Exploring Connections Between Active Learning and Model Extraction

Varun Chandrasekaran
University of Wisconsin-Madison
chandrasekaran@cs.wisc.edu

Kamalika Chaudhuri
University of California San Diego
kamalika@cs.ucsd.edu

Irene Giacomelli*
ISI Foundation
irene.giacomelli@isi.it

Somesh Jha
University of Wisconsin-Madison
jha@cs.wisc.edu

Songbai Yan
University of California San Diego
yansongbai@eng.ucsd.edu

August 30, 2019

Abstract

Machine learning is being increasingly used by individuals, research institutions, and corporations. This has resulted in the surge of Machine Learning-as-a-Service (MLaaS) - cloud services that provide (a) tools and resources to learn the model, and (b) a user-friendly query interface to access the model. However, such MLaaS systems raise privacy concerns such as *model extraction*. In model extraction attacks, adversaries maliciously exploit the query interface to *steal* the model. More precisely, in a model extraction attack, a good approximation of a sensitive or proprietary model held by the server is extracted (*i.e.* learned) by a dishonest user who interacts with the server only via the query interface. This attack was introduced by Tramèr *et al.* at the 2016 USENIX Security Symposium, where practical attacks for various models were shown. We believe that better understanding the efficacy of model extraction attacks is paramount to designing secure MLaaS systems. To that end, we take the first step by (a) formalizing model extraction and discussing possible defense strategies, and (b) drawing parallels between model extraction

*Part of this work was done while the author was a research assistant at the University of Wisconsin-Madison.

and established area of *active learning*. In particular, we show that recent advancements in the active learning domain can be used to implement powerful model extraction attacks, and investigate possible defense strategies.

1 Introduction

Advancements in various facets of machine learning has made it an integral part of our daily life. However, most real-world machine learning tasks are resource intensive. To that end, several cloud providers, such as Amazon, Google, Microsoft, and BigML offset the storage and computational requirements by providing *Machine Learning-as-a-Service (MLaaS)*. A MLaaS server offers support for both the training phase, and a query interface for accessing the trained model. The trained model is then queried by other users on chosen instances (refer Fig. 1). Often, this is implemented in a pay-per-query regime *i.e.* the server, or the model owner via the server, charges the the users for the queries to the model. Pricing for popular MLaaS APIs is given in Appendix A.1.

Current research is focused at improving the performance of training algorithms and of the query interface, while little emphasis is placed on the related security aspects. For example, in many real-world applications, the trained models are privacy-sensitive - a model can (a) leak sensitive information about training data [6] during/after training, and (b) can itself have commercial value or can be used in security applications that assume its secrecy (*e.g.*, spam filters, fraud detection etc. [35, 46, 62]). To keep the models private, there has been a surge in the practice of *oracle access*, or black-box access. Here, the trained model is made available for prediction but is kept secret. MLaaS systems use oracle access to balance the trade-off between privacy and usability.

Despite providing oracle access, a broad suite of attacks continue to target existing MLaaS systems [1, 13]. For example, membership inference attacks attempt to determine if a given data-point is included in the model's training dataset only by interacting with the MLaaS interface (*e.g.* [61]). In this work, we focus on *model extraction attacks*, where an adversary makes use of the MLaaS query interface in order to *steal* the proprietary model (*i.e.* learn the model or a good approximation of it). In an interesting paper, Tramèr *et al.* [65], show that many commonly used MLaaS interfaces can be exploited using only few queries to recover a model's secret parameters. Even though model extraction attacks are empirically proven to be feasible, their work consider interfaces that reveal auxiliary information, such as confidence values together with the prediction output. Additionally, their work does not formalize model extraction. We believe that such formalization is paramount for designing secure MLaaS that are resilient to aforementioned threats. In this paper, we take the first step in this direction. The main contributions of the paper appear in boldfaced captions.

Model Extraction ≈ Active Learning. The key observation guiding our formalization is that the process of model extraction is very similar to *active learning* [59], a special case of semi-supervised machine learning. An active learner learns an approximation of a labeling function f^* through repetitive interaction with an oracle, who is assumed to know f^*. These interactions typically involve the learner sending an instance x to the oracle, and the oracle returning the label $y = f^*(x)$ to the learner. Since the learner can choose the instances to be labeled, the number of data-points needed to learn the labeling function is often much lower than in the normal supervised case. Similarly, in model extraction, the adversary uses a strategy to query a MLaaS server with the following goals: (a) to successfully steal (*i.e.* learn) the model (*i.e.* labeling function) known by the server (*i.e.* oracle), in such a way as to (b) minimize the number of queries made to the MLaaS server, as each query costs the adversary a fixed dollar value. While the overall process of active learning mirrors the general description of model extraction, the entire spectrum of active learning can not be used to study model extraction. Indeed, some scenarios (*e.g.*, PAC active learning) assume that the query instances are sampled from the actual input distribution. However, an attacker is not restricted to such condition and can query any instance. For this reason, we believe that the *query synthesis* framework of active learning, where the learner has the power to generate arbitrary query instances best replicates the capabilities of the adversary in the model extraction framework. Additionally, the query synthesis scenario ensures that we make no assumptions about the adversary's prior knowledge.

Powerful attacks with no auxiliary information. By casting model extraction as query synthesis active learning, we are able to draw concrete similarities between the two. Consequently, we are able to use algorithms and techniques from the active learning community to perform powerful model extraction attacks, and investigate possible defense strategies. In particular, we show that: query synthesis active learning algorithms can be used to perform model extraction on linear classifiers with *no auxiliary information.* Moreover, our evaluation shows that our attacks are better than the classic attacks, such as by Lowd and Meek [46], which have been widely used in the security community (see Section 4).

No "free lunch" for defense. Simple defense strategies such as changing the prediction output with constant and small probability are not effective. However, defense strategies that change the prediction output depending on the instances that are being queried, such as the work of Alabdulmohsin *et al.* [3], are more robust to extraction attacks implemented using existing query synthesis active learning algorithms. However, in Algorithm 1 of Section 4, we show that this defense is not secure against traditional passive learning algorithms. This suggests that there is "no free lunch" – accuracy might have to be sacrificed to prevent model extraction. An in-depth investigation of such a result will be interesting avenue for future work.

Paper structure. We begin with a brief comparison between passive machine learning and active learning in Section 2. This allows us to introduce the notation used in this paper, and review the state-of-the-art for active learning. Section 3 focuses on the formalization of model extraction attacks, casting it into the query synthesis active learning framework, and finally discusses possible defenses strategies. Section 4 reports our experimental findings and demonstrates that query synthesis active learning can be used to successfully perform model

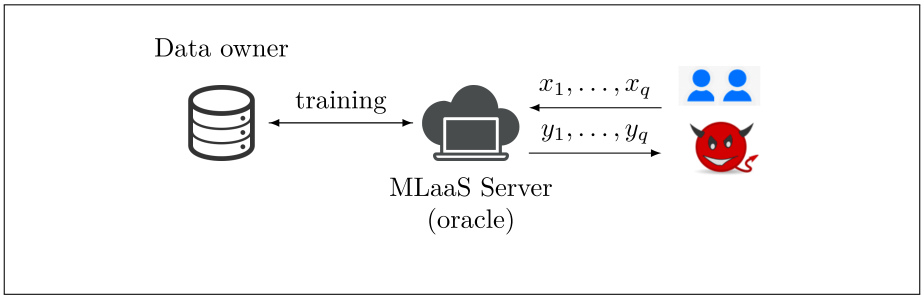


Figure 1: Model extraction can be envisioned as active learning. A data owner, with the help of a MLaaS server, trains a model f^* on its data. The proprietary model is stored by the server, which also answers to queries from users (*i.e.*, $y_i = f^*(x_i)$). In a model extraction attack, a dishonest user tries to exploit this interface to "steal" f^* in the same way as a learner uses answer from a machine-learning oracle in order to learn f^*.

extraction, and evaluates different defense strategies. Specifically, we observe that $0.09 worth Amazon queries are needed to extract most halfspaces when the MLaaS server does not deploy any defense, and $3.65 worth of queries are required to learn a halfspace when it uses data-independent randomization. Finally, we discuss some open issues in Section 5, which provides avenue for future work. Related work is discussed in Section 6, and we end the paper with some concluding remarks.

2 Machine Learning

In this section, we give a brief overview of machine learning, and terminology we use throughout the paper. In particular, we summarize the passive learning framework in subsection 2.1, and focus on active learning algorithms in subsection 2.2. A review of the state-of-the-art of active learning algorithms is needed to explicitly link model extraction to active learning and is presented in Section 3.

2.1 Passive learning

In the standard, passive machine learning setting, the learner has access to a large labeled dataset and uses it in its entirety to learn a predictive model from a given class. Let $\mathbf{X}$ be an instance space, and $\mathbf{Y}$ be a set of labels. For example, in object recognition, $\mathbf{X}$ can be the space of all images, and $\mathbf{Y}$ can be a set of objects that we wish to detect in these images. We refer to a pair $(x, y) \in \mathbf{X} \times \mathbf{Y}$ as a *data-point or labeled instance* (x is the instance, y is the label). Finally, there is a class of functions $\mathcal{F}$ from $\mathbf{X}$ to $\mathbf{Y}$ called the *hypothesis space* that is known in advance. The learner's goal is to find a function $\hat{f} \in \mathcal{F}$ that is a good predictor for the label y given the instance x, with $(x, y) \in \mathbf{X} \times \mathbf{Y}$. To measure how well $\hat{f}$ predicts the labels, a loss function ℓ is used. Given a data-point $z = (x, y) \in \mathbf{X} \times \mathbf{Y}$, $\ell(\hat{f}, z)$ measures the "difference" between $\hat{f}(x)$ and the true label y. When the label domain $\mathbf{Y}$ is finite (classification problem), the 0-1 loss function is frequently used:

$$\ell(\hat{f}, z) = \begin{cases} 0, & \text{if } \hat{f}(x) = y \\ 1, & \text{otherwise} \end{cases}$$

If the label domain $\mathbf{Y}$ is continuous, one can use the square loss: $\ell(\hat{f}, z) = (\hat{f}(x) - y)^2$.

In the passive setting, the PAC (*probably approximately correct*) learning [66] framework is predominantly used. Here, we assume that there is an underlying distribution $\mathcal{D}$ on $\mathbf{X} \times \mathbf{Y}$ that describes the data; the learner has no direct knowledge of $\mathcal{D}$ but has access to a set of training data D drawn from it. The main goal in passive PAC learning is to use the labeled instances from D to produce a hypothesis $\hat{f}$ such that its expected loss with respect to the probability distribution $\mathcal{D}$ is low. This is often measured through the *generalization error* of the hypothesis $\hat{f}$, defined by

$$\mathrm{Err}_{\mathcal{D}}(\hat{f}) = \mathbb{E}_{z \sim \mathcal{D}}[\ell(\hat{f}, z)] \tag{1}$$

More precisely, we have the following definition.

Definition 1 (PAC passive learning [66])**.** An algorithm A is a PAC passive learning algorithm for the hypothesis class $\mathcal{F}$ if the following holds for any $\mathcal{D}$ on $\mathbf{X} \times \mathbf{Y}$ and any $\varepsilon, \delta \in (0, 1)$: If A is given $s_A(\varepsilon, \delta)$ i.i.d. data-points generated by $\mathcal{D}$, then A outputs $\hat{f} \in \mathcal{F}$ such that $\mathrm{Err}_{\mathcal{D}}(\hat{f}) \le \min_{f \in \mathcal{F}} \mathrm{Err}_{\mathcal{D}}(f) + \varepsilon$ with probability at least $1 - \delta$. We refer to $s_A(\varepsilon, \delta)$ as the *sample complexity* of algorithm A.

Remark 1 (Realizability assumption). In the general case, the labels are given together with the instances, and the factor $\min_{f \in \mathcal{F}} \mathrm{Err}_{\mathcal{D}}(f)$ depends on the hypothesis class. Machine learning literature refers to this as *agnostic learning* or the non-separable case of PAC learning. However, in some applications, the labels themselves can be described using a labeling function $f^* \in \mathcal{F}$. In this case (known as *realizable learning*), $\min_{f \in \mathcal{F}} \mathrm{Err}_{\mathcal{D}}(f) = 0$ and the distribution $\mathcal{D}$ can be described by its marginal over $\mathbf{X}$. A PAC passive learning algorithm A in the realizable case takes $s_A(\varepsilon, \delta)$ i.i.d. instances generated by $\mathcal{D}$ and the corresponding labels generated using f^*, and outputs $\hat{f} \in \mathcal{F}$ such that $\mathrm{Err}_{\mathcal{D}}(\hat{f}) \le \varepsilon$ with probability at least $1 - \delta$.

2.2 Active learning

In the passive setting, learning an accurate model (*i.e.* learning $\hat{f}$ with low generalization error) requires a large number of data-points. Thus, the labeling effort required to produce an accurate predictive model may be prohibitive. In other words, the sample complexity of many learning algorithms grows rapidly as $\varepsilon \to 0$ (refer Example 1). This has spurred interest in learning algorithms that can operate on a smaller set of labeled instances, leading to the emergence of *active learning*. In active learning, the learning algorithm is allowed to select a subset of unlabeled instances, query their corresponding labels from an annotator (a.k.a oracle) and then use it to construct or update a model. How the algorithm chooses the instances varies widely. However, the common underlying idea is that by actively choosing the data-points used for training, the learning algorithm can drastically reduce sample complexity.

Formally, an active learning algorithm is an interactive process between two parties - the oracle $\mathcal{O}$ and the learner $\mathcal{L}$. The only interaction allowed is through *queries* - $\mathcal{L}$ chooses $x \in \mathbf{X}$ and sends it to $\mathcal{O}$, who responds with $y \in \mathbf{Y}$ (*i.e.*, the oracle returns the label for the chosen unlabeled instance). This value of (x, y) is then used by $\mathcal{L}$ to infer some information about the labeling procedure, and to choose the next instance to query. Over many such interactions, $\mathcal{L}$ outputs $\hat{f}$ as a predictor for labels. We can use the generalization error (1) to evaluate the accuracy of the output $\hat{f}$. However, depending on the query strategy chosen by $\mathcal{L}$, other types of error can be used.

There are two distinct scenarios for active learning: *PAC active learning* and *Query Synthesis (QS) active learning.* In literature, QS active learning is also known as *Membership Query Learning*, and we will use the two terms synonymously.

2.2.1 PAC active learning

This scenario was introduced by Dasgupta in 2005 [21] in the realizable context and then subsequently developed in following works (*e.g.*, [5, 20, 32]). In this scenario, the instances are sampled according to the marginal of $\mathcal{D}$ over $\mathbf{X}$, and the learner, after seeing them, decides whether to query for their labels or not. Since the data-points seen by $\mathcal{L}$ come from the actual underlying distribution $\mathcal{D}$, the accuracy of the output hypothesis $\hat{f}$ is measured using the generalization error (1), as in the classic (*i.e.*, passive) PAC learning.

There are two options to consider for sampling data-points. In *stream-based sampling* (also called *selective sampling*) , the instances are sampled one at a time, and the learner decides whether to query for the label or not on a per-instance basis. *Pool-based sampling* assumes that all of the instances are collected in a static pool $S \subseteq \mathbf{X}$ and then the learner chooses specific instances in S and queries for their labels. Typically, instances are chosen by $\mathcal{L}$ in a greedy fashion using a metric to evaluate all instances in the pool. This is not possible in stream-based sampling, where $\mathcal{L}$ goes through the data sequentially, and has to therefore make decisions to query individually. Pool-based sampling is extensively studied since it has applications in many real-world problems, such as text classification, information extraction, image classification and retrieval, etc. [47]. Stream-based sampling represents scenarios where obtaining unlabeled data-points is easy and cheap, but obtaining their labels is expensive (*e.g.*, stream of data is collected by a sensor, but the labeling needs to be performed by an expert).

Before describing query synthesis active learning, we wish to highlight the advantage of PAC active learning over passive PAC learning (*i.e.* the reduced sample complexity) for some hypothesis class through Example 1. Recall that this advantage comes from the fact that an active learner is allowed to adaptively

$$f_w(x) = \begin{cases} -1 \text{ if } \langle w, x \rangle < -1 \\ +1 \text{ otherwise} \end{cases}$$

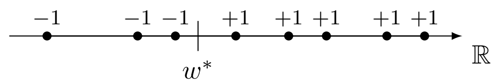


Figure 2: Halfspace classification in dimension 1.

choose the data from which it learns, while a passive learning algorithm learns from a static set of data-points.

Example 1 (PAC learning for halfspaces). Let $\mathcal{F}_{d,HS}$ be the hypothesis class of *d-dimensional halfspaces*, used for binary classification. A function in $f_w \in \mathcal{F}_{d,HS}$ is described by a normal vector $w \in \mathbb{R}^d$ (*i.e.*, $||w||_2 = 1$) and is defined by

$$f_w(x) = \text{sign}(\langle w, x \rangle) \text{ for any } x \in \mathbb{R}^d$$

where given two vectors $a, b \in \mathbb{R}^d$, then their product is defined as $\langle a, b \rangle = \sum_{i=1}^{d} a_i b_i$. Moreover, if $x \in \mathbb{R}$, then $\text{sign}(x) = 1$ if $x \geq 0$ and $\text{sign}(x) = -1$ otherwise. A classic result in passive PAC learning states that $O(\frac{d}{\varepsilon}\log(\frac{1}{\varepsilon}) + \frac{1}{\varepsilon}\log(\frac{1}{\delta}))$ data-points are needed to learn f_w [66]. On the other hand, several works propose active learning algorithms for $\mathcal{F}_{d,HS}$ with sample complexity[1] $\tilde{O}(d\log(\frac{1}{\varepsilon}))$ (under certain distributional assumptions). For example, if the underlying distribution is log-concave, there exists an active learning algorithm with sample complexity $\tilde{O}(d\log(\frac{1}{\varepsilon}))$ [10, 11, 74]. This general reduction in the sample complexity for $\mathcal{F}_{d,HS}$ is easy to infer when $d = 1$. In this case, the data-points lie on the real line and their labels are a sequence of -1's followed by a sequence of $+1$'s. The goal is to discover a point w where the change from -1 to $+1$ happens. PAC learning theory states that this can be achieved with $\tilde{O}(\frac{1}{\varepsilon})$[2] points i.i.d. sampled from $\mathcal{D}$. On the other hand, an active learning algorithm that uses a simple binary search can achieve the same task with $O(\log(\frac{1}{\varepsilon}))$ queries [21] (refer Figure 2).

2.2.2 Query Synthesis (QS) active learning

In this scenario, the learner can request labels for any instance in the input space $\mathbf{X}$, including points that the learner generates *de novo*, independent of the distribution $\mathcal{D}$ (*e.g.*, $\mathcal{L}$ can ask for labels for those x that have zero-probability of being sampled according to $\mathcal{D}$). Query synthesis is reasonable for many problems, but labeling such arbitrary instances can be awkward if the oracle is a human annotator. Thus, this scenario better represents real-world applications where the oracle is automated (*e.g.*, results from synthetic experiments [38]). Since the data-points are independent of the distribution, generalization error is

[1] The $\tilde{O}$ notation ignores logarithmic factors and terms dependent on δ.
[2] More generally, $\tilde{O}(\frac{d}{\varepsilon})$ points.

not an appropriate measure of accuracy of the hypothesis $\hat{f}$, and other types of error are typically used. These new error formulations depend on the concrete hypothesis class $\mathcal{F}$ considered. For example, if $\mathcal{F}$ is the class of boolean functions from $\{0,1\}^n$ to $\{0,1\}$, then the *uniform error* is used. Assume that the oracle $\mathcal{O}$ knows $f^* \in \mathcal{F}$ and uses it as labeling function (realizable case), then the uniform error of the hypothesis $\hat{f}$ is defined as

$$\mathrm{Err}_u(\hat{f}) = \Pr_{x \sim \{0,1\}^n}[\hat{f}(x) \neq f^*(x)]$$

where x is sampled uniformly at random from the instance space $\{0,1\}^n$. Recent work [4,16], for the class of halfspaces $\mathcal{F}_{d,HS}$ (refer to Example 1) use *geometric error*. Assume that the true labeling function used by the oracle is f_{w^*}, then the geometric error of the hypothesis $f_w \in \mathcal{F}_{d,HS}$ is defined as

$$\mathrm{Err}_2(f_w) = ||w^* - w||_2$$

where $||\cdot||_2$ is the 2-norm.

In both active learning scenarios (PAC and QS), the learner needs to evaluate the "usefulness" of an unlabeled instance x, which can either be generated de novo or sampled from the given distribution, in order to decide whether to query the oracle for the corresponding label. In the state of the art, we can find many ways of formulating such *query strategies*. Most of existing literature presents strategies where efficient search through the hypothesis space is the goal (refer the survey by Settles [59]). Another point of consideration for an active learner $\mathcal{L}$ is to decide when to stop. This is essential as active learning is geared at improving accuracy while being sensitive to new data acquisition cost (*i.e.*, reducing the query complexity). While one school of thought relies on the *stopping criteria* based on the intrinsic measure of stability or self-confidence within the learner, another believes that it is based on economic or other external factors (refer [59, Section 6.7]).

Given this large variety within active learning, we enhance the standard definition of a learning algorithm and propose the definition of an active learning system, which is geared towards model extraction. Our definition is informed by the MLaaS APIs that we investigated (more details are present in Appendix A.1).

Definition 2 (Active learning system)**.** Let $\mathcal{F}$ be a hypothesis class with instance space $\mathbf{X}$ and label space $\mathbf{Y}$. An active learning system for $\mathcal{F}$ is given by two entities, the learner $\mathcal{L}$ and the oracle $\mathcal{O}$, interacting via membership queries: $\mathcal{L}$ sends to $\mathcal{O}$ an instance $x \in \mathbf{X}$; $\mathcal{O}$ answers with a label $y \in \mathbf{Y}$. We indicate via the notation $\mathcal{O}_{f^*}$ the realizable case where $\mathcal{O}$ uses a specific labeling function $f^* \in \mathcal{F}$, *i.e.* $y = f^*(x)$. The behavior of $\mathcal{L}$ is described by the following parameters:

1. *Scenario*: this is the rule that describes the generation of the input for the querying process (*i.e.* which instances $x \in \mathbf{X}$ can be queried). In the

PAC scenario, the instances are sampled from the underlying distribution $\mathcal{D}$. In the query synthesis (QS) scenario, the instances are generated by the learner $\mathcal{L}$;

2. *Query strategy*: given a specific scenario, the query strategy is the algorithm that adaptively decides if the label for a given instance x_i is queried for, given that the queries $x_1, \ldots, x_{i-1}$ have been answered already. In the query synthesis scenario, the query strategy also describes the procedure for instance generation.

3. *Stopping criteria*: this is a set of considerations used by $\mathcal{L}$ to decide when it must stop asking queries.

Any system $(\mathcal{L}, \mathcal{O})$ described as above is an active learning system for $\mathcal{F}$ if one of the following holds:

- *(PAC scenario)* For any $\mathcal{D}$ on $\mathbf{X} \times \mathbf{Y}$ and any $\varepsilon, \delta \in (0, 1)$, if $\mathcal{L}$ is allowed to interact with $\mathcal{O}$ using $q_{\mathcal{L}}(\varepsilon, \delta)$ queries, then $\mathcal{L}$ outputs $\hat{f} \in \mathcal{F}$ such that $\mathrm{Err}_{\mathcal{D}}(\hat{f}) \leq \min_{f \in \mathcal{F}} \mathrm{Err}_{\mathcal{D}}(f) + \varepsilon$ with probability at least $1 - \delta$.

- *(QS scenario)* Fix an error measure Err for the functions in $\mathcal{F}$. For any $f^* \in \mathcal{F}$, if $\mathcal{L}$ is allowed to interact with $\mathcal{O}_{f^*}$ using $q_{\mathcal{L}}(\varepsilon, \delta)$ queries, then $\mathcal{L}$ outputs $\hat{f} \in \mathcal{F}$ such that $\mathrm{Err}(\hat{f}) \leq \varepsilon$ with probability at least $1 - \delta$.

We refer to $q_{\mathcal{L}}(\varepsilon, \delta)$ as the *query complexity* of $\mathcal{L}$.

As we will show in the following section (in particular, refer subsection 3.2), the query synthesis scenario is more appropriate in casting model extraction attack as active learning. Note that, other types queries have been studied in literature. This includes the *equivalence query* [5]. Here the learner can verify if a hypothesis is correct or not. We do not consider equivalence queries in our definition because we did not see any of the MLaaS APIs support them.

3 Model Extraction

In subsection 3.1, we begin by formalizing the process of model extraction. We then draw parallels between model extraction and active learning in subsection 3.2. We finally discuss possible defense strategies based on noisy answers in subsection 3.3.

3.1 Model Extraction Definition

We begin by describing the operational ecosystem of model extraction attacks in the context of MLaaS systems. An entity learns a private model f^* from a public class $\mathcal{F}$, and provides it to the MLaaS server. The server provides a client-facing query interface for accessing the model for prediction. For example, in the case of logistic regression, the MLaaS server knows a model represented by parameters $a_0, a_1, \cdots, a_d$. The client issues queries of the form $x = (x[1], \cdots, x[d]) \in \mathbb{R}^d$, and the MLaaS server responds with 0 if $(1 + e^{-a(x)})^{-1} \leq 0.5$ and 1 otherwise, with $a(x) = a_0 + \sum_{i=1}^{d} a_i x[i]$.

Model extraction is the process where an adversary exploits this interface to learn more about the proprietary model f^*. The adversary can be interested in defrauding the description of the model f^* itself (*i.e.*, stealing the parameters a_i as in a reverse engineering attack), or in obtaining an approximation of the model, say $\hat{f} \in \mathcal{F}$, that he can then use for free for the same task as the original f^* was intended for. To capture the different goals of an adversary, we say that the attack is successful if the extracted model is "close enough" to f^* according to an *error function* on $\mathcal{F}$ that is context dependent. Since many existing MLaaS providers operate in a pay-per-query regime, we use query complexity as a measure of efficiency of such model extraction attacks.

Formally, consider the following experiment: an adversary $\mathcal{A}$, who knows the hypothesis class $\mathcal{F}$, has oracle access to a proprietary model f^* from $\mathcal{F}$. This can be thought of as $\mathcal{A}$ interacting with a server S that safely stores f^*. The interaction has several rounds. In each round, $\mathcal{A}$ chooses an instance x and sends it to S. The latter responds with $f^*(x)$. After a few rounds, $\mathcal{A}$ outputs a function $\hat{f}$ that is the adversary's candidate approximation of f^*; the experiment considers $\hat{f}$ a good approximation if its error with respect to the true function f^* held by the server is less then a fixed threshold ε. The error function Err is defined a priori and fixed for the extraction experiment on the hypothesis class $\mathcal{F}$.

Experiment 1 (Extraction experiment)**.** Given a hypothesis class $\mathcal{F} = \{f : \mathbf{X} \to \mathbf{Y}\}$, fix an error function $\mathrm{Err} : \mathcal{F} \to \mathbb{R}$. Let S be a MLaaS server with the knowledge of a specific $f^* \in \mathcal{F}$, denoted by $S(f^*)$. Let $\mathcal{A}$ be an adversary interacting with S with a maximum budget of q queries. The extraction experiment $\mathtt{Exp}^{\varepsilon}_{\mathcal{F}}(S(f^*), \mathcal{A}, q)$ proceeds as follows

1. $\mathcal{A}$ is given a description of $\mathcal{F}$ and oracle access to f^* through the query interface of S. That is, if $\mathcal{A}$ sends $x \in \mathbf{X}$ to S, it gets back $y = f^*(x)$. After at most q queries, $\mathcal{A}$ eventually outputs $\hat{f}$;

2. The output of the experiment is 1 if $\mathrm{Err}(\hat{f}) \leq \varepsilon$. Otherwise the output is 0.

Informally, an adversary $\mathcal{A}$ is successful if with high probability the output of the extraction experiment is 1 for a small value of ε and a fixed query budget q. This means that $\mathcal{A}$ likely learns a good approximation of f^* by only asking q queries to the server. More precisely, we have the following definition.

Definition 3 (Extraction attack)**.** Let $\mathcal{F}$ be a public hypothesis class and S an MLaaS server as explained before. We say that an adversary $\mathcal{A}$, which interacts with S, implements an *ε-extraction attack* of complexity q and confidence γ against the class $\mathcal{F}$ if

$$\Pr[\mathtt{Exp}^{\varepsilon}_{\mathcal{F}}(S(f^*), \mathcal{A}, q) = 1] \geq \gamma$$

for any $f^* \in \mathcal{F}$. The probability is over the randomness of $\mathcal{A}$.

In other words, in Definition 3 the success probability of an adversary constrained by a fixed budget for queries is explicitly lower bounded by the quantity γ.

Before discussing the connection between model extraction and active learning, we provide an example of a hypothesis class that is easy to extract.

Example 2 (Equation-solving attack for linear regression). Let $\mathcal{F}_{d,R}$ be the hypothesis class of regression models from $\mathbb{R}^d$ to $\mathbb{R}$. A function f_a in this class is described by $d+1$ parameters $a_0, a_1, \ldots, a_d$ from $\mathbb{R}$ and defined by: for any $x \in \mathbb{R}^d$,

$$f_a(x) = a_0 + \sum_{i=1}^{d} a_i x_i \,.$$

Consider the adversary $\mathcal{A}_{ES}$ that queries $x^1, \ldots, x^{d+1}$ ($d+1$ instances from $\mathbb{R}^d$) chosen in such a way that the set of vectors $\{(1, x^i)\}_{i=1,\ldots,d+1}$ is linearly independent in $\mathbb{R}^{d+1}$. $\mathcal{A}_{ES}$ receives the corresponding $d+1$ labels, $y_1, \ldots, y_{d+1}$, and can therefore solve the linear system given by the equations $f_a(x^i) = y_i$. Assume that f_{a^*} is the function known by the MLaaS server (*i.e.*, $y_i = f_{a^*}(x^i)$). It is easy to see that if we fix $\mathrm{Err}(f_a) = ||a^* - a||_1$, then $\Pr[\mathtt{Exp}^0_{\mathcal{F}_{d,R}}(S(f_{a^*}), \mathcal{A}_{ES}, d+1) = 1] = 1$. That is, $\mathcal{A}_{ES}$ implements 0-extraction of complexity $d+1$ and confidence 1.

While our model operates in the black-box setting, we discuss other attack models in more detail in Remark 2

3.2 Active Learning and Extraction

From the description presented in the Section 2, it is clear that model extraction in the MLaaS system context closely resembles active learning. The survey of active learning in subsection 2.2 contains a variety of algorithms and scenarios which can be used to implement model extraction attacks (or to study its impossibility). However, not all possible scenarios of active learning are interesting for model extraction. We notice that in the case of model extraction, an adversary $\mathcal{A}$ has no knowledge of the data distribution $\mathcal{D}$. Additionally, such an adversary is not restricted to only considering instances $x \sim \mathcal{D}$ to query. For this reason, we believe that query synthesis (QS) is the right active learning scenario to investigate in order to draw a meaningful parallelism with model extraction. Recall that the query synthesis is the only framework where the query inputs can be generated de novo (*i.e.*, they do not conform to a distribution).

Observation 1: Given a hypothesis class $\mathcal{F}$ and an error function Err, let $(\mathcal{L}, \mathcal{O})$ be an active learning system for $\mathcal{F}$ in the QS scenario (Definition 2). If the query complexity of $\mathcal{L}$ is $q_{\mathcal{L}}(\varepsilon, \delta)$, then there exists and adversary $\mathcal{A}$ that implements ε-extraction with complexity $q_{\mathcal{L}}(\varepsilon, \delta)$ and confidence $1-\delta$ against the class $\mathcal{F}$.

The reasoning for this observation is as follows: Consider the adversary $\mathcal{A}$ that is the learner $\mathcal{L}$ (*i.e.*, $\mathcal{A}$ deploys the query strategy procedure and the stopping criteria that describe $\mathcal{L}$). This is possible because $(\mathcal{L}, \mathcal{O})$ is in the QS scenario and $\mathcal{L}$ is independent of any underlying (unknown) distribution. Let

$q = q_{\mathcal{L}}(\varepsilon, \delta)$ and observe that

$$\begin{aligned}&\Pr[\mathtt{Exp}^{\varepsilon}_{\mathcal{F}}(S(f^*), \mathcal{A}, q) = 1] = \\ &\Pr[\mathcal{A} \text{ outputs } \hat{f} \text{ and } \mathrm{Err}(\hat{f}) \le \varepsilon] = \\ &\Pr[\mathcal{L} \text{ outputs } \hat{f} \text{ and } \mathrm{Err}(\hat{f}) \le \varepsilon] \ge 1 - \delta\end{aligned}$$

Our observation states that any active learning algorithm in the QS scenario can be used to implement a model extraction attack. Therefore, in order to study the security of a given hypothesis class in the MLaaS framework, we can use known techniques and results from the active learning literature. Two examples of this follow.

Example 3 (Decision tree extraction via QS active learning). Let $\mathcal{F}_{n,BF}$ denote the set of boolean functions with domain $\{0,1\}^n$ and range $\{-1,1\}$. The reader can think of -1 as 0 and $+1$ as 1. Using the range of $\{-1,+1\}$ is very common in the literature on learning boolean functions. An interesting subset of $\mathcal{F}_{n,BF}$ is given by the functions that can be represented as a boolean decision tree. A *boolean decision tree* T is a labeled binary tree, where each node v of the tree is labeled by $L_v \subseteq \{1, \cdots, n\}$ and has two outgoing edges. Every leaf in this tree is labeled either $+1$ or -1. Given an n-bit string $x = (b_1, \cdots, b_n), b_i \in \{0,1\}$ as input, the decision tree defines the following computation: the computation starts at the root of the tree T. When the computation arrives at an internal node v we calculate the parity of $\sum_{i \in L_v} b_i$ and go left if the parity is 0 and go right otherwise. The value of the leaf that the computation ends up in is the value of the function. We denote by $\mathcal{F}^m_{n,BT}$ the class of boolean decision trees with n-bit input and m nodes. Kushilevitz and Mansour [43] present an active learning algorithm for the class $\mathcal{F}_{n,BF}$ that works in the QS scenario. This algorithm utilizes the uniform error to determine the stopping condition (refer subsection 2.2). The authors claim that this algorithm has practical efficiency when restricted to the classes $\mathcal{F}^m_{n,BT} \subset \mathcal{F}_{n,BF}$ for any m. In particular, if the active learner $\mathcal{L}$ of [43] interacts with the oracle $\mathcal{O}_{T^*}$ where $T^* \in \mathcal{F}^m_{n,BT}$, then $\mathcal{L}$ learns $g \in \mathcal{F}_{n,BF}$ such that $\Pr_{x \sim \{0,1\}^n}[g(x) \neq T^*(x)] \le \varepsilon$ with probability at least $1 - \delta$ using a number of queries polynomial in n, m, $\frac{1}{\varepsilon}$ and $\log(\frac{1}{\delta})$. Based on Observation 1, this directly translates to the existence of an adversary that implements ε-extraction with complexity polynomial in n, m, $\frac{1}{\varepsilon}$ and confidence $1 - \delta$ against the class $\mathcal{F}^m_{n,BT}$.

Moreover, the algorithm of [43] can be extended to (a) boolean functions of the form $f : \{0, 1, \ldots, k-1\}^n \to \{-1, +1\}$ that can be computed by a polynomial-size k-ary decision tree[3], and (b) regression trees (*i.e.*, the output is a real value from $[0, M]$). In the second case, the running time of the learning algorithm is polynomial in M (refer Section 6 of [43]). Note that the attack model considered here is a stronger model than that considered by [65] because the attacker/learner does not get any information about the internal path of the decision tree (refer Remark 2).

[3] A k-ary decision tree is a tree in which each inner node v has k outgoing edges.

Example 4 (Halfspace extraction via QS active learning). Let $\mathcal{F}_{d,HS}$ be the hypotheses class of d-dimensional halfspaces defined in Example 1. Alabdulmohsin *et al.* [4] present a spectral algorithm to learn a halfspace in the QS scenario that, in practice, outperformed earlier active learning strategies in the PAC scenario. They demonstrate, through several experiments that their algorithm learns $f_w \in \mathcal{F}_{d,HS}$ such that $\|w - w^*\|_2 \leq \varepsilon$ with approximately $2d \log(\frac{1}{\varepsilon})$ queries, where $f_{w^*} \in \mathcal{F}_{d,HS}$ is the labeling function used by $\mathcal{O}$. It follows from Observation 1 that an adversary utilizing this algorithm implements ε-extraction against the class $\mathcal{F}_{d,HS}$ with complexity $\mathcal{O}(d \log(\frac{1}{\varepsilon}))$ and confidence 1. We validate the practical efficacy of this attack in Section 4.

Remark 2 (Extraction with auxiliary information). Observe that we define model extraction for only those MLaaS servers that return *only the label value* y for a *well-formed query* x (*i.e.* in the oracle access setting). A weaker model (*i.e.*, one where attacks are easier) considers the case of MLaaS servers responding to a user's query x even when x is incomplete (*i.e.* with missing features), and returning the label y along with some auxiliary information. The work of Tramèr *et al.* [65] proves that model extraction attacks in the presence of such "leaky servers" are feasible and efficient (*i.e.* low query complexity) for many hypothesis classes (*e.g.*, logistic regression, multilayer perceptron, and decision trees). In particular, they propose an *equation solving attack* [65, Section 4.1] that uses the confidence values returned by the MLaaS server together with the labels to steal the model parameters. For example, in the case of logistic regression, the MLaaS server knows the parameters $a_0, a_1, \ldots, a_d$ and responds to a query x with the label y ($y = 0$ if $(1 + e^{-a(x)}) \leq 0.5$ and $y = 1$ otherwise) *and the value* $a(x)$ as confidence value for y. Clearly, the knowledge of the confidence values allows an adversary to implement the same attack we describe in Example 2 for linear regression models. In [65, Section 4.2], the authors describes a *path-finding attack* that use the leaf/node identifier returned by the server, even for incomplete queries, to steal a decision tree. These attacks are very efficient (*i.e.*, $d+1$ queries are needed to steal a d-dimensional logistic regression model); however, their efficiency heavily relies on the presence of the various forms of auxiliary information provided by the MLaaS server. While the work in [65] performs preliminary exploration of attacks in the black-box setting [17, 46], it does not consider more recent, and efficient algorithms in the QS scenario. Our work explores this direction through a formalization of the model extraction framework that enables understanding the possibility of extending/improving the active learning attacks presented in [65]. Furthermore, having a better understanding of model extraction attack and its unavoidable connection with active learning is paramount for designing MLaaS systems that are resilient to model extraction.

3.3 Defense Strategies

Our main observation is that model extraction in the context of MLaaS systems described at the beginning of Section 3 (*i.e.*, oracle access) is equivalent to QS active learning. Therefore, any advancement in the area of QS active learning directly translates to a new threat for MLaaS systems. In this section, we discuss strategies that could be used to make the process of extraction more difficult.We investigate the link between machine-learning in the noisy setting and model extraction. The design of a good defense strategy is an open problem; we believe this is an interesting direction for future work where the machine learning and the security communities can fruitfully collaborate.

In this section, we assume that the MLaaS server S with the knowledge of f^*, $S(f^*)$, has the freedom to modify the prediction before forwarding it to the client. More precisely, we assume that there exists a (possibly) randomized procedure D that the server uses to compute the answer $\tilde{y}$ to a query x, and returns that instead of $f^*(x)$. We use the notation $S_D(f^*)$ to indicate that the server S implements D to protect f^*. Clearly, the learner that interacts with $S_D(f^*)$ can still try to learn a function f from the noisy answers from the server. However, the added noise requires the process to make more queries, or could produce a less accurate model than f.

3.3.1 Classification case

We focus on the binary classification problem where $\mathcal{F}$ is an hypothesis class of functions of the form $f : \mathbf{X} \to \mathbf{Y}$ and $\mathbf{Y}$ is binary, but our argument can be easily generalized to the multi-class setting.

First, in the following two remarks we recall two known results from the literature [33] that establish information theoretic bounds (i.e., the computational cost is ignored) for the number of queries required to extract the model when any defense is implemented. Let ν be the generalization error of the model f^* known by the server S_D and μ be the generalization error of the model f learned by an adversary interacting with $S_D(f^*)$. Assume that the hypothesis class $\mathcal{F}$ has VC dimension equal to d. Recall that The VC dimension of a hypothesis class $\mathcal{F}$ is the largest number d such that there exists a subset $X \subset \mathbf{X}$ of size d which can be shattered by $\mathcal{F}$. A set $X = \{x_1, \ldots, x_d\} \subset \mathbf{X}$ is said to be shattered by $\mathcal{F}$ if $|\{(f(x_1), f(x_2), \ldots, f(x_d)) : f \in \mathcal{F}\}| = 2^d$.

Remark 3 (Passive learning). Assume that the adversary uses a passive learning algorithm to compute f, such as the Empirical Risk Minimization (ERM) algorithm, where given a labeled training set $\{(X_1, Y_1), \ldots (X_n, Y_n)\}$, the ERM algorithm outputs $\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[f(X_i) \neq Y_i]$. Then, the adversary can learn $\hat{f}$ with excess error ε (*i.e.*, $\mu \leq \nu + \varepsilon$) with $\tilde{O}(\frac{\nu+\varepsilon}{\varepsilon^2} d)$ examples. For any algorithm, there is a distribution such that the algorithm needs at least $\tilde{\Omega}(\frac{\nu+\varepsilon}{\varepsilon^2} d)$ samples to achieve an excess error of ε.

Remark 4 (Active learning). Assume that the adversary uses an active learning algorithm to compute f, such as the disagreement-based active learning algorithm [33]. Then, the adversary achieves excess error ε with $\tilde{O}(\frac{\nu^2}{\varepsilon^2} d\theta)$ queries (where θ is the disagreement coefficient [33]). For any active learning algorithm, there is a distribution such that it takes at least $\tilde{\Omega}(\frac{\nu^2}{\varepsilon^2} d)$ queries to achieve an excess error of ε.

Observe that any defense strategy D used by a server S to prevent the extraction of a model f^* can be seen as a randomized procedure that outputs $\tilde{y}$ instead of $f^*(x)$ with a given probability over the random coins of D. In the discrete case, we represent this with the notation

$$\rho_D(f^*, x) = \Pr[Y_x \neq f^*(x)], \tag{2}$$

where Y_x is the random variable that represents the answer of the server $S_D(f^*)$ to the query x (*e.g.*, $\tilde{y} \leftarrow Y_x$). When the function f^* is fixed, we can consider the supremum of the function $\rho_D(f^*, x)$, which represents the upper bound for the probability that an answer from $S_D(f^*)$ is wrong:

$$\rho_D(f^*) = \sup_{x \in \mathbf{X}} \rho_D(f^*, x).$$

Before discussing potential defense approaches, we first present a general negative result. The following proposition states that that any candidate defense D that correctly responds to a query with probability greater than or equal to $\frac{1}{2} + c$ for some constant $c > 0$ for all instances can be easily broken. Indeed, an adversary that repetitively queries the same instance x can figure out the correct label $f^*(x)$ by simply looking at the most frequent label that is returned from $S_D(f^*)$. We prove that with this extraction strategy, the number of queries required increases by only a logarithmic multiplicative factor.

Proposition 1. *Let $\mathcal{F}$ be an hypothesis class used for classification and $(\mathcal{L}, \mathcal{O})$ be an active learning system for $\mathcal{F}$ in the QS scenario with query complexity $q(\varepsilon, \delta)$. For any D, randomized procedure for returning labels, such that there exists $f^* \in \mathcal{F}$ with $\rho_D(f^*) < \frac{1}{2}$, there exists an adversary that, interacting with $S_D(f^*)$, can implement an ε-extraction attack with confidence $1 - 2\delta$ and complexity $q = \frac{8}{(1-2\rho_D(f^*))^2} q(\varepsilon, \delta) \ln \frac{q(\varepsilon,\delta)}{\delta}$.*

The proof of Proposition 1 can be found in Appendix A.2.1.
Proposition 1 can be used to discuss the following two different defense strategies:

1. Data-independent randomization. Let $\mathcal{F}$ denote a hypothesis class that is subject to an extraction attack using QS active learning. An intuitive defense for $\mathcal{F}$ involves adding noise to the query output $f^*(x)$ independent of the labeling function f^* and the input query x. In other words, $\rho_D(f, x) = \rho$ for any $x \in \mathbf{X}$, $f \in \mathcal{F}$, and ρ is a constant value in the interval $(0, 1)$. It is easy to see that this simple strategy cannot work. It follows from Proposition 1 that if $\rho < \frac{1}{2}$, then D is not secure. On the other hand, if $\rho \geq \frac{1}{2}$, then the server is useless since it outputs an incorrect label with probability at least $\frac{1}{2}$.

Example 5 (Halfspace extraction under noise). For example, we know that ε-extraction with any level of confidence can be implemented with complexity $q = O(d \log(\frac{1}{\varepsilon}))$ using QS active learning for the class $\mathcal{F}_{d,HS}$ *i.e.* for binary classification via halfspaces (refer Example 4). It follows from the earlier discussion that any defense that flips labels with a constant flipping probability ρ does not work. This defense approach is similar to the case of "noisy oracles" studied extensively in the active learning literature [36, 37, 52]. For example, from the machine-learning literature we know that if the flipping probability is exactly

ρ ($\rho \leq \frac{1}{2}$), the AVERAGE algorithm (similar to our Algorithm 1, defined in Section 4) ε-extracts f^* with $\tilde{O}(\frac{d^2}{(1-2\rho)^2}\log\frac{1}{\varepsilon})$ labels [39]. Under bounded noise where each label is flipped with probability at most ρ ($\rho < \frac{1}{2}$), the AVERAGE algorithm does not work anymore, but a modified Perceptron algorithm can learn with $\tilde{O}(\frac{d}{(1-2\rho)^2}\log\frac{1}{\varepsilon})$ labels [72] in a stream-based active learning setting, and a QS active learning algorithm proposed by Chen *et al.* [16] can also learn with the same number of labels. An adversary implementing the Chen *et al.* algorithm [16] is even more efficient than the adversary $\tilde{A}$ defined in the proof of Proposition 1 (*i.e.*, the total number of queries only increases by a constant multiplicative factor instead of $\ln q(\epsilon,\delta)$). We validate the practical efficiency of this attack in Section 4.

2. Data-dependent randomization. Based on the outcome of the earlier discussion, we believe that a defense that aims to protect a hypothesis class against model extraction via QS active learning should implement data-dependent perturbation of the returned labels. That is, we are interested in a defense D such that the probability $\rho_D(f^*, x)$ depends on the query input x and the labeling function f^*. For example, given a class $\mathcal{F}$ that can be extracted using an active learner $\mathcal{L}$ (in the QS scenario), if we consider a defense D such that $\rho_D(f^*, x) \geq \frac{1}{2}$ for some instances, then the proof of Proposition 1 does not work (the argument only works if there is a constant $c > 0$ such that $\rho_D(f^*, x) \leq \frac{1}{2} - c$ for all x) and the effectiveness of the adversary $\tilde{A}$ is not guaranteed anymore[4].

Example 6 (Halfspace extraction under noise). For the case of binary classification via halfspaces, Alabdulmohsin *et al.* [3] design a system that follows this strategy. They consider the class $\mathcal{F}_{d,HS}$ and design a learning rule that uses training data to infer a distribution of models, as opposed to learning a single model. To elaborate, the algorithm learns the mean μ and the covariance Σ for a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ on $\mathcal{F}_{d,HS}$ such that any model drawn from $\mathcal{N}(\mu, \Sigma)$ provides an accurate prediction. The problem of learning such a distribution of classifiers is formulated as a convex-optimization problem, which can be solved quite efficiently using existing solvers. During prediction, when the label for a instance x is queried, a *new* w is drawn at random from the learned distribution $\mathcal{N}(\mu, \Sigma)$ and the label is computed as $y = \mathrm{sign}(\langle w, x\rangle)$. The authors show that this randomization method can mitigate the risk of reverse engineering without incurring any notable loss in predictive accuracy. In particular, they use PAC active learning algorithms [10, 17] (assuming that the underlying distribution $\mathcal{D}$ is Gaussian) to learn an approximation $\hat{w}$ from queries answered in three different ways: (a) with their strategy, *i.e.* using a new model for each query, (b) using a fixed model to compute all labels, and (c) using a fixed model and adding independent noise to each label, *i.e.* $y = \mathrm{sign}(\langle w, x\rangle + \eta)$

[4] Intuitively, in the binary case if $\rho_D(f^*, x_i) \geq \frac{1}{2}$ then the definition of y_i performed by $\tilde{A}$ in step 2 (majority vote) is likely to be wrong. However, notice that this is not always the case in the multiclass setting: For example, consider the case when the answer to query x_i is defined to be wrong with probability $\geq \frac{1}{2}$ and, when wrong, is sampled uniformly at random among the $k-1$ classes that are different to the true class $f^*(x)$, then if k is large enough, y_i defined via the majority vote is likely to be still correct.

and $\eta \leftarrow [-1, +1]$. They show that the geometric error of $\hat{w}$ with respect to the true model is higher in the former setting (*i.e.* in (a)) than in the others. On 15 different datasets from the UC Irvine repository [2], their strategy gives typically an order of magnitude larger error. We empirically evaluate this defense in the context of model extraction using QS active learning algorithms in Section 4.

Continuous case: Generalizing Proposition 1 to the continuous case does not seem straightforward, *i.e.* when the target model held by the MLaaS server is a real-valued function $f^* : \mathbf{X} \rightarrow \mathbb{R}$; A detailed discussion about the continuous case appears in Appendix A.3.

4 Implementation and Evaluation

We carried out experiments to validate our claims that query synthesis active learning can be used to successfully perform model extraction. Our experiments are designed to answer the following three questions: (1) Is active learning practically useful in settings without any auxiliary information, such as confidence values *i.e.* in an oracle access setting?, (2) Is active learning useful in scenarios where the oracle is able to perturb the output *i.e.* in a data-independent randomization setting?, and (3) Is active learning useful in scenarios where the oracle is able to perform more subtle perturbations *i.e.* in a data-dependent randomization setting?

To answer these questions, we focused on learning the hypothesis class of d-dimensional half spaces. To perform model extraction, we implemented two QS algorithms [4, 16] to learn an approximation w, and terminate execution when $||w^* - w||_2 \leq \varepsilon$. The metric we use to capture efficiency is query complexity. To provide a monetary estimate of an attack, we borrow pricing information from the online pricing scheme of Amazon *i.e.* \$0.0001 per query (more details are present in Table 2 in Appendix A.1). We considered alternative stopping criteria, such as measuring the learned model's stability over the N last iterations. Such a method resulted in comparable error and query complexity (refer Appendix A.4.1 for detailed results). For our experiments, the halfspace held by the server/oracle (*i.e.*, the optimal hypothesis w^*) was learned using Python's `scikit-learn` library. All experiments were executed on a Ubuntu 16.04 server with 32 GB RAM, and an Intel i5-6600 CPU clocking 3.30GHz. Our experiments suggest that:

1. QS active learning algorithms are efficient for model extraction, with low query complexity and run-time. For the digits dataset ($d = 64$), the dataset with the largest value of d which we evaluated on, the active learning algorithm implemented required 900 queries to extract the halfspace with geometric error $\varepsilon \leq 10^{-4}$. This amounts to \$0.09 worth of queries.
2. QS active learning algorithms are also efficient when the oracle flips the labels independently with constant probability ρ. This only moderately increases the query complexity (for low values of ρ). For the digits dataset of

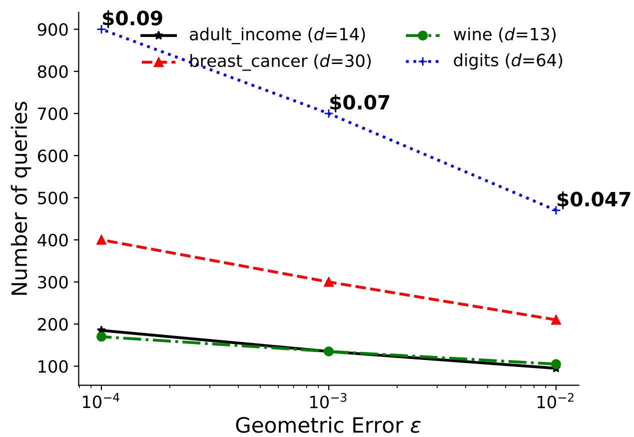


Figure 3: Number of queries needed for halfspace extraction using the version space approximation algorithm. Note that the asymptotic query complexity for this algorithm is $O(d \log \frac{1}{\varepsilon})$. This explains the increase in query complexity as a function of d.

input dimensionality $d = 64$, and a noise threshold $\rho = 0.4$, our algorithm required 36546 queries (or \$3.65) to extract the halfspace with geometric error $\varepsilon \leq 10^{-4}$.

3. State-of-the-art QS algorithms fail to recover the model when the oracle responds to queries using tailored model randomization techniques (refer subsection 3.3, specifically the algorithm by Alabdulmohsin *et al.* [3]). However, passive learning algorithms (refer Algorithm 1) are effective in this setting.

4.1 Detailed Results

We begin by describing evaluation results for the aforementioned three questions. In each figure, we plot the price (*i.e.* \$0.0001 per query) for the most expensive attack we launch to serve as a baseline. We conclude by comparing our approach with the algorithm proposed by Lowd and Meek [46].

Q1. Usefulness in an oracle access setting: We implemented Version Space Approximation proposed by Alabdulmohsin *et al.* [4] in approximately 50 lines of MATLAB. This algorithm operates iteratively, based on the principles of version space learning. A version space [49] is a hierarchical representation of knowledge. It can also be thought of as the subset of hypotheses consistent with the training examples. In each iteration, the algorithm first approximates a version space, and then synthesizes an instance that reduces this approximated version space quickly. The final query complexity for this algorithm is $O(d \log \frac{1}{\varepsilon})$.

Figure 3 plots the number of queries needed to extract a halfspace as a function of termination criterion *i.e.* geometric error ε. As discussed earlier, the query complexity is dependent on the dimensionality of the halfspace to be extracted. Across all values of dimensionality d, observe that with the exponential decrease in error ε, the increase in query complexity is linear - often by a small factor ($1.3\times - 1.5\times$). The implemented query synthesis algorithm involves solving a convex optimization problem to approximate the version space, an operation that is potentially time consuming. However, based on several runs of our experiment, we noticed that the algorithm always converges in < 2 minutes.

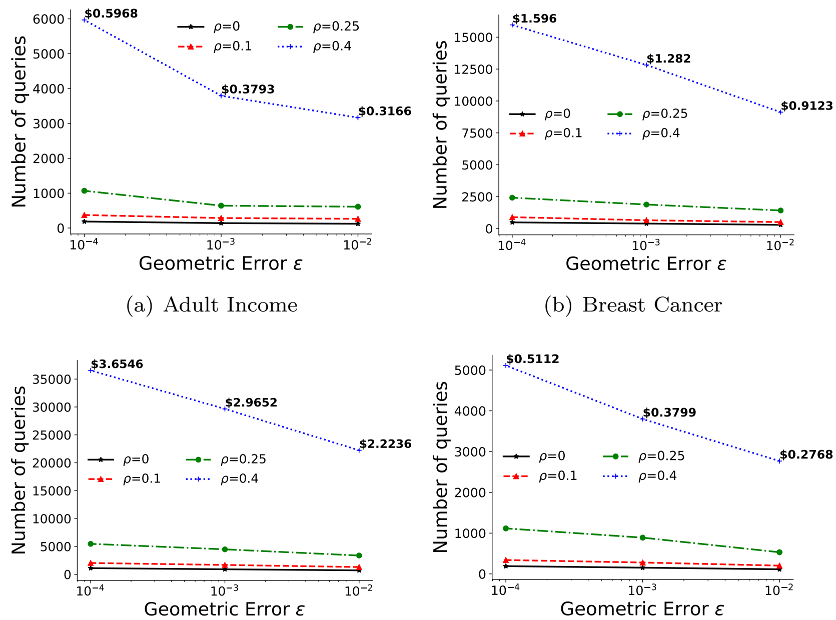
While the equation solving attack proposed by Tramèr *et al.* [65] requires fewer queries, it also requires the actual value of the prediction output *i.e.* $\langle w^*, x\rangle$ as auxiliary information. On the other hand, extraction using query synthesis does not rely on any auxiliary information returned by the MLaaS server to increase its efficiency *i.e.* the only input needed for query synthesis-based extraction attacks is $\text{sign}(\langle w^*, x\rangle)$.

Q2. Resilience to data-independent noise: An intuitive defense against model extraction might be to flip the sign of the prediction output with independent probability ρ *i.e.* if the output $y \in \{1, -1\}$, then $Pr[y \neq \text{sign}\langle w^*, x\rangle] = \rho < \frac{1}{2}$ (refer subsection 3.3). This setting (*i.e.*, noisy oracles) is extensively studied in the machine learning community. Trivial solutions including repeated sampling to obtain a batch where majority voting (determines the right label) can be employed; if the probability that the outcome of the vote is correct is represented as $1 - \alpha$, then the batch size needed for the voting procedure is $k = O(\frac{\log \frac{1}{\alpha}}{|\rho - 0.5|^2})$ *i.e.* there is an increase in query complexity by a (multiplicative) factor k, an expensive proposition. While other solutions exist [51, 71], we implemented the dimension coupling framework proposed by Chen *et al.* [16] in approximately 150 lines of MATLAB. The dimension coupling framework reduces a $d-$dimensional learning problem to $d - 1$ lower-dimensional sub-problems. It then appropriately aggregates the results to produce a halfspace. This approach is resilient to noise *i.e.* the oracle can flip the label with constant probability (known a priori) $\rho < \frac{1}{2}$, and the algorithm will converge with probability $1 - \delta$. The query complexity for this algorithm is $\tilde{O}(d\ (\log \frac{1}{\varepsilon} + \log \frac{1}{\delta}))$.

The results of our experiment are presented in Figure 4. The algorithm is successful in extracting the halfspace for a variety of ρ values. The exact bound is $C(\rho)(d\ (\log \frac{1}{\varepsilon} + \log \frac{1}{\delta})$, where $C(\rho)$ is a function of ρ that is approximately $O(\frac{\rho \log^3(1/\rho)}{\log^2 2(1-\rho)})$. Thus, there is a multiplicative increase in the number of queries with increase in ρ. This introduces a modest increase in complexity in comparison to the noise-free setting. While the increase in pricing is $\approx 40\times$, this results in a worst case expenditure of $\approx \$3.6$ (see Figure 4(c)). The time (and number of queries) taken for convergence is proportional to ρ, ranging from $1 - 20$ minutes for successful completion.

Q3. Resilience to data-dependent noise: As alluded to in subsection 3.3, another defense against extraction involves learning a family of functions very similar to w^* such that they all provide accurate predictions with high probability. Proposed by Alabdulmohsin *et al.* [3], data-dependent randomization enables the MLaaS server to sample a random function for each query *i.e.* for each instance x_i, the MLaaS server obtains a new $w_i \sim \mathcal{N}(\mu, \sigma)$ and responds with $y_i = \text{sign}(\langle w_i, x_i\rangle)$. Thus, this approach can be thought of as flipping the sign of the prediction output with probability $\rho_D(w^*, x_i)$ (see subsection 3.3).

In this algorithm, a separation parameter C determines how close the sam-



(a) Adult Income

(b) Breast Cancer

(c) Digits

(d) Wine

Figure 4: Number of queries needed for halfspace extraction using the dimension coupling algorithm. Note that the asymptotic query complexity for this algorithm is $\tilde{O}(d(\log \frac{1}{\varepsilon} + \log \frac{1}{\delta}))$. This explains the increase in query complexity as a function of d.

ples from $\mathcal{N}(\mu, \sigma)$ are; larger the value of C, closer each sample is (refer Section 4 in [3] for more details). We measure the value of $\rho_D(w^*, x_i)$ as a function of C for those x_i values generated by the dimension coupling algorithm. $\rho_D(w^*, x_i)$ is estimated by (a) obtaining $w_1, \cdots, w_n \sim \mathcal{N}(\mu, \sigma)$, for $n = 1000$, and using them to classify x_i to obtain $y_1 = \text{sign}(\langle w_1, x_i \rangle), \cdots, y_n$, and (b) obtaining the percentage of the prediction outputs that is not equal to $\text{sign}(\langle w^*, x_i \rangle)$. Our hope was that if the value of $\texttt{max}_{\forall x_i}\ \rho_D(w^*, x_i) < \frac{1}{2}$, then an adversary similar to $\tilde{A}$ defined in Proposition 1 could be used to perform extraction.

Figure 5 suggests otherwise; the average value of $\rho_D(w^*, x_i) \approx \frac{1}{2} \pm \gamma$ for some small $\gamma > 0$. Since any adversary will be unable to determine a priori the inputs for which this value is greater than half, neither majority voting, nor the vanilla dimension coupling framework will help extract the halfspace. We believe this is the case for current state-of-the-art algorithms as the instances they synthesize are "close" to the optimal halfspace. To validate this claim, we measured this distance for both the algorithms [4, 16]. We observed that a majority of the points are very close to the halfspace in both cases (see Figure 7 in Appendix A.4.2 for more details).

Such forms of data-dependent randomization, however, are not secure against traditional passive learning algorithms. Such an algorithm takes as input an estimated upper bound $\hat{\sigma}$ for σ. The algorithm first draws $\tilde{O}(\frac{d}{\varepsilon^2} \max(1, d\hat{\sigma}^2))$ instances from the $d-$dimensional unit sphere $\mathbb{S}^{d-1}$ uniformly at random, and

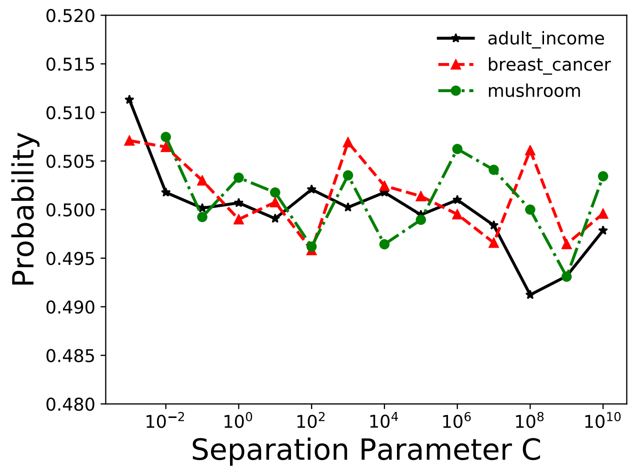


Figure 5: $\texttt{average}\rho_D(w^*, x_i) \approx \frac{1}{2} \pm \gamma$; x_i synthesized by the dimension coupling algorithm.

Algorithm 1 Passive Learning Algorithm that breaks [3]

1: **Input**: variance upper bound $\hat{\sigma} \geq \frac{1}{\sqrt{d}}$, target error ε

2: $m \leftarrow \frac{(15\pi)^2}{\varepsilon^2} d \max(1, d\hat{\sigma}^2) \log \frac{2d}{\delta}$, $l \leftarrow \frac{1}{12d\hat{\sigma}}$

3: Draw $x_1, x_2, \ldots, x_m \in \mathbb{S}^{d-1}$ uniformly at random, and query their labels $y_1, y_2, \ldots, y_m$

4: $v \leftarrow \sum_{i=1}^{m} y_i x_i$

5: **if** $\|v\| \geq l$ **then**

6: Return $w = \frac{v}{\|v\|}$

7: **else**

8: Return FAIL

9: **end if**

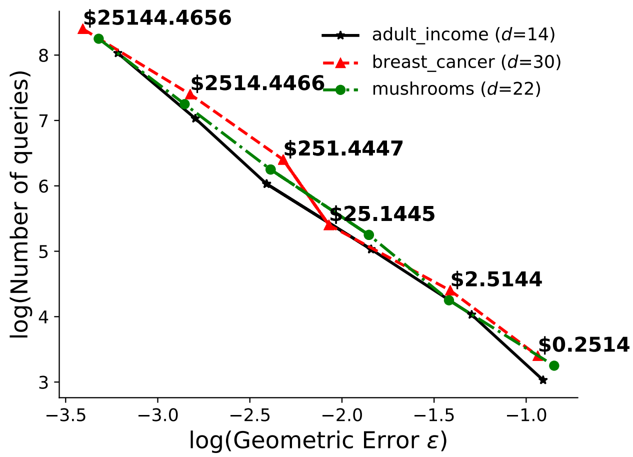


Figure 6: log(Number of queries) needed for halfspace extraction (protected by the defense strategy proposed in [3]) using Algorithm 1. Note that the asymptotic query complexity for this algorithm is $O(\frac{1}{\varepsilon^2} d \max(1, d\hat{\sigma}^2) \log \frac{2d}{\delta})$. This explains the increase in query complexity as a function of d and ε. The large value of $\frac{Cd}{\varepsilon^2}$ dominates the query complexity in this algorithm. The price is plotted for the attack on the breast cancer dataset.

proceeds to have them labeled - by the oracle defined in [3] in this case. It then computes the average $v = \sum_{i=1}^{m} y_i x_i$. $w = \frac{v}{\|v\|}$, the direction of v, is the algorithm's estimate of the classifier w^*, and the length of v is used as an indicator of whether the algorithm succeeds: if this estimated upper bound is correct (i.e. $\sigma \leq \hat{\sigma}$), then with high probability, $\|w - w^*\| \leq \varepsilon$; otherwise it outputs FAIL, indicating the variance bound $\hat{\sigma}$ is incorrect. In such situations, we can reduce $\hat{\sigma}$ and try again. A detailed proof of the algorithm's guarantees is available in Appendix A.2.2.

While the asymptotic bounds for Algorithm 1 are larger than the active learning algorithms discussed thus far, the constant $\mathcal{C} = (15\pi)^2$ can be reduced by a multiplicative factor to reduce the total number of queries used *i.e.* $\frac{\mathcal{C}}{100}$ or $\frac{\mathcal{C}}{1000}$ etc. In Figure 6, we observe that extracting halfspaces with geometric error $\varepsilon \approx 10^{-1}$ requires $\leq 10^4$ queries. While achieving $\varepsilon \approx 10^{-3}$ requires $\approx 10^7$ queries, the algorithm can be executed in parallel enabling faster run-times.

Lowd and Meek Baseline: The algorithm proposed by Lowd and Meek [46] can also be used to extract a halfspace. However, note that this algorithm can only operate in a noise-free setting. This is a severe limitation if a setting where the MLaaS employs defense strategies. From Table 4.1, one can observe that the number of queries required to extract the halfspace is more than the query synthesis algorithms we implemented. For example, consider the breast cancer dataset. The version space algorithm is able to extract a halfspace at a distance of $\varepsilon \leq 10^{-4}$ with 400 queries (or \$0.04). However, the algorithm proposed by Lowd and Meek takes 970 queries for extraction. Additionally, the geometric error of the extracted halfspaces are also higher than those extracted in the query synthesis case. The query complexity of the Lowd and Meek algorithm

Dataset	**Queries**	ε	**Slowdown**
Wine	189	0.071	1.67×
Breast Cancer	940	0.162	3.19×
Digits	1879	0.665	2.62×

Table 1: Number of queries and geometric error observed after extracting halfspaces using the line search procedure proposed by Lowd and Meek. Observe that the geometric error in some cases is large. Slowdown indicates the ratio between number of queries taken for the Lowd and Meek procedure and those taken by the DC^2 algorithm for $\varepsilon = 0.01$, and $\rho = 0$.

is $O(d \log(\frac{1}{a\varepsilon}))$, where $a = \min_{i=1,\cdots,d} \frac{|w_i^*|}{\|w^*\|}$ (w_i^* is the i-th coordinate of the groundtruth classifier w^*). This is worse than the $O(d\log(\frac{1}{\varepsilon}))$ query complexity of classical active learning algorithms. While this algorithm is not tailored to minimize the geometric error, we believe that these results further validate our claim that query synthesis active learning is a promising direction to explore.

5 Discussion

We begin our discussion by highlighting algorithms an adversary could use if the assumptions made about the operational ecosystem are relaxed. Then, we discuss strategies that can potentially be used to make the process of extraction more difficult, and shortcomings in our approach.

5.1 Varying The Adversary's Capabilities

The operational ecosystem in this work is one where the adversary is able to synthesize data-points de novo to extract a model through oracle access. In this section, we discuss other algorithms an adversary could use if this assumption is relaxed. We begin by discussing other models an adversary can learn in the query synthesis regime, and move on to discussing algorithms in other approaches.

Equivalence queries. In her seminal work, Angluin [5] proposes a learning algorithm, L^*, to correctly learn a regular set from any minimally adequate teacher, in polynomial time. For this to work, however, *equivalence queries* are also needed along with membership queries. Should MLaaS servers provide responses to such equivalence queries, different extraction attacks could be devised. To learn linear decision boundaries, Wang *et al.* [69] first synthesize an instance close to the decision boundary using labeled data, and then select the real instance closest to the synthesized one as a query. Similarly, Awasthi *et al.* [8] study learning algorithms that make queries that are close to examples generated from the data distribution. These attacks require the adversary to have access to some subset of the original training data. In other domains, program synthesis using input-output example pairs [24, 31, 56, 68] also follows a similar principle.

If the adversary had access to a subset of the training data, or had prior knowledge of the distribution from which this data was drawn from, it could launch a different set of attacks based on the algorithms discussed below.

Stream-based selective sampling. Atlas *et al.* [7] propose selective sampling as a form of directed search (similar to Mitchell [48]) that can greatly increase the ability of a connectionist network (*i.e.* power system security analysis in their paper) to generalize accurately. Dagan *et al.* [19] propose a method for training probabilistic classifiers by choosing those examples from a stream that are *more informative*. Lindenbaum *et al.* [44] present a lookahead algorithm for selective sampling of examples for nearest neighbor classifiers. The algorithm looks for the example with the highest utility, taking its effect on the resulting classifier into account. Another important application of selective learning was for feature selection [45], an important preprocessing step. Other applications of stream-based selective sampling include sensor scheduling [40], learning ranking functions for information retrieval [73], and in word sense disambiguation [30]. *Pool-based sampling.* Dasgupta [22] surveys active learning in the non-separable case, with a special focus on statistical learning theory. He claims that in this setting, AL algorithms usually follow one of the following two strategies - (i) Efficient search in the hypothesis spaces (as in the algorithm proposed by Chen *et al.* [16], or by Cohn *et al.* [17]), or (ii) Exploiting clusters in the data (as in the algorithm proposed by Dasgupta *et al.* [23]). The latter option can be used to learn more complex models, such as decision trees. As the ideal halving algorithm is difficult to implement in practice, pool-based approximations are used instead such as uncertainty sampling and the query-by-committee (QBC) algorithm [14, 29, 63]. Unfortunately, such approximation methods are only guaranteed to work well if the number of unlabeled examples (i.e. pool size) grows exponentially fast with each iteration. Otherwise, such heuristics become crude approximations and they can perform quite poorly.

5.2 Complex Models

PAC active learning strategies have proven effective in learning DNNs. The work of Sener *et al.* [58] selects the most representative points from a sample of the training distribution to learn the DNN. Papernot *et al.* [53] employ substitute model training - a procedure where a small training subset is strategically augmented and used to train a shadow model that resembles the model being attacked. Note that the prior approaches rely on some additional information, such as a subset of the training data.

Active learning algorithms considered in this paper work in an iterative fashion. Let $\mathcal{H}$ be the entire hypothesis class. At time time $t \geq 0$ let the set of possible hypothesis be $\mathcal{H}_t \subseteq \mathcal{H}$. Usually an active-learning algorithm issues a query at time t and updates the possible set of hypothesis to $\mathcal{H}_{t+1}$, which is a subset of $\mathcal{H}_t$. Once the size of $\mathcal{H}_t$ is "small" the algorithm stops. Analyzing the effect of a query on possible set of hypothesis is very complicated in the context of complex models, such as DNNs. We believe this is a very important and interesting direction for future work.

5.3 Model Transferability

Most work in active learning has assumed that the correct hypothesis space for the task is already known *i.e.* if the model being learned is for logistic regression, or is a neural network and so on. In such situations, observe that the labeled data being used is biased, in that it is implicitly tied to the underlying hypothesis. Thus, it can become problematic if one wishes to re-use the labeled data chosen to learn *another, different* hypothesis space. This leads us to *model transferability*[5], a less studied form of defense where the oracle responds to any query with the prediction output from an entirely different hypothesis class. For example, imagine if a learner tries to learn a halfspace, but the teacher performs prediction using a boolean decision tree. Initial work in this space includes that of Shi *et al.* [60], where an adversary can steal a linear separator by learning input-output relations using a deep neural network. However, the performance of query synthesis active learning in such ecosystems is unclear.

5.4 Limitations

We stress that these limitations are not a function of our specific approach, and stem from the theory of active learning. Specifically: (1) As noted by Dasgupta [21], the label complexity of PAC active learning depends heavily on the specific target hypothesis, and can range from $O(\log \frac{1}{\varepsilon})$ to $\Omega(\frac{1}{\varepsilon})$. Similar results have been obtained by others [34, 50]. This suggests that for some hypotheses classes, the query complexity of active learning algorithms is as high as that in the passive setting. (2) Some query synthesis algorithms assume that there is some labeled data to bootstrap the system. However, this may not always be true, and randomly generating these labeled points *may* adversely impact the performance of the algorithm. (3) For our particular implementation, the algorithms proposed rely on the *geometric error* between the optimal and learned halfspaces. Oftentimes, however, there is no direct correlation between this geometric error and the generalization error used to measure the model's *goodness*.

6 Related Work

Machine learning algorithms and systems are optimized for performance. Little attention is paid to the security and privacy risks of these systems and algorithms. Our work is motivated by the following attacks against machine learning.

1. Causative Attacks: These attacks are primarily geared at *poisoning* the training data used for learning, such that the classifier produced performs erroneously during test time. These include: (a) mislabeling the training data, (b) changing rewards in the case of reinforcement learning, or (c) modifying the sampling mechanism (to add some bias) such that it does not reflect the true underlying distribution in the case of unsupervised learning [55]. The work of Papernot *et al.* [54] modify input features resulting in misclassification by Deep Neural Networks.

2. Evasion Attacks: Once the algorithm has trained successfully, these forms of attacks provide *tailored* inputs such that the output is erroneous. These *noisy inputs* often preserves the semantics of the original inputs, are human imperceptible, or are physically realizable. The well studied area of *adversarial examples* is an instantiation of such an attack. Moreover, evasion attacks can

[5] A special case of agnostic active learning [9].

also be even black-box *i.e.* the attacker needn't know the model. This is because an adversarial example optimized for one model is highly likely to be effective for other models. This concept, known as *transferability*, was introduced by Carlini *et al.* [15]. Notable works in this space include [12, 25, 27, 41, 42, 53, 64, 70]

3. Exploratory Attacks: These forms of attacks are the primary focus of this work, and are geared at learning intrinsics about the algorithm used for training. These intrinsics can include learning model parameters, hyperparameters, or training data. Typically, these forms of attacks fall in two categories - *model inversion*, or *model extraction*. In the first class, Fredrikson *et al.* [28] show that an attacker can learn sensitive information about the dataset used to train a model, given access to side-channel information about the dataset. In the second class, the work of Tramer *et al.* [65] provides attacks to learn parameters of a model hosted on the cloud, through a query interface. Termed *membership inference*, Shokri *et al.* [61] learn the training data used for machine learning by training their own inference models. Wang *et al.* [67] propose attacks to learn a model's hyperparameters.

7 Conclusions

In this paper, we formalize model extraction in the context of Machine-Learning-as-a-Service (MLaaS) servers that return only prediction values (*i.e.*, oracle access setting), and we study its relation with *query synthesis* active learning (Observation 1). Thus, we are able to implement efficient attacks to the class of halfspace models used for binary classification (Section 4). While our experiments focus on the class of halfspace models, we believe that extraction via active learning can be extended to multiclass and non-linear models such as deep neural networks, random forests etc. We also begin exploring possible defense approaches (subsection 3.3). To the best of our knowledge, this is the first work to formalize security in the context of MLaaS systems. We believe this is a fundamental first step in designing more secure MLaaS systems. Finally, we suggest that data-dependent randomization (*e.g.*, model randomization as in [3]) is the most promising direction to follow in order to design effective defenses.

References

[1] https://www.theregister.co.uk/2017/12/18/black_box_ai_attack/, 2017.

[2] https://archive.ics.uci.edu/ml/datasets.html, 2018.

[3] Ibrahim M. Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Adding robustness to support vector machines against adversarial reverse engineering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 231–240, 2014.

[4] Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Efficient active learning of halfspaces via query synthesis. In *AAAI*, pages 2483–2489, 2015.

[5] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106, 1987.

[6] Giuseppe Ateniese, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *IJSN*, 10(3):137–150, 2015.

[7] Les E Atlas, David A Cohn, and Richard E Ladner. Training connectionist networks with queries and selective sampling. In *Advances in neural information processing systems*, pages 566–573, 1990.

[8] Pranjal Awasthi, Vitaly Feldman, and Varun Kanade. Learning using local membership queries. In *Conference on Learning Theory*, pages 398–431, 2013.

[9] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. *Journal of Computer and System Sciences*, 75(1):78–89, 2009.

[10] Maria-Florina Balcan, Andrei Z. Broder, and Tong Zhang. Margin based active learning. In *Learning Theory, 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA, June 13-15, 2007, Proceedings*, pages 35–50, 2007.

[11] Maria-Florina Balcan and Philip M. Long. Active and passive learning of linear separators under log-concave distributions. In *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, pages 288–316, 2013.

[12] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Black-box attacks on deep neural networks via gradient estimation. 2018.

[13] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[14] Klaus Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 59–66, 2003.

[15] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[16] Lin Chen, Seyed Hamed Hassani, and Amin Karbasi. Near-optimal active learning of halfspaces via query synthesis in the noisy setting. In *AAAI*, pages 1798–1804, 2017.

[17] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[18] Hsu D. and Sabato S. Heavy-tailed regression with a generalized median-of-means. In *International Conference on Machine Learning (ICML)*, 2014.

[19] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157. The Morgan Kaufmann series in machine learning,(San Francisco, CA, USA), 1995.

[20] S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *NIPS*, 2007.

[21] Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 235–242, 2005.

[22] Sanjoy Dasgupta. Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781, 2011.

[23] Sanjoy Dasgupta, Daniel J Hsu, and Claire Monteleoni. A general agnostic active learning algorithm. In *Advances in neural information processing systems*, pages 353–360, 2008.

[24] Dana Drachsler-Cohen, Sharon Shoham, and Eran Yahav. Synthesis with abstract examples. In *International Conference on Computer Aided Verification*, pages 254–278. Springer, 2017.

[25] Gamaleldin F Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alex Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both human and computer vision. *arXiv preprint arXiv:1802.08195*, 2018.

[26] Chaudhuri K. et al. Convergence rates of active learning for maximum likelihood estimation. In *Advances in Neural Information Processing Systems*, 2015.

[27] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 1, 2017.

[28] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security Symposium*, pages 17–32, 2014.

[29] Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2):133–168, 1997.

[30] Atsushi Fujii, Takenobu Tokunaga, Kentaro Inui, and Hozumi Tanaka. Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, 24(4):573–597, 1998.

[31] Sumit Gulwani. Synthesis from examples: Interaction models and algorithms. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on*, pages 8–14. IEEE, 2012.

[32] S. Hanneke. A bound on the label complexity of agnostic active learning. In *ICML*, 2007.

[33] Steve Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3):131–309, 2014.

[34] Tibor Hegedűs. Generalized teaching dimensions and the query complexity of learning. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 108–117. ACM, 1995.

[35] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I. P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, AISec 2011, Chicago, IL, USA, October 21, 2011*, pages 43–58, 2011.

[36] Matti Kääriäinen. Active learning in the non-realizable case. In *Algorithmic Learning Theory, 17th International Conference, ALT 2006, Barcelona, Spain, October 7-10, 2006, Proceedings*, pages 63–77, 2006.

[37] Richard M. Karp and Robert Kleinberg. Noisy binary search and its applications. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 881–890, 2007.

[38] Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009.

[39] Adam R. Klivans and Pravesh Kothari. Embedding hard learning problems into gaussian space. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 793–809, 2014.

[40] Vikram Krishnamurthy. Algorithms for optimal scheduling and management of hidden markov model sensors. *IEEE Transactions on Signal Processing*, 50(6):1382–1397, 2002.

[41] Alex Kurakin, Dan Boneh, Florian Tramèr, Ian Goodfellow, Nicolas Papernot, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. 2018.

[42] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[43] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM J. Comput.*, 22(6):1331–1348, 1993.

[44] Michael Lindenbaum, Shaul Markovitch, and Dmitry Rusakov. Selective sampling for nearest neighbor classifiers. In *AAAI/IAAI*, pages 366–371. Citeseer, 1999.

[45] Huan Liu, Hiroshi Motoda, and Lei Yu. A selective sampling approach to active feature selection. *Artificial Intelligence*, 159(1-2):49–74, 2004.

[46] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 641–647, 2005.

[47] Andrew McCallum and Kamal Nigam. Employing EM and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning, Madison, Wisconsin, USA, July 24-27, 1998*, pages 350–358, 1998.

[48] Tom M Mitchell. Generalization as search. *Artificial intelligence*, 18(2):203–226, 1982.

[49] Tom Michael Mitchell. Version spaces: an approach to concept learning. Technical report, STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE, 1978.

[50] Mohammad Naghshvar, Tara Javidi, and Kamalika Chaudhuri. Noisy bayesian active learning. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 1626–1633. IEEE, 2012.

[51] Robert Nowak. Noisy generalized binary search. In *Advances in neural information processing systems*, pages 1366–1374, 2009.

[52] Robert D. Nowak. The geometry of generalized binary search. *IEEE Trans. Information Theory*, 57(12):7893–7906, 2011.

[53] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[54] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[55] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.

[56] Hila Peleg, Shachar Itzhaky, and Sharon Shoham. Abstraction-based interaction model for synthesis. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 382–405. Springer, 2018.

[57] Sabato S. and Munos R. Active regression by stratification. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[58] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. 2018.

[59] B Settles. Active learning literature survey univ. wisconsin-madison, madison, wi, 2009. Technical report, CS Tech. Rep. 1648.

[60] Yi Shi, Yalin Sagduyu, and Alexander Grushin. How to steal a machine learning classifier with deep learning. In *Technologies for Homeland Security (HST), 2017 IEEE International Symposium on*, pages 1–5. IEEE, 2017.

[61] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 3–18. IEEE, 2017.

[62] Nedim Srndic and Pavel Laskov. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 197–211, 2014.

[63] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[64] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[65] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 601–618, 2016.

[66] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[67] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. *arXiv preprint arXiv:1802.05351*, 2018.

[68] Chenglong Wang, Alvin Cheung, and Rastislav Bodik. Interactive query synthesis from input-output examples. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1631–1634. ACM, 2017.

[69] Liantao Wang, Xuelei Hu, Bo Yuan, and Jianfeng Lu. Active learning via query synthesis and nearest neighbour search. *Neurocomputing*, 147:426–434, 2015.

[70] David Warde-Farley and Ian Goodfellow. 11 adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, page 311, 2016.

[71] Songbai Yan, Kamalika Chaudhuri, and Tara Javidi. Active learning from imperfect labelers. In *Advances in Neural Information Processing Systems*, pages 2128–2136, 2016.

[72] Songbai Yan and Chicheng Zhang. Revisiting perceptron: Efficient and label-optimal learning of halfspaces. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1056–1066, 2017.

[73] Hwanjo Yu. Svm selective sampling for ranking with application to data retrieval. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 354–363. ACM, 2005.

[74] Chicheng Zhang and Kamalika Chaudhuri. Beyond disagreement-based agnostic active learning. In *Advances in Neural Information Processing Systems*, pages 442–450, 2014.

A Appendix

A.1 Salient Features Of MLaaS Providers

We present salient featurs of popular MLaaS providers in Table 2. Note that the pricing in the batch setting is for every 1000 queries predicted, while the pricing in the online setting is per query. Observe that all these providers accept queries in the form of membership queries, but there exists no notion of equivalence queries.

	Google	Amazon	Microsoft
PRICING			
• **Batch**	\$0.093*	\$0.1	\$0.5
• **Online**	\$0.056*	\$0.0001	\$0.0005
MODELS			
• **DNNs**	✓	✗	✓
• **Regression**	✓	✓	✓
• **Decision trees**	✓	✗	✓
• **Random forests**	✓	✗	✓
• **Binary & n-ary classification**	✓	✓	✓
QUERY TYPES			
• **Equivalence**	✗	✗	✗
• **Membership**	✓	✓	✓

Table 2: Salient features of popular MLaaS providers. *Google's pricing model is per node per hour.

In Table 3, we summarize the auxiliary information shared by MLaaS providers in the status quo. Note that some of this auxiliary information may not be required for all use-cases.

Models	Google	Amazon	Microsoft
• **DNNs**	Confidence Score	✗	Confidence Score
• **Regression**	Confidence Score	Confidence Score	Confidence Score
• **Decision trees**	Leaf Node	✗	Leaf Node
• **Random forests**	Leaf Node	✗	Leaf Node
• **Binary & n-ary classification**	Confidence Score	Confidence Score	Confidence Score

Table 3: Auxiliary information shared. Leaf node denotes the exact leaf (and not an intermediary node) where the computation halts, and ✗indicates the absence of support for the associated model.

A.2 Proofs

A.2.1 Proof of Proposition 1

Proof. Let $\tilde{A}$ be the adversary that does the following:
for $i = 1, \ldots, q(\varepsilon, \delta)$

1. $\tilde{A}$ uses the query strategy of $\mathcal{L}$ to generate the instance x_i;
2. $\tilde{A}$ queries x_i to $S_D(f^*)$ for r times and defines y_i the most frequent labels among the r answers (we assume r is an even integer).

At the end, $\tilde{A}$ (as the learner $\mathcal{L}$) learns $\hat{f}$ using the points $\{(x_i, y_i)\}_{i=1,\cdots,q(\varepsilon,\delta)}$. Let $q = r \cdot q(\varepsilon, \delta)$, then it holds by the union bound that

$$\Pr[\mathtt{Exp}^{\varepsilon}_{\mathcal{F}}(S_D(f^*), \tilde{A}, q) = 1] \geq$$
$$1 - \delta - (1 - \Pr[\cap_{i=1}^{q(\varepsilon,\delta)} \{y_i = f^*(x_i)\}]) \,.$$

Define X_i^j as the binary random variable that is 1 if and only if the answer to the j-th query of x_i is correct and $X_i = \sum_{j=1}^{r} X_i^j$, then

$$\begin{aligned}\Pr[\cap_{i=1}^{q(\varepsilon,\delta)} \{y_i = f^*(x_i)\}] &\geq \Pr[\cap_{i=1}^{q(\varepsilon,\delta)} \{X_i > r/2\}] \\ &\geq 1 - \sum_{i=1}^{q(\varepsilon,\delta)} \Pr[\{X_i \leq r/2\}]\end{aligned}$$

where the last step follows from the union bound. Now, observe that $\mathbb{E}[X_i] = r(1 - \rho_D(f^*, x_i)) > r/2$ and the Chernoff bound can be applied on each term in the right-hand side. In particular, we have that $\Pr[\{X_i \leq r/2\}] \leq e^{-r\frac{\left(\rho_D(f^*) - \frac{1}{2}\right)^2}{2}}$ and it follows that

$$\Pr[\mathtt{Exp}^{\varepsilon}_{\mathcal{F}}(S_D(f^*), \tilde{A}, q) = 1] \geq 1 - \delta - q(\varepsilon, \delta)\, e^{-r\frac{(\rho_D(f^*) - \frac{1}{2})^2}{2}} \,.$$

By setting $r = \frac{8}{(1-2\rho_D(f^*))^2} \ln \frac{q(\varepsilon,\delta)}{\delta}$ we have $\Pr[\mathtt{Exp}^{\varepsilon}_{\mathcal{F}}(S_D(f^*), \tilde{A}, q) = 1] \geq 1 - 2\delta$. That is, the adversary $\tilde{A}$ implements an ε-extraction with Confidence Score $1 - 2\delta$ and complexity $q = \frac{8}{(1-2\rho_D(f^*))^2} q(\varepsilon, \delta) \ln \frac{q(\varepsilon,\delta)}{\delta}$.

□

A.2.2 Proof of Algorithm 1

Here, we discuss the analysis and proofs associated with Algorithm 1.

Assume unit vector $\mu \in \mathbb{R}^d$ is the ground truth. For each query $x \in \mathbb{R}^d$, a vector w is drawn from $N(\mu, \sigma^2 I)$, and the label $y = \mathrm{sign}(\langle w, x\rangle)$ is returned. The goal of the learner (attacker) is to return $\hat{w}$ such that $\|\mu - \hat{w}\|$ is small.

We have following theoretical guarantees for Algorithm 1.

Proposition 1. *If $\sigma \leq \hat{\sigma}$, then $\|\hat{w} - \mu\| \leq \varepsilon$ with probability at least $1 - \delta$.*

Proposition 2. *If $\sigma \leq \hat{\sigma}$ and $\hat{\sigma} \geq \frac{1}{\sqrt{d}}$, then $\|v\| \geq \frac{1}{12d\hat{\sigma}}$ with probability at least $1 - \delta$.*

Proposition 3. *If $\sigma \geq 20\hat{\sigma} \geq \frac{1}{\sqrt{d}}$, then $\|v\| \leq \frac{1}{\sqrt{148}d\hat{\sigma}}$ with probability at least $1 - \delta$.*

Propositions 1 and 2 guarantees that if the estimated upper bound is correct ($\sigma \leq \hat{\sigma}$), then the algorithm outputs an accurate estimation of μ; Proposition 3 guarantees the algorithm declares failure if the estimated upper bound is too small ($\hat{\sigma} \leq \frac{1}{20}\sigma$).

Intuitively, the average of $y_i x_i$ $(i = 1, \ldots, m)$ points to a direction similar to μ because of the symmetry of distributions of both x and noise: the projection of yx onto all directions perpendicular to μ is distributed symmetrically around 0 and thus has mean 0. The projection of yx onto μ has non-negative mean since the label y is correct (i.e., $yv^\top x \geq 0$) with probability at least $\frac{1}{2}$, and the projection is larger if the noise of y is smaller. Consequently, the scale of the average can be used as an indicator of the noise level. The Propositions can be formally proved by applying concentration inequalities on each direction.

Notation. *Denote by $\mathbb{S}^{d-1}$ the unit sphere $\{x \in \mathbb{R}^d : x^\top x = 1\}$. For any vector $X \in \mathbb{R}^d$, denote by $X^{(i)}$ the i-th coordinate of X. Define $Z_i = Y_i X_i$ for $i = 1, 2, \ldots, m$.*

We need following facts.

Fact 1. $\Pr(w^\top x \geq 0) = \Pr_{\xi \sim N(0,1)}(\xi \geq -\frac{w^{\star\top} x}{\sigma \|x\|})$. *Moreover, for $z \geq 0$, $\frac{1}{2} - \frac{z}{\sqrt{2\pi}\sigma} \leq \Pr_{\xi \sim N(0,1)}(\xi \geq \frac{z}{\sigma}) \leq \max(\frac{1}{6}, \frac{1}{2} - \frac{z}{3\sigma})$.*

Fact 2. *Let $B(x, y) = \int_0^1 (1-t)^{x-1} t^{y-1} dt$ be the Beta function. Then $\frac{2}{\sqrt{d-1}} \leq B(\frac{1}{2}, \frac{d}{2}) \leq \frac{\pi}{\sqrt{d}}$.*

Fact 3. *If $d \geq 2$, then $(1 - \frac{1}{d})^{\frac{d}{2}} \geq \frac{1}{2}$.*

Fact 4. *(Bernstein inequality) If i.i.d. random variables $X_1, \ldots, X_m$ satisfy $|X_i| \leq b$, $\mathbb{E}[X_i] = \mu$, and $\mathbb{E}[X_i^2] \leq r^2$, then with probability at least $1 - \delta$, $|\frac{1}{m}\sum_{i=1}^m X_i - \mu| \leq \sqrt{\frac{2r^2}{m}\log\frac{2}{\delta}} + \frac{2b}{3m}\log\frac{2}{\delta}$.*

Fact 5. *Suppose $(x_1, \ldots, x_d)$ is drawn from the uniform distribution over the unit sphere, then x_1 has a density function of $p(z) = \frac{(1-z^2)^{\frac{d-3}{2}}}{B(\frac{d-1}{2}, \frac{1}{2})}$.*

Without loss of generality, assume $\mu = (1, 0, 0, \ldots, 0)$.
Following three lemmas give concentration of v.

Lemma 1. *For any $k = 2, 3, \ldots, d$, with probability at least $1 - \delta$, $|v^{(k)}| \leq 2\sqrt{\frac{2}{md}\log\frac{2}{\delta}}$.*

Proof. For $k = 2, 3, \ldots, d$, $Z_1^{(k)}, Z_2^{(k)}, \ldots, Z_m^{(k)}$ are i.i.d. random variables bounded by 1. By symmetry, $\mathbb{E}[Z_1^{(k)}] = 0$. $\mathbb{E}[(Z_1^{(k)})^2] = \mathbb{E}[(X^{(i)})^2] = 2\int_0^1 z^2 \frac{(1-z^2)^{\frac{d-3}{2}}}{B(\frac{d-1}{2}, \frac{1}{2})} dz = \frac{1}{B(\frac{d-1}{2}, \frac{1}{2})}\int_0^1 t^{\frac{1}{2}}(1-t)^{\frac{d-3}{2}} dt = \frac{B(\frac{d-1}{2}, \frac{3}{2})}{B(\frac{d-1}{2}, \frac{1}{2})} = \frac{1}{d}$. By the Bernstein inequality, with probability at least $1-\delta$, $|\frac{1}{m}\sum Z_i^{(k)}| \leq \sqrt{\frac{2}{md}\log\frac{2}{\delta}} + \frac{2}{3m}\log\frac{2}{\delta} \leq 2\sqrt{\frac{2}{md}\log\frac{2}{\delta}}$. $\square$

Lemma 2. *With probability at least $1-\delta$, $v^{(1)} \geq \frac{1}{3\pi\sqrt{d}}\min(1, \frac{1}{\sigma\sqrt{d}}) - \sqrt{\frac{1}{2m}\log\frac{1}{\delta}}$.*

Proof. $Z^{(1)}_1, Z^{(1)}_2, \dots, Z^{(1)}_m$ are i.i.d. random variables bounded by 1. Their mean can be lower-bounded as follows.

For for any $0 \le a \le 1$, due to the noise setting, $\mathbb{E}[Y \mid X^{(1)} = a] \ge 0$ and $\mathbb{E}[Y \mid X^{(1)} = -a] \le 0$, so $\mathbb{E}[YX^{(1)} \mid X^{(1)} = a] + \mathbb{E}[YX^{(1)} \mid X^{(1)} = -a] \ge 0$. Consequently we have

$$\begin{aligned}
\mathbb{E}[YX^{(1)}] &\ge \mathbb{E}[YX^{(1)}\mathbb{1}[|X^{(1)}| \ge \frac{1}{\sqrt{d}}]] \\
&= \mathbb{E}[\mathbb{E}[Y \mid X^{(1)}]X^{(1)}\mathbb{1}[|X^{(1)}| \ge \frac{1}{\sqrt{d}}]] \\
&= \mathbb{E}[(1 - 2\Pr[Y = -1 \mid X^{(1)}])X^{(1)}\mathbb{1}[X^{(1)} \ge \frac{1}{\sqrt{d}}]] \\
&\ge (1 - 2 \Pr_{\xi \sim N(0,1)}(\xi \ge \frac{1}{\sigma\sqrt{d}}))\mathbb{E}[X^{(1)}\mathbb{1}[X^{(1)} \ge \frac{1}{\sqrt{d}}]]
\end{aligned}$$

Now, $\Pr_{\xi \sim N(0,1)}(\xi \ge \frac{1}{\sigma\sqrt{d}}) \le \max(\frac{1}{6}, \frac{1}{2} - \frac{1}{3\sigma\sqrt{d}})$. Besides, $\mathbb{E}[X^{(1)}\mathbb{1}[X^{(1)} \ge \frac{1}{\sqrt{d}}]] = \int_{\frac{1}{\sqrt{d}}}^{1} \frac{z(1-z^2)^{\frac{d-3}{2}}}{B(\frac{d-1}{2}, \frac{1}{2})} dz = \frac{(1-\frac{1}{d})^{\frac{d-1}{2}}}{(d-1)B(\frac{d-1}{2}, \frac{1}{2})} \ge \frac{1}{2(d-1)\frac{\pi}{\sqrt{d-1}}} \ge \frac{1}{2\pi\sqrt{d}}$ where the first inequality follows by Fact 2 and 3. Thus, $\mathbb{E}[YX^{(1)}] \ge \frac{1}{3\pi\sqrt{d}} \min(1, \frac{1}{\sigma\sqrt{d}})$.

By the Chernoff bound, with probability at least $1 - \delta$, $v^{(1)} = \frac{1}{m}\sum Z^{(1)}_i \ge \frac{1}{3\pi\sqrt{d}} \min(1, \frac{1}{\sigma\sqrt{d}}) - \sqrt{\frac{1}{2m}\log\frac{1}{\delta}}$. ☐

Lemma 3. *With probability at least* $1 - \delta$, $v^{(1)} \le \frac{2}{\sqrt{2\pi}d\sigma} + \sqrt{\frac{1}{2m}\log\frac{1}{\delta}}$.

Proof. We first give an upper bound of $\mathbb{E}[YX^{(1)}] = 2\int_0^1 z\frac{(1-z^2)^{\frac{d-3}{2}}}{B(\frac{d-1}{2}, \frac{1}{2})}(1 - 2\Pr_{\xi \sim N(0,1)}(\xi \ge \frac{z}{\sigma}))dz$. By Fact 1, $1 - 2\Pr_{\xi \sim N(0,1)}(\xi \ge \frac{z}{\sigma}) \le \frac{2z}{\sqrt{2\pi}\sigma}$, so we have

$$\begin{aligned}
\mathbb{E}[YX^{(1)}] &\le 2\int_0^1 z\frac{(1-z^2)^{\frac{d-3}{2}}}{B(\frac{d-1}{2}, \frac{1}{2})}\frac{2z}{\sqrt{2\pi}\sigma}dz \\
&= \frac{4}{\sqrt{2\pi}\sigma B(\frac{d-1}{2}, \frac{1}{2})}\int_0^1 z^2(1-z^2)^{\frac{d-3}{2}}dz \\
&= \frac{2B(\frac{d-1}{2}, \frac{3}{2})}{\sqrt{2\pi}\sigma B(\frac{d-1}{2}, \frac{1}{2})} \\
&= \frac{2}{\sqrt{2\pi}d\sigma}
\end{aligned}$$

The conclusion follows by the Chernoff bound. ☐

Now we present the proofs for the propositions.

Proof. (of Proposition 1) Since $\|\hat{w}-\mu\|^2 = \|\hat{w}\|^2+\|\mu\|^2-2\mu^\top\hat{w} = 2(1-\mu^\top\hat{w}) = 2(1-\frac{v^{(1)}}{\|v\|})$, to prove $\|\hat{w}-\mu\| \le \varepsilon$, it suffices to show $\frac{v^{(1)}}{\|v\|} \ge 1-\frac{\varepsilon^2}{2}$. Note that $1-(\frac{v^{(1)}}{\|v\|})^2 = \frac{1}{1+\frac{(v^{(1)})^2}{\sum_{k=2}^d (v^{(k)})^2}}$, and $1-(1-\frac{\varepsilon^2}{2})^2 = \varepsilon^2-\frac{\varepsilon^4}{4} \ge \frac{\varepsilon^2}{2} \ge \frac{1}{1+\frac{2}{\varepsilon^2}}$, so it suffices to show $\frac{(v^{(1)})^2}{\sum_{k=2}^d (v^{(k)})^2} \ge \frac{2}{\varepsilon^2}$ with probability at least $1-\delta$.

Now, by Lemma 1 and 2 and a union bound, with probability at least $1-\delta$,

$$\frac{(v^{(1)})^2}{\sum_{k=2}^d (v^{(k)})^2} \ge \frac{(\frac{1}{3\pi\sqrt{d}}\min(1,\frac{1}{\sigma\sqrt{d}})-\sqrt{\frac{1}{2m}\log\frac{d}{\delta}})^2}{4(d-1)\frac{2}{md}\log\frac{2d}{\delta}},$$

which is at least $\frac{2}{\varepsilon^2}$ for our setting of $m = \frac{(15\pi)^2}{\varepsilon^2} d\max(1,d\hat{\sigma}^2)\log\frac{2d}{\delta}$. ◻

Proof. (of Proposition 2) By Lemma 2, with probability at least $1-\delta$, $v^{(1)} = \frac{1}{m}\sum Z_i^{(1)} \ge \frac{1}{3\pi\sqrt{d}}\min(1,\frac{1}{\sigma\sqrt{d}}) - \sqrt{\frac{1}{2m}\log\frac{1}{\delta}} \ge \frac{1}{12\sqrt{d}}\min(1,\frac{1}{\hat{\sigma}\sqrt{d}})$, which implies $\|v\| \ge v^{(1)} \ge \frac{1}{12\sqrt{d}}\min(1,\frac{1}{\hat{\sigma}\sqrt{d}})$. ◻

Proof. (of Proposition 3) By Lemma 1 and 3 and a union bound, with probability at least $1-\delta$,

$$\begin{aligned}
\|v\|^2 &= \sum_{k=1}^d (v^{(k)})^2 \\
&\le 4(d-1)\frac{2}{md}\log\frac{2d}{\delta} + (\frac{2}{\sqrt{2\pi}d\sigma} + \sqrt{\frac{1}{2m}\log\frac{1}{\delta}})^2 \\
&\le \frac{1}{148d^2\hat{\sigma}^2}.
\end{aligned}$$

◻

A.3 Noisy Labels for the Continuous Case

In the continuous, the model extraction problem becomes a regression problem. In [18] the authors consider passive linear regression with squared loss and provide an algorithm that achieves nearly optimal convergence rate $\mathbb{E}[X^\top\hat{w}-Y] - \mathbb{E}[X^\top w^\star - Y] = \tilde{O}\left(\frac{C}{n}\right)$ where the constant C depends on the covariance matrix of X and the error of the optimal linear model $w^\star$. In [57] the authors point out that unlike in the classification case, the $O(\frac{1}{n})$ cannot be improved by active learning, but it provides an algorithm under a stream-based querying model (in fact, they assume the algorithm can draw X from any distribution, which can be implemented by rejection sampling with stream-based querying model) that achieves a learning rate with a better constant factor C. Authors in [26] consider active learning for maximum likelihood estimation (MLE) under the assumption that the model is well-specified ($P(Y|X)$ is given by a model in the model class) and that the Fisher information matrix does not depend on label

y (this assumption holds for linear regression and generalized linear models). It shows that a two-stage algorithm achieves a nearly optimal convergence rate.

The main difficulties of computationally efficient active learning for classification arise because of two factors: (1) how to efficiently find a classifier with the minimum classification error rate; (2) how to select examples for labeling. For (1), it has been shown that optimizing the classification error rate (0-1 loss) with noise is hard in general, and computational efficient solutions with theoretical guarantees are only known under some assumptions of the hypothesis space and noise conditions (for example [16, 39, 72]). For (2), most existing active learning algorithms maintain a candidate set of classifiers either explicitly [32] or implicitly [16, 20, 72], and the noise tolerance is achieved by repetitive querying as in Proposition 1 or a carefully designed sampling schedule to guarantee that the candidate set is "correctly shrunk" with high probability [20, 32, 72]. For regression, most loss functions (for example the squared error, negative log likelihood) are convex, and thus can be optimized efficiently. The labeling strategies in regression are also different: instead of maintaining candidate sets, active regression algorithms [26, 57] often first find a good sampling distribution that optimize some statistics of the covariance matrix and then draw labeled samples from this distribution. Such strategies tolerates noise naturally and de-noising strategies like repetitive querying are not necessary.

A.4 Additional Results

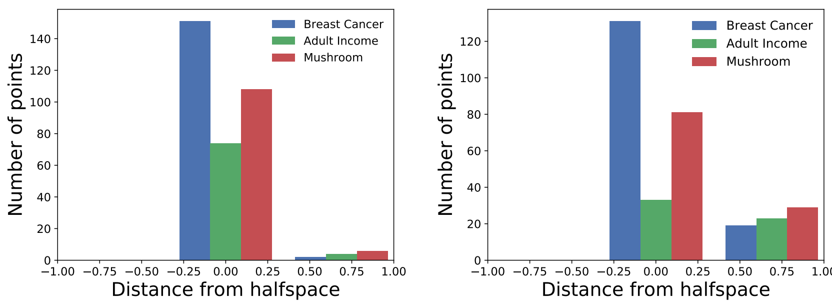
A.4.1 Alternate Stopping Criterion

We investigated if measuring the model's stability over N iterations results in acceptable extraction attacks. Here, we define model stability as the oscillation between the approximation learned at iteration i and at iteration $i+1$. Formally, stability can be defined as $\mathcal{S}_i = ||w_i - w_{i+1}||_2$. Our approach checks if $\mathcal{S}_i \leq \tau$ for $i = 1, \cdots, N$ and terminates execution if the condition is satisfied. We observe that this approach fails for the algorithm proposed by Chen *et al.* [16], as the approximation produced at each iteration differs greatly from the approximation produced in the preceding iteration. The results for the algorithm proposed by Alabdulmohsin *et al.* [4] can be found Table 4.

	N=10		**N=15**		**N=20**		**Baseline**
Dataset	**Queries**	$\hat{\varepsilon}$	**Queries**	$\hat{\varepsilon}$	**Queries**	$\hat{\varepsilon}$	$\varepsilon = 0.001$
Breast Cancer	241	0.0047	247	0.0034	252	0.0031	300
Adult Income	117	0.0019	122	0.0015	127	0.0012	135
Digits	493	0.0077	498	0.0075	503	0.0073	700
Wine	120	0.0016	125	0.0014	130	0.0012	135

Table 4: Model stability results in nominal savings at the expense of a small increase in geometric error ($\hat{\varepsilon}$). The trends are the same for other values of ε.

A.4.2 A Direction For Defense?



(a) Version Space Approximation (b) DC^2

Figure 7: Distance of the instances synthesized by (a) version space approximation algorithm, and (b) dimension coupling algorithm from optimal halfspace.

Recall from earlier discussion that QS active learning algorithms are capable of generating points de novo. It is conceivable that these points are not generated from the distribution from which the training data is sampled from. To this end, we verified if these distributions are indeed different using the Hotelling's T^2 test, specifically for the algorithms proposed by Alabdulmohsin *et al.* [4] and Chen *et al.* [16] under the null hypothesis that the distributions are the same (refer Table 5 and Table 6). We observe that this QS active learning algorithm indeed produces points that are not from the underlying natural distribution. While discarding points that can not be sampled from the training distribution may seem as a tempting defense strategy, it is conceivable that certain real world tasks may query MLaaS providers with outlier points. Further analysis is required to determine how this strategy may effectively be used to defend against model extraction.

Dataset	**t-value**	**n-1**	**p-value**	**Reject Null?**
Breast Cancer	14.24	198	3.70×10^{-32}	✓
Adult Income	9.71	92	9.22×10^{-16}	✓
Mushroom	22.16	599	5.92×10^{-80}	✓

Table 5: Results of the Hotelling T^2 test for multivariate distributions, for n samples. It is observed that the data-points generated by the **version space approximation** algorithm do not lie in the natural distribution underlined by samples from the training data.

Dataset	**t-value**	**n-1**	**p-value**	**Reject Null?**
Breast Cancer	319.27	322	0	✓
Adult Income	467.43	133	9.64×10^{-216}	✓
Mushroom	65.74	222	1.58×10^{-147}	✓

Table 6: Results of the Hotelling T^2 test for multivariate distributions, for n samples. It is observed that the data-points generated by the **DC^2 algorithm** do not lie in the natural distribution underlined by samples from the training data.