Neurocomputing 307 (2018) 195–204

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Understanding adversarial training: Increasing local stability of supervised models through robust optimization

Uri Shaham[a,*], Yutaro Yamada[b], Sahand Negahban[b]

[a] Center for Outcome Research, Yale University, 200 Church st., New Haven, CT 06510, United States
[b] Department of Statistics, Yale University, 24 Hillhouse st., New Haven, CT 06511, United States

ARTICLE INFO

Article history:
Received 27 August 2017
Revised 2 April 2018
Accepted 6 April 2018
Available online 4 May 2018

Communicated by Dacheng Tao

Keywords:
Adversarial examples
Robust optimization
Non-parametric supervised models
Deep learning

ABSTRACT

We show that adversarial training of supervised learning models is in fact a robust optimization procedure. To do this, we establish a general framework for increasing local stability of supervised learning models using robust optimization. The framework is general and broadly applicable to differentiable non-parametric models, e.g., Artificial Neural Networks (ANNs). Using an alternating minimization-maximization procedure, the loss of the model is minimized with respect to perturbed examples that are generated at each parameter update, rather than with respect to the original training data. Our proposed framework generalizes adversarial training, as well as previous approaches for increasing local stability of ANNs. Experimental results reveal that our approach increases the robustness of the network to existing adversarial examples, while making it harder to generate new ones. Furthermore, our algorithm improves the accuracy of the networks also on the original test data.

1. Introduction

Machine learning models might be very unstable locally, and have significantly different outputs on inputs which only slightly differ from one another. This may be the case even for models with high generalization ability (estimated by performance on test data). For example, Szegedy et al. [34], showed that highly performing vision ANNs mis-classify examples that have only barely perceivable (by a human eye) differences from correctly classified examples. Such examples are called *adversarial examples*, and although usually mentioned in the context of ANNs, they are not unique to this model family.

Adversarial examples do not tend to exist naturally in training and test data. Yet, the local instability manifested by their existence is disturbing, for several reasons. First, state-of-the-art models, including, for example, ANNs with super-human performance, may assign examples which are indistinguishable in the natural "human eye" metric to different classes, indicating that the models are far from learning the true class 'concept'. Second, adversarial examples can be generated in structured and automated ways. Third, it has been shown that different models with different architectures which are trained on different training sets tend to misclassify the same adversarial examples in a similar fashion. This can be used to perform attacks on models by making them fail easily and consistently [13] and poses serious security issues.

In recent years, the field of adversarial attacks and defenses has become a highly active research area, primarily associated with deep learning, and many attacks and defenses have been proposed [41]. Improving robustness of neural nets to adversarial examples by adding such examples to the training data is arguably among the first and fundamental defense techniques against adversarial attacks, see, for example, [12,34]. We refer to the usage of adversarial examples during training as "adversarial training". To the best of our knowledge, despite making intuitive sense and yielding impressive empirical results, a more rigor mathematical understanding of adversarial training is lacking. The goal of this manuscript, which appeared in pre-print in 2015, is to provide a framework that yields a full theoretical understanding of adversarial training, as well as new optimization schemes, based on robust optimization. Specifically, we show that generating and using adversarial examples during training of supervised machine learning models (and ANNs in particular) can be derived from the powerful notion of robust optimization, which has many applications in machine learning and is closely related to regularization. We propose a general algorithm for robustification of non-parametric machine learning models, and show that it generalizes several previously proposed approaches for training of ANNs.

Essentially, our algorithm increases the stability of supervised models with respect to perturbations in the input data, through

* Corresponding author.
E-mail addresses: uri.shaham@yale.edu (U. Shaham), yutaro.yamada@yale.edu (Y. Yamada), sahand.negahban@yale.edu (S. Negahban).

https://doi.org/10.1016/j.neucom.2018.04.027
0925-2312/

an iterative minimization-maximization procedure, in which the network parameters are updated with respect to worst-case data, rather than to the original training data. Furthermore, we show connections between our method and existing methods for generating adversarial examples and adversarial training, demonstrating that those methods are special instances of the robust optimization framework. This point yields a principled connection highlighting the fact that the existing adversarial training methods aim to robustify the parameter optimization process. The main application we consider in this manuscript is training of ANNs, to which our algorithm applies naturally and has an efficient implementation. Yet, the algorithm is general and can be applied to any non-parametric supervised model that is trained using gradient-based optimization.

The structure of this paper is as follows: in Section 2 we provide background on adversarial examples and robust optimization. In Section 3, we present our training framework, some of its possible variants and its practical version. Experimental results on ANNs and boosting models are given in Section 4. Some related works are mentioned in Section 5. Section 6 briefly concludes this manuscript.

2. Preliminaries

In this section, we provide elementary background on adversarial examples and robust optimization, required to justify our proposed approach.

2.1. Notation

We denote a labeled training set by $\{(x_i, y_i)\}_{i=1}^m$ where $x_i \in \mathbb{R}^d$ is a set of features and $y_i \in \{1, \ldots, K\}$ is a label. The loss of a model with parameters θ on (x, y) is denoted by $J(\theta, x, y)$ and is a function that quantifies the goodness-of-fit between the parameters θ and the observations (x, y). When holding θ and y fixed and viewing $J(\theta, x, y)$ as a function of x we occasionally write $J_{\theta, y}(x)$. $\Delta_x \in \mathbb{R}^d$ corresponds to a small additive adversarial perturbation, that is to be added to x. By *adversarial example* we refer to the perturbed example, i.e., $\tilde{x}_i = x + \Delta_x$, along with the original label y. We denote the ℓ_p norm for $1 \le p < \infty$ to be $\|x\|_p^p = \sum_{j=1}^d |x(i)|^p$ and denote the ℓ_∞ norm of a vector x to be $\|x\|_\infty = \max_i\{|x(i)|\}$. Given two vectors x and y, the Euclidean inner-product is denoted $\langle x, y\rangle = x^T y = \sum_i x_i y_i$. We denote the gradient of a function $f(x, y)$ with respect to the vector x by $\nabla_x f(x, y)$.

2.2. Adversarial examples

To this day, adversarial examples were primarily discussed in the context of ANNs. They were first introduced by Szegedy et al. [34], who generated an adversarial perturbation Δ_x for a given training point (x, y) by using L-BFGS [38] to solve the box-constrained optimization problem

$$\min_{\Delta_x} c\|\Delta_x\|_2 + J(\theta, x + \Delta_x, y')$$
$$\text{subject to } x + \Delta_x \in [0, 1]^d,$$

and $y' \neq y$. The fundamental idea here is to construct a small perturbation of the data point x in order to force the method to misclassify the training example x with some incorrect label y'.

Goodfellow et al. [12] point out that when the dimension d is large, changing each entry of x by a small value ϵ yields a perturbation Δ_x (such that $\|\Delta_x\|_\infty = \epsilon$), which can significantly change the inner product $w^T x$ of x with a weight vector w. They propose to use an adversarial perturbation defined by

$$\Delta_x = \epsilon \text{sign}(\nabla_x J(\theta, x, y)). \quad (1)$$

Eq. (1) is also known as the "fast gradient sign (FGS) method". We present a simple alternative formulation of the problem to naturally show how the adversarial perturbation in (1) was obtained. If we take a first-order approximation of the loss function around the true training example x with a small perturbation Δ_x

$$J_{\theta,y}(x + \Delta_x) \approx J_{\theta,y}(x) + \langle \nabla J_{\theta,y}(x), \Delta_x\rangle,$$

and maximize the right hand size with respect to Δ_x restricted to an ℓ_∞ ball of radius ϵ, we see that the choice that maximizes the right-hand side is exactly the quantity in Eq. (1). Replacing the ℓ_∞ ball with a ℓ_2 ball yields a perturbation in the direction of the gradient, coined "fast gradient value" [30]. Since in the case of ANN, the gradient $\nabla_x J(\theta, x, y)$ can be computed efficiently using back-propagation [31], this approach for generating adversarial examples is rather fast. In the sequel we will show how the above computation is an example of the framework that we present in this manuscript.

It is reported in [12,34] that adversarial examples that were generated for a specific network were mis-classified in a similar fashion by other networks, with possibly different architectures and using different subsets of the data for training. This phenomenon is known as the "transferability" of adversarial examples [26], and is used to create "black-box" attacks (i.e., where the attacker has no access to the parameters and gradients of the target network).

Goodfellow et al. [12] propose the following adversarial training loss function:

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha) J(\theta, x + \Delta_x, y), \quad (2)$$

with Δ_x as in Eq. (1). They report that the resulting net had improved test set accuracy, as well as better performance on new adversarial examples. They further give intuitive explanations of this training procedure being an adversary game, and a min-max optimization over ℓ_∞ balls. In Section 3.2, we will attempt to make the second interpretation rigor, by deriving it from a Robust Optimization framework.

2.3. Robustification through random perturbations

Let $f(x)$ be the output of a machine learning model on input x. A possible approach to robustification of models is to smooth f (see, for example, [24]). A naive approach to obtain such smoothing is to perturb the model input x at test time. To see this, consider a case where at test time, given input x, the model outputs $f(x + w)$ where $w \sim N(0, \sigma^2 I)$. In this case

$$\mathbb{E}_{w\sim N(0,\sigma^2 I)} f(x + w) \propto \int_w \exp\left(-\frac{\|w\|^2}{\sigma^2}\right) f(x + w) dw$$
$$= f * N(0, \sigma^2 I), .$$

i.e, in expectation, the model output is convolved with a Gaussian. In Section 4, we will demonstrate experimentally that such mechanism indeed improves the stability of a neural net to adversarial examples. However, the approach presented in this manuscript, performs significantly better. In our approach, the robustification of the model is obtained through a modified training procedure, which is based on *robust optimization*. In the remainder of this section, we therefore turn to describe the main ideas of robust optimization, and several applications of it in machine learning.

2.4. Robust optimization

Solutions to optimization problems can be very sensitive to small perturbations in the input data of the optimization problem, in the sense that an optimal solution given the current data may turn into a highly sub-optimal or even infeasible solution given a