Adversarial Robustness: Duality-based Verification

Felix Assion, Matteo Koczorek

Seminar: Machine Learning in Robotics

November 15, 2018

Abstract

It has been proven that all state-of-the-art Machine Learning-modules are vulnerable to adversarial perturbations, i.e. vulnerable to carefully crafted perturbations which are designed to cause mistakes when added to benign input data points. As a consequence, researchers are eager to find ways to defend against these malicious inputs and to increase adversarial robustness of ML-algorithms in general. One particularly interesting sub-branch of defense research is the field of duality-based verification techniques. In this seminar work, we will take a closer look at the key ideas and concepts in this area. The content of this paper strongly relies on the recent publication "Training Verified Learners with Learned Verifiers" by K. Dvijotham et al. (2018).

1 Introduction

The recent impressive advances in the field of Machine Learning (ML), especially in the area of Deep Learning, have lead to an increased interest of companies to apply ML-algorithms in safety- and security-critical application contexts. Examples are the integration of CNN-based semantic segmentation into autonomous cars or neural networks for medical diagnosis. Obviously, these new application contexts have extremely high demands on quality and quality assurance. Most of the time it is not enough to prove high accuracy on well-crafted training, test and validation data sets. Besides handling the common cases of the given data distribution, the trained ML-module should also be able to deal with rare corner cases or even with maliciously crafted input points [1].

Unfortunately, the last few years have shown that current state-of-the-art ML-algorithms, in particular deep neural networks, are quite brittle. With the publications of Szegedy et al. [2] and Goodfellow et al. [3] as a starting point, adversarial examples (or adversarial perturbations) have been recognized as alarming weak points. These adversarial perturbations are created by an adversary with the help of an adversarial attack and often imply erroneous behaviour of the considered ML-module when added to an ordinary input point. Furthermore, these adversarial perturbations are often hard to detect or even imperceptible to the human eye. The imperceptibility is not only challenging for the desired deployment in safety- and security-critical industries, but also hints at a crucial difference between the sensory information processing in humans and in artificial neural networks [4]. Since the discovery of this vulnerability, a lot of different adversarial attacks and defenses (defense strategies) have been published (see: [5], [6]). It has become an arms race between attackers and defenders. Newly published defenses against a set of existing attacks are often rendered useless within a few weeks due to the creation of stronger attacks [7]. The currently most successful and most used defense strategy is adversarial training (e.g. [8]). Here, the developer adds adversarial examples to the training data sets and in this way protects the ML-module against these examples. However, the success of this defense is strongly linked to the adversarial attacks which created the considered adversarial examples. In other words, one can not expect robustness of the ML-algorithm against adversarial attacks that were not considered during the training process. This already shows one big limitation of this approach, since the developer will never be able to consider every possible adversarial attack, i.e. he will never be able to add every possible kind of adversarial examples to the training data sets.

As a consequence, researchers are interested in finding defenses that provably increase adversarial robustness and are not linked to specific adversarial attacks. Ideally, these defenses are able to provide certificates, i.e. are able to verify that for certain predefined input areas no attack can be successful. It is certainly not easy to reach this kind of guarantees due to the fact that it is infeasible to enumerate through all potential input data points of a ML-module. For example, in the context of autonomous driving we simply can not consider every possible traffic condition. However, there are still researchers working heavily on defenses that can provide comparable guarantees. Recent publications in this field of robustness verification apply ideas from convex optimization, duality, satisfiability modulo theory (SMT) and abstract interpretation (e.g. [9], [10], [11]). Here, the authors often try to integrate the robustness goal into the loss function used during training of the ML-module and in this way indirectly motivate adversarial robustness.

In this seminar work, we will take a closer look at one robustness verification publication by K. Dvijotham et al. [12] which makes use of ideas from convex optimization and duality. It should be noted that the considered paper can be seen as an advancement of [13] and [14]. In order to be able to discuss this paper, we will first of all recap basic optimization concepts, in particular duality. Afterwards, we can then summarize the central findings of K. Dvijotham et al., including the new predictor-verifier training framework.

2 Duality

The following discussion is influenced by the duality chapter of the book "Nichtlineare Optimierung" by M. Ulbrich and S. Ulbrich [15]. Let us consider the constrained optimization problem (primal problem):

$$\underset{x\in\mathcal{U}}{min}\, F(x) \; s.t. \; h(x) = 0, \tag{1}$$

where $F : \mathcal{U} \subseteq \mathbb{R}^n \to \mathbb{R}$, $h : \mathcal{U} \subseteq \mathbb{R}^n \to \mathbb{R}^m$ with $\mathcal{U}$ closed subset of $\mathbb{R}^n$. In our cases of interest, these functions F and h will have rather nice mathematical properties. In particular, they will be (locally) Lipschitz continuous and hence, almost everywhere differentiable (Rademacher's Theorem). Solving optimization problems with equality constraints (like (1)) is a difficult task. Often we are not even able to fully determine the set of admissible solutions $\mathcal{A} := \{x \in \mathcal{U} \mid h(x) = 0\}$. Thus, it is desirable to reformulate the original primal problem into an optimization problem that is more tractable. To achieve this, one first of all defines the Lagrange function

$$L(x, \lambda) := F(x) + \lambda^T h(x)$$

with $x \in \mathcal{U}$, $\lambda \in \mathbb{R}^m$. Note that

$$p(x) := \underset{\lambda \in \mathbb{R}^m}{sup}\, L(x, \lambda) = \begin{cases} F(x) & \text{for } x \in \mathcal{A} \\ +\infty & \text{for } x \notin \mathcal{A}. \end{cases}$$

Hence, the original problem (1) is equivalent to

$$\underset{x \in \mathcal{U}}{min}\, p(x). \tag{2}$$

Now, we want to go even further and switch the positions of $\underset{x \in \mathcal{U}}{min}(\cdot)$ (or rather $\underset{x \in \mathcal{U}}{inf}(\cdot)$) and $\underset{\lambda \in \mathbb{R}^m}{sup}(\cdot)$. This finally gives us the dual problem:

$$\underset{\lambda \in \mathbb{R}^m}{sup}\ \underset{x \in \mathcal{U}}{inf}\ L(x, \lambda) \tag{3}$$

It can not be expected that the primal and the dual problem, i.e. (2) and (3), always have the exact same solution. However, one can prove that the dual formulation is very helpful for the development of lower bounds for the primal problem.

Weak Duality Theorem: For $\bar{x} \in \mathcal{A}$ admissible point and $\bar{\lambda} \in \mathbb{R}^m$ the following inequality holds:

$$p(\bar{x}) = \underset{\lambda \in \mathbb{R}^m}{sup}\, L(\bar{x}, \lambda) = F(\bar{x}) \geq d(\bar{\lambda}) := \underset{x \in \mathcal{U}}{inf}\, L(x, \bar{\lambda})$$

Proof: Due to $\bar{x} \in \mathcal{A}$, we have $h(\bar{x}) = 0$ and therefore

$$d(\bar{\lambda}) = \underset{x \in \mathcal{U}}{inf}\, L(x, \bar{\lambda}) \leq L(\bar{x}, \bar{\lambda}) = F(\bar{x})$$

q.e.d.

In other words, the weak duality theorem tells us that (3) can be viewed as a lower bound of (1). Soon we will see that this fact is very helpful in the context of adversarial robustness verification. However, it should be noted that it is not always helpful to optimize the dual instead of the primal problem. For example, solving $d(\bar{\lambda}) := \underset{x \in \mathcal{U}}{inf}\, L(x, \bar{\lambda})$ might again be infeasible. Nevertheless, we now have the necessary tools and definitions for the discussion of [12].

3 Duality-based Verification

In the following, the key results of [12] will be presented. The notation and mathematical formulations will differ slightly from [12]. These changes are primarily motivated by the goal of keeping the notation similar to the book [15] and hence, aligned to the notation of the last section.

As already mentioned in the introduction, we are concerned with verifying that a given ML-module, in our case a simple classification deep neural network, satisfies a predefined adversarial robustness specification. Additionally, if the module does not fulfill the specification, we will try to find ways to improve its adversarial robustness to the degree that the specification is satisfied.

In general, we concentrate on the specification that the considered deep neural network $F(\cdot)$ should not change its classification outcome in the ϵ-neighbourhood / ϵ-ball (with respect to the L^∞-norm) around a given input point x^{nom}. Thus, an adversary can not find successful adversarial examples within this predefined neighbourhood.
To put it in more precise mathematical terms: We have a simple classification deep neural net $F : \mathbb{R}^{n_0} \to \mathbb{R}^{n_K}$, where n_0 is the dimension of an input point and n_K is the number of classes which can be detected by the network. The network outputs logits for every class, i.e. $y^{nom} := \underset{i=1,...,n_K}{argmax} \ F_i(x^{nom})$ is the final classification decision of the network for the input data point $x^{nom} \in \mathbb{R}^{n_0}$. Let us also define the closed L^∞-ball $B_\epsilon(x^{nom}) := \{x_0 \in \mathbb{R}^{n_0} \mid \|x_0 - x^{nom}\|_\infty \leq \epsilon\}$ for $\epsilon > 0$ and let $e_i \in \mathbb{R}^{n_K}$ denote the i-th standard basis vector. Then one can write the above specification as follows: For all $i \in \{1, ..., n_K\}$ we want to have that

$$(e_{y^{nom}} - e_i)^T F(x_0) \geq 0, \ \ \forall x_0 \in B_\epsilon(x^{nom}).$$

As we have already discussed during the first introductory seminar sessions, a deep neural network has a layered architecture and can be viewed as a composition of various similar functions. Now, we want to insert this fact into our formulation of the specification. Let us assume that our neural network $F(\cdot)$ has K layers, denoted by $0, ..., K-1$, with transfer functions $h_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_{k+1}}$ for $k \in \{0, ..., K-1\}$ that map the input of a layer to its output. As a consequence,

$$F(x) = h_{K-1}(h_{K-2}(...(h_0(x_0)))$$

for $x_0 \in \mathbb{R}^{n_0}$. Furthermore, let us define the activation at the k-th layer $x_{k+1} := h_k(x_k)$ for $k \in \{0, ..., K-1\}$ and the collection of activations $x := (x_0, ..., x_K) \in \mathbb{R}^{n_0} \times ... \times \mathbb{R}^{n_K}$. Obviously, this notation implies that $F(x_0) = x_K$.
With this in mind, one can rewrite the specification in the following way:

$$(e_{y^{nom}} - e_i)^T x_K \geq 0, \ \ \forall x_0 \in B_\epsilon(x^{nom}), \forall i \in \{1, ..., n_K\}$$

The specification is violated if an adversary can find a data point $x_0 \in B_\epsilon(x^{nom})$ such that $(e_{y^{nom}} - e_i)^T x_K < 0$ for at least one $i \in \{1, ..., n_K\}$, i.e. the classification network assigns the wrong class to the adversarial example x_0.

Hence, finding an adversarial example for this specification can be formulated as an optimization problem, namely for every $i \in \{1, ..., n_K\}$

$$\begin{aligned} \underset{x:=(x_0,...,x_K)}{min} \ & (e_{y^{nom}} - e_i)^T x_K, \\ s.t. \ & x_0 \in B_\epsilon(x^{nom}) \\ & x_{k+1} - h_k(x_k) = 0, \ k = 0, ..., K-1 \end{aligned} \tag{4}$$

This gives us an optimization problem with equality constraints for every i (similar to the one shown in (1)). From our discussion above, we also know that if every solution of the minimization problem (4) is greater than 0 (for every i), the specification is true, i.e. the neural network is robust with respect to the defined ϵ-ball.

At this point we want to underline that the ideas presented here can easily be adapted to other contexts, e.g. other ML-algorithms and other input norms. The presented techniques are not limited to simple classification neural networks and L^∞-neighbourhoods. The specification with a deep neural network and a L^∞-input norm (imperceptibility metric) is simply a prominent adversarial robustness setting, which has been studied extensively during the last few years.

As a next step, let us consider the Lagrange function of the optimization problem (4), namely

$$L(x, \lambda) := (e_{y^{nom}} - e_i)^T x_K + \sum_{k=0}^{K-1} \lambda_k^T (x_{k+1} - h_k(x_k))$$

with $x := (x_0, ..., x_K) \in \mathbb{R}^{n_0} \times ... \times \mathbb{R}^{n_K}, \lambda := (\lambda_0, ..., \lambda_{K-1}) \in \mathbb{R}^{n_1} \times ... \times \mathbb{R}^{n_K}$. Due to the Weak Duality Theorem (see last chapter), one can be sure that for every $\bar{\lambda}$ and every $\bar{x} \in \mathcal{A} := \{(x_0, ..., x_K) \mid x_0 \in B_\epsilon(x^{nom}), x_{k+1} = h_k(x_k), \ k = 0, ..., K-1\}$ we have

$$(e_{y^{nom}} - e_i)^T \bar{x}_K \geq d(\bar{\lambda}, x^{nom}) := \underset{x=(x_0,...,x_K), x_0 \in B_\epsilon(x^{nom})}{inf} L(x, \bar{\lambda}).$$

This directly implies that if there exists a $\bar{\lambda} \in \mathbb{R}^{n_1} \times ... \times \mathbb{R}^{n_K}$ with $d(\bar{\lambda}, x^{nom}) > 0$ then also $(e_{y^{nom}} - e_i)^T \bar{x}_K > 0$ for all $\bar{x} \in \mathcal{A}$. In this case, the considered ML-module never assigns any input from the ball $B_\epsilon(x^{nom})$ to the class i, i.e. is robust in this sense. This basically gives us a strategy on how to analyze robustness and on how to determine robustness certificates. As explained in the introduction, certificates are crucial when it comes to the deployment of ML-based software in safety and security critical industries. But, we also want to use the above insights for the formulation of a duality-based defense strategy.

The transformation into a defense strategy is now quite straightforward. We are interested in training a ML-module F with parameters θ in a way that $\underset{\bar{\lambda}}{max} \ d(\bar{\lambda}, x^{nom}) > 0$ and x^{nom} arbitrary point from the training / test data sets. Thus, it makes sense to add a suitable penalty term to our general loss function $l(\cdot, \cdot)$ (e.g. entropy or MSE).

For example, the training goal could then be described in the following way

$$\underset{\theta}{min}\ \mathbb{E}_{(x,y)\sim\mathbb{P}}[(1-\kappa)\ l(F(x),y) - \kappa\ \underset{\bar{\lambda}}{max}\ d(\bar{\lambda},x)] \tag{5}$$

with $\kappa \in (0,1)$ a weighting constant and $\mathbb{P}$ the given data distribution.

Unfortunately, this defense strategy still offers a central problem, namely the calculation of $d(\cdot,\cdot)$ and the calculation of $\underset{\bar{\lambda}}{max}\ d(\bar{\lambda},\cdot)$. Evaluating these functions gives again rise to complex optimization problems. This is basically the reason why duality-based verification techniques are still having problems when it comes to scaling to neural networks with a high number of layers and neurons. For example, in [14] the authors try to solve a comparable optimization problem in every single training step of a ML-module and hence, it becomes very computationally expensive. To deal with this problem, [12] offers an interesting alternative, called predictor-verifier training (PVT).

4 Predictor-verifier Training

The authors of [12] argue that for most of the current neural networks the calculation of $d(\cdot)$ can be done in closed form, i.e. can be done with a concrete formula and without further optimization methods. This is due to the rather simple activation functions within state-of-the-art deep neural nets (e.g. ReLU functions). As a consequence, the authors claim that we do not have to worry about this part of the defense strategy. However, the calculation of $\underset{\bar{\lambda}}{max}\ d(\bar{\lambda},x^{nom})$ represents a bigger challenge. The central, helpful presumption of the authors is that the dual variable $\lambda^{\star}(x^{nom}) := \underset{\bar{\lambda}}{argmax}\ d(\bar{\lambda},x^{nom})$ shares a lot of structure across different training samples x^{nom}. To put it in more mathematical terms, one assumes that the dual variable $\lambda^{\star}(\cdot)$ depends rather continuously on the sample data point x^{nom}. It should be noted that [12] does not offer a solid mathematical explanation for this statement.

With this idea in mind, it makes sense to think about "learning" the necessary dual variable $\lambda^{\star}(\cdot)$. Thus, we train a second deep neural network V, called the verifier network, with parameters τ such that $V(x^{nom}) \approx \lambda^{\star}(x^{nom})$ for all input data points x^{nom}. In other words, during training of V we are interested in finding parameters τ such that

$$\underset{\tau}{max}\ \mathbb{E}_{(x,y)\sim\mathbb{P}}[d(V(x),x)]$$

or equivalently

$$\underset{\tau}{min}\ \mathbb{E}_{(x,y)\sim\mathbb{P}}[-d(V(x),x)].$$

The verifier net V is trained alongside the classification neural net F, also called predictor network. Every training step of F will also be a training step of V. Fortunately, our former training objective (5) can now be easily adapted to this double training setting.

We simply have to minimize concerning both parameter sets θ and τ and insert the verifier net into the objective:

$$\underset{\theta,\tau}{min}\, \mathbb{E}_{(x,y)\sim\mathbb{P}}[(1-\kappa)\; l(F(x),y) - \kappa\; d(V(x),x)]$$

The reader should be aware that F and d depend on θ whereas V depends on τ. Training the predictor and verifier at the same time with the above training objective is then called predictor-verifier training (PVT).

This special type of training with an included verifier is a very compelling idea, since it erases the burden of solving an additional optimization problem (calculation of $\lambda^{\star}(\cdot)$) in every training step. Furthermore, after the training process we can use the verifier V for the calculation of certificates. To issue a certificate, we only need to check whether $d(V(x^{nom}), x^{nom})$ is greater than 0.

5 Conclusion

In this seminar work, we gave a quick introduction to the concept of duality and then explained the key ideas presented in [12]. We showed how dual optimization problems can be used for the issue of adversarial robustness certificates as well as for the training of robust ML-modules. At various steps we changed the notation and the mathematical formulations of [12] in order to make it more accessible for the seminar participants. In general, we are very excited about the current developments in the field of duality-based verification. We are convinced that this approach will play a major role in the adversarial robustness verification of deep neural networks. As a next step, researchers should try to further simplify the considered optimization problems and in this way improve the scalability of these methods. Furthermore, it would be interesting to see a more in depth mathematical analysis of PVT.

References

[1] K. Pei et al., ”Towards Practical Verification of Machine Learning: The Case of Computer Vision Systems”, 2017.

[2] C. Szegedy et al., ”Intriguing properties of neural networks”, 2014.

[3] I. Goodfellow et al., ”Explaining and Harnessing Adversarial Examples”, 2015.

[4] W. Brendel, J. Rauber and M. Bethge, ”Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models”, 2018.

[5] N. Akhtar and A. Mian, ”Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey”, 2018.

[6] X. Yuan et al., ”Adversarial Examples: Attacks and Defenses for Deep Learning”, 2018.

[7] A. Raghunathan, J. Steinhardt and P. Liang, ”Certified Defenses against Adversarial Examples”, 2018.

[8] A. Madry et al., ”Towards Deep Learning Models Resistant to Adversarial Attacks”, 2018.

[9] T. Gehr et al., ”AI^2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation, 2018.

[10] G. Katz et al., ”Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks, 2017.

[11] A. Lomusico and L. Maganti, ”An approach to reachability analysis for feed-forward ReLU neural networks, 2017.

[12] K. Dvijotham et al., ”Training Verified Learners with Learned Verifiers, 2018.

[13] K. Dvijotham et al., ”A Dual Approach to Scalable Verification of Deep Networks”, 2018.

[14] E. Wong and J. Kolter, ”Provable defenses against adversarial examples via the convex outer adversarial polytope”, 2017.

[15] M. Ulbrich and S. Ulbrich, ”Nichtlineare Optimierung”, Book (Birkhäuser Verlag), 2012.