A Reconfigurable Multiclass Support Vector Machine Architecture for Real-Time Embedded Systems Classification

Jason Kane, Robert Hernandez, and Qing Yang
University of Rhode Island
Overview

• Background
• Support Vector Machines Introduction
• Existing Classification Implementations
• R²SVM Design
• Performance Results
• Future Work
• Questions
Background

• URI Electrical/Biomed Dept. looked into lower limb prosthesis control with EMG
  – Intel i7 tower implementation prototyped in lab with high accuracy. Had RT issues.
  – FPGA prototype created to resolve RT problem.
Design Goal

• What if we want to perform multiple, accurate classifications in Real-Time?
  – Ex1: Run multiple models and compare outputs.
  – Ex2: Run multiple models for control of a number of subsystem items: Ankle, Knee, Arm, Etc.
Objectives

• Design a real-time energy efficient, accurate, general purpose run-time reconfigurable accelerator for SVM
  – Optimize data paths for pipelining/parallelization
  – Work with 4 common kernels
  – Support up to a maximum #Classes/Features, specified at compilation
  – Allow for targeting of diverse workloads

• Develop a prototype compatible with libsvm to provide direct performance comparisons with existing architectures.
  – Operate with FP in Real-Time

• Evaluate performance using publically available machine learning datasets.
SVM Introduction

• Support Vector Machines (Binary Classification SVM)
  – Classification technique using machine supervised learning
    • 1995 Vapnik & Cortes [1]
  – Training Data is supplied in vector form.
  – Data is mapped into hyperplanes in a high dimension space either directly (Linear SVM) or using a “kernel” function to remap the data into a transformed feature space (Non-Linear SVM).
SVM Introduction

• Support Vector Machines (Binary Classification)
  – Hyperplanes constructed based on **Largest Margin** between data of one class and another.
    • Points lying on this margin are termed “Support Vectors”.
    • Hyperplanes used to classify data
SVM Test Phase

Decision Function:

\[ f(x) = \text{sign} \left( \sum_{i=1}^{l} \alpha_i y_i k(x, x_i) + b \right) \]

Common SVM Kernels:

- **Linear Kernel**
  \[ k(x_i, x_j) = (x_i^T x_j) \]

- **Polynomial Kernel**
  \[ k(x_i, x_j) = (a x_i^T x_j + r)^d \]

- **Gaussian Kernel**
  \[ k(x_i, x_j) = e^{(-\gamma \|x_i - x_j\|^2)} \]

- **Sigmoid Kernel**
  \[ k(x_i, x_j) = \tanh(a x_i^T x_j + r) \]
Multi-Class SVM

• Multi-Class SVM
  – Extension of 1-Class SVM
  – Common Implementations
    • One Versus All - Classifier with highest output function wins.
    • One Versus One - Perform Multiple 1-Class SVM, Class with the most votes wins
Existing SVM Classifier Implementations

• CPU Software Based
  – Matlab libraries, libSVM, svmLite
  – No multithreaded versions

• GPU Based
  – Few public CUDA implementations: KMLib [2], GPUSVM. Neither support more than 2 classes.
  – We use a private library provided by the author of [3].

• FPGA Based
  – Mostly implementation specific designs.
  – None that support multi-class.
$R^2$SVM: High-Level System Architecture

$$f(x) = \text{sign}(\sum_{i=1}^{l} \alpha_i y_i k(x, x_i) + b)$$
Kernel Calculations

- **Goal:**
  - Support 4 Kernels
  - Simplify Routing
  - Faster Clock
  - Pipeline

- **Design:**
  - Input Block
  - Common Block
  - Output Block
Coefficient Weighting
Voting Unit

FP > 0?

Vote Control Logic

Class Counter A
Class Counter B
Valid Cmp
Last Cmp

Incr 1
Class 1 Votes

Incr 2
Class 2 Votes

Incr 3
Class 3 Votes

Incr ...
Class ... Votes

Incr K
Class K Votes

K-way Mux

Compare Class, Votes

Classification Result Valid

Max Votes Reg

Elected Class Reg

Elected Class Output
Stratix V Development Board
Performance Testing

• Single compilation could have been achieved for timing
  – Multiple were made to show performance under different class/feature settings.

• 6 Standard Datasets used for Testing from Statlog/UCI
  – Case Study with data from URI Human-Computer Control Interface

• Compare timing/accuracy with i7-2600 (3.4GHz) and GeForce GT 750M GPU
  – SVM Models loaded prior to test for all architecture
  – Report Prediction Time
## Prediction Results

<table>
<thead>
<tr>
<th>Trial (#Class, #Feat) [Source]</th>
<th>Sub-trial</th>
<th>#SV</th>
<th>FPGA Time (µS)</th>
<th>Average FPGA Speedup</th>
<th># Test Trials</th>
<th>Acc. FPGA (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult (2,123) LibSVM, UCI</td>
<td>Lin</td>
<td>600</td>
<td>13.4</td>
<td>37.3x 14.7x</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Poly</td>
<td>705</td>
<td>14.0</td>
<td>39.5x 23.3x</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>RBF</td>
<td>785</td>
<td>17.2</td>
<td>39.6x 14.0x</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Sig</td>
<td>687</td>
<td>13.5</td>
<td>42.8x 17.2x</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>DNA (3,180) Statlog</td>
<td>Lin</td>
<td>402</td>
<td>10.0</td>
<td>39.7x 17.7x</td>
<td>1187</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Poly</td>
<td>1053</td>
<td>21.8</td>
<td>39.1x 18.3x</td>
<td>1187</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>RBF</td>
<td>696</td>
<td>13.5</td>
<td>53.7x 21.1x</td>
<td>1187</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Sig</td>
<td>444</td>
<td>9.4</td>
<td>48.7x 19.7x</td>
<td>1187</td>
<td>100</td>
</tr>
<tr>
<td>Letter (26,16) Statlog</td>
<td>Lin</td>
<td>6060</td>
<td>125.0</td>
<td>9.0x 6.6x</td>
<td>5000</td>
<td>99.97</td>
</tr>
<tr>
<td></td>
<td>Poly</td>
<td>4408</td>
<td>100.0</td>
<td>10.0x 13.5x</td>
<td>5000</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>RBF</td>
<td>5701</td>
<td>120.0</td>
<td>9.6x 7.2x</td>
<td>5000</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Sig</td>
<td>14602</td>
<td>266.0</td>
<td>14.4x 13.7x</td>
<td>5000</td>
<td>100</td>
</tr>
<tr>
<td>Shuttle (7,9) Statlog</td>
<td>Lin</td>
<td>3994</td>
<td>72.7</td>
<td>5.96x 6.9x</td>
<td>14500</td>
<td>99.97</td>
</tr>
<tr>
<td></td>
<td>Poly</td>
<td>878</td>
<td>21.7</td>
<td>7.4x 14.9x</td>
<td>14500</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>RBF</td>
<td>2208</td>
<td>43.6</td>
<td>7.1x 8.1x</td>
<td>14500</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Sig</td>
<td>4290</td>
<td>78.4</td>
<td>10.2x 8.6x</td>
<td>14500</td>
<td>100</td>
</tr>
<tr>
<td>Satimage (6,36) Statlog</td>
<td>Lin</td>
<td>1216</td>
<td>25.6</td>
<td>11.5x 9.5x</td>
<td>2000</td>
<td>99.95</td>
</tr>
<tr>
<td></td>
<td>Poly</td>
<td>1051</td>
<td>21.7</td>
<td>13.8x 18.1x</td>
<td>2000</td>
<td>99.95</td>
</tr>
<tr>
<td></td>
<td>RBF</td>
<td>1165</td>
<td>23.3</td>
<td>13.5x 11.0x</td>
<td>2000</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Sig</td>
<td>2468</td>
<td>47.3</td>
<td>15.2x 10.4x</td>
<td>2000</td>
<td>100</td>
</tr>
<tr>
<td>Vowel (11,10) UCI</td>
<td>Lin</td>
<td>268</td>
<td>13.7</td>
<td>4.0x 12.1x</td>
<td>463</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Poly</td>
<td>299</td>
<td>13.0</td>
<td>5.1x 6.72x</td>
<td>463</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>RBF</td>
<td>290</td>
<td>56.8</td>
<td>4.0x 11.8x</td>
<td>463</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Sig</td>
<td>427</td>
<td>17.1</td>
<td>6.9x 11.4x</td>
<td>463</td>
<td>100</td>
</tr>
<tr>
<td>Arm1 (7,20) C. Study</td>
<td>RBF</td>
<td>443</td>
<td>14.1</td>
<td>9.1x 11.3x</td>
<td>1818</td>
<td>100</td>
</tr>
<tr>
<td>Arm2 (7,20) C. Study</td>
<td>RBF</td>
<td>572</td>
<td>14.1</td>
<td>8.9x 12.0x</td>
<td>1818</td>
<td>100</td>
</tr>
<tr>
<td>Arm3 (7,20) C. Study</td>
<td>RBF</td>
<td>435</td>
<td>14.0</td>
<td>8.6x 11.4x</td>
<td>1818</td>
<td>100</td>
</tr>
</tbody>
</table>
Prediction Results

• Very few Prediction Mismatches
  – Examined data, determined Single/Double FP Precision issue
  – Double Precision could be implemented
    • Worthwhile?
• FPGA Timing More Consistent than CPU
Conclusions and Future Work

• $R^2$SVM FPGA Prototype Demonstrated Benefits of Architecture
  – Scalability
  – Compatibility with libsvm models
  – Presented performance against CPU/GPU for several real-world datasets
    • Up to 53x faster than CPU
    • Up to 23x faster than GPU

• Future Work
  – Kernel Hardware Alternatives?
  – ASIC optimization?
Questions?
References

