arXiv:1810.06583v5 [cs.LG] 24 May 2019

Concise and Stable Explanations using Adversarial Training

Prasad Chalasani
MediaMath
New York, NY
pchalasani@gmail.com

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

Aravind Sadagopan
MediaMath
New York, NY
arvind.contactme@gmail.com

Xi Wu
Google
Madison, WI
wu.andrew.xi@gmail.com

Abstract

We show new connections between adversarial learning and explainability. One form of explanation of the output of a neural network model in terms of its input features, is a vector of feature-attributions using the Integrated Gradient (IG) method. Two desirable characteristics of an attribution-based explanation are: (1) *sparseness*: the attributions of irrelevant or weakly relevant features should be negligible, thus resulting in *concise* explanations in terms of the significant features, and (2) *stability*: it should not vary significantly within a small local neighborhood of the input. For a general class of convex loss functions, we show two new theoretical results: (a) via a novel analysis of the Stochastic Gradient Descent process, we show that adversarial training using an ℓ_∞-bounded adversary produces sparse models (and hence concise explanations), and (b) we show that natural model-training while encouraging stable IG-explanations (via an extra term in the loss function), is equivalent to adversarial training. We demonstrate the first phenomenon in experiments training logistic regression models on a variety of tabular toy and real-world advertising datasets: adversarial training yields significantly sparser models, with little or no degradation in performance on natural test data.

1 Introduction

Despite the recent dramatic success of deep learning models in a variety of domains, two concerns have surfaced about these models:

Vulnerability to Adversarial Attacks: We can abstractly think of a neural network model as a function $F(\boldsymbol{x})$ of a d-dimensional input vector $\boldsymbol{x} \in \mathbb{R}^d$, and the range of F is either a discrete set of class-labels, or a continuous value such as a probability. Many of these models can be foiled by an adversary who imperceptibly (to humans) alters the input $\boldsymbol{x}$ by adding a perturbation $\delta \in \mathbb{R}^d$ so that $F(\boldsymbol{x}+\delta)$ is very different from $F(\boldsymbol{x})$ [1–4]. *Adversarial training* (or *adversarial learning*) has recently been proposed as a method for training models that are robust to such attacks, by applying techniques from the area of Robust Optimization [5, 6]. The core idea of adversarial training is simple: we define a set S of allowed perturbations $\delta \in \mathbb{R}^d$ that we want to "robustify" against (e.g. S could be the set of δ where $||\delta||_\infty \leq \varepsilon$), and perform model-training using a variant of Stochastic

Preprint. Under review.

Gradient Descent exactly as in natural training, except that each training example x is perturbed adversarially, i.e. replaced by $x + \delta^*$ where $\delta^* \in S$ maximizes the example's loss-contribution.

Explainability: One way to address the well-known lack of explainability of deep learning models is *feature attribution*, which aims to explain the output of a model $F(\boldsymbol{x})$ as a vector $A_F(\boldsymbol{x})$ of the contributions from the input features $\boldsymbol{x}$. A particularly noteworthy feature-attribution technique is the *Integrated Gradients* (IG) method [7], which has a number of intuitively appealing axiomatic properties. Thus the IG-vector $IG_F(x)$ can be viewed as an explanation for $F(x)$. For such an explanation to be human-friendly, it is highly desirable [8] that the IG-vector is *sparse*, i.e., only the features that are truly predictive of the output $F(x)$ should have significant contributions, and irrelevant or weakly-relevant features should have negligible contributions. A sparse attribution makes it possible to produce a *concise* explanation, where only the input features with significant contributions are included. For instance if the model F is used for a loan approval decision, then customers, data-scientists and regulators would like to know the reason for a specific decision in simple terms. In practice however, due to artifacts in the training data, the IG-vector is often not sparse, and irrelevant or weakly-relevant features end up having significant contributions [9]. Another desirable property of a good explanation is *stability*: the IG-vector should not vary significantly within a small local neighborhood of the input x. Similar to the lack of concise explainability, natural training often results in explanations that lack stability [10].

This paper shows new connections between adversarial robustness and the above-mentioned aspects of explainability, namely conciseness and stability of explanations.

Our first set of results demonstrate that adversarial training produces sparse models (which in turn leads to concise explanations). To show this we present a novel analysis of the *expectation* of the weight-change of a feature during an SGD step on a randomly drawn, adversarially perturbed data point (Sec. 3). In particular we show (Theorem 3.1) that for a general class of convex loss functions (which includes popular loss functions like logistic and hinge loss), and adversarial perturbations δ satisfying $||\delta||_\infty \leq \varepsilon$, the weights of "weak" features are on average more aggressively shrunk toward zero than during natural training, and the rate of shrinkage is proportional to the amount by which ε exceeds a certain measure of the "strength" of the feature.

In Section 6 and in the Supplement, we empirically show this phenomenon in several experiments on tabular datasets (including real-world advertising datasets), in the context of learning logistic regression models. In all of our experiments, we find that it is possible to choose an ℓ_∞ bound ε so that adversarial learning under this bound produces sparse models *with little or no drop in performance on natural test data*. Moreover, for the above class of loss functions, we show that there is a simple closed form formula for the worst-case adversarial perturbation δ^*. These results imply an extremely simple and efficient way to produce sparse models without impacting natural accuracy: use $\ell_\infty(\varepsilon)$-adversarial training with a suitable ε. By contrast, previous methods to train sparse logistic regression models (e.g. [9, 11]) require specially implemented optimization procedures.

Secondly, we show a closed form formula for the IG-vector for any 1-layer neural network with an arbitrary differentiable activation function (Sec. 4). This closed form formula enables very efficient computation of the IG vector for a specific example, as well as an "aggregate" IG vector over an entire dataset. Aggregate IG vectors help illuminate the overall importance of features.

Finally, we show theoretically (Sec. 5) that training 1-layer networks naturally, while encouraging stability of explanations (via a suitable term added to the loss function), is in fact equivalent to adversarial training.

Remark. Although our theoretical results in Sections 3 and 5 are stated in terms of loss functions that are typically used to train 1-layer networks (e.g. Logistic or Poisson regression), it is also possible to think of the input vector $\boldsymbol{x}$ as representing the activations at the *penultimate layer of a deep network* – in other words $\boldsymbol{x}$ could either represent "raw" input features in a 1-layer network, or "derived" features in a deep network ([12, 13] take a similar viewpoint). For example one can imagine a transfer learning scenario where the network up to the penultimate layer $\boldsymbol{x}$ is "frozen", and the weights of the final layer are learned.

2 Setup and Assumptions

We assume there is a distribution $\mathcal{D}$ of data points $(\boldsymbol{x}, y)$ where $\boldsymbol{x} \in \mathbb{R}^d$ is an input feature vector, and $y \in \{\pm 1\}$ is its true label[1]. For each $i \in [d]$, the i'th component of $\boldsymbol{x}$ represents an input feature, and is denoted by x_i. The model is assumed to have learnable parameters ("weights") $\boldsymbol{w} \in \mathbb{R}^d$, and for a given data point $(\boldsymbol{x}, y)$, the *loss* is given by some function $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w})$. *Natural model training* [2] consists of minimizing the expected loss, known as *empirical risk*:

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w})]. \tag{1}$$

We sometimes assume the existence of an $\ell_\infty(\varepsilon)$-*adversary* who may perturb the input example $\boldsymbol{x}$ by adding a vector $\delta \in \mathbb{R}^d$ whose ℓ_∞-norm is bounded by ε; such a perturbation δ is referred to as an $\ell_\infty(\varepsilon)$-perturbation. For a given data point $(\boldsymbol{x}, y)$ and a given loss function $\mathcal{L}(.)$, an $\ell_\infty(\varepsilon)$-*adversarial* perturbation is a δ^* that maximizes the *adversarial loss* $\mathcal{L}(\boldsymbol{x}+\delta^*, y; \boldsymbol{w})$.

The aim of *adversarial training*[5] is to train a model that is *robust* to an $\ell_\infty(\varepsilon)$-adversary (i.e. performs well in the presence of such an adversary), and consists of minimizing the expected $\ell_\infty(\varepsilon)$-adversarial loss:

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\max_{||\delta||_\infty \le \varepsilon} \mathcal{L}(\boldsymbol{x}+\delta, y; \boldsymbol{w})]. \tag{2}$$

In the expectations (1) and (2) we often drop the subscript under $\mathbb{E}$ when it is clear that the expectation is over $(\boldsymbol{x}, y) \sim \mathcal{D}$.

For various theoretical results we will make certain assumptions regarding the form and properties of the loss function, the properties of its first derivative, and the data distribution. We highlight these assumptions (with mnemonic names) here for ease of future reference.

Assumption LOSS-INC. *The loss function is of the form $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) = g(-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)$ where g is a non-decreasing function.*

Assumption LOSS-CVX. *The loss function is of the form $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) = g(-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)$ where g is non-decreasing, almost-everywhere differentiable, and convex.*

Section B.1 in the Supplement shows that these Assumptions are satisfied by popular loss functions such as logistic and hinge loss. Incidentally, note that for any differentiable function g, g is convex if and only if its first-derivative g' is non-decreasing, and we will use this property in some of the proofs.

Assumption FEAT-INDEP. *The features $\boldsymbol{x}$ are conditionally independent given the label y, i.e. for any two distinct indices i, j, x_i is independent of x_j given y, or more compactly, $(x_i \perp x_j) \mid y$.*

This Assumption is used to prove Theorem 3.1 in the next Section, and is not as restrictive as it may first appear: one can imagine *clustering* features into groups that are conditionally independent of each other (given the label y), and extend our results to such clustered features; we leave this for future work.

Assumption FEAT-EXP. *For each feature x_i, $i \in [d]$, $\mathbb{E}(x_i|y) = a_i y$ for some constant a_i.*

In Section B.2 (Supplement) we show that Assumption FEAT-EXP is without loss of generality, and it is worth observing that this Assumption implies

$$\mathbb{E}(yx_i) = \mathbb{E}[\mathbb{E}(yx_i|y)] = \mathbb{E}[y\mathbb{E}(x_i|y)] = \mathbb{E}[y^2 a_i] = a_i, \tag{3}$$

where the last equality is due to the fact that $y \in \{\pm 1\}$. Similarly,

$$\mathbb{E}(yx_i|y) = y\mathbb{E}[x_i|y] = y^2 a_i = a_i. \tag{4}$$

Note that for any j, the expectation $\mathbb{E}(yx_j)$ can be thought of as representing the *degree of association*[3] between feature x_j and label y. When the data distribution satisfies Assumption FEAT-EXP, $\mathbb{E}(yx_j) = a_j$, so we refer to a_j as the *directed strength* [4] of feature x_j, and $|a_j|$ is referred to as the *absolute strength* of x_j. In particular when $|a_j|$ is large (small) we say that x_j is a *strong* (*weak*) feature.

[1] It is trivial to convert -1/1 labels to 0/1 labels and vice versa

[2] Also referred to as *standard training* by [5]

[3] When the features are standardized to have mean 0, $\mathbb{E}(yx_j)$ is in fact the covariance of y and x_j.

[4] This is related to the feature "robustness" notion introduced in [12]

3 Analysis of SGD Updates in Adversarial Training

One of the main results of this work is a theoretical characterization of the weight updates during a single SGD step, when applied to a randomly drawn data point $(\boldsymbol{x}, y) \sim \mathcal{D}$ that is subjected to an $\ell_\infty(\varepsilon)$-adversarial perturbation. As a preliminary, it is easy to show the following expressions related to the $\ell_\infty(\varepsilon)$-adversarial perturbation δ^* (See Lemmas 2 and 3 in Section C of the Supplement): For loss functions satisfying Assumption LOSS-INC, the $\ell_\infty(\varepsilon)$-adversarial perturbation δ^* is given by:

$$\delta^* = -y \operatorname{sgn}(\boldsymbol{w})\varepsilon, \tag{5}$$

the corresponding $\ell_\infty(\varepsilon)$-adversarial loss is

$$\mathcal{L}(\boldsymbol{x} + \delta^*, y;\ \boldsymbol{w}) = g(\varepsilon||\boldsymbol{w}||_1 - y\langle \boldsymbol{w},\ \boldsymbol{x}\rangle), \tag{6}$$

and the gradient of this loss w.r.t. a weight w_i is

$$\frac{\partial \mathcal{L}(\boldsymbol{x} + \delta^*, y;\ \boldsymbol{w})}{\partial w_i} = -g'(\varepsilon||\boldsymbol{w}||_1 - y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)\ (yx_i - \operatorname{sgn}(w_i)\varepsilon). \tag{7}$$

In our main result, the expectation of the g' term in (7) plays an important role, so we will use the following notation:

$$\overline{g'} := \mathbb{E}\big[g'(\varepsilon||\boldsymbol{w}||_1 - y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)\big] \tag{8}$$

Ideally, we would like to understand the nature of the weight-vector $\boldsymbol{w}^*$ that minimizes the expected adversarial loss (2). This is quite challenging, so rather than analyzing the *final* optimum of (2), we instead analyze how an SGD-based optimizer for (2) *updates* the model weights $\boldsymbol{w}$. We assume an idealized SGD process: (a) a data point $(\boldsymbol{x}, y)$ is drawn from distribution $\mathcal{D}$, (b) $\boldsymbol{x}$ is replaced by $\boldsymbol{x}' = \boldsymbol{x} + \delta^*$ where δ^* is an $\ell_\infty(\varepsilon)$-adversarial perturbation with respect to the loss function $\mathcal{L}$, (c) each weight w_i is updated by an amount $\Delta w_i = -\partial \mathcal{L}(\boldsymbol{x}', y; \boldsymbol{w})/\partial w_i$ (assuming a unit learning rate to avoid notational clutter). We are interested in the *expectation* of Δw_i, in order to understand what happens to a weight w_i *on average* during a single SGD step. In fact it will be useful to analyze the **expected SGD update** $\overline{\Delta w_i}$ defined as follows:

$$\overline{\Delta w_i} := \begin{cases} -\mathbb{E}\frac{\partial \mathcal{L}}{\partial w_i}, & \text{when } w_i = 0, \\ -\operatorname{sgn}(w_i)\mathbb{E}\frac{\partial \mathcal{L}}{\partial w_i}, & \text{when } w_i \neq 0, \end{cases} \tag{9}$$

When $w_i \neq 0$, a **positive** (**negative**) $\overline{\Delta w_i}$ indicates that during an SGD update on a randomly chosen data-point, on average the weight w_i **expands** (**shrinks**), i.e. maintains its sign and increases (decreases) in absolute value.

The following result characterizes $\overline{\Delta w_i}$ (the proof is in Section D of the Supplement).

Theorem 3.1 (Expected SGD Update in Adversarial Training)**.** *For any loss function $\mathcal{L}$ satisfying Assumption LOSS-CVX and a data distribution $\mathcal{D}$ satisfying Assumptions FEAT-INDEP and FEAT-EXP, if a data point $(\boldsymbol{x}, y)$ is randomly drawn from $\mathcal{D}$, and $\boldsymbol{x}$ is perturbed to $\boldsymbol{x}' = \boldsymbol{x} + \delta^*$, where δ^* is an $\ell_\infty(\varepsilon)$-adversarial perturbation, then under the $\ell_\infty(\varepsilon)$-adversarial loss $\mathcal{L}(\boldsymbol{x}', y; \boldsymbol{w})$, the expected SGD-update of weight w_i, namely $\overline{\Delta w_i}$, satisfies the following properties:*

1. *If $w_i = 0$, then*
$$\overline{\Delta w_i} = \overline{g'}\ a_i. \tag{10}$$
2. *If $w_i \neq 0$, then*
$$\overline{\Delta w_i} \leq \overline{g'}\ [a_i \operatorname{sgn}(w_i) - \varepsilon], \tag{11}$$
and equality holds in the limit as w_i approaches zero,

where $a_i = \mathbb{E}(x_i y)$ is the directed strength of feature x_i from Assumption FEAT-EXP, and $\overline{g'}$ is the expectation in (8).

3.1 Implications of Theorem 3.1

Note that $a_j \operatorname{sgn}(w_j) > 0$ signifies that the sign of the weight w_j agrees with the *directed strength* a_j of feature x_j, and for brevity we simply say that *weight w_j is aligned*. Conversely when $a_j \operatorname{sgn}(w_j) < 0$ we say that *weight w_j is mis-aligned*. With these observations in mind Theorem 3.1 implies the following:

Weights grow from zero in the correct direction. When $w_i = 0$, the expected SGD update $\overline{\Delta w_i}$ is proportional to the directed strength a_i of feature x_i, and if $\overline{g'} \neq 0$, this means that on average the SGD update causes the weight w_i to grow from zero in the *correct direction*. This is what one would expect from an SGD training procedure.

Mis-aligned weights w_i shrink at a rate proportional to $\varepsilon + |a_i|$. When $w_i \neq 0$, if w_i is *mis-aligned*, i.e. $a_i \operatorname{sgn}(w_i) < 0$, the upper bound in (11) is non-positive, and this means the expected SGD update is non-positive, and the weight w_i *shrinks* on average. Thus, mis-aligned weights shrink on average, at a rate proportional to the ℓ_∞ adversarial bound ε plus the absolute feature strength. In other words, all other factors remaining the same, adversarial training (i.e. with $\varepsilon > 0$) shrinks mis-aligned faster than natural training (i.e. with $\varepsilon = 0$).

If w_i is aligned, and $\varepsilon > |a_i|$ then it shrinks at a rate proportional to $\varepsilon - |a_i|$. What happens when a non-zero weight w_i is aligned, i.e. $a_i \operatorname{sgn}(w_i) > 0$? In this case if $\varepsilon > |a_i|$, then the upper-bound (11) on the expected SGD update is once again negative (assuming $\overline{g'} \neq 0$), which means adversarial training with a *sufficiently large* ε that dominates the absolute strength of a feature will cause the weight of that feature to shrink on average. This observation is key to explaining the "feature-pruning" behavior of adversarial training: "weak" features (relative to ε) are weeded out by the SGD updates.

If w_i is aligned, and $\varepsilon < |a_i|$, then w_i expands up to a certain point. If a non-zero weight w_i is aligned and $\varepsilon < |a_i|$, then the upper bound (11) on $\overline{\Delta w_i}$ is non-negative. Since the Theorem states that equality holds in the limit as w_i approaches zero, this means if $|w_i|$ is sufficiently small, the expected SGD update $\overline{\Delta w_i}$ is non-negative, i.e., the weight w_i expands on average. In other words, weights of aligned features expand on average up to a certain point, if ε does not dominate their strength.

Note that Assumption LOSS-CVX implies that $\overline{g'} \geq 0$, and when the model $\boldsymbol{w}$ is "far" from the optimum, the values of $-y\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ will tend to be large, and since g' is a non-decreasing function (Assumption LOSS-CVX), $\overline{g'}$ will be large as well. So we can interpret $\overline{g'}$ as being a proxy for "average model error". Thus during the initial iterations of SGD, this quantity will tend to be large and positive, and shrinks toward zero as the model approaches optimality. Since $\overline{g'}$ appears as a factor in (10) and (11), we can conclude that the above effects will be more pronounced in the initial stages of SGD and less so in the later stages. The experimental results described in Section 6 and in Sections I, J of the Supplement are consistent with several of the above effects.

4 Feature Attribution using Integrated Gradients

Theorem 3.1 showed that $\ell_\infty(\varepsilon)$-adversarial training tends to shrink the *weights* of features that are "weak" (relative to ε). We now show a link between weights and *explanations*, specifically explanations in the form of a vector of feature-attributions given by the *Integrated Gradients* (IG) method [7], which is defined as follows: Suppose $F : \mathbb{R}^d \to \mathbb{R}$ is a real-valued function of an input vector. For example F could represent the output of a neural network, or even a loss function $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w})$ when the label y and weights $\boldsymbol{w}$ are held fixed. Let $\boldsymbol{x} \in \mathbb{R}^d$ be a specific input, and $\boldsymbol{u} \in \mathbb{R}^d$ be a baseline input. The IG is defined as the path integral of the gradients along the straight-line path from the baseline $\boldsymbol{u}$ to the input $\boldsymbol{x}$. The IG along the i'th dimension for an input $\boldsymbol{x}$ and baseline $\boldsymbol{u}$ is defined as:

$$\text{IG}_i^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w}) := (x_i - u_i) \times \int_{\alpha=0}^{1} \partial_i F(\boldsymbol{u} + \alpha(\boldsymbol{x} - \boldsymbol{u}))d\alpha, \tag{12}$$

where $\partial_i F(\boldsymbol{z})$ denotes the gradient of $F(\boldsymbol{v})$ along the i'th dimension, at $\boldsymbol{v} = \boldsymbol{z}$. The vector of all IG components $\text{IG}_i^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w})$ is denoted as $\text{IG}^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w})$.

For general neural networks F, the authors of [7] show that the IG attribution method satisfies a number of intuitively appealing properties that any reasonable attribution method should satisfy. They further show how to approximate the IG integral (12) by a summation involving gradients at m equally-spaced points along the straight-line path from the baseline input $\boldsymbol{u}$ to the actual input $\boldsymbol{x}$. While this approximation is reasonably efficient for a fixed example $\boldsymbol{x}$ and dimension i, it can be prohibitively expensive for computing the IG values across a dataset of millions of examples and

thousands of (sparse) features. A closed form expression for the IG would therefore be of significant interest, especially if the goal is to compute the IG over an entire dataset in order to glean aggregate feature importances. As we see below and in Section 5, such an expression also makes it easier to prove useful theoretical results.

The following Lemma (proved in Sec. E of the Supplement) shows a closed form exact expression for the $\text{IG}^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w})$ when $F(\boldsymbol{x})$ is of the form

$$F(\boldsymbol{x}) = A(\langle \boldsymbol{w},\ \boldsymbol{x} \rangle), \tag{13}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is a vector of weights, A is a differentiable scalar-valued function, and $\langle \boldsymbol{w},\ \boldsymbol{x} \rangle$ denotes the dot product of $\boldsymbol{w}$ and x. Note that this form of F could represent a single-layer neural network with any differentiable activation function (e.g., logistic (sigmoid) activation $A(\boldsymbol{z}) = 1/[1 + \exp(-\boldsymbol{z})]$ or Poisson activation $A(\boldsymbol{z}) = \exp(\boldsymbol{z})$), or a differentiable loss function, such as those that satisfy Assumption LOSS-INC for a fixed label y and weight-vector $\boldsymbol{w}$. For brevity, we will refer to a function of the form (13) as representing a "1-Layer Network", with the understanding that it could equally well represent a suitable loss function.

Lemma 4.1 (IG Attribution for 1-layer Networks)**.** *If $F(\boldsymbol{x})$ is computed by a 1-layer network (13) with weights vector $\boldsymbol{w}$, then the Integrated Gradients for all dimensions of $\boldsymbol{x}$ relative to a baseline $\boldsymbol{u}$ are given by:*

$$\text{IG}^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w}) = [F(\boldsymbol{x}) - F(\boldsymbol{u})] \frac{(\boldsymbol{x} - \boldsymbol{u}) \odot \boldsymbol{w}}{\langle \boldsymbol{x} - \boldsymbol{u},\ \boldsymbol{w} \rangle}, \tag{14}$$

where the $\odot$ operator denotes the entry-wise product of vectors.

From this result it is clear that if the weight-vector $\boldsymbol{w}$ is sparse, then the IG vector will also be correspondingly sparse. Theorem 3.1 suggests that $\ell_\infty(\varepsilon)$-adversarial training tends to shrink "weak" features (even if their weights are "aligned", as we saw in Section 3.1). This is in fact what we see in the experimental results (Section 6): real-world tabular datasets often tend to have several such weak features, and $\ell_\infty(\varepsilon)$-adversarial training (with a suitably chosen ε) of logistic regression models results in a sparse weight-vector, and hence a sparse IG vector as well.

5 Training with Explanation Stability is equivalent to Adversarial Training

Suppose we use the IG method described in Sec. 4 as an explanation for the output of a model $F(\boldsymbol{x})$ on a specific input $\boldsymbol{x}$. A desirable property of an explainable model is that the explanation for the value of $F(\boldsymbol{x})$ is *stable*[10], i.e., does not change much under small perturbations of the input $\boldsymbol{x}$. One way to formalize this is to say the following *worst-case* ℓ_1-norm of the change in IG should be small:

$$\max_{\boldsymbol{x}' \in N(\boldsymbol{x}, \varepsilon)} ||\ \text{IG}^F(\boldsymbol{x}', \boldsymbol{u};\ \boldsymbol{w}) - \text{IG}^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w})||_1, \tag{15}$$

where $N(\boldsymbol{x}, \varepsilon)$ denotes a suitable ε-neighborhood of $\boldsymbol{x}$, and $\boldsymbol{u}$ is an appropriate baseline input vector. If the model F is a single-layer neural network, it would be a function of $\langle \boldsymbol{w},\ \boldsymbol{x} \rangle$ for some weights $\boldsymbol{w}$, and typically when training such networks the loss is a function of $\langle \boldsymbol{w},\ \boldsymbol{x} \rangle$ as well, so we would not change the essence of (15) much if instead of F in each IG, we use $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w})$ for a fixed y; let us denote this function by $\mathcal{L}_y$. Also intuitively, $||\ \text{IG}^{\mathcal{L}_y}(\boldsymbol{x}', \boldsymbol{u}; \boldsymbol{w}) - \text{IG}^{\mathcal{L}_y}(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{w})||_1$ is not too different from $||\ \text{IG}^{\mathcal{L}_y}(\boldsymbol{x}', \boldsymbol{x}; \boldsymbol{w})||_1$. These observations motivate the following definition of *Stable-IG Empirical Risk*, which is a modification of the usual empirical risk (1), with a regularizer to encourage stable IG explanations:

$$\mathbb{E}_{(\boldsymbol{x}, y) \sim \mathcal{D}} \big[\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) \ + \max_{||\ \boldsymbol{x}' - \boldsymbol{x}\ ||_\infty \leq \varepsilon} ||\ \text{IG}^{\mathcal{L}_y}(\boldsymbol{x}, \boldsymbol{x}';\ \boldsymbol{w})||_1 \big]. \tag{16}$$

The following somewhat surprising result is proved in Section F of the Supplement.

Theorem 5.1 (Equivalence of Stable IG and Adversarial Robustness)**.** *For loss functions $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w})$ satisfying Assumption LOSS-CVX, the augmented loss inside the expectation (16) equals the $\ell_\infty(\varepsilon)$-adversarial loss inside the expectation (2), i.e.*

$$\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) \ + \max_{||\ \boldsymbol{x}' - \boldsymbol{x}\ ||_\infty \leq \varepsilon} ||\ \text{IG}^{\mathcal{L}_y}(\boldsymbol{x}, \boldsymbol{x}';\ \boldsymbol{w})||_1 = \max_{||\delta||_\infty \leq \varepsilon} \mathcal{L}(\boldsymbol{x} + \delta, y; \boldsymbol{w}) \tag{17}$$

This implies that for loss functions satisfying Assumption LOSS-CVX, minimizing the Stable-IG Empirical Risk (16) is equivalent to minimizing the expected $\ell_\infty(\varepsilon)$-adversarial loss. In other words, for this class of loss functions, *natural model training while encouraging IG stability is equivalent to $\ell_\infty(\varepsilon)$-adversarial training!* Combined with Theorem 3.1 and the corresponding experimental results in Sec 6, this equivalence implies that, for this class of loss functions, and data distributions satisfying Assumption FEAT-INDEP, the explanations for the models produced by $\ell_\infty(\varepsilon)$-adversarial training are both *concise* (due to the sparseness of the models), and *stable*.

6 Experiments

Based on the implications of Theorem 3.1 discussed in Section 3.1, we conjectured that for loss functions satisfying Assumption LOSS-CVX, $\ell_\infty(\varepsilon)$-adversarial training using SGD (with a suitable value of ε) can effectively act as an aggressive "relevance filter", weeding out the weights of features that are irrelevant or "weak" (relative to ε), even when the data distribution does not strictly satisfy Assumption FEAT-INDEP as required in the proof of that Theorem. We put this conjecture to the test by applying $\ell_\infty(\varepsilon)$-adversarial training to *logistic regression* models for *binary probability prediction* tasks, where the output of the model $F(\boldsymbol{x})$ is to be interpreted as the predicted probability that the label y is 1. We used the Logistic Loss, which satisfies Assumption LOSS-CVX (see Supplement, Section B.1). For such loss functions, the $\ell_\infty(\varepsilon)$-adversarial perturbation (5) is applied to the input examples, *after 1-hot-encoding* all categorical variables.

Specifically, we ran experiments on several tabular datasets: toy datasets from the UCI Data Repository [14], and real world advertising datasets from MediaMath. Our experiments confirm the above conjecture: In all cases, we found that it is possible to chose ε from a certain *goldilocks zone* of "good" values that simultaneously achieves two desirable effects: (a) the feature-weight vector of the resulting models is significantly more sparse compared to natural training (i.e., with $\varepsilon = 0$), and this effect is much more pronounced in the large-scale real-world Advertising dataset from MediaMath, as we see below; (b) compared to natural training, there is little or no impact on performance on natural test data, as measured by the AUC-ROC metric.

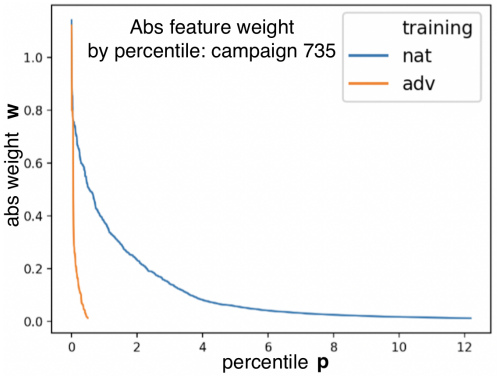
In the MediaMath experiments, we trained logistic regression models on advertising campaign datasets, to predict the probability of positive response to an ad. A given campaign dataset can have as many as 50 million examples, and 400,000 sparse features (due to the presence of several high-cardinality categorical attributes). Given the high dimensionality of the feature-space, it is of considerable practical importance to understand which features have a truly significant impact on the predictions. Figure 1 summarizes results on 6 anonymized campaigns. Results from experiments on the UCI datasets, as well as more details on the MediaMath experiments, are in the Supplement (Sections I, J). We have also made our **code** available in the Supplement.

7 Related Work

In contrast to the growing body of work on defenses against adversarial attacks [15, 5, 4] or explaining adversarial examples [2, 16], the focus of our paper is the connection between adversarial robustness and explainability. We view the process of adversarial training as a tool to produce more explainable models. A recent series of papers [16, 12] essentially argues that adversarial examples exist because standard training produces models that are heavily reliant on highly predictive but *non-robust features* (which is similar to our notion of "weak" features in Sec 3.1) which are vulnerable to an adversary who can "flip" them and cause performance to degrade. This viewpoint would seem to imply a necessary tradeoff between adversarial robustness and *standard* accuracy (i.e. performance on natural test data). In fact [16] constructs a *synthetic* dataset where this tradeoff can be forced: As adversarial accuracy approaches 100%, standard accuracy drops significantly. Nevertheless, our results on tabular *real-world* and toy UCI datasets show that it is possible to use $\ell_\infty(\varepsilon)$-adversarial training to produce models whose standard performance (as measured by AUC-ROC) is on par with that of naturally-trained models, with the added benefit that the weights of weak or irrelevant features have been pruned significantly, and such models lend themselves to more human-friendly and faithful explanations. Indeed the authors of [12] touch upon some connections between explainability and robustness, and conclude, "*As such, producing human-meaningful explanations that remain faithful to underlying models cannot be pursued independently from the training of the models themselves*", by which they are implying that good explainability may require intervening in the model-training

procedure itself; this is consistent with our findings. Another recent related paper [17] analyzes the effect of adversarial training on the interpretability of neural network loss gradients. We discuss other related work in the Supplement Section A.

Campaign	training	AUC	W1P
285	nat	0.560	12.93
	adv ($\varepsilon = 0.01$)	0.556	0.39
479	nat	0.697	11.23
	adv ($\varepsilon = 0.001$)	0.694	6.17
735	nat	0.758	12.20
	adv ($\varepsilon = 0.01$)	0.765	0.51
622	nat	0.565	27.11
	adv ($\varepsilon = 0.01$)	0.561	4.21
594	nat	0.702	16.91
	adv ($\varepsilon = 0.001$)	0.702	1.09
473	nat	0.683	28.02
	adv ($\varepsilon = 0.001$)	0.673	3.58



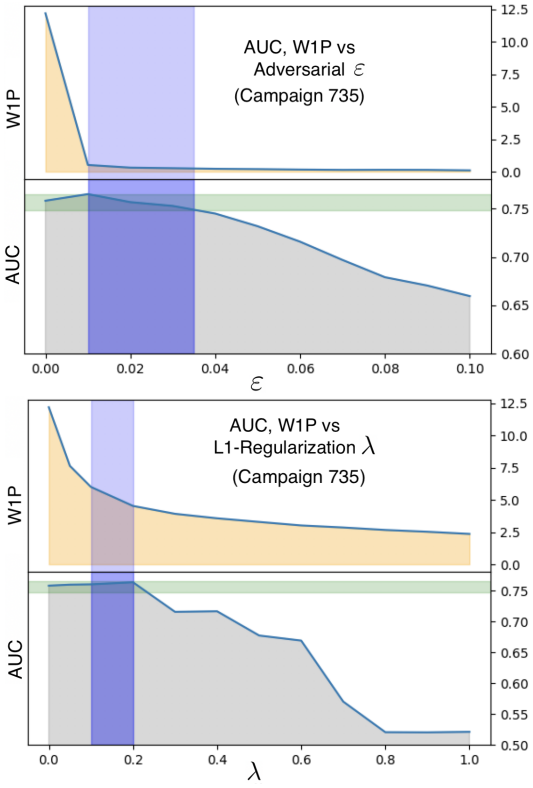


Figure 1: **Top Left:** Comparison of AUC-ROC and feature-concentration (W1P) between natural ("nat") and adversarial ("adv") training on 6 advertising campaigns. W1P is the percent of feature weights whose absolute value is at least 1% of the highest absolute weight. In all cases, adversarial training substantially improves W1P with minimal impact on AUC. The other 3 figures refer to campaign 735. **Bottom Left:** Comparison of drop-off of absolute feature weights in natural and adversarial training ($\varepsilon = 0.01$). In each curve, a point (p, w) indicates that p percent of the absolute weights are larger than w. Each curve is truncated when the absolute weight reaches 1% of the highest absolute weight in the respective model. The adversarially trained model has a much steeper absolute weight drop-off, with only 0.5% of the absolute weights being above the 1% threshold, compared to 12% with natural training. **Top Right:** Variation of AUC and W1P with $\ell_\infty(\varepsilon)$-adversarial training, as ε is increased. Notice that for ε values in the blue vertical band (the "goldilocks zone"), W1P is sharply lower than with natural training ($\varepsilon = 0$) and yet AUC remains within 0.01 of the naturally trained AUC (green horizontal band). **Bottom Right:** Variation of AUC, W1P with natural training, as the L_1-regularization strength λ is varied. Notice that as λ increases, before the AUC degrades significantly, the drop in W1P is significantly less compared to $\ell_\infty(\varepsilon)$-adversarial training.

8 Conclusion

We presented theoretical and experimental results that show a strong connection between adversarial robustness (under ℓ_∞-bounded perturbations) and two desirable properties of model explanations: conciseness and stability. On the theory side we showed two properties of ℓ_∞-bounded adversarial training, for a general class of convex loss functions: (a) it tends to aggressively prune the weights of marginally-relevant features (when the features are conditionally independent given the label), thus yielding concise explanations, and (b) it is equivalent to natural training with a loss function that encourages local stability of the IG vector. We presented extensive experiments that demonstrate

the "feature-pruning" benefits of adversarial training on real-world tabular data-sets, and found no appreciable loss in performance on natural test data. For 1-layer networks we also showed a simple closed form formula for the feature-attribution vector based on Integrated Gradients (IG).

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," arXiv:1312.6199 [cs.CV]. http://arxiv.org/abs/1312.6199.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv:1412.6572 [stat.ML]. http://arxiv.org/abs/1412.6572.

[3] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. Berkay Celik, and A. Swami, "The limitations of deep learning in adversarial settings," arXiv:1511.07528 [cs.CR]. http://arxiv.org/abs/1511.07528.

[4] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases*, pp. 387–402. Springer Berlin Heidelberg, 2013. http://dx.doi.org/10.1007/978-3-642-40994-3_25.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," arXiv:1706.06083 [stat.ML].

[6] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training." Feb., 2018.

[7] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," arXiv:1703.01365 [cs.LG].

[8] C. Molnar, *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

[9] M. Tan, I. W. Tsang, and L. Wang, "Minimax sparse logistic regression for very high-dimensional feature selection," *IEEE Trans Neural Netw Learn Syst* **24** no. 10, (Oct., 2013) 1609–1622. http://dx.doi.org/10.1109/TNNLS.2013.2263427.

[10] D. Alvarez-Melis and T. S. Jaakkola, "On the robustness of interpretability methods," *arXiv preprint arXiv:1806. 08049* (2018) . http://arxiv.org/abs/1806.08049.

[11] M. Tan, I. W. Tsang, and L. Wang, "Towards ultrahigh dimensional feature selection for big data," *J. Mach. Learn. Res.* **15** no. 1, (2014) 1371–1429. http://www.jmlr.org/papers/volume15/tan14a/tan14a.pdf.

[12] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," arXiv:1905.02175 [stat.ML]. http://arxiv.org/abs/1905.02175.

[13] D. Alvarez Melis and T. Jaakkola, "Towards robust interpretability with Self-Explaining neural networks," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., pp. 7775–7784. Curran Associates, Inc., 2018. http://papers.nips.cc/paper/8003-towards-robust-interpretability-with-self-explaining-neural-networks.pdf.

[14] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. http://archive.ics.uci.edu/ml.

[15] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," arXiv:1712.07107 [cs.LG]. http://arxiv.org/abs/1712.07107.

[16] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," arXiv:1805.12152 [stat.ML]. http://arxiv.org/abs/1805.12152.

[17] B. Kim, J. Seo, and T. Jeon, "Bridging adversarial robustness and gradient interpretability," *Safe Machine Learning workshop at ICLR* (2019) .

[18] H. Xu, C. Caramanis, and S. Mannor, "Robustness and regularization of support vector machines," *J. Mach. Learn. Res.* **10** no. Jul, (2009) 1485–1510. http://www.jmlr.org/papers/volume10/xu09b/xu09b.pdf.

[19] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of neural nets through robust optimization," arXiv:1511.05432 [stat.ML]. http://arxiv.org/abs/1511.05432.

[20] S. Sankaranarayanan, A. Jain, R. Chellappa, and S. N. Lim, "Regularizing deep networks using efficient layerwise adversarial training," arXiv:1705.07819 [cs.CV]. http://arxiv.org/abs/1705.07819.

[21] T. Tanay and L. D. Griffin, "A new angle on l2 regularization," *CoRR* **abs/1806.11186** (2018) .

[22] H. Brendan McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica, "Ad click prediction: a view from the trenches,". https://ai.google/research/pubs/pub41159.

[23] O. Chapelle, E. Manavoglu, and R. Rosales, "Simple and scalable response prediction for display advertising," *ACM Trans. Intell. Syst. Technol.* **5** no. 4, (Dec., 2014) 61:1–61:34. http://doi.acm.org/10.1145/2532128.

[24] S. Sperandei, "Understanding logistic regression analysis," *Biochem. Med.* **24** no. 1, (Feb., 2014) 12–18. http://dx.doi.org/10.11613/BM.2014.003.

[25] R. P. Anderson, R. Jin, and G. L. Grunkemeier, "Understanding logistic regression analysis in clinical reports: an introduction," *Ann. Thorac. Surg.* **75** no. 3, (Mar., 2003) 753–757. https://www.ncbi.nlm.nih.gov/pubmed/12645688.

[26] J. Tolles and W. J. Meurer, "Relating patient characteristics to outcomes,". https://www.unav.edu/documents/16089811/16216616/logistic+regression_basico_JAMA16.pdf.

[27] K. El Emam, S. Samet, L. Arbuckle, R. Tamblyn, C. Earle, and M. Kantarcioglu, "A secure distributed logistic regression protocol for the detection of rare adverse drug events," *J. Am. Med. Inform. Assoc.* **20** no. 3, (May, 2013) 453–461. http://dx.doi.org/10.1136/amiajnl-2011-000735.

[28] L. Emfevid and H. Nyquist, *Financial Risk Profiling using Logistic Regression*. PhD thesis, 2018. http://www.diva-portal.org/smash/record.jsf?pid=diva2:1215541.

[29] M. Mironiuc and M.-A. Robu, "Obtaining a practical model for estimating stock performance on an emerging market using logistic regression analysis," *Procedia - Social and Behavioral Sciences* **81** (June, 2013) 422–427. http://www.sciencedirect.com/science/article/pii/S1877042813015218.

[30] C. Bolton, "Logistic regression and its application in credit scoring,". http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1024.2660&rep=rep1&type=pdf.

[31] G. Dong, K. K. Lai, and J. Yen, "Credit scorecard based on logistic regression with random coefficients," *Procedia Comput. Sci.* **1** no. 1, (May, 2010) 2463–2468. http://www.sciencedirect.com/science/article/pii/S1877050910002796.

[32] F. Abramovich and V. Grinshtein, "High-dimensional classification by sparse logistic regression," arXiv:1706.08344 [math.ST]. http://arxiv.org/abs/1706.08344.

[33] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," arXiv:1801.02950 [cs.CR].

[**Code** for the experiments is available at `https://github.com/Ombray/advex`]

Appendix A Additional Related Work

Section 7 discussed some of the work most directly related to this paper. Here we describe some additional related work.

Relation to work on Regularization Benefits of AML. There has been prior work on the *regularization benefits* of adversarial training [18, 1, 2, 19–21], primarily in image-classification applications: when a model is adversarially trained, its classification accuracy on natural (i.e. un-perturbed) test data can improve. All of this prior work has focused on the *performance-improvement* (on natural test data) aspect of regularization, but none have examined the *feature-pruning* benefits explicitly. In contrast to this work, our primary interest is in the explainability benefits of adversarial training, and specifically the ability of adversarial training to significantly improve feature-concentration while maintaining (and often improving) performance on natural test data.

Tabular datasets vs traditional adversarial ML research domains. Although our *theoretical* analysis (Theorem 3.1) is generally applicable to any type of dataset as long as the assumptions required by the theorem (namely, LOSS-CVX, FEAT-INDEP) are satisfied, our *experimental* results are only on *tabular* datasets (as opposed to images, audio or text). We focus on such datasets here because loss functions that satisfy Assumption LOSS-CVX are typically used in training 1-layer neural network models, and the most common such model is logistic regression. Logistic regression models (and variants such a Poisson regression) have been successfully used primarily on tabular datasets, in a variety of domains such as advertising [22, 23], health-care [24–27], and finance [28–31]. In all of these domains, it is crucial to understand which input features are truly important in determining the model's output (either on a single data point or in aggregate across a dataset), and our aim is to show that adversarially-trained logistic regression models yield significantly better feature-attribution by shrinking the weights on irrelevant or weakly-relevant features to negligible levels, with minimal impact on (and often improvement in) performance on natural test data.

Adversarial Training vs ℓ_1-regularization. It might be argued that traditional ℓ_1-regularization can potentially also provide a similar explainability benefit by shrinking the weights of irrelevant features. This does seem to hold for the toy UCI datasets (Supplement Sec. I) with simple optimizers such as pure SGD (Stochastic Gradient Descent). However practitioners typically use more sophisticated optimizers such as FTRL [22] to achieve better model performance on test data, and with this optimizer (especially on the real-world advertising data, as described in Sec. J of the Supplement) the model-concentration produced by adversarial training is significantly better than with ℓ_1-regularization (and ℓ_2-regularization is even worse than ℓ_1-regularization in this respect). Moreover, ℓ_1 or ℓ_2 regularization will not necessarily yield the robustness guarantees provided by adversarial training [2]. In other words adversarial training simultaneously provides robustness to adversarial attacks, as well as explainability benefits, while maintaining or improving model performance on natural test data. These advantages make adversarial training especially compelling for logistic regression models since there is a simple closed form formula (5) for the adversarial perturbation in this case, and so from a computational standpoint it is no more demanding than ℓ_1 or ℓ_2 regularization.

Adversarial Training vs Feature-Selection. Since our results show that adversarial training can effectively shrink the weights of irrelevant or weakly-relevant features (while preserving weights on relevant features), a legitimate counter-proposal might be that one could weed out such features beforehand via a pre-processing step where features with negligible label-correlations can be "removed" from the training process. Besides the fact that this scheme has no guarantees whatsoever with regard to adversarial robustness, there are some practical reasons why correlation-based feature selection is not as effective as adversarial training, in producing pruned models: (a) With adversarial training, one needs to simply try different values of the adversarial strength parameter ε and find a level where accuracy (or other metric such as AUC-ROC) is not impacted much but model-weights are significantly more concentrated; on the other hand with the correlation-based feature-pruning method, one needs to set up an iterative loop with gradually increasing correlation thresholds, and each time the input pre-processing pipeline needs to be re-executed with a reduced set of features. (b)

When there are categorical features with large cardinalities, where just some of the categorical values have negligible feature-correlations, it is not even clear how one can "remove" these specific feature *values*, since the feature itself must still be used; at the very least it would require a re-encoding of the categorical feature each time a subset of its values is "dropped" (for example if a one-hot encoding or hashing scheme is used). Thus correlation-based feature-pruning is a much more cumbersome and inefficient process compared to adversarial training.

Adversarial Training vs Other Methods to Train Sparse Logistic Regression Models. [9, 11] propose an approach to train sparse logistic regression models based on a min-max optimization problem that can be solved by the cutting plane algorithm. This requires a specially implemented custom optimization procedure. By contrast, $\ell_\infty(\varepsilon)$-adversarial training can be implemented as a simple and efficient "bolt-on" layer on top of existing ML pipelines based on TensorFlow, PyTorch or SciKit-Learn, which makes it highly practical. Another paper [32] proposes a feature selection procedure based on penalized maximum likelihood with a complexity penalty on the model size, but once again this requires special-purpose optimization code.

Appendix B Discussion of Assumptions

B.1 Loss Functions Satisfying Assumption LOSS-CVX

We show here that several popular loss functions satisfy the Assumption LOSS-CVX.

Logistic NLL Loss.
$\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) = -\ln(\sigma(y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)) = \ln(1 + \exp(-\langle \boldsymbol{w}, \boldsymbol{x}\rangle))$, which can be written as $g(-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)$ where $g(z) = (1 + e^z)$ is a non-decreasing and convex function.

Hinge Loss
$\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) = (1 - y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)^+$, which can be written as $g(-\langle \boldsymbol{w}, \boldsymbol{x}\rangle)$ where $g(z) = (1 + z)^+$ is non-decreasing and convex.

Softplus Hinge Loss.
$\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) = \ln(1 + \exp(1 - y\langle \boldsymbol{w}, \boldsymbol{x}\rangle))$, which can be written as $g(-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)$ where $g(z) = \ln(1 + e^{1+z})$, and clearly g is non-decreasing. Moreover the first derivative of g, $g'(z) = 1/(1 + e^{-1-z})$ is non-decreasing, and therefore g is convex.

B.2 Assumption FEAT-EXP is without loss of generality

While Assumption FEAT-EXP may seem restrictive, we show that there is in fact no loss of generality in making this assumption, when there are only two possible values of the label.

Lemma 1 (Form of conditional expectation of X given Y when $Y \in \{\pm 1\}$)**.** *Given random variables X, Y where $Y \in \{\pm 1\}$, $\mathbb{E}(X|Y)$ must be of the form:*

$$\mathbb{E}(X|Y) = aY + c \tag{B.18}$$

where

$$a = [\mathbb{E}(X|Y = 1) - \mathbb{E}(X|Y = -1)]/2 \tag{B.19}$$

and

$$c = [\mathbb{E}(X|Y = 1) + \mathbb{E}(X|Y = -1)]/2 \tag{B.20}$$

This implies that the random variable $X' = X - c$ satisfies $\mathbb{E}(X'|Y) = aY$.

Proof. Consider the function $f(Y) = \mathbb{E}(X|Y)$, and let $b_0 := f(-1)$ and $b_1 := f(1)$. Since there are only *two* values of Y that are of interest, we can represent $f(Y)$ by a *linear* function $aY + c$, and it is trivial to verify that $a = (b_1 - b_0)/2$ and $c = (b_1 + b_0)/2$ are the unique values that are consistent with $f(-1) = b_0$ and $f(1) = b_1$. □

A consequence of this Lemma is that a dataset can be transformed into one satisfying Assumption FEAT-EXP by a simple translation of each x_i to a new feature

$$x'_i = x_i - 0.5 * [\mathbb{E}(x_i|y = 1) + \mathbb{E}(x_i|y = -1)]. \tag{B.21}$$

Appendix C Expressions for adversarial perturbation and loss-gradient

We show two simple preliminary results for loss functions that satisfy Assumption LOSS-INC: Lemma 2 shows a simple closed form expression for the $\ell_\infty(\varepsilon)$-adversarial perturbation, and we use this result to derive an expression for the *gradient* of the $\ell_\infty(\varepsilon)$-adversarial loss $\mathcal{L}(\boldsymbol{x}+\delta^*, y; \boldsymbol{w})$ with respect to a weight w_i (Lemma 3).

Lemma 2 (Closed form for $\ell_\infty(\varepsilon)$-adversarial perturbation)**.** *For a data point $(\boldsymbol{x}, y)$, given model weights $\boldsymbol{w}$, if the loss function $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w})$ satisfies Assumption LOSS-INC, the $\ell_\infty(\varepsilon)$-adversarial perturbation δ^* is given by:*

$$\delta^* = -y \operatorname{sgn}(\boldsymbol{w})\varepsilon, \tag{5}$$

and the corresponding $\ell_\infty(\varepsilon)$-adversarial loss is

$$\mathcal{L}(\boldsymbol{x}+\delta^*, y;\ \boldsymbol{w}) = g(\varepsilon||\boldsymbol{w}||_1 - y\langle \boldsymbol{w},\ \boldsymbol{x}\rangle) \tag{6}$$

Proof. Assumption LOSS-INC implies that the loss is non-increasing in $y\langle \boldsymbol{w}, x\rangle$, and therefore the $\ell_\infty(\varepsilon)$-perturbation δ^* of x that maximizes the loss would be such that, for each $i \in [d]$, x_i is changed by an amount ε in the direction of $-y \operatorname{sgn}(w_i)$, and the result immediately follows. □

Lemma 3 (Gradient of adversarial loss)**.** *For any loss function satisfying Assumption LOSS-INC, for a given data point $(\boldsymbol{x}, y)$, the gradient of the $\ell_\infty(\varepsilon)$-adversarial loss is given by:*

$$\frac{\partial \mathcal{L}(\boldsymbol{x}+\delta^*, y;\ \boldsymbol{w})}{\partial w_i} = -g'(\varepsilon||\boldsymbol{w}||_1 - y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)\ (yx_i - \operatorname{sgn}(w_i)\varepsilon) \tag{7}$$

Proof. This is straightforward by substituting the expression (5) for δ^* in $g(-y\langle \boldsymbol{w}, \boldsymbol{x}+\delta^*\rangle)$, and applying the chain rule. □

Appendix D Expectation of SGD Weight Update

The following Lemma will be used to prove Theorem 3.1.

D.1 Upper bound on $\mathbb{E}[Zf(Z,V)]$

Lemma 4 (Upper Bound on expectation of $Zf(Z,V)$ when f is non-increasing in Z, $(Z \perp V)|Y$, and $\mathbb{E}(Z|Y) = \mathbb{E}(Z)$)**.** *For any random variables Z, V, if:*

- *$f(Z,V)$ is non-increasing in Z,*
- *Z, V are conditionally independent given a third r.v. Y, and*
- *$\mathbb{E}(Z|Y) = \mathbb{E}(Z)$,*

then

$$\mathbb{E}[Zf(Z,V)] \ \leq\ \mathbb{E}(Z)\mathbb{E}[f(Z,V)] \tag{D.22}$$

Proof. Let $\overline{z} = \mathbb{E}(Z) = \mathbb{E}(Z|Y)$ and note that

$$\mathbb{E}[Zf(Z,V)] - \mathbb{E}[Z]\mathbb{E}[f(Z,V)] = \mathbb{E}[Zf(Z,V)] - \overline{z}\mathbb{E}[f(Z,V)] \tag{D.23}$$
$$= \mathbb{E}[(Z-\overline{z})f(Z,V)] \tag{D.24}$$

We want to now argue that $\mathbb{E}[(Z-\overline{z})f(\overline{z},V)] = 0$. To see this, apply the Law of Total Expectation by conditioning on Y:

$$\begin{aligned}
\mathbb{E}[(Z-\overline{z})f(\overline{z},V)] &= \mathbb{E}\Big[\mathbb{E}\big[(Z-\overline{z})f(\overline{z},V)|Y\big]\Big] \\
&= \mathbb{E}\Big[\mathbb{E}\big[(Z-\overline{z})|Y\big]\mathbb{E}\big[f(\overline{z},V)|Y\big]\Big] && \text{(since } (Z \perp V)|Y\text{)} \quad \text{(D.25)} \\
&= 0. && \text{(since } \mathbb{E}(Z|Y) = \mathbb{E}(Z) = \overline{z}\text{)} \quad \text{(D.26)}
\end{aligned}$$

Since $\mathbb{E}[(Z-\overline{z})f(\overline{z},V)]=0$, we can subtract it from the last expectation in (D.24), and by linearity of expectations the RHS of (D.24) can be replaced by

$$\mathbb{E}\big[(Z-\overline{z})(f(Z,V)-f(\overline{z},V))\big]. \tag{D.27}$$

That fact that $f(Z,V)$ is non-increasing in Z implies that $(Z-\overline{z})(f(Z,V)-f(\overline{z},V))\leq 0$ for any value of Z and V, with equality when $Z=\overline{z}$. Therefore the expectation (D.27) is bounded above by zero, which implies the desired result. □

Theorem 3.1 (Expected SGD Update in Adversarial Training)**.** *For any loss function $\mathcal{L}$ satisfying Assumption LOSS-CVX and a data distribution $\mathcal{D}$ satisfying Assumptions FEAT-INDEP and FEAT-EXP, if a data point $(\boldsymbol{x},y)$ is randomly drawn from $\mathcal{D}$, and $\boldsymbol{x}$ is perturbed to $\boldsymbol{x}'=\boldsymbol{x}+\delta^*$, where δ^* is an $\ell_\infty(\varepsilon)$-adversarial perturbation, then under the $\ell_\infty(\varepsilon)$-adversarial loss $\mathcal{L}(\boldsymbol{x}',y;\boldsymbol{w})$, the expected SGD-update of weight w_i, namely $\overline{\Delta w_i}$, satisfies the following properties:*

1. *If $w_i=0$, then*
$$\overline{\Delta w_i}=\overline{g'}\;a_i. \tag{10}$$
2. *If $w_i\neq 0$, then*
$$\overline{\Delta w_i}\leq\overline{g'}\;[a_i\,\mathrm{sgn}(w_i)-\varepsilon], \tag{11}$$
and equality holds in the limit as w_i approaches zero,

where $a_i=\mathbb{E}(x_iy)$ is the directed strength of feature x_i from Assumption FEAT-EXP, and $\overline{g'}$ is the expectation in (8).

Proof. Consider the adversarial loss gradient expression (7) from Lemma 3. For the case $w_i=0$, the negative expectation of the adversarial loss gradient is

$$\begin{aligned}\overline{\Delta w_i}&=\mathbb{E}\big[yx_i\,g'(\varepsilon||\boldsymbol{w}||_1-y\langle\boldsymbol{w},\boldsymbol{x}\rangle)\big]\\&=\mathbb{E}\Big[\mathbb{E}\big[yx_i\,g'(\varepsilon||\boldsymbol{w}||_1-y\langle\boldsymbol{w},\boldsymbol{x}\rangle)\;|y\big]\Big]\\&=\mathbb{E}\Big[y\,\mathbb{E}\big[x_i\,g'(\varepsilon||\boldsymbol{w}||_1-y\langle\boldsymbol{w},\boldsymbol{x}\rangle)\;|y\big]\Big],\end{aligned}$$

and in the last expectation above, we note that since $w_i=0$ the argument of g' does not depend on x_i, and by Assumption FEAT-INDEP the features are conditionally independent given y, so the inner conditional expectation can be factored as a product of conditional expectations, which gives

$$\begin{aligned}\overline{\Delta w_i}&=\mathbb{E}\Big[y\mathbb{E}(x_i|y)\mathbb{E}\big[g'(\varepsilon||\boldsymbol{w}||_1-y\langle\boldsymbol{w},\boldsymbol{x}\rangle)\;|y\big]\Big]\\&=\mathbb{E}\Big[y^2a_i\mathbb{E}\big[g'(\varepsilon||\boldsymbol{w}||_1-y\langle\boldsymbol{w},\boldsymbol{x}\rangle)\;|y\big]\Big]&&\text{(due to Assumption FEAT-EXP)}\\&=a_i\mathbb{E}\Big[\mathbb{E}\big[g'(\varepsilon||\boldsymbol{w}||_1-y\langle\boldsymbol{w},\boldsymbol{x}\rangle)\;|y\big]\Big]&&\text{.(since } y=\pm1)\\&=a_i\overline{g'},&&\end{aligned} \tag{D.28}$$

which establishes the first result.

Now consider the case $w_i\neq 0$. Starting with the adversarial loss gradient expression (7) from Lemma 3, multiplying throughout by $-\,\mathrm{sgn}(w_i)$ and taking expectations results in

$$\overline{\Delta w_i}=\mathbb{E}\Big[\big[yx_i\,\mathrm{sgn}(w_i)-\varepsilon\big]\;g'(\varepsilon||w||_1-y\langle\boldsymbol{w},\boldsymbol{x}\rangle)\Big] \tag{D.29}$$

where the expectation is with respect to a random choice of data-point $(\boldsymbol{x},y)$. The argument of g' can be written as

$$\varepsilon||w||_1-y\langle\boldsymbol{w},\boldsymbol{x}\rangle=-\sum_{j=1}^{d}|w_j|(yx_j\,\mathrm{sgn}(w_j)-\varepsilon).$$

For $j\in\{1,2,\ldots,d\}$ if we let Z_j denote the random variable corresponding to $yx_j sgn(w_j)-\varepsilon$, then the expectation (D.30) can be written as

$$\mathbb{E}\left[Z_i\;g'\left(-\sum_{j=1}^{d}|w_j|Z_j\right)\right]=\mathbb{E}\left[Z_i\;g'\left(V-|w_i|Z_i\right)\right], \tag{D.30}$$

where the random variable V denotes the negative sum of the $|w_j|Z_j$ terms over all j *except* $j = i$. Note that the last expectation above is with respect to the distribution of three random variables: Z_i, V, and the random variable Y corresponding to the label y of the data point. Since Z_i is a function of feature x_i and Y, and V is a function of the remaining features and Y, Assumption FEAT-INDEP (the features x_j are conditionally independent given the label Y) implies $(V \perp Z_i)|Y$. Moreover (3) and (4) imply that

$$\mathbb{E}(Z_i) = \mathbb{E}(Z_i|Y) = a_i \operatorname{sgn}(w_i) - \varepsilon. \tag{D.31}$$

Since by Assumption LOSS-CVX, g' is a non-decreasing function, $g'(V - |w_i|Z_i)$ is *non-increasing* in Z_i. Thus all three conditions of Lemma 4 are satisfied, with the random variables Z, V, Y and function f in the Lemma corresponding to random variables Z_i, V, Y and function g' respectively in the present Theorem. It then follows from Lemma 4 that

$$\overline{\Delta w_i} \leq E(Z_i)\overline{g'} \tag{D.32}$$

$$= \overline{g'}[a_i \operatorname{sgn}(w_i) - \varepsilon], \tag{D.33}$$

which establishes the upper bound (11) for any $w_i \neq 0$. Now consider a $w_i \neq 0$ that is infinitesimally close to 0. In this case $\overline{\Delta w_i}$ equals the RHS of (D.30), and if we let $|w_i|$ approach zero,

$$\overline{\Delta w_i} = \mathbb{E}[Z_i g'(V)] = \mathbb{E}\big[\mathbb{E}[Z_i g'(V)|Y]\big] = \mathbb{E}\big[\mathbb{E}[Z_i|Y]\mathbb{E}[g'(V)|Y]\big], \tag{D.34}$$

where the last equality follows from the conditional independence of Z_i and V given Y. Using (D.31) the last expectation above simplifies to $(a_i \operatorname{sgn}(w_i) - \varepsilon)\overline{g'}$, which establishes the equality result for non-zero infinitesimally small w_i. □

Appendix E Proof of Lemma 4.1

Lemma 4.1 (IG Attribution for 1-layer Networks)**.** *If $F(\boldsymbol{x})$ is computed by a 1-layer network (13) with weights vector $\boldsymbol{w}$, then the Integrated Gradients for all dimensions of $\boldsymbol{x}$ relative to a baseline $\boldsymbol{u}$ are given by:*

$$\mathrm{IG}^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w}) = [F(\boldsymbol{x}) - F(\boldsymbol{u})]\frac{(\boldsymbol{x} - \boldsymbol{u}) \odot \boldsymbol{w}}{\langle \boldsymbol{x} - \boldsymbol{u},\ \boldsymbol{w}\rangle}, \tag{14}$$

where the $\odot$ operator denotes the entry-wise product of vectors.

Proof. Since the function F, the baseline input $\boldsymbol{u}$ and weight vector $\boldsymbol{w}$ are fixed, we omit them from $\mathrm{IG}^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w})$ and $\mathrm{IG}_i^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w})$ for brevity. Consider the partial derivative $\partial_i F(\boldsymbol{u} + \alpha(\boldsymbol{x} - \boldsymbol{u}))$ in the definition (12) of $\mathrm{IG}_i(\boldsymbol{x})$. For a given $\boldsymbol{x}$, $\boldsymbol{u}$ and α, let $\boldsymbol{v}$ denote the vector $\boldsymbol{u} + \alpha(\boldsymbol{x} - \boldsymbol{u})$. Then $\partial_i F(\boldsymbol{v}) = \partial F(\boldsymbol{v})/\partial v_i$, and by applying the chain rule we get:

$$\partial_i F(\boldsymbol{v}) := \frac{\partial F(\boldsymbol{v})}{\partial v_i} = \frac{\partial A(\langle \boldsymbol{w},\ \boldsymbol{v}\rangle)}{\partial v_i} = A'(z)\frac{\partial \langle \boldsymbol{w},\ \boldsymbol{v}\rangle}{\partial v_i} = w_i A'(z),$$

where $A'(z)$ is the gradient of the activation A at $z = \langle \boldsymbol{w},\ \boldsymbol{v}\rangle$. This implies that:

$$\begin{aligned}\frac{\partial F(\boldsymbol{v})}{\partial \alpha} &= \sum_{i=1}^{d}\left(\frac{\partial F(\boldsymbol{v})}{\partial v_i}\frac{\partial v_i}{\partial \alpha}\right)\\ &= \sum_{i=1}^{d}[w_i A'(z)(x_i - u_i)]\\ &= \langle \boldsymbol{x} - \boldsymbol{u},\ \boldsymbol{w}\rangle A'(z)\end{aligned}$$

We can therefore write

$$dF(\boldsymbol{v}) = \langle \boldsymbol{x} - \boldsymbol{u},\ \boldsymbol{w}\rangle A'(z)d\alpha,$$

and since $\langle \boldsymbol{x} - \boldsymbol{u},\ \boldsymbol{w}\rangle$ is a scalar, this yields

$$A'(z)d\alpha = \frac{dF(\boldsymbol{v})}{\langle \boldsymbol{x} - \boldsymbol{u},\ \boldsymbol{w}\rangle}$$

Using this equation the integral in the definition of $IG_i(x)$ can be written as

$$\begin{aligned}\int_{\alpha=0}^{1} \partial_i F(\boldsymbol{v}) d\alpha &= \int_{\alpha=0}^{1} w_i A'(z) d\alpha \\ &= \int_{\alpha=0}^{1} w_i \frac{dF(\boldsymbol{v})}{\langle \boldsymbol{x}-\boldsymbol{u},\, \boldsymbol{w}\rangle} \\ &= \frac{w_i}{\langle \boldsymbol{x}-\boldsymbol{u},\, \boldsymbol{w}\rangle} \int_{\alpha=0}^{1} dF(\boldsymbol{v}) \qquad \text{(E.35)} \\ &= \frac{w_i}{\langle \boldsymbol{x}-\boldsymbol{u},\, \boldsymbol{w}\rangle} [F(\boldsymbol{x}) - F(\boldsymbol{u})],\end{aligned}$$

where (E.35) follows from the fact that $(\boldsymbol{x}-\boldsymbol{u})$ and $\boldsymbol{w}$ do not depend on α. Therefore from the definition (12) of $\mathrm{IG}_i(\boldsymbol{x})$:

$$\mathrm{IG}_i(\boldsymbol{x}) = [F(\boldsymbol{x}) - F(\boldsymbol{u})] \frac{(x_i - u_i) w_i}{\langle \boldsymbol{x}-\boldsymbol{u},\, \boldsymbol{w}\rangle},$$

and this yields the expression (14) for $\mathrm{IG}(\boldsymbol{x})$. □

Appendix F Proof of Theorem 5.1

Theorem 5.1 (Equivalence of Stable IG and Adversarial Robustness)**.** *For loss functions $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w})$ satisfying Assumption LOSS-CVX, the augmented loss inside the expectation (16) equals the $\ell_\infty(\varepsilon)$-adversarial loss inside the expectation (2), i.e.*

$$\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) \;+\; \max_{||\,\boldsymbol{x}' - \boldsymbol{x}\,||_\infty \le \varepsilon} ||\,\mathrm{IG}^{\mathcal{L}_y}(\boldsymbol{x}, \boldsymbol{x}';\; \boldsymbol{w})||_1 = \max_{||\delta||_\infty \le \varepsilon} \mathcal{L}(\boldsymbol{x} + \delta, y; \boldsymbol{w}) \qquad (17)$$

Proof. Recall that Assumption LOSS-CVX implies $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) = g(-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)$ for some non-decreasing, differentiable, convex function g. Due to this special form of $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w})$, the function $\mathcal{L}_y$ is a differential function of $\langle \boldsymbol{w}, \boldsymbol{x}\rangle$, and by Lemma 4.1 the i'th component of the IG term in (17) is

$$\mathrm{IG}_i^{\mathcal{L}_y}(\boldsymbol{x}, \boldsymbol{x}';\; \boldsymbol{w}) = \frac{\boldsymbol{w}_i(\boldsymbol{x}' - \boldsymbol{x})_i}{\langle \boldsymbol{w}, \boldsymbol{x}' - \boldsymbol{x}\rangle} \cdot \big(g(-y\langle \boldsymbol{w}, \boldsymbol{x}'\rangle) - g(-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle),\big)$$

and if we let $\Delta = \boldsymbol{x}' - \boldsymbol{x}$ (which satisfies that $\|\Delta\|_\infty \le \varepsilon$), its absolute value can be written as

$$\frac{\big|g(-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle - y\langle \boldsymbol{w}, \Delta\rangle) \;-\; g(-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle)\big|}{|\langle \boldsymbol{w}, \Delta\rangle|} \cdot |\,\boldsymbol{w}_i\, \Delta_i|$$

Let $z = -y\langle \boldsymbol{w}, \boldsymbol{x}\rangle$ and $\delta = -y\langle \boldsymbol{w}, \Delta\rangle$, this is further simplified as $\frac{|g(z+\delta) - g(z)|}{|\delta|} |w_i \Delta_i|$. By Assumption LOSS-CVX, g is convex, and therefore the "chord slope" $[g(z+\delta) - g(z)]/\delta$ cannot decrease as δ is increased. In particular to maximize the ℓ_1-norm of the IG term in Eq (17), we can set δ to be largest possible value subject to the constraint $||\Delta||_\infty \le \varepsilon$, and we achieve this by setting $\Delta_i = -y\,\mathrm{sgn}(\boldsymbol{w}_i)\varepsilon$, for each dimension i. This yields $\delta = \|\,\boldsymbol{w}\,\|_1 \varepsilon$, and the second term on the LHS of (17) becomes

$$\begin{aligned}|g(z+\delta) - g(z)| \cdot \frac{\sum_i |\,\boldsymbol{w}_i\, \Delta_i|}{|\delta|} &= |g(z + \varepsilon \|\,\boldsymbol{w}\,\|_1) - g(z)| \cdot \frac{\sum_i |\,\boldsymbol{w}_i\,|\varepsilon}{\|\,\boldsymbol{w}\,\|_1 \varepsilon} \\ &= |g(z + \varepsilon \|\,\boldsymbol{w}\,\|_1) - g(z)| \\ &= g(z + \varepsilon \|\,\boldsymbol{w}\,\|_1) - g(z)\end{aligned}$$

where the last equality follows because g is nondecreasing. Since $\mathcal{L}(\boldsymbol{x}, y; \boldsymbol{w}) = g(z)$ by Assumption LOSS-CVX, the LHS of (17) simplifies to

$$g(-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle + \varepsilon \|\,\boldsymbol{w}\,\|_1),$$

and by Eq. (5), this is exactly the $\ell_\infty(\varepsilon)$-adversarial loss on the RHS of (17). □

Appendix G Aggregate IG Attribution over a Dataset.

Recall that in Section 4 we defined $\mathrm{IG}^F(\boldsymbol{x}, \boldsymbol{u};\ \boldsymbol{w})$ in Eq. (12) for a *single* input $\boldsymbol{x}$ (relative to a baseline input $\boldsymbol{u}$). This gives us a sense of the "importance" of each input feature in explaining a *specific* model prediction $F(\boldsymbol{x})$. Now we describe some ways to produce *aggregate* importance metrics over an entire dataset. For brevity let us simply write $\mathrm{IG}(\boldsymbol{x})$ and $\mathrm{IG}_i(\boldsymbol{x})$ and omit $F, \boldsymbol{u}$ and $\boldsymbol{w}$ since these are all fixed for a given model and a given dataset.

Note that in Eq. 12, $\boldsymbol{x}$ is assumed to be an input vector in "exploded" space, i.e., all categorical features are (explicitly or implicitly) one-hot encoded, and i is the position-index corresponding to either a specific numerical feature, or a categorical feature-*value*. Note that if i corresponds to a categorical feature-value, then for any input $\boldsymbol{x}$ where $x_i = 0$ (i.e. the corresponding categorical feature-value is not "active" for that input), $\mathrm{IG}_i(\boldsymbol{x}) = 0$. A natural definition of the overall importance of a feature (or feature-value) i for a given model F and dataset $\mathcal{D}$, is the average of $|\mathrm{IG}_i(\boldsymbol{x})|$ over all inputs $\boldsymbol{x} \in \mathcal{D}$, which we refer to as the **Feature Value Impact** $FV_i[\mathcal{D}]$. For a categorical feature with m possible values, we can further define its **Feature-Impact** (FI) as the *sum* of $|FV_i[\mathcal{D}]|$ over all i corresponding to possible values of this categorical feature.

The FI metric is particularly useful to gain an understanding of the aggregate importance of high-cardinality categorical features. For example we measure the feature-concentration of models trained on the MediaMath datasets (which have categorical features with cardinalities in the 100,000 range) in terms of the FI metric (see the FIL1 and FI1P metrics in Section H).

Appendix H Experiments on Real Datasets: Model Training and Evaluation Methodology

We describe here the common aspects of the training and evaluation methodology for the UCI and MediaMath datasets. Any variations specific to the datasets are described in the respective subsections. All the tasks we consider are *binary probability prediction* tasks, where the prediction target is a binary label $y \in \{\pm 1\}$, and the model output should be interpreted as the predicted probability that $y = 1$. (The specific implementations may actually use a 0/1 label instead, but we keep the -1/1 description here as it simplifies some of the analytical expressions). Our code is implemented in Python using the high-level TensorFlow Estimators and Dataset APIs.

It is important to note that all categorical variables are 1-hot encoded *prior* to being fed to the model-training and evaluation code. In other words when we apply the $\ell_\infty(\varepsilon)$-adversarial perturbation (5) to the input vector x, x represents the "flattened" feature-vector, i.e., all categorical features are 1-hot encoded [5] A reasonable question is whether such perturbations are semantically meaningful, and whether they represent legitimate perturbations by an adversary. One could also make the argument that a real adversary would only be able to perturb the original input vector, and so the set of allowed perturbations of x should be restricted to legitimate 1-hot encodings. Indeed some authors have considered this type of restriction in the domain of malware detection [33]. We set aside this issue in this paper, since our interest is more in the model-concentration effect of adversarial training, and less in robustness to real attacks.

Each dataset is divided into train and test subsets. For training on natural examples we use the FTRL optimizer in TensorFlow with ℓ_1-regularization strength (λ) set to 0. (We vary the λ to evaluate the effect of ℓ_1-regularization). Our results are substantially the same regardless of which optimizer we use, e.g. Adam, AdaGrad or simple SGD. We use FTRL mainly because in TensorFlow the FTRL optimizer has an optional λ argument that controls the strength of ℓ_1-regularization. Although our results are qualitatively similar with other optimizers, the best results are obtained using the FTRL optimizer. All model weights are initialized to zero in case of the toy UCI datasets, whereas they are initialized using a Gaussian initializer (with mean 0 and variance 0.001) in the case of the MediaMath ad response-prediction models. Once again our results remain the same whether we use zero or Gaussian initializers. For $\ell_\infty(\varepsilon)$-adversarial training we also use the FTRL optimizer, except that in each mini-batch the examples x are perturbed according to the worst-case perturbation given by Eq. (5).

[5] In practice we use feature-hashing when the cardinality of a categorical feature is large, but this is an implementation detail, and we can conceptually think of the feature as having been explicitly 1-hot encoded

Once a model is trained (adversarially or naturally) we compute two types of metrics:

- An *ML performance* metric, the AUC-ROC (Area Under the ROC Curve) on the held-out *natural test dataset*.
- A few *feature concentration* metrics, defined as follows, where the linear model weight-vector is $w \in \mathbb{R}^d$ (and d is the dimension of the *exploded* feature-space, i.e. after 1-hot encoding).

  WL1: $||w||_1/||w||_\infty$, which is a measure of the overall magnitude of the weights, scaled by the biggest absolute weight. Note that if we multiply all weights by a constant factor, then WtsL1 does not change.

  W1P: The percent of the weights in w whose absolute value is at least 1% of the maximum absolute weight. This can be thought of as a measure of how many feature-weights are "significant", where the threshold of significance is 1% of the biggest absolute weight.

  FIL1: $||FI||_1/||FI||_\infty$, where FI stands for the vector of Feature Impact values $\text{FI}_i[\mathcal{D}]$ (defined in Sec. G), and i ranges over the dimensions in the *original* feature-space (i.e. before 1-hot encoding), and the dataset $\mathcal{D}$ is the *natural training dataset*.

  FI1P: The percent of components of FI (which are all positive by definition) that are at least 1% of the biggest component of FI, again over the natural training dataset.

In all cases the baseline vector $\boldsymbol{u}$ in all IG computations (using (12)) is the all-zeros vector. In the various tables of results, we use the abbreviation `nat` to refer to metrics for the naturally-trained model, and `adv` to refer to metrics for the adversarially-trained model.

Appendix I Experiments on some UCI datasets

We now describe experiments on datasets in the popular UCI repository of ML datasets. We show details of experiments on two of these datasets:

- The **mushroom** dataset consists of 8142 instances, each of which corresponds to a different mushroom species, and has 22 categorical features (and no numerical features), whose cardinalities are all under 10. The task is to classify an instance as edible (label=1) or not (label=0).
- The **spambase** dataset consists of 4601 instances with 57 numerical attributes (and no categorical ones). The instances are various numerical features of a specific email, and the task is classify the email as spam (label = 1) or not (label = 0).

In each case our training and testing methodology is the following: we split the dataset randomly into a 70% training dataset and a 30% test dataset. We train a logistic model using the FTRL optimizer (where the strength λ of ℓ_1 regularization can be adjusted), with all weights initialized to zero. Prior to training and testing, we standardize each numerical feature using its mean and standard deviation in the training dataset. For adversarial training we assume each input x is perturbed by an adversary who is allowed to shift x by a vector δ whose ℓ_∞-norm is bounded by ε. In particular for a given ε we use the closed-form worst-case perturbation formula in Eq. (5), and we set the ℓ_1 regularization parameter λ to 0.

I.1 Mushroom Dataset

In Figure I.2 we show how varying the ε bound on an ℓ_∞ adversary impacts the weight-concentration (measured by W1P, defined in Section H) and the AUC, in order to demonstrate that there is a range of ε values that represents a "goldilocks" zone where there is a significant feature-concentration effect while maintaining the (test-set) AUC within 0.01 of the naturally-trained AUC. With adversarial training using $\varepsilon = 1.6$ the W1P value is only 12% (compared to almost 97% for a naturally-trained model) while AUC (on natural test data) is 0.99, a drop of only 0.01 compared to a naturally trained model.

We show in Figure I.3 how varying the ℓ_1-regularization strength parameter λ impacts AUC and W1P (in natural training, i.e. $\varepsilon = 0$). The figure shows that the highest λ value that maintains an

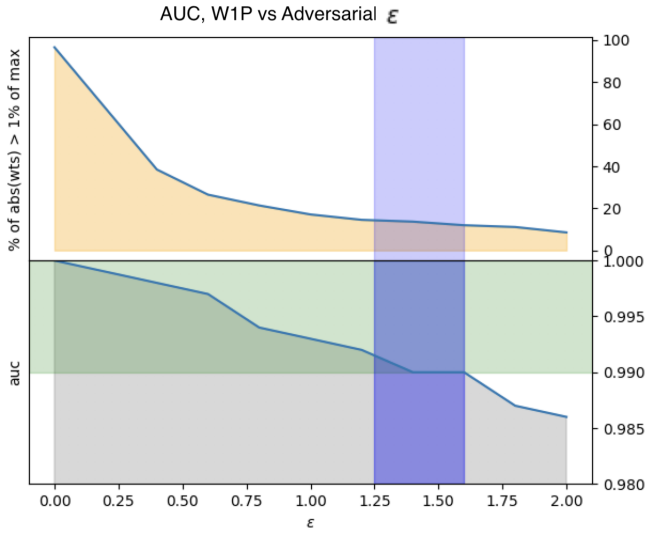


Figure I.2: Variation of W1P and test AUC with increasing ε bound on ℓ_∞ perturbations in adversarial training, for the mushroom dataset. Note that $\varepsilon = 0$ corresponds to natural training, which results in AUC=1.00. The horizontal green band from AUC=0.99 to AUC=1.00 shows the range of AUCs within 0.01 of AUC of the naturally-trained mode. The blue vertical band represents the goldilocks zone of ε values (roughly 1.25 to 1.6) that are high enough to produce significant model-concentration, yet low enough that AUC is maintained within the green band.

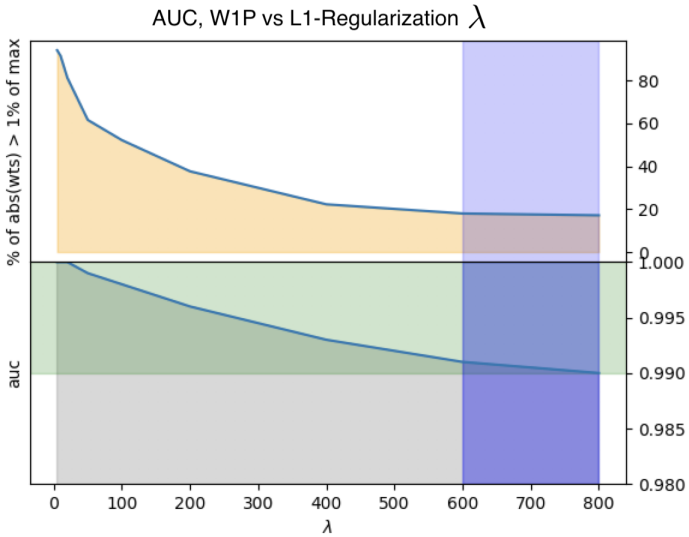


Figure I.3: Variation of W1P and test AUC for naturally-trained models, with increasing ℓ_1-regularization parameter λ. Even with a $\lambda = 600$, the W1P concentration metric is around 18%, considerably higher than the 12% achieved with adversarial training.

AUC above 0.99 is 600, yet it achieves a W1P of around 18%, still significantly higher than the 12% achieved with adversarial training using $\varepsilon = 1.6$.

We now want to show the contrast between the weights learned by natural training and adversarial training with $\varepsilon = 1.6$. Since all features in this dataset are categorical, many with cardinalities close to 10, there are too many features in the "exploded" space to allow a clean display, so we instead look at the aggregate Feature Impact (FI) over the natural training dataset, see Figure I.4. It is worth noting that several features that have a significant impact on the naturally-trained model have essentially no impact on the adversarially trained model. Figures I.5 and I.6 show the drop-off curves of the feature-weights and FI metrics respectively, and once again it is evident that the drop offs are much steeper with adversarial training compared to natural training.

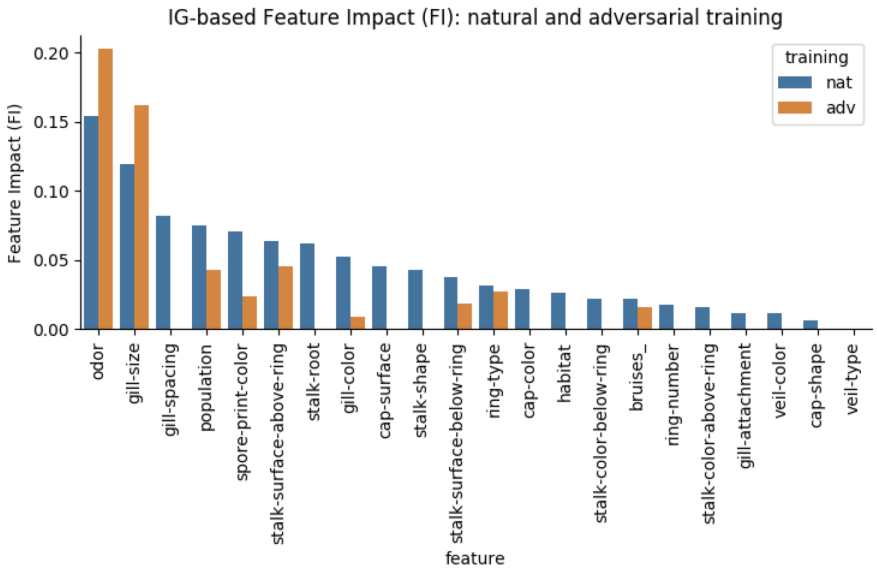


Figure I.4: Comparison of aggregate Feature Impact (FI) for a naturally-trained model, and an adversarially-trained model with $\varepsilon = 1.6$, on the mushroom dataset. The features are arranged left to right in decreasing order of the FI value in the naturally-trained model.

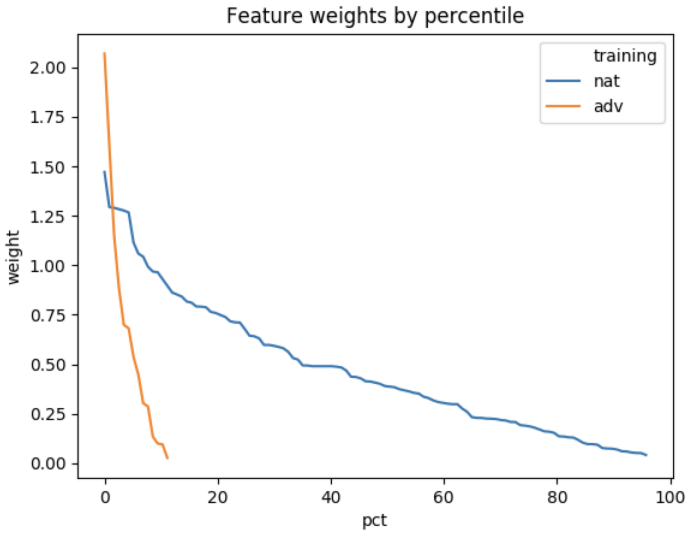


Figure I.5: Comparison of drop-off of absolute feature weights in natural and adversarial training, on the mushroom dataset. In each curve, a point (p, w) indicates that p percent of the absolute weights are larger than w. Both curves are truncated when the absolute weight reaches 1% of the highest absolute weight in the model. The adversarially trained model has a much steeper absolute weight drop-off, with only 12% being above the 1% threshold, compared to 95% with natural training.

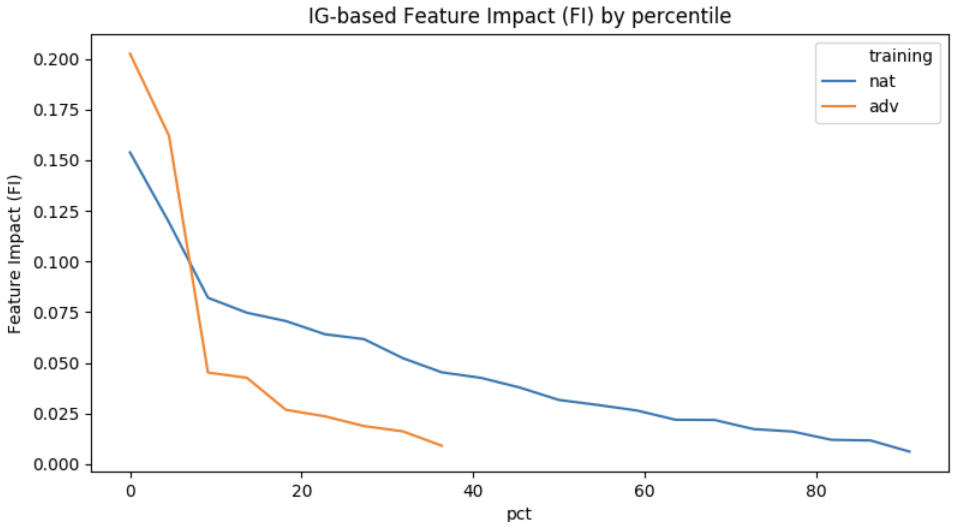


Figure I.6: Comparison of drop-off of aggregate Feature Impact FI (measured over the natural training dataset) resulting from natural and adversarial training, for the mushroom dataset. In each curve, a point (p, w) indicates that p percent of the FI values are larger than w. Both curves are truncated when the FI reaches 1% of the highest FI in the model. The adversarially trained model has a much steeper FI drop-off, with only 36% being above the 1% threshold, compared to 90% with natural training.

I.2 Spambase dataset

As with the mushroom dataset, we start by looking at the impact of varying ε (the adversarial ℓ_∞ bound) on the concentration metric W1P, see Figure I.7. From the chart it is apparent that an adversarially trained model $\varepsilon = 0.6$ achieves a concentration metric W1P of around 53% (meaning that 53% of the model's absolute weights are within 1% of the biggest absolute weight), compared to

96% for a naturally-trained model, and yet yields an AUC (on natural test data) of 0.964, which is no more than 0.01 worse than the AUC of 0.974 for a naturally-trained model.

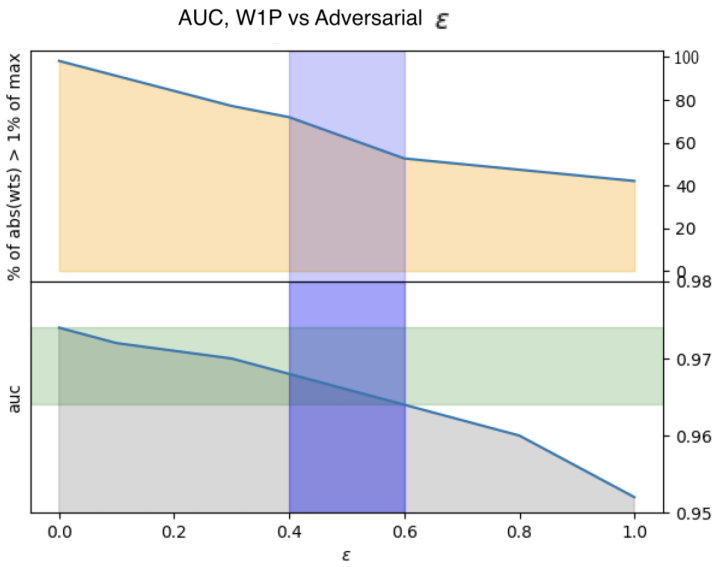


Figure I.7: Variation of W1P and natural test AUC with increasing ε bound on ℓ_∞ perturbations in adversarial training, for the spambase dataset. Note that $\varepsilon = 0$ corresponds to natural training, which results in AUC=0.974. The horizontal green band from AUC=0.964 to AUC=0.974 shows the range of AUCs within 0.01 of AUC of the naturally-trained mode. The blue vertical band represents the range of ε values (0.4 to 0.6) that are high enough to produce significant model-concentration, yet low enough that AUC is maintained within the green band.

Figure I.8 shows that this tradeoff between model-concentration and AUC cannot be achieved by using ℓ_1-regularization.

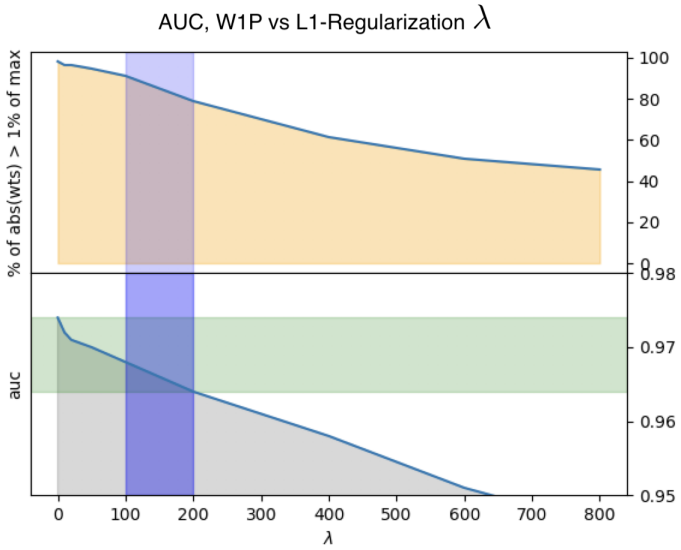


Figure I.8: Variation of W1P and natural test AUC for naturally-trained models, with increasing ℓ_1-regularization parameter λ, for the spambase dataset. Even with a $\lambda = 200$, the W1P concentration metric is as high as 80% (significantly worse than the 53% for adversarial training), and any higher value of λ pushes the AUC to a level worse than 0.01 below the naturally-trained AUC with no regularization.

We now fix $\varepsilon = 0.6$ for adversarial training and show in Figures I.9 and I.10 the bar-plots comparing the weights and Feature-Impacts (FI) respectively, between naturally-trained and adversarially-trained models. Since all features are numerical the number of weights is the same as the number of FIs. However the distribution of the FIs does not follow that of the weights.

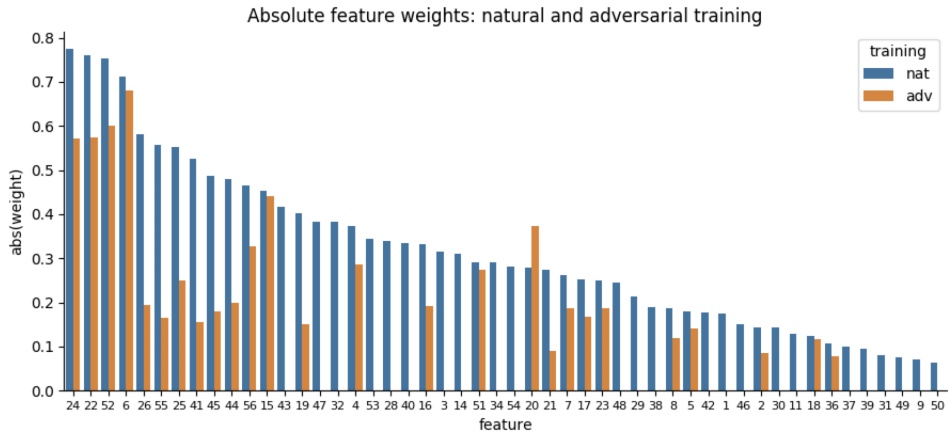


Figure I.9: Comparison of absolute model weights for a naturally-trained model, and an adversarially-trained model with $\varepsilon = 0.6$, on the spambase dataset. The features are arranged left to right in decreasing order of their absolute weight in the naturally-trained model. To avoid clutter, we show only features that have an absolute weight at least 8% of the highest weight (across both models). (The features in this figure and the next are identified by integers.)

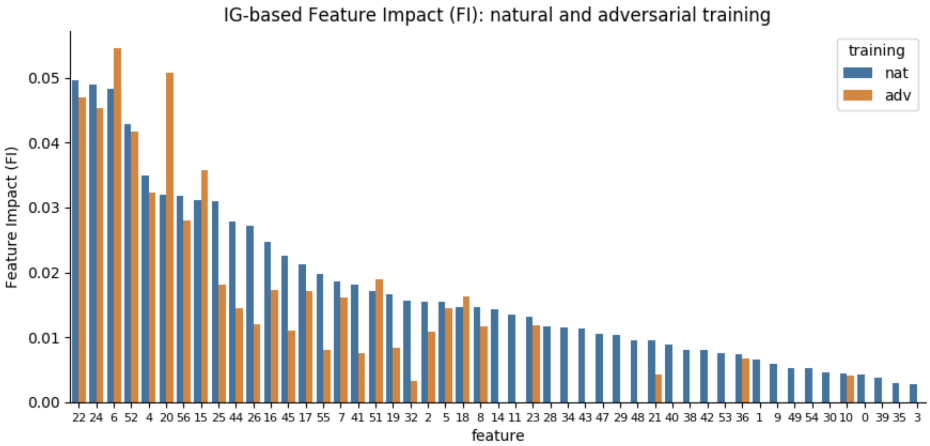


Figure I.10: Comparison of aggregate Feature Impact (FI) for a naturally-trained model, and an adversarially-trained model with $\varepsilon = 0.6$, on the spambase dataset. The features are arranged left to right in decreasing order of their FI in the naturally-trained model. To avoid clutter, we show only features that have an FI at least 5% of the highest FI (across both models).

Figures I.11 and I.12 contrast the drop-off curves of feature-weights and Feature Impacts (FI) respectively between naturally-trained and adversarially-trained models.

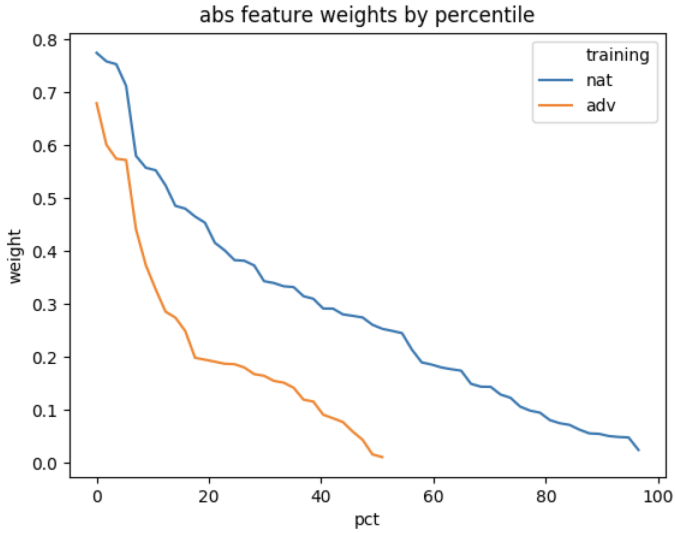


Figure I.11: Comparison of drop-off of absolute feature weights in natural and adversarial training, for the spambase dataset. In each curve, a point (p, w) indicates that p percent of the absolute weights are larger than w. Both curves are truncated when the absolute weight reaches 1% of the highest absolute weight in the model. The adversarially trained model has a much steeper absolute weight drop-off, with only 52% being above the 1% threshold, compared to 96% with natural training.

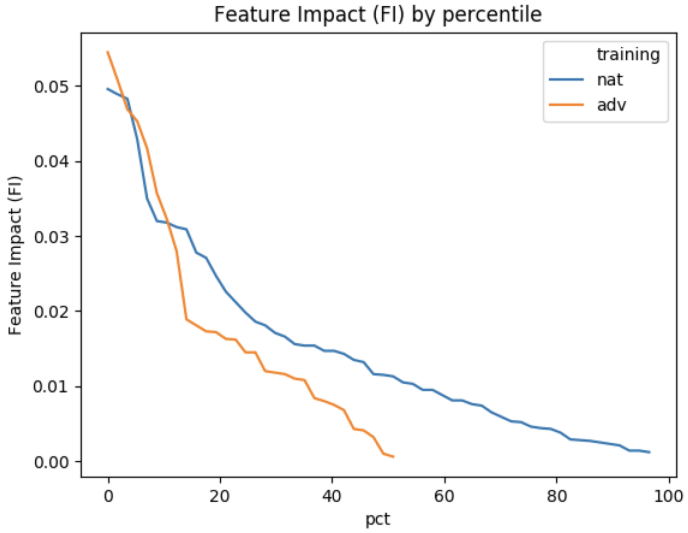


Figure I.12: Comparison of drop-off of aggregate Feature Impact (FI) (computed over the natural training dataset) for a naturally-trained and adversarially-trained model. In each curve, a point (p, w) indicates that p percent of the FI values are larger than w. Both curves are truncated when the FI reaches 1% of the highest FI in the model. The adversarially trained model has a much steeper FI drop-off, with only 52% being above the 1% threshold, compared to 96% with natural training.

Appendix J Experiments with MediaMath Datasets: Ad Conversion Prediction

MediaMath provides a software platform that operates a real-time bidding (RTB) engine which responds to bid-opportunities sent by ad-exchanges. The RTB engine bids on behalf of advertisers who set up ad-campaigns on the platform. A key component in determining bid prices is a prediction of the probability that a consumer exposed to the advertiser's campaign would subsequently perform

a certain designated action (called a "conversion"). MediaMath currently trains a logistic regression model for each campaign to generate these conversion probability predictions. The models are trained on a dataset collected over a number of days, where each record contains various features related to the ad opportunity (such as device type, browser, location, time of day etc), as well as a 0/1 label indicating whether or not a conversion occurred subsequent to ad exposure. The model for each campaign is trained on a sequence of 18 days of data, and validated/tested on the subsequent 3 days of data. The total number of records in each dataset can range from half million to 50 million depending on the campaign. Each record has around 100 features, mostly categorical, and some (such as "siteID") have cardinalities as high as 100,000, and so the dimension of the exploded feature-space (i.e. after 1-hot encoding) is on the order of 400,000. (We use feature-hashing rather than explicit 1-hot encoding to map some of the high-cardinality features to a lower-dimensional vector, but the net effect is similar to 1-hot encoding, except that now each dimension in the 1-hot encoding vector may correspond to multiple features, due to hash collisions)

Given the extremely high dimensionality of the exploded feature-space, it is of considerable practical importance to understand which features have a truly significant impact on the predictions. Specifically, we wish to explore whether adversarial training can yield models that have significantly better feature concentration, while maintaining the AUC within say 0.01 of the naturally-trained model. We have seen strong evidence that this is indeed possible, on the UCI datasets (Section I). We show below that we see a similar phenomenon in the conversion-prediction models.

To study the impact of $\ell_\infty(\varepsilon)$ adversarial training, we performed experiments with a wide range of values of ε and found that for most campaigns, adversarial training with $\varepsilon = 0.01$ or $\varepsilon = 0.001$ results in feature-concentrations significantly better than with natural training, while maintaining AUC (on the validation set) within 0.01 of the AUC of a naturally-trained model. We also experimented with keeping $\varepsilon = 0$ and varying the ℓ_1-regularization parameter λ in the FTRL optimizer, and found that any $\lambda > 0.2$ significantly lowers the AUC of the resulting model, and lower λ values do not yield a feature-concentration as strong as that achieved by adversarial training. Indeed we find that the effects of adversarial training and ℓ_1 regularization are complementary: when an appropriate value of ε is used in conjunction with say $\lambda = 0.01$, we find that ℓ_1 regularization helps to "clean" up the very low feature-weights produced by adversarial training by pushing them to zero.

Table J.1 shows a summary of results on 9 campaigns [6]. In some cases the AUC of the adversarially-trained model is better than that of the naturally-trained model. Recall that the W1P metric measures what percent of dimensions (in the exploded space, after 1-hot encoding) have absolute weights at least 1 percent of the highest absolute weight. Since most features are categorical, W1P is therefore a measure of what percent of *feature-values* are significant to the model. This metric (as well as WL1) falls drastically with adversarial training in all cases, which indicates that several of the feature-values are simply not relevant to predicting the label. There is thus a potentially massive *model-compression* that can be done, and this can have benefits in storing, updating and serving models (MediaMath periodically trains around 40,000 models). Table J.1 also shows the FI1P and FIL1 metrics, which are aggregate feature-impact concentration metrics over the natural training dataset. Note that these are at the *feature* level and not feature-value level. Since the FI measure of a categorical feature aggregates the FVI metric over all values of this feature, the drop in this metric (when we go from natural to adversarial training) is not as dramatic as in the case of WL1 or W1P (and sometimes these are higher than with natural training).

[6] All campaign IDs and feature names are masked for client confidentiality reasons

Campaign	training	AUC	W1P	WL1	FI1P	FIL1
285	nat	0.560	12.93	165.05	0.78	5.80
	adv ($\varepsilon = 0.01$)	0.556	0.39	13.27	0.37	2.87
479	nat	0.697	11.23	92.02	3.20	12.52
	adv ($\varepsilon = 0.001$)	0.694	6.17	66.96	3.13	12.34
735	nat	0.758	12.20	220.97	1.87	21.82
	adv ($\varepsilon = 0.01$)	0.765	0.51	21.02	0.75	16.36
622	nat	0.565	27.11	110.12	6.63	5.73
	adv ($\varepsilon = 0.01$)	0.561	4.21	18.18	2.87	3.55
594	nat	0.702	16.91	172.55	0.86	12.54
	adv ($\varepsilon = 0.001$)	0.702	1.09	19.97	0.77	11.19
473	nat	0.683	28.02	177.36	3.14	14.69
	adv ($\varepsilon = 0.001$)	0.673	3.58	55.68	2.84	15.69
070	nat	0.622	18.53	158.15	4.94	26.21
	adv ($\varepsilon = 0.001$)	0.625	7.55	107.63	4.46	24.00
645	nat	0.573	16.26	251.37	2.78	31.78
	adv ($\varepsilon = 0.01$)	0.627	1.07	34.45	1.12	10.60
733	nat	0.658	27.35	203.73	4.04	11.36
	adv ($\varepsilon = 0.001$)	0.667	9.91	108.03	4.13	11.60

Table J.1: Comparison of AUC and feature-concentration between natural and adversarial training on 9 advertising campaigns. The 4 concentration metrics are defined in Section H. Note that while the AUC is computed on the natural validation set, the concentration metrics FI1P and FIL1 are computed on the natural training dataset. In some campaigns, such as 285, 735 the W1P metric improves by a factor of more than 24.

To illustrate the effect of adversarial training in more detail, we focus on campaign number 735 (the bottom row in Table J.1) and compare the results from natural training and adversarial training (with $\varepsilon = 0.01$). Figure J.13 compares the Feature Impact (FI) values between these models; Figure J.14 compares the feature-weights drop-off curves of these models; and Figure J.15 compares the FI drop-off curves.

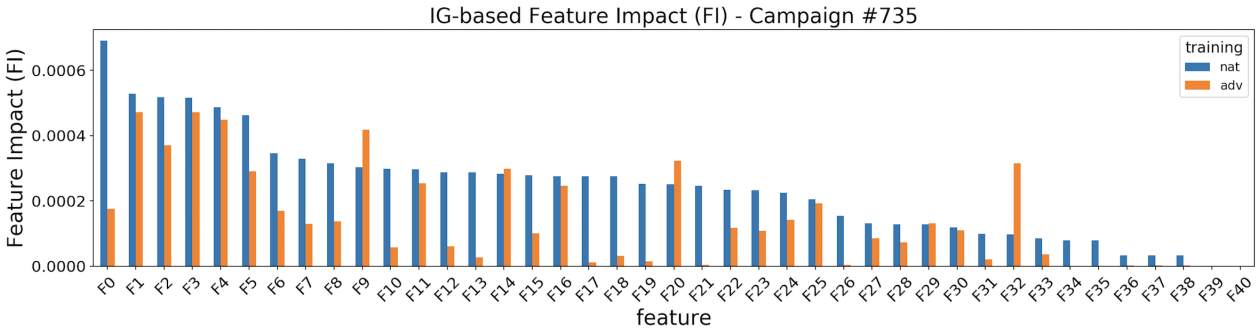


Figure J.13: Comparison of aggregate Feature Impact (FI) for a naturally-trained model, and an adversarially-trained model with $\varepsilon = 0.01$, on the dataset for Campaign 735. The features are arranged left to right in decreasing order of their FI in the naturally-trained model.

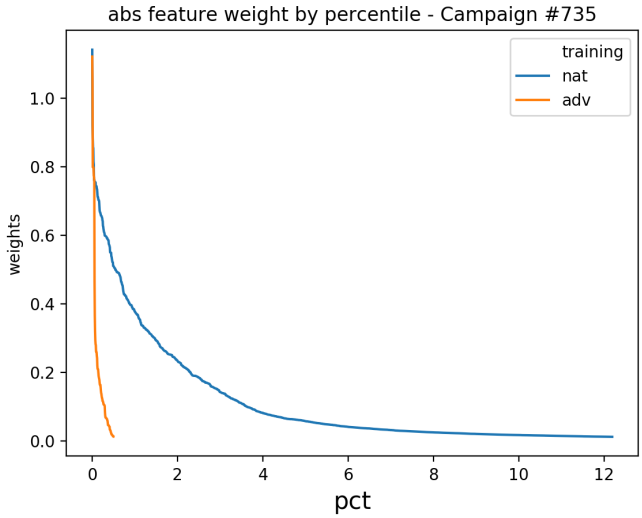


Figure J.14: Comparison of drop-off of absolute feature weights in natural and adversarial training (with $\varepsilon = 0.01$), for Campaign 735. In each curve, a point (p, w) indicates that p percent of the absolute weights are larger than w. Each curve is truncated when the absolute weight reaches 1% of the highest absolute weight in the respective model. The adversarially trained model has a much steeper absolute weight drop-off, with only 0.5% being above the 1% threshold, compared to 12% with natural training (this is consistent with Table J.1).

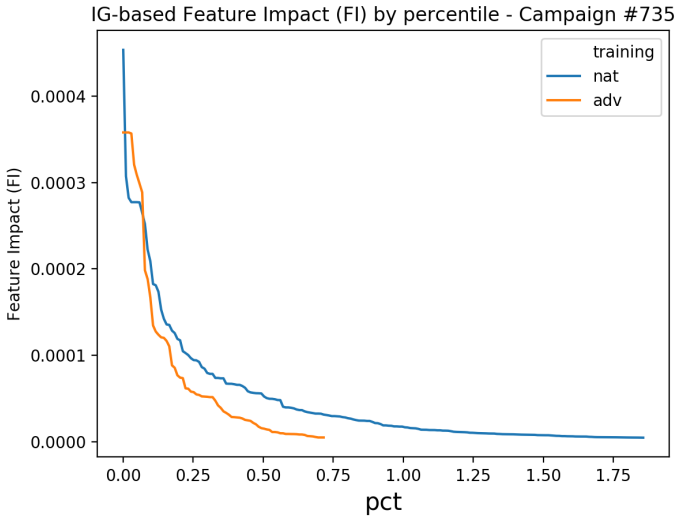


Figure J.15: Comparison of drop-off of aggregate Feature Impact (FI) (computed over the natural training dataset) for a naturally-trained and adversarially-trained model (with $\varepsilon = 0.01$), for Campaign 735. In each curve, a point (p, w) indicates that p percent of the FI values are larger than w. Each curve is truncated when the FI reaches 1% of the highest FI in the respective model. The adversarially trained model has a much steeper FI drop-off, with only 0.75% being above the 1% threshold, compared to 1.87% with natural training. (this is consistent with Table J.1).

Figures J.16 and J.17 contrast the ability of adversarial training and ℓ_1 regularization to improve model concentration while maintaining AUC (on natural test data): adversarial training with $\varepsilon 0.01$ improves the concentration metric W1P to as low as 0.5% (compared to 12% for a naturally trained model, an improvement factor as high as 24), and yet achieves an AUC slightly higher than with natural training. On the other hand with ℓ_1 regularization, using a strength of $\lambda = 0.2$ improves the concentration to 5% (significantly worse than 0.5% for adversarial training) and slightly improves

upon the naturally-trained AUC, but any higher value of λ significantly degrades the AUC, and the W1P concentration metric does not go below 2.5%.

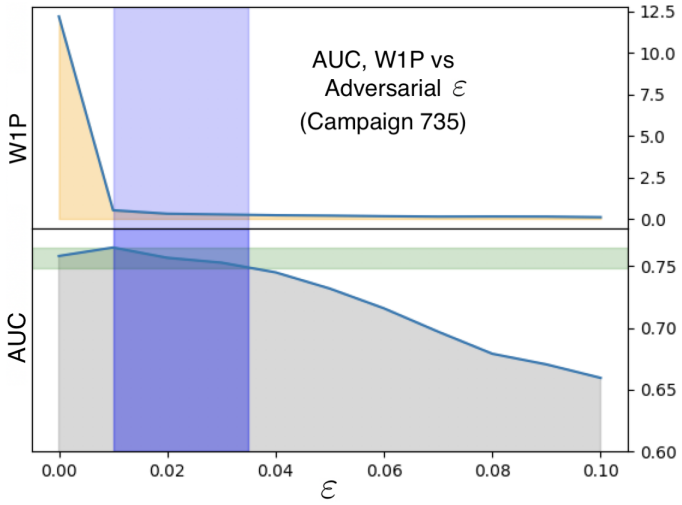


Figure J.16: Variation of W1P and natural test AUC with increasing ε bound on ℓ_∞ perturbations in adversarial training, for campaign 735. Note that $\varepsilon = 0$ corresponds to natural training, which results in AUC=0.758. The horizontal green band lower-bounded by AUC=0.748 represents the range of AUCs within 0.01 of AUC of the naturally-trained mode. The blue vertical band represents the range of ε values (0.01 to 0.03) that are high enough to produce significant model-concentration (i.e. reduction in W1P), yet low enough that AUC is maintained within the green band. For these values of ε, the W1P metric is under 0.5%, meaning that only 0.5% of absolute weights are within 1% of the highest absolute weight.

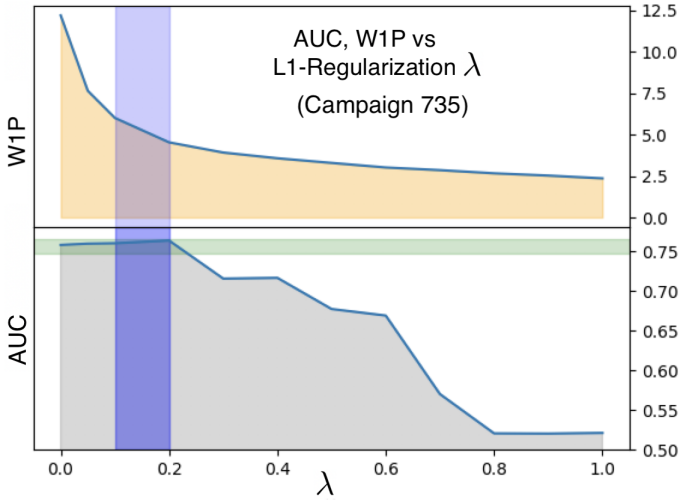


Figure J.17: Variation of W1P and natural test AUC for naturally-trained models, with increasing ℓ_1-regularization parameter λ, for campaign 735. For a $\lambda = 0.2$, the W1P concentration metric is as high as 5% (significantly worse than the 0.5% for adversarial training), and any higher value of λ significantly degrades the AUC.