Building Decision Tree Classifier on Private Data

Author: Wenliang (Kevin) Du and Zhijun Zhan

Center for Systems Assurance
Electrical Engineering and Computer Science
Syracuse University

Overview

- Privacy-preserving Data Mining
- Problem Definition
- Building Block: Scalar Product Protocol
- Decision Tree Building
- Conclusion
Privacy-preserving Data Mining

Alice
Database A

Bob
Database B

Data Mining

Results (classification, association rules, etc.)

Data Partitions

Horizontally partitioned data
Vertically partitioned data

This paper: vertically partitioned data.
Problem Definition

- Alice has a private data set $S_a$.
- Bob has a private data set $S_b$.
- They want to build a decision tree classifier on $[S_a \times S_b]$ (Vertically Partitioned),
- Without disclosing their private data.

Trusted Third Party Model

1/3/2003

1/3/2003
Our solution is based on commodity server model

Commodity Server (CS) Model

Commodity Server (CS)

Commodities

Alice

Bob

S_a

S_b

CS Properties

Assumption: CS cannot collude with either Alice or Bob.
CS is not a trusted party.
CS doesn’t participate in the computation.
CS does not receive private data from Alice or Bob.
The commodities from CS are independent from Alice and Bob’s private data, so they can be generated offline (namely, CS can sell random data for profit 😊).
Security Assumption

- In the Trusted 3rd Party model, Alice and Bob need to assume that the 3rd party does not abuse their private data.
- In the CS model, Alice and Bob need to assume that the CS does not collude with the other party. This security assumption is weaker than the Trusted 3rd Party model, so CS model is more realistic.

The Origin of The CS Model.

- Commodity Server model was proposed by Beaver (1997).
- CS has been used for solving the Private Information Retrieval problem.
Building Block: Scalar Product

- Alice has a private vector A
- Bob has a private vector B
- They want to compute the scalar product of these two vectors
  \[ A \cdot B = \sum A(i)B(i) \]
- Nobody wants to disclose its private data to the other party.

Scalar Product Protocol using Commodity Server

1. \[ A' = A + r_a \]  
   \[ B' = B + r_b \]
2. \[ A'B + r_b \]
3. Alice computes:
   \[ (A'B + r_b) - (R_aB') + r_a \]
   \[ = A\cdot B + r_a + B\cdot r_b - R_a\cdot B - R_a\cdot R_b + r_a \]
   \[ = A\cdot B \]

\[ R_a, R_b: \text{randomly generated vectors.} \]
\[ r_a, r_b: \text{random numbers} \]
Performance

- **Efficiency**
  - Communication cost: 2n
  - Computation cost: 2n
  - Very close to the optimal solution (the one without worrying about the security concerns)
    - Optimal:
      - communication cost: n
      - computation cost: n

Comparing with Existing Work

- **Scalar Product Protocols** based on 2-party model were proposed by:
  - Du and Atallah (2001)
  - Vaidya and Clifton (2002)

- **Advantage**: our scheme is more efficient (close to optimal solutions)

- **Disadvantage**: the security assumption (about collusion) is stronger than the 2-party model
Decision Tree Building

Procedure:
- Evaluate splits for each attribute,
- Select the best split,
- Create partitions using the best split.

How to find the best split?

How to Find The Best Split?

Splitting Index
- Evaluate the “goodness” of a split
- Several splitting schemes have been proposed:
  - Entropy
  - Gini Index
Entropy-based Splitting

We want to measure how good it is to use attribute A to partition S:

\[ S \rightarrow A \]

\[ A = v_1 \quad A = v_2 \quad A = v_3 \]

Entrophy: 

\[ Entropy(S) = -\sum_{j=1}^{m} P_j \log P_j \]

Gain: 

\[ Gain(S, A) = Entropy(S) - \sum_{v \in A} \left( \frac{|S_v|}{|S|} \cdot Entropy(S_v) \right) \]
Selecting the Splitting Attribute

- Compute Gain(S, A) for each attribute A.
- Select the attribute with the largest information gain as the best split for the current node.
- Partition S using the selected attribute.

Challenges in Our Problem

- S is unknown
  - S contains data that satisfy (A_3=1) and (B_2=1)
  - A_3 is known to Alice only
  - B_2 is known to Bob only
- How to compute $P_j$, Entropy(S), Entropy(S_v), |S| and |S_v| without knowing what S contains?
Overview of Our Solution (1)

- Assume records in S must satisfy requirements R.
- Divide R into 2 parts: \( R = (R_a \text{ and } R_b) \)
  - \( R_a \) contains Alice’s attributes
  - \( R_b \) contains Bob’s attributes
- Alice computes vector \( V_a \):
  - \( V_a(i) = 1 \) if the \( i^{th} \) record satisfies \( R_a \); \( V_a(i) = 0 \) otherwise
- Bob computes vector \( V_b \):
  - \( V_b(i) = 1 \) if the \( i^{th} \) record satisfies \( R_b \); \( V_b(i) = 0 \) otherwise
- \( |S| = V_a \cdot V_b \) : use our scalar product protocol!

Overview of Our Solution (2)

- Similarly we can compute the following using scalar product protocol (assume the candidate attribute A belongs to Alice):
  - \( |S_A| : R = (R_a \text{ and } A=v) \) and \( R_b \)
  - \( P_j : R = (R_a \text{ and } Class=j) \) and \( R_b \)
    - or \( R = R_a \) and \( (R_b \text{ and } Class=j) \)
- Finally we can compute Entropy(S), Entropy(\( S_v \)), and Gain(S, A).
Example

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Play Ball</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Rain</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>No</td>
</tr>
</tbody>
</table>

Alice

<table>
<thead>
<tr>
<th>Day</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Ball</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

Bob

How to Find the Best Split

- Assuming we already find the root, which is attribute Outlook.
- We will show how to obtain the best split for the left child of the attribute Outlook.
- We use $\text{Gain}(S, \text{Humidity})$ as an example.
Alice’s Vectors

- Alice computes:
  - $V_a(\text{outlook}=\text{Rain}) = (0,0,1,1,1)$
    - 1 means (outlook=Rain).
  - $V_a(\text{outlook}=\text{Rain}, \text{playball}=\text{No}) = (0,0,0,0,1)$
    - 1 means (outlook=Rain) and (playball=No).
  - $V_a(\text{outlook}=\text{Rain}, \text{playball}=\text{Yes}) = (0,0,1,1,0)$:
    - 1 means (outlook=Rain) and (playball=Yes).

Bob’s Vectors

- Bob computes:
  - $V_b(\text{humidity}=\text{High}) = (1,1,1,0,0)$
    - 1 means (Humidity=High).
  - $V_b(\text{humidity}=\text{Normal}) = (0,0,0,1,1)$
    - 1 means (Humidity=Normal).
Compute \( \text{Entropy}(S_{v=\text{High}}) \) (Example)

- \(|S_{v=\text{High}}| = V_a(\text{outlook}=\text{Rain}) \cdot V_b(\text{humidity}=\text{High})\)
- \(P_0 = P(\text{outlook}=\text{Rain}, \text{playball}=\text{No}, \text{humidity}=\text{High}) = V_a(\text{outlook}=\text{Rain}, \text{playball}=\text{No}) \cdot V_b(\text{humidity}=\text{High})\)
- \(P_1 = P(\text{outlook}=\text{Rain}, \text{playball}=\text{Yes}, \text{humidity}=\text{High}) = V_a(\text{outlook}=\text{Rain}, \text{playball}=\text{Yes}) \cdot V_b(\text{humidity}=\text{High})\)
- \(\text{Entropy}(S_{v=\text{High}}) = -(p_0/|S_{v=\text{High}}|)\log(p_0/|S_{v=\text{High}}|) - (p_1/|S_{v=\text{High}}|)\log(p_1/|S_{v=\text{High}}|)\)
- Similarly, we can compute \(\text{Entropy}(S_{v=\text{Normal}})\), and finally \(\text{Gain}(S, \text{Humidity})\).

Security Analysis

- Information Disclosure
  - From the algorithm
  - From the results (the final decision tree)
- Information Disclosure From our Algorithm
  - The results of the scalar product can disclose some information
  - Better protection scheme can be used (details in the paper)
- Information Disclosure from the final decision tree
  - How much can be disclosed needs more study
Conclusion

- We presented a solution for building a decision tree classifier on vertically partitioned private data.
- The scalar product protocol with a commodity server is used as a basic tool.
- Future studies will focus on developing more efficient solutions.