TABLA: A Unified Template-based Framework for Accelerating Statistical Machine Learning

Divya Mahajan
Jongse Park
Emmanuel Amaro
Hardik Sharma
Amir Yazdanbakhsh
Joon Kyung Kim
Hadi Esmaeilzadeh

Alternative Computing Technologies (ACT) Lab
Georgia Institute of Technology
Data Explosion

- 98,000 Tweets
- 695,000 Updates
- 700,000 Searches
- 11,000,000 Messages
- 168,000,000 Emails

© 2016 D Mahajan ALL RIGHTS RESERVED
Machines learn to extract insights from data

Learning Phase

Training Data → Machine Learning Algorithm

- linear separation
- non-linear separation

Trained Model

Inference Phase

Unseen Data → Predictions

© 2016 D Mahajan ALL RIGHTS RESERVED
Machines learn to extract insights from data

Learning Phase

Training Data

Machine Learning Algorithm

linear separation non-linear separation

Trained Model

Inference Phase

Unseen Data

Predictions
Growing gap between data and compute

Programmability versus efficiency

- CPUs
- GPUs
- FPGAs
- ASICs
An algorithmic approach towards acceleration

- Expertise in Hardware design
- Performance Gap
- Energy Constraints
- Long Design Cycles

- Target a range of algorithms
- Automates Hardware Generation
- Work Top-Down
- Template-based

TABLA

© 2016 D Mahajan ALL RIGHTS RESERVED
Understanding machine learning

Training Data

<table>
<thead>
<tr>
<th>Input (X)</th>
<th>Output (Y*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25,1,76,0</td>
<td>1</td>
</tr>
<tr>
<td>6,43,9,93</td>
<td>6</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>23,56,2,0</td>
<td>12</td>
</tr>
<tr>
<td>12,0,9,0</td>
<td>0</td>
</tr>
</tbody>
</table>

Output (Y*)

Predicted Output (Y)

Intermediate Model

\[ \text{Loss}(w_i) = \sum_{i} ||Y - Y^*|| \]
Algorithmic commonalities

Learning is solving an iterative optimization problem!

Training Data

<table>
<thead>
<tr>
<th>Input (X)</th>
<th>Output (Y*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25,1,76,0</td>
<td>1</td>
</tr>
<tr>
<td>6,43,9,93</td>
<td>6</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>23,56,2,0</td>
<td>12</td>
</tr>
<tr>
<td>12,0,9,0</td>
<td>0</td>
</tr>
</tbody>
</table>

Output (Y*)

Trained Model

Intermediate Model

Predicted Output (Y)

Loss ($w_i$)

$$\text{find}(w_i) \in \{ \text{Loss}(w_i) = \sum_i ||Y - Y^*|| \} \text{ is minimized}$$
Stochastic gradient descent solver

Loss function \( (w_i) = f \left( w_1^{(t)}, w_2^{(t)}, \ldots, w_m^{(t)} \right) \)

\[ w_i^{(t+1)} = w_i^t - u \times \frac{\partial f \left( w_1^{(t)}, w_2^{(t)}, \ldots, w_m^{(t)} \right)}{\partial w_i^{(t)}} \]
Abstraction between hardware and software

Machine Learning Algorithms

- Support Vector Machines
  - Gradient Loss Function
- Regression Analysis
  - Gradient Loss Function
- Back Propagation
  - Gradient Loss Function
- Collaborative Filtering
  - Gradient Loss Function
- Kalman Filters
  - Gradient Loss Function

Stochastic Gradient Descent Solver
(Abstraction between Hardware and Software)

Xeon Phi, GPU, FPGA, CGRA, ASIC
Cross-stack template-based approach

- Programming Language
- Compilation
- Template Architectures
- Accelerator design
Programming interface

Enables programmer to specify the gradient of loss function
Programming interface

Data Declarations

Mathematical Operations
Example: Logistic Regression

\[
\text{Loss Function Gradient}(W^T) = \left\{ \left( \text{sigmoid} \left( \sum_i W^T X_i \right) - Y \right) \cdot X_i \right\} + \lambda \cdot W^T
\]

\[
m = 53
\]
\[
\lambda = 0.1
\]

\[
\text{model\_input} \quad X[m];
\]
\[
\text{model\_output} \quad Y';
\]
\[
\text{model} \quad W[m];
\]
\[
\text{gradient} \quad G[m];
\]
\[
\text{iterator} \quad i[0:m - 1];
\]
\[
S = \text{sum}[i](X[i] \ast W[i]);
\]
\[
Y = \text{sigmoid}(S);
\]
\[
E = Y - Y';
\]
\[
G[i] = X[i] \ast E;
\]
\[
V[i] = \lambda \ast W[i];
\]
\[
G[i] = G[i] + V[i];
\]
Example: Logistic Regression

Loss Function Gradient($W^T$) = \[ \left\{ \left( \text{sigmoid} \left( \sum_i W^T X_i \right) - Y \right) \cdot X_i \right\} + \lambda \cdot W^T \]

m = 53
lambda = 0.1

```plaintext
model_input   X[m];
model_output  Y';
model         W[m];
gradiant      G[m];

iterator      i[0:m - 1];

S = sum [i](X[i] * W[i]);

Y = sigmoid(S);
E = Y - Y';
G[i] = X[i] * E;
V[i] = lambda * W[i];
G[i] = G[i] + V[i];
```
Example: Logistic Regression

\[
\text{Loss Function Gradient}(W^T) = \left\{ \left( \text{sigmoid} \left( \sum_i W^T X_i \right) \right) - Y \right\} \cdot X_i + \lambda \cdot W^T
\]

m = 53
lambda = 0.1

\[
\text{model\_input} \quad X[m];
\text{model\_output} \quad Y';
\text{model} \quad W[m];
\text{gradient} \quad G[m];
\]

iterator i[0:m - 1];

S = sum [i](X[i] * W[i]);

Y = sigmoid(S);
E = Y - Y';
G[i] = X[i] * E;
V[i] = lambda * W[i];
G[i] = G[i] + V[i];
Example: Logistic Regression

\[
\text{Loss Function Gradient}(W^T) = \left\{ \left( \text{sigmoid}\left( \sum_i W^T X_i \right) - Y \right) \cdot X_i \right\} + \lambda \cdot W^T
\]

\begin{align*}
m &= 53 \\
\lambda &= 0.1
\end{align*}

\begin{verbatim}
model_input   X[m];
model_output  Y';
model         W[m];
gradient      G[m];

iterator     i[0:m - 1];
S = sum [i](X[i] * W[i]);
Y = sigmoid(S);
E = Y - Y';
G[i] = X[i] * E;
V[i] = \lambda \cdot W[i];
G[i] = G[i] + V[i];
\end{verbatim}
Example: Logistic Regression

\[
\text{Loss Function Gradient}(W^T) = \left\{ \left( \left( \text{sigmoid} \left( \sum_i W^T X_i \right) \right) - Y \right) \cdot X_i \right\} + \lambda \cdot W^T
\]

\[
\begin{align*}
m &= 53 \\
\lambda &= 0.1 \\
\text{model input} & \quad X[m] \\
\text{model output} & \quad Y' \\
\text{model} & \quad W[m] \\
\text{gradient} & \quad G[m] \\
\text{iterator} & \quad i[0:m - 1] \\
S &= \text{sum} [i](X[i] \ast W[i]) \\
Y &= \text{sigmoid}(S) \\
E &= Y - Y' \\
G[i] &= X[i] \ast E \\
V[i] &= \lambda \ast W[i] \\
G[i] &= G[i] + V[i]
\end{align*}
\]
Example: Logistic Regression

\[
\text{Loss Function Gradient}(W^T) = \left\{ \left( \left( \text{sigmoid} \left( \sum_i W^T X_i \right) \right) - Y \right) \cdot X_i \right\} + \lambda \cdot W^T
\]

\begin{align*}
m &= 53 \\
\lambda &= 0.1 \\
\text{model_input} &\quad X[m] \\
\text{model_output} &\quad Y' \\
\text{model} &\quad W[m] \\
\text{gradient} &\quad G[m] \\
\text{iterator} &\quad i[0:m - 1] \\
S &= \text{sum} \ [i](X[i] \ast W[i]) \\
Y &= \text{sigmoid}(S) \\
E &= Y - Y' \\
G[i] &= X[i] \ast E \\
V[i] &= \lambda \ast W[i] \\
G[i] &= G[i] + V[i]
\end{align*}
1. Appending the stochastic gradient descent solver
2. Creating the dataflow graph of the entire algorithm
3. Create a schedule for this dataflow graph
1. Appending Stochastic Gradient Descent

```python
m = 53
lambda = 0.1
u = 0.1

model_input  X[m];
model_output Y';
model        W[m];
gradient     G[m];

iterator i[0:m - 1];

S = sum [i](X[i] * W[i]);

Y = sigmoid(S);
E = Y - Y';
G[i] = X[i] * E;
V[i] = lambda * W[i];
G[i] = G[i] + V[i];

G[i] = u * G[i];
W[i] = W[i] - G[i];
```
2. Dataflow graph generation

- Multiply
- Sum
- Norm
- Sigmoid
2. Dataflow graph generation

```plaintext
m = 53
lambda = 0.1
u = 0.1

model_input X[m];
model_output Y';
model W[m];
gradient G[m];

iterator i[0:m - 1];

S = \text{sum}[i](X[i] \times W[i]);

Y = sigmoid(S);
E = Y - Y';
G[i] = X[i] \times E;
V[i] = lambda \times W[i];
G[i] = G[i] + V[i];

G[i] = u \times G[i];
W[i] = W[i] - G[i];
```

© 2016 D Mahajan  ALL RIGHTS RESERVED
2. Dataflow graph generation

```plaintext
m = 53
lambda = 0.1
u = 0.1

model_input X[m];
model_output Y;
model W[m];
gradien t G[m];

iterator i[0:m - 1];

S = sum [i](X[i] * W[i]);

Y = sigmoid(S);
E = Y - Y';
G[i] = X[i] * E;
V[i] = lambda * W[i];
G[i] = G[i] + V[i];

G[i] = u * G[i];
W[i] = W[i] - G[i];
```

2. Dataflow graph generation

```plaintext
m = 53
lambda = 0.1
u = 0.1

model_input  X[m];
model_output Y';
model        W[m];
gradien t    G[m];

iterator i[0:m - 1];

S = sum [i](X[i] * W[i]);

Y = sigmoid(S);
E = Y - Y';
G[i] = X[i] * E;
V[i] = lambda * W[i];
G[i] = G[i] + V[i];

G[i] = u * G[i];
W[i] = W[i] - G[i];
```

© 2016 D Mahajan ALL RIGHTS RESERVED
2. Dataflow graph generation

\[
\begin{align*}
\text{m} &= 53 \\
\text{lambda} &= 0.1 \\
\text{u} &= 0.1 \\
\text{model_input} &\quad \text{X[m];} \\
\text{model_output} &\quad \text{Y';} \\
\text{model} &\quad \text{W[m];} \\
\text{gradient} &\quad \text{G[m];} \\
\text{iterator} &\quad \text{i[0:m - 1];} \\
\text{S} &= \text{sum} \left[ i \right] \left( \text{X}[i] \times \text{W}[i] \right); \\
\text{Y} &= \text{sigmoid}(S); \\
\text{E} &= \text{Y} - \text{Y'}; \\
\text{G}[i] &= \text{X}[i] \times \text{E}; \\
\text{V}[i] &= \text{lambda} \times \text{W}[i]; \\
\text{G}[i] &= \text{G}[i] + \text{V}[i]; \\
\text{G}[i] &= \text{u} \times \text{G}[i]; \\
\text{W}[i] &= \text{W}[i] - \text{G}[i];
\end{align*}
\]
2. Dataflow graph generation

\[ m = 53 \]
\[ \text{lambda} = 0.1 \]
\[ u = 0.1 \]

```plaintext
model_input X[m];
model_output Y';
model W[m];
gradients G[m];

iterator i[0:m - 1];

S = sum[i](X[i] * W[i]);

Y = sigmoid(S);
E = Y - Y';
G[i] = X[i] * E;
V[i] = lambda * W[i];
G[i] = G[i] + V[i];

G[i] = u * G[i];
W[i] = W[i] - G[i];
```

Programmer's Code

Diagram of dataflow graph generation
2. Dataflow graph generation

\[
\begin{align*}
    m &= 53 \\
    \lambda &= 0.1 \\
    u &= 0.1 \\
    \text{model input} & \quad X[m]; \\
    \text{model output} & \quad Y'; \\
    \text{model} & \quad W[m]; \\
    \text{gradient} & \quad G[m]; \\
    \text{iterator} & \quad i[0:m-1]; \\
    S &= \text{sum}[i](X[i] \ast W[i]); \\
    Y &= \text{sigmoid}(S); \\
    E &= Y - Y'; \\
    G[i] &= X[i] \ast E; \\
    V[i] &= \lambda \ast W[i]; \\
    G[i] &= G[i] + V[i]; \\
    G[i] &= u \ast G[i]; \\
    W[i] &= W[i] - G[i]; \\
\end{align*}
\]

Diagram of the dataflow graph generation process.
2. Dataflow graph generation

m = 53
lambda = 0.1
u = 0.1

model_input X[m];
model_output Y';
model W[m];
gradien t G[m];

iterator i[0:m - 1];

S = sum [i](X[i] * W[i]);

Y = sigmoid(S);
E = Y - Y';
G[i] = X[i] * E;
V[i] = lambda * W[i];
G[i] = G[i] + V[i];

G[i] = u * G[i];
W[i] = W[i] - G[i];
3. Operation scheduling

Programmer's Code

Stochastic Gradient Descent Solver

© 2016 D Mahajan ALL RIGHTS RESERVED
Design Builder

Together with the model compiler generates the final accelerator design.
Hierarchical template design of accelerator
Template design of Processing Unit
Template design of Processing Engine
## Learning tasks and their topologies

<table>
<thead>
<tr>
<th>Task</th>
<th>Model</th>
<th>Topology</th>
<th># Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Regression Model</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>Logistic Regression</td>
<td>M1: 54</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M2: 200</td>
<td></td>
</tr>
<tr>
<td><strong>Classification</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support Vector Machines</td>
<td>SVM</td>
<td>M1: 54</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M2: 200</td>
<td></td>
</tr>
<tr>
<td><strong>Collaborative Filtering</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recommender Systems</td>
<td>Reco</td>
<td>M1: 1700 x 1000</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M2: 6000 x 4000</td>
<td></td>
</tr>
<tr>
<td><strong>Multilayer Perceptron</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Back Propagation</td>
<td>BackProp</td>
<td>M1: 10 -&gt; 9 -&gt; 1</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M2: 256 -&gt; 128 -&gt; 256</td>
<td></td>
</tr>
<tr>
<td><strong>Regression Analysis</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear Regression</td>
<td>LinearR</td>
<td>M1: 55</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M2: 784</td>
<td></td>
</tr>
</tbody>
</table>
## Evaluation platforms

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Platform</th>
<th>TDP</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA</td>
<td>Xilinx Zynq 7000 ZC702</td>
<td>2W</td>
<td>$129</td>
</tr>
<tr>
<td>CPU</td>
<td>ARM Cortex 15</td>
<td>5W</td>
<td>$191</td>
</tr>
<tr>
<td></td>
<td>Intel Xeon E3-1276 V3</td>
<td>84W</td>
<td>$339</td>
</tr>
<tr>
<td>GPU</td>
<td>Tegra K1 GPU</td>
<td>10W</td>
<td>$191</td>
</tr>
<tr>
<td></td>
<td>GeForce GTX 650 Ti</td>
<td>110W</td>
<td>$150</td>
</tr>
<tr>
<td></td>
<td>Tesla K40</td>
<td>210W</td>
<td>$5499</td>
</tr>
</tbody>
</table>

© 2016 D Mahajan ALL RIGHTS RESERVED
Speedup in comparison to CPUs

- **TABLA** generated accelerators provide $19 \times$ speedup over ARM and $2.9 \times$ speedup over Xeon
- Static scheduling alleviates the traditional Von-Neumann overheads
**Speedup in comparison to GPUs**

- **TABLA** generated accelerators provide **5.9x** speedup over Tegra K1, **1.16x** over GTX 650 Ti and **1.18x** slowdown over Tesla K40.
Performance-per-Watt in comparison to GPUs

• **TABLA** generated accelerators provide $17\times$ performance-per-Watt over Tegra K1, $20\times$ over GTX 650 Ti and $33\times$ over Tesla K40

• Specialized template designs extract fine-grained parallelism while consuming less power
Conclusions

• Exploit algorithmic commonalities to accelerate machine learning

• Create a template-based framework

• Abstract details of hardware from programmer

• Enable FPGAs to be accelerator of choice for machine learning

• [http://act-lab.org/artifacts/tabla](http://act-lab.org/artifacts/tabla)
Thank you