Ray, Distributed Computing, and Machine Learning

Robert Nishihara

11/15/2008
The Machine Learning Ecosystem
The Machine Learning Ecosystem

Training

Machine Learning Ecosystem
The Machine Learning Ecosystem

Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class
The Machine Learning Ecosystem

Multiclass (more than 2) Classification

- One-vs-rest: train separate binary classifiers for each class

Training

Diagnosing Fit

The Residuals

Residuals
The Machine Learning Ecosystem

Multiclass (more than 2) Classification

- One-vs-rest train separate binary classifiers for each class

Stochastic Gradient Descent

- For many learning problems the gradient is a sum:
  \[ \nabla_{\theta} L(\theta) = \frac{1}{n} \sum_{i=1}^{n} (\sigma(\phi(x_i)^T \theta) - y_i) \phi(x_i) \]

- For large n this can be costly

- What if we approximated the gradient by looking at a few random points:
  \[ \nabla_{\theta} L(\theta) \approx \frac{1}{|B|} \sum_{i \in B} (\sigma(\phi(x_i)^T \theta) - y_i) \phi(x_i) \]
The Machine Learning Ecosystem

Training

RL

Machine Learning Ecosystem
The Machine Learning Ecosystem

Training

RL

Machine Learning Ecosystem
The Machine Learning Ecosystem

Machine Learning Ecosystem

RL
The Machine Learning Ecosystem
The Machine Learning Ecosystem
The Machine Learning Ecosystem

- Training
- Model Serving
- RL
The Machine Learning Ecosystem

- Training
- Model Serving
- RL
- Hyperparameter Search

Machine Learning Ecosystem
The Machine Learning Ecosystem

The Bias-Variance Tradeoff

Estimated Model Variance

Bias

Hyperparameter Search
The Machine Learning Ecosystem
The Map Reduce Abstraction (Simpler)

**Map**
- Record
- Map
- Key
- Value
- Key
- Value

Example: Word-Count

**Map**(book):
for (word in book):
    emit (word, 1)

**Reduce**(word, counts) {
    sum = 0
    for count in counts:
        sum += count
    emit (word, SUM(counts))
}

Data Processing

Hyperparameter Search

Machine Learning Ecosystem
The Machine Learning Ecosystem

Training

Model

Serving

Hyperparameter Search

Data Processing

The Map Reduce Abstraction (Simpler)

Example: Word-Count

Reduce(word, counts) {

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```python
lines = spark.textFile("hdfs://file.txt")
errors = lines.filter(lambda s: s.startswith("ERROR")).
messages = errors.map(lambda s: s.split("\t") [2]).count()
messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```
The Machine Learning Ecosystem

Training

Model Serving

Streaming

RL

Data Processing

Hyperparameter Search

Machine Learning Ecosystem
The Machine Learning Ecosystem

Distributed System
- Training
Distributed System
- Model Serving
Distributed System
- Streaming
Distributed System
- RL
Distributed System
- Data Processing
Distributed System
- Hyperparameter Search
The Machine Learning Ecosystem

- **Training**: Horovod, Distributed TF, Parameter Server
- **Model Serving**: Clipper, TensorFlow Serving
- **Streaming**: Flink, many others
- **RL**: Baselines, RLLab, ELF, Coach, TensorFlow, ChainerRL
- **Data Processing**: MapReduce, Hadoop, Spark
- **Hyperparameter Search**: Vizier, many internal systems at companies
The Machine Learning Ecosystem

Distributed System

- Training: Horovod, Distributed TF, Parameter Server
- Model Serving: Clipper, TensorFlow Serving
- Streaming: Flink, many others
- RL: Baselines, RLLab, ELF, Coach, TensorForce, ChainerRL
- Data Processing: MapReduce, Hadoop, Spark
- Hyperparameter Search: Vizier, many internal systems at companies

Why distributed systems?
The Machine Learning Ecosystem

Why distributed systems?
- More work (computation) than one machine can do in a reasonable amount of time.
- More data than can fit in one machine.
# The Machine Learning Ecosystem

## Aspects of a distributed system

<table>
<thead>
<tr>
<th>Distributed System</th>
<th>Example Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Horovod, Distributed TF, Parameter Server</td>
</tr>
<tr>
<td>Model Serving</td>
<td>Clipper, TensorFlow Serving</td>
</tr>
<tr>
<td>Streaming</td>
<td>Flink, many others</td>
</tr>
<tr>
<td>RL</td>
<td>Baselines, RLLab, ELF, Coach, TensorFlow, ChainerRL</td>
</tr>
<tr>
<td>Data Processing</td>
<td>MapReduce, Hadoop, Spark</td>
</tr>
<tr>
<td>Hyperparameter Search</td>
<td>Vizier, many internal systems at companies</td>
</tr>
</tbody>
</table>

## Why distributed systems?

- More **work** (computation) than one machine can do in a reasonable amount of time.
- More **data** than can fit in one machine.
# The Machine Learning Ecosystem

## Distributed System

<table>
<thead>
<tr>
<th>Training</th>
<th>Horovod, Distributed TF, Parameter Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Serving</td>
<td>Clipper, TensorFlow Serving</td>
</tr>
<tr>
<td>Streaming</td>
<td>Flink, many others</td>
</tr>
<tr>
<td>RL</td>
<td>Baselines, RLlab, ELF, Coach, TensorForce, ChainerRL</td>
</tr>
<tr>
<td>Data Processing</td>
<td>MapReduce, Hadoop, Spark</td>
</tr>
<tr>
<td>Hyperparameter Search</td>
<td>Vizier, many internal systems at companies</td>
</tr>
</tbody>
</table>

**Aspects of a distributed system**
- Units of work “tasks” executed in parallel
- Scheduling (which tasks run on which machines and when)
- Data transfer
- Failure handling
- Resource management (CPUs, GPUs, memory)

**Why distributed systems?**
- More **work** (computation) than one machine can do in a reasonable amount of time.
- More **data** than can fit in one machine.
What is Ray?

Aspects of a distributed system
- Units of work “tasks” executed in parallel
- Scheduling (which tasks run on which machines and when)
- Data transfer
- Failure handling
- Resource management (CPUs, GPUs, memory)

Why distributed systems?
- More work (computation) than one machine can do in a reasonable amount of time.
- More data than can fit in one machine.
What is Ray?

Aspects of a distributed system
- Units of work “tasks” executed in parallel
- Scheduling (which tasks run on which machines and when)
- Data transfer
- Failure handling
- Resource management (CPUs, GPUs, memory)

Why distributed systems?
- More work (computation) than one machine can do in a reasonable amount of time.
- More data than can fit in one machine.
<table>
<thead>
<tr>
<th>Distributed System</th>
<th>Training</th>
<th>Horovod, Distributed TF, Parameter Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed System</td>
<td>Model Serving</td>
<td>Clipper, TensorFlow Serving</td>
</tr>
<tr>
<td>Distributed System</td>
<td>Streaming</td>
<td>Flink, many others</td>
</tr>
<tr>
<td>Distributed System</td>
<td>RL</td>
<td>Baselines, RLLab, ELF, Coach, TensorFlow, ChainerRL</td>
</tr>
<tr>
<td>Distributed System</td>
<td>Data Processing</td>
<td>MapReduce, Hadoop, Spark</td>
</tr>
<tr>
<td>Distributed System</td>
<td>Hyperparameter Search</td>
<td>Vizier, many internal systems at companies</td>
</tr>
</tbody>
</table>
Distributed System (Ray)

Libraries

- Training
- Model Serving
- Streaming
- RL
- Data Processing
- Hyperparameter Search

Horovod, Distributed TF, Parameter Server
Clipper, TensorFlow Serving
Flink, many others
Baselines, RLlab, ELF, Coach, TensorForce, ChainerRL
MapReduce, Hadoop, Spark
Vizier, many internal systems at companies
This requires a very **general** underlying distributed system.
This requires a very **general** underlying distributed system.

*Generality comes from tasks (functions) and actors (classes).*
def zeros(shape):
    return np.zeros(shape)

def dot(a, b):
    return np.dot(a, b)
Ray API

Tasks

```python
@ray.remote
def zeros(shape):
    return np.zeros(shape)

@ray.remote
def dot(a, b):
    return np.dot(a, b)
```
Ray API

**Tasks**

```python
@ray.remote
def zeros(shape):
    return np.zeros(shape)
```

```python
@ray.remote
def dot(a, b):
    return np.dot(a, b)
```

```python
id1 = zeros.remote([5, 5])
```
Ray API

@ray.remote
def zeros(shape):
    return np.zeros(shape)

@ray.remote
def dot(a, b):
    return np.dot(a, b)

id1 = zeros.remote([5, 5])
id2 = zeros.remote([5, 5])
Ray API

Tasks

```python
@ray.remote
def zeros(shape):
    return np.zeros(shape)

@ray.remote
def dot(a, b):
    return np.dot(a, b)

id1 = zeros.remote([5, 5])
id2 = zeros.remote([5, 5])
id3 = dot.remote(id1, id2)
```
Ray API

Tasks

```python
@ray.remote
def zeros(shape):
    return np.zeros(shape)

@ray.remote
def dot(a, b):
    return np.dot(a, b)

id1 = zeros.remote([5, 5])
id2 = zeros.remote([5, 5])
id3 = dot.remote(id1, id2)
ray.get(id3)
```
Ray API

Tasks

```python
@ray.remote
def zeros(shape):
    return np.zeros(shape)

@ray.remote
def dot(a, b):
    return np.dot(a, b)

id1 = zeros.remote([5, 5])
id2 = zeros.remote([5, 5])
id3 = dot.remote(id1, id2)
ray.get(id3)
```

Show in Jupyter notebook!
Ray API

Actors

class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value
Ray API

**Actors**

```python
@ray.remote(num_gpus=1)
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
    return self.value
```
Ray API

```
@ray.remote(num_gpus=1)
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
    return self.value

c = Counter.remote()
```
```python
@ray.remote(num_gpus=1)
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter.remote()
id4 = c.inc.remote()
```

**Actors**

```python
Counter
\downarrow
\text{id4}
```
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter.remote()
id4 = c.inc.remote()
id5 = c.inc.remote()
@ray.remote(num_gpus=1)
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter.remote()
id4 = c.inc.remote()
id5 = c.inc.remote()
ray.get([id4, id5])
How does this work under the hood?
How does this work under the hood?

Tasks

```python
@ray.remote
def zeros(shape):
    return np.zeros(shape)

@ray.remote
def dot(a, b):
    return np.dot(a, b)
```

```python
id1 = zeros.remote([5, 5])
id2 = zeros.remote([5, 5])
id3 = dot.remote(id1, id2)
ray.get(id3)
```
How does this work under the hood?
The Ray Architecture
The Ray Architecture
The Ray Architecture
The Ray Architecture

Worker
Object Store
Scheduler
Global Control Store

zeros
dot

id1
id2
id3

zeros
dot
The Ray Architecture
The Ray Architecture
The Ray Architecture

- Worker
- Worker
- Object Store
- Global Control Store
- Scheduler
- Worker
- Worker
- Object Store
- Global Control Store
- Scheduler
- Worker
- Worker
- Object Store
- Global Control Store
- Scheduler

- id1
- id2
- id3
- zeros
- zeros
- dot
- obj1
- obj2
- obj3
The Ray Architecture
The Ray Architecture
The Ray Architecture
The Ray Architecture

Worker  
Object Store  
obj1  obj3  obj2  
Scheduler  

Worker  
Object Store  
obj2  
Scheduler  

Worker  
Object Store  

Worker  

Global Control Store  

zeros  
id1  

zeros  
id2  

dot  
id3  

The Ray Architecture
Parallelism Example: Primality Testing
Parallelism Example: Primality Testing

Write a function to test if $p$ is a prime number (no divisors other than 1 and $p$).
Parallelism Example: Primality Testing

Write a function to test if \( p \) is a prime number (no divisors other than 1 and \( p \)).

```python
def is_prime(p):
    for d in range(2, p):
        if p % d == 0:
            return False
    return True
```
Parallelism Example: Primality Testing

Write a function to test if $p$ is a prime number (no divisors other than 1 and $p$).

```python
def is_prime(p):
    for d in range(2, p):
        if p % d == 0:
            return False
    return True
```

Can this be parallelized?
Parallelism Example: Primality Testing

Write a function to test if \( p \) is a prime number (no divisors other than 1 and \( p \)).

```python
def is_prime(p):
    for d in range(2, p):
        if p % d == 0:
            return False
    return True
```

Can this be parallelized?
Yes! “For loops” are great candidates for parallelization.
Parallelism Example: Primality Testing

Write a function to test if \( p \) is a prime number (no divisors other than 1 and \( p \)).

```python
def is_prime(p):
    for d in range(2, p):
        if p % d == 0:
            return False
    return True
```

Check batches of divisors in parallel
Parallelism Example: Primality Testing

Write a function to test if \( p \) is a prime number (no divisors other than 1 and \( p \)).

\[
\begin{align*}
\text{def } & \text{is\_prime}(p): \\
& \text{for } d \text{ in range}(2, p): \\
& \quad \text{if } p \% d == 0:\ \\
& \quad \quad \text{return False} \\
& \text{return True}
\end{align*}
\]

Check batches of divisors in parallel:

1, 2, 3, ..., \( k \)      \( k+1, \ldots, 2k \)      ...      \( p-k+1, \ldots, p \)
Parallelism Example: Tree Reduction
Parallelism Example: Tree Reduction

Relevant for upcoming homework assignment!
Parallelism Example: Tree Reduction

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

```
@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
```

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

Add a bunch of values (2 at a time).

```python
@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
```
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

```
@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
```

Add a bunch of values (2 at a time).
Add a bunch of values (2 at a time).
Add a bunch of values (2 at a time).

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Parallelism Example: Tree Reduction
Parallelism Example: Tree Reduction

Add a bunch of values (2 at a time).

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
Parallelism Example: Tree Reduction

```
@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
```

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

Add a bunch of values (2 at a time).

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
Parallelism Example: Tree Reduction

Add a bunch of values (2 at a time).

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
Parallelism Example: Tree Reduction

```
@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
```

Add a bunch of values (2 at a time).
Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

Add a bunch of values (2 at a time).

```python
def add(x, y):
    time.sleep(0.5)
    return x + y
```

@ray.remote
```python
def add(x, y):
    time.sleep(0.5)
    return x + y
```
Parallelism Example: Tree Reduction

@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

```
@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
```

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

What does the difference look like in code?

```python
@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
```

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

What does the difference look like in code?

```python
@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y

vals = [1, 2, 3, 4, 5, 6, 7, 8]
while len(vals) > 1:
    new_val = add.remote(vals[0], vals[1])
    vals = [new_val] + vals[2:]
```

Add a bunch of values (2 at a time).
Parallelism Example: Tree Reduction

What does the difference look like in code?

```python
@ray.remote
def add(x, y):
    time.sleep(0.5)
    return x + y
vals = [1, 2, 3, 4, 5, 6, 7, 8]
while len(vals) > 1:
    new_val = add.remote(vals[0], vals[1])
    vals = [new_val] + vals[2:]
```
Parallelism Example: Parameter Server
Parallelism Example: Parameter Server

Diagnosing Fit → The Residuals
Parallelism Example: Parameter Server

Diagnosing Fit → The ReLU Model

Stochastic Gradient Descent

- For many learning problems the gradient is a sum:

\[ \nabla_{\theta} L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left( \sigma \left( \phi(x_i)^T \theta \right) - y_i \right) \phi(x_i) \]

- For large \( n \) this can be costly

- What if we approximated the gradient by looking at a few random points:

\[ \nabla_{\theta} L(\theta) \approx \frac{1}{|B|} \sum_{i \in B} \left( \sigma \left( \phi(x_i)^T \theta \right) - y_i \right) \phi(x_i) \]
Parallelism Example: Parameter Server

Diagnosing Fit → The Red Line

Stochastic Gradient Descent

- For many learning problems the gradient is a sum:

\[
\nabla_{\theta} L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left( \sigma \left( (\phi(x_i)^T \theta) - y_i \right) \phi(x_i) \right)
\]

- For large \( n \) this can be costly.

- What if we approximated the gradient by looking at a few random points:

\[
\nabla_{\theta} L(\theta) \approx \frac{1}{|B|} \sum_{i \in B} \left( \sigma \left( (\phi(x_i)^T \theta) - y_i \right) \phi(x_i) \right)
\]
Parallelism Example: Parameter Server
Parallelism Example: Parameter Server
Parallelism Example: Parameter Server
Parallelism Example: Parameter Server

Parameter Server Actor

Worker

Worker

Worker

Worker

parameters

gradients
The ML ecosystem consists of many **special-purpose systems**

- We are moving toward a single **general-purpose system**
- “Distributed systems” are becoming “libraries” and “applications”
- Examples like MapReduce, tree reduce, parameter server
- Interesting applications will combine all of these elements

Ray is open source at [https://github.com/ray-project/ray](https://github.com/ray-project/ray)

- pip install ray (on Linux/Mac)
- Being developed here at Berkeley!

Will be used in the next homework assignment