CY 7790

Special Topics in Security and Privacy: Machine Learning Security and Privacy

Fall 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science

November 15 2021

Stealing Machine Learning Models via Prediction APIs

Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, Thomas Ristenpart

Presented by Lisa Oakley, CY 7790, Fall 2021

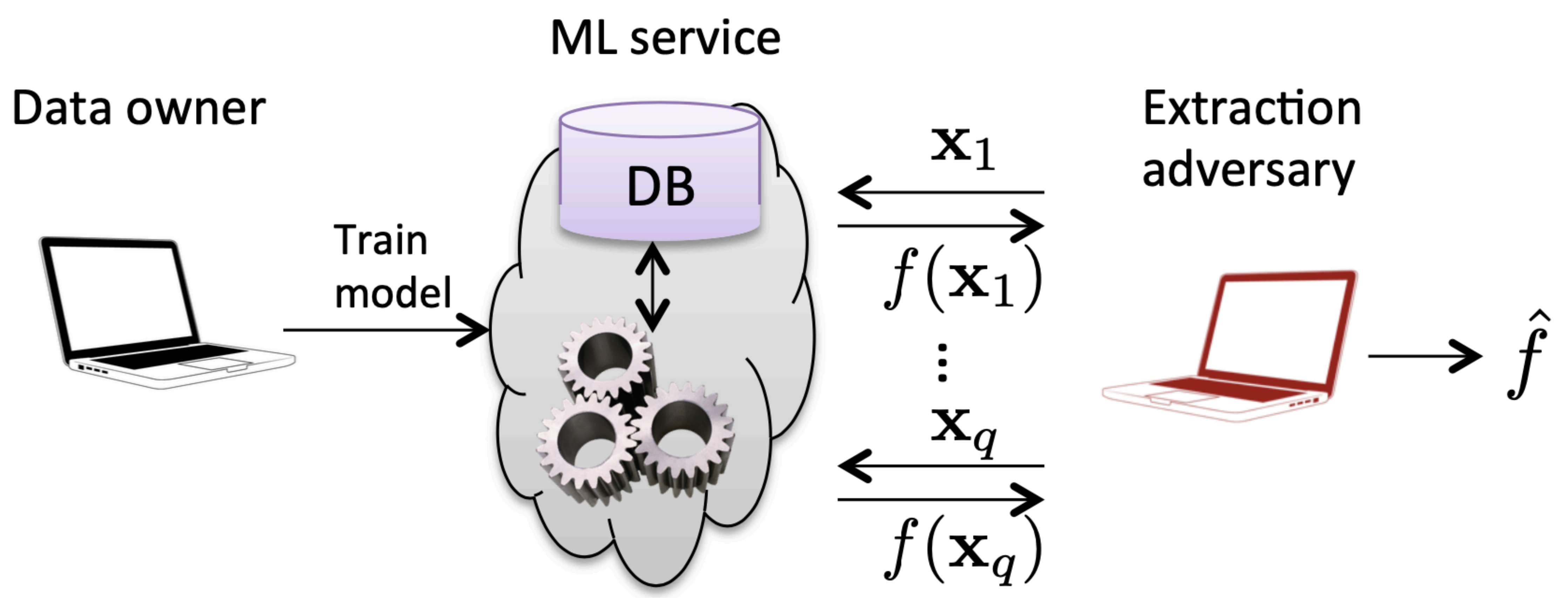


Figure 1: **Diagram of ML model extraction attacks.** A data owner has a model f trained on its data and allows others to make prediction queries. An adversary uses q prediction queries to extract an $\hat{f} \approx f$.

Problem Statement

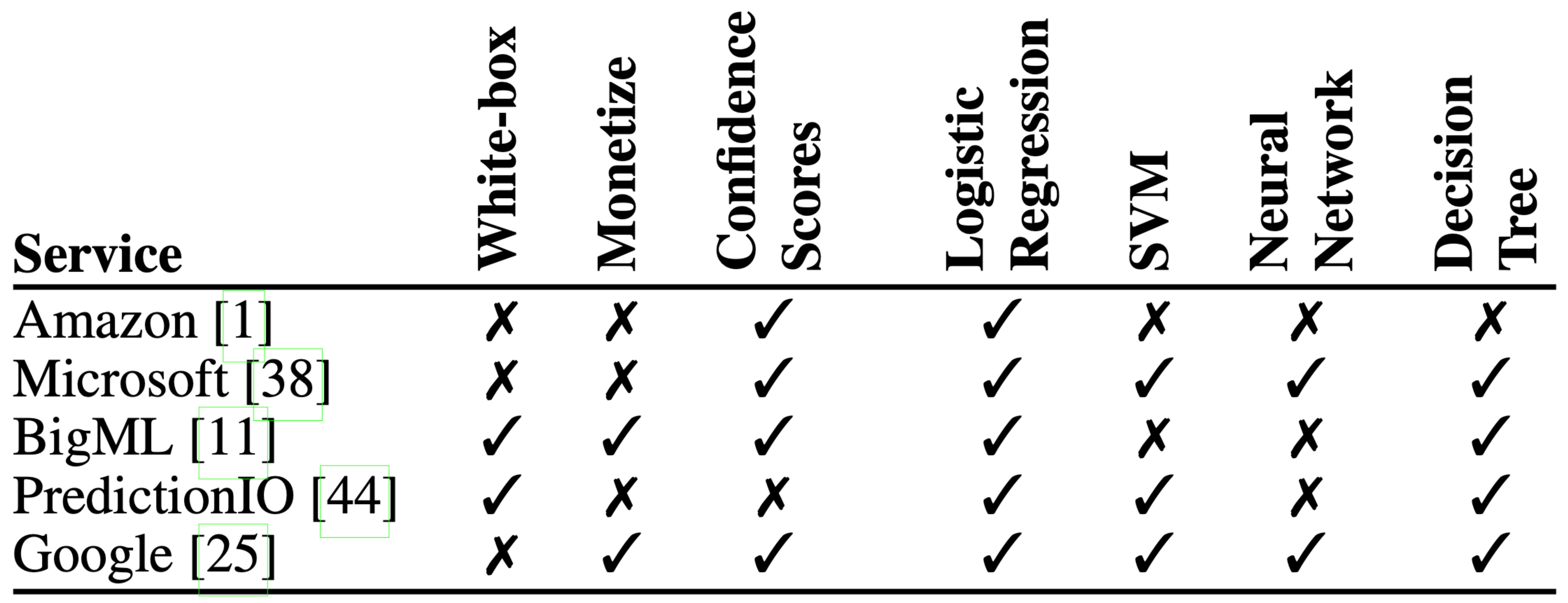
Develop a *Model Extraction Attack,* whereby an attacker with query access to a private target ML model generates a (nearly) equivalent model.

Why a Private Model?

1. Model is **proprietary/ monetized**, pay-per-query

2. **Training data is sensitive** and white-box access to model leaks info about training data.

3. Model is used for **security purposes** where attacker knowledge would improve evasion attacks (e.g. spam detection)



Service	**White-box**	**Monetize**	**Confidence Scores**	**Logistic Regression**	**SVM**	**Neural Network**	**Decision Tree**
Amazon [1]	✗	✗	✓	✓	✗	✗	✗
Microsoft [38]	✗	✗	✓	✓	✓	✓	✓
BigML [11]	✓	✓	✓	✓	✗	✗	✓
PredictionIO [44]	✓	✗	✗	✓	✓	✗	✓
Google [25]	✗	✓	✓	✓	✓	✓	✓

Table 2: **Particularities of major MLaaS providers.** 'White-box' refers to the ability to download and use a trained model locally, and 'Monetize' means that a user may charge other users for black-box access to her models. Model support for each service is obtained from available documentation. The models listed for Google's API are a projection based on the announced support of models in standard PMML format [25]. Details on ML models are given in Appendix A.

Threat Model

Goal: **Model Extraction**

- Attacker aims to learn a model which “closely matches” target model

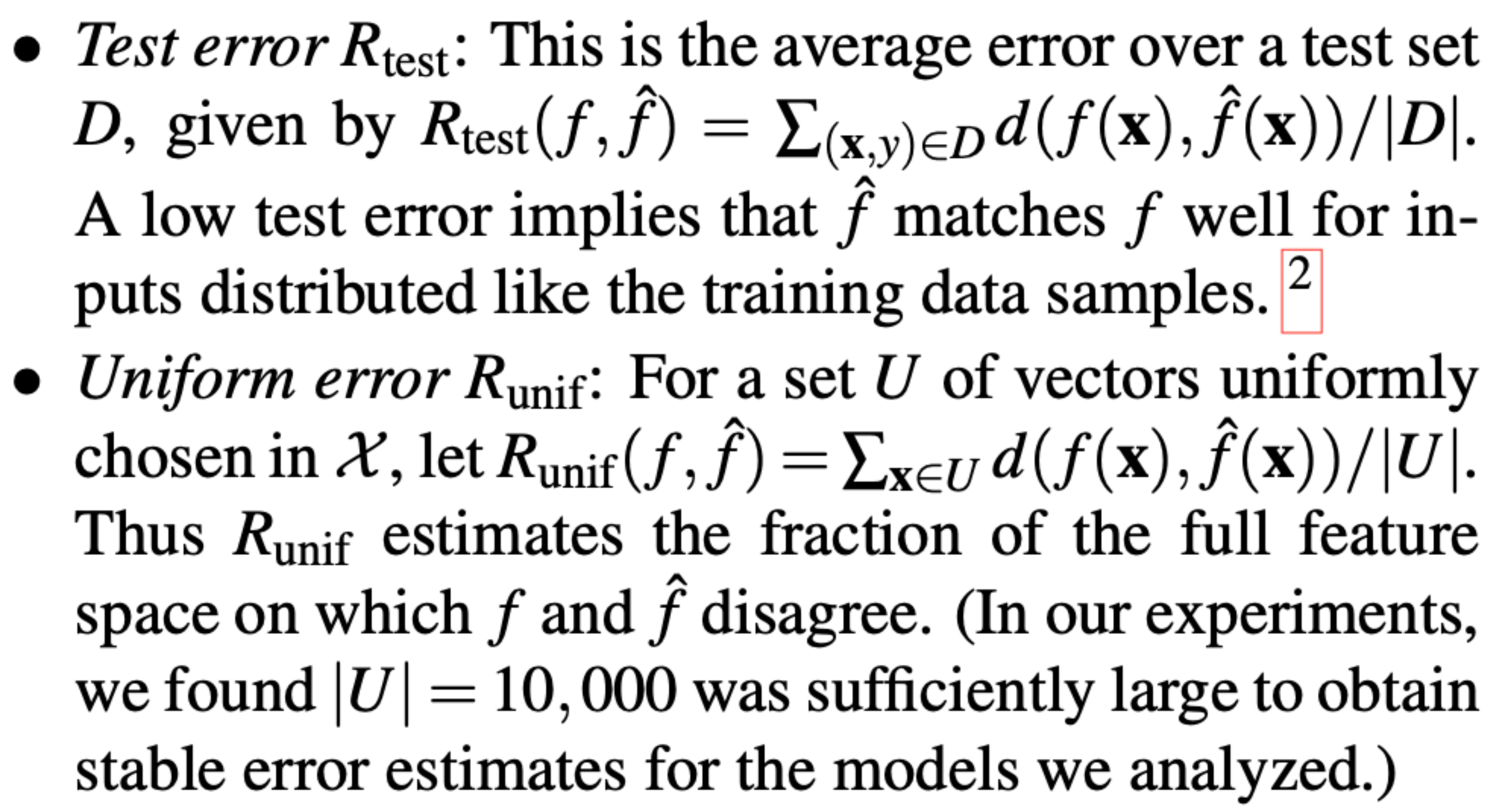
Capabilities: **Query**

- *Direct queries* (no feature extraction)
- *Indirect queries* (feature extraction)

Knowledge: “**Black Box**”

- *Model Class?*
- *Training Algorithm?*
- *Model Hyper-Parameters?*
- *Feature Extraction?*
- *Confidence Scores?*
- *Training Data Statistics?*

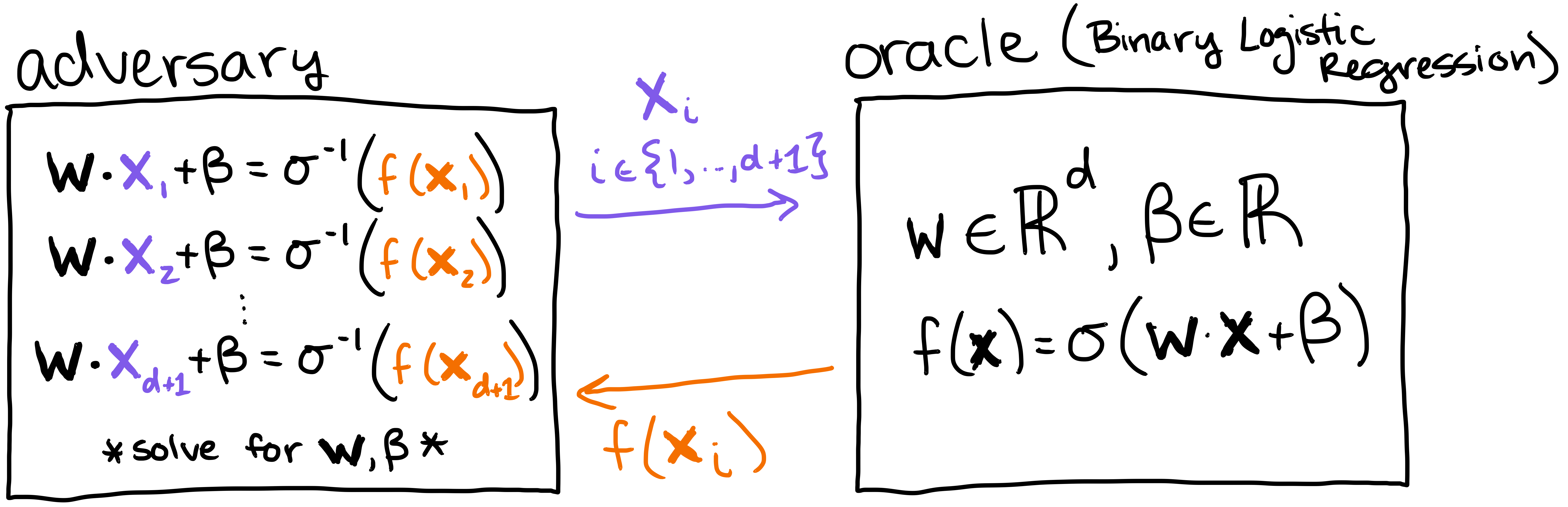
Defining “closely matching”



- *Test error* R_{test}: This is the average error over a test set D, given by $R_{\text{test}}(f,\hat{f}) = \sum_{(\mathbf{x},y)\in D} d(f(\mathbf{x}),\hat{f}(\mathbf{x}))/|D|$. A low test error implies that $\hat{f}$ matches f well for inputs distributed like the training data samples. [2]
- *Uniform error* R_{unif}: For a set U of vectors uniformly chosen in $\mathcal{X}$, let $R_{\text{unif}}(f,\hat{f}) = \sum_{\mathbf{x}\in U} d(f(\mathbf{x}),\hat{f}(\mathbf{x}))/|U|$. Thus R_{unif} estimates the fraction of the full feature space on which f and $\hat{f}$ disagree. (In our experiments, we found $|U| = 10,000$ was sufficiently large to obtain stable error estimates for the models we analyzed.)

Equation-Solving Attacks

For Binary Logistic Regression

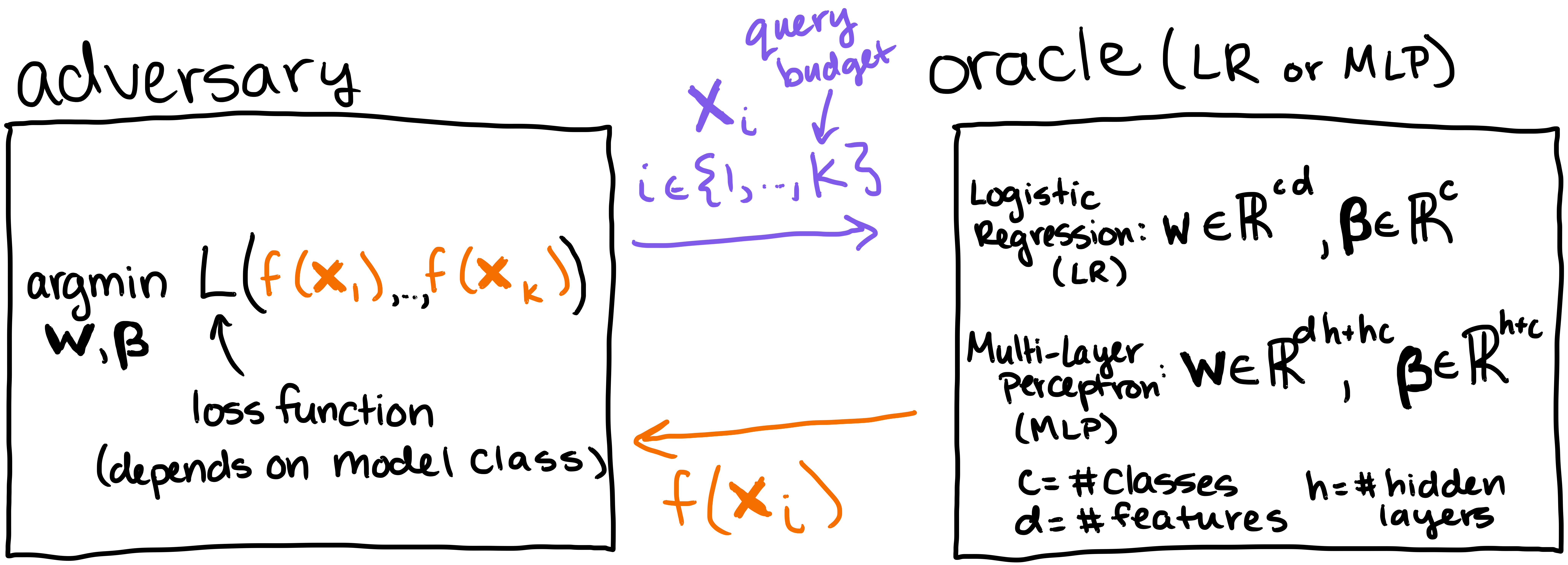

adversary
$w \cdot x_1 + \beta = \sigma^{-1}(f(x_1))$
$w \cdot x_2 + \beta = \sigma^{-1}(f(x_2))$
$\vdots$
$w \cdot x_{d+1} + \beta = \sigma^{-1}(f(x_{d+1}))$
solve for w, β
x_i
$i \in \{1, \ldots, d+1\}$
$f(x_i)$
oracle (Binary Logistic Regression)
$w \in \mathbb{R}^d, \beta \in \mathbb{R}$
$f(x) = \sigma(w \cdot x + \beta)$

$$\sigma(t) = \frac{1}{1+e^{-t}}$$

$$\sigma^{-1}(t) = \ln\left(\frac{t}{(1-t)}\right)$$

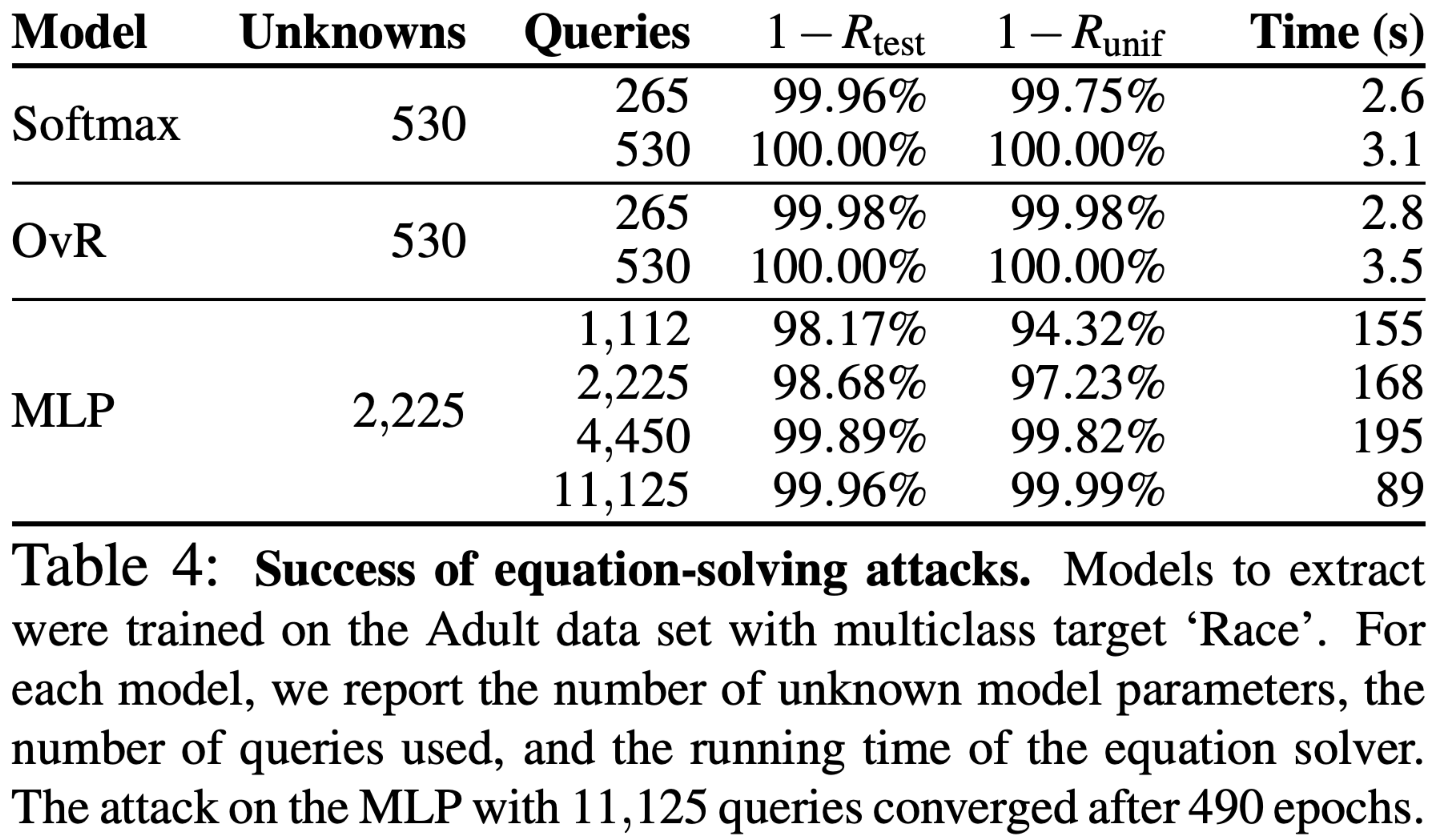
Equation-Solving Attacks

For Multi-Class Logistic Regression and Neural Networks


adversary
$\underset{W,\beta}{\operatorname{argmin}}\ L(f(x_1),\dots,f(x_k))$
loss function
(depends on model class)
query budget
x_i
$i \in \{1,\dots,k\}$
$f(x_i)$
oracle (LR or MLP)
Logistic Regression (LR): $W \in \mathbb{R}^{cd}$, $\beta \in \mathbb{R}^{c}$
Multi-Layer Perceptron (MLP): $W \in \mathbb{R}^{dh+hc}$, $\beta \in \mathbb{R}^{h+c}$
c = # classes
d = # features
h = # hidden layers

Equation-Solving Attacks

For Multi-Class Logistic Regression and Neural Networks



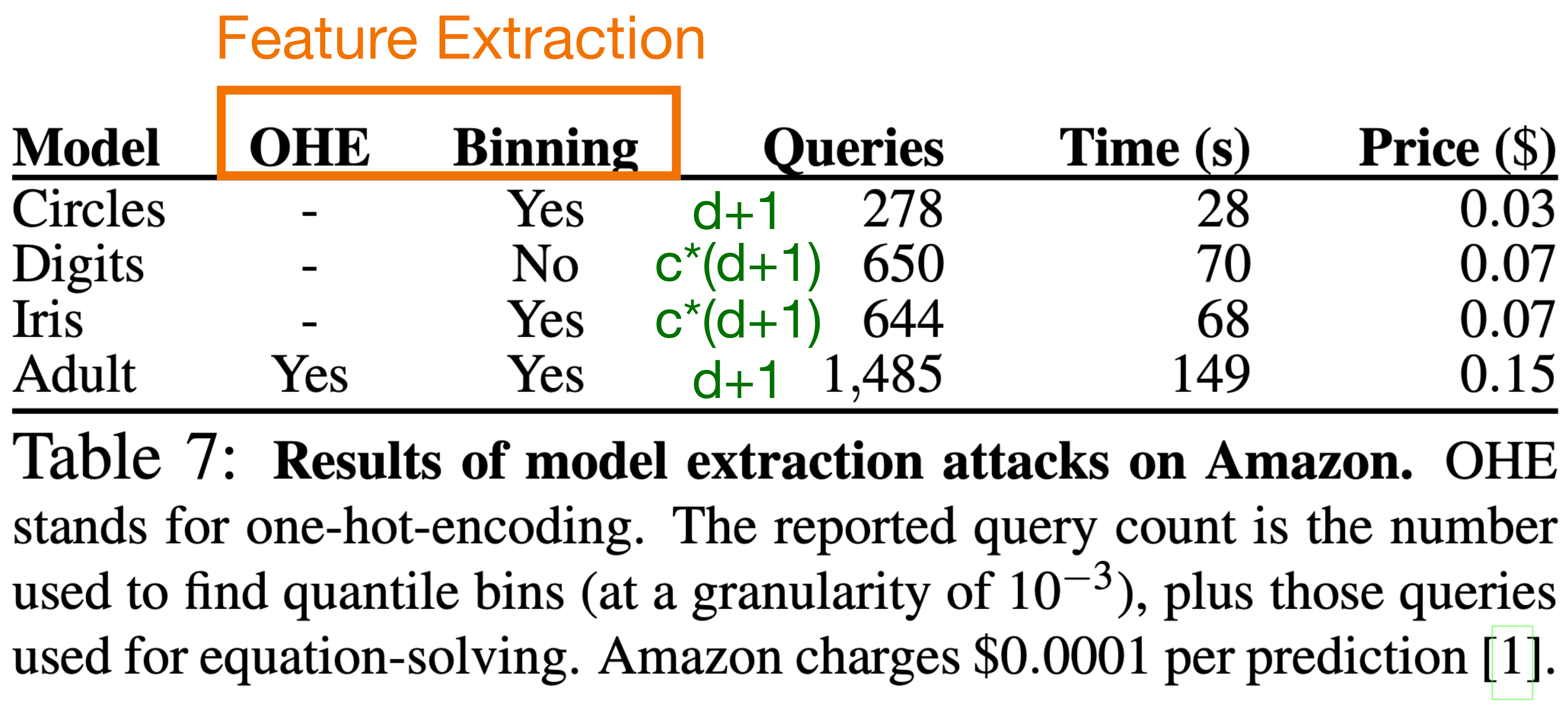
Model	Unknowns	Queries	$1-R_{\text{test}}$	$1-R_{\text{unif}}$	Time (s)
Softmax	530	265	99.96%	99.75%	2.6
		530	100.00%	100.00%	3.1
OvR	530	265	99.98%	99.98%	2.8
		530	100.00%	100.00%	3.5
MLP	2,225	1,112	98.17%	94.32%	155
		2,225	98.68%	97.23%	168
		4,450	99.89%	99.82%	195
		11,125	99.96%	99.99%	89

Table 4: **Success of equation-solving attacks.** Models to extract were trained on the Adult data set with multiclass target 'Race'. For each model, we report the number of unknown model parameters, the number of queries used, and the running time of the equation solver. The attack on the MLP with 11,125 queries converged after 490 epochs.

“Online” Model Extraction Attacks

For Amazon

- Multi-Class Logistic Regression
- Feature Extraction — one hot encoding for categorical, k-quantile bins for numeric
- Line search to find feature extraction parameters
- Queries with missing features result in 0 value in the feature space
- Known: model class, feature extraction class, some hyperparameters



	Feature Extraction					
Model	**OHE**	**Binning**		**Queries**	**Time (s)**	**Price ($)**
Circles	-	Yes	d+1	278	28	0.03
Digits	-	No	c*(d+1)	650	70	0.07
Iris	-	Yes	c*(d+1)	644	68	0.07
Adult	Yes	Yes	d+1	1,485	149	0.15

Table 7: **Results of model extraction attacks on Amazon.** OHE stands for one-hot-encoding. The reported query count is the number used to find quantile bins (at a granularity of 10^{-3}), plus those queries used for equation-solving. Amazon charges $0.0001 per prediction [1].

Path Finding Attacks

For Decision Trees

- Decision Trees do not compute probability, instead partition space.
- Solution: attempt to identify the path the input traversed
- Adaptive query strategy
- Leverage queries with incomplete inputs

Path Finding Attacks

For Decision Trees

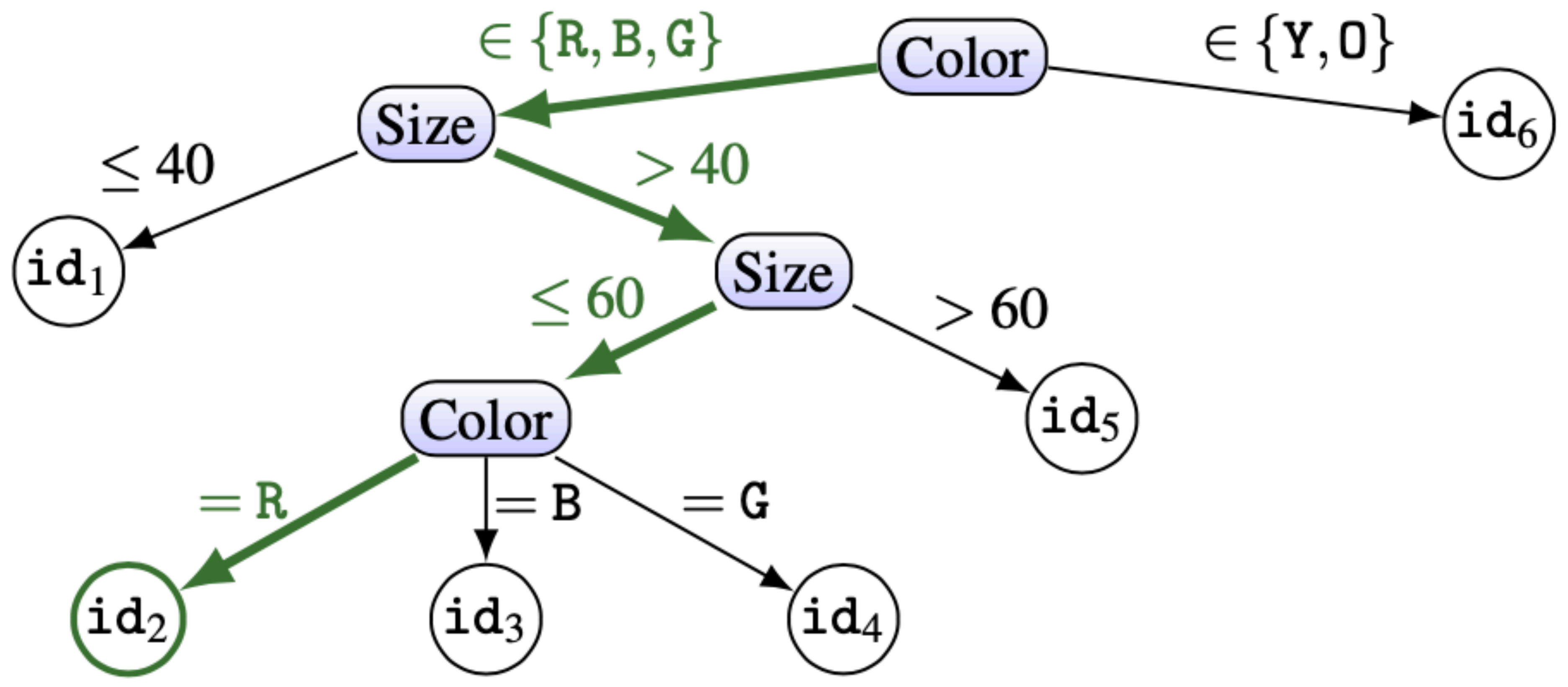


Figure 3: **Decision tree over features Color and Size.** Shows the path (thick green) to leaf $\mathtt{id}_2$ on input $\mathbf{x} = \{Size = 50,\ Color = R\}$.

Basic Algorithm Intuition:

1. Choose random input **x** and get leaf id of tree output
2. Find the set of features (continuous or categorical) such that **x'** in the feature set results in the same leaf id
3. For each leaf id, find this associated feature set

Path Finding Attacks

For Decision Trees

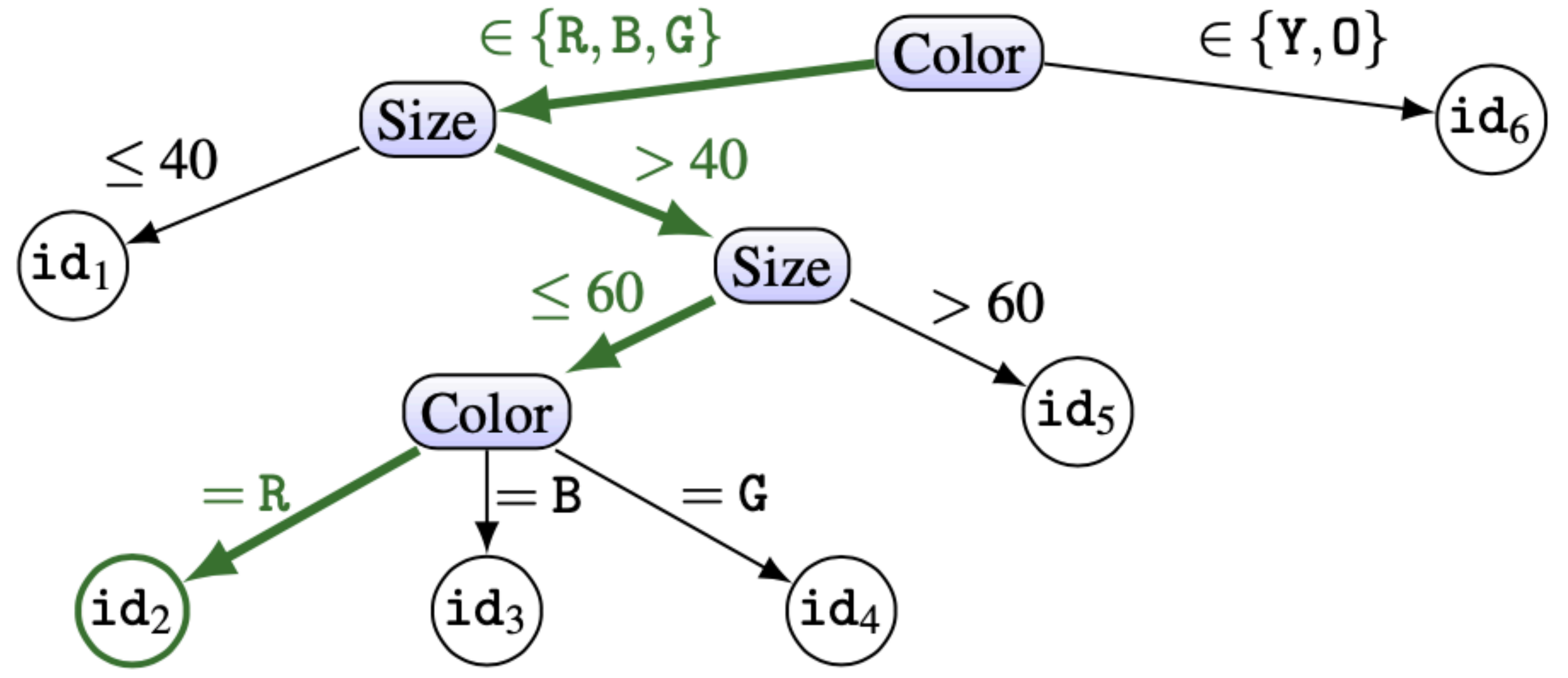


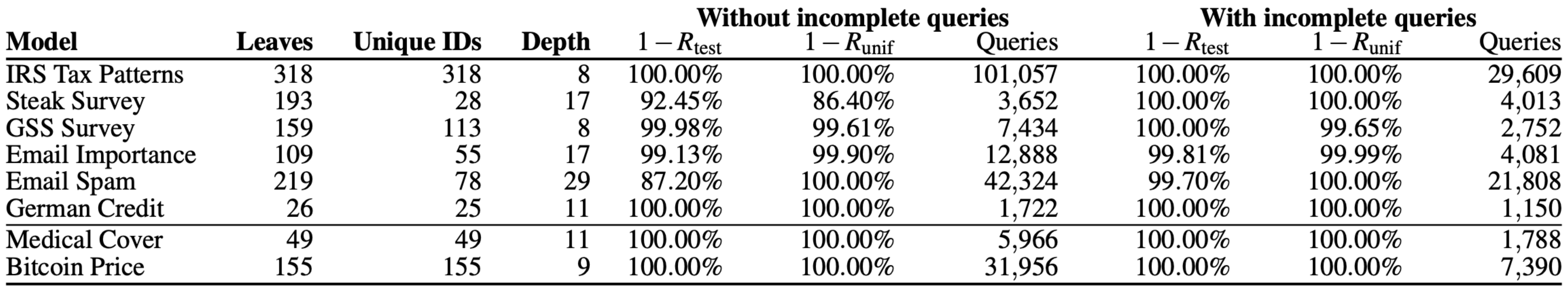
Figure 3: **Decision tree over features Color and Size.** Shows the path (thick green) to leaf id_2 on input $\mathbf{x} = \{Size = 50,\ Color = R\}$.

Top-Down Algorithm Intuition:

1. Start with empty query to get root id
2. Set each feature one by one (leaving the rest empty)
3. When setting a feature results in a new node, this is the feature that this layer of the tree splits
4. Use the basic algorithm to find the splitting criterion of this layer
5. Repeat with the remaining features

“Online” Model Extraction Attacks

For BigML Decision Trees



				Without incomplete queries			With incomplete queries		
Model	**Leaves**	**Unique IDs**	**Depth**	$1 - R_{\text{test}}$	$1 - R_{\text{unif}}$	Queries	$1 - R_{\text{test}}$	$1 - R_{\text{unif}}$	Queries
IRS Tax Patterns	318	318	8	100.00%	100.00%	101,057	100.00%	100.00%	29,609
Steak Survey	193	28	17	92.45%	86.40%	3,652	100.00%	100.00%	4,013
GSS Survey	159	113	8	99.98%	99.61%	7,434	100.00%	99.65%	2,752
Email Importance	109	55	17	99.13%	99.90%	12,888	99.81%	99.99%	4,081
Email Spam	219	78	29	87.20%	100.00%	42,324	99.70%	100.00%	21,808
German Credit	26	25	11	100.00%	100.00%	1,722	100.00%	100.00%	1,150
Medical Cover	49	49	11	100.00%	100.00%	5,966	100.00%	100.00%	1,788
Bitcoin Price	155	155	9	100.00%	100.00%	31,956	100.00%	100.00%	7,390

Table 6: **Performance of extraction attacks on public models from BigML.** For each model, we report the number of leaves in the tree, the number of unique identifiers for those leaves, and the maximal tree depth. The chosen granularity ε for continuous features is 10^{-3}.

Class Label Only Attacks

- **Lowd-Meek Attack**
  - Binary linear classifiers: Find points near decision boundary solve for parameters
  - Kernel methods with computable and invertible projection functions: solve for parameters in linear space
- **Re-training using query responses as labels**
  - Uniform queries: sample m points at random
  - Line search: sample m adaptive queries close to decision boundary
  - Adaptive retraining: query m/r points at random, in batches of m/r, retrain on low-confidence points

Class Label Only Attacks (Binary Target Models)

- **Lowd-Meek Attack**
  - Binary linear classifiers: Find points near decision boundary solve for parameters
  - Kernel methods with computable and invertible projection functions: solve for parameters in linear space
- **Re-training using query responses as labels**
  - Uniform queries: sample m points at random
  - Line search: sample m adaptive queries close to decision boundary
  - Adaptive retraining: query m/r points at random, in batches of m/r, retrain on low-confidence points

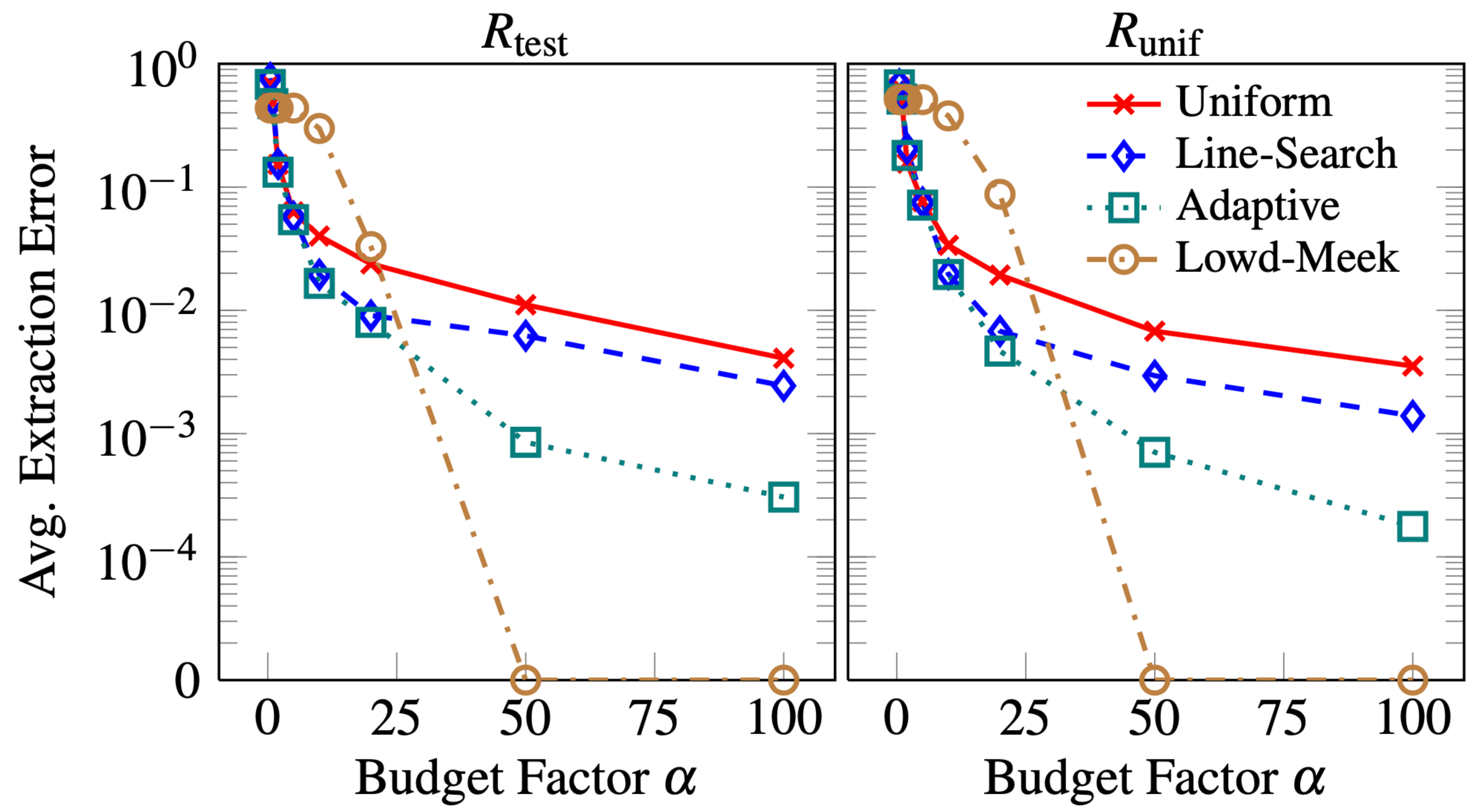


Figure 4: **Average error of extracted linear models.** Results are for different extraction strategies applied to models trained on all binary data sets from Table 3. The left shows R_{test} and the right shows R_{unif}.

Class Label Only Attacks

- **Re-training using query responses as labels**
    - Uniform queries: sample m points at random
    - Line search: sample m adaptive queries close to decision boundary
    - Adaptive retraining: query m/r points at random, in batches of m/r, retrain on low-confidence points

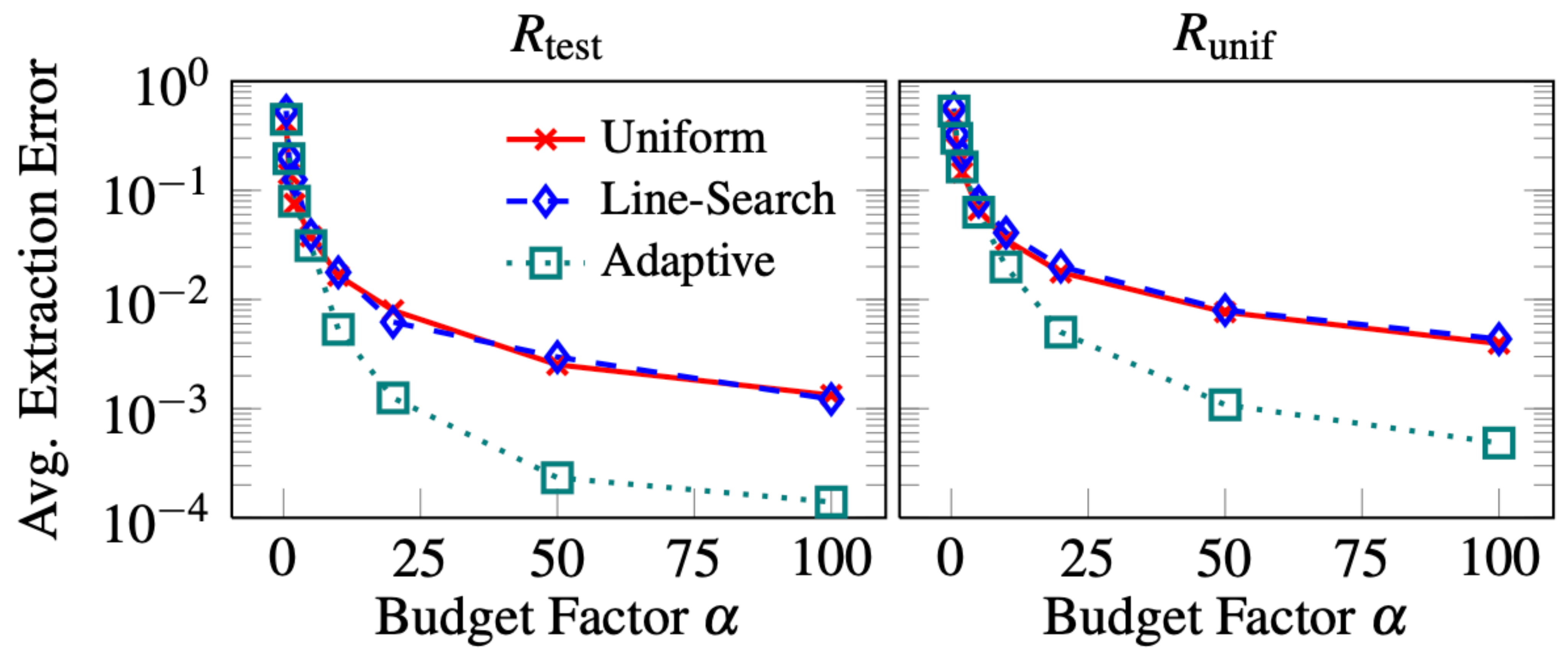


Figure 5: **Average error of extracted softmax models.** Results are for three retraining strategies applied to models trained on all multiclass data sets from Table 3. The left shows R_{test} and the right shows R_{unif}.

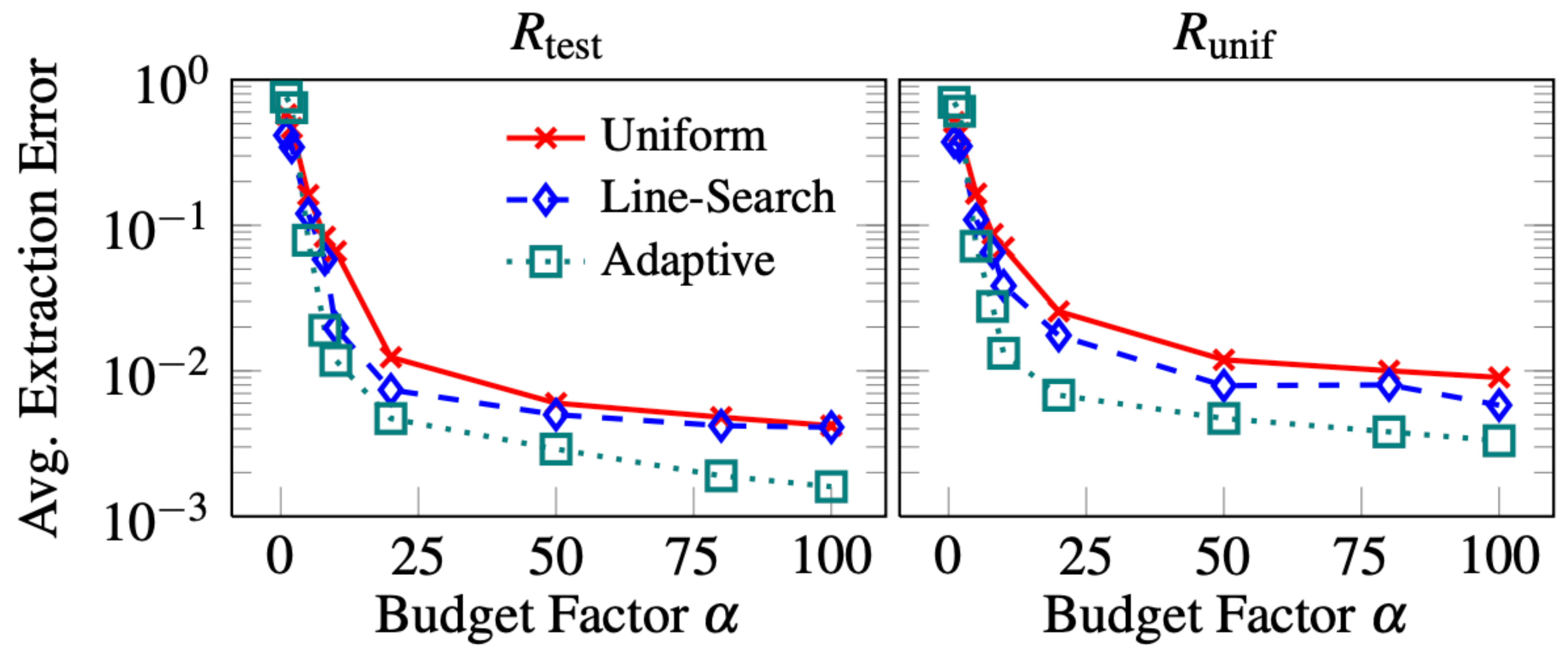


Figure 6: **Average error of extracted RBF kernel SVMs** Results are for three retraining strategies applied to models trained on all binary data sets from Table 3. The left shows R_{test} and the right shows R_{unif}.

Extraction Countermeasures

- **Rounding Confidences**
  - eqn-solving: eliminates guarantees
  - path-finding: increases leaf collision
- **Differential Privacy**
  - Add noise to model weights when querying — prevent distinguishing between close model params
- **Ensemble Methods**
  - Not studied, but may prevent attack

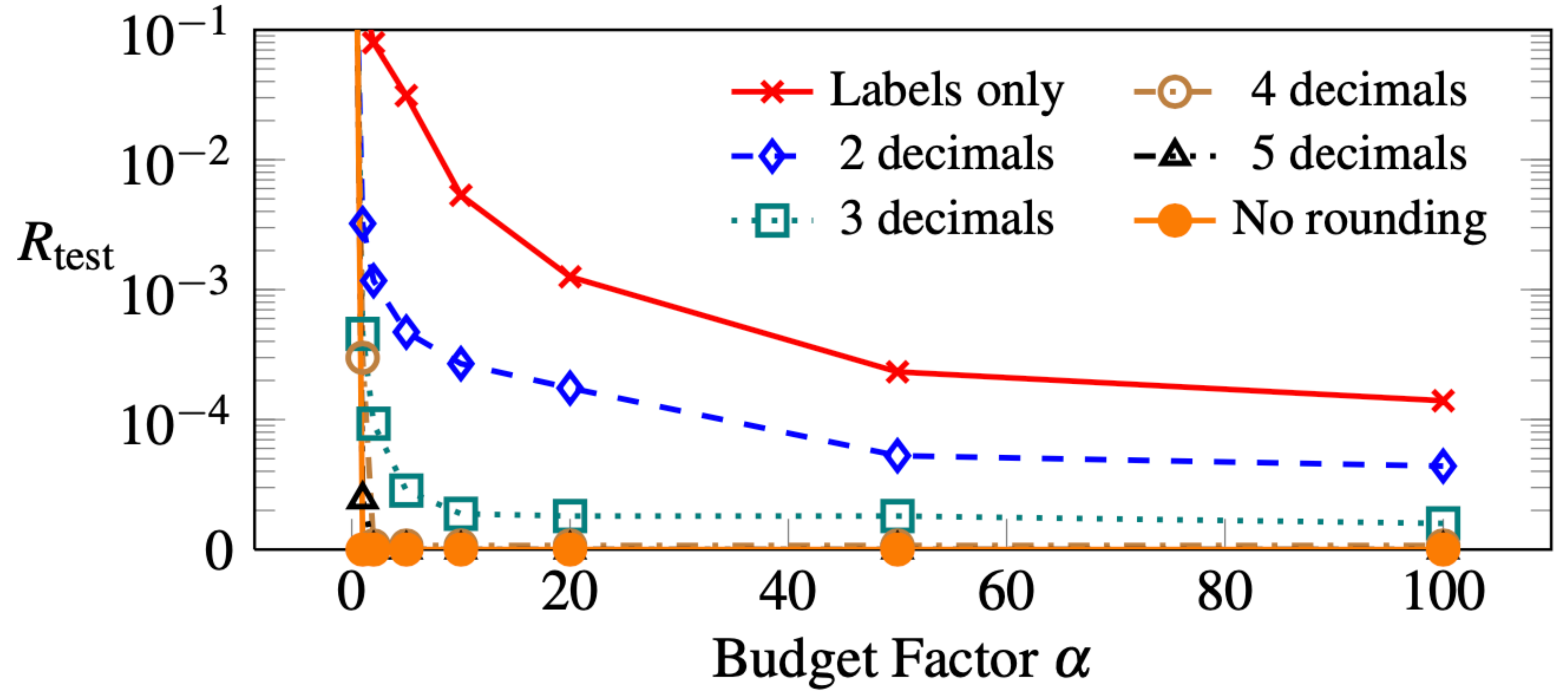


Figure 7: **Effect of rounding on model extraction.** Shows the average test error of equation-solving attacks on softmax models trained on the benchmark suite (Table 3), as we vary the number of significant digits in reported class probabilities. Extraction with no rounding and with class labels only (adaptive retraining) are added for comparison.

Extracting Training Data from Large Language Models

Carlini et al.

Presented by
Apra Gupta

Problem Statement

Demonstration and Empirical Evaluation of Training Data Extraction Attacks on Large Language Models (GPT-2)

Key Contributions:

- Two-step method for black-box (memorized) training data extraction attacks for large language model
- Quantitative Analysis of attack under multiple configurations
- Testable Definition of Memorization
- Study on effect of model size, and string frequency on memorization
- Mitigation Strategy Recommendations

Background on Large Language Models and GPT-2

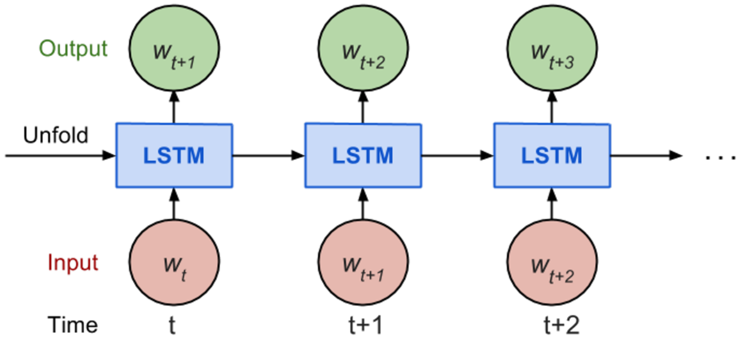
Language Models

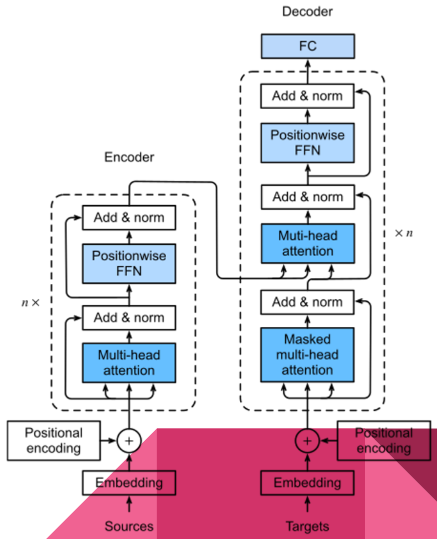
- Given a sequence of length *m,* assign a probability $P(w_1, w_2, \ldots, w_m)$ to the whole sequence: should be able to generate sequences that are coherent in the language they are meant to represent.
- Most State of the Art Neural Language models are trained for the generative task of "next-word-prediction" or "masked word(s) prediction".
  - Given a sequence of words $w_1, w_2, \ldots, w_{n-1}$, outputs a probability for **each token in the vocabulary of being w_n** (Predict the next word in the sequence)
  - Given a sequence of words $w_1, w_2, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n$ outputs a probability for **each token in the vocabulary of being w_i.** (Predict the missing words, usually an objective for bidirectional language models)
  - **Next sentence prediction** is also another objective: given two sentences, the model is simply trained to predict whether sentence 2 follows sentence 1 (BERT)
  - Sometimes trained to predict multiple masked words instead of just one
- By being trained for such tasks, they learn how to interpret a sequence of words in a language and are used in transfer learning scenarios to later fine tune for specific tasks such as QA, Sentiment Analysis, Translation, Summarization etc-

Language Models (cont)

- RNN based
  - Cannot parallelize, processes each part of sequence sequentially (very slow for long sequences)
  - Vanishing gradients, loses information from start of sequence for long sequences
- Attention based (Transformer Models)
  - Uses attention mechanism to process entire sequence at once
  - Encoder - Decoder structure (sequence to sequence tasks)

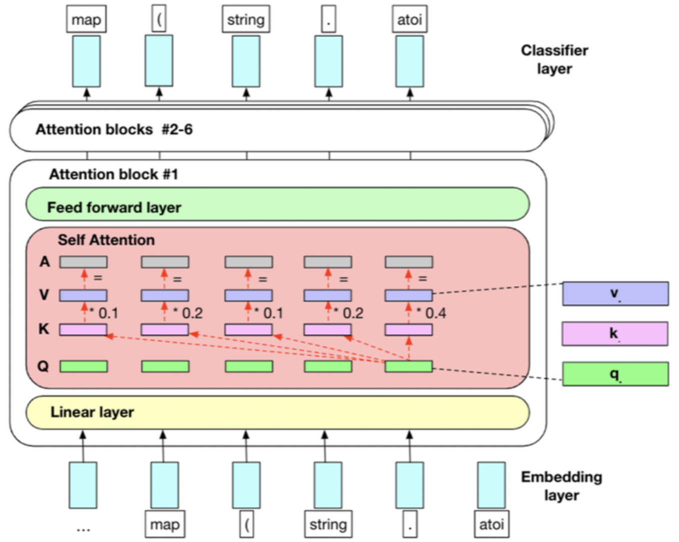
$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$





GPT-2

- Trained for next word prediction (unidirectional)
- Output: probability distribution of next word
- No encoder Layers, only decoders
- Produces one token at a time (autoregressive)
- context size 512-1024 tokens
- 50k vocabulary size
- Trained on 40 GB of text data scraped from the internet: following outbound reddit links
- Data deduplicated at document level



Definitions

Model Knowledge Extraction: A string s is extractable from a Language Model f_θ if there exists a prefix *c* such that:

$$s \leftarrow \underset{s':\ |s'|=N}{\arg\max} f_\theta(s' \mid c)$$

Note:

$f_\theta(s'|c)$ represents likelihood of all sequences of length N. Since computing the most likely sequence is intractable for large N, we replace the argmax with an appropriate sampling strategy

k-Eidetic Memorization: A string *s* is k-eidetic memorized (for k ≥ 1) by a Language Model f_θ if *s* is extractable from f_θ and *s* appears in at most *k* examples in the training data:

$$X : |\{x \in X : s \subseteq x\}| \leq k$$

- Allows us to define memorization as a spectrum
- Smaller values of *k* are more probably harmful/unintended since some memorization is expected for strings with high values of k (the length of *s* may also tell us about the unintentionally of memorization)

Threat Model

Adversarial Objectives: Indiscriminately extract memorized training data from the model (as opposed to aim for targeted pieces of training data).

Adversarial Knowledge: Black Box, no knowledge of training data/architecture/parameters (although authors likely knew of GPT-2 architecture)

Adversarial Capabilities: Query LM for probability distribution of next word in sequence by supplying prior tokens in sequence.

Note: Violation of Contextual Integrity, Data Secrecy

Attack Framework

1. **Text Generation:** Repeatedly sample tokens based on one of the three sampling schemes in an autoregressive fashion to produce sequences that the model considers “high likelihood”.

1. **Predict which of the samples produced in step 1. contain memorized text:** Use **Membership Inference** - which of the samples came directly from the training set? Mainly done by comparing likelihood of sample according to the attacked LM with its likelihood under other conditions. Five comparison techniques discussed.

Attack Configurations

Sampling Schemes

1. **Initialize using start of sequence and proceed with top-n strategy:** set all but top-n token probabilities to 0 and normalize, then sample from that distribution to produce next token (n=40). Lead to multiple repeated sequences.

1. **Sampling with Decaying Temperature (t):** Divide model logits by ‘t’ before applying softmax. Initially decay temperature over first s steps to t=1 (t_{init}=10, s=20). Intended to allow model to explore diverse set of prefixes, before pursuing high confidence path.

1. **Conditioning on Internet Text:** Seed model with random prefixes from “disjoint” set scraped from internet, then proceed with top-n sampling (5-10 context tokens). Meant to seed with prefixes unlikely to be sampled by 1&2 yet *similar* to training data.

Membership Inference Techniques

1. **Low Perplexity by attacked LM:** Predict top-n lowest perplexity samples as memorized.Based on the assumption that the model will assign a high likelihood to sequences seen in its training data.

$$\mathcal{P} = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log f_\theta(x_i|x_1,\ldots,x_{i-1})\right)$$

1. **Compare to Perplexity other language models:** Predict samples with highest ratio of log-perplexities on smaller language models (GPT-2 Small, GPT-2 Medium) compared to the attack model (GPT-2 XL) as memorized. Meant to emulate comparing against models trained on a disjoint set, assumes smaller models memorize less.

2. **Compare to zlib Compression:** Same as above, but compare sample log-perplexity on attack model to number of bits of entropy when sample is compressed with zlib compression. zlib compression emulates a baseline non-neural language model and is good at identifying repeated patterns.

3. **Comparing to Perplexity on Lowercased Sample:** Same as above but compare to log-perplexity of lowercase version of sample on attack model itself. If lowercased version remains low perplexity, sample likely not memorized.

4. **Perplexity on Sliding Window:** 1. but instead of lowest perplexity, use minimum perpecity across any sliding window of 50 tokens. Helps in cases where model confidence reduces when the sample contains one memorized sub-string surrounded by a block of non-memorized (and high perplexity) text.

Experiments: attacks on GPT-2

Method

1. Build 3-datasets of 200,000 256 token generated samples (one for each sampling technique)
2. Order each of the three datasets according to each of 6 membership inference metrics
3. select top-100 (deduplicated) samples according to each metric
4. deduplication through automated fuzzy algorithm based on trigram multiset similarity

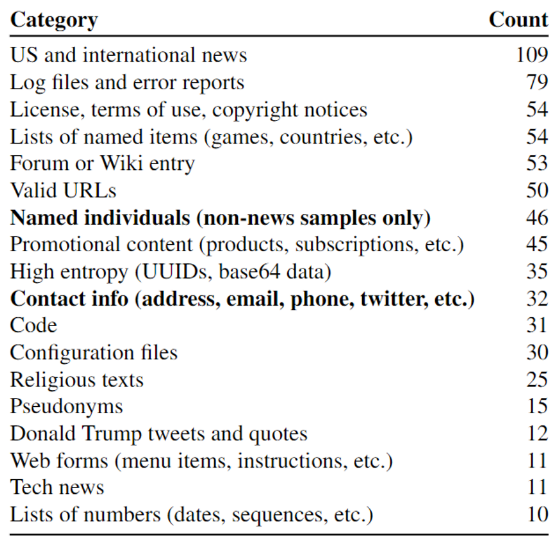
$$|\texttt{tri}(s_1) \cap \texttt{tri}(s_2)| \geq |\texttt{tri}(s_1)|/2.$$

Evaluation

- **Manual inspection**: “we mark a sample as memorized if we can identify a non-trivial substring that returns an exact match on a page found by a Google search.”

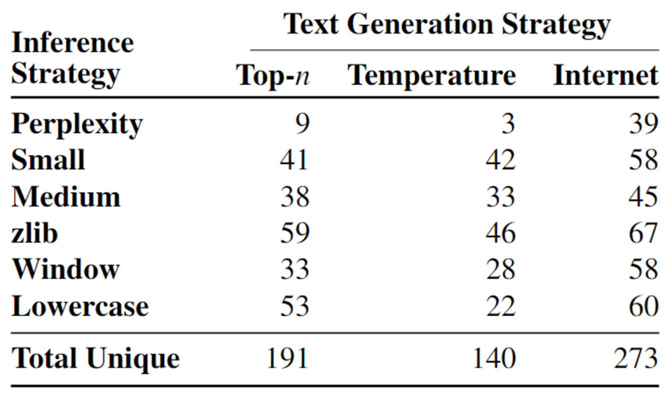
- **Validating on Training Set:** Using limited query access to GPT-2 training data, fuzzy trigram match with training data samples. “We marked samples as memorized memorized if all 3-grams in the memorized sequence occurred in close proximity in the training dataset”. Exact count of occurrence obtained through grep (in some cases only)

Results



Category	**Count**
US and international news	109
Log files and error reports	79
License, terms of use, copyright notices	54
Lists of named items (games, countries, etc.)	54
Forum or Wiki entry	53
Valid URLs	50
Named individuals (non-news samples only)	46
Promotional content (products, subscriptions, etc.)	45
High entropy (UUIDs, base64 data)	35
Contact info (address, email, phone, twitter, etc.)	32
Code	31
Configuration files	30
Religious texts	25
Pseudonyms	15
Donald Trump tweets and quotes	12
Web forms (menu items, instructions, etc.)	11
Tech news	11
Lists of numbers (dates, sequences, etc.)	10

Table 1: Manual categorization of the 604 memorized training examples that we extract from GPT-2, along with a description of each category. Some samples correspond to multiple categories (e.g., a URL may contain base-64 data). Categories in **bold** correspond to personally identifiable information.



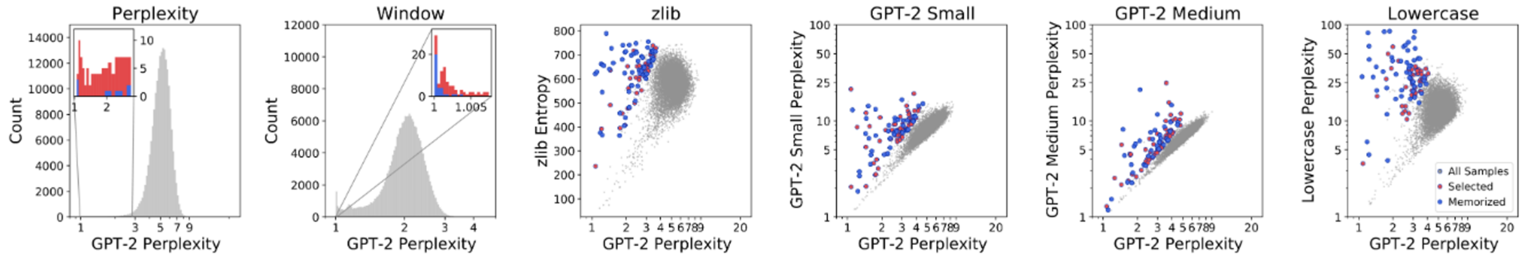
Inference Strategy	**Text Generation Strategy**		
	Top-*n*	**Temperature**	**Internet**
Perplexity	9	3	39
Small	41	42	58
Medium	38	33	45
zlib	59	46	67
Window	33	28	58
Lowercase	53	22	60
Total Unique	191	140	273

Table 2: The number of memorized examples (out of 100 candidates) that we identify using each of the three text generation strategies and six membership inference techniques. Some samples are found by multiple strategies; we identify 604 unique memorized examples in total.

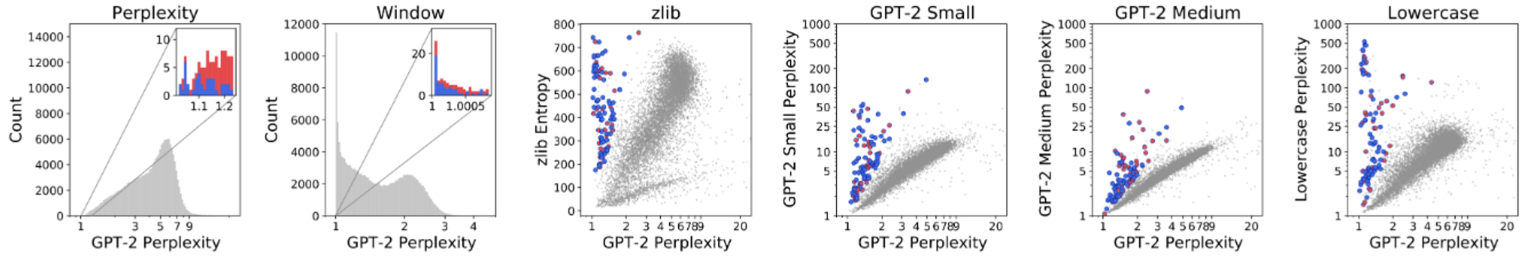
Memorized String	Sequence Length	Occurrences in Data	
		Docs	Total
`Y2...`[redacted]`...y5`	87	1	10
`7C...`[redacted]`...18`	40	1	22
`XM...`[redacted]`...WA`	54	1	36
`ab...`[redacted]`...2c`	64	1	49
`ff...`[redacted]`...af`	32	1	64
`C7...`[redacted]`...ow`	43	1	83
`0x...`[redacted]`...C0`	10	1	96
`76...`[redacted]`...84`	17	1	122
`a7...`[redacted]`...4b`	40	1	311

Table 3: **Examples of $k = 1$ eidetic memorized, high-entropy content that we extract** from the training data. Each is contained in *just one* document. In the best case, we extract a 87-characters-long sequence that is contained in the training dataset just 10 times in total, all in the same document.

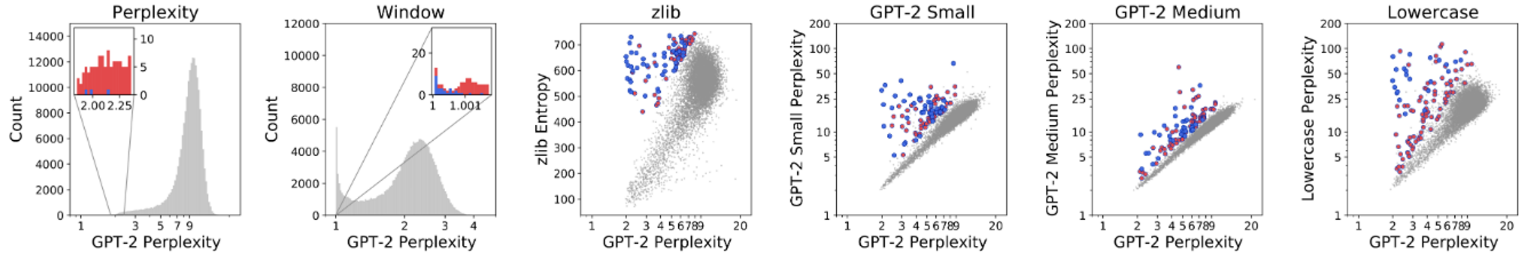
Results: distribution of perplexities



(a) Top-*n* (2.6% duplicates)



(b) Internet (7.1% duplicates)



(c) Temperature (0.6% duplicates)

Experiment: Extracting Long Sequences

“we extend the length of some of the memorized sequences by seeding the model with each sample and continuing to generate. To do this, we apply a beam-search-like decoding method introduced in prior work [8] instead of greedy decoding which often fails to generate long verbatim sequences.”

- identify a piece of source code taken from a repository on GitHub.
- extend this snippet to extract an entire file, namely 1450 lines of verbatim source code.
- extract the entirety of the MIT, Creative Commons, and Project Gutenberg licenses.
- indicates that we could likely extend many memorized strings to much longer snippets of memorized content.

Experiments: Memorization Factors

Study of factors affecting memorization:

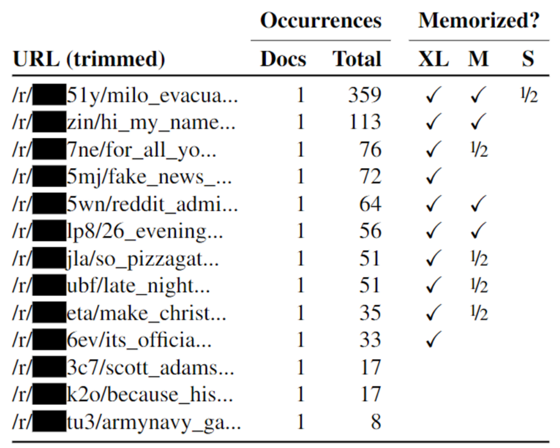
1) How well are *naturally occurring canaries* memorized by GPT-2 based on training dataset occurrence frequency single document)?
2) How does model size affect this memorization (S,M,XL)?

Prefix (taken from single document in pastebin):

```
{"color":"fuchsia","link":"https://www.
reddit.com/r/The_Donald/comments/
```

Results

- top-n sampling used to generate samples (10,1000)
- ✓: full URL exists verbatim in one of the samples blank: not memorized
- ½: memorized with provision of first 6 characters of random token that begins each URL (variation of attack difficulty) + beam search
- blank: URL not found in any samples



	Occurrences		Memorized?		
URL (trimmed)	**Docs**	**Total**	**XL**	**M**	**S**
/r/[redacted]51y/milo_evacua...	1	359	✓	✓	½
/r/[redacted]zin/hi_my_name...	1	113	✓	✓	
/r/[redacted]7ne/for_all_yo...	1	76	✓	½	
/r/[redacted]5mj/fake_news_...	1	72	✓		
/r/[redacted]5wn/reddit_admi...	1	64	✓	✓	
/r/[redacted]lp8/26_evening...	1	56	✓	✓	
/r/[redacted]jla/so_pizzagat...	1	51	✓	½	
/r/[redacted]ubf/late_night...	1	51	✓	½	
/r/[redacted]eta/make_christ...	1	35	✓	½	
/r/[redacted]6ev/its_officia...	1	33	✓		
/r/[redacted]3c7/scott_adams...	1	17			
/r/[redacted]k2o/because_his...	1	17			
/r/[redacted]tu3/armynavy_ga...	1	8			

Table 4: We show snippets of Reddit URLs that appear a varying number of times in a *single* training document. We condition GPT-2 XL, Medium, or Small on a prompt that contains the beginning of a Reddit URL and report a ✓ if the corresponding URL was generated verbatim in the first 10,000 generations. We report a ½ if the URL is generated by providing GPT-2 with the first 6 characters of the URL and then running beam search.

Proposed Mitigation Strategies

1. **Training with Differential Privacy**: slow with accuracy tradeoff
2. **Curating Training Data**: filtering personal information or content with restrictive terms of use
3. **Limiting Impact of Memorization on Downstream Applications:** Preservation of Contextual Privacy
4. **Auditing ML Models for Memorization**

Insights

- **Memorization does not require overfitting :** (average) train and test error are within 10% of each other
- **Larger Models more susceptible to memorization**
- **Memorization can be hard to discover:** highly dependent on sampling seed prefix selection strategies
- **Need for development of mitigation strategies!**

Discussion Points

- Is GPT-2 particularly susceptible to memorization (due to lack of encoder)?
- Extension of attack to models trained for different tasks?
- How can we prevent loss of contextual privacy when fine-tuning for downstream tasks?
- Targeted prefix selection strategies to extract sensitive data
- Differential Privacy guarantees are on a per-user basis: how can we extend this to multiple webpages?