

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

ABRAHAM BOTROS

abotros@stanford.edu

Fall 2014, CS221 & CS229, Stanford University

1 Introduction

For ages, mankind has looked towards the stars and wondered if our Earth is the only place habitable for life. Now, with recent advancements in technology, we can use powerful telescopes, such as NASA’s Kepler Space Telescope [1], to observe stars in the visible universe to see if these extrasolar stars have extrasolar planets (called ‘exoplanets’) like our Sun and its planets. Specifically, looking for phenomena known as ‘planetary transits’ allows us to observe light from far-off stars to determine if they likely have exoplanets orbiting them. Data from such planetary transits and other detection methods has lead scientists to suggest that, over the 100+ billion stars in just our Milky Way galaxy, there may be at least as many planets as well [2].

A planetary transit occurs when an exoplanet crosses in front of a star from the perspective of an observer (in this case, the Kepler telescope). This planetary transit causes a brief dip in the observed brightness of the star. The Kepler telescope gathers time series of observations of brightness from countless stars, with each time series referred to as a ‘light curve’ (see Figure 1). By analyzing these light curves, we can infer if the star might harbor an exoplanet. However, planetary transits are still difficult to identify, as dips in extrasolar star brightness can be due to numerous different reasons, such as binary star systems, pulsating stars, red giants, dust and debris along the line of sight, and any other noisy and unrelated changes in star brightness. Lastly, planetary transits must be both fairly brief and fairly periodic, corresponding to the consistent, repetitive orbiting of a small exoplanet around its parent star.

Due to the challenges that come with the sheer amount of data involved, the difficulty in dealing with time series data, and the complexity in matching the properties that a particular light curve must have to suggest an exoplanet is present, machine learning seems a great fit for this problem. In this project, we apply a variety of machine learning and data manipulation approaches to see if we can accurately classify light curves (using only light curve brightness readings) belonging to extrasolar stars with exoplanets and those without. New and improved algorithms for planetary transit detection allow scientists to better find new Earths out there among the stars.

2 Data

Data was obtained from NASA Kepler observations hosted by the NASA Exoplanet Archive at Caltech. [3] The web interface presents all files in the FITS format [5] with ground-truth labels. We used light curves for confirmed exoplanet-harboring stars (NASA and the astronomical community have concluded that at least one exoplanet orbits this star; hereafter referred to simply as ‘confirmed-planet light curves’)

across all 18 quarters of Kepler light curve data released so far (Q0 through Q17), and light curves for confirmed false-positives (no exoplanets around star, as deemed by NASA/astronomical community; hereafter, ‘false-positive light curves’) from quarters Q1 through Q4 (approximately the same number as confirmed-planet stars across all quarters). We also used only light curves taken at ‘long cadence’ (one ‘cadence’ or observation every 15 minutes; see section 11.1) over each quarter (approximately 90 days). Thus, a given light curve instance consisted of up to about 8,600 measurements; in practice, excluding drop-outs in sensor readings often brought this number down to the 4,000-6,000 range. A little over 2,000 light curves were used for this project (about 1,000 for each of the two classes).

All subsequent processing and analyses was done in Python; in addition, to unpack the FITS files and extract the data, the open-source Astropy library [6] was used. The LIBSVM [7] Python package was used for running SVMs/SVCs, while all other algorithms and code was implemented manually using Python and the Numpy package [8].

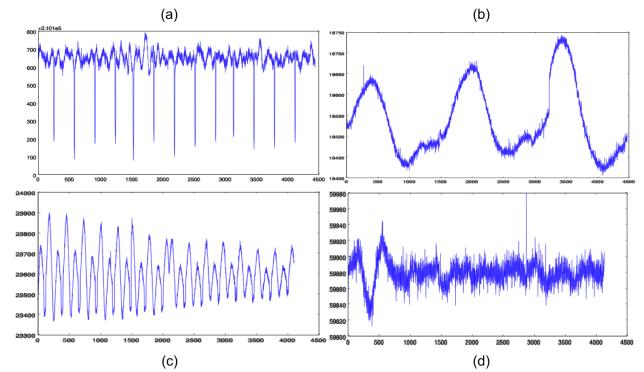


Figure 1: Examples of light curves. (Top) Two differing examples of confirmed-planet light curves. (Bottom) Two differing examples of false-positive light curves. The vertical axis plots the observed brightness (PDC-SAP_FLUX, electrons-per-second flux) of the host star, and the horizontal axis measures the number of valid, observed data points in that quarter.

Within each light curve, for each time point during that quarter, various sensor readings are recorded. In particular, for this project, we want to perform classification using only the minimal sensor readings in the light curve files; moreover, we want to do so using only the changes in brightness (leaving out things like headers about location in space, estimated centroids and centroid changes for host stars, and any other information that could be collected either from the light curve files themselves, or from other outside astrometrical sources). Thus, we focus on each light curve’s ‘PDCAP_FLUX’ reading, which is the flux in units of electrons-per-second con-

tained in the optimal aperture pixels collected by the Kepler spacecraft [9]. In particular, these values are recorded after NASA applied its Presearch Data Conditioning (PDC) module's detrending algorithm to the raw light curve (removing some known error/noise sources and signals). Thus, all data input into the machine learning algorithms discussed in this project will be based solely on raw 'PDCSAP_FLUX' readings.

Figure 1 shows some examples of various light curve PDCSAP_FLUX readings over time. In particular, two confirmed-planet and two false-positive light curves are shown. These curves illustrate some of the fundamental difficulties in attempting classification on this dataset:

- (a) Light curves vary greatly, even at a large scale. For example, in the confirmed-planet class, sometimes (rarely) we get ideal examples (1a), where peaks (representing planetary transits) are very well defined and separable from the otherwise relatively-constant noise; however, we can also get cases where there seems to be more noise than there are defined peaks (1b). Conversely, we can encounter false-positives with numerous peaks (in the form of 1c, and sometimes even closer to those in 1a) that do not pertain to planetary transits (they may correspond to binary star systems, pulsating stars, etc.).
- (b) Light curve brightness levels are often on scales that cannot be directly compared, depending on the overall magnitude of the light from the host star. For example, the units in 1a are an order of magnitude greater than the measurements in 1b.
- (c) Overall, light curve brightness measurements are quite noisy. This includes noise from low-frequency drifting (1b), high-frequency noise (1c), and possible sensor errors. In noisy examples, it is often not immediately clear what should be considered a peak and what should be considered noise (1d is a false-positive example, but many confirmed-planet curves look similar).
- (d) Lastly, there is the problem of somewhat-indirect labeling. In particular, the ground-truth labels are for the actual host star (independent of that quarter's observed light curve), and not for each and every light curve for any quarter that star was observed. Importantly, this means that, just because a curve is from the confirmed-planet class, it is not required that even a single planetary transit be visible in each of its light curves. This is especially true when we consider the time scales involved - each light curve consists of measurements over 90 days, meaning exoplanets with orbital periods greater than that might have no transits

in a given light curve.¹ In general, though, since planetary transits are currently the most common contributor to exoplanet discovery [11], we might expect at least one planetary transit across all observed quarters for a given confirmed-planet scenario, though this is not guaranteed.²

Overall, these issues underscore that the dataset is largely noisy and approximate, with numerous obstacles complicating classification. This is perhaps expected, though, as extra-solar stars and exoplanets themselves vary enormously, and the distances we are dealing with are literally astronomical. For all these reasons, machine learning becomes necessary to solve this problem.

3 Preprocessing

Several preprocessing steps were needed to eliminate noise, normalize between light curves, and extract meaningful features. To reinforce the need for more complicated preprocessing and feature extraction, initial baseline tests using just global statistics of each light curve (mean, median, and standard deviation) as features for the models discussed in Section 5 performed at chance at best (not shown). To illustrate the preprocessing pipeline used here, we will use Figure 2 as a fairly end-to-end example.

3.1 Initial noise removal

Consider the example light curves shown in Figures 1b and 2a (both confirmed-planet examples). Both have significant low-frequency fluctuation, since the curve is oscillating and moving at a relatively large scale. This is irrelevant (and, in fact, distracting and misleading) when looking for planetary transits, as transits must be sharp/brief and cannot be at the scale of the low-frequency fluctuation in these examples. To remove this, we applied a moving average calculation to the raw PDCSAP_FLUX readings³ to calculate a local baseline for each point, then calculated the percentage-change from that corresponding local baseline for each data point. Considering the nearest 15 points - or approximately four hours - on either side seemed to work best. This therefore converted the data from the space of electrons-per-second flux over time (Figure 2a) to percentage-change from a local baseline over time (Figure 2b). As can be seen, this does a great job of eliminating irrelevant overall fluctuation while preserving point-to-point relations and changes.⁴

¹For example, an exoplanet with orbital period similar to Earth's would likely only show a real planetary transit once every three or four quarters. Fortunately, many planets orbit much quicker than Earth; as an extreme example, the fastest known exoplanet orbital period is 0.73 days [11].

²Exoplanets may be discovered/confirmed via other methods besides planetary transit, for example.

³A filter in the style of a FFT high-pass preprocessing step could also perhaps have been used, but Fourier transforms were avoided in this project; see Section 8

⁴Note that we only care about dips in light (negative percentage-changes), so the resulting small positive peaks from percentage-change operation are not at all detrimental.

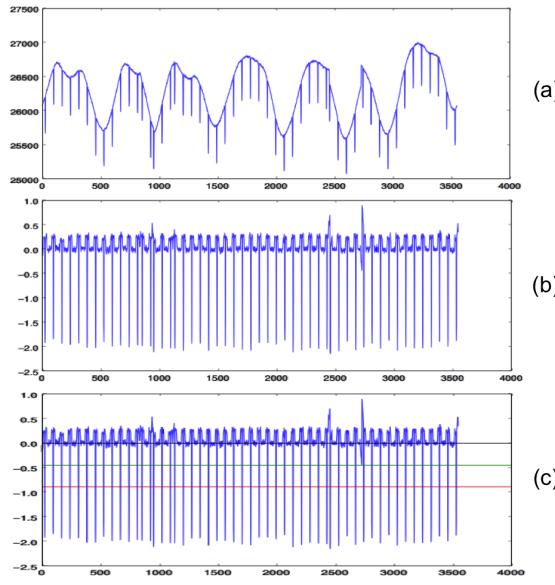


Figure 2: Example of preprocessing steps. Preprocessing outlined for an example confirmed-planet light curve. In all curves, the horizontal axis corresponds to time points, as before. In (a), the vertical axis is the original flux in electrons-per-second, as before in Figure 1; in (b) and (c), the vertical axis is converted to percentage-change from a local baseline using a moving average calculation. In (c), we show the thresholds based on standard deviations below the mean; the green line corresponds to one standard deviation below the mean for this example ('weak peaks'), and the red line corresponds to two standard deviations below ('strong peaks'). As we would hope, the clearly-visible and defined peaks in this example are all labeled as strong peaks.

3.2 Thresholding and peak identification

Now that much of the data has been normalized away, we can simply take value thresholds on the percentage-change metric to identify peaks. This was impossible before due to the non-standardized baselines and units we were dealing with. Using two different thresholds - drops below one and two standard deviations from the mean of the percentage-change data - seemed to work best for identifying peaks; see Figure 2b-c. Intuitively, we want to identify what points in each light curve correspond to strong peaks (and therefore, if they are brief enough, are most likely to correlate to real planetary transits), what points correspond to decent peaks (decent chance of being a real planetary transit), and what points are likely just noise (likely non-peaks). This was done by simply iterating through all points and labeling them as one or more of the following three sets: (1) strong peaks, with percentage-change value less than⁵ two standard deviations below the mean; (2) weak peaks, with percentage-change value less than one standard deviation below the mean⁶; and (3) non-peaks/inter-peak points, with percentage-change above one standard deviation below the mean.

4 Features

Using the output from the preprocessing step outlined above, we can proceed to extract meaningful features from each time series of points. In particular, since time series cannot be eas-

ily and reliably aligned, we need to remove time as a factor when extracting our features. Intuitively, we want to know about the peaks versus the inter-peak stretches. Furthermore, we want to design our features to capture some information regarding: (1) consistent periodicity of peaks - exoplanets should be orbiting their stars at a constant orbital rate, so transits should be relatively periodic; and (2) peaks should be relatively brief - since both Kepler and the exoplanet are moving (possibly even in perpendicular planes), the transit time should be relatively short.

Using 3 as a visual example, we would define both (a) and (c) as peaks for the first threshold, (c) as the only peak for the second threshold, and (b) as a non-peak/inter-peak interval. We would then calculate the following feature set for each light curve to approximate all the meaningful information desired that was discussed above:

1. Fraction of all time points that are in each of the three categories
2. Global mean, median, and standard deviation
3. Number of distinct continuous peaks (two such peaks at (a) and (c) for the first threshold; one at (c) for the second threshold)
4. Peak width (widths of (a) and (c) in time) - mean and standard deviation
5. Peak max magnitude (most negative value of the peak) - mean and standard deviation
6. Overall peak points' values - mean and standard deviation
7. Non-peak/inter-peak width (b) - mean and standard deviation

Where feature subsets for numbers 3 through 7 are computed for both the first threshold and the second threshold separately. Overall, this constitutes 24 different features. To (fairly naively) account for possible relations between these various features (and to be able to place these first-order features into a higher dimension where we might be able to learn a linear boundary that is non-linear in the original feature space), we also take pairwise products of all of these 24 features. When we add in a bias term of 1 (for logistic regression, for example), the first-order features, and all the second-order pairwise features, we get a total number of 325 features, thus meaning that each light curve is represented as a vector in \mathbb{R}^{325} in our feature space.

Since many of the resulting features are still on fairly incomparable scales (which is exponentially exaggerated by some of the learning algorithms used), we normalize each feature using mean removal and dividing by the standard deviation (similar to the preprocessing and normalizing we might do for an algorithm like PCA, for example). In manual inspection of the new scaled feature ranges, this seemed to work well

⁵Note that 'less than' a certain threshold refers to the actual *signed* relation to the mean, and not simply the magnitude distance from the mean.

⁶As a result, points in the two-standard-deviation category are always in the one-standard-deviation category - this intuitively makes more sense since it prevents splitting up a single peak into two perceived peaks at the one-standard-deviation level, and also gave slightly better performance.

in properly scaling features to be comparable to all other features; in testing, this also improved performance (not shown).

To address the concerns we mentioned above, in particular, we model the consistency and periodicity of any planetary transits (peaks in our data) by looking at the standard deviation of the non-peak/inter-peak intervals, and would hope to see minimal standard deviation for confirmed-planet cases. In addition, by inspecting peak width, we approximate how brief peaks for confirmed-planet examples should be in comparison to false-positive examples. Peak max and average values give us information about the strength of the peaks, and our counts and fractions for number of peaks and peak points tell us how many peaks there are and how much they dominate the entire time series.

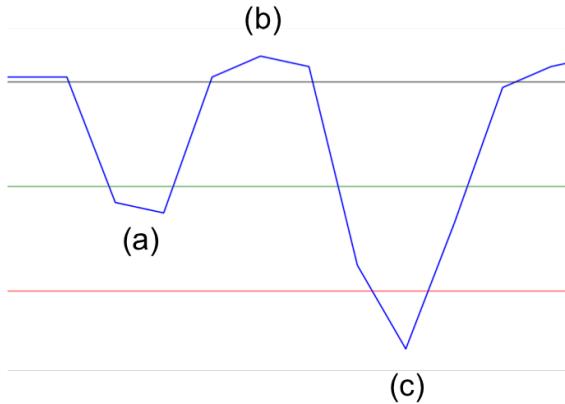


Figure 3: Visual reference for peak feature extraction. Note this is simply a toy example and is not drawn accurately/to scale. The black line represents the mean (usually around 0% percentage-change in the PD-CSAP.FLUX time series), and the green and red line represent the first and second thresholds, respectively. The first threshold corresponds to one standard deviation below the mean, and the second threshold corresponds to two standard deviations below. By separating points as such, we can easily extract features about both the strong peaks, the weak peaks, and the inter-peak intervals.

5 Models and learning algorithms

5.1 Learning algorithms

This is primarily a supervised learning problem, where we want to take the feature vectors constructed above for each light curve and classify them correctly after training with the ground-truth labels. As mentioned, we have two ground-truth classes (confirmed-planet and false-positive), and a large feature vector for each light curve. We used four distinct machine learning algorithms to perform classification on this dataset: **k-nearest neighbors**, **logistic regression**, **softmax regression**, and **SVM**. All algorithms except SVM were implemented manually in Python with Numpy [8] (SVMs were run using LIBSVM [7]).

For **k-nearest neighbors**, at test time, we want to find the k most similar training examples to our test input and simply approximate the proper test label using the ground-truth training labels. To find the nearest neighbor to our test

⁷As we can see, the last parameter in our hypothesis function is redundant, due to sum of total probability (all entries must sum to 1). However, in practice, when we use regularization (for any $\lambda > 0$, the cost function is now strictly convex and we are guaranteed a unique solution. This allows us to keep all the parameters $(\theta_1, \dots, \theta_k)$ without arbitrarily setting one of them to zero [12].

⁸Subject to the constraints: $a_i \geq 0, i = 1, \dots, m$ and $\sum_{i=1}^m a_i y^{(i)} = 0$.

input in our training set ($\mathcal{D}_{\text{train}}$), we find:

$$\arg \min_{(x,y) \in \mathcal{D}_{\text{train}}} \|\phi(x) - \phi(x')\| \quad (1)$$

Then simply return the found label y ; this can be easily generalized to the k-nearest neighbors algorithm, where we instead return the mode of the nearest k neighbors' labels.

In **logistic regression**, we want to find a decision boundary for binary classification. We use regularization and stochastic gradient ascent (SGA) to find the MLE weights:

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

$$\text{Gradient: } \frac{\partial \ell(\theta)}{\partial \theta_j} = (y - h_{\theta}(x)) x_j - \lambda \theta_j$$

$$\text{SGA update: } \theta_j := \theta_j + \alpha \left[(y - h_{\theta}(x)) x_j - \lambda \theta_j \right]$$

For **softmax regression**, we apply K-means (see Section 5.2) to break down the two parent classes into K new child classes. This allows us to do learning in the multi-class classification environment (see Section 6), and in final testing output and test against the two parent class labels. In general, in softmax regression, we modify our logistic regression hypothesis function to handle classification problems where the output label/class y is in $\{1, \dots, k\}$, as partitioned by K-means:

$$h_{\theta}(x) = \begin{bmatrix} \frac{\exp(\theta_1^T x)}{\sum_{j=1}^k \exp(\theta_j^T x)} & \dots & \frac{\exp(\theta_k^T x)}{\sum_{j=1}^k \exp(\theta_j^T x)} \end{bmatrix}^T$$

Where each entry in the hypothesis matrix above is of the form:⁷ $\phi_i = p(y = i|x; \theta) = \frac{\exp(\theta_i^T x)}{\sum_{j=1}^k \exp(\theta_j^T x)}$, and we can use stochastic gradient descent with the following gradient:

$$\nabla_{\theta_j} J(\theta) = - \left[x(1\{y = j\} - p(y = j|x; \theta)) \right] + \lambda \theta_j$$

For **SVMs and support vector classification (SVC)**, we want to find an optimal-margin decision boundary by solving the SVM dual optimization problem:⁸

$$\max_a W(a) = \sum_{i=1}^m a_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} a_i a_j \langle x^{(i)}, x^{(j)} \rangle$$

An RBF/Gaussian kernel was applied to our SVM using LIBSVM.

All hyperparameters (such as regularization terms, number of maximum iterations, step size type, and convergence criteria) for logistic regression and softmax regression were chosen using manually-implemented K-fold cross-validation (due to excessive runtime and limited resources, 5-fold cross-validation was used). Hyperparameters for SVM classification were chosen automatically using cross-validation as part of the given LIBSVM tools.

5.2 Other algorithms

To aid in some of the analyses, other algorithms were enlisted, including **K-means** and **PCA**. K-means was implemented manually in Python using Numpy, while the built-in PCA function of matplotlib [13] was used for running PCA.

K-means is an algorithm for unsupervised clustering where we want to minimize the distortion function:

$$J(c, \mu) = \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

Where we want each training example $x^{(i)}$ to be assigned to the optimal cluster centroid $\mu_{c^{(i)}}$ via its assignment $c^{(i)}$ to minimize J ; we can do this using the K-means algorithm, where we (1) assign each training example to the closest cluster centroid, then (2) move each cluster centroid to the mean of the points assigned to it. For each distinct call to K-means, we repeatedly ran the algorithm with different random initializations to find the assignments that most successfully minimized the distortion value.

PCA, or Principal Components Analysis, allows us to reduce dimensionality (especially helpful considering our very large feature space) by finding the k principal components (which give a lower, k -dimensional basis and approximation for our original features). To do so, we simply find the top k eigenvectors ($u_1, \dots, u_k; k < n$, where n is the length of our original feature vectors) of the empirical covariance matrix from the data (assuming we have already subtracted the mean and divided by the standard deviation, which we discussed in Section 3). Therefore, to represent an original feature vector x in this new basis, we simply compute:

$$y = [u_1^T x \quad \dots \quad u_k^T x]^T \in \mathbb{R}^k$$

We can then do learning in this lower, k -dimensional feature space by transforming all our inputs using this basis, allowing more efficient learning.

6 Results

Using the optimal hyperparameters for each algorithm obtained via K-fold cross validation (as explained in Section 5.1), we obtained the performance metrics shown in Table 1. Performance is in terms of percent-correct, and is shown for the hyperparameters for each algorithm that lead to the best (cross-validation) performance.⁹

⁹For K-nearest neighbors, $K = 3$ performed best. For logistic regression, setting a max iteration limit of 2,000 iterations through the entire dataset, along with zero regularization and step size equal to one over the square root of the number of updates made so far, did the best (though training with only around 500 iterations gave nearly-similar performance). For SVM classification, the best hyperparameters were $c = 512$ and $\gamma = 0.03125$.

¹⁰If $K = 3$, for example, we would have 3 child classes for confirmed-planet curves and 3 for false-positive curves, for a total of $K * 2 = 6$ possible child classes to perform classification over.

¹¹For 10 K-means clusters, the optimal hyperparameters for softmax regression were roughly 1,000 training set iterations using stochastic gradient descent (though training with as few as 200-300 iterations gave nearly-similar performance), with a relatively small regularization parameter (0.001) and dynamic step size (one over square root of number of updates made so far).

Learning algorithm	Train Performance	Test Performance
K-Nearest Neighbors	-	83.9%
Logistic Regression	87.1%	84.5%
SVM, RBF Kernel	82.29%	83.3%
Softmax Regression	87.0%	85.6%

Table 1: Performance of learning algorithms on full training and testing sets.

Before adding softmax regression, logistic regression (with no regularization) had the best test performance on this dataset. However, we wanted to see if we could further improve performance. To do so, we reasoned that, since exoplanets usually fall in different buckets of shapes, sizes, orbits, and other physical characteristics (Earth analogs, Super-Earths, gas giants, etc.), we might be able to first cluster them to learn better decision boundaries for each sub-class of confirmed-planet and false-positive light curves.

This indeed ended up giving a small increase in performance (see Figure 4), though it seems like the softmax regression performance may be highly dependent on the nature of the clustering (the cluster assignments that minimize the K-means distortion function are not necessarily the ones that maximize softmax regression performance).

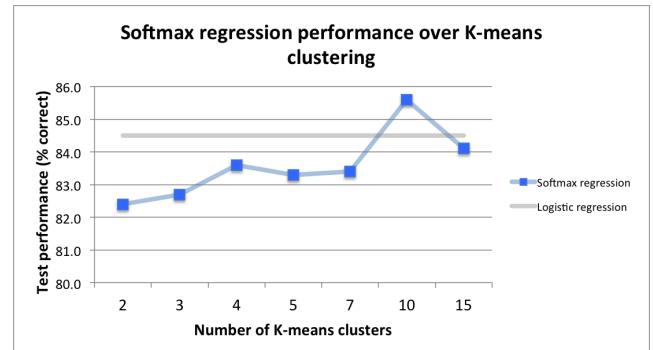


Figure 4: Softmax regression performance as a function of number of K-means clusters. In particular, $K = 10$ enabled the best softmax regression classification performance, surpassing logistic regression.

In particular, we used K-means clustering to first do unsupervised clustering of each of the two parent classes independently. Thus, each of the two parent classes was broken down into K new child classes.¹⁰ In doing so, we might be able to better learn how to distinguish various distinct subtypes (sub-classes or clusters), instead of having to group all instances of each parent class together when learning a decision boundary. To then perform classification, we learn using softmax regression over the new child classes, choose optimal hyperparameters¹¹ using K-fold cross validation over the child class labels on the validation sets, and then at the final

test phase simply output and test against the child’s parent class label instead of the child class label. Putting this all together gave the optimal performance out of all other learning algorithms explored here (though overall, all algorithms performed similarly).

Lastly, we briefly attempted to run PCA to reduce the dimensionality of the feature space, taking only the k top principal components and re-running tests for each value of k over all learning algorithms with their optimal hyperparameters. This worked surprisingly well and speaks to PCA’s ability to reduce dimensionality while preserving variance; Figure 5 shows that performance of most algorithms stayed fairly high even when using a minimal basis of principal components (as few as 5-50 principal components down from our original 325, depending on the algorithm). Of note, logistic regression and softmax regression saw fairly sizeable declines as the number of principal components became quite small; since SVMs were still able to learn in these spaces, we hypothesize that the reduced principal components remaining may have corresponded to some of the more linearly-related features in the dataset, so logistic regression and softmax regression were unable to learn non-linear decision boundaries (the SVM’s RBF kernel might have allowed it to do continue to do so by allowing it to work in a higher-dimensional feature space due to inner products). At any rate, the data shows that we can get away with as few as 50 principal components in almost all cases and still perform fairly similarly to performance using our full feature space!

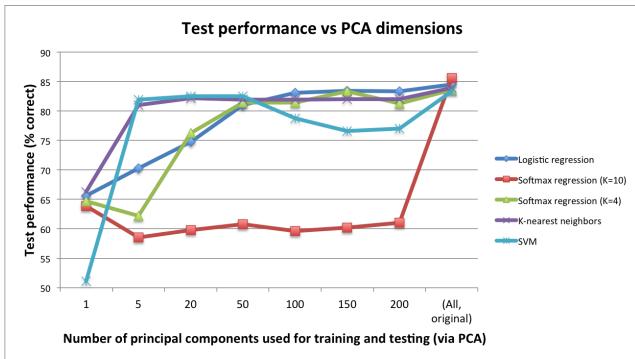


Figure 5: Learning algorithm test performance as a function of PCA component count. Note that most learning algorithms maintained good performance, even when using a very reduced number of features. Also note that two versions of K-means coupled with softmax regression were run for this figure: the performance of softmax regression when K-means clustered points into 10 clusters and PCA reduced dimensionality was very poor, so we also sampled test performance for softmax regression with K-means clustering into 4 clusters (the second-best cluster number from our earlier K-means testing; not shown). In addition, a bias term was still added in to all feature vectors, in addition to the number of principal components shown at each data point. Lastly, the final data points correspond to the original dataset and full feature space, with no PCA applied.

¹²We were unable to find hard numbers regarding corresponding performance from systems used by NASA and other astronomical research groups. However, it is important to note that stars that are labeled as possibly harboring exoplanets due to detected planetary transits first become ‘objects of interest’ or ‘exoplanet candidates’, and are confirmed through more thorough investigation (and possibly other methods, such as spectrometry) before becoming a confirmed exoplanet.

¹³Crowd-sourced efforts like that at PlanetHunters.org attempt to use human pattern recognition to properly classify complex light curves that computer algorithms might have missed. Sites like this include more ‘formal’ training, so human performance likely increases significantly, at least for many of the edge-cases that could be specifically trained on.

7 Error analysis

By examining the examples that were classified incorrectly, we can get a better idea of where the algorithms are not properly learning, and what examples and features might be confusing the algorithms. In particular, across all algorithms, many of the examples similar to those shown in Figure 6 were misclassified. Figure 6a shows a false-positive light curve; however, it appears to have all the characteristics of a confirmed-planet curve, including strong, defined, and consistently periodic peaks. On the other hand, Figure 6b shows a confirmed-planet light curve that appears to be more akin to random noise. Examples like these are extremely tricky for the algorithms shown, as the features for these types of confusing examples are likely very similar to the opposite class. Such light curves exemplify many of the problems outlined in Section 2. Properly classifying such curves would likely involve bringing in outside information other than just raw brightness changes, and might also benefit from linking light curves from the same star across different quarters (see Section 8).

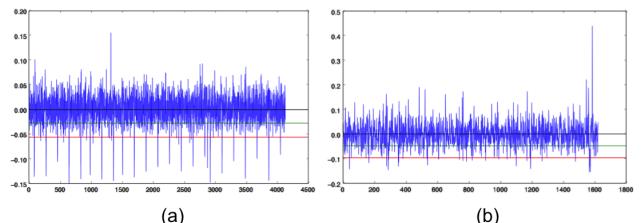


Figure 6: Examples of light curves that were often misclassified by all algorithms. Example (a) is actually a false-positive light curve, though it appears to have distinct, periodic peaks. Example (b) is actually a confirmed-planet light curve, though it looks more like noise. In particular, compare these to the examples in Figure 1.

8 Discussion

Overall, we present a system for preprocessing, feature extraction, and machine learning on a large dataset of Kepler Telescope light curves for binary classification of stars as potentially harboring exoplanets or likely having none. This system performs quite well, with up to around 85% accuracy in testing. As more time passes and more exoplanets are discovered (we are still in the relatively early stages of exoplanet discovery, especially using planetary transits), the system will have more data to train on and could therefore perform even better in the future. In its current state, we guess that it performs likely on a similar level to NASA algorithms, though more literature research is needed to validate this assumption.¹² In addition, using informally-trained human observation as a proxy to an oracle, the author was able to achieve around 70% correct in binary classification, meaning the learning algorithms discussed in this work perform even better than humans with only informal training.¹³

All algorithms performed well on this manually-extracted feature set; however, this is not to say that this was the first feature set tried. Countless different feature extract pipelines, methods, components, and feature vector sizes were tried before reaching the final performances shown above. In particular, for preprocessing and feature extraction, numerous peak thresholds, peak characteristics, normalization/scaling techniques, and set sizes were entertained along the way.

There is likely also much room for improvement - in particular, more analysis could be done to better identify which features are most important and which are distractors by evaluating metrics such as mutual information.¹⁴ In addition, more domain knowledge beyond the author's limited knowledge could be injected into the data extraction and feature selection. While one of the primary goals of this project was to classify using only light curve brightness metrics in a single light curve, incorporating data from other sources (spectrometry, locations in space, knowledge about each star, centroid changes for host stars, etc.) and linking light curves for the same star across different quarters (a given star should only be given one label across all observed light curves/quarters) would likely both help classification.

It should also be noted that Fourier transforms and wavelet analysis could likewise have been applied in the preprocessing and feature extraction steps, but we chose to avoid these techniques for a couple reasons. For one, the author had minimal experience with these techniques before, meaning it would have been quite difficult to properly and thoroughly explore and apply these techniques in the minimal time available. In addition, we reasoned that the information was not in the frequency domain, since we really cared about very small-scale, local, point-to-point changes; thus, doing any transform (whether using a sliding Fourier transform window or wavelet analysis to maintain some time-to-time information) would likely have smoothed out and lost some of the crucial point-to-point information, by definition of these techniques. Lastly, we reasoned that, at best, the output of these techniques would give a somewhat smoothed-out version of the initial data points, and we would still have to run some form of peak detection to eliminate noise and isolate only peaks - which would be much more difficult with smoothed-out data. However, that said, given more time, sliding Fourier transforms and wavelet analysis would be some of the first new techniques to apply to the preprocessing and feature extraction pipeline described here.

Other algorithms could also be tested on this (and any augmented) feature set. For example, SVMs with other kernels, neural networks, and other such algorithms might give good performance here, especially if there are certain nonlinear decision boundaries that we were unable to learn using our approach. Even more hyperparameter tweaking could likely give slightly better performance, too, but the exorbitant time needed to run the algorithms and tests already discussed precluded doing so.

Overall, with minimal domain knowledge and a general

approach to the raw data, given a light curve consisting of brightness readings for far-off extrasolar stars over 90 days, we were able to preprocess and extract features relevant to planetary transits, learn over these features using machine learning algorithms, and classify the stars as having exoplanets or being false-positives with relatively high probability (around 85%). We would say that, as a whole, this project was a success, and could even be used to classify exoplanet-harboring stars and identify exoplanets that have yet to be discovered. We hope to continue to develop and test this system as more exoplanet data surfaces, as it would be very interesting to see if predictions on future data are reliable and can classify exoplanet systems that have not even been confirmed. All in all, this work contributes to mankind's ongoing efforts to better understand our Earth, our universe, and our place in the stars as we search for new Earths on the final frontier.

9 Final note

As a final note, I want to thank all the people who made CS221 and CS229 possible, including Dr. Percy Liang, Dr. Andrew Ng, and all the TA's who put an incredible amount of work into these classes. These classes, and this project, were a great learning experience, so thank you! - Abraham Botros

10 References

- [1] See <http://kepler.nasa.gov/>.
- [2] Cassan, A., D. Kubas, J.-P. Beaulieu, M. Dominik, K. Horne, J. Greenhill, J. Wambsganss, J. Menzies, A. Williams, U. G. Jorgensen, A. Udalski, D. P. Bennett, M. D. Albrow, V. Batista, S. Brillant, J. A. R. Caldwell, A. Cole, Ch. Coutures, K. H. Cook, S. Dieters, D. Dominis Prester, J. Donatowicz, P. Fouque, K. Hill, J. Kains, S. Kane, J.-B. Marquette, R. Martin, K. R. Pollard, K. C. Sahu, C. Vinter, D. Warren, B. Watson, M. Zub, T. Sumi, M. K. Szymanki, M. Kubiak, R. Poleski, I. Soszynski, K. Ulaczyk, G. Pietrzynski, and L. Wyrzykowski. "One or More Bound Planets per Milky Way Star from Microlensing Observations." *Nature* 481.7380 (2012): 167-69. Web.
Also see http://www.nasa.gov/mission_pages/kepler/news/kepler20130103.html, and <http://www.space.com/19103-milky-way-100-billion-planets.html>.
- [3] See <http://exoplanetarchive.ipac.caltech.edu/index.html>.
- [4] See <http://archive.stsci.edu/kepler/>.
- [5] See <http://fits.gsfc.nasa.gov/> and <http://en.wikipedia.org/wiki/FITS>.
- [6] See <http://www.astropy.org/>.
- [7] See <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- [8] See <http://www.numpy.org/>.
- [9] See http://archive.stsci.edu/kepler/manuals/archive_manual.pdf.
- [10] See <http://kepler.nasa.gov/education/activities/transitTracks/>.
- [11] See collected information at http://en.wikipedia.org/wiki/Discoveries_of_exoplanets.
- [12] See http://ufldl.stanford.edu/wiki/index.php/Softmax_Regression and <http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/>.
- [13] See http://matplotlib.org/api/mlab_api.html#matplotlib.mlab.PCA.

¹⁴We started an investigation into correlation of various features with labels, but were unable to complete this in the time allotted, and opted for the PCA approach instead.

11 Supplemental

11.1 ‘Long cadence’ vs ‘Short cadence’ light curves

‘Long’ vs ‘short’ cadence refers to the frequency of readings. In long cadence light curves, exposures are taken every 15 minutes; in short cadence light curves, there is only one minute in between each sampling. Long cadence curves are over one quarter/90 days, while short cadence curves vary over shorter time spans. We found long cadence light curves to be much more abundant in the dataset, and intuitively it seems like they would help avoid some of the noise possibly present at the higher sampling rate. We believe short cadence light curves are supposed to be more helpful for more detailed studies of the actual star, and not as appropriate here for planetary transit detection.