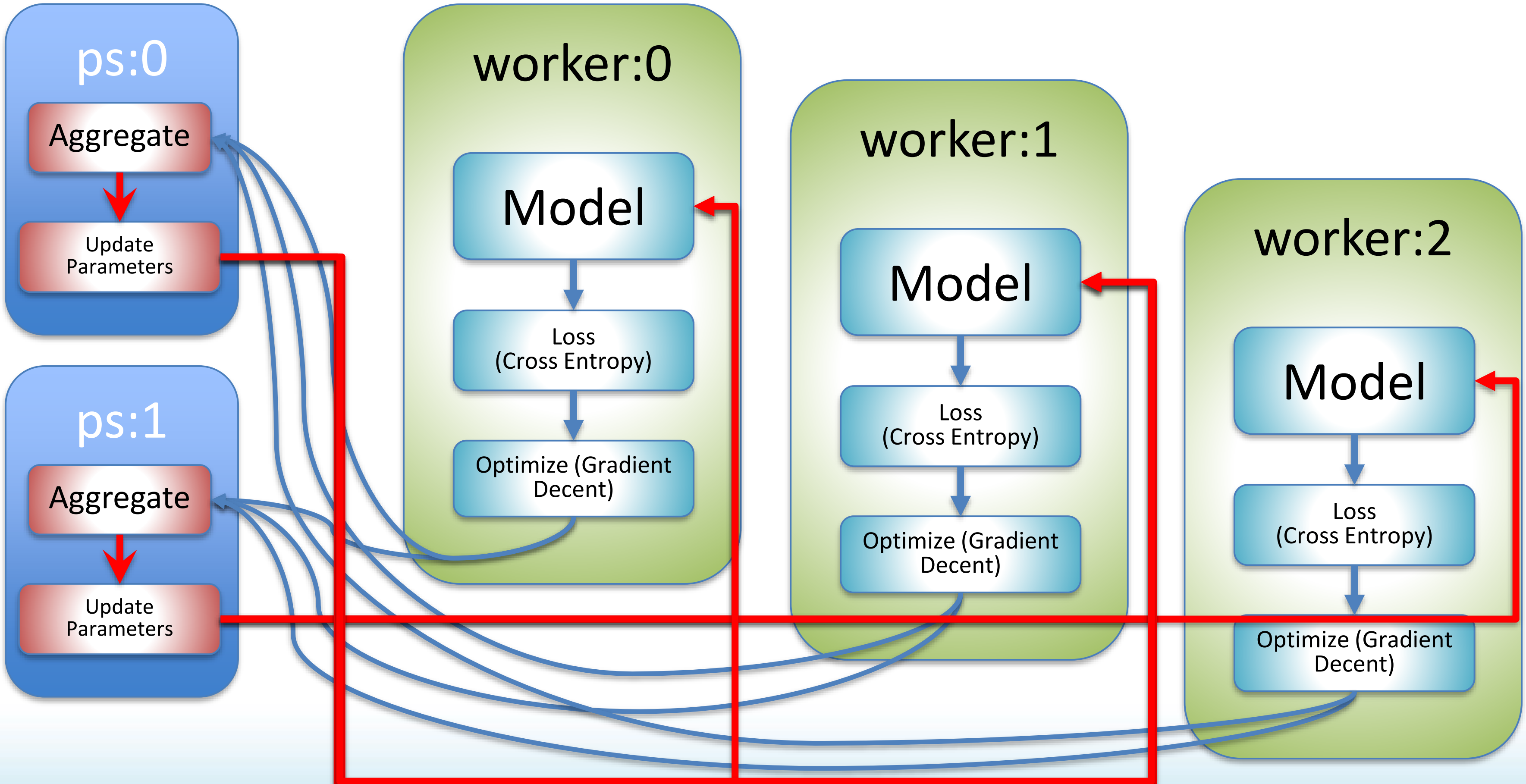
Distributed Training, data distributed



Distributed Training, All Reduce Collective



Distributed TensorFlow: Parameter Sever/Worker Default, Bad Way on HPC



Practical Implementations

- Native Tensorflow
- Native PyTorch
- Horovod
- Cray ML Plugin

Practical Implementations: Native Tensorflow

- Parameter Servers-Workers

```
cluster = tf.train.ClusterSpec({'ps': tf_ps_hosts_ports,
                                'worker': tf_worker_hosts_ports})


server = tf.train.Server(cluster, job_name=job_name,
                         task_index=tf_index,
                         protocol=FLAGS.server_protocol)
```

Build Data, Model, and Training Somewhere Here

```
with tf.train.MonitoredTrainingSession(master=server.target,
                                       is_chief=(tf_index == 0),
                                       checkpoint_dir=FLAGS.checkpoint_dir,
                                       #save_summaries_secs=1800,
                                       save_summaries_steps=PRINT_SUMMERY_EVERY,
                                       config=config,
                                       hooks=hooks) as mon_sess:
  print("worker %s: In MonitoredTrainingSession() context" % tf_index)
  tf.train.start_queue_runners(sess=mon_sess)
```

Practical Implementations: Native PyTorch

- Native MPI tensor serialization!

```
import torch.distributed as dist

dist.init_process_group('mpi')
num_workers = dist.get_world_size()
rank = dist.get_rank()
```

Build Data, Model, and Training Somewhere Here

```
for param in model.parameters():
    if param is not None:
        dist.all_reduce(param.data)
for param in model.parameters():
    if param is not None:
        param /= float(num_workers)
```

Practical Implementations: Horovod

- MPI Wrapper Around Tensorflow
  - https://github.com/horovod/horovod

```
import tensorflow as tf
import horovod.tensorflow as hvd


# Initialize Horovod
hvd.init()

# Pin GPU to be used to process local rank (one GPU per process)
config = tf.ConfigProto()
config.gpu_options.visible_device_list = str(hvd.local_rank())

# Build model...
loss = ...
opt = tf.train.AdagradOptimizer(0.01 * hvd.size())

# Add Horovod Distributed Optimizer
opt = hvd.DistributedOptimizer(opt)

# Add hook to broadcast variables from rank 0 to all other processes dur
# initialization.
hooks = [hvd.BroadcastGlobalVariablesHook(0)]

# Make training operation
train_op = opt.minimize(loss)

# Save checkpoints only on worker 0 to prevent other workers from corrup
checkpoint_dir = '/tmp/train_logs' if hvd.rank() == 0 else None

# The MonitoredTrainingSession takes care of session initialization,
# restoring from a checkpoint, saving to a checkpoint, and closing when
# or an error occurs.
with tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,
                                       config=config,
                                       hooks=hooks) as mon_sess:
  while not mon_sess.should_stop():
    # Perform synchronous training.
    mon_sess.run(train_op)
```

Practical Implementations: Cray ML Plugin

- Cray Optimized MPI Tensor serilization
  - Runs concurrently with standard Tesnorflow

```
import ml_comm as mc
```

Build Data, Model, and Training Somewhere Here

```
tot_model_size = sum([reduce(lambda x, y : x*y, v.get_shape().as_list()) for v in tf.trainable_variables()])
mc.init(1, 1, tot_model_size, "tensorflow")

mc.config_team(0,0,100, FLAGS.num_steps, 2, 1)

class BcastTensors(tf.train.SessionRunHook):
  def __init__(self):
    self.bcast = None

  def begin(self):
    new_vars = mc.broadcast(tf.trainable_variables(), 0)
    self.bcast = tf.group(*[tf.assign(v, new_vars[k]) for k, v  in enumerate(tf.trainable_variables())])
grads_and_vars = optimizer.compute_gradients(total_loss)
grads = mc.gradients([gv[0] for gv in grads_and_vars], 0)
gs_and_vs = [(g,v) for (_,v), g in zip(grads_and_vars, grads)]

train_op = optimizer.apply_gradients(gs_and_vs, global_step=global_step)
hooks = [tf.train.StopAtStepHook(last_step=FLAGS.num_steps), BcastTensors()]
```

```
with tf.train.MonitoredTrainingSession(checkpoint_dir=FLAGS.checkpoint_dir,
                                       save_summaries_steps=20,
                                       save_checkpoint_secs=120,
                                       config=config,
                                       hooks=hooks) as mon_sess:
  print("worker %s: In MonitoredTrainingSession() context" % rank)
  tf.train.start_queue_runners(sess=mon_sess)
```

Other models: Sequence Modeling

- Autoregression

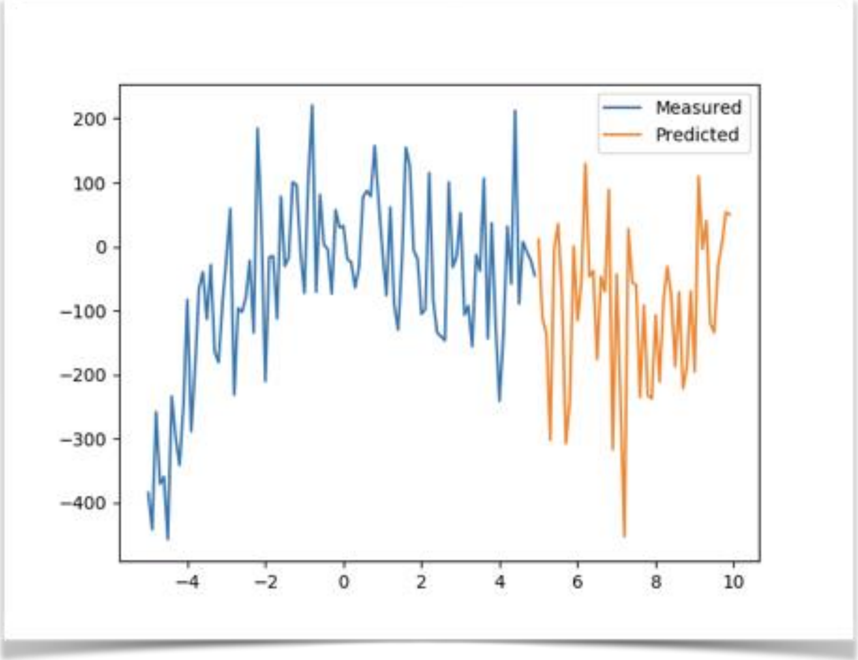
$$X_t = c + \sum_{i=1}^{p} \phi_i B^i X_t + \epsilon_t$$

$\textit{Back Shift Operator}: B^i$

- Autocorrelation

$$R_{XX}(t_1, t_2) = E[X_{t_1} \overline{X_{t_2}}]$$

- Other tasks
  - Semantic Labeling



[art.]	[adj.]	[adj.]	[n.]	[v.]	[adverb]	[art.]	[adj.]	[adj.]	[d.o.]
The	quick	red	fox	jumps	over	the	lazy	brown	dog

Recurrent Neural Networks: Sequence Modeling

- Few projects use pure RNNs, this example is only for pedagogy
- RNN is a model that is as “deep” as the modeled sequence is long
- LSTM’s, Gated recurrent unit
- No Model Parallel distributed training on the market (June 2019)

- Attention and Multi Headed Transformers
  - Still have problem staying small and finding long term relationships