**Generating SVM Tree for unsupervised data**

Dr. Naveen Nekuri, SCIS, UoH

Amrit Majumder, SCIS, UoH

**Abstract:** A clustering technique usually takes all the features of an observation into consideration when grouping the data points. This approach involves certain features which contribute significantly to the grouping of the data points and certain features that contribute comparatively less and even none. Since, clustering the data points requires analysing their respective distances to assume similarity and dissimilarity between observations, the later kind of the features (mentioned above) only contribute to the complexity of the algorithm. Other significant problems involve the number of clusters that are supposed to form in a data set. Many of the popular techniques, directly or indirectly, assume the number of clusters that can form. Therefore becoming a metric to be analysed later, a hyper-parameter. Considering these drawbacks, the aim is to build a clustering algorithm using SVM on a decision tree, based on hierarchical clustering model. This model performs the divisive form of clustering, based on the more significant features, hence using decision trees to achieve that. To achieve a more optimal result, SVM is used for branching.

**KEYWORDS:** Unsupervised Learning, Clustering

1. **INTRODUCTION**

As the name implies, SVM (Burges, C. J. 1998; Cortes, C., & Vapnik, V. 1995) is used upon a `tree' structure, which is basically a decision tree (Brijain et al. 2014). Both primarily belong to the supervised realm of learning. But the task of this project is clustering: an unsupervised design of learning mechanism. At first they seem completely unrelated, but as one can observe, both SVM (Pradhan, A. 2012) and decision trees can be assigned for classification, and both classification and clustering belong to the category of pattern recognition. This gives a way to rethink those supervised models in the realm of unsupervised learning.

Available clustering algorithms (Irani et al. 2016; Pham, D. T., & Afify, A. A. 2007; Xu, R., & Wunsch, D. 2005), of the machine learning realm require certain hyper-parameters that introduce limitations. That is the number of clusters. Setting the number of clusters explicitly means that the model itself is not capable of identifying what constitutes a cluster. For a clustering algorithm, this seems to be a major setback. Though researchers were able to come up with good mechanisms even with requiring this hyper-parameter, ideally the model should be able to automatically do that.

Now that doesn't mean there aren't any algorithms which cannot identify clusters by themselves. Density based Clustering method's most popular algorithm: DBSCAN has been equipped to do just that. Extensive research is also done with this objective (Cheung et al. 2013; Ram, P., & Gray, A. G. 2011, August; Ester et al. 1996, August). Yet none of them poses the capability to be interpretable. A model being interpretable means that it should be able to say which feature contributes to the clustering order. Just like in supervised learning where the linear model's coefficient values represent how much the attribute contributes to distinguishing between classes, or in a decision tree where the root node's attribute has the highest contribution, the ones below it have comparatively lower contribution and so on.

Thus, taking into consideration the specific setbacks, a direct outcome can be concluded upon the objective for building this model. The model thus built, is aimed to be capable of:

* Identifying the number of clusters from within the dataset,
* Being able to interpret feature contributions.

1. **RELATED WORKS**

**2.1 Interpretable Hierarchical Clustering using Unsupervised Decision Tree**

A form of hierarchical clustering technique proposing (Basak & Krishnapuram, 2005) to perform clustering using a decision tree of unsupervised form. The approach follows the traditional idea of shrinking entropy among attributes, but in this case, the entropy refers to the distribution of the data points in the space. Here this particular approach is called the inhomogeneity measure. The authors propose four ways to compute these measures, all of which require a certain similarity value to be computed using the pairwise distance between the data points. The inhomogeneity measure is computed for each attribute and the one with the maximum measure is used for the splitting of the dataset.

Given the attribute, it assumes that the value distribution follows a mixture of gaussian curves, with each curve representing a cluster. Thus the method aims to separate each of these curves using the valley points as split points. The dataset is split into subsets, each constituting a cluster and therefore building the tree.

**2.1 SVM based binary tree**

Another approach to hierarchical clustering, a SVM based binary tree (Elaidi et al. 2018, April), involves continuously splitting the dataset into two groups using a particular proposed technique and SVM, thereby building a binary tree with leaves being sample subsets representing clusters of data points.

The steps involve first using a proposed technique to split the dataset into two groups. The approach is named open window technique. It takes the two most distant points, and uses them as anchors for axes. The dataset is then split into two groups using nearest neighbour technique. Then using the two axes, the data points of the corresponding group are reflected onto the other side of the axis, as if opening the two flaps of a window. Now with the data points separated, the axes are used as margins to determine the boundary, just as in SVM.

This process is continually repeated until each cluster contains a single datapoint or some threshold is met. Thus building a binary tree representing a structure similar to the dendogram of hierarchical clustering models.

**2.3 Others**

Similarly there are many more approaches to hierarchical clustering explored. One approach is on a way of using a concept in Gaussian Mixture model to perform hierarchical clustering (Li, J., & Nehorai, A. 2018). The approach borrows the idea that the distribution of data would resemble a mixture of Gaussian curves, with each curve being assumed to be a cluster. So separating each of these individual curves would in turn separate the clusters. Therefore, the whole process requires dividing the given set of data into subsets from the valley of the data distribution at each step. Similar to the tree building approach, another method aims at building interpretable clustering using an unsupervised binary tree (Fraiman et al. 2013). The procedure includes splitting the dataset into subsets to reduce the heterogeneity. The whole process is described to be a form of CART algorithm with some difference in the pruning process. Another approach involves a novel GH-EXIN neural network being used to find the split points (Cirrincione et al. 2020) to cluster the dataset at each recursive step. Applications of such methods include the automatic segmentation in MRI (Vupputuri et al. 2020), and assessing security issues in web application (Thirumaran et al. 2019, March). These applications use the unsupervised decision tree approach discussed earlier.

A survey on clustering algorithms explores the different approaches in this domain and the popular variants of each approach (Xu, R., & Wunsch, D., 2005), on various similarity measures being used to define clusters (Irani et. al. 2016) and on various applications of the various clustering techniques (Pham et. al. 2007). Among the popular machine learning techniques, support vector machines and decision trees, rank among the top approaches in use. A survey on support vector machines was conducted regarding the quality of service and its applications (Pradhan, A. 2012). For the decision tree model, a survey on various learning algorithms (Brijain et. al. 2014).

Using SVM based decision trees too are explored in supervised versions. One includes building decision trees using SVM at each step to choose the most important attribute and split the given dataset if it contains more than one label (Zhang, D., & Jin, X. B. 2011, July). Using almost similar approach as such for performing nonlinear classification (Nie et al. 2019), and multi class classifications (Madzarov, G., & Gjorgjevikj, D. 2009, May). Other methods involve forming an ensemble of decision trees using twin bounded SVM (Ganaie et al. 2020), building a multilevel SVM (Sakr et al. 2013), proposing data selection methods (Lopez-Chau et al. 2012, November), hierarchical SVM classification (Hao et al. 2007), and so on. On the application side, works like speech emotion detection (Sun et al. 2019), text classifier (Xu et al. 2012), and many such become handy with such approaches.

1. **PROPOSED MODEL**

As true for any decision tree model, here too the tree formation involves first determining an attribute and then using that feature to cluster the data points (in this case). The attribute will be chosen on the basis of certain measurement that demonstrates how well the feature represents the data. After verifying upon the attribute, it is employed as an internal node in the tree from which further branching (i.e. clustering) is decided upon. The branching or dataset splitting will be done using a support vector machine. Being a supervised model, the feature values would first have to be roughly separated and labelled to convert them to supervised form, upon then SVM will be used to find the optimum split point.

**3.1 Attribute Selection**

As discussed in section 3.1, the authors (Basak, J., & Krishnapuram, R. 2005) propose four different ways to choose an attribute. One of them measure the influence of a feature on the basis of the entire dataset and thus provides a well basis for this process:

where, , is the distance between points i and j, with being the two furthest points in the set and being a monotonically increasing function. represents the entropy measure of a feature with respect to the entire dataset.

The function can be any distance function, and must be computed for all pairs possible in the given set. Among them the maximum is identified: , using which all the distance measures are normalised, representing a dissimilarity measure. Two points are most dissimilar if they are furthest apart, i.e. , and the least dissimilar if the measure is zero. But clustering among data points are determined using the similarity among, and thus (similarity measure) is computed. The same is computed after removing a feature from the set, . The difference between the two similarity measure provides the information of how well feature contributes to the similarity between points . With all these components at hand, the entropy is computed, . The feature for which the entropy is minimum is then selected as the one.

**3.2 Dataset split**

As discussed previously, in order for SVM to work, the set needs to be temporarily converted to supervised form. The process first needs an assumption to be made: the feature value distribution resembles a mixture of Gaussian curves. Each curve is a cluster and thus separating each such curve will be the aim of this entire process.

The individual curve can be identified using the valley regions. That valley point could be considered as the split point for separating the two curves, but that is dependent on the height and spread of the two adjacent curves being the same. If the two parameters are not similar, the optimal split point would shift from the detected valley point, but it would still be near to it.

Therefore the detected valley point is taken as a rough estimation of a split point and using it the curves are temporarily labelled: and . Now being a supervised set, the SVM can be applied to find the optimal split point. Note that hard margin SVM cannot be used in this case, since it would simply end up selecting the valley point itself. So a soft margin version is used to enable the flexibility to declare some data points as slack, and choose a more appropriate optimal boundary point. This boundary point will be used as the required split point.

This process is repeated for all the valley points detected, thus acquiring the necessary information to branch out. The parent node from which the branching is carried out will contain the chosen attribute.

**Algorithm 1:** Tree growth grow()

/\* A recursive algorithm grow(Node, depth) \*/

**Input:** Node, depth

// Node contain a dataset or a subset from previous iteration

// Node is the current node of the tree being built

sets ← Node;

**if** *depth < max\_depth* **then**

**for** *each s ∈ sets* **do**

attribute, subset ← get\_split(s)

**if** subset ≥ 1 **then**

// Node is assigned to the attribute

Node(attribute) ← subset;

// recursing to the next level

grow (Node(attribute), depth+1);

**end**

**else**

Node(attribute) ← subset

**end**

**end**

**end**

The Algorithm 1: Tree growth, shows the SVM Tree building algorithm. Note that after an attribute has been chosen, it is removed from the dataset when branching. The algorithm for the dataset split is shown in Algorithm 2: Dataset Splitting.

**Algorithm 2:** Dataset splitting

**Input:** sample\_set

**Output:** Node, attribute

/\* Node contains the whole dataset or a subset from previous iteration \*/

// if sample\_set has more than one attribute

**if** *sample\_set.attributes > 1* **then**

// select the most significant attribute

attribute ← entropy\_measure(sample\_set);

// remove the selected attribute from the sample\_set

sample\_set.attribute ₋ attribute

**end**

**else**

// take the only attribute available

attribute ← sample\_set.attribute

**end**

split\_points ← svm\_splits(attribute);

Node ← subsets of the sample\_set divided using the split\_points

The \_returns the attribute which gave the minimum entropy result. The \_ function returns all the optimal split points found during the process.

**3.3 Model Complexity**

The computationally heaviest task is the attribute selection, with the entropy measure taking up where k is the number of attributes, n is the number of samples and c is the number of clusters formed. The SVM takes up about where s is the number of split points and the tree building process having since the tree doesn’t take balancing into consideration.

1. **RESULTS**

The model is tested on the following datasets:

* Iris
* Wine
* Wisconsin’s Breast Cancer
* Balance Scale
* Liver Disorder
* Glass Identification
* Haberman’s survival
* Thyroid
* Page Blocks

The details to these datasets are below:

| DATASETS | Samples | Features | Class |
| --- | --- | --- | --- |
| Iris | 150 | 4 | 3 |
| Wine | 178 | 13 | 3 |
| Wisconsin’s  Breast Cancer | 569 | 30 | 2 |
| Balance Scale | 625 | 4 | 3 |
| Liver Disorder | 345 | 5 | 2 |
| Glass Identification | 214 | 10 | 7 |
| Haberman’s  Survival | 306 | 3 | 2 |
| Thyroid | 215 | 5 | 3 |
| Page Blocks | 5473 | 10 | 5 |

**4.1 Evaluation Procedure**

All the datasets used here are supervised, therefore the labels can prove effective in comparing against the quality of the clusters.The procedure involves first taking the true labels from the original dataset. For each particular cluster, take out all the data points in it and draw out their respective true labels from the list taken previously. From this sub-list of true labels of the particular cluster considered, determine the mode of these true labels. That is, find out the most frequently occuring label in that sub list. Once the most frequent true label is found, use that label as the predicted label for all the data points in that cluster. Or, in other words, label a cluster with the most frequently occurring true label in it's list of datapoints. Now, as a result, this procedure would label multiple clusters to the same class. That is, within a single class, there may exist more than one cluster of data points.

**4.2 Result Comparison**

The results obtained from SVM Tree is compared along with the result of a reference model; the Unsupervised Decision Tree which has already been compared with other clustering algorithms and shown to perform at their level, even better at some instances. The implementation used cross validation for three different folds to test how different amounts of training samples affect the overall performance.

Table 2: Comparison between SVM Tree and Unsupervised Decision Tree

|  |  | SVM Tree | | UDT | |
| --- | --- | --- | --- | --- | --- |
| Dataset | Folds | Train | Test | Train | Test |
| Iris | 5 | 74.67 | 100 | 77.33 | 96.67 |
| 10 | 84.67 | 100 | 78.67 | 100 |
| 20 | 90.67 | 100 | 81.33 | 100 |
| Wine | 5 | 75.84 | 80.55 | 57.86 | 72.22 |
| 10 | 82.58 | 100 | 72.47 | 100 |
| 20 | 88.20 | 100 | 73.03 | 100 |
| Breast Cancer | 5 | 67.83 | 61.40 | 62.21 | 82.30 |
| 10 | 72.58 | 92.98 | 81.90 | 89.47 |
| 20 | 77.15 | 96.42 | 85.94 | 93.10 |
| Haberman’s  Survival | 5 | 58.16 | 77.04 | 64.70 | 67.21 |
| 10 | 65.35 | 80.64 | 67.32 | 80.65 |
| 20 | 68.62 | 93.75 | 71.24 | 93.75 |
| Thyroid | 5 | 74.88 | 57.16 | 70.23 | 51.16 |
| 10 | 85.11 | 57.14 | 71.62 | 57.14 |
| 20 | 88.37 | 100 | 83.25 | 100 |
| Page  Blocks | 5 | 72.10 | 92.70 | 71.80 | 90.05 |
| 10 | 81.43 | 92.87 | 80.92 | 90.31 |
| 20 | 80.09 | 93.77 | 85.43 | 90.77 |
| Balance Scale | 5 | 62.08 | 76.80 | 47.84 | 78.40 |
| 10 | 71.84 | 77.41 | 72.80 | 75.80 |
| 20 | 75.20 | 87.09 | 68.16 | 87.09 |
| Liver Disorder | 5 | 51.30 | 56.52 | 58.55 | 49.27 |
| 10 | 50.14 | 82.85 | 51.30 | 82.85 |
| 20 | 73.03 | 100 | 68.40 | 82.35 |
| Glass  Identification | 5 | 28.50 | 38.63 | 28.97 | 37.50 |
| 10 | 32.24 | 38.89 | 32.24 | 38.89 |
| 20 | 35.51 | 40 | 35.51 | 40 |

The comparisons show some difference in performance between the two models, though SVM Tree performs better in most cases. The result table for the rest of the datasets are shown following this. There too, both the model’s performance is shown to be close to each other, though SVM Tree out performs, even if by a slight difference.



Figure 1: SVM Tree built for iris

The resulting tree formed shows the different attributes forming the internal nodes and on that basis, the branching being carried out. The above Figure 1, shows the tree built upon the Iris dataset. The image depicts seven clusters formed using two attributes. The resulting model will specify the split points used in each attribute.

1. **CONCLUSION**

It can be assessed that SVM Tree works as well as the unsupervised decision tree which has been compared with other standard algorithms, and has been shown to stand up to them.

As it may also be said that an alternative approach has been found from the reference model. The same measure as proposed there is used for finding out the required attribute. But, when choosing the attribute among the measures, the unsupervised tree considers maximum measure while here the minimum has been chosen. Also finding the split point at the valleys in a Gaussian mixture distribution is also similar in idea. But the procedure on how the split points are determined from those valley points are certainly different. In the implementation, the previous method doesn't exclude an attribute after being used up in an internal node, but it is done for this model.

The SVM Tree tends to under perform for larger trees on a lower number of folds in cross validation. This is dua to the lower number of training points to learn from in lower folds and higher number of test points to preferable clusters. For that it shows higher test performance when the number of test points get lower. It would be an objective conclusion to make that the SVM Tree is successful in achieving the two objectives with which the project was inspired. It showed the ability to cluster inherently from the dataset, and build a model which can be interpreted for the result it provides.

**Reference**

Basak, J., & Krishnapuram, R. (2005). Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE transactions on knowledge and data engineering*, *17*(1), 121-132.

Elaidi, H., Elhaddar, Y., Benabbou, Z., & Abbar, H. (2018, April). An idea of a clustering algorithm using support vector machines based on binary decision tree. In *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)* (pp. 1-5). IEEE.

Li, J., & Nehorai, A. (2018). Gaussian mixture learning via adaptive hierarchical clustering. *Signal Processing*, *150*, 116-121.

Cirrincione, G., Ciravegna, G., Barbiero, P., Randazzo, V., & Pasero, E. (2020). The GH-EXIN neural network for hierarchical clustering. *Neural Networks*, *121*, 57-73.

Fraiman, R., Ghattas, B., & Svarc, M. (2013). Interpretable clustering using unsupervised binary trees. *Advances in Data Analysis and Classification*, *7*(2), 125-145.

Nie, F., Zhu, W., & Li, X. (2019). Decision Tree SVM: An Extension of Linear SVM for Non-linear Classification. *Neurocomputing*.

Ganaie, M. A., Tanveer, M., & Suganthan, P. N. (2020). Oblique decision tree ensemble via twin bounded SVM. *Expert Systems with Applications*, *143*, 113072.

Vupputuri, A., Ashwal, S., Tsao, B., & Ghosh, N. (2020). Ischemic stroke segmentation in multi-sequence MRI by symmetry determined superpixel based hierarchical clustering. *Computers in Biology and Medicine*, *116*, 103536.

Thirumaran, M., Moshika, A., & Padmanaban, R. (2019, March). Hybrid Model for Web Application Vulnerability Assessment Using Decision Tree and Bayesian Belief Network. In *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)* (pp. 1-7). IEEE.

Sun, L., Zou, B., Fu, S., Chen, J., & Wang, F. (2019). Speech emotion recognition based on DNN-decision tree SVM model. *Speech Communication*, *115*, 29-37.

Irani, J., Pise, N., & Phatak, M. (2016). Clustering techniques and the similarity measures used in clustering: A survey. *International journal of computer applications*, *134*(7), 9-14.

Brijain, M., Patel, R., Kushik, M., & Rana, K. (2014). A survey on decision tree algorithm for classification.

Sakr, G. E., & Elhajj, I. H. (2013). Decision confidence-based multi-level support vector machines. *Engineering Applications of Artificial Intelligence*, *26*(8), 1892-1901.

Cheung, Y. M., & Jia, H. (2013). Categorical-and-numerical-attribute data clustering based on a unified similarity metric without knowing cluster number. *Pattern Recognition*, *46*(8), 2228-2238.

Xu, Z., Li, P., & Wang, Y. (2012). Text classifier based on an improved SVM decision tree. *Physics Procedia*, *33*, 1986-1991.

Pradhan, A. (2012). Support vector machine-a survey. *International Journal of Emerging Technology and Advanced Engineering*, *2*(8), 82-85.

Lopez-Chau, A., Garcia, L. L., Cervantes, J., Li, X., & Yu, W. (2012, November). Data selection using decision tree for SVM classification. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence* (Vol. 1, pp. 742-749). IEEE.

Zhang, D., & Jin, X. B. (2011, July). Build decision tree on support vector machine. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)* (Vol. 2, pp. 997-1001). IEEE.

Ram, P., & Gray, A. G. (2011, August). Density estimation trees. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 627-635).

Madzarov, G., & Gjorgjevikj, D. (2009, May). Multi-class classification using support vector machines in decision tree architecture. In *IEEE EUROCON 2009* (pp. 288-295). IEEE.

Pham, D. T., & Afify, A. A. (2007). Clustering techniques and their applications in engineering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *221*(11), 1445-1459.

Hao, P. Y., Chiang, J. H., & Tu, Y. K. (2007). Hierarchically SVM classification based on support vector clustering method and its application to document categorization. *Expert Systems with applications*, *33*(3), 627-635.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, *16*(3), 645-678.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (Vol. 96, No. 34, pp. 226-231).

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), 121-167.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.