Part 2: Definitions and Algorithms with Provable Guarantees

Part 2: Definitions and Algorithms with **Provable Guarantees**

May be essential for your application of interest

Part 2: Definitions and Algorithms

- Exact Unlearning
  - Retrain from Scratch
  - Unlearning with structure
  - SISA
- Approximate Unlearning
  - Differential Privacy
  - Influence Functions

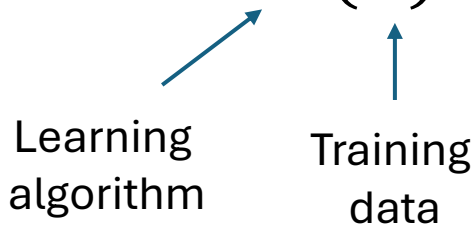
Part 2: Definitions and Algorithms

- **Exact Unlearning**
  - Retrain from Scratch
  - Unlearning with structure
  - SISA
- Approximate Unlearning
  - Differential Privacy
  - Influence Functions

Exact Unlearning

A model has already been trained on a dataset X

$$M(X)$$



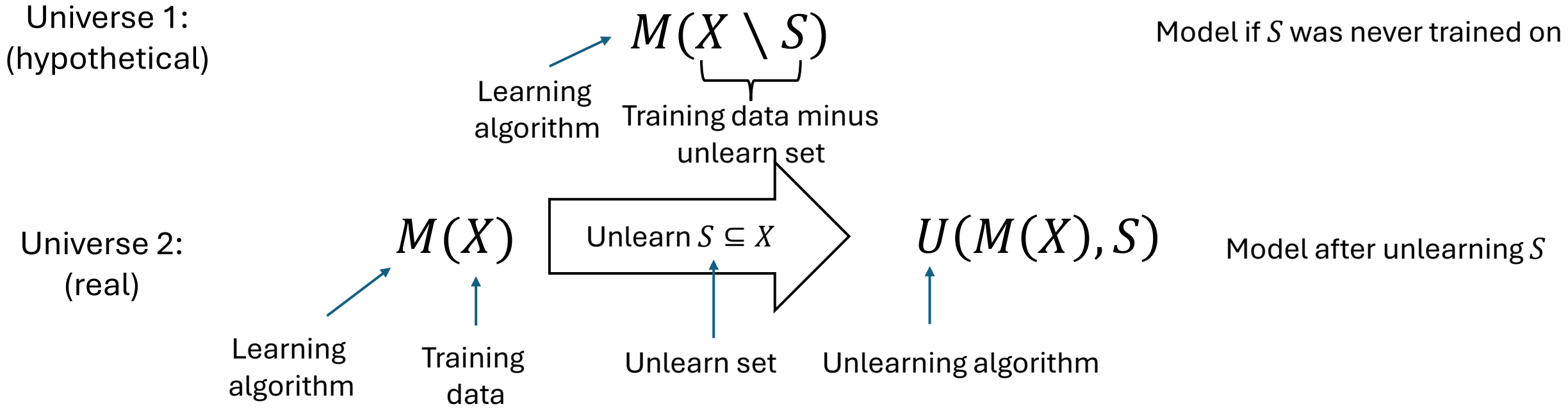
We wish it weren't trained on some "unlearn set" $S \subseteq X$

$$M(\underbrace{X \setminus S}_{\text{Training data minus unlearning set}})$$

Is there an "unlearning algorithm" U that transforms $M(X)$ to $M(X \setminus S)$?

Exact Unlearning

"After unlearning some points, the model should be the exact same as if those points were never trained on"



$$U(M(X), S) = M(X \setminus S)$$

Exact unlearning:
The two models should be identical

A few more nuances

- Unlearning algorithms often need more than the model $M(X)$ and the unlearning set S to perform unlearning
  - Include some auxiliary information Aux: $U(M(X), S, \text{Aux})$
- Learning algorithms are often randomized
  - "Exactly equal" models doesn't make sense if they're randomized
  - Want equality in distribution: $M(X \setminus S) \stackrel{d}{=} U(M(X), S, \text{Aux})$

Part 2: Definitions and Algorithms

- Exact Unlearning
  - **Retrain from Scratch**
  - Unlearning with structure
  - SISA
- Approximate Unlearning
  - Differential Privacy
  - Influence Functions

Naïve Baseline: Retrain from Scratch

- Unlearning alg. ignores the model it has, trains again on $X \setminus S$
- $U(M(X), S, \mathrm{Aux}) = M(X \setminus S)$
  - Choose $\mathrm{Aux} = X$
- Pros: very simple, obviously satisfies exact unlearning
- Cons: very slow, requires a full training procedure even to unlearn one data point
  - Prohibitively expensive – could cost hundreds of millions of dollars
- Motivates the biggest goal in machine unlearning: how do we do unlearning *efficiently* and *effectively*?

Summary: Retrain from Scratch

Consideration	Description

Summary: Retrain from Scratch

Consideration	Description
Unlearning time	Very poor, retrain from scratch is slow

Summary: Retrain from Scratch

Consideration	Description
Unlearning time	Very poor, retrain from scratch is slow
Supported models	All models supported

Summary: Retrain from Scratch

Consideration	Description
Unlearning time	Very poor, retrain from scratch is slow
Supported models	All models supported
Effect of large unlearning sets	Unlearning sets of all sizes are handled at no additional cost

Summary: Retrain from Scratch

Consideration	Description
Unlearning time	Very poor, retrain from scratch is slow
Supported models	All models supported
Effect of large unlearning sets	Unlearning sets of all sizes are handled at no additional cost
Utility	Perfect, same as base model

Summary: Retrain from Scratch

Consideration	Description
Unlearning time	Very poor, retrain from scratch is slow
Supported models	All models supported
Effect of large unlearning sets	Unlearning sets of all sizes are handled at no additional cost
Utility	Perfect, same as base model
Training overhead	Training procedure is unchanged

Summary: Retrain from Scratch

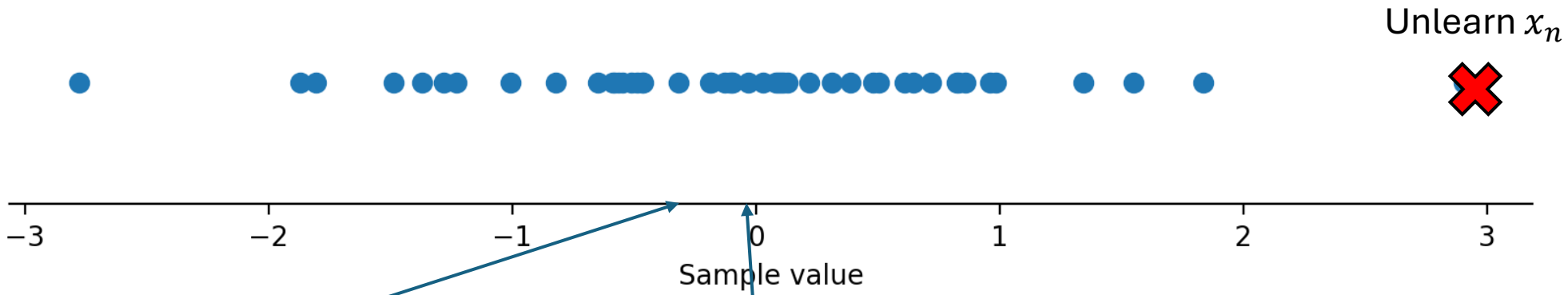
Consideration	Description
Unlearning time	Very poor, retrain from scratch is slow
Supported models	All models supported
Effect of large unlearning sets	Unlearning sets of all sizes are handled at no additional cost
Utility	Perfect, same as base model
Training overhead	Training procedure is unchanged
Auxiliary information	Requires storing entire dataset X

Part 2: Definitions and Algorithms

- Exact Unlearning
  - Retrain from Scratch
  - **Unlearning with structure**
  - SISA
- Approximate Unlearning
  - Differential Privacy
  - Influence Functions

When can structure help? A toy example

Compute mean of the dataset



$$\mu' = \frac{1}{n-1}\sum_{i=1}^{n-1} x_i$$

OR

$$\mu' = \frac{1}{n-1} \cdot (n\mu - x_n)$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Compute mean: $O(n)$

Unlearn point naively: $O(n)$

Unlearn point carefully: $O(1)$

Figure: ChatGPT 5.5 Pro

Unlearning from Structure (Naïve Bayes)

- A bit stylized, but illustrates a more general principle
  - Structured models can permit fast unlearning
  - [Cao and Yang, Oakland 2015]
- Dataset $\{(x_i, y_i)\}_{i=1}^{n}$, $x_i \in \{0,1\}^d$, $y_i \in \{0,1\}$
- Naïve Bayes classifier: for a test example x, predict

$$\hat{y} = \arg\max_y \Pr[Y = y] \prod_{j=1}^{d} \Pr\left[X_j = x_j \mid Y = y\right]$$

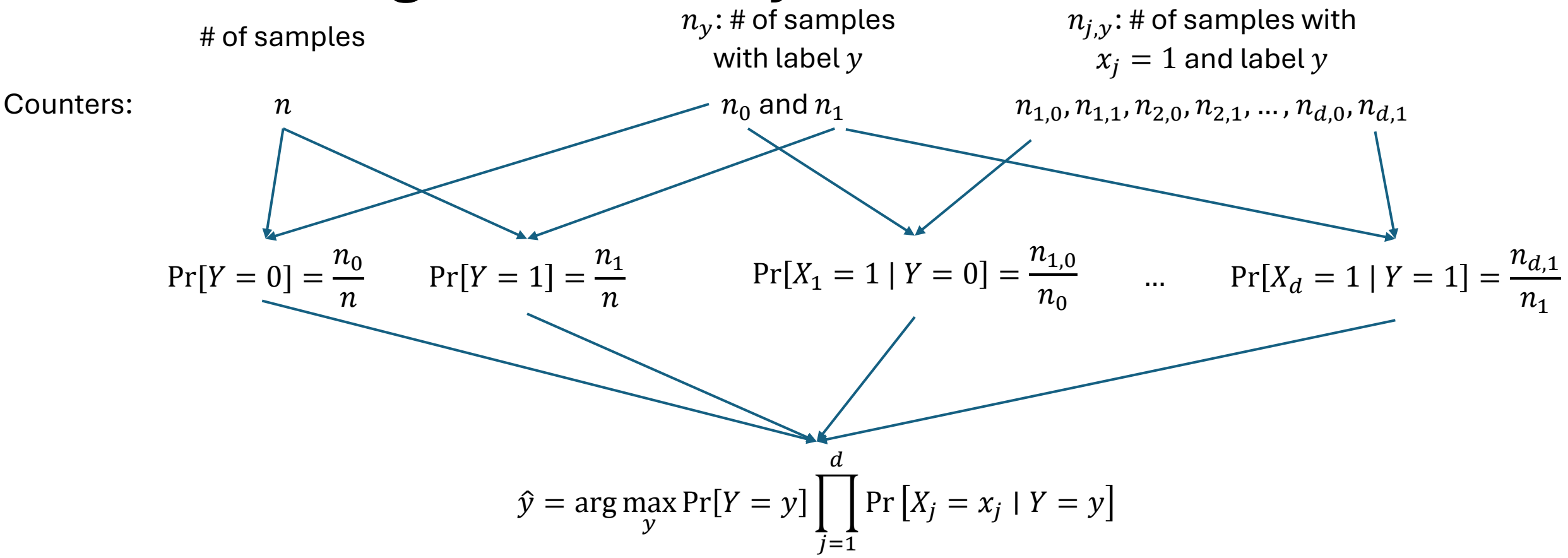
  - MLE assuming features are conditionally independent
- Quantities $\Pr[Y = y]$, $\Pr[X_j = x_j | Y = y]$ estimated from the data

Counters for Naïve Bayes

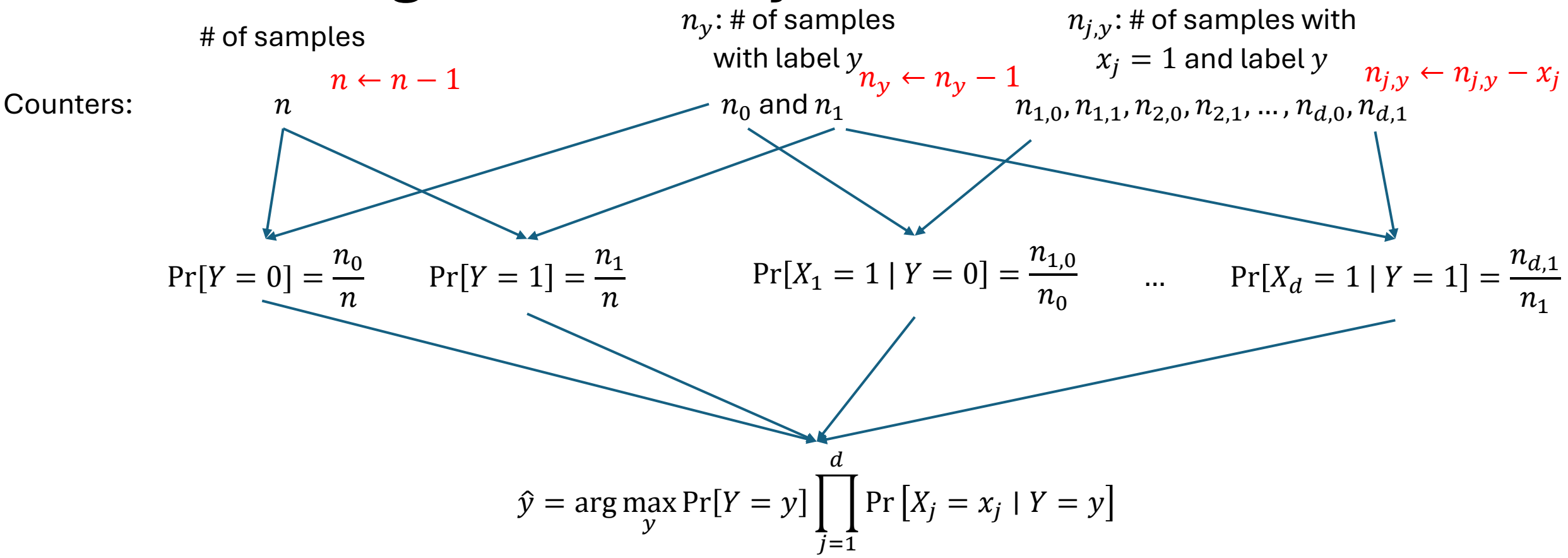
$$\hat{y} = \arg\max_{y} \Pr[Y = y] \prod_{j=1}^{d} \Pr\left[X_j = x_j \mid Y = y\right]$$

- Easy to compute these empirical quantities
  - E.g., $\Pr[Y = y] = \frac{n_y}{n}$, is the fraction of the training dataset with label y
    - n_y: # of training data points w/ label y, n is total # of training data points
  - $\Pr[X_j = x_j | Y = y]$ is similar, fraction of y-labeled samples with value x_j
- Prediction is based only on several counters: can be updated fast!

Unlearning in Naïve Bayes



Unlearning in Naïve Bayes



How to unlearn a point (x, y)?
Update affected counters, recompute probabilities
$O(d)$ time to unlearn a point! Versus $O(nd)$ time for retrain from scratch
...but it's for naïve Bayes... (which might suffice for your use case!)

Summary: Cao-Yang Unlearning

Consideration	Description

Summary: Cao-Yang Unlearning

Consideration	Description
Unlearning time	Very efficient, linear in size of unlearning set

Summary: Cao-Yang Unlearning

Consideration	Description
Unlearning time	Very efficient, linear in size of unlearning set
Supported models	Highly limited, only special model classes

Summary: Cao-Yang Unlearning

Consideration	Description
Unlearning time	Very efficient, linear in size of unlearning set
Supported models	Highly limited, only special model classes
Effect of large unlearning sets	Minimal, time scales linearly in size

Summary: Cao-Yang Unlearning

Consideration	Description
Unlearning time	Very efficient, linear in size of unlearning set
Supported models	Highly limited, only special model classes
Effect of large unlearning sets	Minimal, time scales linearly in size
Utility	Exact unlearning, so no utility loss

Summary: Cao-Yang Unlearning

Consideration	Description
Unlearning time	Very efficient, linear in size of unlearning set
Supported models	Highly limited, only special model classes
Effect of large unlearning sets	Minimal, time scales linearly in size
Utility	Exact unlearning, so no utility loss
Training overhead	Minimal, only need to compute counters

Summary: Cao-Yang Unlearning

Consideration	Description
Unlearning time	Very efficient, linear in size of unlearning set
Supported models	Highly limited, only special model classes
Effect of large unlearning sets	Minimal, time scales linearly in size
Utility	Exact unlearning, so no utility loss
Training overhead	Minimal, only need to compute counters
Auxiliary information	Minimal, only need to store counters

Part 2: Definitions and Algorithms

- Exact Unlearning
  - Retrain from Scratch
  - Unlearning with structure
  - **SISA**
- Approximate Unlearning
  - Differential Privacy
  - Influence Functions

SISA [Bourtoule et al., Oakland 2021]

- Previous unlearning solution: highly model-dependent
- SISA: completely model agnostic!
- SISA = Sharded, Isolated, Sliced, and Aggregated
- Uses ideas from ensemble learning and distributed systems to enable efficient unlearning

(Baby) SISA

- Basic idea: split dataset into t "shards" of size n/t
  - Recommend $t \leq 20$
- Train t models independently, one on each shard
- Aggregate results of t models
  - E.g., majority vote for prediction
- Key observation: unlearning one point affects only one shard!

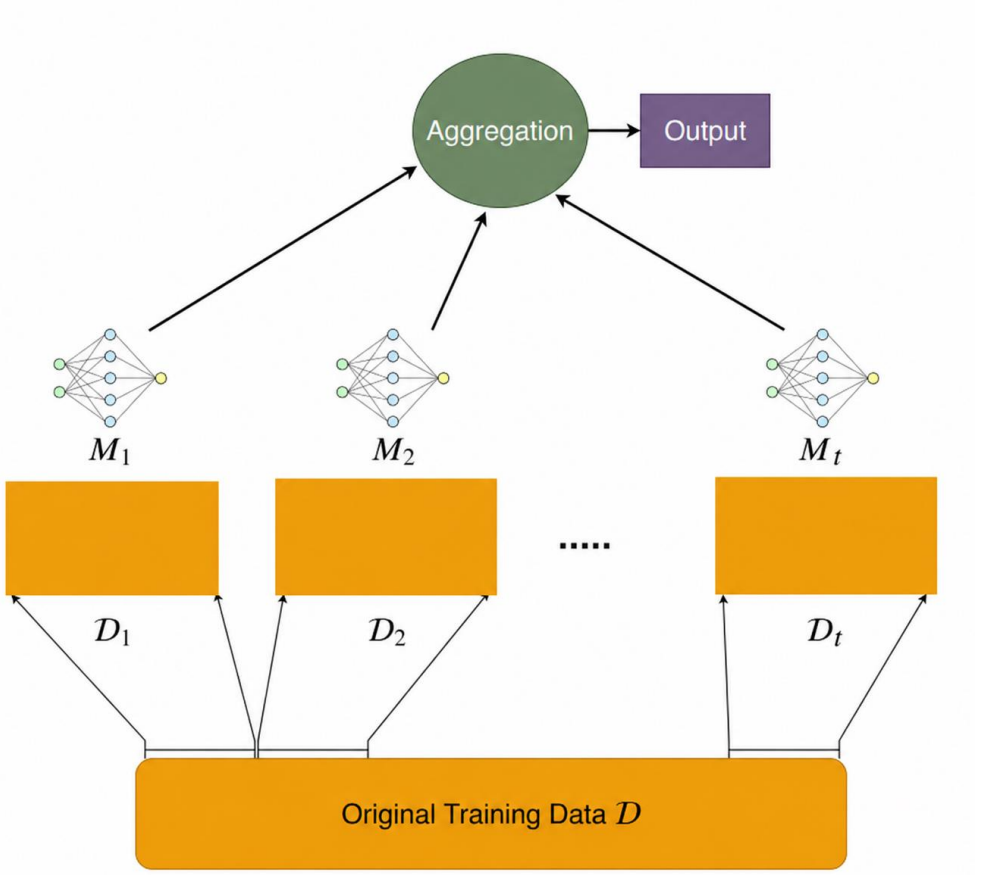


Figure: Modified from [Bourtoule et al., Oakland 2021] with MS Paint, ChatGPT 5.5 Pro

Unlearning with (Baby) SISA

- If an unlearn request falls into shard 2, then only have to retrain model 2
  - Retrain from scratch would have to retrain all models in ensemble
- Saves a factor of t in unlearning time
  - Can be, e.g., a factor of 10

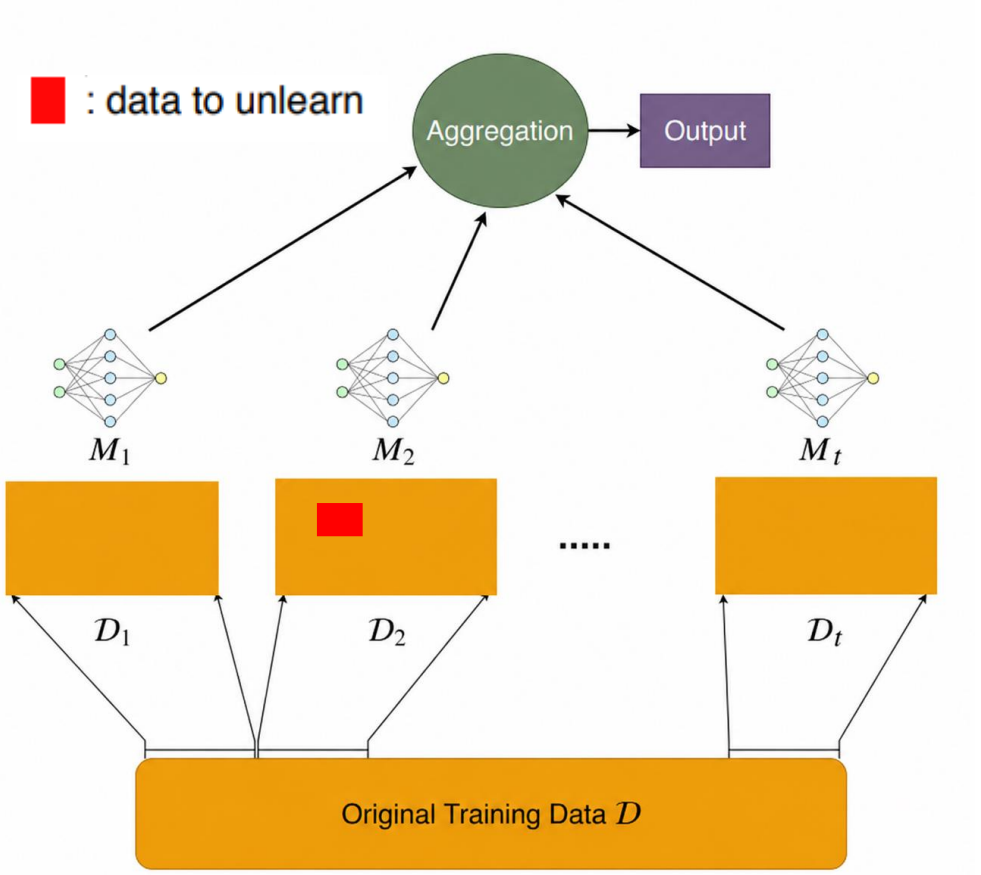


Figure: Modified from [Bourtoule et al., Oakland 2021] with MS Paint, ChatGPT 5.5 Pro

Unlearning with (Baby) SISA

- No additional cost if multiple unlearn requests happen to fall in the same shard!

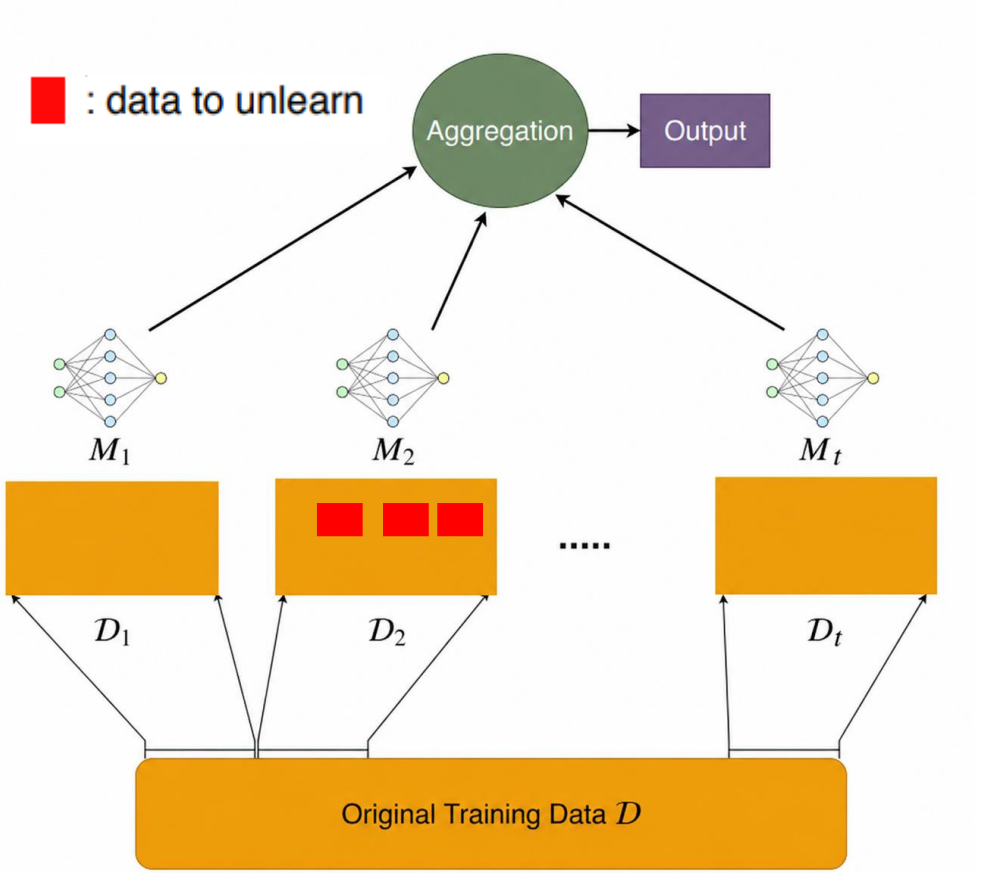


Figure: Modified from [Bourtoule et al., Oakland 2021] with MS Paint, ChatGPT 5.5 Pro

Unlearning with (Baby) SISA

- But can be no faster than retraining from scratch if unlearning requests fall in different shards
- If number of unlearning requests ≫ number of shards, not likely to have much savings
  - So can only handle a small number of points in an unlearning request and save

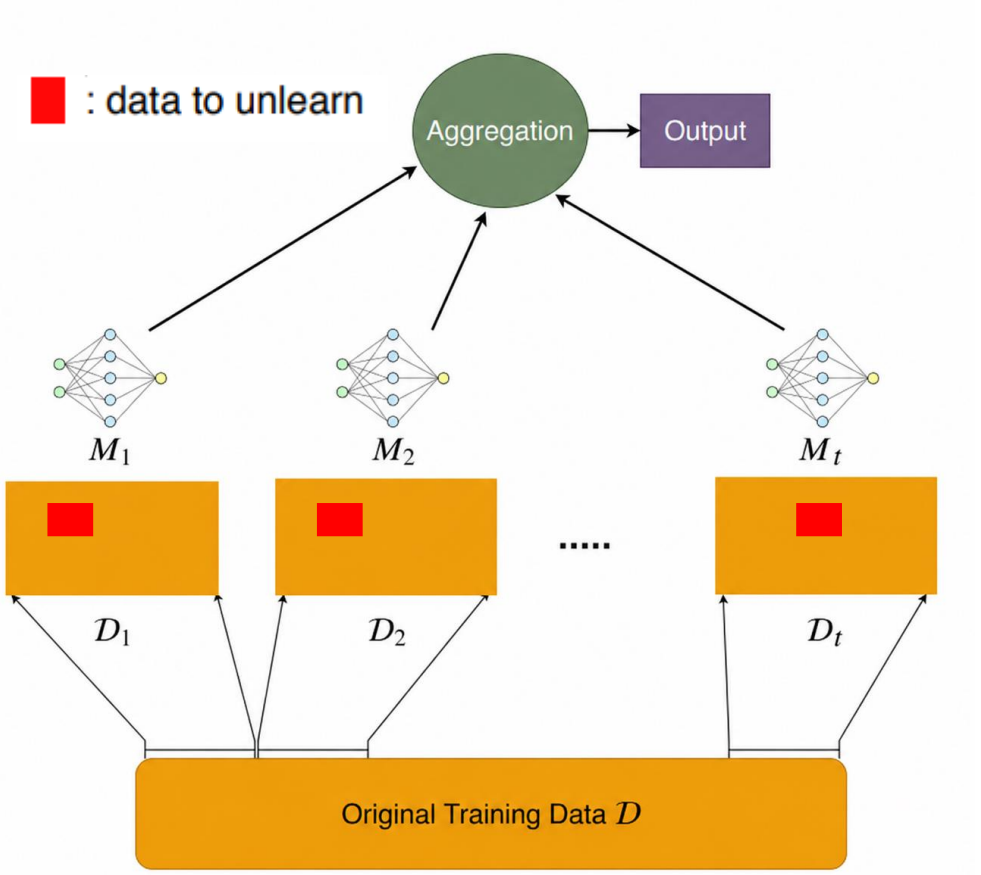


Figure: Modified from [Bourtoule et al., Oakland 2021] with MS Paint, ChatGPT 5.5 Pro

Full SISA

- Add slicing on top
- Each shard is split into R slices
- Train on slice 1 in first epoch,
- slice 1+2 in second epoch,
- slice 1+2+3 in third epoch, …
- Save checkpoint after each epoch
- If unlearn request falls into the last slice, only need to retrain one epoch!

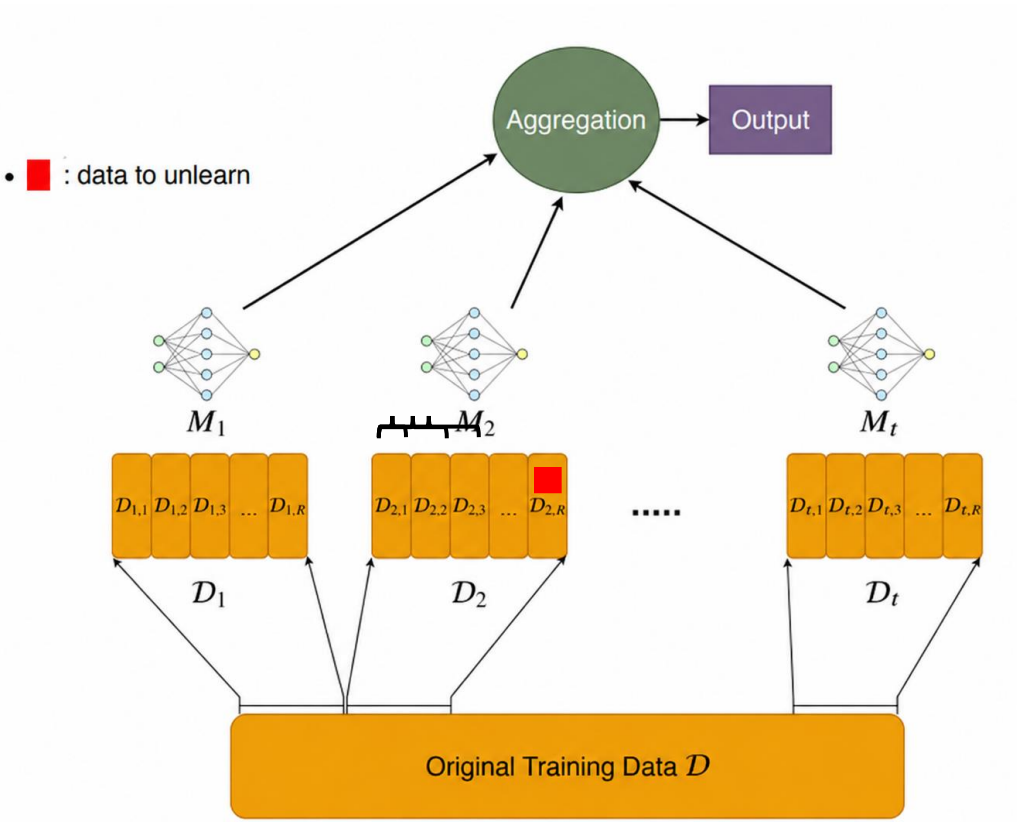


Figure: Modified from [Bourtoule et al., Oakland 2021] with ChatGPT 5.5 Pro

SISA pros, cons, and tradeoffs

- Can speed up unlearning one point by an order of magnitude
  - But savings decay quickly for more requests, if $\geq 3t$
  - Better off retraining from scratch for requests with more than a few dozen points
- Have to choose number of shards t and slices R
  - More shards = faster unlearning
    - But worse utility due to ensembling, plus a larger model to store all the weights
  - More slices = faster unlearning (to a limit)
    - But more memory required to store all the checkpoints

Summary: SISA

Consideration	Description

Summary: SISA

Consideration	Description
Unlearning time	Can be an order of magnitude speedup, though degrades quickly for larger unlearning sets

Summary: SISA

Consideration	Description
Unlearning time	Can be an order of magnitude speedup, though degrades quickly for larger unlearning sets
Supported models	Very flexible, can support any model trained with gradient descent

Summary: SISA

Consideration	Description
Unlearning time	Can be an order of magnitude speedup, though degrades quickly for larger unlearning sets
Supported models	Very flexible, can support any model trained with gradient descent
Effect of large unlearning sets	Reverts to retraining from scratch for moderately sized unlearning sets

Summary: SISA

Consideration	Description
Unlearning time	Can be an order of magnitude speedup, though degrades quickly for larger unlearning sets
Supported models	Very flexible, can support any model trained with gradient descent
Effect of large unlearning sets	Reverts to retraining from scratch for moderately sized unlearning sets
Utility	Exact unlearning, so no utility cost. But switching to an ensemble may result in a hit to utility.

Summary: SISA

Consideration	Description
Unlearning time	Can be an order of magnitude speedup, though degrades quickly for larger unlearning sets
Supported models	Very flexible, can support any model trained with gradient descent
Effect of large unlearning sets	Reverts to retraining from scratch for moderately sized unlearning sets
Utility	Exact unlearning, so no utility cost. But switching to an ensemble may result in a hit to utility.
Training overhead	Minimal

Summary: SISA

Consideration	Description
Unlearning time	Can be an order of magnitude speedup, though degrades quickly for larger unlearning sets
Supported models	Very flexible, can support any model trained with gradient descent
Effect of large unlearning sets	Reverts to retraining from scratch for moderately sized unlearning sets
Utility	Exact unlearning, so no utility cost. But switching to an ensemble may result in a hit to utility.
Training overhead	Minimal
Auxiliary information	Significant, must store entire dataset, along with all the models in ensemble and checkpoints during training

Part 2: Definitions and Algorithms

- Exact Unlearning
  - Retrain from Scratch
  - Unlearning with structure
  - SISA
- **Approximate Unlearning**
  - Differential Privacy
  - Influence Functions

Approximate Unlearning

- Exact unlearning is a very strong requirement!
- Relax to *approximate* unlearning

M and U satisfy *exact unlearning* if,

for any training dataset X and unlearning dataset $S \subseteq X$

$$M(X \setminus S) \stackrel{d}{=} U(M(X), S, \mathrm{Aux})$$

Approximate Unlearning

- Exact unlearning is a very strong requirement!
- Relax to *approximate* unlearning

M and U satisfy *approximate unlearning* if,

for any training dataset X and unlearning dataset $S \subseteq X$

$$M(X \setminus S) \overset{d}{\approx} U(M(X), S, \mathrm{Aux})$$

- How should we define *approximate*/$\approx$?
- Same way as in differential privacy, parameterized by ε, δ
  - [Dwork, McSherry, Nissim, Smith, TCC '06]

Approximate Unlearning

M and U satisfy (ε, δ)-*approximate unlearning* if,

for any training dataset X and unlearning dataset $S \subseteq X$

$$M(X \setminus S) \overset{d}{\approx}_{\varepsilon,\delta} U(M(X), S, \mathrm{Aux})$$

- Operationally: the model behaves very similarly in both universes
- Surprisingly: Can sometimes *mathematically prove* this guarantee!

Part 2: Definitions and Algorithms

- Exact Unlearning
  - Retrain from Scratch
  - Unlearning with structure
  - SISA
- Approximate Unlearning
  - **Differential Privacy**
  - Influence Functions

Differential Privacy (DP)

- A popular and rigorous notion of data privacy
  - Employed in practice by Google, Apple, US Census Bureau, etc.
  - Much more mature than machine unlearning
- M is (ε, δ)-DP if, for any datasets X and X' that differ in exactly one element

$$M(X) \overset{d}{\approx}_{\varepsilon,\delta} M(X')$$

- *Very* similar to approximate machine unlearning definition

Differential Privacy (DP)

- Differential privacy: $M(X) \overset{d}{\approx}_{\varepsilon,\delta} M(X')$
  - Models from training on datasets X and X' that differ in 1 entry are similarly distributed
- Machine Unlearning: $U(M(X), S, \mathrm{Aux}) \overset{d}{\approx}_{\varepsilon,\delta} M(X \setminus S)$
  - Models from unlearning S vs never training on S are similarly distributed
  - What if unlearning algorithm does nothing? $U(M(X), S, \mathrm{Aux}) = M(X)$
  - What if we unlearn only 1 point? Call $X' \triangleq X \setminus S$, differs in 1 entry from X
  - $M(X) \overset{d}{\approx}_{\varepsilon,\delta} M(X')$: the exact same as DP!
- DP implies machine unlearning automatically!

Automatic Unlearning via DP

- Suppose M is an (ε, δ)-DP learning algorithm
- Let $U(M(X), S, \mathrm{Aux}) = M(X)$
  - The unlearning algorithm "does nothing"
- Then M and U satisfy (ε, δ)-approximate unlearning for all training datasets X and unlearning sets $S \subseteq X$ of size 1
  - Without doing anything!
- Any differentially private algorithm has approximately unlearned every set of size 1, simultaneously!
  - A very strong guarantee – can do better with methods tailored to S
    - [Sekhari et al., NeurIPS 2021]

Unlearning Multiple points via DP?

- DP guarantee holds only for differences of size 1
- What if the unlearning set S is of size k?
- Guarantees degrade:

$$U(M(X), S, \mathrm{Aux}) \overset{d}{\approx}_{k\varepsilon,\delta} M(X \setminus S)$$

- This is a dramatically weaker guarantee (exponentially worse in k)
- Can compensate by scaling down privacy parameter ε, but utility hurts fast

Algorithms for training models with DP

- Differentially Private Stochastic Gradient Descent (DPSGD)
  - [Song et al., GlobalSIP ’13], [Bassily, Smith, Thakurta, FOCS ‘14], [Abadi et al., CCS ‘16]
- Serves as a drop-in replacement for SGD
- Any model trained with SGD can be made private with DPSGD
- Utility loss can be modest to significant
  - Can be very bad if privacy parameter ε is very low to handle large unlearning sets S

Summary: Differential Privacy

Consideration	Description

Summary: Differential Privacy

Consideration	Description
Unlearning time	Instant, no modifications required

Summary: Differential Privacy

Consideration	Description
Unlearning time	Instant, no modifications required
Supported models	Any model that can be trained with SGD

Summary: Differential Privacy

Consideration	Description
Unlearning time	Instant, no modifications required
Supported models	Any model that can be trained with SGD
Effect of large unlearning sets	Degrades utility significantly to maintain reasonable guarantees

Summary: Differential Privacy

Consideration	Description
Unlearning time	Instant, no modifications required
Supported models	Any model that can be trained with SGD
Effect of large unlearning sets	Degrades utility significantly to maintain reasonable guarantees
Utility	Degrades quickly for large unlearning sets

Summary: Differential Privacy

Consideration	Description
Unlearning time	Instant, no modifications required
Supported models	Any model that can be trained with SGD
Effect of large unlearning sets	Degrades utility significantly to maintain reasonable guarantees
Utility	Degrades quickly for large unlearning sets
Training overhead	Minimal, in modern pipelines

Summary: Differential Privacy

Consideration	Description
Unlearning time	Instant, no modifications required
Supported models	Any model that can be trained with SGD
Effect of large unlearning sets	Degrades utility significantly to maintain reasonable guarantees
Utility	Degrades quickly for large unlearning sets
Training overhead	Minimal, in modern pipelines
Auxiliary information	None

Part 2: Definitions and Algorithms

- Exact Unlearning
  - Retrain from Scratch
  - Unlearning with structure
  - SISA
- Approximate Unlearning
  - Differential Privacy
  - **Influence Functions**

Influence Functions

- Classic statistical technique, popularized in ML by [Koh and Liang, ICML 2017] to understand influence of individual training points
- “If we changed the weight of a single training data point by an infinitesimal amount, how much would the parameters of a model trained on that dataset change?”
- Recently used by many works as an effective tool for unlearning
  - [Guo et al., ICML 2020], [Golatkar et al., CVPR 2020], [Sekhari et al., NeurIPS 2021], [Suriyakumar and Wilson, NeurIPS 2022], etc.

What is an Influence Function?

- Training dataset $\{z_i\}_{i=1}^n$, strongly convex loss function ℓ
- Empirical risk minimization: $\hat{\theta} \triangleq \arg\min_\theta \frac{1}{n}\sum_{i=1}^n \ell\left(z_i, \theta\right)$
- Increase weight of point z_j by γ_n, let $\hat{\theta}_{\gamma,j}$ be new ERM solution

$$\hat{\theta}_{\gamma,j} \triangleq \arg\min_\theta \frac{1}{n}\sum_{i=1}^n \ell\left(z_i, \theta\right) + \gamma\ell\left(z_j, \theta\right)$$

- Influence function: $\mathcal{I}\left(z_j\right) \triangleq \left.\frac{d\hat{\theta}_{\gamma,j}}{d\gamma}\right|_{\gamma=0} = -H_{\hat{\theta}}^{-1}\nabla_\theta \ell\left(z_j, \hat{\theta}\right)$,
  - $H_{\hat{\theta}} \triangleq \frac{1}{n}\sum_{i=1}^n \nabla_\theta^2 \ell\left(z_i, \hat{\theta}\right)$ is Hessian of the loss at the solution $\hat{\theta}$
- Allows us to approximate the change in θ: $\hat{\theta}_{\gamma,j} \approx \hat{\theta} + \gamma\mathcal{I}\left(z_j\right)$.

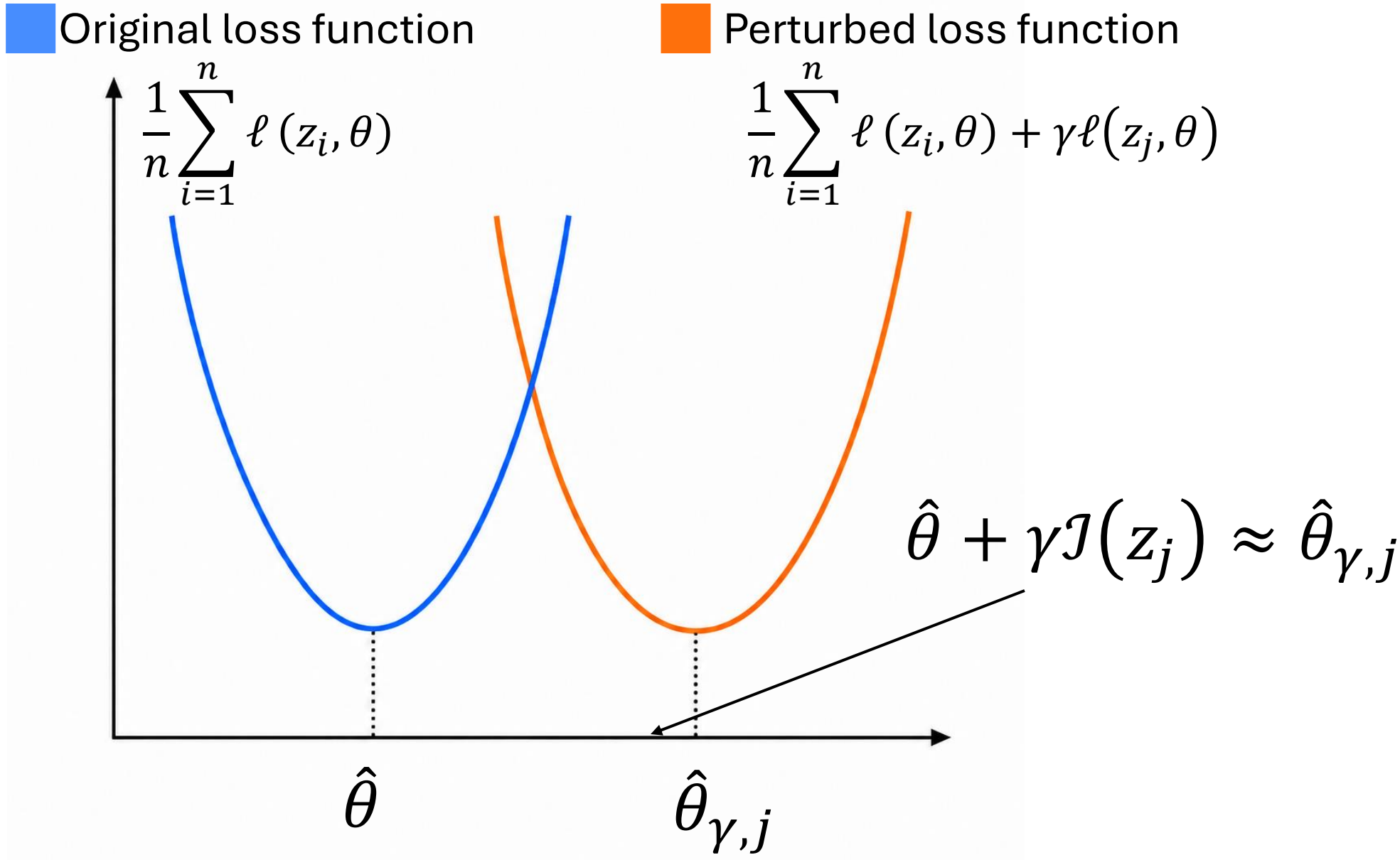


Figure: ChatGPT 5.5 Pro

Using Influence Functions for Unlearning

- ERM solution without z_j sets $\gamma = -1/n$:

$$\hat{\theta}_{-1/n,j} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1} \ell\left(z_i, \theta\right) - \frac{1}{n} \ell\left(z_j, \theta\right)$$

- Use influence function approximation:

$$\hat{\theta}_{-1/n,j} \approx \hat{\theta} - \frac{1}{n} \mathcal{I}\left(z_j\right) = \hat{\theta} + \frac{1}{n} H_{\hat{\theta}}^{-1} \nabla_{\theta} \ell\left(z_j, \hat{\theta}\right).$$

- Influence functions: approximately unlearn a point via a Newton step
- Computational cost?
  - $O(nd^2 + d^3)$ at training time, for computing and inverting Hessian
  - $O(kd + d^2)$ at unlearning time, to compute gradients and matrix-vector prod.

Other notes on Influence Functions

- Only *approximate* the solution after unlearning
- Must add Gaussian noise to satisfy approx. unlearning definition
  - A la Gaussian mechanism in differential privacy
- Biggest restriction: only works with provable guarantees for convex models
  - Some effective applications to non-convex models [McKinney et al., SaTML 2026]

Summary: Influence functions

Consideration	Description

Summary: Influence functions

Consideration	Description
Unlearning time	Reasonable, $O(d^2)$ time

Summary: Influence functions

Consideration	Description
Unlearning time	Reasonable, $O(d^2)$ time
Supported models	Only convex models

Summary: Influence functions

Consideration	Description
Unlearning time	Reasonable, $O(d^2)$ time
Supported models	Only convex models
Effect of large unlearning sets	Minimal, as running time scales gracefully. However, utility of approximation will degrade.

Summary: Influence functions

Consideration	Description
Unlearning time	Reasonable, $O(d^2)$ time
Supported models	Only convex models
Effect of large unlearning sets	Minimal, as running time scales gracefully. However, utility of approximation will degrade.
Utility	Modest decrease in utility

Summary: Influence functions

Consideration	Description
Unlearning time	Reasonable, $O(d^2)$ time
Supported models	Only convex models
Effect of large unlearning sets	Minimal, as running time scales gracefully. However, utility of approximation will degrade.
Utility	Modest decrease in utility
Training overhead	Can be significant, requires computing and inverting Hessian matrix

Summary: Influence functions

Consideration	Description
Unlearning time	Reasonable, $O(d^2)$ time
Supported models	Only convex models
Effect of large unlearning sets	Minimal, as running time scales gracefully. However, utility of approximation will degrade.
Utility	Modest decrease in utility
Training overhead	Can be significant, requires computing and inverting Hessian matrix
Auxiliary information	Requires storing the inverse Hessian matrix

Discussion

- Exact and approximate unlearning definitions
- Variety of algorithms
- Each has its own deficiencies…
  - Unlearning time, low unlearning capacity, poor utility, only works for restricted models, etc.
- But if you must be sure unlearning worked, these are the options!
- Next part: heuristic methods for unlearning
  - Can be more efficient and broadly applicable
  - But may not actually unlearn correctly…