Privacy Preserving Data Mining

Yehuda Lindell

Benny Pinkas

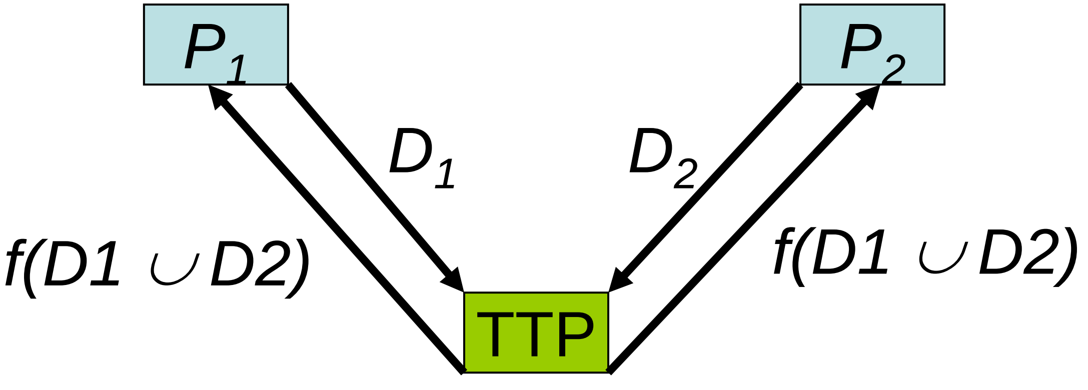
Presenter: Justin Brickell

Mining Joint Databases

- Parties P_1 and P_2 own databases D_1 and D_2
- f is a data mining algorithm
- Compute $f(D_1 \cup D_2)$ without revealing "unnecessary information"

Unnecessary Information

- Intuitively, the protocol should function *as if a trusted third party computed the output*



Simulation

- Let msg(P_2) be P_2's messages
- If S_1 can simulate msg(P_2) to P_1 given only P_1's input and the protocol output, then msg(P_2) must not contain unnecessary information (and vice-versa)
- $S_1(D_1, f(D_1, D_2)) =^C \text{msg}(P_2)$

More Simulation Details

- The simulator S_1 can also recover r_1, the internal coin tosses of P_1
- Can extend to allow distinct $f_1(x,y)$ and $f_2(x,y)$
  - Complicates the definition
  - Not necessary for data mining applications

The Semi-Honest Model

- A *malicious* adversary can alter his input
  - $f(\varnothing \cup D_2) = f(D_2)$!
- A semi-honest adversary
  - adheres to protocol
  - tries to learn extra information from the message transcript

General Secure Two Party Computation

- *Any* algorithm can be made private (in the semi-honest model)
  - Yao's Protocol
- So, why write this paper?
  - Yao's Protocol is inefficient
  - This paper privately computes a *particular* algorithm more efficiently

Yao's Protocol (Basically)

- Convert the algorithm to a circuit
- P_1 hard codes his input into the circuit
- P_1 transforms each gate so that it takes *garbled* inputs to *garbled* outputs
- Using 1-out-of-2 oblivious transfer, P_1 sends P_2 garbled versions of his inputs

Garbled Wire Values

- P_1 assigns to each wire i two random values (W_i^0, W_i^1)
  - Long enough to seed a pseudo-random function F
- P_1 assigns to each wire i a random permutation over {0,1}, $\pi_i : b_i \rightarrow c_i$
- $\langle W_i^{b_i}, c_i \rangle$ is the ‘garbled value’ of wire i

Garbled Gates

- Gate g computes $b_k = g(b_i,b_j)$
- Garbled gate is a table T_g computing

$\langle W_i^{b_i}, c_i\rangle\ \langle W_j^{b_j}, c_j\rangle \rightarrow \langle W_k^{b_k}, c_k\rangle$

  - T_g has four entries:
  - c_i,c_j: $\langle W_k^{g(b_i,b_j)}, c_k\rangle \oplus F\,[W_i^{b_i}](c_j) \oplus F\,[W_j^{b_j}](c_i)$

Yao's Protocol

- P_1 sends
  - P_2's garbled input bits (1-out-of-2)
  - T_g tables
  - Table from garbled output values to output bits
- P_2 can compute output values, but P_1's input and intermediate values appear random

Cost of circuit with *n* inputs and *m* gates

- Communication: *m* gate tables
  - $4m$ · length of pseudo-random output
- Computation: *n* oblivious transfers
  - Typically much more expensive than the *m* pseudo-random function applications
- <u>Too expensive for data mining</u>

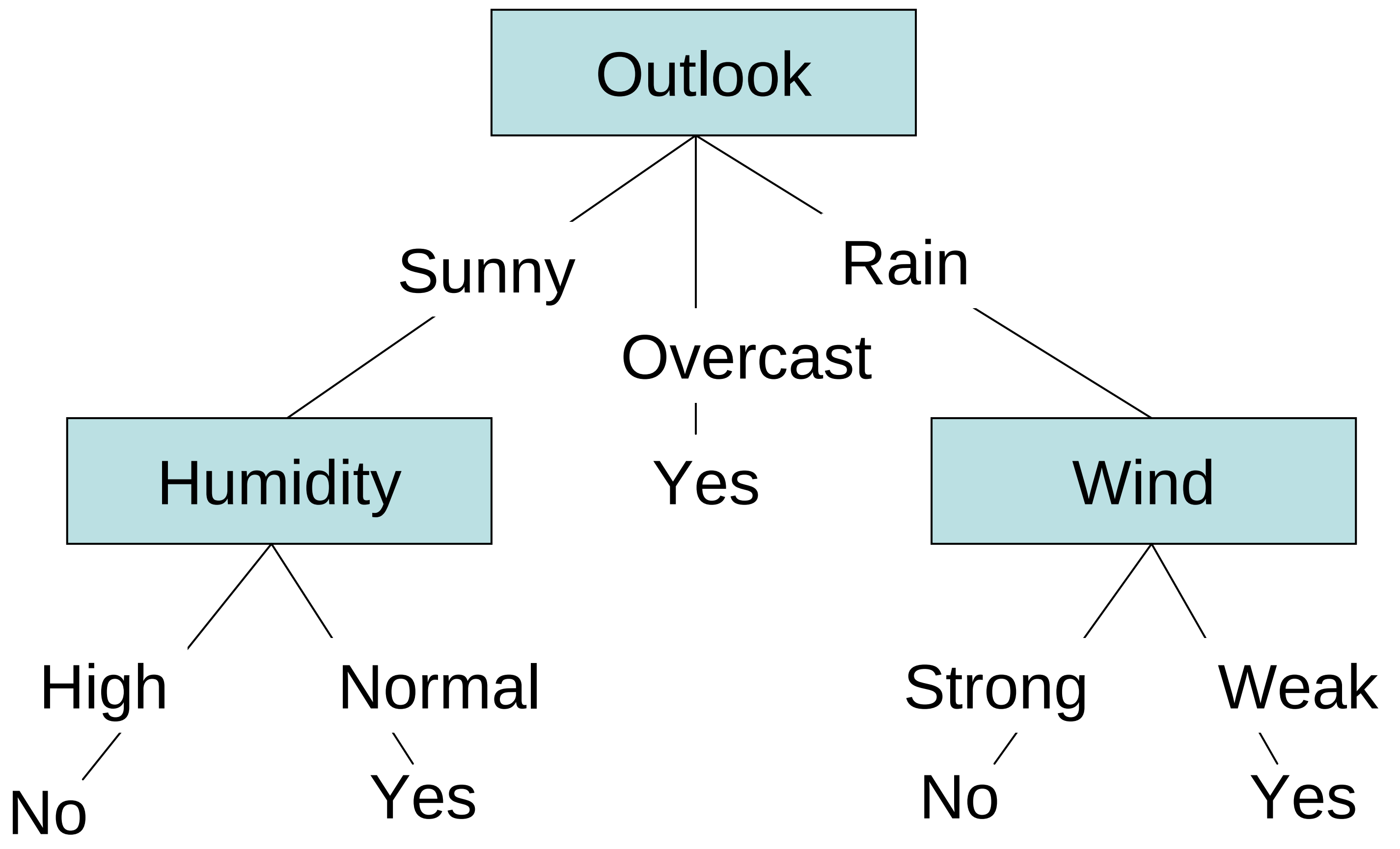
Classification by Decision Tree Learning

- A classic machine learning / data mining problem
- Develop rules for when a *transaction* belongs to a *class* based on its *attribute values*
- Smaller decision trees are better
- ID3 is one particular algorithm

A Database…

Outlook	Temp	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Mild	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

… and its Decision Tree



The ID3 Algorithm: Definitions

- *R*: The set of *attributes*
  - *Outlook, Temperature, Humidity, Wind*
- *C*: the *class* attribute
  - Play Tennis
- *T*: the set of *transactions*
  - The 14 database entries

The ID3 Algorithm

ID3(R,C,T)

- If R is empty, return a leaf-node with the most common class value in T
- If all transactions in T have the same class value c, return the leaf-node c
- Otherwise,
  - Determine the attribute A that *best* classifies T
  - Create a tree node labeled A, recur to compute child trees
  - edge a_i goes to tree ID3(R - {A},C,T(a_i))

The *Best* Predicting Attribute

- Entropy!
- $$H_C(T) = \sum_{i=i}^{l} -\frac{|T(c_i)|}{|T|} \log \frac{|T(c_i)|}{|T|}$$
- $$H_C(T \mid A) = \sum_{j=i}^{m} \frac{|T(a_j)|}{|T|} H_C(T(a_j))$$
- Gain(A) $=^{\text{def}}$ $H_C(T)$ - $H_C(T|A)$
- Find *A* with maximum gain

Why can we do better than Yao?

- Normally, private protocols must hide intermediate values
- In this protocol, the assignment of attributes to nodes is *part of the output* and may be revealed
  - H values are not revealed, just the identity of the attribute with greatest gain
- This allows genuine recursion

How do we do it?

- Rather than maximize gain, minimize
  - $H'_C(T|A) =^{\text{def}} H_C(T|A)\cdot|T|\cdot\ln 2$
- This has the simple formula

$$\hat{H}_C(T \mid A) = \sum_{j=1}^{m}\sum_{i=i}^{l}\left|T\left(a_j, c_i\right)\right| \cdot \ln\left(\left|T\left(a_j, c_i\right)\right|\right) + \sum_{j=1}^{m}\left|T\left(a_j\right)\right| \cdot \ln\left(\left|T\left(a_j\right)\right|\right)$$

- Terms have form $(v_1+v_2)\cdot\ln(v_1+v_2)$
  - P_1 knows v_1, P_2 knows v_2

Private $x \ln x$

- Input: P_1's value v_1, P_2's value v_2
- Auxiliary Input: A large field $\mathcal{F}$
- Output: P_1 obtains $w_1 \in \mathcal{F}$, P_2 obtains $w_2 \in \mathcal{F}$
  - $w_1 + w_2 \approx (v_1 + v_2)\cdot\ln(v_1+v_2)$
  - w_1 and w_2 are uniformly distributed in $\mathcal{F}$

Private $x \ln x$: some intuition

- Compute shares of x and $\ln x$, then privately multiply
- Shares of $\ln x$ are actually shares of n and ε where $x = 2^n(1+\varepsilon)$
  - $-1/2 \le \varepsilon \le 1/2$
  - Uses Taylor expansions

Using the x ln x protocol

- For every attribute A, every attribute-value aj $\in$ A, and every class ci $\in$ C
  - $w_{A,1}(a_j)$, $w_{A,2}(a_j)$, $w_{A,1}(a_j,c_i)$, $w_{A,2}(a_j,c_i)$
  - $w_{A,1}(a_j) + w_{A,2}(a_j) \approx |T(a_j)| \cdot \ln(|T(a_j)|$
  - $w_{A,1}(a_j,c_i) + w_{A,2}(a_j,c_i) \approx |T(a_j,c_i)| \cdot \ln(|T(a_j,c_i)|$

Shares of Relative Entropy

- P_1 and P_2 can locally compute shares

  $$S_{A,1} + S_{A,2} \approx H'_C(T|A)$$

- Now, use the Yao protocol to find the A with minimum Relative Entropy!

A Technical Detail

- The logarithms are only approximate
  - $ID3_\delta$ algorithm
  - Doesn't distinguish relative entropies within δ

Complexity for each node

- For $|R|$ attributes, m attribute values, and l class values
  - $x \ln x$ protocol is invoked $O(m \cdot l \cdot |R|)$ times
  - Each requires $O(\log|T|)$ oblivious transfers
  - And bandwidth $O(k \cdot \log|T| \cdot |S|)$ bits
    - k depends logarithmically on δ
- Depends only logarithmically on |T|
- Only $k \cdot |S|$ worse that non-private distributed ID3

Conclusion

- Private computation of ID3($D_1 \cup D_2$) is made feasible
- Using Yao's protocol directly would be impractical
- Questions?