Machine learning for lock loss analysis

Jameson Graef Rollins

LIGO Laboratory
California Institute of Technology

Gravitational Wave Advanced Detector Workshop
Hamilton Island, Australia
May 11, 2017

Problem of lock loss

The interferometer loses lock and we usually don't know why.

Leads to lost observation time, no BNS observations.

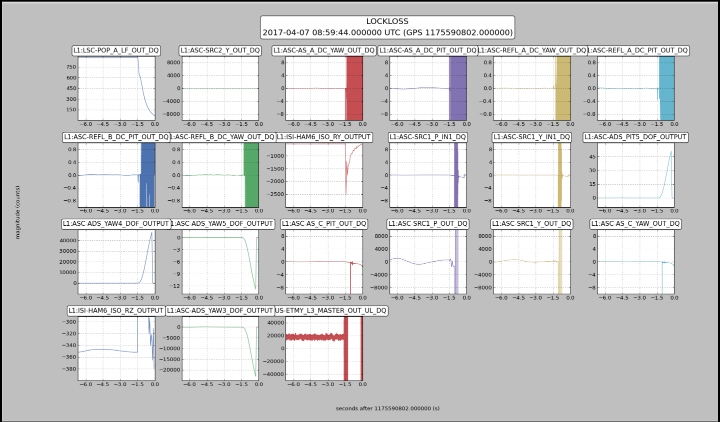
Earthquakes are certainly a known primary culprit, but they only account for a fraction of all loses. Usually we have very little idea what caused the lock loss.

- Cursory look for "lockloss" in logs: only $\sim 15\%$ mention "earthquake".

No systematic studies have been undertaken to understand why lock loses occur.

"Lockloss tool" v1

Plot time series plots of suspected relevant channels for manual inspection. Intuition-based analysis (mostly ineffective).

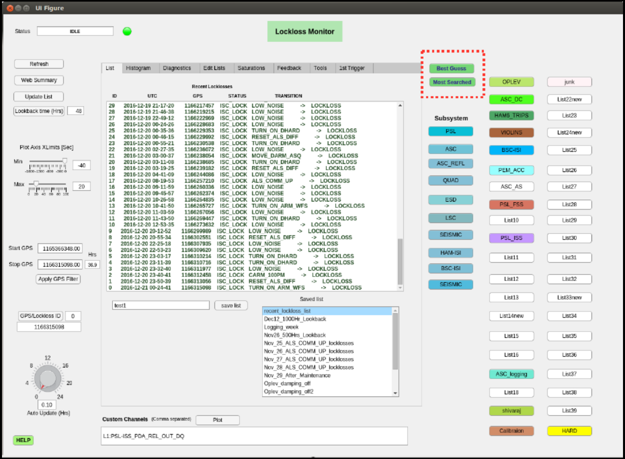


"Innovation" of this tool:

- integrated with Guardian to find lock loss event times
- modern plotting tool that allows zooming or panning in time

“Lockloss tool” v2

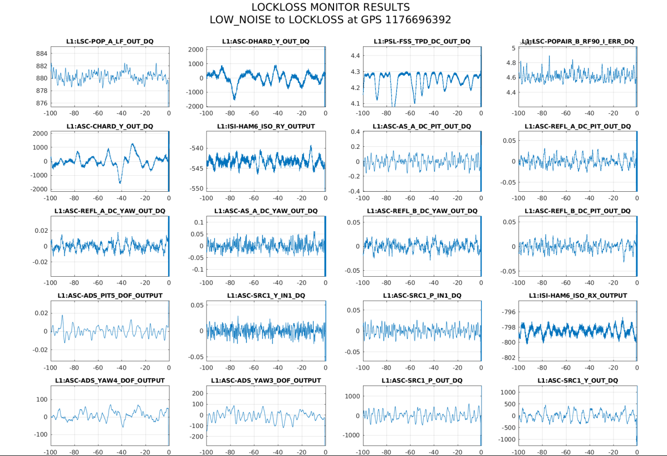
Newer tool, from LSC Fellow Nikhil Mukund:



- more sophisticated GUI
- “best guess” cause analysis looks for “early anomaly” channels
- auto-updating

"Lockloss tool" v2

Newer tool, from LSC Fellow Nikhil Mukund:



- more sophisticated GUI
- "best guess" cause analysis looks for "early anomaly" channels
- auto-updating

More systematic analysis needed

We record ~ 2.5k channels of "fast" data from the LIGO detectors:

- sensor inputs and actuator outputs
- interferometer length and angular control and error signals
- suspension/seismic/aux control/error signals
- physical and environment monitors (seismometers, microphones, magnetometers, pressures/temperature sensors, etc.).

Additionally record ~ 100k of "slow" monitors (intermediate signals and status bits).

Should be able to extract useful information from all this data...

Machine learning to the rescue?

Problem has hallmarks of machine learning problem:

- Lots of data, but unclear relationship between input and output.
- Some labels available ("lockloss" or "not lockloss") but no *a priori* knowledge of causes.
- Lots of variance in data obscures analysis; statistical approach required.

Various types of ML techniques can potentially be leveraged:

classification Determine model that can predict lock loss from available data streams (supervised learning).

clustering Group lock loss events by common features (unsupervised).

dimensionality reduction Reduce full data space to just relevant channels characteristic of learned classes.

Machine learning $\implies$ regression

Machine learning is really just regression, e.g. prediction using statistics: What is the function that produces the observed (desired) output from the given input?

$y = f(X, w)$ What is $f()$? What is w?

Trick comes in how you formulate the problem, and how complicated you expect/allow your function (model) to be.

- Are you predicting category or quantity?
- Do you have a model of the functional relationship?
- Do you have a model of the error on the observations?
- Does the data have labels or not?

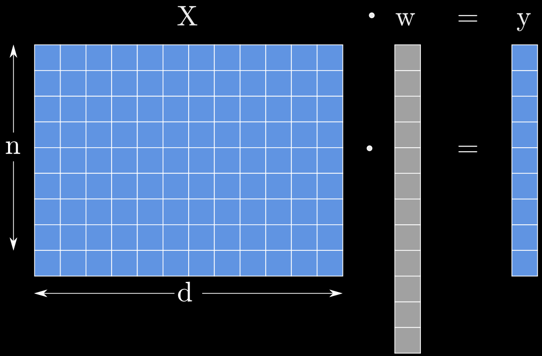
First step: choosing a model and formulating the question

For this lock loss problem we'll start by assuming a very simple **linear model** of the relationship between the data in recorded channels and whether or not we lose lock.

What are the channels, and features in the data from those channels, that predict lock loss?

Review: linear models

Assume outputs, y, are a simple linear combination of inputs, X:



input matrix X: n samples each with d features

Goal is to determine the vector of coefficients, w.

Review: linear least squares regression

A *linear regression* attempts to find the coefficients that minimize the residual sum of squares between the observed output and the response predicted by the linear approximation:

$$||Xw - y||_2{}^2 \to \min_w$$

where $||x||_2$ is called the *2-norm*, which is a specific example of *p-norm*:

$$||x||_p \equiv \left(\sum_i |x_i|^p \right)^{1/p}$$

$$||x||_p{}^p = \sum_i |x_i|^p$$

Review: linear regression with regularization

In order to constrain the coefficients, because e.g. the problem is ill-posed or underdetermined, we can add *regularization*:

$$||Xw - y||_2{}^2 + \alpha ||w||_p{}^p \to \min_w$$

where $\alpha \geq 0$ is the *regularization coefficient*.

The norm of the regularization determines how coefficients are constrained:

$p = 2$ **ridge regression**: forces coefficients to be *small*

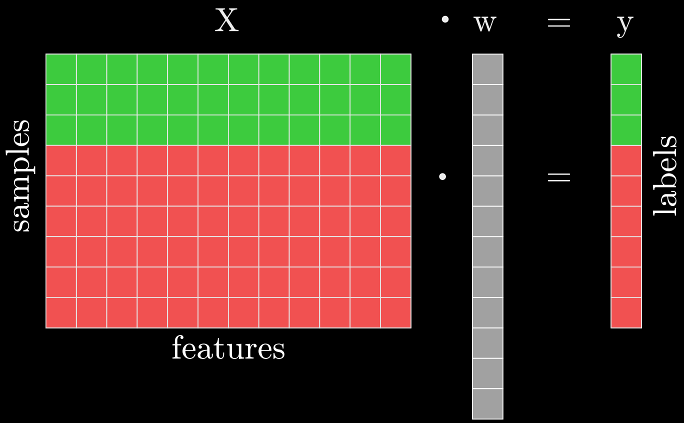
$p = 1$ **LASSO regression**: forces coefficients to be *sparse*

Overview of basic approach

Outline of approach for lock loss analysis:

- Find **labeled samples** of times indicative of lock loss (right before lock loss) and quiescent stable operation (during "low-noise" operation far from lock loss).
- Extract relevant **features** from all available data streams at each sample.
- Apply **regression** to determine which features are indicators of lock loss (binary classification).
- **Cluster** predictive features to find classes/types of lock loss events.
- Examine features indicative of each lock loss class to guide commissioners in how to attack problem.

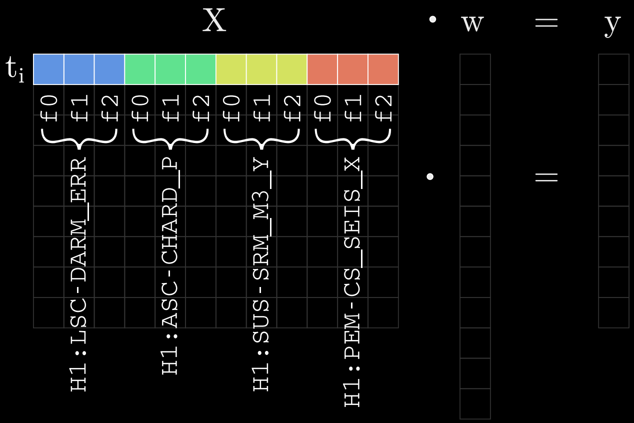
Binary classification



positive sample: **lock-loss** data

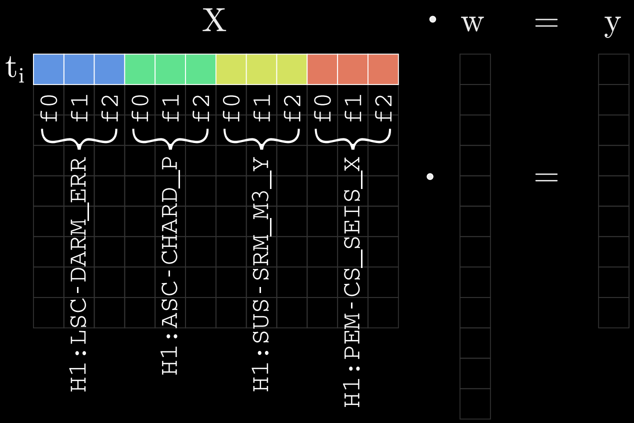
negative sample: **stable lock** data

Feature reduction



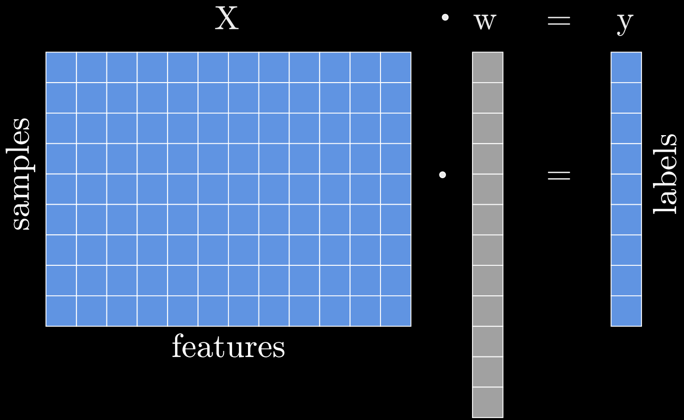
Each sample is a concatenation of vectors of features extracted from each channel at a specified time.

Feature reduction



Relevant features to extract is an open question (see below).

LASSO regression



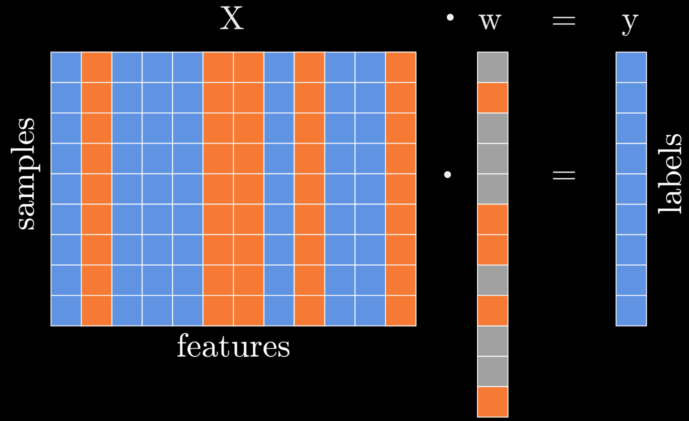
In our case the system is severely **under-determined**:

- d features ($\sim$50k) $\gg$ n samples ($\sim$7k)

Infinite number of solutions to simple LSF.

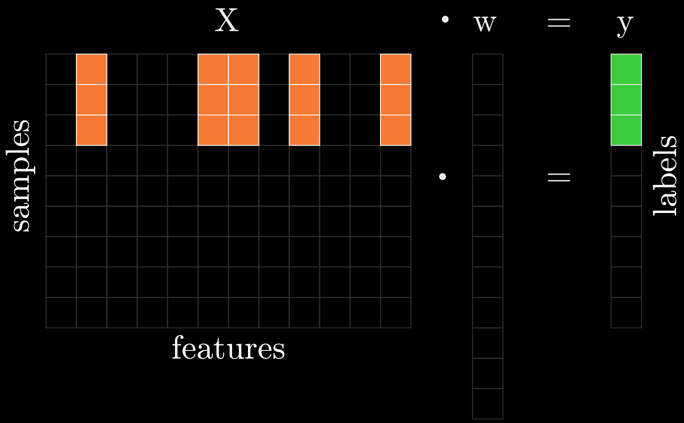
Regularization *required* to solve the problem.

LASSO regression



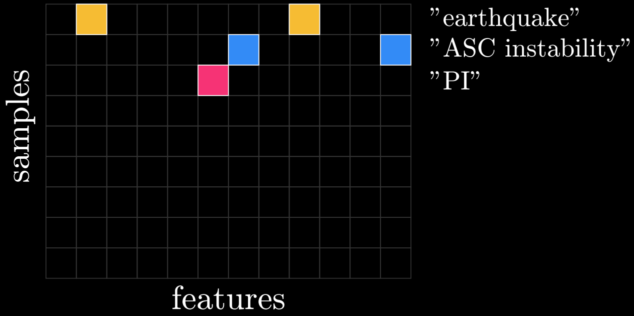
LASSO regression estimates sparse coefficients, effectively picking out particular features that are most predictive of lock loss.

Clustering



Create new input matrix with reduced set of relevant features from positive samples, and *cluster* in that space to look for classes of lock losses.

Classification



Classes are defined by groups of related features (channels/extracted-features).

These feature-defined classes should *hopefully* guide commissioners to underlying problems.

scikit-learn: *de facto* standard for machine learning in Python.

- massive library of pre-vetted algorithms for every conceivable machine learning task:
  - classification
  - regression
  - clustering
  - dimensionality reduction
  - model selection
- standard interface for all algorithms, many useful tools for pre-processing, data reduction, plotting, etc.
- incredible documentation

`http://scikit-learn.org/`

Current status of analysis

- Initially targeting LIGO "O1" data.
  - **200 lock loss events** from nominal low-noise state (positive samples)
  - 7000 "clean lock" samples > 1000 seconds before lock loss (negative samples)
- Reduced set of 190 "relevant" channels (eventually just look at **all** available channels).
- Channel ASDs as initial feature set.
  - ~ 55k features

Input matrix size: **14 GB**

- *MeanShift* clustering algorithm to guess number of relevant clusters. Many others to choose from...
- Building out tools to utilize the LIGO LDAS computer clusters.

Lessons learned so far

Parameter space is very large.

- What are the right features to extract from the samples?
  - ASD?
  - wavelets?
  - some other decomposition?
- Regression parameter, α, determines number of relevant features. What is the right number?
- Which regression and clustering algorithm is best?
- Is a linear model the right one?

Data reduction/preparation is very time consuming.

- 90% of the time is preparing the data

Building infrastructure for ML in LIGO

Actively pushing on tools and infrastructure to lower the bar for doing machine learning in LIGO.

In particular, how can we better leverage the LDAS cluster for instrument science tasks?

jupyterhub on the LDAS clusters jupyter (ipython) notebooks running on heavy-lifting cluster nodes, available through the web without needing to ssh into the cluster.

python cluster mapping convenient utilities for leveraging full cluster resources (thousands of worker nodes at our disposal).

Conclusions

Machine learning is not a turn-key solution. It takes a lot of work to implement.

- Tricky to setup problem in useful/simple way.
- Data reduction is non-trivial, very time-consuming.

Machine learning is not a panacea or a silver bullet. , but once we learn what it can do and how to utilize it it should be a powerful tool to help us answer tricky questions.