Multiclass Classification using SVMs on GPUs

Sergio Herrero

6.338J Applied Parallel Computing
Large Scale SVMs

Serial SVMs
- Osuna 1997
- Joachims 1999
- Platt 1999
- Keerthi 2001
- Fan 2005
- ...

Parallel/Multiprocessor SVMs
- Cao 2006
- Zanni 2006

Distributed/Cluster SVMs
- Graf 2005 (Cascade SVM)
- Lu 2008 (Yahoo)
- Chang 2006 (Google)

GPU SVMs
- Catanzaro 2008
Multiclass SVM

$l$ samples \((\vec{x}_1, y_1), \ldots, (\vec{x}_l, y_l)\) \(\vec{x}_i \in \mathbb{R}^n\), \(y_i \in Y \forall i\) \(Y = \{1, \ldots, M\}\)

\(R \quad M \times N\) Output code \(M\) classes \(N\) tasks \(R_{ij} \in \{-1, 0, 1\}\)

\(f^k(\vec{x}) = (\vec{x}_1, R_{y1k}), \ldots, (\vec{x}_l, R_{ylk})\) \(k = 1..N\) \(\hat{f}^k(\vec{x})\)

\(\hat{y} = \arg\max_{y \in Y} \left\{ \sum_{k=1}^{N} R_{yk} f^k(\vec{x}) \right\}\)

\(\hat{y} = \arg\min_{y \in Y} \left\{ \sum_{k=1}^{N} LOSS(R_{yk}, \hat{f}^k(\vec{x})) \right\}\)
GPUs: CUDA (I)

• CUDA Programming model
• Three key abstractions:
  – Hierarchy of thread groups
  – Shared memory
  – Barrier Synchronization
• Advantages:
  – High throughput in floating point computation (1 TFlop)
  – Aggressive Memory system (4 GB)
  – Fast memory bandwidth (102 GB/s)
GPUs: CUDA (II)

Host

Kernel 1

Device

Kernel 2

Grid 1
- Block (0,0)
- Block (0,1)
- Block (0,2)
- Block (0,3)
- Block (0,4)
- Block (1,0)
- Block (1,1)
- Block (1,2)
- Block (1,3)
- Block (1,4)

Grid 2
- Block (0,0)
- Block (0,1)
- Block (0,2)
- Block (1,0)
- Block (1,1)
- Block (2,0)
- Block (2,1)

Thread (x,y,z)
GPUs: CUDA (III)

Grid

Block (0,0)
- Shared Memory
- Registers
  - Thread (0,0)
  - Thread (1,0)

Block (1,0)
- Shared Memory
- Registers
  - Thread (0,0)
  - Thread (1,0)

Global Memory

Constant Memory

Host
Parallel SMO

\( I_0^k = \{i: y_i^k = 1, 0 < \alpha_i^k < C\} \cup \{i: y_i^k = -1, 0 < \alpha_i^k < C\} \)
\( I_1^k = \{i: y_i^k = 1, \alpha_i^k = 0\} \)
\( I_2^k = \{i: y_i^k = -1, \alpha_i^k = 0\} \)
\( I_3^k = \{i: y_i^k = 1, \alpha_i^k = C\} \)
\( I_4^k = \{i: y_i^k = -1, \alpha_i^k = C\} \)
\( f_{i}^{p,k} = \sum_{j=1}^{l} \alpha_j^k y_j^k (\bar{x}_j - \bar{x}_i) - y_i^k \)
\( b_{up}^{k} = \min\{f_{i}^{p,k}: i \in I_0^k \cup I_1^k \cup I_2^k \cup I_3^k \cup I_4^k \} \)
\( b_{low}^{k} = \max\{f_{i}^{p,k}: i \in I_0^k \cup I_1^k \cup I_2^k \cup I_3^k \cup I_4^k \} \)

 Initialize:
\( \alpha_i^k = 0, f_i^{p,k} = -y_i^k, i \in I_p^k, p = 1 \ldots P, k = 1 \ldots N \)

 Calculate:
\( b_{up}^{k}, I_{up}^{k}, b_{low}^{k}, I_{low}^{k}, p = 1 \ldots P, k = 1 \ldots N \)

 Obtain:
\( b_{up}^{k}, I_{up}^{k}, b_{low}^{k}, I_{low}^{k}, k = 1 \ldots N \)

 Iterate task \( k \) until \( b_{up}^{k} > b_{low}^{k} + 2\tau \)
Optimize \( \alpha_{i}^{up}, \alpha_{i}^{low} \)
Update \( f_i^{p,k}, p = 1 \ldots P \)
Calculate \( b_{up}^{k}, I_{up}^{k}, b_{low}^{k}, I_{low}^{k}, p = 1 \ldots P \)
Obtain \( b_{up}^{k}, I_{up}^{k}, b_{low}^{k}, I_{low}^{k} \)
Repeat
Parallel Tasks (I)

Kernel Caching (Joachims 1999)

$\alpha_{\text{up}}^{\text{new},k}$  $\alpha_{\text{low}}^{\text{new},k}$

AVA

OVA
Parallel Tasks (II)

Tasks

Subsets

Grid Reduction

# of iterations

Task #1
Converged

Task #2
Converged

Task #3
Converged

Task #4
Converged
Performance Results (I)

Host-Device Specifications:

<table>
<thead>
<tr>
<th>Host</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu 8.10 64bit</td>
<td>Tesla C1060</td>
</tr>
<tr>
<td>CPU: Intel Core i7 920 @ 2.67 GHz</td>
<td># Stream Processors: 240</td>
</tr>
<tr>
<td>Memory 6GB (3x2 DDR2)</td>
<td>Frequency of Processors: 1.3GHz</td>
</tr>
<tr>
<td>933 Gflops</td>
<td></td>
</tr>
<tr>
<td>Memory: 4GB DDR3</td>
<td></td>
</tr>
<tr>
<td>Memory Bandwidth: 102GB/s</td>
<td></td>
</tr>
</tbody>
</table>

Datasets:

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Training Points</th>
<th># Testing Points</th>
<th># Features</th>
<th># Classes</th>
<th>C</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>32,561</td>
<td>16,281</td>
<td>123</td>
<td>2</td>
<td>100</td>
<td>0.5</td>
</tr>
<tr>
<td>MNIST</td>
<td>60,000</td>
<td>10,000</td>
<td>780</td>
<td>10</td>
<td>10</td>
<td>0.125</td>
</tr>
</tbody>
</table>
Performance Results (II)

Kernel Cache Hit Rate

# Iterations

MNIST (OVA)

- 1 task
- 2 tasks
- 3 tasks
- 4 tasks
- 5 tasks
- 6 tasks
- 7 tasks
- 8 tasks
- 9 tasks
- 10 tasks
Performance Results (III)

Kernel Cache Hit Rate

# Iterations

5 tasks
15 tasks
25 tasks
35 tasks
45 tasks

MNIST (AVA)
Performance Results (IV)

Accuracy (Binary tasks):

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SVM</th>
<th>Accuracy (%)</th>
<th># SVs</th>
<th>Difference in b (%)</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>GPU</td>
<td>82.697624</td>
<td>18668</td>
<td>0.01</td>
<td>115565</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>82.697624</td>
<td>19058</td>
<td></td>
<td>43735</td>
</tr>
<tr>
<td>MNIST</td>
<td>GPU</td>
<td>96</td>
<td>43730</td>
<td>0.04</td>
<td>69535</td>
</tr>
<tr>
<td></td>
<td>LIBSVM</td>
<td>96</td>
<td>43756</td>
<td></td>
<td>76385</td>
</tr>
</tbody>
</table>

Training Time (Binary & Multiclass):

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GPU (sec)</th>
<th>LIBSVM (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>38.0542</td>
<td>479</td>
<td>12.58731</td>
</tr>
<tr>
<td>OVA (10 tasks)</td>
<td>AVA (45 tasks)</td>
<td>AVA (45 tasks)</td>
<td>~ 7 hours, 53 min</td>
</tr>
<tr>
<td>MNIST</td>
<td>2272.71</td>
<td>27833</td>
<td>22.86392</td>
</tr>
</tbody>
</table>

~ 20 min 😊   ~ 7 hours, 53 min 😞
Performance Results (V)

MNIST (OVA)
1172 Blocks per iteration

MNIST (AVA)
5274 Blocks per iteration
Conclusions:

- Naïve implementation of multiclass SVM:
  - One order of magnitude of speedup compared to LIBSVM
  - Room for improvement
    - Second order heuristics (Keerthi 2001)
    - Sparse matrices (Joachims 2006)
    - Parallel programming experience (me)

- Future work
  - Distributed SVM training on multi GPU scenarios (Graf 2005, Lu 2008)