2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

Killing Four Birds with one Gaussian Process: The Relation between different Test-Time Attacks

Kathrin Grosse
CISPA
Saarland Informatics Campus

Michael T. Smith
Department of Computer Science
University of Sheffield

Michael Backes
CISPA Helmholtz Center
for Information Security

***Abstract*—In machine learning (ML) security, attacks like evasion, model stealing or membership inference are generally studied in individually. Previous work has also shown a relationship between some attacks and decision function curvature of the targeted model. Consequently, we study an ML model allowing direct control over the decision surface curvature: Gaussian Process Classifiers (GPCs). For evasion, we find that changing GPC's curvature to be robust against one attack algorithm boils down to enabling a different norm or attack algorithm to succeed. This is backed up by our formal analysis showing that static security guarantees are opposed to learning. Concerning intellectual property, we show formally that lazy learning does not necessarily leak all information when applied. In practice, often a seemingly secure curvature can be found. For example, we are able to secure GPC against empirical membership inference by proper configuration. In this configuration, however, the GPC's hyper-parameters are leaked, e.g. model reverse engineering succeeds. We conclude that attacks on classification should not be studied in isolation, but in relation to each other.**

I. Introduction

Security researchers study a plethora of attacks on machine learning (ML). In general, each attack is studied individually. For example, *Evasion attacks*, or *adversarial examples*, are small perturbations added to a sample, which is subsequently misclassified. Examples for targeted systems include, but are not limited, to Malware detectors [1], [2], vision for autonomous driving that misclassifies traffic signs [3], and robot visual systems [4]. Securing models against evasion was shown to lead to an arms race [5], [6].

Other attacks harm the intellectual property of the model owner. In *model stealing* [7], [8], the attacker copies a model's functionality without consent of the owner. In *model reverse engineering* [9], hyper parameters of the model are inferred illegitimately. Another attack retrieves the data that was used to train the classifier, called *membership inference* [10], [11]. This corresponds to a privacy breach for the subjects in the data, and/or a financial loss for the owner of the data.

A relationship between decision function curvature and membership inference was shown in [10]. An analogous relationship for evasion has been found in linear models like support vector machines [12] and used for mitigations in deep neural networks [13], [14]. These findings raise several questions. Are all test-time attacks related to decision function curvature? Do changes in curvature have the same effect on all attacks? To answer these questions, we need a model where curvature can be configured. Such a model are Gaussian processes (GPs): Choosing a long *lengthscale* before training yields for example a GP with a flat decision surface.

Studying GP yields two more benefits. GP are often applied in medical settings [15], [16], [17]. Risk assessment for leaked data or learned parameters is thus crucial. Furthermore, GPs provide the means for a rigorous analysis: After training, a GP yields a closed form expression, where classification depends directly on both parameters learned and used training data.

Contributions. Our formal analysis confirms vulnerability towards evasion at test time once the GP has learned. Due to its mathematical form, GP allows to analytically compute the lengthscale iff the training data is known and only one lengthscale used. We further conduct a broad empirical study of vulnerability on six data sets, focusing on decision function curvature. To this end, we introduce two model reverse engineering attacks, one for GP's lengthscale, one for the kernel. Decision function curvature often only changes the kind of attack that succeeds. In evasion, highly optimized attacks tend to fool a steep curvature. This steep curvature also leaks the data. On the other hand, one-step evasion attacks are more effective on flat curvature. This flat curvature also leaks parameters like the lengthscale. In contrast, leakage of the kernel occurs at any lengthscale. We conclude that attacks on classification should not be studied in isolation: mitigating one attack might just enable a different attack.

II. Related Work

To the best of our knowledge, few works have studied the relationship between different attacks. Most works focus on deep learning, and on at most two attacks. For example Suciu et al. [18] study evasion and training time attacks jointly. Song and Mittal [19] show that neural networks that are robust against evasion are more vulnerable against membership inference. Along these lines, there are defenses taking into account several attacks on deep learning [20], [21]. We instead focus on an in depth study of the *relationship* between several attacks, and are unaware of any similar work.

Most formal works of test-time attacks focus on evasion [22], [23], [24]. Our formal evasion analysis for GP is in the finite sample setting. Wang et al. [22] instead give an analysis in the infinite sample limit on k-nearest-neighbors. A formal approach related to membership inference is the recent work on differential privacy for GP (see for example [25]). On the other hand, empirical evasion security has been studied on

978-1-7281-8808-9/20/$31.00 ©2020 IEEE

2020 25th International Conference on Pattern Recognition (ICPR) | 978-1-7281-8808-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPR48806.2021.9413290

GP [26], [27], [28]. GPC also allows one to bound evasion vulnerability [29], [30]. We are not aware of any works studying model reverse engineering or model stealing on GPs.

III. Background

We introduce GP, give a short summary of adversarial learning and finally describe our threat model.

A. Gaussian Process Classification

We use Gaussian Process Classification (GPC) [31] for two classes using the Laplace approximation. The goal is to predict the labels Y_t for the test data points X_t accurately.

We specify k as covariance function or kernel and introduce GP regression (GPR). Assuming the data is produced by a GP,

$$\begin{bmatrix} Y_{\text{tr}} \\ Y_t \end{bmatrix} = \mathcal{N}\left(0, \begin{bmatrix} K_{\text{tr}} & K_{\text{tt}} \\ K_{\text{tt}}^\top & K_t \end{bmatrix}\right) , \tag{1}$$

where Y_{tr} are the training labels, K_{tr} is the covariance of the training data, K_t of the test data, and K_{tt} between test and training data. Having represented the data, we now review how to use this representation for predictions. As we use a Gaussian model, our predictions are Gaussian too, with a predictive mean and a predictive variance which we define now. At a given test point x', assuming a Gaussian likelihood function, the predictive mean y_{t}^* of the latent function on test data is

$$y_{\text{t}}^* = K_{x'}^\top K_{\text{tr}}^{-1} Y_{\text{tr}} , \tag{2}$$

where $K_{x'}^\top$ is the vector with the covariances from test point x' to each training point in X_{tr}. K_{tr}^{-1} is the inverse of K_{tr}.

Classification, unlike regression, has binary outputs and requires a different likelihood and associated link function. We can approximate this in a variety of ways. One of the simplest is the Laplace approximation whose simplicity allows us to formally analyze the GPC. Finally, whenever we write GP, we refer to properties that both GPC and GPR share.

B. Adversarial Machine Learning at Test Time

We review all test time attacks dealt with in this paper: evasion, model reverse engineering, membership inference, and conclude with model stealing.

In **evasion**, the attacker computes a small perturbation δ for a trained classifier $f(X_t) = Y_t$ and a sample x such that

$$\min \delta : f(x) \neq f(x + \delta) \tag{3}$$

the minimal δ changes the classification of x. Many algorithms exist to craft adversarial examples. We now recap the algorithms used in our evaluation. The fast gradient sign method (**FGSM**) [32] is an untargeted one-step attack. A step adds the gradient of the model's loss w.r.t. the input x' to the original sample. The step size is parametrized by ϵ. The Jacobian-based saliency map approach (**JSMA**) [33] picks iteratively a pixel for perturbation that maximizes the output for the target class and minimizes the output for all other classes.

Finally, the $\mathbf{L_x}$ attacks [34] formulate evasion as an iterative optimization problem. The basis, or L_2 attack is formalized as the following optimization problem

$$\min_{\delta} \parallel 0.5(\tanh(\delta) + 1) + x \parallel_2 + sg(0.5(\tanh(\delta) + 1)) ,$$

TABLE I: Attackers knowledge according to the FAIL model. The symbol ✓ denotes 'known' or 'is altered', X the opposite.

Attacker	F	A	I	L
Evasion	✓	X	X	X
Model Extraction l/lengthscale	X	✓	X	✓
Model Extraction k/kernel	X	✓	X	X
Membership Inference	X	✓	✓	✓
Model Stealing	X	✓	X	X

where $\tanh$ ensures that the box-constraint to enforce that no feature is set to higher values than in benign data. Term s trades-off the constraint and function g. This function represents how confidently the network f misclassifies $x+\delta$. Other variants of this attack minimize the L_0 or L_∞ norms [34].

We now review attacks harming intellectual property (IP).

In **model reverse engineering**, given a trained classifier with black box access, the attacker tries to infer hyperparameters of the model using specifically crafted queries [9]. For GPs, possible parameters to be targeted are for example the lengthscale(s) and the chosen covariance function.

Membership inference describes an attack which aims to learn whether or not some samples were used to train the model [35], [10], [11]. Such attacks are generally run in a black box setting, and exploit differences in confidence for trained and unseen data. In contrast to deep learning, a GP is not forced to be overly confident on training data, so these attacks are non-trivial. In our evaluation, we use both confidence (predictive mean) and the predictive variance to deduce this information—a slight variation of known attacks.

A **model stealing** attack aims to reproduce the full black-box model [8], [7]. For GPs, this amounts to finding out all parameters learned during training and which training data was used, as this information defines the GP completely. For GPs, this attack is a combination of the previous two attacks.

C. Threat Model

We specify the different adversaries of our empirical study. In the FAIL [18] model, F denotes the attacker's knowledge about the features. A denotes knowledge about the algorithm applied and I about the training data. L summarizes whether changes to the data by the attacker are constrained. A succinct overview for each attack is given in Table I.

Evasion. Our attacker knows and changes all features, but is oblivious about the training data and the algorithm.

Model reverse engineering (l). The attacker only knows a GPC with an RBF kernel is used. The data knowledge varies from black-box to white box, without modifying samples.

Model reverse engineering (k). The second attacker only knows GPC is applied. Yet, she uses the zero and the ones vector as input, and is thus not constrained on features.

Membership inference. We assume a worst case scenario, where the attacker obtained a large fraction of data labeled as part of the training set. The attack is not tailored for the learning algorithm, and does not alter the input.

Model stealing. In our setting, model stealing on a GP can be seen as a combination of the previous two attacks.

IV. Formal Analysis of Vulnerability

We take advantage that a GP allows a formal analysis. First, we show that learning or generalization enables evasion vulnerability on GP. We then study the interplay of model reverse engineering, membership inference, and model stealing.

A. Evasion Attacks

We first define a classifier that cannot be fooled by an adversarial example. In the following, we show that a classifier fulfilling this definition, and hence a static security guarantee, is opposed to learning. We briefly define *rejection* of a classifier. A classifier can *reject* a sample, in the sense that it does not assign the given sample to any predefined class.

To define a secure classifier, we chose a covariance with compact support [36]: as the distance from the training data increases, it reaches 0. Furthermore, there is a ρ such that for all training points, iff point x' is in the closed ball $B(x_i, \rho)$ around a training point $x_i \in X_{tr}$ with radius ρ, then x' cannot be an example of another class than x_i's class y_i. In other words, all points in the ρ-ball around x are of the same class. We formalize the secure classifier

$$f(x') = \begin{cases} y_i & \text{iff } x' \in B(x_i, \rho) \\ \text{reject} & \text{otherwise} \end{cases}$$

that cannot be fooled: Changing a sample enough to be classified as a different class means to alter x' so much that $x' \in B(x_j, \rho)$ where $y_i \neq y_j$. Then, by our definition, x' is a valid instance of this other class and not an adversarial example. This secure classifier is equivalent to a GP given the following conditions: *First*, GP has a rejection option based on ρ. *Second*, writing $k(x_i, x_j)$ for the covariance between x_i and x_j, there is no point x' such that for two distinct x_i,$x_j \in X_{tr}$ both $k(x_i, x') > 0$ and $k(x_j, x') > 0$.

In other words, we require that GP is able to reject a sample. This can be achieved by setting a threshold on GP's similarity. Condition two states that the similarity between any two training points is zero, independent of their class. Such a GP, however, has as covariance matrix the identity matrix, as the similarity between any two points is zero. Such a covariance matrix does not allow any learning [37]. The details of this equivalence can be found in a long version of this paper. Assuming that the second condition does not hold, training points jointly influence classification and the GP generalizes.

Theorem 1. *Either GP's covariance K is similar to the identity matrix I, or $K \neq I$ and learning occurs. Then, GPR potentially classifies areas outside the ρ-balls. Hence, for a test point x' and its corresponding output p, $p > \rho$ or $p < -\rho$ although $k(x_i, x') < \rho$, where x_i is the closest training point.*

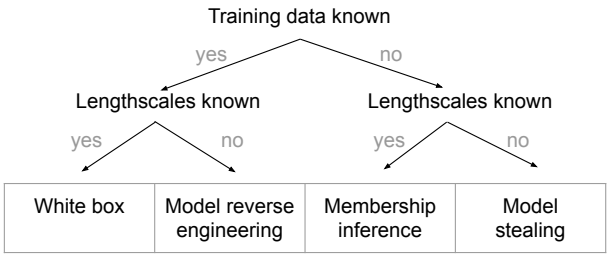


Fig. 1: The relationship of IP based attacks on GP models.

Proof. To be classified, we need a classification output $p > \rho$ or $p < -\rho$. We start with the first case, and write

$$p \leq \sum_i (\rho - \kappa_i) * [K^{-1}]_i * 1 \ , \tag{5}$$

where $[K^{-1}]_i$ is the sum over the inverted covariance matrix column corresponding to point i. Before inversion, this column contains the similarities between i and all other training points. So far, we have ignored that we need a test point to obtain this prediction. Without loss of generality, we pick x' which maximizes the sum under the restriction that x' is in none of the ρ-balls: hence $\rho - \kappa_i$, the covariance to ρ-ball$_i$ is κ_i.

There are two cases. In the first, $p \geq \rho$ and we classify outside the ρ-ball. In the second, $p = 0$ or $0 < p < \rho$. As we choose the maximal x', there are no other points for which $p > \rho$. Then GPR is still secure: no area outside the ρ-ball is classified, as the output is below the defined threshold. It remains to be shown, however, that there is no contradiction for the opposite class. We proceed analogously with an x'' that is chosen to minimize the sum. ■

We used in the proof that the minimal output of a point chosen to maximize the sum is zero. Analogously, the maximal value when minimizing the sum is zero as well. This holds due to the abating property of the kernel: As we move away from the data, eventually all similarities become zero, thus the sum is zero as well. We conclude that generalization enables test time attacks such as evasion or adversarial examples.

B. Attacks against Intellectual Property

As GPs are an instance of lazy learning, in general all training points and parameters are used during inference. Intuitively, this should ease extraction for the attacker. As we show here, this need not be the case. We briefly recap the attacker's goal in each attack. In **model reverse engineering**, she wants to obtain the lengthscale(s), in **membership inference** the full or partial training data, and in **model stealing** both lengthscale(s), and full training data. These attacks are strongly related for GPs, as visible in Figure 1.

We refresh how classification is computed in GPR (introduced in eq. (2)). The posterior mean y^* is given as

$$y_t^* = K_{x'}^T K_{\text{tr}}^{-1} Y_{\text{tr}} = \sum_i k(x' - X_i) * K_i^{-1} * Y_i \ , \tag{6}$$

where we iterate over the n training data points. The covariance metric k is parametrized using l and σ^2 when using the

RBF kernel. As lazy learning is used, one might suspect that we can simply *extract* the stored parameters and training data. For example, independent from the used kernel, we unfold this sum and add the observed output of a GP to obtain an equation system. For simplicity, assume that $Az = y_o$, where z refers to the parameter the attacker wants to retrieve, and y_o is the output observed from the targeted GP. Further A denotes the matrix specified in equation 6, without z.

The interested reader will have noticed, however, that this equation system solves for unknowns in the number of training points whereas we need an equation system solving along the features dimensionality. In terms of the above equation, we are actually interested in $A^T z = y_u$, where y_u is an output per feature (where feature and data point dimension are swapped, or X^T). Hence, y_u is not an output for any GP trained on X: it corresponds to a label per feature. In the original task, the features are "lost" in the Hilbert space of the kernel (covariance), and the attacker has no equation system since there is no y_u.

The existing equation system can only be used to determine the lengthscale iff there is only one global lengthscale set, and the GP has no other unknown parameter. Otherwise, the equation system is not properly specified, and no analytic computation is possible. We thus conclude that lazy-learning, albeit counter-intuitively, is not less privacy resilient that other classifiers. Instead, however, the attacker can take advantage that GPs are deterministic. A GP with the same parameters and data always yields the same output. In the following, empirical section we evaluate this type of attack.

C. Conclusion

GP, as it learns, is vulnerable to evasion attacks. Concerning IP-related attacks, we can exclude the possibility that the attacker analytically determines training data or lengthscales, with the exception of a single learned parameter for all dimensions (for example in a linear kernel).

V. Empirical Study of Vulnerability

We now describe our complementary empirical study. We start with the setting including data-sets, implementation, and parameters. Afterwards, we detail the results on evasion, model reverse engineering and membership inference.

A. Experimental Setting

We first describe the general setting. Specifics are given jointly with the corresponding attacks.

Data and implementation. Our study encompasses several data-sets, including security tasks such as Malware (Hidost [38], Drebin [39]) and Spam detection [40]. Additionally, we investigate fake banknote detection by [40], the MNIST benchmark data set [41], and the SVHN data set [42]. We use Python and GPy for the Gaussian Process approaches [43]. We show further information on the trained GPCs in Fig. II, such as the number of training samples and lengthscales used and achieved accuracies. To obtain adversarial examples, we use Tensorflow [44] and the Cleverhans library 1.0.0 [45] for DNN, and other public implementations [34], [27].

TABLE II: Number of samples used in training n, lengthscales l and accuracies (rejection if $y_t^* = 0$, written Acc_r).

		short l			long l		
Data-set	n	l	Acc_r	Acc	l	Acc_r	Acc
Hidost	500	.5	98.4	98.4	1.9	97.7	99.6
Drebin	750	.5	54.4	94	1.9	94.8	94.8
Spam	500	.3	92.6	91.7	5	92.7	90.2
Bank	500	.3	100	100	2	100	100
MNIST91	500	1	98.9	98.3	8	99.5	99.5
MNIST38	500	1	93.4	93.4	8	97.4	97.1
SVHN91	1500	8	85.4	88.5	16	83.8	87.6
SVHN10	1500	8	88.7	88.7	16	88.7	88.7

Parameter choices. We train our GPC using the RBF kernel with a predefined lengthscale. This GPC is fitted until convergence or for 100 iterations. For each task, we chose two lengthscales that achieve similar accuracy (see Table II). More details on how we determined the two used lengthscales can be found in a long version of this paper.

B. Evasion / Adversarial Examples

We expect that a GP with a long lengthscale misclassifies fewer adversarial examples: A larger perturbation δ is needed to cause the same change in the output.

Setting. To obtain adversarial examples independent of the specific curvature, we do not craft on the GPCs tested. We instead transfer FGSM, JSMA and $\mathbf{L_x}$ attacks from deep neural networks, linear SVM and a GPC substitute. Our intention is to study a wide range of attacks, including optimized, unoptimized, one-step and iterative attacks as well as different metrics (L_0, L_2, and L_∞). We summarize all attacks based on the Jacobian in JBM, sort FGSM according to ϵ and plot the $\mathbf{L_x}$ attacks according to the norm optimized (for example L_2 for the L_2-norm attack).

We compare how well the previously chosen lengthscales recover the correct class when facing adversarial examples. In our plots, a value above zero denotes that the shorter lengthscale classified more data correctly, where the numbers are difference in absolute percent. Below zero, a longer lengthscale (flat curvature) performed better.

Results. We plot the results of our experiments in Fig. 2. A short lengthscale generally classifies more adversarial examples as their original class. In particular on L_∞ attacks (with $\epsilon > 0.01$), a short lengthscale performs better. A long lengthscale is advantageous for optimized attacks like L_2.

We also investigate how lengthscale affects rejection, as our preliminary results show only a slight advantage for steep curvature GPs without rejection. In Fig. 3, a negative number denotes how much absolute percent the reject performs better compared to a classifier without reject. A positive number means that accuracy for rejection is worse. There is no difference in vulnerability to evasion for a long lengthscale. For a short lengthscale, the effect is positive or neutral, with

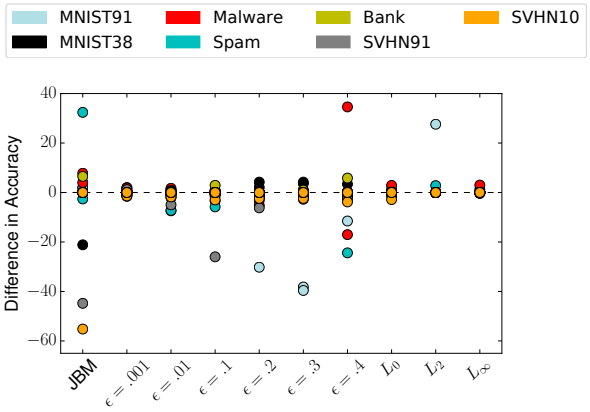


Fig. 2: Vulnerability and Curvature in GPC. Above zero denotes that more examples are correctly classified by a GPC with long l, below zero with short l.

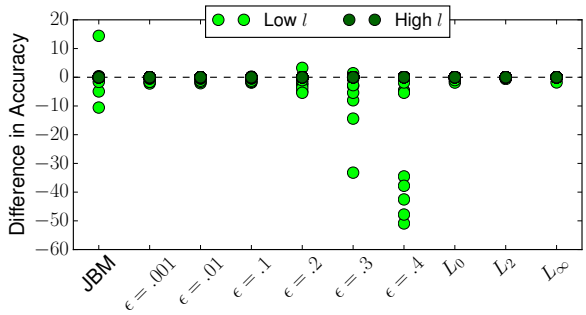


Fig. 3: Vulnerability, Lengthscale and rejection option in GPC. Above zero denotes that more examples are correctly classified or rejected by a GPC without a rejection option.

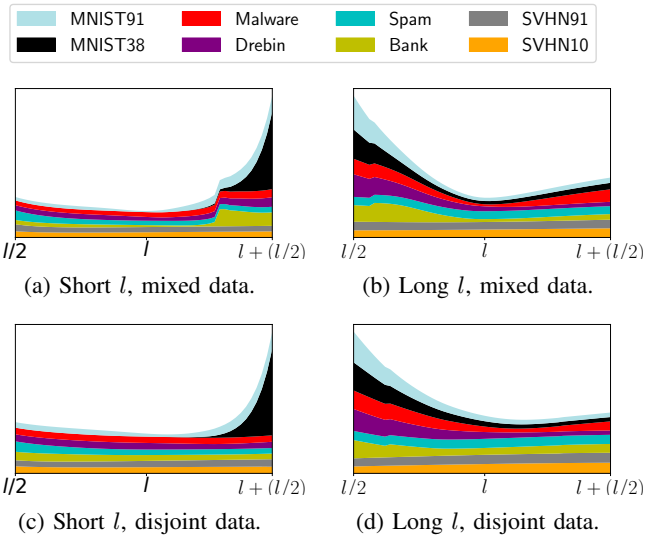
only two negative cases. These two cases stem from the highly imbalanced Hidost data set. By chance, the assignment of the forced classification was in favor of the larger class.

Conclusion. Only classifiers with steep decision functions benefit from rejection. We hypothesize that a short lengthscale allows for larger areas where the rejection area is actually used, whereas a long lengthscale leads to confident classification in areas where no benign data was seen.

C. Model Reverse Engineering

Model reverse engineering refers to the retrieval of hyper-parameters of the model. We introduce two new attacks to reverse engineer GP's lengthscale and kernel.

Setting (lengthscale). We pick the same lengthscales as before and evaluate whether the attacker is able to determine the lengthscale of a target GP. The attack is a binary search to obtain l. The distance between the outputs of two GPs shrink as the lengthscale chosen by the attacker, l_a, approaches the original lengthscale l. We evaluate three settings: Training GPC on the same data as the victim, mixed (half/half) and disjoint data. In each setting, we train 50 GPCs, starting with a lengthscale $l_{a=0} = l/2$ and increasing the lengthscale in



(a) Short l, mixed data. (b) Long l, mixed data.

(c) Short l, disjoint data. (d) Long l, disjoint data.

Fig. 4: Normalized, absolute differences in output for different data sets when binary searching a GP's lengthscale. x-axis is L_a; hence at l, lengthscales are equivalent.

50 steps of $(1/50)l$. We then compute the absolute difference between the outputs of the GPCs on hold out, unused test data.

Results (lengthscale). As GPs are deterministic, all distances decrease towards the original l when the training data is fully known. We thus omit plotting these results, and study the more interesting cases of mixed data.

For mixed data (upper plots of Fig. 4), the results are less clear. In general, distances decrease towards l. Given a long lengthscale, the distances are smallest around l. An exception are SVHN and Spam, where the distances remain constant for all l_as. On Drebin, the distances are smallest around $l + (l/2)$. The results vary for a short lengthscale: for some data sets (MNIST91, Bank) the distance is closest to l, For others (including SVHN and Malware), the smallest distance is $l/2$.

In case of disjoint data sets (bottom plots of Fig. 4), the results are even less pronounced. The distances slightly decrease towards the original lengthscale, yet the average minimum is at a lengthscale $> l$. In case of a short lengthscale, there are no differences at all. An exception are the two MNIST tasks, where again the minimum is $> l$.

In general, a lengthscale can be approximated using binary search. More concretely, the estimate is close when the original lengthscale is long: The difference to the original lengthscale is then between $0.006l$ and $0.008l$. This corresponds to wrongly estimating the largest lengthscale of SVHN by 1.28 (17.28 instead of 16.0) or the smallest (Bank) by 0.16 (estimating 2.16 instead of 2.0). For a short lengthscale, the estimate for MNIST91's lengthscale is around 1.04 instead of 1. For cases except MNIST91, the estimate is inaccurate or indeterminable.

Setting (kernel). The goal of the attacker is to determine the kernel used in a black-box GPC. We assume the victim uses one of the following kernels, RBF (with the same lengthscales as before), linear, or polynomial. We exclude the results

Fig. 5: Stealing the kernel of a GPC (columns), ✓denotes successful extraction. × denotes a failed attack, (✓) that the attack succeeded only in an easy-to-defeat variant. Some cases were not evaluated (/) as test accuracy was too low.

on Drebin with the linear and polynomial kernel, as their accuracies are close to a random guess.

Attack Description (kernel). An RBF-kernel will output close to zero far away from seen data. Hence, we input the target GPC a zero and ones sample and deduce an RBF-kernel is used if the output is close to 0.5. We also use a more extreme, easy to defeat variant of this attack where the given samples contain only features equivalent to ± 10. To preserve feature meaning, we could also compute an *unusual*, far away sample. Due to the diversity of our data-sets, we leave this variant for future work.

In case neither output is 0.5, we run a second round of queries. We assume a linear kernel is limited in its expressivity, and leads to less confidently classified data. We thus submit a batch of test points, and classify a kernel as polynomial (nonlinear) if the distribution of outputs is bi-modal with most values scattered around 0 and 1. We hence compute the mean of the values above and below 0.5. The threshold for the decision is that both means are further apart than 0.7. This threshold was determined on the additional credit data-set [40], which is otherwise not used in this evaluation.

We train different GPCs, each using a different kernel (RBF kernel with several learned lengthscales, linear, polynomial kernel). The attacker determines, with the above heuristics, the used kernel. Our results are depicted in Fig. 5.

Results (kernel). In the majority of cases, the attack succeeds independent of lengthscale or kernel used. In three of eight cases, the linear kernel is wrongly determined as nonlinear, indicating that it confidently classified the data against our expectation. In one case, on the spam data, the polynomial kernel is wrongly determined as RBF kernel. There are also some cases on the Bank and SVHN data-sets where the RBF kernel is only correctly predicted if we use the ± 10-filled samples. Otherwise, the attacker's classification is that the victim uses the polynomial kernel on bank, or the linear kernels on the SVHN tasks.

There are very few differences between using full test data, 500, 50, or as few as 10 samples for the linear/nonlinear query. Only on the Bank data-set the linear kernel was classified as polynomial kernel using 50 samples or less. All other results remained consistent.

Conclusion. Empirically, the lengthscale can be recovered easily if the training data is (partially) known. This relates to the GP being deterministic. Otherwise, the attacker can reasonably well approximate the lengthscale given that the targeted GPC has a long lengthscale. We hypothesize that the long lengthscale is easier to extract as it is less prone to small changes in the data distribution. The kernel is, instead, easy to deduce independent of the lengthscale. Our attack currently fails if the linear kernel fits the data well (MNIST, Malware) or the polynomial kernel's decision boundary passes the origin or the ones vector. With a long lengthscale, the RBF kernel (Bank, SVHN) outputs relatively large values even far away from the data. Yet, we find no absolute value for this to happen. Another natural defense to our attack are custom-based kernels. We leave this cases for future work, and conclude that our heuristic works well for the given data-sets and the kernel set $\{$RBF, linear, polynomial$\}$.

D. Membership Inference

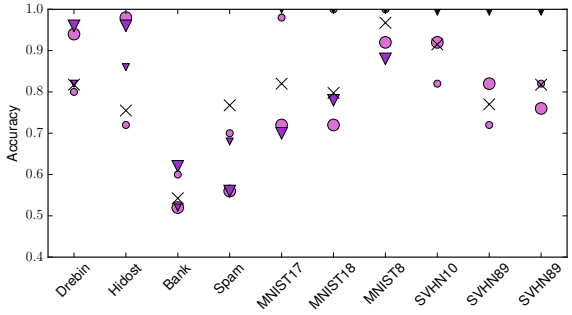
We investigate how good an attacker can determine which points were used in training. First, we study the general setting. Afterwards, we investigate particular settings influencing the attackers success: overfitting, distribution drift, and sparsity.

To study a worst case scenario, the attacker has an oracle that labels a large fraction of the training data as such. This attacker is slightly stronger than the shadow models used in [35]. The attacker trains a fresh classifier that predicts membership for unseen data points.

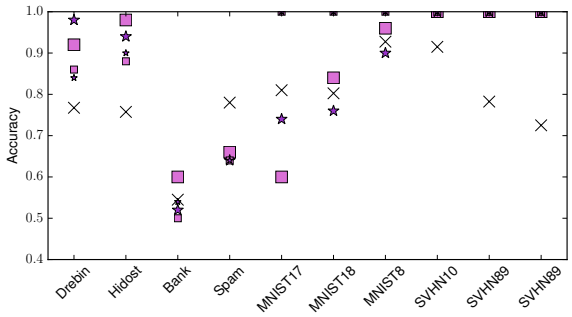
Setting. The target GPs are trained using the same lengthscales as before. We then build a data-set using the output of the GPs and membership labels indicating if a data point was used in training. The data-set is split randomly in test data (50 points) and training data (the remainder). The training data is used to train a fresh classifier. We tested DNN, decision trees, random forests and AdaBoost classifiers. We apply random forest classifiers, as they performed consistently best. We report accuracy and random guess accuracy on the test data.

Results. We train the random forests on predictive mean (dots), variance (triangle) (Fig. 6a), mean and variance (squares), or the unnormalized, latent mean (stars) (Fig. 6b). Overall, using only the predictive mean and a long lengthscale (larger markers), no data set is vulnerable, with the exception of the two Malware data sets. For mean and variance and latent mean settings, the attacker succeeds in both cases on all SVHN tasks or when using a small lengthscale, with the exception of non-vision tasks. The attacker is also successful on the Malware data sets with a long lengthscale.

On the bank and spam data, the attack is never successful. In general, a shorter lengthscale is more vulnerable. On the Malware data sets, the inverse holds: here, a short lengthscale benefits the defender. Before we focus on these cases, however,



(a) Classifier trained on variance (triangles) or mean (dots).



(b) Training on mean and variance (square) or latent mean (star).

Fig. 6: Lengthscale and membership inference on GPC. Bigger symbols denote a long, small symbols a short lengthscale of targeted GPC. x denotes random guess.

we investigate what enables the attacks on the SVHN data and why a short lengthscale is beneficial for the adversary.

Overfitting, distribution drift, and sparsity. We compare training and test accuracies to measure **overfitting**. On the bank data, training and test accuracy are both 100%. On all other data-sets, the difference between test and train accuracy is smaller for a long lengthscale. Hence, slight overfitting occurs at short lengthscales, and enables membership inference.

To analyze **distribution drift**, we measure the standard deviation over the distances between training and test data. As GP adapts the similarity during training, we expect the test data to cause larger variance in the distance if the data is distributed differently. All SVHN and the MNIST8 settings with a small lengthscale show a variance two magnitudes larger between test and training data than among either. Thus, the attack was enabled as training and test data were different from the perspective of GPC. This might imply that the model is not expressive enough to model the data in detail.

Two cases of successful membership inference are left unexplained: the Malware data-sets, Hidost and Drebin. We suspect that **sparsity** causes the vulnerability. The average percentage of features > 0 on the full data-set is $< 0.001\% \pm 0.0006$ on Drebin and $\sim 12\% \pm 3.8$ on Hidost. Next is MNIST (1 vs 7 with around $14\% \pm 4.1$, 1 vs 8 with $\sim 16\% \pm 6.5$ and 8 with $\sim 18\% \pm 5.2$). All other data-sets exhibit less sparsity ($> 20\%$, Spam) or well above 70% (all remaining data-sets).

The difference in sparsity between Hidost and MNIST is small, yet the discrepancy to robust data-sets (Bank, Spam) is large. To account for sparsity, we apply a GPC using inducing variables (GPy's sparse GPC). Such a GPC also optimizes over the training points: the training data is then not directly stored.

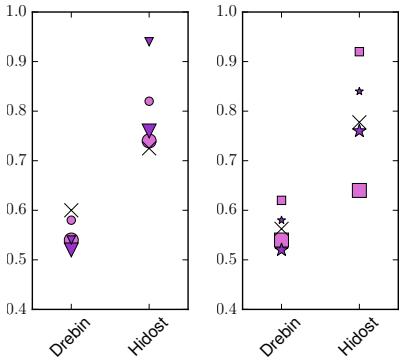


Fig. 7: Accuracy of membership inference on a sparse GPC. Bigger symbols denote a long, small symbols a short lengthscale on targeted GPC. x denotes random guess. Left plot: Classifier trained on variance (triangles) or mean (dots). Right: Training on mean and variance (square) or latent mean (star).

In Fig. 7, we investigate the same settings from the previous study. The attacker's accuracy is now on all settings close to a random guess, with the exception of a short lengthscale for Hidost on mean or variance, latent mean, or mean and variance. For Drebin, a very small improvement over random guess occurs when a short lengthscale is used and the attacker accesses the GP's predictive mean and variance.

Conclusion. Even under a strong attacker, membership inference is not successful when there is no distribution drift, overfitting is properly taken care of, or a sparse GP with a long lengthscale is applied. Robustness of GPs towards membership inference is somewhat expected, as a GP is not required to be overly confident on training data. The effect of the lengthscale is also intuitive. A short lengthscale allows each training point only local influence, easing inference about membership. With a long lengthscale, each point influences a large area, making it harder to locate the exact training point.

VI. Conclusion

We investigated the security of GPs at test time towards evasion, model reverse engineering, membership inference, and model stealing. We conclude that attack vectors on classification should not be seen in isolation, as a mitigation towards one attack might enable or ease another attack.

Formally, we show that evasion is enabled by learning, and any learned GP is vulnerable. Against possible intuition, lazy learning is not per se more vulnerable towards IP attacks. Still, a re-computation of the lengthscale is possible if kernel and the training data are fully known. Yet, no further parameters can be analytically retrieved from given output.

We also study empirical vulnerability, and leveraged the property of a GP to fit a model with a predefined decision

curvature. Our study encompasses six data-sets. Summarizing, a short lengthscale leaks the data, and is vulnerable to optimized evasion attacks. A long lengthscale leaks the parameters of the GP, and is vulnerable to one-step attacks with large ϵ. The kernel can be determined independent of the used lengthscale. We conclude that attacks on classification should not be studied in isolation, but in relation to each other.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers, Christian Rossow, Thomas A. Trost, and David Pfaff for their helpful feedback. This work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA) (FKZ: 16KIS0753). This work has further been supported by the Engineering and Physical Research Council (EPSRC) Research Project EP/N014162/1.

REFERENCES

[1] P. Laskov *et al.*, "Practical evasion of a learning-based classifier: A case study," in *2014 IEEE S&P*, 2014, pp. 197–211.

[2] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[3] C. Sitawarin, A. Nitin Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "DARTS: Deceiving Autonomous Cars with Toxic Signs," *ArXiv e-prints*, Feb. 2018.

[4] M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera, and F. Roli, "Is deep learning safe for robot vision? adversarial examples against the icub humanoid," in *ICCV Workshops 2017*, pp. 751–759.

[5] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," pp. 3–14, 2017.

[6] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," pp. 274–283, 2018.

[7] N. Papernot, P. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *CoRR*, vol. abs/1605.07277, 2016.

[8] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *USENIX*, 2016.

[9] S. J. Oh, M. Augustin, B. Schiele, and M. Fritz, "Towards reverse-engineering black-box neural networks," in *ICLR*, 2018.

[10] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *ACM SIGSAC CCS*, 2017, pp. 603–618.

[11] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models," in *NDSS*, 2019.

[12] P. Russu, A. Demontis, B. Biggio, G. Fumera, and F. Roli, "Secure kernel machines against evasion attacks," in *AISec@CCS*. ACM, 2016.

[13] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *NIPS*, 2017.

[14] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *ICLR*, 2018.

[15] T. Chen, J. Morris, and E. Martin, "Gaussian process regression for multivariate spectroscopic calibration," *Chemometrics and Intelligent Laboratory Systems*, vol. 87, no. 1, pp. 59 – 71, 2007.

[16] M. D. Stevenson, J. Oakley, and J. B. Chilcott, "Gaussian process modeling in conjunction with individual patient simulation modeling: A case study describing the calculation of cost-effectiveness ratios for the treatment of established osteoporosis," *Medical Decision Making*, vol. 24, no. 1, pp. 89–100, 2004.

[17] L. Clifton, D. A. Clifton, M. A. F. Pimentel, P. J. Watkinson, and L. Tarassenko, "Gaussian processes for personalized e-health monitoring with wearable sensors," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 1, pp. 193–197, Jan 2013.

[18] O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras, "When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks," in *USENIX*, 2018, pp. 1299–1316.

[19] R. S. Liwei Song and P. Mittal, "Membership inference attacks against adversarially robust deep learning models," 2019.

[20] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "Prada: protecting against dnn model stealing attacks," pp. 512–527, 2019.

[21] E. Chou, F. Tramèr, G. Pellegrino, and D. Boneh, "Sentinet: Detecting physical attacks against deep learning systems," *arXiv preprint arXiv:1812.00292*, 2018.

[22] Y. Wang, S. Jha, and K. Chaudhuri, "Analyzing the robustness of nearest neighbors to adversarial examples," in *ICML*, 2018, pp. 5120–5129.

[23] A. Fawzi, H. Fawzi, and O. Fawzi, "Adversarial vulnerability for any classifier," in *NIPS*, 2018, pp. 1186–1195.

[24] T. Tanay and L. D. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," *CoRR*, vol. 1608.07690, 2016.

[25] M. T. Smith, M. A. Álvarez, M. Zwiessele, and N. D. Lawrence, "Differentially private regression with gaussian processes," in *AISTATS*, 2018, pp. 1195–1203.

[26] J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani, "Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks," *ArXiv e-prints*, Jul. 2017.

[27] K. Grosse, D. Pfaff, M. T. Smith, and M. Backes, "The limitations of model uncertainty in adversarial settings," *Bayesian Deepl Learning Workshop @NeurIPS*, 2019.

[28] I. Bogunovic, J. Scarlett, S. Jegelka, and V. Cevher, "Adversarially robust optimization with gaussian processes," in *NIPS*, 2018, pp. 5765–5775.

[29] A. Blaas, L. Laurenti, A. Patane, L. Cardelli, M. Kwiatkowska, and S. Roberts, "Robustness quantification for classification with gaussian processes," *AIstats*, 2020.

[30] M. T. Smith, K. Grosse, M. Backes, and M. A. Alvarez, "Adversarial vulnerability bounds for gaussian process classification," *ML with guarantees @NeurIPS*, 2019.

[31] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[32] I. J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[33] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," in *EuroS&P*, 2016.

[34] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE S&P*, 2017, pp. 39–57.

[35] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 3–18.

[36] L. Remaki and M. Cheriet, "Kcs-new kernel family with compact support in scale space: formulation and impact," *IEEE Transactions on Image Processing*, vol. 9, no. 6, pp. 970–981, 2000.

[37] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers, "Fisher discriminant analysis with kernels," in *IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, Aug 1999, pp. 41–48.

[38] N. Šrndić and P. Laskov, "Hidost: a static machine-learning-based detector of malicious files," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 22, Sep 2016. [Online]. Available: https://doi.org/10.1186/s13635-016-0045-0

[39] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket." in *NDSS*, 2014.

[40] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[42] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep and Unsupervised Feature Learning*, 2011.

[43] GPy, "GPy: A gaussian process framework in python," http://github.com/SheffieldML/GPy, since 2012.

[44] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[45] I. J. Goodfellow, N. Papernot, and P. D. McDaniel, "cleverhans v0.1: an adversarial machine learning library," *CoRR*, vol. abs/1610.00768, 2016. [Online]. Available: http://arxiv.org/abs/1610.00768