GPU Accelerated Backtesting and ML for Quant Trading Strategies

GTC 2015 | San José | California

Dr. Daniel Egloff
daniel.egloff@quantalea.net
March 18, 2015
Motivation

- **Goals**
  - Execute automated algorithmic trading strategies
  - Optimize risk return

- **Procedure**
  - Extract signals and build price forecasting indicators from market data
  - Transform indicators into buy / sell decisions
  - Apply portfolio risk management

- **Challenges**
  - Find relevant signals and indicators
  - Engineer and parameterize trading decision
  - Find optimal parameters

- **Approach**
  - Exploit parallelism in the computations
  - Accelerate calculations by using a GPU cluster
Algo Trading Strategies

Market data → Mathematical operations → Trading decision

Input

Output

Configurations
Example

- **Buy signal**
- **Sell signal**

**Chart Description:**
- Fast moving average
- Slow moving average

**Excel Table:**

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>EUR/USD</th>
</tr>
</thead>
<tbody>
<tr>
<td>04/12/2012</td>
<td>22:00</td>
<td>1.3084</td>
</tr>
<tr>
<td>04/13/2012</td>
<td>17:00</td>
<td>1.3119</td>
</tr>
<tr>
<td>04/16/2012</td>
<td>07:00</td>
<td>1.3123</td>
</tr>
<tr>
<td>04/16/2012</td>
<td>21:00</td>
<td>0.0065</td>
</tr>
<tr>
<td>04/17/2012</td>
<td>11:00</td>
<td>0.0065</td>
</tr>
<tr>
<td>04/18/2012</td>
<td>01:00</td>
<td>0.0065</td>
</tr>
<tr>
<td>04/18/2012</td>
<td>15:00</td>
<td>0.0065</td>
</tr>
<tr>
<td>04/19/2012</td>
<td>14:00</td>
<td>0.0065</td>
</tr>
</tbody>
</table>
Markets

- Futures market
  - CME 50 liquid futures
  - Other exchanges

- Equity markets
  - World stock indices

- FX markets
  - 10 major currency pairs
  - 30 alternative currency pairs

- Options markets
  - Options on futures
  - Options on indices
Strategy Universe

Strategy configurations

Configuration $c$  Trading decision $s(c)$  Utility $U(s(c))$

1 buy
-1 sell

P&L / drawdown
Challenge 1: How can we engineer a strategy producing buy / sell decisions?
Random Forests
Random Forests

Strategy configuration c

Features ➔ Random forest ➔ Trading decision \( s(c) \)

1 buy

-1 sell
Training Random Forests

Bootstrapping to create training sets

C 4.5 algorithm for individual tree construction

- Selecting subset of features for tree construction
- Each node is associated with a subset of training samples
- Recursive, starting at the root node
- At each node execute divide and conquer algorithm to find locally optimal choice
  - If samples are in same class (or few class) node is a leaf associated with that class
  - If samples are in two or more classes
    - Calculate information gain for each feature
    - Select feature if largest information gain for splitting
Entropy

\[ T = \text{set of samples associated with node} \]

\[ C_1, ..., C_n = \text{classes of samples} \]

Entropy

\[
Ent(T) = - \sum_{i=1}^{n} \frac{freq(C_i,T)}{|T|} \log_2 \left( \frac{freq(C_i,T)}{|T|} \right)
\]

- Characterizes impurity of samples
- Measure of uncertainty
- Additive: impurity of several subsets is sum of impurities

![Graph showing the relationship between \(\text{Pr}(X = 1)\) and entropy.](image)
Information Gain

\( T_1, \ldots, T_s = \) subsets of \( T \) generated by splitting on selected attribute

Information gain discrete feature

\[
\text{gain}(T_1, \ldots, T_s) = \text{Ent}(T) - \sum_{i=1}^{s} \frac{|T_i|}{|T|} \text{Ent}(T_i)
\]

Information gain continuous feature with optimal splitting threshold

\[
\text{gain}(t) = \text{gain}(T_{\leq t}, T_{> t})
\]

\( t_* = \text{argmax} \ \text{gain}(t) \)

Actual implementation uses ratio information gain over split ratio
Training Individual Trees

Labels from positive or negative market returns

All features

Selected features

Permutations to sort features

Permutated labels

Samples / observations

Label

Selected features
Training Individual Trees

- All features
  - Selected features
    - Weights
      - 201402 .......
    - Permutations to sort features
    - Permutations to sort features
  - Permutations labels
    - Permuted labels
    - Permuted weights
  - Label
Training Individual Trees

1. Optimal split $t_i$

2. Optimal feature $F_i$

Entropy criterion for best feature and split

Permuted labels

Permuted weights

Entropy for every split

$F_i < t_i$
Training Individual Trees

Recursively refine classification: mask data according to classification.

Selected features → Permutations sort features → Permuted labels

Weight → Permutations sort features → Permuted weights
Training Individual Trees

Recursively refine classification: mask data according to classification
GPU Implementation

- **Parallelism at multiple levels**
  - Multiple trees, one for each set of weights
  - Independent features
  - Independent split points
  - Multiple nodes further down the tree

- **GPU kernels can be implemented with standard primitives**
  - Random number generation for weights
  - Parallel scan (cumulative sum)
  - Parallel map
  - Parallel reduction to find optimal feature and split
Strategy Backtesting

Challenge2: How to choose best trading strategy?
Walk Forward Optimization

3-6 months

1 month

In sample

Out of sample

In sample

Out of sample

In sample

Out of sample

shift by 1 month
Trading P&L

Market returns \( r(t_i) \) = \( \log(P(t_i) / P(t_{i-1})) \)

Trading decision \( s(c) \)

P&L(c) = \( <s(c), r> \)
**Optimal Configuration**

<table>
<thead>
<tr>
<th>Configurations</th>
<th>P&amp;L</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(c)</td>
<td></td>
</tr>
<tr>
<td>s(c)</td>
<td></td>
</tr>
<tr>
<td>s(c)</td>
<td></td>
</tr>
<tr>
<td>s(c)</td>
<td></td>
</tr>
<tr>
<td>s(c)</td>
<td></td>
</tr>
<tr>
<td>s(c)</td>
<td></td>
</tr>
<tr>
<td>s(c)</td>
<td></td>
</tr>
<tr>
<td>s(c)</td>
<td></td>
</tr>
<tr>
<td>s(c)</td>
<td></td>
</tr>
<tr>
<td>s(c)</td>
<td></td>
</tr>
</tbody>
</table>

*pick configuration with largest P&L*
Bootstrapping Trading P&L

Configurations

- s(c)
- s(c)
- s(c)
- s(c)
- s(c)
- s(c)

Weights

\[ r \cdot w \]

P&L

- x
- x
- x
- x

Pick configuration with largest P&L
Hypothesis Tests

Null Hypothesis
Trading P&L <= 0

Alternative Hypothesis
Trading P&L > 0
Trading P&L Distribution

optimal configurations for each weight

c* for w = 1
White Reality Check

- Scale exposure according to distance from 0
- Do not trade if negative returns

Market 1

\[ c^* \text{ for } w = 1 \]

…………

Market n

\[ c^* \text{ for } w = 1 \]
GPU Implementation

- Parallelism at multiple levels
  - Multiple markets
  - Independent in-sample / out-of-sample windows
  - Independent strategy configurations
  - Independent time steps for utility functions such as mean return

- GPU kernels can be implemented with standard primitives
  - Random number generation
  - Matrix multiplication (almost, up to return vector scaling the weights)
  - Parallel reduction
GPU Implementation

- GPU grid
  - Multiple markets
  - Independent in-sample / out-of-sample windows

- Per GPU
  - Independent strategy configurations
  - Independent time steps for utility functions such as mean return
Questions ?