Fisher Lipschitz Networks

Contents

1 Brainstorm

I believe that there is no partial isometry between $\mathbb{R}^n$ and Δ^m. This is because they do not have the same sectional curvature: 0 for $\mathbb{R}^n$, strictly positive constant for Δ^m. Thus, I am now interested in the following questions.

- Let $\mathcal{X} \subseteq \mathbb{R}^n$ where $\mathcal{X}$ be an open or closed rectangle. How to endow $\mathcal{X}$ with a metric of fixed curvature?
  - We want to be able to easily measure distance in $\mathcal{X}$.
  - We want this distance to be meaningful in terms of dissimilarity (at least as meaningful as L_2 or L_∞ distances).
  - Can we use the FIM and interpret the induced distance as uncertainty, or semantic distance, or for verification (such as randomized smoothing)?
  - If yes, should we use categorical distributions (then how to interpret it?), or normal distributions (then how to deal with its negative curvature?), or something else?
- How to build an affine function and an activation function such that:

  - The sectional curvature is preserved. Or the sectional curvature is modified by a small, chosen amount.
  - Both the affine and the activation functions are gradient norm preserving (GNP).
- How to choose the loss function? See the papers about consistency of loss for adversarial robustness.
- What are the theoretical benefits (from statistical and/or geometrical perspectives) of the Fisher information metric (FIM)?
  - Should we randomly sample the prediction?
  - Can we trade determinism for robustness?

  It is time to read the Amari & Nagaoka book.

Example of framework:
Input = Normal distribution → LipNet → Output = Categorical distribution

1.1 The two Problems of model stability

Before stating the two Problems, I want to introduce a new terminology. I believe that any reference to "adversary" is misleading. "Adversarial attacks" are not about defending against attackers. It is not about cybersecurity. It is about a discrepancy between AI and human decision boundaries. Hence, I will now talk about **model stability**.

The two Problems of model stability are:

1. How to train models that "understand what they are doing", and how to verify that they understand?
2. How to choose the dissimilarity measure?

1.2 Definitions and facts

Let M and N be connected Riemannian manifolds.
The metric on M is h and its associated distance is d_M.
The metric on N is g and its associated distance is d_N.
Let $f : M \to N$ be a smooth map (f is C^1 almost everywhere).
For all $x \in M$ such that f is C^1 at x, we write $\nabla_x f$ the total derivative of f at x. For $x \in M$ and $\epsilon > 0$, we write $\mathcal{B}(x, \epsilon) = \{y \in M : d_M(x, y) < \epsilon\}$.

Definition 1.1. f is 1-Lipschitz (or Lipschitz for short) if for all $x, y \in M$, $d_N(f(x), f(y)) \leq d_M(x, y)$.

Definition 1.2. f is gradient norm preserving (GNP) if for any smooth function $L : N \to \mathbb{R}$ and almost any $x \in M$:

$$h_x(\text{grad}_x(L \circ f), \text{grad}_x(L \circ f)) = g_{f(x)}(\text{grad}_{f(x)} L, \text{grad}_{f(x)} L).$$

Definition 1.3. f is a local isometry[1] if for all $x \in M$, there is $\epsilon > 0$ such that for all $x_0, x_1 \in \mathcal{B}(x, \epsilon)$, $d_N(f(x_0), f(x_1)) = d_M(x_0, x_1)$.

Fact 1.4. f is a local isometry if and only if[2] $f^* g = h$.

Fact 1.5. If f is GNP, then f is Lipschitz.

Fact 1.6. f is GNP if and only if f is a local isometry.

[1] If M and N have different dimensions, then there is no local isometry as defined here. We need to replace $\mathcal{B}(x, \epsilon)$ by the intersection of $\mathcal{B}(x, \epsilon)$ with the leaf containing x and associated to the foliation $(\ker \nabla f)^\perp$. However, defining these objects is cumbersome so I skip these details for now.

[2] Once again, the correct statement if that $f^* g = h$ on $(\ker \nabla f)^\perp$ only.

1.3 Information geometry and Lipschitz neural networks

Here, I want to discuss the Problem 2 of subsection 1.1.

A caveat In order to define model stability, we need an *a priori* dissimilarity measure in the input space. This may be similar to *feature extraction*.
However, without knowing the task, we don't know if a similar variation across two different features should be considered as a similar "dissimilarity". For example, if you change an object in the foreground of an image, or an object in the background of an image (such that both modifications have the same "amplitude" a priori), one of the modification may change the ground-truth label and not the other, depending on the task. Thus, one of the modification is a candidate for adversarial perturbation but not the other. This is discussed in [1] and [2].

1.3.1 Some ideas

Let $\mathcal{X} \subseteq \mathbb{R}^d$. We may assume that $\mathcal{X}$ is bounded, or even compact.
Let $\mathcal{Y} = \{y_1, \dots, y_c\}$ a finite set of cardinal c.
We assume that there is a distribution $\mathbb{P}_{XY}(x, y) = p(y|x)q(x)$, where q is absolutely continuous w.r.t. the Lebesgue measure on $\mathbb{R}^d$.
We assume that for all $x \in \mathcal{X}$, $p(y|x) \in \mathcal{P}(\mathcal{Y})$.
We have a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$ sampled from $\mathbb{P}_{X,Y}$.

- Pre-training for feature extraction. The goal is to learn a "prior", "personal experience", "natural laws", "input space dissimilarity measure" etc.
  Learn an embedding layer before training on the task. The embedding layer f could satisfy:

  $$f = \arg\max \left(\min_{x_i, x_j \in \mathcal{D}} d(f(x_i), f(x_j)) - \max_{x_i, x_j \in \mathcal{D}} d(f(x_i), f(x_j)) \right).$$

  Then, we could learn a LipNet1 $l(z)$ and predict $y = \arg\max l(z)$ with $z = f(x)$.
- The quotient of the logit space with the vector full of one $\mathbf{1}$, denoted $\mathbb{R}^c/\mathbf{1}$, can be seen as the exponential representation of $\mathcal{P}(\mathcal{Y})$. The e-representation has statistical meaning.
- Maybe, we should discretize $\mathcal{X}$, and associate each similar $x \in \mathcal{X}$ with a discrete representative x_D. Then, each $p(y|x_D)$ is approximated by the empirical distribution of y for all x_i in $\mathcal{D}$ which are associated to x_D.
- Maybe, we should pullback the e-representation (logits) to the input space, and use this task-dependent dissimilarity measure. This is the idea of [3] or [4].
- Maybe, we should restrict the output space to a flat submanifold of $\mathcal{P}(\mathcal{Y})$, such that we can use the Euclidean distance in the input space along with a LipNet1.
- Maybe, we should see $p(y|x, w)$ as being parameterized by both $x \in \mathcal{X}$ and the weights $w \in \mathbb{R}^q$.
- How to deal with "global robustness"? We need to add a class $\perp$ for rejected prediction. A paper by Chen et al. 2023 argues that predicting $\perp$ should have a cost (a rejection cost). We could achieve global certified robustness while minimizing the $\perp$ decision area.

1.3.2 C-maximization

Here are some ideas/intuitions.
We want to represent our dataset as a finite set of functions (or random variables) $\mathcal{D} = \{F_1, \dots, F_n\}$ defined from $\mathcal{Y}$ to $\mathbb{R}$. The pairs (x_i, y_i) are implicitly represented by one (or several) F_i.
For each $x \in \mathcal{X}$, we are looking for a conditional distribution $p(y|x)$. The input x is represented by a (or several) function $F_0 : \mathcal{Y} \to \mathbb{R}$. We know that $p(y|x)$ belongs to:

$$M_{\mathcal{D}} = \{p \in \mathcal{P}(\mathcal{Y}) : \mathbb{E}_p[F_i] = 0, i = 0, \dots, n\}.$$

This is a mixture family that can be seen as a m-flat submanifold of $\mathcal{P}(\mathcal{Y})$. When we change the dataset $\mathcal{D}$ and the input point x, we obtain a different submanifold. The set of all possible such submanifolds is an (overparameterized) foliation of $\mathcal{P}(\mathcal{Y})$ (is it?).
Now, in order to make a prediction, we need to choose a p from $M_{\mathcal{D}}$, according to some maximization criteria. This might be linked to *extremum estimators* as well as *expectation-maximization algorithms*.
Consider the exponential family:

$$E_C = \left\{ q_\theta(y) = \exp\left(C(y) + \sum_{i=0}^{n} F_i(y)\theta^i - \psi(\theta) \right) \right\}. \tag{1}$$

When $\mathcal{D}$ and x are fixed, the family E_C is entirely characterized by the function C. E_C is a e-flat submanifold of $\mathcal{P}(\mathcal{Y})$. We obtain an (overparameterized) foliation of $\mathcal{P}(\mathcal{Y})$ when C ranges across all possible functions $\mathcal{Y} \to \mathbb{R}$.
Information geometry tells us that if E_C and $M_{\mathcal{D}}$ intersects, then they intersect in only one distribution p^* and this distribution verifies:

$$p^* = \arg\max_{p \in M_{\mathcal{D}}} \{ H(p) + \mathbb{E}_p[C] \},$$

where H is the entropy. In particular, when $C \equiv 0$, then p^* is the distribution that maximizes the entropy given the data. I suspect that a network trained with the cross-entropy loss (without any regularization) converges to p^* for $C \equiv 0$. In this case, I guess that (the parameters of) p^* is some kind of maximum likelihood estimator.
Maybe, we can reformulate all regularization methods by choosing another function C in the maximization problem. In the Bayesian setting, the choice of C might correspond to the choice of a prior.
Note that since $\mathcal{Y}$ is finite, the functions $C, F_0, \dots, F_n$ can be seen as vectors of $\mathbb{R}^d$. I am not sure of what happens if $n + 1 > c - 1$ the dimension of $\mathcal{P}(\mathcal{Y})$. I don't know how to link this framework with the neural network framework where the weights are shared across all input x, and where the number of weights is larger than n.

1.4 Miscellaneous

- Why is spectral normalization less good than orthonormalisation.
- Can we imagine a method to *globally constrain* a network such that the "Lipschitz constant of one layer is compensated by another" (such that the overall network is Lipschitz)? In other words, how to obtain a Lipschitz network without orthogonalizing each layer?
- Is the smoothed model obtained with randomized smoothing (before the argmax) a Lipschitz map?
- I should summarize the properties of Lipschitz networks. Why are they so promising?
- From less to more constrained: AllNet → Spectral normalization of affine layers → Orthonormalisation & ReLU (e.g., Parseval) → Orthonormalisation & GroupSort (GNP).

2 Lipschitz Neural Networks

2.1 Béthune et al.

Let AllNet be the class of "all" neural networks. Let LipNet1 be the class of 1-Lipschitz neural networks.

In [5], the authors investigates several properties of Lipschitz neural networks (they say that LipsNet1 networks received a lot of attention recently, see the other papers of this section): expressiveness (and the importance of the hyperparameters of the loss), accuracy-robustness trade-off, generalization.

LipNet1 networks can be used for certifiably robust classifiers as well as Wassertein distance estimation. LipNet1 networks have limited expressiveness for regression tasks, however *this is not true for classification.* Every AllNet network g is L-Lipschitz, thus $f = (1/L)g$ is in LipNet1. f has the same accuracy and also

the *same robustness*[3] *to adversarial attacks* as g. However, computing L is NP-hard.
The main thesis of this paper is that **LipNet1 networks are theoretically better grounded than AllNet networks for classification.**

2.1.1 Notations

The paper deals with binary classification.
The input space $\mathcal{X}$ is assumed to be compact. The observations are iid from $\mathbb{P}_{XY}$. The goal is to learn a classifier modeling the optimal Bayes classifier $\arg\max_{y\in\mathcal{Y}} \mathbb{P}_{Y|X}(y|x)$.
Let P be the input distribution of label +1, and Q be the input distirbution of label -1.
The paper focus on Euclidean norm $\|\cdot\|$ for vectors and spectral norm $\|\cdot\|_2$ for matrices (so no semantic attacks ...).
AllNet includes any feed-forward network of fixed depth (no recurrent mechanisms) using affine layers (including convolutions and batch normalization[4]) and Lipschitz activation functions. I guess it also includes residual connections.
LipNet1 is defined in [6] (next subsection).
The authors use the Deel.Lip library for the experiments. Their LipNet1 networks use:

- Orthogonal weight matrices. Orthogonalization is enforced using Spectral normalization and Björck algorithm.
- GroupSort2 activations (see [6]).

The binary cross-entropy (BCE) loss is defined as $\mathcal{L}_\tau(f(x), y) = -\log\sigma(y\tau f(x))$ with σ the logistic function. Other well-known losses are the Hinge loss, and the HKR.

Benefits of LipNet1 for robustness. Computing certificates for LipNet1 networks does not increase runtime contrary to methods based on bounding boxes or abstract interpretation (formal methods). No need for adversarial training that fails to produce guarantees, or for randomized smoothing which is "costly" (how much costly?). Sounds like the perfect method. But being able to efficiently verify the robustness doesn't imply that the network is actually robust. The certified robustness might be arbitrarily small. The question is: is it possible to train robust networks that are easy to verify?

It is possible to upper-bound the Lipschitz constant of any AllNet network (as the product of the Lipschitz constants of each layer). The authors claim that this bound is too high in practice to obtain meaningful certificates (this is in contradiction with Leino et al. [7]). Moreover, AllNet networks have usually very small robustness radius.

For any $f \in$ LipNet1, the robustness radius ϵ of binary classifier $\text{sign}\circ f$ at example x verifies $\epsilon \geq |f(x)|$.

2.1.2 Expressiveness

For any classifier c with closed pre-images $c^{-1}(\{y\})$, there exists a 1-Lipschitz function f such that $\text{sign}(f(x)) = c(x)$ on $\mathcal{X}$, and such that $\|\nabla_x f\| = 1$ almost everywhere.
P and Q are ϵ-separated if the distance between their supports exceeds $\epsilon > 0$.
If P and Q are ϵ-separated, there exists a network in LipNet1 with error 0. In other words, LipNet1 does not suffer from bias (in the bias-variance sense).
Lipschitz constraint is not a constraint on the shape of the boundary (which can be arbitrarily complex) but on the slope of the landscape of f.
For any $\tau > 0$, there exists a network f^τ in LipNet1 that minimizes the BCE over all LipNet1 networks. Moreover, LipNet1 networks do not suffer of vanishing gradient.
The minimizer of BCE is *not* necessarily a minimizer of the error. Yet, as $\tau \to \infty$, we get asymptotically closer to maximum empirical accuracy. At the same time, higher τ can lead to overfitting. **The reason for the poor accuracy of LipNet1 is the use of an implicit parameter** $\tau = 1$. This has nothing to do

[3]Thus, our method consisting in rescaling the output is doomed to fail (?) Or maybe, rescaling the output will change the gradients w.r.t. the parameters, and hence will change the model obtained after convergence. And, somehow, this new model is more robust? Why?

[4]Batch normalization is affine during inference but not during training.

with the hypothesis space LipNet1 itself.
Conversely, AllNet networks are equivalent to learning a LipNet1 network with $\tau \to \infty$. The Lipschitz constant L for AllNet plays the same role as τ for LipNet1. AllNet networks (without regularization or data augmentation) can always reach 100 % **train** accuracy without generalization guarantees.

2.1.3 Robustness

The accuracy-robustness trade-off doesn't exist in theory (?) But it does exist in practice. However, it must be reminded that this trade-off lives *in the shade of generalization* (?)

Let c be a binary classifier with decision boundary ∂. Let d be any distance function on d. The Signed Distance Function of c, denoted $SDF(c)$ is defined as $SDF(c)(x) = d(x, \partial)$ if $c(x) = +1$ and $SDF(c)(x) = -d(x, \partial)$ if $c(x) = -1$. We have $SDF(c) \in$ LipNet1.

One direction is to start from the set of maximally accurate classifier and find the most robust among them. Let b a Bayes classifier. The function $SDF(b)$ is the 1-Lipschitz function that provides the largest certificates among the classifiers of maximum accuracy. Moreover, those certificates are exactly equal to the distance of adversarial samples (certified robustness = true empirical robustness). This may improve the interpretability of models. Unfortunately, $SDF(b)$ cannot be explicitly constructed since the Bayes classifier b is unknown.

The other direction is to start from the set of maximally robust classifiers and find the most accurate among them. Here, we are interested in maximally *certifiably* robust classifier (not the true empirical robustness which may be higher).
Robustness is only evaluated on well classified examples. The certificate can be both interpreted as a form of "confidence" of the network, and as the minimal perturbations required to switch the class. Hence, high certificate for misclassified examples should be weighted negatively since *confidence in presence of errors is worse.* Thus, the authors introduce the Mean Certifiable Robustness (MCR). Let P be any class (P is equal to P or Q in the previous notations) and y be the label associated to P. The MCR is:

$$\mathcal{R}_{(P,y)}(f) = \mathbb{E}_{x\sim P}\left[\mathbb{1}\left\{yf(x) > 0\right\}|f(x)|\right] + \mathbb{E}_{x\sim P}\left[-\mathbb{1}\left\{yf(x) < 0\right\}|f(x)|\right] = \mathbb{E}_{x\sim P}\left[yf(x)\right].$$

The classifier with highest MCR is $f = y \times \infty$. We can train for the classifier with highest MCR using the loss $\mathcal{L}^W(f(x), y) = -yf(x)$. The classifier with highest MCR is the Wasserstein-1 distance between the distributions P (label +1) and Q (label -1). See the WGAN discriminators paper (Serrurier et al [8]). The optimum of the MCR is invariant by translation and dilatation.

Unfortunately, WGAN discriminators are weak classifiers. For every $\frac{1}{2} \geq \epsilon > 0$, there exists distributions P and Q *with disjoint supports* such that for any f maximizing the MCR, the error of the classifier sign $\circ f$ is larger than $\frac{1}{2} - \epsilon$. In other words, we can find disjoints distributions such that any MCR-optimal function classifies with error arbitrarily close to $\frac{1}{2}$.

We already stated that when using the BCE loss, the accuracy is maximized when $\tau \to \infty$. Conversely, the authors show that when $\tau \to 0$, the robustness is maximized (in the sense of the MCR). The trade-off can be seen as a Pareto front between accuracy and robustness.

2.1.4 Generalization

LipNet1 is a "Glivenko-Cantelli" class, i.e., they are consistent estimators (when using Lipschitz losses). As the size of the training set increases, the training loss becomes a proxy for the test loss. In other words, LipNet1 networks cannot overfit in the limit of very large sample sizes. More formally, the minimum (over Lipschitz functions) of the empirical loss converges to the minimum of the test loss when the sample size goes to infinity. AllNet networks does *not* sastisfy this property.
We already states that LipNet1 has no bias. Now, we know that the variance (i.e., generalization gap) can be made arbitrarily small by increasing the size of the training set. The number of examples required to close the generalization gap is dataset specific. Empirically, low values of the temperature τ require fewer examples. However, remember that low values of τ imply low accuracy (but high robustness).

If during training of an AllNet network, the empirical loss over the training set converges to 0, then *the Lipschitz constant of the network diverges to* ∞. There is at least one weight matrix whose operator norm

converges to ∞. Moreover, the predicted probabilities (after logistic activation) converge to either 0 or 1 (saturated probabilities).
This is important because according to some other works, Lipschitz constant and adversarial vulnerability are related (?) Furthermore, the saturation of the predicted probabilities means that they do not contain any useful information on the true confidence of the classifier.
In AllNet, the infimum of BCE is *not* a minimum. Even with parameters that achieve 100% accuracy, the BCE is not minimized, and the parameters norm will diverge to infinity in order to approach the infimum of the BCE (unless there is weight regularization).
The authors argue that the vanishing gradient issue, due to first-order methods performing poorly, has hidden the caveat of BCE minimization. This is also due to rounding errors in 32 bits floating point arithmetic. If one uses second-order methods in float64, the Lipschitz constant will diverge as expected, while this may not be the case with first-oder methods. Is it what is called "implicit regularization" of SGD?

The certificate $|f(x)|$ can be seen as *confidence*. Let us choose a margin $m > 0$. Define the classifier c_f^m that predicts +1 if $f(x) \geq m$, -1 if $f(x) \leq m$, and $\perp$ otherwise (meaning that f is not confident). This setting is suitable for **PAC learnability**: a theory that finds bounds on the number of train samples required to guarantee that the test error will be below some threshold $0 \leq e < \frac{1}{2}$ with probability at least $1 > \beta \geq 0$. This relies on Vapnik Chervonenkis (VC) dimension. The authors show that the hypothesis class of all c_f^m with f in LipNet1 has finite VC dimension (which can be upper-bounded and lower-bounded). Since I don't know anything about VC dimension and PAC learnability, I don't understand why this is a good thing ...
If the classes are ϵ-separable, then choosing $m = \epsilon$ guarantees that 100% accuracy is reachable. *Prior over the separability of the input space is turned into VC bounds over the space of hypothesis.* When $m = 0$, the VC dimension is infinite and the class is not PAC learnable anymore (whatever that means). In this case, the training error will not converge to the test error in general (whatever the size of the training set). For some reason, it is not in contradiction with the previous result on the consistency of LipNet1.
The VC bound provided by the authors is architecture independent. In practice, this means that the LipNet1 network can be chosen as big as we want without risking overfitting (as long as m is chosen appropriately, whatever that means ??).

2.1.5 Conclusions

Enforcing Lipschitz constraint has been studied for many layers: activations, affine layers, attention layers, recurrent units etc. See **Gradient Norm Preserving (GNP) networks** also called **orthogonal neural networks**. It seems that this is a stronger condition than LipNet1, and it avoids vanishing gradients. GNP networks are achieved using **orthogonal matrices**.
See the GroupSort activation, which is a special case of **Householder reflection** (?)
See **Orthogonal kernels** (not sure what it is).
See **optimization over the Stiefel manifold** (group of orthogonal matrices).
Generalization bounds for Lipschitz classifiers.
Links between *adversarial robustness*, *large margin classifiers*, and *optimization*.
The importance of loss in adversarial robustness.

We should pay attention to the loss rather than to the architecture. Cross-entropy is not necessarily the best choice. Consider margin-based losses (hinge loss, HKR loss).

We should separate the effects of architecture, generalization, and optimization.

What about stability of LipNet1 to random initialization, to hyperparameters?

2.2 Anil et al.

In [6], the authors introduce a new activation function called GroupSort. This activation can be used to build Gradient Norm Preserving networks without sacrificing expressive power too much. Such networks can achieve certified adversarial robustness, as well as tight estimates of Wasserstein distance.

There are two approaches to enforce Lipschitz constraints: **regularization** and **architectural con-**

straints. Regularization methods perform well in practice but do not provably enforce the Lipschitz constraint. Architectural constraints are provable but hurt the expressive power: norm-constrained ReLU networks are unable to approximate the absolute value!

2.2.1 Background

If a function is everywhere differentiable, its Lipschitz constant is bounded by the operator norm of its Jacobian.
Since 1-Lipschitz functions are closed under composition, to build a 1-Lipschitz network, it suffices to compose 1-Lipschitz affine transformations and activations.
For linear transformations, we need to ensure that $\|W\|_{=}\sup_{\|x\|_p=1}\|Wx\|_p \leq 1$. Let us define the mixed matrix norm:

$$\|W\|_{p,q} = \sup_{\|x\|_p=1} \|Wx\|_q.$$

Most common activations (ReLU, tanh, maxout) are 1-Lipschitz.
Applications: Wasserstein distance estimation, adversarial robustness, regularization, stabilizing GAN training ...

Gradient Norm Preservation. If a norm-constrained network f (using spectral norm) uses 1-Lipschitz **element-wise** monotonic activation functions, and if we want $\|\nabla f(x)\|_2 = 1$ almost everywhere, then f **must be linear**.
There is a tension between preserving gradient norm / decreasing Lipschitz constant, and nonlinear processing.
If a norm-constrained network f (using spectral norm) verifies $\|\nabla f(x)\|_2 = 1$ almost everywhere, then each weight matrix W can be replaced with a matrix $\tilde{W}$ whose singular values all equal 1 (without changing the computed function).
So, we are looking for **Lipschitz** networks, with **orthonormal weight matrices**, and **activations that preserve the gradient norm during backpropagation** (and which are expressive enough).

2.2.2 Methods

If we can lean any 1-Lipschitz function, then we can learn any K-Lipschitz function by scaling the output by K.
Now, we look for 1-Lipschitz network architectures (with respect to L_2 and L_∞ norms) by requiring *each* layer to be 1-Lipschitz.
ReLU is *not* gradient norm preserving. GroupSort separates the pre-activations into groups, then sorts each group, and outputs the combined group sorted vector. GroupSort is 1-Lipschitz and gradient norm preserving (its Jacobian is a permutation matrix which preserves any L_p norm). GroupSort is also homogeneous, and we can use any grouping size.
GroupSort2 (with grouping size = 2) is equivalent to absolute value in expressive power (using spectral norm). Using absolute value activation, a network can implement "folding" along any hyperplane. GroupSort can recover many common activations, including ReLU.

In order to enforce L_2 orthonormality (i.e., singular values equal to 1), the authors use Bjorck algorithm. Given a matrix, this algorithm finds the closest orthonormal matrix using the Taylor expansion of the polar decomposition (?)

$$A_{k+1} = \frac{1}{2} A_k \left(3I - A_k^T A_k\right).$$

The authors say that "this algorithm is fully differentiable and thus has a pullback operator for the Stiefel manifold allowing us to optimize over orthonormal matrices directly". I don't know what that means. They use 2 or 3 iterations, and increase to 15 iterations at the end of training.
It is possible to project the weight matrices on the L_2 ball either *after* each gradient descent step, and/or *during* forward pass (if the projection is differentiable, which is the case for Bjorck algorithm). If projected during forward pass, the learnable parameters are unconstrained during training. This is what is done by the authors.

The authors claim that Parseval algorithm [9] can be seen as a special case of Bjorck algorithm.
Spectral normalization is another method, but it only constrains the largest singular value to be smaller than 1, thus it is not gradient norm preserving.
There is another algorithm from the literature to project weight matrices to the L_∞ ball (see Condat 2016).

Provable adversarial robustness. Let K be the Lipschitz constant of some network w.r.t. L_∞ norm. let x be an input with class t and let y be the logits associated to x. The **margin** is:

$$\mathcal{M}(x) = \max\left\{0, y_t - \max_{i \neq t} y_i\right\}.$$

If $\mathcal{M}(x) > K\epsilon/2$, then the network is robust at x to all perturbations δ such that $\|\delta\|_\infty < \epsilon$.
In the following, the networks are trained with L_∞-norm constrained weights using *multi-class hinge loss*:

$$\mathcal{L}(y, t) = \sum_{i \neq t} \max\left\{0, \kappa - (y_t - y_i)\right\}.$$

The hyper-parameter κ controls the margin enforcement. It depends on the Lipschitz constant K and desired perturbation tolerance ϵ. I guess, we must choose $\kappa > K\epsilon/2$ (?)

Dynamical isometry. It turns out that my "isometry condition" has already been studied in other contexts (speed up training by isometric initialization) under the name of "dynamical isometry". Dynamical isometry is defined as having input-output Jacobian with singular values close to 1. A paper (Pennington et al., 2017) shows that **ReLU networks cannot achieve dynamical isometry!**

Universal Approximation of Lipschitz Functions. The main result of the paper is that L_∞-norm constrained networks with GroupSort activations can approximate any 1-Lipschitz function (with range in $\mathbb{R}$) in L_p distance. However, there is no such universal approximation result (not yet) for L_2-norm constrained networks, even if it seems true empirically (Maybe it was proven by Béthune et al. [5]).

2.2.3 Experiments

Lipschitz networks are able to learn "robust features" and have "interpretable gradients". The same is true for adversarially trained networks, but Lipschitz networks haven't seen any adversarial examples during training.

Comparison with Parseval. In Parseval networks [9], the authors use one step of gradient descent of the regularization $\frac{\beta}{2}\|W^T W - I\|_F^2$ after each backward pass.
One step of Bjorck algorithm is equivalent to the Parseval method with $\beta = 0.5$. However, Bjorck algorithm is applied during the forward pass. This is more costly but this guarantees to remain close to the Stiefel manifold during training.
The authord of [6] claim that the Parseval method with $\beta = 0.0003$ (commonly used value) ends up with weight matrices far from orthonormality, which is in contradiction with what is claim by the authors of Parseval networks [9] (if I remember well).

Questions. In Annex H.4, the authors say that "in order to scale the Lipschitz constant of the network, we introduce constant scaling layers in the network such that the product of the constant scale parameters is equal to the Lipschitz constant. As the activation functions are homogeneous, this is equivalent to scaling the output of the network." I wonder if changing the Lipschitz constant makes any difference. If the scaling is done at inference, the underlying classifier is exactly the same. But, scaling during training may impact the training process. This seems to be true when using cross-entropy (this is the *temperature* parameter). However, I am not sure for hinge loss: if the margin is scaled by the same factor, it seems that the gradient will be scaled by the same factor, but the direciton is the same (isn't it? I have to think more about this).

2.3 Serrurier et al.

[8]

2.4 Suggala et al.

In [2], the authors argue that the common definition of adversarial risk is wrong, because it doesn't take into account the possibility that the true label may change under a small perturbation.
The authors thus introduce a better definition of adversarial risk and show that the minimizer of any loss of the form "risk + $\lambda\times$ adversarial risk" is a Bayes optimal classifier[5], thus achieving both high accuracy and high adversarial robustness. This is result is true if using either the $0-1$ loss or the logistic loss[6].

Moreover, this result is still true using the common definition of adversarial risk, *but with a margin condition on a Bayes decision rule.* Intuitively, two different classes in $\mathcal{X}$ cannot be arbitrarily close of each other (under the chosen dissimilarity measure). However, it seems that standard datasets don't exhibit this margin condition. This is why an accuracy-robustness trade-off was observed *when using the common adversarial risk.*

Since standard training already converges to Bayes classifier, do we need adversarial training to achieve robustness? Yes for two reasons, which can be summarized by: the standard and adversarial risks are not calibrated.
First, the results mentioned above are true if the hypothesis class is the set of measurable functions (i.e., we have access to infinite data). If the hypothesis class is smaller, then the optimal classifier in this class for the standard risk may not be the same as the optimal classifier for the "risk + $\lambda\times$ adversarial risk".
Second, if there exist several Bayes optimal classifiers, then the one obtained when minimizing the standard risk may not be the one that minimizes "risk + $\lambda\times$ adversarial risk". Several Bayes classifiers may exist when the data "lies on a low-dimensional manifold" or "is separable" (whatever that means). Seems to be linked with the "boundary tilting" explanation.

I still need to see the links with the works on calibration of surrogate adversarial losses [10, 11].

3 Some conceptual experiments

I want to experiment myself with the setting introduced in Gilmer et al. [12], in particular using the "quadratic network". Is it possible to train a quadratic network such that all parameters are in the correct range? The goal is to address the Problem 1 of subsection 1.1.

- Can we train a 1-Lipschitz quadratic network? The quadratic activation is not Lipschitz. But if we restrict ourselves on a compact input space, the quadratic activation is Lipschitz.
- Can we associate 1-Lipschitz networks with randomized smoothing (doesn't seem to make a lot of sense)?
- Can we associate 1-Lipschitz networks with Fisher information? What if we sample the prediction? Can we improve the robustness at the expense of determinism?
- Study what VDP is doing on this simple task.

3.1 Results

The quadratic network is $f(x) = w\|Wx\|_2^2+b$, where $w, b \in \mathbb{R}$ and $W \in \mathbb{R}^{d_1\times d}$. I use $d = 500$ and $d_1 = 1000$ as in [12].
If s_i are the singular values of W, define $\alpha_i = -w\, s_i/b$. The decision boundary has zero generalization error if and only if $\alpha_i \in [1/R^2, 1]$ for all i. I choose $R = 1.3$.
It seems that one of the parameters w and b is redundant, since only the "radius" of the ellipsoid matter. This radius is defined by the ratio w/b. However, the *sign* of each individual parameter seems important,

[5] A Bayes optimal classifier is any classifier that minimizes the standard risk.

[6] With the assumption that $\mathbb{P}(y = 1|x)$ cannot be arbitrarily close to $1/2$. Note that we are working with binary classification here.

because it defines which side of the ellipsoid has label 1 or -1, i.e., $w > 0$ and $b < 0$ is different from $w < 0$ and $b > 0$. It seems that w and b must be of different sign for the decision boundary to be an ellipsoid.

Using a quadratic network (not Lipschitz). I was able to reproduce the results from [12].
At first, I trained the model with $\approx 200,000$ examples (generated randomly on the fly), but it wasn't enough for the model to reach an empirical error $< 10^{-4}$. The α (singular values of the linear layers, rescaled by the radius of the ellipsoid) looks like an exponential distribution, with most values close to 0.
Then, I trained the model with 50 millions examples. This time, it reaches empirical error $< 10^{-4}$. The distribution of α is monomodal and slightly right-skewed (looks like gamma distribution). Most values ($\approx 80\%$) are in the correct range $[1/R^2, 1]$, but there are still values outside the range.
Questions:

- Training on more data may produce a network with all α in the correct range?
- Is the Lipschitz constant of the linear layer diverging to infinity as claimed by [5]. To be tested, but it seems that the network is actually converging towards an isometry (which is the correct decision boundary). But the last affine transformation (w) may compensate for a diverging Lipschitz constant for W.
- Check the robustness to very small adversarial attacks.

Using a Lipschitz quadratic network. Here, both W and w must be orthogonal. Thus, this set-up is a little bit stupid because the decision boundary is necessarily a sphere ($s_i = 1$ for all i). We have $w = \pm 1$ and b controls the radius of the sphere. It is very easy for the model to reach perfect generalization error (all α in the correct range) since there is only one parameter b.
However, there is an unexpected issue. When the dimension is 1, the set of orthogonal matrices is $\{-1, 1\}$ which is not connected. Thus, if w is initialized to 1 then it cannot switch to -1 during training. Hence, the model reverses the labels and is *perfectly wrong*! The final affine transformation should not be orthogonal, since it is only controlling the radius of the sphere (and the label of each area).
After further testing, it seems that even by initializing w to -1, the model can sometimes switch to 1, achieving 0% accuracy. I cannot explain this phenomenon. If the final affine layer is no longer orthogonal, then the model always converges to 100% accuracy. From now on, we assume that w is not longer restricted to $\{-1, 1\}$.

- Check that the model achieves perfect generalization with few training examples.
- Check that the certified radius are really free of any adversarial examples.

3.2 Discussion

This experiment is kind of artificial because the Lipschitz quadratic network decision boundary is *a priori* a sphere. Since the dataset is spherical, Lipschitz network are by construction advantaged.
What if the dataset was two *ellipsoids* instead of two spheres, but the ellipsoid are spherical in every directions except a few. Then, the Lipschitz quadratic network cannot achieve perfect generalization while the vanilla quadratic network can. However, a 1-Lipschitz network (not quadratic) with several layers and GroupSort activation can achieve perfect generalization. But is it more efficient than a vanilla network, than a vanilla *quadratic* network?
Imagine we have our "ellipsoid dataset". Since the non-spherical directions are rare, there will be very few examples from these directions in the training set, maybe not at all. If you observe a few non-spherical examples from a large dataset of spherical examples, you may discard them as outliers, or noisy examples. Or maybe, you will notice that these non-spherical examples always happen along the same directions. How do you chose to modify your decision boundary to account for these abnormal examples, or to ignore them (or anything in-between)? This depends on your prior (Bayesian setting), your regularization (optimization setting), or the temperature of the BCE (when using 1-LipNet as explained in [5]). And more specifically to the parameter controlling the trade-off between the "bias" (prior/regularization) and the "variance" (overfitting).
But how to you chose such prior? How do you chose your notion of *simplicity*? I guess this is well-studied

problem. It seems to me that the idea is to have a bias toward distributions with maximum entropy. Given the data, the model should use the distribution that predicts the data with the maximum entropy.

- How to find the network with largest entropy given the data? Should the maximum entropy be computed across all distributions, or only across a specific subset?
- How to chose how much to discard some data? It seems to be the "robustness-accuracy trade-off". But [2] claim that there is no such trade-off (may be linked with calibration of loss function). How to reconcile this with the arguments of [5]?
- Have Lipschitz networks good properties with respect to the maximum entropy principle?

References

[1] F. Tramer, J. Behrmann, N. Carlini, N. Papernot, and J.-H. Jacobsen, "Fundamental Tradeoffs between Invariance and Sensitivity to Adversarial Perturbations," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[2] A. S. Suggala, A. Prasad, V. Nagarajan, and P. Ravikumar, "Revisiting Adversarial Risk," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, 2019.

[3] L. Grementieri and R. Fioresi, "Model-centric data manifold: The data through the eyes of the model," *SIAM Journal on Imaging Sciences*, vol. 15, no. 3, 2022.

[4] E. Tron, N. Couellan, and S. Puechmorel, "Canonical foliations of neural networks: Application to robustness," *arXiv:2203.00922 [cs, math, stat]*, Mar. 2022.

[5] L. Béthune, T. Boissin, M. Serrurier, F. Mamalet, C. Friedrich, and A. González-Sanz, "Pay attention to your loss: Understanding misconceptions about 1-Lipschitz neural networks," in *NeurIPS*, 2022.

[6] C. Anil, J. Lucas, and R. Grosse, "Sorting Out Lipschitz Function Approximation," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[7] K. Leino, Z. Wang, and M. Fredrikson, "Globally-Robust Neural Networks," 2021.

[8] M. Serrurier, F. Mamalet, A. González-Sanz, T. Boissin, J.-M. Loubes, and E. del Barrio, "Achieving robustness in classification using optimal transport with hinge regularization," 2021.

[9] M. Cissé, P. Bojanowski, E. Grave, Y. N. Dauphin, and N. Usunier, "Parseval Networks: Improving Robustness to Adversarial Examples," in *Proceedings of the 34th International Conference on Machine Learning*, pp. 854–863, PMLR, 2017.

[10] P. Awasthi, N. S. Frank, A. Mao, M. Mohri, and Y. Zhong, "Calibration and Consistency of Adversarial Surrogate Losses," *35th Conference on Neural Information Processing Systems*, 2021.

[11] P. Awasthi, N. S. Frank, and M. Mohri, "On the Existence of the Adversarial Bayes Classifier," *35th Conference on Neural Information Processing Systems*, 2021.

[12] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. J. Goodfellow, "Adversarial spheres," in *International Conference on Learning Representations*, 2018.