International Journal of Research
Available at
https://www.internationaljournalofresearch.org/
ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

On Distributed Fuzzy Decision Trees For Big Data

Mr GUNTUPALLI VENKATA RAJA **Mr G DILIP KUMAR**

B. Tech Student, Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem, NH-16, Guntur, Andhra Pradesh, India.

Assistant Professor, Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem, on NH-16, Guntur, Andhra Pradesh, India.

Abstract: - *We present a definition of a Fuzzy Prolog Language that models interval-valued Fuzzy Logic, and subsumes former approaches because it uses a truth value representation based on a union of intervals of real numbers and it is defined using general operators that can model different logics. We give the declarative and procedural semantics for Fuzzy Logic programs. In addition, we present the implementation of an interpreter for this language conceived using CLP(R). We have incorporated uncertainty into a Prolog system in a simple way thanks to this constraints system. The implementation is based on a syntactic expansion of the source code during the Prolog compilation.*

Keywords: *Fuzzy Prolog, Modeling Uncertainty, Logic Programming, Constraint Programming Application, Implementation of Fuzzy Prolog.*

1.INTRODUCTION

Decision trees are widely used classifiers, successfully employed in many application domains such as security assessment [1], health system [2] and road traffic congestion [3]. The popularity of decision trees is mainly due to the simplicity of their learning schema. Further, decision trees are considered among the most interpretable classifiers [4], [5], that is, they can explain how an output is inferred from the inputs. Finally, the tree learning process usually requires only a few parameters that must be adjusted. A large number of algorithms have been proposed in the last decades for generating decision trees: most of them are extensions or improvements of the well-known ID3 proposed by Quinlan et al. [6] and CART proposed by Brieman et al. [7]. In a decision tree, each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of the test, and each leaf (or terminal) node holds a class label.

Several works have exploited the possibility of integrating decision trees with the fuzzy set theory to deal with uncertainty [8], [9], leading to the so-called fuzzy decision trees (FDTs). Unlike Boolean decision trees, each node in FDTs is characterized by a fuzzy set rather than a set. Thus, each instance can activate different branches and reach multiple leaves.

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

Both Boolean and fuzzy decision trees are generated by applying a top-down approach that partitions the training data into homogeneous subsets, that is, subsets of instances belonging to the same class [10]. Like classical decision trees, FDTs can be categorized into two main groups, depending on the splitting method used in generating child nodes from a parent node [11]: binary (or two-way) split trees and multi-way split trees. Binary split trees are characterized by recursively partitioning the attribute space into two subspaces so that each parent node is connected exactly with two child nodes. On the other hand, multi-way split trees partition the space into a number of subspaces so that each parent node generates in general more than two child nodes. Since a tree with multi-way splits can be always redrawn as a binary tree, apparently the use of multi way split seems to offer no advantage. We have to consider, however, that binary split implies that an attribute can be used several times in the same path from the root to a leaf. Thus, a binary split tree is generally deeper and sometimes harder to interpret than a multi-way split tree. Further, in some domain, multi-way splits seem to lead to more accurate trees but, since multi-way splits tend to fragment the training data very quickly, they generally need larger data size in order to work effectively.

Typically, FDT learning algorithms require that a fuzzy partition has been already defined upon each continuous attribute. For this reason, continuous attributes are usually discretized by optimizing purposely-defined indexes. Discretization can drastically affect the accuracy of classifiers and therefore should be realized with great care. In authors performed an interesting analysis by investigating how different discretization approaches influence the accuracy and the complexity (in terms of number of nodes) of the generated FDTs: they employed several well-known fuzzy partitioning methods and different approaches for, given a Boolean partition generated by well-known discretization algorithms, defining different types of membership functions. The experimental results reported on 111 different combinations highlight that seven of them outperform the others in both accuracy and number of nodes. FDTs have been mainly used in the literature for classifying small datasets. Thus, FDT learning approaches have focused on increasing classification accuracy, often neglecting time and space requirements, by adopting several heavy tasks such as pruning steps, genetic algorithms, and computation of the optimal split among all points at each node.

Thus, these approaches are not generally suitable for dealing with a huge amount of data. A possible simple solution for applying these approaches would be to select only a subset of data objects by applying some down sampling technique.

However, these techniques may ignore some useful knowledge, making FDT learning approaches purposely designed for managing the overall dataset more desirable and effective. In our context, this means explicitly addressing Big Data. Big Data is a term which identifies datasets so large and complex that traditional data processing approaches are inadequate. Big Data requires specific technologies to support semistructured or unstructured data and scale out with commodity hardware in parallel to cope with ever-growing data volumes.

To address these challenges several solutions have been proposed in the last years, such as (i) cloud computing, an infrastructure layer for big data systems to meet requirements on cost-effectiveness, elasticity, and ability to scale up/out; (ii) distributed file systems and NoSQL databases, for persistent storage and management of massive scheme-free datasets; (iii) MapReduce and Pregel, two programming models proposed by Google for simplifying the distribution of the computation flow across large-scale clusters of machines; (iv) cluster computing frameworks, powerful system-level solutions, like Apache Hadoop and Apache Spark, for distributed data storage and processing, system and failure management, and efficient network bandwidth and disk usage.

Most of the studies recently proposed in the literature for mining big data combine the MapReduce model with the Apache Hadoop and Apache Spark cluster computing frameworks. With regard to classification problems, some recent works have proposed several distributed MapReduce versions of classical algorithms, such as SVM, prototype reduction, kNN, associative classifiers, boosting, decision trees, naive Bayes classifiers and neural networks, investigating performance in terms of speedup. Although with the increase of the number and size of big data, researchers are continuously investigating new algorithms, taking into account not only the accuracy of the classifiers, but also the scalability of the proposed approaches, only few works have integrated fuzzy set theory

Choice trees are broadly utilized classifiers, effectively utilized in numerous application areas, for example, security appraisal, wellbeing framework and street movement clog. The fame of choice trees is for the most part because of the straightforwardness of their learning diagram. Further, choice trees are considered among the most interpretable classifiers, that is, they can clarify how a yield is construed from the sources of info. At last, the tree learning process for the most part requires just a couple of parameters that must be balanced. Countless have been proposed in the most recent decades for producing choice trees: the greater part of them are expansions or upgrades of the notable ID3 proposed by Quinlan et al. and CART proposed by Brieman et al. In a choice tree, each interior (non-leaf) hub indicates a test on a quality, each branch speaks to the result of the test, and each leaf (or terminal) hub holds a class name.

A few works have misused the likelihood of coordinating choice trees with the fluffy set hypothesis to manage vulnerability, prompting the supposed fluffy choice trees (FDTs). Not at all like Boolean choice trees, every hub in FDTs is portrayed by a fluffy set instead of a set. Subsequently, each occurrence can initiate diverse branches and achieve various takes off. Both Boolean and fluffy choice trees are created by applying a best down methodology that segments the preparation information into homogeneous subsets, that is, subsets of occurrences having a place with a similar class. Like traditional choice trees, FDTs can be arranged into two principle gatherings, contingent upon the part strategy utilized in creating tyke hubs from a parent hub: double (or two-way) split trees and multi-way split trees. Twofold split trees are portrayed by recursively dividing the characteristic

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

space into two subspaces so each parent hub is associated precisely with two youngster hubs. Then again, multi-way split trees parcel the space into various subspaces with the goal that each parent hub produces as a rule in excess of two tyke hubs. Since a tree with multi-way parts can be dependably redrawn as a twofold tree, evidently the utilization of multiway split appears to offer no favorable position. We need to consider, in any case, that paired split suggests that a characteristic can be utilized a few times in a similar way from the root to a leaf. In this manner, a paired split tree is by and large more profound and now and then harder to translate than a multi-way split tree. Further, in some space, multi-way parts appear to prompt more exact trees in any case, since multi-way parts tend to section the preparation information rapidly, they by and large need bigger information estimate keeping in mind the end goal to work successfully.

2. RELATED WORK

A number of works have discussed on how a decision tree can be generated efficiently from very large datasets. The various techniques proposed in the literature can be roughly grouped into two categories, which are characterized by performing a pre-sorting of the data or by adopting approximate representations of the data such as samples and/or histograms. While pre-sorting techniques are more accurate, they cannot accommodate very large datasets or streaming data.

One of the oldest approaches in the first category is SLIQ, proposed in. SLIQ reduces decision tree learning time without loss in accuracy by exploiting a pre-sorting technique in the tree-growth phase. This technique is integrated with a breadth-first tree growing strategy to enable classification of disk-resident datasets. SLIQ also uses a tree-pruning algorithm, based on the Minimum Description Length principle, which is inexpensive and results in compact and accurate trees.

However, SLIQ requires that some data per record reside in memory all the time. Since the size of this in-memory data structure grows in direct proportion to the number of input records, this limits the amount of data, which can be classified by SLIQ. SPRINT, proposed in, removes these memory restrictions. Further, it has also been designed to be easily parallelized, achieving good scalability.

As regards the second category, the BOAT algorithm proposed in exploits a novel optimistic approach to tree construction, which generates an initial tree using a small subset of the data and refines it to arrive at the final tree. The authors guarantee that any difference with respect to the real tree (i.e., the tree that would have been constructed by

examining all the data in a traditional way) is detected and corrected. The several levels of the tree are built in only two scans over the training dataset. In a decision tree construction algorithm called SPIES is proposed. SPIES limit the number of possible split points by taking a sample from the data set, partitions the values into intervals and computes the class histograms for candidate split points. This

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

reduces the space complexity of the algorithm and the communication cost between processors.

The different ways to parallelize decision tree learning can be grouped into 4 main categories: i) horizontal, or data-based, parallelism partitions the data so that different processors work on different examples; ii) vertical, or feature based, parallelism enables different processors to consider different attributes; iii) task, or tree node-based, parallelism distributes the tree nodes to the slave processors and iv) hybrid parallelism combines horizontal or vertical parallelism in the first stages of tree construction with task parallelism towards the end. In the authors show how to parallelize two different decision tree learning algorithms, namely C4.5 and the univariate linear discriminant tree proposed in, exploiting horizontal, vertical and task parallelism. Experimental results show that performance of the parallelization highly depends on the dataset, although node-based parallelization shows generally good speed-ups. In the authors exploit an approximate representation and horizontal parallelism. The core of the algorithm is an on-line method for building histograms from streaming data at the processors. The histograms are compressed representations of the data, which can be transmitted to a master processor with low communication complexity. The master processor integrates the information received from all processors and determines which terminal nodes to split and how. A short review on decision tree learning algorithms proposed to handle very large datasets has been presented in. Following the previous works on distributed decision trees, the FDT learning algorithms proposed in this paper exploit both horizontal and task parallelism.

In the last years, some decision tree learning algorithms have been proposed for managing big data by adopting the MapReduce paradigm on the top of Apache Hadoop. MapReduce is based on functional programming and divides the computational flow into two main phases, namely Map and Reduce, which communicate by hkey; valuei pairs. The MapReduce implementation of a distributed decision tree proposed in employs, for instance, four map-reduce stages. The first stage scans the dataset for creating the initial data structures employed in the other three stages. These stages are executed iteratively for, respectively, (i) selecting the best attribute, (ii) updating the statistics for the new nodes and (iii) growing the tree. The experimental results discussed in the paper are limited only to the scalability analysis by varying the number of nodes and the dataset size (up to 3 millions of instances). The effectiveness of the decision trees for managing big data has been demonstrated in real application domains such as stock futures prediction and clinical decision support. Other works exploit decision trees for generating ensemble of classifiers such as random forest.

3. METHODOLOGY

In this section, we first introduce the FDT and the necessary notations used in the paper and then we describe both the MapReduce programming model and the Apache Spark cluster computing framework. This framework is exploited in our DFDT.

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

A. Fuzzy Decision Tree Instance classification consists of assigning a class Cm from a predefined set C = fC1; : : : ;CMg of M classes to an unlabeled instance. Each instance can be described by both numerical and categorical attributes. Let X = fX1; : : : ;XF g be the set of attributes. In case of numerical attributes, Xf is defined on a universe Uf <. In case of categorical attributes, Xf is defined on a set Lf = fLf;1; : : : ;Lf;Tf g of categorical values. An FDT is a directed acyclic graph, where each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of the test, and each leaf (or terminal) node holds one or more class labels. The topmost node is the root node. In general, each leaf node is labeled with one or more classes Cm with an associated weight wm: weight wm determines the strength of class Cm in the leaf node.

Let TR = f(x1; y1); (x2; y2); :::; (xN; yN)g be the training set, where, for each instance (xi; yi), with i = 1; :::;N, yi 2 C and xi;f 2 Uf in case of continuous attribute and xi;f 2 Lf in case of categorical attribute, with f = 1; :::; F. FDTs are generated in a top-down way by performing recursive partitions of the attribute space.

Algorithm 1 shows the scheme of a generic FDT learning process. The Select Attribute procedure selects the attribute used in the decision node and determines the splits generated from the values of this attribute. The selection of the attribute is carried out by using appropriate metrics, which measure the difference between the levels of homogeneity of the class labels in the parent node and in the child nodes generated by the splits. The commonly used metrics are the fuzzy information gain, fuzzy Gini index, minimal ambiguity of a possibility distribution, maximum classification importance of attribute contributing to its consequent and normalized fuzzy Kolmogorov-Smirnov discrimination quality measure. In this paper, we adopt the fuzzy information again, which will be defined in Section IV-B. The splitting method adopted in the Select Attribute procedure determines the attribute to be selected and also the number of child nodes.

In the literature, both multi-way and binary splits are used. We have implemented both the approaches and evaluated their pros and cons. Once the tree has been generated, a given unlabeled instancebx is assigned to a class Cm 2 C by following the activation of nodes from the root to one or more leaves. In classical decision trees, each node represents a crisp set and each leaf is labelled with a unique class label. It follows that bx activates a unique path and is assigned to a unique class. In FDT, each node represents a fuzzy subset. Thus, bx can activate multiple paths in the tree, reaching more than one leaf with different strengths of activation, named matching degrees. Given a current node CN, the matching degree mdCN(bx) of bx with CN is calculated as:

$$md^{CN}(\hat{\mathbf{x}}) = TN(\mu^{CN}(\hat{x}_f), md^{PN}(\hat{\mathbf{x}})) \qquad (1)$$

where TN is a T-norm, CN(bxf) is the membership degree of bxf to the current node CN, which onsiders Xf as splitting attribute, and mdPN(bx) is the matching degree of bx with the parent node PN.

The association degree ADLN m (bx) of bx with the class Cm at leaf node LN is calculated as:

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

where mdLN(bx) is the matching degree of bx with LN and wLN m is the class weight associated with Cm at leaf node LN. In the literature, different definitions have been proposed for weight wLN m. Further, it has been proved that the use of class weights can increase the performance of fuzzy

weighed vote: the class corresponds to the maximum total strength of vote. The total strength of vote for each class is computed by summing all the activation degrees in each leaf for the class. If no leaf has been reached, the instance bx is classified as unknown. In 2004, Google proposed the MapReduce programming framework [5] for distributing the computation flow across large-scale clusters of machines, taking care of communication, network bandwidth, disk usage and possible failures.

Algorithm 1 Pseudo code of a generic FDT learning process.

Require: training set TR, set X of attributes, splitting method $SplitMet$, stopping method $StopMet$

```
1: procedure FDTLEARNING(in: TR, X, SplitMet, StopMet)
2:     root ← create a new node
3:     tree ← TREEGROWING(root, TR, X, SplitMet, StopMet)
4:     return tree
5: end procedure
6: procedure TREEGROWING(in: node, S, X, SplitMet, StopMet)
7:     if STOPMET(node) then
8:         node ← mark node as leaf
9:     else
10:        splits ← SELECTATTRIBUTE(X, S, SplitMet)
11:        for each split_z in splits do
12:            S_z ← get the set of instances from S determined by split_z
13:            child_z ← create one node by using split_z and S_z
14:            node ← connect the node with TREEGROWING(child_z, S_z, X_z, SplitMet, StopMet)
15:        end for
16:    end if
17:    return node
18: end procedure
```

At high level, the framework, which is based on functional programming, divides the computational flow into two main phases, namely Map and Reduce, organized around hkey; valuei pairs. When the MapReduce execution environment runs a user

classifiers. To determine the output class label of the unlabeled instance bx, two different approaches are often adopted in the literature: maximum matching: the class corresponds to the maximum association degree calculated for the instance;

program, the framework automatically partitions the data into a set of independent chunks, that can be processed in parallel by different machines. Each machine can host several Map and Reduce tasks. In the Map phase, each Map task is fed by one chunk of data and, for each hkey; valuei pair as input, it generates a list of intermediate hkey; valuei pairs as output.

In the Reduce phase, all the intermediate results are grouped together according to a key-partitioning scheme, so that each Reduce task processes a list of values associated with a specific key as input for generating a new list of values as output. In general, developers are able to implement parallel algorithms that can be executed across the cluster by simply defining Map and Reduce functions.

4. THE PROPOSED DISTRIBUTED FUZZY DECISION TREE FOR BIG DATA

The proposed distributed approach allows managing a large amount of data: performing the splitting on a group of nodes significantly reduces the number of scans over the training set, but also requires a larger quantity of memory and a longer computation time

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

for each iteration (the computational cost is limited by collecting and aggregating the necessary statistics). Thus, the maximum number maxY of nodes, which can be processed in parallel at each iteration, depends on the memory availability on the cluster. Obviously, the higher the number of categorical values and fuzzy sets defined by the fuzzy partitioning, the higher the memory used for collecting the statistics and the lower the number of nodes that can be processed in parallel at each iteration.

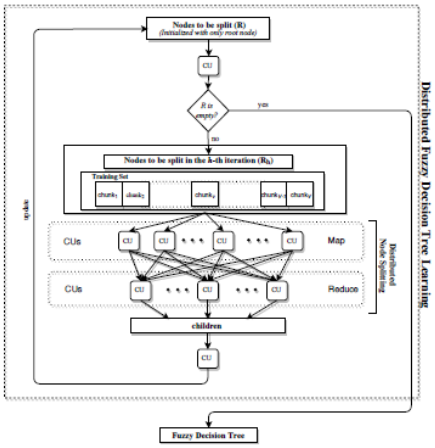

Fig. 6. The overall DFDT Learning approach.

with a complete dataset, varying the number of CUs; iii) ability to efficiently accommodate an increasing dataset size.

As shown in Table I, we employed 10 well-known big datasets freely available from the UCI2 repository. The datasets are characterized by different numbers of input/output instances (from 1 million to 11 millions), classes (from 2 to 50), and attributes (from 10 to 41). For each dataset, we also reportthe number of numeric (num) and categorical (cat) attributes.

5. CONCLUSION

We have proposed a distributed fuzzy decision tree (FDT) learning scheme shaped according to the MapReduce programming model for generating both binary (FBDT) and multi way (FMDT) FDTs from big data. We have first introduced a novel distributed fuzzy discretizer, which generates strong fuzzy partitions for each continuous attribute based on fuzzy information entropy. Then, we have discussed a distributed implementation of an FDT learning algorithm, which employs the fuzzy information gain for selecting the attributes to be used in the decision nodes. We have implemented the FDT learning scheme on the Apache Spark framework.

Experimental results performed on ten real-world big datasets show that our scheme is able to achieve speedup and scalability figures close to the ideal ones. It is worth highlighting that such results can be obtained without adopting any specific dedicated hardware, but rather by using just a few personal computers connected by a Gigabit Ethernet. The results have been compared with the ones obtained by the distributed decision tree (DDT) implemented in the MLlib library on the Apache Spark framework and by the Chi-FRBCS BigData algorithm, a MapReduce distributed fuzzy rule-based classification system. In the comparison we have considered accuracy, complexity and execution time. We have shown that FBDT statistically outperforms FMDT, DDT and Chi-FRBCSBigData in terms of accuracy. In terms of complexity, FBDT and DDT employ a lower (generally one order of magnitude) number of nodes than FMDT. Further, the number of rules, which can be extracted from the three decision

trees, is on average lower than the one of Chi-FRBCS-BigData. From the run-time perspective, FBDT and FMDT require comparable computation times, but both of them are slower than DDT (not surprisingly, considering that FBDT and FMDT perform a fuzzy partitioning step and manage more information, due to fuzzy logic), but faster than Chi-FRBCS-BigData. Finally, computation time scales approximately linear with the number of computational units and instances.

As highlighted in the overall paper, the main reason for proposing FBDT and FMDT is to generate effective and efficient classifiers when managing big data. Obviously, the distribution of the dataset along the computer cluster implies the parallelization of the fuzzy decision tree learning and therefore a faster tree generation. Thus, FBDT and FMDT find a natural use in all the application domains where decision trees have to be generated very quickly from a large amount of data. As an example, the increase of the number of sensors deployed everywhere and the subsequent intent to extract useful knowledge from the data collected by these sensors, has given rise to a growing interest in data mining approaches for streaming data, possibly able to manage concept drift. Most strategies used in this context use sliding windows of fixed or variable sizes and a retraining learning mode. A window is maintained that keeps the most recently acquired examples, and from which older examples are dropped according to some set of rules. Periodically, the retraining learning mode discards the current model and builds a new model from scratch using the buffered data in the windows. An interesting survey of streaming data analysis and concept drift adaptation can be found in [73]. Our fuzzy decision tree learning algorithms result to be particularly suitable for the retraining learning mode, especially when the size of the window is particularly large.

Concluding, we believe that the work presented in this paper is the first extensive study on the application of FDTs to big data, considering both binary and multi-way splits. We expect that the experimental results can be used as baseline for future research in this field.

6. REFERENCES

[1] R. Diao, K. Sun, V. Vittal, R. J. O'Keefe, M. R. Richardson, N. Bhatt, D. Stradford, and S. K. Sarawgi, "Decision tree-based online voltage security assessment using PMU measurements," IEEE Transactions onPower Systems, vol. 24, no. 2, pp. 832–839, 2009.

[2] T. Goetz, The decision tree: Taking control of your health in the new era of personalized medicine. Rodale Inc., 2010.

[3] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw gps data for geographic applications on the web," in Proceedings of the 17th international conference on World Wide Web, 2008, pp. 247–256.

[4] J. Han, M. Kamber, and J. Pei, Data mining: Concepts and techniques. Elsevier, 2011.

[5] L. Rokach and O. Maimon, Data mining with decision trees: Theory and applications. World scientific, 2014.

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

[6] J. R. Quinlan, "Induction of decision trees," Machine learning, vol. 1, no. 1, pp. 81–106, 1986.

[7] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, Classification and regression trees. CRC press, 1984.

[8] C. Z. Janikow, "Fuzzy decision trees: Issues and methods," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 28, no. 1, pp. 1–14, 1998.

[9] Y.-l. Chen, T. Wang, B.-s. Wang, and Z.-j. Li, "A survey of fuzzy decision tree classifier," Fuzzy Information and Engineering, vol. 1, no. 2, pp.149–159, 2009.

[10] J. R. Quinlan, C4.5: Programs for Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[11] X. Liu, X. Feng, and W. Pedrycz, "Extraction of fuzzy rules from fuzzy decision trees: An axiomatic fuzzy sets (AFS) approach," Data & Knowledge Engineering, vol. 84, pp. 1–25, 2013.

[12] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: Data mining, inference and prediction," The Mathematical Intelligencer, vol. 27, no. 2, pp. 83–85, 2005.

AUTHORS :

AKKALA GAYATHRI : Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem, on NH-16, Guntur Andhra Pradesh, India.

GUMMADIDALA ANVESH : Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem, on NH-16, Guntur Andhra Pradesh, India.

KARANAM ASHOK KUMAR : Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem, on NH-16, Guntur Andhra Pradesh, India.