Exoplanet Detection Using Machine Learning

~Tanishq Som

What are Exoplanets and How to detect them?

Planets are heavenly bodies comprised of gas, dust etc., orbiting a star. All planets outside our solar system that revolve around a star are called Exoplanets **[1]**.

But why searching Exoplanets is important?. Is it really worth searching? Whether life is possible outside our solar system? is a profound question of all time. If a planet rich with life is to be found, then it will change humankind forever. Apart from this, this question will also answer the most fundamental questions about our existence. This is where Exoplanet detection comes into the picture.

When it comes to detecting exoplanets, there are several techniques available. Each technique, quite often sensitive to different types of planets, is briefly summarised below. Amongst these techniques the direct imaging and observations of transiting exoplanets have had the greatest impact on atmosphere characterisation and as such these methods will be discussed in more detail in the subsections below.

The main techniques used to detect exoplanets are: [2]

I. Direct imaging

The exoplanet is imaged directly using large telescopes fitted with adaptive optics and coronagraphs. The technique is most sensitive to the warmer, bright (young) and massive exoplanets on wide and/or eccentric orbits (large sky projected separations). The separation from the host star allows for spectra to be obtained directly and allows for the direct measurement of the luminosity.

II. Radial velocity

The exoplanet is detected by measuring the Doppler shift in the host star light, a consequence of the gravitational affects between the two bodies. The technique is most sensitive to exoplanets with a large mass orbiting close to their host star perpendicular to the plane of the sky. The radial velocity technique allows for a minimum mass (dependant on orbital inclination) to be calculated.

III. Transits

The exoplanet is detected by measuring a periodic decrease in the flux received from the host star, as a consequence of the exoplanet transiting in front of the host star. The transiting technique is most sensitive to large exoplanets orbiting close to their host star stars and provides an accurate determination of the planetary radius relative to the host star.

IV. Microlensing

The exoplanet is detected by measuring characteristic light curve changes caused by changes in the lensing effect observed when a star with a planet passes in front of a distant star. The technique is limited to distant one time events and by the lack of accurate determinations of the planet and orbit parameters. It is however a very valuable technique due to the lack of strong radii or mass biasses making it ideal for statistical population studies.

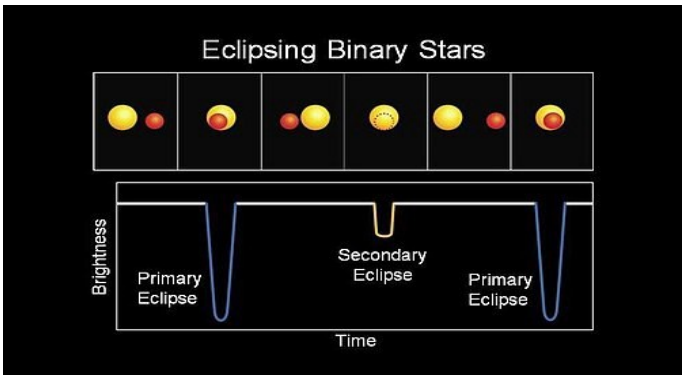
V. Transit timing variations

The exoplanet is detected by observing a change in periodic phenomena due to the presence of an exoplanet. Examples include a change in transit time (known as TTV) of one planet, due to the presence of others in multiple planet systems and pulsar timing, where anomalous movement (measured at radio wavelengths) can be used to infer the presence of a planet.

Due to glaring bright light from a star, Exoplanets are difficult to detect using just a telescope. To answer this problem, scientists have devised a method for the detection of these distinct planets. Instead of directly viewing these planets through telescopes, which is not always feasible, they look out for the effects these planets have on the stars they orbit.

One way to find these planets is by looking out for unsteady stars. A star about which a planet is revolving tends to wobble. This is due to the mass of the revolving planet. Many planets have been discovered using this technique. But the problem is, only massive planets like Jupyter can have a gravitational impact on its star, which can cause the star to wobble. Smaller planets like Earth have less impact on a star, thus making the unsteady motion difficult to detect. Then how to detect smaller Exoplanets? Kepler **[3]** detected smaller planets using ‘transit method’. A transit is when a planet

passes in front of its star and the observer. Due to this transit, there is a small drop in the intensity of the light reaching the observer. Thus making it less bright. A planet revolving around a star will show a periodic dip in light intensity. This can be seen in the below figure



(https://en.wikipedia.org/wiki/Transit_(astronomy))

The primary Eclipse denotes the dip in the intensity of light reaching the observer from the star due to the Exoplanet obstruction. Thus by studying the time interval between consecutive transits, one can classify whether it is a planet or some celestial body.

Training machines to identify candidates for exoplanets

With real datasets provided by NASA and Caltech we can train our machines to detect true candidates for exoplanets. Idea is to create a machine learning model that can predict if an observation is a real candidate for an exoplanet or not. The data was collected by the Kepler mission that revealed thousands of planets out of our Solar System. Kepler was able to find planets by looking for small dips in the brightness of a star when a planet transits in front of it. It is possible to measure the size of the planet based on the depth of the transit and the star's size.

Machine Learning models we can use:

I. KNN

KNN algorithm is one of the classic supervised machine learning algorithms **[4]**, this means that the data used for training a KNN model is a labeled one. It is capable of both binary and multi-class classification **[5]**. In the machine learning terminology, a classification problem is one where, given a list of discrete values as possible prediction outcomes (known as target classes), the aim of the model is to determine which target class a given data point might belong to. For binary classification problems, the number of possible target classes is 2. On the other hand, a multi-class classification problem, as the name suggests, has more than 2 possible target classes. A KNN classifier can be used to solve either kind of classification problems. KNN is a non-parametric algorithm. Therefore, training a KNN classifier doesn't require going through the more traditional approach of iterating over the training data for multiple epochs in order to optimize a set of parameters. Rather, the actual training process in the case of KNN is quite the opposite. Training a KNN model involves simply fitting *(or saving)*all the training data instances into the computer memory at the same time, which technically requires a single training cycle. After this is done, then during the inference stage, where the model has to predict the target class for a completely new data point, the model simply compares this new data with the existing training data instances. Then finally, on the basis of this comparison, the model assigns this new data point to its target class. But now another question arises. What exactly is this comparison that we are talking about, and how does it occur? Well quite honestly, the answer to this question is hidden in the name of the algorithm itself — *K-Nearest Neighbors*.

◆As the first step, our KNN model calculates the distance of this new data point from every single data point within the 'fitted' training data.

◆ Then, in the next step, the algorithm selects 'k' number of training data points that are closest to this new data point in terms of the calculated distance. For the calculation of the distances between the data points, we use the Euclidean distance formula **[6]**

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

◆ Finally, the algorithm compares the target label of these 'k' points that are the nearest neighbors to our new data point. The target label with the highest frequency among these k-neighbors is assigned as the target class to the new data point.

II. Decision Tree

Decision Trees are another Supervised Machine Learning where the data is continuously split according to a certain parameter **[7].** The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split. There are many algorithms out there which construct Decision Trees, but one of the best is called as **ID3 Algorithm [8]**. ID3 Stands for **Iterative Dichotomiser 3**. Before discussing the ID3 algorithm, we'll go through few definitions.

Entropy [9] also called as Shannon Entropy is denoted by H(S) for a finite set S, is the measure of the amount of

uncertainty or randomness in data. Intuitively, it tells us about the predictability of a certain event.

Information Gain [10] Information gain is also called as Kullback-Leibler divergence denoted by IG(S,A) for a set S is the effective change in entropy after deciding on a particular attribute A. It measures the relative change in entropy with respect to the independent variables.

$$IG(S, A) = H(S) - H(S, A)$$

Alternatively,

$$IG(S, A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

where IG(S, A) is the information gain by applying feature A. H(S) is the Entropy of the entire set, while the second term calculates the Entropy after applying the feature A, where P(x) is the probability of event x.

- The initial step is to calculate H(S), the Entropy of the current state.

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

- Next step is to choose the attribute that gives us highest possible Information Gain which we'll choose as the root node.
- Creating ID3 algorithm.ID3 Algorithm will perform following tasks recursively

  1. Create root node for the tree

  2. If all examples are positive, return leaf node 'positive'

  3. Else if all examples are negative, return leaf node

  'negative'

  4. Calculate the entropy of current state H(S)

  5. For each attribute, calculate the entropy with respect to the attribute 'x' denoted by H(S, x)

  6. Select the attribute which has maximum value of IG(S, x)

  7. Remove the attribute that offers highest IG from the set of attributes

  8. Repeat until we run out of all attributes, or the decision tree has all leaf nodes.

III. Random Forest

Ensemble learning is a widely-used and preferred machine learning technique in which multiple individual models, often called base models, are combined to produce an effective optimal prediction model **[11].** Just like decision trees, random forests are a non-parametric ensemble model. The entire random forest algorithm **[12]** is built on top of weak learners (decision trees), giving you the analogy of using trees to make a forest. The term "random" indicates that each decision tree is built with a random subset of data.

The random forest algorithm is based on the **bagging** method.

[13] Bagging, also known as Bootstrap aggregating, is an ensemble learning technique that helps to improve the performance and accuracy of machine learning algorithms. It is used to deal with bias-variance trade-offs and reduces the variance of a prediction model.

Random Forest represents a concept of combining learning models to increase performance (higher accuracy or some other metric).

In a nutshell:

- *N* subsets are made from the original datasets.
- *N* decision trees are build from the subsets
- A prediction is made with every trained tree, and a final prediction is returned as a majority vote

The math behind random forests is the same as with decision trees. You only need to implement two formulas — entropy and information gain.

References

1. **Exoplanets : https://en.wikipedia.org/wiki/Exoplanet**
2. **Methods of Exoplanet Detection: https://en.wikipedia.org/wiki/Methods_of_detecting_exoplanets**
3. **Kepler Telescope: https://en.wikipedia.org/wiki/Kepler_space_telescope**
4. **K-nearest Neighbors: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm**
5. **Multi-class Classification: https://en.wikipedia.org/wiki/Multiclass_classification**
6. **Euclidean distance formula: https://en.wikipedia.org/wiki/Euclidean_distance**
7. **Decision Trees: https://en.wikipedia.org/wiki/Decision_tree_learning**
8. **ID3 Algorithm: https://en.wikipedia.org/wiki/ID3_algorithm**
9. **Entropy: https://en.wikipedia.org/wiki/Entropy_(information_theory)**
10. **Information Gain: https://en.wikipedia.org/wiki/Decision_tree_learning#Information_gain**
11. **Ensemble learning: https://en.wikipedia.org/wiki/Ensemble_learning**
12. **Random Forest Algorithm: https://en.wikipedia.org/wiki/Random_forest**
13. **Bagging (Bootstrap aggregating): https://en.wikipedia.org/wiki/Bootstrap_aggregating**

Model:
All files including data, training and model files are provided on github and google drive.

Google Drive: https://drive.google.com/drive/folders/1jpJxD-x452MG6Hx42pdlgg4wMByLn72l
Github: https://github.com/TanishqSom/Exoplanet-detection