A **GPU-based** framework for multivariate analysis in particle physics

Xuefeng Ding$^{1,2}$

1. Gran Sasso Science Institute, L’Aquila, Italy
2. INFN Sezione di Milano, Milan, Italy

PHYSSTAT-nu 2019
@ CERN, Geneva, Switzerland 25 January 2019
Outline

- The GooStats framework
- Motivation: Borexino spectrum fit
- Package design
- Systematic uncertainty estimation
- Conclusion
GooStats hosted on GitHub

https://github.com/GooStats/GooStats.git

Simplify CMakeLists.txt

- DingXuefeng released this on Nov 25, 2018 · 3 commits to master since this release

- Source code (zip)
- Source code (tar.gz)

- Simplify CMakeLists.txt
- Add `simpleFit` and `statAnalysis` projects, for the JINST paper
What and Why?

- A convenient GPU multivariate analysis framework
- Easy spectrum fitting

```bash
./fit toyMC.cfg out exposure=500
```

**Figure 7.** Left: screen shot of fit result summary. Right: produced figure in the file format of pdf.
GooStats: performance

- Speed up: around 200 times

**Table 1.** Comparison of fitting time between GooStats and original software used by the Borexino collaboration. $T_{\text{tot}}$: total execution time. $N$: the number of iterations taken to converge in MINUIT. $T_{\text{it}}$: average execution time per iteration. Speed up: $T_{\text{it}}(\text{CPU})/T_{\text{it}}(\text{GPU})$.

<table>
<thead>
<tr>
<th>Type Size</th>
<th>CPU</th>
<th>GPU</th>
<th>Speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AMD Opteron(TM) Processor 6238</td>
<td>nVidia Tesla K20m</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$T_{\text{tot}}$ (s)</td>
<td>$N$</td>
<td>$T_{\text{it}}$ (ms)</td>
</tr>
<tr>
<td>400</td>
<td>27.6</td>
<td>1128</td>
<td>24.4</td>
</tr>
<tr>
<td>350</td>
<td>29.4</td>
<td>1331</td>
<td>22.1</td>
</tr>
<tr>
<td>300</td>
<td>22.5</td>
<td>1239</td>
<td>18.2</td>
</tr>
</tbody>
</table>
Motivation: Borexino spectrum fit

- Motivation: spectral-fitting tool for Borexino analysis
Analytical Response Function

\[ M : f(E) \mapsto g(\text{charge}) = \int_{0}^{E_{\text{end}}} dE \cdot f(E) \cdot \text{RPF} [\text{charge}; \mu(E), \text{var}(\mu)] \]

- Analytical shape of spectrum of mono-energetic events
  - **Momentum based approximation**
  - Match the average (energy scale + non-linearity model)
  - Match the variance (energy resolution model)
  - \( \ldots \rightarrow \text{simplified} \)
  - More: “Mask”, “pile-up” etc…
- We can simplify because
  - Borexino response is simple: small FV in center, low energies \( \Rightarrow \) no irregular tail
  - We are not sensitive.. \( \Rightarrow \) small systematics
- **Fit full MC to get the bias introduced in simplification**
Multivariate Likelihood (Davini method)

\[ \mathcal{L}_{\text{MV}} \left( \vec{\theta} \right) = \mathcal{L}_{\text{TFC-sub}} \left( \vec{\theta} \right) \cdot \mathcal{L}_{\text{TFC-tagged}} \left( \vec{\theta} \right) \cdot \mathcal{L}_{\text{RD}} \left( \vec{\theta} \right) \cdot \mathcal{L}_{\text{PS}} \left( \vec{\theta} \right) \]

- Scaling factor introduced to remove bias.
• **Middle Layer** between **GooFit** and **User module**

![Diagram](image-url)

- **GooStats**
  - **InputManager**: collect inputs and build GooPdf object
  - **AnalysisManager**: control the work flow
  - **OutputManager**: save and visualize fit results
- **InputBuilder**: actual object that builds the inputs
- **BatchOutputManager**: TTree and text output
- **PlotManager**: Figure output in the format of pdf and root
- **OutputBuilder**: actual object that builds the outputs
- **GooPdf**: calculating the likelihood/χ²
- **FitManager**: minimize FCN
- **ROOT**: TMinuit
Parameter Synchronization

- Synchronize fit parameters in joint analysis
- Supported by tree type internal data structure
All fit parameters saved in TTree output automatically

More quantities can be added as lambda functions in the output builder

```cpp
root [1] .ls
TFile** test_tree.root
TFile* test_tree.root
  KEY: TTree fit_result;1 Fit result of GooStats
root [2] fit_result->Show(0)

EVENT:0
  default.NReactor = 3.02987
  default.NReactor_err = 0.0159146
  default.Nbkg = 1.02405
  default.Nbkg_err = 0.0162459
  default.U235 = 0.5
  default.U235_err = 0
  default.U238 = 0.2
  default.U238_err = 0
  default.Pu239 = 0.1
  default.Pu239_err = 0
  default.U241 = 0.2
  default.U241_err = 0
  default.LY = 1300
  default.LY_err = 0
  default.qc1 = 2.78788
  default.qc1_err = 0
  default.qc2 = -0.528003
  default.qc2_err = 0
  default.v1 = 0.3
  default.v1_err = 0
  default.vT = 5
  default.vT_err = 0
  default.Reactor_dEvis = 1078.28
  default.Reactor_dEvis_err = 13.2291
  chi2 = 390.448
  NDF = 397
  likelihood = 1883.96
```
Statistical analysis modules

- Adding tasks of statistical analysis by registering new modules inheriting abstract classes

- A few pre-installed modules: ScanPar, DiscoveryTest.

```cpp
PrepareData *data = new PrepareData();
SimpleFit *fit = new SimpleFit();
DiscoveryTest *discovery = new DiscoveryTest();

ana->registerModule(inputManager);
ana->registerModule(data);
ana->registerModule(fit);
ana->registerModule(discovery);
ana->registerModule(outManager);
```
A GPU-based framework for multi-variate analysis in particle physics, Xuefeng Ding

### Physics modules

#### Solar Neutrinos

- Default: Major from 1.8119e+07 days × tons
- Total $\chi^2$/NDF 3436.8 / 3294 p-value 0.041
- $^7$Be cosmo = 24.100 ± 0.019 cpd/100 tons
- $^{10}$B = 118.02 ± 0.21 cpd/100 tons
- $^3$H = 194.216 ± 0.040 cpd/100 tons
- $^{11}$C qch = 0.88 [Fixed]
- $^{11}$C = 165.522 ± 0.017 Bq/ktons
- $^3$He pileup = 5555.5 ± 7.1 cpd/100 tons
- $^6$Li = 22.75 ± 0.11 cpd/100 tons
- $^8$Be = 49.84 ± 0.32 cpd/100 tons
- $^{232}$Th chain = 20 cpd/100 tons [Fixed]
- $^{238}$U chain = 78.4 cpd/100 tons [Fixed]
- $\nu^0(B) = 0.46$ cpd/100 tons [Fixed]
- $\nu^0(Be)_{862} = 46.24 ± 0.15$ cpd/100 tons
- $\nu^0(Be)_{862} = 1.9769 ± 0.0062$ cpd/100 tons
- $\nu^0(CNO) = 4.29 ± 0.097$ cpd/100 tons
- $\nu^0(\nu) = 138.0 ± 5.1$ cpd/100 tons

#### JUNO style

- main from 43829.1 days × tons
- $\chi^2$/NDF 154.8 / 168 p-value 0.758
- Reactor = 1.0015 ± 0.0024 cpd/kttons

---

**charge (LY=1340.0±0.0 p.e./MeV, RPF[1.30±0.00],[2.20±0.00],[0±0])**

**$E_{vis}$(MeV)**
Two ways of estimating $\sigma_{sys}$

- create an ensemble of models according to experiment precision
- Fit the data with varying models through their coordinates
- Generate pseudo-experiments with varying models and take the width of distribution of best-fit
Method 1: fit with varying models

- **Full MC**: during each iteration of the fit, vary $\text{kb} / \text{absorption length spectrum etc.}$ re-simulate and produce new pdf on the fly $\Rightarrow$ when one day computer is fast enough
  - $\sim 200,000$ CPU x years per fit

- **Semi-analytical**: analytical non-linearity model + response Matrix
  - $\sim 30$ minutes per fit

- **Full analytical**
  - $\sim 2$ hours per fit
Problem with scaling response matrix

- Systematic uncertainties of LY can be included by scaling the response matrix.
- This could be dangerous if the interested parameters is correlated with the resolution parameters.
- When LY is scaled, the resolution should also be changed.
Using full analytical response function, we can see the correlation with detector responses.
Method 2: Monte Carlo method

- pseudo-experiment spectra without distortion —> **statistical sensitivity**

- pseudo-experiment spectra with distortion —> **statistical systematic uncertainty**
Conclusion

• GooStats, a GPU based multivariate analysis framework is introduced.

• Statistical analysis module is easy to be implemented as needed. TTree/figure output provided.

• Full analytical response function has advantage that it treated the NL and resolution in a coherent way when evaluating the systematic uncertainties.

• The systematic uncertainties can also be evaluated using Monte Carlo method by looking at change of width of best fit with/without distortion.