Privacy-preserving Data Mining

Konstantin Tretjakov (kt@ut.ee)

17. october 2005

Contents

- Motivation for privacy-preserving data-mining
- Two flavors of PPDM, two approaches.
- Examples of cryptographic techniques:
  - Secure sum
  - 1-out-of-2 OT
  - Secure function computation by Yao's method
  - Secure ID3
- A brief introduction to the randomization approach

Motivations

- Sophisticated statistical analysis methods require data, a lot of it.
- Often those that need statistical analyses are not the ones that have the data.
- The data owners have nothing against those other guys doing their statistics, if only they wouldn't have to disclose their data completely.
- But the statisticians need the data to run the analsysis! What can they do?

Solutions

1. Magic and wizardry

Solutions

1. Magic and wizardry
2. Armed assault or data theft

Solutions

1. Magic and wizardry
2. Armed assault or data theft
3. Trusted third party

Solutions

1. Magic and wizardry
2. Armed assault or data theft
3. Trusted third party
4. PPDM

Two Flavors of PPDM

- Secure multi-party computation: compute $f(D_1, D_2, \ldots, D_n)$ so that D_i are provided by different parties but not disclosed.
- Database randomization: modify the database D to another form D' so that it preserves privacy but still allows to run meaningful analyses.
- The first flavor usually assumes distributed data, the second assumes centralized.

Example: Secure sum

- Problem: parties $P_1, \ldots, P_n$ have *secret values* $u_1, \ldots, u_n$ and wish to compute the sum of these values without disclosing the values themselves.
- Assume: $u_i \in \mathbb{N}$, $\sum_i u_i =: u \in [0, m-1]$
- Solution:
  1. P_1 generates a random r uniformly from $[0, m-1]$.
  2. P_1 passes $r + u_1$ to P_2 (all additions modulo m).
  3. P_2 passes $r + u_1 + u_2$ to P_3
  4. $\ldots$
  5. P_n passes $r + \sum_i u_i$ to P_1.
  6. P_1 subtracts r from the received value getting u.

Example: Secure sum

- The algorithm is secure in the sense that it can be simulated by an *ideal scenario* where every party sends its value to a single trusted server that secretly computes the sum.
- But:
  - The channels between P_i and P_{i+1} must be secure.
  - The parties must be *semi-honest*.

Adversaries

- *Semi-honest* (also *passive*): correctly follow the protocol but might try to use the received messages to crack the protocol.
- *Malicious* (also *active*): might violate the protocol.
- Usually we consider the semi-honest adversaries because it's so much harder to fight the evil guys.
- Usually the adversaries are assumed to have limited computational power (e.g. polynomial computations only).

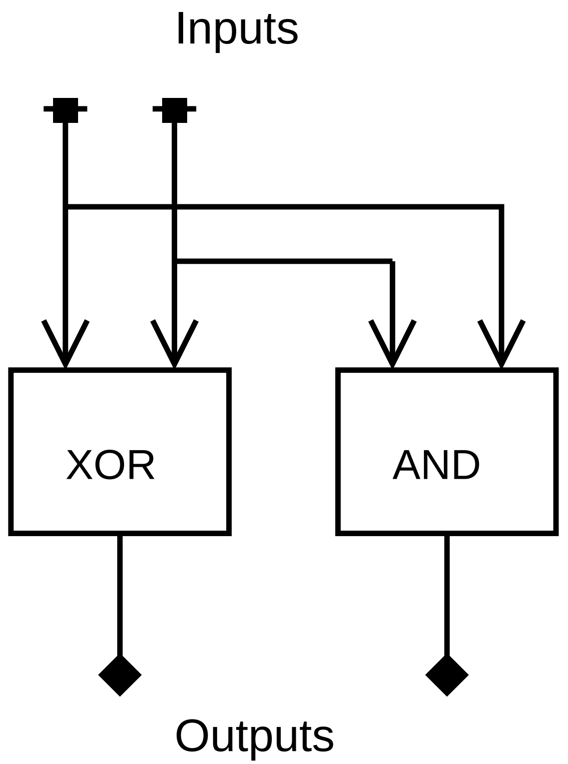
1-out-of-2 Oblivious Transfer

- Bob has two values, y_1 and y_2. Alice wishes to get y_k. However, Alice does not want Bob to know which y_k it wants.
- *1-out-of-2 oblivious transfer (OT) protocol* allows Alice to get y_k from Bob not disclosing which y_k it was that it got.
- Here's a simple example of such a protocol:
  1. Alice sends to Bob two public keys (c_1, c_2) and a proof that she knows the private key for only one of them.
  2. Bob sends to Alice the values y_i encrypted with the corresponding keys: $E_{c_i}(y_i)$.

Secure Function Computation

- Suppose Alice and Bob have pieces of data x and y correspondingly. They want to find out $f(x, y)$ without disclosing their values to each other.
- This can be done using a protocol by Yao (1986).
  - Express f as a combinatorial circuit (e.g. boolean circuit).
  - Bob *garbles* the circuit and his input values, and sends everything to Alice.
  - Alice plugs in her values and computes the function in this "garbled" representation.
  - The results are "ungarbled".

Function as a Circuit



A boolean circuit representation of a function that adds two bits (a half-adder).

Garbling

- For each wire i in the circuit Bob generates random numbers $w_i^{(0)}$ and $w_i^{(1)}$.
- For each gate Bob constructs a table that maps garbled input values to the encryptions of the garbled output values. E.g. for the AND gate:

Wire i	Wire j	Encrypted wire k
$w_i^{(0)}$	$w_j^{(0)}$	$E_{w_i^{(0)},w_j^{(0)}}(w_k^{(0)})$
$w_i^{(0)}$	$w_j^{(1)}$	$E_{w_i^{(0)},w_j^{(1)}}(w_k^{(0)})$
$w_i^{(1)}$	$w_j^{(0)}$	$E_{w_i^{(1)},w_j^{(0)}}(w_k^{(0)})$
$w_i^{(1)}$	$w_j^{(1)}$	$E_{w_i^{(1)},w_j^{(1)}}(w_k^{(1)})$

Garbling

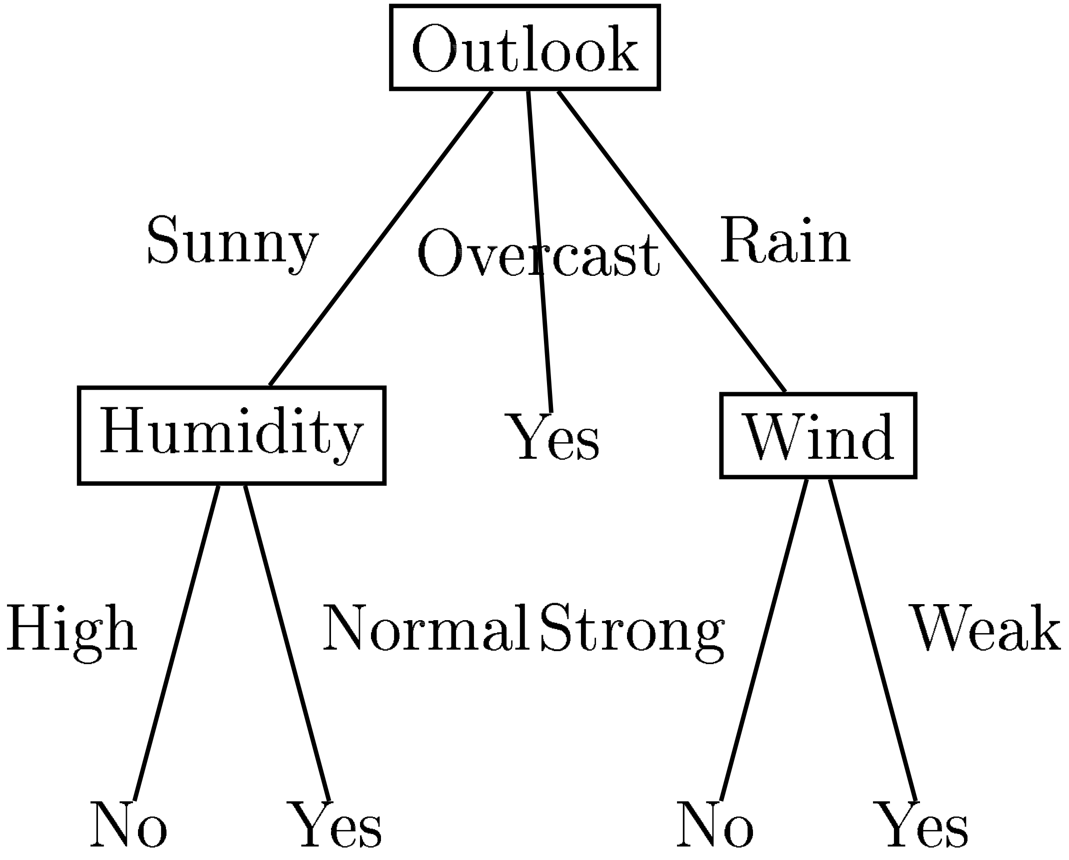
- Bob sends the gate tables and the garbled versions of his input wires to Alice.
- Alice uses 1-out-of-2 OT to obtain the garbled versions of her input wires.
- Alice computes the circuit.
- Bob translates the computed garbled values to the real ones.

Data-mining

- Every data mining algorithm can be descried as a function of the data $f(D)$.
- However, the circuit representation of such a function and the size of its inputs is usually enormous, thus we can't use Yao's algorithm directly and have to make up special optimizations for each data-mining algorithm separately.
- A simple example: calculating the mean of a large set of numbers.
- A more interesting example: ID3.

ID3

- ID3 is an algorithm by Quinlan that constructs *decision trees* from data.



ID3: The algorithm

Input: a database D. Output: a decision tree.

1. If D is empty, return an empty tree.
2. If all the transactions in D have the same class value, return a leaf node indicating this class value.
3. Otherwise, determine the attribute A that *best* classifies the transactions in D.
4. Create a new tree node N, that splits the data on the values of this attribute.
5. For each value a_i of the attribute A attach to N a subtree returned by $ID3(D[A = a_i])$.

Entropy and Mutual Information

- How to find the attribute A that best predicts the class C at node N?
- Use the attribute A that has maximal *information gain*, which is its *mutual information* with C:

$$\mathrm{Gain}(A) = I_D(A;C) = H_D(C) - H_D(C|A)$$

$$H(C) = -\sum_{c \in C} P(c) \log P(c)$$

$$H(C|A=a) = -\sum_{c \in C} P(c|A=a) \log P(A=a)$$

$$H(C|A) = \sum_{a \in A} P(a) H(C|A=a)$$

Secure ID3

- Note that in order to secure ID3 we must secure only the step where the attribute A is computed by maximizing gain.
- Maximizing gain is same as minimizing $H_D(C|A)$.

$$H_D(C|A) = \sum_i \frac{|D([A = a_i])|}{|D|} \times$$

$$\times \sum_k -\frac{|D([A = a_i, C = c_k])|}{|D([A = a_i])|} \log \frac{|D([A = a_i, C = c_k])|}{|D([A = a_i])|}$$

Secure ID3

$$H_D(C|A) = \frac{1}{|D|}\left(-\sum_i \sum_k |D(a_i, c_k)| \log |D(a_i, c_k)| + \right.$$

$$\left. + \sum_k |D(a_i)| \log |D(a_i)| \right)$$

The constant $\frac{1}{|D|}$ can be dropped and we are left with the sum of the form

$$\sum x \log x$$

Secure ID3

- We are left with the sum of the form $\sum x \log x$ where each x can be computed as a sum of two values, one of which is known to Alice, and another—to Bob:

$$\sum (a_i + b_i) \log(a_i + b_i)$$

- We can apply the Yao's algorithm now to compute this sum securely.
- Actually one additional optimization is used in order to make this computation practical. It is a bit too involved so we skip it and say that we are done.

Randomization

- Example situation: the clients submit their age to the server because the server needs to find out their average age.
- The clients could as well add a zero-mean random variable to their age before submitting. If the number of clients is large, the server will get more-or-less the same average.
- More generally: the clients have values x_i from a distribution F_X. They add a shift y_i from distribution F_Y and submit $z_i = x_i + y_i$ to the server.
- The server now wants to restore F_X knowing F_Z and F_Y.

Randomization

Algorithm for restoring F_X proposed by Agrawal:

- Denote by f_X the density of F_X, by f_Y - the density of F_Y.
- Start with initial guess $f_X^0 = U$ and iterate the following step until convergence:

$$f_X^{j+1}(a) = \frac{1}{N} \sum_{i=1}^{N} \frac{f_Y(z_i - a) f_X^j(a)}{\int_{-\infty}^{\infty} f_Y(z_i - z) f_X^j(z) dz}$$

Privacy breaches

- The major difference of the randomization approach is that it always leaks some data.
- This brings the additional requirements to measure privacy and the notion of *privacy breach*.
- The most simple way to measure privacy is using the a-posteriory confidence intervals on x_i-s.
- But:
  - Suppose we add a uniform random number from $[-50, 50]$ to the submitted age. The 100% confidence interval is 100 units wide.
  - However, if the client submits the value 125 we may state with 90% probability that his age is near 75.

Privacy breaches

- Another way is to use information-theoretical terms like mutual information.
- Yet another possibility is to specify explicitly the probabilities of possible privacy breaches.
- The different choices of measures of privacy and the privacy breaches make the randomization approach a bit too ad-hoc.

Summary

- PPDM is a way for performing data-mining securely.
- It is a viable alternative to methods based on magic or armed assault. The trusted server solution still has certain unbeaten advantages however.
- PPDM comes in 2 flavors: secure multi-party computation and randomization. The flavors are often associated with their cryptographic and statistical solutions correspondingly.
- For both flavors there exist effective algorithms for decision-tree induction, association rule mining, clustering and more.

Questions?