The Magic of Random Sampling: From Surveys to Big Data

Edith Cohen

Google Research

Tel Aviv University

Disclaimer: Random sampling is classic and well studied tool with enormous impact across disciplines. This presentation is biased and limited by its length, my research interests, experience, understanding, and being a Computer Scientist. I will attempt to present some big ideas and selected applications. I hope to increase your appreciation of this incredible tool.

Harvard IACS seminar 11/11/2016

What is a sample ?

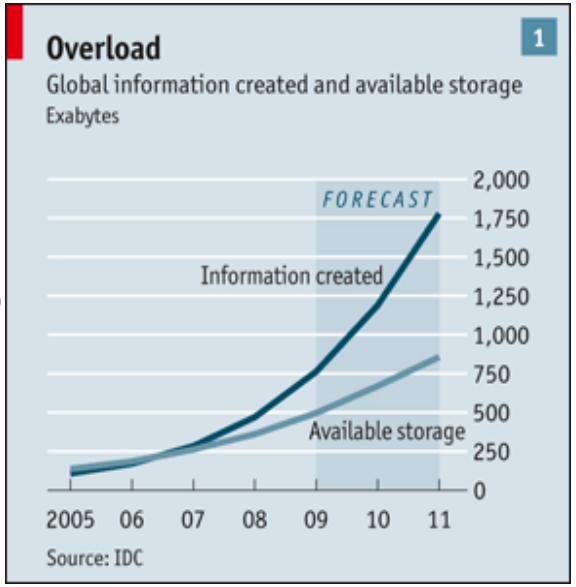
A sample is a summary of the data, in the form of a small set of representatives.

Why use samples?

A sample can be an adequate replacement of the full dataset for purposes such as summary statistics, fitting models, other computation/queries

When to use samples?

- Inherent Limited availability of the data
- Have data but limited resources:
  - Storage: long or even short term
  - Transmission bandwidth
  - Survey cost of sampled items
  - Computation
  - time delay of processing larger data



Data, data, everywhere. Economist 2010

art: pokemon.com

History

Basis of (human/animal) learning from observations

Recent centuries: Tool for surveying populations

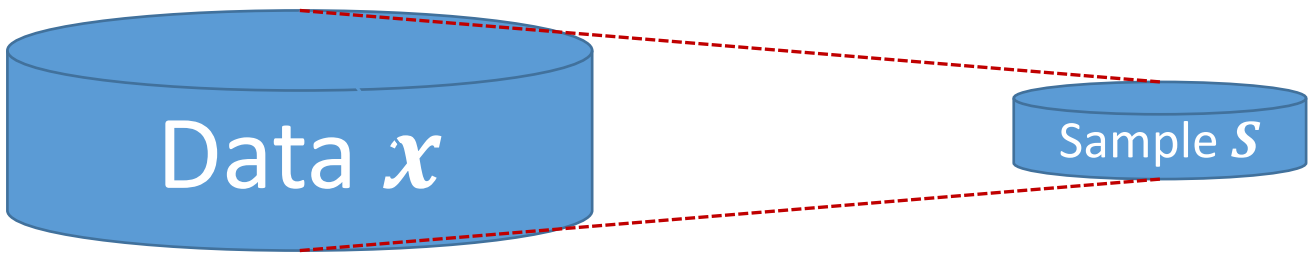
- Graunt 1662: Estimate population of England
- Laplace ratio estimator [1786, 1802]: Estimate the population of France. Sampled "Communes" (administrative districts), counting population ratio to live births in previous year. Extrapolate from birth registrations in whole country.
- Kiaer 1895 "representative method"; March 1903 "probability sampling"
- US census 1938: Use probabily sample to estimate unemployment

⋮

Recent decades: Ubiquitous powerful tool in data modeling, processing, analysis, algorithms design

Sampling schemes/algorithms design goals

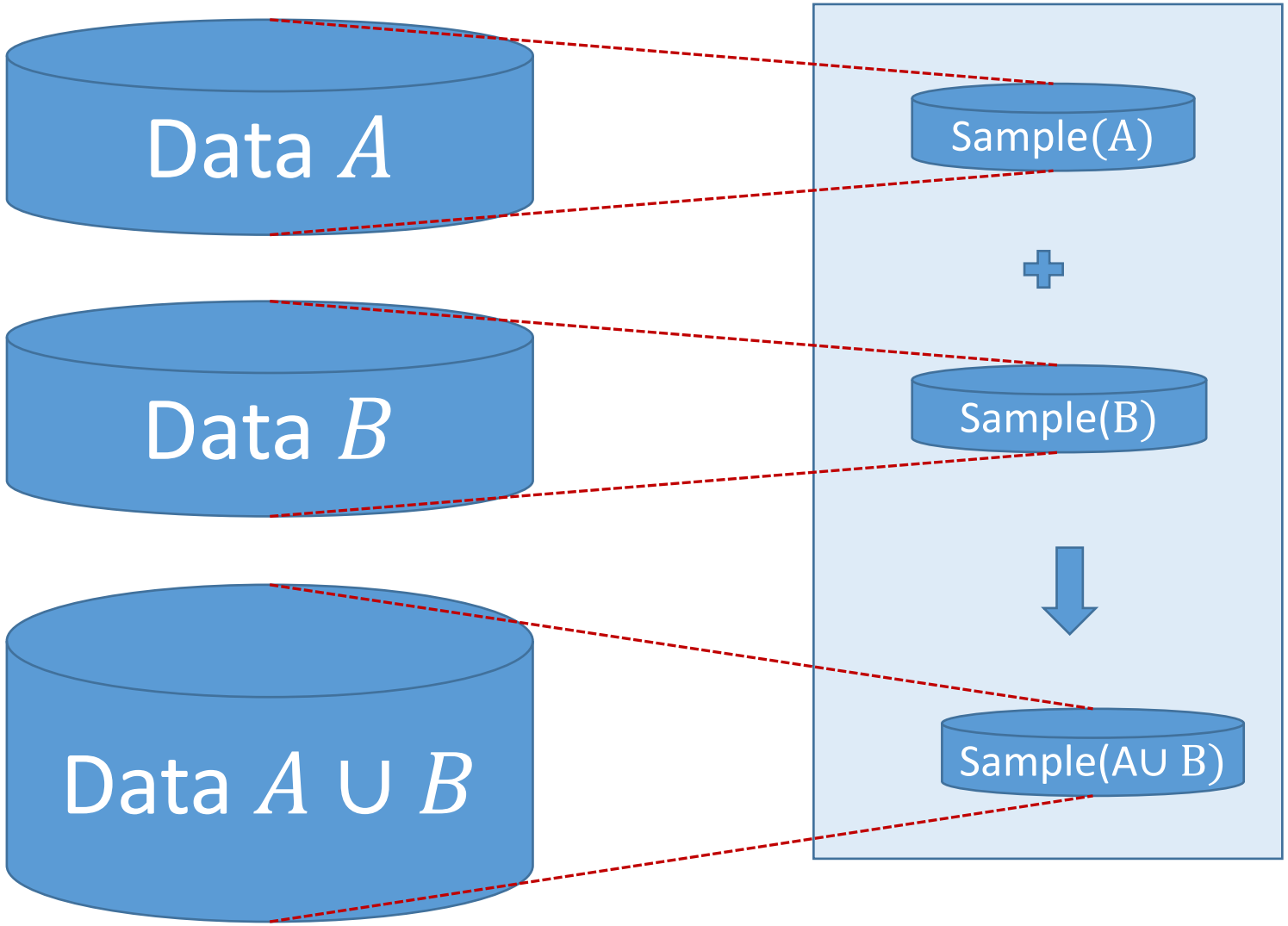
A sample is a lossy summary of the data from which we can approximate (estimate) properties of the full data



Q: $f(\boldsymbol{x})$ **?** $\longrightarrow$ estimator $\hat{f}(\boldsymbol{S})$

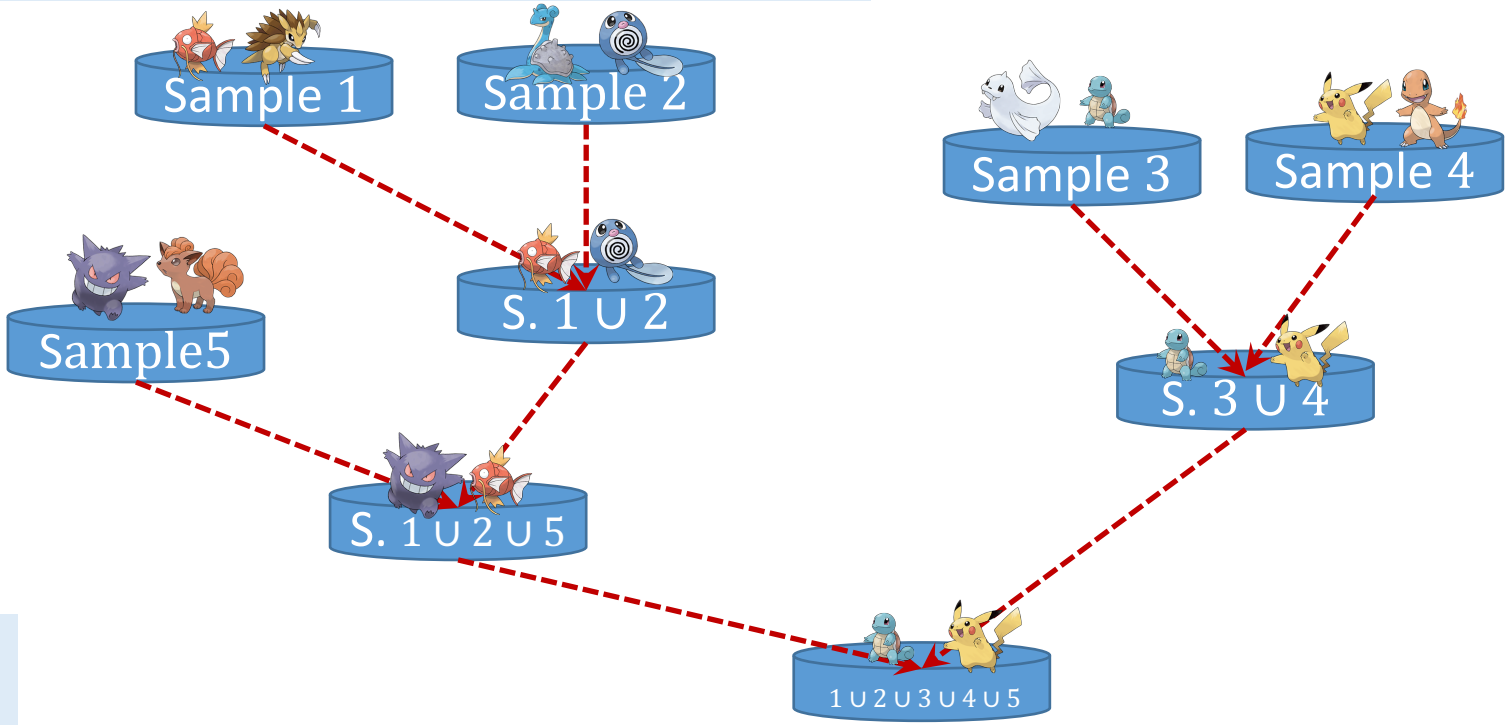
- Optimize sample-size vs. information content (quality of estimates)
  - Sometimes balance multiple query types/statistics over same sample
- Computational efficiency of algorithms and estimators
  - Efficient on modern platforms (streams, distributed, parallel)

Composable (Mergeable) Summaries

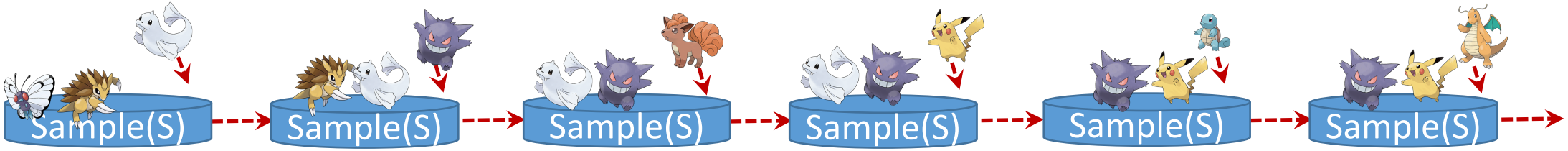


Why composable is useful ?

Distributed data/parallelize computation



Streamed data



Outline

Selected “big ideas” and example applications

- “Basic” sampling: Uniform/Weighted
- The magic of sample coordination
- Efficient computation over Unaggregated data sets (streamed, distributed)

Uniform (equal probability) sampling

Bernoulli sampling: Each item is selected Independently with probability p

Reservoir sampling [Knuth 1968, Vitter 1985] : selects a k-subset (uniformly at random)

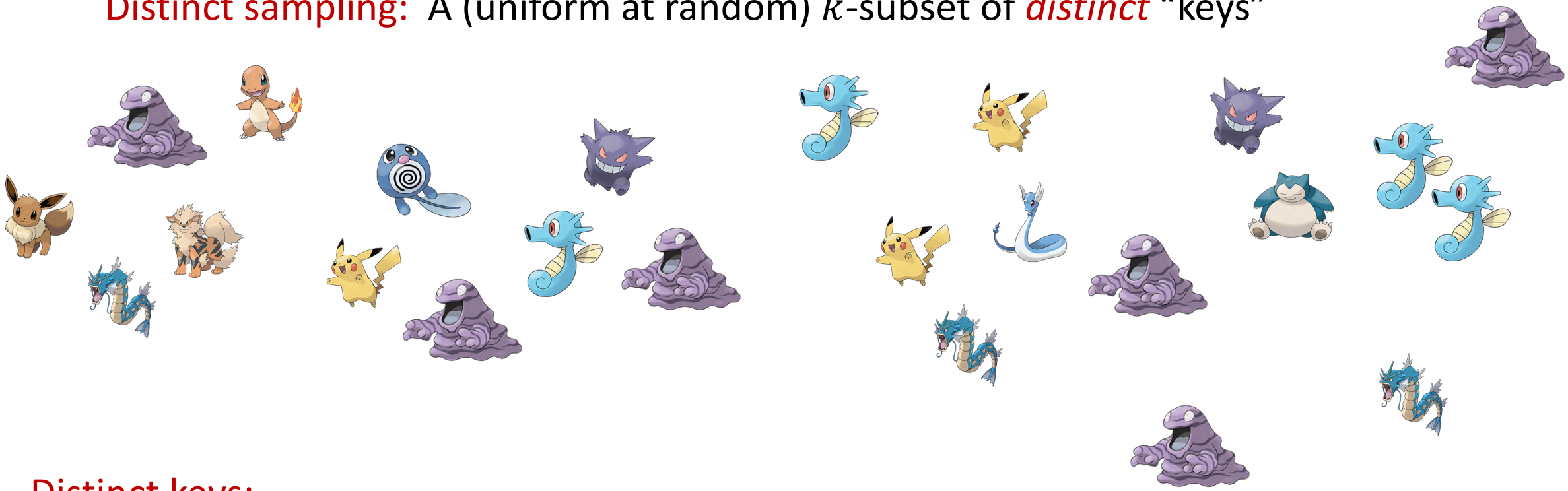
- Fixed sample size k ; Composable (Knuth's) ; state $\propto$ sample size k

- Associate with each item x an independent random number $H(x) \sim U[0,1]$
  - Can use a random hash function, limited precision
- Keep k items with smallest $H(x)$

- Correctness: All k subsets have same probability
- Composability: bottom-k($A \cup B$) $=$ bottom-k(bottom-k(A) $\cup$ bottom−k(B))

Uniform (equal probability) distinct sampling

Distinct sampling: A (uniform at random) k-subset of *distinct* "keys"



Distinct keys:

Uniform (equal probability) distinct sampling

Distinct sampling: A (uniform at random) k-subset of *distinct* "keys"
Distinct counting: #(distinct keys) in (sub)population

Example Applications:

- Distinct search queries
- Distinct source-destination IP flows
- Distinct users in activity logs
- Distinct web pages from logs
- Distinct words in a corpus
- ……

Distinct keys:

Uniform distinct sampling & approximate counting

Distinct sampling: A (uniform at random) k-subset of distinct "keys"
Distinct counting: #(distinct keys) in (sub)population

Computation

- (Naïve algorithm) Aggregate then sample/count: **state ∝ #distinct keys**
- Distinct Reservoir sampling/approximate counting [Knuth 1968]+ [Flajolet &Martin 1985]: composable summary with **state ∝ sample size k**

with key

- Associate with each item x an independent random hash $H(x) \sim U[0,1]$
- Keep k keys with smallest $H(x)$

Estimation from a uniform sample

Domain/Segment queries: Statistic of a selected segment H of population X

Application: Logs analysis for marketing, planning, targeted advertising

Examples: Approximate the number in Pokémons in our population that are

- College-educated water type
- Flying and hatched in Boston
- Weighing less than 1kg

Estimation from a uniform sample

Domain/Segment queries: Statistic of a selected *segment* H of population X

- **Ratio estimator** (when total population size $|\mathrm{X}|$ is known): $\frac{|S \cap H|}{|S|}|\mathrm{X}|$
- **Inverse probability estimator** [Horvitz Thompson 1952] (when $|\mathrm{X}|$ is not available, as with distinct reservoir sampling):

$$\frac{1}{p}|S \cap H| \quad \text{where} \quad p = \mathrm{Prob}\left[H(x) \leq \min_{y \notin S} H(y)\right] = (k+1)^{\mathrm{st}}\text{-smallest } H(x)$$

Properties:

- Unbiased minimum variance estimator
- coefficient of variation: $\frac{\sigma}{\mu} = \sqrt{\frac{|\mathrm{X}|}{|\mathrm{H}|k}}$ "relative error"
- Concentration (Bernstein, Chernoff) - probability of large relative error decreases exponentially

Approximate distinct counting of all data elements

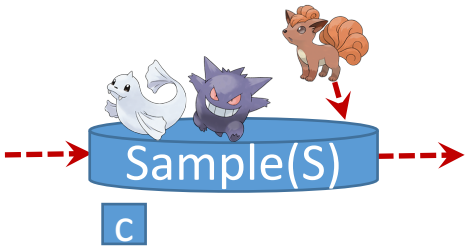
Can use distinct uniform sample, but can do better for this special case

HyperLogLog [Flajolet et al 2007]

- Really small structure: optimal size $O(\epsilon^{-2} + \log\log n)$ for CV $\frac{\sigma}{\mu} = \epsilon$; n distinct keys
- **Idea:** for counting, no need to store keys in sample, suffices to use $k = \epsilon^{-2}$exponents of hashes. Exponents value concentrated so store one and k offsets.

HIP estimators [Cohen '14, Ting '15]: halve the variance to $\frac{1}{2k}$!

- **Idea:** track an estimate count as the structure (sketch) is constructed. Add inverse probability of modifying structure with each modification
- Applicable with all distinct counting structures
- Surprisingly, better estimator than possible from final structure



Approx. count

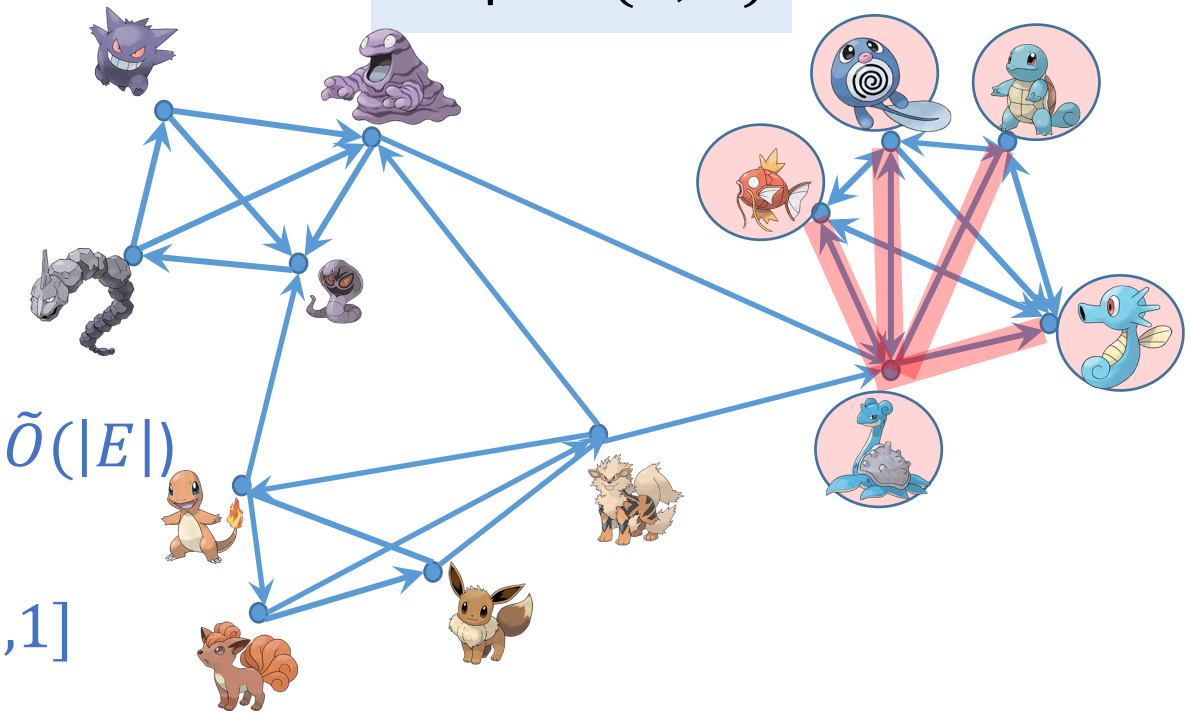
If structure S is modified

$c \mathrel{+}= 1/p(\text{modified})$

Graphs: Estimating cardinalities of neighborhoods, reachability, centralities

Q: number of nodes

- Reachable from 
- Within distance 5 from 



- Exact computation is $O(|E||V|)$
- Sample-based sketches [C' 1997]: near-linear $\tilde{O}(|E|)$

Idea:

- Associate independent random $H(v) \sim U[0,1]$ with all nodes
- Compute for each node u a sketch $S(u)$ of the ϵ^{-2} reachable nodes v with smallest $\mathrm{H}(v)$

Applications: Network analytics, social networks, other data with graph representation

Estimating Sparsity structure of matrix products

Problem: Compute a matrix product $B = A_1 A_2 \cdots A_n \quad n \geq 3$

- A_i are sparse (many more zero entries than nonzero entries)

Q: find best order (minimize computation) for performing multiplication

- From associativity, $B = A_1 A_2 A_3$ can be computed using $(A_1 A_2) A_3$ or $A_1 (A_2 A_3)$
- Computation depends on sparsity structure

[C' 1996]: The sparsity structure of sub-products (number of nonzeros in rows/columns of products) can be approximated in near-linear time (in number of nonzeros in $\{A_i\}$.

Solution: Preprocess to determine the best order, then perform computation

Idea: Define a graph for product by stacking bi-partite graphs G_i that correspond to nonzeros in A_i. Row/column sparsity of product corresponds to reachability set cardinality.

Unequal probability, Weighted sampling

Keys x can have weights w_x

Segment queries: For segment $H \subset \mathrm{X}$, estimate

$$\mathrm{w(H)} = \sum_{x \in H} w_x$$ (weight $\mathrm{w(H)}$ of my Pokémon bag H)

Example applications (logs analysis):

- Key: video-user view weight: watch time Query: Total watch time segmented by user and video attributes
- key: IP network flow weight: bytes transmitted Query: Total bytes transmitted for segment of flows (srcIP, destIP, protocol, port,…) Applications: billing, traffic engineering, planning

Unequal probability, Weighted sampling

- Key x has weights w_x
- Sample S that includes x with probability p_x

Segment query: $\mathrm{w(H)} = \sum_{x \in H} w_x$

Inverse probability estimator [HT 52]:

$$\widehat{w}(\mathrm{H}) = \sum_{x \in H \cap S} \frac{w_x}{p_x}$$

- Unbiased (when $p_x > 0$)
- ! To obtain statistical guarantees on quality we need to sample heavier keys with higher probability.

Poisson Probability Proportional to Size (PPS) [Hansen Hurwitz 1943]

$p_x \propto w_x$ minimizes sum of per-key variance

- CV $\frac{\sigma}{\mu} = \sqrt{\frac{w(\mathrm{X})}{w(H)k}}$ "relative error", concentration

Robust: If probabilities are approximate, guarantees degrade gracefully

Unequal probability, Weighted sampling

Composable weighted sampling scheme with fixed sample size k:
Bottom-k/Order samples/"weighted" reservoir

- Associate with each key x the value $r(x) = \frac{u_x}{w_x}$, for independent random $\mathrm{u_x} \sim D$
- Keep k keys with smallest $r(x)$

Example with $\mathrm{u_x} \sim U[0,1]$

Key x								
w_x	12.4kg	4.0kg	460.0kg	30.0kg	210.0kg	10.0kg	8.0kg	210.0kg
u_x	0.22	0.12	0.31	0.81	0.06	0.72	0.45	0.57
$r(x)$	0.017	0.03	0.00067	0.027	0.00029	0.072	0.05625	0.00271

Unequal probability, Weighted sampling

Composable weighted sampling scheme with fixed sample size k:
Bottom-k/Order samples/"weighted" reservoir

- Associate with each key x the value $r(x) = \frac{u_x}{w_x}$, for independent random $\mathrm{u_x} \sim D$
- Keep k keys with smallest $r(x)$

Estimation: Inverse probability

$\tau =$ smallest r_x of $x \notin S$; $p_x = \mathrm{Prob}[r(x) < \tau] = \mathrm{Prob}_{y \sim D}[y < w_x \tau]$

- Without-replacement $D = \mathrm{EXP}[1]$ [Rosen 72, C' '97 C' Kaplan '07, Eframidis Spirakis '05]
- Priority (sequential Poisson) sampling $D = \mathrm{U}[0,1]$ [Ohlsson '00, Duffield Lund Thorup '07]
  - Essentially same quality guarantees as Poisson PPS

!! works for max-distinct over key value pairs $(e.key, e.w)$ where $w_x = \max_{e|e.key=x} \mathrm{e.w}$

Unequal probability, Weighted sampling

Example Applications:

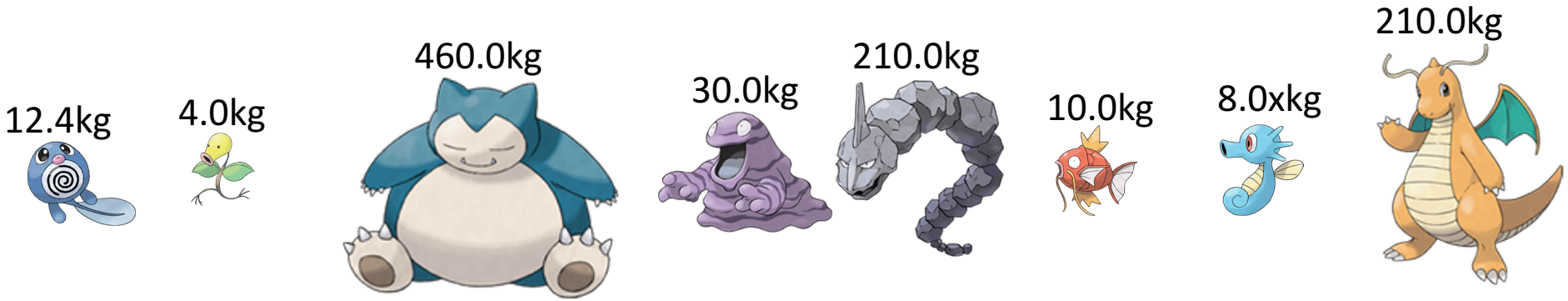
Sampling nonnegative matrix products [Cohen Lewis 1997]
Nonnegative matrices A, B
Efficiently sample entries in the product AB proportionally to their magnitude (without computing the product)
Idea: random walks using edge weight probabilities

Cut/spectral graph sparsifiers of $G = (V, E)$
[Benczur Karger 1996] Sample graph edges and re-weight by inverse probability to obtain a $G' = (V, E')$ such that $|E'| = O(|V|)$ and all cut values are approximately preserved
[Spielman Srivastava 2008] Spectral approximation $L_G \approx L_{G'}$ by using $p_e \propto$(approximate) effective resistance of $e \in E$

Sample Coordination

[Brewer, Early, Joyce 1972] Same set of keys, multiple sets of weights ("instances"). Make samples of different instance as similar as possible.

Monday:



Tuesday:

Survey sampling motivation: Weight evolve but surveys impose burden. Want to minimize the burden and still have a weighted sample of the evolved set.

How to coordinate samples

Coordinated bottom-k samples: Use same $u_x = H(x)$ for all instances

- Associate with each key x the value $r(x) = \frac{u_x}{w_x}$, for independent random hash $u_x \sim D$
- Keep k keys with smallest $r(x)$

Key x								
Mon w_x	12.4kg	4.0kg	460.0kg	30.0kg	210.0kg	10.0kg	8.0kg	210.0kg
Tue w_x	15.0kg	6.0kg	300.0kg	50.0kg	110.0kg	5.0kg	4.0kg	300.0kg
u_x	0.22	0.12	0.31	0.81	0.06	0.72	0.45	0.57
Mon $r(x)$	0.017	0.03	0.00067	0.027	0.00029	0.072	0.05625	0.00271
Tue $r(x)$	0.015	0.02	0.00103	0.0162	0.00055	0.144	0.1125	0.0019

!! Change is minimized given that we have a weighted sample of each instance

Coordination of samples

Very powerful tool for big data analysis with applications well beyond what [Brewer, Early, Joyce 1972] could envision

- **Locality Sensitive Hashing** (LSH) (similar weight vectors have similar samples/sketches)
- **Multi-objective samples** (universal samples): A single sample (as small as possible) that provides statistical guarantees for multiple sets of weights/functions.
- **Statistics/Domain queries that span multiple "instances"** (Jaccard similarity, L_p distances, distinct counts, union size, sketches of coverage functions…)
  - MinHash sketches are a special case with 0/1 weights.
- **Facilitates faster computation of samples**. Example: [C' 97] Sketching/sampling reachability sets and neighborhoods of all nodes in a graph in near-linear time.
- **Facilitates efficient optimization over samples**: Optimize objective over sets of weights/functions/parametrized functions. Example: Centrality/clustering objective [CCK '15], learning [Kingman Welling '14]

Multi-objective Sample

- Same keys can have different “weights:” IP flows have bytes, packets, count
- We want to answer segment queries with respect to all weights.
- **Naïve solution**: 3 disjoint samples
- **Smart solution**: A single multi-objective sample

Multi-objective sample of f-statistics

One set of weights w_x, but are interested in *different functions* $f(w_x)$

	x									
	w_x	135	2	9	18	21	4	11	4	2
f	Count	1	1	1	1	1	1	1	1	1
	$\text{cap}_5(w_x)$	5	2	5	5	5	4	5	4	2
	thresh_{10}	1	0	0	1	1	0	1	0	0

Multi-objective Sample [C' Kaplan Sen '09, C' '15]

Set of functions $\mathrm{w} \in W$ where $\mathrm{w}: \mathrm{X} \to R_+$

- Compute coordinated samples $S^{(i)}$ for each $\mathrm{w}^{(\mathrm{i})} \in W$

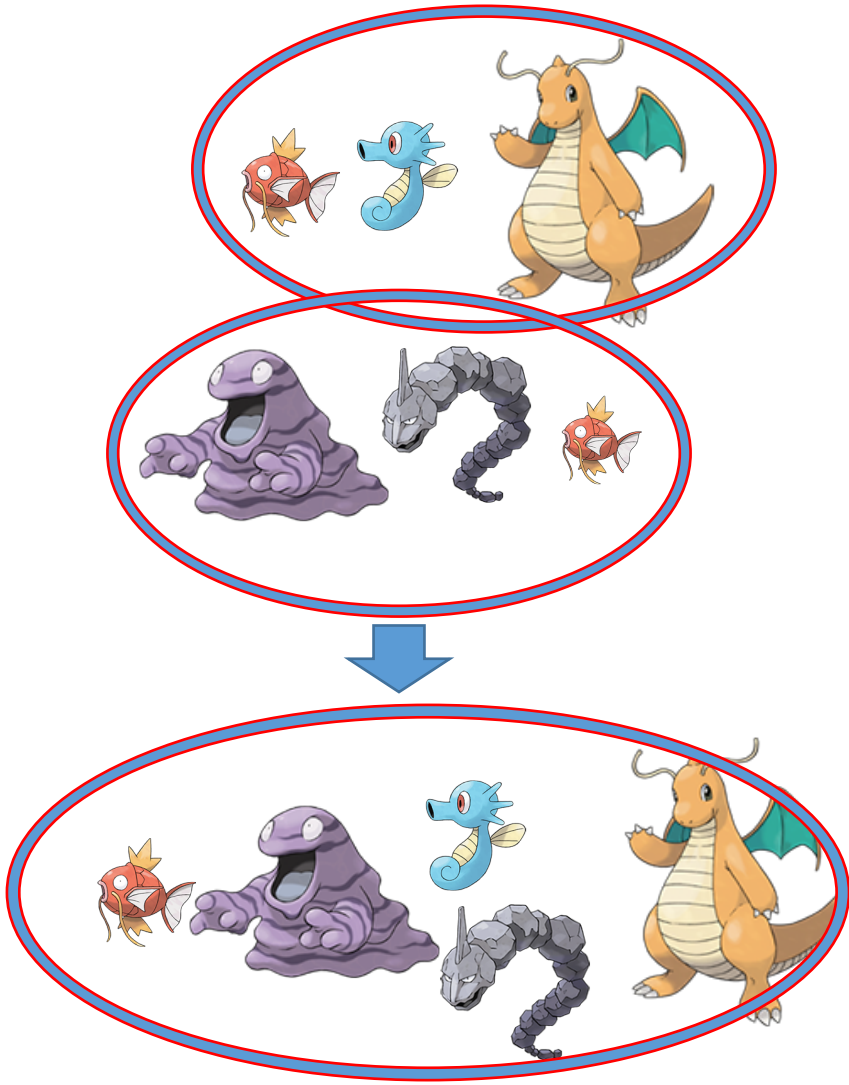
Our multi-objective sample S for W is

- The union $\mathrm{S} = \bigcup_i S^{(i)}$
- Sampling probabilities $p_x = \max_i p^{(i)}(x)$

Theorem: For any domain query: $\mathrm{w}^{(\mathrm{i})}(\mathrm{H}) = \sum_{x \in H} w_x^{(\mathrm{i})}$

The inverse probability estimator $\widehat{w}^{(i)}(\mathrm{H}) = \sum_{x \in H \cap S} \frac{w_x^{(i)}}{p_x}$

Is unbiased and provides statistical guarantees that are at least as strong as an estimator applied to the dedicated sample $S^{(i)}$



Multi-objective sample of statistics

One set of weights w_x, different functions $f(w_x)$

x									
w_x	135	2	9	18	21	4	11	4	2
f Count	1	1	1	1	1	1	1	1	1
f $\text{cap}_5(w_x)$	5	2	5	5	5	4	5	4	2
f thresh_{10}	1	0	0	1	1	0	1	0	0
u_x	0.52	0.24	0.76	0.90	0.14	0.32	0.44	0.07	0.82
$\frac{u_x}{\text{thresh}_{10}(w_x)}$	0.52	∞	∞	0.90	0.14	∞	0.44	∞	∞
$\frac{u_x}{\text{cap}_5(w_x)}$	0.104	0.120	0.152	0.18	0.064	0.080	0.088	0.0175	0.41

Priority (sequential Poisson)

For $k = 3$, the MO sample for $F = \{\text{count}, \text{thresh}_{10}, \text{cap}_5\}$ is:

Multi-objective sample of all monotone statistics

M : All monotone non-decreasing functions f with $f(0) = 0$

Examples: $f(w) = w^p$; $f(w) = \min\{10, w\}$; $f(w) = \log(1 + w)$, ...

Data of key value pairs (x, w_x) : For each f, Instance is $f(w_x)$ for $x \in \mathrm{X}$

Theorem: [C' 97, C' K '07] (threshold functions) [C '15] all M

Multi-objective sample for all monotone statistics M has

- (expected) sample size: $O(k \ln n)$, where $n =$ #keys with $w_x > 0$
- Composable structure of size equal to the sample size

$\Longrightarrow$ Very efficient to compute on streamed/parallel/distributed platforms

Next: Applications to graphs and streams

Multi-objective sample of monotone statistics

Theorem: [C' 97, C' K '07] (threshold functions) [C '15] all M

Multi-objective sample for all monotone statistics M has

- (expected) sample size: $O(k \ln n)$, where $n =$ #keys with $w_x > 0$
- Composable structure of size equal to the sample size

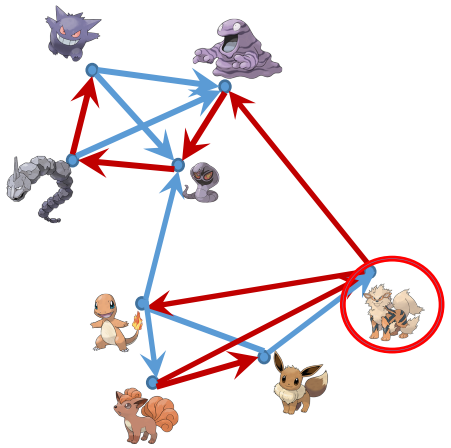
Application: Data Streams time-decaying aggregations

monotone non-increasing $\alpha(x)$, and segment $H \subset V$

$$A_\alpha = \sum_{u \in H} \alpha\,(t_u)$$

- t_u: Elapsed time from start of stream to u
- t_u: Elapsed time from u to current time

Multi-objective sample of monotone statistics



Theorem: [C' 97, C' K '07] (threshold functions) [C '15] all M

Multi-objective sample for all monotone statistics M has

- (expected) sample size: $O(k \ln n)$, where $n =$ #keys with $w_x > 0$
- Composable structure of size equal to the sample size

Application: Centrality of all nodes in a graph $G = (V, E)$ [C' 97 C' Kaplan '04 C '15]

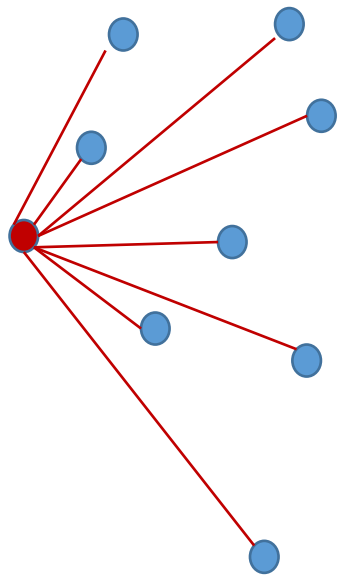
For a node v, monotone non-increasing $\alpha(x)$, and segment $H \subset V$

centrality of v for segment H is

$$\mathrm{C}_\alpha(v, H) = \sum_{u \in H} \alpha\left(d_{vu}\right)$$ (Harmonic centrality: $\alpha(x) = \frac{1}{x}$)

Thm: All-Distances Sketches (ADS) (MO samples) for **all** nodes can be computed in $\tilde{O}(|E|)$ computation. We can estimate $\mathrm{C}_\alpha(v, H)$ for all α, H from ADS(v)

Multi-objective sample of distances to a set of points in a metric space [Chechik C' Kaplan '15]



- Metric space M
- Set of points $P = \{x_1, x_2, \dots, x_n\} \subset M$
- Each point $v \in M$ defines weights $w_v(x_i) = d_{v\,x_i}$
- A multi objective sample of all w_v allows us to estimate for segments $H \subset P$, and any query point v the sum of distances

$$\mathrm{C}(\mathrm{v}, \mathrm{H}) = \sum_{x_i \in H} d_{v\,x_i}$$

Theorem:
- Multi-objective overhead for distances is $O(1)$!
- Can be computed using a near-linear number of distance queries

$\Longrightarrow$ Sample size $O(\epsilon^{-2})$ suffices for estimates of $\mathrm{C}(\mathrm{v}, P)$ for each $v \in M$ with CV ϵ

!!! Can even relax triangle inequality to $\mathrm{d}_{\mathrm{ac}} \leq \rho(d_{ab} + d_{bc})$ (e.g. squared distances)

Estimators for multi-instance aggregates

Set of functions $\mathrm{w} \in W$ where $\mathrm{w}: \mathrm{X} \to R_+$

Coordinated samples $S^{(i)}$ for each $\mathrm{w}^{(\mathrm{i})} \in W$

Example multi-instance aggregations:

- L_p^p distance $\sum_{x \in H} |w^{(1)}(x) - w^{(2)}(x)|^p$
- One-sided L_p^p $\sum_{x \in H} \max\{0, w^{(1)}(x) - w^{(2)}(x)^p$

  Graphs: Influence estimation from node sketches

- max, min, k^{th},
- Union, Jaccard similarity
- Generalized coverage functions

Specific estimators for specific aggregations:

0/1 weights: Size of union [C' 95] Jaccard similarity [Broder '97]

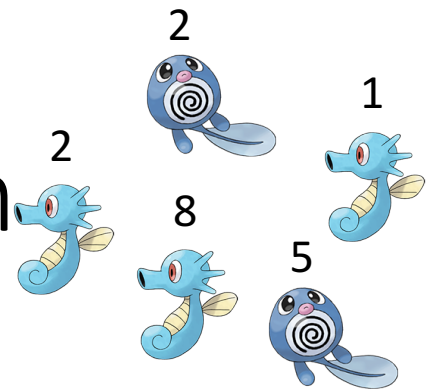
General weights, tighter estimators: max, min, quantiles [C' Kaplan 2009,2011]

Monotone Estimation Problems [C' Kaplan 13, C' 14]:

- Characterization of all functions for which unbiased bounded variance estimators exist
- Efficient (Pareto optimal) estimators (when they exist)

!!! Coordination is essential in getting good estimators. Independent samples will not work.

Distributed/Streamed data elements: Sampling/counting without aggregation



- Data element e has key and value *(e.key,e.value)*
- Multiple elements may have the same key

- Max weight: $w_x = \max_{e|e.key=x} \text{e.value}$

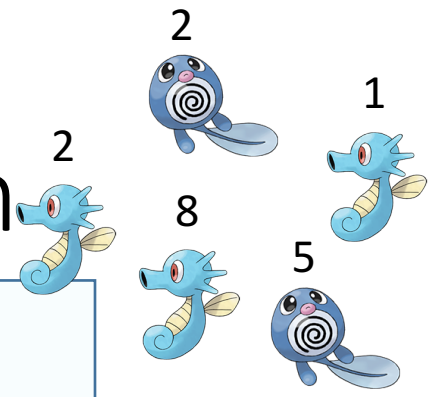
V Composable bottom-k

- Sum of weights: $w_x = \sum_{e|e.key=x} \text{e.value}$

- Segment f-statistics: $\sum_{x \in H} f(w_x)$ *sampling scheme*
- f-statistics: $\sum_{x \in X} f(w_x)$ *Approximate "counting" structure*

- Naïve: Aggregate pairs (x, w_x), then sample - requires state linear in #distinct keys
- Challenge: Sample/Count with respect to $f(w_x)$ using small state (no aggregation).
  - Sampling gold standard: "aggregated" sample size/quality tradeoffs CV= $\frac{\sigma}{\mu} = \frac{1}{\sqrt{k}}$
  - Counting gold standard: like HyperLogLog $O(\epsilon^{-2} + \log\log n)$, CV= ϵ

Distributed/Streamed data elements: Sampling/counting without aggregation

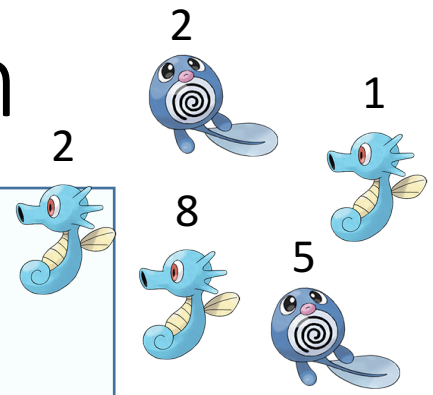


- Data element e has key and value *(e.key,e.value)*
- Multiple elements may have the same key
  - Sum of weights: $w_x = \sum_{e|e.key=x} \mathrm{e.value}$
  - Segment f-statistics: $\sum_{x \in H} f(w_x)$
  - f-statistics: $\sum_{x \in \mathrm{X}} f(w_x)$

- Distinct $f(\mathrm{x}) = 1$ $(x > 0)$: count [Flajolet Martin '85, Flajolet et al '07] sample [Knuth '69]
- Sum $f(\mathrm{x}) = \mathrm{x}$: count [Morris '77], sample [Gibbons Matias '98, Estan Varghese '05, CDKLT '07]
- Frequency moments $f(\mathrm{x}) = \mathrm{x}^{\mathrm{p}}$: count [Alon Matias Szegedy '99, Indyk '01]
- "universal" sketches [Braverman Ostrovsky '10] (count)

But -- Except for sum and distinct, not even close to "gold standard"

Sampling/counting without aggregation



- Data element e has key and value *(e.key,e.value)*
- Multiple elements may have the same key
  - Sum of weights: $w_x = \sum_{e|e.key=x} \mathrm{e.value}$
  - Segment f-statistics: $\sum_{x \in H} f(w_x)$
  - f-statistics: $\sum_{x \in \mathrm{X}} f(w_x)$

[C' 15, C' 16] Sampling and counting near "gold standard" ($\times\sqrt{e/(e-1)} \approx 1.26$)

f: Concave with (sub) linear growth

- Distinct, sum
- Low freq. moments: $f(x) = x^p$ for $p \in [0,1]$
- Capping functions: $Cap_T(x) = \min\{T, x\}$
- Logarithms: $f(x) = \log(1 + x)$

Sampling Ideas: Element process that convert sum to max via distributions to approximate sampling probabilities. Invert the sampling transform for unbiased estimation.

Counting Ideas: Element processing guided by Laplace transform to convert to max-distinct approximate counting problem.

Conclusion

We got a taste of sampling “big ideas” that have tremendous impact on analyzing massive data sets.

- Uniform sampling
- Weighted sampling
- Coordination of samples
  - Multi-objective sample
  - Estimation of multi-instance functions
- Sampling and computing statistics unaggregated (distributed/streamed) elements

Future: Still fascinated by sampling and their applications

Near future directions:

- Extend “gold standard” sampling/counting over unaggregated data and understand limits of approach
- Coordination for better mini-batch selection for metric embedding via SGD
- Multi-objective samples for clustering objectives, understand optimization over coordinated samples



Thank you !