Enhanced Cobweb Clustering for Identifying Analog Galaxies in Astrophysics

Ashwin Satyanarayana
N-913, Dept. of Computer Systems Technology
300 Jay Street,
Brooklyn, New York - 11201
asatyanarayana@citytech.cuny.edu

Viviana Acquaviva
N-828, Dept. of Physics
300 Jay Street,
Brooklyn, New York - 11201
vacquaviva@citytech.cuny.edu

***Abstract*—Clustering, a very popular task in Data Mining, is unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). Clustering has been explored in many different contexts and disciplines. In this paper, we explore using the COBWEB clustering algorithm to identify and group together galaxies whose spectral energy distribution (SED) is similar. We show that using COBWEB drastically reduces CPU time, compared to a systematic one-by-one comparison previously used in astrophysics. We then extend this approach by using COBWEB clustering with Bootstrap Averaging and show that using Bootstrap Averaging produces a more accurate model in roughly the same amount of time as COBWEB.**

I. SED FITTING OF LARGE GALAXY CATALOGS

Understanding the physical properties of galaxies, such as mass, star formation history, age, and dust content, is a key endeavor in Astrophysics. From the very first galaxies whose light comes from over 13 billion years ago to today's rich landscape of shapes, star formation histories and compositions, galaxies help us trace 95% of the universe's evolution. The Spectral Energy Distribution (SED) of a galaxy is a chart of the galaxy's luminosity as a function of wavelength, and it contains information about the physical properties of the galaxy, such as mass, age of the stellar population, star formation history, and dust content. The process of extracting information from the SED (known as *SED fitting*, see e.g., Acquaviva et al [1][2]) is a key step in understanding galaxy formation and evolution. Large ongoing and planned galaxy surveys offer us the opportunity of recovering these properties for an unprecedented number of galaxies (of the order of billions for the Large Synoptic Survey Telescope, or LSST [3]), but this chance comes with a hefty tag in terms of CPU time if every galaxy has to be analyzed individually. However, for each of the surveys, the information contained in the spectrum is compressed and "binned" in a handful of data points. Therefore, in large data sets there will be many galaxies whose SED have the same shape within the observational errors. If these galaxies can be grouped together *before* performing SED fitting, this affords a factor of $\sim N$ improvement in CPU time, where N is the average number of galaxies in each group.

We demonstrate that groups of similar galaxies or "data clusters" exist even in relatively small samples of galaxies by analyzing a sample of 5228 galaxies in the CANDELS GOODS-S catalog [4]. These data were taken by the Hubble Space Telescope in a three-year campaign between 2010 and 2013, and represent one of the best quality, highest signal-to-noise galaxy catalog available today. The dataset has 9 features. The only method mentioned in Astrophysics literature so far for identifying analog galaxies [5] is what we call a "grid search algorithm". In this method, the SED of every galaxy is compared to the SED of every other galaxy in order to find which galaxy has the most "analogs". This group of galaxies is then removed from the list and the process is repeated in order to find the next largest group of analogs, until no more analogs are found. We applied this method to the catalog described above, and found that:

- The average number of analogs for each galaxy is 13.5, i.e., many galaxies have analogs;
- Even the "first pass" of the classification tool about 30 minutes on a 2.2Ghz MacBook Pro computer, and a conservative estimate for completing the classification using this $O(N^2)$ approach is about 20-50 hours.

These results imply that while finding groups of analog galaxies is a promising avenue for reducing the CPU time required by SED fitting, it is necessary to explore different numerical tools in order to speed up the clustering process. **The goal of this paper is to use, for the first time, data mining tools to identify clusters of galaxies efficiently and rapidly.**

The format of the paper is as follows: In the next section we briefly present clustering algorithms and present the COBWEB algorithm. We then discuss our enhanced approach where we use Bootstrap Averaging on COBWEB. We finally show how this algorithm is able to reduce the CPU time used for the classification without sacrificing accuracy.

II. CLUSTERING AND THE COBWEB ALGORITHM

Clustering algorithms have been used in a variety of applications such as cancer research [6], search engines [7], academics [8] and wireless sensor networks [9]. In this paper we focus on using conceptual clustering in the field of astrophysics. Conceptual clustering is normally used in order to discover classes of objects with common characteristics in

large amounts of data. A system employed in this task receives as input a set of observations and outputs a set of classes in which the input is distributed. The observations are described by a predefined set of attributes that take a value from a given set of values. The attributes are chosen such that to represent the observation's characteristics and assist in this way in forming a meaningful clustering scheme. Conceptual clustering systems evaluate these properties using an appropriate objective function and attempt to improve the quality of the clustering by employing a control strategy. In particular, we choose hierarchical conceptual clustering for finding analog galaxies as we do not have to assume any particular number of clusters beforehand. The core idea in hierarchical conceptual clustering, also known as cconnectivity based clustering, is that objects are more related to those near them. These algorithms connect "objects" to form "clusters" based on their distance.

A. Cobweb Algorithm

The COBWEB algorithm [10], an incremental conceptual hierarchical clustering technique, was developed by machine learning researchers in the 1980s for clustering objects in a object-attribute data set. The COBWEB algorithm yields a clustering dendrogram called classification tree that characterizes each cluster with a *probabilistic description*. The COBWEB algorithm constructs a classification tree incrementally by inserting the objects into the classification tree one by one. When inserting an object into the classification tree, the COBWEB algorithm traverses the tree top-down starting from the root node. At each node, the COBWEB algorithm considers 4 possible operations (insert, create, merge, split) and selects one that yields the highest *category utility* (*CU*) function. *CU* attempts to maximize both the probability that two instances in the same category have attribute values in common and the probability that instances from different categories have different attribute values:

$$CU = \sum_C \sum_A \sum_v P(A = v)P(A = v|C)P(C|A = v) \quad (1)$$

$P(A = v|C)$ is the probability that an instance has value v for its attribute A, given that it belongs to category C. The higher this probability, the more likely two instances in a category share the same attribute values. $P(C|A=v)$ is the probability that an instance belongs to category C, given that it has value v for its attribute A. The greater this probability, the less likely instances from different categories will have attribute values in common. $P(A=v)$ is a weight, assuring that frequently occurring attribute values will have stronger influence on the evaluation.

After applying Bayes rule to (1) we get:

$$CU = \sum_C \sum_A \sum_v P(C)P(A = v|C)^2 \quad (2)$$

$\sum_A \sum_v P(A = v|C)^2$ is the expected number of attribute values that one can correctly guess for an arbitrary member of class C. This expectation assumes that a probability matching strategy, in which one guesses an attribute value with a probability equal to its probability of occurring. Without knowing the cluster structure the above term is $\sum_A \sum_v P(A = v)^2$

The final *CU* is defined as the increase in the expected number of attribute values that can be correctly guessed, given a set of n categories, over the expected number of correct guesses without such knowledge. That is:

$$CU = \frac{1}{n}\sum_C P(C) \sum_A \sum_v [P(A = v|C)^2 - P(A = v)^2] \quad (3)$$

The above expression is divided by n to allow comparing different sized clusters. The steps of the COBWEB algorithm are shown in Fig. 1.

```
1  Function Cobweb (object, root)
2    Incorporate object into the root cluster;
3      If root is a leaf then
4        return expanded leaf with the object;
5      else choose the operator that results in the
       best clustering:
6        a) Incorporate the object into the best host;
7        b) Create a new class containing the object;
8        c) Merge the two best hosts;
9        d) Split the best host;
10     If (a) or (c) or (d) then
11       call Cobweb (observation, best host);
```

Fig 1. COBWEB Algorithm

We now discuss some of pros and cons of using COBWEB algorithm for our application of identifying analogs of galaxies.

Pros:

a. *Better Time Complexity:* COBWEB has the time complexity of $O(AVB^2 logK)$, where B is branching factor, A (attributes), V (average number of values), K (classes). Empirically B is chosen between 2 and 5. In comparison, the Grid Search method uses $O(N^2)$ time complexity.

b. *Number of clusters not known beforehand:* In our specific application of identifying groups of analog galaxies, we do not know beforehand the number of clusters needed, which is ideal for Hierarchical clustering techniques like COBWEB.

Cons:

a. *Does not handle noisy data:* COBWEB algorithm does not deal well with noisy data as hierarchical clustering algorithms find it difficult to detect outliers.

b. *Tends to make "bushy" trees*: The higher levels of the tree end up being the most important class categories (because of merge/split causing best breaks to float up).

In order to overcome these cons, we propose using bootstrap averaging, which will eliminate noisy data and create more accurate models. This will be shown in the empirical section of this paper.

B. Cobweb Algorithm with Bootstrap Averaging

Bootstrap Averaging [11] is an ensemble technique that produces replicates of the training dataset by sampling with replacement from the original dataset. This creates bootstrap samples of equal size to the original dataset. Then a model is built on each replicate. Together these models form an ensemble model. The cluster centroids from each of the models are averaged. This approach reduces the variance and also eliminates noisy instances as will be shown later in this paper.

Our prior work [11] showed that Bootstrap (sampling with replacement) Averaging works well with *k*-means clustering algorithm. In our current work, we show that Bootstrap algorithm works not only with partitioning algorithms (i.e. *k*-means) but also with hierarchical algorithms (like COBWEB). Our approach basically consists of 3 steps: (a) sub-sampling the training data (b) cluster each sub-sample using COBWEB and (c) cluster the resulting cluster centers to generate a refined final model.

Our approach builds multiple models by creating small bootstrap samples of the training set and building a model from each, but rather than aggregating like bagging [12], we average similar cluster centers to produce a single model. In this paper we shall focus on bootstrap samples that are smaller than the training data size. This produces results that are comparable with multiple random restarting of COBWEB clustering using all of the training data, but takes far less computation time. For example, when we take *T* bootstrap samples of size 25% of the training data set then the technique takes at least four times less computation time but yields as good as results if we had randomly restarted COBWEB *T* times using all of the training data. The Bootstrap Averaging algorithm is shown in Fig 2.

Input: D: Training Data, T: Number of bags

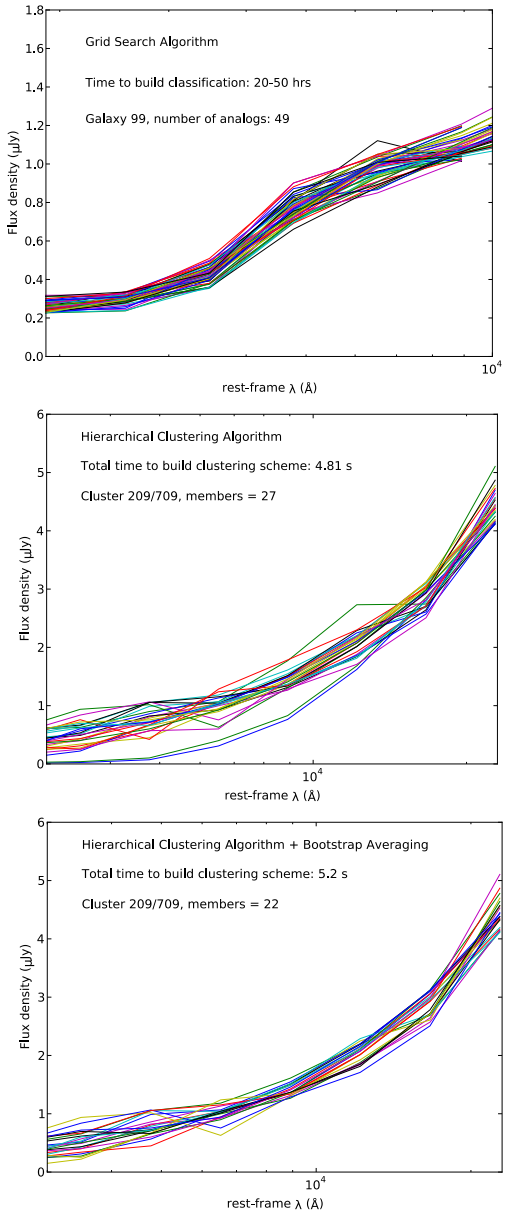
Output: A: Averaged Centroids

```
CobwebWithBootstrapping
    For i = 1 to T
        Xi = Bootstrap (D)
        Ci = COBWEB(N, Xi)
        // Note Ci is the set of k cluster centroids and
        // Ci = {ci1, ci2, .... cik}
    End For
    //Group similar clusters into Bins
    // with the Bin averages stored in B1 ... Bk
    // with sizes S1.... Sk
    For i = 1 to T
        For j = 1 to K
        Index = AssignToBin (cij)
        Bindex += cij
        EndFor
    EndFor
    For i = 1 to K
        Bi / = Sk
        Ai = Bi
    EndFor
```

Fig 2. COBWEB with Bootstrap Averaging

III. RESULTS

We tested the performance of the Hierarchical Clustering (HC) algorithm compared to the Grid Search (GS) method on an out-of-the-box distribution of the COBWEB algorithm (Fisher 1987). The algorithm took about 5 seconds to complete the classification scheme, dividing the sample in 742 clusters. Both algorithms (GS and HC) performed similarly well in identifying analog galaxies, but *the HC algorithm is about 20,000 times faster*. This gap will increase rapidly for bigger samples, since the CPU time required by the HC algorithm scales with sample size much slower than N^2 (between *O(N)* and *O(N log N);* the latter is the "worst case" in which a sample twice as large requires twice as many fundamental shapes).



Figs 3-5: Example of clusters found by the grid-seacrh algorithm (top), COBWEB (middle), and COBWEB with Bootstrap Averaging (bottom). In each of these plots, the x axis shows the wavelength of the observations in Angstroms, while the y axis shows the brightness of the galaxy at each wavelength.

Our first improvement over the public version of COBWEB was to minimize the number of noisy instances (outliers) by using Bootstrap Averaging. This improvement eliminates noisy instances and allows one to build a more powerful classification, with 505 final clusters and ~ 10

members per cluster, at only a 10% cost in terms of CPU time. The performance of the three algorithms is summarized in the table, and two example clusters for grid search and HC (before and after Bootstrap Averaging) are shown in Figs. 3-5. Each figure shows the SEDs of galaxies found to belong to the same cluster. Our results imply that the time required for SED fitting time of these 5228 galaxies could be reduced by a factor 5228/505 ~ 10 by using this powerful classification algorithm. *In a survey like LSST, with billions of SEDs with only 6 features, the number of analog galaxies in each group will be much larger and so will be the improvement in CPU time.*

Algorithm	CPU Time	Number of clusters	Number of analog galaxies per Cluster
Grid Search	20-50 hours	387	13.5
Cobweb	4.87 seconds	742	7.04
Cobweb with Bootstrap averaging	5.24 seconds	505	10.35

Table 1: Comparing 3 algorithms for Clustering galaxies

CONCLUSION

In this paper we have shown that using data mining tasks such as clustering can be very effective in the field of astrophysics, where there is a big data revolution. We came to the empirical conclusion that hierarchical clustering is a suitable tool for identifying groups of similar galaxies quickly and efficiently. However, we found that noisy instances still existed in the clusters. We then resolved this problem by using bootstrap averaging with hierarchical clustering to fix this problem. Further steps would be to inverse-noise weigh the different features to insure that the experimental uncertainties are correctly taken into account in the classification process.

REFERENCES

[1] V. Acquaviva, E. Gawiser, and L. Guaita, “SED fitting with Markov Chain Monte Carlo: methodology and application to LAE galaxies”, Astrophys. J. 737: 37 (2011).

[2] V. Acquaviva, E. Gawiser ,and L. Guaita, “SED fitting: methodology and application to large galaxy surveys” ,Proceedings of the IAU symposium ”The SED of galaxies”, Preston, UK (2011).

[3] Z. Ivezic et al, “LSST: from Science Drivers to Reference Design and Anticipated Data Products”, arXiv:0805.2366 (200*).

[4] Y. Guo et al, “CANDELS Multi-wavelength Catalogs: Source Detection and Photometry in the GOODS-South Field” Astrophys. J., Suppl. Ser., 207, 24 (2013).

[5] Kriek, M., van Dokkum, P. G., Whitaker, K. E., Labbe, I., Franx, M., & Brammer, G. B. 2011, Astrophys. J., 743, 168.

[6] Wang, X. Y., and J. M. Garibaldi. "A comparison of fuzzy and non-fuzzy clustering techniques in cancer diagnosis." *Proceedings of second international conference in Computational Intelligence in Medicine and Healthcare*. 2005.

[7] Liu, Ting, Charles Rosenberg, and Henry A. Rowley. "Clustering billions of images with large scale nearest neighbor search." *Applications of Computer Vision, 2007. WACV'07. IEEE Workshop on*. IEEE, 2007.

[8] Oyelade, O. J., O. O. Oladipupo, and I. C. Obagbuwa. "Application of k means clustering algorithm for prediction of students academic performance." *arXiv preprint arXiv:1002.2425* (2010).

[9] Akkaya, Kemal, Fatih Senel, and Brian McLaughlan. "Clustering of wireless sensor and actor networks based on sensor distribution and connectivity." *Journal of Parallel and Distributed Computing* 69.6 (2009): 573-587.

[10] Fisher, Douglas H. (July 1987). "Improving inference through conceptual clustering". *Proceedings of the 1987 AAAI Conferences*. AAAI Conference. Seattle Washington. pp. 461–465.

[11] I Davidson, A Satyanarayana: Speeding up k-means clustering by Bootstrap Averaging. IEEE Data Mining Workshop on Clustering Large Data Sets, Third IEEE International Conference on Data Mining

[12] L. Breiman. Bagging predictors. Machine Learning, 26(2):123-140, 1996.