Machine Learning for Workflow Handling

Daniel Abercrombie, *MIT*
Allison Reinsvold Hall, *Notre Dame*
Paola Katherine Rozo Bernal, *CERN*
Jean-Roch Vlimant, *CalTech*

July 5, 2017

Introduction

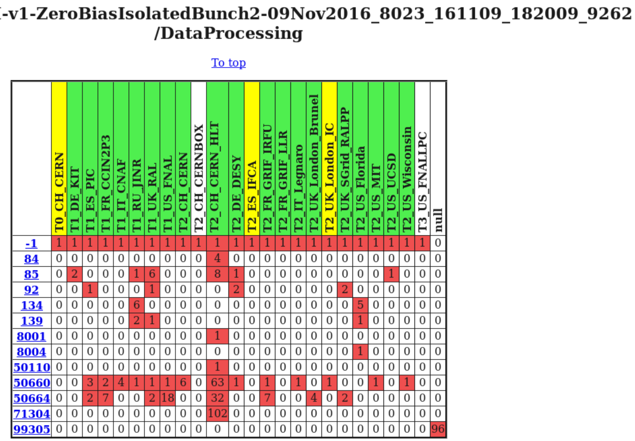
- During central production of datasets, we sometimes get errors such as missing files or high memory usage.
- An operator must look at the error codes and decide what action to take on the workflow.
- Some errors are easy to understand, and there are automated responses.
- An algorithm that encompassed all possible patterns that we can anticipate would be difficult or impossible to maintain.
- Machine learning is a natural solution.

Defining a Model

- For each workflow, we know the number of times each possible error code is thrown at each site.
- This leads to a simple matrix of number of error codes times the number of sites, with each element being the number of matching errors.
- Another feature considered is the site status at the time of the error (enabled, disabled, or in drain).
- We just split these into “good” and “bad” site status, so there is a factor of two more features.

Example of Features

The color of each site corresponds to its status.
(White sites are not tracked by dashboard being used, so they're counted as "good.")



/fabozzi_Run2016H-v1-ZeroBiasIsolatedBunch2-09Nov2016_8023_161109_182009_9262 /DataProcessing

To top

	T0_CH_CERN	T1_DE_KIT	T1_ES_PIC	T1_FR_CCIN2P3	T1_IT_CNAF	T1_RU_JINR	T1_UK_RAL	T1_US_FNAL	T2_CH_CERN	T2_CH_CERNBOX	T2_CH_CERN_HLT	T2_DE_DESY	T2_ES_IFCA	T2_FR_GRIF_IRFU	T2_FR_GRIF_LLR	T2_IT_Legnaro	T2_UK_London_Brunel	T2_UK_London_IC	T2_UK_SGrid_RALPP	T2_US_Florida	T2_US_MIT	T2_US_UCSD	T2_US_Wisconsin	T3_US_FNALPC	null
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
84	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
85	0	2	0	0	0	1	6	0	0	0	8	1	0	0	0	0	0	0	0	0	0	1	0	0	0
92	0	0	1	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	2	0	0	0	0	0	0
134	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
139	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8001	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
50110	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50660	0	0	3	2	4	1	1	1	6	0	63	1	0	1	0	1	0	1	0	0	1	0	1	0	0
50664	0	0	2	7	0	0	2	18	0	0	32	0	0	7	0	0	4	0	2	0	0	0	0	0	0
71304	0	0	0	0	0	0	0	0	0	0	102	0	0	0	0	0	0	0	0	0	0	0	0	0	0
99305	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96

Note the largest 4 error codes: 71304, 50660, and 50664 at T2_CH_CERN_HLT and 99305. We will come back to those.

Unsupervised Learning Before the Operator

- First way we help the operator is to group similar error-type workflows together.
- Multiple workflows can then be acted on at the same time.
- To make the cluster characteristics stable, we include workflow error patterns and site statuses stored over the past few months.
- We are using K-Means clustering. (`http://scikit-learn.org/stable/modules/clustering.html#k-means`)

Quick Definition of K-Means Clustering

- Each point is inside the cluster corresponding to the nearest centroid.
- The centroid location is placed where the variance (the sum of the squared distances from the centroid) of the entire cluster is minimized.

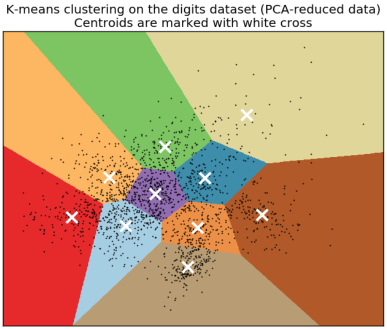
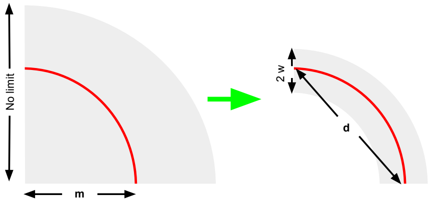


Image taken from scikit-learn.org.

Vector Compression

- Instead of errors times sites, we are using errors plus sites number of features.
- This helps group together similar errors, even if spread across sites.
- Because of potential large geometric distances due to large number of errors, we also compress error and site phase space.

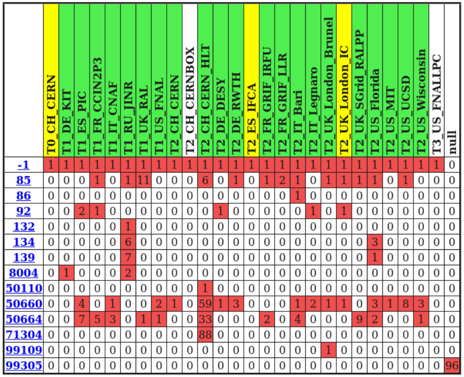
$$\text{distance from origin} = \frac{d}{\sqrt{2}} + 2w\left(\frac{|\vec{v}|}{|\vec{v}| + m} - 0.5\right) \tag{1}$$



An Obviously Similar Workflow

/fabozzi_Run2016H-v1-ZeroBiasIsolatedBunch5-09Nov2016_8023_161109_181336_2225 /DataProcessing

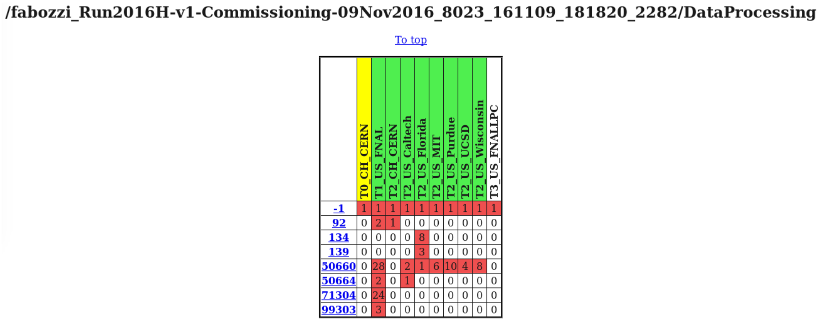
To top



	T0_CH_CERN	T1_DE_KIT	T1_ES_PIC	T1_FR_CCIN2P3	T1_IT_CNAF	T1_RU_JINR	T1_UK_RAL	T1_US_FNAL	T2_CH_CERN	T2_CH_CERNBOX	T2_CH_CERN_HLT	T2_DE_DESY	T2_DE_RWTH	T2_ES_IFCA	T2_FR_GRIF_IRFU	T2_FR_GRIF_LLR	T2_IT_Bari	T2_IT_Legnaro	T2_UK_London_Brunel	T2_UK_London_IC	T2_UK_SGrid_RALPP	T2_US_Florida	T2_US_MIT	T2_US_UCSD	T2_US_Wisconsin	T3_US_FNALLPC	null
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
85	0	0	0	1	0	1	11	0	0	0	6	0	1	0	1	2	1	0	1	1	1	1	0	1	0	0	0
86	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
92	0	0	2	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
132	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
134	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
139	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8004	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50110	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50660	0	0	4	0	1	0	0	2	1	0	59	1	3	0	0	0	1	2	1	1	0	3	1	8	3	0	0
50664	0	0	7	5	3	0	1	1	0	0	33	0	0	0	2	0	4	0	0	0	9	2	0	0	1	0	0
71304	0	0	0	0	0	0	0	0	0	0	88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
99109	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
99305	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96

- This workflow has the same four top error codes as before: 71304, 50660, and 50664 at T2_CH_CERN_HLT and 99305.
- From the task name itself, we can also see that the workflows are similar.

A Less Obvious Similar Workflow



/fabozzi_Run2016H-v1-Commissioning-09Nov2016_8023_161109_181820_2282/DataProcessing

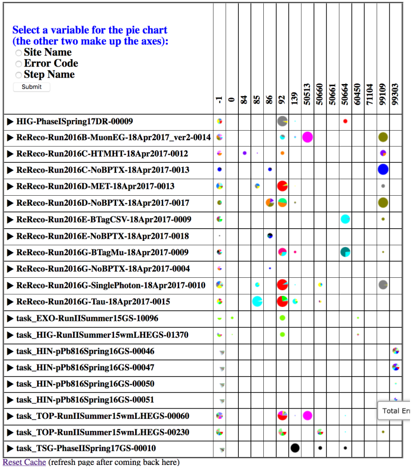
To top

	T0_CH_CERN	T1_US_FNAL	T2_CH_CERN	T2_US_Caltech	T2_US_Florida	T2_US_MIT	T2_US_Purdue	T2_US_UCSD	T2_US_Wisconsin	T3_US_FNALLPC
-1	1	1	1	1	1	1	1	1	1	1
92	0	2	1	0	0	0	0	0	0	0
134	0	0	0	0	8	0	0	0	0	0
139	0	0	0	0	3	0	0	0	0	0
50660	0	28	0	2	1	6	10	4	8	0
50664	0	2	0	1	0	0	0	0	0	0
71304	0	24	0	0	0	0	0	0	0	0
99303	0	3	0	0	0	0	0	0	0	0

- This workflow might not look the exact same at a first glance, but the 71304 and 50660 errors at a different site are likely from similar causes.
- Our algorithm clusters this with the previously shown workflows for the operator to review simultaneously.

Operator View

- A prototype here shows errors codes for each workflow.
- Each color of the pie chart is a different site.
- The size of the pie charts corresponds to the number of errors.



Select a variable for the pie chart
(the other two make up the axes):

- Site Name
- Error Code
- Step Name

Submit

	-1	0	84	85	86	92	139	50513	50660	50661	50664	60450	71104	99109	99303
▶ HIG-PhaseISpring17DR-00009	●					●	●				●				
▶ ReReco-Run2016B-MuonEG-18Apr2017_ver2-0014	●					●	●	●						●	
▶ ReReco-Run2016C-HTMHT-18Apr2017-0012	●		●	●		●								●	
▶ ReReco-Run2016C-NoBPTX-18Apr2017-0013	●				●									●	
▶ ReReco-Run2016D-MET-18Apr2017-0013	●			●		●	●							●	
▶ ReReco-Run2016D-NoBPTX-18Apr2017-0017	●				●	●	●							●	
▶ ReReco-Run2016E-BTagCSV-18Apr2017-0009	●										●			●	
▶ ReReco-Run2016E-NoBPTX-18Apr2017-0018	●				●										
▶ ReReco-Run2016G-BTagMu-18Apr2017-0009	●					●	●				●			●	
▶ ReReco-Run2016G-NoBPTX-18Apr2017-0004	●				●										
▶ ReReco-Run2016G-SinglePhoton-18Apr2017-0010	●			●		●	●		●					●	
▶ ReReco-Run2016G-Tau-18Apr2017-0015	●			●		●	●		●						
▶ task_EXO-RunIISummer15GS-10096	●	●				●						●			
▶ task_HIG-RunIISummer15wmLHEGS-01370	●	●				●						●			
▶ task_HIN-pPb816Spring16GS-00046	●														●
▶ task_HIN-pPb816Spring16GS-00047	●														●
▶ task_HIN-pPb816Spring16GS-00050	●														●
▶ task_HIN-pPb816Spring16GS-00051	●														●
▶ task_TOP-RunIISummer15wmLHEGS-00060	●					●	●	●			●				
▶ task_TOP-RunIISummer15wmLHEGS-00230	●					●			●		●			●	
▶ task_TSG-PhaseIISpring17GS-00010	●						●		●		●				

Total Err

Reset Cache (refresh page after coming back here)

Interlude

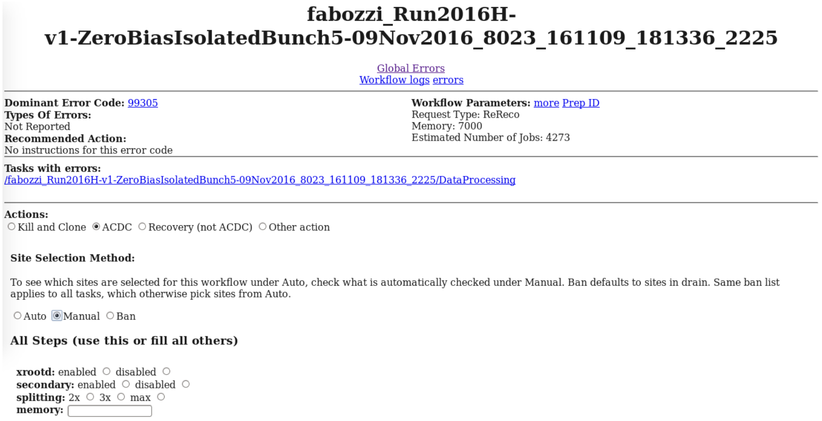
Short version:
We are pulling in information from Request Manager (workflow parameters) and WMStats servers (errors thrown) to inform the operator who has to make a decision.

All of this information can be used for training, but for now, we are using WMStats errors exclusively for this purpose.

Gathering Operator Actions

- We are trying to limit and enumerate possible operator actions.
- Turn command line script into series of options.
- Example actions would be to kill and clone jobs, or to recover partially complete jobs (ACDC).
- Farther parameters to train on are:
  job splitting, enabling XRootD, and requested memory

Gathering Operator Actions



fabozzi_Run2016H-v1-ZeroBiasIsolatedBunch5-09Nov2016_8023_161109_181336_2225

Global Errors
Workflow logs errors

Dominant Error Code: 99305
Types Of Errors:
Not Reported
Recommended Action:
No instructions for this error code

Workflow Parameters: more Prep ID
Request Type: ReReco
Memory: 7000
Estimated Number of Jobs: 4273

Tasks with errors:
/fabozzi_Run2016H-v1-ZeroBiasIsolatedBunch5-09Nov2016_8023_161109_181336_2225/DataProcessing

Actions:
○ Kill and Clone [✓ ACDC] ○ Recovery (not ACDC) ○ Other action

Site Selection Method:

To see which sites are selected for this workflow under Auto, check what is automatically checked under Manual. Ban defaults to sites in drain. Same ban list applies to all tasks, which otherwise pick sites from Auto.

○ Auto [✓ Manual] ○ Ban

All Steps (use this or fill all others)

xrootd: enabled ○ disabled ○
secondary: enabled ○ disabled ○
splitting: 2x ○ 3x ○ max ○
memory:

First Passes with Actions

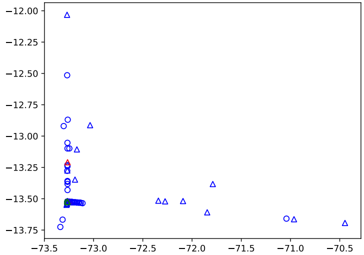
- Very much in progress as we collect operator actions.
- Using full feature vectors without compression, we dump into a neural net.
- Number of features is # Error codes $\times$ # Sites $\times 2 \approx 10000$.
- Parameters for classifier are just the defaults from `sklearn.neural_network.MLPClassifier`: `http://scikit-learn.org/stable/modules/neural_networks_supervised.html#classification`

In the following example, we use the 10000 features to decide between 2 classes.

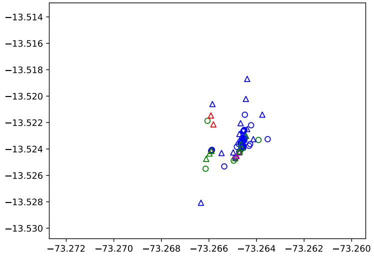
First Passes with Actions

- Predicting action (ACDC vs. clone) is 97% (109/112) for training sample and 82% (84/102) for test sample.
- Below is a plot of the first two principle components.
- PCA does not seem to cause much separation on its own.

First two Principle Components



Zoomed a bit



- Circles are training data, triangles are test data
- Blue correctly identified recovery, green is clone
- Magenta recovery not identifed, and red is clone

Conclusions

- Prototype of interface for operator was built with discrete options
- Capable of clustering workflows with similar errors
- We are recording the actions taken in a manner that can be trained
- [Work in progress]: We have started experimenting with predicting actions