International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 01
January 2017

CART Algorithm for Two and Multi- Party

Divyarth Rai , Raghvendra Kumar
Dept of Computer Engineering LNCT Group of College, Jabalpur, M.P., India

divyarthrai7@gmail.com, Raghvendraagrawal7@gmal.com

Abstract: In this paper we have assumed that participating parties having database that contains private data are vertically partitioned. We have proposed vertically partitioned CART algorithm and for privacy preserving privacy protocol we have combined scalar dot product protocol and xlnx protocol for efficiently preserving privacy of private data. Later we improved privacy preserving CART algorithm for multi-party and for privacy preservation secure sum protocol and secure size of set intersection protocols are used.

Keywords: - Vertical Partitioning, Secure multi party computation, Privacy Techniques, Distributed Database, CART Algorithm.

1. Privacy Preserving CART Algorithm for Two Parties

Let N maps the current node, D = DX ∪ DY maps the current database. Nattr list maps the current test attributes. The PPCART Tree (D, Nattr list) Begin

Algorithm 1: CART for Two Parties

Step1: Create root node R

X computes gini index for all attributes present in DX. Similarly Y computes gini index for all attributes present in DY. Initialize the root with minimum gini index. Attribute of minimum gini index is selected as the attribute maximizes the impurity reduction.

Step 2: If al l records in D have same class value, and then return R as the leaf node with the specified class value.

Each record has been divided in between two parties and both parties share the class labels of all records. So if we want to determine whether the both parties remain with the same single class or not, we have to verify whether the records in DX or DY all belong to the same single class C or not. If they belong to the same single class, then returns the leaf node with that specific class value.

Step 3: If Nattr list is empty or the left records are less than a given value then return R as a leaf node marked with the class value assigned to the most records in S. Both parties share the class labels of all records and the names of all attributes, so they both know whether Nattr list is null or

not. If yes, just scan dataset DX or DY and statistic the most frequent class, marking the leaf with the most frequent class label.

Step 4: A queue Q is initialized to contain the root node.

Step 5: While queue Q is not empty do {

Step 6: Pop out the first node N from Q.

Step 7: Evaluate the gini index for each attribute.

Step 8: Find the best split attribute.

Step 9: If the split attribute is continual then find its partition value.

Step 10: Use the best split attribute to split node N into N1, N2...Nn.

Step 11: For i=1... n

{

If al l records in Ni belong to the same class then return Ni as a leaf

node marked with its class value

Else

Add Ni to Q and go on executing the PPCART Tree (D, Nattr list)

}

}

Step 12: Calculate the classified mistakes of each node and carry on the tree pruning.

Step 13: End

1.1 Computing the Best Split

We need to calculate the gini index for each attribute to acknowledge the best splitting attribute. Let D represent the dataset belonging to the current node N. Let C represents the needed dataset to compute the gini index for each record in the current satisfied node. There will have two kinds of situations:

1. If the attributes of C and the Nattr list belong to the same dataset, either of them can individually calculate gini index.
2. If all the attributes involved in C and the Nattr list do not belong to the same party, neither party can compute the information gain ratio by itself. In this case, everyone needs to union the other party to calculate it. The following three steps will be repeated until we get information gain ratio for all attributes. Finally we choose the attribute with the greatest value as a split attribute for the current node.

Computing L(D, Nattr list):

L represents the logical expression that satisfies the current node N. L rep- resents the logical expression that only involved in DX attributes. LY represents the logical expression that only involved in D' s attributes.

A scan dataset DX and produce a vector of size n. VX (i) = 1 if the ith record satisfies LX else VX (i)=0. X may calculate the value of vector VX by itself.

Similarly, Y may also calculate the value of vector VY.

Let Vi be a vector of size n, Vi(n)=1, if the nth record belongs to class i else Vi(k)=0. V (i) = VX (i)∩ VY (i) means the corresponding record that satisfies both LX and LY

Scalar product VX • VY = Σj=1 to n VX (j) ∗ VY (j)

Means the number of record which LX and LY.

$$Pi = VX \bullet (VY \cap Vj) = (VX \cap Vj) \bullet VY$$

Means the number of satisfies both belonging to class i in partition S. Now we can compute the gini index

$$L(D, NAttr\ list) = \Sigma j=1 \text{ to } k\ (S1j + S2j + ... + Snj)/S \times G(S1j + S2j + ... + Snj) = \Sigma j=1 \text{ to } k\ Pj\ G(S1j + S2j + ... + Snj)$$

1.2 Computing Gain

$$\Delta\ Gain(D, NAttr\ list) = G(S1j + S2j + ... + Snj) - L(D, NAttr\ list)$$

Where L(D, Nattr list) computes the gini index of each attributes and G(S1j + S2j + ... + Snj) computes the gini index of the class values. According to scalar product protocol the semi-honest third party is introduced to compute the scalar product VX •VY without revealing privacy. The result is divided into two parts

$$VX \bullet VY = XX + XY$$

Two parties X and Y respectively shares XX and XY , which can guarantee that X cannot get the contents of Y and Y cannot get the contents of X, so it can preserve their privacy.

According to Xln(X) protocol we can obtain ln(XX + XY) = PX + PY . X and Y respectively shares PX and PY .

$$(XX + XY)\ln(XX + XY) = (XX + XY)(PX + PY) = XX\ PX + XY\ PY + XX\ PY + XY\ PX$$

Where the result of XX PY is divided into two parts QX and QY respectively shared by X and Y. Similarly, the result of XY PX is also divided into two parts SX and SY , which is also respectively shared by X and Y. X can compute WX = XX PX + QX + SX and Y can compute WY = XY PY + QY + SY .So XX + XY ln(XX + XY) = WX + WY , the result is divided into two parts WX and WY ,and respectively shared by X and Y.

Now both parties have their own private data and wishes the other party should not know their private data. So the problems occurred is to find the gini index and differential gini without the knowledge of second party. Let R be the requirement and it is divided into two subsets RX and RY where RX is the subset of requirement involving party X attributes and RY is the subset of requirement involving party Y attributes. Let us consider two vectors VX and VY are of size n respectively. VX (t) = 1 and VY (t) = 1 if tth record satisfies RX

and RY respectively else VX (t) = 0 and VY (t) = 0. Let us consider another vector VB to know if t attribute belong to a class C or not. If VB (C) = 1 then attributes being to class C else VB (C) = 0.

V is a non zero entry where V (t) = VX (t) ∩ VY (t) where (t=1,2,....,n) means V(t) is satisfying both RX and RY . Now party X and Y can compute their own private data by the following formulas for computing scalar product of VX and VY

$$VX \bullet VY = \Sigma t=1 \text{ to } n \; VX(t) * VY(t)$$

For computing PC which means calculating number of occurrences of class C in a partition P is

$$PC = VX \bullet (VY \cap VC) = (VX \cap VC) \bullet VY$$

3. Privacy Preserving CART Algorithm for Multi- Party

Let us consider a view where N number of sites wants to collaborate and compute data mining task using CART algorithm in a secured way.

Algorithm

Step 1: There are N sites participating in the process having A attributes.

Step 2: Each site has a flag F. It is 0 if there is no remaining attribute and 1 if there are attributes remaining.

Step 3: Each site contains a local constraint set (Con) which keeps the values o those attributes that lead to class attribute and changes the value of other attributes to not required (*).

Step 4: Transaction set T partitioned between sites S1....SN

.

Step 5: Site SN holding n class attribute having C1, C2, ..., Cn values.

Begin

{

Step a: if A is empty then

{

A_ empty()

{

S1 chooses a random integer R uniformly from 0 .... m - 1.

S1 sends R + F1 to S2

for J = 2....k - 1 do

site SJ receives R' from SJ −1.

SJ sends R' + FJ mod M to SJ +1

end for

site SN receives R' from SN −1 .

R0 ← R0 + FN modM

S1 and SN uses secure keyed commutative hash keys E1 and EN

S1 sends E1(R) to SN

SN receives E1 (R) and sends EN (E1 (R)) and EN (R0) to S1

S1returns E1 (EN (R0)) = EN (E1 (R))

{⇔ R0 = R ⇔ ΣN

}

FJ = 0 ⇔ 0 attributes remain }

Continue at site SN upto the return

(cont1....contn) ← DistC ount()

DistCount()

{

for all sites SJ except SN

At SJ : XJ ← T ransSet(C ontJ)

TransSet(Cont)

{

}

End for

X=Ø

For all transaction id J T

if tJ satisfies constr

X ← X ∪ J

End if

End for return X

For each class C1,....., Cn

At SN : constr.SN (CR , CJ) /* To include class restriction*/

At SN : XN ← T ransSet(C onN)

C onstJ ← |X1 ∩ X2 ∩ ..... ∩ Xk | using the cardinalty of set intersection protocol.

End for

Return (C ont1,C ont2,....., C ontn)

Build a leaf node with distribution (C ont1 ,...., C ontn)

{

class ← maxJ =1...S consJ

}

Return ID of the constructed node

}

Step b:

Else if C lassNode ← (atSN) CheckSameClass()

CheckSameClass()

{

(Cont1....Contn) ← DistCount()

if ∃I s.t.consI ! = 0 ∩ ∀J ! = I , ConsJ = 0 /* If only one of the count is non zero*/

Build a leaf node with distribution (C ont1....C ontn)

Return ID of the constructed node else

Return false end if

}

Return leaf NodeId ClassNode

Step c:

Else

BestSite ← Diff Gini()

DiffGini()

{

For all sites SN

BestGiniJ ← −1

For each attribute AttrJI at siteSJ

Gini ← D_Gini(AttrJ I)

D_Gini(AttrJ I)

{

P ← DistCount() /* total number of transaction at this node */

PaJ ← DistC ount()

Gini ← GiniI ndex(P)−GiniI ndex(PaJ)∗|PaJ |/|P |

Where |P| isΣm ContJ

End for

Constr.S(Attr, ‘]0) /* update local Tuple*/

Return Gini

}

if Gini > BestGiniJ

BestGiniJ ← Gini BestAttributeJ ← AttrJ I

End if end for

Return maxI , BestGiniI

}

continue execution at Best_Site

create interior node No with attribute

N o.Attr ← BestAttBests ite

for each attribute valueVJ No.attr

C onstr.S(No.Attr, VJ) /* updates local constraints set*/

N odeI d ← P P CART ()

N o.AttrJ ← NodeI d /* add appropriate branch to interior node*/

end for

C onstr.S(Attr, '0) /*Returning to parents; should no longer filter transactions with Attr.*/

Store No locally keyed with N ode_I D

Return Node_ID of interior node No /*Exectution continues at site owning parent node*/

Step d: Build_T ree(T I d, NodeID)

{

End if

}

if NodeId is a leaf node /* Starting site and root node are eturn class or distribution saved in NodeId

Else

No ← local node with ID Node ID

Val ← the value of attributes No. Attr for transaction TID

ChildId ← No. Val

Return ChildId. Site. Build_tree(TID, Child Id)

End if

}

}

2.1 Description

In the algorithm A_empty() checks if A is empty or not. If the result is 0 then no attributes are left and at the end of this step, node is constructed with its ID. A constraint (Con) has be created which helps the sites to keep track of only those attributes and their values which satisfies Con and build its path to class attribute and rest attributes values are changed to not required (≠) values. Initially all the attribute contains not required (≠) value. Constraint just acts as a filter. Constr.S (attribute, value) is a function which converts the attribute with the appropriate values. Attributes that satisfies Con retain their values and rest attributes have not required values. An attribute can satisfy Con if and only if it satisfies the following condition

$$\forall J\ (AttrJ\ (x) = V \Longleftrightarrow C\ on(AttrJ\ ) = V) \cup C\ on(AttrJ\ ) = \neq$$

TransSet (con) returns local transaction satisfying con. DistCount () calculates the class distribution. Two privacy preserving protocols, secure sum protocol is used for to check if any attributes are left and cardinality of set intersection protocol is used to determine the majority class of a node. CheckSameClass() checks if all the transactions belong to same class values. D_Gini() calculates gini index of all the attributes (A). Built_T ree() builds the tree based on NodeId but reveals only the leaf node by classifying the transaction.

2.2. Privacy Preservation of Private Data among Sites

The privacy of private data of all the sites participating is fully secured. Function A_empty() only reveals no data in any site but it checks if any at- tribute left in the site. Since cardinality of set intersection protocol has been used in DistCount() function so it just disclose the combination of constraint set for each class. CheckSameClass() only disclose only the class distribution by discovering the fact that if all transaction are of same class or not and disclose only the class distribution. D_Gini() computes the gini of all attributes and expose only the counts of different attributes and Diff-Gini() only brings out the gini of different sites.

Function Build_Tree() only expose the leaf node by classifying the transactions.

Conclusion

Data mining has numerous classification techniques to organize and classify the knowledge. The ID3 rule has been increased to PPID3 for conserving privacy of personal knowledge and uses info gain as its cacophonic attribute. C4.5 rule is additionally increased to PPC4.5 to avoid any knowledge outpouring and uses Gain quantitative relation as its cacophonic attribute. There square measure numerous strategies, such as, Secure total Protocol, Size of Set Intersection Protocol, Yao's Circuit, Scalar Product Protocol, Xln(X) Protocol, etc. for conserving data's' privacy. CART rule has been increased to PPCART to totally preserve the privacy of personal knowledge and uses gini index as its cacophonic criteria. We've got enforced Privacy conserving CART rule for two parties. For privacy preservation scalar dot product protocol and xlnx protocol is employed to stop knowledge outpouring. Later PPCART rule is extended for multi parties wherever over two parties having vertically partitioned off info will participate in data processing. For preservation of privacy of personal knowledge Secure total and Size of Set Intersection Protocols square measure used. This call tree rule generates the binary tree. Next step of analysis is to seek out the latency and communication overhead of the rule and real time implementation in globe.

References

[1] J. Han and M. Kamber, Data Mining: Concepts and Techniques, 2nd ed. New York: Morgan Kaufmann, 2009.

[2] S. Ceri and G. Pelagatti, Distributed Databases: Principles and Systems. Singapore: Interntional Edition, 1984.

[3] W. Du and Z. Zhan, "Building decision tree classifier on private data," in 14th IEEE International Conference on Privacy, Security and Data Mining, Darlinghurst, Australia, 2002, pp. 1–8.

[4] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, pp. 81–106, 1986.

[5] A. C.-C. Yao, "How to generate and exchange secrets(extended abstract)," in 27th IEEE symposium on Foundation of Computer Science(FOCS), 1986, pp. 162–167.

[6] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in Proceedings of

the 2000 ACM SIGMOD international conference on Management of data, vol. 29, New York, 2000, pp. 439–450.

[7] Y. Lindell and B. Pinkas, "Privacy preserving data mining," Advances in Cryptology - Crypto2000, vol. 1880, pp. 1–26, 2000.

[8] R. Sugumar, C. Jayakumar, and A. Rengarajan, "Design a secure multiparty computation system for privacy preserving data mining," International Journal of Computer Science and Telecommunications, vol. 3, pp. 101–105, 2012.

[9] J. Vaidya, C. Clifton, M. Kantarcioglu, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," in ACM SIGKDD Explorations, vol. 4, no. 2, 2002, pp. 28–34.

[10] C. Clifton and D. Marks, "Security and privacy implementations of data mining," in ACM SIGMOD Workshop on Data Mining and Knowledge Discovery, 1996.

[11] J. Vaidya, "Privacy preserving data mining over vertically partitioned data," Ph.D. dissertation, Purdue University, 2004.

[12] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy preserving decision trees over vertically partitioned data," in ACM Transactions on Knowledge Discovery from Data, vol. 2, no. 3, 2008, pp. 14–41.