Smooth Decision Boundaries with Lipschitz Multi-Layer Perceptrons

Xavier Sécheresse*
Théo Communal*
xavier.secheresse@etu.minesparis.psl.eu
theo.communal@etu.minesparis.psl.eu
Mines Paris, PSL University
Master MVA, ENS Paris Saclay
France

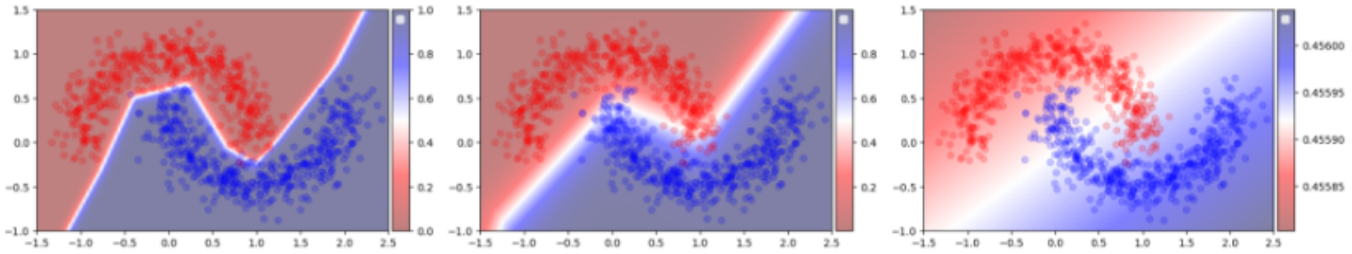

Figure 1: Boundaries of a Lipschitz MLP classifier on the Two-Moons dataset for different values of the penalization term

ABSTRACT

[1] proposed a new architecture of Lipschitz Multi-Layer Perceptrons for shape interpolation and adversarial robustness. This paper aims to demonstrate the efficiency of this type of network for classification. We show that the use of such an architecture yields smooth results on simple binary classification tasks but it is not as efficient as spectral regularization.

KEYWORDS

Lipschitz, Smoothness, Classification, Geometry

ACM Reference Format:
Xavier Sécheresse and Théo Communal. 2023. Smooth Decision Boundaries with Lipschitz Multi-Layer Perceptrons. In . ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

This paper was written for the 2023 session of the "Geometric Data Analysis", lectured by Dr. Jean FEYDY for the MVA Master of ENS Paris-Saclay. When refering to "The Authors", we refer to [1] as the goal of this project is to carry out a critical review of this article.

1 GENERALITIES ON LISPCHITZ NETWORKS

1.1 Definitions and Properties

In this part, we introduce some of the fundamental properties of Lipschitz functions.

*Both authors contributed equally to this research.

, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Definition : A function f_θ with parameter θ is c-Lipschitz with regards to its inputs x, if there exist a constant $c > 0$ such that :

$$\forall x, y, ||f_\theta(x) - f_\theta(y)|| \leq c \cdot ||x - y||$$

where $||\cdot||$ is a norm of choice.

Composition of Lipschitz Functions : If f and g are Lipschitz with constants K_f and K_g, $f \circ g$ is Lipschitz with constant $K_f K_g$.

Since commonly used activation functions are 1-Lipschitz, the task of ensuring a neural network is Lipschitz reduces to constraining the learnable layers to be Lipschitz. Hence, the Lipschitz bound c of a fully-connected network can be estimated with :

$$c = \prod_{i=1}^{L} ||Wi||$$

Equivalence of Matrix Norms : In the definitions above, the norms can be any matrix norms as norms on finite dimensional vectorial spaces are all equivalent. Two norms $\|\cdot\|_p$ and $\|\cdot\|_q$ on a vector space V are said to be equivalent if there exist positive constants c_1 and c_2 such that for all vectors x in V:

$$c_1\|x\|_p \leq \|x\|_q \leq c_2\|x\|_p$$

Thus, optimizing the Lipschitz bound under a particular a given norm will optimize the bound for other norms.

The usual matrix p-norms are listed below:

$$||M||_2 = \sigma_{max}(M)$$

$$||M||_1 = \max_j \sum_i |M_{ij}|$$

$$||M||_\infty = \max_i \sum_j |M_{ij}|$$

1.2 Benefits of smoothness

In their paper, the authors examined the effectiveness of their approach on three tasks where smoothness is a strong asset :

- **Robustness to adversarial attacks :** Adversarial attacks are small, structured changes made to a network's input signal that cause a significant change in output. Robustness for classifiers can be defined by ensuring that inputs in the same ϵ-ball result in the same function output:

  $$\forall x, x', if ||x - x'|| \leq \epsilon \longrightarrow argmax f(x) = argmax f(x')$$

  This definition is closely related to Lipschitz smoothness. However, initial approaches to combat adversarial attacks were more focused on data augmentation methods and smoothness constraints have come into focus only more recently .
- **Shape Interpolation :** Shape interpolation is a technique for generating shapes "between" training shapes. Shapes are encoded with an autoencoder to a latent space and new shapes can be generated by using the decoder. Shape interpolation needs abundant training data to guarantee that the latent space is well structured. A smooth autoencoder like the one proposed by the authors can generate plausible shapes using as few as two training shapes.
- **Test-time optimization :** Reconstruction of complete point clouds from partial data is a very active research topic. However, standard MLP often provides unsatisfying results when applied to this task and usually yields unstable results. The authors showed that the use of a lipschitz MLP can greatly improve the precision of the results on common data sets.

2 TRADITIONAL METHODS TO ENCOURAGE SMOOTHNESS

Traditional methods for promoting smoothness based on Network Regularization have mixed results.

- **Soft constraints through gradient penalties** Regularization techniques that depend on the input of the network lead to smooth behavior around the training samples, but have no guarantee of a smooth behavior beyond them. On the contrary, it may even promote non-smooth behavior by squeezing function changes to locations without training samples. *Example : Dirichlet Energy-regularized loss function* $\mathcal{J}(\theta) = \mathcal{L}(\theta) + \alpha \sum_j ||\frac{\partial f_\theta}{\partial x}(x)||^2$
- **Indirect smoothness constraints :** Other training techniques like early-stopping, dropout, weight decay and learning rate decay that were initially used to limit overfitting produce smooth results. However, there are no theoretical results on how they encourage smoothness.
- **Soft constraints through spectral regularization :** Methods based on penalizing the L2 norm of the weights matrix of the network which is the largest singular value to encourage Lipschitz smoothness.
  Example : $\mathcal{J}(\theta) = \mathcal{L}(\theta) + \lambda \sum_i ||W_i||_p^2$ where $||\cdot||_p$ is a matrix p-norm.

Authors focused on Lipschitz Regularization as its benefits are well beyond promoting smoothness : it has applications in robustness against adversarial attacks and better generalization properties. Several techniques that are both non-heuristic and do not depend on the network's input exist to constrain the Lipschitz constant of the network:

- **Weights Normalization** under different norms : spectral norm, L1, L-∞, L2.
- **Ortonormalizing** the weights matrix to obtain 1-Lipschitz networks.
- **Norm regularization** that penalizes the largest eigenvalue of each weight matrix and thus constrains the Lipschitz bound.

3 SMOOTH MULTI-LAYER PERCEPTRON

3.1 Authors' proposed method

The authors chose the Matrix ∞-norm to measure the Lipschitz constant but this is not a defining approach thanks to norm equivalency. The proposed method for a Lipschitz Smooth MultiLayer Perceptron requires two simple modifications to a standard MLP.

- **Hard Lipschitz constraint through Weight Normalization**. Let c_i be a trainable Lipschitz bound for the i-th layer. It is initialized as defined in Part 1. In each Layer $y = ReLU(W_i x + b_i)$, W_i is replaced by $\hat{W}_i$. Let r be a row of W_i. The row $\hat{r}$ of $\hat{W}_i$ is equal to $r/softmax(c_i)$ if its absolute value row-sum is smaller than $softmax(c_i)$ and equal to r otherwise. Thus, weights are never clipped and this normalization guarantees that the Lipschitz constant is bounded. $softplus(c_i) = log(1 + e^{c_i})$ is used instead of c_i to avoid negative Lipschitz bounds and in most of the cases, as $c_i >> 1$, $softplus(c_i) \sim c_i$
- **Lipschitz Regularisation** The loss is penalized with the learnable Lipschitz bound of the network defined above instead of penalizing directly the weights matrices. This formulation also allows us to transcribe the exponential growth of the Lipschitz constant. The value of α is fixed through parameter sweeping and has the advantage.

  $$\mathcal{J}(\theta, c_1, ..., c_L) = \mathcal{L}(\theta) + \alpha \prod_i softplus(c_i)$$

3.2 Alternatives explored by the Authors

The authors have iterated through multiple formulations before ending up with the method proposed above. In order to compare them, the Authors found the best value of λ in each case independently through parameter sweeping.

Authors first tried using $\mathcal{J}(\theta) = \mathcal{L}(\theta) + \alpha \sum_i ||W_i||_p^2$ where $||\cdot||_p$ is a matrix p-norm. Although this formulation can effectively find a smooth solution, finding a good α is not an easy task as the penalization term fails to capture the exponential growth of the Lipschitz constant with respect to the number of layers of the network (as the Lipschitz constant is bounded by $\prod_{i=1}^{L} ||Wi||_\infty$. This implies that the value of α must be changed when the architecture of the network changes.

As a consequence, they opted instead for a loss $\mathcal{J}(\theta) = \mathcal{L}(\theta) + \alpha \prod_i ||W_i||_p$ that captures the exponential growth of the Lipschitz

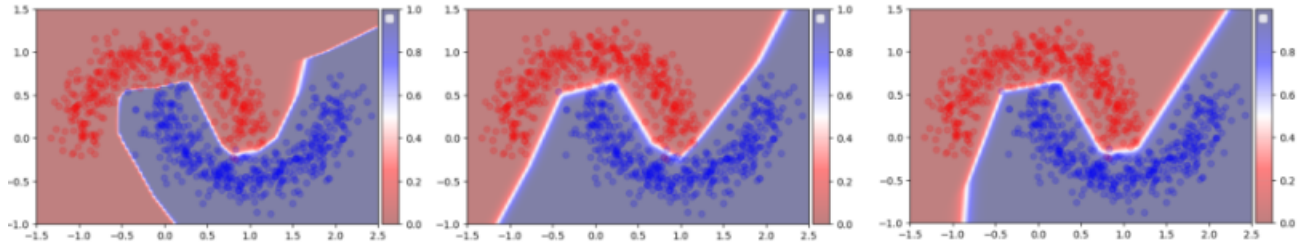

Figure 2: Decision boundaries for the different experiments. (left) normal MLP (center) Authors' Lipschitz MLP (right) Spectral Norm MLP

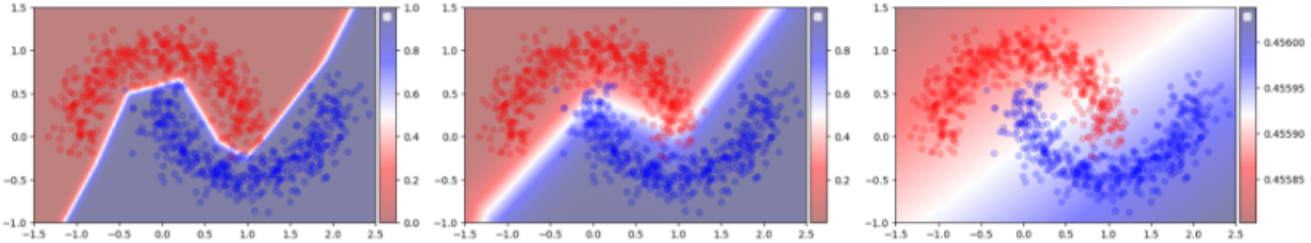

Figure 3: Decision boundaries of the Experiment 2 MLP for different values of the parameter α. (left) good parameter choice ($\alpha = 2 \cdot 10^{-6}$) (center) high regularisation ($\alpha = 1 \cdot 10^{-5}$) (right) very high regularisation ($\alpha = 5 \cdot 10^{-5}$)

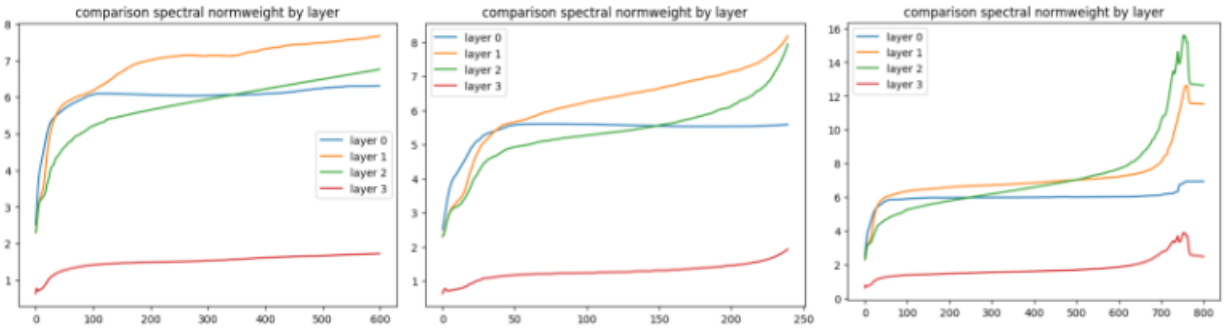

Figure 4: Spectral norm of each layer of the Experiment 2 MLP for different values of the parameter α. (left) good parameter choice ($\alpha = 2 \cdot 10^{-6}$) (center) high regularisation ($\alpha = 1 \cdot 10^{-5}$) (right) very high regularisation ($\alpha = 5 \cdot 10^{-5}$)

constant described beforehand. However, they found out that this method works equally well as their proposed method on narrower networks but converge slower on wider networks leading to a greater number of epochs and training time.

That is how they began considering penalizing on a per-layer learnable Lipschitz bound instead of the matrix weights, first trying the following formulation : $\mathcal{J}(\theta) = \mathcal{L}(\theta) + \alpha \sum_i log(softplus(c_i))$. However, this loss is unbounded below when one of the Lipschitz constant becomes close to zero. In practice, this leads to convergence issues : in their training, one of the Lipschitz constants continued to decrease after one week of training. In the end, the authors settled on the formulation presented above.

4 OUR CONTRIBUTION : APPLICATION TO CLASSIFICATION

Our contribution is twofold :

- Implementing the proposed Lipschitz regularization proposed by the Authors in Pytorch (originally implemented in JAX)
- Testing Lipschitz MLP on classification tasks while the original paper was focused on adversarial robustness, test-time optimzation and shape interpolation. This idea originally comes from [2].

We applied the desired regularization to a standard classification task on the 2 moons dataset. We compared 3 different types of MLP for this task:

- **Experiment 1:** Standard Multi Layer Perceptron (MLP)

- **Experiment 2:** Lipschitz Normalized MLP as proposed by the Authors. The loss is also augmented with $\mathcal{J}(\theta, c_1, ..., c_L) = \mathcal{L}(\theta) + \alpha \prod_i softplus(c_i)$, as proposed by the authors.
- **Experiment 3:** Standard MLP with a spectral norm penalization, in the loss : $\mathcal{J}(\theta) = \mathcal{L}(\theta) + \lambda \prod_i ||W_i||_2$

4.1 Method and dataset

The synthethic data used is the Two Moons dataset with 1000 samples and a noise parameter of 0.15. Different noise parameters (0, 0.1, 0.15 and 0.2) were tested. We settled on 0.15 as it was a good compromise between complexity of the task and and limited number of outliers. We chose to work with such a toy dataset to have good interpretability of the results and a model with limited compute requirements (as we were limited by free Google Colab GPU usage).

Architecture : Due to the simplicity of the input data, the MLP architecture consists in 3 fully-connected layers of 14 neurons, each separated with a ReLU activation function (that is 1-Lipschitz). The Output Layer consists in a single neuron and a Sigmoid activation function (that is 1-Lipschitz).

Loss and metric : We used a Binary Cross-Entropy error loss as basic loss for the 3 models as it usually yields better results for a classification task and also when added to a regularisation parameter. Moreover, we chose the accuracy as metrics for our problem as the 2 classes are balanced. Penalization constants λ and α for experiments 2 and 3 were fixed through parameter sweeping.

Optimizer : For the training, we used Adam optimiser with a constant learning rate of 10^{-3}. This learning rate was chosen through parameter sweeping. Our experimentation showed that a decreasing learning rate did not increase the accuracy at the end of the training for this task.

We plotted the different MLP boundaries on $[-1.5, 2.5] \times [-1, 1.5]$ (which is were all of the data is located). We used a 200x200 grid as a way to ensure reasonable computational times while keeping reasonably precise boundary plots.

4.2 Results and Discussion

Smoothness metric

For each layer, we use $||W_i||_2$ to measure smoothness of the network. We observe on Figure 5 that Experiment 1, that the norm of the weights matrix diverges as training continues because the network overfits the data, thus resulting in non-smooth behavior. On the contrary, Experiment 2 and 3 converges. This shows that the associated networks have expected smooth behavior. Convergence in Experiment 3 is quicker than Experiment 2 as the norm of the weight matrix is directly penalized in the loss. Experiment 2 weights norms converges to a higher value but this is not relevant to assess that the network might be less smooth than Experiment 3. Indeed, Experiment 2 uses the L-∞ norm (as proposed by the authors) whereas our metric is based on the L-2 norm. However, these two norms are equivalent, as shown in Part 1.

4.2.1 Decision boundaries and reliable uncertainty estimates. A characteristic of neural networks trained to minimize classification losses provide unreliable uncertainty estimates especially when the networks are faced with out of distribution data. This topic is covered in more details by [2] They suggest that smooth MLP promotes reliable uncertainty estimate around decision boundaries. This characteristic can be seen on Figure 2 where we show the decision boundaries of the different experiments and see that Experiment 2 and 3 have a larger uncertainty area (white band).

Moreover, Figure 3 illustrates for Experiment 2 that the uncertainty area grows larger as the penalization parameter grows. Figure 4 illustrates the link between smoothness of the network measured with $||W_i||_2$ and uncertainty estimate for Experiment 2 : as the spectral norms grows, the uncertainty band grows.

4.3 Limits of Lipschitz networks

4.3.1 Weak Models. There is a trade-off between imposing constraints on smoothness and model capabilities. While a constant function is very smooth, it lacks practical utility. When hyperparameters are chosen wrongly, a neural network can be "too Lipschitz": decreasing the Lipschitz constant may indeed increase robustness to adversarial examples or facilitate shape interpolation, but once the Lipschitz constant becomes too low, accuracy drops significantly. We show an example of the relationship between α and model capabilities for Experiment 2 on Figure 3 : as α grows larger, the model becomes less expressive and settles on a linear decision boundary as a trade-off between performance and smoothness.

4.3.2 Optimization. Smoothness constraints have a strong effect on optimization. Smoothness regularization techniques of Experiment 2 and 3 affect optimization by changing the loss function. Figure 6 shows that the appropriate number of epochs to reach convergence to optimal accuracy is higher for Experiment 2 than for the unregularized network or spectral constraint network. Slower training of Experiment 2 was already put into evidence by [1] for 2D shape interpolation tasks, but it is also true for binary classification. On the contrary, adding a spectral norm penalization as in Experiment 3 does not increase training time. [2] also shows that changing the learning rate can lead to vastly different smoothness properties. We leave learning rate experiments on the authors' proposed Lipschitz MLP to further work.

REFERENCES

[1] Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. 2022. Learning Smooth Neural Functions via Lipschitz Regularization. (2022).
[2] Mihaela Rosca, Theophane Weber, Arthur Gretton, and Shakir Mohamed. 2021. A case for new neural network smoothness constraints. (2021).

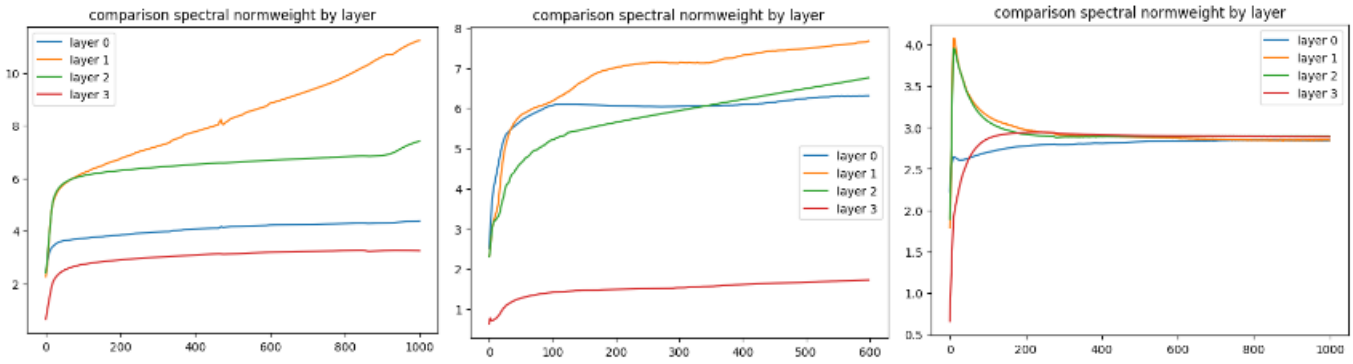

Figure 5: Spectral norm of the weights of each layer for the 3 different experiments. (left) normal MLP (center) lipschitz MLP (right) spectral norm MLP

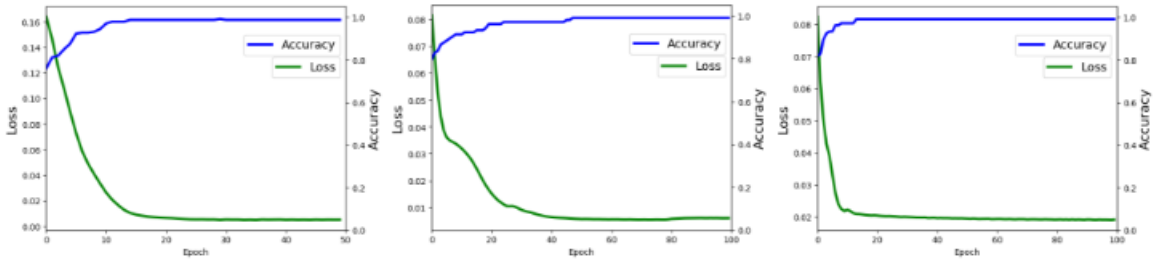

Figure 6: Training loss and corresponding accuracy for the 3 different experiments. (left) normal MLP (center) lipschitz MLP ($\alpha = 2 * 10^{-6}$) (right) spectral norm MLP ($\lambda = 3 * 10^{-2}$)