Large Scale Distributed Deep Networks: Appendix

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen,
Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato,
Andrew Senior, Paul Tucker, Ke Yang, Andrew Y. Ng
{jeff, gcorrado}@google.com
Google Inc., Mountain View, CA

1 Appendix

For completeness, here we provide pseudocode for the model replica (client) side of Downpour SGD (Algorithm 0.1), and Sandblaster L-BFGS (Algorithm 0.2).

Algorithm 1.1: DOWNPOURSGDCLIENT($\alpha, n_{fetch}, n_{push}$)

procedure STARTASYNCHRONOUSLYFETCHINGPARAMETERS($parameters$)
  $parameters \leftarrow$ GETPARAMETERSFROMPARAMSERVER()

procedure STARTASYNCHRONOUSLYPUSHINGGRADIENTS($accruedgradients$)
  SENDGRADIENTSTOPARAMSERVER($accruedgradients$)
  $accruedgradients \leftarrow 0$

main
  global $parameters, accruedgradients$
  $step \leftarrow 0$
  $accruedgradients \leftarrow 0$
  while $true$
  do
    if $(step \bmod n_{fetch}) == 0$
      then STARTASYNCHRONOUSLYFETCHINGPARAMETERS($parameters$)
    $data \leftarrow$ GETNEXTMINIBATCH()
    $gradient \leftarrow$ COMPUTEGRADIENT($parameters, data$)
    $accruedgradients \leftarrow accruedgradients + gradient$
    $parameters \leftarrow parameters - \alpha * gradient$
    if $(step \bmod n_{push}) == 0$
      then STARTASYNCHRONOUSLYPUSHINGGRADIENTS($accruedgradients$)
    $step \leftarrow step + 1$

Sandblaster is a framework for distributed batch optimization procedures. An essential concept in Sandblaster is decomposing operations into local computation on the DistBelief parameter server. By way of example, suppose we have 1 billion parameters and 10 parameter server shards, so that each shard has 1/10 of the parameters. It is possible to decompose L-BFGS into a sequence of scalar-vector products ($\alpha \times \mathbf{x}$) and vector-vector inner products ($\mathbf{x}^T\mathbf{y}$), where each vector is 1 billion dimensional. If one shard is always responsible for the first 1/10 of every vector used internally in L-BFGS, and a second shard is always responsible for the second 1/10 of every vector, and so on up to the final shard always being responsible for the final 1/10 of every vector, it is possible to show that these scalar-vector and vector-vector operations can all be done in a distributed fashion with very little communication, so that any intermediate vector-valued results are automatically stored in the same distributed fashion, and any intermediate scalar-valued result is communicated to all the shards.

Algorithm 1.2: SANDBLASTERLBFGS()

procedure REPLICA.PROCESSPORTION($portion$)
 if (!$hasParametersForStep$)
  then $parameters \leftarrow$ GETPARAMETERSFROMPARAMSERVER()
 $data \leftarrow$ GETDATAPORTION($portion$)
 $gradient \leftarrow$ COMPUTEGRADIENT($parameters, data$)
 $localAccruedGradients \leftarrow localAccruedGradients + gradient$

procedure PARAMETERSERVER.PERFORMOPERATION($operation$)
 $PerformOperation$

main
 $step \leftarrow 0$
 while $true$
 do
  comment: PS: ParameterServer
  $PS.accruedgradients \leftarrow 0$
  while ($batchProcessed < batchSize$)
   do
    for all ($modelReplicas$)**comment:** Loop is parallel and asynchronous
     do
      if ($modelReplicaAvailable$)
       then
        REPLICA.PROCESSPORTION($modelReplica$)
        $batchProcessed \leftarrow batchProcessed + portion$
      if ($modelReplicaWorkDone$ **and** $timeToSendGradients$)
       then
        SENDGRADIENTS($modelReplica$)
        $PS.accruedGradients \leftarrow PS.accruedGradients + gradient$
  COMPUTELBFGSDIRECTION($PS.Gradients, PS.History, PS.Direction$)
  LINESEARCH($PS.Parameters, PS.Direction$)
  PS.UPDATEPARAMETERS($PS.parameters, PS.accruedGradients$)
  $step \leftarrow step + 1$