Notes on Lipschitz Margin, Lipschitz Margin Training, and Lipschitz Margin p-Values for Deep Neural Network Classifiers

George Kesidis and David J. Miller

***Abstract*—We provide a local class purity theorem for Lipschitz continuous, half-rectified DNN classifiers. In addition, we discuss how to train to achieve classification margin about training samples. Finally, we describe how to compute margin p-values for test samples.**

I. Introduction

A variety of papers have been recently produced on "robustifying" Deep Neural Networks (DNNs), particularly to adversarial Test-Time Evasion (TTE) attacks [14], [15], [13]. We discuss some of this work in Sections III.A and IV.A of [9] and argue for the need for TTE-attack detection [8] for robustness.

In this note, we derive a **local class purity** result under the assumption of Lipschitz continuity, discuss Lipschitz margin training, and define an associated p-value. Estimation of the Lipschitz parameter for a given DNN is discussed in, *e.g.,* [12], [14], [16], [4].

II. Margin in DNN classifiers

Consider the DNN $f : \mathbb{R}^n \to (\mathbb{R}^+)^C$ where C is the number of classes. Further suppose that for a test-time, input pattern $x \in \mathbb{R}^n$ to the DNN, the class decision is

$$\hat{c}(x) = \arg\max_i f_i(x),$$

where f_i is the ith component of the C-vector f. That is, we have defined a class-discriminant output layer of the DNN. Here assume that a class for x is chosen arbitrarily among those that tie for the maximum.

Define the **margin** of x as

$$\mu_f(x) \ := \ f_{\hat{c}(x)}(x) - \max_{i \neq \hat{c}(x)} f_i(x) \quad \geq \quad 0. \tag{1}$$

The normalized Lipschitz margin

$$\frac{\mu_f(x)}{f_{\hat{c}(x)}(x)} \tag{2}$$

can roughly be interpreted as a kind of confidence in classifying x to class $\hat{c}(x)$, *cf.,* Section 4.

The authors are with the School of EECS, Pennsylvania State University, University Park, PA, 16803, USA. This research is supported by AFOSR DDDAS grant and Cisco URP gift. Email: {gik2,djm25}@psu.edu

Now suppose the ℓ_∞ (*i.e.,* max-norm) Lipschitz continuity parameter for f is estimated as $L_\infty > 0$ satisfying:[1]

$$\forall x, y, \quad |f(x) - f(y)|_\infty \quad \leq \quad L_\infty |x - y|_\infty.$$

Now consider samples in a open ℓ_∞ hypercube centered at x, *i.e.,*

$$y \in \mathrm{B}_\infty(x, \varepsilon) := \{z \in \mathbb{R}^n \ : \ |x - z|_\infty < \varepsilon\}$$

for $\varepsilon > 0$.

The following "locally robust classification" result depends on the sample-dependent margin. This result is similar to that of [14].

Theorem 2.1: If f is ℓ_∞ Lipschitz continuous with parameter $L_\infty > 0$ and $\mu_f(x) > 0$ then

$$B\left(x, \frac{\mu_f(x)}{2L_\infty}\right)$$

is class pure.

Proof: For any $y \in B(x, \frac{1}{2}\mu_f(x)/L_\infty)$ we get by the assumed Lipschitz continuity that

$$\begin{array}{rcl} \frac{1}{2}\mu_f(x) & > & |f(x) - f(y)|_\infty \\ & := & \max_i |f_i(x) - f_i(y)| \\ & \geq & \max_i |f_i(x)| - |f_i(y)| \quad \text{(triangle inequality)} \\ & = & \max_i f_i(x) - f_i(y) \quad \text{(since } f_i \geq 0) \\ & \geq & f_{\hat{c}(x)}(x) - f_{\hat{c}(x)}(y) \end{array}$$

So,

$$f_{\hat{c}(x)}(y) \quad > \quad f_{\hat{c}(x)}(x) - \frac{1}{2}\mu_f(x). \tag{3}$$

If we instead write $|f_i(y)| - |f_i(x)|$ in the triangle inequality above and then replace $\hat{c}(x)$ by any $i \neq \hat{c}(x)$, we get that

$$\forall i \neq \hat{c}(x), \quad f_i(y) \quad < \quad f_i(x) + \frac{1}{2}\mu_f(x). \tag{4}$$

[1] Note that if f is estimated to have an ℓ_2 parameter L_2, then $L_\infty \leq nL_2$ since, $\forall z, |z|_\infty \leq |z|_2 \leq n|z|_\infty$.

So, by (3) and (4),

$$\begin{aligned}\forall i \neq \hat{c}(x), \quad f_i(y) &< f_i(x) + \frac{1}{2}\mu_f(x) \\ &\leq f_{\hat{c}(x)}(x) - \frac{1}{2}\mu_f(x) \quad \text{(by (1))} \\ &< f_{\hat{c}(x)}(y)\end{aligned}$$

□

III. Lipschitz-margin training

Robust training is surveyed in [15], [9]. We focus herein on attempting to achieve a prescribed Lipschitz margin. Recall that, by Cover's theorem [2], class separation is achieved if the DNN's penultimate layer is sufficiently large.

Let θ represent the DNN parameters. Let $\mathcal{T}$ represent the training dataset and let $c(x)$ for any $x \in \mathcal{T}$ be the *ground truth* class of x.

To try to achieve a common Lipschitz margin of μ for all training samples, [14] suggests to add the margin "to all elements in logits except for the index corresponding to" $c(x)$. For example, train the DNN by finding:

$$\begin{aligned}&\min_\theta - \sum_{x\in\mathcal{T}} \log\left(\frac{f_{c(x)}(x)}{\sum_{i\neq c(x)}(f_i(x)+\mu)}\right) \\ &= \min_\theta - \sum_{x\in\mathcal{T}} \log\left(\frac{f_{c(x)}(x)}{(C-1)\mu + \sum_{i\neq c(x)} f_i(x)}\right) \quad (5)\end{aligned}$$

For a softmax example, one could train the DNN using the modified cross-entropy loss[2]:

$$\min_\theta - \sum_{x\in\mathcal{T}} \log\left(\frac{\mathrm{e}^{f_{c(x)}(x)}}{\mathrm{e}^{f_{c(x)}(x)} + \sum_{i\neq c(x)} \mathrm{e}^{f_i(x)+\mu}}\right) \quad (6)$$

These DNN objectives do not guarantee the margins for training samples will be met.

Alternatively, for each training sample x, one could augment the training set with plural samples y such that $|x-y|_\infty = \mu$ and simply train using an unmodified logit or cross-entropy loss objective.

Alternatively, one could first train an "original" DNN with an unmodified objective and unaugmented training dataset. Then the original DNN is used to produce *adversarial examples* by some strategy, *e.g.,* [10], [5], [1], [8], each of bounded perturbation ($\sim \mu$) starting from training samples. The training dataset is then augmented by these adversarial examples and the DNN retrained (say starting from the parameters of the original DNN). See *e.g.,* [13], [17] (and Sections III.A, IV.A of [9]).

Alternatively, one can achieve Lipschitz-margin DNN training by (dual) optimization of the weighted margin constraints, *e.g.*,

$$\min_\theta \sum_{x\in\mathcal{T}} \lambda_x \left(\frac{\max_{i\neq c(x)} f_i(x) + \mu - f_{c(x)}(x)}{(C-1)\mu + \sum_j f_j(x)}\right), \quad (7)$$

[2]Note that we do not need to exponentiate as we herein assume that, $\forall x, i$, $f_i(x) \geq 0$, *i.e.,* the DNN outputs are "half rectified" [12].

where the DNN mappings f_i obviously depend on the DNN parameters θ, and the weights $\lambda_x \geq 0 \ \forall x \in \mathcal{T}$. For hyperparameter $\delta > 1$, training can proceed simply as:

0 Select initially equal $\lambda_x > 0$, say $\lambda_x = 1 \ \forall x \in \mathcal{T}$.
1 Optimize over θ (train the DNN).
2 If all margin constraints are satisfied then stop.
3 For all $x \in \mathcal{T}$: if margin constraint x is not satisfied then $\lambda_x \to \delta\lambda_x$.
4 Go to step 1.

Again, the parameters of the previous DNN could initialize the training of the next, and an initial DNN can be trained instead by using a logit or cross-entropy loss objective, as above. There are many other variations including also decreasing λ_x when the x-constraint is satisfied, or additively (rather than exponentially) increasing λ_x when they are not, and changing λ_x in a way that depends on the degree of the corresponding margin violation. Clearly this approach may require frequent retraining of the DNN.

Finally, let $-\sum_{x\in\mathcal{T}} L(\theta, x, c(x))$ be a cross-entropy loss. For example, [15] discloses the training problem,

$$\min_\theta \max_{z\in B(x,\mu),\ x\in\mathcal{T}} - \sum_{x\in\mathcal{T}} L(\theta, z, c(x)),$$

but notes that the inner maximization is NP hard [6].

IV. Low-margin atypicality of test samples

Given an arbitrary DNN $f : \mathbb{R}^n \to (\mathbb{R}^+)^C$, let $\mathcal{T}_\kappa$ be the training samples of class $\kappa \in \{1, 2, ..., C\}$, *i.e.,* $\forall x \in \mathcal{T}_\kappa$, $\hat{c}(x) = c(x) = \kappa$. Recall (1) and suppose a Gaussian Mixture Model (GMM) is learned using the log-margins of the training dataset

$$\{\log \mu_f(x) \ : \ x \in \mathcal{T}_\kappa\}$$

by EM [3] using BIC model order control [11] as, *e.g.,* [7]. Let the resulting GMM parameters be $\{w_i, m_i, \sigma_i\}_{k=1}^{I_\kappa}$, where $I_\kappa \leq |\mathcal{T}_\kappa|$ is the number of components, the $w_i \geq 0$ are their weights ($\sum_{i=1}^{I_\kappa} w_i = 1$), the m_i are their means, and the $\sigma_i > 0$ are their standard deviations. So, we can simply compute the **margin p-value** of any **test sample** x,

$$\pi_f(x) \quad = \quad \sum_{i=1}^{I_\kappa} w_i F\left(\frac{\log(\mu_f(x)) - m_i}{\sigma_i}\right)$$

where F is the standard normal c.d.f. That is, $\pi_f(x)$ is the probability that a randomly chosen sample from the same distribution as that of the training samples has smaller margin than the test sample x. So, one can use $\pi_f(x)$ to check whether a test sample x has abnormally small classification margin.

Note that the margin p-value should not be based any "large perturbation" test-time evasion samples *e.g.,* [10], [5], [1], [8] that may be used to augment the training dataset for purposes of robustness.

References

[1] N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proc. IEEE Symposium on Security and Privacy*, 2017.
[2] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14:326334, 1965.

[3] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society.*, 39(1):1–38, 1977.

[4] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G.J. Pappas. Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. https://arxiv.org/pdf/1906.04893.pdf, 2019.

[5] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR*, 2015.

[6] G. Katz, C. Barrett, D.L. Dill, K. Julian, and M.J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proc. Int'l Conf. on Computer Aided Verification (CAV)*. Springer, 2017.

[7] D.J. Miller, Z. Qiu, and G. Kesidis. Parsimonious Cluster-based Anomaly Detection (PCAD). In *Proc. IEEE MLSP*, Aalborg, Denmark, Sept. 2018.

[8] D.J. Miller, Y. Wang, and G. Kesidis. Anomaly Detection of Attacks (ADA) on DNN Classifiers at Test Time. *Neural Computation*, 2019; shorter version in *Proc. IEEE MLSP*, Aalborg, Denmark, Sept. 2018; https://arxiv.org/abs/1712.06646.

[9] D.J. Miller, Z. Xiang, and G. Kesidis. Adversarial Learning in Statistical Classification: A Comprehensive Review of Defenses Against Attacks. *submitted*, 2019, http://arxiv.org/abs/1904.06292.

[10] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Proc. 1st IEEE European Symp. on Security and Privacy*, 2016.

[11] Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[12] C. Szegedy, W. Zaremba, I Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, 2014.

[13] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. In *Proc. ICLR*, 2019.

[14] Y. Tsuzuku, I. Sato, and M. Sugiyama. Lipschitz-margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks. In *Proc NIPS*, 2018.

[15] S. Wang, Y. Chen, A. Abdou, and S. Jana. MixTrain: Scalable Training of Verifiably Robust Neural Networks. https://arxiv.org/abs/1811.02625, Nov. 2018.

[16] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I.S. Dhillon, and L. Daniel. Towards Fast Computation of Certified Robustness for ReLU Networks. https://arxiv.org/abs/1804.09699, Apr. 2018.

[17] H. Zhang, H. Chen, Z. Song, D. Boning, I. Dhillon, and C.-J. Hsieh. The limitations of adversarial training and the blind-spot attack. https://arxiv.org/abs/1901.04684, 2019.