SUBHALO PREDICTION FOR DARK MATTER HALOS

PINNAREE TEA-MANGKORNPAN, DENNIS WANG, MARC WILLIAMSON

Abstract. We explore machine learning techniques for predicting the number of dark matter subhalos, N_{sub}, belonging to a main halo using only those characteristics of the parent halo that are independent of subhalos. Such a model can be used to save computation time by enabling accurate predictions of N_{sub} from lower resolution simulations. Due to a lack of previous work in this area, we thoroughly test multiple types of machine learning models. These include variants of linear regression (LR), a support vector regression (SVR), and a linear regression trees (LRT). We find that locally weighted linear regression with a novel use of principle component analysis performs the best out of all the models, obtaining an average test error of about 7%.

1. Introduction

In the past few decades, cosmology has become an extremely active area of research with landmark surveys such as WFIRST, LSST, EUCLID, and DES. One of the more prominent areas of research in the field is the so called Missing Satellite Problem. The Missing Satellite Problem refers to the difference between the large number of dark matter subhalos that are predicted to belong to main halos, and the much smaller observed number of galaxy satellites[7]. This disagreement indicates either a lack of precision in current observational techniques, or a flaw in physicists understanding of the current models of the universe. In this paper, we investigate the possibility of using machine learning to predict the number of subhalos belonging to a dark matter halo in order to gain insight into the Missing Satellite Problem and to improve computational efficiency in current simulation techniques.

In section 2, we will provide background information on dark matter and the simulations that are used to study dark matter halos. This background will illuminate the connection between subhalos and the Missing Satellite Problem, providing motivation for certain design choices that we made in our machine learning models. Section 3 will briefly describe the dataset that we used for our models. In section 4, we will describe in detail the machine learning algorithms that we used. This will include motivation for each type of model, along with preliminary results in order to obtain a basic idea of the effectiveness of each algorithm. Section 5 will detail two methods of feature selection that we applied to the models in section 4. In section 6, we will present an analysis of weighted linear regression and detail how principle component analysis, (PCA), can be applied to improve the model. Section 7 will be dedicated to fully presenting our results, with a focus on the algorithm that performed the best at predicting N_{sub}. Finally, in section 8, we will analyze our results and discuss applications of the work, along with possible future directions to take this research.

2. Physics Background

Dark matter composes 24% of the energy density of the universe, compared to only 4.6% that is contributed by baryonic matter[8]. In order to understand the universe, it is necessary to understand dark matter. Although it is unknown what exactly dark matter is, physicists can infer some of its properties. Physicists model dark matter as very weakly interacting particles, that only interact gravitationally. In the early universe, the density distribution of dark matter was approximately uniform, with small density perturbations. As the universe evolved, the slightly overdense regions attracted more particles, eventually growing into dark matter halos, which are generally orders of magnitude more massive than galaxies. Dark matter halos thus create large gravitational potential wells which baryonic matter falls into, thus forming galaxies. Each large dark matter halo hosts one galaxy, in addition to having many smaller subhalos that are gravitationally bound to the main halo[1]. These subhalos are host to galaxy satellites, like the Magellanic clouds. This connection between subhalos and galaxy satellites is why studying dark matter subhalos a promising approach to solving the Missing Satellite Problem.

In order to study dark matter halos and subhalos, physicists use n-body simulations to model the entire evolution of the universe[1]. One of the most important properties of these simulations is the resolution. A simulation will be run using a fixed number of particles. Accordingly, each particle must be given a mass in order to accurately simulate the entire universe. If more particles are used, then the individual particle mass is lower, allowing physicists to see smaller structures in their simulations. If a simulation is run with a resolution chosen to study main halos, the obtained N_{sub} might not be representative of the actual number of subhalos, which can be orders of magnitude smaller in mass than the parent halo, because the low resolution might not allow such small structures to form. Therefore, in order to accurately

find N_{sub}, it might be necessary to use a much higher resolution than is required for studying the main halo. Running these high resolution simulations can be computationally expensive and therefore limit the amount of available data. Thus, it is desirable to have a predictive model for N_{sub} that only depends on properties of the parent halo.

3. Dataset

We used the results of the RHAPSODY 8K simulations to train and test our machine learning models[1,2]. We obtained this data courtesy of professor Risa Wechsler at Stanford University. The RHAPSODY simulation contains 96 cluster sized dark matter halos, with virial mass in the range of $M_{vir} = 10^{14.8\pm0.05}h^{-1}M_{\odot}$. Each halo was simulated at two different resolutions: 8192^3 particles, corresponding to a resolution of $1.3 \times 10^8 h^{-1}M_{\odot}$, and 4096^3 particles, corresponding to a resolution of $1.0 \times 10^9 h^{-1}M_{\odot}$[1]. We used the results of the high resolution simulation (8K) so that our models would have the most accurate training data. In the data, we predict the feature $N(v_{max} > 100)$. $v_{max} > 100$ is one criteria that is used for defining a subhalo, and it is the most inclusive such definition in the RHAPSODY simulation. The RHAPSODY simulation keeps track of over 100 halo features, however we had to eliminate about half of these features due to an implicit dependence on knowing the number of subhalos. Using these features in our models would defeat the purpose of creating the model in the first place.

4. Machine Learning Models

We chose to use regression based models instead of discretizing our data and using classification because we wanted to build a model with the lowest possible test error. First, we implement a simple unweighted linear regression in order to get a feel for the data. We then move on to locally weighted linear regression to increase the complexity of our model. For these models, we also explore two different types of feature selection: forward search (a standard algorithm), and tree search (defined later). Finally, we use principle component analysis (PCA) to modify our models. In order to train and test our models, we use the technique of cross-validation: splitting the data into two groups, where 80% of the data is used for training, and the remaining 20% is used to test the model. The training and testing subsets were randomly selected from the 96 halos, and error%ages reflect average values from several trials.

4.1. **Notation.** $\left(x^{(i)}, y^{(i)}\right)$ is a training example, where $i \in \{1, 2, ..., 96\}$, and $x^{(i)}$ is the feature vector while $y^{(i)}$ is the output variable that we are predicting. We use the following matrix definitions...

$$X = \begin{bmatrix} — & \left(x^{(1)}\right)^T & — \\ — & \left(x^{(2)}\right)^T & — \\ & \vdots & \\ — & \left(x^{(m)}\right)^T & — \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

4.2. **Unweighted Linear Regression.** Unweighted linear regression simply fits a straight line to the RHAPSODY data. We used the standard ordinary least squares regression model with the cost function...

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

This model is extremely efficient to implement because it has a normal equation, which optimizes θ in closed form. Unsurprisingly, unweighted linear regression is not a very good model for N_{sub}. It has training error of 6% and a high test error 25% (see the table in section 7). We hypothesize that linear regression underfits the data because it cannot fit nonlinear relationships between the features. We check this hypothesis by using principle component analysis to reduce the entire dataset (without the output feature) to one dimension, and plotting the scatter between N_{sub} and the reduced dimensional data. There is no scatter, which indicates there is no simplistic model for N_{sub}.

4.3. **Locally Weighted Linear Regression.** Locally weighted linear regression (LWLR) combines the simplicity of linear regression with the ability to model nonlinear overall behavior in a data set. This made it our next logical step up complexity from linear regression. Intuitively, given a testing instance, LWR weights the training examples so that training instances "closer" to the testing instance are more important than those farther away. The definition of "closer" is arbitrary. To start, we use standard Euclidean distance in the high dimensional feature space. The weight, $w^{(i)}$ corresponding to training example $x^{(i)}$ is Gaussian...

$$w^{(i)} = \exp\left(-\frac{\left(x - x^{(i)}\right)^2}{2\tau^2}\right)$$

where τ is a parameter that describes the size of the region where highly weighted training instances can exist. Small τ corresponds to only considering very close training examples, while large τ corresponds to a larger range. Thus, smaller τ models nonlinear behavior better. We experimentally determined $\tau = 1.19$ to be the ideal value, and the preliminary results were 3.03% training error and 18.93% test error (see table in section 7). This is a significant improvement over unweighted linear regression in both testing and training error, but the test error is still unacceptably high. At this point, we hypothesize that we are overfitting the data due to the large number of features. This is addressed in section 5.

4.4. **Support Vector Regression.** We use the generalization of the support vector machine for data classification to continuous outputs in order to build a support vector regression model to compare with out linear regression[3]. This model finds a function, f, that has at most ϵ deviation from each training example. This amounts to solving the primal problem...

$$\text{minimize } \frac{1}{2} \parallel w \parallel^2$$

$$\text{subject to } \begin{cases} y^{(i)} - \langle w, x^{(i)} \rangle - b & \leq \epsilon \\ \langle w, x^{(i)} \rangle + b - y^{(i)} & \leq \epsilon \end{cases}$$

We use the Liblinear implementation of SVR, with L2-regularized, L2-loss SVR to solve the dual problem[5]. The results of the SVR model are almost as good as using locally weighted linear regression, but the test error of 11.52% is higher than desired. This paper will focus on using and modifying locally weighted linear regression, so we do not fully analyze the possibility of using SVR. Our further research in this area will involve a much more rigorous treatment of SVR as a possible machine learning model.

5. Feature Selection

We implement two forms of feature selection: standard forward search and a simple method based on linear regression trees. The advantages of feature selection are two-fold. First, it ameliorates overfitting and reduces test error. Second, it helps physicists understand which halo characteristics are most influential in determining the number of subhalos.

5.1. **Forward Search.** The forward search feature selection algorithm works by greedily choosing features that yield models with the best test error performance. The algorithm begins with an empty list of chosen features. It then iterates over all unchosen features, adding each separately to the running list of chosen features and training a corresponding model. The feature that yields the best model (ie lowest test error) is permanently added to the list of chosen features. This algorithm is repeated until the desired number of features has been reached. For our models, we chose to use 15 of the roughly 50 features.

5.2. **Tree Search.** This method of feature selection refers to using a regression tree to select the best predictive features from the data[4]. In order to predict an output for an input vector, the algorithm starts at a root node and follows decision branches until a leaf node is reached. The leaf node contains the prediction for the input vector. At each node in the tree, a feature from the data is selected, and two branches are made leaving the node. The branches have an associated decision rule, for example take the left branch if $x_j < c$, but take the right branch otherwise. The regression tree is constructed from the training data where the outputs are known. There are three characteristics of regression trees that require further explanation: the feature selection, the optimization criterion, and the stopping rule[4]. We utilized the Matlab implementation of regression trees, so we will focus on Matlab's specific implementation.

At each node, a feature needs to be chosen from the data. In order to do this, every binary split on every feature is considered, and the feature with the best optimization criterion is selected[4]. The optimization criterion refers to how the tree chooses a node's associated decision. In Matlab's implementation, the split is chosen in order to minimize the mean squared error of predictions on the training data. The stopping rule refers to the condition that stops a node from branching. For regression, this is when the mean squared error of the node's prediction is less than some tolerance level proportional to the mean squared error for that response in the entire data set. As a regression model, we found that the regression tree did not perform very well, most likely because it bucketized the training data into the nodes of the tree, which requires a large dataset in order to be most accurate. However, the regression tree did identify a set of 15 features as the most predictive of N_{sub}, so we compared this set of features against standard forward searching for 15 features. Using weighted linear regression with forward search we had the following results: test error = 10.60%, training error = 2.26%. Using the tree search, we had test error = 23.24% and training error $\approx$ 0%. Forward search performs much better on the test set of data, so we will use it as our method of feature selection.

5.3. **Results of Feature Selection.** Having identified forward search as the better method of feature selection, we apply it to our locally weighted linear regression model. The results are much improved, with an average of 10.7% test error and 2.1% training error.

6. Improving LWLR with PCA

In locally weighted linear regression, each training instance, $x^{(i)}$, is assigned a weight, $w^{(i)}$, based on the Euclidean distance between the test case, x and $x^{(i)}$. However, this presents a problem because the Euclidean distance scales upwards with the number of features in x and $x^{(i)}$. Specifically...

$$\begin{aligned} d^{(i)} &= \parallel x - x^{(i)} \parallel_2 \\ &= \sqrt{\sum_{j=1}^{k} \left(x_j - x_j^{(i)} \right)^2} \end{aligned}$$

where k is the dimension of the data. This is a significant problem for weighted linear regression because if the distances between points scales upwards, then the tau parameter would need to be scaled upwards in order to keep the model at an optimal fit. During forward search feature selection, features are chosen based on making a LWLR model with the current number of

features, and adding the feature that produces the best model, so k is changing. However, we cannot vary τ during feature selection, since we want to pick a single tau value and then find the best model. To address this problem, we use a modified definition of distance that is independent of the number of features in order to calculate the weights for LWLR. Given a dataset, X, we use PCA to reduce all of the data to a one dimensional representation, $\hat{X}$. Then each $x^{(i)} \in X$ corresponds to $\hat{x}^{(i)} \in \hat{X}$. We split the dataset X into training and testing subsets, and we put the corresponding $\hat{x}$'s in training and testing groups. Then the distance between x and $x^{(i)}$ from X corresponds to the distance between $\hat{x}$ and $\hat{x}^{(i)}$ from $\hat{X}$. We have...

$$\hat{d}^{(i)} = |\hat{x} - \hat{x}^{(i)}|$$

which has the desired behavior of being independent of the number of features in the original dataset, X. The new weights for LWLR are then...

$$\hat{w}^{(i)} = \exp\left(-\frac{\left(\hat{x} - \hat{x}^{(i)}\right)^2}{2\tau^2}\right)$$

The results of this LWLR model using PCA were: test error = 7.73%, and training error = 10.95%. Adding PCA to our model helps performance on the test set, but it causes the training error to increase, compared to the LWLR without PCA model. The increase in training error can be attributed to a loss of information in computing the Euclidean distance from the one dimensional dataset resulting from PCA.

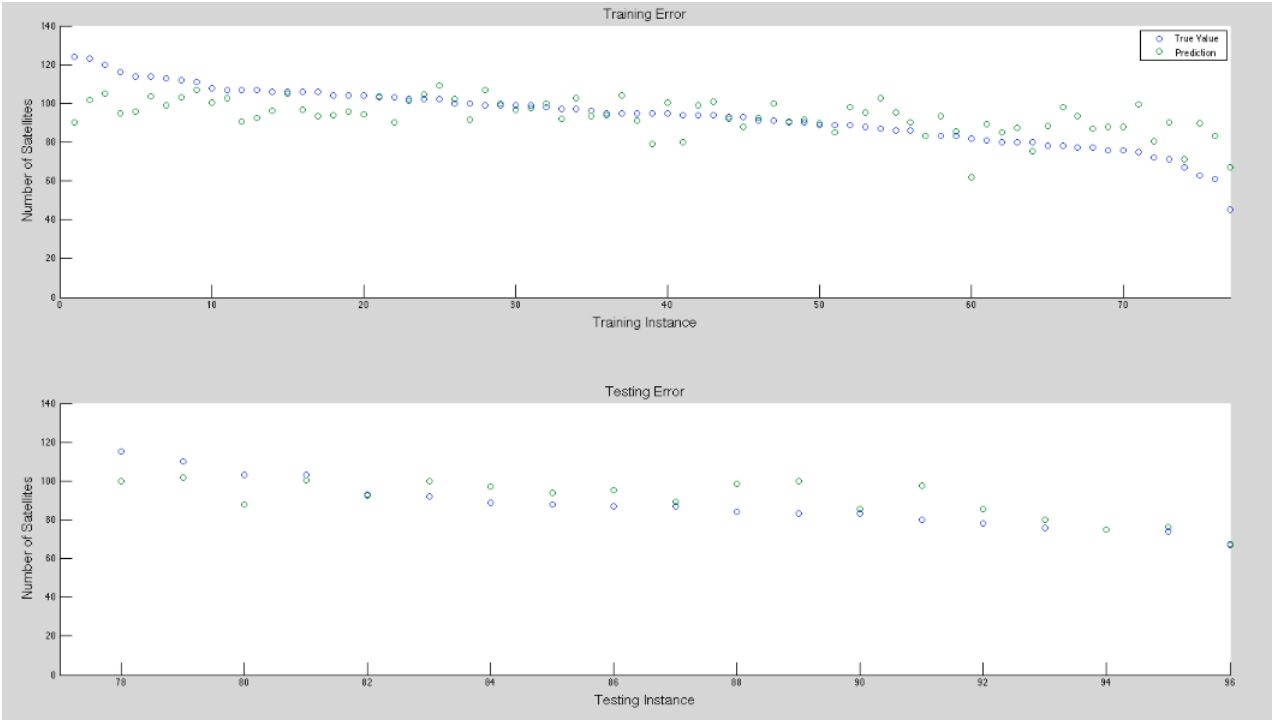
7. Results

The following table organizes the results of each of the machine learning models that we used on the RHAPSODY dataset.

ML Model	PCA	Fwd Search	Test Error %	Train Error %	Tau
Unweighted LR	No	No	25.16	5.27	N/A
Weighted LR	No	No	18.93	3.03	1.19
Weighted LR	Yes	No	18.05	5.24	11.0
Weighted LR	No	Yes	11.0	2.38	0.30
Weighted LR	Yes	Yes	7.84	10.86	12.0
SVR	No	Yes	11.52	11.91	N/A
Regression Tree	No	No	20.05	4.59	N/A

8. Analysis

The above table represents a roughly chronological ordering of the machine learning algorithms that we used. The results involving the various types of linear regression match our expectations: specifically that Unweighted LR has the worst performance due to its simplicity, and the various algorithmic improvements that we implemented, (feature selection and PCA distance) improve the test error compared to the basic locally weighted linear regression. Surprisingly, support vector regression does not perform better than LWLR with PCA and feature selection, although the focus of this project is mainly modifying LWLR, so it is possible that further work with SVR models could yield better results. One important feature of the results is that modifying the distance in LWLR with PCA causes training error to increase from the LWLR method with only feature selection. The explanation for this increase is discussed in section 6. Despite this increase in training error, we believe that LWLR with PCA and feature selection is a better model for predicting N_{sub} because it has lower test error, and it represents a more reasonable trade-off between training and test error.



The above plot shows the results for a sample run of LWLR with PCA and feature selection. One interesting question is whether or not the model is biased. This would manifest itself in a scatter plot of the error versus the true number of subhalos. For example, if the model consistently underestimates N_{sub} when N_{sub} is higher, then the error is biased. This would indicate that there is behavior that our model is failing to accurately predict. This analysis is one course of action that we will take in pursuing our research.

The features that were identified as the best predictors of the number of subhalos, listed best to worst, are: Rvir, vpeak, BCGdom1st, vdfv0, sigmav, alpha2step, log10Mvir, vdfp, pvalign(dm), pvalign(vmax), b/a, pvalign(vpeak), gammanfw-like, c/a, and b/a (v). Some of these features are expected to be good predictors. For example, Rvir is a measure of the size of a halo, and log10Mvir measures its mass. It is unsurprising that the size of a halo is helpful in determining how many subhalos become gravitationally bound to it. However, we do not know exactly how these chosen features interact, nor do we fully understand all of them. One focus of future our future research will be pursuing a better physical understanding of the features and subhalos.

9. Acknowledgements

We would like to thank Professor Risa Wechsler for her guidance in helping us choose this topic of research. We would also like to thank her graduate student, Yao Mao, for his help in explaining the more difficult physics concepts. Finally, we warmly thank Professor Andrew Ng for providing us with the machine learning skills necessary to conduct this research.

10. References

[1] H.Y. Wu et al., "RHAPSODY: I. STRUCTURAL PROPERTIES AND FORMATION HISTORY FROM A STATISTICAL SAMPLE OF RE-SIMULATED CLUSTER-SIZE HALOS," Kavli Institute for Particle Astrophysics and Cosmology, Stanford, CA, Sept. 2012.

[2] H.Y. Wu et al., "RHAPSODY. II. SUBHALO PROPERTIES AND THE IMPACT OF TIDAL STRIPPING FROM A STATISTICAL SAMPLE OF CLUSTER-SIZE HALOS," Kavli Institute for Particle Astrophysics and Cosmology, Stanford, CA, Mar. 2013.

[3] A.J. Smola and B. Sch ¨olkopf, "A Tutorial on Support Vector Regression," Sept. 2003.

[4] MathWorks, Classification Trees and Regression Trees [Online].
Available: http://www.mathworks.com/help/stats/classification-trees-and-regression-trees.html#bsxg_kc.

[5] C.J. Lin, LIBLINEAR [Online]. Available: http://www.csie.ntu.edu.tw/˜cjlin/liblinear/

[6] X. Xu et al., "A FIRST LOOK AT CREATING MOCK CATALOGS WITH MACHINE LEARNING TECHNIQUES," Mar. 2013.

[7] J. Bullock, "Notes on the Missing Satellites Problem," UC Irvine, Sept. 2010.

[8] B. Griswold, What is the Universe Made of? [Online]. Available: http://map.gsfc.nasa.gov/universe/uni_matter.html.