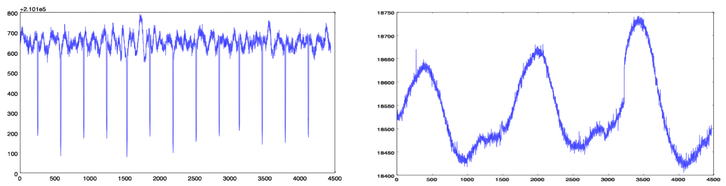
Artificial Intelligence on the Final Frontier:
Using Machine Learning to Find New Earths

Abraham Botros
abotros@stanford.edu

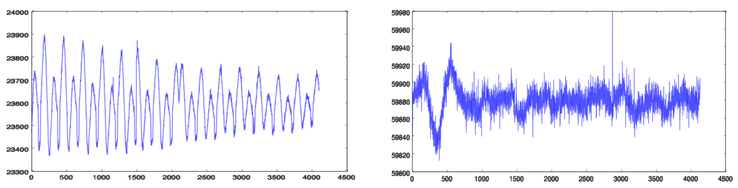
Problem

For ages, mankind has looked to the stars and wondered if a habitable (and inhabited) place like Earth is out there in that dark void. With modern technology, such as NASA's Kepler Space Telescope[1], we can now start to answer such questions. By monitoring changes in observed brightness from distant stars ('light curves'), we can detect phenomena known as 'planetary transits', where an extrasolar planet ('exoplanet') orbits in front of its host star, causing a distinct dip in brightness. However, light curves are often noisy, dense, not directly comparable, and even misleading, especially since transits must be quick and periodic. For all these reasons, we seek to apply machine learning to this problem to algorithmically find evidence of exoplanetary transits – and to find new Earths out there among the stars.

Data

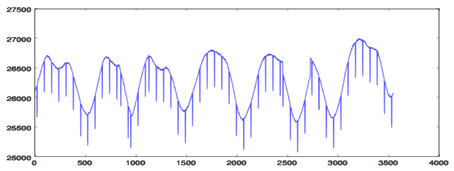


(Top-left) Relatively 'easy' example of a confirmed-planet curve – peaks are obvious, non-peaks fairly stationary. (Top-right) Relatively difficult confirmed-planet example – the entire curve is fluctuating, and peaks are not as obvious. (Bottom-left) Fairly tricky false-positive example, where peaks are periodic and consistent, but no exoplanet is present. (Bottom-right) Standard example of the noisy examples that dominate the dataset, especially the false-positives.

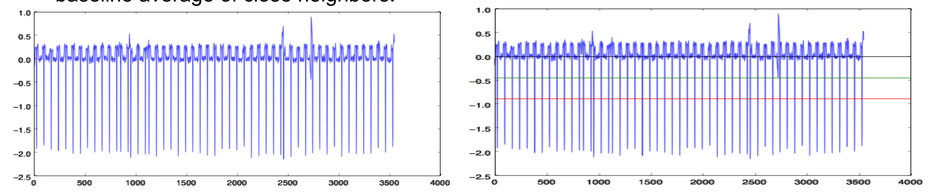


- ~3,000 light curves from the NASA Exoplanet Archive[2]
- Ground-truth labels for all stars: (a) false-positive (no planets actually present), or (b) confirmed-planet (at least one exoplanet present)
- Brightness sampled every 15 minutes for ~90 days
- All files in FITS format; used Astropy[3] Python library
- Main brightness metric: 'PDCSAP_FLUX' – flux (e^-/sec) contained in Kepler optimal aperture

Preprocessing



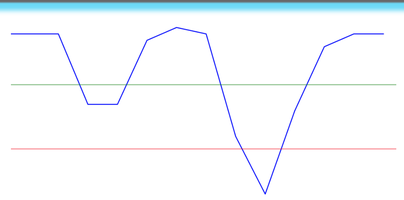
1. Convert to percentage-changes from a local baseline average of close neighbors.



2. Use thresholds to identify peaks for peak feature extraction (below).

- As expected, preliminary analysis showed basic summary statistics (mean, standard deviation, etc.) were nearly identical between classes.
- As a result, more thorough processing of the light curves (in particular, peak feature extraction) was necessary.

Features



- Overall percentage-change mean, median, standard deviation
- Percentage-change non-peak length, peak length, peak max value
  - Mean and standard deviation
  - Evaluated at two separate thresholds (based on standard deviation of percentage-change
- Pairwise products of all above features; all features normalized to [0,1]

Models

- **Nearest Neighbors:** $\arg \min_{(x,y)\in\mathcal{D}_{\text{train}}} \|\phi(x) - \phi(x')\|$
- **Logistic Regression** – with regularization and stochastic gradient ascent (SGA) (hyper-parameters chosen using K-fold cross-validation):

$$\text{MLE: } \arg\max_\theta \ell(\theta) = y \log h_\theta(x) + (1-y)\log\big(1 - h_\theta(x)\big) - \frac{1}{2}\lambda\|\theta\|^2$$

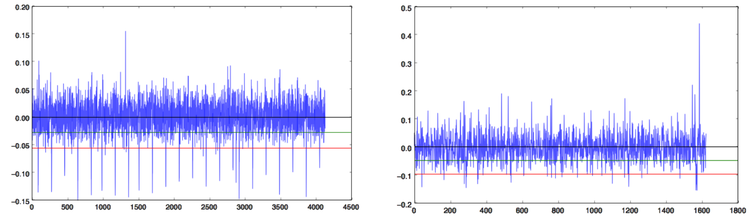
$$\text{Gradient: } \frac{\partial \ell(\theta)}{\partial \theta_j} = \big(y - h_\theta(x)\big)x_j - \lambda\theta_j, \text{ SGA update: } \theta_j := \theta_j + \alpha\Big[\big(y - h_\theta(x)\big)x_j - \lambda\theta_j\Big]$$

- **SVM** with RBF kernel from LIBSVM

Results

Model	Training Error	Test Error
Nearest Neighbors	-	22.6%
Logistic Regression, no regularization	19.5%	18.1%
Logistic Regression, regularization	24.9%	27.8%
SVM	18.3%	19.0%

- Fairly good performance overall – around 80% correct for all models. But still could be better!
- Current features seem to do a decent job of separating the two classes, but likely room for improvement.



- Having trouble believing the left curve above is a false-positive while the right curve is a confirmed-planet curve? So does our algorithm. A significant portion of the train/test error above is from cases like these.

Discussion, Future Work

- Many of the errors come from ambiguous/misleading examples. This is partly the case because stars are actually what are labeled in the NASA database, *not light curves*.
- Since there seem to be various different 'types' within each class, using K-means clustering and then doing multi-class classification might be a good solution.
- New features (higher-dimensional mappings), new algorithms (neural networks), and more examples could all help.
- Discover some new Earths!

References: 1. kelper.nasa.gov. 2. exoplanetarchive.ipac.caltech.edu. 3. astropy.org.