Introducing Sparse Tensor in Apache MXNet

Haibin Lin, @eric-haibin-lin
Motivation - Language Model

Diagram:
- $X_t$ connected to RNN
- RNN connected to Fully Connected
- Fully Connected connected to Softmax
- Softmax connected to $Y_t$
Gradients for Embeddings are Sparse

“The fox jumps over the lazy dog”
Gradients for Embeddings are Sparse

<table>
<thead>
<tr>
<th>King</th>
<th>The</th>
<th>Weather</th>
<th>Lazy</th>
<th>Simple</th>
<th>Quick</th>
<th>Fox</th>
<th>Brown</th>
<th>Naive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

“The fox jumps over the lazy dog”
Gradients for Embeddings are Sparse

“The fox jumps over the lazy dog”

King
The
Weather
Lazy
Simple
Quick
Fox
Brown
Naive
……...
……...
Dog
Over
Young
Jump

indices
data
MXNet Sparse Tensor Types

- Sparse Gradient (e.g. word embeddings in NLP)
- Sparse Data (e.g. recommender systems, social networks, computational ads)

Row Sparse format

```
mx.nd.sparse.RowSparseArray
```

Row ids               data
MXNet Sparse Tensor Types

- Sparse Gradient (e.g. word embeddings in NLP)
- Sparse Data (e.g. recommender systems, social networks, computational ads)

Compressed Sparse Row (CSR) format

mx.nd.sparse.CSRNDArray
MXNet Sparse Feature

- Sparse Data Formats
- Sparse Operators
- Multi-GPU & Multi-Machine Communication
Loading Sparse Data

```python
>>> import mxnet as mx
>>> import scipy.sparse as spsp
>>> sp_matrix = spsp.rand(2, 3, density=0.3, dtype='float32')
>>> print(sp_matrix.toarray())
[[ 0.          0.                0.89449775 ]
[  0.          0.57767099  0.               ]]```
Loading Sparse Data

```python
>>> import mxnet as mx
>>> import scipy.sparse as spsp
>>> sp_matrix = spsp.rand(2, 3, density=0.3, dtype='float32')
>>> print(sp_matrix.toarray())
[[ 0.          0.                0.89449775 ]
 [  0.          0.57767099  0.               ]]

>>> x = mx.ndarray.sparse.csr_matrix(sp_matrix, ctx=mx.cpu())
>>> x
<CSRNDArray 2x3 @cpu(0)>
```
MXNet Sparse Feature

- Sparse Data Formats
- **Sparse Operators**
- Multi-GPU & Multi-Machine Communication
Sparse Matrix-Dense Vector Multiplication

```python
>>> w = mx.nd.array.random.uniform(shape=(3, 1))
>>> w
[[ 0.54881352]
 [ 0.59284461]
 [ 0.85794562]]
<NDArray 3x1 @cpu(0)>

>>> y = mx.nd.array.sparse.dot(x, w)
>>> y
[[ 0.75519383]
 [ 0.34246913]
 [ 0.34246913]]
<NDArray 2x1 @cpu(0)>
```
Sparse Matrix-Dense Vector Multiplication

- KDD Cup 2010 dataset on r4.8xlarge (CPU)

<table>
<thead>
<tr>
<th></th>
<th>Dense Dot</th>
<th>Sparse Dot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>121.38 ms</td>
<td>0.21 ms</td>
</tr>
<tr>
<td>Memory</td>
<td>~ 5 GB</td>
<td>~ 100 MB</td>
</tr>
</tbody>
</table>
MXNet Sparse Feature

- Sparse Data Formats
- Sparse Operators
- Multi-GPU & Multi-Machine Communication
Multi-GPU & Multi-Machine Communication

- Broadcasting the full model consumes lots of bandwidth
Multi-GPU & Multi-Machine Communication

- Broadcasting the partial model required for computation
Scalability Benchmarks

LSTM 2048-512 on Google Billion Words dataset, p2.4xlarge
Getting Started

Examples

Tutorials

Feature Request
https://github.com/apache/incubator-mxnet/issues