Acceleration of multivariate analysis techniques in TMVA using GPUs

International Conference on Computing in High Energy and Nuclear Physics 2012

A. Hoecker, H. McKendrick, J. Theraag, A. Washbrook

University of Edinburgh

24th May 2012
Outline

1. TMVA
2. Artificial Neural Networks
3. Parallelism Approaches
4. Results
5. Discussion
TMVA enables training, testing and performance evaluation of several multivariate classification (and regression) techniques

Specifically designed (but not restricted to) the needs of high-energy physics

**Supervised learning** - training events are used to determine a mapping function to describe a decision boundary

| Rectangular cut optimisation | Projective likelihood estimator (PDE) | Multi-dimensional likelihood estimator | Likelihood estimator using self-adapting phase-space | Support Vector Machines |
|-----------------------------|---------------------------------------|----------------------------------------|----------------------------------------------------|------------------------|
| K-nearest neighbour classifier | H-Matrix discriminant | Linear Discriminant analysis | Artificial Neural Networks | Boosted Decision Trees |
TMVA Workflow

User Training Script

- create
- create
- execute
- execute
- execute
- execute
- execute

ROOT Target File

- uses
- API
- API
- API
- API
- API

User Application Script

- create
- execute
- execute
- execute
- execute
- execute

TMVA::Reader

- API
- API
- API
- API

1. Begin event loop
2. Update event
3. Train MVAs
4. Test MVAs
5. Evaluate MVAs
6. End event loop

- Add Variables
- Add Variables
- Initialise Training and Test Trees
- Book MVA kType, Options
- Book MVA weight file to read
- Compute MVA

Andrew Washbrook

TMVA Acceleration using GPUs
Feasibility Study

- Select one classification method and investigate performance improvements
- Evaluate steps needed for parallelisation
- Determine if methods can be applied to other classification techniques

The MLP Artificial Neural Network technique was chosen for study
Artificial Neural Networks

- Artificial Neural Networks (ANNs) are a biologically inspired machine learning technique to model relationships between input and output data.
- The network is trained to classify input data by the adjustment of connected synapse weights used in neuron activation and response functions.

Overtraining
Multi-layer perceptrons (MLPs) are Feed-forward neural networks that pass data in one direction between input and output, with no loops or cycles

**MLP Calculation Method**

- Events sequentially fed through the network
- Selection of event variables used as input to the first layer of neurons
- Neurons take a number of weighted inputs through their synapses, to form a single output value passed on to the next layer
- **Supervised learning** - results from output layer is used to train and improve the network through back propagation of training errors
- Network is trained over a number of "epochs"
Can the MLP calculation be parallelised (on GPUs)?

- Event-based parallelism
  Implicit training dependency from prior events

- Neuron-based parallelism
  Simultaneous calculation of neuron inputs, functions and error calculations

Hot spot analysis
Traversal of array classes is a significant proportion of the processing time.

### Cumulative Percentage of Processing Time

| % of Total Time | Function                           |
|-----------------|------------------------------------|
| 100             | main                               |
| 97.5            | TrainAllMethods                    |
| 96.4            | TrainMethod                        |
| 96.1            | Train                              |
| 96.0            | BackPropogationMinimize            |
| 83.7            | TrainOneEpoch                      |
| 83.1            | TrainOneEvent                      |
| 57.4            | UpdateNetwork                      |
| 30.8            | ForceNetworkCalculations           |

### Percentage of Processing Time

| % of Total Time | Function                           |
|-----------------|------------------------------------|
| 8.10            | TobjArrayIter::Next                |
| 6.07            | TMVA::TSynapse::CalculateDelta     |
| 4.53            | TobjArray::At                      |
| 3.80            | tanh                               |
| 3.36            | TMVA::TSynapse::AdjustWeight       |
| 3.28            | TMVA::TSynapse::GetWeightedValue   |
| 2.92            | TMVA::TNeuronInputSum::GetInput    |
| 2.34            | malloc                             |
| 2.33            | TMVA::TNeuron::CalculateDelta      |
GPGPUs

- GPUs are being successfully leveraged for general purpose computing and are yielding large performance gains across a number of disciplines.

- Now being adopted in High Energy Physics – especially for time-critical environments such as the ATLAS trigger.

Memory Hierarchy

- Data must be copied to the device before the kernel is invoked.
- Global memory contents retained between kernel operations. Typically O(GB) in size but with low bandwidth.
- Each thread block has access to its own shared memory for the duration of a kernel call. Typically 16-48 KB in size with higher bandwidth.
Two input data samples were used for performance comparisons

Large sample representative of input data used in Higgs analysis

Access to two GPU-enabled servers (note different CPU and GPU models)
### Timing Comparison

#### Setup 1: Intel Xeon X5560 + Nvidia Tesla C1060

| Sample Type | CPU Classification Time | CPU + GPU Classification Time |
|-------------|--------------------------|-------------------------------|
| Small       | 19 sec                   | 121 sec                       |
| Large       | 930 sec                  | 667 sec                       |

#### Setup 2: Intel Xeon E5502 + Nvidia Tesla C2050 (Fermi)

| Sample Type | CPU Classification Time | CPU + GPU Classification Time |
|-------------|--------------------------|-------------------------------|
| Small       | 34 sec                   | 223 sec                       |
| Large       | 1830 sec                 | 1180 sec                      |

**Why are the results inconsistent?**

- GPU utilisation is low in small data sample
- Larger proportion of execution time in kernel initialisation and host to device event transfer
- Speed-up observed as network complexity increases
Event and Epoch Scaling

Number of events and training epochs scales in the same way for both CPU and GPU methods.
Increase in hidden layers (and neurons) does not significantly affect run time for GPU based technique
Parallel Network Training

- Training networks can be run simultaneously on the GPU
- Global memory exhaustion observed over 128 networks
- Use shared memory instead to scale to any number of MP and devices

Why train multiple networks with the same events?

![MLP Classification Time: Number of Networks](chart.png)
Classification power of network depends on choice of input parameters

Use network parallelism as an optimisation technique to determine best parameters for a given training set
Nvidia Kepler GPU

3x
Energy-Efficient Performance

32 cores
192 cores

NVIDIA VGX
GPU-Accelerated VDI
Nvidia Kepler GPU

Dynamic Parallelism

3x Energy-Efficient Performance

NVIDIA VGX GPU-Accelerated VDI

Dynamic Parallelism

Too coarse
Too fine
Adaptive
Improvements

Bias Nodes

- Additional neuron in each of the non-output layers of the network used to shift the activation function $\implies$ faster or superior convergence
- Inclusion causes minor branching in kernel code
- Needs to be included to get equivalent classification results
Improvements

GPU utilisation

- Use shared memory for kernel operations for better performance and inter-device flexibility
- Tune for newer GPU devices (use device cache more effectively)

TMVA Portability

- Incorporate parallel methods for use by other classification techniques

Lots of work needed

- Convert OO data structure to data pipeline
- Kernel specific implementations of each classification method
- Large scale codebase change or "acceleration library"?
Conclusions

- Feasibility study into the acceleration of MLP ANN using GPUs has shown encouraging results.
- Event-based parallelism not possible but speed-up found depending on the complexity of the network.
- Multiple networks can be run simultaneously which could give a qualitative performance gain by input parameter scanning.
- Emerging GPU device features - such as adaptive parallelism and visualisation - may also aid performance in this area.