Data Parallelism

Data parallelism is the most common type of multinode deep network training. Data parallel models split large datasets onto different machines. Most nodes are workers and have access to a fraction of the total data used to train the network. Each worker node has a complete copy of the model being trained. One node is designated as the supervisor that gathers updated weights from the workers at regular intervals and pushes averaged versions of the weights out to worker nodes. Note that you've already seen a data parallel example in this book; the A3C implementation presented in Chapter 8 is a simple example of data parallel deep network training.

As a historical note, Google's predecessor to TensorFlow, DistBelief, was based on data parallel training on CPU servers. This system was capable of achieving distributed CPU speeds (using 32–128 nodes) that matched or exceeded GPU training speeds. Figure 9-6 illustrates the data parallel training method implemented by DistBelief. However, the success of systems like DistBelief tends to depend on the presence of high throughput network interconnects that can allow for rapid model parameter sharing. Many organizations lack the network infrastructure that enables effective multinode data parallel CPU training. However, as the A3C example demonstrates, it is possible to perform data parallel training on a single node, using different CPU cores. For modern servers, it is also possible to perform data parallel training using multiple GPUs stocked within a single server, as we will show you later.

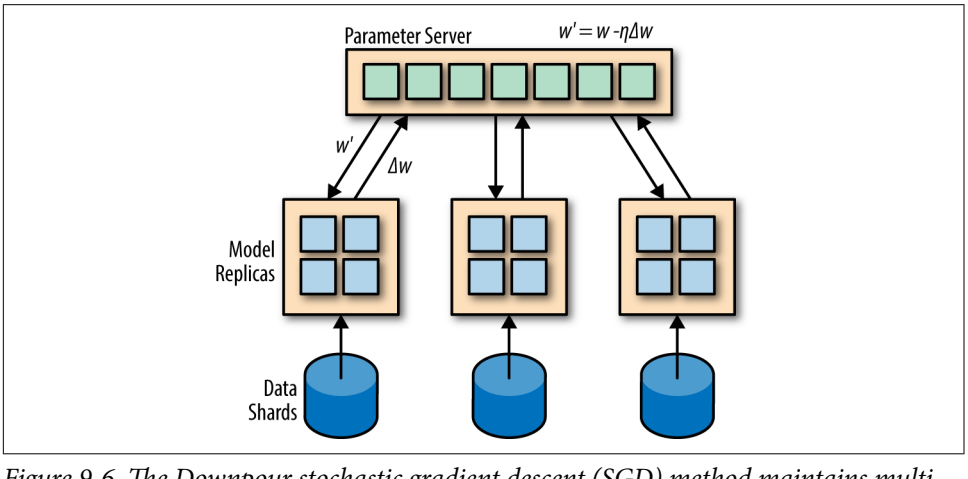

Figure 9-6. The Downpour stochastic gradient descent (SGD) method maintains multiple replicas of the model and trains them on different subsets of a dataset. The learned weights from these shards are periodically synced to global weights stored on a parameter server.

Model Parallelism

The human brain provides the only known example of a generally intelligent piece of hardware, so there have naturally been comparisons drawn between the complexity of deep networks and the complexity of the brain. Simple arguments state the brain has roughly 100 billion neurons; would constructing deep networks with that many "neurons" suffice to achieve general intelligence? Unfortunately, such arguments miss the point that biological neurons are significantly more complex than "mathematical neurons." As a result, simple comparisons yield little value. Nonetheless, building larger deep networks has been a major research focus over the last few years.

The major difficulty with training very large deep networks is that GPUs tend to have limited memory (dozens of gigabytes typically). Even with careful encodings, neural networks with more than a few hundred million parameters are not feasible to train on single GPUs due to memory requirements. Model parallel training algorithms attempt to sidestep this limitation by storing large deep networks on the memories of multiple GPUs. A few teams have successfully implemented these ideas on arrays of GPUs to train deep networks with billions of parameters. Unfortunately, these models have not thus far shown performance improvements justifying the extra difficulty. For now, it seems that the increase in experimental ease from using smaller models outweighs the gains from model parallelism.

Hardware Memory Interconnects

Enabling model parallelism requires having very high bandwidth connections between compute nodes since each gradient update by necessity requires internode communication. Note that while data parallelism requires strong interconnects, sync operations need only be performed sporadically after multiple local gradient updates.

A few groups have used InfiniBand interconnects (InfiniBand is a high-throughput, low-latency networking standard), or Nvidia's proprietary NVLINK interconnects to attempt to build such large models. However, the results from such experiments have been mixed thus far, and the hardware requirements for such systems tend to be expensive.