Comp430

Lecture 3 :

Microdata: Detailed data at the granularity of indiviuals.

Macrodata: The data which is used to train machine learning models.It is not about indiviuals.

PII = Personal Identifying Information

Pseudonymization: Removing PII from the data

Linkage attack

- Governor case
- AOL search log
- Netflix Scandal

Yes, removal of PII is sufficient. But it is not guaranteeing privacy.

Anonymization techniques were proposed to overcome this

k-anonymity,l-diversity,t-closeness

Explicit Identifiers: Information which can specific for the user

ssn,

Quasi Identifiers: Not unique for user, but combination of them can help to detect the user

zip, age, nationality

Sensitive Attribute: What we want to hide from the adversary

disease

K-anonimity

For people who know the quasi-identifiers of the person and want to linked with the db we are using k-anonimity to add an security layer.

We are grouping the users with the same quasi-identifiers. There must exists at least k person on each group. So if the linkage attack occurs, adversary cannot decide which person is the one in the group.

Each records should be indistinguishable in terms of QI's within its group

These groups are called **equivalence classes**

parameter k is the degree of anonymity

k is positive integer, higher is higher anonymity

Attacks on k-anonymity

Homogeneity attack : If sensitive values in an equivalene class are same or lack of diversity we can get information. K-anonymity says nothing about how SI's should be classified.

Background Knowledge Attack: If attacker knows about the other records in the equivalance class it can deduce the other ones.

l-diversity

In the equvialence class we need l-diverse Sensitive attribute. It implies the k diversity because we are talking about equivalence classes.

Distinct l-diversity

That should have l different values in the each Equivalence class.

Probabilistic inference attack: l diversity still gives security issues because the probability of distribution of the the sensitive informations can be uneven.

The solution :

Entropy l-diversity:

Each EC must have not only l different SA, it should also distributed even enough.

To compute evennes per EC.

- Let $p_{(q^*,s)}$ be the fraction of records in an EC q* with SA value equal to s.
  - $p_{(q^*,s)} \in [0,1]$
- A dataset D is entropy l-diverse if for every q*:

$$\underbrace{-\sum_{s \in S} p_{(q^*,s)} \log(p_{(q^*,s)})}_{\text{Entropy of } q^*} \geq \log(\ell)$$

He done couple of examples about the entropy. The cases with most even cases, resulted with bigger l.

Lecture 4:

There are couple of definitions about l diversity. If you choose distinct l diversity definition l is calculated by how many distinct elements. If you choose entropy l diversity definition to define l than you have to do the calculation to measure l.

Recursive (c,l) - diversity

Determine how many times the Sl is occured. Then sort them to decreased order of number of their occurrences in Equivalence class.Then call highest one with r1, and the other ones r2, r3 as continued.

r1 should be less than, sum of the other r's multiplied with constant c.But in the right side of the r's, it is l decide that which one that should we start. For example if l is 4 we should start from r4 and continue to the last term.

The most occurring one should be occur less than the constant c and the summation of others. That means that most frequent sensitive attribute value should not be too frequent

Example1

Let say we have 3 flu, 1 shingles and 2 acne in our EC.

For what smallest value of c is this equivalence class (c,2) diverse?

Sorted order

Flu → n* =3 ⇒ r1

Acne → n* = 2 ⇒ r2

Shingles → n* = 1 ⇒r3

In the question l is given as 2.

3 < c(r2+r3) = 3< c(2+1) = 3<c*3

smallest c is 2 .

Example2

Let say we have 3 flu, 1 shingles and 2 acne in our EC.

For what smallest value of c is this equivalence class (c,3) diverse?

Sorted order

Flu → n* =3 ⇒ r1

Acne → n* = 2 ⇒ r2

Shingles → n* = 1 ⇒r3

In the question l is given as 3.

3< c(r3) = 3<c*1

smallest c is 1.

shouldnt be 4?

Example3

Say that we have c =2. What is the largest l that this EC satisfies (2,l) diversity?

r1<2(rl+r(l+1))..)

3< 2(rl+r(l+1))..)

3/2<(rl+r(l+1))..)

we have to put 2 to achieve this inequality. l=3 does’nt satistfy.

For the privacy smaller c and higher l should be seek. But there is a trade of between them.

What does c means ?

Does satisfy l diversity always beneficial ?

Skewness Atack

If a dataset with %99 hıv- and %1 hıv+, is not satisfy l diversity at first. For Alice, being hıv+ is %1 prob due to the population statistics. But if we are going to make this data to l = 2 diverse. Alice could be in a situation that %50 being hiv+. this is worse than for her to previous data.

Similarity Attack

Even if we have 3-diverse patient table for example, with the linkage, if the diseases are both stomach related, we can deduce information about Bob’s diseases.

l-diversity does not take into account the meaning of the Sl.

Therefore l diversity may not always to good to have.

It is diffucult to achieve.

For example: ın 10k records with %1 hiv+, %99 hiv- there are 10k*%1 = 100 people who have hiv+. to achieve 2 l diverse, you need 100 equvialance class. To find similar QI at 100 might be harder .

t-closeness

The aim is creating a EC that is similar to general population in SI sense.

For example the population that have %99 hıv-, %1 hıv+

The EC that has all of the people has hıv- and none of the people has hıv+ is actually more similar to general population than EC that %50 with hıv- and %50 with hiv+. So the first one is more desirable.

t-closeness Perspective

privacy is measured by the information gain of the attacker/observer.

Information Gain = Posterier Belief- Prior Belief

Prior : %99 hiv- , %1 hiv+

Posterier: %50 hiv+, %50 hiv-

If information gain is low, we have high privacy.

distance t should be close, which means lower in value to say more private. So it is different then l and k because in those higher k and l is better in terms of privacy.

- Q = distribution of SA in the whole table
- P_{q^*} = distribution of SA in equivalence class q*

- q* satisfies t-closeness if: $dist(Q, P_{q^*}) \leq t$
  - dist(.,.) calculates distance between two probability distributions
- A table satisfies t-closeness if all ECs q* in the table satisfy t-closeness.
- Is small t or large t better (how about k and l?)

- What's a good dist(.,.) function?

Calculating Distance

- P = (p_1, p_2, ..., p_m), e.g.: (0.1, 0.6, 0.3)
- Q = (q_1, q_2, ..., q_m), e.g.: (0.2, 0.4, 0.4)

- Option #1: Variational Distance

$$dist(P, Q) = \sum_{i=1}^{m} \frac{1}{2}|p_i - q_i|$$

- P = (p_1, p_2, ..., p_m), e.g.: (0.1, 0.6, 0.3)
- Q = (q_1, q_2, ..., q_m), e.g.: (0.2, 0.4, 0.4)

- Option #2: Kullback-Leibler Divergence

$$\text{dist}(P, Q) = \sum_{i=1}^{m} p_i \log \frac{p_i}{q_i}$$

- P = (p_1, p_2, ..., p_m), e.g.: (0.1, 0.6, 0.3)
- Q = (q_1, q_2, ..., q_m), e.g.: (0.2, 0.4, 0.4)

- Option #3: Earth Mover's Distance
  - Amount of "mass" that should be moved to make P become Q
  - Moving p_1 -> p_3 causes higher distance than p_1 -> p_2

Lecture 5

Salaries

10k 20k 30k 40k

Q = 0.25 0.25 0.25 0.25 ⇒ a distribution of the whole population

p1 = 0.50 0.50 0.0 0.0 ⇒ EC1

p2 = 0 0.50 0.50 0.0 ⇒ EC2

Option 1 Variational distance

dist(P,Q1) = 1/2 * (0.25 +0.25 + 0.25 + 0.25) = 0.5

dist(P,Q2) = 1/2 * (0.25 +0.25 + 0.25 + 0.25) = 0.5

What is the problem ?

In terms of t closeness, variational distance says that both p1 and p2 are equal. But actually, p1 gives more information about the ec. In terms of whole population average, salary of random people is 25k. People chosen from the EC2 also has a average salary info. But people who has chosen from EC1 has less salary than average. Therefore like similarity attack we can deduct about pay range. Our information gain is much higher than the EC2. Therefore, logically p2 has lower t closeness level, conversely variational distance method suggests.

But with the earth mover distance, to make it similar to Q we have much more movement in q1. therefore it has more t-closeness than q2.

Conclusion

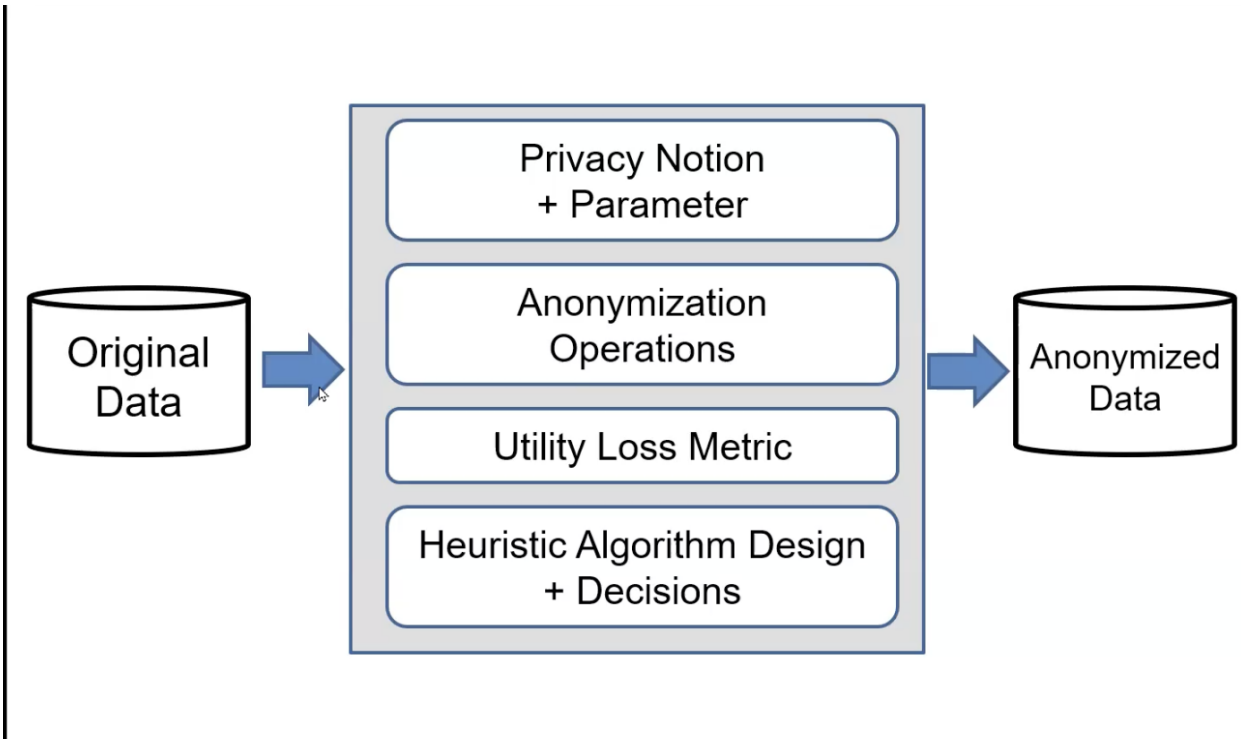
- Removing PII from microdata is not sufficient
  - Vulnerable to linkage attacks
  - Even without EIs, Quasi-Identifiers (QIs) can be used to re-identify individuals
- k-anonymity enforces that each individual must be in an EC of size $\geq$ k
  - An EC consists of QI-wise identical records
  - Stops linkage attack (identity disclosure)
  - But still vulnerable to sensitive attribute disclosure
    - Homogeneity attack – lack of SA diversity
    - Background knowledge attack

- l-diversity fights against SA disclosures
  - Each EC must contain l well-represented SA values
  - Distinct l-diversity, entropy l-diversity, recursive (c,l)-diversity, … that stem from different interpretations of "well-represented"
- Criticisms of l-diversity:
  - Not always necessary: no need to protect HIV-
  - Not always desirable: 2-diverse but 50% HIV+, 50% HIV- leaks a lot of information (skewness attack)
  - Does not consider SA semantics: similarity attack (all salaries low, all diseases related to stomach)
  - Can be difficult to achieve on skewed datasets

- t-closeness is another way to fight SA disclosures
  - Privacy is inverse of "information gain"ed from table
    - Information gain = Adversary's posterior versus prior belief
    - Low information gain is desired
  - Let Q denote the general SA distribution, P denote the SA distribution in a certain EC, then dist(P,Q) should be small – information gain is small
  - If dist(P, Q) $\leq$ t for all ECs, then the table satisfies t-closeness
  - How to measure dist?
    - Variational distance, Kullback-Leibler, EMD, …

Zip	Age	Nationality	Disease
13053	28	Russian	Heart
13068	29	American	Heart
13068	21	Japanese	Flu
13053	23	American	Flu
14853	50	Indian	Cancer
14853	55	Russian	Heart
14850	47	American	Flu
14850	59	American	Flu

Zip	Age	Nationality	Disease
130**	<30	*	Heart
130**	<30	*	Heart
130**	<30	*	Flu
130**	<30	*	Flu
1485*	>40	*	Cancer
1485*	>40	*	Heart
1485*	>40	*	Flu
1485*	>40	*	Flu



Privacy Notion + Parameter

We have covered this.

k-anonymity: parameter k , l-diversity : parameter: l , t closeness: parameter t

Anonymization Operations

Operations that are used to **preserve the truthfulness but reduce precision of data**

Generalizations: Instead of saying 23 years old generalize this by saying <30 years old

Suppression: Change with ** to the specific piece of the information

Supression is like generalize as maximum amount as possible.

Generalization or suppresion of the one EC doesnt mean that you generalize all the others.

Replace specific QI values with more general values until we get k identical values.

Numerical

Start with least significant parts of information

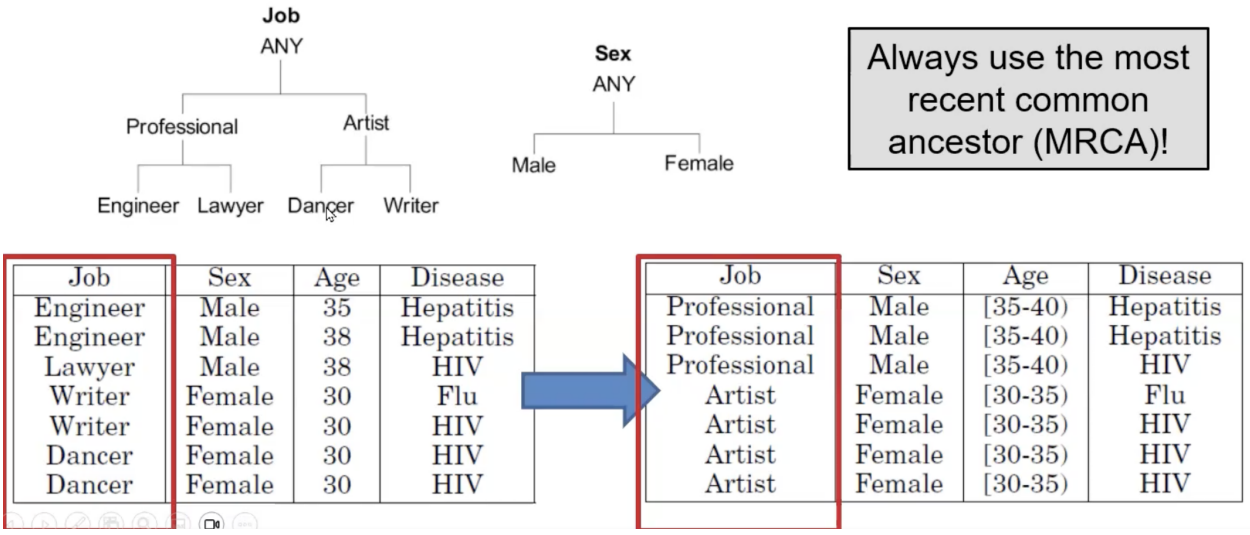
Example: ages{36,39,38} → 3* or [35,40)

Example : zipcode 12345 → 1234* → 123**

Categorical

Use a Domain Generalization Hiearchy (DGH)

Similar to taxonomy tree. It is given to us.



Job	Sex	Age	Disease
Engineer	Male	35	Hepatitis
Engineer	Male	38	Hepatitis
Lawyer	Male	38	HIV
Writer	Female	30	Flu
Writer	Female	30	HIV
Dancer	Female	30	HIV
Dancer	Female	30	HIV

Job	Sex	Age	Disease
Professional	Male	[35-40)	Hepatitis
Professional	Male	[35-40)	Hepatitis
Professional	Male	[35-40)	HIV
Artist	Female	[30-35)	Flu
Artist	Female	[30-35)	HIV
Artist	Female	[30-35)	HIV
Artist	Female	[30-35)	HIV

💡 You can keep age as 30 in the second EC

For example if we have engineer engineer engineer, you dont have to make professional. Always use the most recent common ancestor.

Full Domain vs Local

Full Domain

Generalize all values of a QI to the same DGH level across all records in DB

Every occurence of 12345 is replaced with 1234*

Practical usage + ML is easier

Local Domain

Values can exist in different levels of the DGH, also, same value can be generalized to different levels in the different ECs.

12345 in one EC, 1234→ 1234* in one EC, 12345 → 123** in another EC

Utility loss is typically lower → less lose of data

Confuse the model in ML

Operations that **hurt truthfulness** are not preferred

Perturbation (e.g noise addition): instead of saying 25 years old says 27 years old

Swapping: swapping the age informations

Adding synthetic records: fake records

Utility Loss Metric

Trade of between privacy and utility.

We need to minimize utility loss while achieving k-anonymity > Dont over generalize or over-suppres “Principle of minimal distortion”

How to measure utility loss (information loss) ?

Distortion metric (MD)

Charge penalty to each instance that is generalized or suppressed.

Each generalizition of one level charged +1 penalty unit. Sum over all records and all atributes.

Lecture 6

Distortion metric does not consider that branching factors and depths of DGHs can be different.For example Workclass DGH government can branch 3 thing and self-employed can be branch 2 things. In terms of MD because they are in the same hierarchy, generalizing from their subclasses to them have same impact. But we now that government has more ambiguity to guess which is the one from 3 of them when we compared to self-employed.

To address this issue we have **Loss Metric**

Loss Metric

Takes into account the branching factor + depth

LM for any value = (# of descendant leaves of val -1)/ (total # of leaves in DGH -1)

💡 Every field has itself for descendant

Examples

Total number of leaves in DGH is 8. We are counting from the below.

State-gov + local-gov + federal-gov = government which count as =3

private itself is counted as = 1

self employed counted as = inc + not inc =2

unemployed counted as = without pay + never worked = 2

3+1+2 +2 = 8

Example Lm for private

1-1/8-1 = 0

makes sense because it is not branching.We dont lose anything

Example Lm for workclass

8-1/ 8-1 = 1

makes sense because it is root. If we lost it we lose all info.

Example Lm for government

3-1/ 8-1 = 2/7

Example LM for unemployed

2-1/8-1 = 1/7

Loss Metric is a way to compute single cell, how do we compute full data.

LM cost of a record

LM(rec) = sum of all w(x)*LM(x)

The coefficients are introduced as weights. ⇒ w(x)

It is also answer to how can we give more penalty to some generalization from the other ones by weights.

> 💡 As a rule of thumb, the weights that you summed them of is equal to 1.

LM cost of a table:

LM(T) = sum LM(rec)

- **Loss Metric (LM)**
  - Takes into account the branching factor + depth
  - LM cost of one value (one cell in the table):

$$\mathrm{LM}(\mathrm{val}) = \frac{\#\ \text{of descendant leaves of val} - 1}{\text{total \# of leaves in DGH} - 1}$$

  - LM cost of a record:
    - W_{attr}: custom weight of each attribute, e.g., w_{age}, w_{job}

$$\mathrm{LM}(\mathrm{rec}) = \sum_{\mathrm{val} \in \mathrm{rec}} w_{attr} * \mathrm{LM}(\mathrm{val})$$

  - LM cost of a table: $\mathrm{LM}(T) = \sum_{\mathrm{rec} \in T} \mathrm{LM}(\mathrm{rec})$

Example

W(workclass) = 0.6 and W(age) = 0.4

	Workclass	Age	Disease
record 1:	Government	30-35	HIV
record 2:	Private	30-40	Asthma

DGA for age.

30-40

30-35 35-40

30 31 32 33 34 35 36 37 38 39

LM(dataset) = LM(record1) + LM(record2)

LM(record1) = 0.6* LM(Government) + 0.4* LM(30-35)

= 0.6∗2/7 + 0.4∗4/9 = 0.35

LM(record2) = 0.6* LM(Private) + 0.4* LM(30-40)

= 0.6∗0 + 0.4 ∗ 1 = 0.4

LM(dataset) = 0.35 + 0.4 = 0.75

Implementing these are the hard things. These are tree algorithms at the basic.

Statistical Metrics

- **Query answering on anonymized datasets**
  - How many records w/ zip=13053 and age=37?

Original dataset

Zip	Age	Nationality	Disease
13053	28	Russian	Heart
13068	29	American	Heart
13068	21	Japanese	Flu
13053	23	American	Flu
14853	50	Indian	Cancer
14853	55	Russian	Heart
14850	47	American	Flu
14850	59	American	Flu
13053	31	American	Cancer
13053	37	Indian	Cancer
13068	36	Japanese	Cancer
13068	32	American	Cancer

Answer = 1

Anonymized dataset

Zip	Age	Nationality	Disease
130**	<30	*	Heart
130**	<30	*	Heart
130**	<30	*	Flu
130**	<30	*	Flu
1485*	>40	*	Cancer
1485*	>40	*	Heart
1485*	>40	*	Flu
1485*	>40	*	Flu
130**	30-40	*	Cancer
130**	30-40	*	Cancer
130**	30-40	*	Cancer
130**	30-40	*	Cancer

For the original dataset answer is 1.

But for the anonymized dataset answer is trivial.

First choice the EC that can potentially satisfies this. In the picture it actually the last EC and it has 4 members

We masked 2 number in the zip code, for each number we have 10 possibility.We have 100 posibility and we are looking for 13053 so probability of having this 1/100.

30-40 there are 10 numbers. We are looking for 37 so 1/10 probability.

Anonymized answer:

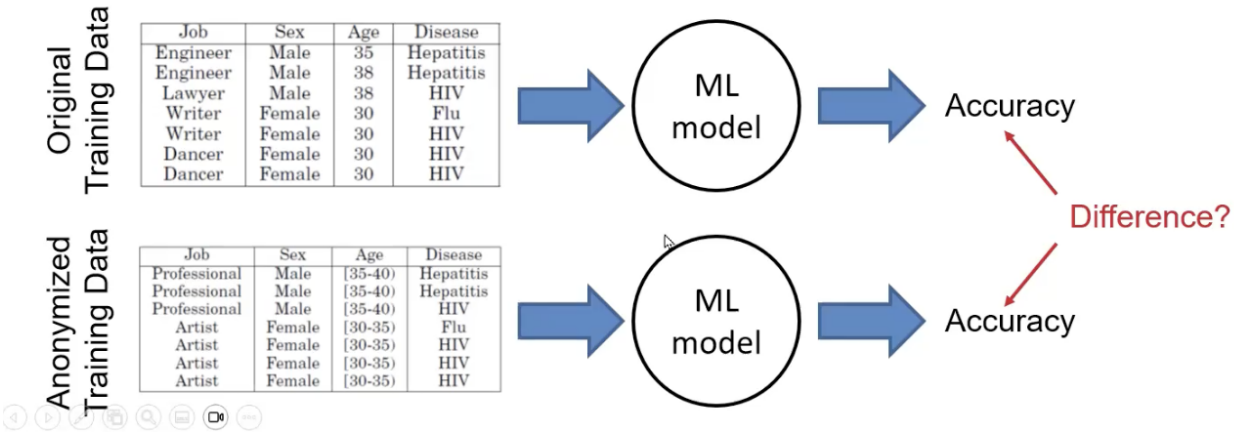
1/100 * 1/10 with 4 member = 0,004

Normal answer:

1

We can find the relative difference with calculated value with normal dataset with private one and divide it with normal one again

- **ML-Based Metrics**
  - Measures the utility loss in using the anonymized data in machine learning



Heuristic Algorithm Design + Decisions

Optimal k-anonymity is NP-hard.

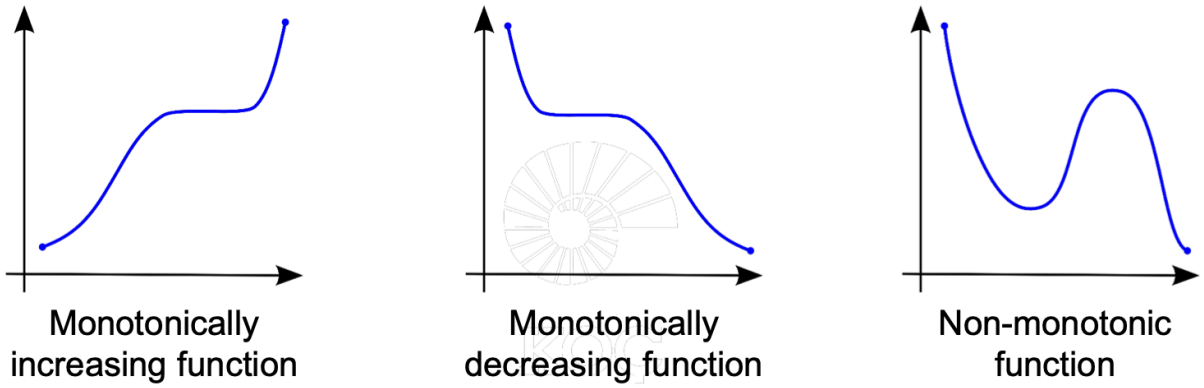
We have to rely on heuristic algorithms.

We should aim to put QI-wise similar records in each EC.

This is a challenge that should be done accurately and efficiently

Monotonocitiy

Is a notion of , how much things are changing. Monotonicly increasing means it is either increase or not changing. monotonically decreasing means it is decreasing or not changing. Non-monotonic functions can be increase or decrease the thing about them is we cannot be sure about their behavior in terms of increase or decrease.



POSSIBLE EXAM QUESTİON

full domain generalization + suppression of QIs

> 💡 In the example, ı get confused about the EC and counting k. But it is actually find the exact SI informations. So basically count the ones with exact SI's and that is k to that value. Even if not same for all we are just discovering the concept. While we are making more generalized we can see we can group more Sı's.

k anonymity is monotonicly increased as number of full domain generalizations increase.

Lets thing about like this. Count the groups that have the similar in QI-wise level. While we are making more full domain generalization we are accomplish bigger groups that have QI-wise similar. Of course it doesnt have to be increase, while we are generalize more, but is a possibility to getting bigger. So generalizing more, means that the groups that are QI-wise similar is either increasing, or staying the same in terms of k-anonymity. So it basically, monotonically increasing function.

What about l- diversity ?

If we are talking about distinct l diversity . l is also monotonicly increasing .

Why ?

So increasing the number of full-domain generalization, creating bigger groups of EC. So those groups has no possiblity of include the less amount of distinct diversity. It should be either increasing or staying the same.

But we cannot say the same thing about other l diversity defitinions such as entropy. It is no guarantee to increase , it is non-monotonic.

K anonymity not decrease when we are generalizing. EC's can only become larger via generalization.

Generalization and suppression are monotonic wrt k-anonymity and distinct l-diversity

Heuristic Approaches

There are 4 categories of heuristic k-anonymization approaches.

Bottom-up approach

Partitioning-based approach

Clustering-based approach

Top-Down approach

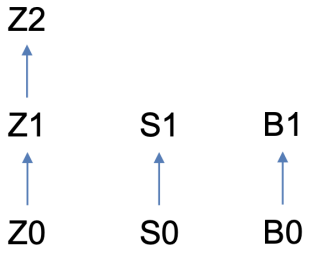
Some use full-domain generalization some use local.

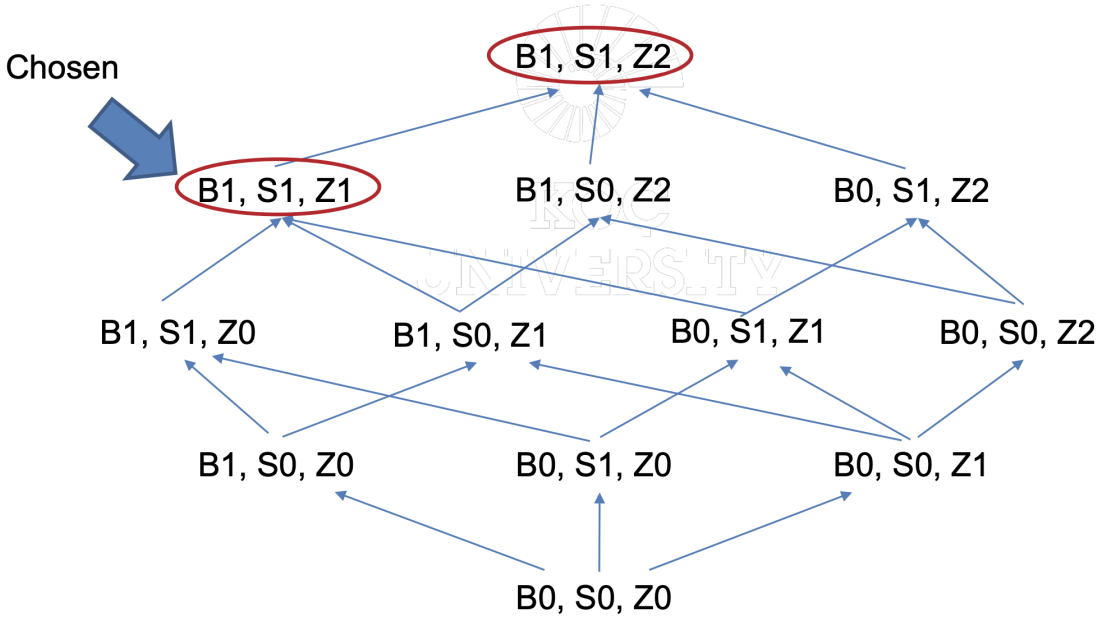
Bottom-Up Approach

Start with raw microdata, keep generalizing to figure out the **minimal generalization necessary** to achieve k-anonymity/ l- diversity.

Creates a **generalization lattice.**

- Say that we have 3 QIs: Z, S, B
  - Z has 2 levels in DGH
  - B and S have 1 level





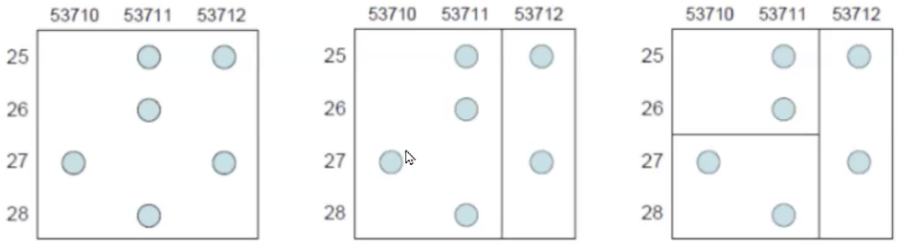
Generalization lattice

Lecture 7

Partitioning-Based Approach

Partitioning-Based Approach

- Think of data that is numeric or can be mapped to a multi-dimensional space
- Uses recursive, greedy partitioning of the space
- Partition(region,k)
  1) Choose a dimension according to some heuristic
  2) Partition the region according to that dimension into sub-regions R1 and R2 where |R1| ≥ k and |R2| ≥ k
  3) Return Partition(R1, k) U Partition(R2,k) //recursion

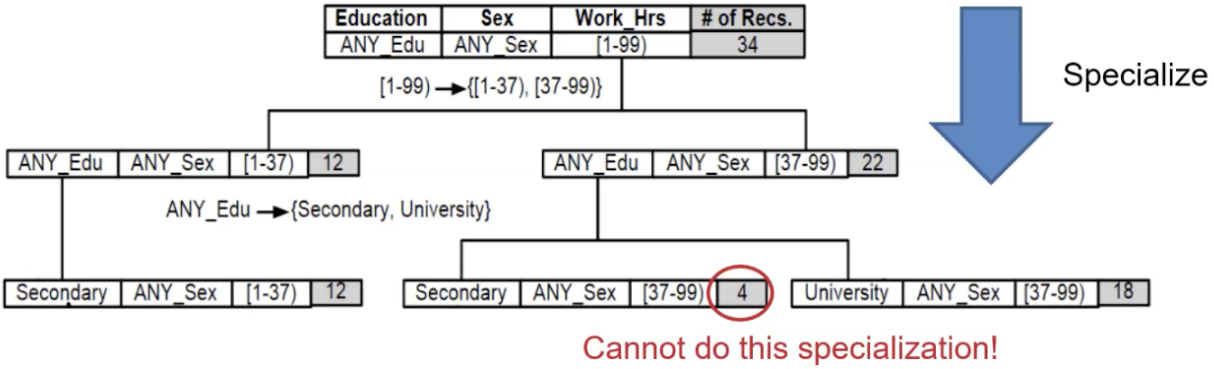


Lm cost of hypotetically putting r1 and r2 in the same cluster in the same EC.

Top Down Approach

Top-Down Approach

- Start with max generalization, keep specializing until k-anonymity is violated
  - Each specialization -> utility loss is reduced, but EC sizes become smaller
  - Monotonicity is a key enabling factor



Choosing the one with the fewest children might be advantegous approach.

local generalization

Non-Tabular Data

So far we anonymized tabular datasets.

But not all data is tabular,

How can we anonymized data such as , location data , graph data, hierarchical data, set-valued data ?

Lets start with set-valued data

It is often used for transactions. Dataset D , is distinct buyable items

/ = {i1,i2... ,in}

X denote an itemset (subset of buyable items)

supp(X): Number of transactions that contain X (**support** of X)

Trans. ID	Bought Items
t_1	milk, bread
t_2	butter
t_3	beer, diapers
t_4	milk, bread, butter
t_5	bread

X = {bread}

supp(X) = 3

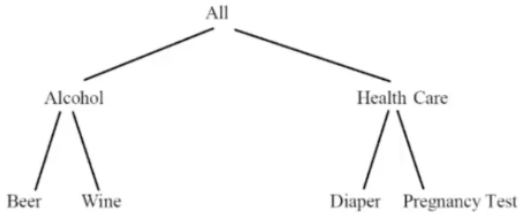
Y = {bread,butter}

supp(Y) = 1

Content and size

- k-anonymity for set-valued data needs to consider:
  - Transaction content and size
- Put T_1-T_2 in one EC, T_3-T_4 in one EC:

Owner	TID	Items Purchased
Alice	T_1	{Beer, Diapers}
Bob	T_2	{Wine, Diapers, Pregnancy Test}
Chris	T_3	{Beer, Wine, Pregnancy Test}
Dan	T_4	{Beer, Wine, Diapers, Pregnancy Test}



TID	Items Purchased
T_1	{Alcohol, Health Care}
T_2	{Alcohol, Health Care}
T_3	{Beer, Wine, Health Care}
T_4	{Beer, Wine, Health Care}

Is this the best anonymization we can have?

We could also keep diaper as it is with t1 and t2 and suppress pregnancy test. In the set-valued data, suppressing means deleting it.

k anonimty itself is not very well structured in order to obtain privacy in set list. For example, Jim and Bob example, if Bob sees the subset of the what jim has, it can easily find other items by distinguish the ones he sees.

Therefore we need to create a different anonymity notion to track it.

k^m Anonymity

Adversary know; at most m items of a transaction. Adversary wants to know the remaining ones. In this set up; no explicit SA, all items are sensitive by default.

K^M anonymity

Satisfies that if adversary knows up to m items, he cannot use this knowledge to identify less than k transactions in the dataset.

id	contents
t_1	$\{a_1, b_1, b_2\}$
t_2	$\{a_2, b_1\}$
t_3	$\{a_2, b_1, b_2\}$
t_4	$\{a_1, a_2, b_2\}$

Does this table satisfy 2^2-anonymity?

- k=2, m=2

No, check $\{a_1, b_1\}$:

- Adversary knows Bob bought $\{a_1, b_1\}$
- Bob must be transaction t_1
- Adversary learns Bob bought b_2

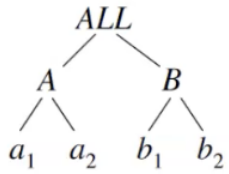
Sup{a1,b1,b2} = 1 < 2

so it does not satisfy the 2^2 anonymity.

---

id	contents
t_1	$\{a_1, b_1, b_2\}$
t_2	$\{a_2, b_1\}$
t_3	$\{a_2, b_1, b_2\}$
t_4	$\{a_1, a_2, b_2\}$

Generalize: $\{a_1, a_2\} \rightarrow A$



id	contents
t'_1	$\{A, b_1, b_2\}$
t'_2	$\{A, b_1\}$
t'_3	$\{A, b_1, b_2\}$
t'_4	$\{A, b_2\}$

Lets generalize it.

In the generalized version.

> 💡 A cannot be written twice in t4 for a1 and a2 because this is set representation.

Does this generalized version satisfy 2^2 anonymity ?

We need to check all supports. How many combinations can be done. m is given as 2 so we need a all subsets of 2's.

sup{A,b1} = 3

sup{A,b2} = 3

sup{b2,b1} = 2

3,3,2 ‘s are results of the supports. All of them are greater or equal to m value which is 2. Therefore we can say this satisfy 2^2 anonymity.

But it doesn't satisfy k = 3^2 anonymity

But i supports k = 2^3 anonymity.

sup(a,b1,b2) = 2

Lecture 8

Location Data

LBS means location based service, that users use on their phone.

Location should be dangerous information for adversary.

Threats

1)Identifying a user using the location in their LBS query

In order to overcome this, location cloaking technique is used. User submits a cloaked region when issuing a LBS query instead of their actual location.

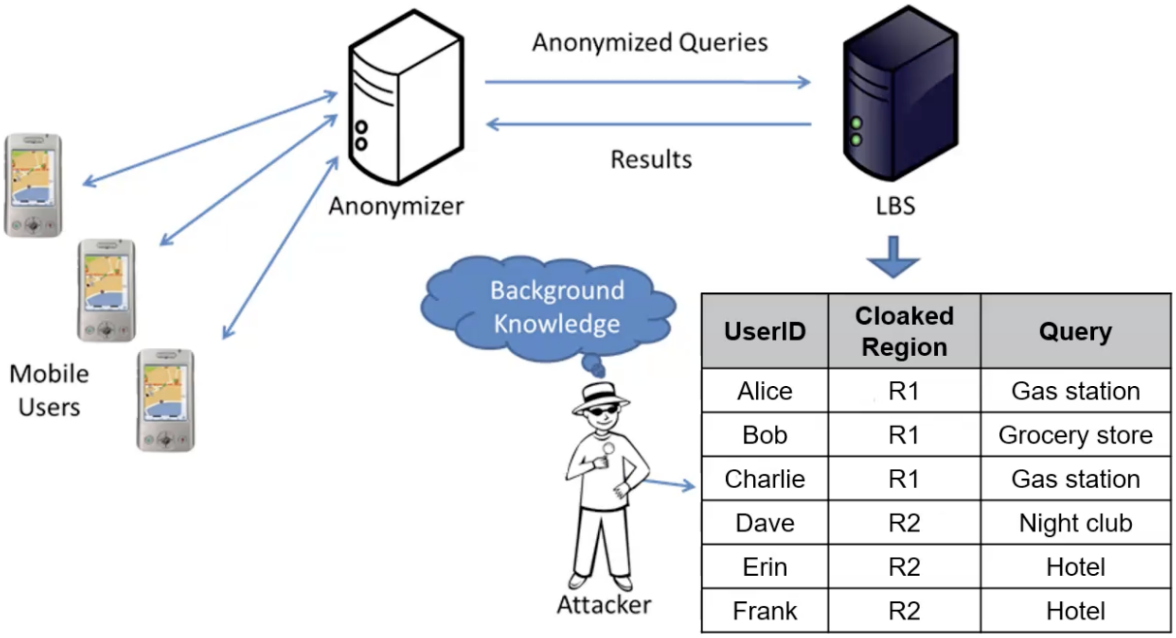
Submitted cloack region must contain at least k-1 other users. Which makes our k-anonymity definition.

Trade-offs

- Speed is a trade of because user must wait for k-1 other users
- Accuracy

- Cloaked regions are imprecise, we could loss in utility.

While k is getting larger to group more cloaked regions, speed and accuracy is getting more complicated to structure.



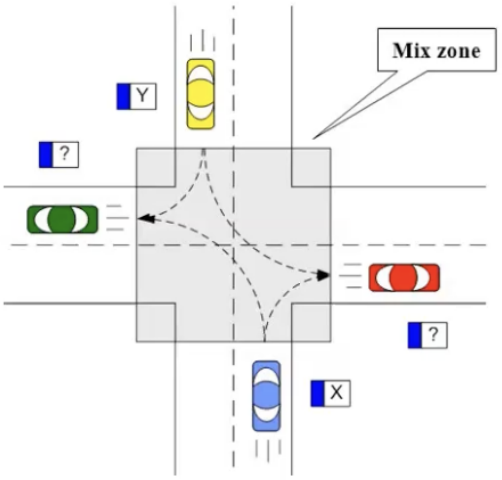
UserID	Cloaked Region	Query
Alice	R1	Gas station
Bob	R1	Grocery store
Charlie	R1	Gas station
Dave	R2	Night club
Erin	R2	Hotel
Frank	R2	Hotel

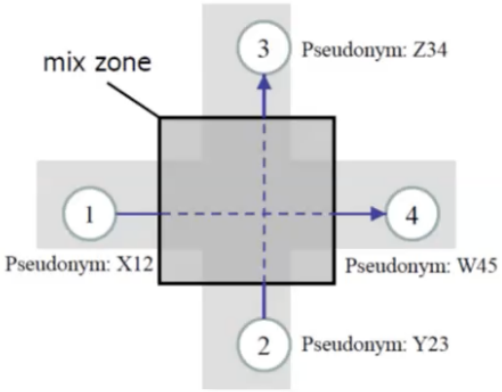
Explained why we need anonymizer

A need for trustable third party, but is this sustainable ?

2)Tracking a user considered as threats.

Assume users continuously use an LBS for the full duration of their mobility such as Navigation Service

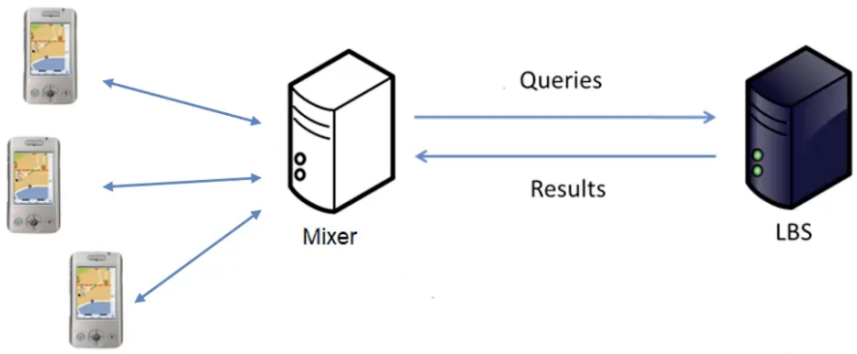




Continuity can be broken with mix zones that swaps the user's pseudonyms.

In the figure above we have 2 anonymity, in terms of k anonymity. 2 users are come in and 2 users are out, we are swapping their identity.

Size and shape of mix zone can be differ according to the need.



- Why do we need to mix multiple users at once?
  - **If we just change one user by keeping everyone else constant, server immediately knows it is that user's pseudonym that was changed.**
- Why the need for geographic proximity of mixed users?
  - **If we mix users that are in different parts of the city (e.g., one in Pendik, one in Sariyer) what happens? Based on previous and next location, server can easily infer who is who.**

Potential Attacks :

1) Timing Attacks

In some scenarios, adversaries can know specifically turning left or right took a long time than the other, or took a longer time than going in the same location. So in the mixed zone, they can track the pseudonimies by tracking the timing.

2)Attacks based on historical traffic patterns

For example in the long bridge, you are going in the same direction for a long time. Sudden changes in your direction can be deduced by the adversaries. So be careful about where you place the mix zones.

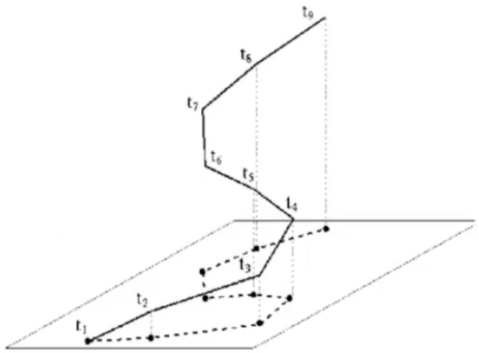
Trajectory Data

Trajectory Data

- **Trajectory** = time-ordered sequence of locations

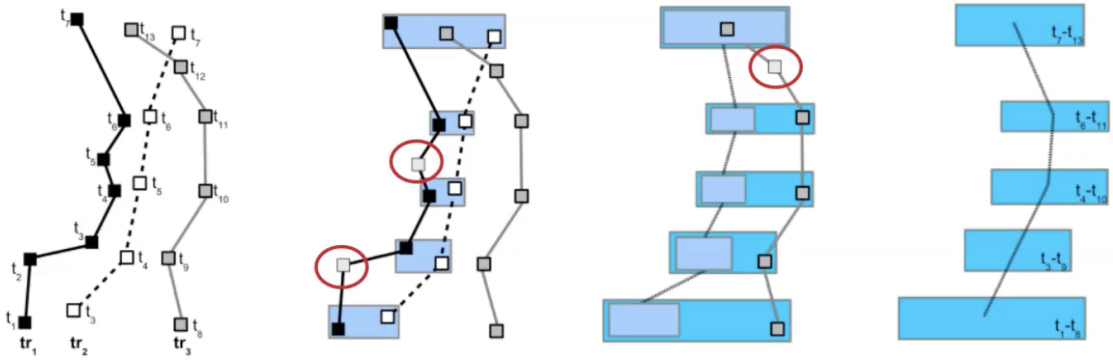
$$<(x_0, y_0, t_0), \ldots, (x_n, y_n, t_n)>$$

- x_i, y_i is the position coordinate relative to the origin
- t_i is the time stamp for the position information



Tid	position (x,y)	time (t)
1	48.890018 2.246100	08:25
1	48.890018 2.246100	08:26
...	...	...
1	48.890020 2.246102	08:40
1	48.888880 2.248208	08:41
1	48.885732 2.255031	08:42
...	...	...
1	48.858434 2.336105	09:04
1	48.853611 2.349190	09:05
...	...	...
2	...	...

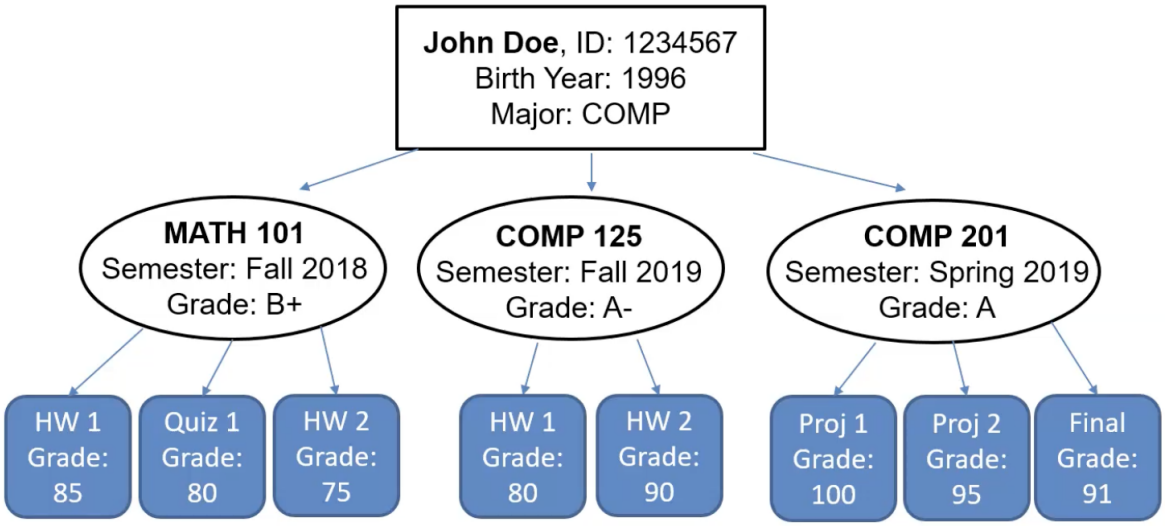
- Consider that we have a database of trajectories
  - Users' location traces have already been collected and stored in a database
  - We want to apply anonymization afterwards



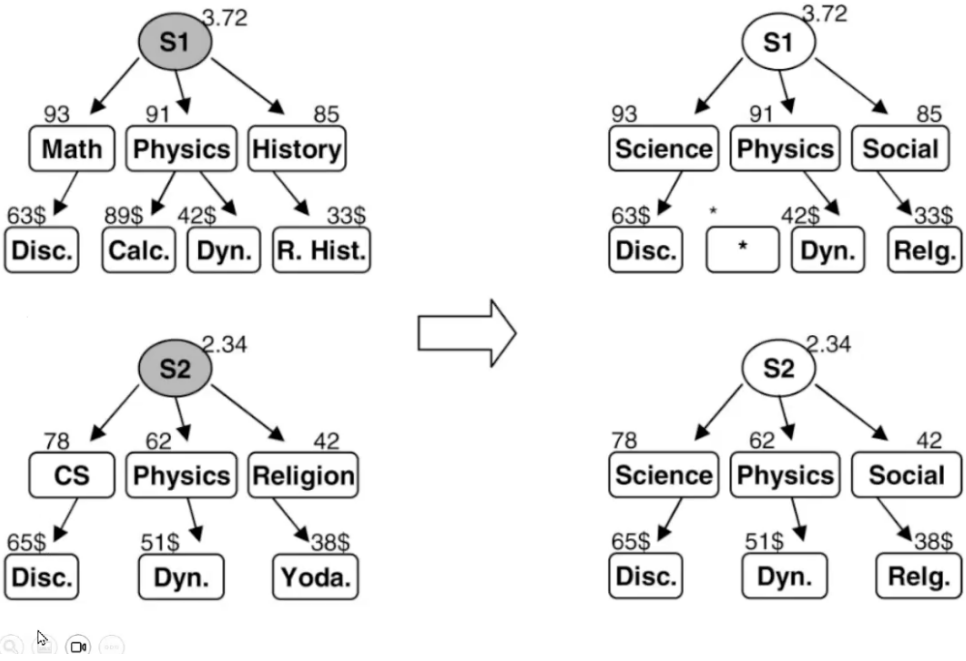
Actual Trajectories | Putting Tr1 and Tr2 in one EC | Adding Tr3 to the same EC | Final EC with 3 trajectories

Hierarchical Data

- "Tree-structured data"
- Became popular after XML, JSON data storage formats and document-oriented NoSQL databases

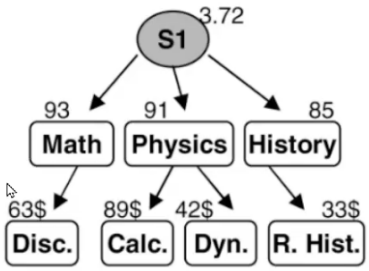


- Full 2-anonymity:



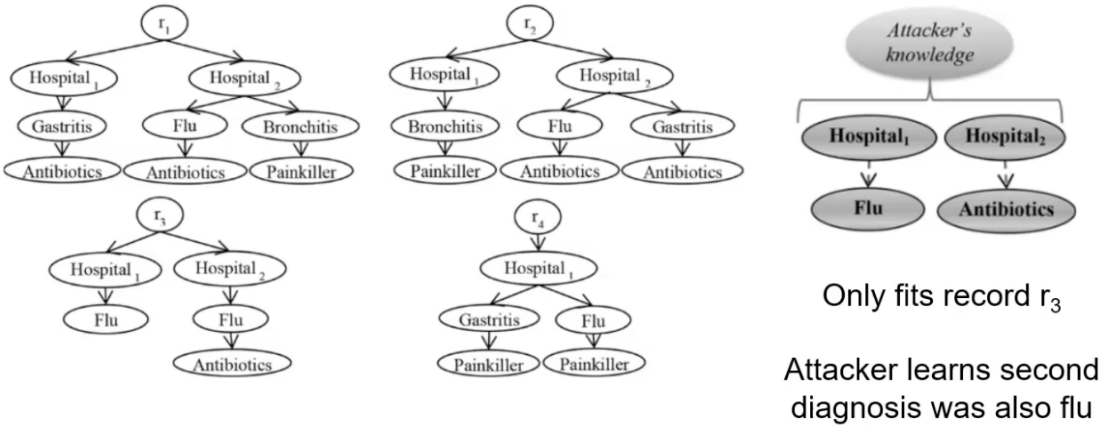
💡 Why we are suppressing the one with 2 books ? → Because if adversary knows that a person has two books, it can detect the person.

- We consider two scenarios



- **Scenario #1:**
  - Data comes with QIs + SAs
    - Course names + book names are QIs
    - Grades + price paid for books are SAs
  - Threat model: Adversary uses QIs to link person to record, thereby infer SAs
  - Solution: Apply full **k-anonymity** and **l-diversity** vertex-by-vertex

- **Scenario #2:**
  - Threat model: Adversary has partial knowledge of user's hierarchical record, wants to learn the rest
    - Similar to k^m-anonymity in set-valued data



In this example, it only be record 3 because it has flu on the hospital 1 and antibiotics on hospital 2.

What is solution for this ?

- Solution: **$k^{(m,n)}$-anonymity**
- A dataset is **$k^{(m,n)}$-** anonymous if any attacker who has knowledge of at most m node labels and n structural relationships (ancestor-descendant) is not able to link an individual to less than k records in the dataset.

m - node labels

n - ancestor descendant relationship

Where is the example ?

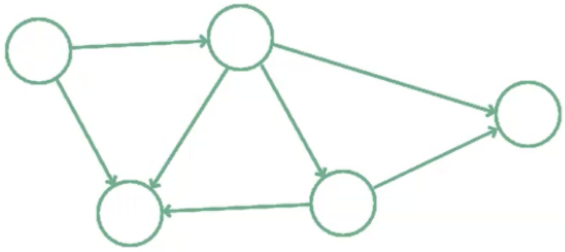
Graph Data

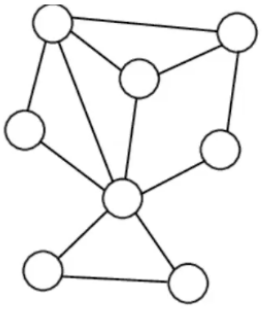
Grap G(V,E)

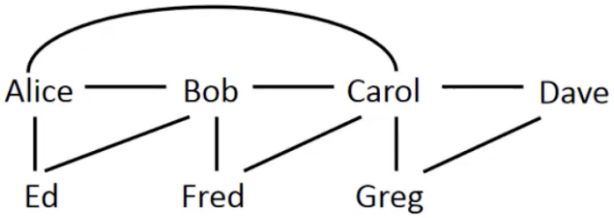
V is the set of vertices (nodes)

E is the set of edges

- Directed vs undirected graphs:
  - Directed: each edge has a direction
    - Instagram followers graph: Alice follows Bob but Bob doesn't follow Alice
  - Undirected: by default, edges are bidirectional
    - LinkedIn friends graph: Alice and Bob add each other to their network







Nodes

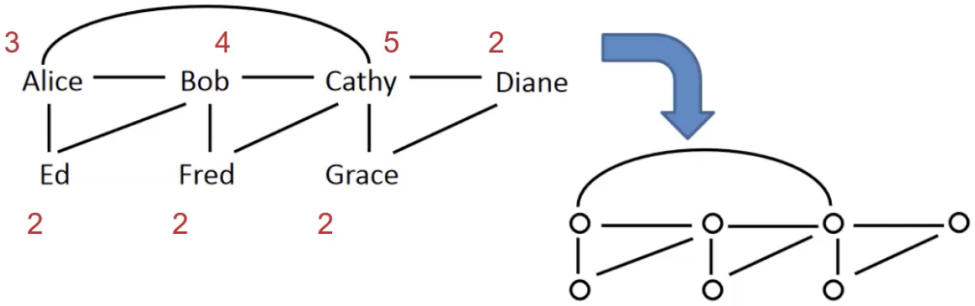
ID	Age	HIV
Alice	25	+
Bob	19	-
Carol	34	+
Dave	45	+
Ed	32	+
Fred	22	-
Greg	44	-

Edges

ID1	ID2
Alice	Bob
Alice	Carol
Alice	Ed
Bob	Carol
Bob	Ed
Bob	Fred
Carol	Dave
Carol	Fred
Carol	Greg
Dave	Greg

We can replace the names with random unnamed structures but negative and positive SI values stays.

- Replace names/EIs with random node IDs



- Alice and Cathy identify themselves based on degree.
- Together they can identify Bob and Ed.
- Thus they learn Bob and Ed are friends!

Naive anonymization doesn't work!

There are 3 threats

Node re-identification

- Attacker learns the anonymized node x corresponds to Alice

Edge disclosure

- Attacker learns two individuals Alice and Bob are connected.
- This type of thread happens from known induvial interaction

Sensitive property inference

- Attacker learns that Alice is HIV positive

Lecture 9

We can anonymized a graph G and can call it G*.

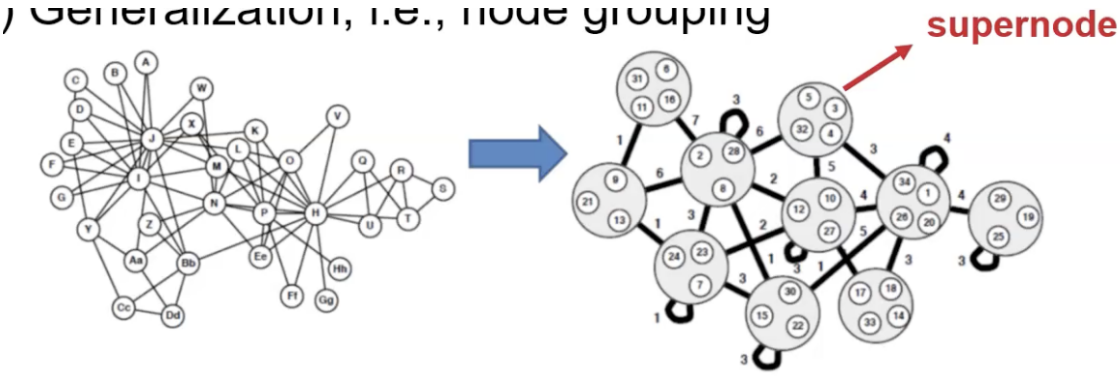
The goal is minimize difference (G,G*)

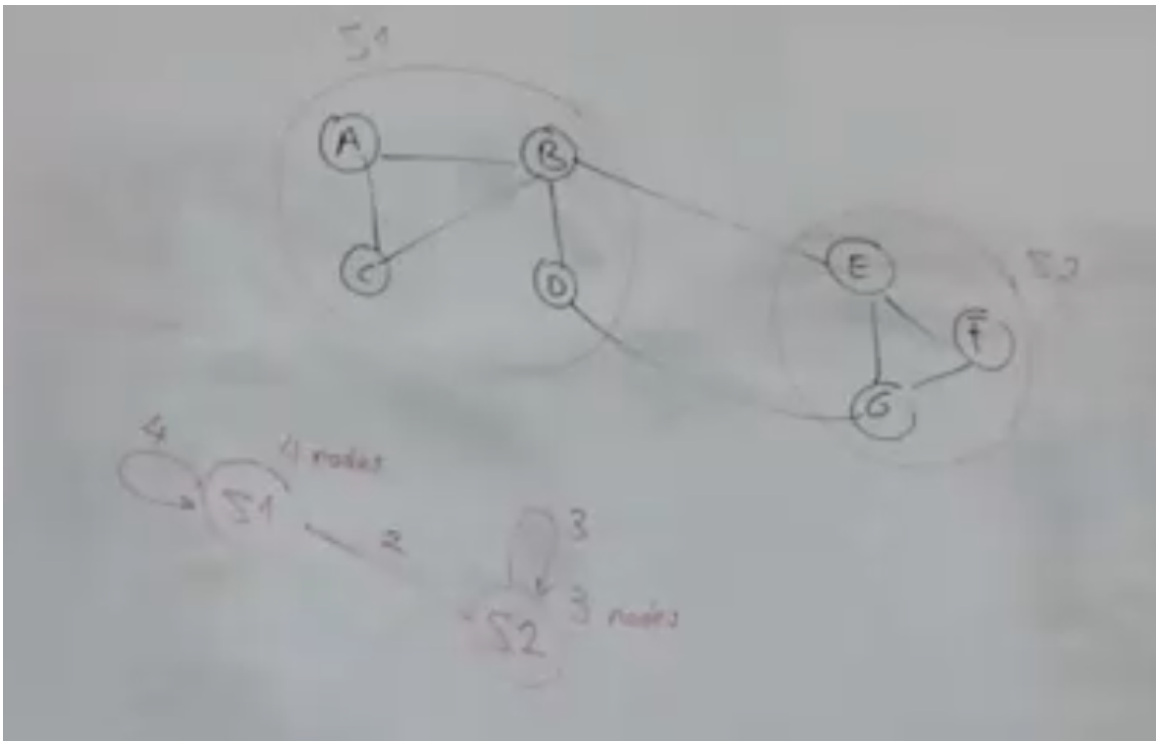
What can be done to anonymize

1) Adding/ remove edges

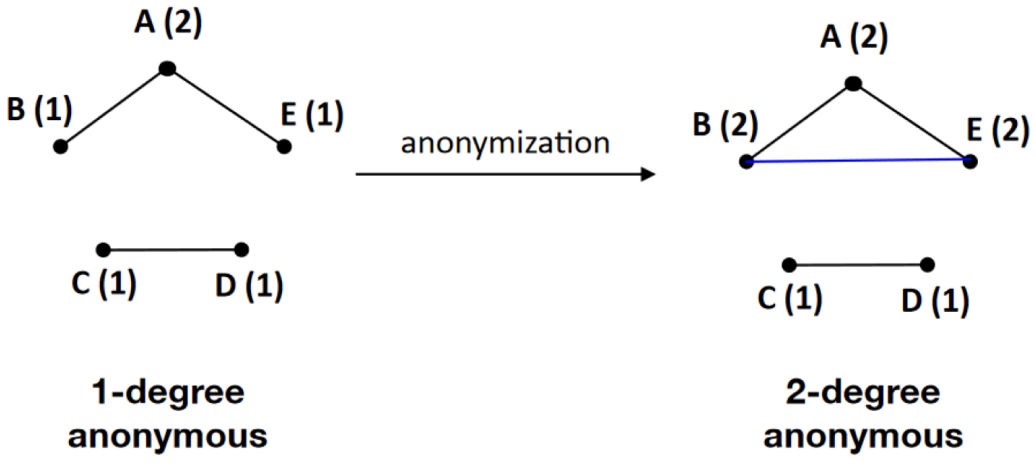
Straightforward

2)Generalization, ie node, grouping





k- degree anonymity



For example at the above example, B,E,C and D are 1 degree therefore it is 4 degree anonymous, but A is just has the one node that has 2 degree so it becomes 1-degree anonymous and make the whole graph one degree anonymous.Reason is this, we can deduct A, so it is bottleneck.

But if we combine B and E, math is like this.

A , B and E has 2 degree therefore they are 3- degree anonymous.

C and D has 1 degree so they are 2- degree anonymous

Therefore the graph itself is 2-degree anonymous because it is a bottleneck.

> 💡 Challenge: Not always possible to realize a degree sequence through edge additions.

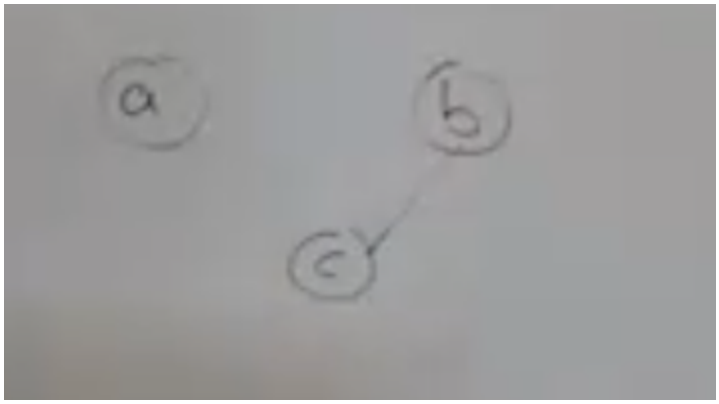
For example:

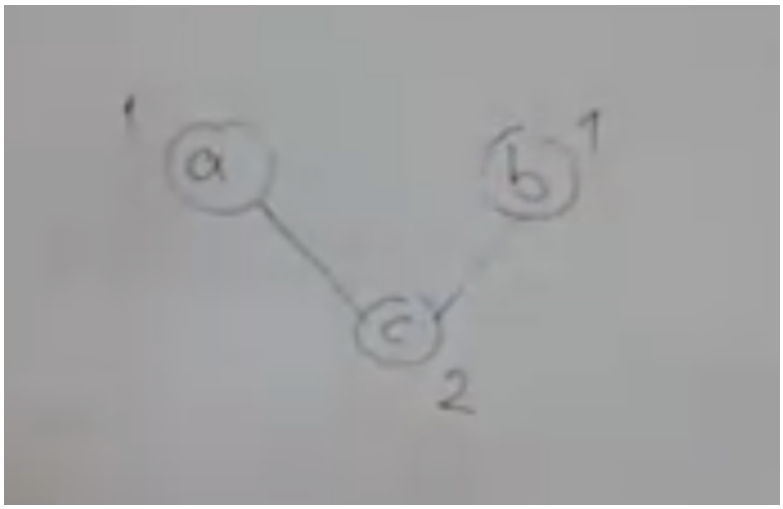
V = {a,b,c} , E = {(b,c)} suggest that A is alone , and B and C is connected.

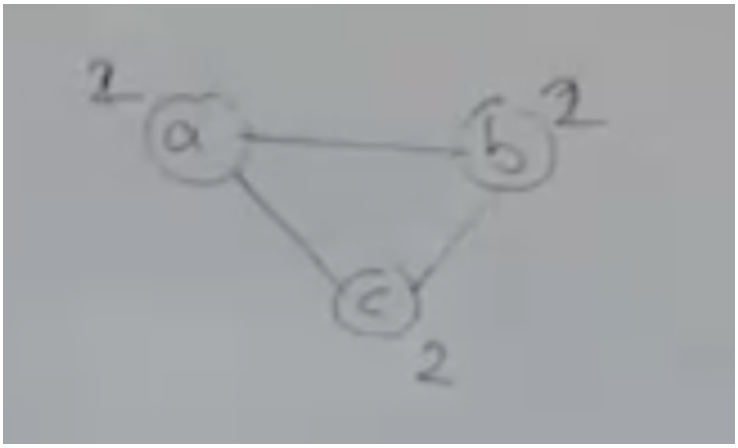
Degree sequence is [0,1,1]

min. change yiels [1,1,1] which makes the graph 3-degree anonymous

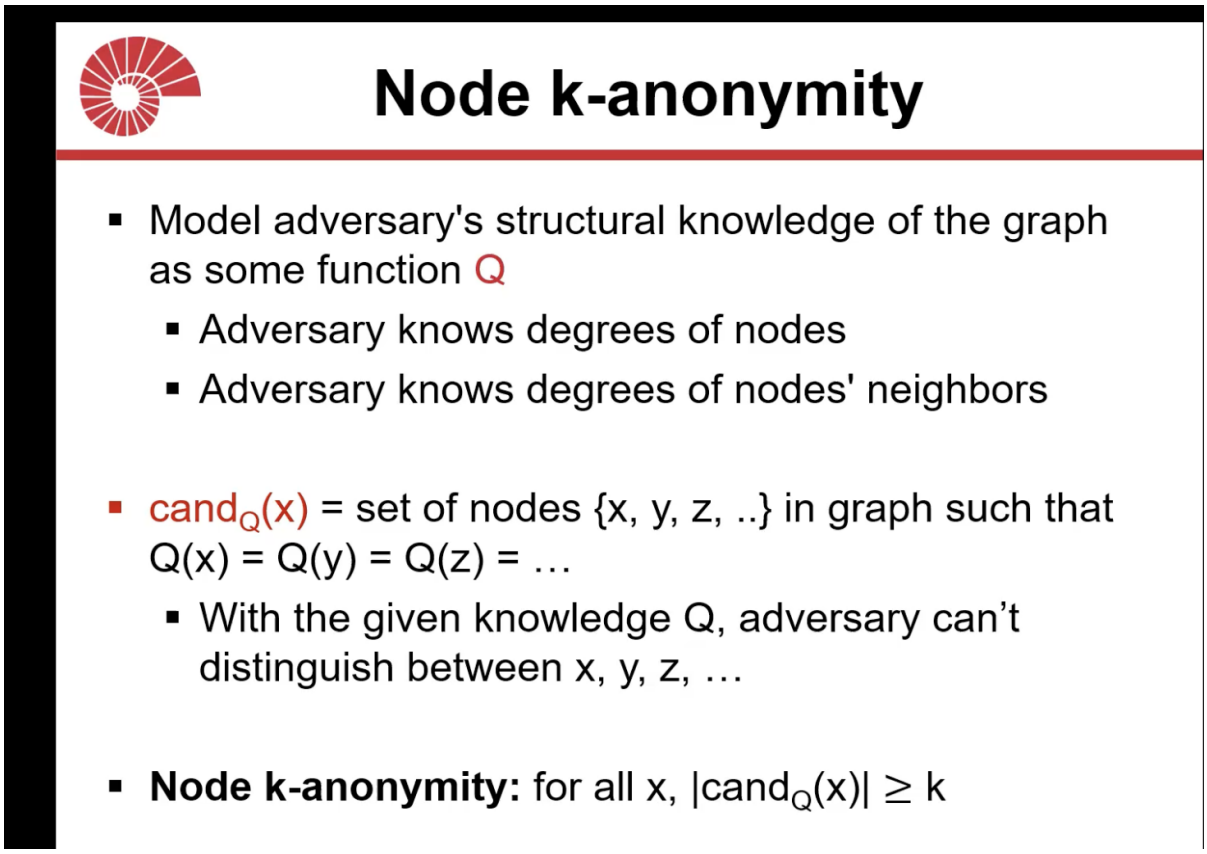
But we cannot do that unless adding self-edges. Adding an edge increase the other one’s degree sequence too, so 3-degree anonymous cannot be satisfy with one move.







Node k-anonymity



I am allowing adversary to know degrees of nodes and degrees of nodes' neighbors but I am not allowing adversary to know degrees of nodes of neighbors neighbors

- Data in the real world seldomly fits into one data type perfectly, therefore we need to consider custom and hybrid data representations.
  - And adapt our privacy definitions accordingly!

- We have methods to anonymize tabular, set-valued, location data individually by themselves.
- **But how about the following?**

Name	Gender	Bought Items	Location
Alice	Male	{milk, butter}	(-45.981, 54.331)
Bob	Male	{laptop, mouse}	(-40.221, 50.684)

- We covered anonymization of various data types
  - Address non-tabular data via extensions of approaches designed for tabular data
- **What to do when you face new data?**
  - Understand & model the data
  - Establish EIs, QIs, SAs, adversary model, …
  - Establish a suitable privacy definition
    - E.g., your own, custom extension of k-anonymity or l-diversity
  - Design and implement an algorithm so that you can enforce your privacy definition
  - Ensure that your approach preserves data utility
- **Use and extend what you already learned!**

Differential Privacy

So far, we are dealing with the privatizing the microdata. The data of individuals, hiding specific informations about indiviuals are our aim.

In the differential privacy, we are dealing with macrodata. Data that we can train machine learning models with it.

1) Privacy for aggregate statistics

Creating a histogram from the data

2) Privacy for interactive mechanics

We cannot fetch the whole data row by row but we can do a statistical queries to our data.

Like COUNT, SUM

Differencing Attack

One can use those queries and by right combinations, they can deduce information. For example with querying the average salary of the comp faculty to db and and applying second query to db that average salary of Comp faculty excluding the professor’s hired in 2020 gives the information about the Emre Hoca’s salary because he is the only one that is hired in 2020.

Law of Info Recovery

Fundamental Law of Information Recovery

Formal theorem by Dinur and Nissim in 2003.

Informal interpretation by Dwork and Roth ‘ Overly accurate answers to too many statistical queries will destroy privacy in a spectacular way

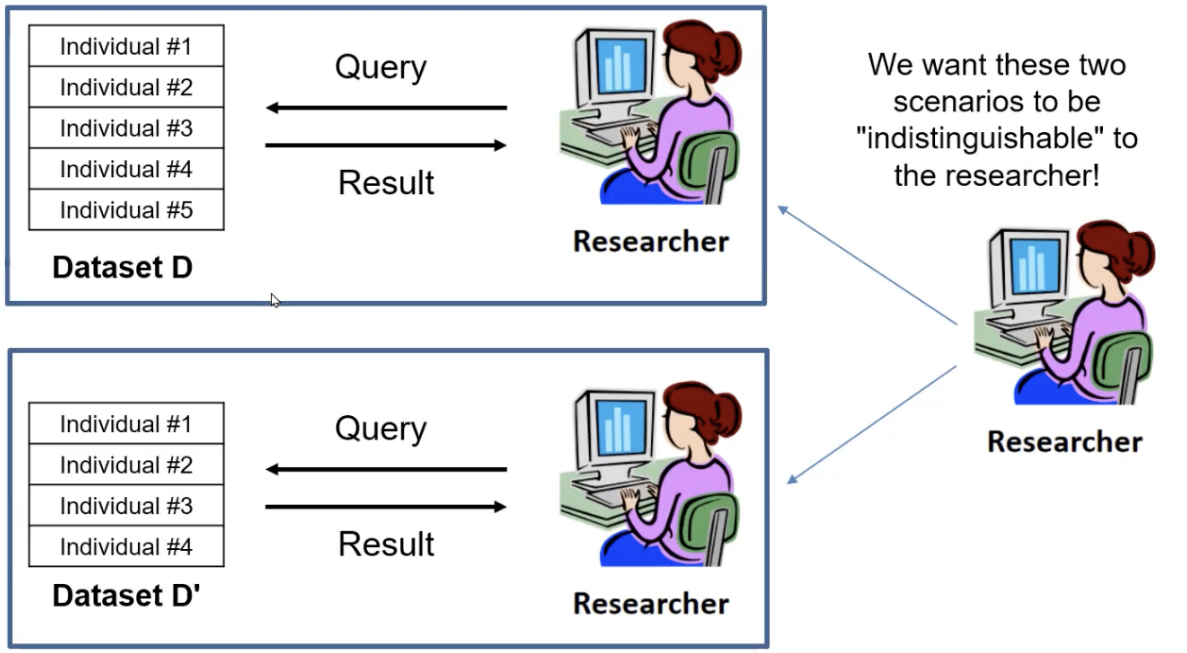
So with sufficient many queries and well answered ones one can re-construct a db with statistical queries.

Relaxed version

Dalenius‘ desideratum

are skipped. He said there are no questions on the midterm related to that topics.

Differential Privacy (DP)



Datasets are different. They are called as **neighbored datasets**. This term is used when they are differ by one record. Which one is different is not important. It can be plus one or minus one.

- Let $\mathbf{D}$ and $\mathbf{D}'$ be two neighboring datasets.
  - (We will clarify neighboring datasets in a moment.)

A randomized algorithm A satisfies ε-differential privacy, if for any two neighboring datasets D and D' and for any output O of A,

$$\frac{\Pr[A(D)=O]}{\Pr[A(D')=O]} \leq e^{\varepsilon}$$

- Smaller ε yields stronger privacy ($\varepsilon \geq 0$).
  - Think about what happens when ε = 0.001 versus ε = 10.

Probability of Algorithm A acting on Dataset D and producing Output O.

divided by

Probability of Algorithm A acting on Dataset D* and producing Output O.

should be smaller than e (which is a natural number e = 2.71)

and it has a exponent epsilon which is a privacy parameter for us.

This should be satisfy for all neighboring D D* and for all outputs of the algorithm Aa

if epsilon is small like 0.001 than e to the epsilon is close to 1 , so in that case our nominator and denomitor should be very similar.

if epsilon is 10, then e to the epsilon is big number. In that case, nominator is more likely to happen. So we can easily distinguish it from D*. **So epsilon itself is strength of our privacy. While it becomes smaller, we get more privacy.**

💡 Database is always lying to us with a noise with probabilistic way. This should be done to achieve privacy. Noise is random

Lecture 10

Neighboring Dataset

Originally defined for tabular data

D and D* differ by one row, either by adding a one row or deleting one row.

Or you can also change a row's content by keeping its size.

There are different ways to define neighboring datasets

Graph Data:

You can create neighboring dataset by adding or deleting one edge. Or you can create by differed one vertex. But this approach also needs you to change all adjacent edges.

Set-Valued Data:

You can define D and D* by changing one transaction.

We should have a metric of how our choice of to define neighboring dataset affects the database.

This metric called **Sensitivity.**

- S(q) = Sensitivity of query q
  - How sensitive is q to changing from D to D'?
  - At most, how much can q be impacted?

$$S(q) = \max_{D,D'} ||q(D) - q(D')||$$

- q can be any query or function
  - e.g.: COUNT query on tabular data
    - How many users in table w/ age between 15 and 25?
    - S(q) = 1

For example if we choose adding or removing a row to achieve D and D*. There are 3 possibility when we are executing a COUNT query on our tabular data.

We might add an row to achieve D*. In that case only difference between query results will be added member so Count query has surplus of 1.

We might delete a row to achieve D*, in that case scenario is same the difference is just the surplus 1.

We could also deleted or added a row that doesnt satisfy query result, so in that case our sensivity doesnt affect at all. It becomes 0.

We should take the maximum effect which is 1.

In the **Multiple COUNT** queries that return values is a list.

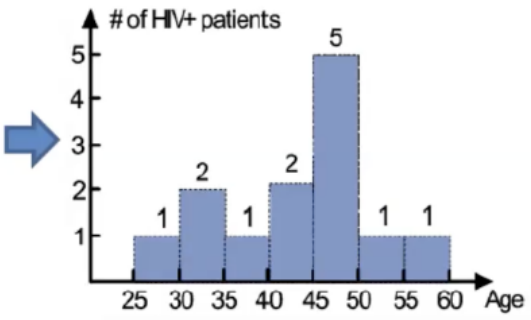
S(q) = number of queries in q at the worst case

For example : [15,20,3,87,32] becomes [16,21,4,88,33] at worst case. S(q) = 5

Be careful about how neigboring datasets are defined. Previous cases was built on neighboring datasets that are constructed with adding or deleting an row. But another neighboring dataset can be defined at modification of row instead of addition or removal. In this case S(q) is also differently calculated.

- Consider the following HISTOGRAM query q

Name	Age	HIV+
Frank	42	Y
Bob	31	Y
Mary	28	Y
Dave	43	N
…	…	…



So if i create an neighboring data with addition or removal S(q) equals 1. Because you are going to add or deleting one.

If you are creating a neighboring data with modification S(q) equals to 2. Because you are changing 2 figures. One with the previous data, and the one is the changed data.

Sensitivity – Exercises

- Same HIV dataset as previous slide
- Neighboring datasets = obtained via modification of exactly one row
- Assume ages are between [0, 100]

- Sensitivity of q = min(Age)?
- Sensitivity of q = max(Age)?
- Sensitivity of q = median(Age)?
- Sensitivity of q = average(Age)?

- Some queries inherently have large sensitivity…

Example

For the min case; assume there is a one record which has age value 100, if we are going to change by modification and make it 0, it becomes minimum. Previously it was 100, and it become 0. Change is 100.

Example

For the max case, assume there is one record which has age value of 0, if we are going to change it by modification and make it 100, 100 becomes the new maximum. So difference is the 100.

Example

For the median case, assume there are 5 records and they are 0, 0,0, 100 ,100 age values. Median is 0, but if we change it by the modification, at the worst case we are changing the median itself and make it 100.

Example

For the average case, 100/size of the data set is our average and in the worst case size of the data set is 1.

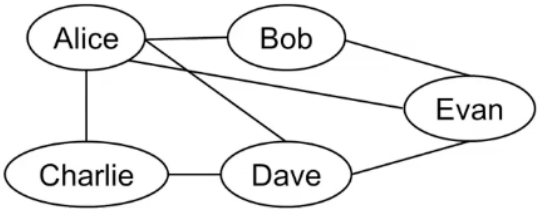
Counting queries are smaller sensitivities according to those queries talked upon.

> 💡 Sensitive is never tied for the particular dataset. It is property of the query, it is a property of how neighbored dataset is defined.

> 💡 HINT ABOUT MIDTERM. He will ask a question about defining sensitivity. you can define it not on all possible neighbouring data sets, but you can measure sensitivity according to a given dataset and its neighbors. And in that case our previous edge cases might not be obtainable in real world stage.

Another examples

- Graph data



- q = Count # of total edges in graph
  - Sensitivity under edge DP?
    - Add/remove one edge to graph
  - Sensitivity under node DP?
    - Add/remove one vertex and all adjacent edges to it

- q2 = Count # of vertices that have degree $\geq n$

q = count # of total edges in graph

Sensitivity under edge differential privacy ?

Add/remove one edge to graph differs by one

Sensitivity under node differential privacy ?

Add/remove one vertex and all adjacent edges to it.

For example I have graph that has 4 vertices. I can add an new vertices and bind all of the other vertices by edges. Therefore my sensitivity becomes number of vertices(nodes).

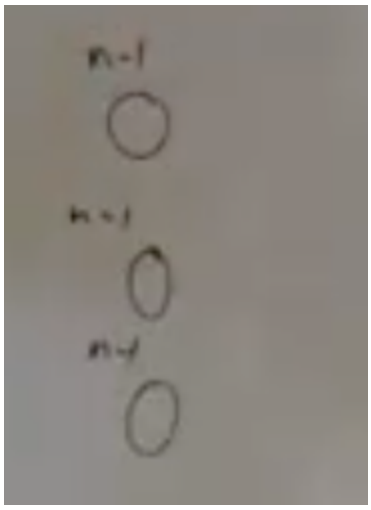
q2= Count # of vertices that have degree ≥ n

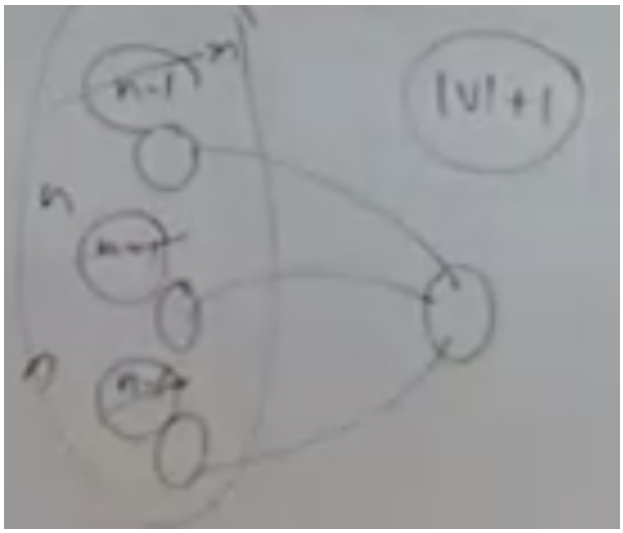
Under edge differential privacy :

The answer is 2. Because let say we have two nodes that has degree n-1. If we connect them with edge, both of the node increased by one , and our total count # of vertices that have degree >n increased by two.

Under node differential privacy:

Lets think about a scenario that all of our nodes has n-1 degree, when we are adding a new node, and connect to the all nodes with edges, our sensitivity under the node differential privacy becomes number of nodes +1.





So far

- Let $\mathbf{D}$ and $\mathbf{D}'$ be two neighboring datasets.

A randomized algorithm $\boldsymbol{A}$ satisfies $\boldsymbol{\varepsilon}$-differential privacy, if for any two neighboring datasets $\boldsymbol{D}$ and $\boldsymbol{D'}$ and for any output $\boldsymbol{O}$ of $\boldsymbol{A}$,

$$\frac{\Pr[\boldsymbol{A}(\boldsymbol{D})=\boldsymbol{O}]}{\Pr[\boldsymbol{A}(\boldsymbol{D}')=\boldsymbol{O}]} \leq e^{\varepsilon}$$

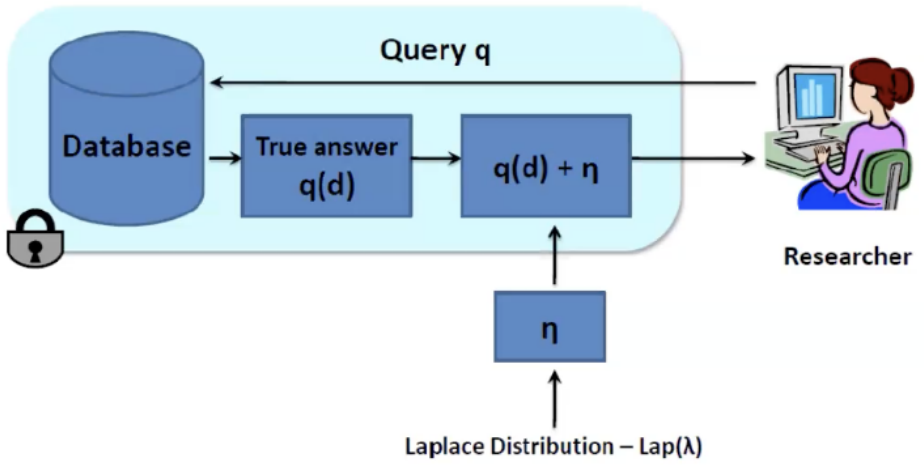
- $\boldsymbol{\varepsilon} \geq 0$ is the privacy parameter.
  - Smaller $\boldsymbol{\varepsilon}$ yields stronger privacy.
- S($\boldsymbol{q}$) = Sensitivity of query $\boldsymbol{q}$
  - Max change in query result from $\mathbf{D}$ to $\mathbf{D}'$

Now we are going to combine epsilon and sensitivity.

Laplace Noise

In order to satisfy Epsilon Differential Privacy, we can use Laplace Noise

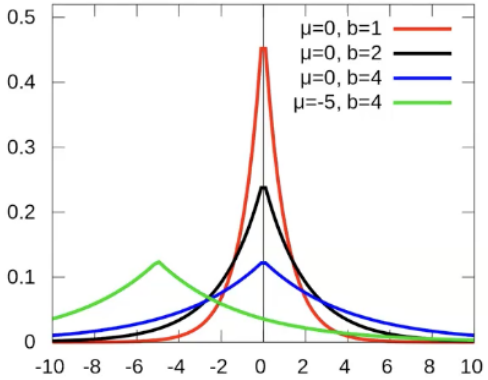
- In order to satisfy ε-DP, we can use Laplace noise
  - Add random noise to query answers from a properly calibrated Laplace distribution



Just result is visible to the researcher.

- Draw a random sample from Laplace distribution
  - Two parameters: mean μ and scale b
- **Theorem:** Adding random noise drawn from a Laplace distribution with $\mu = 0$ and $b \geq S(q)/\varepsilon$ satisfies ε-DP.
  - $\mu = 0$, good for utility
  - What happens when..
    - S(q) increases / decreases?
    - ε increases / decreases?

$$f(x \mid \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$



Mean of the Laplace distribution should be 0.

b is the scale parameter, it should be higher and equal than sensitivity divided by epsilon.

For the smaller b distribution is more dense around the mean. While b is getting larger, you are more likely to see more divergent one. So higher b means more noise.

mean determines where the mean is , the u symbol

If we keep epsilon fixed, but we have a query

Higher sensitivity means that when you move a dataset to neighboring dataset, that change is actually going to change the result of your query a lot.

In order to response to not the response to not leak the fact whether it was originally D or D*. You have to add a lot of noise to the answer of the query, so

you can hide that if is in D or D*

When epsilon is small we have higher privacy.

if you decrease epsilon, stricter privacy, b goes up thus, noise amount goes up.

He discuss that one-time access vs reasking the question . If you allow one-time access, epsilon is equal to one for example ; and than you say I allow this person to 100 times , then each of the epsilon one accesses are accumulate to 100. Which is by definitional of DP, much higher epsilon means less privacy.

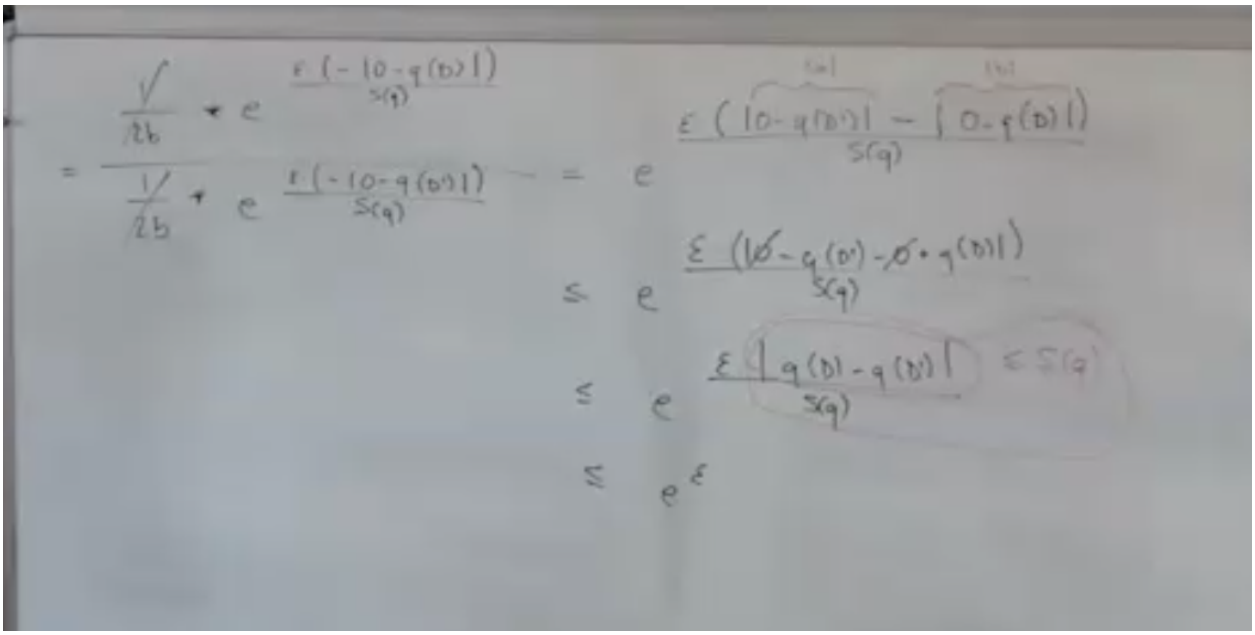
One- time access ?

Okay here is the deal that I understood from the questions. If one person asking a query to the database, database caching the answer of the question, so if the same person ask the same question again, there is no privacy budget is spent, because the answer is already cached. But if another person asking the same question, we should spent a privacy budget, we cannot cache of course.

Proof of the Laplace Mechanism

- Let q be a numeric query to answer w/ ε-DP.
- Step-by-step **algorithm**:
  1) Compute the real answer of query: q(D)
  2) Find sensitivity of q: S(q)
  3) Draw a random sample from Lap(0, $S(q)/\varepsilon$)
     - Or Lap($S(q)/\varepsilon$) for short
  4) Add the random sample r to real answer q(D): q'(D) = q(D) + r, where q'(D) becomes the noisy answer
  5) Return q'(D) to the user

Proof of Laplace Mechanism



Lecture 11

We have learnt the Laplace Mechanism at the previous lecture. In the laplace, query has numeric output, so adding noise make sense. But there are some queries that doesnt have numeric output, like what is the most popular eye color or what is the most common nationality. In those queries domain includes discrete outputs. This is where Exponential Mechanism appear in order to anonymize,

- Define a score function q_F : D x R -> $\mathbb{R}$
  - D is the dataset
  - R is the domain of discrete outputs
  - $\mathbb{R}$ is the (infinite) set of real numbers
  - q_F : Custom **goodness score** of each candidate output from R

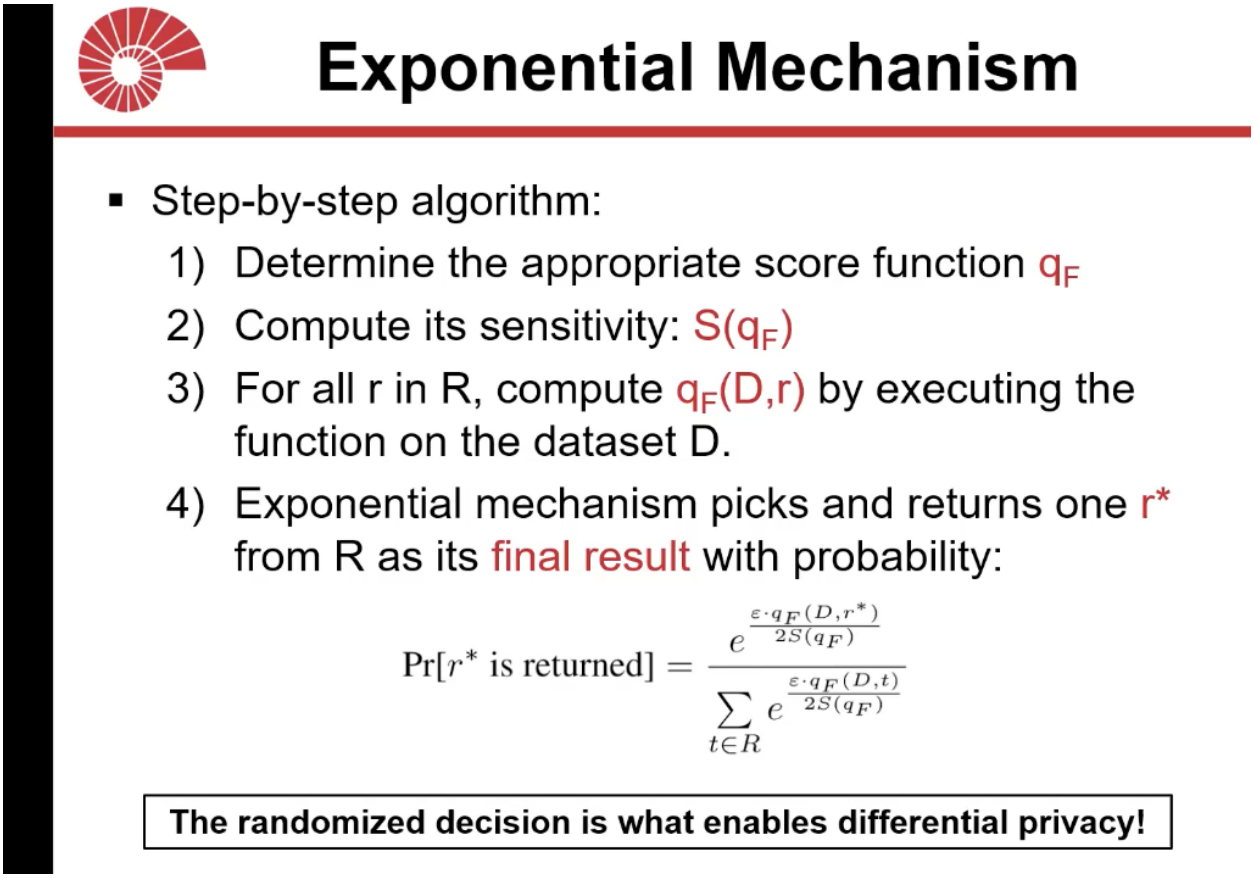
it use like score function that we learnt in the machine learning.

- Example: "What is the most common nationality?"
  - Suppose there exist 4 nationalities
  - R = {Turkish, Greek, American, Chinese}
  - q_F(D, nationality) = # of people in dataset D that have this nationality
    - q_F(D, Turkish) = # of Turkish people in D
    - q_F(D, Greek) = # of Greek people in D
    - ...
- What is the sensitivity of S(q_F)?
  - Same idea as before, but needs to consider r in R

$$\max_{r \in R, D, D'} |q_F(D, r) - q_F(D', r)|$$

In the Laplace mechanism, our sensitivity doesnt take care of the possible outputs. This is the difference between Laplace mechanism and exponential mechanism.

In the query of the example, “What is the most common nationality ?” with the scoring function qF(D, nationality) is the number of people in dataset D that have this nationality, sensitivity is 1 if our DP based on addition/removal.



In step 4, the equation basically saying to us the probability of certain r* is returned to the user is going to be equal to e to the power epsilon times score of that output divided by two times sensitivity of the qf function, divided by summation of the all of the expontions for all possible outputs.

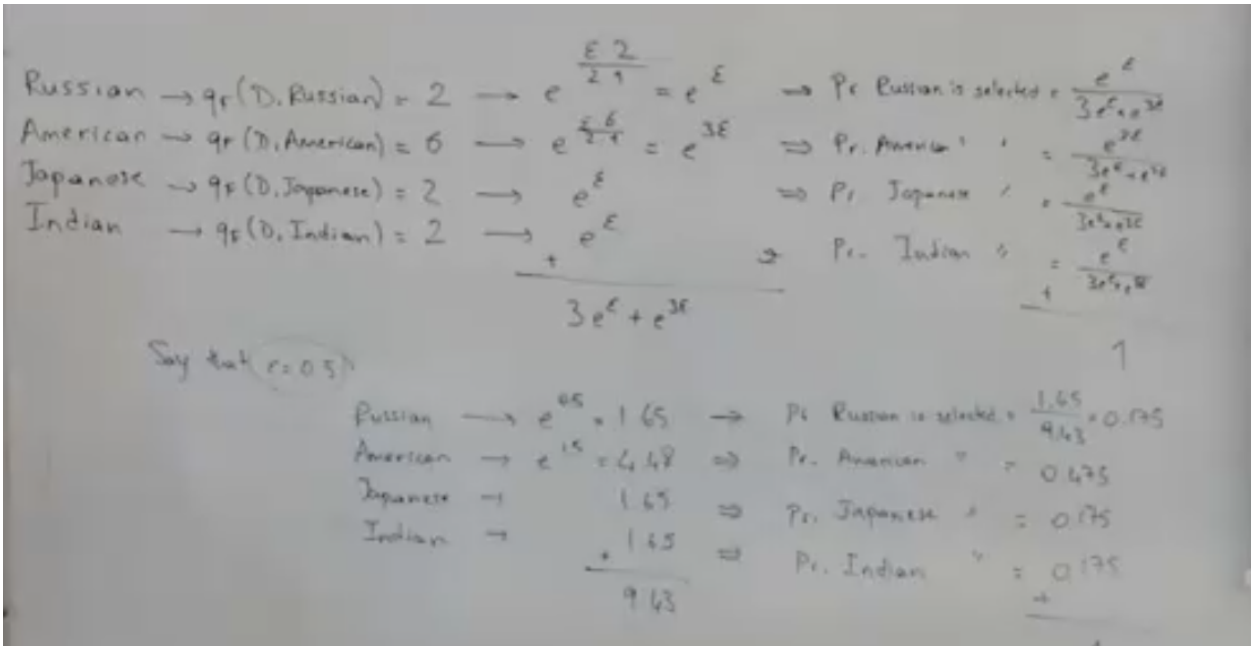
It is not alway

- "What is the most common nationality?"
  - R = {Russian, American, Japanese, Indian}
  - q_F and $S(q_F)=1$, same as before

q_F(D, Russian) = 2
q_F(D, American) = 6
q_F(D, Japanese) = 2
q_F(D, Indian) = 2

$$\Pr[r^* \text{ is returned}] = \frac{e^{\frac{\varepsilon \cdot q_F(D,r^*)}{2S(q_F)}}}{\sum_{t \in R} e^{\frac{\varepsilon \cdot q_F(D,t)}{2S(q_F)}}}$$

Zip	Age	Nationality	Disease
13053	28	Russian	Heart
13068	29	American	Heart
13068	21	Japanese	Viral
13053	23	American	Viral
14853	50	Indian	Cancer
14853	55	Russian	Heart
14850	47	American	Viral
14850	59	American	Viral
13053	31	American	Cancer
13053	37	Indian	Cancer
13068	36	Japanese	Cancer
13068	32	American	Cancer



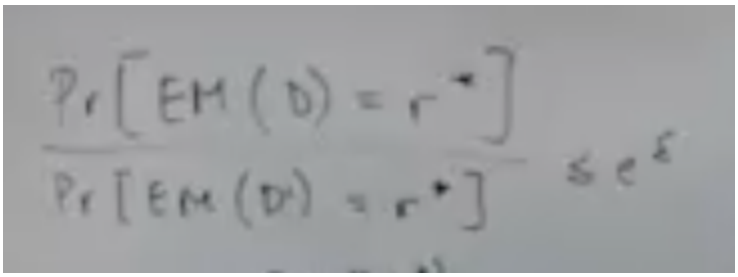
if we set the epsilon more large, that means less privacy, that makes the higher probability, american in that case is higher.

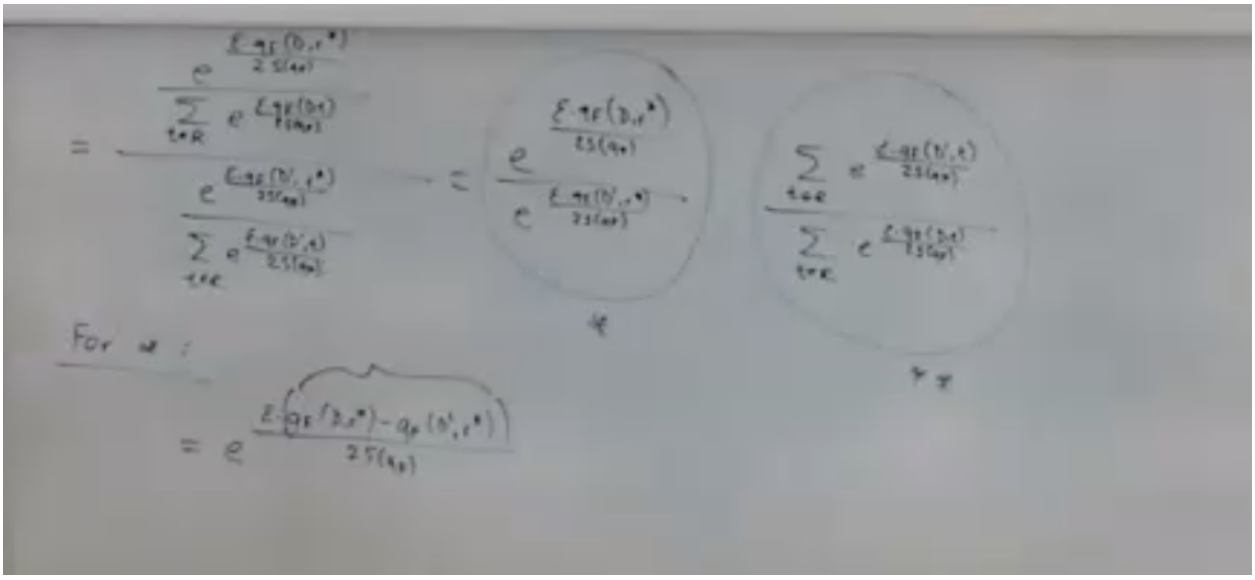
- **Rationale:** Makes "high quality" outputs (high scoring outputs) exponentially more likely to be returned
- **Factors** that impact accuracy:
  - Negatively related to $S(q_F)$
  - Positively related to privacy budget ε
- **Important:** Only the output of Exponential Mechanism (EM) is shown to the analyst/adversary, remaining computations remain internal
  - "What is the most common nationality?" -> “American” is the only thing observed by the adversary
  - Why must this be the case?

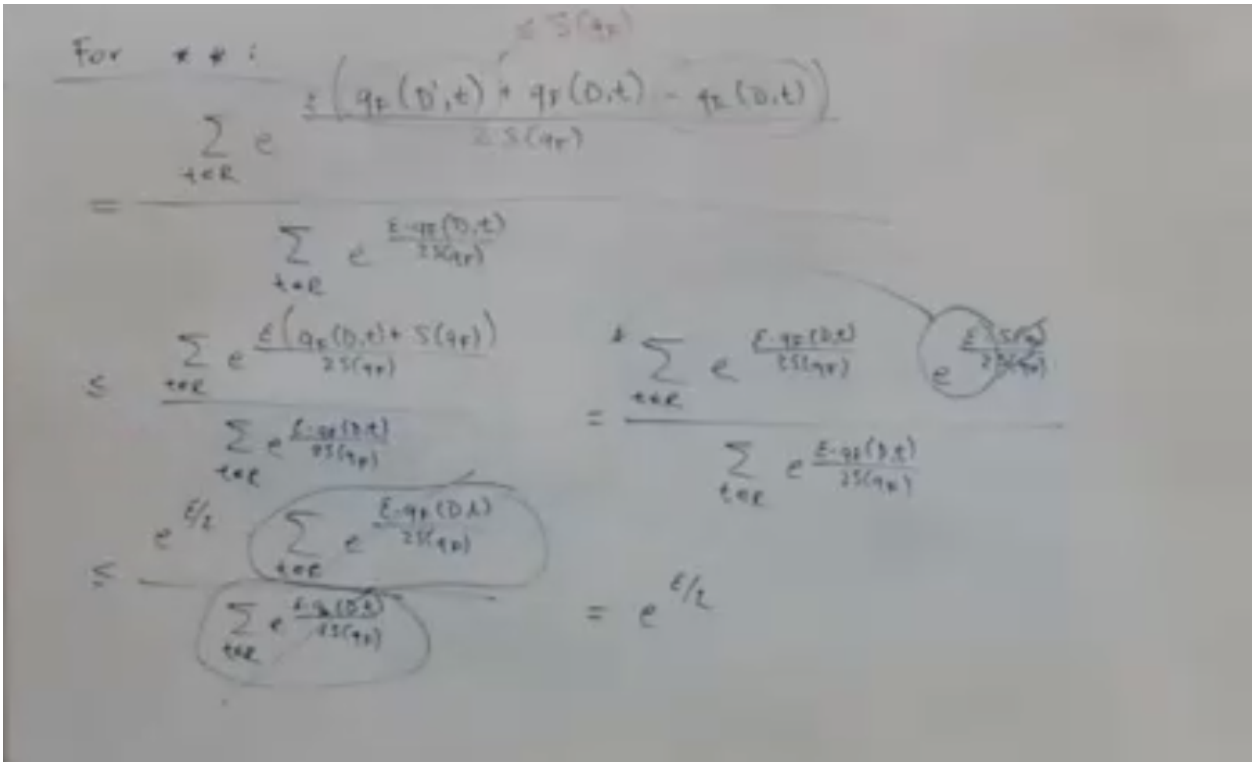
At the below, he makes a proof of that exponential mechanism satisfies epsilon - Differential Privacy.

Proof of straithforward, just replace the equations with formulas and divide the equation to 2 parts. At the summation part only trick is add and substract score function at dataset D with t. He is not going to ask a proof in the midterm, but that might be useful on the homework.

ok, he change his mind, he might ask a proof that whether it satisfies the epsilon differential privacy, but the mechanism is much simpler he says.







Composition Properties

If i run the same query many times,

I might have multiple accesses.

In general I might be interested in asking different queries, multiple of times

Up to this point we talked about single access numeric access, or single discrete access that we talked about Laplace and Exponential mechanism. Now we are thinking about multiple access queries with composition properties.

Composition Properties

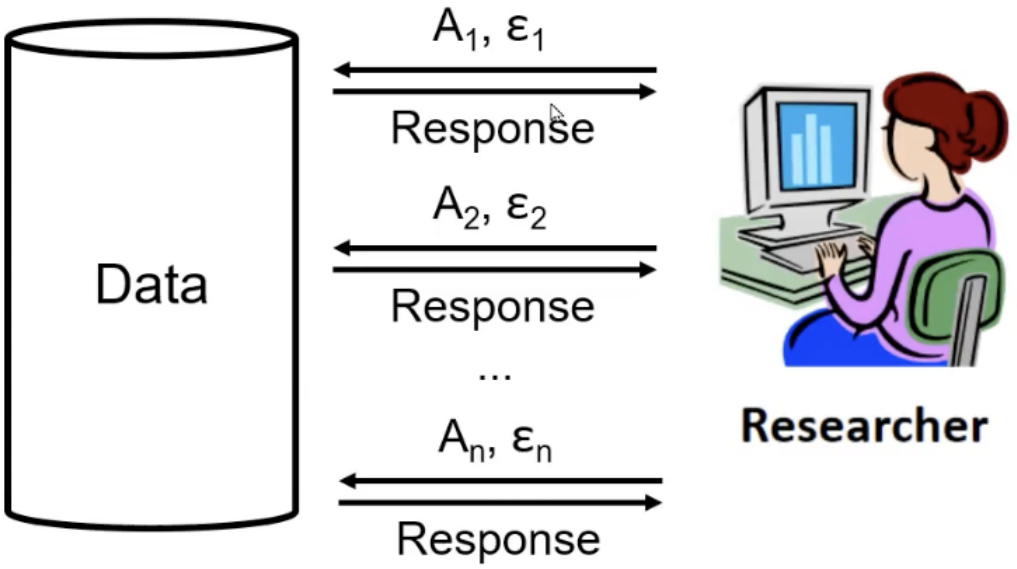
The idea of composition properties or the idea of generalizing those one time access to more complex algorithms . we are going to take data mining algorithms, and we are breaking down to simple steps than we are going to solve each of the steps with exponential and Laplace mechanism then we are going to use composition properties to reason about total privacy of the total algorithm.

Composition properties are;

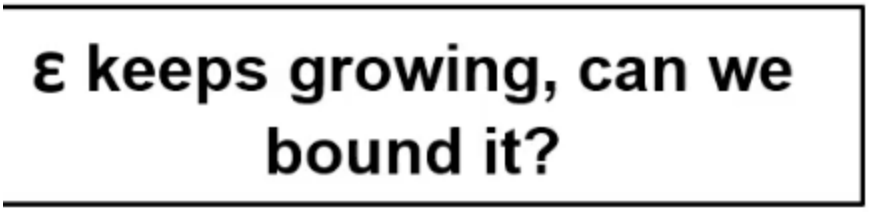
Sequential Composition

For all algorithms for each of them satisfy epsilon DP, the combination of their output satisfies epsilon dp with all of their epsilons accumulative

- If $A_1, A_2, ..., A_n$ are n algorithms that each satisfy ε_i-DP for $1 \leq i \leq n$, then the combination of their outputs satisfies ε-DP with $\varepsilon = \varepsilon_1 + \varepsilon_2 + ... + \varepsilon_n$
- **Meaning:**
  - I have small algorithms, each satisfying DP
  - I perform them sequentially (one after the other) on the same dataset
  - The process, as a whole, satisfies DP with an accumulated privacy budget
    - Budgets of the small algorithms are added up
    - Total privacy loss becomes larger and larger



More algorithm that we run, more privacy loss we encounter.



The way to bound it with **budget perspective**. Total spending is bounded by epsilon. Researcher should spend piece by piece.

> 💡 This is just for the single individual

It needs budget policies that user figure out how much of the epsilon should it spend for the algorithm.

If researcher would know how many algorithm that he run, he can spend epsilon/n for each algorithm

Or he can use geometric series. spend epsilon/2, epsilon/4, epsilon/8.. for each algorithm iteration spending epsilon becomes smaller so the each time privacy is increase therefore more noise is involved.it become more meaningless. But you can guarantee that your epsilon never depleted because you used geometric series. Total value is never reached.

Parallel Composition

- If $A_1, A_2, \ldots, A_n$ are n algorithms that each satisfy ε_i-DP for $1 \leq i \leq n$, and they access disjoint subsets of the dataset D, then their combination satisfies $\max(\varepsilon_i)$-DP.

- **Meaning:** If I query different subsets of the data, then I don't have to pay the price of sequential composition
  - Intuitively, why is this the case?

- Utility-wise, parallel composition yields a huge improvement over sequential composition.
  - Why?

Disjoint means that two Algorithm cannot be satisfied at the same time.

Remember, HIV + and HIV- patients considered as disjoint in this case. This is considered because we dont know the summation, we cannot deduce hıv- from the hıv+ patients

- **Which composition to apply?**

- q1 = number of HIV+ patients that have blonde hair
- q2 = number of HIV+ patients

- q3 = number of patients that are HIV+
- q4 = number of patients that are HIV-

- q5 = number of patients that are HIV+
- q6 = number of patients that have age [10, 20]

First: sequential bc they are not disjoint

Second: parallel

Third: sequential, they are not disjoint

Immunity to post-process

Post-Processing

- If A_1 is an algorithm that satisfies ε_1-DP, any post-processing or future use of A_1's output remains ε_1-DP.

- **Meaning:**
  - I take what I learned differentially privately
  - I can do whatever I want with it
  - And it won't break the DP guarantee
    - As long as I don't peek at the private dataset again
    - I must pay a new price ε_2 to peek at the dataset

Lecture 12

Introduction

- Previously, we covered:
  - DP fundamentals – definition of DP, sensitivity, neighboring datasets
  - Building blocks: Laplace, Exponential
  - Composition properties
- **Next:** Let's look at their applications to different types of datasets and tasks
- **Two important aspects:**
  - (1) Checking/proving if given algorithm satisfies DP
  - (2) Turning a non-private algorithm into a DP algorithm

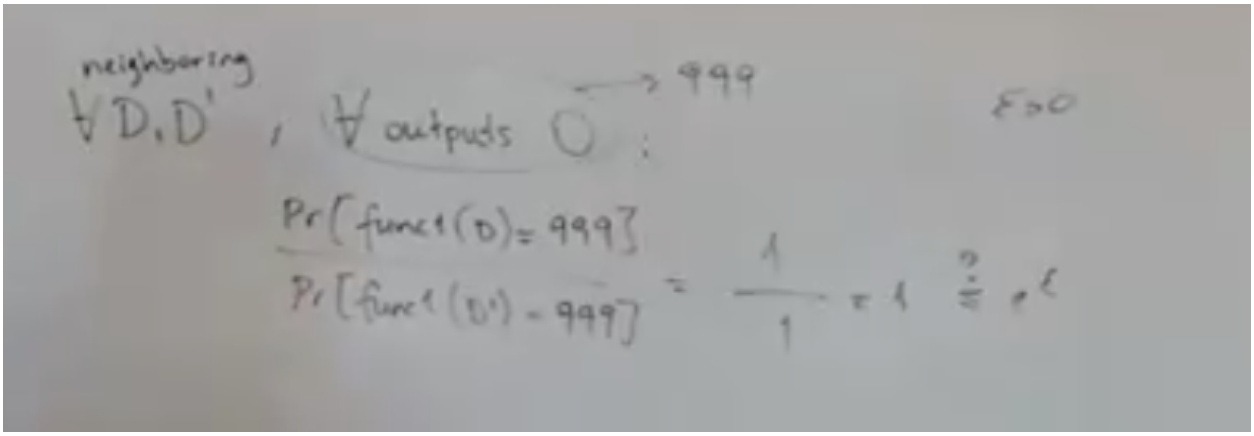
- Given the following algorithm/function, decide whether it satisfies DP or not.

- func1: Always return 999
  - Regardless of dataset and/or query

```
def func1(dataset, query, epsilon):
    return 999
```

Name	Age	HIV+
Frank	42	Y
Bob	31	Y
Mary	28	Y
Dave	43	N
...	...	...

Intuitively: Zero privacy leakage here!



It is DP and it is the trivial case. You cannot differentiate whether it is coming from D or D* for all outputs because all of the outputs are same.

- func2: Add fixed noise of 10
  - Assume query is a count query

```
def func2(dataset, query, epsilon):
    return query(dataset) + 10
```

Name	Age	HIV+
Frank	42	Y
Bob	31	Y
Mary	28	Y
Dave	43	N
...	...	...

This is a very good question. Being DP means that we should not distinguish from the output, that is from dataset D or the neighbored dataset D*. In a more formal language, for all outputs, it should satisfy equation at below. Remember that epsilon is a positive number.And it should be satisfied for all outputs of algorithm A.

- Let $\mathbf{D}$ and $\mathbf{D}'$ be two neighboring datasets.

A randomized algorithm A satisfies ε-differential privacy, if for any two neighboring datasets D and D' and for any output O of A,

$$\frac{\Pr[A(D)=O]}{\Pr[A(D')=O]} \leq e^{\varepsilon}$$

- $\varepsilon \geq 0$ is the privacy parameter.
  - Smaller ε yields stronger privacy.
- S(q) = Sensitivity of query q
  - Max change in query result from $\mathbf{D}$ to $\mathbf{D}'$

Therefore, look at the question again. If we find a one example that doesnt satisfy the equation, we can say that it is not DP because it must hold for all of the

outputs.

Let our query is a count query and it is asking number of HIV+ people.

Real output is 3 of course, but it is adding 10 as noise , and returning 13 to the user. And for all other output probability is 0 other than 13, because there is no randomness included.

Let say in the neighbored dataset, one of the hiv+ people changed with hiv- and our query result become 12.

So look at the equation for the output O = 13, The nominator is 1 of course, but for the denominator, other than O=12, probability is equal to 0 in the neighbored dataset

Therefore, left hand side becomes infinity and infinity is not smaller than 2 to some positive number.

Therefore it is not DP

- func3: Return a row from the original data

```
def func3(dataset, query, epsilon):
    # some computation here
    return computation_result, dataset[0]
```

Name	Age	HIV+
Frank	42	Y
Bob	31	Y
Mary	28	Y
Dave	43	N
...	...	...

Okay so we know that if we get one counterexample, than it is not DP.

Counterexample follows this, we are returning Frank's row as a output on the nominator at the equation.

If our neighbour dataset removes the first row, than our first row is changed at the denominator. So O= Frank is no longer probable, therefore it makes the denominator 0. Same logic apply here, infinity cannot be smaller than some e'to positive number.

It is not DP

- func4: Add Laplace noise to query answer
  - We already know this satisfies DP ☺
  - For count query, S(q) = 1

```
def func4(dataset, query, epsilon):
    return query(dataset) + Lap(1/epsilon)
```

Name	Age	HIV+
Frank	42	Y
Bob	31	Y
Mary	28	Y
Dave	43	N
...	...	...

It is DP. It is the case that we talked about.Laplacian mechanism gives us randomness to not differentiate where the output comes from.

Remarks

- An algorithm **A** is said to be **deterministic** if, given a particular input, **A** always produces the same output

- Most DP algorithms are **randomized**
  - Laplace, Exponential, ... are randomized
  - By definition, DP is probabilistic
    - Needs randomized mechanisms to satisfy

- **Exception:** “Trivial" or data-independent algorithms which are not impacted by the dataset **at all**
  - E.g.: **func1**

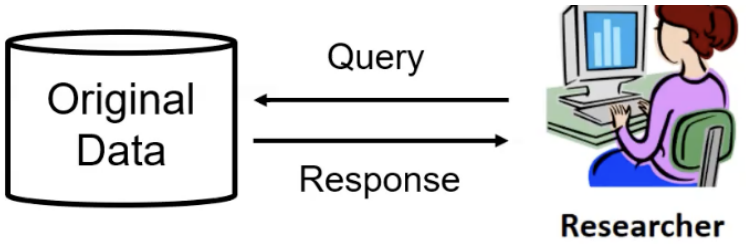
DP Proofs

- In general: How to prove or disprove an algorithm satisfies DP?
  - We need to reason about Pr[A(D)=O] and Pr[A(D')=O] under different A, D, D', O

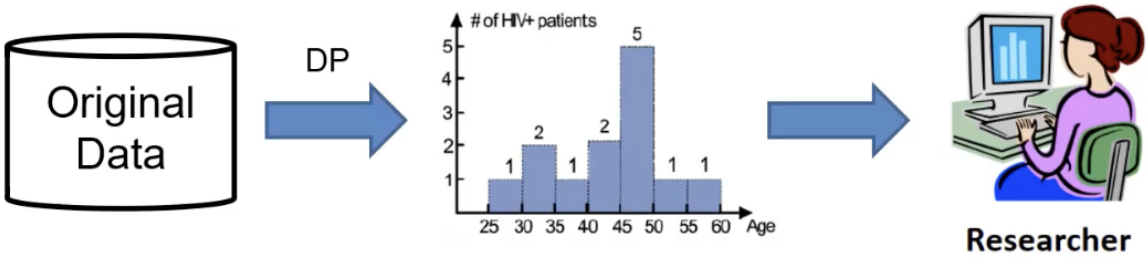
- **To prove:** show DP equation is satisfied
  - For all ε or for some ε?
- **To disprove:** show violation of DP with D, D', O
  - One violation (counter-example) is sufficient
  - By definition, DP says it must hold for ALL neighboring D, D' and for ALL outputs

Interactive vs non-interactive

- (1) Interactive:



- (2) Non-interactive:

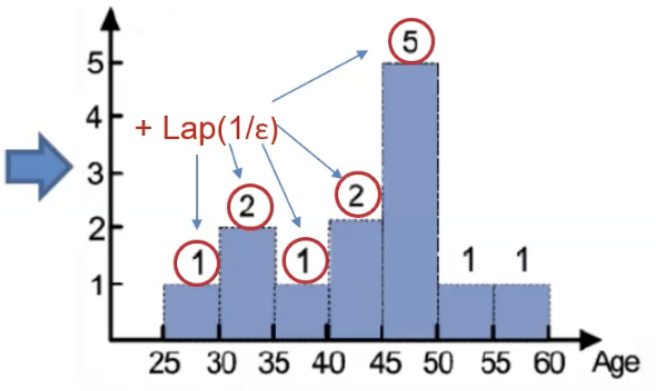


- Artifact generated once using DP algorithm
- Can be re-used or re-shared arbitrarily (thanks to DP's post-processing property)

DP Histogram

- **Case study #1:** Differentially private histogram
  - Age distribution of HIV+ patients
  - Add Lap(1/ε) noise to each bin
    - Because sensitivity is...?

Name	Age	HIV+
Frank	42	Y
Bob	31	Y
Mary	28	Y
Dave	43	N
…	…	…



In this example sensivity is 1 when we are considering removing/adding a row. So we can add a noise to each of the bin

💡 Be careful about the sensitivity. If we have also data in the row about how many times this person visit the hospital, and lets say it can be mostly 100. Instead of creating a histogram of hiv status, if we have a bar chart of hospital visits, adding Lap(1/epsilon) is not sufficient. Because sensitivity is not 1. Adding a person with 100 hospital visit change the bar chart for 100 value, therefore sensitivity is 100.

Geospatial Histogram

- Add Laplace noise to the count in each histogram cell
- Release the noisy counts (noisy histogram)

We have something similar to this at the second homework

The idea is we are given location of the taxis, and wanted to draw distribution of the map of the Porto.

Add lap(1/epsilon) because sensitivity is one.

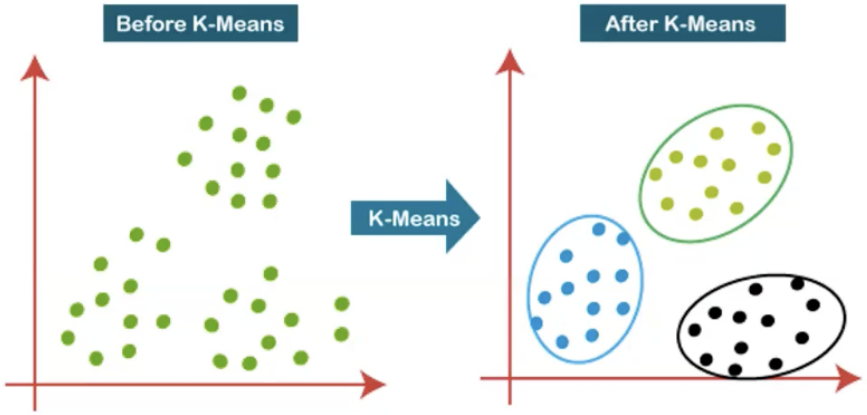
More Complex Algorithms

- Cleverly combine building blocks (Laplace and Exponential mechanisms) with composition properties (sequential, parallel, post-processing) to create powerful, complex algorithms.

- Oftentimes complex algorithms can be broken down into small, individual steps.
  - Solve each step with Laplace/Expo
  - Use composition to combine individual steps

k-means clustering

K-Means Clustering



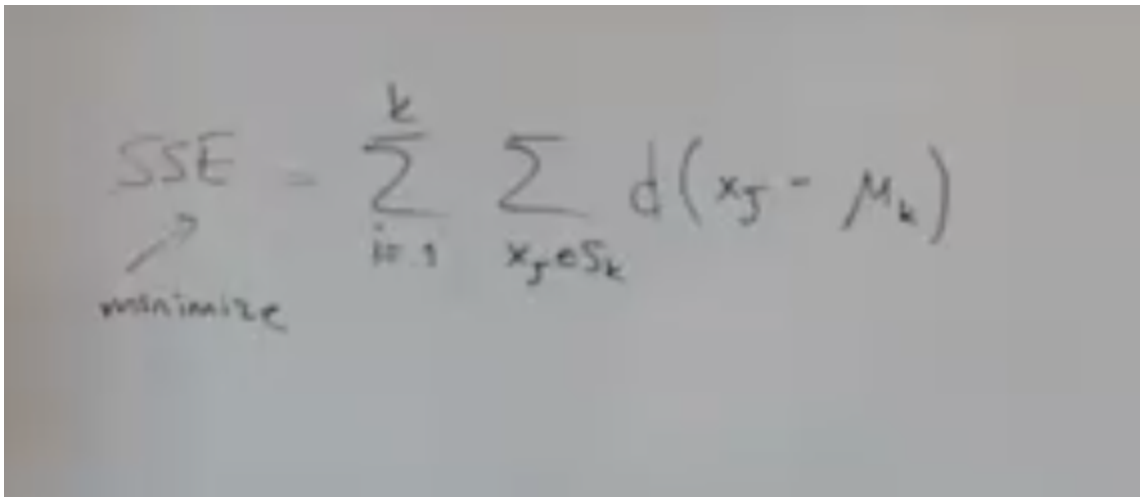
Partition a set of points $x_1, x_2, \ldots, x_n$ into k clusters $S_1, S_2, \ldots, S_k$ such that the SSE is minimized:

$$\sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

μ_i: mean of cluster S_i

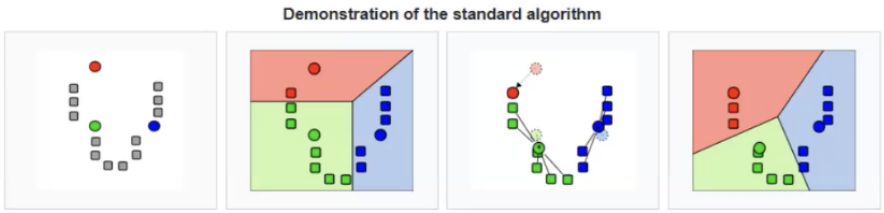
Goal in this divison to clusters is make the sum square error minimum

What sum square error means that for all clusters and for all points in that clusters (xj), the distance between xj and mean of the cluster should be minimized.



K-Means Algorithm

- Initialize a set of k centers
- Repeat until convergence:
  1. Assign each point to its nearest center
  2. Update the set of centers
- Output final set of k centers and points in each cluster



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Source: Wikipedia

- Initailize clusters

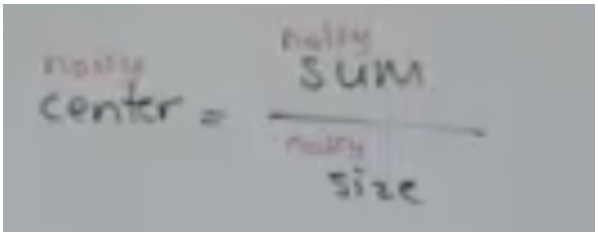
- Gray squares are data points. We have 3 centers which are circles.
- We have to repeat certain loop until we reach certain condition
- Assign each point to its nearest center. After this process finish, update the set of centers by selecting mean of the data points. After this process is finished continue to the loop until we reached convergence. Convergence means that we cannot make any new move.

Our goal is make this algorithm differentially private. We can return final set of centers of each clusters but we can not return the direct points on the each cluster. We showed that if we return the raw data from the dataset, we cannot satisfy DP. You can see at func3. So we can return number of point in each cluster.

DP k-means

- Initialize a set of k centers
- Fix # of iterations to T, repeat T times: ← Each iteration gets ε/T budget
  1. Assign each point to its nearest center
  2. To update the set of centers:
     a) Compute noisy size of cluster ← S = 1
     b) Compute noisy sum of cluster ← S = Dom
     c) Center = noisy sum / noisy size
- Output the final centers and sizes of each cluster

In order to make this algorithm DP, we have to update its sum and its size with a noise.Lets say we choose laplacian mechanism to update those. We need sensitivity and budget.



Okay first calculate the sensitivity.

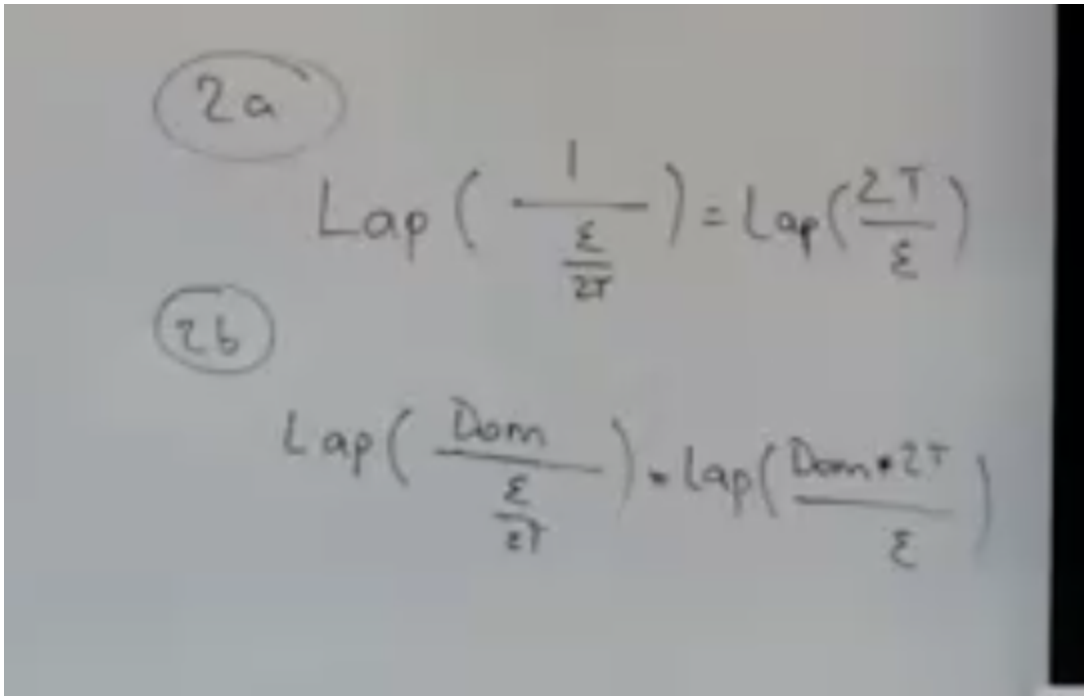
Sensitivity of noisy size is 1, because every record is considered one addition or removal

Sensitivity of noisy sum is changed according to the Domain. If the sum domain can be 100 for example sensitivity can be that.

Okay but that doesnt enough to give those as parameter to the Laplacian mechanism. Because laplace mechanism also want epsilon value, but as we talked before it might need budget logic involved.

So lets calculate the budget needed per iteration.

As we say if we have fix # of iteration as T. Each payment for algorithm is per T/epsilon. But in our loop, we are doing to calculation. One budget should be separated to noisy size, the other budget should be separated to noisy sum. Therefore for each iteration, parameter should be T/2epsilon.



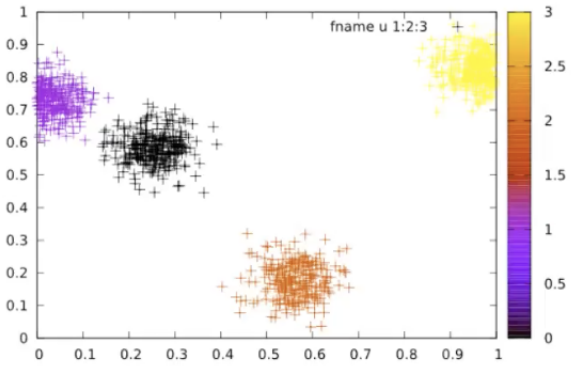
- Initialize a set of k centers
- Fix # of iterations to T, repeat T times: ← Each iteration gets ε/T budget
  1. Assign each point to its nearest center
  2. To update the set of centers:
     a) Compute noisy size of cluster ← Lap(2T/ε)
     b) Compute noisy sum of cluster ← Lap(2T*Dom/ε)
     c) Center = noisy sum / noisy size
- Output the final centers and sizes of each cluster

💡 You can also first anonymzed the data and that apply k-means clustering.

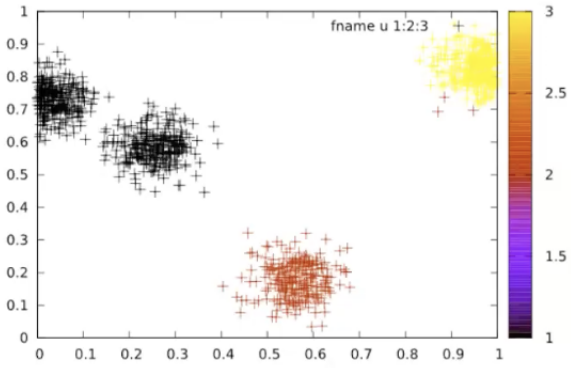
Sample Execution

- T=10 iterations, random initialization

Original K-means algorithm



Laplace K-means algorithm

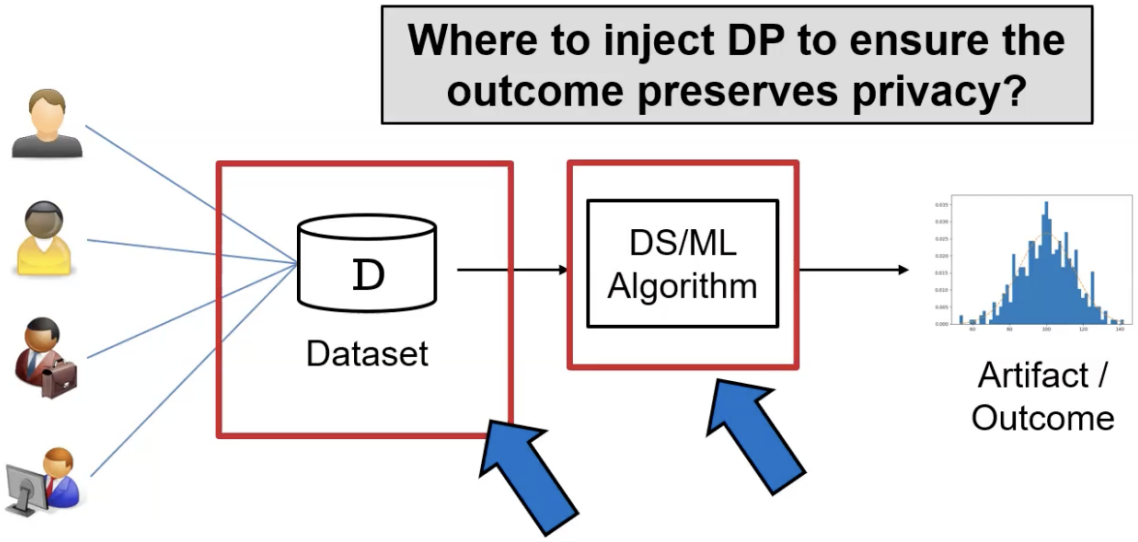


- Not ideal: the DP version has some “utility loss”
  - Utility loss is the unfortunate reality of differential privacy
  - How to reduce utility loss? ← a research question ☺

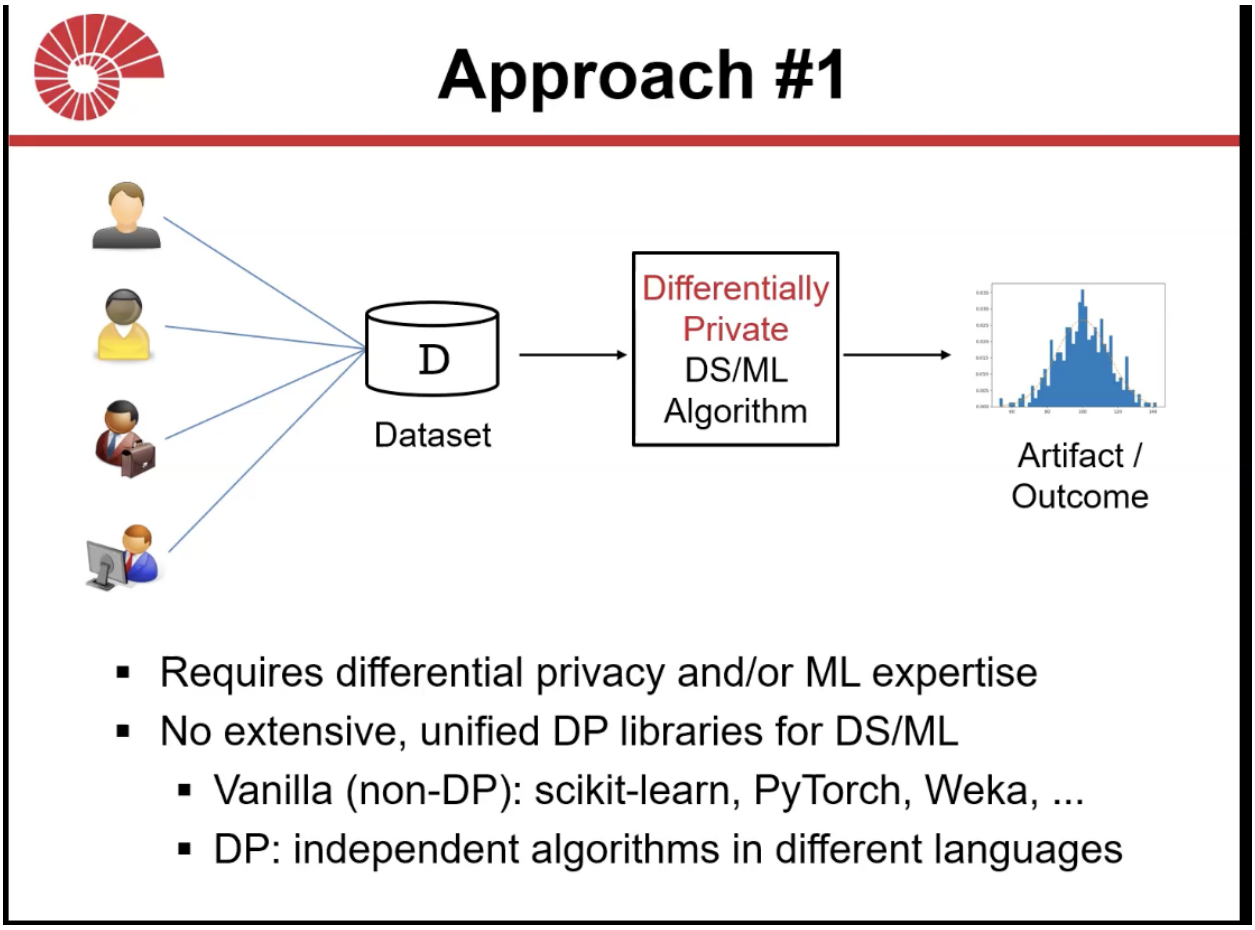
Lecture 13

Where to inject privacy?

- **Case Study #3:** Generating differentially private "synthetic data"



You can either inject privacy in the algorithm that runs with the dataset, or you can inject privacy into the dataset than use the private dataset.



Modifying this kind of algorithm, knows the expertise of differential privacy.

For all the different algorithms, we need different libraries to inject them differential privacy.

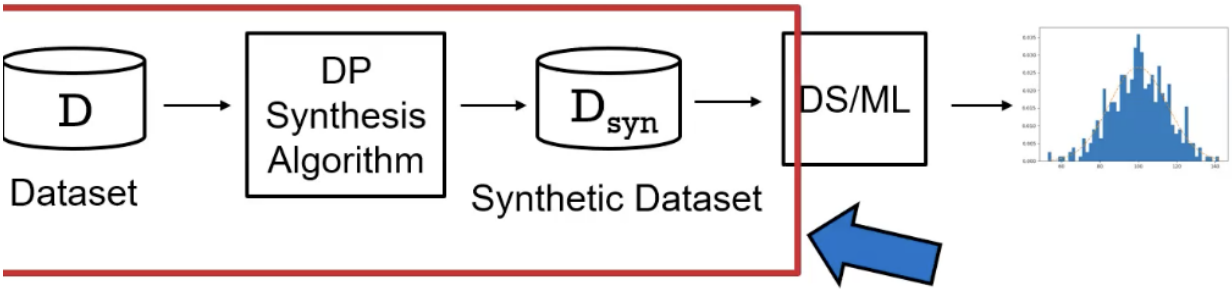
DP Synthetic Data

DP synthesis algorithm transforms the original dataset to a synthetic dataset

DS/ML on the synthetic dataset does not violate DP, due to DP's post-processing property

Can use existing vanilla DS/ML libraries!

Research challenge: How to construct a DP synthesis algorithm so that $\mathbf{D}_{\mathbf{syn}}$ resembles **D**?



The other approach, as we spoke, instead of taking the dataset directly to the DP algorithm, you can take the dataset and feed it to the DP Synthesis algorithm which will inject the dp privacy by creating synthetic dataset.

Why we need synthetic dataset ?

> None of the records can be real records corresponding to real individuals - Why? → if we output any real records our DP is not satisfied . As we spoke on function 3, if our outcome has dependance on any single input record we violate DP.

So our synthetic dataset needs to be a probabilistic approach that generates fake records that do not have one to one correspondence to any individual.

Once we can have the synthetic dataset we can use whatever method that we want without thinking about differential privacy. Because it is already satisfied.

DP for Non-Tabular Data

- How you define "**neighboring datasets**" is important

CustomerID	TransactionID	BasketContent
1	1234	{Aspirin, Panadol}
1	4234	{Aspirin, Sudafed}
2	9373	{Tylenol, Cepacol}
2	9843	{Aspirin, Vitamin C, Sudafed}
3	2941	{Tylenol, Cepacol}
3	2753	{Aspirin, Cepacol}
4	9643	{Aspirin, Vitamin C}
4	9691	{Aspirin, Ibuprofen, Panadol}
5	5313	{Panadol, Vitamin C}
5	1003	{Tylenol, Cepacol, Ibuprofen}
6	5636	{Tylenol, Panadol, Cepacol}
6	3478	{Panadol, Sudafed, Ibuprofen}

D' = D U {customer}

or

D' = D U {transaction}

or

D' = D U {basket item}?

What matters is how we are going to define neighboring datasets.

We can say to datasets can differ by one customer, or one transaction or one basket item.

Lets exercise about considering different neighboring datasets definitions.

For query q = # of aspirins bought ?

For D′ = D U {customers} = number of transactions allowed* as many as one person can buy

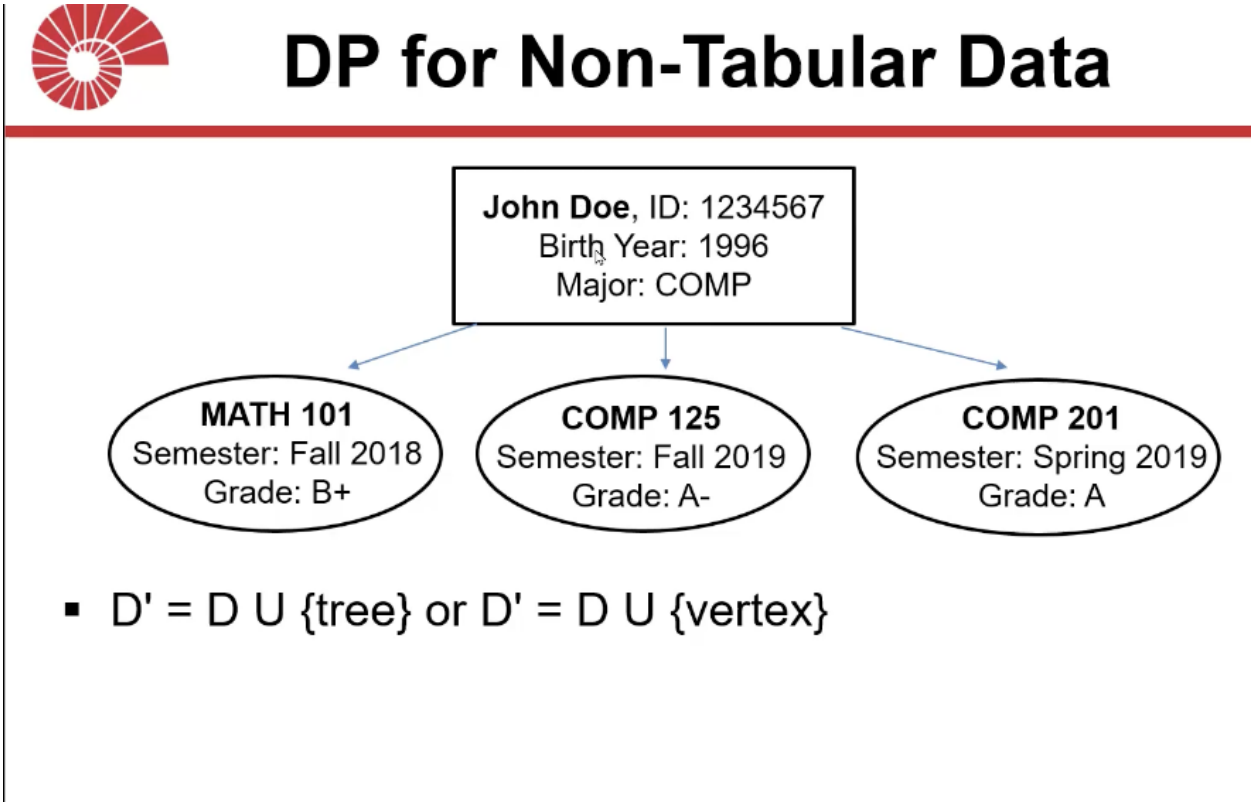
For D′ = D U {transaction} = as many as one person can buy

For D′ = D U {basket item} = 1

As we can guess, more information that we hide, our sensitivity will go up .

Our general procedure still applies:

Calculate sensitivity → use laplace/exponential to calculate noise → use composition (sequential or parallel)



We can define neighboring datasets into two ways.

For query q1 “how many people born in 1996 ? ”

For D′ = D U {tree} = 1

For D′ = D U {vertex} = 1 (think about root vertex)

For query q2 “cumulative enrollment to Comp125? ”

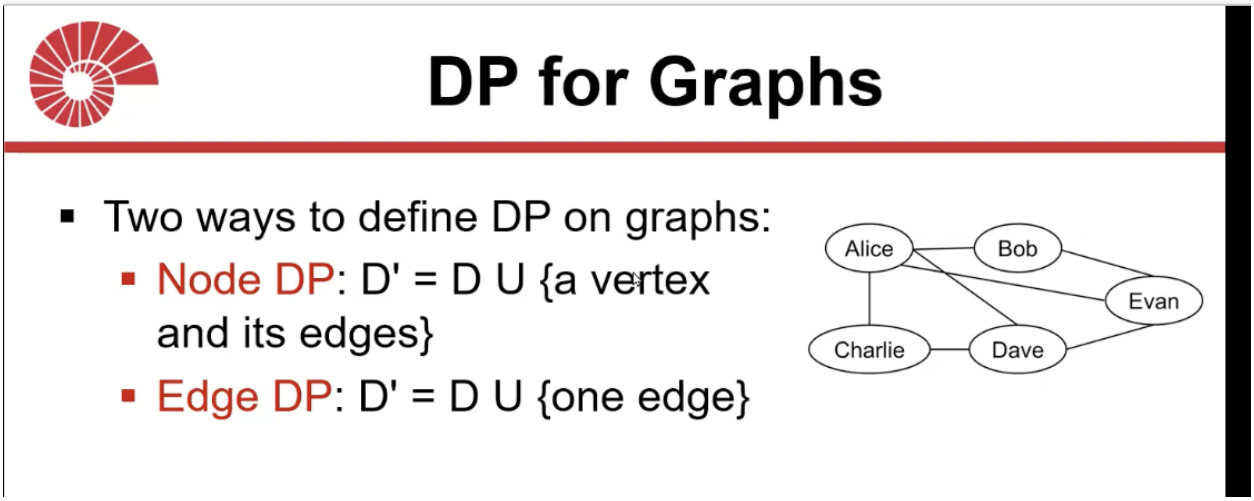
For D′ = D U {tree} = how many times allowed that one person took comp125

For D′ = D U {vertex} = 1

For query q3 “cumulative enrollment to Comp courses ”

For D′ = D U {tree} = however many times a student can take computer courses

For D′ = D U {vertex} = 1



For query q = Count # of total edges in graph

For D′ = D U {a vertex and its edges} = number of nodes (you can create a node and add edge to all of the nodes present)

For D′ = D U {one edge} = 1

For query q2 = Count # of vertices that have degree ≥n

For D′ = D U {a vertex and its edges} = number of nodes +1 (all of the previous nodes are n-1 degree, you added one node and completed all of the other nodes to n degree, and simultaneously your node has also reach n degree)

For D′ = D U {one edge} = 2 (you can add an edge between two vertices that both have n-1 degree)

- **Node DP** is inherently more difficult
  - Sensitivity is O(|V|) for many types of queries
  - Hence, **Edge DP** has received more attention in the DP literature

Sensitivity Bounding

In cases where sensitivity is unbounded or very large, we can engineer an upper bound on it.

For example:

We assume each user makes max 5 transactions

or We assume each transaction contains max 10 items.

Those are also reasonable assumptions because they are true for 99% of the users and transactions in the dataset.

We can enforce this either using sampling or truncating.

Sampling: For users with more than 5 transactions, randomly keep 5 of their transactions and discard the rest.

Truncating: For transactions that contain > 10 items, keep the first 10, discard the rest.

Case Study: Search Logs

- Search engines (eg: Google) keep users' **search logs**
  - <UserID, search_keywords, time, clicked_URL>
- Useful for: search result caching, "did you mean", re-ranking search results, advertising, ...

	User ID	Query	Time	URL
1	User1	Tax ssn 111223333	2008-01-05 08:10	
2	User2	Restaurant arlington wi	2008-01-03 10:20	local.yahoo.com/...
3	User2	Restaurant arlington wi	2008-01-03 10:22	www.gorestaurants.net/...
4	User2	70 single men	2008-01-05 14:30	
5	User2	chen family tree	2008-01-06 20:01	chenfamilytree.com
6	User3	www.some-church.com	2008-01-08 10:35	www.some-church.com
7	User3	Tax for pastor	2008-01-13 22:50	answers.yahoo.com/...

People and their search on the internet

- Can we release any raw log entry under DP?
  - No, recall our example: func3
  - We should be releasing noisy aggregate statistics

- Say that we want to construct a **histogram** of search keywords; let D be a search log dataset, D' = D U {one user}
  - Sensitivity of histogram?
    - Too high (or infinity)
  - Sensitivity bounding → Pick at most m search keywords from each user

Alice = car, bicycle

Bob = apples, bananas, koç university

Sam = Sam Altman, Open AI, Ward

Lets say we will construct a histogram of the keywords and our neighboring dataset definition to datasets that differ by one user.

What is the sensitivity of the histogram ?

Too high or infinity - because one user can make a search that whatever times he want.

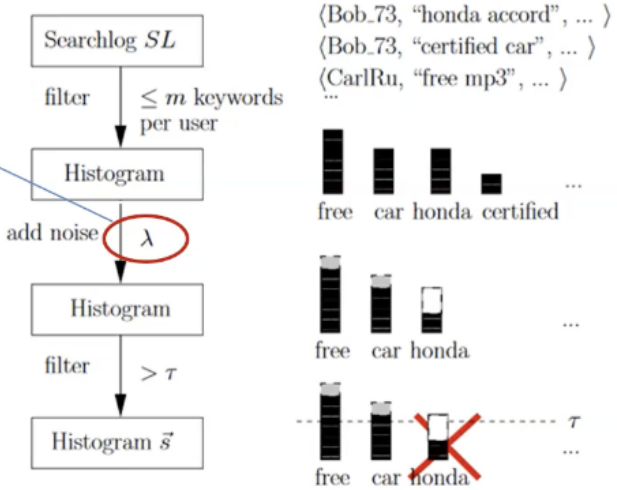
What should we do ? → we should bound the sensitivity. Pick at most m search keywords from each user

Lets say our m is 2. And lets say we use truncation, In that scenario our sensitivty becomes 2.

DP Search Logs

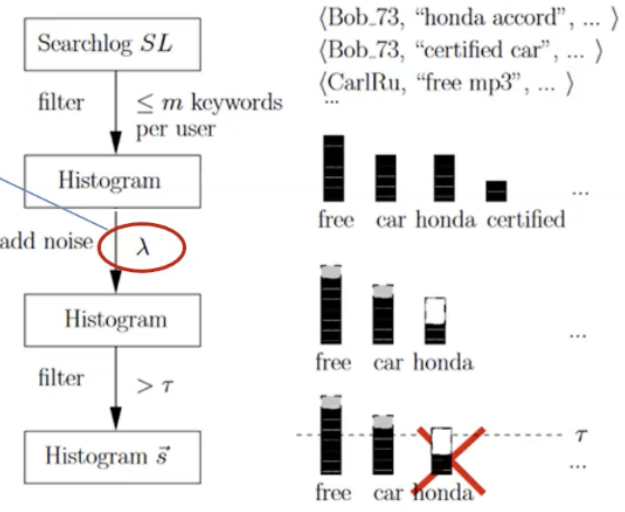
- Publishing frequent search keywords

Laplace with
scale = m/ε



- Publishing frequent search keywords

Laplace with
scale = m/ε



How about we asked for the top-k keywords (not their counts)?

Same question, but you can't use Laplace?

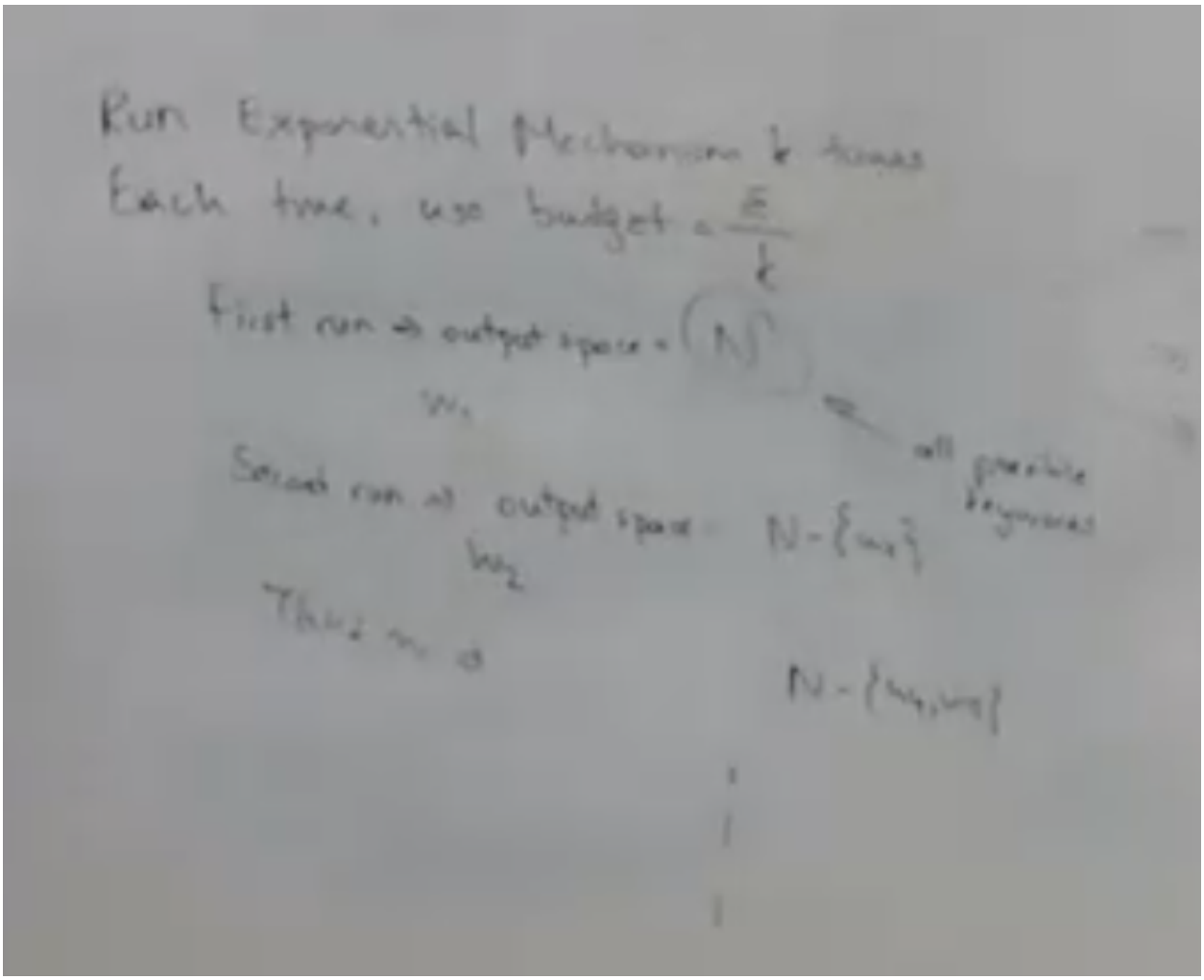
Let say we have a search log like this, we apply this kind of truncation so our sensitivity is balanced by m. We are wanted to give top k most popular keywords in terms of number of occurrence but we are not allowed to Laplace Mechanism.

We have to use another differential private mechanism to solve this problem.

Exponential mechanism is a good idea. How can we use exponential mechanism as to find top k ?

We should set up a exponential mechanism in a way that output space of the exponential mechanism is all possible keywords. Score functions should be show frequency of the keywords. Sensitivity of our score function is m (whatever bound that we are using) . When we execute this exponential mechanism an it gives us a randomized choice and it gives the most popular keyword. But we are wanted to give top-k keyword.

In that case algorithm is follows like this



At every run, remove the output from the output space. And run again with same logic do this for k times

💡 In terms of computational time it is more expensive then laplacian. But it has one advantage in situation like this

Probability of returning a more optimal value is exponentially higher. Whereas in the laplacian you adding values to indivual counts so relation is linear.

DP-SGD

- **DP-SGD:** Making deep learning differentially private
  - Differentially private stochastic gradient descent

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

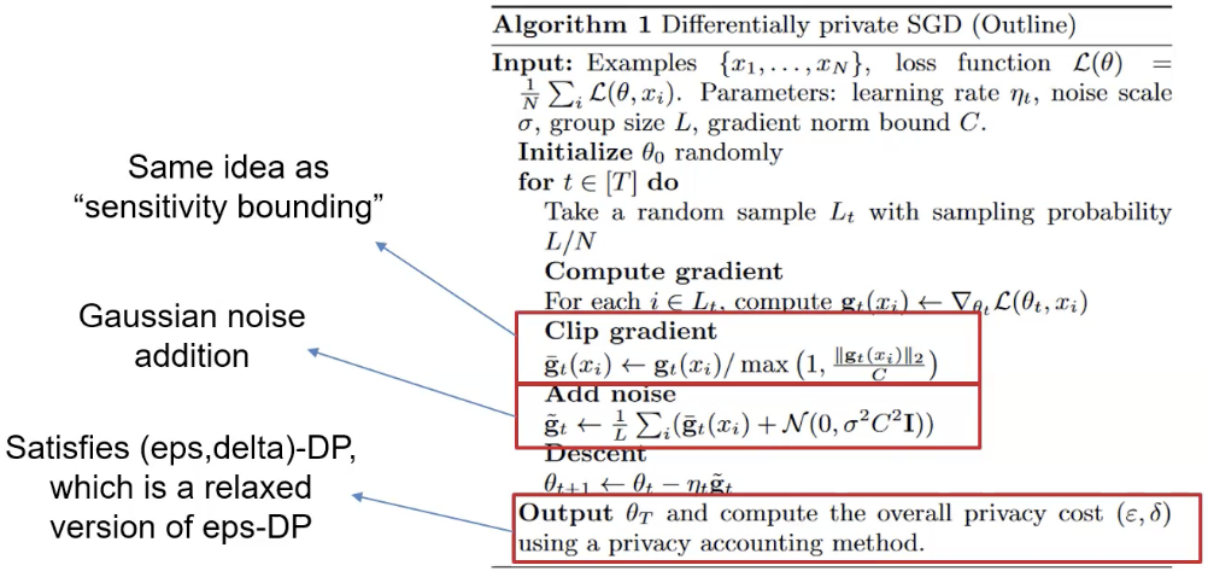
Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\sum_i (\bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

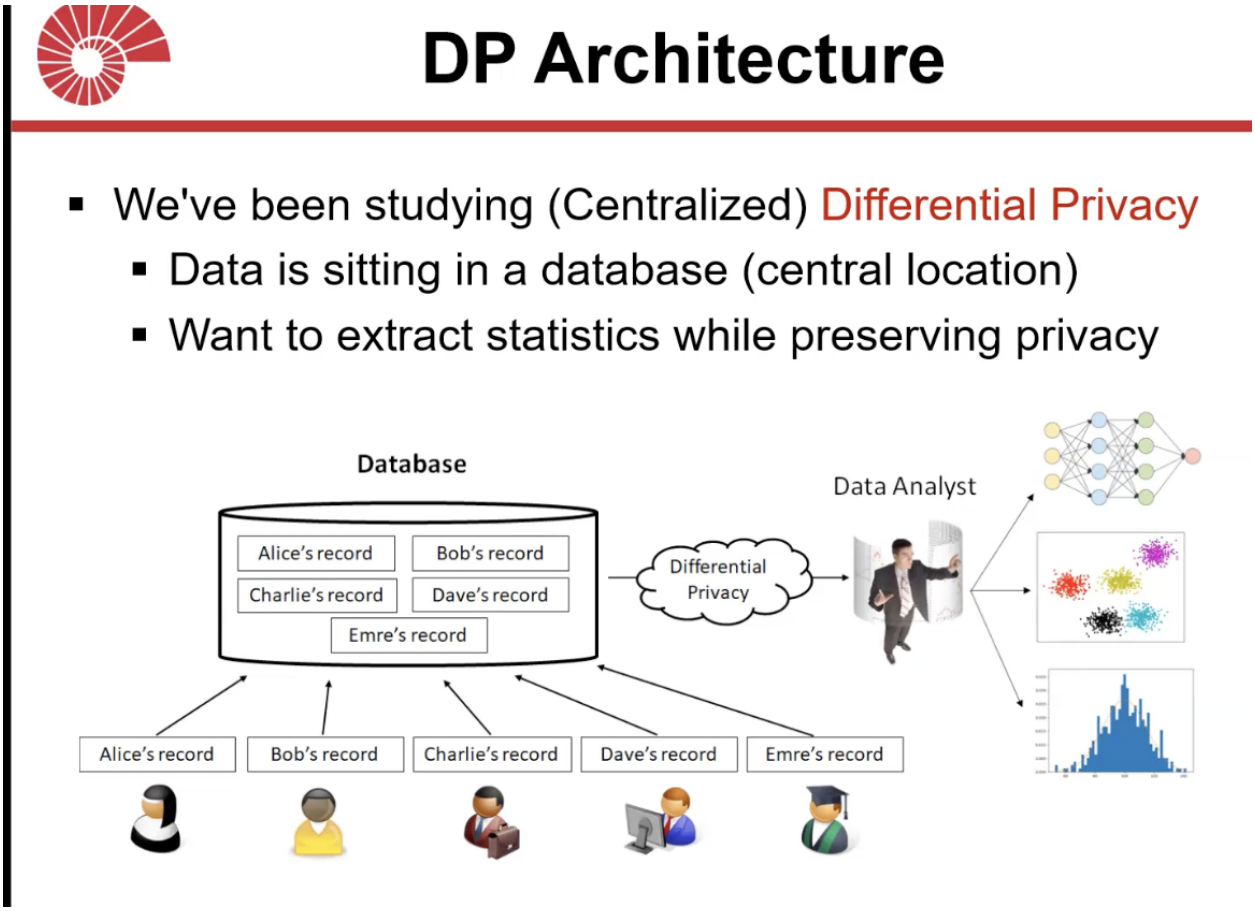


Conclusion

- **What did we learn?**
  - Checking if DP holds or not
    - If it holds, we must be able to prove it for all D, D', O
    - If it doesn't hold, we must provide a counter-example
  - Extracting simple statistics with DP, e.g., histograms
  - Converting non-private complex algorithms into DP algorithms, e.g., k-means, DP-SGD
  - DP for non-tabular data
    - And the potential variations in sensitivity
  - The "sensitivity bounding" trick
    - Enforced via sampling and truncating

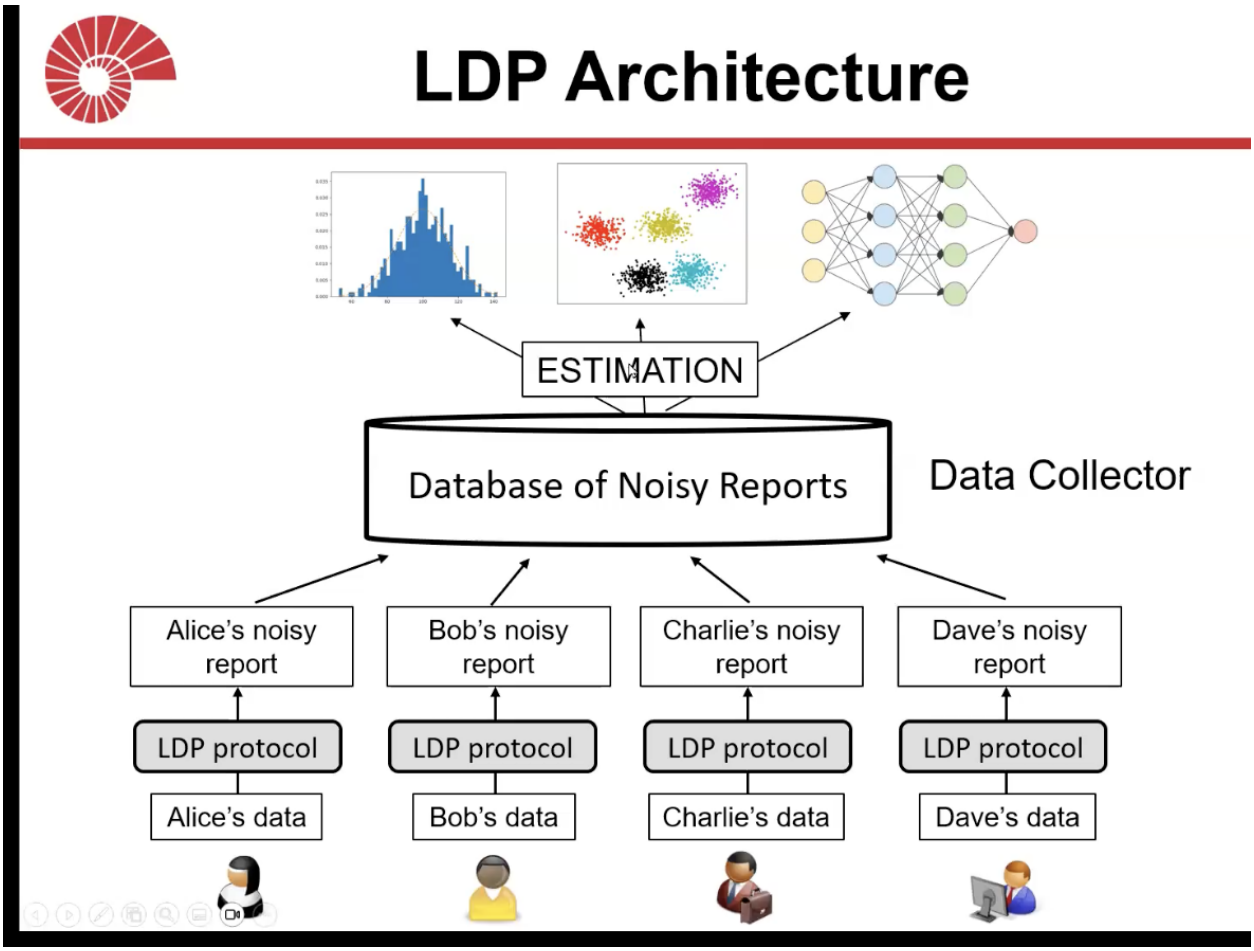
Lecture 14

Local Differential Privacy



Shortcomings of DP

- Nowadays much data is sitting on users' smart devices in distributed fashion, not in a centralized location
  - Smartphone data vs hospital database
- Service providers want to **collect** this data
- **DP:** Collect the data first, then apply privacy
  - Data collector is untrusted (do you trust Google?)
  - How can the user know if the DP layer will be implemented correctly and always enforced?
  - Single point of failure: side-channel attacks, data breaches, ...
- **LDP:** Apply privacy on user's device first, then collect



So, in the differential privacy that we learn which is the generalized version works for a data sit on the single dataset. But in nowadays, we dont have such structures, we all have data that is wanted from service provider conversely wanted from the database. For example our data in the mobile devices is wanted, but we dont want to give it with real way, to them to privatize because we dont trust them. So we need a ways to implement local differential privacy in our devices and give the noisy data.

"Local" DP

- Each user applies DP on their own local device, prior to data collection
  - Then combine all results to get a final answer
  - Server never observes users' real data

- But how about **utility**?
  - Instead of adding noise once, add noise m times
  - Fortunately, noise can be cancelled out or subtracted out (this is done in the estimation phase)
  - Expected noise in LDP is still higher than centralized DP

$$\mathrm{Var}\left(\sum_{i=1}^{m} X_i\right) = \sum_{i=1}^{m} \mathrm{Var}(X_i)$$

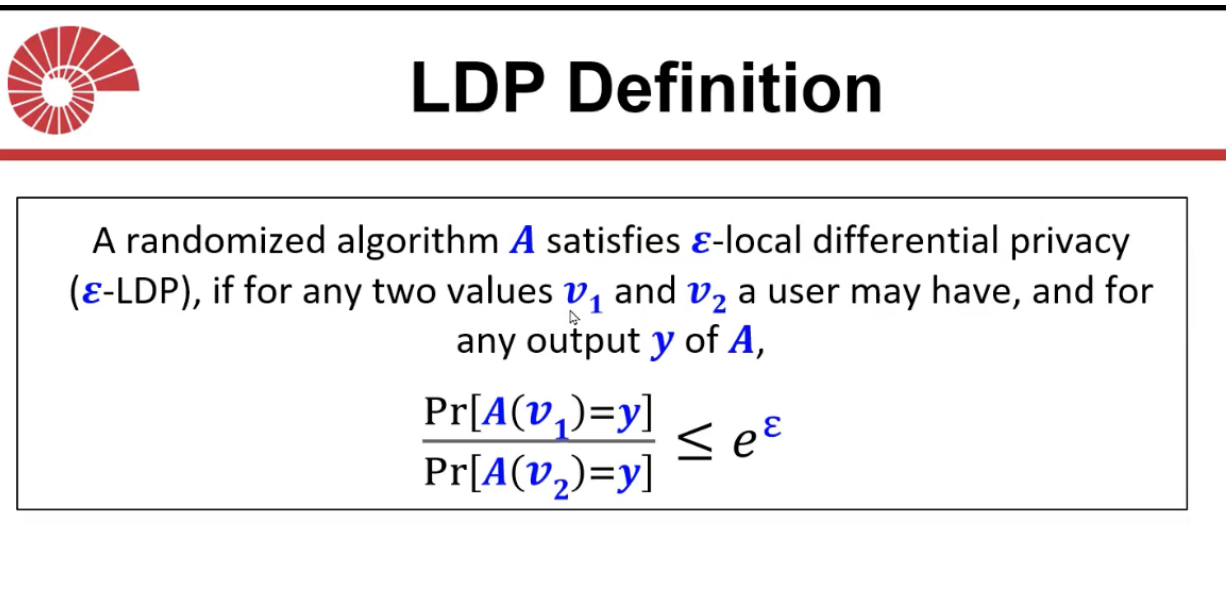
What about utility ?

So in the generalized version we apply the noise just once. So for example in the laplacian we have one random variable.

But in the local DP we should add noise one by one.That is increasing noise much and this is the key challenge in the local DP We have 1 random variables in the generalized one for the dataset, but in the local dp we have random variables for # of users. In the general Dp variance of our noise comes from single random variable , in local DP variance is summation of N (#of user's) random variable.

$$\mathrm{Var}\left(\sum_{i=1}^{m} X_i\right) = \sum_{i=1}^{m} \mathrm{Var}(X_i)$$

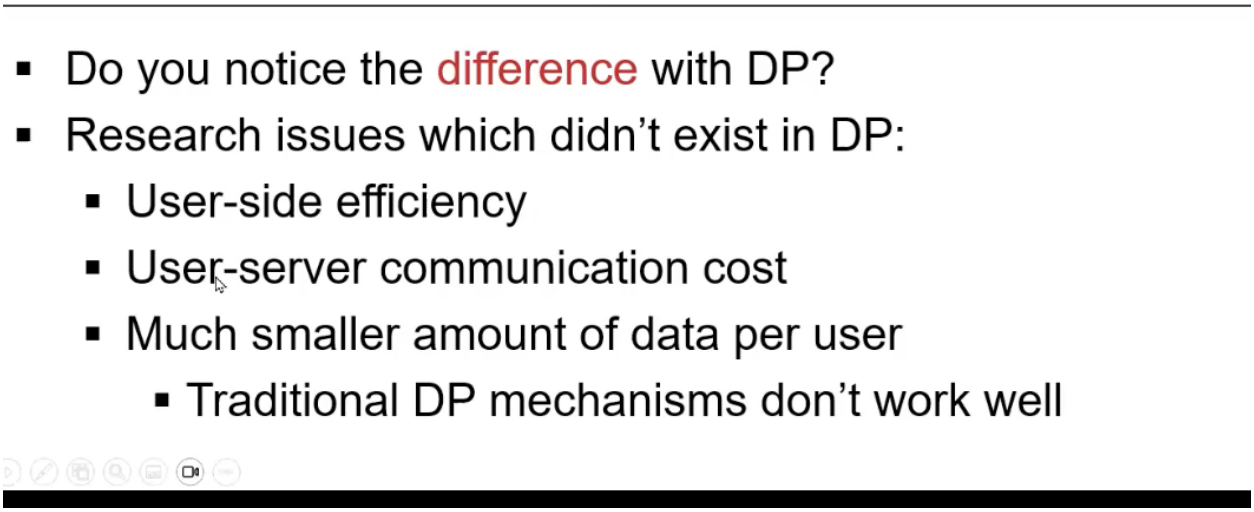
Those are equal, so this is the proof we add more noise



Key difference is;

At the DP , we are executing our algorithms on datasets.

In LDP, we are executing our algorithms on individual values.



Algorithms should be simple because it is runned on person's device. Output should also be small in order to communicate easily

Algorithms that run well on generalized differential privacy were not work well on the local differential privacy. We have to come up with new algorithms

Randomized Response

- Building block of many LDP protocols
- Decades old survey collection technique
  - First proposed by Warner in 1965

- **Data collector:** Did you vote for the communist party?
- **Each person:**
  - Flip a secret coin
  - If heads, answer truthfully; if tails, flip coin again
  - If heads, answer "yes"; if tails, answer "no"

User's answer is truthful with what probability?

Plausible deniability

People who answer this have %75 change to say truth. %50 from first coin flip and %25 from second coin flip.

But he has %50 change of deniability because the second coin flip has give %25 percent denial rate

Randomized Response

- For any individual user, his/her true answer cannot be learned by the data collector with 100% certainty
- But the data collector can recover aggregate statistics pertaining to the general population
  - **If n_v out of n people indeed voted for the communist party, how many "Yes" answers do we expect to see?**

$$E[I_v] = 0.75n_v + 0.25(n - n_v)$$

  - **Then, the following is an "unbiased estimate" of the # of communist party voters based on the survey:**

$$c(n_v) = \frac{I_v - 0.25n}{0.5}$$

$$E[c(n_v)] = \frac{E[I_v] - 0.25n}{0.5} = \frac{0.75n_v + 0.25(n - n_v) - 0.25n}{0.5} = n_v$$

If we know that everybody answers with this logic, data collector can recover aggregate statistics relating to the general population

n = # of users who answer the question

nv = # of users whose true value = v

Iv = # of users whose reported value = v

From the n people, if nv people indeed voted the communist party. How many “yes” answers do we expect to see when a question asked to n people “Did you vote for the communist party ?”

Expectation of Iv is asked

E[Iv] = nv * 3/4 + (n-nv)*1/4

that means that, from the

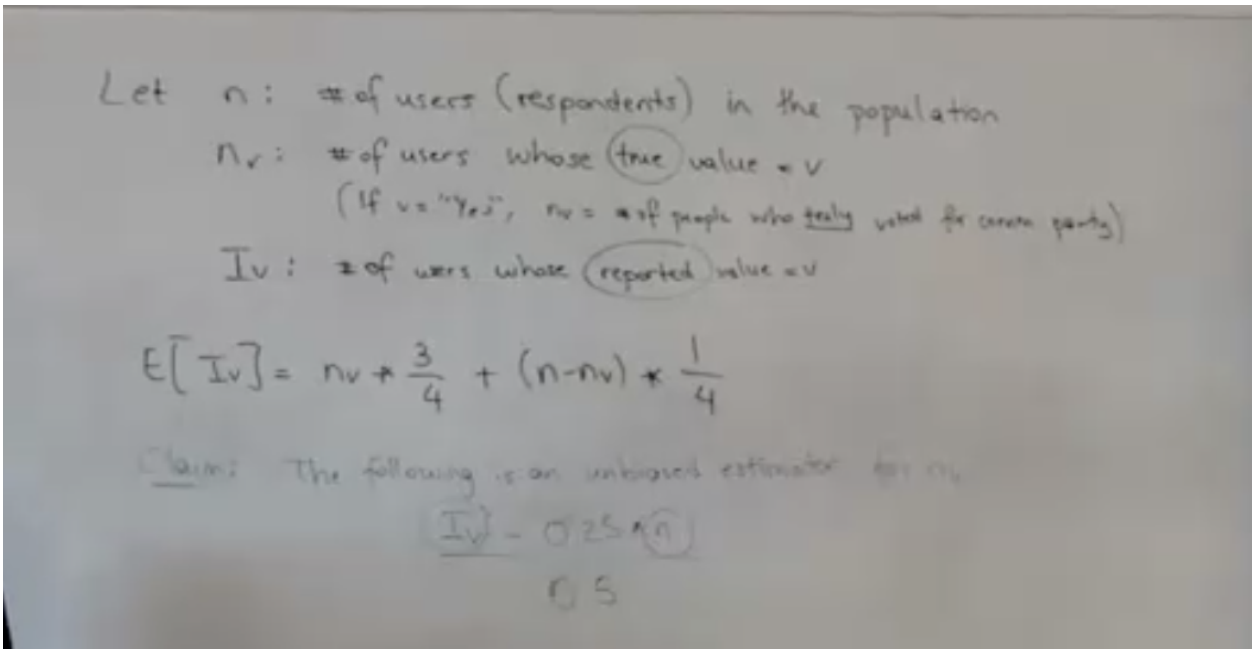
unbiased estimate = is what we are trying to find

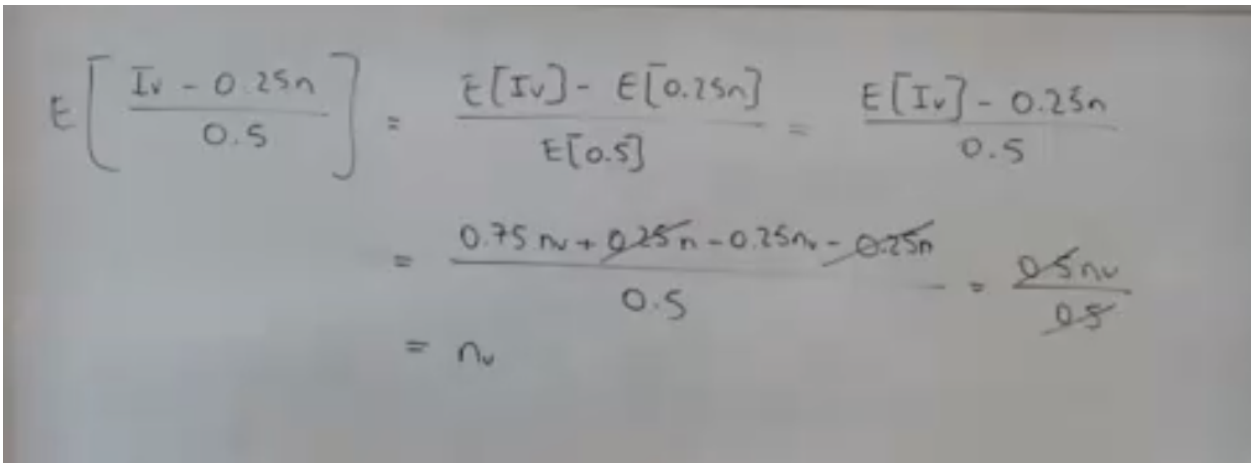
There is a applicant equation that gives unbiased estimate.

$$c(n_v) = \frac{I_v - 0.25n}{0.5}$$

Take the expectation, of the Iv-0.25n/0.5 and see does it give the nv.

It actually does





RR Example

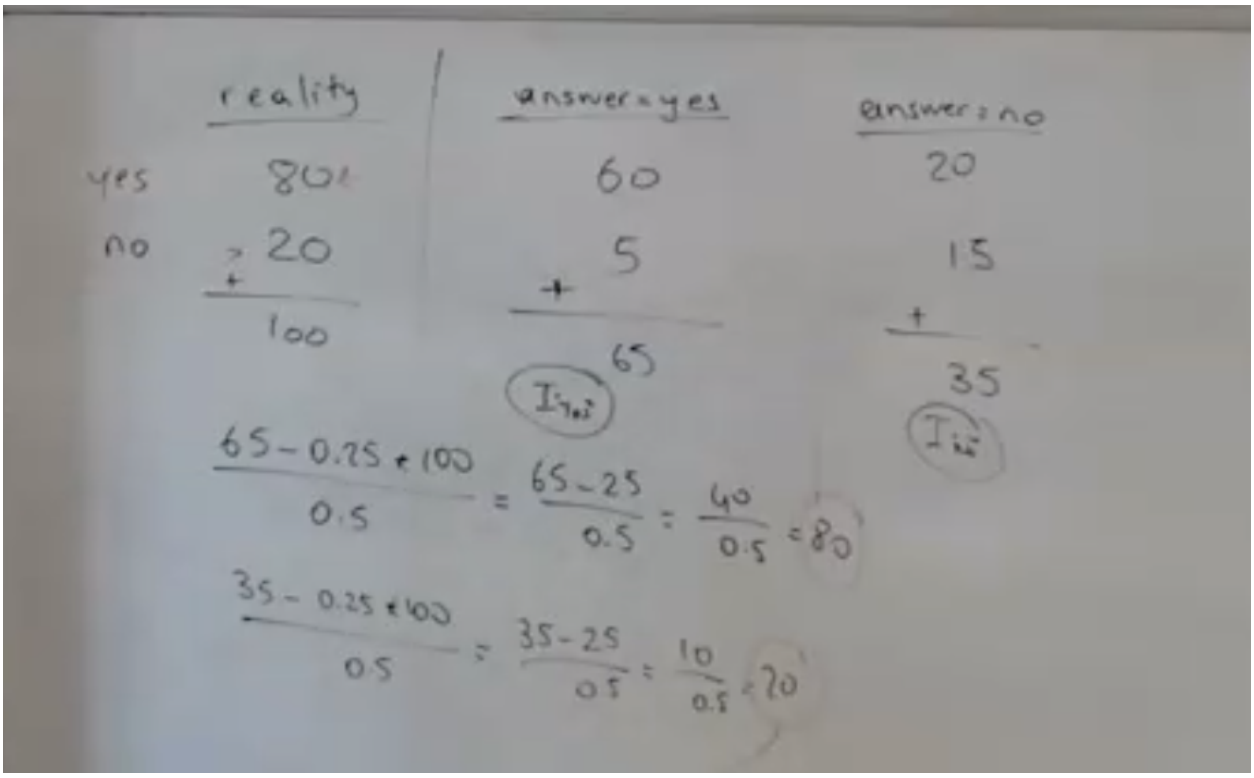
- Communist party voter answers **"yes"** w.p. 75% and **"no"** w.p. 25%

	truth	->yes	->no
yes	80	40+20	0+20
no	20	0+5	10+5

$$c(n_v) = \frac{I_v - 0.25n}{0.5}$$

observed	65	35
estimate	80	20

Seems like this whole process is working perfectly…
What causes the “error” or the “utility loss”?

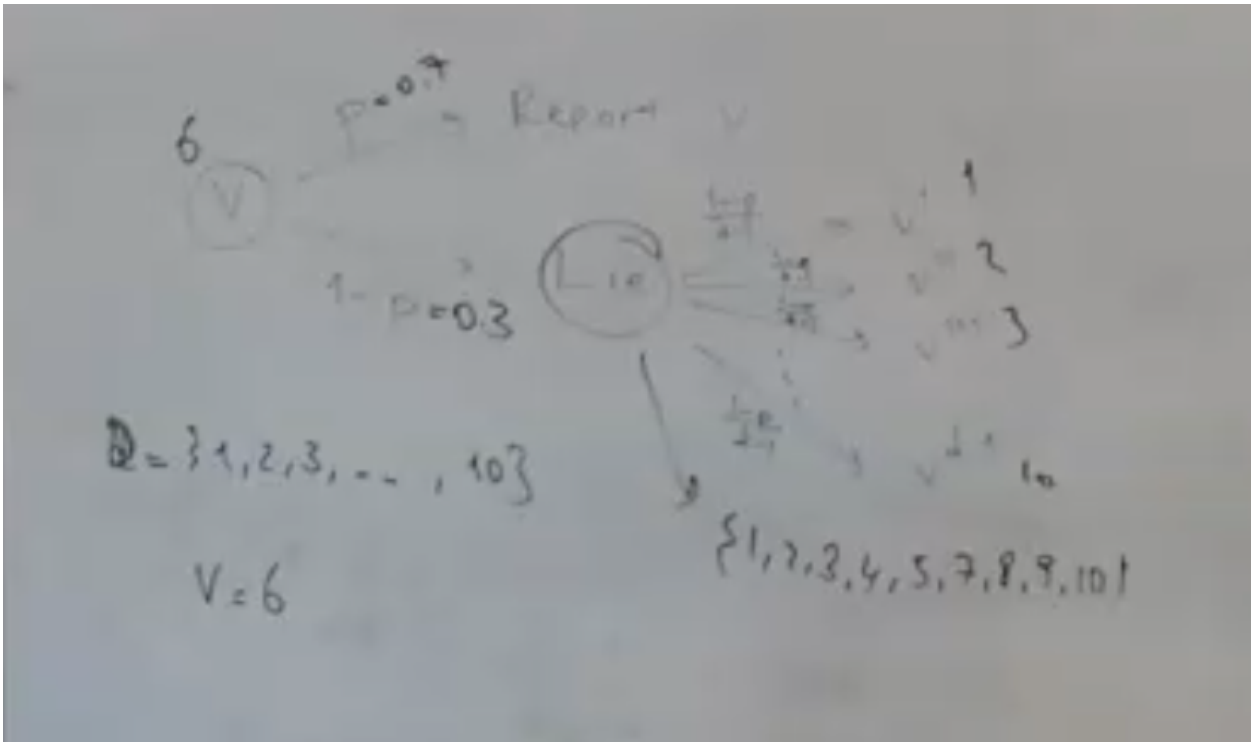


Utility lost comes because we assume that all the answered yes and answered no comes from the result of the coin flips which all the coins are not biased. But in reality there are noises in the fields. Not 60, we get 62 and not 5 we get 7 and so on.

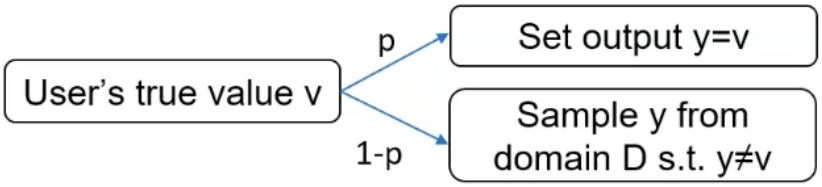
Generalized RR

- **Generalized Randomized Response (GRR)**
  - An LDP protocol to support non-binary answers and arbitrary lying probability (controlled by ε)

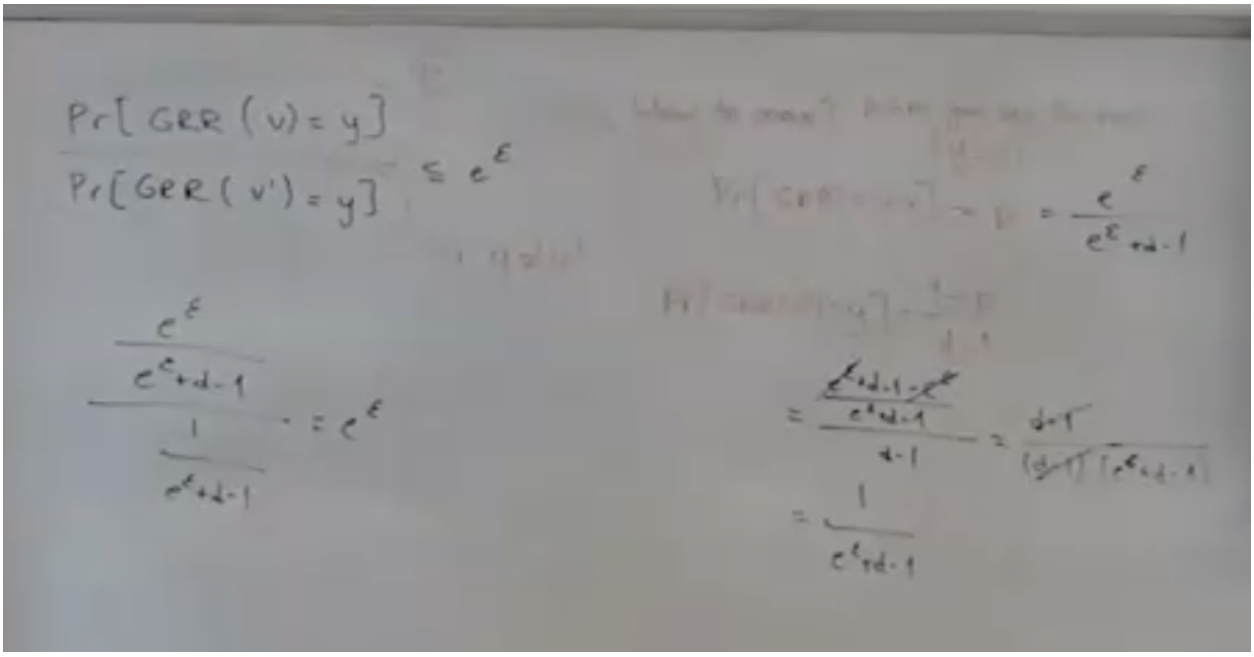
- **User's reporting strategy in GRR:**
- Let user's value v be from domain $D = \{1,2, \dots, d\}$
- Toss a biased coin, prob of heads = p
  - If it comes up heads, report true value: y = v
  - If it comes up tails, pick reported value y uniformly at random from D \ {v}
    - Probability that a certain fake item is reported is: $q = \frac{1-p}{d-1}$



If p is set in this way, than this algorithm satisfies epsilon local differential privacy.



- Satisfies ε-LDP when $p = \frac{e^{\varepsilon}}{e^{\varepsilon}+d-1}$
- Why? Let's do the proof…



In that proof, we write the LDP imposition. We first think how can we choose the max value for this nominator. This is the strategy because we are trying to find max value for nominator and min value for the denominator, therefore the number on the left hand side will be larger value that it will get. Even if can smaller than the right hand side, this proof continous for all.

So to become max, nominator can be truth value ⇒ p

And at the denominator (1-p) for the lying and d-1 for choosing from the lying ones. ⇒ (1-p)/(d-1).

So put the value for p and see the it holds the LDP imposition.

Lecture 15

On the previous lecture, we take care of the user’s side. User’s data is privatized with noise in the user’s side. Now lets talk about how the server aggregate that data and made estimation.

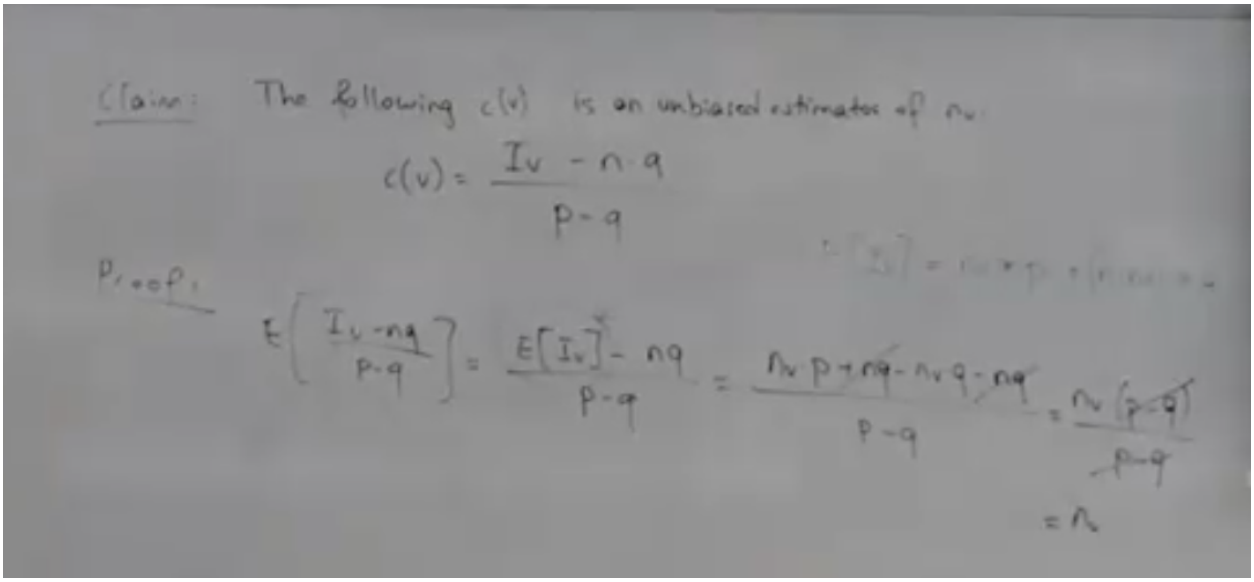
Generalized RR

- **Server-side aggregation + estimation:**

- n_v users possess value v, I_v is the number of reports
- $E[I_v] = n_v \cdot p + (n - n_v) \cdot q$

- Unbiased estimator: $c(v) = \frac{I_v - n \cdot q}{p - q}$

- How do we prove this estimator is unbiased?



Challenge in GRR

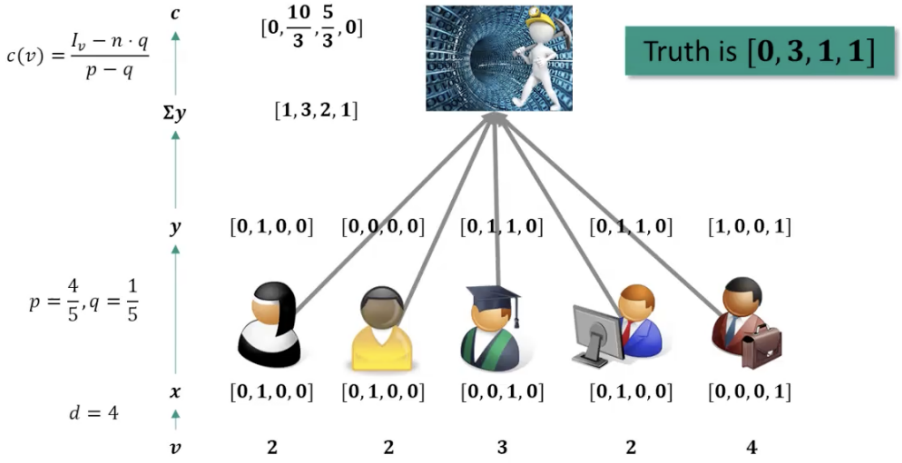
- Intuitively, higher p is good for utility $p = \frac{e^{\varepsilon}}{e^{\varepsilon} + d - 1}$
- However, when d is large, p becomes small
  - In large domains, GRR works poorly!

ε	$p(d = 2)$	$p(d = 8)$	$p(d = 128)$	$p(d = 1024)$
0.1	0.52	0.14	0.009	0.001
1	0.73	0.28	0.021	0.003
2	0.88	0.51	0.055	0.007
4	0.98	0.89	0.301	0.051

- Can we get rid of the dependency on d?
  - Yes, protocols such as RAPPOR and OUE do this

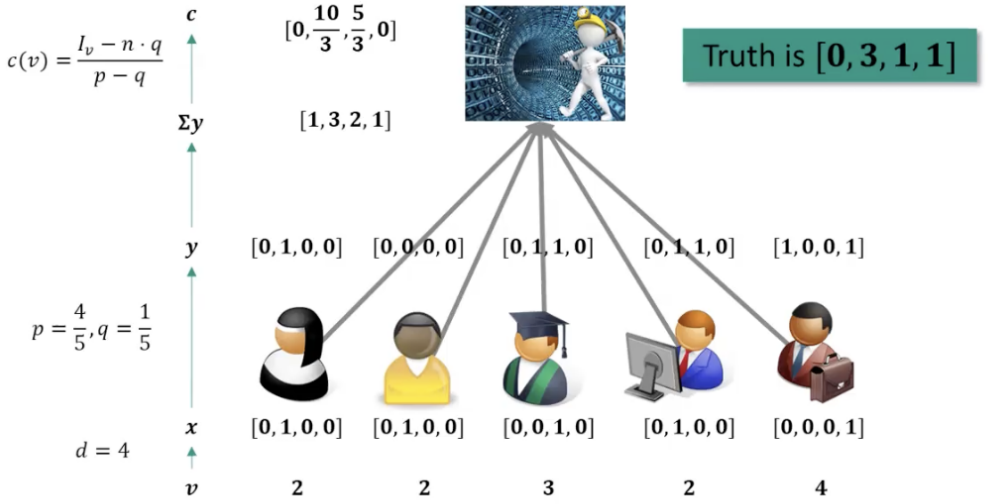
Simple RAPPOR

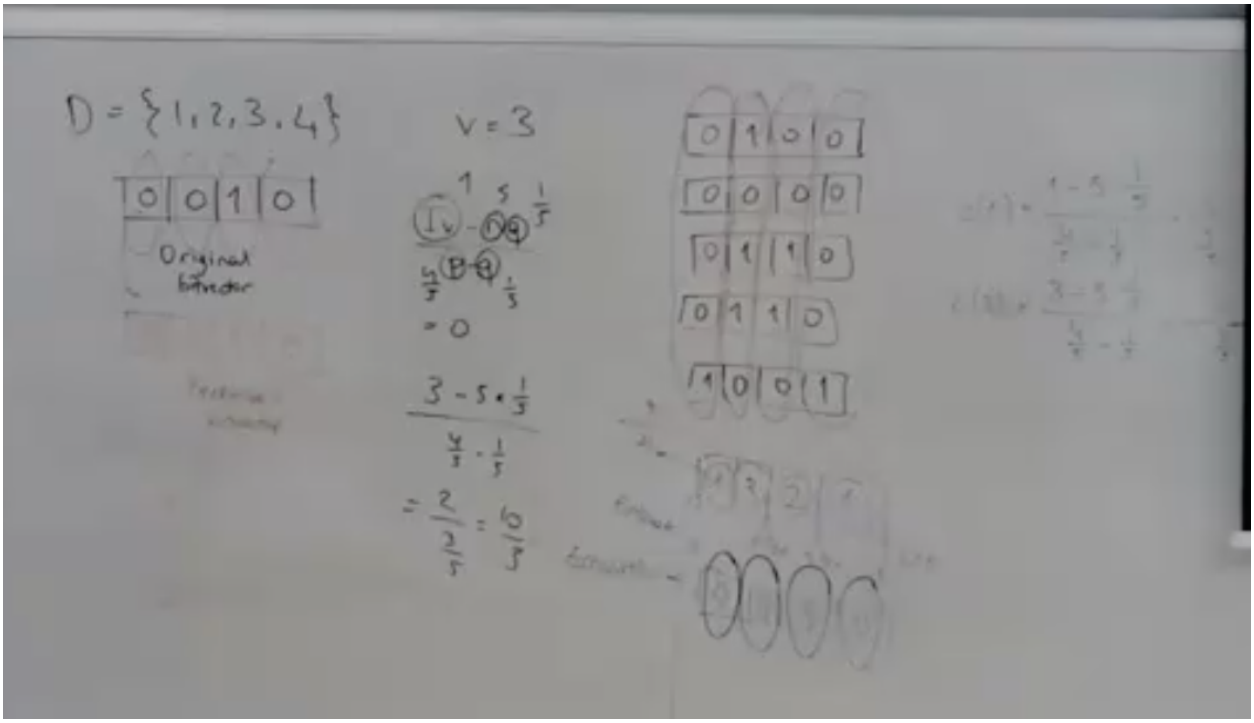
- Server-side estimation formula stays the same
  - Must be applied index by index
  - p and q are different in RAPPOR (versus GRR)



Simple RAPPOR

- Server-side estimation formula stays the same
  - Must be applied index by index
  - p and q are different in RAPPOR (versus GRR)





The data is taken from the users, n is the total number of user, q and p are the probabilities,, Iv is the data which is given at the total.

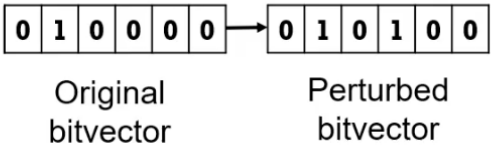
In this way you are calculating server estimation.

Simple RAPPOR

- **RAPPOR:** Developed by Google, implemented in Chrome
  - Here is a simple version of RAPPOR

- **User's reporting strategy**:
- Encode true value v into a bitvector $\boldsymbol{x} \coloneqq \vec{0}, \quad \boldsymbol{x}[v] \coloneqq 1$
  - E.g.: $D = \{1,2,3,4\}, \quad v = 3, \quad$ then x = [0,0,1,0]
- Perturb bitvector bit-by-bit
  - Preserve bit w.p. p
    - (1->1 or 0->0)
  - Flip bit w.p. q
    - (1->0 or 0->1)

$$p = \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1} \quad q = \frac{1}{e^{\varepsilon/2}+1}$$

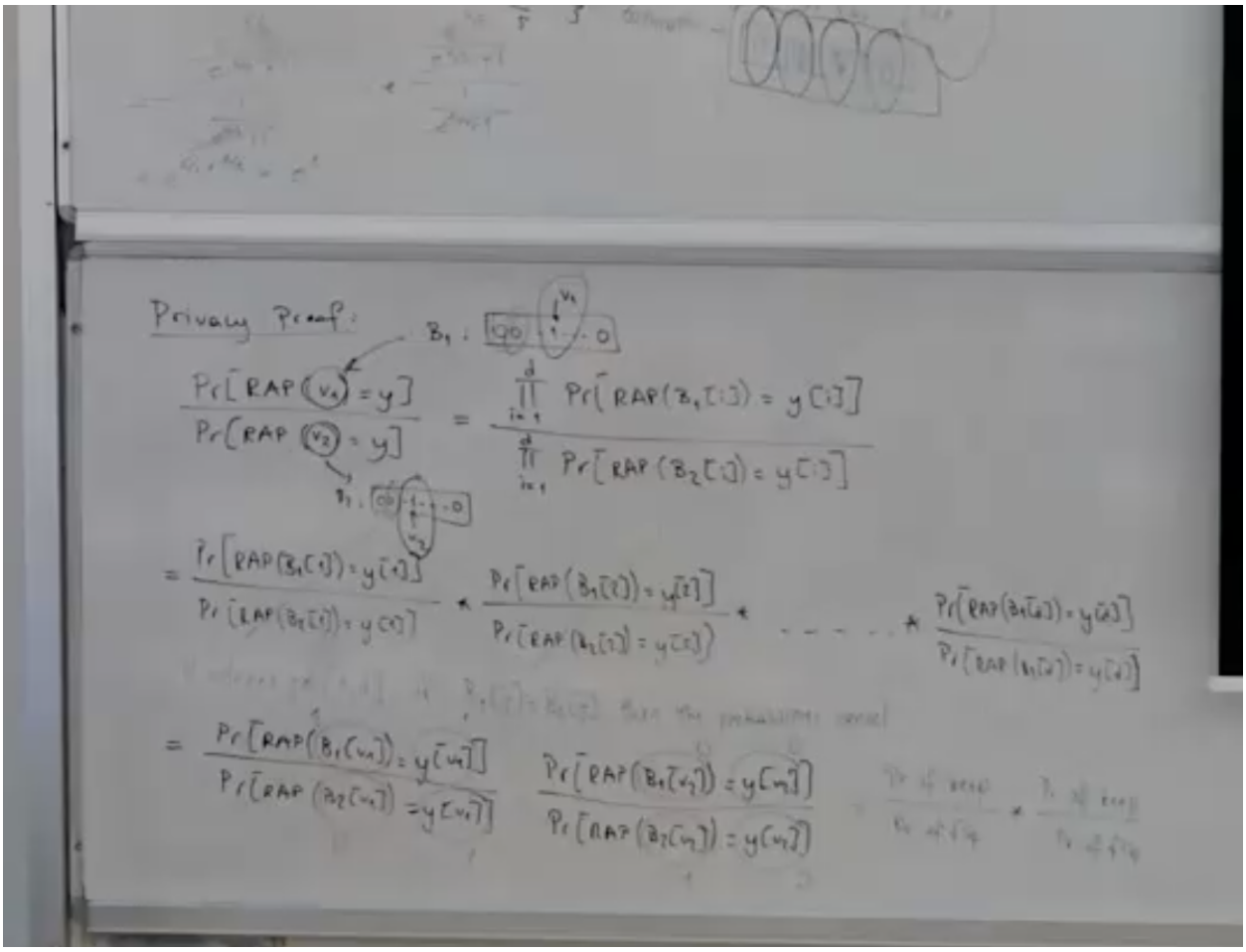


Proof of RAPPOR satisfiy LDP

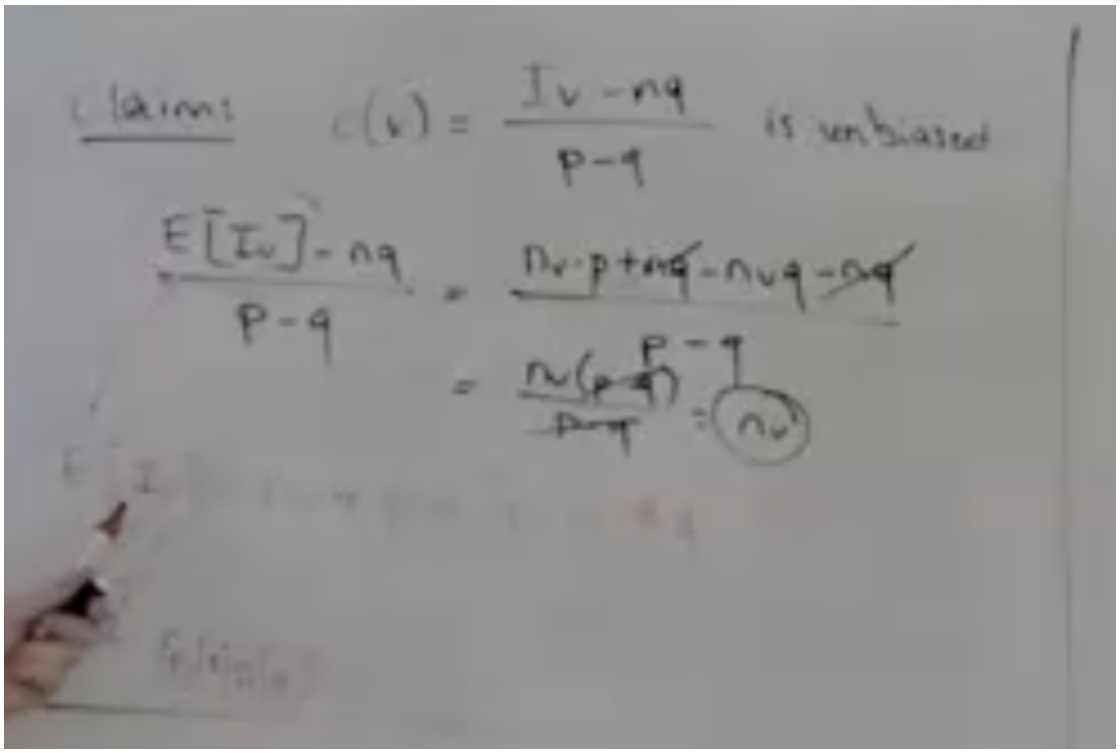
- How do you prove that Simple RAPPOR satisfies ε-LDP?
  - What is the "input" of the protocol?
  - What are the possible outputs?
  - What is the max probability odds ratio that can happen?

$$\frac{\Pr[A(v_1)=y]}{\Pr[A(v_2)=y]} \leq e^{\varepsilon}$$

Ikı adet proof var, birinci LDPyi sağlaması, ikincisi ise unbiased estimtor.



Burada pay yüksek olcak o yüzden değişmeme olasılığı olan p yi alıyosun payda ilk başta, 1→1 / 0→1 oluyor yani p/q, ikincide output 0 0→0 olması / 1→0 olması yani y,ine p/q bunları çarpınca bakıyosun ki sağlıyor e to epsilonu


Claim: $c(v) = \frac{Iv - nq}{p-q}$ is unbiased
$\frac{E[Iv] - nq}{p-q} = \frac{nv \cdot p + nq - nvq - nq}{p-q}$
$= \frac{nv(p-q)}{p-q} = nv$

E[Iv] = nv * p + (n-nv)* q

OUE

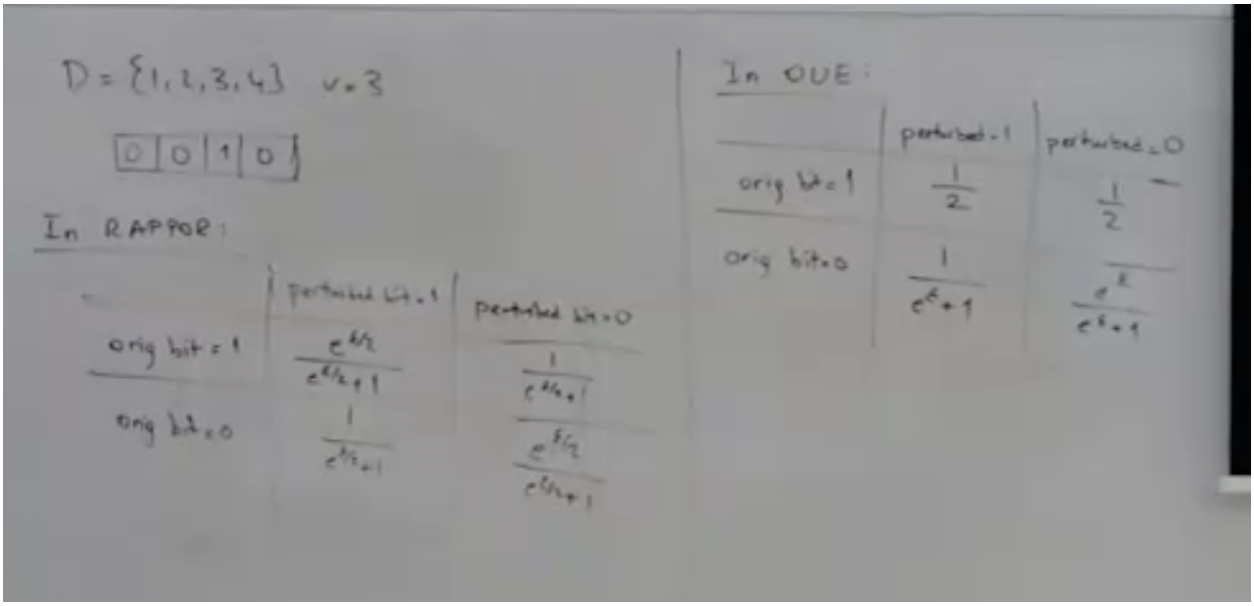
- **Optimized Unary Encoding (OUE)**
  - User's true value is encoded into a bitvector
    - This step is same as Simple RAPPOR
  - When perturbing bit-by-bit, the bit keeping and bit flipping probabilities are different than RAPPOR
    - Uneven treatment of 1 bit vs 0 bits

RAPPOR:

$$\Pr[B'_\ell[i] = 1] = \begin{cases} \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1} & \text{if } B_\ell[i] = 1 \\ \frac{1}{e^{\varepsilon/2}+1} & \text{if } B_\ell[i] = 0 \end{cases}$$

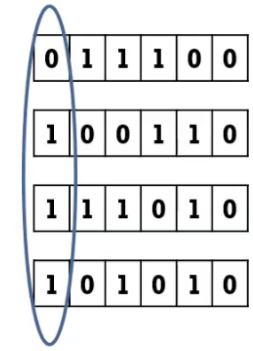
OUE:

$$\Pr[B'_\ell[i] = 1] = \begin{cases} \frac{1}{2} & \text{if } B_\ell[i] = 1 \\ \frac{1}{e^{\varepsilon}+1} & \text{if } B_\ell[i] = 0 \end{cases}$$



OUE

- Server-side estimation in OUE:
  - Bit-by-bit (position-by-position)
  - Since user-side bit keeping/flipping probabilities are different, server-side estimation formula is also different



$\widehat{C}(1) = 3$

Observed number of 1s in position i

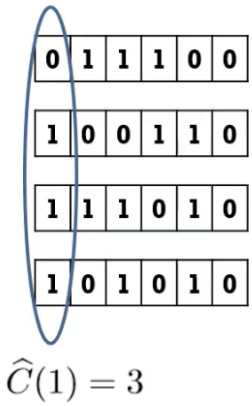
# of users

$$\bar{C}(i) = \frac{2 \cdot \left((e^{\varepsilon} + 1) \cdot \widehat{C}(i) - n\right)}{e^{\varepsilon} - 1}$$

Estimated number of 1s in position i

OUE

- Server-side estimation in OUE:
  - Bit-by-bit (position-by-position)
  - Since user-side bit keeping/flipping probabilities are different, server-side estimation formula is also different



$\widehat{C}(1) = 3$

Observed number of 1s in position i

# of users

$$\bar{C}(i) = \frac{2 \cdot \left((e^{\varepsilon} + 1) \cdot \widehat{C}(i) - n\right)}{e^{\varepsilon} - 1}$$

Estimated number of 1s in position i

He is going to ask the proof of estimated nubmer

$$\bar{C}_{(i)} = \frac{2 \cdot \left[(e^{\varepsilon}+1) \cdot \hat{C}_{(i)} - n \right]}{e^{\varepsilon}-1}$$

$$E(\hat{C}_{(i)}) = \frac{1}{2} \cdot \bar{C}_i + \frac{1}{1+e^{\varepsilon}} \cdot (n - \bar{C}_i)$$

$$E\left[\frac{2\left[(e^{\varepsilon}+1) \cdot \hat{C}_{(i)} - n \right]}{e^{\varepsilon}-1} \right] = \frac{2\left[(e^{\varepsilon}+1) \cdot E(\hat{C}_{(i)}) - n \right]}{e^{\varepsilon}-1}$$

Plug the $E(\hat{C}_{(i)})$

$$= \frac{2(e^{\varepsilon}+1) \cdot E(\hat{C}_{(i)}) - 2n}{e^{\varepsilon}-1}$$

$$= \frac{2(e^{\varepsilon}+1) \cdot \left(\frac{\bar{C}_i}{2} + \frac{(n-\bar{C}_i)}{1+e^{\varepsilon}} \right) - 2n}{e^{\varepsilon}-1} = \frac{(e^{\varepsilon}+1)\bar{C}_i + 2(n-\bar{C}_i) - 2n}{e^{\varepsilon}-1}$$

$$= \frac{e^{\varepsilon}\bar{C}_i + \bar{C}_i + 2n - 2\bar{C}_i - 2n}{e^{\varepsilon}-1} = \frac{\bar{C}_i (e^{\varepsilon}-1)}{e^{\varepsilon}-1}$$

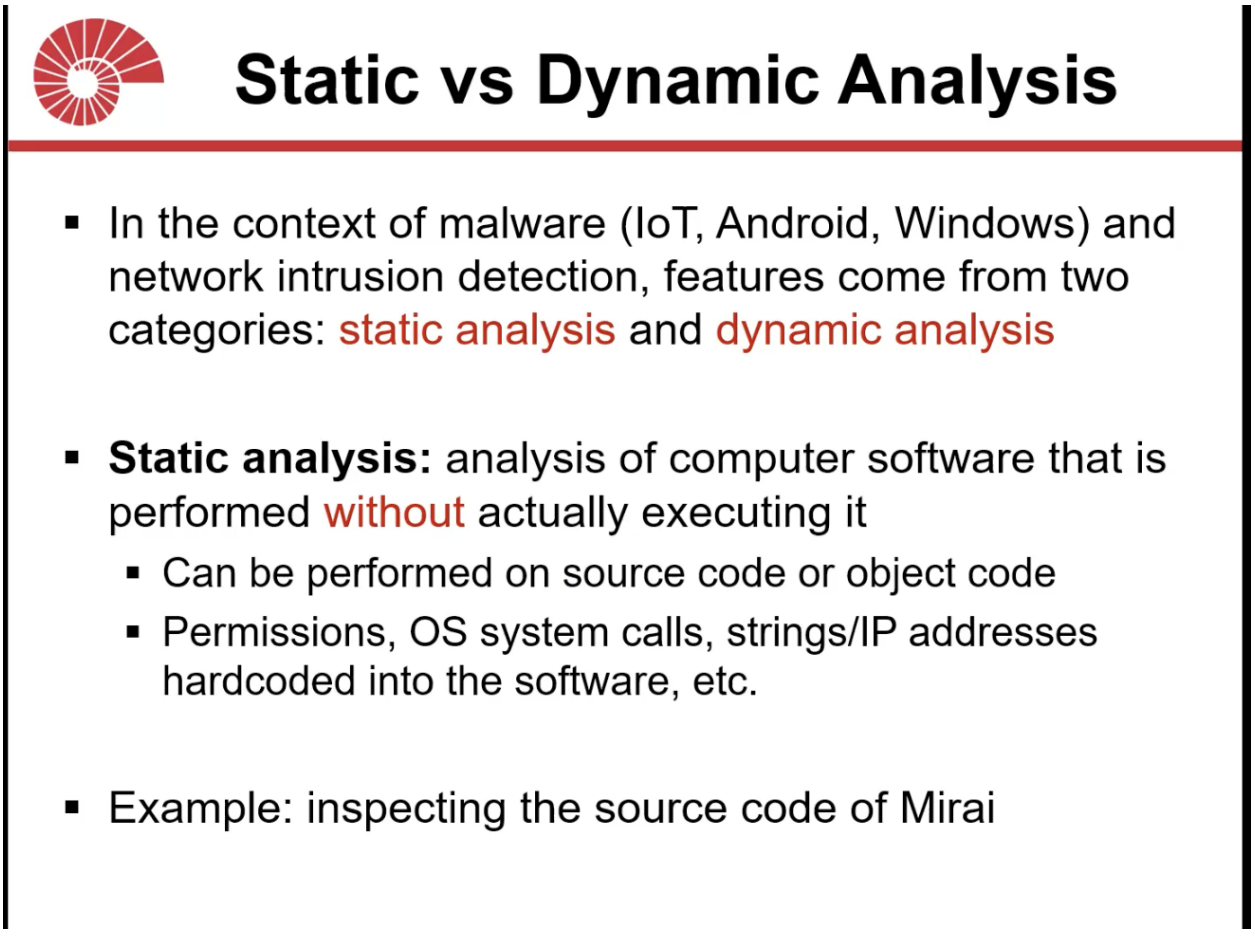
$$= \bar{C}_i$$

Lecture 16

OUE, OLH ve DP nin kalanlarını işliyor. Bunlar sınavda çıkmayacak.

Machine Learning for Cybersecurity

featurelar 2 türlü extraxt edilebilir



neye bastığında neyin açılacağı registry de depolanıyor

NIDS Case Study (KDD-Cup)

- Network traffic is labelled as belonging to one of 5 categories: **benign**, **DOS**, **R2L**, **U2R**, **probe**
  - A category can contain multiple attacks

Dos: Denial of Service

R2L: Remote to Local

U2R: User to root

Probe: Network probing

Bilevel Optimization

- We can formulate the problem of poisoning as a bilevel optimization problem
  - *L:* loss function
  - *(x_c, y_c):* features and label of added poison

$$\begin{aligned} &\arg\max_{x_c} \quad AdvGoal(D_{test}, \theta^*) \\ &\text{such that:} \quad \theta^* = \arg\min_{\theta} \mathcal{L}(D_{train} \cup \{x_c, y_c\}, \theta) \end{aligned}$$

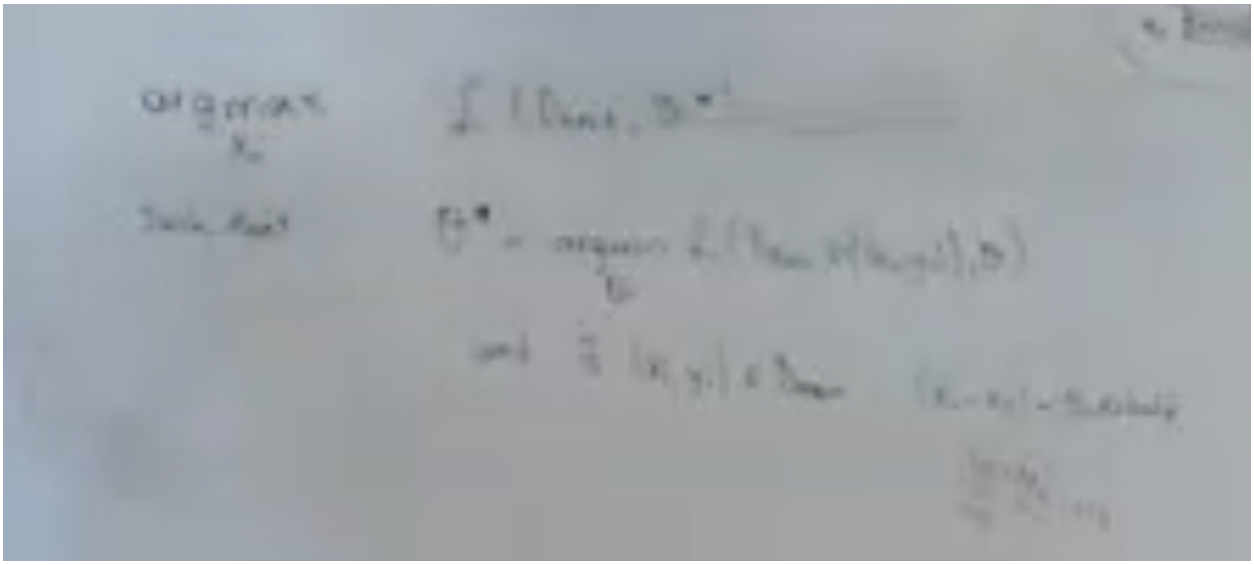
- What can be different *AdvGoal*? Targeted/untargeted?

Untarget ta loss oluyor Dtestte 0megae prime

targetta probabiltiy of xtarget in test dataset is misclassified

eger birden fazla poison varsa U Dpoison

bunu sorucak

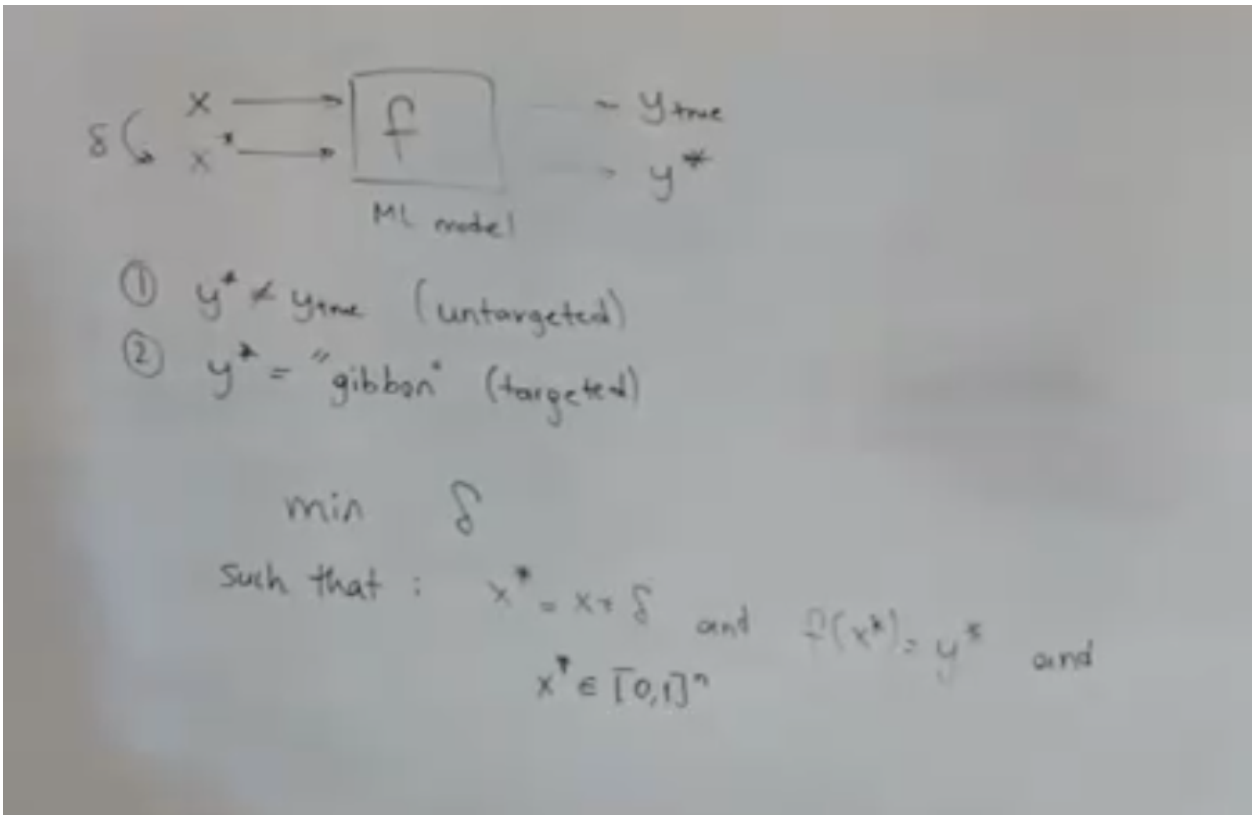


burda da defansı gösteriyor

xc,yc nin yanında bir xi,yi daha olmalı yakın olan ve aynı labela sahip olmalılar

adverserial single level çünkü sadece test var

poisoningte de training



Cross data transferiability

Cross model transferiabltiy