Parallel Gradient Descent for Multilayer Feedforward Neural Networks

Palash Goyal\textsuperscript{1}  Nitin Kamra\textsuperscript{1}  Sungyong Seo\textsuperscript{1}  Vasileios Zois\textsuperscript{1}

\textsuperscript{1}Department of Computer Science
University of Southern California

May 9, 2016
Outline

1. Introduction
2. Gradient Descent
3. Forward Propagation and Backpropagation
4. Parallel Gradient Descent
5. Experiments
6. Results and analysis
Outline

1. Introduction
2. Gradient Descent
3. Forward Propagation and Backpropagation
4. Parallel Gradient Descent
5. Experiments
6. Results and analysis
Introduction

- How to learn to classify objects from images?
- What algorithms to use?
- How to scale up these algorithms?
Classification

- Dataset \( \mathcal{D} = \{ x^{(i)}, y^{(i)} \}_{i=1:N} \) with \( x^{(i)} \in \mathbb{R}^D \) and labels \( y^{(i)} \in \mathbb{R}^P \)
- Make accurate prediction \( \hat{y} \) on unseen data point \( x \)
- Classifier (parameters \( \theta \)) approximates label as: \( y \approx \hat{y} = F(x; \theta) \)
- Classifier learns parameters (\( \theta \)) from data \( \mathcal{D} \) to minimize a pre-specified loss function
Neuron

\[ a = f(w^T x + b) \]

- \( w \in \mathbb{R}^n = \text{Weight vector} \)
- \( b \in \mathbb{R} = \text{Scalar bias} \)
For each layer,

\[ z_l = (W_l)^T x_l + b_l; \quad a_l = f(z_l) \]

- \( W^l \in \mathbb{R}^{n_{l-1} \times n_l} = \text{Weight vector} \)
- \( b^l \in \mathbb{R}^{n_l} = \text{Scalar bias} \)
1. Introduction

2. Gradient Descent

3. Forward Propagation and Backpropagation

4. Parallel Gradient Descent

5. Experiments

6. Results and analysis
Gradient Descent

Minimize the Mean-Squared Error loss:

\[ \mathcal{L}_{MSE}(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - f(x^{(i)}; \theta))^2 \]

Algorithm: Gradient Descent

1. Initialize all weights \((\theta)\) randomly with small values close to 0.
2. Repeat until convergence {

\[ \theta_k := \theta_k - \alpha \frac{\partial \mathcal{L}_{MSE}}{\partial \theta_k} \quad \forall k \in \{1, 2, \ldots, K\} \]

} 

Minibatch gradient descent considers a subset of examples
Outline

1. Introduction
2. Gradient Descent
3. Forward Propagation and Backpropagation
4. Parallel Gradient Descent
5. Experiments
6. Results and analysis
Algorithm 3 Forward Propagation

**Input:** Example $x$, parameters $[W_{2:L}, b_{2:L}]$

**Output:** $z_l(x), a_l(x) \quad \forall l = 1 : L$

$z_1(x) := x, a_1(x) := x$

for $l = 2 : L$ do

$z_l(x) = (W_l)^T a_{l-1}(x) + b_l$

$a_l(x) = \sigma(z_l)$

end for
Algorithm 4 Backpropagation

**Input:** Example $x$, label $y$, parameters $[W_{2:L}, b_{2:L}]$

**Output:** Derivatives $\{\frac{\partial L_{MSE}}{\partial b_l}\}_{l=2:L}, \{\frac{\partial L_{MSE}}{\partial W_l}\}_{l=2:L}$

Compute $z_l(x), a_l(x) \; \forall l = 1 : L$ with a forward pass

$$\delta_L := \frac{\partial L_{MSE}}{\partial a_L} \circ \sigma'(z_L(x))$$

**for** $l = L : 2$ **do**

$$\frac{\partial L_{MSE}}{\partial b_l} := \delta_l$$

$$\frac{\partial L_{MSE}}{\partial W_l} := a_{l-1} \delta_l^T$$

$$\delta_{l-1} := (W_l \delta_l) \circ \sigma'(z_{l-1}(x))$$

**end for**
Outline

1. Introduction
2. Gradient Descent
3. Forward Propagation and Backpropagation
4. Parallel Gradient Descent
5. Experiments
6. Results and analysis
Parallelizing Gradient Descent

Two ways to parallelize:

- **Parallelize Gradient Descent:**
  Derivative of the loss function has the following form:

  \[
  \frac{\partial L_{MSE}}{\partial \theta_k} = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i; \theta)) \frac{\partial f(x_i; \theta)}{\partial \theta_k}
  \]

  Distribute training examples, compute partial gradients, sum up partial gradients

- **Parallelize Backpropagation:**
  Parallelize matrix vector multiplications in forward propagation and backpropagation algorithms
MNIST dataset

- 28x28 images of handwritten digits
- 50,000 training examples, 10,000 test examples, 10,000 validation examples
- Labels: 0 to 9 (one-hot encoding)
Experiments

**Network structures**

<table>
<thead>
<tr>
<th>Network</th>
<th># Layers (In, Hidden, Out)</th>
<th># Nodes (In, Hidden, Out)</th>
<th># Num Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network1</td>
<td>1,1,1</td>
<td>784,1024,10</td>
<td>800,000</td>
</tr>
<tr>
<td>Network2</td>
<td>1,2,1</td>
<td>784,1024,1024,10</td>
<td>1,860,000</td>
</tr>
</tbody>
</table>

- Serial, Parallelize over examples (Pthreads, CUDA)
- Serial (BLAS), Parallelize matrix computations (BLAS)
- Serial (Keras:Theano), Parallel (Keras:Theano), GPU (Keras:Theano)

Analyze time per epoch, gigaflops for each implementation
Analyze speedup from parallelization over serial counterparts
Outline

1. Introduction
2. Gradient Descent
3. Forward Propagation and Backpropagation
4. Parallel Gradient Descent
5. Experiments
6. Results and analysis
Results - Time per Epoch

Net-2h (Effect of batch size on performance - seconds)

- Theano CUDA
- Theano parallel
- Theano serial
- Parallel (BLAS)
- Serial (BLAS)
- CUDA Optimized
- CUDA
- Parallel
- Serial

Batch sizes: 4096, 2048, 1024, 512, 256, 128
Results - Gigaflops

Net-2h (Effect of batch on performance - GFLOPS)

- Theano CUDA
- Theano parallel
- Theano serial
- Parallel (BLAS)
- Serial (BLAS)
- CUDA Optimized
- CUDA
- Parallel
- Serial

Legend:
- 4096
- 2048
- 1024
- 512
- 256
- 128
Results - Speedup

Net-2h (Speedup over the respective serial implementation)
Analysis

- **Our implementation**
  - Parallel computing average speedup $\approx 10$
  - Training time decreases as minibatch size decreases

- **BLAS**
  - Parallelizing each matrix vector product gives even faster results
  - Speedup independent of batch size, but less than our implementation

- **CUDA**
  - Our CUDA implementation gives about $\approx 20\times$ speedup
  - If number of neurons per layer are not perfect multiple of 32 then some threads do not participate in computation

- **Theano**
  - Apparently combines both types of parallelization
  - Theano CUDA scales very fast with batch size
Combine the two parallelization techniques: Split training examples amongst threads, further hierarchically parallelize matrix computations for each individual example.
Thank you

Questions?