Machine Learning
IBM Spark Services / Allen Telescope Array

1. Data Mining the ATA 10-Year Archives

2. Using Spark to enable new types of observations

3. Signal Classification - including real-time triage

Machine Learning
IBM Spark Services / Allen Telescope Array

1. Data Mining the ATA 10-Year Archives

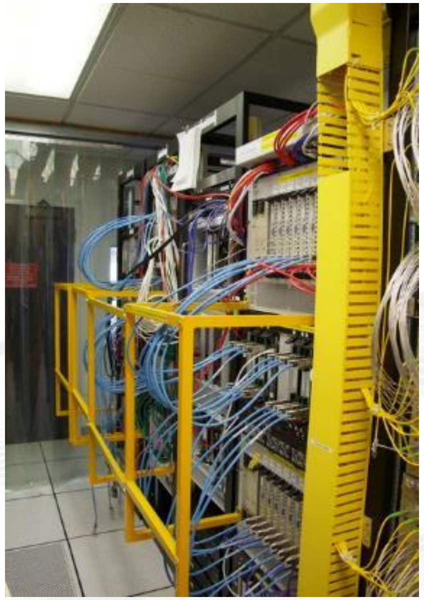
2. Using Spark to enable new types of observations

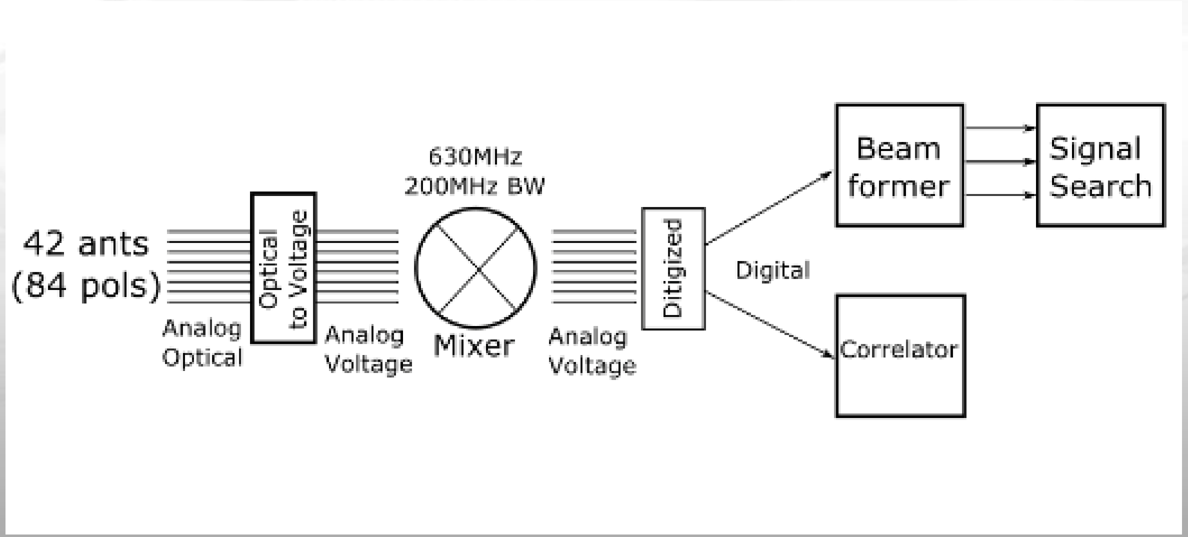
3. Signal Classification - including real-time triage

Overview of SystemML

Allen Telescope Array

- Allen Telescope Array (ATA) – Phased Array Synthetic Dish – 3 Beams
- 42 receiving dishes, each 6 meters diameter
- 1GHz to 10GHz receiving capability, 100MHz bandwidth
- 4.5TB data coming from the beamformers every hour
- Only the data with detected signals is saved for later analysis





IBM Spark@SETI – Greenbank Observatory Data

Greenbank SDFITS Analysis

In [104]:
```
from astropy.io import fits as pyfits
from astropy import units as u
from astropy.coordinates import SkyCoord
import numpy as np
import matplotlib.pyplot as plt
import pyspeckit
%matplotlib inline
```

GBT SDFIT Summary

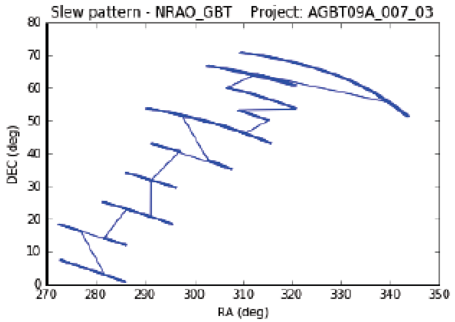
In [50]:
```
fits_file = "/Users/graham/Documents/Work/GBT/AGBT09A_007_03.raw.acs.fits"
hdulist = pyfits.open(fits_file)
fitsdata = hdulist[1].data
hdulist.info()
print('\nObserver: '+ fitsdata['OBSERVER'][0])
print('Project: ' + hdulist[1].header['PROJID'])
print('Telescope: ' + hdulist[0].header['ORIGIN'])
print('Observation type: '+ hdulist[0].header['INSTRUME'])
print('Object: '+ fitsdata['OBJECT'][0])
print('Date: '+ fitsdata['DATE-OBS'][0])
```

```
Filename: /Users/graham/Documents/Work/GBT/AGBT09A_007_03.raw.acs.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU      12   ()
1    SINGLE DISH  BinTableHDU    229   53664R x 70C   ['32A', '1D', '22A', '1D',
 '1D', '1D', '16384E', '16A', '6A', '8A', '1D', '1D', '1D', '4A', '1D', '4A', '1
D', '1I', '32A', '32A', '1J', '32A', '16A', '1E', '8A', '1D', '1D', '1D', '1D',
'1D', '1D', '1D', '1D', '1D', '1D', '1D', '1D', '8A', '1D', '1D', '12A', '1I', '
1I', '1D', '1D', '1I', '1A', '1I', '1I', '16A', '16A', '1J', '1J', '22A', '1D',
'1D', '1I', '1A', '1D', '1E', '1D', '1A', '1A', '8A', '1E', '1E', '16A', '1I', '
1I', '1I']

Observer: Jay Lockman
Project: AGBT09A_007_03
Telescope: NRAO Green Bank
Observation type: Spectrometer
Object: G35.0+5.0
Date: 2009-01-17T19:52:49.00
```

Greenbank Dish Slew Patten

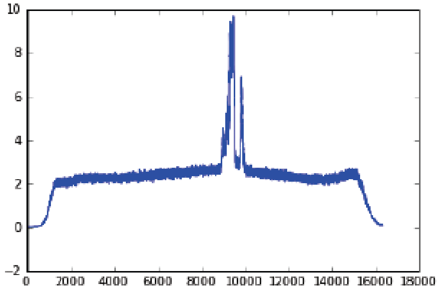
Spectra tuned to the 21-cm transition of neutral hydrogen to generate neutral hydrogen maps

In [106]:
```
# Display RA/DEC plot of dish slew patterns during observations
# Convert GBT galactic coordinates to RA/DEC
c = SkyCoord(fitsdata['CRVAL2'], fitsdata['CRVAL3'], frame='galactic', unit='deg'
)
c_radec = c.transform_to('icrs')
plt.plot(c_radec.ra.deg, c_radec.dec.deg)
plt.title("Slew pattern - " + hdulist[0].header['TELESCOP'] + "    Project: " + h
dulist[1].header['PROJID'])
plt.ylabel('DEC (deg)')
plt.xlabel('RA (deg)')
plt.show()
```



Inspection of Spectral Observation Data

In [119]:
```
flux = fitsdata['DATA']
plt.plot(flux[0])
plt.show()
```



1. Data Mining the ATA 10-Year Archives

- 200M signal event records
- 360K multi-band compAmp files (candidate signals) – original data or waterfall plot pngs
- Example archive data mining:
  - Looking for targets with unusually consistent corrected (heliocentric) Doppler Drift over multi-year spans

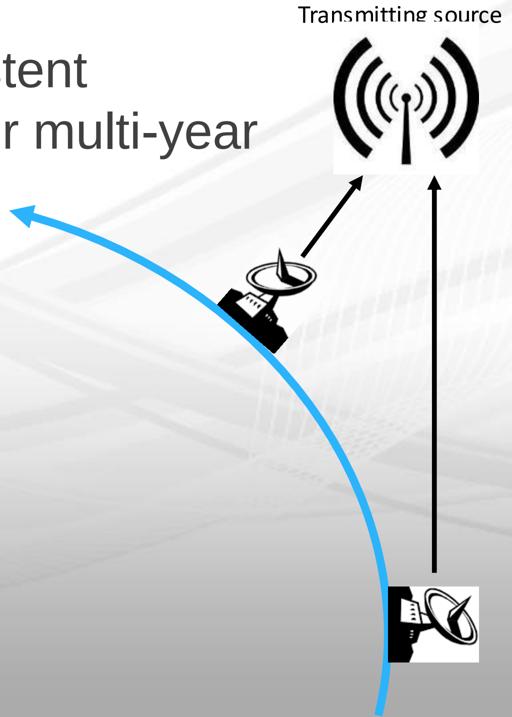
Joined by unique ID

UniqueId	Time	ActTyp	TgtId	catalog	RA2000Hr
antisolar_14_36_0_1	2010-06-11 21:02:54	target1off	109471	habcat	13.236

Dec2000Deg	Power	SNR	FreqMHz	DriftHz/s	WidHz
-14.282	166	NULL	1424.97454	-0.315	0.662

SigTyp	PPeriodS	NPul	IntTimeS	TscpAzDeg	Pol
CwP	NULL	NULL	98	180.33	both

TscpElDeg	BeamNo	SigClass	SigReason	CandReason	
34.862	3	Cand	PsPwrT	SnMulBm	



Heliocentric Drift – Computed for 200M records in 13m 24sec

JPL Algorithm

In [46]:
```python
# CALCULATE CORRECTED (HELIOCENTRIC) DRIFT AND ACCLERATION

latitude = 40.8178      # [deg] latitude of the ATA location
rotation_speed = 465    # [m/s] on the equator
rotation_accel = 0.034  # [m/s^2] on the equator
speed_of_light = 3e8    # [m/s]
D2R = np.pi/180
Earth_tilt = D2R * 23.43928
   # gravitational acceleratation due to the Sun = G_constant * Sun_mass / AU^2
Sun_g = 6.674e-11 * 1.989e30 / (1.496e11 * 1.496e11)
    # average Earth speed = 2*pi*AU/year, converted to [m/s]
Earth_speed = 29785.68

def shared_items_short_calc(time):
    shared_items = {}
    # Note: time here is in years, as opposed to centuries in the JPL document
    shared_items['axis'] = 1.00000261 + 0.0000000562 * time
    shared_items['eccentricity'] = 0.01671123 - 0.0000004392 * time
    shared_items['longitude'] = D2R * (100.46457166 + 359.9937244981 * time)
    shared_items['perihelion'] = D2R * (102.93768193 + 0.0032327364 * time)
    shared_items['mean_anomaly'] = np.fmod(shared_items.get('longitude') - shared_items.get('perihelic
    shared_items['ecc_anomaly'] = ecc_anomaly_calc(shared_items.get('mean_anomaly'), shared_items.get(
    shared_items['orbital_x'] = -(shared_items.get('axis')) * (np.cos (shared_items.get('ecc_anomaly')
    shared_items['orbital_y'] = -(shared_items.get('axis')) * np.sqrt (1-shared_items.get('eccentricit
    shared_items['heliocentric_x'] = shared_items.get('orbital_x') * np.cos (shared_items.get('perihel
    shared_items['equatorial_x'] = shared_items.get('heliocentric_x')
    ecc_adjustment = 1 / (1 - shared_items.get('eccentricity') * np.cos (shared_items.get('ecc_anomaly
    shared_items['orbital_vx'] = -(shared_items.get('axis')) * np.sin (shared_items.get('ecc_anomaly')
    shared_items['orbital_vy'] = shared_items.get('axis') * np.sqrt (1-shared_items.get('eccentricity'
    shared_items['heliocentric_vx'] = shared_items.get('orbital_vx') * np.cos (shared_items.get('perih
    shared_items['equatorial_vx'] = shared_items.get('heliocentric_vx')

    return shared_items

def shared_items_full_calc(time):
    # Note: time here is in years, as opposed to centuries in the JPL document
    shared_items = shared_items_short_calc(time)
    shared_items['inclination'] = -0.00001531 - 0.0001294668 * time
    shared_items['heliocentric_y'] = (shared_items.get('orbital_x') * np.sin (shared_items.get('perihe
    shared_items['heliocentric_z'] = (shared_items.get('orbital_x') * np.sin (shared_items.get('perihe
    shared_items['equatorial_y'] = shared_items.get('heliocentric_y') * np.cos (Earth_tilt) - shared_i
    shared_items['equatorial_z'] = shared_items.get('heliocentric_y') * np.sin (Earth_tilt) + shared_i
    shared_items['heliocentric_vx'] = shared_items.get('orbital_vx') * np.cos (shared_items.get('perih
    shared_items['heliocentric_vy'] = (shared_items.get('orbital_vx') * np.sin (shared_items.get('peri
    shared_items['heliocentric_vz'] = (shared_items.get('orbital_vx') * np.sin (shared_items.get('peri
    shared_items['equatorial_vy'] = shared_items.get('heliocentric_vy') * np.cos (Earth_tilt) - shared
    shared_items['equatorial_vz'] = shared_items.get('heliocentric_vy') * np.sin (Earth_tilt) + shared

    return shared_items

def ecc_anomaly_calc(mean_anomaly, eccentricity):
     # we solve Kepler's equation by 3 iterations of Newton's method
    ecc_anomaly = mean_anomaly + eccentricity * np.sin (mean_anomaly)
    ecc_anomaly += (mean_anomaly - ecc_anomaly - eccentricity * np.sin (ecc_anomaly)) / (1 - eccentricity
    ecc_anomaly += (mean_anomaly - ecc_anomaly - eccentricity * np.sin (ecc_anomaly)) / (1 - eccentricity * np.cos (ecc_anomaly))
    ecc_anomaly += (mean_anomaly - ecc_anomaly - eccentricity * np.sin (ecc_anomaly)) / (1 - eccentricity * np.cos (ecc_anomaly))
    return ecc_anomaly

def rel_shift_calc(azimuth, elevation):
    velocity_factor = np.sin(D2R*azimuth)*np.cos(D2R*elevation)
    rel_shift = rotation_speed/speed_of_light*velocity_factor*np.cos(D2R*latitude)
    return rel_shift
```

```python
azimuth = exo_rdd.map(lambda p: Row(azimuth=p.azimuth))
elevation = exo_rdd.map(lambda p: Row(elevation=p.elevation))
J2000_time = exo_rdd.map(lambda p:Row(J2000_time=p.J2000_time))
freq = exo_rdd.map(lambda p: Row(freq=p.freq))
drift = exo_rdd.map(lambda p: Row(drift=p.drift))
RA = exo_rdd.map(lambda p: Row(RA=p.RA))
Dec = exo_rdd.map(lambda p: Row(Dec=p.Dec))

#Compute the Doppler effect arising from Earth's rotation

# Doppler shift relative to frequency: to be multiplied by frequency in Hz
relative_diurnal_shift = exo_rdd.map(lambda line: Row(value=relative_diurnal_shift_calc(line)))
# resulting shift in Hz
diurnal_shift = exo_rdd.map(lambda line: Row(value=diurnal_shift_calc(line)))
# Doppler drift relative to frequency: to be multiplied by frequency in Hz
relative_diurnal_drift = exo_rdd.map(lambda line: Row(value=relative_diurnal_drift_calc(line)))
# resulting drift in Hz/s
diurnal_drift = exo_rdd.map(lambda line: Row(value=diurnal_drift_calc(line)))

# these two are computed just as a sanity check
#(for a hypothetical signal coming from the vernal equinox)
vernal_orbital_drift = exo_rdd.map(lambda line: Row(value=vernal_orbital_drift_calc(line)))
vernal_orbital_shift = exo_rdd.map(lambda line: Row(value=vernal_orbital_shift_calc(line)))

doppler_rdd = exo_rdd.map(lambda line: Row(UniqueId=line.UniqueId, TgtId=line.TgtId, timeInSeconds=line.timeInSeconds,
    Time=line.Time, power=line.power, freq=int(line.freq), drift=line.drift, RA=line.RA, Dec=line.Dec,
    corrected_drift=round(corrected_drift_calc(line),4),  corrected_acceleration=round(corrected_acceleration(line),4),
    CandReason=line.CandReason))
```

Heliocentric Drift Data Mining Pipeline with IBM Spark@SETI

SignalDB → Filter to Candidate Signals → Calculate Corrected Drift → Sequence by time grouped by target, corrected drift → Rank by longest unbroken sequence with same drift → Get observation line items for selected targets → Report

UniqueId	TgtId	Time	RA	Dec	freq	power	drift	corrected_drift
keplerLBand_9335_1019_31_9472308	150002	4/11/2012 4:17	289.214996	47.883999	1620.57019	172	-0.068	-0.00228
kepler8ghz_23973_1004_3_12719091	150002	7/21/2014 2:42	289.214996	47.883999	3711.37915	3579.89209	-0.09	0.001362
kepler8ghz_23973_1018_5_12720747	150002	7/21/2014 2:42	289.214996	47.883999	3722.89			
kepler8ghz_23973_1013_12_12721289	150002	7/21/2014 2:42	289.214996	47.883999	3719.008			
kepler8ghz_23982_1006_0_12807130	150002	7/21/2014 3:11	289.214996	47.883999	3769.439			
kepler8ghz_24646_1001_11_17268731	150002	7/26/2014 2:26	289.214996	47.883999	3609.280			
kepler8ghz_24646_1009_9_17268595	150002	7/26/2014 2:26	289.214996	47.883999	3615.475			
kepler8ghz_24646_1021_0_17269024	150002	7/26/2014 2:26	289.214996	47.883999	3625.203			
kepler8ghz_25255_1016_8_17662001	150002	7/29/2014 4:09	289.214996	47.883999	3559.148			
kepler8ghz_25256_1002_10_17662435	150002	7/29/2014 4:13	289.214996	47.883999	3566.776			
kepler8ghz_25256_1009_7_17662702	150002	7/29/2014 4:13	289.214996	47.883999	3572.255			
kepler8ghz_25789_1007_11_17861540	150002	8/2/2014 1:58	289.214996	47.883999	3489.662			
kepler8ghz_25789_1007_12_17861541	150002	8/2/2014 1:58	289.214996	47.883999	3489.714			
kepler8ghz_25791_1002_16_17862204	150002	8/2/2014 2:04	289.214996	47.883999	3485.771			
kepler8ghz_25791_1004_5_17862275	150002	8/2/2014 2:04	289.214996	47.883999	3486.898			
kepler8ghz_25791_1010_9_17862262	150002	8/2/2014 2:04	289.214996	47.883999	3492.018			
kepler8ghz_25791_1022_0_17862219	150002	8/2/2014 2:04	289.214996	47.883999	3501.387			
kepler8ghz_25791_1015_16_17862436	150002	8/2/2014 2:04	289.214996	47.883999	3496.370			
kepler8ghz_25791_1018_12_17862632	150002	8/2/2014 2:04	289.214996	47.883999	3498.623			
kepler8ghz_25791_1018_3_17862623	150002	8/2/2014 2:04	289.214996	47.883999	3498.213			

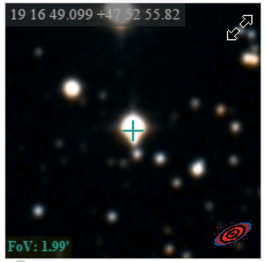
Basic data :

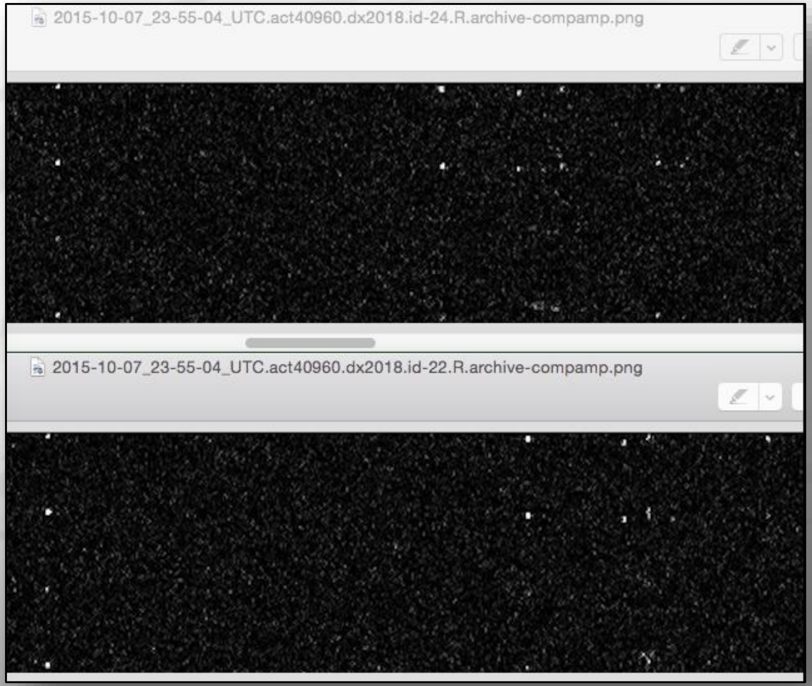
KOI-87.01 -- Extra-solar Confirmed Planet

Other object types: Pl (), Pl? (KOI)
ICRS coord. *(ep=J2000)* : 19 16 52.19 +47 53 04.0 ( ) [ ] D ~
FK5 coord. *(ep=J2000 eq=2000)* : 19 16 52.19 +47 53 04.0 [ ]
FK4 coord. *(ep=B1950 eq=1950)* : 19 15 28.15 +47 47 37.0 [ ]
Gal coord. *(ep=J2000)* : 079.0919 +15.7923 [ ]

SIMBAD query around with radius 2 arcmin

Interactive AladinLite view



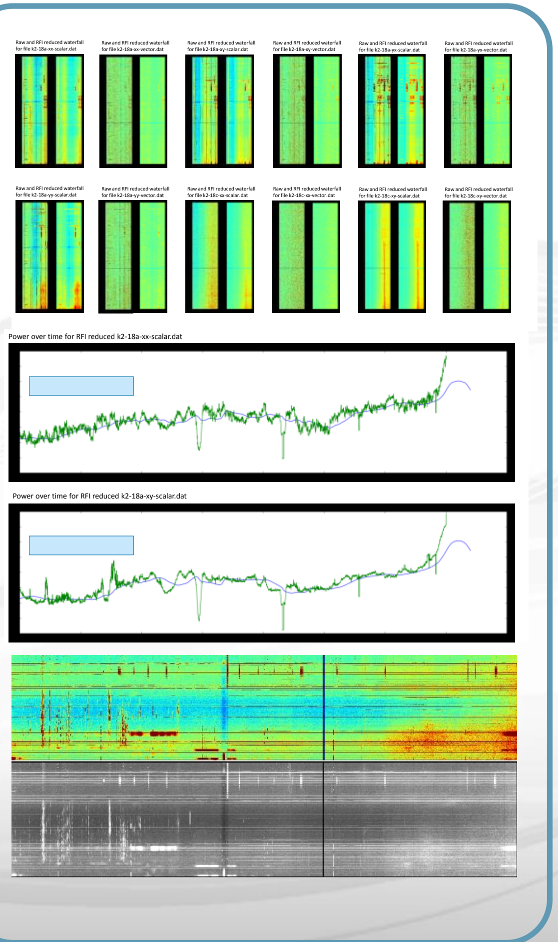


2. IBM Spark and New Observation Campaigns

- ~5TB per observation – direct streaming to IBM Tape Drive installed at ATA (IBM TS2270 high capacity 5-15TB per cartridge)
- Tapes received by IBM Cloud Storage in San Jose for ground-to-cloud upload into Object Store in IBM Cloud Storage, accessed by IBM Spark Services using SWIFT
- Examples:
  - Leakage detection using known exoplanet occultations – data folding of multiple occultation events to look for slight dips in overall power
  - Eavesdropping targets: neighboring but non-binary stars with close to zero angular separation – wide band analytics (SWAC)

Identifier	Otype	ICRS (J2000) RA	ICRS (J2000) DEC	distance	distance unit
2MASS J21103096-2710513	*	317.629	-27.18092	22	pc
2MASS J21103147-2710578	*	317.63117	-27.18272	16	pc

Eavesdropping analytics on IBM Spark@SETI
Example: Stellar pair separated by ~6pc
~5.2TB data collected for wide band analysis



Output of Jupyter Notebook - IBM Spark@SETI
K2-18 b Occultation Observation : 03/30/2016 23:28 UT

3. Signal Classification

- Supervised and unsupervised Machine Learning
- Initial focus is on finding suitable scalar features
- Collaborating with NASA under signed Space Act Agreement
- Stanford research teams using IBM Spark@SETI platform – researching advanced feature extraction for use with scikit-learn

3. Image Classification – A Simple Example

In [1]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
import pandas as pd
dir_name = '/ata_ibm_seti/signals/waterfalls/'
std_time_array = []
std_freq_array = []
file_list = os.listdir( dir_name )
```

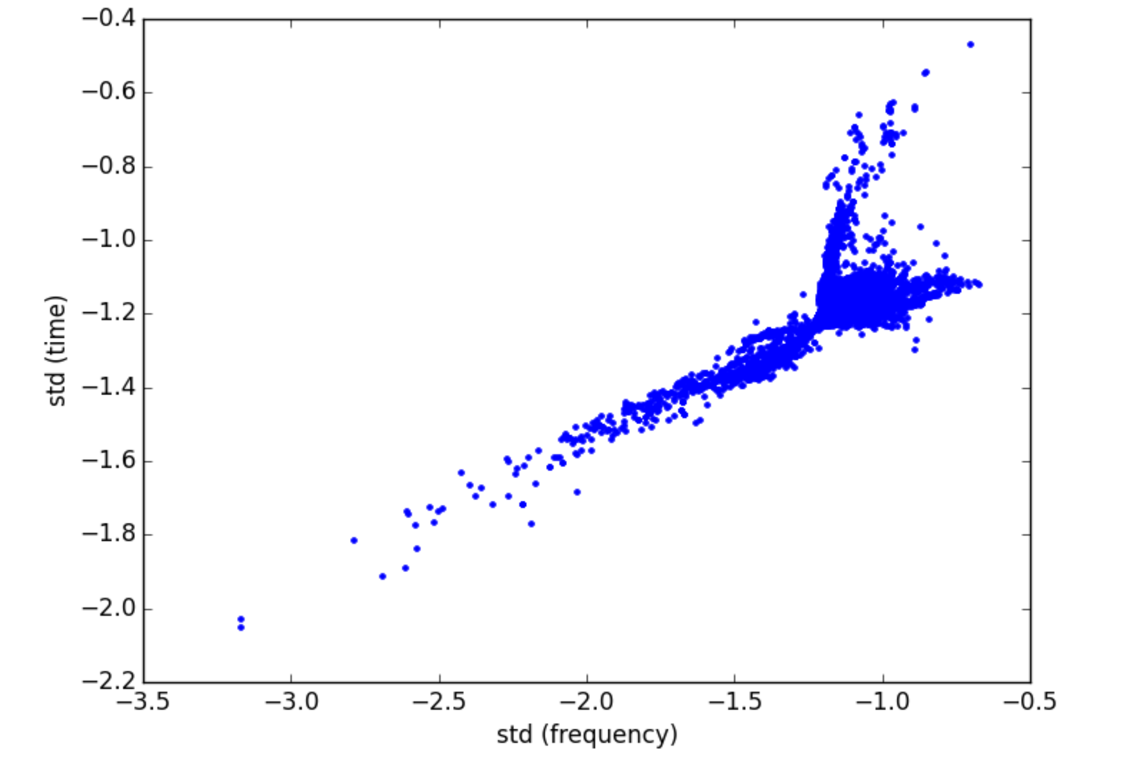
In [2]:
```
for file in file_list:

    if str.find(file, '.png') != -1:

          image_this = img.imread( dir_name + file )

        std_time = np.mean( np.std( image_this, axis = 0 ) )
        std_freq = np.mean( np.std( image_this, axis = 1 ) )

        std_time_array = np.append( std_time_array, std_time )
        std_freq_array = np.append( std_freq_array, std_freq )
```

(Note: Non-parallelized code)

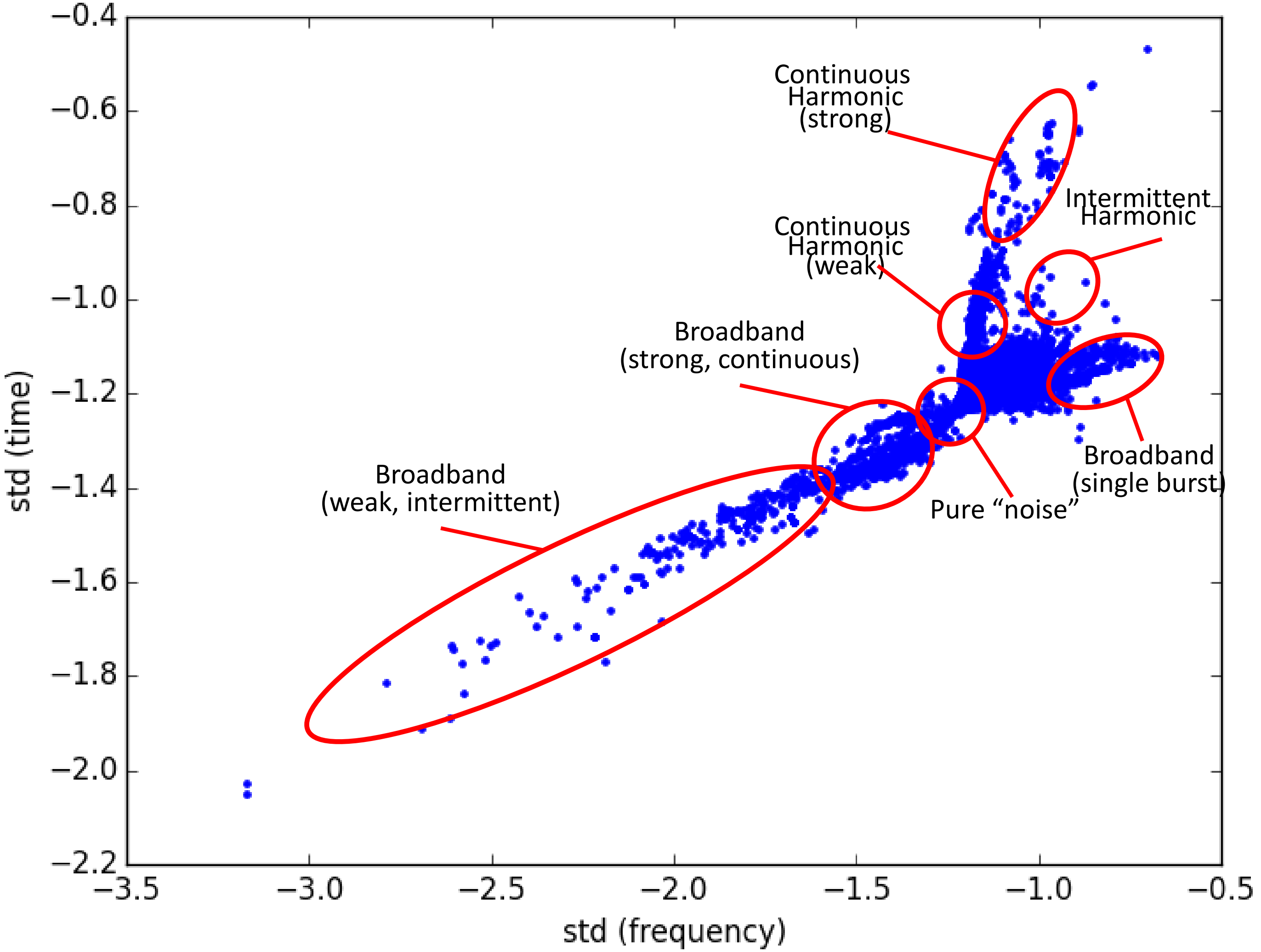
In [ ]:
```
plt.plot( np.log10(std_freq_array), np.log10(std_time_array), 'b.',label='Waterfall Parameters' )
plt.xlabel('std (frequency) ')
plt.ylabel('std (time) ')
plt.show()
```

Standard Deviation

Standard Deviation



3. Signal Classification – A Simple Example



3. Signal Classification – Scalar Features

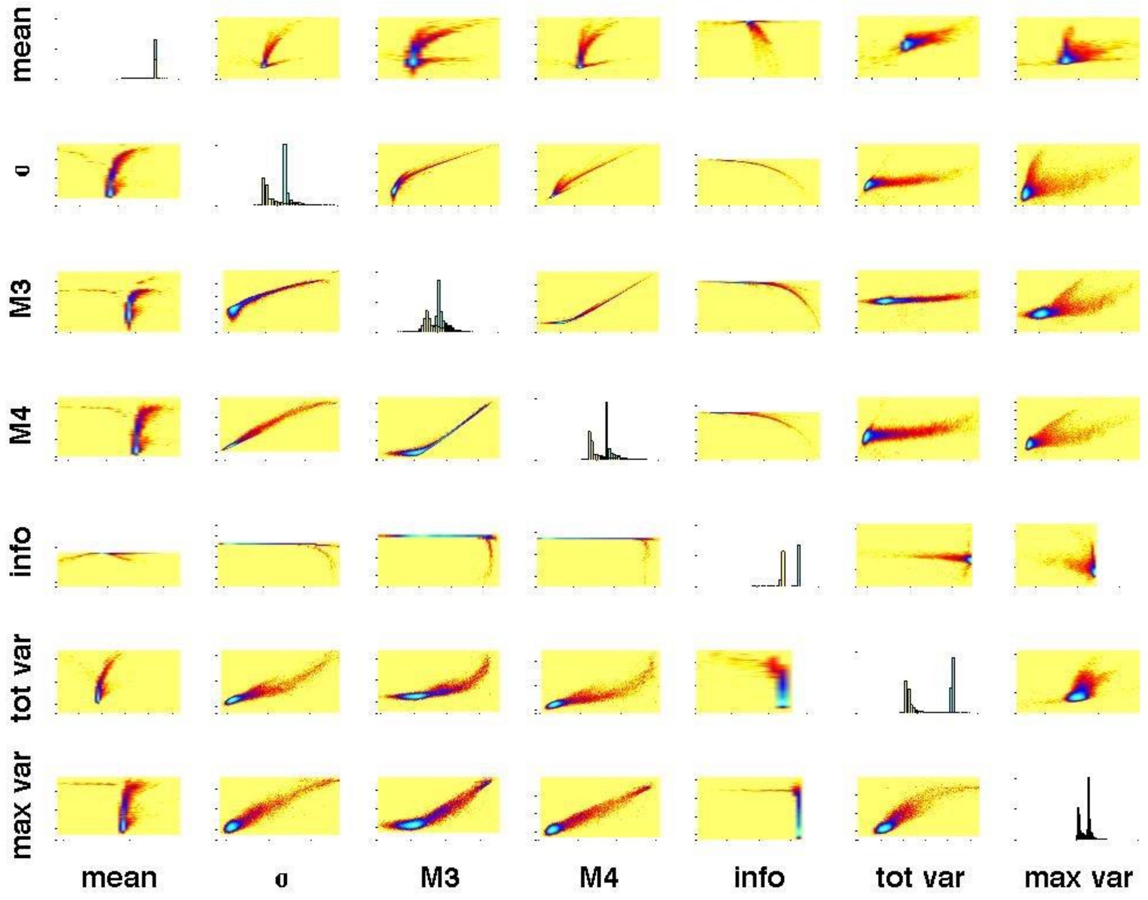
1D Signal Variants

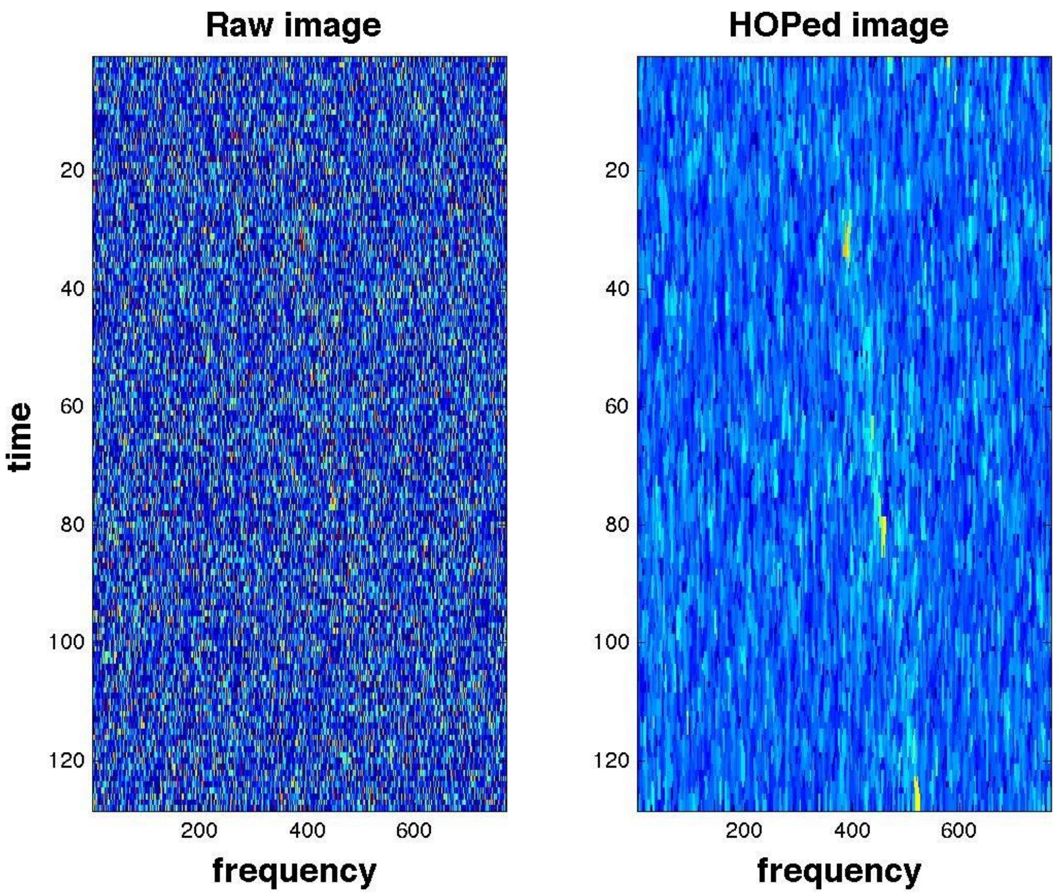
$X_{n,m}$ = waterfall plot amplitude
n = frequency index
m = time index

Variable	Frequency (Spectrum)	Time (Light Curve)
Projected	$\mathrm{Op}[\ \sum_m X_{n,m}\ ]$	$\mathrm{Op}[\ \sum_n X_{n,m}\ ]$
Slice-wise	$\sum_m \mathrm{Op}[\ X_{n,m}\ ]$	$\sum_n \mathrm{Op}[\ X_{n,m}\ ]$
Projected Difference	$\mathrm{Op}[\ \Delta_n \sum_m X_{n,m}\ ]$	$\mathrm{Op}[\ \Delta_m \sum_n X_{n,m}\ ]$
Slice-wise Difference	$\sum_m \mathrm{Op}[\ \Delta_n X_{n,m}\ ]$	$\sum_n \mathrm{Op}[\ \Delta_m X_{n,m}\ ]$

- Mean value $\bar{X} = \frac{1}{N}\Sigma_{n=1}^{N} X_n$
- Standard deviation $\sigma_X^2 = \frac{1}{N}\Sigma_{n=1}^{N}(X_n - \bar{X})^2$
- 3rd moment $= \frac{1}{N}\Sigma_{n=1}^{N}(X_n - \bar{X})^3$
- 4th moment $= \frac{1}{N}\Sigma_{n=1}^{N}(X_n - \bar{X})^4$
- Shannon Information $I = \int P(x,y)\ logP(x,y)dxdy$
- Total Variation $= \sum_{n=1}^{N-1} |(X_{n+1} - X_n)|$
- Maximum Variation $= \max_n |(X_{n+1} - X_n)|$

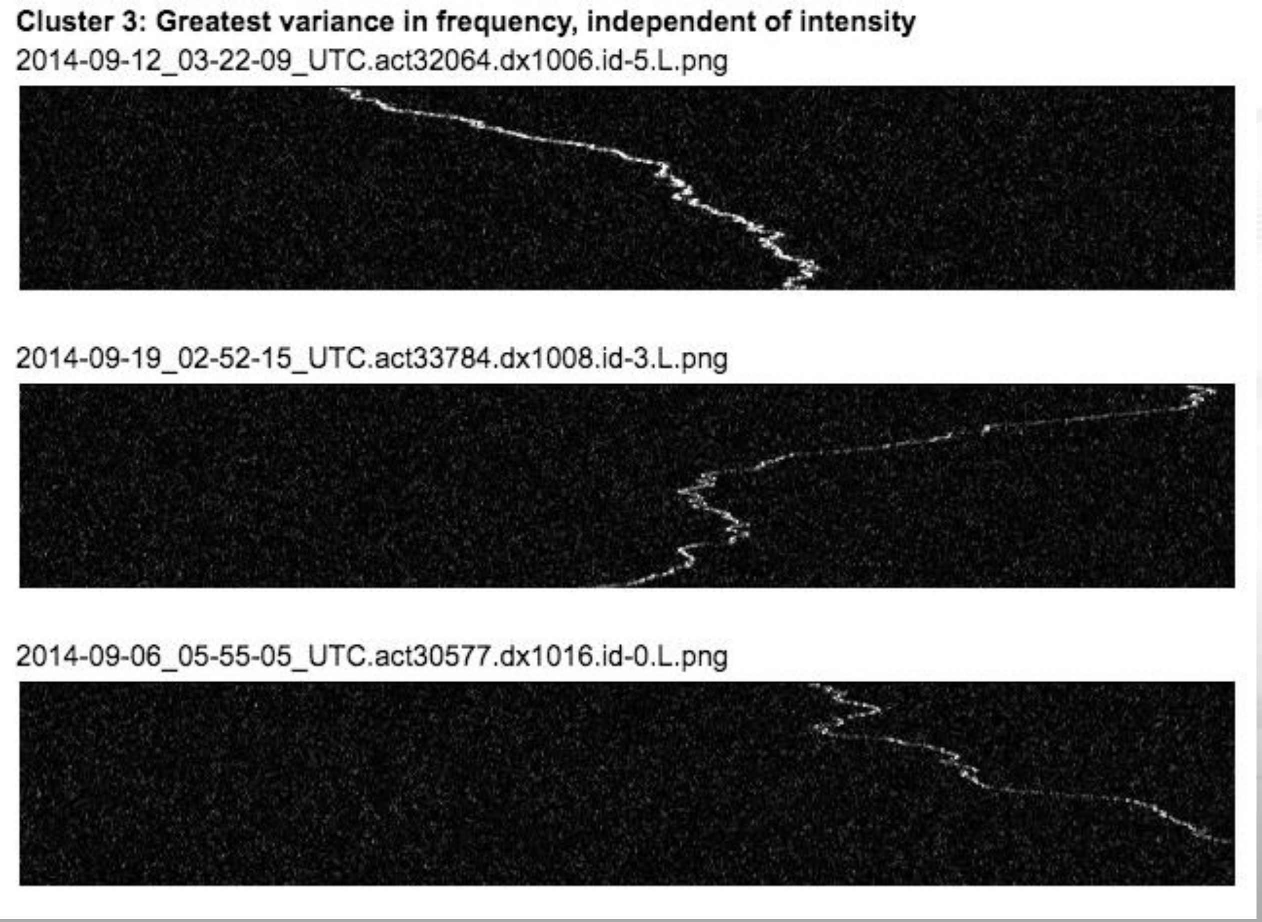
3. Signal Classification – Scalar Features





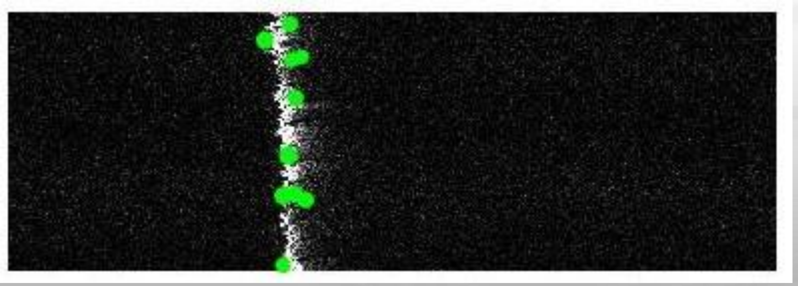
3. Signal Classification – Stanford Research - Experimentation with Novel Feature Extraction

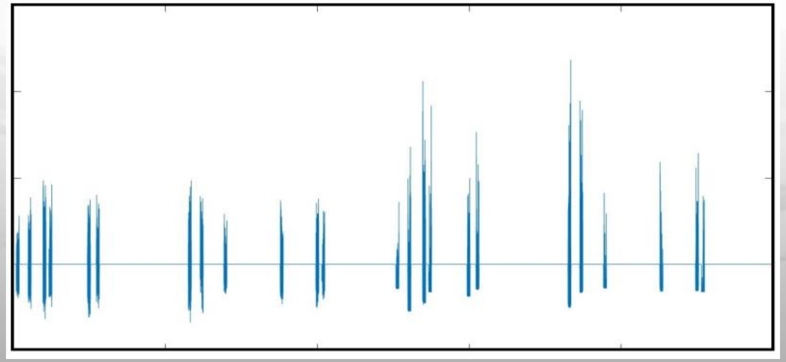
- Example test case: “Squiggle” signals – random modulation of a narrow band signal



3. Signal Classification – Stanford Research

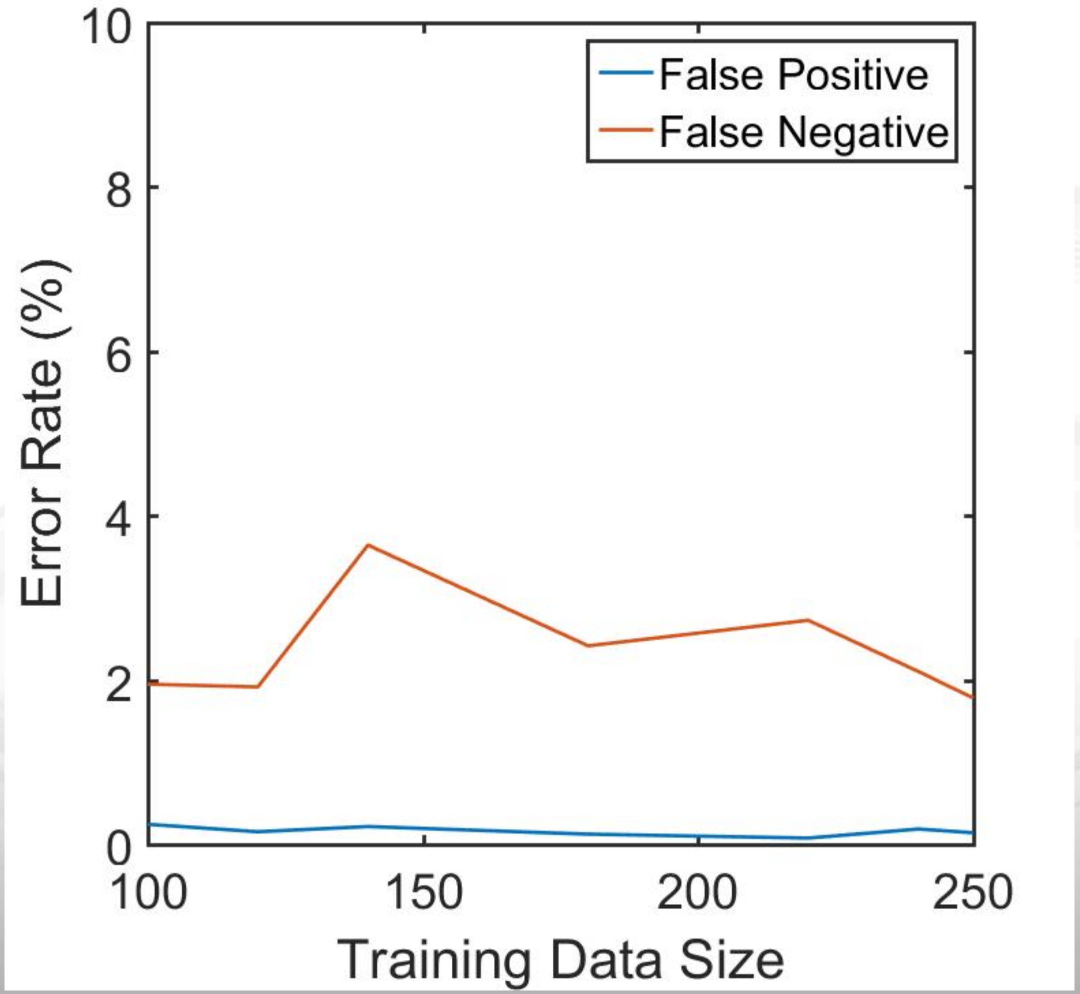
- Feature Compression - Fisher Vectors
- Fisher Vector Calculation:
  1. Find SIFT features (Scale-invariant feature transform - used for image recognition … scalar & drift independent
  2. Collect them into a gaussian mixture model
  3. Residuals from the model are then fisher vectors





3. Signal Classification – Stanford Research

- Fisher Classification Scheme
- Scheme
  - Training: create clusters
  - Classification: nearest cluster
- Best case error:
  - 14/833 false pos.
  - 18/7438 false neg.

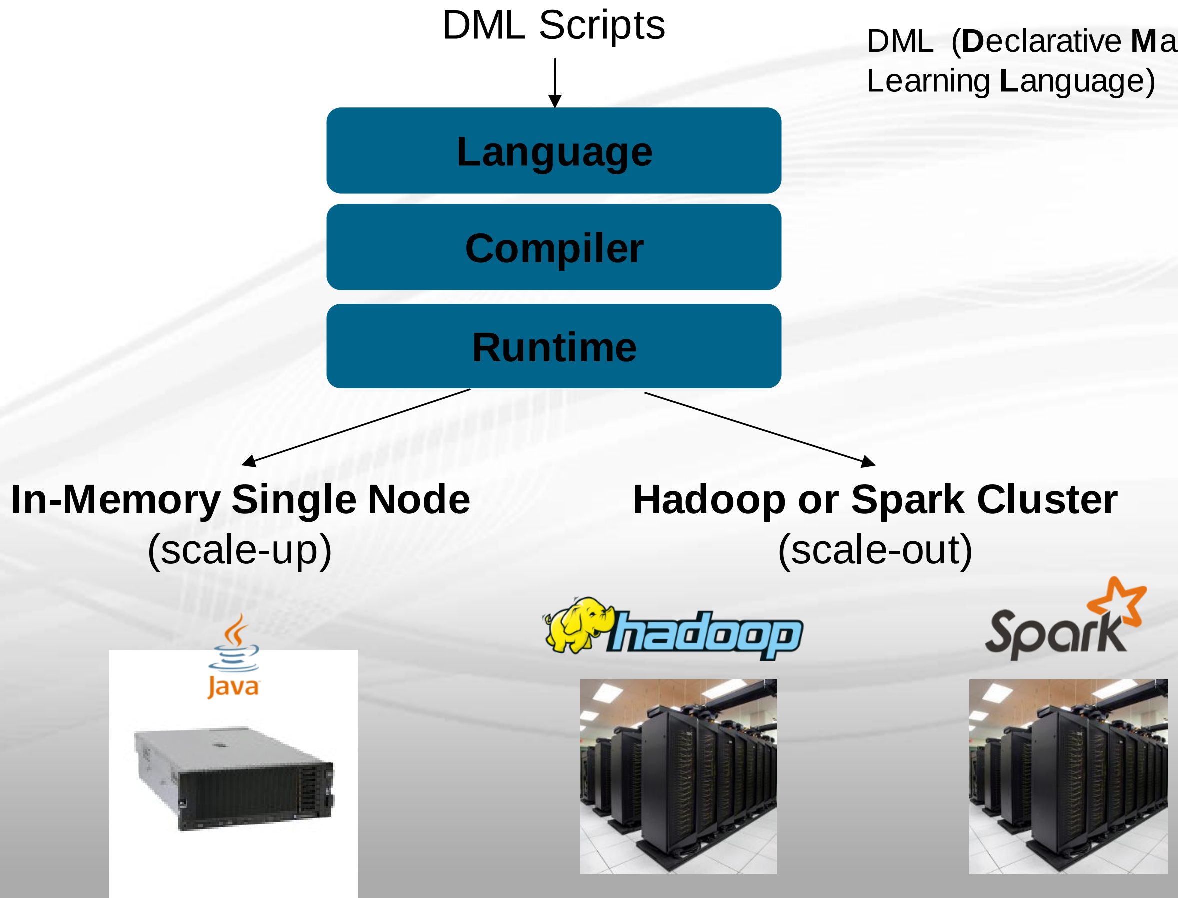


IBM SystemML

- Apache Incubator Project

What is SystemML

- In a nutshell
  - Provides a language for data scientists to implement machine learning algorithms
    - Also comes with approx. 20 algorithms pre-implemented
  - Compiles execution plans ranging from single node (scale up multi threaded) to scale out (MapReduce, Spark)
  - Runs in embeddable, standalone, and cluster mode
- Status of SystemML
  - Shipped with IBM BigInsights 4.x
    - Runs scalable algorithms through IBM Big R
  - Apache SystemML Incubator project
    - http://systemml.apache.org
  - Ongoing research effort at IBM Almaden Research Center



DML (**D**eclarative **M**achine **L**earning **L**anguage)

SystemML Overview

- Machine learning language for data scientists ("The SQL for ML")
  - Productivity of data scientists
  - Declarative, high-level language with R-like syntax (also Python)
- Compiler
  - Cost-based optimizer to generate execution plans, parallelize
    - Based on data and system characteristics
  - Operators for in-memory single node and cluster execution
- Performance & Scalability through scale-up and scale-out
- Broad class of algorithms and growing

```
# LINEAR REGRESSION USING CONJUGATE GRADIENT METHOD
...
w = matrix (0, rows = m, cols = 1);

r = - t(X) %*% y;
p = - r;
norm_r2 = sum (r ^ 2);
norm_r2_target = norm_r2 * $tolerance ^ 2;

while (i < max_iteration & norm_r2 > norm_r2_target)
{
        q = t(X) %*% X %*% p + lambda * p;
        alpha = norm_r2 / sum (p * q);
        w = w + alpha * p;
        r = r + alpha * q;
        old_norm_r2 = norm_r2;
        norm_r2 = sum (r ^ 2);
        p = -r + (norm_r2 / old_norm_r2) * p;
        i = i + 1;
}
...
write (w, $B);
```