Privacy-Preserving Data Mining

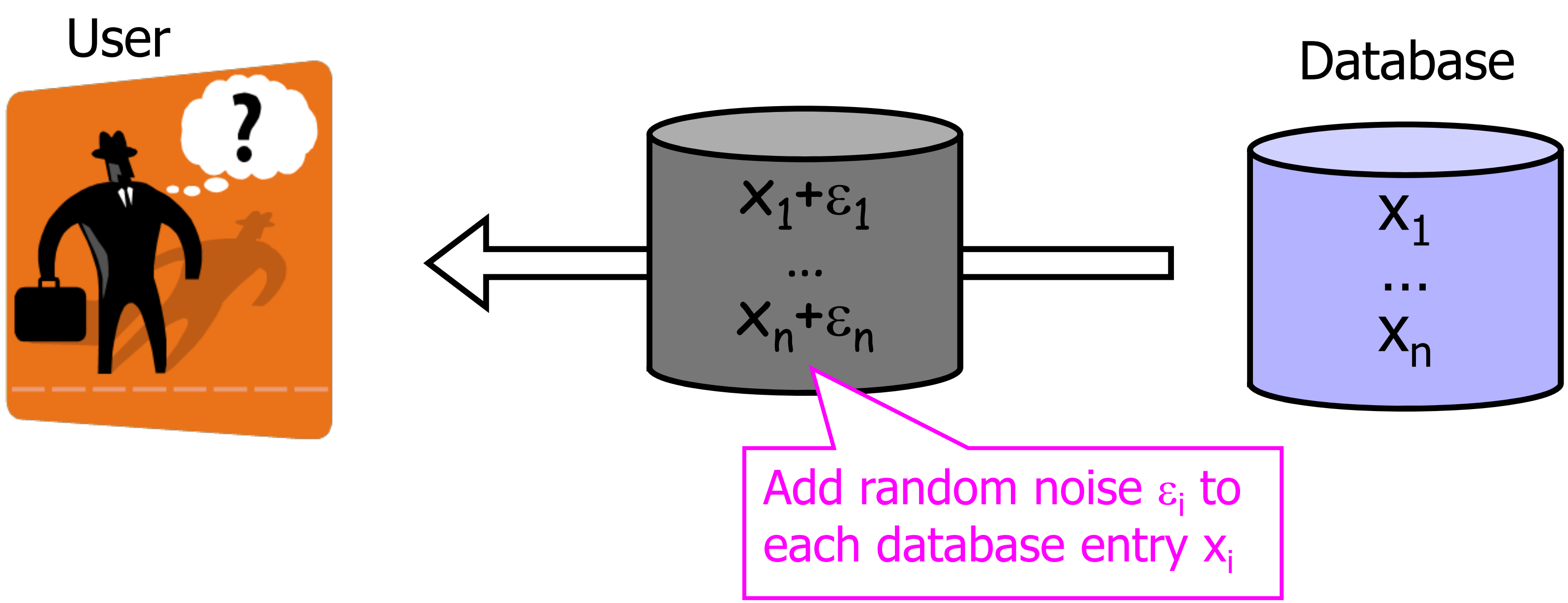
Vitaly Shmatikov

Reading Assignment

- Evfimievski, Gehrke, Srikant. "Limiting Privacy Breaches in Privacy-Preserving Data Mining" (PODS 2003).
- Blum, Dwork, McSherry, and Nissim. "Practical Privacy: The SuLQ Framework" (PODS 2005).

Input Perturbation

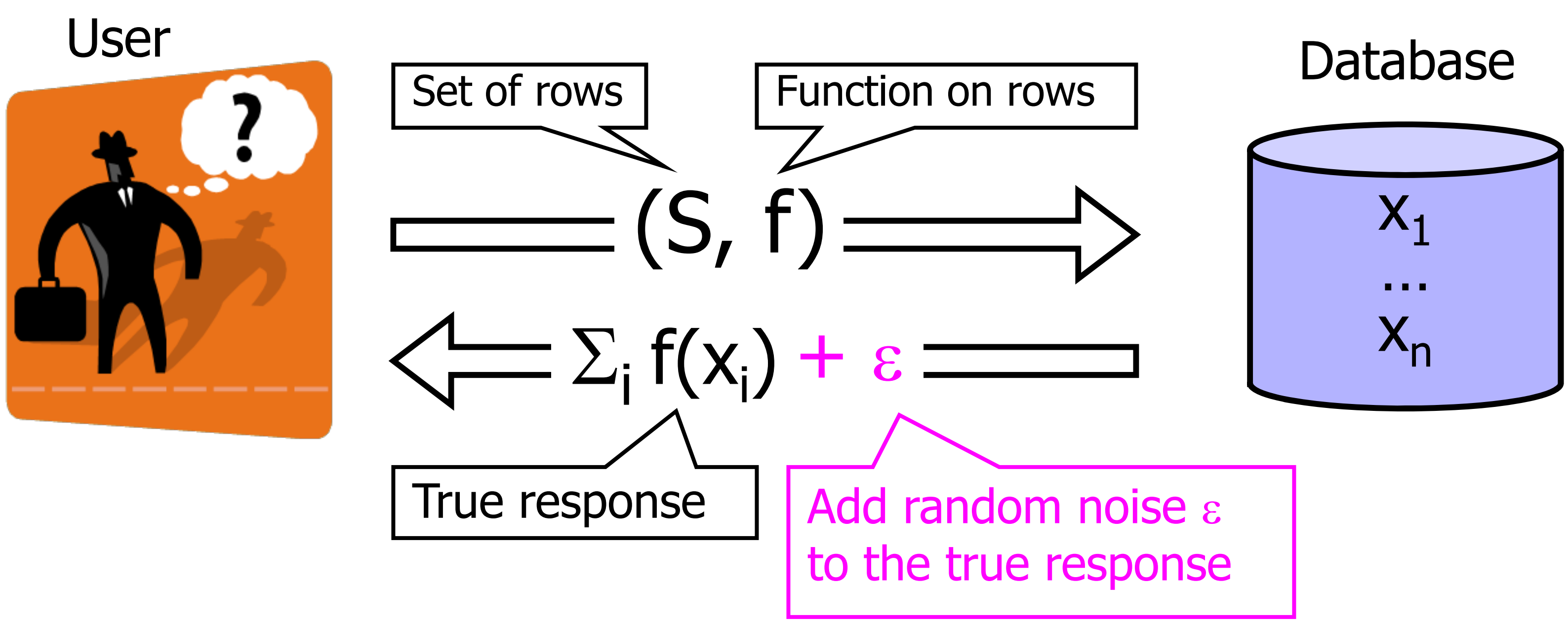
- Reveal entire database, but randomize entries



For example, if distribution of noise has mean 0, user can compute average of x_i

Output Perturbation

- Randomize response to each query



Concepts of Privacy

- Weak: no single database entry has been revealed
- Stronger: no single piece of information is revealed (what’s the difference from the “weak” version?)
- Strongest: the adversary’s beliefs about the data have not changed

Kullback-Leibler Distance

- Measures the “difference” between two probability distributions

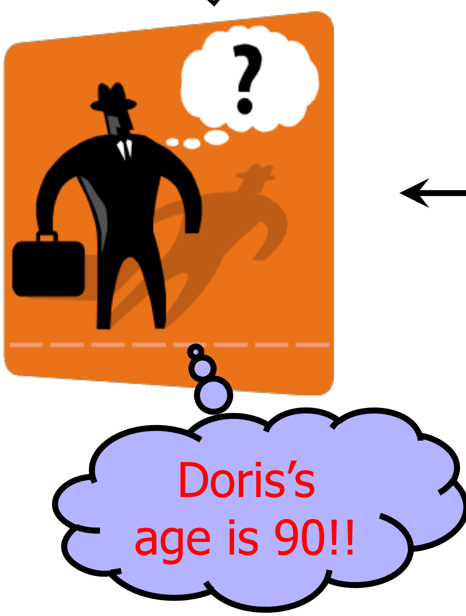
$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Privacy of Input Perturbation

- X is a random variable, R is the randomization operator, Y=R(X) is the perturbed database
- Naïve: measure mutual information between original and randomized databases
  - Average KL distance between (1) distribution of X and (2) distribution of X conditioned on Y=y
  - $E_y\,(KL(P_{X|Y=y} \,||\, P_x))$
    - Intuition: if this distance is small, then Y leaks little information about actual values of X
- Why is this definition problematic?

Input Perturbation Example

Age is an integer between 0 and 90



Gladys: 72
Doris: 110
Beryl: 85

Name: Age database

Gladys: 85
Doris: 90
Beryl: 82

Randomize database entries by adding random integers between -20 and 20

Randomization operator has to be public (why?)

Privacy Definitions

- Mutual information can be small on average, but an individual randomized value can still leak a lot of information about the original value
- Better: consider some property $Q(x)$
  - Adversary has a priori probability P_i that $Q(x_i)$ is true
- Privacy breach if revealing $y_i=R(x_i)$ significantly changes adversary's probability that $Q(x_i)$ is true
  - Intuition: adversary learned something about entry x_i (namely, likelihood of property Q holding for this entry)

Example

- Data: $0 \leq x \leq 1000$, $p(x=0)=0.01$, $p(x \neq 0)=0.00099$
- Reveal $y=R(x)$
- Three possible randomization operators R
  - $R_1(x)$ = x with prob. 20%; uniform with prob. 80%
  - $R_2(x) = x+\xi \bmod 1001$, ξ uniform in [-100,100]
  - $R_3(x) = R_2(x)$ with prob. 50%, uniform with prob. 50%
- Which randomization operator is better?

Some Properties

- $Q_1(x)$: $x=0$; $Q_2(x)$: $x \notin \{200, ..., 800\}$
- What are the a priori probabilities for a given x that these properties hold?
  - $Q_1(x)$: 1%, $Q_2(x)$: 40.5%
- Now suppose adversary learned that $y=R(x)=0$. What are probabilities of $Q_1(x)$ and $Q_2(x)$?
  - If $R = R_1$ then $Q_1(x)$: 71.6%, $Q_2(x)$: 83%
  - If $R = R_2$ then $Q_1(x)$: 4.8%, $Q_2(x)$: 100%
  - If $R = R_3$ then $Q_1(x)$: 2.9%, $Q_2(x)$: 70.8%

Privacy Breaches

- $R_1(x)$ leaks information about property $Q_1(x)$
  - Before seeing $R_1(x)$, adversary thinks that probability of $x=0$ is only 1%, but after noticing that $R_1(x)=0$, the probability that $x=0$ is 72%
- $R_2(x)$ leaks information about property $Q_2(x)$
  - Before seeing $R_2(x)$, adversary thinks that probability of $x \notin \{200, ..., 800\}$ is 41%, but after noticing that $R_2(x)=0$, the probability that $x \notin \{200, ..., 800\}$ is 100%
- Randomization operator should be such that posterior distribution is close to the prior distribution for <u>any</u> property

Privacy Breach: Definitions

[Evfimievski et al.]

- Q(x) is some property, ρ_1, ρ_2 are probabilities
  - ρ_1~"very unlikely", ρ_2~"very likely"
- Straight privacy breach:

  $P(Q(x)) \le \rho_1$, but $P(Q(x) \mid R(x)=y) \ge \rho_2$

  - Q(x) is unlikely a priori, but likely after seeing randomized value of x
- Inverse privacy breach:

  $P(Q(x)) \ge \rho_2$, but $P(Q(x) \mid R(x)=y) \le \rho_1$

  - Q(x) is likely a priori, but unlikely after seeing randomized value of x

Transition Probabilities

- How to ensure that randomization operator hides every property?
  - There are $2^{|X|}$ properties
  - Often randomization operator has to be selected even before distribution P_x is known (why?)
- Idea: look at operator's transition probabilities
  - How likely is x_i to be mapped to a given y?
  - Intuition: if all possible values of x_i are equally likely to be randomized to a given y, then revealing $y=R(x_i)$ will not reveal much about actual value of x_i

Amplification

[Evfimievski et al.]

- Randomization operator is γ-amplifying for y if

$$\forall\, x_1, x_2 \in V_x : \frac{p(x_1 \to y)}{p(x_2 \to y)} \le \gamma$$

- For given ρ_1, ρ_2, no straight or inverse privacy breaches occur if

$$\frac{\rho_2}{\rho_1} \frac{(1 - \rho_1)}{(1 - \rho_2)} > \gamma$$

Amplification: Example

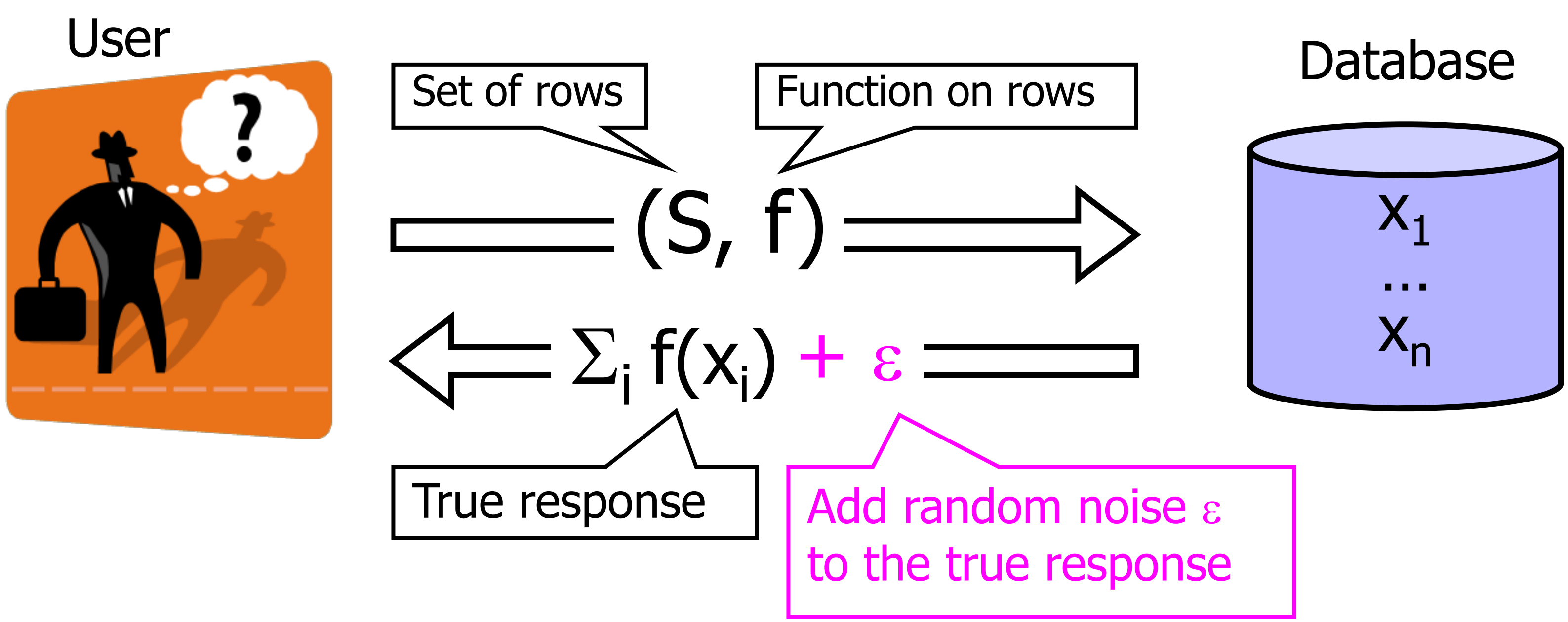
- For example, for randomization operator R_3,

  $p(x \rightarrow y) = ½ (1/201 + 1/1001)$ if $y \in [x-100, x+100]$

  $= 1/2002$ otherwise
- Fractional difference = $1 + 1001/201 < 6$ (= γ)
- Therefore, no straight or inverse privacy breaches will occur with $\rho_1=14\%$, $\rho_2=50\%$

Output Perturbation Redux

- Randomize response to each query



Formally…

- Database is n-tuple $D = (d_1, d_2 \ldots d_n)$
  - Elements are not random; adversary may have a priori beliefs about their distribution or specific values
- For any predicate $f: D \rightarrow \{0,1\}$, $p^{i,f}(n)$ is the probability that $f(d_i)=1$, given the answers to n queries as well as all other entries d_j for $j \neq i$
  - $p^{i,f}(0)$=a priori belief, $p^{i,f}(t)$=belief after t answers
  - Why is adversary given all entries except d_i?
- $conf(p) = \log p / (1-p)$
  - From raw probability to "belief"

Privacy Definition Revisited

[Blum et al.]

- Idea: after each query, adversary's gain in knowledge about any individual database entry should be small
  - Gain in knowledge about d_i as the result of $(n+1)^{st}$ query = increase from $conf(p^{i,f}(n))$ to $conf(p^{i,f}(n+1))$
- (ε,δ,T)-privacy: for every set of independent a priori beliefs, for every d_i, for every predicate f, with at most T queries

$$\Pr[conf(p_T^{i,f}) - conf(p_0^{i,f}) > \varepsilon] \le \delta$$

Limits of Output Perturbation

- Dinur and Nissim established fundamental limits on output perturbation (PODS 2003)

  … The following is less than a sketch!
- Let n be the size of the database (# of entries)
- If $O(n^{½})$ perturbation applied, adversary can extract entire database after poly(n) queries
- …but even with $O(n^{½})$ perturbation, it is unlikely that user can learn anything useful from the perturbed answers (too much noise)

The SuLQ Algorithm

[Blum et al.]

- The SuLQ primitive
  - Input: query (predicate on DB entries) $g: D \rightarrow [0,1]$
  - Output: $\Sigma\, g(d_i) + N(0,R)$
    - Add normal noise with mean 0 and variance R to response
- As long as T (the number of queries) is sub-linear in the number of database entries, SuLQ is (ε,δ,T)-private for $R > 8T\log^2(T/\delta)/\varepsilon^2$
  - Why is sublinearity important?
- Several statistical algorithms can be computed on SuLQ responses

Computing with SuLQ

- k-means clustering
- ID3 classifiers
- Perceptron
- Statistical queries learning
- Singular value decomposition

- Note: being able to compute the algorithm on perturbed output is not enough (why?)

k-Means Clustering

- Problem: divide a set of points into k clusters based on mutual proximity
- Computed by iterative update
  - Given current cluster centers $\mu_1, \ldots, \mu_n$, partition samples $\{d_i\}$ into k sets $S_1, \ldots, S_n$, associating each d_i with the nearest μ_j
  - For $1 \le j \le k$, update $\mu'_j = \Sigma_{i \in S_j} d_i \,/\, |S_j|$
- Repeat until convergence or for a fixed number of iterations

Computing k-Means with SuLQ

- Standard algorithm doesn't work (why?)
- Have to modify the iterative update rule
  - Approximate number of points in each cluster S_j

    $\underline{S}'_j$ = SuLQ($f(d_i)=1$ iff $j=\arg\min_j ||m_j-d_i||$)
  - Approximate means of each cluster

    $\underline{m}'_j$ = SuLQ($f(d_i)=d_i$ iff $j=\arg\min_j ||m_j-d_i||$) / $\underline{S}'_j$
- Number of points in each cluster should greatly exceed $R^{1/2}$ (why?)

ID3 Classifiers

- Work with multi-dimensional data
  - Each datapoint has multiple attributes
- Goal: build a decision tree to classify a datapoint with as few decisions (comparisons) as possible
  - Pick attribute A that "best" classifies the data
    - Measure entropy in the data with and without each attribute
  - Make A root node; out edges for all possible values
  - For each out edge, apply ID3 recursively with attribute A and "non-matching" data removed
  - Terminate when no more attributes or all datapoints have the same classification

Computing ID3 with SuLQ

- Need to modify entropy measure
  - To pick best attribute at each step, need to estimate information gain (i.e., entropy loss) for each attribute
    - Harder to do with SuLQ than with raw original data
  - SuLQ guarantees that gain from chosen attribute is within Δ of the gain from the actual "best" attribute.
- Need to modify termination conditions
  - Must stop if the amount of remaining data is small (cannot guarantee privacy anymore)