Misura della massa del W con tecniche di Big Data

Elisabetta Manca - Scuola Normale Superiore, INFN Pisa & CERN
Chapter 3. Measuring the $W$ mass

3.2.1 Lepton transverse momentum and transverse mass

The process under study is the following:

$p + p \rightarrow W + X$ and $W \rightarrow l + \nu$

where $X$, called the "hadronic recoil", is the set of all other particles produced in the collision, typically from the hadronization of the proton remnants and of the gluon initial state radiation.

At the generator level, meaning with no reconstruction or pileup effects, the recoil momentum is equal and opposite to the $W$ one. The $W$ boson then decays into a lepton and a neutrino: the former is well measured, whereas the latter cannot be detected. The two main experimental variables sensitive to the $W$ mass are the lepton transverse momentum and the transverse mass.

To explain the reason why these variables are useful, we can proceed by steps. To start with, suppose that the $W$ boson is produced with no transverse momentum: in this case, in its rest frame, the lepton energy is half of the $W$ mass, $2\frac{m_W}{2}$, and the lepton transverse momentum is related with its decay angle; when the decay is orthogonal with respect to the beam-line the lepton transverse momentum reaches its maximum value, equal to half of the $W$ mass. But since the lepton transverse momentum is invariant under boosts along the beam axis, a measurement of the distribution of the lepton transverse momentum gives a measurement of the $W$ mass.

This is exactly true only in the assumption that the lepton mass can be neglected, that is perfectly verified. The correction to the transverse momentum due to the lepton mass scales like $m_l^2 / m_W^2$. In the case of the muon this correction is larger than the electron, and is of the order of $10^6$, much smaller than the target precision for this measurement.

Both plots show events simulated with Pythia8 standalone (for further details about this software see Ref. [33]).
however transverse quantities are not Lorentz invariant measurement is strongly dependent on the production model.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_T - p_T^\mu$, $W^\pm$, $e-\mu$</td>
<td>80369.5</td>
<td>6.8</td>
<td>6.6</td>
<td>6.4</td>
<td>2.9</td>
<td>4.5</td>
<td>8.3</td>
<td>5.5</td>
<td>9.2</td>
<td>18.5</td>
<td>29/27</td>
</tr>
</tbody>
</table>

uncertainty on ATLAS measurement dominated by theory modelling.
in our paper we have shown that it is possible to unfold the rapidity distribution of the W directly from data.

About the rapidity and helicity distributions of the W bosons produced at LHC

error bars in the fit smaller than PDF prediction!
what about transverse momentum?

things start to look much more complicated…

\[ A_i \text{ coefficients are functions of } W \text{ production and multiply spherical harmonics of 2nd order} \]

\[
\frac{d\sigma}{dp_T^2 dY d\cos \theta d\phi} = \frac{3}{16\pi} \frac{d\sigma^{U+L}}{dp_T^2 dY} [(1 + \cos^2 \theta) + 1/2 A_0 (1 - 3 \cos^2 \theta) + A_1 \sin(2\theta) \cos \phi + 1/2 A_2 \sin^2 \theta \cos(2\phi) + A_3 \sin \theta \cos \phi + A_4 \cos \theta + A_5 \sin^2 \theta \sin(2\phi) + A_6 \sin(2\theta) \sin \phi + A_7 \sin \theta \sin \phi] \]
given $y$ and $W \ p_T$, a lepton in the lab frame has a $p_T$, $\eta$ and a $\phi$ (with respect to the $\phi$ of the $W$)

however the three variables have a constraint (in rest frame $E^* = M/2$)

$y = 0, \ W \ p_T = 20 \text{ GeV}$
given $y$ and $W \ p_T$, a lepton in the lab frame has a $p_T$, $\eta$ and a $\phi$ (with respect to the $\phi$ of the $W$).

however the three variables have a constraint (in rest frame $E^*=M/2$)

$y = 0$, $W \ p_T = 20$ GeV
\[ y = 0, \ W \ p_T = 20 \text{ GeV} \]
angular coefficients templates
changing basis from helicity amplitudes to angular coefficients

\[
\frac{d\sigma_{W\pm}}{d|y|}_{LR,0} \rightarrow \frac{d\sigma_{W\pm}^{UL}}{d|y|dq_T}, \quad A_{0,1,2,3,4}^\pm(q_T, |y|), \quad M_W
\]

and adding a finer binning in W pT

it is possible to measure simultaneously the double differential distribution of the W rapidity and pT \textit{and its mass}
Uncertainty on PDF negligible (<3 MeV)

Uncertainty on scale much reduced (residual unc. likely due to stat. fluctuation)

projections to data sample collected by CMS in full Run2 show 5 MeV uncertainty is within reach
### CMS data taking period

<table>
<thead>
<tr>
<th></th>
<th>2016</th>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of events into acceptance</td>
<td>35 fb$^{-1}$</td>
<td>45 fb$^{-1}$</td>
<td>65 fb$^{-1}$</td>
</tr>
<tr>
<td></td>
<td>~100 M</td>
<td>~130 M</td>
<td>~185 M</td>
</tr>
</tbody>
</table>

NB: for the analysis to work, MC must have ~10 more statistics than data!

**do we have the tools to process such a huge number of events?**
the new ROOT interface for expressing parallelism

ROOT

Datasource

ROOT
CSV
Apache Arrow
SQLite
ATLAS' xAOD

Range
Filter

Define

p_\text{x} \quad p_\text{y} \quad p_\text{z} \quad E \quad \text{myvar}

Datasource

Results

histograms, profiles

new ROOT files

cut-flow reports

data reductions
(mean, sum,..)

any user-defined operation

your analysis as a graph of operations

lazy execution guarantees that the event loop is run only once

developed to be transparently parallelisable through Implicit MultiThreading

Write datasets to disk, also in parallel.

lazy execution guarantees that the event loop is run only once

developed to be transparently parallelisable through Implicit MultiThreading
the idea: wrap RDF into a mini-framework that executes some modules following a graph path

a group of actions on RDF

= i.e. apply a calibration, filter events, fill histograms....
some action

= a python class
= a C++ class (inherits from a virtual mother class)

class filter:
    def __init__(self, string):
        self.myTH1 = []
        self.myTH2 = []
        self.myTH3 = []
        self.string = string
    def run(self,d):
        self.d = d.Filter(self.string)
        return self.d
    def getTH1(self):
        return self.myTH1
    def getTH2(self):
        return self.myTH2
    def getTH3(self):
        return self.myTH3

#include "filter.h"
RNode filter::run(RNode d){
    auto d1=d.Filter("Mystring_");
    return d1
}
std::vector<ROOT::RDF::RResultPtr<TH1D>> filter::getTH1(){
    return _h1List;
}
}
std::vector<ROOT::RDF::RResultPtr<TH2D>> filter::getTH2(){
    return _h2List;
}
}
std::vector<ROOT::RDF::RResultPtr<TH3D>> filter::getTH3(){
    return _h3List;
}
}
an example of action: extraction of the “Angular Coefficients” from a WJets Montecarlo

spherical harmonics 2nd order (W has spin 1!)

\[ m = \frac{\sum P_k(\cos \theta, \phi)w_i}{\sum w_i} \]

for each bin of W p_{T} and y

for each harmonics \( k = 0, \ldots, 7 \)

\[ \sigma_m = \sqrt{\frac{\sum w_i^2}{\sum w_i}} \]

implementation:

- 1 TH2 filled with \( w \)

in RDF language:

about 10 Filters and 10 Defines

for each harmonics (0 to 7):

- 1 TH2 filled with \( P_k \) and weighted with \( w \)
- 1 TH2 filled with \( P_k^2 \) (to compute variance)
results

a scaling plot: 2*192 cores Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz
Lots of SSD storage, bleeding edge hardware
@ Scuola Normale Superiore
27 GB input data (~4000 cluster)

currently investigating bottlenecks over 100 cores

dates at level of MHz!!
full data statistics can be processed in 5 minutes